Laborator 1 Metode Numerice
- Toolbox-urile Symbolic -

În mediul numeric al Matlab, pentru a beneficia de facilităţile oferite de calculul simbolic, avem la dispoziţie Toolbox-urile Symbolic Math. Există două astfel de toolbox-uri:

a) Symbolic Math Toolbox: permite accesarea pachetului de algebră liniară

b) Extended Symbolic Math: extinde facilităţile înglobate în toolbox-ul precedent şi permite accesul la pachetele de programare şi facilităţi definite de utilizator

Prin intermediul Toolbox-ului Symbolic Math sunt definite un nou tip de date MATLAB, numite obiecte simbolice. Obiectele simbolice sunt utilizate pentru a reprezenta variabile simbolice, expresii sau matrice şi sunt prelucrate prin intermediul aritmeticii raţionale.

Aceste se definesc cu ajutorul declaraţiilor `sym` sau `syms`.

Astfel, putem realiza apeluri de forma:

```
x = sym('x'); % se construieşte o variabilă simbolică x
syms x y z; % sunt construite variabilele simbolice x,y,z
```

Prezentăm în continuare câteva din opţiunile existente în Toolbox-ul Symbolic Math.

Derivarea funcţiilor

În Matlab, o expresie simbolică poate fi derivată cu ajutorul comenzii `diff`. De exemplu, dacă dorim să derivăm în raport cu x funcţia $f(x) = \sin(ax)\cos(bx), a, b \in \mathbb{R}$, procedăm astfel: într-un script Matlab, introducem urmatoarele secvenţe de cod:

```
syms x a b
f=sin(a*x)*cos(b*x);
f_deriv=diff(f,x)
```

urmând ca rezultatul compilării să fie:

```
f_deriv =
```

```
a*cos(a*x)*cos(b*x) - b*sin(a*x)*sin(b*x)
```

De asemenea, putem calcula derivata de ordinul n a unei funcţii f folosind comanda `diff(f,x,n)`, unde `n` este un număr întreg. Dacă dorim să calculăm derivata de ordinul al doilea a funcţiei $f(x) = xe^x$, folosim următoarele secvenţe de cod:

```
syms x a b
f=x*exp(x);
f_deriv_ord2=diff(f,x,2)
```

Rezultatul va fi:

```
f_deriv_ord2 =
 2*exp(x) + x*exp(x)
```

Pentru informaţii suplimentare se pot consulta şi alte opţiuni folosind `help sym/diff` sau `doc sym/diff`.

Integrarea funcţiilor

Pentru a calcula primitiva unei funcţii $f(x)$, folosim comanda `int(f,x)`. Vom calcula primitiva funcţiei $f(x) = e^{2x}\cos(2ax), a \in \mathbb{R}$ (în acest exemplu considerăm liniile de cod scrise direct în fereastra de comandă a Matlab, nu într-un script separat; rămâne la decizia utilizatorului varianta pentru care optează):

```
>> syms a x
>> f=exp(2*x)*cos(2*a*x);
>> int(f,x)

ans =

(exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
```

Dacă în continuare folosim comanda `diff(ans,x)`, nu vom obţine expresia lui `f`, ci o expresie echivalentă. Putem opta pentru o simplificare a expresiei folosind comanda `simple(diff(ans,x))` prin care vom fi informaţi asupra regulilor folosite şi se va afişa expresia iniţială a lui f:

```
>> simple(diff(ans,x))

simplify:
exp(2*x)*cos(2*a*x)
radsimp:
(2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
```

```
   -(exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   simplify(Steps = 100):
   exp(2*x)*cos(2*a*x)
   combine(sincos):
   (2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
   - (exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   combine(sinhcosh):
   (2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
   - (exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   combine(ln):
   (2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
   - (exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   factor:
   exp(2*x)*cos(2*a*x)
   expand:
   (8*exp(2*x)*cos(a*x)^2)/(4*a^2 + 4) - (4*a^2*exp(2*x))/(4*a^2 +
4)
   - (4*exp(2*x))/(4*a^2 + 4) + (8*a^2*exp(2*x)*cos(a*x)^2)/(4*a^2
+ 4)
   combine:
   (2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
   - (exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   rewrite(exp):
   (2*exp(2*x)*(exp(-a*x*2*i) + exp(a*x*2*i) + 2*a*((exp(-a*x*2*i)*i)/2

   - (exp(a*x*2*i)*i)/2)))/(4*a^2 + 4) - (exp(2*x)*(4*a*((exp(-a*x*2*i)*i)/2

   - (exp(a*x*2*i)*i)/2) - 4*a^2*(exp(-a*x*2*i)/2 + exp(a*x*2*i)/2)))
   /(4*a^2 + 4)
   rewrite(sincos):

   - ((cos(x*2*i) - sin(x*2*i)*i)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2
+ 4) + (2*(cos(x*2*i) - sin(x*2*i)*i)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2
+ 4)
   collect(x):
   (2*exp(2*x)*(2*cos(2*a*x) + 2*a*sin(2*a*x)))/(4*a^2 + 4)
    - (exp(2*x)*(4*a*sin(2*a*x) - 4*a^2*cos(2*a*x)))/(4*a^2 + 4)
   ans =
   exp(2*x)*cos(2*a*x).
```

În cazul integralei definite folosim comanda `int(f,x,a,b)`, unde `a` şi `b` sunt limitele de integrare, iar `f` este o funcţie ce depinde de `x`. Calculăm integrala $\int_0^{\pi} x^2 \sin(2x)dx$:

```
   >> clear
```

```
>> syms x
>> int('x^2*sin(2*x)',x,0,pi)

ans =

-pi^2/2.
```

Dacă nu se reuşeşte determinarea integralei sub formă analitică, putem opta pentru una din metodele de aproximare numerică, aşa cum vom proceda în ultimul capitol al lucrării. Astfel, dacă dorim să calculăm $\int_0^{\pi} e^{\sin(2x)}dx$, procedăm astfel:

```
>> clear
>> syms x
>> int('exp(sin(2*x))',x,0,pi)
Warning:  Explicit integral could not
be found.

ans =

int(exp(sin(2*x)), x == 0..pi)

>> quad('exp(sin(2*x))',0,pi)
ans =
 3.9775.
```

Pentru informaţii suplimentare se pot consulta şi alte opţiuni folosind `help sym/int` sau `doc sym/int`.

Serii Taylor

Pentru a genera dezvoltarea în serie Taylor a unei expresii simbolice în jurul unei valori considerate pentru argument, folosim funcţia `taylor(functie,x,x0,'Order',n)`, unde `functie` reprezintă expresia funcţiei ce se doreşte a fi dezvoltată, depinzând de `x`, `n` reprezintă ordinul dorit, iar `x0` valoarea în jurul căreia se face dezvoltarea. Astfel, dacă de exemplu dorim să calculăm dezvoltarea de ordinul 6 a lui e^{2x} în jurul lui $x = 0$, procedăm astfel:

```
>> clear
>> syms x
>> dezv_Taylor=taylor(exp(2*x),x,0,'Order',6)

dezv_Taylor =
```

```
pi - x - (pi - x)^3/6 + (pi - x)^5/120.
```

Dacă dorim o afişare mai apropiată de formatul expresiei din dezvoltarea clasică, prin rezolvare matematică, folosim comanda `pretty(dezv_Taylor)`, care va genera expresia:

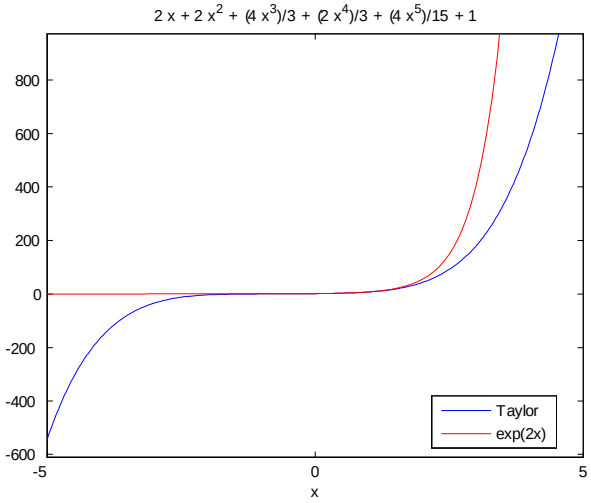
```
4/15x^5+2/3x^4+4/3x^3+2x^2+2x+1.
```

Comparăm în continuare aproximanta pentru $x = 3$ cu valoarea exactă:

```
>> aprox=subs(dezv_Taylor,x,3)
aprox =
899/5
>> val=exp(3)
val =
20.0855
>> err=abs(1-aprox/val)
err =
3625453610879087/455933551425425
>> eval(err)
ans =
7.9517.
```

Putem vizualiza grafic cele două aproximante folosind de exemplu comanda `ezplot()` şi `plot()` pentru reprezentare grafică (vezi Capitolul!!!!), astfel:

```
clear
syms x
dezv_Taylor=taylor(exp(2*x),x,0,'Order',6);
ezplot(dezv_Taylor,[-5,5]), hold on
x=-5:0.1:5;
plot(x,exp(2*x),'r-'), hold off
legend('Taylor','exp(2x)','Location','Best')
```



În secvenţele de cod anterioare am folosit şi comanda `subs()`, care realizează înlocuirea unui parametru cu o valoarea sau cu un alt parametru. Pentru detalii suplimentare legate de comanda subs() sau taylor(), se pot consulta: `help sym/subs` sau `doc sym/subs`, respectiv `help sym/taylor` sau `doc sym/taylor`.

Calculul limitelor de funcţii

Pentru determinarea limitelor de şiruri sau funcţii folosim comanda `limit()` cu diferite forme de apel, după cum se poate constata în `help sym/limit` sau `doc sym/limit.`

Dacă dorim să calculăm $\lim_{x\to 0}\frac{\cos x-1}{x}$, procedăm astfel:

```
>> syms x
>> limit((cos(x)-1)/x,x,0)

ans =

0
```

Cu ajutorul funcţiei implicite `limit()`, avem posibilitatea să determinăm şi limitele laterale ale unei funcţii într-un punct. În secvenţele care urmează, vom determina limita la stânga şi limita la dreapta a funcţiei $f(x)=\frac{\cos x}{x}$:

```
>> syms x
>> ls=limit(cos(x)/x,x,0,'left')

ls =
```

```
-Inf

>> ld=limit(cos(x)/x,x,0,'right')

ld =

Inf
```

Pentru detalii privind modul de utilizare a comenzii `limit()` a se vedea `help sym/limit` sau `doc sym/limit`.

Rezolvarea ecuaţiilor şi sistemelor de ecuaţii

Rezolvarea ecuaţiilor şi sistemelor de ecuaţii se face în Matlab cu ajutorul funcţiei solve(), pentru care avem mai multe variante de apel, în funcţie de caz. Pentru detalii în acest sens, se va putea consulta `help sym/solve` sau `doc sym/solve`. Prezentăm în continuare câteva exemple de implementare.

Dacă de exemplu, se doreşte rezolvarea ecuaţiei $x^3 - 2x^2 + 2x - 4 = 0$, procedăm astfel:

```
>> clear
>> syms x
>> ec='x^3-2*x^2+2*x-4=0';
>> x=solve(ec,x)

x =

 2
 2^(1/2)*i
 -2^(1/2)*i
```

Din cele trei soluţii obţinute, o putem selecta pe cea de-a doua folosind apelul `x(2)`:

```
>> x(2)

ans =

2^(1/2)*i
```

În continuare, dorim să rezolvăm sistemul liniar

$$\begin{cases} x - 2y + 3z = -3 \\ 2x - 2y + z = 2 \\ x - y - 2z = 6. \end{cases}$$

Pentru aceasta, folosim comenzile:

```
>> clear, syms x y z
>> ec1='x-2*y+3*z=-3';
>> ec2='2*x-2*y+z=2';
>> ec3='x-y-2*z=6';
>> [x,y,z]=solve(ec1,ec2,ec3,x,y,z)

x =

1

y =

-1
z =

-2.
```

Este important să reţinem faptul că este importantă ordinea în care sunt date variabilele de ieşire în lista de parametrii.

Dacă se doreşte rezolvarea următorului sistem neliniar

$$\begin{cases} 2xy - x^3 = 0 \\ y = 2 + x^2, \end{cases}$$

procedăm similar şi avem astfel instrucţiunile:

```
>> clear, syms x y
>> ec1='2*x*y-x^3=0';
>> ec2='y=2+x^2';
>> [x,y]=solve(ec1,ec2,x,y)

x =

 0
 2*i
 -2*i


y =

 2
 -2
 -2.
```