Laboratorij i vještine - MATLAB

Simbolički paket

Željko Ban
Mato Baotić
Jadranko Matuško

Fakultet elektrotehnike i računarstva

2014/2015

Easy-to-use grafičke funkcije

- Easy-to-use grafičke funkcije namjenjene su prikaz 2D ili 3D matematičkih funkcija zadanih na jedan od sljedećih načina:
  - eksplicitno,
  - implicitno,
  - parametarski.
- Korištenje ovih funkcija ne zahtjeva prethodno definiranje niza vrijednosti odnosno mreže vrijednosti nad kojim se izračunava funkcija koju treba iscrtati.
- Kod easy-to-use funkcija zadavanje funkcije u obliku stringa pretpostavlja da se radi o operacijama na poljima, tj. `’x^2+x^2’` zapravo znači `x.^2+y.^2`.
- imena funkcija su jednaka kao kod standardnih grafičkih funkcija uz dodan prefiks ez

Easy-to-use grafičke funkcije

ezplot

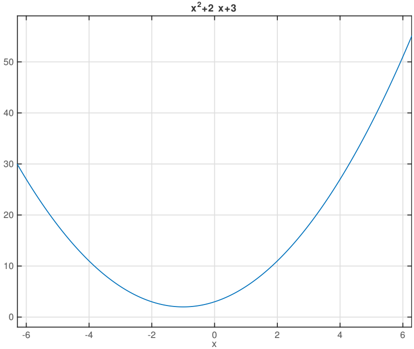
- Funkcija `ezplot(f)` osnovna je funkcija za prikaz 2D funkcija.
- Argument funkcije može biti simbolička funkcija (kasnije će biti pokazano kako se ona definira unutar simboličkog paketa) ili se može izravno zadati kao npr. `ezplot('x^2+2*x+3')`.
- Ukoliko nije eksplicitno drugačije navedeno funkcija se iscrtava na intervalu $[-2\pi, 2\pi]$.
- Funkcija prima dodatne parametre poput granica područja na kojem se iscrtava funkcija, vrsta i boja linije, i slično.

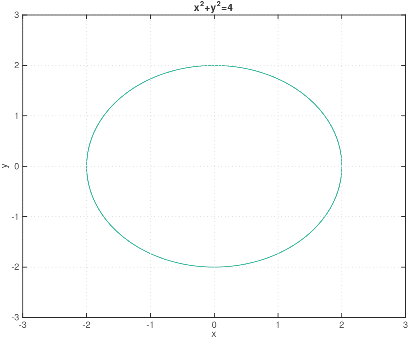
Easy-to-use grafičke funkcije

Primjer: crtanje eksplicitno i implicitno zadane funkcije s funkcijom ezplot

```
>> ezplot('x^2+2*x+3')
>> grid on
```

```
>> ezplot('x^2+y^2=4',...
[-3,3,-3,3])
>> grid on
```



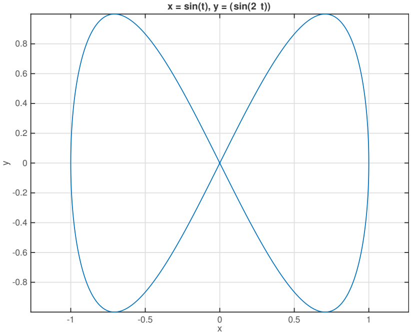


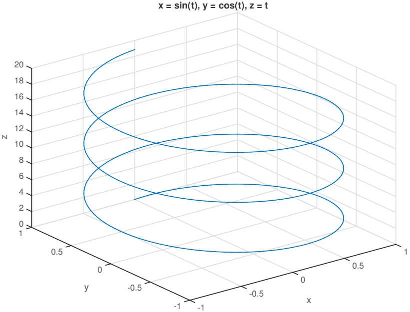
Easy-to-use grafičke funkcije

Primjer: crtanje 2D i 3D parametarski zadane krivulje (ezplot i ezplot3)

```
>> ezplot('sin(t)',...
'(sin(2*t))',[0,2*pi]);
>> grid on;
```

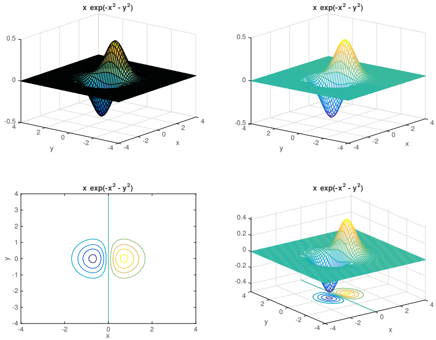
```
>> ezplot3('sin(t)','cos(t)',...
't',[0,6*pi])
```





Easy-to-use grafičke funkcije

```
subplot(221),
ezsurf('x*exp(-x^2 - y^2)',...
[-4,4,-4,4])
subplot(222),
ezmesh('x*exp(-x^2 - y^2)',...
[-4,4,-4,4])
subplot(223),
ezcontour('x*exp(-x^2 - y^2)',...
[-4,4,-4,4])
subplot(224),
ezmeshc('x*exp(-x^2 - y^2)',...
[-4,4,-4,4])
```



Poboljšanje performanci m-funkcija/skripta

m-funkcije vs m-skripta

- Sa stajališta brzine izvođenja preporuča se korištenje m-funkcija u odnosu na m-skripta
- Prilikom izvršavanja m-skripta naredbe se interpretira (engl. interpret) naredba po naredba i prilikom svakog poziva se ovaj postupak ponavlja.
- M-funkcija se prevede (engl. compile) kao cjelina i sprema u memoriju samo prilikom prvog poziva funkcija, dok se prilikom narednih poziva koristi već ranije prevedena funkcija.

Poboljšanje performanci m-funkcija/skripta

Prealokacija memorije

- Korisnik se prilikom pisanja svojeg koda u pravilu ne mora brinuti oko alociranja memorije za pojedine varijable u programu.
- Memorija se dinamički alocira kada se za to ukaže potreba.
- Problem se javlja ako se dimenzija vektora/matrice kontinuirano povećava (npr. unutar for petlje), što rezultira time da se u svakom koraku nanovo alocira dostatan blok memorije te se prethodni sadržaj kopira na novu lokaciju.
- Problem se rješava na način da se potrebni blok memorije unaprijed alocira (prealocira).

Poboljšanje performanci m-funkcija/skripta

Primjer: prealokacija memorije

```
% Bez prealokacije
x = 0;
for k = 2:1000000
   x(k) = x(k-1) + 5;
end
```

```
>> tic,test_mem_aloc1,toc
Elapsed time is 0.214507 seconds.
```

```
% prealokacija verzija 1
x = zeros(1,1000000);
for k = 2:1000000
   x(k) = x(k-1) + 5;
end
```

```
>> tic,test_mem_aloc2,toc
Elapsed time is 0.014794 seconds.
```

```
% prealokacija verzija 2
x(1,1000000)=0;
for k = 2:1000000
   x(k) = x(k-1) + 5;
end
```

```
>> tic,test_mem_aloc3,toc
Elapsed time is 0.012562 seconds.
```

Poboljšanje performanci M-funkcija/skripta

In-place funkcije

- Tipične deklaracije funkcije

```
function y = myfunc (x)
function [a b c] = myfunc (x)
```

- *In-place* deklaracije funkcije

```
function x = myfunc (x)
function [x b c] = myfunc (x)
```

- Najpovoljnija situacija je kada su svi ulazi ujedno i izlazne varijable,
- Funkcija mora biti pozvana s istim ulazom/izlazom.

Poboljšanje performanci M-funkcija/skripta

Vektorizacija koda

- Preferira se korištenje vektoriziranog kod umjesto FOR/WHILE petlji (uz uključen Just-In-Time prevoditelja - predefinirana postavka)

```
function TestFor
y=zeros(1,1000001);
tic;
% Izvedba s for petljom
i = 0;
for t = 0:.00001:10
i = i + 1;
y(i) = sin(t);
end
toc;
```

```
function TestVec
y=zeros(1,1000001);
tic;
% Vektorizirana izvedba
t = 0:.00001:10;
y = sin(t);
toc;
```

```
>> TestFor
Elapsed time is 0.038448 seconds.
>> TestVec
Elapsed time is 0.012799 seconds.
```

Poboljšanje performanci M-funkcija/skripta

Vektorizacija koda

- Uz isključen JIT prevoditelj

```
>> feature accel off
>> TestFor
Elapsed time is 2.212907 seconds.
>> TestVec
Elapsed time is 0.012340 seconds.
```

- U slučaju korištenja samo jedne jezgre, tj. ako pokrenemo Matlab naredbom matlab -singleCompThread dobije se:

```
>> TestFor
Elapsed time is 0.037907 seconds.
>> TestVec
Elapsed time is 0.024861 seconds.
```

Poboljšanje performanci M-funkcija/skripta

Vektorizacija koda korištenjem logičkih polja

Razmotrimo primjer matrice a u kojoj je potrebno korjenovati članove koji su veći od 5, a kvadrirati one koji su manji ili jednaki 5.

Izvedba s FOR petljom

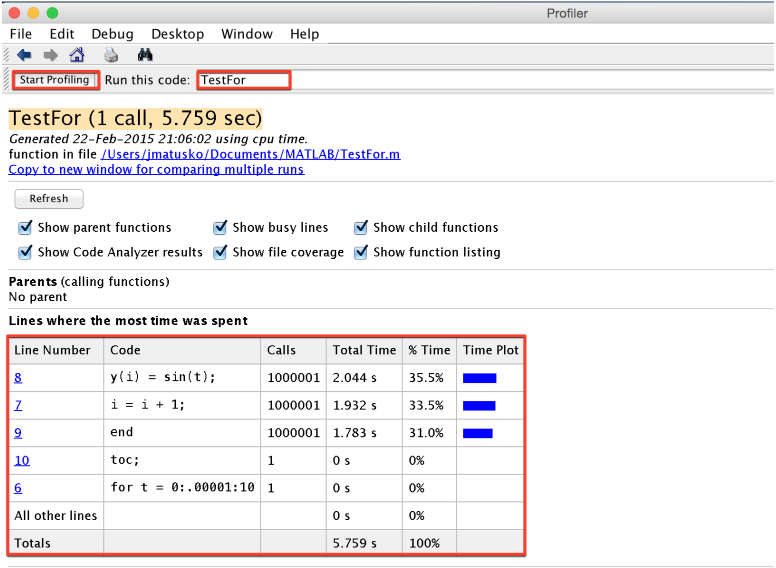
```
for ii = 1:size(a,1)
  for jj = 1:size(a,2)
    if a(ii,jj) > 5
      a(ii,jj) = sqrt(a(ii,jj));
    else
      a(ii,jj) = a(ii,jj)^2;
    end
  end
end
```

Vektorizirana izvedba

```
b = a > 5;
a(b) = sqrt(a(b));
a(~b) = a(~b).^2;
```

Poboljšanje performanci M-funkcija/skripta

Analiza izvođenja funkcije - Alat PROFILER



Line Number	Code	Calls	Total Time	% Time	Time Plot
8	`y(i) = sin(t);`	1000001	2.044 s	35.5%	
7	`i = i + 1;`	1000001	1.932 s	33.5%	
9	`end`	1000001	1.783 s	31.0%	
10	`toc;`	1	0 s	0%	
6	`for t = 0:.00001:10`	1	0 s	0%	
All other lines			0 s	0%	
Totals			5.759 s	100%	

Poboljšanje performanci M-funkcija/skripta

Analiza izvođenja funkcije - Alat PROFILER

Function listing

Color highlight code according to: time

time	calls	line	
		1	`function TestFor`
0.02	1	2	`y=zeros(1,1000001);`
	1	3	`tic;`
		4	`% For loop form`
	1	5	`i = 0;`
	1	6	`for t = 0:.00001:10`
1.78	1000001	7	`i = i + 1;`
1.80	1000001	8	`y(i) = sin(t);`
2.04	1000001	9	`end`
	1	10	`toc;`

Vremenski kritični dio koda

Simbolički paket

- Simbolički paket je opcionalni dodatak programskom sustavu Matlab koji omogućava provođenje sljedećih proračuna u Matlabu:
  - Deriviranje, uključivo i parcijalno deriviranje,
  - Izračun neodređenih i određenih integrala,
  - Proračun graničnih vrijednosti izraza,
  - Provođenje matričnih operacija,
  - Rješavanje algebarskih i diferencijalnih jednadžbi,
  - Provođenje integralnih transformacija.
- Za obavljanje simboličkih proračuna Matlab koristi programski sustav MuPAD®.

Simbolički objekti

Definiranje simboličkog objekta

- Simbolički objekti predstavljaju specijalni tip podataka koji se koristi za simboličke proračune u Matlabu.
- Kreiranje simboličkih objekata obavlja se naredbom sym

```
>> x = sym('x');
>> a = sym('alpha');
```

odnosno skraćeno naredbom syms

```
>> syms x alpha
```

- Na temelju kreiranih symboličkih varijabli može se se kreirati **simbolička funkcija** kao:

```
>> f=x^2+3*y^2;
```

Obavljanje simboličkih proračuna

Uvođenje ograničenja/pretpostavki u simbolički izračun

- Prilikom rješavanja simboličkog problema ponekad je potrebno uvesti određena ograničenja ili pretpostavke na varijable koje se koriste u proračunu, kao npr. da je varijabla realna.

```
>> syms x;
>> solve('x^3+1==0')
ans =
                    -1
 1/2 - (3^(1/2)*i)/2
 (3^(1/2)*i)/2 + 1/2
```

```
>> assume(x,'real');
>> solve('x^3+1==0')
ans =
-1
```

Obavljanje simboličkih proračuna

Pojednostavljenje izraza - simplify

- Matlab simbolički paket sadrži više funkcija koje omogućavanju pojednostavljenja rezultata simboličkih proračuna.
- Osnovna funkcija za pojednostavljivanje izraza je `simplify(f)`

```
>> f=(1/(x+y)-1/(x-y)+2*x/(x^2-y^2))/(1/(x+y))
f =
(x + y)*((2*x)/(x^2 - y^2) - 1/(x - y) + 1/(x + y))
>> simplify(f)
ans =
2
```

```
>> z = sym('(1 + sqrt(5))/2');
>> f = z^2 - z - 1
f =
(5^(1/2)/2 + 1/2)^2 - 5^(1/2)/2 - 3/2
>> simplify(f)
ans =
0
```

Obavljanje simboličkih proračuna

Pojednostavljenje izraza - expand

- Funkcija `expand(f)` transformira izraz u oblik sume umnožaka pojedinih varijabli odnosno elementarnih funkcija.

```
>> syms x
f = (x - 1)*(x - 2)*(x - 3);
>> expand(f)
ans =
x^3 - 6*x^2 + 11*x - 6
```

```
>> syms x y
f = cos(x + y);
>> expand(f)
ans =
cos(x)*cos(y) - sin(x)*sin(y)
```

Obavljanje simboličkih proračuna

Pojednostavljenje izraza - factor

- Funkcija `factor(f)` transformira izraz u oblik umnoška faktora nižih stupnjeva s realnim koeficijentima.

```
>> syms x
f = x^3 - 6*x^2 + 11*x - 6;
>> factor(f)
ans =
[ x - 3, x - 1, x - 2]
```

```
>> syms x
f = x^6 + 1;
>> factor(f)
ans =
[ x^2 + 1, x^4 - x^2 + 1]
```

Obavljanje simboličkih proračuna

Pojednostavljenje izraza - collect

- Funkcija `collect(f,var)` transformira izraz u oblik polinoma po varijabli `var`

```
syms x y
>> f=x^2*y + y*x - x^2 - 2*x;
>> collect(f, x)
ans =
(y - 1)*x^2 + (y - 2)*x

>> collect(f, y)
ans =
(x^2 + x)*y - x^2 - 2*x
```

Obavljanje simboličkih proračuna

Pojednostavljenje izraza - subexpr

- Funkcija `subexpr(f)` pojednostavljuje izraz na način da u njemu pronalazi zajedničke podizraze i zamjenjuje ih s novom simboličkom varijablom.

```
>> syms a b c x
>> solutions = solve(a*x^2 + b*x + c == 0, x)
solutions =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)

>> subexpr(solutions)
sigma =
(b^2 - 4*a*c)^(1/2)
ans =
 -(b + sigma)/(2*a)
 -(b - sigma)/(2*a)
```

Obavljanje simboličkih proračuna

Zamjena varijable/izraza - subs

- Funkcija `subs(f,old,new)` zamjenjuje u simboličkom izrazu `f` varijablu ili podizraz `old` s novom varijablom ili numeričkom vrijednošću `new`.

```
>> syms a b
>> subs(a + b, a, 4)
ans =
b + 4
```

```
>>subs(a*b^2, a*b, 5)
ans =
5*b
```

Obavljanje simboličkih proračuna

Deriviranje izraza - diff

- Funkcija `diff(f,n,var)` proračunava simbolički n-tu derivaciju izraza `f` po varijabli `var`.

```
>> syms x
f(x) = sin(x^2);
>> df = diff(f)
df(x) =
2*x*cos(x^2)
```

```
>> syms x y
>> diff(x*cos(x*y), y, 2)
ans =
-x^3*cos(x*y)
```

Obavljanje simboličkih proračuna

Integriranje simboličkog izraza - int

- Funkcija `int(f,x)` proračunava simbolički integral (neodređeni ili određeni) izraza `f` po varijabli `x`.

```
>> syms x n
f = x^n;
>> int(f)
ans =
piecewise([n == -1, log(x)], [n ~= -1, x^(n + 1)/(n + 1)])
```

```
>> assume(n~= -1)
>> int(f)
ans =
x^(n + 1)/(n + 1)
```

Obavljanje simboličkih proračuna

Integriranje simboličkog izraza - int

- Da bi se proračunao određeni integral potrebno je u funkciji int dodatno definirati granice integracije kao `int(f,x,[a,b])` ili `int(f,[a,b])`.

```
>> syms x
>> a=0; b=1;
>> f = log(x)*sqrt(x);
>> int(f, a, b)
ans =
-4/9
```

Obavljanje simboličkih proračuna

Rješavanje algebarskih jednadžbi - solve

- Funkcija `solve(izraz)` rješava jednadžbu izraz=0 odnosno izraz, ako `izraz` predstavlja jednadžbu.
- Ako je izraz `izraz` funkcija više varijabli tada se dodatnim parametrom specificira po kojoj se varijabli jednadžba rješava kao `solve(izraz,x)`.

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

Obavljanje simboličkih proračuna

Rješavanje algebarskih jednadžbi - solve, višestruka rješenja

- Funkcija `solve(izraz)` ne vraća sva rješenja ako su ona periodička.

```
syms x
solx = solve(cos(x) == -sin(x), x)
solx =
-pi/4
```

- Sva rješenja kao i odgovarajući uvjeti uz koje ta rješenja vrijede dobiju se postavljanjem parametra `ReturnConditions` na vrijednost `'true'`.

```
[solx param uvjet]=solve(cos(x)==-sin(x),x,'ReturnConditions',true)
solx =
pi*k - pi/4
param =
k
uvjet =
in(k, 'integer')
```

Obavljanje simboličkih proračuna

Rješavanje algebarskih jednadžbi - solve, višestruka rješenja

- Ako sada trebamo iz skupa rješenja `solx` pronaći ona na intervalu $-2\pi \leq solx \leq 2\pi$

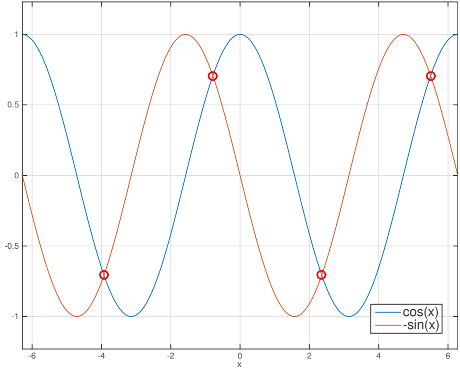
```
>> assume(uvjet)
>> solk = solve(-2*pi<solx, solx<2*pi, k)
solk =
 -1
  0
  1
  2
```

- Primjetimo da se prethodna jednadžba rješava po parametru k, te da bismo dobili rješenja originalne jednadžbe na intervalu $(-\pi,\pi)$ trebamo rješenja `solk` uvrstiti u opće rješenje `solx`(sljedeći slajd).

Obavljanje simboličkih proračuna

Rješavanje algebarskih jednadžbi - solve, vešestruka rješenja

```
>> xvalues = subs(solx, solk)
xvalues =
 -(5*pi)/4
     -pi/4
  (3*pi)/4
  (7*pi)/4
```



Obavljanje simboličkih proračuna

Rješavanje diferencijalnih jednadžbi - dsolve

- Funkcija `dsolve(eqns)` rješava diferencijalnu jednadžbu odnosno sustav diferencijalnih jednadžbi dan s Funkcija `eqns`.

```
>> syms f(t)
dsolve(diff(f) == f + sin(t))
ans =
C5*exp(t) - sin(t)/2 - cos(t)/2
```

- Početni se uvjeti mogu definirati kao dodatni argument funkcije kod njenog poziva kao `dsolve(eqns, conds)`.

```
>> dsolve(diff(f) == f + sin(t),f(0)==0)
ans =
exp(t)/2 - cos(t)/2 - sin(t)/2
```

Obavljanje simboličkih proračuna

Matrične operacije

- Matrice se u simboličkom paketu definiraju na jednak način kao u slučaju numeričkih matrica, navodeći pojedine elemente u uglatim zagradama.

```
>> syms a_11 a_12 a_21 a_22
>> A=[a_11 a_12; a_21 a_22]
A =
[ a_11, a_12]
[ a_21, a_22]
```

- Na ovako definiranu simboličku matricu mogu se primjeniti standardne matrične operacije, što je ilustrirano sljedećim primjerima.

```
>> B=inv(A)
B =
[  a_22/(a_11*a_22 - a_12*a_21), -a_12/(a_11*a_22 - a_12*a_21)]
[ -a_21/(a_11*a_22 - a_12*a_21),  a_11/(a_11*a_22 - a_12*a_21)]
```

Obavljanje simboličkih proračuna

Matrične operacije (2)

- Korištenjem funkcije `suberxpr` prethodni se izraz može sažeto napisati uvodeći prikladnu pokratu.

```
>> subexpr(B)
sigma =
1/(a_11*a_22 - a_12*a_21)
ans =
[  a_22*sigma, -a_12*sigma]
[ -a_21*sigma,  a_11*sigma]
```

- Zajednički faktor u rezultatu prethodne operacije je sigma=1/det(A), što je razvidno iz sljedećeg primjera.

```
>> det(A)
ans =
a_11*a_22 - a_12*a_21
```

Obavljanje simboličkih proračuna

Matrične operacije (3)

- Za danu matricu A mogu proračunati karakteristični polinom (funkcija `charpoly`)

```
>> charpoly(A)
ans =
[ 1, - a_11 - a_22, a_11*a_22 - a_12*a_21]
```

- odnosno karakteristične vrijednosti (funkcija `eig`).

```
>> eig(A)
ans =
a_11/2+a_22/2-(a_11^2-2*a_11*a_22+a_22^2+4*a_12*a_21)^(1/2)/2
a_11/2+a_22/2+(a_11^2-2*a_11*a_22+a_22^2+4*a_12*a_21)^(1/2)/2
```

- Trag matrice A dobije se primjenom funkcije `trace`.

```
>> trace (A)
ans =
a_11 + a_22
```

Obavljanje simboličkih proračuna

Proračun graničnih vrijednosti izraza - limit

- Funkcija `limit(f,var, value)` proračunava graničnu vrijednost izraza `f`, kada varijabla `var` teži vrijednosti `value`

```
>> syms x h
>> limit(sin(x)/x)
ans =
1
>> limit((sin(x + h) - sin(x))/h, h, 0)
ans =
cos(x)
```

- Funkcija također omogućuje proračun jednostranih graničnih vrijednosti (lijeva i desna) na sljedeći način `limit(f,var, value,'left')`

```
>> syms x, limit(1/x,x,0,'left')
ans =
-Inf
```

```
>> syms x, limit(1/x,x,0,'right')
ans =
Inf
```

Obavljanje simboličkih proračuna

Suma reda - sysum

- Funkcija `symsum(f,x,a,b)` proračunava sumu reda čemu pri se varijabla x u izrazu f poprima cjelobrojne vrijednosti od iznosa a do iznosa b.
- Ako se funkcija pozove kao `symsum(f)` tada se proračunava suma reda pri čemu se x mijenja od 0 do $x-1$.

```
>> syms k
>> symsum(k^2)
ans =
k^3/3 - k^2/2 + k/6
```

```
>> symsum(k^2,0,5)
ans =
55
```

Obavljanje simboličkih proračuna

Razvoj u Taylorov red - taylor

- Funkcija `taylor(f,var,a)` razvija simboličku funkciju f po varijabli var oko točke a u Taylorov red do uključivo reda 5 (predefinirana postavka).
- Red ostatka Taylorovog reda može se defnirati kod poziva funkcije kao dodatni parametar funkcije `taylor(f,var,a,'Order',10)`

```
>> syms x
>> taylor(exp(x))
ans =
x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
```

```
>> taylor(exp(x),x,1,'Order',3)
ans =
exp(1) + exp(1)*(x - 1) + (exp(1)*(x - 1)^2)/2
```

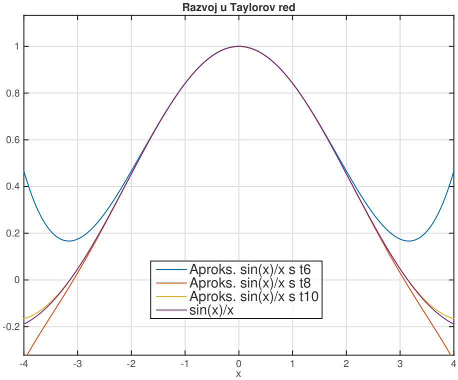
Obavljanje simboličkih proračuna

Primjer: Razvoj u Taylorov red funkcije sin(x)/x

```
>> syms x
>> f=sin(x)/x;
```

```
>> t8 = taylor(f,'Order',8)
t8 =
-x^6/5040+x^4/120-x^2/6+1
>> t10 = taylor(f,'Order',10)
t10 =
x^8/362880-x^6/5040+x^4/120-...
x^2/6+1
```

```
>> ezplot(t6, [-4, 4])
>> hold on
>> ezplot(t8, [-4, 4])
>> ezplot(t10, [-4, 4])
>> ezplot(f, [-4, 4])
legend('Aproks.sin(x)/x s t6',...
'Aproks. sin(x)/x s t8',...
'Aproks. sin(x)/x s t10',...
'sin(x)/x')
title('Razvoj u Taylorov red')
```



Obavljanje simboličkih proračuna

Primjer: pronalaženje asimptota i ekstrema funkcije

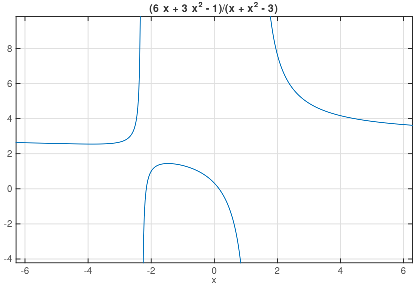
Neka je zadana funkcija:

$$f(x) = \frac{3x^2 + 6x - 1}{x^2 + x - 3}$$

Potrebno je odrediti i ucrtati asimptote, kritične točke i točke infleksije.

- Određivanje asimptota

```
% kreiranje funkcije
syms x
num = 3*x^2 + 6*x -1;
denom = x^2 + x - 3;
f = num/denom;
ezplot(f);
```



Obavljanje simboličkih proračuna

Primjer: pronalaženje asimptota i ekstrema funkcije

- Horizontalnu asimptotu određujemo kao graničnu vrijednost funkcije $f(x)$ kada $x \to \infty$ odnosno $x \to -\infty$.

```
% horizontalna asimptota
>> limit(f, inf)
ans =
3
>> limit(f, -inf)
ans =
3
```

- Vertikalne asimptote tražimo u točkama gdje je nazivnik izraza $f(x)$ jednak nuli.

```
>> sol = solve(denom==0)
sol =
 - 13^(1/2)/2 - 1/2
   13^(1/2)/2 - 1/2
```

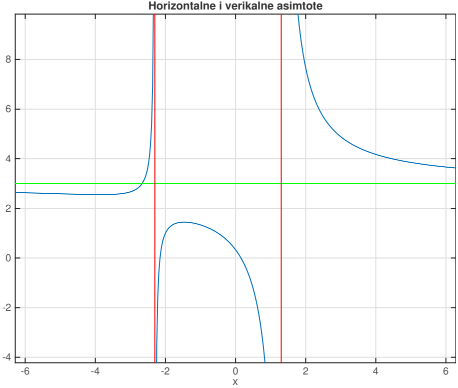
```
>> limit(f,x,sol(1),'left')
ans =
Inf
>> limit(f,x,sol(1),'right')
ans =
-Inf
```

Obavljanje simboličkih proračuna

Primjer: pronalaženje asimptota i ekstrema funkcije

- Crtanje asimptota.

```
ezplot(f)
hold on
% crtanje horizontalnih asimptota
plot([-2*pi 2*pi],...
 [3 3],'g')
% Crtanje vertikalnih asimptota
plot(double(sol(1))*[1 1],...
 [-5 10],'r')
plot(double(sol(2))*[1 1],...
 [-5 10],'r')
title('Horizontalne i ...
vertikalne asimptote')
hold off
```



Obavljanje simboličkih proračuna

Primjer: pronalaženje asimptota i ekstrema funkcije

- Pronalaženje maksimuma i minimuma funkcije

```
f1 = diff(f);
f1 = simplify(f1)
f1 =
-(3*x^2+16*x+17)/(x^2+x-3)^2
```

```
crit_pts = solve(f1==0)
crit_pts =
 - 13^(1/2)/3 - 8/3
   13^(1/2)/3 - 8/3
```

```
ezplot(f), hold on
plot(double(crit_pts),...
 double(subs(f,crit_pts)),'ro')
title('Maksimum i minimum f-je')
text(-5.5,3.2,'Lokalni minimum')
text(-2.5,2,'Lokalni maksimum')
```