Matlab 学习笔记

(Symbolic Math Toolbox 20140523)

1 Symbolic Objects

Symbolic Variables

To declare variables x and y as symbolic objects use the syms command:

```
syms x y
```

Symbolic Numbers

Symbolic Math Toolbox software also enables you to convert numbers to symbolic objects. To create a symbolic number, use the sym command:

```
a = sym('2')
```

If you create a symbolic number with 15 or fewer decimal digits, you can skip the quotes:

```
a = sym(2)
```

```
a = sqrt(sym(2))
```

you get the precise symbolic result:

```
a =

2^(1/2)
```

To evaluate a symbolic number numerically, use the double command:

```
double(a)
```

You also can create a rational fraction involving symbolic numbers:

```
sym(2)/sym(5)

ans =

2/5
```

or more efficiently:

```
sym(2/5)
```

2 Create Symbolic Variables

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

3 Create Symbolic Expressions

```
phi = sym('(1 + sqrt(5))/2');
```

Now suppose you want to study the quadratic function f = ax 2 + bx + c .

```
syms a b c x
f = a*x^2 + b*x + c;
```

4 Create Symbolic Functions

```
syms f(x, y)
```

This syntax creates the symbolic function f and symbolic variables x and y .

Note that `sym` only creates the function. It does not create symbolic variables that represent

its arguments. You must create these variables before creating a function: syms x y;

```
f(x, y) = sym('f(x, y)');
```

After creating a symbolic function, you can differentiate, integrate, or simplify it, substitute its arguments with values, and perform other mathematical operations. For example, find the second derivative on f(x, y) with respect to variable y . The result d2fy is also a symbolic function.

```
d2fy = diff(f, y, 2)
```

Now evaluate f(x, y) for x = y + 1 :

```
f(y + 1, y)
```

5 Create a Matrix of Symbolic Variables

```
syms a b c
A = [a b c; c a b; b c a]
```

or

```
A = sym('A', [2 4])
```

To control the format of the generated names of matrix elements, use %d in the first argument:

```
A = sym('A%d%d', [2 4])

A =

[ A11, A12, A13, A14]

[ A21, A22, A23, A24]
```

6 Create a Matrix of Symbolic Numbers

By applying sym to A

```
A = sym(A)
```

7 Find Symbolic Variables in Expressions, Functions, Matrices

```
syms a b n t x

f = x^n;

g = sin(a*t + b);

symvar(f)

symvar(g, 2)
```

8 Find a Default Symbolic Variable

If you do not specify an independent variable when performing substitution, differentiation, or integration, MATLAB uses a default variable. The default variable is typically the one closest alphabetically to x or, for symbolic functions, the first input argument of a function. To find which variable is chosen as a default variable, use the `symvar(f, 1)` command.

9 Simplify Symbolic Expressions

```
phi = sym('(1 + sqrt(5))/2');

f = phi^2 - phi - 1
```

returns

```
f =
(5^(1/2)/2 + 1/2)^2 - 5^(1/2)/2 – 3/2
```

You can simplify this answer by entering

```
simplify(f)
```

and get a very short answer:

```
ans =
0
```

Symbolic simplification is not always so straightforward. There is no universal simplification function, because the meaning of a simplest representation of a symbolic expression cannot be defined clearly. Different problems require different forms of the same mathematical expression.

```
syms x
f = (x ^2- 1)*(x^4 + x^3 + x^2 + x + 1)*(x^4 - x^3 + x^2 - x + 1);
expand(f)
g = x^3 + 6*x^2 + 11*x + 6;
factor(g)
h = x^5 + x^4 + x^3 + x^2 + x;
horner(h)
```

10 Substitutions in Symbolic Expressions

Substitute Symbolic Variables with Numbers

You can substitute a symbolic variable with a numeric value by using the `subs` function. For example, evaluate the symbolic expression f at the point x = 1/3:

```
syms x
f = 2*x^2 - 3*x + 1;
subs(f, 1/3)
ans =
2/9
```

Substitute in Multivariate Expressions

When your expression contains more than one variable, you can specify the variable for which you want to make the substitution. For example, to substitute the value x = 3 in the symbolic expression

```
syms x y
```

```
f = x^2*y + 5*x*sqrt(y);
```

enter the command

```
subs(f, x, 3)

ans =

9*y + 15*y^(1/2)
```

Substitute One Symbolic Variable for Another

```
subs(f, y, x)
```

Substitute a Matrix into a Polynomial

Element-by-Element Substitution. To substitute a matrix at each element, use the subs command:

```
syms x

f = x^3 - 15*x^2 - 24*x + 350;

A = [1 2 3; 4 5 6];

subs(f,A)
```

Substitution in a Matrix Sense.

```
syms x

f = x^3 - 15*x^2 - 24*x + 350;

A = magic(3);

b = sym2poly(f) % Get a row vector containing the numeric coefficients of the polynomial f

polyvalm(b,A)
```

Substitute the Elements of a Symbolic Matrix

```
syms a b c

A = [a b c; c a b; b c a]

alpha = sym('alpha');

beta = sym('beta');

A(2,1) = beta;

A = subs(A,b,alpha)
```

11 Differentiate Symbolic Expressions

Expressions with One Variable

```
syms x

f = sin(x)^2;

diff(f)
```

Partial Derivatives

```
syms x y

f = sin(x)^2 + cos(y)^2;

diff(f)

diff(f, y)
```

Second Partial and Mixed Derivatives

```
syms x y

f = sin(x)^2 + cos(y)^2;

diff(f, y, 2)

diff(diff(f, y), x)
```

12 Integrate Symbolic Expressions

You can perform symbolic integration including:

• Indefinite and definite integration

• Integration of multivariable expressions

Indefinite Integrals of One-Variable Expressions

```
syms x

f = sin(x)^2;
```

To find the indefinite integral, enter

```
int(f)

ans =

x/2 – sin(2*x)/4
```

Indefinite Integrals of Multivariable Expressions

If the expression depends on multiple symbolic variables, you can designate a variable of integration. If you do not specify any variable, MATLAB chooses a default variable by the proximity to the letter x :

```
syms x y n

f = x^n + y^n;

int(f)
```

You also can integrate the expression f = x^n + y^n with respect to y

```
int(f, y)
```

Definite Integrals

To find a definite integral, pass the limits of integration as the final two arguments of the int function:

```
syms x y n

f = x^n + y^n;

int(f, 1, 10)
```

13 Solve Equations

Solve Algebraic Equations with One Symbolic Variable

Use the double equal sign (==) to define an equation.

```
syms x

solve(x^3 - 6*x^2 == 6 – 11*x)

ans =

1

2

3
```

If you do not specify the right side of the equation, solve assumes that it is zero:

```
syms x

solve(x^3 - 6*x^2 + 11*x – 6)
```

Solve Algebraic Equations with Several Symbolic Variables

If an equation contains several symbolic variables, you can specify a variable for which this equation should be solved.

```
syms x y

solve(6*x^2 - 6*x^2*y + x*y^2 - x*y + y^3 - y^2 == 0, y)

ans =

1

2*x

-3*x
```

Solve Systems of Algebraic Equations

You also can solve systems of equations. For example:

```
syms x y z

[x, y, z] = solve(z == 4*x, x == y, z == x^2 + y^2)
```

14 Create Plots of Symbolic Functions

Explicit Function Plot

The simplest way to create a plot is to use the `ezplot` command:

```
syms x

ezplot(x^3 - 6*x^2 + 11*x - 6)

hold on
```

Implicit Function Plot

Using ezplot , you can also plot equations. For example, plot the following equation over –1 < x < 1:

```
syms x y

ezplot((x^2 + y^2)^4 == (x^2 - y^2)^2, [-1 1])
```

3-D Plot

3-D graphics is also available in Symbolic Math Toolbox. To create a 3-D plot, use the `ezplot3` command.

```
syms t

ezplot3(t^2*sin(10*t), t^2*cos(10*t), t)
```

Surface Plot

If you want to create a surface plot, use the ezsurf command. For example, to plot a paraboloid z = x^2 + y^2 , enter:

```
syms x y

ezsurf(x^2 + y^2)

hold on

zlabel('z')

title('z = x^2 + y^2')

hold off
```

15 Assumptions on Symbolic Objects

Default Assumption

In Symbolic Math Toolbox, symbolic variables are complex variables by default.

If z is complex, assumptions(z) returns an empty symbolic object:

```
assumptions(z)
```

Set Assumptions

```
syms x

assume(x >= 0)
```

`assume` replaces all previous assumptions on the variable with the new assumption. If you want to add a new assumption to the existing assumptions, use `assumeAlso`.

```
assumeAlso(x,'integer')
```

Alternatively, you can set an assumption while declaring a symbolic variable using `sym` or `syms` .

```
a = sym('a', 'real');

b = sym('b', 'real');

c = sym('c', 'positive');
```

or more efficiently:

```
syms a b real

syms c positive
```

Check Existing Assumptions

```
assumptions(x)
```

```
assumptions
```

Delete Symbolic Objects and Their Assumptions

When you delete a symbolic object from the MATLAB workspace using

```
clear x
```

the assumption that x is real stays in the symbolic engine. If you declare a new symbolic variable x later, it inherits the assumption that x is real instead of getting a default assumption. If later you solve an equation and simplify an expression with the symbolic variable x , you could get incomplete results.

For example, the assumption that x is real causes the polynomial x 2 + 1 to have no roots:

```
syms x real

clear x

syms x

solve(x^2 + 1 == 0, x)

Warning: Explicit solution could not be found.

> In solve at 81

ans =

[ empty sym ]
```

To clear the assumption, enter

```
syms x clear
```

After you clear the assumption, the symbolic object stays in the MATLAB workspace. If you want to remove both the symbolic object and its assumption, use two subsequent commands:

1 To clear the assumption, enter

```
syms x clear
```

2 To delete the symbolic object, enter

```
clear x;
```

16 Limits

Symbolic Math Toolbox software enables you to calculate the limits of functions directly. The commands

```
syms h n x
limit((cos(x+h) - cos(x))/h, h, 0)
```

which return

```
ans =
-sin(x)
```

and

```
limit((1 + x/n)^n, n, inf)
```

which returns

```
ans =
exp(x)
```

17 Integration

`int`

18 Symbolic Summation

You can compute symbolic summations, when they exist, by using the `symsum` command.

```
syms x k
s1 = symsum(1/k^2, 1, inf)
s2 = symsum(x^k, k, 0, inf) % 1 + x + x^2 + x^3 + …
```

19 Taylor Series

The statements

```
syms x
f = 1/(5 + 4*cos(x));
T = taylor(f, 'Order', 8)
```

return

```
T =
(49*x^6)/131220 + (5*x^4)/1458 + (2*x^2)/81 + 1/9
```

which is all the terms up to, but not including, order eight in the Taylor series for f(x).

The command

```
pretty(T)
```

prints T in a format resembling typeset mathematics.

20 Simplifications

This toolbox provides several functions that apply various algebraic and trigonometric identities to transform one representation of a function into another, possibly simpler, representation. These functions are `collect` , `expand` , `horner` , `factor` , `simplifyFraction` and `simplify` .

21 Substitute with subexpr

```
syms a x
```

```
s = solve(x^3 + a*x + 1)
```

Continuing with the example

```
r = subexpr(s)
```

22 Control Accuracy of Variable-Precision Computations

```
vpa(sym(1/3) + 1/2)
```

23 Jordan Canonical Form

```
J = jordan(A)
```

24 Singular Value Decomposition

```
svd(A)
```

25 Solve a System of Algebraic Equations

```
help solve
```

```
syms u v x y
```

```
S = solve(x + 2*y == u, 4*x + 5*y == v);
```

26 Solve a Single Differential Equation

Use `dsolve` to compute symbolic solutions to ordinary differential equations.

You can specify the equations as symbolic expressions containing `diff` or as strings with the letter D to indicate differentiation.

First-Order Linear ODE

Suppose you want to solve the equation y'(t) = t*y . First, create the symbolic function y(t) :

```
syms y(t)
```

Now use dsolve to solve the equation:

```
y(t) = dsolve(diff(y) == t*y)
```

Solve the same ordinary differential equation, but now specify the initial condition y(0) = 2 :

```
syms y(t)
y(t) = dsolve(diff(y) == t*y, y(0) == 2)
```

Nonlinear ODE

Nonlinear equations can have multiple solutions, even if you specify initial conditions. For example, solve this equation:

```
syms x(t)
x(t) = dsolve((diff(x) + x)^2 == 1, x(0) == 0)
```

Second-Order ODE with Initial Conditions

```
syms
Dy = diff(y);
y(x) = dsolve(diff(y, 2) == cos(2*x) - y, y(0) == 1, Dy(0) == 0);
y(x) = simplify(y)
```

Third-Order ODE

```
syms u(x)
```

```
Du = diff(u);

D2u = diff(u, 2);

u(x) = dsolve(diff(u, 3) == u, u(0) == 1, Du(0)

== -1, D2u(0) == pi)
```

27 Solve a System of Differential Equations

```
syms f(t) g(t)

S = dsolve(diff(f) == 3*f + 4*g, diff(g) == -4*f + 3*g)
```

28 Compute Fourier and Inverse Fourier Transforms

The Fourier transform of a function f(x) is defined as

$$F[f](w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx,$$

and the inverse Fourier transform (IFT) as

$$F^{-1}[f](x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(w)e^{iwx}dw.$$

For example, consider the Fourier transform of the Cauchy density function,

(π(1 + x ^2)) ^–1 :

```
syms x

cauchy = 1/(pi*(1+x^2));

fcauchy = fourier(cauchy)
```

To recover the Cauchy density function from the Fourier transform, call `ifourier` :

```
finvfcauchy = ifourier(fcauchy)
```

29 Compute Laplace and Inverse Laplace Transforms

`Laplace`

`ilaplace`

30 Compute Z-Transforms and Inverse Z-Transforms

`ztrans`

`iztrans`

31 Create Plots

`ezplot`

`ezpolar`

`scatter`

`sphere`

```
syms x

h = matlabFunction(exp(x/2)*sin(10*x));

fplot(h, [0 10])

hold on

title('exp(x/2)*sin(10*x)')

hold off
```

32 Generate C or Fortran Code

To generate code from a symbolic expression g , enter either `ccode`(g) or `fortran`(g) .

For example:

```
syms x y

z = 30*x^4/(x*y^2 + 10) - x^3*(y^2 + 1)^2;

fortran(z)
```

To generate a file containing code, either enter `ccode(g,'file',' filename ')` or `fortran(g,'file',' filename ')` .

33 Generating a Function Handle

`matlabFunction` can generate a function handle from any symbolic expression. For example:

```
syms x y
```

```
r = sqrt(x^2 + y^2);
ht = matlabFunction(tanh(r))
```

You can use this function handle to calculate numerically:

```
ht(.5,.5)
```

Control the Order of Variables

`matlabFunction` generates input variables in alphabetical order from a symbolic expression.

You can specify the order of input variables in the function handle using the `vars` option.

```
syms x y z
r = sqrt(x^2 + 3*y^2 + 5*z^2);
ht1 = matlabFunction(tanh(r), 'vars', [y x z])
ht2 = matlabFunction(tanh(r), 'vars', {'x', 'y', 'z'})
ht3 = matlabFunction(tanh(r), 'vars', {'x', [y z]})
```

Generate a File

```
syms x y t
z = (x^3 - tan(y))/(x^3 + tan(y));
w = z/(1 + t^2);
F = [w,(1 + t^2)*x/y; (1 + t^2)*x/y,3*z - 1];
matlabFunction(F,'file','testMatrix.m')
```

Name Output Variables

To customize the names of output variables, use the `output` option:

```
syms x y z
r = x^2 + y^2 + z^2;
q = x^2 - y^2 - z^2;
f = matlabFunction(r, q, 'file', 'new_function',...
'outputs', {'name1','name2'})
```

The generated function returns name1 and name2 as results:

```
function [name1,name2] = new_function(x,y,z)
…
```

34 Generate MATLAB Function Blocks

Before you can convert a symbolic expression to a MATLAB Function block, create an empty model or open an existing one:

```
new_system('my_system')

open_system('my_system')
```

Create a symbolic expression and pass it to the matlabFunctionBlock command. Also specify the block name:

```
syms x y

r = sqrt(x^2 + y^2);

matlabFunctionBlock('my_system/my_block', r)
```

35 Special Functions of Applied Mathematics

You can enter the command

```
mfunlist
```

to see the list of functions available for `mfun` .

```
x = -50:50;

C = mfun('FresnelC',x);

S = mfun('FresnelS',x);
```

36 MuPAD in Symbolic Math Toolbox

A MuPAD engine is a separate process that runs on your computer in addition to a MATLAB process. A MuPAD engine starts when you first call a function that needs a symbolic engine, such as `syms` .

```
mupad

mupadwelcome
```