

Analyzing Quantum Many-Body Systems with ITensor and PastaQ

Matthew Fishman

Center for Computational Quantum Physics (CCQ)

Flatiron Institute, NY

<https://mtfishman.github.io/>

February 26, 2022



Who am I?

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



mtfishman.github.io



universität
wien

Who am I?

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



mtfishman.github.io



universität
wien

Who am I?

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



mtfishman.github.io



universität
wien

Who am I?

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



universität
wien

Who am I?

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



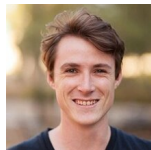
mtfishman.github.io



universität
wien

Who am I?

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



universität
wien

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

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

What is the Center for Computation 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 Computation 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: computers clusters, software development, algorithm development, and scientific applications.

What is the Center for Computation 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: computers 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 Computation 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: computers 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 Computation 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: computers 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).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.



What is ITensor?

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.
- ▶ I took over development in 2018, still co-developed with Miles.



What is ITensor?

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.
- ▶ I took over development in 2018, still co-developed 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).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.
- ▶ I took over development in 2018, still co-developed 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).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.
- ▶ I took over development in 2018, still co-developed with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.
- ▶ Website: itensor.org.



What is ITensor?

- ▶ Stands for “Intelligent Tensor”.
- ▶ Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- ▶ Development taken over by **Miles Stoudenmire** (CCQ) around 2011.
- ▶ I took over development in 2018, still co-developed with Miles.
- ▶ Originally written in C++. I led the port to Julia starting in 2019.
- ▶ **Katie Hyatt** (AWS) wrote the GPU backend in Julia.
- ▶ Website: itensor.org.
- ▶ Paper: arxiv.org/abs/2007.14822.



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.

What is Julia?

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

What is Julia?

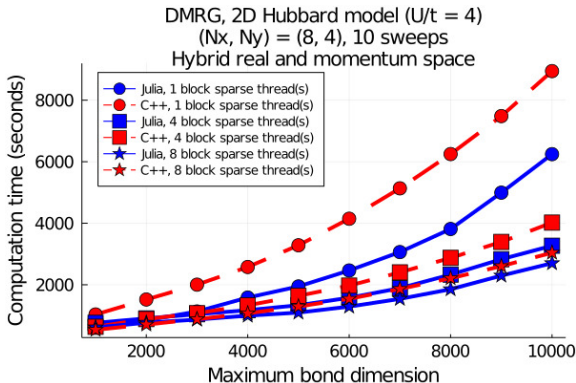
- ▶ Porting the full ITensor library took about $1\frac{1}{2}$ years.
- ▶ Much less code, easier development.

What is 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.

What is 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 PastaQ?

Pasta

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



What is PastaQ?

Pasta

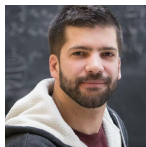
- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.



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?

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 extendable gate definitions.



What is PastaQ?



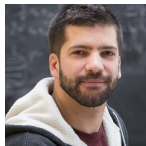
- ▶ Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.
- ▶ Quantum computing extension to ITensor in Julia.
 - ▶ Tensor network-based quantum state and process tomography.
 - ▶ Extensive and extendable gate definitions.
 - ▶ Built-in and easily extendable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



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 extendable gate definitions.
 - ▶ Built-in and easily extendable circuit definitions.
 - ▶ 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 extendable gate definitions.
 - ▶ Built-in and easily extendable circuit definitions.
 - ▶ Approximate circuit evolution and optimization with MPS/MPO, etc.
- ▶ Find out more: github.com/GTorlai/PastaQ.jl



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).
- ▶ Locally gate structure or interactions matching 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).
- ▶ Locally gate structure or interactions matching 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).
- ▶ Locally gate structure or interactions matching 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?

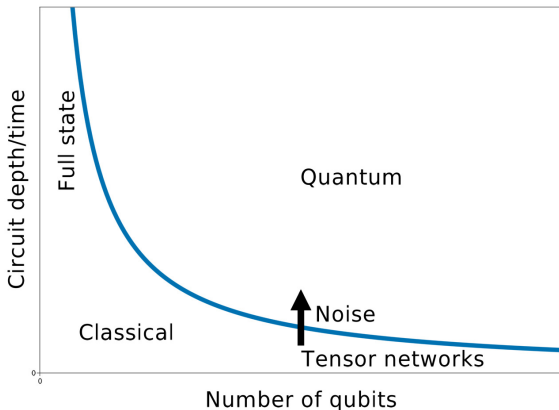
- ▶ Good for many sites or qubits but limited to shorter times/circuit depths (though better with noise).
- ▶ Locally gate structure or interactions matching the graph structure of the tensor network.
- ▶ If you need very long time evolution/many layers but few qubits, use full state simulation.
 - ▶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

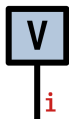
When should I use tensor networks?

- ▶ In my opinion, tensor networks are the best general purpose tool we have right now for studying quantum many-body systems (while we wait for a general purpose quantum computer).

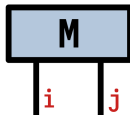


What are tensor networks?

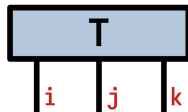
Vector V_i
Order-1 tensor



Matrix M_{ij}
Order-2 tensor

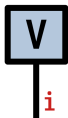


Tensor T_{ijk}
Order-3 tensor

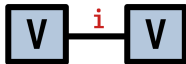


What are tensor networks?

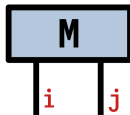
Vector V_i
Order-1 tensor



Vector inner
product



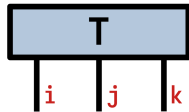
Matrix M_{ij}
Order-2 tensor



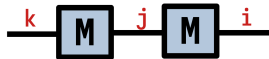
Matrix-vector
multiplication



Tensor T_{ijk}
Order-3 tensor

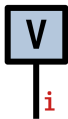


Matrix-matrix
multiplication

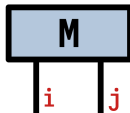


What are tensor networks?

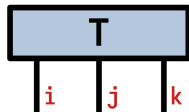
Vector V_i
Order-1 tensor



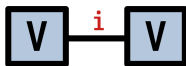
Matrix M_{ij}
Order-2 tensor



Tensor T_{ijk}
Order-3 tensor



Vector inner
product



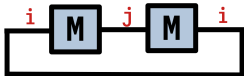
Matrix-vector
multiplication



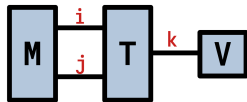
Matrix-matrix
multiplication



Trace of matrix-matrix
multiplication



General tensor
network



How do I install ITensor/PastaQ?

1. Download Julia: julialang.org/downloads

How do I install ITensor/PastaQ?

1. Download Julia: julialang.org/downloads
2. Launch Julia:

```
1 $ julia
```

How do I install ITensor/PastaQ?

1. Download Julia: 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);
6
7  julia> using PastaQ
8
9  julia> gates = [("X", 1), ("CX", (1, 3))];
10
11 julia> = 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
```



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

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

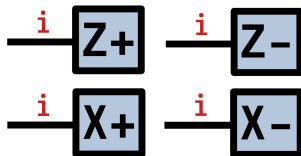
Tutorial: One-site state basics

```
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)
```

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

Tutorial: One-site state basics

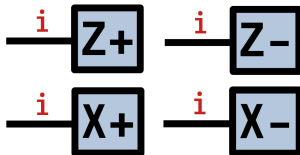
```
1 Zp = ITensor([1, 0], i)
2 Zm = ITensor([0, 1], i)
3 Xp = ITensor([1, 1]/√2, i)
4 Xm = ITensor([1, -1]/√2, i)
```



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 (Zp + Zm)/√2
2 dag(Zp) * Xp
3 (dag(Zp) * Xp)[]
4 inner(Zp, Xp)
5 norm(Xp)
```

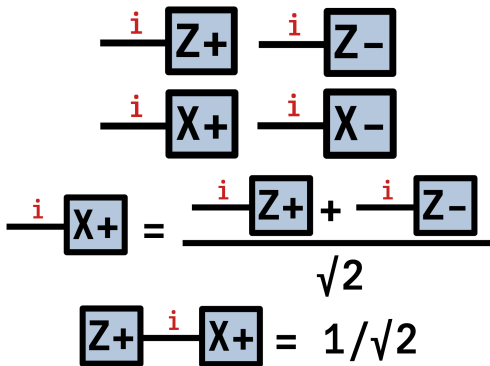


$$\begin{aligned} &\approx Xp \\ &\approx \text{ITensor}(1/\sqrt{2}) \\ &\approx 1/\sqrt{2} \\ &\approx 1/\sqrt{2} \\ &\approx 1 \end{aligned}$$

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)
```

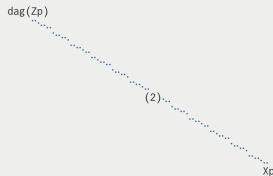
```
1 (Zp + Zm)/√2
2 dag(Zp) * Xp
3 (dag(Zp) * Xp)[]
4 inner(Zp, Xp)
5 norm(Xp)
```



Tutorial: One-site state basics

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

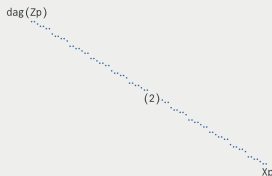
```
julia> @visualize dag(Zp) * Xp
```



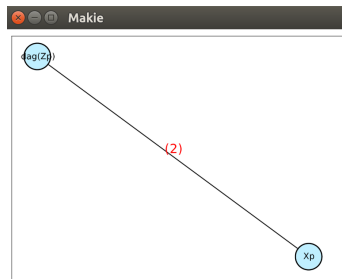
Tutorial: One-site state basics

```
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")
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

“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
```

“ $S=1/2$ ” 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 (Zp + Zm)/sqrt(2)
4 (Zp - Zm)/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-”

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 inner(Zp, iXm)
4 inner(Zm, iXm)
```

$\approx \text{im} * X_m$

$\approx \text{im}/\sqrt{2}$

$\approx -\text{im}/\sqrt{2}$

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 inner(Zp, iXm)
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)

false

Tutorial: Priming

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

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

Tutorial: Priming

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

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

i \neq j

(dim=2|id=837)

(dim=2|id=837)'

(dim=2|id=837)'

false

(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)
4 i'
5 i == i'
6 noprime(i')
```

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

$$\begin{array}{c} \underline{i} \\ \text{prime}(\underline{i}) = \underline{i'} \\ \underline{i} \neq \underline{i'} \end{array}$$

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
6 Z = ITensor(z, i', dag(i))
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

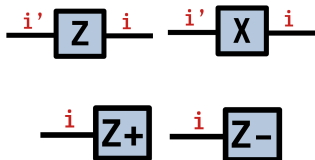
X

$|Z+\rangle$

$|Z-\rangle$

Tutorial: One-site operators

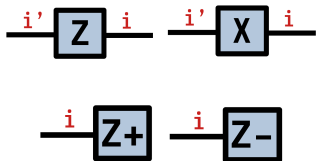
```
1 Z = op("Z", i)
2 X = op("X", i)
3
4 Zp = state("Z+", i)
5 Zm = state("Z-", i)
```



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 XZp = X * Zp
2 XZp == Zm
3 XZp == Zm'
4 noprime(XZp) == Zm
```



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

false

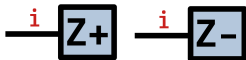
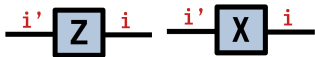
true

true

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 XZp = X * Zp
2 XZp == Zm
3 XZp == Zm'
4 noprime(XZp) == Zm
```



Tutorial: One-site operators

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

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

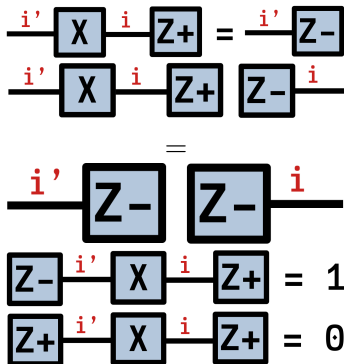
error: not a scalar value

$$\approx 1$$

$$\approx 0$$

Tutorial: One-site operators

- 1 $XZ_p = X * Z_p$
- 2
- 3 $\text{inner}(Z_m, XZ_p)$
- 4
- 5 $\text{inner}(Z_m', XZ_p)$
- 6 $\text{inner}(Z_p', XZ_p)$



Tutorial: One-site operators

1 $XZ_p = X * Z_p$

2

3 $\text{inner}(Z_m, XZ_p)$

4

5 $\text{inner}(Z_m', XZ_p)$

6 $\text{inner}(Z_p', XZ_p)$

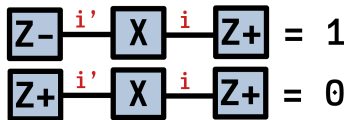
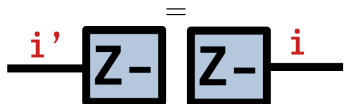
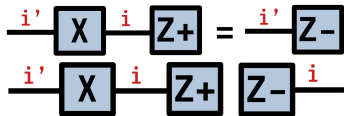
1 $(\text{dag}(Z_m)' * X * Z_p)[]$

2 $\text{inner}(Z_m', X, Z_p)$

3

4 $XZ_p = \text{apply}(X, Z_p)$

5 $\text{inner}(Z_m, XZ_p)$



≈ 1

≈ 1

$= \text{noprime}(X * Z_p)$

≈ 1

Tutorial: One-site operators

```

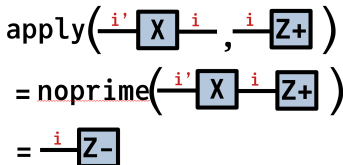
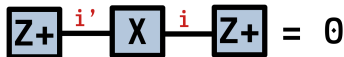
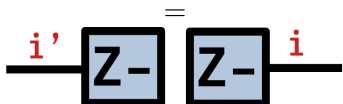
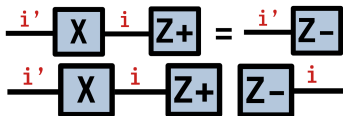
1  XZp = X * Zp
2
3  inner(Zm, XZp)
4
5  inner(Zm', XZp)
6  inner(Zp', XZp)

```

```

1  (dag(Zm)' * X * Zp)[]
2  inner(Zm', X, Zp)
3
4  XZp = apply(X, Zp)
5  inner(Zm, XZp)

```



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
```

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

```
1 im * X
```

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


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
```

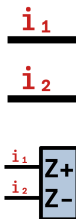
(dim=2|id=505|"S=1/2")
(dim=2|id=576|"S=1/2")

false

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

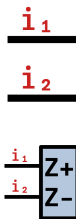
Tutorial: Two-site states

```
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
8 ZmZp = Zm1 * Zp2
```

$$|Z_+\rangle_1$$
$$|Z_+\rangle_2$$

$$|Z_-\rangle_1$$
$$|Z_-\rangle_2$$

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

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

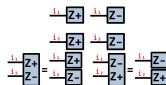
Tutorial: Two-site states

```
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
8 ZmZp = Zm1 * Zp2
```

i_1

i_2



Tutorial: Two-site states

```
1  = ITensor(i1, i2)
2  [i1=>1, i2=>2] = 1/√2
3  [i1=>2, i2=>1] = 1/√2
4
5  = (Zp1 * Zm2 +
6      Zm1 * Zp2)/√2
```

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

From single-site states

Tutorial: Two-site states

```
1  = ITensor(i1, i2)
2  [i1=>1, i2=>2] = 1/√2
3  [i1=>2, i2=>1] = 1/√2
4
5  = (Zp1 * Zm2 +
6      Zm1 * Zp2)/√2
```


$$\psi = \frac{Z_{p1} * Z_{m2} + Z_{m1} * Z_{p2}}{\sqrt{2}}$$

Tutorial: Two-site states

```
1  = ITensor(i1, i2)
2  [i1=>1, i2=>2] = 1/√2
3  [i1=>2, i2=>1] = 1/√2
4
5  = (Zp1 * Zm2 +
6      Zm1 * Zp2)/√2
```



$$\approx 1$$
$$\approx 1/\sqrt{2}$$

```
1  inner( , )
2  inner(ZpZm, )
3
4  U, S, V = svd(ZmZp, i1)
5  s = diag(S)
6
7  U, S, V = svd( , i1)
8  s = diag(S)
```

$$\approx [1, 0]$$

$$\approx [1/\sqrt{2}, 1/\sqrt{2}]$$

Tutorial: Two-site states

```

1  = ITensor(i1, i2)
2  [i1=>1, i2=>2] = 1/√2
3  [i1=>2, i2=>1] = 1/√2
4
5  = (Zp1 * Zm2 +
6      Zm1 * Zp2)/√2

```



```

1  inner( , )
2  inner(ZpZm, )
3
4  U, S, V = svd(ZmZp, i1)
5  s = diag(S)
6
7  U, S, V = svd( , i1)
8  s = diag(S)

```



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 sites

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

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 sites

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

Alternative:

Build from single-site
operators.

Less error-prone.

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 sites

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



Tutorial: Two-site operators

[TODO: Add visualization of $\langle H \rangle$]

```
1 ZpZp = Zp1 * Zp2
2
3
4
5 (dag(ZpZp)' * H * ZpZp)[]
6 inner(ZpZp', H, ZpZp)
7 inner(ZpZp, apply(H, ZpZp))
```

```
# Expectation value:
#  $\langle H \rangle = \langle Z+Z+ | H | Z+Z+ \rangle$ 
```

```
#  $\approx -1$ 
```

Tutorial: Two-site operators

[TODO: Add visualization of $\langle H \rangle$]

```
1 ZpZp = Zp1 * Zp2
2
3
4
5 (dag(ZpZp)' * H * ZpZp)[]
6 inner(ZpZp', H, ZpZp)
7 inner(ZpZp, apply(H, ZpZp))
```

Expectation value:
$\langle H \rangle = \langle Z+Z+ | H | Z+Z+ \rangle$

≈ -1

```
1 D, U = eigen(H)
2 diag(D)
```

$\approx [-\sqrt{2}, -1, 1, \sqrt{2}]$

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

# CRy( $\theta$ )
```

Tutorial: Custom two-site operators

[TODO: Add visualization of CH|ZpZm>]

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

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

Tutorial: Custom two-site operators

[TODO: Add visualization of $\text{CH}|Z_p Z_m\rangle$]

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

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

```
1 ZpZm = Zp1 * Zm2  
2  
3 CH_Xm = apply(CH, ZpZm)  
4  
5 CH_Xm  $\approx$  Zp1 * Xm2
```

```
#  $|Z+Z-\rangle = |Z+\rangle_1 |Z-\rangle_2$   
#  $\text{CH}|Z+Z-\rangle = |Z+X-\rangle$   
# true
```


Tutorial: Two-site state optimization

[TODO: Add visualization of minimizing $\langle \psi | H | \psi \rangle$]

```
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
```

```
# Function to minimize:
# Expectation value of the
# energy.
#
#  $E(\psi) = \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$ 
```

Tutorial: Two-site state optimization

[TODO: Add visualization of minimizing $\langle \mathbf{v} | \mathbf{H} | \mathbf{v} \rangle$]

```
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
```

```
# Function to minimize:
# Expectation value of the
# energy.
#
#  $E(\psi) = \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$ 
```

```
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
```

```
# Simple gradient descent.
# Must provide function f(x)
# to minimize and  $\partial f(x)$ ,
# the gradient of f at x.

#  $\gamma$  is the gradient
# descent step size.
```

Tutorial: Two-site state optimization

[TODO: Add visualization of minimizing $\langle \mathbf{v} | \mathbf{H} | \mathbf{v} \rangle$]

```
1  $\psi_0 = (\text{Zp1} * \text{Zm2} +$   
2  $\text{Zm1} * \text{Zp2})/\sqrt{2}$ 
```

```
3
```

```
4  $E(\psi_0)$ 
```

```
5
```

```
6 using Zygote: gradient
```

```
7  $\partial E(\psi) = \text{gradient}(E, \psi)[1]$ 
```

```
8
```

```
9 norm( $\partial E(\psi_0)$ )
```

```
#  $|\psi_0\rangle = (|Z+Z+\rangle +$   
#  $|Z-Z-\rangle)/\sqrt{2}$ 
```

```
#  $\approx -1$ 
```

```
# Using Zygote for automatic  
# differentiation of the energy.
```

```
#  $\approx 2$ 
```

Tutorial: Two-site state optimization

[TODO: Add visualization of minimizing $\langle \mathbf{v} | \mathbf{H} | \mathbf{v} \rangle$]

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

```
#  $|\psi_0\rangle = (|Z+Z+\rangle +$   
#       $|Z-Z-\rangle) / \sqrt{2}$ 
```

```
#  $\approx -1$ 
```

```
# Using Zygote for automatic  
# differentiation of the energy.
```

```
#  $\approx 2$ 
```

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

```
# Minimize over  $\psi$ :  
#  $\mathbf{E}(\psi) = \langle \psi | \mathbf{H} | \psi \rangle / \langle \psi | \psi \rangle$ 
```

```
#  $(-1, 2)$ 
```

```
#  $(-1.4142131, 0.0010865277)$ 
```

```
#  $\approx (-\sqrt{2}, 0)$ 
```

Tutorial: Two-site circuit optimization

[TODO: Add visualization of U]

```
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
```

```
# PastaQ notation:
u( $\theta$ , j1, j2) = [
    ("Ry", j1, (;  $\theta$ = $\theta$ [1])),
    ("Ry", j2, (;  $\theta$ = $\theta$ [2])),
    ("CNOT", j1, j2),
    ("Ry", j1, (;  $\theta$ = $\theta$ [3])),
    ("Ry", j2, (;  $\theta$ = $\theta$ [4])),
]
U( $\theta$ , i1, i2) =
    buildcircuit(u( $\theta$ , 1, 2), [i1, i2])
```

Tutorial: Two-site circuit optimization

[TODO: Add visualization of minimizing $\langle 0 | U H U | 0 \rangle$]

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

```
# References state:
#  $|0\rangle = |Z+Z+\rangle$ 

# Find  $\theta$  that minimizes:
#  $E(\theta) = \langle 0 | U(\theta)^\dagger H U(\theta) | 0 \rangle$ 
#            $= \langle \theta | H | \theta \rangle$ 
```

Tutorial: Two-site circuit optimization

[TODO: Add visualization of minimizing $\langle 0 | U H U | 0 \rangle$]

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

```
# References state:
#  $|0\rangle = |Z+Z+\rangle$ 

# Find  $\theta$  that minimizes:
#  $E(\theta) = \langle 0 | U(\theta)^\dagger H U(\theta) | 0 \rangle$ 
#            $= \langle \theta | H | \theta \rangle$ 
```

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

```
#  $(-1, \sqrt{3}/2)$ 
#  $(-1.4142077, 0.0017584116)$ 
#  $\approx (-\sqrt{2}, 0)$ 
```

Tutorial: Two-site fidelity optimization

[TODO: Add visualization of minimizing $\langle \psi | U | \psi_0 \rangle$]

```
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    $\psi_\theta = \text{apply}(U(\theta, i1, i2), \psi_0)$ 
8   return -abs(inner( $\psi, \psi_\theta$ ))^2
9 end
```

```
# Reference state:
#  $|0\rangle = |Z+Z+\rangle$ 

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

# Find  $\theta$  that minimizes:
#  $F(\theta) = -|\langle \psi | U(\theta) | 0 \rangle|^2$ 
```


Tutorial: Two-site fidelity optimization

[TODO: Add visualization of minimizing $\langle \psi | U | \psi_0 \rangle$]

```
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     $\psi_\theta = \text{apply}(U(\theta, i1, i2), \psi_0)$ 
8    return -abs(inner( $\psi, \psi_\theta$ ))^2
9  end
```

```
# Reference state:
#  $|0\rangle = |Z+Z+\rangle$ 

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

# Find  $\theta$  that minimizes:
#  $F(\theta) = -|\langle \psi | U(\theta) | 0 \rangle|^2$ 
```

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

```
#  $\approx (-0.5, 0.5)$ 
#  $\approx (-0.9938992, 0.07786879)$ 
```

Tutorial: n -site states with MPS

[TODO: Add visualization of $\langle Z_p | \rangle$]

```
1  n = 30
2  i = [Index(2, "S=1/2")
3      for j in 1:n]
4
5  Zp = MPS(i, "Z+")
6  Zm = MPS(i, "Z-")
7
8  = (Zp + Zm)/√2
9
```

n -site state

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

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

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

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

Tutorial: n -site states with MPS

[TODO: Add visualization of $\langle Z_p | \rangle$]

```
1  n = 30
2  i = [Index(2, "S=1/2")
3      for j in 1:n]
4
5  Zp = MPS(i, "Z+")
6  Zm = MPS(i, "Z-")
7
8  = (Zp + Zm)/√2
9
```

n -site state

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

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

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

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

```
1  maxlinkdim(Zp)
2  maxlinkdim( )
3  inner(Zp, Zp)
4  inner(Zm, Zp)
5  norm( )
6
```

1 (product state)

2 (entangled state)

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

$\langle Z- | Z+ \rangle \approx 0$

$(\langle Z+ | + \langle Z- |)(|Z+ \rangle + |Z- \rangle)/2$

≈ 1

Tutorial: n -site states with MPS

[TODO: Add visualization of minimizing $\langle \psi | H | \psi \rangle$]

```
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)
```

```
# j = n ÷ 2
# Xj

# Xj|Z+Z+...Z+⟩ =
#   |Z+Z+...Z-...Z+⟩

# |Z+Z+...Z-...Z+⟩
```

Tutorial: n -site states with MPS

[TODO: Add visualization of minimizing $\langle \psi | H | \psi \rangle$]

```
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)
```

```
# j = n ÷ 2
# Xj
# Xj|Z+Z+...Z+⟩ =
#   |Z+Z+...Z-...Z+⟩
# |Z+Z+...Z-...Z+⟩
```

```
1  maxlinkdim(XjZp)
2  inner(Zp, XjZp)
3  inner(XjZp, apply(Xj, Zp))
```

```
# 1 (product state)
# ≈ 0
# ≈ 1
```

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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
```

n sites

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

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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
```

n sites

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

```
1 h = 0.5
2 H = MPO(ising(n; h=h), i)
3 maxlinkdim(H)
4 Zp = MPS(i, "Z+")
5 inner(Zp', H, Zp)
```

= 3 (local Hamiltonian)

$\langle Z_+ Z_+ \dots Z_+ | H | Z_+ Z_+ \dots Z_+ \rangle$

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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      $\text{maxdim}=10, \text{cutoff}=1\text{e-}5)$ 
```

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

Minimize over ψ :
$\langle\psi|\text{H}|\psi\rangle/\langle\psi|\psi\rangle$

DMRG solves:
$\text{H}|\psi\rangle \approx |\psi\rangle$

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

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

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

Minimize over ψ :

$\langle\psi|\mathbf{H}|\psi\rangle/\langle\psi|\psi\rangle$

DMRG solves:

$\mathbf{H}|\psi\rangle \approx |\psi\rangle$

```
1   $\text{maxlinkdim}(\psi_0)$ 
2   $\text{maxlinkdim}(\psi)$ 
3   $\text{maxlinkdim}(\psi_{\text{dmrg}})$ 
4   $\mathbf{E}(\psi_0), \text{norm}(\partial(\psi_0))$ 
5   $\mathbf{E}(\psi), \text{norm}(\partial\mathbf{E}(\psi))$ 
6   $\mathbf{E}(\psi_{\text{dmrg}}), \text{norm}(\partial\mathbf{E}(\psi_{\text{dmrg}}))$ 
```

= 1 (product state)

= 3 (entangled state)

= 2 (entangled state)

$\approx (-29, 5.4772256)$

$\approx (-31.0317917, 0.06673780)$

$\approx (-31.0356110, 0.02413237)$

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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 states with MPS

[TODO: Add visualization of H]

```
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    $U_\theta$  = Ry_layer( $\theta[1:n]$ , i)
4   for l in 1:(nlayers - 1)
5      $\theta_l$  =  $\theta[(1:n) .+ l * n]$ 
6      $U_\theta$  = [ $U_\theta$ ; CX_layer(i)]
7      $U_\theta$  = [ $U_\theta$ ; Ry_layer( $\theta_l$ , i)]
8   end
9   return  $U_\theta$ 
10 end
```

```
#  $U(\theta) = \text{Ry}_1(\theta_1) \dots \text{Ry}_n(\theta_n)$ 
#            $\text{CX}_{1,2} \dots \text{CX}_{n-1,n}$ 
#            $\text{Ry}_1(\theta_{n+1}) \dots \text{Ry}_n(\theta_{2n})$ 
#           ...
```

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

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

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

Minimize over θ :

$|\theta\rangle = U(\theta)|0\rangle$

$E(\theta) = \langle \theta | H | \theta \rangle$

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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  $\theta = \text{minimize}(\text{E}, \partial\text{E}, \theta_0;$ 
10     nsteps=20,  $\gamma=0.1$ )
```

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

Minimize over θ :

$|\theta\rangle = U(\theta)|0\rangle$

$E(\theta) = \langle\theta|H|\theta\rangle$

```
1 maxlinkdim( $\psi_0$ )
2 maxlinkdim( $\psi_\theta$ )
3 E( $\theta_0$ ), norm( $\partial\text{E}(\theta_0)$ )
4 E( $\theta$ ), norm( $\partial\text{E}(\theta)$ )
```

1 (product state)

2 (entangled state)

(-29, 5.773335)

(-31.017062, 0.000759)

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
1  $\psi_0 = \text{MPS}(i, \text{"Z+"})$ 
2  $\text{nlayers} = 6$ 
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  $\theta = \text{minimize}(F, \partial F, \theta_0;$ 
10    $\text{nsteps}=20, \gamma=0.1)$ 
```

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

Minimize over θ :

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

Tutorial: n -site states with MPS

[TODO: Add visualization of H]

```
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,  $\gamma=0.1$ )
```

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

Minimize over θ :

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

$\quad = -|\langle \psi | \theta \rangle|^2$

```
1 maxlinkdim( $\psi_0$ )
2 maxlinkdim( $\psi_\theta$ )
3 F( $\theta_0$ ), norm( $\partial\text{F}(\theta_0)$ )
4 F( $\theta$ ), norm( $\partial\text{F}(\theta)$ )
```

1 (product state)

2 (entangled state)

(-0.556066, 0.895717)

(-0.995230, 0.048939)

Future directions

- ▶ 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!).

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs
- ▶ Many ongoing projects and directions: quantum chemistry (for example UCC), real space parallel DMRG, TDVP, and TEBD, MPO compression tools, general approximate contraction techniques for unstructured networks, contracting and optimizing general tensor networks with AD, infinite MPS and tensor network tools like VUMPS and TDVP, trying out different network topologies for noisy circuit tomography, simulation and optimization.

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs
- ▶ Many ongoing projects and directions: quantum chemistry (for example UCC), real space parallel DMRG, TDVP, and TEBD, MPO compression tools, general approximate contraction techniques for unstructured networks, contracting and optimizing general tensor networks with AD, infinite MPS and tensor network tools like VUMPS and TDVP, trying out different network topologies for noisy circuit tomography, simulation and optimization.
- ▶ Building general tools, looking for people with problems and code to contribute.

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs
- ▶ Many ongoing projects and directions: quantum chemistry (for example UCC), real space parallel DMRG, TDVP, and TEBD, MPO compression tools, general approximate contraction techniques for unstructured networks, contracting and optimizing general tensor networks with AD, infinite MPS and tensor network tools like VUMPS and TDVP, trying out different network topologies for noisy circuit tomography, simulation and optimization.
- ▶ Building general tools, looking for people with problems and code to contribute.

Future directions

- ▶ 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!).
- ▶ More general built-in tensor networks beyond MPS/MPO
- ▶ More HPC with multithreaded and multiprocessor parallelism and GPUs
- ▶ Many ongoing projects and directions: quantum chemistry (for example UCC), real space parallel DMRG, TDVP, and TEBD, MPO compression tools, general approximate contraction techniques for unstructured networks, contracting and optimizing general tensor networks with AD, infinite MPS and tensor network tools like VUMPS and TDVP, trying out different network topologies for noisy circuit tomography, simulation and optimization.
- ▶ Building general tools, looking for people with problems and code to contribute.