用现实核力求解np束缚态

$$\begin{aligned} \text{s-wave:} \quad & |\alpha_1\rangle = |l=0; j=1\rangle \\ \text{d-wave:} \quad & |\alpha_2\rangle = |l=2, j=1\rangle \end{aligned} \tag{1}$$

$$|\phi\rangle = \frac{1}{E-T} V |\phi\rangle \tag{2}$$

$$\begin{aligned} \langle k_i\alpha|\phi\rangle &= \sum_{\alpha'} \frac{1}{E-T} \sum_j \langle k_i\alpha| V |k_j\alpha'\rangle \langle k_j\alpha'|\phi\rangle \\ &= \sum_{\alpha'} \frac{1}{E-\frac{k_i^2}{2\mu}} \int V(k_i\alpha, k_j\alpha') \langle k_j\alpha'|\phi\rangle k_j^2 \mathrm{d}k_j \\ &= \sum_{\alpha'} \sum_j \omega_j k_j^2 \frac{1}{E-\frac{k_i^2}{2\mu}} V(k_i\alpha, k_j\alpha') \langle k_j\alpha'|\phi\rangle \end{aligned} \tag{3}$$

对α展开求和，得到：（设$c_{ij} = \omega_j k_j^2/(E-k_i^2/2\mu)$）

$$\begin{aligned} \langle k_i0|\phi\rangle &= \sum_j c_{ij} V(k_i0, k_j0) \langle k_j0|\phi\rangle + \sum_j c_{ij} V(k_i0, k_j2) \langle k_j2|\phi\rangle \\ \langle k_i2|\phi\rangle &= \sum_j c_{ij} V(k_i2, k_j0) \langle k_j0|\phi\rangle + \sum_j c_{ij} V(k_i2, k_j2) \langle k_j2|\phi\rangle \end{aligned} \tag{4}$$

写成分块矩阵的形式（长度为2*np）：

$$\begin{pmatrix} \phi_0 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} A_{00} & A_{02} \\ A_{20} & A_{22} \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_2 \end{pmatrix} \tag{5}$$

束缚态波函数（长度为np）：$\Phi = \phi_0 + \phi_2$

D波的概率（将Φ归一化后）：$D\% = \langle\phi_2|\Phi\rangle = \langle\phi_2|\phi_2\rangle$

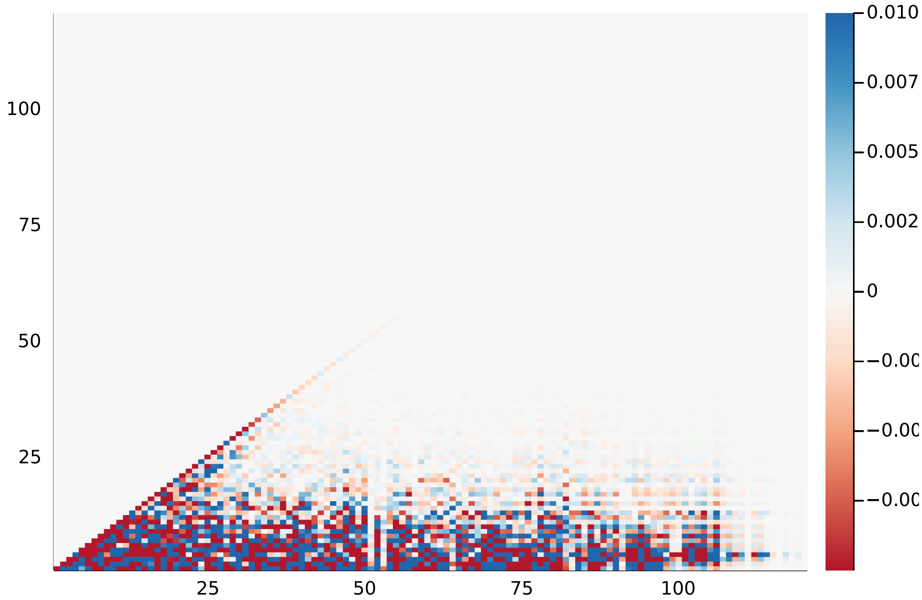
验证自洽性:

$$\begin{aligned} \langle\phi| T+V |\phi\rangle =& \langle\phi| T |\phi\rangle + \langle\phi| V |\phi\rangle \\ =& \sum_\alpha \sum_i \omega_i k_i^2 \langle\phi|k_i\alpha\rangle \frac{k_i^2}{2\mu} \langle k_i\alpha|\phi\rangle + \sum_{\alpha\alpha'} \sum_{ij} \omega_i\omega_j k_i^2 k_j^2 \langle\phi|k_i\alpha\rangle V(k_i\alpha, k_j\alpha') \langle k_j\alpha'|\phi\rangle \\ =& \mathrm{sum}((\phi_0^2+\phi_2^2) * \omega * k^4)/(2\mu) + \sum_{ij} \omega_i\omega_j k_i^2 k_j^2 \phi_0(i)\phi_0(j) V_{00}(ij) + \cdots \end{aligned} \tag{6}$$

Krylov子空间算法

对象：大型稀疏矩阵的本征值问题

本问题中的A矩阵:



目的：找到模方最大的特征值 $\to$ 幂迭代

$$K_n = [b,\ Ab,\ A^2b,\ \cdots,\ A^{n-1}b] \tag{7}$$

故使用方法：**Arnoldi iteration**

理解

$H(n,n)$为$A(N,N)$在子空间K_n内的表达，$Q(N,n)=\{q_n(N)\}$为子空间的基

$$H_n = Q_n^* A Q_n \tag{8}$$

$$H_n = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix} \tag{9}$$

若H_n的本征值与本征向量为$\{\lambda\}$和$\{v\}$（长度为n），则A的近似本征值与本征向量为$\{\lambda\}$和$\{Qv\}$（长度为N）。

操作

通过Gram–Schmidt process正交化基底$q_1, q_2, q_3 \cdots, q_n$，叫作Arnoldi vectors，它们张成了Krylov subspace $\mathcal{K}_n$。具体操作为：

- 任意归一化初始向量q_1（事实上似乎需要一些针对性选取更好）
- 对于$k=2,3,\cdots$

  $q_k \leftarrow Aq_{k-1}$　# 幂迭代操作

  - 对于$j=1,2,\cdots,k-1$

    $h_{j,k-1} \leftarrow q_j^* q_k$

$q_k \leftarrow q_k - h_{j,k-1} q_j$ # 施密特正交化

- ◦ $h_{k,k-1} \leftarrow ||q_k||$ # 为H_{k-1}的拓展部份，求本征值时只考虑$k-1$维的部分
- ◦ $q_k \leftarrow q_k / h_{k,k-1}$ # 归一化基底

```julia
function arnoldi_bases(A, q0, miniter, maxiter, tol;
    howmany=1, kstep=1
)
    # q0: 随机初始向量
    # miniter: 限制最小维度，不然容易两次偶然相等
    # maxiter: 限制最大维度，不应到计算到和A一样
    # howmany: 需要计算多少个 howmany > miniter+1
    # tol:  收敛精度要求多少
    # howmany: 需要多少个，只支持幅度最大本征值的筛选
    # kstep: 如果需要的维度比较高，不必每次计算本征值，间隔几步算一次就行
    n = size(A, 1) # 矩阵大小
    H = zeros(maxiter + 1, maxiter) # 子空间投影最大大小
    Q = zeros(n, maxiter + 1) # Krylov基矢
    Q[:, 1] .= normalize(q0) # 初始化第一个基矢
    λ = zeros(ComplexF64, howmany) # 储存howmany个本征值的绝对值
    λ0 = zeros(ComplexF64, howmany) # 比较
    k_cut = 0 # 记录跳出循环的截断位置
    converge_info = false # 是否收敛
    for k in 2:maxiter+1
        k_cut = k
        λ0 .= λ
        v = @view Q[:, k]
        mul!(v, A, @view Q[:, k-1])
        for j in 1:k-1
            vj = @view Q[:, j]
            H[j, k-1] = dot(vj, v)
            v .-= H[j, k-1] .* vj
        end
        H[k, k-1] = norm(v)
        normalize!(v)
        if k > miniter && k > howmany + 2 && k % kstep == 0 # 满足这些才计算一次本征值
            # 求解本征值，按照绝对值大小进行排序
            λ .= eigvals(H[1:(k-1), 1:(k-1)], sortby=abs)[end:-1:end-howmany+1]
            if all(abs.(λ .- λ0) .< tol)
                println("Converged when k = $(k) \n λ = $(λ)")
                converge_info = true
```

```
                break
            end
        end
        if k > maxiter
            println("Probably not converged")
        end
    end
    return λ, Q[:, 1:k_cut-1], H[1:(k_cut-1), 1:(k_cut-1)], converge_info
end
```