Paweł Plaskura
Dr. Eng.
Faculty of Social Sciences, Jan Kochanowski University, Branch in Piotrków Trybunalski,
Piotrków Trybunalski, Poland
ORCID ID: 0000-0002-8257-781X
p.plaskura@unipt.pl

DERIVWWW - WEB-BASED SYMBOLIC DIFFERENTIATION SYSTEM

Abstract. The article presents an online system for symbolic differentiation. It shows how derivatives are calculated. The trees are used for internal representation of formulas. Derivatives are generated by tree transformations. Presented algorithms are part of the microsystems simulator Dero. They are required by the calculation algorithms such as Newton-Raphson. They can be used to generate derivatives in model description languages and for automatic derivative calculation. DerivWWW can be used in didactics. The system can serve as a tool for students to count function derivatives at the point. It can also be used in teaching on the technical fields of study. Symbolic derivatives are saved in the Tex format, allowing easy integration with other software. The developed and implemented algorithms are discussed. Examples of use are given.

Keywords: symbolic differentiation; web based systems; learning tools; ICT in Education

1. INTRODUCTION

Simulators of analog electronic circuits [2, 14] have become an essential tool for designers of electronic circuits and since the 90s of the last century also become a tool of education. They are widely used in universities and more often in schools. They enable the acceleration of the process of education and significant improvement of the quality of education. The use of computer technology greatly reduces expenditures on laboratory equipment. It allows the realization of the so-called virtual laboratories. It opens the way for quick and easy self-education, and distance education.

Very rapid development of simulators in the 70s of the last century led to the development of a number of advanced computing techniques [7, 5, 4, 15, 13]. In other areas, progress was not as fast. More common problem was the simulation of systems with mixed type of signals from different environments (since the early 90s of the twentieth century). Advanced simulation techniques in the field of electronics were used in other fields. Methods and tools that enable simulation of microsystems [1, 18] (e.g. the electro-mechanical systems) were also developed. The basis of this type of system simulation is a behavioural description of the system [3] in specially designed languages, such as: EMDL [6], MDL [12], VHDL-AMS [9], Verilog [17]. They provide a description of the simulated system using formulas (mathematical equations). Models of elements described in such languages can be used by simulators. The main problem in behavioural modelling is to calculate partial derivative of function. They are required by the calculation algorithms. Derivatives can be given explicitly. In the case of more complex functions it is very cumbersome (the most common procedure followed in the past). Derivatives can be also calculated numerically but it slows down the process of simulation. The best solution is the automatic generation of derivatives in the symbolic form. The symbolic form can be compiled into model code. It can be done during reading of the model.

The aim of the article is to present the developed system of symbolic differentiation and its possible practical application in didactics.

2. THETHEORETICAL BACKGROUNDS

Automatic generation of derivatives in the symbolic form requires representation of the formula in the form of tree [19]. The tree can be created during input parsing. The symbolic representation of the derivative can be calculated by transforming the tree. The presented algorithm is a simplified version of the algorithm developed for the Dero [12] simulator.

The tree shown has one root (Figure 1). It allows representation of the elementary operations.

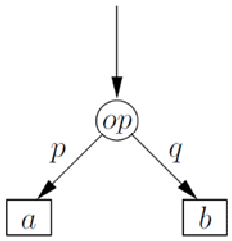


Figure 1: Tree

The elementary arithmetic operations are placed in the nodes. They are divided into two groups:

- basic operations: addition, subtraction, multiplication, division, exponentiation;
- function calls.

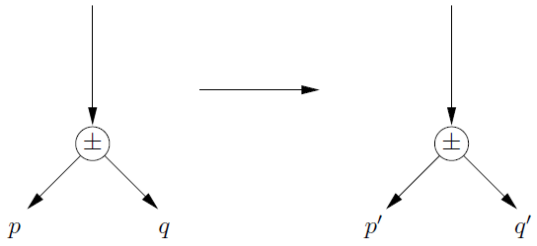
Variables (a,b) or constant values are located in the leaves of the tree. The branches p and q indicate the nodes or leaves. The tree is created during formula parsing by the parser module. The parser uses lexical analyzer (lex [8]) collaborating with yacc [8]. The formula is represented in the computer memory in the form of a tree. The leaves should keep the variables/constants while the elementary operations are placed in the vertices.

2.1. Differentiation algorithm

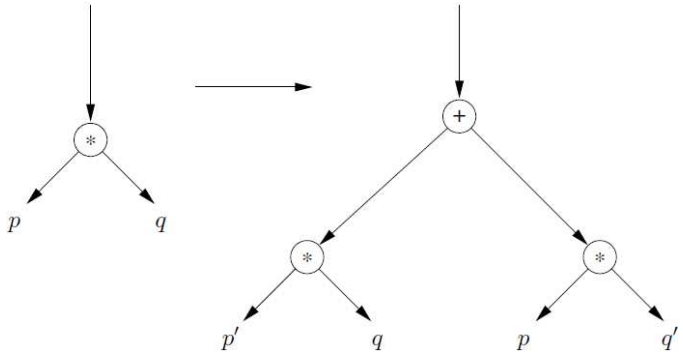
The algorithm transforms the tree to determine the derivative relative to the selected variable placed in the leaf. Transformation begins from the root and ends on leaves. The conversion is performed on the vertices of the tree. The transformed tree may have new branches. Converted tree gives ability to generate symbolic representation of the data stored in the tree. Representation of the basic mathematical operations and their derivatives is shown in Figure 2. Symbol ' denotes the derivative. For example, the derivatives of basic expressions are shown below:

$$\begin{aligned} (p \pm q)' &= p' \pm q' \\ (pq)' &= p'q + pq' \\ (p/q)' &= \frac{p'q - pq'}{q^2} \end{aligned}$$

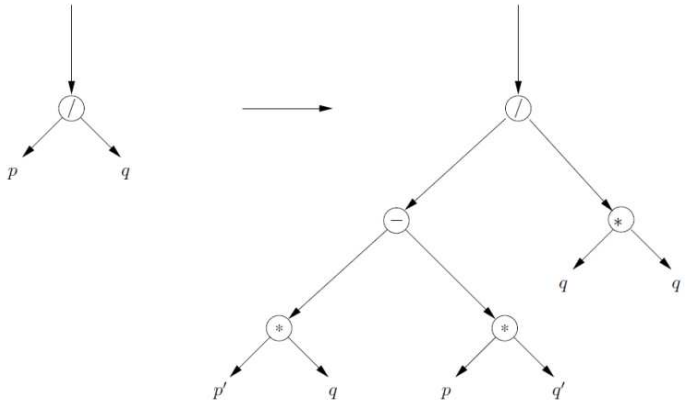
The above shown transformations are shown in Algorithm 1. The algorithm starts at the root. The first node of the tree w1 (first layer) is converted, and then the lower level of nodes (lower layer) is converted. The algorithm terminates when the leaves are achieved.



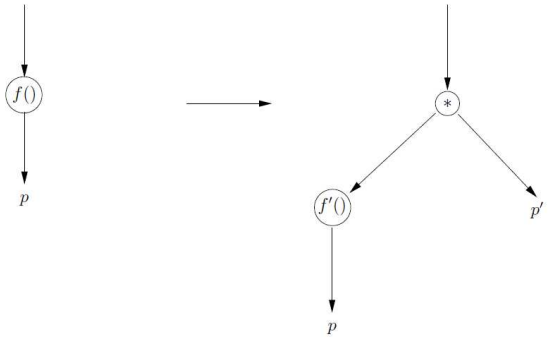
(a) addition and subtraction (p ± q)′ = p′ ± q′



(b) multiplication (pq)′ = p′q + pq′



(c) division (p/q)′ = p′q-pq′



*(d) function f(p)′ = f(p)′ * p′*

Figure 2: Transformations of tree nodes for basic arithmetic operations

Algorithm 1 Basic transformations of the tree.

```
i - layer index
j - index in the layer
n_i - the number of vertices in the layer
d - number of layers
for i = d...1 do
    for j = 1...n_i do
        transform the node w_j in the layer i
    end for
end for
```

Transformed tree describes the derivative with respect to all variables (tree leaves). In order to generate the derivative with respect to one variable (e.g. p) the tree must be reduced. Reduction of the tree removes the branches which are equals to 0. Let us consider the derivative

$$(pq)' = p'q + pq'$$

Its representation is shown in Figure 2b. If the derivative is calculated with respect to p, then *pq'* equals 0, because *q* is constant. The branch *pq'* can be reduced and it can be replaced with the leaf with the 0 value (Figure 3). The process described above is realized by Algorithm 2.

Algorithm 2 Reduction of tree branch.

```
i - index layer, j - index in the layer
n_i - the number of vertices in the layer
d - number of layers
for i = d...1 do
    for j = 1...n_i do
        if w_j ==' *' then
            if w_j → p == 0 w_j → q == 0 then
                eliminate node w_j
            end if
        end if
        if w_j ==' /' then
            if w_j → q ==' 0' then
                error
            end if
        end if
        if w_j → p ==' 0' then
            eliminate branch p
        end if
        if w_j → q ==' 0' then
            eliminate branch q
        end if
    end for
end for
```

The algorithm takes into account all nodes from the lowest level, i.e. vertices located close to the leaf. After reduction of all vertices of the lowest layer it goes to the higher level. The algorithm terminates when reaches root of the tree. After that the tree contains the calculated derivative with respect to the selected variable. Variables and constants are stored in the leaves of the tree. The symbolic form of derivative can be automatically generated from the tree. Generation of the symbolic form of derivative requires swap of the branches (algorithm 3). Example of such conversion is shown in Figure 4.

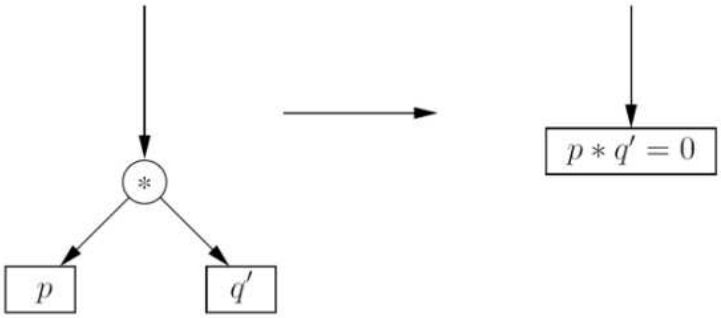


Figure 3: Reduction of the tree branch for the multiplication operator

Algorithm 3 Generation of derivative in the symbolic form.

i - index layer
j - index in the layer
n_i - the number of vertices in the layer
d - number of layers
for $i = d \dots 1$ **do**
  for $j = 1 \dots n_i$ **do**
    if $w_j == ' + - */'$ **then**
      replace the leaf w_j by the node l_j
      remember action p op q
      $l_j \leftarrow q_{w_p} + w_j + q_{w_q}$
    else
      replace the leaf w_j by the node l_j
      remember action p op q
      $l_j \leftarrow f(p_{w_j})$
    end if
  end for
end for

The algorithm starts with the lowest level conversion of vertices. The vertex is replaced by the leaf, which stores operation performed on the branches *p* and *q*. The algorithm terminates when it reaches the root of the tree. The tree is reduced to a single leaf. Therefore the leaf contains symbolic form of derivative.

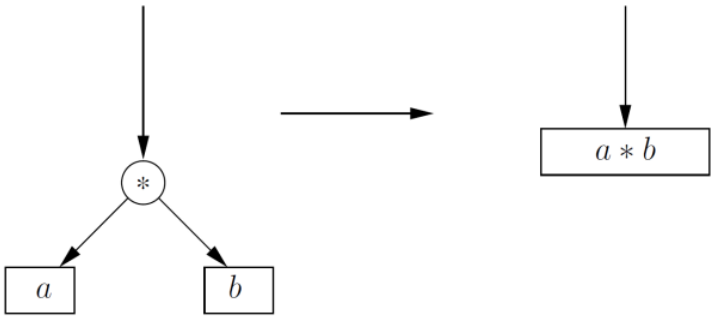


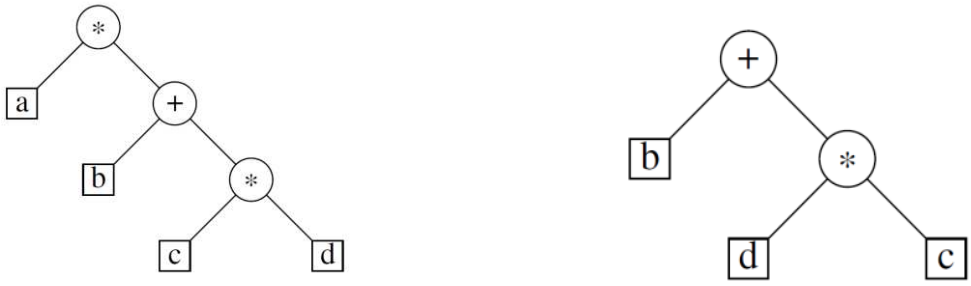
Figure 4: Generation of symbolic form for multiplication

2.2. Example

Let us consider the arithmetic expression (1) and its tree shown in Figure 5a.

$$a(b + dc) \tag{1}$$

The derivative of (1) with respect to *a* is shown in Figure 5b.



(a) parsing tree *(b) derivative with respect to a*

Figure 5: Representation of expression (1) and its derivatives

3. THE RESULTS AND DISCUSSION

The above algorithm has been implemented in practice in the simulator Dero [12, 13] in more extended form. It has also been implemented as a separate package Deriv [11]. Deriv is available as a separate package (*.deb) for Linux (Debian) operating system. The program is written in C++. Its use is restricted by RSA key [16]. The basic arithmetic operators (e.g.: +,-, /, *) as well as many built-in functions are implemented, e.g.:

abs(x) - *absolute value,*
floor(x) - *the largest integer value not greater than x,*
one(x) - *number one,*
sqrt(x) - *square root,*
sin(x) - *sine,*
asin(x) - *arc sine,*
sinh(x) - *hyperbolic sine,*
cos(x) - *cosine*
acos(x) - *arc cosine,*
cosh(x) - *hyperbolic cosine,*
exp(x) - *exponential function,*
expo(x) - *exponential function with limits,*
log(x) - *natural logarithm,*
log10(x) - *logarithm,*
min(x, y) - *the smallest value of x or y,*
max(x, y) - *the largest value of x or y,*
tan(x) - *tangent,*
atan(x) - *arc tangent,*
tanh(x) - *hyperbolic tangent.*

Documentation and examples are available in English and Polish. The program does not have a graphical interface. It has to be run from the command line:

deriv -c certyficate.crt -i mytest.txt

where: *certyficate.crt* is the file containing RSA key certificate, *mytest.txt* is a sample input file (listing 1). The syntax of the input file is described in the documentation of Deriv.

Listing 1: A sample input file.

```
.OPTI DFDXVAL=1
.OPTI FDER=1
.OPTI DEBU_LEX=0
.FUNCTION INPUT(REAL x,REAL y)
.LOCAL REAL xx, REAL aa;
.BEGIN
y=1;
x=0.5;
PRINT("INPUT VARS");
PRINT("x =", x+2 * x );
PRINT("y =",y);
PRINT("PARTIAL DERIVATIVES");
DFDX ( x+2*x+1/y, y);
DFDX ( x^2, y);
DFDX ( x^2, x);
DFDX ( (x+1)^^^(x+2), x);
DFDX ( 1/x, x);
DFDX ( sin(x), x);
DFDX ( cos(x), x);
DFDX ( tan(x), x);
DFDX ( tanh(x+y), x);
DFDX ( atan(x+y), x);
DFDX ( tan(x+y), x);
DFDX ( expo(x+y), x);
DFDX ( exp(x+y), x);
DFDX ( x/y, x);
DFDX ( sqrt(1), x);
DFDX ( abs(x), x);
PRINT("------------------");
DFDX ( asin(x), x);
PRINT("------------------");
DFDX ( acos(x), x);
PRINT("------------------");
.END
```

3.1. DerivWWW - Web interface for Deriv

DerivWWW was created as an interface to Deriv - http://deriv.aiva.pl. Documentation and examples have been included. The calculated derivatives can be simplified by using the right option. The client-server architecture was used. DerivWWW was designed and written in PHP5 [10]. It requires the apache2 web server. The derivatives are generated in the text format, but they are displayed in the graphic format *.png because the system is dedicated for humans, not robots.

LaTeX was used to generate the graphic file representing the derivative as well as the graphical representation of the unique text which has to be recognized by human on the first use.

3.2. The algorithm implemented in DerivWWWDeriv

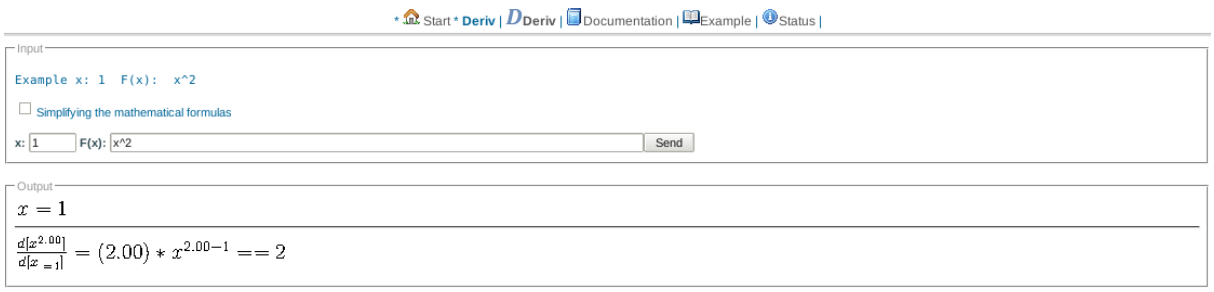
DerivWWW works in the client-server architecture (Algorithm 4). The client is a web browser. The data from the form are sent to the server. The server sends back the results or generates error messages in the case of errors.

Algorithm 4 The algorithm of the system DerivWWW.

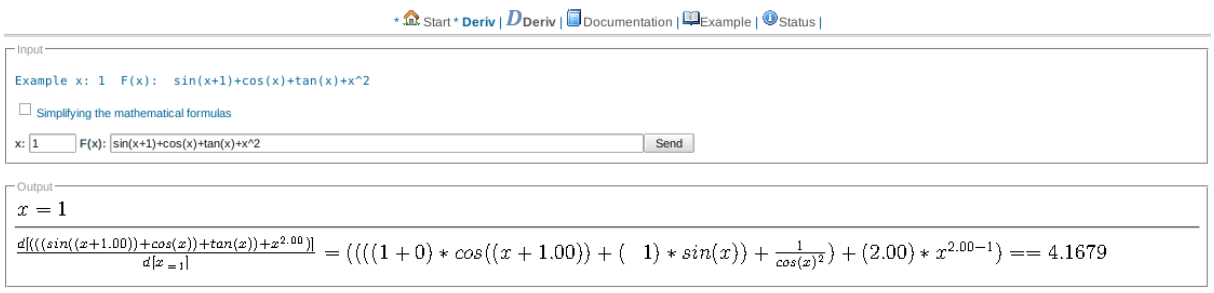
```
enter validation code - check whether human                ▷ on the client side
Require: input formula
send data to the server                                    ▷ on the server side
receive data
check authentication
validate data
save data in the file
run deriv
check execution errors
if errors occurs then
    send error message
else
    generate file for LaTeX describing the derivative
    compile the LaTeX file
    convert the LaTeX output file to the graphic format
    send the data back to the client
end ifEnd
```

3.3. Examples

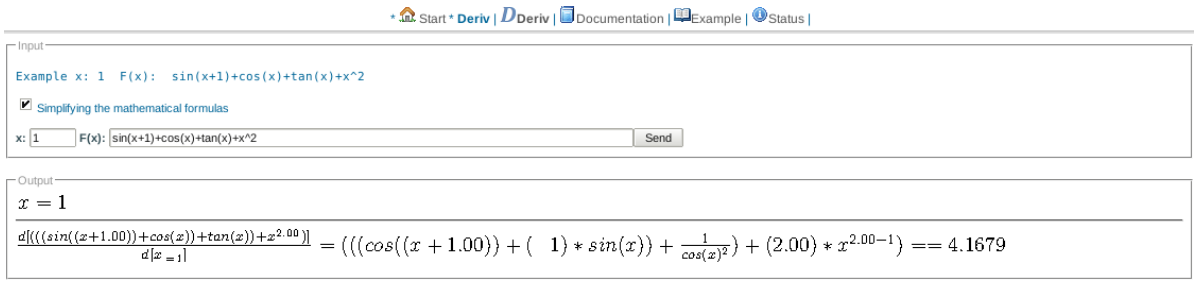
Examples of using DerivWWW are shown in Figure 6. The derivative in Figure 6c has been simplified.



(a) f(x) = x2



(b) f(x) = sin(x + 1) + cos(x) + tan(x) + x2



(c) f(x) = sin(x + 1) + cos(x) + tan(x) + x2 - the simplification

Figure 6: Examples

4. CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

The study shows the algorithm of symbolic derivatives generation by transformation of a tree. The derivative with respect to the variable is generated by the tree reduction. The algorithm has been practically implemented in the Deriv. Deriv allows the calculation of the derivative at a point. In practical applications the presented algorithm may be inefficient. The efficiency may be improved by storing additional data describing derivatives with respect to the input variables. The modified and more complex algorithm has been implemented in the simulator Dero [12].

The web based interface was created (DerivWWW). The system can be used in the teaching process for such courses as:

– numerical methods,
– programming,
– simulation systems.

DerivWWW was already used by students in the didactical process. At the moment the further development of the Deriv is not planned as it is a part of the Dero [12]. DerivWWW can easily be modified to commit individual requirements. The input language makes it easy to integrate with other systems.

REFERENCES

[1] Stephen E.Senturia, "Simulation and design of microsystems: a 10-year perspective", *Sensors and Actuators,* A67:1–7, 1998.(in English).
[2] Marc. E. Herniter, "Schematic Capture with MicroSimPSpice". Prentice Hall, Upper Saddle River, NJ, 1998..(in English).
[3] A. R. Newton, J. D. Deutch. "Data-flow based behavioral-level simulation and synthesis". Proc. IEEE ICCAD, Sept 1983..(in English).
[4] J. Ogrodzki, "Circuit simulation methods and algorithms". CRC Press, Boca Raton, Florida,USA, 1994..(in English).
[5] J. Ogrodzki, "Computer analysis of electronic circuitsPWN", Warsaw, 1995.(in Polish).
[6] D. Bukat, J. Ogrodzki, "Compact modelling in circuit simulation: the general purpose analyzer OPTIMA 3". ISCAS 94 PROCEEDINGS, pages 383–386, 1994.(in English).
[7] Leon O. Chua Pen-Min Lin, ""Computer analysis of electronic circuits.WNT, Warsaw, 1981.(in Polish).
[8] Tom Niemann. Lex &Yacc Tutorial. WWW page. http://epaperpress.com/lexandyacc/.(in English).
[9] IEEE Standard VHDL (Integrated with VHDL-AMS changes). Institute of Electrical and Electronics Engineers, 345 East 47th Street, New York, NY 10017, USA, 1 Aug 1998.(in English).
[10] PHP. PHP developers WWW page. http://www.php.net/docs.php.(in English).
[11] Paweł Plaskura. ""Deriv www page" . WWW project page. http://deriv.aiva.pl.(in English).
[12] Paweł Plaskura, "Dero v4 microsystems simulator". s. AIVA, 2013. ISBN: 978-83-937245-1-2.(in Polish).
[13] Paweł Plaskura, ""Advanced methods of electronic simulation". *Computational methods and algorithms*. AIVA, 2013. ISBN: 978-83-937245-0-5.(in Polish).
[14] J. Ogrodzki, P. Plaskura,, "OPTIMA v4 User Manual. Institute of Electronic Systems", Warsaw University of Technology, Warsaw, 1999.(in English).
[15] A. Richard Newton, "Resve Saleh, Shyh-JyeJou. Mixed-mode simulation and analog multilevel simulation". Kluwer Academic Publishers", Norwell, Massachusetts 02061,USA, 1994.(in English).
[16] Bruce Schneier, "Cryptography for Practitioners: Protocols", *Algorithms, and Source Programs in C.WNT,* Warsaw, 2002.(in Polish).
[17] Deepak Kumar Tala, "Verilog Tutorial". WWW page. http://www.asic-world.com/verilog/veritut.html.(in English).
[18] D. C. van Duyn. "Modelling and simulation of solid-state transducers: the thermal and electrical energy domain" . Sensors and Actuators, A41:268–274, Jan 1994.(in English).
[19] Niklaus Wirth, "Algorithms + Data Structures = Programs.WNT", Warsaw, 2004(in Polish).

Text of the article was accepted by Editorial Team 10.02.2017

DERIVWWW – СИСТЕМА СИМВОЛІЧНОГО ДИФЕРЕНЦІЮВАННЯ НА БАЗІ WEB ТЕХНОЛОГІЙ

Павел Пласкура
доктор технічних наук
факультет соціальних наук, університет Яна Кочановського, філія в Піотркув Трибунальському,
Піотркув Трибунальський, Польща
ORCID ID 0000-0002-8257-781X
p.plaskura@unipt.pl

Анотація. У статті представлена електронна символічна система диференціювання. Показано, як розраховуються похідні. Дерева використовуються для внутрішнього представлення формул. Похідні генеруються шляхом перетворення дерева. Представлені алгоритми є частиною мікросистем симулятора Dero. Вони необхідні для розрахунку алгоритмів, таких як Ньютон-Рафсон. Вони можуть бути використані для генерації похідних у мовах опису моделі та для автоматичного розрахунку похідних. DerivWWW можна використовувати в дидактиці. Система може служити інструментом для того, щоб учні могли відраховувати функціональні похідні в точці. Вона також може бути використана у викладанні предметів технічного напряму. Символічні похідні зберігаються у форматі Tex, що дозволяє легко інтегруватися з іншим програмним забезпеченням. У статті обговорюються розроблені й упроваджені алгоритми. Наведені приклади їх використання.

Ключові слова: символічне диференціювання; веб-системи; інструменти навчання; ІКТ в освіті.

DERIVWWW – СИСТЕМА СИМВОЛИЧЕСКОГО ДИФФЕРЕНЦИРОВАНИЯ НА ОСНОВЕ WEB ТЕХНОЛОГИЙ

Павел Пласкура
доктор технических наук
факультет социальных наук, университет Яна Кочановского, филиал в Пиотркув Трибунальськом, Пиотркув Трибунальський, Польша
ORCID ID 0000-0002-8257-781X
p.plaskura@unipt.pl

Аннотация. В статье представлена электронная символическая система дифференцирования. Показано, как рассчитываются производные. Деревья используются для внутреннего представления формул. Производные генерируются путем преобразования дерева. Представленные алгоритмы является частью микросистем симулятора Dero. Они необходимы для расчета алгоритмов, таких как Ньютон-Рафсона. Они могут быть использованы для генерации производных в языках описания модели и для автоматического расчета производных. DerivWWW можно использовать в дидактике. Система может служить инструментом для того, чтобы ученики могли отсчитывать функциональные производные в точке. Она также может быть использована в преподавании предметов технического направления. Символические производные сохраняются в формате Tex, что позволяет легко интегрироваться с другим программным обеспечением. В статье обсуждаются разработанные и внедренные алгоритмы. Приведены примеры их использования.

Ключевые слова: символическое дифференцирование; веб-системы; инструменты обучения; ИКТ в образовании.