Helium Atom

Charalampos Pozoukidis & Dimitris Pozoukidis
✉ cpozouki@auth.gr & dpozouki@auth.gr

April 13, 2021

Contents

1 Introduction

The goal of this paper is to calculate the ground state energy of the Helium Atom as accurately as possible. The SCHRODINGER equation of the Helium atom can't be solved analytically, therefore approximation methods have to be used in order to solve this problem. The term in the HAMILTON operator which does not allow us to solve the problem analytically is the one which describes the repulsion of the two electrons and it's also this term which won't allow us to use perturbation theory to calculate the ground state energy *. The theorem of HYLLERAAS and UNDHEIM is therefore the most effective approach to solve this problem. HYLLERAAS used the theorem in 1930 to calculate the ground state energy of the helium atom, in this paper we will see how with the help of modern computers we can make even more accurate calculations than he did back then.

* For more information on why perturbation theory fails to give accurate results see David Tong's lecture notes on Approximations in Quantum Mechanics .

2 Basic Theory

In the chapters below we are going to present the basic theorems and results that are needed to calculate the ground state energy of the Helium atom. The HYLLERAAS and UNDHEIM theorem is a generalization of the Variational method in quantum mechanics and we will start with that. Later in this chapter we are gonna present the ground state wave function of the Hydrogen atom without deriving it and explain how it will help us calculate the ground state energy of the Helium atom.

2.1 The Variational Method

Theorem: Consider an arbitrary state $|\psi\rangle$. The expected value of the energy obeys the inequality

$$\langle E\rangle = \langle\psi|H|\psi\rangle \geq E_0$$

Proof: We expand $|\psi\rangle = \sum_n a_n|n\rangle$ with $\sum_n |a_n|^2 = 1$ to ensure that $\langle\psi \mid \psi\rangle = 1$. Then

$$\begin{aligned}\langle E\rangle &= \sum_{n,m=0}^{\infty} a_m^\star a_n \langle m|H|n\rangle = \sum_{n,m=0}^{\infty} a_m^\star a_n E_n \delta_{mn} \\ &= \sum_{n=0}^{\infty} |a_n|^2 E_n = E_0 \sum_{n=0}^{\infty} |a_n|^2 + \sum_{n=0}^{\infty} |a_n|^2 (E_n - E_0) \geq E_0\end{aligned}$$

Now consider a family of states, $|\psi(\alpha)\rangle$, depending on some number of parameters α_i. If we like, we can relax our assumption that the states are normalised and define

$$E(\alpha) = \frac{\langle\psi(\alpha)|H|\psi(\alpha)\rangle}{\langle\psi(\alpha) \mid \psi(\alpha)\rangle}$$

We still have

$$E(\alpha) \geq E_0 \quad \text{for all } \alpha$$

The most stringent bound on the ground state energy comes from the minimum value of $E(\alpha)$ over the range of α. This, of course, obeys

$$\left.\frac{\partial E}{\partial \alpha_i}\right|_{\alpha=\alpha^*} = 0$$

giving us the upper bound $E_0 \leq E(\alpha_\star)$. If these arbitrary states (or trial wave-functions) are chosen in a physically motivated way this upper bound may be very close to the exact energy of the actual state.

2.2 The theorem of HYLLERAAS and UNDHEIM

As mentioned before this theorem is a generalization of the Variational method. In a basis of dimension D the SCHRODINGER equation in matrix form is:

$$\sum_{n'=1}^{D} H_{nn'} a_{n'} = E a_n,\ n = 1, ..., D$$

Where the HAMILTON operator is a function of α just like in the Variational method. The eigenvalues of this matrix equation are now functions of both the variational parameter α and the Dimension of the Basis we used D. The theorem states that for any D, α:

$$E_0 \leq E(\alpha, D)$$

We can see that for $D = 1$ the HYLLERAAS and UNDHEIM theorem coincides with the variational method. Our purpose is to show that for higher number of dimensions D the energy eigenvalue converges to a single value, which is going to be our projected exact energy of the ground state.

2.3 Hydrogen ground state wave function

The Hydrogen atom is one of those problem that can be solved analytically. The HAMILTON operator for the Hydrogen is:

$$H = -\frac{\hbar^2}{2m}\Delta - \frac{Ze^2}{|\vec{r}|}$$

And the wave function which describes its ground state

$$\phi(\vec{r}) = \exp\left(\frac{-Z}{r_0} r\right)$$

We can see that this HAMILTON operator bears a resemblance to the Helium one, which is:

$$H = -\frac{\hbar^2}{2m}\Delta_1 - \frac{Ze^2}{|\vec{r_1}|} - \frac{\hbar^2}{2m}\Delta_2 - \frac{Ze^2}{|\vec{r_2}|} + \frac{e^2}{|\vec{r_1} - \vec{r_2}|} \tag{1}$$

The reason the Helium atom can't be solved analytically is the term which describes the repulsion between the two electrons. If not for this term the wave function would be a copy of the hydrogen wave function, one for each electron. For this reason one may propose to use their product as trial wave functions:

$$\Phi(\vec{r_1}, \vec{r_2}) = \phi(\vec{r_1})\phi(\vec{r_2}) = \exp\left(\frac{-Z}{\alpha r_0}(r_1 + r_2)\right)$$

Where the parameter α is added so the variational method may be used, r_0 is the BOHR radius and we use r instead of $\vec{r}$ for brevity. While this trial function is a good start it still needs a few corrections to capture the physics of the problem. The electrons push each other away, therefore we must add a term to the trial wave function that's going to reflect that.

$$\Phi_m(\vec{r_1}, \vec{r_2}) = |\vec{r_1} - \vec{r_2}|^m \text{exp}\left(\frac{-Z}{\alpha r_0}(r_1 + r_2)\right), \;\; m \geq 0$$

This is the basis HYLLERAAS used to compute the energy. To make the basis more complete we will add a few extra terms:

$$\tilde{\Phi}_{jkm}(\vec{r_1}, \vec{r_2}) = (r_1 + r_2)^j (r_1 - r_2)^k |\vec{r_1} - \vec{r_2}|^m \text{exp}\left(\frac{-Z}{\alpha r_0}(r_1 + r_2)\right), \;\; j, k, m \geq 0 \quad (2)$$

Where k has to be even for the PAULI principle to hold (the spin of state of the electrons are considered to be in an anti-symmetric state). These functions are not orthonormal and are therefore denoted by a tilde symbol. In the next section we will see how to construct a matrix equation using these basis wave functions.

2.4 The matrix SCHRODINGER equation

The SCHRODINGER equation in general operator form

$$H|\Psi\rangle = E|\Psi\rangle, \;\; H = -\frac{\hbar^2}{2m}\Delta + V(\vec{r})$$

we use an expansion in the discrete basis

$$\sum_{j,k,m} H\tilde{a}_{jkm}|\tilde{\Phi}_{jkm}\rangle = E\sum_{j,k,m} \tilde{a}_{jkm}|\tilde{\Phi}_{jkm}\rangle$$

For brevity we will refer to n as the set j, k, m

$$\sum_{n,n'} |\tilde{\Phi}_{n'}\rangle\langle\tilde{\Phi}_{n'}|H\tilde{a}_n|\tilde{\Phi}_n\rangle = E\sum_{n,n'} |\tilde{\Phi}_{n'}\rangle\langle\tilde{\Phi}_{n'}|\tilde{a}_n|\tilde{\Phi}_n\rangle$$

Which we are going to re-write as:

$$\sum_{n,n'}(\tilde{H}_{n'n} - E\tilde{N}_{n'n})\tilde{a}_n = 0$$

With:

$$\tilde{H}_{n'n} = \langle\tilde{\Phi}_{n'}|H|\tilde{\Phi}_n\rangle \quad (3)$$

and:

$$\tilde{N}_{n'n} = \langle\tilde{\Phi}_{n'}|\tilde{\Phi}_n\rangle \quad (4)$$

This is a generalized eigenvalue problem. To use the HYLLERAAS and UNDHEIM theorem we first need to re-write this in the usual eigenvalue problem. Solving this problem boils down to the orthonormalization of our basis wave functions and finding the desired eigenvalue which is the same in both the regular and the generalized eigenvalue problem. Luckily for us our program can do it automatically and solve immediately for the eigenvalue.

3 Computational Part

Now that we have developed the basic theory we move forward to the computational part which is going to provide us our desired result.

3.1 Calculations of Matrix Elements

To solve the actual eigenvalue problem we have to calculate the matrix elements described in section 2.4 specifically the ones in equation (3) and (4). While these integrals can be solved analytically the calculation is quite tiresome, therefore we will leave the calculations and only present the final results:

$$\begin{aligned}\tilde{N}_{nn'} =& 2\pi^2(J+K+M+5)!\left(\frac{1}{M+2}\right)\\ &\cdot\left(\frac{1}{K+1}-\frac{1}{K+3}-\frac{1}{K+M+3}+\frac{1}{K+M+3}\right)\left(\frac{1}{2\lambda}\right)^{J+K+M+6}\end{aligned} \tag{5}$$

Where we used $J = j + j', M = m + m', K = k + k'$ and $\lambda = Z/\alpha r_0$. These abbreviations will be used for the following matrix elements also. The HAMILTON operator will be divided into three different parts to avoid mistakes while typing the program and to make the calculations easier.

$$H = T + C + W \tag{6}$$

Where:

$$C = -\frac{Ze^2}{r_1} - \frac{Ze^2}{r_2},\ \ T = -\frac{\hbar}{2m}(\Delta_1 + \Delta_2),\ \ W = \frac{e^2}{|\vec{r_1} - \vec{r_2}|} \tag{7}$$

Now we present the results of the matrix elements for each of the operators. For the COULOMB potential between the nucleus and the electrons.

$$\tilde{C}_{nn'} = -Ze^2\hat{C}_{nn'} \tag{8}$$

Where:

$$\hat{C}_{nn'} = 8\pi^2(J+K+M+4)!\left(\frac{1}{M+2}\right)\cdot\left(\frac{1}{K+1}-\frac{1}{K+M+3}\right)\left(\frac{1}{2\lambda}\right)^{J+K+M+5} \tag{9}$$

For the COULOMB potential between the two electrons.

$$\tilde{W}_{nn'} = e^2\hat{W}_{nn'} \tag{10}$$

Where:

$$\hat{W}_{nn'} = N_{JK,M-1} \tag{11}$$

Lastly the matrix elements for the kinetic energy

$$\tilde{T}_{nn'} = \frac{\hbar^2}{2m}\hat{T}_{nn'} \tag{12}$$

Where:

$$\hat{T}_{nn'} = T^1_{nn'} + T^2_{nn'} \tag{13}$$

With:

$$T^1_{nn'} = 2\left(\lambda^2 N_{JKM} - J\lambda N_{J-1,KM} + jj' N_{J-2,KM} + kk' N_{JK,M-2}\right) \tag{14}$$

and:

$$T^2_{nn'} = \frac{1}{2}\big[-M\lambda(\hat{C}_{JKM} - \hat{C}_{J,K+2,M-2}) + (mj' + m'j)(\hat{C}_{J-1,KM} - \hat{C}_{J-1,K+2,M-2}) + (mj' + m'j)(\hat{C}_{J-1,KM} - \hat{C}_{J-1,K+2,M-2})\big] \tag{15}$$

The reason the T operator is divided into two operators is the following: T^2 has parts which are not defined for $m = m' = M = 0$. This may seem to be a problem at first but it's not. The reason is that the multiplying factor, before the undefined operators $N_{JK,-2}, \hat{C}_{J,K+2,-2}, \hat{C}_{J-1,K+2,-2}$ and $\hat{C}_{J-1,K+2,2}$, is zero. Therefore we can still use them without a problem. This means that while "filling" the matrix T with its elements we first must check for the necessary conditions for m, m' and M. Below we present the code which is responsible for defining these Matrix elements :

```
def func_n(J,K,M):
   return 2*3.14**2*math.factorial(J+K+M+5)*(1/(M+2))\
   *(1/(K+1)-1/(K+3)-1/(K+M+3)+1/(K+M+5))*(1/(2*cons.p))**(J+K+M+6)

def func_c(J,K,M):
   return
       8*3.14**2*math.factorial(J+K+M+4)*(1/(M+2))*(1/(K+1)-1/(K+M+3))\
   *(1/(2*cons.p))**(J+K+M+5)

def func_w(J,K,M):
   return myFn.func_n(J,K,M-1)

def func_t1(J,K,M,j_1,k_1,m_1,j_2,k_2,m_2):
   return
       2*(cons.p**2*myFn.func_n(J,K,M)-J*cons.p*myFn.func_n(J-1,K,M)\
   +j_1*j_2*myFn.func_n(J-2,K,M)+k_1*k_2*myFn.func_n(J,K-2,M))
```

```
def func_t2(J,K,M,j_1,k_1,m_1,j_2,k_2,m_2):
  return
     2*m_1*m_2*myFn.func_n(J,K,M-2)+0.5*(-M*cons.p*(myFn.func_c(J,K,M)-\
  myFn.func_c(J,K+2,M-2))+(m_1*j_2+m_2*j_1)*(myFn.func_c(J-1,K,M)-\
  myFn.func_c(J-1,K+2,M-2))+(m_1*k_2+m_2*k_1)\
  *(myFn.func_c(J+1,K,M-2)-myFn.func_c(J-1,K,M)))
```

```
'''
Each function defined above serves the purpose of calculating a
    single matrix element, for a specific set of J K and W.

The elements of the N,C and W matrices only depend on the values of
    J,K and W, whereas the matrices t1 and t2 also need j_i,k_i and
    m_i specified. Each function then proceeds by executing basic
    calculations with the help of the "Math" package responsible of
    calculating the factorials needed.
'''
```

3.2 Construction of Basis wave functions

Now we have to choose the dimension of our basis, which is also going to determine the dimensions of the matrices presented in the section 3.1. To generate a set of basis wave functions we simply must choose a few set of integers (j, k, m). HYLLERAAS in his historical calculation used three basis wave-functions which corresponds to the set $(j, k, m) = (0, 0, 0), (0, 0, 1)$ and $(0, 0, 2)$ which stands for a three dimensional basis. In this paper the dimension of the basis will be the number of different ways for which the inequality below holds:

$$j + k + m < n \tag{16}$$

Where n is an integer. For example if $n = 3$, the allowed combinations of (j, k, m) would be:

$$(0, 0, 0), (0, 0, 1), (0, 0, 2), (2, 0, 0), (1, 0, 1), (1, 0, 0), (2, 0, 0)$$

which would correspond to a seven-dimensional basis. The code which generate this set of vectors is given below:

```
def makeVectorList(t,n):
    if t==1:
      vectorList=[]
      for i in range (0,n):
        vectorList.append([i])
    else:
      newVectorList=[]
      vectorList=myFn.makeVectorList(t-1,n)
      for i in vectorList:
        sum=np.sum(i)
        for j in range (0,n-sum):
          i.append(j)
          icopy=i.copy()
          newVectorList.append(icopy)
          i.pop()
      vectorList=newVectorList
    return vectorList
```

```
'''
In the code seen above we create a list containing our basis vectors.

In other words, all possible permutations of i,j,k are found such
    that j+k+m does not exceed n.

This is done by finding all permutations for t=1 (1 dimensional
    vectors) and then adding more dimensions appending all
the numbers that are smaller than the current n-(i,j,k sum). This of
    course is generalized to work for all possible values of t and n.
'''
```

We remind that k must be even because of the PAULI principle. Therefore we must exclude all the vectors were this condition does not hold:

```
def kSymmetry(vector):
      newVector=[]
      for i in vector:
         if i[1]%2==0:
            copy=i.copy()
            newVector.append(copy)
      return newVector
```

```
'''
Here we get rid of all the basis vectors that have non even k values
    by checking the second element of each vector.
If that element is not divisible by 2, the vector is being tossed.
'''
```

With these set of vectors we are now in place to "fill" our matrices with their respective matrix elements.

3.3 Construction of the matrices

Our goal in this section is to construct the matrices which were presented in section 3.1. regardless of the dimensions which we chose. Before we do so though we have to specify our constants, some physical and others related to inequality (16):

```
class myCons:
   t=3
   n=3
   e_2=14.4
   a=0.1
   r_0=0.52917
   Z=2
   p=Z/(a*r_0)

   def setA(x):
      myCons.a=x
      myCons.p=myCons.Z/(myCons.a*myCons.r_0)
```

```
'''
t=The size of the vector list vectors. In our example t will be 3
   because of the extra 3 terms used to differentiate the Helium
   wave function from Hydrogen atom wave function.

n=The maximum sum of j,k and m. This constant has no physical
   significance. It is used to determine the number of basis vectors
   used for our wave function. The value of n will be changed
   throughout the program, to achieve for accurate results.

e_2=Electron charge squared.


a=The constant used to "guess" the correct wave function. This
   constant will again be changed by our program to yield the best
   result possible.

r=

Z=Atomic number of the element we are studying.

p=

setA= a setter function to give other classes of our program the
   possibility of overwriting the existing value of a. Subsequently
   p is also changed automatically due to the correlation of a and p.
'''
```

With that out of the way we are now ready to construct the Matrices which are described in section 3.1.. The code which is going to allow us to do this is the following:

```
def matrix(cons,fn,vectorList):
      l=len(vectorList)
      matrix=np.zeros((l,l))
      for i in range (l):
         for j in range (l):

               j1=vectorList[i][0]
               j2=vectorList[j][0]

               k1=vectorList[i][1]
               k2=vectorList[j][1]

               m1=vectorList[i][2]
               m2=vectorList[j][2]

               J=j1+j2
               K=k1+k2
               M=m1+m2

               matrix[i,j]=cons*fn(J,K,M)
      return matrix
```

```
'''
First and foremost the dimensions of our matrices depends on the
   number of our basis vectors (the size of our vectorList). Once
   the dimensions our known, the matrices can be initialized and are
   ready to be filled.

To calculate each and every element we use the functions defined
   previously. This can be done by using the corresponding fuction
   depending on what matrix needs to be filled. (func_w is used to
   fill matrix W etc.)
'''
```

This code works for the N, C and W matrices but not for the T for the reasons stated in section 3.1, the code which allows us to "fill" the T matrix is:

```
def tMatrix(cons,fn1,fn2,vectorList):
      l=len(vectorList)
      matrix=np.zeros((l,l))

      for i in range (l):
         for j in range (l):

               j1=vectorList[i][0]
               j2=vectorList[j][0]

               k1=vectorList[i][1]
               k2=vectorList[j][1]

               m1=vectorList[i][2]
```

```
                m2=vectorList[j][2]

                J=j1+j2
                K=k1+k2
                M=m1+m2

                if M==0:

                   matrix[i,j]=cons*fn1(J,K,M,j1,k1,m1,j2,k2,m2)
                else:
                   matrix[i,j]=cons*fn1(J,K,M,j1,k1,m1,j2,k2,m2)+\
                      cons*fn2(J,K,M,j1,k1,m1,j2,k2,m2)
      return matrix
```

```
'''
The logic used to fill matrices t1 and t2 is the same as before. The
   only difference is that we have vary the calculation of our
   matrix elements depending on the value of M.
'''
```

Equipped with these matrices we are now ready to compute the eigenvalues of the generalized eigenvalue problem presented in section 2.4.

4 Final Results and discussion

4.1 Numerical Results

Luckily for us python is able to calculate the eigenvalues of a generalized eigenvalue problem quite easily. What we need to do is to make the final eigenvalue a function of α so we can use the HYLLERAAS and UNDHEIM theorem. Python will then allow us to minimize this function and give us the desired upper limit of the Helium Atom ground state energy. The code together with the result is shown below:

```
def calcHamiltonian(T,C,W):
      H=T+C+W
      return H
```

```
'''
Our hamiltonian matrix was broken up into three seperate matrices to
   simplify the calculations and help with the maintenance and
   testing of our program. Here the three matrices are unified and
   yield the hamiltionian matrix.
'''
```

```
class main:
   def main():
      vectorList=myFn.makeVectorList(cons.t,cons.n)
      vectorList=myFn.kSymmetry(vectorList)

      N=myFn.matrix(1, myFn.func_n, vectorList)
      W=myFn.matrix(cons.e_2, myFn.func_w, vectorList)
      C=myFn.matrix(-cons.Z*cons.e_2, myFn.func_c, vectorList)

      T=myFn.tMatrix(0.5*cons.r_0*cons.e_2,
         myFn.func_t1,myFn.func_t2, vectorList)

      H=myFn.calcHamiltonian(T,C,W)

      eigvals, eigvecs = eigh(H, N, eigvals_only=False)

      energy=eigvals[0]

      return energy
```

```
'''
This is our main function invoking every function needed to
   calculate the final result of the Helium ground state energy.

The list of basis vectors is created and modified so that our wave
   function obeys to the pauli principle.

N,W,C and T matrices are calculated, all using functions we have
   already created.

Using the matrices above, the hamiltonian is calculated.

Now we are ready to obtain our final result. This is done by
   calculating the eigenvalues of the generalized eigenvalue
   problem. The first eigenvalue corresponds to the helium ground
   state energy.

'''
```

```
def f(a):
      cons.setA(a+1e-6)

      energy=main.main()

      return energy
```

```
'''
The calculated ground state energy of the helium atom will vary
   depending on our initial guess of the wave function. Here a trick
```

```
    is used to help calculate the energy for different values of the
    constant a. Due to the variational principle, the lowest possible
    energy gives us the best approximation of the ground state
    energy. So we use a function offered by the "Sci Py" package
    which allows us to find the value of a yielding the lowest
    possible energy.
'''
```

```
optA = optimize.brent(main.f)

print("min a:",optA)
print("energy:",main.f(optA))
```

```
'''
optA is the optimal a value for our wave function. Using optA we
    once again calculate the ground state energy which now
    corresponds to our best possible guess.
'''
```

If we run this code for $n = 8$, the final result for the ground state energy of the Helium Atom yields $-79.017\ eV$ while the experimentally measured value is $-79.005\ eV$.

For higher numbers of n the energy varies only slightly. Despite of that we can see that the final result is extremely close to experimentally measured value! What we can also observe is that as the dimensions of our basis is increased the eigenvalue converges to a single value (Figure 1), exactly like the HYLLERAAS and UNDHEIM theorem states, which means it must be very close to the exact eigenvalue (which it is).

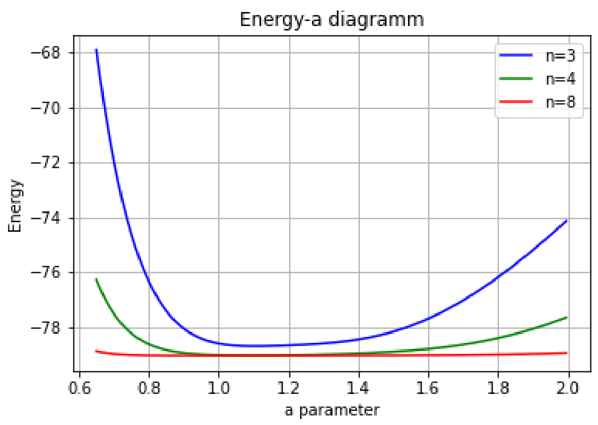

Figure 1: Convergence of the eigenvalue, we remind that n is not the dimension of the basis but the one defined in the inequality (16)

4.2 Two-electron Atoms

While this paper was originally intended to treat the Helium atom we noticed that the wave functions in equation (2) needn't necessarily apply to the Helium atom exclusively but to any two-electron atom, therefore it generalizes quite easily. The only thing we need to change is the charge of the atom nucleus. The code runs the same as before and gives results extremely close to the experimentally measured value. Our results together with the experimentally measured values (all in eV) can be seen in the table below:

Atom	$-E_{\text{cal}}$	$-E_{\text{exp}}$	$\frac{\Delta E}{E} \times 100$
H^-	14.361	14.360	0.007
He	79.017	79.005	0.015
Li^+	198.104	198.094	0.005
Be^{2+}	371.601	371.615	0.004
B^{3+}	599.516	599.598	0.013
C^{4+}	881.852	882.084	0.026
N^{5+}	1218.612	1219.113	0.041
O^{6+}	1609.795	1610.737	0.058

4.3 How did we calculate lower energies than the experimental value?

At this point you may have noticed a contradiction in our results, in many of them we calculate an energy eigenvalue lower than the experimentally measured energy. Is there an inconsistency with the Variational method and the HYLLERAAS and UNDHEIM theorem? The answer is no. These theorems state that with our trial wave functions we can't calculate an energy lower in value than that of the exact eigenvalue of the HAMILTON operator, but the experimentally measured energy is not the exact eigenvalue of the HAMILTON operator in equation (1)! The HAMILTON operator presented in equation (1) is only an approximation of the "true" HAMILTON operator that describes the Helium Atom. While this HAMILTON operator is an excellent approximation it remains an approximation which does not take into account any magnetic effects, it does not take into account potentials due to the electron and nucleus spin, it isn't a HAMILTON operator which takes special relativity into account etc. Therefore the experimentally measured value is the exact eigenvalue of the "true" HAMILTON operator and not the one described in equation (1).

Figure 2: These Polar bears have intentionally not been deleted (because they are cute).

References

[1] E.W.Schmid, G.Spitz, and W.Losch. *Theoretische Physik mit dem Personal Computer* (German).

[2] David Tong. *Lectures on Topics in Quantum Mechanics, Approximation Methods.*

[3] Experimental Data, NIST Atomic Spectra Database Ionization Energies Data

[4] Pictures of Polar bears, Polar Bear facts