Roots of Equations

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 201, Fall Semester 2023

September 30, 2023

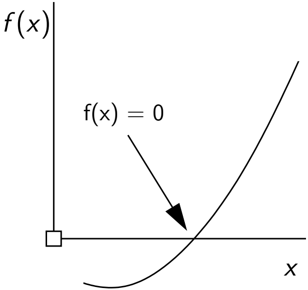
Overview

Part 3

Numerical Solution of Equations

Numerical Solution of Equations

Math Problem. Given f(x), find a value of x such that f(x) = g(x), f(x) = constant, or f(x) = 0.

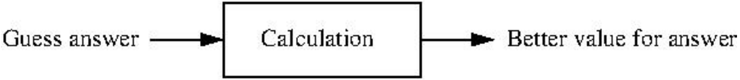


All forms may be put in the format F(x) = 0.

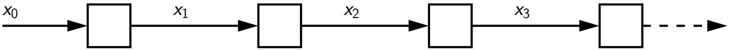
Iterative Methods

Iterative Methods

Procedure. Solve problem through a sequence of approximations:



Apply process iteratively:



Ideally, $x_0, x_1, \cdots, x_n$ will converge to the true answer.

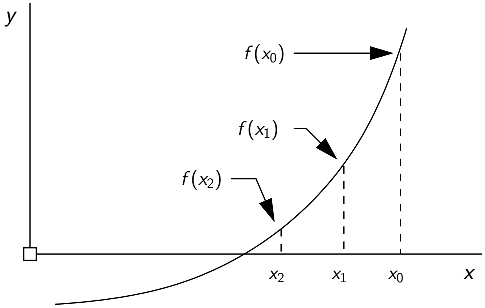
Potential problems:

- Sequence may not converge.
- Convergence may be slow.

Problem Solving Strategies

Problem Solving Strategies

Open Methods: Methods may involve one or more initial guesses, but no need to bracket a solution.



- Algorithms are designed to provide updates: Newton Raphson Iteration, Modified Newton Raphson.

Newton Raphson Iteration

Newton-Raphson Iteration

Derivation of Numerical Procedure. Starting point $(x_0, f(x_0))$.

We wish to find a steplength $h = x_1$ - x_0 that will provide an improved estimate of the root.

Using first-order Taylor's expansion:

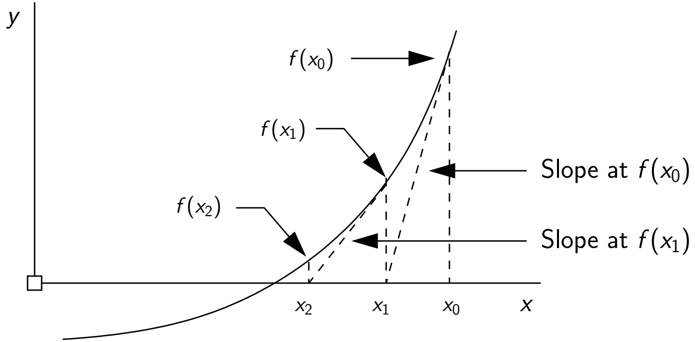
$$f(x_1) = f(x_0) + hf'(x_0) + O(h^2) = 0.0 \tag{6}$$

Next, neglect $O(h^2)$ terms, rarrange, and generalize:

$$x_{n+1} = x_n - \left[\frac{f(x_n)}{f'(x_n)}\right]. \tag{7}$$

Newton-Raphson Iteration

Schematic: Two iterations of Newton-Raphson.



Sequence of estimates is: x_0, x_1, x_3, ...

Newton-Raphson Iteration

Example 1. Solve $f(x) = 0$ where

$$f(x) = xsin(x) - 3cos(x) \tag{8}$$

Differentiating,

$$f^{'}(x) = sin(x) + xcos(x) + 3sin(x) \tag{9}$$

The Newton-Raphson update is:

$$x_{n+1} = x_n - \left[\frac{x_n sin(x_n) - 3cos(x_n)}{sin(x_n) + x_n cos(x_n) + 3sin(x_n)}\right]. \tag{10}$$

This gives: $x_0 = 0.8$, x_1 - 1.24, $x_2 = 1.1927$ ....

Newton-Raphson Iteration

Example 2. Demonstrate use of newton-raphson algorithm by computing roots of the quadratic equation

$$f(x) = (x - 3) * (x - 3) - 2; \quad (11)$$

The derivative is given by:

$$df(x)/dx = 2x - 6. \quad (12)$$

The source code is partitioned into two Python:

1. Solutions.py: Contains function for newton raphson algorithm.
2. TestNewtonRaphson.py. main test program + f1(x) and df1(x).

Program Source Code

```
# =====================================================================
# TestNewtonRaphson01.py: Use newton raphson algorithm to compute roots of
# equations.
#
# Written By: Mark Austin                                   February 2023
# =====================================================================

import math;
import Solutions;

# Mathematical functions: (x-3)*(x-3) - 2 = 0 ...

def f1(x):
    return (x-3)*(x-3)-2;

def df1(x):
    return 2*(x-3);

# main method ...

def main():
    print("--- Enter TestNewtonRaphson01.main()          ... ");
    print("--- ======================================== ... ");

    print("--- ");
    print("--- Case Study 1: Solve (x-3)*(x-3)-2 = 0, Initial guess: x0 = -10 ... ");
    print("--- ============================================================ ... ");
```

Program Source Code

```
    # Initialize problem setup ...

    x0 = -10.0;
    tolerance     = 0.001
    maxiterations = 100

    print("--- Inputs:")
    print("---   x0 = {:5.2f} ...".format(x0) )
    print("---   tolerance      = {:8.5f} ...".format(tolerance) )
    print("---   max iterations = {:8.2f} ...".format(maxiterations) )

    # Compute roots to equation ...

    print("--- Execution:")
    root, i, converged = Solutions.newtonraphson(f1, df1, x0, tolerance, maxiterations )

    # Summary of computations ...

    print("--- Output:")
    print("---   root = {:10.5f} ...".format(root) )
    print("---   f(root) --> {:12.5e} ...".format( f1(root)) )
    print("---   no iterations = {:d} ...".format(i) )
    print("---   converged: {:s} ...".format( str(converged) ) )

    print("--- ");
    print("--- Case Study 2: Solve (x-3)*(x-3)-2 = 0, Initial guess: x0 = 10 ... ");
    print("--- ============================================================ ... ");

    # Initialize problem setup ...
```

Program Source Code

```
        x0 = 10.0;
        tolerance     = 0.001
        maxiterations = 100

        print("--- Inputs:")
        print("---   x0 = {:5.2f} ...".format(x0) )
        print("---   tolerance      = {:8.5f} ...".format(tolerance) )
        print("---   max iterations = {:8.2f} ...".format(maxiterations) )

        # Compute roots to equation ...

        print("--- Execution:")
        root, i, converged = Solutions.newtonraphson(f1, df1, x0, tolerance, maxiterations )

        # Summary of computations ...

        print("--- Output:")
        print("---   root = {:10.5f} ...".format(root) )
        print("---   f(root) --> {:12.5e} ...".format( f1(root)) )
        print("---   no iterations = {:d} ...".format(i) )
        print("---   converged: {:s} ...".format( str(converged) ) )

        print("--- ========================================= ... ");
        print("--- Leave TestNewtonRaphson01.main()           ... ");

    # call the main method ...
```

Newton-Raphson Iteration

Abbreviated Output: Case Study 1, x0 = -10.

```
--- Inputs:
---   x0 = -10.00 ...
---   tolerance      =  0.00100 ...
---   max iterations =   100.00 ...
--- Execution:
---   Initial Conditions:
---     x0     --> -1.00000e+01 ...
---     f(x0)  -->  1.67000e+02 ...
---     df(x0) --> -2.60000e+01 ...
---   Main Loop for Newton Raphson Iteration:
---   Iteration 01: dx = 6.42308e+00, x = -3.57692e+00, f(x) -> 4.12559e+01
---   Iteration 02: dx = 3.13641e+00, x = -4.40508e-01, f(x) -> 9.83710e+00
...
---   Iteration 06: dx = 2.60526e-03, x =  1.58578e+00, f(x) -> 6.78739e-06
---   Iteration 07: dx = 2.39970e-06, x =  1.58579e+00, f(x) -> 5.75895e-12
--- Output:
---   root =    1.58579 ...
---   f(root) -->  5.75895e-12 ...
---   no iterations = 7 ...
---   converged: True ...
```

Newton-Raphson Iteration

Abbreviated Output: Case Study 2, x0 = 10.

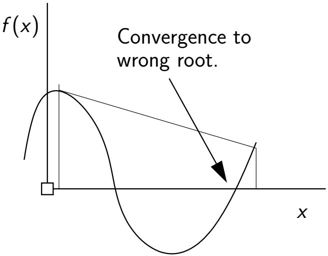
```
--- Inputs:
---   x0 = 10.00 ...
---   tolerance      =  0.00100 ...
---   max iterations =   100.00 ...
--- Execution:
---   Initial Conditions:
---     x0     -->  1.00000e+01 ...
---     f(x0)  -->  4.70000e+01 ...
---     df(x0) -->  1.40000e+01 ...
---   Main Loop for Newton Raphson Iteration:
---   Iteration 01: dx = -3.35714e+00, x = 6.64286e+00, f(x) -> 1.12704e+01
---   Iteration 02: dx = -1.54692e+00, x = 5.09594e+00, f(x) -> 2.39296e+00
...
---   Iteration 05: dx = -4.02419e-03, x = 4.41422e+00, f(x) -> 1.61941e-05
---   Iteration 06: dx = -5.72546e-06, x = 4.41421e+00, f(x) -> 3.27804e-11
--- Output:
---   root =    4.41421 ...
---   f(root) -->  3.27804e-11 ...
---   no iterations = 6 ...
---   converged: True ...
```

Modified

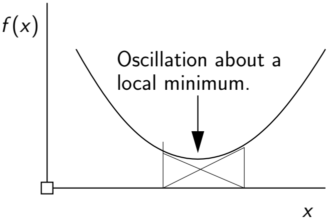
Newton Raphson Iteration

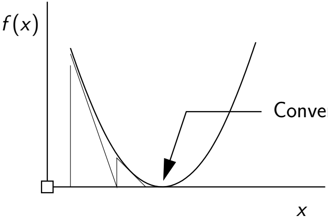
Limitations of Newton-Raphson Iteration

No Solutions



Many Solutions





Convergence to multiple roots: $\left[\frac{f(x)}{f'(x)}\right] = 0/0.$

Modified Newton-Raphson Iteration

Derivation of Numerical Procedure. In the case of multiple roots, we can improve on N-R by solving an equivalent problem:

$$F(x) = \left[\frac{f(x_n)}{f'(x_n)}\right] = 0. \tag{13}$$

Same solutions as f(x) = 0, but they occur as single roots.

Differentiating,

$$\frac{d}{dx}[F(x)] = \left[\frac{f(x_n)}{f'(x_n)}\right] = \left[\frac{(f'(x_n))^2 - f(x)f''(x)}{[f'(x_n)]^2}\right]. \tag{14}$$

Modified Newton-Raphson Iteration

Substituting into N-R formula:

$$x_{n+1} = x_n - \left[\frac{f(x_n) f'(x_n)}{(f'(x_n))^2 - f(x) f''(x)} \right]. \qquad (15)$$

Example 1. The function

$$f(x) = x^2 - 4x + 4, \ f'(x) = 2x - 4, \ f''(x) = 2. \qquad (16)$$

has a double root at $x = 2$. Using Newton-Raphson:

$$x_0 = 3.0$$

$$x_1 = 3.0 - \left[\frac{f(3.0)}{f'(3.0)} \right] = 3 - 1/2 = 2.5.$$

Modified Newton-Raphson Iteration

$$x_2 = 2.50 - \left[\frac{f(2.5)}{f'(2.5)}\right] = 2.25.$$

$$x_3 = 2.25 - \left[\frac{f(2.25)}{f'(2.25)}\right] = 2.125.$$

Using Modified Newton-Raphson:

$$x_0 = 3.0$$

$$x_1 = 3.0 - \left[\frac{f(3.0)f'(3.0)}{(f'(3.0))^2 - f(3.0)f''(3.0)}\right]$$

$$= 3.0 - \left[\frac{2}{2}\right] = 2.0. \quad \text{Exact answer in one step!}$$

Modified Newton-Raphson Iteration

Test Program Source Code:

```
# ==========================================================================
# TestModifiedNewtonRaphson01.py: Use modified newton raphson algorithm to
# compute solutions to equations having double roots.
#
# Written By: Mark Austin                                     February 2023
# ==========================================================================

import math;
import Solutions;

# Mathematical functions: (x-2)*(x-2) = 0 ...

def f1(x):
    return (x-2)*(x-2);

def df1(x):
    return 2*(x-2);

def ddf1(x):
    return 2;

# main method ...

def main():
    print("--- Enter TestModifiedNewtonRaphson01.main()  ... ");
    print("--- ========================================= ... ");
```

Modified Newton-Raphson Iteration

Test Program Source Code: Continued ...

```

    print("--- ");
    print("--- Case Study 1: Solve (x-2)*(x-2) = 0, Initial guess: x0 = 3      ... ");
    print("--- ============================================================== ... ");

    # Initialize problem setup ...

    x0 = 3.0;
    tolerance     = 0.001
    maxiterations = 100

    print("--- Inputs:")
    print("---   x0 = {:5.2f} ...".format(x0) )
    print("---   tolerance      = {:8.5f} ...".format(tolerance) )
    print("---   max iterations = {:8.2f} ...".format(maxiterations) )

    # Compute roots to equation ...

    print("--- Execution:")
    root, i, converged = Solutions.modifiednewtonraphson(f1, df1, ddf1, x0, tolerance, m

    # Summary of computations ...

    print("--- Output:")
    print("---   root = {:10.5f} ...".format(root) )
    print("---   f(root)    --> {:12.5e} ...".format(   f1(root)) )
```

Modified Newton-Raphson Iteration

Test Program Source Code: Continued ...

```
    print("---   df(root)  --> {:16.8e} ...".format(  df1(root)) )
    print("---   ddf(root) --> {:16.8e} ...".format( ddf1(root)) )
    print("---   no iterations = {:d} ...".format(i) )
    print("---   converged: {:s} ...".format( str(converged) ) )

    print("--- ");
    print("--- Case Study 2: Solve (x-2)*(x-2) = 0, Initial guess: x0 = -3   ... ");
    print("--- ============================================================== ... ");

    # Initialize problem setup ...

    x0 = -3.0;
    tolerance     = 0.001
    maxiterations = 100

    ... lines of source code removed ...
    ... details are identical to case study 1 ...

    print("--- ======================================== ... ");
    print("--- Leave TestModifiedNewtonRaphson01.main()  ... ");

# call the main method ...

main()
```

Modified Newton-Raphson Iteration

Abbreviated Output: Case Study 1: Initial guess: x0 = 3

```
--- Inputs:
---   x0 =  3.00 ...
---   tolerance      =  0.00100 ...
---   max iterations =   100.00 ...
--- Execution:
---   Initial Conditions:
---     x0     -->  3.00000e+00 ...
---     f(x0)  -->  1.00000e+00 ...
---   Main Loop for Modified Newton Raphson Iteration:
---   Iteration 01: dx = -1.00000e+00, x = 2.00000e+00, f(x) -> 0.00000e+00
--- Output:
---   root =    2.00000 ...
---   f(root)    -->   0.00000000e+00 ...
---   df(root)   -->   0.00000000e+00 ...
---   ddf(root) -->   2.00000000e+00 ...
---   no iterations = 1 ...
---   converged: True ...
```

Modified Newton-Raphson Iteration

Abbreviated Output: Case Study 2: Initial guess: x0 = -3

```
--- Inputs:
---   x0 = -3.00 ...
---   tolerance      =  0.00100 ...
---   max iterations =   100.00 ...
--- Execution:
---   Initial Conditions:
---     x0     --> -3.00000e+00 ...
---     f(x0)  -->  2.50000e+01 ...
---   Main Loop for Modified Newton Raphson Iteration:
---   Iteration 01: dx = 5.00000e+00, x = 2.00000e+00, f(x) -> 0.00000e+00
--- Output:
---   root =    2.00000 ...
---   f(root)    -->   0.00000000e+00 ...
---   df(root)   -->   0.00000000e+00 ...
---   ddf(root) -->   2.00000000e+00 ...
---   no iterations = 1 ...
---   converged: True ...
```

Python Code Listings

Code 2: Newton Raphson Algorithm

```python
# ==============================================================================
# Calculate the root of an equation by the Newton Raphson method.
#
# Args: f (function): equation f(x).
#       df (function): derivative of quation f(x).
#       x0 (float): initial guess.
#       toler (float): tolerance (stopping criterion).
#       iter_max (int): maximum number of iterations (stopping criterion).
#
# Returns:
#       root (float): root value.
#       iter (int): number of iterations used by the method.
#       converged (boolean): flag to indicate if the root was found.
# ==============================================================================

import math

def newtonraphson(f, df, x0, toler, iter_max):

    fx  = f(x0)
    dfx = df(x0)
    x   = x0

    print("---   Initial Conditions: ")
    print("---     x0     --> {:12.5e} ...".format( x0 ) );
    print("---     f(x0)  --> {:12.5e} ...".format( f(x0) ) );
    print("---     df(x0) --> {:12.5e} ...".format( df(x0) ) );
```

Code 2: Newton Raphson Algorithm

```
    print("---   Main Loop for Newton Raphson Iteration: ")

    converged = False
    for i in range(1, iter_max + 1):

        # Compute update to root estimate ...

        delta_x = -fx / dfx
        x += delta_x
        fx  = f(x)
        dfx = df(x)

        print("---   Iteration {:03d}: dx = {:12.5e}, x = {:12.5e}, f(x) --> {:12.5e} ..

        # Check for convergence ...

        if math.fabs(delta_x) <= toler and math.fabs(fx) <= toler or dfx == 0:
            converged = True
            break

    root = x
    return root, i, converged
```

Code 3: Modified Newton Raphson

```
# ==========================================================================
# Calculate the root of an equation by the Modified Newton Raphson method.
#
# Args: f (function): equation f(x).
#       df (function): derivative of f(x).
#       ddf (function): second derivative of f(x).
#       x0 (float): initial guess.
#       toler (float): tolerance (stopping criterion).
#       iter_max (int): maximum number of iterations (stopping criterion).
#
# Returns:
#       root (float): root value.
#       iter (int): number of iterations used by the method.
#       converged (boolean): flag to indicate if the root was found.
# ==========================================================================

import math

def modifiednewtonraphson(f, df, ddf, x0, toler, iter_max):

    fx   = f(x0)
    dfx  = df(x0)
    ddfx = ddf(x0)
    x = x0

    print("---   Initial Conditions: ")
    print("---     x0     --> {:12.5e} ...".format( x0 ) );
    print("---     f(x0)  --> {:12.5e} ...".format( f(x0) ) );
```

Code 3: Modified Newton Raphson

```python

    print("---   Main Loop for Modified Newton Raphson Iteration: ")

    converged = False
    for i in range(1, iter_max + 1):

        # Compute update to root estimate ...

        delta_x = -((fx*dfx)/(dfx*dfx - fx*ddfx))
        x        = x + delta_x
        fx    = f(x)
        dfx   = df(x)
        ddfx = ddf(x)

        print("---   Iteration {:03d}: dx = {:12.5e}, x = {:12.5e}, f(x) --> {:12.5e} ..

        # Check for convergence ...

        if math.fabs(delta_x) <= toler and math.fabs(fx) <= toler or dfx == 0:
            converged = True
            break

    root = x
    return root, i, converged
```