

PART I

Lecture Notes on *Numerical Solution of* *Root-Finding Problems*

MATH 435

Professor Biswa Nath Datta
Department of Mathematical Sciences
Northern Illinois University
DeKalb, IL. 60115 USA

E-mail: dattab@math.niu.edu

URL: www.math.niu.edu/~dattab

PART I.

Numerical Methods for Root Finding Problem

1 PART I: Numerical Methods for the Root-Finding Problem

1.1 Introduction

In this lecture, we will discuss numerical methods for the **Root-Finding Problem**. As the title suggests, the Root-Finding Problem is the problem of finding a root of the equation $f(x) = 0$, where $f(x)$ is a function of a single variable x . Specifically, the problem is stated as follows:

The Root-Finding Problem
Given a function $f(x)$, find $x = \xi$ such that $f(\xi) = 0$.

The number ξ is called a **root** of $f(x) = 0$ or a **zero** of the function $f(x)$. The function of $f(x)$ may be algebraic or trigonometric functions. **The well-known examples of algebraic functions are polynomials.** A polynomial of degree n is written as $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$. The numbers $a_i, i = 0, \dots, n$ are called the coefficients of the polynomial. If $a_n = 1$, the polynomial $P_n(x)$ is called the **monic polynomial**.

Applications of the Root-Finding Problem

The root-finding problem is one of the most important computational problems. It arises in a wide variety of practical applications in **physics, chemistry, biosciences, engineering**, etc. As a matter of fact, determination of any unknown appearing implicitly in scientific or engineering formulas, gives rise to a root-finding problem. We consider one such simple application here.

(INCOMPLETE)

1.2 Bisection-Method

As the title suggests, the method is based on repeated bisections of an interval containing the root. The basic idea is very simple.

Basic-Idea:

Suppose $f(x) = 0$ is known to have a real root $x = \xi$ in an interval $[a, b]$.

- Then **bisect** the interval $[a, b]$, and let $c = \frac{a+b}{2}$ be the middle point of $[a, b]$. If c is the root, then we are done. Otherwise, one of the intervals $[a, c]$ or $[c, b]$ will contain the root.

- Find the one that contains the root and bisect that interval again.

- Continue the process of bisections until the root is trapped in an interval as small warranted by the desired accuracy.

To implement the idea, we need to know which of the two intervals in each iteration contains the root of $f(x) = 0$. The **Intermediate Mean-Value Theorem** of calculus can help us identify the interval in each iteration. For a proof of this theorem, see any calculus book.

Intermediate Mean-Value Theorem (IMV Theorem)

Let $f(x)$ be a continuous function defined on $[a, b]$, such that

$f(a)$ and $f(b)$ are of opposite signs (i.e., $f(a)f(b) < 0$).

Then there is a root $x = c$ of $f(x) = 0$ in $[a, b]$.

Algorithm 1.1 The Bisection Method for Rooting-Finding

Inputs: (i) $f(x)$ - The given function

(ii) a_0, b_0 - The two numbers such that $f(a_0)f(b_0) < 0$.

Output: An approximation of the root of $f(x) = 0$ in $[a_0, b_0]$.

For $k = 0, 1, 2, \dots$, do until satisfied:

- Compute $c_k = \frac{a_k + b_k}{2}$.
- Test if c_k is the desired root, if so, stop.
- If c_k is not the desired root, test if $f(c_k)f(a_k) < 0$. If so, set $b_{k+1} = c_k$ and $a_{k+1} = a_k$. Otherwise, set $a_{k+1} = c_k$, $b_{k+1} = b_k$.

End.

Example 1.1 $f(x) = x^3 - 6x^2 + 11x - 6$.

Let $a_0 = 2.5$, $b_0 = 4$. Then $f(a_0)f(b_0) < 0$. Then there is a root in $[2.5, 4]$.

Iteration 1. $k = 0$:

$$c_0 = \frac{a_0 + b_0}{2} = \frac{4 + 2.5}{2} = \frac{6.5}{2} = 3.25.$$

Since $f(c_0)f(a_0) = f(3.25)f(2.5) < 0$, set $b_1 = c_0$, $a_1 = a_0$.

Iteration 2. $k = 1$:

$$c_1 = \frac{3.25 + 2.5}{2} = 2.8750.$$

Since $f(c_1)f(a_1) > 0$, set $a_2 = 2.875$, $b_2 = b_1$

Iteration 3. $k = 2$:

$$c_2 = \frac{a_2 + b_2}{2} = \frac{2.875 + 3.250}{2} = 3.0625$$

Since $f(c_2)f(a_2) = f(3.0625)f(2.875) < 0$, set $b_3 = c_2$, $a_3 = a_2$.

Iteration 4. $k = 3$

$$c_3 = \frac{a_3 + b_3}{2} = \frac{2.875 + 3.0625}{2} = 2.9688.$$

It is clear that the iterations are converging towards the root $x = 3$.

Notes:

1. From the statement of the Bisection algorithm, it is clear that **the algorithm always converges**.
2. The example above shows that the convergence, however, can be very slow.

Stopping Criteria

Since this is an iterative method, we must determine some stopping criteria that will allow the iteration to stop. Here are some commonly used stopping criteria.

Let ϵ be the tolerance; that is, we would like to obtain the root with an error of at most of ϵ . Then

Accept $x = c_k$ as a root of $f(x) = 0$ if any of the following criteria is satisfied:

1. $|f(c_k)| \leq \epsilon$ (*The functional value is less than or equal to the tolerance*).
2. $\frac{|c_{k-1} - c_k|}{|c_k|} \leq \epsilon$ (*The relative change is less than or equal to the tolerance*).
3. $\frac{(b - a)}{2^k} \leq \epsilon$ (*The length of the interval after k iterations is less than or equal to the tolerance*).
4. *The number of iterations k is greater than or equal to a predetermined number, say N .*

Comments:

1. Criterion 1 can be misleading, since, it is possible to have $|f(c_k)|$ very small, even if c_k is not close to the root. **(Do an example to convince yourself that it is true).**
2. Criterion 3 is based on the fact that after k steps, the root will be computed with error at most $\left(\frac{b_0 - a_0}{2^k}\right)$.

Number of Iterations Needed in the Bisection Method to Achieve Certain Accuracy

Let's now find out what is the minimum number of iterations N needed with the Bisection method to achieve a certain desired accuracy.

Criterion 3 can be used to answer this. The interval length after N iterations is $\frac{b_0 - a_0}{2^N}$.

So, to obtain an accuracy of ϵ , using the Criterion 3, we must have $\frac{b_0 - a_0}{2^N} \leq \epsilon$.

That is, $2^{-N}(b_0 - a_0) \leq \epsilon$

$$\text{or } 2^{-N} \leq \frac{\epsilon}{(b_0 - a_0)}$$

$$\text{or } -N \log_{10} 2 \leq \log_{10} \left(\frac{\epsilon}{b_0 - a_0} \right)$$

$$\text{or } N \log_{10} 2 \geq -\log_{10} \left(\frac{\epsilon}{b_0 - a_0} \right).$$

$$\text{or } N \geq \frac{-\log_{10} \left(\frac{\epsilon}{b_0 - a_0} \right)}{\log_{10} 2}.$$

$$\text{or } N \geq \frac{[\log_{10}(b_0 - a_0) - \log_{10}(\epsilon)]}{\log_{10} 2}$$

Theorem 1.1 *The number of iterations N needed in the bisection method to obtain an accuracy of ϵ is given by*

$$N \geq \frac{[\log_{10}(b_0 - a_0) - \log_{10}(\epsilon)]}{\log_{10} 2}$$

■

Example 1.2 *Suppose we would like to determine a priori the minimum number of iterations needed in the bisection algorithm, given $a_0 = 2.5$, $b_0 = 4$, and $\epsilon = 10^{-3}$. By Theorem 1.1, we have*

$$N \geq \frac{\log_{10}(1.5) - \log_{10}(10^{-3})}{\log_{10} 2} = \frac{\log_{10}(1.5) + 3}{\log_{10}(2)} = 10.5507.$$

Thus, a minimum of 11 iterations will be needed to obtain the desired accuracy using the Bisection Method.

Remarks: Since the number of iterations N needed to achieve a certain accuracy depends upon the initial length of the interval containing the root, it is desirable to choose the initial interval $[a_0, b_0]$ as small as possible.

1.3 Fixed-Point Iteration

A number ξ is a **fixed point** of a function $g(x)$ if $g(\xi) = \xi$.

Suppose that the equation $f(x) = 0$ is written in the form $x = g(x)$; that is, $f(x) = x - g(x) = 0$. Then any fixed point ξ of $g(x)$ is a root of $f(x) = 0$; because $f(\xi) = \xi - g(\xi) = \xi - \xi = 0$. Thus a root of $f(x) = 0$ can be found by finding a fixed point of $x = g(x)$, which corresponds to $f(x) = 0$.

Finding a root of $f(x) = 0$ by finding a fixed point of $x = g(x)$ immediately suggests an iterative procedure of the following type.

Start with an initial guess x_0 of the root, and form a sequence $\{x_k\}$ defined by

$$x_{k+1} = g(x_k), \quad k = 0, 1, 2, \dots$$

If the sequence $\{x_k\}$ converges, then $\lim_{k \rightarrow \infty} x_k = \xi$ will be a root of $f(x) = 0$.

The question therefore rises:

Given $f(x) = 0$, how to write $f(x) = 0$ in the form $x = g(x)$, so that starting with any x_0 in $[a, b]$, the sequence $\{x_k\}$ defined by $x_{k+1} = g(x_k)$ is guaranteed to converge?

The simplest way to write $f(x) = 0$ in the form $x = g(x)$ is to add x on both sides, that is,

$$x = f(x) + x = g(x).$$

But it does not very often work.

To convince yourself, consider **Example 1.1** again. Here

$$f(x) = x^3 - 6x^2 + 11x - 6 = 0.$$

Define $g(x) = x + f(x) = x^3 - 6x^2 + 12x - 6$. We know that there is a root of $f(x)$ in $[2.5, 4]$; namely $x = 3$. Let's start the iteration $x_{k+1} = g(x_k)$ with $x_0 = 3.5$,

then we have:

$$\begin{aligned} x_1 &= g(x_0) = g(3.5) = 5.3750 \\ x_2 &= g(x_1) = g(5.3750) = 40.4434 \\ x_3 &= g(x_2) = g(40.4434) = 5.6817 \times 10^4 \\ x_4 &= g(x_3) = g(5.6817 \times 10^4) = 1.8340 \times 10^{14} \end{aligned}$$

The sequence $\{x_k\}$ is clearly diverging.

The convergence and divergence of the fixed-point iteration are illustrated by the following graphs.

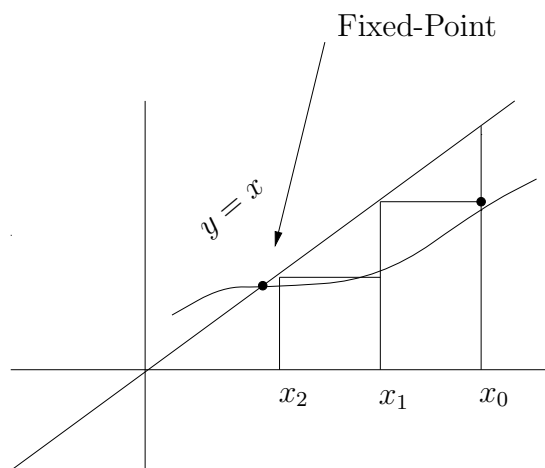


Figure 1.1: Convergence of the Fixed-Point Iteration

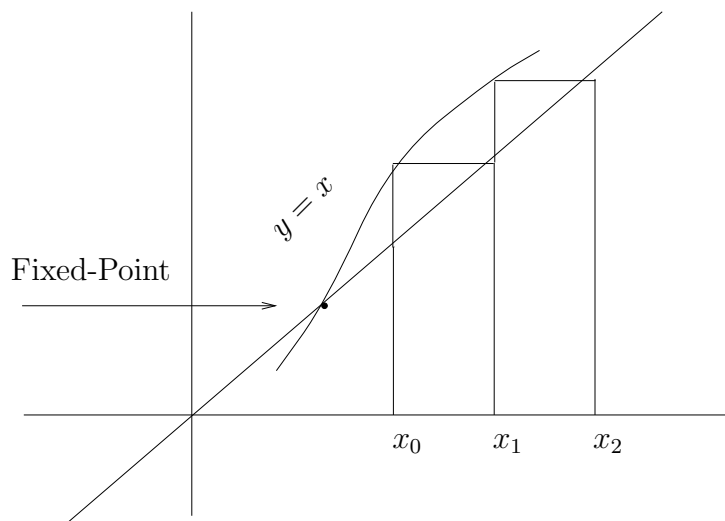


Figure 1.2: Divergence of the Fixed-Point Iteration

The following theorem gives a sufficient condition on $g(x)$ which ensures the convergence of the sequence $\{x_k\}$.

Theorem 1.2 (Fixed-Point Iteration Theorem): Let $f(x) = 0$ be written in the form $x = g(x)$. Assume that $g(x)$ has the following properties:

1. For all x in $[a, b]$, $g(x) \in [a, b]$; that is $g(x)$ takes every value between a and b .

2. $g'(x)$ exists on (a, b) with the property that there exists a positive constant $r < 1$ such that

$$|g'(x)| \leq r,$$

for all x in (a, b) .

Then

(i) there is a unique fixed point $x = \xi$ of $g(x)$ in $[a, b]$.

(ii) For any x_0 in $[a, b]$, the sequence $\{x_k\}$ defined by

$$x_{k+1} = g(x_k), \quad k = 0, 1, \dots$$

converges to the fixed point $x = \xi$; that is to the root ξ of $f(x) = 0$.



The proof of the Theorem requires the **Mean Value Theorem** of Calculus;

The Mean Value Theorem (MVT)

Let $f(x)$ be a continuous function on $[a, b]$, and be differentiable on (a, b) .

Then there is a number c in (a, b) such that $\frac{f(b) - f(a)}{b - a} = f'(c)$.

Proof of the Fixed-Point Theorem:

The proof comes in three parts: **Existence, Uniqueness, and Convergence**. We first prove that there is a root of $f(x) = 0$ in $[a, b]$ and it is unique. Since a fixed point of $x = g(x)$ is a root of $f(x) = 0$, this amounts to proving that there is a fixed point in $[a, b]$ and it is unique. We then prove that the sequence $x_{k+1} = g(x_k)$ converges to the root.

Existence:

If $a = g(a)$, then a is a fixed point. If $b = g(b)$, then b is a fixed point. If not, **because of the assumption 1**, $g(a) > a$, and $g(b) < b$. Thus, $f(a) = g(a) - a > 0$, and $f(b) = g(b) - b < 0$. Also $f(x) = g(x) - x$ is continuous on $[a, b]$.

Thus, by applying the **Intermediate Mean Value Theorem** to $f(x)$ in $[a, b]$, we conclude that there is a root of $f(x)$ in $[a, b]$. **This proves the existence.**

Uniqueness (by contradiction):

To prove uniqueness, suppose ξ_1 and ξ_2 are for two fixed points in $[a, b]$, and $\xi_1 \neq \xi_2$. Now, apply the MVT to $g(x)$ in $[\xi_1, \xi_2]$, we can find a number c in (ξ_1, ξ_2) such that

$$\frac{g(\xi_2) - g(\xi_1)}{(\xi_2 - \xi_1)} = g'(c).$$

Since $g(\xi_1) = \xi_1$, and $g(\xi_2) = \xi_2$, we have

$$\frac{\xi_2 - \xi_1}{\xi_2 - \xi_1} = g'(c).$$

That is, $g'(c) = 1$, which is a contradiction to our assumption 2. Thus, ξ_1 can not be different from ξ_2 .

Convergence:

Let ξ be the root of $f(x) = 0$. Then the **absolute error** at step $k + 1$ is given by $e_{k+1} = |\xi - x_{k+1}|$. To prove that the sequence converges, we need to show that $\lim_{k \rightarrow \infty} e_{k+1} = 0$.

To show this, we apply the Mean Value Theorem to $g(x)$ in $[x_k, \xi]$.

Then $\frac{g(\xi) - g(x_k)}{\xi - x_k} = g'(c)$

Where $x_k < c < \xi$.

That is, $\frac{\xi - x_{k+1}}{\xi - x_k} = g'(c)$ (note that $g(\xi) = \xi$ and $x_{k+1} = g(x_k)$)

Taking absolute values on both sides we have

$$\frac{|\xi - x_{k+1}|}{|\xi - x_k|} = |g'(c)|$$

or

$$\frac{e_{k+1}}{e_k} = |g'(c)|$$

Since $|g'(c)| \leq r$, we have $e_{k+1} \leq r e_k$.

Similarly, we have $e_k \leq r e_{k-1}$. Thus $e_{k+1} \leq r^2 e_{k-1}$.

Continuing in this way, we finally have $e_{k+1} \leq r^{k+1} e_0$, where e_0 is the initial error. (That is, $e_0 = |x_0 - \xi|$).

Since $r < 1$, we have $r^{k+1} \rightarrow 0$ as $k \rightarrow \infty$.

Thus,

$$\begin{aligned} \lim_{k \rightarrow \infty} e_{k+1} &= \lim_{k \rightarrow \infty} |x_{k+1} - \xi| \\ &\leq \lim_{k \rightarrow \infty} r^{k+1} e_0 \\ &= 0. \end{aligned}$$

This proves that the sequence $\{x_k\}$ converges $x = \xi$ and that $x = \xi$ is the only fixed point of $g(x)$. ■

In practice, it is not easy to verify Assumption 1. The following corollary of the fixed-point theorem is more useful in practice because the conditions here are more easily verifiable. We leave the proof of the corollary as an **[Exercise]**.

Corollary 1.1 *Let $g(x)$ be continuously differentiable in some open interval containing the fixed point ξ , and let $|g'(\xi)| < 1$, then there exists a number $\epsilon > 0$ so that the iteration $x_{k+1} = g(x_k)$ converges whenever x_0 is chosen in $|x_0 - \xi| \leq \epsilon$.*

Example 1.3 *Let $f(x) = x^3 - 6x^2 + 11x - 6 = 0$. Choose $a = 2.5$, $b = 4$. Let's write $f(x) = 0$ in the form $x = g(x)$ as follows: $x = \frac{1}{11}(-x^3 + 6x^2 + 6) = g(x)$.*

Then

$$g'(x) = \frac{1}{11}(-3x^2 + 12x).$$

It is easy to verify graphically and analytically that $|g'(x)|$ is less than 1 for all x is $(2.5, 4)$, (note that $g'(4) = 0$, and $g'(2.5) = 1.0227$).

So, by Corollary 1.1, the iteration $x_{k+1} = g(x_k)$ should converge to the root $x = 3$ for any $x_0 \in [2.5, 4]$. This is verified as follows:

$$\begin{aligned} x_0 &= 3.5 \\ x_1 = g(x_0) &= 3.3295 \\ x_2 = g(x_1) &= 3.2367 \\ x_3 = g(x_2) &= 3.1772 \\ x_4 = g(x_3) &= 3.1359 \\ x_5 = g(x_4) &= 3.1059 \\ x_6 = g(x_5) &= 3.0835 \\ x_7 = g(x_6) &= 3.0664 \\ x_8 = g(x_7) &= 3.0531 \\ x_9 = g(x_8) &= 3.0427 \\ x_{10} = g(x_9) &= 3.0344 \\ x_{11} = g(x_{10}) &= 3.0278 \end{aligned}$$

The sequence is clearly converging to the root $x = 3$. This can be justified from the above theorem.

Example 1.4 *Find a root of $f(x) = x - \cos x = 0$ in $[0, \frac{\pi}{2}]$.*

Let's write $x - \cos x = 0$ as $x = \cos x = g(x)$. Then $|g'(x)| = |\sin x| < 1$ in $(0, \frac{\pi}{2})$. Thus, by Corollary 1.1 should converge. Indeed,

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= \cos x_0 = 1 \\
x_2 &= \cos x_1 = 0.54 \\
x_3 &= \cos x_2 = 0.86 \\
&\vdots \\
x_{17} &= 0.73956 \\
x_{18} &= \cos x_{17} = 0.73955
\end{aligned}$$

1.4 The Newton-Raphson Method

The Newton-Raphson method, described below, shows that there is a special choice of $g(x)$ that will allow us to readily use the results of Corollary 1.1.

Assume that $f''(x)$ exists and is continuous on $[a, b]$ and ξ is a **simple root** of $f(x)$, that is, $f(\xi) = 0$ and $f'(\xi) \neq 0$.

Choose

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

$$\text{Then } g'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}.$$

$$\text{Thus, } g'(\xi) = \frac{f(\xi)f''(\xi)}{(f'(\xi))^2} = 0, \text{ since } f(\xi) = 0, \text{ and } f'(\xi) \neq 0.$$

Since $g'(x)$ is continuous, this means that there exists a small neighborhood around the root $x = \xi$ such that for all points x in that neighborhood, $|g'(x)| < 1$.

Thus if $g(x)$ is chosen as above and the **starting approximation** x_0 is chosen sufficiently **close to the root** $x = \xi$, then the fixed-point iteration is guaranteed to converge, according to the corollary above.

This leads to the following well-known classical method, known as, the **Newton-Raphson** method.

(Remark: In many text books and literature, the Newton-Raphson Method is simply called the **Newton Method**. But, the title “The Newton-Raphson Method” is justified. See an article by [] in SIAM Review (1996)).

Algorithm 1.2 *The Newton-Raphson Method*

Inputs: $f(x)$ - The given function
 x_0 - The initial approximation
 ϵ - The error tolerance
 N - The maximum number of iterations

Output: An approximation to the root $x = \xi$ or a message of failure.

Assumption: $x = \xi$ is a simple root of $f(x)$.

For $k = 0, 1, \dots$ do until convergence or failure.

- Compute $f(x_k), f'(x_k)$.

- Compute $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

- Test for convergence or failure:

$$\left. \begin{array}{l} \text{If } |f(x_k)| < \epsilon \\ \text{or } \frac{|x_{k+1} - x_k|}{|x_k|} < \epsilon \end{array} \right] \text{stopping criteria.}$$

or $k > N$, Stop.

End

Definition 1.1 *The iterations $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ are called **Newton's iterations**.*

Remarks:

1. If none of the above criteria has been satisfied within a predetermined, say N , iterations then the method has failed after the prescribed number of iterations. In that case, one could try the method again with a different x_0 .
2. A judicious choice of x_0 can sometimes be obtained by drawing the graph of $f(x)$, if possible. However, there does not seem to exist a clear-cut guideline of how to choose a right starting point x_0 that guarantees the convergence of the Newton-Raphson Method to a desired root.

Some Familiar Computations Using the Newton-Raphson Method

1. **Computing the Square Root of a Positive Number A:** Compute \sqrt{A} , where $A > 0$.

Computing \sqrt{A} is equivalent to solving $x^2 - A = 0$. The number \sqrt{A} , thus, may be computed by applying the Newton-Raphson Method to $f(x) = x^2 - A$.

Since $f'(x) = 2x$, we have the following Newton iterations to generate $\{x_k\}$:

Newton-Raphson Iterations for Computing \sqrt{A}

Input: A - A positive number
Output: An approximation to \sqrt{A} .
Step 1. Guess an initial approximation x_0 to \sqrt{A} .
Step 2. Compute the successive approximations $\{x_k\}$ as follows:
 For $k = 0, 1, 2, \dots$, do until convergence

$$x_{k+1} = x_k - \frac{x_k^2 - A}{2x_k} = \frac{x_k^2 + A}{2x_k}$$

End

Example 1.5 Let $A = 2$, $x_0 = 1.5$

Iteration 1. $x_1 = \frac{x_0^2 + A}{2x_0} = \frac{(1.5)^2 + 2}{3} = 1.4167$

Iteration 2. $x_2 = \frac{x_1^2 + A}{2x_1} = 1.4142$

Iteration 3. $x_3 = \frac{x_2^2 + A}{2x_2} = 1.4142$

2. **Computing the nth Root**

It is easy to see that the above Newton-Raphson Method to compute \sqrt{A} can be easily generalized to compute $\sqrt[n]{A}$. In this case $f(x) = x^n - A$, $f'(x) = nx^{n-1}$.

Thus Newton's Iterations in this case are:

$$x_{k+1} = x_k - \frac{x_k^n - A}{nx_k^{n-1}} = \frac{(n-1)x_k^n + A}{nx_k^{n-1}}.$$

3. Finding the Reciprocal of a Nonzero Number A

The problem here is to compute $\frac{1}{A}$, where $A \neq 0$. Again, the Newton-Raphson Method can be applied with $f(x) = \frac{1}{x} - A = 0$. We have $f'(x) = -\frac{1}{x^2}$. The sequence x_{k+1} is given by

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{\frac{1}{x_k} - A}{-\frac{1}{x_k^2}} = x_k(2 - Ax_k).$$

The Newton-Raphson Iteration for Computing $\frac{1}{A}$

Input: A - A nonzero number
Output: An approximation to $\frac{1}{A}$.
Step 1. Guess an initial approximation x_0 to $\frac{1}{A}$.
Step 2. For $k = 0, 1, 2, \dots$ do until convergence

$$x_{k+1} = x_k(2 - Ax_k)$$

End

Example 1.6 Let $A = 3$

The problem is to find $\frac{1}{3}$

$$f(x) = \frac{1}{x} - 3.$$

Take $x_0 = 0.25$

$$\text{Then } x_1 = x_0(2 - Ax_0) = 0.3125$$

$$x_2 = x_1(2 - Ax_1) = 0.3320$$

$$x_3 = x_2(2 - Ax_2) = 0.3333$$

Note that 0.3333 is the 4-digit approximation of $\frac{1}{3}$ (correct up to 4 decimal figures).

A Geometric Interpretation of the Newton-Raphson Method

- The first approximation x_1 can be viewed as the x -intercept of the tangent line to the graph of $f(x)$ at $(x_0, f(x_0))$.

- The second approximation x_2 is the x -intercept of the tangent line to the graph of $f(x)$ at $(x_1, f(x_1))$. And so on.

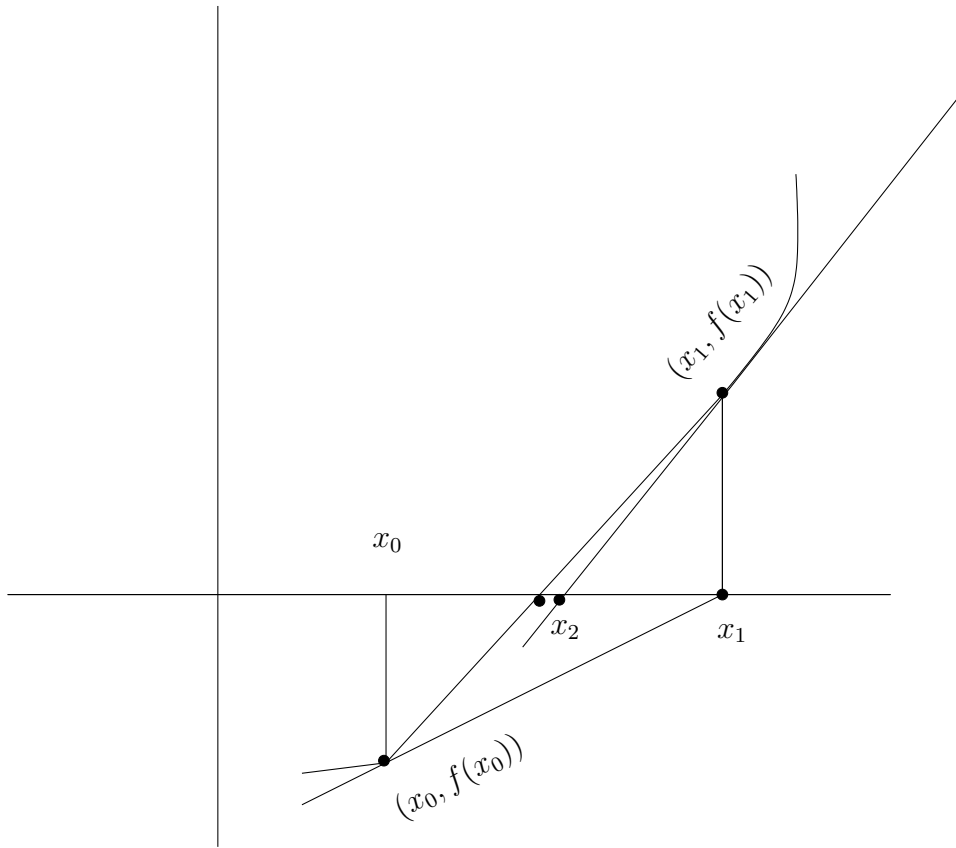


Figure 1.3: Graphical Representation of the Newton-Raphson Method.

1.5 The Secant Method

A major disadvantage of the Newton-Raphson Method is the requirement of finding the value of the derivative of $f(x)$ at each approximation. There are functions for which this job is either extremely difficult (if not impossible) or time consuming. A way-out is to approximate the derivative by knowing the values of the function at that and the previous approximation. Knowing $f(x_k)$ and $f(x_{k-1})$, we can approximate $f'(x_k)$ as:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Then, the Newton iterations in this case are:

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)}{f'(x_k)} \\ &\approx x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} \end{aligned}$$

This consideration leads to the **Secant method**.

Algorithm 1.3 *The Secant Method*

Inputs: $f(x)$ - The given function
 x_0, x_1 - The two initial approximations of the root
 ϵ - The error tolerance
 N - The maximum number of iterations

Output: An approximation of the exact solution ξ or a message of failure.

For $k = 1, 2, \dots$ do

- Compute $f(x_k)$ and $f(x_{k-1})$.
- Compute the next approximation: $x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$
- Test for convergence or maximum number of iterations: If $|x_{k+1} - x_k| < \epsilon$ or if $k > N$, Stop.

End

Note: The secant method needs two approximations x_0, x_1 to start with, whereas the Newton-Raphson method just needs one, namely, x_0 .

Example 1.7 $f(x) = x^2 - 2 = 0$.

Let $x_0 = 1.5, x_1 = 1$

Iteration 1. $k = 1$.

$$\begin{aligned}x_2 &= x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= 1 - \frac{(-1)(-.5)}{-1 - .25} = 1 + \frac{.5}{1.25} = 1.4\end{aligned}$$

Iteration 2

$$\begin{aligned}
x_3 &= x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} \\
&= 1.4 - \frac{(-0.04)(0.4)}{(-0.04) - (-1)} \\
&= 1.4 + 0.0167 \\
&= 1.4167
\end{aligned}$$

Iteration 3

$$\begin{aligned}
x_4 &= x_3 - \frac{f(x_3)(x_3 - x_2)}{f(x_3) - f(x_2)} \\
&= 1.4142.
\end{aligned}$$

Iteration 4

$$x_5 = x_4 - \frac{f(x_4)(x_4 - x_3)}{f(x_4) - f(x_3)} = 1.4142$$

Remarks: By comparing the results of this example with those obtained by the Newton-Raphson (**Example 1.5**) method, we see that *it took 4 iterations by the secant method to obtain a 4-digit accuracy of $\sqrt{2} = 1.4142$; whereas for the Newton-Raphson method, this accuracy was obtained just after 2 iteration.*

In general, **the Newton-Raphson method converges faster than the secant method.** The exact rate of convergence of these two and the other methods will be discussed in the next section.

A Geometric Interpretation of the Secant Method

- x_2 is the x -intercept of the secant line passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$.
- x_3 is the x -intercept of the secant-line passing through $(x_1, f(x_1))$ and $(x_2, f(x_2))$.
- And so on.

Note: That is why the method is called the secant method.

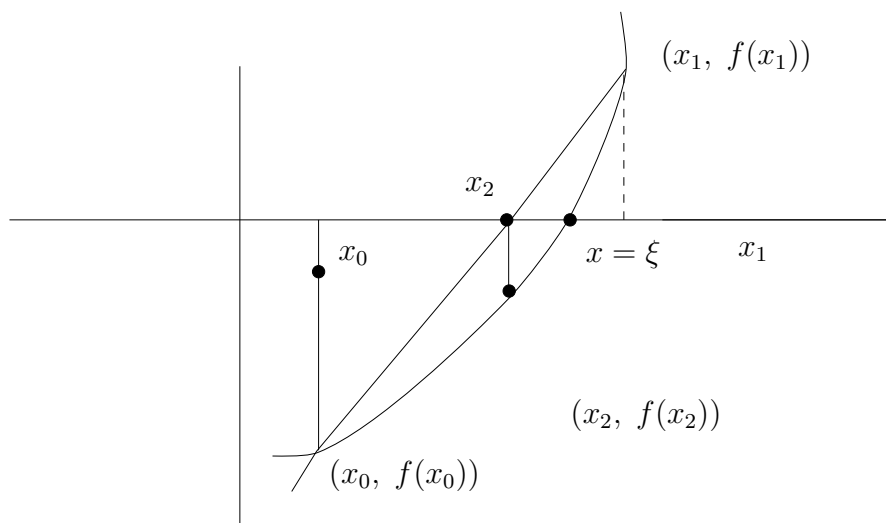


Figure 1.4: Geometric Interpretation of the Secant Method.

1.6 The Method of False Position (The Regular Falsi Method)

In the Bisection method, every interval under consideration is guaranteed to have the desired root $x = \xi$. That is why, the Bisection method is often called a **bracket method**, because every interval brackets the root. However, the Newton-Raphson method and the secant method are not bracket methods in that sense, because there is no guarantee that the two successive approximations will bracket the root. **Here is an example.**

Example 1.8 Consider solving $f(x) = \tan(\pi x) - 6 = 0$, using the **Secant Method**.

Take: $x_0 = 0$ $x_1 = 0.48$

(Note that $\xi = 0.44743154$ is a root of $f(x) = 0$ lying in the interval $[0, 0.48]$).

Iteration 1

$$x_2 = x_1 - f(x_1) \frac{(x_1 - x_0)}{f(x_1) - f(x_0)} = 0.181194$$

Iteration 2

$$x_0 = 0.48, \quad x_1 = 0.181194$$

$$x_2 = 0.286187$$

Iteration 3

$$x_0 = 0.181194, \quad x_1 = 0.286187$$

$x_2 = 1.091987$. Clearly, *The iterations are diverging.*

However, *the secant method can easily be modified to make it a bracket method.* The idea is as follows:

- Choose the initial approximations x_0 and x_1 such that $f(x_0)f(x_1) < 0$, ensuring that the root is in $[x_0, x_1]$.
- Compute x_2 as in the Secant method; that is, take x_2 as the x -intercept of the secant line passing through x_0 and x_1 .
- Compute x_3 out of the three approximation x_0, x_1 , and x_2 as follows:
First, compute $f(x_1)f(x_2)$; if it is negative, then $[x_1, x_2]$ brackets the root and x_3 is the x -intercept of the secant line joining $(x_1, f(x_1))$ and $(x_2, f(x_2))$ (as in the Secant method). Otherwise, x_3 will be computed as the x -intercept of the line joining $(x_0, f(x_0))$ and $(x_2, f(x_2))$.
- Continue the process.
- This yields a method called, the **Method of False Position** or the **Method of Regula Falsi**.

Algorithm 1.4 *The Method of False Position*

Input: (i) $f(x)$ - The given function
(ii) x_0, x_1 - The two initial approximations $f(x_0)f(x_1) < 0$.
(iii) ϵ - The tolerance
(iv) N The maximum number of iterations

Output: An approximation to the root $x = \xi$.

Step 0. Set $i = 1$.

Step 1. Compute $x_{i+1} = x_i - f(x_i) \frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$

If $|\frac{x_{i+1} - x_i}{x_i}| < \epsilon$ or $i \geq N$. Stop.

Step 2. If $f(x_i)f(x_{i+1}) < 0$, then, set $x_{i+1} \equiv x_i, x_i \equiv x_{i-1}$.

Otherwise, set $x_{i+1} \equiv x_i, x_{i+1} \equiv x_i$. Return to Step 0.

End

Example 1.9 Consider again the previous example: (**Example 1.8**)

$$f(x) = \tan(\pi x) - 6 = 0.$$

Let the interval be $[0, 0.48]$.

Then $\xi = 0.44743154$ is the root of $f(x)$ in $[0, 0.48]$.

Let's start with the same initial approximations:

$$x_0 = 0, \quad x_1 = 0.48; \quad f(x_0)f(x_1) < 0$$

Iteration 1

$$x_2 = x_1 - f(x_1) \frac{x_1 - x_0}{f(x_1) - f(x_0)} = 0.181192$$

$$f(x_2)f(x_1) < 0$$

$$\text{Set } x_0 = 0.48, \quad x_1 = 0.181192$$

Iteration 2

$$x_2 = x_1 - f(x_1) \frac{x_1 - x_0}{f(x_1) - f(x_0)} = 0.286186$$

$$f(x_2)f(x_1) > 0$$

$$\text{Set } x_0 = 0.48, \quad x_1 = 0.286186$$

Iteration 3

$$x_2 = 0.348981$$

$$f(x_2)f(x_1) > 0$$

$$\text{Set } x_0 = 0.48, \quad x_1 = 0.348981$$

Iteration 4

$$x_2 = 0.387053$$

Iteration 5

$$x_2 = 0.410305$$

1.7 Convergence Analysis of the Iterative Methods

We have seen that the Bisection method always converges, and each of the methods: the Newton-Raphson Method, the Secant method, and the method of False Position, converges under certain conditions. The question now arises: **when a method converges, how fast it converges?** In other words, what is the rate of convergence? To this end, we first define:

Definition 1.2 (Rate of Convergence of an Iterative Method). *Suppose that the sequence $\{x_k\}$ converges to ξ . Then the sequence $\{x_k\}$ is said to converge to ξ with the order of convergence α if there exists a positive constant p such that*

$$\lim_{k \rightarrow \infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^\alpha} = \lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k^\alpha} = p.$$

Thus if $\alpha = 1$, the convergence is **linear**. If $\alpha = 2$, the convergence is **quadratic**; and so on. The number α is called the **convergence factor**.

Using the above definition, we will now show that the rate of convergence for the Fixed Point Iteration method is usually linear but that of the Newton-Raphson method is quadratic, and that of the Secant method is superlinear.

Rate of Convergence of the Fixed-Point Iteration Method

Recall that in the proof of the **Fixed-Point Iteration Theorem**, we established that

$$e_{k+1} = |g'(c)|e_k,$$

where c is a number between x_k and ξ .

$$\text{So, } \lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k} = \lim_{k \rightarrow \infty} |g'(c)|.$$

Since $\{x_k\} \rightarrow \xi$, and c is in (x_k, ξ) , it follows that $\lim_{k \rightarrow \infty} |g'(c)| = |g'(\xi)|$.

This gives $\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k} = |g'(\xi)|$. Thus $\alpha = 1$.

We, therefore, have

The fixed-point iteration

$$x_{k+1} = g(x_k), \quad k = 0, 1, \dots$$

converges linearly.

Rate of Convergence of the Newton-Raphson Method

To study the rate of convergence of the Newton-Raphson Method, we need Taylor's Theorem.

Taylor's Theorem of order n

Suppose that the function $f(x)$ possesses continuous derivatives of order up to $(n + 1)$ in the interval $[a, b]$, and p is a point in this interval. Then for every x in this interval, there exists a number c (depending upon x) between p and x such that

$$f(x) = f(p) + f'(p)(x - p) + \frac{f''(p)}{2!}(x - p)^2 \\ + \cdots + \frac{f^n(p)}{n!}(x - p)^n + R_n(x),$$

where $R_n(x)$, called the **remainder after n terms**, is given by:

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n + 1)!}(x - p)^{n+1}$$

Let's choose a small interval around the root $x = \xi$. Then for any x in this interval, we have, by Taylor's theorem of order 1, the following expansion of the function $g(x)$:

$$g(x) = g(\xi) + (x - \xi)g'(\xi) + \frac{(x - \xi)^2}{2!} g''(\eta_k)$$

where η_k lies between x and ξ .

Now, for the Newton-Raphson Method, we have seen that $g'(\xi) = 0$.

$$\text{So, } g(x) = g(\xi) + \frac{(x - \xi)^2}{2!} g''(\eta_k)$$

$$\text{or } g(x_k) = g(\xi) + \frac{(x_k - \xi)^2}{2!} g''(\eta_k).$$

$$\text{or } g(x_k) - g(\xi) = \frac{(x_k - \xi)^2}{2!} g''(\eta_k).$$

Since $g(x_k) = x_{k+1}$, and $g(\xi) = \xi$, this give

$$x_{k+1} - \xi = \frac{(x_k - \xi)^2}{2!} g''(\eta_k)$$

That is, $|x_{k+1} - \xi| = \frac{|x_k - \xi|^2}{2} |g''(\eta_k)|$

or $\frac{e_{k+1}}{e_k^2} = \frac{|g''(\eta_k)|}{2}$.

Since η_k lies between x and ξ for every k , it follows that

$$\lim_{k \rightarrow \infty} \eta_k = \xi.$$

So, we have $\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k^2} = \lim_{k \rightarrow \infty} \frac{|g''(\eta_k)|}{2} = \frac{|g''(\xi)|}{2}$.

This proves

The Newton-Raphson Method Converges Quadratically.

The quadratic convergence roughly means that the accuracy gets doubled at each iteration.

The Rate of Convergence for the Secant Method.

It can be shown [Exercise] that *if the Secant method converges, then the convergence factor α is approximately 1.6 for sufficiently large k* . This is said as “**The rate of convergence of the Secant method is superlinear**”.

1.8 A Modified Newton-Raphson Method for Multiple Roots

Recall that the most important underlying assumption in the proof of quadratic convergence of the Newton-Raphson method was that $f'(x)$ is not zero at the approximation $x = x_k$, and in particular $f'(\xi) \neq 0$. This means that ξ is a simple root of $f(x) = 0$.

The question, therefore, is:

How can the Newton-Raphson method be modified for a multiple root so that the quadratic convergence can be retained?

Note that if $f(x)$ has a root ξ of multiplicity m , then

$$f(\xi) = f'(\xi) = f''(\xi) = \dots = f^{(m-1)}(\xi) = 0,$$

where $f^{(m-1)}(\xi)$ denotes the $(m-1)$ th derivative of $f(x)$ at $x = \xi$. In this case $f(x)$ can be written as $f(x) = (x - \xi)^m h(x)$, where $h(x)$ is a polynomial of degree at most $m - 2$.

In case $f(x)$ has a root of multiplicity m , it was suggested by Ralston and Robinowitz (1978) that, the following modified iteration will ensure quadratic convergence in the Newton-Raphson method, under certain conditions [**Exercise**].

The Newton-Raphson Iterations for Multiple Roots:

Method I

$$x_{i+1} = x_i - \frac{mf(x_i)}{f'(x_i)}$$

Since in practice, the multiplicity m is not known apriori, another **useful modification is to apply the Newton-Raphson iteration to the function:**

$$u(x) = \frac{f(x)}{f'(x)}$$

in place of $f(x)$.

Thus, in this case, the modified Newton-Raphson iteration becomes $x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)}$.

Note that, since $f(x) = (x - \xi)^m h(x)$, $u'(x) \neq 0$ at $x = \xi$.

Since $u'(x_i) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2}$

we have

The Newton-Raphson Iterations for Multiple Roots

Method II

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - [f(x_i)]f''(x_i)}$$

Example:

$$\begin{aligned} f(x) &= e^x - x - 1 \\ f'(x) &= e^x - 1 \\ f''(x) &= e^x \end{aligned}$$

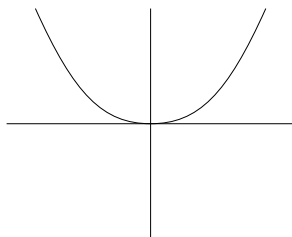


Figure 1.5: The Graph of $e^x - x - 1$.

There are two roots at $x = 0$ (See the graph above!). So, $m = 2$.

Method I

$$x_{i+1} = x_i - \frac{2f(x_i)}{f'(x_i)}$$

$$x_0 = 0.5$$

Iteration 1. $x_1 = 0.5 - \frac{2(e^{0.5} - 0.5 - 1)}{(e^{0.5} - 1)} = .0415 = 4.15 \times 10^{-2}$

Iteration 2. $x_2 = 0.0415 - \frac{2(e^{0.0415} - 0.0415 - 1)}{e^{0.0415} - 1} = .00028703 = 2.8703 \times 10^{-4}$.

Method II

$$f(x) = e^x - x - 1.$$

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{(f'(x_i))^2 - ((f(x_i)f''(x_i)))}$$

$$x_0 = 0.5$$

Iteration 1. $x_1 = -0.0493 = -4.93 \times 10^{-2}$

Iteration 2. $x_2 = -4.1180 \times 10^{-4}$

1.9 The Newton-Raphson Method for Polynomials

A major requirement of all the methods that we have discussed so far is the evaluation of $f(x)$ at successive approximations x_k .

If $f(x)$ happens to be a polynomial $P_n(x)$ (which is the case in many practical applications), then there is an extremely simple classical technique, known as **Horner's Method** to compute $P_n(x)$ at $x = z$. *The value $P_n(z)$ can be obtained recursively in n -steps from the coefficients of the polynomial $P_n(x)$. Furthermore, as a by-product, one can get the value of the derivative of $P_n(x)$ at $x = z$; that is $P'_n(z)$.* Then, the scheme can be incorporated in the Newton-Raphson Method to compute a zero of $P_n(x)$.

(Note that the Newton-Raphson Method requires evaluation of $P_n(x)$ and its derivate at successive iterations).

Horner's Method

Let $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ and let $x = z$ be given. We need to compute $P_n(z)$ and $P'_n(z)$.

Let's write $P_n(x)$ as:

$$P_n(x) = (x - z) Q_{n-1}(x) + b_0,$$

where $Q_{n-1}(x) = b_nx^{n-1} + b_{n-1}x^{n-2} + \dots + b_2x + b_1$.

Thus, $P_n(z) = b_0$.

Let's see how to compute b_0 recursively by knowing the coefficients of $P_n(x)$

From above, we have

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = (x - z)(b_nx^{n-1} + b_{n-1}x^{n-2} + \cdots + b_1) + b_0$$

Comparing the coefficients of like powers of x from both sides, we obtain

$$\begin{aligned} b_n &= a_n \\ b_{n-1} - b_n z &= a_{n-1} \Rightarrow b_{n-1} = a_{n-1} + b_n z \\ &\vdots \end{aligned}$$

and so on.

In general, $b_k - b_{k+1}z = a_k \Rightarrow b_k = a_k + b_{k+1}z$. $k = n - 1, n - 2, \dots, 1, 0$.

Thus, knowing the coefficients $a_n, a_{n-1}, \dots, a_1, a_0$ of $P_n(x)$, the coefficients b_n, b_{n-1}, \dots, b_1 of $Q_{n-1}(x)$ can be computed recursively starting with $b_n = a_n$, as shown above. That is, $b_k = a_k + b_{k+1}z$, $k = n - 1, k = 2, \dots, 1, 0$.

Again, note that $P_n(z) = b_0$. So, $P_n(z) = b_0 = a_0 + b_1 z$.

That is, if we know the coefficients of a polynomial, we can compute the value of the polynomial at a given point out of these coefficients recursively as shown above.

Algorithm 1.5 Evaluating Polynomial: $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$
at $x = z$.

Inputs: (1) a_0, a_1, \dots, a_n of $P_n(x)$ -The coefficients of the polynomial $P_n(x)$.

(2) z -The number at which $P_n(x)$ has to be evaluated.

Outputs: $b_0 = P_n(z)$.

Step 1. Set $b_n = a_n$.

Step 2. For $k = n - 1, n - 2, \dots, 1, 0$ do.

Compute $b_k = a_k + b_{k+1}z$

Step 3. Set $P_n(z) = b_0$

End

It is interesting to note that as a by-product of above, we also obtain $P'_n(z)$, as the following computations show.

Finding $P'_n(x)$

$$P_n(x) = (x - z)Q_{n-1}(x) + b_0$$

$$\text{Thus, } P'_n(x) = Q_{n-1}(x) + (x - z)Q'_{n-1}(x)$$

$$\text{So, } P'_n(z) = Q_{n-1}(z).$$

This means that knowing the coefficients of $Q_{n-1}(x)$ we can compute $P'_n(z)$ recursively out of those coefficients. The coefficients of the polynomial $Q_{n-1}(z)$ can be obtained exactly in the same way as above, by replacing $a_k, k = n, \dots, 1$ by $b_k, k = n, n - 1, \dots, 1$ and labeling b 's by c 's, and a 's by b 's. And, then $Q_{n-1}(z) = P'_n(z) = c_1$. Note that in computing $P_n(z)$, we generated the numbers b_0, b_1, \dots, b_n , from the coefficients a_0, a_1, \dots, a_n of $P_n(x)$. So, in computing $P'_n(z)$, we need to generate the numbers c_1, c_2, \dots, c_{n-1} from the coefficients $b_1, b_2, \dots, b_{n-1}, b_n$ of the polynomial $Q_{n-1}(z)$, which have already been computed by Algorithm 1.5.

Then, we have the following scheme to compute $P'_n(z)$:

Computing $P'_n(z)$

Inputs: (1) b_1, \dots, b_n - The coefficients of the polynomial $Q_{n-1}(x)$ obtained from the previous algorithm
 (2) A number z .

Outputs: $c_1 = Q_{n-1}(z)$.

Step 1. Set $c_n = b_n$,

Step 2. For $k = n - 1, n - 2, \dots, 2, 1$ do

Compute $c_k = b_k + c_{k+1}z$,

End

Step 3. Set $P'_n(z) = c_1$.

Example 1.10 Let $P_3(x) = x^3 - 7x^2 + 6x + 5$.

Let $z = 2$.

Here $a_0 = 5, a_1 = 6, a_2 = -7, a_3 = 1$.

Generate the sequence $\{b_k\}$ from $\{a_k\}$:

$$b_3 = a_3 = 1$$

$$b_2 = a_2 + b_3z = -7 + 2 = -5$$

$$b_1 = a_1 + b_2z = 6 - 10 = -4$$

$$b_0 = a_0 + b_1z = 5 - 8 = -3.$$

$$\boxed{\text{So, } P_3(2) = b_0 = -3}$$

Generate the sequence $\{c_k\}$ from $\{b_k\}$.

$$c_3 = b_3 = 1$$

$$c_2 = b_2 + c_3z = -5 + 2 = -3$$

$$c_1 = b_1 + c_2z = -4 - 6 = -10.$$

$$\boxed{\text{So, } P'_3(2) = -10.}$$

The Newton-Raphson Method with Horner's Scheme

We now describe the Newton-Raphson Method for a polynomial using Horner's scheme. Recall that the Newton-Raphson iteration for finding a root of $f(x) = 0$ is:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

In case $f(x)$ is a polynomial $P_n(x)$ of degree n : $f(x) = P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, the above iteration becomes:

$$x_{k+1} = x_k - \frac{P_n(x_k)}{P'_n(x_k)}$$

If the sequence $\{b_k\}$ and $\{c_k\}$ are generated using Horner's Scheme, at each iteration we then have

$$\boxed{x_{k+1} = x_k - \frac{b_0}{c_1}}$$

(Note that at the iteration k , $P_n(x_k) = b_0$ and $P'_n(x_k) = c_1$).

Algorithm 1.6 *The Newton-Raphson Method with Horner's Scheme*

Input: a_0, a_1, \dots, a_n - The coefficients of $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$.
 x_0 - The initial approximation
 ϵ - The tolerance
 N - The maximum number of iterations to be performed.

Output: An approximation of the root $x = \xi$ or a failure message.

For $k = 0, 1, 2, \dots$, do

Step 1. $z = x_k, b_n \equiv a_n, c_n \equiv b_n$

Step 2.

Computing b_1 and c_1
For $j = n - 1, n - 2, \dots, 1$ do
$b_j \equiv a_j + z b_{j+1}$
$c_j \equiv b_j + z c_{j+1}$
End.

Step 3. $b_0 = a_0 + z b_1$

Step 4. $x_{k+1} = x_k - b_0/c_1$

Step 5. If $|x_{k+1} - x_k| < \epsilon$ or $k > N$, stop.

End

Notes:

1. The outer-loop corresponds to the Newton-Raphson Method
2. The inner-loop and the statement $b_0 = a_0 + z b_1$ corresponds to the evaluation of $P_n(x)$ and its derivative $P_n'(x)$ at successive approximations.

Example 1.11 Find a root of $P_3(x) = x^3 - 7x^2 + 6x + 5 = 0$, using the Newton-Raphson method and Horner's scheme.

$P_3(x)$ has a root between 1.5 and 2.

Let $x_0 = 2$

Iteration 1. $k = 0$:

$$b_0 = -3, c_1 = -10$$

$$x_1 = x_0 - \frac{b_0}{c_1} = 2 - \frac{3}{10} = \frac{17}{10} = 1.7$$

Iteration 2. $k = 1$:

$$z = x_1 = 1.7, b_3 = 1, c_3 = b_3 = 1.$$

$$b_2 = a_2 + z b_3 = -7 + 1.7 = -5.3$$

$$c_2 = b_2 + z c_3 = -5.3 + 1.7 = -3.6$$

$$b_1 = a_1 + z b_2 = 6 + 1.7(-5.3) = -3.0100$$

$$c_1 = b_1 + z c_2 = -3.01 + 1.7(-3.6) = -9.130$$

$$b_0 = a_0 + z b_1 = 5 + 1.7(-3.0100) = -0.1170$$

$$x_2 = x_1 - \frac{b_0}{c_1} = 1.7 - \frac{0.1170}{9.130} = 1.6872$$

(The exact root, correct up to four decimal places is 1.6872).

1.10 The Müller Method: Finding a Pair of Complex Roots.

The Müller Method is an extension of the Secant Method. It is conveniently used to approximate a **complex root** of a real equation.

Note that none of the methods: The Newton-Raphson, the Secant, the Fixed Point iteration Methods, etc. can produce an approximation of a complex root starting from real approximations.

The Müller Method is capable of doing so.

- Recall that starting with two initial approximations x_0 and x_1 , the Secant Method finds the next approximation x_2 as the x -intercept of the line passing through $(x_0, f(x_0))$, and $(x_1, f(x_1))$.
- In the Müller Method, starting with three initial approximations x_0 , x_1 and x_2 , the next approximation x_3 is obtained as the x -intercept of the parabola P passing through $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$.
- Since a parabola can intersect x -axis at two points, the one that is close to x_2 is taken as x_3 .
- So, in Müller's method, x_3 is obtained as follows:

1. Find the parabola P passing through $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$.

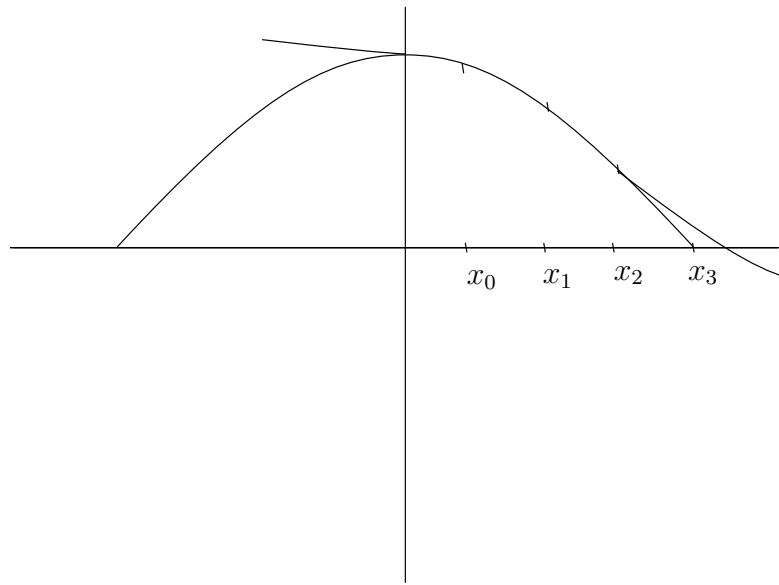
2. Solve the quadratic equation representing the parabola P and take x_3 as the root closest to x_2 .

(Note that even when x_0, x_1 , and x_2 are real, x_3 can be complex.)

The process, of course, can be continued to obtain other successive approximations. That is, x_4 is obtained by solving the quadratic equation representing the parabola P passing through $(x_1, f(x_1))$, $(x_2, f(x_2))$, and $(x_3, f(x_3))$, and taking x_4 as the zero that is closest to x_3 ; and so on.

Remarks: Note that it is advisable to relabel the approximation at the beginning of each iteration so that we need to store only 3 approximations per iteration.

Geometric Interpretation of the Müller Method



We now give the details of how to obtain x_3 starting from x_0 , x_1 , and x_2 .

Let the equation of the parabola P be $P(x) = a(x - x_2)^2 + b(x - x_2) + c$.

Since $P(x)$ passes through $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$, we have

$$P(x_0) = f(x_0) = a(x_0 - x_2)^2 + b(x_0 - x_2) + c$$

$$P(x_1) = f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c$$

$$P(x_2) = f(x_2) = c.$$

Knowing $c = f(x_2)$, we can now obtain a and b by solving the first two equations:

$$\begin{cases} a(x_0 - x_2)^2 + b(x_0 - x_2) = f(x_0) - c \\ a(x_1 - x_2)^2 + b(x_1 - x_2) = f(x_1) - c. \end{cases}$$

Or

$$\begin{pmatrix} (x_0 - x_2)^2 & (x_0 - x_2) \\ (x_1 - x_2)^2 & (x_1 - x_2) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f(x_0) - c \\ f(x_1) - c \end{pmatrix}.$$

Knowing a and b from above, we can now obtain x_3 by solving $P(x_3) = 0$.

$$P(x_3) = a(x_3 - x_2)^2 + b(x_3 - x_2) + c = 0.$$

To avoid **catastrophic cancellation**, instead of using the traditional formula, we use the following formula for solving the quadratic equation $ax^2 + bx + c = 0$.

<p>Solving the Quadratic Equation</p>
--

$ax^2 + bx + c = 0$ $x = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$
--

The sign in the denominator is chosen so that it is the largest in magnitude, guaranteeing that x_3 is closest to x_2 .

In our case, $x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$

or $x_3 = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$.

Note: one can also give on explicit expressions for a and b as follows:

$$b = \frac{(x_0 - x_2)^2 [f(x_1) - f(x_2)] - (x_1 - x_2)^2 [f(x_0) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}$$

$$a = \frac{(x_1 - x_2) [f(x_0) - f(x_2)] - (x_0 - x_2) [f(x_1) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}$$

However, for computational accuracy, it may be easier to solve the following 2×2 linear system to obtain a and b , once $c = f(x_2)$ is computed.

Algorithm 1.7 The Müller Method

Inputs: (i) $f(x)$ - The given function
(ii) x_0, x_1, x_2 - Initial approximations
(iii) ϵ - Tolerance
(iv) N - The maximum number of iterations.

Output: An approximation of the root $x = \xi$

Step 1: Evaluate $f(x_0), f(x_1),$ and $f(x_2)$.

Step 2: Compute $a, b,$ and $c,$ as follows:

2.1 $c = f(x_2)$.

2.2 Solve the 2×2 linear system for a and b :

$$\begin{pmatrix} (x_0 - x_2)^2 & (x_0 - x_2) \\ (x_1 - x_2)^2 & (x_1 - x_2) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f(x_0) - c \\ f(x_1) - c \end{pmatrix}.$$

Step 3: Compute $x_3 = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$ choosing $+$ or $-$ so that the denominator is the largest in magnitude.

Step 4: Test if $|x_3 - x_2| < \epsilon$ or if the number of iterations exceeds N . If so, stop. Otherwise go to Step 5.

Step 5: Relabel x_3 as $x_2,$ x_2 as $x_1,$ and x_1 as $x_0,$ and return to Step 1.

Example 1.12 $f(x) = x^3 - 2x^2 - 5$

The roots of $f(x)$ are $\{2.6906, -0.3453 \pm 1.3187i\}$

We will try to approximate the pair of complex roots here.

Iteration 1: Initial Approximations: $x_0 = -1, x_1 = 0, x_2 = 1$

Step 1: $f(x_0) = -8, f(x_1) = -5, f(x_2) = -6$.

Step 2: Compute $a, b,$ and c :

2.1 $c = f(x_2) = -6$

2.2 a and b are computed by solving the 2×2 linear system:

$$\left. \begin{array}{l} 4a - 2b = -2 \\ a - b = 1 \end{array} \right\}$$

This gives $a = -2, b = -3$.

Step 3: Compute $x_3 = x_2 - \frac{2c}{b - \sqrt{b^2 - 4ac}} = x_2 + \frac{12}{-3 - \sqrt{9 - 48}} = 0.25 + 1.5612i$

Iteration 2. Relabeling x_3 as x_2 , x_2 as x_1 , and x_1 as x_0 , we have

$$x_0 = 0, x_1 = 1, x_2 = 0.25 + 1.5612i$$

Step 1: $f(x_0) = -5$, $f(x_1) = -6$, and $f(x_2) = -2.0625 - 5.0741i$

Step 2: $c = f(x_2) = -2.0625 - 5.0741i$

a and b are obtained by solving the 2×2 linear system:

$$\begin{pmatrix} x_2^2 & -x_2 \\ (x_1 - x_2)^2 & (x_1 - x_2) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f(x_0) - c \\ f(x_1) - c \end{pmatrix},$$

which gives $a = -0.75 + 1.5612i$, $b = -5.4997 - 3.1224i$.

Step 3: $x_3 = x_2 - \frac{2c}{b - \sqrt{b^2 - 4ac}} = x_2 - (0.8388 + 0.3702i) = -0.5888 + 1.1910i$

Iteration 3: Again, relabeling x_3 as x_2 , x_2 as x_1 , and x_1 as x_0 , we have

$$x_0 = 1, x_1 = 0.25 + 1.5612i, x_2 = -0.5888 + 1.1910i$$

Step 1: $f(x_0) = -6$, $f(x_1) = -2.0627 - 5.073i$, $f(x_2) = -0.5549 + 2.3542i$

Step 2: $c = f(x_2) = -0.5549 + 2.3542i$

a and b are obtained by solving the system:

$$\begin{pmatrix} (x_0 - x_2)^2 & (x_0 - x_2) \\ (x_1 - x_2)^2 & (x_1 - x_2) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f(x_0) - c \\ f(x_1) - c \end{pmatrix},$$

which gives $a = -1.3888 + 2.7522i$, $b = -2.6339 - 8.5606i$.

Step 3: $x_3 = x_2 - \frac{2c}{b - \sqrt{b^2 - 4ac}} = -0.3664 + 1.3508i$

Iteration 4

$$x_3 = -0.3451 + 1.3180i$$

Iteration 5

$$x_3 = -0.3453 + 1.3187i$$

Iteration 6

$$x_3 = 0.3453 + 1.3187i$$

Since $\left| \frac{x_3 - x_2}{x_2} \right| < 5.1393 \times 10^{-4}$, we stop here.

Thus, an approximate pair of complex roots is: $\alpha_1 = -0.3455 + 1.3187i$, $\alpha_2 = -0.3455 - 1.3187i$.

Verify:

$$(x - \alpha_1)(x - \alpha_2)(x - 2.6906) = x^3 - 1.996x^2 - 5$$

Example 1.13 $f(x) = x^3 - 7x^2 + 6x + 5$

All three roots are real: 5.8210, 1.6872, -0.5090. We will try to approximate the root 1.6872.

Iteration 1.

$$x_0 = 0, x_1 = 1, x_2 = 2$$

Step 1: $f(x_0) = 5$, $f(x_1) = 5$, $f(x_2) = -3$.

Step 2: Compute $c = f(x_2) = -3$.

Then a and b are obtained by solving the 2×2 linear system:

$$\begin{pmatrix} (x_0 - x_2)^2 & (x_0 - x_2) \\ (x_1 - x_2)^2 & (x_1 - x_2) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 - c \\ 5 - c \end{pmatrix}$$

Or

$$\begin{pmatrix} 4 & -2 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \end{pmatrix}, \text{ which gives } a = -4, b = -12.$$

Step 3: Compute $x_3 = x_2 - \frac{2c}{b - \sqrt{b^2 - 4ac}}$
 $= x_2 - 0.2753 = 1.7247$ (choosing the negative sign in the determinant).

Iteration 2.

$$x_2 = x_3 = 1.7247, x_1 = x_2 = 2, x_0 = x_1 = 1$$

Step 1: $f(x_0) = 5$, $f(x_1) = -3$, $f(x_2) = -0.3441$

Step 2: Compute $c = -0.3441$

Then a and b are obtained by solving the 2×2 linear system:

$$\begin{pmatrix} 0.5253 & -0.7247 \\ 0.0758 & 0.2753 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 - c \\ 3 - c \end{pmatrix},$$

which gives $a = -2.2753$, $b = -9.0227$.

Step 3: Compute $x_3 = x_2 - \frac{2c}{b - \sqrt{b^2 - 4ac}} = x_2 - 0.0385 = 1.6862$

1.11 Sensitivity of the Roots of a Polynomial

The roots of a polynomial can be very sensitive to small perturbations of the coefficients of the polynomial. For example, take

$$p(x) = x^3 - 2x^2 + 1.$$

The roots $x_1 = -1$, $x_2 = -1$. Now perturb only the coefficient -2 to -1.9999 , leaving the other two coefficients unchanged. The roots of the perturbed polynomial are: $1 \pm .01i$. Thus, **both the roots become complex**. One might think that this happened because of the roots of $p(x)$ are multiple roots. It is true that the multiple roots are prone to perturbations, but **the roots of a polynomial with well-separated roots can be very sensitive too!!**

The well-known example of this is the celebrated **Wilkinson polynomial**:

$$p(x) = (x - 1)(x - 2)\dots(x - 20).$$

The roots of this polynomial equation are 1, 2, ..., 20 and thus very well-separated. But, a small perturbation of the coefficient x^{19} which is -210 will change the roots completely. Some of them will even become complex.

Try this yourself using MATLAB function **roots**.

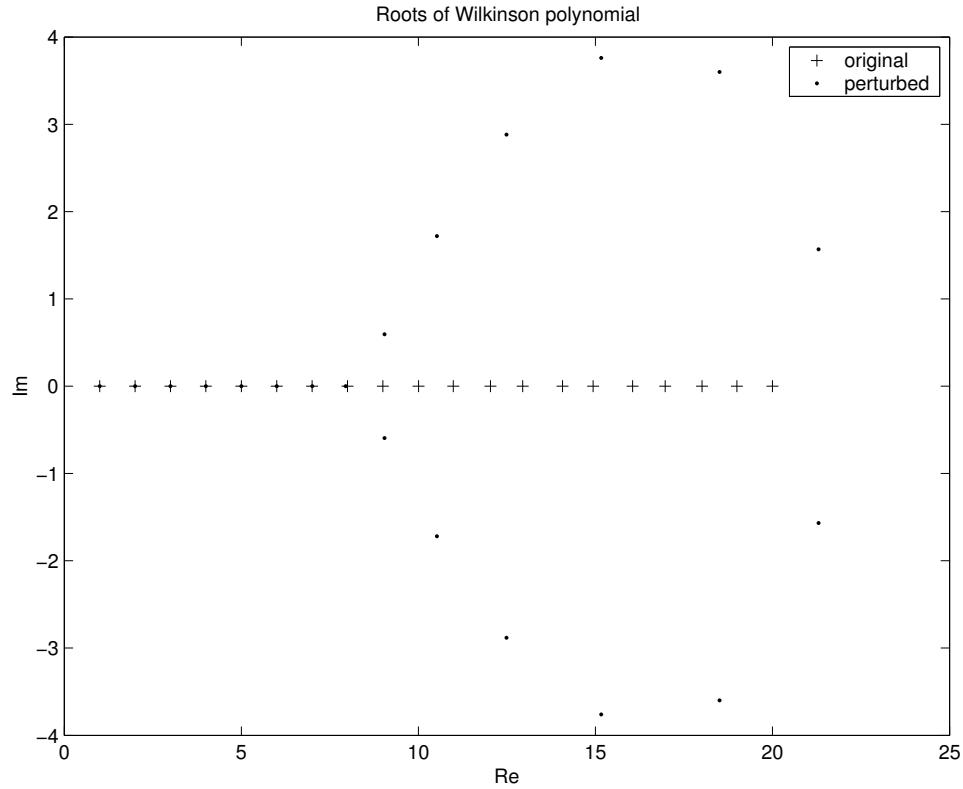


Figure 1.6: Roots of the Wilkinson Polynomial

1.12 Deflation Technique

Deflation is a technique to compute the other roots of $f(x) = 0$, once an approximate real root or a pair of complex roots are computed. Then if $x = \alpha$ is an approximate root of $P(x) = 0$, where $P(x)$ is a polynomial of degree n , then we can write

$$P(x) = (x - \alpha)Q_{n-1}(x)$$

where $Q_{n-1}(x)$ is a polynomial of degree $n - 1$.

Similarly, if $x = \alpha \pm i\beta$ is an approximate pair of complex conjugate roots, then

$$P(x) = (x - \alpha - i\beta)(x - \alpha + i\beta)Q_{n-2}(x)$$

where $Q_{n-2}(x)$ is a polynomial of degree $n - 2$. **Moreover, the zeros of $Q_{n-1}(x)$ in the first case and those of $Q_{n-2}(x)$ in the second case are also the zeros of $P(x)$.** The coefficients of $Q_{n-1}(x)$ or $Q_{n-2}(x)$ can be generated using **synthetic division** as in Horner's method, and then any of the root-finding procedure obtained before can be applied to compute a root or a pair of roots of $Q_{n-2}(x) = 0$ or $Q_{n-1}(x)$. The procedure can be continued until all the roots are found.

Example 1.14 Consider finding the roots of $f(x) = x^4 + 2x^3 + 3x^2 + 2x + 2 = 0$. It is easy to see that $\pm i$ is a pair of complex conjugate roots of $f(x) = 0$. Then we can write

$$f(x) = (x + i)(x - i)(b_3x^2 + b_2x + b_1)$$

or

$$x^4 + 2x^3 + 3x^2 + 2x + 2 = (x^2 + 1)(b_3x^2 + b_2x + b_1).$$

Equating coefficients of x^4 , x^3 , and x^2 on both sides, we obtain

$$b_3 = 1, \quad b_2 = 2, \quad b_1 + b_3 = 3,$$

giving $b_3 = 1, b_2 = 2, b_1 = 2$.

Thus, $(x^4 + 2x^3 + 3x^2 + 2x + 2) = (x^2 + 1)(x^2 + 2x + 2)$.

The zeros of the deflated polynomial $Q_2(x) = x^2 + 2x + 2$ can now be found by using any of the methods described earlier.

EXERCISES

1. Suppose that the bisection method is used to find a zero of $f(x) = x^3 - 7x + 6$ in $[0.5, 2]$ with an accuracy of 10^{-5} . How many minimum numbers of iterations will be needed?
2. (a) Prove Corollary 1.1 of the Lecture Notes; that is, prove that if $g(x)$ is continuously differentiable in some open interval containing the fixed points ξ , and if $|g'(\xi)| < 1$, then there exists a number $\epsilon > 0$ so that the iteration

$$x_{k+1} = g(x_k)$$

converges whenever x_0 is chosen in $|x_0 - \xi| \leq \epsilon$.

- (b) Using the above Corollary, find the iteration function $g(x)$ and an interval $[a, b]$ such that

$$f(x) = x - \tan x$$

always converges. Find the solution with an accuracy of 10^{-4} .

- (c) Find a zero of the function $f(x) = e^{-x} - \sin x$ in $[0.5, 07]$, by choosing $g(x) = x + f(x)$ and using Corollary 1.1.
3. Derive an algorithm to compute $\sqrt[n]{A}$ based on Newton's method and apply your algorithm to compute $\sqrt[5]{30}$, by choosing a convenient initial approximation.
4. Repeat the problem No 2 with the secant method and compare the speed of convergence.
5. (a) Sketch a graph of $f(x) = x + \ln x$ and show that $f(x)$ has one and only one zero ξ in $0 < x < \infty$, but that the related fixed point iteration $x_{i+1} = g(x_i)$, x_0 is given, $x_0 \neq \xi$, with $g(x) = -\ln x$, does not converge to ξ even if x_0 is arbitrary close to ξ .

(b) How can one rewrite the iteration equation so as to obtain convergence? Show for this iteration the uniqueness of the fixed point ξ . Give its rate of convergence.
6. The iteration $x_{n+1} = 2 - (1 + c)x_n + cx_n^3$ will converge to $\xi = 1$ for some values of c (provided x_0 is chosen sufficiently close to ξ). Find the values of c for which this is true. For what value of c will the convergence be quadratic?
7. Use fixed point iteration to find a zero of $f(x) = x^3 - 7x + 6$ on $[0, 1.5]$.
8. The function $f(x) = x^3 - 2x^2 + x - 2$ has a pair of zeros $x = \pm i$. Approximate them using Muller's method.