Analyzing Quantum Many-Body Systems with ITensor and PastaQ

Matthew Fishman

Center for Computational Quantum Physics (CCQ)

Flatiron Institute, NY

mtfishman.github.io

github.com/mtfishman/ITensorTutorials.jl

February 27, 2022

Who am I?

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

mtfishman.github.io

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

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).
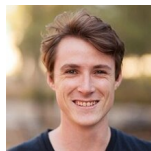
▶ Thesis on developing tensor network algorithms.

mtfishman.github.io

FLATIRON
INSTITUTE
Center for Computational
Quantum Physics

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

FLATIRON INSTITUTE
Center for Computational Quantum Physics

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).

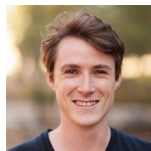mtfishman.github.io

FLATIRON INSTITUTE
Center for Computational
Quantum Physics

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)?

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 developing 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 developing ITensor around 2011.

▶ I started working on ITensor 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).
- **Miles Stoudenmire** (CCQ) started developing ITensor around 2011.
- I started working on ITensor 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).

► **Miles Stoudenmire** (CCQ) started developing ITensor around 2011.

► I started working on ITensor 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).

▶ **Miles Stoudenmire** (CCQ) started developing ITensor around 2011.

▶ I started working on ITensor 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.

▶ 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 developing ITensor around 2011.
- I started working on ITensor 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.
- Used in over **400** research papers.
- Website: itensor.org.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** of UCI (inventor of the DMRG algorithm).
- **Miles Stoudenmire** (CCQ) started developing ITensor around 2011.
- I started working on ITensor 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.
- Used in over **400** research papers.
- 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.
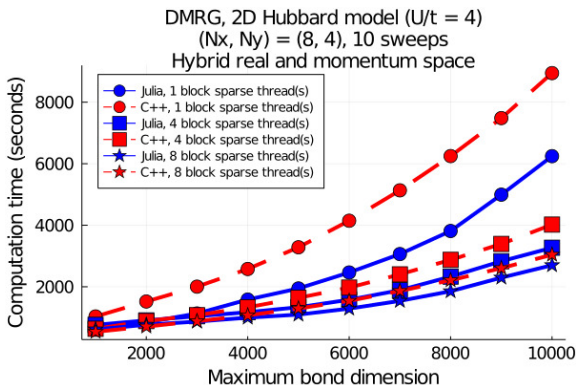
DMRG, 2D Hubbard model (U/t = 4)
(Nx, Ny) = (8, 4), 10 sweeps
Hybrid real and momentum space

What is PastaQ?

► Initiated by **Giacomo Torlai** (AWS)
  while he was a postdoc at CCQ,
  co-developed by Giacomo and me.
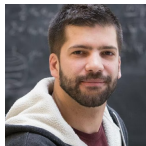
What is PastaQ?

- ▶ 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.

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.
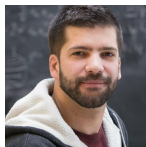
What is PastaQ?

- ▶ 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.

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.
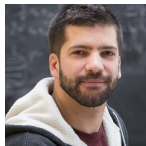
What is PastaQ?

- ▶ 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).
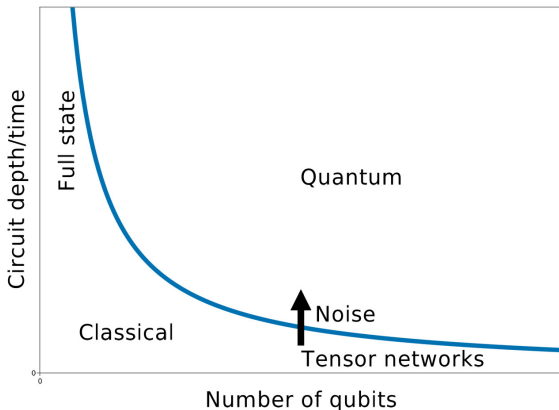
What are tensor networks?

Vector V_i
Order-1 tensor
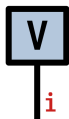
Matrix M_{ij}
Order-2 tensor
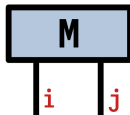
Tensor T_{ijk}
Order-3 tensor
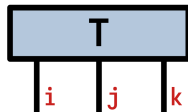
What are tensor networks?
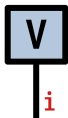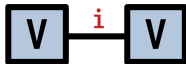
Vector V_i
Order-1 tensor

Matrix M_{ij}
Order-2 tensor

Tensor T_{ijk}
Order-3 tensor
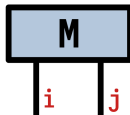
Vector inner
product

Matrix-vector
multiplication
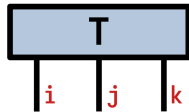
Matrix-matrix
multiplication
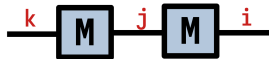
What are tensor networks?
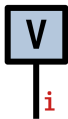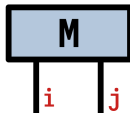
Vector V_i	Matrix M_{ij}	Tensor T_{ijk}
Order-1 tensor	Order-2 tensor	Order-3 tensor
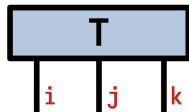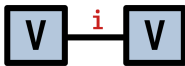
Vector inner product	Matrix-vector multiplication	Matrix-matrix multiplication
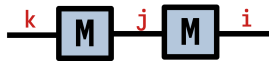
Trace of matrix-matrix multiplication
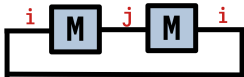
General tensor network
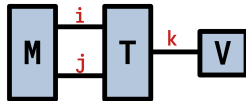
How do I install ITensor/PastaQ?

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> $\psi$ = 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)
```


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

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)
```

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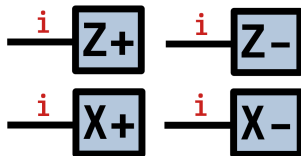
```
1  (Zp + Zm)/√2
2  dag(Zp) * Xp
3  (dag(Zp) * Xp)[]
4  inner(Zp, Xp)
5  norm(Xp)
```
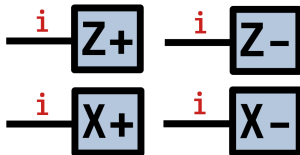

$\approx$ Xp
$\approx$ ITensor($1/\sqrt{2}$)
$\approx$ $1/\sqrt{2}$
$\approx$ $1/\sqrt{2}$
$\approx$ 1

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
```
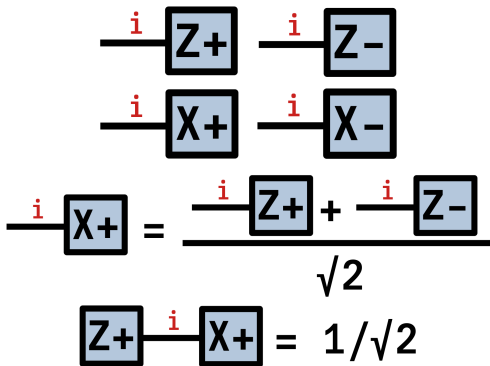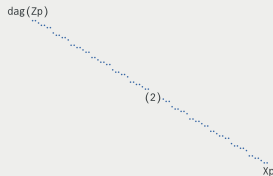
Tutorial: One-site state basics


```
1  using
       ITensorUnicodePlots
2
3  @visualize dag(Zp) * Xp
```
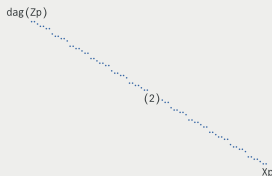
```
1  using ITensorGLMakie
2
3  @visualize dag(Zp) * Xp
```
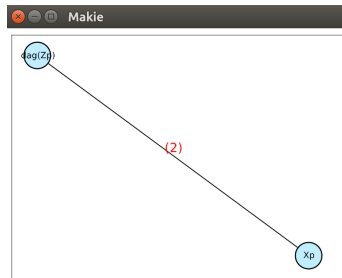
Tutorial: One-site states

```
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)/√2
4  (Zp - Zm)/√2
```

Tutorial: Custom one-site states

```julia
import ITensors: state



function state(
  ::StateName"iX-",
  ::SiteType"S=1/2"
)
  return [im -im]/√2
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)
```

$\approx$ im * Xm

$\approx$ im/√2
$\approx$ -im/√2

Tutorial: Custom one-site states

```julia
import ITensors: state



function state(
  ::StateName"iX-",
  ::SiteType"S=1/2"
)
  return [im -im]/√2
end
```

Overload ITensors.jl behavior

Define a state with the name "iX-"

```julia
iXm = state("iX-", i)

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

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

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
```

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

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
```

$$\frac{}{}_{\textcolor{red}{i}} \neq \frac{}{}_{\textcolor{red}{j}}$$

(dim=2|id=837)

(dim=2|id=837)'
(dim=2|id=837)'
true
true

Tutorial: Priming

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

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

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

$$\underline{\textcolor{red}{i}}$$

$$\texttt{prime}\left(\underline{\textcolor{red}{i}}\right) = \underline{\textcolor{red}{i'}}$$

$$\underline{\textcolor{red}{i}} \neq \underline{\textcolor{red}{i'}}$$

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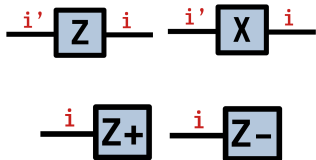
```
1  Z = op("Z", i)
2  X = op("X", i)
3
4  Zp = state("Z+", i)
5  Zm = state("Z-", i)
```

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
```
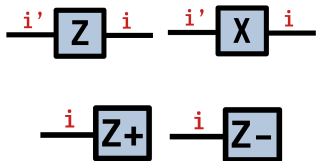
$X|Z+\rangle = |Z-\rangle$
false
true
true

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
```
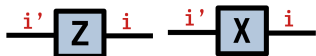
Tutorial: One-site operators

```
1   XZp = X * Zp
2
3   dag(Zm) * XZp
4
5   dag(Zm') * XZp
6   dag(Zp') * XZp
7
8   dag(Zm)' * X * Zp
9   inner(Zm', X, Zp)
```

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

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

$$\langle Z - |X|Z+\rangle \approx 1$$
$$\langle Z + |X|Z+\rangle \approx 0$$

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

Tutorial: One-site operators

```
1   XZp = X * Zp
2
3   dag(Zm) * XZp
4
5   dag(Zm') * XZp
6   dag(Zp') * XZp
7
8   dag(Zm)' * X * Zp
9   inner(Zm', X, Zp)
```

Tutorial: One-site operators

```
1  XZp = X * Zp
2
3  dag(Zm) * XZp
4
5  dag(Zm') * XZp
6  dag(Zp') * XZp
7
8  dag(Zm)' * X * Zp
9  inner(Zm', X, Zp)
```

```
1  XZp = apply(X, Zp)
2  inner(Zm, XZp)
```


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

≈ 1

Tutorial: One-site operators

```
1  XZp = X * Zp
2
3  dag(Zm) * XZp
4
5  dag(Zm') * XZp
6  dag(Zp') * XZp
7
8  dag(Zm)' * X * Zp
9  inner(Zm', X, Zp)
```

```
1  XZp = apply(X, Zp)
2  inner(Zm, XZp)
```

Tutorial: Custom one-site operators

```julia
import ITensors: op

function op(
  ::OpName"iX",
  ::SiteType"S=1/2"
)
  return [
    0 im
    im 0
  ]
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
```

Tutorial: Custom one-site operators

```julia
import ITensors: op

function op(
  ::OpName"iX",
  ::SiteType"S=1/2"
)
  return [
    0 im
    im 0
  ]
end
```

```julia
op("iX", 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)
```

```
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")

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
```
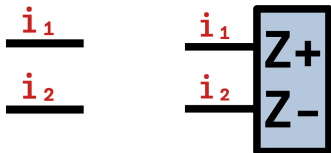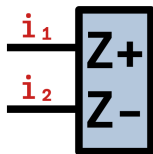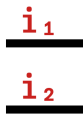


```
1   Zp1 = state("Z+", i1)
2   Zp2 = state("Z+", i2)
3
4   Zm1 = state("Z-", i1)
5   Zm2 = state("Z-", i2)
6
7   ZpZm = Zp1 * Zm2
8   ZmZp = Zm1 * Zp2
```

$|Z+\rangle_1$
$|Z+\rangle_2$

$|Z-\rangle_1$
$|Z-\rangle_2$

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

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
```
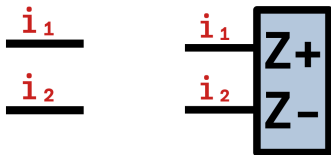


```
1  Zp1 = state("Z+", i1)
2  Zp2 = state("Z+", i2)
3
4  Zm1 = state("Z-", i1)
5  Zm2 = state("Z-", i2)
6
7  ZpZm = Zp1 * Zm2
8  ZmZp = Zm1 * Zp2
```
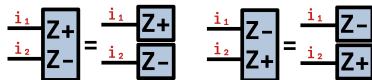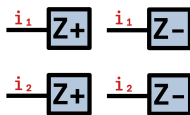
Tutorial: Two-site states

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

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

From single-site states

Tutorial: Two-site states
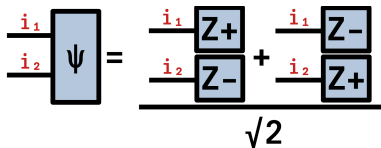
```
1   ψ = ITensor(i1, i2)
2   ψ[i1=>1, i2=>2] = 1/√2
3   ψ[i1=>2, i2=>1] = 1/√2
4
5   ψ = (Zp1 * Zm2 +
6        Zm1 * Zp2)/√2
```

Tutorial: Two-site states

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



```
1   inner(ψ, ψ)
2   inner(ZpZm, ψ)
3
4   U, S, V = svd(ZmZp, i1)
5   s = diag(S)
6
7   U, S, V = svd(ψ, i1)
8   s = diag(S)
```
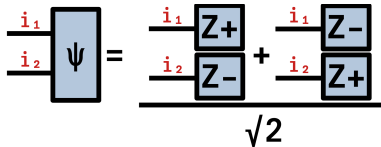
≈ 1
$\approx 1/\sqrt{2}$

$\approx [1, 0]$

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

Tutorial: Two-site states

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



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

Tutorial: Two-site states


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

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

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

Tutorial: Two-site operators

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

Make a Hamiltonian:
Transverse field Ising ($n = 2$)

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

Alternative:
Build from single-site operators.
(Less error-prone.)

```
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)
```

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 ($n = 2$)

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

```
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)
```
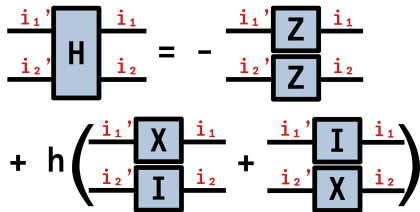
Tutorial: Two-site operators

```
1   ZpZp = Zp1 * Zp2
2
3
4
5   (dag(ZpZp)' * H * ZpZp)[]
6   inner(ZpZp', H, ZpZp)
7   inner(ZpZp, apply(H, ZpZp))
```
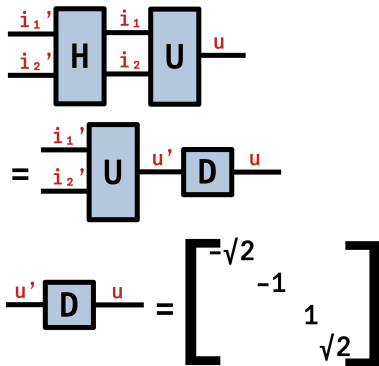
Expectation value:
$\langle H \rangle = \langle Z{+}Z{+}|H|Z{+}Z{+}\rangle$

$\approx$ -1

Tutorial: Two-site operators

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

Tutorial: Two-site operators

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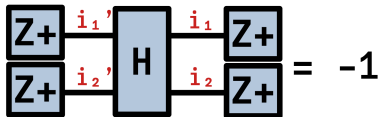


```
1  D, U = eigen(H)
2  diag(D)
```
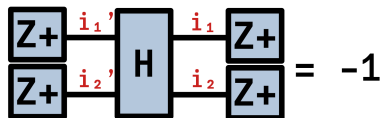
$$\approx [-\sqrt{2}, -1, 1, \sqrt{2}]$$
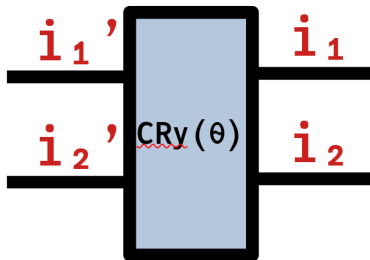
Tutorial: Two-site operators

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

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

Tutorial: Custom two-site operators

```julia
import ITensors: op

function op(
  ::OpName"CRy",
  ::SiteType"S=1/2";
  θ
)
  c = cos(θ/2)
  s = sin(θ/2)
  return [
    1 0 0  0
    0 1 0  0
    0 0 c -s
    0 0 s  c
  ]
end
```

Controlled-Ry (CRy)
rotation gate

$CRy(\theta)$

Tutorial: Custom two-site operators

```julia
import ITensors: op

function op(
  ::OpName"CRy",
  ::SiteType"S=1/2";
  θ
)
  c = cos(θ/2)
  s = sin(θ/2)
  return [
    1 0 0  0
    0 1 0  0
    0 0 c -s
    0 0 s  c
  ]
end
```

Tutorial: Custom two-site operators

```
1   CH = op("CRy", i1, i2;
2           θ=π/2)
```

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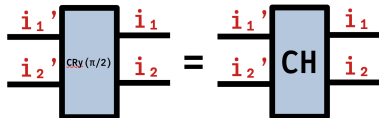CH = CRy($\theta=\pi/2$)

Tutorial: Custom two-site operators
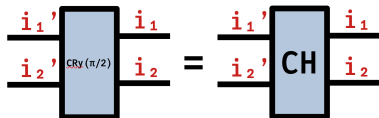
```
1    CH = op("CRy", i1, i2;
2            θ=π/2)
```

Tutorial: Custom two-site operators

```
1   CH = op("CRy", i1, i2;
2            θ=π/2)
```



```
1   ZpZm = Zp1 * Zm2
2
3   CH_Xm = apply(CH, ZpZm)
4
5   CH_Xm ≈ Zp1 * Xm2
```
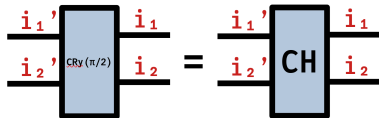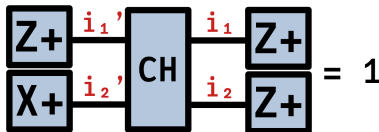
$|Z+Z\text{-}\rangle = |Z+\rangle_1 |Z\text{-}\rangle_2$

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

true

Tutorial: Custom two-site operators

```
1  CH = op("CRy", i1, i2;
2          θ=π/2)
```



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

Tutorial: Two-site state optimization

```
1  function E(ψ)
2    ψHψ = inner(ψ', H, ψ)
3    ψψ = inner(ψ, ψ)
4    return ψHψ / ψψ
5  end
```

Function to minimize: Expectation value of the energy.

$$E(\psi) = \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$$
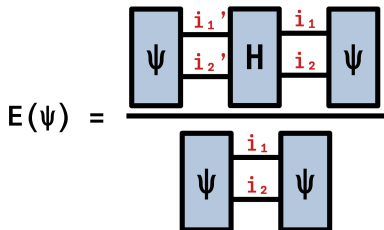
Tutorial: Two-site state optimization

```
1  function E(ψ)
2    ψHψ = inner(ψ', H, ψ)
3    ψψ = inner(ψ, ψ)
4    return ψHψ / ψψ
5  end
```

Tutorial: Two-site state optimization

```
1  function E(ψ)
2    ψHψ = inner(ψ', H, ψ)
3    ψψ = inner(ψ, ψ)
4    return ψHψ / ψψ
5  end
```

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


$$E(\psi) =$$

Simple gradient descent.
Must provide function f(x)
to minimize and ∂f(x),
the gradient of f at x.
  γ is the gradient
descent step size.

Tutorial: Two-site state optimization

```
1  function E(ψ)
2    ψHψ = inner(ψ', H, ψ)
3    ψψ = inner(ψ, ψ)
4    return ψHψ / ψψ
5  end
```

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


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

Tutorial: Two-site state optimization

```
1   ψ₀ = (Zp1 * Zm2 +
2           Zm1 * Zp2)/√2
3
4   E(ψ₀)
5
6   using Zygote: gradient
7   ∂E(ψ) = gradient(E, ψ)[1]
8
9   norm(∂E(ψ₀))
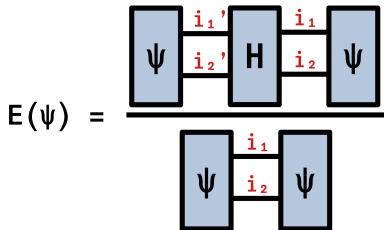```

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

$\approx -1$

Using Zygote for automatic differentiation of the energy.
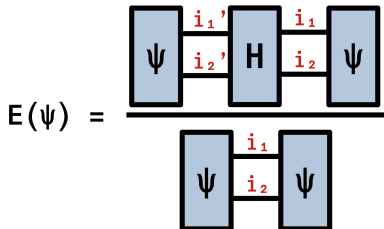
$\approx 2$

# Tutorial: Two-site state optimization

```
1   ψ₀ = (Zp1 * Zm2 +
2          Zm1 * Zp2)/√2
3
4   E(ψ₀)
5
6   using Zygote: gradient
7   ∂E(ψ) = gradient(E, ψ)[1]
8
9   norm(∂E(ψ₀))
```

Tutorial: Two-site state optimization

```
1    ψ₀ = (Zp1 * Zm2 +
2          Zm1 * Zp2)/√2
3
4    E(ψ₀)
5
6    using Zygote: gradient
7    ∂E(ψ) = gradient(E, ψ)[1]
8
9    norm(∂E(ψ₀))
```

```
1    ψ = minimize(E, ∂E, ψ₀;
2          nsteps=10, γ=0.1)
3
4    E(ψ₀), norm(∂E(ψ₀))
5    E(ψ), norm(∂E(ψ))
6
```


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

Minimize over ψ:
$E(\psi) = \langle\psi|H|\psi\rangle/\langle\psi|\psi\rangle$

(-1, 2)
(-1.4142131, 0.0010865277)
$\approx$ (-√2, 0)

Tutorial: Two-site state optimization

```
1   ψ₀ = (Zp1 * Zm2 +
2          Zm1 * Zp2)/√2
3
4   E(ψ₀)
5
6   using Zygote: gradient
7   ∂E(ψ) = gradient(E, ψ)[1]
8
9   norm(∂E(ψ₀))
```

```
1   ψ = minimize(E, ∂E, ψ₀;
2          nsteps=10, γ=0.1)
3
4   E(ψ₀), norm(∂E(ψ₀))
5   E(ψ), norm(∂E(ψ))
6
```


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

Tutorial: Two-site circuit optimization

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

Tutorial: Two-site circuit optimization

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

```
# PastaQ notation:
u(θ, j1, j2) = [
  ("Ry", j1, (; θ=θ[1])),
  ("Ry", j2, (; θ=θ[2])),
  ("CNOT", j1, j2),
  ("Ry", j1, (; θ=θ[3])),
  ("Ry", j2, (; θ=θ[4])),
]
U(θ, i1, i2) =
  buildcircuit(u(θ, 1, 2), [i1, i2])
```
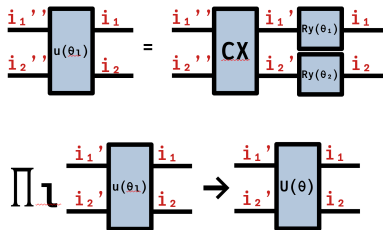
Tutorial: Two-site circuit optimization

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

Tutorial: Two-site circuit optimization

```
1   ψ₀ = Zp1 * Zp2
2
3
4   function E(θ)
5     ψθ = apply(U(θ), ψ₀)
6     return inner(ψθ', H, ψθ)
7   end
```

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

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

Tutorial: Two-site circuit optimization


```
1  ψ₀ = Zp1 * Zp2
2
3
4  function E(θ)
5    ψ_θ = apply(U(θ), ψ₀)
6    return inner(ψ_θ', H, ψ_θ)
7  end
```

Tutorial: Two-site circuit optimization

```
1   ψ₀ = Zp1 * Zp2
2
3
4   function E(θ)
5     ψ_θ = apply(U(θ), ψ₀)
6     return inner(ψ_θ', H, ψ_θ)
7   end
```



```
1   θ₀ = [0, 0, 0, 0]
2   θ = minimize(E, ∂E, θ₀;
3          nsteps=40, γ=0.5)
4
5   E(θ₀), norm(∂E(θ₀))
6   E(θ), norm(∂E(θ))
7
```

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

Tutorial: Two-site circuit optimization

```
1   ψ₀ = Zp1 * Zp2
2
3
4   function E(θ)
5     ψ_θ = apply(U(θ), ψ₀)
6     return inner(ψ_θ', H, ψ_θ)
7   end
```



```
1   θ₀ = [0, 0, 0, 0]
2   θ = minimize(E, ∂E, θ₀;
3         nsteps=40, γ=0.5)
4
5   E(θ₀), norm(∂E(θ₀))
6   E(θ), norm(∂E(θ))
7
```

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

Tutorial: Two-site fidelity optimization

```
1  ψ₀ = Zp1 * Zp2
2  ψ = (ZpZp + ZmZm) / √2
3
4
5
6  function F(θ)
7    ψ_θ = apply(U(θ, i1, i2), ψ₀)
8    return -abs(inner(ψ, ψ_θ))^2
9  end
```

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

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

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

Tutorial: Two-site fidelity optimization

```
1   ψ₀ = Zp1 * Zp2
2   ψ = (ZpZp + ZmZm) / √2
3
4
5
6   function F(θ)
7     ψθ = apply(U(θ, i1, i2), ψ₀)
8     return -abs(inner(ψ, ψθ))^2
9   end
```

Tutorial: Two-site fidelity optimization

```
1  ψ₀ = Zp1 * Zp2
2  ψ = (ZpZp + ZmZm) / √2
3
4
5
6  function F(θ)
7    ψ_θ = apply(U(θ, i1, i2), ψ₀)
8    return -abs(inner(ψ, ψ_θ))^2
9  end
```


$$F(\theta) = -\left\| \; \psi \quad \theta \; \right\|^2$$

```
1  θ₀ = [0, 0, 0, 0]
2  θ = minimize(F, ∂F, θ₀;
3        nsteps=50, γ=0.1)
4
5  F(θ₀), norm(∂F(θ₀))
6  F(θ), norm(∂F(θ))
7
```

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

Tutorial: Two-site fidelity optimization

```
1  ψ₀ = Zp1 * Zp2
2  ψ = (ZpZp + ZmZm) / √2
3
4
5
6  function F(θ)
7    ψ_θ = apply(U(θ, i1, i2), ψ₀)
8    return -abs(inner(ψ, ψ_θ))^2
9  end
```


$$F(\theta) = -\left\| \psi \rule{0pt}{2.5ex}_{i_1}^{\,i_1} \theta \right\|^2$$

```
1  θ₀ = [0, 0, 0, 0]
2  θ = minimize(F, ∂F, θ₀;
3         nsteps=50, γ=0.1)
4
5  F(θ₀), norm(∂F(θ₀))
6  F(θ), norm(∂F(θ))
7
```

$\approx$ (-0.5, 0.5)
$\approx$ (-0.9938992, 0.07786879)
$\approx$ (-1, 0)

Tutorial: n-site states with MPS

```
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    maxlinkdim(Zp)
```

n-site state

$|Z + Z + \ldots Z+\rangle$
$|Z - Z - \ldots Z-\rangle$

1 (product state)

Tutorial: n-site states with MPS

```
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   maxlinkdim(Zp)
```

Tutorial: n-site states with MPS

```
1   ψ = (Zp + Zm)/√2
2
3
4   maxlinkdim(ψ)
```
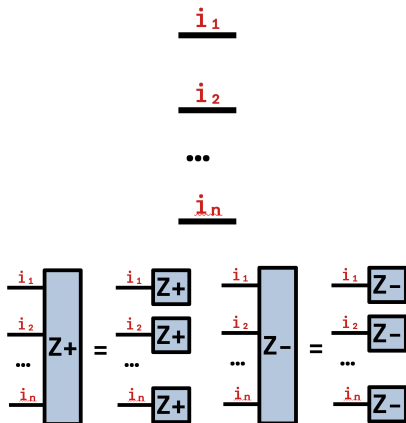
$(|Z + Z + \ldots Z+\rangle +$
$|Z - Z - \ldots Z-\rangle)/\sqrt{2}$

2 (entangled state)

Tutorial: n-site states with MPS

```
1   ψ = (Zp + Zm)/√2
2
3
4   maxlinkdim(ψ)
```

Tutorial: n-site states with MPS

```
1   ψ = (Zp + Zm)/√2
2
3
4   maxlinkdim(ψ)
```



```
1   inner(Zp, Zp)
2   inner(Zm, Zp)
3   norm(ψ)
4   inner(Zp, ψ)
```

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

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

$||\psi\rangle| \approx 1$

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

Tutorial: n-site states with MPS

```
1  ψ = (Zp + Zm)/√2
2
3
4  maxlinkdim(ψ)
```



```
1  inner(Zp, Zp)
2  inner(Zm, Zp)
3  norm(ψ)
4  inner(Zp, ψ)
```
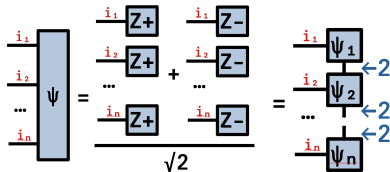
Tutorial: n-site states with MPS

```
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)
```

$j = n/2$
X_j

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

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

Tutorial: n-site states with MPS

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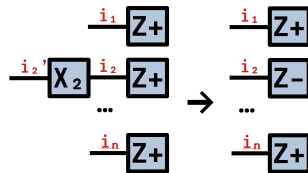
Tutorial: n-site states with MPS

```
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)
```
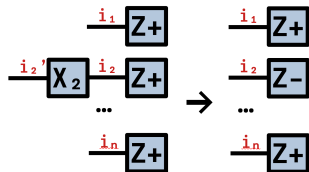


```
1    maxlinkdim(XⱼZp)
2    inner(Zp, XⱼZp)
3    inner(XⱼZp, apply(Xⱼ, Zp))
```

1 (product state)
≈ 0
≈ 1

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
```

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
```
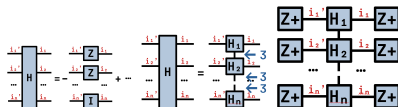
Tutorial: n-site operators with MPO

```
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
```



```
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)
6
```
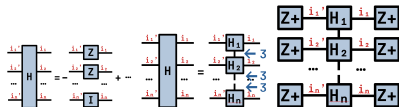
$= 3$ (local Hamiltonian)

$\langle Z+Z+...Z+|H|Z+Z+...Z+\rangle$
$\approx$ -(n - 1) $=$ -29

Tutorial: n-site state optimization

```
1   ψ₀ = MPS(i, "Z+")
2
3   ψ = minimize(E, ∂E, ψ₀;
4       nsteps=50, γ=0.1,
5       maxdim=10, cutoff=1e-5)
6
7   E_dmrg, ψ_dmrg = dmrg(H, ψ₀;
8       nsweeps=10,
9       maxdim=10, cutoff=1e-5)
```

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

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

DMRG solves:
$H|\psi\rangle \approx |\psi\rangle$

Tutorial: n-site state optimization

```
1   ψ₀ = MPS(i, "Z+")
2
3   ψ = minimize(E, ∂E, ψ₀;
4       nsteps=50, γ=0.1,
5       maxdim=10, cutoff=1e-5)
6
7   E_dmrg, ψ_dmrg = dmrg(H, ψ₀;
8       nsweeps=10,
9       maxdim=10, cutoff=1e-5)
```


$$E(\psi) = \frac{\begin{matrix} \psi_1 \overset{i_1'}{-} H_1 \overset{i_1}{-} \psi_1 \\ \psi_2 \overset{i_2'}{-} H_2 \overset{i_2}{-} \psi_2 \\ \cdots \quad \cdots \\ \psi_n \overset{i_n'}{-} H_n \overset{i_n}{-} \psi_n \end{matrix}}{\langle \psi | \psi \rangle}$$

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

Tutorial: n-site state optimization

```
1   ψ₀ = MPS(i, "Z+")
2
3   ψ = minimize(E, ∂E, ψ₀;
4         nsteps=50, γ=0.1,
5         maxdim=10, cutoff=1e-5)
6
7   E_dmrg, ψ_dmrg = dmrg(H, ψ₀;
8         nsweeps=10,
9         maxdim=10, cutoff=1e-5)
```


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

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

```
1   maxlinkdim(ψ₀)
2   maxlinkdim(ψ)
3   maxlinkdim(ψ_dmrg)
4   E(ψ₀), norm(∂(ψ₀))
5   E(ψ), norm(∂E(ψ))
6   E(ψ_dmrg), norm(∂E(ψ_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 circuit optimization*

```
1   Ry_layer(θ, i) = [op("Ry", i[j]; θ=θ[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(θ, i) = [op("Ry", i[j]; θ=θ[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(θ, i; nlayers)
2     n = length(i)
3     U_θ = Ry_layer(θ[1:n], i)
4     for l in 1:(nlayers - 1)
5       θ_l = θ[(1:n) .+ l * n]
6       U_θ = [U_θ; CX_layer(i)]
7       U_θ = [U_θ; Ry_layer(θ_l, i)]
8     end
9     return U_θ
10  end
```

$$U(\theta) = Ry_1(\theta_1) \dots Ry_n(\theta_n)$$
$$* \; CX_{1,2} \dots CX_{n-1,n}$$
$$* \; Ry_1(\theta_{n+1}) \dots Ry_n(\theta_{2n})$$
$$* \; \dots$$

Tutorial: n-site circuit optimization

```
1  Ry_layer(θ, i) = [op("Ry", i[j]; θ=θ[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(θ, i; nlayers)
2     n = length(i)
3     U_θ = Ry_layer(θ[1:n], i)
4     for l in 1:(nlayers - 1)
5        θ_l = θ[(1:n) .+ l * n]
6        U_θ = [U_θ; CX_layer(i)]
7        U_θ = [U_θ; Ry_layer(θ_l, i)]
8     end
9     return U_θ
10  end
```
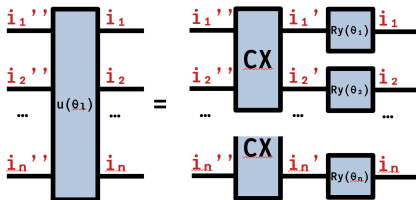

$$\prod_l u(\theta_l) \rightarrow U(\theta)$$

Tutorial: n-site states with MPS

```
1   ψ₀ = MPS(i, "Z+")
2   nlayers = 6
3   function E(θ)
4     ψθ = apply(U(θ, i; nlayers), ψ₀)
5     return inner(ψθ', H, ψθ)
6   end
7
8   θ₀ = zeros(nlayers * n)
9   θ = minimize(E, ∂E, θ₀;
10        nsteps=20, γ=0.1)
```

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

Minimize over θ:
$|\theta\rangle = U(\theta)|0\rangle$
$E(\theta) = \langle\theta|H|\theta\rangle$

Tutorial: n-site states with MPS

```
1   ψ₀ = MPS(i, "Z+")
2   nlayers = 6
3   function E(θ)
4     ψθ = apply(U(θ, i; nlayers), ψ₀)
5     return inner(ψθ', H, ψθ)
6   end
7
8   θ₀ = zeros(nlayers * n)
9   θ = minimize(E, ∂E, θ₀;
10        nsteps=20, γ=0.1)
```

Tutorial: n-site states with MPS

```
1    ψ₀ = MPS(i, "Z+")
2    nlayers = 6
3    function E(θ)
4      ψ_θ = apply(U(θ, i; nlayers), ψ₀)
5      return inner(ψ_θ', H, ψ_θ)
6    end
7
8    θ₀ = zeros(nlayers * n)
9    θ = minimize(E, ∂E, θ₀;
10         nsteps=20, γ=0.1)
```

```
1    maxlinkdim(ψ₀)
2    maxlinkdim(ψ_θ)
3    E(θ₀), norm(∂E(θ₀))
4    E(θ), norm(∂E(θ))
```


1 (product state)
2 (entangled state)
(-29, 5.773335)
(-31.017062, 0.000759)

Tutorial: n-site states with MPS

```
1   ψ₀ = MPS(i, "Z+")
2   nlayers = 6
3   function E(θ)
4     ψθ = apply(U(θ, i; nlayers), ψ₀)
5     return inner(ψθ', H, ψθ)
6   end
7
8   θ₀ = zeros(nlayers * n)
9   θ = minimize(E, ∂E, θ₀;
10        nsteps=20, γ=0.1)
```

```
1   maxlinkdim(ψ₀)
2   maxlinkdim(ψθ)
3   E(θ₀), norm(∂E(θ₀))
4   E(θ), norm(∂E(θ))
```

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!