Interactive Environment for the Taylor integration (in 3D Stereo)

Alexander Gofen

The Smith-Kettlewell Eye Research Institute, 2318 Fillmore St.,
San Francisco, CA 94115, USA

Keywords: Taylor solver, Unlimited order, 3D Stereo animation,

Abstract. A cross-platform interactive application called the Taylor Center performs integration of ODE's with unlimited order of approximation, finite step, and high accuracy, using the Modern Taylor method. It exports the solution in different formats, graphs it in 2D and 3D anaglyphic stereo, and performs real-time animation of the motion. Designed under Windows in Delphi-6, the package implements a powerful Graphical User Interface, and is portable to windowed versions of Linux too. The downloadable Demo of the package [6] displays a variety of meaningful examples including such illustrative ones as the Problem of Three and Four Bodies and "Choreography". The application represents a kernel, which may be developed further in several directions, discussed in the article.

1. Introduction

Unlike earlier packages [1, 2], this sophisticated Taylor Solver is designed for PCs as *interactive application* with extensive Graphical User Interface. It is a cross platform Windows/Linux project developed in Borland's Delphi (a dialect of Pascal having such advance features as dynamic arrays, objects, and sophisticated graphics).

This All-In-One application is called the Taylor Center (according to the metaphor of a "Music Center") because it offers the environment, where researchers can input (import) ODE's in symbolic format, vary the parameters, explore the Taylor expansions and their convergence radius, integrate the equations with high accuracy (up to all available float point digits), store or export the results, graph the solutions and "play" them dynamically as bullet trajectories in real time animation. The feature making this application unique is that it can display non-planar 3D trajectories as anaglyphic stereo pairs (for viewing through Red/Blue glasses). In particular, it implements a 3D cursor (controlled by a conventional mouse). This cursor with "tactile" audio feedback permits user to "touch" points of non-planar curves and view their 3D coordinates.

Actually the Graphical Interface itself, reflecting all the specifics of interactive Taylor integration, became an important goal of the design: the goal of employing all the sophistication of the Windows GUI to control the environment for performing the Taylor integration and exploration of the solution in most comfortable manner.

From the programmatic point of view, any Taylor solver differs from conventional integrators in that the input – an Initial Value Problem – should be provided not simply as a numeric vector and a subroutine computing the right hand parts, but rather as arithmetic expressions themselves, representing the right hand parts in order to enable *automatic differentiation* (understood as optimized classical formulas for N-th order derivatives applied to sequences of basic expressions [3]).

Correspondingly, the output and interaction of a Taylor solver with other applications has its specifics too. The result is not just tabulated values of the solution, but rather its expansion into the Taylor series (an analytical element), or a sequence of such elements (although the tabulated solution may be exported also).

In this application the equations and the initial value problem must be inputted in common mathematical notation and Pascal syntax. Thus, the right hand parts are expected to be *finite* arithmetic expressions of any complexity over simple variables (no array elements or loops in this version).

2. Overview of the algorithm

The processing is based on *one-time* parsing of arithmetic expressions (representing ODE's), and *repeated* evaluation of the obtained postfix representation.

In general, any sequence of explicit equations

$$\begin{aligned} &u_0 = f_0(\{numeric\ values\ only\}); \\ &u_1 = f_1(u_0,\ \{numeric\ values\}); \\ &\ldots \\ &u_n = f_n(u_0,\ u_1, \ldots,\ u_{n-1},\ \{numeric\ values\}). \end{aligned} \qquad \textbf{(1)}$$

given in a form of arithmetic expressions, may be parsed into a postfix sequence. Evaluation of the postfix sequence consists of locating triplets (Operand1, Operand2, Operation) in it, evaluating elementary equations

$$\text{Result} = \text{Operand1} \circ \text{Operand2} \qquad \textbf{(2)}$$

and replacing the triplets with the corresponding Result until the whole sequence reduces to just one term – the final result of the evaluation.

Now consider an Initial Value Problem for an autonomous system of explicit 1st orders ODE's:

$$u_k' = f_k(u_1, \ldots,\ u_m), \quad u_k|_{t=a} = a_k, \quad k=1,\ldots m \qquad \textbf{(3)}$$

where u_k' means derivatives with respect to t ("hidden" in one of variables u_k). Equations (3) allow obtaining the set of first order derivatives u_k' and all higher order derivatives $u_k^{(N)}$ as well: that is an essence of the classical Taylor method. The Modern Taylor Method [3, 4] capitalizes on the fact that however complicated the right hand parts are, they are comprised of a sequence of simple formulas (instructions). Indeed, the instructions are exactly those resulting from the process of *postfix evaluation*. If the set of the allowed operations consists of the four arithmetic operations (and also raising to power, exponent, logarithm, trigonometric functions), computing the N-th order derivatives requires no more than $O(N^2)$ operations [3, 4]. Moreover, all ODE's with the so called *elementary* right hand parts (practically all ODE's used in applications) are reducible to the above mentioned rational and special functions format [3, 4].

To achieve more flexibility in specifying an Initial Value problem, the source data is organized in four sections in the following order: Symbolic constants, Initial values, Auxiliary variables, ODE's (Fig 2).

All the equations are processed in the following three stages:

(1) They are parsed line by line into the postfix sequences for the subsequent automatic differentiation. As a result of this *one time* procedure, a postfix sequence (actually a sequence of instructions for an abstract processor) is created.

(2) The differentiation process up to order N is performed. The set of coefficients of the Taylor expansion (also called *analytic elements*) for each variable is obtained;

(3) With the obtained Taylor coefficients, the convergence radius R is estimated, an integration step $h<R$ determined, and the Taylor expansion is used to increment the variables. Then, the stages 2 and 3 may be repeated as many times as necessary (unless a singularity is reached).

In this evaluation, instead of matching arithmetic operations to the corresponding CPU instructions, the operations are matched to the corresponding formulas of automatic differentiation, playing role of "instructions"

```
type TInstruction = record
      ty : (cc, cv, vc, vv); {flags: (C)onstant  or (V)ariable}
      op : char;             {operation: +, -, *, /, exp, ln, cos, …}
      i1, i2, i3 : word      {addresses, i.e. indexes of the array}
      end;
```

of an abstract AD Processor (Fig 1). Thus, the Processor performs iterative differentiation up to order N of statements (2)

$$Result^{(N)} := (Operand1 \circ Operand2)^{(N)}, \qquad N = 0, 1, 2, \ldots$$

according to the formulas of AD for computing N-th order derivatives of certain functions. For example, if the operation is multiplication, the Processor implements the Leibnitz formula

$$(uv)^{(N)} = \sum_{i=0}^{N} u^{(i)} v^{(N-i)}$$

and similarly for the other operations [3, 4] (the derivatives are normalized, meaning $u^{(i)}/i!$).

The set of instructions would remain the same even if we need to switch from real numbers to complex numbers, or to the arithmetic of interval pairs, or from one real type to another, including the cases of software-emulated floating point types of any given length (Fig 1).

Similarly, this very set of instructions may be trivially transformed into instructions of Assembler, or in a high level programming language. After feeding such code segment into the corresponding compiler, we would obtain the object code performing Automatic Differentiation for the given initial value problem.

3. Connected problems and dependent variables as terminal values

This package also allows simultaneous integration of several specially related Initial Value Problems called *connectable*. Two Initial value problems for autonomous systems

$u_1' = f_1(u_1, \ldots u_m);$	$u_1 = c_1$	$u_1' = g_1(u_1, \ldots u_m);$	$u_1\|_{u2=d2} = d_1$
$u_2' = f_2(u_1, \ldots u_m);$	$u_2\|_{u1=c1} = c_2$	$u_2' = g_2(u_1, \ldots u_m);$	$u_2 = d_2$
. . . .		. . . .	
$u_m' = f_m(u_1, \ldots u_m);$	$u_m\|_{u1=c1} = c_m$	$u_m' = g_m(u_1, \ldots u_m);$	$u_m\|_{u2=d2} = d_m$

are called *connectible* if both systems have the same number of similarly ordered equations and variables, and the initial values of the second one correspondingly equal to the terminal values of the first, their (hidden) independent variables being not necessarily the same. The Taylor Center makes it possible to deal with several connectible problems simultaneously and therefore to integrate piecewise-analytical solutions defined by different systems of ODE's on different intervals.

There is an important special case of connectible problems, when the solutions (trajectories) of both systems are geometrically *identical*, although the equations are different. In particular, there is an important sub-case when the systems depend on different *independent* variables (the first on u_1 and the second on u_2) and the variables $u_2(u_1)$ and $u_1(u_2)$ are reciprocally inverse functions:

$du_1/du_1 = 1;$	$u_1 = c_1$	$du_1/du_2 = 1/f_2(u_1, \ldots u_m);$	$u_1\|_{u2=d2} = d_1$
$du_2/du_1 = f_2(u_1, \ldots u_m);$	$u_2\|_{u1=c1} = c_2$	$du_2/du_2 = 1$	$u_2 = d_2$
$du_3/du_1 = f_3(u_1, \ldots u_m);$	$u_3\|_{u1=c1} = c_3$	$du_3/du_2 = f_3(u_1, \ldots u_m)/f_2(u_1, \ldots u_m);$	$u_3\|_{u2=d2} = d_3$
. . . .		. . . .	
$du_m/du_1 = f_m(u_1, \ldots u_m);$	$u_m\|_{u1=c1} = c_m$	$du_m/du_2 = f_m(u_1, \ldots u_m)/f_2(u_1, \ldots u_m);$	$u_m\|_{u2=d2} = d_m$

The system for the inverse function $u_1(u_2)$ is used for integration until u_2 reaches a given terminal value, in particular zero (u_2 is the *dependent* variable in the first system, yet *independent* in the second system). Thus, obtaining *exact* zeros of the solution $u_2(u_1)$ without iterations becomes possible (if only points of singularity do not occur on the way).

4. Interface and environment

To provide the user-friendly interactive control over a variety of parameters and modes of integration, the package implements powerful features and objects of the Windows GUI (represented in the Visual Components Library of Delphi). The outline and logical structure of the system are shown in Fig 1.

The system in its current state represents a kernel written in a high level scientific programming language (Object Pascal); the kernel which may be further developed in different directions and for different platforms.

An initial value problem must be entered or loaded from a file into the four edit controls (Fig 2). After successful compilation, users can specify different parameters and modes controlling the integration process (Fig 3), and the curves to be graphed. The integration process itself may be visualized. The result of integration – analytic elements – are stored and used for graphing the curves or to export the numerical solution into other systems. Moreover, the solution may be "played" as a real time dynamic animation of bullets moving along the trajectories in 2D or 3D stereo (viewed through Red/Blue glasses).

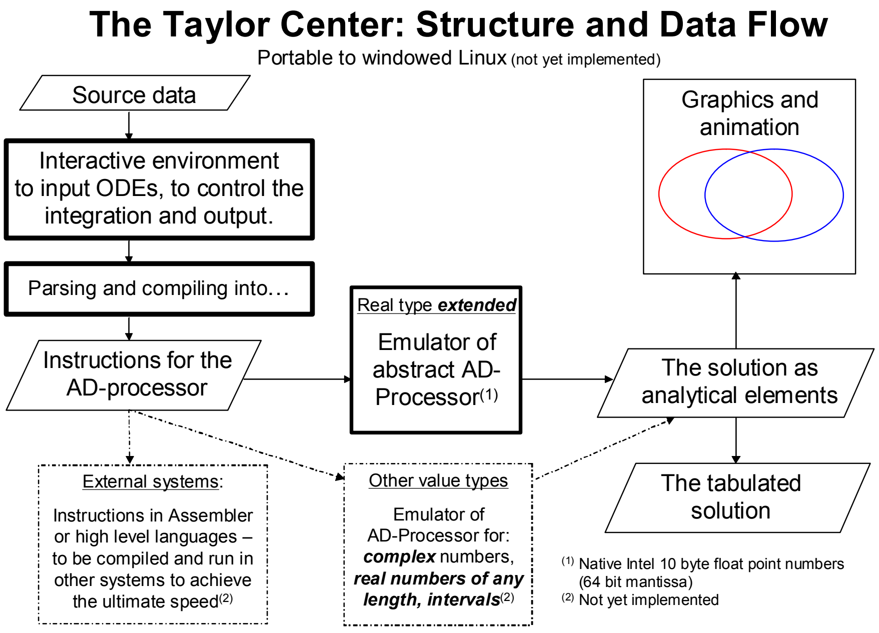
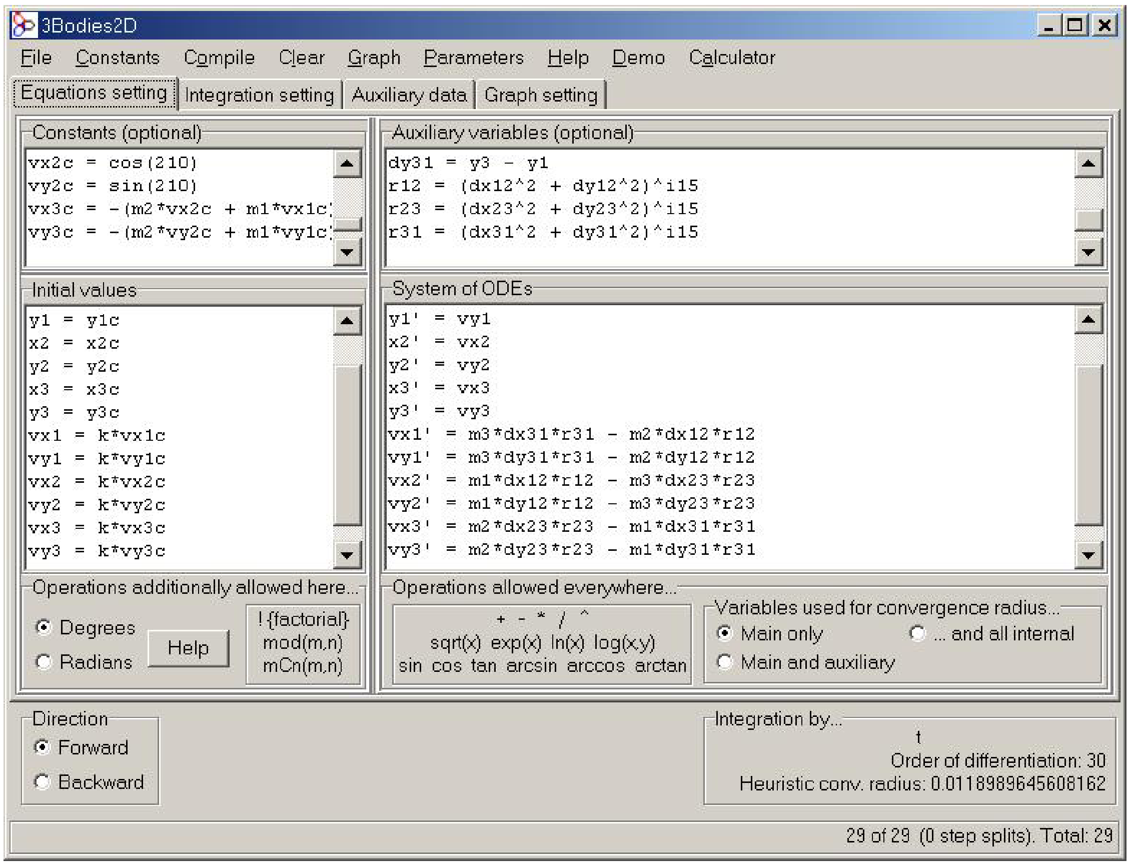

The Taylor Center: Structure and Data Flow
Portable to windowed Linux (not yet implemented)
Source data
Interactive environment to input ODEs, to control the integration and output.
Parsing and compiling into…
Instructions for the AD-processor
Real type *extended*
Emulator of abstract AD-Processor(1)
The solution as analytical elements
Graphics and animation
The tabulated solution
External systems:
Instructions in Assembler or high level languages – to be compiled and run in other systems to achieve the ultimate speed(2)
Other value types
Emulator of AD-Processor for: ***complex*** numbers, ***real numbers of any length, intervals***(2)
(1) Native Intel 10 byte float point numbers (64 bit mantissa)
(2) Not yet implemented

Fig 1. The Taylor Center outline.


3Bodies2D
File Constants Compile Clear Graph Parameters Help Demo Calculator
Equations setting | Integration setting | Auxiliary data | Graph setting
Help

Constants (optional)

```
vx2c = cos(210)
vy2c = sin(210)
vx3c = -(m2*vx2c + m1*vx1c
vy3c = -(m2*vy2c + m1*vy1c
```

Auxiliary variables (optional)

```
dy31 = y3 - y1
r12 = (dx12^2 + dy12^2)^i15
r23 = (dx23^2 + dy23^2)^i15
r31 = (dx31^2 + dy31^2)^i15
```

Initial values

```
y1 = y1c
x2 = x2c
y2 = y2c
x3 = x3c
y3 = y3c
vx1 = k*vx1c
vy1 = k*vy1c
vx2 = k*vx2c
vy2 = k*vy2c
vx3 = k*vx3c
vy3 = k*vy3c
```

System of ODEs

```
y1' = vy1
x2' = vx2
y2' = vy2
x3' = vx3
y3' = vy3
vx1' = m3*dx31*r31 - m2*dx12*r12
vy1' = m3*dy31*r31 - m2*dy12*r12
vx2' = m1*dx12*r12 - m3*dx23*r23
vy2' = m1*dy12*r12 - m3*dy23*r23
vx3' = m2*dx23*r23 - m1*dx31*r31
vy3' = m2*dy23*r23 - m1*dy31*r31
```

Operations additionally allowed here...
[✓ Degrees] Radians
!{factorial} mod(m,n) mCn(m,n)

Operations allowed everywhere...
+ - * / ^
sqrt(x) exp(x) ln(x) log(x,y)
sin cos tan arcsin arccos arctan

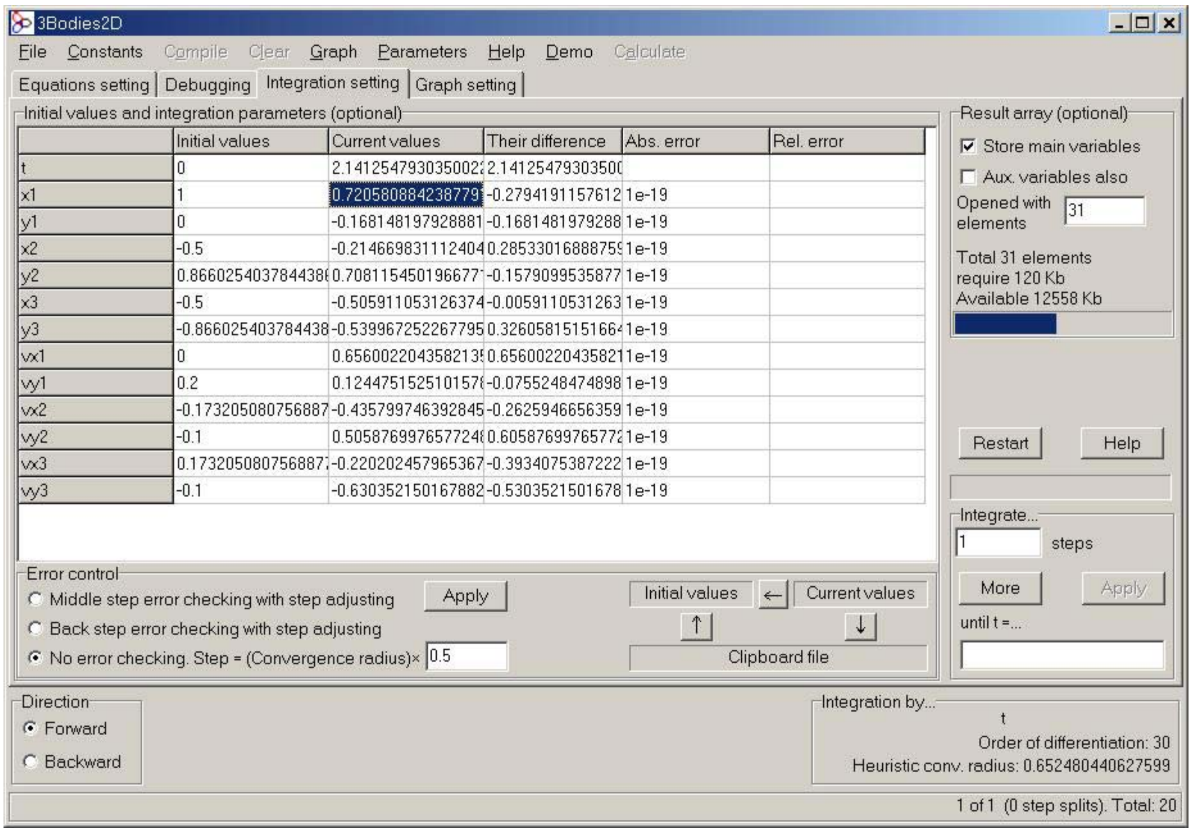
Variables used for convergence radius...
[✓ Main only] ... and all internal
Main and auxiliary

Direction
[✓ Forward] Backward

Integration by...
t
Order of differentiation: 30
Heuristic conv. radius: 0.0118989645608162

29 of 29 (0 step splits). Total: 29

Fig 2. The main panel


3Bodies2D
File Constants Compile Clear Graph Parameters Help Demo Calculate
Equations setting | Debugging | Integration setting | Graph setting
Restart
Help
More
Apply
Initial values
Current values
Clipboard file

Initial values and integration parameters (optional)

	Initial values	Current values	Their difference	Abs. error	Rel. error
t	0	2.141254793035002	2.14125479303500		
x1	1	0.720580884238779	-0.2794191157612	1e-19	
y1	0	-0.168148197928881	-0.1681481979288	1e-19	
x2	-0.5	-0.214669831112404	0.28533016888759	1e-19	
y2	0.8660254037844386	0.708115450196677	-0.1579099535877	1e-19	
x3	-0.5	-0.505911053126374	-0.0059110531263	1e-19	
y3	-0.866025403784438	-0.539967252267795	0.32605815151664	1e-19	
vx1	0	0.656002204358213	0.65600220435821	1e-19	
vy1	0.2	0.124475152510157	-0.0755248474898	1e-19	
vx2	-0.173205080756887	-0.435799746392845	-0.2625946656359	1e-19	
vy2	-0.1	0.505876997657724	0.60587699765772	1e-19	
vx3	0.173205080756887	-0.220202457965367	-0.3934075387222	1e-19	
vy3	-0.1	-0.630352150167882	-0.5303521501678	1e-19	

Result array (optional)
[✓] Store main variables
[ ] Aux. variables also
Opened with elements: 31
Total 31 elements require 120 Kb
Available 12558 Kb

Error control
( ) Middle step error checking with step adjusting
( ) Back step error checking with step adjusting
[✓ No error checking. Step = (Convergence radius)×] 0.5

Integrate... 1 steps
until t =...

Direction
[✓ Forward]
( ) Backward

Integration by... t
Order of differentiation: 30
Heuristic conv. radius: 0.652480440627599

1 of 1 (0 step splits). Total: 20

Fig 3. The integration setting page

Summarizing all the features, with the current version of the Taylor Center users can:

- Specify and study the Initial Value Problems for virtually any system of ODE's in the standard format of explicit 1st order ODE's with numeric and symbolic constants and parameters;
- Perform numerical integration of Initial Value Problems with the high accuracy (up to all 64-bit binary digits of mantissa), while the step of integration remains finite and does not approach zero (because the order of approximation is high: 30 or any higher).
- Obtain the solution as a set of analytical elements – Taylor expansions covering the required domain, exportable to other systems either as is, or in tabular format;
- Study Taylor expansions and the radius of convergence for the solution at all points of interest (with the only limitation that the terms in the series do not exceed the maximum value of about 10^{4932} implied by the 10-byte implementation of the real type "extended");
- Perform integration either "blindly", or graphically visualized; either a given number of steps, or until an independent variable reaches a terminal value, or until a *dependent* variable reaches a terminal value;
- Switch integration between several versions of ODE's defining the same trajectory with respect to different independent variables. For example, it is possible to switch the integration by *t* to that by *x* or *y* in order to reach the terminal value (or zeros) of the dependent variable;
- Integrate piecewise-analytical ODE's;

- Specify different methods of controlling the accuracy and the step size;
- Specify accuracy for individual components either as an absolute or relative error tolerance, or both;
- Graph color curves (trajectories) for any pair of variables of the solution up to 7 on one screen, either as plane projections, or as 3D stereo images to be viewed through anaglyphic (Red/Blue) glasses. The 3D cursor with audio feedback (controlled by a conventional mouse) enables "tactile" exploration of the curves literally "hanging in thin air";
- "Play" dynamically the near-real time motion of bullets along the computed trajectories either as 2D or 3D stereo animation;
- Explore several meaningful examples supplied with the package such as the problem of Three or Four Bodies, "Choreography". Symbolic constants and expressions make it possible to parameterize the equations and initial values trying different initial configurations of special interest.

The current kernel version may be further developed in different directions (Fig 1):

- To implement other types of arithmetic (complex numbers, intervals);

- To generate compileble code in Assembler or in a high level language;

- To simultaneously integrate an *array* of Initial Value Problems;

- To compute derivatives by parameters.

- To implement special loops and array variables (to automate expressions of high complexity in the right hand sides such as sums of many terms).

References

[1] Chang Y. F., Corliss G.: ATOMFT: Solving ODEs and DAE Using Taylor Series. Computers Math. Applic. Vol. 28, No. 10-12, 1994. pp. 209-233.
[2] Chang Y. F. ATOMFT User Manual. 1986. Claremont McKenna College, Claremont, CA, 91711
[3] Gofen, A. M.: Taylor Method of Integrating Ordinary Differential Equations: the Problem of Steps and Singularities. Cosmic Research, Vol. 30, No. 6, pp. 581-593 (1992)
[4] Gofen, A. M.: ODEs and Redefining the Concept of Elementary Functions. In: P.M.A. Sloot et al. ICCS 2002, LNCS 2329. Springer-Verlag, Berlin 2002.
[5] Gofen, A. M.: The Taylor Center for PCs. In: P.M.A. Sloot et al. ICCS 2002, LNCS 2329. Springer-Verlag, Berlin 2002.
[6] Gofen, A. M.: The Taylor Center Demo for PCs and the User Manual. http://www.ski.org/gofen/TaylorMethod.htm