Import sympy and the FermionicSpace class.

In [1]:
```
import sympy as sp
from sympy.physics.quantum.dagger import Dagger
from identical.spaces import FermionicSpace
```

Enable pretty printing. Requires MathJax to display the latex output inline in a browser.

In [2]:
```
sp.init_printing(use_latex=True, wrap_line=True)
```

Define an instance of the FermionicSpace class. The argument specifies the number of fermionic modes in the space. Let us use 3 fermions as an example.

In [4]:
```
h = FermionicSpace(3)
```

Get the (rather trivial) basis for this fermionic space as a list of column vectors.

In [5]:
```
h.basis
```

Out[5]:

$$\left[\left[\begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix}\right], \left[\begin{matrix}0\\0\\0\\0\\0\\0\\1\\0\end{matrix}\right], \left[\begin{matrix}0\\0\\0\\0\\0\\1\\0\\0\end{matrix}\right], \left[\begin{matrix}0\\0\\0\\0\\1\\0\\0\\0\end{matrix}\right], \left[\begin{matrix}0\\0\\0\\1\\0\\0\\0\\0\end{matrix}\right], \left[\begin{matrix}0\\0\\1\\0\\0\\0\\0\\0\end{matrix}\right], \left[\begin{matrix}0\\1\\0\\0\\0\\0\\0\\0\end{matrix}\right], \left[\begin{matrix}1\\0\\0\\0\\0\\0\\0\\0\end{matrix}\right]\right]$$

with the first (leftmost) state corresponding to the vacuum, and the last (rightmost) state to all 3 fermions occupied.

There exists a 1-1 relationship (modulo phases) between these basis states and the occupation (Fock) states. Get the Fock basis, in the same order as h.basis, as a list of column vectors.

In [6]:
```
h.fock_basis
```

Out[6]:
$$\left[\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}\right]$$

Get the creation operators (c_i^) as a list of matrices in the original h.basis

In [7]:
```
h.C
```

Out[7]:
$$\left[\begin{bmatrix}0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\end{bmatrix}, \begin{bmatrix}0&0&1&0&0&0&0&0\\0&0&0&-1&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&-1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}, \begin{bmatrix}0&0&0&0&1&0&0&0\\0&0&0&0&0&-1&0&0\\0&0&0&0&0&0&-1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix}\right]$$

And the annihilation operators

In [8]:
```
h.A
```

Out[8]:
$$\left[\begin{bmatrix}0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\end{bmatrix}, \begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&-1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&-1&0&0\end{bmatrix}, \begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&-1&0&0&0&0&0&0\\0&0&-1&0&0&0&0&0\\0&0&0&1&0&0&0&0\end{bmatrix}\right]$$

Create a fermion in the zeroth mode by applying the relevant creation operator on the vacuum state.

In [9]:
```
vac = h.vac
s0 = h.C[0]*vac
s0
```

Out[9]:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Project the state s0 back onto the Fock basis (#FIXME: does NOT work for superpositions Fock state right now.)

In [11]:
```
h.get_occupations(s0)
```

Out[11]:

$$\left(\left[\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right], \quad [1] \right)$$

Create a tunnelling hamiltonian between modes 0, 1. (A la Flensberg, PRL 106, 090503 (2011))

In [13]:
```
nu = sp.symbols('nu', is_complex=True)
nubar = sp.conjugate (nu)
h_tunnel = (nu*h.C[0] - nubar*h.A[0])*(h.C[1] + h.A[1])
h_tunnel
```

Out[13]:

$$\begin{bmatrix} 0 & 0 & 0 & -\nu & 0 & 0 & 0 & 0 \\ 0 & 0 & -\overline{\nu} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\nu & 0 & 0 & 0 & 0 & 0 & 0 \\ -\overline{\nu} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\nu \\ 0 & 0 & 0 & 0 & 0 & 0 & -\overline{\nu} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & -\overline{\nu} & 0 & 0 & 0 \end{bmatrix}$$

The hamiltonian is obviously hermitian, but we can double check

In [14]:
```
h_tunnel - Dagger (h_tunnel)
```

Out[14]:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Create a fock state. Show the real-space (Slater determinant) representation of this state (#FIXME: Not sure if slater works for superpositions of Fock states)

In [18]:
```
s1 = h.add_fock_state([1,0,0])
h.print_slater(s1)
```

Out[18]: ϕ_1

Similarly, the anit-symmetric wavefunctions of 2, 3 fermions:

In [19]:
```
s2 = h.add_fock_state([1,1,0])
h.print_slater(s2)
```

Out[19]:

$$-\frac{\sqrt{2}}{2}\left(\phi_1\phi_2 - \phi_2\phi_1\right)$$

In [20]:
```
s3 = h.add_fock_state([1,1,1])
h.print_slater(s3)
```

Out[20]:

$$-\frac{\sqrt{3}}{3}\left(\phi_1\phi_2\phi_3 - \phi_1\phi_3\phi_2 - \phi_2\phi_1\phi_3 + \phi_2\phi_3\phi_1 + \phi_3\phi_1\phi_2 - \phi_3\phi_2\phi_1\right)$$