Simulating Anyonic Spin Chains With TensorKit.jl

Kevin Vervoort

Quantum Group

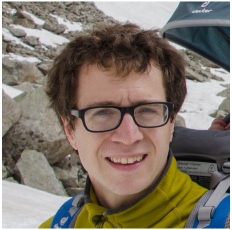
Ghent University

November 30, 2023

Introduction

- Interested in quantum many-body physics through tensor networks
- Various Julia packages to handle large-scale tensor network calculations
- Using Julia to simulate models with categorical symmetries

TensorKit.jl



Jutho Haegeman

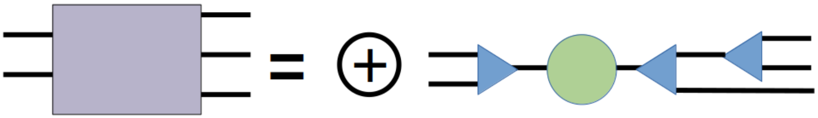
Lukas Devos

TensorKit.jl

- Main feature is symmetric tensor networks
- Symmetries are very important as they provide a lot of structure to the problem
- Fully utilizing the symmetries is very beneficial for both the theoretical understanding and numerical simulation
- Idea: Impose symmetries directly on the tensors.

Symmetric Tensor Networks

- Decompose every tensor in a structure part and a data part
- The structure part is constructed from the data of the symmetry e.g. Fusion rules and F-symbols
- Data part contains the numerical values of the components
- Huge reduction in parameter space
- Currently many groups, but also Fibonacci and Ising category



CategoryData.jl

Lukas Devos

Jacob Bridgeman

CategoryData.jl

- Adds extra possible categorical symmetries to TensorKit.jl
- Includes data for multiplicity free unitary (braided) fusion categories up to rank 6
- Can be used as resource to look up data like F-symbols, R-symbols
- Allows for exploring many interesting systems with exotic symmetries by using tensor networks

MPSKit.jl and MPSKitModels.jl

Maarten Van Damme

Lukas Devos

MPSKit.jl and MPSKitModels.jl

MPSKit.jl

- Implementations of (in)finite MPS and MPO using TensorKit.jl
- Includes MPS-algorithms like DMRG, VUMPS, ...

MPSKitModels.jl

- Implementations of model Hamiltonians for MPSKit.jl
- Convenient functions to define various Hamiltonians on different lattices

Anyonic Spin Chain

- Generalization of spin chains with anyonic degrees of freedom
- Provides a rich class of interacting models
- Based on an input fusion category
- Hilbert space consists of allowed fusiontrees

Golden Chain[1]

- Simplest anyonic spin chain
- Based on the input fusion category **Fib**
- Fixes all outgoing labels of the fusion trees on the object τ
- Exhibits a critical phase corresponding to the Tricritical Ising CFT

[1] Feiguin, A., Trebst, S., Ludwig, A., Troyer, M., Kitaev, A., Wang, Z., & Freedman, M. (2007). Interacting Anyons in Topological Quantum Liquids: The Golden Chain. Phys. Rev. Lett., 98, 160409.

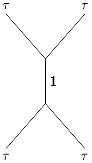
Golden Chain: Choosing input category

```
using TensorKit, MPSKit, MPSKitModels, Plots, Polynomials, CategoryData
D = 70; # Bond dimension

#Define input category
𝒞 = Fib
# Objects of the input category
objects = Object{𝒞}
dimsum = sum(dim, values(objects))
```

Golden Chain: Hamiltonian

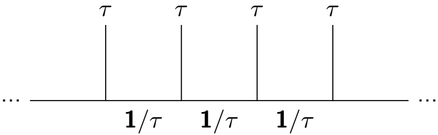
$$H = \sum_i O_{i,i+1} \tag{1}$$

$$O_{i,j} = - \tag{2}$$




```
#Physical (outer) leg space
P = Vect[objects](2=>1)

# Define local Hamiltonian term
O = TensorMap(ones, P ⊗ P ← P ⊗ P)
blocks(O)[objects(1)] *= -1
blocks(O)[objects(2)] *= 0

# Define Hamiltonian on infinte chain
H = @mpoham sum(O{i,j} for (i,j) in nearest_neighbours(InfiniteChain(1)))
```

Golden Chain: MPS-Ansatz

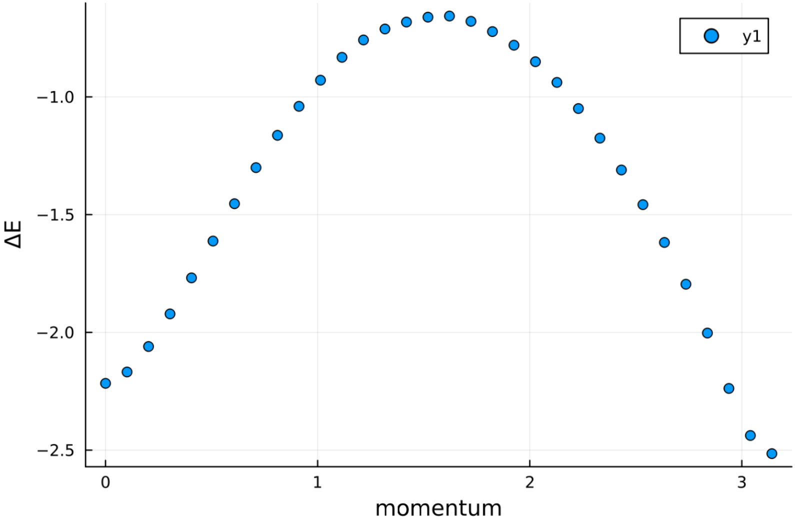



```
# Define virtual space of total dimension D
V = Vect[objects](1=>round(D/dimsum), 2=>round(D/dimsum))

# Define an initial random uniform MPS ansatz
ψ₀ = InfiniteMPS([P], [V])
```

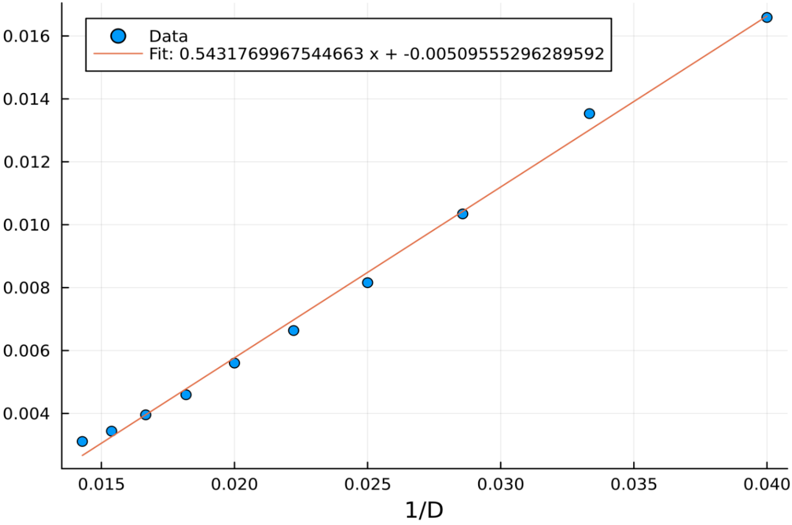
Golden Chain: Groundstate and Excitations

```
# Find groundstate starting from our initial MPS
ψ_gs, envs_gs, _ = find_groundstate(ψ₀, H, VUMPS(maxiter=1000,verbose=true))

# Define momentum space
kspace = range(0, π, 32)

# Find first excited state at every point in kspace
Es, _ = excitations(H, QuasiparticleAnsatz(), kspace, ψ₀, envs_gs)

# Find the energy gap of the system
ΔE, _ = findmin(real.(Es))

# Plot dispersion relation
plot(kspace, real.(Es); xaxis="momentum", yaxis="ΔE", seriestype=:scatter)
```

Golden Chain: Dispersion Relation



Golden Chain: Gap extrapolation

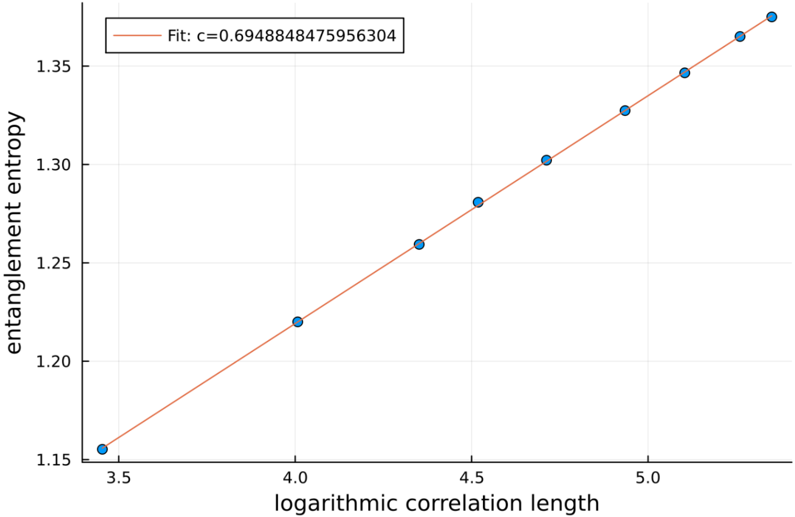


Golden Chain: Central Charge

$$S \propto \frac{c}{6} \log \xi \tag{3}$$

```
S = real(entropy(ψ_gs)[1])
ξ = correlation_length(ψ_gs)
```

Golden Chain: Central Charge



Haagerup Chain[2]

- Based on the Haagerup fusion category $\mathcal{H}_3$
- Has 6 objects $\{\alpha, \alpha^2, \alpha^3, \rho, \alpha\rho, \alpha^2\rho\}$
- Exotic fusion category
- To simulate reuse exact the same code but just change the input category

[2] Huang, T.C., Lin, Y.H., Ohmori, K., Tachikawa, Y., & Tezuka, M. (2022). Numerical Evidence for a Haagerup Conformal Field Theory. Phys. Rev. Lett., 128, 231603.

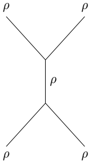
Haagerup Chain: What to change?

```
using TensorKit, MPSKit, MPSKitModels, Plots, Polynomials, CategoryData
D = 70; # Bond dimension

#Define input category
𝒞 = H3
# Objects of the input category
objects = Object{𝒞}
dimsum = sum(dim, values(objects))
```

Haagerup Chain: Hamiltonian

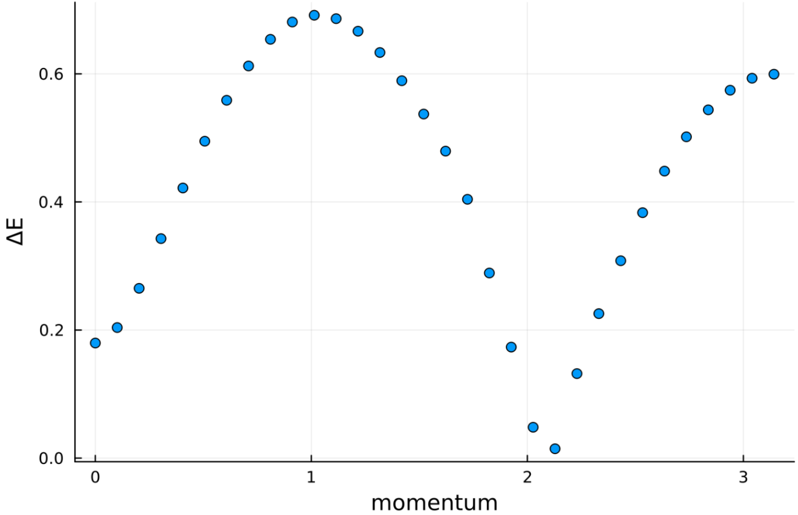
$$H = \sum_i O_{i,i+1} \tag{4}$$

$$O_{i,j} = - \tag{5}$$

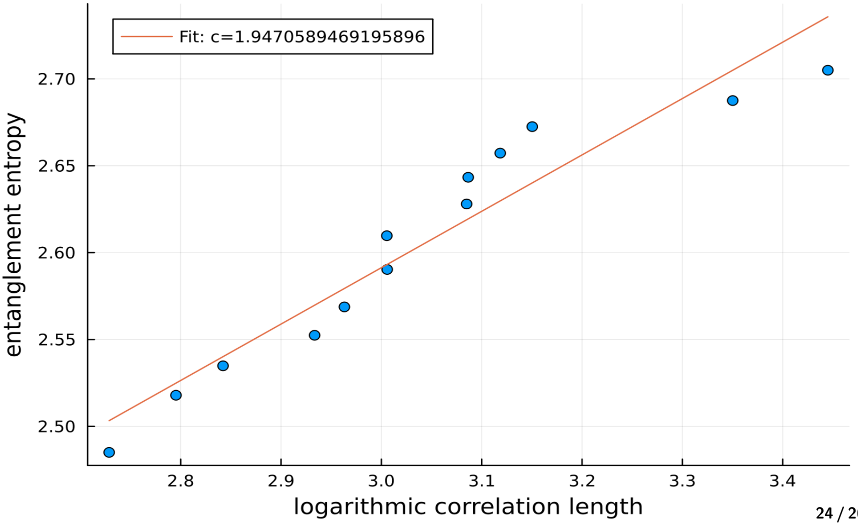



```
#Physical (outer) leg space
P = Vect[objects](4=>1)

# Define local Hamiltonian term
O = TensorMap(ones, P ⊗ P ← P ⊗ P)
blocks(O)[objects(1)] *= 0
blocks(O)[objects(4)] *= -1
blocks(O)[objects(5)] *= 0
blocks(O)[objects(6)] *= 0

# Define Hamiltonian on infinte chain
H = @mpoham sum(O{i,j} for (i,j) in nearest_neighbours(InfiniteChain(1)))
```

Haagerup Chain: Dispersion Relation



Haagerup Chain: Central Charge



Conclusion

- TensorKit.jl implements symmetric tensor networks, giving siginificant speedups
- CategoryData.jl provides implementations of many different categories
- MPSKit.jl and MPSKitModels allow for investigating of various models with categorical symmetries

Reach Out

Kevin.Vervoort@ugent.be

`https://quantumghent.github.io`

Packages:

- TensorKit.jl: `https://github.com/Jutho/TensorKit.jl`
- CategoryData.jl: `https://github.com/lkdvos/CategoryData.jl`
- MPSKit: `https://github.com/maartenvd/MPSKit.jl`
- MPSKitModels: `https://github.com/maartenvd/MPSKitModels.jl`