Trucs et Astuces

de Gérard Langlet

Un moyen simple et pédagogique - sans utiliser la primitive diadique (!) donnant directement, trop facilement dirons-nous, les combinaisons - d'imprimer les coefficients du binôme, consiste à utiliser le polynôme `x+1` (dont les coefficients sont 1 1 sous forme de vecteur) et à itérer l'élévation du polynôme en puissances successives, ce qui donne, en programmation structurée, grâce à la fonction générale MP de multiplication de polynômes[1], écrite en une ligne sans parenthèses :

```
    ∇ R←A MP B;D;K;V
[1] A←,A ◇ D←ΦK,0⌈¯1+K←⍴B←,B ◇ R←A∘.×B ◇ V←1-+\K⍴1 ◇ R←+/V⊖R⌿D⍴0
    ∇

      1 ◇ +W←V←1 1 ◇ '+W←W MP V'do 9
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

Bien sûr, en imprimant les mêmes résultats **modulo 2**, ou en remplaçant la fonction `MP par MPB` (multiplication de polynômes binaires, dans laquelle l'opération **multiplication** devient une *conjonction logique* et l'**addition** une *différence logique* donc un Ou Exclusif), on retrouve le triangle - fractal - de Sierpinski :

```
      ▽ R←A MPB B;b;D;V

    [1]A←,A ◇ D←⌽b,¯1+b←⍴B←,B ◇ R←A∘.∧,B ◇ V←1-+\b⍴1◇ R←≠/V⊖R⌿D⍴0 1 ◇
1+V←W←1 1 ◇ '+2⊥W←W MP V'do9

                                        ou bien:

          ◇ 1+V←W←1 1 ◇ '+W←W MPB V'do 9

    1

    1 1

    1 0 1

    1 1 1 1

    1 0 0 0 1

    1 1 0 0 1 1

    1 0 1 0 1 0 1

    1 1 1 1 1 1 1 1

    1 0 0 0 0 0 0 0 1

    1 1 0 0 0 0 0 0 1 1

    1 0 1 0 0 0 0 0 1 0 1
```

3. **POLYNOMES:**

Sous APL_SE, APL*PLUS II et III, il est possible de cliquer deux fois sur un nom inexistant dans la session et d'éditer par défaut cet objet qui devient initialement un vecteur de caractères. Même si, sous l'éditeur, on crée par exemple un objet de 2 lignes, l'objet en question ne changera pas de type (en l'absence de la frappe d'une commande de type <Ctrl> A).

Soit donc un objet `P` vecteur (avec un retour du chariot comme séparateur de ligne) qui s'imprime sur 2 lignes comme :

```
      P
 3
x -1
```

Un non APListe reconnaîtra sans grande difficulté un polynôme dans le contenu de P.

La fonction POL va convertir un polynôme à une variable, écrit de manière conventionnelle, en vecteur de coefficients par puissances décroissantes de ladite variable (x) de sorte que l'on puisse effectuer aisément des traitements en APL :

Bien entendu, P pourrait contenir des espaces ou des parenthèses, et des monômes en n'importe quel ordre.

```
      POL P
1 0 0 ¯1
```

Bien entendu, il existe dans la zone une fonction inverse LOP qui restitue P sous forme de caractères, exprimé en puissances décroissantes, à partir d'un vecteur de coefficients numériques :

```
      LOP 1 0 0 ¯1
 3
x -1

      LOP 6 21 ¯14 0 ¯3 21 ¯8 ¯1 ¯1
 8  7  6 4  3  2
6x +21x -14x -3x +21x -8x -x-1
```

Si la variable Q contient par exemple (les exposants ne sont pas obligatoires) :

```
x-1
```

on obtiendra un résultat sous forme polynomiale en écrivant :

```
   P DIP Q
 2
x +x+1
```

ce qui montre au passage que la limite du quotient polynomial, pour x tendant vers 1, vaut 3.

La fonction DIP fait le même travail que la fonction DP, mais, en plus, rend son résultat sous forme de polynôme en utilisant LOP[2] pour l'argument de sortie R.

```
    ∇ R←A DP B;a;J
[1]  R←0⍴A←POL A ⋄ J←+/∧\0=A ⋄ a←⍴A←J↓A ⋄ J←+/∧\0=B←POL B ⋄ B←J↓B
[2]  'J←1↑A⋄R←R,J←J÷1↑B⋄a←⍴A←1↓A-a↑B×J'do 1+a-⍴B ⋄ 'mal⋄A'do 0∨.≠A
    ∇
```

(La fonction LOP est beaucoup plus compliquée - surtout si on essaie de la programmer vectoriellement - que DP ou MP - voir le Courrier des Lecteurs pour le listage de cette dernière - car la notation mathématique conventionnelle, que l'on apprend à l'école regorge d'exceptions et d'ellipses; c'est pourquoi K. Iverson imagina... APL.)

La fonction mal émet un son aigu si le reste de la division polynomiale n'est pas identiquement nul. (Le reste se trouve également affiché dans ce cas).

```
      ▽ mal
[1]   ⎕SOUND 2000 500
      ▽
```

On n'en est pas encore au véritable calcul formel[3], capable de développer des identités remarquables, mais c'est un début...

4. **On peut faire bien des choses avec APL_SE!**

La Zone COCSE.AWS

Cette petite zone de travail (une `vingtaine` de K) est en fait un puissant simulateur d'APL étendu à diverses fonctionnalisés utiles pour des démonstrations, en général, et en algèbre binaire en particulier.

Elle contient les fonctions indispensables (tirées de la zone plus importante COCRAY utilisable en `APLPLUS`, qui en contient plus de 600 mais ne peut se charger sous APL_SE gratuit et plus limité en place-mémoire).

On passe en « session spéciale » grâce à l'appel de la fonction `INTERP` (fonction niladique) dont on peut sortir par la flèche droite (« … »).

En session sous l'interprète `INTERP` , la ligne d'exécution vire au rouge

Règles

1°) Une fois le simulateur activé, toute séquence d'instructions présente sur la même ligne d'exécution

- à condition que les expressions soient séparées les unes des autres par 5 espaces - est considérée, sans aucun séparateur (tel que le diamant), comme s'éxécutant *en parallèle*. Bien entendu, sur une machine séquentielle, les exécutions ont lieu l'une après l'autre et la modification d'un résultat dans l'une des expressions peut entraîner des résultats imprévus; en général, ce n'est pas ce que l'on cherche à faire dans une démonstration. Les résultats des expressions peuvent être de taille différente et de type différent. Mais le résultat qui va s'afficher sera en fait un Vecteur de Résultats, limité pour l'affichage à la largeur de l'écran soit 80 caractères (voir ci-dessous le point 2°).

2°) Tout objet APL (numérique puisque nous sommes en fait sous `APLPLUS`) ne contenant que des 0 ou des 1, donc interprétable en binaire, s'affiche en pixels, les 1 étant remplacés par des petits pavés (jaunes sur l'écran, noirs sur une imprimante) et les 0 étant remplacés par un fond continu (violet sur l'écran, blanc sur l'imprimante).

3°) Lorsque le résultat est un vecteur d'objets, ce résultat prend, en nombre de lignes, la taille du plus grand. On ne peut pas afficher d'objets de plus de 24 lignes (limitation pour les démonstrations). Les objets sont séparés par une barre verticale semi-graphique, laquelle apparaît sur l'écran plutôt comme une double barre verticale, négatif de la précédente, à cause du choix des couleurs. Il n'apparaît qu'un seul vecteur de résultats même si plusieurs expressions différentes en sont à l'origine; ainsi, lorsqu'une première expression a pour résultat une hypermatrice, ce dernier résultat peut devenir un vecteur d'objets matrices en nombre égal au nombre de plans de l'hypermatrice, et si une seconde expression est exécutée à droite de la même ligne, ses résultats - scalaire ou vecteur d'objets - viendront se concaténer à droite du vecteur d'objets du résultats final.

4°) Sous cet interprète spécial, certains caractères symboliques n'ont pas la signification usuelle de l'APL.

Le double astérisque

Ainsi, le double astérisque diadique ** de `M⋆⋆N` va prendre la signification « élévation de la matrice M, supposée binaire et carrée, à la puissance N en algèbre entière modulo 2 ».

De la même façon, `**M` (forme monadique) est la puissance du groupe cyclique de la matrice `M` si celle-ci est binaire[4], carrée et inversible modulo 2. Si la matrice n'est pas inversible, la réponse est 0. (Le signe de `**M` est toujours le déterminant, calculé modulo 2, d'une matrice carrée binaire). Lorsque `M` est inversible, le résultat de `M⋆⋆ ⋆⋆M` est toujours, par définition, une matrice-unité.

Le double point horizontal

Le symbole « .. » (double point horizontal) considéré comme un symbole unique, correspond, dans son acception monadique, à l'inversion modulo 2 généralisée de matrice carrée binaire, et dans son acception diadique, au produit matriciel modulo 2 de matrices binaires conformes. : une expression telle que `..M` va donc délivrer un résultat que l'on obtiendrait - si `M` est inversible - aussi par `1=|ŽM` à condition que `M` ne dépasse pas un certain rang[5]. L'expression `M.. ..M` produit une matrice-unité si `M` est une matrice inversible. Mais `..M` produit toujours un résultat, que `M` soit inversible ou non ! Il s'agit, dans le cas d'une matrice non inversible, d'une inverse **généralisée**.

Dans une matrice non inversible, le déterminant est nul; les lignes (ou les colonnes) ne constituent pas une base vectorielle de N vecteurs (si N est le rang), mais il existe en général - sauf pour une matrice identiquement nulle - une sous-base de n vecteurs indépendants, avec n<N. L'inverse généralisée obtenue ici - c'est une question de choix - est celle qui a ses n vecteurs vraiment indépendants dans les premières colonnes, les colonnes finales étant complétées par les colonnes finales d'une matrice-unité de sorte que l'ensemble des N vecteurs forme à nouveau une base : en conséquence, toute inverse généralisée de matrice non inversible est inversible. En inversant à nouveau cette dernière matrice, on va reconnaître les premières colonnes d'une matrice initiale non inversible (parfois décalées à gauche si ladite matrice initiale possédait des colonnes nulles ou bien des colonnes identiques à des colonnes précédentes). L'inverse généralisée d'une matrice identiquement nulle est alors une matrice-unité de même rang.

On pourra vérifier les identités suivantes (le symbole non APL ⍝ va, dans ce texte, signaler les identités) :

`M⍟⍟¯1+k×**M ⍝ ..M` si M est inversible

`M⍟⍟1+k×**M ⍝ M` si M est inversible

`k` étant un entier quelconque (négatif, nul ou positif) bien sûr représentable en machine[6].

Ce n'est pas difficile : il suffit de juxtaposer, sous l'interprète `INTERP`, les expressions à comparer, séparées par 5 espaces, et de vérifier *de visu*[7] si l'on obtient bien le même résultat.

Autres fonctions utiles

La conjonction ET

Même hors session INTERP, on peut se servir de la conjonction ET (ici non booléenne) pour former, à partir de A et de B, deux objets quelconques d'APL-ISO, une structure correspondant à un vecteurs d'objets dont les composantes seront A et B. D'ailleurs, l'expression monadique ET A forme un vecteur d'objet à une seule composante, laquelle est A. Par extension (et contrairement à ce que feraient les APL étendus), l'expression A ET B ET C engendre un vecteur de trois composantes A, B et C et non pas une structure à plusieurs niveaux, ce qui est beaucoup plus simple... Et ainsi de suite[8]. D'ailleurs le véritable fonction de la *conjonction* ET dans la langue française consiste à créer des vecteurs de concepts et non pas des structures hiérarchiques; si on veut parler de linguistique[9], alors respectons-la.

Initialement, l'intérêt était surtout la compression d'information, parce qu'APLPLUS code les booléens en 2 octets, ce qui représente un gâchis formidable bien que relativement efficace en performances.

Supposons que A, B et C soient des matrices carrées binaires, respectivement de rang 16, 15 et 16. On aura 256+225+256 bits soit en tout 737 bits, codés sur 1474 octets dans la zone (du moins pour les seuls contenus des variables). Si on écrit simplement R,,A ET B ET C , la dimension de R atteindra seulement 134 octets - soit un gain d'un facteur 11. Et plus les objets sont gros, plus le gain est grand; cela vaut la peine d'engranger dans 200 k ce que l'on devrait mettre dans plus de 2 Mégaoctets de manière conventionnelle.

La zone COCSE contient un tel objet (une variable) dénommée ROBEE. Lorsqu'on est sous INTERP, l'expression ½ROBEE répond 3, car ROBEE possède bien 3 composantes, lesquelles sont les célèbres ROBY, MAYA et GROBY, artistes bien connus dans ces chroniques, en fait comme par hasard des matrices binaires carrées de rang respectivement 16, 15 et 16.

Dans la même zone COCSE, la variable MAYA existe aussi sous forme de matrice de bits APL-ISO; par contre, les objets ROBY et GROBY ne sont pas des variables mais des *fonctions* niladiques avec argument de sortie, dont le résultat est la matrice respectivement souhaitée, mais *extraite* de ROBEE:

```
      ' R←ROBY
[1]  R←dans ROBEE
      '
```

```
      ' R←GROBY
[1]  R←dans FIN ROBEE
      '
```

La sous-fonction `dans` (en anglais « `in` ») a une syntaxe diadique telle que l'expression `1 dans ROBEE` fournit la première composante de `ROBEE`. Mais le 1 est facultatif en syntaxe monadique, d'où son omission. La fonction `FIN` (version monadique) considère l'objet à partir de sa dernière composante, donc le retourne de l'intérieur. `GROBY` est bien la troisième et dernière composante de `ROBEE`.

L'intérêt du simulateur `INTERP` apparaît rapidement lorsqu'on souhaite faire des démonstrations sous APL*PLUS ou APL_SE car on pourra écrire directement :

```
ROBEE
```

et voir les trois matrices d'un seul coup d'œil en pixels, puis reprendre la même instruction (grâce à la gestion d'écran) et rajouter à gauche `..` de manière à voir les trois matrices inverses, puis recommencer en rajoutant encore à gauche `..` de manière à inverser les trois matrices une seconde fois. Sachant que `MAYA et GROBY` sont inversibles mais non pas `ROBY`, on va reproduire la moitié gauche de ce dernier personnage, tandis que les matrices `MAYA et GROBY` seront restaurées intégralement.

La fonction HASB

· L'instruction `HASB N` met dans la variable B un vecteur binaire de dimension N bits. Ceci permet d'expérimenter les fonctionnalités vectorielles décrites dans cette zone.

La fonction HASMI

· L'instruction `HASMI N` met dans la variable `M` une matrice binaire de rang N tirée au hasard et garantie inversible. Ceci permet d'expérimenter les fonctionnalités matricielles décrites dans cette zone.

Ces deux instructions n'ont pas de résultat

Lorsqu'une instruction est une affectation de variable, le contenu de cette variable va, au contraire, s'afficher sous l'interprète `INTERP`.

De même, sous l'interprète `INTERP`, le résultat de l'expression précédemment exécutée se trouve dans une pseudo-variable, toujours accessible dans l'instruction suivante grâce au symbole « zilde » *Ø*, accessible au clavier APL par la combinaison de touches <Alt> $. (Cette décision un peu arbitraire mais bien pratique provient du fait qu'on n'a pas besoin de vecteur numérique vide dans les démonstrations...).

Module FORTRAN

Le calcul de la puissance du groupe cyclique d'une matrice binaire inversible fait appel à un module compilé écrit en FORTRAN Microsoft 8 bits (se rappeler qu'APL*PLUS est une simple application DOS). Lorsqu'une matrice `M`, argument droit de ** est donnée, un fichier appelé `MATR.TXT`, tout à fait éditable, sert d'entrée au module FORTRAN[10]. Le module s'appelle par une simple commande programmable ⍞CMD'RACRF51' car, bien sûr, il n'existe pas de ⎕NA en APL_SE. Le résultat est mis dans le fichier `MATR.RES` lequel est analysé dans la session pour produire la réponse. Comme ce FORTRAN ne peut coder des entiers super-longs (et qu'il importe de définir des dimensions maximales), le logiciel ne peut traiter de matrices au-delà du rang 32, ce qui est largement suffisant pour des démonstrations. La puissance P du groupe cyclique d'une matrice binaire inversible de ***rang*** N ne peut pas dépasser 2^N. Pour N égal à 16, P vaut 65535 et sera trouvé en un temps raisonnable : le module FORTRAN est au moins 20 fois plus rapide, lorsque P est suffisamment élevé, que son équivalent APL - à cause des boucles que l'on n'a pas pu éviter jusqu'alors[11].

Attention : pour une matrice **non** inversible, le logiciel FORTRAN ne calcule pas d'abord le déterminant modulo 2 (évidemment nul); ce n'est qu'après dépassement de la limite éventuelle de P imposée par le rang N que le résultat est remis à 0; c'est pourquoi le calcul de `**ROBY` dont le résultat est nul sera plus lent que celui de `**GROBY` dont le résultat n'est pas nul. Mais on peut néanmoins opérer vectoriellement sur `ROBEE`, vecteur de 3 matrices binaires dont deux sont inversibles modulo 2; le résultat de `**ROBEE` sera alors un vecteur de 3 entiers, en l'occurrence 0 12282 6132 obtenu en une trentaine de secondes sur un IBM Thinkpad 340CSE, (donc un simple 486 portable - sans coprocesseur - à 25 MHz).

Sous `INTERP`, on pourra expérimenter l'instruction : `ROBEE⋆⋆ ⋆⋆ROBEE` dont le résultat est un vecteur de trois matrices-unité, respectivement de rang 16, 15 et 16. (Le simulateur est même capable de comprendre une instruction telle que `(**ROBEE)⋆⋆ROBEE` donc avec des arguments commutés[12].

Toutes les modifications potentielles de la syntaxe APL conventionnelle font appel à des symboliques simples; lorsqu'il s'agit d'algèbre matricielle binaire (modulo 2), les symboles sont doubles : deux points, deux astérisques ...etc et font toujours partie des caractères ASCII que l'on peut frapper sur n'importe quel clavier. En un quart de siècle d'APL, je n'ai jamais vu ** figurer dans aucune instruction. Quant au double point, il n'interfère pas le moins du monde avec la syntaxe APL dans laquelle son emploi était interdit. Le symbole : parfaitement récupérable si on n'utilise jamais d'étiquettes mais de la programmation structurée, peut servir[13] pour obtenir la transformée *cognitive* d'un objet d'information quelconque (vecteur, matrice,

hyper-matrice ...etc). La syntaxe est monadique. De même, le symbole .: monadique (deux caractères) définit la transformée *hélicoïdale* d'un objet d'information quelconque (vecteur, matrice, hyper-matrice ...etc). Le *pariton*[14] (matrice des intégrales successives) d'une séquence binaire quelconque - un vecteur de bits - s'obtient par le symbole monadique Œ: (deux caractères). Le *géniton* - cas particulier de pariton pour une *séquence primordiale* c'est-à-dire un vecteur commençant par 1 suivi uniquement de zéros, s'obtient par le double symbole __ (deux blancs soulignés successifs, de sorte que `__N` équivaut à `Œ:N†1`).

Autres fonctionnalités

Sous `INTERP`, un point d'exclamation tout seul (niladique) reproduit sur l'écran l'ensemble des instructions précédemment exécutées.

Sous `INTERP`, un point d'interrogation tout seul (niladique) reproduit sur l'écran l'ensemble des substitutions et équivalences possibles - non APL à proprement parler - entre symboles ou groupes de symboles et les fonctions de la zone. (Attention : seules les fonctions décrites ici sont présentes dans la zone COCSE; les autres se trouvent dans la zone expérimentale COCRAY laquelle n'est pas pour l'instant disponible).

5. **Le chapeau mexicain ou Graphiques 3D:**

```
      ⍴CHAPEAU2

109

      CHAPEAU2

¯8.1 8.1 0.8

¯8.1 8.1 0.8

           2 2 1÷2 2 2 1÷2

10×(SIN(((X)+(Y))))÷(((X)+(Y)))
```

La variable `CHAPEAU2` est un vecteur de caractères éditable tel quel par l'éditeur APL. Les quatre lignes se lisent comme suit : *pour X variant de -8.1 à 8.1 par pas de 0.8 et pour Y variant de -8.1 à 8.1 par pas de 0.8, calculer la formule ...* etc, sachant qu'un potache non-APListe devrait comprendre qu'il s'agit de la formule du *chapeau mexicain*, utilisée non seulement dans les livres de maths mais aussi en physique à

propos de l'électrodynamique quantique. Pour le profane, on n'utilise que les symboles APL, aussi présents sur toute bonne calculette, à savoir + - × ÷, les exponentielles étant exprimées sous forme... d'exposants, et les fonctions habituelles de la trigonométrie par les abréviations fonctionnelles `SIN` , `COS` ... etc.

La syntaxe d'emploi est fort simple. Une instruction en langue naturelle telle que :

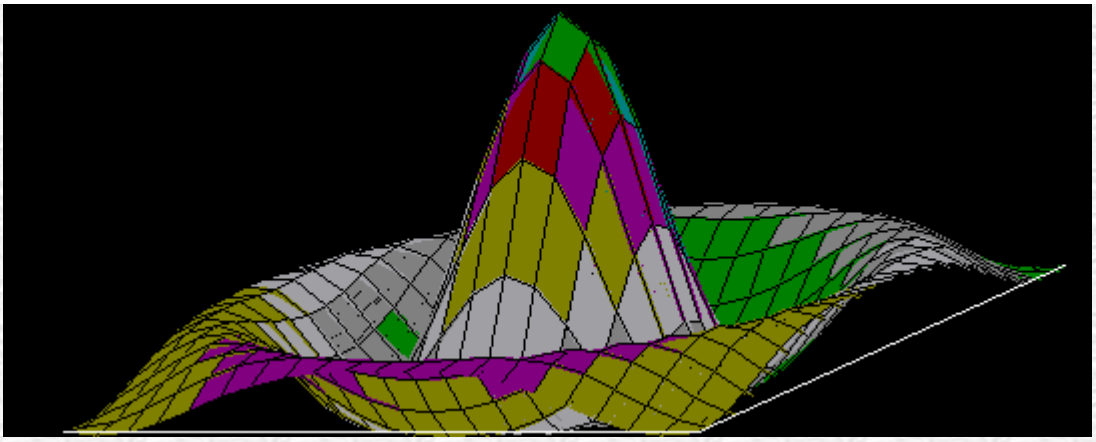
`COURBE DE CHAPEAU2` va provoquer le calcul d'un tableau de 22 lignes et de 22 colonnes (*Z* en fonction de *X* et de *Y*) avec apparition du message suivant :

```
dimension du Tableau : 22 22


j'ai compris :

10 mul (SIN(((X⋆2) plus (Y⋆2))⋆1 div 2)) div (((X⋆2) plus (Y⋆2))⋆1 div 2 )
```

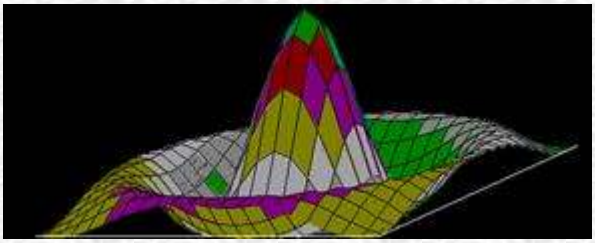
(ceci n'étant que la transcription en APL de ce qui s'exécute), puis du graphique (de type graphique `G` dans le cas présent).

Rappel : le graphique, tracé an APL*PLUS ou APL_SE est isotrope en fenêtre DOS. Après tracé, on passe en fenêtre « Windows » (non isotrope : les horizontales sont légèrement plus longues que les verticales, ce qui, en mathématiques n'a pas beaucoup d'importance tant qu'il s'agit de graphes). Le graphique en couleurs est alors transféré - sans intervention d'aucun logiciel - directement dans un fichier « Word » bien qu'il soit en « bitmap » couleurs, comme un vulgaire caractère (par un couper-coller) :

La session APL en fenêtre « Windows » possède un petit carré gris à gauche de l'en-tête de la fenêtre. Il suffit de cliquer à la souris sur ce carré pour faire apparaître le menu « Edition », lequel a deux sous-options actives « Marque » et « Copie Entree ». Il suffit de relâcher le bouton de la souris sur « Marque » pour que l'on puisse au curseur (toujours à l'aide de la souris), marquer - par un rectangle jaune - la partie à copier. Puis, par choix à la souris, de la sous-option « Copie Entree » du même menu « Edition », la partie sélectionnée de l'écran graphique est sélectionnée - ceci se manifeste par un passage bref en video inverse; il ne reste plus qu'à changer d'application pour coller au bon endroit, par exemple sous « Word6 » ledit graphique (commande <Ctrl> V). Si le graphique est inséré dans un tableau plus petit que lui, sa taille va s'ajuster automatiquement.



Voici ce que cela donne en plus petit :



Les couleurs (arbitraires) dépendent de Z fonction de X et de Y. Les facettes rectangulaires sont triées en fonction de la profondeur, ce qui donne l'ordre du dessin et permet le recouvrement des parties cachées, selon l'algorithme du *peintre*.

```
     ∇ ⍵ COURBE ⍺;⍺⍺;PARAMETRES
[1]    PARAMETRES←⍺ ⋄ ⍺⍺←0 ⋄ '⍺⍺←⍵' if 0<⎕NC '⍵' ⋄ XYZ2 ⍺⍺
     ∇


     ∇ ⍺←DE ⍺ fonction neutre
     ∇


     ∇ XYZ2 I;X;Y;Z
[1]    EFFACE ⋄ Z←CREEXY ⋄ '' ⋄ DESSINE I
```

```
      ∇


      ∇ EFFACE fonction d∇effacement de l∇écran

   [1]   ⎕TCFF

      ∇


      ∇ R←CREEXY;b;D;d;E;L;M;N;P;T;V;Z;⎕IO;⎕ELX

   [1]  ⍝ ANALYSE LES 2 PREMIERES LIGNES DE ''PARAMETRES'':  Xmin,
Xmax, pas

   [2]  ⍝ Ymin, Ymax, pas. PUIS CREE LES VECTEURS X ET Y
 CORRESPONDANTS, PUIS

   [3]  ⍝ RESULTAT: LE TABLEAU R=F(X,Y) DE L'EXPRESSION DE LA  3ème
LIGNE;

   [4]  ⍝ X ou Y accolé à un nombre est lu comme X ou Y à la
 puissance CE NOMBRE;

   [5]  ⍝           2

   [6]  ⍝ Exemple: X ou (X2) est X au carré. METTRE  EXPLICITEMENT
DES PARENTHESES

   [7]   ⎕IO←1 ⋄ 'R←qR qBOX R' if 1=⍴⍴R←PARAMETRES ⋄ L←⍴X←,ev  R[1;]

   [8]   L←3∨.≠L,⍴Y←ev R[2;] ⋄ b←' ' ⋄ 'R←R[3;]' if~L←4≤1↑⍴R

   [9]   'D←b≠E←R[3;]⋄R←R[4;]⋄R[D/⍳⍴R]←D/E' if L ⋄ V←X

   [10]  ⍎E←'M←1↑1↓V⋄D←M-L←1↑V⋄V←M⍳L+P×V←0,⍳1⌈⌈D÷P←1↑2↓V' ⋄ X←V
    ⋄ V←Y ⋄ ⍎E ⋄ Y←V

   [11]  L←R∈'XY' ⋄ L←1+L∧1⌽R∈N←qD,'¯.' ⋄ R←L/R

   [12]  R←R qSSN       'θXXθX★θYYθY★',,4⌽'θ',')','★',N,'θ',')',[1.5]N

   [13]  d←'EQUATION MAL ECRITE ou PARAMETRES FOIREUX' ⋄
       ⎕ELX←'d⋄→'

   [14]  ⎕ELX←'⎕DM'

   [15]  E←R qSSN 'θ+θ plus θ-θ minus θ×θ mul θ÷θ div '

   [16]  METS 'J''ai compris:' EN VERT EN 11 0 ⋄ METS E EN  JAUNE EN
12 0 ⋄ R←⍎E

   [17] ⍝ fonctionne sur carte HERCULES ou EGA
```

```
    [18] ⍝ Essayer: COURBE CHAPEAU   (ou CHAPEAU2)

         ∇

         ∇ R←ev B Cette fonction remplace - par ¯

    [1]    R←qEA B qSSN '/-/¯'

         ∇

         ∇ ⍞←qEA ⍙;⎕ELX Cette fonction est une sorte d'Exécute si tu
peux

    [1]    ⎕ELX←'⍞←⍬' ◇ ⍞←⍙ ⋄

         ∇

         ∇ ⍙R←⍙A if ⍙B;⍙E;⍙L;⎕ELX

    [1]    ⎕ELX←'⎕DM' ◇ ⍙L←⍙B>0 ◇ ⍙R←⍙L/⍙A ◇ ⍙E←0≠1↑0⍴⍙R ◇ ⍙⍙E/⍙R ◇
⍙⍙E/'⍙R←0 0⍴⍙R' ◇ ⍙L←0<⍙B←⍙B-1 ◇ →⍙L/1

         ∇

         ∇ METS B Mettre sur l'écran le message B

    [1]    cMSG MSG B

         ∇

         ∇ C MSG B

    [1]    ⍬ ◇ W←pMSG,¯2↑ 0 1 ,⎕PW⌊⍴B←⍕B ◇ W ⎕WPUT B ◇ W ⎕WPUT C

         ∇

         ∇ DESSINE
I;A;B;C;c;D;d;E;e;F;G;H;i;J;j;K;k;L;l;M;m;N;n;O;P;p;Q;R;S;T;t;U;V;v;W;
⎕IO;⎕ELX
```

```
    [1]   bon ◇ err ◇ ⎕IO←1 ◇ t←-1↑1↓⎕AI ◇ ⍝ Résultat: temps consommé
en secondes

    [2]   F←800+100×her ◇ d←100-50×her ◇ ERRG←'OK'

    [3]   K←'Dimension du Tableau ',⍕D←⍴Z ◇ METS K EN MAGENTA EN 4 0

    [4]   Z←-,Z ◇ K←0,⍳¯2+⍴X ◇ N←⍴Y ◇ J←×/D ◇ Y←J⍴-Y ◇ bon ◇ T←Z,
[1.5]Y

    [5]   X←J⍴D[2]/X ◇ Eff 'Z Y' ◇ T←X,T ◇ Eff 'X' ◇ L←O,N+⌽O←2↑E←4⍴⍳3

    [6]   L←L∘.+0,⍳¯2+N ◇ L←L∘.+K×N ◇ A←-○÷6 ◇ U←T+.×1⌽1⊖⍉ro⍙ ◇ U[;3]
←T[;3]

    [7]   Eff 'T' ◇ A←○¯0.1 ◇ U←U+.×2⌽2⊖⍉ro⍙

    [8]   T←⍳≠U ◇ D←0 ◇ 'U[;D]←U[;D]-T[D←D+1]' if 3 ◇ Eff 'T' ◇ U[;1]
←U[;1]×3÷4

    [9]   D←F÷⌈/⌈/U ◇ U←D×U ◇ U←⌊U ◇ U←d+U ◇ D←4,×/1↓⍴L ◇ L←D⍴L ◇ W← 3
2

    [10]  V←U[L;3] ◇ U← 0 ¯1 ↓U ◇ 'L←L[;⍋+≠V]' if I=1 ◇ L←⍒L ◇ j←÷3 ◇
⍝I←0≠1↑,I

    [11]  c← 1 3 , 2 2 ⍴⌽W ◇ P←'gfin◇restec' ◇ i←'memec◇initg 1' ◇ A←0
◇ Q← 1 4 2

    [12]  'p←''1⎕GLINE T◇⎕AV[177]⎕GPAINT H◇⎕AV[220]⎕GPAINT H◇2⎕GLINE
T''' if her

    [13]  'p←''1⎕GLINE T◇0⎕GPAINT H◇N⎕GPAINT H◇N⎕GLINE l◇0⎕GLINE t'''
if ega

    [14]  M←⌈/V←+/U[L;2] ◇ H←⍳K←8 ◇ H←H÷K ◇ D←⌽1,⍴V ◇ C←D⍴1+
+/V∘.>C+⍳+H×M-C←⍳/V

    [15]  S←'T←Q⍴U[A←A+1;;]◇N←1+C[A;1]◇H←M[A;]◇l←T[;2 3;]◇t←T[;2 1
3;]◇p if mg'

    [16]  C←1+C-7×8=C ◇ G←'A←A+1◇N←C[A;1]◇A,,U[L[A;K];]◇N,¯1,M[A;]' ◇
'G←S' if mg

    [17]  'C[;]←1' if her∧I<0 ◇ B←'A<D' ◇ ⎕ELX←'ERRG←⎕DM' ◇ 'VXYZ2' if
I=1

    [18]  'i if mg◇L←⊖L,L[;1]◇C⎕GLINE U[L;]◇15⎕GLINE 1 3 2⍴U[1,J-2↑N-
1;]' if I=¯1

    [19]  e←'n←÷L←1⌈300⌊⌊3×⌈/∣,v◇L←n×¯1+⍳L◇G←⍴v←L∘.×v[1;;]◇V←v+G⍴W'

    [20]  m←'v←T[D⌊K+1;;]-W←V←T[,K←K+1;;]◇e if
0∧.>,v[;;1]◇C[K;1]⎕GLINE V◇0⎕GLINE W'

    [21]  'i if mg◇L←⊖L[;⍳3]◇D←1↑⍴T←U[L;]◇K←0◇m if D◇n←-N,¯3
```

```
⁻2◇0□GLINE n↑T◇15□GLINE 1 3 2ρU[1,J-2↑N-1;]' if I=0 ◇ t←t+1↑1↓□AI ◇ P if mg
        ▽


        ▽ VXYZ2;D;J;s;V
    [1]   M←L[; 2 2 2 ρ 2 1 4 1 2 3 4 3] ◇ M←-/[4]U[M;⍳2]
    [2]   D←Φ3,J←ρs←⍳,M[;;1;]pv2 M[;;2;] ◇ L←DρL[; 2 3 ρ 1 2 4 3 2 4]
    [3]   □←D←+/V←Cte<s ◇ s←V/s ◇ MM←□←s[10↑⍋s] ◇ U←U[L←V≠L;⍳2] ◇ C←V≠2≠C
    [4]   M←+/[2]U ◇ M←M÷3 ◇ M←M,1 ◇ i if mg
    [5]   S←'□DM' ◇ ERRG←'OK' ◇ □ELX←'ERRG←□DM◇P if mg◇□ELX←S◇→' ◇ G wh B
        ▽


        ▽ R←A pv2 B;Q Produit vectoriel de A par B
    [1]   Q←⁻1+ρρA ◇ Q←Qρ';' ◇ ⍎'R←-/A[',Q,'1 2]×B[',Q,'2 1]'
        ▽


        ▽ Eff △ Efface les variables devenues inutiles
    [1]   △←□EX △
        ▽


        ▽ bon Signal sonore indiquant que tout va bien
    [1]   □SOUND 50 100
        ▽


        ▽ △E wh △L;△C Fonction "tant_que"
    [1]   △C←⍎⍕△L ◇ ⍎△C/△E ◇ →△C/1
        ▽
```

```
      ▽ err
[1]   ⎕ELX←⎕ALX←'⎕DM' ◇ ERREUR←'...OK...'
      ▽
```

fonctions non listées ici :

initg fonction de début de graphique (initialise le mode en fonction de la carte)

gfin fonction de fin de graphique (restaure le mod « texte »)

memec fonction de mise en mémoire de l▽écran de texte

restec fonction de restauration de l▽écran de texte

qR qBOX R équivalent de ⎕R ⎕BOX R en APL68000 : transforme R en vecteur si R est une matrice de

caractères, et réciproquement

ega variable (1 ou 0 si carte ega ou cga)

her variable (1 ou 0 si carte "Hercules" ou non)

mg variable (1 ou 0 selon que l'on désire des messages ou non)

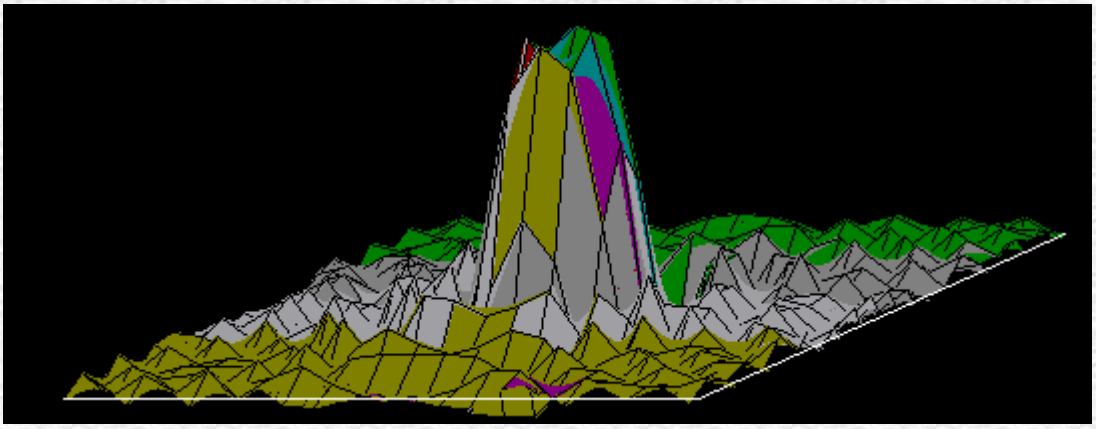
Cte variable scalaire numérique (valant ici 100)

On peut proposer, à partir du chapeau mexicain, la variable AIGUILLE :

```
      AIGUILLE
112
      AIGUILLE
-8.1 8.1 0.8
-8.1 8.1 0.8
           2    2               2    2
```

```
    10×(SIN(((X )+(Y ))EXP .8))÷(((X )+(Y ))EXP .7)
```

de telle sorte que l'instruction `COURBE AIGUILLE` donne l'image suivante (relative à un paysage fractal quelque part au Nouveau Mexique ou dans le Colorado) :

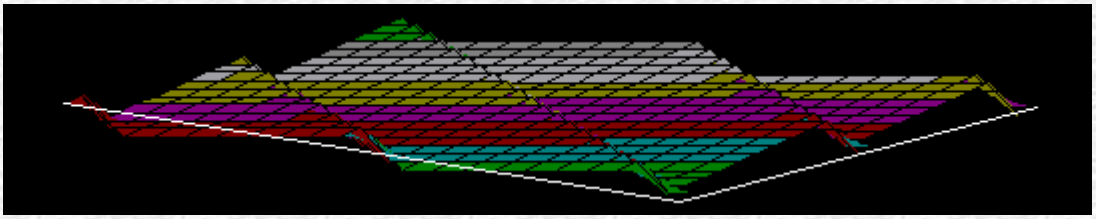


```
                                                                    ∇ R←A
EXP B

    [1]   cartes '★'

        ∇


        ∇ cartes ⍙;⍫; ' ';⍫⍫

    [1]   ⍫⍫←'R←A∘.',⍙,' B' ◇ ⍫←1=⍴⍴B ◇ '⍫⍫ if ⍫←⍫∧1=⍴⍴A' if    ⍫∧ ' '←0
<⎕NC 'A'

    [2]   ⍫⍫← ' '/' A ' ◇ ⍫⍫←'R←',⍫⍫, ',' B' ◇ ⍫⍫ if~⍫

        ∇
```

La fonction `EXP` est une simple exponentielle en produit extérieur. Toutefois, l'utilisation de la sous-fonction `cartes` permet de généraliser le concept. Voici ce que l'on obtient à partir de la variable `PYR:`

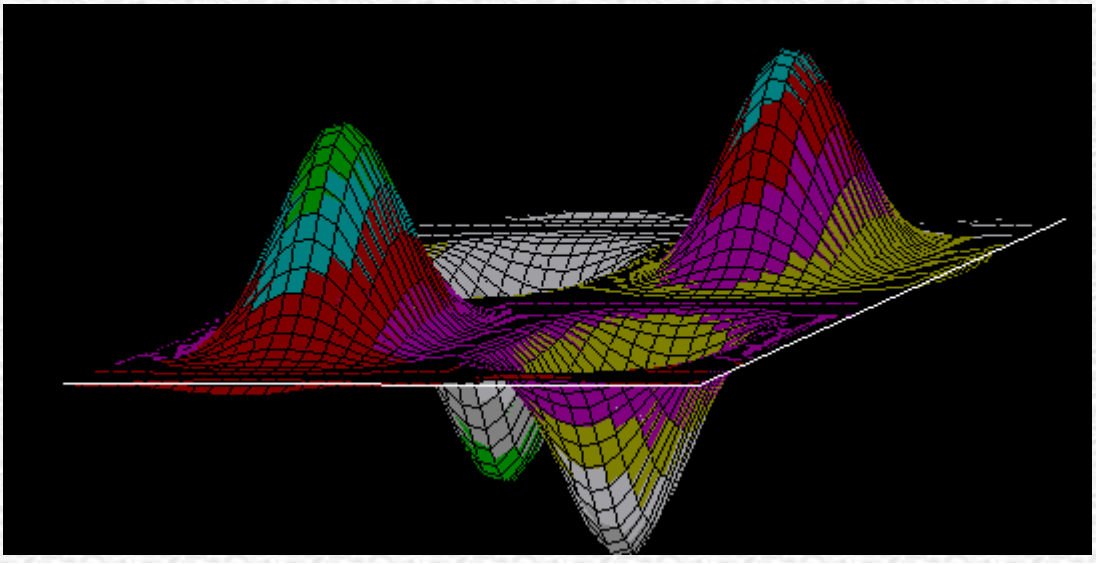
```
      PYR (de dimension 76 caractères, une pyramide Maya ?) :

    -3 3 0.3
```

```
-3 3 0.3

          2   2

¯0.4×˜20×(EXP(-((X  )+(Y  ))))×(X+Y)
```



On peut conserver en mémoire une foule d'équations, en APL*PLUS ou en APL_SE et ainsi explorer les fonctions Z=f(X,Y) à peu de frais. Exemple :

```
   C3 Le graophique a alors 36×71 soit 2556 points

-3.5 3.5 .2

-3.5 3.5 .1

      3     3

3×((SIN X) )×((SIN Y) )
```



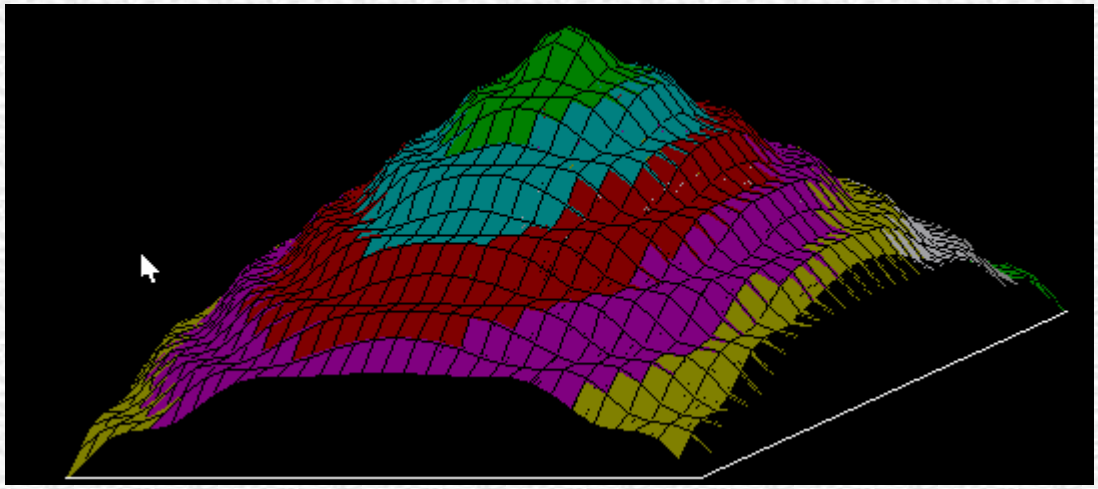
C2

-30 30 2

-30 30 2

2 2 .5

-Z-SIN Z←((X)+(Y))



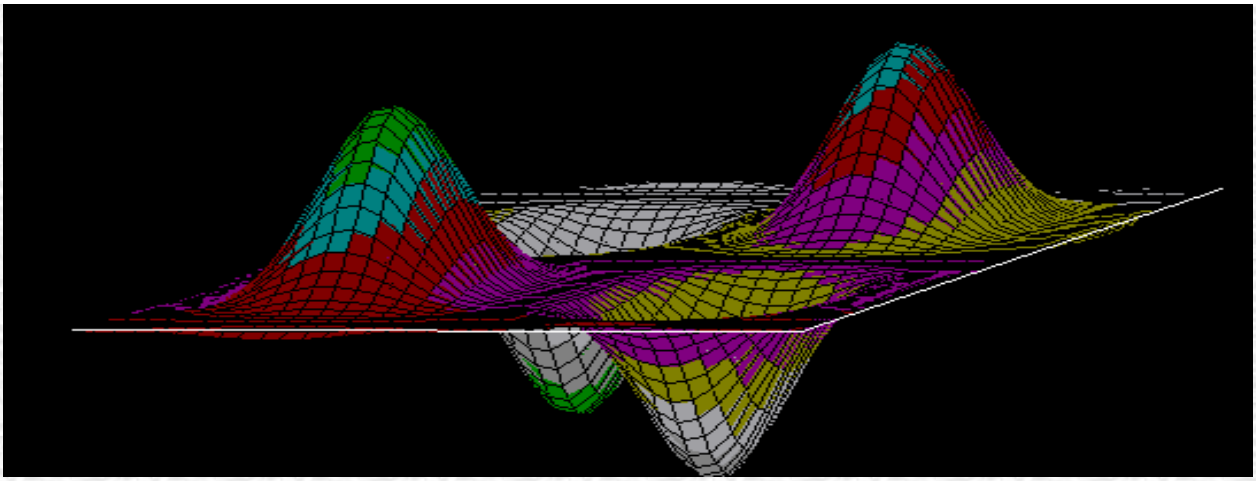
C1 seulement 661 points soit 31×21

-3 3 0.2

-3 3 0.3

2 2

-4×(EXP(-((X)+(Y))))×(X+Y)



Bibliographie

par AFAPL

Vivek S. Borkar, *Probability Theory, an advanced couse*, Springer New York (1995) ISBN 0-307-94558-X, 138 p.

Ouvrage court de Mathématiques pures, ultra-modernes.

D. C. Rapaport, *The Art of Molecular Dymamics Simulation,* Cambridge University Press (UK) (1995) ISBN 0-521-44561-2, 400 p.

Livre orienté vers l'Informatique. Nombreux programmes de simulation en langage C.

Jean-Loup Chrétien, *Sonate au clair de terre*, Denoël, (1994) ISBN 2.207.24153.X, 230 p.

Général de réserve de l'Armée de l'air, Jean-Loup Chrétien est le premier spationaute français a être parti dans l'espace. Un de ses coéquipiers s'appelle Vladimir Djanibekov. Commandeur de la Légion d'honneur, cité à l'ordre du mérite et héros de l'Union Soviétique, il est aujourd'hui directeur des astronautes du Centre national d'Etudes spatiales.

[1] La fonction MP est d'ailleurs plus rapide que les fonctions généralement proposées pour multiplier les polynômes.

[2] Rien à voir avec Ferdinand Lop, célèbre personnage du Quartier Latin d'il y a plus d'un quart de siècle.

[3] En 1975, dans un article appelé « Le Scalaire Neutre », paru dans la revue française INTER-APL (édité par SLIGOS), j'avais introduit le type polynôme; cet article traitait des objets de type et de contenu indéterminés, aussi appelés « Jokers » et symbolisés par un petit rond (<maj> J au clavier APL) : un polynôme à une variable était un scalaire, provenant du codage d'un vecteur numérique dans une base indéterminée par une instruction telle que °⍵ ou seulement ⍵ , puisqu'il n'existait pas de fonction monadique pour le symbole ⍵ . On pouvait alors effectuer des opérations sur les polynômes comme sur de braves objets APL.

[4] La puissance P du groupe cyclique d'une matrice binaire inversible est la puissance à laquelle il faut élever la matrice (grâce à P-1 produits matriciels effectués modulo 2 par elle-même). Bien entendu, les algorithmes développés pour calculer P ne font pas appel à cette définition, purement mathématique (voir la section FORTRAN).

[5] Le **rang** d'une matrice doit être considéré ici au sens mathématique et non au sens APL usuel; si une matrice M est de **rang** N, elle a pour dimension APL : 2½N . Le « domino » d'APL ne résiste pas à des rangs élevés (souvent au-delà de 8), à cause des troncatures et approximations de l'arithmétique dite flottante, tandis que l'inversion modulo 2 donne des résultats TOUJOURS justes. Comme on peut exprimer beaucoup de problèmes sous forme d'équations modulo 2, leur résolution devient aisée et, surtout, indépendante de la machine. Seuls les cryptographes sont au courant de ce genre de technique qui devrait se généraliser... à condition de vouloir sciemment s'y frotter.

[6] Il suffit que l'exposant proposé pour la matrice n'excède pas ¯1+2**31 s'il est positif. On se rendra compte, si l'on effectue une telle expérimentation, que les algorithmes untilisés pour l'exponentiation de matrices binaires sont très rapides. Elever une matrice binaire (de rang raisonnable, c'est-à-dire dont la représentation tient sur l'écran) à la puissance 1000000000 n'est pas une gageure : le résultat sera exact et obtenu en peu de temps. Il en va tout autrement pour l'obtention de la puissance du groupe cyclique d'une grosse matrice...

[7] L'œil humain reste le meilleur outil de vérification que nous possédions... en double.

[8] En fait, ET remplace la « strand notation » d'APL2 à un seul niveau, pour un interprète APL*PLUS, c'est déjà bien suffisant ! Mais les concepts vont plus loin : sous INTERP, on obtiendra le même résultat sans ET,

simplement en séparant A, B et C qui peuvent alors être des expressions définissant ces objets, par 5 blancs.

[9] et même de « Dictionnaire »

[10] Il existe aussi un module écrit en Turbo-Pascal (Borland).

[11] Si des lecteurs ont des idées pour obtenir P plus rapidement, leurs suggestions sont tout à fait bienvenues.

[12] Il semble vicieux, avec ce simulateur, d'utiliser des expressions parenthésées; d'ailleurs, dans toute la zone comme dans toutes les zones que nous offrirons à la Bourse d'Echange, aucune fonction n'en contient - comme si les parenthèses n'avaient jamais fait partie de la syntaxe d'APL.

[13] On s'en est servi dans une autre publication, pour symboliser l'opérateur JUSTE. Dans la zone complète COCRAY, fonctionnant en APL*PLUS, cette dernière extension fonctionne en même temps que le même symbole définit, en tant que fonction, la transformée cognitive. En outre, dans COCRAY, une expression telle que 5:N a pour résultat le quinto de **rang** N. car il ne faut pas oublier que : peut aussi acquérir une syntaxe diadique en tant que fonction (il se passe la même chose, après tout, avec les symboles / et \ de la syntaxe APL classique).

[14] Rappelons que dans un pariton (pour simplifier, d'une séquence vectorielle de dimension 2^k avec k entier non négatif, auquel cas le pariton est une matrice binaire carrée), la dernière ligne contient la séquence (régénérée par intégration), la dernière colonne contient la transformée cognitive et la seconde diagonale contient la transformée hélicoïdale. Dans cette zone COCSE, l'algorithme utilisé pour construire le pariton à partir d'un vecteur binaire de dimension non égale à 2^k a toujours comme résultat une matrice carrée, les dernières intégrales présentes dans les dernières lignes étant alors omises. L'argument droit de Œ: peut être un vecteur de caractères, auquel cas la représentation interne (8 bits par caractères) est prise en compte. Les arguments en caractères sont aussi admis pour les transformées. On peut bien sûr transformer bidimensionnellement un vecteur de matrices tel que ROBEE. Le résultat contiendra alors le vecteur des trois matrices, transformées bidimensionnelles, cognitives ou hélicoïdales selon la fonction utilisée. Rappelons que ces transformées sont équivalentes à la grande famille de transformations orthogonales habituellement en usage (cf. Fourier, Walsh, Hadamard ou les ondelettes) dans le « monde numérique ».