Politechnika Wrocławska

Obliczenia smboliczne w MATLABie

Zastosowanie programu MATLAB w obliczeniach inżynierskich

Wojciech Myszka

Katedra Mechaniki, Inżynierii Mechanicznej i Biomedycznej

27 października 2024

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

1. Struktury danych
2. Zmienne tekstowe
3. Obliczenia symboliczne
4. Rozwiązywanie równań
5. Układy równań
6. Calculus

Struktury danych

Politechnika Wrocławska

Wstęp

1. Bardzo często mamy do czynienia z danymi, które nie mają *jednolitego* (tekst, wartości liczbowe) formatu.
2. Na przykład prowadząc eksperyment notujemy nie tylko wartości liczbowe jakiegoś parametru, ale chcemy też opisać obiekt eksperymentu.
3. Trzeba stworzyć coś w rodzaju bazy danych:
   - wpisy mają formę rekordów
   - poszczególne pola zawierają różne wartości.
4. W wielu językach programowania realizowane jest to z użyciem **struktur**.

Politechnika Wrocławska

Struktury I

- **Struktura** to rodzaj pojemnika mogącego zawierać dane różnego typu.
- Każda zmienna typu struktura ma swoją nazwę
- Dodatkowo każde pole ma swoją nazwę
- Dostęp do zawartości pola odbywa się łącząc nazwę zmiennej (struktury) z nazwą pola znakiem kropka .

```
Obiekt.typ = 'szczur';
Obiekt.nazwa = 'Ziutek';
Obiekt.waga = 0.35; % w kilogramach
Obiekt.data_przyjecia = datetime(2021,01,15);
```

- Można użyć alternatywnej formy

Struktury II

```
Obiekt = struct('typ', 'szczur', 'nazwa', 'Ziutek',
         'waga', 0.35, 'data_przyjecia',
         datetime(2021,01,15))
```

- Odpytanie o wartość:

```
>> Obiekt
Obiekt =
  struct with fields:
                 typ: 'szczur'
              nazwa: 'Ziutek'
               waga: 0.3500
    data_przyjecia: 15-Jan-2021
```

Funkcje I

1. `struct2table()` konwersja do tabeli

```
>> struct2table(Obiekt)
ans =
  1×4 table
     typ       nazwa      waga    data_przyjecia
    ------    ------    ----    --------------
    szczur    Ziutek    0.35     15-Jan-2021
```

Funkcje II

Variables - ans

ans

1x4 table

	1 typ	2 nazwa	3 waga	4 data_przyjecia	5
1	szczur	Ziutek	0.3500	15-Jan-2021	
2					

2. `table2struct()` konwersja „w drugą stronę”

Zmienne tekstowe

char vs string I

1. Dwie możliwości
   - `a = 'Ala ma kota'`
   - `b = "Ala ma kota"`

```
>> whos
  Name        Size                Bytes  Class     Attributes
  a           1x1                   158  string
  b           1x11                   22  char
```

2. Każdy element tablicy typu `char` zawiera jeden znak
3. Każdy element tablicy typu `string` zawiera jeden napis
4. Napisy mogą być łączone (konkatenowane) z użyciem operatora +
5. `c = string(a)` zamienia char w string

char vs string II

```
>> whos c
  Name      Size              Bytes  Class     Attributes
  c         1x1                 158  string
```

6. W szczególności ta funkcja może być użyta do konwersji liczby na napis

```
>> string(2/3)
ans =
    "0.66667"
```

7. Funkcje `lower()` i `upper()` zamieniają litery „wielkie” w „małe” (lub odwrotnie)

```
>> lower("Ala ma kota") + upper("Ala ma kota")
ans =
    "ala ma kotaALA MA KOTA"
```

char vs string III

8. Funkcja `split()` rozpakowuje napis na wyrazy"

```
>> split("Ala ma kota")
ans =
  3×1 string array
    "Ala"
    "ma"
    "kota"
```

9. Funkcja `double()` zamienia napis na wartość

```
>> double("3.14e20")
ans =
   3.1400e+20
>> double(["3.14e20" "2" "12.5"])
ans =
```

char vs string IV

```
   1.0e+20 *
    3.1400    0.0000    0.0000
>> ans(2)
ans =
     2
>> double(["3.14" "2" "12.5"])
ans =
    3.1400    2.0000   12.5000
```

Obliczenia symboliczne

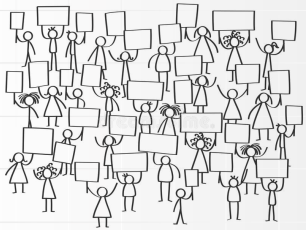
Smbolic Math Toolbox

- Rozwiązywanie równań
- Wykresy
- Przekształcanie wyrażeń matematycznych
- Calculus (całki, pochodne,...)

Politechnika Wrocławska

Symboliczne wartości, zmienne, wyrażenia

Polecenie `sym` służy do tworzenia (deklarowania?) obiektów symbolicznych



Operatory I

W zasadzie podobne jak w przypadku „zwykłych” obliczeń

Arytmetyczne

`minus`	Symbolic subtraction
`plus`	Symbolic addition
`times`	Symbolic array multiplication
`ldivide`	Symbolic array left division
`rdivide`	Symbolic array right division
`power`	Symbolic array power
`nthroot`	Nth root of symbolic numbers
`mtimes`	Symbolic matrix multiplication
`mldivide`	Symbolic matrix left division
`mrdivide`	Symbolic matrix right division

Operatory II

`mpower`	Symbolic matrix power
`transpose`	Symbolic matrix transpose
`ctranspose`	Symbolic matrix complex conjugate transpose

Operatory III

Relacyjne

`eq`	Define symbolic equation
`ge`	Define greater than or equal to condition
`gt`	Define greater than relation
`le`	Define less than or equal to condition
`lt`	Define less than relation
`ne`	Define inequality

Operatory IV

Logiczne

`and`	Logical AND for symbolic expressions
`not`	Logical NOT for symbolic expressions
`or`	Logical OR for symbolic expressions
`xor`	Logical XOR for symbolic expressions

Operatory V

Funkcje

`all`	Test whether all equations and inequalities represented as elements of symbolic array are valid
`any`	Test whether at least one of equations and inequalities represented as elements of symbolic array is valid
`has`	Check if expression contains particular subexpression
`hasSymType`	Determine whether symbolic object contains specific type
`in`	Numeric type of symbolic input

Operatory VI

`isAlways`	Determine if symbolic conditions are true for all values of variables
`isequaln`	Test symbolic objects for equality, treating `NaN` values as equal
`isfinite`	Check whether symbolic array elements are finite
`isinf`	Check whether symbolic array elements are infinite
`isnan`	Check whether symbolic array elements are `NaN`s
`isSymType`	Determine whether symbolic object is specific type
`logical`	Determine if symbolic equation, inequality, or condition is true

Operatory VII

`symtrue`	Symbolic logical constant true *(Since R2020a)*
`symfalse`	Symbolic logical constant false *(Since R2020a)*

Operatory VIII

Operatory modulo

`mod`	Symbolic modulus after division
`powermod`	Modular exponentiation
`quorem`	Quotient and remainder
`rem`	Remainder after division

Operatory IX

Liczby zespolone

`abs`	Symbolic absolute value (complex modulus or magnitude)
`angle`	Symbolic polar angle
`conj`	Complex conjugate of symbolic input
`imag`	Imaginary part of complex number
`real`	Real part of complex number

Granice, pochodne i całki

funkcja	Krótki opis
`limit`	Limit of symbolic expression
`diff`	Differentiate symbolic expression or function
`functionalDerivative`	Functional derivative (variational derivative)
`int`	Definite and indefinite integrals
`vpaintegral`	Numerical integration using variable precision
`changeIntegrationVariable`	Integration by substitution
`integrateByParts`	Integration by parts
`release`	Evaluate integrals

Szeregi I

Series Expansions

funkcja	opis
`pade`	Pade approximant
`rsums`	Interactive evaluation of Riemann sums
`series`	Puiseux series
`taylor`	Taylor series

Szeregi II

Sums & Products

funkcja	opis
`cumprod`	Symbolic cumulative product
`cumsum`	Symbolic cumulative sum
`symprod`	Product of series
`symsum`	Sum of series
`vpasum`	Numerical summation using variable precision

Politechnika Wrocławska

Transformaty

funkcja	opis
`fourier`	Fourier transform
`ifourier`	Inverse Fourier transform
`htrans`	Hilbert transform
`ihtrans`	Inverse Hilbert transform
`laplace`	Laplace transform
`ilaplace`	Inverse Laplace transform
`ztrans`	Z-transform
`iztrans`	Inverse Z-transform
`sympref`	Set symbolic preferences

Politechnika Wrocławska

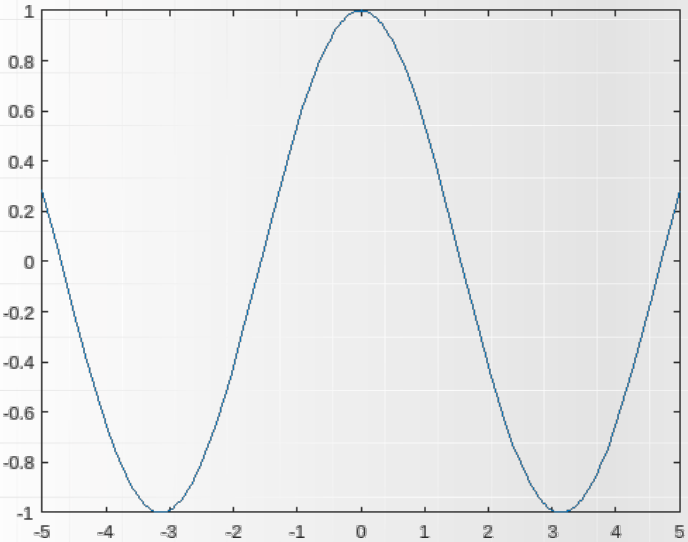
Wykresy funkcji symbolicznych I

Dobra widomośc jest taka, że poprawnie działają funkcjie

- `fplot()`
- `fplot3()`
- `fimplicit()`
- `fimplicit3()`
- `fmesh()`
- `fsurf()`
- `fcontour()`

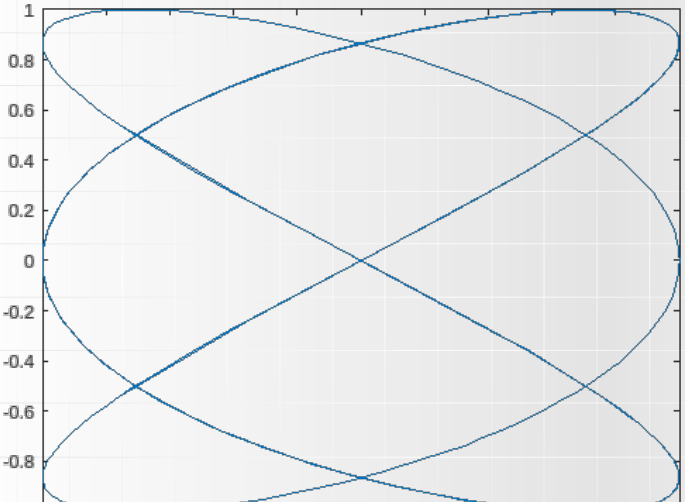
Politechnika Wrocławska

```
syms f(x)
f(x) = cos(x);
fplot(f)
```

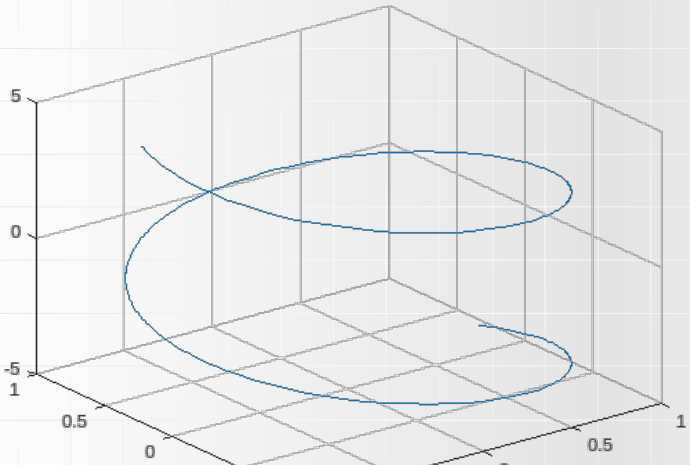


Politechnika Wrocławska

```
syms t
x = cos(3*t);
y = sin(2*t);
fplot(x,y)
```

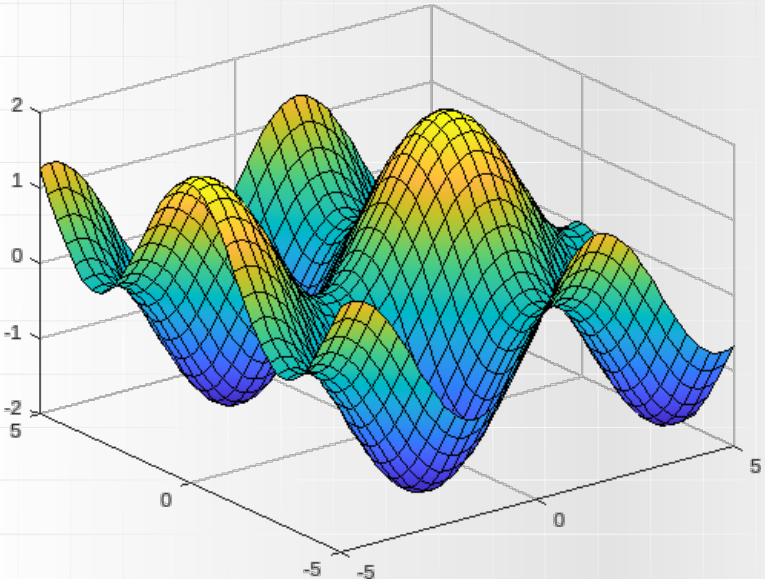


```
syms t
xt = sin(t);
yt = cos(t);
zt = t;
fplot3(xt,yt,zt)
```



Politechnika Wrocławska

```
syms x y
fsurf(sin(x)+cos(y))
```



Rozwiązywanie równań

Funkcja solve() I

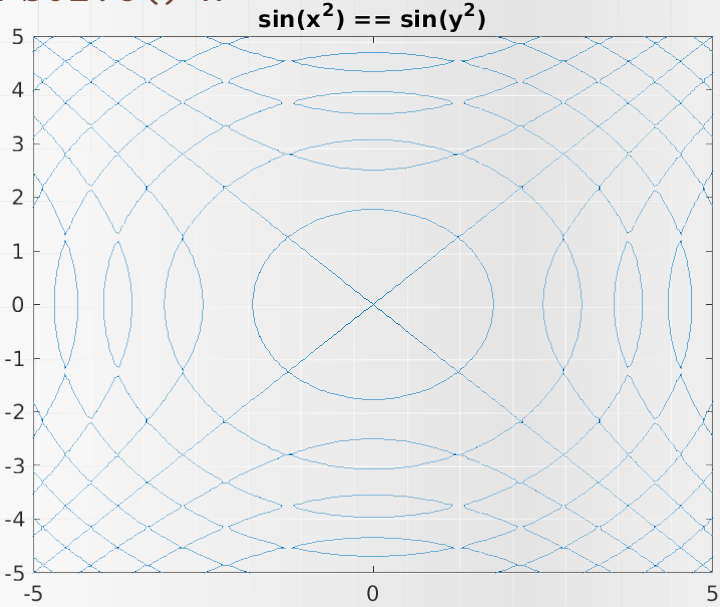
Rozwiązywanie rónań symbolicznych

```
syms x
eqn = sin(x) == cos(x);
solve(eqn,x)
ans =
pi/4
```

Obrazowanie rozwiązania

```
syms x y
eqn = sin(x^2) == sin(y^2);
fimplicit(eqn)
```

Funkcja `solve()` II



Układy równań

Rezultat polecenia `solve()` I

Załóżmy, że mamy układ równań

$$x^2y^2 = 0 \quad x - \frac{y}{2} = \alpha$$

poszukujemy wartości x i y będących rozwiązaniem..

Tworzymy zmienne symboliczne

```
syms x y a
```

Jest kilka sposobów na „wydobycie" rozwiązań.

Pierwszy z nich to przekazać wyniki funkcji jako tablicę

```
[solx,soly] = solve(x^2*y^2 == 0, x-y/2 == a)
```

Rezultat polecenia `solve()` II

$$\text{solx} = \begin{pmatrix} a \\ 0 \end{pmatrix}$$

$$\text{soly} = \begin{pmatrix} 0 \\ -2\,a \end{pmatrix}$$

Jeżeli zmienimy pierwsze równanie na $x^2y^2 = 1$ okaże się, że układ równań ma więcej niż jedno rozwiązanie

```
[solx,soly] = solve(x^2*y^2 == 1, x-y/2 == a)
```

$$\text{solx} = \begin{pmatrix} \frac{a}{2} - \frac{\sqrt{a^2-2}}{2} \\ \frac{a}{2} - \frac{\sqrt{a^2+2}}{2} \\ \frac{a}{2} + \frac{\sqrt{a^2-2}}{2} \\ \frac{a}{2} + \frac{\sqrt{a^2+2}}{2} \end{pmatrix}$$

Rezultat polecenia `solve()` III

$$\text{soly} = \begin{pmatrix} -a - \sqrt{a^2 - 2} \\ -a - \sqrt{a^2 + 2} \\ \sqrt{a^2 - 2} - a \\ \sqrt{a^2 + 2} - a \end{pmatrix}$$

Problem pojawi się w przypadku, gdy rozwiązujemy układ równań o dużej liczbie niewiadomych. Te podejście staje się niewygodne

```
[x1,x2,x3,x4,x5,x6,x7,x8,x9,x10] = solve(...)
```

W takim przypadku, można poprosić o zapisanie rezultatów w jednej zmiennej: będzie to struktura o składowych nazwanych poszczególnymi nazwami zmiennych

Na przykład rozwiązując układ równań

Rezultat polecenia `solve()` IV

$$u^2 - v^2 = a^2, \quad u + v = 1, \quad a^2 - 2 * a = 3$$

.

Funkcja zwraca strukturę:

```
syms u v a
S = solve(u^2 - v^2 == a^2, u + v == 1, a^2 - 2*a == 3)
```

```
S = struct with fields:
    a: [2x1 sym]
    u: [2x1 sym]
    v: [2x1 sym]
```

Zatem rozwiązanie dla a znajdziemy w polu `S.a`

Rezultat polecenia `solve()` V

```
S.a
```

$$\text{ans} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$$

Podobnie w przypadku `u` i `v`. Każda składowa jest tablicą, więc można wybierać poszczególne rozwiązania.

```
s2 = [S.a(2),S.u(2),S.v(2)]
```

$$\text{s2} = \begin{pmatrix} 3 & 5 & -4 \end{pmatrix}$$

lub utworzyć macierz rozwiązań

```
M = [S.a,S.u,S.v]
```

$$\text{M} = \begin{pmatrix} -1 & 1 & 0 \\ 3 & 5 & -4 \end{pmatrix}$$

Równanie liniowe

```
clear u v x y
syms u v x y
eqns = [x + 2*y == u, 4*x + 5*y == v];
S = solve(eqns)
```

```
S = struct with fields:
    x: (2*v)/3 - (5*u)/3
    y: (4*u)/3 - v/3
```

- w przypadku wielu zmiennych „wyjście” funkcji `solve()` to struktura, której składowe są wyrażeniami będącymi rozwiązaniem równania. Zatem można to przerobić na tablicę

```
sol = [S.x;S.y]
```

$$\text{sol} = \begin{pmatrix} \frac{2v}{3} - \frac{5u}{3} \\ \frac{4u}{3} - \frac{v}{3} \end{pmatrix}$$

Równanie kwadratowe I

```
syms a b c d x
eqn = a*x^2 + b*x +c == 0
S = solve(eqn)
S =
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Wszysto znacznie lepiej wygląda w wersji *livescript*

Politechnika Wrocławska

Równanie kwadratowe II

```
syms a b c d x
eqn = a*x^2 + b*x +c == 0
```

eqn =

$$a\,x^2 + b\,x + c = 0$$

```
S = solve(eqn)
```

S =

$$\begin{pmatrix} -\frac{b+\sqrt{b^2-4\,a\,c}}{2\,a} \\ -\frac{b-\sqrt{b^2-4\,a\,c}}{2\,a} \end{pmatrix}$$

Równanie kwadratowe III

```
eqn1 = a*x^3 + b*x^2 + c*x +d == 0
S1 = solve(eqn)
S1 =
root(a*z1^3 + b*z1^2 + c*z1 + d, z1, 1)
root(a*z1^3 + b*z1^2 + c*z1 + d, z1, 2)
root(a*z1^3 + b*z1^2 + c*z1 + d, z1, 3)
```

Czyli kiepsko... Trzeba dodać dodatkowy parametr

```
S1 = solve(eqn1,x,'MaxDegree',3)
```

Ale odpowiedzi nie zacytuję, bo za długa...

Teraz coś prostszego

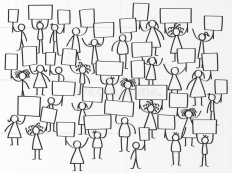
Równanie kwadratowe IV

```
eqn3 = x*x == 2
eqn3 =
x^2 == 2
>> solve(eqn3)
ans =
 2^(1/2)
-2^(1/2)
>> x=sym('x','positive')
x =
x
>> solve(eqn3)
ans =
2^(1/2)
```

bo interesują mnie tylko pierwiastki dodatnie.

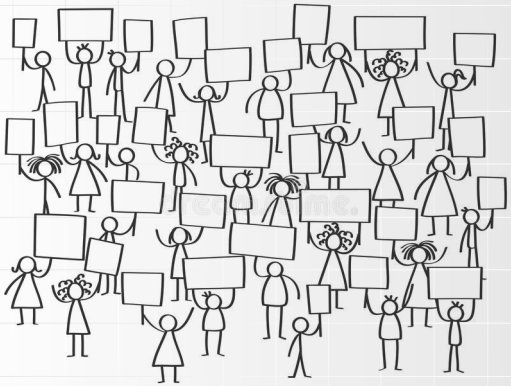
Politechnika Wrocławska

Wielomiany symboliczne



Politechnika Wrocławska

Używanie jednostek podczas obliczeń



Calculus

Politechnika Wrocławska

Granice I

Służy do tego funkcja `limit()`

- `limit(f,var,a)` — $\lim_{var \to a} f(var)$
- `limit(f,a)`— korzysta z domyślnej zmiennej
- `limit(f)` — granica funkcji w zerze
- `limit(f,var,a,'left')` — granica lewostronna
- `limit(f,var,a,'right')` — granica prawostronna

```
syms x h
f = sin(x)/x;
limit(f,x,0)
```

Granice II

ans =

1

A teraz coś bardziej skomplikowanego

```
f = (sin(x+h)-sin(x))/h;
limit(f,h,0)
```

ans = $\cos(x)$

Zmienna „domyślna” I

- do zdefiniowania „zmiennej domyślnej“ służy funkcja `symvar()`
- `symvar(expr)` szuka w wyrażeniu `expr` symboli innych niż `i`, `j`, `pi`, `nan`, `inf`, `eps`
- `C` jest tablicą zawierającą wszystkie zmienne symboliczne

```
C = symvar('cos(pi*x - beta1)')
C =
  2×1 cell array
    {'beta1'}
    {'x'    }
```

- *zmienna domyślna* to zmienna zwracana przez `symvar(fun, 1)`, ale najbliższa brzmieniem do `x`

Zmienna „domyślna” II

```
f=cos(pi*x - beta1)
symvar(f)
ans =
[beta1, x]
symvar(f, 1)
ans =
x
```

Pochodne I

Funkcja `diff()`

- `Df = diff(f)` — pierwsza pochodna po zmiennej domyślnej
- `Df = diff(f,n)` — `n`-ta pochodna po zmiennej domyślnej
- `Df = diff(f,var)` — pochodna po zmiennej `var`
- `Df = diff(f,var,n)` — `n`-ta pochodna po zmiennej `var`
- `Df = diff(f,var1,...,varN)` — pochodna po zmiennych `var`
- `Df = diff(f,mvar)`

```
syms f(x)
f(x) = sin(x^2);
Df = diff(f,x)
```

$$\mathrm{Df}(x) = 2\,x\,\cos\left(x^2\right)$$

Wylicz wartość pochodnej w punkcie `x = 2`.

Pochodne II

```
Df2 = Df(2)
```

Df2 $= 4\cos(4)$

A teraz wartość przybliżóna

```
double(Df2)
```

```
ans = -2.6146
```

Całki I

Funkcja `int()`

- `F = int(expr)`
- `F = int(expr,var)`
- `F = int(expr,a,b)`
- `F = int(expr,var,a,b)`

Dodatkowe parametry można definiować odwołując się do nich przez ich nazwę

```
F = int(___, Name, Value)
```

Może to być istotne w zaawansowanych problemach.

Mamy funkcję dwu zmiennych `x` i `z`.

Całki II

```
syms x z
f(x,z) = x/(1+z^2);
```

Najopierw policzmy całkę po x

```
Fx = int(f,x)
```

$$\mathrm{Fx}(x, z) = \frac{x^2}{2\left(z^2+1\right)}$$

A teraz po z

```
Fz = int(f,z)
```

$$\mathrm{Fz}(x, z) = x\,\mathrm{atan}\left(z\right)$$

Obowiązuje uwaga o zmiennej domyślnej

```
var = symvar(f,1)
```

Całki III

var $= x$

```
F = int(f)
```

$$\mathrm{F}(\mathrm{x},\ \mathrm{z}) = \frac{x^2}{2\left(z^2+1\right)}$$