Méthodes itératives

Correction de l'exercice 2 p. 23 (SCILAB) Pour résoudre $x^2 - a = 0$ par un point fixe différent de Newton, on définit la suite récurrente

$$x_{k+1} = x_k - \mu(x_k^2 - a), \quad k \geq 0,$$

où $\mu \in \mathbb{R}$ est un paramètre à choisir.

La fonction Scilab s'écrira (on a rajouté μ en paramètre)

```
function [x,iter]=PointFixeRacineCarree(a,x0,tol,mu)
  x=x0;
  erreur=1;
  iter=0;
  while (erreur>tol) & (iter<100)
    iter=iter+1;
    y=x-mu*(x^2-a);
    erreur=abs(y-x);
    x=y;
    //disp(x)
  end
endfunction
```

Elle donne les résultats suivants (convergence ou divergence selon les valeurs de x_0 ou mu.

```
-->Warning :redefining function: PointFixeRacineCarree



-->[x,iter]=PointFixeRacineCarree(4,3,0.0001,0.1)
 iter  =

    17.
 x  =

    2.0001157

-->[x,iter]=PointFixeRacineCarree(4,3,0.0001,-0.1)
 iter  =

    16.
 x  =

   Inf

-->[x,iter]=PointFixeRacineCarree(4,-3,0.0001,0.1)
 iter  =
```

```
    16.
 x  =

  -Inf

-->[x,iter]=PointFixeRacineCarree(4,-3,0.0001,-0.1)
 iter  =

    17.
 x  =

  - 2.0001157

-->[x,iter]=PointFixeRacineCarree(4,3,0.0001,0.2)
 iter  =

    2.
 x  =

    2.

-->[x,iter]=PointFixeRacineCarree(4,3,0.0001,0.15)
 iter  =

    10.
 x  =

    2.0000561
```

Exercice 4 p. 23 (SCILAB) Programme ordre_cv_ptfixes.sce

```
//ordres de convergence pour Newton et des methodes de point fixe
// pour la resolution de f(x)=0
clear
sol=3;
deff('y=f(x)','y=x.^2-sol^2') //definition de la fonction
//methode de Newton
K=25;//nombre maximal d'iterations
x0=6;
x=x0;y=x0;z=x0;
Tk=[];//initialisation du tableau des abscisses
Tx=[];//initialisation du tableau des ordonnees
Ty=[];
Tz=[];
for k=1:K
    x=x-f(x)/(2*x);//methode de Newton
    y=y-0.1*f(y);//un premier point fixe
    z=z-0.05*f(z);//un deuxieme point fixe
    Tk=[Tk k];
```

```
    Tx=[Tx abs(x-sol)];
    Ty=[Ty abs(y-sol)];
    Tz=[Tz abs(z-sol)];
end
clf
plot2d(Tk',[Tx' Ty' Tz'],logflag='nl',leg='Newton@PF0.1@PF0.05')
xtitle('Methodes de Point fixe', 'k','erreur')
```

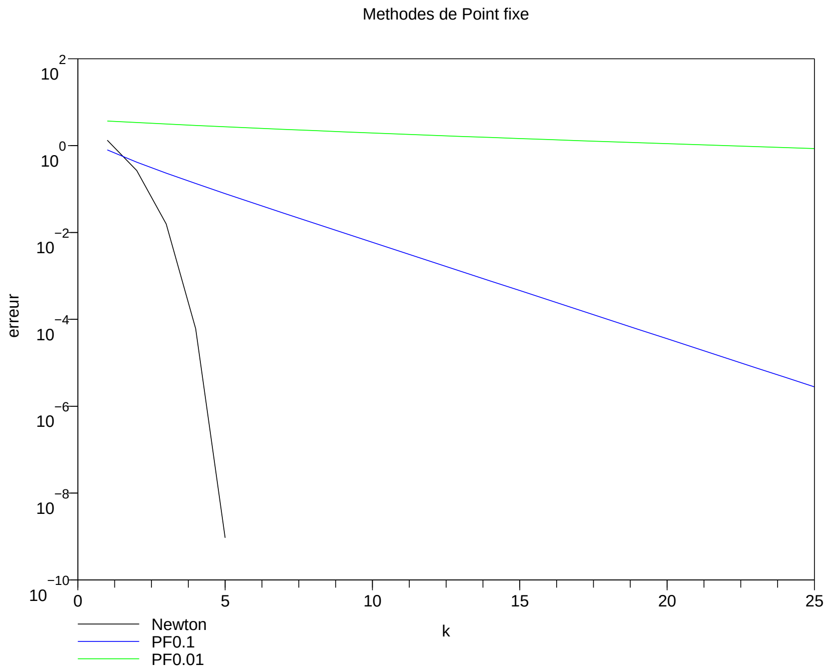

Figure 1: Exécution de ordre_cv_ptfixes.sce

Correction de l'exercice p. 23 (MAPLE)

```
NEWTON
> restart;with(linalg):
Warning, the protected names norm and trace have been redefined and unprotected

> g1:=(x,y)->x**2*y**2;
                                          2   2
                     g1 := (x, y) -> x   y
> f1:=(a,b)->eval(grad(g1(x,y),[x,y]),[x=a,y=b]);f1(x,y);
  f1 := (a, b) -> eval(grad(g1(x, y), [x, y]), [x = a, y = b])
                          [      2      2  ]
                          [2 x y  , 2 x  y]
> Df1:=(a,b)->eval(hessian(g1(x,y),[x,y]),[x=a,y=b]);Df1(x,y);
Df1 := (a, b) -> eval(hessian(g1(x, y), [x, y]), [x = a, y = b])
                           [    2         ]
                           [2 y     4 x y]
                           [              ]
                           [          2 ]
```

```
                         [4 x y  2 x  ]
> Newton:=proc(f,Df,x0,kmax)
> local S,z,k,n;
> n:=nops(x0);
> z:=evalf(x0);
> S:=z;
> for k from 1 to kmax-1 do
> z:= evalf(evalm(z-inverse(Df(seq(z[i],i=1..n)))&*f(seq(z[i],i=1..2)) ) ) ;
> S:=S,[seq(z[i],i=1..n)];
> od;
> RETURN(S);
> end:
> S:=Newton(f1,Df1,[1,2],10);
S := [1., 2.], [0.6666666666, 1.333333333], [0.4444444444, 0.8888888885],

  [0.2962962963, 0.5925925923], [0.1975308643, 0.3950617281],

  [0.1316872429, 0.2633744854], [0.08779149525, 0.1755829902],

  [0.05852766352, 0.1170553268], [0.03901844234, 0.07803688457],

  [0.02601229486, 0.05202458972]
```