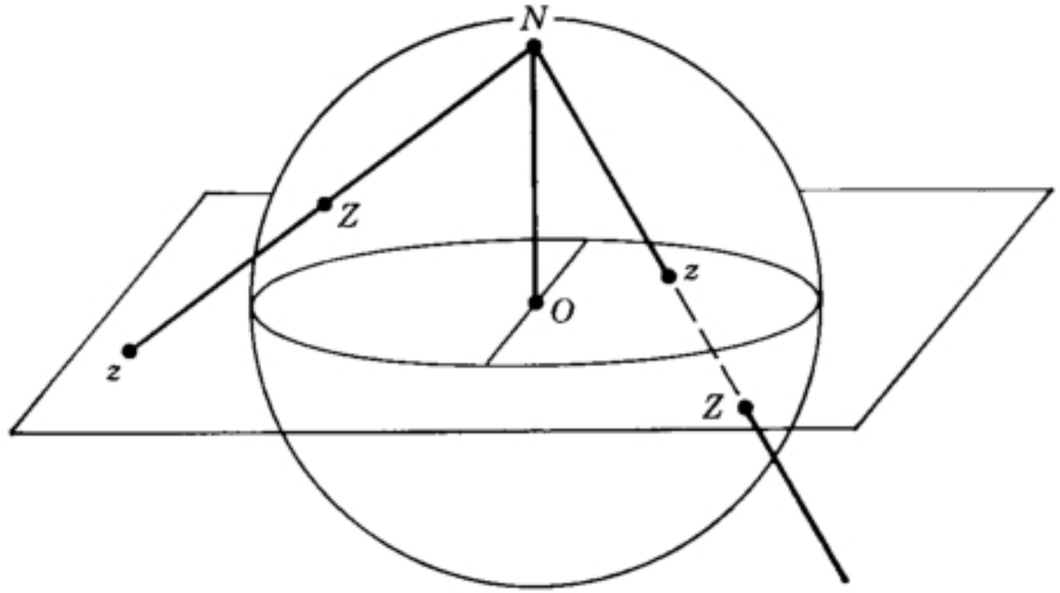
spheres: a python toolbox for higher spin and symmetrization

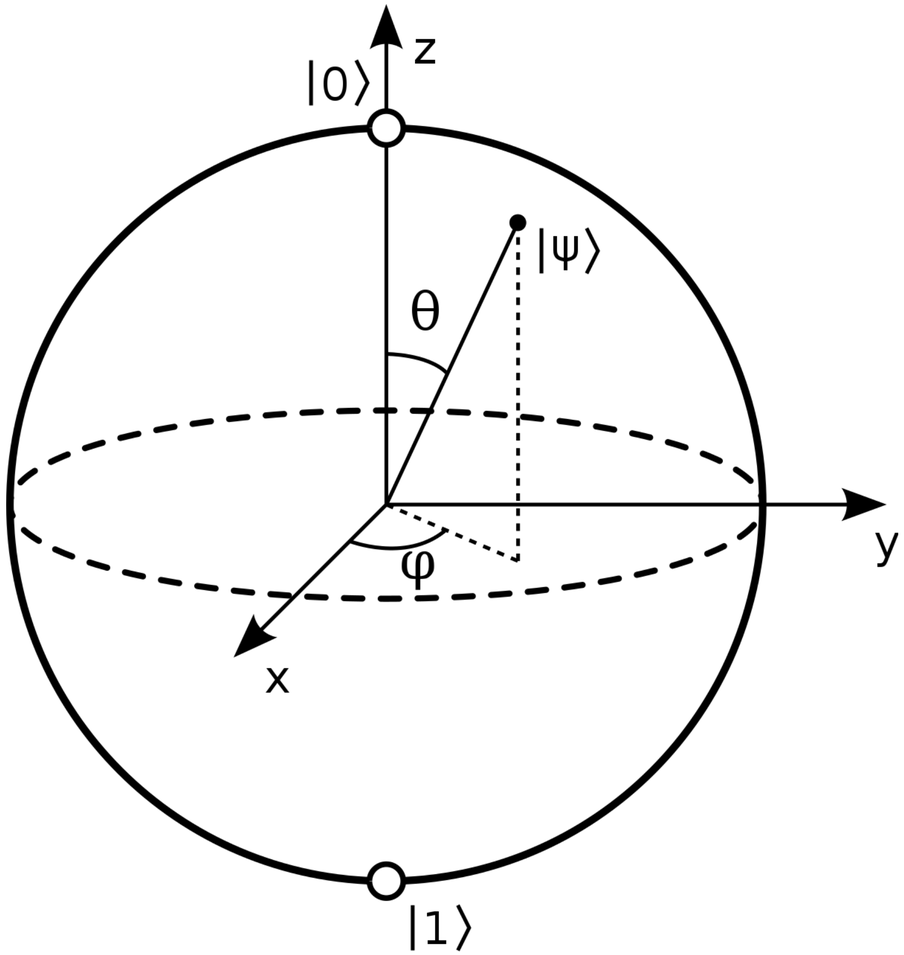
Matthew Weiss



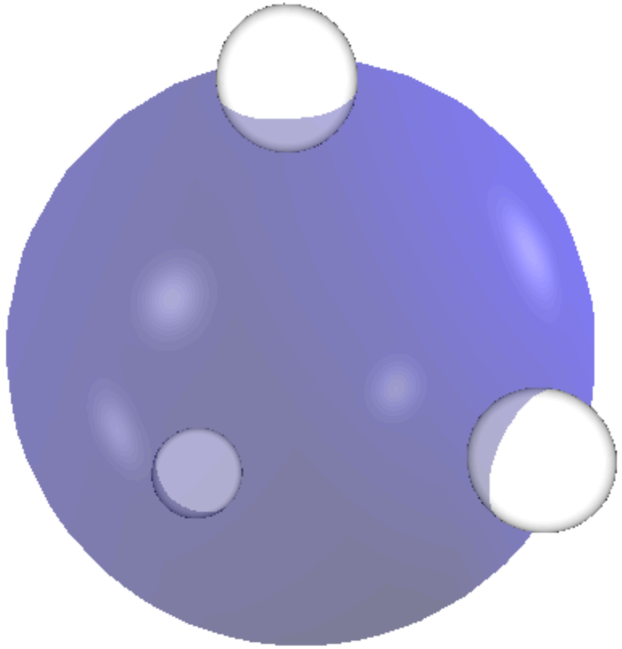
https://github.com/heyredhat/spheres

Mentor: Vesselin

We’re all familiar with the Bloch sphere representation of a qubit…

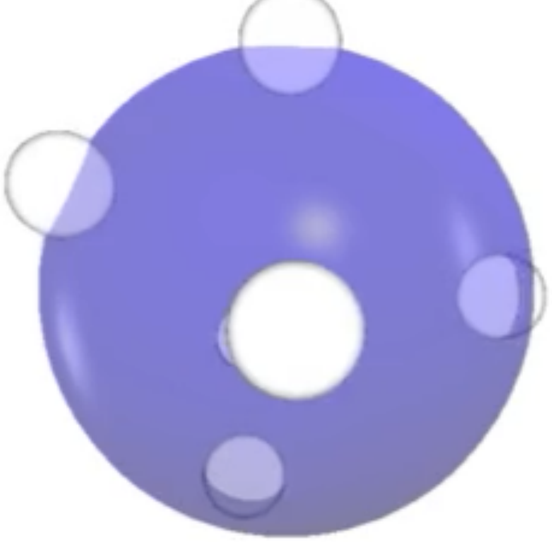


It’s perhaps less well known that as early as 1932,
Ettore Majorana generalized this representation for any spin-j…

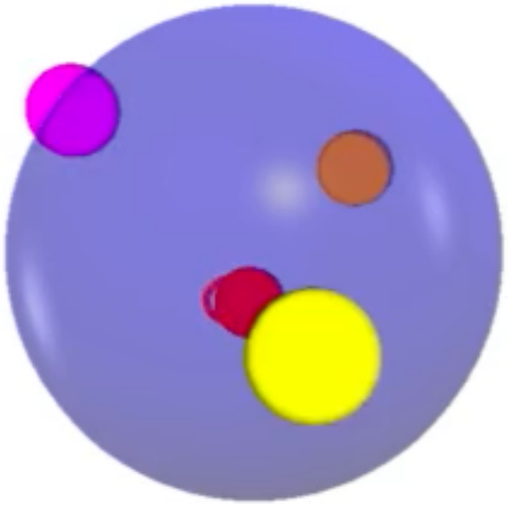


A spin-j state can be represented by a constellation of 2j points on the sphere.

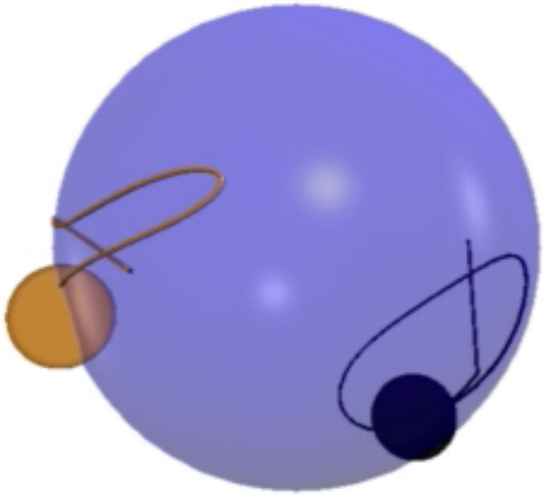
These points are often called “stars.”



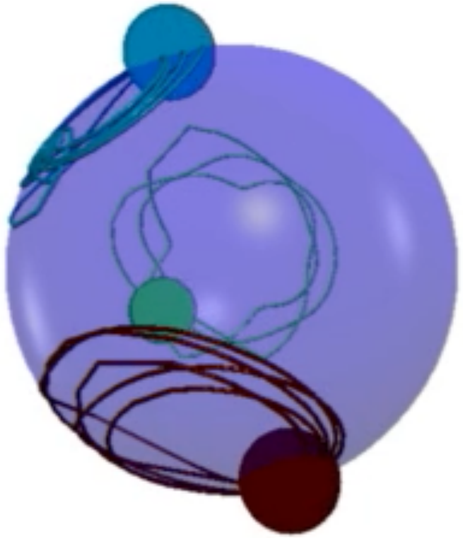
Y Rotation



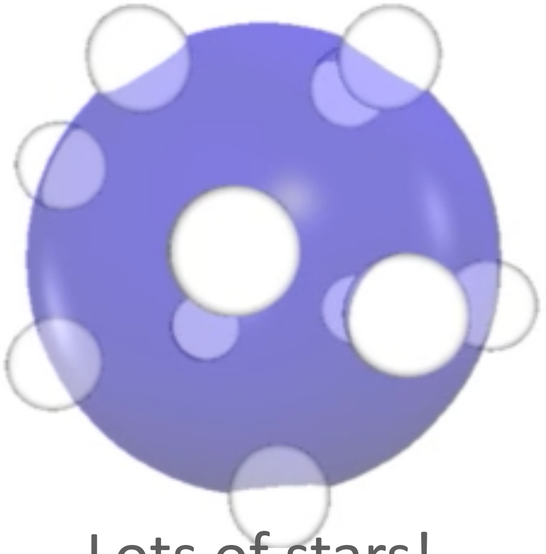
Perturbed out of an eigenstate



Nice patterns



Swaps



Lots of stars!

The Majorana Polynomial

Given a spin-j state in the |j, m> representation: $|\psi\rangle = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \end{pmatrix}$

Form a complex polynomial in an unknown z:

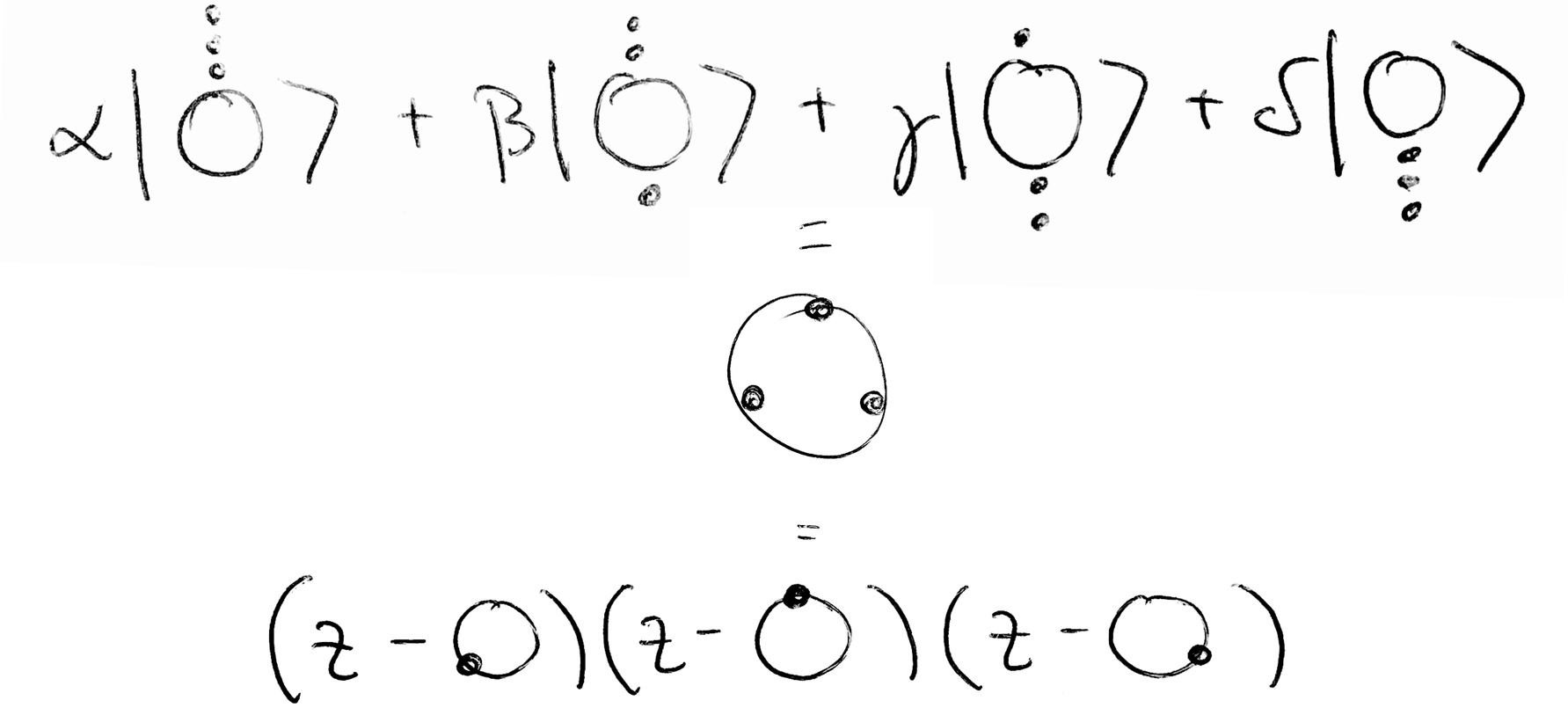
$$p(z) = \sum_{m=-j}^{m=j} (-1)^{j+m} \sqrt{\binom{2j}{j-m}} a_{j+m} z^{j-m}$$

Its roots, stereographically projected from the complex plane to the sphere, yield the constellation.

If you lose a degree, add a root “at infinity.”

The upshot:

By the fundamental theorem of algebra,
a spin-j state *factorizes* into 2j pieces.

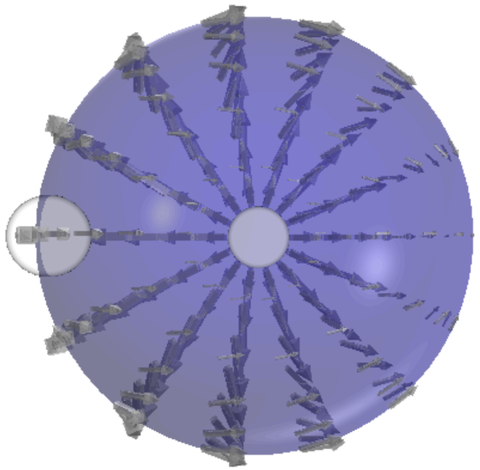


This turns out to be very useful if you want to prepare such states.

The Majorana representation can also be understood in terms of:
spin coherent states.

Spin coherent states have all their angular momentum concentrated in one direction on the sphere.

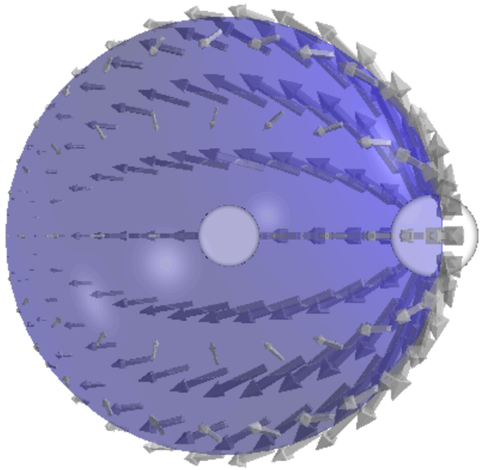
For a given j, they form an (overcomplete) basis for spin states.



Up to a normalization,

$$\langle \tilde{z} \mid \psi \rangle \approx p(z)$$

Where $\tilde{z}$ is the point on the sphere antipodal to z .



The spin coherent amplitude will always be 0 opposite to a Majorana star.

The Majorana stars are those points for which there is a 0% chance that all the angular momentum will be concentrated in the opposite direction.

Moreover, we can use the Majorana representation to understand:
permutation symmetric multiqubit states.

There's a correspondence between |j, m> basis states
and basis states for the permutation symmetric subspace of 2j qubits.

For spin-$\frac{3}{2}$:

$$|\tfrac{3}{2},\tfrac{3}{2}\rangle \rightarrow \ |\uparrow\uparrow\uparrow\rangle$$

$$|\tfrac{3}{2},\tfrac{1}{2}\rangle \rightarrow \tfrac{1}{\sqrt{3}}\Big(\ |\uparrow\uparrow\downarrow\rangle + |\uparrow\downarrow\uparrow\rangle + |\downarrow\uparrow\uparrow\rangle\Big)$$

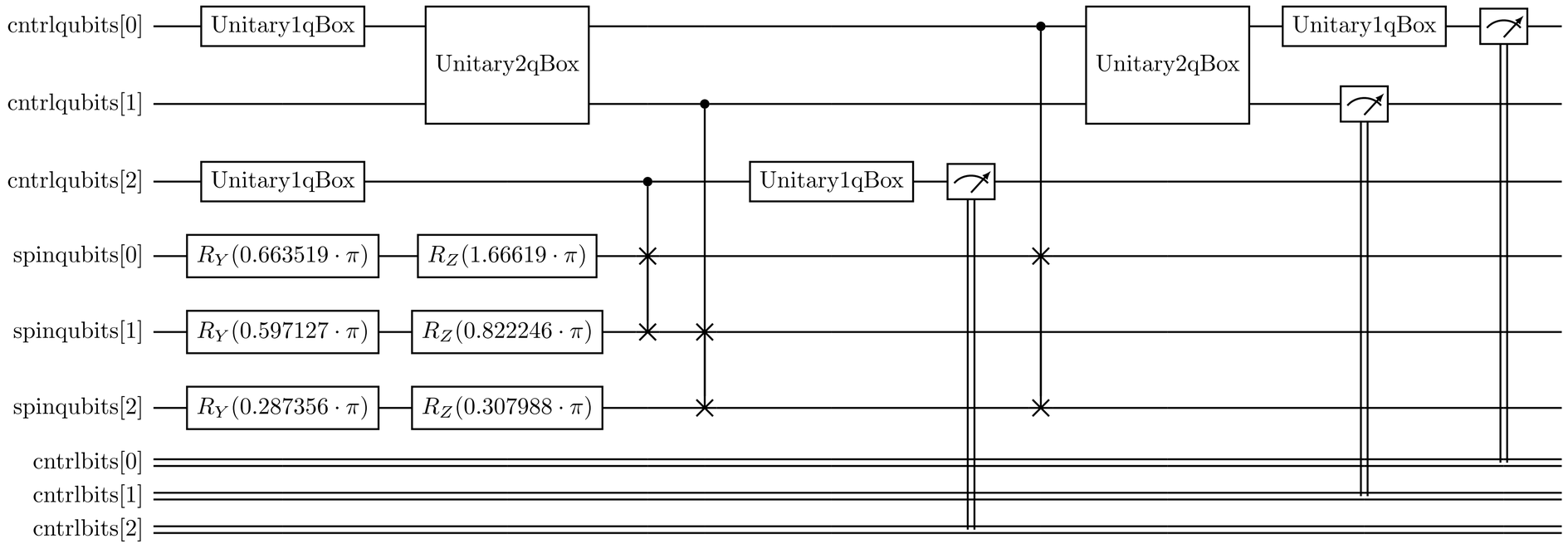
$$|\tfrac{3}{2},-\tfrac{1}{2}\rangle \rightarrow \tfrac{1}{\sqrt{3}}\Big(\ |\downarrow\downarrow\uparrow\rangle + |\downarrow\uparrow\downarrow\rangle + |\uparrow\downarrow\downarrow\rangle\Big)$$

$$|\tfrac{3}{2},-\tfrac{3}{2}\rangle \rightarrow \ |\downarrow\downarrow\downarrow\rangle$$

Alternatively, take:

$P(|\,\phi_0\rangle, |\,\phi_1\rangle, |\,\phi_2\rangle, \ldots)$, where $|\phi_i\rangle$ are 2j qubits
oriented in the directions of the Majorana points,
and P is the permutation symmetric tensor product.

This is suitable for implementation on qubit based quantum computers.



Above, we prepare a spin-3/2 state.
Three qubits are rotated to point in the Majorana directions.
Three auxiliary qubits are initialized in a special state, after which we cascade a series of controlled-swaps, and then undo the initialization.
If we postselect on the control qubits all being 0, the three original qubits will be in the desired permutation symmetric state.

Implemented with Qiskit and Pytket.

Requires $n(n-1)/2$ control qubits, where $n = 2j$.

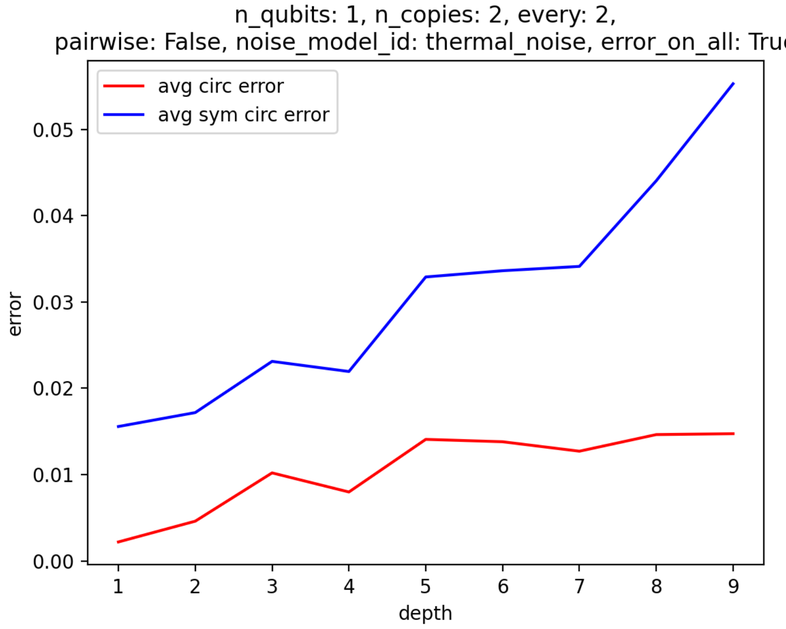
The symmetrization scheme can be used to project more than just qubits!

Explored **stabilization via symmetrization**:

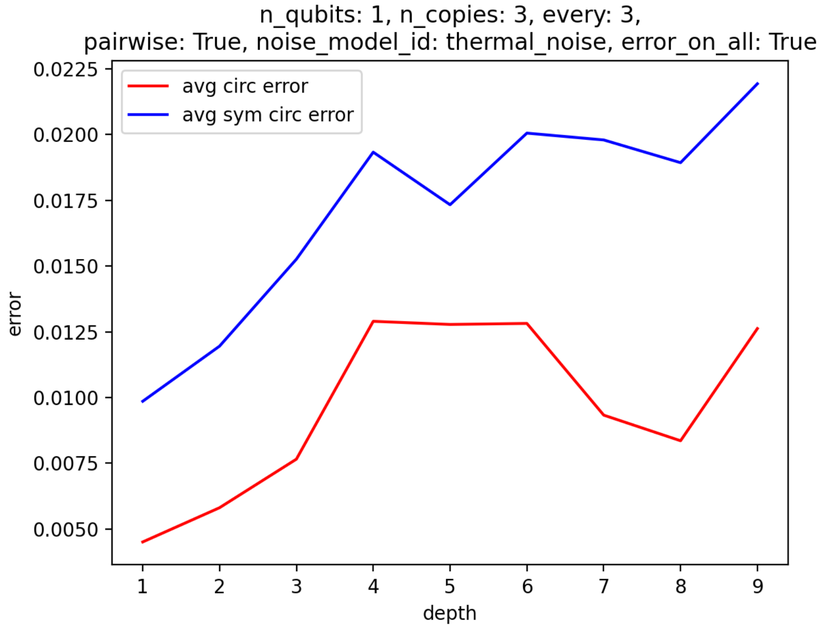
Run n copies of your quantum circuit in parallel.
Periodically project them into the permutation symmetric subspace.

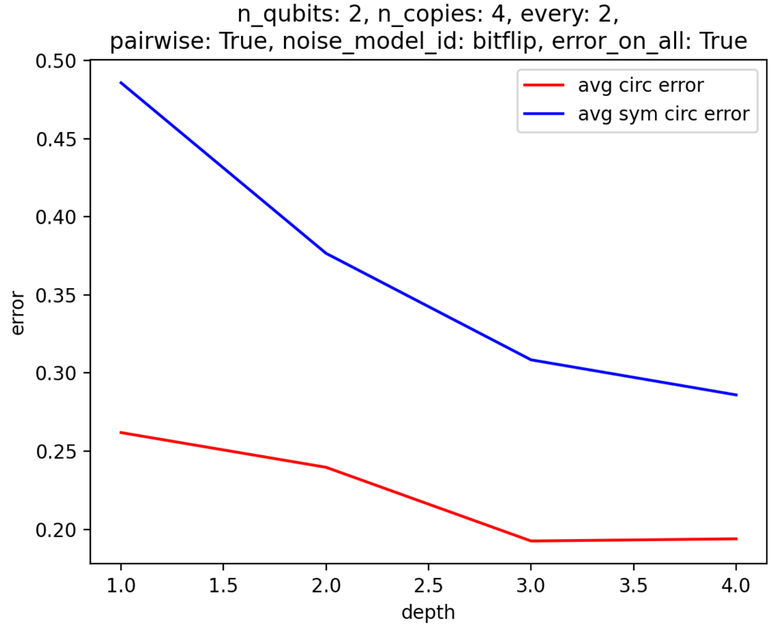
In theory, each copy should already be identical,
in which case projection will have no effect.

Any asymmetry is the result of some error,
which maybe gets projected away.


n_qubits: 1, n_copies: 2, every: 2,
pairwise: False, noise_model_id: thermal_noise, error_on_all: True
avg circ error
avg sym circ error
error
depth


n_qubits: 1, n_copies: 2, every: 3,
pairwise: False, noise_model_id: thermal_noise, error_on_all: True
avg circ error
avg sym circ error
error
depth


n_qubits: 1, n_copies: 3, every: 3,
pairwise: True, noise_model_id: thermal_noise, error_on_all: True
avg circ error
avg sym circ error
error
depth


n_qubits: 2, n_copies: 4, every: 2,
pairwise: True, noise_model_id: bitflip, error_on_all: True
avg circ error
avg sym circ error
error
depth


n_qubits: 2, n_copies: 2, every: 4,
pairwise: True, noise_model_id: ibmq_16_melbourne, error_on_all: True
avg circ error
avg sym circ error
error
depth

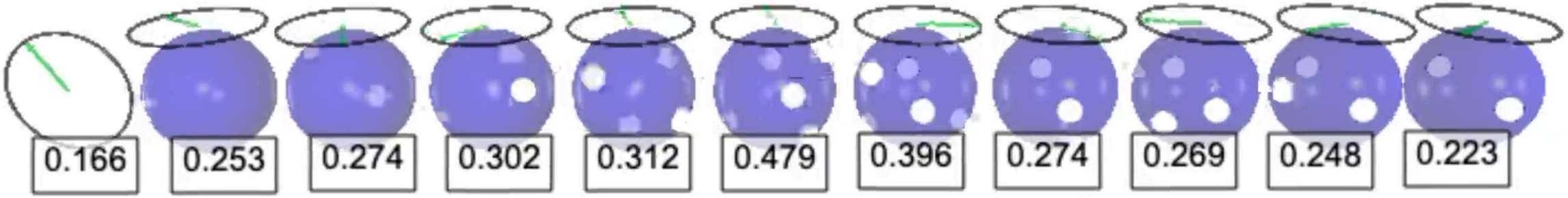
But there’s more!

A spin-j state can also be understood as:

a fixed total energy subspace of two quantum harmonic oscillators.

Reinterpret the Fock space of two oscillators as a tower of spin-j states.

Think of it like the “second quantization” of a qubit.



$$\sigma_X \to b^\dagger a + a^\dagger b$$

$$\sigma_Y \to i(b^\dagger a - a^\dagger b)$$

$$\sigma_Z \to a^\dagger a - b^\dagger b$$

For spin-$\frac{3}{2}$:

$$|\tfrac{3}{2}, \tfrac{3}{2}\rangle \to |\,30\rangle$$

$$|\tfrac{3}{2}, \tfrac{1}{2}\rangle \to |\,21\rangle$$

$$|\tfrac{3}{2}, -\tfrac{1}{2}\rangle \to |\,12\rangle$$

$$|\tfrac{3}{2}, -\tfrac{3}{2}\rangle \to |\,03\rangle$$

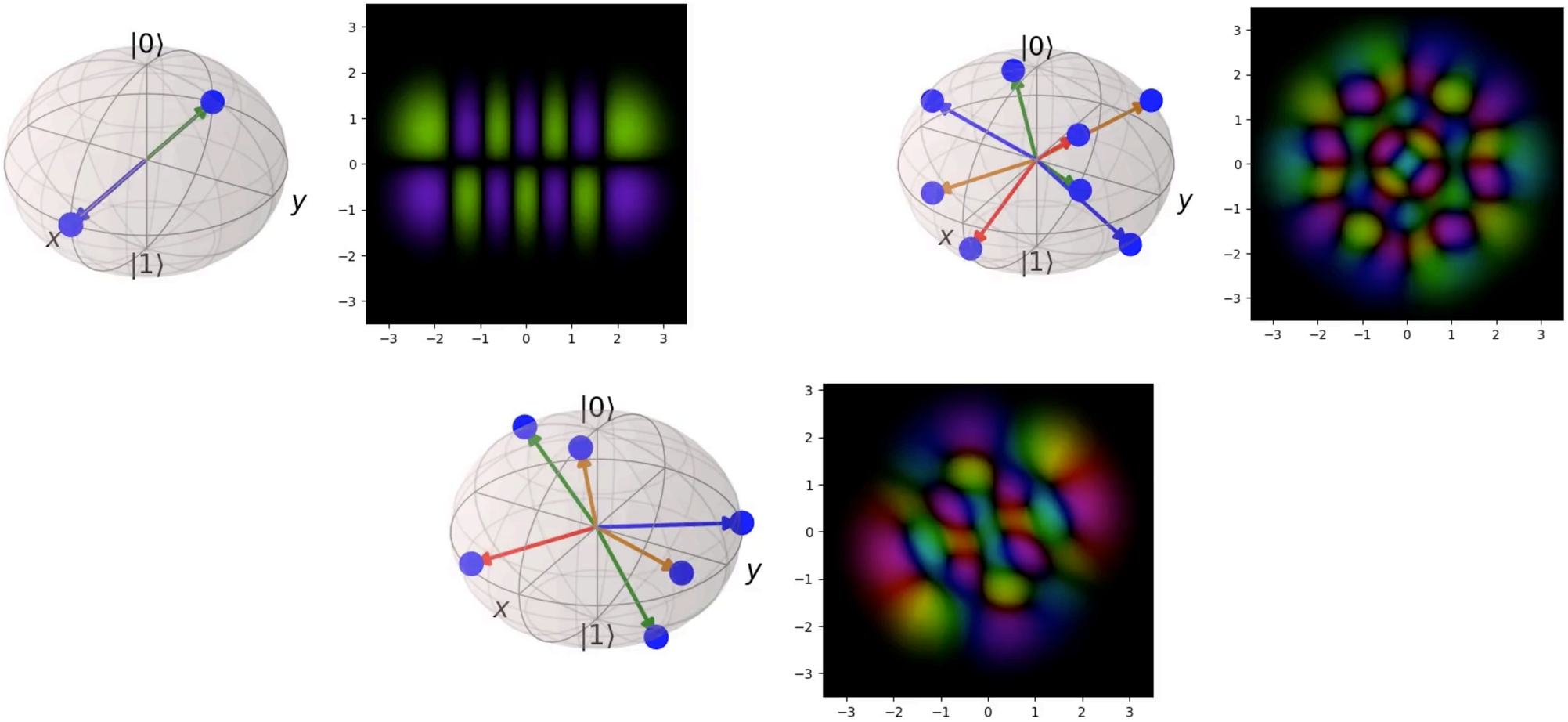
This is suitable for implementation on photonic quantum computers. In StrawberryFields:

```
sph = [np.array([theta_phi[0]/2, theta_phi[1]]) for theta_phi in spin_sph(spin)]
prog = sf.Program(n_modes)
with prog.context as q:
    for i in range(0, n_modes-1, 2):
        Fock(1) | q[i]
        theta, phi = sph[int(i/2)]
        BSgate(theta, phi) | (q[i], q[i+1])
    for i in range(2, n_modes-1, 2):
        BSgate() | (q[0], q[i])
        BSgate() | (q[1], q[i+1])
    for i in range(2, n_modes):
        MeasureFock(select=0) | q[i]
return prog
```

To prepare a spin-3/2 state (with 3 Majorana stars):
Start with 3 pairs of oscillators. Prepare each pair in the |10> state. Send each pair through a beam splitter to rotate them to the desired Majorana directions. We want our spin-j state to be encoded in the first pair (a_0 and a_1), so apply 50/50 beamsplitters to the first of each pair and a_0, and to the second of each pair and a_1. Then postselect on all the other pairs having 0 photons.
The spin-j state will be encoded in a_0 and a_1.

Other features:

- The representation can be generalized to mixed states and operators.
- The Majorana stars can be used to calculate the Berry phase (ongoing).
- They can be used to calculate geometric measures of entanglement (ongoing).
- They can be used, remarkably, to represent the classical modes of structured Gaussian light beams:



Conclusion

- We can represent a spin-j state as a polynomial which factors into roots, yielding a constellation of “stars” on the extended complex plane (the sphere).
- We can represent a spin-j state as the symmetrized tensor product of 2j qubits, oriented in the directions of the stars.
- We can generate circuits that prepare such states on qubit based QC’s.
- We can use symmetrization to (maybe) do error correction.
- We can represent a spin-j state in terms of two quantum harmonic oscillators.
- We can generate circuits that prepare such states on photonic QC’s.
- There are numerous surprising applications of this construction.
- The library makes the relevant classical and quantum calculations easy.
- Very useful for visualization!

Thanks for listening!