Równania różniczkowe zwyczajne

Zagadnienie: znajdź funkcję x(t)

$\frac{dx}{dt} = F(x, t)$
$x(t_0) = x_0$

Metoda naiwna

$\frac{dx}{dt} = \frac{x(t+h) - x(t)}{h} = F(x, t) \Rightarrow x(t + h) = x(t) + h F(x, t)$

Szereg Taylora na ratunek

Przypomnienie: wzór Taylora: x(t+h) = $\sum_{n=0}^{\infty} \frac{1}{n!} x^{(n)}(t) h^n$

Znając wartość funkcji x w czasie t, możemy wyznaczyć jej wartość w punkcie t+h, jeśli znamy pochodne F(x,t)

- Dla problemu: $\frac{dx}{dt} = F(x, t)$, $x(t_0) = x_0$ znamy pierwszą pochodną!
- Metoda Eulera: tylko wyraz liniowy w h: $x(t+h) = x(t) + x'(t) h = x(t) + F(x, t) h$
- Metoda Eulera wyższego rzędu:
  $x(t+h) = x(t) + F(x, t) h + \frac{1}{2} F'(x, t) h^2 + \frac{1}{6} F''(x, t) h^3$

Metoda Euler'a

⋮

- Przykład: Rozwiąż równanie metodą Eulera (liniową): $\frac{dx}{dt} = x(t)$, $x(t=0) = x_0 = 2$, $h = 10^{-2}$
  $x(t + h) = x(t) + x(t)\, h$

In[∘]:=
```
Clear[x]
sol = DSolveValue[{x'[t] == x[t], x[0] == 2}, x, t]
```

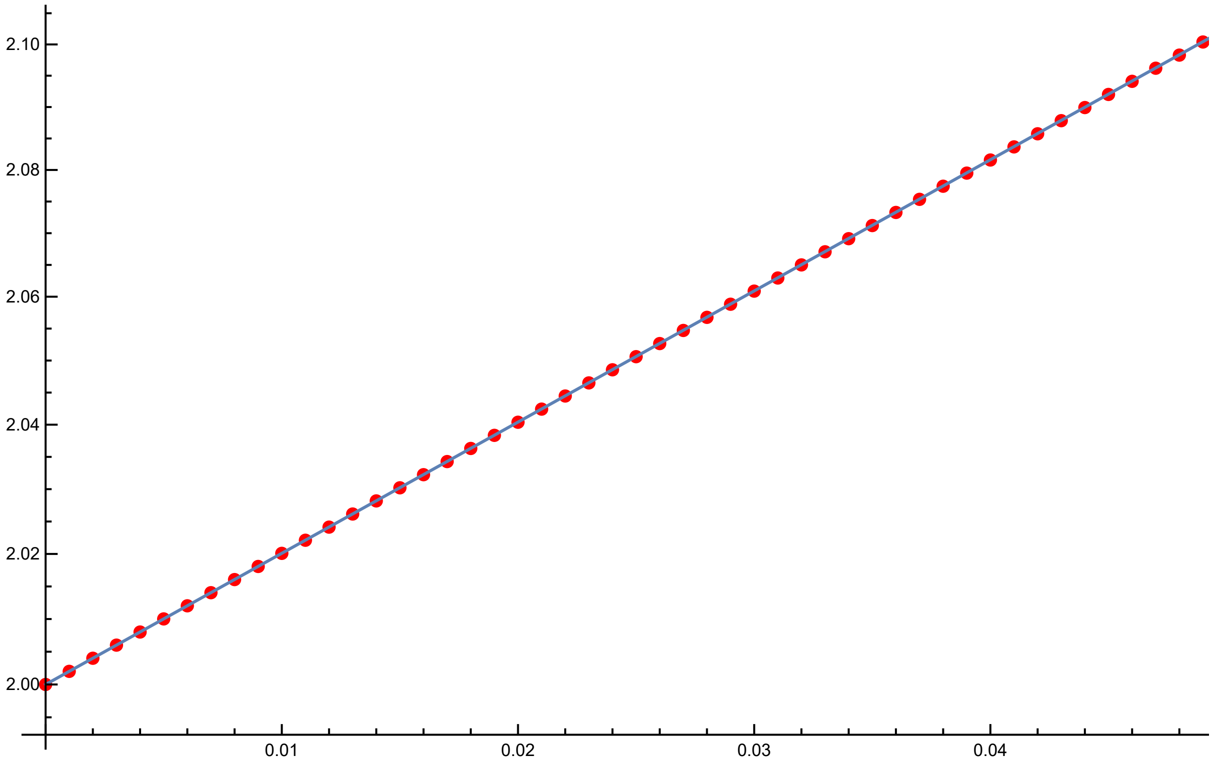
Out[∘]=
```
Function[{t}, 2 e^t]
```

In[∘]:=
```
x = {{0, 2}};
h = 10^-3
For[i = 2, i < 2000, i++,
 AppendTo[x, {N[x[[i - 1, 1]] + h], N[x[[i - 1, 2]] + x[[i - 1, 2]] h]}]
]
Show[{
  ListLogPlot[x[[ ;; 50]], PlotStyle → Red],
  LogPlot[{sol[t]}, {t, 0, 2}]}]
Show[{
  ListLogPlot[x, PlotStyle → Red],
  LogPlot[{sol[t]}, {t, 0, 20}]}]
```

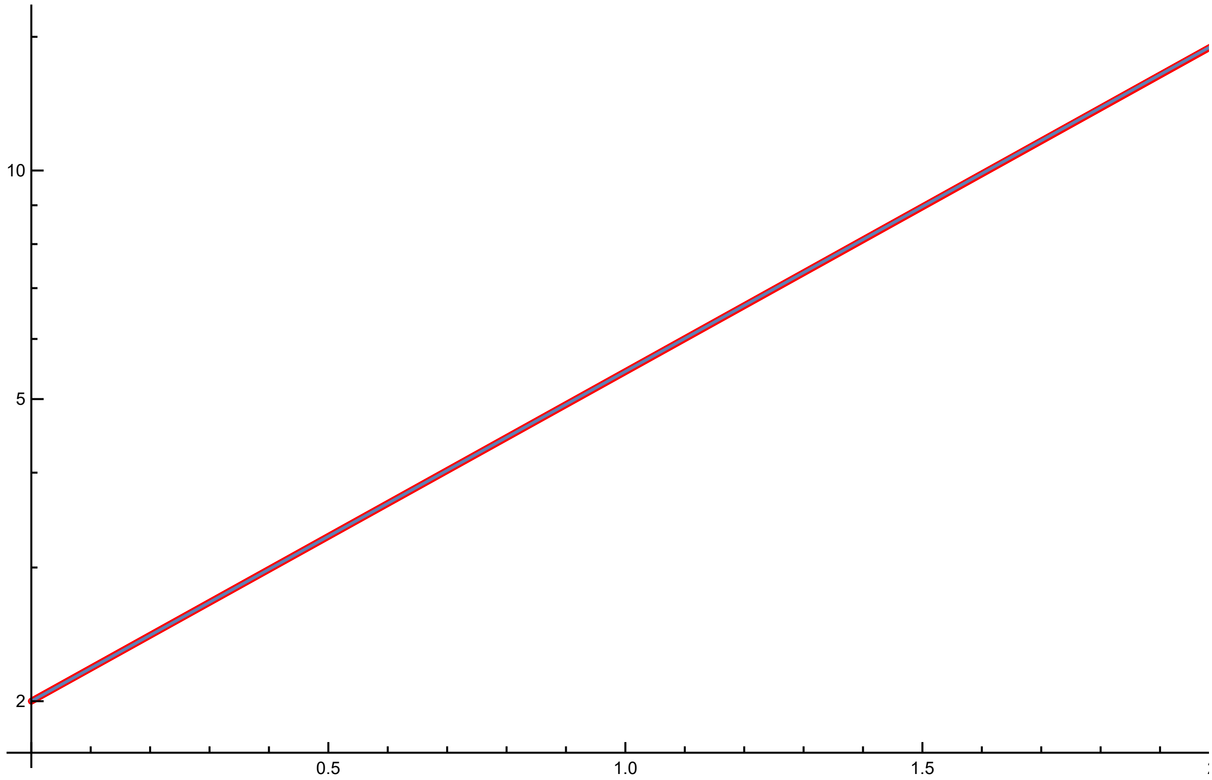
Out[∘]=

$\frac{1}{1000}$

Out[∘]=



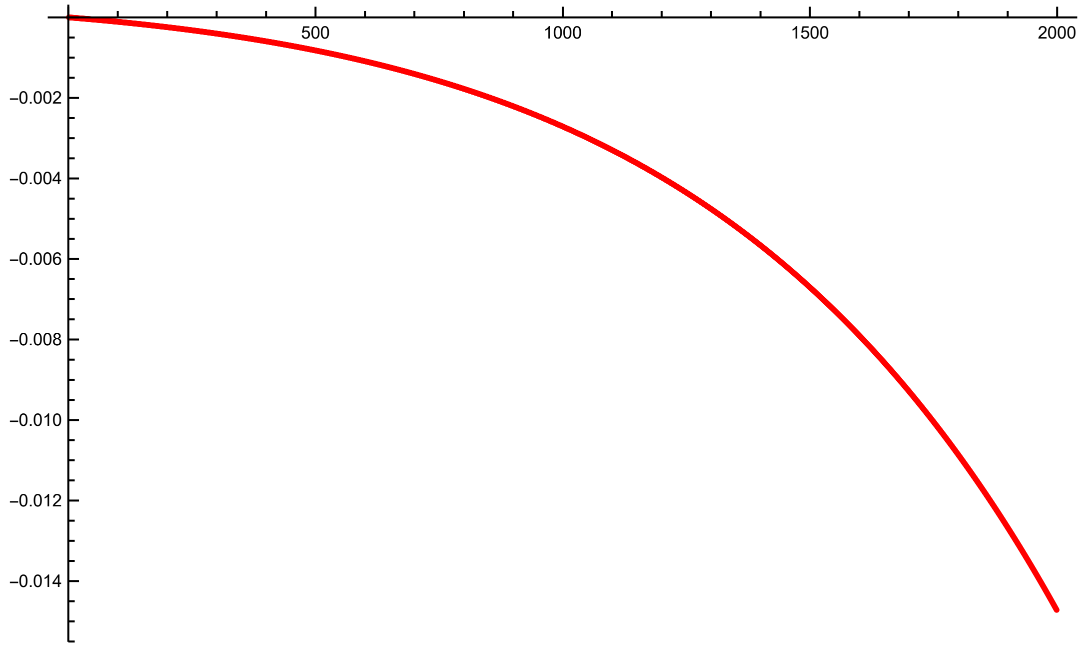
Out[ ]=



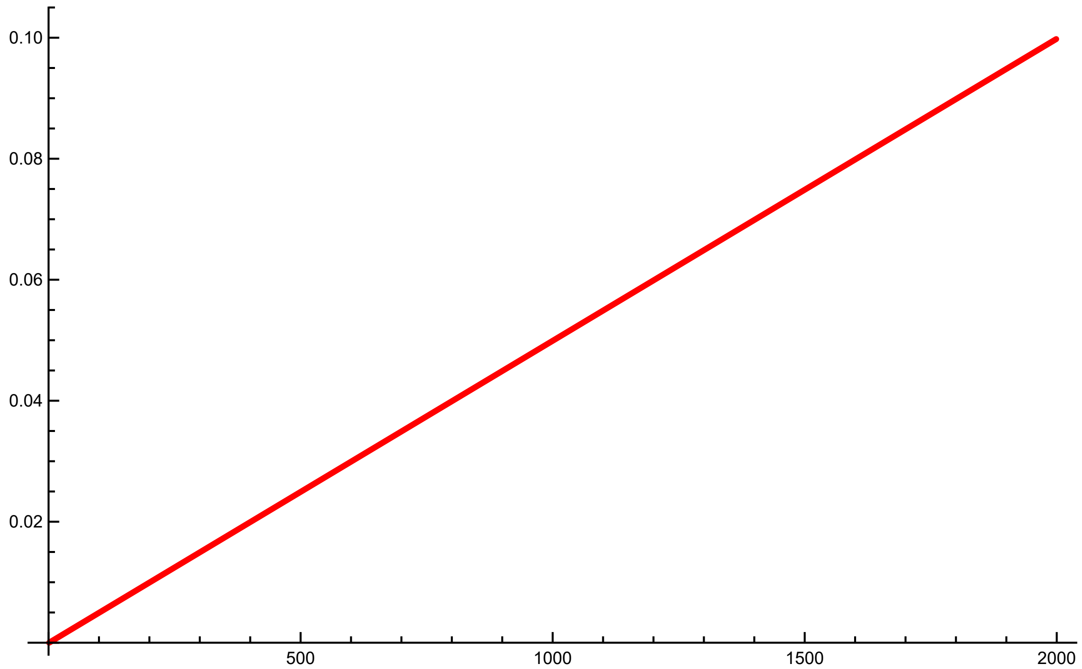
In[ ]:=

```
exact = Table[{t0, N@sol[t0]}, {t0, 0, x[[-1, 1]] + h, h}];
Show[{
  ListPlot[(x[[All, 2]] - exact[[All, 2]]), PlotStyle → Red]}]
Show[{
  ListPlot[Abs[(x[[All, 2]] - exact[[All, 2]])/exact[[All, 2]]] * 100, PlotStyle → Red]}]
```

Out[ ]=



Out[ ]=



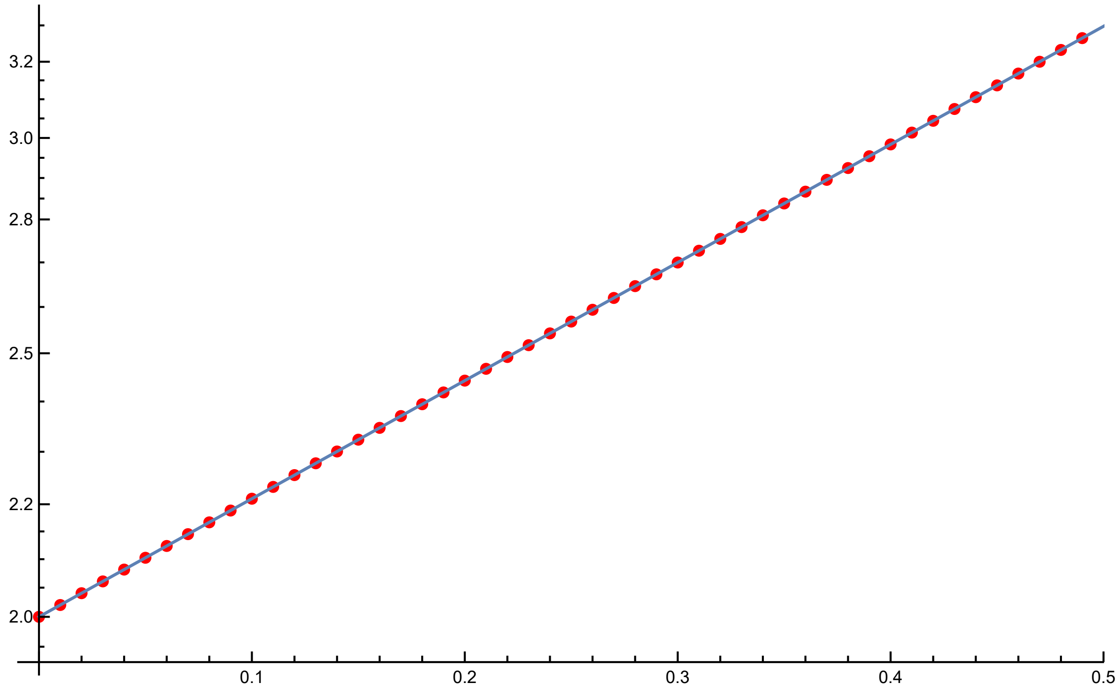
- Metoda wyższego rzędu:
  $x(t+h) = x(t) + x(t)\,h + x(t)\,\frac{h^2}{2} + x(t)\,\frac{h^3}{6}$

In[◦]:=
```
x = {{0, 2}};
h = 10^-2
For[i = 2, i < 2000, i++,
 AppendTo[x, {
   N[x⟦i - 1, 1⟧ + h],
   N[x⟦i - 1, 2⟧ + x⟦i - 1, 2⟧ h + 1/2 x⟦i - 1, 2⟧ h^2 + 1/6 x⟦i - 1, 2⟧ h^3]}]
]
Show[{
  ListLogPlot[x⟦ ;; 50⟧, PlotStyle → Red],
  LogPlot[{sol[t]}, {t, 0, 2}]}, ImageSize → Large]
Show[{
  ListLogPlot[x⟦ ;; 200⟧, PlotStyle → Red],
  LogPlot[{sol[t]}, {t, 0, 2}]}, ImageSize → Large]
```

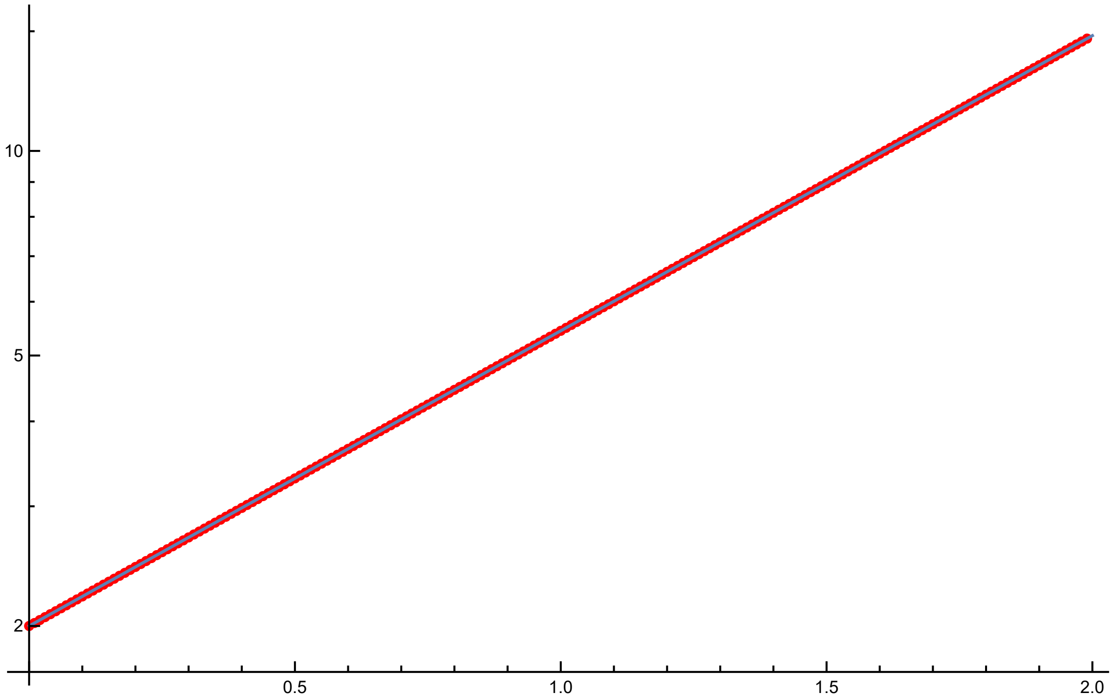
Out[◦]=

$$\frac{1}{100}$$

Out[◦]=

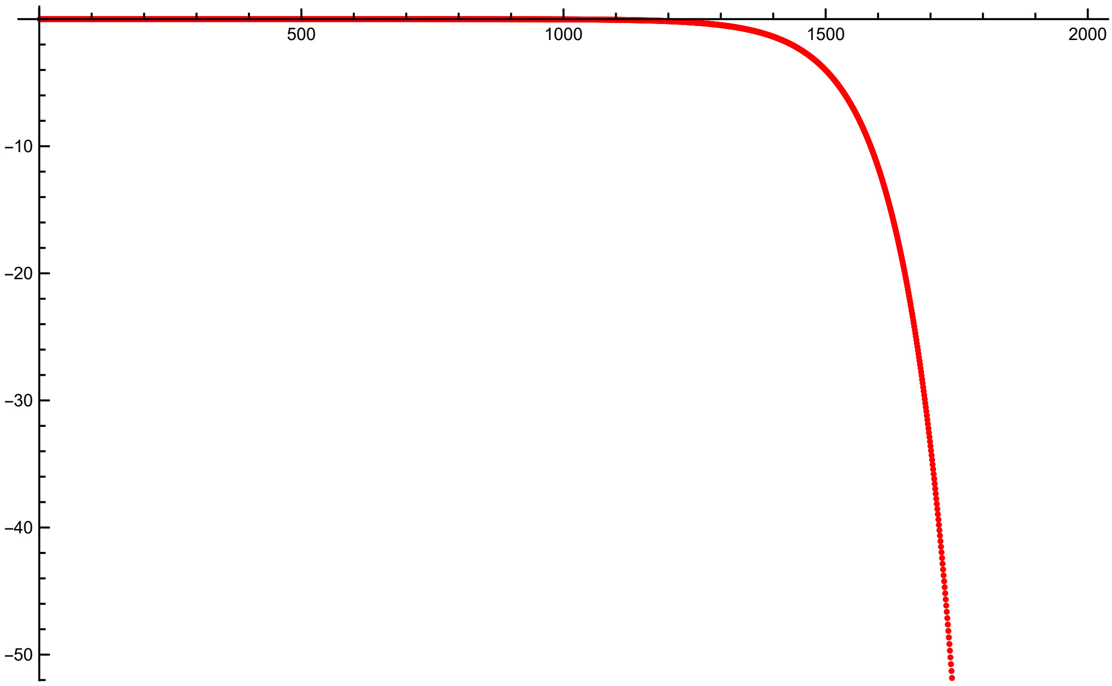


Out[ ]=

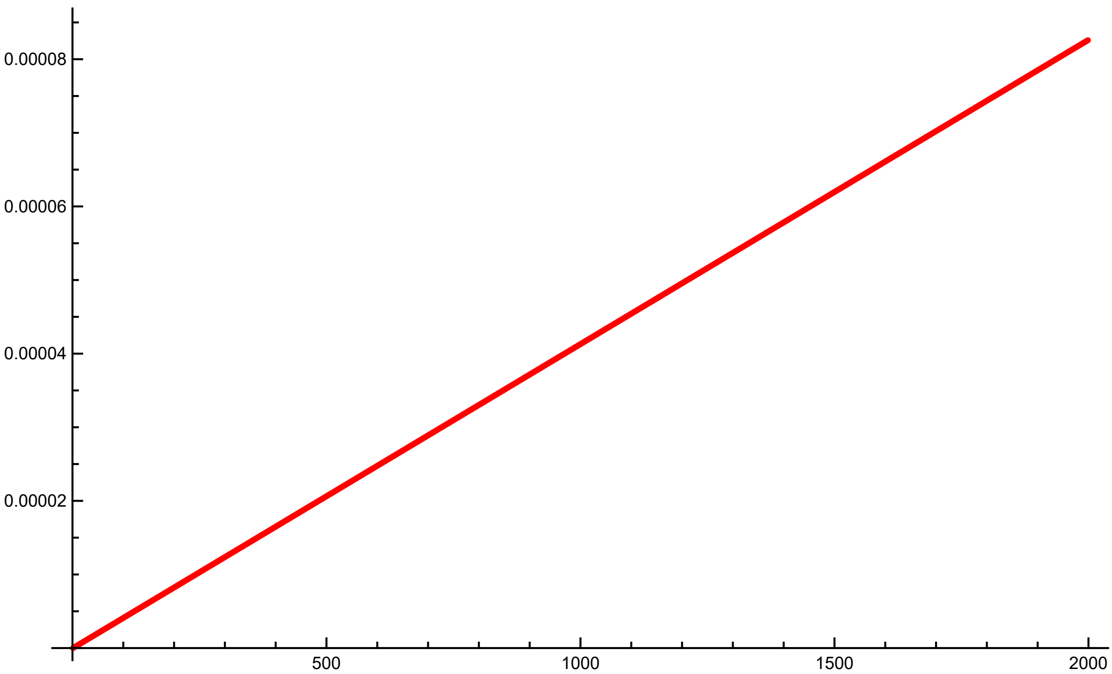


In[∘]:=
```
exact = Table[{t0, N@sol[t0]}, {t0, 0, x[[-1, 1]], h}];
Show[{
  ListPlot[(x[[All, 2]] - exact[[All, 2]]), PlotStyle → Red]}, ImageSize → Large]
Show[{
  ListPlot[Abs[(x[[All, 2]] - exact[[All, 2]])/exact[[All, 2]]] * 100, PlotStyle → Red]}, ImageSize → Large]
```

Out[∘]=



Out[∘]=



- Przykład: Rozwiąż równanie metodą Eulera (liniową): $\frac{dx}{dt} = x(t) * \cos(x(t))$, $x(t = 0) = x_0 = 1$
  $x(t + h) = x(t) + x(t) \cos(x(t))\, h$

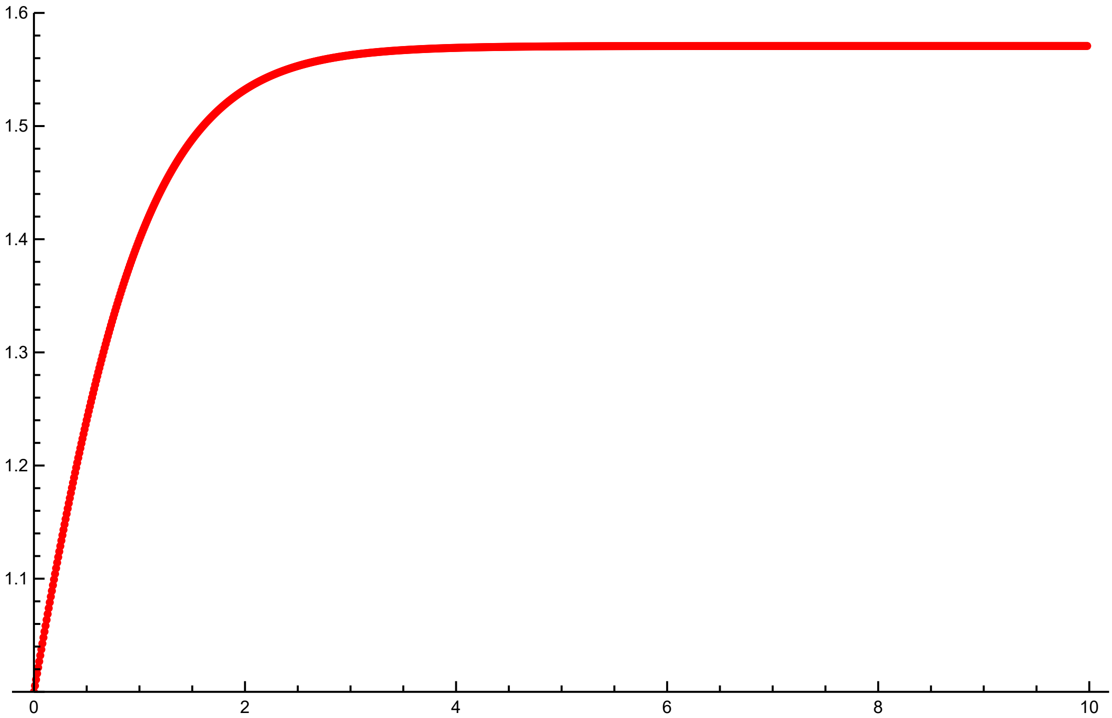
In[∘]:=
```
Simplify[D[y[t] × Cos[y[t]], t] /. {y'[t] → y[t] × Cos[y[t]]}]
```

Out[∘]=
```
Cos[y[t]] × y[t] (Cos[y[t]] - Sin[y[t]] × y[t])
```

In[∘]:=
```
x = {{0, 1}};
h = 10^-2
For[i = 2, i < 1000, i++,
 AppendTo[x, {
   N[x⟦i - 1, 1⟧ + h],
   N[x⟦i - 1, 2⟧ + x⟦i - 1, 2⟧ * Cos[x⟦i - 1, 2⟧] h]}]
]

Show[{
  ListPlot[x, PlotStyle → Red, PlotRange → {1, 1.6}]
 }, ImageSize → Large]
```

Out[∘]=
$$\frac{1}{100}$$

Out[∘]=


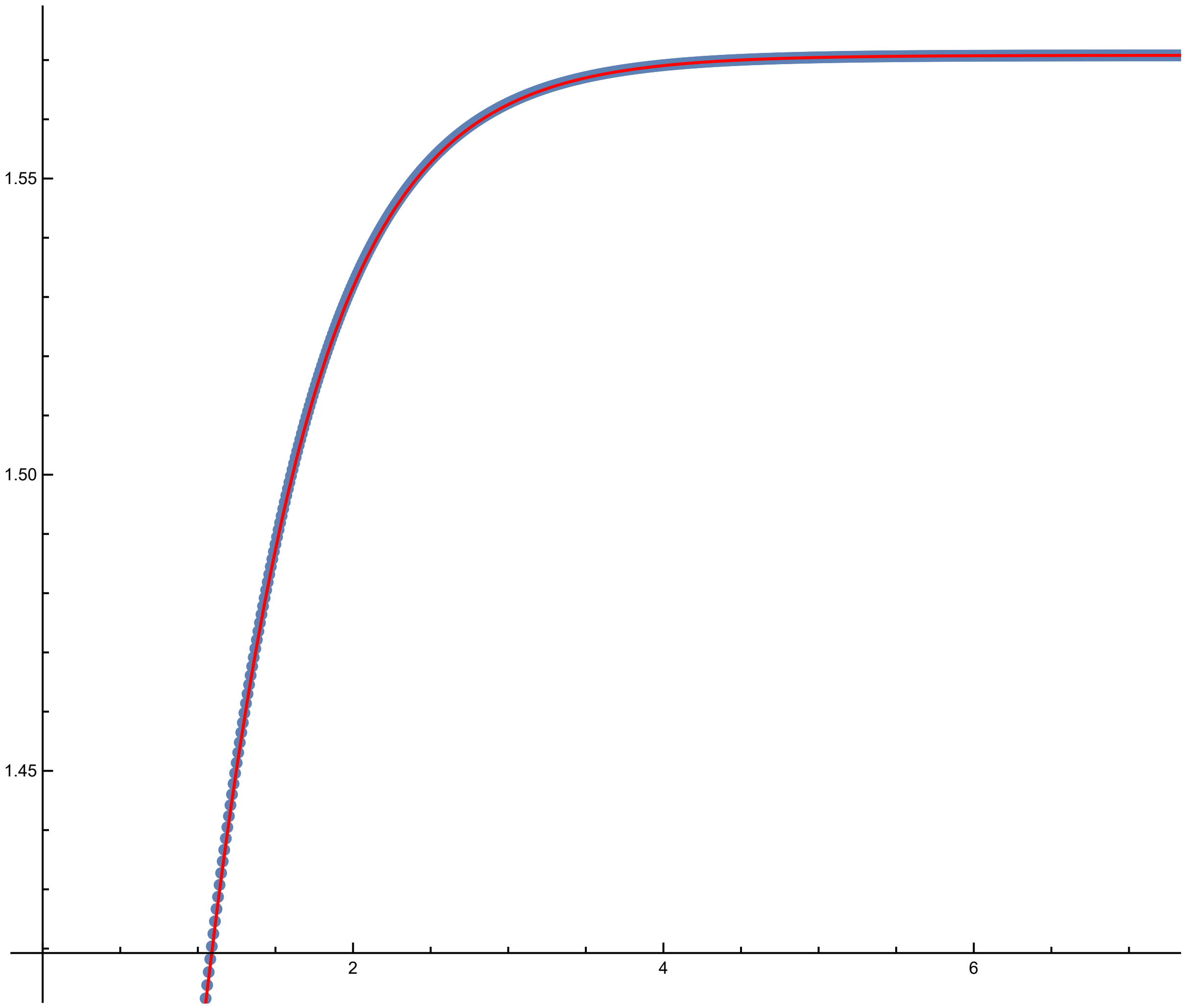
In[∘]:=

```
sol = NDSolve[{y'[t] == y[t] × Cos[y[t]], y[0] == 1}, y, {t, 0, 10}]
Show[{
  ListPlot[x],
  Plot[{y[t] /. sol}, {t, 0, 10},
   PlotRange → All, PlotLegends → "Expressions", PlotStyle → Red]
 }]
```

Out[∘]=

{{y → InterpolatingFunction[ Domain: {{0., 10.}} Output: scalar]}}

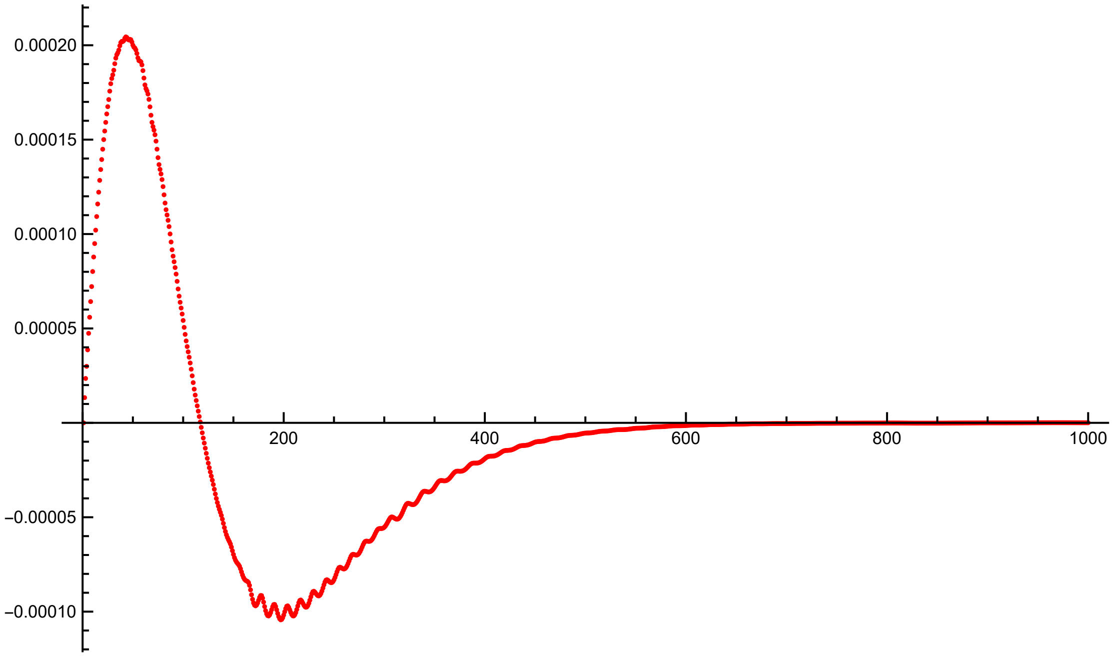
Out[∘]=



In[∘]:=
```
exact = Table[{t0, N[y[t0] /. sol][[1]]}, {t0, 0, x[[-1, 1]] + h, h}];

Show[{
  ListPlot[(x[[All, 2]] - exact[[All, 2]])/exact[[All, 2]] * 100, PlotStyle → Red, PlotRange → All]},
 ImageSize → Large]
```

Out[∘]=



In[∘]:=
```
Simplify[D[f[t] * Cos[f[t]], t] /. {f'[t] → f[t] × Cos[f[t]]}]
```

Out[∘]=

```
Cos[f[t]] × f[t] (Cos[f[t]] - f[t] × Sin[f[t]])
```

In[∘]:=
```
Length[x]
Length[exact]
```

Out[∘]=

```
1000
```

Out[∘]=

```
1999
```

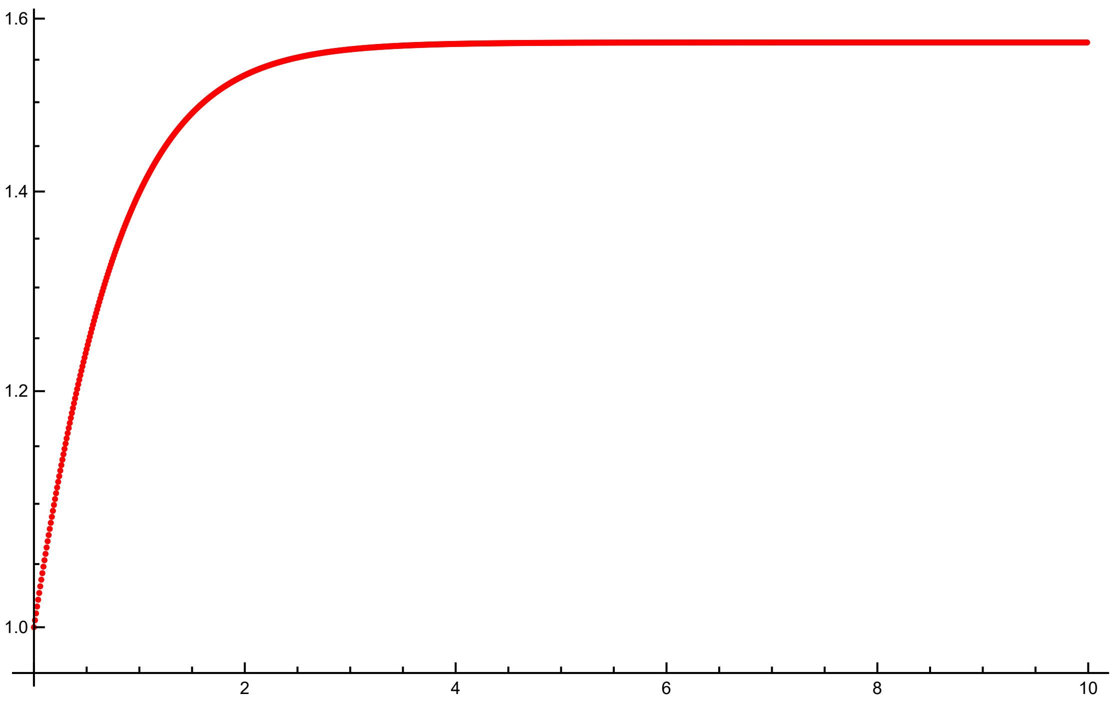
In[ ]:=
```
x = {{0, 1}};
h = 10^-2
For[i = 2, i ≤ 1000, i++,
 AppendTo[x, {
    N[x[[i - 1, 1]] + h],
    N[x[[i - 1, 2]] + x[[i - 1, 2]] * Cos[x[[i - 1, 2]]] h +
      h^2/2 x[[i - 1, 2]] * Cos[x[[i - 1, 2]]] (Cos[x[[i - 1, 2]]] - x[[i - 1, 2]] * Sin[x[[i - 1, 2]]])]}];
]

Show[{
  ListLogPlot[x, PlotStyle → Red]
 }, ImageSize → Large]
Show[{
  ListPlot[(x[[All, 2]] - exact[[All, 2]])/exact[[All, 2]] * 100, PlotStyle → Red]},
 ImageSize → Large, PlotRange → All]
```

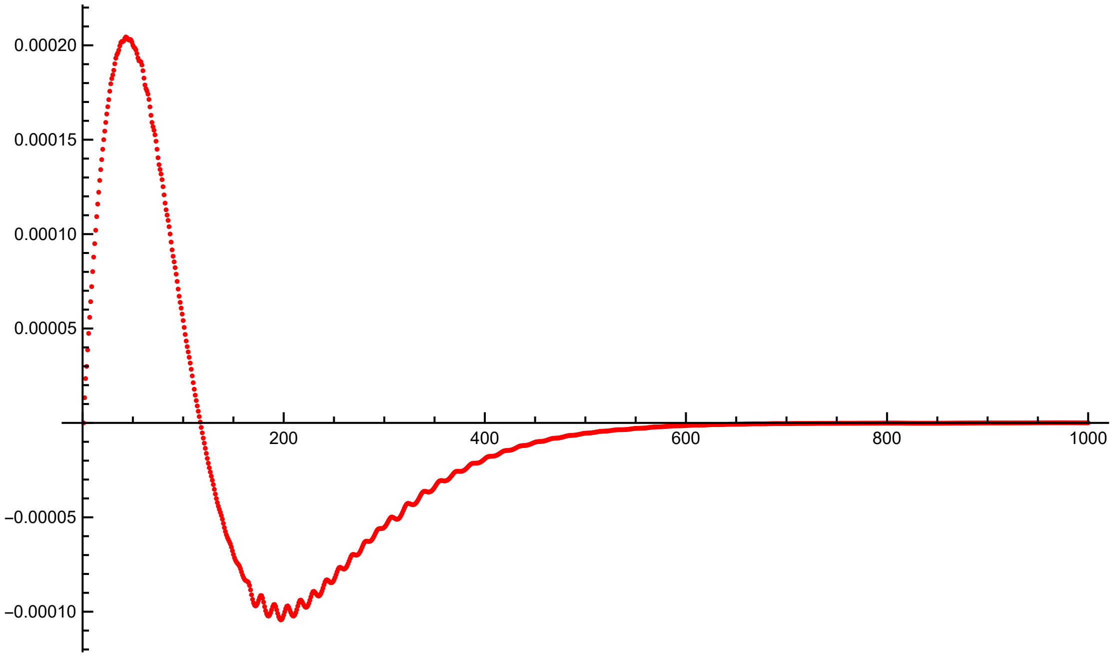
Out[ ]=

$$\frac{1}{100}$$

Out[ ]=



Out[ ]=



0.00020
0.00015
0.00010
0.00005
−0.00005
−0.00010
200
400
600
800
1000

Równania różniczkowe drugiego rzędu

Problem: Rozwiąż równanie oscylatora: $\frac{d^2 x}{dt^2} + b\,\frac{dx}{dt} + c\,x = 0$

In[∘]:=
```
Clear[x]
sol = DSolveValue[x''[t] + b x'[t] + c x[t] == 0, x, t]
```

Out[∘]=

$$\text{Function}\left[\{t\},\ e^{\frac{1}{2}\left(-b-\sqrt{b^2-4c}\right)t}\,\mathbb{C}_1 + e^{\frac{1}{2}\left(-b+\sqrt{b^2-4c}\right)t}\,\mathbb{C}_2\right]$$

Jak zastosować metodę Eulera? Wprowadzamy nową zmienną $\frac{dx}{dt} = v(t)$

$$\frac{dv}{dt} + b\,v(t) + c\,x(t) = 0$$
$$\frac{dx}{dt} = v(t)$$

$$\frac{dv}{dt} = -\,b\,v(t) - c\,x(t)$$
$$\frac{dx}{dt} = v(t)$$

$$v(t + \Delta t) = v(t) - (\,b\,v(t) + c\,x(t))\,\Delta t$$
$$x(t + \Delta t) = x(t) + v(t)\,\Delta t$$

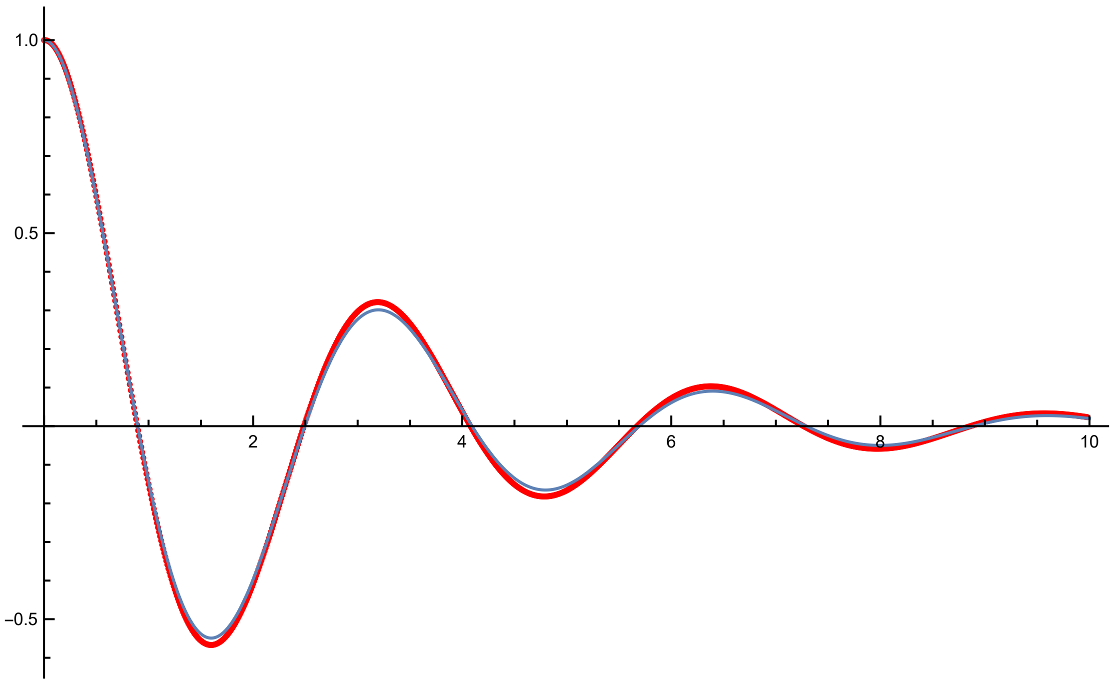
Przykład 1: x(0) = 1, v(0) = 0, b = 0.75, c = 4

In[ ]:=
```
x = {{0, 1}};
v = {{0, 0}};
h = 10^-2;
b = 0.75;
c = 4;
g[t_] :=
 e^(-0.375 t) (Cos[1.964529205687714 t] + 0.1908854288927333` Sin[1.964529205687714` t])
For[i = 2, i < 1000, i++,
 AppendTo[v, {
   N[v[[i - 1, 1]] + h],
   N[v[[i - 1, 2]] - (b v[[i - 1, 2]] + c x[[i - 1, 2]]) h]}];
 AppendTo[x, {
   N[x[[i - 1, 1]] + h],
   N[x[[i - 1, 2]] + v[[i - 1, 2]] * h]}]

]

Show[{
   ListPlot[x, PlotStyle → Red, PlotRange → All],
   Plot[g[t], {t, 0, 10}, PlotRange → All]
  }, ImageSize → Large]
```

Out[ ]=



In[ ]:=
```
DSolveValue[{y''[t] + b y'[t] + c y[t] == 0, y[0] == 1, y'[0] == 0}, y, t]
```

Out[ ]=

```
Function[{t}, 1. e^(-0.375 t) (1. Cos[1.96453 t] + 0.190885 Sin[1.96453 t])]
```

Przykład 2: Wahadło. Rozwiąż równanie wahadła:

$\frac{d^2x}{dt^2} + \sin(x(t)) = 0$ z warunkami początkowymi $x(0) = 1$, $x'(0) = 0$, a następnie porównaj wynik z przybliżeniem $\sin(x) \approx x$

$$\frac{dv}{dt} + \sin(x(t)) = 0$$
$$\frac{dx}{dt} = v(t)$$

$v(t+\Delta t)=v(t) - \sin(x(t))\,\Delta t$

$x(t+\Delta t) = x(t) + v(t)\,\Delta t$

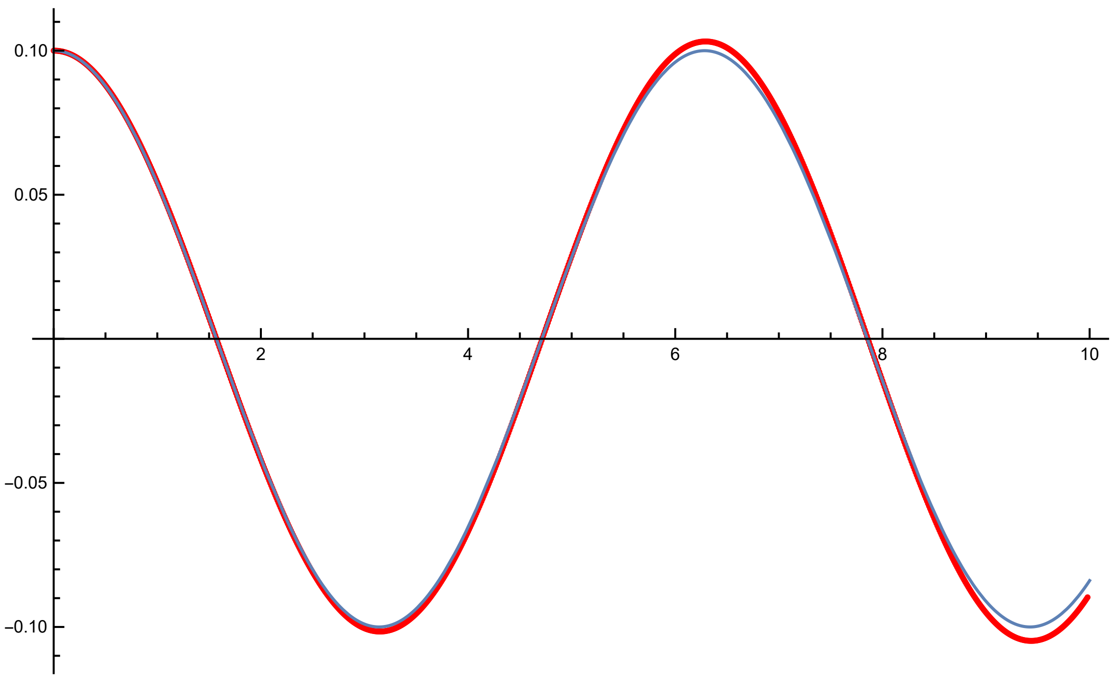
In[∘]:=
```
x0 = 0.1
x = {{0, x0}};
v = {{0, 0}};
h = 10^-2;
For[i = 2, i < 1000, i++,
 AppendTo[v, {
   N[v[[i - 1, 1]] + h],
   N[v[[i - 1, 2]] - Sin[x[[i - 1, 2]]] h]}];
 AppendTo[x, {
   N[x[[i - 1, 1]] + h],
   N[x[[i - 1, 2]] + v[[i - 1, 2]] * h]}]

]

Show[{
  ListPlot[x, PlotStyle → Red, PlotRange → All],
  Plot[x0 Cos[t], {t, 0, 10}, PlotRange → All]
 }, ImageSize → Large]
```

Out[∘]=
```
0.1
```

Out[∘]=



In[ ]:=
```
DSolve[{y''[t] + y[t] == 0, y[0] == 1, y'[0] == 0}, y, t]
```
Out[ ]=
```
{{y → Function[{t}, Cos[t]]}}
```

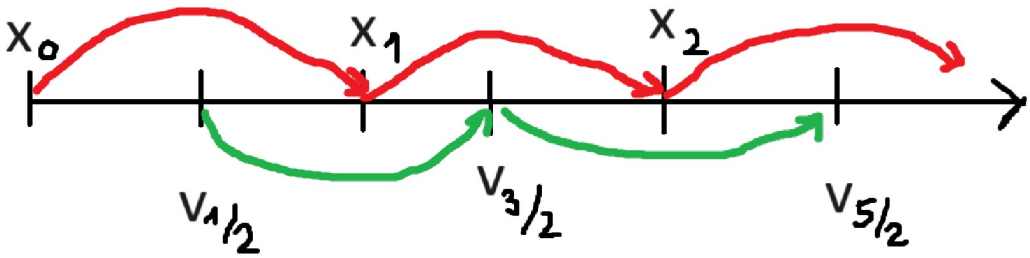
- Metoda Eulera może powodować niestabilności!

Leap-frog

$\frac{d^2 x}{dt^2} = A(x(t))$

Najpierw obliczamy prędkość w połowie kroku: $v\left(t+\frac{\Delta t}{2}\right)=v\left(t-\frac{\Delta t}{2}\right)+A(x(t))\,\Delta t$,
i korzystamy z niej aby obliczyć położenie:
$x(t+\Delta t)=x(t)+v\left(t+\frac{\Delta t}{2}\right)\Delta t$

Out[ ]=



Co jeśli nasze warunki początkowe to:
$x(t_0) = x_0,\ v(t_0)=v_0$? Do metody leapfrog potrzebujemy v w połowie kroku (Δt):
$v\left(t+\frac{1}{2}\Delta t\right)=v(t)+\frac{1}{2}\Delta t\,A(x)$

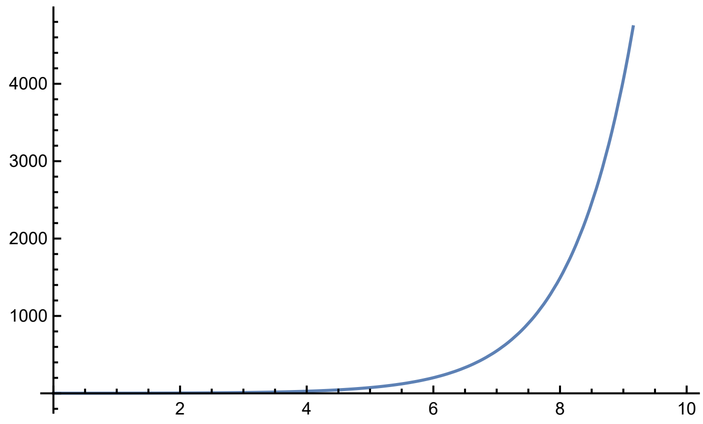
- Przykład: x''(t) = x(t), x(0) = 1, x'(0) = 0 - rozwiązanie analityczne to x(t)=cosh(t)

In[ ]:=
```
sol = DSolveValue[{y''[t] == y[t], y[0] == 1, y'[0] == 0}, y, t]
Plot[sol[t], {t, 0, 10}]
```

Out[ ]=

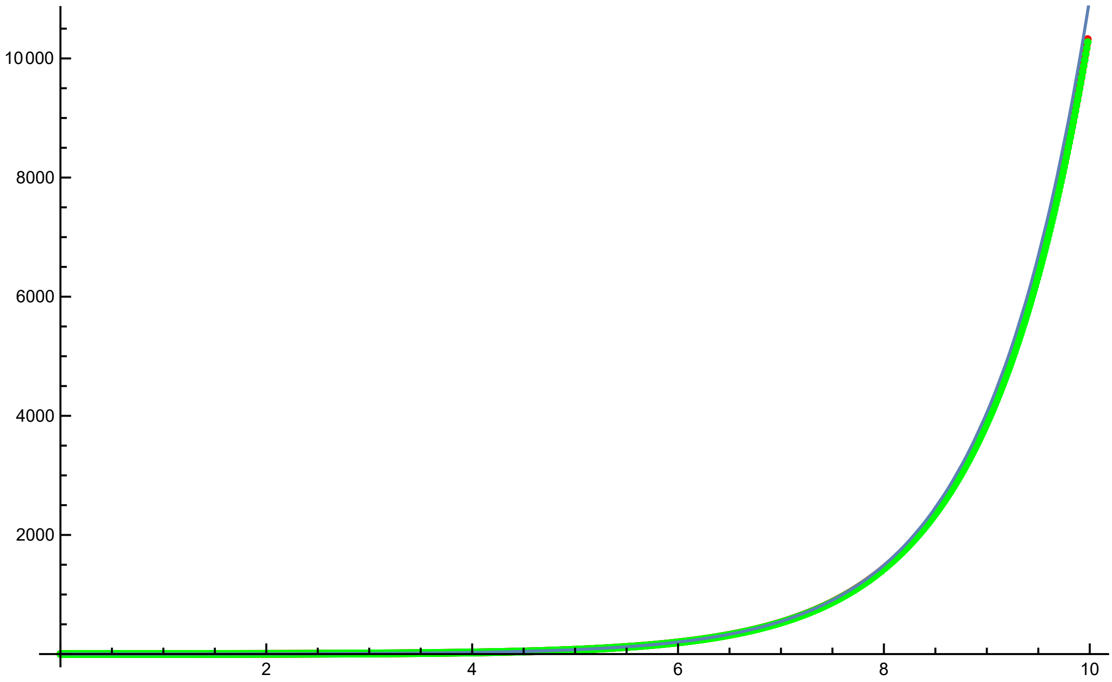
Function$\left[\{t\},\ \frac{1}{2}\, e^{-t}\left(1+e^{2t}\right)\right]$

Out[ ]=



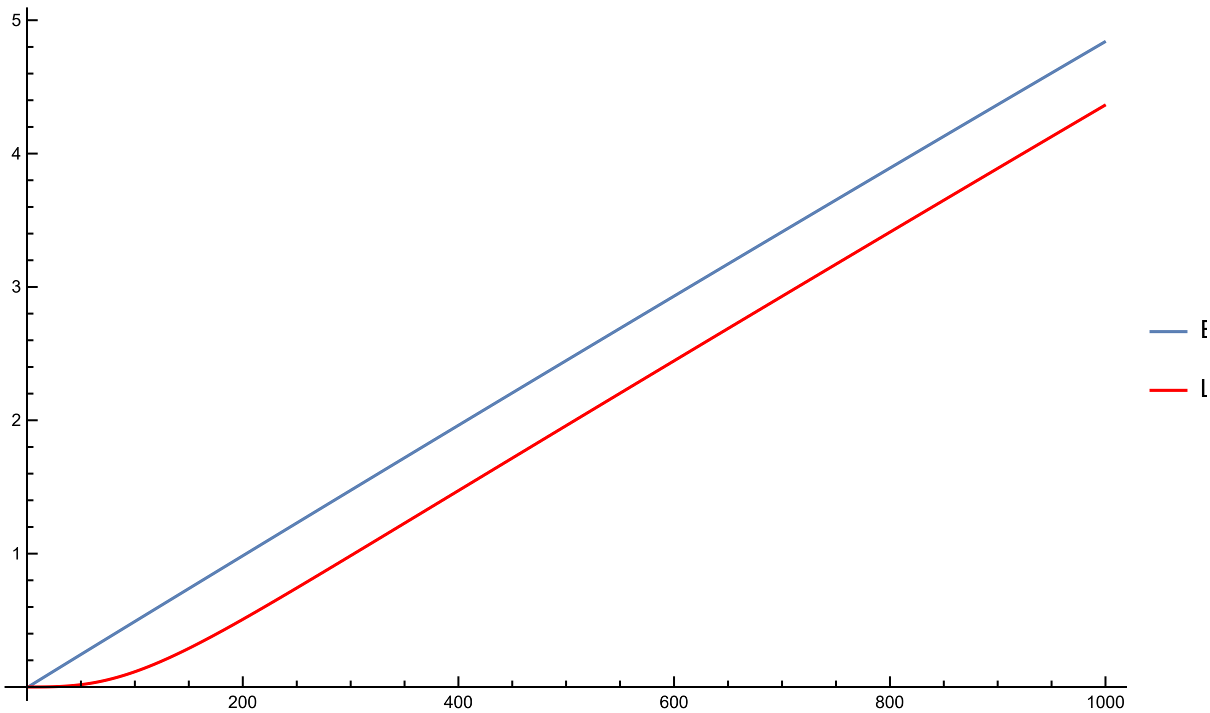
Równanie x''(t)=x(t), możemy zapisać w następujący sposób: v'(t)=x(t), v(t) = x'(t)

Out[ ]=



Błędy względne:

Out[ ]=



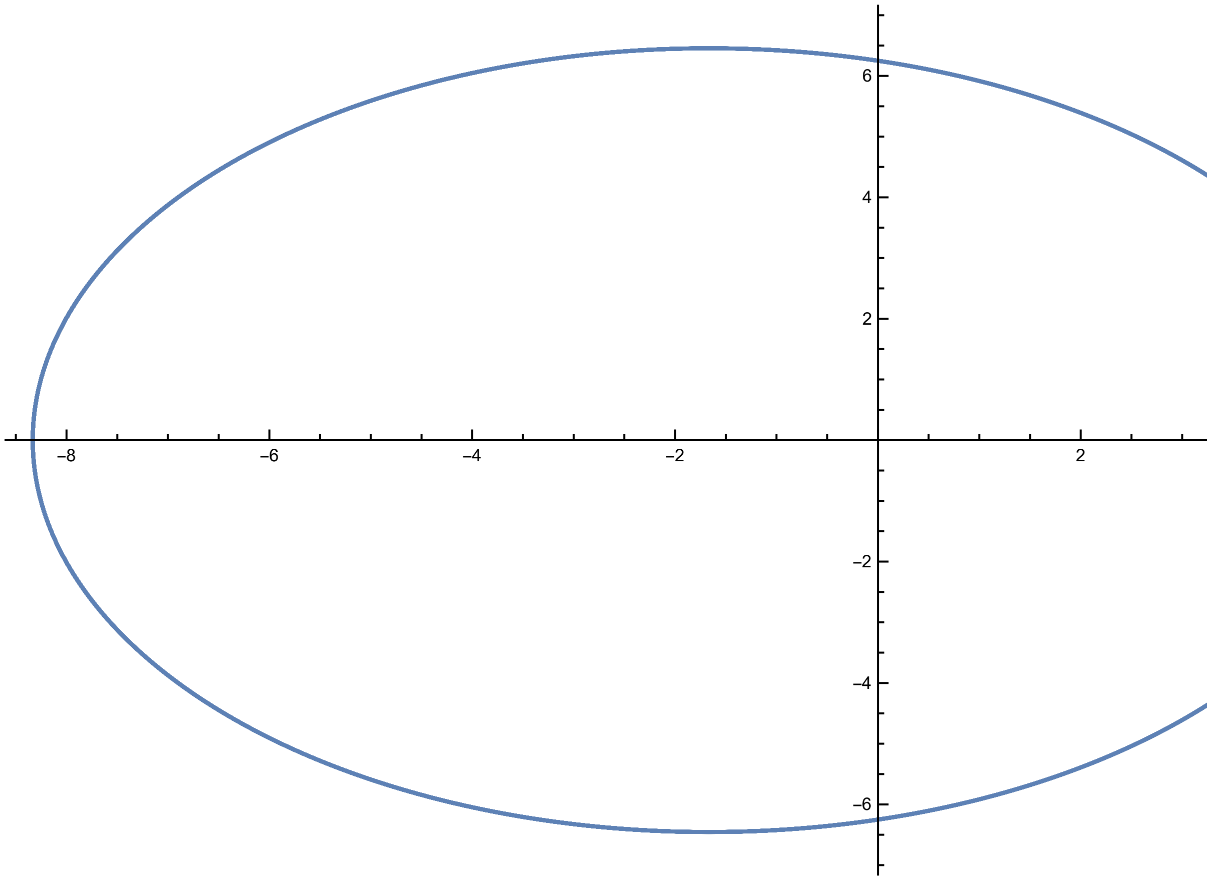
Metody Verlet'a

Równanie: $x''(t) = A(x)$, $x(t_0) = x_0$, $x'(t_0) = v_0$, $x_n = x(t_n)$, $t_n = t_0 + n\Delta t$

$x_1 = x_0 + v_0 \Delta t + \frac{1}{2} A(x_0) \Delta t^2$
$x_{n+1} = 2 x_n - x_{n-1} + A(x_n) \Delta t^2$
Uwaga! Powyższy zapis działa też dla wektorów

Problem dwóch ciał:
$x1''(t) = \frac{-x1}{(x1^2+y1^2)^{3/2}}$,
$y1''(t) = \frac{-y1}{(x1^2+y1^2)^{3/2}}$,

Out[ ]=



Velocity Verlet

Najpierw obliczamy:
$v\left(t + \frac{\Delta t}{2}\right) = v(t) + \frac{\Delta t}{2} A(y(t))$
wykorzystując $v\left(t + \frac{\Delta t}{2}\right)$, znajdujemy: $y(t + \Delta t) = y(t) + v\left(t + \frac{\Delta t}{2}\right) \Delta t$,
znając y(t+Δt) obliczamy A(y(t + Δt)), a na sam koniec otrzymujemy prędkość:

$v(t+\Delta\mathrm{t}) = v\left(t+\frac{\Delta\mathrm{t}}{2}\right) + \frac{\Delta\mathrm{t}}{2}\, A(y(t+\Delta\mathrm{t}))$

$y(t+\Delta\mathrm{t}) = y(t) + v(t)\,\Delta\mathrm{t} + \frac{1}{2}\, A(y(t))\,\Delta\mathrm{t}^2$

$v(t+\Delta\mathrm{t}) = v(t) + \frac{A(y(t))+A(y(t+\Delta\mathrm{t}))}{2}\,\Delta\mathrm{t}$

Do obliczenia A(y(t + Δt))

Metody explicit vs implicit (jawne vs uwikłane?)

Euler: f'(x) = A(f, x)

- explicit (forward):
  $\frac{f(x+h)-f(x)}{h} = A(f(x), x)$
  f(x+h) = f(x) + A(f(x), x)h
- implicit (backward formula):
  $\frac{f(x+h)-f(x)}{h} = A(f(x+h), x)$
  f(x+h) = f(x) + A(f(x+h), x+h) h
  żeby dostać f(x+h) musimy rozwiązać dodatkowe równanie! (które może być nieliniowe)

Podsumowanie

- Metody jawne są łatwiejsze do zaimplementowania i szybsze
- Metody uwikłane wymagają rozwiązania dodatkowego równania/układu równań, ale są dokładniejsze

Metody Runge-Kutty

Wstęp: równanie y' = f(y,t)

Powyższe równanie możemy od-całkować:

$y(t+h) = y(t) + \int_t^{t+h} f(y(t), t)\, dt$

Przybliżmy tą całkę korzystając z metody trapezów:

$y(t+h) = y(t) + \frac{h}{2}\left(f(y(t+h), t+h) + f(y(t), t) \right)$ <- metoda uwikłana! Nie znamy y(t+h) do obliczenia f(y(t+h),t+h).

Rozwiązanie: skorzystajmy z metody Eulera: y(t+h) ≈ Y(t) = y(t) + h f(y(t),t), co daje rezultat:

$y(t+h) = y(t) + \frac{h}{2}\left(f(Y(t), t+h) + f(y(t), t) \right)$

Metoda Runge-Kutty czwartego rzędu:

Analogicznie do poprzedniego przykładu ale oparta na metodzie Simpsona:

- y' = f(y,t)
- $y(t+h) = y(t) + \frac{h}{6}\left(Y_1 + 2 Y_2 + 2 Y_3 + Y_4 \right)$
- $Y_1 = f(y(t), t)$
  $Y_2 = f\left(y(t) + \frac{h}{2} Y_1, t + \frac{h}{2}\right)$
  $Y_3 = f\left(y(t) + \frac{h}{2} Y_2, t + \frac{h}{2}\right)$
  $Y_4 = f(y(t) + hY_3, t+h)$

Euler vs RK2 vs RK4

Rozwiążemy równanie $y'(t) = -y^2$, $y(1) = 1$ trzema poznanymi metodami

In[∘]:=
```
sol = DSolveValue[{y'[t] == -y[t]^2, y[1] == 1}, y, t]
```

Out[∘]=
```
Function[{t}, 1/t]
```

In[∘]:=
```
D[-y[t]^2, t] /. {y'[t] → -y[t]^2}
D[2 y[t]^3, t] /. {y'[t] → -y[t]^2}
D[-6 y[t]^4, t] /. {y'[t] → -y[t]^2}
```

Out[∘]=
```
2 y[t]^3
```

Out[∘]=
```
-6 y[t]^4
```

Out[∘]=
```
24 y[t]^5
```

Out[ ]=