

组会

薄纪铮

2023.3.28

验证 HO basis 的有效性

方式

- 将 np 的 Hamiltonian 放到 HO basis 下去计算基态

流程

- 写出 HO basis
- 计算 Hamiltonian 的矩阵元
- 使用 lapack 中的对角化求特征值的函数计算基态

写 HO basis

步骤

- 选择公式 $\psi_n(x) = N_n \exp(-\frac{1}{2}\alpha^2 x^2) H_n(\alpha x)$, 其中归一化系数 $N_n = [\alpha/\sqrt{\pi} 2^n n!]^{\frac{1}{2}}$
- 令 $\alpha = 1$, 并不影响 basis 的正交完备性
- 写归一化系数 N_n
- 写 Hermitian 多项式 $H_n(z)$
- 做乘积得到 HO basis

归一化系数 N_n

用 fortran 写系数 $N_n = 1/\sqrt{\pi}2^n n!$

- 因为阶乘没有内置，所以主要内容就是写阶乘 $n!$

阶乘函数 $factoria(n)$ 的程序实现

```
54  real function factorial(n) !阶乘函数
55  implicit none
56  integer::i,n             !输入为整数
57  if(n==0) then
58  factorial=1
59  else
60
61  factorial=1
62
63  do i=1,n
64  factorial=factorial*i
65
66  end do
67  end if
68
69  end function
```

Figure 1: 阶乘函数

归一化系数 N_n

N_n 的程序实现

在令常数 π 取 8 位小数后，可以得到 N_n

```
real*8,parameter :: PI=3.14159265
```

```
71 real function N_n(n)
72     use parameter      !用到PI
73     use func           !用到前面写的阶乘函数func
74     integer::n        !调用阶乘时，阶乘函数定义的输入是整型数
75     real*8::alpha
76     alpha=1.0
77     N_n=sqrt(alpha/(sqrt(PI)*2**n*factorial(n)))
78
79 end function
```

Hermitian 多项式

基本公式选择

- 方法 1

$$\begin{aligned} H_n(z) &= (2z)^n - n(n-1)(2z)^{n-2} \\ &\quad + \frac{n(n-1)(n-2)(n-3)}{2!} (2z)^{n-4} - \dots + \dots \\ &\quad + (-1)^{\lfloor \frac{n}{2} \rfloor} \cdot \frac{n!}{\lfloor \frac{n}{2} \rfloor!} (2z)^{n-2\lfloor \frac{n}{2} \rfloor} \end{aligned}$$

其中 $\lfloor \frac{n}{2} \rfloor$ 是不大于 $\frac{n}{2}$ 的最大整数

$$\lfloor \frac{n}{2} \rfloor = \begin{cases} \frac{n}{2} & (n \text{ 为偶数}) \\ \frac{n-1}{2} & (n \text{ 为奇数}) \end{cases}$$

Hermitian 多项式

基本公式选择

- 方法 2: 用递推: $H_{n+1}(z) - 2zH_n(z) + 2nH_{n-1}(z) = 0$

如何选取

- 选择方法 1: 它是一个直接的展开形式, 虽然较复杂, 但调用可以直接得第 n 个 Hermitian 多项式, 只需做一次求和
- 若选择递推方法 2: 它需要前面所有的 Hermitian 多项式去做递推才能得到第 n 个, 形式简单, 但计算量会多一些 (因为每一个 $H_n(z)$ 都是一个长的数组)

Hermitian 多项式

化简

- 将方法 1 的求和化简为对通项求和的形式

$$H_n(z) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^k \frac{n!}{k!(n-2k)!} (2z)^{n-2k}$$

- 这就是程序里面用的循环体

Hermitian 多项式

程序实现

```
54   real function Hermitian(n,z)!n从0开始取
55
56   implicit none
57   integer::k,n
58   real::z
59   Hermitian=0
60   if (mod(n,2)==0) then !偶数除以2余数为0
61       do k=0,n/2
62           Hermitian=Hermitian+(-1)**k*factorial(n)
63   &       *(2*z)**(n-2*k)/(factorial(k)*factorial(n-2*k))
64       end do
65   else           !else里面就是奇数情况
66       do k=0,(n-1)/2
67           Hermitian=Hermitian+(-1)**k*factorial(n)
68   &       *(2*z)**(n-2*k)/(factorial(k)*factorial(n-2*k))
69       end do
70   end if
71
72   end function
```

HO basis $\psi_n(z)$

将 HO basis 写成 function

- $\psi_n(z) = N_n H_n(z) \exp(-z^2/2)$

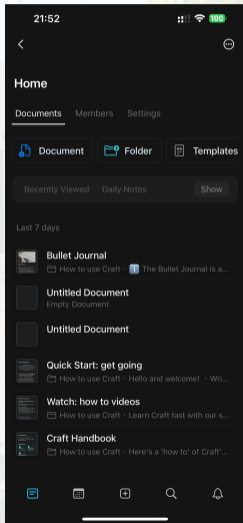
```
74      real function HObasis(n,z)
75          implicit none
76          integer::n
77          real::z
78          HObasis=N_n(n)*Hermitian(n,z)*exp(-z*z/2)
79      end function
```

移动端使用

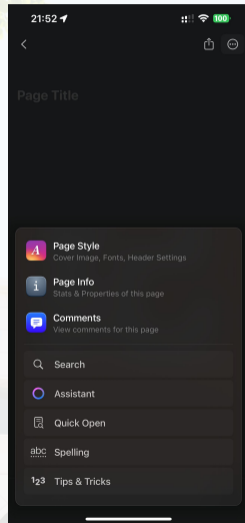
- 在 Apple 商店里安装 craft
- 在 craft 里面随意创建一个 document
- 在更多选项里找到”Assistant” 项即可使用



薄纪铮



组会

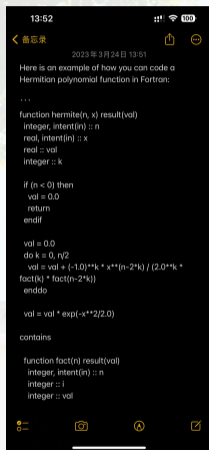


2023.3.28

GPT 用于学习工作

- 构造程序结构（不知道该怎么做的情下去建立思路）
- 验证自己所写内容的正确性

GPT 的具体例子



13:52

< 备忘录

2023年3月24日 13:51

Here is an example of how you can code a Hermitian polynomial function in Fortran:

```
...
function hermite(n, x) result(val)
integer, intent(in) :: n
real, intent(in) :: x
real :: val
integer :: k

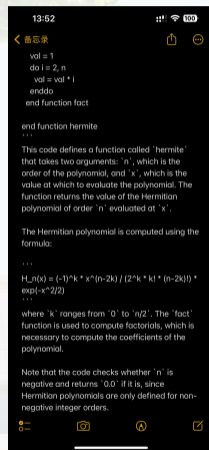
if (n < 0) then
val = 0.0
return
endif

val = 0.0
do k = 0, n/2
val = val + (-1)**k * x**(n-2*k) / (2.0**k *
fact(k) * fact(n-2*k))
enddo

val = val * exp(-x**2/2.0)

contains

function fact(n) result(val)
integer, intent(in) :: n
integer :: i
integer :: val
```



13:52

< 备忘录

```
val = 1
do i = 2, n
val = val * i
enddo
end function fact

end function hermite
...
```

This code defines a function called "hermite" that takes two arguments: "n", which is the order of the polynomial, and "x", which is the value at which to evaluate the polynomial. The function returns the value of the Hermitian polynomial of order "n" evaluated at "x".

The Hermitian polynomial is computed using the formula:

$$H_n(x) = (-1)^k * x^{n-2k} / (2^k * k! * (n-2k)!) * \exp(-x^2/2)$$

where "k" ranges from "0" to "n/2". The "fact" function is used to compute factorials, which is necessary to compute the coefficients of the polynomial.

Note that the code checks whether "n" is negative and returns "0.0" if it is, since Hermitian polynomials are only defined for non-negative integer orders.

Figure 2: 给 GPT 提问：如何用通项求和实现 Hermitian 多项式，以此来验证自己总结推导通项的正确性



Thank You!