# Problem 1.1: Bisection Method

**Learning Objectives:**

- Understand the algorithm of the Bisection method.
- Learn to code this algorithm.
- Use Bisection method for computing the roots of non linear equations.

**Bisection Method Algorithm:**

Let us consider a continuous function *f(x)* which is defined on the closed interval [*a, b*], is given with *f(a)* and *f(b)* of different signs. Then there exists a point *c* belong to (*a, b*) for which *f(c) = 0*. The iteration formula for approximating next root using the 'Bisection method' is

$$c=\frac{a+b}{2}=b-\frac{b-a}{2}$$

Follow the below procedure to get the root of the equation *f(x) = 0* :

1. Find two points, say *a* and *b* such that $f(a)f(b) < 0$
2. Find the midpoint of a and b, say *c,* i.e. $c = (a+b)/2$
3. *c* is the root of the given function if $f(c) = 0$; else follow the next step
4. Divide the interval [*a, b*] – If $f(a)f(c) <0$, there exist a root between *a* and *c* – else there exist a root between *c* and *b*
5. Repeat from step 2 until $f(c) = 0$.

**Sample Input/output:**

```
user@host:~

user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Iter     a         b         c        f(a)      f(b)      f(c)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1  1.250000  1.500000  1.375000 -1.796875  2.375000  0.162109
  2  1.250000  1.375000  1.312500 -1.796875  0.162109 -0.848389
  3  1.312500  1.375000  1.343750 -0.848389  0.162109 -0.350983
  4  1.343750  1.375000  1.359375 -0.350983  0.162109 -0.096409
  5  1.359375  1.375000  1.367188 -0.096409  0.162109  0.032356
  6  1.359375  1.367188  1.363281 -0.096409  0.032356 -0.032150
  7  1.363281  1.367188  1.365234 -0.032150  0.032356  0.000072
  8  1.363281  1.365234  1.364258 -0.032150  0.000072 -0.016047
  9  1.364258  1.365234  1.364746 -0.016047  0.000072 -0.007989
 10  1.364746  1.365234  1.364990 -0.007989  0.000072 -0.003959
 11  1.364990  1.365234  1.365112 -0.003959  0.000072 -0.001944
 12  1.365112  1.365234  1.365173 -0.001944  0.000072 -0.000936
 13  1.365173  1.365234  1.365204 -0.000936  0.000072 -0.000432
 14  1.365204  1.365234  1.365219 -0.000432  0.000072 -0.000180
 15  1.365219  1.365234  1.365227 -0.000180  0.000072 -0.000054
 16  1.365227  1.365234  1.365231 -0.000054  0.000072  0.000009
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Approximate root = 1.365231
```

**Tasks:**

1. Write a program to find a solution using the 'Bisection method' of the function $f(x) = x^3 + 4x^2 - 10$ with in the interval [*1.25, 1.5*] and a tolerance of $10^{-6}$. Show the steps, the program uses to achieve this tolerance.

# Problem 1.2: False Position Method

**Learning Objectives:**

- Understand the algorithm of the False Position method.
- Learn to code this algorithm.
- Use False Position method for computing the roots of non linear equations.

**False Position Method Algorithm:**

Let us consider a continuous function *f(x)* which is defined on the closed interval [*a, b*], is given with *f(a)* and *f(b)* of different signs. Then there exists a point *c* belong to (*a, b*) for which $f(c) = 0$. The iteration formula for approximating next root using the 'False Position method' is

$$c=\frac{af(b)-bf(a)}{f(b)-f(a)}=b-\frac{f(b)(b-a)}{f(b)-f(a)}$$

Follow the below procedure to get the root of the equation $f(x) = 0$ :

1. Find two points, say *a* and *b* such that $f(a)f(b) < 0$
2. Find a point *c* belong to (*a, b*), using $c = ( af(b) - bf(a) )/( f(b) - f(a) )$
3. *c* is the root of the given function if $f(c) = 0$; else follow the next step
4. Divide the interval [*a, b*] – If $f(a)f(c) <0$, there exist a root between *a* and *c* – else there exist a root between *c* and *b*
5. Repeat from step 2 until $f(c) = 0$.

**Sample Input/output:**

```
user@host:~
user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - - - - - - - -  - - - - - - - -
Iter     a         b         c       f(a)      f(b)      f(c)
- - - - - - - - - - - - - - - - - - - - - - - - - -  - - - - - - - -
   1  1.250000  1.500000  1.357678 -1.796875  2.375000 -0.124250
   2  1.357678  1.500000  1.364753 -0.124250  2.375000 -0.007868
   3  1.364753  1.500000  1.365200 -0.007868  2.375000 -0.000495
   4  1.365200  1.500000  1.365228 -0.000495  2.375000 -0.000031
   5  1.365228  1.500000  1.365230 -0.000031  2.375000 -0.000002
- - - - - - - - - - - - - - - - - - - - - - - - - -  - - - - - - - -
Approximate root = 1.365230
```

**Tasks:**

1. Write a program to find a solution using the 'False Position method' of the function $f(x) = x^3 + 4x^2 - 10$ with in the interval [*1.25, 1.5*] and a tolerance of $10^{-6}$ . Show the steps, the program uses to achieve this tolerance.

# Problem 2.1: Fixed Point Method

**Learning Objectives:**

- Understand the algorithm of the Fixed Point method.
- Learn to code this algorithm.
- Use Fixed Point method for computing the roots of non linear equations.

**Fixed Point Algorithm:**

Let us consider a continuous function $f(x)$. For finding a root of the equation $f(x) = 0$, we rewrite this equation in the form $x = g(x)$, where $g(x) = x + f(x)$. If there exists a point $c$ for which $c = g(c)$ then this $c$ will also satisfy the equation $f(c) = 0$. The iteration formula for approximating next root using the 'Fixed Point method' is

$$x_{n+1} = g(x_n) \quad , \quad \text{where} \quad g(x_n) = x_n + f(x_n)$$

Follow the below procedure to get the root of the equation $f(x) = 0$ :

1. Find a initial guess $x_0$
2. Calculate the next guess by $x_1 = g(x_0)$
3. $x_1$ is the root of the given function if $f(x_1) = 0$; else follow the next step
4. Update the initial guess by $x_0 \leftarrow x_1$
5. Repeat from step 2 until $f(x_1) = 0$.

**Sample Input/output:**

```
user@host:~

user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - - - - -
Iter    x0       x1        g(x0)     f(x1)
- - - - - - - - - - - - - - - - - - - - - - -
   1  1.500000  1.286954  1.286954 -1.243483
   2  1.286954  1.402541  1.402541  0.627450
   3  1.402541  1.345458  1.345458 -0.323340
   4  1.345458  1.375170  1.375170  0.164948
   5  1.375170  1.360094  1.360094 -0.084596
   6  1.360094  1.367847  1.367847  0.043270
   7  1.367847  1.363887  1.363887 -0.022163
   8  1.363887  1.365917  1.365917  0.011344
   9  1.365917  1.364878  1.364878 -0.005808
  10  1.364878  1.365410  1.365410  0.002973
  11  1.365410  1.365138  1.365138 -0.001522
  12  1.365138  1.365277  1.365277  0.000779
  13  1.365277  1.365206  1.365206 -0.000399
  14  1.365206  1.365242  1.365242  0.000204
  15  1.365242  1.365224  1.365224 -0.000105
  16  1.365224  1.365233  1.365233  0.000054
  17  1.365233  1.365228  1.365228 -0.000027
  18  1.365228  1.365231  1.365231  0.000014
  19  1.365231  1.365230  1.365230 -0.000007
- - - - - - - - - - - - - - - - - - - - - - -
Approximate root = 1.365230
```

**Tasks:**

1. Write a program to find a solution using the 'Fixed Point method' of the function $f(x) = x^3 + 4x^2 - 10$ with a initial guess $1.5$ and a tolerance of $10^{-6}$. Show the steps, the program uses to achieve this tolerance.

## Problem 2.2: Newton-Raphson Method

**Learning Objectives:**

- Understand the algorithm of the Newton-Raphson method.
- Learn to code this algorithm.
- Use Newton-Raphson method for computing the roots of non linear equations.

**Newton-Raphson Method Algorithm:**

Let us consider a continuous function *f(x)*. For finding a root of the equation $f(x) = 0$, we have to find a point *c* for which $f(c) = 0$. The iteration formula for approximating next root using the 'Newton-Raphson method' is

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

Follow the below procedure to get the root of the equation $f(x) = 0$ :

1. Find a initial guess $x_0$
2. Calculate the next guess by $x_1 = x_0 - f(x_0)/f'(x_0)$
3. $x_1$ is the root of the given function if $f(x_1) = 0$; else follow the next step
4. Update the initial guess by $x_0 \leftarrow x_1$
5. Repeat from step 2 until $f(x_1) = 0$.

**Sample Input/output:**

```
user@host:~
user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - - - - - - - - - -
Iter    x0       x1        f(x0)     f'(x0)    f(x1)
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1  1.500000  1.373333  2.375000 18.750000  0.134345
  2  1.373333  1.365262  0.134345 16.644800  0.000528
  3  1.365262  1.365230  0.000528 16.513917  0.000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
Approximate root = 1.365230
```

**Tasks:**

1. Write a program to find a solution using the 'Newton-Raphson method' of the function $f(x) = x^3 + 4x^2 - 10$ near *1.5* and a tolerance of $10^{-6}$ . Show the steps, the program uses to achieve this tolerance.

# Problem 2.3: Secant Method

**Learning Objectives:**

- Understand the algorithm of the Secant method.
- Learn to code this algorithm.
- Use Secant method for computing the roots of non linear equations.

**Secant Method Algorithm:**

Let us consider a continuous function $f(x)$. For finding a root of the equation $f(x) = 0,$ we have to find a point $c$ for which $f(c) = 0$. The iteration formula for approximating next root using the 'Secant method' is

$$x_{n+2} = \frac{x_n f(x_{n+1}) - x_{n+1} f(x_n)}{f(x_{n+1}) - f(x_n)} = x_{n+1} - \frac{f(x_{n+1})(x_{n+1} - x_n)}{f(x_{n+1}) - f(x_n)}$$

Follow the below procedure to get the root of the equation $f(x) = 0$ :

1. Find two initial guesses $x_0$ and $x_1$
2. Calculate the next guess by $x_2 = x_1 - f(x_1)(x_1 - x_0)/(f(x_1) - f(x_0))$
3. $x_2$ is the root of the given function if $f(x_2) = 0;$ else follow the next step
4. Update the initial guesses by $x_0 \leftarrow x_1$ *and* $x_1 \leftarrow x_2$
5. Repeat from step 2 until $f(x_2) = 0.$

**Sample Input/output:**

```
user@host:~

user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Iter    x0       x1        x2        f(x0)     f(x1)     f(x2)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1  1.500000  2.000000  1.397849  2.375000 14.000000  0.547307
  2  2.000000  1.397849  1.373352 14.000000  0.547307  0.134651
  3  1.397849  1.373352  1.365358  0.547307  0.134651  0.002113
  4  1.373352  1.365358  1.365231  0.134651  0.002113  0.000008
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Approximate root = 1.365231
```

**Tasks:**

1. Write a program to find a solution using the 'Secant method' of the function $f(x) = x^3 + 4x^2 - 10$ with the initial guesses $\{1.5, 2.0\}$ and a tolerance of $10^{-6}$ . Show the steps, the program uses to achieve this tolerance.

# Problem 3.1: Forward-difference quotient

**Learning Objectives:**

- Understand the Forward-difference quotient.
- Use Forward-difference quotient for computing the Numerical Differentiation of a function.

**Forward-difference quotient:**

Let us consider a function $f(x)$, where $x \in [a, b]$ and $f \in C^2[a, b]$. Then to approximate $f'(x)$ using the 'Forward-difference quotient', we can use the following formula

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f(x_{i+1}) - f(x_i)}{h} \quad \text{or} \quad f'(x_i) = \frac{f(x_i + h) - f(x_i)}{(x_i + h) - x_i} = \frac{f(x_i + h) - f(x_i)}{h}$$

where $x_i \in [a, b)$, and that $x_{i+1} = x_i + h$ for some $h > 0$ that is sufficiently small to ensure that $x_{i+1} \in [a, b]$.

Follow the below procedure to get the Numerical Differentiation of the function $f(x)$ using Forward-difference quotient:

1. Get the values of $a$, $b$, and $n$.
2. Compute $h = (b-a)/n$
3. Get or compute the values of $x_i$ and $f(x_i)$, for each $i$ where $i=0,1,2, \dots n$, and ensuring that $x_0$=a, and $x_n$=b, where $x_{i+1} = x_i + h$ for $i=0,1,2, \dots ,n-1$
4. Calculate the differentiation of the $f(x_i) \leftarrow (f(x_{i+1})-f(x_i))/h$, for $i=0,1,2, \dots ,n-1$

**Sample Input/output:**

```
user@host:~
user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - -
  i     x[i]      f(x[i])   f'(x[i])
- - - - - - - - - - - - - - - - - - -
  0   0.000000   1.000000   0.200000
  1   0.200000   1.040000   0.600000
  2   0.400000   1.160000   1.000000
  3   0.600000   1.360000   1.400000
  4   0.800000   1.640000   1.800000
  5   1.000000   2.000000   2.200000
  6   1.200000   2.440000   2.600000
  7   1.400000   2.960000   3.000000
  8   1.600000   3.560000   3.400000
  9   1.800000   4.240000   3.800000
 10   2.000000   5.000000     - - -
- - - - - - - - - - - - - - - - - - -
```

**Tasks:**

1. Write a program to find numerical differentiation of the function $f(x) = x^2 + 1$, where $x \in [0.0, 1.0]$ and the number of interval n = 10, using the 'Forward-difference quotient' .
2. Write a program to find $f'(x)$ of the following tabulated function $f(x)$ using the 'Forward-difference quotient'.

| $x$ | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1.0 | 1.04 | 1.16 | 1.36 | 1.64 | 2.0 | 2.44 | 2.96 | 3.56 | 4.24 | 5.0 |

## Problem 3.2: Backward-difference quotient

**Learning Objectives:**

- Understand the Backward-difference quotient.
- Use Backward-difference quotient for computing the Numerical Differentiation of a function.

**Backward-difference quotient:**

Let us consider a function $f(x)$, where $x \in [a, b]$ and $f \in C^2[a, b]$. Then to approximate $f'(x)$ using the 'Backward-difference quotient', we can use the following formula

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} = \frac{f(x_i) - f(x_{i-1})}{h} \quad \text{or} \quad f'(x_i) = \frac{f(x_i) - f(x_i - h)}{x_i - (x_i - h)} = \frac{f(x_i) - f(x_i - h)}{h}$$

where $x_i \in (a, b]$, and that $x_{i-1} = x_i - h$ for some $h > 0$ that is sufficiently small to ensure that $x_{i-1} \in [a, b]$.

Follow the below procedure to get the Numerical Differentiation of the function $f(x)$ using Backward-difference quotient:

1. Get the values of $a$, $b$, and $n$.
2. Compute $h = (b-a)/n$
3. Get or compute the values of $x_i$ and $f(x_i)$, for each $i$ where $i=0,1,2, \ldots n$, and ensuring that $x_0$=a, and $x_n$=b, where $x_{i-1} = x_i - h$ for $i=1,2,3, \ldots ,n$
4. Calculate the differentiation of the $f(x_i) \leftarrow (f(x_i)-f(x_{i-1}))/h$, for $i=1,2,3, \ldots ,n$

**Sample Input/output:**

```
user@host:~

user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - -
  i     x[i]       f(x[i])   f'(x[i])
- - - - - - - - - - - - - - - - - - -
  0   0.000000   1.000000     - - -
  1   0.200000   1.040000   0.200000
  2   0.400000   1.160000   0.600000
  3   0.600000   1.360000   1.000000
  4   0.800000   1.640000   1.400000
  5   1.000000   2.000000   1.800000
  6   1.200000   2.440000   2.200000
  7   1.400000   2.960000   2.600000
  8   1.600000   3.560000   3.000000
  9   1.800000   4.240000   3.400000
 10   2.000000   5.000000   3.800000
- - - - - - - - - - - - - - - - - - -
```

**Tasks:**

1. Write a program to find numerical differentiation of the function $f(x) = x^2 + 1$, where $x \in [0.0, 1.0]$ and the number of interval n = 10, using the 'Backward-difference quotient' .
2. Use the following data, write a program to find $f'(x)$ using the 'Backward-difference quotient'.

| $x$ | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1.0 | 1.04 | 1.16 | 1.36 | 1.64 | 2.0 | 2.44 | 2.96 | 3.56 | 4.24 | 5.0 |

# Problem 3.3: Central-difference quotient

**Learning Objectives:**

- Understand the Central-difference quotient.
- Use Forward-difference quotient for computing the Numerical Differentiation of a function.

**Central-difference quotient:**

Let us consider a function $f(x)$, where $x \in [a, b]$ and $f \in C^2[a, b]$. Then to approximate $f'(x)$ using the 'Central-difference quotient', we can use the following formula

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f(x_{i+1}) - f(x_i)}{h} \quad \text{or} \quad f'(x_i) = \frac{f(x_i + h) - f(x_i)}{(x_i + h) - x_i} = \frac{f(x_i + h) - f(x_i)}{h}$$

where $x_i \in (a, b)$, and that $x_{i+1} = x_i + h$ & $x_{i-1} = x_i - h$ for some $h > 0$ that is sufficiently small to ensure that $x_{i-1}$, $x_{i+1} \in [a, b]$.

Follow the below procedure to get the Numerical Differentiation of the function $f(x)$ using Central-difference quotient:

1. Get the values of $a$, $b$, and $n$.
2. Compute $h = (b-a)/n$
3. Get or compute the values of $x_i$ and $f(x_i)$, for each $i$ where $i=0,1,2, \ldots n$, and ensuring that $x_0$=a, and $x_n$=b, where $x_{i+1} = x_i + h$ for $i=0,1,2, \ldots ,n-1$
4. Calculate the differentiation of the $f(x_i) \leftarrow$ $(f(x_{i+1})-f(x_{i-1}))/(2h)$, for $i=1,2,3, \ldots ,n-1$

**Sample Input/output:**

```
user@host:~
user@host:~$ ./a.out

- - - - - - - - - - - - - - - - - - - -
  i     x[i]      f(x[i])   f'(x[i])
- - - - - - - - - - - - - - - - - - - -
  0   0.000000   1.000000    - - -
  1   0.200000   1.040000   0.400000
  2   0.400000   1.160000   0.800000
  3   0.600000   1.360000   1.200000
  4   0.800000   1.640000   1.600000
  5   1.000000   2.000000   2.000000
  6   1.200000   2.440000   2.400000
  7   1.400000   2.960000   2.800000
  8   1.600000   3.560000   3.200000
  9   1.800000   4.240000   3.600000
 10   2.000000   5.000000    - - -
- - - - - - - - - - - - - - - - - - - -
```

**Tasks:**

1. Write a program to find numerical differentiation of the function $f(x) = x^2 + 1$, where $x \in [0.0, 1.0]$ and the number of interval n = 10, using the 'Central-difference quotient' .
2. Write a program to find $f'(x)$ of the following tabulated function $f(x)$ using the 'Central-difference quotient'.

| $x$ | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1.0 | 1.04 | 1.16 | 1.36 | 1.64 | 2.0 | 2.44 | 2.96 | 3.56 | 4.24 | 5.0 |