MATLAB Lecture 7 – Calculus

微积分

Ref: Symbolic Math Toolbox→Using the Symbolic Math Toolbox →Calculus

- **Vocabulary:**

calculus 微积分	function 函数
composite/compound function 复合函数	inverse function 反函数
limit 极限	derivative 导数
differentiation 微分	differential quotient 微商，导数
indefinite integral 不定积分	definite integral 定积分
Taylor series 泰勒级数	Taylor expansion 泰勒展开式
Taylor formula 泰勒公式	item/term 项
summation 求和	accumulate 累加
singleton 单元素	dimension 维数
index/subscript/suffix 下标	the N-th order difference N 阶微分

- **Some functions**

compose finverse limit diff int symsum taylor *gradient *sum *prod

- **Calculus**

This section explains how to use the Symbolic Math Toolbox to perform many common mathematical operations.

✧ **Function composition**

compose (f,g) returns f(g(y)) where f = f(x) and g = g(y). Here x is the symbolic variable of f as defined by findsym and y is the symbolic variable of g as defined by findsym.

compose (f,g,z) returns f(g(z)) where f = f(x), g = g(y), and x and y are the symbolic variables of f and g as defined by findsym.

compose (f,g,x,z) returns f(g(z)) and makes x the independent variable for f. That is, if f = cos(x/t), then compose (f,g,x,z) returns cos(g(z)/t) whereas compose (f,g,t,z) returns cos(x/g(z)).

compose (f,g,x,y,z) returns f(g(z)) and makes x the independent variable for f and y the independent variable for g. For f = cos(x/t) and g = sin(y/u), compose (f,g,x,y,z) returns cos(sin(z/u)/t) whereas compose (f,g,x,u,z) returns cos(sin(y/z)/t).

Examples:

```
>> syms x y z t u;
>> f = 1/(1 + x^2); g = sin(y); h = x^t; p = exp(-y/u);
```

```
>> compose(f,g);
```
% $\frac{1}{1+\sin^2 y}$

```
>> compose(f,g,t);
```
% $\frac{1}{1+\sin^2 t}$

```
>> compose(h,g,x,z);        % sin^t z

>> compose(h,g,t,z);        % x^(sin z)

>> compose(h,p,x,y,z);      % e^((-z/u)^t)

>> compose(h,p,t,u,z);      % x^(e^(-y/z))
```

NOTICE: the order of the codes is important.

```
>> clear; syms x y z; z=exp(y); y=sin(x); compose(z, y), z
ans =
exp(sin(x))
z =
exp(y)
>> clear; syms x y z; y=sin(x); z=exp(y); compose(z, y), z      %NOT RECOMMEND
ans =
exp(sin(sin(x)))
z =
exp(sin(x))
>> clear; syms u v y z; y=sin(u); z=exp(v); compose(z, y)      %RECOMMEND to use…
                    different variables as independent variable for different functions
```

✧ Functional inverse

g = finverse(f) returns the functional inverse of f. f is a scalar sym representing a function of exactly one symbolic variable, say 'x'. Then g is a scalar sym that satisfies g(f(x)) = x.

g = finverse(f,v) uses the symbolic variable v, where v is a sym, as the independent variable. Then g is a scalar sym that satisfies g(f(v)) = v. Use this form when f contains more than one symbolic variable.

Examples:

```
>> finverse(1/tan(x));       %returns atan(1/x).
>> f = x^2+y;
>> finverse(f,y);        %returns -x^2+y.
>> finverse(f)
Warning: finverse(x^2+y) is not unique.
> In sym.finverse at 43
ans =
  (-y+x)^(1/2)
```

✧ Limit of an expression

limit(F,x,a) takes the limit of the symbolic expression F as x → a.

limit(F,a) uses findsym(F) as the independent variable.

limit(F) uses a = 0 as the limit point.

limit(F,x,a,'right') or limit(F,x,a,'left') specify the direction of a one-sided limit.

Examples:

```
>> syms x a t h;
>> limit(sin(x)/x)        %evaluate  lim_{x→0} sin x / x
ans =
1
>> limit((x-2)/(x^2-4),2)        %evaluate  lim_{x→2} (x-2)/(x^2-4)
ans =
1/4
>> limit((1+2*t/x)^(3*x),x,inf)        %evaluate  lim_{x→∞} (1+2t/x)^(3x)
ans =
exp(6*t)
>> limit(1/x,x,0,'right')        %evaluate  lim_{x→0+} 1/x
ans =
Inf
>> limit(1/x,x,0,'left')        %evaluate  lim_{x→0-} 1/x
ans =
-Inf
>> limit((sin(x+h)-sin(x))/h,h,0)        %evaluate  lim_{h→0} (sin(x+h)-sin x)/h
ans =
cos(x)
>> v = [(1 + a/x)^x, exp(-x)];
>> limit(v,x,inf,'left')        %evaluate  lim_{x→∞} (1+a/x)^x ,  lim_{x→∞} e^{-x}
ans =
  [ exp(a),          0]
```

✧ Difference and approximate derivative

diff (X), for a vector X, is [X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)].

diff (X), for a matrix X, is the matrix of row differences, [X(2:n,:) - X(1:n-1,:)].

diff (X,N) is the N-th order difference along the first non-singleton dimension (denote it by dim). If N >= size(X,dim), diff takes successive differences along the next non-singleton dimension.

diff (X,N,DIM) is the Nth difference function along dimension DIM. If N >= size(X,DIM), diff returns an empty array.

Examples:

```
>> h = .001; x = 0:h:pi;
```

>> diff(sin(x.^2))/h; % $\frac{\sin(i)^2-\sin(i-h)^2}{h}, i=h:h:pi$, there are approximation …

to 2*cos(x.^2).*x, x=0:h:pi

>> diff((1:10).^2) % evaluate $(i+1)^2-i^2, i=1,2,...,9$

```
ans =
     3     5     7     9    11    13    15    17    19
>>X = [3 7 5; 2 5 7]; diff(X)      %the same as diff(X,1,1)
ans =
    -1    -2     2
>> diff(X,1,2)
ans =
     4    -2
     3     2
>> diff(X,2,2)      %the 2nd order difference along the dimension 2
ans =
    -6
    -1
>> diff(X,3,2)
ans =
  Empty matrix: 2-by-0
>> syms x y; y=atan((x+1)/(x-1));
>> yx=diff(y,x)
yx =
 (1/(x-1)-(x+1)/(x-1)^2)/(1+(x+1)^2/(x-1)^2)
>> yxx=diff(y,x,2)
yxx =
(-2/(x-1)^2+2*(x+1)/(x-1)^3)/(1+(x+1)^2/(x-1)^2)-(1/(x-1)-(x+1)/(x-1)^2)/
(1+(x+1)^2/(x-1)^2)^2*(2*(x+1)/(x-1)^2-2*(x+1)^2/(x-1)^3)
>> syms x y
>> z=x^4+y^4-cos(2*x+3*y);
```

>> zx=diff(z,x) %compute $\frac{\partial(x^4+y^4-\cos(2x+3y))}{\partial x}$

```
zx =
4*x^3+2*sin(2*x+3*y)
```

>> zy=diff(z,y) %compute $\frac{\partial z}{\partial y}$

```
zy =
4*y^3+3*sin(2*x+3*y)
```

```
>> zxx=diff(zx,x)      %compute ∂²z/∂x², equivalent to zxx=diff(z,x,2)

zxx =
12*x^2+4*cos(2*x+3*y)
```

✧ Integrate

int(S) is the indefinite integral of S with respect to its symbolic variable as defined by findsym. S is a sym (matrix or scalar). If S is a constant, the integral is with respect to 'x'.

int(S,v) is the indefinite integral of S with respect to v. v is a scalar sym.

int(S,a,b) is the definite integral of S with respect to its symbolic variable from a to b. a and b are each double or symbolic scalars.

int(S,v,a,b) is the definite integral of S with respect to v from a to b.

Examples:

```
>> syms x x1 alpha u t;
>> A = [cos(x*t), sin(x*t);-sin(x*t), cos(x*t)];
>> int(1/(1+x^2))      %compute ∫ 1/(1+x²) dx  without constant item C

ans =
atan(x)

>> int(sin(alpha*u),alpha)    %compute ∫ sin(αu) dα  without constant item C

ans =
-1/u*cos(alpha*u)

>> int(x1*log(1+x1),0,1)      %compute ∫₀¹ x₁ ln(1+x₁) dx₁

ans =
1/4
>> int(4*x*t,x,2,sin(t))
ans =
2*t*(sin(t)^2-4)

>> int([exp(t),exp(alpha*t)])      %compute [∫ eᵗ dt,  ∫ e^(αt) dt]

ans =
  [                  exp(t), 1/alpha*exp(alpha*t)]

>> int(A,t)      %compute [∫ cos xt dt,  ∫ sin xt dt;  ∫ −sin xt dt,  ∫ cos xt dt]

ans =
[ 1/x*sin(x*t),   -cos(x*t)/x]
[     cos(x*t)/x, 1/x*sin(x*t)]
```

✧ Taylor series expansion

taylor(f) is the fifth order Maclaurin polynomial approximation to f.

Three additional parameters can be specified, in almost any order.

taylor(f,n) is the (n-1)-st order Maclaurin polynomial.

taylor (f,a) is the Taylor polynomial approximation about point a.

taylor (f,x) uses the independent variable x instead of findsym(f).

Examples:

```
>> taylor(exp(-x))      %evaluate the first 6 items of Taylor series expansion at 0
ans =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
>> taylor(log(x),6,1)      %evaluate the first 6 items of Taylor series expansion at 1
ans =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
>> taylor(sin(x),6,pi/2)      %evaluate the first 6 items of Taylor series expansion at pi/2
ans =
1-1/2*(x-1/2*pi)^2+1/24*(x-1/2*pi)^4
>> taylor(sin(x)*t,t)       %evaluate the first 6 items of Taylor series expansion responding …
                            to t
ans =
sin(x)*t
>> taylor(sin(x)*t,x)    %evaluate the first 6 items of Taylor series expansion responding …
                             to x

ans =
x*t-1/6*t*x^3+1/120*t*x^5
```

✧ Sum of elements

S = sum(X) is the sum of the elements of the vector X.

If X is a matrix, S is a row vector with the sum over each column.

If X is floating point, that is double or single, S is accumulated natively, that is in the same class as X, and S has the same class as X.

If X is not floating point, S is accumulated in double and S has class double.

S = sum(X,DIM) sums along the dimension DIM.

Examples:

```
>> X = [0 1 2; 3 4 5];
>> sum(X)      %evaluate a row vector with the sum over each column
ans =
     3     5     7
>> sum(X,1)      %evaluate a row vector with the sum over each column
ans =
     3     5     7
>> sum(X,2)      %evaluate a column vector with the sum over each row
ans =
     3
```

```
12
```

✧ Symbolic summation

symsum(S) is the indefinite summation of S with respect to the symbolic variable determined by findsym.

symsum(S,v) is the indefinite summation with respect to v.

symsum(S,a,b) and symsum(S,v,a,b) are the definite summation from a to b.

Examples:

```
>> syms k n
>> a1=simple(symsum(k))      %evaluate ∑_{i=0}^{k-1} i
a1 =
1/2*k*(k-1)
>> a2=simple(symsum(k,0,n-1))      %evaluate ∑_{k=0}^{n-1} k
a2 =
1/2*n*(n-1)
>> a3=simple(symsum(k,0,n))      %evaluate ∑_{k=0}^{n} k
a3 =
1/2*n*(n+1)
>> a4=simple(symsum(k^2,0,n))      %evaluate ∑_{k=0}^{n} k^2
a4 =
1/6*n*(n+1)*(2*n+1)
>> symsum(k^2,0,10)      %evaluate ∑_{k=0}^{10} k^2
ans =
385
>> symsum(k^2,11,10)      %evaluate ∑_{k=11}^{10} k^2
ans =
0
>> symsum(1/k^2)      %evaluate ∑_{i=1}^{k} 1/i^2
ans =
-Psi(1,k)
>> symsum(1/k^2,1,Inf)      %evaluate ∑_{k=1}^{∞} 1/k^2
ans =
1/6*pi^2
```