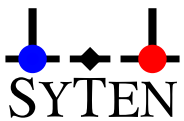
Global Symmetries in Tensor Networks

Claudius Hubig

MPQ

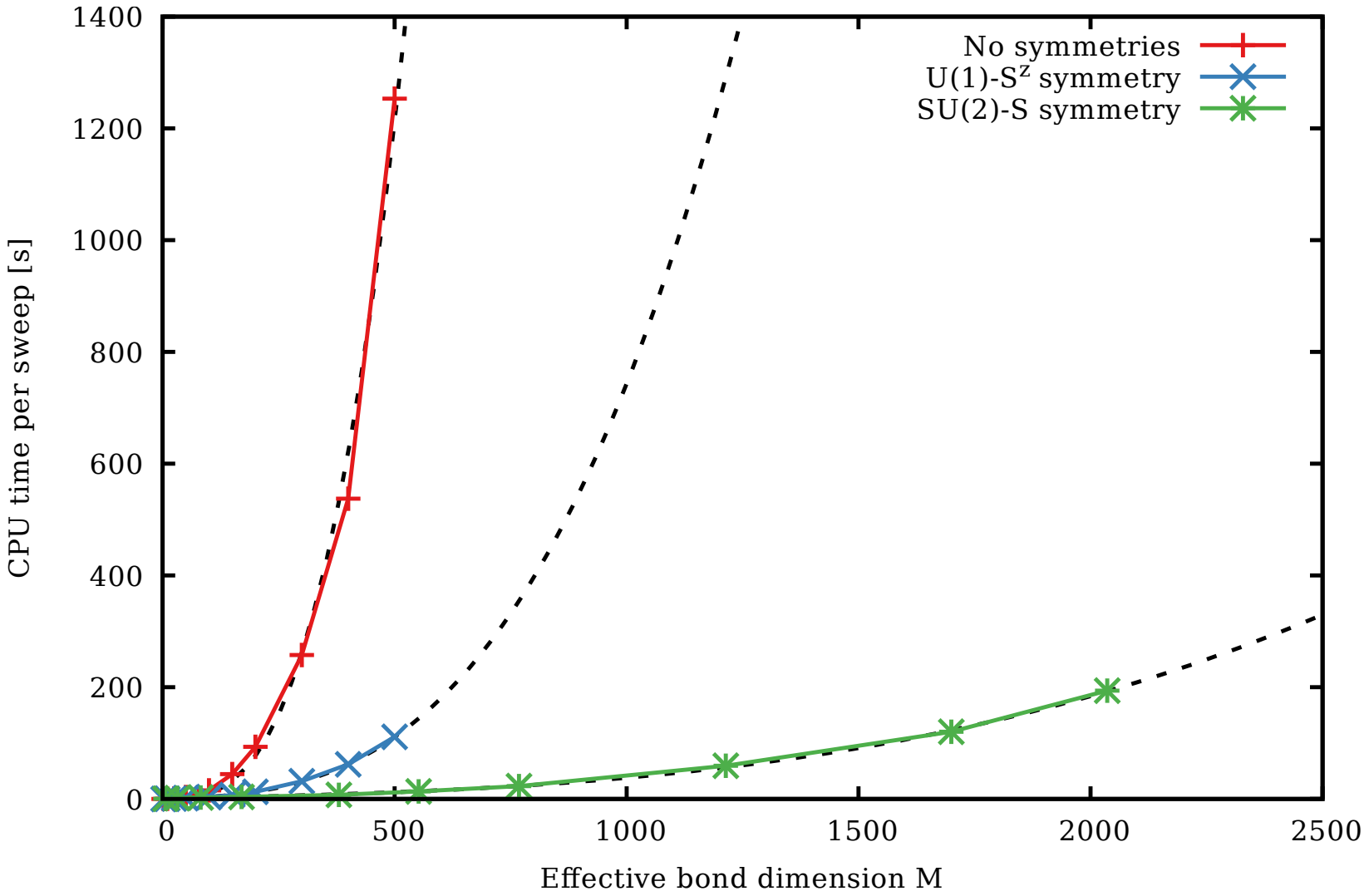


What?

- $\mathrm{U}(1)$: particle number N, spin projection S^z
- $\mathbb{Z}_k$: pseudomomentum k, parity p
- $\mathrm{SU}(2)$: total spin S, particle-hole symmetry C
- $\mathrm{SO}(3)$: angular momentum L
- $\mathrm{SU}(3)$, $\mathrm{Sp}(6)$, ...
- any combination of the above

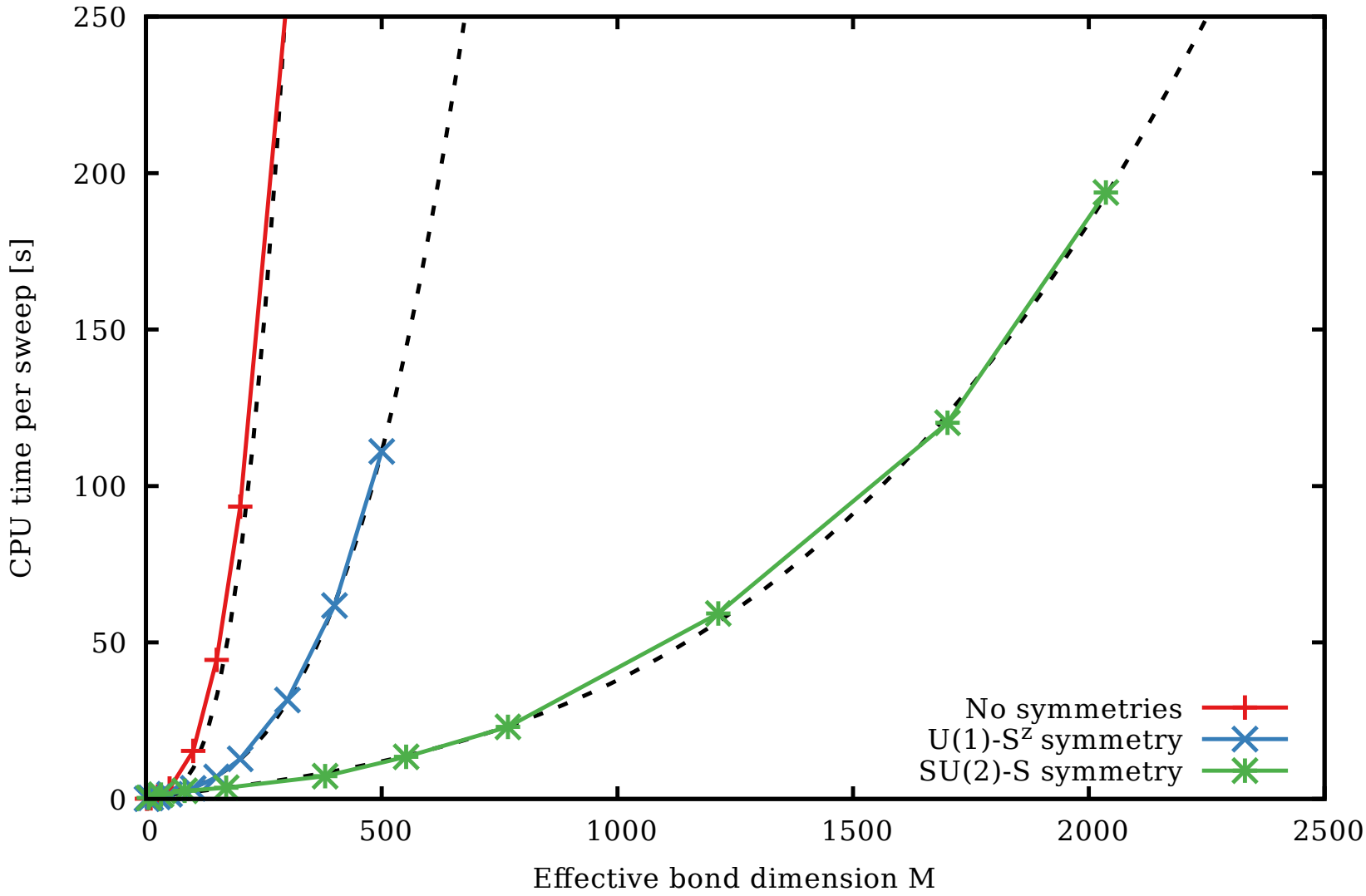
- Not: gauge/"local" symmetries
- Not: space group symmetries (mirror, inversion)
- Not: global symmetries without local eigenstates

Computational speed-up



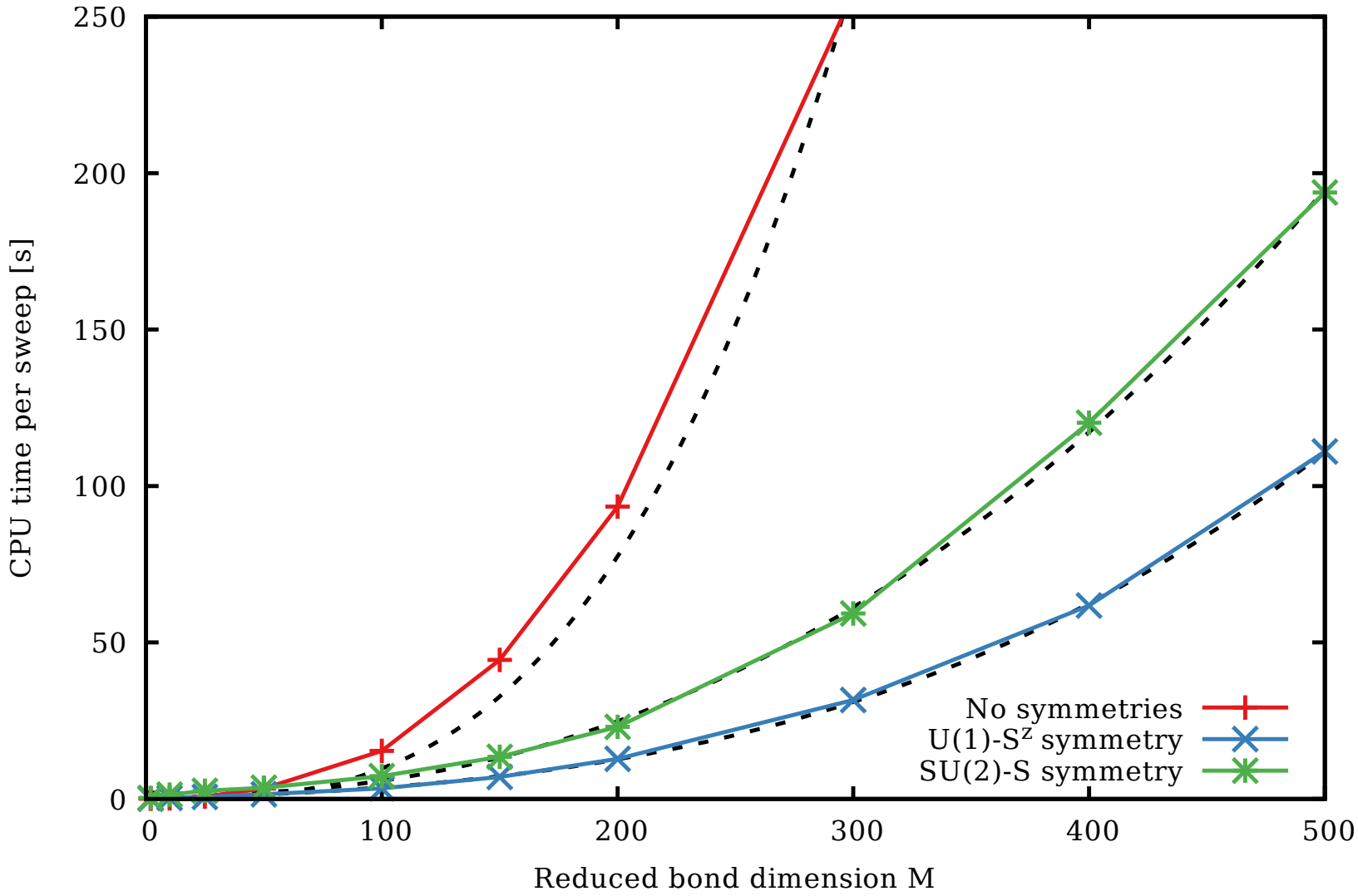
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



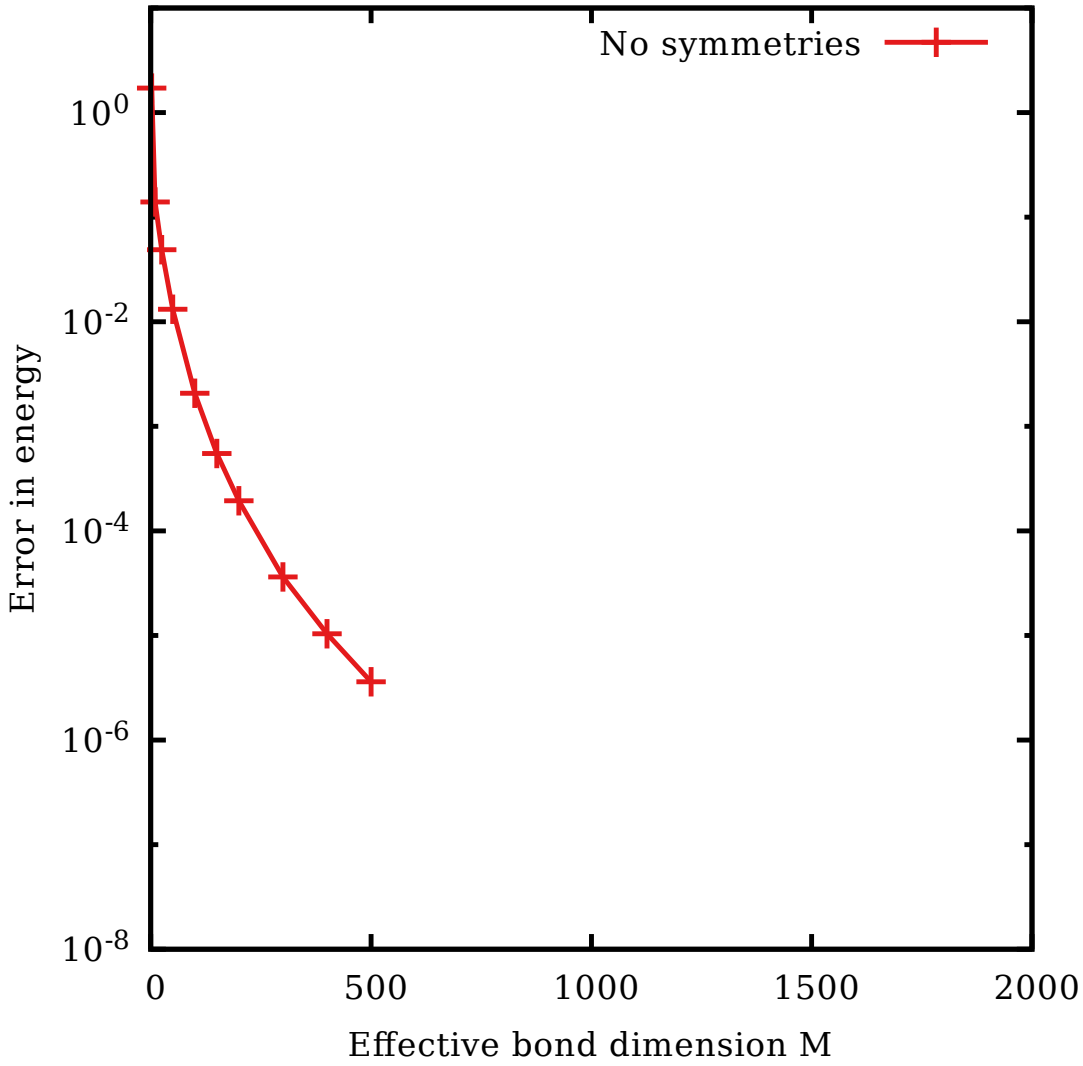
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



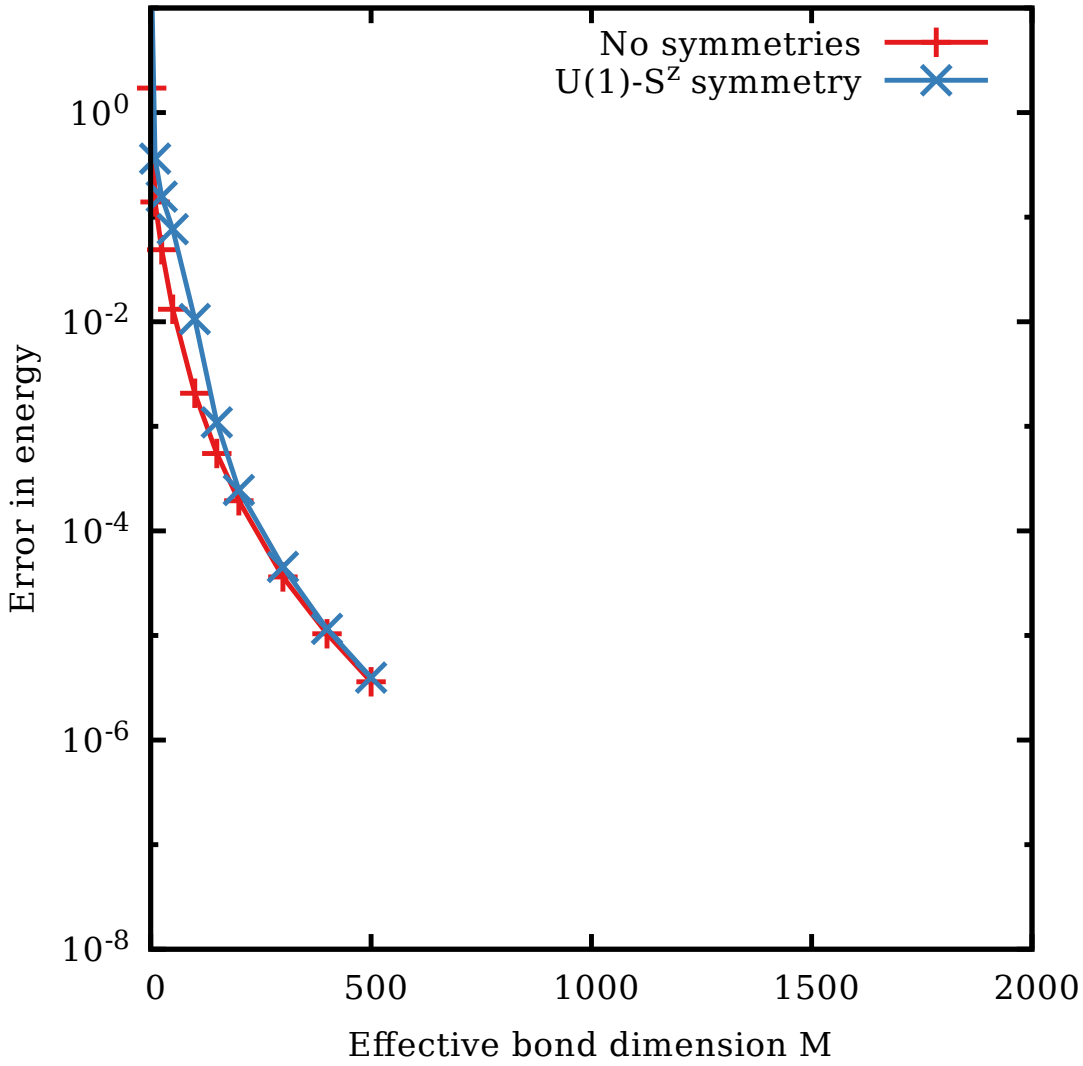
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



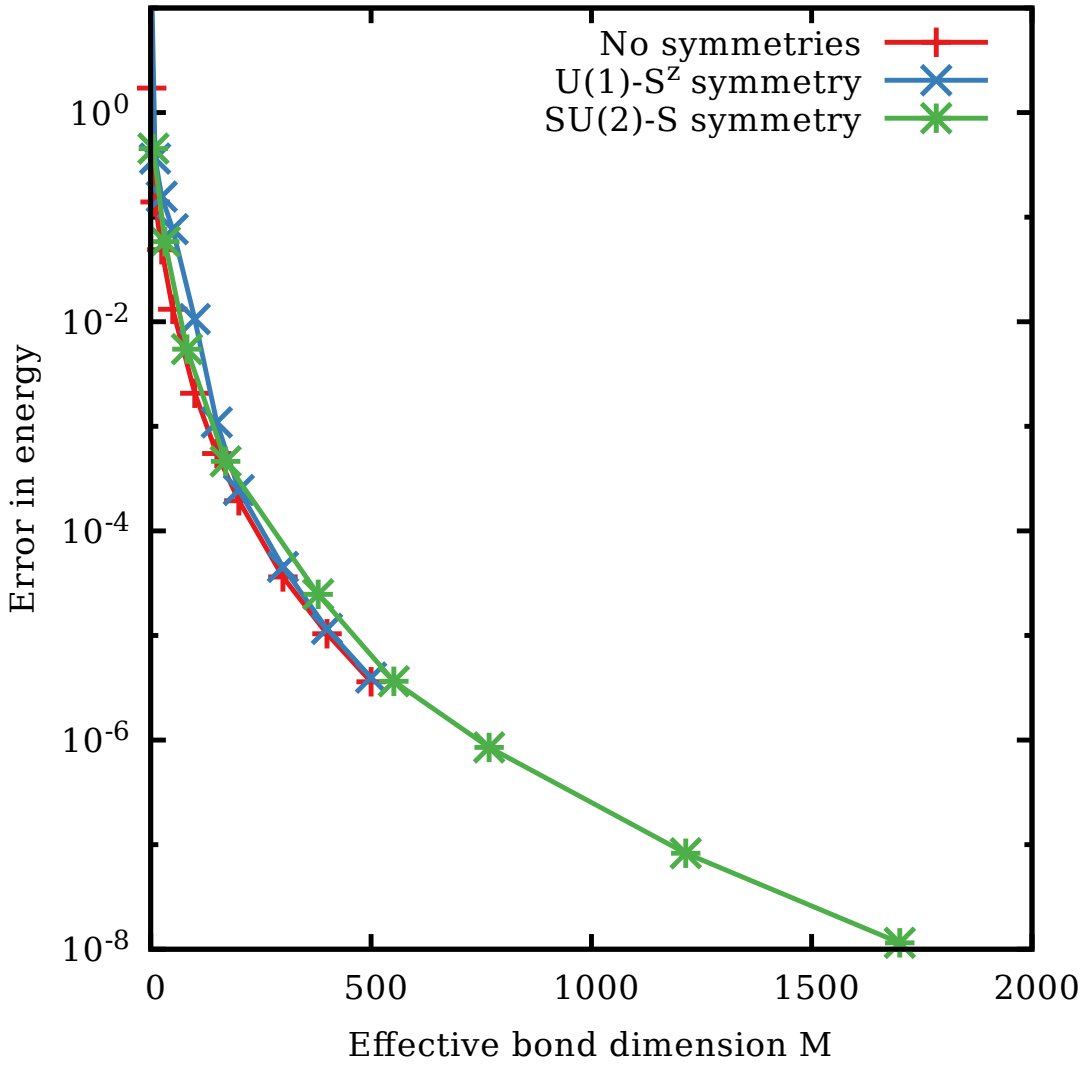
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



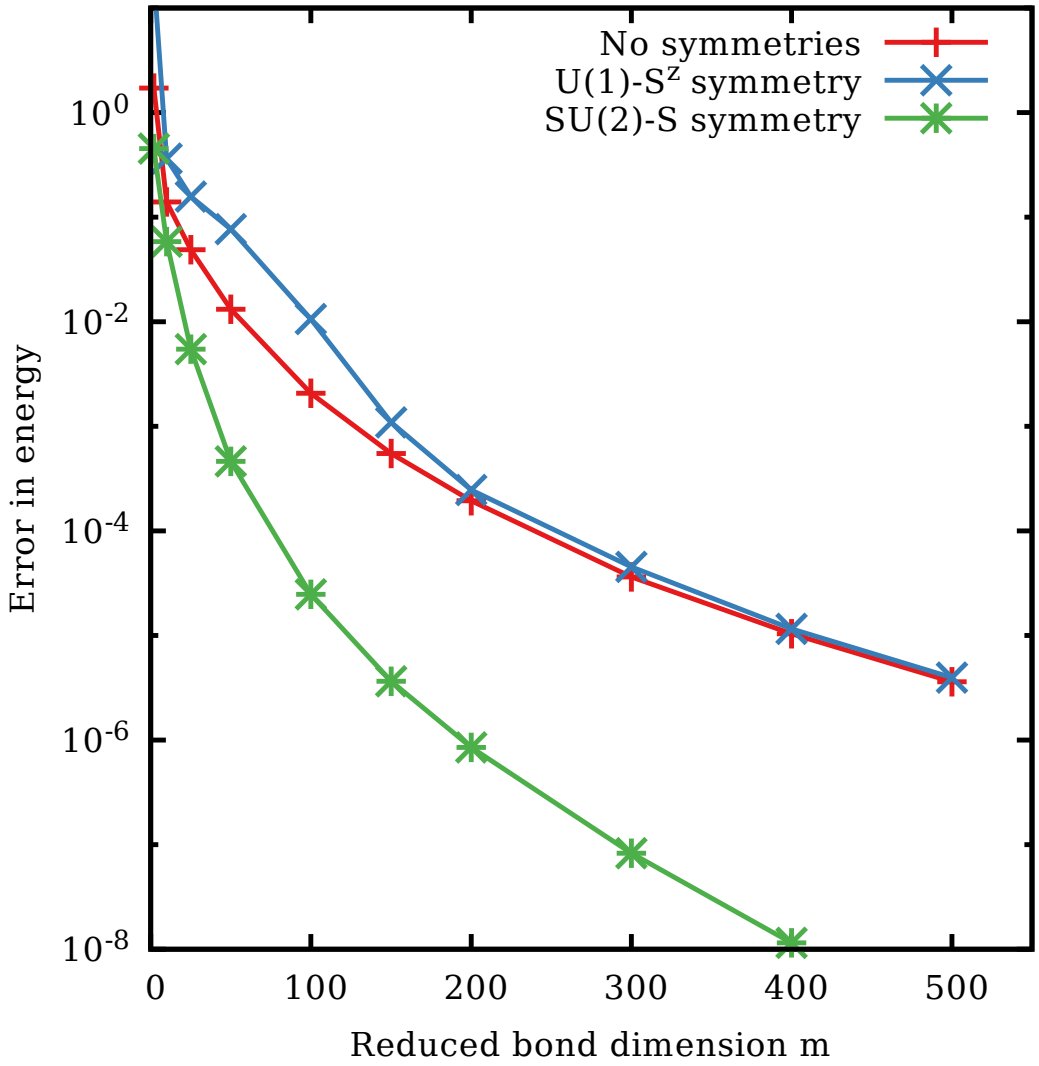
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



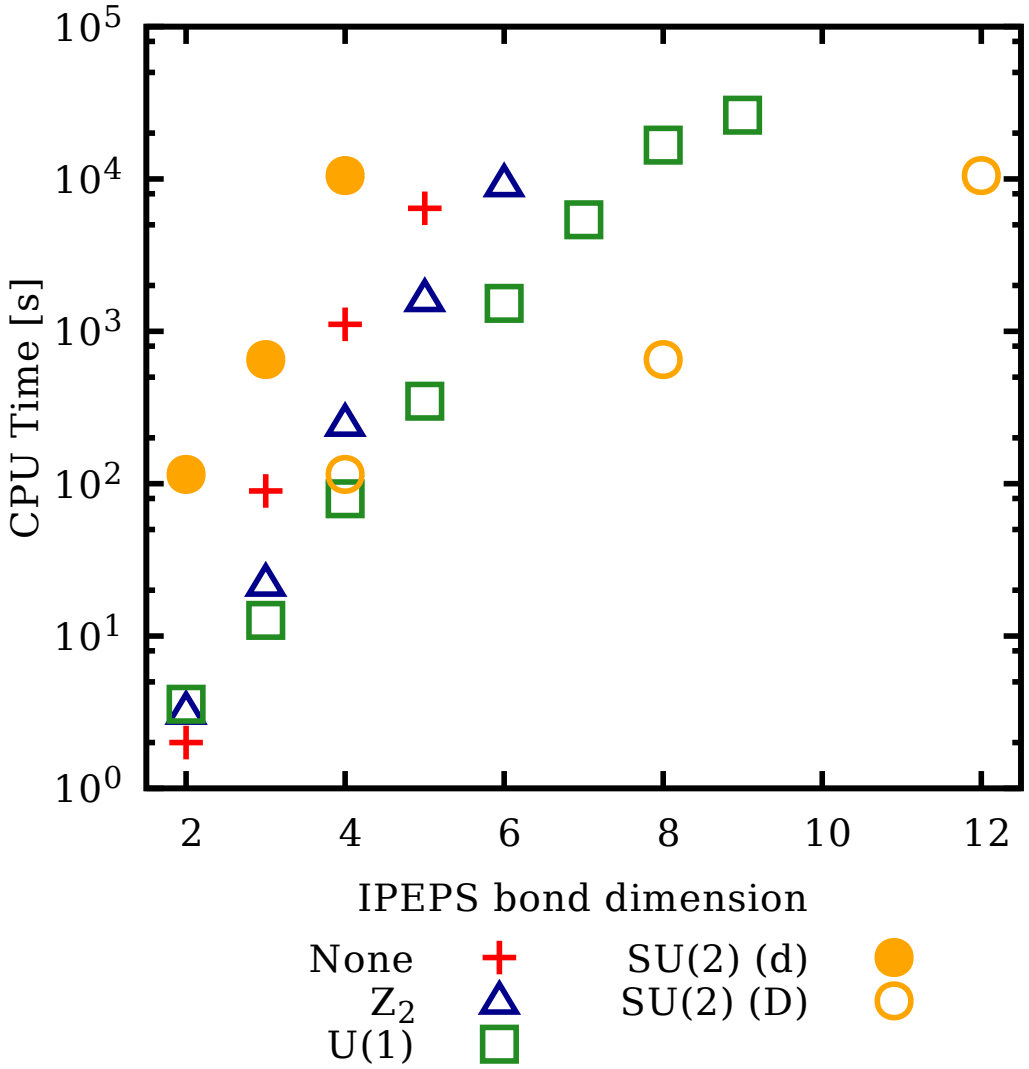
$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



iPEPS on square lattice Heisenberg spins (XXX), fast full update

Selection of quantum number sector

$$\Delta S = E^0_{S=1} - E^0_{S=0}$$

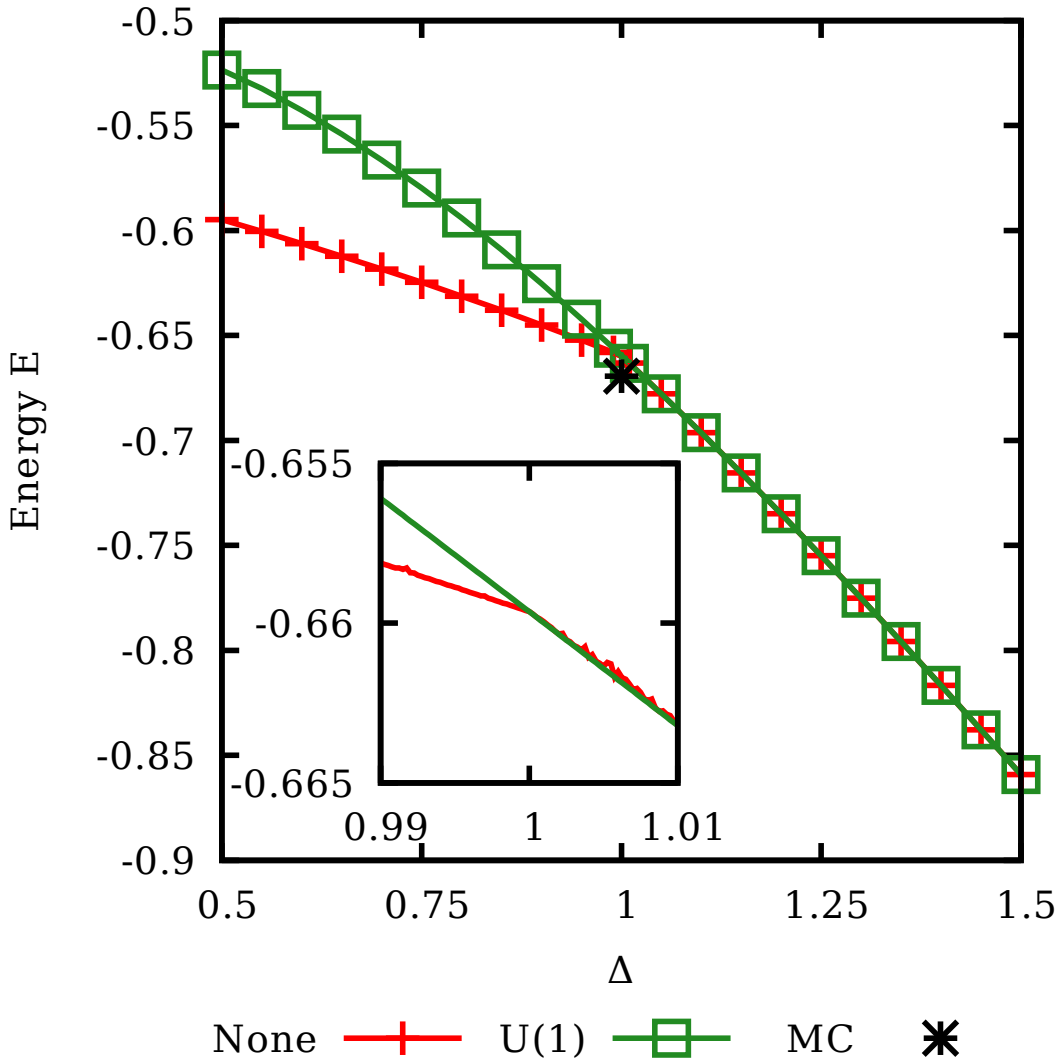
Without symmetries:

1. DMRG for $|0\rangle$
2. DMRG for $|1\rangle$ by orthogonalising against $|0\rangle$ (relatively unstable)

With symmetries:

1. DMRG for $|0\rangle_{S=0}$
1. DMRG for $|0\rangle_{S=1}$

Detection of symmetry breaking



iPEPS on square lattice Heisenberg spins (XX+ΔZ), fast full update, $D = 2$

Why?

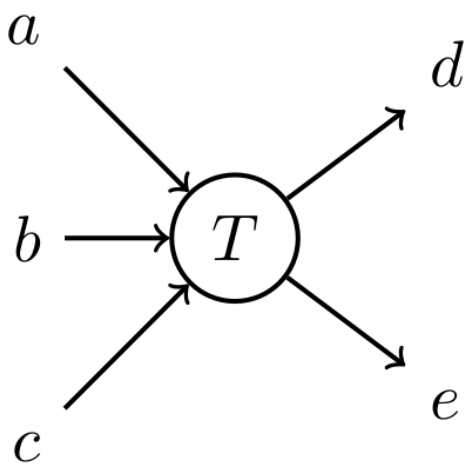
detection of symmetry

stable gap calculations/sector selection

computational speed-up (typically 10x per "step")

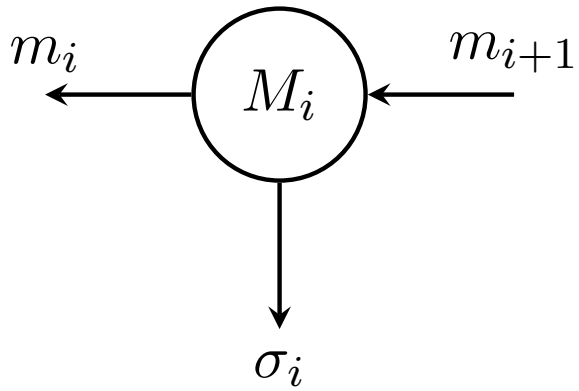
$$T : A \otimes B \otimes C \to D \otimes E$$

$$T : |a\rangle_A \otimes |b\rangle_B \otimes |c\rangle_C \mapsto \sum_{de} T^{de}_{abc} |d\rangle_D \otimes |e\rangle_E$$

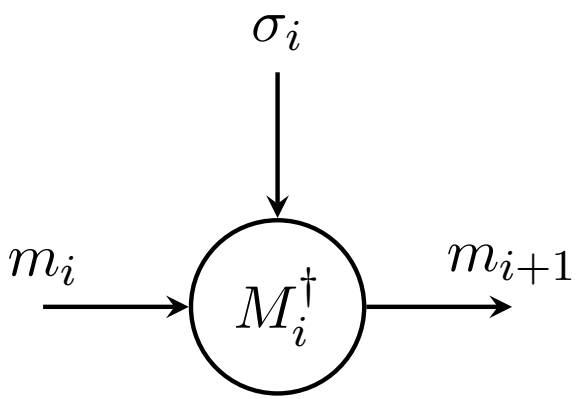


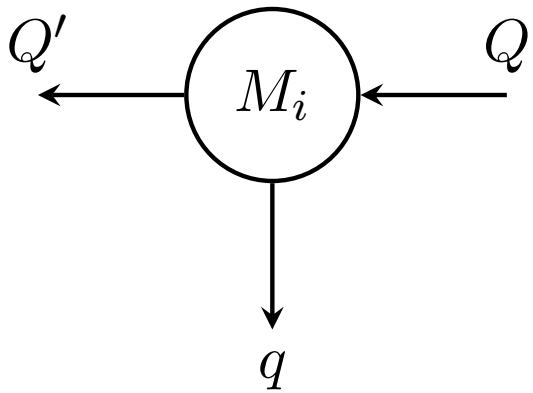
$$T = T^{de}_{abc} |d\rangle |e\rangle \langle c| \langle b| \langle a|$$

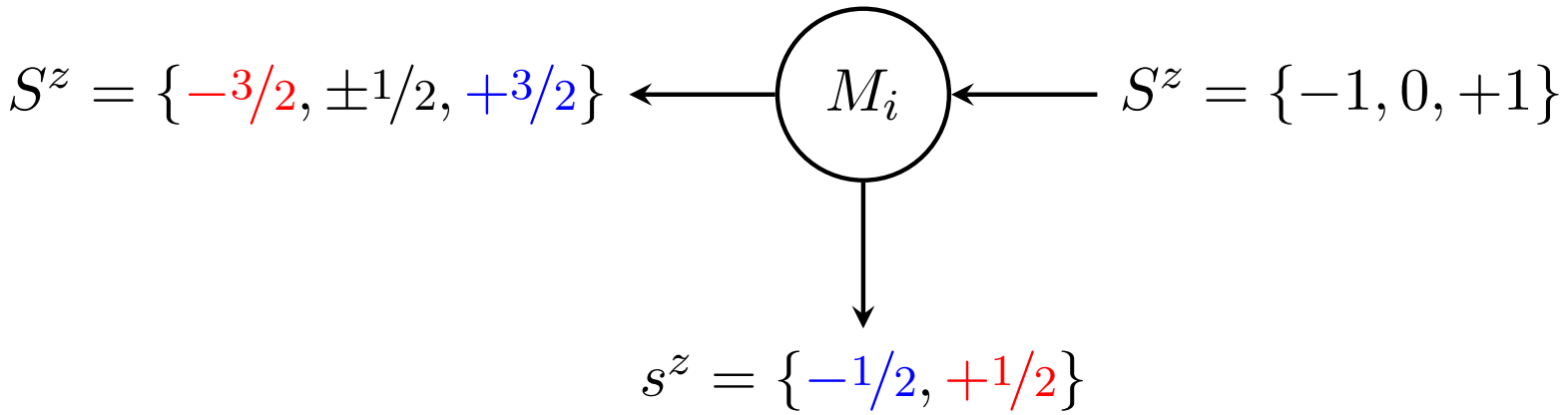
$$M_i = M^{\sigma_i, m_i}_{i; m_{i+1}} |m_i\rangle |\sigma_i\rangle \langle m_{i+1}|$$

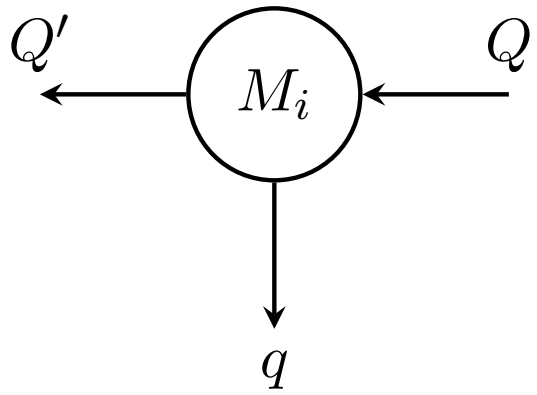


$$M_i^\dagger = \left(M_{i;m_{i+1}}^{\sigma_i, m_i} \right)^\star |m_{i+1}\rangle\langle\sigma_i|\langle m_i|$$




Q'
M_i
Q
q


$S^z = \{-3/2, \pm 1/2, +3/2\}$
M_i
$S^z = \{-1, 0, +1\}$
$s^z = \{-1/2, +1/2\}$



$$Q' + q = Q$$

$$T = \bigoplus_i b_i$$

$$T = \bigoplus_i b_i$$

$$\bigotimes_{x \in \text{inc. legs}} q_i(x) \neq \bigotimes_{y \in \text{out. legs}} q_i(y) \Rightarrow b_i \equiv 0$$

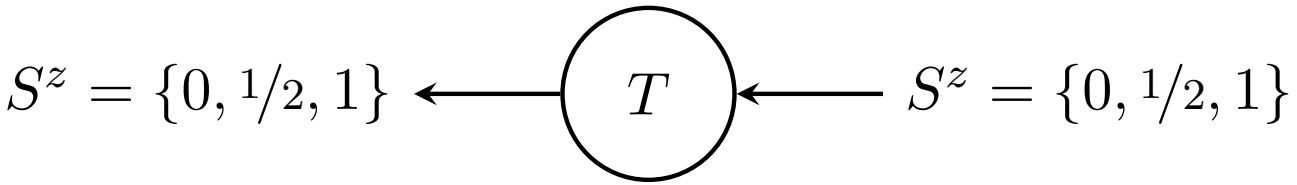
for each symmetry independently!

$$T = \bigoplus_i b_i$$

$$\bigotimes_{x \in \text{inc. legs}} q_i(x) \neq \bigotimes_{y \in \text{out. legs}} q_i(y) \Rightarrow b_i \equiv 0$$

for each symmetry independently!

`Tensor = Vector<Block>`



$$T = b_{0,0} \oplus b_{1/2,1/2} \oplus b_{1,1} = \begin{pmatrix} b_{0,0} & 0 & 0 \\ 0 & b_{1/2,1/2} & 0 \\ 0 & 0 & b_{1,1} \end{pmatrix}$$

```
Tensor = Vector<Block>;


class Block {
  Vector<Sector> qnums;       // one Sector per leg
  ...
};

Sector = Vector<QLabel>;      // one label per symmetry

QLabel = HalfInteger;         // maybe Vec<HalfInteger>
```

```
Tensor = Vector<Block>;


class Block {
  Vector<Sector> qnums;       // one Sector per leg
  DenseTensor reduced;        // dense (numpy-like) tensor
  ...
};


Sector = Vector<QLabel>;      // one label per symmetry


QLabel = HalfInteger;         // maybe Vec<HalfInteger>
```

$$R = T \cdot S$$

$$R = T \cdot S$$

- Sort blocks of T and S into input buckets of same qnums on contracted legs
- worklist W: list of pairs of matching blocks
- sort W by output qnums into worklist buckets
- for each worklist bucket:
  - contract dense tensors of each pair
  - sum up all contractions
  - add block to result tensor R

Additional concerns:

- pre-transpose all blocks?
  trade-off CPU/RAM!
- assemble small blocks to larger dense block to hand off to BLAS?
  BLAS more efficient for larger matrices but parallelisation better for small ones
- optimal data structure to make finding pairs cheap?
  - sorted vector/map for tensor?
  - flat label vector for comparison only?
  - small stack size of blocks?

$$T_{\ldots,l,\ldots} \to Q_{\ldots,l',\ldots} \cdot R^{l'}_{l}$$

$$T_{\ldots,l,\ldots} \to Q_{\ldots,l',\ldots} \cdot R^{l'}_{l}$$

- collect all blocks transforming as q' on specific leg l
- arrange as rows of matrix (column is l)
- QR matrix
- take r as block of R with qnum q'
- decompose q into blocks of Q

$$T = \bigoplus_i b_i$$

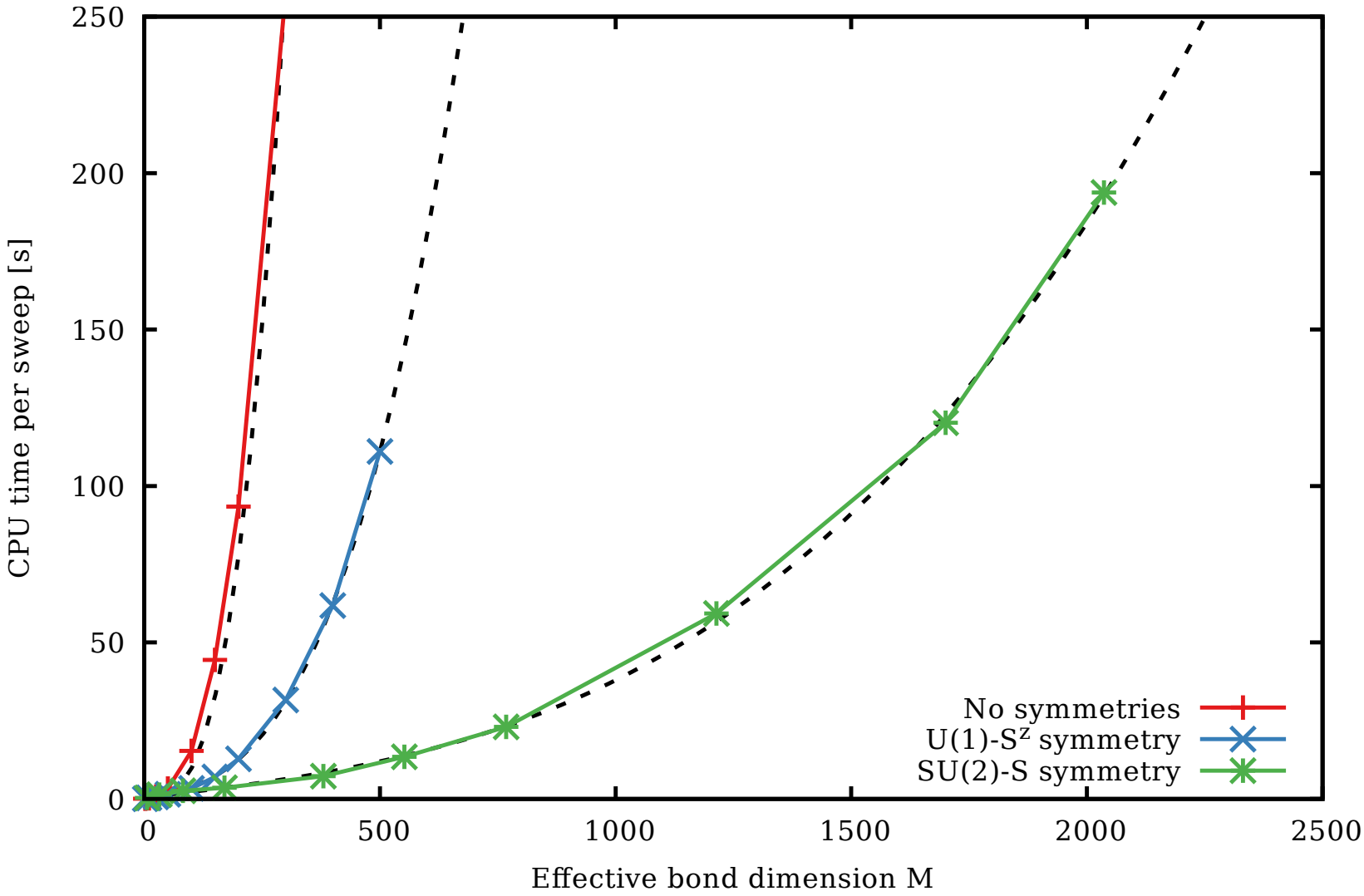
each block b_i:

- one sector per leg with one label per symmetry
- one dense, reduced block
- some extras:
  - transposed block?
  - cache-friendly qnums?
  - overall factor?

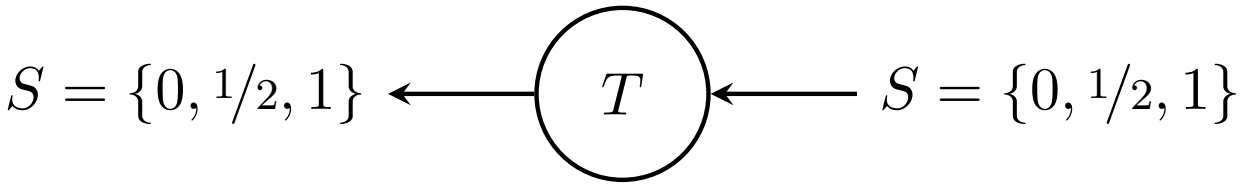
tensor T overall:

- leg directions
- fermionic order?

Computational speed-up



$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion



$$T = b_{0,0} \oplus b_{1/2,1/2} \oplus b_{1,1} = \begin{pmatrix} b_{0,0} & 0 & 0 \\ 0 & b_{1/2,1/2} & 0 \\ 0 & 0 & b_{1,1} \end{pmatrix}$$

$$b_{1/2,1/2} = \begin{pmatrix} b_{+1/2,+1/2} & 0 \\ 0 & b_{-1/2,-1/2} \end{pmatrix}$$

$$b_{+1/2,+1/2} \equiv b_{-1/2,-1/2}$$

$$b_{1,1} = \begin{pmatrix} b_{+1,+1} & 0 & 0 \\ 0 & b_{+0,+0} & 0 \\ 0 & 0 & b_{-1,-1} \end{pmatrix}$$

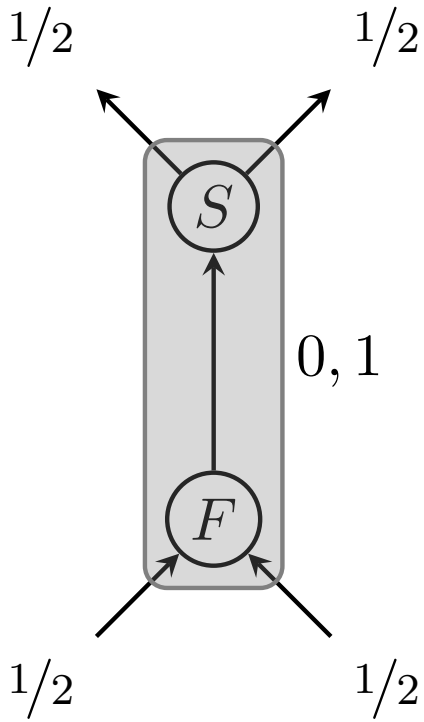
$$b_{+1,+1} \equiv b_{0,0} \equiv b_{-1,-1}$$

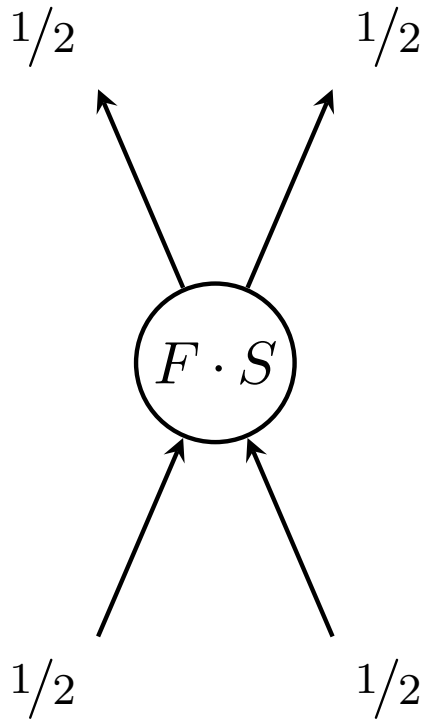
$$b_{S,S} = b_{+S,+S} \bigotimes \mathbf{1}_{2S+1,2S+1}$$

$$b_i = r_i \bigotimes_{\gamma=1}^{N} c_i^{\gamma}$$

$$b_i = r_i \bigotimes_{\gamma=1}^{N} c_i^{\gamma}$$

- for abelian symmetries: c_i^{γ} 1-dim. dummys (0 if block forbidden)
- for non-abelian rank-2: c_i^{γ} proportional to identities
- for non-abelian rank-3: c_i^{γ} proportional to standard Clebsch-Gordan coefficients
- for non-abelian rank-4: c_i^{γ} proportional to generalised CGC


1/2
1/2
S
0, 1
F
1/2
1/2


1/2
1/2
F · S
1/2
1/2

Tensor $F \cdot S$:

```
Tensor of rank 4 with the following bases:
Leg 1: inc Basis (1 Syms): [<1, [SU(2)_S:{+0.5}@2+]>]
Leg 2: inc Basis (1 Syms): [<2, [SU(2)_S:{+0.5}@2+]>]
Leg 3: out Basis (1 Syms): [<1, [SU(2)_S:{+0.5}@2+]>]
Leg 4: out Basis (1 Syms): [<2, [SU(2)_S:{+0.5}@2+]>]
Composed of the following 2 blocks:
```

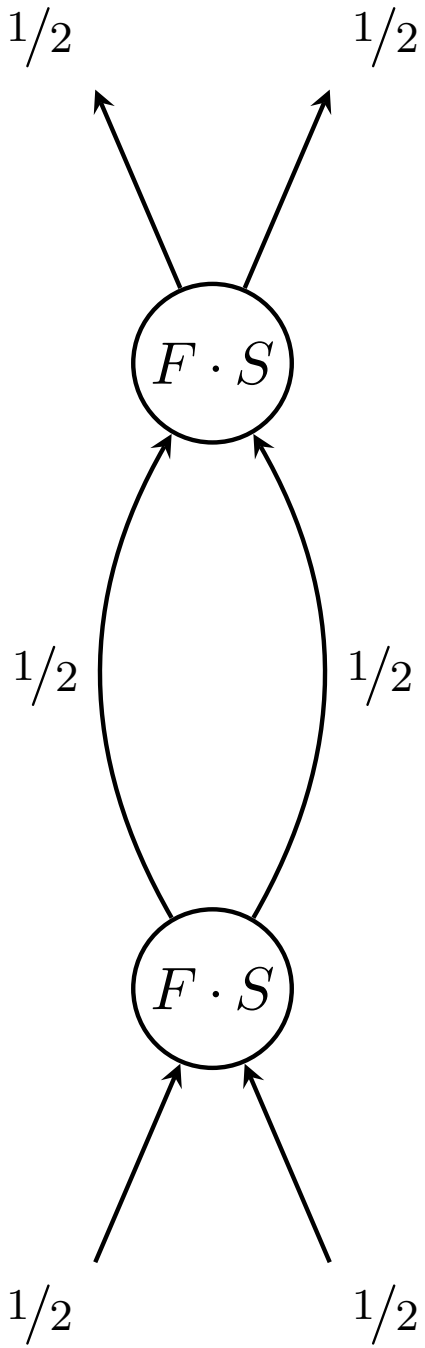
```
Tensor block of rank 4 and 1 symmetries. isZero = +0.
Reduced tensor is: [ST:_]
Start: {1, 2, 1, 2}/4
[0, 0, r, c]
   (+1.0,+0.0)  (+0.0,+0.0)
[0, 1, r, c]
   (+0.0,+0.0)  (+1.0,+0.0)
End: {1, 2, 1, 2}/4
Idx 1: SU(2)_S:{+0.5}@2+;  Idx 2: SU(2)_S:{+0.5}@2+;
Idx 3: SU(2)_S:{+0.5}@2+;  Idx 4: SU(2)_S:{+0.5}@2+
```

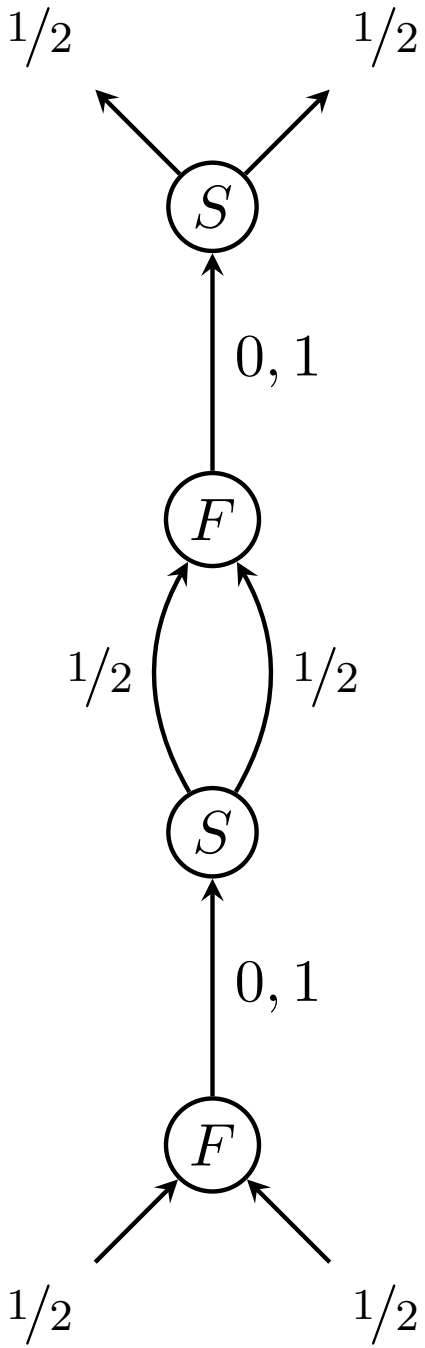
```
CGC space 1 is:({Sparse<4> in {2, 2, 2, 2}:
  [{0, 0, 0, 0} => +1.0] [{1, 0, 1, 0} => +0.5]
  [{0, 1, 1, 0} => +0.5] [{1, 0, 0, 1} => +0.5]
  [{0, 1, 0, 1} => +0.5] [{1, 1, 1, 1} => +1.0]})
```

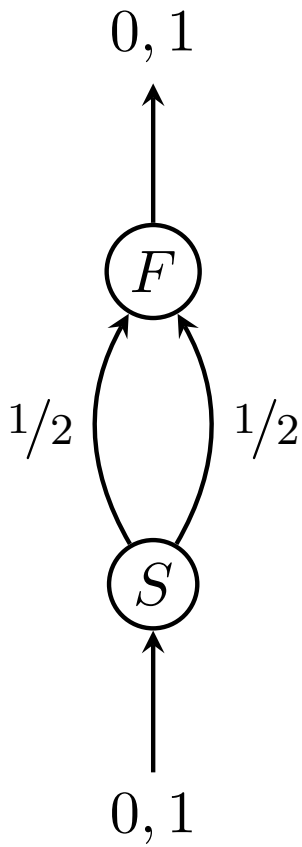
Block 1: $1/2 \times 1/2 \to 1 \to 1/2 \times 1/2$

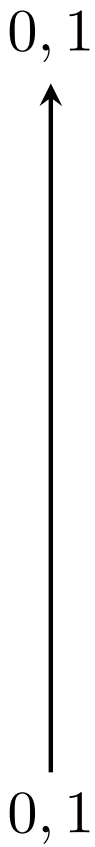
```
Tensor block of rank 4 and 1 symmetries. isZero = +0.
Reduced tensor is: [ST:_]
Start: {1, 2, 1, 2}/4
[0, 0, r, c]
   (+1.0,+0.0)  (+0.0,+0.0)
[0, 1, r, c]
   (+0.0,+0.0)  (+1.0,+0.0)
End: {1, 2, 1, 2}/4
Idx 1: SU(2)_S:{+0.5}@2+;  Idx 2: SU(2)_S:{+0.5}@2+;
Idx 3: SU(2)_S:{+0.5}@2+;  Idx 4: SU(2)_S:{+0.5}@2+

CGC space 1 is:({Sparse<4> in {2, 2, 2, 2}:
  [{1, 0, 1, 0} => +0.5] [{0, 1, 1, 0} => -0.5]
  [{1, 0, 0, 1} => -0.5] [{0, 1, 0, 1} => +0.5]})
```

Block 2: $1/2 \times 1/2 \to 0 \to 1/2 \times 1/2$


1/2
1/2
F · S
1/2
1/2
F · S
1/2
1/2

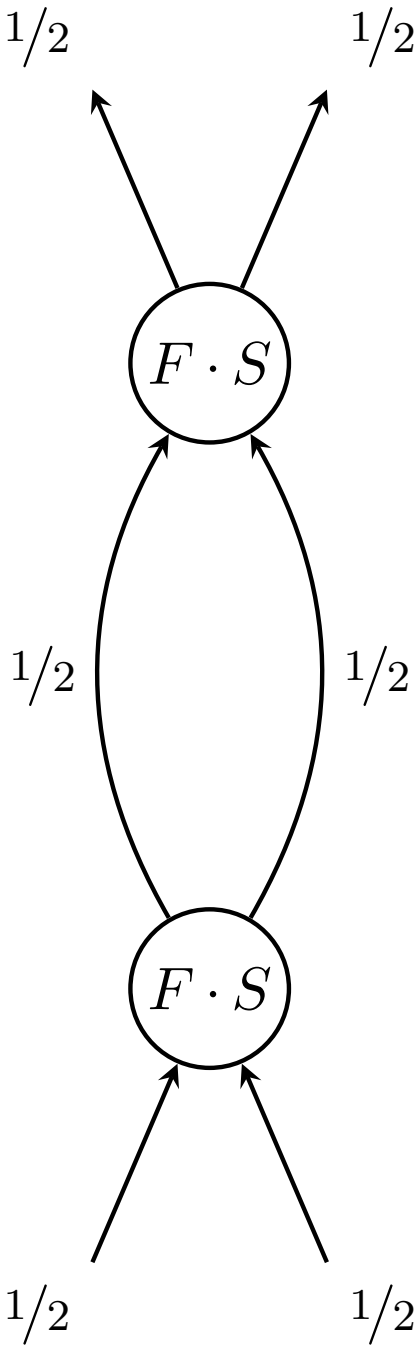

1/2
1/2
S
0, 1
F
1/2
1/2
S
0, 1
F
1/2
1/2

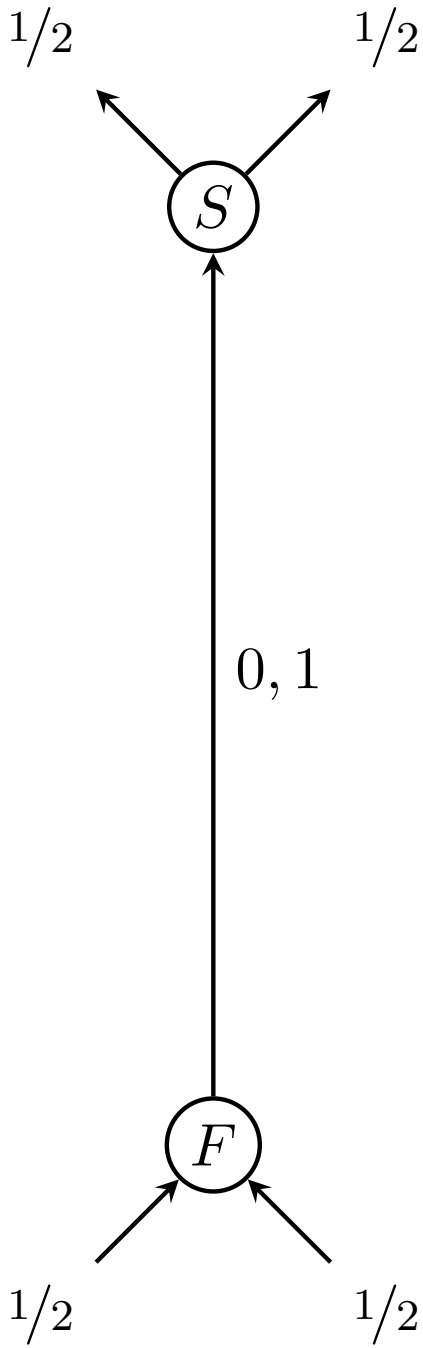




Clebsch-Gordans in $(0, 1/2, 1/2) \times (1/2, 1/2, 1)$ cancel!

$\Rightarrow 2 \times 2$ blocks just give 2 blocks


1/2
1/2
F · S
1/2
1/2
F · S
1/2
1/2


1/2
1/2
S
0, 1
F
1/2
1/2

- construct rank-3 as fuses/splits of two bases
  numerical evaluation of Clebsch-Gordans for all symmetries
- construct initial higher-rank from rank-3 tensors
- (potentially) define MPO tensors by hand

A. Weichselbaum, Non-abelian symmetries in tensor networks, Annals of Physics 327 (2012), Appendix B
CH, Symmetry-Protected Tensor Networks, PhD thesis, LMU 2017, Sec. 2.2.3

```
Tensor = Vector<Block>;


class Block {
  Vector<Sector> qnums;       // one Sector per leg
  DenseTensor reduced;        // dense (numpy-like) tensor
  Vector<SparseTensor> cgcs; // one sparse CGC per symmetry
  ...
};


Sector = Vector<QLabel>;     // one label per symmetry


QLabel = HalfInteger;        // maybe Vec<HalfInteger>
```

In tensor products:

- contract r_i and c_i^γ independently
- add up only blocks with parallel c_i^γ

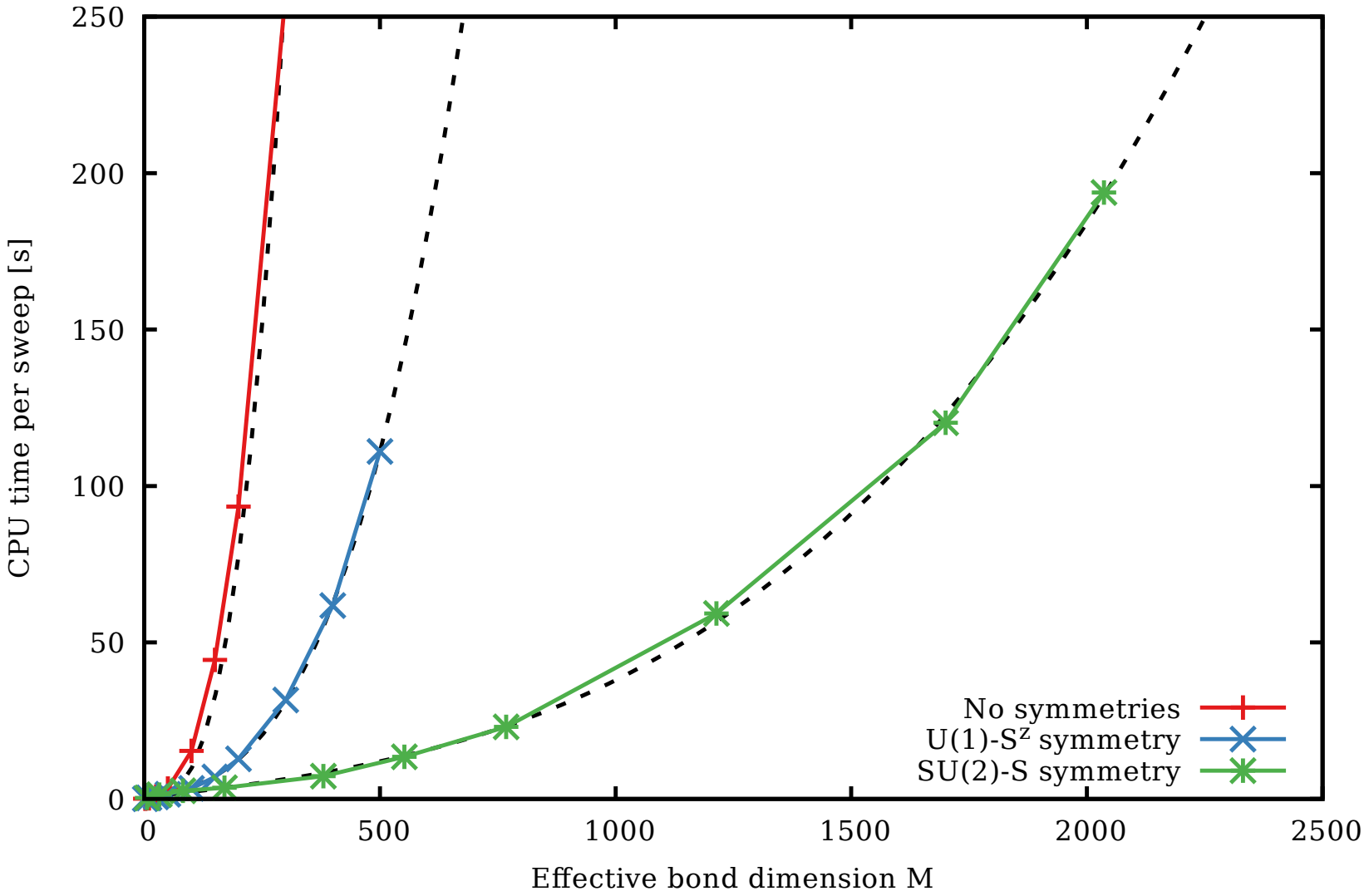
In SVDs/QRs etc:

- CGCs contracted over all but one leg are proportional to identity
- for QRs: ignore, copy from T to Q, initialise R as identities
  scale blocks of Q by factor *after* QR
- for SVDs: scale dense coefficients to make up for factor and obtain right singular values

Numerical inaccuracies: errors accumulate in c_i^γ

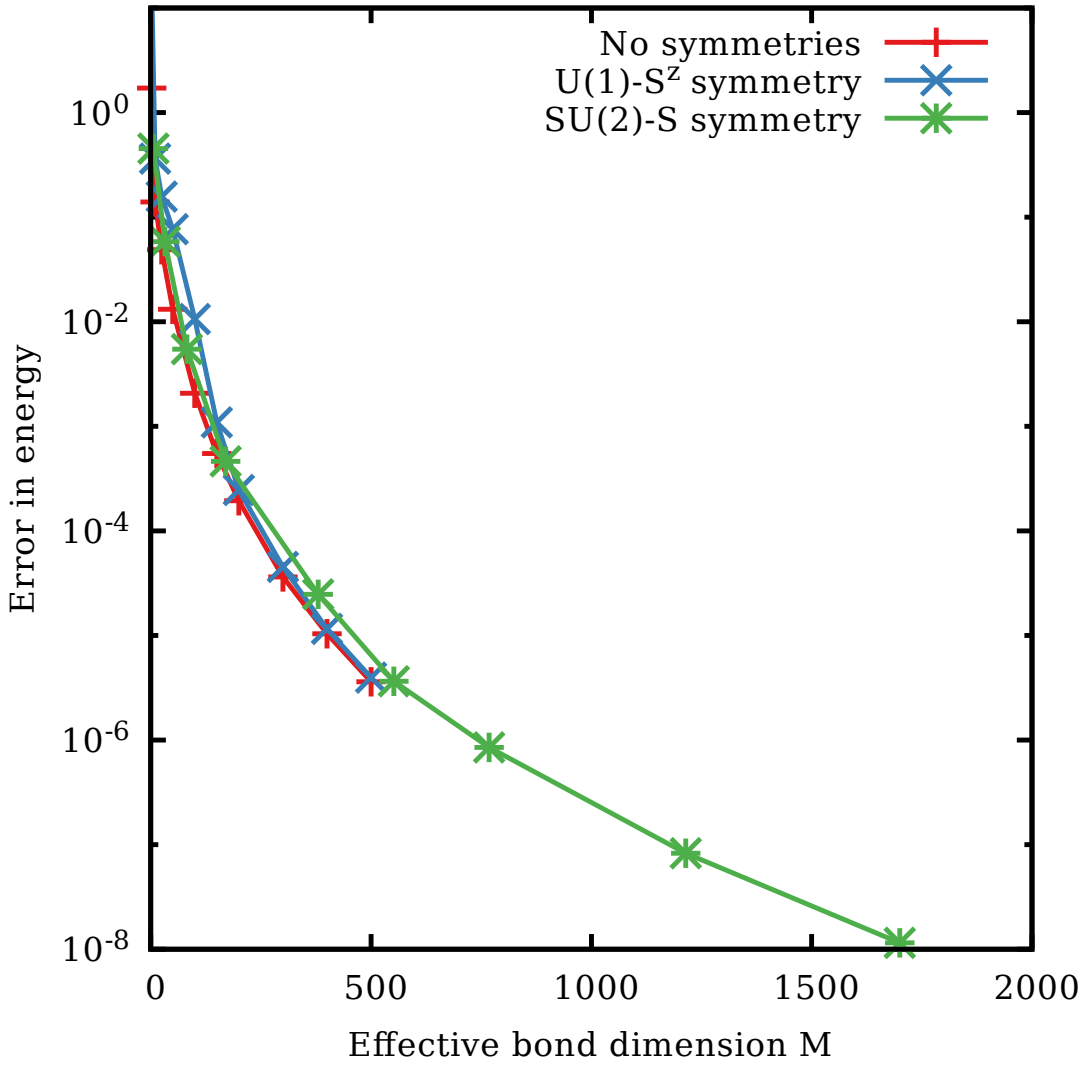
- avoid unnecessary operations on c_i^γ
- regularise: $[-1.001, 0.999, 1.01] \to [-1, 1, 1]$
- reconstruct: merge all incoming & outgoing legs, fixup CGCs, split up
- minimise tensor ranks where possible (also reduces number of blocks)

Computational speed-up



$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Computational speed-up



$L = 100$ Heisenberg chain (XXX), PBC in MPO, 1DMRG with subspace expansion

Convergence problems amplified if symmetries implemented

- choose sectors optimally & flexibly! (DMRG3S, 2-site full update etc.)
- include additional noise terms (even entirely random)
- ... or tunnelling Hamiltonians
- check *all* relevant sectors for ground state
- during SVD: distribute states to sectors according to singular values

How?

$$T = \bigoplus_i \left(r_i \bigotimes_{\gamma=1}^{N} c_i^\gamma \right)$$

- each block stores quantum numbers of every symmetry on every leg
- only symmetry-allowed blocks non-zero
- structure of c_i^γ handles inner multiplicity
- different c_i^γ for outer multiplicity
- tensor products:
  find matching blocks, contract r_i and c_i^γ independently
- decompositions:
  find all blocks with same qnums on leg, construct matrix, decompose
- keep convergence issues in mind

Existing implementations

inclusion criteria: has both a website and symmetries and I could find it

Abelian symmetries:

- ITensor (itensor.org, cite)
- Uni10 (uni10.gitlab.io, LGPL v3, non-abelian work in progress?)
- TeNPy (tenpy.github.io, GPL v3)
- Tensor Network Theory (tensornetworktheory.org, ?)
- OpenSourceMPS (is.gd/zIOLKI, cite)

SU(2), MPS:

- Matrix Product Toolkit (is.gd/UOZMNK, cite)
- CheMPS2 (is.gd/seiiqa, arXiv:1312.2415, GPL v2 or later)

SU(N), arbitrary rank:

-  SyTen (syten.eu, within collaborations)

- DMRG & MPS:
  I. P. McCulloch: PhD thesis, 2002 *and* J. Stat. Mech. 2007-10
- explicit CGC:
  A. Weichselbaum: Ann. of Phys. 327, 2012;
  CH: PhD thesis, LMU, 2017
- implicit CGC:
  S. Singh et al: PRA 82, 2010; PRB 83, 2011; PRB 86, 2012
- iPEPS:
  Liu et al, PRB 91, 2015; CH, 1808.10804
- MPOs:
  CH et al, PRB 95, 2017
- DMRG convergence:
  S. R. White, PRB 72, 2005; CH et al: PRB 91, 2015
- Fermions:
  T. Barthel et al, PRA 80, 2009; N. Bultinck et al, PRB 2017

claudius-hubig.eu/talks/201811_dresden.pdf