Exact diagonalization studies of quantum lattice models

Zlatko Papic

Institute of Physics, Belgrade, 8/10/2008

Exact diagonalization

- Start from Schroedinger eq.
  $H\Phi_n = E_n\Phi_n$
- Choose basis:
  $\{\Psi_i\}_{i=1}^N$

$$\left.\begin{array}{l} H\Phi_n = E_n\Phi_n \\ \{\Psi_i\}_{i=1}^N \end{array}\right\} H \to (H_{ij})$$

- for quantum lattice models, H is:
  - always hermitian
  - can be even symmetric (real)
  - hopefully sparse – O(N) nonzero

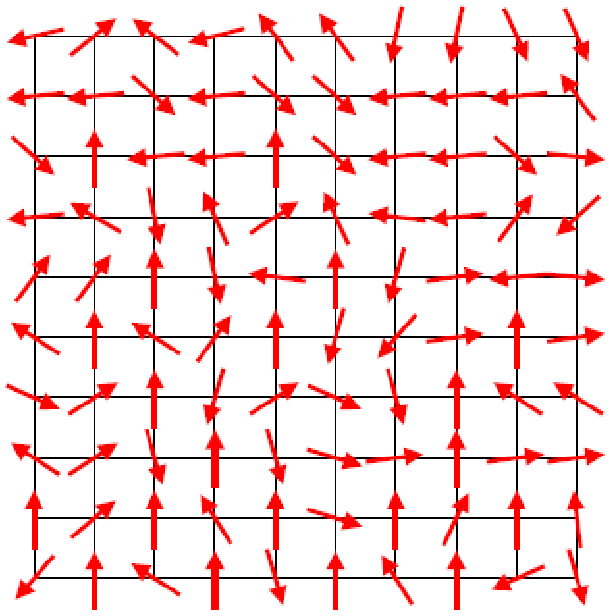
ED procedure

- choose initial basis (in Fock space)
- If possible find better one employing symmetries
- numerical/virtual representation of H
- find eigenvalues/eigenvectors
- calculate observable's expectation values etc.

Exponential barrier

- Heisenberg model S=1/2

$$H = -\sum_{<i,j>} J_{ij} \vec{S}_i \vec{S} j$$



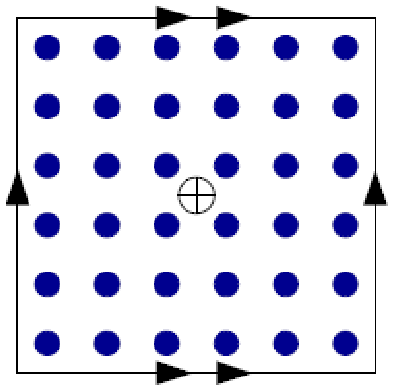
Hilbert space dim:

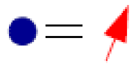
$٢^{١٠\times١٠}$!

Hard wall:
S=1/2 systems ~40 sites
Hubbard at half filling ~20

We can do a little better still...

$$H = -\sum_{<i,j>} J_{ij}\vec{S}_i\vec{S}_j$$





- Full Hilbert space:

$$dim = 2^{36} \sim 70 \cdot 10^9$$

- Symmetry Sz=0:

$$dim = \frac{36!}{18!18!} \sim 9 \cdot 10^9$$

- Spin inversion symmetry

$$dim = \frac{1}{2}\frac{36!}{18!18!}$$

- Space group symmetries (translation, rotation)

$$dim \sim \frac{1}{2}\frac{1}{36{\cdot}4}\frac{36!}{18!18!} \sim 30 \cdot 10^6$$

Gain 2500!

Why use ED then?

- Robust, unbiased and completely versatile – almost anything can be calculated!
- There are models which are not easy to access via other models (e.g. frustrated magnets)
- Error is at least as low as 10^(-14) – numerical precision
- Exploiting symmetries reduces computational effort and gives physical information about eigenstates (good quantum numbers)

Example

$$|\sigma_1, \ldots, \sigma_N >$$

$$Integer = \sum_{k=0}^{N-1} i_k 2^k$$

$$G = TG \times PG$$

$$\{\Psi_{sym}\}$$

$$|1\rangle := |\uparrow\uparrow\downarrow\rangle \qquad |3\rangle := |\downarrow\downarrow\uparrow\rangle$$
$$|2\rangle := |\uparrow\uparrow\uparrow\rangle \qquad |4\rangle := |\downarrow\downarrow\downarrow\rangle$$

$$|5\rangle := |\downarrow\uparrow\downarrow\rangle \qquad |7\rangle := |\uparrow\downarrow\uparrow\rangle$$
$$|6\rangle := |\uparrow\downarrow\downarrow\rangle \qquad |8\rangle := |\downarrow\uparrow\uparrow\rangle$$

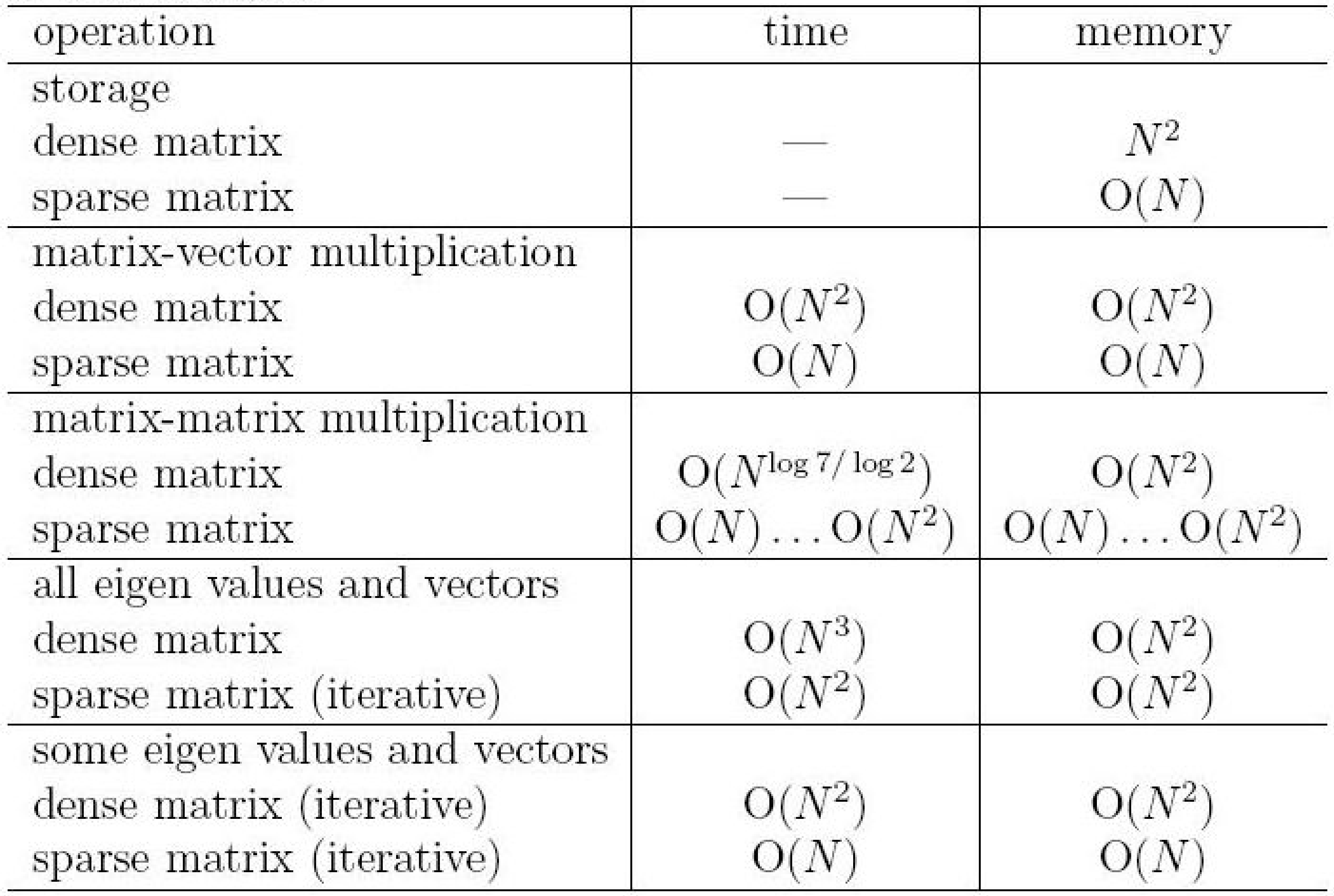
$$H_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \qquad H_0 = \begin{pmatrix} \begin{pmatrix} 0 & 0 & -\sqrt{2} \\ 0 & -1 & 0 \\ -\sqrt{2} & 0 & 0 \end{pmatrix} & & \\ & \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & \\ & & \begin{pmatrix} 0 & 0 & \sqrt{2} \\ 0 & -1 & 0 \\ \sqrt{2} & 0 & 1 \end{pmatrix} \end{pmatrix}$$

Diagonalization routine

- If H is dense or system small enough – use Jacobi, Householder, LAPACK…
  (all these apply **orthogonal transformations** to H until tridiagonal form, then quickly diagonalize)
- If H is sparse – use ARPACK, IETL/ALPS, DiagHam
  (these are **iterative** solvers based on variants of **Lanczos algorithm** which preserves the sparseness of H)

Dense vs. sparse

Table 7: Time and memory complexity for operations on sparse and dense $N \times N$ matrices



operation	time	memory
storage		
dense matrix	—	N^2
sparse matrix	—	$O(N)$
matrix-vector multiplication		
dense matrix	$O(N^2)$	$O(N^2)$
sparse matrix	$O(N)$	$O(N)$
matrix-matrix multiplication		
dense matrix	$O(N^{\log 7/\log 2})$	$O(N^2)$
sparse matrix	$O(N) \ldots O(N^2)$	$O(N) \ldots O(N^2)$
all eigen values and vectors		
dense matrix	$O(N^3)$	$O(N^2)$
sparse matrix (iterative)	$O(N^2)$	$O(N^2)$
some eigen values and vectors		
dense matrix (iterative)	$O(N^2)$	$O(N^2)$
sparse matrix (iterative)	$O(N)$	$O(N)$

Lanczos iterations

1. **Starting conditions:**
   $|U_1\rangle := |V\rangle$ with V random starting vector
   $\||U_1\rangle\| = 1 \ , \ |U_0\rangle = 0 \ , \ b_0 = 1, \ , \ k = 0$

2. **Iteration, building the T-matrix**

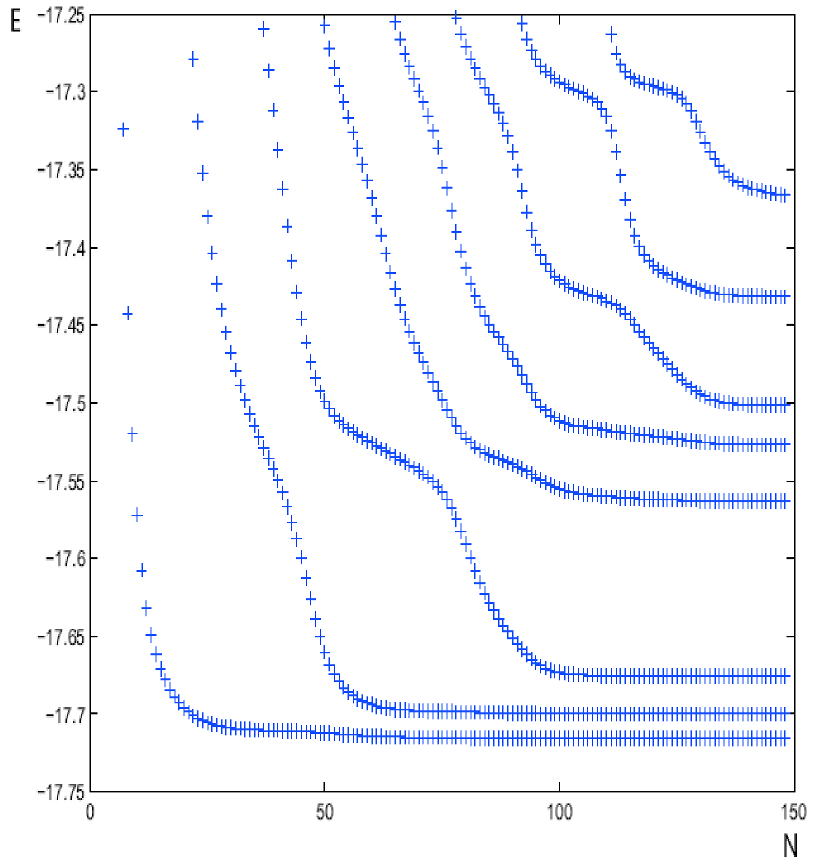
```
while($b_k \neq 0$)
$|U_{k+1}\rangle = |r_k\rangle / b_k; k = k + 1; a_k = \langle U_k|H|U_k\rangle$
$|r_k\rangle = H|U_k\rangle - a_k|U_k\rangle - b_{k-1}|U_{k-1}\rangle;$
$b_k = \||r_k\rangle\|_2$
end
```

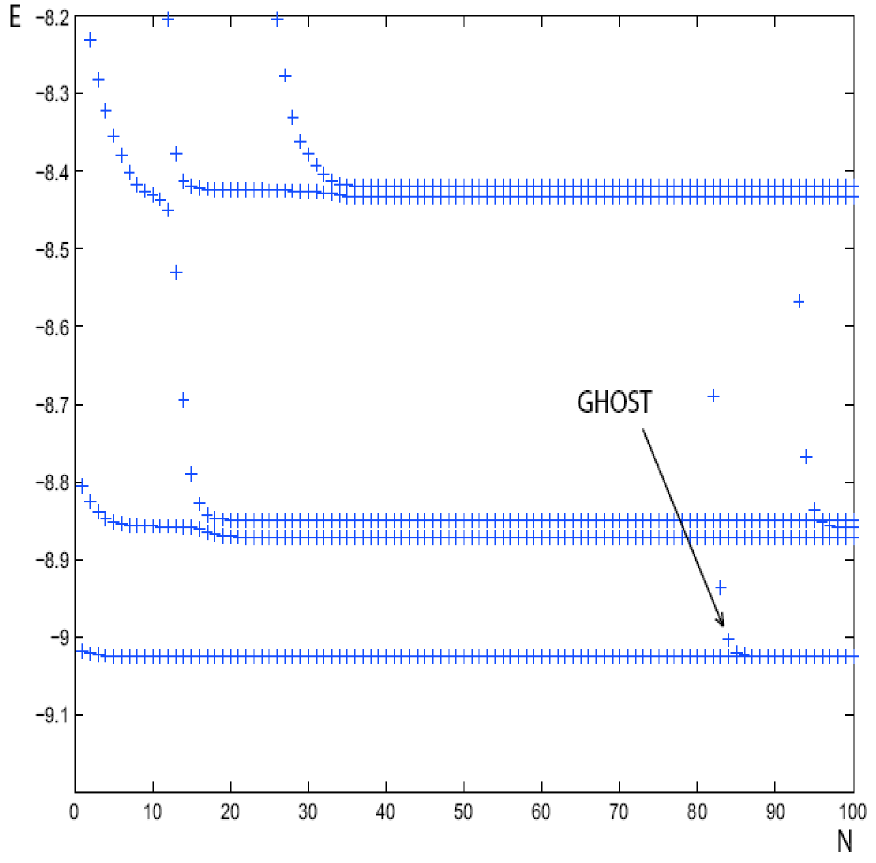
$$T = \begin{pmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & & \ddots & & \\ & & & & \ddots & b_{m-1} \\ & & & & b_{m-1} & a_m \end{pmatrix}$$

Why use Lanczos?

- Because for large scale problems there is nothing else!
- Lanczos is **fast** – sometimes as few as 100 iterations are enough to get groundstate with precision 10^(-8)!
- Memory requirements are **low** – need to store 2-4 vectors only !
  Matrix in principle need not be stored, only action H*|vec> is required

Lanczos problems

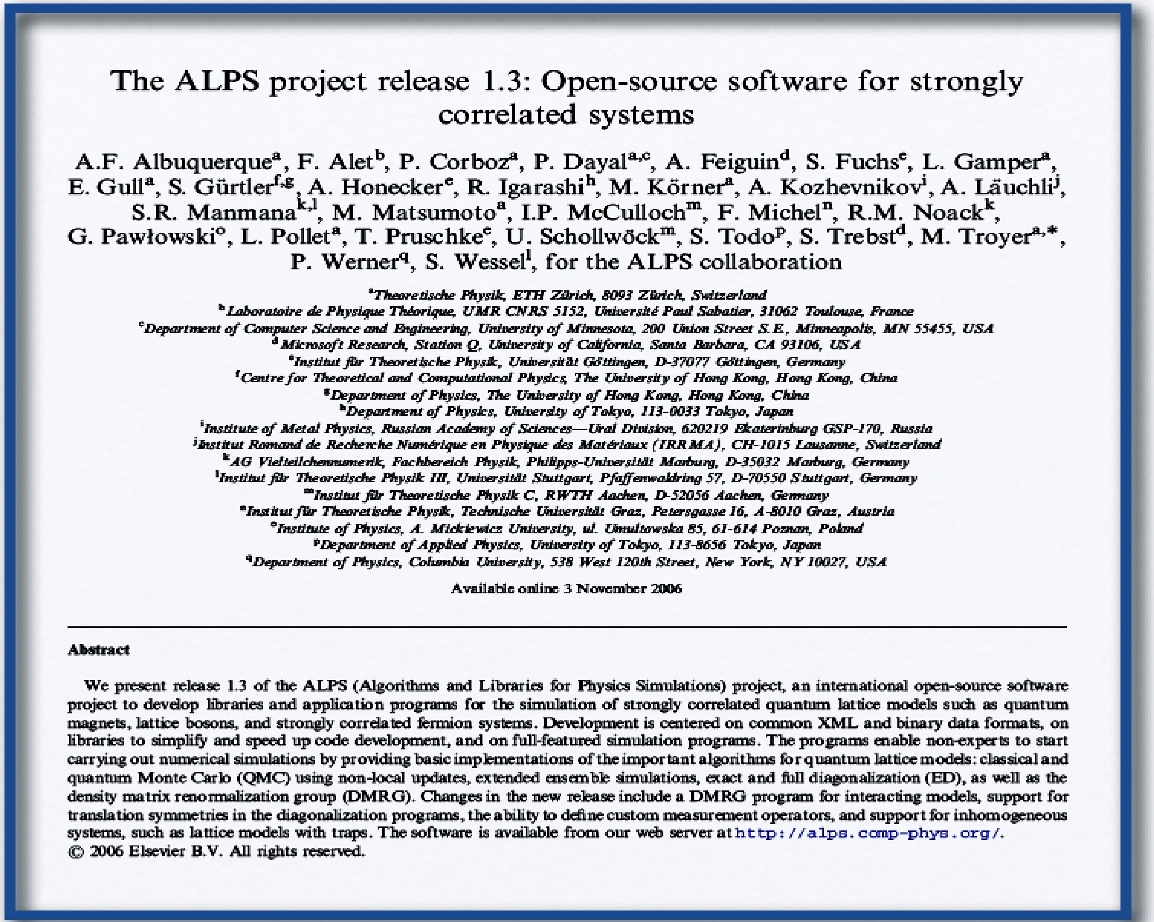




Main piece of wisdom

- DO NOT start writing your own code from the scratch (unless really forced to)

ALPS = Algorithms and Libraries for Physics Simulations



The ALPS project release 1.3: Open-source software for strongly correlated systems

A.F. Albuquerque[a], F. Alet[b], P. Corboz[a], P. Dayal[a,c], A. Feiguin[d], S. Fuchs[e], L. Gamper[a], E. Gull[a], S. Gürtler[f,g], A. Honecker[e], R. Igarashi[h], M. Körner[a], A. Kozhevnikov[i], A. Läuchli[j], S.R. Manmana[k,l], M. Matsumoto[a], I.P. McCulloch[m], F. Michel[n], R.M. Noack[k], G. Pawłowski[o], L. Pollet[a], T. Pruschke[e], U. Schollwöck[m], S. Todo[p], S. Trebst[d], M. Troyer[a,*], P. Werner[q], S. Wessel[l], for the ALPS collaboration

[a] Theoretische Physik, ETH Zürich, 8093 Zürich, Switzerland
[b] Laboratoire de Physique Théorique, UMR CNRS 5152, Université Paul Sabatier, 31062 Toulouse, France
[c] Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, USA
[d] Microsoft Research, Station Q, University of California, Santa Barbara, CA 93106, USA
[e] Institut für Theoretische Physik, Universität Göttingen, D-37077 Göttingen, Germany
[f] Centre for Theoretical and Computational Physics, The University of Hong Kong, Hong Kong, China
[g] Department of Physics, The University of Hong Kong, Hong Kong, China
[h] Department of Physics, University of Tokyo, 113-0033 Tokyo, Japan
[i] Institute of Metal Physics, Russian Academy of Sciences—Ural Division, 620219 Ekaterinburg GSP-170, Russia
[j] Institut Romand de Recherche Numérique en Physique des Matériaux (IRRMA), CH-1015 Lausanne, Switzerland
[k] AG Vielteilchennumerik, Fachbereich Physik, Philipps-Universität Marburg, D-35032 Marburg, Germany
[l] Institut für Theoretische Physik III, Universität Stuttgart, Pfaffenwaldring 57, D-70550 Stuttgart, Germany
[m] Institut für Theoretische Physik C, RWTH Aachen, D-52056 Aachen, Germany
[n] Institut für Theoretische Physik, Technische Universität Graz, Petersgasse 16, A-8010 Graz, Austria
[o] Institute of Physics, A. Mickiewicz University, ul. Umultowska 85, 61-614 Poznan, Poland
[p] Department of Applied Physics, University of Tokyo, 113-8656 Tokyo, Japan
[q] Department of Physics, Columbia University, 538 West 120th Street, New York, NY 10027, USA

Available online 3 November 2006

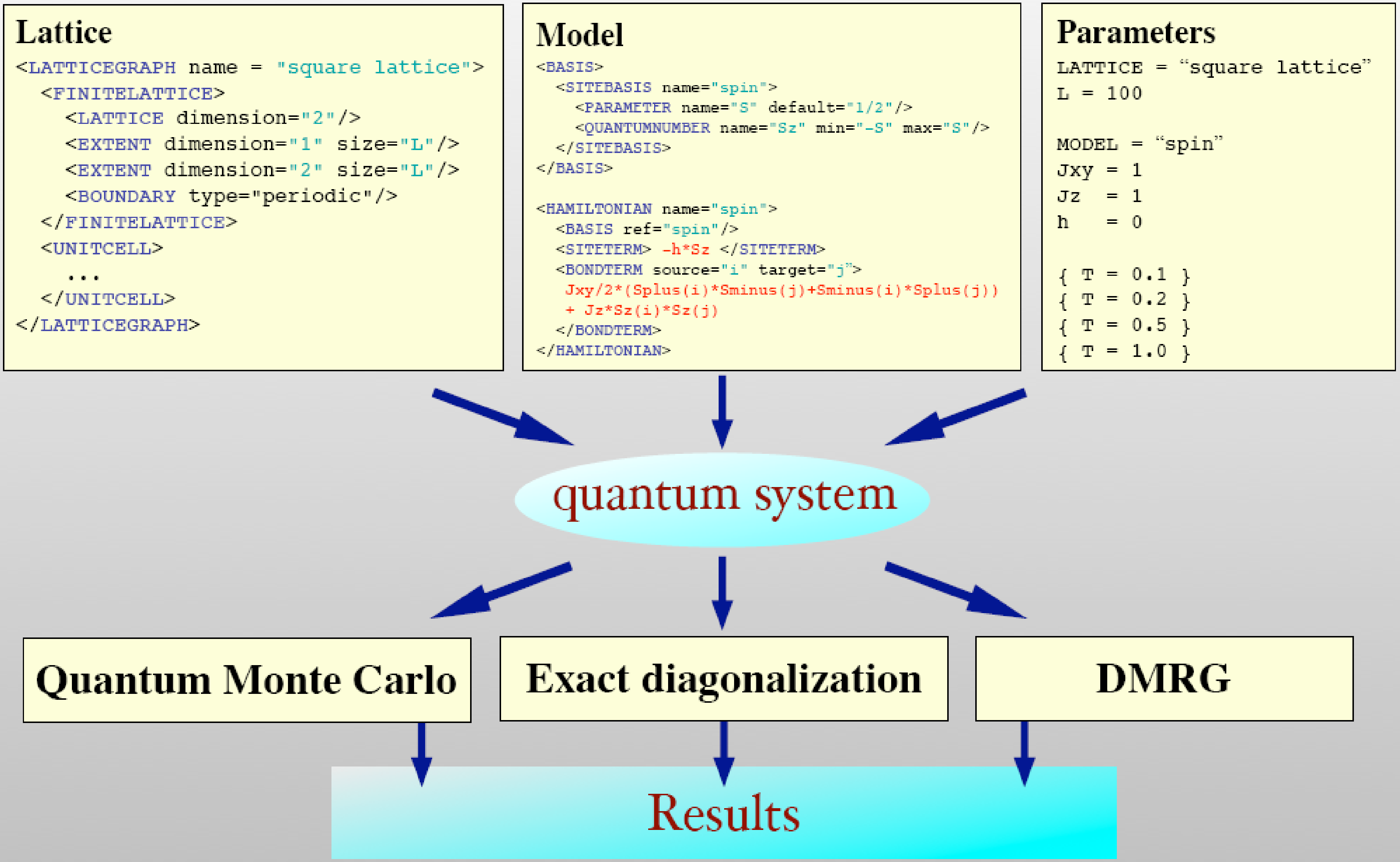
Abstract

We present release 1.3 of the ALPS (Algorithms and Libraries for Physics Simulations) project, an international open-source software project to develop libraries and application programs for the simulation of strongly correlated quantum lattice models such as quantum magnets, lattice bosons, and strongly correlated fermion systems. Development is centered on common XML and binary data formats, on libraries to simplify and speed up code development, and on full-featured simulation programs. The programs enable non-experts to start carrying out numerical simulations by providing basic implementations of the important algorithms for quantum lattice models: classical and quantum Monte Carlo (QMC) using non-local updates, extended ensemble simulations, exact and full diagonalization (ED), as well as the density matrix renormalization group (DMRG). Changes in the new release include a DMRG program for interacting models, support for translation symmetries in the diagonalization programs, the ability to define custom measurement operators, and support for inhomogeneous systems, such as lattice models with traps. The software is available from our web server at http://alps.comp-phys.org/.

© 2006 Elsevier B.V. All rights reserved.

JPSJ Suppl. **74**, 30 (2005) [cond-mat/0410407]
J. Mag. Mag. Mat. **310**, 1187 (2007)

Simulations with ALPS



Lattice

```
<LATTICEGRAPH name = "square lattice">
  <FINITELATTICE>
    <LATTICE dimension="2"/>
    <EXTENT dimension="1" size="L"/>
    <EXTENT dimension="2" size="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL>
    ...
  </UNITCELL>
</LATTICEGRAPH>
```

Model

```
<BASIS>
  <SITEBASIS name="spin">
    <PARAMETER name="S" default="1/2"/>
    <QUANTUMNUMBER name="Sz" min="-S" max="S"/>
  </SITEBASIS>
</BASIS>

<HAMILTONIAN name="spin">
  <BASIS ref="spin"/>
  <SITETERM> -h*Sz </SITETERM>
  <BONDTERM source="i" target="j">
   Jxy/2*(Splus(i)*Sminus(j)+Sminus(i)*Splus(j))
   + Jz*Sz(i)*Sz(j)
  </BONDTERM>
</HAMILTONIAN>
```

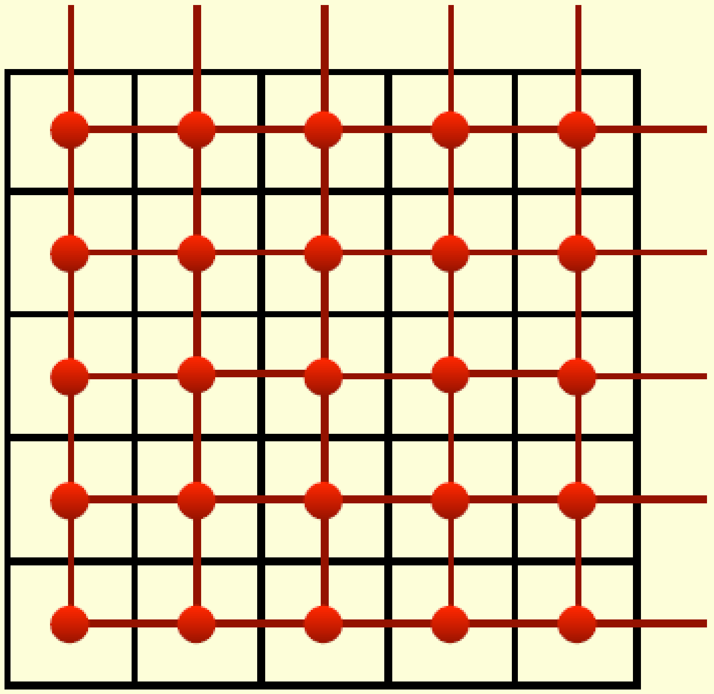
Parameters

```
LATTICE = “square lattice”
L = 100

MODEL = “spin”
Jxy = 1
Jz  = 1
h   = 0

{ T = 0.1 }
{ T = 0.2 }
{ T = 0.5 }
{ T = 1.0 }
```

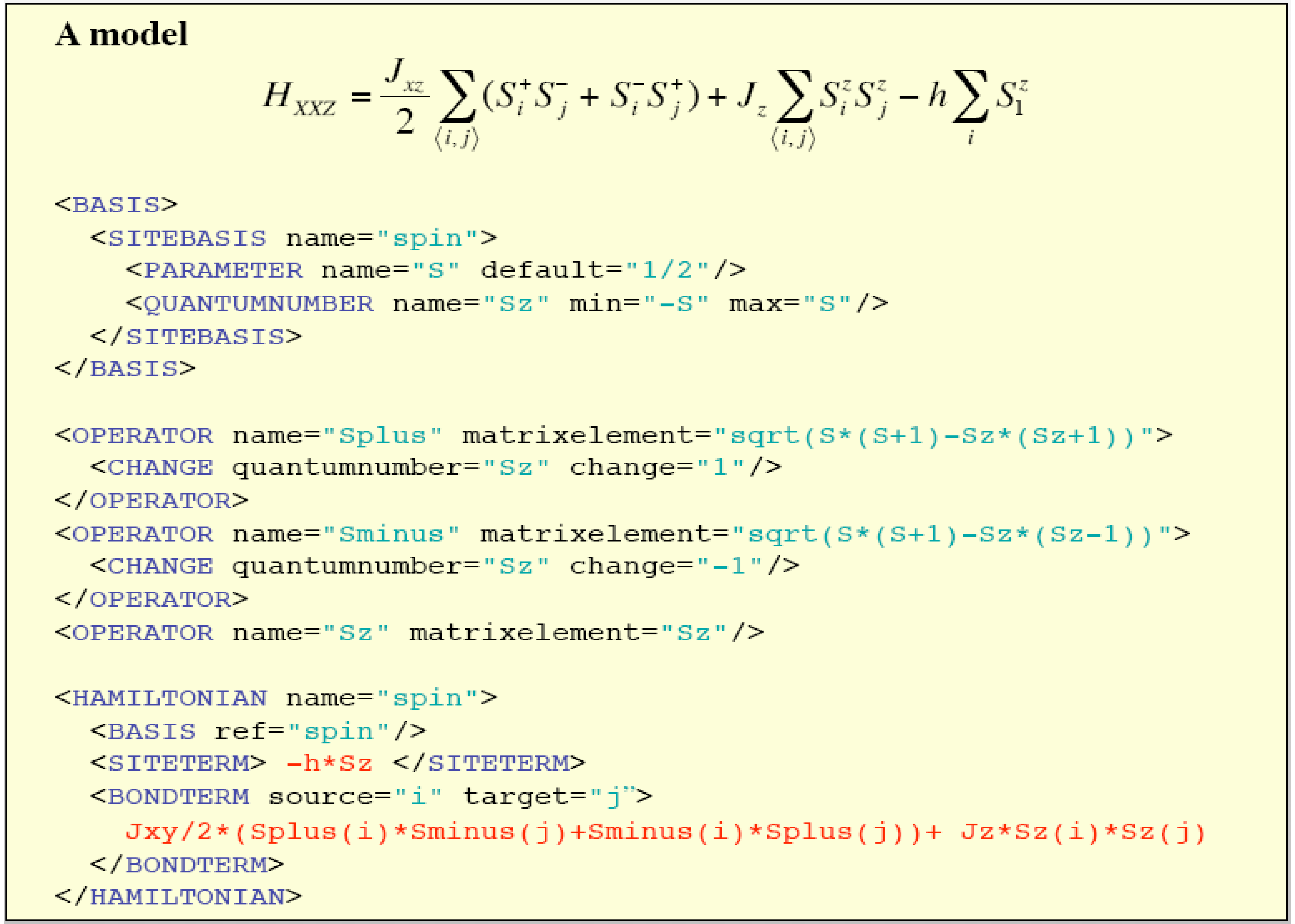
Lattice

A lattice

```
<LATTICEGRAPH name = "square lattice">
  <FINITELATTICE>
    <LATTICE dimension="2"/>
    <EXTENT dimension="1" size="L"/>
    <EXTENT dimension="2" size="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL>
    <VERTEX/>
    <EDGE type="1">
      <SOURCE vertex="1" offset="0 0"/>
      <TARGET vertex="1" offset="0 1"/>
    </EDGE>
    <EDGE type="2">
      <SOURCE vertex="1" offset="0 0"/>
      <TARGET vertex="1" offset="1 0"/>
    </EDGE>
  </UNITCELL>
</LATTICEGRAPH>
```



Model

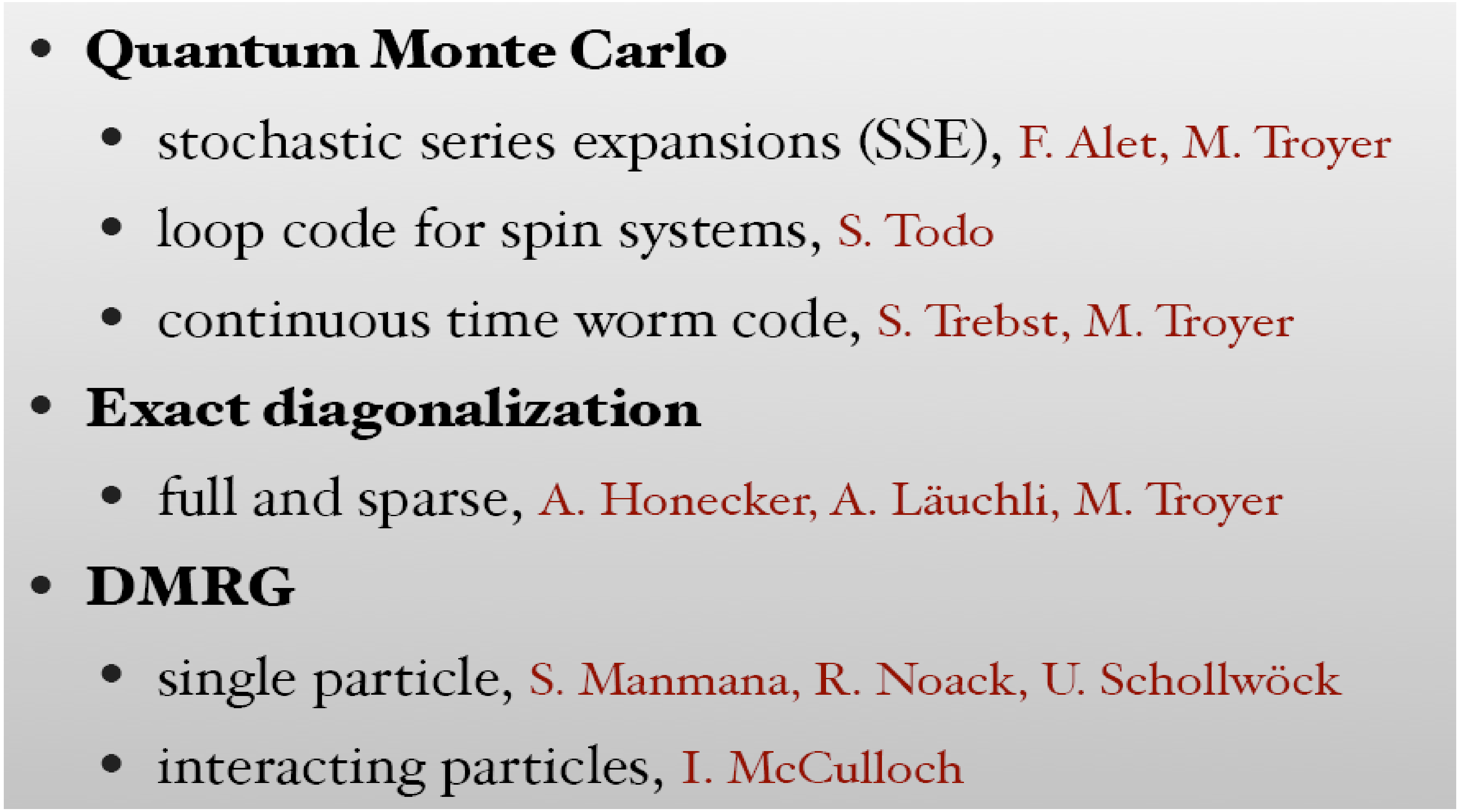


A model

$$H_{XXZ} = \frac{J_{xz}}{2}\sum_{\langle i,j\rangle}(S_i^+S_j^- + S_i^-S_j^+) + J_z\sum_{\langle i,j\rangle}S_i^zS_j^z - h\sum_i S_1^z$$

```
<BASIS>
  <SITEBASIS name="spin">
    <PARAMETER name="S" default="1/2"/>
    <QUANTUMNUMBER name="Sz" min="-S" max="S"/>
  </SITEBASIS>
</BASIS>

<OPERATOR name="Splus" matrixelement="sqrt(S*(S+1)-Sz*(Sz+1))">
  <CHANGE quantumnumber="Sz" change="1"/>
</OPERATOR>
<OPERATOR name="Sminus" matrixelement="sqrt(S*(S+1)-Sz*(Sz-1))">
  <CHANGE quantumnumber="Sz" change="-1"/>
</OPERATOR>
<OPERATOR name="Sz" matrixelement="Sz"/>

<HAMILTONIAN name="spin">
  <BASIS ref="spin"/>
  <SITETERM> -h*Sz </SITETERM>
  <BONDTERM source="i" target="j">
    Jxy/2*(Splus(i)*Sminus(j)+Sminus(i)*Splus(j))+ Jz*Sz(i)*Sz(j)
  </BONDTERM>
</HAMILTONIAN>
```

Utilities

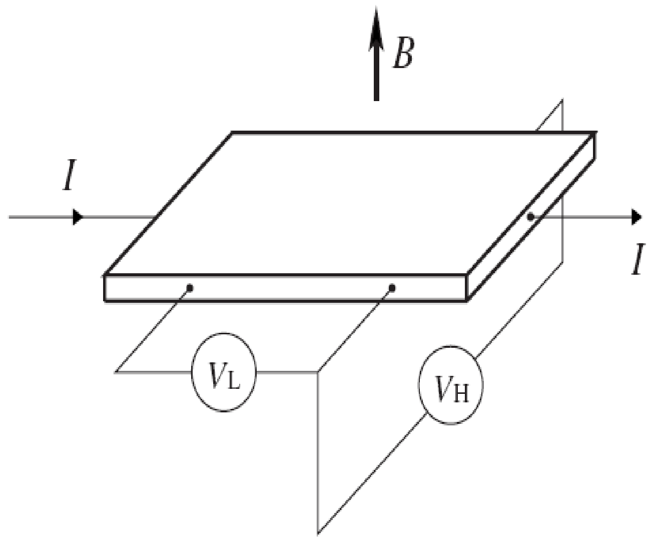


- **Quantum Monte Carlo**
  - stochastic series expansions (SSE), F. Alet, M. Troyer
  - loop code for spin systems, S. Todo
  - continuous time worm code, S. Trebst, M. Troyer
- **Exact diagonalization**
  - full and sparse, A. Honecker, A. Läuchli, M. Troyer
- **DMRG**
  - single particle, S. Manmana, R. Noack, U. Schollwöck
  - interacting particles, I. McCulloch

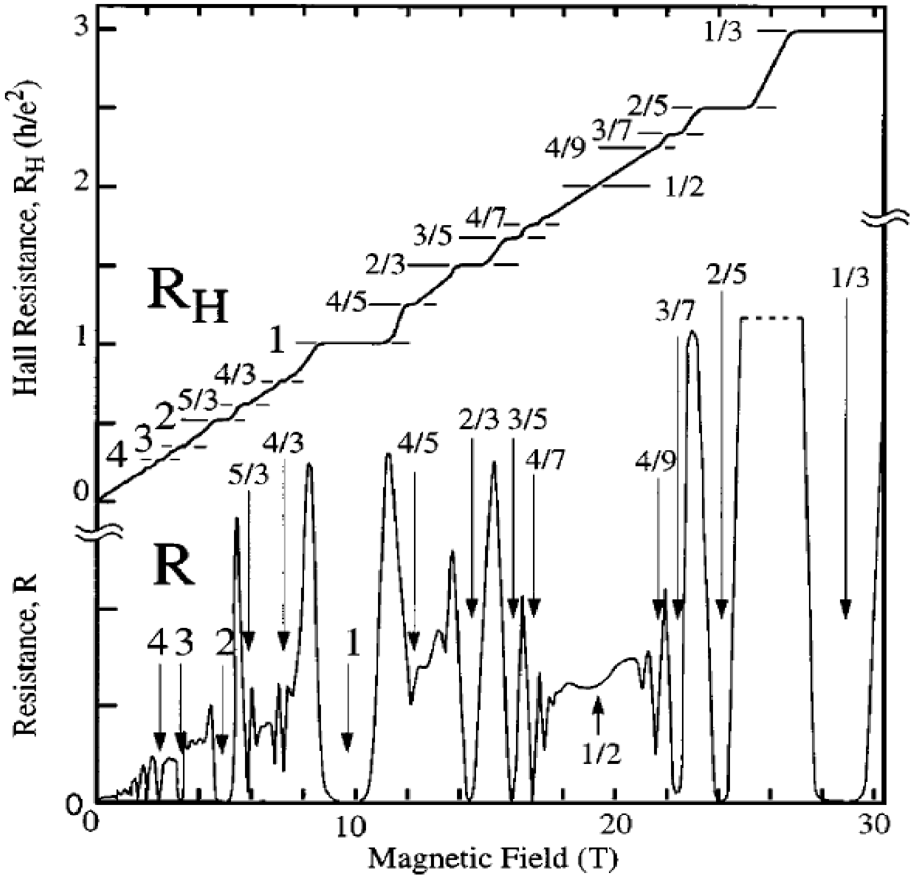
When use the ALPS?

- For complicated lattices – many possibilities for abstract implementation of symmetries
- Whenever Hamiltonian is sufficiently simple i.e. short range – this precludes Fractional Quantum Hall Effect

Fractional Quantum Hall Effect



$$\nu = \frac{N_{electrons}}{N_{fluxquanta}}$$



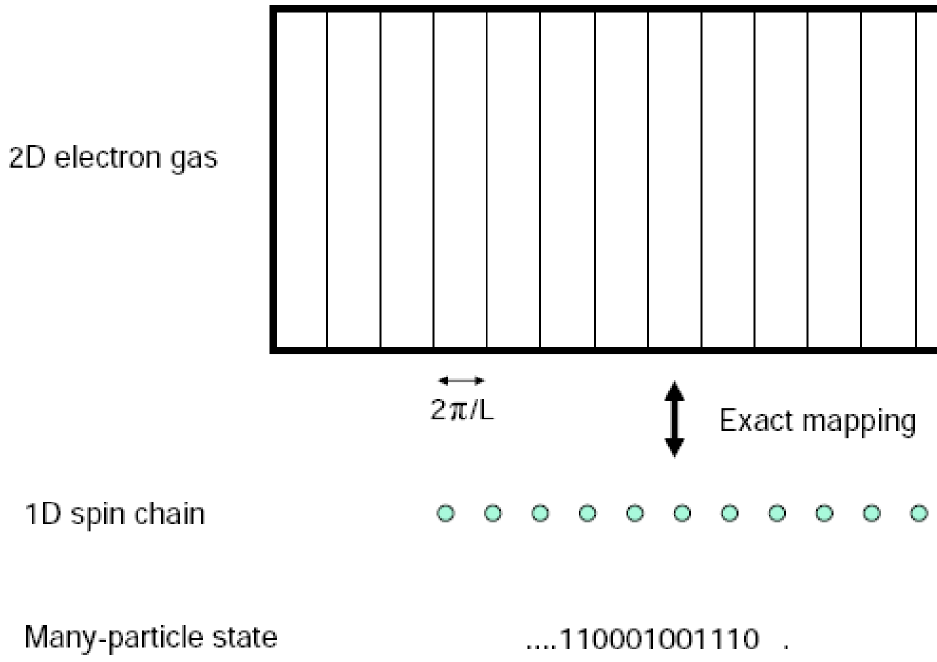
Single particle – Landau levels

$$H = \frac{1}{2m}(\vec{p} + \frac{e}{c}\vec{A})^2, \nabla A = B\vec{e}_z$$

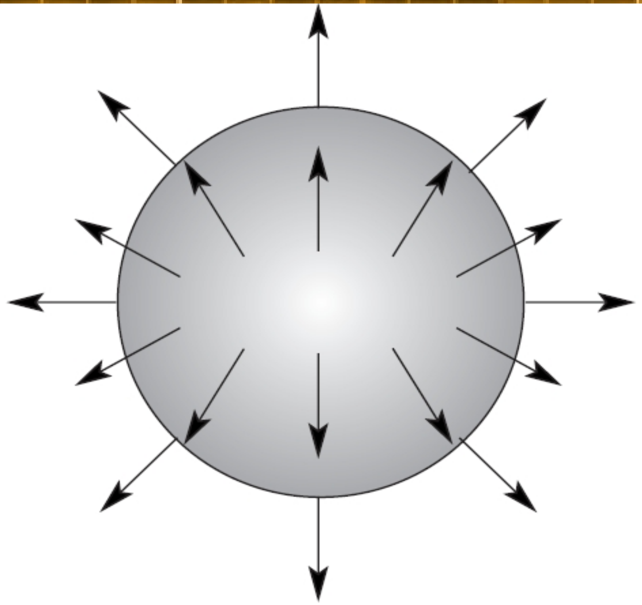
- Landau gauge

$$\vec{A} = B(-y, 0, 0) \qquad \Psi \sim \exp^{-(y-k_x)^{٢}/٢} H_n(y - k_x)$$

$$E_n = \hbar\omega_C(n + \frac{١}{٢})$$



FQHE - Sphere and Torus

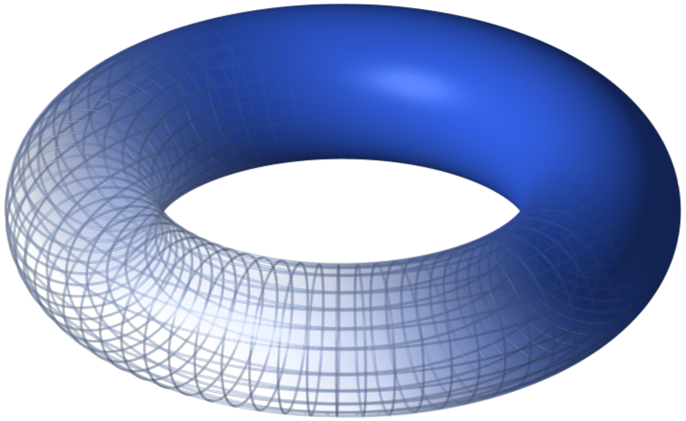


$$\nu = \lim_{N\to\infty} \frac{N}{2Q}$$

$$l = |Q|, |Q|+1, ...; m = -l, ..., l$$

Degeneracy of LLs = 2l +1

Diagonalize in invariant subspace of $\vec{L}^2, L_z, \vec{S}^2, S_z$



Must use magnetic translation symmetry and their projective reps

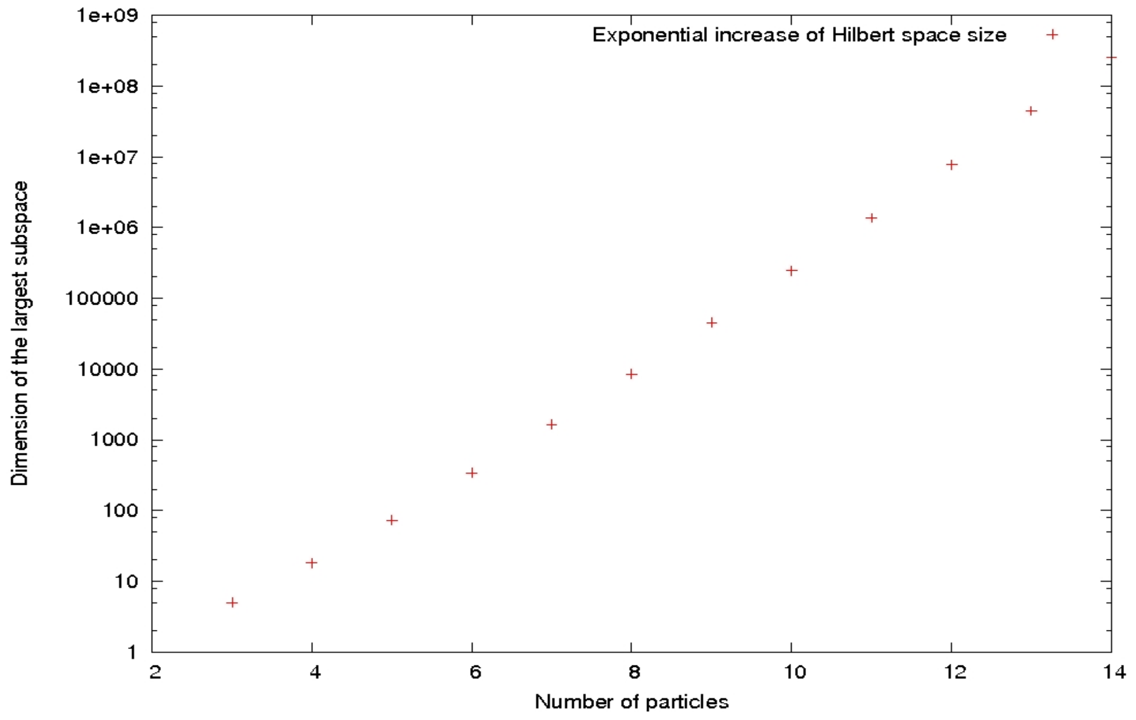
DiagHam

- Nicolas Regnault, ENS Paris
  http://www.phys.ens.fr/~regnault/
- Set of C++ for exact diagonalization; in-built programs for
  Spin Chains   Quantum Dots   FQHE
- Available from Subversion with GNU license

```
>svn checkout https://www.nick-ux.org/diagham/svn/trunk DiagHam
>./configure [options]
>make
```

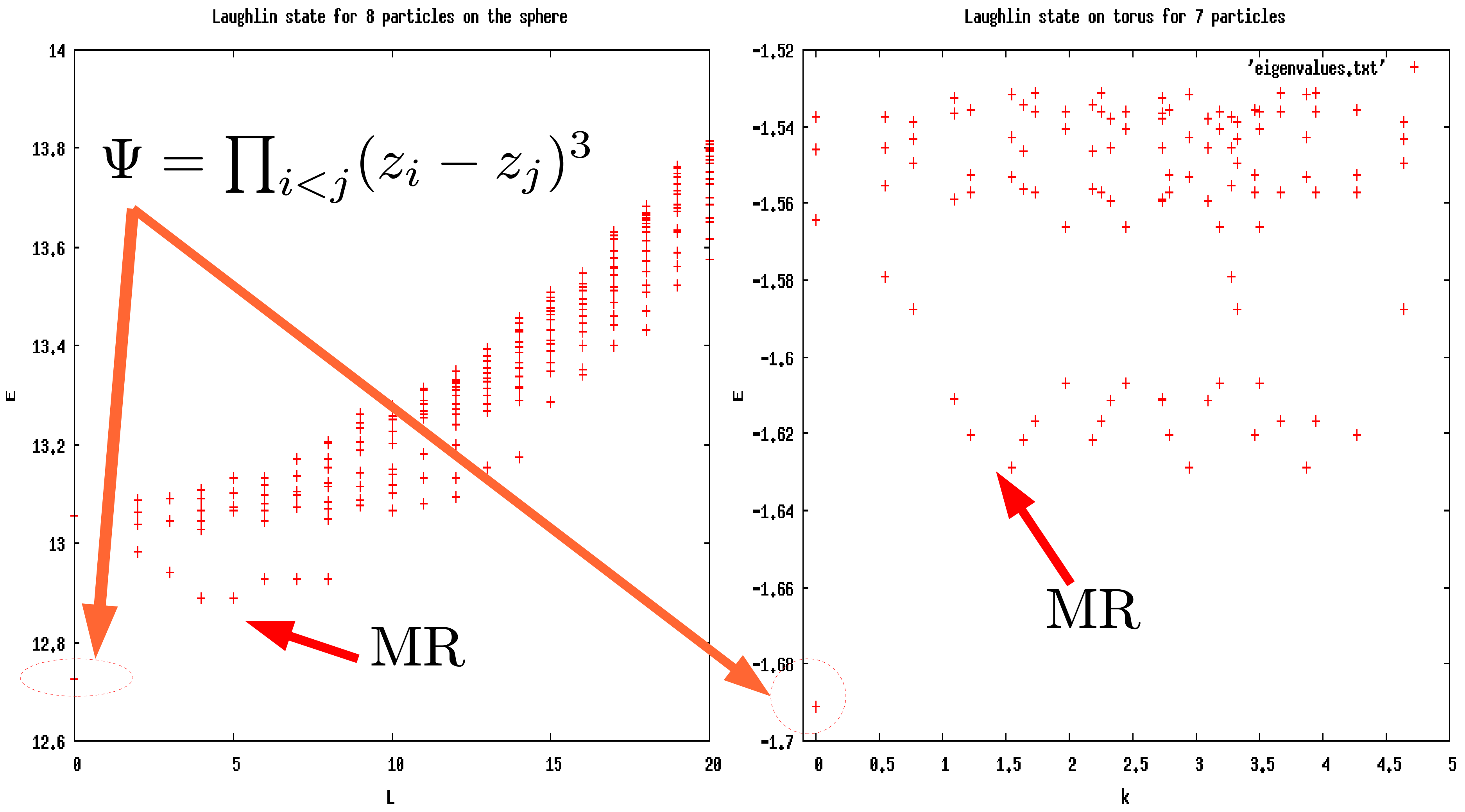
DiagHam

- It is able to handle spaces of dimension 10^8 for FQHE on Sphere
- As long as Hilbert space and Hamiltonian can be defined, DiagHam can solve the model

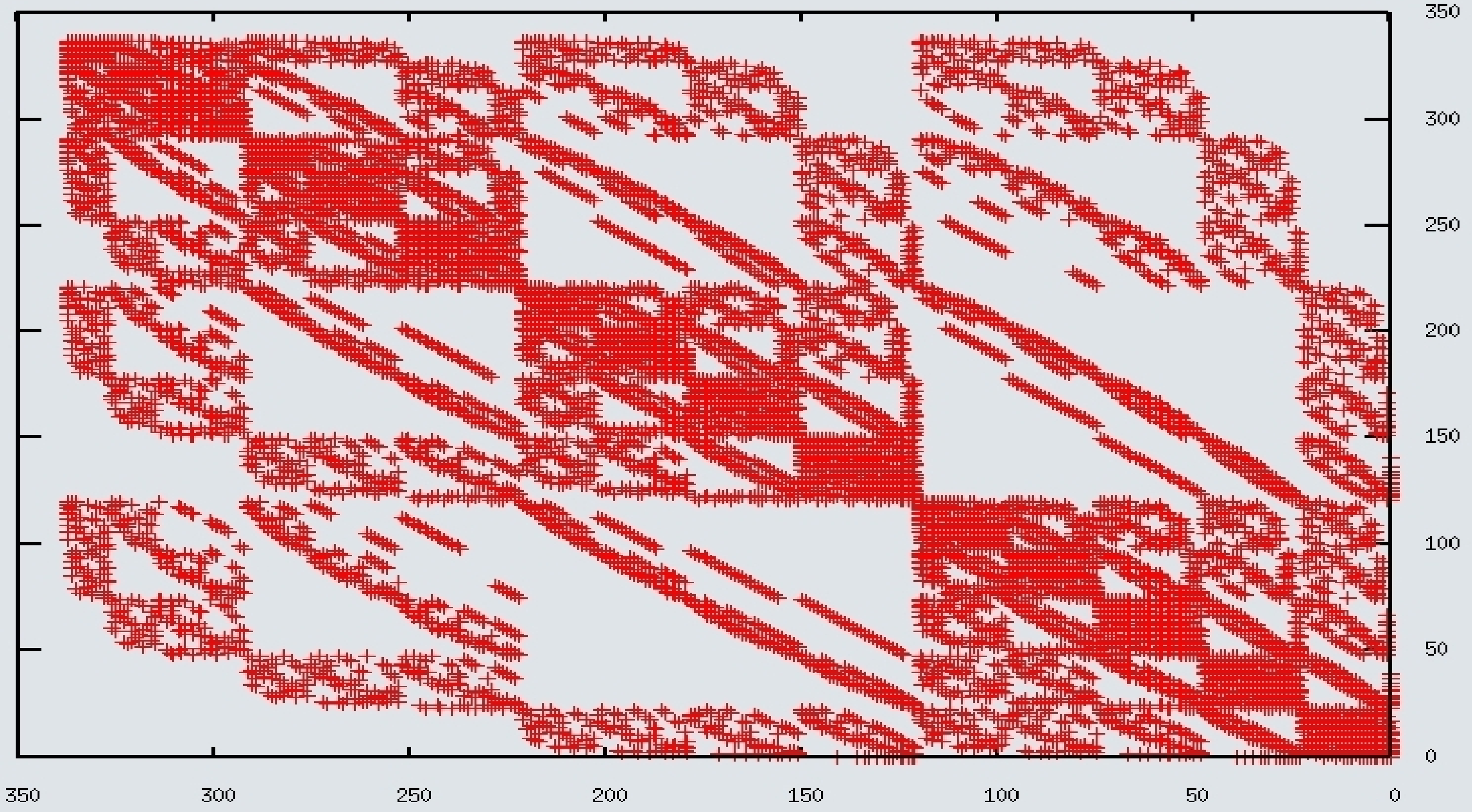


Example: Laughlin problem

$$\Psi = \prod_{i<j}(z_i - z_j)^3$$



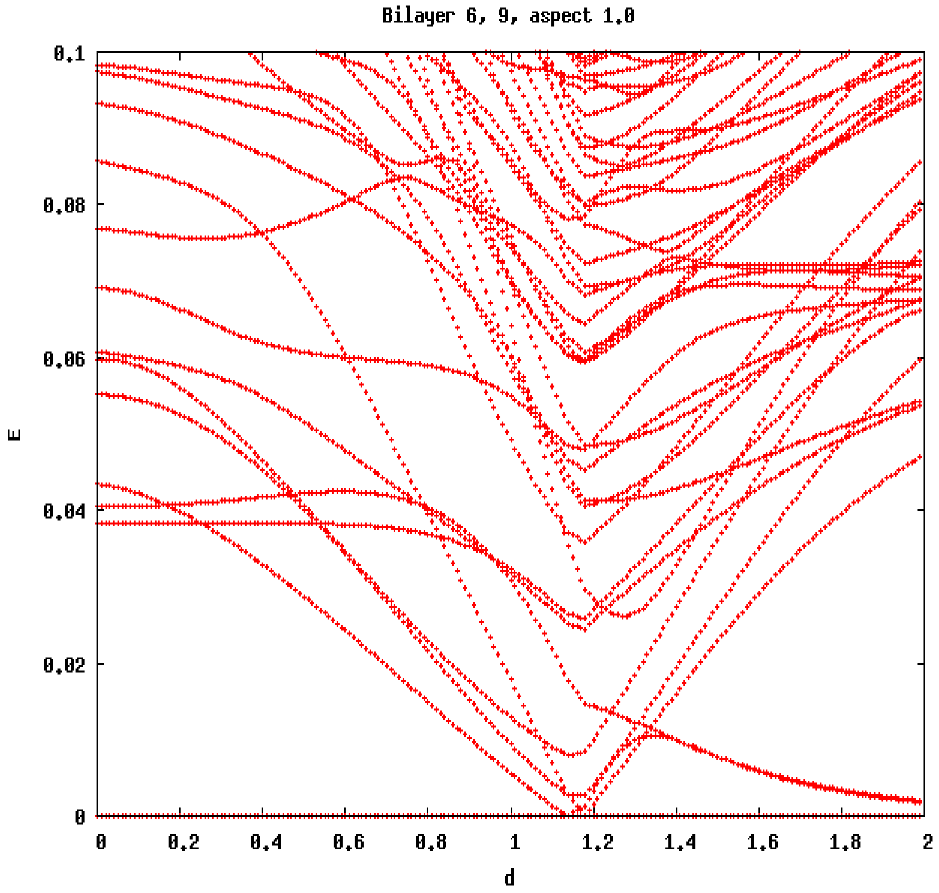
Hamiltonian



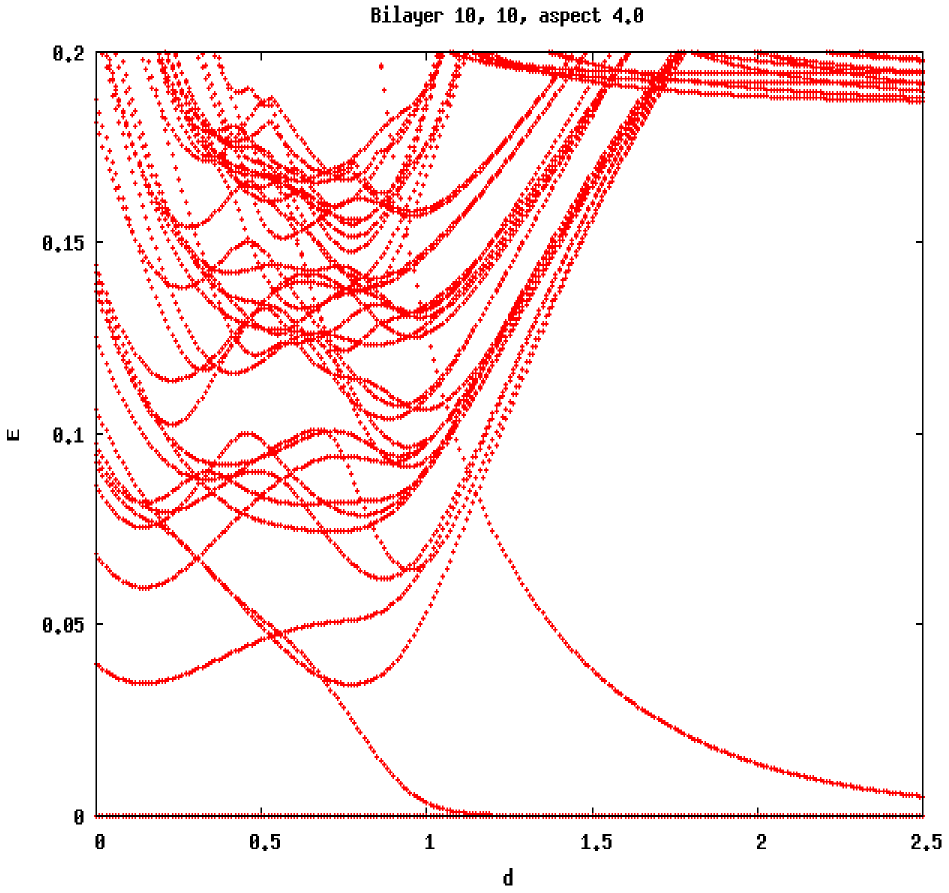
DiagHam Options

- system options :
  - -p, - -nbr-particles :   number of particles
  - -l, - -lzmax : twice the maximum momentum for a single particle
  - - -initial-lz : twice the inital momentum projection for the system
  - - -nbr-lz :   number of lz value to evaluate
  - - SU(2), SU(3), SU(4) spin projection
- parallelization options :
  - -S, - -SMP : enable SMP mode
  - - -processors : number of processors to use in SMP mode
- Lanczos options : ....
- precalculation options :
  - -m, - -memory : amount of memory that can be allocated for fast multiplication
  - - -fast-search : amount of memory that can be allocated for fast state search (in Mbytes)
- misc options :
  - -h, - -help

QHBilayer



Bilayer 6, 9, aspect 1.0
E
d
0.1
0.08
0.06
0.04
0.02
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2



Bilayer 10, 10, aspect 4.0
E
d
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2
2.5

Conclusions

- Exact diagonalization studies are essential in strongly correlated electrons (like FQHE) where nonpertubative insight is needed to describe even the basic physics
- Depending on the properties investigated, one may exploit different geometries to perform ED
- Unfortunately, ED will remain limited up to ∼ 20 particles (Monte Carlo can go much higher, but it requires knowledge of the wavefunction; DMRG can reach ∼ 30 particles, but no progress is expected beyond that limit).