# Pédagogie, récursivité et vitesse

Michel J. Dumontier

**Résumé :** Dans le numéro 35 des "Nouvelles d'APL", page 46 la petite phrase "nous reprenons nos cahiers poussiéreux" était bien intentionnelle, le présent article tente de mettre les points sur les i, d'autant plus que nous avons reçu des reproches injustifiés.

On fait souvent le reproche à APL d'être abscons: si une fois, qui n'est pas coutume, un auteur essaie de mettre de la clarté en s'appuyant sur des données supposées connues (tout au moins apprises au Lycée: Cramer : ça vous dit quelque chose ?),au risque de fournir une fonction récursive susceptible d'être plus lente qu'une non récursive et qui en définitive est tout aussi rapide: il ne faut pas lui en vouloir!

### 1) Pédagogie

On apprend à l'école que la matrice adjointe d'une matrice est la transposée de la matrice formée par les cofacteurs (tous les cofacteurs de chaque élément de la matrice).
Comment peut-on écrire d'une manière plus compréhensive la fonction suivante ?

```
      ∇ Z←ADJOINT A
   [1]  Z←⍉COFACTORS A
      ∇
```

exemple :

```
         COFACTORS ⎕←3 3⍴9?9
    2 7 3
    8 6 1
    5 4 9
     50 ¯67   2
    ¯51   3  27
    ¯11  22 ¯44
        2 7 3+.×50 ¯67 2
   ¯363
        DET 3 3⍴2 7 3 8 6 1 5 4 9
   ¯363
```

C'est bien beau d'avoir écrit cela mais on sent la récursivité venir si on n'y prend pas garde, mais poursuivons. Pour chaque couple I,J nous allons calculer le cofacteur de l'élément A[I,J].

```
      ∇ CO←COFACTORS A;I;J;⎕IO
   [1]  CO←⍳0 ◇ I←1
   [2]  INC:J←0
   [3]  IT:J←J+1
   [4]  CO←CO,A COFACTOR I,J
   [5]  →(J<(⍴A)[1])/IT
   [6]  I←I+1
   [7]  →(I≤1↑⍴A)/INC
   [8]  CO←(⍴A)⍴CO
   ∇
```

En ligne 4 on concatène tous les cofacteurs et en ligne 8 on reforme la matrice des cofacteurs. (NB. je dis bien on concatène des cofacteurs qui sont des scalaires et non des matrices comme certain(s) ont pu le croire!)
Jusqu'à présent, rien de sorcier, oui, mais il faut écrire maintenant la fonction quicalcule le cofacteur en I,J et c'est là qu'on a besoin du déterminant d'une matrice d'ordre N-1 qui, si on continue avec la facilité de compréhension, nous conduit à la récursivité:

```
      ∇ Z←A COFACTOR V;SIGN;⎕IO
[1]    ⎕IO←1 ◇ SIGN←¯1⋆+/V
[2]    Z←SIGN×DET A MINOR V
      ∇
```

Il ne nous reste plus qu'à écrire le mineur en I,J mais c'est là chose mineure.

```
      ∇ Z←A MINOR V;Q;⎕IO
    [1]    ⎕IO←1 ◇ Q←(V[1]≠ι(ρA)[1])/[1]A
    [2]    Z←(V[2]≠ι(ρA)[2])/Q
      ∇
```

Et le déterminant s'écrit avec une facilité déconcertante qui nous rappelle les bancsde l'école.

```
      ∇ Z←DET M ;⎕IO
   [1]    ⎕IO←1 ◇ Z←M[1;]+.×(COFACTORS M)[1;]
      ∇
```

Il en est de même de la fonction d'inversion de matrice:

```
      ∇ Z←INV M
   [1]    →((DET M)=0)/FIN
   [2]    Z←(ADJOINT M)÷DET M ◇ →0
   [3]    FIN:'DET =0'
      ∇
```

Nous avons donc une version récursive du calcul du déterminant mais néanmoins claire.Ceci, pour la pédagogie.

### 2) Récursivité et vitesse d'exécution.

La récursivité, c'est bien joli, mais cela se paie en vitesse d'exécution.
Mais qu'en est-il vraiment ici ?
Pour les besoins de la cause, nous opérions en APL sur des matrices d'ordre peu élevécar nous opérions dans Z/pZ avec p=256 autrement dit, dès l'ordre 6 nous mettions en oeuvre des nombres supérieurs au plus grand entier d'APL (de l'ordre de $10^{12}$ ce qui rendait impossible le calcul du modulo.
C'est pourquoi nous avions utilisé ensuite le langage J qui ne connaît pas de limite aux entiers.
(NB. contrairement à ce que certain(s) ont pu le croire, ce n'est pas la fonction DET qui était en cause, car elle peut calculer des déterminants de matrices de n'importe quel ordre tant que ce déterminant ne dépasse pas la valeur limite des réels en APL).

### 3) benchmarks :

```
      A←6 6ρ36?256
      A
  51 144 185 136 141 145
  52 166 210  21  83 213
  55 208 163  12 110 148
 206  19 147  64 197 105
   4  31   8 254 190 231
 139 175  43 237 204 101

DET A
1.62519988E13
```

Calcul du temps en millisecondes

```
      TIME 'K←DET A'
0
```

Pas de chance, (pour les contradicteurs!).
Nous allons prendre une matrice d'ordre 20. la récursivité prend du temps.

```
      D←20 20⍴?400⍴400

      D
 314 104 344  53 317  49 185 301 202   3 205 244   7 113 174  75 375 372 312 173
 198  30  33 251 379  16 284 173 204 161  32 108 147  99 149 349  70 193  39  30
 383 105 156  78 245 396  79 230 182  68 147   1 305 138  40 186 277 386  89 259
  71 362 325 340  99   5 228 184 106 206 179 360  22 149 366 211 253  14 391  26
 142 217 340 180 385 183 344   8 348 339   3  42 142   4 251 263 298 256 103 240
 195  17  58 302 197 362 119 350 388  45 229  56 288 181 311 350 293 283 165 329
 326 338 210 161 235 177 295 296 109 384 297 275 171 166 338  15  46 357 274 165
 116 347 100 281 265 169   8 246 193 130  51 207 162 171 193 317 134 343  21   1
 161 223 244 329 172 145 171 320  40 194 191 190 228   6  33 131 153  97 255 378
  78 174  24 245 237 347 355 320 327  21 319  35  58 163  87  31 125 310 244  93
 389 222 355 226 305 255  89  90 167 200  42 298 176 371 313  48 310 332  19 384
 327  71 225  49  31 216 108  59 232 392 257  32 367  58 305 348 204 166 383  55
 312 320 382  61 222 106 365 308 165  38 182 171 294 260 317 266 249 119 221 395
 274  95 300   4 239 149 330  75  84 266 361 246 123 288 131 310 187  52  57  84
 329 342 111 316 349 274 316 299  94 312 228 309 369 394 392 365 326 279 346 168
 365  13 252 199 333 108 227 202 313   1 211 116 252 324  26 324 165 218 267 201
 144 380 204 130 251 370 380 192 173 212  20  52 286 109  54 304 316 339 181 274
  85 137 183 229 322 368  66 230 398  35 156 261 137 176  84 164 326 257 175  20
 114  91 146 122 206  75 136  99 335 319 215 183 154 395  34  67  45  73 108 356
 224 301 360 114 361  66  96 299 325 325 333  48 173 372 297 307  88 135 236 233


    DET D
¯2.708884334E50
```

Prenons une fonction TIME qui calcule le temps en millisecondes et testons-la.

```
      TIME 'K←⎕DL 1'
1040
      TIME 'K←⎕DL 2'
2030
```

Pour être plus précis, calculons 10 fois le déterminant, nous diviserons par 10 pour avoir le calcul exact.

```
    ∇ F;K
[1]   K←DET D ⋄ K←DET D
[2]   K←DET D ⋄ K←DET D
[3]   K←DET D ⋄ K←DET D
[4]   K←DET D ⋄ K←DET D
[5]   K←DET D ⋄ K←DET D
    ∇

      TIME 'F'
110
```

11 millisecondes pour calculer le déterminant d'une matrice d'ordre 20.

Evidemment la valeur de ce déterminant dépasse largement la limite des entiers APL.

Nous nous en sortons mieux qu'on aurait pu le craindre, mais pour satisfaire la curiosité de certains, nous donnerons ici deux versions récursives dont l'une est de toute beauté et due à Kenneth Iverson [1] et une version non récursive.

Commençons par la version récursive autre que celle d'Iverson.
NB : il n'y a plus ici de but pédagogique!

```
 ∇ Z←DET1 M;J;Q
[1]    →(1=ρ,M)ρ0,Z←,M
[2]    →L2×ι(2=ρρM)∧=/ρM
[3]    →0,ρ⎕←'MAUVAISE STRUCTURE'
[4]    L2:→0×ι(1↑ρM)<J←(M[1;]=0)ιZ←,0
[5]    M←(J-1)⌽M
[6]    M←M-M[;1]∘.×M[1;]÷Z←M[1;1]
[7]    Z←(¯1⋆J-1)×Z×DET1 1 1↓M
     ∇

      ⎕VR 'G'
     ∇ G;K
[1]    K←DET1 D ◇ K←DET1 D
[2]    K←DET1 D ◇ K←DET1 D
[3]    K←DET1 D ◇ K←DET1 D
[4]    K←DET1 D ◇ K←DET1 D
[5]    K←DET1 D ◇ K←DET1 D
     ∇
```

Vérifions d'abord si elle fonctionne :

```
      DET D
¯2.708884334E50
      DET1 D
¯2.708884334E50
```

OK

Calcul du temps :

```
      TIME 'G'
60
```

6 millisecondes pour calculer le déterminant d'une matrice d'ordre 20.

(1) Two combinatoric operators : Kenneth Iverson congrès APL76 Ottawa.
Voici maintenant la version récursive de Ken Iverson:

```
DET:-/×/ 1 0 1 ⍉ω[PERM ρω;]
PERM:(0 MOD Z)[;(ιρP)+P-(ρP←2|+≠Z←REP ω)ρ 0 1]
MOD:ω[0;],[0]Z+ω[(1↑ρZ)ρ0;]≤Z←α MOD 1 0 ↓ω:0=1↑ρω:ω
REP:R⊤ι×/R←⌽1+ι1↑ω
```

L'inconvénient d'écrire un papier en temps réel, c'est qu'il peut y avoir des surprises,surtout lorsqu'on utilise les fonctions des autres. Ce n'est pas une catastrophe, je m'attendais à pouvoir calculer le déterminant  d'une matrice d'ordre 20 mais au delà de 7 on obtient WS FULL à la fonction PERM.
C'est un vieux papier et APL a longtemps été un outil expérimental qui malheureusement coinçait souvent à un ordre peu élevé, surtout lorsqu'on ne prenait pas de précautions:

```
      ρPERM 7 7
7 5040
      ρPERM 8 8
WS FULL
```

Nous donnons quand même le résultat:
110 ms pour calculer le déterminant d'une matrice d'ordre 7.
C'est plus lent, plus encombrant, mais c'est quand même beau!

Pour terminer en beauté, donnons la fonction de généralisation du déterminant.

```
GD:⍎(α,'/R')[0 2 1 2 3 ,0⍴R← 1 0 1 ⍉ω[PERM ⍴ω;]]
```

Celle-ci permet de calculer le déterminant comme nous venons de le faire

par : '-×' GD M
le permanent par: '+×' GD M
si la matrice logique 2|M couvre un carré latin par '∨∧' GD 2|M
le nombre de carrés latins distincts de 2|M  par '+∧' GD 2|M

Pour finir, la version non récursive:

```
    ∇ Z←DET3 M;K;S;G;F;P
[1]   K←⍳(S←1)↑⍴M
[2]   S←S×¯1*P≠G←¯1+P+F⍳⌈/F←|M[K;P←(⍳0)⍴1↑K]
[3]   M[P,G;]←M[G,P;]
[4]   M[K;]←M[K;]-(M[K←1↓K;P]÷M[P;P])∘.×M[P;]
[5]   →(1<⍴K)/2
[6]   Z←×/S,1 1⍉M
    ∇

    ∇ H;K
[1]   K←DET3 D ⋄ K←DET3 D
[2]   K←DET3 D ⋄ K←DET3 D
[3]   K←DET3 D ⋄ K←DET3 D
[4]   K←DET3 D ⋄ K←DET3 D
[5]   K←DET3 D ⋄ K←DET3 D
    ∇

      TIME 'H'
50
```

5 millisecondes pour calculer le déterminant d'une matrice 20x20

Assurons nous que les affectations ne coûtent rien:

```
    ∇ TT;K
[1]   K← D ⋄ K← D
[2]   K← D ⋄ K← D
[3]   K← D ⋄ K← D
[4]   K← D ⋄ K← D
[5]   K← D ⋄ K← D
    ∇
      TIME 'TT'
0
```

Si quelqu'un a une fonction récursive ou non qui calcule le déterminant plus rapidement que la fonction DET3, nous sommes preneurs.