Title: Tutorial: Coarse-graining of Spin Foams

Date: Jun 22, 2017 03:45 PM

URL: http://pirsa.org/17060087

Abstract:

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

1 (1 von 14) 154%

Tutorial: Symmetries in tensor networks

Sebastian Steinhaus

sebastian.steinhaus@desy.de

II. Institute for Theoretical Physics
University of Hamburg

Making Quantum Gravity computable
@ Perimeter Institute, Waterloo

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Emmy Noether-Programm
DFG

DEU 15:57 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

2 (2 von 14) 154%

Topic of this tutorial

- Tensor network algorithms: computational time and memory grow exponentially with number of variables
  - Singular Value Decomposition
  - Summation over variables
- Use symmetries / properties of theory
  - Non-Abelian: take care of 'magnetic' indices
  - Only compute / sum over non-vanishing components
- Work with smaller tensors
  - 3-valent algorithm vs. 4-valent algorithm

$SU(2)_k \times SU(2)_k$ Barrett-Crane intertwiner and spin net models

- Triangular algorithm
- 'Superindices'

This tutorial is not technical! For details see the following article:
[Dittrich, Schnetter, Seth, Steinhaus PRD '16 (1609.02429)]

Sebastian Steinhaus (UHH) Tut.: Symm. in tensor networks Making QG computable 2 / 14

DEU 15:57 22.06.2017

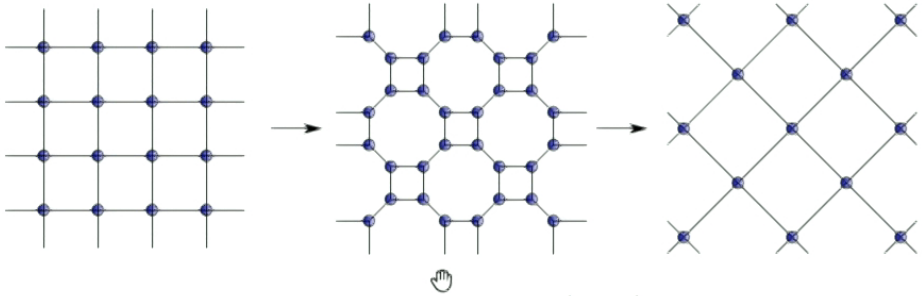
SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

3 (3 von 14) 154%

Setup: $\mathrm{SU}(2)_k \times \mathrm{SU}(2)_k$ intertwiner models



$$T(\{j_i^\pm\},\{m_i^\pm\}) = \begin{array}{ccccc} & & j_4^\pm, m_4^\pm & & \\ j_1^\pm, m_1^\pm & - & \bullet & - & j_3^\pm, m_3^\pm \\ & & j_2^\pm, m_2^\pm & & \end{array}$$

- $\mathrm{SU}(2)_k$ is a quantum group
- $j_i^\pm \in \{0, 1, \ldots, j_{\max}\}$ with $j_{\max} = \frac{k}{2}$
- $m_i \in \{-j_i, -j_i+1, \ldots, j_i-1, j_i\} \to 2j_i + 1$ indices

Summing over all these indices is completely hopeless!

Sebastian Steinhaus (UHH) Tut.: Symm. in tensor networks Making QG computable 3 / 14

DEU 15:59 22.06.2017

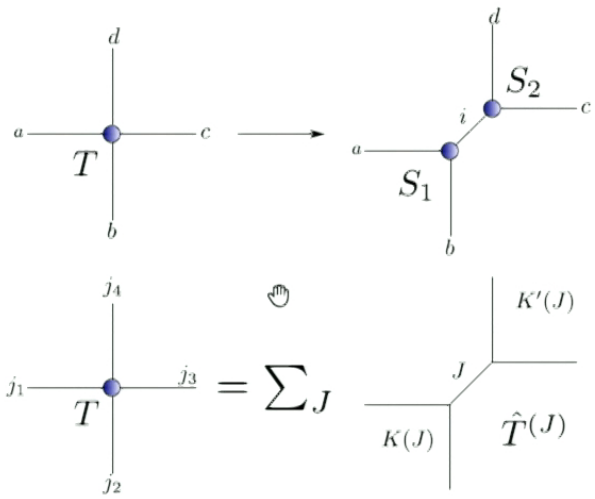
SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

4 (4 von 14) 154%

Recoupling basis: decoupling magnetic indices



- Change of basis adapted to intended splitting
- 'Intermediate' spin J: (j_1, j_2) and (j_3, j_4) couple to J

$$T(\{j_i^{\pm}\}, \{m_i^{\pm}\}) = \sum_{J^{\pm}} \hat{T}^{(J^{\pm})}(\{j_i^{\pm}\}) \quad \times \text{ magnetic indices terms}$$

DEU 16:00 22.06.2017

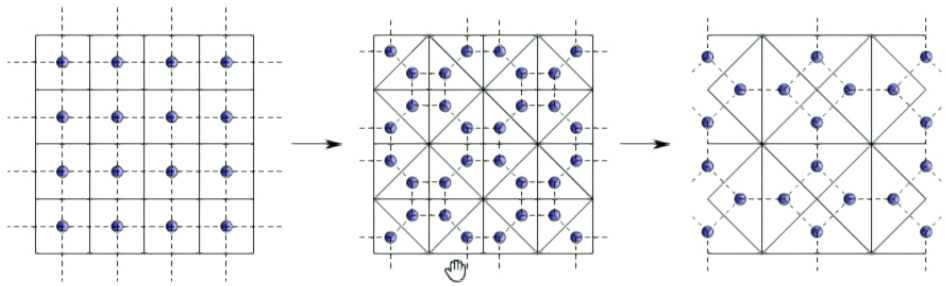
SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

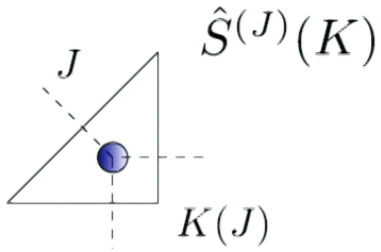
Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

5 (5 von 14) 154%

Triangular algorithm





- Work with 3-valent tensor instead
- $\hat{T}^{(J^\pm)}(j_1, j_2, j_3, j_4) \to \hat{S}^{(J^\pm)}(j_1, j_2)$
- Less memory used:
  - Smaller tensors
  - Compute each 'block' J separately

Indispensable to study $\mathrm{SU}(2)_k \times \mathrm{SU}(2)_k$ spin nets. (More on that later.)

DEU 16:01 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

6 (6 von 14) 154%

Algorithm: basic definitions

- Open file `Intertwiner-no-super` in `IJulia`.
- Beginning: basic quantum group definitions
  - `p` determines maximal spin `k`
  - `p=12` $\rightarrow$ `k=2`
  - `p=16` $\rightarrow$ `k=3`
  - `p=20` $\rightarrow$ `k=4`
- `computeInitial_S(alpha::Float64)` computes initial tensor `S[J1,j1p,j2p,J3,j1m,j2m]`:

$$S_\alpha^{(J^+,J^-)}(j_1^+,j_2^+,j_1^-,j_2^-) = \underbrace{f(\alpha,\{J^\pm\},\{j_i^\pm\})}_{\text{dimension factors}}\,\delta_{J^+,J^-}\delta_{j_1^+,j_1^-}\delta_{j_2^+,j_2^-}$$

This is the Barrett-Crane intertwiner model with parameter α.

Sebastian Steinhaus (UHH) Tut.: Symm. in tensor networks Making QG computable 6 / 14

DEU 16:02 22.06.2017

AppData/Local/Julia-0.5... | Intertwiner-code-triangu... | Intertwiner-code-super-t... | spin-net-code-super-tria... | Quantum Frusta-importa... | Quantum Cuboid - impo...

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

In [1]:
```
const p = 20
# const p = 16
const k = div(p, 4) - 1
const z = (k + 1)^2
const num_q_values = p   # this should suffice
const TOLERANCE = 1.0e-13;


const A = exp(2im * pi / p);


kronecker_delta(num1::Int, num2::Int) = Int(num1 == num2)


minus_one_pow(n::Int) = ifelse(iseven(n), 1, -1)


@inline sqrt_minus_one_pow(n::Int) = ifelse(iseven(n), 1+0im, 0+1im)


function chop{T<:Real}(num::T)
    ifelse(abs(num) >= TOLERANCE, num, T(0))
end


function chop{T<:Complex}(num::T)
    T(chop(real(num)), chop(imag(num)))
end


function q0(n::Int)
    # note this could be simplified!
    An = A^n
    chop(real((An - 1/An) / (A - 1/A)))
end
```

DEU 16:03 22.06.2017

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

```
    return S1
end

function main(alpha::Float64,N::Int64)
    S1 = computeInitial_S(alpha)
    svdmat1 = zeros(k+1,k+1)
    for i in 1:N
        println("Iteration ",i)
        norm = 0.
        S1new = complex(zeros(k+1,k+1,k+1,k+1,k+1,k+1))
        for J1 in 1:k+1, J3 in 1:k+1
            blocks1 = computeBlocks(S1,J1,J3)
            (U,s,V) = computeSvd(blocks1)
            if J1 == 1 && J3 == 1
                norm = s[1]
            end
            S1new[J1,:,:,J3,:,:] = computeNewTensor(J1,J3,U,s,norm)
            svdmat1[J1,J3] = chop(s[1]/norm)
        end
        println(svdmat1)
```

DEU 16:03 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

6 (6 von 14) 154%

Algorithm: basic definitions

- Open file `Intertwiner-no-super` in `IJulia`.
- Beginning: basic quantum group definitions
  - `p` determines maximal spin `k`
  - `p=12` $\rightarrow$ `k=2`
  - `p=16` $\rightarrow$ `k=3`
  - `p=20` $\rightarrow$ `k=4`
- `computeInitial_S(alpha::Float64)` computes initial tensor `S[J1,j1p,j2p,J3,j1m,j2m]`:

$$S_\alpha^{(J^+,J^-)}(j_1^+,j_2^+,j_1^-,j_2^-) = \underbrace{f(\alpha,\{J^\pm\},\{j_i^\pm\})}_{\text{dimension factors}}\,\delta_{J^+,J^-}\delta_{j_1^+,j_1^-}\delta_{j_2^+,j_2^-}$$

This is the Barrett-Crane intertwiner model with parameter α.

Sebastian Steinhaus (UHH) Tut.: Symm. in tensor networks Making QG computable 6 / 14

DEU 16:04 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

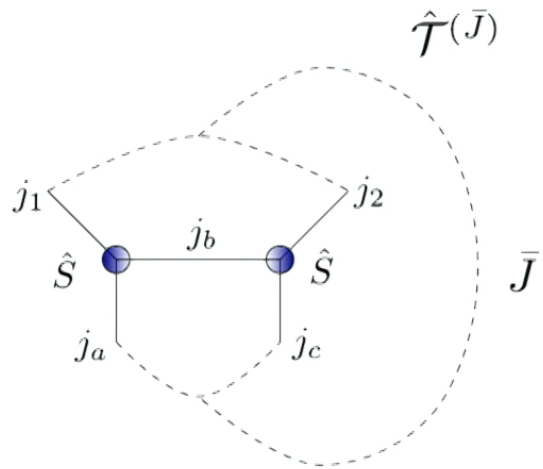
Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

7 (7 von 14) 154%

Algorithm: Intermediate 4-valent tensor

- `computeBlocks(...)` glues two 3-valent tensors together.
  - Sum over j_b / `jb`.
  - Multiply by two $\{6j\}$ symbols to get 'block' `J1,J3`.



- Returns a matrix to which we apply the singular value decomposition.

DEU 16:04 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

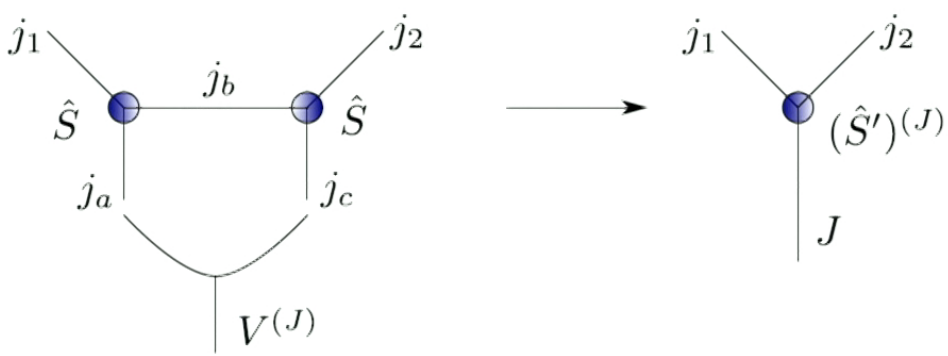
Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

8 (8 von 14) 154%

SVD and new tensor

- `computeSvd(...)` compute the singular value decomposition of said matrix.
  - Return singular vectors `U`, `V` and singular values s.
  - We only keep the largest SV per block `J1,J3`.
- `computeNewTensor(...)` gives 'block' `J1,J3` of renormalized tensor S
  - We apply `V` to the lower legs of 4-valent tensor.

$$\sum_b M_{ab} V_{i,b} = \sum_j U_{a,j}\, s_j\, (V_{b,j})^\dagger\, V_{b,i} = U_{a,i}\, s_i$$



DEU 16:06 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

9 (9 von 14) 154%

Run the code

- `main(N,alpha)` sums up the entire code.
  - `N` number of iterations
  - `alpha` parameter of the model
- Run the code by clicking on `Cell>RunAll`.
- Singular values enough to identify whether a fixed point is reached.
- Play around with the numbers and try also `p=16,20`.

Code takes considerable amount of time!
We compute and sum over many components which vanish due to symmetry!
How can we use the symmetry to optimize the code?

DEU 16:07 22.06.2017

AppData/Local/Julia-0.5 | Intertwiner-code-triangu | Intertwiner-code-super-t | spin-net-code-super-tria | Quantum Frusta-importa | Quantum Cuboid - impo

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

In [1]:
```
const p = 20
#^const p = 16
const k = div(p, 4) - 1
const z = (k + 1)^2
const num_q_values = p   # this should suffice
const TOLERANCE = 1.0e-13;


const A = exp(2im * pi / p);


kronecker_delta(num1::Int, num2::Int) = Int(num1 == num2)


minus_one_pow(n::Int) = ifelse(iseven(n), 1, -1)


@inline sqrt_minus_one_pow(n::Int) = ifelse(iseven(n), 1+0im, 0+1im)


function chop{T<:Real}(num::T)
    ifelse(abs(num) >= TOLERANCE, num, T(0))
end


function chop{T<:Complex}(num::T)
    T(chop(real(num)), chop(imag(num)))
end


function q0(n::Int)
    # note this could be simplified!
    An = A^n
    chop(real((An - 1/An) / (A - 1/A)))
end
```

DEU 16:07 22.06.2017

AppData/Local/Julia-0.5... | Intertwiner-code-triangu... | Intertwiner-code-super-t... | spin-net-code-super-tria... | Quantum Frusta-importa... | Quantum Cuboid - impo...

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

In [2]:
```
@time main(0.75,20)
```

```
Iteration 1
[1.0 0.0 0.118465; 0.0 0.982401 0.0; 0.118465 0.0 1.0]
Iteration 2
[1.0 0.0 0.100201; 0.0 0.998907 0.0; 0.100201 0.0 1.0]
Iteration 3
[1.0 0.0 0.0468542; 0.0 0.999756 0.0; 0.0468542 0.0 1.0]
Iteration 4
[1.0 0.0 0.0201386; 0.0 0.999999 0.0; 0.0201386 0.0 1.0]
Iteration 5
[1.0 0.0 0.00824568; 0.0 1.0 0.0; 0.00824568 0.0 1.0]
Iteration 6
[1.0 0.0 0.00336701; 0.0 1.0 0.0; 0.00336701 0.0 1.0]
Iteration 7
[1.0 0.0 0.00137462; 0.0 1.0 0.0; 0.00137462 0.0 1.0]
Iteration 8
[1.0 0.0 0.000561217; 0.0 1.0 0.0; 0.000561217 0.0 1.0]
Iteration 9
[1.0 0.0 0.000323531; 0.0 1.0 0.0; 0.000323531 0.0 1.0]
Iteration 10
[1.0 0.0 0.000186791; 0.0 1.0 0.0; 0.000186791 0.0 1.0]
Iteration 11
[1.0 0.0 0.000107844; 0.0 1.0 0.0; 0.000107844 0.0 1.0]
Iteration 12
[1.0 0.0 6.22637e-5; 0.0 1.0 0.0; 6.22637e-5 0.0 1.0]
Iteration 13
[1.0 0.0 3.59479e-5; 0.0 1.0 0.0; 3.59479e-5 0.0 1.0]
Iteration 14
[1.0 0.0 2.07546e-5; 0.0 1.0 0.0; 2.07546e-5 0.0 1.0]
Iteration 15
[1.0 0.0 1.19826e-5; 0.0 1.0 0.0; 1.19826e-5 0.0 1.0]
Iteration 16
[1.0 0.0 6.91818e-6; 0.0 1.0 0.0; 6.91818e-6 0.0 1.0]
Iteration 17
[1.0 0.0 3.99422e-6; 0.0 1.0 0.0; 3.99422e-6 0.0 1.0]
Iteration 18
[1.0 0.0 2.30606e-6; 0.0 1.0 0.0; 2.30606e-6 0.0 1.0]
```

DEU 16:08 22.06.2017

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

```
            blocks1 = computeBlocks(S1,J1,J3)
            (U,s,V) = computeSvd(blocks1)
            if J1 == 1 && J3 == 1
                norm = s[1]
            end
            S1new[J1,:,:,J3,:,:] = computeNewTensor(J1,J3,U,s,norm)
            svdmat1[J1,J3] = chop(s[1]/norm)
        end
        println(svdmat1)
        S1 = S1new
    end
end
```

Out[1]: main (generic function with 1 method)

In [*]:
```
@time main(0.75,20)
```

```
Iteration
```

In [ ]:

In [ ]:

DEU 16:09 22.06.2017

AppData/Local/Julia-0.5... | Intertwiner-code-triangu... | Intertwiner-code-super-t... | spin-net-code-super-tria... | Quantum Frusta-importa... | Quantum Cuboid - impo...

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-triangular.ipynb

jupyter Intertwiner-code-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help | Julia 0.5.2

Code | CellToolbar

Out[1]: main (generic function with 1 method)

In [*]:
```
@time main(0.75,20)
```

```
Iteration 1
[1.0 0.05759 0.05759 0.0; 0.05759 0.989746 0.05759 0.05759; 0.05759 0.05759 0.989746 0.05759; 0.0 0.05759 0.05759 1.0]
Iteration 2
[1.0 0.0565594 0.0565594 0.0; 0.0565594 0.999832 0.0564431 0.0565299; 0.0565594 0.0564431 0.999832 0.0565299; 0.0 0.0565299 0.0
565299 0.999995]
Iteration 3
[1.0 0.0279302 0.0279593 0.000996503; 0.0279302 0.999968 0.0279044 0.0279639; 0.0279593 0.0279044 0.999968 0.0279481; 0.0009965
03 0.0279639 0.0279481 1.0]
Iteration 4
[1.0 0.0135784 0.0135549 0.000244651; 0.0135784 1.0 0.0135705 0.0135546; 0.0135549 0.0135705 1.0 0.0135799; 0.000244651 0.01355
46 0.0135799 1.0]
Iteration 5
[1.0 0.00723563 0.00584106 5.74672e-5; 0.00723563 1.0 0.00723553 0.00584107; 0.00584106 0.00723553 1.0 0.00723553; 5.74672e-5
 0.00584107 0.00723553 1.0]
Iteration 6
[1.0 0.00411367 0.00223543 1.34785e-5; 0.00411367 1.0 0.00411367 0.00223543; 0.00223543 0.00411367 1.0 0.00411367; 1.34785e-5
 0.00223543 0.00411367 1.0]
Iteration 7
[1.0 0.00258929 0.000859449 3.45125e-6; 0.00258929 1.0 0.00258929 0.000859449; 0.000859449 0.00258929 1.0 0.00258929; 3.45125e-
6 0.000859449 0.00258929 1.0]
Iteration 8
[1.0 0.00208184 0.00046513 1.50174e-6; 0.00208184 1.0 0.00208184 0.00046513; 0.00046513 0.00208184 1.0 0.00208184; 1.50174e-6
 0.00046513 0.00208184 1.0]
Iteration 9
```

In [ ]:

In [ ]:

DEU 16:10 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

10 (10 von 14) 154%

$SU(2)_k$ coupling rules

- Open `IJulia` notebook `Superindices`
- Spins j_1, j_2 can couple to J if

$$
\begin{aligned}
j_1 + j_2 &\geq J && (1)\\
j_1 + J &\geq j_2 && (2)\\
j_2 + J &\geq j_1 && (3)\\
j_1 + j_2 + J &\in \mathbb{N} && (4)\\
j_1 + j_2 + J &\leq \frac{p}{2} - 2 && (5)
\end{aligned}
$$

Exercise 1: Coupling rules

Write a function of three spins `j1,j2,j` that return `1` if these rules are satisfied and `0` if not.

How can we use this to compute / store / sum over less indices?

DEU 16:11 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

11 (11 von 14) 154%

Superindex 1

- Write a function that counts all allowed combinations (j_1, j_2, J).
- Translation between index and these combinations
- With respect to `J`:
  - Index `K`: index for allowed combinations `j1,j2`
  - Range of `K`: `RangeK[J]`
  - Array `S[J,K,(1,2)]` $\rightarrow$ `(j1,j2)`
  - Array `Kf[J,j1,j2]` $\rightarrow$ `K`
- Return `RangeK,S,KF`

Exercise 2: Superindex `K`

Write a function that computes these indices.

DEU 16:12 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

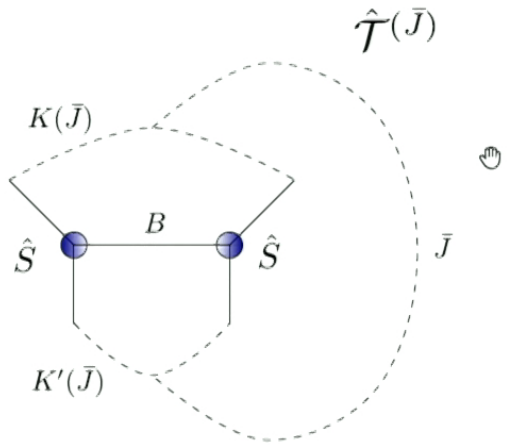
Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

12 (12 von 14) 154%

Superindex 2

- Similarly count allowed $\{6j\}$ symbols (J, K_1, K_2, j)



- With respect to `J,K1,K2`:
  - Index `B`: index for allowed `J,K1,K2,j`
  - Range of `B`: `RangeB[J,K1,K2]`
  - Array `Jf[J,K1,K2,B]` $\rightarrow$ `j`
- Return `RangeB,Jf`.

DEU 16:13 22.06.2017

SF-Tensor-Tutorial.pdf - Adobe Acrobat Reader DC

Datei Bearbeiten Anzeige Fenster Hilfe

Start Werkzeuge SF-Tensor-Tutorial.... Anmelden

13 (13 von 14) 154%

Improved code

Exercise 3: Superindex 2

Write a function returning these indices.

Hint: Look at `Kf[j,S[J,K1,1]]` and `Kf[j,S[J,K2,2]]`.

file:///C|/Users/stein/Documents/Quantum%20Gravity/Talk_Tutorial_tensor/Kf[j,S[J,K2,2]]

- In `Intertwiner-code-super-triangular` you find the code in which these indices are fully implemented.

Exercise 4: Scanning the BC intertwiner model

Run simulations for different `alpha` and `p`.

There are two attractive fixed points. Find the `alpha` for which one starts on a fixed point! (for all `p`)

DEU 16:14 22.06.2017

AppData/Local/Julia-0.5... | Intertwiner-code-triangu... | Intertwiner-code-super-t... | spin-net-code-super-tria... | Quantum Frusta-importa... | Quantum Cuboid - impo...

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-super-triangular.ipynb

jupyter Intertwiner-code-super-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

In [ ]:

```
const p = 12
# const p = 16
const k = div(p, 4) - 1
const z = (k + 1)^2
const num_q_values = p   # this should suffice
const TOLERANCE = 1.0e-13;


const A = exp(2im * pi / p);


kronecker_delta(num1::Int, num2::Int) = Int(num1 == num2)


minus_one_pow(n::Int) = ifelse(iseven(n), 1, -1)


@inline sqrt_minus_one_pow(n::Int) = ifelse(iseven(n), 1+0im, 0+1im)


function chop{T<:Real}(num::T)
    ifelse(abs(num) >= TOLERANCE, num, T(0))
end


function chop{T<:Complex}(num::T)
    T(chop(real(num)), chop(imag(num)))
end


function q0(n::Int)
    # note this could be simplified!
    An = A^n
    chop(real((An - 1/An) / (A - 1/A)))
end
```

DEU 16:14 22.06.2017

AppData/Local/Julia-0.5.. | Intertwiner-code-triangu | Intertwiner-code-super-t | spin-net-code-super-tria | Quantum Frusta-importa | Quantum Cuboid - impo

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/Intertwiner-code-super-triangular.ipynb

jupyter Intertwiner-code-super-triangular Last Checkpoint: 19 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help Julia 0.5.2

Code CellToolbar

```
                norm = s[1]
            end
            S1new[J1,:,J3,:] = computeNewTensor(J1,J3,U,s,norm)
            svdmat1[J1,J3] = chop(s[1]/norm)
        end
        println(svdmat1)
        S1 = S1new
    end
end
```

Out[1]: main (generic function with 1 method)

In [2]:
```
@time main(0.5,20)
```

```
Iteration 1
[1.0 0.0445107 0.0481865 0.0 0.0164339; 0.0445107 0.993169 0.0469326 0.0480998 0.0; 0.0481865 0.0469326 0.98984 0.0469326 0.048
1865; 0.0 0.0480998 0.0469326 0.993169 0.0445107; 0.0164339 0.0 0.0481865 0.0445107 1.0]
Iteration 2
[1.0 0.0363371 0.0520786 0.0 0.00710107; 0.0363371 0.999932 0.0478145 0.0521138 0.0; 0.0520786 0.0478145 0.999893 0.0478145 0.0
520786; 0.0 0.0521138 0.0478145 0.999932 0.0363371; 0.00710107 0.0 0.0520786 0.0363371 1.0]
Iteration 3
[1.0 0.0145025 0.0286928 0.000840208 0.00130728; 0.0145025 0.999987 0.0285363 0.0287079 0.000840208; 0.0286928 0.0285363 0.9999
79 0.0285363 0.0286928; 0.000840208 0.0287079 0.0285363 0.999987 0.0145025; 0.00130728 0.000840208 0.0286928 0.0145025 1.0]
Iteration 4
[1.0 0.00402625 0.017111 0.000253909 0.000301563; 0.00402625 1.0 0.0171049 0.0171108 0.000253909; 0.017111 0.0171049 1.0 0.0171
049 0.017111; 0.000253909 0.0171108 0.0171049 1.0 0.00402625; 0.000301563 0.000253909 0.017111 0.00402625 1.0]
Iteration 5
[1.0 0.00111285 0.0102266 8.79659e-5 0.000113039; 0.00111285 1.0 0.0102261 0.0102265 8.79659e-5; 0.0102266 0.0102261 1.0 0.0102
261 0.0102266; 8.79659e-5 0.0102265 0.0102261 1.0 0.00111285; 0.000113039 8.79659e-5 0.0102266 0.00111285 1.0]
Iteration 6
[1.0 0.000307585 0.00611281 3.1407e-5 4.04772e-5; 0.000307585 1.0 0.00611273 0.0061128 3.1407e-5; 0.00611281 0.00611273 1.0 0.0
```

DEU 16:16 22.06.2017

AppData/Local/Julia-0.5... | Intertwiner-code-triangu... | Intertwiner-code-super-t... | spin-net-code-super-tria... | Quantum Frusta-importa... | Quantum Cuboid - impo...

localhost:8888/notebooks/AppData/Local/Julia-0.5.2/spin-net-code-super-triangular.ipynb

jupyter spin-net-code-super-triangular Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Help | Julia 0.5.2

Code | CellToolbar

In [ ]:

```
const p = 12
# const p = 16
const k = div(p, 4) - 1
const z = (k + 1)^2
const num_q_values = p   # this should suffice
const TOLERANCE = 1.0e-13;


const A = exp(2im * pi / p);


kronecker_delta(num1::Int, num2::Int) = Int(num1 == num2)


minus_one_pow(n::Int) = ifelse(iseven(n), 1, -1)


@inline sqrt_minus_one_pow(n::Int) = ifelse(iseven(n), 1+0im, 0+1im)


function chop{T<:Real}(num::T)
    ifelse(abs(num) >= TOLERANCE, num, T(0))
end


function chop{T<:Complex}(num::T)
    T(chop(real(num)), chop(imag(num)))
end


function q0(n::Int)
    # note this could be simplified!
    An = A^n
    chop(real((An - 1/An) / (A - 1/A)))
end
```

DEU 16:58 22.06.2017