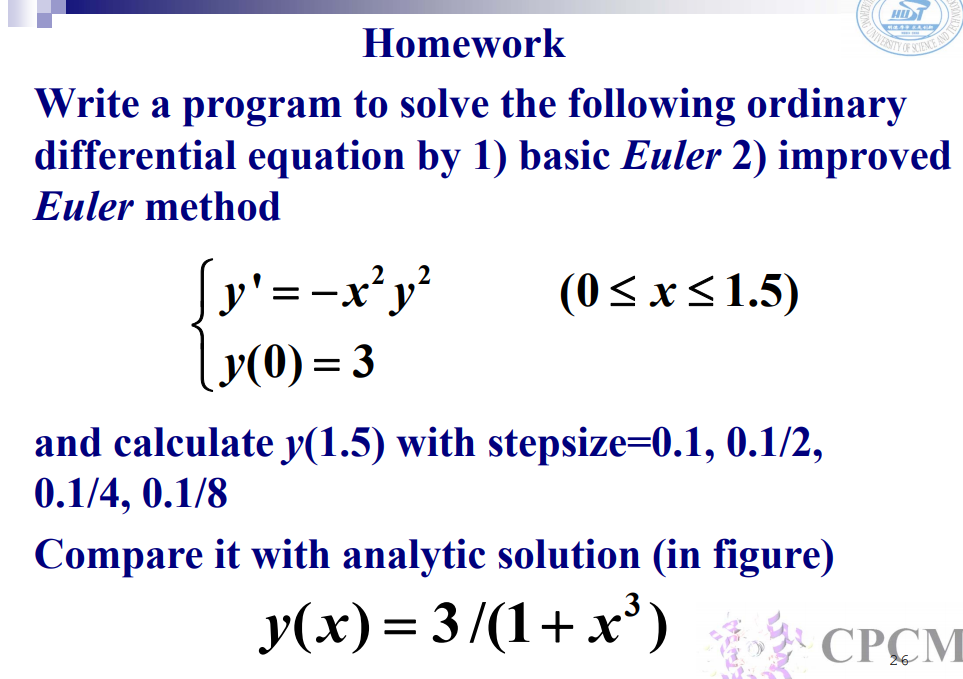
# The sixth schoolwork of Computational Physics

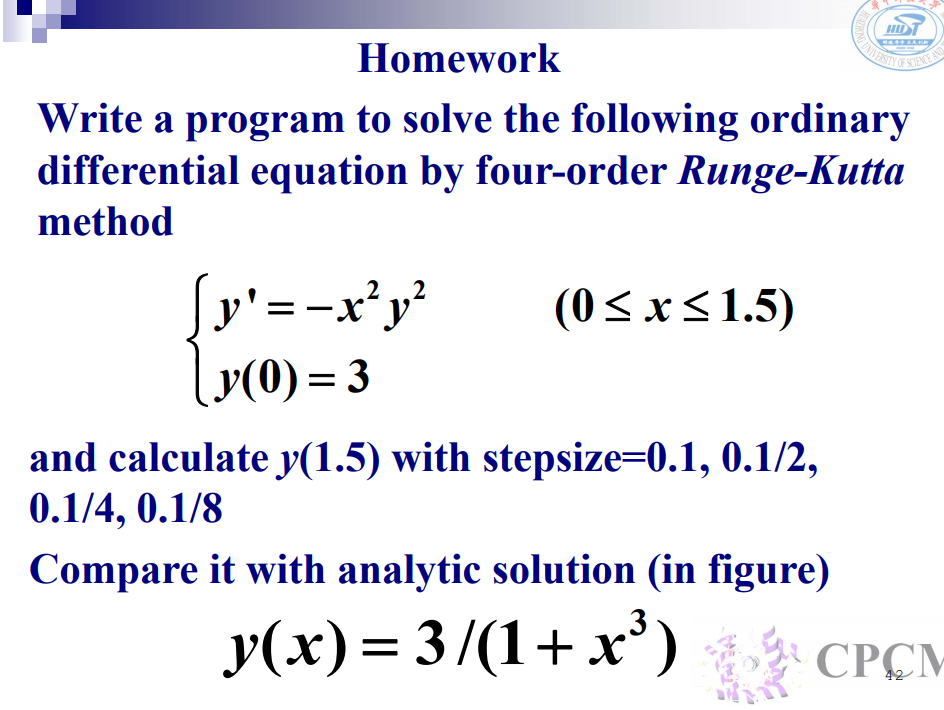
万炫均 物理1701 U201710170

**Description of this chapter:**

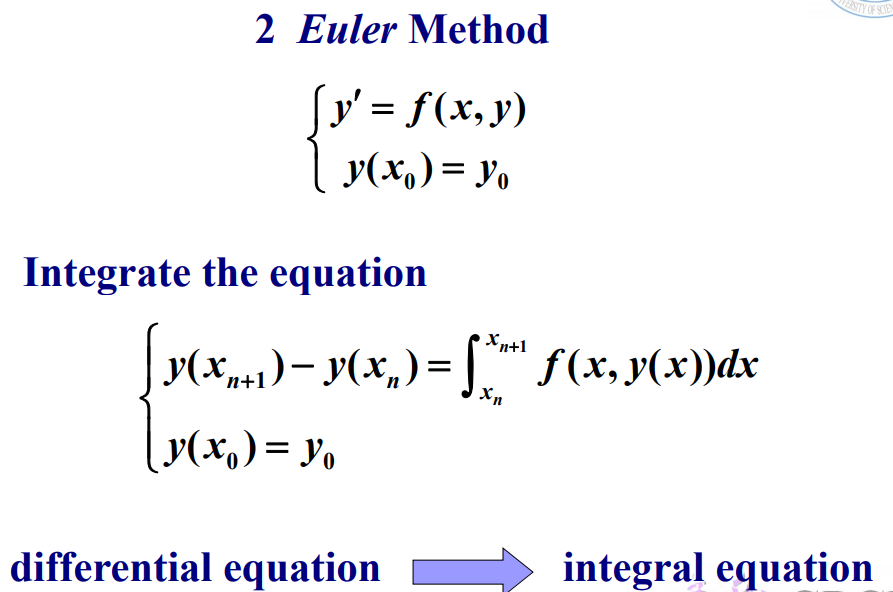
Solving ordinary derivative equations is a very common problem in physics. Here we use basic Euler method and Runge-Kutta method to solve ordinary equations by iterative methods.

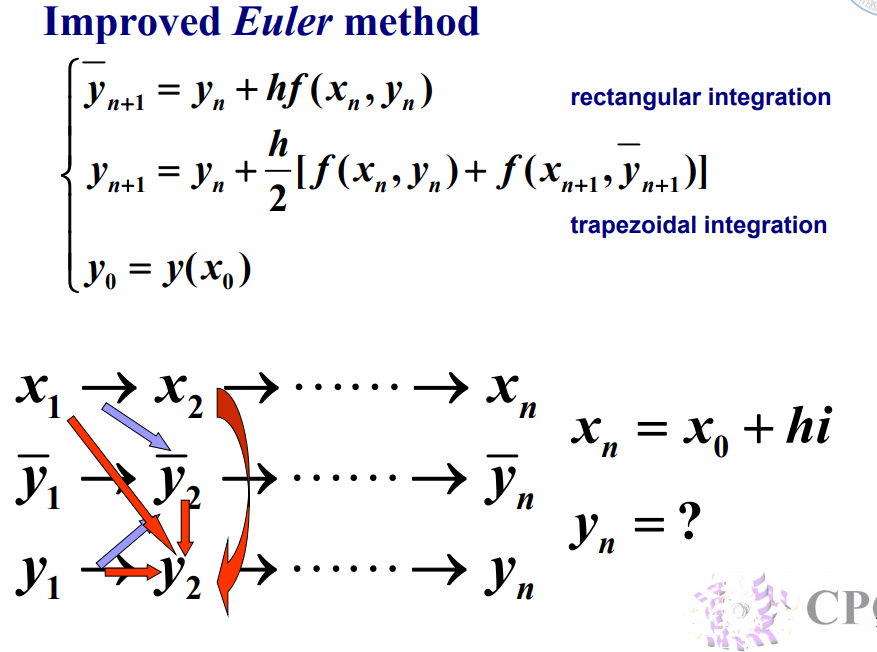
**Description of the problem**



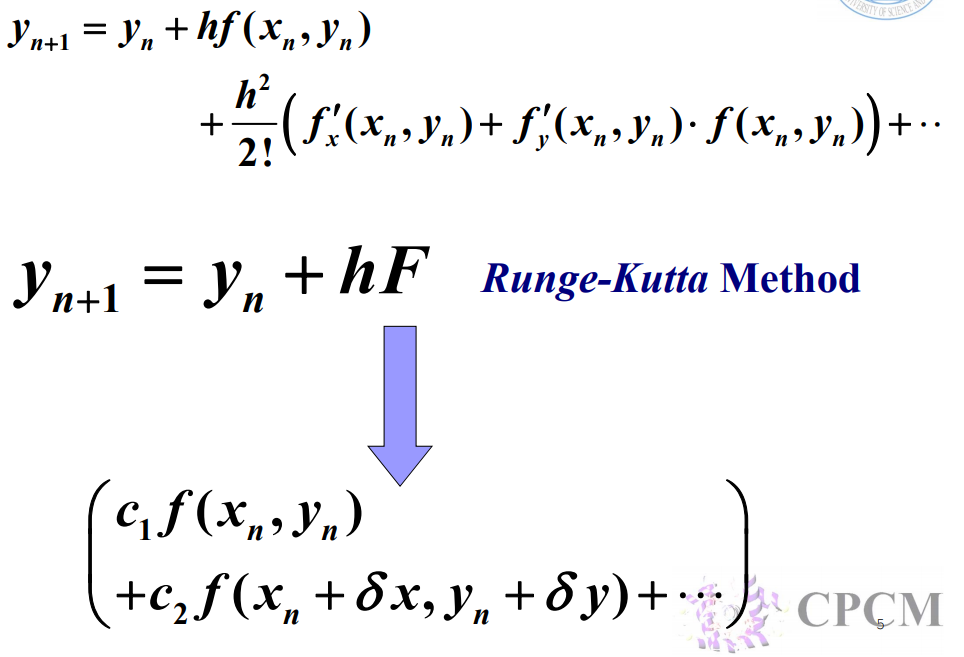


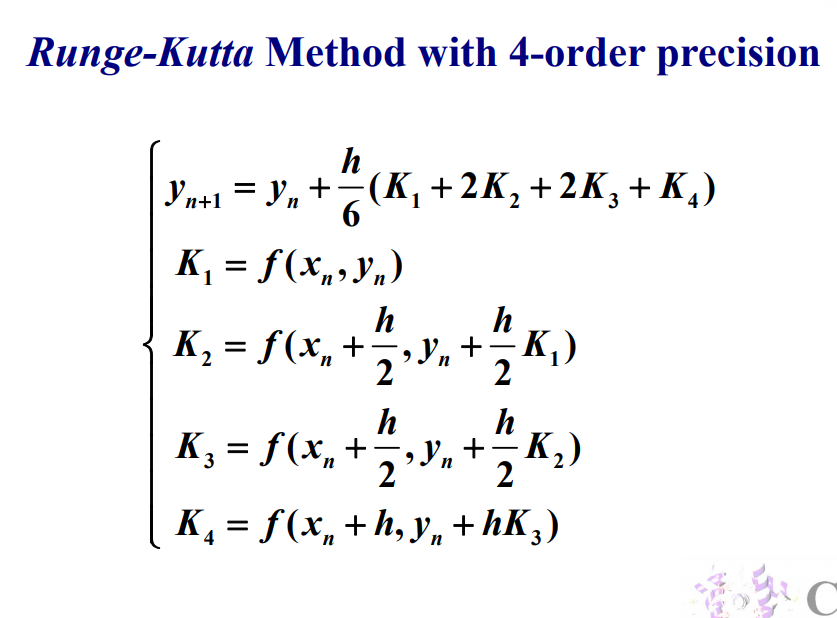
* **Formula to use**
  + Euler Method



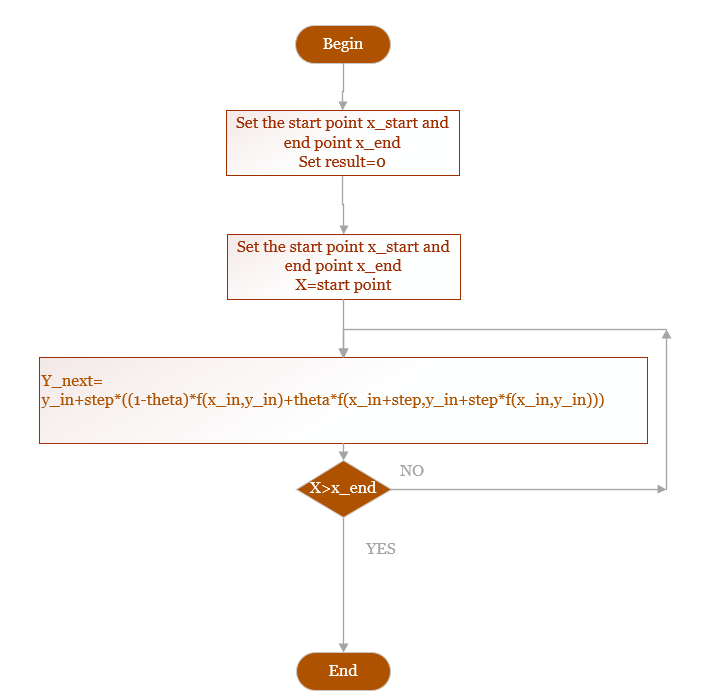


* + Runge-Kutta Method

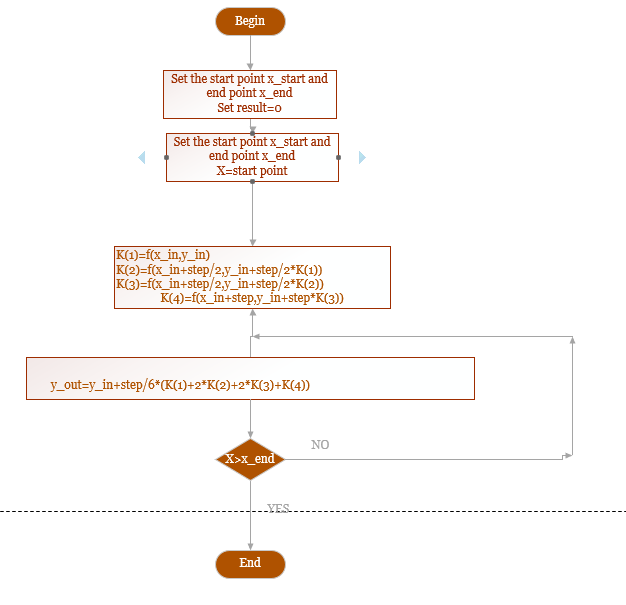




* **Flow chart**
  + **Euler Method Flowchart**



* + **Runge-Kutta Method Flowchart**



* **Source code**
* program MainProgram
* do i=1,4
* call Euler0(0.1/dble(i))
* call Euler5(0.1/dble(i))
* call RK4(0.1/dble(i))
* end do
* end program MainProgram
* subroutine Euler0(step)
* !TODO\_add\_body
* use ODE\_Solver
* use IO
* real\*8,intent(in)  :: step
* real\*8 :: x,y,y\_next
* character(len=64) :: path
* procedure (func), pointer :: f\_ptr => null()
* external f
* f\_ptr => f
* x=0
* y=3
* write (path,"(a,f4.3,a)")"./data/euler0\_",step,".txt"
* call ClearFile(path)
* do while(x<=1.5)
* call EulerSolver(x,y,f\_ptr,step,dble(0),y\_next)!0 for explicit Euler solver
* call WriteNumToFile(path,y\_next,.true.)
* y=y\_next
* x=x+step
* end do
* end subroutine Euler0
* subroutine Euler5(step)
* !TODO\_add\_body
* use ODE\_Solver
* use IO
* real\*8,intent(in)  :: step
* real\*8 :: x,y,y\_next
* character(len=64) :: path
* procedure (func), pointer :: f\_ptr => null()
* external f
* f\_ptr => f
* x=0
* y=3
* write (path,"(a,f4.3,a)")"./data/euler5\_",step,".txt"
* call ClearFile(path)
* do while(x<=1.5)
* call EulerSolver(x,y,f\_ptr,step,dble(0.5),y\_next)!0 for explicit Euler solver
* call WriteNumToFile(path,y\_next,.true.)
* y=y\_next
* x=x+step
* end do
* end subroutine Euler5
* subroutine RK4(step)
* !TODO\_add\_body
* !TODO\_add\_body
* use ODE\_Solver
* use IO
* real\*8,intent(in)  :: step
* real\*8 :: x,y,y\_next
* character(len=64) :: path
* procedure (func), pointer :: f\_ptr => null()
* external f
* f\_ptr => f
* x=0
* y=3
* write (path,"(a,f4.3,a)")"./data/rk4\_",step,".txt"
* call ClearFile(path)
* do while(x<=1.5)
* call RungeKutta4(x,y,f\_ptr,step,y\_next)!0 for explicit Euler solver
* call WriteNumToFile(path,y\_next,.true.)
* y=y\_next
* x=x+step
* end do
* end subroutine RK4
* function f (x,y)
* real\*8, intent (in) :: x,y
* real\*8 :: f
* f=-(x\*\*2)\*(y\*\*2)
* end function f

module ODE\_Solver

    implicit none

    abstract interface

        !Function form definition of function f to define the function pointer

        function func (x,y)

            real\*8, intent (in) :: x,y

            real\*8 :: func

        end function func

    end interface

contains

    subroutine EulerSolver(x\_in,y\_in,f\_ptr,step,theta,y\_out)

        !y\_in:The current inputted y value

        !x\_in:Thecurrent x value

        !f\_ptr:The function pointer to f(x,y)

        !step:The step interval

        !theta:0 for the explicit euler function,1 for the implicit euler function

        !y\_out:The out put of the next y value

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

        real\*8,intent(in) :: y\_in,x\_in,step

        real\*8,optional::theta

        real\*8,intent(out) :: y\_out

        if (.not.present(theta))then

            theta = 0.5

        end if

        y\_out=y\_in+step\*((dble(1)-theta)\*f\_ptr(x\_in,y\_in)+theta\*f\_ptr(x\_in+step,y\_in+step\*f\_ptr(x\_in,y\_in)))

    end subroutine

    subroutine RungeKutta4(x\_in,y\_in,f\_ptr,step,y\_out)

        !y\_in:The current inputted y value

        !x\_in:Thecurrent x value

        !f\_ptr:The function pointer to f(x,y)

        !step:The step interval

        !y\_out:The out put of the next y value

        procedure (func), pointer :: f\_ptr

        real\*8,intent(in) :: y\_in,x\_in,step

        real\*8,intent(out) :: y\_out

        !Local vars

        real\*8 :: K(4)

        !Setup K values

        K(1)=f\_ptr(x\_in,y\_in)

        K(2)=f\_ptr(x\_in+step/2,y\_in+step/2\*K(1))

        K(3)=f\_ptr(x\_in+step/2,y\_in+step/2\*K(2))

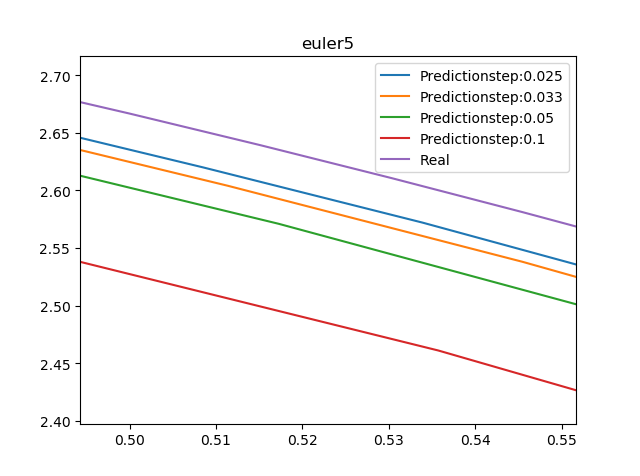
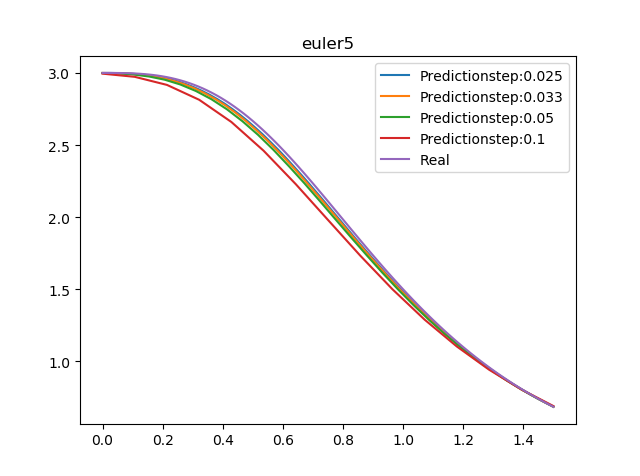
        K(4)=f\_ptr(x\_in+step,y\_in+step\*K(3))

        y\_out=y\_in+step/6\*(K(1)+2\*K(2)+2\*K(3)+K(4))

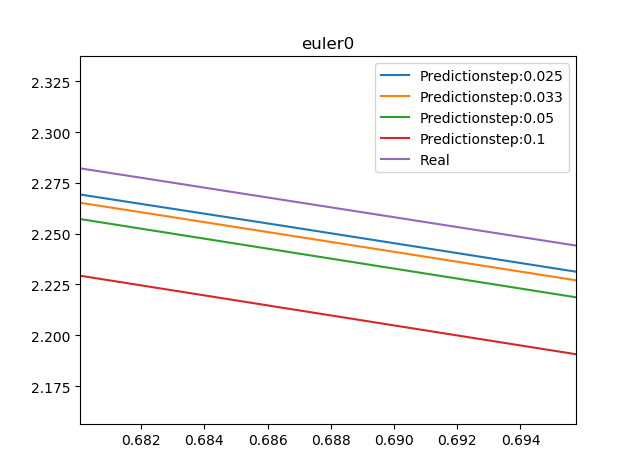
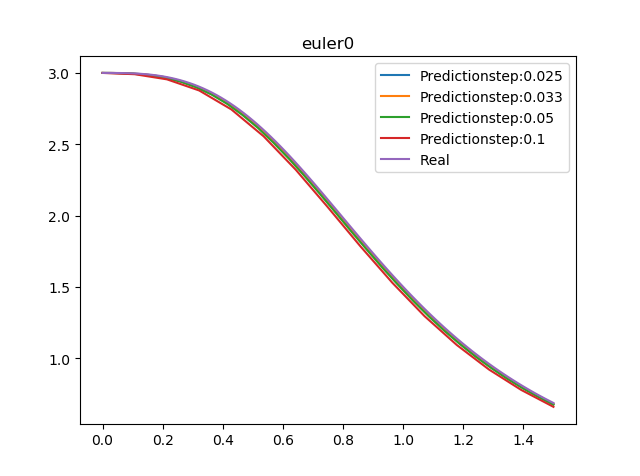
    end subroutine

end module ODE\_Solver

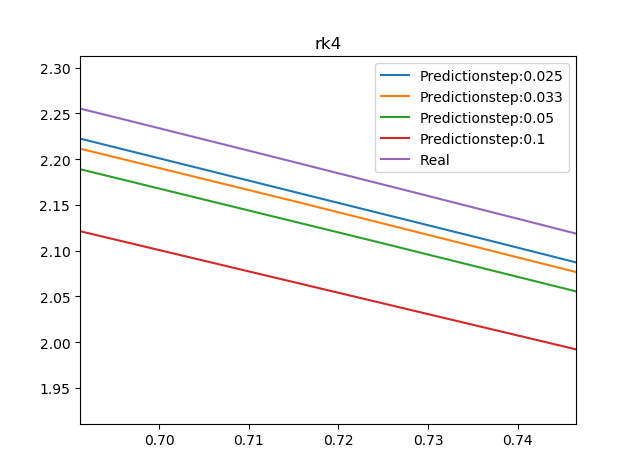
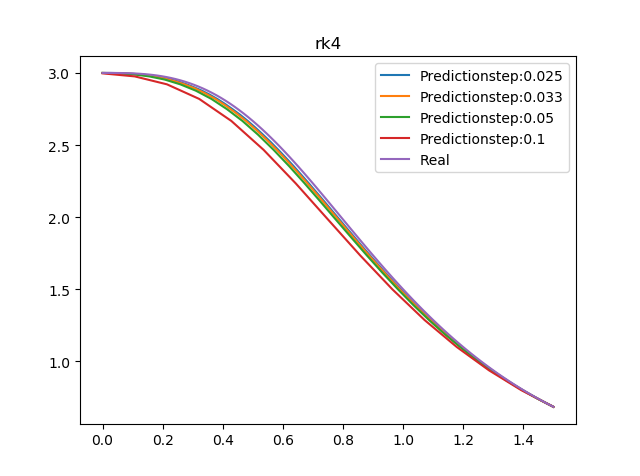
* **Example and Result**
  + **Improved Euler Method**



* + **Explicit Euler Method**



* + **4-order Runge-Kutta method**



* **Demo**

Check the folder ”ODE” in the directory and follow the instruction to set up the matrices and vector