1 Details of the capacity algorithm

We determine the side length w by first determining the *resolution* of the set of modules (discussed later) and confirming that it is less than 0.5 units in each dimension. We then started with a small N-dimensional box of radius 0.5, and we incrementally expand the box outward and check whether the frontier region contains any collisions, splitting this frontier into a set of N-dimensional boxes and thoroughly checking each of these boxes for the starting point's representation. There's no analytic way to detect whether a particular set of module phases occur together within a N-dimensional box, but in some cases it's trivial. For example, if one of the phases never occurs anywhere in the box (while accounting for Δ), then it's clear that the set of phases never co-occur in the box. We combine this heuristic with a divide-and-conquer algorithm to split the box into smaller boxes in which at least one phase never occurs. For each box we test a few points in the box to see if they have this representation, then we test whether the box excludes any of the individual phases, and if both checks fail we split the box in half and try again on each half. If the representation is not present, this process will successfully partition the box into smaller boxes that each have this property. If the representation does occur, this process will narrow in on a range of points with representations similar to this representation until it finds it. In this way, we thoroughly search the N-dimensional volume for collisions without needing to choose a fixed high sampling density.

In each module we anchor $\phi_0 = \vec{0}$ to location $\vec{x}_0 = \vec{0}$, so A assigns phase $\vec{\phi}$ to a location $\vec{x}$ via $A\vec{x} \mod 1$. In the algorithm above we need to check whether various N-dimensional boxes contain any points near phase $\vec{0}$. To perform this check in a way that properly considers distances between phases, we decompose A into two matrices: P, a linear mapping from $\mathbb{R}^N$ to the plane $\mathbb{R}^2$, and L which specifies the lattice on that plane, such that $A = L^{-1}P$. We always set L to create a hexagonal lattice, i.e.

$$L = \begin{bmatrix} \cos 0^\circ & \cos 60^\circ \\ \sin 0^\circ & \sin 60^\circ \end{bmatrix}. \tag{1}$$

The matrix P maps a location to a point on the plane that contains the hexagonal lattice:

$$P = \frac{1}{\lambda} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \hat{v}_1 & \hat{v}_2 & \cdots & \hat{v}_N \end{bmatrix}^{-1}. \tag{2}$$

Here $\hat{v}_1$ and $\hat{v}_2$ are orthogonal N-dimensional unit vectors that define this plane, and $\hat{v}_3, \ldots, \hat{v}_N$ define the kernel of P. The period or scale of the grid is defined by λ. The distance between two phases is equal to the shortest distance between the phases on this plane.

To determine whether an N-dimensional box contains a point with a phase near $\vec{0}$, we apply transformation P to each corner of the box to obtain its shadow on the plane. We enumerate nearby lattice points on the plane, drawing circles around each lattice point with diameter equal to the phase resolution Δ, and then we check whether any of these circles intersect the shadow or are contained within the shadow. We combine this technique with the above "divide-and-conquer" and "expanding box" techniques to determine the range over which the code is unique.

We defined the *resolution* of a set of modules as the distance one must move along each dimension before the representation becomes distinguishable from the starting representation. It's possible for this resolution to be more precise in some dimensions than others, for example if all of the modules have similar projection planes and similar kernels the resolution on the projection plane will be precise but along the kernel it will be more coarse. To obtain a single number characterizing this resolution, we numerically computed the smallest N-dimensional hypercube centered at the origin for which every point on the hypercube's surface was distinguishable from the origin. We checked each face of the hypercube using the same divide-and-conquer strategy as above. The resolution is equal to half the side-length of this smallest hypercube, and we computed it to a precision of 0.01 units.

1.1 Phase distance

We base our distance computations in the space of the vector of grid phases with respect to the equivalence relation defined by the lattice (each module encodes a 2D phase on a 2D torus, which we can understand as the quotient of Euclidean space with a hexagonal lattice). We now define a metric on the sets of phases by taking the maximum of their component-wise distances, i.e. for two sets of phases $\boldsymbol{\phi} = (\phi^1, \ldots, \phi^M)$ and $\boldsymbol{\psi} = (\psi^1, \ldots, \psi^M)$ we define

$$d(\boldsymbol{\phi}, \boldsymbol{\psi}) = \max\left\{d_1(\phi^1, \psi^1), \ldots, d_M(\phi^M, \psi^M)\right\},$$

where d_i $(i = 1, \ldots, M)$ denote the metrics inherited from $\mathbb{R}^2$ with respect to each module's underlying lattices.