Équation quadratique [ss05] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 16 mai 2018

Table des matières

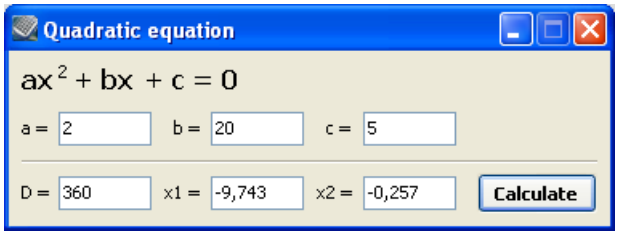
Python - Équation quadratique (Solution)

Mots-Clés Algorithmes paramétrés ■
Requis Structures de base, Structures conditionnelles ■
Difficulté • ○ ○

Objectif
Cet exercice détermine le nombre et les solutions de l'équation quadratique

$$a\,x^2 + b\,x + c = 0$$

en examinant tous les cas, même quand certains coefficients sont nuls.
(image : http://kalkules.com/index.php?page=tools&language=fr)



1 Équation quadratique / pgquadratik

1.1 Équation linéaire

Propriété

Dans $\mathbb{R}$, l'équation linéaire $a\,x + b = 0$ a pour solution :

- Si a n'est pas nul : une solution ($x = -b/a$)
- Sinon si b n'est pas nul : aucune solution
- Sinon (a et b nuls) : l'ensemble $\mathbb{R}$

Écrivez le **profil** d'une procédure `resoudreDegre1(a,b,n,x)` qui résout l'équation linéaire $a\,x + b = 0$ en restituant le nombre de solutions dans `n` (entier) et l'éventuelle solution dans `x` (réel).

Solution Paramètres

Entrants : `a` et `b`

Sortants : `n` et `x`

Écrivez le corps de la procédure.
Dans le cas de l'ensemble $\mathbb{R}$ solution, mettez -1 dans `n`.

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

```
def resoudreDegre1(a, b):
    """  Résout l'équation linéaire a*x+b=0 dans R

     :param a: coefficient de x
     :param b: coefficient constant
     :return: le tuple (nombre de solutions,solution)
    """
    n = None
    x = None
    if a != 0.0:
        n = 1
        x = -b / a
    elif b != 0.0:
        n = 0
    else:
        n = -1
    return (n, x)
```

1.2 Équation quadratique

Propriété

L'équation quadratique $a\,x^2 + b\,x + c = 0$ a pour discriminant :

$$\Delta = b^2 - 4\,a\,c$$

Dans $\mathbb{R}$, ses solutions sont :

- Si Δ est positif : deux racines $(-b \pm \sqrt{\Delta})/(2a)$
- Sinon si Δ est nul : une racine double $(-b)/(2a)$
- Sinon (Δ est négatif) : aucune racine réelle

Écrivez le **profil** d'une procédure `resoudreDegre2(a,b,c,n,x1,x2)` qui résout l'équation quadratique $a\,x^2+b\,x+c=0$ avec `a` **supposé non nul**, en restituant le nombre de solutions dans `n` (entier) et les éventuelles solutions dans `x1` (réel) et `x2` (réel).

Solution Paramètres

Entrants : Les réels `a,b,c`

Sortants : Un entier `n` et les réels `x1,x2`

Écrivez le corps de la procédure.

Outil Python

La fonction racine carrée `sqrt(x)` est définie dans la bibliothèque `math`.

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

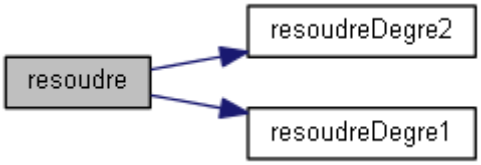
```
def resoudreDegre2(a, b, c):
    """  Résout l'equation quadratique a*x^2+b*x+c=0 dans R (a non nul)

     :param a: coefficient de x^2
     :param b: coefficient de x
     :param c: coefficient constant
     :return: le tuple (nombre de solutions,première solution,deuxième solution)
    """
    n = None
    x1 = None
    x2 = None
    delta = b * b - 4 * a * c
    if delta > 0.0:
        n = 2
        x1 = (-b + math.sqrt(delta)) / (2 * a)
        x2 = (-b - math.sqrt(delta)) / (2 * a)
    elif delta < 0.0:
        n = 0
```

```
    else:
        n = 1
        x1 = (-b) / (2 * a)
    return (n, x1, x2)
```

1.3 Résolution

Déduisez une procédure `resoudre(a,b,c,n,x1,x2)` qui résout l'équation quadratique $ax^2 + bx + c = 0$ dans $\mathbb{R}$ en restituant le nombre de solutions dans `n` (entier) et les éventuelles solutions dans `x1` (réel) et `x2` (réel).



Solution simple

On teste le premier coefficient `a` et on appelle la procédure `resoudreDegre2` (s'il est non nul) ou la procédure `resoudreDegre1` (cas sinon) avec les bons paramètres.

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

```
def resoudre(a, b, c):
    """  Résout l'équation quadratique a*x^2+b*x+c=0 dans R

     :param a: coefficient de x^2
     :param b: coefficient de x
     :param c: coefficient constant
     :return: le tuple (nombre de solutions,première solution,deuxième solution)
    """
    n = None
    x1 = None
    x2 = None
    if a != 0.0:
        n, x1, x2 = resoudreDegre2(a, b, c)
    else:
        n, x1 = resoudreDegre1(b, c)
    return (n, x1, x2)
```

Enfin écrivez une procédure `afficherSolutions(n,x1,x2)` qui, selon le nombre de solutions `n` (entier 2, 1, 0 ou négatif), affiche (où `[x]` désigne le contenu de `x`) :

```
==> Deux racines: [x1] et [x2] # cas n=2
==> Une racine: [x1] # cas n=1
==> Aucune solution dans R # cas n=0
==> L'ensemble R # cas autre
```

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

```
def afficherSolutions(n, x1, x2):
    """  Affiche les solutions de l'équation quadratique dans R

     :param n: nombre de solutions
     :param x1: première solution
     :param x2: deuxième solution
    """
    if n == 2:
        print("==> Deux racines: ", x1, " et ", x2, sep="")
    elif n == 1:
        print("==> Une racine: ", x1, sep="")
    elif n == 0:
        print("==> Aucune solution dans R")
    elif n == -1:
        print("==> L'ensemble R")
```

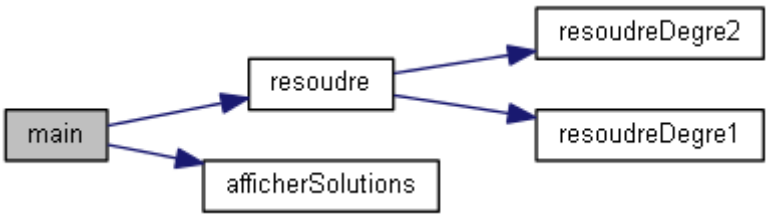
1.4 Programme principal

Écrivez une procédure `test_quadratik1` qui :

- Saisit les coefficients réels d'une équation quadratique.
- Calcule le nombre de solutions et les éventuelles solutions réelles.
- Puis affiche les solutions.

Affichez l'invite :

```
Coefficients a b c de l'équation?
```



Testez le programme sur les trois équations suivantes :

- L'équation : $-2x^2 + 7x + 15$ qui a deux racines réelles : -1.5 et 5
- L'équation : $x^2 - 2x + 1$ qui a une racine double : 1
- L'équation : $x^2 + 3x + 3$ qui n'a pas de racine réelles

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

```
def test_quadratik1():
    """ @test """
    print("Coefficients a b c de l'equation? ", sep="", end="")
    a = float(input())
    b = float(input())
    c = float(input())
    ns, x1, x2 = resoudre(a, b, c)
    afficherSolutions(ns, x1, x2)
```

Cas de solutions très différentes
On pourra consulter l'exercice @[Équation quadratique] (dans les conditionnelles) qui traitent le cas de solutions très différentes. ■

1.5 Résolution dans les complexes

Dans le cas où le discriminant Δ est négatif,

- La partie réelle est donnée par la formule : $-b/(2a)$.
- Et la partie imaginaire par : $\pm(\sqrt{-\Delta}/(2a))i$ où i représente la valeur $\sqrt{-1}$.

Copiez/collez la procédure `resoudreDegre2` en la procédure `resoudreDegre2C`.
Modifiez-la de sorte à résoudre **toutes** les solutions de l'équation quadratique.
Dans ce cas, mettez l'entier -2 dans `n`.

De même, copiez/collez la procédure `resoudre` en la procédure `resoudreC` et modifiez-la de sorte à appeler la procédure de résolution dans $\mathbb{C}$.

Enfin, copier/collez la procédure `afficherSolutions` en la procédure `afficherSolutionsC` et modifiez-la de sorte qu'elle affiche :

```
==> Deux racines sur R: [x1] et [x2] # cas n=2
==> Une racine: [x1] # cas n=1
==> Aucune solution # cas n=0
==> L'ensemble R # cas n=-1 (degré 1)
==> Deux racines sur C: [x1]-[x2]i et [x1]+[x2]i
```

Validez vos procédures avec la solution.

Solution Python @[pgquadratik.py]

```
def resoudreDegre2C(a, b, c):
    """  Résout l'équation quadratique a*x^2+b*x+c=0 dans C

     :param a: coefficient de x^2
     :param b: coefficient de x
```

```
     :param c: coefficient constant
     :return: le tuple (nombre de solutions,partie réelle,partie imaginaire)
    """
    n = None
    x1 = None
    x2 = None
    n, x1, x2 = resoudreDegre2(a, b, c)
    if n == 0:
        delta = b * b - 4 * a * c
        n = -2
        x1 = (-b) / (2 * a)
        x2 = math.sqrt(-delta) / (2 * a)
    return (n, x1, x2)
```

```
def resoudreC(a, b, c):
    """  Résout l'équation quadratique a*x^2+b*x+c=0 dans C

     :param a: coefficient de x^2
     :param b: coefficient de x
     :param c: coefficient constant
     :return: le tuple (nb solutions,1ère sol/partie réelle,2ième sol/partie imaginaire)
    """
    n = None
    x1 = None
    x2 = None
    if a != 0.0:
        n, x1, x2 = resoudreDegre2C(a, b, c)
    else:
        n, x1 = resoudreDegre1(b, c)
    return (n, x1, x2)
```

```
def afficherSolutionsC(n, x1, x2):
    """  Affiche les solutions de l'équation quadratique dans C

     :param n: nombre de solutions
     :param x1: première solution ou partie réelle
     :param x2: deuxième solution ou partie imaginaire
    """
    if n == -2:
        print("==> Deux racines sur C: ", x1, "-", x2, "i et ", x1, "+", x2, "i", sep="")
    else:
        afficherSolutions(n, x1, x2)
```

Copiez/collez la procédure `test_quadratik1` en la procédure `test_quadratik2` puis modifiez-la de sorte à appeler les procédures de résolution dans $\mathbb{C}$.

Testez avec $(a,b,c) = (2,6,1)$, $(3,3,0)$, $(1,3,1)$, $(0,12,-3)$, $(3,6,3)$ et $(2,-4,3)$.

Validez votre procédure avec la solution.

Solution Python @[pgquadratik.py]

```
def test_quadratik2():
    """ @test """
    print("Coefficients a b c de l'equation? ", sep="", end="")
    a = float(input())
    b = float(input())
    c = float(input())
    ns, x1, x2 = resoudreC(a, b, c)
    afficherSolutionsC(ns, x1, x2)
```

2 Références générales

Comprend [Chappelier-CPP1 :c02 :et01], [Maunoury-AL1 :c05 :ex05], [Lemaitre-CC1 :c5 :ex1], [Gottfried-CC1 :c7 :ex39..ex42] ■