Basis, Matrix, ExpressionFor

```
In[ ]:= << Q3`
```

Basis

```
In[ ]:= Let[Qubit, S]
```

```
In[ ]:= bs = Basis[{S[1, $], S[2, $]}]
```

Out[ ]=

$\{|0_{S_1}0_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle, |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle\}$

Shorthand expressions

```
In[ ]:= Basis@{S[1, $], S[2, $]}
```

Out[ ]=

$\{|0_{S_1}0_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle, |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle\}$

```
In[ ]:= Basis@S[{1, 2}, $]
```

Out[ ]=

$\{|0_{S_1}0_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle, |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle\}$

```
In[ ]:= Basis@S@{1, 2}
```

Out[ ]=

$\{|0_{S_1}0_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle, |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle\}$

```
In[ ]:= new = Basis@S@{2, 1}
```

Out[ ]=

$\{|0_{S_1}0_{S_2}\rangle, |1_{S_1}0_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle\}$

```
In[ ]:= tbl = Transpose@{bb = Row /@ Tuples[{0, 1}, 2], bs, bb, new} // KetRegulate;
        TableForm[tbl,
         TableHeadings → {Range[4], {Row@S[{1, 2}, $], "", Row@S[{2, 1}, $], ""}}]
```

Out[ ]//TableForm=

	$S_1\ S_2$		$S_2\ S_1$	
1	00	$\vert 0_{S_1}0_{S_2}\rangle$	00	$\vert 0_{S_1}0_{S_2}\rangle$
2	01	$\vert 0_{S_1}1_{S_2}\rangle$	01	$\vert 1_{S_1}0_{S_2}\rangle$
3	10	$\vert 1_{S_1}0_{S_2}\rangle$	10	$\vert 0_{S_1}1_{S_2}\rangle$
4	11	$\vert 1_{S_1}1_{S_2}\rangle$	11	$\vert 1_{S_1}1_{S_2}\rangle$

```
In[ ]:= Let[Boson, c]
```

In[ ]:=
```
bs = Basis[c, S@{1, 2}]
```
Out[ ]=

$\{|0_c 0_{S_1} 0_{S_2}\rangle, |0_c 0_{S_1} 1_{S_2}\rangle, |0_c 1_{S_1} 0_{S_2}\rangle, |0_c 1_{S_1} 1_{S_2}\rangle, |1_c 0_{S_1} 0_{S_2}\rangle, |1_c 0_{S_1} 1_{S_2}\rangle, |1_c 1_{S_1} 0_{S_2}\rangle, |1_c 1_{S_1} 1_{S_2}\rangle, |2_c 0_{S_1} 0_{S_2}\rangle, |2_c 0_{S_1} 1_{S_2}\rangle, |2_c 1_{S_1} 0_{S_2}\rangle, |2_c 1_{S_1} 1_{S_2}\rangle, |3_c 0_{S_1} 0_{S_2}\rangle, |3_c 0_{S_1} 1_{S_2}\rangle, |3_c 1_{S_1} 0_{S_2}\rangle, |3_c 1_{S_1} 1_{S_2}\rangle, |4_c 0_{S_1} 0_{S_2}\rangle, |4_c 0_{S_1} 1_{S_2}\rangle, |4_c 1_{S_1} 0_{S_2}\rangle, |4_c 1_{S_1} 1_{S_2}\rangle, |5_c 0_{S_1} 0_{S_2}\rangle, |5_c 0_{S_1} 1_{S_2}\rangle, |5_c 1_{S_1} 0_{S_2}\rangle, |5_c 1_{S_1} 1_{S_2}\rangle\}$

Matrix vs ExpressionFor

In[ ]:=
```
Let[Qubit, S]
```

Matrix Representation of Vectors

Consider a typical initialization state.

In[ ]:=
```
in = Ket[S@{1, 2} → 0]
```
Out[ ]=

$|0_{S_1} 0_{S_2}\rangle$

Construct a superposition state by applying the Hadamard operator on each qubit.

In[ ]:=
```
v = S[1, 6] ** S[2, 6] ** in
```
Out[ ]=

$\frac{1}{2}|0_{S_1} 0_{S_2}\rangle + \frac{1}{2}|0_{S_1} 1_{S_2}\rangle + \frac{1}{2}|1_{S_1} 0_{S_2}\rangle + \frac{1}{2}|1_{S_1} 1_{S_2}\rangle$

Represent the state vector in a matrix form.

In[ ]:=
```
col = Matrix[v, S[{1, 2}, $]]
```
Out[ ]=

SparseArray[Specified elements: 4 Dimensions: {4}]

In[ ]:=
```
col // MatrixForm
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

Shorthand expressions

In[ ]:=
```
Matrix[v, S[{1, 2}]] // Normal
```
Out[ ]=

$\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$

In[∘]:=
```
Matrix[v, S@{1, 2}] // Normal
```
Out[∘]=

$$\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}$$

Convert it back to the state expression form.

In[∘]:=
```
new = ExpressionFor[col, S[{1, 2}, $]]
```
Out[∘]=

$$\frac{1}{2} \left|0_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|0_{S_1} 1_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 1_{S_2}\right\rangle$$

In[∘]:=
```
new - v
```
Out[∘]=

0

Shorthand expressions

In[∘]:=
```
ExpressionFor[col, S[{1, 2}]]
```
Out[∘]=

$$\frac{1}{2} \left|0_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|0_{S_1} 1_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 1_{S_2}\right\rangle$$

In[∘]:=
```
ExpressionFor[col, S@{1, 2}]
```
Out[∘]=

$$\frac{1}{2} \left|0_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|0_{S_1} 1_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 0_{S_2}\right\rangle + \frac{1}{2} \left|1_{S_1} 1_{S_2}\right\rangle$$

Matrix Representation of Operators

Consider a typical two-qubit Hamiltonian. For your information, this is the famous Heisenberg model of interacting spins.

In[∘]:=
```
H = Total[S[1, All] ** S[2, All]]
```
Out[∘]=

$$S_1^x S_2^x + S_1^y S_2^y + S_1^z S_2^z$$

Represent it in a matrix form.

In[∘]:=
```
mat = Matrix[H, S@{1, 2}]
```
Out[∘]=

SparseArray[Specified elements: 8 Dimensions: {4, 4}]

In[∘]:=
```
MatrixForm[mat]
```
Out[∘]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Convert it back to an operator expression.

```
In[ ]:= op = ExpressionFor[mat, S@{1, 2}]
```

Out[ ]=

$S_1^z\, S_2^z + 2\ S_1^+\, S_2^- + 2\ S_1^-\, S_2^+$

```
In[ ]:= PauliForm[op]
```

Out[ ]=

$Z \otimes Z + 2\ X^- \otimes X^+ + 2\ X^+ \otimes X^-$

```
In[ ]:= new = Elaborate[op]
```

Out[ ]=

$S_1^x\, S_2^x + S_1^y\, S_2^y + S_1^z\, S_2^z$

```
In[ ]:= PauliForm[new]
```

Out[ ]=

$X \otimes X + Y \otimes Y + Z \otimes Z$

Application: Eigenvalue Problem

Suppose you want to calculate the eigenvalues and corresponding eigenvectors of the following operator.

```
In[ ]:= H = Total[S[1, All] ** S[2, All]]
```

Out[ ]=

$S_1^x\, S_2^x + S_1^y\, S_2^y + S_1^z\, S_2^z$

First, calculate the matrix representation of the operator.

```
In[ ]:= mat = Matrix[H, S@{1, 2}];
        mat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calculate the eigenvalues.

```
In[ ]:= Eigenvalues[mat]
```

Out[ ]=

```
{-3, 1, 1, 1}
```

Calculate the corresponding eigenvectors.

```
In[ ]:= vv = Eigenvectors[mat]
```

Out[ ]=

```
{{0, -1, 1, 0}, {0, 0, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 0}}
```

In the result above, there are four eigenvectors that are represented in the column-vector form.

Convert the eigenvectors into the state expression form.

In[ ]:=
```
vec = Map[ExpressionFor[#, S@{1, 2}] &, vv]
```
Out[ ]=

$\{-|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |0_{S_1}0_{S_2}\rangle\}$

In[ ]:=
```
ExpressionFor[#, S@{1, 2}] & /@ vv
```
Out[ ]=

$\{-|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |0_{S_1}0_{S_2}\rangle\}$

You can get the eigenvalues and corresponding eigenvectors at once.

In[ ]:=
```
Eigensystem[mat]
```
Out[ ]=
```
{{-3, 1, 1, 1}, {{0, -1, 1, 0}, {0, 0, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 0}}}
```

In[ ]:=
```
{val, vec} = Eigensystem[mat]
```
Out[ ]=
```
{{-3, 1, 1, 1}, {{0, -1, 1, 0}, {0, 0, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 0}}}
```

Check the eigenvalues.

In[ ]:=
```
val
```
Out[ ]=
```
{-3, 1, 1, 1}
```

Check the eigenvectors.

In[ ]:=
```
vec
```
Out[ ]=
```
{{0, -1, 1, 0}, {0, 0, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 0}}
```

Converting an operator to matrix and then back to state expression is boring.

In[ ]:=
```
H = Total[S[1, All] ** S[2, All]]
```
Out[ ]=

$S_1^x S_2^x + S_1^y S_2^y + S_1^z S_2^z$

In[ ]:=
```
ProperValues[H]
```
Out[ ]=
```
{-3, 1, 1, 1}
```

In[ ]:=
```
ProperStates[H]
```
Out[ ]=

$\{-|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |0_{S_1}0_{S_2}\rangle\}$

In[ ]:=
```
{val, vec} = ProperSystem[H]
```
Out[ ]=

$\{\{-3, 1, 1, 1\}, \{-|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |0_{S_1}0_{S_2}\rangle\}\}$

In[ ]:= val

Out[ ]=

$\{-3, 1, 1, 1\}$

In[ ]:= vec

Out[ ]=

$\{-|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |1_{S_1}1_{S_2}\rangle, |0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle, |0_{S_1}0_{S_2}\rangle\}$

Summary

Functions

- **Basis**
- **Matrix**
- **ExpressionFor**
- **Eigenvalues**, **Eigenvectors**, **Eigensystem**
- **ProperValues**, **ProperStates**, **ProperSystem**

Related Links

- S. Wolfram (2017), “An Elementary Introduction to Wolfram Language,” 2nd edition (2017).
- The Wolfram Language: Fast Introduction for Math Students
- The Wolfram Language: Fast Introduction for Programmers