The ALPS Project

Open Source Software for Strongly Correlated Systems

Matthias Troyer, ETH Zürich

for the ALPS collaboration

The ALPS collaboration

ETH Zürich, Switzerland
- Philippe Corboz
- Lukas Gamper
- Emanuel Gull
- Lode Pollet
- Matthias Troyer

IRRMA Lausanne, Switzerland
- Andreas Läuchli

RWTH Aachen, Germany
- Ulrich Schollwöck

Universität Marburg, Germany
- Reinhard Noack
- Salvatore Manmana

Institute of Metal Physics, Ekaterinburg, Russia
- Anton Kozhevnikov

A. Mickiewicz University, Poznan, Poland
- Grzegorz Pawłowski

University of Queensland, Australia
- Ian McCulloch

University of New South Wales, Australia
- Fabricio Albuquerque

Universität Stuttgart, Germany
- Stefan Wessel

Université de Toulouse, France
- Fabien Alet

TU Graz, Austria
- Franz Michel

UC Santa Barbara, USA
- Adrian Feiguin
- Simon Trebst

UC Davis, USA
- Munehisa Matsumoto

Columbia University, USA
- Philipp Werner

Honk Kong University, China
- Siegfried Gürtler

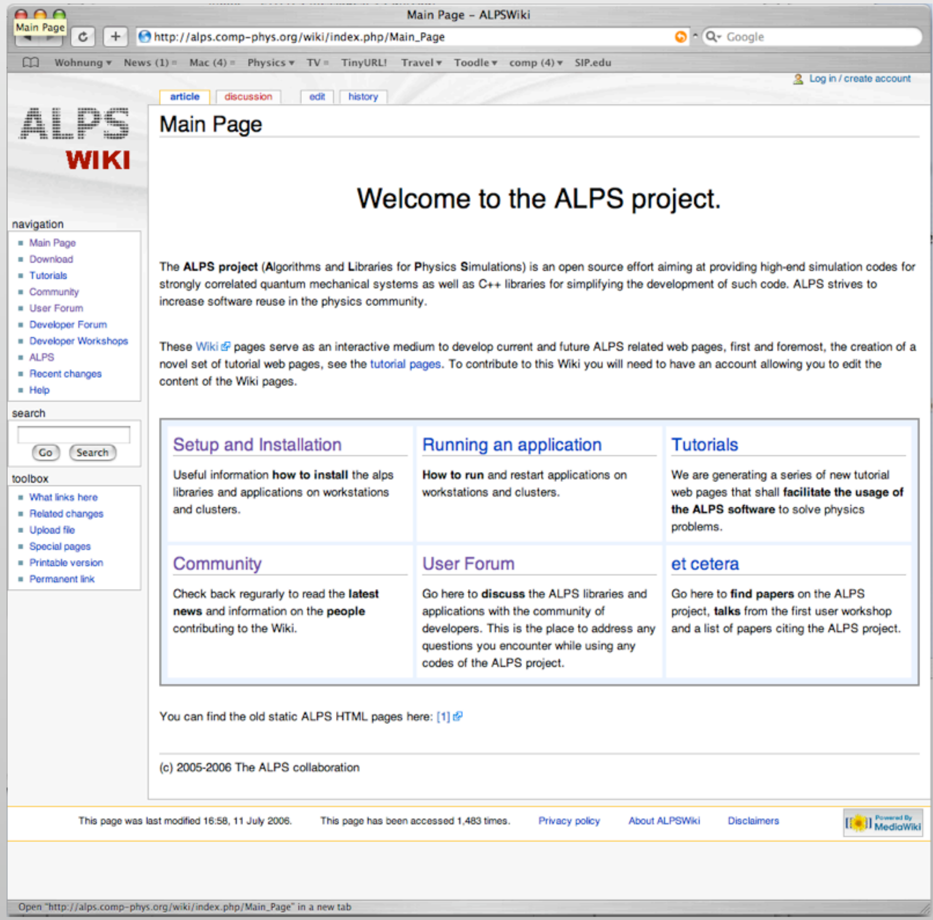
University of Tokyo, Japan
- Ryo Igarashi
- Synge Todo

Universität Göttingen, Germany
- Sebastian Fuchs
- Andreas Honecker
- Thomas Pruschke

The ALPS project

Algorithms and **L**ibraries for **P**hysics **S**imulations

- **open source** data formats, libraries and simulation codes for quantum lattice models
- download codes from website **http://alps.comp-phys.org**



Simulation codes of quantum lattice models

Simulation codes of quantum lattice models

- **The status quo**
  - individual codes
  - model-specific implementations
  - growing complexity of methods

Simulation codes of quantum lattice models

- **The status quo**
  - individual codes
  - model-specific implementations
  - growing complexity of methods

- **ALPS**
  - community codes
  - generic implementations
  - simplified code development
  - common file formats

Simulation codes of quantum lattice models

- **The status quo**
  - individual codes
  - model-specific implementations
  - growing complexity of methods

- **ALPS**
  - community codes
  - generic implementations
  - simplified code development
  - common file formats

Key Technologies

Generic Programming in C++
- flexibility
- high-performance

Standard C++ Libraries
- fast development

XML / XSLT for Input/Output
- portability
- self-explanatory

MPI/OpenMP for Parallelization

Three tiers of ALPS

Three tiers of ALPS

1. **Standard data formats and interfaces** to facilitate
    - exchange, archiving and querying of simulation results
    - exchange of simulation and analysis tools

Three tiers of ALPS

1. **Standard data formats and interfaces** to facilitate
    - exchange, archiving and querying of simulation results
    - exchange of simulation and analysis tools
2. **Libraries**
    - to support standard data formats and interfaces
    - to ease building of parallel simulation programs

Three tiers of ALPS

1. **Standard data formats and interfaces** to facilitate
   - exchange, archiving and querying of simulation results
   - exchange of simulation and analysis tools
2. **Libraries**
   - to support standard data formats and interfaces
   - to ease building of parallel simulation programs
3. **Applications**
   - to be used also by non-experts
   - implement modern algorithms for a large class of models

The ALPS project

Algorithms and **L**ibraries for **P**hysics **S**imulations

The ALPS project

Algorithms and **L**ibraries for **P**hysics **S**imulations

- The **simulation codes** include
  - Classical and Quantum Monte Carlo
  - Exact and Full Diagonalization
  - Density Matrix Renormalization Group (DMRG)

The ALPS project

Algorithms and **L**ibraries for **P**hysics **S**imulations

- The **simulation codes** include
  - Classical and Quantum Monte Carlo
  - Exact and Full Diagonalization
  - Density Matrix Renormalization Group (DMRG)
- **Motivation**
  - established algorithms
  - increased demand for reliable simulations from theorists and experimentalists

What is XML?

- e**X**tensible **M**arkup **L**anguage
- We mix text with “tags” defining the function of the text
- Example: HTML

```
<HTML>
  <H1>Header</H1>
  <P>A paragraph …..
   ……. And below it an image</P>
  <IMG source=“image.jpg”/>
</HTML>
```

What is XML?

- e**X**tensible **M**arkup **L**anguage
- We mix text with “tags” defining the function of the text
- Example: HTML

Opening tag

```
<HTML>
  <H1>Header</H1>
  <P>A paragraph …..
   ……. And below it an image</P>
  <IMG source=“image.jpg”/>
</HTML>
```

What is XML?

- e**X**tensible **M**arkup **L**anguage
- We mix text with “tags” defining the function of the text
- Example: HTML

Opening tag

```
<HTML>
  <H1>Header</H1>
  <P>A paragraph …..
   ……. And below it an image</P>
  <IMG source=“image.jpg”/>
</HTML>
```

Closing tag starts with /

What is XML?

- e**X**tensible **M**arkup **L**anguage
- We mix text with “tags” defining the function of the text
- Example: HTML

Opening tag

```
<HTML>
  <H1>Header</H1>
  <P>A paragraph …..
   ……. And below it an image</P>
  <IMG source=“image.jpg”/>
</HTML>
```

Tag ending with / is both opening and closing

Closing tag starts with /

What is XML?

- eXtensible Markup Language
- We mix text with "tags" defining the function of the text
- Example: HTML

Opening tag

```
<HTML>
  <H1>Header</H1>
  <P>A paragraph …..
   ……. And below it an image</P>
  <IMG source="image.jpg"/>
</HTML>
```

Contents

Tag ending with / is both opening and closing

Closing tag starts with /

What is XML?

- eXtensible Markup Language
- We mix text with “tags” defining the function of the text
- Example: HTML

Opening tag

```
<HTML>
   <H1>Header</H1>
   <P>A paragraph …..
    ……. And below it an image</P>
   <IMG source=“image.jpg”/>
</HTML>
```

Contents

An attribute

Tag ending with / is both opening and closing

Closing tag starts with /

Why use XML?

- Plain text file:

```
# first row parameters
10 0.5 10000 1000
# mean, error
-10.451 0.043
```

XML:

```
<PARAMETER name="L">10</PARAMETER>
<PARAMETER name="T">0.5</PARAMETER>
<PARAMETER name="SWEEPS">10000</PARAMETER>
<PARAMETER name="THERMALIZATION">1000</PARAMETER>
<AVERAGE name="Energy">
  <MEAN> -10.451 </MEAN>
  <ERROR> 0.043 </ERROR>
</AVERAGE>
```

- Which is easier to understand?
- Which is better machine-readable?
- Which one will you understand in a few years?

Why use XML?

- Extending the data format: let’s add random number generator type and seed

- Plain text file:

```
# first row parameters
10 0.5 10000 1000 12
# random number generator
“Mersenne Twister”
# mean, error
-10.451 0.043
```

XML:

```
<PARAMETER name=“L”>10</PARAMETER>
<PARAMETER name=“T”>0.5</PARAMETER>
<PARAMETER name=“SWEEPS”>10000</PARAMETER>
<PARAMETER name=“THERMALIZATION”>1000
</PARAMETER>


<PARAMETER name=“SEED”>12</PARAMETER>
<RNG name=“Mersenne Twister”/>
<AVERAGE name=“Energy”>
  <MEAN> -10.451 </MEAN>
  <ERROR> 0.043 </ERROR>
</AVERAGE>
```

- The change in the text file format might break your program
- The additional XML tag is no problem

Calculating π

- We calculated π

```
<RNG name="RanF"/>

<AVERAGE name="Pi">
  <MEAN>  3.1566 </MEAN>
  <ERROR> 0.0048 </ERROR>
  <COUNT> 33554432 </COUNT>
</AVERAGE>
```

- Now that we know that RanF is a bad generator we know which data to throw away

XSLT transformations

XSLT transformations

- But the XML file is ugly to look at ...

XSLT transformations

- But the XML file is ugly to look at …

XSLT transformations

- But the XML file is ugly to look at ...
- No problem, it is meant only for the computer's eyes

XSLT transformations

- But the XML file is ugly to look at …
- No problem, it is meant only for the computer's eyes

XSLT transformations

- But the XML file is ugly to look at …
- No problem, it is meant only for the computer's eyes
- XSLT transforms ("stylesheets") allow conversions into any format
    - Other XML
    - HTML
    - Plain text

XSLT transformations

- But the XML file is ugly to look at ...
- No problem, it is meant only for the computer's eyes
- XSLT transforms ("stylesheets") allow conversions into any format
  - Other XML
  - HTML
  - Plain text
- I'll show you an example ...

Why use XML?

- Contents marked up with context
  - Reduces data rot
  - Increases portability of data
- Extensible
  - Can add new contents without breaking old programs
- XSLT
  - Can use “stylesheets” to display/convert contents into any other format
- ISO standard
  - Many tools available: editors, browsers, databases, ...

Simulations with ALPS

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>
```

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>
```

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 }
```

Simulations with ALPS

Lattice

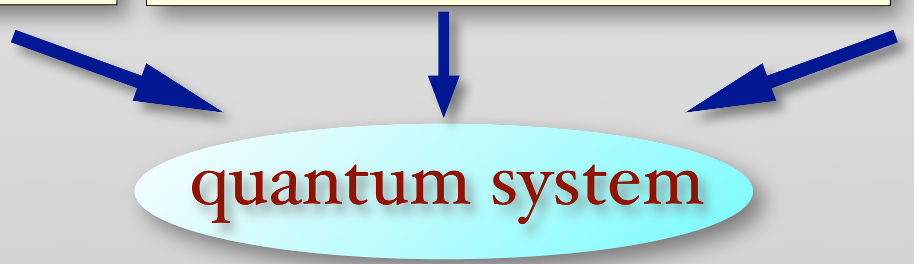
```
<LATTICEGRAPH name = "square lattice">
  <FINITELATTICE>
    <LATTICE dimension="2"/>
    <EXTENT dimension="1" size="L"/>
    <EXTENT dimension="2" size="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL>
    ...
  </UNITCELL>
</LATTICEGRAPH>
```

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 }
```



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 }
```

quantum system

Quantum Monte Carlo

Exact diagonalization

DMRG

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 }
```

quantum system

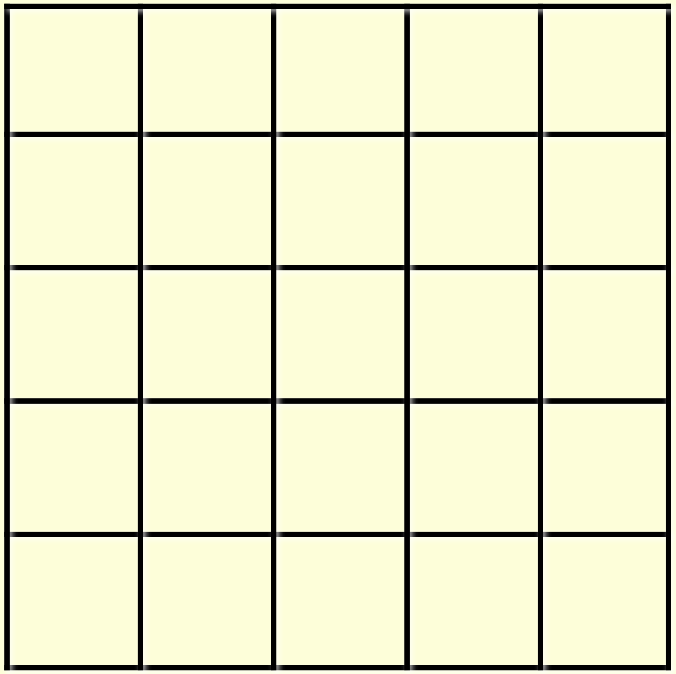
Quantum Monte Carlo | **Exact diagonalization** | **DMRG**

Results

The ALPS lattice library

A lattice

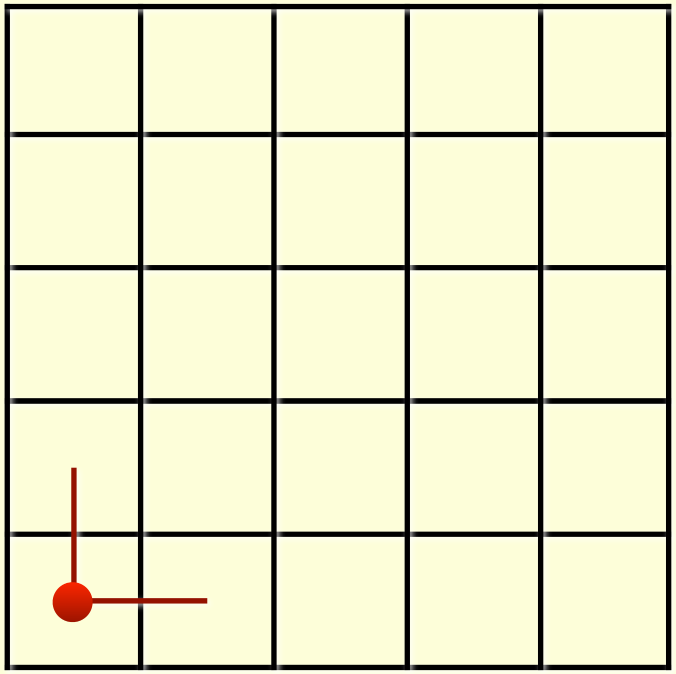
```
<LATTICEGRAPH name = "square lattice">
  <FINITELATTICE>
    <LATTICE dimension="2"/>
    <EXTENT dimension="1" size="L"/>
    <EXTENT dimension="2" size="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL>
    <VERTEX/>
    <EDGE type="1">
      <SOURCE vertex="1" offset="0 0"/>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



The ALPS lattice library

A lattice

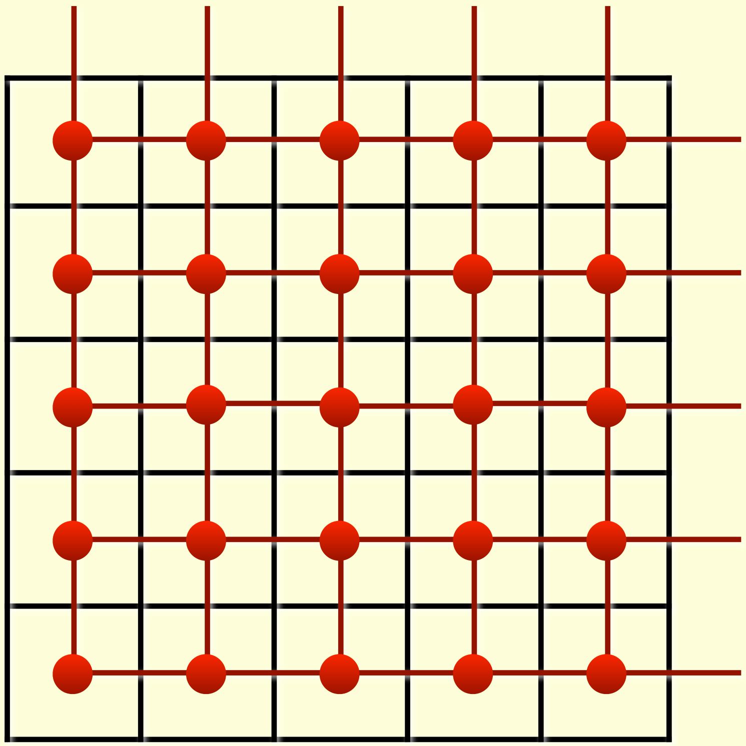
```
<LATTICEGRAPH name = "square lattice">
  <FINITELATTICE>
    <LATTICE dimension="2"/>
    <EXTENT dimension="1" size="L"/>
    <EXTENT dimension="2" size="L"/>
    <BOUNDARY type="periodic"/>
  </FINITELATTICE>
  <UNITCELL>
    <VERTEX/>
    <EDGE type="1">
      <SOURCE vertex="1" offset="0 0"/>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



The ALPS lattice library

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>
```



The ALPS model library

A model

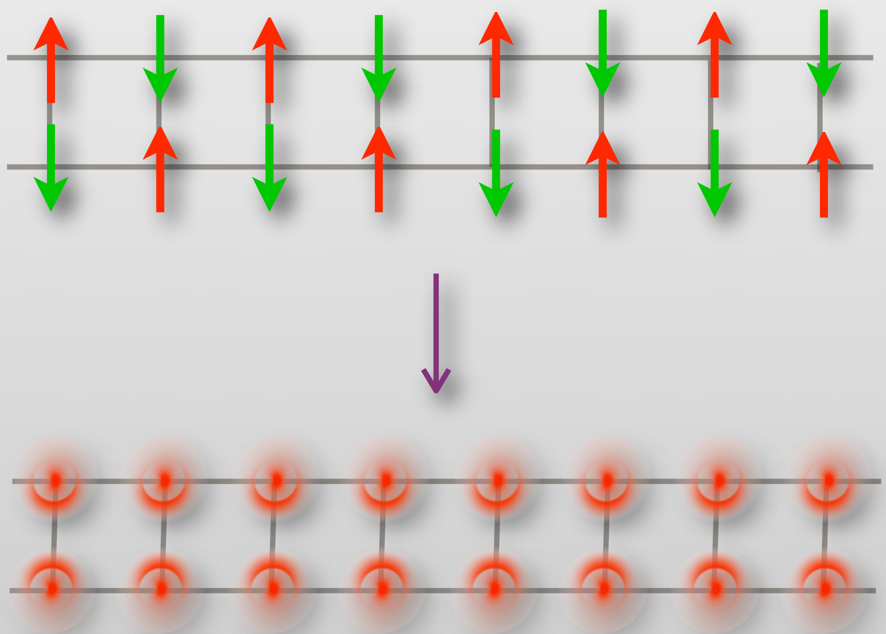
$$H_{XXZ} = \frac{J_{xz}}{2} \sum_{\langle i,j \rangle} (S_i^+ S_j^- + S_i^- S_j^+) + J_z \sum_{\langle i,j \rangle} S_i^z S_j^z - h \sum_i S_1^z$$

```
<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>
```

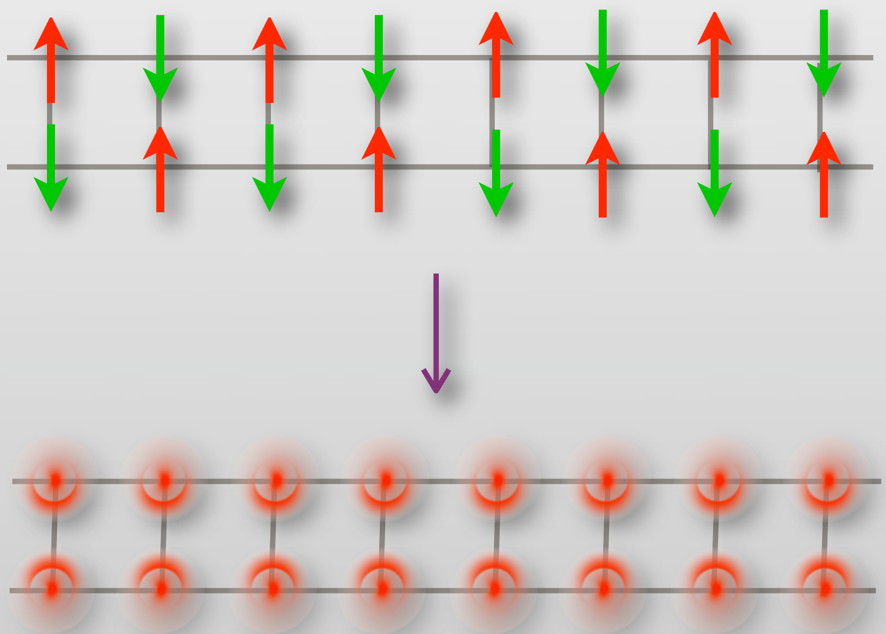
Current applications

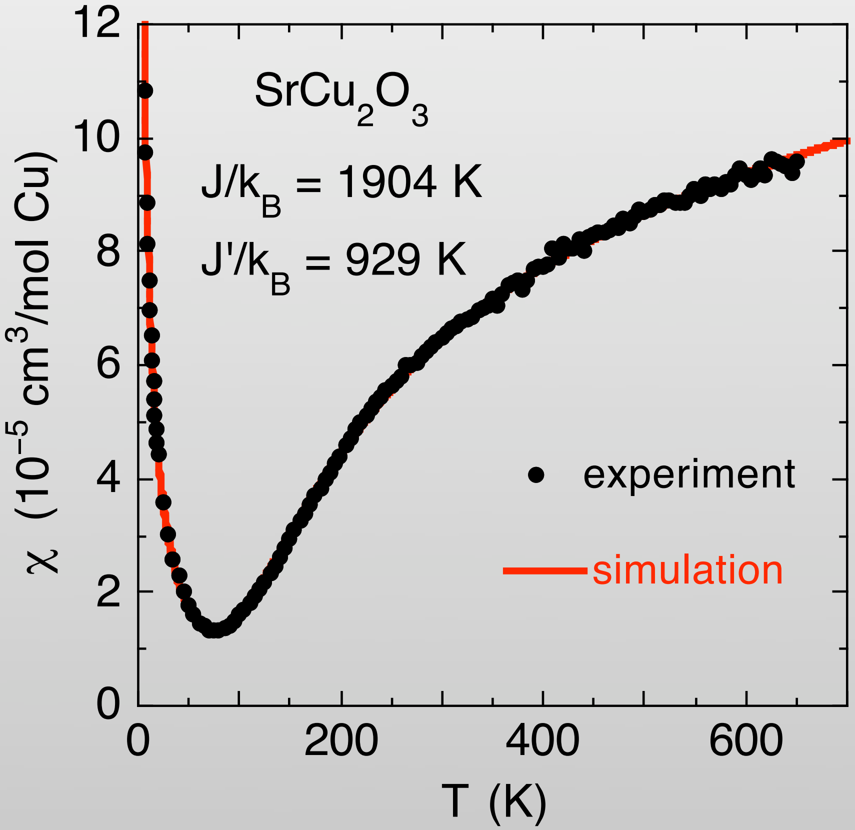
- **Classical Monte Carlo**
  - local and cluster updates for classical spin systems, M. Troyer
- **Quantum Monte Carlo**
  - stochastic series expansions (SSE), F. Alet, L. Pollet, M. Troyer
  - loop code for spin systems, S. Todo
  - continuous time worm code, S. Trebst, M. Troyer
  - extended ensemble simulations, S. Wessel, N. Stoop
- **Exact diagonalization**
  - full and sparse, A. Honecker, A. Läuchli, M. Troyer
- **DMRG**
  - single particle, S. Manmana, R. Noack, U. Schollwöck
  - interacting particles, A. Feiguin

Quantum spin ladders

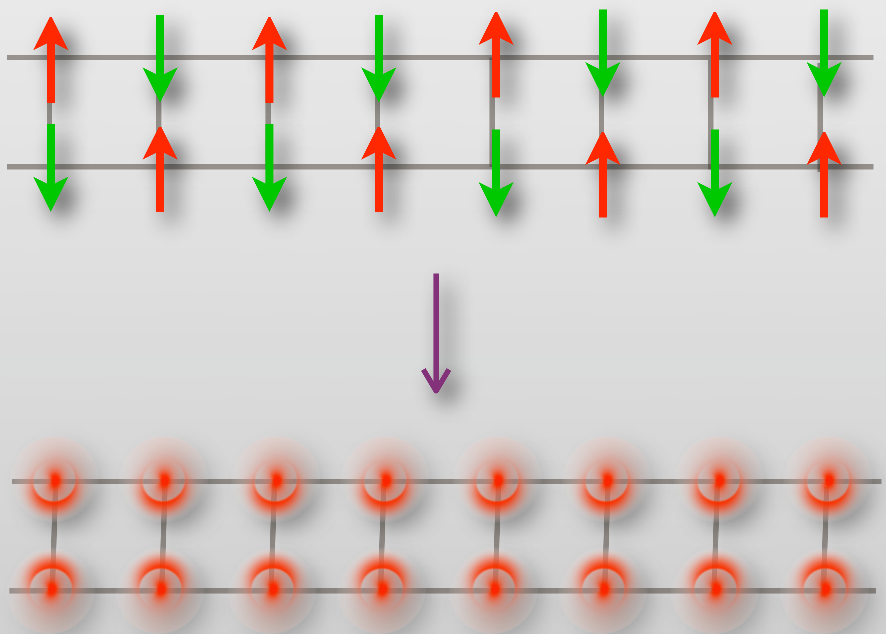


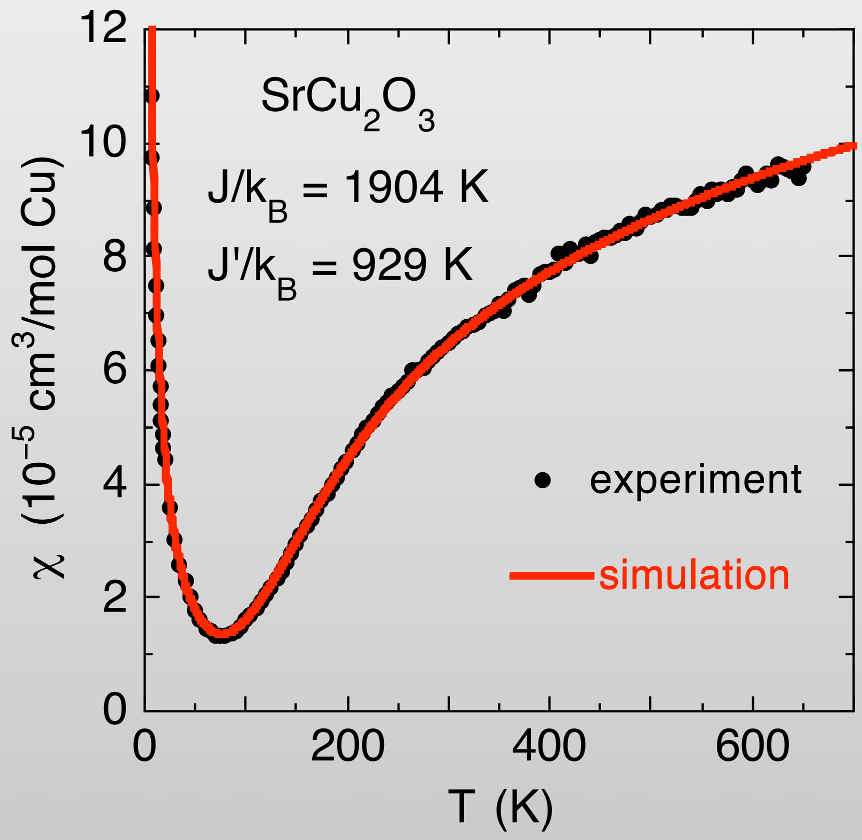
Quantum spin ladders



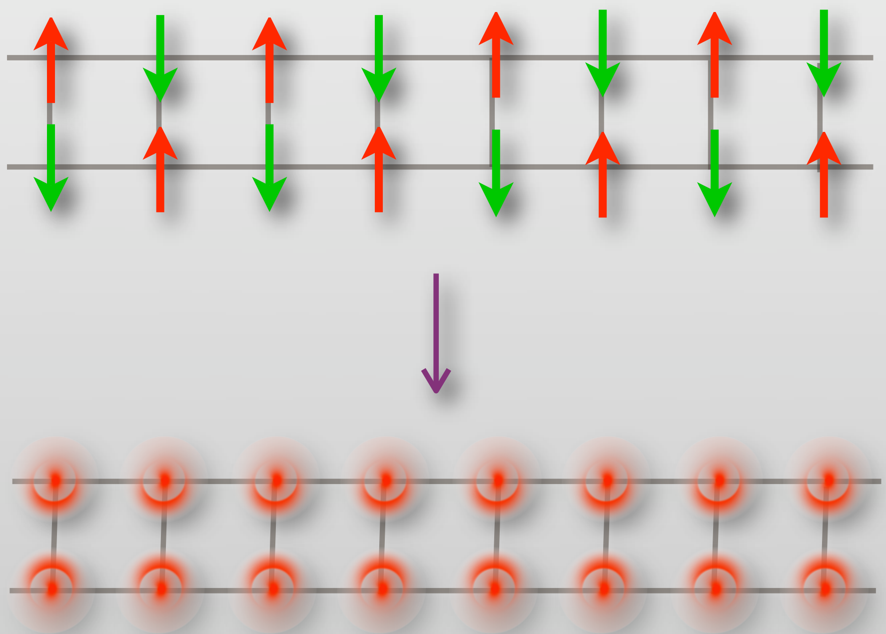


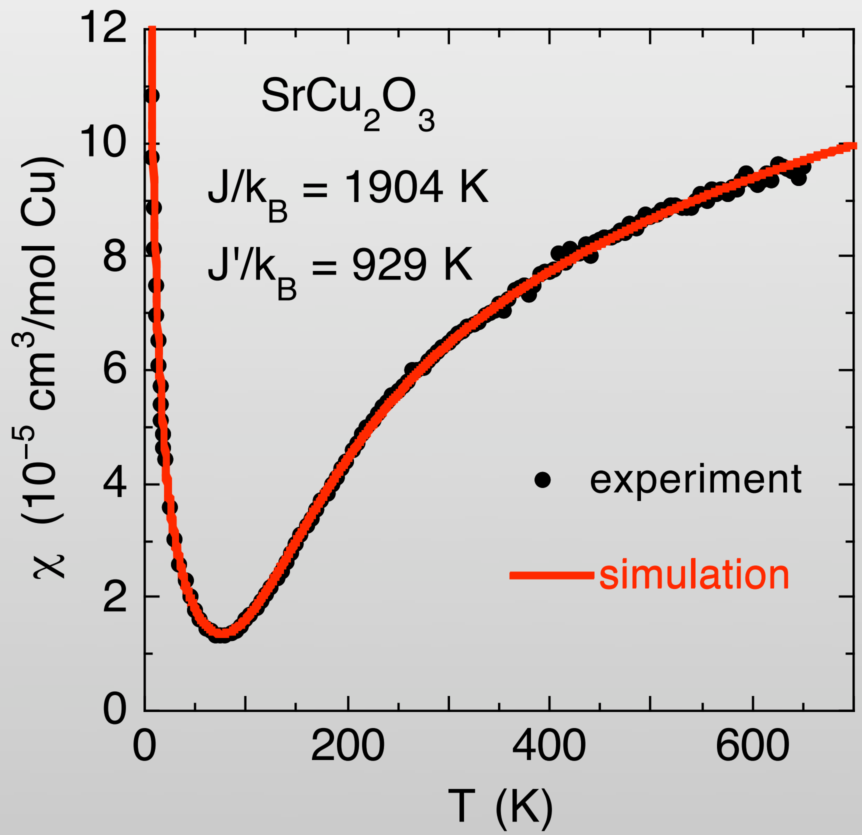
Quantum spin ladders





Quantum spin ladders



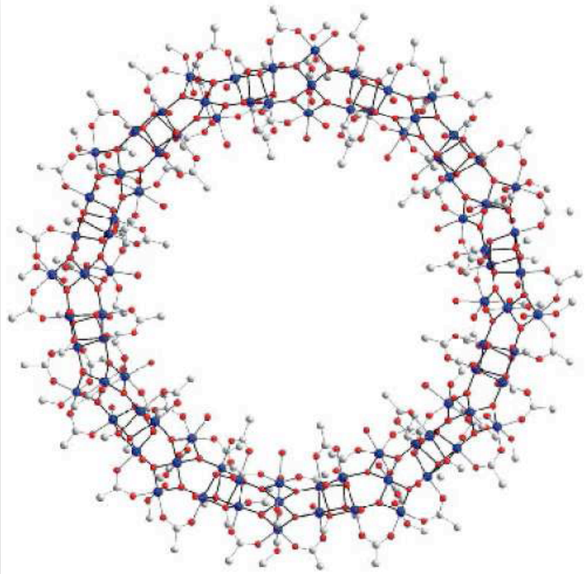


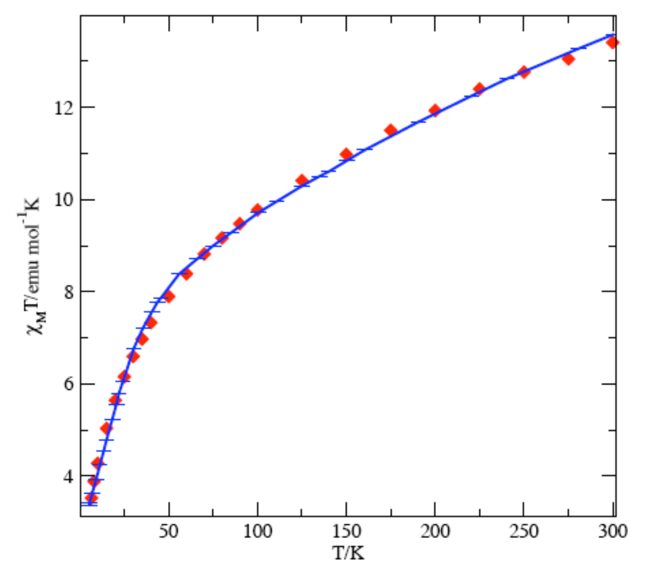
➡ compare microscopic models to experiments

Mn-84 molecules

Vassilis Tangoulis, in preparation

- How can we microscopically model interactions in Mn-84?





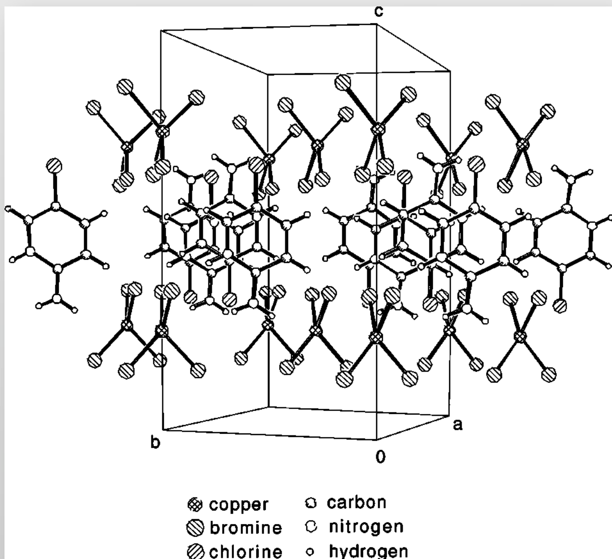
ALPS QMC codes

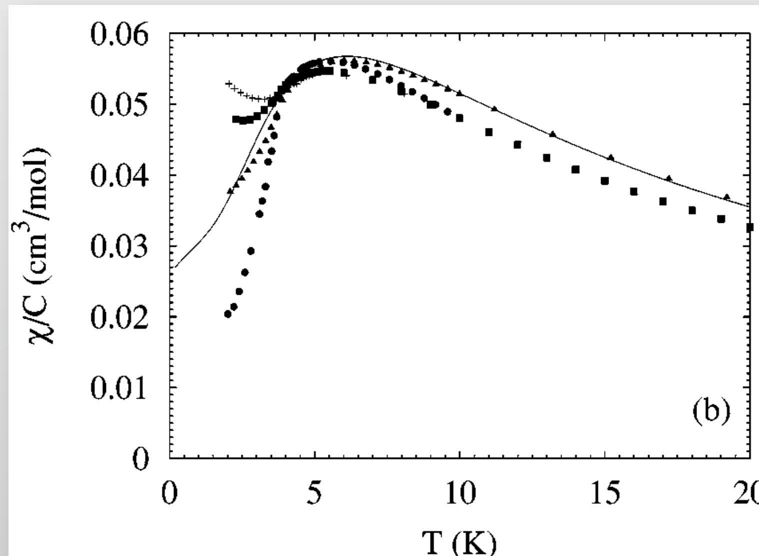
Numerical evaluation of susceptibility for full molecule:
Fit of magnetic interaction strength.

Low-dimensional quantum magnets

C.P. Landee et al., Phys. Rev. B **65**, 144412 (2002)

- How to characterize newly synthesized materials?

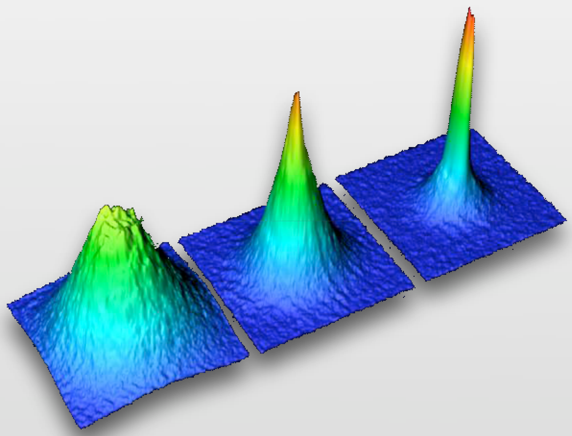




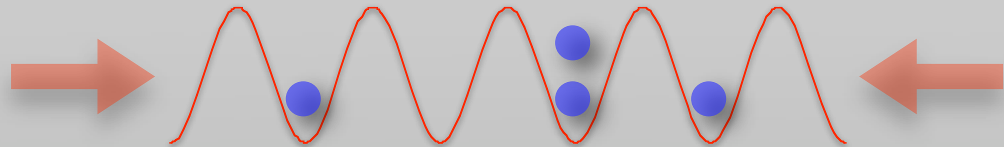
ALPS QMC codes

Numerical evaluation of susceptibility for 2D QHAF:
Fit of magnetic interaction strength.

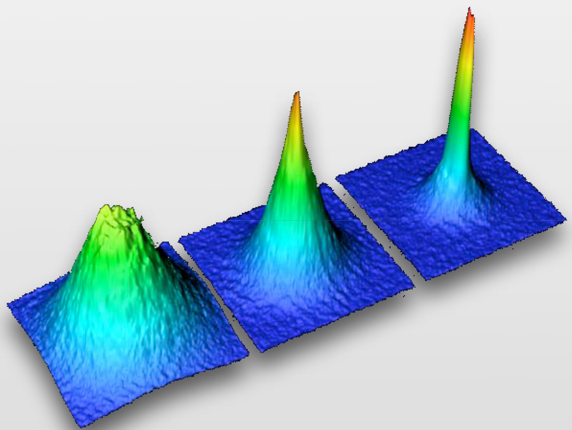
BEC in ultracold atomic gases



T. Esslinger, ETH Zürich

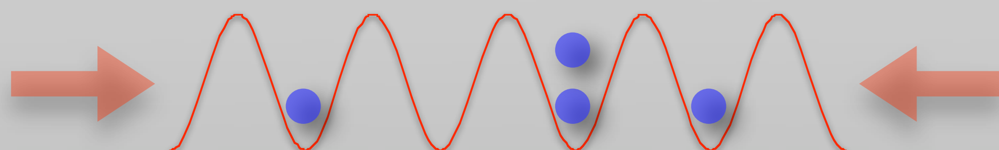


BEC in ultracold atomic gases

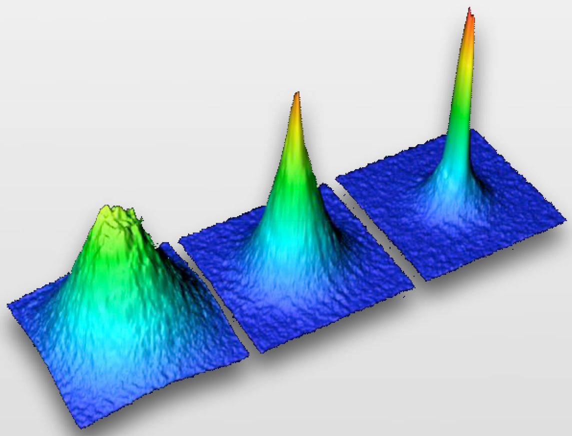


T. Esslinger, ETH Zürich

- Ultracold ^{87}Rb atoms form a Bose-Einstein condensate (BEC).
  - first observed in 1995
  - sympathetic cooling of fermionic ^{40}K atoms (2004)

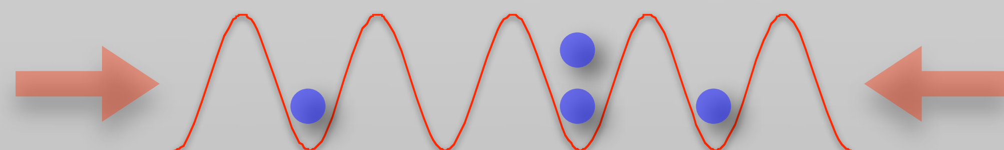


BEC in ultracold atomic gases



T. Esslinger, ETH Zürich

- Ultracold ^{87}Rb atoms form a Bose-Einstein condensate (BEC).
  - first observed in 1995
  - sympathetic cooling of fermionic ^{40}K atoms (2004)
- Standing laser waves form an optical lattice.

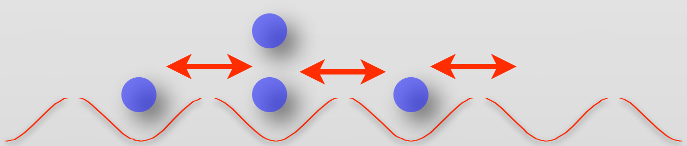


Realization of the Bose-Hubbard model

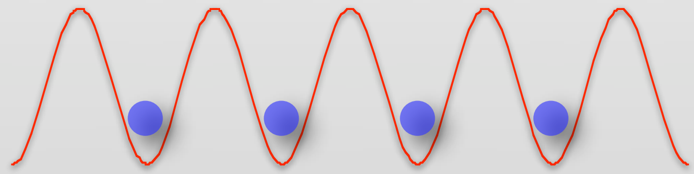
S. Wessel et al., Phys. Rev. A **70**, 053615 (2004)
O. Gygi et al., Phys. Rev. A **73**, 063606 (2006)

$$H = -t \sum_{\langle ij \rangle} \left(b_i^\dagger b_j + \mathrm{h.c.} \right) + U \sum_i n_i (n_i - 1)/2 - \mu \sum_i n_i + V \sum_i r_i^2 n_i$$

local density



suprafluid
coherent BEC



Mott-isolator
incoherent

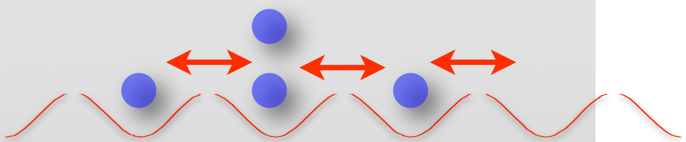
ALPS QMC codes

Numerical simulation of experimental setup:
60^2 sites and harmonic trapping potential

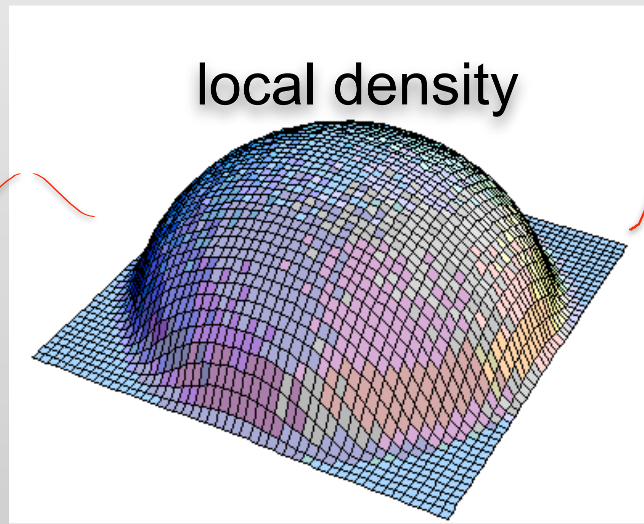
Realization of the Bose-Hubbard model

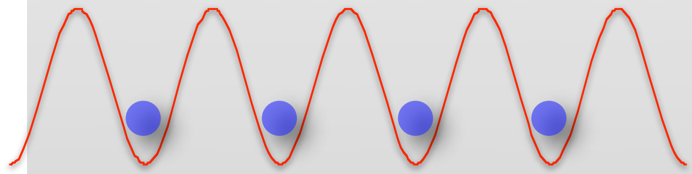
S. Wessel et al., Phys. Rev. A **70**, 053615 (2004)
O. Gygi et al., Phys. Rev. A **73**, 063606 (2006)

$$H = -t \sum_{\langle ij \rangle} \left(b_i^\dagger b_j + \text{h.c.} \right) + U \sum_i n_i(n_i - 1)/2 - \mu \sum_i n_i + V \sum_i r_i^2 n_i$$



suprafluid
coherent BEC





Mott-isolator
incoherent

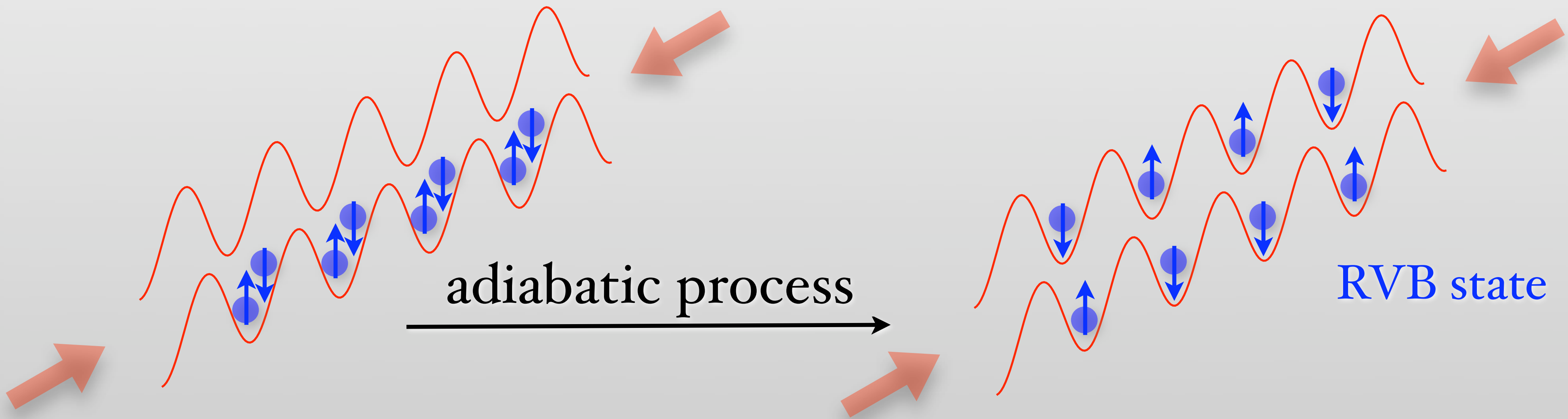
ALPS QMC codes

Numerical simulation of experimental setup:
60^2 sites and harmonic trapping potential

Ultracold fermionic atoms

S. Trebst *et al.*, Phys. Rev. Lett. **96**, 250402 (2006)

- How can we cool down fermions to some 0.01 T_F ?



ALPS exact diagonalization codes

Excitation spectra of intermediate states.

Time-evolution of proposed adiabatic processes.

Ab-initio simulations of quantum magnets

- Simulate realistic magnetic models instead of toy models
  - obtain microscopic exchange constants from LDA+U
  - simulate quantum spin models using these exchange constants
- Was done by hand in the past
  - CaV_2O_3, MgV_2O_3, CaV_3O_7, CaV_4O_9
  - Korotin, Elfimov, Anisimov, Troyer and Khomskii, PRL '99
- Can we automate this?

ALPS Interface to band structure codes

- ORNL is developing standard XML I/O data formats and helper libraries for band structure codes
- Implementation in Stuttgart TB-LMTO-ASA band structure code by Anton Kozhevnikov (Ekaterinburg)
- Simple helper tool by Anton Kozhevnikov creates ALPS input file from XML output of LDA+U code
- Automated workflow from crystal structure to magnetic properties

Example: $SrCu_2O_3$