How to Cite:
Nadhim, A. I., & Al-Jilawi, A. S. (2022). The bridge between Newton's method and Newton-Raphson's method in numerical optimization. *International Journal of Health Sciences*, *6*(S1), 5249–5267. https://doi.org/10.53730/ijhs.v6nS1.6042

The bridge between Newton's method and Newton-Raphson's method in numerical optimization

Ammar Imad Nadhim
College of Education for Pure Science, University of Babylon, Hilla, Iraq
Email: ammar.imad.pure289@student.uobabylon.edu.iq

Dr. Ahmed Sabah Al-Jilawi
College of Education for Pure Science, University of Babylon, Hilla, Iraq
Email: ahmed.aljelawy@uobabylon.edu.iq

***Abstract*---**One of the most important tools for numerical analysis in this study was the Newton-Raphson's method mathematical model, which was used for a variety of purposes including practical research, data mining of the approach's core principles, convergence findings. Unconstrained minimization, constrained equality issues, convex programming and inner point techniques are a few of the optimization topics we study in-depth at our lab. Instead, the details of these approaches and how they might be used to numerical optimization with Python are explored in greater depth. The concept of linear approximation underpins the process of solving numerical functions and selecting the optimal answer. This procedure, when applied effectively, is commonly employed in homes that have been demolished.

***Keywords*---**Newton's method, Newton-Raphson's method, Numerical Optimization.

Introduction

One of the most powerful techniques in process integration is optimization. In the context of optimization, "best" is defined as the best option out of several possible choices. By minimizing or optimizing an objective function, the degree of goodness of a solution is determined (e.g., cost), [1,2,3]. Constraints and system model are used as a guidance in the search process To put it another way, optimization seeks to increase (or decrease) the objective function's value while taking into account a range of limitations (called constraints). In terms of equality and inequality, these limits are articulated.. The "objective function" is an evaluation measure that may be minimized or maximized under a variety of design

restrictions in numerical optimization. Computational and numerical techniques to issue solving speed up the time it takes to find a solution. Several strategies exist to reduce what may be a difficult issue in analytical mathematics to simple algebra, whether the goal is integration or the resolution of complex differential equations [4,5,6,19].

Iterative procedures that converge to a solution (for a certain class of problems) and/or heuristics that provide approximate answers to specific concerns can all be used by researchers to solve problems. Examples include Newton's and Raphson's algorithms. Equations can be effectively solved numerically using Newton-approach, Raphson's or Newton Method. It is based on the simple idea of linear approximation, as is the case with the majority of differential calculus

Methodology

Newton's method of optimization

Newton utilized an iterative method to discover the roots of a differentiable function F, which are the solutions to the equation F (x) = 0. Newton's technique may be used to identify the derivative's roots (solutions to f'(x) = 0), which are also known as "crucial points" of the product [7,8,9]. Minimization of the function is the most important consideration when optimizing a process. Take a look at what happens when you only have one actual variable. We will get into a more general and useful multivariate case later on in this section.

We are trying to find a solution to the optimization problem $\min_{x \in R} f(x)$ F: R→ R is a twice-differentiable function. Newton's technique attempts to address this problem by using the following formula:

$$X_{k+1} = X_k - \frac{f'(X_k)}{f''(X_k)}$$

Newton - Raphson's method of optimization

Because it was named after Isaac Newton and Joseph Raphson's, it is known as the Newton–Raphson's approach. It is a technique for determining the roots (or zeroes) of a real-valued function (or zeroes). Simplest: It begins with a single-variable function that may be used to discover the root of a real number, the derivative of the function, and an estimate for the root. The function should work if it fulfills all of the requirements and the initial estimate is realistic

$$.X_1 = X_0 - \frac{f(X_0)}{f'(X_0)}$$

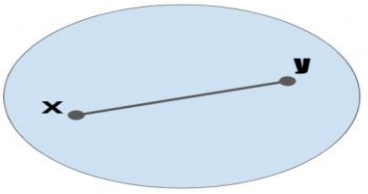
It represents the root more accurately than x_0 which does not. The x-axis intersects the tangent of the graph of f at $(x_0, f(x_0))$ at $(x_1, 0)$: At the original position, the new estimate is the only root of the linear approximation. This is how it works; the procedure is done as many times as necessary

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

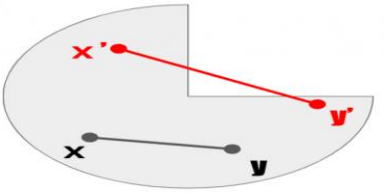
Convex set and convex function

It is also possible that S be a vector or affine space, or any other type of ordered field. Euclidean spaces are affine spaces, among other things. If the line connecting all of C's x and y points is inside C, it is convex. This indicates that for all values of x and y in C and all values of t in the range [0, 1], the affine combination (1-t) x + ty is a component of C. This demonstrates that convexity does not change as a result of affine transformations. This also implies that paths connect convex sets within a real or complex topological vector space, which is correct [10,11,12].

If every point on the line segment linking x and y, save the end points, is inside C's topological space, then C is strictly convex. To put it another way, a set C is "convex" if both its borders and center are extreme points, and it is also closed [16,17,18].



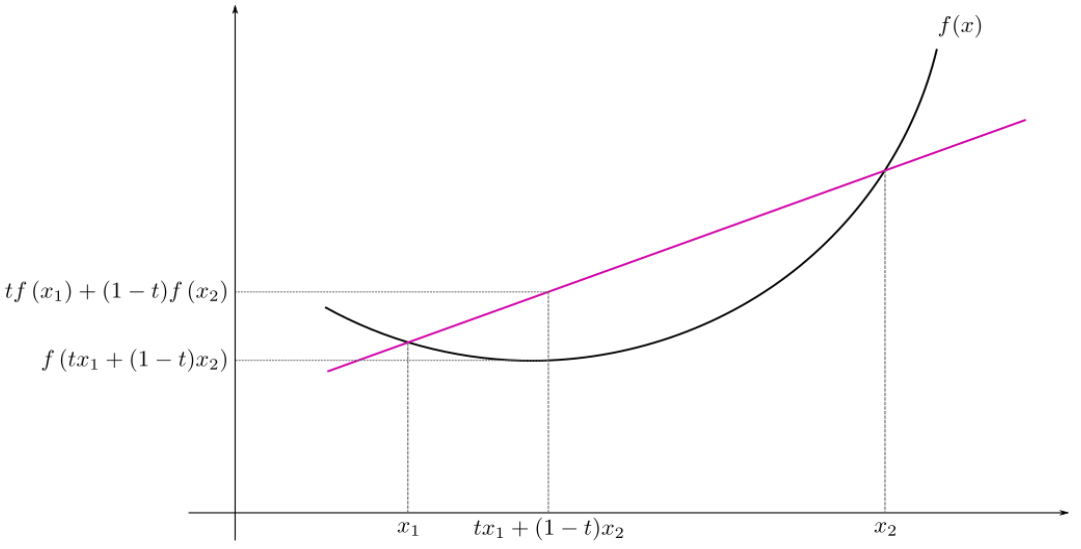
Convex Set



Non-Convex Set

If there is a convex set above the graph of a convex function, The convex function is a function that has a convex shape The function is convex when the line segment connecting any two points on a real-valued function's graph does not fall below the chart linking the two points. If the epigraph (a group of points on or around a text) is not clear, we will need to complete the following: You will need to consider what you know and do not know about the real vector space R. It is only convex if and only if one or more of the following conditions are satisfied. f: R→R. For all 0≤ t ≤ 1 and all $x_1, x_2 \in X$:

$$f(t\,x_1 + (1-t)x_2\ ) \leq t\,f(x_1) + (1-t\,)f(x_2)$$



Data Analysis

Compare the Method of Newton and the Method of Newton-Raphson's

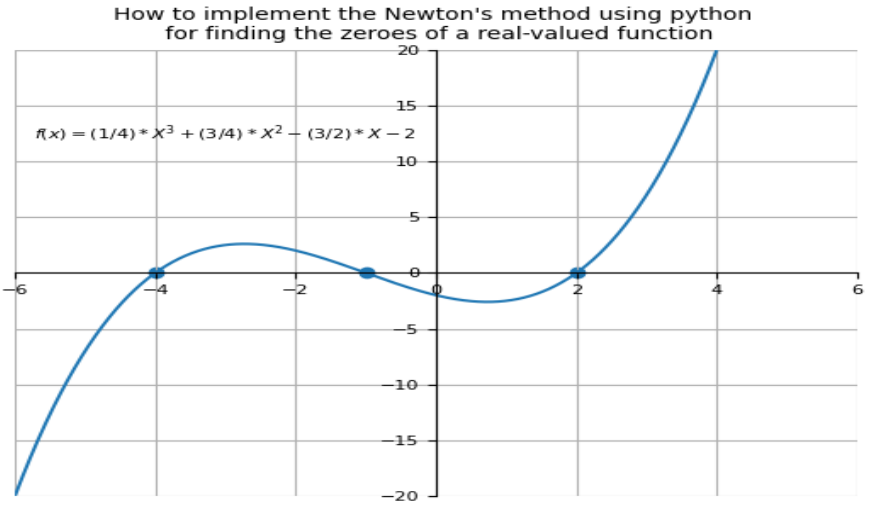
In this section, the implementation of compares shows us the result of newton Raphson's better than newton method.

Python Code in Newton's Method

```
from scipy import misc
def NewtonsMethod(f, x, tolerance=0.000001):
    while True:
        x1 = x - f(x) / misc.derivative(f, x)
        t = abs(x1 - x)
        if t < tolerance:
            break
        x = x1
    return x
def f(x):
    return (1.0/4.0)*x**3+(3.0/4.0)*x**2-(3.0/2.0)*x-2
x = 4
x0 = NewtonsMethod(f, x)
x1=NewtonsMethod(f, x)
x2= NewtonsMethod(f, x)
x3= NewtonsMethod(f, x)
x4= NewtonsMethod(f, x)
print('x: ', x)
print('x0: ', x0)
print("f(x0) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x1: ', x1)
print("f(x1) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x2: ', x2)
print("f(x2) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x3: ', x3)
print("f(x3) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
print('x4: ', x4)
print("f(x4) = ", ((1.0/4.0)*x0**3+(3.0/4.0)*x0**2-(3.0/2.0)*x0-2 ))
#!/usr/bin/env python
from pylab import *
t = arange(-6.0, 4.0, 0.01)
s= t*t*t/4.0+3.0*t*t/4.0-3*t/2.0-2.0
ax = subplot(111)
ax.plot(t, s)
ax.scatter([-4,-1,2],[0,0,0])
ax.grid(True)
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.spines['top'].set_color('none')
ax.set_xlim(-6,6)
ax.set_ylim(-20,20)
text(-3.0, 12,
    r"$f(x)=(1/4)*X^3+(3/4)*X^2-(3/2)*X-2$", horizontalalignment='center',
    fontsize=8)
plt.title("How  to implement the Newton's method using python \n for finding the zeroes of a real-valued function",fontsize=10)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.savefig('NewtonMethodExemple.png')
```

Newton method implementation

Iteration	X_n	$f\ (\ x_n)$
0	x_0 = 2.0000002745869883	$f(x_0)$ = 1.2356416165815176e-06
1	x_1 = 2.0000002745869883	$f(x_1)$ = 1.2356416165815176e-06
2	x_2 = 2.0000002745869883	$f(x_2)$ = 1.2356416165815176e-06
3	x_3 = 2.0000002745869883	$f(x_3)$ = 1.2356416165815176e-06
4	x_4 = 2.0000002745869883	$f(x_4)$ = 1.2356416165815176e-06
The required root is 2.0000002745869883		



This figure above compute the root value for the x-axis as (-4, -1, 2), and we choose the root to be (2)

Python Code in Newton-Raphson's Method

```
# Defining Function
def f(x):
    return 2 * x ** 3 - 4 * x ** 2 - 7 * x + 5
# Defining derivative of function
def g(x):
    return 6 * x ** 2 - 8 * x - 7
# Implementing Newton Raphson Method
def newtonRaphson(x0, e, N):
    print('\n\n*** NEWTON RAPHSON METHOD IMPLEMENTATION ***')
    step = 1
    flag = 1
    condition = True
    while condition:
        if g(x0) == 0.0:
            print('Divide by zero error!')
            break
        x1 = x0 - f(x0) / g(x0)
        print('Iteration-%d, x1 = %0.6f and f(x1) = %0.6f' % (step, x1, f(x1)))
        x0 = x1
        step = step + 1
        if step > N:
            flag = 0
            break
        condition = abs(f(x1)) > e
    if flag == 1:
        print('\nRequired root is: %0.8f' % x1)
    else:

    print('\nNot Convergent.')
# Input Section
x0 = input('Enter Guess: ')
e = input('Tolerable Error: ')
N = input('Maximum Step: ')
# Converting x0 and e to float
x0 = float(x0)
e = float(e)
# Converting N to integer
N = int(N)
# Note: You can combine above three section like this
# x0 = float(input('Enter Guess: '))
# e = float(input('Tolerable Error: '))
# N = int(input('Maximum Step: '))
# Starting Newton Raphson Method
newtonRaphson(x0, e, N)
```

Attempt to Guess Correctly: 20.00001 is the maximum allowable error
The maximum number of steps that may be taken is 11

Newton Raphson's method implementation

Iteration	x_n	f (x_n)
1	x1 = 11.000000	f(x1) = 2106.000000
2	x1 = 7.662441	f(x1) = 616.280611
3	x1 = 5.492274	f(x1) = 177.243452
4	x1 = 4.129410	f(x1) = 48.715654
5	x1 = 3.347167	f(x1) = 11.755884
6	x1 = 2.995656	f(x1) = 1.900353
7	x1 = 2.912593	f(x1) = 0.095266
8	x1 = 2.907968	f(x1) = 0.000288
9	x1 = 2.907954	f(x1) = 0.000000
The required root is: 2.90795416		

Results and Discussion

The maximum and lowest values of functions and the roots were determined using Newton's approach. Instead of the original function having approximate roots of the value of the function is equal to zero, thus use Newton's technique up to derivative function to determine its origins, beginning with the initial approximation x_0. Then, using the tangent lines of the f graph, construct a series of approximations x_1, x_2, x_3 by combining the first and second derivatives; we may get four iterations. As the number of iterations grows, the value becomes a constant value (2.0000002745869883).

Newton Raphson's approach consists of the solution of equations of the type f (x) = 0. We fabricate. It's the first guess at We are seeking for x_0, which is the root. x_1, x_2, x_3,... then by iterating on the first derivative to discover the roots, where we needed nine iterations until we got the value of the root (2.90795416), indicating that the more iterations in this procedure, the lower the root values [13,14,15].

Conclusion

In this paper, we have developed a numerical method based on comparing Newton's and Newton-Raphson's methods of the optimization problem. We now have achieved the optimal value depending on the investigation of numerical analysis and the result new approach showed that newton Raphson's technique was more robust than newton we used python code for implementation.

References

[1] Kutz, J. N. (2013). Data-driven modeling & scientific computation: methods for complex systems & big data. Oxford University Press.
[2] R. L. Burden and J. D. Faires, Numerical Analysis (Brooks/Cole, 1997).
[3] R. K. Ahuja, T. L. Magnanti, AND J. B. ORLIN, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, N.J., 1993.
[4] H. Akaike, On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method, Annals of the Institute of Statistical Mathematics, 11 (1959), pp. 1–17.

[5] M. AL-BAALI, Descent property and global convergence of the Fletcher-Reeves method with inexact line search, I.M.A. Journal on Numerical Analysis, 5 (1985), pp. 121–124.
[6] E. D. ANDERSEN AND K. D. Andersen, Presolving in linear programming, Mathematical Programming, 71 (1995), pp. 221–245.
[7] The Mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm, in High Performance Optimization, T. T. H. Frenk, K. Roos and S. Zhang, eds., Kluwer Academic Publishers, 2000, pp. 197–232.
[8] E. D. Andersen, J. Gondzio, C. Mesz 'Aros', And X. Xu, Implementation of interior-point methods for large scale linear programming, in Interior Point Methods in Mathematical Programming, T. Terlaky, ed., Kluwer, 1996, ch. 6, pp. 189–252.
[10] E. Anderson, Z. BAI, C. Bischof, J. Demmel, J. Dongarra, J. du Croz, A. Greenbaum, S. Hammarling, A. Mckenney, S. Ostrouchov, AND D. Sorensen, Lapack User's Guide, SIAM, Philadelphia, 1992.
[11] R EFERENCES M. ANITESCU, On solving mathematical programs with complementarity constraints as nonlinear programs, SIAM Journal on Optimization, 15 (2005), pp. 1203–1236.
[12] Arki Consulting And Development A/S, CONOPT version 3, 2004.
[13] B. M. AVERICK, R. G. CARTER, J. J. MORE', AND G. XUE, The MINPACK-2 test problem collection,
[14] Al-Jilawi, A. S., & Abd Alsharify, F. H. (2022). Review of Mathematical Modelling Techniques with Applications in Biosciences. Iraqi Journal For Computer Science and Mathematics, 3(1), 135-144.
[15] Alridha, A., Wahbi, F. A., & Kadhim, M. K. (2021). Training analysis of optimization models in machine learning. International Journal of Nonlinear Analysis and Applications, 12(2), 1453-1461.
[16] Kadhim, M. K., Wahbi, F. A., & Hasan Alridha, A. (2022). Mathematical optimization modeling for estimating the incidence of clinical diseases. International Journal of Nonlinear Analysis and Applications, 13(1), 185-195.
[17] Alridha, A., Salman, A. M., & Al-Jilawi, A. S. (2021, March). The Applications of NP-hardness optimizations problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012179). IOP Publishing.
[18] Salman, A. M., Alridha, A., & Hussain, A. H. (2021, March). Some Topics on Convex Optimization. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012171). IOP Publishing.
[19] Alridha, A., & Al-Jilawi, A. S. (2021, March). Mathematical Programming Computational for Solving NP-Hardness Problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012137). IOP Publishing.