Wykład 7

Grawitacja: zagadnienie Ziemia-Słońce, spadek swobodny, prędkość graniczna, prosty algorytm rozwiązujący równania różniczkowe, funkcje Block i Module, komentarz o pętlach

In[1]:=
```
ClearAll["Global`*"]
```
wyczyść wszystko

Równania grawitacji

Podejście naiwne

$$F = \frac{GMm}{r^2}$$

$$\begin{cases} m\frac{d^2x}{dt^2} + \frac{GMmx}{(x^2+y^2)^{\frac{3}{2}}} = 0 \quad \square \\ m\frac{d^2y}{dt^2} + \frac{GMmy}{(x^2+y^2)^{\frac{3}{2}}} = 0 \quad \square \end{cases}$$

Przyjmujemy taki układ jednostek, dla którego GM = 4 π^2

In[2]:=
```
Options[NDSolve]
```
opcje rozwiąż numerycznie równanie różniczkowe

Out[2]=
$$\{\text{AccuracyGoal} \to \text{Automatic}, \text{Compiled} \to \text{Automatic}, \text{DependentVariables} \to \text{Automatic},$$
$$\text{DiscreteVariables} \to \{\}, \text{EvaluationMonitor} \to \text{None}, \text{InterpolationOrder} \to \text{Automatic},$$
$$\text{MaxStepFraction} \to \frac{1}{10}, \text{MaxSteps} \to \text{Automatic}, \text{MaxStepSize} \to \text{Automatic},$$
$$\text{Method} \to \text{Automatic}, \text{NormFunction} \to \text{Automatic}, \text{PrecisionGoal} \to \text{Automatic},$$
$$\text{StartingStepSize} \to \text{Automatic}, \text{StepMonitor} \to \text{None}, \text{WorkingPrecision} \to \text{MachinePrecision}\}$$

In[3]:=
```
?? NDSolve
```

NDSolve[*eqns*, *u*, {*x*, x_{min}, x_{max}}] finds a numerical solution to the ordinary differential equations *eqns* for the function *u* with the independent variable *x* in the range x_{min} to x_{max}.
NDSolve[*eqns*, *u*, {*x*, x_{min}, x_{max}}, {*y*, y_{min}, y_{max}}] solves the partial differential equations *eqns* over a rectangular region.
NDSolve[*eqns*, *u*, {*x*, *y*} ∈ Ω] solves the partial differential equations *eqns* over the region Ω.
NDSolve[*eqns*, *u*, {*t*, t_{min}, t_{max}}, {*x*, *y*} ∈ Ω] solves the time-dependent partial differential equations *eqns* over the region Ω.
NDSolve[*eqns*, {u_1, u_2, …}, …] solves for the functions u_i. ≫

```
Attributes[NDSolve] = {Protected}

Options[NDSolve] =
 {AccuracyGoal → Automatic, Compiled → Automatic, DependentVariables → Automatic,
  DiscreteVariables → {}, EvaluationMonitor → None, InterpolationOrder → Automatic,
  MaxStepFraction → 1/10, MaxSteps → Automatic, MaxStepSize → Automatic,
  Method → Automatic, NormFunction → Automatic, PrecisionGoal → Automatic,
  StartingStepSize → Automatic, StepMonitor → None, WorkingPrecision → MachinePrecision}
```

In[4]:=
```
row1 =.
row2 =.
metody =.
metody = {Automatic,  "ImplicitRungeKutta", "ExplicitRungeKutta" , "BDF"}
          automatyczny
row1 = x''[t] + (4 π^2) x[t] / (x[t]^2 + y[t]^2)^(3/2)
row2 = y''[t] + (4 π^2) y[t] / (x[t]^2 + y[t]^2)^(3/2)
orbit = NDSolve[
        rozwiąż numerycznie równanie różniczkowe
    {row1 == 0,  row2 == 0,
     x[0] == 1,
     y[0] == 0,
     x'[0] == -π,
     y'[0] == 2 π
    }, {x, y}, {t, 0, 1.6}, Method → #] & /@ metody
                                 metoda
```

Out[7]= {Automatic, ImplicitRungeKutta, ExplicitRungeKutta, BDF}

Out[8]= $x''(t) + \frac{4\pi^2 x(t)}{\left(x(t)^2 + y(t)^2\right)^{3/2}}$

Out[9]= $\frac{4\pi^2 y(t)}{\left(x(t)^2 + y(t)^2\right)^{3/2}} + y''(t)$

Out[10]= {{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: Output: s

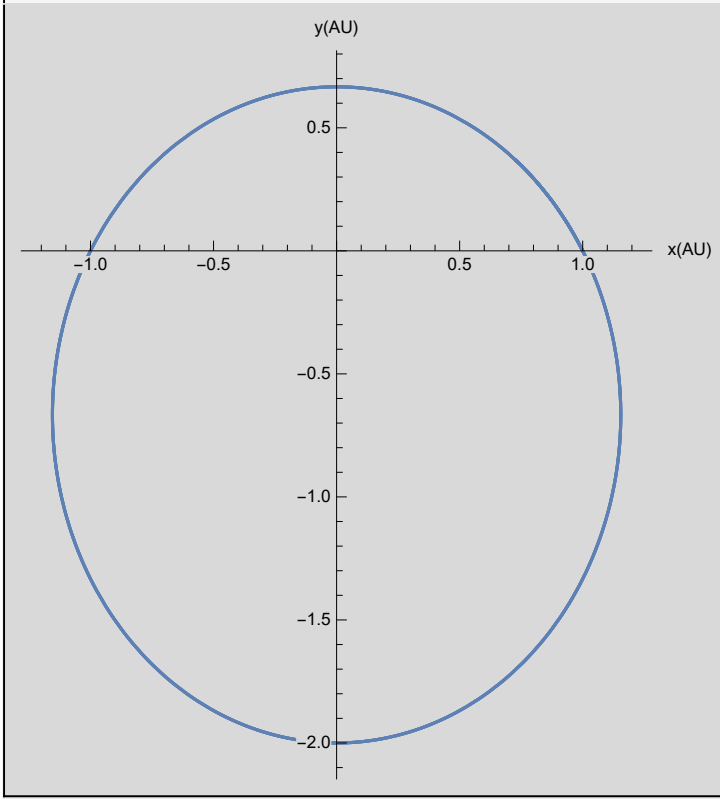
{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: Output: s

{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: Output: s

{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: Output: s

In[11]:=
```
Show@(ParametricPlot[Evaluate[{x[t], y[t]} /. #], {t, 0, 1.6},
 AspectRatio → Automatic, AxesLabel → {"x(AU)", "y(AU)"}] & /@ orbit)
```
pokaż | wykres parametry··· | oblicz
format obrazu | automatyczny | oznaczenia osi

Out[11]=



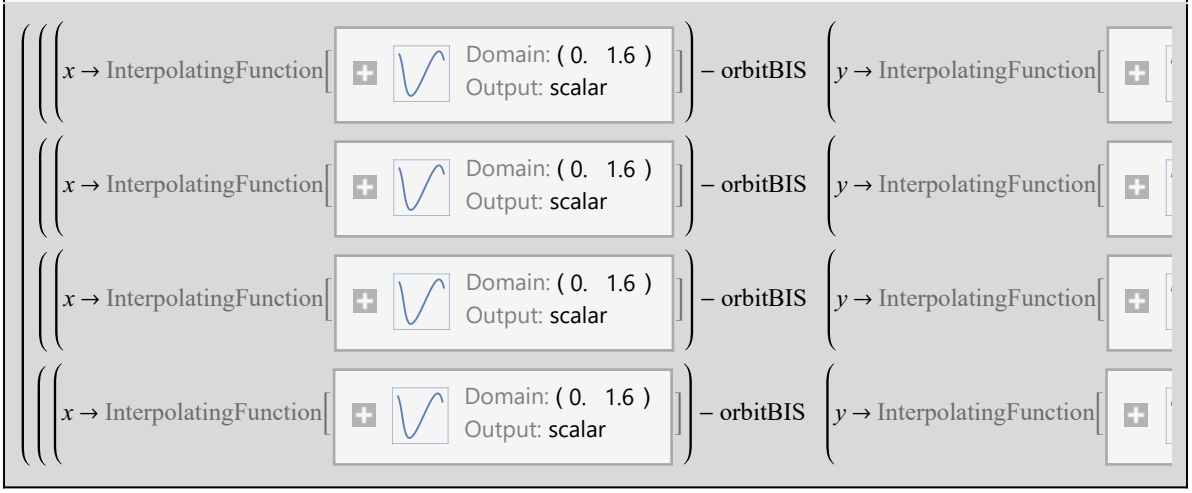
In[12]:=
```
MapThread[{##} &, {orbit, {Blue, Red, Pink, Green}, metody}]
```
zastosuj w wątku | niebi··· | cz··· | różowy | zielony

Out[12]=

{{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: (0. 1.6) Output: scalar]},
{{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: (0. 1.6) Output: scalar]},
{{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: (0. 1.6) Output: scalar]},
{{x → InterpolatingFunction[Domain: (0. 1.6) Output: scalar], y → InterpolatingFunction[Domain: (0. 1.6) Output: scalar]}

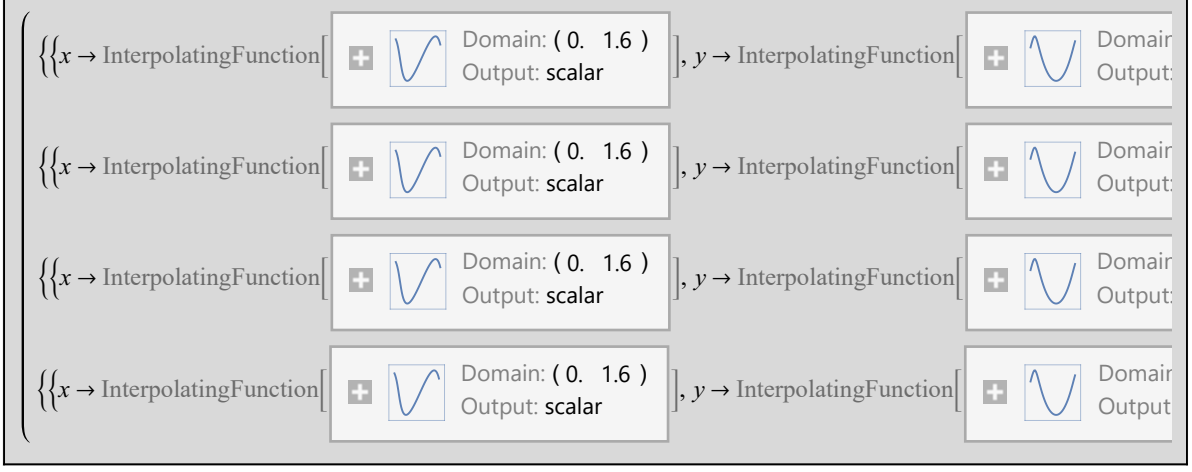
In[13]:=
```
% - orbitBIS
```

Out[13]=



In[14]:=
```
orbitBIS = MapThread[{#1, #2, #3} &, {orbit, {Blue, Red, Pink, Green}, metody}]
```
zastosuj w wątku | niebi… | cz… | różowy | zielony

Out[14]=



Albo

In[15]:=

```
Transpose[{orbit, {Blue, Red, Pink, Green}, metody}]
transpozycja        niebi··· cz··· różowy zielony
orbitBIS - %
```

Out[15]=

```
{{x → InterpolatingFunction[ Domain: (0. 1.6) Output: scalar ], y → InterpolatingFunction[ Domain Output:
{{x → InterpolatingFunction[ Domain: (0. 1.6) Output: scalar ], y → InterpolatingFunction[ Domain Output:
{{x → InterpolatingFunction[ Domain: (0. 1.6) Output: scalar ], y → InterpolatingFunction[ Domain Output:
{{x → InterpolatingFunction[ Domain: (0. 1.6) Output: scalar ], y → InterpolatingFunction[ Domain Output
```

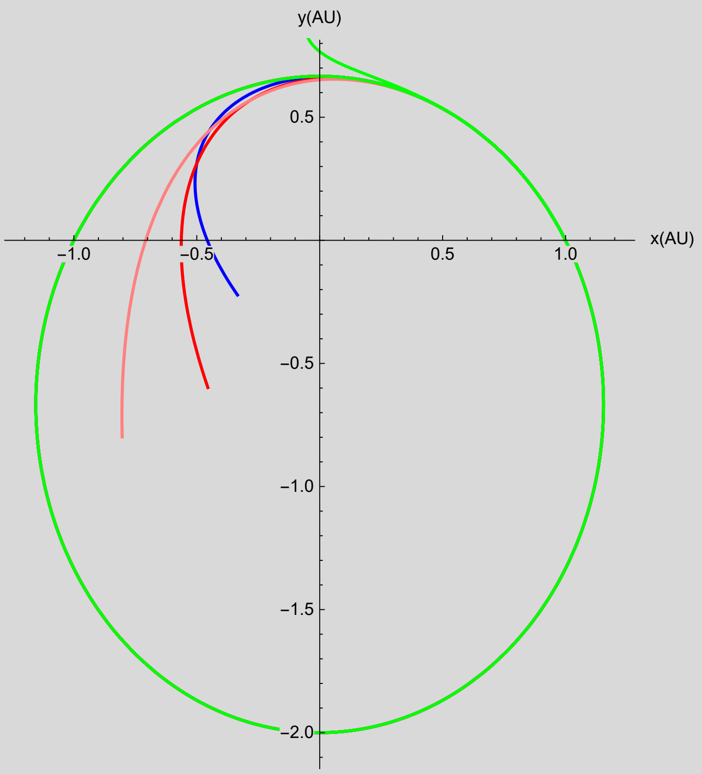
Out[16]=

$$\begin{pmatrix} (0 & 0) & 0 & 0 \\ (0 & 0) & 0 & 0 \\ (0 & 0) & 0 & 0 \\ (0 & 0) & 0 & 0 \end{pmatrix}$$

In[17]:=

```
Show@(ParametricPlot[Evaluate[{x[t], y[t]} /. First@#],
pokaż  wykres parametry···  oblicz         pierwszy
    {t, 0, 1.8}, AspectRatio → Automatic, AxesLabel → {"x(AU)", "y(AU)"},
                 format obrazu  automatyczny  oznaczenia osi
    PlotStyle → #[[2]], ImageSize → Medium] & /@ orbitBIS)
    styl grafiki       rozmiar obrazu  średnia wielkość
```

Out[17]=

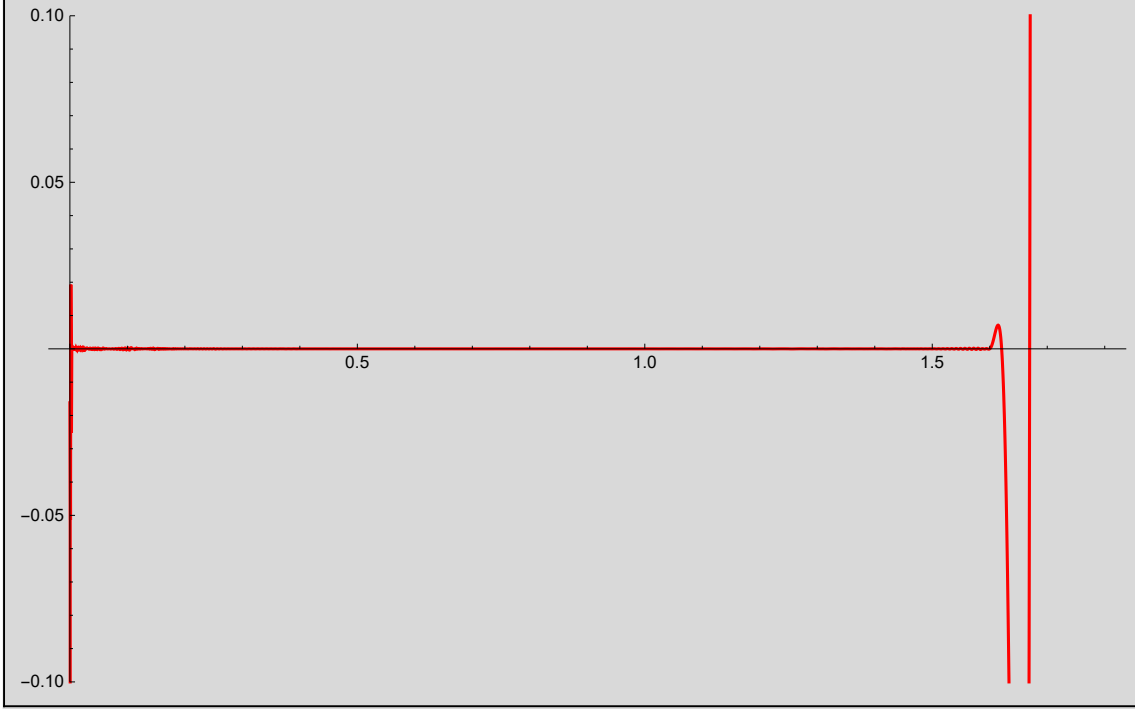


In[18]:=
```
GraphicsRow@
 rząd z grafikami
  (Plot[{row1, row2} /. First@#, {t, 0, 1.8}, PlotRange → {-0.001, 0.001},
   wykres                pierwszy              zakres wykresu
      ImageSize → Tiny, PlotLabel → Last@#] & /@ orbitBIS)
       rozmiar obrazu  malutki  etykieta grafiki  ostatni
```

Out[18]=

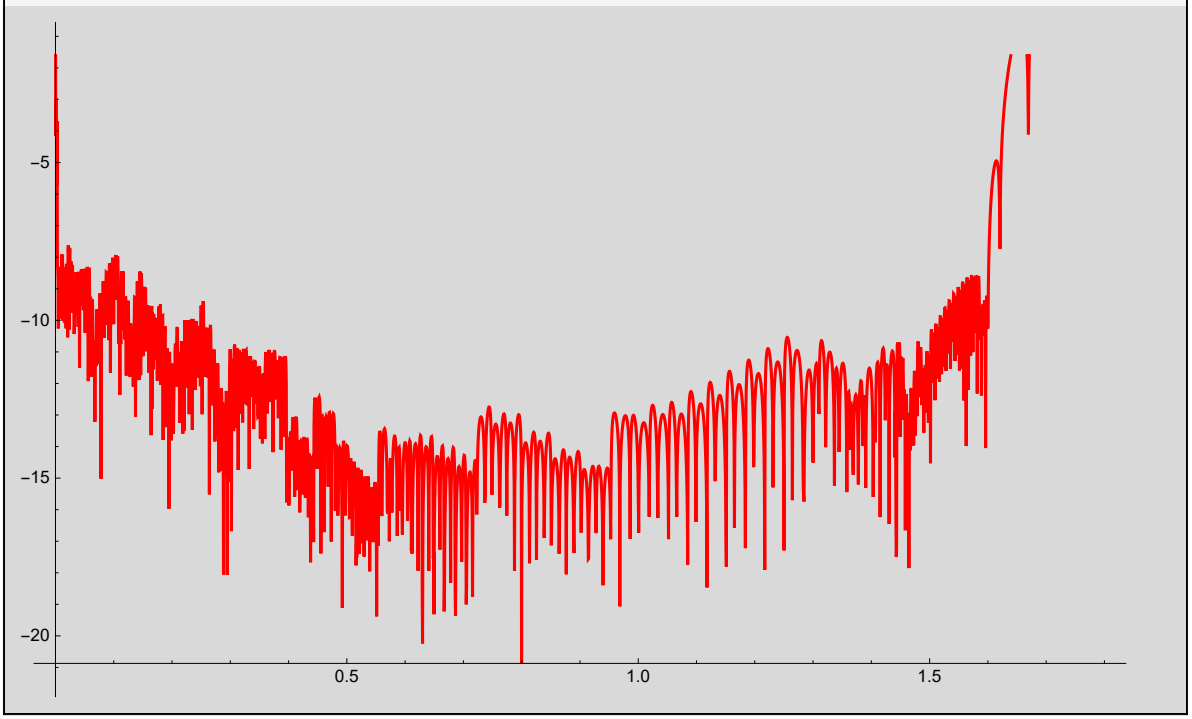
In[19]:=
```
Plot[row1 /. orbit[[1]], {t, 0, 1.8},
 wykres
  PlotRange → {-0.1, 0.1}, ImageSize → Large, PlotStyle → Red]
   zakres wykresu          rozmiar obrazu  duży  styl grafiki  czerw
```

Out[19]=



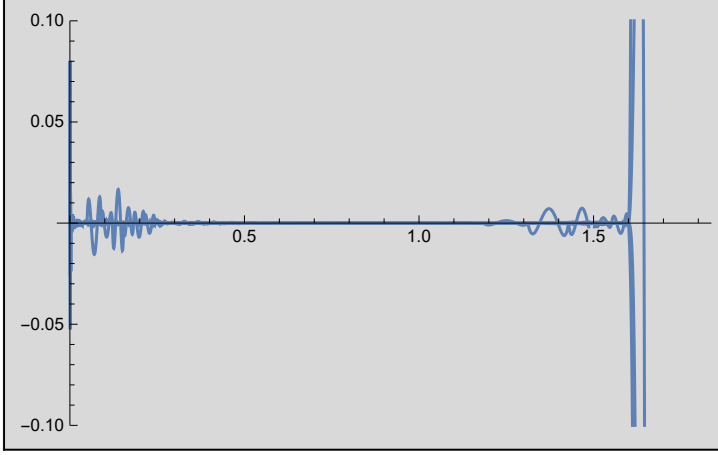
In[20]:=
```
Plot[{Log[Abs[row1]]} /. orbit[[1]], {t, 0, 1.8}, ImageSize → Large, PlotStyle → Red]
```
wykres | lo··· | wartość bezwzględna | rozmiar obrazu | duży | styl grafiki | czerw

Out[20]=



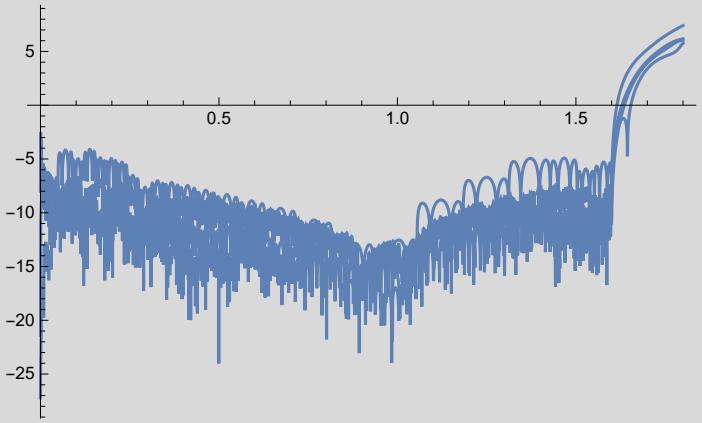
In[21]:=
```
Plot[{row2} /. orbit, {t, 0, 1.8}, PlotRange → {-0.1, 0.1}]
```
wykres | zakres wykresu

Out[21]=



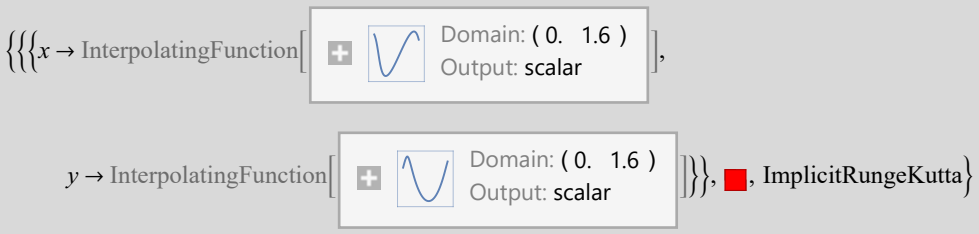
In[22]:=
```
Plot[{Log[Abs[row2]]} /. orbit, {t, 0, 1.8}]
```
wykres | lo··· | wartość bezwzględna

Out[22]=



In[23]:=
```
orbitBIS[[2]]
```

Out[23]=

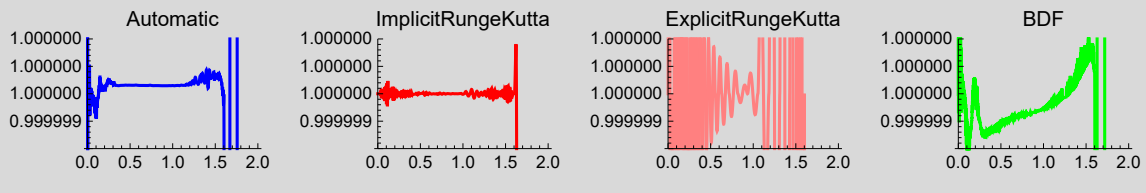


Zachowanie energii

In[24]:=
```
Energy[t_] := (x'[t]^2 + y'[t]^2) / 2 - (4 π^2) / (x[t]^2 + y[t]^2)^(1 / 2)
```

In[25]:=
```
Plot[Evaluate[Energy[t] / Energy[0] /. First@#], {t, 0, 2}, PlotStyle → #[[2]],
    PlotLabel → Last@#, PlotRange → {{0.999999, 1.000001}}] & /@ orbitBIS;
GraphicsRow[%, ImageSize -> Full]
```
wyk··· | oblicz | pierwszy | styl grafiki | etykieta grafiki | ostatni | zakres wykresu | rząd z grafikami | rozmiar obrazu | kompletny

Out[26]=

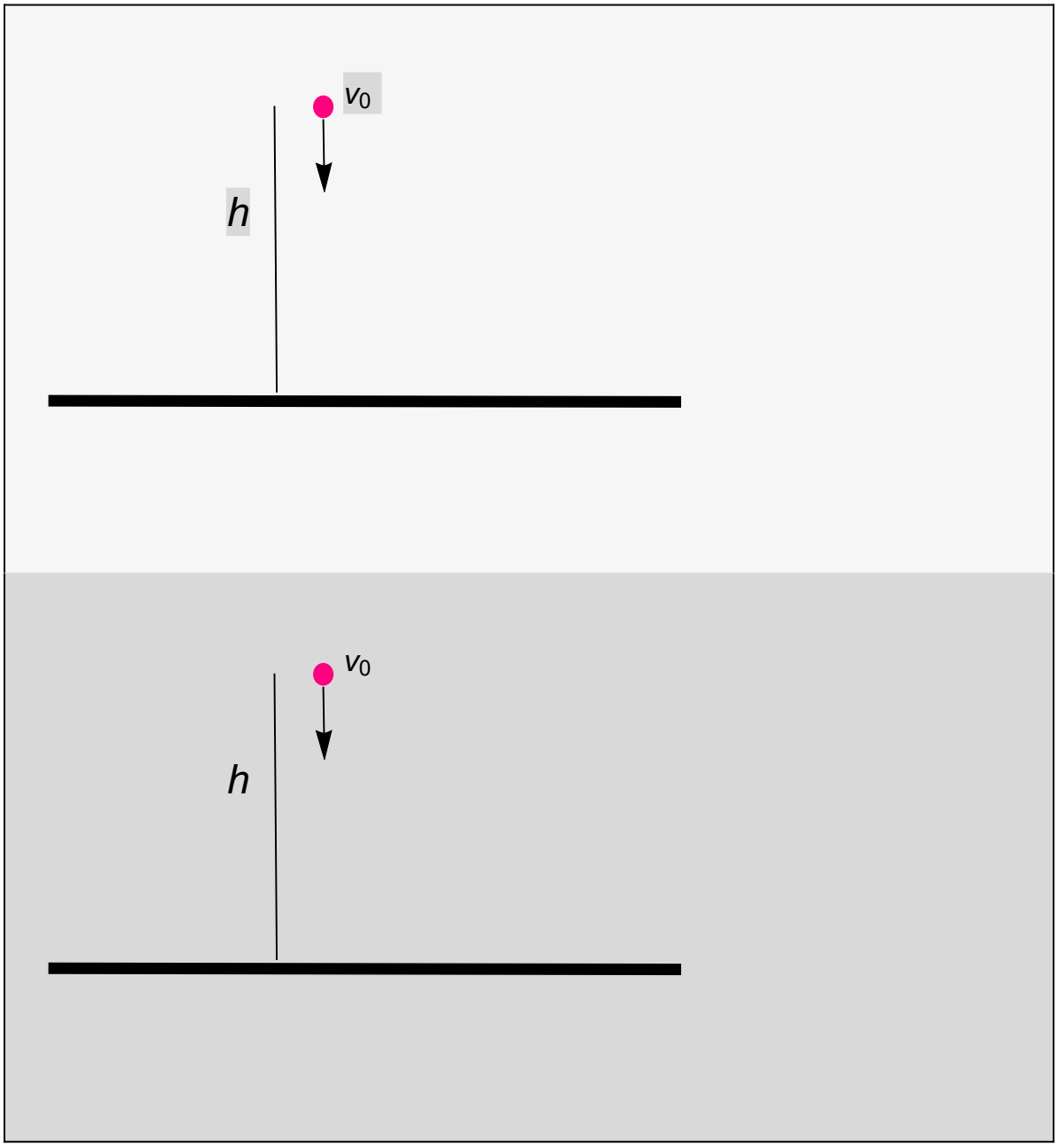


In[27]:=
```
Clear[orbit]
```
wyczyść

Spadek swobodny z oporem

F=-m g- b v

In[28]:=



Out[28]=

Równania ruchu oraz rozwiązanie

UWAGA, zamiast rozwiązywać równanie różniczkowe drugiego rzędu, możemy rozwiązać układ równań pierwszego rzędu.

In[29]:=
```
Clear[sol, v, y]
sol = Flatten@DSolve[{
    y'[t] == v[t],
    v'[t] == -g - b v[t] / m,
    y[0] == h,
    v[0] == v0
   }, {y[t], v[t]}, t] // FullSimplify
```

Out[30]=
$$\left\{v(t) \to \frac{e^{-\frac{b t}{m}} (b\, v0 + g\, m) - g\, m}{b},\ y(t) \to \frac{1}{b^2}\left(b^2 h - m\, e^{-\frac{b t}{m}} (b\, v0 + g\, m) - b\, g\, m\, t + b\, m\, v0 + g\, m^2\right)\right\}$$

Prędkość graniczna (krytyczna) – maksymalna prędkość obiektu bezwładnie spadającego, w ośrodku niebędącym próżnią, przy której siła grawitacji (skierowana w dół) jest równoważona przez siłę oporu aerodynamicznego (skierowaną w górę). W wyniku równoważenia się sił obiekt porusza się (spada) ruchem jednostajnym.

In[31]:=
```
Limit[First[sol][[2]], t → Infinity, Assumptions → {b > 0, m > 0}]
```

Out[31]=
$$-\frac{g\, m}{b}$$

In[32]:=
```
(* Podstawmy za bt/m → x*)
Last[First[sol]]
Limit[%, t → Infinity, Assumptions → {b > 0, m > 0}]
vt = Simplify[Abs[%], {b > 0, g > 0, m > 0}]
```

Out[32]=
$$\frac{e^{-\frac{b t}{m}} (b\, v0 + g\, m) - g\, m}{b}$$

Out[33]=
$$-\frac{g\, m}{b}$$

Out[34]=
$$\frac{g\, m}{b}$$

Używamy jednostek w czasu [m/b], natomiast dla położenia $\left[gm^2/b^2\right]$ oraz dla prędkość [gm/b]

In[35]:=
```
Expand[v[t] /. sol]
```
rozszerz
```
% /. {g m / b → 1}
% /. {b → m}
```

Out[35]= $\frac{g\,m\,e^{-\frac{b\,t}{m}}}{b} - \frac{g\,m}{b} + \text{v0}\,e^{-\frac{b\,t}{m}}$

Out[36]= $\text{v0}\,e^{-\frac{b\,t}{m}} + e^{-\frac{b\,t}{m}} - 1$

Out[37]= $e^{-t}\,\text{v0} + e^{-t} - 1$

In[38]:=
```
SetDelayed[xxxxxx, 2]
```
przypisz z opóźnieniem

In[39]:=
```
? xxxxxx
```

```
Global`xxxxxx
```

xxxxxx := 2

In[40]:=
```
Set[xxxx, 2]
```
przypisz

Out[40]= 2

In[41]:=
```
? xxxx
```

```
Global`xxxx
```

xxxx = 2

In[42]:=
```
V[t_, v0_] = (Expand[v[t] /. sol] /. {g m / b → 1}) /. {b → m}
```
rozszerz

Out[42]= $e^{-t}\,\text{v0} + e^{-t} - 1$

In[43]:=
```
V[t, -3]
```

Out[43]= $-2\,e^{-t} - 1$

Zauważmy, że wówczas prędkość graniczna wyniesie:

In[44]:=
```
vt
% /. {g m / b → 1}
% /. {b → m}
```

Out[44]=
$$\frac{g\,m}{b}$$

Out[45]=
$$1$$

Out[46]=
$$1$$

In[47]:=
```
ListaPrędkości = {Abs[V[t, 1.2]], Abs[V[t, -3]], Abs[V[t, -0.6]], Abs[V[t, 0]], 1}
                  wartość bezwzględna wartość bezwzględ… wartość bezwzględna wartość bezwzględna
StyleRysownia = {{Pink, Dotted}, {Red}, {}, {Blue}, {Dashing[{0.02}]}}
                  różowy wykropko… czerwony niebieski specyfikacja linii przerywanej
```

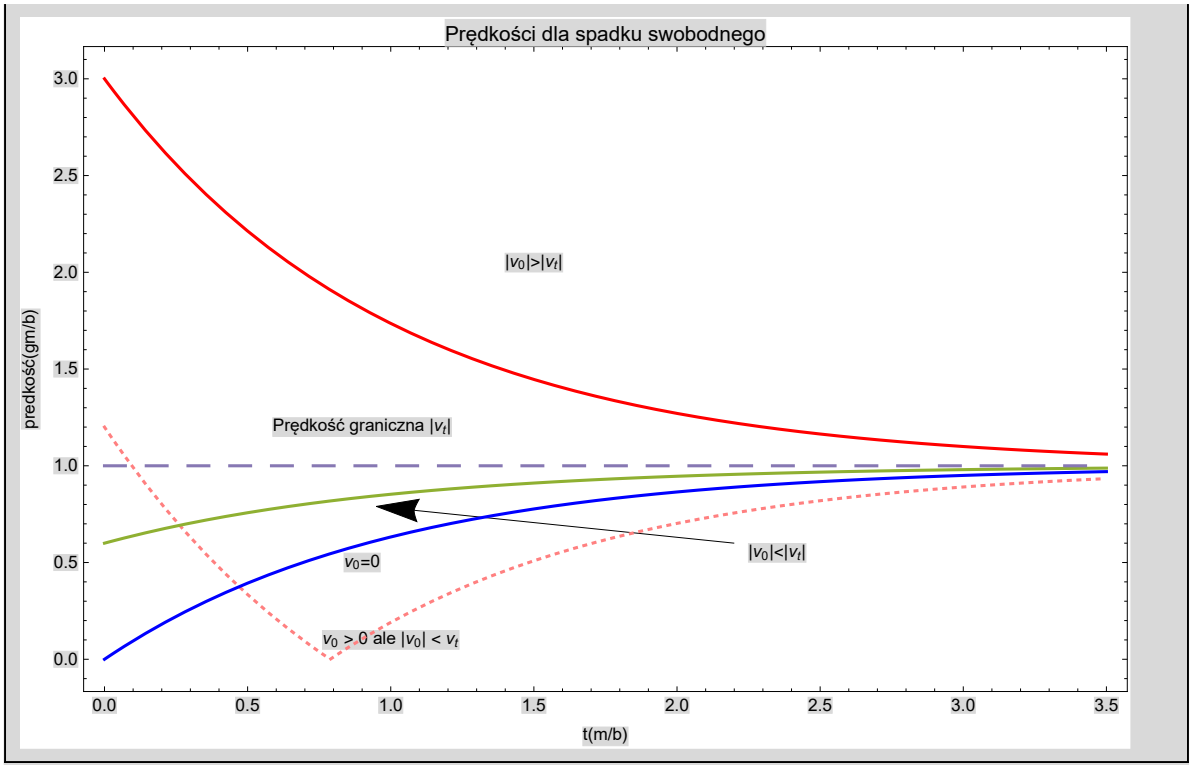
Out[47]=
$$\{|-1+2.2\,e^{-t}|,\ |-1-2\,e^{-t}|,\ |-1+0.4\,e^{-t}|,\ |-1+e^{-t}|,\ 1\}$$

Out[48]=
$$\{\{\blacksquare, \text{Dashing}[\{0, \text{Small}\}]\}, \{\blacksquare\}, \{\}, \{\blacksquare\}, \{\text{Dashing}[\{0.02\}]\}\}$$

In[49]:=

```
Plot[
 wykres
 Evaluate[ListaPrędkości],
  oblicz
 {t, 0, 3.5}, PlotRange → Full,
                zakres wykresu  kompletny
 Frame → True,
  ramka    prawda
 FrameLabel → {"t(m/b)", "predkość(gm/b)"},
  etykieta ramki
 PlotStyle → StyleRysownia,
  styl grafiki
 Prolog → {
  tło grafiki
   Text["v₀=0", {0.9, 0.5}],
    tekst
   Text["v₀ > 0 ale |v₀| < vₜ", {1.0, 0.1}],
    tekst
   Text["|v₀|<|vₜ|", {2.35, 0.55}],
    tekst
   Text["|v₀|>|vₜ|", {1.5, 2.05}],
    tekst
   Text["Prędkość graniczna |vₜ|", {0.9, 1.2}],
    tekst
   Arrow[{{2.2, 0.6}, {0.95, 0.79}}]
    strzałka
  },
 ImageSize → Large,
  rozmiar···      duży
 Background → White,
  tło             biały
 PlotLabel → "Prędkości dla spadku swobodnego"
  etykieta grafiki
]
```

Out[49]=



Używamy jednostek w czas w m/b, położenie gm^2/b^2 oraz prędkość gm/b

Block and Module

In[50]:=

```
? Module
```

Module[{*x*, *y*, …}, *expr*] specifies that occurrences of the symbols *x*, *y*, … in *expr* should be treated as local.
Module[{*x* = x_0, …}, *expr*] defines initial values for *x*, …. ≫

What Module[vars,body] does is to treat the form of the expression body at the time when the module is executed as the “code” of a Wolfram Language program. It examines the expression, and when any of the vars explicitly appears in this “code”, it is considered to be local. Normal evaluation then proceeds.

In[51]:=

```
? Block
```

Block[{*x*, *y*, …}, *expr*] specifies that *expr* is to be evaluated with local values for the symbols *x*, *y*, ….
Block[{*x* = x_0, …}, *expr*] defines initial local values for *x*, …. ≫

Block[vars,body] does not look at the form of the expression body. Instead, it records the current value of each of the vars. Throughout the evaluation of body, the block uses local values for the vars, and then restores their original values once the evaluation of body is finished.

In[52]:=

```
Clear[i, m, a]
```

wyczyść

In[53]:=
```
m = i^2
```
Out[53]= i^2

In[54]:=
```
Block[{i = a}, i + m]
```
blok

Out[54]= $a^2 + a$

In[55]:=
```
Module[{i = a}, i + m]
```
moduł

Out[55]= $a + i^2$

Naiwne rozwiązywanie równań różniczkowych

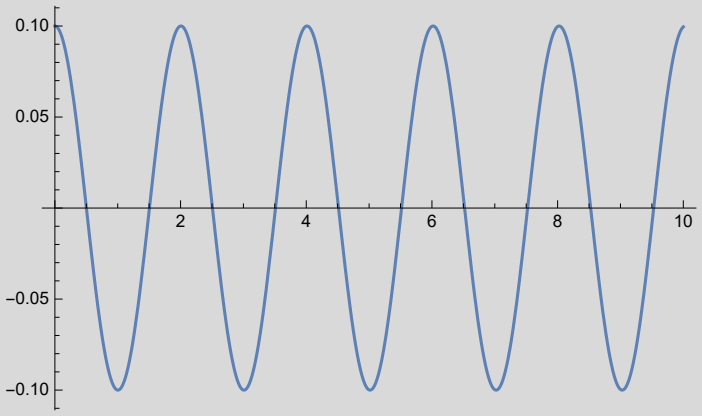
Wahadło Matematyczne

In[56]:=
```
solM = Flatten@DSolve[{θ''[t] + 9.81 θ[t] == 0, θ'[0] == 0, θ[0] == Phi0}, θ[t], t]
```
spłaszcz rozwiązywanie równań różniczkowych

Out[56]= $\{\theta(t) \to 1.\,\text{Phi0}\cos(3.13209\,t)\}$

In[57]:=
```
wahadloScisle = Plot[Evaluate[θ[t] /. solM /. Phi0 → 0.1], {t, 0, 10}]
```
wyk··· oblicz

Out[57]=



Mamy równania

θ''[t] + 9.81 θ[t] = 0,
θ'[0] = 0
θ[0] = Phi0

Podstawiamy:

$\theta'(t) = (\theta(t+h) - \theta(t))/h$

$$\theta''(t) = (\theta(t+h) - 2\,\theta(t) + \theta(t-h))/h^2$$

Równanie rekurencyjne:

θ_{i+1}-2 θ_i + θ_{i-1}== −9.81 h^2 θ_i

θ_{i+1}=−(9.81 h^2 + 2)θ_i-θ_{i-1}
θ_1=θ_0

In[58]:=
```
{θ''[t] + 9.81 θ[t] == 0, θ'[0] == 0, θ[0] == Phi0} /.
 θ''[t] → (θ[i + 1] - 2 θ[i] + θ[i - 1]) / h^2
% /. θ[t] → θ[i]
Solve[First[%], θ[i + 1]] // Simplify
```
rozwi… pierwszy uprość

Out[58]=
$$\left\{\frac{\theta(i-1) - 2\,\theta(i) + \theta(i+1)}{h^2} + 9.81\,\theta(t) = 0,\ \theta'(0) = 0,\ \theta(0) = \text{Phi0}\right\}$$

Out[59]=
$$\left\{\frac{\theta(i-1) - 2\,\theta(i) + \theta(i+1)}{h^2} + 9.81\,\theta(i) = 0,\ \theta'(0) = 0,\ \theta(0) = \text{Phi0}\right\}$$

Out[60]=
$$\left\{\left\{\theta(i+1) \to \left(2. - 9.81\,h^2\right)\theta(i) - 1.\,\theta(i-1)\right\}\right\}$$

In[61]:=
```
ConstantArray[0, 10]
```
stała tablica

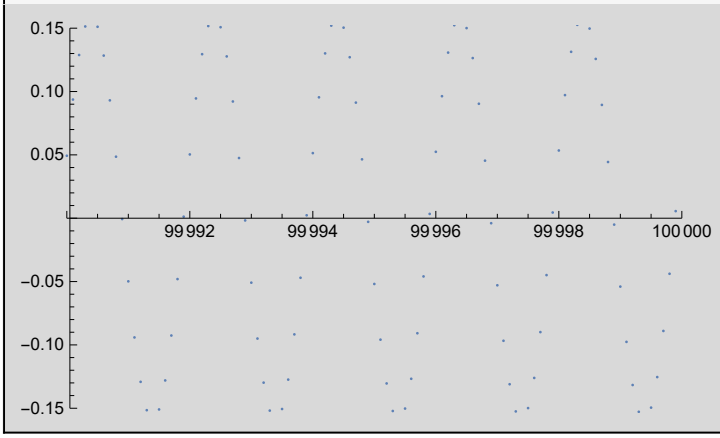
Out[61]=
$$\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

In[62]:=
```
Clear[Rozwiaz]
wyczyść
Rozwiaz[n_, h_, θ0_, θP0_] := Module[{θ, tab},
                                moduł
  θ = ConstantArray[0, n];
      stała tablica
  θ[[1]] = θ0;
  θ[[2]] = θ0;

  Do[θ[[k]] = (-9.81 h^2 + 2.) θ[[k - 1]] - θ[[k - 2]], {k, 3, n}];
  rób
  tab = Table[h (k - 1), {k, 1, n}];
        tabela
  (* Transpose[{θ,tab}];
     transpozycja
  Print[{θ,tab}];*)
  drukuj
  Return[Transpose[{tab, θ}]]
  zwróć  transpozycja
 ]
```

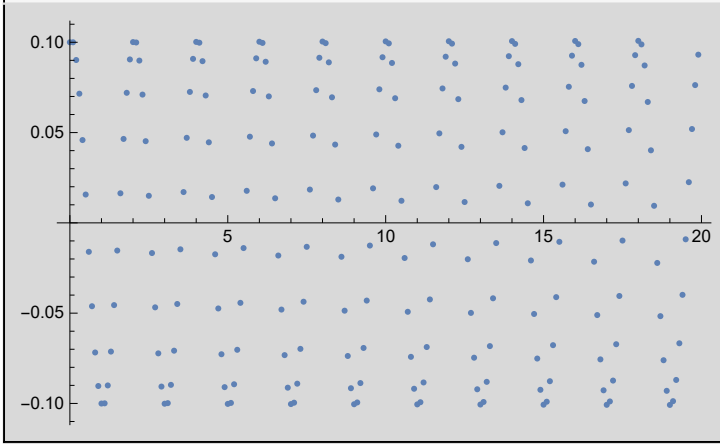
In[64]:=
```
Rozwiaz[100, 0.00001, N[Pi / 20], 0];
                      pi
whadloMoje = ListPlot[Rozwiaz[1 000 000, 0.1, N[Pi / 20], 0],
               wykres danych z listy          pi
  PlotRange → {{99 990, 100 000}, {-.15, 0.15}}]
  zakres wykresu
```

Out[65]=

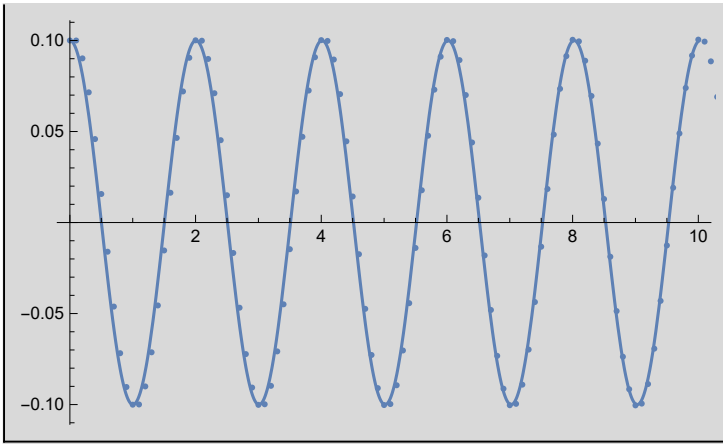


In[66]:=
```
ListPlot[Rozwiaz[200, 0.1, 0.1, 0]]
wykres danych z listy
Show[wahadloScisle, %]
pokaż
```

Out[66]=



Out[67]=



Nie używaj loopów jeśli nie musisz

In[68]:=
```
sum =.
sum = 0
D
```
oblicz pochodną

Out[69]= 0

Out[70]= D

In[71]:=
```
? Do
```

Do[*expr*, *n*] evaluates *expr* *n* times.
Do[*expr*, {*i*, i_{max}}] evaluates *expr* with the variable *i* successively taking on the values 1 through i_{max} (in steps of 1).
Do[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Do[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Do[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Do[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] evaluates *expr* looping over different values of *j* etc. for each *i*. ≫

In[72]:=
```
? For
```

For[*start*, *test*, *incr*, *body*] executes *start*, then repeatedly evaluates *body* and *incr* until *test* fails to give True. ≫

In[73]:=
```
Clear[sum]
```
wyczyść

In[74]:=
```
Clear[sum]
```
wyczyść
```
sum = 0
Timing[For[n = 1, n < 1 000 001, n++, sum = sum + n]]
```
czas u··· dla

Out[75]= 0

Out[76]= {1.26563, Null}

In[77]:=
```
Timing[Plus @@ Range[1 000 001]]
```
czas u··· suma zakres

Out[77]= {0.1875, 500 001 500 001}

In[78]:=
```
Timing[Total@Range[1 000 001]]
```
czas u··· oblicz··· zakres

Out[78]= {0.015625, 500 001 500 001}