

La Méthode De Newton-Raphson

HILAIRE FERNANDES

hilaire@drgeo.eu

3 juin 2019

Résumé

La méthode Newton-Raphson est un algorithme de recherche de zéro d'une fonction réelle par approximations linéaires successives. Après avoir discuté de ses ramifications historiques et de ses principes mathématiques, nous en proposons une implémentation interactive dans un environnement adapté de géométrie interactive et de programmation. Nous concluons ensuite sur sa pertinence dans le cadre de l'enseignement en secondaire II.

1. Introduction

La méthode de Newton-Raphson[3] est un procédé très efficace dans la recherche du zéro d'une fonction réelle, sa convergence est généralement bien plus rapide que celle de la méthode dichotomique. Elle comporte cependant des chausse-trappes qu'une étude de la fonction et de sa courbe permettent d'éviter.

Elle est attribuée aux mathématiciens Newton et Raphson pour leur contribution respective en 1685 et 1690. Elle est toutefois le fruit d'une lente maturation au cours des ans, avant et après l'époque de Newton. Son procédé définitif tel qu'utilisé aujourd'hui est attribué au mathématicien Thomas Simpson en 1740. Avant, elle est dérivée des travaux du mathématicien François Viète et des mathématiciens iraniens Sharaf al-Din al-Tusi et Jamshīd al-Kāshī. Enfin elle est une généralisation de la méthode de Héron ou méthode babylonienne d'extraction d'une racine carrée.

2. Méthode de Héron

Cette technique extrait la racine carrée d'un nombre positif a . A savoir résoudre l'équation $x^2 = a$ ou rechercher le zéro de la fonction $f(x) = x^2 - a$. La méthode est avant tout géométrique, il s'agit de rechercher par itérations successives des rectangles de plus en plus proche d'un carré et d'aires toujours égales à a .

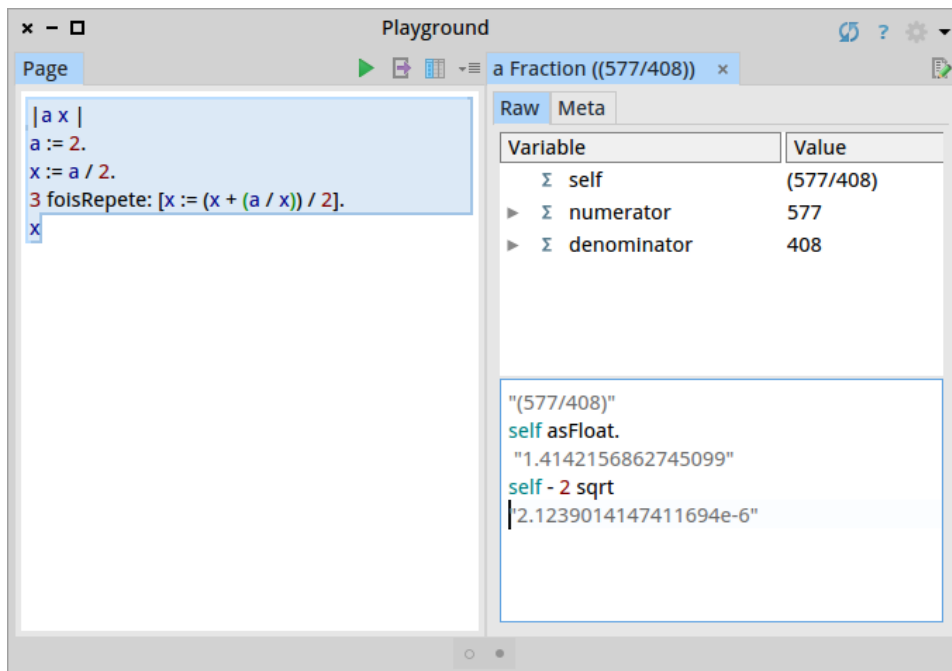
Ainsi si un premier rectangle est choisi avec une dimension x , sa deuxième dimension est obligatoirement $\frac{a}{x}$. Pour le rectangle suivant, et pour se rapprocher du carré, il est choisi

comme première dimension la moyenne des dimensions du rectangle précédent, à savoir : $\frac{x + \frac{a}{x}}{2}$. Cela nous donne une suite de la forme $x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right)$ convergente vers \sqrt{a} .

Plus pratiquement, il est aisé d'écrire un petit programme en Smalltalk depuis l'espace de travail de Dr. Geo[1] :

```
| a x |
a := 2.
x := a / 2.
3 foisRepete: [ x := (x + (a / x)) / 2].
x
```

Cadeau, en exécutant le programme, nous obtenons une approximation sous forme de fraction.



Variable	Value
Σ self	(577/408)
▶ Σ numerator	577
▶ Σ denominator	408

```
"(577/408)"
self asFloat.
"1.4142156862745099"
self - 2 sqrt
|2.1239014147411694e-6"
```

Figure 1. Approximation à 10^{-5} dès la 3ème itération.

Ce qui serait marrant c'est de visualiser cette suite de rectangles convergents vers un carré. Quelques légers ajouts pour construire les rectangles suffisent. Une variable supplémentaire y – 2^{ème} dimension du rectangle – est ajoutée pour plus de clarté :

```
| a x y figure |
a := 2.
x := a / 2.
y := a / x.
figure := DrGeoFigure nouveau pleinEcran afficherAxes.
(figure polygone: { 0@0 . x@0 . x@y. 0@y}) nommer: x.
3 foisRepete: [
x := (x + (a / x)) / 2.
y := a / x.
(figure polygone: { 0@0 . x@0 . x@y. 0@y}) nommer: x asFloat]
```

On observe visuellement que la convergence vers une forme carrée est très rapide, même en étant méchant avec une valeur initiale de 0,5 pour x .

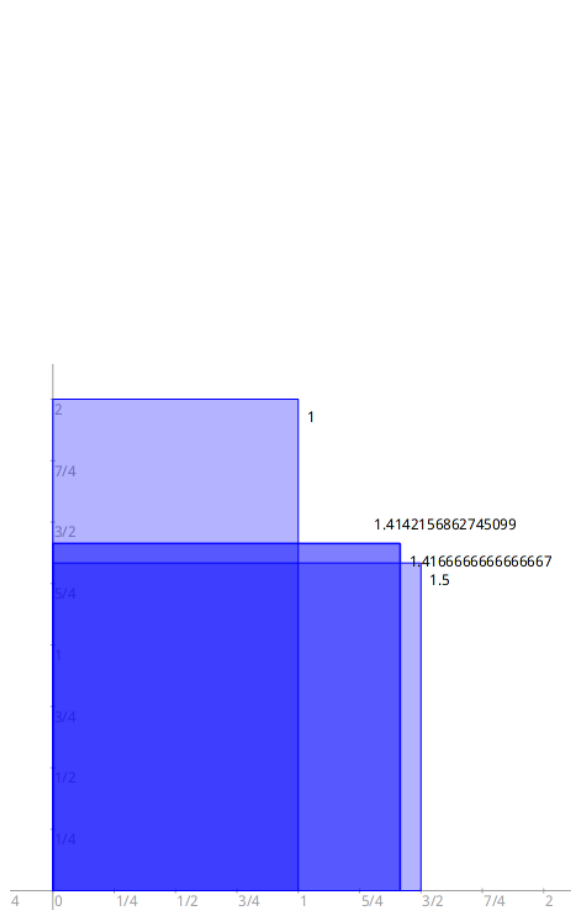


Figure 3. Valeur initiale de 1 pour x .

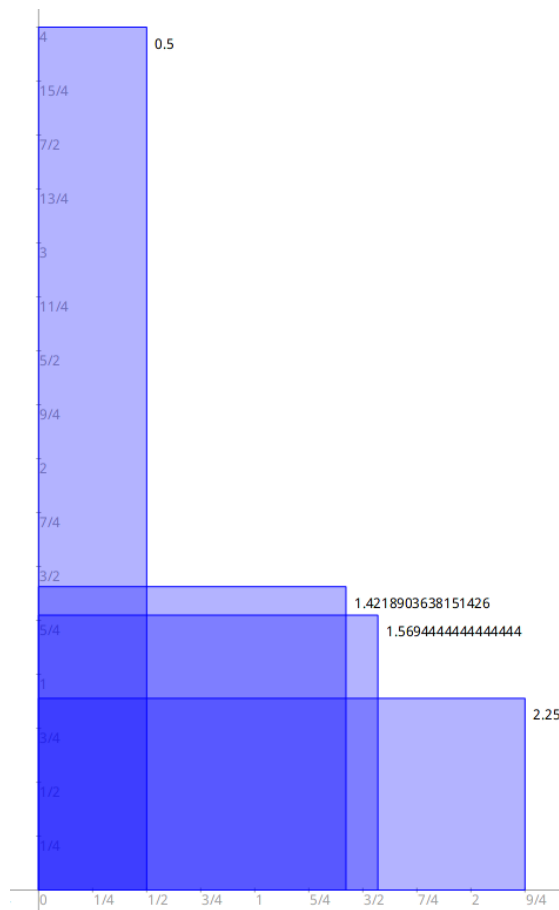


Figure 2. Valeur initiale de 0,5 pour x .

C'est déjà plus visuel mais ce n'est pas très dynamique. Dans la figure finale, il serait encore plus amusant de faire varier dynamiquement la valeur de a , ainsi que la valeur initiale de x pour bien observer comment fonctionne la convergence selon les valeurs de départ. Pour ce faire, il est nécessaire de torturer un peu le programme en utilisant massivement des points – sommets des rectangles – dont les coordonnées sont définies par des blocs de code – bouts de programme. C'est ce qui rend dynamique la figure produite.

```
| a b p1 p2 p3 figure |
figure := DrGeoFigure nouveau pleinEcran afficherAxes.
a := figure entier: 2 a: 0@ -1 min: 2 max: 10 nom: 'a' afficherValeur:
true.
b := figure decimal: 1 a: 0@ -2 min: 0.1 max: 10 nom: 'valeur
initiale' afficherValeur: true.
p2 := figure point: [
| x y |
x := b valeur.
y := a valeur / x.
x @ y].
```

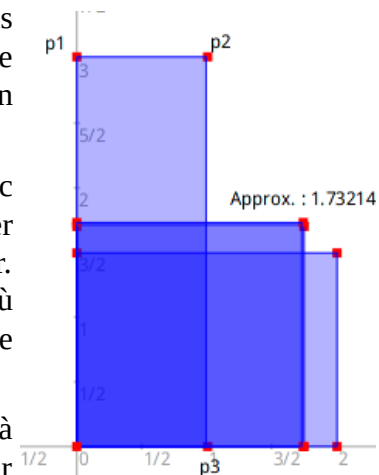
```

p1 := figure point: [ :parent | 0 @ parent y] parent: p2.
p3 := figure point: [ :parent | parent x @ 0] parent: p2.
figure polygone: { 0@0 . p1 . p2 . p3}.
3 foisRepete: [
  p2 := figure point: [ :parent |
    | x y |
    x := (parent x + (a valeur / parent x)) / 2.
    y := a valeur / x.
    x @ y] parent: p2.
  p1 := figure point: [ :parent | 0 @ parent y] parent: p2.
  p3 := figure point: [ :parent | parent x @ 0] parent: p2.
  figure polygone: { 0@0 . p1 . p2 . p3}].
figure point: [
  p2 nommer: 'Approx. : ', p2 x asString.
  0@0 ]

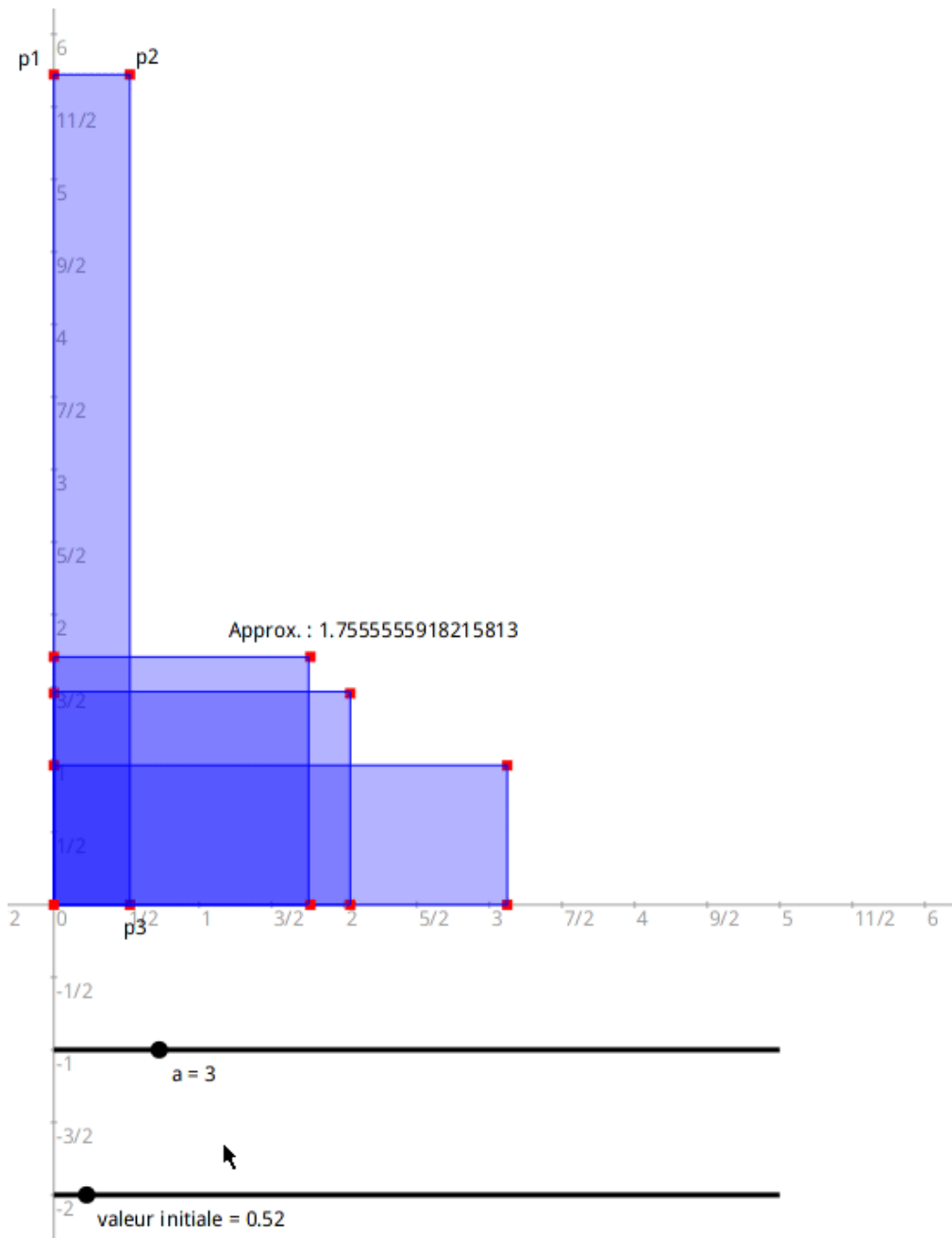
```

Quelques commentaires sur ce programme :

- Les objets **a** et **b** sont des réglettes. La première pour choisir dynamiquement la valeur a dont nous souhaitons la racine carrée. La deuxième pour choisir dynamiquement la valeur initiale, le terme x_0 de la suite. Pour obtenir la valeur d'une réglette il suffit de lui demander en lui envoyant le message **#valeur** : par exemple **a valeur** ou **b valeur**.
- Les sommets **p2**, **p1** et **p3** sont définis les uns dépendamment des autres. Le sommet **p2** est celui qui porte la suite de proche en proche et les sommets **p1** et **p3** en dépendent simplement par un truchement des coordonnées.
- En dehors de la boucle, le sommet **p2** est défini par son bloc de code qui interroge les valeurs des objets **a** et **b** pour fixer le terme initial – x , largeur du rectangle – et y , sa hauteur. Les sommets **p1** et **p3** sont définis à partir de **p2**, d'où l'utilisation du message **#point:parent:** avec **p2** comme argument parent.
- Dans la boucle **#foisRepete**, **p2** est défini itérativement à partir de l'objet **p2** précédent – argument parent – pour récupérer le terme x_n de la suite et calculer le terme suivant x_{n+1} .
- En dehors de la boucle, la dernière instruction est une astuce pour un affichage dynamique de l'approximation obtenue au dernier terme de la suite.



La figure résultante donne la liberté de modifier la valeur entière a dont nous souhaitons une approximation de la racine carré mais aussi la valeur initiale choisie pour x . En jouant avec cette dernière, nous observons combien elle influe sur la convergence et l'importance de ne pas la choisir n'importe comment.



Dans la section suivante nous nous intéressons à la méthode de Newton-Raphson et comment la méthode de Héron en est un cas particulier.

3. Méthode de Newton-Raphson

D'un point de vue géométrique cette méthode revient à substituer une série de droites à la courbe de la fonction dont nous recherchons un zéro. Ces droites sont les tangentes, meilleures approximations locales de degré 1 à la courbe. Un choix important est celui de la première tangente. Pour quelle abscisse x_0 celle-ci sera-t-elle choisie ? Comment ce choix influe-t-il sur la convergence vers le zéro de la fonction.

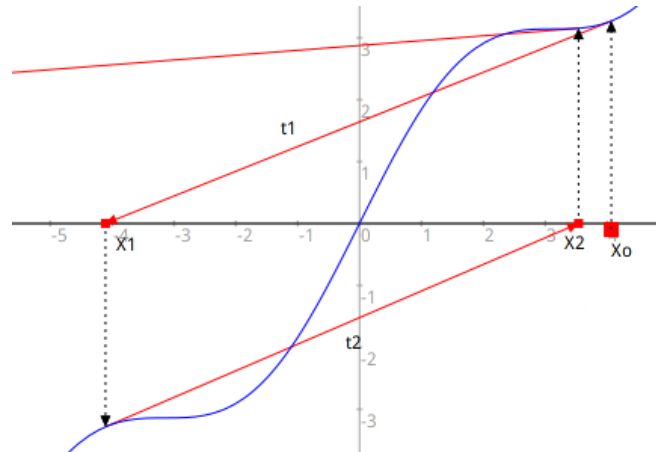


Figure 5: Convergence et divergence sont les dilemmes de la méthode !

Dans la figure ci-dessus nommons f la fonction tracée. A partir d'une première approximation x_0 de la racine recherchée, nous déduisons l'équation de t_1 :

$$y = f'(x_0)(x - x_0) + f(x_0)$$

L'approximation x_1 est déduite comme point d'intersection de t_1 avec l'axe des abscisses :

$$0 = f(x_0) + (x_1 - x_0)f'(x_0)$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Nous généralisons par une suite récurrente :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Dans la cas de la méthode de Héron, la fonction dont le zéro est recherché est :

$$f(x) = x^2 - a$$

Sa fonction dérivée est :

$$f'(x) = 2x$$

La suite récurrent devient donc :

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n}$$

$$x_{n+1} = x_n - \frac{x_n}{2} + \frac{a}{2x_n}$$

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right)$$

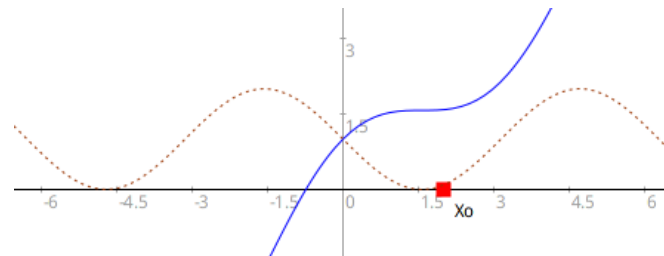
Nous retrouvons bien la méthode de Héron vue en début d'article.

Pour rendre visuel l’algorithme, nous avons besoin d’une fonction. Nous choisissons $f(x) = \cos(x) + x$, elle est amusante car elle contient des chausse-trapes et admet une seule racine. Nous souhaitons bien sûr afficher sa courbe.

Nous avons également besoin de disposer de sa fonction dérivée, il serait aisé de la déterminer algébriquement mais nous nous contentons de la fonction dérivée du pauvre qui nous rend tout de même bien service et ne demande aucun effort lors de modifications ultérieures de la définition de f . Son écriture en code Smalltalk dérouté le lecteur non habitué. En effet il n’y a pas d’opérateurs arithmétiques dans ce langage mais uniquement des messages binaires évalués de gauche à droite, la multiplication est donc bien évaluée après la soustraction. Pour la clarté de lecture il est possible d’écrire des parenthèses autour de la soustraction.

```
| figure f df X0 Xn Mn|
figure := DrGeoFigure nouveau pleinEcran afficherAxes.
f := [:x | x cos + x ].
df := [:x | (f valeur: x + 1e-8) - (f valeur: x) * 1e8].
figure courbeDe: f de: -20 a: 20.
(f figure courbeDe: df de: -20 a: 20) pointille couleur: Color brown.
X0 := 2.
Xn := (figure point: X0@0) large; nommer: 'Xo'.
```

Petit raffinement parfaitement inutile, nous affichons également la courbe de sa fonction dérivée.



Quelques commentaires sur le code :

- Une fonction est représentée à l’aide d’un bloc de code ayant un argument, ici x .
- Le code $x \text{ cos}$, se comprend comme l’envoi du message $\#COS$ à l’objet x pour lui demander son cosinus.
- Comprendre la définition de la dérivée dans le bloc de code df demande de connaître les priorités des messages Smalltalk, voir ressource en annexe[2].
- Le point Xn porte dans son abscisse la suite des approximations x_n de la racine.
- Le point Mn – non utilisé dans le code précédent – représente les différents points de la courbe où la tangente est déterminée.

Pour construire la série, nous complétons le code avec une boucle construisant les sommets Xn et Mn :

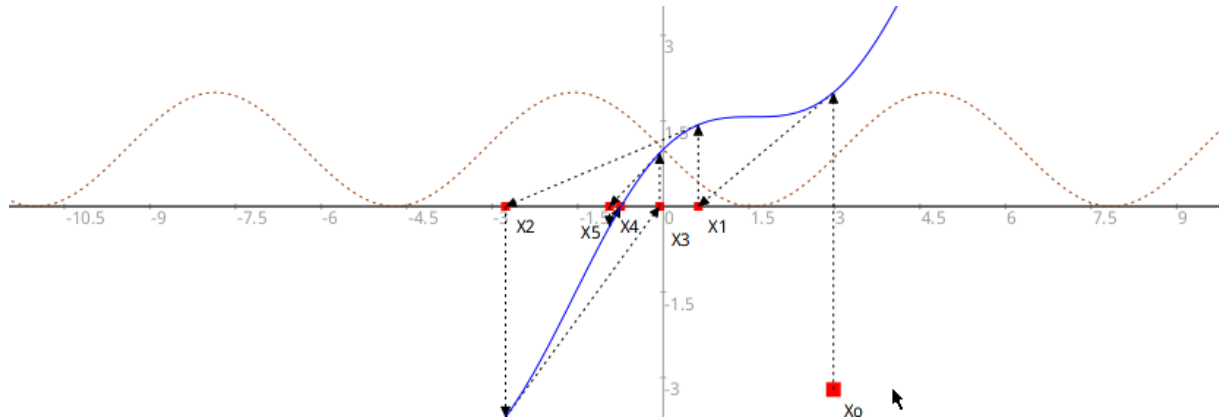
```
1 a: 5 faire: [ :n |
Mn := figure
point: [ :point | point x @ (f valeur: point x)]
parent: Xn.
Xn := figure
point: [ :point |
point x - ( (f valeur: point x) / (df valeur: point x) ) @ 0]
```



```
parent: Mn.  
xn nommer: 'X', n asString]
```

Afin d'explorer interactivement le comportement de la méthode, les différents points sont définis par des blocs de code calculant leurs coordonnées. Ces blocs sont paramétrés par le terme précédent dans la séquence de l'algorithme. Pour suivre le comportement de l'algorithme, nous construisons des segments fléchés reliant les points X_n et M_n .

Dans la figure finale, en attrapant et en déplaçant le point X_0 , nous pouvons explorer dynamiquement le comportement de l'algorithme. Certaines situations sont cauchemardesques.



4. Conclusion

Dans notre étude sur la méthode de Newton-Raphson, nous nous sommes intéressés en premier lieu à celle de Héron pour le calcul d'une racine carrée. Celle-ci présente l'avantage de se transcrire très facilement en un programme de 5 lignes parfaitement abordable pour les élèves. Pour rendre compte de sa dimension géométrique, nous avons ensuite enrichi le programme avec la construction des rectangles successifs convergents vers le carré solution. Les modifications apportées sont encore parfaitement accessibles aux élèves. Enfin pour explorer dynamiquement la convergence de l'algorithme selon ses paramètres initiaux – approximation de départ et carré de la racine – le programme fut à nouveau modifié. L'utilisation de code embarqué dans les objets sommets des rectangles est un petit saut conceptuel, mais il n'est pas alourdi par la difficulté de la récurrence déjà abordée dans les versions précédentes du programme.

La méthode de Newton-Raphson, généralisation de la méthode de Héron, a été programmée avec une approche analytique et géométrique. Encore une fois l'utilisation de point géométrique avec code embarqué permet la conception d'une figure dynamique pour explorer le fonctionnement de l'algorithme.

Pour son enseignement des mathématiques, nous avons montré comment le professeur peut produire en quelques lignes de code des documents dynamiques pour illustrer des savoirs. En France, cet algorithme figure au programme d'enseignement des mathématiques de 1^{ère} : « Exemple d'algorithme : méthode de Newton, en se limitant à des cas favorables ». A Genève, il s'enseigne au collège dans le cours d'applications mathématiques.

Dans le cadre de l'enseignement de la programmation, cette approche progressive partant de la méthode de Héron pour aboutir à celle de Newton-Raphson constitue un fil conducteur pour un travail avec les élèves. Plusieurs concepts de programmation sont introduits

progressivement tout en faisant des mathématiques : variable, boucle, fonction anonyme – bloc de code, utilisation de l'abstraction de la programmation objet.

L'utilisation de l'objet informatique bloc de code, aussi appelé fonction anonyme, donne cette possibilité créative d'enrichir les programmes et figures produites avec un comportement dynamique. Daniel Ingalls principal architecte du langage Smalltalk le définit ainsi : « *L'objectif du projet Smalltalk est de fournir un support informatique pour l'esprit créatif en chacun de nous.* »[4]. Puisse-t-il en être ainsi.

ANNEXE A

Programme complet d'une figure interactive montrant la méthode de Newton-Raphson.

```
| figure f df X0 Xn Mn|
figure := DrGeoFigure nouveau pleinEcran afficherAxes.
f := [ :x | x cos + x ].
df := [ :x | (f valeur: x + 1e-8) - (f valeur: x) * 1e8].
figure courbeDe: f de: -20 a: 20.
(f figure courbeDe: df de: -20 a: 20) pointille couleur: Color brown.
X0 := 2.
Xn := (figure point: X0@0) large; nommer: 'Xo'.
1 a: 5 faire: [ :n |
  Mn := figure
    point: [ :point | point x @ (f valeur: point x)]
    parent: Xn.
  Mn cacher.
  (figure segmentDe: Xn a: Mn) pointille flecheFin.
  Xn := figure
    point: [ :point |
      point x - ( (f valeur: point x) / (df valeur: point x) ) @ 0]
    parent: Mn.
  (figure segmentDe: Mn a: Xn) pointille flecheFin.
  Xn nommer: 'X', n asString]
```

Ressources

- [1] [Dr. Geo](#), environnement adapté de géométrie interactive et de programmation
- [2] [Syntaxe Smalltalk](#) (dont messages)
- [3] [The Newton-Raphson Method](#)
- [4] [Design Principles Behind Smalltalk](#)