Addressing Exascale Challenges for Numerical Algorithms of Strongly Correlated Lattice Models

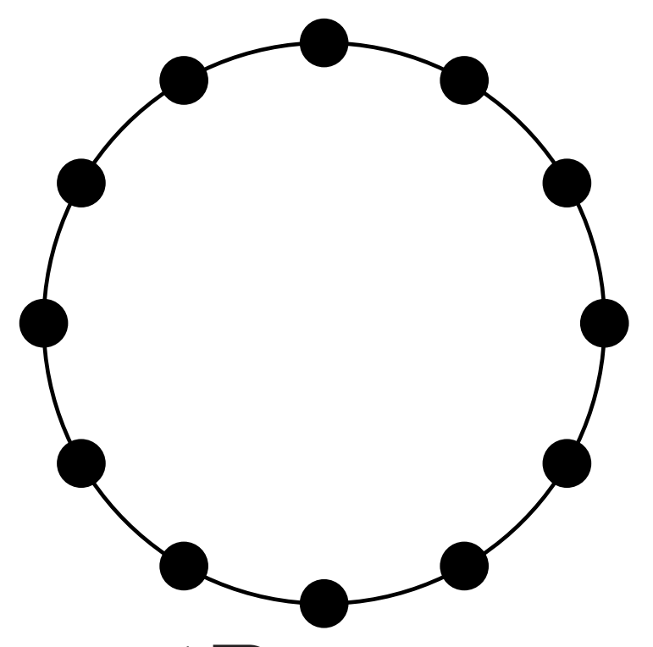
Samuel Gozel and Andreas M. Läuchli
Laboratory for Theoretical and Computational Physics, Paul Scherrer Institute, CH-5232 Villigen-PSI, Switzerland

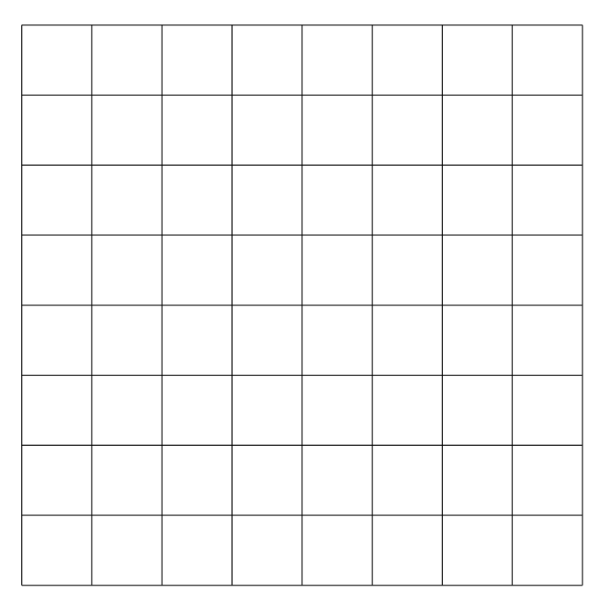
Introduction

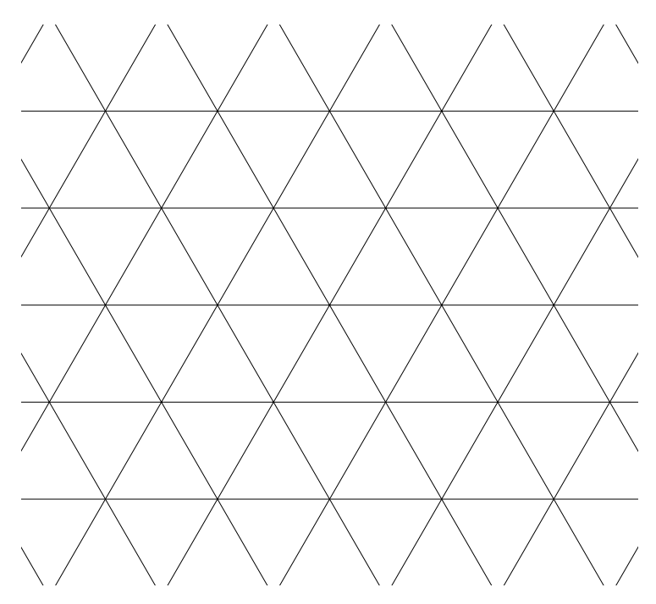
Strongly correlated lattice models are a class of quantum models where particles are localized at some lattice sites and are strongly interacting with each other. The Heisenberg model, for instance, describes the interaction of spins on a lattice. Solving such a model amounts to diagonalizing a symmetric (or hermitian) matrix, the Hamiltonian, for a few lowest eigenvalues. Since the matrix dimension scales exponentially in the number of particles, it is necessary to take symmetries into account to achieve large system sizes [1]. These symmetries allow to write the Hamiltonian in block diagonal form. Keeping the Hamiltonian matrix of the wanted block in sparse format is not an option as its dimension is of order $10^{11}-10^{12}$ for system sizes of interest. In the large-scale Lanczos algorithm, it is thus necessary to perform the matrix vector multiplication (MVM) by building the Hamiltonian on the fly. This construction, however, is far from trivial and implies various types of operations. This so-called "Exact Diagonalization" (ED) algorithm has been successfully implemented using multi-core and multi-node architectures (`pthreads`, OpenMP, MPI) [2, 3]. Given the current evolution in the architecture of HPC clusters, our aim is to develop a performance portable ED algorithm running on GPUs.

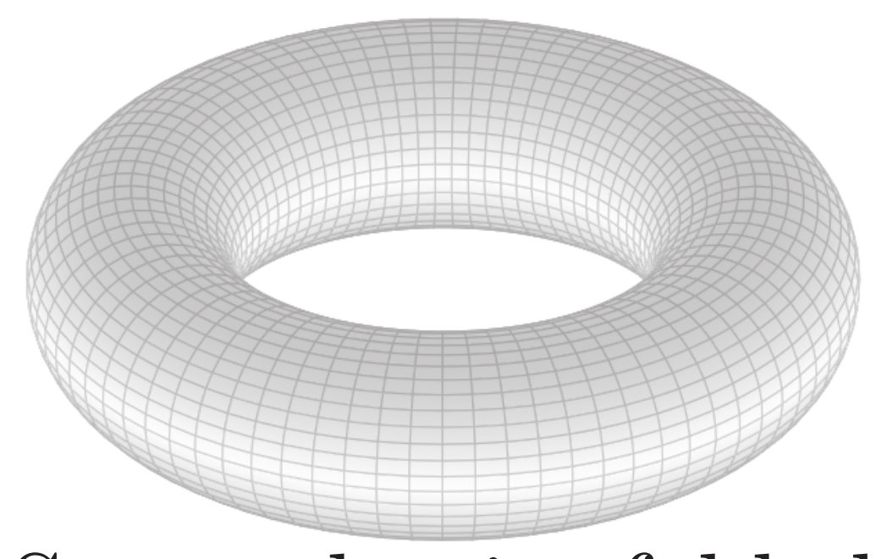
Physical models

Spin lattice models describe the behavior of magnetic ions in crystals. The underlying lattice may be of different types. In practice, we impose periodic boundary conditions to reproduce the real "infinite-size" system and to exploit translation symmetry.


1D ring


Square lattice


Triangular lattice


Square lattice folded into a torus

We then define an interaction Hamiltonian as a sum of local, non-commuting terms

$$\mathcal{H} = \sum_k \mathcal{H}_k, \quad \text{possible interactions: } \mathcal{H}_k = \mathbf{S}_i \cdot \mathbf{S}_j, \quad S_i^x S_j^x + S_i^y S_j^y + \Delta S_i^z S_j^z, \quad \ldots$$

The symmetries are spatial symmetries (translation, reflection, rotation) and spin symmetries (spin inversion, spin U(1) and spin SU(2) symmetries).

Large-scale problem

The dimension of the entire vector space describing the states of the model scales exponentially with the system size N as d^N, where d is the "local" vector space dimension ($d = 2$ for a $S = 1/2$ spin). The symmetries, however, allow to significantly reduce the effective vector space dimension D (but not its scaling). Below we show the dimension of the matrix of the systems that we target, together with the memory required to store one real Lanczos vector in double precision (64 bits).

# of sites N	46	48	50	52
2^N	7.0e13	2.8e14	1.1e15	4.5e15
D	8.9e10	3.4e11	1.3e12	4.8e12
Memory TB	0.7	2.7	10	38
Status	state-of-the-art		our goal	

We need to store 3 arrays (64 bits) of dimension D.

Algorithm

We use the `Kokkos` library [4] to achieve portability. In the MVM, there are 2 major `for` loops: one (small) loop over all bonds in the lattice, thus of order $\mathcal{O}(N)$, and the large loop over all states (D). Within these loops, there are 3 loops over $T = \mathcal{O}(N)$ elements (see ① and ②).

`states` = array of `uint64` of dimension D
u, w = arrays of `double` of dimension D

MVM $u \leftarrow Hw$ (focus on off-diagonal part)

```
...(diag part trivial)
for $b \in$ Bonds do
    for $j = 0, ..., D-1$ do
        if check(states$[j]$, b) then
            $u(j)$ += computeOffDiag(j, states, $b$, $w$)
```

Check `states`$[j]_{b_1}$!= `states`$[j]_{b_2}$

out = computeOffDiag(j, states, b, w)

```
outState = flip(states$[j]$, $b$)
outRepres = search_repres(outState)                    → ①
outIndex = binary_search(outRepres, states)
inAmpl = get_amplitude(states[j])                      } → ②
outAmpl = get_amplitude(outRepres)                     }
out = $J_{\text{off-diag}} w[\texttt{outIndex}]\sqrt{\texttt{inAmpl}}/\sqrt{\texttt{outAmpl}}$
```

① outRepres = search_repres(state)

```
outRepres = state
for $t = 1, ..., T$ do
    temp = applySymOp($t$, state)
    if temp < outRepres then
        outRepres = temp
```

`applySymOp()` contains a loop over N elements.

① is a `min` reduction ($T = \mathcal{O}(N)$).

② adds precomputed elements stored in a table of dimension $T = \mathcal{O}(N)$ when matching condition is true.

② ampl = get_amplitude(repres)

```
ampl = 1
for $t = 1, ..., T$ do
    temp = applySymOp($t$, repres)
    if temp == repres then
        ampl += phaseLookup$[t]$
```

Outlook

Exact diagonalization of quantum lattice models translates into a large-scale eigenvalue problem for a symmetric (or hermitian matrix). The constraint of building the matrix of the Hamiltonian on the fly for memory reasons leads to a complicated MVM where several loops appear when treating an off-diagonal term. Some loops contain `if` statements which cannot easily be removed. It is yet unclear what is the best method for executing such a MVM on a cluster of GPUs.

References

[1] H. Q. Lin, Phy. Rev. B **42** 6561 (1990)
[2] A. Weisse, Phys. Rev. E **87**, 043305 (2013)
[3] Wietek & Läuchli, Phys. Rev. E **98**, 033309 (2018)
[4] Ch. R. Trott *et al.*, IEEE Transactions on Parallel and Distributed Systems **33**, 805 (2022)