1 Introduction

Soit $f(x)$ une fonction relle. On cherche ses racines, c'est--dire les solutions de l'quation

$$f(x) = 0 \tag{1}$$

Nous allons utiliser la *mthode de Newton* qui consiste chercher la solution comme limite de la suite:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\,;\; n = 1, 2, \ldots \tag{2}$$

x_0 tant choisi alatoirement.

2 Mthodes et programme

2.1 Aspects thoriques

L'algorithme issu de la mthode de Newton est le suivant :

Choix alatoire de x_0,
$$x_{n+1} = g(x_n)$$

o

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Il s'agit d'un algorithme itratif qui permet de remplacer le problme initial – recherche de racines de la fonction $f(x)$ – par la recherche de point fixe de la fonction $g(x)$:

$$x = g(x)$$

Comme pour tout algorithme itratif, quelques questions se posent:

(a) Est-ce que la suite (2) converge?

(b) Si la suite x_n converge, est-ce que sa limite est bien la solution recherche de l'quation (1)?

(c) Comment choisir la condition initiale pour garantir la convergence vers la solution de l'quation (1)?

Voici un thorme (voir [1]) qui donne une rponse aux questions (a) et (b).

Soit ξ un point fixe de la fonction $g(x)$:

$$\xi = g(\xi)$$

Si la drive $g'(x)$ est continue et si

$$|g'(\xi)| < 1$$

alors il existe un intervalle $[a, b]$ tel que $\xi \in [a, b]$ et que pour tout $x_0 \in [a, b]$ la suite rcurrente

$$x_0,\ x_{n+1} = g(x_n),\ n = 1, 2, 3, \ldots$$

converge vers ξ.

Revenons la mthode de Newton qui rsout l'quation (1). Soit ξ la solution de l'quation: $f(\xi) = 0$. Alors, ξ est le point fixe de la fonction

$$g(x) = x - \frac{f(x)}{f'(x)}$$

On calcule la drive:

$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$$

Quand $x = \xi$ on a:

$$g'(\xi) = 0 < 1$$

Donc on peut appliquer ici le thorme 2.1.

On remarque cependant que l'intervalle $[a, b]$ de convergence n'est pas dfini explicitement. La question (c) sur le choix de la condition initiale dans l'algorithme reste donc ouverte. Voici un thorme qui donne une prcision utile:

Si $f \in C^2[a, b]$ vrifie les conditions suivantes:

(i) $f(a) \cdot f(b) < 0$

(ii) $\forall x \in [a, b] \quad f'(x) \neq 0$

(iii) $\forall x \in [a, b] \quad f''(x) \neq 0$

alors, en choisissant $x_0 \in [a, b]$ tel que

$$f(x_0)f''(x_0) > 0$$

la suite

$$x_0,\ x_{n+1} = g(x_n),\ n = 1, 2, 3, \ldots$$

converge vers l'unique solution ξde l'quation (1) dans $[a, b]$.

Nous avons appliqu la mthode de Newton au calcul des racines carres de nombres rels (voir [1]). Soit $\alpha > 0$ le nombre dont on cherche la racine carre. Pour ce faire, on doit rsoudre l'quation

$$x^2 - \alpha = 0$$

Alors $f(x) = x^2 - \alpha$.

Les approximations successives de $\sqrt{\alpha}$ par la mthode de Newton s'crivent:

$$x_0, \quad x_{n=1} = x_n - \frac{f(x_n)}{f'(x_n)} = \frac{x_n}{2} + \frac{\alpha}{2x_n}$$

Le programme qui ralise cet algorithme permet de calculer la racine carre d'un nombre positif arbitraire avec une prcision voulue. Le source se trouve en annexe.

En ce qui concerne le choix de l'intervalle l'application du thorme 2.1, dans le cas particulier de la fonction $f(x) = x^2 - \alpha$, permet de dduire les conditions suivantes que doivent tre vrifies par l'intervalle $[a, b]$ et le point initial x_0 :

$$\begin{aligned} a^2 < \alpha < b^2 \\ 0 \notin [a, b] \\ x_0^2 > \alpha \end{aligned} \tag{3}$$

Remarquons que, contrairement au cas gnral, la vrification de ces conditions peut tre facilement programme. Cependant, nous avons dcid de ne pas programmer la vrification de ces conditions du thorme pour permettre la rutilisation de notre programme pour la recherche des racines d'autres fonctions. En effet, nous avons dfini la fonction $f(x)$ et ses drives sous forme de fonctions indpendantes. Le programme principal qui ralise la mthode fait appel ces fonctions. Pour l'utiliser avec d'autres fonctions il suffit de changer les dfinitions dans les fichiers correspondants. Cela offre une plus grande gnralit, mais au prix de choix d'intervalle presque manuel.

Le programme propose l'utilisateur de choisir un intervalle et un point initial et vrifie juste si une racine peut tre trouve dans l'intervalle choisi:

$$f(a)f(b) < 0$$

Dans le cas de rponse ngative l'utilisateur peut modifier son choix ou abandonner.

2.2 Vitesse de convergence

Une fois que la mthode est dfinie et sa convergence est assure nous pouvons nous poser une nouvelle question: quel est le nombre d'itrations faire pour obtenir une prcision de calcul voulue? La rponse cette question dans le cas de la mthode de Newton est donne par le thorme suivant:

Soit ξ une racine simple de $f(x)$:

$$0 = f(\xi), \quad f'(\xi) \neq 0, \quad f''(\xi) \neq 0$$

Soit $[a, b]$ l'intervalle de convergence tabli dans le thorme 2.2. Soit

$$x_0 \in [a, b], \quad x_{n+1} = g(x_n)$$

la suite qui converge vers ξ. Si on note

$$e_n = x_n - \xi$$

l'erreur d'approximation, alors on a pour cette erreur l'estimation suivante

$$|e_{n+1}| \cong \left| \frac{f''(\xi)}{f'(\xi)} \right| \cdot \frac{|e_n|^2}{2}$$

Le dernier thorme montre que la convergence est quadratique. On dit dans ces cas que la mthode rcurrente est d'ordre 2.

La difficult principale de la mthode de Newton est le choix de l'intervalle dans lequel se trouve la racine. L'application du thorme 2.2 chaque fonction particulire aboutit un ensemble des conditions dont il faut vrifier la satisfaction. Dans le cas gnral c'est une tche difficile, parce qu'il faut crire – pour chaque fonction – un programme particulier qui vrifier numriquement ces conditions. En particulier c'est la condition 2 du thorme qui est la plus difficile vrifier. En effet, nous ne pouvons pas parcourir numriquement tous les points d'un

intervalle sur l'axe rel. Donc cette condition peut tre vrifie seulement en un nombre fini de points.

On peut choisir un rseau de points trs dense en supposant que la fonction $f(x)$ ne prsente pas de variations importantes sur de petits intervalles (dj une restriction !). Mais cela impose un temps de calcul important pour effectuer le contrle.

2.3 Programmes

Le programme `racine.sci` est divis en trois parties :

Initialisations diverses.- En particulier nous dfinissons la forme de la fonction f et de ses deux premires drives. Le programme demande aussi l'utilisateur de fournir les valeurs :

- des paramtres de la fonction (ici il s'agit du paramtre α);
- de la valeur du pas de discrtisation `dt` pour le calcul de la fonction (un choix de `dt = 0.001` est, en gnral bon);
- du nombre maximal `maxIter` d'itrations effectuer (en fonction de la nature de la fonction on choisit `maxIter` entre 50 et 1000, voire plus, si ncessaire)

1e partie.- Recherche d'un intervalle dans lequel se trouve un nombre impair de racines, en utilisant le test $f(a)\dot{f}(b) < 0$ ou a et b sont les extrmits de l'intervalle. Ces extrmits sont fournies par l'utilisateur et si le test n'est pas vrifi, alors l'utilisateur est invit donner un nouvel intervalle, ou, ventuellement, de s'arrter.

2e partie.- valuation de la racine de la fonction dans l'intervalle retenu lors de la 1e partie en utilisant l'algorithme itratif expos plus haut.

L'arrt des itrations de cet algorithme peut se faire de deux faons :

- Soit la prcision voulue pour le calcul de la racine est atteinte.
- Soit le nombre d'itrations devient plus grand qu'un nombre maximal d'itrations – `maxIter` – fourni par l'utilisateur.

La valeur de la prcision pour le calcul de la racine n'est pas fournie par l'utilisateur mais calcule par le programme en utilisant le rsultat du thorme 2.2 et la prcision de la machine. Nous devons arrter les itrations ds que $|e_{n+1}| \cong eps$. D'aprs (2.2), nous avons : $|e_n| \cong \left(2 \cdot eps \cdot \left|\frac{f'(\xi)}{f''(\xi)}\right|\right)^{-\frac{1}{2}}$. Du fait que nous ne connaissons pas les valeurs de $f'(\xi)$ et $f''(\xi)$, nous prenons la valeur du rapport gale 1 et nous avons ainsi pour la prcision la valeur

$$precision = \sqrt{2 \cdot eps}$$

Le programme la fin des calculs, cre un fichier de compte rendu, appel ici `newton.crd`, qui contient le dtail des calculs et qui peut tre repris tel quel dans un fichier TEX. De plus le programme cre un graphique qui reprsente la fonction f dans l'intervalle choisi et l'approximation pour le calcul de la racine.

[width=10cm,height=10cm]newton1

Figure 1: Reprsentation graphique des approximations successives. $x_0 = 0.95$, $\alpha = 0.0025$

[width=10cm,height=10cm]newton2

Figure 2: Reprsentation graphique de l'approximation de $\sqrt{2}$. $x_0 = 10$, $\alpha = 2$

Limitations.- 1° Il faut noter, et ceci constitue la principale limitation de la mthode de Newton et, par voie de consquence du programme aussi, que nous ne pouvons calculer que des racines qui sont des nombres relles.
2° Il va de soi, que les paramtres de la fonction doivent aussi tre des nombres rels.

3 Rsultats

Les rsultats du programme pour $\alpha = 0.0025$ sont donns par la table 1 pour diffrentes valeurs de la prcision.

La figure 1 fournit la reprsentation graphique du droulement de l'algorithme.

4 Discussion

Les rsultats obtenus par l'excution du programme, qu'on peut voir sur la table 1 sont corrects.

Afin de tester la robustesse de l'algorithme nous avons excut le programme en prenant $\alpha = 10^{-16}$. Les rsultats donns par la table 2 montrent que les performances de l'algorithme ne se dtriorent pas.

D'autre part pour tester la validit de la mthode de Newton pour le calcul de racines carres, nous avons utilis le programme pour calculer la racine carre de 2. Le tableau 3 et la figure 2 fournissent les rsultats. Nous pouvons constater que l'approximation de $\sqrt{2}$ est excellente.

5 Conclusions

Nous avons appliqu la mthode de Newton pour calculer les racines carres des nombres rels positifs. Nous avons pu constater que cette mthode a de bonnes performances de convergence pour le calcul des racines dans ce cas. Il serait donc intressent de tester la mthode de Newton la localisation des racines des polynmes.

D'autre part il serait aussi intressent de comparer la mthode de Newton pour le calcul de la racine carre avec d'autres mthodes d'analyse numrique.

Exemple :

```
========================================================================

Mthode de Newton pour le calcul des racines de fonction y = xˆ2-alpha

Valeur de alpha         =  2.50000000e-003

Pas de discrtisation    =  1.00000000e-003

Nombre max d'itrations =       1000

Approximation en fonction de la valeur de la prcision

     Prcision     Racine x              f(x)            Nb iter

2.10734243e-008  5.00000000e-002    5.63785130e-018         5

Vitesse de convergence

Prcision =  2.10734243e-008

Itration       x
     1  1.00000000e-001
     2  6.25000000e-002
     3  5.12500000e-002
     4  5.00152439e-002
     5  5.00000023e-002

========================================================================
```

Table 1: Rsultats numriques pour $\alpha = 0.0025$

```
==========================================================================

Mthode de Newton pour le calcul des racines de fonction y = x^2-alpha

Valeur de alpha         =  1.00000000e-016

Pas de discrtisation    =  1.00000000e-005

Nombre max d'itrations =       1000

Approximation en fonction de la valeur de la prcision

     Prcision     Racine x              f(x)            Nb iter

2.10734243e-008  1.45941684e-008    1.12989751e-016        23

Vitesse de convergence

Prcision =  2.10734243e-008

Itration       x
     1  1.00000000e-001
     2  5.00000000e-002
     3  2.50000000e-002
     4  1.25000000e-002
     5  6.25000000e-003
     6  3.12500000e-003
     7  1.56250000e-003
     8  7.81250000e-004
     9  3.90625000e-004
    10  1.95312500e-004
    11  9.76562503e-005
    12  4.88281257e-005
    13  2.44140639e-005
    14  1.22070340e-005
    15  6.10352109e-006
    16  3.05176874e-006
    17  1.52590075e-006
    18  7.62983143e-007
    19  3.81557104e-007
    20  1.90909594e-007
    21  9.57167010e-008
    22  4.83807254e-008
    23  2.52238321e-008

==========================================================================
```


Table 2: Rsultats numriques pour $\alpha = 10^{-16}$

```
========================================================================

Mthode de Newton pour le calcul des racines de fonction y = x^2-alpha

Valeur de alpha         =  2.00000000e+000

Pas de discrtisation    =  1.00000000e-003

Nombre max d'itrations =       1000

Approximation en fonction de la valeur de la prcision

     Prcision     Racine x              f(x)            Nb iter

2.10734243e-008  1.41421356e+000    4.44089210e-016         9

Vitesse de convergence

Prcision =  2.10734243e-008

Itration       x
     1  1.00000000e-001
     2  1.00500000e+001
     3  5.12450249e+000
     4  2.75739214e+000
     5  1.74135758e+000
     6  1.44494338e+000
     7  1.41454033e+000
     8  1.41421360e+000
     9  1.41421356e+000

========================================================================
```

Table 3: Rsultats numriques pour 2

References

[1] M. SCHATZMAN, Analyse numrique, Masson, 1991

A Programme

N.B. Le listing des programmes figure ici titre d'exemple, afin de vous donner une ide de la manire crire des programmes *modulaires* et *suffisamment comments*. Il ne doit pas faire partie de votre rapport. Par contre les programmes, prts fonctionner, doivent tre envoys avec votre rapport.

```
//  RACINES D'UNE FONCTION PAR LA MTHODE DE NEWTON

//  Ce programme permet de calculer la racine carre d'un nombre alpha
//  en utilisant la mthode de Newton.


//************************************************************************
//
//                              Partie Initialisations
//                              ======================
//
//  Saisie des valeurs
//    - du paramtre  alpha de la fonction,
//    - du pas de discrtisation dt,
//    - du nombre d'itrations maximal maxIter,
//
//  Dfinition de la fonction f et de ses deux premires drives fdot et f2dot.
//
//************************************************************************


//  On peut dfinir une fonction et l'enregistrer dans un fichier.
//  Pour pouvoir faire appel  cette fonction on la dclare dans le programme principal
//   l'aide de la commande "getf".
//  Ici on dfinit de cette faon la fonction f(x)=x*x-alpha

   getf('f.sci');

//  Une autre faon de dfinir une fonction, directement dans le corps du programme principal.
//  Ici on dfinit de cette faon les drives 1e et 2e de la fonction f(x).

   deff('y=fdot(x,alpha)','y=2*x');
   deff('y=f2dot(x,alpha)','y=2');

//  On forme la fonction g(x) qui dfini le calcul rcurrent des approximations successives.

   deff('y=g(x,alpha)','y=x-f(x,alpha)/fdot(x,alpha)');
   deff('y=gdot(x,alpha)','y=f2dot(x,alpha)*f(x,alpha)/(fdot(x,alpha)*fdot(x,alpha))');

//************************************************************************

//  Le programme demande  l'utilisateur de saisir certaines informations.
//  Pour cela il existe deux mthodes: en ligne de commande de Scilab ou
//   l'aide d'une fentre de dialog. La fonction lecParams utilise la 1e mthode.

   getf('lecParams.sci');
```

```
   [alpha, dt, maxIter] = lecParams();

   getf('plotGraphe.sci');    //  Fonction pour le trac de la courbe y=f(x).

//************************************************************************
//
//                              Premire Partie
//                              ==============
//
//  Recherche d'un intervalle dans lequel se trouve une racine de la fonction
//
//************************************************************************

   write(%io(2), "Le programme va verifier les conditions d''existence d''une racine "...
                 " dans l''intervalle indiqu");

//  Le choix de l'intervalle  dans lequel le programme va chercher une racine
//  est difficile. Nous avons  laiss ce choix entirement  l'utilisateur.
//  Le programme aide l'utilisateur en lui indiquant si la racine peut tre trouve ou non
//  dans l'intervalle qu'il a choisi.
//  Si la condition f(a)*f(b)<0 n'est pas vrifie, l'utilisateur peut changer d'intervalle
//  autant de fois qu'il le souhaite.
//  le programme prvoit deux possibilits de sortie de cette phase :
//    soit le bon intervalle est trouv  (indicateur "test" positionn  0)
//    soit l'utilisateur dcide d'abandonner   (indicateur "decision" positionn
//                                                    0 par l'utilisateur)



//  Boucle sur les diffrentes valeurs de la prcision des calculs

    write(%io(2), "    ");
    epsilon = sqrt(2* %eps)
    printf("Prcision  = %10.8e\n", epsilon);

    test=1;
    decision = 1;

    [test, decision, a, b, x0] = recInt(test, decision);

    if ~decision
        write(%io(2), "FIN par abandon ");
      else
        write(%io(2), "L''intervalle a t trouv. ");

//************************************************************************
//
//                        Deuxime Partie
//                        ==============
//
//  valuation de la racine de la fonction
//
//************************************************************************
```

```
        [tblPrec, tblConv] = evalRac(a, b, x0, alpha, dt, epsilon, maxIter);

        write(%io(2), "C''est fini! ");


//        Trac du graphe de la fonction

        nuEcran = 0;
        plotGraphe(a, b, alpha, dt, nuEcran);

//  Ouverture du fichier de compte rendu "newton.crd"  en criture

        nomFicCR = 'newton.crd';
        lst = mopen(nomFicCR, 'w');

//  criture du compte rendu

        ficCR(tblConv, tblPrec, lst);

//  Fermeture du fichier de compte rendu "newton.crd"

        file('close',lst);
    end;

//  FIN DU PROGRAMME
//*************************************************************************

function y = f(x,alpha);

//*************************************************************************
//
//   Fonction y = f(x)
//
//  Variables et tableaux :
//                en entre : x     = valeur de la variable indpendante x
//                           alpha = valeur dont on cherche la racine carre
//                en sortie : y     = erreur de l'approximation
//
//*************************************************************************

    y=x**2-alpha;

//*************************************************************************

function [alpha, dt, maxIter] = lecParams();

//*************************************************************************
//
//  Fonction qui permet la saisie des paramtres du problme
//  en utilisant la ligne de commande Scilab
//
//  Variables et tableaux :
//                en sortie : alpha   = valeur dont on cherche la racine carre
//                            dt      = pas de discrtisation
```

```
//                          maxIter = nombre max d'itrations autorise
//
//************************************************************************

//  La fonction "write" permet d'afficher un message sur l'cran
//  La fonction "read" permet de lire  des valeurs saisies au clavier

   write(%io(2), "Donnez le nombre dont vous voulez calcule la racine carr ");
   alpha = read(%io(1),1,1, '(e10.0)');

   write(%io(2), "Quelle pas de discrtisation dt souhaitez vous?");
   dt = read(%io(1),1,1, '(e10.0)');


   write(%io(2), "Entrez un nombre d''itrations  ne pas dpasser ");
   maxIter = read(%io(1),1,1, '(e10.0)');


//************************************************************************

function[test, decision, a, b, x0] = recInt(test, decision)

//************************************************************************
//
//  Fonction qui calcule un intervalle dans lequel se trouve
//  une racine de multiplicit impaire. Dans ce cas
//  la variable test est positionn  1.
//  La recherche peut tre abandonne  la demande de l'utilisateur.
//  Dans ce cas la variable test est positionne  0.
//
//  Variables et tableaux :
//                en sortie : a, b     = les bornes de l'intervalle
//                            x0       = point initial
//      en entre et sortie : test     = variable indiquant si la recherche
//                                      a t abandonne ou non. (N.B. En entre
//                                      test doit tre positionn  1)
//
//************************************************************************

        while test&decision

//           Une autre possibilit d'interaction avec l'utilisateur :
//           une fentre de saisie ralise par la commande "x_mdialog"
//           avec affichage, le cas chant, des valeurs par dfaut.

             data=x_mdialog('Choix  de l''intervalle de recherche',...
             [' Entrez a';' Entrez b';'Point initial';],['0';'1';'0.1';]);

//           Rcupration des donnes de la fentre de dialogue

            D=evstr(data);
            a=D(1);
            b=D(2);
            x0=D(3);
```

```
            if f(a,alpha)*f(b,alpha)>0
                write(%io(2), "La condition n''est pas vrifie. ");
                write(%io(2), "Pour continuer tapez 1 pour abandonner tapez 0 ");
                decision= read(%io(1),1,1, '(e10.0)');
                test=1;
              else write(%io(2), "La condition est  vrifie. ");
                test=0;
            end;

        end;  // Boucle "while"

//***************************************************************************


function [tblPrec, tblConv] = evalRac(a, b, x0, alpha, dt, epsilon,
maxIter)

//***************************************************************************
//
//   Fonction qui calcule la valeur de la racine
//
//  Variables et tableaux :
//                en sortie : a, b     = les bornes de l'intervalle
//                            x0       = point initial
//                            alpha    = valeur dont on cherche la racine carre
//                            dt       = pas de discrtisation
//                            epsilon = valeur de prcision (dpend de la machine)
//                            maxIter = nombre max d'itrations autorise
//                            iterPrec = numro de l'itration.
//                en sortie : tblPrec  = tableau contenant la prcision de la solution,
//                                       la solution, la valeur de la fonction et
//                                       le nombre d'itrations effectues.
//                            tblConc  = tableau contenant la vitesse de convergence
//
//***************************************************************************


//  Calcul des approximations successives de la racine

            x=x0;
            tblConv(1) = x0;
            y1=g(x,alpha);
            iter = 1;
            while (abs(y1-x) > epsilon)&(iter < maxIter)
                x = y1;
                tblConv(iter+1) = x;
//                printf("approximation: x= %10.8e\n",x);
                y1=g(x,alpha);
                iter =iter+1;
            end;
            res = f(y1,alpha);
            tblPrec(1) = epsilon;
            tblPrec(2) = y1;
```

```
            tblPrec(3) = res;
            tblPrec(4) = iter;
            printf("Solution: x= %10.8e, valeur fct = %10.8e, Nb itr = %6.0f, Prec = %10.8e\n", ...
                    y1, res, iter, epsilon);


//************************************************************************

function [] = ficCR(tblConv, tblPrec, lst);

//************************************************************************
//
//   Ecriture du fichier de Compte Rendu
//
//  Variables et tableaux :
//                en entre : tblConv, tblPrec = tableaux de compte rendu
//                           lst              = nom du fichier o sera stock le compte rendu
//
//************************************************************************


    disp('Ecriture du fichier de Compte Rendu. Patientez svp.');
    fprintf(lst,'\\begin{verbatim}  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'========================================================================== \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Mthode de Newton pour le calcul des racines de fonction y = x^2-alpha \n');
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Valeur de alpha         =  %10.8e\n',alpha);
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Pas de discrtisation   =  %10.8e\n',dt);
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Nombre max d''itrations = %10.0f\n',maxIter);
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Approximation en fonction de la valeur de la prcision \n')
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'     Prcision     Racine x              f(x)            Nb iter \n');
    fprintf(lst,'  \n');
    fprintf(lst,'%10.8e  %10.8e    %10.8e    %6.0f \n', tblPrec(1),...
                tblPrec(2), tblPrec(3), tblPrec(4));
    fprintf(lst,'  \n');
    fprintf(lst,'  \n');
    fprintf(lst,'Vitesse de convergence \n')
    fprintf(lst,'  \n');
    fprintf(lst,'Prcision =  %10.8e\n',tblPrec(1));
    fprintf(lst,'  \n');
    fprintf(lst,'Itration       x  \n');
    for i = 1:tblPrec(4)
        fprintf(lst,'%6.0f  %10.8e\n', i, tblConv(i));
```

```
    end;
    fprintf(lst,'  \n');
    fprintf(lst,'=============================================================================  \n');
    fprintf(lst,'  \n');

//***************************************************************************


function [] = plotGraphe(a, b, alpha, dt, nuEcran);

//***************************************************************************
//
//  Trac d'une fonction donn par f(.) dans l'intervalle [a, b]
//
//  Variables et tableaux :
//                en entre : alpha   = valeur dont on cherche la racine carre
//                           dt      = pas de discrtisation
//                           a,b     = extrmits de l'intervalle de variation pour f
//                           nuEcran = numro d'cran pour le trac
//
//***************************************************************************

//  Prparation de  graphes

//  valuation de l'axe des abscises

      t=a:dt:b;
//  Calcul pour chaque point de l'axe des abscises, de l'ordonne correspondante

      for i=1:length(t)
          f_graph(i)=f(t(i),alpha);
      end
      zero_graph=zeros(1,length(t));

//  Initialisation d'une fentre graphique

      xset('window',nuEcran); // cre une fentre et lui associe un numro
      xbasc();
      xselect();

//  La fonction plot2D trace ici deux courbes.
//  Chaque courbe est trace point par point
//  et chaque point est dfini par deux coordonnes : (x,y).
//  Ainsi une courbe est dfinie par deux tableaux : X qui regroupe les coordonnes x
//                                                     de tous les points de la courbe
//                                                  et Y qui regroupe toutes les ordonnes.
//  S'il faut tracer plusieurs courbes on doit indiquer
//  en premier argument de plot2D  tous les tableaux des abscises
//  et en deuxime argument tous les tableaux des ordonnes.
//  Voir l'aide de Scilab pour plus d'informations sur plot2d


      plot2d([t;t]',[zero_graph;f_graph']',[5,2],"161","zero@f(x)",[2,10,2,10])
```

```
//*************************************************************************
}
```