Transformée de Fourier rapide généralisée

par Charles Hubert

Introduction

Pour le calcul de la transformée de Fourier, je présente ici une fonction dont j'avais écrit la première version en APLSV ; je l'ai réadaptée et améliorée pour APL*PLUS/PC puis étendue pour APL+DOS. Elle fonctionne aussi en APL+WIN. Elle devrait fonctionner en APL2, mais elle n'utilise pas les complexes de cet interprète, car ils n'existent pas dans les APL*PLUS.

Elle est généralisée dans trois sens :

1) Elle accepte des données complexes représentées par deux réels ;

2) Le nombre de valeurs (la période) peut être autre chose qu'une puissance de 2, ce qui implique de lui associer une fonction décomposant ce nombre en facteurs premiers ;

3) Elle traite plusieurs ensembles de données à la fois et accepte les tableaux gigognes.

4) Les fonctions: On peut les trouver dans le fichier APL*PLUS " trsf.sf" ; la composante 1 contient une table des matieres, les composantes suivantes contiennent les □VR de ces fonctions.

Description des deux fonctions

```
    ∇  y←a Fourier x;⎕io;c;e;i;j;k;l;m;n;p;q;r
[1]   ⍝∇ (4|a) =       0         1         2         3
[2]   ⍝∇ poids =  exp(-iωt)  exp(iωt)/n  exp(-iωt)/n  exp(iωt)
[3]    ⎕io←0 ⋄ a←↑a
[4]    x←c[;j←¯2↓⍳⍴⍴x]x
[5]    x←((×/r),2)⍴((r←¯1↓⍴x),2)↑x
[6]    x←⊃[0]∈¨⊂y←x++/,x-x
[7]    (r n)←2⍴⍴x ⋄ p←FacPre n ⋄ x←x÷↑a⍵ 1 n n 1
[8]    p←p,(×\¯1↓1,p),(1↓1),⌈0.5¯1↓1←⍵×\1,⍵p
[9]   ⍝ e←⍉ 2 1 ∘.oo(⍳n)÷n×↑a⍵ ¯0.5 0.5
[10]   e←1oo(n-⍳(2×n)-(4×n)|((↑a⍵ 4 ¯4)×⍳n)∘.+ 0 n)÷¯2×n
[11]   i←↑⍴p ⋄ →B
[12]  A:(q k l m)←p[i;] ⋄ c←e[n|(⍳m)∘.×k×⍳q;]
[13]   x← r n 1 (q+q)⍴q⍋ 0 1 3 2 4 ⍉ r l q k 2 ⍴x
[14]   x←(-/x×(⍴x)⍴k⍋c),+/x×(⍴x)⍴k⍋⍵c
[15]  B:→(0≤i←i-1)⍴A
[16]   (∈y)←, 2 0 1 ⍉x ⋄ y←⊃[j]y
    ∇
```

```
    ∇  k←FacPre n;⎕io;⎕ct;d;e;p;q
[1]   ⍝∇ p←FacPre n ; p = facteurs premiers de n
[2]    ⎕ct←⎕io←0 ⋄ e← 2 3 5 7 11 ⋄ p←e,1
[3]    d←1+(~2⍵⊃∨/d⍴¨e↑¨1)/⍳d←⍵/e
[4]    k←⍵n←↑n ⋄ n←1⌈⌊0.5+⍵n
[5]   A:e←(0=e|n)/e ⋄ →D
[6]   B:n←⌊0⊃q ⋄ k←k,1↑e
[7]   C:→(1⊃q←(0,1↑e)↓n)↓B ⋄ e←1↓e
[8]   D:→(0<⍴e)⍴C ⋄ p←d←¯1↑p
[9]    →(0<⍴e←(p≤⌊n*0.5)/p)⍴A
[10]   k←(k,⌊n)~1
    ∇
```

Fourier calcule la transformée de Fourier ; elle utilise **FacPre** pour décomposer la

dimension **n** de la période en facteurs premiers. **FacPre** peut être utilisée pour autre chose bien sûr. Pour la rapidité de **Fourier** mieux vaut que ces facteurs premiers soient petits ; les valeurs 2, 3, 5 permettent toutes les puissances de 2 mais aussi 24, 360, 1000, 1440...

Fourier calcule la somme

$$y_j = \sum_k x_k \exp(s\ 2\pi i jk/n) \qquad s \in \{+1, -1\} \qquad k \in \{1, 1/n\}$$

La constante **s** est le signe moins pour la transformée directe et le signe plus pour la transformée inverse ; la constante **k** prend une valeur pour la transformée directe suivant la convention adoptée et l'autre valeur pour la transformée inverse. On utilise **Fourier** de la façon suivante

```
y←opt Fourier x
```

opt est une option qui est prise modulo 4 et

```
opt = 0    =>    s = -1        k = 1
opt = 1    =>    s = +1        k = 1/n
opt = 2    =>    s = -1        k = 1/n
opt = 3    =>    s = +1        k = 1
```

x est un tableau de dimension **(nbCas,n,nbCol)**. **Fourier** ramène la dernière dimension à 2 par

```
(nbCas,n,2)↑x
```

La première colonne contient les parties réelles, la deuxième les parties imaginaires. Si on ne donne qu'une colonne, elle est alors complétée par des parties imaginaires nulles : les données sont réelles. **n** est le nombre de valeurs complexes dans la période. Quant à **nbCas** il structure les `(×/nbCas)` ensembles de données de même période qu'on traite à la fois.

De plus chaque case du tableau **x** peut être un tableau de profondeur quelconque, **Fourier** traite les cas correspondants simultanément. Il faut que les dimensions d'une même matrice **(n,2)** soient compatibles c'est-à-dire que l'opération **+/+/x** soit permise.

Le résultat **y** a la même structure que

```
(nbCas,n,2)⍴(n×2)/+/+/x
```

Par exemple si **x** est une matrice de dimension **(n,2)**, chaque case contenant soit un scalaire soit un vecteur de dimension V, le calcul est fait comme si chaque scalaire était étendu V fois dans un vecteur ; chaque case du résultat **y** contient un vecteur de dimension V dont les composantes d'une même place résultent des composantes de la même place du **x** étendu. Ainsi

```
  0 Fourier 12 1 ⍴ 0 (1 2) 3 4 5 (6 7) 8 9 0 11 12 13
```

donne le même résultat que

```
( 0 Fourier 12 1 ρ 0  1    3 4 5  6    8 9 0 11 12 13 ),¨
  0 Fourier 12 1 ρ 0    2  3 4 5    7  8 9 0 11 12 13
```

Commentaires sur la fonction Fourier

Si **T** est la période d'un phénomène continu échantillonné, on peut expliquer **Fourier [2]** en posant

```
ω = 2πj/T = pulsation de l'harmonique numéro j
t = kT/n  = instant d'échantillonnage numéro k
```

ainsi les commentaires **Fourier[1]** et **[2]** rappellent ce que la fonction calcule.

Fourier[4] ramène les **(×/nbCas)** ensembles au cas d'une matrice dont chaque case est un tableau. **Fourier[5]** force **x** à une matrice à 2 colonnes. **Fourier[6]** étend les cases de **x** qu'il faut, mémorise dans **y** la structure obtenue et en fait un tableau simple de rang 3 de dimension **(NbCas,n,2)** car la suite de la fonction est plus rapide sur un tableau simple que sur un tableau gigogne.

Fourier[10] calcule ce que **Fourier[9]** explique en commentaire, mais avec une précision améliorée, car la réduction à des sinus d'angles dans l'intervalle [-π/2, +π/2] est réalisée sur des entiers. Elle construit la table **e** des exp(**s** iωt) utiles pour la suite du calcul.

La boucle **Fourier[12]** à **[15]** est exécutée pour chaque facteur premier de **n**.

Pour analyser **Fourier[14]** il faut noter que si **A+iB** et **C+iD** sont deux vecteurs complexes rangés dans des matrices **AB** et **CD** à 2 colonnes, on peut développer leur produit scalaire ainsi :

```
(A + iB )(C + iD ) + (A + iB )(C + iD ) + ... =
  0    0   0    0     1    1   1    1

(A C - B D + A C - B D + ...) + i(A D + B C + A D + B C + ...)
  0 0   0 0   1 1   1 1           0 0   0 0   1 1   1 1
```

et on calcule ses parties réelle et imaginaire respectivement par

```
-/(,AB)×,CD        +/(,AB)×,⌽CD
```

Fourier[12] et **[13]** préparent les tableaux **x** et **c** nécessaires à l'exécution de **Fourier [14]**.

Fourier[16] reconstitue la structure attendue pour le résultat.

La fonction **Fourier** présentée ci-dessus fut construite directement à partir de la théorie ; elle n'est inspirée par aucun programme antérieur en Fortran, C ou autre.