# Βασική ενεργειακή στάθμη του ατόμου του Ηλίου.

Μια υπολογιστική προσέγγιση.

Ποζουκίδης Χ.
Ποζουκίδης Δ.

# Στόχος της εργασίας:

Υπολογισμός της βασικής ενεργειακής στάθμης του ατόμου του Ηλίου.

# Στόχος της εργασίας:

Υπολογισμός της βασικής ενεργειακής στάθμης του ατόμου του Ηλίου.

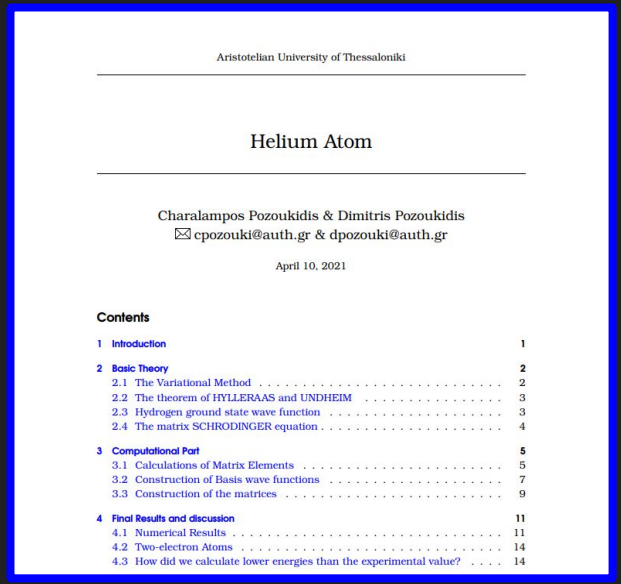
Για ποιό λόγο είναι αναγκαία η χρήση υπολογιστικών μεθόδων;

# Εισαγωγικά για την εργασία

Η παρουσίαση αυτή μαζί με την αντίστοιχη εργασία, όπως και τον κώδικα που χρησιμοποιήθηκε μπορεί να βρεθεί στο παρακάτω link:

Βασική Ενεργειακή στάθμη του ατόμου του Ηλίου.

Θα ανέβει άμεσα και στο e-learning του μαθήματος.



Aristotelian University of Thessaloniki

Helium Atom

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

April 10, 2021

Contents

# Εργαλεία

- Mathematica: Για τον υπολογισμό των ολοκληρωμάτων και την επιβεβαίωση των αποτελεσμάτων.

- Python: Για την αυτοματοποίηση πολύπλοκων διαδικασιών και το πλήθος συναρτήσεων υπολογιστικών μαθηματικών.

# Δομή παρουσίασης

- Μέρος Α’: Παρουσίαση της θεωρίας.
- Μέρος Β’: Δημιουργία των προγραμμάτων.
- Μέρος Γ’: Παρουσίαση αποτελεσμάτων.

# Μέρος Α’

Βασική θεωρία.

# Άτομο Υδρογόνου

Το άτομο του υδρογόνου μπορεί να λυθεί αναλυτικά.

# Άτομο Υδρογόνου

Το άτομο του υδρογόνου μπορεί να λυθεί αναλυτικά.

Η Χαμιλτονιανή για το άτομο του υδρογόνου:

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

# Άτομο Υδρογόνου

Λύνουμε για αυτή τη Χαμιλτονιανή την εξίσωση Schrödinger χωρίζοντάς την σε ακτινικό και γωνιακό κομμάτι εκμεταλλευόμενοι την σφαιρική συμμετρία.

# Άτομο Υδρογόνου

Λύνουμε για αυτή τη Χαμιλτονιανή την εξίσωση Schrödinger χωρίζοντάς την σε ακτινικό και γωνιακό κομμάτι εκμεταλλευόμενοι την σφαιρική συμμετρία.

Ύστερα αναπτύσσοντας σε δυναμοσειρά τη κυματοσυνάρτηση μπορούμε να βρούμε τη λύση για τη βασική ενεργειακή στάθμη του υδρογόνου :

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

# Άτομο του Ηλίου

Η Χαμιλτονιανή του ατόμου του Ηλίου:

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

# Άτομο του Ηλίου

Στο άτομο του ηλίου αλλάζει το φορτίο του πυρήνα που δεν επηρεάζει όμως την μεθοδολογία λύσης της διαφορικής.

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

# Άτομο του Ηλίου

Η εμφάνιση όμως του δεύτερου ηλεκτρονίου δυσκολεύει σημαντικά τη διαδικασία, λόγο του όρου άπωσης μεταξύ των ηλεκτρονίων.

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

# Άτομο του Ηλίου

Η εμφάνιση όμως του δεύτερου ηλεκτρονίου δυσκολεύει σημαντικά τη διαδικασία, λόγο του όρου άπωσης μεταξύ των ηλεκτρονίων.

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

Με τη προσθήκη αυτή στη Χαμιλτονιανή, η εξίσωση Schrödinger δεν μπορεί να λυθεί πλέον αναλυτικά.

# Εξίσωση Schrödinger

Η εξίσωση Schrödinger:

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

# Εξίσωση Schrödinger

Η εξίσωση Schrödinger:

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

Για τη δική μας εφαρμογή είναι απαραίτητη η μετατροπή της σε εξίσωση πινάκων.

# Εξίσωση Schrödinger

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

Αναπτύσσουμε τη κυματοσυνάρτησή μας σε μια διακριτή βάση, όχι ορθοκανονική.

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

Από εδώ και στο εξής για συντομία θα χρησιμοποιούμε n αντί των j,k και m.

# Εξίσωση Schrödinger

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

Με τη προσθήκη μιάς πλήρους βάσης:

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

# Εξίσωση Schrödinger

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

Με τη προσθήκη μιάς πλήρους βάσης:

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

Καταλήγουμε έτσι στη γενικευμένη εξίσωση ιδιοτιμών:

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

$$\tilde{H}_{n'n} = \left\langle\tilde{\Phi}_{n'}|H|\tilde{\Phi}_n\right\rangle$$

$$\tilde{N}_{n'n} = \left\langle\tilde{\Phi}_{n'}\mid\tilde{\Phi}_n\right\rangle$$

# Μέθοδος των μεταβολών

Έστω μια αυθαίρετη κατάσταση $|\psi\rangle$

Για αυτή τη κατάσταση θα ισχύει:

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

# Μέθοδος των μεταβολών

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

Απόδειξη:

# Μέθοδος των μεταβολών

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

Απόδειξη:

$$|\psi\rangle = \sum_n a_n |n\rangle$$

# Μέθοδος των μεταβολών

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

Απόδειξη:

$$|\psi\rangle = \sum_n a_n |n\rangle$$

$$\langle E \rangle = \sum_{n,m=0}^{\infty} a_m^\star a_n \langle m|H|n\rangle = \sum_{n,m=0}^{\infty} a_m^\star a_n E_n \delta_{mn}$$

$$= \sum_{n=0}^{\infty} |a_n|^2 E_n = E_0 \sum_{n=0}^{\infty} |a_n|^2 + \sum_{n=0}^{\infty} |a_n|^2 (E_n - E_0) \geq E_0$$

# Μέθοδος των μεταβολών

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

Αν η κυματοσυνάρτησή μας εξαρτάται από μια παράμετρο a τότε προκύπτει το αποτέλεσμα:

$$E(\alpha) \geq E_0$$

Και η καλύτερη προσέγγιση που μπορούμε να κάνουμε προκύπτει για a* τέτοιο ώστε:

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

# Θεώρημα Hylleraas και Undheim

Η μέθοδος των μεταβολών μπορεί να γενικευθεί σε περισσότερες διαστάσεις με το θεώρημα των Hylleraas και Undheim.

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

# Προσέγγιση του Ηλίου.

Υπενθυμίζουμε τη κυματοσυνάρτηση της βασική στάθμης του Υδρογόνου:

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

Μιάς και όπως δείξαμε πριν οι Χαμιλτονιανές των δύο ατόμων μοιάζουν, θα μπορούσαμε να προσεγγίσουμε τη βασική στάθμη του Ηλίου μέσω της παραπάνω κυματοσυνάρτησης.

# Προσέγγιση του Ηλίου.

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

Μια πρώτη προσέγγιση:

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

# Προσέγγιση του Ηλίου.

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

Μια πρώτη προσέγγιση:

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

# Προσέγγιση του Ηλίου.

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

Μια πρώτη προσέγγιση:

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

Η κυματοσυνάρτηση αυτή επιτρέπει στα δύο ηλεκτρόνια να έχουν ακριβώς την ίδια θέση.

# Προσέγγιση του Ηλίου.

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

Προσέγγιση Hylleraas:

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

# Προσέγγιση του Ηλίου.

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

Προσέγγιση Hylleraas:

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

Φροντίζει να μη μπορούν να έχουν τη ίδια θέση τα δύο ηλεκτρόνια, και βγάζει καλά αποτελέσματα.

# Προσέγγιση του Ηλίου.

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

Βελτίωση της προσέγγισης Hylleraas:

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

Αυτή θα είναι και η προσέγγιση που θα χρησιμοποιήσουμε για τους υπολογισμούς στην εργασία αυτή.

# Προσέγγιση του Ηλίου.

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

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

- α είναι η παράμετρος που μας αφήνει να χρησιμοποιήσουμε το θεώρημα των Hylleraas και Undheim.
- Οι μεταβλητές j,k και m είναι θετικοί ακέραιοι.
- Για να υπακούει στην αρχή του Pauli η κυματοσυνάρτηση το k πρέπει να είναι ζυγός αριθμός.
- Όσους περισσότερους συνδυασμούς συμπεριλάβουμε, τόσο καλύτερη θα είναι η προσέγγισή μας.

# Μέρος Β’

Εφαρμογή υπολογιστικών μεθόδων.

# Ορισμός Σταθερών

```
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)
```

Πριν ξεκινήσουμε οποιαδήποτε άλλη διαδικασία, αρχικά ορίζουμε τις σταθερές.

# Ορισμός Σταθερών

```
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)
```

Οι σταθερές t και n είναι για τη δημιουργία της βάσης.

# Ορισμός Σταθερών

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

```
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)
```

Η σταθερά t είναι ο αριθμός των όρων με τους οποίους πολλαπλασιάζουμε την κυματοσυνάρτηση του υδρογόνου για να τη διαφοροποιήσουμε.

Για το συγκεκριμένο φυσικό πρόβλημα θα παραμείνει 3.

# Ορισμός Σταθερών

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

```
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)
```

Η σταθερά t είναι ο αριθμός των όρων με τους οποίους πολλαπλασιάζουμε την κυματοσυνάρτηση του υδρογόνου για να τη διαφοροποιήσουμε.

Για το συγκεκριμένο φυσικό πρόβλημα θα παραμείνει 3.

t=1:

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

# Ορισμός Σταθερών

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

```
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)
```

n είναι το μέγιστο επιτρεπτό άθροισμα τον j,k και m. Δεν έχει κάποια φυσική σημασία αλλά υπάρχει για να ελέγχει των αριθμό των διανυσμάτων που θα χρησιμοποιήσουμε.

$$j+k+m<n$$

# Ορισμός Σταθερών

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

```
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)
```

Οι υπόλοιπες σταθερές είναι φυσικές σταθερές όπως το φορτίου του ηλεκτρονίου στο τετράγωνο, η ακτίνα Bohr, ο ατομικός αριθμός του Ηλίου.

Το p είναι απλά μια έκφραση των παραπάνω που εμφανίζεται συχνά, ενώ το a είναι η σταθερά με την οποία διαφοροποιήσαμε τον βασικό όρο της κυματοσυνάρτησης του υδρογόνου.

# Ορισμός Σταθερών

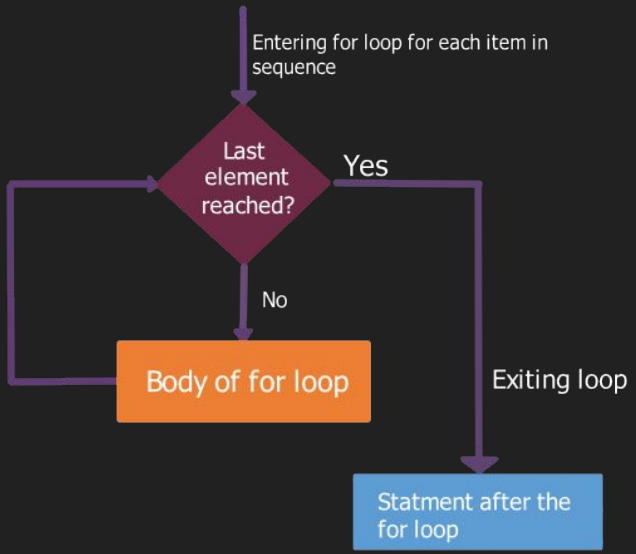
```
class myCons:
    t=3
    n=3
    e_2=14.4
    a=0.1
    r_0=0.52917
    Z=2
    p=Z/(a*r_0)

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

Συνάρτηση για να έχουμε πρόσβαση στη μεταβλητή a από άλλες κλάσεις.

Η μεταβλητή p ανανεώνεται αυτόματα με μια αλλαγή της a.

# for loops

```
for i in range (a,b):
  #code
```

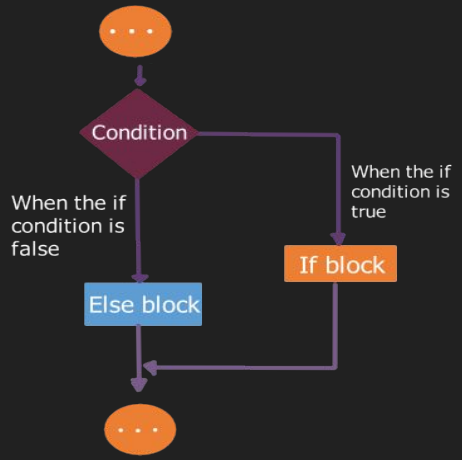



Η βασική δομή ενός “for loop”.

Η μεταβλητή i παίρνει όλες τις τιμές στο διάστημα [a,b) με βήμα 1, και σε κάθε επανάληψη εκτελούνται οι εντολές της for.

# if statement

```
if condition==True:
  #code if True
else:
  #code if False.
```




Η βασική δομή ενός “if statement”.

Αν ισχύει η συνθήκη εκτελείται το κομμάτι κώδικα κάτω από το if.

Αν δεν ισχύει η συνθήκη, εκτελείται ο κώδικας κάτω από το else.

# Δημιουργία βάσης

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

```
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
```

Εδώ φαίνεται πως μπορούμε να φτιάξουμε μια βάση συναρτήσεων που θα αποτελούν τη κυματοσυνάρτησή μας.

# Δημιουργία βάσης

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

```
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
```

Εδώ φαίνεται πως μπορούμε να φτιάξουμε μια βάση συναρτήσεων που θα αποτελούν τη κυματοσυνάρτησή μας.

Φτιάχνουμε μια συνάρτηση, που θα μας επιστρέφει μια λίστα από διανύσματα.

# Δημιουργία βάσης

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

```
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
```

Εδώ φαίνεται πως μπορούμε να φτιάξουμε μια βάση συναρτήσεων που θα αποτελούν τη κυματοσυνάρτησή μας.

Φτιάχνουμε μια συνάρτηση, που θα μας επιστρέφει μια λίστα από διανύσματα.

Δέχεται σαν μεταβλητές το t και n.

# Δημιουργία βάσης

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

```
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
```

Εδώ φαίνεται πως μπορούμε να φτιάξουμε μια βάση συναρτήσεων που θα αποτελούν τη κυματοσυνάρτησή μας.

Φτιάχνουμε μια συνάρτηση, που θα μας επιστρέφει μια λίστα από διανύσματα.

Δέχεται σαν μεταβλητές το t και n.

Η βάση δημιουργείται χρησιμοποιώντας μια “recursive” συνάρτηση η οποία καλεί την εαυτό της κατά τη διάρκεια εκτέλεσής της (όπως το n!).

# Δημιουργία βάσης

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

`t=1:`

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

`n=3:`

$$m<3$$

# Δημιουργία βάσης

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

Διανύσματα βάσης για t=1 και n=3:

$$\Phi_0\left(\overrightarrow{r_1},\overrightarrow{r_2}\right)=\exp\left(\frac{-Z}{\alpha r_0}\left(r_1+r_2\right)\right)$$

$$\Phi_1\left(\overrightarrow{r_1},\overrightarrow{r_2}\right)=\left|\overrightarrow{r_1}-\overrightarrow{r_2}\right|\exp\left(\frac{-Z}{\alpha r_0}\left(r_1+r_2\right)\right)$$

$$\Phi_2\left(\overrightarrow{r_1},\overrightarrow{r_2}\right)=\left|\overrightarrow{r_1}-\overrightarrow{r_2}\right|^2\exp\left(\frac{-Z}{\alpha r_0}\left(r_1+r_2\right)\right)$$

# Δημιουργία βάσης

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

t=3:

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

n=3:

$$j+k+m<3$$

# Δημιουργία βάσης

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

Διανύσματα βάσης για t=3 και n=3:

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

# Δημιουργία βάσης

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

```
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
```

Πως λειτουργεί ο κώδικας:

1.Φτιάχνουμε τη πιο απλή βάση για t=1 ανάλογα με το n.

2.Επεκτείνουμε τη βάση μέχρι να φτάσουμε στο t που επιθυμούμε.

# Δημιουργία βάσης

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

```
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
```

Παράδειγμα για n=3:

Βάση για t=1:

$$[(0),(1),(2)]$$

# Δημιουργία βάσης

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

```
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
```

Παράδειγμα για n=3:

Βάση για t=1:

$$[(0),(1),(2)]$$

Βάση για t=2:

$$[(0,0),(0,1),(0,2),(1,0),(1,1),(2,0)]$$

# Δημιουργία βάσης

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

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

“Καθαρίζουμε” της βάση ώστε η κυματοσυνάρτησή μας να υπακούει στην αρχή του Pauli.

Ελέγχουμε αν το K είναι άρτιος. Αν ναι, προστίθεται στη λίστα με τα διανύσματα βάσης.

# Υπολογισμός ολοκληρωμάτων

Αρχικά πρέπει να υπολογίσουμε τον πίνακα της Χαμιλτονιανής και τον πίνακα Ν.

$$\tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$

$$\tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

# Υπολογισμός ολοκληρωμάτων

$$\tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$\tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Για να διευκολύνουμε τους υπολογισμούς μπορούμε να χωρίσουμε τη Χαμιλτονιανή σε τρία μέρη.

$$H = T + C + W$$

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

# Υπολογισμός ολοκληρωμάτων

$$\tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$\tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Για να διευκολύνουμε τους υπολογισμούς μπορούμε να χωρίσουμε τη Χαμιλτονιανή σε τρία μέρη.

$$H = T + C + W$$

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

Η φυσική σημασία των πινάκων είναι η συνεισφορά του φορτίου του πυρήνα, της κινητικής ενέργειας και του όρου άπωσης στη τελική ενέργεια.

# Υπολογισμός ολοκληρωμάτων

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Χρήση της εντολής Integrate για τριπλά ολοκληρώματα στη Mathematica:

$$\int_{-1}^{1} \int_{0}^{\infty} \int_{0}^{\infty} x^2 y^2 (x+y)^J (x-y)^K \left(x^2+y^2-2(xyz)\right)^{M/2} \exp(-2l(x+y)) dy dx dz$$

# Υπολογισμός ολοκληρωμάτων

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Χρήση της εντολής Integrate για τριπλά ολοκληρώματα στη Mathematica:

$$\int_{-1}^{1} \int_{0}^{\infty} \int_{0}^{\infty} x^2 y^2 (x+y)^J (x-y)^K \left(x^2 + y^2 - 2(xyz)\right)^{M/2} \exp(-2l(x+y)) dy dx dz$$

Αναμενόμενο Αποτέλεσμα μετά από πολλαπλασιασμό με $4\pi^2$:

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

# Υπολογισμός ολοκληρωμάτων

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Απλοποίηση ολοκληρώματος:

$$\int_0^\infty \int_{-v}^{v} \int_{-1}^{1} \frac{1}{16} \left(v^2 - w^2\right)^2 v^J w^K \left(\frac{1}{2}\left(v^2 + w^2 - u\left(v^2 - w^2\right)\right)\right)^{M/2} \exp(-2av) du dw dv$$

Αναμενόμενο Αποτέλεσμα μετά από πολλαπλασιασμό με $4\pi^2$:

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

# Υπολογισμός ολοκληρωμάτων

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Απλοποίηση ολοκληρώματος και αντικατάσταση J,K και M:

$$\int_0^\infty \int_{-v}^{v} \int_{-1}^{1} \frac{1}{16} \left(v^2 - w^2\right)^2 v^J w^K \left(\frac{1}{2}\left(v^2 + w^2 - u\left(v^2 - w^2\right)\right)\right)^{M/2} \exp(-2av) du dw dv$$

# Υπολογισμός ολοκληρωμάτων

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} | H | \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Απλοποίηση ολοκληρώματος και αντικατάσταση J,K και M:

$$\int_0^\infty \int_{-v}^{v} \int_{-1}^{1} \frac{1}{16} \left(v^2 - w^2\right)^2 v^J w^K \left(\frac{1}{2}\left(v^2 + w^2 - u\left(v^2 - w^2\right)\right)\right)^{M/2} \exp(-2av) du dw dv$$

Αναμενόμενο Αποτέλεσμα μετά από πολλαπλασιασμό με $4\pi^2$:

$$\tilde{N}_{11} = 0.00453301$$

*Ανάλογα βγαίνουν και τα υπόλοιπα ολοκληρώματα.

# Υπολογισμός στοιχείων πίνακα

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} \middle| H \middle| \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

```
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)
```

Παραπάνω βλέπουμε πως υπολογίζουμε τα στοιχεία των πινάκων N,C και W.
Ανάλογα υπολογίζουμε και τον πίνακα T.

# Δημιουργία πινάκων

```
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
```

$$J = j_1 + j_2 \quad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} \middle| H \middle| \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \quad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Εδώ ξεκινάμε να γεμίσουμε τους πίνακες που χρειαζόμαστε, υπολογίζοντας τα στοιχεία τους, με τις προηγούμενες συναρτήσεις.

# Δημιουργία πινάκων

```
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
```

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} \middle| H \middle| \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Εδώ ξεκινάμε να γεμίσουμε τους πίνακες που χρειαζόμαστε, υπολογίζοντας τα στοιχεία τους, με τις προηγούμενες συναρτήσεις.

Περνάμε μια φορά από κάθε στοιχείο του πίνακα.

# Δημιουργία πινάκων

```
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
```

$$J = j_1 + j_2 \qquad \tilde{H}_{n'n} = \left\langle \tilde{\Phi}_{n'} \middle| H \middle| \tilde{\Phi}_n \right\rangle$$
$$K = k_1 + k_2$$
$$M = m_1 + m_2 \qquad \tilde{N}_{n'n} = \left\langle \tilde{\Phi}_{n'} \mid \tilde{\Phi}_n \right\rangle$$

Εδώ ξεκινάμε να γεμίσουμε τους πίνακες που χρειαζόμαστε, υπολογίζοντας τα στοιχεία τους, με τις προηγούμενες συναρτήσεις.

Περνάμε μια φορά από κάθε στοιχείο του πίνακα.

Για κάθε στοιχείο καλούμε την αντίστοιχη συνάρτηση για τον υπολογισμό της τιμής του.

# Υπολογισμός Χαμιλτονιανής

$$H = T + C + W$$

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

Για τον υπολογισμό της Χαμιλτονιανής προσθέτουμε τους πίνακες T,C και W.

# Συνάρτηση main.

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

```
def main():
```

Ήρθε η ώρα να χρησιμοποιήσουμε όλες τις παραπάνω συναρτήσεις.

# Συνάρτηση main.

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

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

Φτιάχνουμε τη βάση.

# Συνάρτηση main.

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

```
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)
```

Φτιάχνουμε τους πίνακες.

# Συνάρτηση main.

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

```
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)
```

Υπολογίζουμε τη Χαμιλτονιανή.

# Συνάρτηση main.

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

```
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)
```

Λύνουμε το γενικευμένο πρόβλημα ιδιοτιμών.

# Συνάρτηση main.

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

```
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]
```

Η πρώτη ενέργεια είναι αυτή της βασικής στάθμης του Ηλίου.

# Συνάρτηση main.

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

```
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]
```

Η πρώτη ενέργεια είναι αυτή της βασικής στάθμης του Ηλίου.

*Για το τυχαίο a που έχουμε ορίσει.

# Υπολογισμός βέλτιστου a.

$$E(\alpha) \geq E_0$$

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

  energy=main()

  return energy
```

Φτιάχνουμε μια συνάρτηση που θα υπολογίζει την ενέργεια σαν συνάρτηση του α.

# Τελικές εντολές

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

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

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

optA είναι η τιμή του a που μας δίνει τα βέλτιστα αποτελέσματα.

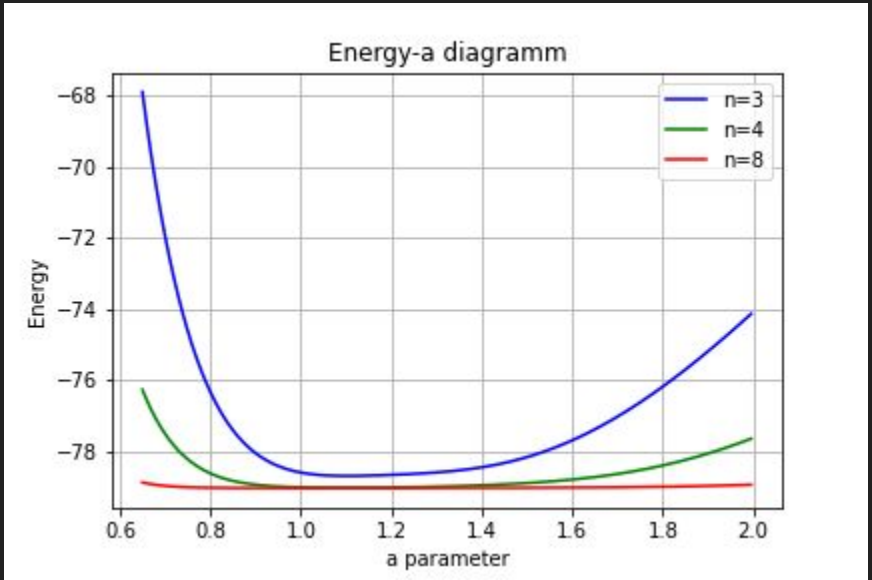
Για να τη βρούμε, χρησιμοποιούμε τη συνάρτηση “optimize” που μας προσφέρει το υπολογιστικό πακέτο “SciPy”.

Τέλος τυπώνουμε τα τελικά μας αποτελέσματα: Την τιμή του a και την ενέργεια της βασικής στάθμης του Ηλίου.

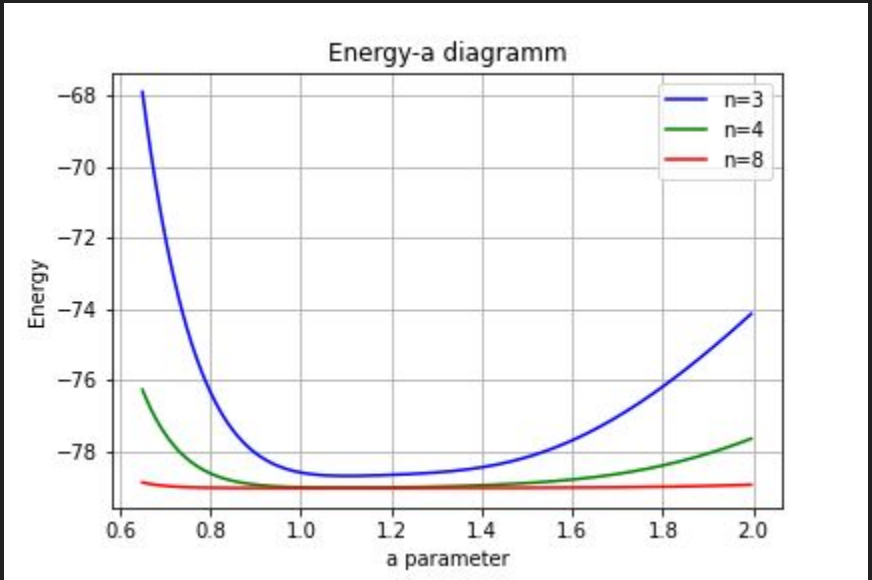
# Μέρος Γ’

Τελικά αποτελέσματα & σχόλια.

# Τελικά αποτελέσματα.



# Τελικά αποτελέσματα.




Ενέργεια:-79.017eV.

# Τελικά αποτελέσματα.

| Atom | $-E_{\text{cal}}$ | $-E_{\text{exp}}$ | $\frac{\Delta E}{E} \times 100$ |
|---|---|---|---|
| He | 79.017 | 79.005 | 0.015 |

# Τελικά αποτελέσματα.

| Atom | $-E_{\mathrm{cal}}$ | $-E_{\mathrm{exp}}$ | $\frac{\Delta E}{E} \times 100$ |
|---|---|---|---|
| $H^-$ | 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 |

Τελικά αποτελέσματα.

$$E(\alpha) \geq E_0$$

| 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 |

# Σχόλια

| Atom | $-E_{\text{cal}}$ | $-E_{\text{exp}}$ | $\frac{\Delta E}{E} \times 100$ |
|---|---|---|---|
| $H^-$ | 14.361 | 14.360 | 0.007 |

Στα αποτελέσματα μας βγάζουμε τιμές ενεργειών μικρότερες από τις πειραματικές, παραβιάζεται το θεώρημα των Hylleraas και Undheim?

# Σχόλια

| Atom | $-E_{\text{cal}}$ | $-E_{\text{exp}}$ | $\frac{\Delta E}{E} \times 100$ |
|---|---|---|---|
| $H^-$ | 14.361 | 14.360 | 0.007 |

Στα αποτελέσματα μας βγάζουμε τιμές ενεργειών μικρότερες από τις πειραματικές, παραβιάζεται το θεώρημα των Hylleraas και Undheim?

Όχι, γιατί η πειραματικά μετρημένη ενέργεια δεν είναι η ιδιοτιμή της Χαμιλτονιανής που παρουσιάσαμε!

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

# Σχόλια

| Atom | $-E_{\text{cal}}$ | $-E_{\text{exp}}$ | $\frac{\Delta E}{E} \times 100$ |
|---|---|---|---|
| $H^-$ | 14.361 | 14.360 | 0.007 |

Στα αποτελέσματα μας βγάζουμε τιμές ενεργειών μικρότερες από τις πειραματικές, παραβιάζεται το θεώρημα των Hylleraas και Undheim?

Όχι, γιατί η πειραματικά μετρημένη ενέργεια δεν είναι η ιδιοτιμή της Χαμιλτονιανής που παρουσιάσαμε!

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

Αγνοούμε την κίνηση του Πυρήνα, μαγνητικά φαινόμενα, Σπίν, Σχετικότητα...

# Πηγές

[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.