# Racines d'équations

## I. Racine d'une équation de la forme f(x)=0

On suppose que f est une fonction continue, et a et b deux réels tels que $f(a)\,f(b)<0$.

### I. 1 Recherche de la racine par dichotomie

Ecrire une procédure qui prend f, a, b et une précision e comme variables d'entrée, et qui détermine par dichotomie une racine de f située entre a et b. On affichera le nombre d'itérations effectuées.

```
> dichotomie:=proc(f,a,b,e)
> local a1,b1,c1,f1,iter,nn;
> a1:=a; b1:=b;
> iter:=evalf(ln((b1-a1)/e)/ln(2));
> for nn from 1 to iter do
> c1:=evalf((a1+b1)/2);
> f1:=f(c1);
> if(evalf(f1*f(a1))<0) then
> b1:=c1;
> else
> a1:=c1;
> end if;
> end do;
> print(evalf(c1),evalf(f1),iter);
> end proc;
```

*dichotomie* := **proc**(*f*, *a*, *b*, *e*)

**local** *a1*, *b1*, *c1*, *f1*, *iter*, *nn*;

*a1* := *a*;

*b1* := *b*;

*iter* := evalf(ln((*b1* − *a1*) / *e*) / ln(2));

**for** *nn* **to** *iter* **do**

*c1* := evalf(1 / 2∗*a1* + 1 / 2∗*b1*);

*f1* := *f*(*c1*);

**if** evalf(*f1*∗*f*(*a1*)) < 0 **then** *b1* := *c1* **else** *a1* := *c1* **end if**

**end do**;

print(evalf(*c1*), evalf(*f1*), *iter*)

**end proc**

Tester la procédure sur les fonctions f(x)=-2+3x, $f(x)=x^3-6x^2+11x-6$

et $f(x)=\frac{1}{2}-\mathbf{e}^{(-10x)}$

en cherchant une racine entre 0 et 5. A-t-on trouvé toutes les racines? (on pourra visualiser la fonction).

```
> ff1:=unapply(-2+3*x,x);
```

$$ff1 := x \to -2 + 3\,x$$

```
> er:=10^(-6); dichotomie(ff1,0,5,er);
```

$$er := \frac{1}{1000000}$$

$$0.6666672229,\ 0.1669\ 10^{-5},\ 22.25349666$$

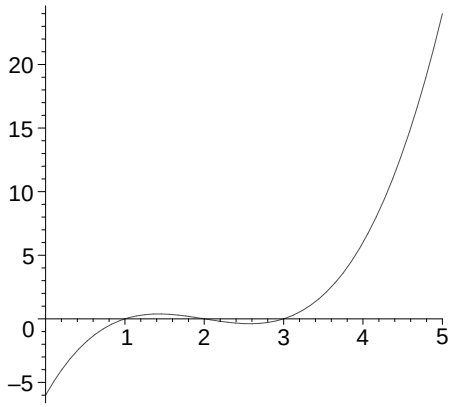
```
> ff2:=unapply(x^3-6*x^2+11*x-6,x);
```

$$ff2 := x \to x^3 - 6\,x^2 + 11\,x - 6$$

```
> dichotomie(ff2,0,5,er);
```

$$3.000000716,\ 0.143\ 10^{-5},\ 22.25349666$$

```
> plot(ff2,0..5);
```




On n'a trouvé qu'une des trois racines.
Il est important avant d'utiliser cette méthode de visualiser la fonction, pour vérifier sa monotonie.

```
> ff3:=unapply(1/2-exp(-10*x),x);
```

$$ff3 := x \to \frac{1}{2} - \mathbf{e}^{(-10\,x)}$$

```
> dichotomie(ff3,0,5,er);
```

$$0.06931424141,\ -0.23832\ 10^{-5},\ 22.25349666$$

```
>
```

## I.2 Méthode la position fausse

Ecrire de meme une procédure qui trouve une racine par la méthode de la position fausse.

```
> fausse:=proc(f,a,b,e)
> local a1,b1,c1,f1,iter;
> a1:=a; b1:=b;
> iter:=0; f1:=f(b1);
> while evalf(abs(f1))>e do
> c1:=evalf(b1-(b1-a1)*f(b1)/(f(b1)-f(a1)));
> iter:=iter+1;
> f1:=f(c1);
> if(evalf(f1*f(a1))<0) then
> b1:=c1;
> else
> a1:=c1;
```

```
> end if;
>
> end do;
> print(evalf(c1),evalf(f1),iter);
> end proc;
```

*fausse* := **proc**(*f*, *a*, *b*, *e*)
**local** *a1*, *b1*, *c1*, *f1*, *iter*;
*a1* := *a*;
*b1* := *b*;
*iter* := 0;
*f1* := *f*(*b1*);
**while** *e* < evalf(abs(*f1*)) **do**
*c1* := evalf(*b1* − (*b1* − *a1*)**f*(*b1*) / (*f*(*b1*) − *f*(*a1*)));
*iter* := *iter* + 1;
*f1* := *f*(*c1*);
**if** evalf(*f1***f*(*a1*)) < 0 **then** *b1* := *c1* **else** *a1* := *c1* **end if**
**end do**;
print(evalf(*c1*), evalf(*f1*), *iter*)
**end proc**

Tester sur les memes fonctions qu'en I.1, et comparant la vitesse de convergence.

```
> fausse(ff1,0,5,er);
```

0.6666666667, 0., 1

```
> fausse(ff2,0,5,er);
```

1., 0., 1

```
> fausse(ff3,0,5,er);
```

$0.06931479593, 0.3894\ 10^{-6}, 17$

Dans le cas de la 3e fonction, essayer de calculer le point intermédiaire avec et sans evalf...
Remarquons que la méthode marche plutot mal dans le dernier cas car la fonction est très loin d'etre linéaire. Mais c'est quand meme mieux qu'avec la dichotomie.

## I.3 Méthode de Newton-Raphson

Cette fois-ci, on donne un seul point a au voisinage de la racine, mais il faut que la fonction f soit dérivable.

Ecrire une procédure qui prend f, a, et une précision e comme variables d'entrée, et qui détermine par la méthode de Newton-Raphson une racine de f située au voisinage de a. On affichera le nombre d'itérations effectuées.

```
> nr:=proc(f,a,e)
> local a1, iter;
> fp:=D(f);
> a1:=a; iter:=0;
> while abs(evalf(f(a1)))>e do
> iter := iter+1;
> a1:=evalf(a1-f(a1)/fp(a1));
```

```
> end do;
> print(a1,f(a1),iter);
> end proc;
'Warning, \'fp\' is implicitly declared local to procedure \'nr\'\n'
```

$nr := \mathbf{proc}(f, a, e)$

**local** $a1, iter, fp$;

$fp := \mathrm{D}(f)$;

$a1 := a$;

$iter := 0$;

**while** $e < \mathrm{abs}(\mathrm{evalf}(f(a1)))$ **do** $iter := iter + 1$; $a1 := \mathrm{evalf}(a1 - f(a1) / fp(a1))$

**end do**;

$\mathrm{print}(a1, f(a1), iter)$

**end proc**

```
> nr(ff1,0,er);
```

$$0.6666666667, 0., 1$$

```
> nr(ff2,0,er);
```

$$1.000000002, 0.2\ 10^{-8}, 6$$

```
> nr(ff2,5,er);
```

$$2.999999997, 0., 7$$

```
> nr(ff3,0,er);
```

$$0.06931471690, -0.58\ 10^{-8}, 4$$

Comparer avec ce que donne solve et fsolve.

```
> solve(ff1(x),x);
```

$$\frac{2}{3}$$

```
> fsolve(ff1);
```

$$0.6666666667$$

```
> solve(ff2(x),x);
```

$$1, 2, 3$$

```
> fsolve(ff2);
```

$$1.000000000, 3.000000000, 2.000000000$$

```
> solve(ff3(x),x);
```

$$\frac{1}{10}\ln(2)$$

```
> fsolve(ff3);
```

$$0.06931471806$$

## II. Racines de polynomes

Trouver toutes les fonctions de maple qui donnent les racines d'un polynome.
Tester sur le polynome suivant:

$$p1 = x^4 + \frac{23\,x^3}{2} - \frac{31\,x^2}{2} - 102\,x + 144$$

```
> p1:=x^4+23*x^3/2-31*x^2/2-102*x+144;
```

$$p1 := x^4 + \frac{23}{2} x^3 - \frac{31}{2} x^2 - 102\,x + 144$$

```
> irreduc(p1);
```

$$false$$

```
> solve(p1,x);
```

$$\frac{3}{2}, -12, -\frac{1}{2} + \frac{\sqrt{33}}{2}, -\frac{1}{2} - \frac{\sqrt{33}}{2}$$

```
> fsolve(p1,x);
```

-12., -3.372281323, 1.500000000, 2.372281323

```
> realroot(p1);
```

[[0, 2], [2, 4], [-8, 0], [-16, -8]]

```
> roots(p1);
```

$$\left[\left[\frac{3}{2}, 1\right], [-12, 1]\right]$$

```
> factor(p1);
```

$$\frac{(x+12)\,(2\,x-3)\,(x^2+x-8)}{2}$$

```
> irreduc(x^2+x-8);
```

$$true$$