PH566- Advanced Simulation Methods in Physics

- 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 ..
    integer :: a1, b1, ..               ---- Declare variables. MUST be done at start
      ⋮
    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.
$\geq$ - .ge.	$\leq$ - .le.
$=$ - .eq.	$\neq$ - .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")
```
→ means file is already present. "New" would create a blank file

↳ #, used for reference.

```
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 – _8.30

  8.3×10^5 in E8.2 – _8.30E+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.

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

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

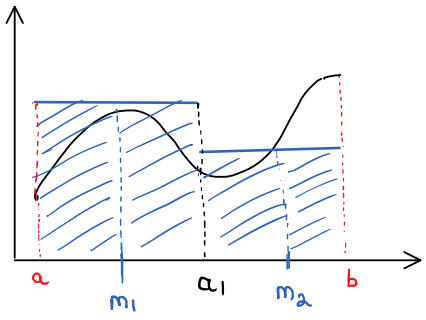
Statistically takes 45% more steps, but each step is cheaper.

* Integration

1) Midpoint Rule.

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

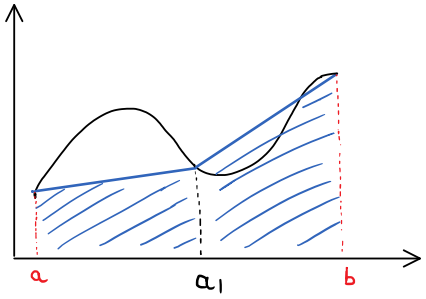
$$\Delta x = \frac{b-a}{n}$$



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

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

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

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

$$\vdots$$

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