**Radial Wave Function of Hydrogen Atom**

陳光悦

透過氫原子模型的radio wave function，可以得到氫原子的能量eigenvalue和eigenvector。相關的公式推導如下：

令

此時，動能項、庫倫位能項、角動量位能項。令可以用sin函數展開，即。

則，動能項：

角動量位能項：

庫倫位能項：

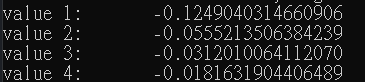
當，即可表示為。*H*視為hamiltonian operator，*E*為氫原子模型中的能量eigenvalue，*C*則為eigenvector。

在標準的氫原子模型中，能級可以表示為：

為了簡化計算，在計算時將一些基本常數皆設為1。即在假設的情況下，可將上述公式簡化為：

根據上述公式，可以計算出前幾個能級的理論特徵值：

而透過我的程式，得到前幾個能級的能量eigenvalue：



可以看到value 1到value 4的計算值，與到的理論值基本吻合。

程式中無法計算出基態時的能量，是因為在程式碼中，角動量量子數被設為1，而基態對應的角動量量子數應為0。根據量子力學的理論，角動量量子數和主量子數的關係是。因此，只有在時，才能計算出的能量值。

程式中積分的部分，是以Gaussian-Legendre quadrature實現，使用事先計算好的節點與權重來計算定積分的近似值。節點是Legendre polynomials的根，而權重則是根據這些節點再去做計算得出來的。

其計算定積分方式如下：

程式中最終選定用96個節點的Gaussian-Legendre quadrature，其計算出的能量值可以到，再往後的誤差會逐漸加劇。若想減少誤差，則需選用更多的節點。

以下附上主程式碼：

int main(){

int i, j, n, m;

double l, L;

double ansOne, ansTwo;

double ansK, ansV1, ansV2, ansS;

double \*\*H, \*\*S, \*eigenvalue, \*q;

double \*\*K, \*\*V1, \*\*V2;

//初始化參數

n = 50; //以sin函數展開50項

L = 50.0; //radial 的範圍，0~50

l = 1.0; //角動量量子數

H = dmatrix(1, n, 1, n);

K = dmatrix(1, n, 1, n);

V1 = dmatrix(1, n, 1, n);

V2 = dmatrix(1, n, 1, n);

S = dmatrix(1, n, 1, n);

eigenvalue = dvector(1, n);

q = dvector(1, n);

for(i=1; i<=n; i++){

for(j=1; j<=n; j++){

ansK = IntK(0, L, i, j);

ansV1 = IntV1(0, L, i, j);

ansV2 = IntV2(0, L, i, j);

H[i][j] = ansK + ansV1 + ansV2;

K[i][j] = ansK;

V1[i][j] = ansV1;

V2[i][j] = ansV2;

ansS = IntS(0, L, i, j);

S[i][j] = ansS;

}

}

hseigen(n, H, S, eigenvalue);

mergeSort(eigenvalue, H, 1, n);

for(i=1; i<=n; i++){

printf("value %d: %25.16lf\n", i, eigenvalue[i]);

}

printf("\n");

for(i=1; i<=n; i++){

printf("Eigenvector %d: ", i);

for(j=1; j<=n; j++){

printf("%25.16lf ", H[j][i]);

}

printf("\n");

}

free\_dvector(q, 1, n);

free\_dvector(eigenvalue, 1, n);

free\_dmatrix(H, 1, n, 1, n);

free\_dmatrix(S, 1, n, 1, n);

return 0;

}

//integral of kinetic from a to b

double IntK(double a, double b, int n, int m){

int i;

double L = b-a;

double one, two;

double sum = 0.0;

for(i=0; i<48; i++){

one = (b-a)/2.0 \* (xi[i]+1) + a;

two = (b-a)/2.0 \* (xi[i]\*-1+1) + a;

sum = sum + wi[i] \* cos(PI\*n\*one / L) \* one\*one \* (1.0/2.0)\*(PI\*n/L)\*(PI\*m/L) \* cos(PI\*m\*one / L);

sum = sum + wi[i] \* cos(PI\*n\*two / L) \* two\*two \* (1.0/2.0)\*(PI\*n/L)\*(PI\*m/L) \* cos(PI\*m\*two / L);

}

return (b-a)/2.0 \* sum;

}

//integral of potential of Coulomb from a to b

double IntV1(double a, double b, int n, int m){

int i;

double L = b-a;

double one, two;

double sum = 0.0;

for(i=0; i<48; i++){

one = (b-a)/2.0 \* (xi[i]+1) + a;

two = (b-a)/2.0 \* (xi[i]\*-1+1) + a;

sum = sum + wi[i] \* sin(PI\*n\*one / L) \* (-1.0\*one) \* sin(PI\*m\*one / L);

sum = sum + wi[i] \* sin(PI\*n\*two / L) \* (-1.0\*two) \* sin(PI\*m\*two / L);

}

return (b-a)/2.0 \* sum;

}

//integral of potential of angular momentum from a to b

double IntV2(double a, double b, int n, int m){

int i;

double L = b-a;

int l = 1;

double one, two;

double sum = 0.0;

for(i=0; i<48; i++){

one = (b-a)/2.0 \* (xi[i]+1) + a;

two = (b-a)/2.0 \* (xi[i]\*-1+1) + a;

sum = sum + wi[i] \* sin(PI\*n\*one / L) \* l\*(l+1)/2 \* sin(PI\*m\*one / L);

sum = sum + wi[i] \* sin(PI\*n\*two / L) \* l\*(l+1)/2 \* sin(PI\*m\*two / L);

}

return (b-a)/2.0 \* sum;

}

//integral of S from a to b

double IntS(double a, double b, int n, int m){

int i;

double L = b-a;

double one, two;

double sum = 0.0;

for(i=0; i<48; i++){

one = (b-a)/2.0 \* (xi[i]+1) + a;

two = (b-a)/2.0 \* (xi[i]\*-1+1) + a;

sum = sum + wi[i] \* sin(PI\*n\*one / L) \* one\*one \* sin(PI\*m\*one / L);

sum = sum + wi[i] \* sin(PI\*n\*two / L) \* two\*two \* sin(PI\*m\*two / L);

}

return (b-a)/2.0 \* sum;

}