What is the CS Decomposition, and how does one use it to do quantum compiling?(by rrtucci)

Suppose that $U$ is an $N \times N$ unitary matrix, where $N$ is an even number. The CSD (Cosine Sine Decomposition) Theorem states\(^1\) that one can always express $U$ in the form

\[
U = \begin{bmatrix}
  L_0 & 0 \\
  0 & L_1
\end{bmatrix}
D
\begin{bmatrix}
  R_0 & 0 \\
  0 & R_1
\end{bmatrix},
\]

where the left and right matrices $L_0, L_1, R_0, R_1$ are $\frac{N}{2} \times \frac{N}{2}$ unitary matrices, and

\[
D = \begin{bmatrix}
  D_{00} & D_{01} \\
  D_{10} & D_{11}
\end{bmatrix},
\]

\[
D_{00} = D_{11} = \text{diag}(C_1, C_2, \ldots, C_{\frac{N}{2}}),
\]

\[
D_{01} = \text{diag}(S_1, S_2, \ldots, S_{\frac{N}{2}}), \quad D_{10} = -D_{01}.
\]

For all $i \in \mathbb{Z}_{\frac{N}{2}}$, $C_i = \cos \theta_i$ and $S_i = \sin \theta_i$ for some angle $\theta_i$. Eqs.(1) can be expressed more succinctly as

\[
U = (L_0 \oplus L_1)e^{i\sigma_Y \otimes \Theta}(R_0 \oplus R_1),
\]

where $\Theta = \text{diag}(\theta_1, \theta_2, \ldots, \theta_{\frac{N}{2}})$.

**Figure 1:** Diagrammatic representation of CS decomposition, Eq.(1).

Fig. 1 is a diagrammatic representation of the CSD. Note that: (1) Matrix $U$ is assigned to the incoming arrow. (2) Matrix $D$ is assigned to the node. (3) Matrices $R_0 \oplus R_1$ and $L_0 \oplus L_1$ are each assigned to an outgoing arrow.

The CS decomposition was first used for quantum compiling in Tuc99[2]. In the Tuc99 compiling algorithm, the CSD is used recursively. A nice way of picturing

---

\(^1\)Actually, this is only a special case of the CSD Theorem as stated in Ref.[1]—the case which is most relevant to quantum computing. The general version of the CSD Theorem does not restrict the dimension of $U$ to be even, or even restrict the blocks into which $U$ is partitioned to be of equal size.
Figure 2: A CSD binary tree. It arises from the recursive application of the CSD.

this recursive use of the CSD is to represent each CSD application by a node, as in Fig.1. The recursion connects these nodes so as to form a binary tree, as shown in Fig.2. In Fig.2, we start with an initial unitary matrix $U_{in}$ entering the root node, which we define as level 0. Without loss of generality, we can assume that the dimension of $U_{in}$ is $2^{N_B}$ for some $N_B \geq 1$. (If initially $U_{in}$’s dimension is not a power of 2, we replace it by a direct sum $U_{in} \oplus I_r$ whose dimension is a power of two.) We apply the CSD method to $U_{in}$. This yields for level 0 a D matrix $D^{(1)}$, two unitary matrices $L^{(1)}_0$ and $L^{(1)}_1$ on the left side and two unitary matrices $R^{(1)}_0$ and $R^{(1)}_1$ on the right side. Then we apply the CSD method to each of the 4 matrices $L^{(1)}_0$, $L^{(1)}_1$, $R^{(1)}_0$, and $R^{(1)}_1$ that were produced in the previous step. Then we apply the CSD method to each of the 16 $R$ and $L$ matrices that were produced in the previous step. And so on. The nodes of level $N_B$ don’t have $R$, $L$ arrows coming out of them since the $D$ matrices for those nodes are all $1 \times 1$.

Call a central matrix either (1) a single D matrix, or (2) a direct sum $D_1 \oplus D_2 \oplus \cdots \oplus D_r$ of D matrices, or (3) a diagonal unitary matrix. From Fig.2, it is clear that the initial matrix $U_{in}$ can be expressed as a product of central matrices, with each node of the tree providing one of the central matrices in the product. We can use this factorization of $U_{in}$ into central matrices to compile $U_{in}$, if we can find a method for decomposing any central matrix into a SEO. Tuc99 gives such a method.

The Tuc99 algorithm is implemented in a computer program called Qubiter, available at Ref.[3].

References

