

ZBIGNIEW MROZEK*

TEACHING, MODELING AND VISUALISATION OF ORDINARY DIFFERENTIAL EQUATIONS

NAUCZANIE, MODELOWANIE I WIZUALIZACJA RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH

Abstract

Advances in computer technology and increased interest in dynamical systems influence the way of teaching ordinary differential equations. The paper presents inquiry oriented teaching, usage of modeling, visualisation and interactive web services. Last chapter describes the ways of using MATLAB or public domain software (e.g. Octave) to solve ordinary differential equations.

Keywords: dynamical system, ordinary differential equation

Streszczenie

Postęp w technologii komputerowej oraz wzrost zainteresowania modelowaniem systemów dynamicznych wpływa na sposób nauczania matematyki, w tym równań różniczkowych zwyczajnych. Przedstawiono podejścia: nauczania przez zadawanie pytań, wykorzystanie modelowania, wizualizacji i interaktywnych usług sieci web. Ostatni rozdział opisuje sposoby wykorzystania środowiska MATLAB lub oprogramowania dostępnego jako public domain (np. Octave) do rozwiązywania równań różniczkowych zwyczajnych.

Słowa kluczowe: równanie różniczkowe zwyczajne, system dynamiczny

*Faculty of Electrical and Computer Engineering, Cracow University of Technology; zbig-niew.mrozek@pk.edu.pl

1. Introduction

In most cases teaching of linear second order ODE (ordinary differential equation), as seen in major textbooks, is done in a very procedural manner where the emphasis lies in identifying the type of equation and then apply a number of well established steps that yield to the solution [14].

Need to improve effectivity of teaching differential equations is not new. The importance of visualization of solutions was known to Karl Menger. In his paper [8] he noted that "the simplest introduction to the theory of differential equations is offered by nature". He suggested sprinkle fine iron splinters in magnetic field to visualize solution of the 1-st order ODE (Fig.1).

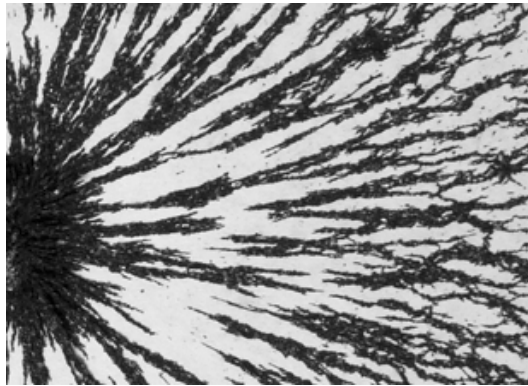


Fig. 1: Magnet and fine iron splinters, ODE visualization proposed by Menger (see also Fig.2)

Gian-Carlo Rota [17] shows how little the content of the ODE courses has changed since Cauchy. Even the order of presentation of the topics has not been altered. Rota was invited to prepare several addresses for Mathematical Association of America. At the 1997 meeting, he described lecture notes for introductory course in differential equations by Cauchy and the book on Differential Equations by Boole. Both were published at about the same time in the nineteenth century. Cauchy notes are written in an attractive, flowing style. Half of the book by Boole describes the solution of the first order differential equations. But the Boole's techniques are not very useful now. Rota claims that only two of them have survived, separation of variables and changes of variables.

2. Elementary course of differential equations

In an elementary course of differential equations, students should learn [17] a few basic concepts that they will remember for the rest of their lives, such as the universal occurrence of the exponential function, stability, the relationship between trajectories and integrals of systems, phase plane analysis, manipulation of the Laplace transform, perhaps even the fascinating relationship between partial fraction decompositions and convolutions via Laplace transforms. Students should also learn the basic theorems.

2.1. The normal system of first order ODEs

The normal system of first order ODEs will be considered.

$$\begin{pmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_n \end{pmatrix} = \begin{pmatrix} Y_1(y_1, \dots, y_n; t) \\ \vdots \\ Y_n(y_1, \dots, y_n; t) \end{pmatrix} \quad (2.1)$$

Many kinds of ODEs can be reduced to the normal system, e.g. n th order ODE

$$\frac{d^n y}{dt^n} = F\left(t, y, \frac{dy}{dt}, \dots, \frac{d^{(n)}y}{dt^{(n)}}\right) \quad (2.2)$$

can be changed to the normal system of first order ODE (3) substituting $y_1 = y, y_2 = \frac{dy}{dt}, \dots, y_n = \frac{d^{n-1}y}{dt^{n-1}}$

$$\begin{pmatrix} \frac{dy}{dt} \\ \vdots \\ \frac{d^{n-1}y}{dt^{n-1}} \end{pmatrix} = \begin{pmatrix} y_2 \\ \vdots \\ F(t, y_1, \dots, y_n) \end{pmatrix} \quad (2.3)$$

2.2. Theorems on finding solutions of normal system of first order ODEs

The Picard–Lindelöf theorem, Picard’s existence theorem, Cauchy–Picard or Cauchy–Lipschitz theorem is an important theorem on the existence and uniqueness of solutions to first-order equations with given initial conditions.

Theorem 1. The Cauchy–Picard existence and uniqueness theorem

Assume $f : R^n \rightarrow R^n$ has a continuous derivative. For every initial condition x_0 there exists $r > 0$ such that on the time interval $[0; r)$ there exists exactly one solution of the initial value problem

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

Since there is no restriction on F to be linear, this also applies to **non-linear equations**, and it can also be respectively applied to **systems of first order ODE equations** [1].

2.3. Inquiry oriented teaching of differential equations

Inquiry oriented teaching is based on guiding students with questioning, prompting and useful feedback [11, 19, 20]. Students are asked for reasoning and justification. The main results emerge through whole class discussion and oral presentation done by students.

Kwon and Rasmussen [5, 6, 15] engaged students in the Inquiry Oriented Differential Equations (IO-DE) project. This was a collaborative effort to explore the prospects and possibilities for improving undergraduate mathematics education, using differential equations as a case example.

Rasmussen, at all [16] conducted an evaluation study between students taking part in IO-DE project and traditionally taught classes. They compared students' skills and conceptual understandings of central ideas and analytic methods for solving differential equations. Students taking part in project, regardless of academic backgrounds and gender differences, outperformed traditionally taught students on the post-test. Thus, the IO-DE approach could enhance long-term retention of mathematics for all students. The drawback is that the IO-DE needs much more work for botch, students and professors [11]. More mathematical problems useful for the ODE class may be found in [18].

2.4. Team-based labs of differential equations

The course which covers traditional topics such as first-order equations, second-order linear equations with constant coefficients, Laplace transforms, and systems of first-order equations has been team-taught by two instructors; one from the School of Mathematics and Computer Science and one from the School of Engineering and Technology [13]. The emphasis was on the real-life modeling applications of differential equations.

Examples of teaching a mathematics course with an emphasis on applications have been reported in literature numerous times [4]. Students are assigned challenging real-life problems which are typically open-ended and unstructured. The process of linking the mathematical theory with applications has also led to introduce team-based labs into a differential equations course in which students are exposed to extensive numerical experimentation. Another approach is to provide the students with easy-to-use technological tools such as MATLAB programs or interactive web sites.

3. Modeling and simulation in teaching ODE

Computer Algebra Systems (CAS) such as Maple, MATLAB or Mathematica are capable of yielding the numerical or analytical solution. One may choose also public domain software such as COPASI, ECLlect2, FuncDesigner, GNU Octave, Julia, Modelica, Scicoslab, Scilab. Using user friendly software, the student's focus is shifted

from applying the well defined pattern of steps, to that of understanding the derivation of these steps [14].

Computer software plays a central role supporting modeling and simulation in all engineering areas. Thus numerical methods become an important tool to support teaching. Numerical methods and symbolic manipulation are implemented as computer programs, and the results are immediately visualized using powerful graphic software [2]. Even final reports can be automatically generated and printed using e.g. MATLAB Report Generator or menu option Publish (in MATLAB [10]).

3.1. Interactive web resources

A good example of Interactive web resources is *Interactive Differential Equations* page <http://www.aw-bc.com/ide> arranged and supported by Pearson Addison-Wesley. The team of authors, from three universities (Cornell, California Polytechnic and St. Louis) prepared 31 problems and many additional sub-problems in area of linear algebra, systems of differential equations, 1st and 2nd order ODE, series solutions, chaos and bifurcation.

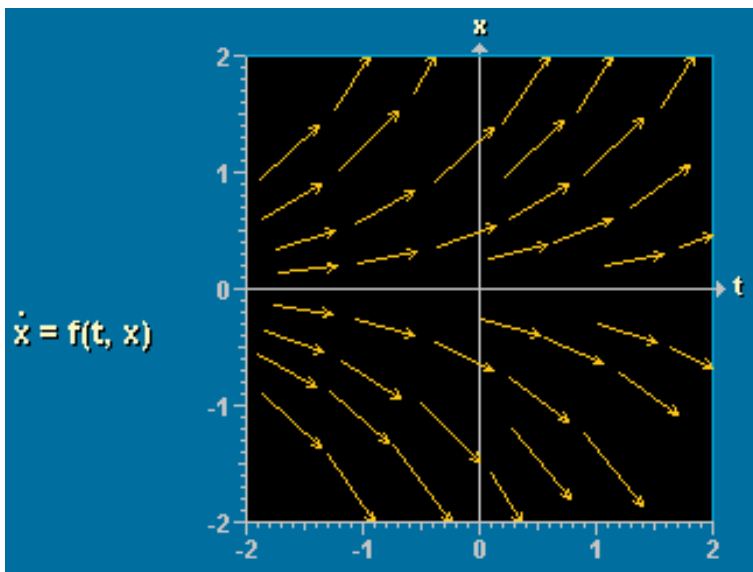


Fig. 2: Interactive web resource [3]: visualization of slope field for $\dot{x} = x$

Each problem starts with few pages of pdf file describing the given problem (may be printed). The tool instruction page describes user interface: how to use buttons and sliders in actual problem. To start simulation one has to setup the initial conditions (IC). This is done by clicking the plot area with mouse at the point chosen to be new

IC. To get family of plots, one has to use several starting points. Operating interactive web is very easy, intuitive and user friendly.

A nice example from this web is *Graphing Differential Equations>Slope Fields* (Fig.2). It may be found on the site:

<http://www.aw-bc.com/ide/idefiles/media/JavaTools/slopefld.html>. To start, one has to click inside any plot area to setup the initial condition for the equations. All problems are ready to use in class or as homework for students and are completely free.

4. MATLAB[®], Simulink[®] and other tools

MATLAB is a general high level programming language and programming environment. It is user friendly and integrates reliable algorithms of applied mathematics and numerous expansion modules focused on specific fields of application. MATLAB enables to easily solve a variety of problems in science, industry, medicine, economy and many other areas by facilitating access to efficient computational algorithms and the ability to visualize the results of computation [10, 13]. Many advanced tools such as toolbox libraries, Simulink and blockset libraries and many other) are available for purchase

MATLAB successfully replaces universal programming languages (Fortran, C, C#, C++) in the area of scientific and technical calculations. Its professional math library is based on optimized packages: **LAPACK** (*Linear Algebra Package*) and **BLAS** (*Basic Linear Algebra Subroutines*).

4.1. ODE algorithms in MATLAB

MATLAB uses effective variable order and variable step solvers for normal system of first order ODEs. The ODE solvers are [7]:

- **ode45** formula, the Dormand-Prince pair is used as a "first try" for most problems;
- **ode23** solver is based on an explicit Runge-Kutta (2,3) pair of Bogacki and Shampine. It may be more efficient than ode45 at crude tolerances and in the presence of mild stiffness;
- **ode113** solver is Adams-Bashforth-Moulton PECE solver. It is a multistep solver and needs solutions at several preceding time points to compute the current solution.

For stiff problems there are four solvers

- **ode15s** is a multistep solver based on the numerical differentiation formulas (NDFs). Optionally, it uses the backward differentiation formulas (BDFs, the

Gear method). The `ode15s` and `ode23t` can solve some DAEs (*Differential Algebraic Equations*) of the form $M(t, y)y' = f(t, y)$, where $M(t, y)$ is singular. The DAEs must be of index 1.

- `ode23s` is based on a modified Rosenbrock formula of order 2. It is a one-step solver and may be more efficient than `ode15s`
- `ode23t` is an implementation of the trapezoidal rule using a "free" interpolant. It may be used for moderately stiff problems and DAEs.
- `ode23tb` is an implementation of TR-BDF2, an implicit Runge-Kutta formula using a trapezoidal rule step and then a backward differentiation formula of order 2.

Fully implicit differential equations of the form $f(t, y, \dot{y}) = 0$ may be solved with solver `ode15i` using the variable order BDF method.

4.2. Other problems related to differential equations

MATLAB may be used to solve many problems related to differential equations, as

- boundary value problem for ODE
- delay differential equation initial value problem
- partial differential equations (PDE): 1-D initial-boundary value problem for parabolic or elliptic PDE. To solve PDE in two-space dimensions (2-D) and time, one should buy *the Partial Differential Equations Toolbox*TM, which extends MATLAB.
- numerical integration and differentiation may be done in MATLAB with quadratures, double and triple integrals, and multidimensional derivatives

4.3. ODE example

Given 3rd order Cauchy problem:

$$\ddot{y} + 6\dot{y} + 11y + 6y = 0, \quad y(0) = 0, \quad \dot{y}(0) = 0, \quad \ddot{y}(0) = 1 \quad (4.1)$$

4.3.1. Solving ODE example with `odeXX` solver

MATLAB ODE solvers accept only set of first-order differential equations. To solve higher-order ODEs, one has to rewrite equation (1) as an equivalent system $\dot{y} = f(t, y)$ of first-order ODE

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{pmatrix} = \begin{pmatrix} y_2 \\ y_3 \\ -6y_3 - 11y_2 - 6y_1 \end{pmatrix}; \quad \begin{pmatrix} \dot{y}_1(0) \\ \dot{y}_2(0) \\ \dot{y}_3(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (4.2)$$

The next step is to prepare few lines of MATLAB code (M-function) that evaluates the right hand side of (2). This is a very flexible approach as any nonlinearities may be easily computed in MATLAB. There is even more flexibility using `ode15i` solver, which accepts fully implicit set of differential equations of the form $f(t, y, \dot{y}) = 0$. The MATLAB function for problem (2) is

```
function [ dydt ] = odefun3(t,y) % as needed by odeXX solvers
% computes vector of derivatives of equivalent ODE system
dydt=[y(2); y(3); -6*y(3)-11*y(2)-6*y(1)];
%vector dydt=[dydt(1); dydt(2); dydt(3)]
```

The above function (% means start of comment) will be used by solver (e.g. `ode23`) to compute new values of derivatives for all needed t and y . The solver may be called from the MATLAB **Command Window** as

```
>>[t,y]=ode23('odefun3',[0,10],[0,0,1]); plot(t,y(:,1))
%compute and plot solution
```

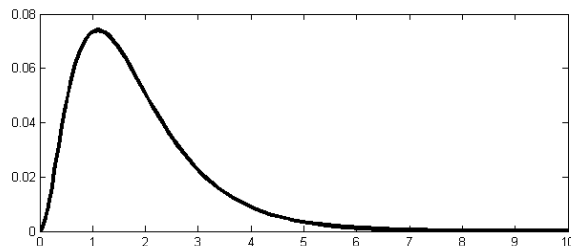


Fig. 3: Numerical solution of 3-rd order ordinary differential equation (1)

Solution may be examined for any change done in equation or its initial conditions, as presented on Fig.4. On slider movement the β parameter in ODE $\ddot{x} + \beta\dot{x} + x = 0$ is changed and new plots are immediately generated. Visualisation is supported with GUI (*graphical user interface*), already included in MATLAB.

4.3.2. Analytical solution of ODE

The analytical solution of ODE may be obtained using the `dsolve` command if additional MATLAB library *Symbolic Math Toolbox* is available. The code needed for $\ddot{y} + 6\dot{y} + 11y = 0$ is:

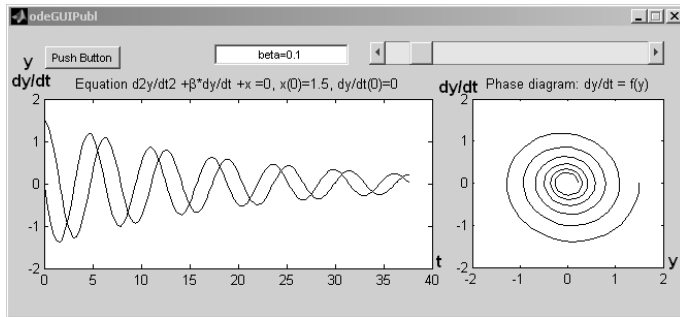


Fig. 4: Graphical User Interface (GUI) and visualisation in MATLAB.

```
>> y=dsolve('D3y+6*D2y+11*Dy+6*y = 0','D2y(0)=1, Dy(0)=0, y(0)=0')
y = exp(-t)/2 - exp(-2*t) + exp(-3*t)/2
```

Solution may be plotted using `ezplot(y, [0,10])`. where $[0, 10]$ is the time span for plot. If initial conditions are not given, the general solution will be computed.

```
>> y=dsolve('D3y+6*D2y+11*Dy+6*y = 0')
y = C1*exp(-2*t) + C2*exp(-t) + C3*exp(-3*t)
```

where C_1 , C_2 , C_3 are arbitrary constants used to satisfy the initial conditions.

4.4. Solving ODE using MATLAB/Simulink

Simulink is integrated with MATLAB. It provides a graphical editor, customizable block libraries (blocksets) and solvers for modeling and simulation of dynamic systems.

Using Simulink, there is no need to prepare any lines of program code, as blocks from Simulink library are already linked to executable code. There are integrators, amplifiers, adders and other useful blocks in the Simulink library. For example, the integrator block computes the value of the integral of its input signal with respect to time and initial condition. Icon of the integrator block shows $\frac{1}{s}$, the Laplace transform of the unit step function $1(t)$.

Figure 5 shows Simulink model of 3rd order ODE (1). There are three integrators (icon is $\frac{1}{s}$), three amplifiers (its gain is equal to the coefficients in ODE equation) and one adder. The scope block is used to visualize the ODE solution as a function of time.

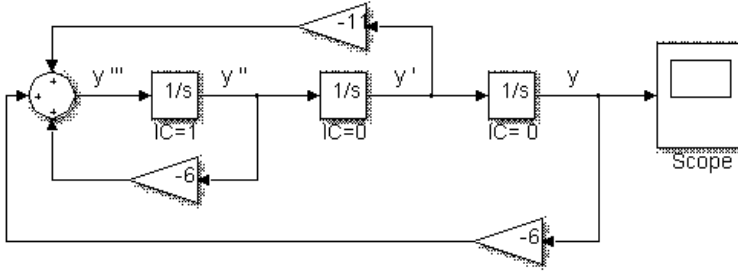


Fig. 5: Graphical model of 3rd order ODE using three integrators ($1/s$)

5. Comparison between modeling and visualisation in MATLAB and its GNU/GPL alternative

5.1. GNU Octave

Matlab-compatible ODE functions are provided by the `odepkg` package included in standard distribution of Octave [12]. The `lsode` function uses Hindmarsh's ODE solver and is replacement for `ode23` and other ode solvers from MATLAB. Here is an example of solving ode problem (4) in Octave. The `lsode` uses the same parameters as `ode23`, but order of parameters is changed.

```
t=[0:0.1:10]';
[X, ISTATE, MSG] = lsode ('odefun3',[0,0,1],t);
plot(t,X(:,1))
```

The symbolic computation package is not included in Octave.

5.2. Modelica[®] language and Modelica libraries

Modelica [9] is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems. It has free libraries with elements used in mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented models. To use Modelica, a user environment as Dymola, MapleSim, SimulationX or Vertex is needed. Another option is SimForge, entirely free graphical Modelica editor released under the terms of the GPL license.

Modelica models can be imported into Simulink using the export features of Dymola, MapleSim, SimulationX and Vertex. Industry is increasingly using Modelica for model based development. Many automotive companies, such as Audi, BMW, Daimler, Ford, Toyota, VW use Modelica to design energy efficient vehicles and/or improved air conditioning systems. Also power plant providers (ABB, EDF, Siemens) and many

other companies use Modelica. Modelica is a more advanced package and should be widely used in education. Figure 6 shows models of the same 22 kW DC motor prepared with Simulink and Modelica.

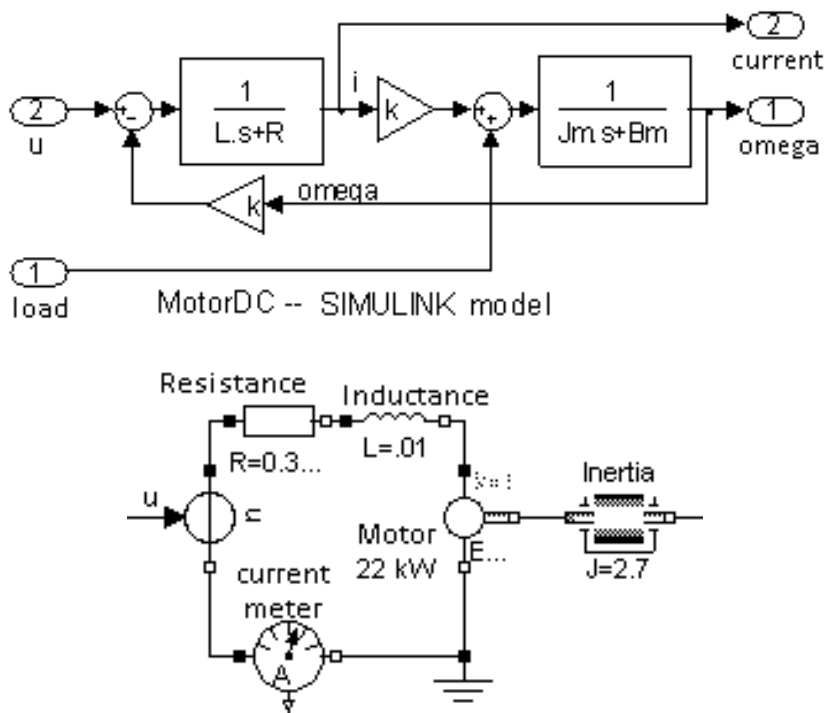


Fig. 6: Models of the 22 kW DC motor built in Simulink and Modelica. Adder, transfer functions e.g. $\frac{1}{Ls+R}$ and amplifiers k are used in the Simulink model. Modelica uses models of physical elements of real system.

6. Conclusion

Most students take the differential equations course in order to master techniques to be later applied in solving the real world problems in their profession. They should learn a few basic concepts that they will remember for the rest of their lives and they should learn how to use the available computer software to model and simulate the real-life applications of differential equations.

References

- [1] Boyce W.E, Diprima R.C, *Elementary Differential Equations and Boundary Value Problems*, (4th Edition), Wiley International, John Wiley & Sons, ISBN 0-471-83824-1, 1986.
- [2] Cariaga E. A., Nualart M. C., *Teaching and learning iterative methods for solving linear systems using symbolic and numeric software*, Comput Appl Eng Educ 10, 2002, 51—58.
- [3] *Interactive Differential Equations*, Pearson Addison-Wesley, <http://www.aw-bc.com/ide>, 2013.
- [4] Kadijevich D., Haapasalo L., Hvorecky J., *Using technology in applications and modelling*, Teaching Mathematics and its Applications, 24(2-3), 2005, 114—122.
- [5] Kwon O., Rasmussen C., *Towards an inquiry approach to undergraduate mathematics* Proceedings of the 4th East Asia Regional Conference on Mathematics Education, Penang, Malaysia, 2007, 39—46.
- [6] Kwon, O., at all. *Students' Retention of Mathematical Knowledge and Skills in Differential Equations*, School Science and Mathematics, 105(5), 2005, 1—13.
- [7] *MATLAB, Simulink*, <http://www.mathworks.com/products/>, 2013.
- [8] Menger K., *On the Teaching of Differential Equations*, The American Mathematical Monthly, Vol. 51, No. 7 1944, 392—395.
- [9] Modelica and the Modelica Association, <http://www.modelica.org>, 2014.
- [10] Mrozek B., Mrozek Z., *MATLAB i Simulink. Poradnik użytkownika*, Wydanie III, Helion, 2010.
- [11] Nabb K., *Inquiry in Differential Equations: A Teacher's Reflections*, 38th AMATYC Annual Conference, <http://www.amatyc.org/Events/conferences/2012/Jacksonville/proceedings.html>, Jacksonville, Florida, 2012.
- [12] Octave, <http://www.octave.org> and <http://octave.sourceforge.net/odepkg>
- [13] Padir Taskin, et al., *Teaching differential equations in a diverse classroom*, 2008 Annual Conference and Exposition, AC 2008-1548, American Society for Engineering Education, 2008.
- [14] Paraskakis I., *Rethinking the Teaching of Differential Equations through the Constructivism Paradigm*, In: ICALT 2003 - 2003 IEEE International Conference on Advanced Learning Technologies 9-11 July, 2003, Athens, Greece, 2003, 506—510.

- [15] Rasmussen Chris, Whitehead Karen, *Learning and Teaching Ordinary Differential Equations*, <http://www.maa.org/t-and-l/sampler/research-sampler.html>, The Mathematical Association of America, RESEARCH SAMPLER, No. 7, January 2003.
- [16] Rasmussen, at all. *Capitalizing on advances in mathematics and K-12 mathematics education in undergraduate mathematics: An inquiry-oriented approach differential equations*, Asia Pacific Education Review, 7(1), 2006, 85—93.
- [17] Rota Gian-Carlo *Ten Lessons I Wish I Had Learned Before I Started Teaching Differential Equations*, <http://euler.slu.edu/Dept/Faculty/marks/Pedagogy/RotaOnTeachingDiffEqns.pdf>, 1997.
- [18] *Teacher package: Differential equations, Plus Magazine*, University of Cambridge, <http://plus.maths.org/content/teacher-package-differential-equations>.
- [19] Wagner J., Speer N., Rossa B., *Beyond mathematical content knowledge: A mathematician's knowledge needed for teaching an inquiry-oriented differential equations course*, Journal of Mathematical Behavior 26, 2007, 247—266.
- [20] Zadawanie dobrych pytań
<http://www.ceo.org.pl/pl/cyfrowaszkola/kurs/zadawanie-dobrych-pytan>.