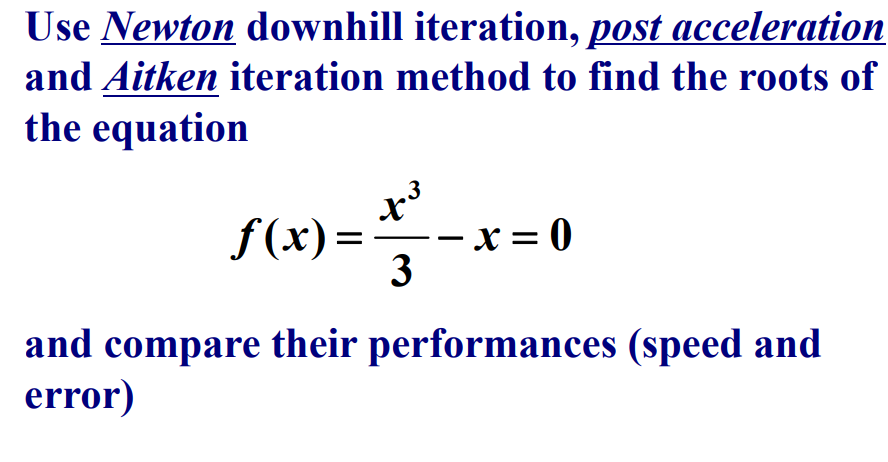
# The second schoolwork of Computational Physics

万炫均 物理1701 U201710170

**Description of this chapter:**

For this chapter, we try various of computing methods to find out roots of functions, which is a quite basic problem in mathematic.

* **Description of the problem**



* **Formula to use**

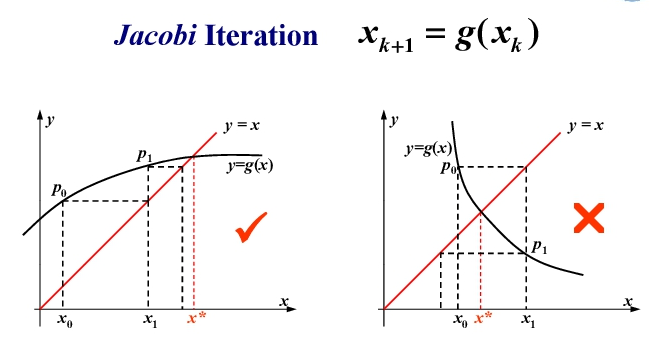
Here we will use four methods in total:

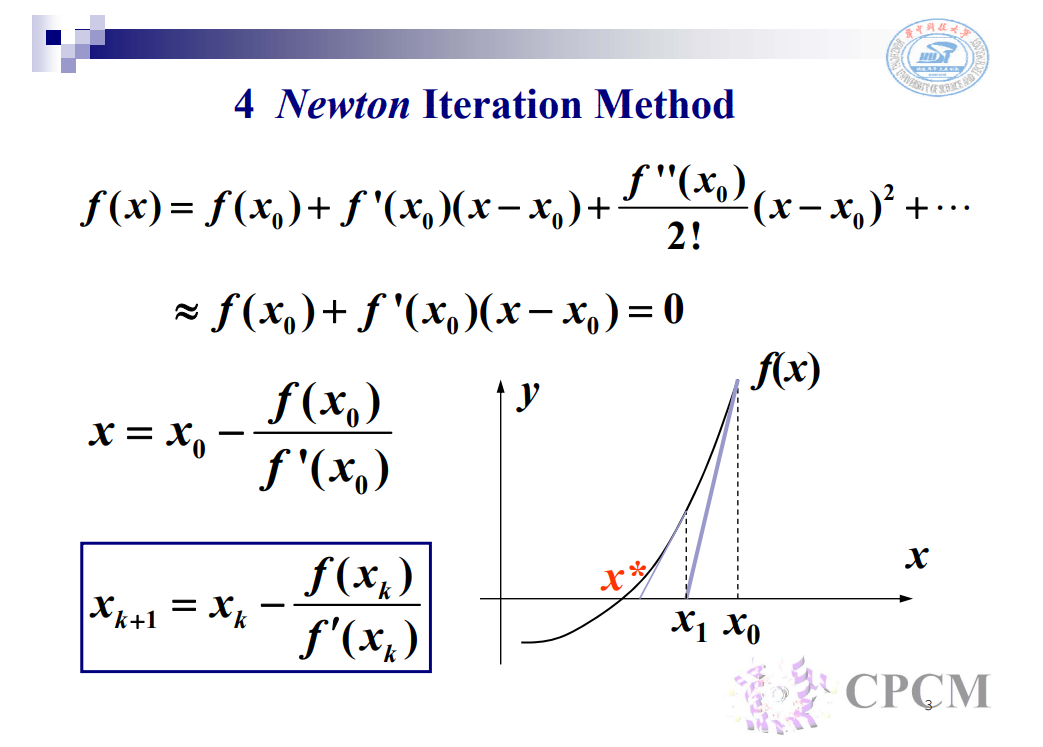
Jacobi method,

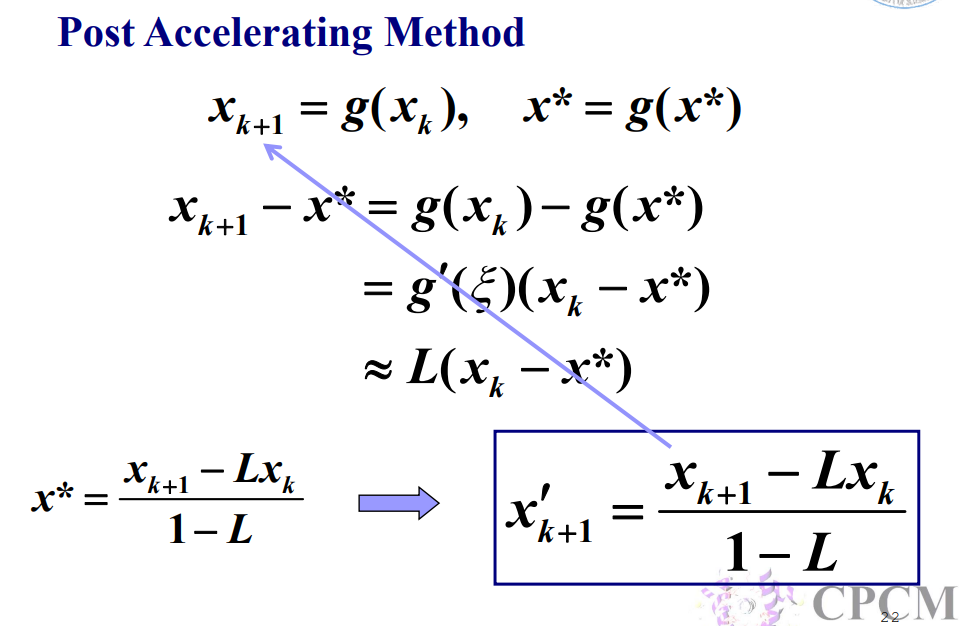
Newtown downhill method,

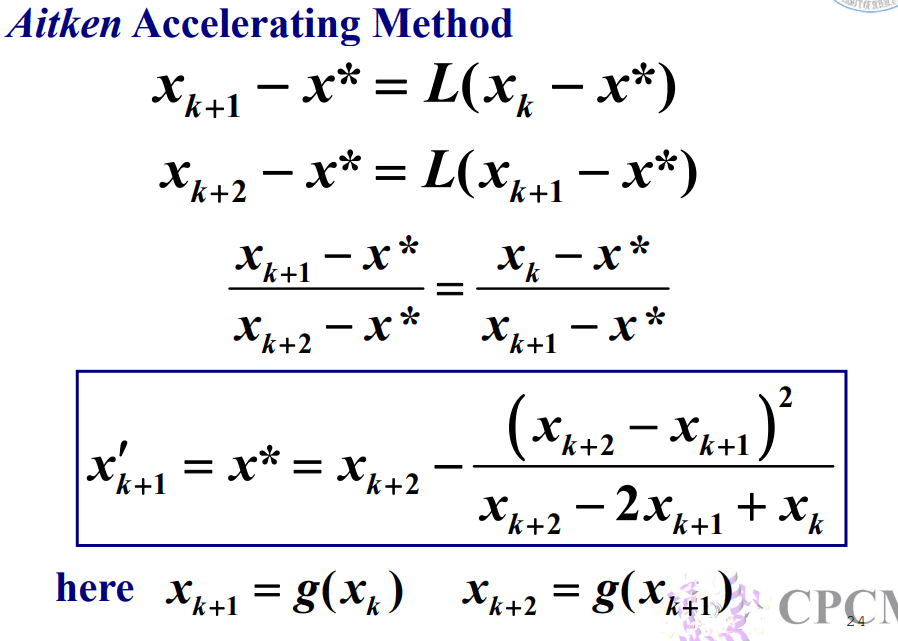
Post acceleration method,

Aitken method.



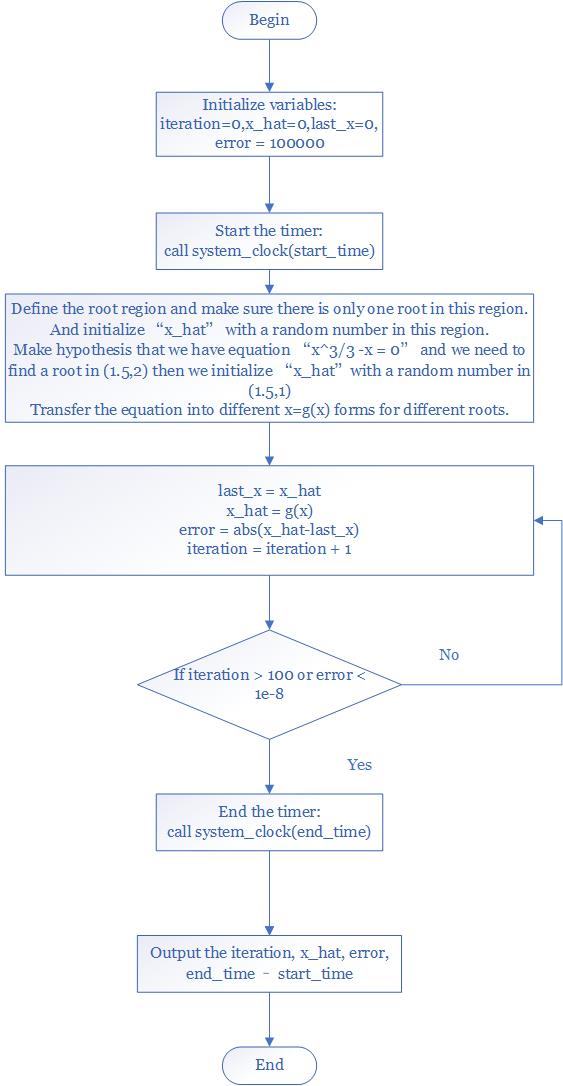




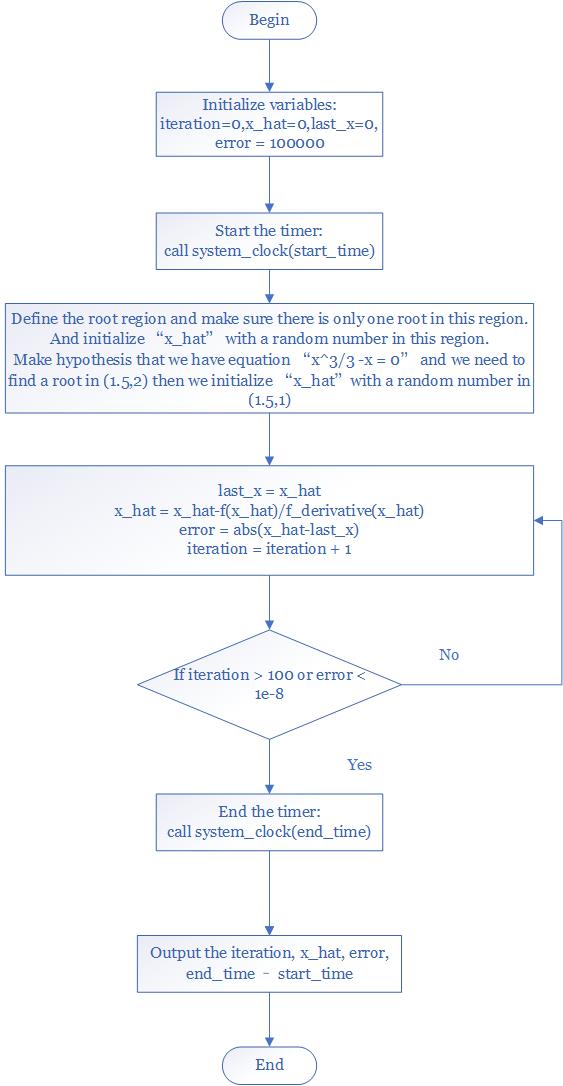


* **Flow chart**

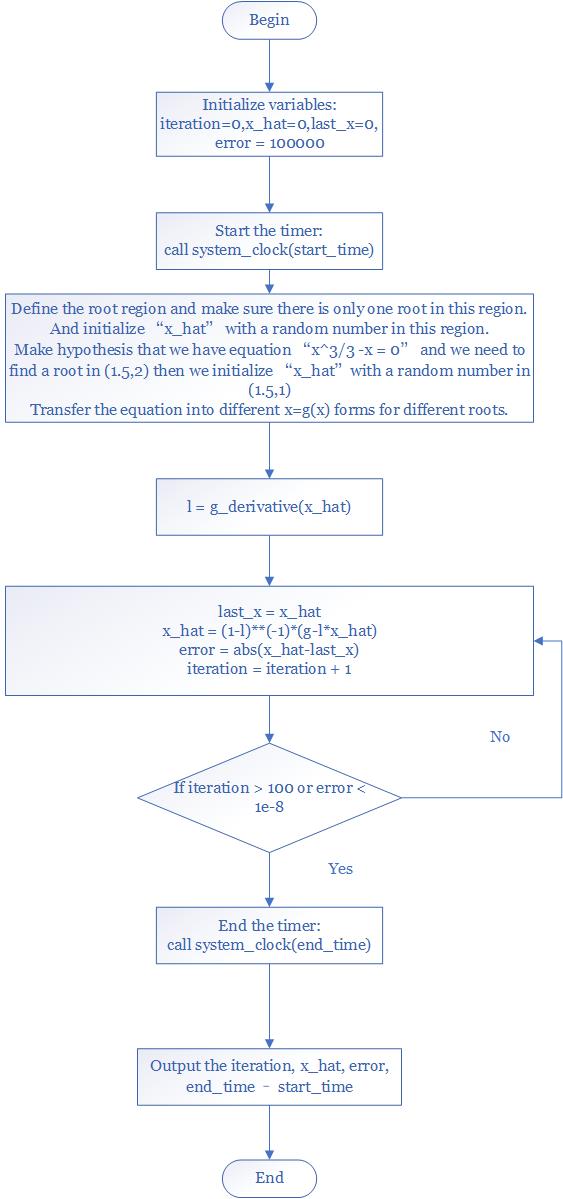
1. **Jacobi Method Flow Chart**

****

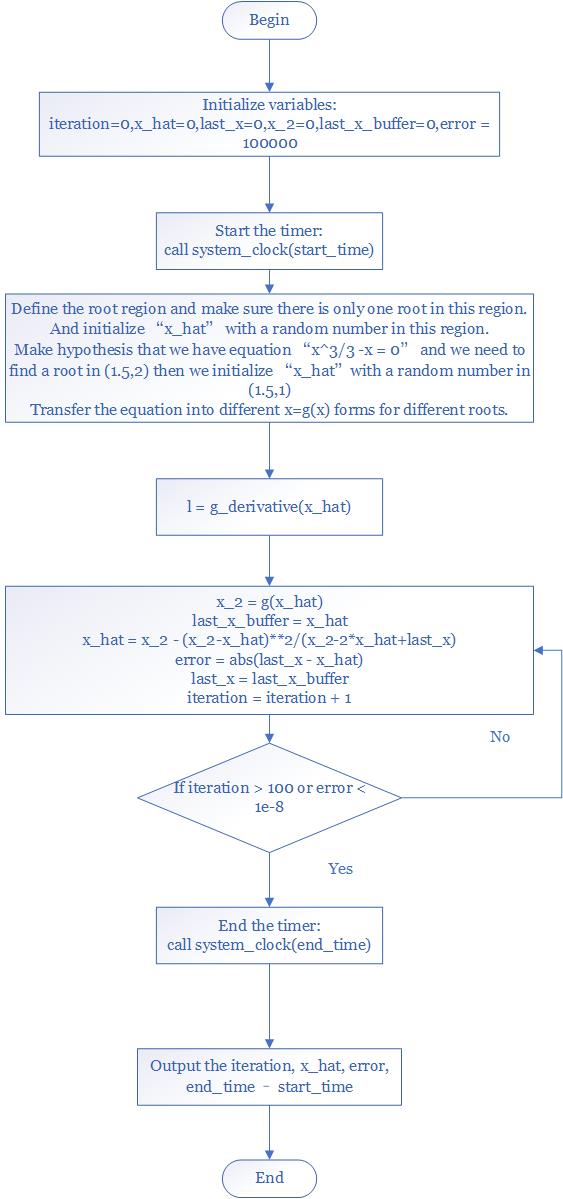
1. **Newton Downhill Method Flow Chart**



1. **Post Accelerating Method Flow Chart**

****

1. **Aitken Method Flow Chart**

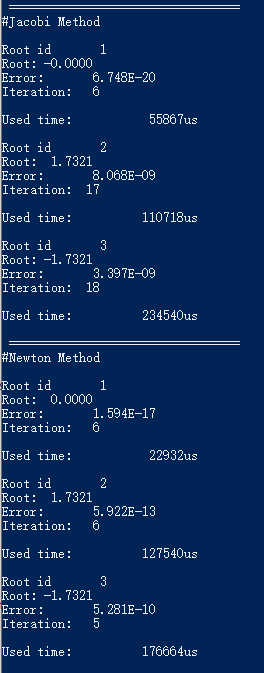
****

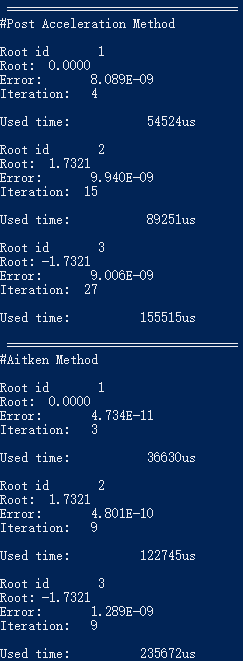
* **Source Code**

Here is the code. I have compressed the four methods of finding all the three roots into one program so it is easy to run the program.

* program Root
* real\*8 :: x\_hat,error
* integer :: iteration
* *!Clock to measure*
* integer\*8 :: start\_time,end\_time
* *!Procedures for different methods*
* do j = 1,4
* *!Start the system timer*
* call system\_clock(start\_time)
* *!Loop for different roots*
* print \*,"================================="
* do i = 1,3
* select case(j)
* case(1)
* call JacobiCalculate(x\_hat,error,iteration,i)
* if (i==1) then
* print "(a,/)","#Jacobi Method"
* end if
* case(2)
* call NewtonCalculate(x\_hat,error,iteration,i)
* if (i==1) then
* print "(a,/)","#Newton Method"
* end if
* case(3)
* call AccCalculate(i, x\_hat,error,iteration)
* if (i==1) then
* print "(a,/)","#Post Acceleration Method"
* end if
* case(4)
* call AitkenCalculate(i, x\_hat,error,iteration)
* if (i==1) then
* print "(a,/)","#Aitken Method"
* end if
* case default
* print "(a,/)","Unknown procedure id!!!"
* end select
* *!Stop the system timer*
* call system\_clock(end\_time)
* print "(a,i8)","Root id",i
* print "(a,f8.4,/,a,es16.3,/,a,i4,/a,es16.3)","Root:",x\_hat,"Error:",error,"Iteration:",iteration
* print "(a,i16,/)","Used time:",(end\_time-start\_time)
* end do
* end do
* end program Root
* *!The core calculation subroutine*
* subroutine JacobiCalculate(x\_hat,error,iteration,root\_id)
* implicit none
* real\*8 :: g1
* real\*8 :: g2
* integer,intent(in):: root\_id
* real\*8,intent(out) :: x\_hat,error
* integer,intent(out) :: iteration
* real\*8 :: last\_x
* iteration = 0
* x\_hat = 0
* last\_x = 0
* *!Set a large number*
* error = 1000000
* do while(abs(error)>1d-8 .and. iteration < 100)
* last\_x = x\_hat

* *!Choose different functions for different roots*
* select case(root\_id)
* case (1)
* x\_hat = g1(x\_hat)
* *!Set the x a random number in (-1,0)*
* if ( iteration == 0 ) then
* call random\_number(x\_hat)
* x\_hat = x\_hat\*(-1)
* end if
* case (2)
* x\_hat = g2(x\_hat)
* *!Set the x a random number in (1,2)*
* if ( iteration == 0 ) then
* call random\_number(x\_hat)
* x\_hat = x\_hat+1
* end if
* case (3)
* x\_hat = g2(x\_hat)
* *!Set the x a random number in (-2,-1)*
* if ( iteration == 0 ) then
* call random\_number(x\_hat)
* x\_hat = x\_hat\*(-1)-1
* end if
* end select
* error = abs(x\_hat-last\_x)
* iteration = iteration + 1
* end do
* end subroutine JacobiCalculate
* subroutine NewtonCalculate(x\_hat,error,iteration,root\_id)
* integer,intent(in) :: root\_id
* real\*8,intent(out) ::x\_hat,error
* integer,intent(out) :: iteration
* real\*8 :: f,f\_derivative
* real\*8 :: last\_x
* *!Initialize the x with a random number in different regions*
* select case(root\_id)
* case (1)
* *!Set the x a random number in (0,1)*
* call random\_number(x\_hat)
* case (2)
* *!Set the x a random number in (1,2)*
* call random\_number(x\_hat)
* x\_hat = x\_hat+1
* case (3)
* *!Set the x a random number in (-2,-1)*
* call random\_number(x\_hat)
* x\_hat = (x\_hat+1)\*(-1)
* end select
* iteration = 0
* *!Set a large number*
* error = 10000000
* *!Main calculation loop*
* do while(abs(error)>1d-8 .and. iteration < 100)
* last\_x = x\_hat
* x\_hat = x\_hat-f(x\_hat)/f\_derivative(x\_hat)
* error = abs(last\_x - x\_hat)
* iteration = iteration + 1
* end do
* end subroutine
* *!Post Acceleration Method*
* subroutine AccCalculate(root\_id, x\_hat,error,iteration)
* implicit none
* integer,intent(in) :: root\_id
* real\*8,intent(out) :: x\_hat,error
* integer,intent(out) :: iteration
* real\*8 :: g,g1,g2,g1\_derivative,g2\_derivative,l
* real\*8 :: last\_x
* *!Initialize the x with a random number in different regions*
* *!Initialize the factor l*
* select case(root\_id)
* case (1)
* *!Set the x a random number in (0,1)*
* call random\_number(x\_hat)
* l = g1\_derivative(x\_hat)
* case (2)
* *!Set the x a random number in (1,2)*
* call random\_number(x\_hat)
* x\_hat = x\_hat+1
* l = g2\_derivative(x\_hat)
* case (3)
* *!Set the x a random number in (-2,-1)*
* call random\_number(x\_hat)
* x\_hat = x\_hat\*(-1)-1
* l = g2\_derivative(x\_hat)
* end select
* iteration = 0
* *!Set a large number*
* error = 10000000
* do while(abs(error)>1d-8 .and. iteration < 100)
* last\_x = x\_hat
* select case(root\_id)
* case(1)
* g = g1(x\_hat)
* case default
* g = g2(x\_hat)
* end select
* x\_hat = (1-l)\*\*(-1)\*(g-l\*x\_hat)
* error = abs(last\_x - x\_hat)
* iteration = iteration + 1
* end do
* end subroutine AccCalculate
* *!Aitken Calculation*
* subroutine AitkenCalculate(root\_id, x\_hat,error,iteration)
* implicit none
* integer,intent(in) :: root\_id
* real\*8,intent(out) :: x\_hat,error
* integer,intent(out) :: iteration
* real\*8 :: g,g1,g2,g1\_derivative,g2\_derivative,l
* real\*8 :: last\_x,last\_x\_buffer,x\_2
* *!Initialize the x with a random number in different regions*
* *!Initialize the factor l*
* select case(root\_id)
* case (1)
* *!Set the x a random number in (0,1)*
* call random\_number(x\_hat)
* l = g1\_derivative(x\_hat)
* case (2)
* *!Set the x a random number in (1,2)*
* call random\_number(x\_hat)
* x\_hat = x\_hat+1
* l = g2\_derivative(x\_hat)
* case (3)
* *!Set the x a random number in (-2,-1)*
* call random\_number(x\_hat)
* x\_hat = x\_hat\*(-1)-1
* l = g2\_derivative(x\_hat)
* end select
* select case(root\_id)
* case(1)
* last\_x = g1(x\_hat)
* x\_hat = g1(last\_x)
* case default
* last\_x = g2(x\_hat)
* x\_hat = g2(last\_x)
* end select
* iteration = 0
* *!Set a large number*
* error = 10000000
* *!Main loop*
* do while(abs(error)>1d-8 .and. iteration < 100)
* select case(root\_id)
* case(1)
* x\_2 = g1(x\_hat)
* case default
* x\_2 = g2(x\_hat)
* end select
* last\_x\_buffer = x\_hat
* x\_hat = x\_2 - (x\_2-x\_hat)\*\*2/(x\_2-2\*x\_hat+last\_x)
* error = abs(last\_x - x\_hat)
* last\_x = last\_x\_buffer
* iteration = iteration + 1
* end do
* end subroutine AitkenCalculate
* *!Functions definitions*
* function f(x)
* implicit none
* real\*8 :: x
* real\*8 :: f
* f = x\*\*3/3-x
* end function
* function f\_derivative(x)
* implicit none
* real\*8 :: x
* real\*8 :: f\_derivative
* f\_derivative = x\*\*2-1
* end function
* *!g1 = x\*\*3/3*
* function g1(x)
* implicit none
* real\*8 :: x
* real\*8 :: g1
* g1 = x\*\*3/3
* end function
* function g1\_derivative(x)
* implicit none
* real\*8 :: x
* real\*8 :: g1\_derivative
* g1\_derivative = x\*\*2
* end function
* *!g2 = x\*\*(1.0/3)/3*
* function g2(x)
* implicit none
* real\*8 :: x
* real\*8 :: g2
* if ( x>0 ) then
* g2 = (3\*x)\*\*(1.0/3)
* else
* g2 = sign(abs(3\*x)\*\*(1.0/3.0),x)
* end if
* end function
* *!g2 = x\*\*(1.0/3)/3*
* function g2\_derivative(x)
* implicit none
* real\*8 :: x
* real\*8 :: g2\_derivative
* if ( x>0 ) then
* g2\_derivative = (3\*x)\*\*(dble(-2.0)/dble(3.0))
* else
* g2\_derivative = sign(abs(3\*x)\*\*(dble(-2.0)/dble(3.0)),x)
* end if
* end function
* **Results and examples:**





Iteration Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **X1** | **X2** | **X3** |
| Jacobi | 6 | 17 | 18 |
| Newton Downhill | 6 | 6 | 5 |
| Post Acc | 4 | 15 | 27 |
| Aitken Acc | 3 | 9 | 9 |

Execution Time Comparison (unit = us)

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **X1** | **X2** | **X3** |
| Jacobi | 55867us | 110718us | 234540us |
| Newton Downhill | 22932us | 127540us | 176664us |
| Post Acc | 54524us | 89251us | 155515us |
| Aitken Acc | 36630us | 122745us | 235672us |

* **Demo:**

Check the folder “Root” in the directory.

Run the program “a.exe”