Iterated Quantum Products, Sums, and Infinite Sums

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx
With contributions by Jean-Daniel Bancal

Introduction

This is a tutorial on the use of Quantum *Mathematica* add-on to work with iterated quantum products $\otimes_{\square}^{\square}\square$ and sums $\sum_{\square}^{\square}\square$ of expressions in Dirac notation.

Load the Package

First load the Quantum`Notation` package. Write:
Needs["Quantum`Notation`"];
then press at the same time the keys SHIFT-ENTER to evaluate. *Mathematica* will load the package:

```
Needs["Quantum`Notation`"]
```

```
Quantum`Notation` Version 2.2.0. (July 2010)
A Mathematica package for Quantum calculations in Dirac bra-ket notation
by José Luis Gómez-Muñoz

Execute SetQuantumAliases[] in order to use
  the keyboard to enter quantum objects in Dirac's notation
SetQuantumAliases[] must be executed again in each new
  notebook that is created, only one time per notebook.
```

In order to use the keyboard to enter quantum objects write:
SetQuantumAliases[ ];
then press at the same time the keys SHIFT-ENTER to evaluate. The semicolon prevents *Mathematica* from printing the help message. Remember that SetQuantumAliases[ ] must be evaluated again in each new notebook:

```
SetQuantumAliases[];
```

Iterated Quantum Products $\otimes_{j=1}^{n} a_j$

This is a large quantum product entered in a naive way. Press [ESC]eket[ESC] for each eigen-ket template, and press [ESC]tp[ESC] for each infix tensor product symbol:

$$\left|16_{\hat{4}}\right\rangle \otimes \left|25_{\hat{5}}\right\rangle \otimes \left|36_{\hat{6}}\right\rangle \otimes \left|49_{\hat{7}}\right\rangle \otimes \left|64_{\hat{8}}\right\rangle$$

$$\left|16_{\hat{4}},\ 25_{\hat{5}},\ 36_{\hat{6}},\ 49_{\hat{7}},\ 64_{\hat{8}}\right\rangle$$

There is a more convinient notation for large quantum products. Press the keys:
[ESC]qp[ESC]
then press the key [TAB] several times in order to select the lowest "place-holder" (square). Then press the keys:
j=4[TAB]8[TAB][ESC]eket[ESC][TAB][ESC]po[ESC]
then press the key [TAB] several times to select the remaining leftmost "place-holder" (square). Then press the keys:
j[TAB]2[TAB]j
Finally press at the same time the keys SHIFT-ENTER to evaluate:

$$\bigotimes_{j=4}^{8} \left|(j)^2_{\hat{j}}\right\rangle$$

$$\left|16_{\hat{4}},\ 25_{\hat{5}},\ 36_{\hat{6}},\ 49_{\hat{7}},\ 64_{\hat{8}}\right\rangle$$

The "InputForm" version is the command QuantumProduct, which has the same syntax as the standard *Mathematica* commands Product, Sum and Table:

$$\text{QuantumProduct}\left[\left|(j)^2_{\hat{j}}\right\rangle,\ \{j,\ 4,\ 8\}\right]$$

$$\left|16_{\hat{4}},\ 25_{\hat{5}},\ 36_{\hat{6}},\ 49_{\hat{7}},\ 64_{\hat{8}}\right\rangle$$

An increment different from one can be specified using the "InputForm" version, QuantumProduct. It has the same syntax as standard *Mathematica* commands like Product, Sum and Table:

$$\text{QuantumProduct}\left[\left|(j)^2_{\hat{j}}\right\rangle,\ \{j,\ 4,\ 8,\ 2\}\right]$$

$$\left|16_{\hat{4}},\ 36_{\hat{6}},\ 64_{\hat{8}}\right\rangle$$

If the initial value is ommited, then it is assumed to be one:

$$\bigotimes_{j}^{8} \left|(j)^2_{\hat{j}}\right\rangle$$

$$\left|1_{\hat{1}},\ 4_{\hat{2}},\ 9_{\hat{3}},\ 16_{\hat{4}},\ 25_{\hat{5}},\ 36_{\hat{6}},\ 49_{\hat{7}},\ 64_{\hat{8}}\right\rangle$$

InputForm version giving the same result as above:

$$\text{QuantumProduct}\left[\left|(j)^2_{\hat{j}}\right\rangle,\ \{j,\ 8\}\right]$$

$$\left|1_{\hat{1}},\ 4_{\hat{2}},\ 9_{\hat{3}},\ 16_{\hat{4}},\ 25_{\hat{5}},\ 36_{\hat{6}},\ 49_{\hat{7}},\ 64_{\hat{8}}\right\rangle$$

A specific list of index values can be given:

$$\bigotimes_{j}^{\{10,20,100\}} \left| (j)^2{}_{\hat{j}} \right\rangle$$

$$\left| 100_{\hat{10}}, 400_{\hat{20}}, 10\,000_{\hat{100}} \right\rangle$$

InputForm version giving the same result as above:

$$\text{QuantumProduct}\left[\left| (j)^2{}_{\hat{j}} \right\rangle, \{j, \{10, 20, 100\}\}\right]$$

$$\left| 100_{\hat{10}}, 400_{\hat{20}}, 10\,000_{\hat{100}} \right\rangle$$

This is the quantum product of a quantum function (operator) evaluated at different values of its argument:

```
SetQuantumObject[q];
```

$$\bigotimes_{k=5}^{11} q[2\,k]$$

```
q[10] · q[12] · q[14] · q[16] · q[18] · q[20] · q[22]
```

InputForm version giving the same result as above:

```
SetQuantumObject[q];
QuantumProduct[q[2 k], {k, 5, 11}]
```

```
q[10] · q[12] · q[14] · q[16] · q[18] · q[20] · q[22]
```

Iterated quantum products can be calculated, notice that the upper limit of the inner index depends on the outer index:

```
SetQuantumObject[b];
```

$$\bigotimes_{k=3}^{5} \bigotimes_{j=1}^{k-2} b[k, j]$$

```
b[3, 1] · b[4, 1] · b[4, 2] · b[5, 1] · b[5, 2] · b[5, 3]
```

Here we obtain the same result as above using two nested QuantumProduct:

```
SetQuantumObject[b];
QuantumProduct[QuantumProduct[b[k, j], {j, 1, k - 2}], {k, 3, 5}]
```

```
b[3, 1] · b[4, 1] · b[4, 2] · b[5, 1] · b[5, 2] · b[5, 3]
```

Here we obtain the same result as above using one QuantumProduct with two iterators. Notice the order of the iterators, which follows the conventions of the iterators used in standard *Mathematica* commands like Product, Sum and Table:

```
SetQuantumObject[b];
QuantumProduct[b[k, j], {k, 3, 5}, {j, 1, k - 2}]
```

```
b[3, 1] · b[4, 1] · b[4, 2] · b[5, 1] · b[5, 2] · b[5, 3]
```

Complex products can be generated:

```
SetQuantumObject[b];
```

$$\bigotimes_{i}^{\{x,y\}} \bigotimes_{j}^{3} \bigotimes_{k}^{j+1} b[i, j, k]$$

```
b[x, 1, 1] · b[x, 1, 2] · b[x, 2, 1] · b[x, 2, 2] · b[x, 2, 3] · b[x, 3, 1] ·
 b[x, 3, 2] · b[x, 3, 3] · b[x, 3, 4] · b[y, 1, 1] · b[y, 1, 2] · b[y, 2, 1] ·
 b[y, 2, 2] · b[y, 2, 3] · b[y, 3, 1] · b[y, 3, 2] · b[y, 3, 3] · b[y, 3, 4]
```

We clear the definitions used in the previous examples:

```
SetQuantumScalar[b, q];
Clear[b, q]
```

Finite Sums

You can calculate sums that include Quantum expressions. Press [ESC]si[ESC] for the sigma template. The keys [ESC]qs[ESC] (as in quantum sum) give the same template:

$$\sum_{j=4}^{8} \left| j_{\hat{q}} \right\rangle$$

$$\left| 4_{\hat{q}} \right\rangle + \left| 5_{\hat{q}} \right\rangle + \left| 6_{\hat{q}} \right\rangle + \left| 7_{\hat{q}} \right\rangle + \left| 8_{\hat{q}} \right\rangle$$

The "InputForm" version must be typed using the command "QuantumSum" instead of "Sum":

$$\text{QuantumSum}\left[\left| j_{\hat{q}} \right\rangle, \{j, 4, 8\}\right]$$

$$\left| 4_{\hat{q}} \right\rangle + \left| 5_{\hat{q}} \right\rangle + \left| 6_{\hat{q}} \right\rangle + \left| 7_{\hat{q}} \right\rangle + \left| 8_{\hat{q}} \right\rangle$$

An increment different from one can be specified using the "InputForm" version, QuantumSum. It has the same syntax as standard *Mathematica* commands like Product, Sum and Table:

$$\text{QuantumSum}\left[\left| j_{\hat{q}} \right\rangle, \{j, 4, 8, 2\}\right]$$

$$\left| 4_{\hat{q}} \right\rangle + \left| 6_{\hat{q}} \right\rangle + \left| 8_{\hat{q}} \right\rangle$$

Iterated sums:

$$\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{i=1}^{4} |i_{\hat{s}}\rangle\cdot\langle(j+k)_{\hat{s}}|$$

$$|1_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |2_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |3_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |4_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + |1_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |2_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |3_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |4_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}|$$

And the TraditionalForm of the expression can be easier to visualize:

$$\text{TraditionalForm}\left[\sum_{k=1}^{2}\sum_{j=1}^{3}\sum_{i=1}^{4} |i_{\hat{s}}\rangle\cdot\langle(j+k)_{\hat{s}}|\right]$$

$$|1_{\hat{s}}\rangle\langle 2_{\hat{s}}| + |2_{\hat{s}}\rangle\langle 2_{\hat{s}}| + |3_{\hat{s}}\rangle\langle 2_{\hat{s}}| + |4_{\hat{s}}\rangle\langle 2_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\langle 3_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\langle 3_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\langle 3_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\langle 3_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\langle 4_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\langle 4_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\langle 4_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\langle 4_{\hat{s}}| + |1_{\hat{s}}\rangle\langle 5_{\hat{s}}| + |2_{\hat{s}}\rangle\langle 5_{\hat{s}}| + |3_{\hat{s}}\rangle\langle 5_{\hat{s}}| + |4_{\hat{s}}\rangle\langle 5_{\hat{s}}|$$

The "InputForm" version must be typed using the command "QuantumSum" instead of "Sum":

$$\text{QuantumSum}\left[|i_{\hat{s}}\rangle\cdot\langle(j+k)_{\hat{s}}|,\ \{k, 1, 2\},\ \{j, 1, 3\},\ \{i, 1, 4\}\right]$$

$$|1_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |2_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |3_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |4_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + 2\,|1_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|2_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|3_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + 2\,|4_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}| + |1_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |2_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |3_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}| + |4_{\hat{s}}\rangle\cdot\langle 5_{\hat{s}}|$$

Here is an example where the expressions depends on only one sumation index and the lower and upper limits are symbolic. It is slow, it takes several seconds to evaluate:

$$\sum_{k=a}^{p}\sum_{j=d}^{n}\sum_{i=c}^{m} |j_{\hat{s}}\rangle\cdot\langle(j+1)_{\hat{s}}|$$

$$(1-c+m)\,(1-a+p)\sum_{j=d}^{n} |j_{\hat{s}}\rangle\cdot\langle(1+j)_{\hat{s}}|$$

If the initial value of an index is ommited, then it is assumed to be equal to one:

$$\sum_{j}^{3} |j_{\hat{s}}\rangle\cdot\langle(j+1)_{\hat{s}}|$$

$$|1_{\hat{s}}\rangle\cdot\langle 2_{\hat{s}}| + |2_{\hat{s}}\rangle\cdot\langle 3_{\hat{s}}| + |3_{\hat{s}}\rangle\cdot\langle 4_{\hat{s}}|$$

A list of explicit values can be given for the index

$$\sum_{j}^{\{50,100,200\}} |j_{\hat{s}}\rangle \cdot \langle (j+1)_{\hat{s}}|$$

$$|50_{\hat{s}}\rangle \cdot \langle 51_{\hat{s}}| + |100_{\hat{s}}\rangle \cdot \langle 101_{\hat{s}}| + |200_{\hat{s}}\rangle \cdot \langle 201_{\hat{s}}|$$

It is possible to operate with sums. The piecewise answer to this calculation depends on the value of n

$$\langle 7_{\hat{q}}| \cdot \sum_{i=1}^{n} \text{f[i]}\ |i_{\hat{q}}\rangle$$

$$\begin{cases} \text{f[7]} & n > 6 \\ 0 & \text{True} \end{cases}$$

You can use the standard *Mathematica* notation for assumptions:

$$\text{Simplify}\left[\langle 7_{\hat{q}}| \cdot \sum_{i=1}^{n} \text{f[i]}\ |i_{\hat{q}}\rangle, n > 10\right]$$

```
f[7]
```

The Assuming command is another way to specify assumptions. Remember that Assuming only works on those *Mathematica* commands that have the option Assumptions (commands like Simplify, FullSimplify, Integrate, etc, but other commands just ignorate the Assuming command).

$$\text{Assuming}\left[n > 10, \text{Simplify}\left[\langle 7_{\hat{q}}| \cdot \sum_{i=1}^{n} \text{f[i]}\ |i_{\hat{q}}\rangle\right]\right]$$

```
f[7]
```

Known Issue: Operations with Several Nested Sums are Very Slow

When complex symbolic calculations are involved (complex for a computer is not the same as complex for a human), the calculation can be **very slow**. Next calculation can take **several minutes** in your computer. Notice that the first and third sums are actually identity operators, therefore the answer is equivalent to the second sum:

$$\left(\sum_{i=1}^{d} |i_{\hat{a}}\rangle \cdot \langle i_{\hat{a}}|\right) \cdot \left(\sum_{j=1}^{d}\sum_{k=1}^{d} \text{foo[j, k]} * |j_{\hat{a}}\rangle \cdot \langle k_{\hat{a}}|\right) \cdot \left(\sum_{m=1}^{d} |m_{\hat{a}}\rangle \cdot \langle m_{\hat{a}}|\right)$$

$$\sum_{m=1}^{d}\left(\sum_{j=1}^{d} \text{foo[j, m]}\ |j_{\hat{a}}\rangle \cdot \langle m_{\hat{a}}|\right)$$

Infinite Sums

Infinite sums can be used:

$$\left(\sum_{j=-\infty}^{\infty} \left|\, (3\, j)_{\hat{q}} \right\rangle \cdot \left\langle j_{\hat{q}} \right|\right) \cdot \left(\alpha \left|\, 10_{\hat{q}} \right\rangle + \beta \left|\, 80_{\hat{q}} \right\rangle\right)$$

$$\alpha \left|\, 30_{\hat{q}} \right\rangle + \beta \left|\, 240_{\hat{q}} \right\rangle$$

The dummy index m in the first sum and the dummy index m in the second sum are treated as different symbols. Notice the dot [ESC]on[ESC] between the sums, which means that the sums are **not** being multiplied, there **is** a "noncommutative quantum application" between them, each sum is consider as an operator:

$$\left(\sum_{m=-\infty}^{\infty} \left|\, (3\, m)_{\hat{q}} \right\rangle \cdot \left\langle m_{\hat{q}} \right|\right) \cdot \left(\sum_{m=-\infty}^{\infty} \left|\, (5\, m)_{\hat{q}} \right\rangle \cdot \left\langle m_{\hat{q}} \right|\right) \cdot \left|\, 10_{\hat{q}} \right\rangle$$

$$\left|\, 150_{\hat{q}} \right\rangle$$

The command DefineOperatorOnKets[ ] can be used instead of $\sum_{\square}^{\square} \square$ for creating faster operators

We can use a doubly-infinite sum to define a rising-index operator

$$\text{dirac1} = \sum_{k=-\infty}^{\infty} \left|\, (k+1)_{\hat{a}} \right\rangle \cdot \left\langle k_{\hat{a}} \right|$$

$$\sum_{k=-\infty}^{\infty} \left|\, (k+1)_{\hat{a}} \right\rangle \cdot \left\langle k_{\hat{a}} \right|$$

Now dirac1 is an operator

$$\text{dirac1} \cdot \left|\, 7_{\hat{a}} \right\rangle$$

$$\left|\, 8_{\hat{a}} \right\rangle$$

Powers of this operator give the expected answer

$$\text{dirac1}^3 \cdot \left|\, 7_{\hat{a}} \right\rangle$$

$$\left|\, 10_{\hat{a}} \right\rangle$$

The hermitian conjugate of the operator also works.

$$(\text{dirac1})^{\dagger} \cdot \left|\, 7_{\hat{a}} \right\rangle$$

$$\left|\, 6_{\hat{a}} \right\rangle$$

It is possible to define new operators this other way. Notice that we must define the action of both the operator and its hermitian conjugate:

```
DefineOperatorOnKets[ope2, {| x_m̂_⟩ ⧴ | (x + 1)m̂⟩}];
DefineOperatorOnKets[(ope2)†, {| x_m̂_⟩ ⧴ | (x - 1)m̂⟩}];
```

Now ope2 works the same as dirac1

```
ope2 · | 7â⟩
```

```
| 8â⟩
```

Powers of this operator also give the expected answer

```
ope2³ · | 7â⟩
```

```
| 10â⟩
```

The hermitian conjugate of the operator also works. Notice that the action of both the operator and its hermitian conjugate was defined above with two separated DefineOperatorOnKets[]

```
(ope2)† · | 7â⟩
```

```
| 6â⟩
```

The operators defined with DefineOperatorOnKets are usually faster than operators defined with sums of Dirac expressions.

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx