Solution of Algebraic and Transcendental Equations
Lab Manual 01

Prepared By:
Md. Azmain Yakin Srizon
Assistant Professor
Department of Computer Sciecne & Engineering
Rajshahi University of Engineering & Technology

Reference Book:
Introductory Methods of Numerical Analysis by S. S. Sastry (5th Edition)
Last Updated: September 9, 2024

1 Bisection Method

1.1 Procedure for Bisection Method

1. **Initial Setup**:
   - Choose two initial guesses, a and b, such that $f(a) \times f(b) < 0$, meaning the function $f(x)$ changes signs between a and b. This ensures that there is at least one root of $f(x) = 0$ in the interval $[a, b]$.
   - Set the desired tolerance level ϵ, which defines how accurate the root approximation should be.
2. **Iteration Process**:
   - Compute the midpoint of the interval:
   $$m = \frac{a + b}{2}$$
   - Evaluate the function at the midpoint $f(m)$.
   - **Check for Root**:
     - If $f(m) = 0$, then m is the exact root, and the procedure can terminate.
     - Otherwise, determine which subinterval contains the root:
       * If $f(a) \times f(m) < 0$, then the root lies in the interval $[a, m]$. Set $b = m$.
       * If $f(m) \times f(b) < 0$, then the root lies in the interval $[m, b]$. Set $a = m$.
3. **Convergence Criteria**:
   - Check if the width of the interval $|b - a|$ is less than the tolerance ϵ. If so, stop the iteration and accept m as the approximate root.
   - If not, repeat the iteration process with the updated interval.
4. **Repeat**:
   - Continue this process until the desired accuracy is achieved or the maximum number of iterations is reached.

1.2 Pseudo Code for Bisection Method (Example 2.1)

1. **Define Function:**
$$f(x) = x^3 - x - 1$$

2. **Input:**
   - Interval endpoints $a = 1$, $b = 2$
   - Tolerance $\epsilon = 0.0001$
   - Maximum iterations (optional)

3. **Check Validity:**
   - If $f(a) \times f(b) \geq 0$, then output "Invalid initial guesses" and stop.

4. **Iteration:**
   - Set iteration = 0
   - While $\frac{(b-a)}{2} > \epsilon$, do:
     (a) Increment iteration count.
     (b) Compute midpoint $m = \frac{a+b}{2}$.
     (c) Print the current iteration, a, b, m, and $f(m)$.
     (d) If $f(m) = 0$, then:
       – Output "Exact root found at m" and stop.
     (e) If $f(a) \times f(m) < 0$, then:
       – Set $b = m$ (root lies in $[a, m]$).
     (f) Else:
       – Set $a = m$ (root lies in $[m, b]$).

5. **Output:**
   - After loop ends, compute final midpoint $m = \frac{a+b}{2}$.
   - Output "The approximate root is m".

1.2.1 Sample Input and Output

Sample Output:

```
Iteration       a               b               m               f(m)
---------------------------------------------------------------------------
1               1.000000        2.000000        1.500000        0.875000
2               1.000000        1.500000        1.250000        -0.296875
3               1.250000        1.500000        1.375000        0.224609
4               1.250000        1.375000        1.312500        -0.051514
5               1.312500        1.375000        1.343750        0.082611
6               1.312500        1.343750        1.328125        0.014576
7               1.312500        1.328125        1.320313        -0.018711
8               1.320313        1.328125        1.324219        -0.002128
9               1.324219        1.328125        1.326172        0.006209
10              1.324219        1.326172        1.325195        0.002037
11              1.324219        1.325195        1.324707        -0.000047
12              1.324707        1.325195        1.324951        0.000995
13              1.324707        1.324951        1.324829        0.000474
---------------------------------------------------------------------------
The approximate root is: 1.324768
```

1.3 Pseudo Code for Bisection Method (Example 2.2)

1. **Define Function:**
$$\mathrm{f}(x) = x^3 - 2x - 5$$
2. **Input:**
   - Interval endpoints $a = 2$, $b = 3$
   - Tolerance $\epsilon = 0.0001$
   - Maximum iterations (optional)
3. **Check Validity:**
   - If $\mathrm{f}(a) \times \mathrm{f}(b) \geq 0$, then output "Invalid initial guesses" and stop.
4. **Iteration:**
   - Set iteration $= 0$
   - While $\frac{(b-a)}{2} > \epsilon$, do:
     (a) Increment iteration count.
     (b) Compute midpoint $m = \frac{a+b}{2}$.
     (c) Print the current iteration, a, b, m, and $\mathrm{f}(m)$.
     (d) If $\mathrm{f}(m) = 0$, then:
       – Output "Exact root found at m" and stop.
     (e) If $\mathrm{f}(a) \times \mathrm{f}(m) < 0$, then:
       – Set $b = m$ (root lies in $[a, m]$).
     (f) Else:
       – Set $a = m$ (root lies in $[m, b]$).
5. **Output:**
   - After loop ends, compute final midpoint $m = \frac{a+b}{2}$.
   - Output "The approximate root is m".

1.3.1 Sample Input and Output

Sample Output:

```
Iteration       a               b               m               f(m)
---------------------------------------------------------------------------
1               1.000000        2.000000        1.500000        0.875000
2               1.000000        1.500000        1.250000        -0.296875
3               1.250000        1.500000        1.375000        0.224609
4               1.250000        1.375000        1.312500        -0.051514
5               1.312500        1.375000        1.343750        0.082611
6               1.312500        1.343750        1.328125        0.014576
7               1.312500        1.328125        1.320313        -0.018711
8               1.320313        1.328125        1.324219        -0.002128
9               1.324219        1.328125        1.326172        0.006209
10              1.324219        1.326172        1.325195        0.002037
11              1.324219        1.325195        1.324707        -0.000047
12              1.324707        1.325195        1.324951        0.000995
13              1.324707        1.324951        1.324829        0.000474
---------------------------------------------------------------------------
The approximate root is: 1.324768
```

1.4 Pseudo Code for Bisection Method (Example 2.3)

1. **Define Function:**

$$\mathrm{f}(x) = x \cdot e^x - 1$$

2. **Input:**
   - Interval endpoints $a = 0$, $b = 1$
   - Tolerance $\epsilon = 0.0001$
   - Maximum iterations (optional)

3. **Check Validity:**
   - If $\mathrm{f}(a) \times \mathrm{f}(b) \geq 0$, then output "Invalid initial guesses" and stop.

4. **Iteration:**
   - Set iteration $= 0$
   - While $\frac{(b-a)}{2} > \epsilon$, do:
     (a) Increment iteration count.
     (b) Compute midpoint $m = \frac{a+b}{2}$.
     (c) Print the current iteration, a, b, m, and $\mathrm{f}(m)$.
     (d) If $\mathrm{f}(m) = 0$, then:
        – Output "Exact root found at m" and stop.
     (e) If $\mathrm{f}(a) \times \mathrm{f}(m) < 0$, then:
        – Set $b = m$ (root lies in $[a, m]$).
     (f) Else:
        – Set $a = m$ (root lies in $[m, b]$).

5. **Output:**
   - After loop ends, compute final midpoint $m = \frac{a+b}{2}$.
   - Output "The approximate root is m".

1.4.1 Sample Input and Output

Sample Output:

```
Iteration        a              b              m              f(m)
--------------------------------------------------------------------------
1                0.000000       1.000000       0.500000       -0.175639
2                0.500000       1.000000       0.750000       0.587750
3                0.500000       0.750000       0.625000       0.167654
4                0.500000       0.625000       0.562500       -0.012782
5                0.562500       0.625000       0.593750       0.075142
6                0.562500       0.593750       0.578125       0.030619
7                0.562500       0.578125       0.570313       0.008780
8                0.562500       0.570313       0.566406       -0.002035
9                0.566406       0.570313       0.568359       0.003364
10               0.566406       0.568359       0.567383       0.000662
11               0.566406       0.567383       0.566895       -0.000687
12               0.566895       0.567383       0.567139       -0.000013
13               0.567139       0.567383       0.567261       0.000325
--------------------------------------------------------------------------
The approximate root is: 0.567200
```

2 The Method of False Position

2.1 Procedure for Method of False Position

1. **Initial Setup:**
   - Choose two initial guesses a and b such that $f(a) \times f(b) < 0$, meaning the function $f(x)$ changes signs between a and b.
   - Set the desired tolerance level ϵ, which defines how accurate the root approximation should be.
2. **Iteration Process:**
   - Compute the point c where the straight line joining $(a, f(a))$ and $(b, f(b))$ crosses the x-axis. The formula for c is:
   $$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$
   - Evaluate $f(c)$.
   - **Check for Root:**
     - If $f(c) = 0$, then c is the exact root, and the procedure can terminate.
     - Otherwise, update the interval:
       * If $f(a) \times f(c) < 0$, set $b = c$ (root lies in $[a, c]$).
       * If $f(b) \times f(c) < 0$, set $a = c$ (root lies in $[c, b]$).
3. **Convergence Criteria:**
   - Check if the absolute difference $|b - a|$ or $|f(c)|$ is less than the tolerance ϵ. If so, stop the iteration and accept c as the approximate root.
4. **Repeat:**
   - Repeat the process until the desired accuracy is achieved or the maximum number of iterations is reached.

2.2 Pseudo Code for Method of False Position (Example 2.4)

1. **Define Function:**
   $$f(x) = x^3 - 2x - 5$$
2. **Input:**
   - Interval endpoints $a = 2$, $b = 3$
   - Tolerance $\epsilon = 0.0001$
   - Maximum number of iterations (optional)
3. **Check Validity:**
   - If $f(a) \times f(b) \geq 0$, output "Invalid initial guesses" and stop.
4. **Iteration:**
   - While $|b - a| \geq \epsilon$, do:
     (a) Compute the point c using the False Position formula:
     $$c = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$$
     (b) Print the current values of a, b, c, and $f(c)$.
     (c) If $|f(c)| < \epsilon$, output "Root found at c" and stop.
     (d) If $f(a) \times f(c) < 0$, set $b = c$ (root is in $[a, c]$).
     (e) Else, set $a = c$ (root is in $[c, b]$).
5. **Output:**
   - Output the approximate root c after the loop ends.

2.2.1 Sample Input and Output

Sample Output:

```
Iteration        a               b               c               f(c)
---------------------------------------------------------------------------
1                2.000000        3.000000        2.058824        -0.390800
2                2.058824        3.000000        2.081264        -0.147204
3                2.081264        3.000000        2.089639        -0.054677
4                2.089639        3.000000        2.092740        -0.020203
5                2.092740        3.000000        2.093884        -0.007451
6                2.093884        3.000000        2.094305        -0.002746
7                2.094305        3.000000        2.094461        -0.001012
8                2.094461        3.000000        2.094518        -0.000373
9                2.094518        3.000000        2.094539        -0.000137
10               2.094539        3.000000        2.094547        -0.000051
---------------------------------------------------------------------------
The approximate root is: 2.094547
```

2.3 Pseudo Code for Method of False Position (Example 2.5)

1. **Define Function:**
$$f(x) = x^{2.2} - 69$$
2. **Input:**
   - Interval endpoints $a = 5$, $b = 8$
   - Tolerance $\epsilon = 0.0001$
   - Maximum number of iterations (optional)
3. **Check Validity:**
   - If $f(a) \times f(b) \geq 0$, output "Invalid initial guesses" and stop.
4. **Iteration:**
   - While $|b - a| \geq \epsilon$, do:
     (a) Compute the point c using the False Position formula:
     $$c = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$$
     (b) Print the current values of a, b, c, and $f(c)$.
     (c) If $|f(c)| < \epsilon$, output "Root found at c" and stop.
     (d) If $f(a) \times f(c) < 0$, set $b = c$ (root is in $[a, c]$).
     (e) Else, set $a = c$ (root is in $[c, b]$).
5. **Output:**
   - Output the approximate root c after the loop ends.

2.3.1 Sample Input and Output

Sample Output:

```
Iteration        a              b              c              f(c)
-----------------------------------------------------------------------
1                5.000000       8.000000       6.655990       -4.275625
2                6.655990       8.000000       6.834002       -0.406148
3                6.834002       8.000000       6.850670       -0.037554
4                6.850670       8.000000       6.852209       -0.003464
5                6.852209       8.000000       6.852351       -0.000319
6                6.852351       8.000000       6.852364       -0.000029
-----------------------------------------------------------------------
The approximate root is: 6.852364
```

3 The Iteration Method

3.1 Procedure for Iteration Method (Fixed-Point Iteration)

1. **Initial Setup:**
   - Rewrite the equation $f(x) = 0$ in the form $x = g(x)$, where $g(x)$ is a continuous function.
   - Choose an initial guess x_0 for the solution.
   - Set the desired tolerance level ϵ, which defines how accurate the root approximation should be.
2. **Iteration Process:**
   - Compute the next approximation $x_{n+1} = g(x_n)$.
   - Check the difference between successive approximations $|x_{n+1} - x_n|$.
3. **Convergence Criteria:**
   - If $|x_{n+1} - x_n| < \epsilon$, then stop the iteration and accept x_{n+1} as the approximate root.
4. **Repeat:**
   - Continue the iteration process until the convergence criteria are met, or the maximum number of iterations is reached.
5. **Output:**
   - Output the approximate root x_{n+1} as the solution of the equation $f(x) = 0$.

3.2 Pseudo Code for Iterative Method (Example 2.6)

1. **Define Function:**
$$g(x) = \frac{1}{\sqrt{x+1}}$$
2. **Input:**
   - Initial guess x_0
   - Tolerance $\epsilon = 10^{-4}$
   - Maximum number of iterations (optional)
3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation:
     $$x_{n+1} = g(x_n) = \frac{1}{\sqrt{x_n + 1}}$$
     (b) Print the current values of x_n, x_{n+1}, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.
4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

3.2.1 Sample Input and Output

Sample Output:

```
Iteration        x_n            g(x_n)          |x_{n+1} - x_n|
----------------------------------------------------------------
1                0.500000       0.816497        0.316497
2                0.816497       0.741964        0.074533
3                0.741964       0.757671        0.015707
4                0.757671       0.754278        0.003393
5                0.754278       0.755007        0.000729
6                0.755007       0.754850        0.000157
7                0.754850       0.754884        0.000034
----------------------------------------------------------------
The approximate root is: 0.754884
```

3.3 Pseudo Code for Iteration Method (Example 2.7)

1. **Define the iterative function:**
$$g(x) = \frac{\cos(x) + 3}{2}$$

2. **Input:**
   - Initial guess x_0
   - Tolerance $\epsilon = 0.001$ (correct to 3 decimal places)
   - Maximum number of iterations (optional, for example, 100)

3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation:
     $$x_{n+1} = g(x_n) = \frac{\cos(x_n) + 3}{2}$$
     (b) Print the current values of x_n, x_{n+1}, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.

4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

3.3.1 Sample Input and Output

Sample Output:

```
Iteration        x_n            g(x_n)          |x_{n+1} - x_n|
-----------------------------------------------------------------
1                1.000000       1.770151        0.770151
2                1.770151       1.400982        0.369170
3                1.400982       1.584500        0.183518
4                1.584500       1.493148        0.091351
5                1.493148       1.538785        0.045637
6                1.538785       1.516003        0.022782
7                1.516003       1.527383        0.011380
8                1.527383       1.521700        0.005683
9                1.521700       1.524538        0.002839
10               1.524538       1.523121        0.001418
11               1.523121       1.523829        0.000708
-----------------------------------------------------------------
The approximate root is: 1.524
```

3.4 Pseudo Code for Iteration Method (Example 2.8)

1. **Define the iterative function:**

$$g(x) = \frac{1}{e^x}$$

2. **Input:**
   - Initial guess x_0
   - Tolerance $\epsilon = 0.0001$ (correct to 4 decimal places)
   - Maximum number of iterations (optional, for example, 100)

3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation:

$$x_{n+1} = g(x_n) = \frac{1}{e^{x_n}}$$

     (b) Print the current values of x_n, x_{n+1}, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.

4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

3.4.1 Sample Input and Output

Sample Output:

```
Iteration        x_n            g(x_n)          |x_{n+1} - x_n|
----------------------------------------------------------------
1                0.500000       0.606531        0.106531
2                0.606531       0.545239        0.061291
3                0.545239       0.579703        0.034464
4                0.579703       0.560065        0.019638
5                0.560065       0.571172        0.011108
6                0.571172       0.564863        0.006309
7                0.564863       0.568438        0.003575
8                0.568438       0.566409        0.002029
9                0.566409       0.567560        0.001150
10               0.567560       0.566907        0.000652
11               0.566907       0.567277        0.000370
12               0.567277       0.567067        0.000210
13               0.567067       0.567186        0.000119
14               0.567186       0.567119        0.000067
----------------------------------------------------------------
The approximate root is: 0.5671
```

4 Newton-Raphson Method

4.1 Procedure for Newton-Raphson Method

1. **Initial Setup:**
   - Given the equation $f(x) = 0$, choose an initial guess x_0 close to the expected root.
   - Define the derivative of the function $f'(x)$, which is required for the Newton-Raphson iteration.
   - Set the desired tolerance ϵ, which defines how accurate the root approximation should be.
2. **Iteration Process:**
   - Compute the next approximation using the Newton-Raphson formula:
   $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
   - Check the difference between successive approximations $|x_{n+1} - x_n|$.
3. **Convergence Criteria:**
   - If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the approximate root.
4. **Repeat:**
   - Repeat the iteration process until the convergence criteria are met or the maximum number of iterations is reached.
5. **Output:**
   - Output the approximate root x_{n+1} as the solution to the equation $f(x) = 0$.

4.2 Pseudo Code for Newton-Raphson Method (Example 2.10)

1. **Define the functions:**
   $$f(x) = x^3 - 2x - 5$$
   $$f'(x) = 3x^2 - 2$$
2. **Input:**
   - Initial guess x_0
   - Tolerance $\epsilon = 0.0001$ (correct to 4 decimal places)
   - Maximum number of iterations (optional, e.g., 100)
3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation using the Newton-Raphson formula:
     $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
     (b) Print the current values of x_n, $f(x_n)$, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.
4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

4.2.1 Sample Input and Output

Sample Output:

```
Iteration        x_n             f(x_n)          |x_{n+1} - x_n|
-----------------------------------------------------------------
1                2.000000        -1.000000       0.100000
2                2.100000        0.061000        0.005432
3                2.094568        0.000186        0.000017
-----------------------------------------------------------------
The approximate root is: 2.0946
```

4.3 Pseudo Code for Newton-Raphson Method (Example 2.11)

1. **Define the functions:**

$$f(x) = x\sin(x) + \cos(x)$$
$$f'(x) = x\cos(x) - \sin(x)$$

2. **Input:**
   - Initial guess $x_0 = 3.1416$ (starting point close to π)
   - Tolerance $\epsilon = 0.0001$ (correct to 4 decimal places)
   - Maximum number of iterations (optional, e.g., 100)

3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation using the Newton-Raphson formula:

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

     (b) Print the current values of x_n, $f(x_n)$, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.

4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

4.3.1 Sample Input and Output

Sample Output:

```
Iteration        x_n             f(x_n)          |x_{n+1} - x_n|
-----------------------------------------------------------------
1                3.141600        -1.000023       0.318317
2                2.823283        -0.066186       0.022103
3                2.801180        -0.007369       0.002478
4                2.798702        -0.000833       0.000280
5                2.798422        -0.000094       0.000032
-----------------------------------------------------------------
The approximate root is: 2.7984
```

4.4 Pseudo Code for Newton-Raphson Method (Example 2.12)

1. **Define the functions:**
$$f(x) = x - e^{-x}$$
$$f'(x) = 1 + e^{-x}$$

2. **Input:**
   - Initial guess $x_0 = 1$
   - Tolerance $\epsilon = 0.0001$ (correct to 4 decimal places)
   - Maximum number of iterations (optional, e.g., 100)

3. **Iteration:**
   - Set $n = 0$ (initial iteration)
   - While $|x_{n+1} - x_n| \geq \epsilon$, do:
     (a) Compute the next approximation using the Newton-Raphson formula:
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
     (b) Print the current values of x_n, $f(x_n)$, and $|x_{n+1} - x_n|$.
     (c) If $|x_{n+1} - x_n| < \epsilon$, stop the iteration and accept x_{n+1} as the root.
     (d) Update $x_n = x_{n+1}$ and increment n.

4. **Output:**
   - Output the approximate root x_{n+1} after convergence.

4.4.1 Sample Input and Output

Sample Output:

```
Iteration       x_n             f(x_n)          |x_{n+1} - x_n|
----------------------------------------------------------------
1               1.000000        0.632121        0.462117
2               0.537883        -0.046100       0.029104
3               0.566987        -0.000245       0.000156
4               0.567143        -0.000000       0.000000
----------------------------------------------------------------
The approximate root is: 0.5671
```

5 Ramanujan's Method

5.1 Ramanujan's Method

Srinivasa Ramanujan (1887–1920) described an iterative method which can be used to determine the smallest root of the equation:

$$f(x) = 0$$

where $f(x)$ is of the form:

$$f(x) = 1 - (a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \dots)$$

For smaller values of x, we can write:

$$[1 - (a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \dots)]^{-1} = b_1 + b_2x + b_3x^2 + \dots$$

Expanding the left-hand side by the binomial theorem, we obtain:

$$1 + (a_1x + a_2x^2 + a_3x^3 + \dots) + (a_1x + a_2x^2 + a_3x^3 + \dots)^2 + \dots = b_1 + b_2x + b_3x^2 + \dots$$

Comparing the coefficients of like powers of x on both sides of this equation, we get:

$$b_1 = 1, \quad b_2 = a_1, \quad b_3 = a_1^2 + a_2 = a_1b_2 + a_2b_1$$

and for general n:

$$b_n = a_1b_{n-1} + a_2b_{n-2} + \dots + a_{n-1}b_1 \quad \text{for} \quad n = 2, 3, \dots$$

Without any proof, Ramanujan states that the successive convergents $\frac{b_n}{b_{n+1}}$ approach a root of the equation $f(x) = 0$, where $f(x)$ is given by the series expansion above.

5.2 Pseudo Code for Modified Ramanujan's Method (Example 2.14)

1. **Initialize Coefficients:**
   - Set the coefficients based on the given equation:

     $$a_1 = \frac{11}{6}, \quad a_2 = -1, \quad a_3 = \frac{1}{6}$$

   - Set higher order coefficients $a_4 = a_5 = a_6 = \dots = 0$.
2. **Initialize b_n Terms:**
   - Set $b_1 = 1$.
3. **Set Convergence Criteria:**
   - Set the desired tolerance $\epsilon = 0.001$.
4. **Iterative Calculation of b_n Terms:**
   - For $n = 2$ to the desired number of terms, perform the following steps:
     (a) If $n = 2$:
       - Compute b_2:

         $$b_2 = a_1$$

     (b) If $n = 3$:
       - Compute b_3:

         $$b_3 = a_1b_2 + a_2b_1$$

     (c) If $n \geq 4$:
       - Compute b_n using the recursive formula:

         $$b_n = a_1b_{n-1} + a_2b_{n-2} + a_3b_{n-3}$$

5. **Calculate Successive Ratios of b_n Terms:**

- For each $n \geq 2$:
  (a) Compute the ratio:
  $$R_n = \frac{b_{n-1}}{b_n}$$

6. **Check for Convergence:**
   - For each $n \geq 4$:
     (a) Compute the difference between successive ratios:
     $$\Delta R = |R_n - R_{n-1}|$$
     (b) If $\Delta R < \epsilon$, stop the iteration.
     (c) Accept R_n as the approximate smallest root.
7. **Output:**
   - Output the approximate smallest root:
   $$x \approx R_n$$

5.2.1 Sample Input and Output

Sample Output:

```
b_3 = 2.361111
b_2 / b_3 = 0.776471
b_4 = 2.662037
b_3 / b_4 = 0.886957
b_5 = 2.824846
b_4 / b_5 = 0.942365
b_6 = 2.910365
b_5 / b_6 = 0.970616
b_7 = 2.954497
b_6 / b_7 = 0.985063
b_8 = 2.977020
b_7 / b_8 = 0.992434
b_9 = 2.988434
b_8 / b_9 = 0.996181
b_10 = 2.994191
b_9 / b_10 = 0.998077
b_11 = 2.997087
b_10 / b_11 = 0.999034
b_12 = 2.998541
b_11 / b_12 = 0.999515
b_13 = 2.999269
b_12 / b_13 = 0.999757
b_14 = 2.999634
b_13 / b_14 = 0.999878
```

5.3 Pseudo Code for Ramanujan's Method (Example 2.15)

1. **Initialize Coefficients:**
   - Set the coefficients based on the given equation:
   $$a_1 = 1, \quad a_2 = 1, \quad a_3 = \frac{1}{2}, \quad a_4 = \frac{1}{6}, \quad a_5 = \frac{1}{24}$$
2. **Initialize b_n Terms:**
   - Set $b_1 = 1$.

3. **Compute b_n Terms:**
   - For $n = 2$ to $n = 6$, compute b_n as follows:
     - $b_2 = a_1$
     - $b_3 = a_1 b_2 + a_2 b_1$
     - $b_4 = a_1 b_3 + a_2 b_2 + a_3 b_1$
     - $b_5 = a_1 b_4 + a_2 b_3 + a_3 b_2 + a_4 b_1$
     - $b_6 = a_1 b_5 + a_2 b_4 + a_3 b_3 + a_4 b_2 + a_5 b_1$
4. **Calculate Successive Ratios:**
   - For each $n \geq 2$, compute the ratio:
   $$R_n = \frac{b_{n-1}}{b_n}$$
5. **Check for Convergence:**
   - If $|R_n - R_{n-1}| < \epsilon$ (where $\epsilon = 0.0001$), stop the iteration.
   - Accept R_n as the approximate root.
6. **Output:**
   - Output the approximate root R_n.

5.3.1 Sample Input and Output

Sample Output:

```
b_1 = 1.000000
b_2 = 1.000000
b_3 = 2.000000
b_4 = 3.500000
b_5 = 6.166667
b_6 = 10.875000

Ratios of consecutive b_n terms:
b_1 / b_2 = 1.000000
b_2 / b_3 = 0.500000
b_3 / b_4 = 0.571429
b_4 / b_5 = 0.567568
b_5 / b_6 = 0.567050
```

5.4 Pseudo Code for Ramanujan's Method (Example 2.16)

1. **Initialize Coefficients:**
   - Set the coefficients based on the given equation:
   $$a_1 = 2, \quad a_2 = 0, \quad a_3 = -\frac{1}{6}, \quad a_4 = 0, \quad a_5 = \frac{1}{120}, \quad a_6 = 0, \quad a_7 = -\frac{1}{5040}$$
2. **Initialize b_n Terms:**
   - Set $b_1 = 1$.
3. **Set Tolerance:**
   - Set the tolerance level for convergence: $\epsilon = 0.0001$.
4. **Compute b_n Terms:**
   - For $n = 2$ to $n = 8$, compute b_n using the recursive formula:
     - $b_2 = a_1$

- $b_3 = a_1 b_2 + a_2 b_1$
- $b_4 = a_1 b_3 + a_2 b_2 + a_3 b_1$
- $b_5 = a_1 b_4 + a_2 b_3 + a_3 b_2 + a_4 b_1$
- $b_6 = a_1 b_5 + a_2 b_4 + a_3 b_3 + a_4 b_2 + a_5 b_1$
- $b_7 = a_1 b_6 + a_2 b_5 + a_3 b_4 + a_4 b_3 + a_5 b_2 + a_6 b_1$
- $b_8 = a_1 b_7 + a_2 b_6 + a_3 b_5 + a_4 b_4 + a_5 b_3 + a_6 b_2 + a_7 b_1$

5. **Calculate Successive Ratios:**
   - For each $n \geq 2$, compute the ratio:
   $$R_n = \frac{b_{n-1}}{b_n}$$
   - Compute the ratios:
   $$R_2 = \frac{b_1}{b_2}, \quad R_3 = \frac{b_2}{b_3}, \quad \ldots, \quad R_8 = \frac{b_7}{b_8}$$
6. **Check for Convergence:**
   - If $|R_8 - R_7| < \epsilon$, stop the iteration and accept R_8 as the approximate root.
7. **Output:**
   - Output the approximate root R_8.

5.4.1 Sample Input and Output

Sample Output:

```
b_1 = 1.000000
b_2 = 2.000000
b_3 = 4.000000
b_4 = 7.833333
b_5 = 15.333333
b_6 = 30.008333
b_7 = 58.727778
b_8 = 114.933135

Ratios of consecutive b_n terms:
b_1 / b_2 = 0.500000
b_2 / b_3 = 0.500000
b_3 / b_4 = 0.510638
b_4 / b_5 = 0.510870
b_5 / b_6 = 0.510969
b_6 / b_7 = 0.510973
b_7 / b_8 = 0.510973

Convergence reached. Approximate root = 0.510973
```

5.5 Pseudo Code for Ramanujan's Method

1. **Initialize Coefficients:**
   - Set the coefficients based on the given equation:
   $$a_1 = 1, \quad a_2 = -\frac{1}{4}, \quad a_3 = \frac{1}{36}, \quad a_4 = -\frac{1}{576}, \quad a_5 = \frac{1}{14400}, \quad a_6 = -\frac{1}{518400}, \quad a_7 = \frac{1}{25401600}$$
2. **Initialize b_n Terms:**
   - Set $b_1 = 1$.
3. **Set Tolerance:**

- Set the tolerance level for convergence: $\epsilon = 0.001$.

4. **Compute b_n Terms:**
   - For $n = 2$ to $n = 8$, compute b_n using the recursive formula:
     - $b_2 = a_1$
     - $b_3 = a_1 b_2 + a_2 b_1$
     - $b_4 = a_1 b_3 + a_2 b_2 + a_3 b_1$
     - $b_5 = a_1 b_4 + a_2 b_3 + a_3 b_2 + a_4 b_1$
     - $b_6 = a_1 b_5 + a_2 b_4 + a_3 b_3 + a_4 b_2 + a_5 b_1$
     - $b_7 = a_1 b_6 + a_2 b_5 + a_3 b_4 + a_4 b_3 + a_5 b_2 + a_6 b_1$
     - $b_8 = a_1 b_7 + a_2 b_6 + a_3 b_5 + a_4 b_4 + a_5 b_3 + a_6 b_2 + a_7 b_1$

5. **Calculate Successive Ratios:**
   - For each $n \geq 2$, compute the ratio:
   $$R_n = \frac{b_{n-1}}{b_n}$$
   - Compute the ratios:
   $$R_2 = \frac{b_1}{b_2}, \quad R_3 = \frac{b_2}{b_3}, \quad \ldots, \quad R_8 = \frac{b_7}{b_8}$$

6. **Check for Convergence:**
   - If $|R_8 - R_7| < \epsilon$, stop the iteration and accept R_8 as the approximate root.

7. **Output:**
   - Output the approximate root R_8.

5.5.1 Sample Input and Output

Sample Output:

```
b_1 = 1.000000
b_2 = 1.000000
b_3 = 0.750000
b_4 = 0.527778
b_5 = 0.366319
b_6 = 0.253542
b_7 = 0.175388
b_8 = 0.121312

Ratios of consecutive b_n terms:
b_1 / b_2 = 1.000000
b_2 / b_3 = 1.333333
b_3 / b_4 = 1.421053
b_4 / b_5 = 1.440758
b_5 / b_6 = 1.444810
b_6 / b_7 = 1.445607
b_7 / b_8 = 1.445760

Convergence reached. Approximate root = 1.445760
```

6 The Secant Method

6.1 Procedure for the Secant Method

1. **Given Equation:**
   - Consider a nonlinear equation of the form $f(x) = 0$.
2. **Initial Guess:**
   - Choose two initial approximations x_0 and x_1 such that $f(x_0)$ and $f(x_1)$ are close to zero.
3. **Iterative Formula:**
   - Use the following secant method formula to find the next approximation x_{n+1}:

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

   - This formula is derived from the secant line passing through the points $(x_n, f(x_n))$ and $(x_{n-1}, f(x_{n-1}))$, which approximates the root of $f(x)$.
4. **Convergence Criteria:**
   - Check if the absolute difference $|x_{n+1} - x_n|$ is less than a predefined tolerance ϵ. If:

$$|x_{n+1} - x_n| < \epsilon$$

     then accept x_{n+1} as the approximate root and stop the iteration.
5. **Repeat:**
   - If the convergence criterion is not met, set $x_{n-1} = x_n$ and $x_n = x_{n+1}$ and repeat the process until convergence.
6. **Output:**
   - Output the approximate root x_{n+1} once the convergence criterion is satisfied.

6.2 Pseudo Code for Secant Method

1. **Input:**
   - The function $f(x) = x^3 - 2x - 5$.
   - Initial guesses x_0 and x_1.
   - Tolerance ϵ for convergence.
   - Maximum number of iterations.
2. **Initialize:**
   - Set $n = 0$ (iteration counter).
3. **Iterative Process:**
   - Repeat until the maximum number of iterations is reached or convergence is achieved:
     (a) Compute $f(x_0)$ and $f(x_1)$.
     (b) Check if $|f(x_1) - f(x_0)|$ is too small. If so, stop the iteration and print an error message to avoid division by zero.
     (c) Use the Secant Method formula to calculate the next approximation x_2:

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

     (d) Calculate the absolute difference $|x_2 - x_1|$.

(e) If $|x_2 - x_1| < \epsilon$, then stop the iteration and accept x_2 as the root.

(f) Update the values:

$$x_0 = x_1, \quad x_1 = x_2$$

(g) Increment the iteration counter n.

4. **Check for Maximum Iterations:**

   - If the maximum number of iterations is reached without convergence, print the last approximation x_2 as the final result.

5. **Output:**

   - Output the root x_2 once the convergence criteria $|x_2 - x_1| < \epsilon$ is satisfied.

6.2.1 Sample Input and Output

Sample Output:

```
Iteration        x_n            f(x_n)          |x_{n+1} - x_n|
----------------------------------------------------------------
1                3.000000       16.000000       0.941176
2                2.058824       -0.390800       0.022440
3                2.081264       -0.147204       0.013560
4                2.094824       0.003044        0.000275
5                2.094549       -0.000023       0.000002

Convergence reached. Root = 2.094551
```

THE END