NUMERIC INTEGRATION METHODS

Function Handles: A function handle is a data type that stores an association to a function. For example, you can use a function handle to construct anonymous functions or specify call back functions. Also, you can use a function handle to pass a function to another function, or call local functions from outside the main function.

What Is a Function Handle?

1) A function handle is a MATLAB® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from. Typical uses of function handles include:
2) Pass a function to another function (often called *function functions*). For example, passing a function to integration and optimization functions, such as integral and fzero.
3) Specify callback functions. For example, a callback that responds to a UI event or interacts with data acquisition hardware.
4) Construct handles to functions defined inline instead of stored in a program file (anonymous functions).
5) Call local functions from outside the main function.

Creating Function Handles: To create a handle for a function, precede the function name with an @ sign. For example, if you have a function called myfunction, create a handle named f as follows:

f = @myfunction;

Anonymous Functions: You can create handles to anonymous functions. An anonymous function is a one-line expression-based MATLAB function that does not require a program file. Construct a handle to an anonymous function by defining the body of the function, anonymous_function, and a comma-separated list of input arguments to the anonymous function, arglist. The syntax is:

>> h = @(*arglist*)*anonymous_function*

For example, create a handle, sqr, to an anonymous function that computes the square of a number, and call the anonymous function using its handle.

```
>> sqr = @(n) n.^2;
>> x = sqr(3) % x =    9
```

Integral : Numerical integration

Syntax:

q = integral(fun,xmin,xmax)

q = integral(fun,xmin,xmax,Name,Value)

Description:

- q = integral(fun,xmin,xmax) numerically integrates function fun from xmin to xmax using global adaptive quadrature and default error tolerances.

- q = integral(fun,xmin,xmax,Name,Value) specifies additional options with one or more Name,Value pair arguments. For example, specify 'WayPoints' followed by a vector of real or complex numbers to indicate specific points for the integrator to use.

Here,

- fun — represents function handle which defines the function to be integrated from xmin to xmax.
- xmin — Lower limit of *x*
- xmax — Upper limit of *x*

Examples:

1) **Improper Integral**

   Create the function $f(x) = e^{-x^2}(\ln x)^2$. Evaluate the integral from x=0 to x=Inf.

   ```
   >> fun = @(x) exp(-x.^2).*log(x).^2;
   >> q = integral(fun,0,Inf)
   ```

2) **Parameterized Function**

   Create the function $f(x) = 1/(x^3 - 2x - c)$ with one parameter, *c*. Evaluate the integral from x=0 to x=2 at c=5.

   ```
   >> fun = @(x,c) 1./(x.^3-2*x-c);
   >> q = integral(@(x)fun(x,5),0,2)
   ```

3) **Singularity at Lower Limit**

   Create the function $f(x) = \ln(x)$. Evaluate the integral from x=0 to x=1 with the default error tolerances.

   ```
   >> fun = @(x)log(x);
   >> format long
   >> q1 = integral(fun,0,1)
   ```

integral2 : To numerically evaluate double integral

Syntax:

q = integral2(fun,xmin,xmax,ymin,ymax)
q = integral2(fun,xmin,xmax,ymin,ymax,Name,Value)

Description:

- q = integral2(fun,xmin,xmax,ymin,ymax) approximates the integral of the function z = fun(x,y) over the planar region xmin ≤ x ≤ xmax andymin(x) ≤ y ≤ ymax(x).
- q = integral2(fun,xmin,xmax,ymin,ymax,Name,Value) specifies additional options with one or more Name,Value pair arguments.

Here,

- fun —defines the function to be integrated over the planar region xmin ≤ x ≤ xmax and ymin(x) ≤ y ≤ ymax(x). The function fun must accept two

arrays of the same size and return an array of corresponding values. It must perform element-wise operations.

- xmin — Lower limit of *x*
- xmax — Upper limit of *x*
- ymin — Lower limit of *y*
- ymax — Upper limit of *y*

Examples:

1) **Integrate Triangular Region with Singularity at the Boundary**

The function

$$f(x,y) = \frac{1}{(\sqrt{x+y})(1+x+y)}$$

is undefined when *x* and *y* are zero. integral2 performs best when singularities are on the integration boundary.

Create the anonymous function. Integrate over the triangular region bounded by $0 \le x \le 1$ and $0 \le y \le 1 - x$.

```
>> fun = @(x,y) 1./( sqrt(x + y) .* (1 + x + y).^2 )
>> ymax = @(x) 1 - x;
>> q = integral2(fun,0,1,0,ymax)
```

2) **Evaluate Double Integral in Polar Coordinates**

Define the function

$$f(\theta,r) = \frac{r}{\sqrt{r\cos\theta + r\sin\theta}\,(1 + r\cos\theta + r\sin\theta)^2}$$

Define a function for the upper limit of *r*. Integrate over the region bounded by $0 \le \theta \le \pi/2$ and $0 \le r \le r_{max}$.

```
>> fun = @(x,y) 1./( sqrt(x + y) .* (1 + x + y).^2 );
>> polarfun = @(theta,r) fun(r.*cos(theta),r.*sin(theta)).*r;
>> rmax = @(theta) 1./(sin(theta) + cos(theta));
>> q = integral2(polarfun,0,pi/2,0,rmax)
```

integral3: Numerically evaluate triple integral

Syntax:

q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax)

q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name,Value)

Description:

- q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax) approximates the integral of the function z = fun(x,y,z) over the region xmin ≤ x ≤ xmax, ymin(x) ≤ y ≤ ymax(x) and zmin(x,y) ≤ z ≤ zmax(x,y).
- q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name,Value) specifies additional options with one or more Name,Value pair arguments.

Here,

- fun — defines the function to be integrated over region xmin ≤ x ≤ xmax, ymin(x) ≤ y ≤ ymax(x), and zmin(x,y) ≤ z ≤ zmax(x,y). The functionfun must accept three arrays of the same size and return an array of corresponding values. It must perform element-wise operations.
- xmin — Lower limit of *x*
- xmax — Upper limit of *x*
- ymin — Lower limit of *y*
- ymax — Upper limit of *y*
- zmin — Lower limit of *z*
- zmax — Upper limit of *z*

Examples:

- **Triple Integral with Finite Limits**
  Define the anonymous function $f(x, y, z) = y \sin x + z \cos x$. Integrate over the region $0 \le x \le \pi$, $0 \le y \le 1$, and $-1 \le z \le 1$.

  ```
  >> fun = @(x,y,z) y.*sin(x)+z.*cos(x)
  >> q = integral3(fun,0,pi,0,1,-1,1)
  ```

- **Integral Over the Unit Sphere in Cartesian Coordinates**
  Define the anonymous function $f(x, y, z) = x \cos y + x^2 \cos z$. Define the limits of integration.

  ```
  >> fun = @(x,y,z) x.*cos(y) + x.^2.*cos(z)
  >> xmin = -1;
  >> xmax = 1;
  >> ymin = @(x)-sqrt(1 - x.^2);
  >> ymax = @(x) sqrt(1 - x.^2);
  >> zmin = @(x,y)-sqrt(1 - x.^2 - y.^2);
  >> zmax = @(x,y) sqrt(1 - x.^2 - y.^2);
  >> q = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax)
  ```

- **Evaluate Improper Triple Integral of Parameterized Function**
  Define the anonymous parameterized function $f(x, y, z) = 10/(x^2 + y^2 + z^2 + a)$. Evaluate the triple integral over the region $-\infty \le x \le 0$, $-100 \le y \le 0$, and $-100 \le z \le 0$.

  ```
  >> a = 2;
  >> f = @(x,y,z) 10./(x.^2 + y.^2 + z.^2 + a);
  >> format long
  >> q1 = integral3(f,-Inf,0,-100,0,-100,0)
  ```

Trapz: Trapezoidal numerical integration

Syntax

Q = trapz(Y)

Q = trapz(X,Y)

Q = trapz(___,dim)

Description

- Q = trapz(Y) returns the approximate integral of Y via the trapezoidal method with unit spacing. The size of Y determines the dimension to integrate along: If Y is a vector, then trapz(Y) is the approximate integral of Y. If Y is a matrix, then trapz(Y) integrates over each column and returns a row vector of integration values. If Y is a multidimensional array, then trapz(Y) integrates over the first dimension whose size does not equal 1. The size of this dimension becomes 1, and the sizes of other dimensions remain unchanged.
- Q = trapz(X,Y) integrates Y with spacing increment X. By default, trapz operates on the first dimension of Y whose size does not equal 1. length(X) must be equal to the size of this dimension . If X is a scalar, then trapz(X,Y) is equivalent to X*trapz(Y).
- Q = trapz(___,dim) integrates along the dimension dim using any of the previous syntaxes. You must specify Y, and optionally can specify X. The length of X, if specified, must be the same as size(Y,dim). For example, if Y is a matrix, then trapz(X,Y,2) integrates each row of Y.

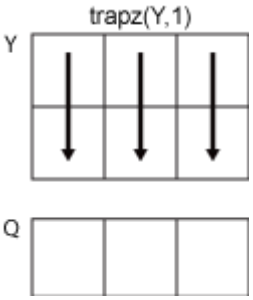
Here,

- Y — specified as a vector, matrix, or multidimensional array. By default, trapz integrates along the first dimension of Y whose size does not equal
- X — specified as 1 (default), a uniform scalar spacing, or a vector of nonuniform spacings. If X is a vector, then length(X) must be the same as the size of the integration dimension in Y.
- dim — specified as a positive integer scalar. If no value is specified, then the default is the first array dimension whose size does not equal 1.

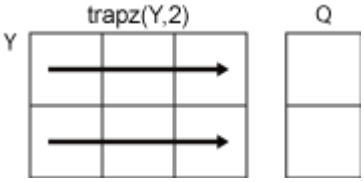
Trapezoidal Method: trapz performs numerical integration via the trapezoidal method. This method approximates the integration over an interval by breaking the area down into trapezoids with more easily computable areas.

Consider a two-dimensional input array, Y:

trapz(Y,1) works on successive elements in the columns of Y and returns a row vector of integration values.



trapz(Y,2) works on successive elements in the rows of Y and returns a column vector of integration values.



If dim is greater than ndims(Y), then trapz returns an array of zeros of the same size as Y.

Commands for Symbolic and Numeric Method for Single Integration:

Evaluate following using MATLAB:

$$\int_0^1 \frac{x^a - 1}{logx}\ dx$$

Using Symbolic:

```
>> syms x % declare symbolic variable
>> a = 3 % assign value for constant a
>> f = (x ^a  - 1) / log(x) % enter function
>> ans= int(f,x,0,1) % integrate function over x for given limits
>> eval(ans) % display answer in numeric form
```

Using Numeric:

```
>> f = @(x,a) (x .^a  - 1) ./ log(x) % x and a are declared as variables
>> integral(@(x) f(x,3),0,1) % f is integrated over x with a = 3
```

Note: Prefer numeric integration function (integral, integral2, integral3) over symbolic integration function (int).