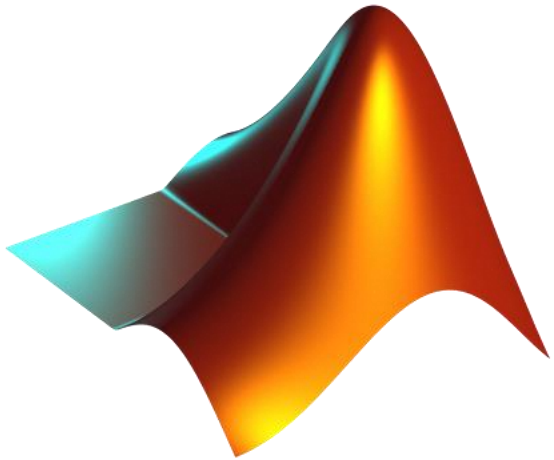
Programowanie w zastosowaniach inżynierskich

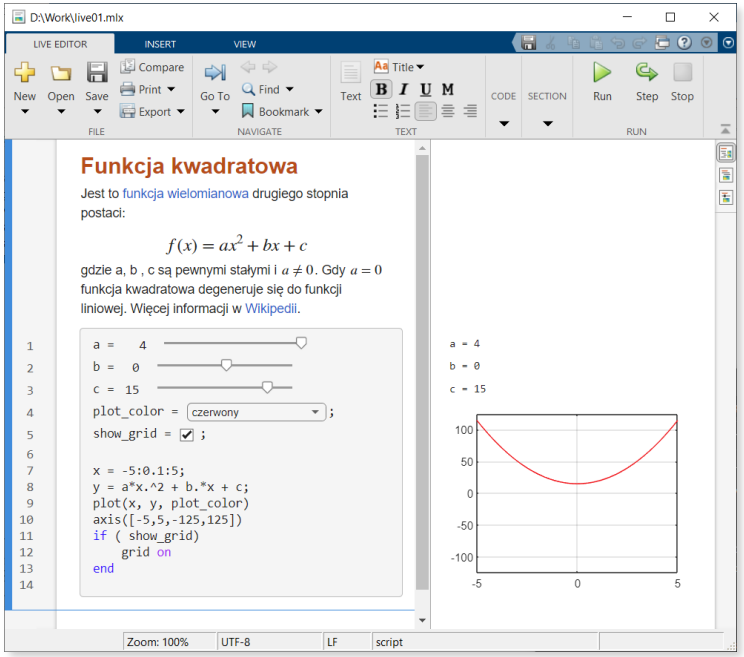


Live scripts
Obliczenia symboliczne
Instrukcja warunkowa

Live scripts

Live script (aktywny skrypt) to interaktywny dokument MATLAB-a (rozszerzenie mlx), który łączy kod programu ze sformatowanym tekstem, grafiką oraz elementami multimedialnymi w środowisku Live Editor. Cechy aktywnych skryptów:

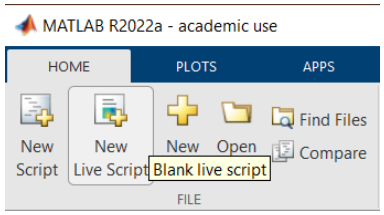
- Umożliwiają wprowadzanie rozbudowanych komentarzy (tekst, grafika, multimedia) opisujących problem i kod będący jego rozwiązaniem;
- Kod może być uzupełniony o interaktywne elementy sterujące, które umożliwiają wybór wartości zmiennych, ustalanie parametrów, itp.
- Kod umieszczony w skrypcie jest uruchamiany w środowisku Live editor;
- Skrypt przechowuje i wyświetla wyniki razem z kodem, który je wytworzył;
- Skrypty mogą być eksportowane do formatów PFD, Word, HTML, LaTeX.



Wymagania

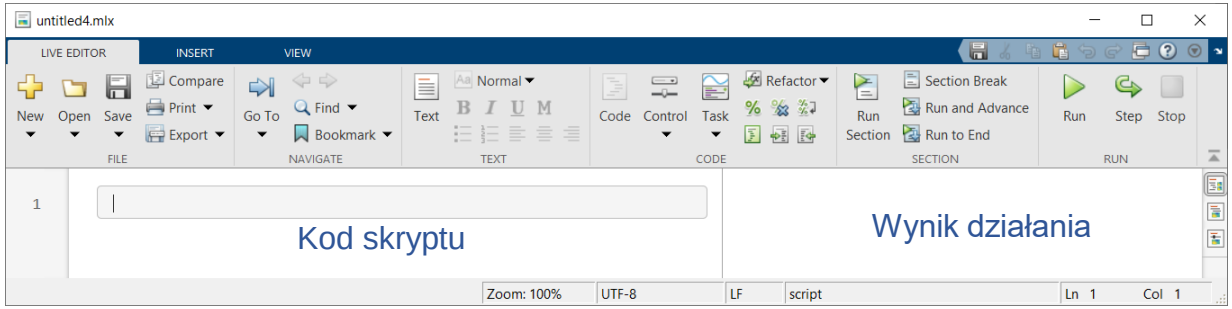
- MATLAB 2016a live scripts
- MATLAB 2018a live functions

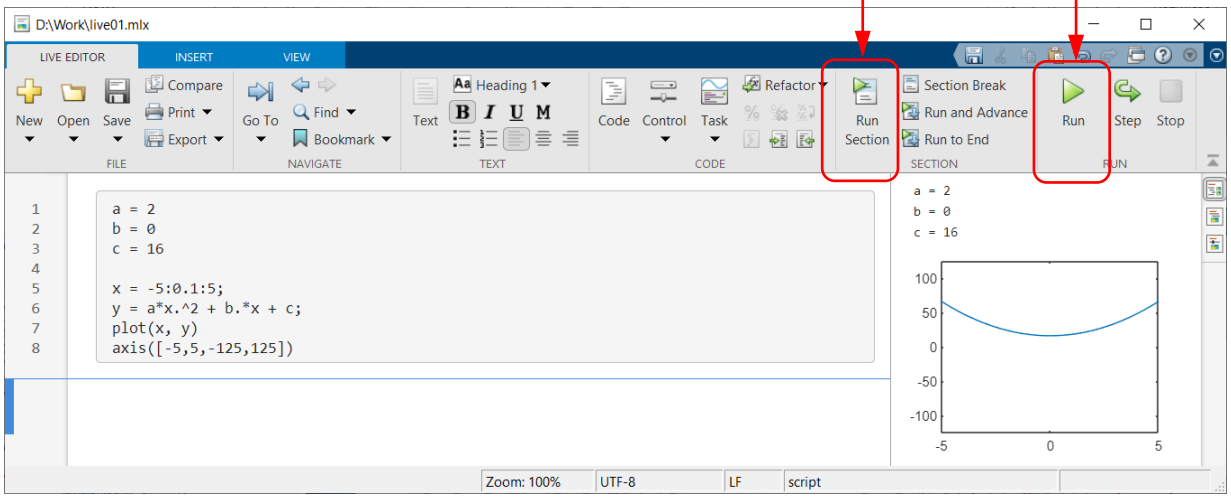
Tworzenie Live script



`Home > New Live Script`

Polecenie (Command window): **`edit script_name.mlx`**





Uruchomienie:
- pojedyncza sekcja
- całość

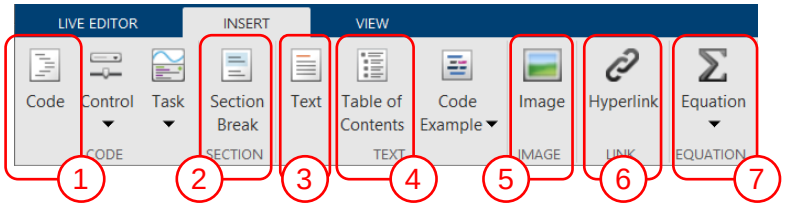
Kod skryptu

```
a = 2
b = 0
c = 16
x = -5:0.1:5;
y = a*x.^2+b.*x+c;
plot(x, y)
axis([-5,5,-125,125])
```

Uwaga: instrukcja `axis` ustawia zakresy osi wykresu.

Live script – komentarze

Wstawianie sekcji i komentarzy

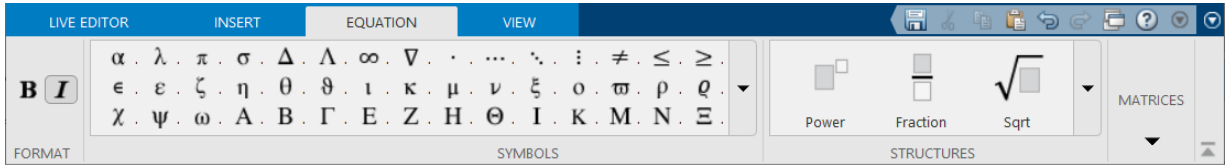


1 – sekcja kodu 2 – koniec sekcji 3 – sekcja tekstowa

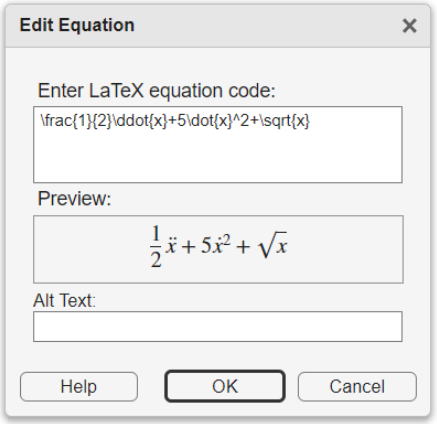
4 – spis treści (lista sekcji z odnośnikami)

5 – grafika 6 – odnośnik 7 – wzór

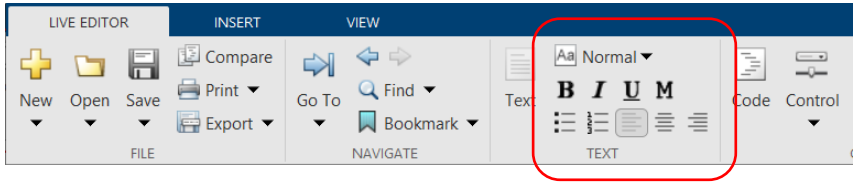
Edytor wzorów (7)

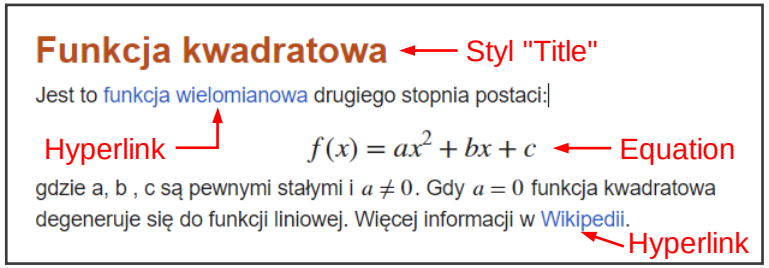


Uwaga: poza edytorem graficznym Live script editor umożliwia wprowadzanie wzorów zgodnie e standardem LaTeX (druga opcja w menu Equation).



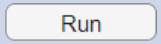
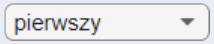
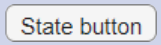
Formatowanie tekstu





Live script – elementy sterujące

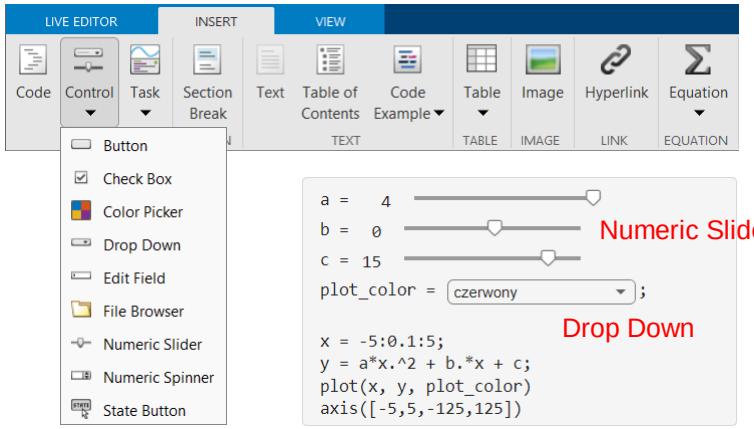
Interactive Controls to zestaw elementów sterujących umożliwiających interaktywną zmianę wartości zmiennych podczas działania skryptu. Zmiana wartości wykonana za pomocą takiego elementu powoduje uruchomienie kodu i wykonanie instrukcji zawartych w bieżącej sekcji lub całym skrypcie (zależnie od ustawień elementu).

Element	*Nazwa*	*Zastosowanie*
	Button	Uruchamiane skryptu
	Check Box	Ustawianie wartości logicznej (True/False)
	Color Picker	Wybór koloru
	Drop-Down List	Wybór jednej z dozwolonych wartości (lista)
	Edit Field	Wprowadzanie wartości z klawiatury
	File Browser	Wybór pliku ze standardowego okna dialogowego
	Numeric Slider	Zmiana wartości numerycznej przy pomocy suwaka
	Numeric Spinner	Zmiana wartości numerycznej w polu edycyjnym
	State Button	Ustawianie wartości logicznej (alternatywa dla Check Box)

Uwaga: zestawienie elementów sterujących dla wersji 2023b

Live script – elementy sterujące

Dodawanie elementów sterujących



W celu dodania elementu ustawiającego wartość zmiennej należy zaznaczyć wartość przypisaną do zmiennej i wybrać odpowiedni element. Powiązanie ze zmienną wykonywane jest automatycznie.

Uwaga: w menu Control aktywne są elementy zgodne z typem zmiennej.

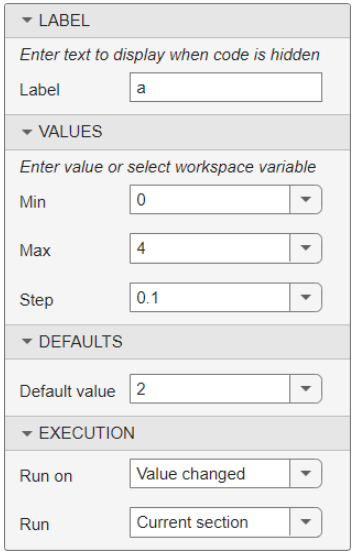
Konfiguracja elementów sterujących

`Menu podręczne > Configure control` lub podwójne kliknięcie

Sekcje okna konfiguracyjnego:

- Label – etykieta wyświetlana gdy kod jest ukryty;
- Values – ustawienia dotyczące wartości (specyficzne dla elementu);
- Defaults – wartość domyślna;
- Execution – sposób działania:
  - Run on – reakcja na zmianę wartości: Value changing lub Value changed ciągła zmiana lub po przestawieniu elementu,
  - Run – zasięg działania.

Konfiguracja *Numeric slider*



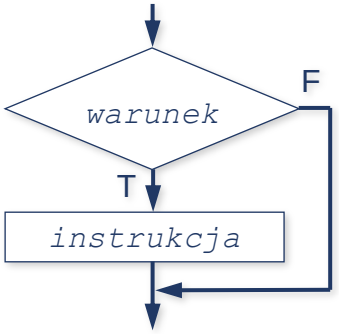
Instrukcja warunkowa

Instrukcja sterująca – element języka programowania, który służy do określenia kolejności wykonania instrukcji zawartych w kodzie programu.

Instrukcja warunkowa – wprowadza rozgałęzienie w kodzie programu, tworząc alternatywne sekwencje instrukcji.

Wariant I

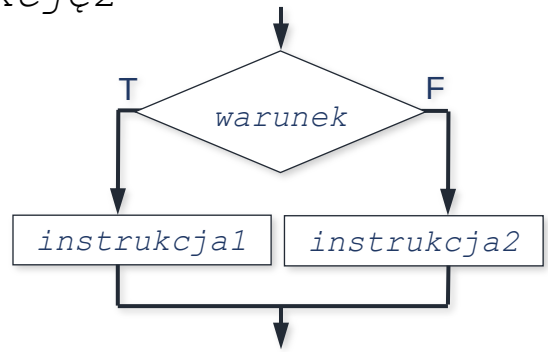
Jeżeli *warunek* jest prawdziwy wykonaj *instrukcję* (grupę instrukcji).




```
if warunek
  instrukcja
end
```

Wariant II

Jeżeli *warunek* jest prawdziwy wykonaj *instrukcję1* w przeciwnym wypadku *instrukcję2*




```
if warunek
  instrukcja1
else
  instrukcja2
end
```

Instrukcja warunkowa – przykład

```
a =   4
b =   0
c =  12
plot_color = czerwony ;
show_grid = ☐ ;

x = -5:0.1:5;
y = a*x.^2 + b.*x + c;
plot(x, y, plot_color)
axis([-5,5,-125,125])
xticks(-4:2:4)
if ( show_grid )
    grid on
end
```

`show_grid` – zmienna logiczna

`Check Box` – ustawienie wartości zmiennej show_grid

`grid on` – polecenie ustawiające widoczność siatki (domyślnie niewidoczna)

`xticks` – gęstość podziałki w osi x

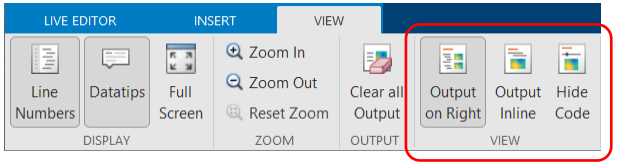
```
a =   4
b =   0
c =  12
plot_color = czerwony ;
show_grid = ☑ ;

x = -5:0.1:5;
y = a*x.^2 + b.*x + c;
plot(x, y, plot_color)
axis([-5,5,-125,125])
xticks(-4:2:4)
if ( show_grid )
    grid on
end
```


```
if ( show_grid )
  grid on
end
```

Uwaga: Jeżeli zmienna **`show_grid`** jest ustawiona na `True` (warunek prawdziwy) wykonywana jest instrukcja **`grid on`**, która ustawia widoczność siatki. Gdy **`show_grid`** jest ustawiona na `False` (warunek fałszywy) instrukcja **`grid on`** jest pomijana i siatka pozostaje niewidoczna.

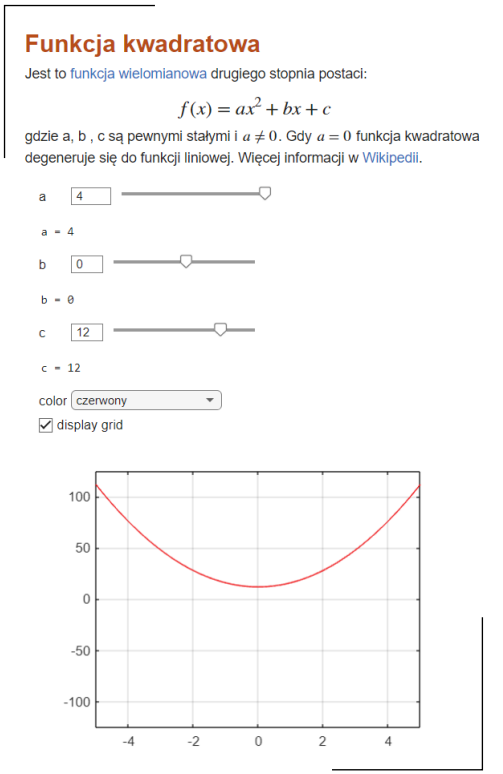
Live script – widoki

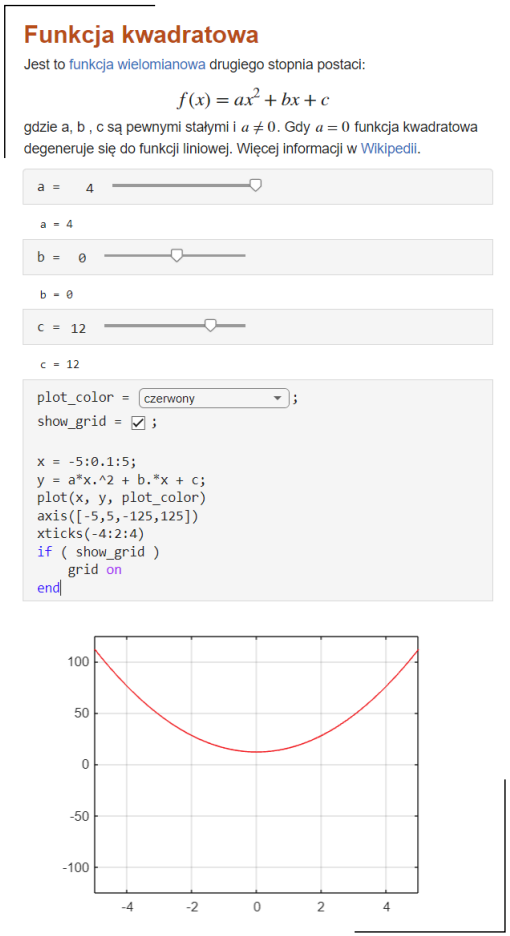


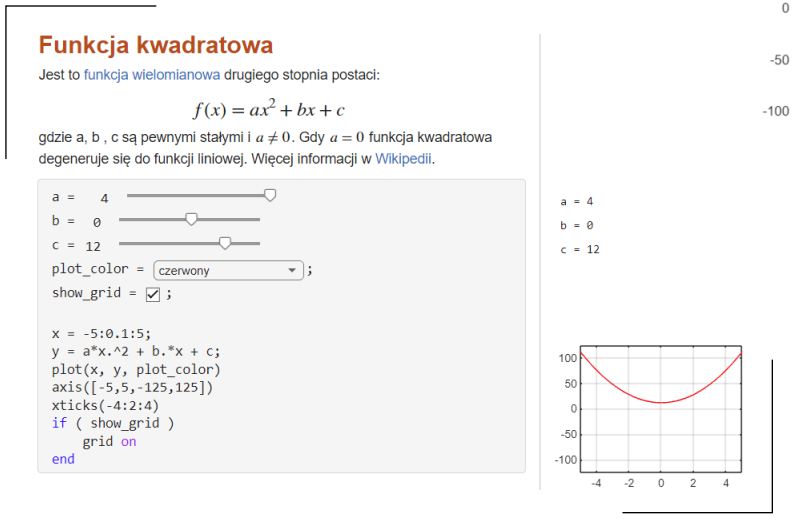
Output on Right wyniki wyświetlane na prawo obok kodu, który je generuje.

Output Inline każdy wynik wyświetlany bezpośrednio pod kodem.

Hide Code kod ukryty.





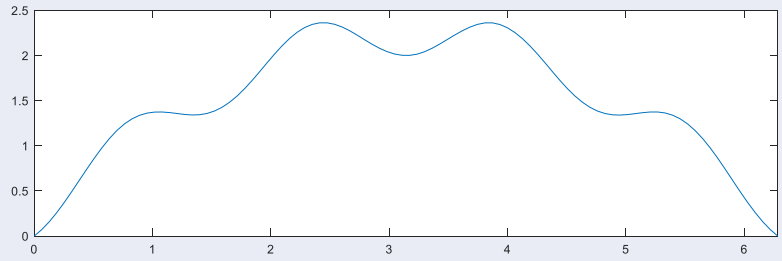


Więcej informacji : mathworks.com

Obliczenia numeryczne i symboliczne

Obliczenia numeryczne – metody rozwiązywania problemów matematycznych wykorzystujące wyłącznie działania na liczbach. Uzyskiwane wyniki na ogół są przybliżone, jednak zazwyczaj dokładność obliczeń może być dobrana do potrzeb.

Obliczenia symboliczne – metody rozwiązywania problemów matematycznych wykorzystujące operacje na symbolach. Uzyskiwane rozwiązania mają postać równań lub funkcji opisujących dokładne rozwiązanie problemu.

Rozwiązanie numeryczne	***Rozwiązanie symboliczne***
równanie algebraiczne $x^2 + 2x + a = 0$ x niewiadoma, a parametr	
dla $a = 5$: $x_1 \approx -3{,}44949$, $x_2 \approx 1{,}44949$	$x = \pm\sqrt[2]{1-a} - 1$
równanie różniczkowe $\dot{y}(t) = sin(at) + cos(bt), \quad y(0) = 0$ $y(t)$ niewiadoma funkcja, a, b parametry	
dla $a = 4$ i $b = 0.5$ w czasie $t \in [0, 2\pi]$ 	$y(t) = \frac{1}{a} - \frac{\cos(at)}{a} + \frac{\sin(bt)}{b}$

Symbolic Math Toolbox udostępnia zestaw funkcji do przeprowadzania obliczeń symbolicznych. Najważniejsze zastosowania:

- algebra liniowa,
- rachunek różniczkowy i całkowy,
- rozwiązywanie równań algebraicznych i różniczkowych,
- upraszczanie i transformacja wyrażeń matematycznych.

Tworzenie obiektów symbolicznych

Instrukcja	*Przykład*	*Opis*
`syms n1 n2 … nN`	`syms a b c`	Tworzy skalarne zmienne symboliczne o nazwach określonych przez *n1, n2, …, nN*
`syms n1 n2 … nN set`	`syms q w real`	Tworzy skalarne zmienne symboliczne przypisując im ograniczenia wyspecyfikowane przez *set*, dozwolone wartości: `real, positive, integer, rational`
`syms fun(v1,v2,…,vN)`	`syms f(x,y)`	Tworzy symboliczną funkcję *fun* oraz symboliczne zmienne *v1*, *v2*, …, *vN* reprezentujące jej argumenty
`n = sym('n')`	`x=sym('x')`	Tworzy skalarną zmienną symboliczną o nazwie *n*
`n = sym('n', set)`	`q=sym('q',real)`	Tworzy skalarną zmienną symboliczną z ograniczeniami
`v = sym(num)`	`f=sym(1/7)`	Konwertuje wartość numeryczną *num* na symboliczną wartość numeryczną

Wyrażenie symboliczne jest dowolnym wyrażeniem zawierającym zmienne symboliczne. Do budowy takich wyrażeń mogą być wykorzystane standardowe operatory i wbudowane funkcje Matlab-a.

Manipulowanie wyrażeniami symbolicznymi

Funkcja	*Przykład*	*Opis*
`collect(exp,v)`	`collect((x-5)^2+(y-2)^2,x)` `x^2-10*x+(y-2)^2+25`	Grupuje współczynniki zmiennej `v`
`expand(exp)`	`expand(sin(x+y))` `cos(x)*sin(y)+cos(y)*sin(x)`	Rozwija wyrażenie `exp`
`fplot(exp,r)`	`fplot(2*x^2+5, [-3,3])`	Wykreśla funkcję jednej zmiennej opisanej przez `exp` w przedziale `r` (dwuelementowy wektor)
`poly2sym(c,var)`	`poly2sym([2,3,0,1], y)` `2*y^3 + 3*y^2 + 1`	Tworzy wielomian zmiennej `var` o współczynnikach określonych wektorem `c`
`simplify(exp)`	`simplify((x^2+5*x+6)/(x+2))` `x+3`	Upraszcza wyrażenie *exp*
`subs(exp,v1,v2)`	`subs(1/2+sin(y), y, pi/4)` `2^(1/2)/2 + 1/2`	Podstawia za zmienną symboliczną `v1` wartość `v2` (obiekt symboliczny, liczba) w wyrażeniu *exp*
`vpa(exp,d)`	`vpa(sin(x)+1/7, 5)` `sin(x) + 0.14286`	Wyznacza wartości stałych symbolicznych z dokładnością do d miejsc po kropce

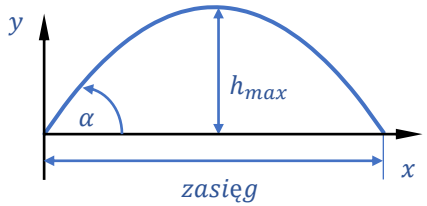
Rozwiązywanie równań algebraicznych

`S = solve(eqn, var)`

funkcja rozwiązuje równanie opisane wyrażeniem *eqn* ze względu na zmienną *var*.

Przykład. Rzut ukośny

```
>> syms x(t) y(t) v0 a g
>> x(t) = v0*t*cos(a);
>> y(t) = v0*t*sin(a)-g*t^2/2;
>> T = solve(y==0,t)
   T =
        0
      (2*v0*sin(a))/g
>> T = T(2);
>> z = subs(x, t, T)
   z =
      (2*v0^2*cos(a)*sin(a))/g
>> h = subs(y, t, T/2)
   h =
      (v0^2*sin(a)^2)/(2*g)
```



$$x(t) = v_0\, t \cos\alpha$$

$$y(t) = v_0\, t \sin\alpha - \frac{g\, t^2}{2}$$

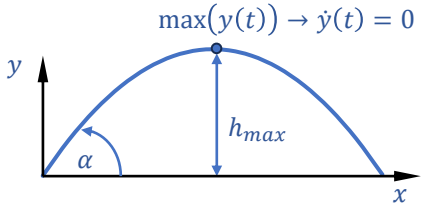
```
% uproszczenie wyrażenia na "z"
>> z = simplify(z)
   z =
      (v0^2*sin(2*a))/g
% wyrażenie w czytelnej formie
>> pretty(z)
      2
   v0  sin(2 a)
   ------------
         g
```

`Df = diff(f, var, n)`

wyznacza pochodną funkcji `f` rzędu `n` ze względu na zmienną `var`

Przykład. Rzut ukośny (alternatywne wyznaczenie h_{max})

```
>> syms y(t) v0 a g
>> y(t) = v0*t*sin(a)-g*t^2/2;
>> T = solve(diff(y,t)==0,t)
   T =
      (v0*sin(a))/g
>> h = y(T)
   h =
      (v0^2*sin(a)^2)/(2*g)
>> pretty(h)
      2        2
   v0  sin(a)
   -----------
       2 g
```



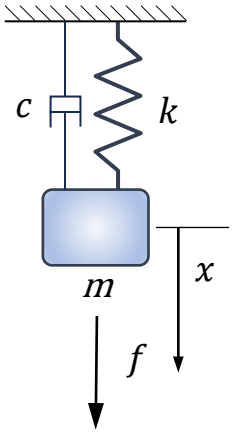
$$y(t) = v_0\, t \sin\alpha - \frac{g\, t^2}{2}$$

Rozwiązywanie równań różniczkowych

`S = dsolve(eqn, cond)`

Funkcja rozwiązuje równanie różniczkowe `eqn` z warunkami początkowymi `cond`

Przykład. Układ masa-sprężyna

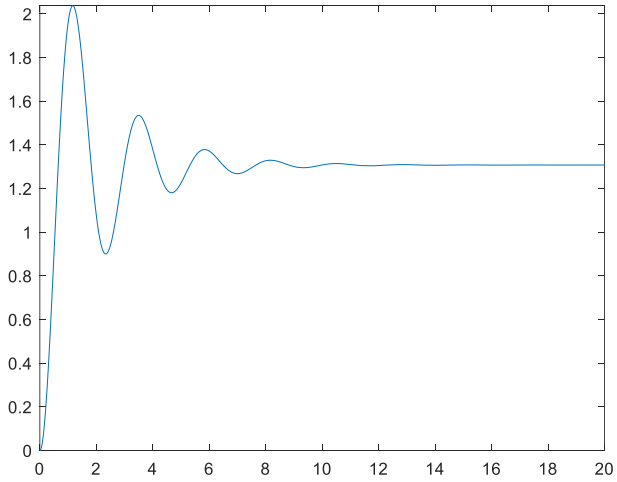


$$m\ddot{x} + c\dot{x} + kx = f(t)$$

$$f = m\,g$$

$$x = 0,\ \dot{x} = 0$$

```
>> syms x(t)
>> syms m c k g positive
>> E = m*diff(x,t,2)+c*diff(x,t)+k*x == m*g
>> Dx(t) = diff(x(t),t);
>> X(t) = dsolve(E, [x(0)==0, Dx(0)==0]);
>> X(t) = subs(X,[m,c,k,g], [2,2,15,9.81]);
>> fplot(X(t),[0,20])
```



Obliczenia symboliczne w Live Script

D:\Work\mass_spring.mlx

LIVE EDITOR | INSERT | VIEW

New | Open | Save | Compare | Print | Export | Go To | Find | Bookmark | Text | Normal | Refactor | Code | Control | Task | SECTION | Run | Step | Stop

FILE | NAVIGATE | TEXT | CODE | RUN

```
1  syms x(t)
2  syms m c k g positive
3
4  E = m*diff(x,t,2) + c*diff(x,t) + k*x == m*g
5
6  Dx(t) = diff(x(t),t);
7  X(t) = dsolve(E, [x(0)==0, Dx(0)==0])
8
9  X(t) = subs(X,[m, c, k, g], [2, 2, 15, 9.81]);
10
11 fplot(X(t),[0,20])
```

E(t) =

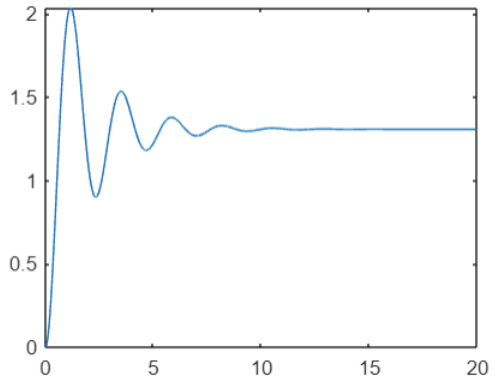
$$m\frac{\partial^2}{\partial t^2}x(t) + c\frac{\partial}{\partial t}x(t) + k\,x(t) = g\,m$$

X(t) =

$$\frac{g\,m}{k} + \frac{g\,m\,e^{-\frac{t\,(c+\sigma_1)}{2\,m}}\,(c-\sigma_1)}{2\,k\,\sigma_1} - \frac{g\,m\,e^{-\frac{t\,(c-\sigma_1)}{2\,m}}\,(c+\sigma_1)}{2\,k\,\sigma_1}$$

where

$$\sigma_1 = \sqrt{c^2 - 4\,k\,m}$$



Zoom: 100% | UTF-8 | LF | script | Ln 3 | Col 1