A quantum computer can be simulated by applying rotations to a unit vector $u \in \mathbb{C}^{2^n}$ where n is the number of qubits. For example, four qubits would have $u \in \mathbb{C}^{16}$. The dimension is 2^n because a register with n qubits has 2^n eigenstates. Quantum operations are "rotations" because they preserve $|u| = 1$.

The Eigenmath function $rotate(u, s, k, \ldots)$ rotates vector u and returns the result. Vector u is required to have 2^n elements where n is an integer from 1 to 15. Arguments $s, k, \ldots$ are a sequence of rotation codes where s is an upper case letter and k is a qubit number from 0 to $n-1$. Rotations are evaluated from left to right. The available rotation codes are

C, k	Control prefix
H, k	Hadamard
P, k, ϕ	Phase modifier
Q, k	Quantum Fourier transform
V, k	Inverse quantum Fourier transform
W, k, j	Swap bits
X, k	Pauli X
Y, k	Pauli Y
Z, k	Pauli Z

Control prefix C, k modifies the next rotation code so that it is a controlled rotation with k is the controlling qubit. Fourier rotations Q, k and V, k are applied to qubits 0 through k. (Q and V ignore any control prefix.)

Error codes

1 Argument u is not a vector or does not have 2^n elements where $n = 1, 2, \ldots, 15$.

2 Unexpected end of argument list (i.e., missing argument).

3 Bit number format error or range error.

4 Unknown rotation code.

Eigenstates $|j\rangle$ are represented by the following vectors. (Each vector has 2^n elements.)

$$
\begin{aligned}
|0\rangle &= (1, 0, 0, \ldots, 0) \\
|1\rangle &= (0, 1, 0, \ldots, 0) \\
|2\rangle &= (0, 0, 1, \ldots, 0) \\
&\vdots \\
|2^n - 1\rangle &= (0, 0, 0, \ldots, 1)
\end{aligned}
$$

A quantum computing algorithm is a sequence of rotations applied to the initial state $|0\rangle$. (Mathematically, the sequence can be collapsed into a single rotation.) Let ψ be the final state of the quantum computer after all the rotations have been applied. Like any other state, ψ is a linear combination of eigenstates.

$$\psi = \sum_{j=0}^{2^n-1} c_j |j\rangle, \quad |\psi| = 1$$

The final step is to measure ψ which will change it to an eigenstate $|j\rangle$ with probability

$$P_j = c_j c_j^*$$

The output from a real quantum computer is always an eigenstate. For example, the result of a two qubit quantum computer is either $|0\rangle$, $|1\rangle$, $|2\rangle$, or $|3\rangle$. In general the c_j's cannot be observed directly. However, we can run the same calculation multiple times to obtain a probability distribution. From the probability distribution we can estimate $c_j c_j^*$ for each eigenstate.

Eigenmath code snippets for before and after *rotate*:

Initialize $\psi = |0\rangle$.

```
n = 4           -- number of qubits
N = 2^n         -- number of eigenstates
psi = zero(N)
psi[1] = 1
```

Compute the probability distribution for state ψ.

```
P = psi conj(psi)
```

Hence

$$\begin{aligned}
\texttt{P[1]} &= \text{probability that } |0\rangle \text{ will be the result} \\
\texttt{P[2]} &= \text{probability that } |1\rangle \text{ will be the result} \\
\texttt{P[3]} &= \text{probability that } |2\rangle \text{ will be the result} \\
&\vdots \\
\texttt{P[N]} &= \text{probability that } |N-1\rangle \text{ will be the result}
\end{aligned}$$

Draw the probability distribution.

```
xrange = (0,N)
yrange = (0,1)
draw(P[ceiling(x)],x)
```

Compute an expectation value.

```
sum(k,1,N, (k - 1) P[k])
```

Make the high order bit "don't care."

```
for(k,1,N/2, P[k] = P[k] + P[k + N/2])
```

Hence for $N = 16$

$$\begin{aligned}
\texttt{P[1]} &= \text{probability that the result will be } |0\rangle \text{ or } |8\rangle \\
\texttt{P[2]} &= \text{probability that the result will be } |1\rangle \text{ or } |9\rangle \\
\texttt{P[3]} &= \text{probability that the result will be } |2\rangle \text{ or } |10\rangle \\
&\vdots \\
\texttt{P[8]} &= \text{probability that the result will be } |7\rangle \text{ or } |15\rangle
\end{aligned}$$