Paper presented at the Seminar "APL X, Diegem, Belgium, 25 November, 1987". Organised by the Genootschap Toegepaste Wiskunde en Informatieverwerking, Technologisch Instituut K.VIV, and the Belgian APL-CAM Users Society BACUS.

APL- ISO versus APL 2

Gérard A. Langlet
CEA/IRDI/DESICP/DPC/SCM
C.E.N. Saclay
F-91191 Gif-sur-Yvette

Résumé

De nombreux conférenciers nous présentent souvent les fantastiques avantages d'APL2 comparé à l'APL-ISO. Mais combien de temps devrons-nous attendre encore pour disposer d'une ou plusieurs implantations complètes et assez performantes sur des machines de bureau ou même portatives ? Et à quel coût ? Sans opérateurs définissables par l'utilisateur, sans arithmétique complexe et sans extensions géniales, avec l'APL-ISO seul dans de minuscules zones de travail, nous reste-t-il encore un espoir de résoudre les problèmes sympathiques qu'APL2 est censé traiter bien mieux ?

Nous prenons le pari : dans bon nombre de cas, on peut écrire des codes à peu près aussi courts et plus performants, en APL-ISO pur, que ceux déjà présentés à la gloire d'APL2. APL, même dit APL1, n'est-il plus un prodigieux outil de réflexion ?

Abstract

Many speakers often display the fantastic advantages of APL2 compared to ISO-APL. But how long will we have to wait for some good and rather complete implementations on lap-size or desk computers? And at what price? With no user-defined operators, no complex number arithmetic and no sophisticated extensions, with ISOAPL al one in tiny workspaces, can we keep a small hope of solving the nice problems that APL2 is supposed to handle much better ?

We take the bet: in quite a lot of cases, it is possible to write short code in pure ISO-APL that will run faster than sorne exampIes already written in order to glorify APL2. Isn't APL, even APL-1, a marvelous tool of thought anymore?

APL2, APL Dyalog, NARS et APL90 représentent actuellement les principaux produits à la disposition de l'utilisateur dans le domaine de l'APL étendu. Et alors. APL tout court, que d'aucuns ont tendance à qualifier d'ancien, ou d'APL 1, mais que nous apellerons dorénavant l'APL ISO ou OSI, a-t-il cessé d'exister, ou ne représente-t-il plus aucun intérêt ?

D'abord, ne nous leurrons pas : Aucun APL étendu ne fonctionne actuellement sur micro-ordinateur dans une version complète et performante ; il faut encore soit accéder à un gros système soit posséder un bon "mini" équipé par exemple du système UNIX. Mais la tendance évolutive du rapport performances/coût permettra certainement, dans un proche avenir, de disposer de ces APL étendus sur de petites machines, à condition que les implanteurs daignent y consentir, et que le prix des logiciels suive la baisse constante de celui des machines.

Les APL étendus ne sont pas compatibles entre eux dans la totalité de leurs innovations, et, bien qu'un groupe international de l'ISO tente de concilier les parties, on peut prévoir de longues années de réflexion et de discussions avant ou'un consensus véritable se dégage et aboutisse à une norme de près d'un millier de pages… Rendez-vous en 199X ?

La norme internationale APL-ISO (et AFNOR) (IS086) à l'élaboration de laquelle de nombreuses personnes ont activement participé, doit déjà permettre l'écriture de programmes dits conformes, capable de produire les mêmes résultats sur différentes implantations, y compris, à de rares exceptions près, celles des APL étendus.

Contrairement à ce qui se passait autrefois, de nombreux utilisateurs actuels programment sous plusieurs implantations d'APL dans leur travail quotidien, et cette évolution ira en progressant parallèlement au foisonnement fabuleux des micro-ordinateurs. En ce qui me concerne, je pratique de façon quasi-quotidienne APL 90 et APL Dyalog en tant qu'APL étendus, sur machine UNIX, mais aussi APL*PLUS sur PC IBM et compatibles, ainsi que trois versions d'APL.68000 sur des machines différentes, sans compter les versions sur gros systèmes. Comment voudriez-vous, dans ces conditions, parvenir à maîtriser cette Tour de Babel en puissance, sachant qu'en outre, APL n'est pas le seul langage utilisé au laboratoire…

Bien sûr, cela exige de connaître un tant soit peu les systèmes d'exploitation des machines, les possibilités d'interfaces avec les autres langages ou les fonctionnalités annexes telles que les fichiers APL, les fichiers dits "natifs", les graphiques, la souris, la couleur ainsi que le multifenêtrage. Nécessité oblige : dans la mesure du possible, les mêmes programmes se retrouvent dans les différentes implantations, et la seule manière simple de tenir correctement les rênes reste encore de se conformer le plus possible à la norme. Cette dernière phrase contient en soi une restriction bien compréhensible. Il n'existe actuellement aucun APL respectant la norme dans son intégralité ; par contre, tous la suivent, dans ce que l'on pourrait appeler leur noyau, disons en gros, à 98 % ; il émerge donc un tronc commun d'au moins 95 % entre tous les APL par rapport à la norme ISO. Mais qui connait déjà à fond et exploite correctement 95 % des immenses possibilités d'APL non étendu ?

Toutefois, doit-on se priver de certaines merveilleuses fontionnalités, autres que celles décrites dans la norme ou celles relatives aux tableaux généralisés, alors qu'elles vous permettent d'accélérer notablement les performances de vos programmes ou de réduire leur encombrement. Citons, à titre d'exemple, la sélection multiple ("replicate"), le tri alphabétique, l'identité ("match"), le traitement des chaînes de caractères ou la fonctionnalité de contrôle d'écran (ŒCC) pour APL.68000, ou encore l'éditeur en plein écran ou l'appel de sous-programmes écrits en langage compilé pour APL*PLUS sur PC.

Certaines innovations, non normalisées, comme l'obtention des caractères superposés sans obligation d'utiliser la touche de "retour en arrière", sont également bien pratiques.

Il vaut mieux, une bonne fois pour toutes, simuler ces précieuses fonctionnalités dans les implantations d'APL qui ne les possèdent pas, sous formes de fonctions optimisées. Tout le reste du logiciel restera commun à toutes les versions, et seules quelques fonctions utilitaires diffèreront d'une version à l'autre.

Pour revenir à l'argument précédent, on peut, sans fausse modestie, avancer que la norme APL ISO constitue actuellement le plus vaste ensemble de fonctionnalités jamais normalisé à l'échelle mondiale pour un langage de programmation. Avant de prétendre qu'elle est insuffisante, et de vouloir lui donner des allures de "Sphère de Hubble", respectons-la, puis essayons de réfléchir quelque peu sur des cas concrets.

Il est assez fréquent que dans la littérature "APListe", on nous vante les mérites d'APL2 ou de ses cousins en disant : vous voyez, grâce aux nouvelles fonctionnalités, dont vous serez hélas privés à tout jamais si vous ne vous décidez pas à rejoindre nos rangs sur le champ, vous pourrez "simplifier" vos programmes et réaliser des merveilles. Cela s'avère vrai, heureusement, dans de nombreux cas, mais pas toujours… La métaphore cé1èbre du marteaupilon utilisé pour écraser les mouches reste d'actualité, particulièrement en ce qui concerne APL2.

Prenons parmi tant d'autres le cas de la géométrie fractale. Il s'agit du problème du triangle équilatéral dans lequel on substituera à chacun des côtés un contour de quatre segments formant saillie (Man77) :

_____ est remplacé par ___/____

Puis, chaque segment obtenu de cette manière se trouve lui-même remplacé par un contour homomorphe, et ainsi de suite, jusqu'à obtenir une figure dite "fractale", dont il est impossible de mesurer la longueur, comme c'est par exemple le cas pour les côtes de la Bretagne ou le contour des cristaux de neige. L'effet de "sensation fractale" débute en fait, sur un écran graphique, lorsque la longueur des segments élémentaires devient du même ordre de grandeur que la résolution en points élémentaires ("pixels"). Le processus de génération d'un contour fractal est itératif, donc profondément "anti-APL". Il peut faire appel à la récursivité, ce qui provoque vite des empilements de contextes trop volumineux pour une zone de travail de micro-ordinateur.

On vous dit : le problème est trivial si vous disposez d'APL2, avec les tableaux généralisés, les nombres complexes, les opérateurs définissables par l'utilisateur, plus quelques autres extensions certes géniales, ainsi que cette magnifique interface graphique qui s'appelle GDDM. Mais… sur la plupart des machines, on n'a guère accès à tout cela ! Par contre, le cerveau possède des milliards de neurones qu'il serait malvenu de laisser chômer en pareil cas. Attaquons-nous donc à ce problème en nous imposant, de plus, les limitations suivantes :

Sur un écran banal, par exemple celui d'un téléviseur, ou de votre micro-ordinateur préféré, vous devez faire tenir à la fois le programme et le dessin final sachant :

a) que vous n'accédez à aucune des fonctionnalités miraculeuses mentionnées ci-dessus,
b) que la taille de la zone de travail disponible est très réduite,
c) que vous n'avez pas la patience d'attendre 107 ans le résultat escompté ni l'intention de régler une facture salée en fin de mois.

De plus, vous décidez, pour faciliter la lisibilité et la transcription des algorithmes en d'autres langages, de n'utiliser qu'un sous-ensemble très restreint de l'APL-ISO : Pour corser le tout, vous prétendez ignorer l'usage des étiquettes, et, pourquoi pas, des parenthèses : voilà une fort prétentieuse gageure. Ah, j'oubliais : le logiciel final ne doit pas être beaucoup plus long que celui présenté comme la preuve de l'efficacité d'APL2…

En général, lorsqu'on se fixe de telles conditions, quelle que soit la nature du problème posé, on trouve, en APL, des solutions très simples et surtout encore plus performantes que celles auxquelles on pensait parvenir, car, plus on simplifie, plus on voit apparaître de nouvelles simplifications ; une longue expérience a montré que ceci s'applique, entre autres, aussi bien aux problèmes physico-mathématiques qu'au traitement de textes et aux systèmes dits experts.

Pronosons donc quelques solutions que vous pourrez tester sur votre PC :

- La fonction graphique "PLR" dépend de l'implantation. Elle est donnée ici pour l'APL*PLUS PC. (Voir (Lan87-2) pour la version APL.68000 sur le QL de Sinclair.)
- Les fonctions "inig" et "fing" ouvrent et ferment respectivement le mode graphique le plus répandu (CGA) sur les PC et compatibles. La variable "gc" correspond au nom de la carte graphique du PC, par exemple 'IBMCOLOR' ou 'HERCULES'.

Première Solution : Cinq Petites Fonctions

On évite partiellement la récursivité, donc le remplissage rapide de la zone de travail, en effectuant les itérations par un renvoi au début de la fonction F3.

Les coordonnées calculées pour l'ensemble des points sont conservées dans la variable globale R. La variable ECH correspond à l'échelle (prendre par exemple 800).

La partie purement graphique ne s'exécute que lorsque tous les points ont été effectivement calculés. À titre de précaution, pour éviter l'explosion de la petite zone de travail, les coordonnées accumulées pourraient être converties en entiers. Bien sûr, il existe tout de même une limite, car :

à l'ordre 0, le triangle équilatéral ne contient que 3 segments,
à l'ordre 1, l'étoile à 6 branches en contient 12, … et
à l'ordre 5, la figure se compose de 3072 segments ou points.

Toutefois, à partir de l'ordre N=4, la figure est difficile à obtenir dans une zone de taille très réduite.

```
  ∇ TRIANGLE1 N;C;D;J;K;L
[1] K←1,-J←÷3⋆0.5 ⋄ C←0.5×Φ÷L←J,1 ⋄ D←÷2/ 3 2 1.5
      ⋄ N F1 0 0 ,C,2↑1 ⋄ PLR
  ∇
  ∇ N F1 B;C
[11] R←⌊0.5+ECH×B,2↑B ⋄ C←N=0 ⋄ →C/0 ⋄ N F2 B ⋄ R←⌊0.5+ECH×R
  ∇
  ∇ N F2 B;C
[1] R←2↑B ⋄ R F3 2↓B ⋄ C←0=N←N-1 ⋄ →C/0 ⋄ N F2 R
  ∇
  ∇ A F3 B;C
[1] R←⍳0
[2] R←R,A F4 2↑B ⋄ C←2>ρB ⋄ ⍎C/'→0,R←R,B' ⋄ A←2↑B
      ⋄ →□LC,B←2↓B
  ∇
  ∇ R←A F4 B;C
[1] R←B-A ⋄ C←R+.xL ⋄ C←C,R ⋄ R←R,R+.×K ⋄ R←D×R,C ⋄ R←A,R+6ρA
  ∇
```

Logiciel graphique pour PC :

```
  ∇ PLR ;C;D
[1] C←0∈ρR ⋄ →C/0 ⋄ C←1≥ρD←ρR ⋄ D←Φ2,D÷2 ⋄ ⍎C/'R←DρR'
[2] D←ρR ⋄ R←R-Dρ⌊≠R ⋄ D←1,ρR←R, 1 2 ↑R ⋄ inig ⋄ R←DρR
      ⋄ 1 □GLINE R ⋄ fing
  ∇
  ∇ inig
[1] 0 0 ρ0□GINIT gc
  ∇
  ∇ fing
[1] 0 0 ρ3 □INT 16,0ρ□INKEY
  ∇
```

Le caractère "diamant" ⋄ est une fonctionnalité optionnelle de la norme d'APL. Il sert à juxtaposer plusieurs instructions sur une même ligne. (Dans les discussions en cours, relatives à l'étude d'une norme étendue, il est question de rendre obligatoires toutes les fonctionnalités actuellement optionnelles.)

N.B. : Dans la plupart des implantations (APL.68000, APL*PLUS PC et APL2), l'expression `D←2/÷3 2 1.5` remplace avantageusement l'expression classique `D←÷3 3 2 2 1.5 1.5` grâce à l'extension "replicate" que la norme ISO ne cautionne pas encore, mais que toute la communauté approuve unanimement.

Deuxième Solution : En Trois Fonctions Hélas Trop Longues…

Mais, cette fois, l'exécution s'avèrera plus rapide, car on tient compte des propriétés géométriques de la figure : les coordonnées des segments composant la totalité du contour sont déduites, par deux rotations et deux symétries, d'un motif élémentaire assez réduit. Dans ce cas également, la totalité des points est calculée avant l'exécution du graphique proprement dit et reste disponible dans la variable globale R. Ici, la variable ECH a été rendue locale.

```
  ∇ TRIANGLE2 N;B;D;M;ECH
[1] ECH←130 ⋄ D←3★0.5 ⋄ B←¯3,D, ¯1,D,0,D+D×N>O ⋄ M←ROT 60
      ⋄ RE ⋄ D←Φ2,0.5×ρR
[2] D←ρR←DρR ⋄ D←Dρ ¯1 1 ⋄ R←ECH×R⍪ 1 0 ↓D×⊖R ⋄ D← 1 0 ↓R
      ⋄ D←0.5+D ⋄ D←⌊D
[3] N←ρB←⌊0.5+R+.×ROT 120 ⋄ R←⊖ 1 0 ↓B×Nρ¯1 1 ⋄ D←0ρR←D⍪R
      ⋄ B←0ρR←B⍪R ⋄ PLR
  ∇

  ∇ RE;C;D
[1] D←ρR←B ⋄ C←0≥N←N−1 ⋄ →C/0 ⋄ B←B−R←Dρ2↑B ⋄ B←R+B÷3
      ⋄ R←B−C←Dρ¯2↑B
[2] D←Φ2,0.5×ρR ⋄ R←DρP ⋄ R←R×Dρ¯1 1 ⋄ D←ρB←B,2↓C+,⊖R
      ⋄ D←Φ2,0.5×ρR←B−Dρ2↑B
[3] R←DρR ⋄ D←ρR←R+.×M ⋄ R←R+Dρ¯2↑B ⋄ B←B,2↓,R ⋄ →1
  ∇

  ∇ R←ROT D
[1] R← 2 2 ρR,Φ 1 ¯1 ×R← 2 1 ○○D÷180
  ∇
```

Notes : Le symbole `⍪` peut être remplacé par `,[⎕IO]` s'il n'est pas disponible. Certaines expressions pourraient être condensées ; elles sont ici séparées exprès pour éviter le diagnostic "WS FULL" à un ordre élevé.

Le logiciel graphique (PLR) est le même que précédemment.

Troisième Solution : Trois Fonctions Très Condensées

On ne désire plus conserver en mémoire les coordonnées des points calculés, mais on souhaite atteindre un ordre élevé (par exemple 6, ce qui implique 12288 segments), toujours sur micro-ordinateur, et produire le graphique au fur et à mesure du calcul. Bien sûr, on boucle sur chaque segment, mais on a remarqué que chaque segment se déduisait du précédent par une translation élémentaire et une rotation, soit de 60 degrés, soit de ¯120 degrés.

La fonction BIT évalue a priori et récursivement, sous forme binaire, cette alternance de rotations ; la matrice correspondante est préparée à l'avance. En APL.68000, les constantes binaires sont codées en bits, il apparait donc très intéressant de conserver

cette information très compacte. Sur le PC, il n'existe que des entiers courts pour coder les booléens, mais la taille de l'espace de travail supporte allègrement ce gaspillage !

Effectivement, le graphique démarre immédiatement, et l'on peut se permettre de superposer les figures obtenues pour différents ordres.

Le logiciel graphique (fonction "PL", spécifique ici d'APL*PLUS PC) est vraiment très simple. Les appels aux fonctions "inig" et "fing" (voir ci-dessus) ont été incluses dans la fonction maîtresse. ROT est la même fonction que précédemment et n'est appelée qu'une fois. L'échelle est définie sous forme de constantes dans la fonction maîtresse.

```
  ∇ TRIANGLE3 N;A;L;M;P;Q
[1] R← 200 750 ◇ P←R←R,R+2↑810÷3⋆N ◇ BIT ◇ A←1+¯2×¯1ΦA
     ◇ M←ROT 60 ◇ inig
[2] PL Q←R ◇ L←4ρ2↓Q ◇ N←-≠ 2 2ρ2ΦQ
     ◇ L←p∨.≠R←L+¯4↑N+.×M×1↑A←1ΦA ◇ →L/□LC
[3] fing
  ∇

  ∇ BIT
[1] A←N≤1 ◇ ⍎A/'A←-1+N ◇ →0,A←A↑1' ◇ N←N-1 ◇ BIT
     ◇ A←,0,1,0,A∘.∨,0
  ∇

  ∇ R←ROT D
[1] R← 2 2 ρR,Φ 1 ¯1 ×R← 2 1 ○○D÷180
  ∇
```

Logiciel graphique pour PC :

```
  ∇ PL B;C
[1] C←0∈ρB ◇ →C/0 ◇ 1 □GLINE 1 2 2 ρB
     ⍝ B: vecteur Xi, Yi, Xf, Yf
  ∇
```

En bref, le pari initial a été plus que tenu, car, même si la zone de travail n'offre que 24 K octets, ce qui est le cas dans un QL Sinclair de plus bas de gamme, les trois solutions cohabitent avec d'autres utilitaires dont un mesureur de performances et surtout avec un éditeur en plein écran, lui-même entièrement écrit en APL, capable de traiter aussi bien les fonctions que les variables, numériques ou alphanumériques. Il reste encore près de 16 K octets disponibles, sachant que l'éditeur occupe 6 Koctets à lui tout seul.

Il n'est pas question ici de décrire en détail une foule d'applications dans des domaines parfois très spécialisés comme la cristallographie ou les bases de données de formules chimiques complexes ; nous en avons déjà discuté, par exemple lors de précédentes journées APL de l'AFCET. Depuis cette époque, nous avons entrepris une modélisation systématique de tous les algorithmes utiles, sur de petits micro-ordinateurs en priorité, pour nous obliger à rester dans la norme ; une fois cette étape effectuée, nous utilisons à fond, et seulement si nécessaire, les fonctionnalités nouvelles des APL étendus, le plus souvent pour le traitement de chaines de caractères. De plus,le style de programmation choisi, sans parenthèses ni retour en arrière - seuls :

```
→0          →1          et          →□LC
```

sont autorisés - permet de traduire très rapidement (et peut-être un jour facilitera une conversion automatique) en langage compilable (Fortran 77, Pascal ou C) les algorithmes très performants mis au point en APL.

Bien que notre travail comporte assez souvent des applications de physique et de géométrie, l'implantation des nombres complexes en APL ne nous apporte jusqu'ici aucun avantage certain (cette opinion n'étant certes pas partagée par tout le monde), car ils sont restreints à des applications à deux dimensions et remplissent très vite la

mémoire, tandis que le calcul matriciel, tel qu'il existe depuis plus de 15 ans en APL "de base", se trouve encore malheureusement ignoré de nombreux utilisateurs. Il offre des possibilités considérables pour les graphiques en trois dimensions.

La gestion des évènements est trop disparate dans de nombreuses implantations d'APL et parfois d'emploi trop compliqué et fouillé ; seul `□EA` (présent dans APL2 et APL90) nous semble vraiment utile.

Quelques extensions comme la réduction par un vecteur d'entiers ("replicate") ou le tri des vecteurs et des tableaux de caractères, extensions sur lesquelles un accord des implanteurs semble acquis, ont aussi apporté un confort certain.

En ce qui concerne les tableaux généralisés, l'expérience montre que dans la plupart des cas, des vecteurs de vecteurs ou des vecteurs de tableaux à deux dimensions seraient bien suffisants. Les opérateurs définis par l'utilisateur méritent une exploration spéciale, mais si l'on n'en dispose guère, il est relativement aisé de les simuler. Nous attendons beaucoup, par contre, des primitives graphiques (Lan87) et des extensions "orientées vers les objets" (Gir87) qui commencent à voir le jour çà et là, et ouvrent la voie vers des horizons assez prometteurs, tout en correspondant à un sérieux besoin, ressenti depuis longtemps par une grande partie de la communauté APL, d'ouverture vers des domaines modernes de l'informatique.

. . .

Références

ISO (1986) ; Texte pour une Proposition de Norme Internationale (DIS 8485) - Langages de Programmation - APL ; ISO/TC97/SC22 /WG3/N55.

Graphical Data Display Manager - Programming Reference ; IBM FN SC33-0101.

B.B. Mandelbrot (1977) ; Fractals, Form, Chance and Dimension ; W.B. Freeman & Co. ; ISBN 0-7167-0473-0.

G. Leeten (1987) ; Complex Numbers in APL2 and their Applications ; APL-CAM Journal, Vol. 9, No. 2, pp. 398-418 (cf. aussi APL IX, Sofitel Bruxelles, Novembre 1986).

G.A. Langlet (1987) ; How Graphies Could be Simplified in APL ; APL87 Conference Proceedings, Dallas, Texas, pp. 419-423.

G.A. Langlet (1987) ; L'APL ISO n'est-il pas suffisant ? ; Journées APL de l'AFCET, Saint-Etienne, France, Juin 1987.

J.J. Girardot & S. Sako (1987) ; An Object Oriented Extension to APL ; APL87 Conference Proceedings, Dallas, Texas, pp. 128137.

Annexe

Programmes cités dans (Lee87) en APL2, pour l'obtention du Flocon de Koch :

```
  ∇ Z←A G1 B
[1] ⍝ Triadic Koch Snowflake
[2] Z←(B-A)÷3
[3] Z←(A+0,Z,(Z×(3⋆.5)×1D30),(Z×2)),B
  ∇

  ∇ Z←N(GEN FRAC)INIT
[1]  ⍝ FRAC est un opérateur (récursif) défini par
l'utilisateur en APL2
[2] Z←INIT
[3] →(N=0)/0
[4] Z←∈(-( ⍴Z)≠⍳⍴Z)↓¨"Z←2 GEN/Z
[5] Z←(N-1)(GEN FRAC)Z
  ∇
```

Syntaxe : MDRAW N (G1 FRAC) TRI

avec :

MDRAW : fonction graphique de dessin utilisant GDDM,
N : ordre, N∈0,1,2,3,4,... ,
TRI : coordonnées ini tiales des sommets du triangle (vecteur complexe).