

# CBNST LAB PRACTICAL

Date: September 16, 2021

Name : Deepankar Sharma  
Course: BCA  
Student Id : 20041299  
University Roll No: 2092014

## Iteration Method

### Algorithm:

1. Start
2. Define function as  $f(x)$
3. Define convergent form  $g(x)$
4. Input:
  - a. Initial guess  $x_0$
  - b. Tolerable Error  $e$
  - c. Maximum Iteration  $N$
5. Initialize iteration counter:  $step = 1$
6. Do
  - $x_1 = g(x_0)$
  - $step = step + 1$
  - If  $step > N$ 
    - Print "Not Convergent"
    - Stop
  - End If
  - $x_0 = x_1$
  - While  $abs\ f(x_1) > e$
7. Print root as  $x_1$
8. Stop

# CBNST LAB PRACTICAL

Date: September 16, 2021

## Code:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define phi(x) (7 + log10(x)) / (2)

// double differential(float x0)
// {
//     double grad = phi(x0 + 0.0000001 * x0) - phi(x0);
//     grad /= (0.0000001 * x0);

//     printf("gradient= %f\n", grad);
//     if (grad < 0)
//     {
//         grad = grad - 2 * grad;
//     }
//     printf("|gradient|= %f\n", grad);

//     return grad;
// }

double differential(double x0)
{
    const double delta = 1.0e-10;
    double x1 = x0 - delta;
    double x2 = x0 + delta;

    double y1 = phi(x1);
    double y2 = phi(x2);

    double grad= (y2 - y1) / (x2 - x1);
    printf("gradient= %f\n", grad);
    if (grad < 0)
    {
        grad = grad - 2 * grad;
    }
    printf("|gradient|= %f\n", grad);

    return grad;
```

# CBNST LAB PRACTICAL

Date: September 16, 2021

```
}
```

```
int main()
{
    int k = 0;
    double x1, x0;
    double allErr;
    printf("Enter the allowed Error: ");
    scanf("%lf", &allErr);
    int i1, i2;
    printf("Enter the interval lower limit: ");
    scanf("%d", &i1);
    printf("Enter the interval upper limit: ");
    scanf("%d", &i2);
```

```
    printf("\nEnter the initial guess x0: ");
    scanf("%lf", &x0);
    {
        if (x0 <= i2 && x0 >= i1)
        {
            double grad = differential(x0);
```

```
            if (grad <= 1)
            {
                x1 = x0;
                do
                {
                    k++;
                    x0 = x1;
                    x1 = phi(x0);
                    printf("x after iteration %d is: %lf\n", k, x1)
;
                } while (fabs(x1 - x0) > allErr);
            }
```

```
        else
        {
            printf("|gradient|=%f is not less than 1, Hence cannot apply Iteration Method !!!!\n", grad);
```

```
        exit(0);
```

# CBNST LAB PRACTICAL

Date: September 16, 2021

```
    }  
  
    printf("\n\nOne root is %lf obtained at %d th iteratio  
n \n", x1, k);  
    }  
    else  
    {  
        printf("You entered wrong initial guess, needed someth  
ing between %d and %d !!!", i1, i2);  
    }  
}  
}
```

The screenshot displays the Visual Studio Code interface with the file `03_iterationMethod.cpp` open. The code defines a function `double differential(float x0)` that calculates the root of a function using the iteration method. The function includes headers `<stdio.h>`, `<math.h>`, and `<stdlib.h>`. It defines a function `phi(x) = (7 + log10(x)) / (2)`. The `differential` function takes an initial guess `x0` and iterates until the allowed error (0.0001) is reached. The terminal output shows the execution of the program, where the user enters an initial guess of 1, and the program outputs the root value 3.789275 after 5 iterations.

```
1 #include <stdio.h>  
2 #include <math.h>  
3 #include <stdlib.h>  
4  
5 #define phi(x) (7 + log10(x)) / (2)  
6  
7 // double differential(float x0)  
8 // {  
  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
Windows PowerShell  
Copyright (C) 2015 Microsoft Corporation. All rights reserved.  
  
PS E:\03 Semester\CBNST\Unit 01> cd "E:\03 Semester\CBNST\Unit 01\"; if ($?) { g++ 03_iterationMethod.cpp -o 03_iterationMethod }; if ($?) { .\03_iterationMethod }  
Enter the allowed Error: 0.0001  
Enter the interval lower limit: 0  
Enter the interval upper limit: 1  
  
Enter the initial guess x0: 1  
gradient= 0.217146  
[gradient]= 0.217146  
x after iteration 1 is: 3.500000  
x after iteration 2 is: 3.772034  
x after iteration 3 is: 3.788288  
x after iteration 4 is: 3.789221  
x after iteration 5 is: 3.789275  
  
One root is 3.789275 obtained at 5 th iteration  
PS E:\03 Semester\CBNST\Unit 01> ]
```