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

June 18, 2022

FLATIRON INSTITUTE

Center for Computational Quantum Physics

Who am I?

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

mtfishman.github.io

Who am I?

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

mtfishman.github.io

Who am I?

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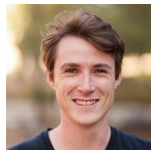- ▶ Thesis on developing tensor network algorithms.

mtfishman.github.io

FLATIRON
INSTITUTE
Center for Computational
Quantum Physics

universität
wien

Who am I?

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

▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).

▶ Thesis on developing tensor network algorithms.

▶ Associate data scientist at CCQ since 2018.

mtfishman.github.io

Who am I?

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

- ▶ Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).

- ▶ Thesis on developing tensor network algorithms.

- ▶ Associate data scientist at CCQ since 2018.

- ▶ Develop ITensor with **Miles Stoudenmire** (CCQ).

mtfishman.github.io

FLATIRON INSTITUTE
Center for Computational Quantum Physics

universität wien

Who am I?

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

- Visited **Frank Verstraete** and **Jutho Haegeman** during my PhD (Vienna, Ghent).

- Thesis on developing tensor network algorithms.

- Associate data scientist at CCQ since 2018.

- Develop ITensor with **Miles Stoudenmire** (CCQ).

- Develop PastaQ with **Giacomo Torlai** (AWS).
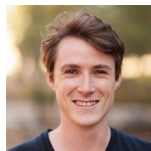
mtfishman.github.io

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

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

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

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

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

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

▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.

▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.

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

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

- ▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.

- ▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.

- ▶ Reach out to us for software needs (new features, bugs), algorithm development, high performance computing, etc.

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

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

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

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

▶ FI has 5 research centers: math, biology, astrophysics, neuroscience, and quantum.

▶ Supports computational research: computer clusters, software development, algorithm development, and scientific applications.

▶ Reach out to us for software needs (new features, bugs), algorithm development, high performance computing, etc.

▶ CCQ has expertise in QMC, quantum embedding (DMFT), tensor networks, quantum dynamics, machine learning, quantum computing, quantum chemistry, etc.

▶ We are hiring postdocs, full-time scientists, part-time and full-time software developers, interns, etc.

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

What is ITensor?

▶ Stands for "Intelligent Tensor".

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.
- **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.
- **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.
- I started working on ITensor in 2018, co-develop with Miles.

What is ITensor?

▶ Stands for "Intelligent Tensor".

▶ Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.

▶ **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.

▶ I started working on ITensor in 2018, co-develop with Miles.

▶ Originally written in C++. I led the port to Julia starting in 2019.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.
- **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.
- I started working on ITensor in 2018, co-develop with Miles.
- Originally written in C++. I led the port to Julia starting in 2019.
- **Katie Hyatt** (AWS) starting writing the GPU backend in Julia at the end of 2019.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.
- **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.
- I started working on ITensor in 2018, co-develop with Miles.
- Originally written in C++. I led the port to Julia starting in 2019.
- **Katie Hyatt** (AWS) starting writing the GPU backend in Julia at the end of 2019.
- Used in over **400** research papers.

What is ITensor?

- Stands for "Intelligent Tensor".
- Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.
- **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.
- I started working on ITensor in 2018, co-develop with Miles.
- Originally written in C++. I led the port to Julia starting in 2019.
- **Katie Hyatt** (AWS) starting writing the GPU backend in Julia at the end of 2019.
- Used in over **400** research papers.
- Website: itensor.org.

What is ITensor?

▶ Stands for "Intelligent Tensor".

▶ Started by **Steve White** (UCI, inventor of the DMRG algorithm) around 2009.

▶ **Miles Stoudenmire** (CCQ) started working on ITensor at the end of 2010.

▶ I started working on ITensor in 2018, co-develop with Miles.

▶ Originally written in C++. I led the port to Julia starting in 2019.

▶ **Katie Hyatt** (AWS) starting writing the GPU backend in Julia at the end of 2019.

▶ Used in over **400** research papers.

▶ Website: itensor.org.

▶ Paper: arxiv.org/abs/2007.14822

Porting to Julia

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

Porting to Julia

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

Porting to Julia

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

Porting to Julia

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

▶ Much less code, easier development.

▶ Now that it is ported, we have many more features: GPU, autodiff, infinite MPS, visualization, etc.

▶ Great numerical libraries, code ended up faster.

DMRG, 2D Hubbard model (U/t = 4)
(Nx, Ny) = (8, 4), 10 sweeps
Hybrid real and momentum space

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 PastaQ?

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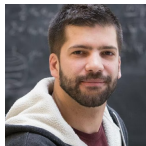
What is PastaQ?

▶ 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.

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

  - ▶ Quantum computing extension to ITensor in Julia.
    - ▶ Tensor network-based quantum state and process tomography.
    - ▶ Extensive and customizable gate definitions.

What is PastaQ?

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

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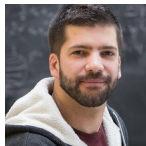
► Initiated by **Giacomo Torlai** (AWS) while he was a postdoc at CCQ, co-developed by Giacomo and me.

► Quantum computing extension to ITensor in Julia.
  ► Tensor network-based quantum state and process tomography.
  ► Extensive and customizable gate definitions.
  ► Built-in and easily extendable circuit definitions.
  ► Noisy circuit evolution with customizable noise models.

What is PastaQ?

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

  - ▶ Quantum computing extension to ITensor in Julia.
    - ▶ Tensor network-based quantum state and process tomography.
    - ▶ Extensive and customizable gate definitions.
    - ▶ Built-in and easily extendable circuit definitions.
    - ▶ Noisy circuit evolution with customizable noise models.
    - ▶ Approximate circuit evolution and optimization with MPS/MPO, etc.

What is PastaQ?

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

  - Quantum computing extension to ITensor in Julia.
    - Tensor network-based quantum state and process tomography.
    - Extensive and customizable gate definitions.
    - Built-in and easily extendable circuit definitions.
    - Noisy circuit evolution with customizable noise models.
    - Approximate circuit evolution and optimization with MPS/MPO, etc.
  - Find out more: github.com/GTorlai/PastaQ.jl

When should I use tensor networks?

► In my opinion, tensor networks are the best general purpose tool we have right now for simulating and analyzing quantum computers.
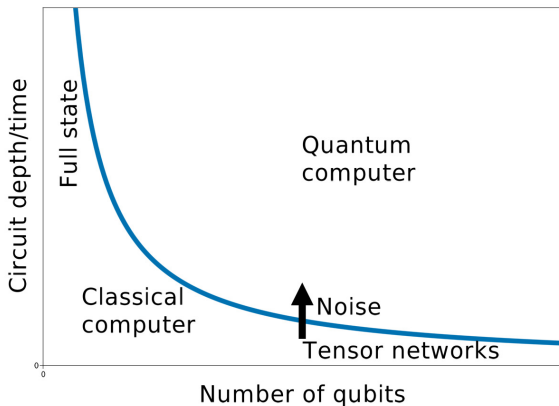
When should I use tensor networks?

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

When should I use tensor networks?

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

When should I use tensor networks?

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

When should I use tensor networks?

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

When should I use tensor networks?

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

When should I use tensor networks?

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

What are tensor networks?

Vector V_i
Order-1 tensor

Matrix M_{ij}
Order-2 tensor

Tensor T_{ijk}
Order-3 tensor

What are tensor networks?

Vector V_i
Order-1 tensor
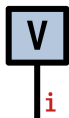
Matrix M_{ij}
Order-2 tensor

Tensor T_{ijk}
Order-3 tensor
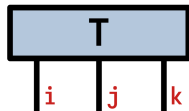
$\langle V | V \rangle = V_i V_i$

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

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

What are tensor networks?

Vector V_i
Order-1 tensor

Matrix M_{ij}
Order-2 tensor

Tensor T_{ijk}
Order-3 tensor

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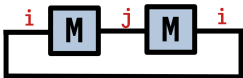
$M|V\rangle = M_{ji}V_i$
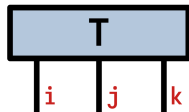
$MM = M_{kj}M_{ji}$

$tr(MM) = M_{ij}M_{ji}$
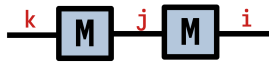
$M_{ij}T_{ijk}V_k$
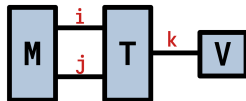
How do I install ITensor/PastaQ?

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);
```

How do I install ITensor/PastaQ?

Now you can use ITensors and PastaQ:

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

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

```julia
using ITensors

i = Index(2)


```

```julia
Zp = ITensor(i)
Zp[i=>1] = 1

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)
```
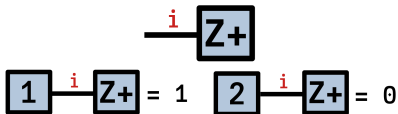
Tutorial: One-site state basics

```
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)
```

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

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

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)
```
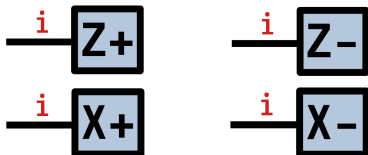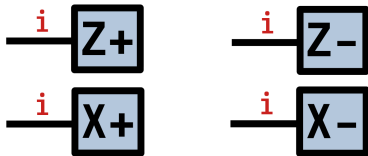


```
1  Xp = (Zp + Zm)/√2
2
3
4
5  (dag(Zp) * Xp)[]
6  inner(Zp, Xp)
```

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

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

Tutorial: One-site state basics

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

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

Tutorial: One-site state basics

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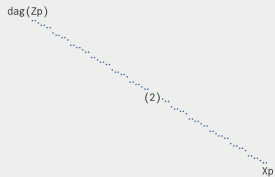
Tutorial: One-site state basics


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



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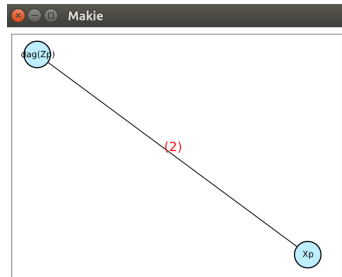
Tutorial: One-site states

```
1    i = Index(2, "S=1/2")
2
3
4
5
6
7
8
9
```

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

"S=1/2" defines an operator basis

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

Tutorial: One-site states

```
1    i = Index(2, "S=1/2")
2
3
4
5
6
7
8
9
```

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

"S=1/2" defines an operator basis

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

```
1  Zp=state("Zp",i)
2  Zm=state("Zm",i)
3  Xp=state("Xp",i)
4  Xm=state("Xm",i)
```

```
1  ITensor([1,0],i)
2  ITensor([0,1],i)
3  ITensor([1,1]/√2,i)
4  ITensor([1,-1]/√2,i)
```

Tutorial: Custom one-site states

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

Overload ITensors.jl behavior

Define a state with the name "iX-"

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

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

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

Tutorial: Custom one-site states

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

Overload ITensors.jl behavior

Define a state with the name "iX-"

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

Tutorial: Priming

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

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

Tutorial: Priming

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

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

Tutorial: Priming

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

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

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

(dim=2|id=837)

(dim=2|id=837)'

Tutorial: Priming

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

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

Tutorial: One-site operators

```
1  i = Index(2, "S=1/2")
2  Z = ITensor(i', i)
3  Z[i'=>1, i=>1] = 1
4  Z[i'=>2, i=>2] = -1
```

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

Tutorial: One-site operators

```
1  i = Index(2, "S=1/2")
2  Z = ITensor(i', i)
3  Z[i'=>1, i=>1] = 1
4  Z[i'=>2, i=>2] = -1
```

Tutorial: One-site operators

```
1  i = Index(2, "S=1/2")
2  Z = ITensor(i', i)
3  Z[i'=>1, i=>1] = 1
4  Z[i'=>2, i=>2] = -1
```

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


Matrix representation Z

Convert to ITensor

Use predefined definition

Tutorial: One-site operators

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

Z
X

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

Tutorial: One-site operators
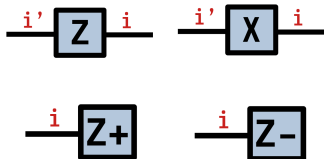
```
1  Z = op("Z", i)
2  X = op("X", i)
3
4  Zp = state("Zp", i)
5  Zm = state("Zm", i)
```

Tutorial: One-site operators

```
1  Z = op("Z", i)
2  X = op("X", i)
3
4  Zp = state("Zp", i)
5  Zm = state("Zm", i)
```
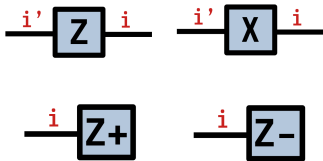
```
1  X * Zp == Zm'
2  noprime(X * Zp) == Zm
```


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

Tutorial: One-site operators

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

```
1  X * Zp == Zm'
2  noprime(X * Zp) == Zm
```
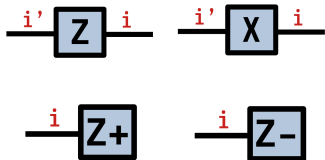
Tutorial: One-site operators

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

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

Tutorial: One-site operators

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

Tutorial: One-site operators

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


$= 1$

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

$X|Z+\rangle$

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

Tutorial: One-site operators

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



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

Tutorial: Custom one-site operators

Overload ITensors.jl
behavior

```julia
import ITensors: op

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

Tutorial: Custom one-site operators

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

```julia
import ITensors: op

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

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



```julia
X * im
```

Tutorial: Two-site states

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

$(\text{dim}=2|\text{id}=505|\text{“S}=1/2\text{”})$
$(\text{dim}=2|\text{id}=576|\text{“S}=1/2\text{”})$

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

Tutorial: Two-site states

```
1  i1 = Index(2, "S=1/2,i1")
2  i2 = Index(2, "S=1/2,i2")
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,i1")
2  i2 = Index(2, "S=1/2,i2")
3
4  i1 != i2
5
6  ZpZm = ITensor(i1, i2)
7  ZpZm[i1=>1, i2=>2] = 1
```
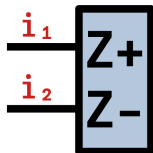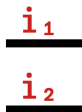
```
1  Zp1=state("Zp",i1)
2  Zp2=state("Zp",i2)
3  Zm1=state("Zm",i1)
4  Zm2=state("Zm",i2)
5
6
7  ZpZm = Zp1 * Zm2
```
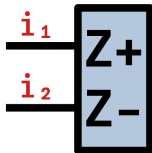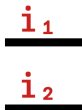

$|Z+\rangle_1$
$|Z+\rangle_2$

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

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

Tutorial: Two-site states

```
1   i1 = Index(2, "S=1/2,i1")
2   i2 = Index(2, "S=1/2,i2")
3
4   i1 != i2
5
6   ZpZm = ITensor(i1, i2)
7   ZpZm[i1=>1, i2=>2] = 1
```
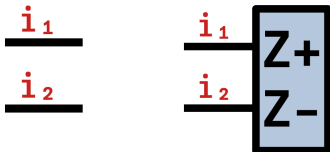
```
1   Zp1=state("Zp",i1)
2   Zp2=state("Zp",i2)
3   Zm1=state("Zm",i1)
4   Zm2=state("Zm",i2)
5
6
7   ZpZm = Zp1 * Zm2
```

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(ZpZm, ψ) == 1/√2
2
3
4  _, S, _ = svd(ZpZm, i1);
5  diag(S) == [1, 0]
6
7  _, S, _ = svd(ψ, i1);
8  diag(S) == [1/√2, 1/√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(ZpZm, ψ) == 1/√2
2
3
4  _, S, _ = svd(ZpZm, i1);
5  diag(S) == [1, 0]
6
7  _, S, _ = svd(ψ, i1);
8  diag(S) == [1/√2, 1/√2]
```

Tutorial: Two-site operators

```
1  H = ITensor(i1', i2',
2               i1, i2)
3  H[i1'=>2, i2'=>1,
4    i1=>2, i2=>1] = -1
5  # …
```

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',
2              i1, i2)
3  H[i1'=>2, i2'=>1,
4    i1=>2, i2=>1] = -1
5  # …
```

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  Id2 = op("Id", i2)
5  Z2 = op("Z", i2)
6  X2 = op("X", i2)
7
8  ZZ = Z1 * Z2
9  XI = X1 * Id2
10 IX = Id1 * X2
11
12 h = 0.5
13 H = -ZZ + h * (XI + IX)
```
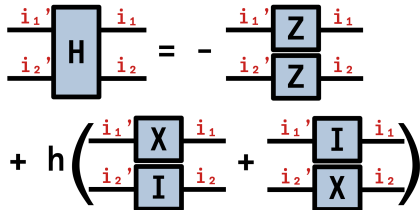
Tutorial: Two-site operators

```
1   inner(Zp1' * Zp2', H,
2         Zp1 * Zp2)
```

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

Tutorial: Two-site operators

```
1  inner(Zp1' * Zp2', H,
2        Zp1 * Zp2)
```


$= -1$
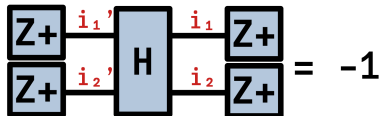
Tutorial: Two-site operators


```
1  D, U = eigen(H);
2  diag(D) ≈ [-√2, -1, 1, √2]
```
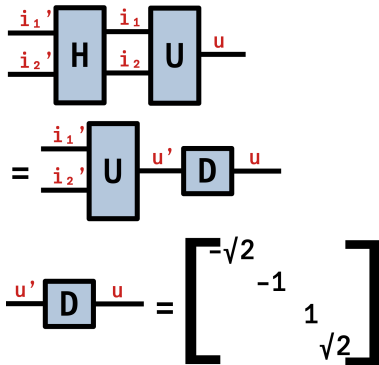
Tutorial: Custom two-site operators

```julia
1   import ITensors: op
2
3   function op(
4     ::OpName"CRy",
5     ::SiteType"S=1/2"; θ)
6     c = cos(θ/2)
7     s = sin(θ/2)
8     return [
9       1 0 0  0
10      0 1 0  0
11      0 0 c -s
12      0 0 s  c]
13  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
```
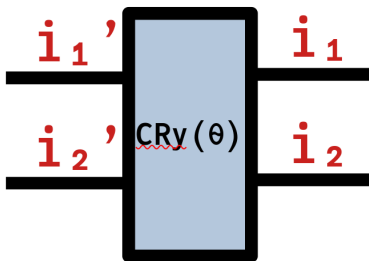
Tutorial: Custom two-site operators

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

Controlled-rotation gate
CRy$(\pi/2)$

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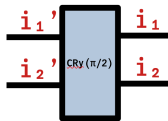
```
1  CRy = op("CRy", i1, i2; θ=π/2)
```

Tutorial: Custom two-site operators

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

```
1   Xp1=state("Xp",i1)
2   Xm1=state("Xm",i1)
3   Xp2=state("Xp",i2)
4   Xm2=state("Xm",i2)
5
6   apply(CRy, Zm1 * Zp2) ≈
7          Zm1 * Xp2
```
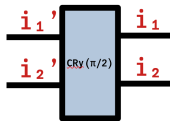

$$\mathrm{CRy}(\pi/2)|\mathrm{Z\text{-}Z+}\rangle = |\mathrm{Z\text{-}X+}\rangle$$

Tutorial: Custom two-site operators

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

```
1  Xp1=state("Xp",i1)
2  Xm1=state("Xm",i1)
3  Xp2=state("Xp",i2)
4  Xm2=state("Xm",i2)
5
6  apply(CRy, Zm1 * Zp2) ≈
7         Zm1 * Xp2
```

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
```

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

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
```


$E(\psi) =$

Gradient descent.

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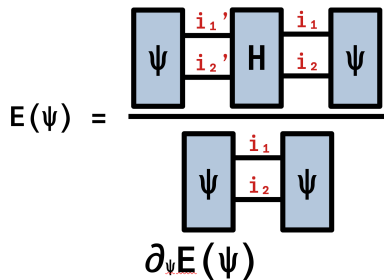
```
1  function minimize(f, ∂f, x;
2                     nsteps, γ)
3    for n in 1:nsteps
4      x = x - γ * ∂f(x)
5    end
6    return x
7  end
```


$$E(\psi) =$$

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

Tutorial: Two-site state optimization

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


$$E(\psi) = \frac{\boxed{\psi} \; \overset{i_1'}{\underset{i_2'}{}} \; \boxed{H} \; \overset{i_1}{\underset{i_2}{}} \; \boxed{\psi}}{\boxed{\psi} \; \overset{i_1}{\underset{i_2}{}} \; \boxed{\psi}}$$

$$\partial_\psi E(\psi)$$

Tutorial: Two-site state optimization

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


$$E(\psi) = \frac{}{}$$

$$\partial_\psi E(\psi)$$

```
1   ψ = minimize(E, ∂E, ψ0;
2                nsteps=10, γ=0.1)
3
4   E(ψ0) == -1
5   E(ψ) # ≈ -1.4142131 ≈ -√2
6
```

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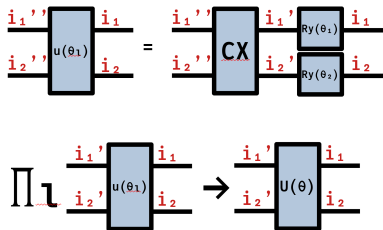$$\min_\psi \langle\psi|H|\psi\rangle / \langle\psi|\psi\rangle$$

Tutorial: Two-site circuit optimization

```
1   # Circuit as a vector of
         gates:
2   U(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
```
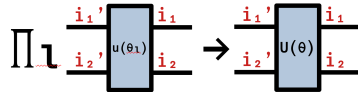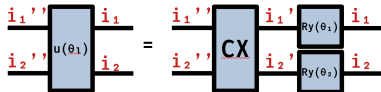
Tutorial: Two-site circuit optimization

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

Tutorial: Two-site circuit optimization

```
1   ψ0 = Zp1 * Zp2
2
3
4   function E(θ)
5       ψ_θ = apply(U(θ, i1, i2), ψ
            0)
6       return inner(ψ_θ', H, ψ_θ)
7   end
```
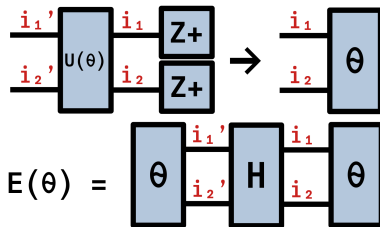
References state:
$$|0\rangle = |Z+Z+\rangle$$

$\min_\theta E(\theta)$
  $= \min_\theta \langle 0|U(\theta)^\dagger \, H \, U(\theta)|0\rangle$
  $= \min_\theta \langle \theta|H|\theta\rangle$

Tutorial: Two-site circuit optimization

```
1   ψ0 = Zp1 * Zp2
2
3
4   function E(θ)
5       ψ_θ = apply(U(θ, i1, i2), ψ
            0)
6       return inner(ψ_θ', H, ψ_θ)
7   end
```
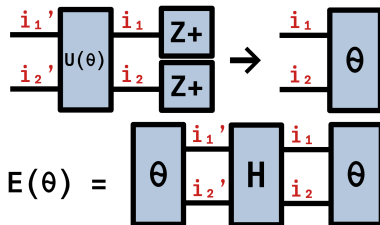
Tutorial: Two-site circuit optimization

```
1   ψ0 = Zp1 * Zp2
2
3
4   function E(θ)
5     ψ_θ = apply(U(θ, i1, i2), ψ0)
6     return inner(ψ_θ', H, ψ_θ)
7   end
```

```
1   θ_0 = [0, 0, 0, 0]
2   ∂E(θ) = gradient(E, θ)[1]
3   θ = minimize(E, ∂E, θ_0;
4           nsteps=40, γ=0.5)
5
6   E(θ_0) == -1
7   E(θ) == -1.4142077 ≈ -√2
```


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

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), ψ
         0)
8    return -abs(inner(ψ, ψθ))
         ^2
9  end
```
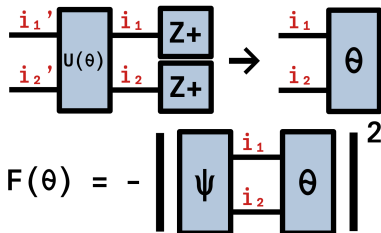
Reference state:
$$|0\rangle = |Z+Z+\rangle$$

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

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

Tutorial: Two-site fidelity optimization
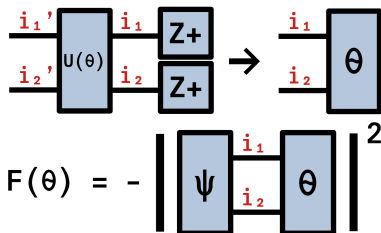
```
1   ψ0 = Zp1 * Zp2
2   ψ = (ZpZp + ZmZm) / √2
3
4
5
6   function F(θ)
7     ψθ = apply(U(θ, i1, i2), ψ
        0)
8     return -abs(inner(ψ, ψθ))
        ^2
9   end
```


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

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), ψ
         0)
8    return -abs(inner(ψ, ψθ))
         ^2
9  end
```

```
1  θ0 = [0, 0, 0, 0]
2  ∂F(θ) = gradient(F, θ)[1]
3  θ = minimize(F, ∂F, θ0;
4        nsteps=50, γ=0.1)
5
6  F(θ0) == -0.5
7  F(θ) == -0.9938992 ≈ -1
```


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

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

Tutorial: n-site states with MPS

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

n-site state

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

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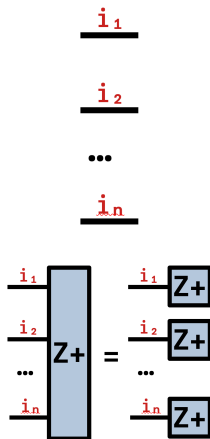
```
1    n = 30
2    i = [Index(2, "S=1/2")
3                    for j
     in 1:n]
4
5
6    Zp = MPS(i, "Zp")
7
8    maxlinkdim(Zp) == 1 #
         product state
```

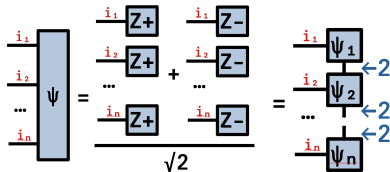# Tutorial: n-site states with MPS

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

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

Tutorial: n-site states with MPS

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

Tutorial: n-site states with MPS

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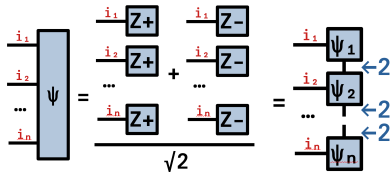


```
1    inner(Zp, Zp) == 1
2
3
4    inner(Zp, ψ) == 1/√2
```

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

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

Tutorial: n-site states with MPS
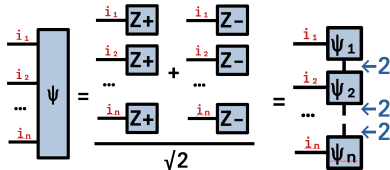

```
1    ψ = (Zp + Zm)/√2
2
3
4    maxlinkdim(ψ) == 2 #
         entangled state
```



```
1    inner(Zp, Zp) == 1
2
3
4    inner(Zp, ψ) == 1/√2
```
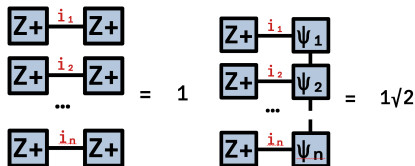
Tutorial: n-site states with MPS

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

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

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

Tutorial: n-site states with MPS

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

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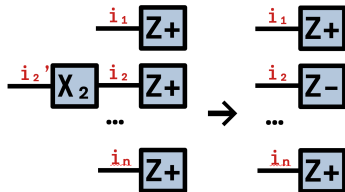
```
1    j = n ÷ 2
2    Xⱼ = op("X", i[j])
3
4    XⱼZp = apply(Xⱼ, Zp)
5
6
7    state = [k == j ? "Zm" : "Zp"
8                              for k
        in 1:n]
9    XⱼZp = MPS(i, state)
```



```
1    maxlinkdim(XⱼZp) == 1
2    inner(Zp, XⱼZp) == 0
```
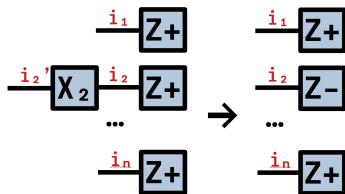
Tutorial: n-site operators with MPO

```julia
function ising(n; h)
  H = OpSum()
  for j in 1:(n - 1)
    H -= "Z", j, "Z", j + 1
  end
  for j in 1:n
    H += h, "X", j
  end
  return H
end

h = 0.5 # field
H = MPO(ising(n; h), i)
maxlinkdim(H) == 3
```

n sites

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

Tutorial: n-site operators with MPO

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

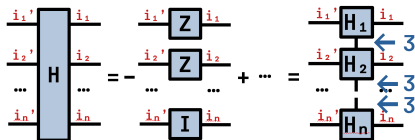# Tutorial: n-site operators with MPO

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



```julia
1  Zp = MPS(i, "Zp")
2  inner(Zp', H, Zp) == -(n-1)
3
```

Tutorial: n-site state optimization

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


$$E(\psi) = \frac{\begin{matrix}\psi_1 - H_1 - \psi_1 \\ \psi_2 - H_2 - \psi_2 \\ \vdots \\ \psi_n - H_n - \psi_n\end{matrix}}{\langle \psi | \psi \rangle}$$

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

Tutorial: n-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  ψ0 = MPS(i, "Zp")
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)
```


$$E(\psi) = \frac{\psi_1 - i_1' - H_1 - i_1 - \psi_1}{\langle \psi | \psi \rangle}$$

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

Tutorial: n-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  ψ0 = MPS(i, "Zp")
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)
```

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

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

```
1  maxlinkdim(ψ0) == 1
2  maxlinkdim(ψ) == 3
3  maxlinkdim(ψ_dmrg) == 3
4  E(ψ0) == -29
5  E(ψ) == -31.0317917
6  E(ψ_dmrg) == -31.0356110
```

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    for l in 1:(nlayers - 1)
4      θₗ = θ[(1:n) .+ l * n]
5      U_θ = [U_θ; Ry_layer(θₗ, i)
            ]
6      U_θ = [U_θ; CX_layer(i)]
7    end
8    return U_θ
9  end
```

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

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      for l in 1:(nlayers - 1)
4          θₗ = θ[(1:n) .+ l * n]
5          U_θ = [U_θ; Ry_layer(θₗ, i)
             ]
6          U_θ = [U_θ; CX_layer(i)]
7      end
8      return U_θ
9  end
```


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

Tutorial: *n*-site states with MPS

```
1   ψ0 = MPS(i, "Zp")
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, "Zp")
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)
```
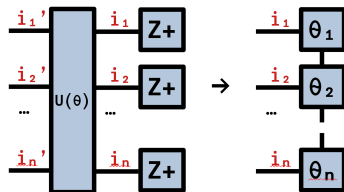
```
1  maxlinkdim(ψ0) = 1
2  maxlinkdim(ψθ) = 3
3  E(θ0) == -29
4  E(θ) == -31.017062
```

Tutorial: n-site states with MPS


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

Tutorial: n-site states with MPS

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

```
1   maxlinkdim(ψ0) == 1
2   maxlinkdim(ψ_θ) == 2
3   F(θ0) == -0.556066
4   F(θ) == -0.995230
```


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

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

Other features not mentioned

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

Other features not mentioned

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

Other features not mentioned

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

▶ Time evolution with trotterization (ITensor/PastaQ).
   AD/optimization and TDVP support in progress.

▶ Quantum state and process tomograph (PastaQ).

Other features not mentioned

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

▶ Time evolution with trotterization (ITensor/PastaQ).
   AD/optimization and TDVP support in progress.

▶ Quantum state and process tomograph (PastaQ).

▶ Predefined differentiable circuits (PastaQ).

Other features not mentioned

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

Other features not mentioned

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

Other features not mentioned

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

Other features not mentioned

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

Future directions/in progress

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

Future directions/in progress

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

- ▶ More general built-in tensor networks beyond MPS/MPO (tree tensor networks and PEPS, general contraction and optimization, use in circuit evolution and tomography, etc.).

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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

Future directions/in progress

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