CALCUL DE RACINES EN « J »
(suite)

Robert Coquidé

Dans « les nouvelles d'APL » n° 30 , la **pro-conjonction**
NEWTON =.]-([. %].) a été utilisée
de la manière suivante : x_{n+1} **=. (f NEWTON** f_p**)** x_n
où les verbes f et f_p représentent une fonction et sa dérivée première,
$x_0, x_1, x_2 \ldots$ est une suite convergeant vers X, racine de l'équation f(X) = 0.
Cette conjonction utilise la formule mathématique (de NEWTON):

$$x_{n+1} = x_n - \frac{f(x_n)}{f_p(x_n)}$$

Son avantage est une convergence rapide (ordre 2) si X est une racine simple.
Son inconvénient est une convergence lente (ordre 1) si la racine X est d'ordre supérieur à 1.

Une formule mathématique (de NEWTON « modifiée ») permet d'améliorer
les performances mais utilise également fs (dérivée seconde) :

$$x_{n+1} = x_n - \frac{f(x_n) f_p(x_n)}{f_p(x_n)^2 + k f(x_n) f_s(x_n)}$$

k= 0 formule de NEWTON « normale »
k= -1 convergence d'ordre 2 pour une racine d'ordre quelconque
k= -0.5 convergence d'ordre 3 pour une racine simple.

Voici une **pro-conjonction** utilisant cette formule de « NEWTON modifiée »:

```
NEWK =.  2 : '-`%`+/y.,(1 1,n.)**/(m.,: 1 1 0{m.)/. y.'
```

Utilisation : x_{n+1} **=. (f `fp`fs NEWK k)** x_n

avec k = 0 , _1 ou _0.5

Remarque :
l'expression f `$f_p$`f_s est un **gérondif** , sorte de « vecteur de verbes » très pratique et puissant en langage **J** (un prochain article sera consacré aux **gérondifs**, « vecteurs et matrices de verbes », et leurs possibilités de programmation).

Exemple d'utilisation de la pro-conjonction NEWK

La fonction $f(x) = 9x^2 - 12xCos(3\pi x) + 2Cos(6\pi x) + 2$ a une racine double : 2/3.

```
f =.   3  :  '(y.*(9*y.)-12*2 o. o. 3*y.)+2*1+2 o. o. 6*y.'
  f   0.666666666666666666                    NB.  verbe f
0
fp =.  3  :  '(18*y.*1+o. 2*1 o. o. 3*y.)-12*(2 o. o. 3*y.)+o. 1 o. o. 6*y.'
  fp   0.666666666666666666          NB.  dérivée première fp
0
fs =.  3  :  '18+1p1 * +/(72,(y.*108p1),_72p1)*(1 2 2) o. (1 1 2)*(3p1)*y.'
  fs   0.666666666666666666          NB.  dérivée seconde fs
18
A =. f `fp`fs   NEWK      NB. A :  pro-adverbe  ;  (k  A) : pro-verbe
(P  =. 9 ! :11)  8       NB. 8 chiffres affichés
```

NB. Demande de 18 itérations ; valeur initiale 0.8

(,.(0 A)^:(i. 18) 0.8);(,.(_0.5 A)^:(i. 18) 0.8);(,.(_1 A)^:(i. 18) 0.8)

0.8	0.8	0.8
0.75742422	0.73866536	0.69035650
0.72955101	0.70429402	0.66201012
0.70997265	0.68491518	0.66582546
0.69593834	0.67458327	0.66664465
0.68589205	0.66971234	0.66666665
0.67882459	0.66774445	0.66666667
0.67401492	0.66703369	0.66666667
0.67089727	0.66678990	0.66666667
0.66899378	0.66670785	0.66666667
0.66790068	0.66668040	0.66666667
0.66730468	0.66667125	0.66666667
0.66699148	0.66666819	0.66666667
0.66683061	0.66666718	0.66666667
0.66674903	0.66666684	0.66666667
0.66670795	0.66666672	0.66666667
0.66668733	0.66666668	0.66666667
0.66667701	0.66666667	0.66666667

On constate :
Pour k = 0 la méthode « classique » de NEWTON a une convergence lente.
Pour k =_0.5 la méthode « modifiée » n'est pas sensiblement meilleure.
Pour k = _1 la méthode « modifiée » est nettement plus performante.
Cela s'explique par le fait que la racine calculée est d'ordre 2.

```
t=3D. 'corde tang ortho oblique ordre3 toujours2 dichot newtondescartes =
E P'
t=3D. t,' OBL ORTHO NEWTON nat NEWT ND DIC DESC '

SCRIPTNAMES=3D.t
   E=3D. 1!:2&2
   P=3D. 9!:11=20
  corde =3D. 3 : 0     NB. Verbe
('ab')=3D.y.
c=3D. a-(f a)*(a-b)%((f a)-(f b))
 ((0< f c)#a),c,((0>f c)#b)
:
[:
)
 =20
  tang =3D. 3 : 0        NB. Verbe
y.-(f y.)%fp y.
:
[:
)

  ortho =3D. 3 : 0       NB. Verbe
y.-(f y.)*(fp y.)%1+(fp y.)^2
:
[:
)
ORTHO =3D. 12 : ']-(x.@])*(y.@])%>:@(*:@(y.@]))' NB. Conjonction

  oblique =3D. 3 : 0     NB. Verbe
y. -(f y.)%pente
:
[:
)
OBL =3D. ]-(([.@])@(%&].))  NB. Conjonction

  ordre3 =3D. 3 : 0 NB. Verbe
y.-(f y.)*(fp y.)%(*:&fp y.)-0.5*(f y.)*(fs y.)
:
[:
)

  toujours2 =3D. 3 : 0   NB. Verbe
y.-(f y.)*(fp y.)%(*:&fp y.)-(f y.)*(fs y.)
:
[:
)

  dichot =3D. 3 : 0      NB. Verbe
('ab') =3D.y.
```

```
 d =3D.f c =3D. 0.5*(a+b)
  ((d>0)#a),c,(d<0)#b=20
 :
[:
)

  newtondescartes =3D. 3 : 0      NB. Verbe
('ab')=3D.y.
(a-(f a)*(a-b)%((f a)-(f b))),(b-(f b)%(fp b))
:
[:
)

nat =3D. 4!:0  NB. 0 : pro-nom          1 : pro-adverbe
              NB. 2 : pro-conjonction  3 : pro-verbe
      NB. NEWTON : Conjonction- Methode de NEWTON=20
NEWTON =3D. ]-([. % ].)      =20
     NB. ND : Conjonction- Methode de Newton Descartes
ND =3D. ]-([.*(-/,1:)%(-/@[.,(].@}.))) =20
     NB. DIC : Adverbe - methode dichotomique
DIC =3D. 1 : ',(~:/"1*u.m)#m=3D.2 2$(-:+/y.),|.y.'
     NB. DESC : Adverbe - Methode de Descartes (cordes)
DESC=3D. 1 : =
'((3=3D#d)#2$c),(2=3D#d)#d=3D./:~c,((*u.y.)~:*u.c=3D.(-/(|.y.)*u.y.)%-/u.=
y.)#y.'
     NB. NEWK : Conjonction - Newton modifie
     NB. (f`f1`f2 NEWK k) x0   avec k =3D 0  _1 ou _0.5
NEWK =3D. 2 : '-`%`+/y.,(1 1,n.)**/(m.,: 1 1 0{m.)/. y.'
```