**CHEM 555-75** **WORKSHEET 5** **Fall 2021**

Instructor: Prof. Lee M. Thompson E-mail: *lee.thompson.1@louisville.edu* Office: CB 251

1. For a particle in a $L \times 2L$ box, determine the energy of states $|n_x = 2, n_y = 8\rangle$ and $|4, 4\rangle$. What can you say about the energy of these two states?

$E_{n_x n_y} = \frac{n_x^2 h^2}{8mL_x^2} + \frac{n_y^2 h^2}{8mL_y^2} = \frac{h^2}{32mL^2}(4n_x^2 + n_y^2)$
$E_{2,8} = \frac{h^2}{32mL^2}(4 \times 2^2 + 8^2) = \frac{5h^2}{2mL^2}$
$E_{4,4} = \frac{h^2}{32mL^2}(4 \times 4^2 + 4^2) = \frac{5h^2}{8mL^2}$
As $E_{2,8} = E_{4,4}$ and $L_x \neq L_y$, $|2, 8\rangle$ and $|4, 4\rangle$ are accidentally degenerate.

2. Write the kinetic energy matrix element for particle in a box wavefunctions. Use the fact that the particle in a box operators are orthonormal (i.e. $\langle \Psi_a | \Psi_b \rangle = \delta_{ab}$).

$\langle \Psi_a | \hat{K} | \Psi_b \rangle = \int_0^L dx \sqrt{\frac{2}{L}} \sin\left(\frac{a\pi}{L}x\right) \frac{-\hbar^2}{2m} \frac{d^2}{dx^2} \sqrt{\frac{2}{L}} \sin\left(\frac{b\pi}{L}x\right)$
$\langle \Psi_a | \hat{K} | \Psi_b \rangle = \frac{-\hbar^2}{2m} \int_0^L dx \sqrt{\frac{2}{L}} \sin\left(\frac{a\pi}{L}x\right) \frac{d^2}{dx^2} \sqrt{\frac{2}{L}} \sin\left(\frac{b\pi}{L}x\right)$
$\langle \Psi_a | \hat{K} | \Psi_b \rangle = \frac{\hbar^2 b^2 \pi^2}{2mL^2} \int_0^L dx \sqrt{\frac{2}{L}} \sin\left(\frac{a\pi}{L}x\right) \sqrt{\frac{2}{L}} \sin\left(\frac{b\pi}{L}x\right)$
$\langle \Psi_a | \hat{K} | \Psi_b \rangle = \frac{\hbar^2 b^2 \pi^2}{2mL^2} \langle \Psi_a | \Psi_b \rangle$
$\langle \Psi_a | \hat{K} | \Psi_b \rangle = \frac{\hbar^2 a^2 \pi^2}{2mL^2} \delta_{ab}$

3. The wavefunction $\psi_0(x) = N_0 \exp\{-ax^2\}$ is a solution to the Schrödinger equation for the harmonic oscillator with parabolic potential energy function $V(x) = \frac{1}{2}kx^2$ providing that a suitable value of a is chosen to satisfy the boundary conditions. By acting on the wavefunction with the Hamiltonian and ensuring the wavefunction obeys the boundary conditions, write an expression for $a$ as a function of the force constant $k$ and the mass $m$ and determine the expression for the energy.

$\hat{H}\psi_0(x) = E_0\psi_0(x) \Rightarrow (\hat{K} + \hat{V})\psi_0(x) = E_0\psi_0(x) \Rightarrow -\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi_0(x) + \frac{1}{2}kx^2\psi_0(x) = E_0\psi_0(x)$
$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}(N_0 \exp\{-ax^2\}) + \frac{1}{2}kx^2(N_0 \exp\{-ax^2\}) = E_0(N_0 \exp\{-ax^2\})$
$-\frac{\hbar^2}{2m}N_0 2a \exp\{-ax^2\}(2ax^2 - 1) + \frac{1}{2}kx^2(N_0 \exp\{-ax^2\}) = E_0(N_0 \exp\{-ax^2\})$
$(\frac{a\hbar^2}{m} + x^2(-\frac{2a^2\hbar^2}{m} + \frac{1}{2}k))N_0 \exp\{-ax^2\} = E_0(N_0 \exp\{-ax^2\})$
$E_0 = \frac{a\hbar^2}{m} + x^2(-\frac{2a^2\hbar^2}{m} + \frac{1}{2}k)$ but energy cannot be a function of $x$ so:
$-\frac{2a^2\hbar^2}{m} + \frac{1}{2}k = 0$
$\frac{2a^2\hbar^2}{m} = \frac{1}{2}k$
$a = \sqrt{\frac{mk}{4\hbar^2}}$
$E_0 = \frac{a\hbar^2}{m}$

4. The wavefunction for a 1s orbital in the hydrogen atom using polar coordinates is $\psi_{1s}(d, \theta, \phi) = N_{1s}e^{-\beta r}$ where $N_{1s}$ is a normalization constant which may be ignored. Using the Hamiltonian for the hydrogen atom in polar coordinates, find the quantity $\beta$ for which the 1s orbital is a solution of the Schrödinger equation and calculate its energy.

$\hat{H} = -\frac{\hbar^2}{2\mu}\left(\frac{1}{r}\frac{\partial^2}{\partial r^2}r + \frac{1}{r^2}\Lambda^2 + \frac{Ze^2}{4\pi\varepsilon_0 r}\right)$

$\Lambda^2 = \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2} + \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\sin\theta\frac{\partial}{\partial\theta}$

$\hat{H} = -\frac{\hbar^2}{2\mu}\left(\frac{1}{r}\frac{\partial^2}{\partial r^2}r + \frac{1}{r^2}\Lambda^2 + \frac{Ze^2}{4\pi\varepsilon_0 r}\right)\psi_{1s} = E\psi_{1s}$

$\frac{1}{r}\frac{\partial^2}{\partial r^2}re^{-\beta r} = \frac{1}{r}\frac{\partial}{\partial r}\left(e^{-\beta r - \beta r e^{-\beta r}}\right) = \left(\frac{-2}{r} + \beta\right)\beta e^{-\beta r}$

Wavefunction has no dependence on $\theta$ or $\phi$, so remaining terms must be zero.

$\left\{-\frac{\hbar^2}{2\mu}\left(\frac{-2}{r} + \beta\right)\beta - \frac{Ze^2}{4\pi\varepsilon_0 r}\right\}e^{-\beta r} = Ee^{-\beta r}$

$E = -\frac{\hbar^2}{2\mu}\left(\frac{-2}{r} + \beta\right)\beta - \frac{Ze^2}{4\pi\varepsilon_0 r}$

Energy cannot be a function of r so terms that have r dependence must be zero.

$E = -\frac{\hbar^2\beta^2}{2\mu}$

$-\frac{\hbar^2}{2\mu}\frac{-2\beta}{r} - \frac{Ze^2}{4\pi\varepsilon_0 r} = 0$

$\frac{\hbar^2\beta}{\mu r} = \frac{Ze^2}{4\pi\varepsilon_0 r}$

$\beta = \frac{Ze^2\mu}{4\pi\varepsilon_0\hbar^2}$

$E = \frac{Z^2e^4\mu}{32\pi^2\varepsilon_0^2\hbar^2}$

5. In the last worksheet, we went over how to use some features of the MQCPack library. This library is an extension to standard Fortran. This week we are going to learn some basic code structures and notation in Fortran. As you should have become familiar with using vim, I am not going to explicitly write all the commands here.

Open a new file with the extension <file-name>.f03 and write the program name at the top followed by the line "implicit none". On the next line, add "integer::i,max_i=5". We will now add a 'do' loop (frequently referred to as a 'for' loop in other languages) which allows us to repeatedly execute code whilst incrementing a variable. Add the following code to your file:

```
1 do i=1,max_i
2   write(*,*) 'i equals', i
3 endDo
```

Don't forget to add the end program line after. To compile, type 'gfortran -o <file-name><file-name>.f03'. You need to make sure that you have loaded the gcc 9.2.0 module like you did last week. Once you have compiled, if you type 'ls' you should be able to see a file named <file-name>which will possibly be a different font to the other files because it is an executable file. To run type './<file-name>' on the command line and press enter. You should see the output listing the numbers from 1 to 5. What happens if you have two variables that you would like to iterate over? Then you can have a nested loop. Modify your code as follows:

```
1 do i=1,max_i
2   do j=max_j,1,-1
```

```
3      write(*,*) 'i,j equals', i,',',j
4    endDo
5  endDo
```

Don't forget to include j and max_j=5 in the variable typing section. Recompile and run this code and you should see that for every i, j counts down from 5 to 1, showing how the third variable in the do loop notation provides an increment in the loop (e.g. try changing the value to -2 and you will see it only counts odd numbers). You can also make the limits dependent on the other variable, for example:

```
1 do i=1,max_i
2    do j=1,i-1
3      write(*,*) 'i,j equals', i,',',j
4    endDo
5 endDo
```

will compute all the pairs of i and j such that j<i.

A 'while' loop does not require any variables to be set, but continues executing until some condition is met. The equivalent functionality of a 'do' loop can be obtained with the following code:

```
1 i=1
2 do while (i.le.5)
3    write(*,*) 'i equals', i
4    i = i +1
5 endDo
```

A typical use of a while loop is to perform some action, test the result of the action, and exit the loop if the action has reached a criteria. To test the criteria requires the use of an 'if' block (which can be used anywhere, not just inside a 'while' loop):

```
1  i=1
2  do while (.true.)
3     write(*,*) 'i equals', i
4     if(i.lt.5) then
5        i = i +1
6        cycle
7     else
8        exit
9     endIf
10    write(*,*) 'cycle restarts loop so this line not executed. Remove cycle so this text will print'
11 endDo
12 write(*,*) 'exited loop, i equals',i
```

An 'if' block can have many conditions using the 'elseIf' statement. Add the line 'character(len=10)::fruit' to the variable typing section and remove the previous code (except for the implicit none and begin and end program statements. Then add the following code:

```
fruit ='orange'
if ( fruit .eq.'pear') then
  write(*,*) ' fruit is pear'
elseIf ( fruit .eq.'apple') then
  write(*,*) ' fruit is apple'
elseIf ( fruit .eq.'plum') then
  write(*,*) ' fruit is plum'
elseIf ( fruit .eq.'orange') then
  write(*,*) ' fruit is orange'
else
  write(*,*) 'We have a mystery fruit'
endIf
```

Compiling and running the code will provide the answer 'fruit is orange'. Now change fruit to 'guava' and you will see that the code is unable to identify it as guava, returning 'We have a mystery fruit'. Rather than recompiling the code every time you change the fruit, you can read in a fruit from the terminal. Modify the code as follows, making sure to add the character type declaration in the preamble:

```
character(len=256):: fruit

write(*,*) 'Please enter fruit name'
read(5,*) fruit
if ( fruit .eq.'pear') then
  write(*,*) ' fruit is pear'
elseIf ( fruit .eq.'apple') then
  write(*,*) ' fruit is apple'
elseIf ( fruit .eq.'plum') then
  write(*,*) ' fruit is plum'
elseIf ( fruit .eq.'orange') then
  write(*,*) ' fruit is orange'
else
  write(*,*) 'We have a mystery fruit'
endIf
```

Alternatively, it may be preferred to input your fruit at the same time you run the calculation, in which case you can modify the code as follows:

```
character(len=256):: fruit
integer :: nargs

nargs = command_argument_count()
if (nargs.ne.1) then
  write(*,*) 'A single fruit must be specified with the command'
  stop
endIf
call get_command_argument(1,fruit)

if ( fruit .eq.'pear') then
  write(*,*) ' fruit is pear'
elseIf ( fruit .eq.'apple') then
  write(*,*) ' fruit is apple'
elseIf ( fruit .eq.'plum') then
  write(*,*) ' fruit is plum'
elseIf ( fruit .eq.'orange') then
  write(*,*) ' fruit is orange'
else
  write(*,*) 'We have a mystery fruit'
endIf
```

Compile and run this code specifying zero, one and two fruits. You can see the code only works when a single fruit is specified.