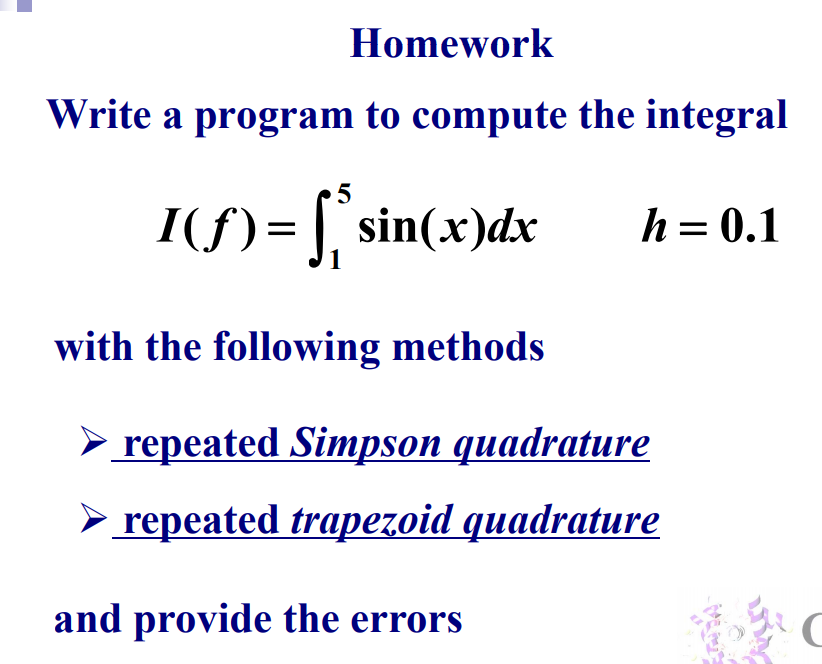
# The third schoolwork of Computational Physics

万炫均 物理1701 U201710170

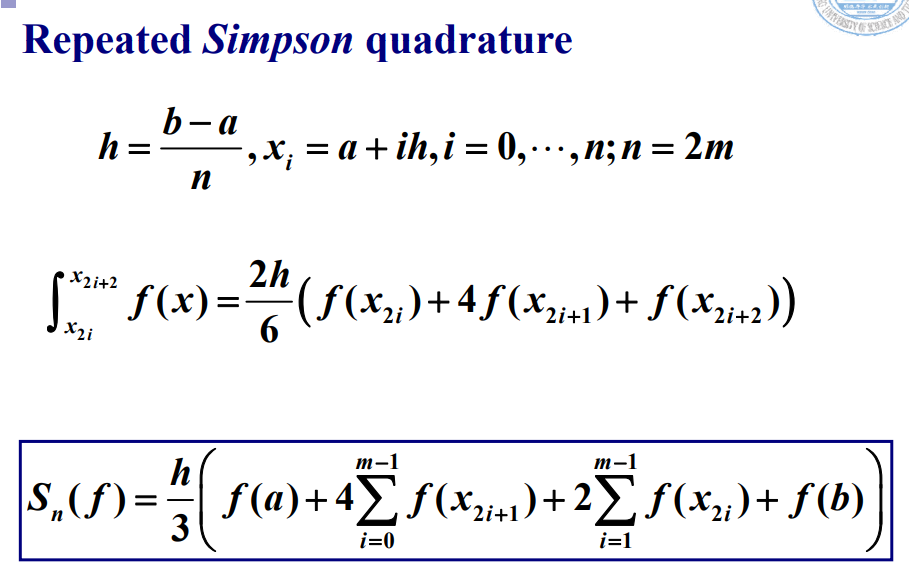
**Description of this chapter:**

Integration is always a common problem when solving Physics problems. Howerer solving formulas by ourselves can be a really hard work when the formula to integrate can be very complex. And even some times there is no specific result of an integration. Thus we can turn to computer to help us calculate some integration in value.

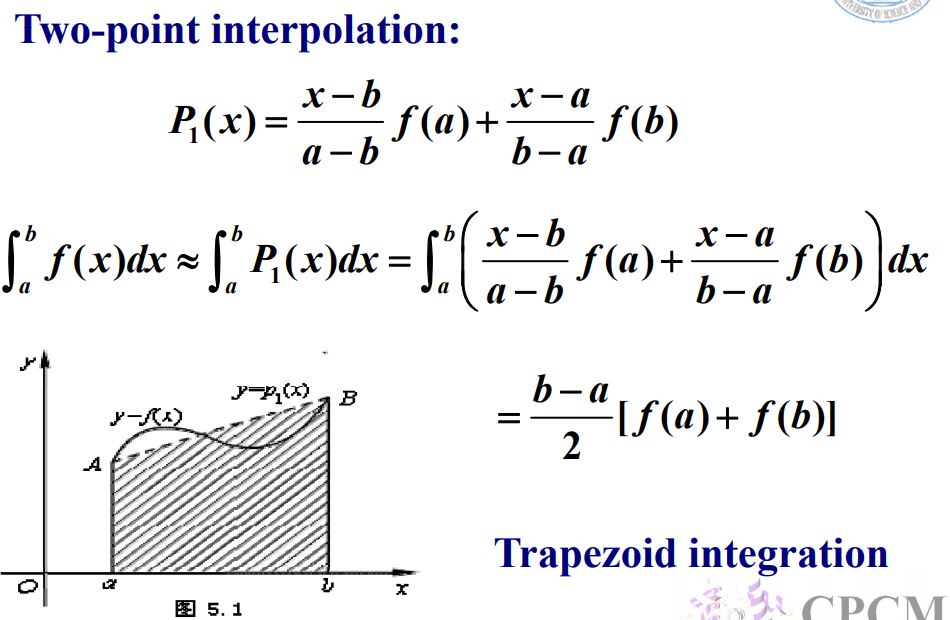
* **Description of the problem**



* **Formula to use**
  + Simpson quadrature



* + Trapezoid quadrature



* **Flow chart**
* **Source code**
* program Main
* use Integrate
* real\*8 :: simpson\_result,trapezoid\_result,real\_result,f\_origin
* real\*8 :: step
* external f
* procedure(func),pointer ::  f\_ptr=>null()
* f\_ptr => f
* step =0.01
* call Simpson(f\_ptr,dble(1.0),dble(5.0),step,simpson\_result)
* call Trapezoid(f\_ptr,dble(1.0),dble(5.0),step,trapezoid\_result)
* real\_result = f\_origin(dble(5.0))-f\_origin(dble(1.0))
* print "(a,es20.10)","Real result",real\_result
* print "(a,es20.10)","Simpson Integration:",simpson\_result
* print "(a,es20.10)","Simpson Error:",abs(simpson\_result-real\_result)
* print "(a,es20.10)","Trapezoid Integration:",trapezoid\_result
* print "(a,es20.10)","Trapezoid Error:",abs(trapezoid\_result-real\_result)
* end program Main
* function f(x)
* real\*8,intent(in) :: x
* real\*8 :: f
* *!TODO\_add\_body*
* f=sin(x)
* end function f
* function f\_origin(x)
* real\*8 :: x
* real\*8 :: f\_origin
* *!TODO\_add\_body*
* f\_origin=-cos(x)
* end function f\_origin

module Integrate

    abstract interface

        function func(x)

            real\*8,intent(in) :: x

            real\*8 :: func

        end function func

    end interface

contains

    subroutine Simpson(f\_ptr, x\_start,x\_end,step,result)

        procedure(func),pointer,intent(in) ::  f\_ptr

        real\*8,intent(in)  :: x\_start,x\_end,step

        real\*8,intent(out) :: result

*!Local vars*

        real\*8 :: x,h

        result = 0

        x=x\_start

        h=step

        do while(.true.)

            if(x+h>x\_end)then

                h=x\_end-x

            end if

            result=result+h/6\*(f\_ptr(x)+4\*f\_ptr(x+h/2)+f\_ptr(x+h))

            x=x+h

            if (x>=x\_end)then

                exit

            end if

        end do

*!TODO\_add\_body*

    end subroutine Simpson

    subroutine Trapezoid (f\_ptr, x\_start,x\_end,step,result)

        procedure(func),pointer,intent(in) ::  f\_ptr

        real\*8,intent(in)  :: x\_start,x\_end,step

        real\*8,intent(out) :: result

*!Local vars*

        real\*8 :: x,h

        result = 0

        x=x\_start

        h=step

        do while(.true.)

            if(x+h>x\_end)then

                h=x\_end-x

            end if

            result=result+(f\_ptr(x)+f\_ptr(x+h))\*h/2

            x=x+h

            if (x>=x\_end)then

                exit

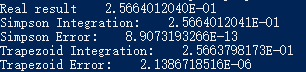
            end if

        end do

    end subroutine Trapezoid

end module Integrate

* **Example and Result**



* **Demo**

Check the folder ”Equations” in the directory and follow the instruction to set up the matrices and vectors