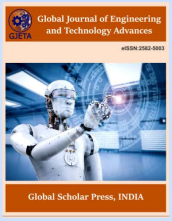
Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/



(Research Article)

Check for updates

Automating Symbolic Partial Differentiation on Multivariate Polynomials

Henry Samambgwa * and Thomas Musora

Department of Mathematics and Statistics, School of Natural Sciences and Mathematics, Chinhoyi University of Technology, 78 Magamba Way, Off Chirundu Road, Chinhoyi, Zimbabwe.

Global Journal of Engineering and Technology Advances, 2025, 25(03), 286-292

Publication history: Received on 12 November 2025; revised on 29 December 2025; accepted on 31 December 2025

Article DOI: https://doi.org/10.30574/gjeta.2025.25.3.0363

Abstract

This study develops a novel implementation for symbolic partial differentiation of multivariate polynomials in MATLAB. Pseudocode, a flowchart and MATLAB source code are developed. The program is then tested for samples of multivariate polynomials and partial differentiation operations. It is demonstrated that the program generates results with 100% accuracy. Applications of the developed program are unlimited, with potential uses in economic, engineering and scientific neural network and computer-based model and systems.

Keywords: Symbolic Partial Differentiation; Multivariate Polynomial; MATLAB Source Code; Algorithm

1. Introduction

Three differentiation techniques are commonly used in machine learning and artificial intelligence research; algorithmic differentiation, symbolic differentiation and numerical differentiation [1]. Algorithmic differentiation is the technique of generating a derivative program for a given function program [2]. Different programs can be used for estimation at various input ranges [3]. Variations of algorithmic differentiation include derivation from first principles [4]. Algorithmic differentiation is widely preferred in gradient based optimisation programs [5].

Numerical differential approximations are the last resort when automatic differentiation and symbolic differential fail for lack of known programs or analytic functions [6]. Neural network models depend on empirical data [7] and their outputs are more suited for numerical differentiation than symbolic methods. Samambgwa et al [8] developed a MATLAB implementation for isolating multivariate terms from neural network models, which promises potential in the future of extracting analytic models from neural networks. Numerical differentiation is limited in that it produces approximations at best and requires numerous iterations thereby burdening computational resources [9]. MATLAB is a favourable platform for executing numerous sequential mathematical operations to yield accurate results [10].

Symbolic differentiation has advantage when machine learning is used with agent-based models [11], as agents can be represented analytically [12]. The development of symbolic analysis algorithms is crucial for the achievement of artificial general intelligence [13].

Integral and differential models and analysis techniques remain relevant in artificial intelligence and neural network research. Polynomial integrals occur in the study of physics informed neural networks [14], where solutions can be inferred through partial differentiation. Differential equations are critical in analysis of dynamic systems as they capture the underlying controlling influences [15]. Differentiation is critical in optimising machine learning models and gradient based algorithms [16].

* Corresponding author: Henry Samambgwa.

Various scholars have contributed significant developments in the research area. Thota [17] developed an algorithm that simplifies differential algebraic equations into ordinary differential equations, with implementations for the MATLAB environment. Liu and Liu [18] used partial derivative operators in closure spaces to obtain analytic partial derivatives for systems of homogenous multivariate polynomial equations. Halim and Yeak [16] developed algorithms and python implementations for automatic differentiation yielding numerical derivatives. Kourounis et al, [19] developed a C++ implementation for partial differentiation in univariate functions. Brahmam et al [20] developed a Mathematica implementation for symbolic differentiation of exponential functions.

Application of polynomial differentiation techniques can be extended to a wide range of systems of equations since most can be transformed to polynomial form [20].

Aim

This study sought to develop an algorithm and MATLAB implementation that accurately executes symbolic partial differentiation on multivariate polynomials.

Objectives

- Develop pseudocode that executes symbolic partial differentiation on multivariate polynomials,
- Develop the flow chart for symbolic partial differentiation of a multivariate polynomial,
- Develop MATLAB source code that executes symbolic partial differentiation on multivariate polynomials.

2. Methodology

2.1. Derivative function

The derivative of a function f, in x, with respect to x, denoted $f'(x)$, is a function that outputs the gradient of the function $f(x)$ at any point x.

2.2. Symbolic polynomial differentiation

Differentiation is the process of obtaining a derivative from its function. The steps to execute symbolic differentiation are outlined:

Suppose f, a function of x, denoted $f(x)$, is a polynomial term in x, written:

$$f(x) = kx^a, \quad (1)$$

Where:

k is the coefficient, and

a is the power. (a is also the degree of the polynomial)

The derivative of f with respect to x, denoted $f'(x)$, is obtained by multiplying the coefficient, k, by the power, a, and then reducing the power by 1. That is:

$$f'(x) = akx^{a-1}. \quad (2)$$

2.3. Polynomial partial differentiation

For purposes of illustration, we will consider the case of a polynomial with two variables. The principles can be extended and apply to an arbitrary number of variables. The general term in a bivariate polynomial in x and y is of the form:

$$kx^a y^b$$

Where:

k is the coefficient,

a is the power of x, and

b is the power of y.

The partial derivative of f, with respect to x, denoted, f_x, where f is a function of x and y, denoted, $f(x, y)$, is obtained by multiplying the constant, k, by the power of x, a, and then reducing the power of x by 1. That is:

If $$f(x, y) = kx^a y^b, \tag{3}$$

Then $$f_x = akx^{a-1} y^b, \tag{4}$$

And, similarly, the partial derivative of $f(x, y)$ with respect to y, denoted f_y, is given by:

$$f_y = bkx^a y^{b-1}. \tag{5}$$

The principle is extended and applicable to an arbitrary number of variables.

2.4. Higher order partial derivatives

Higher order derivatives are obtained by repeating the procedures in equations (4) and (5), on the prevailing coefficients and powers at each step. The second partial derivatives w.r.t. x and y are denoted f_{xx} and f_{yy} respectively. Other higher order derivatives are denoted f_{xy}, f_{xxy}, f_{xxyy}, and so on. The number of appearances of a variable x or y in the subscript indicates the number of times f is differentiated partially w.r.t. that variable.

2.5. Symbolic differentiation algorithm

In order to execute partial differentiation in code, there is need to represent coefficients and powers in a programing variable and data type that is accommodated by the programming environment. We use an array with three rows to represent polynomials, where the rows represent the coefficient, power of x, and the power of y.

For example, the polynomial $3 + x - 2y - 4x^2y^5 + 2x^3y + 5x^3y^3$ is represented by the array:

$$\begin{bmatrix} 3 & 1 & -2 & -4 & 2 & 5 \\ 0 & 1 & 0 & 2 & 3 & 3 \\ 0 & 0 & 1 & 5 & 1 & 3 \end{bmatrix}.$$

Partial differential operations are represented as 2-dimensional vectors, the values indicating the number of times to differentiate partially w.r.t. x and y respectively.

For example:

$f_x \rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $f_y \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $f_{xxy} \rightarrow \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $f_{xxyyyyy} \rightarrow \begin{bmatrix} 2 \\ 5 \end{bmatrix}$, and so on . . .

2.6. Pseudo code

In the case of a bivariate polynomial, the algorithm takes input in the form of a 3-row array (representing the polynomial) and a 2-dimensional vector (representing the partial differentiation operator). In general, for an arbitrary number of variables, n, the input array will have $n + 1$ rows and the differential vector will be n-dimensional. Execution then follows the steps outlined in the pseudocode in **Figure 1.**

Step1: Read the size of the input array (number of rows and columns)

Step 2: Set numrows = number of rows

Step 3: Set numcols = number of columns

Step 4: Read differentiation vector

Step 5: Repeat sub-step 1 and 2 for row = 2 to to row = numrows

Sub-step 1: Set d = element row-1 in differential vector

Sub-step 2: Repeat sub-sub-steps 1 and 2 as many times as d

Sub-sub-step 1: multiply row 1 by current row

Sub-sub-step 2: subtract 1 from each number in current row

Figure 1 Pseudocode for symbolic partial differentiation algorithm

Flowchart

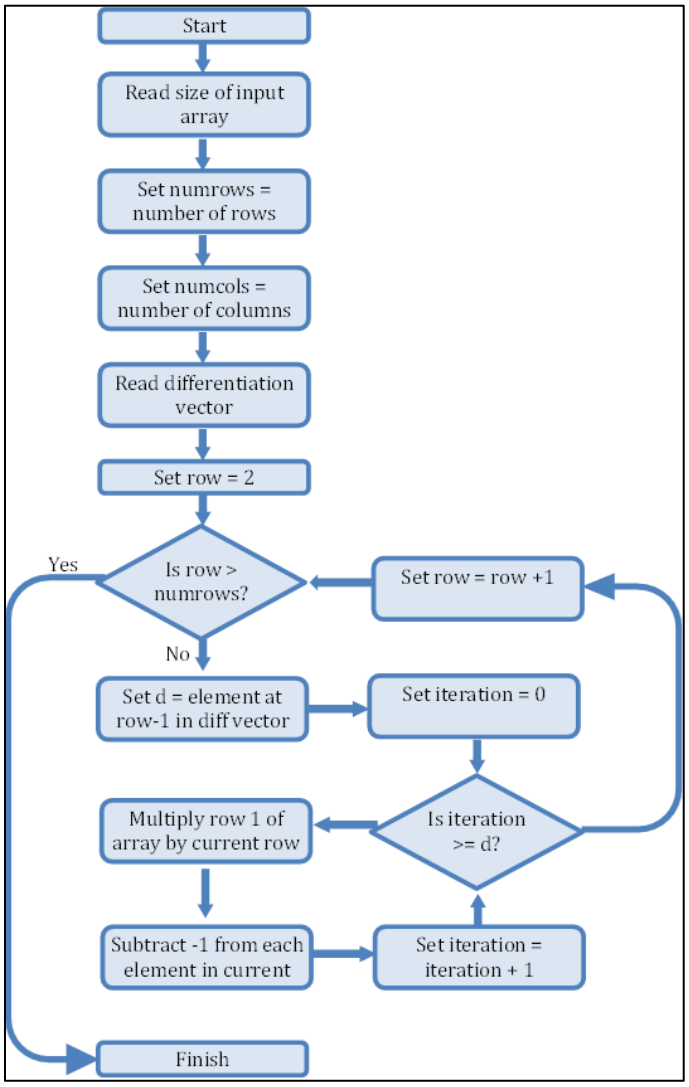


Figure 2 Flowchart for symbolic partial differentiation algorithm.

The algorithm has two main loops. The minor loop differentiates a row (representing indices of a particular variable) partially as many times as indicated by the corresponding element (indicating how many times to differentiate w.r.t. its corresponding variable) in the differential matrix. The major loop selects through the rows (each row representing indices of a particular variable) from the second row (representing the first variable) to the last row (representing the last variable).

2.7. MATLAB programming source code

The code snippet in **Figure 3** was used to execute the symbolic partial differentiation algorithm in the MATLAB online [22] environment.

```
%User sets input array (matrix) M
%User sets differential vector D
s = size(M);        %read dimensions of M
numrows = s(1); %set number of rows to first element of size vector s
numcols = s(2);  %set number of columns to first element of size vector s
row = 2; %set row = 2

while row <= numrows   %loop until row > numrows
        d = D(row-1);              %set d = element at row - 1 in D
        iteration = 0;             %set iteration = 0
        while iteration < d        %loop until iteration >= d
                col = 1;           % set col = 1
                while col <= numcols              %loop until col > numcols
                        M(1,col) = M(1,col) * M(row,col);          % multiply first row by current row
                        M(row,col) = M(row,col) - 1;               %subtract 1 from each element in current row
                        col = col + 1;                             %increment column by 1
                end
                iteration = iteration + 1;                 %increment iteration by 1
        end
        row = row + 1;                                                  %increment row by 1
end
M       %display output matrix
```

Figure 3 MATLAB source code that executes symbolic partial differentiation

3. Analysis

Three multivariate polynomials were used to test the symbolic partial differentiation implementation in MATLAB. For each polynomial, three partial differential operators were executed. The polynomials, partial derivative operators, matrix representations of the polynomial, vector representations of the partial derivative operators and the resulting matrix representations of the partial derivatives as well as the symbolic partial derivatives are shown in **Table 1**.

Table 1 MATLAB inputs and outputs of symbolic partial differentiation operations.

Function	Partial derivative operator	Matrix (array) representation of function	Vector representation of partial derivative operator	Matrix (array) representation of partial derivative	Symbolic form of partial derivative
$2r + 5t - 6rt + 3r^2t^2$	f_r	$\begin{bmatrix} 2 & 5 & -6 & 3 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & -6 & 6 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$	$2 - 6t + 6rt^2$

$2r+5t$ $-6rt$ $+3r^2t^2$	f_t	$\begin{bmatrix}2&5&-6&3\\1&0&1&2\\0&1&1&2\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0&5&-6&6\\1&0&1&2\\-1&0&0&1\end{bmatrix}$	$5-6r+6r^2t$
$2r+5t$ $-6rt$ $+3r^2t^2$	f_{rtt}	$\begin{bmatrix}2&5&-6&3\\1&0&1&2\\0&1&1&2\end{bmatrix}$	$\begin{bmatrix}1\\2\end{bmatrix}$	$\begin{bmatrix}0&0&0&12\\0&-1&0&1\\-2&1&-1&0\end{bmatrix}$	$12r$
$4a^4b^2c^5$ $-3a^2b^3c$	f_{abbc}	$\begin{bmatrix}4&-3\\4&2\\2&3\\5&1\end{bmatrix}$	$\begin{bmatrix}1\\2\\1\end{bmatrix}$	$\begin{bmatrix}160&-36\\3&1\\0&1\\4&0\end{bmatrix}$	$160a^3c^4$ $-36ab$
$4a^4b^2c^5$ $-3a^2b^3c$	f_{abbb}	$\begin{bmatrix}4&-3\\4&2\\2&3\\5&1\end{bmatrix}$	$\begin{bmatrix}1\\3\\0\end{bmatrix}$	$\begin{bmatrix}0&-36\\3&1\\-1&0\\5&1\end{bmatrix}$	$-36ac$
$4a^4b^2c^5$ $-3a^2b^3c$	f_{aaacc}	$\begin{bmatrix}4&-3\\4&2\\2&3\\5&1\end{bmatrix}$	$\begin{bmatrix}3\\0\\2\end{bmatrix}$	$\begin{bmatrix}1920&0\\1&-1\\2&3\\3&-1\end{bmatrix}$	$1920ab^2c^3$
$3xy^3$ $-2x^2y$ $+x^5$	f_{xxy}	$\begin{bmatrix}3&-2&1\\1&2&5\\3&1&0\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$	$\begin{bmatrix}0&-4&0\\-1&0&3\\2&0&-1\end{bmatrix}$	-4
$3xy^3$ $-2x^2y$ $+x^5$	f_{xxxy}	$\begin{bmatrix}3&-2&1\\1&2&5\\3&1&0\end{bmatrix}$	$\begin{bmatrix}3\\1\end{bmatrix}$	$\begin{bmatrix}0&0&0\\-2&-1&2\\2&0&-1\end{bmatrix}$	0
$3xy^3$ $-2x^2y$ $+x^5$	f_{xyy}	$\begin{bmatrix}3&-2&1\\1&2&5\\3&1&0\end{bmatrix}$	$\begin{bmatrix}1\\2\end{bmatrix}$	$\begin{bmatrix}18&0&0\\0&1&4\\1&-1&-2\end{bmatrix}$	$18y$

4. Discussion

All nine tested symbolic partial differentiation operations yielded 100% accurate results. It was demonstrated that the developed symbolic partial differentiation algorithm and MATLAB source code were fully functional and highly reliable. The program can be used to perform any symbolic partial differentiation operation on multivariate polynomials with an arbitrary number of variables. This program can be used in analysis of large systems of polynomials thereby introduces high-capacity capabilities for research and applications.

Automating symbolic partial differentiation extends the accuracy and capabilities of neural network and other computer-based models for analysing multivariate polynomial problems in economics, engineering, science and various other research studies.

5. Conclusion

This study successfully developed and validated implementations for symbolic partial differentiation on multivariate polynomials. This opens up a wide range of potential applications across disciplines in research and applications. Computer based systems can be built with the derived capabilities for improved accuracy and capacity. Further research work may be pursued to develop programs to execute symbolic partial differentiation for non-polynomial functions such as exponential, logarithmic and trigonometric functions.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Wang, L., Zhao, J. (2023). Architecture of Advanced Numerical Analysis Systems: Designing a Scientific Computing System using OCaml. Architecture of Advanced Numerical Analysis Systems.

[2] Baydin, A. G., Pearlmutter, B. A., Radul, A. A. Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research.

[3] Scibior, A., Wood, F. (2021). Differentiable particle filtering without modifying the forward pass. arXiv. Cornell University.

[4] Krieken, E., Tomczak, J. M., Teije, A. T. (2021). Storchastic: A framework for general stochastic automatic differentiation. Advances in Neural Information Processing Systems.

[5] Corliss, G., Faure, C., Griewank, A., Hascoet, L., Naumann, U. (2002). Automatic differentiation of algorithms: from simulation to optimization. Springer Science & Business Media.

[6] Lorberbom, G., Maddison, C. J., Heess, N., Hazan, T., Tarlow, D. (2020). Direct policy gradients: Direct optimization of policies in discrete action spaces. Advances in Neural Information Processing Systems. Curran Associates, Inc.

[7] Musora, T. (2023). Application of Panel Data Analysis and Modelling to Economic Data: A Case of Determinants of Economic Growth for SADC and Zimbabwe. Chinhoyi University of Technology.

[8] Samambgwa, H., Musora, T., Kamusha, J. (2025). Deriving mathematical models from neural networks: A method for deducing individual effects of factors on a response variable. World Journal for Advanced Engineering Technology and Sciences.

[9] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P. (2007). Numerical recipes 3rd edition: The art of scientific computing. Cambridge University Press.

[10] Samambgwa, H., Musora, T. (2025). Automating upper triangular matrix neutralisation for minimal error. World Journal of Advanced Engineering Technology and Sciences.

[11] Chopra, A., Rodríguez, A., Subramanian, J., Krishnamurthy, B., Prakash, B. A., Raskar, R. (2022). Differentiable agent-based epidemiology. arXiv. Cornell University.

[12] Silverman, E., Gostoli, U., Picascia, S., Almagor, J.,McCann, M., Shaw, R., Angione, C. (2021). Situating agent-based modelling in population health research. Emerging Themes in Epidemiology.

[13] Hans, J. (2025). A Hybrid Cognitive Architecture for AGI: Bridging Symbolic and Subsymbolic AI. International Journal for Multidisciplinary Research.

[14] Mazraeh, H. D., Parand, K. (2025). An innovative combination of deep Q-networks and context-free grammars for symbolic solutions to differential equations. Elsevier.

[15] Yang, J., Nhat, M., Hu, B., Cao, Y., Dehmamy, N., Walters, R., Yu, R. (2025). Discovering Symbolic Differential Equations with Symmetry Invariants. arXiv. Cornell University.

[16] Halim, M. A. S., Yeak H. S. (2024). Introduction to Automatic Differentiation and Neural Differentiation Equation. Proceedings of Science and Mathematics. Universiti Teknologi Malaysia.

[17] Thota, S. (2020). On Solving System of Linear Differential-Algebraic Equations Using Reduction Algorithm. International Journal of Mathematics and Mathematical Sciences.

[18] Liu, D., Liu, S. (2025). Unified Solving of Multivariate Polynomial Systems via Hierarchical Differential Algebraic Closure: A Foundation for Symbolic-Numeric Quantum Computation. Cambridge University Press.

[19] Kourounis, D., Gergidis, L., Saunders, M., Walther, A., Schenk, O. (2017). Compile-Time Symbolic Differentiation Using C++ Expression Templates. arXiv. Cornell University.

[20] Brahmam, O., Navaj, C., Modi, R., Gupta, Y. (2025). Recursive Power-Exponential Functions and Their Symbolic Differentiation: A New Framework for Modeling Hypergrowth in AI and Complexity Theory. International Journal of Research and Analytic Reviews.

[21] Kramer, B., Pogudin, G. (2025). Discovering Polynomial and Quadratic Structure in Nonlinear Ordinary Differential Equations. arXiv. Cornell University.

[22] MATLAB online. (2025). Accessed at: matlab.mathworks.com. Mathworks.