7 SYMBOLIC COMPUTATIONS

Table of Contents

Initialization

```
% reduce needless whitespace
format compact
% reduce irritations
more off
% start a diary
%diary lectureN.txt
```

INTRODUCTION

Normally Matlab generates *numbers*, not *formulae*. If you ask Matlab to find the roots where a function is zero, maybe using `fzero`, it gives you numbers. If you ask Matlab to integrate a function from a lower limit to an upper limit using `integral` or `quad`, it gives you a number. Etcetera,

But sometimes you want a formula instead of a number. For example, you might want the derivative or antiderivative of a function. Either one is a formula, not a number. Also, sometimes you want to see an *exact* number, and non-integer numbers in Matlab have round-off errors.

What you need for such purposes is a *symbolic computations* program. Such a program is available inside Matlab as the "Symbolic Math Toolbox". Normally MathWorks charges separately for this package. However, it is included in the "Student Edition", and within Matlab on the COE computers.

In this section we will illustrate how normal Matlab computations and symbolic ones differ. We will look at a simple function, a quadratic in fact.

One thing to remember: Be sure to inform Matlab with the `syms` or `sym` command when variables and/or numbers are intended to be symbolic. Normal variables are names of storage locations with a number in it. But a symbolic variable does not store a number; at all times it can stand for *any* number. So a normal variable `x` is very different from a symbolic variable `x`, and Matlab must know which of the two `x` is.

```
% the example quadratic as a normal Matlab function
qNum = @(x) -4*x.^2+3*x+12
```

```
% we can integrate it between, say, 0 and 2
qNumInt02=integral(qNum,0,2)
disp('We got a floating point number.')

% find a root
qNumRoot=fzero(qNum,1)
disp('We got a floating point number.')

% tell Matlab that, from now on, x is a symbolic variable
syms x

% the example quadratic now as a symbolic function
qSym=-4*x^2+3*x+12

% we can integrate it between, say, 0 and 2
qSymInt02=int(qSym,x,[0 2])
disp('We got an exact number.')

% but we can also find the anti-derivative
qSymInt=int(qSym,x)
disp('We got a function, the anti-derivative.')
disp('Reformat this with "expand":')
expand(qSymInt)

% and we can find the derivative
qSymDiff=diff(qSym,x)
disp('We got a function, the derivative.')

% we can find both roots (note ==, not =)
qSymRoots=solve(qSym == 0,x)
disp('We got both roots as symbolic expressions.')
disp('To see the numbers, use "double":')
double(qSymRoots)

% we can factor the quadratic, but with some trouble
qSymFactors=factor(qSym)
disp('Oops!  Try "help sym/factor", not "help factor".')
%help sym/factor
qSymFactors=factor(qSym,'FactorMode','full')
disp('Use "prod" to combine the factors:')
prod(qSymFactors)

% we can easily factor a quadratic with rational roots
qSymRat=-4*x^2+3*x+27
qSymRatFactors=factor(qSymRat)
```

```
qNum =
    @(x)-4*x.^2+3*x+12
qNumInt02 =
   19.3333
We got a floating point number.
qNumRoot =
    2.1472
We got a floating point number.
```

```
qSym =
- 4*x^2 + 3*x + 12
qSymInt02 =
58/3
We got an exact number.
qSymInt =
(x*(- 8*x^2 + 9*x + 72))/6
We got a function, the anti-derivative.
Reformat this with "expand":
ans =
- (4*x^3)/3 + (3*x^2)/2 + 12*x
qSymDiff =
3 - 8*x
We got a function, the derivative.
qSymRoots =
 3/8 - 201^(1/2)/8
 201^(1/2)/8 + 3/8
We got both roots as symbolic expressions.
To see the numbers, use "double":
ans =
   -1.3972
    2.1472
qSymFactors =
[ -1, 4*x^2 - 3*x - 12]
Oops!  Try "help sym/factor", not "help factor".
qSymFactors =
[ -1, x + 201^(1/2)/8 - 3/8, x - 201^(1/2)/8 - 3/8]
Use "prod" to combine the factors:
ans =
(x + 201^(1/2)/8 - 3/8)*(201^(1/2)/8 - x + 3/8)
qSymRat =
- 4*x^2 + 3*x + 27
qSymRatFactors =
[ -1, 4*x + 9, x - 3]
```

SIMPLIFYING ANSWERS

In classes like Analysis in Mechanical Engineering, you are required to simplify your answers. Symbolic math to the rescue!

Watch it, however. If you try to simplify, say, a numeric ratio like 629/969 as

```
simplify(629/969)
```

then Matlab sees "629/969", evaluates that as 0.9491... and gives that to the symbolic `simplify` function. Of course `simplify` cannot make any sense out of 0.9491.... In fact, it will refuse to cooperate. What you need to do is tell Matlab that the entire "629/969" is to be treated as a symbolic expression, to be given to `simplify` "as is". You can do that with the `sym` function.

```
% not so easy to see that 17 is a common factor
%ratioSimplified=simplify(629/969)
ratioSimplified=simplify(sym('629/969'))
```

```
ratioSimplified =
```

```
37/57
```

VERY HIGH ACCURACY

Function vpa, ("variable precision arithmetic"), will give you numbers to arbitrarily high accuracy.

But watch it again.

```
% Matlab gives vpa a number equal to pi^2/6 to 16 digits:
disp('vpa(pi^2/6) is wrong:')
piSqOver6=vpa(pi^2/6,50)

% Matlab gives vpa the symbolic string pi^2/6:
disp('vpa(sym(''pi^2/6'')) is correct:')
piSqOver6=vpa(sym('pi^2/6'),50)
```

```
vpa(pi^2/6) is wrong:
piSqOver6 =
1.6449340668482264060656916626612655818462371826172
vpa(sym('pi^2/6')) is correct:
piSqOver6 =
1.6449340668482264364724151666460251892189499012068
```

SOLVING EQUATIONS

The Symbolic Toolbox can solve quite a lot of equations exactly.

If it cannot, it will drop back to a numerical solution.

Solving quadratic equations

Suppose you no longer remembered the solution to the quadratic equation

$$ax^2 + bx + c = 0$$

The symbolic toolbox can give it to you

```
% tell Matlab that the variables are symbolic
syms a b c x qGen

% define the generic quadratic polynomial
qGen=a*x^2+b*x+c

% find the symbolic solution
qGenRoots=solve(qGen == 0,x)
disp('Reformat using "pretty":')
pretty(qGenRoots)

% see whether we get back our previous solution
qGenRootsTest=subs(qGenRoots,{a b c},{-4 3 12})

% we can create a normal Matlab function for the roots
```

```
qGenRootsFun=matlabFunction(qGenRoots)

% test it
qGenRootsFunTest=qGenRootsFun(-4,3,12)
```

```
qGen =
a*x^2 + b*x + c
qGenRoots =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
Reformat using "pretty":
/             2          \
|   b + sqrt(b  - 4 a c) |
| - -------------------- |
|            2 a         |
|                        |
|             2          |
|   b - sqrt(b  - 4 a c) |
| - -------------------- |
\            2 a         /

qGenRootsTest =
 201^(1/2)/8 + 3/8
 3/8 - 201^(1/2)/8
qGenRootsFun =
    @(a,b,c)[(b.*(-1.0./2.0)-sqrt(a.*c.*-4.0+b.^2).*(1.0./2.0))./a;
(b.*(-1.0./2.0)+sqrt(a.*c.*-4.0+b.^2).*(1.0./2.0))./a]
qGenRootsFunTest =
    2.1472
   -1.3972
```

Solving other equations

Some other equations Matlab manages to solve

```
% make sure x and y are symbols
syms x y

% exponentials can be negative for complex arguments
expSol=solve(exp(x) == -1,x)

% the next expression to solve, but we will write it out
qxySym=a*x*y - b*x - 1

% a x y - b x - 1 can be solved for either x or y
xSol=solve(a*x*y - b*x - 1 == 0, x)
ySol=solve(a*x*y - b*x - 1 == 0, y)

% sometimes rewriting the equation can help
xCoefs=collect(5*x*y-9*x-1,x)

% solve can solve systems of equations
[xSol,ySol] = solve(x+y == 7, x-y == 1',x,y)
```

```
% if it cannot find an exact solution, it switches to numerical
[xSol,ySol] = solve(x^2+cos(y) == 7, cosh(x)-y == 1,x,y)
```

```
expSol =
pi*1i
qxySym =
a*x*y - b*x - 1
xSol =
-1/(b - a*y)
ySol =
(b*x + 1)/(a*x)
xCoefs =
(5*y - 9)*x - 1
xSol =
4
ySol =
3
Warning: Cannot solve symbolically. Returning a numeric approximation
 instead.
xSol =
-2.533048428764151769367806268 3806
ySol =
5.335624774863588094906010707 8797
```

PARTIAL FRACTIONS

In analyzing the dynamics of controlled systems, you often encounter ratios of big polynomials. Then you want to take these ratios apart in simpler fractions. The reason is that these simpler fractions tell you many important things. For example, they tell you whether, if the system is disturbed, it will return to its normal position, versus, say, crash. And, if so, they will also tell you how fast the system will return to normal.

Symbolic function `partfrac` can do it.

```
% make sure x is still symbolic
syms x

% example symbolic ratio
ratSym=(2*x^2-3*x+1)/(x^3+2*x^2-9*x-18)

% you could first look at the factors
ratSymFactors=factor(ratSym)
disp('Multiply out:')
prod(ratSymFactors)

% get the partial fractions
ratPartFrac=partfrac(ratSym)
```

```
ratSym =
-(2*x^2 - 3*x + 1)/(- x^3 - 2*x^2 + 9*x + 18)
ratSymFactors =
[ 2*x - 1, x - 1, 1/(x - 3), 1/(x + 3), 1/(x + 2)]
Multiply out:
ans =
((2*x - 1)*(x - 1))/((x + 2)*(x - 3)*(x + 3))
```

```
ratPartFrac =
1/(3*(x - 3)) - 3/(x + 2) + 14/(3*(x + 3))
```

FUNCTION MANIPULATIONS

Below are some more example manipulations

```
% make sure x, etc, are still symbolic
syms x a b c

% dealing with an integrals you cannot do
intHard1=int(1/(x*(a*x^2+b*x+c)^(1/2)))
intHard2=int(1/(x*(a*x^2+b*x+c)^(3/2)))
intHard2=int(1/(x*(a*x^2+b*x+c)^(3/2)), ...
             'IgnoreSpecialCases',1,'IgnoreAnalyticConstraints',1)
disp('Do not pay too much for the Toolbox.')

% sometimes int is useful to identify a function
intUnk=int(sin(x)/x,x)

% writing the Taylor series of, say, sinint
sinxOverxTaylor=taylor(sin(x)/x)
sinxOverxTaylor=taylor(sin(x)/x,'Order',10)
sinintTaylor=int(sinxOverxTaylor)

% see the logic
disp('Examine t(3)/t(1):')
simplify(-(x^3/18)/x)
factor(18)
disp('Examine t(5)/t(3):')
simplify(-(x^5/600)/(x^3/18))
factor(100)
disp('Examine t(7)/t(5):')
simplify(-(x^7/35280)/(x^5/600))
factor(294)
disp('Examine t(9)/t(7):')
simplify(-(x^9/3265920)/(x^7/35280))
factor(648)

% Matlab has "funtool" to play with functions:
%funtool
```

```
intHard1 =
-log(b/2 + c/x + (c^(1/2)*(a*x^2 + b*x + c)^(1/2))/x)/c^(1/2)
intHard2 =
int(1/(x*(a*x^2 + b*x + c)^(3/2)), x)
intHard2 =
int(1/(x*(a*x^2 + b*x + c)^(3/2)), x, 'IgnoreSpecialCases', true,
 'IgnoreAnalyticConstraints', true)
Do not pay too much for the Toolbox.
intUnk =
sinint(x)
sinxOverxTaylor =
x^4/120 - x^2/6 + 1
```

```
sinxOverxTaylor =
x^8/362880 - x^6/5040 + x^4/120 - x^2/6 + 1
sinintTaylor =
x^9/3265920 - x^7/35280 + x^5/600 - x^3/18 + x
Examine t(3)/t(1):
ans =
-x^2/18
ans =
     2     3     3
Examine t(5)/t(3):
ans =
-(3*x^2)/100
ans =
     2     2     5     5
Examine t(7)/t(5):
ans =
-(5*x^2)/294
ans =
     2     3     7     7
Examine t(9)/t(7):
ans =
-(7*x^2)/648
ans =
     2     2     2     3     3     3     3
```

End lesson 7

Published with MATLAB® R2015b