Applied Mathematics & Information Sciences Letters
An International Journal

http://dx.doi.org/10.12785/amisl/010301

Using the generated function as a numerical differentiation method

*Sameh Abdelwahab Nasr Eisa**

Department of mathematics,New Mexico Institute of Mining and Technology, USA

Received: 2 Mar. 2013, Revised: 13 Jul. 2013, Accepted: 18 Jul. 2013
Published online: 1 Sep. 2013

Abstract: In this paper we present a method and algorithm for computing the differentiation numerically at specific point. We have shown results for a lot of functions at many different points, and compared them with results from other methods to show the applicability and the simplicity for the proposed method. We introduced the algorithm through Matlab code to be used directly from the users.

Keywords: numerical differentiation, numerical methods, polynomials, derivatives computing.
AMS Subject classification: 65D25, 65D05, 65L99**.**

1 Introduction

Nowadays, numerical differentiation is not an attractive area of interest for most of the researchers, as it has been investigated deeply for decades after Newton's introduction for the calculus science. A lot of historical contributions still in use until this time by the researchers. As an example, using Lagrange interpolation to deduce the derivatives. The idea of using Lagrange interpolation depends on approximating an interval from a function to specific polynomial with a specific degree, and then we apply the differentiation on the deduced polynomial instead of the function.

We introduced in our review paper [7] the possibility of using the generated function to replace an interval from the function to polynomials. Although, the generated function gives the same results as Lagrange interpolation, it depends on deducing the polynomial's coefficients using simple and programmable algorithm by solving system of linear equations. The simplicity of using the generated function [7] inspired us to enlarge our imagination, so we started questioning the ability of using the generated function as a numerical integration method, which led us to the paper [9] and using it for solving non linear equations, which led us to the paper [8].

We have cited number of text books, which helped us through our literature review [1,2,3,6] and [10], these texts substituted the lack of modern papers in numerical differentiation. Van der Monde matrix, played an essential rule in the development of Lagrange interpolation, so it was a logical call to cite the paper [5] to be aware about this contribution. We have cited some of the other works, which seemed interesting and relevant to us during our research.

In this paper we investigated and verified the possibility of using the generated function as a numerical differentiation method. As a result, we introduced an algorithm, which simulates the proposed method, in order to compute the derivatives numerically for a function at specific points. The algorithm is not limited to the computation of the first derivative, but it can efficiently compute further orders of derivatives like the second, the third...etc.

Our method and the resultant algorithm depend on the approximation of an interval from the function to polynomials using the procedures in [7], and then we apply the differentiation on the polynomials. The results urge us to present this paper to the audience, especially the engineers or the applied mathematicians, because our method and algorithm will allow the scientific community to compute the derivatives smoothly and we were working to ensure that by providing Matlab code in the paper, which is ready to be used.

* Corresponding author e-mail: seisa@nmt.edu,sameheisa235@hotmail.com

2 The methodology

We can assume the generated function that replaces an interval from the function to be differentiated in a general polynomial form

$$g(x)|_{x_1}^{x_2} = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 \ldots a_n x^n \,, \qquad (1)$$

x_1 & x_2 *are the interval limits*

We can format the derivathves of $g(x)$to be

$$\begin{aligned} g(x)^{'} &= a_1 + 2a_2 x \ \ldots n a_n x^{n-1}, \\ g(x)^{''} &= 2a_2 + 3a_3 x \ldots n(n-1)\, a_n x^{n-2}, \end{aligned} \qquad (2)$$

It is noticeable and by induction that for the derivative of order, m

$$g(x)^{(m)} = \sum_{i=0}^{n-m} \frac{(i+m)!}{i!} a_{i+m} x^i \quad 0 \le m \le N \qquad (3)$$

2.1 The concept of limitation

We have to consider and choose the degree n for the polynomial wisely. We can't consider high degrees of n because of the oscillations of the high polynomial degrees. On the other hand, we can't just consider low degree of n because the accuracy of the approximation will decrease. However, we will make a limitation for choosing the degree n, as we will choose low degrees relatively but in a narrow intervals. As we focus on computing the derivatives at specific point, we will consider a small interval that contains this point, and make the approximation for this interval using the generated function with low degrees relatively.

2.2 The logical steps for making the nemerical differentiation

First: The definition of the interval

In this step we must to define an interval that contains the point on x-axis in which the derivatives are calculated over it. We will assume this point to be x_0 and the limits of the interval will be $[\chi_0 - \Delta\chi, \chi_0 + \Delta\chi]$ the value of $\Delta\chi$ must be small ratio like 0.01 or 0.1.

Second: detenmining $g(x)$ for the inrterval

Generally, by assuming $g(x)$ with n degree and $n+1$ terms, then we will need $n+1$ points on the interval $[\chi_0 - \Delta\chi, \chi_0 + \Delta\chi]$ in order to determine the coefficients $a_0, a_1, a_2 \ldots a_n$. As a result,

$$d = \frac{(x_0 + \Delta x) - (x_0 - \Delta x)}{n} = \frac{2\Delta x}{n} \qquad (4)$$

d is the distance between two sucssesive points

The system of linear equations used fon determiring $a_0, a_1, a_2 \ldots a_n$ matrix form

$$XA = F \qquad (5)$$

Where,

$$A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}_{(n+1)x1} , F = \begin{bmatrix} f(\chi_0 - \Delta\chi) \\ f(\chi_0 - \Delta\chi + d) \\ f(\chi_0 - \Delta\chi + 2d) \\ \vdots \\ f(\chi_0 + \Delta\chi) \end{bmatrix}_{(n+1)x1}$$

$$and\ X = \begin{bmatrix} 1 & (\chi_0 - \Delta\chi) & (\chi_0 - \Delta\chi)^2 & \ldots & (\chi_0 - \Delta\chi)^n \\ 1 & (\chi_0 - \Delta\chi + d) & (\chi_0 - \Delta\chi + d)^2 & \ldots & (\chi_0 - \Delta\chi + d)^n \\ 1 & (\chi_0 - \Delta\chi + 2d) & (\chi_0 - \Delta\chi + 2d)^2 & \ldots & (\chi_0 - \Delta\chi + 2d)^n \\ \vdots & \vdots & & & \\ 1 & (\chi_0 + \Delta\chi) & (\chi_0 + \Delta\chi)^2 & \ldots & (\chi_0 + \Delta\chi)^n \end{bmatrix}_{(n+1)x(n+1)} \qquad (6)$$

Third: apply the differentiation on the deduced $g(x)$ and substitute with values

We will chose a derivaeive order (m) and specific point (x) and then we can use Eq.(3) to determine the final rtsultant value.

2.3 Example collects the logical steps

Example (0): determine the first derivative for Ln(x) at x=5.

Step (1): we will select $\Delta x = 0.1$ then we have interval from 4.9 to 5.1.

Step (2): we will use g(x) with 3 terms (2^{nd} degree)

$$g(\chi)\Big|_{4.9}^{5.1} = \sum_{i=0}^{2} a_i x^i = a_0 + a_1 x + a_2 x^2 \qquad (7)$$

Then, $d = \frac{5.1-4.9}{3-1} = 0.1$ and the system of linear equations will be

$$\begin{aligned} In(4.9) &= a_0 + 4.9a_1 + 4.9^2 a_2 \\ In(5) &= a_0 + 5a_1 + 5^2 a_2 \\ In(5.1) &= a_0 + 5.1a_1 + 5.1^2 a_2 \end{aligned} \qquad (8)$$

After we solve the above three equations then,
$g(x) = 0.1.9204 + 0.400066x - 0.020004x^2$

Step (3): ue will substitute in Eq.(3) with m=1, n=2 and x=5 then $\frac{dg(x)}{dx} = 0.400066 - 2 * 0.020004x$

The result = 0.200026

We know that $\frac{dln(x)}{dx} = \frac{1}{x}$ exactly, so at x=5 the exact result will be 0.2. So, we notice the error amount is 0.000026.

3 Properties of the method

3.1 The restriction of choosing the derivative order (m) and the polynomial degree (n)

By studying Eq.(3), its noticeable that there must be some restrictions. Obviously, we have to ensure that the considered polynomial degree n will not vanish with the derivative order m . As a result, we have to be restricted with the inequality

$$n \geqslant m+1 \tag{9}$$

3.2 The combination between $\Delta\chi, n$ and the accuracy

We prefer to choose reasonable degree of n to avoid both of the oscillations and the decreasing of the accuracy, so, it will be logical not to excide the 10^{th} polynomial degree. Also, it seemed logical to choose $\Delta\chi$ small to have more accurate approximation, but very small values of $\Delta\chi$ may cause singularity and problems in the calculation procedure. As a result, we will prefer to choose it between 0.01 and 0.1. In order to reach the demanded accuracy, we have to achieve consistency in our choice for $\Delta\chi$ and n

4 Matlab code to simulate our algorithm followed by results and comparisons

The code presented in this paper capable of asking the user to enter $\Delta x, n, m$ & X_0 and the program will check the condition explained in the section [3.2.], and then the program will compute the result.

Example (1): We have run the code for $\Delta x = 0.1, n = 10$ & $different\ m$ and we collected the results in table.1 , and we compared it with the results from the ready programs package for numerical differentiation in programs like ”matlab” or ”winplot” and our results have proved the applicability of this method to be used in the scientific computing.

5 Conclucion

We have shown to the scientific community an easy, simple and applicable method and algorithm, in order to compute the derivatives for a function at specific point numerically. Our work is an extension for other works that depend on the generated function [7] to make numerical methods, as we published before a method and algorithm to make numerical integration [9] and solve equations [8], and in this paper we proposed numerical differentiation method following the same concepts.

We believe that this paper will help the scientific community by the attached Matlab code, which is ready to run. The comparisons and the examples introduced in this paper must urge the audience to use the proposed algorithm and code.

The code:

```
format long
%the below three lines to store Δx,m & n %
delta=input('enter the value of delta_x:  ');
m=input('enter the order of the derivative:   ');
n=input('enter the polynomial degree of g(x):  ');
%the below if condition to check the condition in [3.2.]%
if n>=m+1
    %the below line to enter the value of x_0  %
    x_0=input('enter value on x_axis:  ');
    d=2*delta/n;
    x=x_0-delta:d:x_0+delta;
    %the below line to ask the user to enter the function%
    F=input('enter function to be differentiated f(x):   ');
    X=[];
    %the below for loops to construct the matrix in Eq.(6)%
    for i=1:n+1
        B=[];
        for j=1:n+1
            B=[B (x_0-delta+(i-1)*d)^(j-1)];
        end
        X=[X;B];
    end
    A=X\F';
    result=0;
    %the below for loop to compute the final result Eq.(3)%
    for i=0:n-m

result=result+(factorial(i+m)/factorial(i))*A(i+1+m)*x_0^i;
    end
    result
else
    disp('error in the condition n>=m+1')
end
```

note: the lines with % in the start and in the end are in a green color only for explanation.

Table 1: maay computations for functions and several derivatives compared with some of the other known results

Function to be differentiated	Value of x_0	Value of m	Result from our method	Result from other numerical methods
Ln(x)	3	1	0.33333	0.33333
Ln(x)	4	2	-0.062499	-0.0625
Ln(x)	5	3	0.016	0.016
(2^x)/ Ln(x)	8	2	50.70416	50.70416
x^3	3	1	27	27
sin(x)	4	3	0.653643	0.653643
sin(x)	6	2	0.279415	0.27942
(cos(sin(x))^2	9	1	0.6688	0.6688
Exp(x)/(x^3)	10	1	15.418526	15.41853
1/(x^3+2x+1)	2	3	-0.15314526	-0.15315
1/ln(x)	4	2	0.0794398	0.07944
Exp(-x^2)	1	2	0.73575888	0.73576
xsin(x^2)	15	1	-10.74503	-10.74503

References

[1] Dennis G. Zill, Michael R. Cullen, Advanced Engineering Mathematics, Fourth Edition, Jones and Bartlett Publ., MA (2010).
[2] Erwin Kreyszig, Advanced Engineering Mathematics, Seventh Edition, Ohio State University, Columbus, Ohio (2005).
[3] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. Computer Methods for Mathematical Computations. Englewood Cliffs, NJ: Prentice-Hall, (1977).
[4] James M. Hyman and Bernard Larrouturou, The numerical differentiation of discrete functions using polynomial interpolation methods, Applied Mathematics and Computation, **10-12**, 487-506 (1982).
[5] J. N. Lyness and C. B. Moler, Van der Monde systems and numerical differentiation, Numerishe Mathematike, **8**, 458-464 (1966).
[6] Josef Stoer and Roland Bulirsch. Introduction to Numerical Analysis. New York: Springer-Verlag, (1980).
[7] S. A. N. Eisa, Approximation series (function) using polynomials for replacing functions and mathematical curves or random curves, International journal of pure and applied mathematics, **68**, 145-164 (2011).
[8] S. A. N. Eisa, Solving equations numerically and by MATLAB code using the generated function with three terms (second order equation), Journal of computing, **3**, (2011).
[9] S. A. N. Eisa, Using the Generated Function as a Numerical Integration Method with High Accurate Results, Applied Mathematical Sciences, **5**, 3811-3826 (2011).
[10] Sloughter, Dan, Difference Equations to Differential Equations (2000).

S. Eisa is a PHD student at New Mexico Tech, USA. Received Bachelor degree in electronics, communications engineering from Alexandria University, Egypt. All of his post graduate studies are in applied mathematics. His research interests are in applied and engineering mathematics, mathematical modeling, computational mathematics and processing. The kind of researches where the mathematics plays an essential role are the main core of his interests. He joined recently some journals and conferences as editorial member or reviewer.