Teaching Mathematics and Statistics in Sciences - IPA HU-SRB/0901/221/088
Interesting Mathematical Problems in Sciences and Everyday Life - 2011

Investigation of impulsive systems with *Mathematica*

János Karsai
Department of Medical Physics and Informatics
University of Szeged, Hungary
Korányi fasor 9, Szeged, 6720 Hungary
karsai@dmi.u-szeged.hu
http://www.model.u-szeged.hu

Let a motion be described by a differential equation, but when some conditions hold an instantaneous effects change the position and/or velocity of the moving body. Such systems are called impulsive. They are mathematical models of several phenomena in all fields of science. Since their theoretical study is quite complicated (combination of differential or difference equations), computer-aided methods are particularly important in both theoretical studies and applications. The author developed packages in *Mathematica* to solve and visualize different types of impulsive systems. In this paper, we give several examples to demonstrate the most important properties of impulsive systems and the main features of the developments in *Mathematica*. See the web-page http://www.model.u-szeged.hu/index.php?action=edoc for [13] to find details. The package is an attachment of this paper.

1. Introduction

In everyday life, it happens very often that under some conditions, an effect causes changes under very short time or either instantaneously. In this case, we speak about impulses. Continuous and impulsive effects can appear together, they are called impulsive systems. For example, we can meet such systems at discrete or piecewise control, repeated drug administration, harvesting, pulse acceleration (control) of missiles, etc. Some examples can be found in our references. Due to the complicated mathematical description, the experimental study is useful and informative, and sometimes it is the only way to investigate the properties of such phenomena.

In order to help mathematical research and applications in sciences, the author developed tools [13,14] in *Mathematica* to visualize and solve different types of impulsive systems. Note that the novelty of the mathematical theory and the arising technical difficulties causes that such packages are not developed yet in any other mathematical systems. The package has been continuously updated, and now essentially improved in *Mathematica* 8. Some special features are: numerical solution of impulsive systems; visualization of impulse fields; plotting the jump surfaces; functions for phase mappings.

This paper is based on [13] and [14]. Here we apply our developments to study some problems of various fields, as well as to introduce the reader to the theory, and some qualitative methods.

First, let us consider impulsive systems with fixed impulse instants. Then, we deal with general systems in which both the impulse instants and the impulses depend on the state variables. As special cases, we consider autonomous systems. Note that the complete *Mathematica* code is attached to the paper, or the current version can be downloaded from te given webpage. To run the statements loading the packages is assumed (see the Appendix for the summary).

2. Systems with fixed instants of impulses

Theory [13]

Let the infinite sequence $\{t_i\}$ be monotone increasing and unbounded. Let the function $f : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n$ be continuous, except at $t = t_i$ $(i = 1, 2, ...)$ where it is continuous from left, and it can have discontinuity of the first kind. In addition, let the functions $I_i : \mathbb{R}^n \to \mathbb{R}^n$ be continuous. Then, the system

$$\begin{aligned} x' &= f(t, x), \text{ if } t \neq t_i, \\ x(t_i + 0) &= I_i(x(t_i - 0)), \ (i = 1, 2, 3, ...) \end{aligned} \tag{1}$$

is an impulsive system (IDE) with impulses at fixed instants. Sometimes, it is comfortable to denote the impulses by $I(t_i, x)$ $(I : \{t_i\}\times\mathbb{R}^n \to \mathbb{R}^n)$.
Consider now an initial value

$$x(t_0 + 0) = x_0. \tag{2}$$

The solution $x(t; t_0 + 0, x_0)$ of the initial value problem (i.v.p.) (1)-(2) can be obtained as follows. Assuming $t_k \le t_0 < t_{k+1}$, take the maximal solution $x(t)$ of the initial-value-problem

$$\begin{aligned} x' &= f(t, x), \\ x(t_0 + 0) &= x_0 \end{aligned} \tag{3}$$

defined on the interval $[t_0, T)$. If $T \le t_{k+1}$, then the problem is solved, if $T > t_{k+1}$, consider the solution on $[t_0, t_{k+1}]$, and continue it by the solution of the i.v.p.

$$\begin{aligned} x' &= f(t, x), \\ x(t_{k+1} + 0) &= I_{k+1}(x(t_{k+1} - 0)). \end{aligned} \tag{4}$$

Through the paper we assume that the solutions of the above ininitial value problems are unique. We can see that both ordinary differential equations and difference equations are special cases of such systems by taking $I_k(x) = x$ and $f(t, x) = 0$, respectively. Hence, the properties of continuous and discrete effects can appear together. In addition, the impulse instants are not necessarily uniformly distributed, and it results in rather unusual properties. For theoretical details, see [2,3,13,19].

We emphasize that the initial condition is always $x(t_0 + 0) = x_0$ (right-hand-side limit) and the solution begins with the solution of the ODE part. It is obvious that the local existence and local continuability is determined by the ODE part. If the solutions of the ODE are continuable to infinity, then the solutions of the IDE can be continued to infinity providing that the sequence $\{t_k\}$ has no finite accumulation point.

A new phenomenon appears if an impulse function $I_k(x)$ is not one-to-one. The backward continuability is not unique, since there exists at least two different solutions becoming identical on $[t_k, \infty)$.

The equilibria of (1) are the constant solutions, i.e., the solutions of the system $f(t, x) = 0$, $I_k(x) = x$ $(k = 1, 2, ...)$.

In order to help the experimental study of system (1), we need to work with the continuous and discrete effects, and hence we need the following tools:

- Describe the IDE.
- Vector fields to show the directions of solutions of the ODE part.
- Impulse fields to show the jumps by the impulses.
- Solve the initial value problems of the IDE.
- Visualize the solutions, trajectories.

As known for ODE's, the direction field $\{1, f(t, x)\}$ gives the tangent vectors of the integral curves in the extended phase space $R\times R^n$. We can also introduce the impulse field $\{0, I(t_i, x) - x\}$ in $R\times R^n$ that gives the jumps by the impulses. We emphasize that both the direction and the length of the vectors are essential. If $f(t, x) = f(x)$ (the ODE is autonomous), the tangent vectors of the trajectories of the ODE part are given by the

vector field $f(x)$ in the phase space R^n. Analogously, if $I_{k+1}(x) = I(x)$, then the impulse vectors $I(x) - x$ can also be shown in the phase space. Note that although the impulse is independent of the time, the system is not autonomous, since the behavior depends on the instants t_i.

Next, we consider the simple example of an impulsively perturbed harmonic oscillator to present the tools developed, as well as we consider the experimental aid of some qualitative methods for impulsive systems with fixed impulse instants.

A damped harmonic oscillator under impulse effect

Consider the harmonic oscillator, and let the speed be impulsively modified periodically. We obtain the following system

$$\begin{aligned} x' = y,\ y' = -x,\ \text{if } t \neq t_i, \\ x(t_i + 0) = x(t_i - 0),\ y(t_i + 0) = I_i\,(y(t_i - 0)), \end{aligned} \tag{5}$$

where $t_i = i\,T$. The case of continuous damping effect would be $x' = y,\ y' = -x - a\,y$. Now, let the oscillator be damped impulsively in the form

$$y(t_i + 0) = b_i\,y(t_i - 0)$$

where $0 \le b_i \le 1$. If $b_i = 0$, then the speed becomes zero (resulting nonuniquiness problems of the i.v.p, that cannot happen for ODE'S), and no impulse works if $b_i = 1$. Note that the oscillating body is "kicked" back by $b_i < 0$, and for $|b_i| > 1$ the speed is impulsively increased. The detailed theoretical study can be found in [5-12]. Now, consider the damping case step by step.

The system

The right-hand-side of the ODE is written as the vector fields require.

```
var = {x, y};
rhs = ( 0  1
       -1  0 ).var;
```

The behavior of the harmonic oscillatior is know, so we do not consider it separately.

Let $T = 1$ and `tn` be the sequence of impulse instants. The impulses are given as follows:

```
T = 1.; b = 0.7;
tn = Table[n T, {n, 1, 100}];
FixedImpulse[n_, tn_List, u_List] := {u[[1]], b * u[[2]]};
```

Here `u` is the phase variable. This parameter structure is assumed by every commands. The window parameters are

```
t0 = 0; t1 = 6;
x1 = y1 = -1.; x2 = y2 = 1.;
```

Before doing anything, load the needed packages.

```
Needs["VectorFieldPlots`"];
```

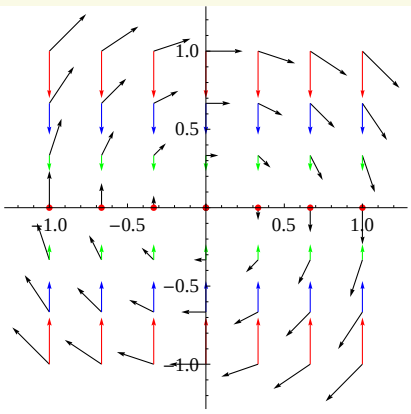
```
SetDirectory["FileName" /.NotebookInformation[EvaluationNotebook[]] /.
   FrontEnd`FileName[d_List, nam_, ___] :> ToFileName[d]];
<< package//impulseplot.m
<< package//idesolve.m
```

```
<< package//phase2d.m
```

Vector fields

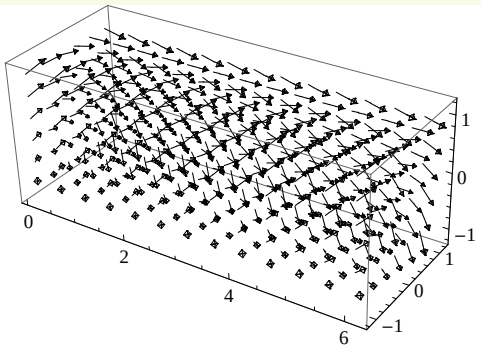
Since the right-hand-sides are time-independent, we can plot both the vector field of the ODE and the impulse field in the $\{x, y\}$ phase plane.

```
Fields2D = DoubleFieldPlot[{rhs, FixedImpulse[1, tn, var] - var},
  {x, x1, x2}, {y, y1, y2}, Axes → True, PlotPoints → 7]
```

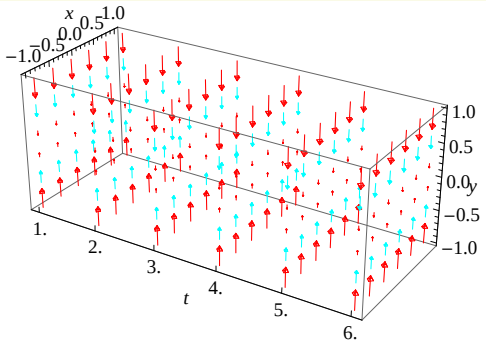


The black arrows belong to the vector field of ODE, the colored ones show the impulses at points $\{t_i, x, y\}$, where $t_i \in$ `tn`. The behavior of the trajectories cannot be conjectured by this figure, since the impulse instants are not shown here. To have more information, we need the fields in the extended phase space.

```
dx = dy = 0.4;
FieldODE3D = VectorFieldPlot3D[Flatten[{1, rhs}], {t, t0, t1, T / 2},
  {x, x1, x2, dx}, {y, y1, y2, dy}, Axes -> True, VectorHeads -> True]
```

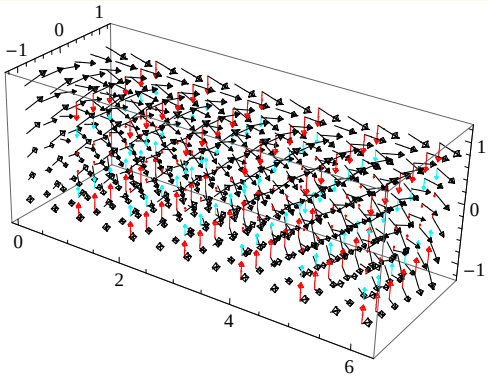


```
FieldIMP3D = FixedImpulseFieldPlot3D[FixedImpulse, tn, {t, t0, t1},
  {x, x1, x2, dx}, {y, y1, y2, dy}, ColorFunction -> (Hue[#1] &)]
```



The impulse vectors are plotted at the elements of `tn` between `t0` and `t1`. The meaning of the parameters of the statement `FixedImpulseFieldPlot3D` is obvious. Note that functions for scalar systems are also available (`FixedImpulseFieldPlot`, `AnimateImpulse`, `StackImpulse`).

```
Fields3D = Show[FieldODE3D, FieldIMP3D]
```



Solving the system

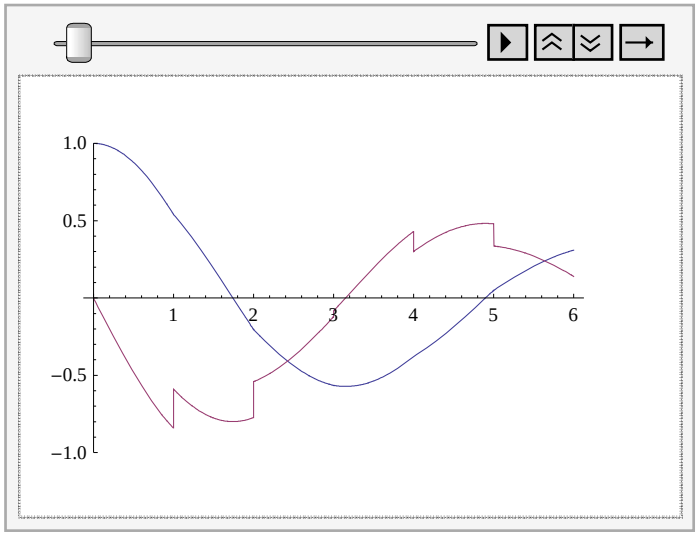
Now, let us solve the system. The initial time is `t0`, and the solutions start from the unit circle of the phase space. The solver command is `IDESolve`, that creates InterpolatingFunction objects. It solves the system intervalwise on the intervals $(t_i, t_{i+1}]$ using `NDSolve`, and the obtained `InterpolatingFunction` objects are joined at each step. This function requires only the right-hand-side of the system (as the vector field plots), and can find the solutions for several initial values. The result is a list of parametric curves.

```
IC = Table[ N[{Cos[u], Sin[u]}], {u, 0, 2 π, π/8}];
Traj[t_] = IDESolve[rhs, var, tn, FixedImpulse, IC, {t, t0, t1}];
```

Visualizing the solutions

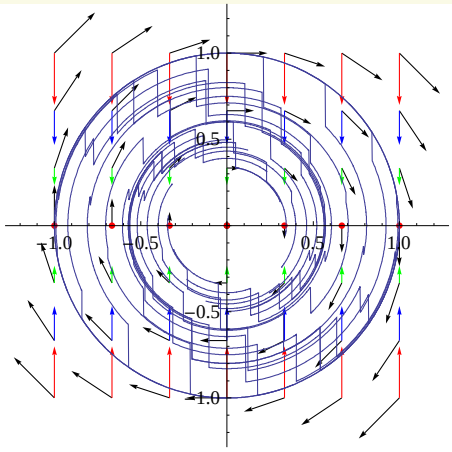
Basic visualizations of the solutions can be done by combinations of Plot, ParametricPlot and ParametricPlot3D commands. The coordinates are well visible by using the following statement:

```
Map[Plot[#, {t, t0, t1}, PlotRange -> {-1, 1}, ImageSize → {250, 180}] &,
  Traj[t]] // ListAnimate
```



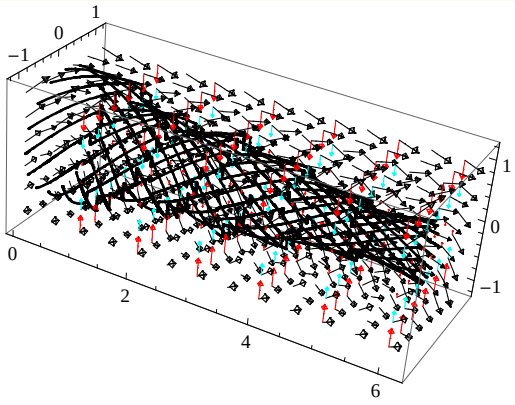
Obviously, the continuous curve is $x(t)$, the discontinuous is $y(t) = x'(t)$. Let us plot the trajectories together with the fields:

```
PlotTraj = ParametricPlot[Traj[t], {t, t0, t1}];
Show[Fields2D, PlotTraj]
```



Let us consider the integral curves and the vector fields in 3D:

```
PlotTraj3D = ParametricPlot3D[Map[Prepend[#, t] &, Traj[t]],
   {t, t0, t1}, PlotStyle → {Thickness[0.005]}, ImageSize → {250, 200}];
Show[Fields3D, PlotTraj3D]
```



We can observe immediately that if $y(t_i) = 0$, the damping is ineffective. It is an important fact in the research on asymptotic stability properties of this system (see [10,12]). In the special case $T = \pi$, $t_i = i\,\pi$, there exists a solution not tending to zero, if $T < \pi$, the zero solution is attractive [6].

Asymptotic behavior, the method of auxiliary functions

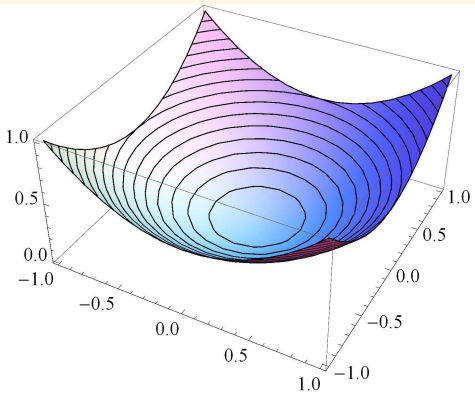
Since the most of the systems are theoretically unsolvable, qualitative methods are of importance to investigate the properties of the solutions without explicitly knowing them. The method of auxiliary functions, the second method of Liapunov, is one of the most effective qualitative methods in studying asymptotic behavior (see, for example, [2], [13], [18] and [19]). The basis of the method is very simple. Here, we only show the general idea on our example, how the method can be aided by *Mathematica* for impulsive systems.

The total energy of the system (5) is

```
V[{x_, y_}] := 1/2 {x, y}.{x, y}
```

The graph of V can be plotted:

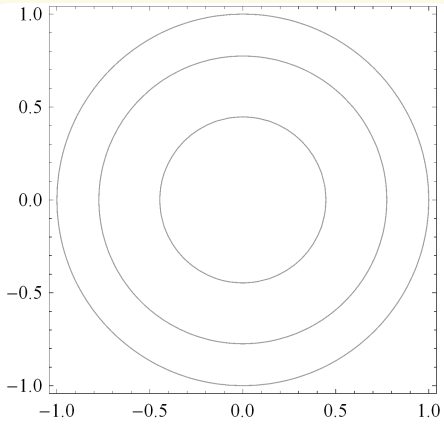
```
Plot3D[V[var], {x, x1, x2}, {y, y1, y2},
 BoxRatios → Automatic, MeshFunctions → {#3 &}]
```



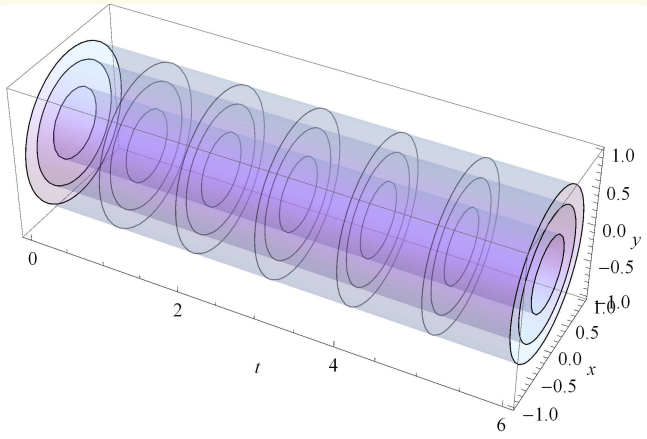
Since the energy function is positive definite, the contour lines in the phase space R^2 are closed around the origin (circles), and cylinders around the axis t in the extended phase space.

```
level = {0.1, 0.3, 0.5};
```

```
ContourV = ContourPlot[V[var], {x, x1, x2}, {y, y1, y2},
  Contours → level, ContourShading → False]
```




```
ContourV3D =
 ContourPlot3D[V[{x, y}], {t, t0, t1}, {x, x1, x2}, {y, y1, y2}, Contours → level,
      ContourStyle -> {Opacity[0.2]},
      MeshStyle -> {Opacity[0.5]},
      MeshShading -> {Automatic},
      MeshFunctions -> {#1 &}, Mesh → {Take[tn, 5]},
      AxesLabel -> {t, x, y}, BoxRatios → Automatic]
```



We investigate how $V(x,\ y)$ changes along the solutions, i.e., we study $V(x(t),\ y(t))$. For example, if $V(x(t),\ y(t))$ is decreasing, the trajectories cross the contour lines from outside to inside. If it is the case for every small enough contour lines, the zero solution is stable. For the details of the stability properties and theorems, see, for example [2,13,18,19]. The main point of the method is that the change of the function $V(x(t),\ y(t))$ can be followed by the derivative along the solutions and by the jumps at the impulses without knowing the solutions themselves.

As it is well known in the qualitative theory of differential equations, for $t \neq t_i$ $(i = 1,\ 2,\ ...)$, $V(x(t),\ y(t))$ is differentiable for $t \neq t_i$ $(i = 1,\ 2,\ ...)$ and

$$\frac{d}{d\,t}\,V(x(t),\ y(t)) = \text{grad}\,V\,(x,\ y)\,.\,\{x',\ y'\} = \{x,\ y\}\,.\,\{y,\ -x\} = 0,$$

i.e., the scalar product of grad V (normal vector of the contour lines of V) and the tangent vector of the trajectories is zero, since the harmonic oscillator preserves energy.

```
dV = D[V[var], {var}].rhs
```

```
0
```

On the other hand, at the instants t_i $(i = 1, 2, ...)$, the change of the energy is discrete, so we have to calculate the jump of the energy:

$$V(x(t_i + 0),\ y(t_i + 0)) - V(x(t_i - 0),\ y(t_i - 0)).$$

In our example, we have:

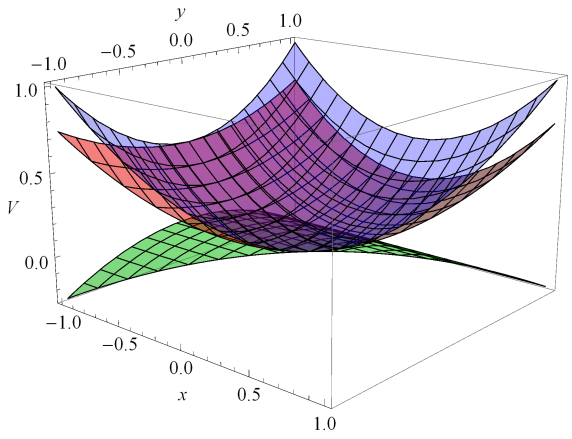
```
Simplify[V[FixedImpulse[1, tn, var]] - V[var]]
```

```
-0.255 y^2
```

which shows that the impulses decrease the energy whenever $y(t_i - 0) \neq 0$. The theoretical study would lead very far, so we consider only the visualization tools in *Mathematica.* Observe that in our case both the continuous and impulsive effects change the energy in the same direction. The behavior is more complicated if they work against each other. This will be the case in section 6, where we study the physical pendulum with external forces.

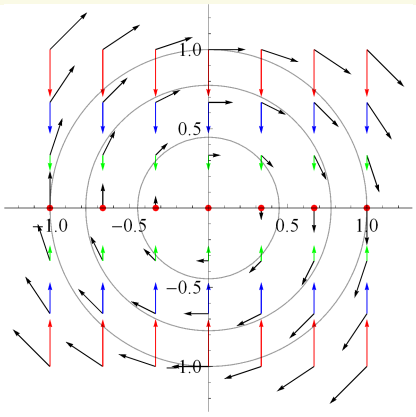
Let us show the functions $V(x(t_i + 0),\ y(t_i + 0))$, $V(x(t_i - 0),\ y(t_i - 0))$ as well their difference at t_1, colored by red, blue and green, respectively. In this special example, the impulses are independent of i.

```
Plot3D[{V[{x, y}], V[FixedImpulse[1, tn, {x, y}]],
  V[FixedImpulse[1, tn, var]] - V[var]}, {x, x1, x2}, {y, y1, y2},
    BoxRatios → Automatic, PlotStyle →
  {{Blue, Opacity[0.3]}, {Red, Opacity[0.5]}, {Green, Opacity[0.5]}},
    RegionFunction → ((#2 > #1) &), AxesLabel → {x, y, V}]
```



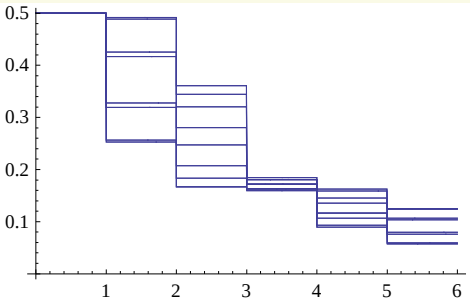
This figure agrees with the relation between the trajectories and contour lines. To really visualize the method, we can show the contour lines of the energy function and the vector fields.

```
Show[Fields2D, ContourV]
```



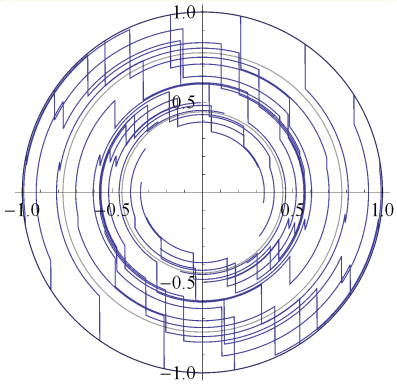
Note that these plots can be also applied to localize experimentally the trajectories in other cases, for example to find limit cycles. If the solutions are known, we can obtain more attractive figures. To plot the energy along the solutions is very simple.

```
Plot[Map[V[#] &, Traj[t]], {t, t0, t1}]
```



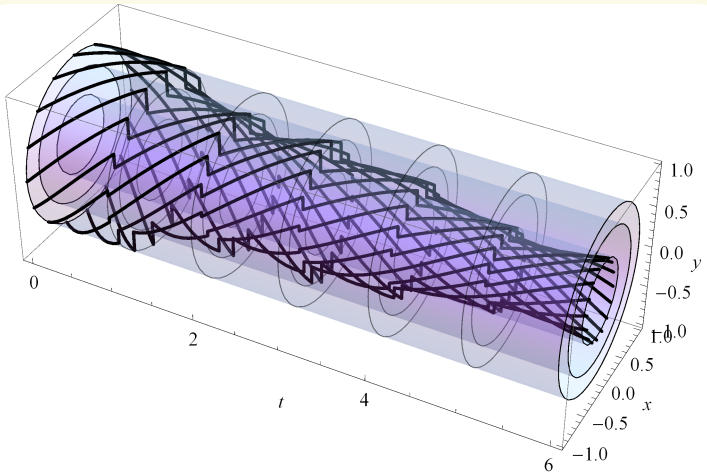
The contour lines of the energy function and trajectories are

```
Show[PlotTraj, ContourV]
```



Finally, consider the behavior of the solutions in the extended phase space.

```
Show[ContourV3D, PlotTraj3D, PlotRange → All, BoxRatios → Automatic]
```



The figures help to lead to the conjecture that every solution tends to zero as $t \to \infty$. This is really true as proved in [5]. The proof for this simple example is not too difficult, but we have to note that the problem in general is far from trivial. We recommend to use our programmes above to investigate the following systems:

◆ Continuously damped oscillator

Either for the damped equation

$$x'' + a(t)\,x' + x = 0, \quad a(t) \geq 0, \tag{6}$$

or for the analogous impulsive system (5) with $0 \leq b_i \leq 1$, there is no necessary and sufficient condition proved yet for the asymptotic stability of the zero solution.

◆ Oscillators with nonlinear elastic force

The problem is even more complicated for nonlinear oscillators. In particular, we recommend the reader to do experiments for the system

$$\begin{gathered} x'' + |x|^{\alpha}\,\mathrm{sign}(x) = 0, \ \alpha > 0, \ \alpha \neq 1, \\ x'(n\,T + 0) = b_i\,x'(n\,T - 0), T > 0, b_i \in R. \end{gathered} \tag{7}$$

It can be proved that this system has infinitely many periodic solutions [11], what is not true for the linear case.

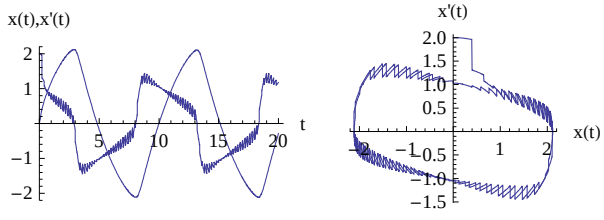
◆ Oscillators with nonlinear damping

If the impulse is nonlinear, such as

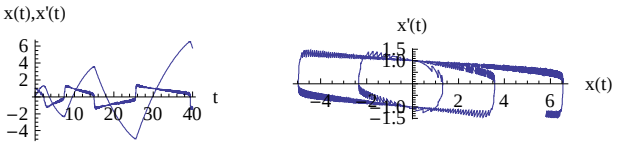
$$x'(t_i + 0) = |x'(t_i - 0)|^{\beta}\,\mathrm{sign}(x'(t_i - 0)), \tag{8}$$

we can obtain relaxation oscillations [13] an analog of the Rayleigh equation. The qualitative properties of such impulses are not investigated yet. For illustrations, let us consider the trajectories of this system with different settings:

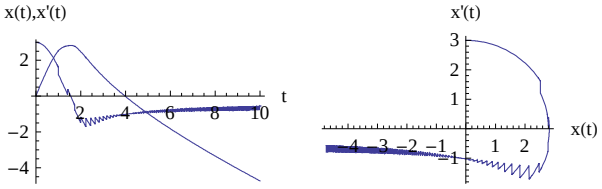
– $\beta = 0.4;\ t_i = 0.2\,i$



– $\beta = 0.4$ and $t_i = i^{0.6}$



– $\beta = 0.4;\ t_i = (i)^{0.5}$



3. Systems with general impulses

Theory [13]

Now, consider general impulses, which appear when the integral curve of solutions of the system

$$x' = f(t, x),\ (f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n).$$

meet hypersurfaces in the extended phase space. More precisely, let the scalar fields $S_1,\ S_2,\ ...,\ S_i,\ ... : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ be continuous, and the functions (impulses) $I_1,\ I_2,\ ...,\ I_i,\ ... : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be given. The equations $S_i(t, x) = 0$ define surfaces in $\mathbb{R} \times \mathbb{R}^n$. Let us assume that if $S_i(t, x) = S_j(t, x) = 0$ $(i, j = 1, 2, ...)$, then $I_i(t, x) = I_j(t, x)$, i.e., the impulse system must be well-defined. Consider a solution $x(t)$ of the ODE. If $S_i(\tau, x(\tau)) = 0$ at a moment τ, then let us continue $x(t)$ with the solution of the ODE with the initial value $I_i(\tau, x(\tau))$, i.e., $x(\tau + 0) = I_i(\tau, x(\tau - 0))$. By this method we defined a general impulsive system with state-dependent impulses:

$$\begin{aligned} \boldsymbol{x' = f(t, x),\ \text{if}\ S_i(t, x(t)) \neq 0;} \\ \boldsymbol{x(\tau + 0) = I_i(\tau, x(\tau - 0)),\ \text{if}\ S_i(\tau, x(\tau)) = 0,\ (i = 1,\ 2,\ 3,\ ...).} \end{aligned} \tag{9}$$

Assume that f is continuous if $S_i(t, x) \neq 0$, and it has discontinuity of first kind if $S_i(t, x) = 0$ $(i = 1,\ 2,\ 3,\ ...)$, there exist the "half-sided" limits $\lim_{s \to t, u \to x, S_i(s,u) < 0} f(s, u)$ and $\lim_{s \to t, u \to x, S_i(s,u) > 0} f(s, u)$. The solutions are left-continuous, piecewise differentiable functions, having discontinuities of first kind at the instants of impulses.

It is obvious that the systems with fixed instants of impulses can be obtained by giving $S_i(t, x) = t - t_i$. If $S_i(t, x) = S_i(x)$ $(i = 1,\ 2,\ ...)$, then we obtain autonomous impulses. For this caase we will consider an example in section 4.

Note that a function S_i can define infinitely many instants. For example, $t_i = i\,T$ can be given by $\sin\left(\frac{\pi t}{T}\right) = 0$. Hence, $S_i(t, x) = 0$ can mean several, even infinitely many surfaces. The impulses $I_i(t, x)$ $(i = 1,\ 2,\ ...)$ can also be expressed together. Hence, the system (9) can be written in the form

$$\begin{aligned} \boldsymbol{x' = f(t, x),\ \text{if}\ S(t, x(t)) \neq 0,} \\ \boldsymbol{x(\tau + 0) = I(\tau, x(\tau - 0)),\ \text{if}\ S(\tau, x(\tau)) = 0.} \end{aligned} \tag{10}$$

For technical reason, we keep the original form (9) .

Equilibria on $[t_0, T)$ are the constant solutions. A point $\bar{x} \in \mathbb{R}^n$ is an equilibrium if and only if $f(t, \bar{x}) \equiv 0$ and $I_k(\tau, \bar{x}) = \bar{x}$ for every k, provided $S_k(\tau, \bar{x}) = 0$ and $t, \tau \in [t_0, T)$.

Properties of the surfaces $S_k(\tau, x) = 0$ are of importance. For example, if $S_k(\tau, x) = 0$ is invariant for the ODE, the integral curves can remain on it, and hence the next impulse instant cannot be determined.

In addition, continuability of the solutions essentially depends on the impulses and the shape of the surfaces $S_k(\tau, x) = 0$. A solution can meet the surface $S_k(\tau, x) = 0$ several times, such that the solution is beaten back from the surface. This is the case if $S_k(\tau, x(\tau)) = 0$, $S_k(t, x(t)) < 0\ (> 0)$ for $t < \tau$ and also $S_k(\tau, I_i(\tau, x(\tau - 0))) < 0\ (> 0)$. Graphically, the solution cannot go to the other side of the surface. This phenomenon cannot happen for impulses at fixed instants.

We can study system (9) in *Mathematica* similarly to systems with fixed impulse instants, but the implementations of the commands are completely different:

- Describe the ODE as before.
- For impulses, needed to give the scalar fields $S_k(t, x)$ $(\mathbb{R}\times\mathbb{R}^n \to \mathbb{R})$ and the impulse mappings $I_i(\tau, x)$ $(\mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n)$.
- Visualize the Impulse (jump) mappings, i.e., the image of the surfaces $S_k(t, x) = 0$ by the impulse $I_i(t, x)$.
- Solve the initial value problems of the IDE.
- Visualize the solutions, trajectories.

Now, we will illustrate the main features, and consider an example with strange behavior.

A strange scalar system

Consider the following system [2]

$$\begin{gathered} \boldsymbol{x' = 0,\ \text{for}\ \ S\,(t,\ x(\tau - 0)) \neq 0,} \\ \boldsymbol{x(\tau + 0) = x(\tau - 0)^2\,\text{sign}(x(\tau - 0)),\ \ \text{for}\ \ S\,(t,\ x(\tau - 0)) = 0,} \end{gathered} \qquad \textbf{(11)}$$

where $S\,(t,\ x) = \sin(\pi(t - x))$, if $|\,x\,| < 2$. This system can be also written in the form

$$\begin{gathered} x' = 0,\ \ \text{for}\ \ t \neq \tau_i(x), \\ x(\tau + 0) = x(\tau - 0)^2\,\text{sign}(x(\tau - 0)),\ \ \text{if}\ \ t = \tau_i(x)\ \ (i = 1,\ 2,\ 3,\ \ ...), \end{gathered}$$

where $\tau_i(x) = x\ +\ 6\,i$, if $|\,x\,| < 2$.

Let us define the system in *Mathematica.* The variables, the right-hand-side and the initial conditions are given as before.

First, load the needed packages.

```
SetDirectory["FileName" /.NotebookInformation[EvaluationNotebook[]] /.
   FrontEnd`FileName[d_List, nam_, ___] :> ToFileName[d]];
<< package//impulseplot.m
<< package//iderksolve.m
```

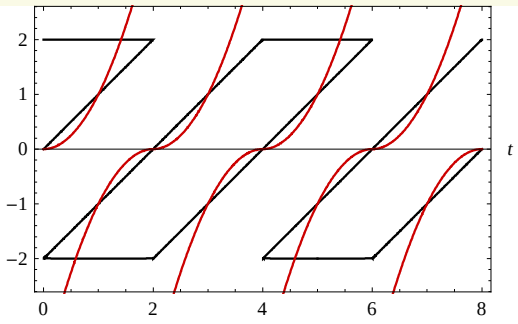
```
var = {x}; rhs = {0}; t0 = 0; t1 = 8; dt = 0.01; x1 = -2.5; x2 = 2.5;
IC = {{-2}, {-1}, {-1.1}, {-0.9}, {1}, {0.9}, {1.1}, {2.}};
```

Impulses are given in a finite list $\{S_i, I_i, d_i\}$, i.e., $\{\{S_1, I_1, d_1\}, \{S_2, I_2, d_2\}, \ ...\}$, where $d_i \in \{0, 1\}$ is a technical parameter, needed for the solver routine. There is a technical restriction: the impulse functions must be given coordinatewise in lists.

```
S[t_, x_] := Which[Abs[x] < 2, Sin[1/2 π (t - x)], True, 1];
Impulse = {{S[t, x], {x^2 Sign[x]}, 0}};
```

We can visualize the impulse effect with `JumpPlot` on the "surface" $S(t, x) = 0$, i.e., the points $\{t, x\}$ satisfying $S(t, x) = 0$, and the image $\{t, I(t, x)\}$ together. Obviously, $x = 1$ is a fixed point, we will find beating effects for $x \in (-1, 0)$ and $x \in (1, 2)$. The third element (important later in the solving process) in the impulse list is now used to color the contour lines (black and blue for $d = 0$ and $d = 1$, respectively). For 2D systems, the analogous `JumpPlot3D` statement can be applied.

```
JumpPlot[Impulse[[1]], {t, t0, t1}, {x, x1, x2}, AspectRatio → Automatic]
```



Now, solve the system using the `IDERKSolve` statement. The solver algorithm and some technical issues are descibed below:

The meaning and structure of the parameters will be obvious below. In solving the system to find the next instant when the impulses occur, we have to solve the equation `Si(t, x(t)) = 0` while $x(t)$ is being solved simultaneously. Hence, this equality should be verified in the solving process of the ODE part. It can be done by EventLocator in NDSOLVE. Instead, to solve the ODE between two impulses, we use a Runge-Kutta method with fixed step size (the implementation by R. Maeder [16]), and instead of solving `Si(τl, x(τl)) = 0`, we verify the change of sign of the functions `Si(t, x(t))` stepwise. Note that wrong impulses can appear if the functions S_i are not continuous!

Let the last step be $t = s_l$. The solution is of the form $\{\{s_0, x_0\}, \{s_1, x_1\}, \ldots, \{s_l, x_l\}, \{s_{l+1}, x_{l+1}\}\}$. If every product $S_i(s_l, x_l)\, S_i(s_{l+1}, x_{l+1})$ is positive, we continue the solution of the ODE from $\{s_{l+1}, x_{l+1}\}$ and find $\{s_{l+2}, x_{l+2}\}$. If for one $S_i(t, x(t))$ we obtain $S_i(s_l, x_l)\, S_i(s_{l+1}, x_{l+1}) \le 0$, then $S_i(\tau_l, x(\tau_l)) = 0$ for some $\tau_l \in [s_l, s_{l+1}]$, and the corresponding impulse $I_i(\tau_l, x(\tau_l))$ should be applied. We do not use finer approximation, only apply $I_i(s_l, x_l)$ or $I_i(s_{l+1}, x_{l+1})$ depending $d_i = 0$ or $d_i = 1$ in the description of the impulse, and the solution of the ODE will be continued from the list $\{\{s_0, x_0\}, \{s_1, x_1\}, \ldots, \{s_l, x_l\}, \{s_l, I(s_l, x_l)\}\}$ or $\{\{s_0, x_0\}, \{s_1, x_1\}, \ldots, \{s_l, x_l\}, \{s_{l+1}, I(s_{l+1}, x_{l+1})\}\}$, respectively. Different settings can result in different solutions. In the case $d_i = 0$ (impulse at s_l) the solver can cause wrong beating. For $d_i = 1$, the solver can avoid real beating, since the solution can continue on the "other side" of the surface $S_i(t, x) = 0$. Note that if the functions $S_i(t, x) = S_i(t)$, i.e., the case of fixed instants of impulses, the setting must be $d_i = 1$!

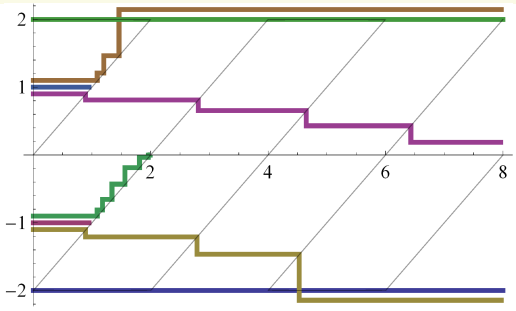
The result is a list `{sol1,...,solM}`, where `soli` is the solution belonging to the i^{th} initial value. The structure of each solution is `{{t0, x1(t0), ..., xn(t0)}, ..., {ti, x1(ti), ..., xn(ti)} ...}`.
Now, turn back to our example, and solve the system with $d = 0$ (the system will have beating effect, and we must not ignore it).

```
sol = IDERKSolve[rhs, Impulse, var, IC, {t, t0, t1, dt}]
```

An underflow can occur for the solution with $x(0) = -0.9$ because of its unusual behavior (see the figure and explanations below).
Plot the solutions and the set $S(t, x) = 0$ together:

```
plt1 = ListPlot[sol, Joined → True, PlotStyle → {Thickness[0.01]}];
plt2 = ContourPlot[S[t, x], {t, t0, t1}, {x, x1, x2},
```

```
    Contours → {0}, PlotPoints → 100, ContourShading → False];
Show[plt1, plt2, PlotRange → All]
```



We can see that the behavior of the solutions mainly depend on the initial values. In particular, it can be proved that:

- If $|x_0| \geq 2$, there is no impulse effect on the solution.
- If $1 < x_0 < 2$, then there are finite impulses on the same "surface" (beating effect).
- If $-2 \leq x_0 < -1$, then there are finite impulses at different "surfaces".
- The points $x_0 = \pm 1$ and $x_0 = 0$ are fixed points.
- If $0 < x_0 < 1$, then there are infinitely many impulses at different "surfaces" at the instants τ_i, $\lim_{i\to\infty}\tau_i = \infty$, and $\lim_{i\to\infty} x(\tau_i \pm 0) = 0$.
- If $-1 < x_0 < 0$, then there are infinitely many impulses at the same "surface" at the instants τ_i, $\lim_{i\to\infty}\tau_i = 2$, and $\lim_{i\to\infty} x(\tau_i \pm 0) = 0$. For $t > 2$, the solution is identically zero.

Some notes

It can happen that `IDERKSolve` does not detect $S_i(t, x(t)) = 0$, if the solution only touches and does not cross this surface.

The program applies the first impulse in the list `Impulse`, for which the inequality $S_i(s_l, x_l)\, S_i(s_{l+1}, x_{l+1}) \leq 0$ holds. Although the surfaces $S_i(t, x(t)) = 0$ are assumed disjoint (i, 1, 2, ...), they can be close to each other. If the step size `dt` is not small enough, the condition $S_i(s_l, x_l)\, S_i(s_{l+1}, x_{l+1}) \leq 0$ can hold for several functions S_i, and hence the solver may not choose the real surface. If it seems to happen, use smaller step size and zooming-in technique for better approximation.

4. Example: swinging

Every parent knows how to give children a swing. The models of swinging are simply variations of the damped nonlinear pendulum with an external force. It can be described by the system

$$x' = y,\ y' = -a\,y - \sin(x)\ (a > 0). \qquad (12)$$

This equation and the following figures are well known, but let us see the vector field and the trajectories as strarting points before the other cases.

```
<< package//odesolve.m
```

```
a = 0.2;
var = {x, y};
swing := {y, -a y - Sin[x]};
x1 = -π; x2 = 3 π; y1 = -3; y2 = 3;
t0 = 0; t1 = 20.;
```

The total energy of this system is

```
V[{x_, y_}] = y^2/2 + ∫_0^x Sin[u] du;
```

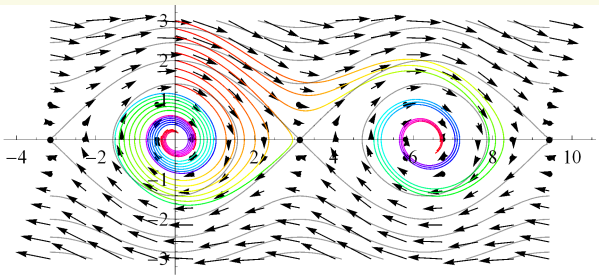
```
ContourSwingV =
  ContourPlot[V[var], {x, x1, x2}, {y, y1, y2}, ContourShading → False];
SwingField = VectorFieldPlot[swing, {x, x1, x2},
   {y, y1, y2}, Axes → True, ScaleFactor → 1];
Show[ContourSwingV, SwingField, AspectRatio → Automatic];
```

Now, ODESolve is a user-friendly form for NDSolve:

```
IC = Table[ N[{0., u}], {u, 1.4, 3, 0.2}];
PendulumTraj[t_] = ODESolve[swing, var, IC, {t, t0, t1}];
```

Consider the trajectories:

```
PendPlot = ParametricPlot[PendulumTraj[t],
  {t, t0, t1}, PlotRange → All, ColorFunction → (Hue[#3] &)];
Show[SwingField, PendPlot, ContourSwingV]
```



What is important in the point of view of giving a swing in the next sections is that the pendulum can turn over the upper equilibrium, if the speed is too high. Hence too good swings with small friction or air-drag are dangerous.

The external force (giving a swing by parents) can depend on the time, the position and the velocity of the pendulum either continuously or using impulses. Note that kids can swing by themselves without an external hand, as they change the length of the pendulum by moving periodically their legs (see [4] and the references therein).

Let us consider different cases of the parents' techniques: different external forces and different strategies to to choose the "best" time for the force. The reader, later, can combine and investigate them to find a really good one.

Continuous, periodic external force

A simple periodic external force can be $\cos(t)$, and the model is

$$x' = y,\ \ y' = -a\,y - \sin(x) + \cos(t)\ \ (a > 0). \qquad (13)$$

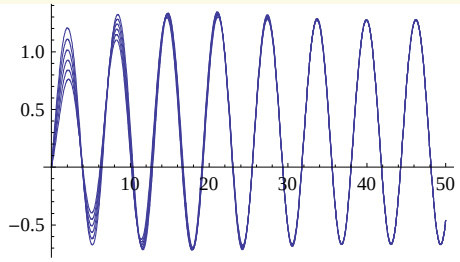
Although this model is very simple, mathematically it is very interesting. It was proved only in 2007 by T. Csendes, B. Bánhelyi, and L. Hatvani [4] that this equation with $a = 0.1$ can produce chaos. We can obtain similar behavior, if the external force is a sign-keeping periodic function with period long enough (longer then the oscillation time of the solutions). We let the reader study the cases with short period. Remember that the oscillation time depends on total energy. Here, let the forced system be

$$x' = y,\ \ y' = -a\,y - \sin(x) + 0.5\left(\cos\tfrac{t}{2}\right)^2\ \ (a > 0). \tag{14}$$

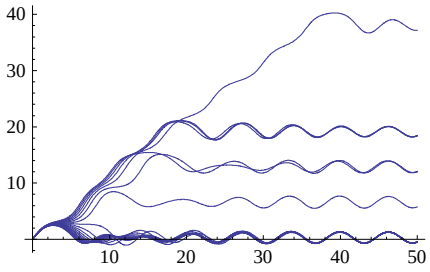
```
force = {0, 0.5 Cos[t/2]^2};
```

For small initial speed, the motion is asymtotically periodic. To see this, let us solve the system, and plot the solutions.

```
t0 = 0; t1 = 50.;
IC = Table[ N[{0., u}], {u, 0.3, 0.8, 0.1}];
PendulumTraj[t_] = ODESolve[swing + force, var, IC, {t, t0, t1}];
Plot[PendulumTraj[t][[All, 1]], {t, t0, t1}, PlotRange → All]
```



If the parent has too much energy and give a large initial speed, the motion can become rather unpredictable:



Periodic impulsive swinging

Now, let the external force be impulsive. Then the pendulum is swinging by equation (12), and at certain instants, an impulsive effect increase the speed to compensate the damping. One possibility is a periodically given constant impulse that turns the swing toward the lower equilibrium:

$$\begin{gathered} x' = y,\ \ y' = -a\,y - \sin(x),\ \ a > 0,\ \text{if}\ \ t \neq k\,T, \\ x(k\,T + 0) = x(k\,T - 0), \\ y(k\,T + 0) = y(k\,T - 0) - A_k\,\operatorname{sign}(x(k\,T - 0)),\ A_k > 0. \end{gathered} \tag{15}$$

A "sophisticated" impulse should also take into account the value of the speed.

Based on the example in section 2, system (15) can be easily investigated in *Mathematica*. For the qualitative properties several questions may arise. What properties on a, T and $\{b_k\}$ can guarantee a stable nontrivial periodic swinging? What impulses can result chaos in this system? We lead to (try to) answer these questions to the reader, and we will do only some basic investigations.

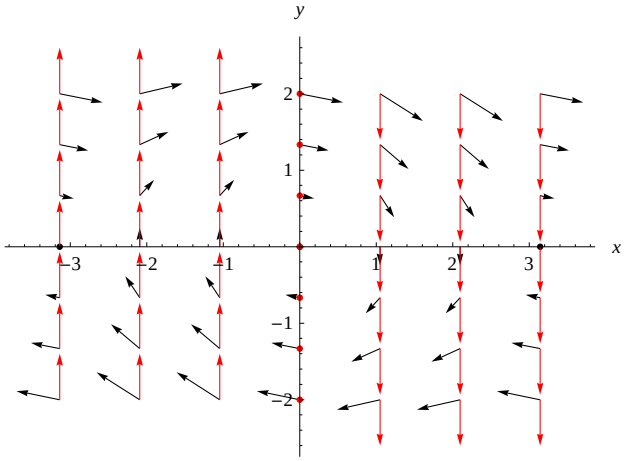
```
T = 1.; A = 0.6;
tn := Table[n T, {n, 1, 100}];
SwingImpulse[n_, tn_List, u_List] := {u[[1]], u[[2]] - A Sign[u[[1]] ]};
```

```
x1 = -π; x2 = π; y1 = -2; y2 = 2;
```

```
t0 = 0; t1 = 40.;
```

The vector fields show the direction of the motion in the phase space:

```
Fields2D = DoubleFieldPlot[{swing, SwingImpulse[1, tn, var] - var},
  {x, x1, x2}, {y, y1, y2}, Axes → True, PlotPoints → 7, AxesLabel → {x, y}]
```

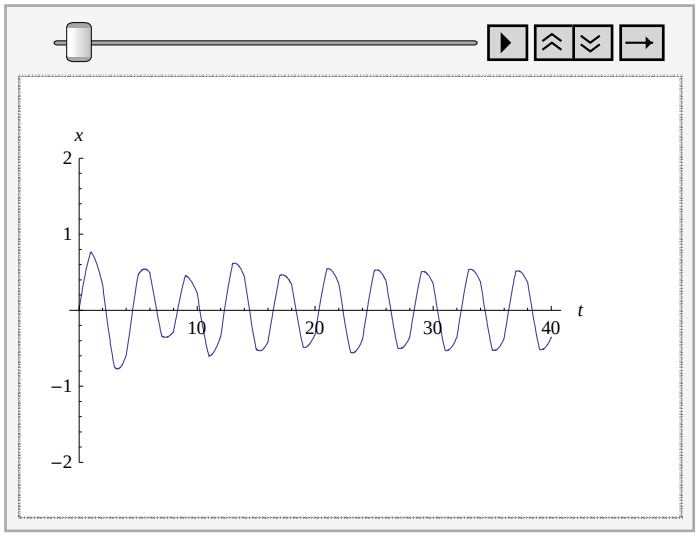


Consider the solutions:

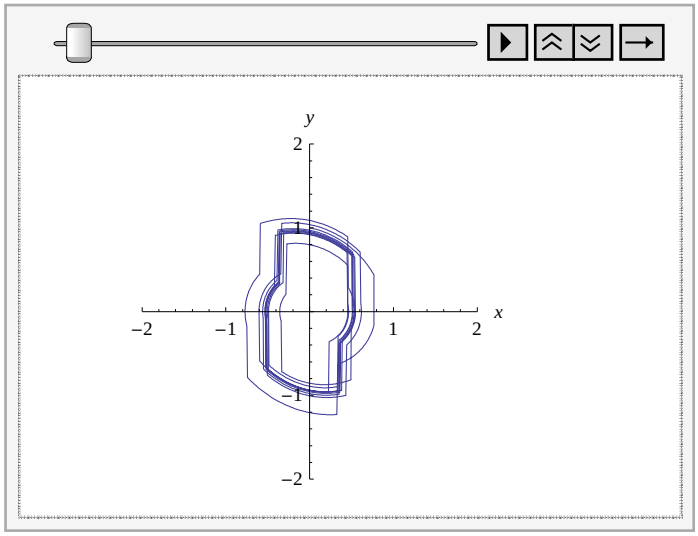
```
IC = Table[ N[{0., u}], {u, 1, 2.4, 0.2}];
SwingTraj[t_] = IDESolve[swing, var, tn, SwingImpulse, IC, {t, t0, t1}];
```

Animate the solutions $x(t)$ and the trajectories $\{x(t), y(t)\}$ with respect to the initial speed:

```
Map[Plot[#, {t, t0, t1}, PlotRange -> {-2, 2}, ImageSize → {250, 180},
    AxesLabel → {t, x}] &, SwingTraj[t][[All, 1]]] // ListAnimate
```



```
Map[ParametricPlot[#, {t, t0, t1}, PlotRange → {{-2, 2}, {-2, 2}},
    ImageSize → {250, 180}, AxesLabel → {x, y}] &, SwingTraj[t]] // ListAnimate
```



We can have the conjecture by our figures that this method looks good enough to provide a stable swinging. Every solution looks like tending asymptotically to a periodic motion. But it depends mainly on the initial conditions. In addition, it can be of interest to investigate the dependence on the period and value of the impulse. It is also worth to investigate some cases of large speed.

Smart state-dependent swinging

Consider now some cases of state-dependent swinging. The parent stands on one side of the swing, hence we can assume that the impulses are given for $x(t) \geq 0$. A rather practised parent would do the following: he or she pushes back the swing if the velocity is zero, and the force (impulse on the speed) not only shows toward the lower equilibrium, but it is reciprocally proportional to $x(t)$, for not too big angle (such as $x(t) < \frac{4\pi}{5}$). For example, let the impulse be $y(t+0) = \alpha\left(\hat{x} - \mathrm{Max}\left(\left|x(t)\right|, \hat{x}\right)\right)\mathrm{sign}(-x(t-0))$. Now, such a system (15) takes the form:

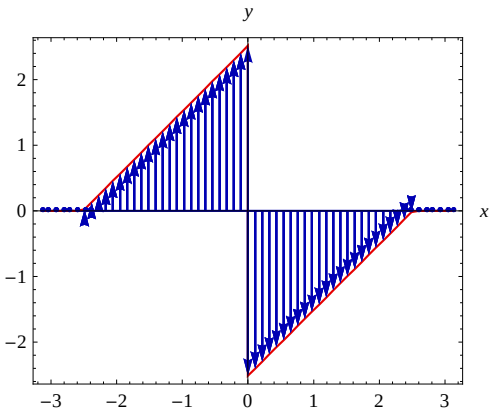
```
x1 = -π; x2 = π; y1 = -2; y2 = 2;
```

```
SmartImp[x_, y_] = {x,
    Piecewise[{{(4 π/5 - Min[Abs[x], 4 π/5.]) Sign[-x], (Abs[x] ≤ 4 π/5)}, {y, True}}]};
SmartImpulse = {{y , SmartImp[x, y], 1}};
```

One can verify that the parameter value 0 in `Impulse` would result in nonrealistic solutions. The system is autonomous, hence we can plot the vector field and the jumps in the 2D phase space. The vector field of the damped pendulum is well known:

Use the statement `AutonomousJumpPlot2D` to visualize the impulse mappings.

```
AutonomousJumpPlot2D[SmartImpulse[[1]], {x, x1, x2},
 {y, -2, 2}, PlotPoints → 30, PlotRange → All, AspectRatio → Automatic]
```

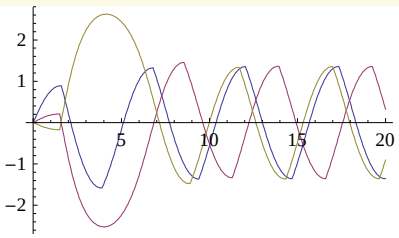


Solve the system and visualize the solutions:

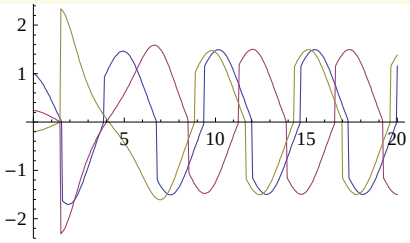
```
t0 = 0; t1 = 20; dt = 0.05; IC = {{0, 1}, {0., 0.24}, {0, -0.2}};
```

```
smartswing = IDERKSolve[swing, SmartImpulse, var, IC, {t, t0, t1, dt}];
```

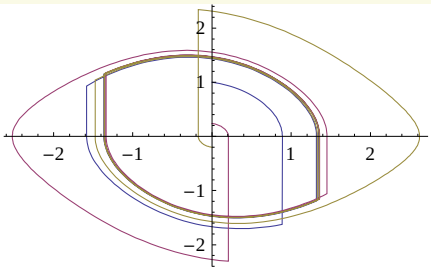
```
ListPlot[smartswing /. {s_?NumericQ, u_?NumericQ, v_?NumericQ} → {s, u},
 PlotRange -> All, Joined → True]
```




```
ListPlot[smartswing /. {s_?NumericQ, u_, v_} → {s, v}, Joined → True]
```




```
ListPlot[smartswing /. {s_?NumericQ, u_, v_} → {u, v}, Joined → True]
```



We can see, that the trajectories approximate a limit cycle in the phase plane.

◆ Exercise

We recommend the reader do more experiments and conclude the precise mathematical proof.

Lazy swinging

A lazy parent does not care about the direction of the motion, he pushes the swing toward the equilibrium ($x(t)\, y(t) \geq 0$) if it reaches a given position $\hat{x}$, but he tries to be smart enough to pay attention to speed, i.e., it cannot be too big (to avoid the swing turning over). The strength of the impulse be reciprocally proportional to the angle. This is a quite, since the damping can be too large and hence the swing cannot reach again the required position. Now, such a system can be:

```
x1 = -π; x2 = π; y1 = -2; y2 = 2;
```

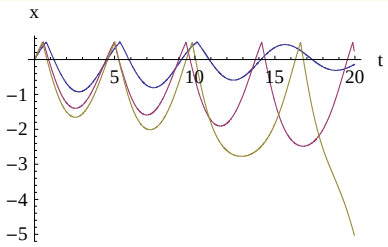
```
LazyImp[x_, y_] = {x, Piecewise[{{- 2 y, 0 ≤ y ≤ 2}, {y, True}}]};
LazyImpulse = {{x - 0.5 , LazyImp[x, y], 0}};
```

Solve the system, and then plot the solutions and the trajectories.

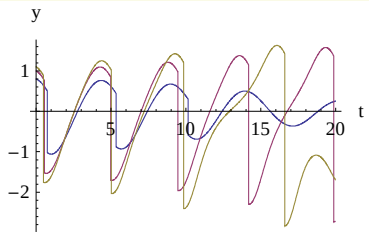
```
t0 = 0; t1 = 20; dt = 0.01; IC = {{0, 0.8}, {0, 1}, {0., 1.1}};
```

```
lazyswing = IDERKSolve[swing, LazyImpulse, var, IC, {t, t0, t1, dt}];
```

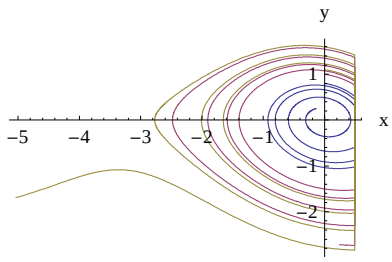
```
ListPlot[lazyswing /. {s_?NumericQ, u_?NumericQ, v_?NumericQ} → {s, u},
 PlotRange -> All, AxesLabel → {"t", "x"}, Joined → True]
```




```
ListPlot[lazyswing /. {s_?NumericQ, u_, v_} → {s, v},
 PlotRange → All, AxesLabel → {"t", "y"}, Joined → True]
```



```
ListPlot[lazyswing /. {s_?NumericQ, u_, v_} → {u, v},
 Joined → True, PlotRange → All, AxesLabel → {"x", "y"}]
```



We can see that there must be a periodic solution, but it is unstable. This "lazy" method cannot lead to stable swinging. As before, we leave this problem to the reader.

◆ Exercise

We recommend the reader to find and investigate other "smart" swinging strategies.

5. The motion of a bouncing ball

Let us consider another well known phenomenon: the motion of a bouncing ball. The question is how to make the motion of the ball periodic, and is the possible periodic motion stable?

The free bouncing

Let the ball fall down from a given height h (some initial speed is possible). Let $x(t)$ denote the distance of the ball from the floor. The motion is described by $x'' = -g$ where g is the gravitational acceleration. When the ball reaches the floor at τ ($x(\tau) = 0$) a collision happens, and

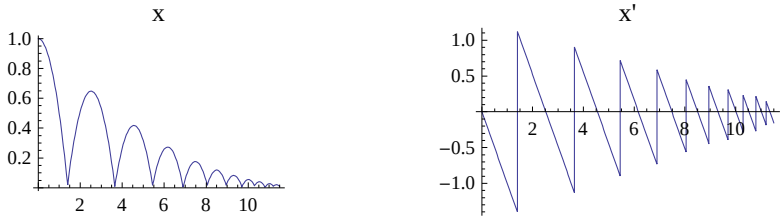
$$x'(\tau + 0) = -\alpha\, x'(\tau - 0)$$

where $0 \le \alpha \le 1$. If $\alpha = 1$, then the collision is perfectly elastic. If $\alpha = 0$, then the collision is perfectly inelastic, thus the ball stays on the floor. Below, $v(t) = x'(t)$ and for the simplicity we take $g = 1$ and $\alpha = 0.8$. The solutions are:

```
Clear[α];
var = {x, v}; eqnparm = {g → 1, α → 0.8}; xvdot = {v, -g} /. eqnparm;
Impulse = {{x, {x, -α v}, 0}} /. eqnparm;
t0 = 0; t1 = 12; dt = 0.05; x0list = {{1., 0}};
```

```
sol=IDERKSolve[xvdot,Impulse,var,x0list,{t,t0,t1,dt}];
```

```
cimke = {"x", "x'"};
ListSolPlot[sol, cimke, ImageSize → {200, 100}][[1]]
```



We can see the well known fact that the ball asymptotically tends to the floor. To try to keep the ball bouncing, we consider two special cases: bouncing at fixed height and fixed instanst.

Giving a beat at a given height

Let the ball move upwards and let $x(t) < h$. As it reaches $x(\tau) = h$ at some τ, let us beat it back, i.e.,

$$x'(\tau + 0) = -\beta\, x'(\tau - 0), \tag{16}$$

where $\beta > 1$, as the energy must be increased. Let us do some experiments.

Changing the initial speed

Fix the constants α and β, and let the ball start from the position $0 < x = h_0$, at different values of initial speed. Comparing to the previous example, we need only to modify the variable `Impulse`. Here $h = 1$, $\beta = 1.3$, $h_0 = 1.5$.

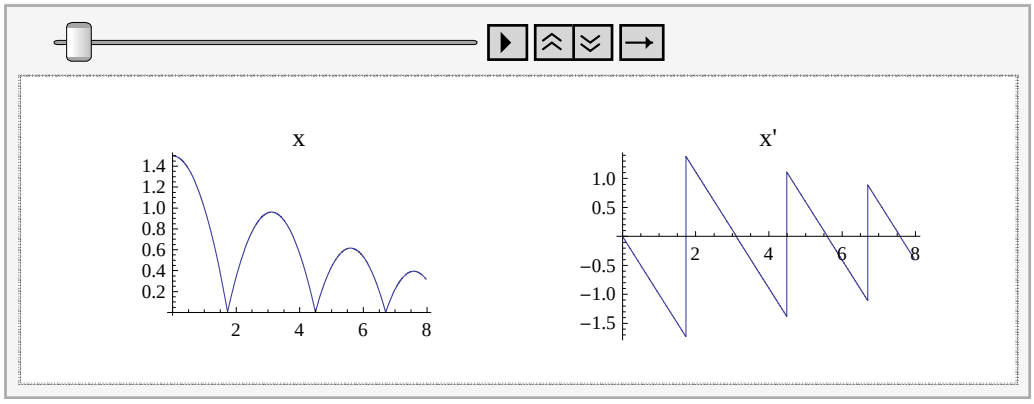
```
var = {x, v};
eqnparm = {g → 1., α → 0.8, h0 → 1.5, h -> 1., β → 1.3};
xvdot = {v, -g} /. eqnparm;
Impulse = {{x, {x, -α v}, 0},
    {Piecewise[{{x - h, v ≥ 0}, {1, True}}], {x, -β v}, 0}} /. eqnparm;
t0 = 0; t1 = 8; dt = 0.005;
```

The ball is beaten back only when it is moving upwards ($v \geq 0$), hence the `Piecewice` function is used in the definition of `Impulse`.

```
x0list = Table[{h0, -v0}, {v0, 0, 3, 0.2}] /. eqnparm;
sol = IDERKSolve[xvdot, Impulse, var, x0list, {t, t0, t1, dt}];
```

Plot the solutions and their derivatives:

```
label = {"x", "x'"};
ListSolPlot[sol, label, ImageSize → {200, 100}] // ListAnimate
```



We can observe that for any (either small or large) initial speed the ball loses energy at each collision with the floor, hence it can reach the needed lattitude at most finite times. We encourage the reader to prove it. Hence, to keep the ball bouncing, the lost energy must be compesated. Consider such a case in the next point.

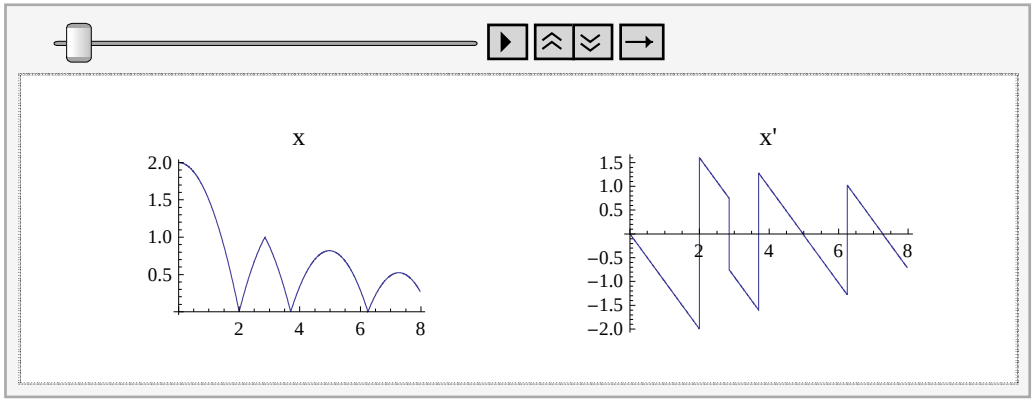
Changing the beating force (by constant β)

Let the ball just fall down ($x(0) = h_0,\ x'(0) = 0$), and then change the constant β in formula (16) interactively. Fix the constant α (depending on the properties of the floor). We need only to modify the variable `Impulse` in the previous example. Here $h = 1,\ \ h_0 = 1.5$, $\alpha = 0.8$, and $\beta \in [1, 2]$. Note that h_0 must be big enough to give enough energy to the ball to reach h!

```
var = {x, v};
eqnparm = {g → 1., α → 0.8, h0 → 1.5, h → 1};
xvdot = {v, -g} /. eqnparm;
Impulse = {{x, {x, -α v}, 0},
    {Piecewise[{{x - h, v ≥ 0}, {1, True}}], {x, -β v}, 0}} /. eqnparm;
t0 = 0; t1 = 8; dt = 0.005;
x0list = {{2, 0}};
sol = Table[
   IDERKSolve[xvdot, Impulse, var, x0list, {t, t0, t1, dt}][[1]], {β, 1, 2, 0.1}];
```

Plot the solutions and their derivatives. Move the slider to see the different cases:

```
label = {"x", "x'"};
ListSolPlot[sol, label, ImageSize → {200, 100}] // ListAnimate
```



We can observe at small β, that the ball loses energy and can reach the needed height at most finite times. On the other hand, for big β the ball bounces infinite times, the oscillation time (time elapsing between reaching the extremal positions) is smaller and smaller. So this bouncing strategy is not smart enough. A good strategy should takte into account both the current height and velocity. Here we recommend the reader to do some exercises and experiments.

Beating at fixed instants

We can also try to bounce the ball by beating at fixed instants using the following rule:

$$x'(i\,T + 0) = -\beta\,|\,x'(i\,T - 0)\,|,\ \ T > 0,\ \ \beta \geq 1.$$

Let us fix now α and β, and change the time T. The current parameters are $\alpha = 0.8,\ \ \beta = 1.5,\ \ h = 1,\ \ h_0 = 1.5$.

Animation: changing *T*

```
Clear[x, v, T, g, α, β];
var = {x, v}; eqnparm = {g → 1., α → 0.8, h0 → 1.5, v0 → 0., β → 1.5};
```

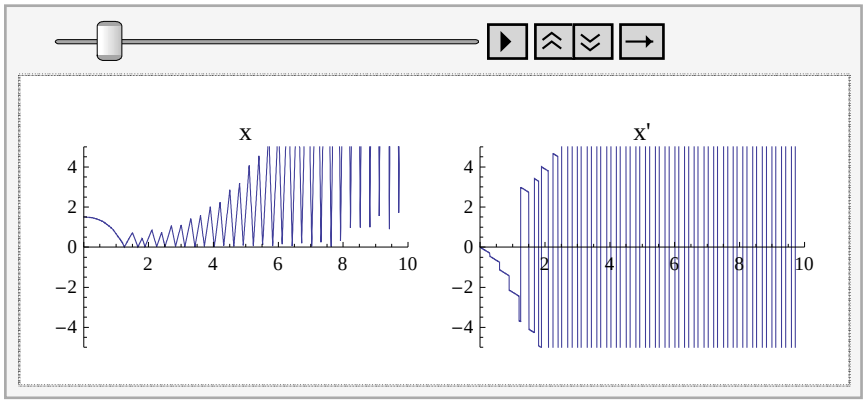
```
xdot = {v, -g} /. eqnparm;
Impulse = {{x, {x, -α v}, 0}, {Sin[π t / T], {x, -β Abs[v]}, 1}} /. eqnparm;
```

```
t0 = 0; t1 = 10; dt = 0.005; x0list = {{h0, v0}} /. eqnparm;
```

Solve the system and plot the solutions:

```
sol = Table[
   IDERKSolve[xdot, Impulse, var, x0list, {t, t0, t1, dt}][[1]], {T, 0.2, 1.5, 0.1}];
```

```
ListSolPlot[sol, {"x", "x'"}, PlotRange → {{t0, t1}, {-5, 5}}] // ListAnimate
```



We can see that this strategy looks worse than beating at fixed height. Even more, rare (increase *T*) big beating can result in strange behavior (try it!).

6. Conclusions

In this paper, we gave an introduction by examples to the computer-aided study, the theory, special properties, as well as some qualitative methods of impulsive systems. Our examples illustrated that the formal description, theoretical research on the qualitative properties is much more complicated than for the ordinary differential equations. Hence, the computer experiments are of great importance. We presented some new and/or specialized form of built-in tools in *Mathematica* for such systems. The complete package contains many more functions to help qualitative methods such as Poincaré maps, phase maps, but they will be subjects of other papers.

The packages used in the paper are summarized in the Appendix and are available on the web-site http://www.-model.u-szeged.hu.

7. Acknowledgement

Research supported by the Hungarian National Foundation for Scientific Research Grant No. T 049516, and co-financed by the European Union through the Hungary-Serbia Cross-border Cooperation programme in the frame of the project IPA HU-SRB/0901/221/088.

References

[1] Z. Agur et al., Pulse mass measless vaccination across age cohorts, *Proc. Natl. Acad. Sci. USA*, **Vol**(90), 1993 pp. 11698-11702.

[2] D. D. Bainov, P. S. Simeonov, *Systems with Impulse Effect, Stability, theory and Applications*, Wiley, 1989.

[3] D. D. Bainov, P. S. Simeonov, *Impulsive Differential Equations, Asymptotic Properties of the Solutions*, World Scientific, 1995.
[4] T. Csendes, B. Bánhelyi, and L. Hatvani, Towards a computer - assisted proof for chaos in a forced damped pendulum equation, *Journal of Computational and Applied Mathematics,* **Vol**(199), 2007 pp. 378 - 383.
[5] G. R. Graef, J. Karsai, On the Asymptotic Behavior of Solutions of Impulsively Damped Nonlinear Oscillator Equations, *J. Computational and Appied. Mathematics*, **Vol**(71), 1996 pp. 147-162.
[6] G. R. Graef, J. Karsai, On Irregular Growth and Impulse Effects in Oscillator Equations, *Proceedings of the 2nd International Conference on Difference Equations and Applications, Veszprém, 1995*, Gordon and Breach, 1997 pp. 253-262.
[7] G. R. Graef, J. Karsai, On the Asymptotic Properties of Nonoscillatory Solutions of Impulsively Damped Nonlinear Oscillator Equations, *Dynamics of Continuous, Discrete and Impulsive Systems*, **Vol**(3), 1997 pp. 151-166.
[8] J. R. Graef, J. Karsai J., On the oscillation of impulsively damped halflinear oscillators, *Electronic J. on Qualitative Theory of Differential Equations*, **Vol**(11), 2000.
[9] J. R. Graef, J. Karsai J., Oscillation and nonoscillation in nonlinear impulsive systems with increasing energy, *Discrete and Continuous Dynamical Systems,* **Spec. Vol.**, 2001 pp. 161-173.
[10] P. Hartman, On a theorem of Milloux, *Amer. J. Math.*, **Vol**(70), 1948 pp. 395-399.
[11] J. Karsai J., J. R. Graef, Behavior of Solutions of Impulsively Perturbed Non-Halflinear Oscillator Equations, *J. Math. Anal. Appl.*, **Vol**(244), 2000 pp. 77-96.
[12] Karsai J., Graef J. R., The Milluox-Hartman theorem for impulsive systems, *Dynamics of Continuous, Discrete and Impulsive Systems*, **Vol**(6), 1999 pp. 155-168.
[13] J. Karsai, *Models of impulsive phenomena : Mathematica experiments, in Hungarian*, Typotex, Budapest, 2002.
[14] J. Karsai, Mathematica-aid to study impulsive system, *Web-Proceedings of the International Mathematica Symposium 2008, Maastricht, The Netherlands*, http://bmiaserver.bmt.tue.nl/eProceedings/WWW/IMS 2008 e-Proceedings.html
[15] Liu X., *Stability problems of impulsive systems, Differential equations and control theory*, Dekker, 1996, 187-201.
[16] R. E. Maeder, *Programming in Mathematica*, Second Edition, Addison - Wesley, 1991.
[17] R. E. Maeder, *The Mathematica Programmer I-II*, Academic Press, 1996.
[18] N. Rouche, P. Habets, M. Laloy, *Stability theory: The direct method of Liapunov* (in Hungarian), Műszaki Könyvkiadó, Budapest, 1984.
[19] A. M. Samoilenko, N. A. Perestyuk, *Impulsive Differential Equations,* World Scientific, 1994.
[20] Shulgin B. et al, Pulse vaccination strategy in the SIR Epidemic model, *Bull. Mathematical Biology*, **Vol**(60), 1998 pp. 1123-1148.

Appendix. Summary of the packages

The following packages have to be loaded:

- VectorFieldPlots` *Mathematica* standard package for vector fields
- Impulseplot.m Visualization of impulses, jumps
- Idesolve.m Solve systems with fixed impulse instants
- IdeRKsolve.m Solve general systems with a Runge - Kutta method

IDESolve package for systems with fixed impulse instants

```
IDESolve[ODErhs,var,tn,Imp,IClist,{t,t0,t1},opt]
```

It solves a system for several initial conditions based on `NDSolve`.

`ODESolve[ODErhs,var,IClist,{t,t0,t1},opt]`

It solves an ODE for several initial conditions with `NDSolve`.

IDERKSolve package for general impulses

`IDERKSolve[ODErhs,Impulse,var,IClist,{t,t0,t1,dt}]`

It solves general and autonomous systems based on a Runge-Kutta method implemented in [16].

ImpulsePlot package to visualize impulses

`FixedImpulseFieldPlot[Imp,tn,{t,t1,t2},{x,x0,x1,dx},opt]`

Plots scalar $I(t_i, x)$ impulse field in the system $\{t, x\}$ for $t_i \in$ tn, t1 $\le t_i \le$ t2.

`AnimateImpulse[Imp,tn,{t,t1,t2},{x,x0,x1},opt]`

A 2D animation (table) of the impulse mappings $I(t_i, x)$ by $t_i \in$ tn, t1 $\le t_i \le$ t2.

`StackImpulse[Imp,tn,NN,{x,x0,x1},opt]`

Stack in 3D ($\{t, x, y\}$) the impulse mappings $I(t_i, x)$ for $t_i \in$ tn, t1 $\le t_i \le$ t2.

`DoubleFieldPlot[{fld1_,fld2_},{x_,x0_,x1_},{y_,y0_,y1_}, opt___]`

Plots two vector fields, the first one is with option `ScaleFactor->None`.

```
ContourFieldPlot2D[surf,fld,{x,x0,x1},{y,y0,y1},
    {ContourOpt,FieldOpt,GraphicsOpt}]
```

Plots vectors of the field `fld` starting out of the contour line `surf == 0`.

```
FixedImpulseFieldPlot3D[Imp,tn,{t,t1,t2},{x,x0,x1,dx},
    {y,y0,y1,dy},opt]
```

Plots a planar $I(t_i, x, y)$ impulse field in the system $\{t, x, y\}$ for $t_i \in$ tn, t1 $\le t_i \le$ t2.

`JumpPlot[SS,II,{t, t0, t1},{x, x0, x1},opt]`

Visualizes the general scalar impulse mapping. The curve `SS == 0` and its image transformed by the impulse `II` are plotted in the system $\{t, x\}$.

`JumpPlot3D[SS,II,{t,t0,t1},{x,x0,x1},{y,y0,y1},opt]`

Az `SS`=0 and its image transformed by the impulse `II` are plotted in the system $\{t, x, y\}$.

Visualizes the general 2D impulse mapping. The surface `SS = 0` and its image transformed by the impulse `II` are plotted in the system $\{t, x, y\}$.

`AutonomousJumpPlot2D[SS,II,{x,x0,x1},{y,y0,y1},opt]`

The mapping of the impulses `II` on the curve `SS = 0` are plotted in the system $\{x, y\}$.

`AutonomousJumpPlot3D[SS,II,{x,x0,x1},{y,y0,y1},opt]`

The mapping of the impulses `II` on the surface `SS = 0` are plotted in the system $\{x, y, z\}$.

`ContourLinePlot3D[f,{t,t0,t1}{x,x0,x1},{y,y0,y1},opt]`

Special version of ParametricPlot3D to plots the $f(t, x, y)$ contour lines in $\mathbb{R}^3$ for $t =$ const. (use Mesh option to set them) as well as the contour surfaces with opacity (use option Contours).