T³ EUROPE

Teachers Teaching with Technology™

T³ Europe webinar series 2022

Sharing Inspiration 2022 - A touch of STEM

Numerical Methods of Solving Equations to Develop Computational Thinking and Math Learning

Raul Aparício Gonçalves

Math Teacher at Secondary School of Ermesinde and T3 member - Portugal

t3raul@gmail.com

TEN SKILLS FOR THE FUTURE WORKFORCE

1. Sense-making
2. Social Intelligence
3. Novel and adaptive Thinking
4. Cross-cultural competency
5. Computational Thinking
6. New Media Literacy
7. Transdisciplinarity
8. Design Mindset
9. Cognitive Load Management
10. Virtual Collaboration

DEFINITION: *ability to translate vast amounts of data into abstract concepts and to understand data-based reasoning*

5. Computational Thinking

As the amount of data that we have at our disposal increases exponentially, many more roles will require computational thinking skills in order to make sense of this information. Novice-friendly programming languages and technologies that teach the fundamentals of programming virtual and physical worlds will enable us to manipulate our environments and enhance our interactions. The use of simulations will become a core expertise as they begin to feature regularly in discourse and decision-making. HR departments that currently value applicants who are familiar with basic applications, such as the Microsoft Office suite, will shift their expectations, seeking out resumes that include statistical analysis and quantitative reasoning skills.

In addition to developing computational thinking skills, workers will need to be aware of its limitations. This requires an understanding that models are only as good as the data feeding them—even the best models are approximations of reality and not reality itself. And second, workers must remain able to act in the absence of data and not become paralyzed when lacking an algorithm for every system to guide decision making.

Scratch is an interactive learning environment developed by Lifelong Kindergarten Group at the MIT Media Lab. It teaches young people the fundamentals of computational methodology in a fun, low risk environment.

DIGITAL TECHNOLOGY

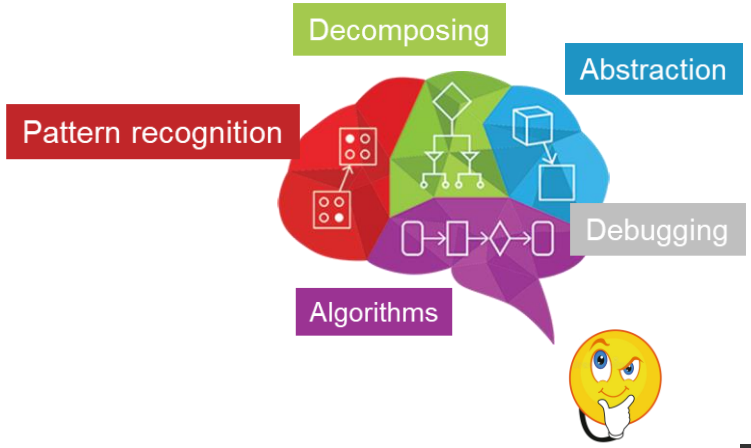
COMPUTATIONAL THINKING

PROGRAMMING

Involves solving problems that are likely to have solutions that can be operationalized through a computer

While learning maths

Computational Thinking is in the math curriculum with the specific aim of improving students performance and increasing their interest in mathematics



Davies, A., Fidler, D., & Gorbis. D.
https://legacy.iftf.org/futureworkskills/

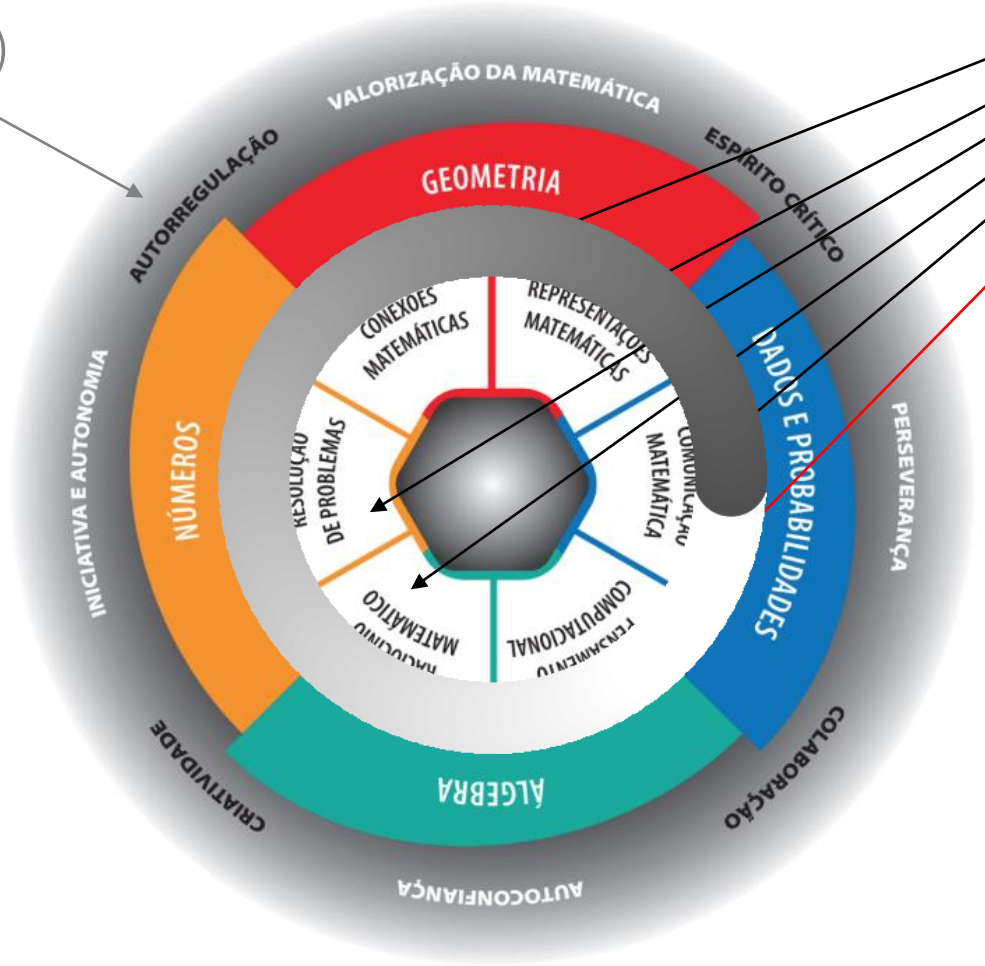
MAKING THE FUTURE WITH FORESIGHT

Math Skills (6)

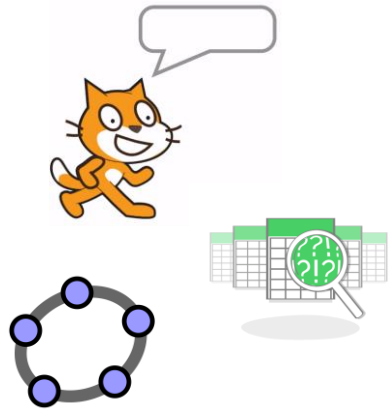
General Skills (8)

Geometry

Numbers



Computational Thinking



Etc...

Data and Probabilites

Algebra

MATH CURRICULUM IN

SECONDARY SCHOOL 15 – 17 years old Comming "soon" - 2024

"Clarifying notes are included for each theme, namely with regard to the suggestion of: activities for the development of Computational Thinking, using examples; proposals for possible deepening of some themes or alternative approaches; bibliographical references that include documents and resources to support the teacher's work."

from the public discussion draft

Aproximated solving equations			
Bolzano-Cauchy Theorem	To know the intermediate value theorem (Bolzano-Cauchy).	To present as an objective the resolution by approximate methods of any equation, taking into account that there are many situations in which it is not possible to obtain an analytical resolution with an exact solution. Promote the systematic use of technology in the application of the different methods listed for obtaining approximate solutions to equations Propose the development of programs in Python to implement Newton-Raphson's and Bisection's methods. Illustrate from a concrete example the geometric interpretation of Newton-Raphson´s method.	
Bisection Method	To know and apply the Bissection method.		
Newton-Raphson Method	To know and apply the Newton-Raphson method.		

from the public discussion draft

Computational Thinking

When working with algorithms, students should be encouraged to practice rigor and systematic verification and control practices. It will be important to promote abstraction in students, encouraging them to collect essential information for solving the proposed task (or situation). Students should be encouraged to identify the important elements in the algorithm creation process and to establish order among them. The recognition of patterns in the task (or situation) presented or in similar problems, previously solved, may contribute to facilitate the structuring of the algorithm to be developed. Before writing the program in the Python language, it is convenient to describe the algorithm in natural language.

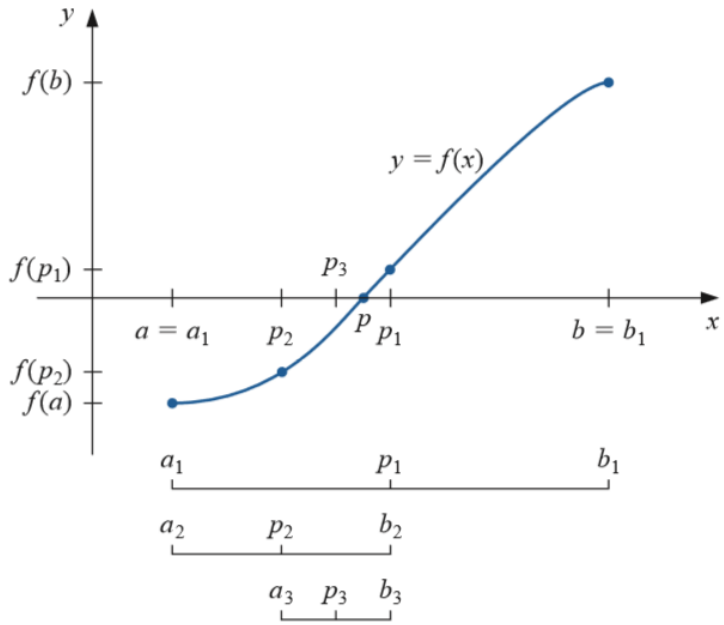
The Bisection method is analogous to the binary search method on an ordered list and, among the interval section methods, it is the one that, in general, produces better estimates for an equal number of evaluations of the function.

from the public discussion draft

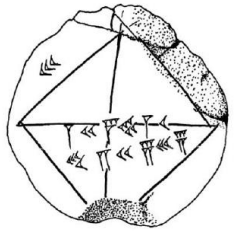
SECONDARY SCHOOL 15 – 17 years old Comming "soon" - 2024

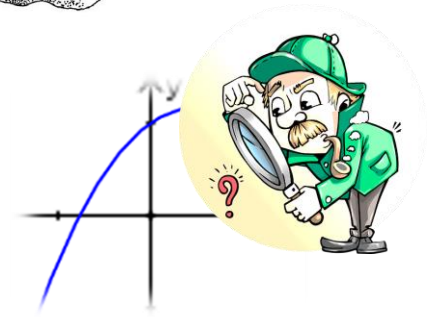
Example of a Python program to calculate the approximation of a root in a given interval using the Bisection method.

```
def bissecao(f,a,b,erro):
    fa = f(a)
    fb = f(b)
    if fa*fb > 0:
       print("A função tem imagens com o mesmo sinal nos extremos deste intervalo")
       exit(0)

    while b-a > 2*erro:
        c = (a+b)/2.
        fc = f(c)
        if fa*fc < 0:
            b, fb = c, fc
        elif fa*fc > 0:
            a, fa = c, fc
        else:
            return c
    return c

def f(x):
    return x**3-5*x**2+2*x-3

print(bissecao(f,0,5,0.000001))
```



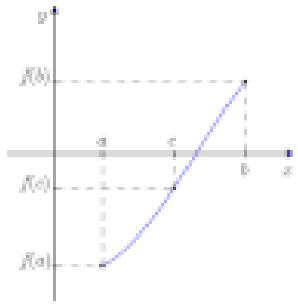
SOLVING EQUATIONS

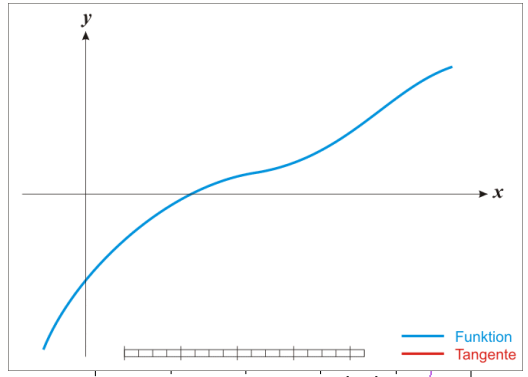


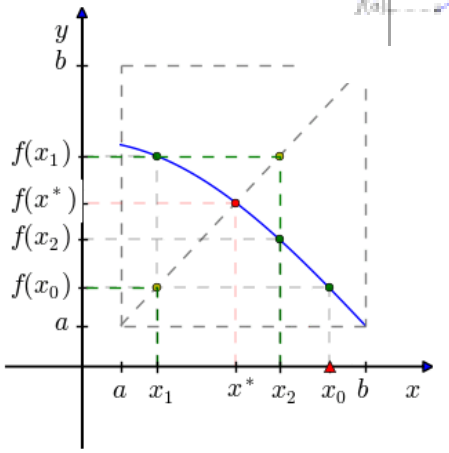


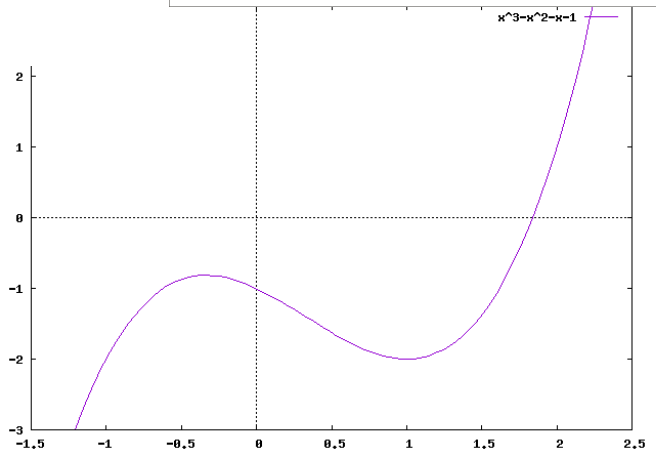
One of the oldest and most important problems in mathematics and other sciences

When direct methods fail, we rely exclusively on numerical methods to look for approximations of solutions.

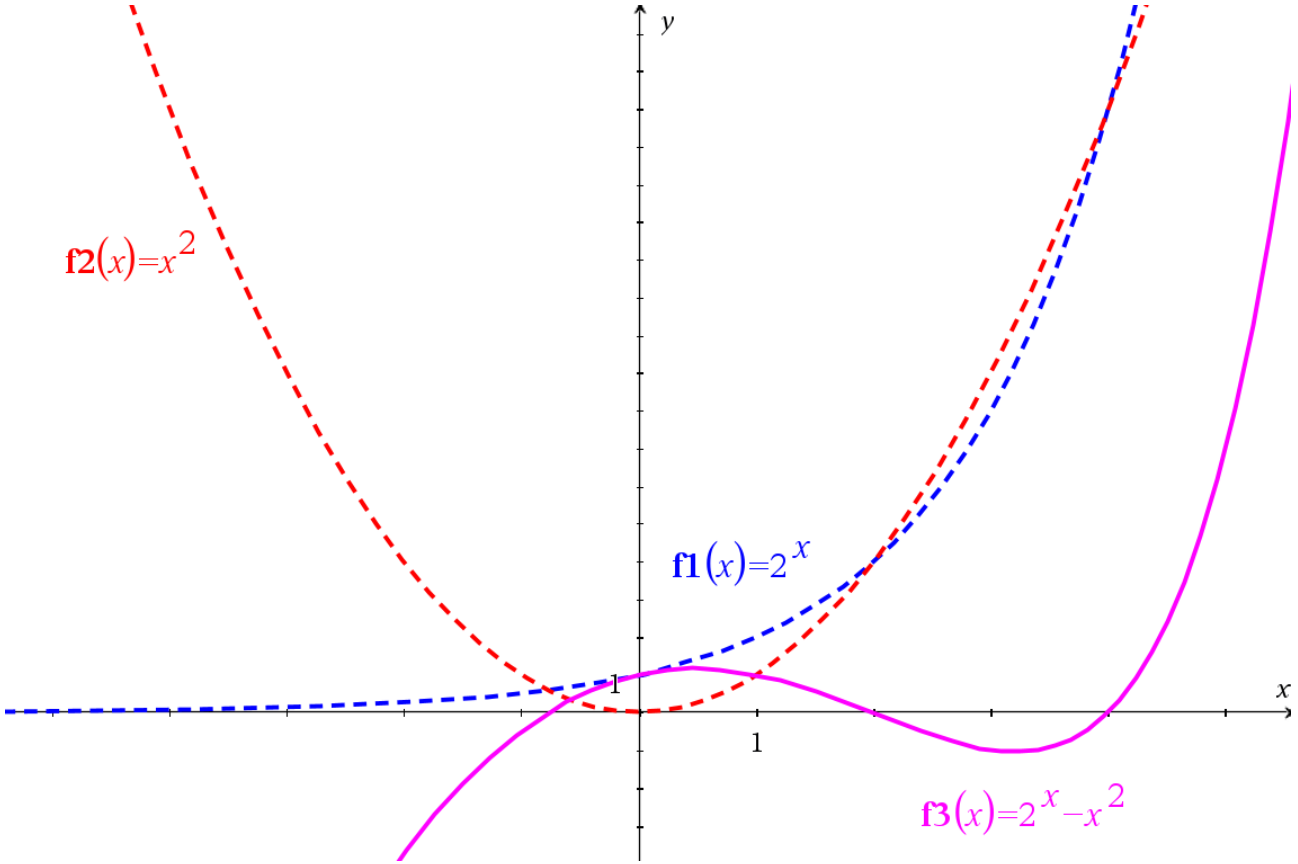








SOLVING EQUATIONS



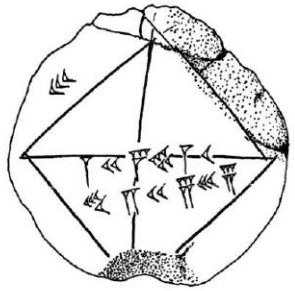
Hum..., teacher, you only have to tell me the formula used by the calculator to find the solution and it will be easy for me. I will follow that formula.

João Sousa, 2002

Fixed point method

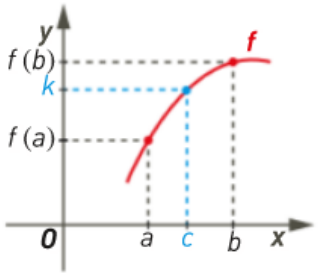
To find { Solution of $f(x) = 0$
Fixed point p of φ -> $\varphi(p) = p$

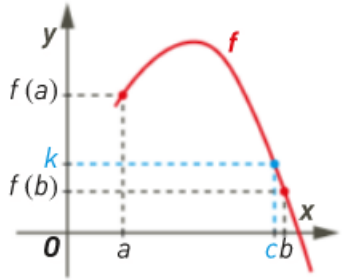
$$\underbrace{x^2 - 2 = 0}_{f(x)} \longrightarrow x^2 = 2 \longrightarrow x = \frac{2}{x} \longrightarrow 2x = x + \frac{2}{x} \longrightarrow \underbrace{x = \frac{1}{2}\left(x + \frac{2}{x}\right)}_{\varphi(x)}$$

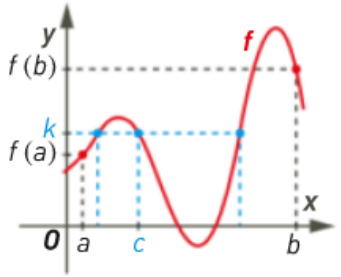


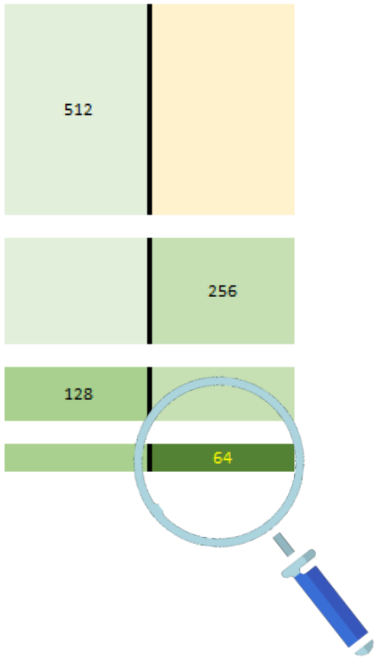
Bolzano-Cauchy theorem (Intemediate value theorem) and **Bisection method**

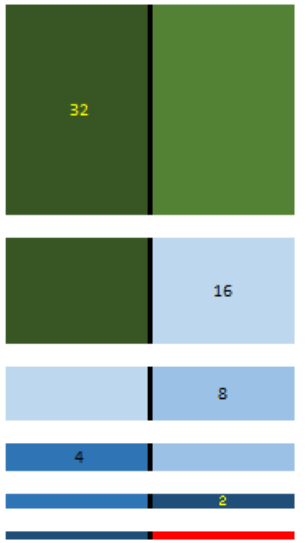
If f is a continuous function wose domain contains the interval [a,b], then it takes on any given value between f(a) and f(b).



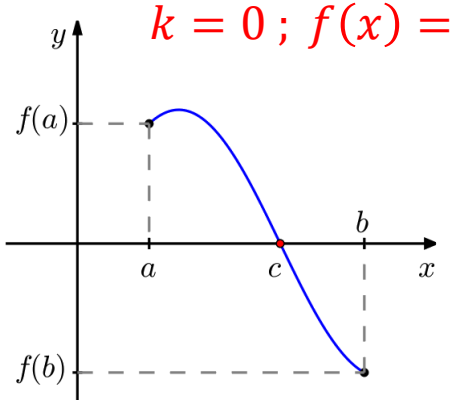


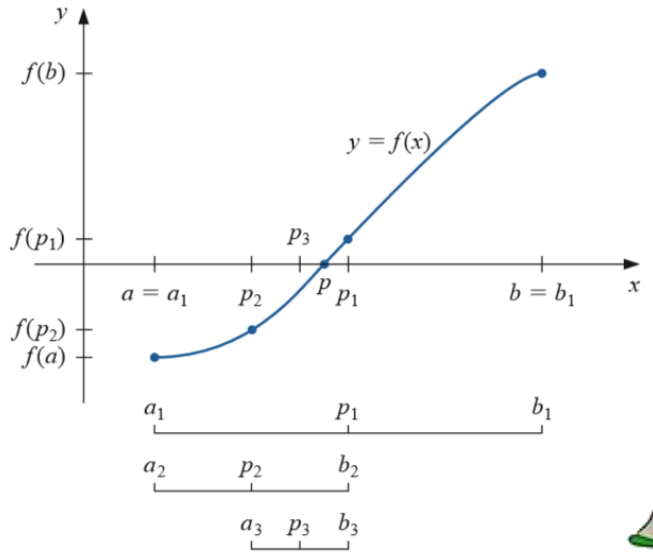


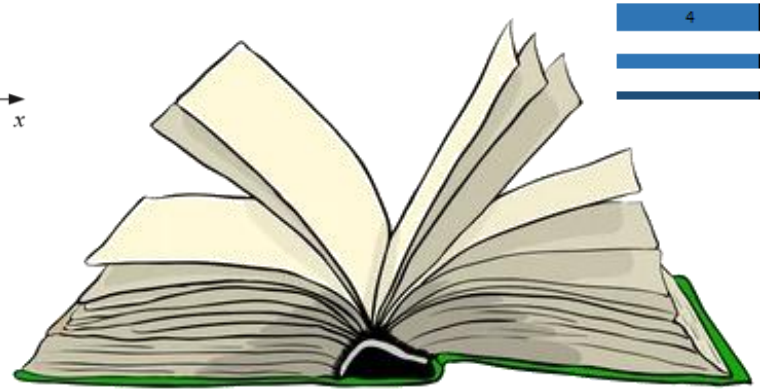




$k = 0\ ;\ f(x) = 0$







Computational Thinking

When working with algorithms, students should be encouraged to practice rigor and systematic verification and control practices. It will be important to promote abstraction in students, encouraging them to collect essential information for solving the proposed task (or situation). Students should be encouraged to identify the important elements in the algorithm creation process and to establish order among them. The recognition of patterns in the task (or situation) presented or in similar problems, previously solved, may contribute to facilitate the structuring of the algorithm to be developed. **Before writing the program in the Python language, it is convenient to describe the algorithm in natural language.**

The Bisection method is analogous to the binary search method on an ordered list and, among the interval section methods, it is the one that, in general, produces better estimates for an equal number of evaluations of the function.

from the public discussion draft

ALGORITHM

While $b - a > error$ **do** — $[a, b]$ *– inicial interval*; *any value in the interval is admited*

  $m \leftarrow \frac{a+b}{2}$ — *central value on* $[a, b]$

  if $f(m)$ *and* $f(a)$ have *oposite signs* — f *is the function to check*

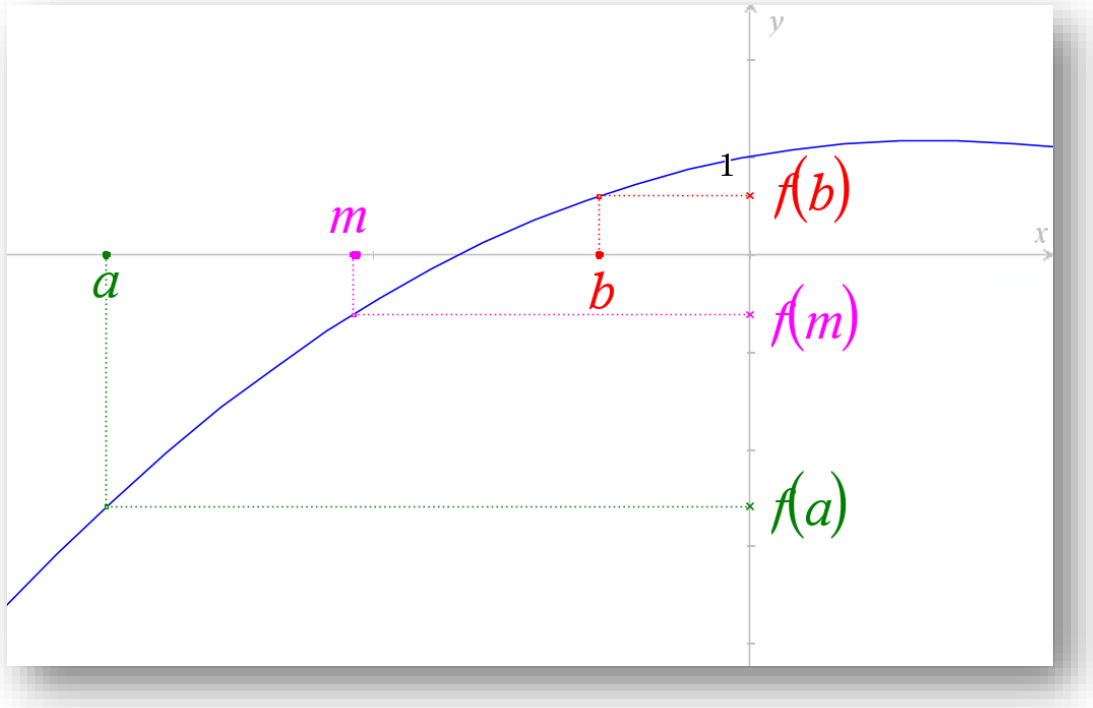
    then

      $b \leftarrow m$ — *solution is in the interval* $[a, m]$

    else

      $a \leftarrow m$ — *solution is in the interval* $[m, b]$

End



Bolzano-Cauchy theorem (Intemediate value theorem) and **Bisection method**

ALGORITHM

While $b - a > error$ **do**

$m \leftarrow \frac{a+b}{2}$

if $f(m)$ *and* $f(a)$ have *oposite signs*

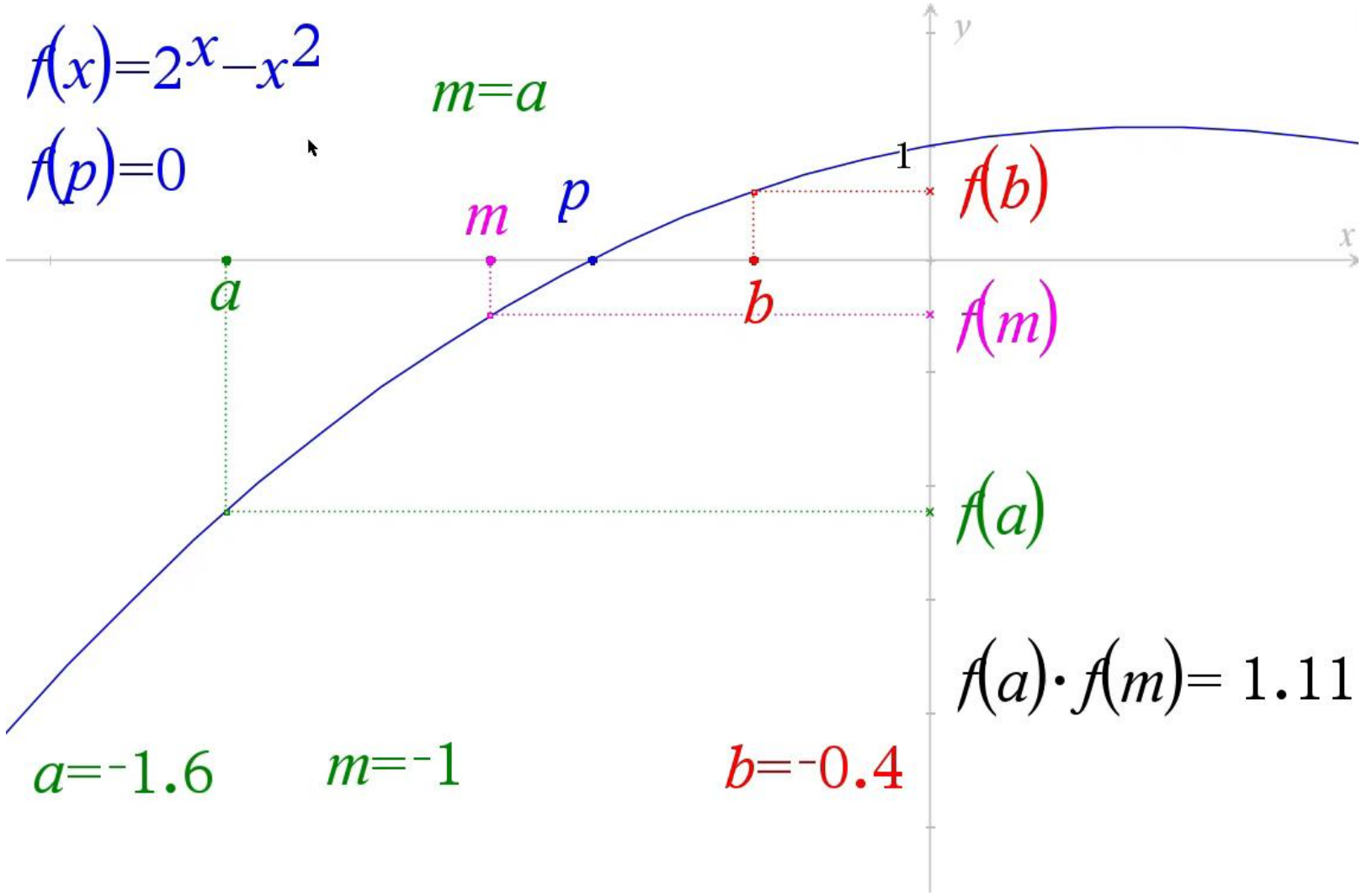
then

$b \leftarrow m$

else

$a \leftarrow m$

End



Bolzano-Cauchy theorem (Intemediate value theorem) and **Bisection method**

ALGORITHM

While $b - a > error$ **do**

$m \leftarrow \frac{a+b}{2}$

if $f(m)$ *and* $f(a)$ have *oposite signs*

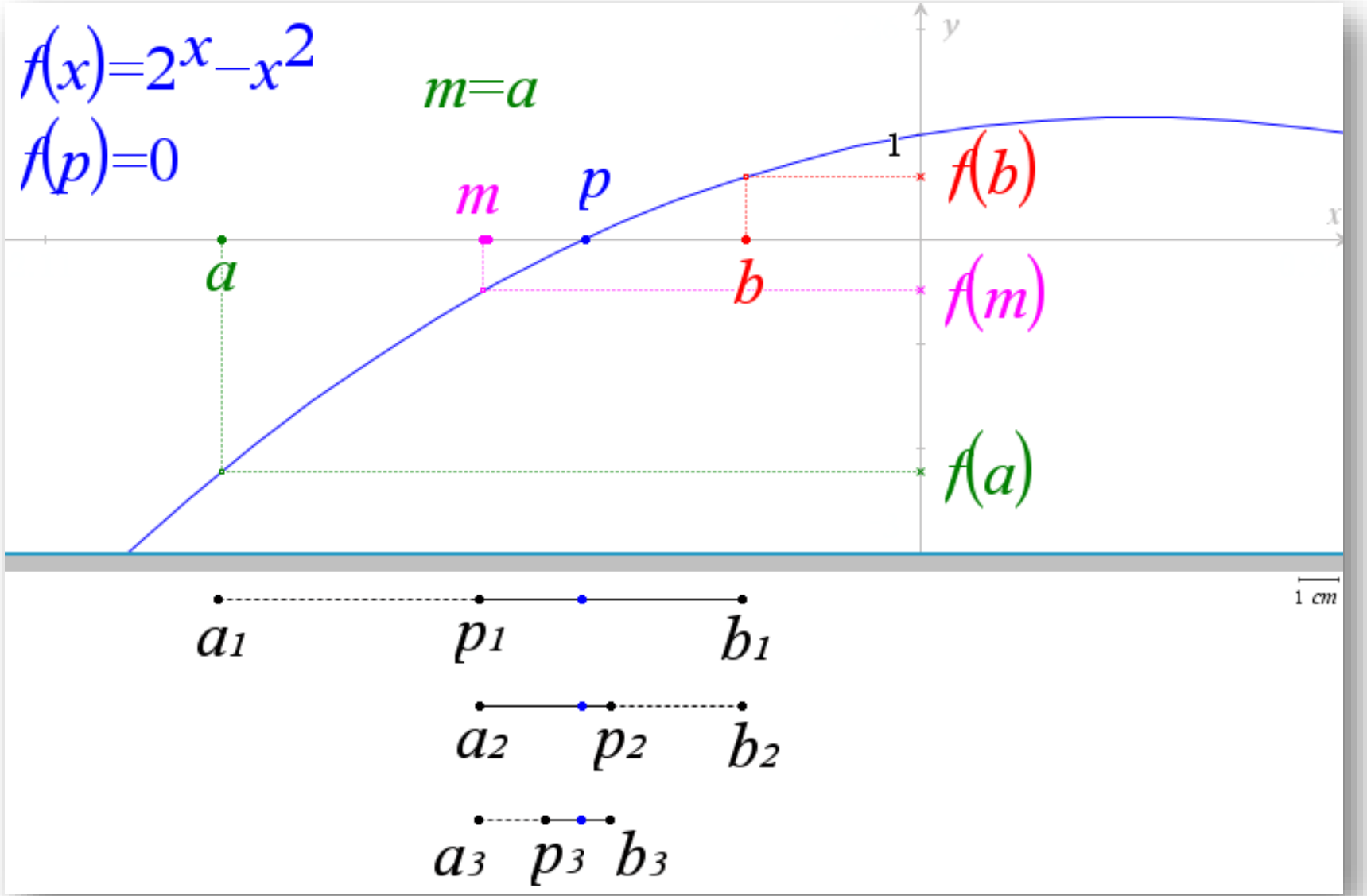
then

$b \leftarrow m$

else

$a \leftarrow m$

End



Bolzano-Cauchy theorem (Intemediate value theorem) and **Bisection method**

While $b - a > error$ **do**

- $m \leftarrow \frac{a+b}{2}$
- **if** $f(m)$ *and* $f(a)$ have *oposite signs*
  - **then**
    - $b \leftarrow m$
  - **else**
    - $a \leftarrow m$

End

bis1.py 3/12

```
def f(x):
    return 2**x-x**2

def bis(a,b,error):
    while b-a>error:
        m=(a+b)/2
        if f(a)*f(m)<0:
            b=m
        else:
            a=m
    return a,b
```

$2 \times error$

$return \ \frac{a+b}{2}$

With this change, the central value will aproach the solution less than the defined error

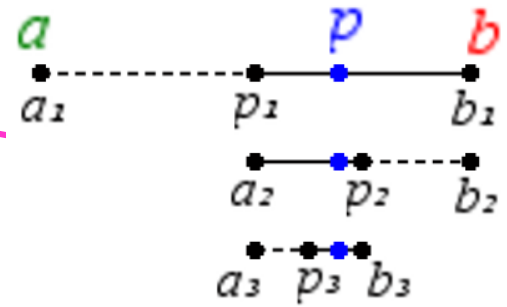
Shell Python 7/7

```
>>>#Running bis1.py
>>>from bis1 import *
>>>bis(-1,0,0.01)
(-0.7734375, -0.765625)
>>>bis(-1,0,0.001)
(-0.767578125, -0.7666015625)
>>>
```

All values in the interval aproximate the solution less than the defined error
Its central value aproximate the solution less than the half of the defined error

Bolzano-Cauchy theorem (Intemediate value theorem) and **Bisection method**


```
from ti_system import *
clear_history()

def f(x):
  return x**2-2

a=float(input("In [a,b], a = "))
b=float(input("In [a,b], b = "))

def bis(a,b,error):
  while b-a>2*error:
    m=(a+b)/2
    if f(m)==0:
      print("Exact solution")
      break
    if f(a)*f(m)<0:
      b=m
    else:
      a=m
  return (a+b)/2

if f(b)*f(a)>0:
  print("No sign change. Give other endpoints")
elif f(a)*f(b)==0:
  if f(a)==0:
    print("{} is the exact solution".format(a))
  else:
    print("{} is the exact solution".format(b))
else:
  error=float(input("Tolerance = "))
  print(bis(a,b,error))
```




```
from math import *
```

$$|p_1 - p| \leq \frac{b_1 - a_1}{2} = \frac{b - a}{2}$$

$$|p_2 - p| \leq \frac{b_2 - a_2}{2} = \frac{\frac{b_1 - a_1}{2}}{2} = \frac{b - a}{2^2}$$

$$|p_n - p| \leq \frac{b - a}{2^n}$$

$$n = Int\left(\log_2 \frac{b - a}{\varepsilon}\right) + 1$$

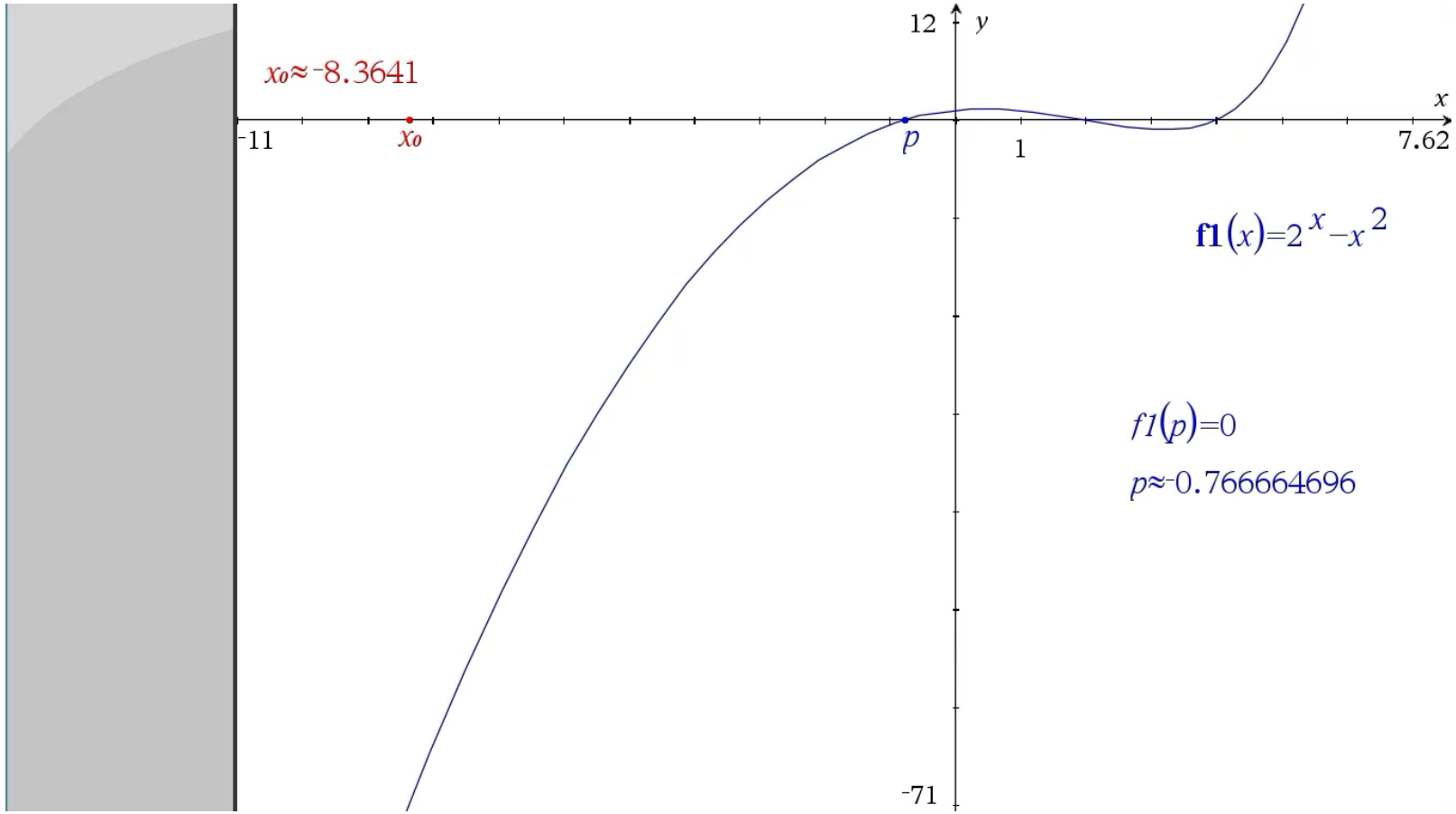
$$\varepsilon < b - a$$

```
def bis(a,b,error):
  nitera=int(log((b-a)/error,2))+1
  for k in range(nitera):
    m=(a+b)/2
    if f(m)==0:
      print("Exact solution")
      break
    if f(a)*f(m)<0:
      b=m
    else:
      a=m
  return m
```

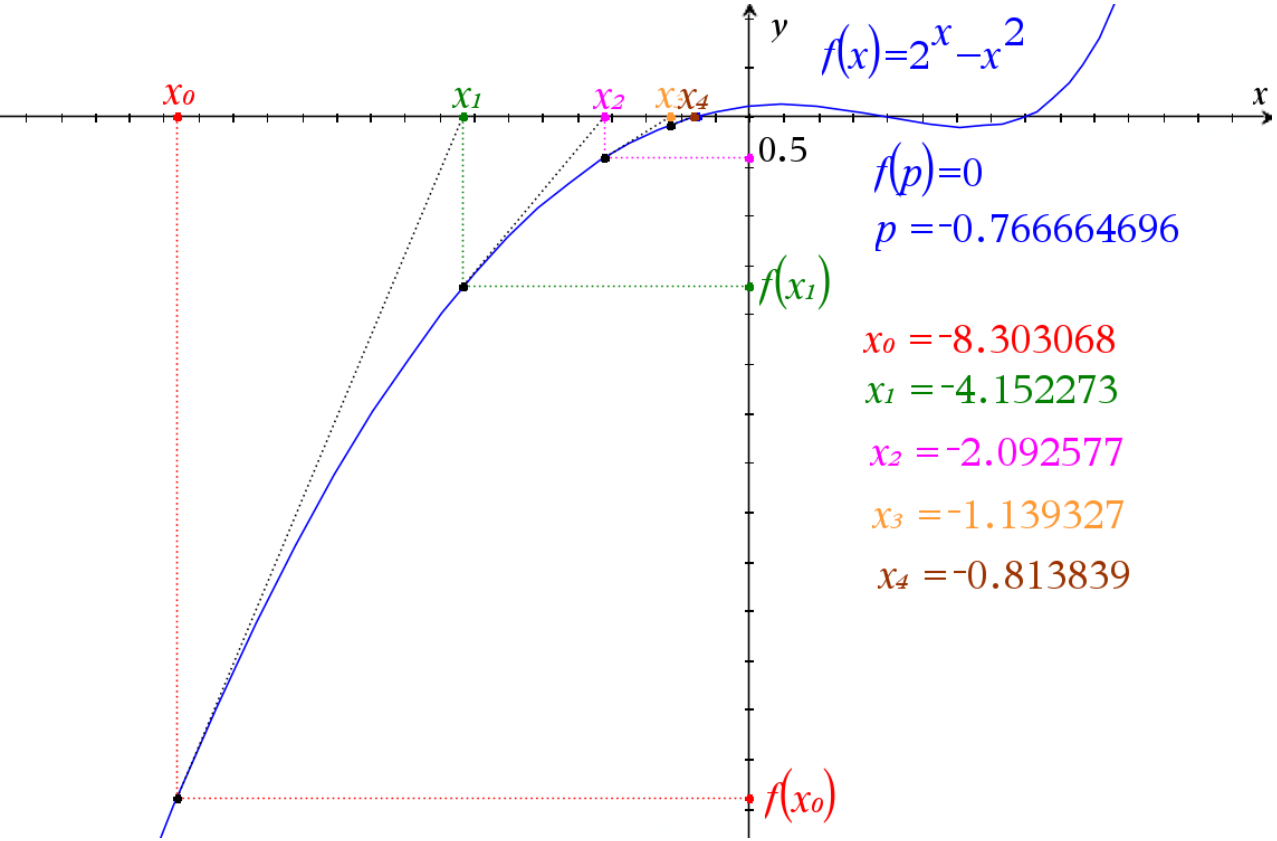
Bisection Method	To know and apply the Bissection method.	Promote the systematic use of technology in the application of the different methods listed for obtaining approximate solutions to equations
Newton-Raphson method Method	To know and apply the Newton-Raphson method.	Propose the development of programs in Python to implement Newton-Raphsons's and Bisection's methods. Illustrate from a concrete example the geometric interpretation of Newton-Raphson's method.

from the public discussion draft

Illustrate from a concrete example the geometric interpretation of Newton-Raphson´s method.



Newton-Raphson´s method



$$y - f(x_0) = f'(x_0) \times (x - x_0)$$

y ↑ 0 ; x ↑ x_1

$$0 - f(x_0) = f'(x_0) \times (x_1 - x_0)$$

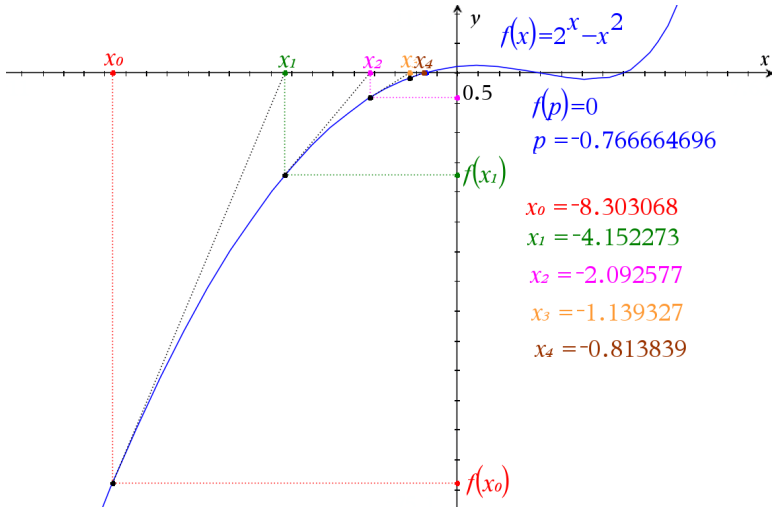
x_1 x_1 x_2 x_1

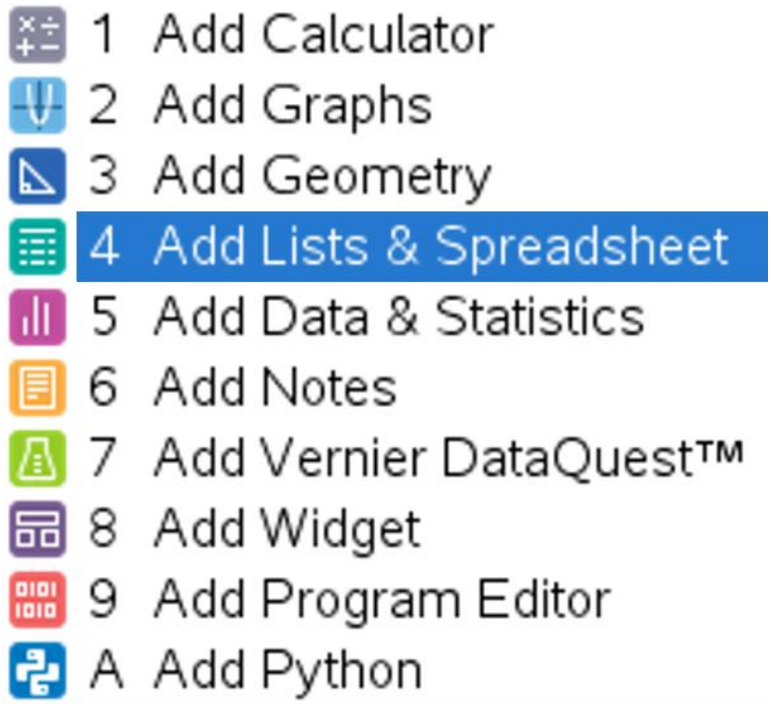
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

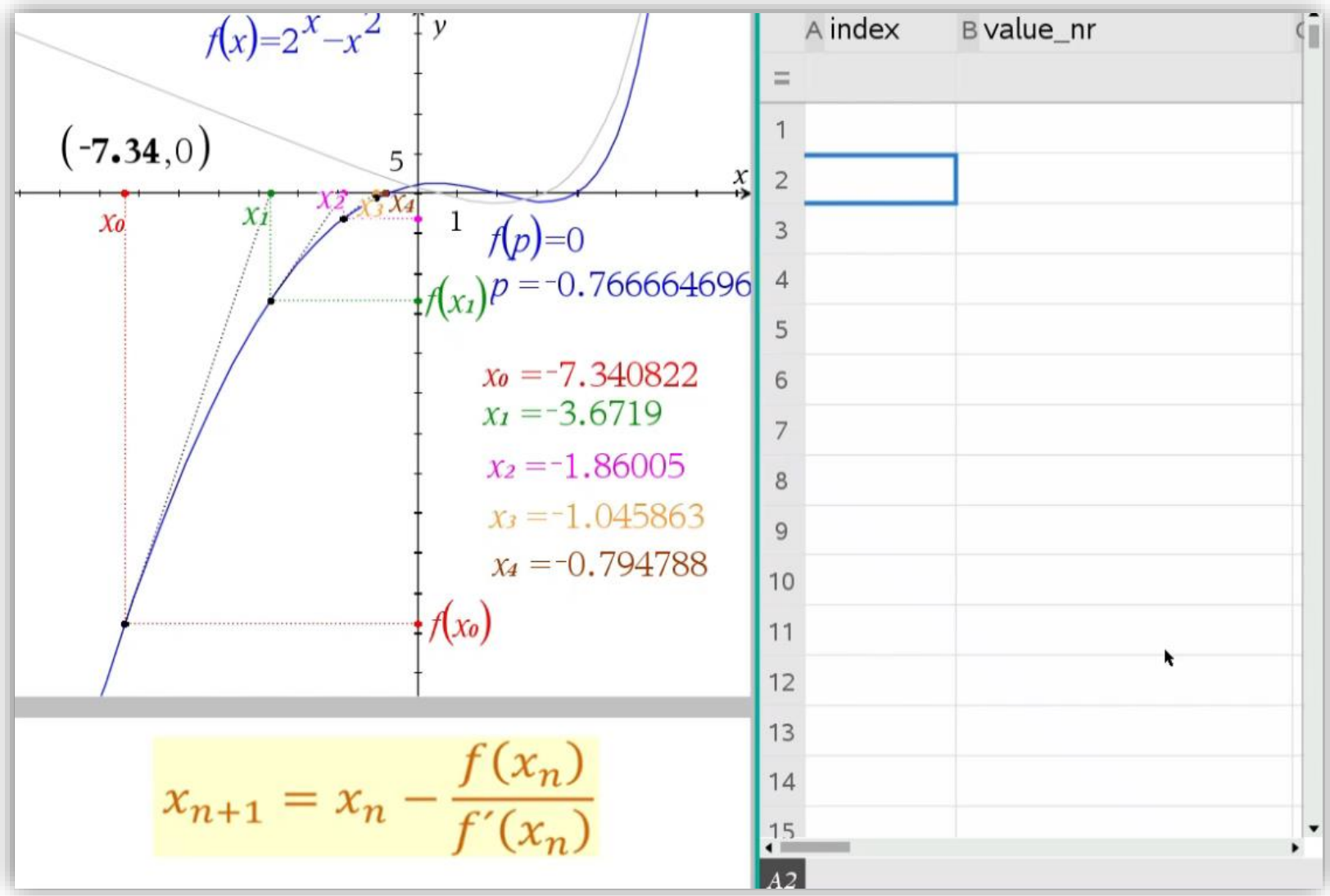
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

x_2 x_1

Newton-Raphson´s method







Bisection Method	To know and apply the bisection method.	Promote the systematic use of technology in the application of the different methods listed for obtaining approximate solutions to equations
Newton-Raphson method Method	To know and apply the Newton-Raphson method.	Propose the development of programs in Python to implement Newton-Raphsons's and Bisection's methods. Illustrate from a concrete example the geometric interpretation of Newton-Raphson's method.

from the public discussion draft

Propose the development of programs in Python to implement Newton-Raphson's … method...

Method Newton-Raphson method Method	To know and apply the Newton-Raphson method.	Promote the systematic use of technology in the application of the different methods listed for obtaining approximate solutions to equations Propose the development of programs in Python to implement Newton-Raphsons's and Bisection's methods. Illustrate from a concrete example the geometric interpretation of Newton-Raphson's method.

from the public discussion draft

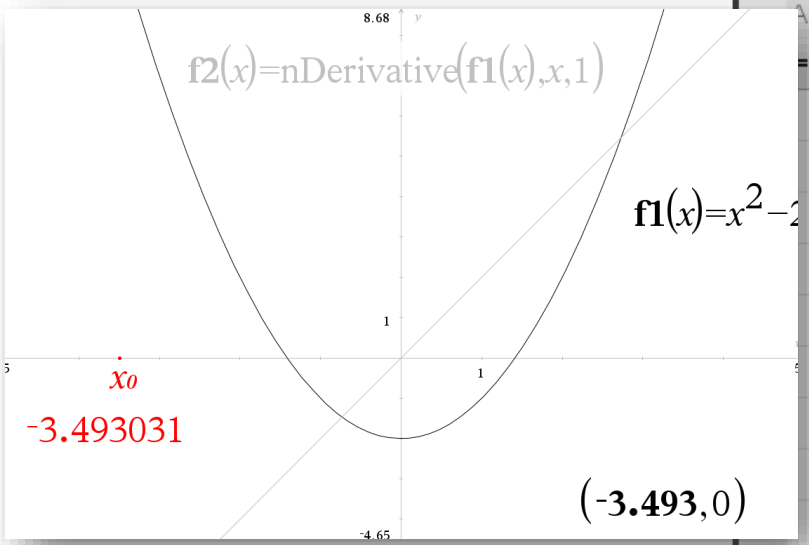
Propose the development of programs in Python to implement Newton-Raphson's … method...

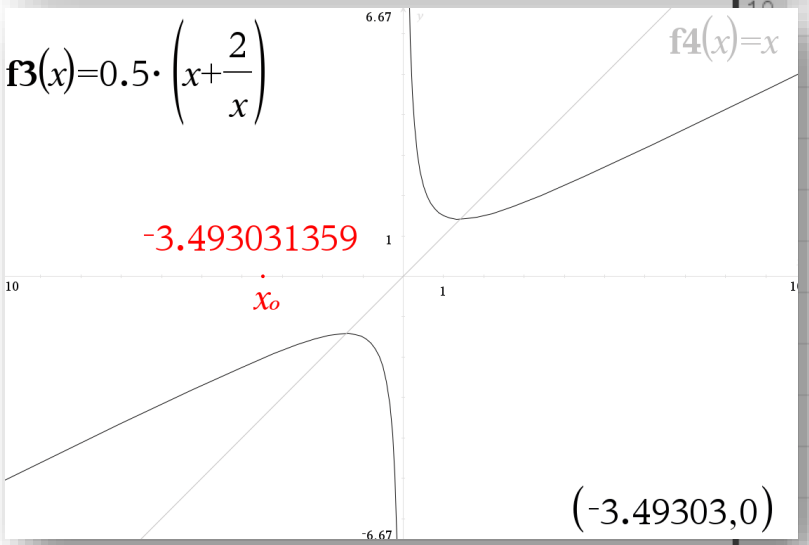

```
from math import *

def f(x):
    return 2**x-x**2

def df(x):
    return log(2,exp(1))*2**x-2*x

x=float(input("initial value = "))
n=int(input("n° of iterations = "))

for k in range(n):
    x=x-f(x)/df(x)
    print(x)
```

```
from math import *

def f(x):
    return 2**x-x**2

def df(x):
    return log(2,exp(1))*2**x-2*x

x=float(input("initial value = "))
e=float(input("error = "))

x1=x-f(x)/df(x)
print(x1)
while abs(x1-x)>=e:
    x=x1
    x1=x-f(x)/df(x)
    print(x1)
```

COMPARING SOME NUMERICAL METHODS TO SOLVE EQUATIONS

$$x^2 - 2 = 0$$



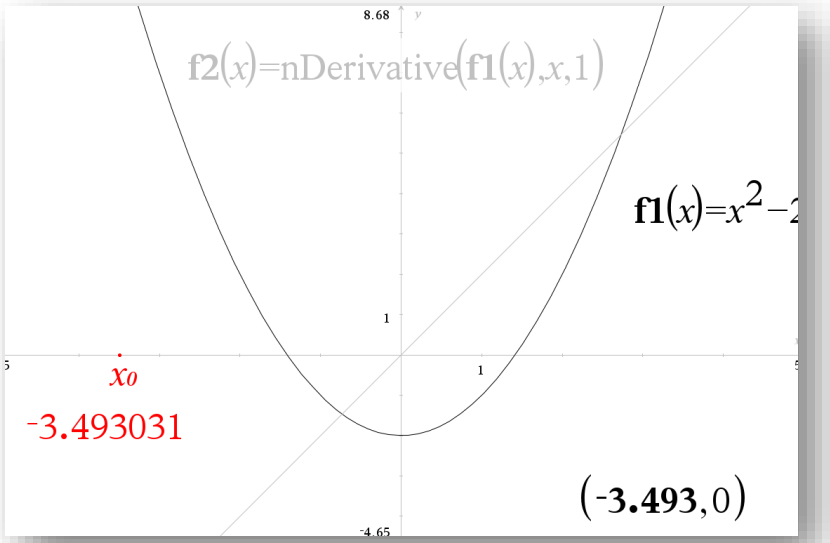


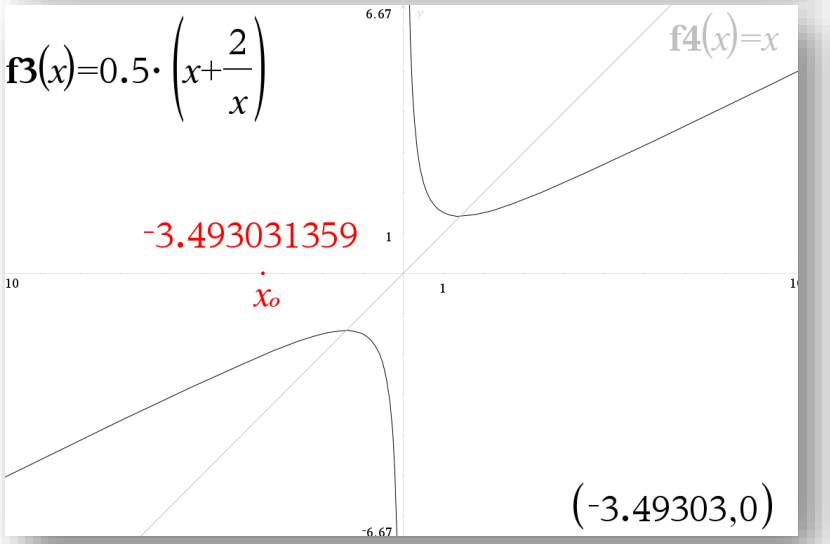
	A index	B newt_r	C bisect_mi	D fix_p	E a_i	F b_i	G f_a	H f_m	I
1	=seq(n,								
2	0	-3.49303135889	1.5	-3.49303135889	1	2	-1.	0.25	
3	1	-2.03279996872	1.25	-2.03279996872	1	1.5	-1.	-0.4375	
4	2	-1.50833230204	1.375	-1.50833230204	1.25	1.5	-0.4375	-0.1093...	
5	3	-1.41715002974	1.4375	-1.41715002974	1.375	1.5	-0.1093...	0.06640...	
6	4	-1.41421660469	1.40625	-1.41421660469	1.375	1.4375	-0.1093...	-0.0224...	
7	5	-1.41421356238	1.421875	-1.41421356238	1.40625	1.4375	-0.0224...	0.02172...	
8	6	-1.41421356237	1.4140625	-1.41421356237	1.40625	1.421875	-0.0224...	-4.2724...	
9	7	-1.41421356237	1.41796875	-1.41421356237	1.4140625	1.421875	-4.2724...	0.01063...	
10	8	-1.41421356237	1.416015625	-1.41421356237	1.4140625	1.41796875	-4.2724...	0.00510...	
11	9	-1.41421356237	1.4150390625	-1.41421356237	1.4140625	1.416015625	-4.2724...	0.00233...	
12	10	-1.41421356237	1.41455078125	-1.41421356237	1.4140625	1.4150390625	-4.2724...	9.53912...	
13	11	-1.41421356237	1.41430664063	-1.41421356237	1.4140625	1.41455078125	-4.2724...	2.63273...	
14	12	-1.41421356237	1.41418457031	-1.41421356237	1.4140625	1.41430664063	-4.2724...	-8.2001...	
15	13	-1.41421356237	1.41424560547	-1.41421356237	1.41418457...	1.41430664063	-8.2001...	9.06325...	
16	14	-1.41421356237	1.41421508789	-1.41421356237	1.41418457...	1.41424560547	-8.2001...	4.31481...	
17	15	-1.41421356237	1.4141998291	-1.41421356237	1.41418457...	1.41421508789	-8.2001...	-3.8843...	
18	16	-1.41421356237	1.4142074585	-1.41421356237	1.41419982...	1.41421508789	-3.8843...	-1.7264...	
19	17	-1.41421356237	1.41421127319	-1.41421356237	1.41420745...	1.41421508789	-1.7264...	-6.4747...	
20	18	-1.41421356237	1.41421318054	-1.41421356237	1.41421127...	1.41421508789	-6.4747...	-1.0799...	
21	19	-1.41421356237	1.41421413422	-1.41421356237	1.41421318...	1.41421508789	-1.0799...	1.61741...	

C bisect_mi

COMPARING SOME NUMERICAL METHODS TO SOLVE EQUATIONS

$$x^2 - 2 = 0$$





	newt_r		fix_p
0	-3.49303135889	≠	-3.49303135889
1	-2.03279996872		-2.03279996872
2	-1.50833230204	=	-1.50833230204
3	-1.41715002974		-1.41715002974
4	-1.41421660469		-1.41421660469
5	-1.41421356238		-1.41421356238
6	-1.41421356237	⇐	-1.41421356237
7	-1.41421356237		-1.41421356237
8	-1.41421356237		-1.41421356237
9	-1.41421356237		-1.41421356237
10	-1.41421356237		-1.41421356237
11	-1.41421356237		-1.41421356237
12	-1.41421356237		-1.41421356237
13	-1.41421356237		-1.41421356237
14	-1.41421356237	⇒	-1.41421356237
15	-1.41421356237		-1.41421356237
16	-1.41421356237		-1.41421356237
17	-1.41421356237		-1.41421356237
18	-1.41421356237		-1.41421356237
19	-1.41421356237		-1.41421356237

WHY?

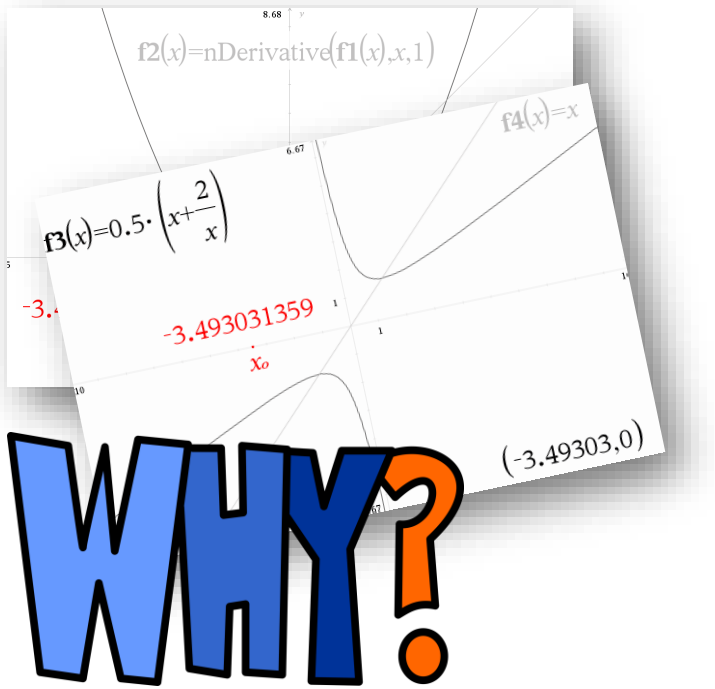
COMPARING SOME NUMERICAL METHODS TO SOLVE EQUATIONS

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

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

$$x = \varphi(x)$$

$$\varphi(x) = 0.5\left(x + \frac{2}{x}\right)$$



Newton-Raphson´s method iteration formula

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

$$x_1 = x_0 - \frac{{x_0}^2 - 2}{2x_0}$$

$$x_1 = \frac{2{x_0}^2 - {x_0}^2 + 2}{2x_0}$$

$$x_1 = \frac{{x_0}^2 + 2}{2x_0}$$

$$x_1 = \frac{1}{2}\left(\frac{{x_0}^2 + 2}{x_0}\right)$$

$$x_1 = \frac{1}{2}\left(x_0 + \frac{2}{x_0}\right)$$

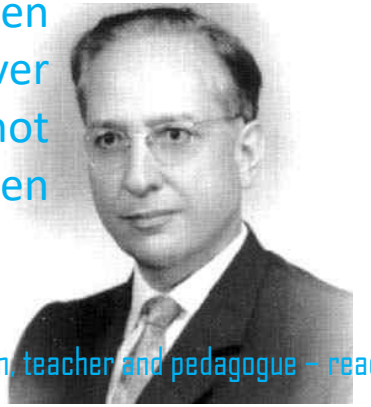
Fixedpoint method iteration formula

BUT IT IS NOT TRUE THAT BOTH METHODS ARE THE SAME IN GENERAL

Hum..., teacher, you only have to tell me the formula used by the calculator to find the solution and it will be easy for me. I will follow that formula.

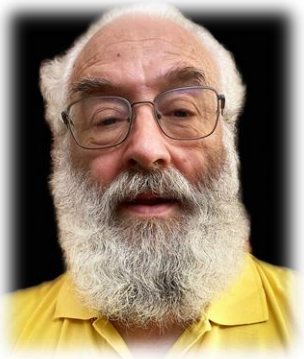
João Sousa - 2002 (my student)

"If someone asks them how to calculate all the roots of a given algebraic equation, of arbitrary degree, with whatever approximation one wants, they will have to admit that they do not know. This is a good example of how traditional teaching has been removed from reality.



José Sebastião e Silva - 60s of the 20th century (Mathematician, teacher and pedagogue – readins)

Rui Ralha – UMinho in 2014 (Mathematician and teacher - supervisor of my master's dissertation)



Jaime Carvalho e Silva – since 2000 (Mathematician and teacher – One of the biggest influencer of mathematics curricula in Portugal, unfortunately after "politicians" – influenced my master degree investigations).

APM Associação de Professores de Matemática

Agrupamento De Escolas De Ermesinde

python

Teachers Teaching with Technology™

Sharing Inspiration 2022 - A touch of STEM

T³ Europe webinar series 2022

Numerical Methods of Solving Equations to Develop Computational Thinking and Math Learning

Raul Aparício Gonçalves

t3raul@gmail.com