44

Automatic Differentiation in Intelligent Tutoring Systems

Vladislav Katkov and Ada Novosselova
Institute of Engineering Cybernetics, National Academy of Sciences, 6 Surganov Street, Minsk 220012, Belarus
katkov@newman.bas-net.by
Byelorussian State University, 4 Scorina Avenue, Minsk 220050, Belarus

Keywords: learning environments, intelligent tutoring systems, automatic differentiation

Abstract: The automatic differentiation is widely used with the decision of a wide class of problems. The expansion of languages C/C++, Fortran-77 and Pascal for these purposes and its realisation as system Diff are described. The application of the offered approach is shown by development several tutoring systems: Graph – for drawing of the graphs, Autopracticum – for automation tutoring works on numerical analysis and mathematical modelling and data analysis. The system Diff is realised on IBM PCs under supervising of Windows 95/97/98.

1. INTRODUCTION

Use of the computer for calculation of the value of function given by the formula has been known for a long time – from the moment of the computer's appearance. In many tasks along with value of function it is often required to know its derivatives. The existing practice of such tasks is that all analytical manipulations are carried out manually beforehand, for stages of algorithm development, and then the program is written and executed. This approach has three problems. First, the analytical manipulations can appear rather labour consuming and, as they are carried out by the person, the probability of mistakes is high. Second, these mistakes are in the program

D. Watson et al. (eds.), *Networking the Learner*

and in practice they cannot be discovered except by programmer's methods. Consequently the process of program debugging becomes longer. Third, with change of initial functions it is necessary to repeat the calculations anew and to correct the program; in many cases new mistakes penetrate into the program and it will require the repeated debugging.

As the process of differentiation is strictly formalised, it can be automated with the help of the computer. Attempts to make it were undertaken repeatedly. Apparently, Kahrimanian (1953) and Nolan (1953) were the first. The Kantorovich's article (Kantorovich 1957) provided the determining influence on works on the analytical calculation. The formalism for representation of any expression as a sequence of tetrads consisting of operands, binary and monadic operations were offered in it. The program CODEX and its further modification SUPER-CODEX intended for automatic differentiation on a Fortran were developed at the University of Wisconsin-Madison (USA) under a management of Reiter (Rall 1981) in mid-sixties. The procedures of differentiation were included in standard library of compiler; as a result the applied program practically did not require preliminary processing and at once passed on compilation and execution. In 1992 at Argonne National Laboratory and Rice University (USA) the system ADIFOR for automatic differentiation on Fortran-77 was developed under approximately the same scheme that was used in CODEX and SUPER-CODEX (Bischof et al. 1992).

The *Diff* system developed in the Institute of Engineering Cybernetics, National Academy of Sciences, Belarus, (Katkov 2000a) calculates the partial and total derivatives, derivatives of implicit functions, derivatives of functions given by sequence of formulas, Jacobian, Hessian matrix (matrix from second order derivatives), and Taylor's series etc. For inclusion of differentiation in C-programs the language was extended by the description of functions and was complemented by corresponding library of procedures for calculation of derivative, Jacobian, expansion in Taylor's series and other means.

Recently there was a new application of differentiation connected with sensitivity analysis of numerical solution to the errors or uncertainties in the input data and parameters (Dongming et al. 1997). If the solution is calculated by finite-difference methods it results in necessity of analytical calculation and numeric estimation of several millions derivatives already for grids of the order 50 x 50 nodes. Manually it's practically impossible to fulfil this work. This problem is easily resolved using automatic differentiation systems.

2. AUTOMATIC DIFFERENTIATION IN MATHMATICS COURSES

This technique is widely applied in the educational process. For example, with study of Newton's method for minimization of multivariable function student can investigate speed of convergence for different input functions, different initial approximations, and different conditions of the iterative loop termination. We list some other themes in which the automatic differentiation is effectively used.

- Calculation of a root of the equation by Newton's method – use second derivative for choice of initial approximation and first derivative with calculation of next iteration.
- Drawing of the graph of function – calculation of first derivative for finding of minimum/maximum of function and second derivative for calculation of points of inflection, areas of convexity/concavity of function, finding asymptotes.
- Calculation of limits with use of l'Hospitale's rule.
- The calculation of integral with an estimation of the remainder term – for example, in Simpson's method is required to know fourth derivative.
- The solution of the differential equation by expansion in Taylor's series.
- The calculation of numerical values of Lipschitz's constant with an estimation of the differential equation solution.
- The investigatio of the singular points of the differential equation $y' = P(x,y)/Q(x,y)$ – expansion of P and Q in Taylor's series.
- The construction of tangents to a curve, calculation of radius of curvature, length of a curve, evolute and evolvent.
- The summation of functional series with an estimation of the remainder term.
- The calculation of integral by a method of differentiation on parameter etc.

3. AN EXAMPLE OF *DIFF* SYSTEM USE

Suppose it is required to calculate a minimum of function of two variables by Newton's method. The method has a square-law speed of convergence, but its use in practice demands hand-operated differentiation with all negative consequences.. As rule, the user is compelled to use other, less effective algorithms.

Algorithm. Let (x_0, y_0) is an initial approximation to a point of minimum. The next approximation is calculated under the formulas

$$x_{k+1} = x_k - (h_{2,2}fx_k - h_{1,2}fy_k)/\det(h), \quad y_{k+1} = y_k - (-h_{2,1}fx_k + h_{1,1}fy_k)/\det(h),$$

where $\{fx, fy\}'$ is gradient of function $f(x,y)$, $\{h_{i,j}\}$ is Hessian matrix: $h_{1,1} = f_{xx}$, $h_{1,2} = h_{2,1} = f_{xy}$, $h_{2,2} = f_{yy}$, and det (h) is determinant of a matrix h. The calculations are performed by iterations so long as two consecutive approximations will differ less than on ε.

The program for function $f(x,y) = 100(y-x^2)^2 + (1-x)^2$ can be written down in the following form:

```
// Standard introduction to the program
$func_i f(x,y) = 100*(y-x^2)^2 + (1-x)^2;
$func_i fx(x,y) = $diff(f,x);       // Gradient of f(x,y)
$func_i fy(x,y) = $diff(f,y);
$func_i H11(x,y) = $diff(fx,x);  // Hessian matrix
$func_i H12(x,y) = $diff(fx,y);
$func_i H22(x,y) = $diff(fy,y);

 void Minimum()              // Main program
{ const double eps=1e-6;    // Error
  double x0=-1.2, y0=1.0;   // Initial approximation
  double h11, h12, h22, det; // Hessian matrix h and its determinant
  double fx, fy, dx, dy;     // Gradient of f(x,y)
  int iter = 0;              // Iteration counter
  do
   { iter++; h11=H11_c(x0,y0); h12=H12_c(x0,y0);
     h22=H22_c(x0,y0); fx=fx_c(x0,y0); fy=fy_c(x0,y0);
     det=h11*h22-h12*h12;
     dx=(h22*fx-h12*fy)/det; dy=(h11*fy-h12*fx)/det;
     x0-=dx; y0-=dy;
     printf(/* iter=..., x=..., y=..., f(x,y)=...*/);
   } while (sqrt(dx*dx+dy*dy)>=eps;
   return;
}
```

Let us make some explanation. A line '$func_i f(x,y)=100*(y-x^2)^2 +(1-x)^2;' describes the function: its name (f), arguments names (x, y) and expression. In a line '$func_i fx(x,y)=$diff(f,x);' one more function (derivative $\partial f/\partial x$) is defined; the differentiation operation '$diff(...)' is used for it. To carry out the computation it is necessary to transform the function definition '$func_i ...' into C-text. Just it also is done by a description '$func_i fx(x, y) = ...', which translates the announcement 'fx' into the function's definition 'fx_c ...' in C-language and inserts it into the program, adding suffix 'c' to name 'fx'. In result the standard call 'fx_c(...)' in the C-program may be used. For example, for 'fx_c' the next text will be generated

```
double fx_c(double x, double y)
{ return (-400.0*(y-x*x)*x-2.0*(1.0-x)); }
```

The following results will be printed after program updating by the *Diff* system, compilation and execution:

```
iter = 1    x = -1.17...      y =  1.3...       f(x,y) =     4.7...
iter = 2    x =  0.7631...    y = -3.1...       f(x,y) = 1411.8...
iter = 3    x =  0.7634...    y =  0.58...      f(x,y) =     0.05...
iter = 4    x =  0.9999953    y =  0.94...      f(x,y) =     0.31...
iter = 5    x =  0.9999957    y =  0.9999914    f(x,y) =     0.0000000
iter = 6    x =  1.0000000    y =  1.0000000    f(x,y) =     0.0000000
iter = 7    x =  1.0000000    y =  1.0000000    f(x,y) =     0.0000000
```

If there is a need to solve the same task by another method that is not requiring differentiation, for example, by Nelder-Mead algorithm (Himmelblau 1972) with the same initial data, 179 iterations will be required. As we see, the application of automatic differentiation allows giving 'a second life' to such powerful method, as Newton's one.

4. *GRAPH*: INTELLIGENT TUTORING SYSTEM FOR GRAPH DRAWING

The function graph is powerful and convenient tool for the research of functional dependencies in mathematics and applications. Graph drawing is widely used in various activities at schools, colleges, universities, R&D institutes, in business, in a science, and in industry. This activity is realised in many applied programs, for example, in universal systems of numerical and analytical manipulations *Mathematica* and MATLAB. The universality and the multipurpose of such systems means that the user is compelled in the beginning to study them, and only after that they be able to modest solve the simple task – to draw the graph relevant to their function. For example, the *Mathematica* user's guide occupies 1400 pages.

The purpose of our work – to propose the simple and effective system for graph drawing not requiring any appreciable efforts in mastering it by the novice (Katkov 2000b). At the same time for the experienced user there are a additional possibilities allowing users to build and analyse the complex graphs of functions and to represent the result in a convenient form. It is permitted to make some transformations over graph for the reception of the required image or analysis of function. The system was developed in two

variants: *School*, for the beginners, and *Basic*, in which all facilities of the system are accessible. The *Graph* system works in a semi-automatic mode and provides help for the user in inconvenient settings. Basically it works through updating of the singular points and work to infinity.

The system has the built-in library of the most widespread functions, in which for each function will be given its formula, graph with some typical values of parameters and brief explanation. The library consists of two parts: libraries of school functions (algebraic functions, trigonometrical functions and inverse to them, exponent, logarithm, power, hyperbolic and other functions) and basic library (special functions).

The *Graph* system has the interconnected set of help and demonstration examples. There are rich means of polygraphic representation of the graphs for the most popular textual and graphic editors.

4.1 Construction of Function Graph

The process of graph drawing consists from the following steps:

- To determine domain of function definition.
- To calculate the singular points, points of maximum and minimum, points of inflection and points of function discontinuity.
- To calculate asymptotes vertical, horizontal and sloping. To find limits of function in the chosen points, including indefinites of a kind 0/0, ∞/∞, $0\cdot\infty$, $\infty-\infty$, 1^{∞}, 0^{0}, ∞^{0}.
- To set the sizes of the graphs, to draw axes to put inscriptions.
- To draw the graph of function.
- To edit a figure before printing or insert in the document.

The domain of definition. The *Graph* system calculates the domain of definition for what writes out restrictions on argument of function: the denominator is not equal to zero, the argument of the 'log' function is positive, the subradical expression for a root of an even degree is non-negative etc.

The singular points and function breaks points, points of maximum and minimum, points of inflection and points of function discontinuity are defined. The points of a maximum and minimum, and also point of inflection are calculated with use of the first and second derivatives.

The calculation of asymptotes vertical, horizontal and sloping. The parameters k and b of sloping asymptote $y = k \cdot x + b$ calculated on well-known formulas. The uncovering of indefinitesses 0/0, ∞/∞ and others with use of l'Hospitale's rule is made up in a semi-automatic mode, when the user can see intermediate result and to operate the analytical calculations.

The setting of the graph sizes, drawing of coordinate axes and inscriptions. Before drawing of the graph it is necessary to specify its sizes,

to draw axes of coordinates, to choose types of scales for axes of coordinates (usual, logarithmic, half-logarithmic), to superscribe numbers on them, to set drawing a coordinate grid, if it is necessary. The graph can be supplied with inscriptions and explanatory text.

Drawing of the graphs. The *Graph* system works in a dialogue mode and can simultaneously build the graphs of two functions $f(x)$ and $g(x)$ in two different windows or in one window by superposition. The graphs are drawn in Cartesian coordinates. The function can contain no more than one parameter. The *Graph* system can build family of the graphs with various values of parameters. Each graph is represented by a curve, at which the color, thickness and type of a line (from an offered set) are easy selected.

The function is set analytically in one of the following forms:

a) Explicitly, as $y = f(x)$, $a \le x \le b$.

b) Parametric, as $x = X(t)$, $y = Y(t)$, $a \le t \le b$.

c) Implicitly, as $f(x, y) = 0$, $a \le x \le b$ or $c \le y \le d$.

Edition of a figure for printing or insert in the document. The user can transform a figure to a textual or graphic file of the appropriate textual or graphic editor. It is supposed to use such formats, as .doc, .rtf, .pdf, .bmp, html, .pcx, .gif, .jpg. It is possible also to convert symbolic expression for function and its derivatives into a textual file for WinWord, Adobe Acrobat etc. The *Graph* system can translate the function definition and its derivatives in languages C/C++, Fortran, and Pascal.

5. *AUTOPRACTICUM*: INTELLIGENT TUTORING SYSTEM

The intelligent system of automation of learning process in mathematics should have the following minimal features (Katkov and Novosselova 1997).

- Symbols manipulations: automatic differentiation, simplification of expressions, reduction of similar terms and so on.
- Drawing of the graphs of functions.
- Friendly interface: the native language, dialogue menu, convenient features of the definition of input parameters and presentations of results, and help.
- Ergonomics requirements: minimisation of number of control keys, colour, sound and so on.

Proposed below is a model of an integrated approach to the process of automated teaching (training). The model includes: (1) maximal use of computer for routine work; (2) some popular symbolic manipulations; (3) broad use of textual and graphic help menus for topics under study; (4) use of ergonomic means for control of the teaching system (colour, sound,

dynamic pictures, light buttons/keys, menus); (5) presentation of results in an easy way (graphs, diagrams, tables, pictograms).

The *Autopracticum* system provides an approach which makes the system carry out all the routine while leaving the intellectual part for the student (well-grounded selection of the numerical method, examination of its properties and peculiarities, analysis of results). The system is programmed to keep up basic didactic principles: problem-oriented education, use of visual aids, activity and consciousness, simplicity, systematic approach and consistency, sound knowledge and recurrence.

The *Autopracticum* system has the following sections that calculate roots of non-linear equations, calculate integral values, draw function graphs, and carry out mathematical modelling and data analysis.

The system is intended for the students 2-4 courses and is used in educational process of the Byelorussian state university. Use of *Autopracticum* raises efficiency of educational process in 3-5 times.

5.1 Method of practical work in numerical analysis

When assigned problems to get roots of non-linear equations, the student, using the *Autopracticum* system, can: (1) view the function's graph to visually define the segment where the root is located; (2) choose the appropriate method using a concise help menu in the form of static or motion pictures (Newton's method, for example); (3) enter the initial data: function $f(x)$, segment $[a, b]$, and the error of calculation; (4) run the calculation with the system automatically calculating the $f'(x)$ and $f''(x)$ derivatives in the analytic form, selecting a or b as the initial x_0 approximation by $f'(x_0) \cdot f''(x_0)$ sign, creating the calculation formula $x_{n+1} = x_n - f(x_n)/f'(x_n)$, making the calculations and producing the result – the root value and the number of iterations; (5) store the results of calculations and print them as a report at the end of the work.

With the results derived by different methods for different initial data at hand, the student can assess the advantages and disadvantages of the examined algorithms.

6. CONCLUSION

As well as for any other tool of programming, the area of its real application depends on needs and ingenuity of the user. We propose that the automatic differentiation technique will allow the wider application of mathematical methods in the most various applied areas alongside with the numerical analysis.

REFERENCES

Bischof, C., Carle, A., Corliss, G., Griewank, A. and Hovland, P. (1992) ADIFOR – Generating Derivative Codes from Fortran Programs. *Scientific Programming*, Vol. 1, pp. 11-29.

Dongming, H., Byun, D.W. and Odman, M.T. (1997) An automatic differentiation technique for sensitivity analysis of numerical advection schemes in air quality model. *Atmospheric Environment*, Vol. 31, # 6, pp. 879-888.

Himmelblau, D.M. (1972) *Applied Nonlinear Programming*. McGraw-Hill Book Company.

Kahrimanian, H.G. (1953) *Analytical Differentiation by a Digit Computer*. M. A. Thesis, Temple Univ, Philadelphia.

Kantorovich, L.V. (1957) About mathematical symbolics convenient for performance of machine calculations. *Doklady Akademii Nauk SSSR*, 11, pp. 738-741 (in Russian).

Katkov, V.L. (2000a) Automatic differentiation and its applications. *Vesti National Academy of Sciences*, Belarus, 1, pp. 68-75 (in Russian).

Katkov, V.L. (2000b) The Graph system for function graph construction. In *Proceedings of Conference on Educational Uses of Information and Communication Technologies*. 16th World Computer Congress 2000, Beijing, August, p. 240.

Katkov, V.L. and Novosselova A.N. (1997) Intelligent Tutoring in Numerical Methods. In *Proc. of AI-ED World Conference on Artificial Intelligence in Education*, Kobe, Japan, August, pp. 607-608.

Nolan, J. (1953) *Analytical Differentiation by a Digit Computer*. M. A. Thesis, M. A. MIT, Cambridge.

Rall, L. B. (1981) *Automatic Differentiation: Techniques and Applications*. Lectures Notes in Computer Science, Vol. 120. Springer-Verlag, Berlin Heidelberg New York, pp. 1-160.

BIOGRAPHIES

Professor Vladislav Katkov is a head of laboratory and works in the field of computer algebraic systems and automation of educational process in a higher school. Under his management some systems symbol manipulations and development of the computer programs by a top down method were created. Recently his interests have been concentrated on writing of the electronic textbooks on computer science.

Ada Novosselova is lecturer, she has developed intelligent tutoring system *Autopracticum* for automation of laboratory works on numerical methods, mathematical modeling and analysis of the data. Further work on the development of the system is being undertaken.