UCDAVIS

Aleksander Zujev

Advisor: Professor Scalettar

Lanczos algorithm: Hard-core bosons in 1-D

- 0. Motivation
- 1. Introduction
- 2. Programming
- 3. Checks of code
- 4. Results
- 5. Future Work

0. Motivation

Lanczos algorithm is suited for calculating the system at zero temperature, when methods for finite temperature don't work.

Possible use:

• Exercise.

• Possibly people experiment with such systems - supercooled atoms trapped in 1-D lattice made of magnetic field/laser beam?

1. Introduction

Hard-core bosons in 1-D lattice

Lattice of N sites. Each site contains 0 or 1 particle.
E.g. wave function in coordinate space:
$\Psi > =$

| 0 1 1 0 1 0 1 1 >

$$H = -t \sum_{<ij>} \left(a_i^\dagger a_j + a_j^\dagger a_i \right) + V \sum_{<ij>} \hat{n}_i \hat{n}_j$$

1-D:

$$H = -t \sum_{i} \left(a_i^\dagger a_{i+1} + a_{i+1}^\dagger a_i \right) + V \sum_{i} \hat{n}_i \hat{n}_{i+1}$$

Commutation relation for hard-core bosons - I'll leave it as an exercise for listeners.

Partition function

(Ditto)

2. Programming

For lattice of N sites 2^N basis vectors. Eg N=4:

```
| 0 0 0 0 >
| 0 0 0 1 >
| 0 0 1 0 >
| 0 0 1 1 >
| 0 1 0 0 >
| 0 1 0 1 >
| 0 1 1 0 >
| 0 1 1 1 >
| 1 0 0 0 >
| 1 0 0 1 >
| 1 0 1 0 >
| 1 0 1 1 >
| 1 1 0 0 >
| 1 1 0 1 >
| 1 1 1 0 >
| 1 1 1 1 >
```

I code them literally - as binary numbers, or in decimal: 0, 1, 2, 3, ... $2^N - 1$.

3. Checks of code

• If $t = 0$, and we have only near neighbour repulsion, then for half-filled lattice we expect checker-board arrangement:

$$|\Psi_0 >= \alpha|1\ 0\ 1\ 0\ ...... > +\beta|0\ 1\ 0\ 1\ ...... >$$

Check: N=8, V=1, t=0.01:

$$|\Psi_0 >= 0.74\ |\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ > \ +\ 0.68\ |\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ >$$

Looks Ok.
But, for t = exactly zero, the result is not as good:
$|\psi_0 >= -0.86|01010101 > +0.34|01111000 > +0.23|11100001 > -0.15|00011110 >$
$+0.13|10000111 > +0.10|11000011 > +0.09|00111100 > +0.07|00001111 > +0.05|01100101 >$
$-0.05|00011101 > +0.04|01001011 > -0.04|10001011 > +0.04|10010110 > -0.04|10110001 >$
$-0.04|11100010 > +0.04|00101011 > -0.03|00100111 > +0.03|01011001 > +0.03|00110101 >$
$-0.03|10111000 > +0.03|11010010 > +0.03|01101001 > +0.03|10110100 > +0.03|01011010 >$
$-0.03|11001001 > +.....$
Should investigate this.

Also, checked for eigenvalues for $N = 2$

4. Results

$< n(k=0) >$ = number of particles with zero momentum

$$< n(k) > = < \Psi_0 | \frac{1}{N} \sum_{nl} a_n a_l^\dagger e^{ik(n-l)} | \Psi_0 >$$

$$< n(k=0) > = < \Psi_0 | \frac{1}{N} \sum_{nl} a_n a_l^\dagger | \Psi_0 >$$

For $|\Psi_0 >= \alpha |1\ 0\ 1\ 0\ ......> + \beta |0\ 1\ 0\ 1\ ......>$, it's easy to show

$$< n(k=0) > = \frac{1}{2}$$

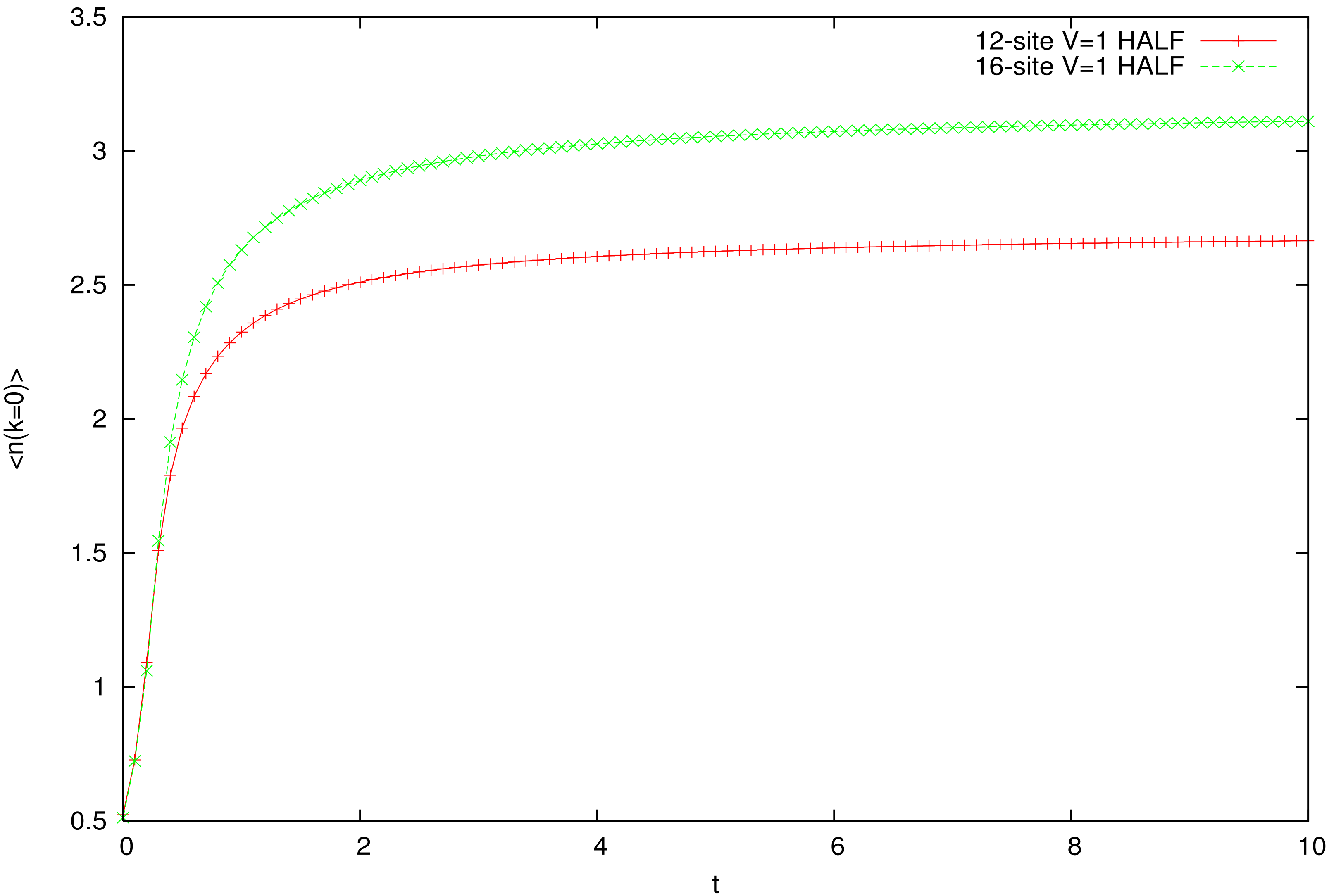

Figure 1: Plots of $\langle n(k=0)\rangle$ vs t for lattices of N=12 and N=16. For $t=0$, $\langle n(k=0)\rangle = \frac{1}{2}$, as it should be.

Number of Lanczos iterations

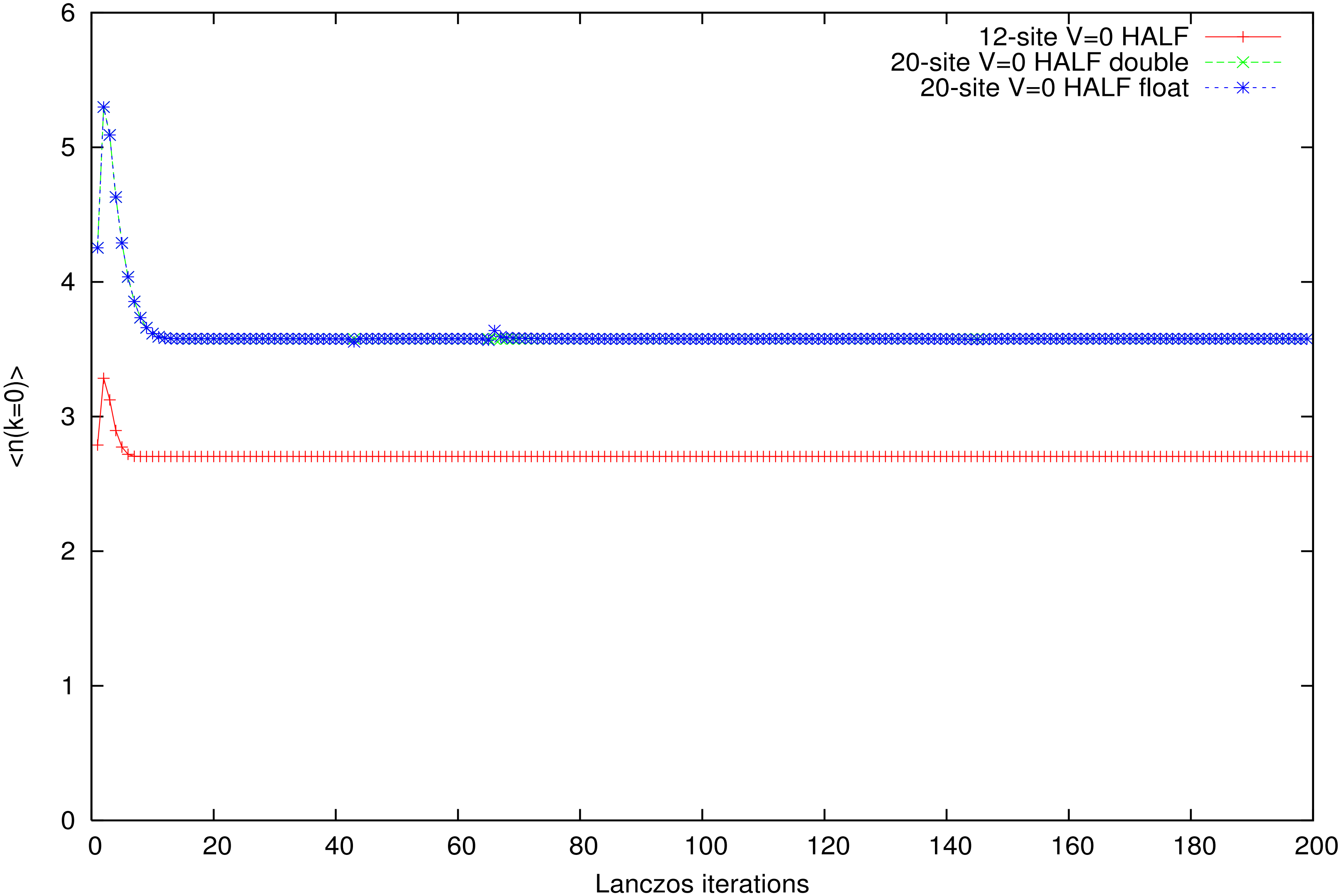

Figure 2: Plots of $\langle n(k=0) \rangle$ vs number of Lanczos iterations. Looks like about 20 iterations may be enough.

Power law at $V = 0$: $\langle n(k=0)\rangle = aN^{\alpha}$?

Calculating $\langle n(k=0)\rangle$ for $V = 0$, $N = 4, 6, ...22, 24$, and power x as $n(N+2) = n(N)((N+2)/N)^x$:

```
*a=4*n(a)=1.307409**b=6*n(b)=1.832614**x=0.83286092963792б**
*a=6*n(a)=1.832614**b=8*n(b)=2.156479**x=0.56567110606112 4**
*a=8*n(a)=2.156479**b=10*n(b)=2.443424**x=0.55983484409717б**
*a=10*n(a)=2.443424**b=12*n(b)=2.703882**x=0.55554694484911 7**
*a=12*n(a)=2.703882**b=14*n(b)=2.943963**x=0.55185040230082 4**
*a=14*n(a)=2.943963**b=16*n(b)=3.167759**x=0.54869318392636 7**
*a=16*n(a)=3.167759**b=18*n(b)=3.378167**x=0.54599422163043 1**
*a=18*n(a)=3.378167**b=20*n(b)=3.577318**x=0.54365814917564 7**
*a=20*n(a)=3.577318**b=22*n(b)=3.766837**x=0.54162419726112 1**
*a=22*n(a)=3.766837**b=24*n(b)=3.947988**x=0.53981934789817 4**
```

x (or α) is close to $1/2$. Accidentally, or asymptotically $\to$ $1/2$ at $N \to \infty$?

5. Future Work

- Find in literature or calculate analytically $< n(k=0) > (N)$, $(V=0$, half-filling).

- Programming: For half-filling, use basis economically (only corresponding subset of full basis).

- Calculate deviation, error-bar, etc.

-

- Program some other problems using Lanczos algorithm.

-