Lodz University of Technology
Department of Automation, Biomechanics and Mechatronics

Application of Matlab and Mathematica environments to visualize the operation of dynamic systems

Dariusz Grzelczyk
dariusz.grzelczyk@p.lodz.pl

HR EXCELLENCE IN RESEARCH

ECIU university

Plan of presentation

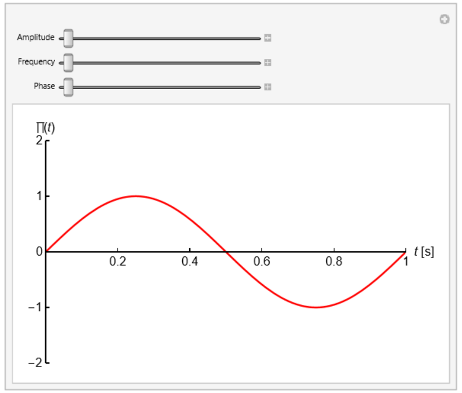
- Motivation
- Basic information about Matlab and Mathematica environments
- Animations in Matlab – fundamental examples
- Model creations in Matlab – fundamental examples
- Connecting NI USB-6002 card to Matlab
- Animations in Mathematica – fundamental examples
- Model creations in Mathematica –examples
- Conclusions

What animation mean?

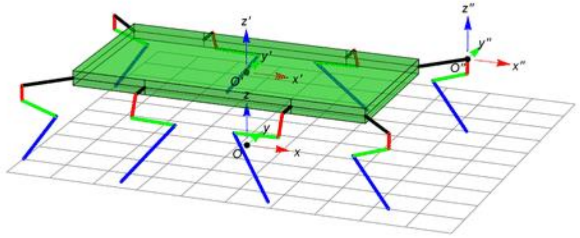
The first fundamental question in this presentation is: What does animation mean?

ANIMATION:

- a method to visualize data changing over time



- a method of manipulating a figure/object to appear as a moving image



Advantages of animation

- Easy interpretation and better understanding of the presented data
- Presentations are more attractive and attract attention of the listeners
- Using as a supplementary material at submission of the online version of a published research article

Supplementary Multimedia Data in your Article!

Did you know that Elsevier journals accept electronic supplementary material to support and enhance your research?

Supplementary files offer authors additional possibilities to publish supporting applications, movies, animation sequences, high-resolution images, background datasets, sound clips and more. Supplementary files supplied will be published online alongside the electronic version of your article in Elsevier Web products, including ScienceDirect: http://www.sciencedirect.com.

Animation - software

There are many different free and commercial software for creating animations.

Mechanical Engineering problems from mathematical point of view:

- equation or system of equations,
- linear or nonlinear equations,
- stationary or non-stationary equations,
- ordinary or partial differential equations,
- differential, integral or differential/integral equations.

Computer Algebra System (CAS) Symbolic Algebra System (SAS)

any mathematical software with the ability to manipulate mathematical expressions in a way similar to the traditional manual computations of mathematicians and scientists

- Commercial software:

Maple™ Mathcad 15.0 MATLAB Wolfram Mathematica

- Open source software:

Maxima Programming Language GNU Octave Scilab

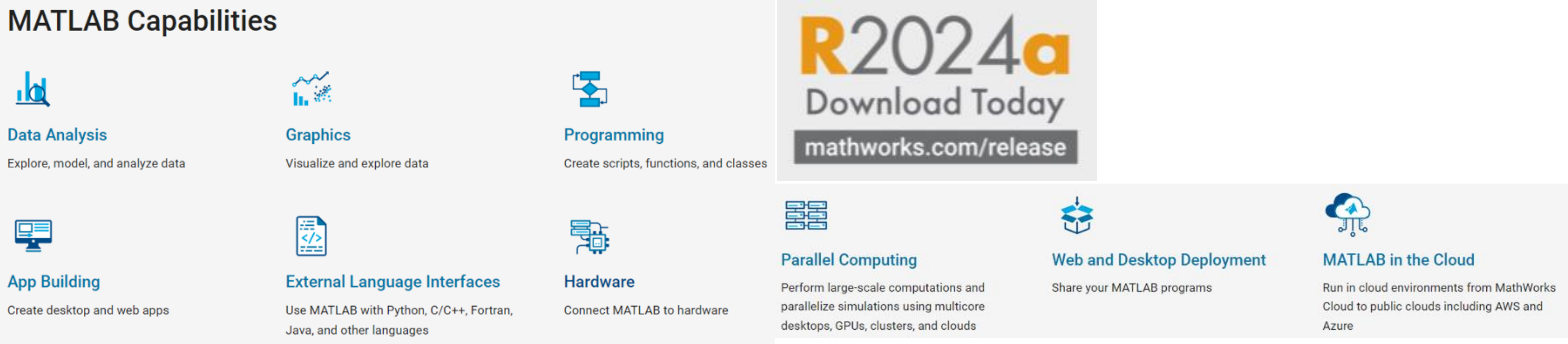
Matlab

„programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models”

- First version of MATLAB was completed in the late 1970s. The software was presented to the public in February 1979 at the Naval Postgraduate School, California. Early versions of MATLAB were simple matrix calculators with 71 pre-built functions. The first Matlab was not a programming language; it was a simple interactive matrix calculator; no programs, no toolboxes, no graphics, no ODEs or FFTs.

- PC-MATLAB was first released as a commercial product in 1984 at the Automatic Control Conference in Las Vegas.

- Major Updates of R2024a Release: Computer Vision Toolbox; Deep Learning Toolbox; GPU Coder; Instrument Control Toolbox; Satellite Communications Toolbox; UAV Toolbox.



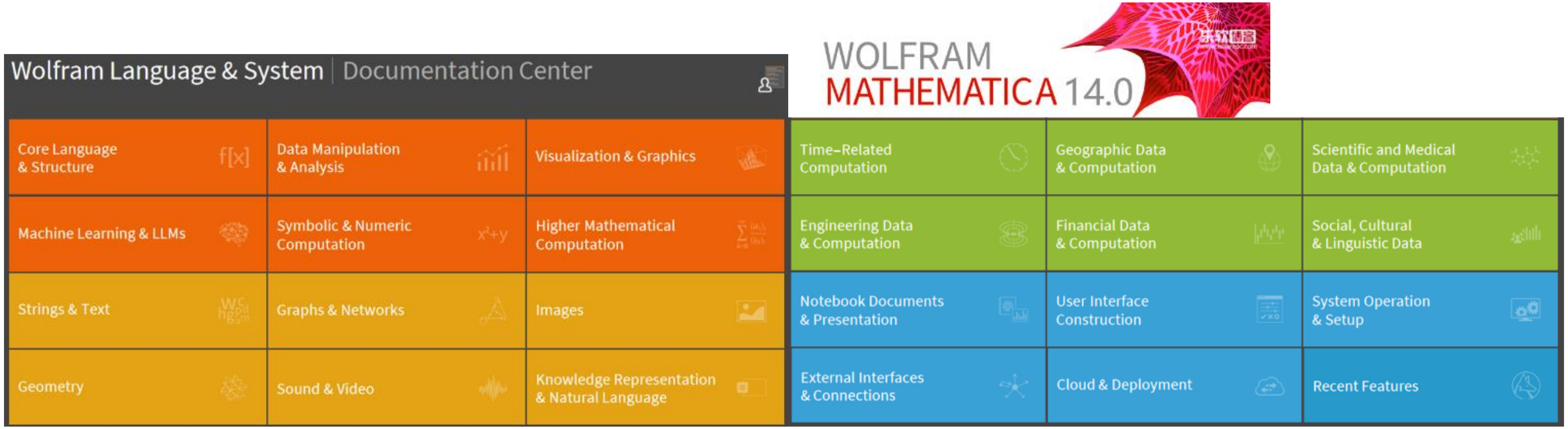
Mathematica

„high-powered computation with thousands of Wolfram Language functions, natural language input, real-world data, mobile suport"

- Wolfram Mathematica was started by Stephen Wolfram, and developed by Wolfram Research of Champaign, Illinois. The Wolfram Language is the programming language used in Mathematica.
- Mathematica 1.0 was released on June 23, 1988 in Champaign, Illinois and Santa Clara, California. Mathematica 1.0 had a set of 554 built-in functions.
- Today Mathematica has over six thousands commands and functions.

Mathematica 1.0 | June 1988

- Initial release of Mathematica



Animation in Matlab

Animation

Animating plots

Create animations to visualize data changing over time. Display changing data in real time or record a movie or GIF to replay later.

Lorenz system in Matlab

The Lorenz system is a system of ordinary differential equations first studied by mathematician and meteorologist Edward Lorenz in the 1960s. It has been proposed as approximation of Navier-Stokes equations.

MATLAB

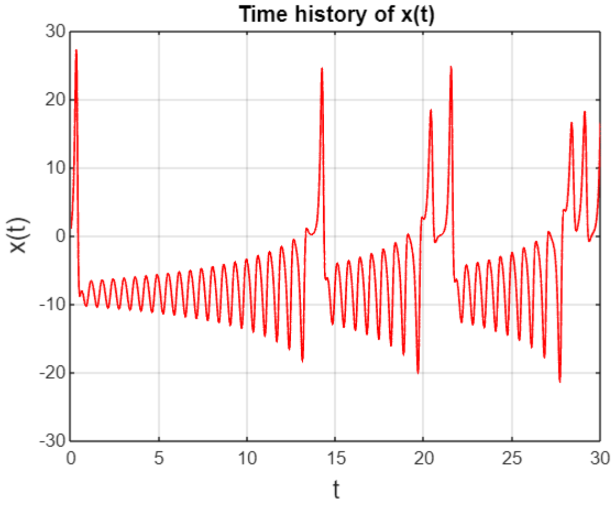
Variables: x, y, z; Parameters: $\sigma = 10, b = \frac{8}{3}, \rho = 28$ (chaotic behaviour)

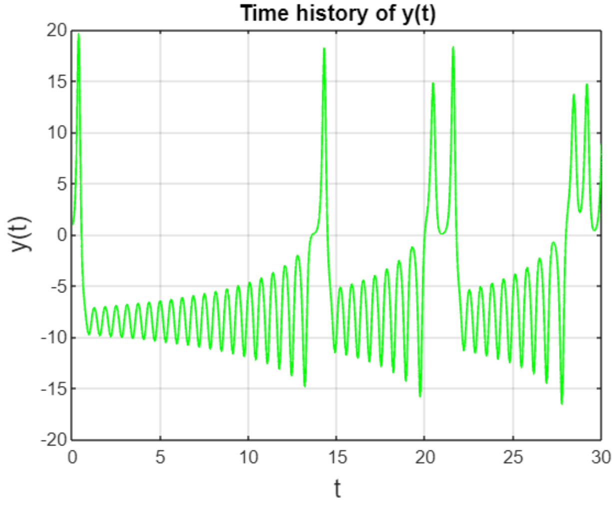
$$\text{ODEs:} \begin{cases} \frac{dx}{dt} = \sigma(y - x) \\ \frac{dy}{dt} = (\rho - z)x - y \\ \frac{dz}{dt} = xy - \beta z \end{cases}$$

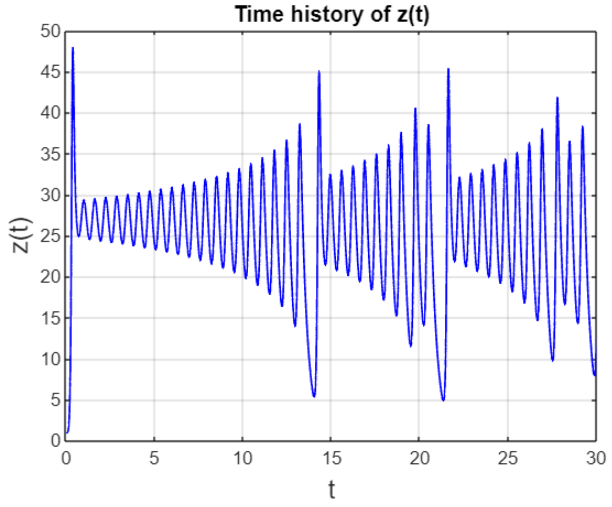
$$\text{Critical points:} \begin{cases} (0,0,0) \\ \left(\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho - 1\right) \\ \left(-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho - 1\right) \end{cases}$$

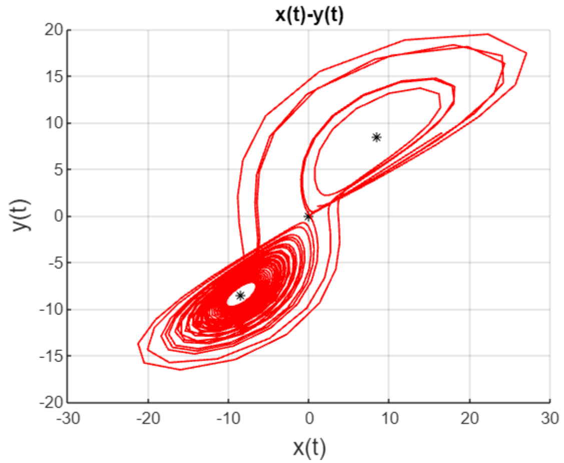
```
clear; syms t x(t) y(t) z(t)
syms sigma beta rho
t0=0;tf=30; sigma=10;beta=8/3;rho=28;
eq0=[0,0,0];
eq1=[sqrt(beta*(rho-1)),sqrt(beta*(rho-1)),rho-1];
eq2=[-sqrt(beta*(rho-1)),-sqrt(beta*(rho-1)),rho-1];
x0=1;y0=1;z0=1;
eqn1=diff(x(t),t)==sigma*(y(t)-x(t));
eqn2=diff(y(t),t)==(rho-z(t))*x(t)-y(t);
eqn3=diff(z(t),t)==x(t)*y(t)-beta*z(t);
eqns=[eqn1,eqn2,eqn3];
[V]=odeToVectorField(eqns);
M=matlabFunction(V,'vars',{'t','Y'});
sol=ode45(M,[t0,tf],[x0,y0,z0]);
```

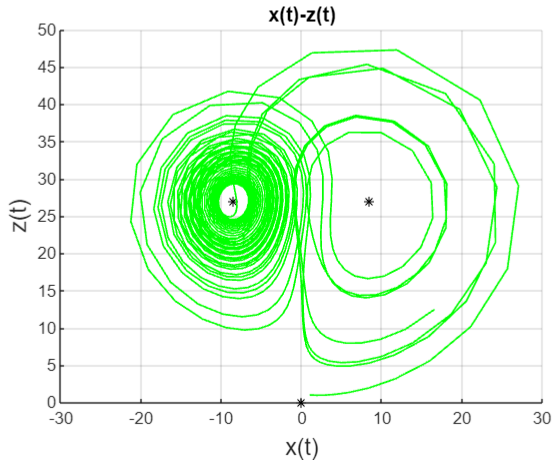
Lorenz system in Matlab

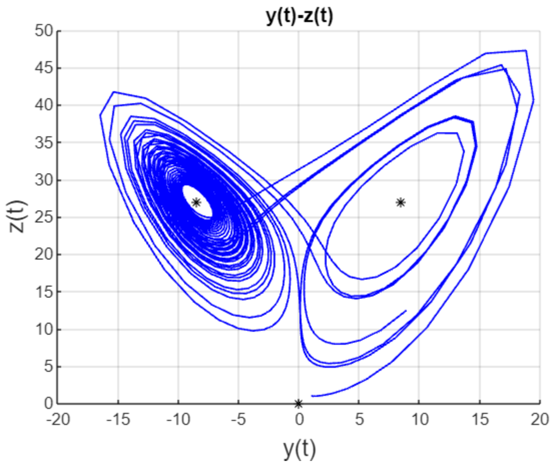


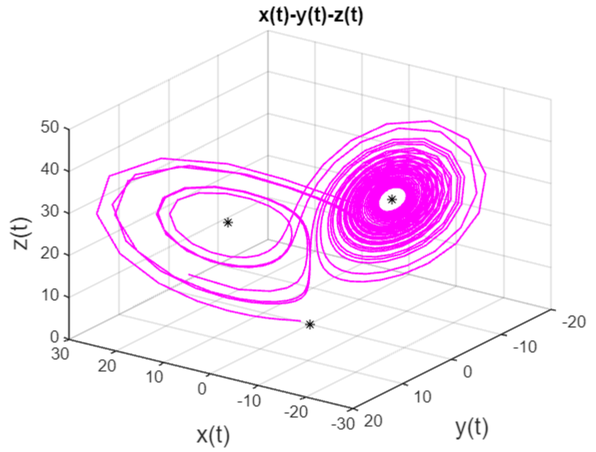












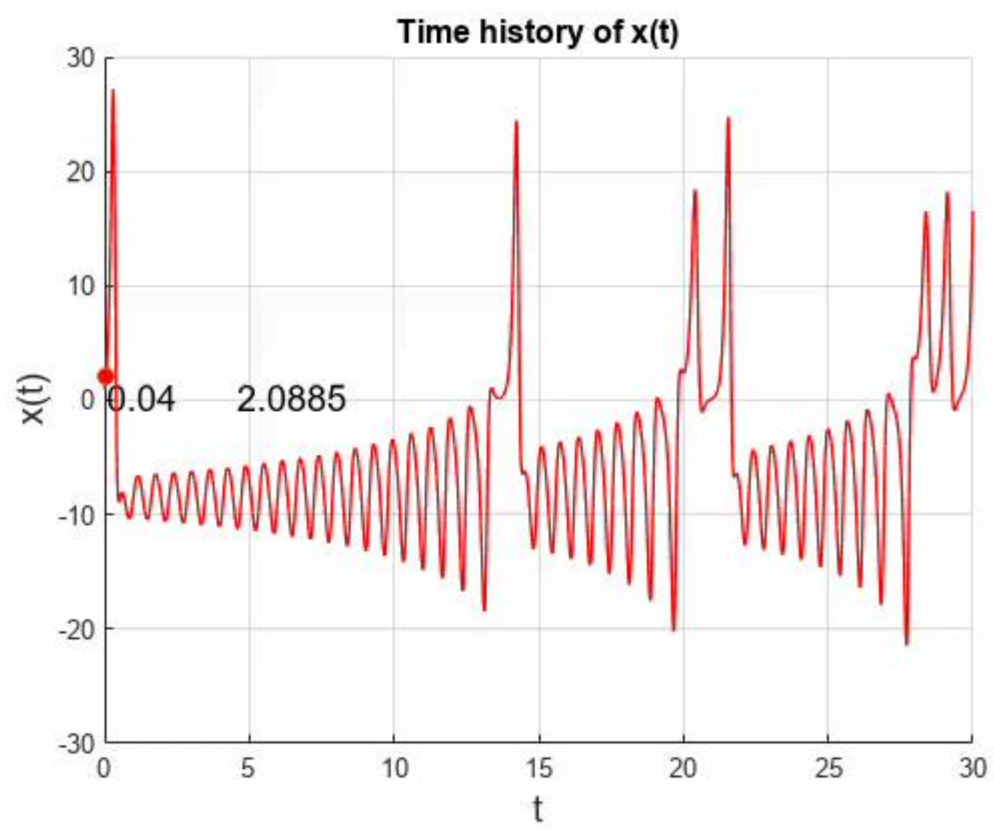
Trace Marker Along Line

This example shows how to trace a marker along a line by updating the data properties of the marker.
Move the marker along the line by updating the XData and YData properties in a loop.
num2str Convert numbers to character array

```
dt=0.04; T=t0:dt:tf
X=deval(sol,T,1)
Y=deval(sol,T,2)
Z=deval(sol,T,3)
```

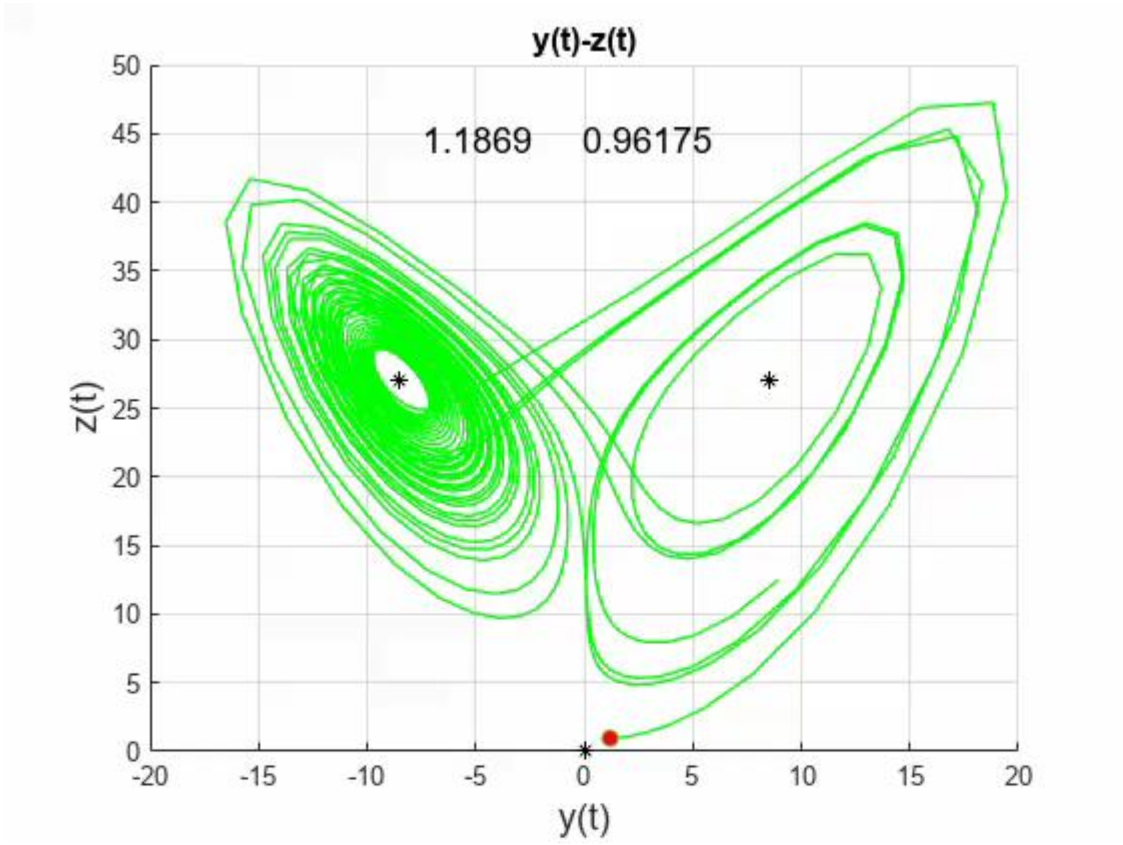
```
T = 1×751
          0    0.0400    0.0800    0.1200    0.1600    0

X = 1×751
     1.0000    2.0885    3.5177    5.6507    8.9094    13

                                   2.6634    4.1840    6

                                   1.2922    2.0994    4
```

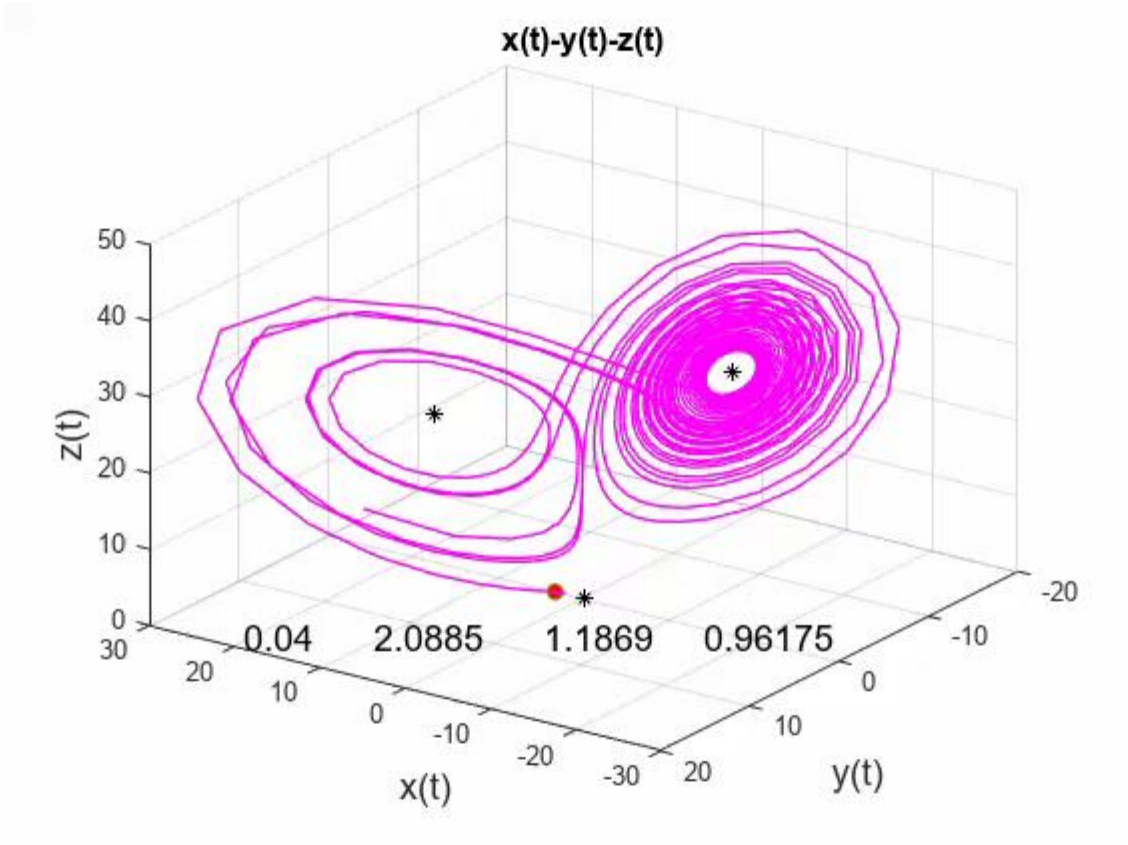
```
figure; h1 = hgtransform('Parent',gca); hold on
fplot(@(x)deval(sol,x,1),[t0 tf],'LineWidth',1,'Color','r')
p1=plot(T(1),X(1),'o','MarkerFaceColor','red');
plot(T(1),X(1),'o','Parent',h1);
hold off
txt1 = text(T(1),X(1),num2str([T(1),X(1)]),'Parent',h1,...
'VerticalAlignment','top','FontSize',14);
xlabel('t','FontSize',14), ylabel('x(t)','FontSize',14)
title('Time history of x(t)','FontSize',12), grid on
xlim([t0 tf]), ylim([-30 30])
for k=2:length(T)
p1.XData=T(k); p1.YData=X(k);
m1=makehgtform('translate',T(k)-T(1),X(k)-X(1),0);
h1.Matrix=m1; txt1.String=num2str([T(k),X(k)]);
drawnow, pause(0.01)
end
```



Trace Marker Along Line

This example shows how to trace a marker along a line by updating the data properties of the marker.

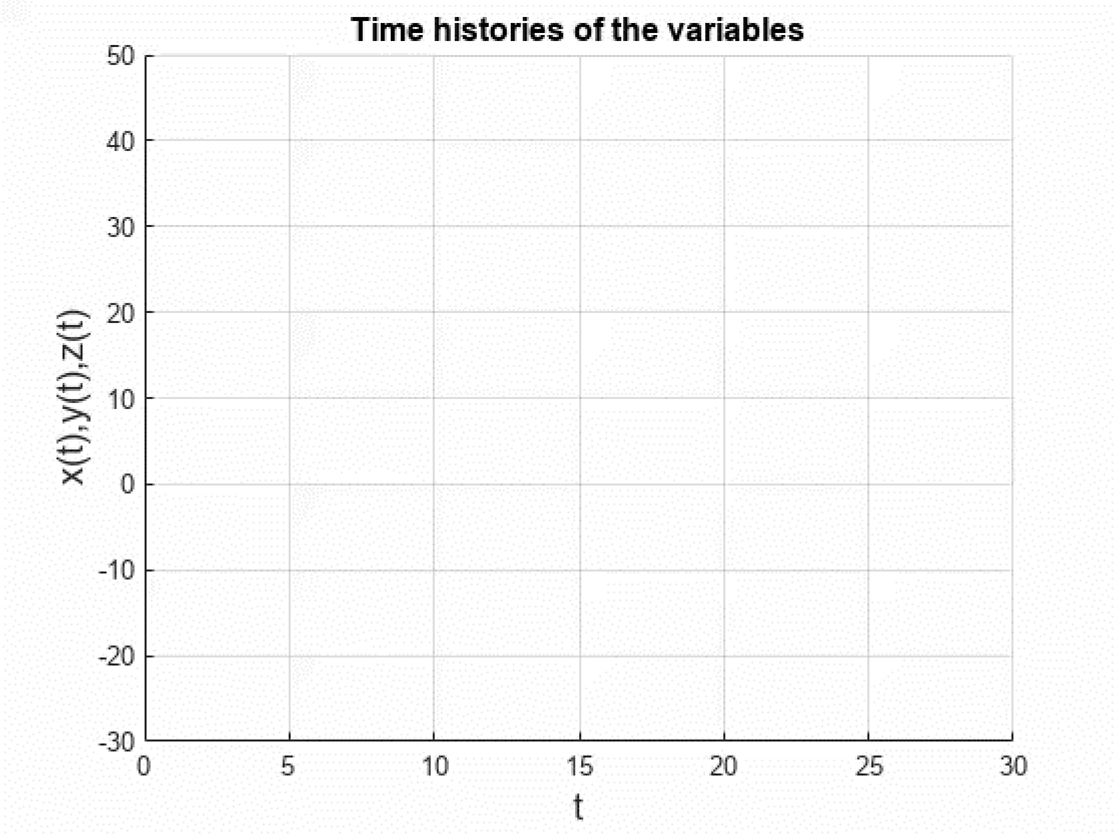




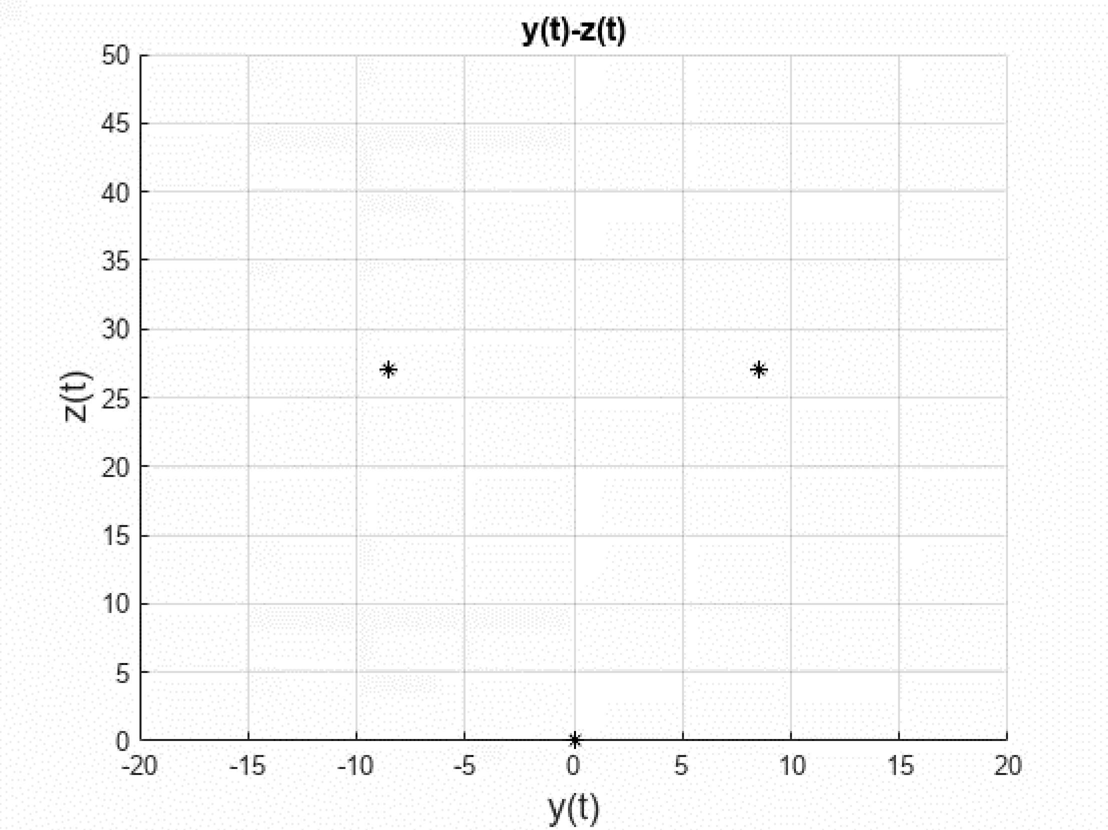
Line Animations

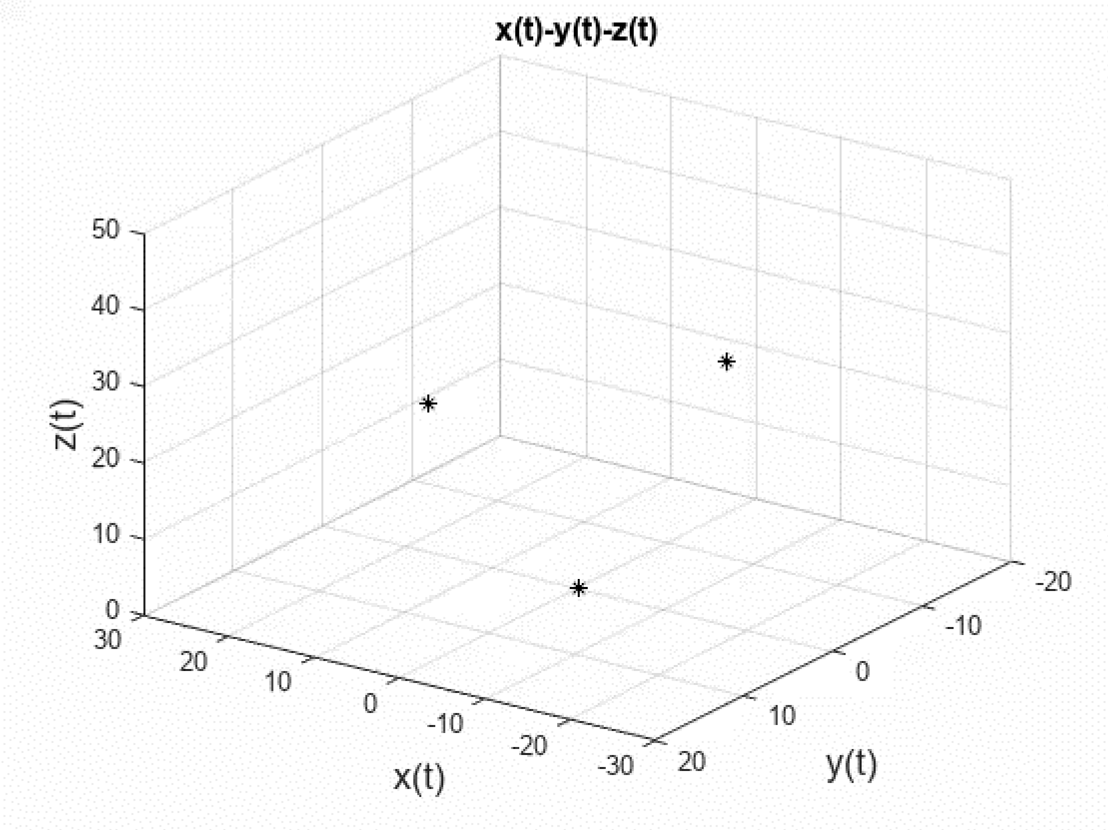
Animation of three growing lines of different colors. The animatedline function allows to add new points (addpoints) to a line without redefining existing points. Use a drawnow or drawnow limitrate command to display the updates on the screen after adding the new points.

```
figure;
a1=animatedline('Color',[1 0 0]);
a2=animatedline('Color',[0 1 0]);
a3=animatedline('Color',[0 0 1]);
xlabel('t','FontSize',14), ylabel('x(t),y(t),z(t)','FontSize',14)
title('Time histories of the variables','FontSize',12), grid on
axis([t0 tf -30 50])
for k=1:length(T)
% first line
tk = T(k); xk = X(k); addpoints(a1,tk,xk);
% second line
tk = T(k); yk = Y(k); addpoints(a2,tk,yk);
% third line
tk = T(k); zk = Z(k); addpoints(a3,tk,zk);
pause(0.05); drawnow limitrate
end
```



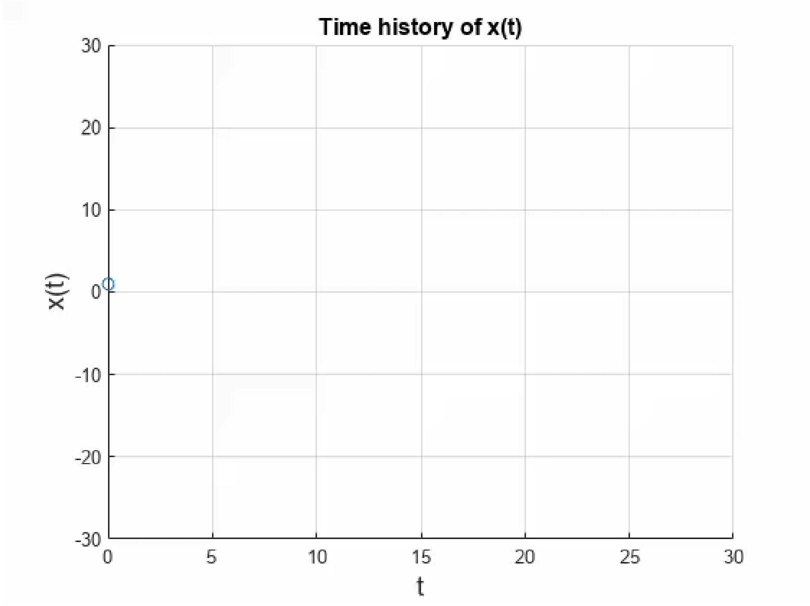
Line Animations



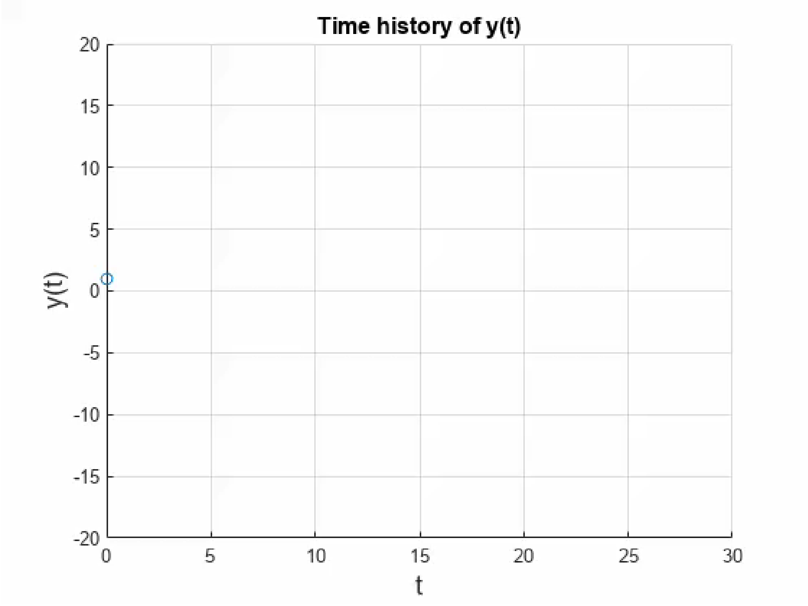


comet

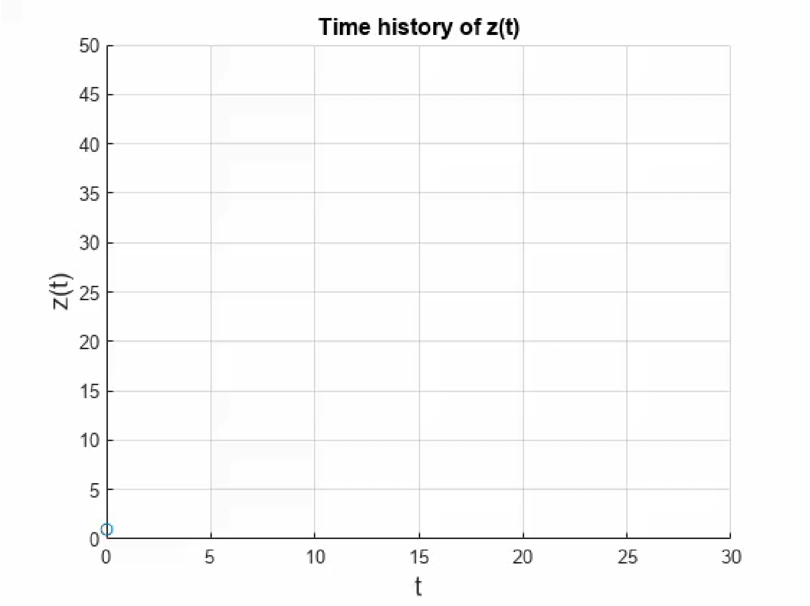
comet(T,X,0.05)



comet(T,Y,0.2)

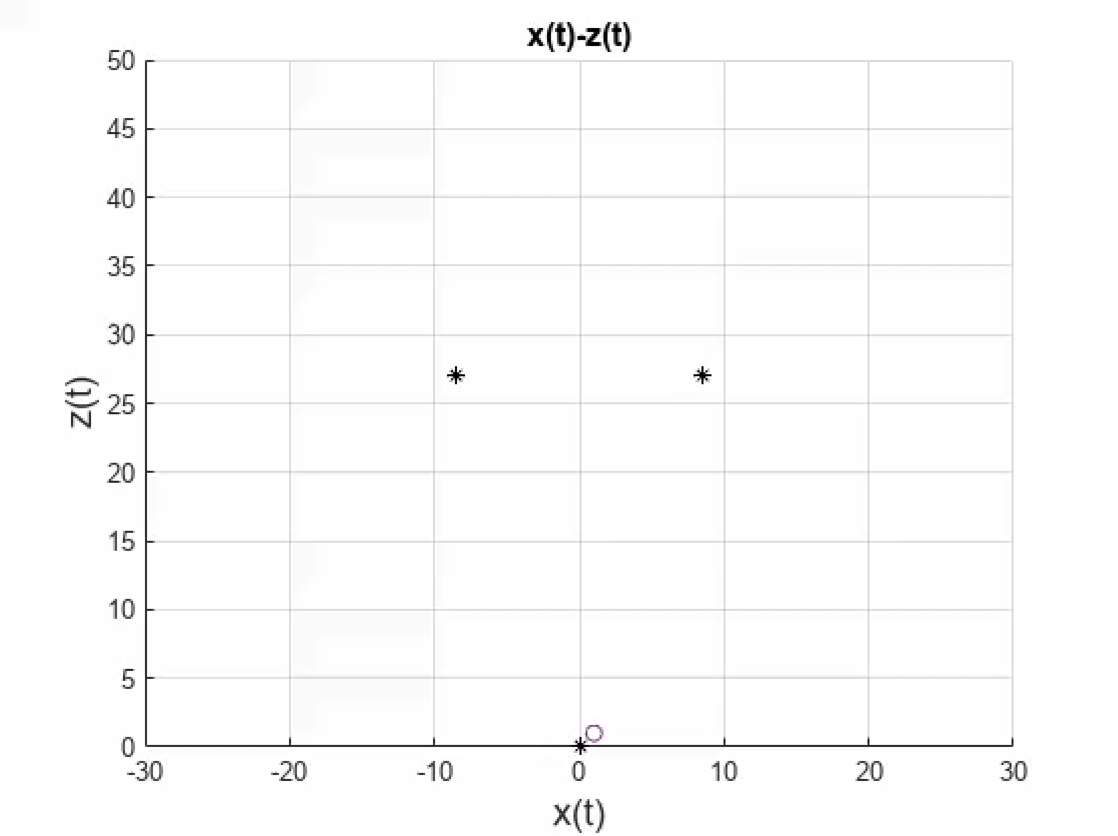


comet(T,Z,0.6)

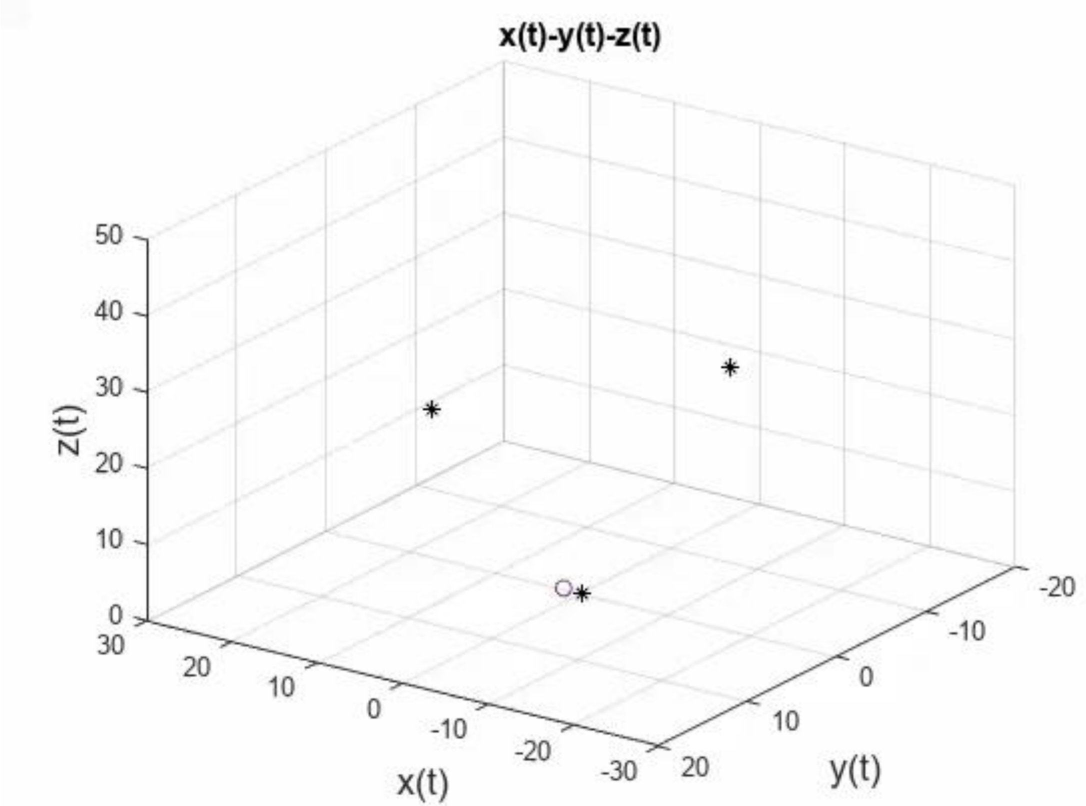


comet3

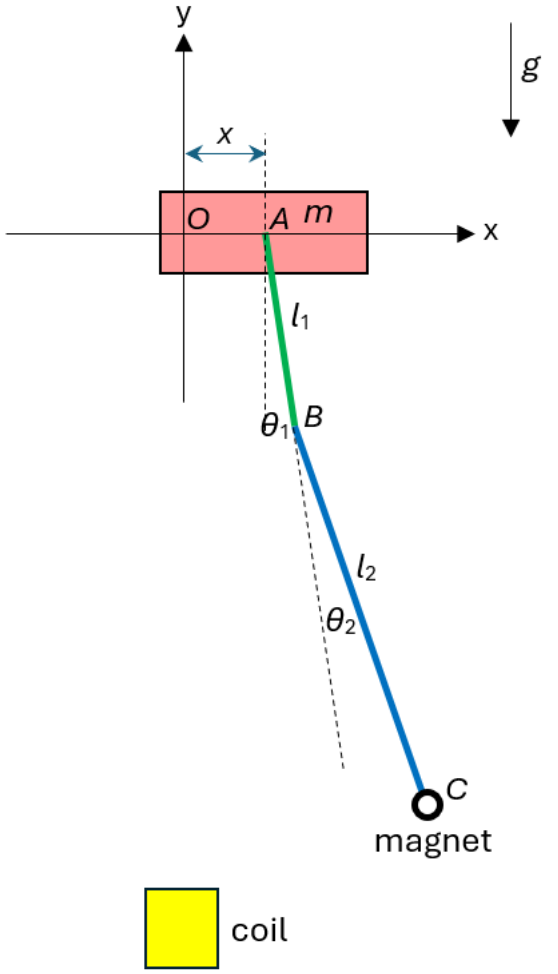
comet(X,Z)



comet3(X,Y,Z)

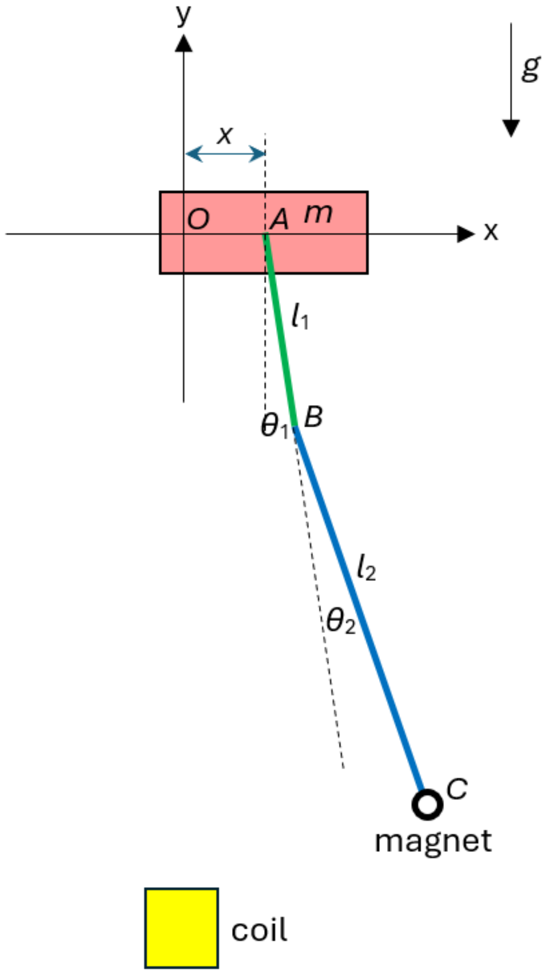


Example - 3-DOF system

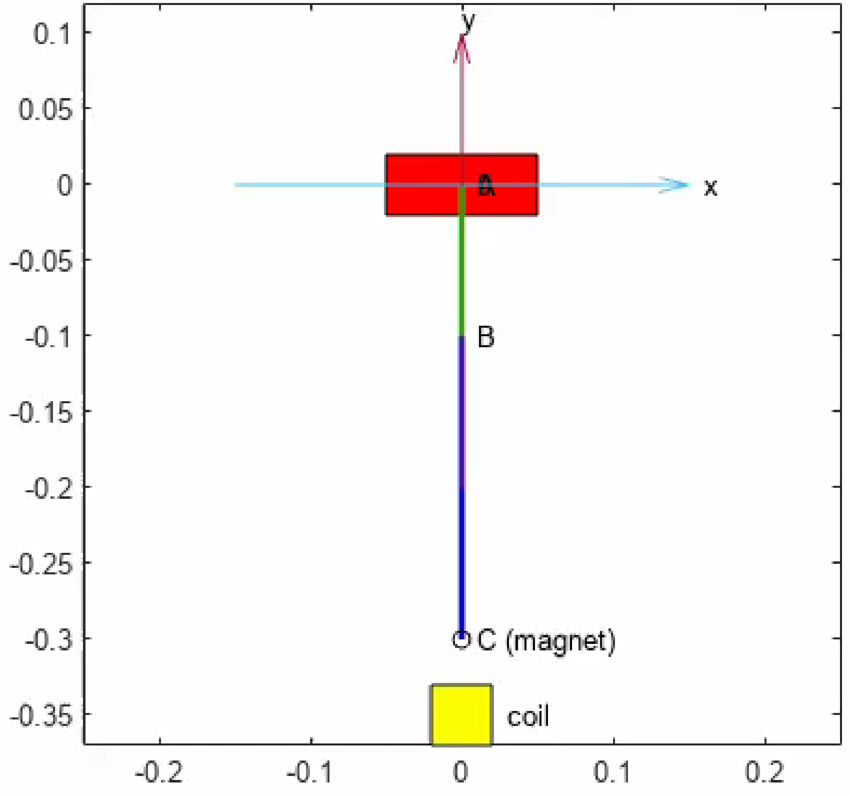



```
close all; clear all; clc;
t0=0; tf=30; n=1000; t=linspace(t0,tf,n);
l1=0.1;l2=0.2; omega=2; x0=0.05;theta10=0.19;theta20=0.22;
Ax=x0*sin(omega*t); Ay=0*cos(omega*t);
theta1=theta10*sin(omega*t); theta2=theta20*sin(omega*t);
Bx=Ax+l1*sin(theta1); By=Ay-l1*cos(theta1);
Cx=Ax+l1*sin(theta1)+l2*sin(theta1+theta2);
Cy=Ay-l1*cos(theta1)-l2*cos(theta1+theta2);
drawArrow = @(x,y) quiver(x(1),y(1),x(2)-x(1),y(2)-y(1),0);
dx=0.1; dy=0.04; axislimits=[-0.25 0.25 -0.37 0.12];
```

Example - 3-DOF system




```
for k=1:n
 plot([Ax],[Ay],'--',0,0,'ko');
 axis equal; hold on; axis(axislimits);
 sliderX=[Ax(k)-dx/2,Ax(k)-dx/2,Ax(k)+dx/2
 sliderY=[Ay(k)-dy/2,Ay(k)+dy/2,Ay(k)+dy/2
 coilX=[0-0.02,0-0.02,0+0.02,0+0.02,0-0.02
 coilY=[-0.35-0.02,-0.35+0.02,-0.35+0.02,-0
 fill(sliderX,sliderY,'r'); fill(coilX,coilY,'y');
 plot([Ax(k),Bx(k)],[Ay(k),By(k)],'g','LineWid
 plot([Bx(k),Cx(k)],[By(k),Cy(k)],'b','LineWid
 text(0.15+0.01,0,'x'); text(0,0.1+0.01,'y'); t
 text(Ax(k)+0.01,Ay(k),'A'); text(Bx(k)+0.01,
 text(Cx(k)+0.01,Cy(k),'C (magnet)'); text(0.
 x1 = [-0.15 0.15]; y1 = [0 0]; drawArrow(x1
 x2 = [0 0]; y2 = [-0.2 0.1]; drawArrow(x2,y2
 hold off; pause(0.01);
end
```



NI USB-6002 with Matlab

PC with MATLAB environment

NI USB-6002

=> NI-DAQmx Support from Data Acquisition Toolbox

```
% Display a List of Available Devices
% Use daqlist to display a list of devices
% available to your machine and MATLAB.
d = daqlist("ni")
```

d = 1×4 *table*

	DeviceID	Description	Model	DeviceInfo
1	"Dev1"	"National Instruments(TM) USB-6002"	"USB-6002"	1×1 DeviceInfo

```
% To obtain more information about a particular device,
% view the "DeviceInfo" table cell for it.
deviceInfo = d(1, "DeviceInfo")
```

deviceInfo = 1×1 *table*

	DeviceInfo
1	1×1 DeviceInfo

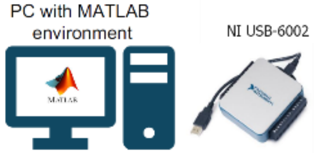
```
% daq function create DataAcquisition device interface for National Instruments
dq = daq("ni")
```

```
dq =
DataAcquisition using National Instruments(TM) hardware:

                  Running: 0
                     Rate: 1000
        NumScansAvailable: 0
         NumScansAcquired: 0
           NumScansQueued: 0
   NumScansOutputByHardware: 0
                RateLimit: []

Show channels
Show properties and methods
```

```
% Add input channel(s) to device interface
addinput(dq, "Dev1", "ai0", "Voltage"), addinput(dq, "Dev1", "ai1", "Voltage")
dq.Rate = 100 % set numbers of samples per second
time_of_measurement = 10.0 % set total time of data measurement (collection)
```

```
dq =
DataAcquisition using National Instruments(TM) hardware:

                  Running: 0
                     Rate: 100
        NumScansAvailable: 0
         NumScansAcquired: 0
           NumScansQueued: 0
   NumScansOutputByHardware: 0
                RateLimit: [0.1000 25000]

Show channels
Show properties and methods

time_of_measurement = 10
```

NI USB-6002 with Matlab

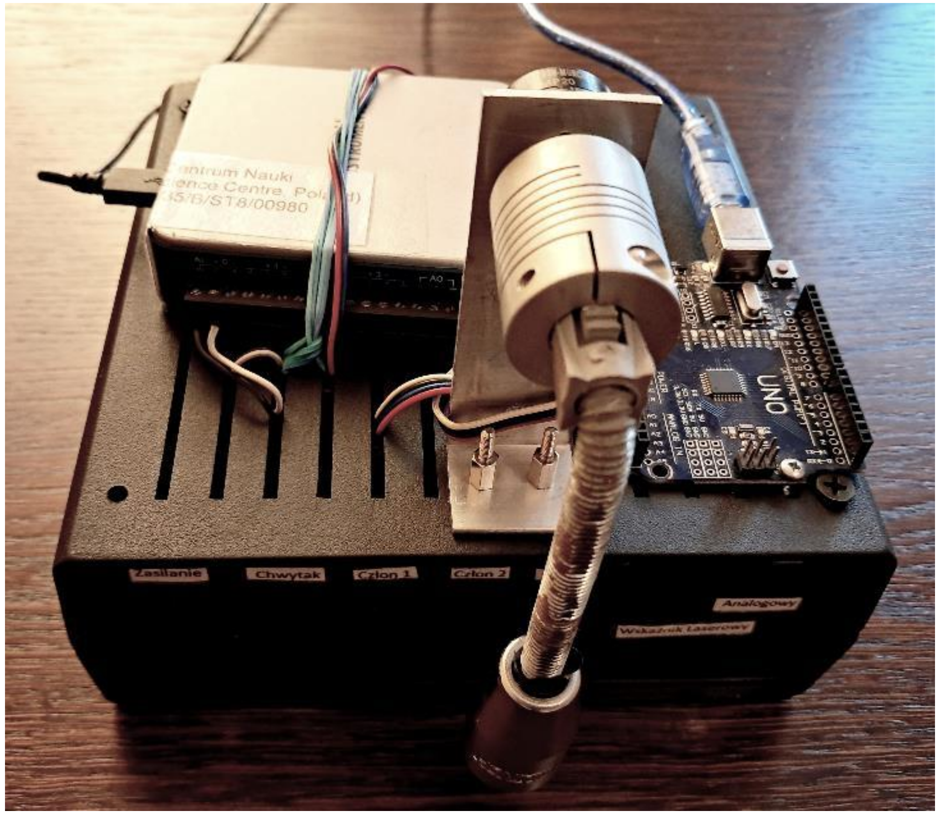


Example – damped physical pendulum

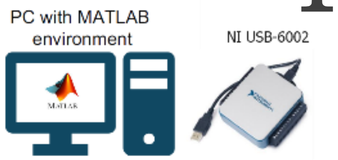
```
%collecting measurement data
data=read(dq,seconds(time_of_measurement))
```

data = 1000×2 timetable

	Time	Dev1_ai0	Dev1_ai1
1	0 sec	-0.2657	-0.2654
2	0.01 sec	-0.2664	-0.2664
3	0.02 sec	-0.2661	-0.2693
4	0.03 sec	-0.2683	-0.2673
5	0.04 sec	-0.2677	-0.2706
6	0.05 sec	-0.2690	-0.2686
7	0.06 sec	-0.2680	-0.2715
8	0.07 sec	-0.2706	-0.2699
9	0.08 sec	-0.2690	-0.2719



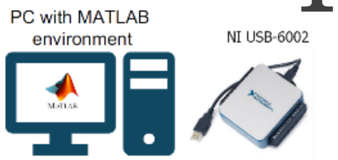
Damped physical pendulum

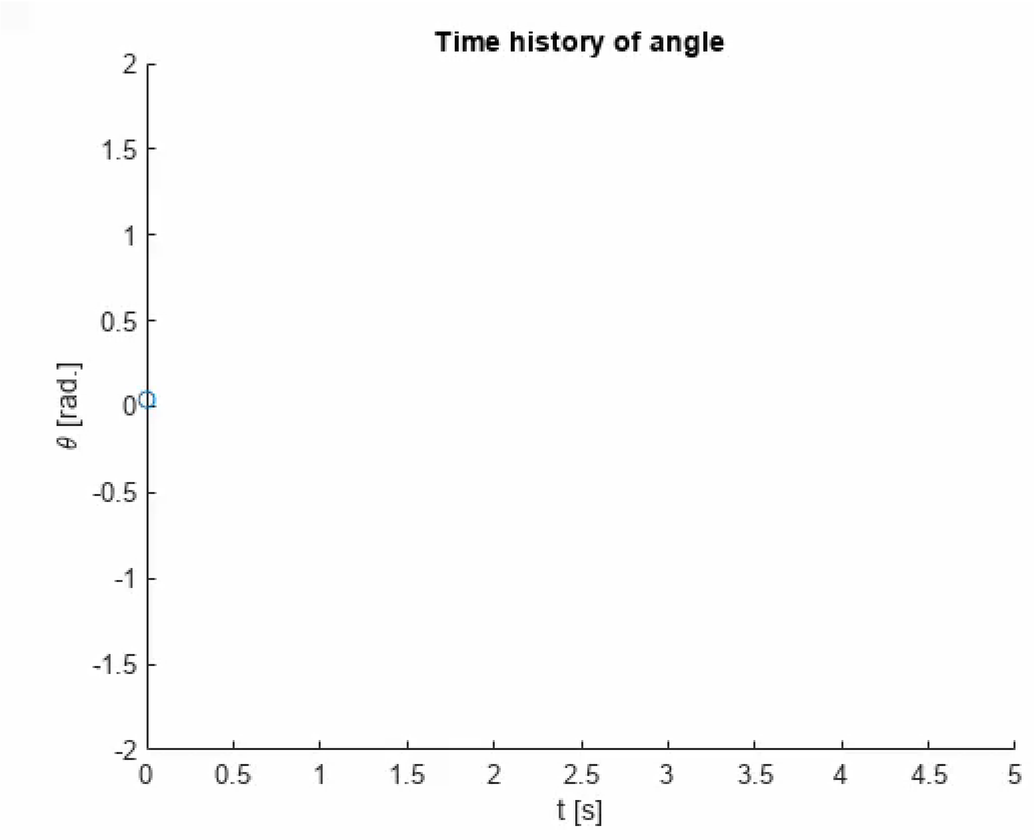


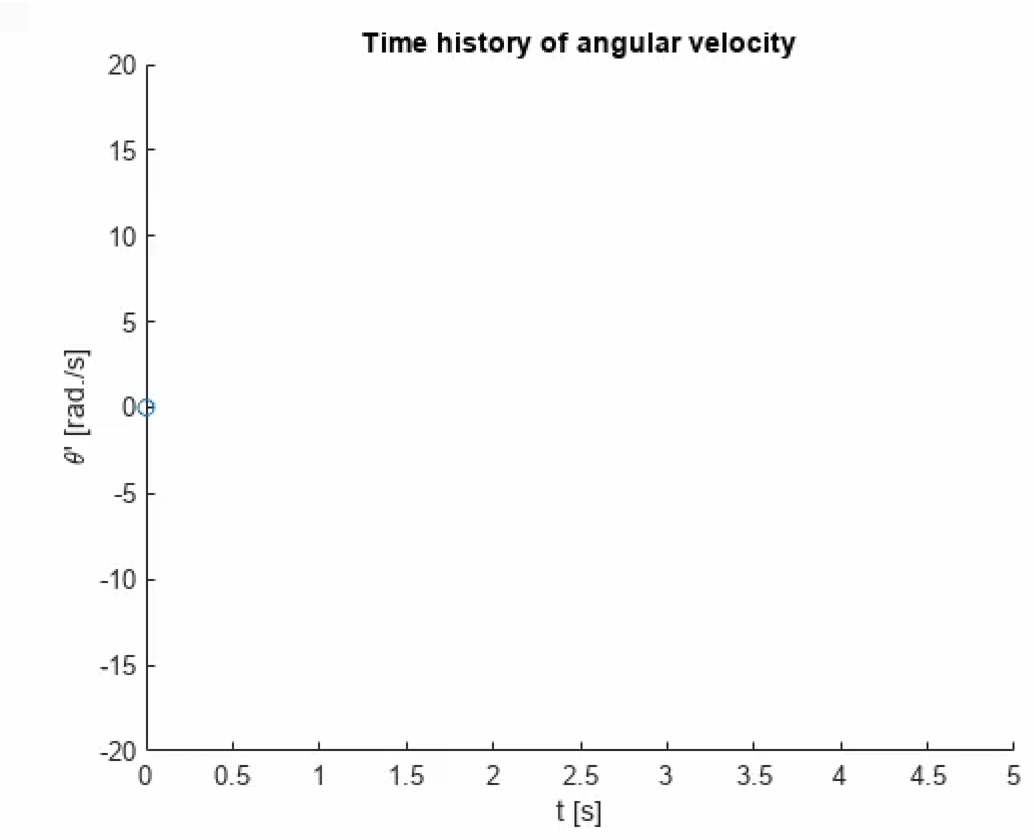

```
TIME=data.Time; DATA=data.Variables; N=length(TIME);
DTIME=linspace(0,time_of_measurement-1/dq.Rate,N)';
VOLTAGE1=DATA(1:end,1); VOLTAGE2=DATA(1:end,2);
ANGLE=-(VOLTAGE1-2.132)*(0.5*3.1415926)/1.3821;
VELOCITY=zeros(N,1);
for k=2:N
VELOCITY(k)=((ANGLE(k)-ANGLE(k-1))/(DTIME(k)-DTIME(k-1)));
end

figure;
%plot(DTIME, ANGLE,'r')
hold on
xlabel('t [s]'); ylabel('\theta [rad.]');
title('Time history of angle');
xlim([0 time_of_measurement]); ylim([-2 2]);
comet(DTIME, ANGLE), hold off
figure;
%plot(DTIME, VELOCITY,'g')
hold on
xlabel('t [s]'); ylabel('\theta'' [rad./s]');
title('Time history of angular velocity');
xlim([0 time_of_measurement]); ylim([-20 20]);
comet(DTIME, VELOCITY), hold off
```

```
figure; %plot(ANGLE, VELOCITY,'b')
hold on
xlabel('\theta [rad.]'); ylabel('\theta'' [rad./s]');
title('Phase portrait’); xlim([-2 2]); ylim([-20 20]);
comet(ANGLE, VELOCITY)
hold off
%animation
figure; l=0.12; Cx=l*sin(ANGLE); Cy=-l*cos(ANGLE);
drawArrow = @(x,y) quiver(x(1),y(1),x(2)-x(1),y(2)-y(1),0);
axislimits=[-0.12 0.12 -0.14 0.05];
for k=1:N
plot([0],[0],'--',0,0,'ko');
axis equal; hold on; axis(axislimits);
x1 = [-0.04 0.04]; y1 = [0 0]; drawArrow(x1,y1);
x2 = [0 0]; y2 = [-0.04 0.04]; drawArrow(x2,y2);
plot([0,Cx(k)],[0,Cy(k)],'g','LineWidth',2);
plot([Cx(k)],[Cy(k)],'ko','MarkerFaceColor','k');
text(0+0.002,0+0.005,'0');
text(0.04+0.002,0,'x');
text(0,0.04+0.005,'y');
text(Cx(k)-0.003,Cy(k)-0.005,'tip');
hold off; pause(0.001);
end
```

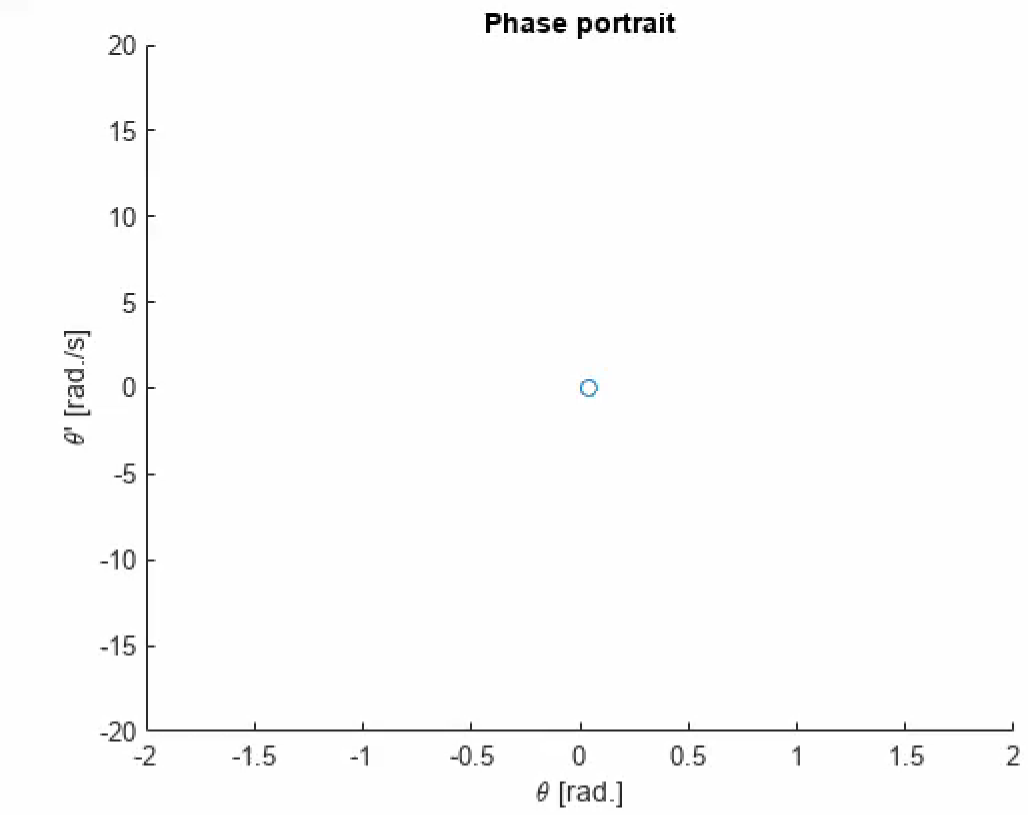
Damped physical pendulum

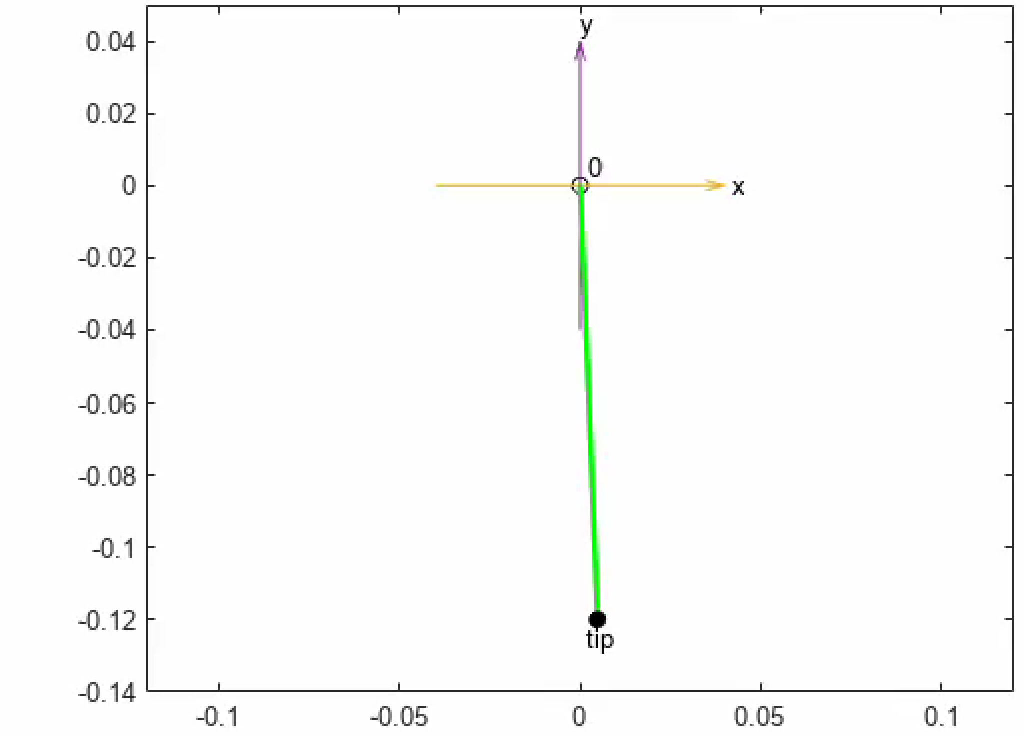






Damped physical pendulum





Animate - Mathematica

Animate

```
A1 = Animate[{t, Sin[t]}, {t, -1., 1.}]
Export[NotebookDirectory[] <> "A1.gif", A1, "AnimationRepetitions" -> Infinity]
```

Animate [*expr*, { u, u_{min}, u_{max} }]

generates an animation of *expr* in which u varies continuously from u_{min} to u_{max}.

```
A2 = Animate[{t, Sin[t]}, {t, -1, 1, 0.1}]
```

Animate [*expr*, { u, u_{min}, u_{max}, du }]

takes u to vary in steps du.

```
A3 = Animate[{t, Sin[t]}, {t, {-1.0, -0.3, 0.1, 0.2, 0.5, 1.0}}]
```

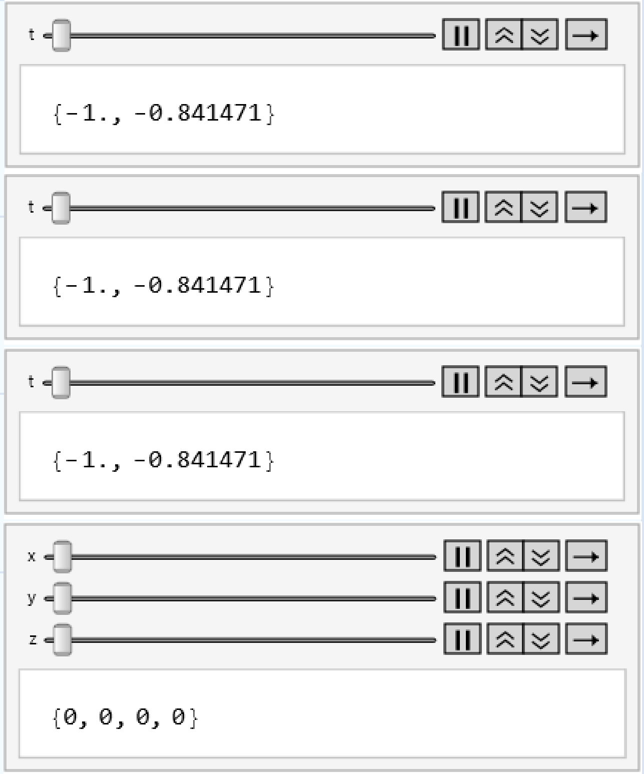
Animate [*expr*, { u, { u_1, u_2, … } }]

makes u take on discrete values u_1, u_2, ….

```
A41 = Animate[{x, y, z, x + y + z}, {x, 0, 1}, {y, 0, 1}, {z, 0, 1}]
```

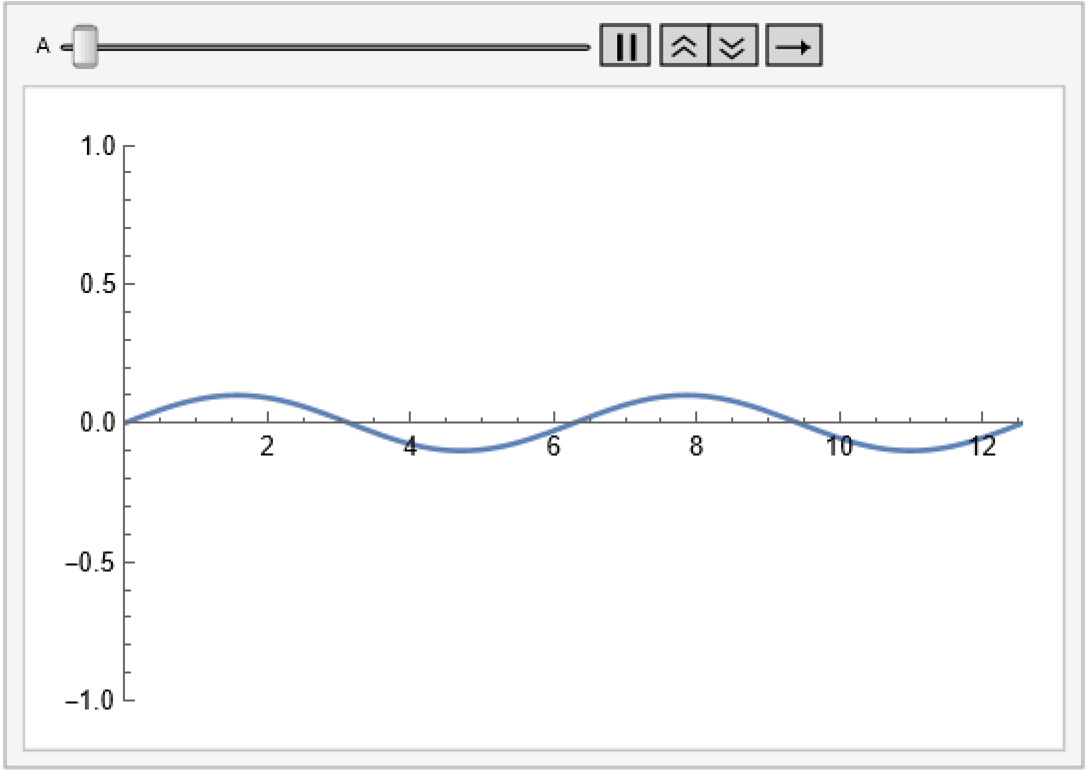
Animate [*expr*, { u, … }, { v, … }, …]

varies all the variables u, v, ….

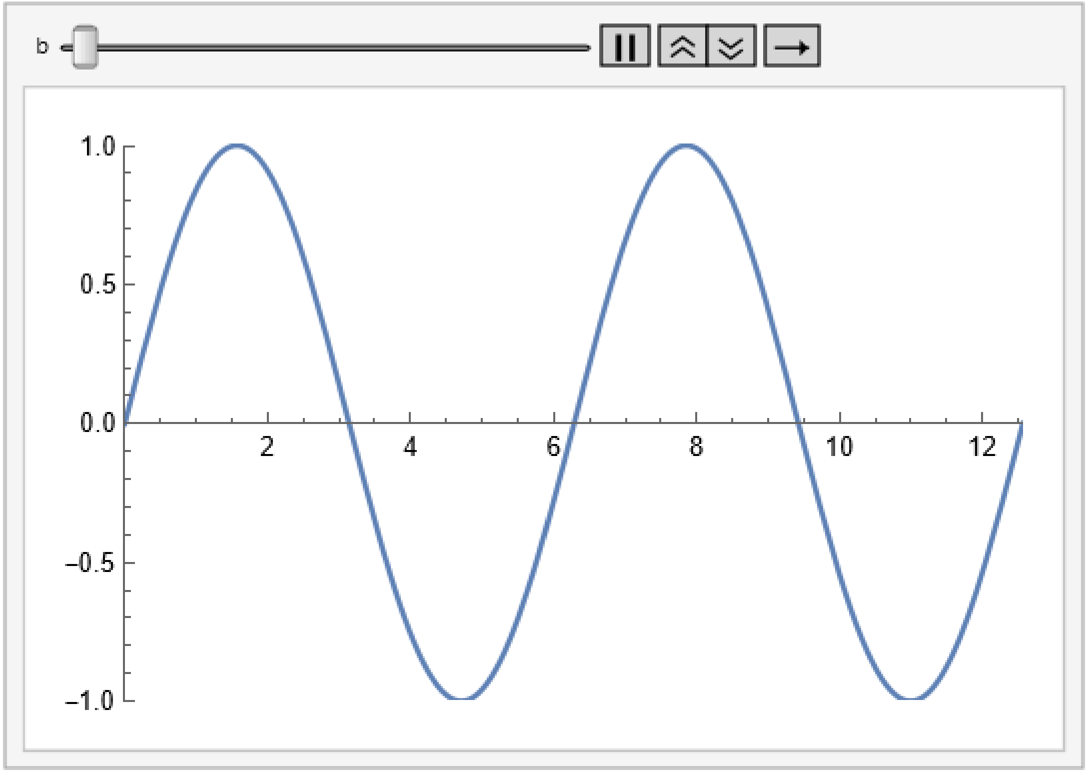


Animate - Mathematica

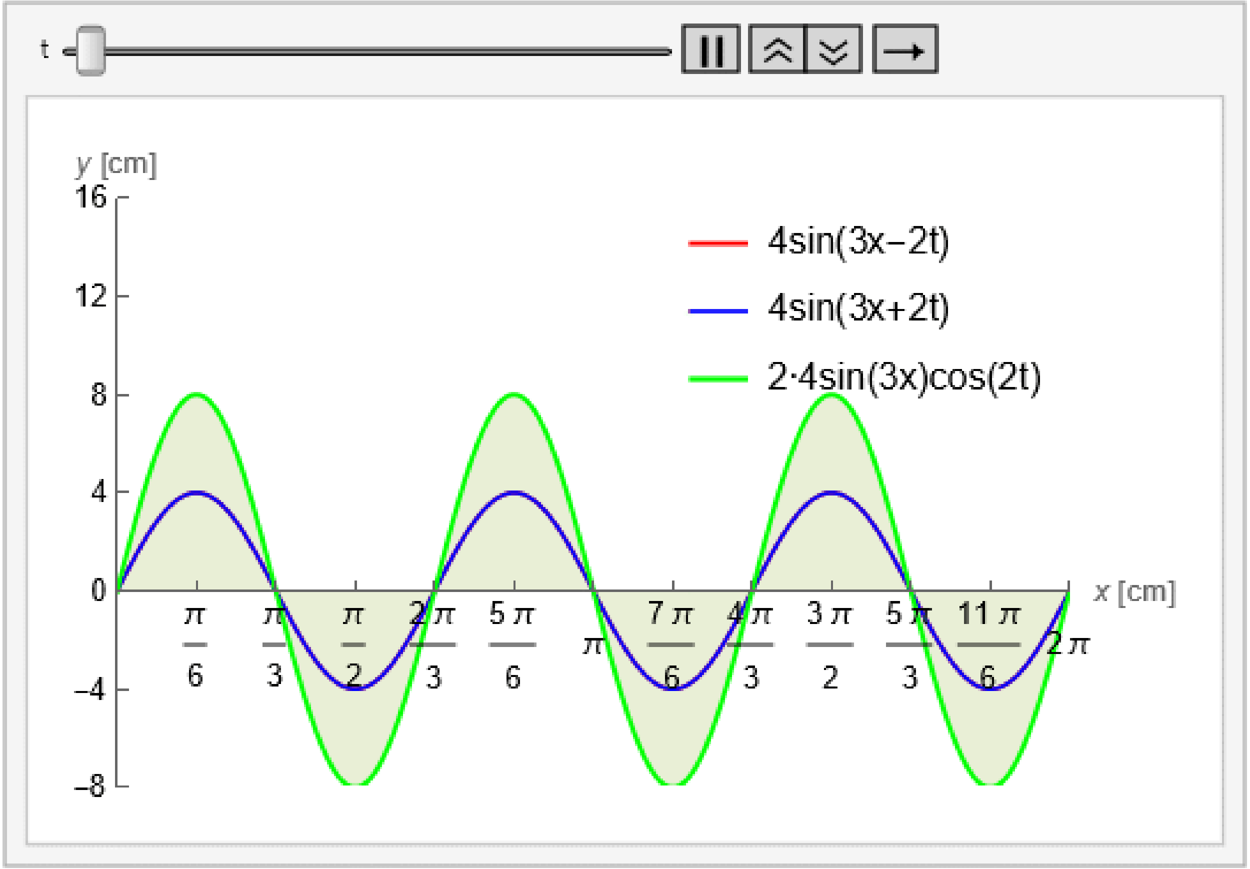
```
W1 = Animate[Plot[A Sin[t], {t, 0, 4 Pi}, PlotRange -> {{0, 4 Pi}, {-1, 1}}], {A, 0.1, 1}, AnimationRunning -> True]
```




```
W2 = Animate[Plot[Sin[t + b], {t, 0, 4 Pi}, PlotRange -> {{0, 4 Pi}, {-1, 1}}], {b, 0, 2 Pi}, AnimationRunning -> True]
```

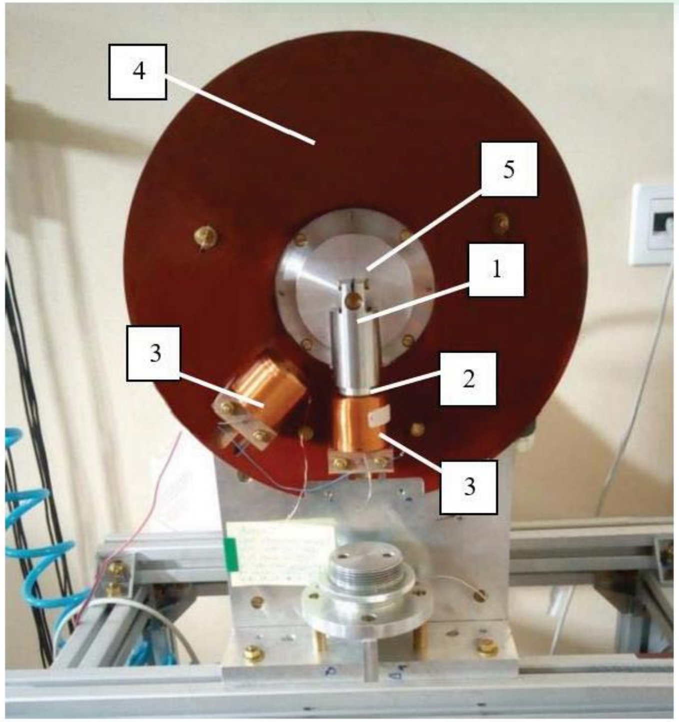


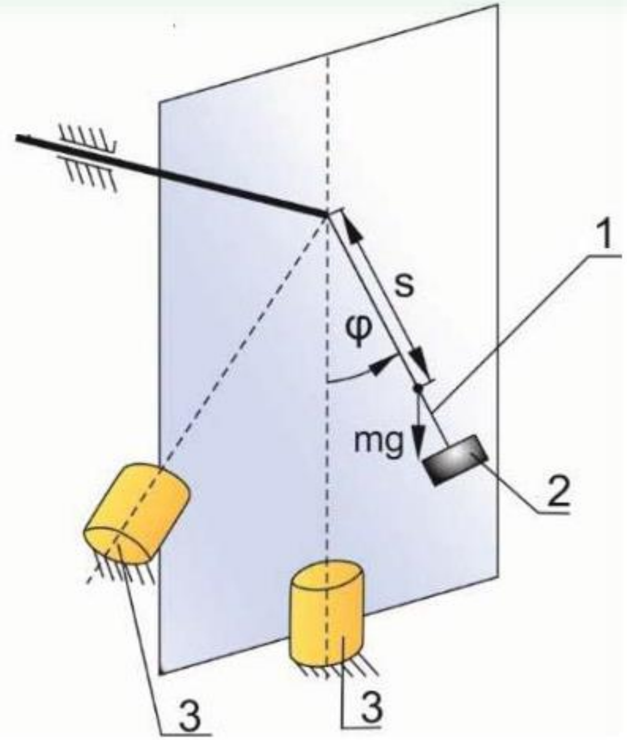
Example - Standing wave

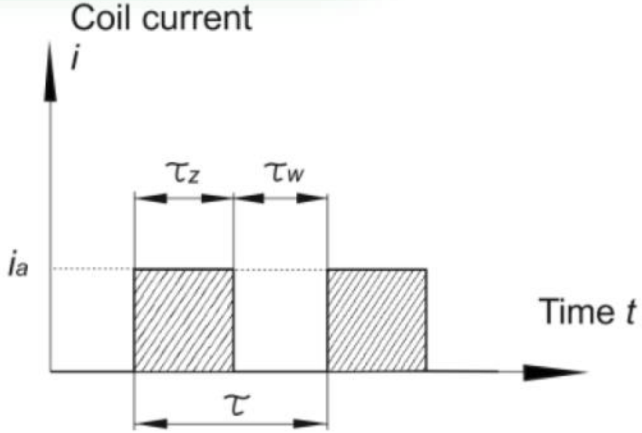


Example – Aerostatic pendulum

$$I\ddot{\varphi} + c\dot{\varphi} + mgs\sin\varphi = M_{1mag}(\varphi, i) + M_{2mag}(\varphi, i)$$





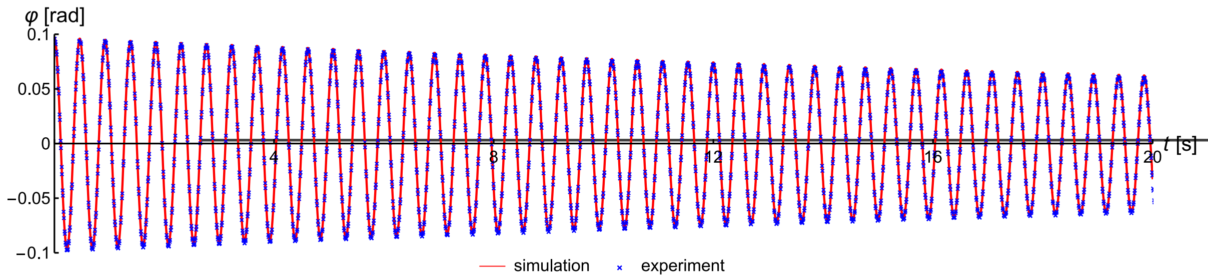


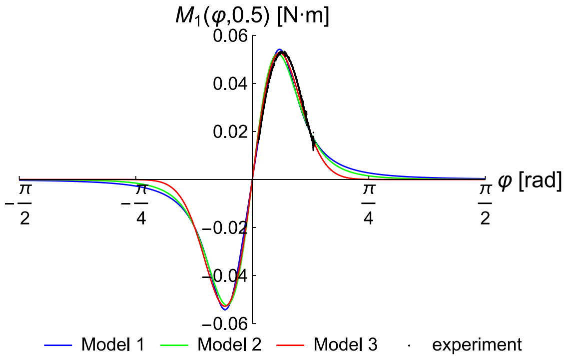
$$M_{1mag}(\varphi, i) = Aie^{-\lambda\varphi^2}\varphi,$$

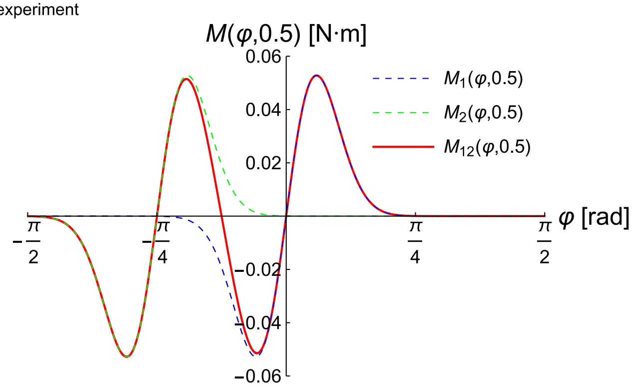
$$M_{2mag}(\varphi, i) = Aie^{-\lambda\left(\varphi+\frac{\pi}{4}\right)^2}(\varphi + \pi/4),$$

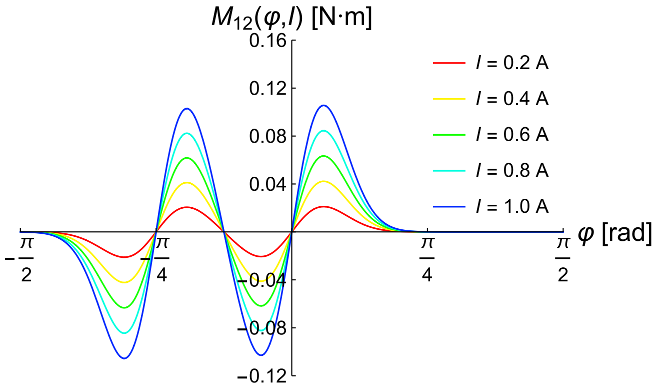
Experimental rig: 1 – physical pendulum, 2 – neodymium magnet, 3 – electric
textolite board, 5 – aluminium disk.

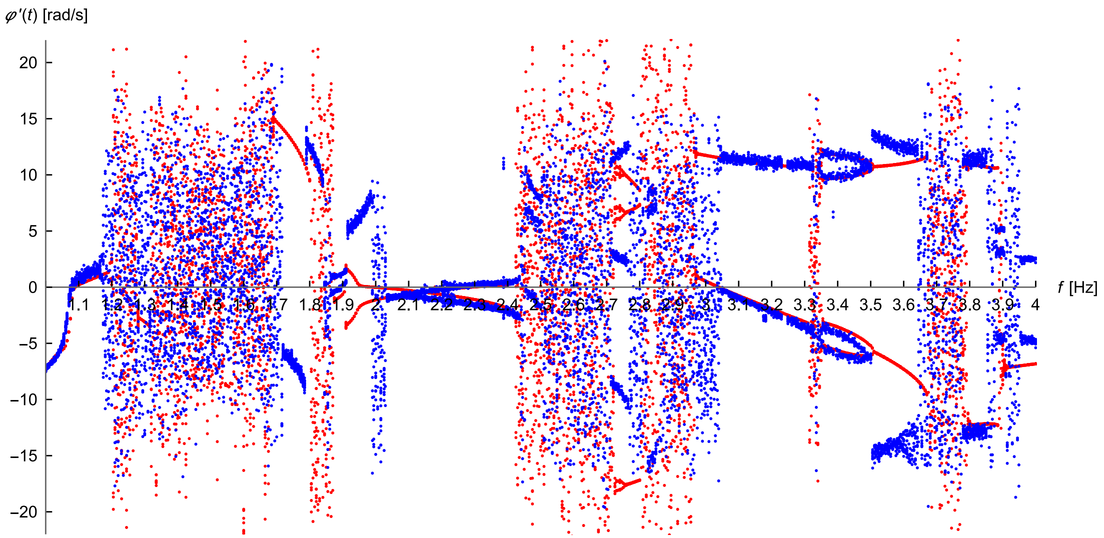
Identification

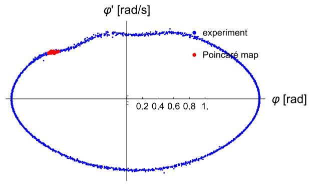


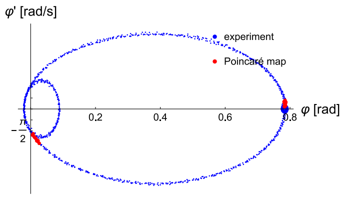


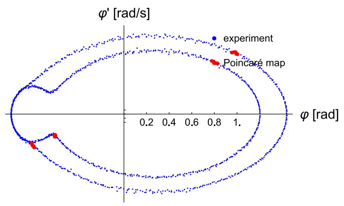




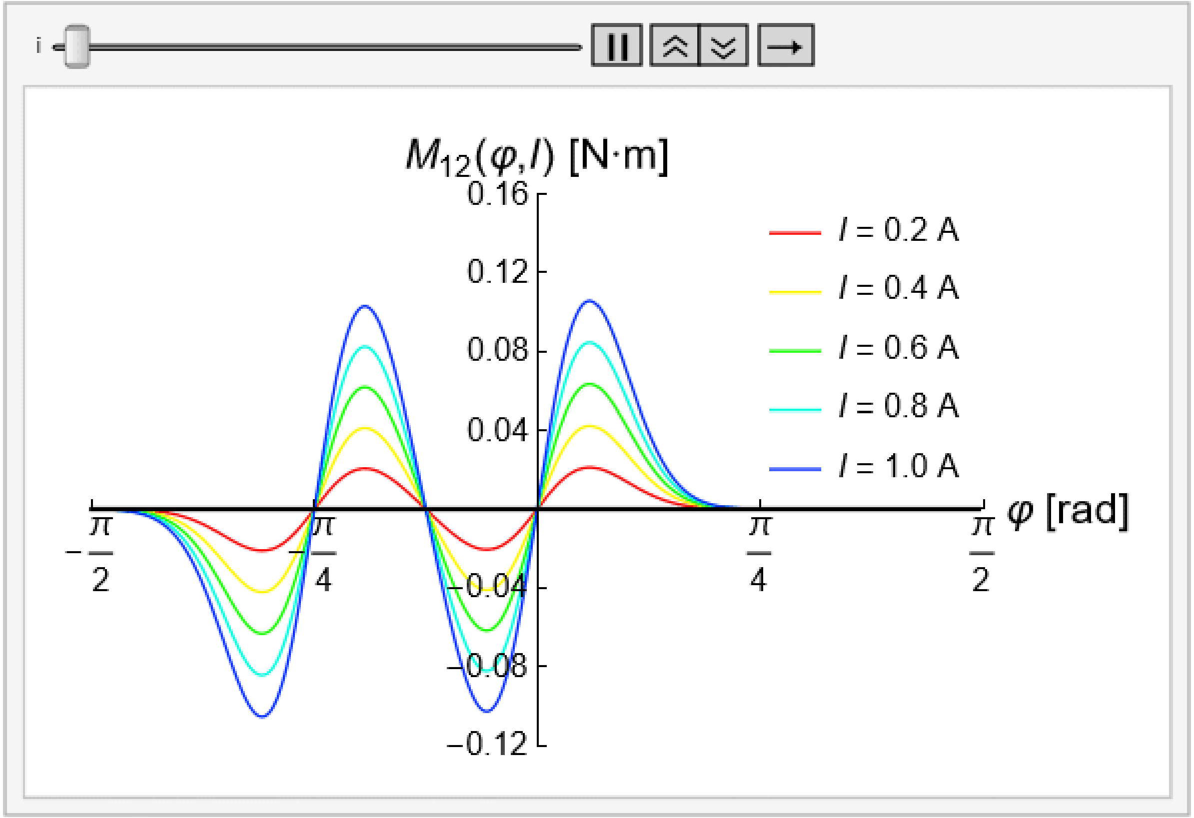




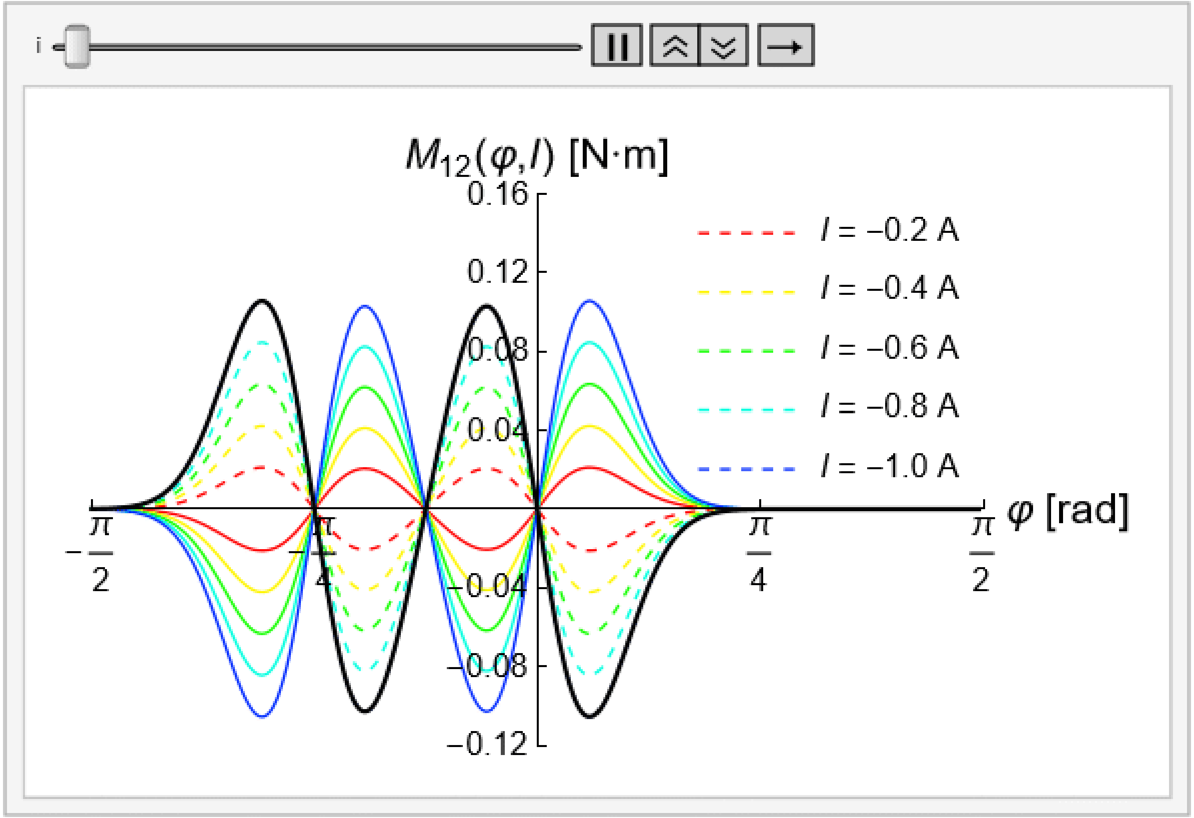




Animations

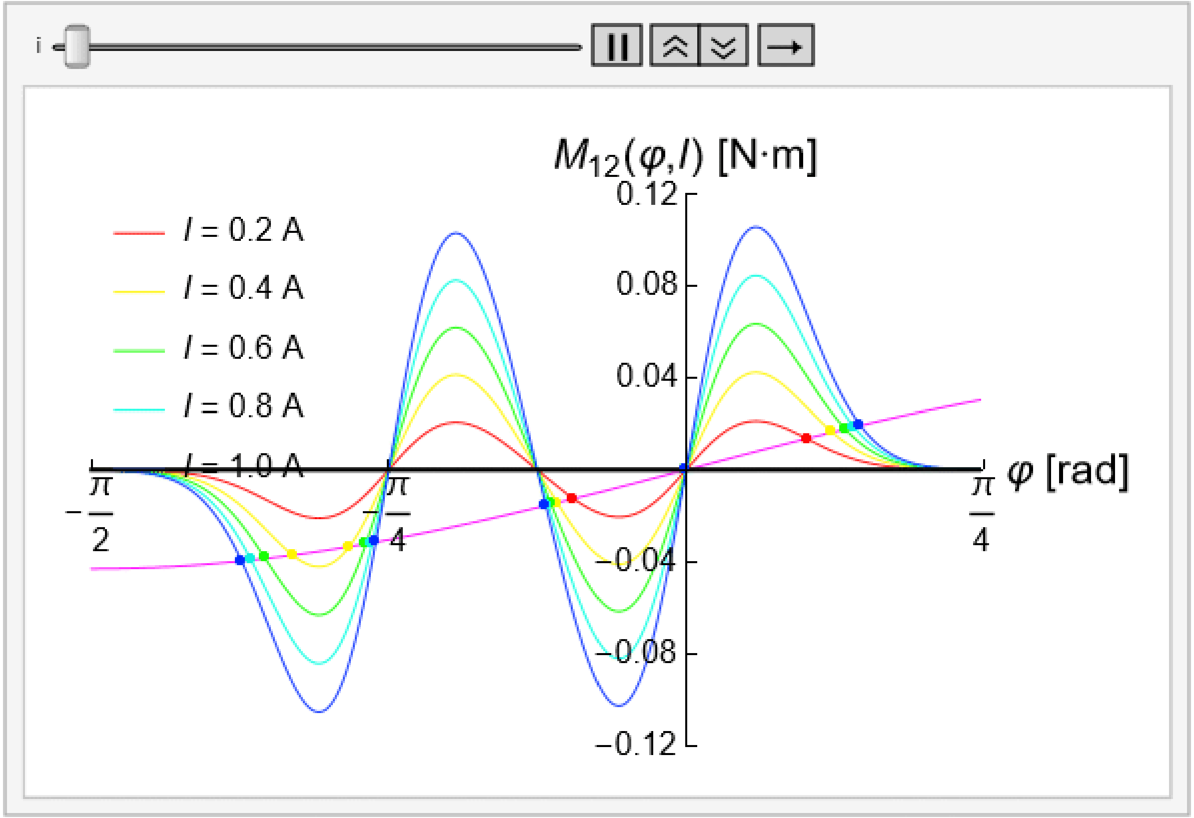


i
$M_{12}(\varphi,I)$ [N·m]
0.16
0.12
0.08
0.04
−0.04
−0.08
−0.12
$-\frac{\pi}{2}$
$-\frac{\pi}{4}$
$\frac{\pi}{4}$
$\frac{\pi}{2}$
φ [rad]
I = 0.2 A
I = 0.4 A
I = 0.6 A
I = 0.8 A
I = 1.0 A

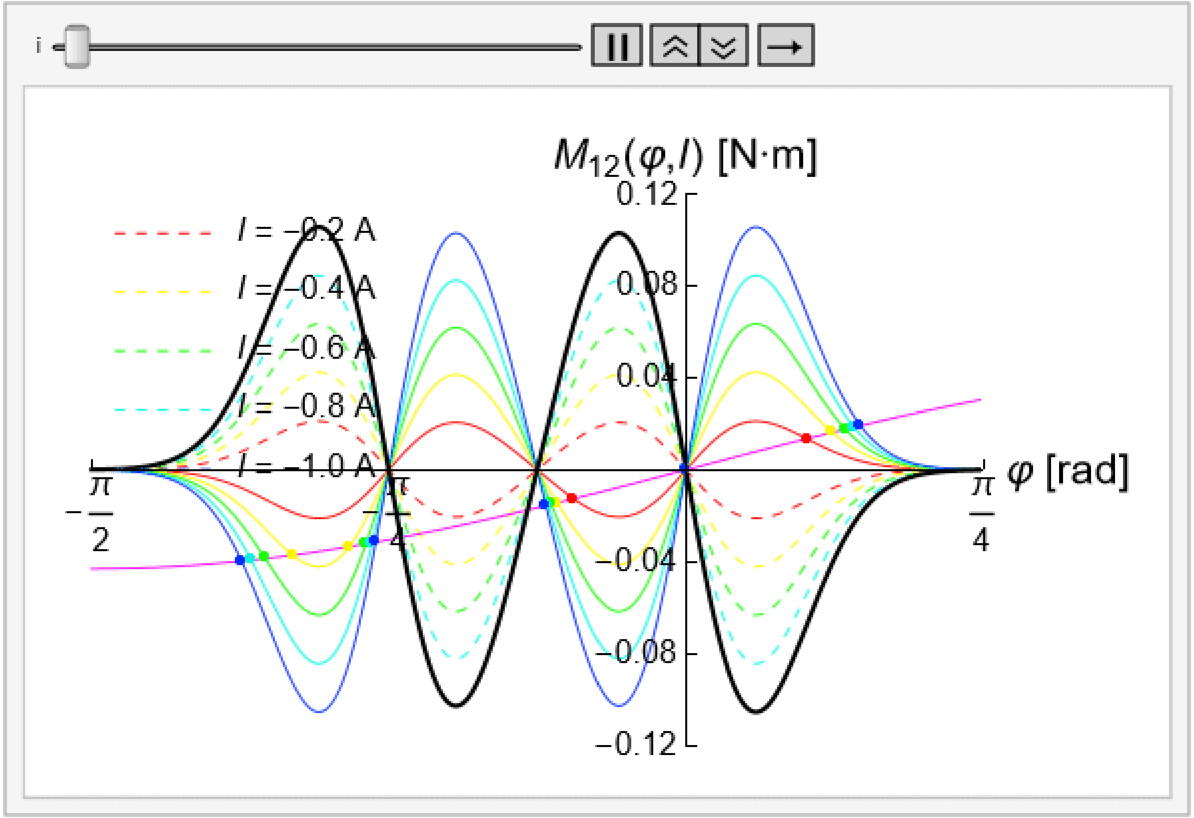


i
$M_{12}(\varphi,I)$ [N·m]
0.16
0.12
0.08
0.04
−0.04
−0.08
−0.12
$-\frac{\pi}{2}$
$-\frac{\pi}{4}$
$\frac{\pi}{4}$
$\frac{\pi}{2}$
φ [rad]
I = −0.2 A
I = −0.4 A
I = −0.6 A
I = −0.8 A
I = −1.0 A

Animations

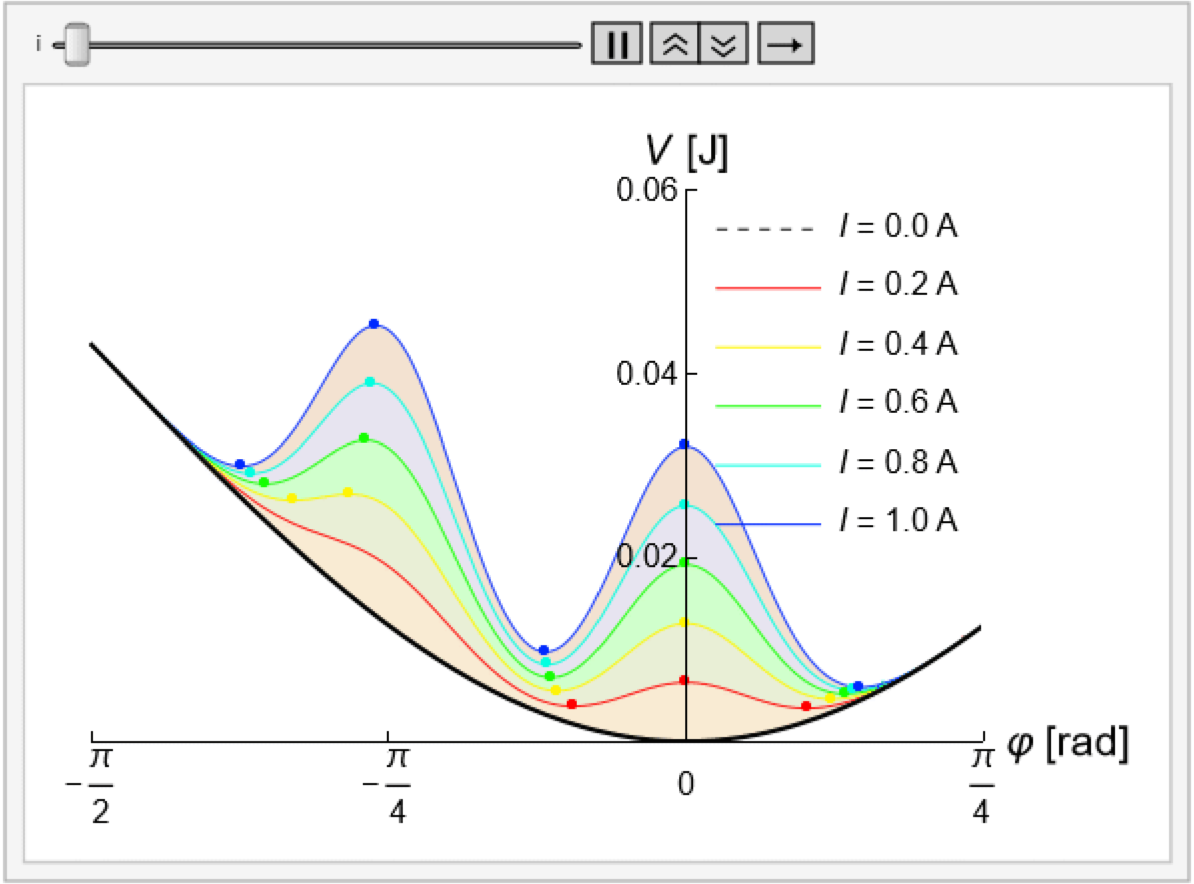


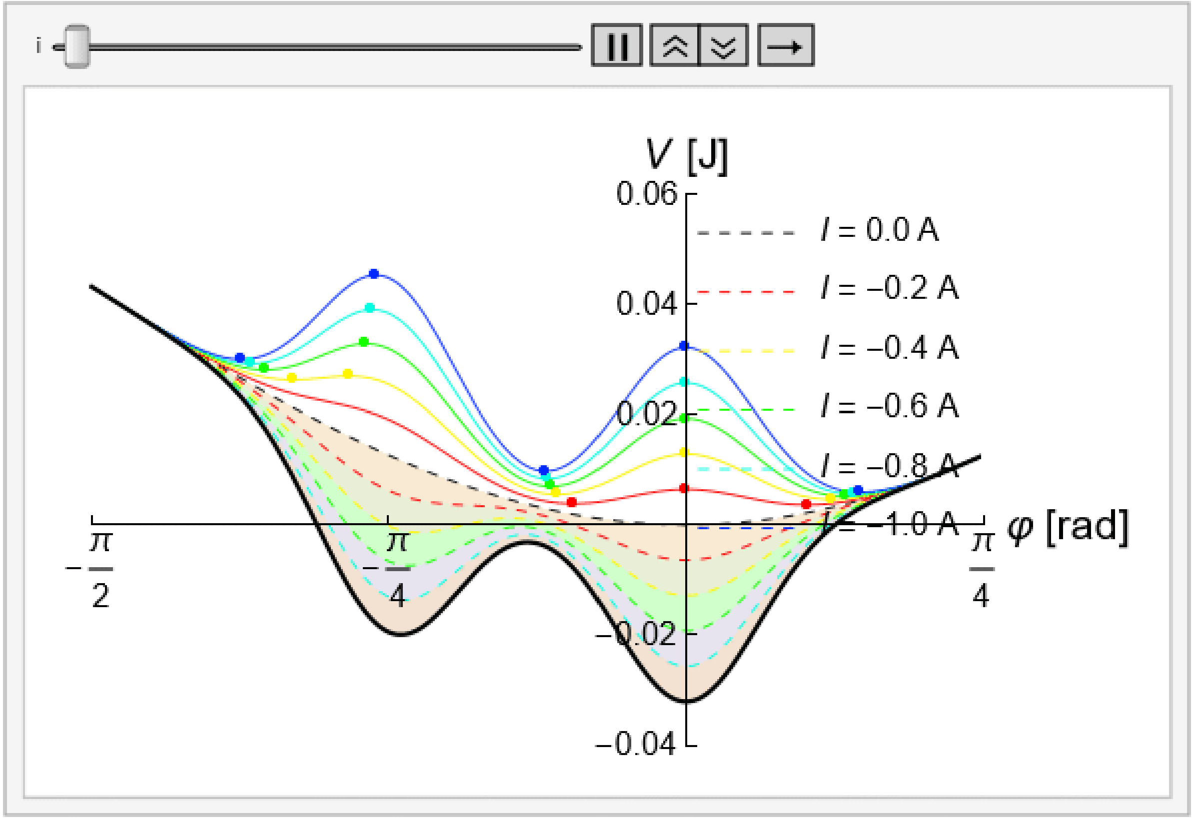
i
$M_{12}(\varphi,I)$ [N·m]
0.12
0.08
0.04
−0.04
−0.08
−0.12
I = 0.2 A
I = 0.4 A
I = 0.6 A
I = 0.8 A
I = 1.0 A
$-\frac{\pi}{2}$
$-\frac{\pi}{4}$
$\frac{\pi}{4}$
φ [rad]



i
$M_{12}(\varphi,I)$ [N·m]
0.12
0.08
0.04
−0.04
−0.08
−0.12
I = −0.2 A
I = −0.4 A
I = −0.6 A
I = −0.8 A
I = −1.0 A
$-\frac{\pi}{2}$
$-\frac{\pi}{4}$
$\frac{\pi}{4}$
φ [rad]

Animations


i
V [J]
0.06
0.04
0.02
I = 0.0 A
I = 0.2 A
I = 0.4 A
I = 0.6 A
I = 0.8 A
I = 1.0 A
φ [rad]
−π/2
−π/4
0
π/4


i
V [J]
0.06
0.04
0.02
−0.02
−0.04
I = 0.0 A
I = −0.2 A
I = −0.4 A
I = −0.6 A
I = −0.8 A
I = −1.0 A
φ [rad]
−π/2
−π/4
π/4

Animations

```
OXYZ = {{Black, Ball[{0, 0, 0}, 0.001]}, {Red, Arrowheads[0.02], Arrow[{{0, 0, 0}, {0.1, 0, 0}}, 0.02]},
         czarny kula                      cz⋯  groty strzałek    strzałka
   {Green, Arrowheads[0.02], Arrow[{{0, 0, 0}, {0, 0.1, 0}}, 0.02]}, {Blue, Arrowheads[0.02], Arrow[{{0, 0, 0}, {0, 0, 0.1}}, 0.02]},
    zielony groty strzałek    strzałka                                niebi⋯ groty strzałek    strzałka
   Text["x", {0.1, 0, 0}], Text["y", {0, 0.1, 0}], Text["z", {0, 0, 0.1}], Text["O", {-0.007, -0.007, -0.007}]};
   tekst                   tekst                   tekst                   tekst

PODSTAWA = {{Gray, Opacity[0.9], Cuboid[{0, -0.1, -0.265}, {0.05, 0.1, -0.25}]}, {Gray, Opacity[0.9], Cuboid[{-0.015, -0.1, -0.265}, {0, 0.1, 0}]}};
             szary nieprzezroczyst⋯ prostopadłościan                               szary nieprzezroczyst⋯ prostopadłościan

TARCZA = {{Brown, Opacity[1.0], Cylinder[{{0, 0, 0}, {0.01, 0, 0}}, 0.14]}, {Gray, Opacity[.9], Cylinder[{{0, 0, 0}, {0.011, 0, 0}}, 0.03]}};
           brązowy nieprzezroczyst⋯ walec                                      szary nieprzezroczys⋯ walec
```

Animations

Wolfram Mathematica

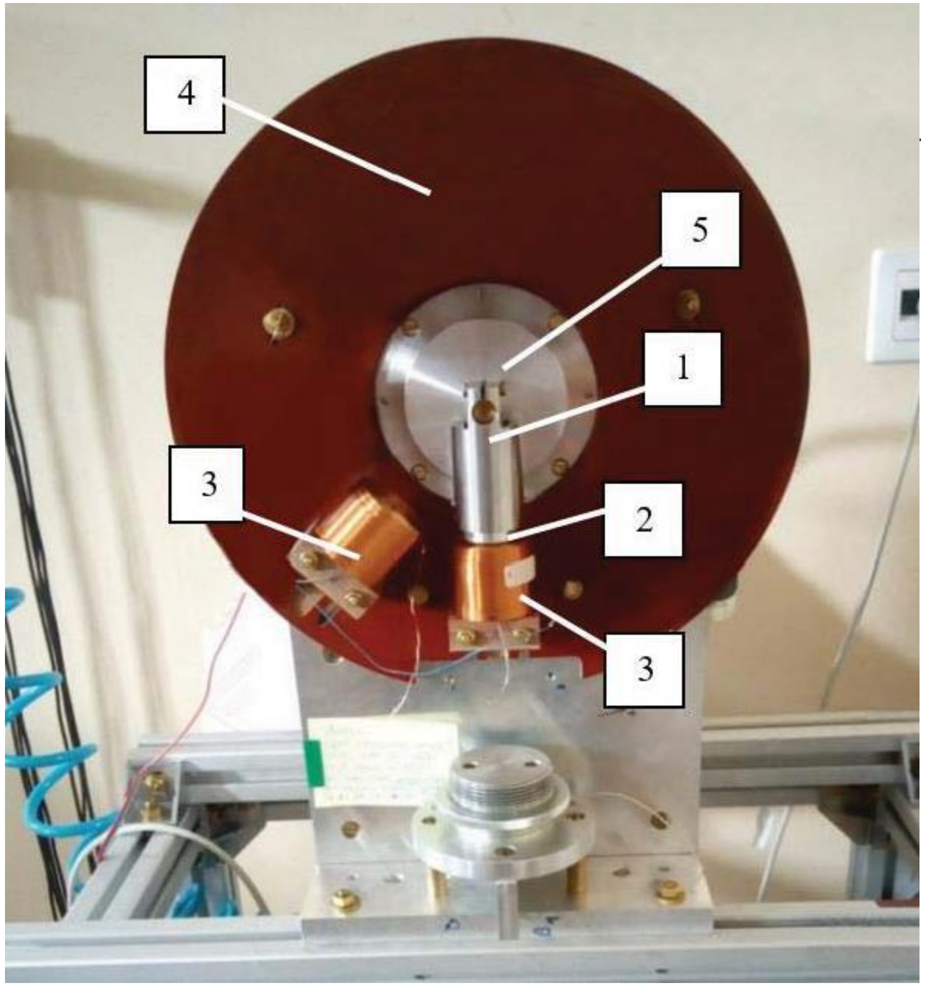
rdzenie cewek

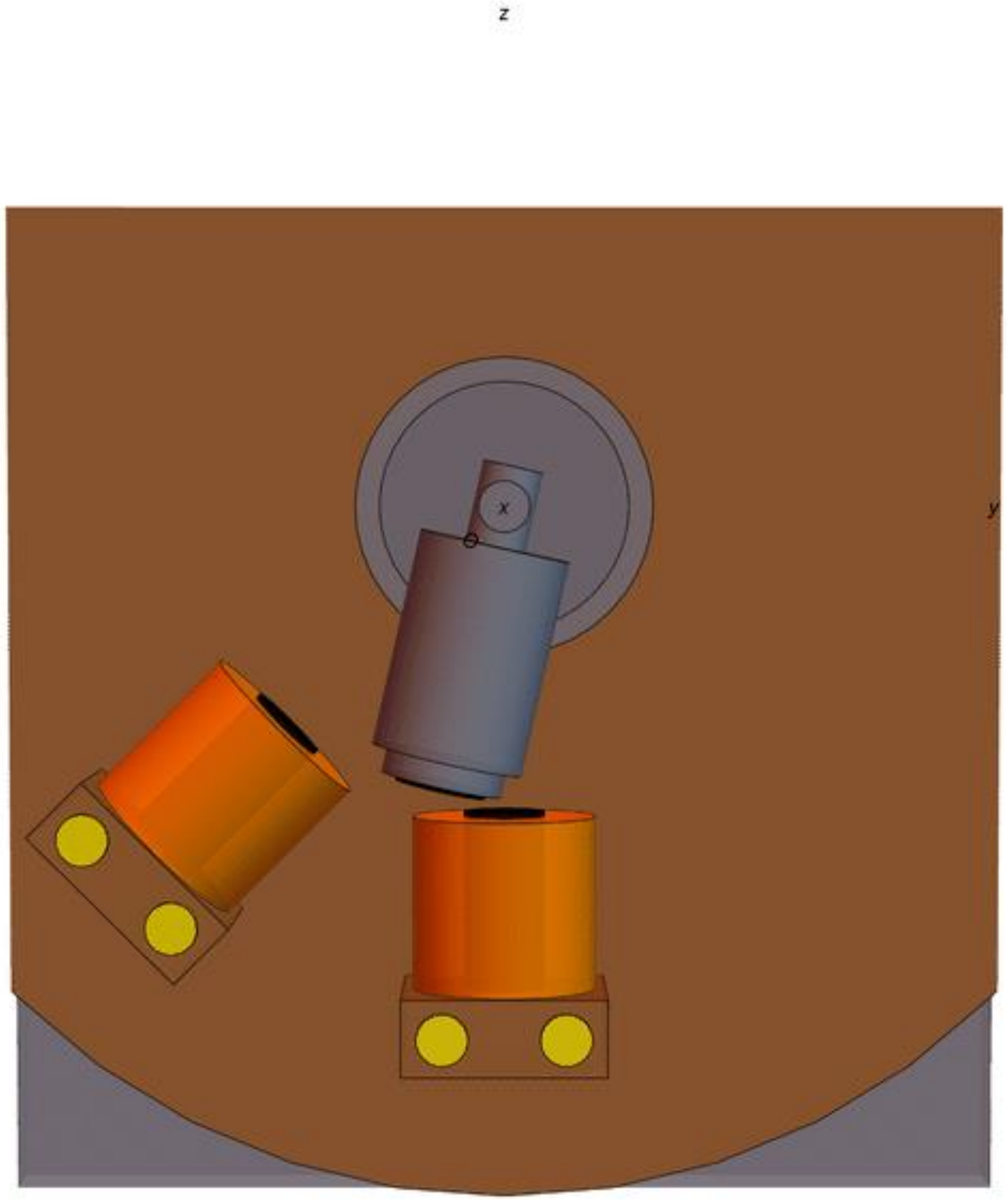
```
In[6]:= RDZEN1 = Rotate[{{Brown, Opacity[0.9], Translate[Cuboid[{0, -0.02, 0}, {0.04, 0.02, 0.015}], {0.01, 0, -0.11}]},
                  obróć       brązowy nieprzezroczyst··· przesuń ró··· prostopadłościan
        {Yellow, Opacity[.8], Translate[Cylinder[{{0, 0, 0}, {0.002, 0, 0}}, 0.005], {0.04 + 0.01, -0.012, 0.015 / 2 - 0.11}]},
         żółty    nieprzezroczys··· przesuń ró··· walec
        {Yellow, Opacity[.8], Translate[Cylinder[{{0, 0, 0}, {0.002, 0, 0}}, 0.005], {0.04 + 0.01, 0.012, 0.015 / 2 - 0.11}]},
         żółty    nieprzezroczys··· przesuń ró··· walec
        {Black, Opacity[.9], Translate[Cylinder[{{0, 0, 0}, {0, 0, 0.035}}, 0.008], {0.02 + 0.01, 0, 0.015 - 0.11}]}}, 0 Degree, {1, 0, 0}];
         czarny  nieprzezro          przesuń ró··· walec                                                               stopień

In[7]:= RDZEN2 = Rotate[{{Brown, Opacity[0.9], Translate[Cuboid[{0, -0.02, 0}, {0.04, 0.02, 0.015}], {0.01, 0, -0.11}]},
                  obróć       brązowy nieprzezroczyst··· przesuń ró··· prostopadłościan
        {Yellow, Opacity[.8], Translate[Cylinder[{{0, 0, 0}, {0.002, 0, 0}}, 0.005], {0.04 + 0.01, -0.012, 0.015 / 2 - 0.11}]},
         żółty    nieprzezroczys··· przesuń ró··· walec
        {Yellow, Opacity[.8], Translate[Cylinder[{{0, 0, 0}, {0.002, 0, 0}}, 0.005], {0.04 + 0.01, 0.012, 0.015 / 2 - 0.11}]}(*,
         żółty    nieprzezroczys··· przesuń ró··· walec
        {Orange,Opacity[.9],Translate[Cylinder[{{0,0,0},{0,0,0.034}},0.018],{0.02+0.01,0,0.015-0.11}]}*),
        {Black, Opacity[.9], Translate[Cylinder[{{0, 0, 0}, {0, 0, 0.035}}, 0.008], {0.02 + 0.01, 0, 0.015 - 0.11}]}}, -45 Degree, {1, 0, 0}];
         czarny  nieprzezro          przesuń ró··· walec                                                                 stopień

In[8]:= WALEK = {{Gray, Opacity[.9], Cylinder[{{0.012, 0, 0}, {0.015, 0, 0}}, 0.025]}, {Gray, Opacity[.9], Cylinder[{{0, 0, 0}, {0.04, 0, 0}}, 0.005]}};
                  szary  nieprzezroczys··· walec                                          szary  nieprzezroczys··· walec

In[5]:= WAHADLO = Translate[{{Gray, Opacity[.9], Cylinder[{{0, 0, 0.008}, {0, 0, -0.008}}, 0.006]},
                  przesuń równo  szary  nieprzezroczys··· walec
        {Gray, Opacity[.9], Cylinder[{{0, 0, -0.008}, {0, 0, -0.051}}, 0.015]}, {Gray, Opacity[.9], Cylinder[{{0, 0, -0.051}, {0, 0, -0.056}}, 0.012]},
         szary  nieprzezroc:       walec                                        szary                 walec
        {Black, Opacity[1.0], Cylinder[{{0, 0, -0.056}, {0, 0, -0.057}}, 0.009]}}, {0.03, 0, 0}];
         czarny  nieprzezroczyst··· walec
```

Animations

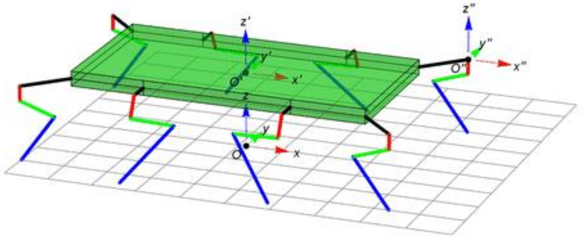
```
TABANIM =
  Table[
  tabela
   Graphics3D[{OXYZ, PODSTAWA, TARCZA, RDZEN1, RDZEN2, {Orange, Opacity[0.5 + (0.4 / 0.5) * tabi[[τ]]],
   trójwymiarowa grafika                                pomara⋯ nieprzezroczystość
      Rotate[Translate[Cylinder[{{0, 0, 0}, {0, 0, 0.034}}, 0.018], {0.02 + 0.01, 0, 0.015 - 0.11}], 0 Degree, {1, 0, 0}]},
      obróć  przesuń ró⋯ walec                                                                        stopień
     {Orange, Opacity[0.5 + (0.4 / 0.5) * tabi[[τ]]], Rotate[Translate[Cylinder[{{0, 0, 0}, {0, 0, 0.034}}, 0.018], {0.02 + 0.01, 0, 0.015 - 0.11}],
      pomara⋯ nieprzezroczystość                       obróć  przesuń ró⋯ walec
       -45 Degree, {1, 0, 0}], WALEK, Rotate[WAHADLO, tabFI[[τ]], {1, 0, 0}]}(**)}, PlotRange → {{-0.14, 0.14}, {-0.14, 0.14}, {-0.14, 0.14}},
           stopień                 obróć                                         zakres wykresu
    ViewVector → {0.6, -0.4 * 0, 0.4 * 0}(*,PlotRange→{{-0.35,0.35},{-0.35,0.35},{-0.3,0.3}},ViewVector→{0,0,10}*), ImageSize → {Large}, Boxed → True],
    wektor widzenia                                                                                                   rozmiar obrazu  duży   dodaj ⋯ prawda
   {τ, 1, tk / krok, 2}(*,AnimationRunning→False*)];

                              Export[NotebookDirectory[] <> "Pendulum2024.gif",
                              eksportuj katalog notatnika
                               Join[TABANIM(*,Reverse[TABANIM]*)], "AnimationRepetitions" → Infinity,
                               połącz                                                      nieskończono
                               ImageSize → {Automatic, 600}, ImageResolution → 300]
                               rozmiar obrazu  automatyczny  rozdzielczość obrazu
```

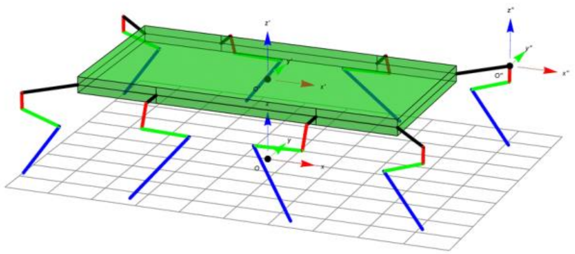
Animations

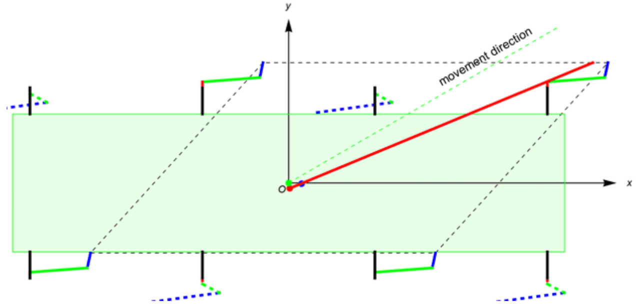


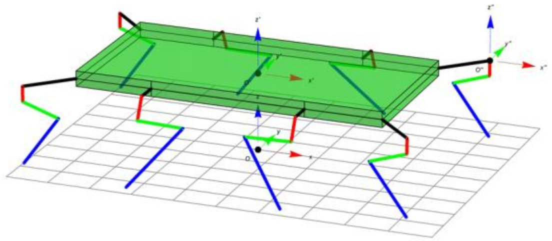


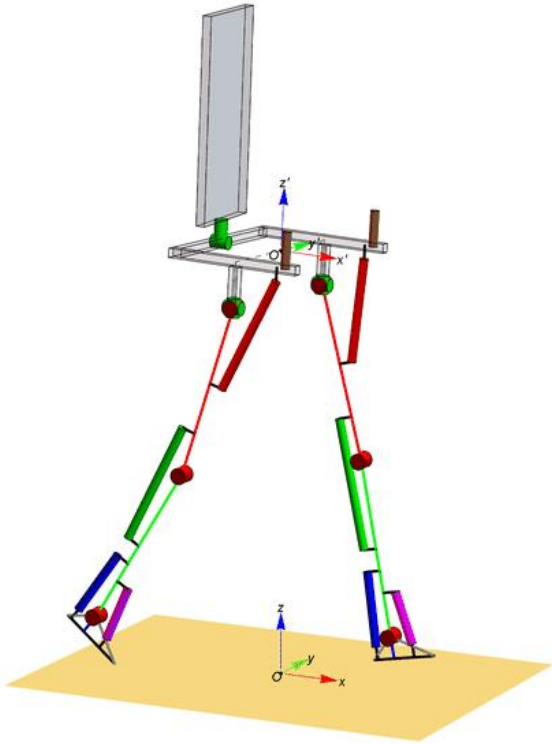
Other animations











Conclusions

- Easy interpretation and better understanding of the presented data
- Presentations are more attractive and attract attention of the listeners
- Animations can be used as supplementary material at submission the online version of a published research paper