UNIVESITY OF ESWATINI
FACULTY OF SCIENCE AND ENGINEERING
DEPARTMENT OF PHYSICS

Examination 2020/2021

COURSE NAME: ADVANCED COMPUTATIONAL PHYSICS

TIME ALLOWED: 3 hours

Answer All Questions in Section A. Choose **two (2)** Questions in section B.

THIS PAPER IS NOT TO BE OPENED UNTIL PERMISSION HAS BEEN GIVEN BY THE INVIGILATOR.

Section A

Answer **all** questions in this section.

Question 1

Evaluate the following expressions by writing a Fortran program for each, printing out the answers on the screen.

(a) $y = s^2 + r^3$, where $s = \cos x$, $r = e^x$ and $x = 0.5$.

[3 marks]

(b) $y = \tanh(f(z))$, where $f(z) = \log\sqrt{\dfrac{1+z}{1-z}}$, and $z = \dfrac{1}{2}$

[4 marks]

(c) $y = \cos^2(\theta) - \sin^2(\theta)$, where $\theta = \dfrac{\pi}{2}$.

[3 marks]

(d) $y = \sin^{-1}(x) + \cos^{-1}(x)$, for any value of x.

[5 marks]

Question 2

Give the output of the following programs. (Or mark NO OUTPUT if the program won't compile or doesn't output anything.) You don't need to include leading or trailing blanks. For `real` values, assume outputs have 4 digits to the right of the decimal point by default.

(a)
```
PROGRAM meloveprog
    IMPLICIT NONE
   write(*,*) "I love programming!"
END PROGRAM meloveprog
```

[2 marks]

(b)
```
PROGRAM mevarassn
    IMPLICIT NONE
    INTEGER :: woop
    woop = 10
    write(*,*) 'woop = ', woop
END PROGRAM mevarassn
```

[3 marks]

(c)
```
PROGRAM meif
    IMPLICIT NONE
    INTEGER :: input1, input2
    read(*,*) input1, input2
    write(*,*) "Lookie!"
    IF (input1 < input2) THEN
      write(*,*) "The first is less!"
    ELSE IF (input1 > input2) THEN
      write(*,*) "The first is greater!"
    ELSE
      write(*,*) "Huh?"
    END IF
     write(*,*) "Get lost!"
END PROGRAM meif
```

with the following input;

(i) $\{2, 3\}$

[3 marks]

(ii) $\{22, 22\}$

[3 marks]

(d)
```
PROGRAM medonest
   IMPLICIT NONE
   INTEGER :: i, j, aggregate = 0
   DO j = 1, 2
     DO i = 1, 3
        aggregate = aggregate + (j * i)
     END DO
   END DO
   write(*,*) aggregate
END PROGRAM medonest
```

[4 marks]

(e)
```
PROGRAM meexpr5
   IMPLICIT NONE
   LOGICAL :: p = .TRUE., q = .FALSE., r
   r = (p .AND. q) .OR. (p .EQV. q)
   write(*,*) r
END PROGRAM meexpr5
```

[5 marks]

Question 3

(a) The simple program below is written to read in the radius and calculate the area of the corresponding circle and volume of the sphere.

```
1  !  ************************************************
2  program  Cirle
3  !  ************************************************
4  implicit  none
5  real*8  PI,r,A,V
6
7    PI=4.0d0*atan(1.0d0)
8    write(*,*)  'please  enter  the  value  of  thr  radius,r'
9    read(*,*)  r
10
11    A=PI*r**2
12    V=4*PI*r**3
13   write(*,*)  'The  area  is,', A
14   write(*,*)  'The  Volume  is,', V
15 end
```

(i) Modify this program such that it stops with a warning if the radius given is negative.

[4 marks]

(ii) Modify the program such that it computes a complex value of the volume and area, given a complex value of the radius.

[4 marks]

(b) Modify the code in (a) such that it reads $10cm \leq r \leq 20cm$, in steps of $0.2cm$ from a file and writes out the area and a volume in another file.

[7 marks]

Section B

Choose two (2) questions in this section

Question 4

A classical numerical problem is the summation seris to evaluate a function. Consider the infinite series for $\sin x$;

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots \approx \sum_{n=1}^{N} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}$$

For $0 \leq x \leq 2\pi$ radians(**choose 30 values in this interval with equal spacing**) compute the $\sin x$ using the above approximation. Choose your N such that $\left|\frac{n^{th} \text{ term}}{\text{sum}}\right| \leq 10^{-15}$. Compare your results with the "exact" value obtained from the intrisic function '$\sin x$' in a table, also showing x in `radians` and `degrees`. Plot your results in a well labelled graph.

[25 marks]

Question 5

The temperature distribution $u(x,t)$ in a thin rod satisfies equation

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}$$

with boundary conditions

$$u(0,t) = u(1,t) = 0.$$

The initial temperature distribution at $t = 0$ is given by the function

$$u(x,0) = \begin{cases} 0.5 \text{ if } x \in [x_1, x_2] \\ 0.3 \text{ if } x \notin [x_1, x_2] \end{cases}$$

where $x_1 = 0.25$ and $x_2 = 0.75$. Calculate the temperature distribution $u(x,t_f)$ for $t_f = 0.0001, 0.001, 0.01, 0.05$. Take $N_x = 100$ and $N_t = 1000$.

[25 marks]

Question 6

In the appendix the program `LaplaceEq.f90` computes the electrostatic potential around conductors. The computation is performed on a square lattice of linear dimension L. A relaxation method is used to converge to the solution of Laplace equation for the potential. Construct the corresponding plots of $V(i,j)$ for;

(a) $L = 31$, $V_1 = 100$, $V_2 = 100$

[12 marks]

(b) $L = 20$, $V_1 = 100$, $V_2 = 0$

[13 marks]

Appendix

```
1  program laplace_em
2   implicit none
3  !P defines the size of the arrays and is equal to L
4   integer,parameter          :: P=31
5   logical,dimension(P,P)     :: isConductor
6   real(8),dimension(P,P)     :: V
7  !V1 and V2 are the values of the potential on the interior
8  !conductors. epsilon is the accuracy desired for the convergence
9  !of the relaxation method in subroutine laplace()
10  real(8)                    :: V1,V2,epsilon
11  integer                    :: L
12
13 !We ask the user to provide the necessary data: V1,V2 and epsilon
14  L = P
15  write(*,*) 'Enter V1,V2:'
16  read(*,*) V1,V2
17  write(*,*)'Enter epsilon:'
18  read(*,*) epsilon
19  write(*,*) 'Starting Laplace:'
20  write(*,*) 'Grid Size= ',L
21  write(*,*) 'Conductors set at V1= ',V1,' V2= ',V2
22  write(*,*) 'Relaxing with accuracy epsilon= ',epsilon
23 !The arrays V and isConductor are initialized
24  call initialize_lattice(V,isConductor,L,V1,V2)
25 !We enter initialized V,isConductor. On exit the
26 !routine gives the solution V
27  call laplace(V,isConductor,L,epsilon)
28 !We print V in a file.
29  call print_results(V,L)
30
31 end program laplace_em
32 !*************************************************************
33 !subroutine initialize_lattice
34 !Initializes arrays V(L,L) and isConductor(L,L).
35 !V(L,L)= 0.0  and isConductor(L,L)= .FALSE. by default
36 !isConductor(i,j)= .TRUE. on boundary of lattice where V=0
37 !isConductor(i,j)= .TRUE. on sites with i= L/3+1, 5<= j <= L-5
38 !isConductor(i,j)= .TRUE. on sites with i=2*L/3+1, 5<= j <= L-5
39 !V(i,j) = V1  on all sites with i= L/3+1, 5<= j <= L-5
40 !V(i,j) = V2  on all sites with i=2*L/3+1, 5<= j <= L-5
41 !V(i,j) = 0   on boundary (i=1,L and j=1,L)
42 !V(i,j) = 0   on interior sites with isConductor(i,j)= .FALSE.
43 !INPUT:
44 !integer L: Linear size of lattice
45 !real(8)  V1,V2: Values of potential on interior conductors
46 !OUTPUT:
47 !real(8)  V(L,L): Array provided by user. Values of potential
48 !logical isConductor(L,L): If .TRUE.  site has fixed potential
49 !                          If .FALSE. site is empty space
50 !*************************************************************
51 subroutine initialize_lattice(V,isConductor,L,V1,V2)
52  implicit none
53  integer                    :: L
54  logical,dimension(L,L)     :: isConductor
55  real(8),dimension(L,L)     :: V
56  real(8)                    :: V1,V2
```

```
57  integer                  :: i,j
58
59 !Initialize to 0 and .FALSE (default values for boundary and
60 !interior sites).
61  V           =  0.0D0
62  isConductor = .FALSE.
63 !We set the boundary to be a conductor: (V=0 by default)
64  do i=1,L
65   isConductor(1,i) = .TRUE.
66   isConductor(i,1) = .TRUE.
67   isConductor(L,i) = .TRUE.
68   isConductor(i,L) = .TRUE.
69  enddo
70 !We set two conductors at given potential V1 and V2
71  do i=5,L-5
72   V           (  L/3+1,i) =  V1
73   isConductor(  L/3+1,i) = .TRUE.
74   V           (2*L/3+1,i) =  V2
75   isConductor(2*L/3+1,i) = .TRUE.
76  enddo
77
78 end subroutine initialize_lattice
79 !*****************************************************************
80 !subroutine laplace
81 !Uses a relaxation method to compute the solution of the Laplace
82 !equation for the electrostatic potential on a 2 dimensional
83 !squarelattice of linear size L.
84 !At every sweep of the lattice we compute the average Vav of the
85 !potential at each site (i,j) and we immediately update V(i,j)
86 !The computation continues until Max |Vav-V(i,j)| < epsilon
87 !INPUT:
88 !integer L: Linear size of lattice
89 !real(8) V(L,L): Value of the potential at each site
90 !logical isConductor(L,L): If .TRUE.  potential is fixed
91 !                          If .FALSE. potential is updated
92 !real(8) epsilon: if Max |Vav-V(i,j)| < epsilon return to
93 !callingprogram.
94 !OUTPUT:
95 !real(8) V(L,L): The computed solution for the potential
96 !*****************************************************************
97 subroutine laplace(V,isConductor,L,epsilon)
98  implicit none
99  integer :: L
100  logical,dimension(L,L)  :: isConductor
101  real(8),dimension(L,L)  :: V
102  real(8)                 :: epsilon
103  integer                 :: i,j,icount
104  real(8)                 :: Vav,error,dV
105
106  icount = 0                 !counts number of sweeps
107  do while (.TRUE.)          !an infinite loop:
108   error = 0.0D0             !Exit when error<epsilon
109   do  j=2,L-1
110    do i=2,L-1
111 !We change V only for non conductors:
112     if( .NOT. isConductor(i,j))then
113      Vav = ( V(i-1,j)+V(i+1,j)+V(i,j+1)+V(i,j-1)) * 0.25D0
114      dV  = DABS(V(i,j)-Vav)
```

```
115       if(error .LT. dV ) error = dV !maximum error
116       V(i,j) = Vav            ! we immendiately update V(i,j)
117      endif
118     enddo
119    enddo
120    icount = icount + 1
121    write(*,*) icount,' err= ',error
122    if( error .LT. epsilon) return !return to main program
123   enddo
124
125 end subroutine laplace
126 !*****************************************************************
127 !subroutine  print_results
128 !Prints the array V(L,L) in file "data"
129 !The format of the output is appropriate for the splot function
130 !of gnuplot: Each time i changes an empty line is printed.
131 !INPUT:
132 !integer L: size of array V
133 !real(8) V(L,L): array to be printed
134 !OUTPUT:
135 !no output
136 !*****************************************************************
137 subroutine  print_results(V,L)
138  implicit none
139  integer                   :: L
140  real(8),dimension(L,L)  :: V
141  integer                   :: i,j
142
143  open(unit=11,file="data")
144  do i=1,L
145   do j =1,L
146    write(11,*)i,j,V(i,j)
147   enddo
148   write (11,*)'' !print empty line for gnuplot, separate isolines
149  enddo
150
151 end subroutine print_results
```