Fiche n^0 6. Résolution d'équations non-linéaires en langage C

Exercice 1 – Méthode de point-fixe

Soit g la fonction qui à x associe $\exp(x) - 3x$.
a) Montrer que l'équation $g(x) = 0$ possède deux solutions, l'une dans $[0, 1]$, et l'autre dans $[\frac{3}{2}, 2]$. (On indique que $\exp(\frac{3}{2}) < \frac{9}{2}$ et que $\exp(2) > 6$).

b) La suite définie par $x^{(0)} = 0$ et $x^{(n+1)} = \frac{\exp(x^{(n)})}{3}$ converge-t-elle vers la solution de l'équation $\exp(x) = 3x$ située dans $[0, 1]$? Cette même méthode itérative permet elle de s'approcher de la solution de cette équation située dans $[\frac{3}{2}, 2]$?

Soit f la fonction définie sur $[0, 1]$ par $f(x) = \cos(x)$. On cherche à calculer une approximation de la solution de l'équation $f(x) = x$.
c) Montrer que cette équation possède une solution unique dans $[0, 1]$, que l'on notera x^*.

Pour approcher x^*, on définit une suite $\left(x^{(n)}\right)$, pour $n \geq 0$ par $x^{(0)} = 1$ et $x^{(n+1)} = f\left(x^{(n)}\right)$.
d) montrer que

$$\left(x^{(n+1)} - x^*\right) = -2 \sin\left(\frac{x^{(n)} + x^*}{2}\right) \sin\left(\frac{x^{(n)} - x^*}{2}\right)$$

e) en déduire la convergence de la suite vers x^*, asymptotiquement comme une suite géométrique de raison $-\sin(x^*)$. Est ce le résultat attendu ?

Solution

a) La dérivée de g est $g'(x) = \exp(x) - 3$, qui ne s'annule qu'une fois ; donc g étant continue, l'équation $g(x) = 0$ possède au plus deux solutions. Mais par ailleurs, $g(0) = 1 > 0$ et $g(1) = e - 3 < 0$ donc comme g est continue, par le TVI g s'annule sur $[0, 1]$. Enfin $g(\frac{3}{2}) = \exp(\frac{3}{2}) - \frac{9}{2} < 0$ selon l'énoncé, et $g(2) = \exp(2) - 6 > 0$ selon l'énoncé, donc comme g est continue, par le TVI g s'annule sur $[\frac{3}{2}, 2]$.

b) La suite proposée est une itération de point-fixe pour la fonction $f(x) = \frac{\exp(x)}{3}$. Si cette suite itérative converge, la limite vérifie $x = f(x)$, qui équivaut à $g(x) = 0$. Or un théorème nous dit que si on a un intervalle $[a, b]$ qui est stable par f, si $|f'(x)| < 1$ sur $[a, b]$ et si $x^{(0)} \in [a, b]$ alors la suite itérative va converger vers le point-fixe situé dans $[a, b]$. Ici on cherche à vérifier que tout ceci est vrai avec $[a, b] = [0, 1]$. On a tout d'abord bien $x^{(0)} = 0 \in [0, 1]$. Ensuite, $|f'(x)| \leq \frac{e}{3} < 1$ sur $[0, 1]$ et enfin, f est croissante et continue sur $[0, 1]$, donc $f([0, 1]) = [f(0), f(1)] = [\frac{1}{3}, \frac{e}{3}] \subset [0, 1]$. En revanche autour de l'autre solution, on a $f'(x) > \frac{\exp(\frac{3}{2})}{3} > 1$ et la suite itérative ne peut pas converger (sauf si bien sûr l'itéré initial est égal à la solution...).

c) L'intervalle $[0, 1]$ est stable par f et sa dérivée y est de module strictement inférieur à 1 ; donc il existe une unique solution à l'équation $f(x) = x$ sur cet intervalle. Et par ailleurs les itérations de point-fixe proposées dans la question suivante vont converger vers cette unique solution, puisque l'itéré initial est dans l'intervalle.

d) Par différence des deux égalités $x^{(n+1)} = f\left(x^{(n)}\right)$ et $x^* = f\left(x^*\right)$ (puisque x^* est point-fixe de f), on obtient :

$$x^{(n+1)} - x^* = \cos(x^{(n)}) - \cos(x^*),$$

et les formulaires trigonométriques usuels fournissent le résultat.

e) On sait que la suite converge vers x^*, donc asymptotiquement, lorsque $x^{(n)}$ est proche de x^*, le terme $\left(\frac{x^{(n)}+x^*}{2}\right)$ tend vers x^* et le terme $\sin\left(\frac{x^{(n)}-x^*}{2}\right)$ se comporte comme $\left(\frac{x^{(n)}-x^*}{2}\right)$; on conclut que $x^{(n+1)} - x^*$ se comporte comme $-\sin(x^*)(x^{(n)} - x^*)$, ce qui indique bien une convergence de la suite comme une suite géométrique de raison $-\sin(x^*)$. Ce qui est attendu selon un théorème vu en cours est bien une convergence asymptotiquement géométrique avec comme raison $f'(x*)$, ce qui correspond bien ici.

Exercice 2 **– Calcul approché de $\sqrt{a}$ par la méthode de Newton**

a) Ecrire l'algorithme de Newton pour la résolution de $x^2 - a = 0$, où a est un réel strictement positif.

b) Lorsque $a = 2$ et que l'itéré initial est $x^{(0)} = 2$, calculer les trois premiers itérés de cette suite sous forme fractionnaire et sous forme décimale approchée ; comparez avec $\sqrt{2}$ dont une valeur approchée à 10^{-10} près est $1,4142135624$.

c) Montrer que $(x^{(n+1)} - \sqrt{a}) = \frac{1}{2x^{(n)}}(x^{(n)} - \sqrt{a})^2$. En déduire que si $x^{(0)} \geq 0$, alors $x^{(n)} \geq \sqrt{a}$ pour tout $n \geq 1$ puis montrer que la suite est décroissante à partir de $n = 2$.

d) En déduire que la suite converge quadratiquement vers $\sqrt{a}$. Est ce le résultat attendu ?

Solution

a) Les itérations de Newton pour trouver un zéro d'une fonction f sont données par $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$ lorsque le dénominateur est non-nul. Ici, cela fournit $x^{(n+1)} = x^{(n)} - \frac{(x^{(n)})^2 - a}{2x^{(n)}} = \frac{1}{2}\left(x^{(n)} + \frac{a}{x^{(n)}}\right)$.

b) $x^{(1)} = \frac{3}{2} = 1,5$, $x^{(2)} = \frac{17}{12} \approx 1,4166666666667$, $x^{(3)} = \frac{577}{408} \approx 1,41421568627$. Au bout de 3 itérations, on a déjà une précision d'environ 2×10^{-6}, qu'on aurait obtenue seulement en environ 19 itérations de la méthode de dichotomie avec comme intervalle de départ $[1, 2]$; en effet $\frac{(2-1)}{2^{19}} \approx 1,91 \times 10^{-6}$.

c) On commence par remarquer selon l'expression de la récurrence donnée en a) que $x^{(n+1)}$ et $x^{(n)}$ seront de même signe. Plaçons nous alors comme suggéré dans le cas $x^{(0)} \geq 0$, pour lequel nous avons donc $x^{(n)} \geq 0$. Nous avons de plus

$$(x^{(n+1)} - \sqrt{a}) = \frac{1}{2x^{(n)}}\left((x^{(n)})^2 + a - 2x^{(n)}\sqrt{a}\right)$$

ce qui fournit le résultat et implique la positivité de $(x^{(n+1)} - \sqrt{a})$ puisque nous avons montré la positivité de $x^{(n)}$. De plus, on a $x^{(n+1)} - x^{(n)} = \frac{1}{2}\left(\frac{a}{x^{(n)}} - x^{(n)}\right)$ qui est bien négatif pour $n \geq 1$ puisque $x^{(n)} \geq \sqrt{a}$ pour $n \geq 1$.

d) La suite est minorée et décroissante et donc convergente. Par ailleurs, on peut majorer $\frac{1}{2x^{(n)}}$ par $\frac{1}{2\sqrt{a}}$ pour tout $n \geq 1$, et on a donc $|x^{(n+1)} - \sqrt{a}| \leq \frac{1}{2\sqrt{a}}(x^{(n)} - \sqrt{a})^2$, ce qui est bien une convergence (au moins) quadratique. C'était le résultat attendu car un théorème vu en cours dit que la méthode de Newton converge quadratiquement vers la solution x^* de $f(x) = 0$ lorsque $f'(x^*) \neq 0$ (à condition que la fonction soit suffisamment dérivable et l'itéré initial suffisamment proche de x^*).

Exercice 3 **– Implémentation en langage C**

On veut trouver des solutions approchées aux équations non-linéaires des Exercices 1 et 2 en utilisant des itérations de point-fixe et des itérations de Newton. Mais on souhaite écrire des procédures qui puissent être réutilisées pour n'importe quelle fonction.

L'idée est de passer la fonction (et sa dérivée pour les itérations de Newton) en paramètre de la procédure de résolution. On propose donc l'implémentation suivante

```
// fichier  main.c

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "newton.h"
#include "f1.h"


int main()
```

```
{

int N;
double eps;
double x0 ;
double xres1;

N = 100;              // ou à lire dans un fichier de données
eps = 1.e-6;          // c.f. TPs précédents
x0=1.0;

printf("Nombre maximal d iterations: %d \n",N);
printf("Tolerance: %f \n",eps);
printf("Itere initial: %f \n",x0);

xres1 = newton(N,eps,x0,f1,deriv_f1); // le nom d'une fonction sert
                                      // aussi de pointeur vers cette fonction

printf("racine carree de deux %lf \n",xres1);


return 0;
}
```

Avec les déclarations suivantes :

```
// fichier newton.h

#ifndef NEWTON_H
#define NEWTON_H

#include <math.h>
#include <stdlib.h>
#include <stdio.h>

double newton(int nb_it_max, double tolerance, double itere_initial,
double (*fonc) (double), double (*deriv_fonc) (double));

// double (*fonc) (double) est un pointeur de fonction retournant
// un double et ayant comme argument un double.
// Plus généralement type_a (*f) (type_b, type_c) est un pointeur
// de fonction retournant une valeur de type type_a et ayant comme argument
// les types type_b et type_c

#endif
```

et

```
// fichier f1.h

#ifndef F1_H
#define F1_H

#include<math.h>
```

```
double f1 ( double );
double deriv_f1 ( double );

#endif
```

Dans le cas particulier du calcul de $\sqrt{2}$*, on aura la définition suivante :*

```
// fichier f1.c

#include "f1.h"

double f1 (double x)
{
 return pow(x,2) - 2.0;
}

double deriv_f1 (double x)
{
return 2.*x;
}
```

Il vous faut donc écrire le fichier `newton.c` *dont on donne les premières lignes :*

```
#include "newton.h"

double newton(int nb_it_max, double tolerance, double itere_initial,
double (*fonc) (double), double (*deriv_fonc) (double))
{
double itere_courant = itere_initial; // initialisation

// C'est ici que vous intervenez, sachant que pour évaluer
// la fonction pour l'argument itere_courant,
// on doit respecter la syntaxe: (*fonc) (itere_courant)
// exemple:


printf("residu %lf \n",(*fonc) (itere_courant));

return itere_courant; // resultat de la procédure
}
```

Dans le fichier `main.c` *utiliser cette procédure pour trouver une solution approchée de l'équation* $\cos(x) = x$*.*

Exercice 4 **– Variables globales**

Si on veut enchainer les calculs de racines carrées, on ne veut pas avoir à récrire des fonctions `f1`*,* `f2`*, ..., mais on ne veut pas non plus utiliser une fonction à deux arguments, pour ne pas avoir à modifier la fonction newton qui prend comme argument des pointeurs de fonction de* $\mathbb{R}$ *dans* $\mathbb{R}$*.*

On peut alors utiliser une variable globale qui va jouer le rôle de second argument de la fonction `f1`*, selon le modèle suivant :*

```
// fichier  main.c

#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>

#include "newton.h"
#include "f1_gen.h"

double a; // variable "globale" qui sera connue des fonctions appelées dans le main.

int main()

{
int N;
double eps;
double x0 ;
double xres1,xres2;

N = 100;              // ou à lire dans un fichier de données
eps = 1.e-6;          // c.f. TPs précédents
x0=1.0;

printf("Nombre maximal d iterations: %d \n",N);
printf("Tolerance: %f \n",eps);
printf("Itere initial: %f \n",x0);

a = 2; // cette valeur de la variable globale sera vue dans f1_gen.
xres1 = newton(N,eps,x0,f1_gen,deriv_f1_gen);
printf("racine carree de deux %lf \n",xres1);

a = 3; // cette valeur de la variable globale sera vue dans f1_gen.
xres2 = newton(N,eps,x0,f1_gen,deriv_f1_gen);
printf("racine carree de trois %lf \n",xres2);

return 0;
}
```

et on a alors

```
// fichier f1_gen.c

#include "f1_gen.h"

double f1_gen (double x)
{
  extern double a; // on indique qu'il existe ailleurs une variable nommée "a"
                   // qui sera utilisée ici
  return pow(x,2) - a;
}

double deriv_f1_gen (double x)
{
return 2.*x;
}
```

Solution