Saghar Gorjiduz | 95243096
Tara Barghian | 97243009
Mohammad Hashemi | 97243073
Instructor: Dr. Yaser Shekofte
June 5, 2020

Matlab #6 Report

Matlab Programming Workshop | Spring 99

6-1 : assume - assumptions

Name	Syntax	Description
assume	`assume(condition)`	States that condition is valid. assume is not additive. Instead, it automatically deletes all previous assumptions on the variables in condition.
	`assume(expr,set)`	States that expr belongs to set. assume deletes previous assumptions on variables in expr.
	`assume(expr,'clear')`	Clears all assumptions on all variables in expr.

Name	Syntax	Description
assumptions	`assumptions(var)`	Returns all assumptions that affect variable var. If var is an expression or function, assumptions returns all assumptions that affect all variables in var.
	`assumptions`	Returns all assumptions that affect all variables in MATLAB® Workspace.

```
1  %report6 problem1
2  clc;clear;close all
3
4  syms x;
5  %assume(x<5 | x>-5 )
6  assumeAlso( x/2 , 'integer')
7  assumeAlso(x~=0)
8  assumptions
9  solve(x<5,x>-5,x) %print even numbers in (-5,5) but 0
```

Command Window

```
ans =

[ in(x/2, 'integer'), x ~= 0]


ans =

 -4
 -2
  2
  4
```

6-2 : double

Name	Syntax	Description
double	Y = double(X)	If you have an array of a different type, such as single or int8, then you can convert that array to double precision using the double function.

```
%report6 problem2

clc;clear; close all
x = sym ([1,2,3 ; 4,5,6 ; 7,8,9]);
y=double(x);

whos x
whos y
```

```
Command Window
  Name        Size              Bytes  Class     Attributes

  x           3x3                   8  sym

  Name        Size              Bytes  Class     Attributes

  y           3x3                  72  double

fx >>
```

6-3 : symvar

Name	Syntax	Description
symvar	C = symvar(expr)	Searches the expression, `expr`, for identifiers other than i, j, pi, inf, nan, eps, and common functions. These identifiers are the names of variables in the expression. `symvar` returns the identifiers in a cell array of character vectors, C. If `symvar` finds no identifiers, then C is an empty cell array.

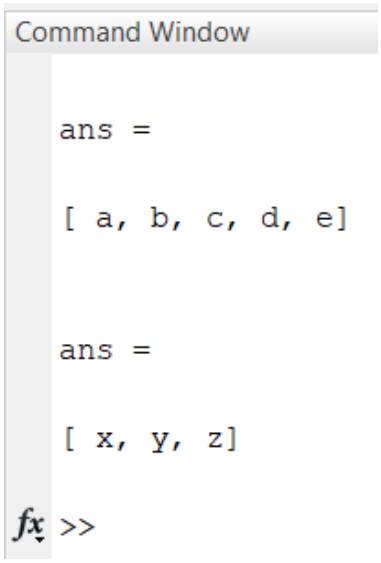
```
%report6 problem3
clc;clear; close all

%Find all symbolic variables in an expression. symvar returns the variables in alphabetical order.

syms  a b c d e
avr = (a+b+c+d+e)/5;
symvar(avr)

%Find the first three symbolic variables in an expression.
%symvar chooses variables that are alphabetically closest to x and returns them in alphabetical order.

syms a b x y z
f = cos(a)*exp(x^2/(sin(3*y-b)))+z;
symvar(f,3)
```



6-4 : diff

Name	Syntax	Description
diff	`Y = diff(X)`	Calculates differences between adjacent elements of X along the first array dimension whose size does not equal 1. If X is a vector of length m, then `Y = diff(X)` returns a vector of length m-1. If `X` is a 0-by-0 empty matrix, then `Y = diff(X)` returns a 0-by-0 empty matrix.
	`Y = diff(X, n)`	Calculates the nth difference by applying the `diff(X)` operator recursively n times. In practice, this means `diff(X,2)` is the same as `diff(diff(X))`.
	`Y = diff(X, n, dim)`	Is the nth difference calculated along the dimension specified by dim. The dim input is a positive integer scalar.

We Use the **`diff`** function to approximate partial derivatives with the syntax `Y = diff(f)/h`, where `f` is a vector of function values evaluated over some domain, `X`, and `h` is an appropriate step size.

```
Y = ((diff(f)/h)/h)/h    %third derivative
```

```
5 -  X = [2 4 9 10 11 32];
6 -  Y1 = diff(X)
7 -  Y2 = diff(X, 2)
8
9    %use diff to approximate the derivatives of functions
10 - h = 0.001;   %step size
11 - X = 0:h:5;
12 - Y = X .^ 3 + 2 * X .^ 2;
13 - deriY_1 = diff(Y) / h;    % deriY_1 = 3x^2 + 2x
14 - deriY_2 = diff(deriY_1) / h;  %deriY_2 = 6x + 2
15 - plot(X(:,1:length(deriY_1)),deriY_1,'r', ...
16       X,Y,'b', ...
17       X(:,1:length(deriY_2)),deriY_2,'k');
18 - axis([0, 5, 0, 100]);
19 - legend({"Y' = 3x^2 + 2x", 'Y = x^3 + 2x^2', "Y''= 6x + 2"})
```

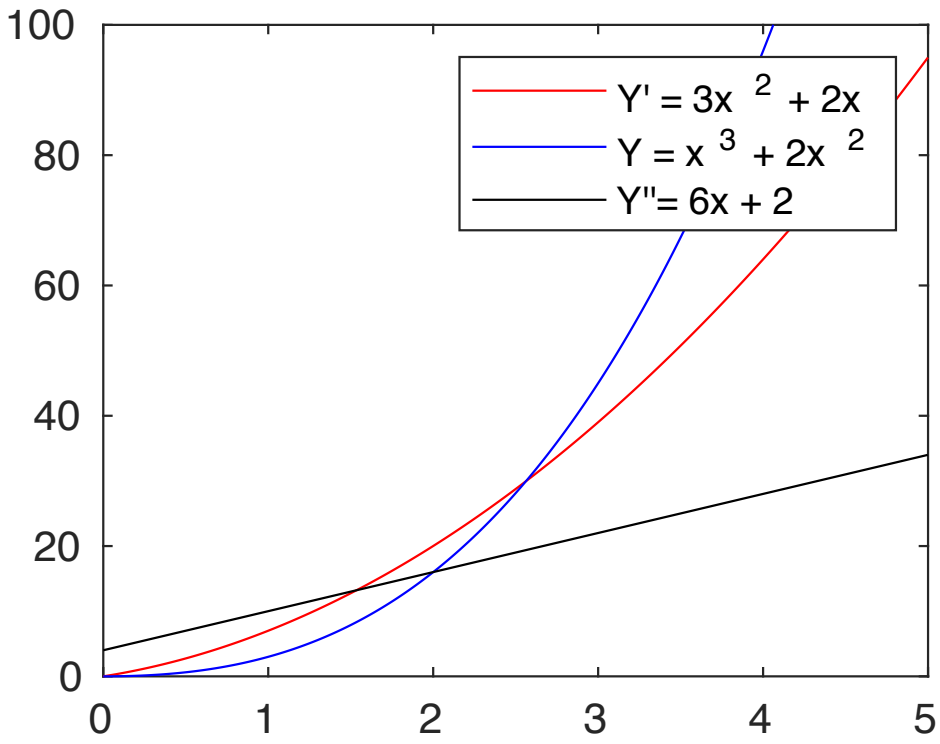
COMMAND WINDOW

```
Y1 =

     2     5     1     1    21


Y2 =

     3    -4     0    20
```



6-5 : Partial Derivatives

$$z(x,y) = xy + x^2 \quad , \quad w(x,y) = e^{2xy} + \cos(\frac{x}{3}) - \sin^2(4y)$$

```
%report6 problem5

clc;clear;close all
syms x y

z = x*y + x^2;
deri1_z_x = diff(z, x)  % = y + 2x
deri2_z_x = diff(diff(z,x), x)  % = 2
deri1_z_y = diff(z, y)  % = x
deri2_z_y = diff(diff(z,y), y)  % = 0

w = exp(2*x*y) + cos(x/3) + sin(4*y)^2;
deri1_w_x = diff(w, x)  % = 2ye^(2xy) - (1/3)sin(x/3)
deri2_w_x = diff(diff(w,x), x)  % = 4(y^2)*e^(2xy) - (1/9)cos(x/3)
deri1_w_y = diff(w, y)  % = 2xe^(2xy) - 8sin(4y)cos(4y)
deri2_w_y = diff(diff(w,y), y)  % = 4(x^2)*e^(2xy) - ...
```

Command Window

```
deri1_z_x =

2*x + y


deri2_z_x =

2


deri1_z_y =

x


deri2_z_y =

0


deri1_w_x =

2*y*exp(2*x*y) - sin(x/3)/3


deri2_w_x =

4*y^2*exp(2*x*y) - cos(x/3)/9


deri1_w_y =

2*x*exp(2*x*y) + 8*cos(4*y)*sin(4*y)


deri2_w_y =

32*cos(4*y)^2 - 32*sin(4*y)^2 + 4*x^2*exp(2*x*y)
```

6-6 : int

$$I_1 = \int_0^4 e^{-3x}dx = ? \quad , \quad I_2 = \int_0^{+\infty} e^{-x}dx = ? \quad , \quad I_3 = \int_{-\infty}^{+\infty} e^{-x^2}dx = ?$$

Name	Syntax	Description
int	`F = int(expr)`	Computes the indefinite integral of `expr`. `int` uses the default integration variable determined by `symvar(expr, 1)`. If `expr` is a constant, then the default integration variable is `x`.
	`F = int(expr, var)`	computes the indefinite integral of `expr` with respect to the symbolic scalar variable `var`.
	`F = int(expr, var, a, b)`	computes the definite integral of `expr` with respect to the symbolic scalar variable `var` from `a` to `b`.

```
1        %report6 problem6
2
3 -      clc;clear;close all
4 -      syms x
5
6 -      expr1 = e^(-3*x);
7 -      I1 = int(expr1, 0, 4)
8
9 -      expr2 = e^(-x);
10 -     I2 = int(expr1, 0, inf)
11
12 -     expr3 = e^(-(x^2));
13 -     I3 = int(expr1, -inf, inf)
```

```
Command Window

I1 =

1/3 - exp(-12)/3


I2 =

1/3


I3 =

Inf

fx >>
```

6-7 : limit

Name	Syntax	Description
limit	`limit(f,var,a)`	Returns the Bidirectional Limit of the symbolic expression f when var approaches a.
	`limit(f,a)`	Uses the default variable found by symvar.
	`limit(f)`	Returns the limit at 0.
	`limit(f,var,a,'left')`	Returns the Left Side Limit of f as var approaches a.
	`limit(f,var,a,'right')`	Returns the Right Side Limit of f as var approaches a.

```
1       %report6 problem7
2 -     clc;clear;
3
4 -     syms x
5       %first we make our function the find the limit of it by two given number
6       %2+,2-,3+,3-
7 -     f=((x^2-4)^2)/((x-2)*(x-3));
8 -     limit(f,x,2,'right')
9 -     limit(f,x,2,'left')
10 -    limit(f,x,3,'right')
11 -    limit(f,x,3,'left')
```

```
ans =

0


ans =

0


ans =

Inf


ans =

-Inf

fx >>
```

6-8 : finverse

Name	Syntax	Description
finverse	`g = finverse(f)`	Returns the inverse of function f, such that f(g(x)) = x. If f contains more than one variable, use the next syntax to specify the independent variable.
	`g = finverse(f,var)`	Uses the symbolic variable var as the independent variable, such that f(g(var)) = var.

```
%report6 problem8
clc;clear;
syms  x  y
finverse (exp(x-5*y),x)
%%%
syms  x  y
finverse (log( x-y ),y)
%%%
sym x ;
y(x)=cos(x)
z = finverse(y)
```

```
ans =

5*y + log(x)


ans =

x - exp(y)


y(x) =

cos(x)


z(x) =

acos(x)

fx >>
```