MATLAB

Lecture 8

Symbolic Algebra

Matlab allows to do symbolic algebra

Learn by examples:

factorising

log-linearisation

Symbolic Algebra

primes

Generate list of prime numbers

p = primes(n)

p = primes(n) returns a row vector of the prime numbers less than or equal to n. A prime number is one that has no factors other than 1 and itself.

p = primes(37)

p = 2 3 5 7 11 13 17 19 23 29 31 37

Symbolic Algebra

factor

Prime factors

f = factor(n)

f = factor(n) returns a row vector containing the prime factors of n.

```
f = factor(123)
f =
   3   41
```

Symbolic Algebra

syms

Shortcut for constructing symbolic objects

syms arg1 arg2 ... is a shortcut for

```
arg1 = sym('arg1');
arg2 = sym('arg2');
```

Example

```
syms x y;
f = x^2 - y^2;
fff = factor(f)

fff =

(x - y)*(x + y)
```

Symbolic Algebra

Examples:

```
exp(sym(pi))

ans =

exp(pi)
```

suppose you want to study the quadratic function f

```
syms a b c x

 f = a*x^2 + b*x + c

f =

a*x^2 + b*x + c
```

Symbolic Algebra

Examples:

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

A =

[a, b, c]
[c, a, b]
[b, c, a]

```
sum(A(1,:))
```

return

ans = a + b + c

Symbolic Algebra

Functions:

Suppose we created symbolic function f

```
syms  x y
f = x^3*y^3
```

After creating a symbolic function, you can differentiate, integrate, or simplify it, substitute its arguments with values, and perform other mathematical operations.

Symbolic Algebra

Functions:

To differentiate a symbolic expression, use the **diff** command

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

```
syms  x y
f = x^3*y^3
diff(f)
 ans =

3*x^2*y^3
```

Symbolic Algebra

Functions:

For multivariable expressions, you can specify the differentiation variable. If you do not specify any variable, MATLAB® chooses a default variable

(**X** in previous example)

To differentiate the symbolic expression **f** with respect to a variable **y**, enter:

diff(f,y)

ans =

3*x^3*y^2

Symbolic Algebra

Functions:

Second Partial and Mixed Derivatives

To take a second derivative of the symbolic expression f with respect to a variable y, enter:

```
diff(f,y,2)
 ans =
 6*x^3*y
```

```
syms x y
f = sin(x)^2 + cos(y)^2;
 diff(f, y, 2)

ans =
2*sin(y)^2 - 2*cos(y)^2
```

Symbolic Algebra

integrate Symbolic Expressions

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

To find the indefinite integral

```
int(f)
ans =
 x/2 - sin(2*x)/4
```

```
syms x y n
f = x^n + y^n;
 int(f, y)
ans =
x^n*y + (y*y^n)/(n + 1)
int(f, n)
ans =
 x^n/log(x) + y^n/log(y)
```

Symbolic Algebra

- **Solve Equations**
- You can solve different types of symbolic equations including:
- Algebraic equations with one symbolic variable
- Algebraic equations with several symbolic variables
- Systems of algebraic equations

Symbolic Algebra

Solve Equations

- Define an equation.
- Then you can solve the equation by calling the solve function

- syms a b c x
- >> eqn = a*x^2 + b*x + c ;
- >> sol = solve(eqn)
-
- sol =
-
- -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
- -(b - (b^2 - 4*a*c)^(1/2))/(2*a)

Symbolic Algebra

simplify

Simplify representation of an expression

```
syms x a b c
simplify(sin(x)^2 + cos(x)^2)
simplify(exp(c*log(sqrt(a+b))))

ans =

1

 ans =

(a + b)^(c/2)
```

Symbolic Algebra

simplify

```
syms x
M = [(x^2 + 5*x + 6)/(x + 2), sin(x)*sin(2*x) + cos(x)*cos(2*x);
                    (exp(-x*i)*i)/2 - (exp(x*i)*i)/2, sqrt(16)];
simplify(M)
 ans =
 [  x + 3, cos(x)]
[ sin(x),      4]
```

```
syms x
f = ((exp(-x*i)*i)/2 - (exp(x*i)*i)/2)/(exp(-x*i)/2 + ...
                                   exp(x*i)/2);
simplify(f)
 ans =
 tan(x)
```

Symbolic Algebra

EXPAND

Symbolic expansion.

EXPAND(S) writes each element of a symbolic expression S as a product of its factors. EXPAND is most often used on polynomials, but also expands trigonometric, exponential and logarithmic functions.

Examples:

```
expand((x+1)^3)   returns  x^3+3*x^2+3*x+1
expand(sin(x+y))  returns  sin(x)*cos(y)+cos(x)*sin(y)
expand(exp(x+y))  returns  exp(x)*exp(y)
```

```
syms x
f = (x ^2- 1)*(x^4 + x^3 + x^2 + x + 1)*(x^4 - x^3 + x^2 - x + 1);
expand(f)
ans = x^10 – 1
```

The same result as `simplify(f)`

Symbolic Algebra

```
syms x
g = x^3 + 6*x^2 + 11*x + 6;
factor(g)
ans =
 (x + 3)*(x + 2)*(x + 1)

>> simplify(g)
 ans =
 (x + 1)*(x + 2)*(x + 3)

horner(g)
 ans =
 x*(x*(x + 6) + 11) + 6
```

The nested (Horner) representation of a polynomial is the most efficient for numerical evaluations

Symbolic Algebra

Substitutions in Symbolic Expressions

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

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

ans =

   0.2222
```

Symbolic Algebra

Substitutions in Symbolic 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);
subs(f, x, 3)

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

Symbolic Algebra

Substitutions in Symbolic Expressions

- **Substitute One Symbolic Variable for Another**
- You also can substitute one symbolic variable for another symbolic variable. For example to replace the variable y with the variable x

```
syms x y
f = x^2*y + 5*x*sqrt(y);
subs(f, y, x)
ans = x^3 + 5*x^(3/2)
```

Symbolic Algebra

To find symbolic variables in an expression, function, or matrix, use **symvar**

```
syms a b n t x ;
g = sin(a*t + b);
symvar(g)
ans = [ a, b, t]
```

- **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**

```
syms s t          f = s + t;
                  symvar(f, 1)
                  ans = t
```

Symbolic Algebra

plotting symbolic functions

ezplot

2-D

```
syms x;
f = x^3 - 6*x^2 + 11*x - 6;
ezplot(f)
```

Symbolic Algebra

plotting symbolic functions

Ezplot 2-D

```
 syms t
x = t*sin(5*t);
y = t*cos(5*t);
>> x
x =
t*sin(5*t)
>> y
y =
t*cos(5*t)
>> ezplot(x, y)
```

Symbolic Algebra

plotting symbolic functions

EZSURF

Easy to use 3-D colored surface plotter

EZSURF(FUN) plots a graph of the function FUN(X,Y) using SURF.

FUN is plotted over the default domain -2*PI < X < 2*PI, -2*PI < Y < 2*PI.

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

Symbolic Algebra

plotting symbolic functions

EZSURF

```
 f=real(5*tan(x+i*y));
>> ezsurf(f)
 f=real(5*atan(x+i*y));
>> ezsurf(f)
        syms s t
        r = 2 + sin(7*s + 5*t);
        x = r*cos(s)*sin(t);
        y = r*sin(s)*sin(t);
        z = r*cos(t);
>> ezsurf(x,y,z)
```

Symbolic Algebra

Taylor expansion

Any smooth function f(x) can be approximated by a polynomial. The higher degree of polynomial is, the more accurate approximation is.

Example: Taylor expansion up to 3d order:

$f(x) = f(x_0) + f'(x_0)(x-x_0)+(1/2!)f''(x_0)(x-x_0)^2$

$+(1/3!)f'''(x_0)(x-x_0)^3+O((x-x_0)^4)$

Where $O((x-x_0)^4)$ is of order $(x-x_0)^4$

Symbolic Algebra

Taylor expansion

Example: $f(x) = \exp(x)$ at $x_0 = 0$

$\exp(x) = 1+x+(1/2!)\,x^2+(1/3!)x^3+O(x^4)$

Example: $f(x) = \log(1+x)$ at $x_0 = 0$

$\log(1+x) = x-(1/2!)\,x^2+2(1/3!)\,x^3+O(x^4)$

Example: $f(x) = (1+x)^a$ at $x_0 = 0$

$(1+x)^a = 1+ax+a(a-1)(1/2!)\,x^2$

$+a(a-1)(a-2)(1/3!)\,x^3+O(x^4)$

Symbolic Algebra

Why do we need it: often equations are very complex, and their approximations are simple. You need to decide around which point x_0 to approximate.

```
syms x y x0 y0;
f =y*exp(x-x0)-x*log(y);
ft=taylor(f,[x,y],[x0,y0],'order',3)
```

Symbolic Algebra

In Economics we often log-linearise: instead of polynomials of $x - x_0$, we use polynomials of $\log(x/x_0)$

Symbolic Algebra: Log-Linearisation up to Second Order

- Functuon $f(x, y)$ should be log-linearised up to second order around point (x_0, y_0)
- Taylor expansion gives us

$$
\begin{aligned}
f(x, y) \quad = \quad & f(x_0, y_0) + f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0) \\
& + \frac{1}{2} f_{xx}(x_0, y_0)(x - x_0)^2 + \frac{1}{2} f_{yy}(x_0, y_0)(y - y_0)^2 \\
& + f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + O(3)
\end{aligned}
$$

Symbolic Algebra: Log-Linearisation up to Second Order

- We want

$$
\begin{aligned}
f(x,y) \quad &= \quad f(x_0,y_0) + C_x \ln\frac{x}{x_0} + C_y \ln\frac{y}{y_0} \\
&\quad + C_{xx}\left(\ln\frac{x}{x_0}\right)^2 + C_{yy}\left(\ln\frac{y}{y_0}\right)^2 \\
&\quad + C_{xy}\left(\ln\frac{x}{x_0}\right)\left(\ln\frac{y}{y_0}\right) + O(3)
\end{aligned}
$$

  this representation is frequently used in macroeconomic problems because terms can be interpreted as 'percentage deviations from the steady state', recall that $\ln\left(\frac{x}{x_0}\right) = \ln\left(\frac{x-x_0}{x_0}+1\right) \approx \frac{x-x_0}{x_0}$ is the first-order Taylor approximation.

Symbolic Algebra: Log-Linearisation up to Second Order

- Introduce new variables

$$
\begin{aligned}
u &= \ln \frac{x}{x_0} \\
v &= \ln \frac{y}{y_0}
\end{aligned}
$$

- Then

$$
f(x, y) = f(x_0 e^u, y_0 e^v) = g(u, v)
$$

- We do standard Taylor expansion up to second order around $(u_0, v_0) = (0, 0)$

Symbolic Algebra: Log-Linearisation up to Second Order

```
syms x y x0 y0 u v;
f = y*exp(x - x0) - x*log(y);
ft = taylor(f, [x, y], [x0, y0], 'Order', 3)
g = subs(f,[x, y],[x0*exp(u),y0*exp(v)]);
gt = taylor(g, [u, v], [0, 0], 'Order', 3)
```

Symbolic Algebra: Log-Linearisation up to Second Order

- What if `taylor` is unavailable?
- Taylor expansion

```
syms x y x0 y0 u v;
f = y*exp(x - x0) - x*log(y);
df = jacobian(f,[x, y]);
d2f =jacobian(df, [x, y]);
ft =subs(f,[x, y],[x0, y0]) ...
+ subs(df,[x, y],[x0, y0])*[x-x0;y-y0] ...
+ 1/2*[x-x0,y-y0]*subs(d2f,[x, y],...
[x0, y0])*[x-x0;y-y0];
```