

# Analyzing Quantum Many-Body Systems with ITensor and PastaQ

Matthew Fishman

Center for Computational Quantum Physics (CCQ)

Flatiron Institute, NY

[mtfishman.github.io](https://mtfishman.github.io)

[github.com/mtfishman/ITensorTutorials.jl](https://github.com/mtfishman/ITensorTutorials.jl)

March 1, 2022

# *Who am I?*

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.



[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

## *Who am I?*

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.
- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).



[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

# Who am I?

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.
- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).
- ▶ Thesis on developing tensor network algorithms.



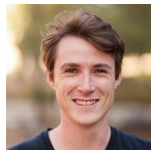
[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

# Who am I?

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.
- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).
- ▶ Thesis on developing tensor network algorithms.
- ▶ Associate data scientist at CCQ since 2018.



[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

# Who am I?

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.
- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).
- ▶ Thesis on developing tensor network algorithms.
- ▶ Associate data scientist at CCQ since 2018.
- ▶ Develop **ITensor** with **Miles Stoudenmire** (CCQ).



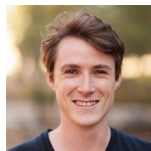
[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

# Who am I?

- ▶ PhD from Caltech with **John Preskill** and **Steve White** (UCI) in 2018.
- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).
- ▶ Thesis on developing tensor network algorithms.
- ▶ Associate data scientist at CCQ since 2018.
- ▶ Develop **ITensor** with **Miles Stoudenmire** (CCQ).
- ▶ Develop **PastaQ** with **Giacomo Torlai** (AWS).



[mtfishman.github.io](https://mtfishman.github.io)



universität  
wien

# *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.



## *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.
- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.

## *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.
- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.
- ▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.

## *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.
- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.
- ▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.
- ▶ Reach out to us for software needs (new features, bugs), algorithm development, high performance computing, etc.

## *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.
- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.
- ▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.
- ▶ Reach out to us for software needs (new features, bugs), algorithm development, high performance computing, etc.
- ▶ CCQ has expertise in QMC, quantum embedding (DMFT), tensor networks, quantum dynamics, machine learning, quantum computing, quantum chemistry, etc.

## *What is the Center for Computational Quantum Physics (CCQ)?*

- ▶ Part of the Flatiron Institute (NYC), funded by the Simons Foundation.
- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.
- ▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.
- ▶ Reach out to us for software needs (new features, bugs), algorithm development, high performance computing, etc.
- ▶ CCQ has expertise in QMC, quantum embedding (DMFT), tensor networks, quantum dynamics, machine learning, quantum computing, quantum chemistry, etc.
- ▶ We are hiring postdocs, full-time scientists, part-time and full-time software developers, interns, etc.

# *What is the Center for Computation Quantum Physics (CCQ)?*



SIMONS FOUNDATION

# *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).





## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.
- ▶ Used in over **400** research papers.



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.
- ▶ Used in over **400** research papers.
- ▶ Website: [itensor.org](https://itensor.org).



## *What is ITensor?*

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ **Miles Stoudenmire** (CCQ) started working on ITensor around 2011.
- ▶ I started working on ITensor in 2018, co-develop with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.
- ▶ Used in over **400** research papers.
- ▶ Website: [itensor.org](https://itensor.org).
- ▶ Paper: [arxiv.org/abs/2007.14822](https://arxiv.org/abs/2007.14822)



## *Porting to Julia*

- ▶ Porting the full ITensor library took about  $1\frac{1}{2}$  years.



## *Porting to Julia*

- ▶ Porting the full ITensor library took about  $1\frac{1}{2}$  years.
- ▶ Much less code, easier development.

## *Porting to Julia*

- ▶ Porting the full ITensor library took about  $1\frac{1}{2}$  years.
- ▶ Much less code, easier development.
- ▶ Now that it is ported, we have many more features: GPU, autodiff, infinite MPS, visualization, etc.

## Porting to Julia

- ▶ Porting the full ITensor library took about  $1\frac{1}{2}$  years.
- ▶ Much less code, easier development.
- ▶ Now that it is ported, we have many more features: GPU, autodiff, infinite MPS, visualization, etc.
- ▶ Great numerical libraries, code ended up faster.



# *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.

## *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.

## *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.
- ▶ Fast like C/C++, high level features of Python, all in one language.

## *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.
- ▶ Fast like C/C++, high level features of Python, all in one language.
- ▶ Just-in-time compiled, garbage collected, package manager, plotting, great numerical libraries, etc.

## *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.
- ▶ Fast like C/C++, high level features of Python, all in one language.
- ▶ Just-in-time compiled, garbage collected, package manager, plotting, great numerical libraries, etc.
- ▶ Julia is great, you should use it.



## *What is Julia?*



- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.
- ▶ Fast like C/C++, high level features of Python, all in one language.
- ▶ Just-in-time compiled, garbage collected, package manager, plotting, great numerical libraries, etc.
- ▶ Julia is great, you should use it.
- ▶ I could say more, ask me about it.

## *What is Julia?*

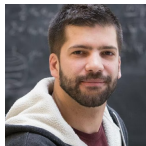


- ▶ Started in 2009 at MIT, v0.1 released in 2013, v1 released in 2018.
- ▶ Language is very stable, ecosystem is growing very quickly.
- ▶ Fast like C/C++, high level features of Python, all in one language.
- ▶ Just-in-time compiled, garbage collected, package manager, plotting, great numerical libraries, etc.
- ▶ Julia is great, you should use it.
- ▶ I could say more, ask me about it.
- ▶ Find out more at: [julialang.org](https://julialang.org).

*What is PastaQ?*

# Pasta

- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.



*What is PastaQ?*

# Pasta

- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.



*What is PastaQ?*

# Pasta

- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.



*What is PastaQ?*

# Pasta

- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.
  - ▶ Extensive and customizable gate definitions.

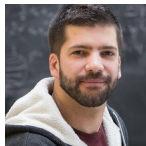


*What is PastaQ?*

# Pasta



- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.
  - ▶ Extensive and customizable gate definitions.
  - ▶ Built-in and easily extendable circuit definitions.



*What is PastaQ?*

# Pasta

- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.
  - ▶ Extensive and customizable gate definitions.
  - ▶ Built-in and easily extendable circuit definitions.
  - ▶ Noisy circuit evolution with customizable noise models.





*What is PastaQ?*



- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.
  - ▶ Extensive and customizable gate definitions.
  - ▶ Built-in and easily extendable circuit definitions.
  - ▶ Noisy circuit evolution with customizable noise models.
  - ▶ Approximate circuit evolution and optimization with MPS/MPO, etc.



*What is PastaQ?*

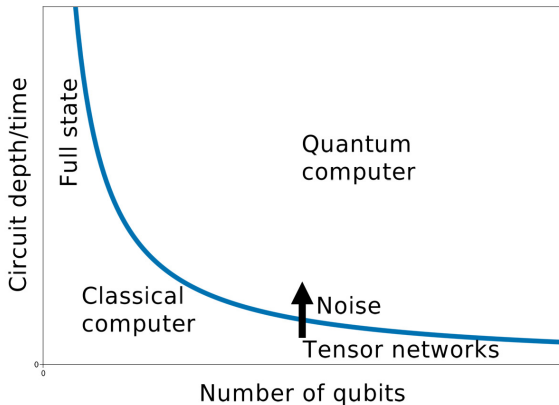


- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
  - ▶ Tensor network-based quantum state and process tomography.
  - ▶ Extensive and customizable gate definitions.
  - ▶ Built-in and easily extendable circuit definitions.
  - ▶ Noisy circuit evolution with customizable noise models.
  - ▶ Approximate circuit evolution and optimization with MPS/MPO, etc.
- ▶ Find out more: [github.com/GTorlai/PastaQ.jl](https://github.com/GTorlai/PastaQ.jl)



## *When should I use tensor networks?*

- In my opinion, tensor networks are the best general purpose tool we have right now for simulating and analyzing quantum computers.



## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).

## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Best when local gate structure or interactions match the graph structure of the tensor network.

## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Best when local gate structure or interactions match the graph structure of the tensor network.
- ▶ If you need very long time evolution/many layers but few qubits, use full state simulation.

## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Best when local gate structure or interactions match the graph structure of the tensor network.
- ▶ If you need very long time evolution/many layers but few qubits, use full state simulation.
  - ▶ ITensor and PastaQ handle this seamlessly.

## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Best when local gate structure or interactions match the graph structure of the tensor network.
- ▶ If you need very long time evolution/many layers but few qubits, use full state simulation.
  - ▶ ITensor and PastaQ handle this seamlessly.
  - ▶ This is not the focus of ITensor and PastaQ at the moment, specialized libraries like [Yao.jl](#) may be faster.



## *When should I use tensor networks?*

- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Best when local gate structure or interactions match the graph structure of the tensor network.
- ▶ If you need very long time evolution/many layers but few qubits, use full state simulation.
  - ▶ ITensor and PastaQ handle this seamlessly.
  - ▶ This is not the focus of ITensor and PastaQ at the moment, specialized libraries like [Yao.jl](#) may be faster.
- ▶ Tensor networks are a common, general language for reasoning about quantum many-body systems (for example, quantum circuits).

# *What are tensor networks?*

Vector  $V_i$   
Order-1 tensor



Matrix  $M_{ij}$   
Order-2 tensor

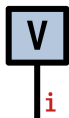


Tensor  $T_{ijk}$   
Order-3 tensor



# What are tensor networks?

Vector  $V_i$   
Order-1 tensor



$$\langle V|V \rangle = V_i V_i$$



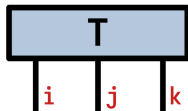
Matrix  $M_{ij}$   
Order-2 tensor



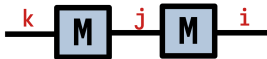
$$M|V \rangle = M_{ji} V_i$$



Tensor  $T_{ijk}$   
Order-3 tensor



$$MM = M_{kj} M_{ji}$$



# What are tensor networks?

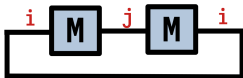
Vector  $V_i$   
Order-1 tensor



$$\langle V|V \rangle = V_i V_i$$



$$\text{tr}(MM) = M_{ij} M_{ji}$$



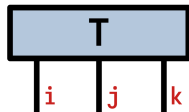
Matrix  $M_{ij}$   
Order-2 tensor



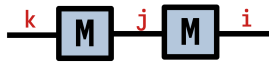
$$M|V \rangle = M_{ji} V_i$$



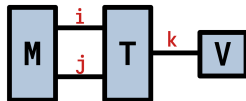
Tensor  $T_{ijk}$   
Order-3 tensor



$$MM = M_{kj} M_{ji}$$



$$M_{ij} T_{ijk} V_k$$



## *How do I install ITensor/PastaQ?*

1. Download Julia: [julialang.org/downloads](https://julialang.org/downloads)

## *How do I install ITensor/PastaQ?*

1. Download Julia: [julialang.org/downloads](https://julialang.org/downloads)
2. Launch Julia:

```
1 $ julia
```

## *How do I install ITensor/PastaQ?*

1. Download Julia: [julialang.org/downloads](https://julialang.org/downloads)
2. Launch Julia:

```
1 $ julia
```

3. Type the following commands:

```
1 julia> using Pkg
2
3 julia> Pkg.add("ITensors")
4 [...]
5
6 julia> Pkg.add("PastaQ")
7 [...]
```

## *How do I install ITensor/PastaQ?*

Now you can use ITensors and PastaQ:

```
1 julia> using ITensors
2
3 julia> i = Index(2);
4
5 julia> A = ITensor(i);
```



## *How do I install ITensor/PastaQ?*

Now you can use ITensors and PastaQ:

```
1 julia> using ITensors
2
3 julia> i = Index(2);
4
5 julia> A = ITensor(i);
```

```
1 julia> using PastaQ
2
3 julia> gates = [("X", 1), ("CX", (1, 3))];
4
5 julia>  $\psi$  = runcircuit(gates);
```

## *Tutorial: One-site state basics*

```
1  using ITensors
2
3  i = Index(2)
4
5
```

Load ITensor

2-dimensional labeled  
Hilbert space  
(dim=2|id=510)

## *Tutorial: One-site state basics*

```
1 using ITensors
2
3 i = Index(2)
4
5
```

**i**

## Tutorial: One-site state basics

```
1 using ITensors
2
3 i = Index(2)
4
5
```

```
1 Zp = ITensor(i)
2 Zp[i=>1] = 1
3
4 Zp = ITensor([1, 0], i)
```

i

$$|Z+\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Construct from a Vector

## Tutorial: One-site state basics

```
1 using ITensors
```

```
2
```

```
3 i = Index(2)
```

```
4
```

```
5
```

```
1 Zp = ITensor(i)
```

```
2 Zp[i=>1] = 1
```

```
3
```

```
4 Zp = ITensor([1, 0], i)
```

i

$$\boxed{1} \overset{i}{-} \boxed{Z+} = 1 \quad \boxed{2} \overset{i}{-} \boxed{Z+} = 0$$

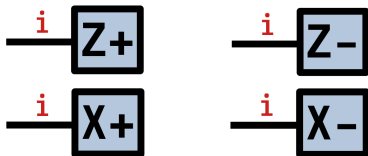
## *Tutorial: One-site state basics*

```
1 Zp = ITensor([1, 0], i)
2 Zm = ITensor([0, 1], i)
3 Xp = ITensor([1, 1]/√2, i)
4 Xm = ITensor([1, -1]/√2, i)
```

$$\begin{aligned} |Z+\rangle &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, & |Z-\rangle &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ |X+\rangle &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} / \sqrt{2}, & |X-\rangle &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} / \sqrt{2} \end{aligned}$$

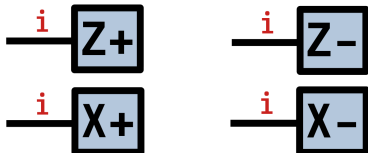
## Tutorial: One-site state basics

```
1 Zp = ITensor([1, 0], i)
2 Zm = ITensor([0, 1], i)
3 Xp = ITensor([1, 1]/sqrt(2), i)
4 Xm = ITensor([1, -1]/sqrt(2), i)
```



## Tutorial: One-site state basics

```
1 Zp = ITensor([1, 0], i)
2 Zm = ITensor([0, 1], i)
3 Xp = ITensor([1, 1]/√2, i)
4 Xm = ITensor([1, -1]/√2, i)
```



```
1 Xp = (Zp + Zm)/√2
2
3
4
5 dag(Zp) * Xp
6 inner(Zp, Xp)
```

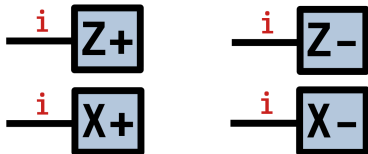
$$|X+\rangle = (|Z+\rangle + |Z-\rangle)/\sqrt{2}$$

$$\langle Z+ | X+\rangle \approx 1/\sqrt{2}$$



## Tutorial: One-site state basics

```
1 Zp = ITensor([1, 0], i)
2 Zm = ITensor([0, 1], i)
3 Xp = ITensor([1, 1]/√2, i)
4 Xm = ITensor([1, -1]/√2, i)
```



```
1 Xp = (Zp + Zm)/√2
```

$$\text{Diagram of } X_+ = \frac{\text{Diagram of } Z_+ + \text{Diagram of } Z_-}{\sqrt{2}}$$

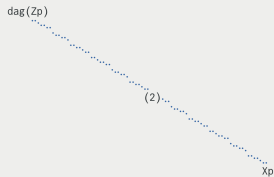
```
5 dag(Zp) * Xp
6 inner(Zp, Xp)
```

$$\text{Diagram of } Z_+ \text{ connected to Diagram of } X_+ = 1/\sqrt{2}$$

## Tutorial: One-site state basics

```
1 using ITensorUnicodePlots
2
3 @visualize dag(Zp) * Xp
```

```
julia> @visualize dag(Zp) * Xp
```



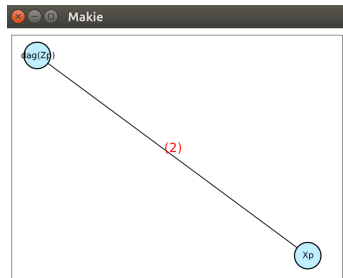
# Tutorial: One-site state basics

```
1 using ITensorUnicodePlots
2
3 @visualize dag(Zp) * Xp
```

```
julia> @visualize dag(Zp) * Xp
```



```
1 using ITensorGLMakie
2
3 @visualize dag(Zp) * Xp
```



## *Tutorial: One-site states*

```
1 i = Index(2, "S=1/2")
```

(dim=2|id=25|"S=1/2")

"S=1/2" defines an operator  
basis

Additionally:  
"Qubit", "Qudit",  
"Electron", ...

## Tutorial: One-site states

```
1 i = Index(2, "S=1/2")
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

(dim=2|id=25|"S=1/2")

"S=1/2" defines an operator basis

Additionally:

"Qubit", "Qudit",

"Electron", ...

```
1 Zp = state("Z+", i)
```

```
2 Zm = state("Z-", i)
```

```
3 Xp = state("X+", i)
```

```
4 Xm = state("X-", i)
```

```
1 ITensor([1 0], i)
```

```
2 ITensor([0 1], i)
```

```
3 ITensor([1 1]/√2, i)
```

```
4 ITensor([1 -1]/√2, i)
```

## *Tutorial: Custom one-site states*

```
1 import ITensors: state
2
3
4 function state(
5     ::StateName "iX-",
6     ::SiteType "S=1/2"
7 )
8     return [im -im]/√2
9 end
```

Overload ITensors.jl  
behavior

Define a state with the  
name “iX-”

## Tutorial: Custom one-site states

```
1 import ITensors: state
2
3
4 function state(
5     ::StateName "iX-",
6     ::SiteType "S=1/2"
7 )
8     return [im -im]/√2
9 end
```

Overload ITensors.jl  
behavior

Define a state with the  
name “iX-”

```
1 iXm = state("iX-", i)
2
3
4 inner(Zm, iXm)
```

$$|iX-\rangle = i|X-\rangle$$

$$\langle Z - |iX-\rangle = -i/\sqrt{2}$$

## Tutorial: Custom one-site states

```
1 import ITensors: state
2
3
4 function state(
5     ::StateName "iX-",
6     ::SiteType "S=1/2"
7 )
8     return [im -im]/√2
9 end
```

Overload ITensors.jl  
behavior

Define a state with the  
name “iX-”

```
1 iXm = state("iX-", i)
2
3
4 inner(Zm, iXm)
```

$$\text{---} \boxed{\text{iX+}} = \left( \text{---} \boxed{\text{X+}} \right) * i$$



## *Tutorial: Priming*

```
1 i = Index(2)
2 j = Index(2)
3
4 i ≠ j
```

(dim=2|id=837)  
(dim=2|id=899)

## *Tutorial: Priming*

```
1 i = Index(2)
2 j = Index(2)
3
4 i ≠ j
```

i ≠ j

## Tutorial: Priming

```
1 i = Index(2)
2 j = Index(2)
3
4 i ≠ j
```

i ≠ j

```
1 i = Index(2)
2
3 prime(i) == i'
4
5 i ≠ i'
6 noprime(i') == i
```

(dim=2|id=837)

(dim=2|id=837)'

## Tutorial: Priming

```
1 i = Index(2)
2 j = Index(2)
3
4 i ≠ j
```

```
1 i = Index(2)
2
3 prime(i) == i'
4
5 i ≠ i'
6 noprime(i') == i
```

$$\underline{i} \neq \underline{j}$$

$$\text{prime}\left(\frac{i}{\underline{\quad}}\right) = \underline{i'}$$
$$\underline{i} \neq \underline{i'}$$

## *Tutorial: One-site operators*

```
1 Z = ITensor(i', i)
2 Z[i'=>1, i=>1] = 1
3 Z[i'=>2, i=>2] = -1
```

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

## *Tutorial: One-site operators*

```
1  Z = ITensor(i', i)
2  Z[i'=>1, i=>1] = 1
3  Z[i'=>2, i=>2] = -1
```



## Tutorial: One-site operators

```
1 Z = ITensor(i', i)
2 Z[i'=>1, i=>1] = 1
3 Z[i'=>2, i=>2] = -1
```



```
1 z = [
2     1 0
3     0 -1
4 ]
5 Z = ITensor(z, i', dag(i))
6
7
8 Z = op("Z", i)
```

Matrix representation Z

Convert to ITensor

Use predefined definition

## *Tutorial: One-site operators*

```
1  Z = op("Z", i)
```

```
2  X = op("X", i)
```

```
3
```

```
4  Zp = state("Z+", i)
```

```
5  Zm = state("Z-", i)
```

Z

X

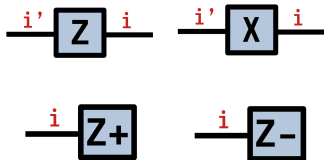
$|Z+\rangle$

$|Z-\rangle$



## Tutorial: One-site operators

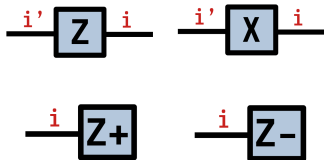
```
1 Z = op("Z", i)
2 X = op("X", i)
3
4 Zp = state("Z+", i)
5 Zm = state("Z-", i)
```



## Tutorial: One-site operators

```
1 Z = op("Z", i)
2 X = op("X", i)
3
4 Zp = state("Z+", i)
5 Zm = state("Z-", i)
```

```
1 X * Zp == Zm'
2 noprime(X * Zp) == Zm
```

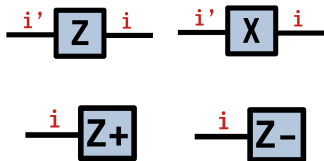


$$X|Z+\rangle = |Z-\rangle$$

## Tutorial: One-site operators

```
1 Z = op("Z", i)
2 X = op("X", i)
3
4 Zp = state("Z+", i)
5 Zm = state("Z-", i)
```

```
1 X * Zp == Zm'
2 noprime(X * Zp) == Zm
```



## *Tutorial: One-site operators*

```
1 dag(Zm)' * X * Zp  
2 inner(Zm', X, Zp)
```

$$\langle Z - |X|Z+ \rangle \approx 1$$

## *Tutorial: One-site operators*

- 1 `dag(Zm)' * X * Zp`
- 2 `inner(Zm', X, Zp)`

$$\boxed{Z-} \overset{i'}{\text{---}} \boxed{X} \overset{i}{\text{---}} \boxed{Z+} = 1$$

## Tutorial: One-site operators

```
1 dag(Zm)' * X * Zp
2 inner(Zm', X, Zp)
```

$$\boxed{Z-} \overset{i'}{\text{---}} \boxed{X} \overset{i}{\text{---}} \boxed{Z+} = 1$$

```
1 apply(X, Zp) ==
2   noprime(X * Zp)
3
4
5
6 inner(Zm, apply(X, Zp))
```

$$X|Z+\rangle$$

$$\langle Z- | X | Z+ \rangle \approx 1$$

## Tutorial: One-site operators

```
1 dag(Zm)' * X * Zp
2 inner(Zm', X, Zp)
```

$$\boxed{Z-} \overset{i'}{\text{---}} \boxed{X} \overset{i}{\text{---}} \boxed{Z+} = 1$$

```
1 apply(X, Zp) ==
2   noprime(X * Zp)
3
4
5
6 inner(Zm, apply(X, Zp))
```

$$\begin{aligned} & \text{apply}\left(\overset{i'}{\text{---}} \boxed{X} \overset{i}{\text{---}}, \overset{i}{\text{---}} \boxed{Z+}\right) \\ &= \text{noprime}\left(\overset{i'}{\text{---}} \boxed{X} \overset{i}{\text{---}} \boxed{Z+}\right) \\ &= \overset{i}{\text{---}} \boxed{Z-} \end{aligned}$$

## *Tutorial: Custom one-site operators*

```
1  import ITensors: op
2
3  function op(
4      ::OpName"iX",
5      ::SiteType"S=1/2"
6  )
7      return [
8          0 im
9          im 0
10     ]
11  end
```

Overload ITensors.jl  
behavior



## *Tutorial: Custom one-site operators*

```
1  import ITensors: op
2
3  function op(
4      ::OpName"iX",
5      ::SiteType"S=1/2"
6  )
7      return [
8          0 im
9          im 0
10     ]
11  end
```

$$\text{---} \overset{i'}{\boxed{iX}} \text{---}^i = \left( \text{---} \overset{i'}{\boxed{X}} \text{---}^i \right) * i$$

## Tutorial: Custom one-site operators

```
1 import ITensors: op
2
3 function op(
4     ::OpName"iX",
5     ::SiteType"S=1/2"
6 )
7     return [
8         0 im
9         im 0
10    ]
11 end
```

```
1 op("iX", i)
```

$$\text{---} \overset{i'}{\boxed{iX}} \text{---} = \left( \text{---} \overset{i'}{\boxed{X}} \text{---} \right) * i$$

## Tutorial: Custom one-site operators

```
1 import ITensors: op
2
3 function op(
4     ::OpName "iX",
5     ::SiteType "S=1/2"
6 )
7     return [
8         0 im
9         im 0
10    ]
11 end
```

```
1 op("iX", i)
```

$$\text{---} \overset{i'}{\boxed{iX}} \text{---} = \left( \text{---} \overset{i'}{\boxed{X}} \text{---} \right) * i$$

```
1 X * im
```

## Tutorial: Two-site states

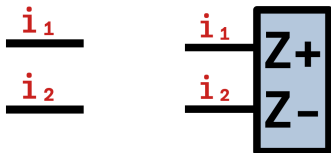
```
1 i1 = Index(2, "S=1/2")
2 i2 = Index(2, "S=1/2")
3
4 i1 ≠ i2
5
6 ZpZm = ITensor(i1, i2)
7 ZpZm[i1=>1, i2=>2] = 1
```

(dim=2|id=505|“S=1/2”)  
(dim=2|id=576|“S=1/2”)

$$|Z+\rangle_1 |Z-\rangle_2 = |Z+Z-\rangle$$

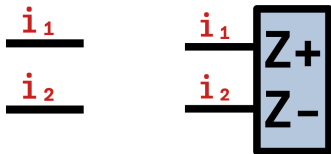
## Tutorial: Two-site states

```
1 i1 = Index(2, "S=1/2")
2 i2 = Index(2, "S=1/2")
3
4 i1 ≠ i2
5
6 ZpZm = ITensor(i1, i2)
7 ZpZm[i1=>1, i2=>2] = 1
```



## Tutorial: Two-site states

```
1 i1 = Index(2, "S=1/2")
2 i2 = Index(2, "S=1/2")
3
4 i1 ≠ i2
5
6 ZpZm = ITensor(i1, i2)
7 ZpZm[i1=>1, i2=>2] = 1
```

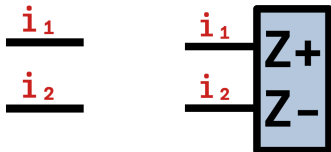


```
1 Zp1 = state("Z+", i1)
2 Zp2 = state("Z+", i2)
3
4 Zm1 = state("Z-", i1)
5 Zm2 = state("Z-", i2)
6
7 ZpZm = Zp1 * Zm2
```

 $|Z+\rangle_1$  $|Z+\rangle_2$  $|Z-\rangle_1$  $|Z-\rangle_2$  $|Z+Z-\rangle = |Z+\rangle_1 |Z-\rangle_2$

## Tutorial: Two-site states

```
1 i1 = Index(2, "S=1/2")
2 i2 = Index(2, "S=1/2")
3
4 i1 != i2
5
6 ZpZm = ITensor(i1, i2)
7 ZpZm[i1=>1, i2=>2] = 1
```



```
1 Zp1 = state("Z+", i1)
2 Zp2 = state("Z+", i2)
3
4 Zm1 = state("Z-", i1)
5 Zm2 = state("Z-", i2)
6
7 ZpZm = Zp1 * Zm2
```



## *Tutorial: Two-site states*

```
1   $\psi = \text{ITensor}(i1, i2)$ 
2   $\psi[i1=>1, i2=>2] = 1/\sqrt{2}$ 
3   $\psi[i1=>2, i2=>1] = 1/\sqrt{2}$ 
4
5   $\psi = (Z_{p1} * Z_{m2} +$ 
6       $Z_{m1} * Z_{p2})/\sqrt{2}$ 
```

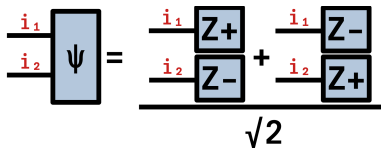
$$(|Z+\rangle|Z-\rangle + |Z-\rangle|Z+\rangle)/\sqrt{2}$$

From single-site states



## Tutorial: Two-site states

```
1   $\psi = \text{ITensor}(i1, i2)$ 
2   $\psi[i1=>1, i2=>2] = 1/\sqrt{2}$ 
3   $\psi[i1=>2, i2=>1] = 1/\sqrt{2}$ 
4
5   $\psi = (\text{Zp1} * \text{Zm2} +$ 
6         $\text{Zm1} * \text{Zp2})/\sqrt{2}$ 
```

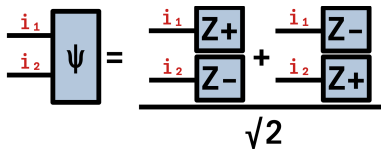


## Tutorial: Two-site states

```

1   $\psi = \text{ITensor}(i1, i2)$ 
2   $\psi[i1=>1, i2=>2] = 1/\sqrt{2}$ 
3   $\psi[i1=>2, i2=>1] = 1/\sqrt{2}$ 
4
5   $\psi = (\text{Zp1} * \text{Zm2} +$ 
6       $\text{Zm1} * \text{Zp2})/\sqrt{2}$ 

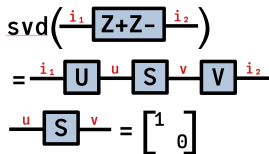
```



```

1   $\text{inner}(\text{ZpZm}, \psi) == 1/\sqrt{2}$ 
2
3
4   $\text{U}, \text{S}, \text{V} = \text{svd}(\text{ZmZp}, i1)$ 
5   $\text{diag}(\text{S}) == [1, 0]$ 
6
7   $\text{U}, \text{S}, \text{V} = \text{svd}(\psi, i1)$ 
8   $\text{diag}(\text{S}) == [1/\sqrt{2}, 1/\sqrt{2}]$ 

```



## Tutorial: Two-site states

```

1   $\psi = \text{ITensor}(i1, i2)$ 
2   $\psi[i1=>1, i2=>2] = 1/\sqrt{2}$ 
3   $\psi[i1=>2, i2=>1] = 1/\sqrt{2}$ 
4
5   $\psi = (\text{Zp1} * \text{Zm2} +$ 
6       $\text{Zm1} * \text{Zp2})/\sqrt{2}$ 

```

```

1   $\text{inner}(\text{ZpZm}, \psi) == 1/\sqrt{2}$ 
2
3
4   $\text{U}, \text{S}, \text{V} = \text{svd}(\text{ZmZp}, i1)$ 
5   $\text{diag}(\text{S}) == [1, 0]$ 
6
7   $\text{U}, \text{S}, \text{V} = \text{svd}(\psi, i1)$ 
8   $\text{diag}(\text{S}) == [1/\sqrt{2}, 1/\sqrt{2}]$ 

```

$$\psi = \frac{\begin{matrix} i_1 \\ i_2 \end{matrix} \begin{matrix} \boxed{Z+} \\ \boxed{Z-} \end{matrix}}{\sqrt{2}} + \frac{\begin{matrix} i_1 \\ i_2 \end{matrix} \begin{matrix} \boxed{Z-} \\ \boxed{Z+} \end{matrix}}{\sqrt{2}}$$

$$\text{svd}\left(\begin{matrix} i_1 \\ i_2 \end{matrix} \boxed{Z+Z-} \right) = \begin{matrix} i_1 \\ u \end{matrix} \boxed{U} \begin{matrix} u \\ v \end{matrix} \boxed{S} \begin{matrix} v \\ i_2 \end{matrix} \boxed{V}$$

$$\begin{matrix} u \\ v \end{matrix} \boxed{S} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\text{svd}\left(\begin{matrix} i_1 \\ i_2 \end{matrix} \boxed{\psi} \right) = \begin{matrix} i_1 \\ u \end{matrix} \boxed{U} \begin{matrix} u \\ v \end{matrix} \boxed{S} \begin{matrix} v \\ i_2 \end{matrix} \boxed{V}$$

$$\begin{matrix} u \\ v \end{matrix} \boxed{S} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 0 \end{bmatrix}$$

## Tutorial: Two-site operators

```
1 H = ITensor(i1', i2', i1, i2)
2 H[i1'=>2, i2'=>1,
3   i1=>2, i2=>1] = -1
4 # ...
```

Make a Hamiltonian:

Transverse field Ising ( $n = 2$ )

$$H = -\sum_j^{n-1} Z_j Z_{j+1} + h \sum_j^n X_j$$

## Tutorial: Two-site operators

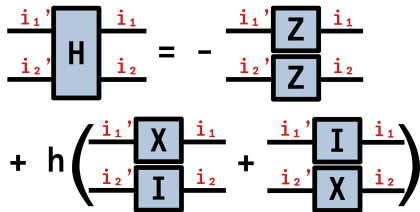
```
1 H = ITensor(i1', i2', i1, i2)
2 H[i1'=>2, i2'=>1,
3   i1=>2, i2=>1] = -1
4 # ...
```

```
1 Id1 = op("Id", i1)
2 Z1 = op("Z", i1)
3 X1 = op("X", i1)
4 # ...
5
6 ZZ = Z1 * Z2
7 XI = X1 * Id2
8 IX = Id1 * X2
9
10 h = 0.5
11 H = -ZZ + h * (XI + IX)
```

Make a Hamiltonian:

Transverse field Ising ( $n = 2$ )

$$H = -\sum_j^{n-1} Z_j Z_{j+1} + h \sum_j^n X_j$$



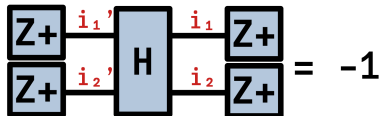
## *Tutorial: Two-site operators*

1 `inner(Zp' * Zp', H, Zp * Zp)`

$$\langle Z+Z+ | H | Z+Z+ \rangle$$

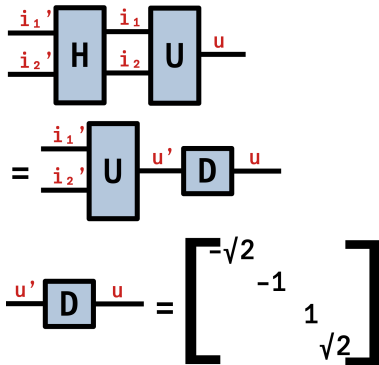
## *Tutorial: Two-site operators*

$$1 \quad \text{inner}(Z_p' * Z_p', H, Z_p * Z_p)$$



## Tutorial: Two-site operators

- 1  $D, U = \text{eigen}(H)$
- 2  $\text{diag}(D) == [-\sqrt{2}, -1, 1, \sqrt{2}]$





## *Tutorial: Custom two-site operators*

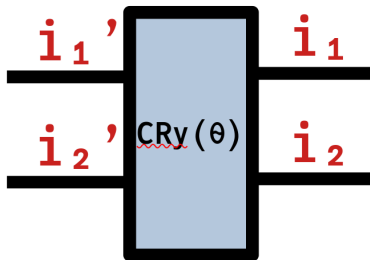
```
1 import ITensors: op
2
3 function op(
4     ::OpName"CRy",
5     ::SiteType"S=1/2";
6      $\theta$ 
7 )
8     c = cos( $\theta/2$ )
9     s = sin( $\theta/2$ )
10    return [
11        1 0 0 0
12        0 1 0 0
13        0 0 c -s
14        0 0 s  c
15    ]
16 end
```

Controlled-Ry (CRy)  
rotation gate

$\text{CRy}(\theta)$

## Tutorial: Custom two-site operators

```
1  import ITensors: op
2
3  function op(
4      ::OpName"CRy",
5      ::SiteType"S=1/2";
6       $\theta$ 
7  )
8      c = cos( $\theta/2$ )
9      s = sin( $\theta/2$ )
10     return [
11         1 0 0 0
12         0 1 0 0
13         0 0 c -s
14         0 0 s  c
15     ]
16 end
```



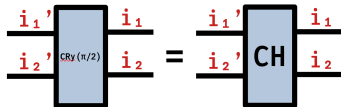
## *Tutorial: Custom two-site operators*

1 CH = op("CRy", i1, i2;  $\theta=\pi/2$ )

Controlled-Hadamard gate  
CH = CRy( $\theta=\pi/2$ )

## *Tutorial: Custom two-site operators*

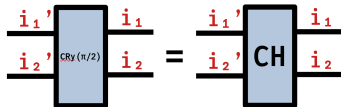
1 CH = op("CRy", i1, i2;  $\theta=\pi/2$ )



## Tutorial: Custom two-site operators

```
1 CH = op("CRy", i1, i2;  $\theta=\pi/2$ )
```

```
1 apply(CH, Zp * Zm) ==  
2 Zp1 * Xm2
```

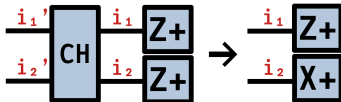
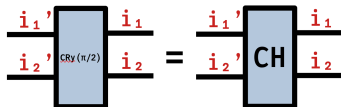


$$\text{CH}|Z+Z-\rangle = |Z+X-\rangle$$

## Tutorial: Custom two-site operators

```
1 CH = op("CRy", i1, i2;  $\theta=\pi/2$ )
```

```
1 apply(CH, Zp * Zm) ==  
2 Zp1 * Xm2
```



## *Tutorial: Two-site state optimization*

```
1 function E( $\psi$ )  
2    $\psi H \psi = \text{inner}(\psi', H, \psi)$   
3    $\psi\psi = \text{inner}(\psi, \psi)$   
4   return  $\psi H \psi / \psi\psi$   
5 end
```

$$E(\psi) = \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$$

## Tutorial: Two-site state optimization

```
1 function E( $\psi$ )  
2    $\psi H \psi = \text{inner}(\psi', H, \psi)$   
3    $\psi \psi = \text{inner}(\psi, \psi)$   
4   return  $\psi H \psi / \psi \psi$   
5 end
```

$$E(\Psi) = \frac{\begin{array}{|c|} \hline \Psi \\ \hline \end{array} \begin{array}{|c|} \hline i_1' \\ \hline i_2' \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline i_2 \end{array} \begin{array}{|c|} \hline \Psi \\ \hline \end{array}}{\begin{array}{|c|} \hline \Psi \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline i_2 \end{array} \begin{array}{|c|} \hline \Psi \\ \hline \end{array}}$$



## Tutorial: Two-site state optimization

```
1 function E( $\psi$ )
2    $\psi H \psi = \text{inner}(\psi', H, \psi)$ 
3    $\psi \psi = \text{inner}(\psi, \psi)$ 
4   return  $\psi H \psi / \psi \psi$ 
5 end
```

```
1 function minimize(f,  $\partial f$ , x;
2   nsteps,  $\gamma$ )
3   for n in 1:nsteps
4     x = x -  $\gamma * \partial f(x)$ 
5   end
6   return x
7 end
```

$$E(\Psi) = \frac{\begin{array}{|c|} \hline \Psi \\ \hline \end{array} \begin{array}{|c|} \hline i_1' \\ \hline \end{array} \begin{array}{|c|} \hline i_2' \\ \hline \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline \end{array} \begin{array}{|c|} \hline i_2 \\ \hline \end{array} \begin{array}{|c|} \hline \Psi \\ \hline \end{array}}{\begin{array}{|c|} \hline \Psi \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline \end{array} \begin{array}{|c|} \hline i_2 \\ \hline \end{array} \begin{array}{|c|} \hline \Psi \\ \hline \end{array}}$$

Gradient descent.

$f(x)$ : function to minimize.

$\partial f(x)$ : gradient of  $f$ .

$\gamma$ : step size.

## Tutorial: Two-site state optimization

```
1 function E( $\psi$ )
2    $\psi H \psi = \text{inner}(\psi', H, \psi)$ 
3    $\psi \psi = \text{inner}(\psi, \psi)$ 
4   return  $\psi H \psi / \psi \psi$ 
5 end
```

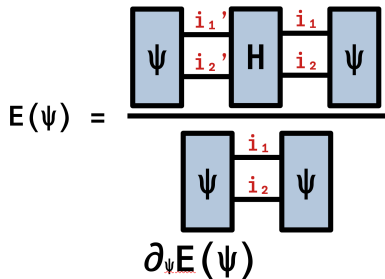
```
1 function minimize(f,  $\partial f$ , x;
2   nsteps,  $\gamma$ )
3   for n in 1:nsteps
4      $x = x - \gamma * \partial f(x)$ 
5   end
6   return x
7 end
```

$$E(\psi) = \frac{\begin{array}{|c|} \hline \psi \\ \hline \end{array} \begin{array}{|c|} \hline i_1' \\ \hline \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline \end{array} \begin{array}{|c|} \hline \psi \\ \hline \end{array}}{\begin{array}{|c|} \hline \psi \\ \hline \end{array} \begin{array}{|c|} \hline i_2' \\ \hline \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \begin{array}{|c|} \hline i_2 \\ \hline \end{array} \begin{array}{|c|} \hline \psi \\ \hline \end{array}}$$

$$\min_{\psi} E(\psi) \quad \partial_{\psi} E(\psi)$$

## Tutorial: Two-site state optimization

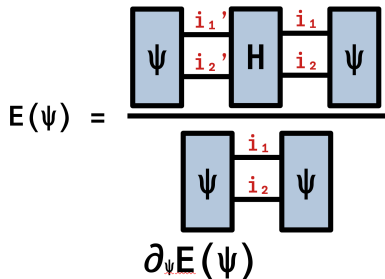
```
1  $\psi_0 = (Z_{p1} * Z_{m2} +$   
2  $Z_{m1} * Z_{p2})/\sqrt{2}$   
3  
4  $E(\psi_0) == -1$   
5  
6 using Zygote # autodiff  
7  
8  $\partial E(\psi) = \text{gradient}(E, \psi)[1]$   
9  $\text{norm}(\partial E(\psi_0)) == 2$ 
```



## Tutorial: Two-site state optimization

```
1  $\psi_0 = (Z_{p1} * Z_{m2} +$   
2  $Z_{m1} * Z_{p2})/\sqrt{2}$   
3  
4  $E(\psi_0) == -1$   
5  
6 using Zygote # autodiff  
7  
8  $\partial E(\psi) = \text{gradient}(E, \psi)[1]$   
9  $\text{norm}(\partial E(\psi_0)) == 2$ 
```

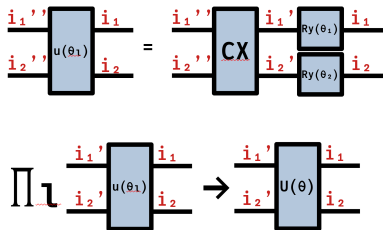
```
1  $\psi = \text{minimize}(E, \partial E, \psi_0;$   
2  $\text{nsteps}=10, \gamma=0.1)$   
3  
4  $E(\psi_0) == -1$   
5  $E(\psi) == -1.4142131 \approx -\sqrt{2}$   
6
```



$$\min_{\psi} E(\psi) = \min_{\psi} \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$$

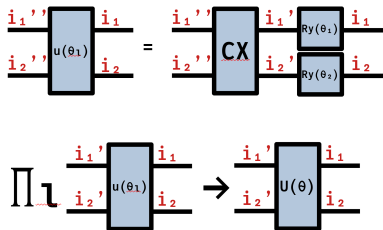
## Tutorial: Two-site circuit optimization

```
1 # Circuit as a vector of gates:
2 U( $\theta$ , i1, i2) = [
3   op("Ry", i1;  $\theta$ = $\theta$ [1]),
4   op("Ry", i2;  $\theta$ = $\theta$ [2]),
5   op("CX", i1, i2),
6   op("Ry", i1;  $\theta$ = $\theta$ [3]),
7   op("Ry", i2;  $\theta$ = $\theta$ [4]),
8 ]
9
10
11
```



## Tutorial: Two-site circuit optimization

```
1 # PastaQ notation:
2 u(θ) = [
3   ("Ry", 1, (; θ=θ[1])),
4   ("Ry", 2, (; θ=θ[2])),
5   ("CNOT", 1, 2),
6   ("Ry", 1, (; θ=θ[3])),
7   ("Ry", 2, (; θ=θ[4])),
8 ]
9
10 U(θ, i1, i2) =
11   buildcircuit(u(θ), [i1, i2])
```



## *Tutorial: Two-site circuit optimization*

```
1   $\psi_0 = Z_{p1} * Z_{p2}$ 
2
3
4  function E( $\theta$ )
5     $\psi_\theta = \text{apply}(U(\theta, i1, i2), \psi_0)$ 
6    return inner( $\psi_\theta$ , H,  $\psi_\theta$ )
7  end
```

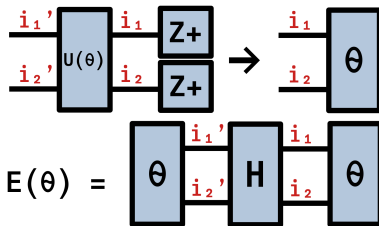
References state:

$$|0\rangle = |Z+Z+\rangle$$

$$\begin{aligned} \min_{\theta} E(\theta) \\ &= \min_{\theta} \langle 0 | U(\theta)^\dagger H U(\theta) | 0 \rangle \\ &= \min_{\theta} \langle \theta | H | \theta \rangle \end{aligned}$$

## Tutorial: Two-site circuit optimization

```
1  $\psi_0 = Z_{p1} * Z_{p2}$ 
2
3
4 function E( $\theta$ )
5    $\psi_\theta = \text{apply}(U(\theta, i_1, i_2), \psi_0)$ 
6   return inner( $\psi_\theta'$ , H,  $\psi_\theta$ )
7 end
```

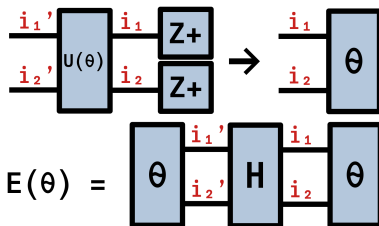




## Tutorial: Two-site circuit optimization

```
1  $\psi_0 = Z_{p1} * Z_{p2}$ 
2
3
4 function E( $\theta$ )
5    $\psi_\theta = \text{apply}(U(\theta, i_1, i_2), \psi_0)$ 
6   return inner( $\psi_\theta$ , H,  $\psi_\theta$ )
7 end
```

```
1  $\theta_0 = [0, 0, 0, 0]$ 
2  $\partial E(\theta) = \text{gradient}(E, \theta)[1]$ 
3  $\theta = \text{minimize}(E, \partial E, \theta_0;$ 
4   nsteps=40,  $\gamma=0.5$ )
5
6  $E(\theta_0) == -1$ 
7  $E(\theta) == -1.4142077 \approx -\sqrt{2}$ 
```



$$E(\theta) = \begin{array}{c} \begin{array}{|c|} \hline \theta \\ \hline \end{array} \begin{array}{|c|} \hline i_1' \\ \hline i_2' \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \begin{array}{|c|} \hline i_1 \\ \hline i_2 \end{array} \begin{array}{|c|} \hline \theta \\ \hline \end{array} \end{array}$$

$$\min_{\theta} E(\theta) \quad \partial_{\theta} E(\theta)$$

## *Tutorial: Two-site fidelity optimization*

```
1   $\psi_0 = Z_p1 * Z_p2$ 
2   $\psi = (Z_pZ_p + Z_mZ_m) / \sqrt{2}$ 
3
4
5
6  function F( $\theta$ )
7     $\psi_\theta = \text{apply}(U(\theta, i1, i2), \psi_0)$ 
8    return -abs(inner( $\psi, \psi_\theta$ ))^2
9  end
```

Reference state:

$$|0\rangle = |Z+Z+\rangle$$

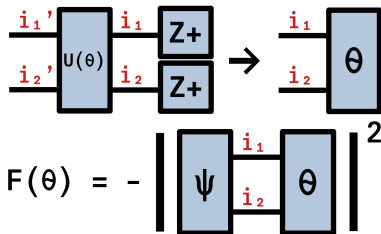
Target state:

$$|\psi\rangle = (|Z+Z+\rangle + |Z-Z-\rangle)/\sqrt{2}$$

$$\min_{\theta} F(\theta) = \min_{\theta} -|\langle\psi|U(\theta)|0\rangle|^2$$

## Tutorial: Two-site fidelity optimization

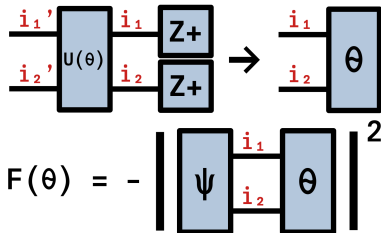
```
1  $\psi_0 = Z_{p1} * Z_{p2}$ 
2  $\psi = (Z_p Z_p + Z_m Z_m) / \sqrt{2}$ 
3
4
5
6 function F( $\theta$ )
7    $\psi_\theta = \text{apply}(U(\theta, i_1, i_2), \psi_0)$ 
8   return  $-\text{abs}(\text{inner}(\psi, \psi_\theta))^2$ 
9 end
```



## Tutorial: Two-site fidelity optimization

```
1  $\psi_0 = Z_{p1} * Z_{p2}$ 
2  $\psi = (Z_p Z_p + Z_m Z_m) / \sqrt{2}$ 
3
4
5
6 function F( $\theta$ )
7    $\psi_\theta = \text{apply}(U(\theta, i_1, i_2), \psi_0)$ 
8   return -abs(inner( $\psi, \psi_\theta$ ))^2
9 end
```

```
1  $\theta_0 = [0, 0, 0, 0]$ 
2  $\partial F(\theta) = \text{gradient}(F, \theta)[1]$ 
3  $\theta = \text{minimize}(F, \partial F, \theta_0;$ 
4   nsteps=50,  $\gamma=0.1$ )
5
6  $F(\theta_0) == -0.5$ 
7  $F(\theta) == -0.9938992 \approx -1$ 
```



$$\min_{\theta} F(\theta) \quad \partial_{\theta} F(\theta)$$

## Tutorial: $n$ -site states with MPS

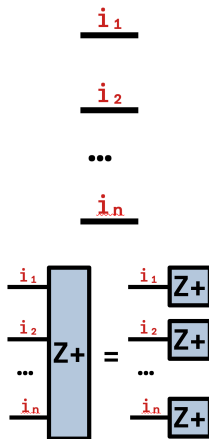
```
1  n = 30
2  i = [Index(2, "S=1/2")
3       for j in 1:n]
4
5
6  Zp = MPS(i, "Z+")
7
8  maxlinkdim(Zp) == 1 #
   product state
```

$n$ -site state

$|Z + Z + \dots Z+\rangle$

# Tutorial: $n$ -site states with MPS

```
1  n = 30
2  i = [Index(2, "S=1/2")
3       for j in 1:n]
4
5
6  Zp = MPS(i, "Z+")
7
8  maxlinkdim(Zp) == 1 #
   product state
```



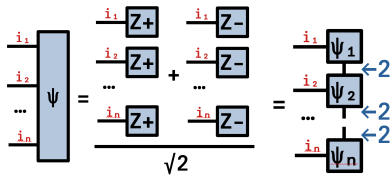
## *Tutorial: n-site states with MPS*

```
1   $\psi = (Z_p + Z_m)/\sqrt{2}$ 
2
3
4  maxlinkdim( $\psi$ ) == 2 #
    entangled state
```

$$\frac{(|Z + Z + \dots Z + \rangle + |Z - Z - \dots Z - \rangle)}{\sqrt{2}}$$

# Tutorial: $n$ -site states with MPS

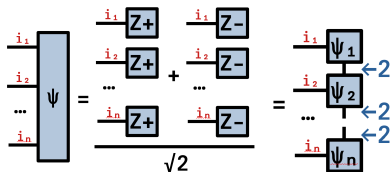
- 1  $\psi = (Z_p + Z_m)/\sqrt{2}$
- 2
- 3
- 4  $\text{maxlinkdim}(\psi) == 2 \#$   
entangled state





# Tutorial: $n$ -site states with MPS

- 1  $\psi = (Z_p + Z_m)/\sqrt{2}$
- 2
- 3
- 4  $\text{maxlinkdim}(\psi) == 2$  #  
entangled state



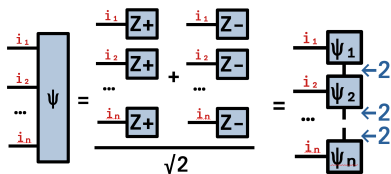
- 1  $\text{inner}(Z_p, Z_p) == 1$
- 2
- 3
- 4  $\text{inner}(Z_p, \psi) == 1/\sqrt{2}$

$$\langle Z_+ | Z_+ \rangle = 1$$

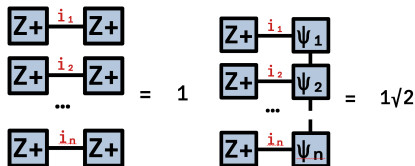
$$\langle Z_+ | \psi \rangle = 1/\sqrt{2}$$

# Tutorial: $n$ -site states with MPS

- 1  $\psi = (Z_p + Z_m)/\sqrt{2}$
- 2
- 3
- 4  $\text{maxlinkdim}(\psi) == 2$  #  
entangled state



- 1  $\text{inner}(Z_p, Z_p) == 1$
- 2
- 3
- 4  $\text{inner}(Z_p, \psi) == 1/\sqrt{2}$



## Tutorial: $n$ -site states with MPS

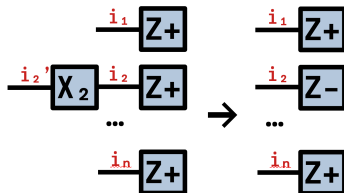
```
1  j = n ÷ 2
2  Xj = op("X", i[j])
3
4  XjZp = apply(Xj, Zp)
5
6
7  state = [k == j ? "Z-" : "Z+"
8           for k in 1:n]
9  XjZp = MPS(i, state)
```

$$X_j |Z + Z + \dots Z + \rangle = \\ |Z + Z + \dots Z - \dots Z + \rangle$$

$$|Z + Z + \dots Z - \dots Z + \rangle$$

## Tutorial: $n$ -site states with MPS

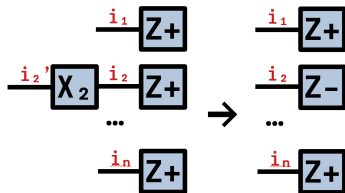
```
1  j = n ÷ 2
2  Xj = op("X", i[j])
3
4  XjZp = apply(Xj, Zp)
5
6
7  state = [k == j ? "Z-" : "Z+"
8           for k in 1:n]
9  XjZp = MPS(i, state)
```



## Tutorial: $n$ -site states with MPS

```
1  j = n ÷ 2
2  Xj = op("X", i[j])
3
4  XjZp = apply(Xj, Zp)
5
6
7  state = [k == j ? "Z-" : "Z+"
8           for k in 1:n]
9  XjZp = MPS(i, state)
```

```
1  maxlinkdim(XjZp) == 1
2  inner(Zp, XjZp) == 0
```



## *Tutorial: n-site operators with MPO*

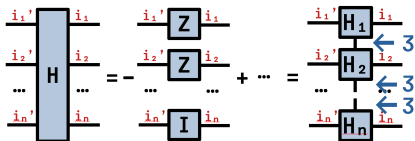
```
1 function ising(n; h)
2   H = OpSum()
3   for j in 1:(n - 1)
4     H -= "Z", j, "Z", j + 1
5   end
6   for j in 1:n
7     H += h, "X", j
8   end
9   return H
10 end
11
12 h = 0.5
13 H = MPO(ising(n; h=h), i)
14 maxlinkdim(H) == 3
```

$n$  sites

$$H = -\sum_j^{n-1} Z_j Z_{j+1} + h \sum_j^n X_j$$

## Tutorial: $n$ -site operators with MPO

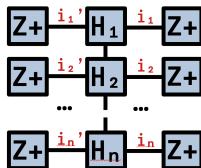
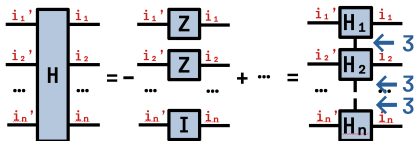
```
1 function ising(n; h)
2   H = OpSum()
3   for j in 1:(n - 1)
4     H -= "Z", j, "Z", j + 1
5   end
6   for j in 1:n
7     H += h, "X", j
8   end
9   return H
10 end
11
12 h = 0.5
13 H = MPO(ising(n; h=h), i)
14 maxlinkdim(H) == 3
```



## Tutorial: $n$ -site operators with MPO

```
1 function ising(n; h)
2   H = OpSum()
3   for j in 1:(n - 1)
4     H -= "Z", j, "Z", j + 1
5   end
6   for j in 1:n
7     H += h, "X", j
8   end
9   return H
10 end
11
12 h = 0.5
13 H = MPO(ising(n; h=h), i)
14 maxlinkdim(H) == 3
```

```
1 Zp = MPS(i, "Z+")
2 inner(Zp', H, Zp) == -(n-1)
```





## Tutorial: $n$ -site state optimization

```
1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2
3   $\psi = \text{minimize}(\text{E}, \partial\text{E}, \psi_0;$ 
4       $\text{nsteps}=50, \gamma=0.1,$ 
5       $\text{maxdim}=10, \text{cutoff}=1\text{e-}5)$ 
6
7   $\text{E}_{\text{dmrg}}, \psi_{\text{dmrg}} = \text{dmrg}(\text{H}, \psi_0;$ 
8       $\text{nsweeps}=10,$ 
9       $\text{maxdim}=10, \text{cutoff}=1\text{e-}5)$ 
```

$$E(\psi) = \frac{\langle \psi | \psi \rangle}{\langle \psi | \psi \rangle}$$

$\min_{\psi} E(\psi) \quad \partial_{\psi} E(\psi)$

# Tutorial: $n$ -site state optimization

```

1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2
3   $\psi = \text{minimize}(\text{E}, \partial\text{E}, \psi_0;$ 
4       $\text{nsteps}=50, \gamma=0.1,$ 
5       $\text{maxdim}=10, \text{cutoff}=1\text{e-}5)$ 
6
7   $\text{E}_{\text{dmrg}}, \psi_{\text{dmrg}} = \text{dmrg}(\text{H}, \psi_0;$ 
8       $\text{nsweeps}=10,$ 
9       $\text{maxdim}=10, \text{cutoff}=1\text{e-}5)$ 

```

```

1   $\text{maxlinkdim}(\psi_0) == 1$ 
2   $\text{maxlinkdim}(\psi) == 3$ 
3   $\text{maxlinkdim}(\psi_{\text{dmrg}}) == 3$ 
4   $\text{E}(\psi_0) == -29$ 
5   $\text{E}(\psi) == -31.0317917$ 
6   $\text{E}(\psi_{\text{dmrg}}) == -31.0356110$ 

```

$$E(\psi) = \langle \psi | \psi \rangle$$

$$\min_{\psi} E(\psi) \quad \partial_{\psi} E(\psi)$$

## *Tutorial: n-site circuit optimization*

```
1 Ry_layer( $\theta$ , i) = [op("Ry", i[j];  $\theta=\theta[j]$ ) for j in 1:n]
2 CX_layer(i) = [op("CX", i[j], i[j+1]) for j in 1:2:(n-1)]
```

## *Tutorial: n-site circuit optimization*

```
1 Ry_layer( $\theta$ , i) = [op("Ry", i[j];  $\theta=\theta[j]$ ) for j in 1:n]
2 CX_layer(i) = [op("CX", i[j], i[j+1]) for j in 1:2:(n-1)]
```

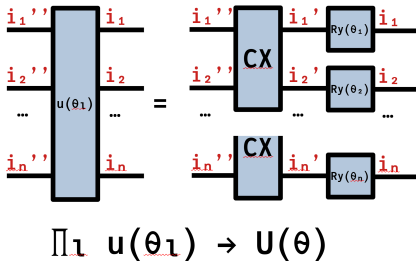
```
1 function U( $\theta$ , i; nlayers)
2   n = length(i)
3   for l in 1:(nlayers - 1)
4      $\theta_l = \theta[(1:n) .+ l * n]$ 
5      $U_\theta = [U_\theta; \text{Ry\_layer}(\theta_l, i)]$ 
6      $U_\theta = [U_\theta; \text{CX\_layer}(i)]$ 
7   end
8   return  $U_\theta$ 
9 end
```

$$\begin{aligned} U(\theta) = & \text{Ry}_1(\theta_1) \dots \text{Ry}_n(\theta_n) \\ & * \text{CX}_{1,2} \dots \text{CX}_{n-1,n} \\ & * \text{Ry}_1(\theta_{n+1}) \dots \text{Ry}_n(\theta_{2n}) \\ & * \dots \end{aligned}$$

## Tutorial: $n$ -site circuit optimization

```
1 Ry_layer( $\theta$ , i) = [op("Ry", i[j];  $\theta=\theta[j]$ ) for j in 1:n]
2 CX_layer(i) = [op("CX", i[j], i[j+1]) for j in 1:2:(n-1)]
```

```
1 function U( $\theta$ , i; nlayers)
2   n = length(i)
3   for l in 1:(nlayers - 1)
4      $\theta_l = \theta[(1:n) .+ l * n]$ 
5      $U_\theta = [U_\theta; \text{Ry\_layer}(\theta_l, i)]$ 
6      $U_\theta = [U_\theta; \text{CX\_layer}(i)]$ 
7   end
8   return  $U_\theta$ 
9 end
```

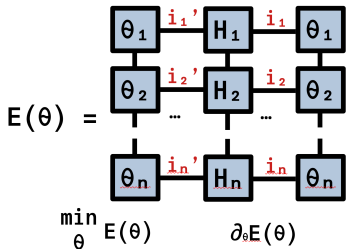
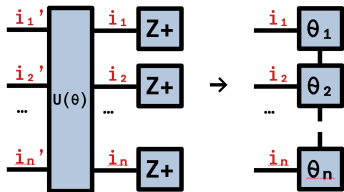


# Tutorial: $n$ -site states with MPS

```

1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2  nlayers = 6
3  function E( $\theta$ )
4     $\psi_\theta = \text{apply}(U(\theta, i; \text{nlayers}), \psi_0)$ 
5    return inner( $\psi_\theta$ , H,  $\psi_\theta$ )
6  end
7
8   $\theta_0 = \text{zeros}(\text{nlayers} * n)$ 
9   $\theta = \text{minimize}(E, \partial E, \theta_0;$ 
10    nsteps=20,  $\gamma=0.1)$ 

```



# Tutorial: $n$ -site states with MPS

```

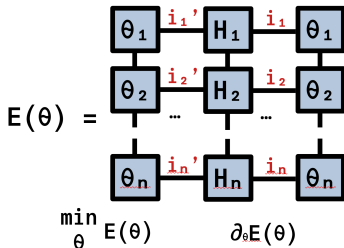
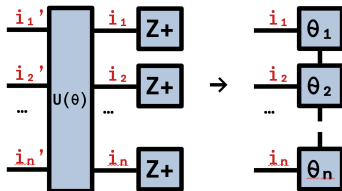
1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2  nlayers = 6
3  function E( $\theta$ )
4     $\psi_\theta = \text{apply}(\text{U}(\theta, i; \text{nlayers}), \psi_0)$ 
5    return inner( $\psi_\theta$ , H,  $\psi_\theta$ )
6  end
7
8   $\theta_0 = \text{zeros}(\text{nlayers} * n)$ 
9   $\theta = \text{minimize}(\text{E}, \partial \text{E}, \theta_0;$ 
10    nsteps=20,  $\gamma=0.1$ )

```

```

1  maxlinkdim( $\psi_0$ ) = 1
2  maxlinkdim( $\psi_\theta$ ) = 3
3   $\text{E}(\theta_0) == -29$ 
4   $\text{E}(\theta) == -31.017062$ 

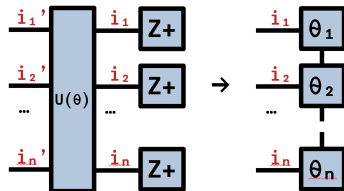
```



# Tutorial: $n$ -site states with MPS

```

1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2  nlayers = 6
3  function F( $\theta$ )
4     $\psi_\theta = \text{apply}(U(\theta, i; \text{nlayers}), \psi_0)$ 
5    return  $-\text{abs}(\text{inner}(\psi', \psi_\theta))^2$ 
6  end
7
8   $\theta_0 = \text{zeros}(\text{nlayers} * n)$ 
9   $\theta = \text{minimize}(\text{F}, \partial \text{F}, \theta_0;$ 
10    nsteps=20,  $\gamma=0.1$ )
  
```



$$F(\theta) = - \left| \begin{array}{ccc} \psi_1 & i_1 & \theta_1 \\ \psi_2 & i_2 & \theta_2 \\ \vdots & \vdots & \vdots \\ \psi_n & i_n & \theta_n \end{array} \right|^2$$

$$\min_{\theta} F(\theta) \quad \partial_{\theta} F(\theta)$$



# Tutorial: $n$ -site states with MPS

```

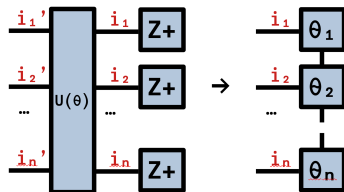
1   $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2  nlayers = 6
3  function F( $\theta$ )
4     $\psi_\theta = \text{apply}(\text{U}(\theta, i; \text{nlayers}), \psi_0)$ 
5    return -abs(inner( $\psi'$ ,  $\psi_\theta$ ))2
6  end
7
8   $\theta_0 = \text{zeros}(\text{nlayers} * n)$ 
9   $\theta = \text{minimize}(\text{F}, \partial \text{F}, \theta_0;$ 
10    nsteps=20,  $\gamma=0.1$ )

```

```

1  maxlinkdim( $\psi_0$ ) == 1
2  maxlinkdim( $\psi_\theta$ ) == 2
3  F( $\theta_0$ ) == -0.556066
4  F( $\theta$ ) == -0.995230

```



$$F(\theta) = - \left| \begin{array}{ccc} \psi_1 & i_1 & \theta_1 \\ \psi_2 & i_2 & \theta_2 \\ \vdots & \vdots & \vdots \\ \psi_n & i_n & \theta_n \end{array} \right|^2$$

$\min_{\theta} F(\theta) \qquad \partial_{\theta} F(\theta)$

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).
- ▶ Predefined differentiable circuits (PastaQ).

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).
- ▶ Predefined differentiable circuits (PastaQ).
- ▶ Monitered quantum circuits (PastaQ).

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).
- ▶ Predefined differentiable circuits (PastaQ).
- ▶ Monitered quantum circuits (PastaQ).
- ▶ GPU support (ITensor/PastaQ). Dense only for now, will support block sparse.

## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).
- ▶ Predefined differentiable circuits (PastaQ).
- ▶ Monitered quantum circuits (PastaQ).
- ▶ GPU support (ITensor/PastaQ). Dense only for now, will support block sparse.
- ▶ Conserved quantities with multithreaded block sparse tensors (ITensor/PastaQ).



## *Other features not mentioned*

- ▶ Noisy state and process simulation (PastaQ).  
AD/optimization support in progress.
- ▶ Time evolution with trotterization (ITensor/PastaQ).  
AD/optimization and TDVP support in progress.
- ▶ Quantum state and process tomograph (PastaQ).
- ▶ Predefined differentiable circuits (PastaQ).
- ▶ Monitered quantum circuits (PastaQ).
- ▶ GPU support (ITensor/PastaQ). Dense only for now, will support block sparse.
- ▶ Conserved quantities with multithreaded block sparse tensors (ITensor/PastaQ).
- ▶ More that I am probably forgetting... Ask Giacomo and me for more details.

## *Future directions/in progress*

- ▶ More AD, make ITensor fully differentiable (have some work to do, like tensor decompositions and general network contractions, more MPS/MPO functions. You will find bugs, please contribute gradient definitions!).

## *Future directions/in progress*

- ▶ More AD, make ITensor fully differentiable (have some work to do, like tensor decompositions and general network contractions, more MPS/MPO functions. You will find bugs, please contribute gradient definitions!).
- ▶ More general built-in tensor networks beyond MPS/MPO (tree tensor networks and PEPS, general contraction and optimization, use in circuit evolution and tomography, etc.).

## *Future directions/in progress*

- ▶ More AD, make ITensor fully differentiable (have some work to do, like tensor decompositions and general network contractions, more MPS/MPO functions. You will find bugs, please contribute gradient definitions!).
- ▶ More general built-in tensor networks beyond MPS/MPO (tree tensor networks and PEPS, general contraction and optimization, use in circuit evolution and tomography, etc.).
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs.

## *Future directions/in progress*

- ▶ More AD, make ITensor fully differentiable (have some work to do, like tensor decompositions and general network contractions, more MPS/MPO functions. You will find bugs, please contribute gradient definitions!).
- ▶ More general built-in tensor networks beyond MPS/MPO (tree tensor networks and PEPS, general contraction and optimization, use in circuit evolution and tomography, etc.).
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs.
- ▶ Improved gradient optimization: higher order derivatives, preconditioners, isometrically constrained gradient optimization, etc.

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).



## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).
- ▶ Quantum chemistry (for example UCC).

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).
- ▶ Quantum chemistry (for example UCC).
- ▶ Distributed computing: real space parallel circuit evolution, TDVP, and DMRG, distributed tensors, etc.

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).
- ▶ Quantum chemistry (for example UCC).
- ▶ Distributed computing: real space parallel circuit evolution, TDVP, and DMRG, distributed tensors, etc.
- ▶ Infinite MPS and tensor network tools like VUMPS and TDVP.

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).
- ▶ Quantum chemistry (for example UCC).
- ▶ Distributed computing: real space parallel circuit evolution, TDVP, and DMRG, distributed tensors, etc.
- ▶ Infinite MPS and tensor network tools like VUMPS and TDVP.
- ▶ Improved MPO compression for time evolving operators.

## *Future directions/in progress*

- ▶ Free fermions/matchgates and conversion to tensor networks.
- ▶ Automatic fermions (**Miles Stoudenmire** (CCQ), **Jing Chen** (CCQ, Zapata)).
- ▶ Non-abelian symmetries (**Miles Stoudenmire**).
- ▶ Quantum chemistry (for example UCC).
- ▶ Distributed computing: real space parallel circuit evolution, TDVP, and DMRG, distributed tensors, etc.
- ▶ Infinite MPS and tensor network tools like VUMPS and TDVP.
- ▶ Improved MPO compression for time evolving operators.
- ▶ I'm interested in building general tools and algorithms, and I'm looking for people with problems to solve. Also, we need help, code to contributions are welcome!