- We will be using the Language Fortran in this course. This section contains the syntax for the language.

* **Basic Syntax** –

```
program <name>                          Begin program scope, like main
    real :: a, b, c ..
                                        Declare variables. MUST be done at start
    integer :: a1, b1, ..
       ⋮
    write(*,*) "Enter value"
    read(*,*) a                         Use Vars in program
    write(*,*) "Value: ", a
       ⋮
end program <name>                      End program
```

* **If-then-Else**

```
if (<condition>) then
    ⋮
else if (<condition>) then
    ⋮
else
    ⋮
end
```

* **Logical Operators**

> - .gt.	< - .lt.
≥ - .ge.	≤ - .le.
= - .eq.	≠ - .ne.

* Do Loops

```
do i = 1,n
  :
end do
```
integer i takes 1 to n

```
do while (<condition>)
  :
end do
```
repeat until false

* Reading from and Writing to files

open (<Id>, file = "<name>", status = "old")

<Id> → #, used for reference.

"old" → means file is already present.

"New" would create a blank file

write (<Id>, *) " Hey"

- Write to that file

Read (<Id>, *) a

- Read from the file and store in a.

- At end of last line, you must add a new line. Otherwise, fortran encounters end of file before /n and gives an error.

* Formatting

- Write statements can be formatted in 3 ways.

  1) Ix — Integer with x columns

  2) Fw. d - Fraction with total w columns, d decimal reserved.

  3) Ew.d - Scientific with total w, d for decimal

- Examples :- 8.3 in F5.2 — _ 8 . 3 0

  8.3×10^5 in E8.2 - _ 8 . 3 0 E + 5

- Used as

```
3 format (I7)
write (*,3) a
```

* ## Function

```
<return-type> function f(<arguments>)
    real :: ....   — declare arguments here
    ⋮
    f = result   — Equate f to return value
    return
end
```

* ## Subroutine

– Used when multiple return values are needed.

In main program

```
call function(i1, i2, o1, o2)
       name
```

Syntax

```
subroutine function(i1, i2, o1, o2)
             name
    real :: ...   declare arguments
    declare i1, i2   — Say i1, i2 are inputs
    ⋮
    return
end
```

* Finding Approximate Root for function

1) Newton-Raphson method

A fairly simple way to find roots. We draw a tangent at any guess of the root, then switch over to the x-intercept of the tangent.

$$x_{n+1} = x_n - f(x_n) / f'(x_n)$$

This usually works very well. The req. are:-

1) $f'(x)$ is continuous → graph is smooth

2) $f'(x) \neq 0$

2) Secant Method

Instead of tangent, we use a secant for estimation.

$$x_{n+1} = x_n - f(x_n) / Q(x_n, x_{n-1})$$

$$Q(x_n, x_{n-1}) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

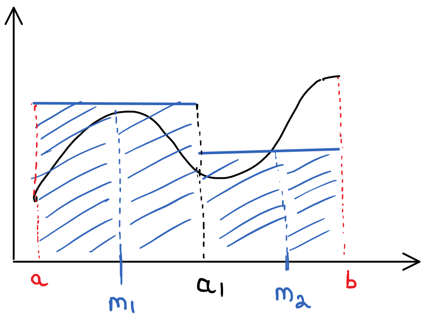
Statistically takes 45% more steps, but each step is cheaper.

* Integration

1) Midpoint Rule.

- We need to compute $\int_a^b f(x)\,dx$. Divide the interval into n parts.

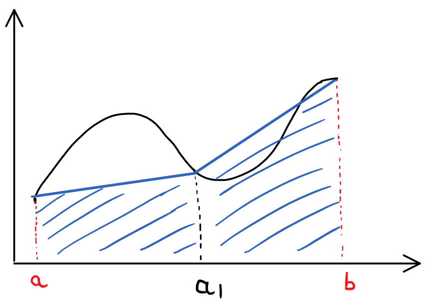
$$\Delta x = \frac{b-a}{n}$$



$$\Rightarrow I = \Delta x \sum f(m_i)$$

2) Trapezoidal rule

Instead of taking f at midpoints, we consider a trapezoid.



$$I_1 = \frac{1}{2}\Delta x\left(f(x_0) + f(x_1)\right)$$

$$I_2 = \frac{1}{2}\Delta x\left(f(x_1) + f(x_2)\right) \quad \rightarrow \quad I = \sum I_i$$

$$\vdots$$

3) Simpson's 1/3rd Rule

$$I_1 = \frac{\Delta x}{3}\left(f(x_0) + 4f(x_1) + f(x_2)\right)$$

$$I_2 = \frac{\Delta x}{3}\left(f(x_2) + 4f(x_3) + f(x_4)\right)$$

$$\vdots$$

$$I_{net} = \sum I_i$$

Solving Differential Equations

We shall initially look at first order differential equations, and the techniques used to solve them. But before that, lets set a notation so that rest of the text is easy to follow.

$$\frac{dy}{dx} = f(x,y)$$

We have also been provided with the values of $\left.\frac{dy}{dx}\right|_{x=a}$ and $y(a)$; we wish to find the value of $y(b)$.

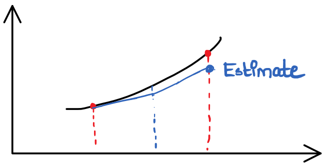
1) Euler's method

Take a step size h. Smaller step size gives better accuracy.

Let $(x_0, y_0) = (a, y(a))$. Perform the following recursion.

$$x_{n+1} \leftarrow x_n + h$$

$$y_{n+1} \leftarrow y_n + h \cdot f(x_n, y_n)$$



Stop when $x_{n+1} > b$. The value of y_n is the estimated value of $y(b)$.

2) 2^{nd} Order Runge Kutta

$$k_1 = h f(x_n, y_n)$$

$$k_2 = h f\left(x_n + h/2,\ y_n + k_1 h/2\right)$$

$$y_{n+1} \leftarrow y_n + k_2$$

$$x_{n+1} \leftarrow x_n + h$$

3) 4^{th} Order Runge-Kutta

$$K_1 = hf(x_n, y_n)$$
$$K_2 = hf(x_n + h/2, y_n + K_1/2)$$
$$K_3 = hf(x_n + h/2, y_n + K_2/2)$$
$$K_4 = hf(x_n + h, y_n + K_3)$$

$$\Rightarrow \quad y_{n+1} \leftarrow y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$
$$x_{n+1} \leftarrow x_n + h$$

* Second Order differential equations

The above methods can be quite easily applied to second order DE as well, after some clever manipulation has been done.

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right) \longrightarrow \text{Let } \frac{dy}{dx} = v$$

$$\Rightarrow \left.\begin{aligned} \frac{dv}{dx} &= f(x, y, v) \\ \frac{dy}{dx} &= v \end{aligned}\right\} \text{Simultaneously solve both } 1^{st} \text{ order DE}$$

We are provided with the values of y, $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ at $x = a$. We wish to find value of $y(b)$.

1) Euler's Method

$$x_{n+1} \longleftarrow x_n + h$$
$$y_{n+1} \longleftarrow y_n + hv_n$$
$$v_{n+1} \longleftarrow v_n + hf(x_n, y_n, v_n)$$

2) 4th Order Runge Kutta's Method

$K_1 = v_n \,, \qquad l_1 = f(x_n, y_n, v_n)$

$K_2 = v_n + h l_1/2 \,, \quad l_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{hK_1}{2}, K_2\right)$

$K_3 = v_n + h l_2/2 \,, \quad l_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{hK_2}{2}, K_3\right)$

$K_4 = v_n + h l_3 \,, \quad l_4 = f\left(x_n + h, y_n + hK_3, K_4\right)$

→

$y_{n+1} \leftarrow y_n + \frac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)$

$v_{n+1} \leftarrow v_n + \frac{h}{6}\left(l_1 + 2l_2 + 2l_3 + l_4\right)$

$x_{n+1} \leftarrow x_n + h$