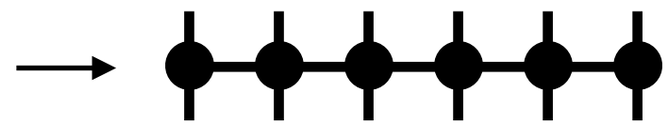
ITensors.jl and Running DMRG

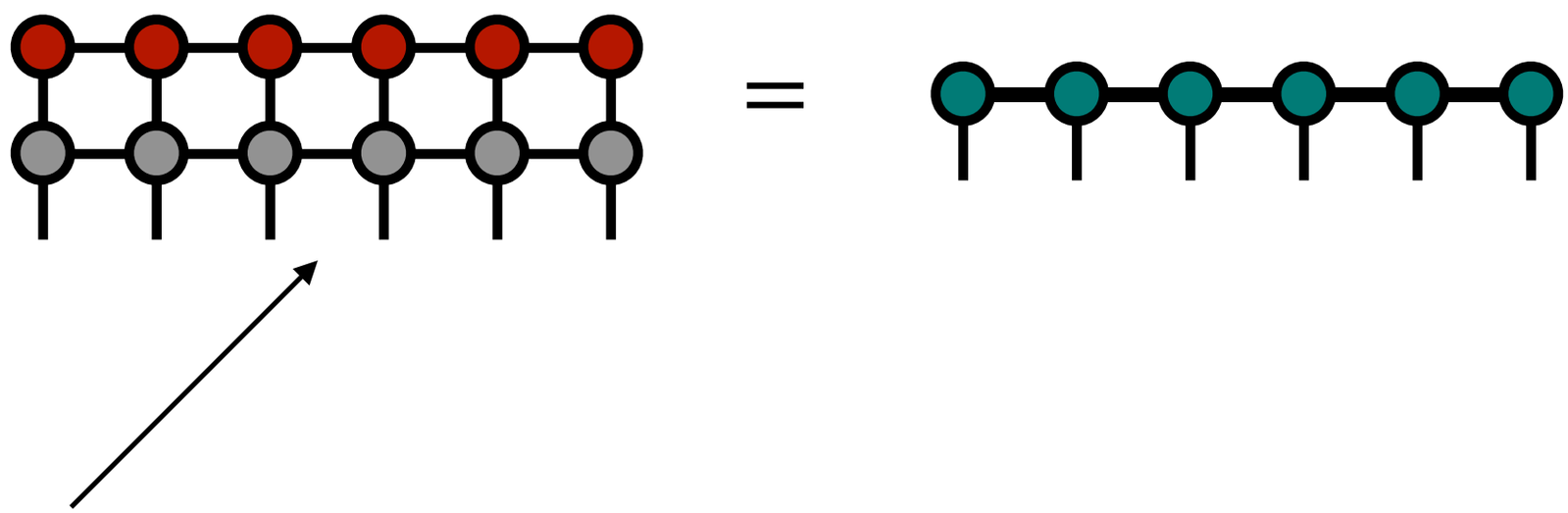
$$\sum_j S^z_j S^z_{j+1} \longrightarrow$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```



This Talk

- Making MPO's using OpSum
- Running DMRG and TDVP
- Computing observables from MPS



Matrix product operator (MPO) network

An MPO is to an MPS as a
matrix is to a vector

MPS Algorithms

We offer MPS and "MPO" algorithms through the ITensorMPS package

```
using ITensors
using ITensorMPS
```

MPS Algorithms

The ITensorMPS package offers many helpful algorithms for working with MPS and MPO's

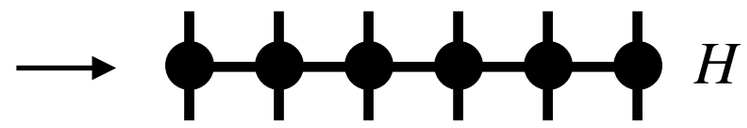
- OpSum system – making MPOs from operators
- dmrg, apply, tdvp – computing ground states and dynamics
- expect – computing expected values of operators
- correlation_matrix – compute correlation functions
- inner – overlap MPS and MPOs
- contract, sum – algebra of MPS and MPO

MPS Algorithms

OpSum – powerful "domain-specific language" (DSL)
for making MPOs from math expressions

$$\sum_j S^z_j S^z_{j+1} \longrightarrow$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```



MPS Algorithms

Starting from beginning: first make an array of "sites"
Just a Julia array of Index objects

```
sites = siteinds("S=1/2",N)
```

MPS Algorithms

Next fill up OpSum with "terms" of the operator

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end
```

Internal data of "terms" similar to:

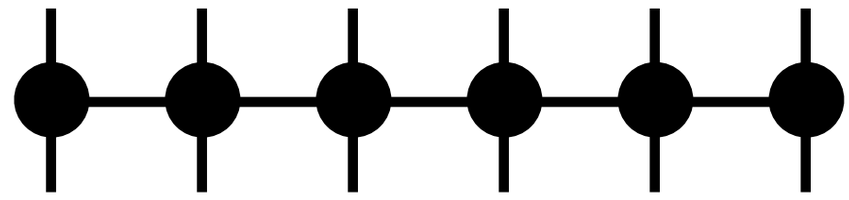
(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

MPS Algorithms

Finally, construct MPO – terms are compressed [1,2]

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$H =$ 

Optimal bond dimension typically reached

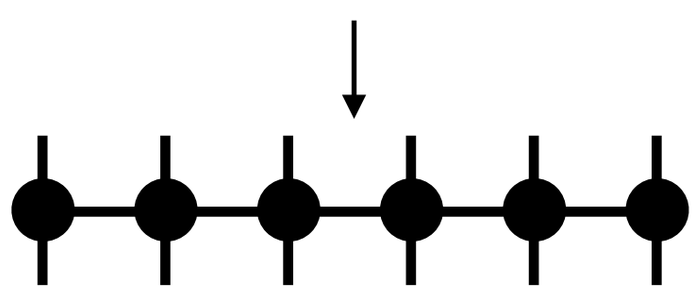
[1] Chan, Keselman, Nakatani, Li, White, J. Chem. Phys 145 (2016)
[2] Parker, Zaletel, Phys. Rev. B, 102, 035147 (2020)

MPS Algorithms

Wide range of operators can be made

$$H = \sum_j S^z_j S^z_{j+1}$$

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms += "Sz",j, "Sz",j+1
end

H = MPO(terms,sites)
```

$H =$ 

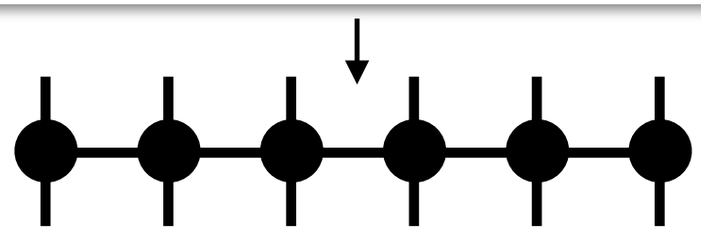
MPS Algorithms

Wide range of operators can be made

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

↓

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms +=       "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

$H =$ 

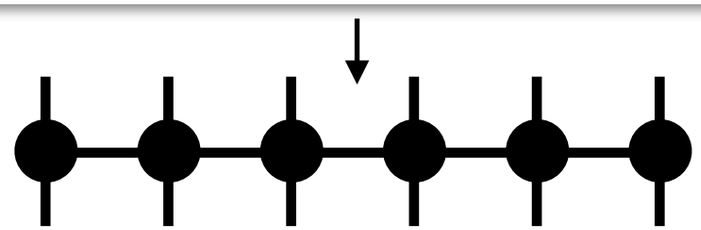
MPS Algorithms

Changing site type automatically gives correct operators

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

↓

```
sites = siteinds("S=1/2",N)

terms = OpSum()
for j=1:N-1
  terms +=      "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

$H =$ 

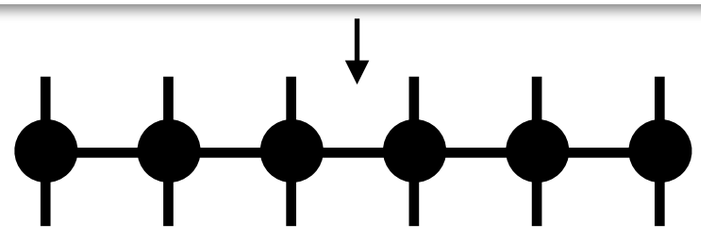
MPS Algorithms

Changing site type automatically gives correct operators

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

↓

```
sites = siteinds("S=1",N)

terms = OpSum()
for j=1:N-1
  terms +=      "Sz",j, "Sz",j+1
  terms += 1/2,"S+",j, "S-",j+1
  terms += 1/2,"S-",j, "S+",j+1
end

H = MPO(terms,sites)
```

$H =$ 

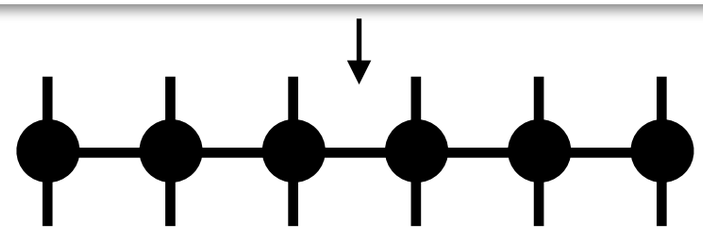
MPS Algorithms

Other particles (bosons, fermions) possible too

$$H = \sum_j c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j$$

↓

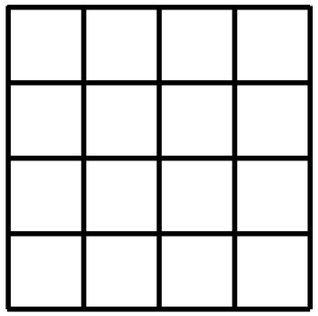
```
sites = siteinds("Fermion",N)

terms = OpSum()
for j=1:N-1
  terms += "Cdag",j, "C",j+1
  terms += "C",j+1, "Cdag",j+1
end

H = MPO(terms,sites)
```

$H =$ 

MPS Algorithms

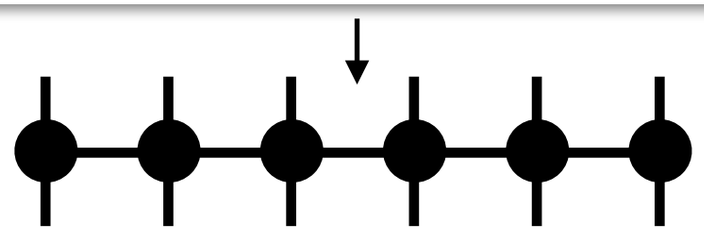
And quasi-two-dimensional systems

$$H = \sum_{\langle ij \rangle} S_i^z S_j^z + \frac{1}{2} S_i^+ S_j^- + \frac{1}{2} S_i^- S_j^+$$



```
lattice = square_lattice(Nx, Ny; yperiodic=false)

terms = OpSum()
for bond in lattice
  terms +=       "Sz", bond.s1, "Sz", bond.s2
  terms += 1/2, "S+", bond.s1, "S-", bond.s2
  terms += 1/2, "S-", bond.s1, "S+", bond.s2
end

H = MPO(terms, sites)
```

$H =$



MPS Algorithms

Next let's look at dmrg, apply, and tdvp

- ☑ OpSum system – making MPOs from operators
- ☐ dmrg, apply, tdvp – computing ground states and dynamics
- ☐ expect – computing expected values of operators
- ☐ correlation_matrix – compute correlation functions
- ☐ inner – overlap MPS and MPOs
- ☐ contract, sum – algebra of MPS and MPO

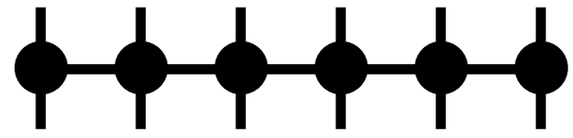
MPS Algorithms

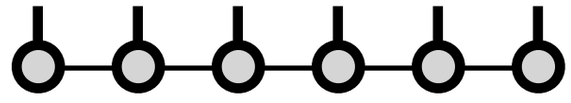
ITensorMPS offers "black box" DMRG algorithm

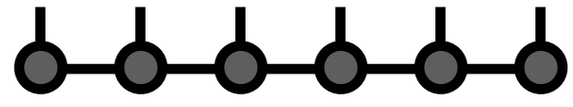
```
using ITensors, ITensorMPS

N = 100
sites = siteinds("S=1/2", N)

terms = OpSum()
for j in 1:(N - 1)
  terms +=       "Sz", j, "Sz", j + 1
  terms += 0.5, "S+", j, "S-", j + 1
  terms += 0.5, "S-", j, "S+", j + 1
end
H = MPO(terms, sites)

psi0 = random_mps(sites; linkdims=10)

nsweeps = 5
maxdim = [10, 20, 100, 100, 200]
cutoff = [1E-11]
energy, psi = dmrg(H, psi0; nsweeps, maxdim, cutoff)
```

$$H = \sum_j S^z_j S^z_{j+1} + \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1}$$

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$H =$ 

$\psi_0 =$ 

$\psi =$ 

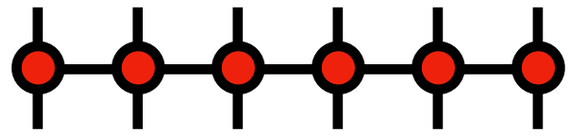
MPS Algorithms

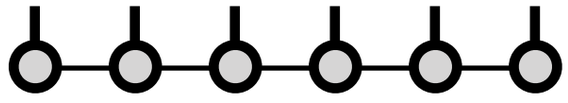
TDVP is similar, for time evolution by some H

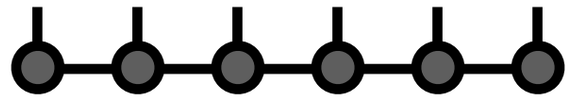
```
using ITensors, ITensorMPS

N = 100
sites = siteinds("S=1/2", N)

terms = OpSum()
for j in 1:(N - 1)
  terms +=       "Sz", j, "Sz", j + 1
  terms += 0.5, “S+", j, "S-", j + 1
  terms += 0.5, "S-", j, "S+", j + 1
end
H = MPO(terms, sites)

psi0 = random_mps(sites; linkdims=10)

t = 10
time_step = 0.1
maxdim = 200
cutoff = 1E-10
psi_t = tdvp(H, -im*t, psi0; maxdim, cutoff, time_step)
```

$$H = \sum_j S_j^z S_{j+1}^z + \frac{1}{2} S_j^+ S_{j+1}^- + \frac{1}{2} S_j^- S_{j+1}^+$$

(1.0,"Sz",1,"Sz",2), (1.0,"Sz",2,"Sz",3), ...

$e^{-iHt} =$ 

$\psi(0) =$ 

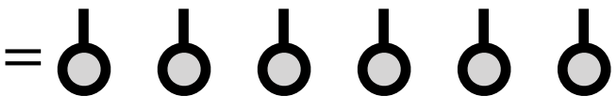
$\psi(t) =$ 

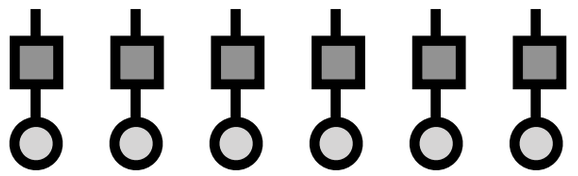
MPS Algorithms

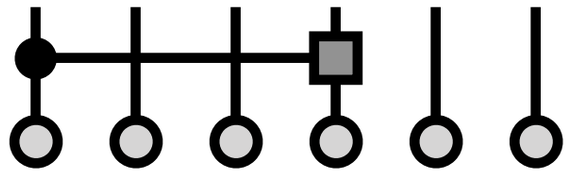
The **apply** function allows evolution by gates

```
using ITensors, ITensorMPS

N = 10
s = siteinds("Qubit", N)

# Start with the state |0000...⟩
ψ0 = MPS(s, "0")

# Make the operators
X = [op("X", s[j]) for j in 1:N]
CX = [op("CX", s[i], s[j]) for i in 1:N, j in 1:N]

# Change to the state |1010...⟩
gates = [X[j] for j in 1:2:N]
ψ = apply(gates, ψ0; cutoff=1e-15)
@assert inner(ψ, MPS(s, j -> isodd(j) ? "1" : "0")) ≈ 1

# Change to the state |10111011...⟩
append!(gates, [CX[j, j + 3] for j in 1:4:(N - 3)])
ψ = apply(gates, ψ0; cutoff=1e-15)
@assert inner(ψ, MPS(s, ["1", "0", "1", "1", "1", "0", "1", "1", "1", "0"])) ≈ 1
```

$\psi_0 =$



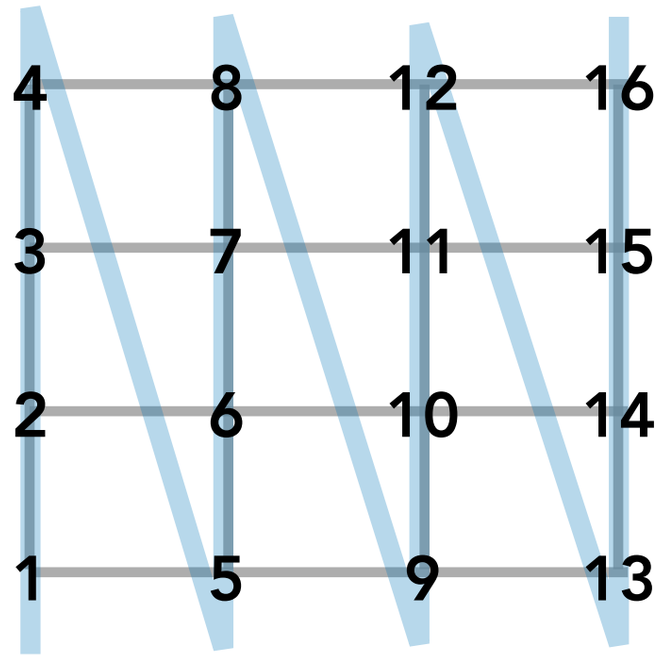




MPS Algorithms

To perform DMRG on a 2D system, just need a 1D assignment of the sites:

```
using ITensors, ITensorMPS

N = 100
sites = siteinds("S=1/2", N)

lattice = square_lattice(nx, ny; yperiodic = true)

terms = OpSum()
for bond in lattice
  i,j = bond.s1, bond.s2
  terms +=       "Sz", i, "Sz", j
  terms += 0.5, "S+", i, "S-", j
  terms += 0.5, "S-", i, "S+", j
end
H = MPO(terms, sites)

psi0 = random_mps(sites; linkdims=10)

nsweeps = 5
maxdim = [10, 20, 100, 100, 200]
cutoff = [1E-11]
energy, psi = dmrg(H, psi0; nsweeps, maxdim, cutoff)
```



lattice =

(1,5), (1,2), (2,6), (2,3), ...

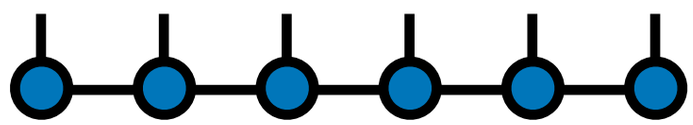
MPS Algorithms

Can use expect, correlation_matrix, and inner to analyze MPS

- ☑ OpSum system – making MPOs from operators
- ☑ dmrg, apply, tdvp – computing ground states and dynamics
- ☐ expect – computing expected values of operators
- ☐ correlation_matrix – compute correlation functions
- ☐ inner – overlap MPS and MPOs
- ☐ contract, sum – algebra of MPS and MPO

MPS Algorithms

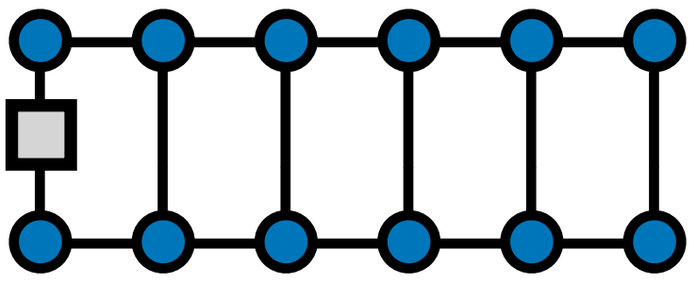
If we have an MPS and want expected values of local operators, can use expect function

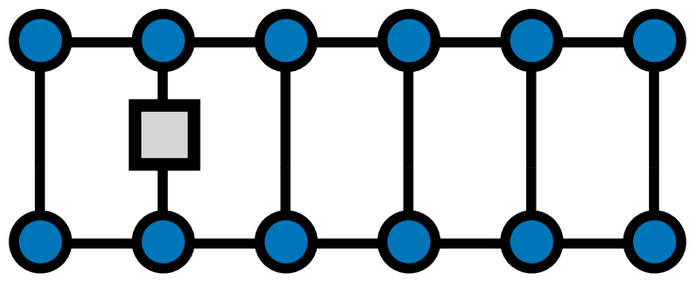
$\Psi =$  could be output of DMRG, TDVP

Then, for example, calling expect gives

```
sz = expect(psi,"Sz")
```

where sz is an array such that

sz[1] = 

sz[2] = 

etc.

MPS Algorithms

Of course, can use various operators
for spins, particles, or qubit sites

Some examples:

```
sz = expect(psi,"Sz")
```
magnetization of spins

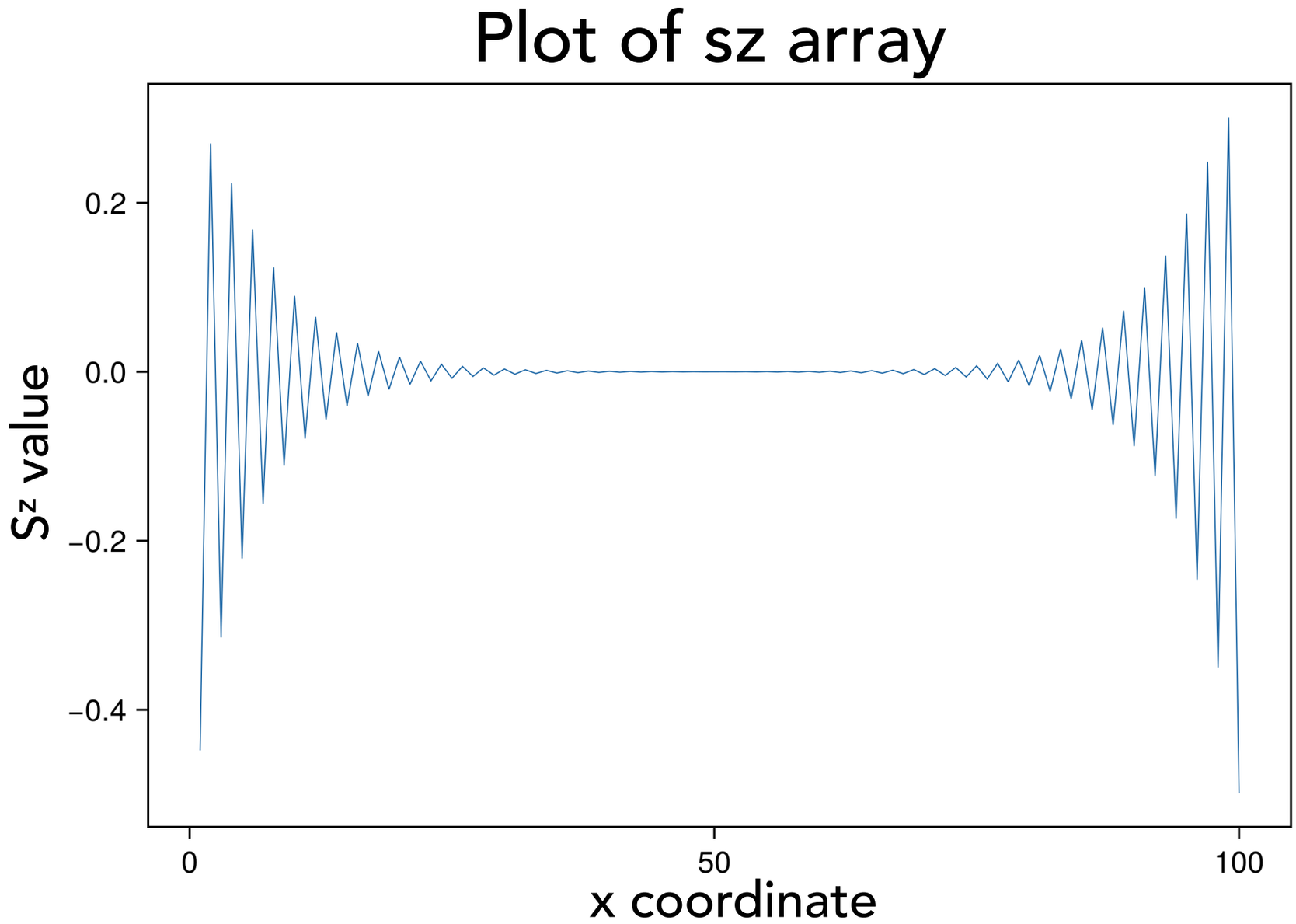
```
density = expect(psi,"N")
```
density of particles

```
Xvals = expect(psi,"X")
```
<X> over a set of qubits

MPS Algorithms

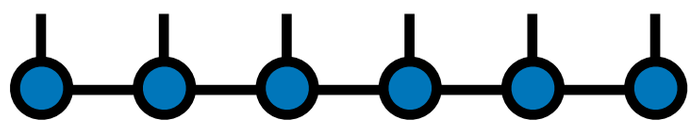
Example output:

```
energy, psi = dmrg(H, psi0; nsweeps, maxdim, cutoff)
sz = expect(psi,"Sz")
```



MPS Algorithms

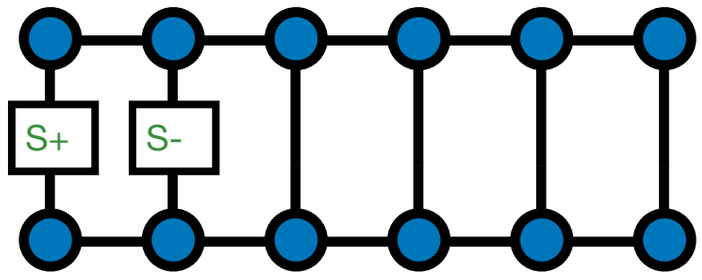
The correlation_matrix function also computes expected values but of "two point" functions i.e. correlators

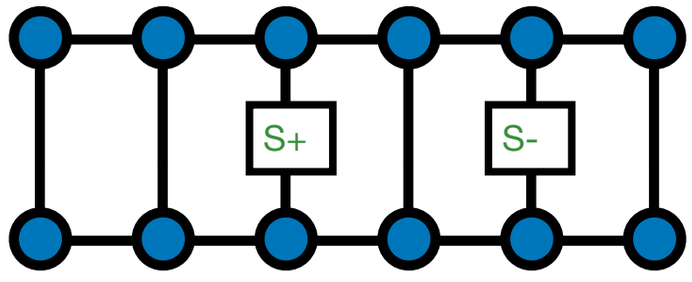
$\Psi =$  could be output of DMRG, TDVP

Then, calling correlation_matrix gives

```
spsm = correlation_matrix(psi,"S+","S-")
```

where spsm is a Matrix such that

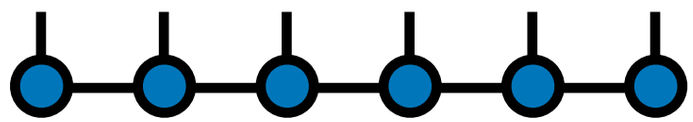
spsm[1,2] = 

spsm[3,5] = 

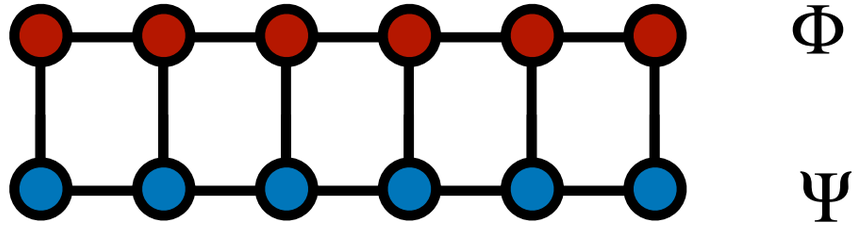
etc.

MPS Algorithms

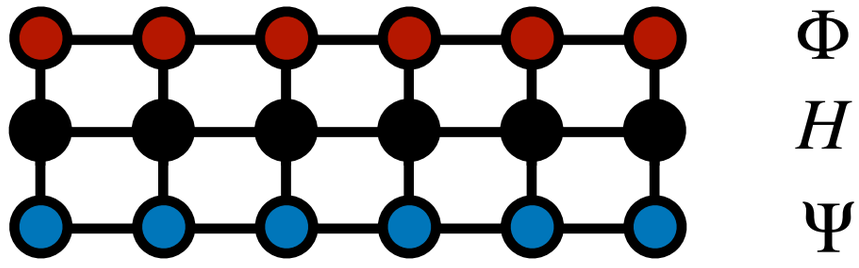
The inner function lets us analyze MPS through overlaps with other MPS and MPOs

$\Psi =$  could be output of DMRG, TDVP

Then, calling inner with another MPS gives

inner(psi,phi) = 

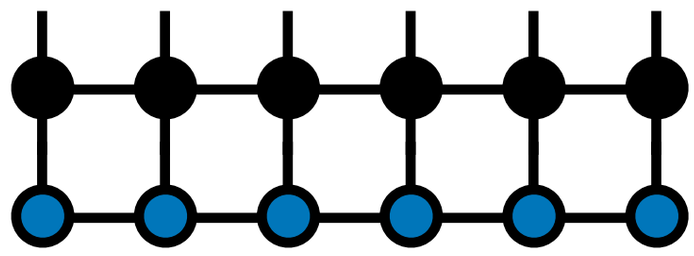
Or including an MPO like

inner(phi',H,psi) = 

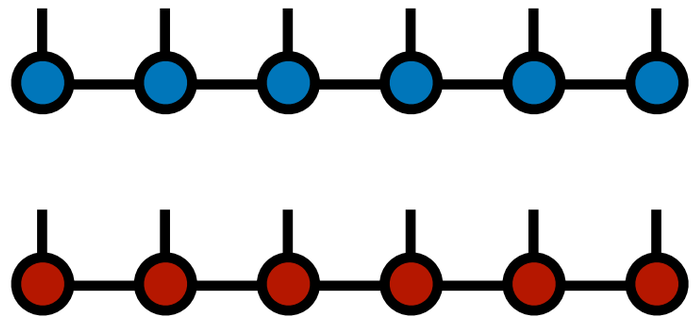
MPS Algorithms

Finally MPS and MPO tensor networks can be contracted and added with contract and add

contract(W, psi; cutoff) =



add(psi, phi; cutoff) =



Summary

ITensorMPS package offers convenient tools for constructing MPO operators and running algorithms like DMRG

Up Next

After a break, we will get hands-on experience using ITensorMPS for real calculations