iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** XII **Month of publication:** December 2019

DOI: http://doi.org/10.22214/ijraset.2019.12014

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XII, Dec 2019- Available at www.ijraset.com

Newton Gregory Method

Vishal V. Mehtre[1], Sudhanshu Pathak[2]
[1]*Assistant Professor,* [2]*Student, Department of Electrical Engineering, Bharati Vidyapeeth Deemed(To Be) University, College of Engineering, Pune, India*

Abstract: ***This paper shows how Average value based approach and Newton Gregory Formulae can be used in a successful way to model the locomotion of human knee joint. A Novel modeling technique based on Average Value algorithm has been developed for a healthy human knee joint locomotion. The developed mathematical model will become a guide line for the design of drive mechanism having similar motion. The requirements of this approach are a set of reading from the real time system. In this paper we considered subjects performing gait on normal floor. The knee joint locomotion is modeled from the data acquired by means of calculating the base value and the variational components. The entire procedure is achieved through the video picture of the locomotion captured by single or multiple cameras with proper resolution. The results obtained from the proposed model and actual results of locomotion of human knee joint were giving close results.***

I. INTRODUCTION

Interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable, while the process of computing the value of the function outside the given range is called extrapolation.
Forward Differences: [1]
The differences
y1 – y0, y2 – y1, y3 – y2, ……, yn – yn–1
when denoted by
dy0, dy1, dy2, ……, dyn–1
are respectively, called the first forward differences. Thus the first forward differences are :

A. Newton's Gregory Forward Interpolation Formula : [2]

This formula is particularly useful for interpolating the values of f(x) near the beginning of the set of values given. h is called the interval of difference and **u = (x – a) / h**, Here a is first term.

1) Example

a) Input : Value of Sin 52
b) Output : Value at Sin 52 is 0.788003

Below is the implementation of newton forward interpolation method. [3]

C++

```
// CPP Program to interpolate using
// newton forward interpolation
#include <bits/stdc++.h>
using namespace std;

// calculating u mentioned in the formula
float u_cal(float u, int n)
{
    float temp = u;
    for (int i = 1; i < n; i++)
        temp = temp * (u - i);
    return temp;
}
// calculating factorial of given number n
int fact(int n)
```

```
{
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}
int main()
{
    // Number of values given
    int n = 4;
    float x[] = { 45, 50, 55, 60 };
    // y[][] is used for difference table
    // with y[][0] used for input
    float y[n][n];
    y[0][0] = 0.7071;
    y[1][0] = 0.7660;
    y[2][0] = 0.8192;
    y[3][0] = 0.8660;
    // Calculating the forward difference
    // table
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < n - i; j++)
            y[j][i] = y[j + 1][i - 1] - y[j][i - 1];
    }
    // Displaying the forward difference table
    for (int i = 0; i < n; i++) {
        cout << setw(4) << x[i]
             << "\t";
        for (int j = 0; j < n - i; j++)
            cout << setw(4) << y[i][j]
                 << "\t";
        cout << endl;
    }
    // Value to interpolate at
    float value = 52;

    // initializing u and sum
    float sum = y[0][0];
    float u = (value - x[0]) / (x[1] - x[0]);
    for (int i = 1; i < n; i++) {
        sum = sum + (u_cal(u, i) * y[0][i]) /
                              fact(i);
    }
    cout << "\n Value at " << value << " is "
         << sum << endl;
    return 0;

}
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XII, Dec 2019- Available at www.ijraset.com

2) Output

```
45  0.7071  0.0589  -0.00569999  -0.000699997
50  0.766  0.0532  -0.00639999
55  0.8192  0.0468
60  0.866
Value at 52 is 0.788003
```

3) Backward Differences: The differences y1 – y0, y2 – y1, ……, yn – yn–1 when denoted by dy1, dy2, ……, dyn, respectively, are called first backward difference. Thus the first backward differences are[3]

B. Newton's Gregory Backward Interpolation Formula

This formula is useful when the value of f(x) is required near the end of the table. h is called the interval of difference and u = (x – an)/h, Here an is last term. [4]

1) Example

a) Input: Population in 1925

b) Output: Value in 1925 is 96.8368

Below is the implementation of newton backward interpolation method. [4]

C++

```
// CPP Program to interpolate using
// newton backward interpolation
#include <bits/stdc++.h>
using namespace std;
// Calculation of u mentioned in formula
float u_cal(float u, int n)
{
    float temp = u;
    for (int i = 1; i < n; i++)
        temp = temp * (u + i);
    return temp;
}
// Calculating factorial of given n
int fact(int n)
{
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}
int main()
{
    // number of values given
    int n = 5;
    float x[] = { 1891, 1901, 1911,
                1921, 1931 };
    // y[][] is used for difference
    // table and y[][0] used for input
    float y[n][n];
    y[0][0] = 46;
    y[1][0] = 66;
    y[2][0] = 81;
```

```
    y[3][0] = 93;
    y[4][0] = 101;
    // Calculating the backward difference table
    for (int i = 1; i < n; i++) {
        for (int j = n - 1; j >= i; j--)
            y[j][i] = y[j][i - 1] - y[j - 1][i - 1];
    }
    // Displaying the backward difference table
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++)
            cout << setw(4) << y[i][j]
                 << "\t";
        cout << endl;
    }
    // Value to interpolate at
    float value = 1925;
    // Initializing u and sum
    float sum = y[n - 1][0];
    float u = (value - x[n - 1]) / (x[1] - x[0]);
    for (int i = 1; i < n; i++) {
        sum = sum + (u_cal(u, i) * y[n - 1][i]) /
                                fact(i);
    }
    cout << "\n Value at " << value << " is "
         << sum << endl;
    return 0;
}
```

2) Output

```
 46
 66     20
 81     15     -5
 93     12     -3      2
101      8     -4     -1     -3
Value at 1925 is 96.8368
```

II. CONCLUSION

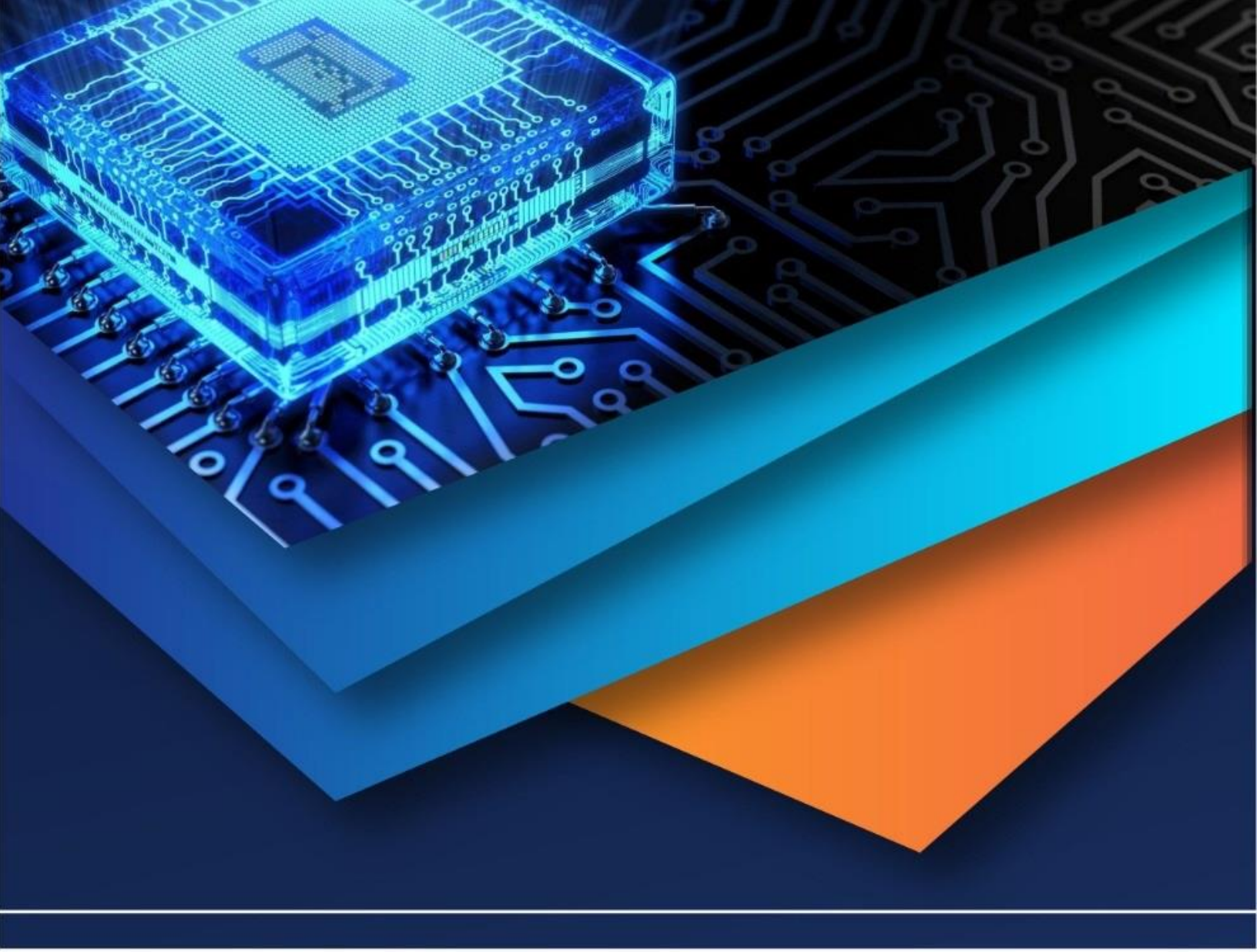
According to the analysis the performance of Newton interpolation formula on different types of functions is presented. Experimental results show that for reconstructing a signal it works better for the area where signal values are relatively constant or increasing. In conclusion, it can be said that this formula is designed for a function whose value will increase or remain constant with the independent variable.

III. ACKNOWLEDGEMENT

We would like to express our special thanks of greatefullness to Dr. D.S. Bankar, Head, Department of electrical engineering for able Guidance and support for completing the research paper. I would like to thank the faculty member of the department of electrical engineering who helped us with extended support

REFERENCES

[1] Elements of numerical analysis
[2] Radhey S. Gupta Retrieved 2016-10-08
[3] PI A source Book (edition 2015)
[4] W. Weisstein,Eric. "Extended Mean-Value Theorem" Retrieved 2018-10-08.
[5] "Cauchy's Mean Value Theorem" Retrieved 2018-10-08.
[6] Besenyei, A. (September 17, 2016). "A brief history of the mean value theorem"



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY