ENGR 1100

Week 10- Numerical Analysis I

(Class lecture)

Learning Objectives

Upon completion this module, students will be able to:

1. Understand what numerical methods are and why they are important
2. Understand the different types of numerical methods
3. Be able to solve a single-variable equation in MATLAB
4. Be able to find the minimum and maximum of a function over a certain range of independent variable x.
5. Understand what numerical integration is and be able to perform numerical integration in MATLAB

Numerical Methods

- What are numerical methods?

  A *numerical method* is a technique for computing a numerical approximation of the solution to a mathematical problem. Numerical methods used when *exact* or *analytical* solutions are unavailable or impractical to use.

- Why is numerical methods important?

  Engineering problems frequently arise in which exact analytical solutions are not available (no closed form algebraic solutions). Numerical methods are the answer to this. If you study any sort of engineering science you need inevitably learn some corner of numerical analysis.

Examples of Applications

- The roots of high degree polynomials.
- The solution of simple differential equations on irregular domains.
- Maximum load a bridge can support (Civil Engineering)
- Reaction time of a chemical process (Chemical Engineering)
- Expected return of a product portfolio (Industrial and Operations Engineering)

Categories of Numerical Methods

- Linearization
- Finding Roots of Functions
- Solving Systems of Equations
- Optimization
- Numerical Integration and Differentiation

Topics under Numerical Analysis

In the next few modules, we are going to cover,

1. Solving an equation with one unknown
2. Finding a minimum or maximum value of a function
3. Numerical integration
4. Solving a first-order, ordinary, differential equation

Numerical Analysis

Go over the following resources to learn about numerical analysis topics which include solving single-variable equations, finding min and max a given function and integrating a function using numerical methods.

Newton's Method Explained

What is Numerical Integration?

Optimization: Min and Max of a Function

Numerical Analysis in MATLAB

Go over the following resources to learn about numerical analysis in MATLAB.

MATLAB: fzero() function

MATLAB: fminbnd() function

MATLAB: integral() function

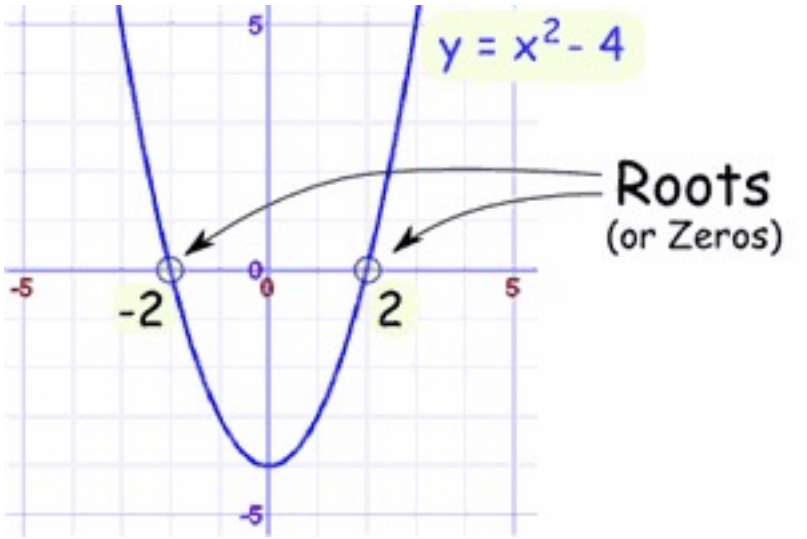
MATLAB: trapz() function

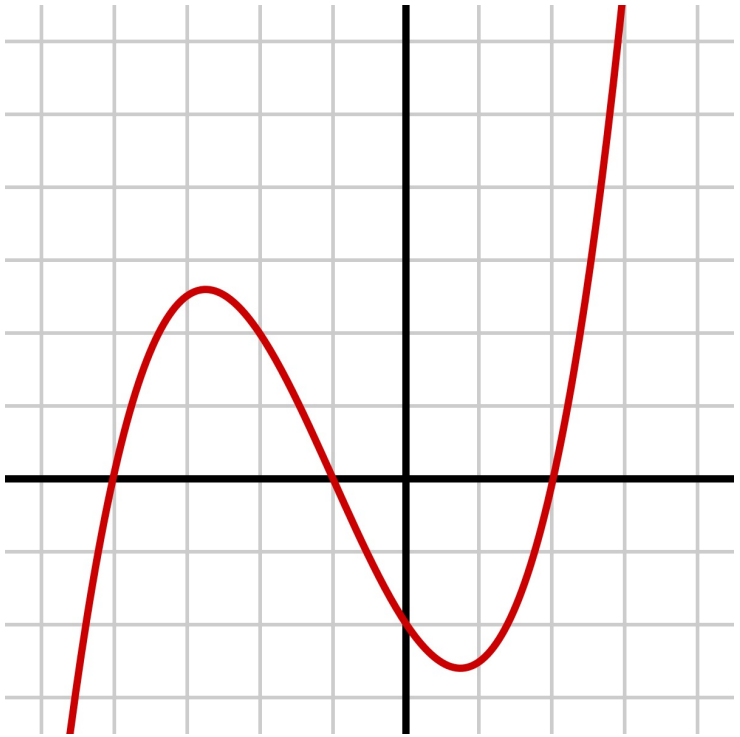
MATLAB: quadgk() function

SOLVE SINGLE-VARIABLE EQUATION

Solving an equation with one unknown

A *solution* or *root* of the equation is a value of the variable that makes the equation *f(x)=0* be true. Graphically, a solution is a point where the function *f(x)* touches or crosses the x-axis.





The equation $f(x) = 0$

- No solutions
  - Example: $f(x) = 0$ has no solutions if $f(x) = 5$
- A finite number of solutions
  - Example (one solution): $f(x) = 0$ has exactly one solution if $f(x) = x$
  - Example (two solutions): $f(x) = 0$ has exactly two solutions if $f(x) = 1 - x^2$
- An infinite number of solutions
  - Example: $f(x) = 0$ has an infinite number of solutions if $f(x) = \sin(x)$

Numerical solution

A numerical solution or root is a value of *x* that makes the function be zero or almost zero.

A general **procedure** for finding a numerical solution is,

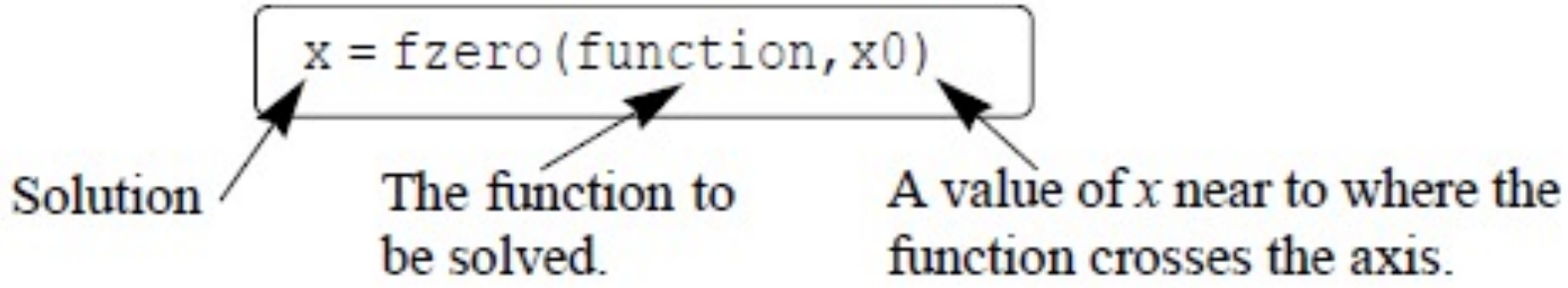
1. The user provides an initial guess of the numerical root, which the procedure uses as its current estimate of the numerical root
2. If the current estimate makes the function be almost zero, use that estimate as the numerical root and stop computing
3. If the current estimate does not make the function be almost zero, find a new estimate that makes the function be closer to zero and go to Step 2

Stopping Rules

- The procedure lets the user define what "**almost zero**" means, e.g., the absolute value of the estimate must be $< 10^{-6}$
- Another way for the procedure to stop is when the current and previous estimates are **almost the same**
  - The procedure lets the user define "almost the same", e.g., absolute value of the difference is less than 10^{-4}

MATLAB function

Use the MATLAB function **`fzero()`** to find a root of a function



- `fzero` is a **function function**, so it accepts another function (the function to be solved) as an input argument

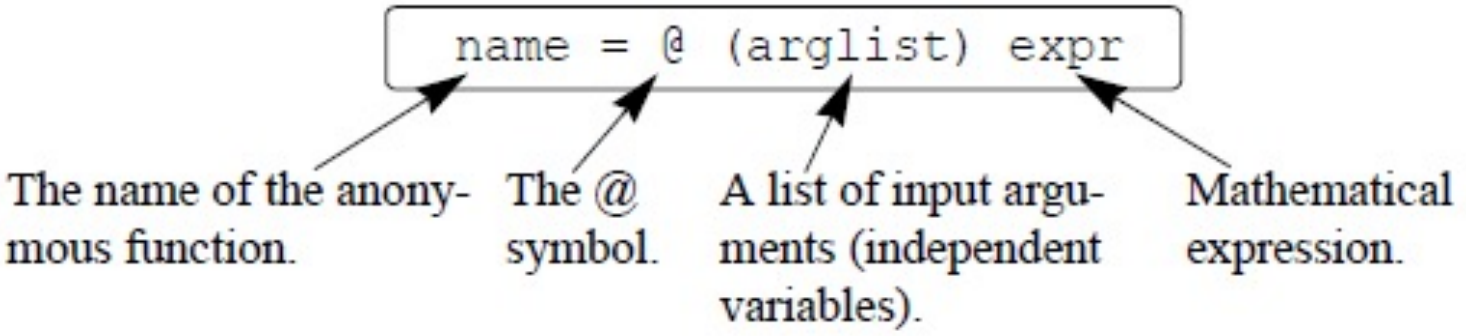
Recap: Anonymous function

An ***anonymous function*** is a function defined without using a separate function file

- For short (one line!) functions only
- Usually written by user
- Specified within another function or body of code, not in separate file
- Often used with MATLAB functions that operate on other functions, e.g.,
  - Find area under a function
  - Find derivatives of a function
  - Find optimal value of a function

Anonymous function

We can make anonymous function in Command Window, script file, or inside user-defined function. Form is:



For example

```
cube = @(x) x^3
```

Arguments of `fzero`

- `x`, the numerical solution, is a **scalar**
- `function` is the function to be solved. You can enter it several different ways:
  - Simplest way is to enter the mathematical expression as a text string
  - Make a user-defined function in a file and then pass the function handle
  - Make an anonymous function and then pass the name of the anonymous function

More on function f(x)

You must write `function` in the standard form ***f(x) = 0***. You might have to rewrite the function you're given in the problem, e.g.,

- If you're given $xe^{-x} = 0.2$, you have to rewrite the equation as $xe^{-x} - 0.2 = 0$ and then pass the function $\mathrm{f}(x) = xe^{-x} - 0.2$ to `fzero`
- If you were to pass this function to `fzero` as a string, it would be `'x*exp(-x)-0.2'`

Math function defined by a function file

Find a solution of f(x) =0 with function $f(x) = x^3 - 2x - 5$

1. Plot the function

```
fplot(@(x)x.^3-2.*x-5, [-5 5])
```

2. First, define the function called f.m

```
function y = f(x)
y = x.^3 - 2.*x - 5;
```

3. Find zero of f(x) near 2

```
fun = @f; % function
x0 = 2; % initial point
z = fzero(fun,x0)
```

Please try other methods yourself to find the roots.

More on x_0

`x0` can be a scalar or a two-element vector

- If it is a scalar, it has to be a value of *x* near the point where the function crosses the x-axis
- If it is a vector, the two elements have to be points on opposite sides of the solution
  - This implies that `f(x0(1))` and `f(x0(2))` have different signs

A good way to get an **initial value** x_0 is to plot the function and see where it crosses the x-axis.

- Use `fplot()`

More on x_0

- If a function has **more than one** solution, you can find the different solutions by calling `fzero` once for each solution and passing a different initial value `x0` each time
  - If function is a **polynomial**, using `roots` to find the solutions will give you all of them at once
- In many applications you can estimate the domain of the solution
- Often when a function has more than one solution, only one of the solutions will have a physical meaning

Example

- Root starting with one point

  Calculate π by finding the zero of the sine function near 3

```
fun = @sin; % function
x0 = 3; % initial point
x = fzero(fun,x0)
```

- Root within an interval

  Find the zero of cosine between 1 and 2

```
fun = @cos; % function
x0 = [1 2]; % initial interval
x = fzero(fun,x0)
```

Example

- `fzero` finds zeros of a function only if they cross the x-axis, not just touch it
- If `fzero` can't find a solution, it returns `NaN`

```
>> fzero( @(x)x^2, 0.1 ) % touches at x=0
Exiting fzero: aborting search for an interval
containing a sign change because NaN or Inf function
value encountered during search.
(Function value at -1.37296e+154 is Inf.)
Check function or try again with a different
starting value.
ans =
   NaN
```

Summary

How to solve a single-variable equation g(x)=c in MATLAB?

- Step1: Transform the function if needed: f(x)=g(x)-c
- Step2: Plot the new function f(x) using MATLAB command fplot() and find initial value of x_0
- Step3: Use MATLAB function fzero() to find the value of x such that f(x)=0

What to do if f(x) is a polynomial function?

MATLAB Example

Determine the solution of the equation

$$xe^{-x} = 0.2$$

MINIMUM & MAXIMUM OF A FUNCTION

Minimum of a function

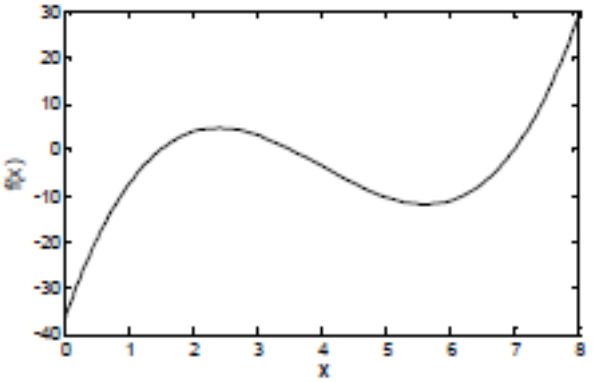
If we want to find the minimum or maximum of a function $y = f(x)$ in a certain region of the horizontal axis. We can do that with

```
[x fval] = fminbnd(function,x1,x2)
```

- `function` – is a continuous function of a single independent variable *x*
  - `function` can be a string or a handle
  - `function` must accept a single scalar *x* & must return a single scalar
- `x1` and `x2` are the left and right boundaries of the region on the x-axis, i.e., `x1 < x2`

Example

For example, consider the function $f(x) = x^3 - 12x^2 + 40.25x - 36.5$, which is plotted in the interval $0 \le x \le 8$ in the figure on the right. It can be observed that there is a local minimum between 5 and 6, and that the absolute minimum is at $x = 0$. Using the `fminbnd` command with the interval $3 \le x \le 8$ to find the location of the local minimum and the value of the function at this point gives:




```
>> [x fval]=fminbnd('x^3-12*x^2+40.25*x-36.5',3,8)
x =
    5.6073
fval =
  -11.8043
```

The local minimum is at $x = 5.6073$. The value of the function at this point is -11.8043.

Note – instead of passing function as a string, can also pass as a function handle, i.e.,

```
>> f=@(x)x^3 - 12*x^2 + 40.25*x - 36.5;
>> [x fval]=fminbnd(f,3,8)
```

Maximum of a function

To find the maximum of a function, find the minimum of the negative of the function!

EXAMPLE – find the maximum of $f(x) = xe^{-x} - 0.2$ in the range $0 \leq x \leq 8$

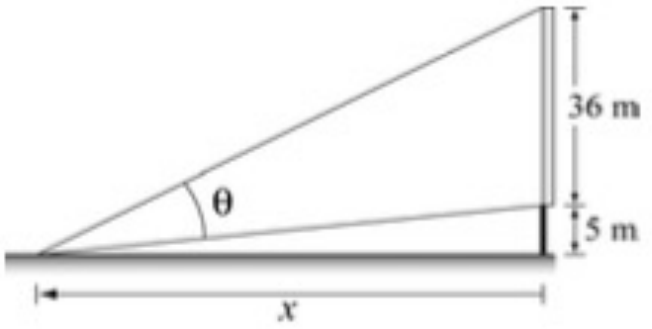
```
>> f = @(x) x*exp(-x) - 0.2;
>> [x fval] = fminbnd( @(x)-f(x), 0, 8 )
x = 1.0000
fval = -0.1679
```

so maximum is 0.1679

MATLAB Application

Sample Problem 9-4: Maximum viewing angle

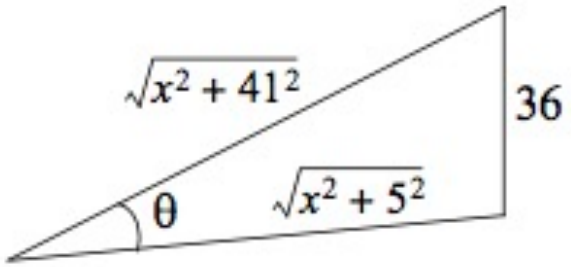
To get the best view of a movie, a person has to sit at a distance x from the screen such that the viewing angle θ is maximum. Determine the distance x for which θ is maximum for the configuration shown in the figure.



Solution

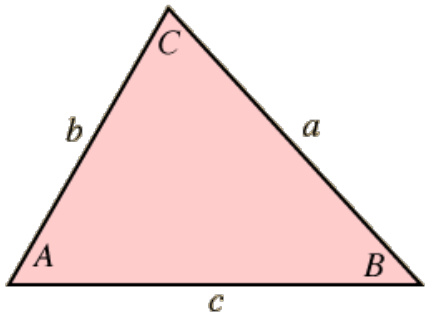
Solution

The problem is solved by writing a function for the angle θ in terms of x, and then finding the x for which the angle is maximum. In the triangle that includes θ, one side is given (the height of the screen), and the other two sides can be written in terms of x, as shown in the figure. One way in which θ can be written in terms of x is by using the Law of Cosines:



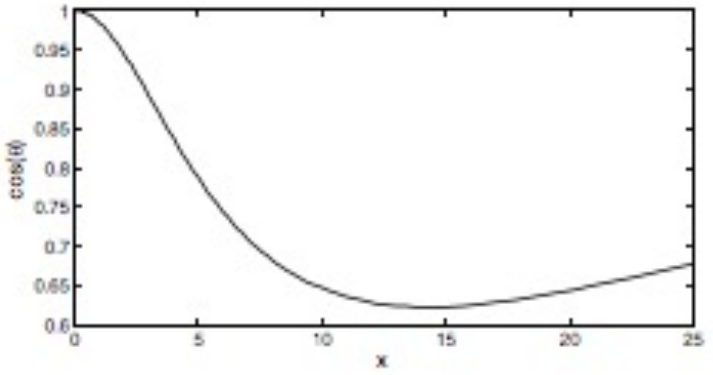
$$\cos(\theta) = \frac{(x^2+5^2)+(x^2+41^2)-36^2}{2\sqrt{x^2+5^2}\sqrt{x^2+41^2}}$$

$$c^2 = a^2 + b^2 - 2ab\cos C$$



Solution-Cont'd

The angle θ is expected to be between 0 and π/2. Since $\cos(0) = 1$ and the cosine is decreasing with increasing θ, the maximum angle corresponds to the smallest $\cos(\theta)$. A plot of $\cos(\theta)$ as a function of x shows that the function has a minimum between 10 and 20.



Exercise

Using MATLAB's built-in function `fminbnd`, determine the minimum and the maximum of the function

$$f(x) = \frac{2+(x-1.45)^2}{3+3.5(0.8x^2-0.6x+2)}$$

```
clear, clc
F = @ (x) (2+(x-1.45).^2)./(3+3.5*(0.8*x.^2-0.6*x+2));
Finv = @ (x) -(2+(x-1.45).^2)./(3+3.5*(0.8*x.^2-
0.6*x+2));
fplot(F,[-10 20])
xlabel('x')
ylabel('f(x)')
[xmin, fmin]=fminbnd(F,0,5)
[xmmax, fmax]=fminbnd(Finv,-5,0)
fmax=-fmax
```

NUMERICAL INTEGRATION

Numerical Integration

- Integration is a common mathematical operation in science and engineering.

  Examples including calculating area and volume, velocity from acceleration, and work from force and displacement, etc.

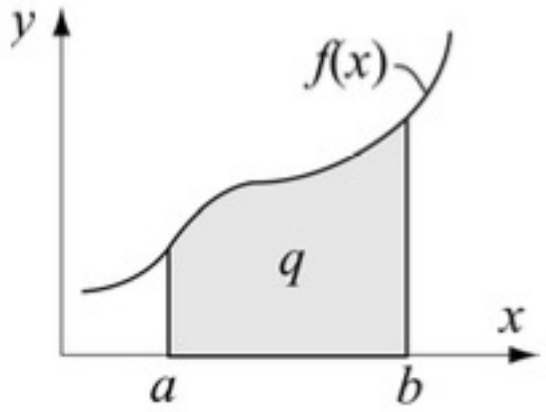
- Why is numerical integration important?

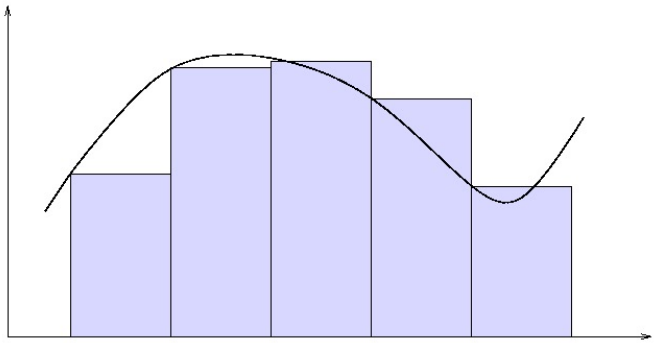
  It is impractical or impossible to solve many integrals analytically if the function is complicated. In application of engineering, the integrand can be a function or a set of data points.

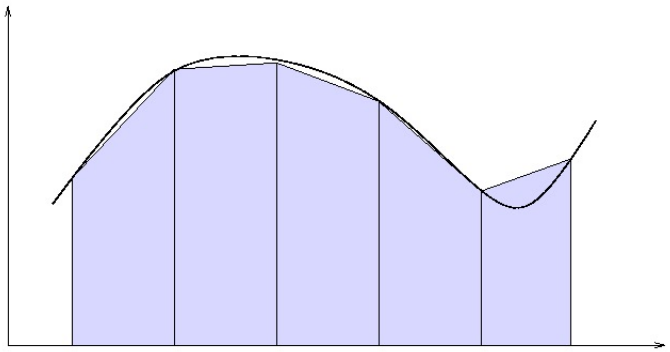
What is numerical integration?

Use numerical methods (approximation) to calculate the definite integral.

$$q = \int_a^b f(x)dx$$



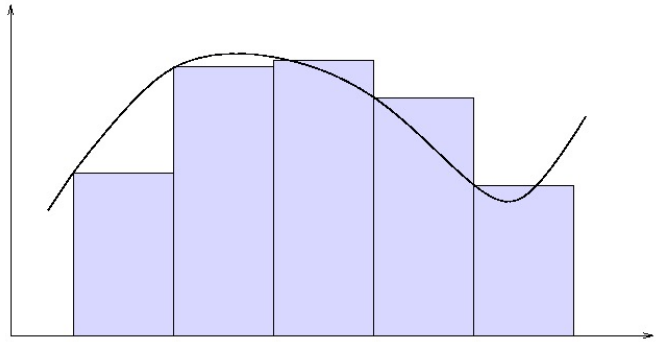


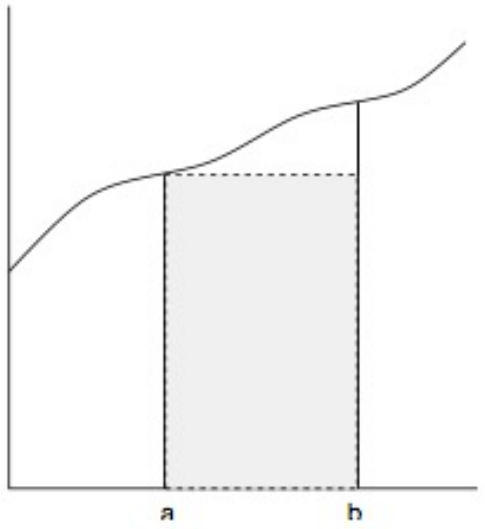


Rectangle Rule

In rectangle rule, we approximate each small area using rectangle. The height is specified by the left endpoint a.

$$I_R = f(a) \cdot (b - a)$$

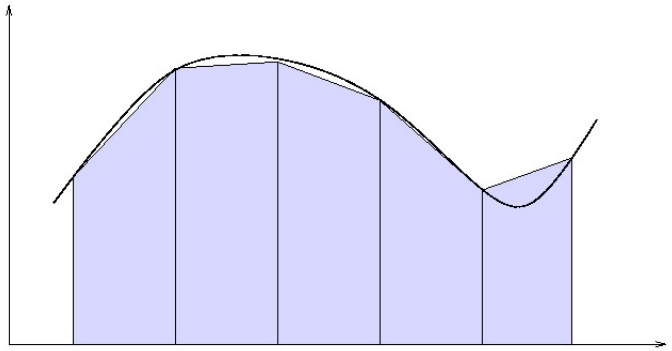


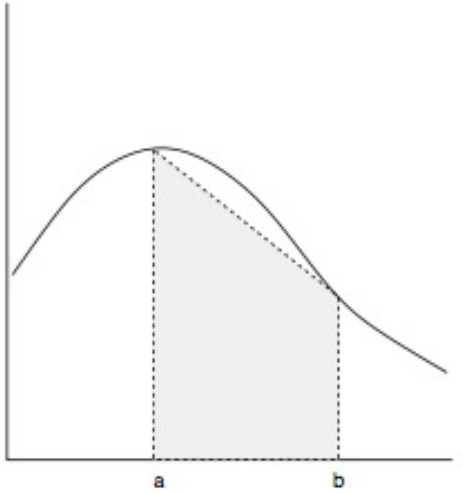


Trapezoid Rule

In the trapezoid rule, we approximate each small area using values at the both ends (a, b).

$$I_T = \frac{f(a)+f(b)}{2} \cdot (b-a)$$
$$= \frac{1}{2} f(a) \cdot (b-a) + \frac{1}{2} f(b) \cdot (b-a)$$

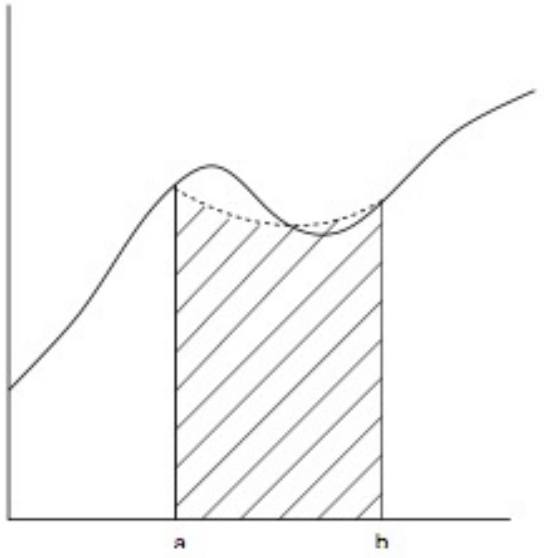




Simpson's rule

In Simpson's rule, we approximate each area using three equidistant interpolation points [a, (a+b)/2, b]. Use a polynomial of degree 2 to approximate the curve. Next, calculate the integral of this polynomial over range of [a, b].

$$I_S = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

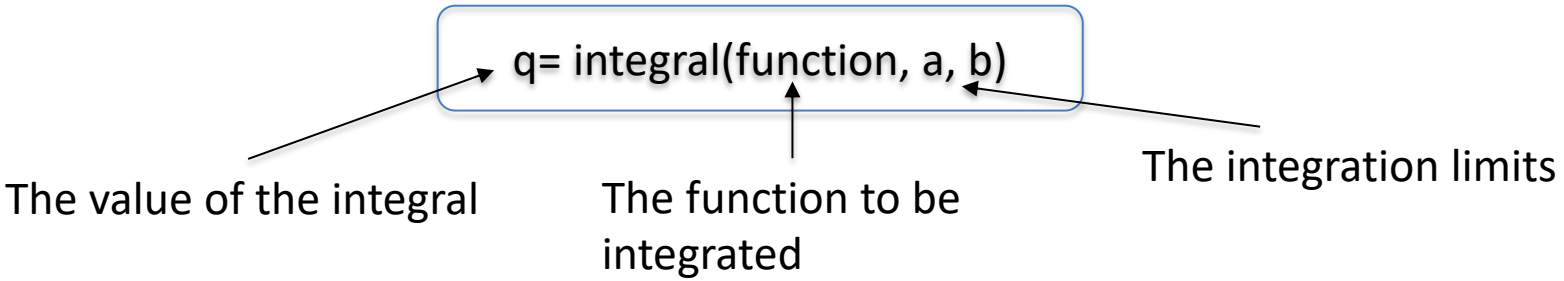


MATLAB commands

The three functions for numerical integration are `integral`, `quadgk`, and `trapz`.

- `integral()` and `quadgk()` are similar. Both takes function as input argument
- `trapz()` takes data points as input argument
- The input argument `function` must be able to take a vector argument. Be sure to use `.*`, `./`, or `.^`

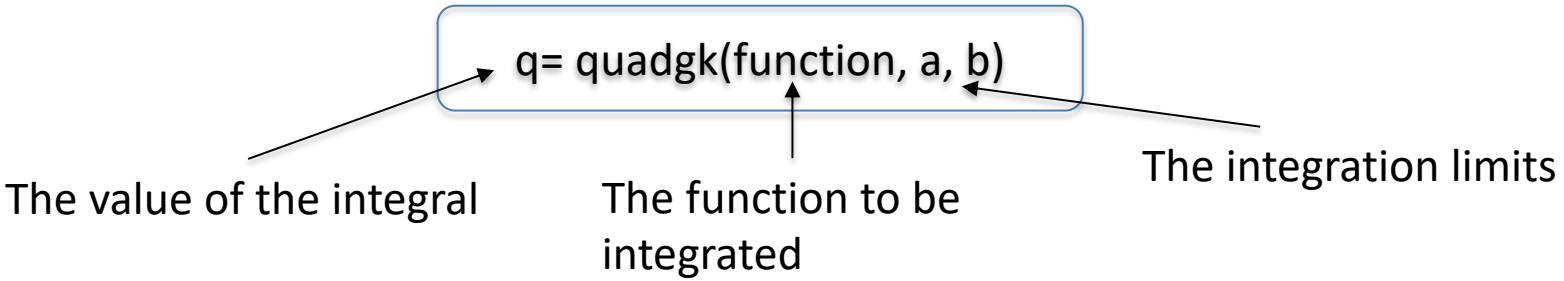
The `integral` command



- Uses numerical technique called ”global adaptive quadrature method", which can be more efficient for high accuracies and smooth integrals
- Example: compute $\int_0^1 x^2 dx$ using integral()

```
>> integral(@(x)x.*x, 0, 1 )
ans = 0.3333
```

The `quad` command



- Use this when integrand can be written as a MATLAB function
- User must make sure `function` does not have a vertical asymptote in interval `[a,b]`
- The function f(x) must be written using element-wise operations
- `quad` calculates integral with absolute error smaller than *1.0e–6*
- Uses numerical technique called "adaptive Simpson method"

Example

Use `quadgk` and an anonymous function to compute $\int_0^1 x^2 dx$
(answer is 1/3)

```
>> quadgk(@(x)x.*x, 0, 1 )
ans = 0.3333
```

The `trapz` command

```
q = trapz( x, y )
```

- Use when given *(x,y)* points, not function
  - Often occurs with experimental data
- Uses trapezoidal method of numerical integration
- `x` and `y` must be same length
- The values in `x` must be in ascending order
- The spacing between consecutive values of the sorted `x` can be non-uniform

Example

Use `trapz` and one hundred random points to compute $\int_0^1 x^2 dx$ (exact answer is 1/3)

```
>> x=sort(abs(rand(1,100));
>> y = x .* x;
>> trapz( x, y )
ans = 0.3272
```

Summary on Numerical Integration

How to solve perform numerical integration in MATLAB?

- Step1: Check if the integrand is a function or a set of data points
- Step2: Use MATLAB function `integral()`, `quadgk()`, or `trapz()`

  **Note that The input argument function must be able to take a vector argument