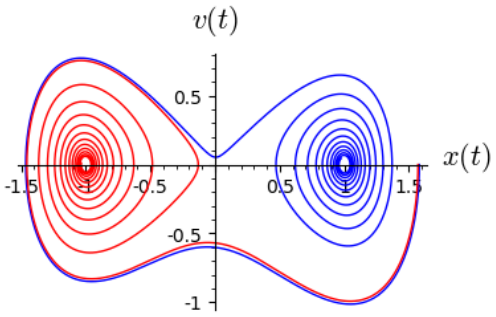
Problems in Physics

(with SageMath)



Marcin Kostur, Jerzy Łuczka, Łukasz Machura

Institute of Physics
University of Silesia
Poland

August 21, 2019

Download Juptyer Notebook files, pdf and html files of this book from
https://github.com/marcinofulus/Dynamical_Systems

Contents

1 Deterministic Dynamics

1.1 Introduction: modeling of processes using differential equations

One of the basic laws of physics is Newton's second law. It describes classic mechanical systems. These systems are the idealization of real systems occurring in the world around us. In the simplest version, Newton's dynamics principle for one particle moving only along one coordinate axis, e.g. along the OX axis, can be formulated as follows:

"Particle motion is determined by the forces that act on the particle."

From a mathematical point of view, motion of the particle is described by the Newton equation:

$$ma = F$$

There are three quantities in this equation:

- m is the mass of the particle - a is particle acceleration - F is the force acting on the particle.

Since the motion occurs along the OX axis (so we assume for simplicity), the force F works only in the OX direction and the acceleration a is along the OX axis. We know from the course of physics that particle acceleration is a derivative (with respect to time) of the particle velocity v. In turn, the particle velocity v is a derivative of the particle position x:

$$a = \frac{d}{dt}v = \frac{d}{dt}\frac{d}{dt}x = \frac{d^2x}{dt^2}$$

In a general form, the force

$$F = F(x, v, t) = F(x, dx/dt, t)$$

may depend on the particle position x, its velocity $v = dx/dt$ and time t.

Using the definition of acceleration, the Newton equation can be rewritten as follows:

$$m\frac{d^2x}{dt^2} = F\left(x, \frac{dx}{dt}, t\right) \qquad (1)$$

In this way we obtain a differential equation that describes one-dimensional motion of the particle along the OX axis. What can we say about this equation:

- This is a second order differential equation because a second order derivative d^2x/dt^2 appears. - This is an ordinary differential equation, because there are no partial derivatives but only derivatives with respect to one variable - in this case, time derivatives t. - Newton's equation alone is not enough to describe the motion of a particle. We must set initial conditions for this equation. Because this is a second-order differential equation, we must set two initial conditions: the initial coordinate $x(t_0) = x_0$ and the initial velocity $v(t_0) = v_0$ of the particle. The initial conditions can be set at any time t_0, but usually the initial time is $t_0 = 0$.

Equation (1) can be presented in an equivalent form:

$$\frac{dx}{dt} = v \qquad (2a)$$

$$\frac{dv}{dt} = \frac{1}{m} F\left(x, v, t\right) \qquad (2b)$$

where we introduced a new variable v which is the particle velocity. In this way we obtain a set of 2 first order differential equations. As we will see later, such manipulation is useful when introducing the concept of phase space for differential equations. If the force F does not explicitly depend on time, then the system of equations

$$\frac{dx}{dt} = v \qquad (3a)$$

$$m\frac{dv}{dt} = F(x, v) \qquad (3b)$$

is called autonomous. In other words, it is a set of 2 autonomous ordinary differential equations. We say then that its phase space is 2-dimensional.

If the particle moves on the 2-dimensional (X, Y) plane, then Newton's equations have the form:

$$m\frac{d^2x}{dt^2} = F\left(x, y, \frac{dx}{dt}, \frac{dy}{dt}, t\right) \qquad (4a)$$

$$m\frac{d^2y}{dt^2} = G\left(x, y, \frac{dx}{dt}, \frac{dy}{dt}, t\right) \qquad (4b)$$

where F and G are force components acting on the particle towards x and y. In general, these forces depend on both components (x, y) of the particle position, its velocity components $(dx/dt, dy/dt)$ and time t. If the force components F and G do not depend explicitly on time, then proceeding as before we get the set:

$$\frac{dx}{dt} = v \qquad (5a)$$

$$\frac{dy}{dt} = u \qquad (5b)$$

$$m\frac{dv}{dt} = F(x, y, v, u) \qquad (5c)$$

$$m\frac{du}{dt} = G(x, y, v, u) \qquad (5d)$$

It is a set of 4 autonomous ordinary differential equations. We say then that its phase space is 4-dimensional.

For a particle moving in space (X, Y, Z), we have 3 second-order differential equations. If the three force components do not explicitly depend on time, then proceeding as before we get a system of 6 first-order differential equations and the phase space is 6-dimensional. In general, for N particles moving in space, the phase space has the dimension $6N$. The analysis of such equations exceeds the capabilities of modern mathematics in the sense that we know little about the general properties of specific systems that we model. This means that we have to use numerical methods and the computer is an indispensable tool for analysis. Above we gave one example of modeling. It is based on Newton's formalism and Newton's equations of motion which can be used to describe the dynamics of classical particles. Sometimes it is convenient to use other formalism such as Lagrange's formalism or Hamilton's formalism. In many cases, all three formalisms are equivalent. For so-called systems with constraints, it is convenient to use Lagrange formalism or Hamilton's formalism. Defining the system of differential equations as autonomous, we assumed that force does not depend explicitly on time. This may seem like a restriction. This is not true. Non-autonomous systems can be reduced to autonomous systems by introducing an additional independent variable, an additional "position". We will show it on a simple example. Let's consider a particle moving along the X-axis. The particle is affected by a friction force proportional to velocity of the particle, $F = -\gamma v$, the potential force $F(x) = -V'(x)$ comes from the potential energy $V(x)$ (abbreviated as potential). This force is a negative potential gradient (i.e. derivative of $V'(x)$). In addition, $F(t) = A\cos(\omega t)$ periodically acts on the particle. Newton's equation has the form

$$m\ddot{x} = -\gamma\dot{x} - V'(x) + A\cos(\omega t) \qquad (6)$$

where dots denote derivatives with respect to time and apostrophe denotes derivative with respect to the position x:

$$\dot{x} = \frac{dx}{dt}, \qquad \ddot{x} = \frac{d^2x}{dt^2}, \qquad V'(x) = \frac{dV(x)}{dx}$$

The equation (6) can be rewritten in the form of a set of 3 autonomus differential equations:

$$\dot{x} = v \qquad (7a)$$

$$m\dot{v} = -\gamma v - V'(x) + A\cos(z) \qquad (7b)$$

$$\dot{z} = \omega \qquad (7c)$$

We show equivalence as follows:

- in the equation $(7b)$, replace v from the equation $(7a)$ by $v = \dot{x}$, remembering that $\dot{v} = \ddot{x}$ - the equation $(7c)$ can be integrated and we get $z = \omega t$; we insert this expression into the equation $(7b)$.

In this way we obtain the equation (6). We see that the equation (6) is equivalent to a set of 3 first-order autonomous differential equations. Therefore the phase space corresponding to the system (6) is 3-dimensional. An important hint from this example is how to obtain an autonomous system of first-order differential equations. The number of these equations defines the phase space of the system. The dimension of this space is one of the most important characteristics. Please remember that!

Physics also uses the partial differential equation apparatus: the Schrodinger equation, the wave equation, the diffusion equation, Maxwell equations. Their analysis is much more difficult. There are special and specific mathematical methods to obtain information about the properties of systems described by such equations. The phenomenological modeling method is used in many fields of science (chemistry, biology, sociology, economic sciences). To illustrate how to use it we will give one more example.

1.2 Modeling of the growth process

Growth processes occur in many areas of Nature. You don't have to be a keen observer to see what can grow around us. We are considering one of the possible classes of growth processes: change of population of hares or bacteria, change of cash deposits in bank deposits, an increase in the concentration of substances in chemical reactions or an increase in the number of cancer cells. Often, growth processes are accompanied by processes of decline (disappearance, death, ...). Here, we neglect them.

Let's consider a specific example: an increase in money of a bank deposit after some period of time. Let's assume that at the moment t we have in a bank $x(t)$ (e.g. dollars). We ask how much money will increase after some time h, or how much money will be at the time of $t+h$. We begin to model this process. Let's denote $x(t+h)$ a bank deposit at time $t+h$. This amount consists of staring deposit $x(t)$ at the initial time t and δ - interest increase, i.e.,

$$x(t+h) = x(t) + \delta$$

The growth of δ is proportional to the initial amount of money $x(t)$, the amount of interest k and how long (h) we keep money on deposit, i.e.,

$$\delta \propto x(t), \qquad \delta \propto k, \qquad \delta \propto h$$

It means that in a simple modeling:

$$\delta = kx(t)h$$

That's why

$$x(t+h) = x(t) + kx(t)h$$

We can rewrite this relationship in the form

$$\frac{x(t+h) - x(t)}{h} = kx(t)$$

In the limit of small time increments $h \to 0$, the left side is a derivative leading to the relation

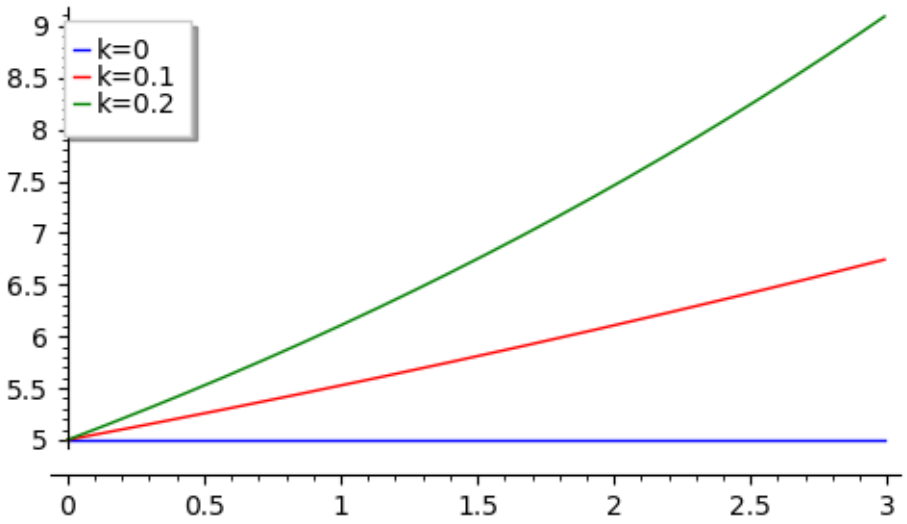
$$\frac{dx(t)}{dt} = kx(t), \qquad x(0) = x_0$$

where x_0 is the initial value of our deposit. In this way we derived an equation describing the dynamics of money growth of our bank deposit. This is an ordinary 1st order, autonomous differential equation. Its phase space is 1-dimensional.

Below we show solutions to this equation for 3 different k values.

[1]:
```
var('N1,N2,N3')
T = srange(0,3,0.01)
sol=desolve_odeint( vector([0, 0.1*N2,  0.2*N3]), [5,5,5],T,[N1,N2,N3])##␣
 ↪rozwizania dla rónych wartoci k=0,0.1, 0.2

line( zip ( T,sol[:,0]) ,figsize=(5, 3),legend_label="k=0") +\
 line( zip ( T,sol[:,1]) ,color='red',legend_label="k=0.1")+\
 line( zip ( T,sol[:,2]) ,color='green',legend_label="k=0.2") ## pokazujemy␣
 ↪rozwizania dla rónych wartoci k=-1, 0, 0.5
```

[1]:



Other growth processes can also be modeled by this equation. It can be the starting point for modifications, generalizations, extensions, etc. A simple extension consists in making the growth rate factor k dependent on additional factors. For example, for modeling of the population growth of hares, we can assume that the growth rate k depend on the number of hares in the population: a large amount of hares causes a large consumption of food, which in turn results in a decrease in the amount of food and difficulties in capture of food. As a result, k's growth rate decreases. In other words, k should be a decreasing function of $x(t)$. There are infinitely many such functions. E.g

$$k \to k(x) = exp(-bx), \qquad b > 0$$

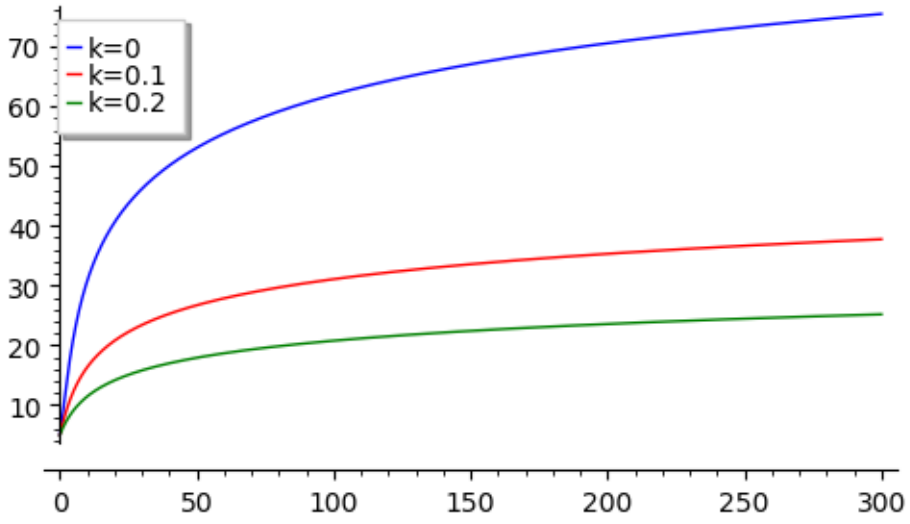
is the descending function of x. Now the differential equation has the form

$$\frac{dx}{dt} = xe^{-bx}, \quad x = x(t), \quad x(0) = x_0$$

What are the effects of such a change? We show this in the figure below. We note that the population growth rate is decreasing compared to the previous case. The model can be extended to include death processes: those natural and those due to the existence of predators that eat individuals of the population. The simple prey-predator model is 2-dimensional: it describes changes in the prey population and changes in the predator population. It is an autonomous system of 2 ordinary differential equations.

[2]:
```
var('x,y,z')
U = srange(0,300,0.01)
sol=desolve_odeint( vector([x*exp(-0.1*x),  y*exp(-0.2*y),  z*exp(-0.3*z)]),␣
 ↪[5,5,5],U,[x,y,z])## rozwizania dla rónych wartoci k=0,0.1, 0.2

line( zip ( U,sol[:,0]) ,figsize=(5, 3),legend_label="k=0") +\
 line( zip ( U,sol[:,1]) ,color='red',legend_label="k=0.1")+\
 line( zip ( U,sol[:,2]) ,color='green',legend_label="k=0.2") ## pokazujemy␣
 ↪rozwizania dla rónych wartoci k=-1, 0, 0.5
```

[2]:



1.3 Discrete time modeling

For the previous example we get the relation for the increment of the bank deposit:

$$x(t+h) = x(t) + khx(t)$$

If the changes occur not continuously but discreetly (e.g. every 1 day, every hour) then the time step h is discrete. We denote

$$x_n = x(t), \qquad x_{n+1} = x(t+h)$$

and then the equation for increment has the form

$$x_{n+1} = x_n + \alpha x_n, \qquad \alpha = kh$$

In this way we obtain the equation with a discrete time. The general form of this type of equation is

$$x_{n+1} = f(x_n)$$

which tells us what the value in the next $n+1$ depends on the value in the previous step n. This equation is also called a recurrence equation. Depending on the form of the $f(x)$ function, we get different models of system dynamics.

The set of 2 equations with discrete time has the form

$$x_{n+1} = f(x_n, y_n)$$

$$y_{n+1} = g(x_n, y_n)$$

Qualitative analysis of such a system is very difficult.

2 System of autonomous differential equations

In the previous part we presented two examples of systems described by ordinary differential equations. The first class of systems is described by classical mechanics and its Newton equations. Another class of systems is the phenomenological equation describing growth processes, chemical kinetics processes, population dynamics in biological systems, etc. These two classes of systems are particular cases of a set of autonomous ordinary differential equations in the form

$$\frac{dx_1}{dt} = F_1(x_1, x_2, x_3, ..., x_n)$$

$$\frac{dx_2}{dt} = F_2(x_1, x_2, x_3, ..., x_n)$$

$$\vdots$$

$$\frac{dx_n}{dt} = F_n(x_1, x_2, x_3, ..., x_n)$$

This set of equations can be written in shorter vector notation as

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x}), \qquad \vec{x}(0) = \vec{x}_0 \qquad (1)$$

where vectors $\vec{x}$ and $\vec{F}$ read

$$\vec{x} = [x_1, x_2, x_3, ..., x_n], \qquad \vec{F} = [F_1, F_2, F_3, ..., F_n]$$

and a set of initial conditions is:

$$\vec{x}(0) = [x_1(0), x_2(0), x_3(0), ..., x_n(0)] = \vec{x}_0 = [x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, ..., x_n^{(0)}]$$

The subscript n tell us how many differential equations are "hidden" in the above vector notation. In other words, we consider the set of n differential equations characterized by n scalar functions $F_i(x_1, x_2, x_3, ..., x_n), (i = 1, 2, 3, ..., n)$. Note that we are considering F_i functions that do not explicitly depend on time. In this case, we say that it is the autonomous system of n ordinary differential equations. In addition, the vector $\vec{F}$ can be regarded as a vector field associated with the system of differential equations or a vector field generated by the system of differential equations.

2.1 Existence and uniqueness of solutions

In order to describe phenomena using ordinary differential equations with corresponding initial conditions at e.g. $t = 0$, we need to know solutions for times $t > 0$ (time evolutions). **EXAMPLE 1:**

The differential equation

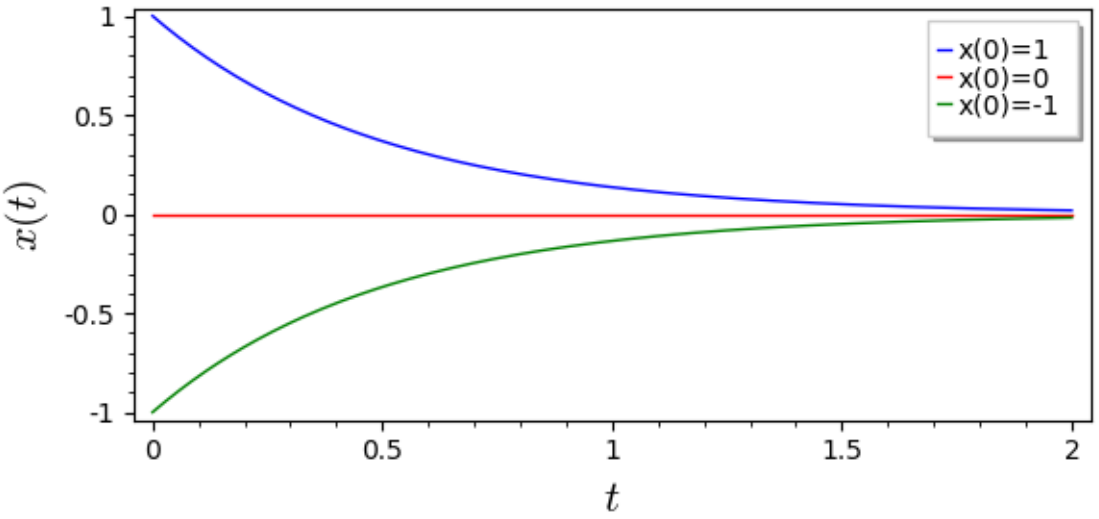
$$\frac{dx}{dt} = -2x, \qquad x(0) = x_0$$

is a linear differential equation with all possible values $x \in (-\infty, \infty)$. It is one of the simplest and frequently exploited equation. One can check that its solution is given by the function

$$x(t) = x_0 e^{-2t}$$

and fulfills the initial condition $x(0) = x_0$. This exponential function is defined for all times $t \in (-\infty, \infty)$. It is the only solution and unique solution. Below, we depict solutions for selected initial conditions. Let us note that all solutions tend to the same final value $x = 0$ when $t \to \infty$. It is called a stationary state (stationary solution, steady-state, ...).

[1]:
```
var('t')
g(t,a) = a*exp(-2*t)
p1=plot(g(t,1),(t,0,2),figsize=(6, 3), legend_label="x(0)=1", color='blue' )
p2=plot(g(t,0),(t,0,2),figsize=(6, 3), legend_label="x(0)=0", color='red' )
p3=plot(g(t,-1),(t,0,2),figsize=(6, 3), legend_label="x(0)=-1", color='green' )
show(p1+p2+p3, axes_labels=[r'$t$',r'$x(t)$'], frame=True, axes=False)
```



EXAMPLE 2:

Equation

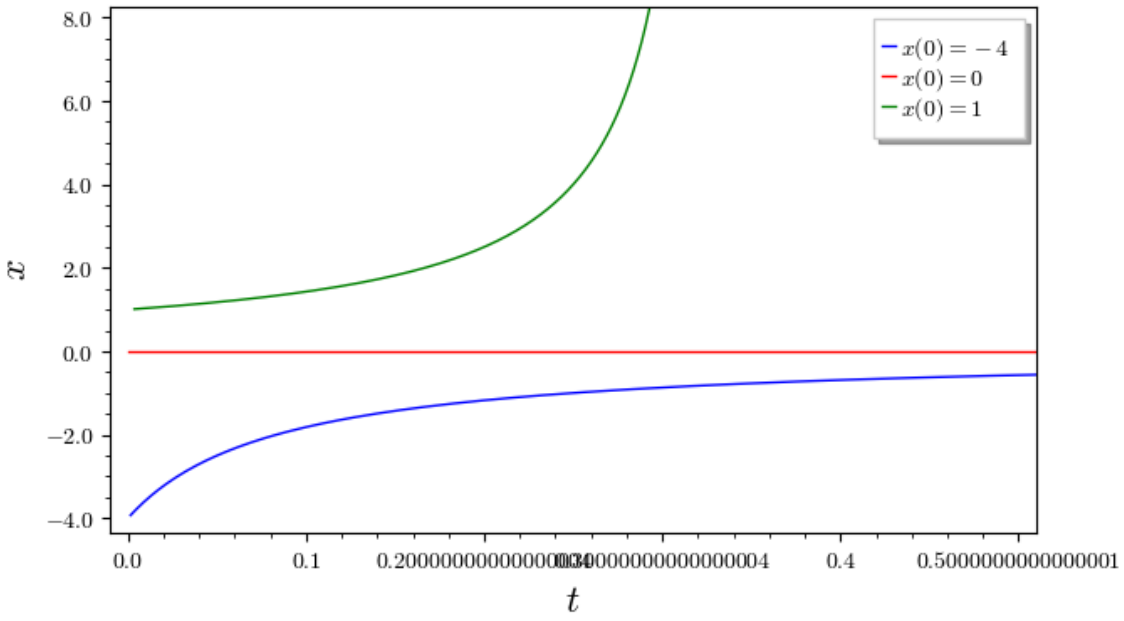
$$\frac{dx}{dt} = 3x^2, \qquad x(0) = x_0$$

is a nonlinear differential equation. The right side of this equation is determined for all values $x \in (-\infty, \infty)$. It can be solved by the variable separation method. We obtain the function

$$x(t) = \frac{x_0}{1 - 3x_0 t}$$

which is the solution and obeys the initial condition $x(0) = x_0$. This function is not defined for all finite values of time $t \in (-\infty, \infty)$. There are restrictions on its value. But this is the only solution and unique solution.

[2]:
```
var('t')
#detect_poles - wykrywanie i rysowanie biegunów
g=plot(-4.0/(1 +12*t),t,0,5,detect_poles='show',legend_label=r'$x(0)= -4$',␣
 ↪color='blue')
g+=plot(lambda t: 0.0,t,0,5,legend_label=r'$x(0)=0$',color='red')
g+=plot(1.0/(1-3*t),t,0,0.33,detect_poles='show',␣
 ↪legend_label=r'$x(0)=1$',color='green')
g.show(axes_labels=[r'$t$',r'$x$'],tick_formatter='latex',xmin=0,xmax=0.5,ymin=-4.
 ↪1,ymax=8, figsize=(7,4), frame=True, axes=False)
```



The solutions with the negative initial condition $x(0) < 0$ (blue curve) are determined for all $t > 0$. The same applies to the solution for the initial condition $x(0) = 0$ (red curve). However, the solution with a positive initial condition $x(0) > 0$ diverges at finite time t (green curve). **EXAMPLE 3:**

The differential equation

$$\frac{dx}{dt} = 2\sqrt{x}, \qquad x(0) = x_0 \geq 0$$

is a nonlinear differential equation. The right hand side of this equations is determined for the non-negative values of x. One can solve this equation by the method of variable separation. The result

is:

$$x(t) = (t + \sqrt{x_0})^2$$

This function is defined for al $t > 0$. If the initial condition $x(0) > 0$ then it is the only solution. If, however, the initial condition is $x(0) = 0$ then there are 2 solutions! One of the possible solution is the function

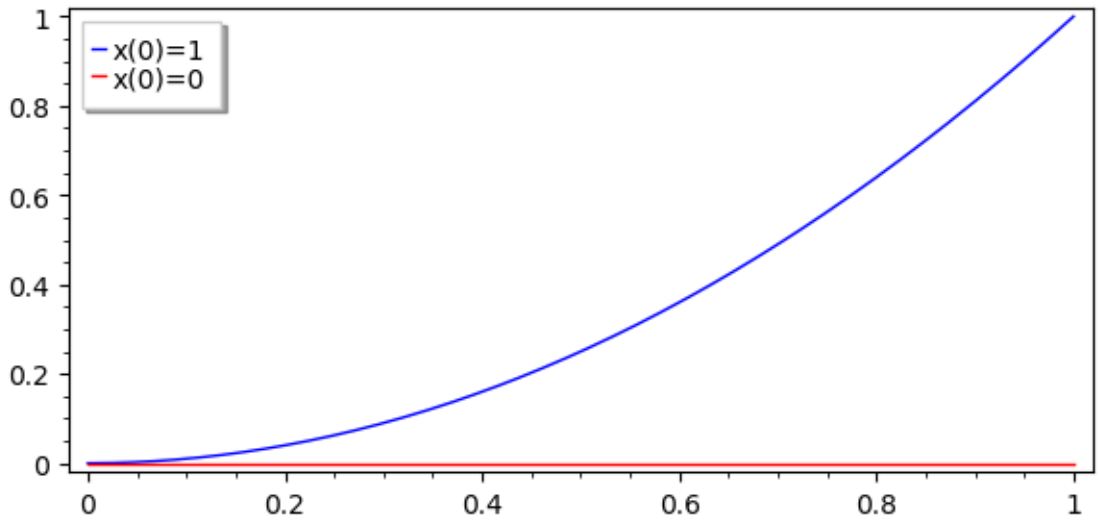
$$x(t) = x_1(t) = t^2, \qquad x(0) = 0$$

The second solution reads:

$$x(t) = x_2(t) = 0, \qquad x(0) = 0$$

In this case, there is not a unique solution. If this differential equation describes a real process, it would be impossible to predict the future evolution of the system!

[3]:
```
var('t')
p1=plot(t*t,(t,0,1),figsize=(6, 3), legend_label="x(0)=1", color='blue' )
p2=plot(0,(t,0,1),figsize=(6, 3), legend_label="x(0)=0", color='red' )
show(p1+p2, frame=True, axes=False)
```



Why in the last example there are two solutions, not only one solution? It is behaviour of the function $\sqrt{x}$ which is in the right hand side of this equation. For the uniqness the function should satisfied the so-called Lipschitz condition.

We say that the function $f(x)$ satisfies the Lipschitz condition on the open set U if there is such a constant $L > 0$ that

$$|f(x) - f(y)| \leq L|x - y|$$

for all $x, y \in U$. The Lipschitz condition can be written as

$$|f(x+h) - f(x)| \leq Lh \qquad \text{or as} \qquad \frac{f(x+h) - f(x)}{h}| \leq L$$

It follows that if $f(x)$ has a finite value of derivative (the derivative at a given point is bounded), then it satisfies the Lipschitz condition. There are of course non-fifferentialbe functions that satisfy the Lipschitz condition.

Picard's theorem: If the function $f(x)$ is continuous in U and satisfies the Lipschtz condition in U then the differential equation

$$\frac{dx}{dt} = f(x), \qquad x(0) = x_0$$

has exactly one solution in U. There are several versions of this theorem, but for us this simplest version is enough.

In the case of a system of differential equations, the Lipschitz condition has the form

$$|F_i(x_1, x_2, x_3, ..., x_n) - F_i(y_1, y_2, y_3, ..., y_n)| \leq L \sum_{k=1}^{n} |x_k - y_k|$$

This inequality has to be fulfilled for all functions F_i and Picard's theorem is formulated in a similar way. The Lipschitz condition is satisfied if partial derivatives are baunded,

$$|\frac{\partial F_i}{\partial x_k}| \leq K$$

for positive K.

Now we can answer why in the third example the solution is non-uniques: the function $f(x) = 2\sqrt{x}$ does not satisfy the Lipschitz condition at the point $x = 0$ because the derivative

$$\frac{df(x)}{dx} = \frac{1}{\sqrt{x}}$$

at this point $x = 0$ is divergent. At points $x > 0$ the derivative is bounded and Picard's theorem is satisfied. Therefore, solutions are unique for all initial conditions $x(0) > 0$.

[ ]:

2.2 PHASE SPACE

Once again, we rewrite (1) differential equations in notation:

$$\frac{dx_1}{dt} = F_1(x_1, x_2, x_3, ..., x_n)$$

$$\frac{dx_2}{dt} = F_2(x_1, x_2, x_3, ..., x_n)$$

.....

$$\frac{dx_n}{dt} = F_n(x_1, x_2, x_3, ..., x_n)$$

The above system of differential equations defines a certain dynamic system (the mathematical definition of a dynamic system can be very abstract, but for our needs, what we have written is enough). The set of all possible $\{x_1, x_2, x_3, ..., x_n\}$ values creates a set which we call the phase space of the (2) system. The dimension of this space is n, which is the same as there are differential equations. Depending on the context, we will use different notations for the same equations. Examples:

1. One differential equation. Usually we will use the notation

$$\frac{dx}{dt} = \dot{x} = f(x)$$

The phase space is 1-dimensional.

2. Two differential equations. Usually we will use the notation

$$\frac{dx}{dt} = \dot{x} = f(x, y)$$

$$\frac{dy}{dt} = \dot{y} = g(x, y)$$

The phase space is 2-dimensional.

3. Three differential equations. Usually we will use the notation

$$\frac{dx}{dt} = \dot{x} = f(x, y, z)$$

$$\frac{dy}{dt} = \dot{y} = g(x, y, z)$$

$$\frac{dz}{dt} = \dot{z} = h(x, y, z)$$

The phase space is 3-dimensional.

4. The Newton's equation for a particle moving only along the OX axis and which is driven by the force $F(x)$ dependent only on the particle coordinate:

$$m\ddot{x} = F(x)$$

where m is the mass of the particle. This is a 2-nd order differential equation and it is equivalent to a system of 2 first-order differential equations:

$$\dot{x} = v$$

$$\dot{v} = \frac{1}{m}F(x)$$

The phase space is 2-dimensional: it is a set of all possible positions and velocities of the particle, $\{x, v\}$. Despite its simplicity, this model is extremely important. It provides a starting point for understanding many crucial aspects of dynamical systems.

2.3 Geometric properties of phase space

2.3.1 Phase curves

To avoid abstract definitions at this stage, as an example we consider a 2-dimensional dynamical system defined by a set of differential equations:

$$\dot{x} = f(x, y), \qquad x(0) = x_0$$

$$\dot{y} = g(x, y), \qquad y(0) = y_0$$

The phase space of this system is 2-dimensional. It can be a plane or part of it. But it can be a more complicated 2-dimensional set. For example, it can be a sphere (similar to the surface of a ball), it can be a torus (similar to a bicycle inner tube). These can be even more complicated 2-dimensional objects. But for our purposes it is enough to consider the plane. We introduce a Cartesian coordinate system with the OX and OY axes on the plane. The initial condition $\{x_0 = x(0), y_0 = y(0)\}$ is the point on this plane. We solve the above system of differential equations numerically using the definition of serivative and applying the simplest scheme:

$$\frac{x(t+h) - x(t)}{h} = f(x(t), y(t))$$

$$\frac{y(t+h) - y(t)}{h} = g(x(t), y(t))$$

We will re-write these equation in the form

$$x(t+h) = x(t) + f(x(t), y(t))h$$

$$y(t+h) = y(t) + g(x(t), y(t))h$$

I. Numerical calculations must be started with the initial condition at the moment $t = 0$, i.e. in the first step we calculate

$$x_1 = x(h) = x(0) + f(x(0), y(0))h$$

$$y_1 = y(h) = y(0) + g(x(0), y(0))h$$

On the plane we get a point with coordinates $\{x_1, y_1\}$. Let's mark it on the plane. We now have 2 points:

$$\{x_0, y_0\}, \qquad \{x_1, y_1\}$$

II. In the next step we choose $t = h$ time:

$$x_2 = x(h+h) = x(2h) = x(h) + f(x(h), y(h))h$$

$$y_2 = y(h+h) = y(2h) = y(h) + g(x(h), y(h))h$$

We will use the above notation: $x_1 = x(h), y_1 = y(h)$ and write these equations in the form

$$x_2 = x_1 + f(x_1, y_1)h$$

$$y_2 = y_1 + g(x_1, y_1)h$$

On the plane we get a point with coordinates $\{x_2, y_2\}$. Let's mark it on the plane. We now have 3 points:

$$\{x_0, y_0\}, \qquad \{x_1, y_1\}, \qquad \{x_2, y_2\}$$

III. We can see immediately that in step 3 we get the equations

$$x_3 = x_2 + f(x_2, y_2)h$$

$$y_3 = y_2 + g(x_2, y_2)h$$

and we get the point with coordinates $\{x_3, y_3\}$.

IV. We notice that in the n-th step we get equations

$$x_n = x_{n-1} + f(x_{n-1}, y_{n-1})h$$

$$y_n = y_{n-1} + g(x_{n-1}, y_{n-1})h$$

V. More often you write what you get in the next step, i.e. n + 1:

$$x_{n+1} = x_n + f(x_n, y_n)h$$

$$y_{n+1} = y_n + g(x_n, y_n)h$$

We obtain recurrence equations that allow us to determine the evolution of the system, i.e. the numerical solution of the system. On the XY plane we get a collections of points with coordinates

$$\{x_n, y_n\}$$

If the time step h is small enough, we connect this series of points with a solid line and we get a curve on the plane. This curve is called the phase curve of the dynamical system. It is one of the most important geometric object related to the system of differential equations. Having drawn such a phase curve, we can conclude about the evolution of the system and the characteristics of the system behavior. Below we present two examples: phase curves for a harmonic oscillator and a damped harmonic oscillator.

A. HARMONIC OSCILLATOR For the harmonic oscillator, the force acting on the particle has the form

$$F = -kx$$

where k is characterized by "springiness" of the spring. The Newton equation then reads:

$$m\ddot{x} = -kx, \qquad \text{lub w postaci} \qquad \ddot{x} = -(k/m)x = -\omega^2 x$$

where $\omega^2 = k/m$. This second order al differentiequation is equivalent to 2 first order equations:

$$\dot{x} = y, \qquad x(0) = x_0$$

$$\dot{y} = -\omega^2 x, \qquad y(0) = y_0$$

A. DAMPED HARMONIC OSCILLATOR The previous example of the harmonic oscillator is rather idealized bacause it is assumed that tha particle moves in a vacuum. In real situation, it moves in in air, water or other liquid. Then the particle is affected by an additional force, namely the friction (damping) force F_d. The friction force is proportional to the velocity of the particle and opposite to the direction of motion, i.e.,

$$F_d = -\gamma_0 v = -\gamma_0 \dot{x}$$

where γ_0 is called the friction (or damping) coefficient. In such a case, the Newton equation takes the form

$$m\ddot{x} = -kx - \gamma_0 \dot{x}, \qquad \text{or in the form} \qquad \ddot{x} = -\frac{k}{m}x - \frac{\gamma_0}{m}x = -\omega^2 x - \gamma \dot{x}$$

where $\omega^2 = k/m$ and $\gamma = \gamma_0/m$. The above equation is equivalent to 2 first-order differential equations:

$$\dot{x} = y, \qquad x(0) = x_0$$

$$\dot{y} = -\gamma y - \omega^2 x, \qquad y(0) = y_0$$

Of course, when $\gamma = 0$, then we get the harmonic oscillator equation without friction (undamped motion). Below the phase curves for these 2 examples are depicted.

Harmonic oscillator without friction

[4]:
```
var('x y')

def schemat_eulera2D(vec, ics, Tlist):
    i = 0
    dx = [ics[0]]
    dy = [ics[1]]
    h = Tlist[i+1] - Tlist[i]

    iks(x,y) = vec[0R]*h
    igrek(x,y) = vec[1R]*h

    for time in Tlist:
        dx.append(dx[i] + iks(dx[i],dy[i]))
        dy.append(dy[i] + igrek(dx[i],dy[i]))
        i += 1

    return zip(dx,dy)
```

[5]:
```
@interact(layout={'top':[['_omega','x0','y0']],'bottom':[['T','h']]})
```

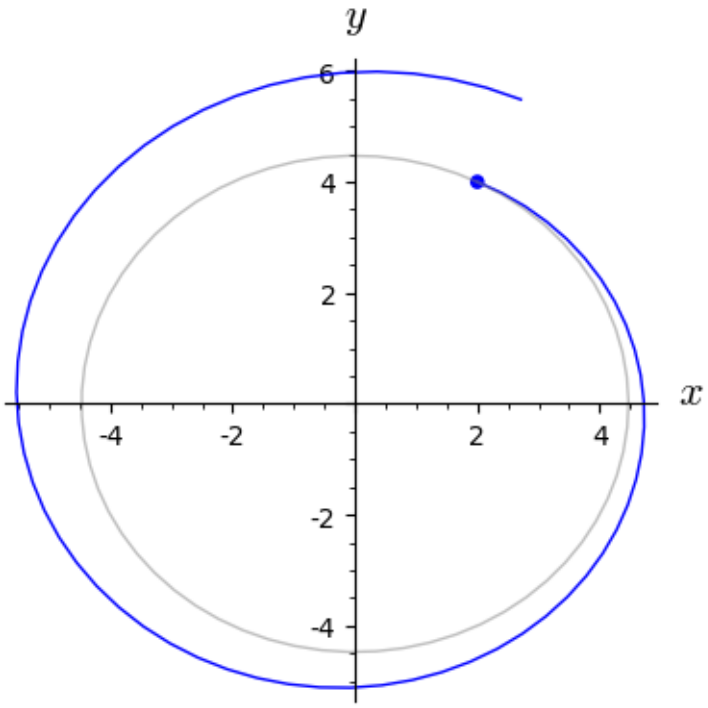
```
def _(title=['a','b'], h=selector(['0.005','0.01','0.05','0.1','0.5','1'],␣
↪default='0.1', buttons=True),x0=input_box(2,label=r'$x_0$', width=10),␣
↪y0=input_box(4,label=r'$y_0$', width=10), T=input_box(0, width=10),␣
↪_omega=input_box(1,label=r'$\omega$', width=10)):
    global oscylator_nietlumiony, background

    if T == 0:
        T = 2*pi/_omega

    listT = srange(0,T,float(h), include_endpoint=True)
    background = desolve_odeint(vector([y,-_omega^2*x]), [x0, y0], srange(0,T+0.
↪1,0.1,include_endpoint=True), [x,y])
    oscylator_nietlumiony = schemat_eulera2D([y,-_omega^2*x], [x0, y0], listT)

    print r'Parametry modelu'
    html(r'$\omega=%s, x_0=%s, y_0=%s$'%(_omega,x0,y0))
    print r'Parametry symulacji'
    html(r'$h=%s, T=%s$'%(h,T))
    print '\nDla T=0 lista generowana jest automatycznie dla jednego okresu␣
↪wasnego oscylatora'
```

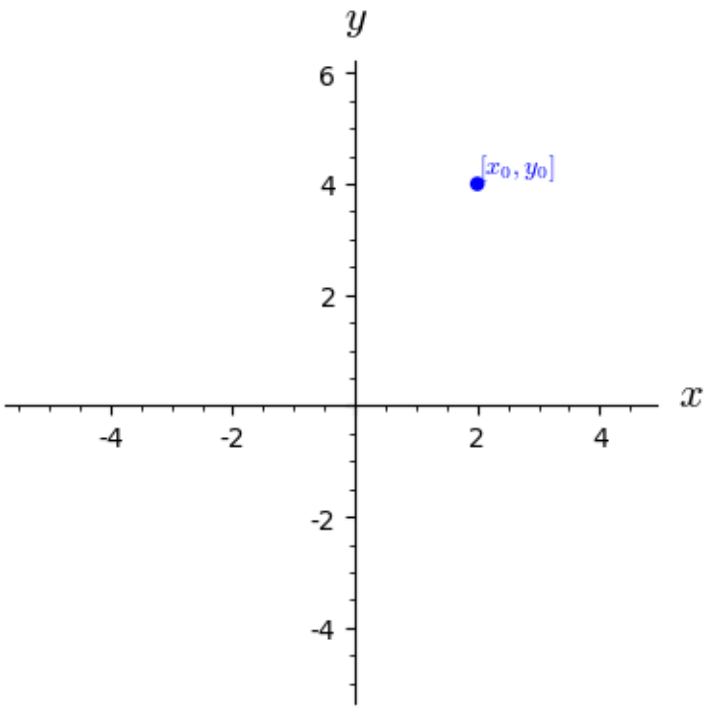
```
Parametry modelu
Parametry symulacji

Dla T=0 lista generowana jest automatycznie dla jednego okresu wasnego
oscylatora
```

[6]:
```
@interact
def _(krok=slider(1, len(oscylator_nietlumiony), 1, default=1, label=r'krok')):

    buf = zip(*oscylator_nietlumiony)
    minx, maxx, miny, maxy = min(buf[0]), max(buf[0]), min(buf[1]), max(buf[1])
    kroki = oscylator_nietlumiony[:krok]
    kroki_plot = list_plot(kroki, figsize=(4,4), axes_labels=[r'$x$',r'$y$'],␣
↪size=30, xmin=minx, xmax=maxx, ymin=miny, ymax=maxy)

    txt_plot =␣
↪text(r'$[x_0,y_0]$',kroki[0],vertical_alignment='bottom',horizontal_alignment='left')
    for i in range(1,len(kroki)):
        txt_plot +=␣
↪text(r'$[x_{%d},y_{%d}]$'%(i,i),kroki[i],vertical_alignment='bottom',horizontal_alignmen

    full_plot = list_plot(oscylator_nietlumiony, plotjoined=True, figsize=(4,4),␣
↪axes_labels=[r'$x$',r'$y$'])
    full_plot += list_plot(background.tolist(), plotjoined=True, color='grey',␣
↪alpha=0.5)
    print("krzywe fazowe dla oscylatora bez tarcia")
    (full_plot+kroki_plot).show()
```

```
    (kroki_plot+txt_plot).show()
```

krzywe fazowe dla oscylatora bez tarcia





In the case of an ideal (not damped) oscillator, the phase curves are closed and it means that the motion is periodic. The period of such periodic motion is the time needed for the particle starting from a point e.g. $\{x_0, y_0\}$ and moving along a phase curve to reach the same point $\{x_0, y_0\}$ again. In the case of a damped oscillator, the phase curve is a spiral that winds up to $\{0, 0\}$. Spiral motion means that both x and $v = y$ decrease over time and for long times the location of x and the speed of v go to zero, i.e. the particle slows down and finally stops. This is damped motion: the vibration amplitude decreases over time. One can observe more complex phase curves but the general rule is: when x grows, it means the particle's position increases. When y decreases, it means that the particle velocity decreases. When x decreases, the position coordinate of the particle decreases. As y increases, the particle velocity increases.

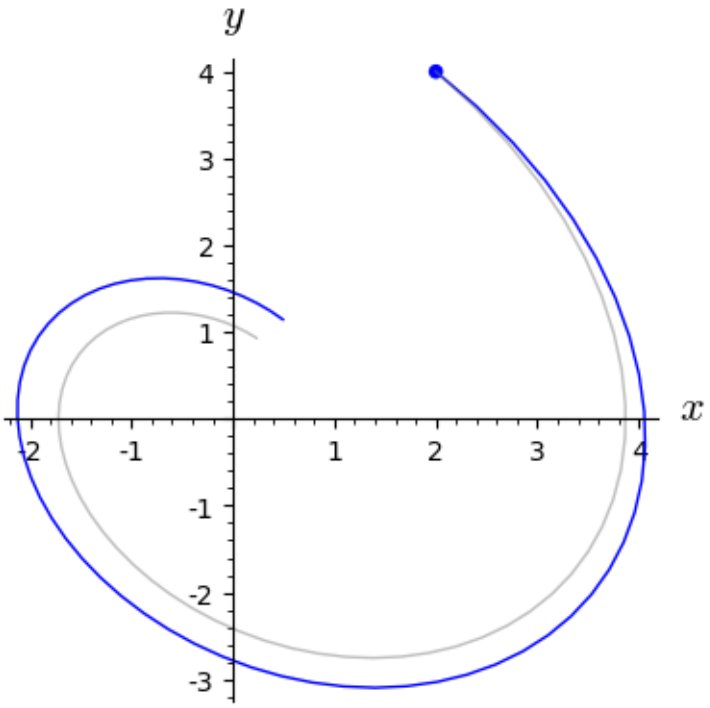
[7]:
```
@interact(layout={'top':[['_omega','_gamma','x0','y0']],'bottom':[['T','h']]})
def _(title=['a','b'], h=selector(['0.05','0.01','0.1','0.5','1'], default='0.1',␣
 ↪buttons=True),x0=input_box(2,label=r'$x_0$', width=10),␣
 ↪y0=input_box(4,label=r'$y_0$', width=10), T=input_box(0, width=10),␣
 ↪_omega=input_box(1,label=r'$\omega$', width=10), _gamma=input_box(0.
 ↪5,label=r'$\gamma$', width=10)):
    global oscylator_tlumiony, glob_gamma, glob_omega, background2
    glob_gamma = _gamma
    glob_omega = _omega

    if T == 0:
        T = 2*pi/_omega

    listT = srange(0,T,float(h),include_endpoint=True)
```

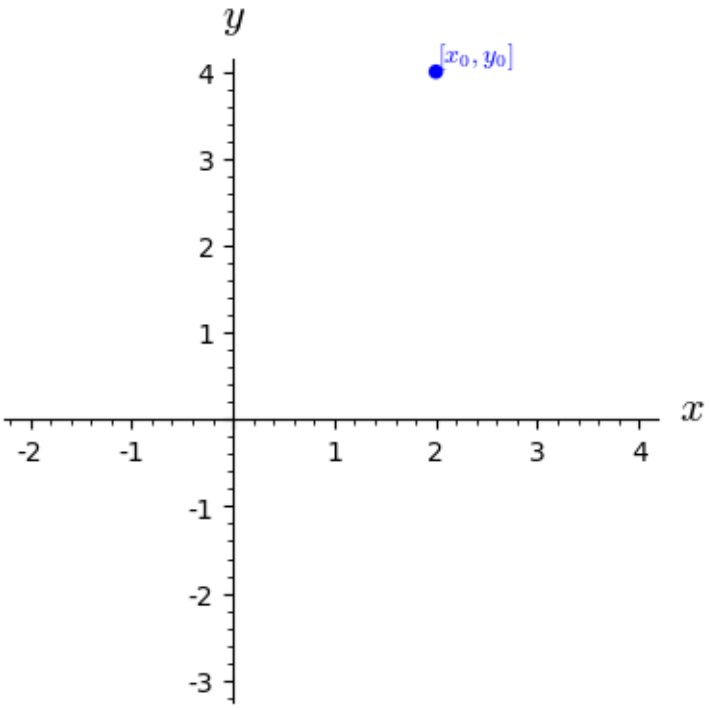
```
    background2 = desolve_odeint(vector([y,-_omega^2*x-_gamma*y]), [x0, y0],␣
↪srange(0,2*pi/_omega+0.1,0.1,include_endpoint=True), [x,y])
    oscylator_tlumiony = schemat_eulera2D([y,-_omega^2*x-_gamma*y], [x0, y0],␣
↪listT)

    print r'Parametry modelu'
    html(r'$\gamma=%s, \omega=%s, x_0=%s, y_0=%s$'%(_gamma,_omega,x0,y0))
    print r'Parametry symulacji'
    html(r'$h=%s, T=%s$'%(h,T))

    print '\nDla T=0 lista generowana jest automatycznie dla jednego okresu␣
↪wasnego oscylatora'
```

```
Parametry modelu
Parametry symulacji

Dla T=0 lista generowana jest automatycznie dla jednego okresu wasnego
oscylatora
```

[8]:
```
@interact
def _(krok=slider(1, len(oscylator_tlumiony), 1, default=1, label=r'krok')):

    buf = zip(*oscylator_tlumiony)
    minx, maxx, miny, maxy = min(buf[0]), max(buf[0]), min(buf[1]), max(buf[1])
    kroki = oscylator_tlumiony[:krok]
    kroki_plot = list_plot(kroki, figsize=(4,4), axes_labels=[r'$x$',r'$y$'],␣
↪size=30, xmin=minx, xmax=maxx, ymin=miny, ymax=maxy)

    txt_plot =␣
↪text(r'$[x_0,y_0]$',kroki[0],vertical_alignment='bottom',horizontal_alignment='left')
    for i in range(1,len(kroki)):
        txt_plot +=␣
↪text(r'$[x_{%d},y_{%d}]$'%(i,i),kroki[i],vertical_alignment='bottom',horizontal_alignmen

    full_plot = list_plot(oscylator_tlumiony, plotjoined=True, figsize=(4,4),␣
↪axes_labels=[r'$x$',r'$y$'])
    full_plot += list_plot(background2.tolist(), plotjoined=True, color='grey',␣
↪alpha=0.5)

    print("krzywe fazowe dla oscylatora tumionego")
    (full_plot+kroki_plot).show()
    (kroki_plot+txt_plot).show()
```

```
krzywe fazowe dla oscylatora tumionego
```


y
x
4
3
2
1
-1
-2
-3
-2
-1
1
2
3
4


y
x
$[x_0, y_0]$
4
3
2
1
-1
-2
-3
-2
-1
1
2
3
4

2.3.2 VECTOR FIELD

Right sides of the system of differential equations

$$\dot{x} = f(x,y), \qquad x(0) = x_0$$

$$\dot{y} = g(x,y), \qquad y(0) = y_0$$

can be treated as components of a certain vector field:

$$\vec{F} = [F_x, F_y] = [f(x,y), g(x,y)]$$

At each point $\{x,y\}$ of the coordinate plane we draw a vector with components $[f(x,y), g(x,y)]$. In this way we get a vector field. Well, but what information can be obtained from this vector field. Let's do the following numerical exercise: We start from the initial condition $\{x_0, y_0\}$ and draw a vector with $[f(x_0,y_0), g(x_0,y_0)]$ components at this point:

$$\text{at the point} \quad \{x_0, y_0\} \quad \text{we draw a vector of components} \quad [f(x_0,y_0), g(x_0,y_0)]$$

As before, we numerically solve the system of differential equations and calculate $\{x_1, y_1\}$:

$$\text{at the point} \quad \{x_1, y_1\} \quad \text{we draw a vector of components} \quad [f(x_1,y_1), g(x_1,y_1)]$$

Then we calculate $\{x_2, y_2\}$:

$$\text{at the point} \quad \{x_2, y_2\} \quad \text{we draw a vector of components} \quad [f(x_2,y_2), g(x_2,y_2)]$$

In the nth step of the iteration, we calculate $\{x_n, y_n\}$:

$$\text{at the point} \quad \{x_n, y_n\} \quad \text{we draw a vector of components} \quad [f(x_n,y_n), g(x_n,y_n)]$$

Because all of the above points $\{x_i, y_i\}$ are located on the phase curve, the vector $[f(x_i,y_i), g(x_i,y_i)]$ is attached to the phase curve. We note that these vectors are tangent to the phase curve. If $\{x_i, y_i\}$ has interpretations of the position of the particle, then the vector $[f(x_i,y_i), g(x_i,y_i)]$ can be interpreted as components of the volocity vector because $\dot{x} = f(x,y)$ and $\dot{y} = g(x,y)$. We know that $\dot{x} = v_x$ is the x-component, while $\dot{y} = v_y$ is the y-component of the velocity. In other words, the resulting vector field is the velocity field of the fictitious particle.

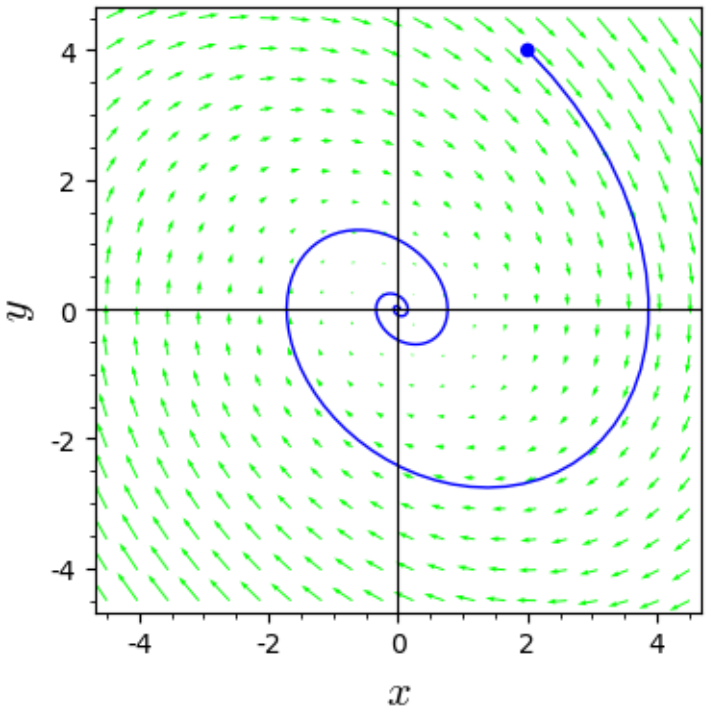
[9]:
```
@interact(layout={'top':[['_omega','_gamma','x0','y0']],'bottom':[['T','h']]})
def _(title=['a','b'], h=selector(['0.05','0.01','0.1','0.5','1'], default='0.1',␣
 ↪buttons=True),x0=input_box(2,label=r'$x_0$', width=10),␣
 ↪y0=input_box(4,label=r'$y_0$', width=10), T=input_box(20, width=10),␣
 ↪_omega=input_box(1,label=r'$\omega$', width=10), _gamma=input_box(0.
 ↪5,label=r'$\gamma$', width=10)):
    global oscylator_tlumiony, glob_gamma, glob_omega
```

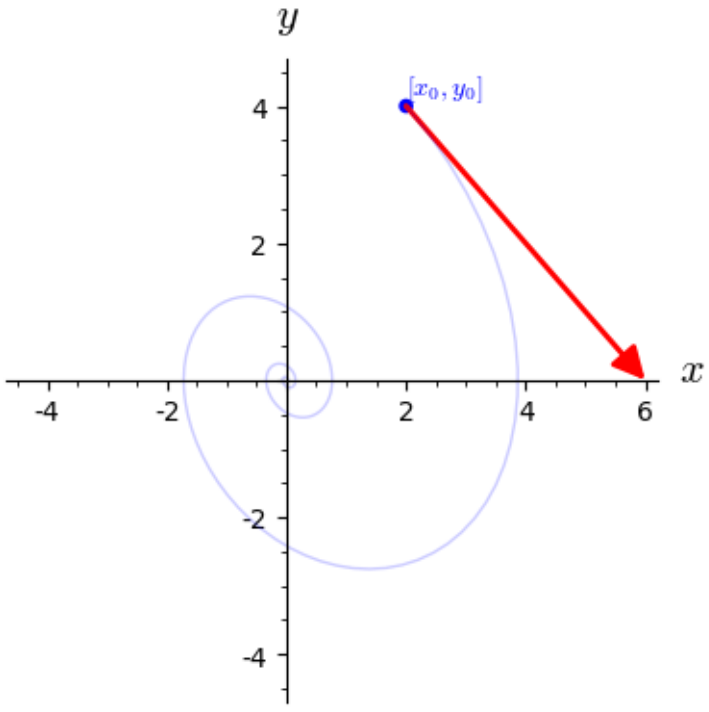
```
    glob_gamma = _gamma
    glob_omega = _omega

    listT = srange(0,T,float(h))
    oscylator_tlumiony = desolve_odeint(vector([y,-_omega^2*x-_gamma*y]), [x0,␣
↪y0], listT, [x,y])

    print r'Parametry modelu'
    html(r'$\gamma=%s, \omega=%s, x_0=%s, y_0=%s$'%(_gamma,_omega,x0,y0))
    print r'Parametry symulacji'
    html(r'$h=%s, T=%s$'%(h,T))
```

```
Parametry modelu
Parametry symulacji
```

[10]:
```
vf = lambda u,a,b: (u[0]+u[1],u[1]-a*u[0]-b*u[1])

@interact
def _(krok=slider(1, len(oscylator_tlumiony), 1, default=1, label=r'krok')):
    kroki = oscylator_tlumiony[:krok]
    kroki_plot = list_plot(kroki.tolist(), figsize=(4,4),␣
↪axes_labels=[r'$x$',r'$y$'], size=30, xmin=-4.5, xmax=4.5, ymin=-4.5, ymax=4.5)

    pole_wektorowe =␣
↪arrow(kroki[0],vf(kroki[0],glob_omega^2,glob_gamma),color='red',xmax=vf(kroki[0],glob_om
    for krok in kroki[1:]:
        pole_wektorowe +=␣
↪arrow(krok,vf(krok,glob_omega^2,glob_gamma),color='red', width=.4)

    txt_plot =␣
↪text(r'$[x_0,y_0]$',kroki[0],vertical_alignment='bottom',horizontal_alignment='left')
    for i in range(1,len(kroki)):
        txt_plot +=␣
↪text(r'$[x_{%d},y_{%d}]$'%(i,i),kroki[i],vertical_alignment='bottom',horizontal_alignmen

    shadowplot = list_plot(oscylator_tlumiony.tolist(), plotjoined=True,␣
↪figsize=(4,4), axes_labels=[r'$x$',r'$y$'], alpha=0.2)
    full_plot = list_plot(oscylator_tlumiony.tolist(), plotjoined=True,␣
↪figsize=(4,4), axes_labels=[r'$x$',r'$y$']) +␣
↪plot_vector_field([y,-glob_omega^2*x-glob_gamma*y], (x, -4.5, 4.5), (y, -4.5, 4.
↪5), plot_points=20, color='lime')

    print("krzywe fazowe dla oscylatora tumionego")
    (full_plot+kroki_plot).show()
    (shadowplot+kroki_plot+txt_plot+pole_wektorowe).show()
```

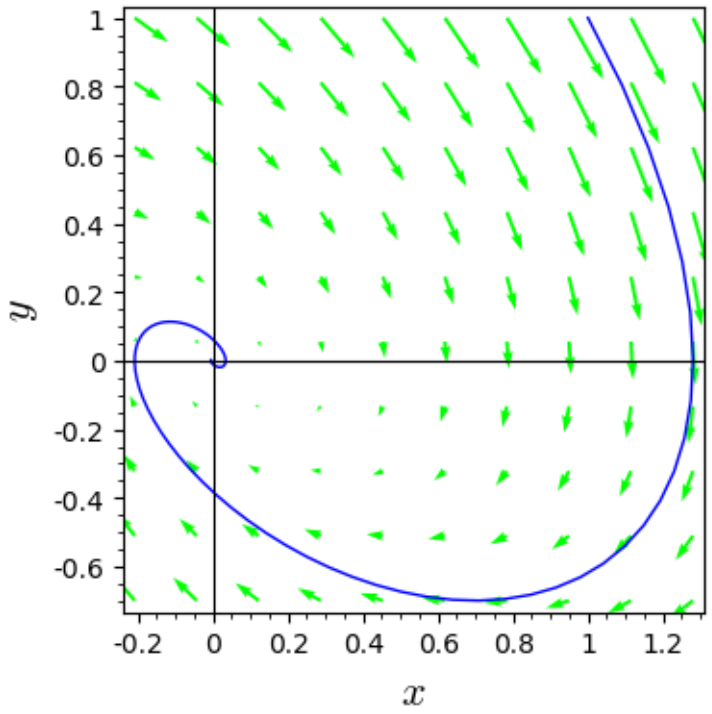
```
krzywe fazowe dla oscylatora tumionego
```


4
2
0
-2
-4
y
-4
-2
0
2
4
x


y
$[x_0, y_0]$
4
2
-2
-4
-4
-2
2
4
6
x

Below you will find a cell in which we encourage everyone to experiment with different models. Have fun...

[11]:
```
#########
# Model #
#########

# zmienne
var('x y')

# parametry
# UWAGA: jeeli Twój model bdzie zaleny od innych parametrów
#        tu wanie musisz je wszystkie wyspecyfikowa
alpha = 1

# warunki pocztkowe
x0 = 1
y0 = 1

# model
dx = y
dy = -alpha*x - y

# czas (T) i krok (h) symulacji
T = 12
h = 0.1


######################################################
# Symulacje + wizualizacja                           #
######################################################
listT = srange(0,T,float(h), include_endpoint=True)
numeryka = desolve_odeint(vector([dx, dy]), [x0, y0], listT, [x,y])

przestrzen_fazowa = list_plot(numeryka.tolist(), plotjoined=True, figsize=(4,4),␣
 ↪axes_labels=[r'$x$',r'$y$'])
pole_wektorowe = plot_vector_field([dx,dy], (x, numeryka[:,0].min(),\
                                             numeryka[:,0].max()), \
                                    (y, numeryka[:,1].min(), numeryka[:,1].max()),␣
 ↪plot_points=10, color='lime')

show(przestrzen_fazowa+pole_wektorowe)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
y
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
x

3 Attractors in dynamical systems

3.1 What is an attractor?

An attractor A is such a set in the phase space of the dynamical system that many trajectories starting even very far from this set tends this set A when $t \to \infty$. As an example, let us consider a damped oscillator. Starting form arbitrary initial conditions $\{x(0), v(0)\}$, the particle always tends to the point $\{0,0\}$ which is an attractor for the damped oscillator. In this case, the attractor is a point in the phase space: $A = \{0,0\}$. It attracts trajectories with different initial conditions. But it is not necessary to attract all trajectories. For a given dynamical system there may be many attractors, even infinitely many. Attractors may have a simple structure: this may be a point, a few points, a curve such as a circle or a deformed ellipse, a part of the plane, a torus, a part of space. Attractors can also have a complicated structure: it can be a fractal set, i.e. a set with a non-integer dimension, e.g. 0.63, 2.06. Such an attractor is called a strange attractor.

The next notion is the basins of attraction B for the attractor A. It is the set:

$$B(A) = \{\vec{x}_0 : \lim_{t\to\infty} \vec{x}(t;\vec{x}_0) \in A\}$$

where $\vec{x}(t;x_0)$ is the trajectory starting from the initial condition $\vec{x}_0$, e.g. the solution of the system of differential equations with the corresponding initial conditions $\vec{x}_0$. So, the basin of attraction for the attractor A is a set of all initial conditions for which the corresponding trajectories tend to A when $t \to \infty$.

3.1.1 EXAMPLE 1: Damped harmonic oscillator

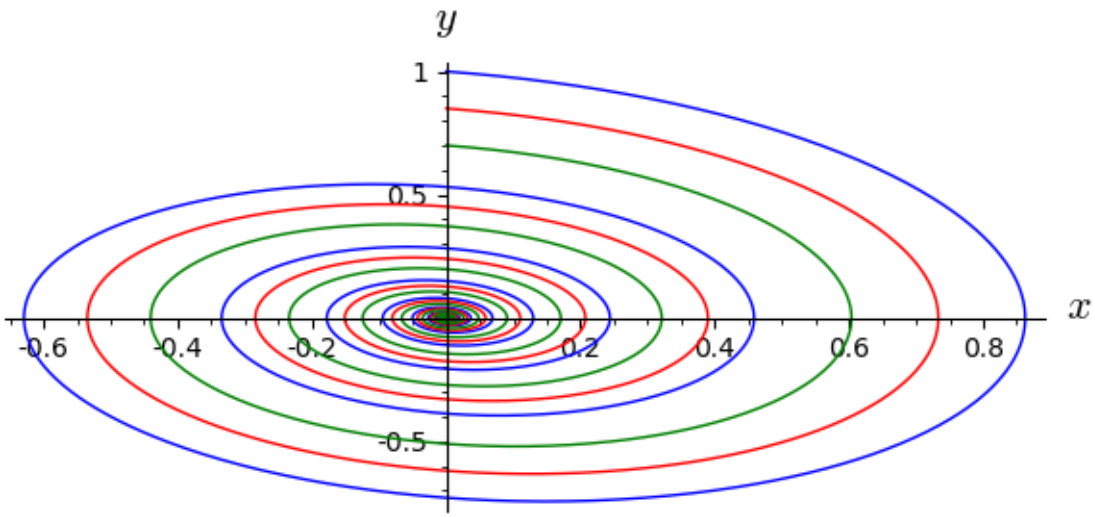
$$\ddot{x} = -\omega^2 x - \gamma\dot{x}, \qquad V(x) = \frac{1}{2}\omega^2 x^2$$

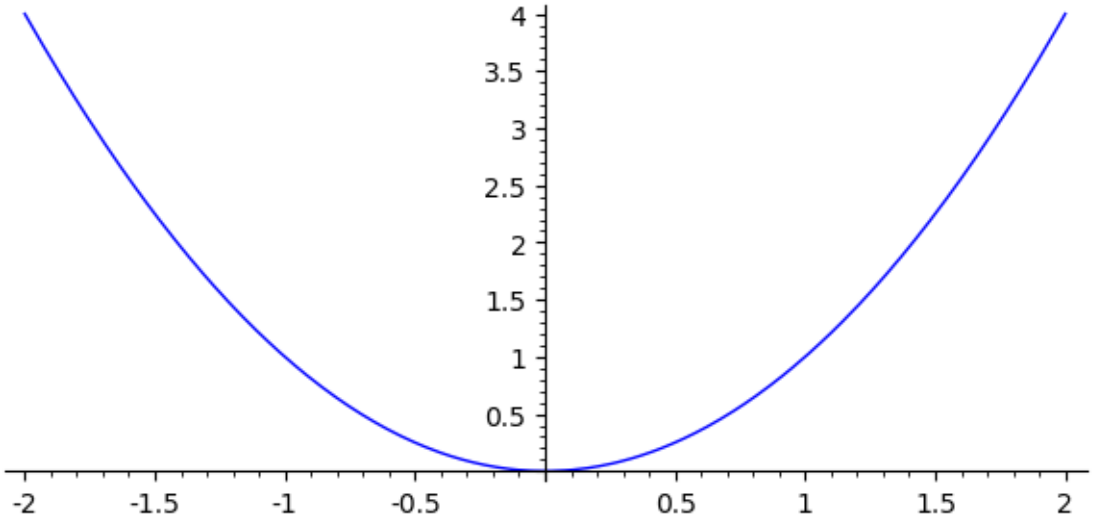
There is only one attractor: this is the point $A = (0,0)$. The basin of attraction is the whole phase plane $B = \{x, v\}$. is the attraction pool.

[1]:
```
var('x,y')
T = srange(0,50,0.01)
sol1 = desolve_odeint(vector([y,-x -0.2*y]), [0,1], T, [x,y]).tolist()
sol2 = desolve_odeint(vector([y,-x -0.2*y]), [0,0.85], T, [x,y])
sol3 = desolve_odeint(vector([y,-x -0.2*y]), [0,0.7], T, [x,y])
p1 = plot(x^2, -2, 2,figsize=(6,3), )
g1 = list_plot(sol1, plotjoined=True, \
               figsize=(6,3),axes_labels=[r'$x$',r'$y$'])
g1 += list_plot(sol2.tolist(), plotjoined=True,\
                figsize=(6,3),color="red", axes_labels=[r'$x$',r'$y$'])
g1 += list_plot(sol3.tolist(), plotjoined=True,\
                figsize=(6,3),color="green", axes_labels=[r'$x$',r'$y$'])
table([["potencja kwadratowy","oscylator tumiony"],[p1,g1]])
print("all solutions tend to the point (0,0) which is an attractor for the damped␣
 ↪oscillator ")
```

```
#g1.save('sage_chI024_01.pdf')
#g1.save('sage_chI024_01.png')

g1.show(),p1.show()
```

all solutions tend to the point (0,0) which is an attractor for the damped oscillator





[1]: (None, None)

3.1.2 EXAMPLE 2: The non-linear (bistable) damped oscillator described by the Newton equation:

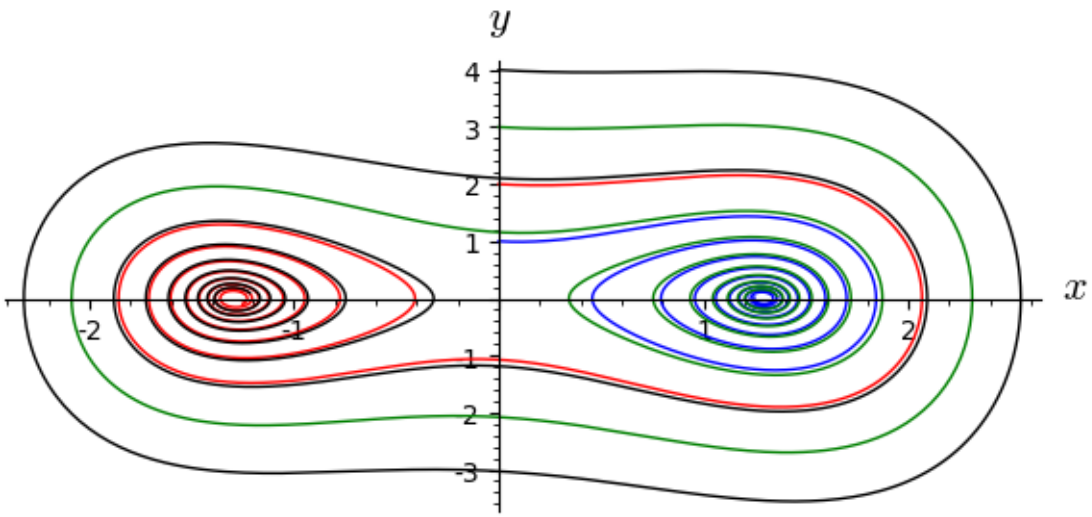
$$\ddot{x} = ax - bx^3 - \gamma\dot{x}, \qquad V(x) = \frac{1}{4}bx^4 - \frac{1}{2}ax^2$$

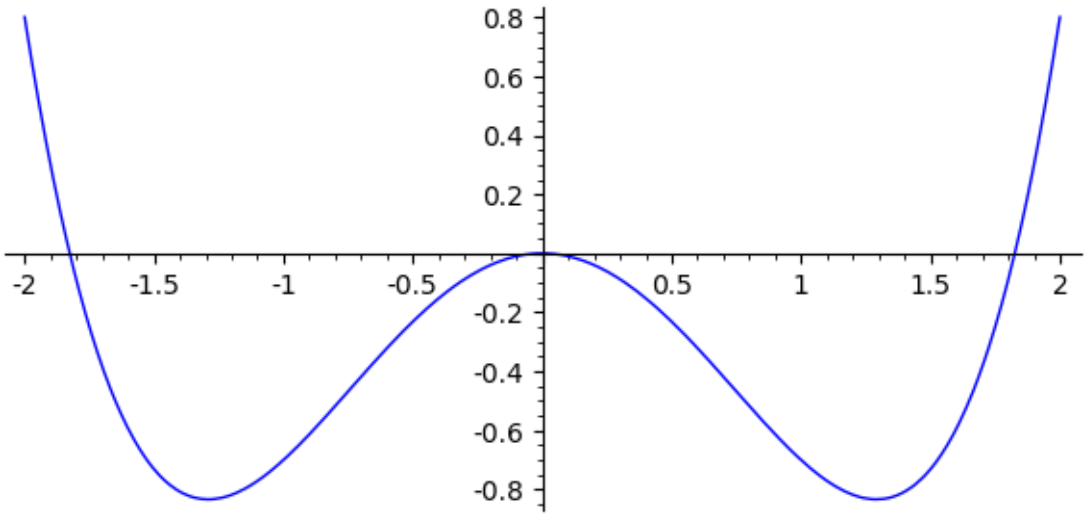
There are two attractors: $A_1 = (-x_s, 0)$ and the symmetric $A_2 = (x_s, 0)$, where x_s is the minimum of the bistable potential $V(x)$. The 2-dimensional phase space (i.e. the plane) is divided into 2 basins of attractions. . The transparent visualization is presented on our website:

http://visual.icse.us.edu.pl/wizualizacje/mechanika-teoretyczna/zobacz/BasenyPrzyciagania/

[2]:
```
var('x,y')
T1 = srange(0,30,0.01)
so1=desolve_odeint(vector([y,2*x-1.2*x^3 -0.2*y]), [0,1], T1, [x,y])
so2=desolve_odeint(vector([y,2*x-1.2*x^3 -0.2*y]), [0,2], T1, [x,y])
so3=desolve_odeint(vector([y,2*x-1.2*x^3-0.2*y]), [0,3], T1, [x,y])
so4=desolve_odeint(vector([y,2*x-1.2*x^3-0.2*y]), [0,4], T1, [x,y])
p11=plot(0.3*x^4 - x^2, -2, 2,figsize=(6,3), )
g11=list_plot(so1.tolist(), plotjoined=True,\
              figsize=(6,3),axes_labels=[r'$x$',r'$y$'])
g11 +=list_plot(so2.tolist(), plotjoined=True,\
                figsize=(6,3),color="red", axes_labels=[r'$x$',r'$y$'])
g11 +=list_plot(so3.tolist(), plotjoined=True,\
                figsize=(6,3),color="green", axes_labels=[r'$x$',r'$y$'])
g11 +=list_plot(so4.tolist(), plotjoined=True,\
                figsize=(6,3),color="black", axes_labels=[r'$x$',r'$y$'])
table([["potencja bistabilny","oscylator nieliniowy tumiony"]])
print(" solutions tend to the point $(-x_s,0)$ or to the point  $(x_s,0)$ ")

#g11.save('sage_chI024_02.pdf')
#g11.save('sage_chI024_02.png')

g11.show()
p11.show()
```

```
solutions tend to the point $(-x_s,0)$ or to the point  $(x_s,0)$
```



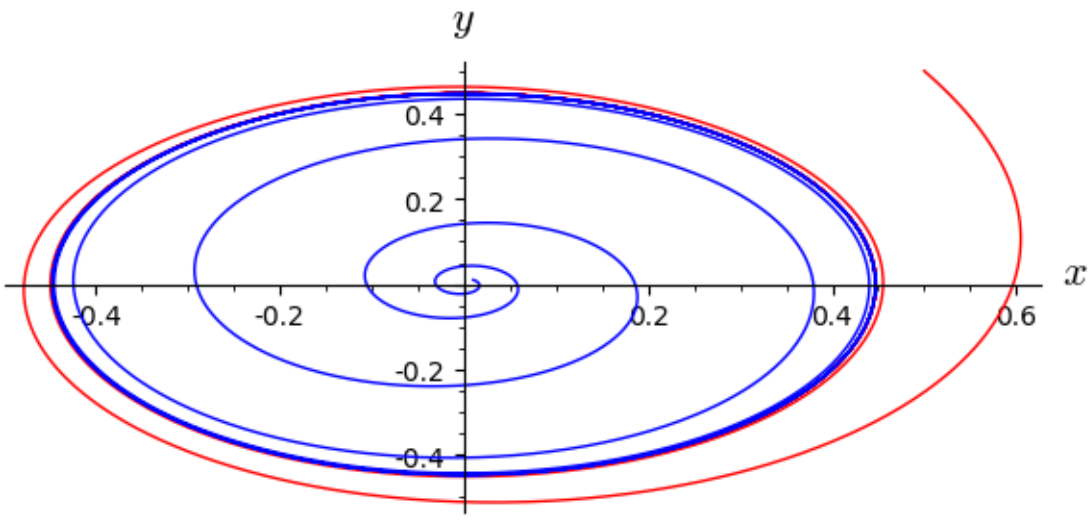


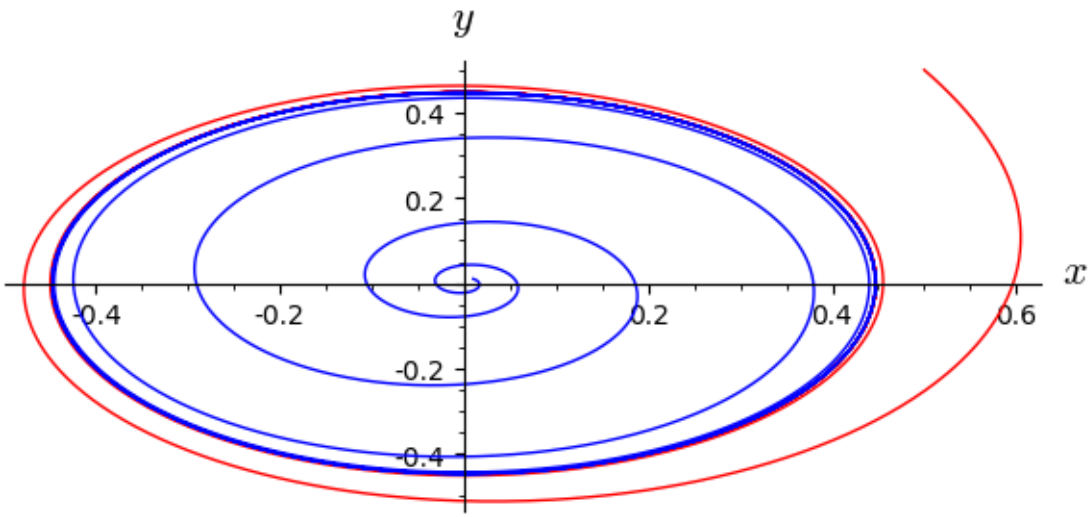
There are two basins of attractions and corresponding two attractors. The basins of attractions are more complicated: the plane is devided into two sets of initial conditions of the stripe structure.

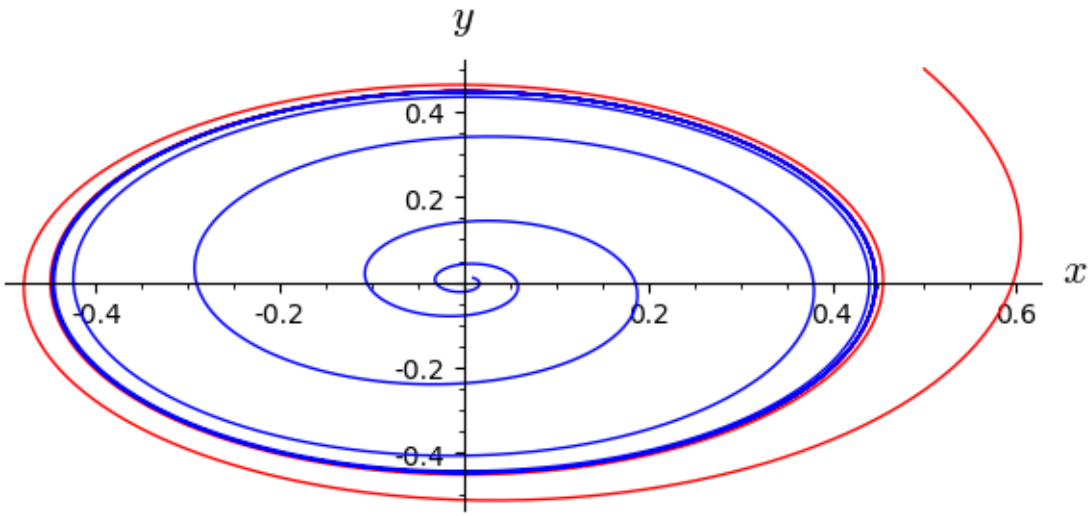
3.1.3 EXAMPLE 3: Limit cycle

The attractor is a closed curve (circle, ellipse, any other closed curves). Below are two examples taken from biological models.

[3]:
```
var('x,y')
T3 = srange(0,50,0.01)
de1=y+x*(0.2-(x^2+y^2))
de2=-x+y*(0.2-(x^2+y^2))
s1=desolve_odeint(vector([de1, de2]), [0.5,0.5], T3, [x,y])
s2=desolve_odeint(vector([de1, de2]), [0.01, 0.01], T3, [x,y])
h1=list_plot(s1.tolist(),\
             plotjoined=True, figsize=(6,3),\
             color="red",axes_labels=[r'$x$',r'$y$'])
h2=list_plot(s2.tolist(), \
             plotjoined=True, \
             figsize=(6,3),axes_labels=[r'$x$',r'$y$'])
show(h1+h2)


(h1+h2).show()#save('sage_chI024_03.pdf')
(h1+h2).show()#save('sage_chI024_03.png')
```

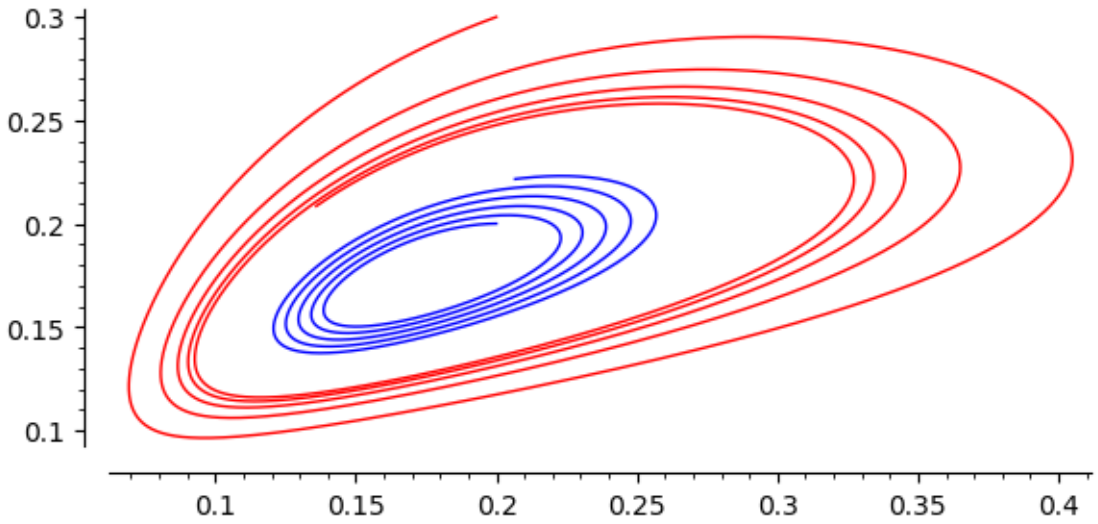


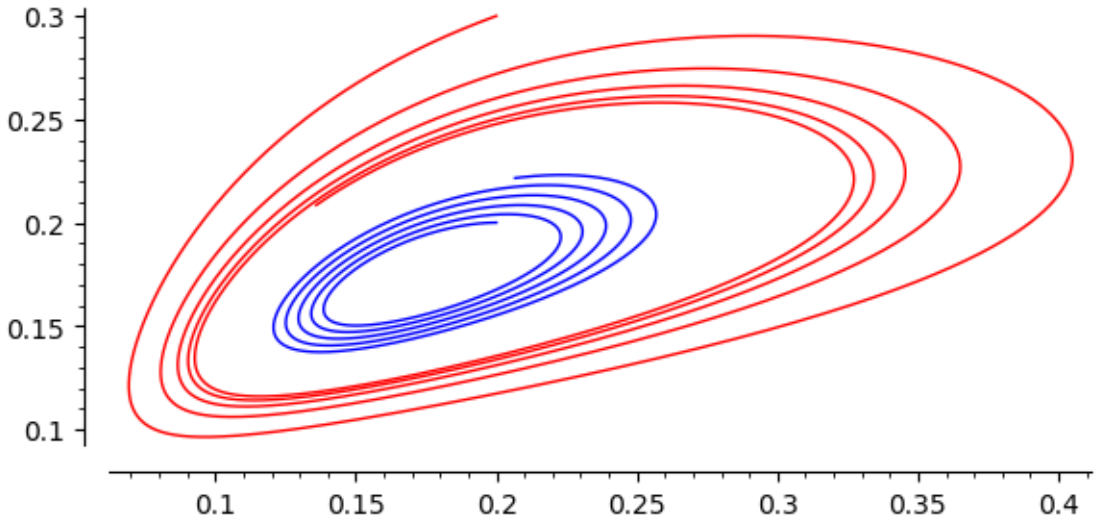


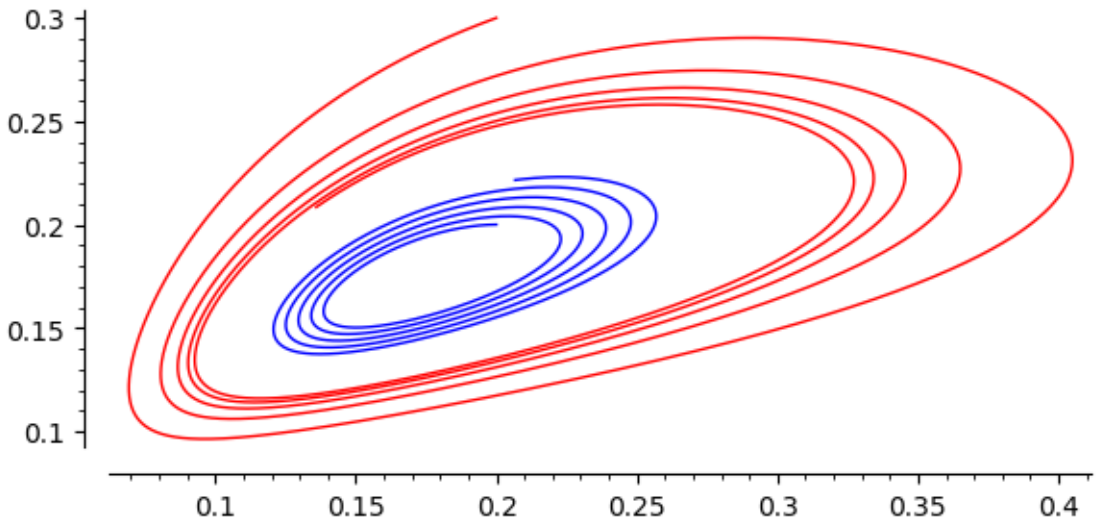


[4]:
```
a, b, d = 1.3, 0.33, 0.1
F(x,y)=x*(1-x) - a*x*y/(x+d)
G(x,y)= b*y*(1-y/x)
T = srange(0,80,0.01)
sl1=desolve_odeint(vector([F,G]), [0.2,0.3], T, [x,y])
sl2=desolve_odeint(vector([F,G]), [0.2,0.2], T, [x,y])
j1=list_plot(sl1.tolist(), plotjoined=True, color="red", figsize=(6, 3))
j2=list_plot(sl2.tolist(), plotjoined=True,  figsize=(6, 3))
show(j1+j2)
```

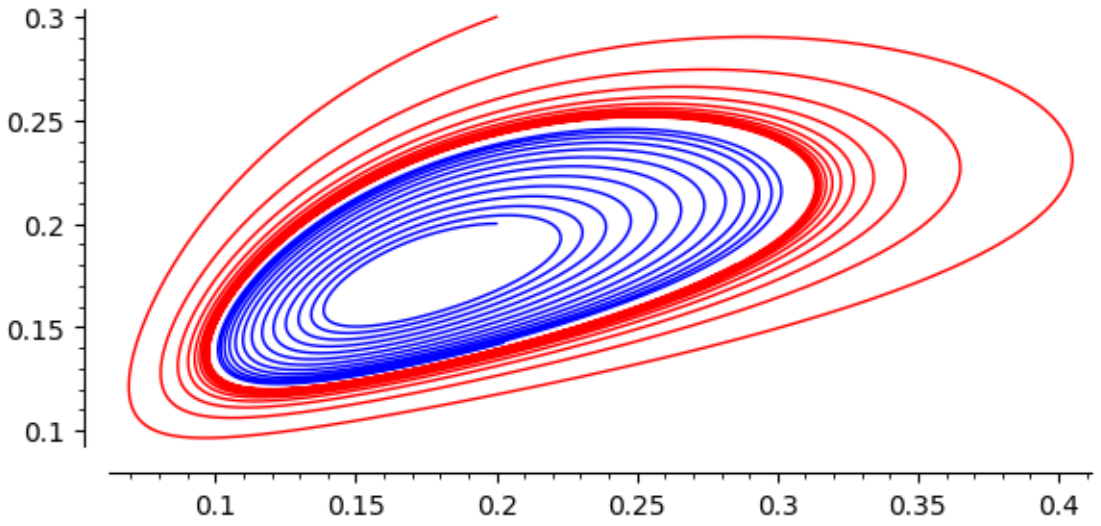
```
(j1+j2).show()#save('sage_chI024_04.pdf')
(j1+j2).show()#save('sage_chI024_04.png')
```

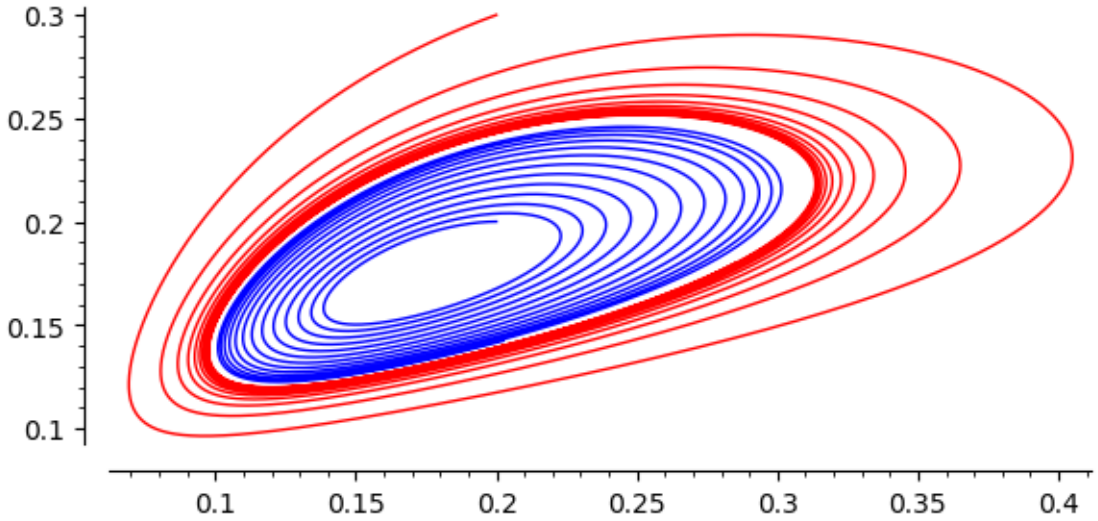


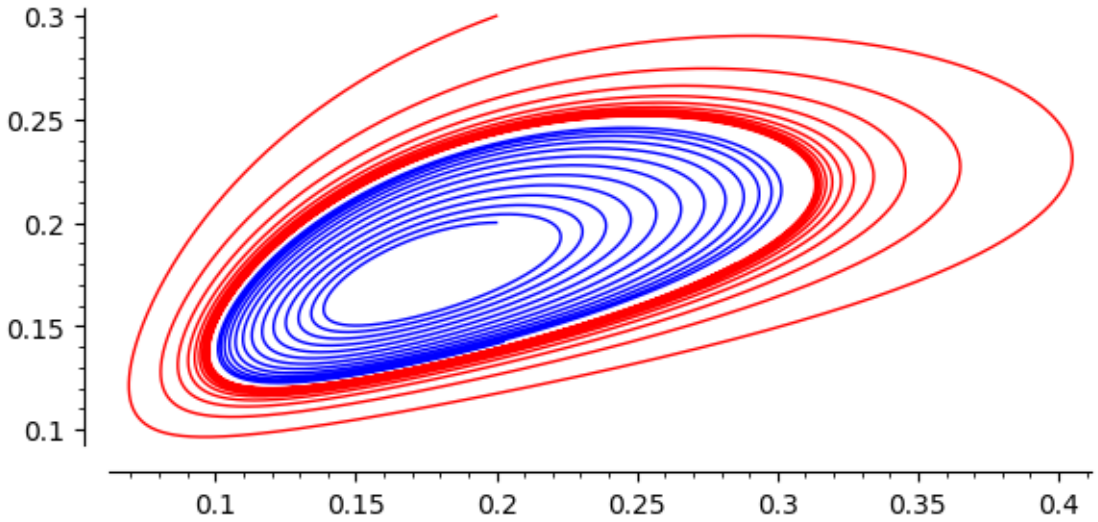




[5]:
```
T = srange(0,200,0.01)
sl1=desolve_odeint(vector([F,G]), [0.2,0.3], T, [x,y])
sl2=desolve_odeint(vector([F,G]), [0.2,0.2], T, [x,y])
j1=list_plot(sl1.tolist(), plotjoined=True, color="red", figsize=(6, 3))
j2=list_plot(sl2.tolist(), plotjoined=True,  figsize=(6, 3))
show(j1+j2)


(j1+j2).show()#save('sage_chI024_05.pdf')
(j1+j2).show()#save('sage_chI024_05.png')
```







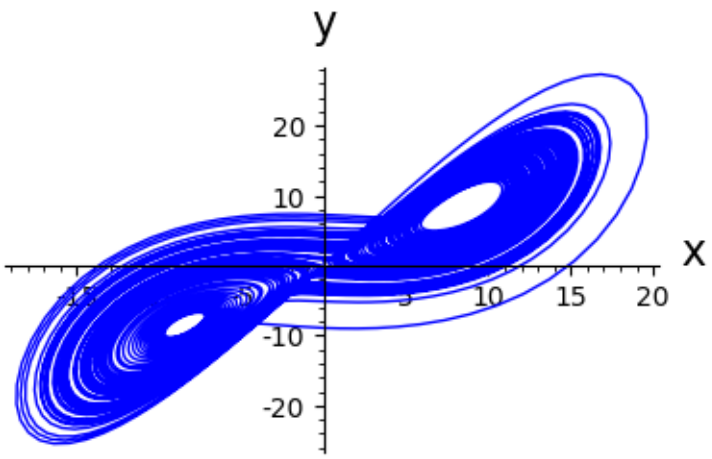
3.1.4 Lorenz attractor

This is an example of the strange attractor. The simplest definition is that it has a fractal structure.

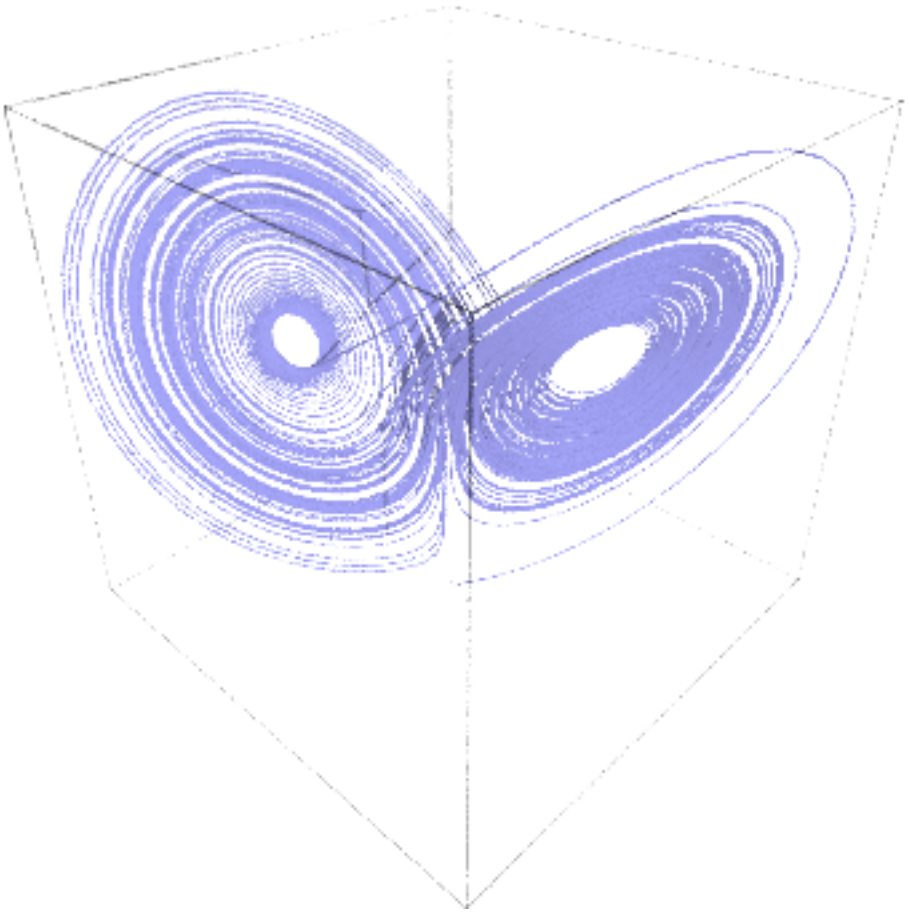
[6]:
```
reset()
var('x y z')

rho=28
sigma=10
```

```
beta=8/3

F1 = sigma*(y-x)
F2 = x*(rho-z) - y
F3 = x*y - beta*z

T = srange(0,100,0.01)
atraktor_lorenza = desolve_odeint(vector([F1,F2,F3]), [0,0.5,1], T, [x,y,z])

p2d = list_plot(zip(atraktor_lorenza[:,0],atraktor_lorenza[:,1]),\
                plotjoined=True, figsize=4, axes_labels=['x','y'])
p3d = list_plot(atraktor_lorenza.tolist(),\
                plotjoined=True, viewer='tachyon', figsize=4)

print "2D rysunek atraktora Lorenza"
p2d.show()

print "3D rysunek atraktora Lorenza"
p3d.show()


#p2d.save('sage_ch1024_06.pdf')
#p2d.save('sage_ch1024_06.png')
```

```
2D rysunek atraktora Lorenza
```




```
3D rysunek atraktora Lorenza
```



4 Conservative and dissipative systems

In the theory of dynamical systems, two concepts play an important role: conservative systems and dissipative systems. Again, for clarity, consider the example of a 3-dimensional phase space system:

$$\dot{x} = F_1(x, y, z), \quad x(0) = x_0$$

$$\dot{y} = F_2(x, y, z), \quad y(0) = y_0$$

$$\dot{z} = F_3(x, y, z), \quad z(0) = z_0$$

Let $D(0)$ is a set in the phase space of the system and let $V(0)$ be a volume of this set. It contains all possible initial conditions

$$\{x_0, y_0, z_0\} \in D(0)$$

Under evolution according to the above system of differential equations, every point (x_0, y_0, z_0) in the phase space from this set will evolve after t to the point $(x(t), y(t), z(t))$. The set of these points at the moment t forms the set $D(t)$ of the volume $V(t)$. The question is:

$$\text{when} \quad V(t) = V(0)$$

In other words, when the dynamical system preserves the phase-space volume. We will examine this problem. We will introduce new notation:

$$x_t = x(t), \qquad y_t = y(t), \qquad z(t) = z_t$$

The phase-space volume of initial conditions at time $t = 0$ is

$$V(0) = \int\int\int_{D(0)} dx_0 dy_0 dz_0$$

The phase-space volume at time t is

$$V(t) = \int\int\int_{D(t)} dx_t dy_t dz_t$$

The evolution of the system is nothing but the change of variables $(x_0, y_0, z_0) \to (x_t, y_t, z_t)$. Let's make this change of variables in the second integral:

$$M(t) = \int\int\int_{D(t)} dx_t dy_t dz_t = \int\int\int_{D(0)} \frac{\partial(x_t, y_t, z_t)}{\partial(x_0, y_0, z_0)} dx_0 dy_0 dz_0 = \int\int\int_{D(0)} J(t) dx_0 dy_0 dz_0 \tag{1}$$

where J is a Jacobian of the transformation $(x_t, y_t, z_t) \rightarrow (x_0, y_0, z_0)$. If the phase volume does not change over time (it is a constant function), then its derivative

$$\frac{dM(t)}{dt} = \int\int\int_{D(0)} \frac{dJ(t)}{dt} dx_0 dy_0 dz_0 \qquad (2)$$

is zero. If

$$\frac{dJ(t)}{dt} = 0 \qquad \text{then} \qquad \frac{dM(t)}{dt} = 0 \qquad \text{it means that} \qquad M(t) = M(0)$$

So we start calculations

$$\frac{dJ(t)}{dt} = \frac{d}{dt}\frac{\partial(x_t, y_t, z_t)}{\partial(x_0, y_0, z_0)} = \frac{d}{dt}\begin{bmatrix} \frac{\partial x_t}{\partial x_0} & \frac{\partial x_t}{\partial y_0} & \frac{\partial x_t}{\partial z_0} \\ \frac{\partial y_t}{\partial x_0} & \frac{\partial y_t}{\partial y_0} & \frac{\partial y_t}{\partial z_0} \\ \frac{\partial z_t}{\partial x_0} & \frac{\partial z_t}{\partial y_0} & \frac{\partial z_t}{\partial z_0} \end{bmatrix}$$

The above determinant should be calculated and remember that solutions of differential equations

$$x_t = x_t(x_0, y_0, z_0), \qquad y_t = y_t(x_0, y_0, z_0), \qquad z_t = z_t(x_0, y_0, z_0)$$

depend on the initial conditions $\{x_0, y_0, z_0\}$. After expanding the determinant, the following types of expressions appear

$$\frac{d}{dt}\frac{\partial x_t}{\partial z_0} = \frac{\partial}{\partial z_0}\frac{dx_t}{dt} = \frac{\partial}{\partial z_0}\dot{x}_t = \frac{\partial}{\partial z_0}F_1(x_t, y_t, z_t) = \frac{\partial F_1}{\partial x_t}\frac{\partial x_t}{\partial z_0} + \frac{\partial F_1}{\partial y_t}\frac{\partial y_t}{\partial z_0} + \frac{\partial F_1}{\partial y_t}\frac{\partial y_t}{\partial z_0} \qquad (1)$$

As you can see, in this simple case, we have to carry out cumbersome calculations. It is much better to use a symbolic calculations using **SageMath**.

To carry out the proof, it is best to circumvent the limitations of operations on expressions with derivatives in Sage. The determinant's derivative is automatically made, then the substitution is manually executed:

$$\frac{\partial}{\partial z_0}\dot{x}_t = \frac{\partial F_1}{\partial x_t}\frac{\partial x_t}{\partial z_0} + \frac{\partial F_1}{\partial y_t}\frac{\partial y_t}{\partial z_0} + \frac{\partial F_1}{\partial y_t}\frac{\partial y_t}{\partial z_0}$$

- first - let us define some helper variables

[1]:
```
var('x y z t')
xy_wsp = [('x','x'),('y','y')]+[('z','z')]
N = len(xy_wsp)
J  = matrix(SR,N)


for i,(v,lv) in enumerate(xy_wsp):
```

```
    for j,(u,lu) in enumerate(xy_wsp):
        J[i,j] = var("d%sd%s"%(v,u),latex_name=r'\displaystyle\frac{\partial
↪%s_t}{\partial %s_0}'%(lv,lu))
        var("dF%sd%s"%(v,u),latex_name=r'\displaystyle\frac{\partial
↪F_%s}{\partial %s_t}'%(lv,lu))

to_fun = dict()
for v in J.list():
    vars()[str(v).capitalize()] = function(str(v).capitalize())(t)
    var("%sd"%str(v))
    to_fun[v]=vars()[str(v).capitalize()]
    to_fun[vars()[str(v)+"d"]]=vars()[str(v).capitalize()].diff()
to_var = dict((v,k) for k,v in to_fun.items())

to_rhs = dict()
for i,(v,lv) in enumerate(xy_wsp):
    for j,(u,lu) in enumerate(xy_wsp):
        to_rhs[vars()["d%sd%sd"%(v,u)]] =
↪sum([vars()["dF%sd%s"%(v,w)]*vars()["d%sd%s"%(w,u)] for w,wl in xy_wsp])
```

The dictionary `to_rhs` implements in fact equation for $\frac{d}{dt}\frac{\partial x_t}{\partial z_0}$ 1:

[2]:
```
to_rhs[dzdxd]
```

[2]:
```
dFzdx*dxdx + dFzdy*dydx + dFzdz*dzdx
```

Now we have Jacobian J, expressed in variables `dxdy...` etc.:

[3]:
```
J
```

[3]:
```
[dxdx dxdy dxdz]
[dydx dydy dydz]
[dzdx dzdy dzdz]
```

All partial derivatives are independent variables, if we want to compute their time derivatives we need to use `to_fun` substitution dictionaty:

[4]:
```
J.subs(to_fun)
```

[4]:
```
[Dxdx(t) Dxdy(t) Dxdz(t)]
[Dydx(t) Dydy(t) Dydz(t)]
[Dzdx(t) Dzdy(t) Dzdz(t)]
```

Hence, we can used above substitution, compute the time derivative and go back to independet varaibles in a following way:

[5]:
```
print J.subs(to_fun).det().diff(t).subs(to_var)
```

```
-(dydzd*dzdy + dydz*dzdyd - dydyd*dzdz - dydy*dzdzd)*dxdx - (dydz*dzdy -
dydy*dzdz)*dxdxd + (dxdzd*dzdy + dxdz*dzdyd - dxdyd*dzdz - dxdy*dzdzd)*dydx +
(dxdz*dzdy - dxdy*dzdz)*dydxd - (dxdzd*dydy + dxdz*dydyd - dxdyd*dydz -
dxdy*dydzd)*dzdx - (dxdz*dydy - dxdy*dydz)*dzdxd
```

We use in this moment a substitution `to_rhs`, which make use of right hand sides of our ODE:

[6]:
```
print J.subs(to_fun).det().diff(t).subs(to_var).subs(to_rhs)
```

```
-(dFzdx*dxdx + dFzdy*dydx + dFzdz*dzdx)*(dxdz*dydy - dxdy*dydz) + (dFydx*dxdx +
dFydy*dydx + dFydz*dzdx)*(dxdz*dzdy - dxdy*dzdz) - (dFxdx*dxdx + dFxdy*dydx +
dFxdz*dzdx)*(dydz*dzdy - dydy*dzdz) + ((dFzdx*dxdz + dFzdy*dydz +
dFzdz*dzdz)*dydy - (dFzdx*dxdy + dFzdy*dydy + dFzdz*dzdy)*dydz - (dFydx*dxdz +
dFydy*dydz + dFydz*dzdz)*dzdy + (dFydx*dxdy + dFydy*dydy +
dFydz*dzdy)*dzdz)*dxdx - ((dFzdx*dxdz + dFzdy*dydz + dFzdz*dzdz)*dxdy -
(dFzdx*dxdy + dFzdy*dydy + dFzdz*dzdy)*dxdz - (dFxdx*dxdz + dFxdy*dydz +
dFxdz*dzdz)*dzdy + (dFxdx*dxdy + dFxdy*dydy + dFxdz*dzdy)*dzdz)*dydx +
((dFydx*dxdz + dFydy*dydz + dFydz*dzdz)*dxdy - (dFydx*dxdy + dFydy*dydy +
dFydz*dzdy)*dxdz - (dFxdx*dxdz + dFxdy*dydz + dFxdz*dzdz)*dydy + (dFxdx*dxdy +
dFxdy*dydy + dFxdz*dzdy)*dydz)*dzdx
```

SageMath can simplify this formula, which needs to be divided be original Jacobian:

[7]:
```
final = J.subs(to_fun).det().diff(t).subs(to_var).subs(to_rhs)/J.det()
final.simplify_full()
```

[7]:
```
dFxdx + dFydy + dFzdz
```

Finally, by algebraic manipulations we obtained the relation

$$\frac{dJ(t)}{dt} = J(t)\left[\frac{\partial F_1}{\partial x_t} + \frac{\partial F_2}{\partial y_t} + \frac{\partial F_3}{\partial z_t}\right] = J(t)\ \mathrm{div}\vec{F}$$

The expression in parentheses is called a divergence of the vector field $\vec{F} = [F_1, F_2, F_3]$. We insert it to Eq. (2) with the result

$$\frac{dV(t)}{dt} = \int\int\int_{D(0)} \frac{dJ(t)}{dt} dx_0 dy_0 dz_0 = \int\int\int_{D(0)} J(t)\ \mathrm{div}\vec{F} dx_0 dy_0 dz_0 = \int\int\int_{D(t)} \mathrm{div}\vec{F} dx_t dy_t dz_t \qquad (3)$$

where the operation of inverse transformation has been performed similar like in Eq.(1). Note that when

$$\mathrm{div}\vec{F} = 0$$

then

$$\frac{dV(t)}{dt} = 0 \qquad \text{and it means that} \qquad V(t) = const. = V(0)$$

This result can be generalized to an arbitrary number of differential equations:

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x}), \qquad \vec{x} = [x_1, x_2, x_3, ..., x_n], \qquad \vec{F} = [F_1, F_2, F_3, ..., F_n]$$

and we can formulate

4.1 THEOREM

If the divergence of the vector field generated by the set of differential equation is zero,

$$\text{div}\vec{F} = \sum_i \frac{\partial F_i}{\partial x_i} = 0$$

then the phase-space volume is preserved, $V(t) = V(0)$. Such systems are called conservative. If the volume $V(t)$ decreases as a function of time then the system is called DISSIPATIVE:

$$\frac{dV(t)}{dt} < 0$$

If

$$\text{div}\vec{F} = C_0 = const. \qquad (4)$$

then from Eq. (3) it follows that

$$\frac{dV(t)}{dt} = C_0 V(t) \qquad (5)$$

and then we can know if the system is dissipative or not.

Example 1: Damped harmonic oscillator

$$\dot{x} = y = F_1(x, y), \qquad x(0) = x_0$$

$$\dot{y} = -\gamma y - \omega^2 x = F_2(x, y), \qquad y(0) = y_0$$

We can calculate the divergence:

$$\text{div}\vec{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} = -\gamma$$

Eq. (4) takes the form

$$\frac{dV(t)}{dt} = -\gamma V(t), \qquad \text{its solution is} \qquad V(t) = V(0)e^{-\gamma t}$$

So, the volume $V(t)$ is a decreasing function of time and the system is dissipative.

Example 2: Lorenz system

$$\dot{x} = \sigma(y - x) = F_1(x, y, z), \qquad x(0) = x_0$$

$$\dot{y} = x(\rho - z) - y = F_2(x, y, z), \qquad y(0) = y_0$$

$$\dot{z} = xy - \beta z = F_3(x, y, z),, \qquad z(0) = z_0$$

All parameters of the model are positive: $\sigma, \rho, \beta > 0$. Let us calculate the divergence of the vector field $\vec{F} = [F_1, F_2, F_3]$:

$$\text{div}\vec{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z} = -\sigma - 1 - \beta < 0$$

Again, as in the previous example the divergence is negative and in consequence the phase volume is a decreasing function of time. Hence, the system is dissipative.

5 Chaos in dynamical systems

Consider a set of autonomous differential equations of the form

$$\frac{d\vec{x}}{dt} = \vec{F}(\vec{x}), \qquad \vec{x}(0) = \vec{x}_0, \qquad \vec{x} = [x_1, x_2, ..., x_n]$$

From mathematics (lets us recall the Picard–Lindelöf theorem, Picard's existence theorem, Cauchy–Lipschitz theorem), existence and uniqueness theorem gives a set of conditions under which the above set of differential equations has a unique solution. It means that in the phase space of the system, there is only one trajectory

$$\vec{x}(t) = \vec{x}(t; \vec{x}_0)$$

starting from the initial point $\vec{x}_0$ and reaching the point $\vec{x}(t)$ at time t. Trajectories can not intersect each other. If the set of differential equations describes evolution of some system, it means that we can predict the future from the past. It is what we call determinism or predictability. In the history of science, physics has been succesful because from laws of physics people have been able to predict many interesting and unusual phenomena which have been experimentally confirmed. An example of the deterministic theory is classical mechanics. Theorems about the uniqueness of solutions for Newton equations gave hope for total determinism and predictability of the motion of individual particles. This hope from a practical point of view turned out to be a dream. In the 1950s it was shown that from a practical point of view the determinism of Newton's mechanics is illusory and the established belief in the predictability of simple mechanical systems has collapsed. Numerous examples appeared, and later a mathematical theory, showing the impossibility of predicting the temporal evolution of simple mechanical systems. We emphasize that this is about practical aspects of predictability. From a mathematical point of view, predictability is still valid. A clear example of unpredictability in practice is the weather forecast, which proves everyday day. Below we will present issues that will show us what unpredictability means in deterministic theory. We will show why the evolution determined by the determinism of Newton's equations is unpredictable. This deterministic unpredictability has its name: deterministic chaos.

[ ]:

5.1 MODEL OF CHAOS: A BISTABLE SYSTEM (Duffing oscillator)

In order to understand well the essence of the chaotic behavior of dynamicAL systems, we will CONSIDER a simple example from classical Newton's mechanics. We consider a one-dimensional motion of a particle along the OX axis described by the Newton equation:

$$m\ddot{x} = F(x, \dot{x}, t) = ax - bx^3 - \gamma\dot{x} + A\cos(\Omega t), \qquad x(0) = x_0, \quad \dot{x}(0) = v(0) = v_0$$

This model seems to be very simple:

- it is a particle movement under influence of the force $F(x) = ax - bx^3 (a, b > 0)$

- there is dissipation (friction) described by the Stokes force $F(v) = -\gamma v = -\gamma\dot{x}$. It is the same as in the damped oscillator

- the periodic force $F(t) = A\cos(\Omega t)$ drives the particle

Note that the average value of the froce $F(t)$ over one period $T = 1/\Omega$ is zero. The froce $F(x) = x - x^3$ is potential. The corresponding potential of $V(x)$ has the form:

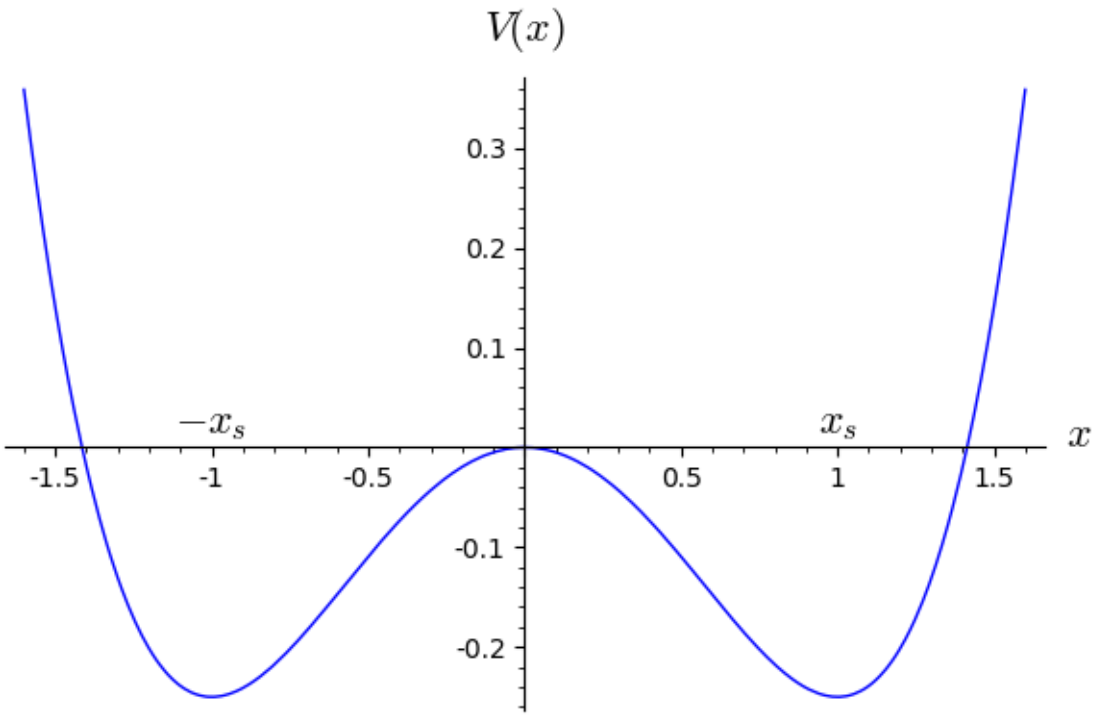
$$V(x) = -\int F(x)dx = \frac{1}{4}bx^4 - \frac{1}{2}ax^2$$

This potential is called a bistable potential and is one of the most important model of potentials in physics, starting from the Newton theory, throughout the theory of phase transitions, the theory of activation of chemical reactions, and ending on the theory of elementary particles (the Higgs mechanism). Below we show a graph of this potential (to put it precisely: it is potential energy).

The above Newton equation has several realizations.

1. The first embodiment: A metal ball suspended on a weightless rod in the field of two magnets (which model the bistable potential). A periodic electric field $A\cos(\omega t)$ runs on the ball in the horizontal direction. This system is shown in the picture.
2. The second embodiment: Electronic circuit, which is described in detail in the work: B. K. Jones and G. Trefan, Am. J. Phys. 69 (2001) p. 464. "The Duffing oscillator: A precise electronic analog chaos demonstrator for the undergraduate laboratory"

[1]:
```
# Przeskalowany potencja bistabilny: a=b=1
p = plot(0.25*x^4 - 0.5*x^2, (x,-1.6,1.6), figsize=(6,4),␣
 ↪axes_labels=[r'$x$',r'$V(x)$'])
p += text("$-x_s$",(-1,0.025),fontsize=16, color='black')
p += text("$x_s$",(1,0.025),fontsize=16, color='black')
p.show()
```



5.2 Scaling

The above model contains 6 parameters: $\{m, a, b, \gamma, A, \Omega\}$. Some parameters can be eliminated by rescaling the equation to a dimensionless form. There are several options. Usually we start with time and position scaling. We make the followin coice: The new dimensionless time τ has the form:

$$s = \frac{t}{\tau_0}, \qquad \tau_0^2 = \frac{m}{a}$$

We define the new dimensionless position as

$$X = \frac{x}{L}, \qquad L^2 = \frac{a}{b}$$

Then, the dimensionless form of the traffic equation is as follows:

$$\ddot{X} = X - X^3 - \gamma_0 \dot{X} + A_0 \cos(\omega_0 s), \qquad X(0) = X_0, \quad \dot{X}(0) = \dot{X}_0$$

There are currently 3 rescaled parameters:

$$\gamma_0 = \frac{\gamma}{m}\tau_0 = \frac{\tau_0}{\tau_L}, \quad \tau_L = \frac{m}{\gamma}, \qquad A_0 = \frac{A}{aL}, \qquad \omega_0 = \tau_0 \Omega$$

Note that τ_L has a time unit and is a secod characteristic time, which is sometimes called the Langevin relaxation time. To interpret it, you need to study the equation of a free particle:

$$m\ddot{x} = -\gamma\dot{x}, \qquad \text{lub} \qquad \dot{v} = -\frac{\gamma}{m}v$$

The solution of this linear equation is the exponential function:

$$v(t) = v(0)\exp[-\gamma t/m] = v(0)\exp[-t/(m/\gamma)]$$

Now it can be seen that the m/γ is the characteristic time of relaxation of the velocity of the free particle in the environment. It is the rate at which the particle slows down due to friction. In the following, we will use only the scaled equation and for simplicity we omi subscripts in the parameters. It means thatwe will analyze the equation in the form

$$\ddot{x} = x - x^3 - \gamma\dot{x} + A\cos(\omega_0 t), \qquad x(0) = x_0, \quad \dot{x}(0) = \dot{y}_0 = v_0$$

where the potential has the form

$$V(x) = -\int F(x)dx = \frac{1}{4}x^4 - \frac{1}{2}x^2$$

The scaled equation looks like the starting dimensional equation with the parameters: $m = 1, a = 1, b = 1$. Note that after the scaling procedure the Newton equation contains only 3 dimensionless parameters: $\{\gamma, A, \omega_0\}$.

[ ]:

5.2.1 STEP 1: THE CONSERVATIVE SYSTEM

In the first step, we consider the simplest case (remember the scaled form, in which the mass is $m = 1$):

$$\ddot{x} = x - x^3 = -V'(x), \qquad x(0) = x_0, \quad \dot{x}(0) = v(0) = v_0$$

It is equivalent to a set of 2 autonomous, first order differential equations:

$$\dot{x} = v, \qquad x(0) = x_0$$

$$\dot{v} = x - x^3, \qquad v(0) = v_0$$

This means that the phase space is 2-dimensional. Such a case has already been considered: it is a conservative system with one degree of freedom. There is one constant of motion (one integral of the movement), namely the total energy of the system:

$$\frac{1}{2}\dot{x}^2(t) + V(x(t)) = const. = E = E_k + E_p = \frac{1}{2}\dot{x}^2(0) + V(x(0)) = \frac{1}{2}v_0^2 + V(x_0)$$

which consists of the kinetic energy E_k and the potential energy E_p. The constant E is determined by the initial conditions $x(0) = x_0$ and $v(0) = v_0$. Because the total energy of the system is conserved, the motion is periodic. There are no attractors and there are no asymptotically stable stationary states. Phase curves are closed which means that the particle moves periodically over time. Depending on the initial conditions, the amplitude of oscillations is greater or smaller, because the initial conditions determine the value of the constant E. If the two initial conditions (x_{01}, v_{01}) and (x_{02}, v_{02}) are slightly different:

$$|[x_{01}^2 + v_{01}^2] - [x_{02}^2 + v_{02}^2]| < \delta << 0$$

i.e., it is small then the phase curves are slightly different and the particle movement for these two initial conditions is slightly different. We say then that the system is insensitive to changing the initial conditions. As can be seen from the above formula, two different initial conditions mean that the system has two different energies E. This in turn means that the frequency of periodic motion will also be different. The frequency difference causes the particles to slowly move away from each other.
Below we present the potential and phase curves for this case.

[2]:
```
#parametry dla wizualizacji
var('x v')
x0, v0 = 1.5, 0.2
V = x^4/4 - x^2/2
E = V(x=x0) + v0^2

#prawo zachowania energii
PZE = v^2 + V == E

#wychylenia ekstremalne
print "ekstremalne wychylenia dla (x0,v0) = (%.2f,%.2f)"%(x0,v0)
rozw = solve(PZE(v=0), x); show(rozw)
xmin = min([i.rhs() for i in rozw if imag(i.rhs()) == 0])
xmax = max([i.rhs() for i in rozw if imag(i.rhs()) == 0])

#i jego rozwizanie
print "ekstremalne prdkoci dla (x0,v0) = (%.2f,%.2f)"%(x0,v0)
rozw = solve(PZE, v); show(rozw)
v1=rozw[0].rhs()
v2=rozw[1].rhs()
vmax = abs(v1(x=0))

#krzywe fazowe
start_point = (x0,V(x=x0))
p0 = point(start_point,size=30) +␣
 ↪text(r"$x_0$",start_point,vertical_alignment='bottom',horizontal_alignment='left')
p1 = plot(V,(x,xmin,xmax))
p21 = plot(v1, (x,xmin,xmax), color='red')
```

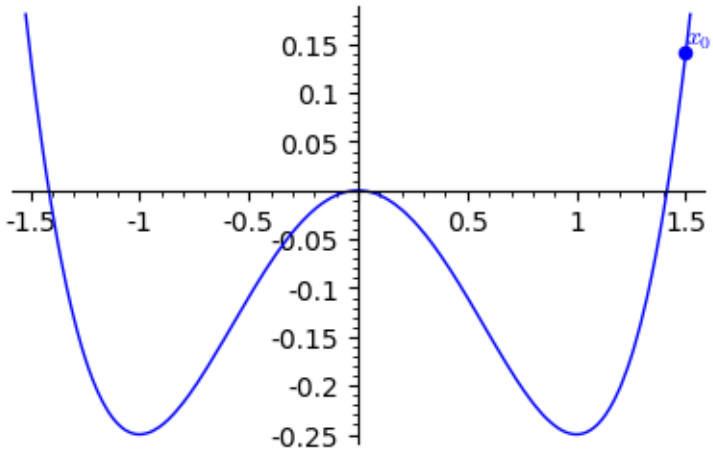
```
p22 = plot(v2, (x,xmin,xmax), color='green')
(p0+p1).show(figsize=4)
(p21+p22).show(figsize=4)
```

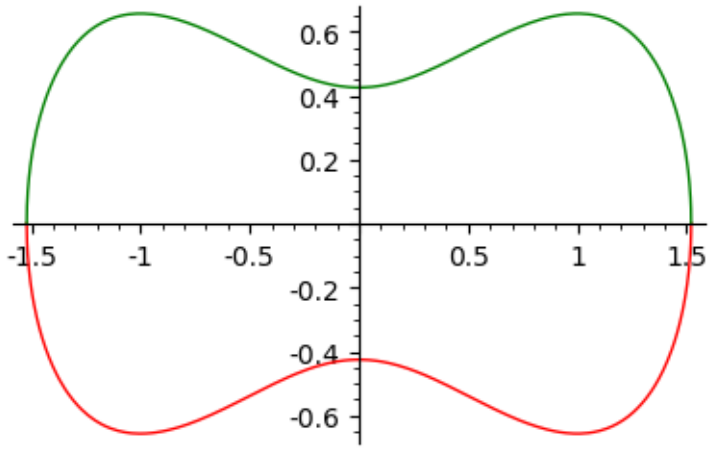
```
ekstremalne wychylenia dla (x0,v0) = (1.50,0.20)
```

```
[x == -1/2*sqrt(1/5*sqrt(689) + 4), x == 1/2*sqrt(1/5*sqrt(689) + 4), x == -1/2*sqrt(-1/5*sq
```

```
ekstremalne prdkoci dla (x0,v0) = (1.50,0.20)
```

```
[v == -1/40*sqrt(-400*x^4 + 800*x^2 + 289), v == 1/40*sqrt(-400*x^4 + 800*x^2 + 289)]
```





5.2.2 STEP 2: THE DISSIPATIVE SYSTEM (EFFECT OF FRICTION)

In the second step, we add friction and consider the equation of motion in the form:

$$\ddot{x} = x - x^3 - \gamma\dot{x}, \qquad x(0) = x_0, \quad \dot{x}(0) = v(0) = v_0$$

It is equivalent to a set of 2 autonomous first order differential equations,

$$\dot{x} = v, \qquad x(0) = x_0$$

$$\dot{v} = x - x^3 - \gamma v, \qquad v(0) = v_0$$

Such a case was also considered: it is a dissipative system with one degree of freedom. The energy of the system is not conserved because of dissipation. The total energy of the system decreases in time. In this system there are 3 stationary states. These states are determined by the equations:

$$v = 0, \qquad x - x^3 - \gamma v = 0, \qquad \text{hence} \qquad \{x_{s0} = 0, v = 0\} \quad \{x_{s1} = 1, v = 0\} \quad \{x_{s1} = -1, v = 0\}$$

The stationary states $ x_{s1} = 1$ and $ x_{s2} = -1$ are stable. The $x_{s0} = 0$ state is unstable. There are 2 attractors $A_1 = x_{s1} = 1$ and $A_2 = x_{s2} = -1$ and 2 regions of attraction $B(A_1)$ and $B(A_2)$. Their sum $B(A_1) \cup B(A_2) = R^2$ is the whole plane. Phase curves always tend to one of the attractors or an unstable stationary state. If the two initial conditions (x_{01}, v_{01}) and (x_{02}, v_{02}) differ slightly:

$$|[x_{01}^2 + v_{01}^2] - [x_{02}^2 + v_{02}^2]| < \delta << 0$$

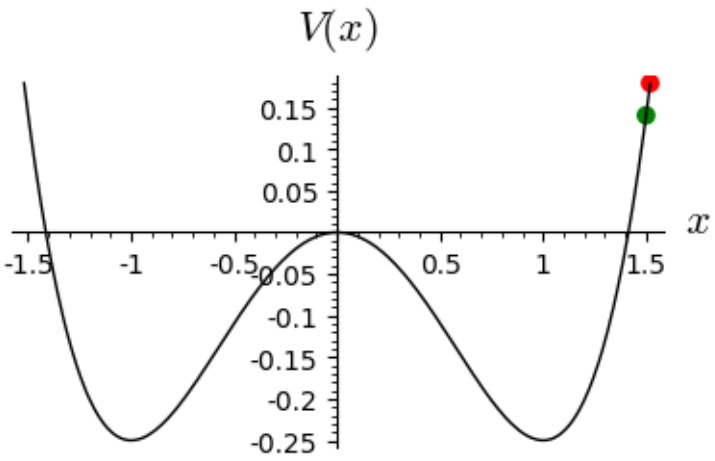
and they are in the same basin of attraction, the phase curves are slightly different and the particle movement for these two initial conditions is slightly different. We say then that the system is insensitive to changing the initial conditions. However, if the two initial conditions $(x_{01}, v_{01}) \in B(A_1)$ and $(x_{02}, v_{02}) \in B(A_2)$ differ slightly, but are in two various basins of attraction $B(A_1)$ and $B(A_2)$, the trajectories will start to differ significantly after a certain time, will be attracted to two different attractors and tend to two different stationary states $ x_{s1} = 1$ and $ x_{s2} = -1$. Nevertheless, in such a situation we say that the system is insensitive to a change in the initial conditions in the sense mentioned above.

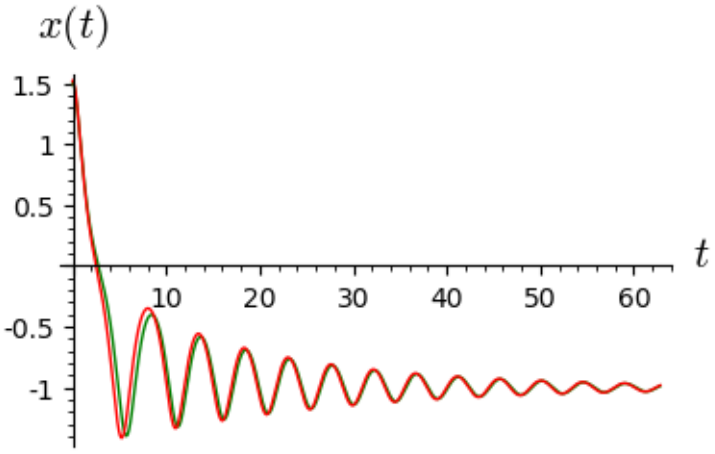
Diagram for basins of attraction for bistable potential:

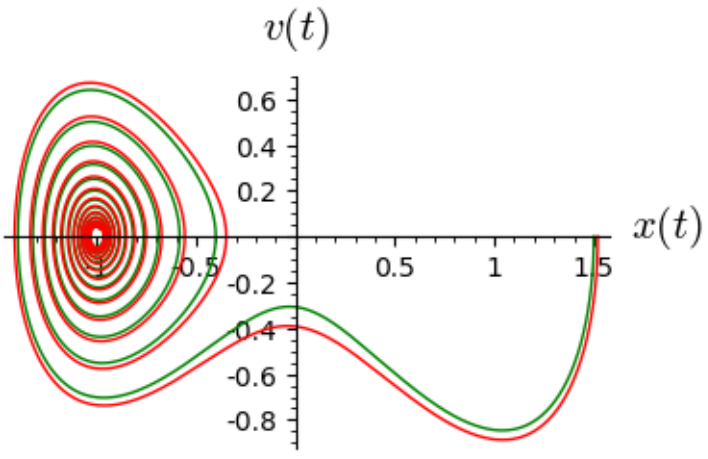
The blue color is the area of initial conditions that are "attracted" to the attractor $(1, 0)$, to the right minimum potential. The red color is the area of initial conditions that are "attracted" to the attractor $(-1, 0)$, to the left minimum potential. Depending on the value of the damping constant γ, this diagram takes a slightly different shape, but the structure of the two-colored stripes remains. The edge of the regions of gravity is a smooth curve whose size is 1. If the initial conditions are exactly on this edge, the particle moves to unstable stationary state $(x = 0, v = 0)$ (maximum potential).

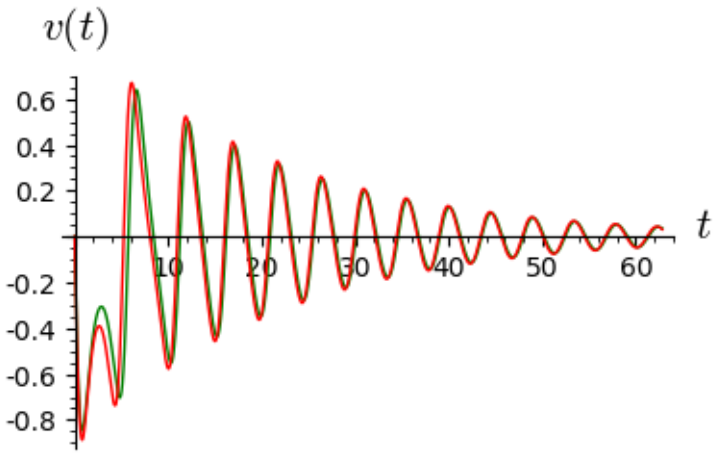
[3]:
```
# wykresy dla przypadku z tumieniem
var('x v')
x01, v01 = 1.50, 0
```

```
x02, v02 = 1.52, 0

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.1

# numeryczne rozwiazanie równa ruchu
T = srange(0,20*pi,0.01)
num1 = desolve_odeint(vector([v,F-g*v]), [x01,v01], T, [x,v])
num2 = desolve_odeint(vector([v,F-g*v]), [x02,v02], T, [x,v])

#krzywe fazowe
lt  = plot(V, (x, -max([abs(x01),abs(x02)]),max([abs(x01),abs(x02)])),␣
 ↪color='black', figsize=4)
lt += point((x01,V(x=x01)), color='green', size=50, axes_labels=['$x$','$V(x)$'])
lt += point((x02,V(x=x02)), color='red', size=50)
lb  = list_plot(num1.tolist(), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'])
lb += list_plot(num2.tolist(), plotjoined=True, color='red', figsize=4)
rt  = list_plot(zip(T,num1[:,0].tolist()), plotjoined=True, color='green',␣
 ↪axes_labels=['$t$','$x(t)$'])
rt += list_plot(zip(T,num2[:,0].tolist()), plotjoined=True, color='red',␣
 ↪figsize=4)
rb  = list_plot(zip(T,num1[:,1].tolist()), plotjoined=True, color='green',␣
 ↪axes_labels=['$t$','$v(t)$'])
rb += list_plot(zip(T,num2[:,1].tolist()), plotjoined=True, color='red',␣
 ↪figsize=4)

html("""
rozwizania z warunkami pocztkowymi
($x_{01},v_{01}$)=(%.2f,%.2f)
($x_{02},v_{02}$)=(%.2f,%.2f)
d do tego samego atraktora:
(x,v)=(-1,0)

"""%(x01,v01,x02,v02))
lt.show(),rt.show(),lb.show(),rb.show()
```


V(x)
0.15
0.1
0.05
x
-1.5
-1
-0.5
0.05
0.5
1
1.5
-0.1
-0.15
-0.2
-0.25


x(t)
1.5
1
0.5
t
10
20
30
40
50
60
-0.5
-1


v(t)
0.6
0.4
0.2
x(t)
-0.5
0.5
1
1.5
-0.2
-0.4
-0.6
-0.8

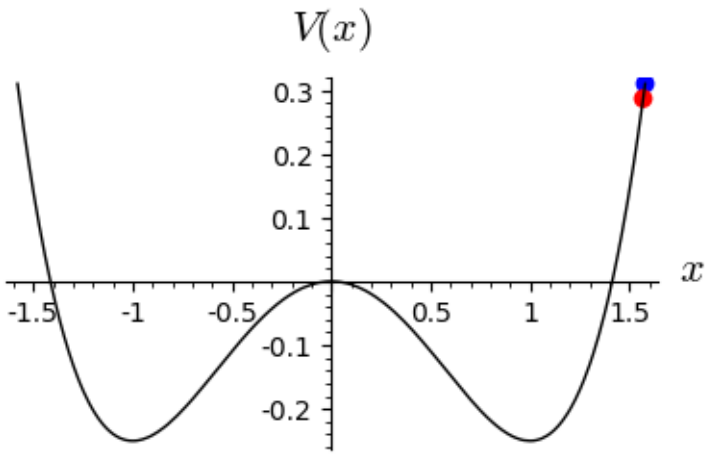


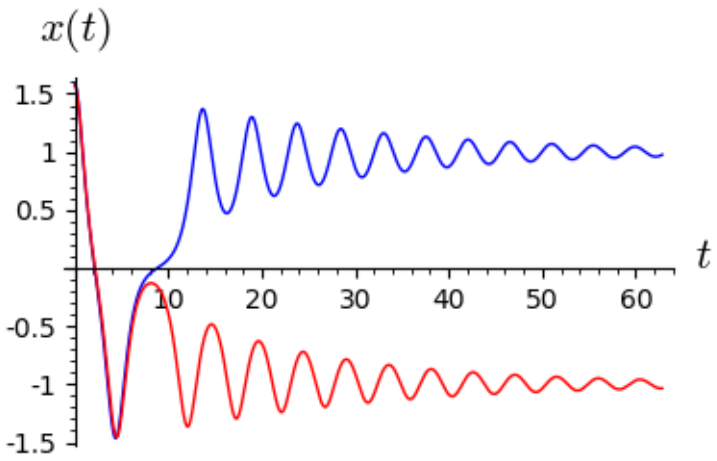
[3]:
```
(None, None, None, None)
```

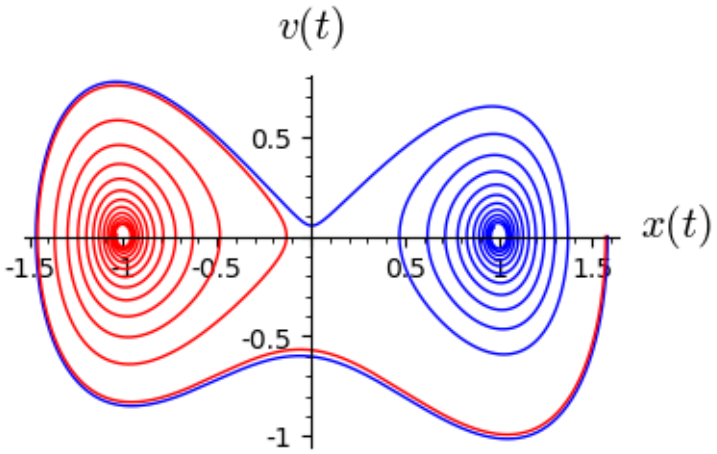
In the above figures, 2 initial conditions lie in the same basin of attraction of the $(-1, 0)$ attractor. This means that 2 initial conditions are located in the red area. The system is not sensitive to changing the initial conditions when they are in the same basin of attraction.

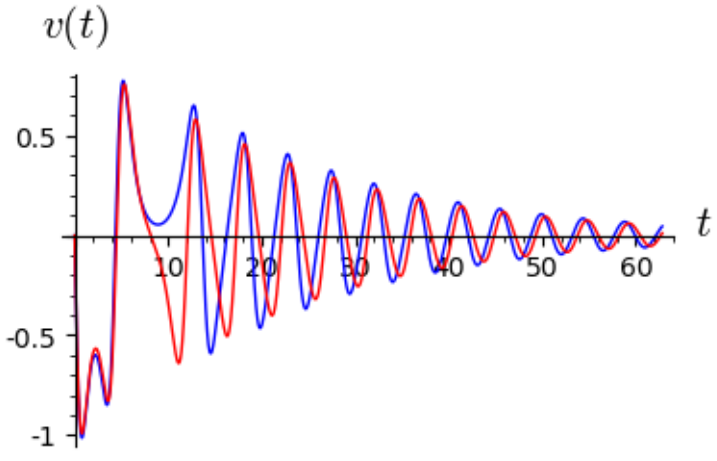
[4]:
```
var('x v')
x01, v01 = 1.58, 0
x02, v02 = 1.57, 0


F = x-x^3
V = -integrate(F,x)

g = 0.1

T = srange(0,20*pi,0.01)
num1 = desolve_odeint(vector([v,F-g*v]), [x01,v01], T, [x,v])
num2 = desolve_odeint(vector([v,F-g*v]), [x02,v02], T, [x,v])

# wykresy funkcji
lt  = plot(V, (x,␣
 ↪-max([abs(x01),abs(x02)]),max([abs(x01),abs(x02)])),color='black',  figsize=4)
lt += point((x01,V(x=x01)), color='blue', size=50, axes_labels=['$x$','$V(x)$'])
lt += point((x02,V(x=x02)), color='red', size=50)
lb  = list_plot(num1.tolist(), plotjoined=True, color='blue',␣
 ↪axes_labels=['$x(t)$','$v(t)$'])
lb += list_plot(num2.tolist(), plotjoined=True, color='red', figsize=4)
```

```
rt  = list_plot(zip(T,num1[:,0].tolist()), plotjoined=True, color='blue',␣
 ↪axes_labels=['$t$','$x(t)$'])
rt += list_plot(zip(T,num2[:,0].tolist()), plotjoined=True, color='red',␣
 ↪figsize=4)
rb  = list_plot(zip(T,num1[:,1].tolist()), plotjoined=True, color='blue',␣
 ↪axes_labels=['$t$','$v(t)$'])
rb += list_plot(zip(T,num2[:,1].tolist()), plotjoined=True, color='red',␣
 ↪figsize=4)

html("""
rozwizania z warunkami pocztkowymi
($x_{01},v_{01}$)=(%.2f,%.2f)
($x_{02},v_{02}$)=(%.2f,%.2f)
d do rónych atraktorów:
(x,v)=(1,0)
(x,v)=(-1,0)

"""%(x01,v01,x02,v02))


lt.show(),rt.show(),lb.show(),rb.show()
```




$x(t)$
1.5
1
0.5
-0.5
-1
-1.5
10
20
30
40
50
60
t


$v(t)$
0.5
-0.5
-1
-1.5
-1
-0.5
0.5
1
1.5
$x(t)$


$v(t)$
0.5
-0.5
-1
10
20
30
40
50
60
t

[4]: `(None, None, None, None)`

In the above figures, 2 initial conditions are located in two different basins of attraction. This means that the first initial condition lies in the blue area on the attraction diagram, while the second initial condition lies in the red area of the basin of attraction. These two initial conditions are located close to the border of two basins of attraction. Therefore, the system is sensitive to changing the initial conditions, provided that they lie in two different basins of attraction. But this is not yet the criterion of chaotic property of the system.

5.2.3 STEP 3: THE SYSTEM DRIVEN BY TIME-PERIODIC FORCE

In the third step, we add the time-periodic force and consider the equation of motion in the initial full form:

$$\ddot{x} = x - x^3 - \gamma\dot{x} + A\cos(\omega_0 t), \qquad x(0) = x_0, \quad \dot{x}(0) = v(0) = v_0$$

It is equivalent to the system of 3 differential equations, autonomous, first order:

$$\dot{x} = v, \qquad x(0) = x_0$$

$$\dot{v} = x - x^3 - \gamma v + A\cos z, \qquad v(0) = v_0$$

$$z = \omega_0, \qquad z(0) = 0$$

This means that the phase space is 3-dimensional. Mathematicians prefer to rewrite the above system of equations for the "traditional" 3 variables (x, y, z) in the form:

$$\dot{x} = y, \qquad x(0) = x_0$$

$$\dot{y} = x - x^3 - \gamma y + A\cos z, \qquad y(0) = y_0$$

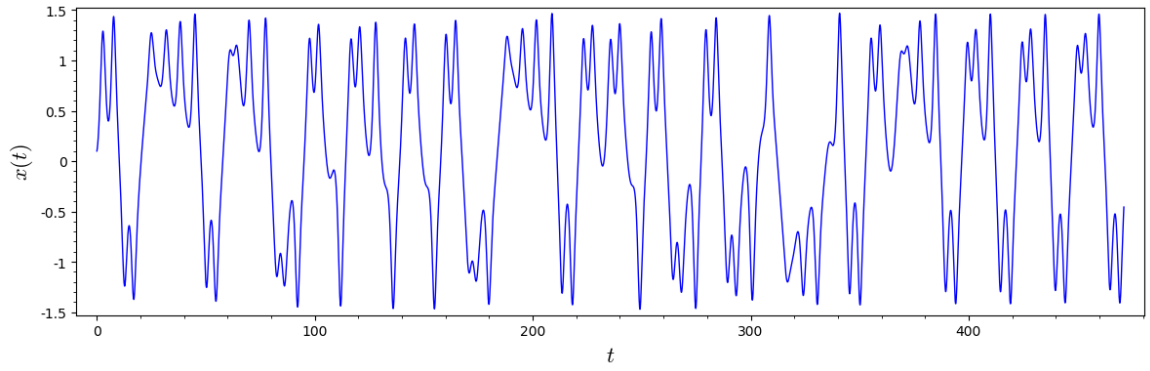
$$z = \omega_0, \qquad z(0) = 0$$

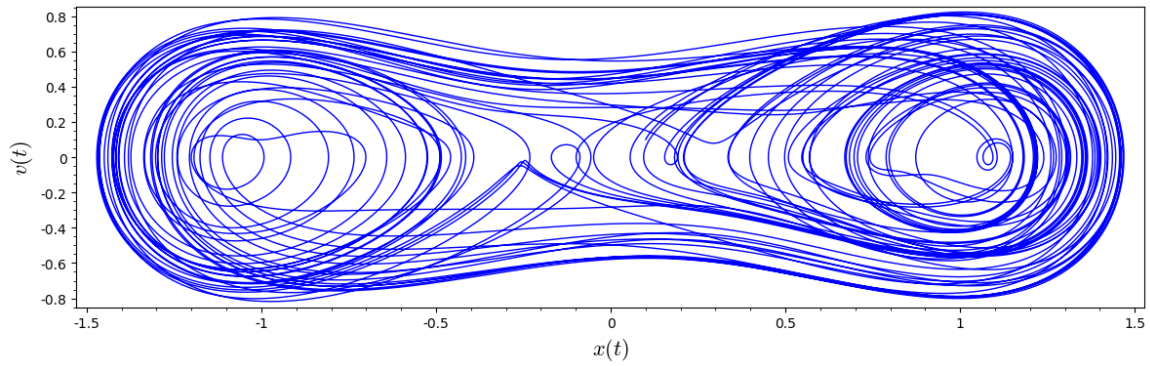
that is, the velocity v of the particle is now denoted as $v = y$.

It turns out that this system exhibits radically different properties from the previous two cases. From the physics point of view, the particle moves in a bistable potential. Because the potential tends to infinity when the position tends to infinity, the movement of the particle is bounded; the particle is trapped in the potential and can not escape to infinity. The friction force pushes the particle to one of ("old") stationary states x_{s1} or x_{s2}. In turn, the external periodic force pumps energy into the system

and counteracts the friction force. The particle no longer tends to a time-independent stationary state, it will not stop for long times but it will non-stop move and will never rest. The inertial effects associated with mass of the particls, which are reflected in the word $\dot{y}$, i.e. acceleration of the particle, become significant. As a consequence, the system does not have a stationary state in the form of a point in the phase space as it was in case 2. All these factors become essential for understanding the complex properties of particle evolution.

[5]:
```
var('x y z')
T = srange(0,150*pi,0.01)
sol=desolve_odeint( vector([y,x-x^3-0.26*y+0.3*cos(z), 1]), [0.1,0.1,0],T,[x,y,z])
t = line(zip(T,sol[:,0]), figsize=(12,4), axes_labels=["$t$","$x(t)$"], frame=1,␣
 ↪axes=0)
b = line(zip(sol[:,0],sol[:,1]), figsize=(12,4), axes_labels=["$x(t)$","$v(t)$"],␣
 ↪frame=1, axes=0)
t.show()

b.show()
```





The arrangement behavior presented above seems to be irregular, which is confirmed by the graph presenting the projection of the phase curve on the (X, Y) plane, location - velocity. The temporal

evolution of $x(t)$ looks like a random time series. But is this really a regime of chaotic behavior in the system? There are several measures that allow you to answer this question.

[ ]:

5.2.4 PERIOD-1 SOLUTIONS

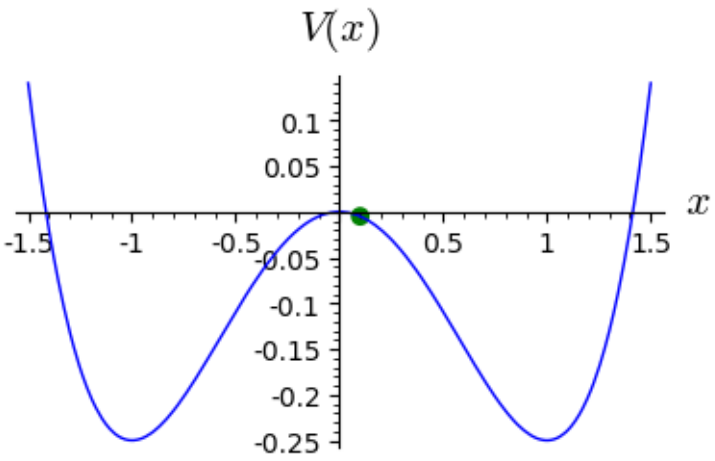
There are 3 dimensionless parameters in the model: γ - the friction coefficient, A - the external time-peiodic force of amplitude and frequency ω_0. Below we will show some characteristic trajectories of the system. We will start with a simple periodic evolution, periodic movement with the so-called period-1 solution. Let's assume the following parameter values:

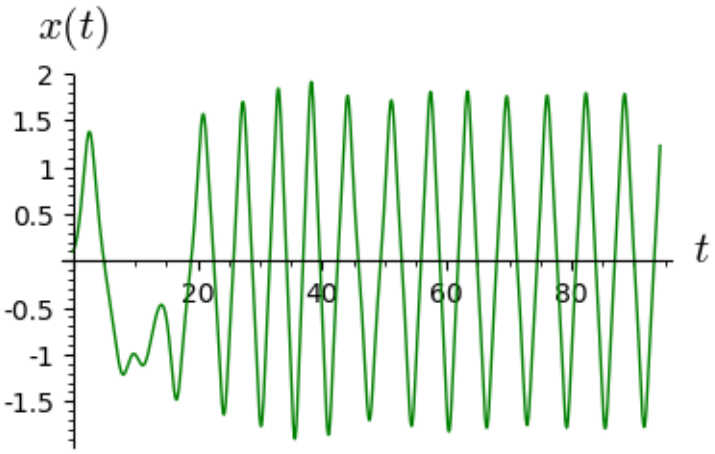
$$\gamma = 0.15, \qquad A = 0.3, \qquad \omega_0 = 1$$

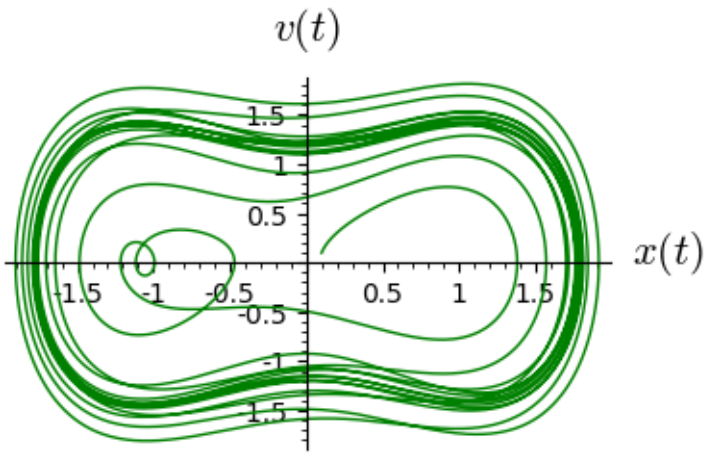
In this case, we observe regular trajectory. If we slightly disturb the initial conditions, the new trajectory is also regular (you have to be careful when we say "we will slightly disturb").

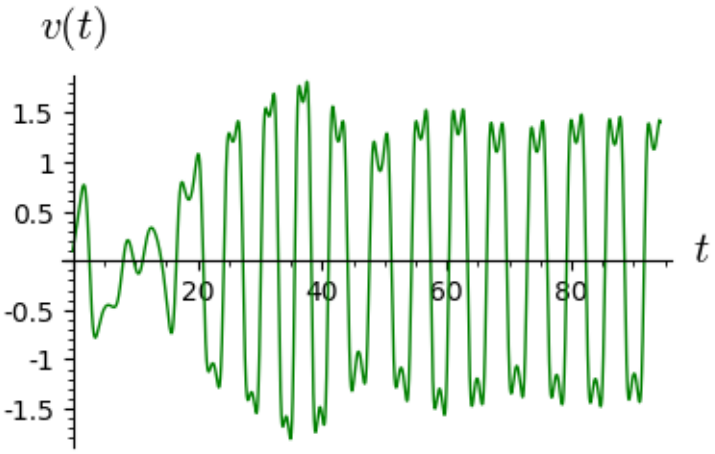
[6]:
```
# wykresy dla przypadku z tumieniem
var('x y z')
x0, y0, z0 = 0.1,0.1,0
kolor = 'green'

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.1
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,30*pi,0.01)
num = desolve_odeint(vector([dx,dy,dz]), [x0,y0,z0], T, [x,y,z])

# wykresy funkcji
xmin = 1.5
lt  = plot(V, (x,-xmin,xmin), figsize=4)
lt += point((x0,V(x=x0)), color=kolor, size=50, axes_labels=['$x$','$V(x)$'])
lb  = list_plot(zip(num[:,0],num[:,1]), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
rt  = list_plot(zip(T,num[:,0].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$x(t)$'], figsize=4)
rb  = list_plot(zip(T,num[:,1].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$v(t)$'], figsize=4)
```

```
html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z warunkami pocztkowymi
$(x_0,y_0,z_0) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x0,y0,z0))
lt.show(),rt.show(),lb.show(),rb.show()
```







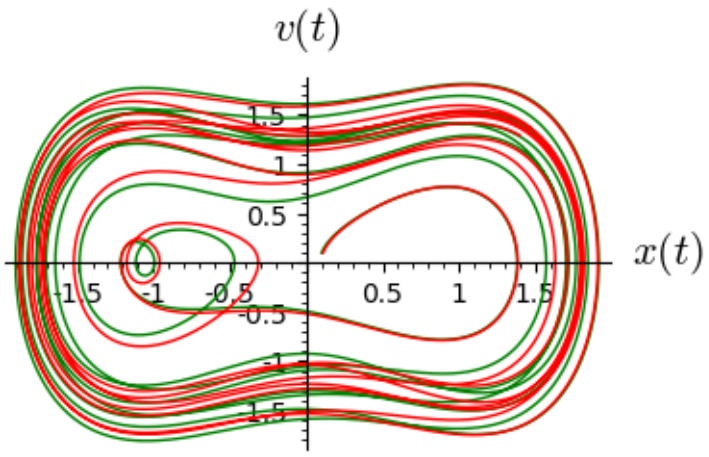


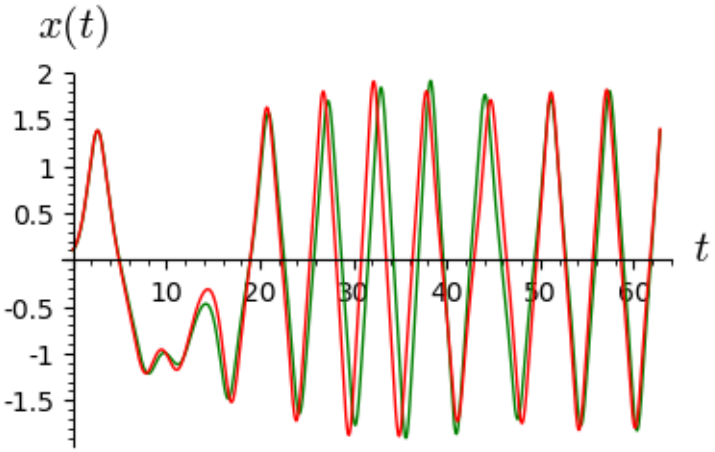
[6]:
```
(None, None, None, None)
```

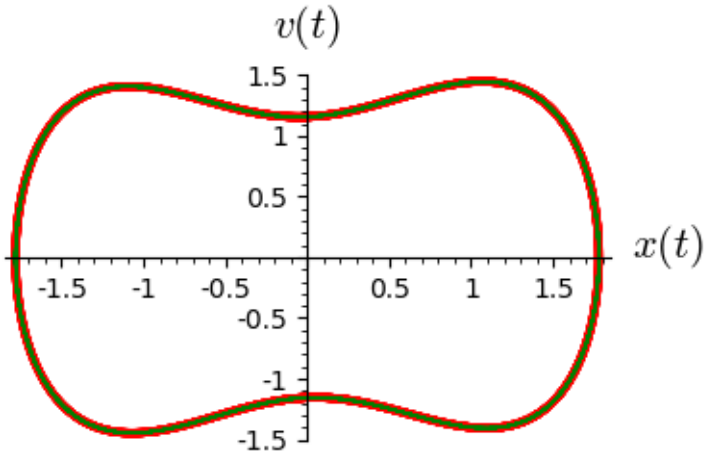
Let us now look at two trajectories starting from close initial conditions. Let us consider their initial and asymptotic (for long times) evolution.

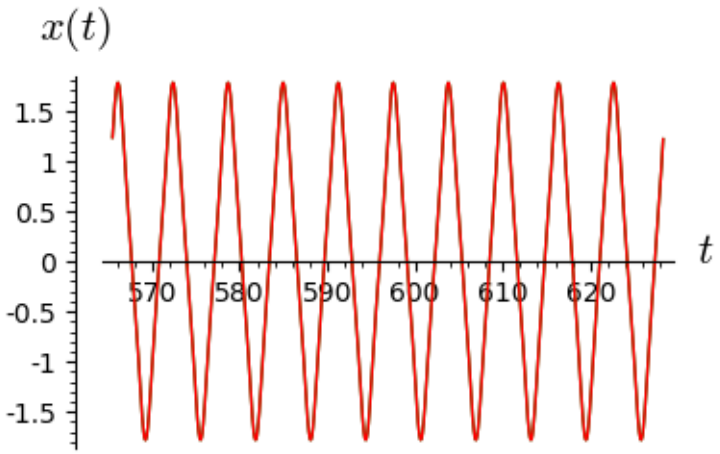
[7]:
```
reset()
# wykresy dla przypadku z tumieniem
var('x y z')
x01, y01, z01 = 0.1,0.1,0
x02, y02, z02 = 0.11,0.1,0

# sia
F = x-x^3
V = -integrate(F,x)
```

```
# tarcie: parametr gamma
g = 0.1
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,200*pi,0.01)
num1 = desolve_odeint(vector([dx,dy,dz]), [x01,y01,z01], T, [x,y,z])
num2 = desolve_odeint(vector([dx,dy,dz]), [x02,y02,z02], T, [x,y,z])

lnum = int(len(num1[:,0])/10)
trans1 = num1[:lnum]
asymp1 = num1[-lnum:]
trans2 = num2[:lnum]
asymp2 = num2[-lnum:]

# wykresy funkcji
lt = list_plot(zip(trans1[:,0],trans1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lt += list_plot(zip(trans2[:,0],trans2[:,1]), plotjoined=True, color='red')
rt = list_plot(zip(T[:lnum],trans1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rt += list_plot(zip(T[:lnum],trans2[:,0].tolist()), plotjoined=True, color='red')
lb = list_plot(zip(asymp1[:,0],asymp1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lb += list_plot(zip(asymp2[:,0],asymp2[:,1]), plotjoined=False, color='red')
rb = list_plot(zip(T[-lnum:],asymp1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rb += list_plot(zip(T[-lnum:],asymp2[:,0].tolist()), plotjoined=True, color='red')

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z rónymi warunkami pocztkowymi
$(x_{01},y_{01},z_{01}) = (%.2f,%.2f,%.2f)$
$(x_{02},y_{02},z_{02}) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x01,y01,z01,x02,y02,z02))

lt.show(),rt.show(),lb.show(),rb.show()
```


$v(t)$
1.5
1
0.5
-1.5
-1
-0.5
0.5
1
1.5
-0.5
-1
-1.5
$x(t)$


$x(t)$
2
1.5
1
0.5
-0.5
-1
-1.5
10
20
30
40
50
60
t


$v(t)$
1.5
1
0.5
-1.5
-1
-0.5
0.5
1
1.5
-0.5
-1
-1.5
$x(t)$



[7]: `(None, None, None, None)`

In the upper two diagrams, the regime of short times is presented. Because the 2 initial conditions are slightly different, the initial evolution is slightly different. The red and green colors are distinguishable in the upper right figure showing the evolution of $x(t)$ for short times. If we look at the regime of long times (the two lower diagrams), we will notice a deep similarity in evolution: the phase curves are closed so it is a simple periodic motion, resembling a slightly deformed $\sin(\alpha t)$ or $\cos(\alpha t)$ function. It is a periodic function with a characteristic one-only T period. Therefore, we say that there is a periodic motion at period 1. The two curves on the lower right figure are not distinguishable. You can do numeric experience and choose different initial conditions. We will see that trajectories tend to the same periodic solution, they are attracted to this periodic solution. In other words, this closed phase curve with period 1 is ATTRACTOR. This attractor is called a period-1 attractor or attractor of period 1. One could ask: what is the basin of attraction for this attractor? In order to answer this question, one should examine numerically, for example, the square of initial conditions (x_0, y_0) and select those initial conditions that tend to r the above phase curve of period 1. It turns out that the basin of attraction is a "decent" set whose edge is a smooth curve, just like in the case illustrated above for a system with only friction, without periodic force.

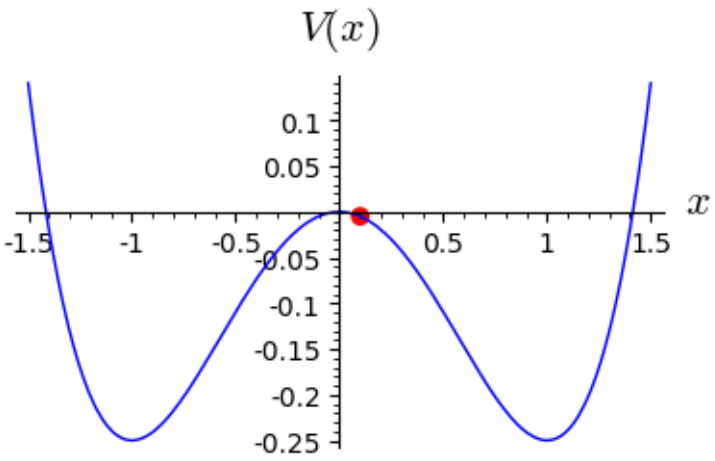
5.2.5 PERIOD-3 SOLUTIONS

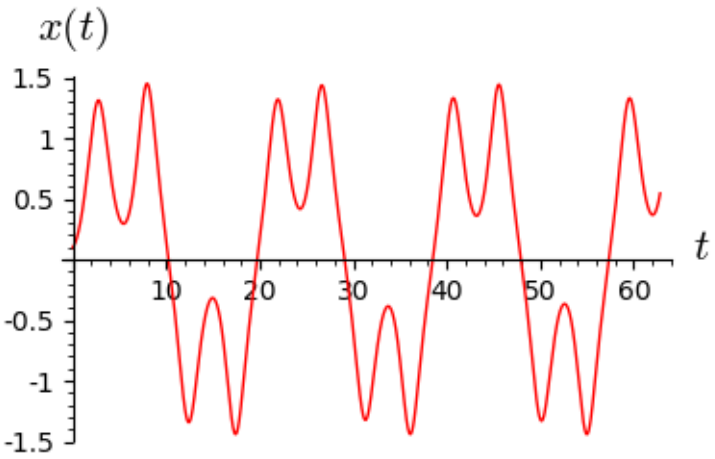
Let's assume the following parameter values:

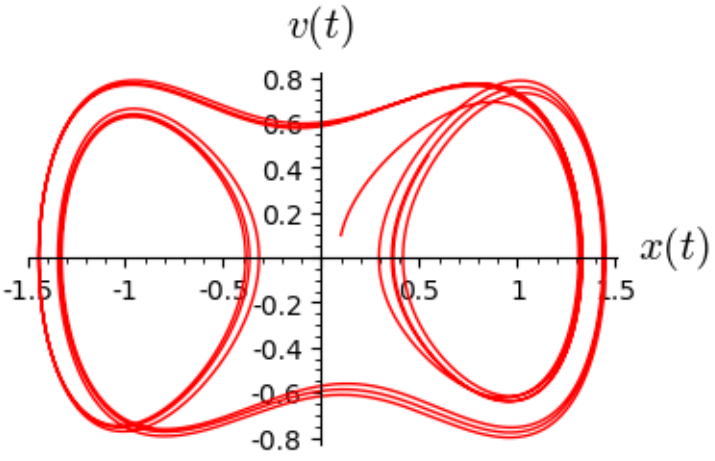
$$\gamma = 0.22, \qquad A = 0.3, \qquad \omega_0 = 1$$

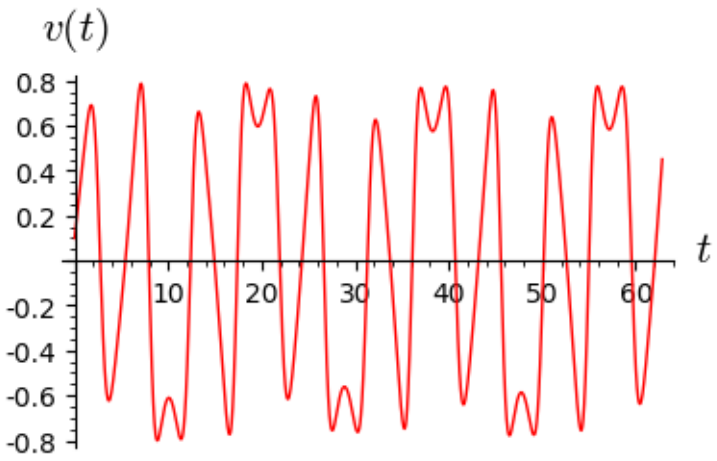
In this case, we also observe periodic motion, but a bit more complicated. It is not a simple periodic movement, but so-called period-3 solution, i.e. now the period is 3 times longer than in the previous case.

[8]:
```
# wykresy dla przypadku z tumieniem
var('x y z')
x0, y0, z0 = 0.1,0.1,0
kolor = 'red'

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.22
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,20*pi,0.01)
num = desolve_odeint(vector([dx,dy,dz]), [x0,y0,z0], T, [x,y,z])

# wykresy funkcji
xmin = 1.5
lt  = plot(V, (x,-xmin,xmin), figsize=4)
lt += point((x0,V(x=x0)), color=kolor, size=50, axes_labels=['$x$','$V(x)$'])
lb  = list_plot(zip(num[:,0],num[:,1]), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
rt  = list_plot(zip(T,num[:,0].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$x(t)$'], figsize=4)
rb  = list_plot(zip(T,num[:,1].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$v(t)$'], figsize=4)

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z warunkami pocztkowymi
$(x_0,y_0,z_0) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x0,y0,z0))

lt.show(),rt.show(),lb.show(),rb.show()
```


V(x)
0.1
0.05
-0.05
-0.1
-0.15
-0.2
-0.25
x
-1.5
-1
-0.5
0.5
1
1.5


x(t)
1.5
1
0.5
-0.5
-1
-1.5
t
10
20
30
40
50
60


v(t)
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
x(t)
-1.5
-1
-0.5
0.5
1
1.5

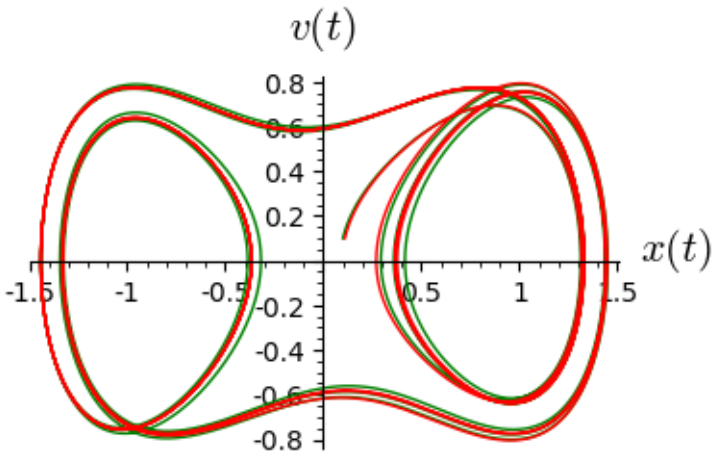


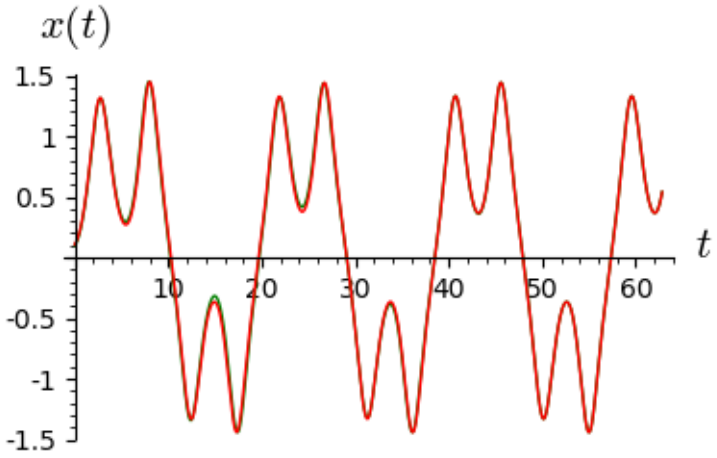
[8]: `(None, None, None, None)`

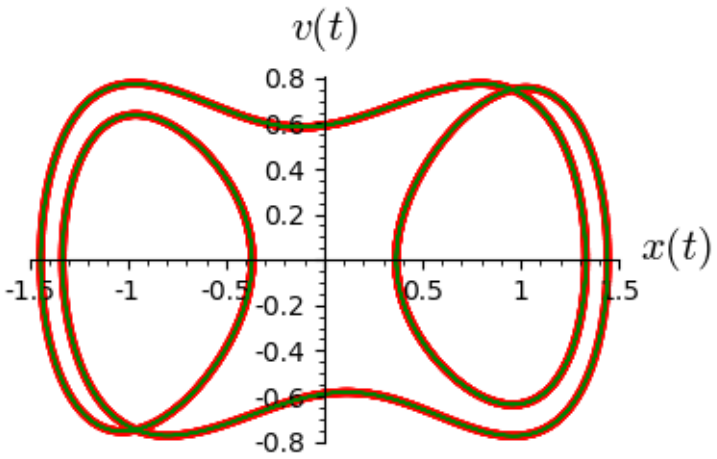
And again we will see how the initial evolution differs from that after a long time.

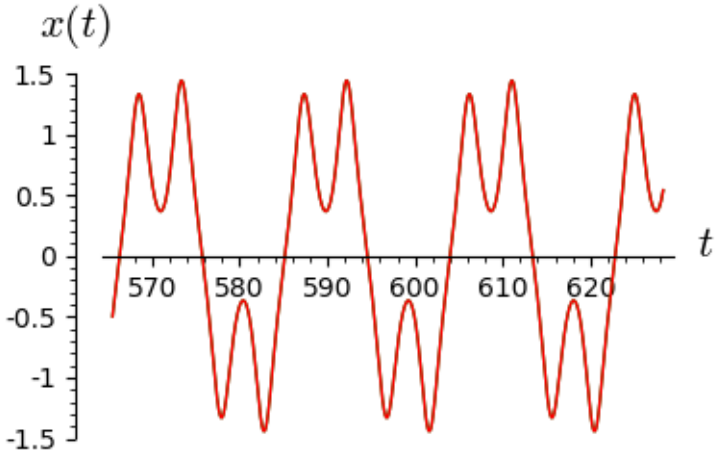
[9]:
```
reset()
var('x y z')
x01, y01, z01 = 0.10,0.1,0
x02, y02, z02 = 0.11,0.1,0

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.22
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,200*pi,0.01)
num1 = desolve_odeint(vector([dx,dy,dz]), [x01,y01,z01], T, [x,y,z])
num2 = desolve_odeint(vector([dx,dy,dz]), [x02,y02,z02], T, [x,y,z])

lnum = int(len(num1[:,0])/10)
```

```
trans1 = num1[:lnum]
asymp1 = num1[-lnum:]
trans2 = num2[:lnum]
asymp2 = num2[-lnum:]

# wykresy funkcji
lt = list_plot(zip(trans1[:,0],trans1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lt += list_plot(zip(trans2[:,0],trans2[:,1]), plotjoined=True, color='red')
rt = list_plot(zip(T[:lnum],trans1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rt += list_plot(zip(T[:lnum],trans2[:,0].tolist()), plotjoined=True, color='red')
lb = list_plot(zip(asymp1[:,0],asymp1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lb += list_plot(zip(asymp2[:,0],asymp2[:,1]), plotjoined=False, color='red')
rb = list_plot(zip(T[-lnum:],asymp1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rb += list_plot(zip(T[-lnum:],asymp2[:,0].tolist()), plotjoined=True, color='red')

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z rónymi warunkami pocztkowymi
$(x_{01},y_{01},z_{01}) = (%.2f,%.2f,%.2f)$
$(x_{02},y_{02},z_{02}) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x01,y01,z01,x02,y02,z02))
lt.show(),rt.show(),lb.show(),rb.show()
```




x(t)
1.5
1
0.5
-0.5
-1
-1.5
10
20
30
40
50
60
t


v(t)
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
-1.5
-1
-0.5
0.5
1
1.5
x(t)


x(t)
1.5
1
0.5
0
-0.5
-1
-1.5
570
580
590
600
610
620
t

[9]:
```
(None, None, None, None)
```

For long times, the phase curves are closed, but they are not curves like a deformed ellipse. These are curves with 2 loops. Nevertheless, the movement is periodic. Like the previous case, you can do numeric experience and choose different initial conditions. We will see that many trajectories tend to the same periodic orbit, they are attracted to this phase curve closure. In other words, this phase curve with period 3 is ATRACTOR. This attractor is called a periodic attractor with a period of 3 or 3-period attractor. The attraction pool for this attractor on the initial conditions plane (x_0, y_0) is a "decent" set of dimension 2 (ie a piece of plane) whose edge is a smooth curve.

5.2.6 CHAOTIC MOVEMENT

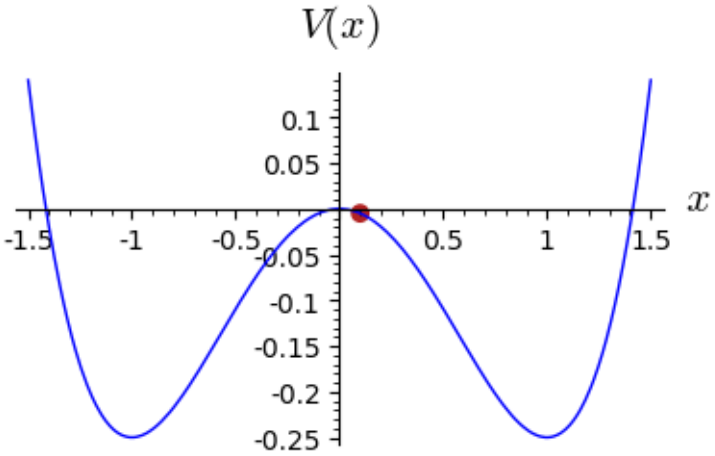
Let's assume the following parameter values:

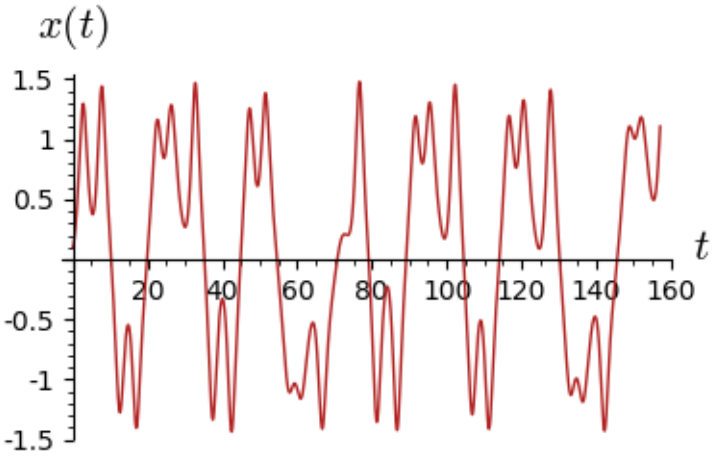
$$\gamma = 0.25, \qquad A = 0.3, \qquad \omega_0 = 1$$

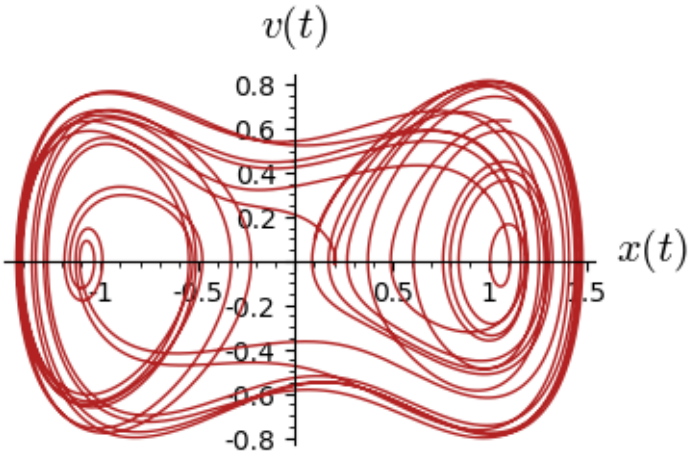
In this case, we observe a movement that seems to be extremely irregular, chaotic.

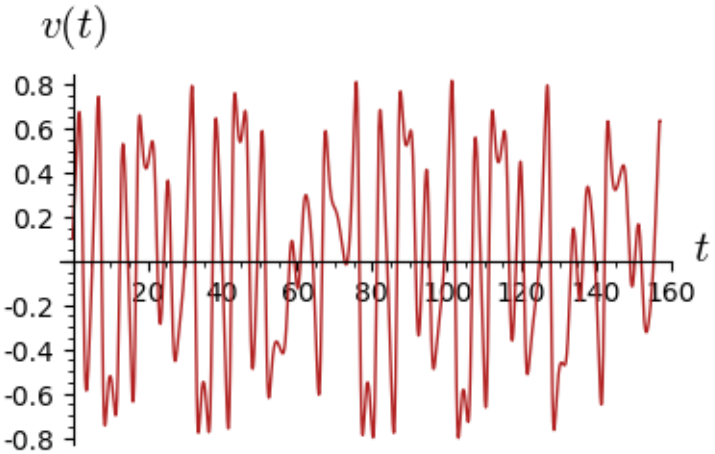
[10]:
```
# wykresy dla przypadku z tumieniem
var('x y z')
x0, y0, z0 = 0.1,0.1,0
kolor = 'firebrick'

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.25
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,50*pi,0.01)
num = desolve_odeint(vector([dx,dy,dz]), [x0,y0,z0], T, [x,y,z])

# wykresy funkcji
xmin = 1.5
lt  = plot(V, (x,-xmin,xmin), figsize=4)
lt += point((x0,V(x=x0)), color=kolor, size=50, axes_labels=['$x$','$V(x)$'])
```

```
lb  = list_plot(zip(num[:,0],num[:,1]), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
rt  = list_plot(zip(T,num[:,0].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$x(t)$'], figsize=4)
rb  = list_plot(zip(T,num[:,1].tolist()), plotjoined=True, color=kolor,␣
 ↪axes_labels=['$t$','$v(t)$'], figsize=4)

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z warunkami pocztkowymi
$(x_0,y_0,z_0) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x0,y0,z0))

lt.show(),rt.show(),lb.show(),rb.show()
```







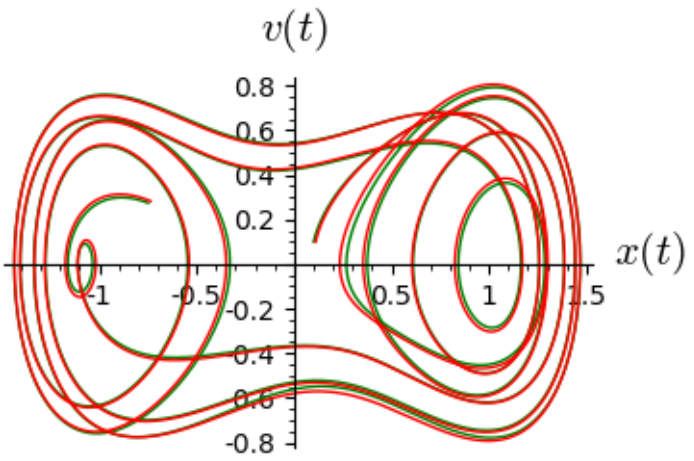


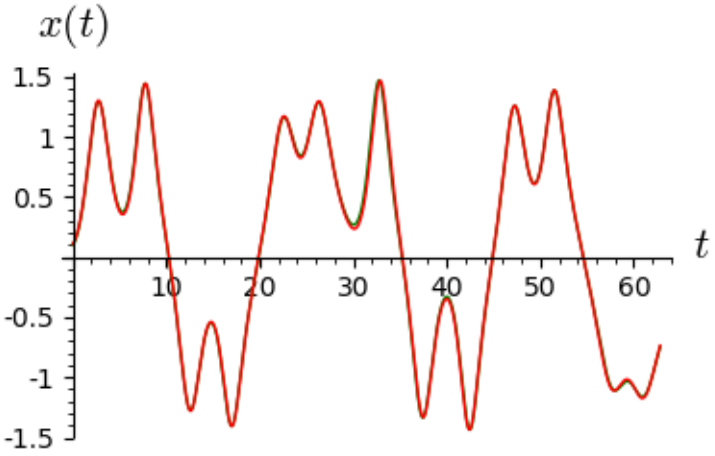
[10]:
```
(None, None, None, None)
```

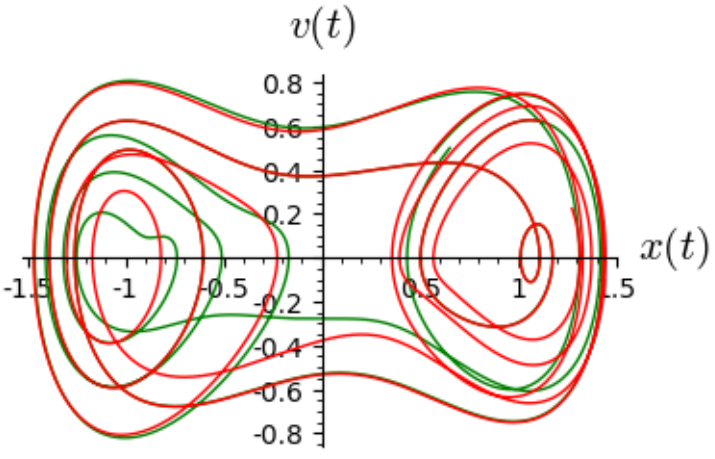
Let's see how solutions with 2 close initial conditions are evolving this time.

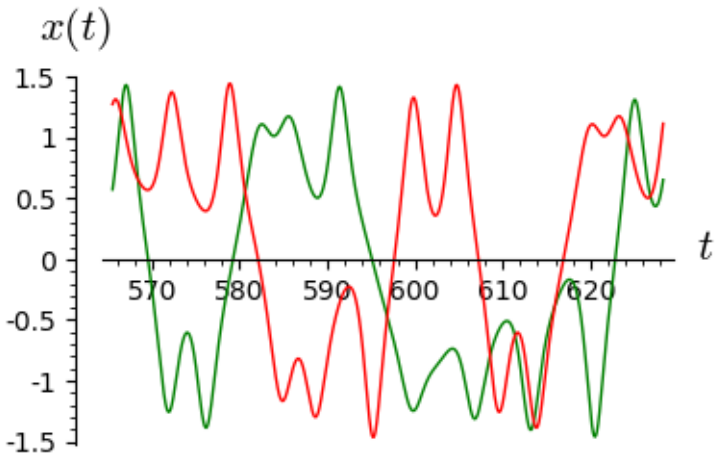
[11]:
```
reset()
# wykresy dla przypadku z tumieniem
var('x y z')
x01, y01, z01 = 0.1,0.1,0
x02, y02, z02 = 0.11,0.1,0

# sia
```

```
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.25
A = 0.3
w = 1

# ukad róniczkowych równa ruchu
dx = y
dy = F - g*y + A*cos(z)
dz = w

# numeryczne rozwiazanie równa ruchu
T = srange(0,200*pi,0.01)
num1 = desolve_odeint(vector([dx,dy,dz]), [x01,y01,z01], T, [x,y,z])
num2 = desolve_odeint(vector([dx,dy,dz]), [x02,y02,z02], T, [x,y,z])

lnum = int(len(num1[:,0])/10)
trans1 = num1[:lnum]
asymp1 = num1[-lnum:]
trans2 = num2[:lnum]
asymp2 = num2[-lnum:]

# wykresy funkcji
lt = list_plot(zip(trans1[:,0],trans1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lt += list_plot(zip(trans2[:,0],trans2[:,1]), plotjoined=True, color='red')
rt = list_plot(zip(T[:lnum],trans1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rt += list_plot(zip(T[:lnum],trans2[:,0].tolist()), plotjoined=True, color='red')
lb = list_plot(zip(asymp1[:,0],asymp1[:,1]), plotjoined=True, color='green',␣
 ↪axes_labels=['$x(t)$','$v(t)$'], figsize=4)
lb += list_plot(zip(asymp2[:,0],asymp2[:,1]), plotjoined=True, color='red')
rb = list_plot(zip(T[-lnum:],asymp1[:,0].tolist()), plotjoined=True,␣
 ↪color='green', axes_labels=['$t$','$x(t)$'], figsize=4)
rb += list_plot(zip(T[-lnum:],asymp2[:,0].tolist()), plotjoined=True, color='red')

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z rónymi warunkami pocztkowymi
$(x_{01},y_{01},z_{01}) = (%.2f,%.2f,%.2f)$
$(x_{02},y_{02},z_{02}) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x01,y01,z01,x02,y02,z02))

lt.show(),rt.show(),lb.show(),rb.show()
```


v(t)
0.8
0.6
0.4
0.2
-1
-0.5
-0.2
-0.4
-0.8
0.5
1
x(t)


x(t)
1.5
1
0.5
-0.5
-1
-1.5
10
20
30
40
50
60
t


v(t)
0.8
0.6
0.4
0.2
-1.5
-1
-0.5
-0.2
-0.4
-0.6
-0.8
0.5
1
x(t)



[11]: `(None, None, None, None)`

The initial evolution of two solutions is indistinguishable (because two initial conditions are very close to each other). After some characteristic time, sometimes called Lyapunov time, the trajectories start to diverge more and more, they start to diverge: see the red and green trajectories on the lower right figure.

In the chaotic regime, these two trajectories move away from each other in an exponentially fast way determined by the relation:

$$|x_1(t) - x_2(t)| = |x_1(0) - x_2(0)|e^{\lambda t}, \qquad \lambda > 0$$

or

$$|\Delta x(t)| = |\Delta x_0|e^{\lambda t}, \qquad \lambda > 0$$

where λ is called the Lyapunov exponent.

Separation and divergence of two trajectories increase and a dilemma arises: which trajectory is correct if our apparatus does not distinguish between close initial conditions. Determinism becomes illusory. We can not predict the proper evolution of the system. The chaotic regime presented above is not the only one. In the system there are many such regimes of parameters (γ, A, ω) in which chaotic movement appears. It should be noted that for long times many trajectories generated by different initial conditions behave very similarly, many trajectories are attracted. Here, too, there is an attractor and its basin of attraction. However, this attractor is strange: its dimension is not an integer and the attractor is a fractal. That's why it's called a strange attractor.

5.3 Problems:

1. Let $\gamma = 0.1$, $\omega_0 = 1.4$, $(x_0, y_0, z_0) = (-0.5, -0.2, 0)$. Change the parameter $A = 0.1, 0.32, 0.338, 0.35$. Observe the scenario of period-doubling:

(i) period-1 solutions

(ii) period-2 solutions

(iii) period-4 solutions

(iv) period-8 solutions

....

(v) is there an irregular, chaotic movement?

2. Examine the system behavior for the following parameter values: $\gamma = 1.35 - 1.38$, $A = 1$, $\omega_0 = 1$, $(x_0, y_0, z_0) = (0.0, 0.5, 0)$.
3. Do the same for $\gamma = 0.5$, $A = 0.34875$, $\omega_0 = 1$, $(x_0, y_0, z_0) = (0, 0, 0)$.

6 Period Doubling Bifurcation Route to Chaos

By changing the parameters of the system and initial conditions, we can control time evolution of the system. We have seen that in the previously analyzed system, there are periodic solutions. Periodic solutions can have a simple structure characterized by one characteristic period T (or frequency). There are also more complicated periodic solutions: with periods 2, 3, 4, etc. Note that the periodic motion of period 3 is repeated after 3 times longer than the period 1 movement. Therefore, the regularity of movement can be observed after 3 times longer. The period-20 solutions are repeated after a time 20 times longer than the period-1 solutions. Therefore, the regularity of movement is observed after time 20 times longer. Periodic movement with a period of 2000 is repeated after a time of 2000 times longer than that of period 1. Therefore, the regularity of motion can be observed after a time of 200 times longer. By increasing the periodicity of movement to infinity, we notice that the regularity of movement repeats after an infinite time, i.e. the movement becomes irregular for the observer. The trajectory looks as if it is random, chaotic. The motion is still deterministic, but complicated, erratic, irregular. In some cases, the system is extremely sensitive to initial conditions: for two different, but very little different initial conditions, the corresponding trajectories diverge from one another after some characteristic time. If we reduce the distance between the initial conditions, then the time after which you can distinguish between 2 trajectories is longer, but sooner or later, the trajectories begin to diverge. From a practical point of view, initial conditions can be given with finite accuracy, but not with zero accuracy, as assumed in mathematical theorems. Therefore, in a regime in which the system is sensitive to initial conditions, in practice the uncertainty of initial conditions causes uncertainty of time evolution. This can be formulated in the mathematical sense in the following way:

Let $x(t)$ be the trajectory with the initial condition $x(0)$, and $X(t)$ is e the trajectory with the initial condition $X(0)$. Let two initial conditions differ by a small amount:

$$|X(0) - x(0)| = \epsilon_0$$

where |...| means the distance with respect to a prescribed metrix. Ley us say that: If they differ by such a size or smaller, then they are indistinguishable to us; we treat them as the same one trajectory within the measurement error. We ask what the difference is

$$|X(t) - x(t)| = \epsilon(t)$$

after time t due to the uncertainty of initial condition ϵ_0. Sensitivity to initial conditions means that 2 trajectories move away from each other at a very fast, exponential rate:

$$\epsilon(t) = \epsilon_0 e^{\lambda t}$$

where λ is called the exponent of Lyapunov. If $\lambda 0$ the trajectories diverge. If $\lambda < 0$ or $\lambda = 0$ two trajectories do not diverge and we can predict evolution of the system.

Let δ be the accuracy of our apparatus to distinguish various trajectories. If $\epsilon(t) < \delta$ then two trajectories become indistinguishable to us. If however $\epsilon(t) > \delta$ (when two trajectories become distinguishable to us), then we can not predict further evolution of the system: We do not know which trajectory starts from the initial point $x(0)$ and which starts from $X(0)$. Therefore we do not know which trajectory is correct: $x(t)$ or $X(t)$? How long do our predictions lose meaning? After such time that

$$\epsilon(t) = \epsilon_0 e^{\lambda t} > \delta$$

that is after time

$$t_l > \frac{1}{\lambda} \ln \left[\frac{\delta}{\epsilon_0} \right]$$

For times $t < t_l$ predictions are correct.

Let us assume that the initial state can be fixed with the accuracy of $\epsilon_0 = 10^{-8}$ and let $\delta = 10^{-5}$ be satisfied tolerance for us (although it is 3 orders of magnitude worse than for the initial condition). How long are our predictions acceptable. For times $t \in (t_0 = 0, t_1)$, where:

$$t_1 \approx \frac{1}{\lambda} \ln \left[\frac{\delta}{\epsilon_0} \right] = \frac{1}{\lambda} \ln \left[\frac{10^{-5}}{10^{-8}} \right] = \frac{1}{\lambda} \ln \left[10^3 \right] = \frac{3}{\lambda} \ln 10$$

Suppose someone is able to prepare the initial state with much better accuracy, namely 1000 times better, i.e. $\epsilon_0 = 10^{-11}$. How long can we predict the evolution of the system? The answer is: for times $t \in (0, t_2)$ where:

$$t_2 \approx \frac{1}{\lambda} \ln \left[\frac{10^{-5}}{10^{-11}} \right] = \frac{1}{\lambda} \ln \left[10^6 \right] = \frac{6}{\lambda} \ln 10 = 2t_1$$

This is just only 2 times longer !! You can see that when the system is in a chaotic regime, the predictability time is very limited. Increasing the accuracy of initial conditions by 1000 times results in extending the predictability time by only 2 times. This is the problem with the weather forecast. We can increase the network of measuring points, and weather prediction is only reasonable for a few days ahead.

The question of whether the system shows chaotic properties or not, is not easy to answer. Because the system of differential equations usually can not be solved analytically, one must rely on computer methods. On one hand, the system is sensitive to initial conditions, on the other hand the numerical method itself and computer calculations are burdened with errors that can not be eliminated. It may happen that it is not the property of the system but computer artifacts that create the illusion of chaos. Fortunately, there are several characteristics which allow to solve this proplem. Here are the characteristics:

1. Lyapunov exponents λ_i
2. Power spectrum of the signal $P(\omega)$
3. $C(\tau)$ - correlation function
4. Poincare mapping
5. Kolmogorov entropy K

Examination of all characteristics is onerous and time-consuming, but eliminates the possibility of confusion in determining chaoticity. We will present the main features of these quantities that occur in a chaotic and non-chaotic regime.

[1]:
```
#
```

6.1 Route to chaos: period doubling

We will now present a standard scenario of the transition to chaos, which is called the route to chaos by period doubling. It is a universal scenario, occurring both in systems with continuous time and in discrete systems. It has been confirmed in many experiments with a variety of physical systems. It is based on the same idea presented below. This scenario is regular: bifurcations occur that change the periodicity of periodic orbits. The period 1 orbit (i.e., the period is the same as the external time periodic force) bifurcates into the orbit of period 2 if we change, for example, the amplitude A of the driving external force. The period 2 orbit is bifurcated into the orbit of period 4, which in turn bifurcates into the orbit of period 8. This scenario is repeated when the amplitude A changes. In the case shown in the figures below, the amplitude bifurcation values are (these are not exact but approximated values):

$$A_1 = 0.34357; \quad A_2 = 0.35506; \quad A_3 = 0.35785; \quad A_4 = 0.35846; \quad \ldots\ldots \quad A_\infty = 0.3586$$

The period of regularity of the orbit increases, until the period is infinitely long and the orbit appears to be chaotic. In such scenarios, there is a universal parameter, called the Feigenbaum constant. It is defined as the limit of the sequence

$$\delta_n = \frac{A_n - A_{n-1}}{A_{n+1} - A_n}$$

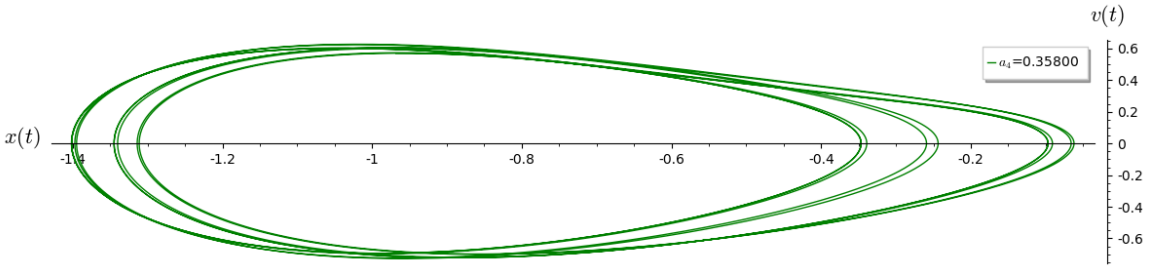
The limit of this sequence is

$$\lim_{n\to\infty} \delta_n = 4.6692.....$$

This is the value of the Feigenbaum constant. The same number appears in many continuous and discrete systems, although the dynamics can be completely different. To this day, it is not known why it is so, but it is. Some call it enigmatically the "class of universality" of chaotic phenomena.

[2]:
```
# wykresy dla przypadku z tumieniem
var('x y z')
x0, y0, z0 = -0.5,-0.1,0
kolor = ['blue','red','green','black','orange']

# sia
F = x-x^3
V = -integrate(F,x)

# tarcie: parametr gamma
g = 0.5
w = 1
#punkty bifurkacji: 0.34357;  0.35506; 0.35785; 0.35846;  ostatni 0.3586
Akeys = ['$a_1$','$a_2$','$a_3$','$a_4$']
Aval  = [0.325,0.354,0.357,0.358]
```

```
A = dict(zip(Akeys,Aval))
p = A
j=0
for a in A.keys():
    # ukad róniczkowych równa ruchu
    dx = y
    dy = F - g*y + A[a]*cos(z)
    dz = w

    # numeryczne rozwiazanie równa ruchu
    T = srange(0,100*pi,0.01)
    num = desolve_odeint(vector([dx,dy,dz]), [x0,y0,z0], T, [x,y,z])
    figsize = [12,3] if a == '$a_4$' else 3.5
    start, stop = int(len(num[:,0])*0.8), len(num[:,0])
    p[a] = list_plot(zip(num[:,0][start:stop],num[:,1][start:stop]), \
                     plotjoined=True, color=kolor[j],␣
 ↪axes_labels=['$x(t)$','$v(t)$'],\
                     legend_label='%s=%.5f'%(a,A[a]), figsize=figsize)
    j+=1

html("""Ukad równa róniczkowych
$\dot{x} = %s$
$\dot{y} = %s$
$\dot{z} = %s$
z warunkami pocztkowymi
$(x_0,y_0,z_0) = (%.2f,%.2f,%.2f)$
"""%(dx,dy,dz,x0,y0,z0))

table([[p['$a_1$'],p['$a_2$'],p['$a_3$']]])
p['$a_4$'].show()

PunktyBifurkacji = [0.34357,0.35506,0.35785,0.35846]
i = 2
delta_2 = (PunktyBifurkacji[i-1] - PunktyBifurkacji[i-2])/(PunktyBifurkacji[i] -␣
 ↪PunktyBifurkacji[i-1])
i = 3
delta_3 = (PunktyBifurkacji[i-1] - PunktyBifurkacji[i-2])/(PunktyBifurkacji[i] -␣
 ↪PunktyBifurkacji[i-1])

table([['$\delta_1$',delta_2],['$\delta_2$',delta_3],['$\dots$',''],['$\lim_{n \;␣
 ↪->\;  \infty} \delta_n$',4.6692]])
```



[2]:

`$\delta_1$`	`4.11827956989245`
`$\delta_2$`	`4.57377049180331`
`$\dots$`	
`$\lim_{n \; ->\;  \infty} \delta_n$`	`4.66920000000000`

6.2 Lyapunov exponents

For the Duffing oscillator driven by the time-periodic force, the phase space is 3-dimensional. Therefore in fact there are 3 Lyapunov exponents. To explain this problem, we need to consider the set of initial conditions that form e.g. a ball K in the 3-dimensional phase space. There are 3 axes and radiuses into the direction of three axes have the same value, say $R_1 = R_2 = R_3$. If we start to iterate equations for $x(t), y(t), z(t)$ starting from all initial points of the ball K, then the set of points contained initially in the ball will change its shape. The ball will no longer be a ball. The ball is deformed in all three directions of (x, y, z) in the phase space and the shape is determined by the three Lyapunov exponents $\lambda_1, \lambda_2, \lambda_3$. If the system is chaotic, then the ball deformes in such a way that R_1 increases in one direction and R_2 decreases in the other one, taking the shape of e.g. an ellipsoid. In this case, we can define three Lyapunov exponents measuring the deformations of the ellipsoid in three mutually perpendicular directions. The number of Lyapunov exponents is therefore dependent on the dimension of phase space of the system. They signs are one of the criteria of chaotic motion:

if at least one of the Lyapunov exponent is positive, the system exhibits chotic properties.

If the ellipsoid extends in one direction, its axis (say R_1) in this direction increases and the Lyapunov exponent is positive, $\lambda_1 > 0$. If e.g. R_2 decreases then the corresponding Lyapunov exponent is negative, $\lambda_2 < 0$.

The two trajectories initially being close to each other propagate over time at the distance (one dimension) $l(t) \propto e^{\lambda_1 t}$, in turn the surface (two dimensions) changes at the rate $S(t) \propto e^{(\lambda_1+\lambda_2)t}$ and the volume (three dimensions) M changes at the rate $M(t) \propto e^{(\lambda_1+\lambda_2+\lambda_3)t}$. In the chaotic regime, at least one of the Lyapunov exponents is positive. This means that in the phase space trajectories diverge in one direction. If all three exponents are negative, the system is in regular (regular, quasi-periodic). There are no analytical methods to calculate the Lapunov exponents. Numeric methods are also not simple. The algorithms used to determine $\lambda_1, \lambda_2, \lambda_3$ can be found in the literature.

In the case of the Duffing oscillator, partial information about the Lyapunov exponents can be obtained.

1. The third equation for the auxiliary z variable can be solved with the function

$$z(t) = \omega t + c$$

Certainly two close trajectories $z_1(t) = \omega t + c_1$ and $z_2(t) = \omega t + c_2$ for the moment $t = 0$ do not diverge exponentially because

$$|z_1(t) - z_2(t)| = |c_1 - c_2|$$

Therefore, one of the exponents is zero, e.g.

$$\lambda_3 = 0$$

2. Recall here that the Duffing oscillator is described by the set of equations

$$\dot{x} = F_1 = y, \qquad x(0) = x_0$$

$$\dot{y} = F_2 = x - x^3 - \gamma y + A\cos z, \qquad y(0) = y_0$$

$$z = F_3 = \omega, \qquad z(0) = 0$$

Let's examine how the phase volume of the system changes over time. To do this, we must calculate the divergence of the vector field

$$div\vec{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z} = -\gamma$$

It means that the phase volume in the 3-dimensional space is decreasing with time:

$$M(t) \propto e^{-\gamma t}$$

On the other hand, as we said above,

$$M(t) \propto e^{(\lambda_1+\lambda_2+\lambda_3)t}$$

It follows that the sum of all exponents is constant and equals to

$$\lambda_1 + \lambda_2 + \lambda_3 = -\gamma$$

i.e. only the damping constant γ determines the rate of phase volume reduction. Because $\lambda_3 = 0$, we get an interesting relationship between the other two exponents:

$$\lambda_1 + \lambda_2 = -\gamma$$

In a chaotic regime, one of the exponents is positive, eg $\lambda_1 > 0$ and the second exponent must be negative, e.g. $\lambda_2 < 0$

$$\lambda_1 > \lambda_3 > \lambda_3, \qquad \lambda_1 > 0, \qquad \lambda_2 < 0, \qquad \lambda_3 = 0$$

We draw attention to the fact that the ellipsoid in the three-dimensional phase space extends in one direction, shrinks in the other direction and does not change in the third direction, and the ellipsoid volume decreases all the time. This is what looks like in a chaotic regime. In a non-chaotic regime: the ellipsoid can shrinks in one direction, can shrinks in the other direction, and does not change in the third direction, and the ellipsoid volume decreases all the time. The attractors that we showed previously exist in the 3-dimensional phase space, but because the phase volume is constantly decreasing, dimension of the attractor e must be less than 3. Under non-chaotic regime, the n-period attractors (curves) have the dimension 1. Attractors in the regime chaotic have a dimension greater than 1, but smaller than 3. Kaplan and Yorke (1979) hypothesized that there is a connection between the fractal dimension of the attractant D_A and the Lyapunov exponents. This relation has the form:

$$D_A = 2 + \frac{\lambda_1}{|\lambda_3|} > 2$$

If we analyze the attractor's dimension in the Poincarego mapping (on the plane), then this dimension is smaller:

$$d_A = D_A - 1$$

This is only a hypothesis, but in many cases confirmed by numerical experiments.

[3]:
```
#
```

6.3 Power spectrum

This is another quantity that can be an indicator of chaotic behavior of the deterministic system. The notion of power spectrum is well established in the theory of signals, treated as an information carrier. In general, signals can be deterministic (as in our case) and random (stochastic). In the engineering sense, the signal is any function of time. Signals which are represented by distributions (generalized functions) are lso considered. Only few simple signals can be described by mathematical formulas. Most of the signals we encounter in practice are so complex and irregular that their direct description as a function of time is troublesome. Therefore, you should use their different representations. The representation of the signal is a kind of symbolic description, sometimes with a significant degree of abstraction. Its essence is that it contains full information about the signal, although usually expressed in a different language than the direct language in terms of the time function. This means that knowing the signal, we can clearly determine its representation, and knowing this representation - reproduce the signal. There are many ways to represent signals. One of them is Fourier analysis using Fourier transforms or Fourier series.

Let us recall the concept of Fourier transform of functions or distributions. In the simplest terms, the Fourier transform $\hat{f}(\omega)$ of the function $f(t)$ is the integral

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} \mathrm{e}^{i\omega t} f(t) dt$$

where ω is any real number.

When we are interested in analysis of the signal represented by $f(t) = (x(t), y(t), z(t), ...)$, we define a slightly different Fourier transform as the limiting value of the integral:

$$\hat{f}(\omega) = \lim_{T\to\infty} \int_{0}^{T} \mathrm{e}^{i\omega t} f(t) dt$$

In practice, we never make the exact limit of $T \to \infty$, but we consider a sufficiently long time when a steady state is reached and transient effects in evolution disappear. Due to the $\mathrm{e}^{i\omega t}$ the Fourier transform is a complex function. Therefore, the function in the form

$$P(\omega) = |\hat{f}(\omega)|^2$$

is analysed. It is called the power spectrum of the signal $f(t)$. In some cases, it has an interpretation of power, and the number of ω is a frequency that is positive, $\omega > 0$. In general, its relationship with power (in the physical sense) is loose. This power spectrum is defined differently than in the theory of stochastic stationary processes where it is the Fourier transform of the correlation function $C(t)$ of the stochastic process. In order to have an intuition about the properties of the Fourier transform and the power spectra, it suffices to consider several cases of the function $f(t)$.

Case 1: One harmonics (monochromatic wave)

$$f_1(t) = A\cos(\Omega t), \qquad \hat{f}_1(\omega) = A\int_{0}^{\infty} \mathrm{e}^{i\omega t}\cos(\Omega t) dt = \frac{\pi}{2} A\delta(\omega - \Omega)$$

The Fourier transform is the Diraca delta δ, i.e. there is one peak in the power spectrum (which in practice is always finite).

Case 2: Several harmonics

$$f_2(t) = \sum_{k=1}^{n} A_k \cos(\Omega_k t), \qquad \hat{f}_2(\omega) = \sum_{k=1}^{n} A_k \int_{0}^{\infty} \mathrm{e}^{i\omega t}\cos(\Omega_k t) dt = \frac{\pi}{2} \sum_{k=1}^{n} A_k \delta(\omega - \Omega_k)$$

The Fourier transform is the sum of the shifted Dirac δ deltas, i.e. in the power spectrum there is a series of peaks (which in practice are always finite). Note that for Fourier transforms defined in this way, there is no power spectrum according to the above definition because in the strict mathematical sense $\delta^2(\omega - \Omega)$ is not defined. However, we need a practical method to check the chaoticity of the process and we usually sample the signal for discrete values of time $(t_0, t_1, t_2, ...t_{N-1})$. Therefore, we should use the Discrete Fourier Transform for the signal respresented by the sequence

$$\{x_0, x_1, x_2, ..., x_{N-1}\}$$

It transforms into a finite sequence of amplitudes

$$\{A_0, A_1, A_2, ..., A_{N-1}\}$$

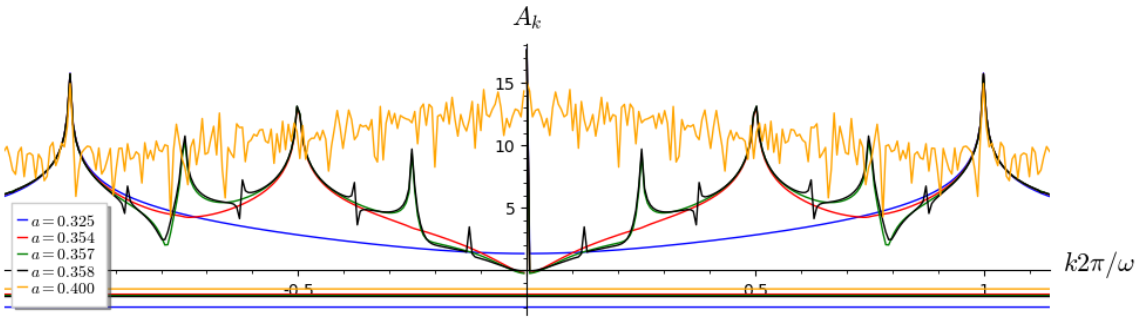
according to the relations:

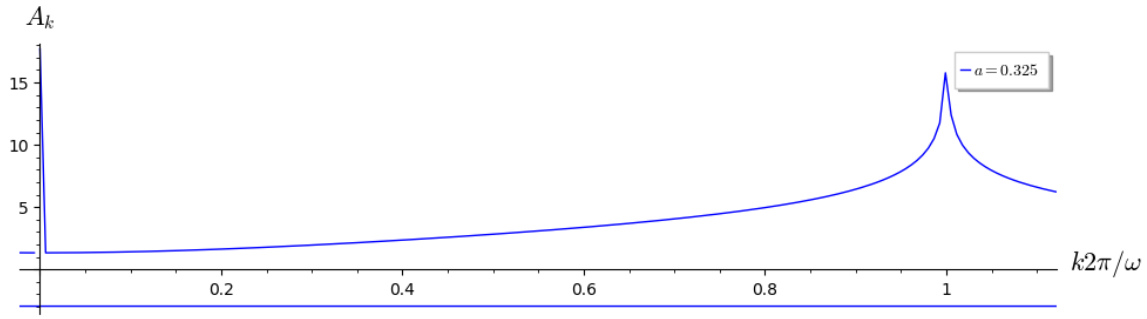
$$A_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n/N}, \qquad x_n = \frac{1}{N}\sum_{k=0}^{N-1} A_k e^{2\pi i k n/N}$$

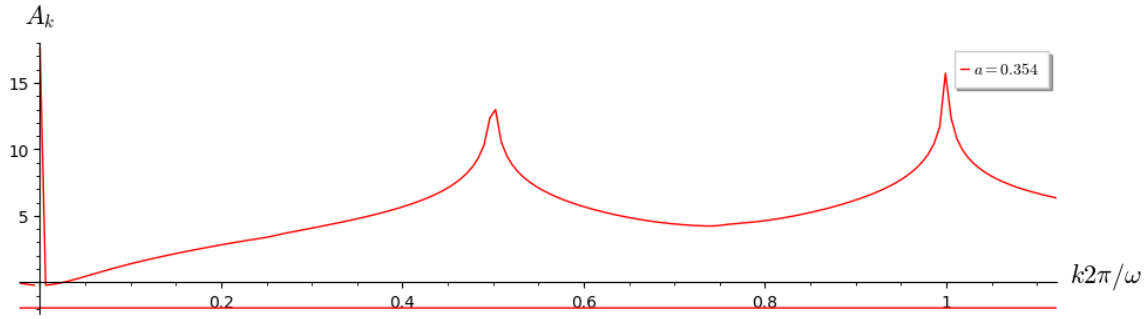
For a sufficiently large number of N (in many cases 100 is sufficient), the compatibility between the true Fourier transform and the Discrete Fourier Transform is surprisingly good.

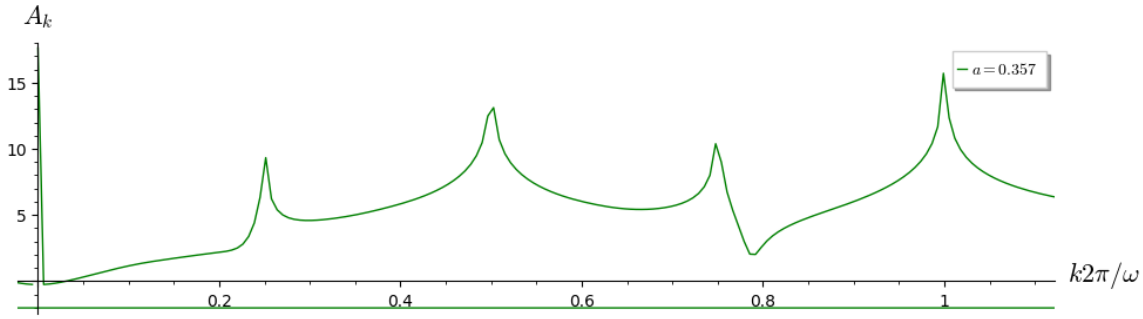
[4]:
```
var('x y z')
g, w0 = 0.5, 1
x0, y0, z0 = 0.1, 0.1, 0

Aval = [0.325,0.354,0.357,0.358,0.4]
kolor = ['blue','red','green','black','orange']
p = []

j = 0
for a in Aval:
    dx = y
    dy = x - x**3 - g*y + a*cos(z)
    dz = w0

    h = 0.1
    T = 1100
    skip = 100
    iskip = int(skip/h)
    listT = srange(0,T,h, include_endpoint=0)
    num = desolve_odeint(vector([dx, dy, dz]), [x0, y0, z0], listT, [x,y,z])
    iks = num[:,0].tolist()[iskip:]

    freq = [i/(T-skip) for i in range(len(iks)/2)] +\
           [-len(iks)/(T-skip) + i/(T-skip) for i in range(len(iks)/2,len(iks))]
    freq = [f*2.*n(pi)/w0 for f in freq]

    vx = vector(iks)
    A = vx.fft().apply_map(lambda x:x.abs2())
    p.append(list_plot(zip(freq,A.apply_map(lambda x:x.log())),plotjoined=True, \
                       color=kolor[j], legend_label=r"$a = %.
 ↪3f$"%a,figsize=[10,3]))

    j += 1

xx = 1.1
sum(p).show(figsize=[10,3],xmin=-xx,xmax=xx,axes_labels=[r'$k 2 \pi/
 ↪\omega$',r'$A_k$'])
for _p in p:
```

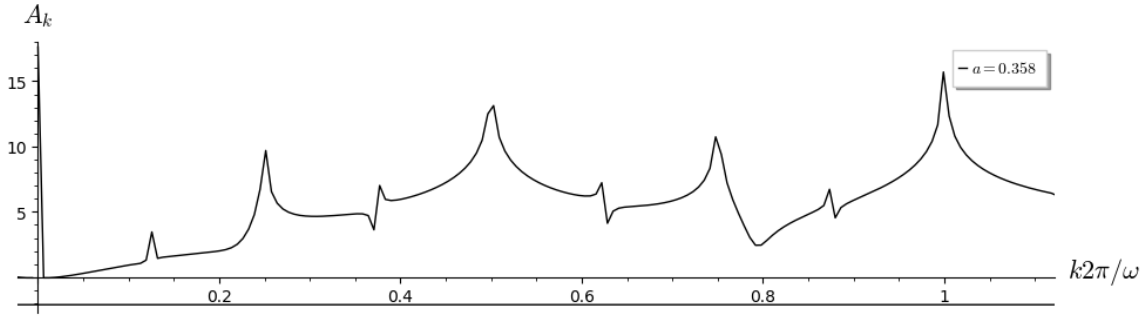
```
show(_p,xmin=0,xmax=xx,axes_labels=[r'$k 2 \pi/\omega$',r'$A_k$'])
```

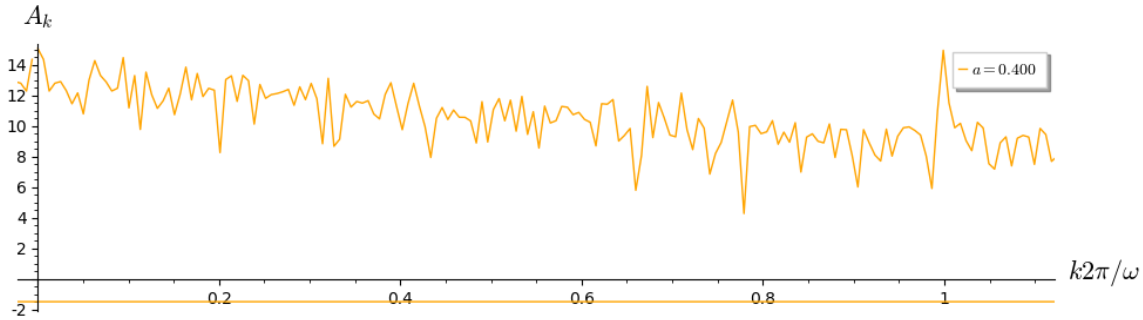












6.4 Correlation function

If we analyse a deterministic process, the meaning of average values is not useful. But if the deterministic process is ergodic (a difficult concept!) then average values of some quantities are well defined and the average over realizations of the proces (or over an ensemble) is equivalent to averaging over time. If the process is additionally stationary, the correlation function $C(\tau)$ for the deterministic process can be defined. In our case: for coordinate or velocity, it is defined by the relationship:

$$C(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T [x(t+\tau) - \langle x(t+\tau)\rangle][x(t) - \langle x(t)\rangle]dt, \qquad \langle x(t)\rangle = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)dt$$

In particular, for $\tau = 0$ we get a variance of the process:

$$C(0) = \lim_{T\to\infty} \frac{1}{T} \int_0^T [x(t) - \langle x(t)\rangle]^2 dt$$

or

$$C(0) = \langle [x(t) - \langle x(t)\rangle]^2 \rangle = \langle x^2(t)\rangle - \langle x(t)\rangle^2.$$

If the solution $x(t)$ is known then depending on the form of this solution, SAGE will also manage to solve the integral. If the analytical formula is beyond the possibilities of symbolic computation, we can always generate a time series $x = \{x_1, x_2, \dots\}$. The implementation of the correlation function in SAGE will not be a numerical problem. We can try to formulate the problem ourselves, or use the finance package methods.

[5]:
```
# samodzielna definicja
def korelator(dane, tau=0):
    ret = None
    if tau == 0:
        return 1
    else:
        # funkcja autokorelacji jest symetryczna
        tau = abs(tau)

        # odjcie redniej
        m = mean(dane)
        dane = [dane[i] - m for i in xrange(len(dane))]

        v = vector(dane)
        sigma = v.dot_product(v)
        if tau < len(dane):
            ret = v[:-tau].dot_product(v[tau:])
        ret /= sigma
    return ret
```

Now we calculate this correlation function for the Duffing oscillator.

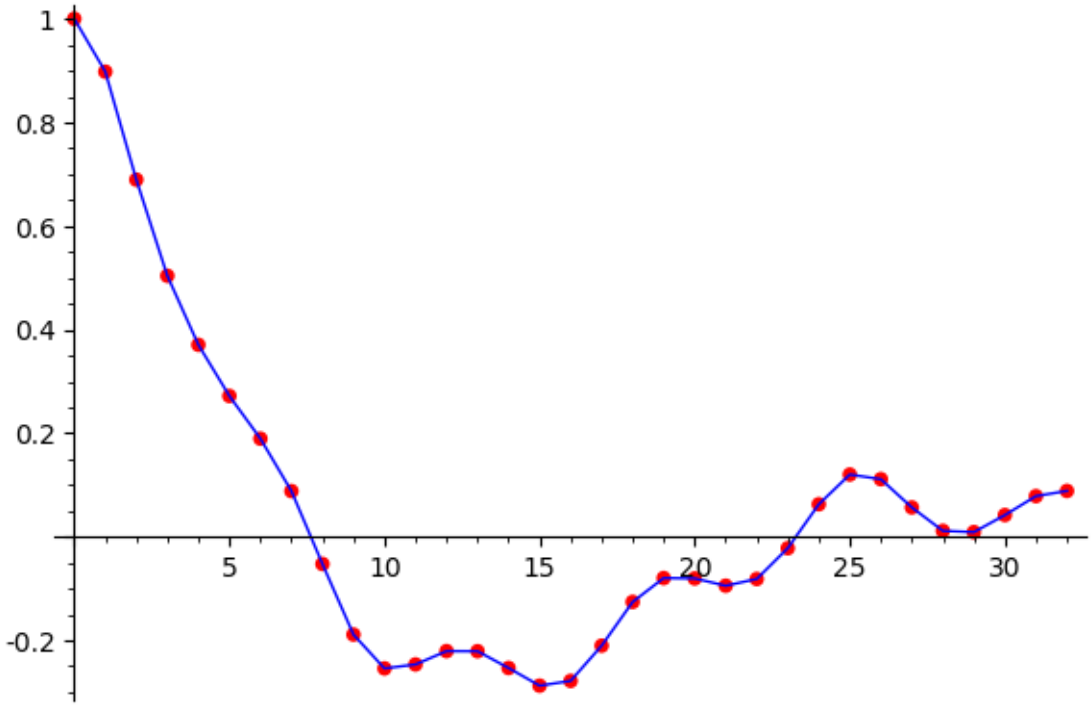
[6]:
```
var('x y z')
a, g, w0 = 0.3, 0.26, 1
x0, y0, z0 = 0.1, 0.1, 0

dx = y
dy = x - x**3 - g*y + a*cos(z)
dz = w0

h = 0.1
```

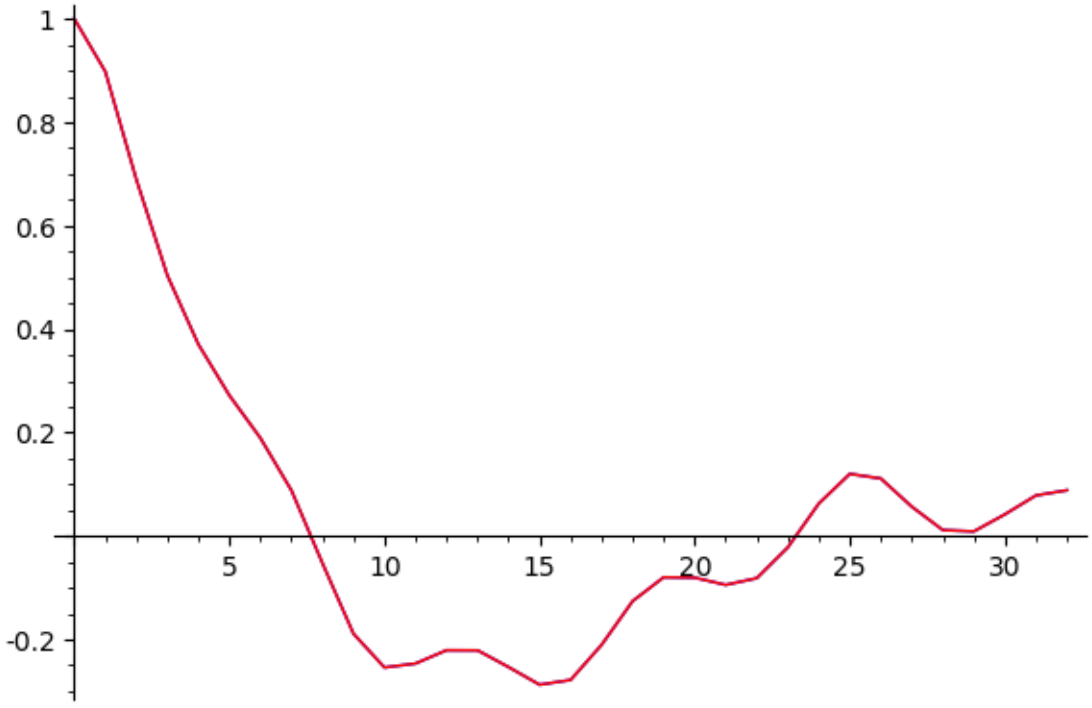
```
T = 1000
listT = srange(0,T,float(h), include_endpoint=True)
num = desolve_odeint(vector([dx, dy, dz]), [x0, y0, z0], listT, [x,y,z])
```

We will use both our function and the finance package embedded in SAGE to calculatde the (auto) function of correlation for position and for speed.

[7]:
```
#x
dane = num[:,0].tolist()

# nasz korelator
my_acorr = [korelator(dane,i*10) for i in range(33)]

# funkcja SAGE
v = finance.TimeSeries(dane)
sage_acorr = [v.autocorrelation(i*10) for i in range(33)]
```

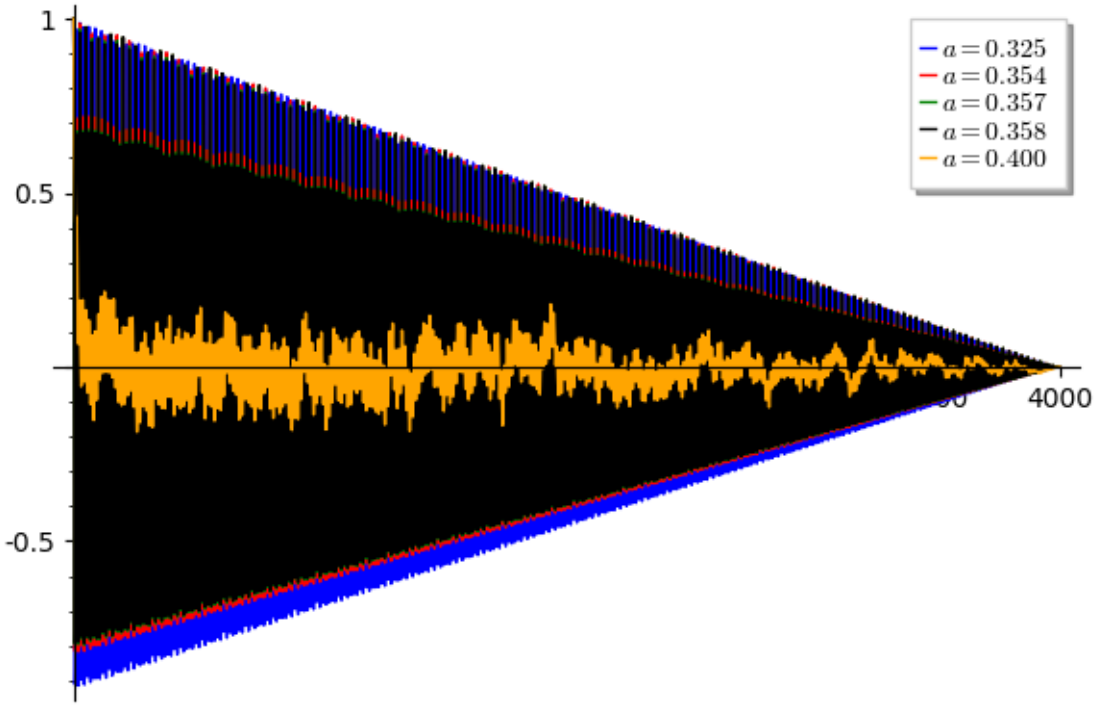
[8]:
```
im=3

(list_plot(my_acorr, plotjoined=True) +\
 list_plot(sage_acorr, plotjoined=False, size=30, color='red')).show()
#.save('sage_chII012_0%d.pdf'%im,figsize=[8,3],␣
 ↪axes_labels=[r"$\tau$",r"$C(\tau)$"])

(list_plot(my_acorr, plotjoined=True) +\
 list_plot(sage_acorr, plotjoined=True,markersize=30, color='red')).show()
#.save('sage_chII012_0%d.png'%im,figsize=[8,3],␣
 ↪axes_labels=[r"$\tau$",r"$C(\tau)$"])
```

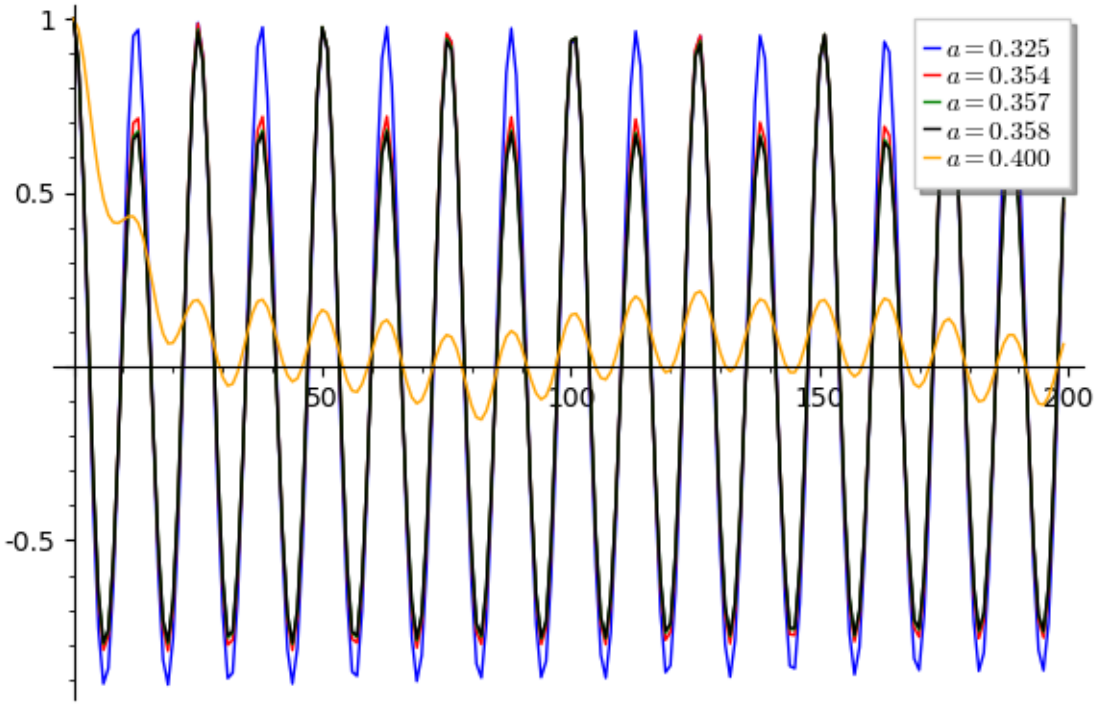

1
0.8
0.6
0.4
0.2
-0.2
5
10
15
20
25
30

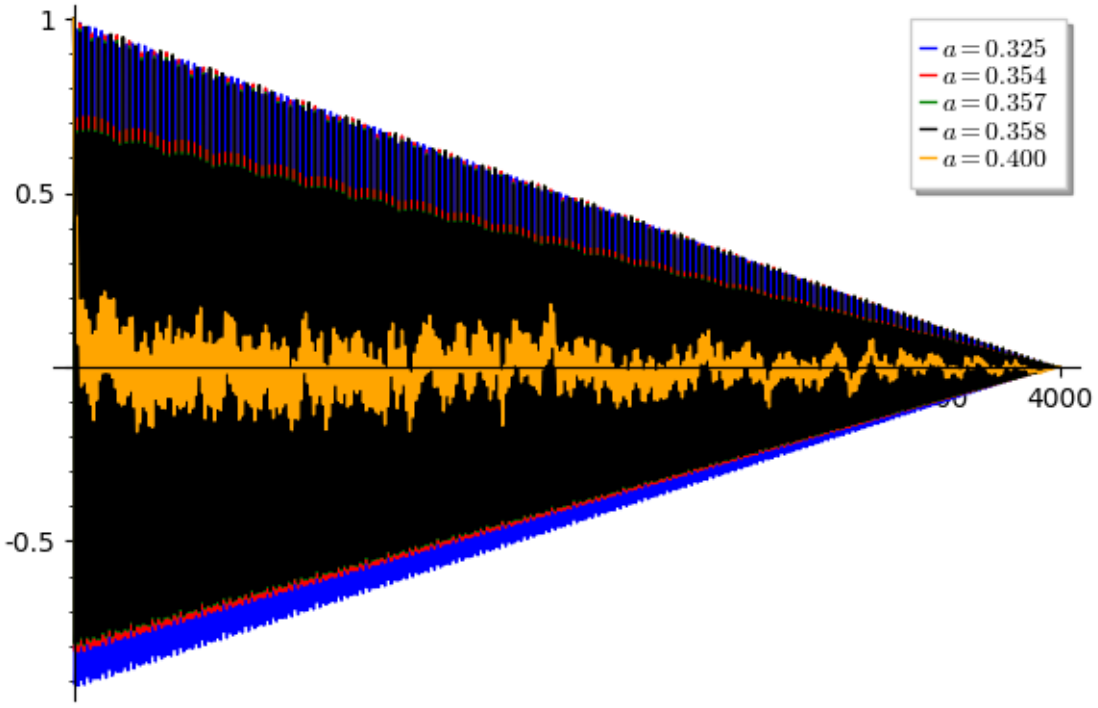

1
0.8
0.6
0.4
0.2
-0.2
5
10
15
20
25
30

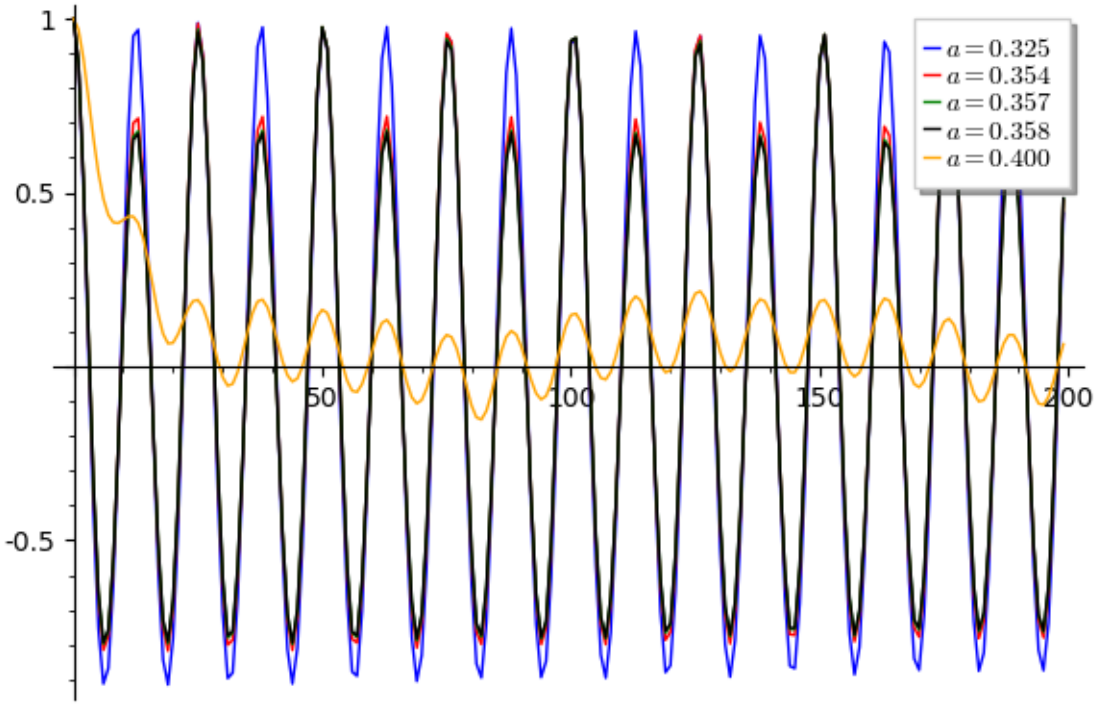
The above method can be repeated for all points close to bifurcations for the Duffing oscillator.

[9]:
```
var('x y z')
g, w0 = 0.5, 1
x0, y0, z0 = 0.1, 0.1, 0

Aval = [0.325,0.354,0.357,0.358,0.4]
p, ps = [], []
kolor = ['blue','red','green','black','orange']
j = 0
for a in Aval:
    dx = y
    dy = x - x**3 - g*y + a*cos(z)
    dz = w0

    h = 0.1
    T = 2000
    listT = srange(0,T,h, include_endpoint=True)
    num = desolve_odeint(vector([dx, dy, dz]), [x0, y0, z0], listT, [x,y,z])

    d = (num[:,0]-mean(num[:,0])).tolist()
    v = finance.TimeSeries(d)
    kor = [v.autocorrelation(i*5) for i in range(len(d)/5)]
    p.append(list_plot(kor, plotjoined=True, color=kolor[j],\
                       legend_label=r"$a = %.3f$"%a))
    ps.append(list_plot(kor[:len(kor)/20], plotjoined=True,\
                        color=kolor[j], legend_label=r"$a = %.3f$"%a))

    #list_plot(zip(d,num[:,1].tolist()),plotjoined=1,color='red').show()
    j += 1

#wykresy
im = 4
sum(p).show()#save('sage_chII012_0%d.
 ↪png'%im,axes_labels=[r'$\tau$',r'$C(\tau)$'], figsize=[8,3])
sum(ps).show()#save('sage_chII012_0%da.
 ↪png'%im,axes_labels=[r'$\tau$',r'$C(\tau)$'], figsize=[8,3])
sum(p).show()#save('sage_chII012_0%d.
 ↪pdf'%im,axes_labels=[r'$\tau$',r'$C(\tau)$'], figsize=[8,3])
sum(ps).show()#save('sage_chII012_0%da.
 ↪pdf'%im,axes_labels=[r'$\tau$',r'$C(\tau)$'], figsize=[8,3])
```


1
0.5
-0.5
4000
$a = 0.325$
$a = 0.354$
$a = 0.357$
$a = 0.358$
$a = 0.400$


1
0.5
-0.5
50
100
150
200
$a = 0.325$
$a = 0.354$
$a = 0.357$
$a = 0.358$
$a = 0.400$


1
0.5
-0.5
4000
$a = 0.325$
$a = 0.354$
$a = 0.357$
$a = 0.358$
$a = 0.400$


1
0.5
-0.5
50
100
150
200
$a = 0.325$
$a = 0.354$
$a = 0.357$
$a = 0.358$
$a = 0.400$

6.5 Odwzorowanie (cicie) Poincarego

Odwzorowanie Poincarego jest innym przedstawieniem dynamiki ukadu. Najprociej jest to wytumaczy na przykadzie oscylatora Duffinga. Jego przestrze fazowa jest 3-wymiarowa. Ruch w trzecim wymiarze jest jednostajny, $z(t) = \omega_0 t$. Rzut orbity na paszczyzn (x, y) jest przedstawiony w postaci krzywych fazowych w poprzednich czciach ksiki. Jak wida, we wszystkich przykadach krzywe fazowe na paszczynie s ograniczone na pewnym obszarze (x, y). We wszystkich rozpatrzywanych przypadach ruch wydaje sie by prawie-periodyczny: ukad cigle powraca w te same obszary. Mona zbudowa nastpujce przedstawienie tego ruchu. Okres siy periodycznej wynosi

$$T = \frac{2\pi}{\omega_0}$$

Wprowadzamy dyskretny czas

$$t_n = nT, \qquad n = 1, 2, 3, ...$$

Zapisujemy pooenie i prdko czstki w dyskretnych chwilach czasu:

$$x_n = x(t_n), \qquad y_n = y(t_n), \qquad x(0) = x_0, \qquad y(0) = y_0$$

Wspórzdne tych punktów nanosimy na paszczyzn. Otrzymujemy odwzorowanie które nazywamy odwzorowaniem Poincarego. Obrazowo mówic mona w 3-wymiarowej przestrzeni fazowej wprowadzi paszczyzn, tak aby nigdzie nie bya styczna do trajektorii i bya transwersalna do trajektorii (cilej mówic do potoku fazowego), czyli aby trajektoria przecinaa paszczyzn, a nie bya równolega do niej (nie omijaa j).

Odwzorowanie Poincarego to przyporzdkowanie:

$$x_{n+1} = \mathcal{G}(x_n)$$

Jawna konstrukcja tego odwzorowania z wyjciowego ukadu równa róniczkowych jest moliwa tylko w bardzo specjalnych przypadkach. W przypadku oscylatora Duffinga, nie mona otrzyma jawnej postaci tego odwzorowania. Jedynie uycie komputera pozwala na graficzne przedstawienie funkcji $\mathcal{G}$. Jakie wnioski pyn z takiego przedstawienia. 1. Gdyby trajektoria bya krzyw zamknit w ksztacie elipsy (atraktor o okresie 1) to na ciciu Poincarego otrzymalibymy 1 punkt:

2. Gdyby trajektoria bya atraktorem o okresie 2 to na ciciu Poincarego otrzymalibymy 2 punkty:
3. Gdyby trajektoria bya chaotyczna, to za kadym razem przebiega przez inne punkty paszczyzny i tworzy zbiór skadajcy sie z nieskoczenie wielu punktów. Poniej pokazano takie odwzorowanie dla oscylatora Duffinga.

Jeeli jestemy w stanie zbudowa graficznie przedstawienie Poincarego danego ukadu dynamicznego z cigym czasem, wówczas moemy rozpozna takie reimy które s "podejrzane" o wasnoci chaotyczne. Numerycznie nie powinno nastrcza to wikszych problemów. Jeeli znamy ω_0 bd okres powrotu do obliczenia cicia to wystarczy wykorzysta poniszy kod Sage. Zwracamy jedynie uwag na to, e odpowiednio "gsty" obraz uzyskamy dla bardzo dugich przebiegów (duych T).

6.6 Poincarego map (section)

The Poincare map is another representation of dynamics of the system. Poincaré maps are used to investigate periodic or quasi-periodic dynamical systems. We can explain it for our 3-dimensional

system describing the Duffing oscillator. In 3-dimensional space we should choose a plane in it but in a suitable way. For periodic or quasiperiodic motion, the trajectory of the particle will intersect the plane. After same time, it again intersects the plane in the same point or not. And it will be repeated after some time. We can formulate it in the following way: Let period of the periodic force is

$$T = \frac{2\pi}{\omega_0}$$

We introduce a discrete time

$$t_n = nT, \qquad n = 1, 2, 3, ...$$

We record the position and velocity of the particle at discrete times:

$$x_n = x(t_n), \qquad y_n = y(t_n), \qquad x(0) = x_0, \qquad y(0) = y_0$$

We put the coordinates of these points on the plane. We get the mapping that we call Poincare mapping. Figuratively speaking, you can enter a plane in the 3-dimensional phase space so that it is nowhere tangent to the trajectory and is transversal to the trajectory (more specifically to the phase flow), so that the trajectory intersects the plane and is not parallel to it (does not omit it) .

Poincare's mapping is the assignment:

$$x_{n+1} = \mathcal{G}(x_n)$$

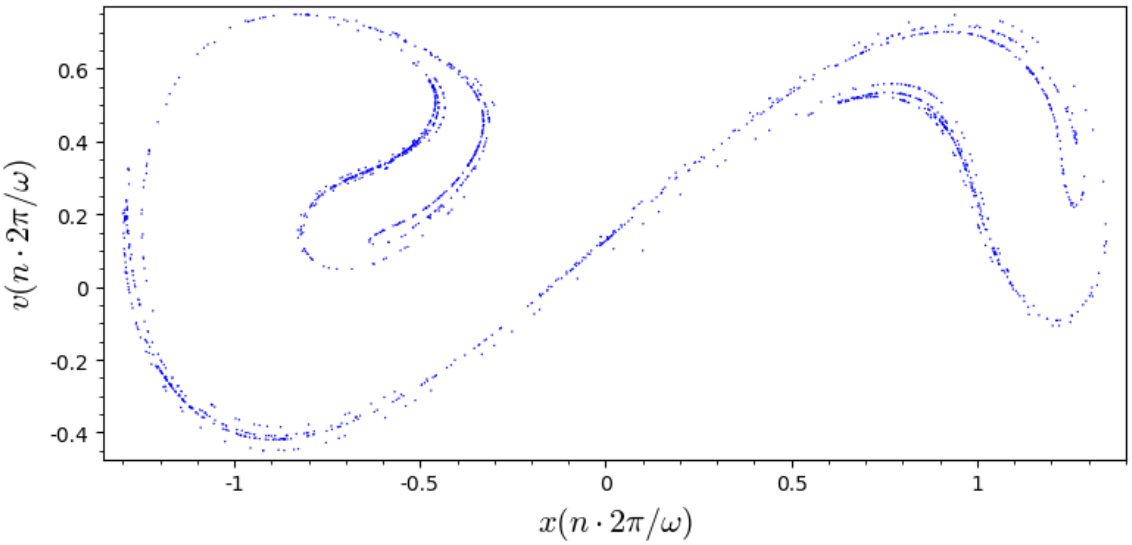
The explicit construction of this mapping from the initial system of differential equations is possible only in very special cases. In the case of the Duffing oscillator, it is not possible to obtain an explicit form of this mapping. Only the use of a computer allows graphic representation of the $\mathcal{G}$ function. What conclusions come from such a presentation.

1. If the trajectory was a curve in the shape of an ellipse (attractor of period 1), we would receive 1 point on a Poincare section.
2. If the trajectory is an attractor of period 2, we would observe 2 points on the Poincare section.
3. If the trajectory was chaotic, then each time it runs through other points of the plane and creates a set consisting of infinitely many points. The mapping for the Duffing oscillator is shown below.

If we are able to construct Poincare's representation of a given dynamic system with continuous time, then we can recognize regimes that are "suspicious" of chaotic properties. Numerically, it should not cause any serious problems. If we know ω_0 or the return period to calculate the cut, just use the Sage code below. We only pay attention to the fact that a "dense" image is obtained for very long runs.

[10]:
```
reset()
var('x y z')

# parametry ukadu równa róniczkowych
```

```
a, g = 0.3, 0.26

# czstotliwo (do obliczania cicia Poincarego)
w0 = 1

# wartoci pocztkowe
x0, y0, z0 = 0.1, 0.1, 0

#ukad równa róniczkowych
dx = y
dy = x - x**3 - g*y + a*cos(z)
dz = w0


#krok co jaki wypenia si ma nasza lista
#rozwiza ustawiamy równy okresowi
h = 2.0*pi/w0

###
#symulacje
###
T = 10000
listT = srange(0,T,float(h), include_endpoint=True)
sol = desolve_odeint(vector([dx, dy, dz]), [x0, y0, z0], listT, [x,y,z])

#i sam rysunek cicia
p = points(zip(sol[:,0],sol[:,1]), figsize=(8,4), axes_labels=["$x(n\cdot2 \pi/
 ↪\omega)$","$v(n\cdot2 \pi/\omega)$"], frame=1, axes=0, size=1)

im=5
#p.save('sage_chII012_0%d.pdf'%im)
#p.save('sage_chII012_0%d.png'%im)


p.show()
```



6.7 Examples of chaos in Nature

We have to distinguish chaotic processes from random processes. Chaotic processes are deterministic, and stochastic processes are random processes. Chaotic processes are investigated by mathematicians, physicists, chemists, biologists, sociologists, meteorologists, astrophysicists, information theory and neuroscience. In all these branches of science, there are deterministic models exhibiting chaotic properties. Thousands of works on chaotic systems have been published since the 1960s. Mathematicians say that almost all dynamical systems are chaotic, and only a small part of all systems do not show this property. Mathematicians argue that the phase space of the system modeled by the autonomous system of differential equations must be at least 3-dimensional in order to observe chaotic behabior. For discrete systems, there are no such limitations: one recursive equation $x_{n+1} = f(x_n)$ can also show chaotic properties.

Below are some examples of real phenomena showing chaotic properties.

1. Dynamics of liquid and turbulence
2. Lasers
3. Electronic systems
4. Plasma
5. Chemical reactions

On the Wikipedia's website with Chaos Theory, you can find other examples and basic works on the subject. At the end of this part, we must mention the man who started it all in 1961. It was Edward Lorenz, a mathematician and American meteorologist who analyzed one of the simplest models to predict weather. It is with his name associated the "butterfly effect" illustrating the extraordinary sensitivity of dynamics to initial conditions: can the butterfly movement in Brazil cause a tornado

in Texas. In this pictorial saying, the essence of chaos is contained: The butterfly disturbs the air movement locally through its flight. This disturbed air movement increases and causes more and more large weather changes, radically changes the "trajectory" leading to a tornado that will appear over Texas. Can a butterfly actually be so dangerous?