Physics 562 - Computational Physics

Midterm 3

Josh Fernandes
Department of Physics & Astronomy
California State University Long Beach

April 22, 2014

Abstract

This paper examines two different questions.

1 Problem 1

2 The Fortran95 code

Numtype is the same for problems 1 and 2.

Listing 1: Module `NumType`

```

module NumType

    save
    integer, parameter ::   dp = kind(1.d0)
    real(dp), parameter ::  pi = 4*atan(1._dp)
    complex(dp), parameter ::   iic = (0._dp,1._dp),&
                                one = (1._dp,0._dp),&
                                zero = (0._dp,0._dp)


end module NumType
```

Listing 2: `newton.f95`

```
!this program solves a set of linear equations
module setup

    use numtype
    use matrix
    implicit none
    integer, parameter  :: nv = 3, np = 3
    real(dp)            :: a(1:np)

end module setup


program newton

    use setup
    implicit none

    real(dp)            :: x(nv), dx(nv), f(nv), jacobi(nv,nv), ff
    real(dp), parameter :: eps = 1.e-10_dp
    integer             :: maxstep, i

    a(1:np) = (/ 0.2_dp, 0.2_dp, -100._dp /) !(a,b,c)
    x(1:nv) = (/ -1._dp, 0._dp, 0._dp /)    !(x,y,z)

    maxstep = 15

    do i = 1, maxstep

        call func(nv,x,f,jacobi,ff)
        print '(3f12.4, 3x, 3e12.4, 3x, e12.4)', &
                x(1:nv), f(1:nv), ff

        if (ff <= eps ) exit

        call dgei(nv,jacobi,nv)

        dx(1:nv) = matmul( jacobi(1:nv,1:nv), -f(1:nv))

```

```
        x(1:nv) = x(1:nv)+dx(1:nv)

    end do


end program newton

subroutine func(n,x,f,jmat,ff)

    use setup
    implicit none
    real(dp) :: x(n), f(n), jmat(n,n), ff
    integer :: n

    f(1) = -(x(2)+x(3))
    f(2) = x(1)+a(1)*x(2)
    f(3) = a(2)+x(3)*(x(1)-a(3))

    ff = sqrt(dot_product(f(1:n),f(1:n)))

    jmat(1,1) = 0
    jmat(2,1) = 1
    jmat(3,1) = x(3)

    jmat(1,2) = -1
    jmat(2,2) = a(1)
    jmat(3,2) = 0

    jmat(1,3) = -1
    jmat(2,3) = 0
    jmat(3,3) = x(1)-a(3)

end subroutine func
```

The main program is `newton` and it begins with its own module. The code is run by typing `./newton`.

3 Problem 2

4 The Fortran95 code

Listing 3: `mtestthisone.f95`

```

module setup

    use NumType
    implicit none
    integer,    parameter   ::  n_basis=10,lwork=2*n_basis+1
    real(dp),   parameter   ::  mass=1.0_dp, hbar=1.0_dp,&
                                omega_h=2._dp,omega_b=1._dp

end module setup

program matrix

    use setup
    implicit none

    complex(dp) ::  x_mat(0:n_basis,0:n_basis+1), p_mat(0:n_basis,0:n_bas
                    x2_mat(0:n_basis,0:n_basis), p2_mat(0:n_basis,0:n_bas
                    h_mat(0:n_basis,0:n_basis),f(0:n_basis,0:n_basis), &
                    work(lwork),c(0:n_basis,0:n_basis),d(0:n_basis,0:n_ba
    integer :: n,m, info, i, j,ipiv(n_basis)
    real(dp) :: rwork(3*(n_basis-2)), w_eigen(n_basis+1)


    x_mat = 0._dp
    p_mat = 0._dp

    do n=0,n_basis-1
        x_mat(n,n+1) = sqrt(hbar/(2*mass*omega_b))*sqrt(n+1._dp)
        x_mat(n+1,n) = x_mat(n,n+1)
    end do
    x_mat(n_basis,n_basis+1) = sqrt(hbar/(2*mass*omega_b))*sqrt(n_basis+1


```

```
    do n=0,n_basis-1
        p_mat(n,n+1) = -iic*sqrt(mass*hbar*omega_b/2)*sqrt(n+1._dp)
        p_mat(n+1,n) = -p_mat(n,n+1)
    end do
    p_mat(n_basis,n_basis+1) = -sqrt(mass*hbar*omega_b/2)*sqrt(n_basis+1.


  ! do n=0,n_basis
!        print '(12f10.5)', x_mat(n,0:n_basis+1)
!    end do
!
!    print *, '----------'
!
!    do n=0,n_basis
!        print '(12f10.5)', p_mat(n,0:n_basis+1)
!    end do


     print *, '=========================================================


    h_mat(0:n_basis,0:n_basis) = 1/(2*mass) * &
        matmul(p_mat(0:n_basis,0:n_basis+1),conjg(transpose(p_mat(0:n_bas
        mass*omega_h**2/2 * &
        matmul(x_mat(0:n_basis,0:n_basis+1),transpose(x_mat(0:n_basis,0:n

    h_mat(0,5) = h_mat(0,5) + (hbar*omega_h/2) !add contribution
    h_mat(5,0) = h_mat(5,0) + (hbar*omega_h/2)

    print *, '---------------hamiltonian␣matrix---------------'
    do n=0,n_basis
        print '(12f10.5)', h_mat(n,0:n_basis)
    end do

    call zheev('n','u',n_basis+1,h_mat,n_basis+1,w_eigen,work,lwork,rwork


    print *, '----------'
    print *, info

```

```
    print *, 'eigenvalue', w_eigen(1:n_basis+1)

    do i = 1,10
        print *,'␣␣vector␣␣', dble(h_mat(0:n_basis,i))
    end do

    !do n=1,n_basis
!         print '(12f10.5)', h_mat(1:n_basis,n)
!     end do


    print *, '-------␣Orthogonality␣-------␣'

    do j = 1,10
            do i = 1,10
                print *, i, j, dot_product(h_mat(1:10,i) ,h_mat(1:10,j) )

            end do
        end do


        print *, '-------␣completeness␣&␣spectral␣decomp-------␣'

        do j = 1,n_basis
            do i = 1,n_basis
                f(i,j)= dot_product(h_mat(i,1:n_basis) ,h_mat(j,1:n_basis
                *w_eigen(1:n_basis))  !determines if matrix is complete
            end do
        end do


        Do i= 1,10
            print '(10f7.2)', f(i,1:10)
        end do


        print *, '++++++++++++++++++++++++++++++++++++'

        c(1:n_basis,1:n_basis) = conjg(transpose(h_mat(1:n_basis,1:n_basi

```

```fortran
        f(1:n_basis,1:n_basis) = matmul(c(1:n_basis,1:n_basis),h_mat(1:n_
        f(1:n_basis,1:n_basis) = matmul(h_mat(1:n_basis,1:n_basis),c(1:n_

        Do i= 1,n_basis
            print '(10f7.2)', f(i,1:n_basis)
        end do

        print *, '+++++++++++++++++++++++++++++++++++++++++++++'

        f(1:n_basis,1:n_basis)=matmul(h_mat(1:n_basis,1:n_basis),h_mat(1:
        d(1:n_basis,1:n_basis) = matmul(c(1:n_basis,1:n_basis),f(1:n_basi

        Do i= 1,n_basis
            print '(10f7.2)', d(i,1:n_basis)
        end do


        print *, '++++++++ inversions of e+++++++++++++ '
        info = 0
        call zgetrf(n_basis,n_basis,h_mat, n_basis, ipiv,info)

        if(info/= 0) stop 'stop'

        call zgetri(n_basis,h_mat,n_basis,ipiv,work, lwork,info)

        if(info/= 0) stop 'stop'

        Do i= 1,n_basis
            print '(6f7.2,5x,6f7.2)', c(i,1:n_basis)-h_mat(i,1:n_basis)
        end do




end program matrix
```

5 Results

Here are the results for the first problem. This is for part a. It would seem that changing the value of c significantly changes the resulting values of x,y, and z. When $a = .2, b = .2, c = 5.7$, then $x = .0070, y = -0.0351, z = 0.0351$. When $a = .2, b = .2, c = 5.6$, then $x = .0072, y = -0.0358, z = 0.0358$. When $a = .2, b = .2, c = 5.8$, then $x = .0069, y = -0.0345, z = 0.0345$. When $a = .2, b = .2, c = 5$, then $x = .0080, y = -0.0401, z = 0.0401$. When $a = .2, b = .2, c = 100$, then $x = .0004, y = -0.02, z = 0.02$. Making C negative just changes the sign of x,y, and z. For part b, there were only two solutions. When $x = ..0070, y = -0.0351, z = 0.0351$ and $x = 5.6930, y = -28.4649, z = 28.4649,$. It didn't matter how far away the initial conditions were, they would always settle to one of the two conditions. In contrast, changing c almost always resulted in a different solution for x,y,z.

for the second problem, I should be getting a matrix with 1's on the diagonal to prove that it is orthogonal. That is not what I got.

References

[1] M. Metcalf, J. Reid and M. Cohen, *Fortran 95/2003 explained.* Oxford University Press, 2004.