Computationally efficient differenceless derivatives with equidistant steps and it's applications

Yuri Mahotin
email: yuri.mahotin@yahoo.com

Computationally efficient differenceless derivatives with equidistant steps have been developed, which makes it possible to calculate an unlimited number of derivatives. The new algorithm can be applied in various fields of science and technology. As an example, we provide step-by-step instructions on how to improve the accuracy of the predicted trajectory of a flying missile.

Derivatives, and especially numerical derivatives, are widely used in mathematics, physics, applied sciences, and almost anywhere where they are needed. Differenceless derivatives with equidistant steps were presented in work [1]. The basic equation for calculating n derivatives, written in matrix form is

$$\begin{pmatrix} (\sum i^2)h & \frac{1}{2!}(\sum i^3)h^2 & \frac{1}{3!}(\sum i^4)h^3 & \dots & \frac{1}{n!}(\sum i^{n+1})h^n \\ (\sum i^3)h & \frac{1}{2!}(\sum i^4)h^2 & \frac{1}{3!}(\sum i^5)h^3 & \dots & \frac{1}{n!}(\sum i^{n+2})h^n \\ (\sum i^4)h & \frac{1}{2!}(\sum i^5)h^2 & \frac{1}{3!}(\sum i^6)h^3 & \dots & \frac{1}{n!}(\sum i^{n+3})h^n \\ \dots & \dots & \dots & \dots & \dots \\ (\sum i^{n+1})h & \frac{1}{2!}(\sum i^{n+2})h^2 & \frac{1}{3!}(\sum i^{n+3})h^3 & \dots & \frac{1}{n!}(\sum i^{n+n})h^n \end{pmatrix} \begin{pmatrix} f' \\ f'' \\ f''' \\ \dots \\ f^{(n)} \end{pmatrix} = \begin{pmatrix} \sum (i\Delta f(ih)) \\ \sum (i^2\Delta f(ih)) \\ \sum (i^3\Delta f(ih)) \\ \dots \\ \sum (i^n\Delta f(ih)) \end{pmatrix} \quad (1)$$

where h is step size for derivatives estimation, $\Delta f(h)=f(x_0+h)-f(x_0)$, x_0 is the point at which derivatives are calculated, n - number of derivatives, $\sum()\rightarrow\sum_{i=1}^{m}()$, m - the number of points for which the inequality $m\geq n$ must be satisfied. One of the significant inconveniences of using expression (1) in practice is that for each new value of h it is necessary to find the inverse matrix or re-solve the system of linear equations, which can be time-consuming for a large value of n. We will transform this expression into a form most convenient for solving practical problems.

We will limit ourselves to the special case of equation (1) when the number of points m is equal to the number of derivatives n, and therefore the symbol $\sum()\rightarrow\sum_{i=1}^{n}()$. First, let's look at the left side of equation (1). For brevity of description, we introduce the vector of derivatives $\boldsymbol{d}(x_0)=\left(f'(x_0),f''(x_0),f'''(x_0),\dots,f^{(n)}(x_0)\right)^T$. we define two matrices $\boldsymbol{H}$ and $\boldsymbol{S}$

$$\boldsymbol{H}=\begin{pmatrix} h & 0 & 0 & ... & 0 \\ 0 & h^2 & 0 & ... & 0 \\ 0 & 0 & h^3 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & h^n \end{pmatrix},\ \boldsymbol{S}=\begin{pmatrix} (\sum i^2) & \frac{1}{2!}(\sum i^3) & \frac{1}{3!}(\sum i^4) & ... & \frac{1}{n!}(\sum i^{n+1}) \\ (\sum i^3) & \frac{1}{2!}(\sum i^4) & \frac{1}{3!}(\sum i^5) & ... & \frac{1}{n!}(\sum i^{n+2}) \\ (\sum i^4) & \frac{1}{2!}(\sum i^5) & \frac{1}{3!}(\sum i^6) & ... & \frac{1}{n!}(\sum i^{n+3}) \\ ... & ... & ... & ... & ... \\ (\sum i^{n+1}) & \frac{1}{2!}(\sum i^{n+2}) & \frac{1}{3!}(\sum i^{n+3}) & ... & \frac{1}{n!}(\sum i^{n+n}) \end{pmatrix}$$

Then the left side of the equation can be written as $\boldsymbol{SH}\,\boldsymbol{d}(x_0)$. Now we transform the right side of the equation, for this we introduce the vector

$\Delta\boldsymbol{f}(\boldsymbol{h})=(f(x_0+h)-f(x_0), f(x_0+2h)-f(x_0), f(x_0+3h)-f(x_0), ..., f(x_0+nh)-f(x_0))^T$ and the matrix

$$\boldsymbol{J}=\begin{pmatrix} 1 & 2 & 3 & ... & n \\ 1 & 2^2 & 3^2 & ... & n^2 \\ 1 & 2^3 & 3^3 & ... & n^3 \\ ... & ... & ... & ... & ... \\ 1 & 2^n & 3^n & ... & n^n \end{pmatrix}$$

In the newly introduced notations, equation (1) is written as $\boldsymbol{SH}\,\boldsymbol{d}(x_0)=\boldsymbol{J}\Delta\boldsymbol{f}(h)$. Formally the solution to this equation is $\boldsymbol{d}(x_0)=\boldsymbol{H}^{-1}\boldsymbol{S}^{-1}\boldsymbol{J}\Delta\boldsymbol{f}(h)$. Next, we define the matrix $\boldsymbol{G}=\boldsymbol{S}^{-1}\boldsymbol{J}$ and finally write the solution to equation (1) in the form $\boldsymbol{d}(x_0)=\boldsymbol{H}^{-1}\boldsymbol{G}\Delta\boldsymbol{f}(h)$. We will note the important advantages of the obtained solution. Firstly, the calculation of derivatives is reduced to simple arithmetic operations, multiplying a matrix by a vector. Secondly, the inverse matrix $\boldsymbol{H}$ can be easily found analytically. Thirdly, the matrix $\boldsymbol{G}$ does not depend on the step size h and point x_0, and does not depend on the function $f(x)$, and therefore, for a given number of unknown derivatives n, it is sufficient to calculate the matrix $\boldsymbol{G}$ once and use it for different functions and steps. For the convenience of subsequent presentation, we will rewrite the last equation in expanded form

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \\ f'''(x_0) \\ ... \\ f^{(n)}(x_0) \end{pmatrix}=\begin{pmatrix} \frac{1}{h} & 0 & 0 & ... & 0 \\ 0 & \frac{1}{h^2} & 0 & ... & 0 \\ 0 & 0 & \frac{1}{h^3} & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \frac{1}{h^n} \end{pmatrix}\begin{pmatrix} g_{11} & g_{12} & g_{13} & ... & g_{1n} \\ g_{21} & g_{22} & g_{23} & ... & g_{2n} \\ g_{31} & g_{32} & g_{33} & ... & g_{3n} \\ ... & ... & ... & ... & ... \\ g_{n1} & g_{n2} & g_{n3} & ... & g_{nn} \end{pmatrix}\begin{pmatrix} f(x_0+h)-f(x_0) \\ f(x_0+2h)-f(x_0) \\ f(x_0+3h)-f(x_0) \\ ... \\ f(x_0+nh)-f(x_0) \end{pmatrix} \quad (2)$$

Where matrix $\boldsymbol{G}$ is represented by elements of matrix g_{ij}.

We will give some examples for expression (2) for small values of unknown derivatives n. Our presentation will be detailed enough to be understood by both students and engineers, as well as people interested in mathematics. All formulas obtained below can be easily implemented in C and other

programming languages. We will start with the simplest case, let two points x_0 and x_0+h be given, then $n=1$, substituting these values into expression (2) we get

$$f'(x)=\frac{1}{h}(f(x_0+h)-f(x_0)) \tag{3}$$

These are simple forward difference approximations of the first derivative, but we are not interested in this, we need to calculate derivatives of the highest possible order. This can be seen as a side effect, for $n=1$ the expression (2) is not a matrix expression. For all other cases $n>1$ we have developed a C program to calculate the matrix $\boldsymbol{G}$ for arbitrary values of n, however there are limitations here which we will discuss as we go along. If three points x_0, x_0+h and x_0+2h are given, then $n=2$ and expression (2) has the form

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \end{pmatrix} = \begin{pmatrix} \frac{1}{h} & 0 \\ 0 & \frac{1}{h^2} \end{pmatrix} \begin{pmatrix} 2.0 & -0.5 \\ -2.0 & 1.0 \end{pmatrix} \begin{pmatrix} f(x_0+h)-f(x_0) \\ f(x_0+2h)-f(x_0) \end{pmatrix} \tag{4}$$

and allows us to calculate two derivatives. For four points x_0, x_0+h, x_0+2h and x_0+3h, three derivatives are calculated using the following expression

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \\ f'''(x_0) \end{pmatrix} = \begin{pmatrix} \frac{1}{h} & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 \\ 0 & 0 & \frac{1}{h^3} \end{pmatrix} \begin{pmatrix} 3.0 & -1.5 & \frac{1}{3} \\ -5.0 & 4.0 & -1.0 \\ 3.0 & -3.0 & 1.0 \end{pmatrix} \begin{pmatrix} f(x_0+h)-f(x_0) \\ f(x_0+2h)-f(x_0) \\ f(x_0+3h)-f(x_0) \end{pmatrix} \tag{5}$$

In the following example the number of points is 5 which allows us to calculate four derivatives

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \\ f'''(x_0) \\ f^{(4)}(x_0) \end{pmatrix} = \begin{pmatrix} \frac{1}{h} & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{h^3} & 0 \\ 0 & 0 & 0 & \frac{1}{h^4} \end{pmatrix} \begin{pmatrix} 4.00 & -3.00 & 1.33 & -0.25 \\ -8.66 & 9.50 & -4.66 & 0.92 \\ 9.00 & -12.0 & 7.00 & -1.50 \\ -4.00 & 6.00 & -4.00 & 1.00 \end{pmatrix} \begin{pmatrix} f(x_0+h)-f(x_0) \\ f(x_0+2h)-f(x_0) \\ f(x_0+3h)-f(x_0) \\ f(x_0+4h)-f(x_0) \end{pmatrix} \tag{6}$$

When calculating expression (6) we encountered an unexpected problem - numerical noise. We have rounded all values to two decimal places. For a more complete understanding of this problem, Appendix A presents the values of the elements of matrix $\boldsymbol{G}$ written in C. We used 64-bit floating point arithmetic, and what surprised us was that even for such a small 4x4 matrix, errors appeared when calculating the inverse matrix. To calculate five derivatives, expression (2) looks like

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \\ f'''(x_0) \\ f^{(4)}(x_0) \\ f^{(5)}(x_0) \end{pmatrix} = \begin{pmatrix} \frac{1}{h} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{h^3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{h^4} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{h^5} \end{pmatrix} \begin{pmatrix} 5.00 & -5.00 & 3.33 & -1.25 & 0.20 \\ -12.8 & 17.8 & -13.0 & 5.08 & 0.83 \\ 17.7 & -29.5 & 25.5 & -10.2 & 1.75 \\ -14.0 & 26.0 & -24.0 & 11.0 & -2.0 \\ 5.00 & -10.0 & 10.0 & -5.00 & 1.00 \end{pmatrix} \begin{pmatrix} \Delta f(h) \\ \Delta f(2h) \\ \Delta f(3h) \\ \Delta f(4h) \\ \Delta f(5h) \end{pmatrix} \quad (7)$$

More precise values of the elements of the matrix $\boldsymbol{G}$ can be found in Appendix B. For the following example $n=6$ we provide formula in a different form, The elements of the matrix $\boldsymbol{G}$ can be found in Appendix C. Here we will make one remark, if in our presentation the indices of the elements of vectors and matrices start with 1, then in the Appendices the indices start with 0.

$$\begin{pmatrix} f'(x_0) \\ f''(x_0) \\ f'''(x_0) \\ f^{(4)}(x_0) \\ f^{(5)}(x_0) \\ f^{(6)}(x_0) \end{pmatrix} = \begin{pmatrix} \frac{1}{h} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{h^3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{h^4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{h^6} \end{pmatrix} \begin{pmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} & g_{16} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} & g_{26} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} & g_{36} \\ g_{41} & g_{42} & g_{43} & g_{44} & g_{45} & g_{46} \\ g_{51} & g_{52} & g_{53} & g_{54} & g_{55} & g_{56} \\ g_{61} & g_{62} & g_{63} & g_{64} & g_{65} & g_{66} \end{pmatrix} \begin{pmatrix} f(x_0+h)-f(x_0) \\ f(x_0+2h)-f(x_0) \\ f(x_0+3h)-f(x_0) \\ f(x_0+4h)-f(x_0) \\ f(x_0+5h)-f(x_0) \\ f(x_0+6h)-f(x_0) \end{pmatrix} \quad (8)$$

For the other two examples $n=7$ and $n=8$ we do not give formulas, as they do not fit on the page, however the matrices $\boldsymbol{G}$ can be found in the Appendices D and E. We limited ourselves to $n=8$ due to the very large numerical noise. After conducting research we found a simple explanation for this fact. Consider the matrix element $\boldsymbol{S}$ in the lower right corner for the next value $n=9$, $\frac{1}{n!}\left(\sum_{i=1}^{9} i^{n+n}\right)$. Let's write the sum as ($1+2^{18}+...+9^{18}$). We use 64-bit floating point arithmetic with an accuracy of about 16 digits after the decimal point, for an example see any Appendix. Thus, the sum from 1 to 9 is replaced by the sum from 2 to 9, but then a completely different matrix $\boldsymbol{S}$ is obtained, and consequently, the solutions of the system of equations will not be the values of the derivatives.

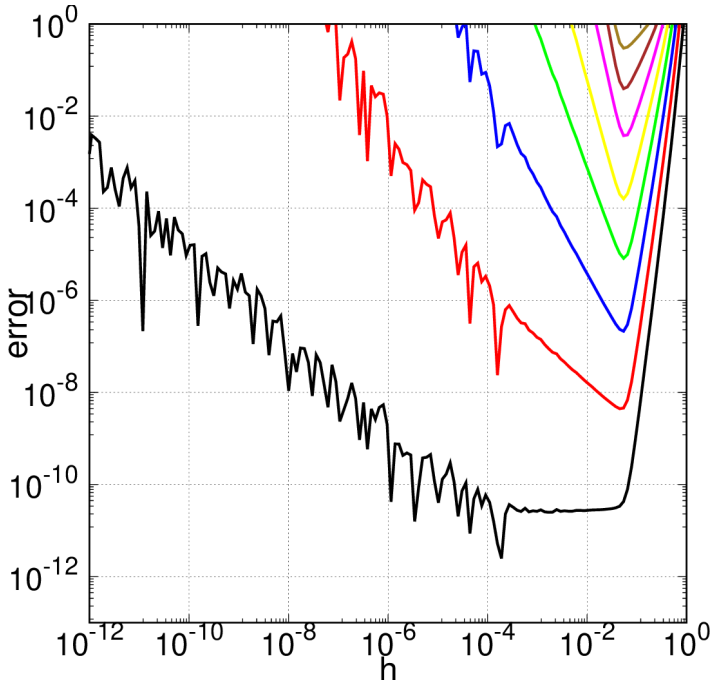

Figure 1

To check if we made any mistakes when obtaining the formulas presented above, we wrote a C program for calculating derivatives based on formulas (3-8) and using the ***G*** matrices from Appendices A, B, C, D, E. First of all, we are interested in the accuracy of calculating derivatives for a large value of $n=8$ and a larger number of derivatives. As usual we took the exponential function $\exp(x)$ at the point $x_0=0$ where all derivatives are equal to one. The calculation results are shown in Figure 1.The lower black curve is the accuracy of the first derivative, the curve just above is the accuracy of the second derivative, and so on up to the last brown curve is the accuracy of the eighth derivative. Some conclusions from the results obtained. As the derivative number increases, the calculation error increases. The optimal step for calculating the derivatives h_{opt} , defined as the minimum of the curves shown, is about 0.03, $h_{opt}\approx 0.03$, and is the same for all eight derivatives. The optimal step is quite large and this is a very good property. Let us recall that the optimal step for forward difference approximation is about 4^{-8}, and it is a very small value. A large optimal step is preferable to use when calculating derivatives from measured data, sometimes we will call this as extraction of derivatives from measured data.

Next, we will consider some aspects of the use of differenceless derivatives for solving applied problems. One of the pressing problems in the real world is predicting the trajectories of various moving objects. For example, an aircraft needs to know the predicted flight path, especially during strong winds, during autopilot, during landing and autolanding. Recently, smart car have become widely used, they calculate the trajectory, and the more accurately the predicted trajectory is, the smarter the car becomes. For ocean-going vessels it is also necessary to know the predicted trajectory, especially in rough stormy weather, when docking and, just in case, to avoid colliding with an oncoming vessel. Of course, let's not forget about interceptor missiles. Nowadays, there are situations when the speed of an interceptor missile is less than the speed of the intercepted missile. In this case, it is necessary to know the predicted trajectories for both missiles very precisely in order to accurately determine the time and direction of the interceptor missile launch. All these problems can be generalized by one mathematical methodology, trajectory prediction using differenceless derivatives. How to calculate these derivatives is described above.

Let's conduct the following thought experiment. Let's assume that the missile moves in a circle with a radius of 1, in what follows we will use relative units of measurement. The coordinates of a moving missile are described by the cosine and sine functions. Without loss of generality, we will

consider only one coordinate x which is described by the cosine $\cos(x)$. Briefly about how we will conduct a thought experiment. For two, three, four, ..., nine points of a known flight path, we will calculate one, two, three, ..., eight derivatives using formula (2). After this, using the Taylor series

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{1}{2!}f''(x_0)(x-x_0)^2+\frac{1}{3!}f'''(x_0)(x-x_0)^3+\cdots+\frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n \qquad (9)$$

and the calculated derivatives, we will predict the trajectory of motion. Our goal is to answer the simple question of whether we can predict the trajectory of a missile. For the experiments we chose a large value $h=-0.1$, the minus sign indicates that we are using known coordinates of the trajectory.

In the simplest experiment, two points $x=(0,-0.1)$ are given, the values of the function $f(x)$ are $\cos(0)$ and $\cos(-0.1)$. The first derivative is calculated using formula (3) and the predicted trajectory using formula (9) for $n=1$. The calculation results are shown in Figure 2, the black curve is the cosine, the two black dots were used to extract the derivative and the red line is the predicted trajectory. It is easy to see that the predicted trajectory does not look like a cosine at all, and therefore two points are not enough to predict a circular trajectory.

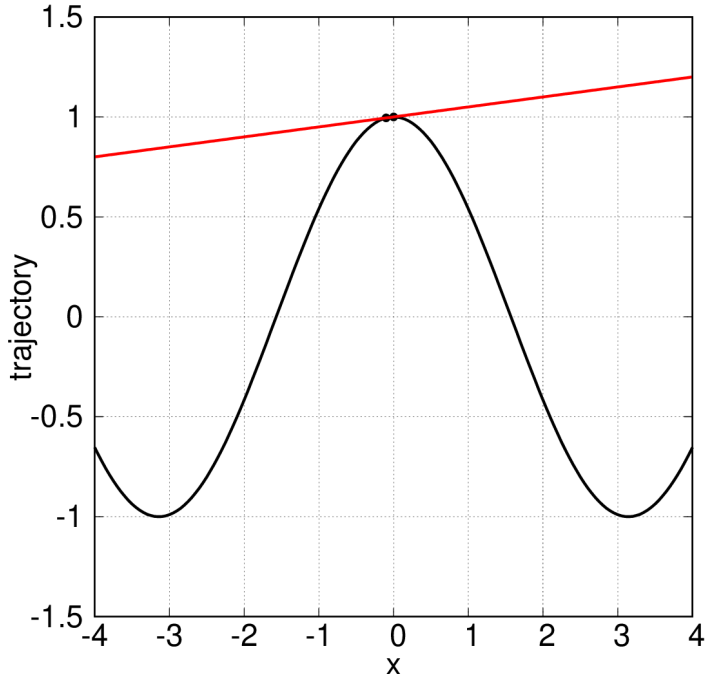

Figure 2

For three points $x=(0,-0.1,-0.2)$ only the first two derivatives can be calculated and the predicted trajectory is shown in Figure 3. The approximation in a small neighborhood of the point x=0 has improved somewhat, but the predictions are also far from cosine, it is only a quadratic function.

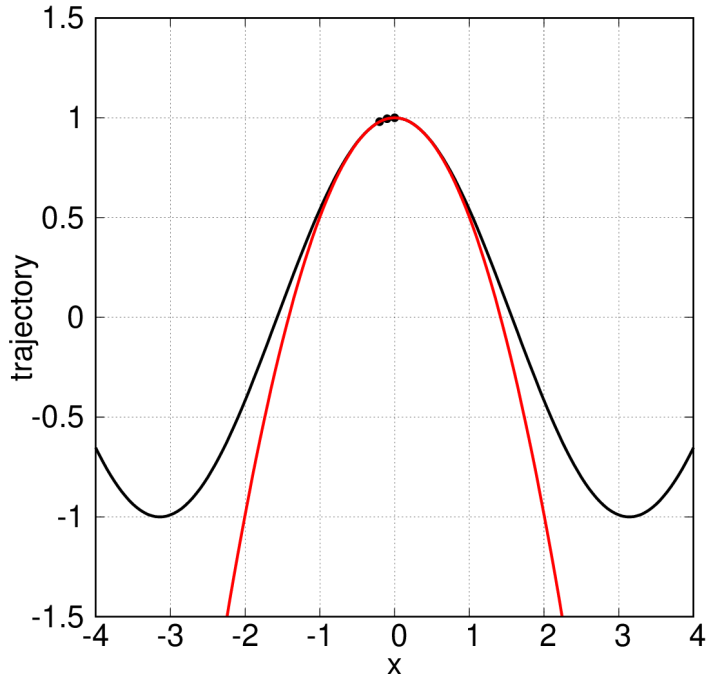

Figure 3

The results of calculating the predicted trajectory for four points are presented in Figure 4. There is no difference here from the previous case, but we leave this example for the sake of completeness.

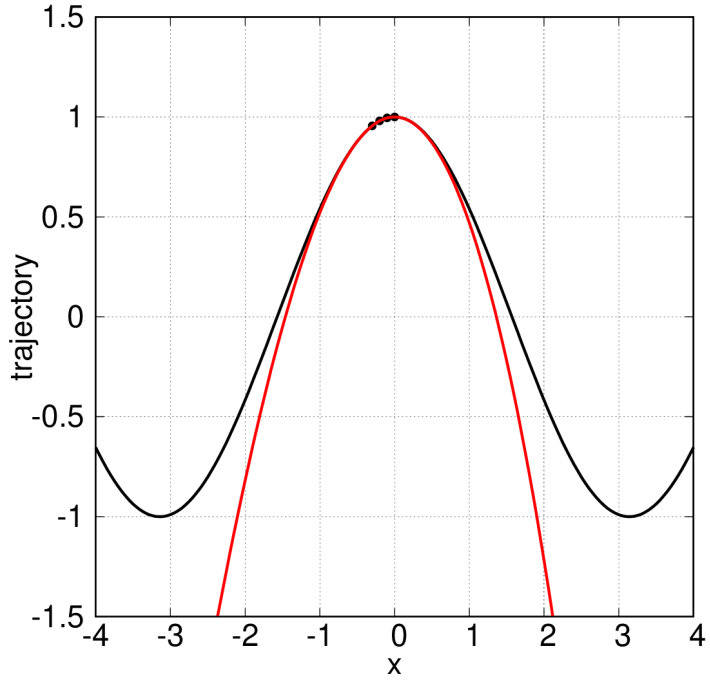

Figure 4

The first interesting result we found was when using five points $x=(0,-0.1,-0.2,-0.3,-0.4)$, in this case four derivatives were calculated using formula (6). The

predicted trajectory, Figure 5, began to resemble a cosine, although the prediction accuracy is still very poor far from $x=0$.

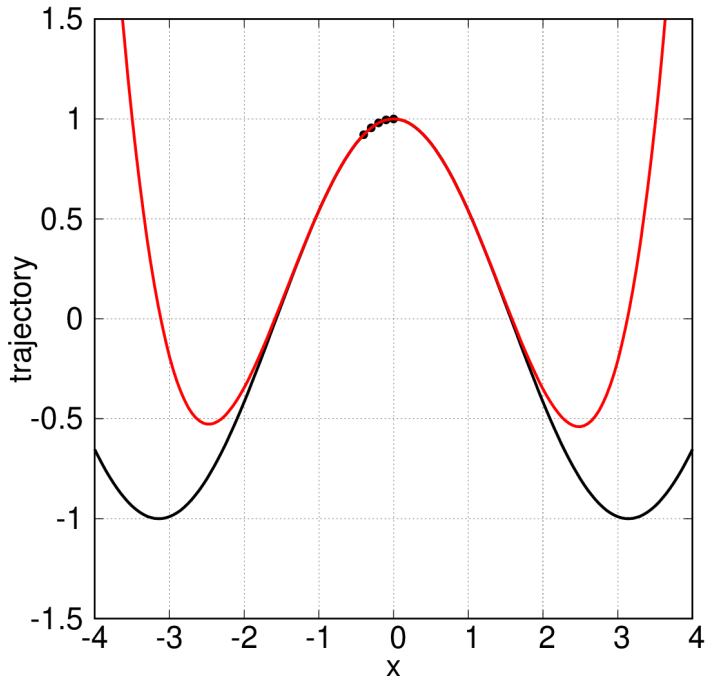

Figure 5

For six points, Figure 6, we obtained a result similar to the previous ones, but with slightly worse accuracy.

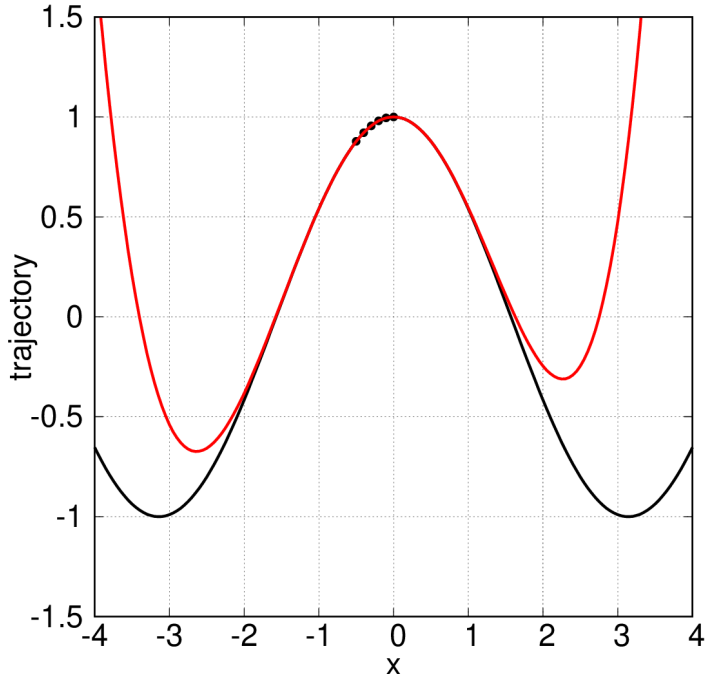

Figure 6

More accurate results are shown by calculations using seven and eight points, Figure 7 and Figure 8 respectively.

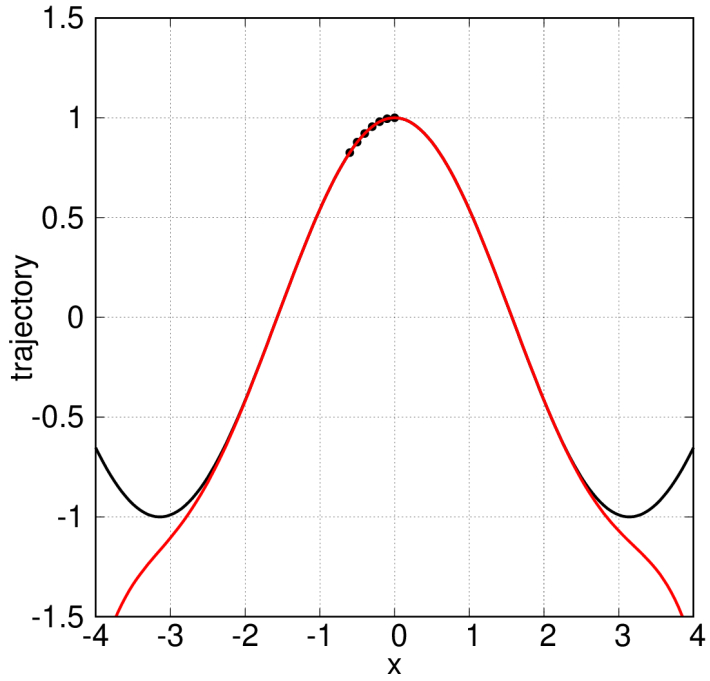

Figure 7

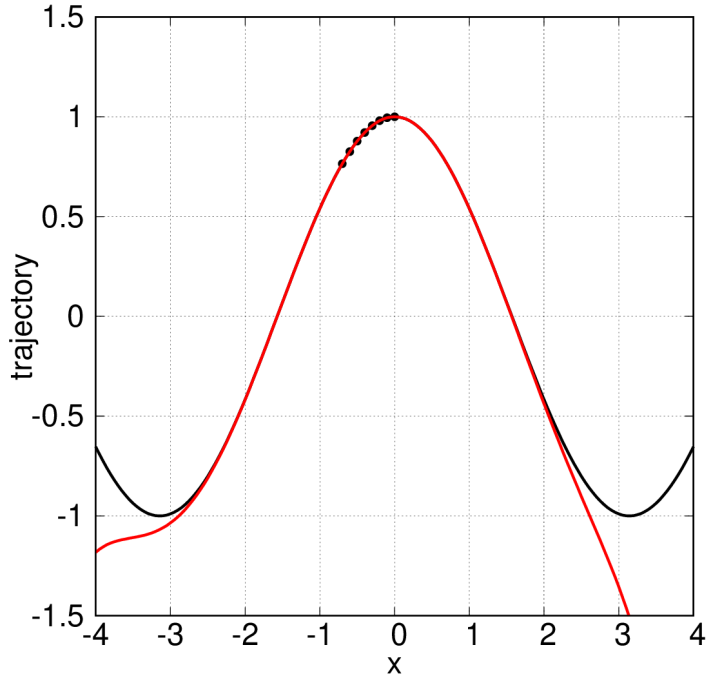

Figure 8

And finally, we got the best results using nine points. In this experiment we set the following values $x=(0,-01,\ldots,-0.8)$. We calculated eight derivatives using the general formula (2) for $n=8$ and the elements of the matrix $\boldsymbol{G}$ from Appendix E. Using these derivative values, we calculated the predicted trajectory using expression (9) for $x_0=0$. The predicted trajectory of motion (red curve) is

shown in Figure 9, the nine black dots are the points used to predict the trajectory and the black curve is the cosine curve plotted for comparison.

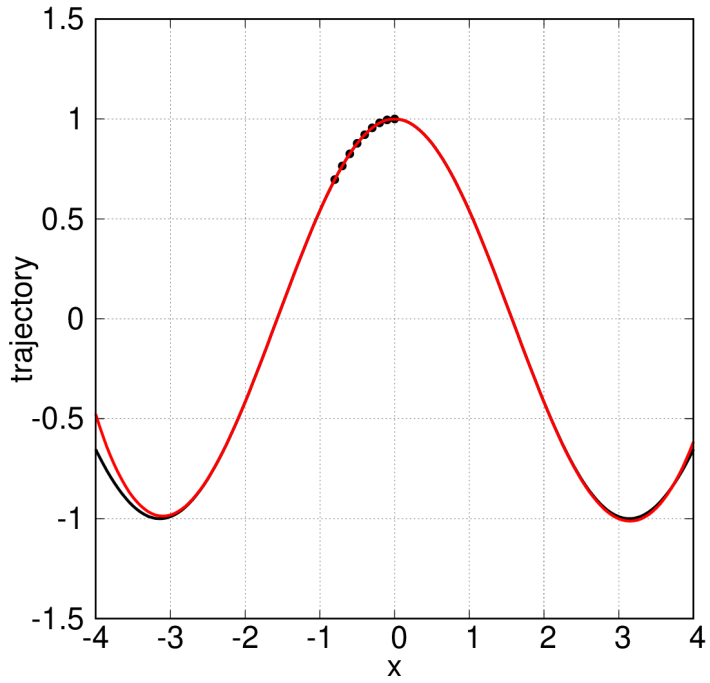

Figure 9

This Figure 9 clearly shows that if you know the trajectory of a missile in the $-0.8 \le x \le 0$ range, you can predict the trajectory of the missile with very good accuracy in the $-3 < x < 3$ range, or, in other words, if you know the trajectory of a missile a quarter of a circle, you can predict the trajectory of a missile half a circle ahead. This is the main result of our thought experiment.

Our study is limited to nine derivatives due to the fact that we use 64-bit floating point arithmetic, but if one use high-precision floating-point arithmetic when calculating the $\boldsymbol{G}$ matrix, one can calculate an unlimited number of derivatives. The calculations themselves according to expression (8) can be performed with normal precision, since matrix $\boldsymbol{G}$ is a well-defined matrix.

CONCLUSION

We have developed a methodology, computationally efficient differenceless derivatives with equidistant steps, which allows us to calculate an unlimited number of derivatives. Calculating derivatives comes down to performing simple arithmetic operations, multiplying a $n \times n$ matrix by a vector of function values of dimension n , the function values can be either from other calculations or from measurements. The new methodology can be used to solve various applied problems. We found that the predicted trajectory of the flying missile shows a very high prediction accuracy, which is difficult to achieve for other methods. The new method of calculating derivatives has only one drawback: it requires equidistant steps, which suggests the need for further research on differenceless derivatives with non-equidistant steps.

ABOUT COPYRIGHT

REFERENCES

1. Yuri Mahotin, Differenceless derivatives with equidistant steps, viXra:2401.0127, 2024

ABOUT AUTHOR

Yuri Mahotin. Currently on sabbatical due to health problems, unemployed. His current interests include the theory of dark matter and dark energy, as well as the theory and application of differenceless derivatives. Previously, he worked at Oracle Corp., Silvaco, Inc., Synopsys, Inc., and in Russia and the former USSR. He graduated from the Physics and Engineering Department of the Novosibirsk Electrotechnical Institute in 1984.

APPENDIX A

G[0][0] = +4.0000000000000027e+00;
G[0][1] = -3.0000000000000044e+00;
G[0][2] = +1.3333333333333361e+00;
G[0][3] = -2.5000000000000061e-01;
G[1][0] = -8.6666666666666750e+00;
G[1][1] = +9.5000000000000124e+00;
G[1][2] = -4.6666666666666741e+00;
G[1][3] = +9.1666666666666829e-01;
G[2][0] = +9.0000000000000089e+00;
G[2][1] = -1.2000000000000014e+01;
G[2][2] = +7.0000000000000098e+00;
G[2][3] = -1.5000000000000018e+00;
G[3][0] = -4.0000000000000044e+00;
G[3][1] = +6.0000000000000071e+00;
G[3][2] = -4.0000000000000044e+00;
G[3][3] = +1.0000000000000009e+00;

APPENDIX B

G[0][0] = +4.9999999999997167e+00;
G[0][1] = -4.9999999999995302e+00;
G[0][2] = +3.3333333333329325e+00;
G[0][3] = -1.2499999999998241e+00;
G[0][4] = +1.9999999999996843e-01;
G[1][0] = -1.2833333333332373e+01;
G[1][1] = +1.7833333333331737e+01;
G[1][2] = -1.2999999999998639e+01;
G[1][3] = +5.0833333333327353e+00;
G[1][4] = -8.3333333333322912e-01;
G[2][0] = +1.7749999999998384e+01;
G[2][1] = -2.9499999999997318e+01;
G[2][2] = +2.4499999999997712e+01;
G[2][3] = -1.0249999999998995e+01;
G[2][4] = +1.7499999999998241e+00;
G[3][0] = -1.3999999999998535e+01;
G[3][1] = +2.5999999999997566e+01;

G[3][2] = -2.3999999999997925e+01;
G[3][3] = +1.0999999999999087e+01;
G[3][4] = -1.9999999999998419e+00;
G[4][0] = +4.9999999999994191e+00;
G[4][1] = -9.9999999999990354e+00;
G[4][2] = +9.9999999999991775e+00;
G[4][3] = -4.9999999999996385e+00;
G[4][4] = +9.9999999999993605e-01;

APPENDIX C

G[0][0] = +5.9999999999994253e+00;
G[0][1] = -7.4999999999980353e+00;
G[0][2] = +6.6666666666632288e+00;
G[0][3] = -3.7499999999968678e+00;
G[0][4] = +1.1999999999985440e+00;
G[0][5] = -1.6666666666645646e-01;
G[1][0] = -1.7399999999997824e+01;
G[1][1] = +2.9249999999992419e+01;
G[1][2] = -2.8222222222208813e+01;
G[1][3] = +1.6499999999987708e+01;
G[1][4] = -5.3999999999942716e+00;
G[1][5] = +7.6111111111030993e-01;
G[2][0] = +2.8999999999995605e+01;
G[2][1] = -5.7624999999984475e+01;
G[2][2] = +6.1999999999972317e+01;
G[2][3] = -3.8374999999974513e+01;
G[2][4] = +1.2999999999988084e+01;
G[2][5] = -1.8749999999982663e+00;
G[3][0] = -3.0999999999994586e+01;
G[3][1] = +6.8499999999980659e+01;
G[3][2] = -8.0666666666631954e+01;
G[3][3] = +5.3499999999967926e+01;
G[3][4] = -1.8999999999984986e+01;
G[3][5] = +2.8333333333311259e+00;
G[4][0] = +1.9999999999996110e+01;
G[4][1] = -4.7499999999985974e+01;
G[4][2] = +5.9999999999974712e+01;
G[4][3] = -4.2499999999976566e+01;
G[4][4] = +1.5999999999989036e+01;
G[4][5] = -2.4999999999983942e+00;
G[5][0] = -5.9999999999987308e+00;
G[5][1] = +1.4999999999995394e+01;
G[5][2] = -1.9999999999991665e+01;
G[5][3] = +1.4999999999992262e+01;
G[5][4] = -5.9999999999963674e+00;
G[5][5] = +9.9999999999945643e-01;

APPENDIX D

G[0][0] = +6.9999999997593108e+00;

G[0][1] = -1.0499999999263041e+01;
G[0][2] = +1.1666666665445398e+01;
G[0][3] = -8.7499999988086063e+00;
G[0][4] = +4.1999999993124177e+00;
G[0][5] = -1.1666666664484353e+00;
G[0][6] = +1.4285714282726758e-01;
G[1][0] = -2.2299999998942848e+01;
G[1][1] = +4.3949999996758372e+01;
G[1][2] = -5.2722222216845026e+01;
G[1][3] = +4.0999999994750787e+01;
G[1][4] = -2.0099999996969011e+01;
G[1][5] = +5.6611111101488518e+00;
G[1][6] = -6.9999999986777084e-01;
G[2][0] = +4.2533333330774980e+01;
G[2][1] = -9.8224999992144021e+01;
G[2][2] = +1.2966666665362308e+02;
G[2][3] = -1.0604166665392506e+02;
G[2][4] = +5.3599999992639638e+01;
G[2][5] = -1.5408333330995220e+01;
G[2][6] = +1.9333333330131381e+00;
G[3][0] = -5.5499999995998351e+01;
G[3][1] = +1.4199999998769647e+02;
G[3][2] = -2.0316666664622147e+02;
G[3][3] = +1.7599999998001653e+02;
G[3][4] = -9.2499999988451265e+01;
G[3][5] = +2.7333333329663731e+01;
G[3][6] = -3.4999999994981863e+00;
G[4][0] = +4.9166666662585889e+01;
G[4][1] = -1.3499999998743922e+02;
G[4][2] = +2.0583333331244538e+02;
G[4][3] = -1.8833333331290675e+02;
G[4][4] = +1.0349999998819067e+02;
G[4][5] = -3.1666666662913713e+01;
G[4][6] = +4.1666666661519685e+00;
G[5][0] = -2.6999999997495880e+01;
G[5][1] = +7.7999999992284827e+01;
G[5][2] = -1.2499999998716196e+02;
G[5][3] = +1.1999999998744008e+02;
G[5][4] = -6.8999999992736377e+01;
G[5][5] = +2.1999999997691134e+01;
G[5][6] = -2.9999999996831548e+00;
G[6][0] = +6.9999999992905577e+00;
G[6][1] = -2.0999999997812420e+01;
G[6][2] = +3.4999999996357900e+01;
G[6][3] = -3.4999999996435484e+01;
G[6][4] = +2.0999999997938005e+01;
G[6][5] = -6.9999999993445101e+00;

G[6][6] = +9.9999999990967581e-01;

APPENDIX E

G[0][0] = +8.0000000134147236e+00;
G[0][1] = -1.4000000043652339e+01;
G[0][2] = +1.8666666748177008e+01;
G[0][3] = -1.7500000095553037e+01;
G[0][4] = +1.1200000072013843e+01;
G[0][5] = -4.6666667007329892e+00;
G[0][6] = +1.1428571520993387e+00;
G[0][7] = -1.2500000110276233e-01;
G[1][0] = -2.7485714350214927e+01;
G[1][1] = +6.2100000209853206e+01;
G[1][2] = -8.9022222614030497e+01;
G[1][3] = +8.6375000459282958e+01;
G[1][4] = -5.6400000346133041e+01;
G[1][5] = +2.3811111274852408e+01;
G[1][6] = -5.8857143301388533e+00;
G[1][7] = +6.4821429101857575e-01;
G[2][0] = +5.8166666843304903e+01;
G[2][1] = -1.5294166724125918e+02;
G[2][2] = +2.3910000107267780e+02;
G[2][3] = -2.4283333459066375e+02;
G[2][4] = +1.6303333428088612e+02;
G[2][5] = -7.0125000448249352e+01;
G[2][6] = +1.7566666788286511e+01;
G[2][7] = -1.9541666811837786e+00;
G[3][0] = -8.7733333659403556e+01;
G[3][1] = +2.5481666772717645e+02;
G[3][2] = -4.2880000197960248e+02;
G[3][3] = +4.5804166898690767e+02;
G[3][4] = -3.1813333508188475e+02;
G[3][5] = +1.4015000082717779e+02;
G[3][6] = -3.5733333557764126e+01;
G[3][7] = +4.0291666934572277e+00;
G[4][0] = +9.5833333751684123e+01;
G[4][1] = -2.9833333469377345e+02;
G[4][2] = +5.3250000253922508e+02;
G[4][3] = -5.9666666964266994e+02;
G[4][4] = +4.3016666890936358e+02;
G[4][5] = -1.9500000106094058e+02;
G[4][6] = +5.0833333621216298e+01;
G[4][7] = -5.8333333676746406e+00;
G[5][0] = -7.3000000362844659e+01;
G[5][1] = +2.3900000117978382e+02;
G[5][2] = -4.4700000220184893e+02;
G[5][3] = +5.2250000258047453e+02;
G[5][4] = -3.9100000194459636e+02;

G[5][5] = +1.8300000091992547e+02;
G[5][6] = -4.9000000249609002e+01;
G[5][7] = +5.7500000297986844e+00;
G[6][0] = +3.5000000193388480e+01;
G[6][1] = -1.1900000062872745e+02;
G[6][2] = +2.3100000117331717e+02;
G[6][3] = -2.8000000137502246e+02;
G[6][4] = +2.1700000103617563e+02;
G[6][5] = -1.0500000049017922e+02;
G[6][6] = +2.9000000133013600e+01;
G[6][7] = -3.5000000158706825e+00;
G[7][0] = -8.0000000482245532e+00;
G[7][1] = +2.8000000156767751e+01;
G[7][2] = -5.6000000292537699e+01;
G[7][3] = +7.0000000342815810e+01;
G[7][4] = -5.6000000258332605e+01;
G[7][5] = +2.8000000122208689e+01;
G[7][6] = -8.0000000331629053e+00;
G[7][7] = +1.0000000039563020e+00;