Exercice 5-1 : Tour d'horizon des différentes méthodes

Pour chaque méthode, nous donnons une ébauche du programme utilisé et des résultats partiels.

a) Dichotomie ou bissection
Nous avons utilisé le programme du texte (listing 5-1) pour obtenir les valeurs ci-dessous.

```
abscisse gauche: 0
abscisse droite: 1
itération      xg              xm              xd
1           0.500000        0.500000        1.000000
2           0.500000        0.750000        0.750000
3           0.500000        0.625000        0.625000
4           0.562500        0.562500        0.625000
5           0.593750        0.593750        0.625000
...
10          0.615234        0.616211        0.616211
...
20          0.615468        0.615468        0.615469
```

Vous voyez que la longueur de l'intervalle qui contient la racine passe de 1 à 2^{-10} à la dixième itération. On admet que la solution est $x^* = 0{,}615468$.

b) *Regula falsi*
Ébauche du programme :

```
TOL = 1E-6;
deff("y = fn(x)","y =  x - 0.2*sin(x) - 0.5")
xg = input("abscisse gauche: ");
xd = input("abscisse droite: ");
fg = fn(xg); fd = fn(xd); fm = (fg+fd)/2;
while(abs(fm) > TOL)
    xm = xg - fg*(xg-xd)/(fg-fd);
    fm = fn(xm);
    if(fm*fg < 0)
      xd = xm; fd = fm;
    else
       xg = xm; fg = fm;
    end
end
xm = xg - fg*(xd-xg)/(fd-fg);
temp = [xm,fn(xm)];
mprintf("\n%12.6f\t%12.6f\t\n",temp);
```

Résultats :

```
abscisse gauche: 0
abscisse droite: 1
itération      xg            xm            xd
  1         0.601174      0.601174      1.000000
  2         0.615040      0.615040      1.000000
  3         0.615455      0.615455      1.000000
  4         0.615468      0.615468      1.000000
```

Vous constatez que l'une des abscisses (x_d ici) reste constante ; c'est un comportement assez général de cette méthode qui se produit chaque fois que la dérivée seconde de f garde un signe constant dans l'intervalle. Divers perfectionnements de cet algorithme ont été proposés, qui assurent une convergence plus rapide en permettant des modifications des deux bornes de l'intervalle (voir l'article *false position method* dans Wikipedia en anglais : `http://en.wikipedia.org/wiki/False_posi tion_method`).

c) Point fixe
La fonction $f(x) = 0{,}2\sin(x) + 0{,}5$ a une dérivée inférieure à 1 en valeur absolue dans tout l'intervalle ; l'algorithme du point fixe peut s'appliquer. Il suffit de répéter l'instruction `x = fn(x)` jusqu'à convergence. À partir de $x_0 = 0$, la valeur $x^* = 0{,}6154682$ est atteinte en 9 itérations.

d) Méthode de Newton
La partie principale du programme s'écrit comme suit :

```
TOL = 1E-6;
deff("y = fn(x)","y =  x - 0.2*sin(x) - 0.5")
deff("y = dfdx(x)", "y = 1 - 0.2*cos(x)")
x = input("valeur initiale: ");
while abs( fn(x)) > TOL
    x =  x - fn(x)/dfdx(x);
end
mprintf("racine: %14.8f\tvaleur de la fonction:
        %14.8f\n",x,fn(x));
```

Voici un exemple d'exécution

```
valeur initiale: 0
itération       x             fn            dfdx
1           0.625000        0.007981        0.837807
2           0.615474        0.000005        0.836700
3           0.615468        0.000000        0.836699
racine:     0.61546817 valeur de la fonction:     0.00000000
```

e) Méthode de la sécante
Comme vous le voyez sur le programme résumé, cet algorithme utilise la même formule que la méthode *regula falsi* ; seul le choix des abscisses successives est différent. Une conséquence est qu'il n'est pas nécessaire de choisir des valeurs initiales encadrant la racine.

```
TOL = 1E-6;
deff("y = fn(x)","y =  x - 0.2*sin(x) - 0.5")
xavder = input("abscisse gauche: ");
xder = input("abscisse droite: ");
favder = fn(xavder); fder = fn(xder);
while abs(xder - xavder) > TOL
    x = xder - fder*(xder-xavder)/(fder-favder);
    f = fn(x);
    favder = fder; xavder = xder;
    xder = x; fder = f;
end
x = xder - fder*(xder-xavder)/(fder-favder) ;
mprintf("racine: %14.8f\tvaleur de la fonction:
        %14.8f\n",x,fn(x));
```

L'exécution nous a donné

```
abscisse gauche: 0
abscisse droite: 0.3
itération    xavder             x             xder
1           0.300000        0.622675        0.622675
2           0.622675        0.615333        0.615333
3           0.615333        0.615468        0.615468
4           0.615468        0.615468        0.615468
      x                fn(x)
  0.61546817         0.00000000
```