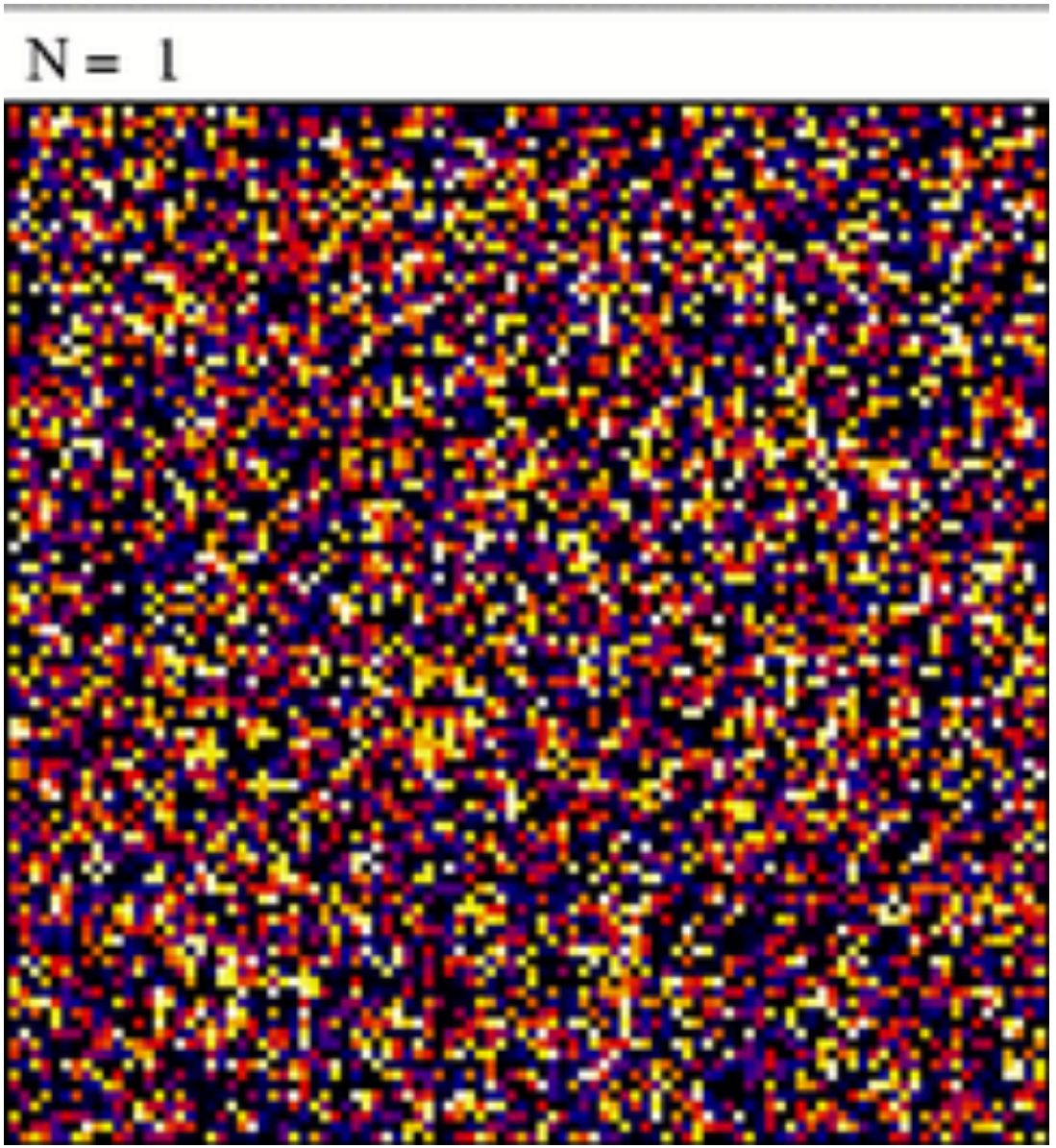
First excited state of a 100*100 box vs number of iterations (N)

Graphing:

$$|\Psi_0(x,y)|^2$$

Starting from a random state

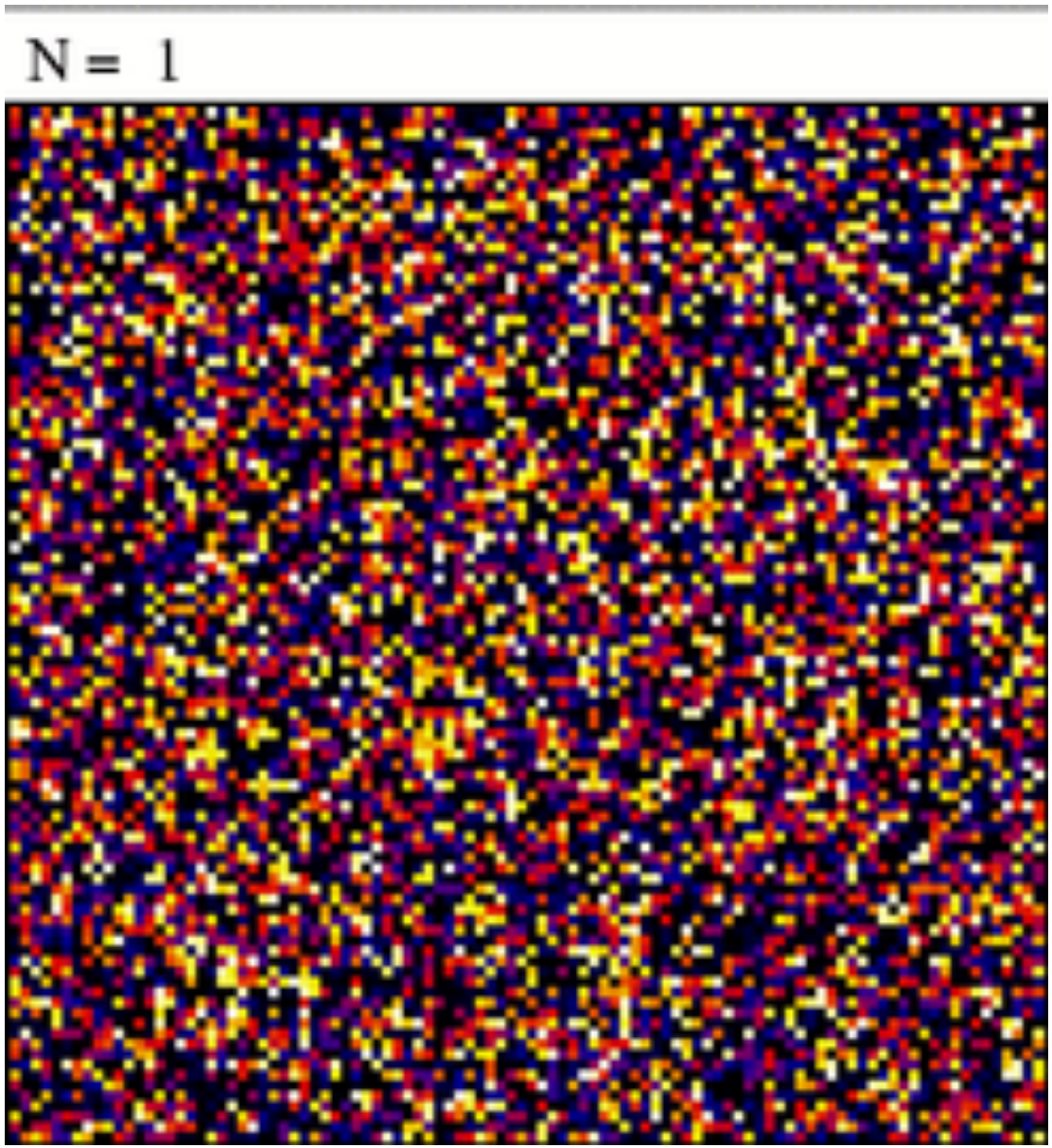


Second excited state of a 100*100 box vs number of iterations (N)

Graphing:

$|\Psi_0(x, y)|^2$

Starting from a random state



The first excited state should be doubly-degenerate

➢ Lanczos only gives one state out of a degenerate multiplet

Go back to the Krylov space

$$H^m|\Psi\rangle = \sum_k C_k E_k^m |\Psi_k\rangle$$

If states k, j are degenerate, we have a term

$$E_j^m (C_j|\Psi_j\rangle + C_k|\Psi_k\rangle)$$

For any m, this vector points in the same direction in the subspace spanned by $|\Psi_j\rangle,\ |\Psi_k\rangle$

Acting with H cannot "separate" degenerate states

Since the Lanczos basis spans the same Krylov space, we only get one state out of a degenerate multiplet of states

➢ the particular linear combination depends on the initial state

Numerical round-off errors can lead to apparent degeneracies (multiple copies of the same state). This indicates that the scheme breaks down as the basis becomes non-orthogonal.

Potential problem:

The normalization constants N_m can become very large (think of E_0^m)

Solution:

generate the normalized basis directly

- start with $|\phi_0\rangle$ arbitrary, normalized, and then

$$|\phi_1\rangle = \frac{1}{N_1}\big(H|\phi_0\rangle - a_0|\phi_0\rangle\big).$$

$$|\phi_{m+1}\rangle = \frac{1}{N_{m+1}}\big(H|\phi_m\rangle - a_m|\phi_m\rangle - N_m|\phi_{m-1}\rangle\big) = \frac{|\gamma_m\rangle}{N_{m+1}}$$

The definition of N_m is different, and no b_m:

$$\begin{aligned} a_m &= \langle\phi_m|H|\phi_m\rangle \\ N_m &= \langle\gamma_m|\gamma_m\rangle^{-1/2} \end{aligned}$$

Generate $|\gamma_m\rangle$ first, normalize to get N_{m+1}

The H-matrix is

$$\begin{aligned} \langle\phi_{m-1}|H|\phi_m\rangle &= N_m \\ \langle\phi_m|H|\phi_m\rangle &= a_m \\ \langle\phi_{m+1}|H|\phi_m\rangle &= N_{m+1} \end{aligned}$$

Example in two dimensions: box with open boundaries

Constructing $H|f_n\rangle$ (open corresponds to hard walls)

State n stored in `f1(nx*ny)`
State $H|f_n\rangle$ constructed in `f2(nx*ny)`

labeling for 4*4 elements

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

t = hopping (kinetic) matrix element

- consider hopping into all boxes j

```
subroutine hoperation(f1,f2)
f2(:)=vpot(:)*f1(:)
do j=1,nx*ny
   x=1+mod(j-1,nx)
   y=1+(j-1)/nx
   if (x.ne.1)  f2(j-1)=f2(j-1)-t*f1(j)
   if (x.ne.nx) f2(j+1)=f2(j+1)-t*f1(j)
   if (y.ne.1)  f2(j-nx)=f2(j-nx)-t*f1(j)
   if (y.ne.ny) f2(j+nx)=f2(j+nx)-t*f1(j)
enddo
```

One step in the iteration of the a and b coefficients

$$|f_1\rangle = H|f_0\rangle - a_0|f_0\rangle$$

$$|f_{n+1}\rangle = H|f_n\rangle - a_n|f_n\rangle - b_{n-1}|f_{n-1}\rangle$$

$$a_n = H_{nn}/N_n, \qquad b_{n-1} = N_n/N_{n-1}$$

here m=n+1

```
if (m==1) then
   call hoperation(f0,f1)
   aa(0)=dot_product(f0,f1)
   f1=f1-aa(0)*f0
   nn(1)=dot_product(f1,f1)
else
   call hoperation(f1,f2)
   aa(m-1)=dot_product(f1,f2)/nn(m-1)
   bb(m-2)=nn(m-1)/nn(m-2)
   f2=f2-aa(m-1)*f1-bb(m-2)*f0
   nn(m)=dot_product(f2,f2)
   f0=f1
   f1=f2
endif
```

The method of constructing the normalized states directly is very similar (program on-line)

The full basis and Hamiltonian construction

Random initial state

```
do i=1,n
   psi(i)=rand()-0.5d0
enddo
norm=1.d0/sqrt(dot_product(psi,psi))
psi(:)=psi(:)*norm
```

Perform `niter` Lanczos steps and diagonalize

```
f0(:)=psi(:)
nn(0)=1.d0
Do m=1,niter
   perform code on previous page
enddo
d(:)=aa(:)
e(:)=sqrt(bb(:))
call diatri(niter,d,e,eig,states)
```

Calculation of the states

In order to calculate states (wave functions) we have to perform another Lanczos procedure, since we have not saved all the states $|f_n\rangle$

If we want the m-th lowest state, we transform with the m-th eigenvector obtained in the diagonalization. The eigenvectors are in the matrix `states`; `vec=states(:,m)`

Normalized states $|\phi_n\rangle = N_n{}^{-1/2}|f_n\rangle$

```
f0=psi
psi=psi*vec(0)
call hoperation(n,f0,f1)
f1=f1-aa(0)*f0
psi=psi+vec(1)*f1/sqrt(nn(1))
do i=2,niter-1
   call hoperation(n,f1,f2)
   f2=f2-aa(i-1)*f1-bb(i-2)*f0
   psi=psi+vec(i)*f2/sqrt(nn(i))
   f0=f1
   f1=f2
enddo
```