PRACTICAL QUANTUM CIRCUITS FOR BLOCK ENCODINGS OF SPARSE MATRICES

Chao Yang

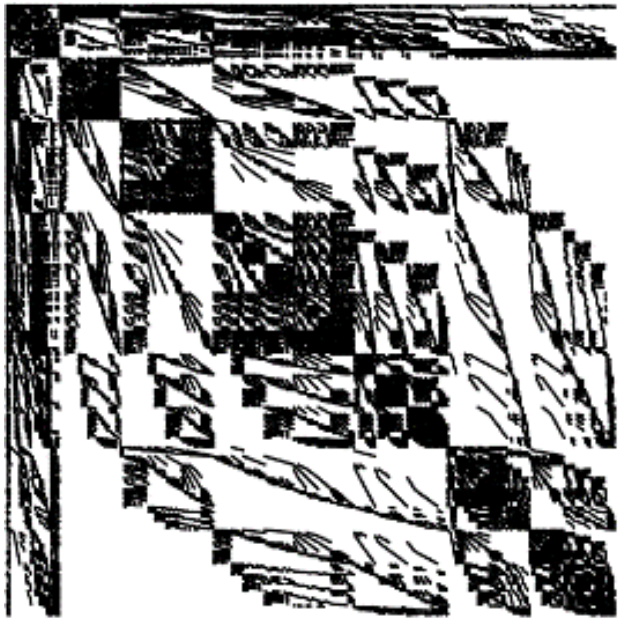
Computational Research Division

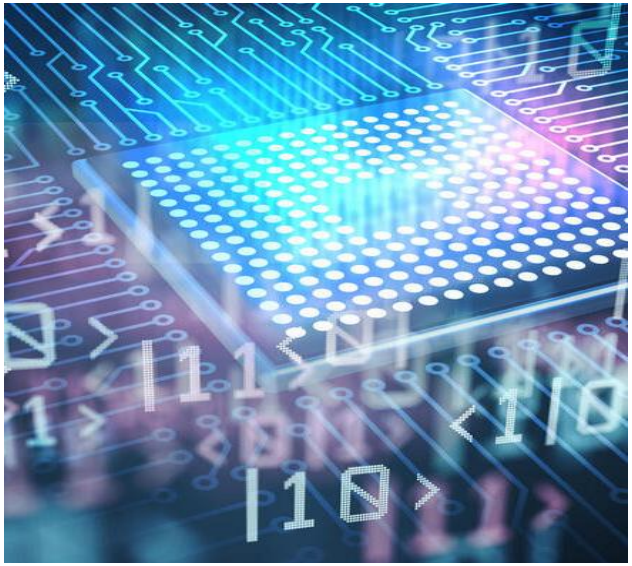
Lawrence Berkeley National Laboratory

Joint work with Daan Camps, Lin Lin and Roel Van Beeumen

Outline

- Large-scale numerical linear algebra (the classical view)
- Quantum algorithms for linear algebra
  - Block encoding
  - Quantum singular value (eigenvalue) transformation
- Efficient block encoding circuits for well structure sparse matrices
- Quantum walk circuit

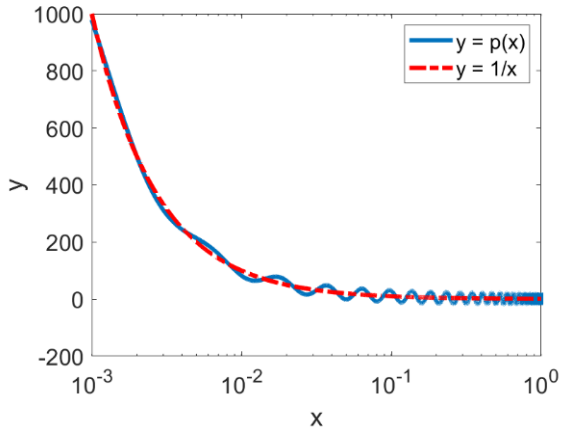


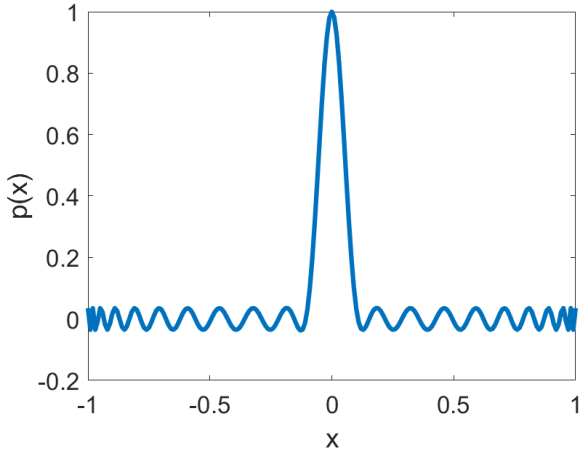


Sparse linear algebra and iterative methods

- Linear systems $Ax = b$, $A \in \mathbb{C}^{N\times N}$, $N = 2^n$
  - $x = A^{-1}b \approx p_k(A)b$
- Least squares $\min_x \|Ax - b\|_2$, $A \in \mathbb{C}^{m\times N}$, $N \geq m$
  - $x = A^{\dagger}b = (A^*A)^{-1}A^*b \approx p_k(A^*A)A^*b$
- Eigenvalue problem: $Ax = \lambda x$
  - $x \approx \delta(A - \lambda I)x_0 = p_k(A)x_0$

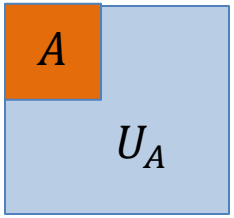
Iterative Methods: A is large but sparse or Av multiplication can be performed efficiently





Quantum (Sparse) Linear Algebra

- Challenge:
  - A is generally not unitary
  - Standard linear algebra operations are non-trivial on a quantum computer, e.g., axpy, dot product etc.
- Solution:
  - Embed A into a much larger unitary U_A that can be decomposed into an efficient quantum circuit (block encoding)



  - Embed $p(A)$ into a much larger unitary U (without forming $p(A)$ explicitly) that can be expressed in terms of U_A and $U_A^\dagger$ (quantum signal processing/quantum singular value transformation)
  - Apply $U_{p(A)}$ to a carefully prepare state, and make measurements

    - Childs, Kathari and Somma, SIAM J. Comput, 2017
    - Gilyén, Su, Low, and Wiebe, ACM SIGACT STOC., 2019

Classical vs Quantum LA

	Classical	Quantum
Theory	Polynomial approximation	Polynomial approximation
Method	Power method, Chebyshev semi-iterative, Krylov subspace methods	Block encoding, QSVT
Practice	Fast sparse matrix-vector multiplication (SpMV), matrix ordering, blocking,	**Efficient block encoding circuits**
Complexity	• number of iterations (degree of polynomial) depends on $\kappa(A)$, error tolerance ε • Flops in SpMV: $O(N)$, $O(N\log N)$ • Data movement	• Degree of polynomial (query complexity) depends on $\kappa(A)$, error tolerance ε • Gate count: poly$(\log N)$ • Circuit topology etc.

Assume fault tolerant quantum computing
Focus on block encoding/QSVT based QLA

Working with block encoded A

- Computing $A|\psi\rangle$:
  - Block encode A into a (much larger) unitary U_A: $U_A = \begin{pmatrix} A & * \\ * & * \end{pmatrix}$
  - apply U to an "extended" vector $\underbrace{|0\rangle}_{\text{ancilla}} \underbrace{|\psi\rangle}_{\text{system}} = \begin{pmatrix} \psi \\ 0 \end{pmatrix}$

$$U_A|0\rangle|\psi\rangle = \begin{pmatrix} A & * \\ * & * \end{pmatrix}\begin{pmatrix} \psi \\ 0 \end{pmatrix} = \begin{pmatrix} A\psi \\ * \end{pmatrix} = \begin{pmatrix} A\psi \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ * \end{pmatrix} = |0\rangle|A\psi\rangle + |1\rangle|*\rangle$$

  - "Retrieve" the product $A\psi$ by measuring the ancilla qubits

$$(|0\rangle\langle 0| \otimes I)(|0\rangle|A\psi\rangle + |1\rangle|*\rangle) = |0\rangle|A\psi\rangle$$

Block encode matrix polynomial

- Goal: compute $p(A)\psi$, for some initial state ψ using a unitary transformation U (that can be efficient encoded by a quantum circuit)
- Embed $p(A)$ into a larger unitary, i.e., construct

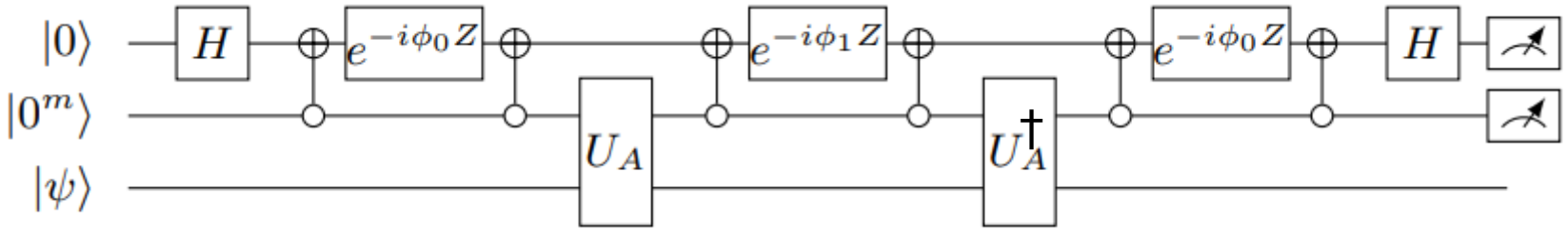
$$U_{p(A)} = \begin{pmatrix} p(A) & * \\ * & * \end{pmatrix}$$

- Prepare initial state $|\hat{\psi}\rangle = |0\rangle|\psi\rangle = \begin{pmatrix} \psi \\ 0 \end{pmatrix}$
- Quantum linear algebra algorithm

$$U_{p(A)}|\hat{\psi}\rangle = |0\rangle|p(A)\psi\rangle + |1\rangle|\phi\rangle$$

Block encoding circuit for $p_2(A)$

- Follows from quantum eigenvalue (singular value) transformation theory
- Phase angles ϕ_0, ϕ_1, ϕ_2 are determined by $p_2(t)$. Can be computed numerically by, e.g., QSPPACK



Dong, Meng, Whaley and Lin, PRA, 2021

How to block encode A (properly scaled)

- Block encode a scalar $0 < \alpha < 1$

$$U = \begin{pmatrix} \alpha & -\sqrt{1-\alpha^2} \\ \sqrt{1-\alpha^2} & \alpha \end{pmatrix}$$

- Block encode a properly scaled matrix (not practical)

$$U = \begin{pmatrix} A & -\sqrt{I - A^* A} \\ \sqrt{1 - A^* A} & A \end{pmatrix}$$

- For a sparse A
  - Encode the position of the nonzero matrix elements
  - Encode the numerical value of the nonzero matrix elements

A more practical example

- $A = \frac{1}{2}\begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_1 \end{pmatrix}$, with $0 \leq |\alpha_1|, |\alpha_2| \leq 1$
- $U_A = \frac{1}{2}\begin{pmatrix} U_\alpha & -U_\beta \\ U_\beta & U_\alpha \end{pmatrix}$, where

$$U_\alpha = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_1 & -\alpha_2 \\ \alpha_2 & \alpha_1 & -\alpha_2 & \alpha_1 \\ \alpha_1 & -\alpha_2 & \alpha_1 & \alpha_2 \\ -\alpha_2 & \alpha_1 & \alpha_2 & \alpha_1 \end{pmatrix}, U_\beta = \begin{pmatrix} \beta_1 & \beta_2 & \beta_1 & -\beta_2 \\ \beta_2 & \beta_1 & -\beta_2 & \beta_1 \\ \beta_1 & -\beta_2 & \beta_1 & \beta_2 \\ -\beta_2 & \beta_1 & \beta_2 & \beta_1 \end{pmatrix}$$

with $\beta_1 = \sqrt{1 - \alpha_1^2}$ and $\beta_2 = \sqrt{1 - \alpha_2^2}$

Quantum circuit

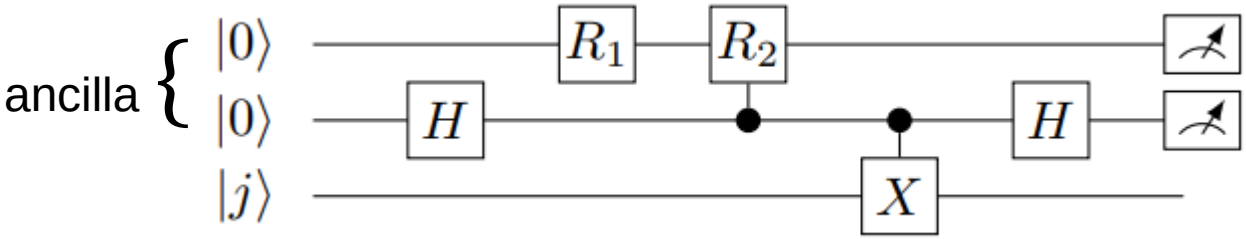
- $U_A = \frac{1}{2}\begin{pmatrix} U_\alpha & -U_\beta \\ U_\beta & U_\alpha \end{pmatrix}$

$= (I \otimes H \otimes I)(R_1 \otimes I \otimes I)[(I \otimes E_1 + R_2 \otimes E_2) \otimes I][I \otimes (E_1 \otimes I + E_2 \otimes X)] \otimes [I \otimes H \otimes I]$

with

$E_1 = \begin{pmatrix} 1 & \\ & 0 \end{pmatrix} = |0\rangle\langle 0|,\ E_2 = \begin{pmatrix} 0 & \\ & 1 \end{pmatrix} = I - E_1,\ H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},\ X = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix}$

$R_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{pmatrix},\ R_2 = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{pmatrix},\ \theta_1 = \cos^{-1}\alpha_1,\ \theta_2 = \cos^{-1}\alpha_2 - \theta_1$



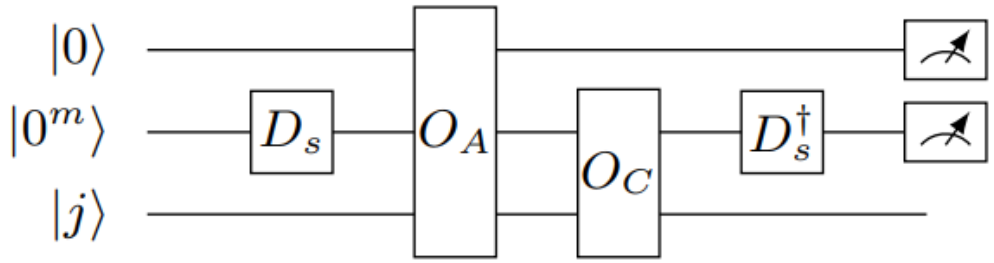
Block encode a sparse matrix

- Use $R = \begin{pmatrix} \alpha & -\sqrt{1-\alpha^2} \\ \sqrt{1-\alpha^2} & \alpha \end{pmatrix}$ to block encode nonzero elements
- Need to encode the location of nonzero elements: $c(\ell, j)$ defines the row index of the ℓth nonzero in column j
- Use controlled operation to perform targeted rotation (what rotation to perform depends on ℓ and j
- Need ancilla qubits: 1 for rotation, $m \leq n$ for non-zero index
- Can't just perform targeted rotation for each non-zero element separately (that would require at least $N = 2^n$ gates whereas our target is $O(\mathrm{poly}(n))$

General structure: block encoding for s-sparse A

- Each column of A has $s = 2^m$ nonzero elements

U_A :



- $D_s|0^m\rangle = \frac{1}{\sqrt{s}}\sum_{\ell=0}^{s-1}|\ell\rangle$ (diffusion operator)
- $O_C|\ell\rangle|j\rangle = |\ell\rangle|c(\ell,j)\rangle$ $c(\ell,j)$ gives the row index of the ℓth nonzero in column j
- $O_A|0\rangle|\ell\rangle|j\rangle = \left(A_{c(\ell,j),j}|0\rangle + \sqrt{1-\left|A_{c(\ell,j),j}\right|^2}|1\rangle\right)|\ell\rangle|j\rangle$
- Verify: $\langle 0|\langle 0^m|\langle i|U_A|0\rangle|0^m\rangle|j\rangle = \frac{1}{s}A_{c(\ell,j),j}\delta_{i,c(\ell,j)} = \frac{1}{s}A_{ij}$

Proof:

- Apply $(I_2 \otimes O_C)O_A(I_2 \otimes D_s \otimes I_n)$ to the jth column of the idenity

$$|0\rangle |0^m\rangle |j\rangle \xrightarrow{D_s} \frac{1}{\sqrt{s}} \sum_{\ell \in [s]} |0\rangle |\ell\rangle |j\rangle$$

$$\xrightarrow{O_A} \frac{1}{\sqrt{s}} \sum_{\ell \in [s]} \left(A_{c(j,\ell),j} |0\rangle + \sqrt{1 - |A_{c(j,\ell),j}|^2} |1\rangle \right) |\ell\rangle |j\rangle$$

$$\xrightarrow{O_c} \frac{1}{\sqrt{s}} \sum_{\ell \in [s]} \left(A_{c(j,\ell),j} |0\rangle + \sqrt{1 - |A_{c(j,\ell),j}|^2} |1\rangle \right) |\ell\rangle |c(j,\ell)\rangle .$$

- Apply $(I_2 \otimes D_s \otimes I_n)$ to the ith column of I

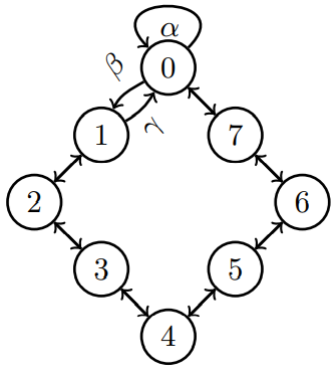
$$|0\rangle |0^m\rangle |i\rangle \xrightarrow{D_s} \frac{1}{\sqrt{s}} \sum_{\ell' \in [s]} |0\rangle |\ell'\rangle |i\rangle .$$

- Take the inner product

$$\langle 0| \langle 0^m| \langle i| U_A |0\rangle |0^m\rangle |j\rangle = \frac{1}{s} \sum_{\ell} A_{c(j,\ell),j} \delta_{i,c(j,\ell)} = \frac{1}{s} A_{ij}.$$

Example 1: banded circulant matrix

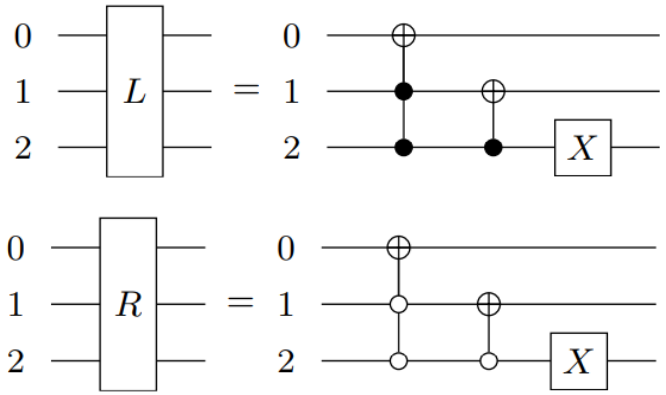
$$A = \begin{pmatrix} \alpha & \gamma & 0 & \cdots & \beta \\ \beta & \alpha & \ddots & \ddots & 0 \\ 0 & \beta & \ddots & \gamma & \vdots \\ \vdots & \ddots & \ddots & \alpha & \gamma \\ \gamma & 0 & \cdots & \beta & \alpha \end{pmatrix}$$



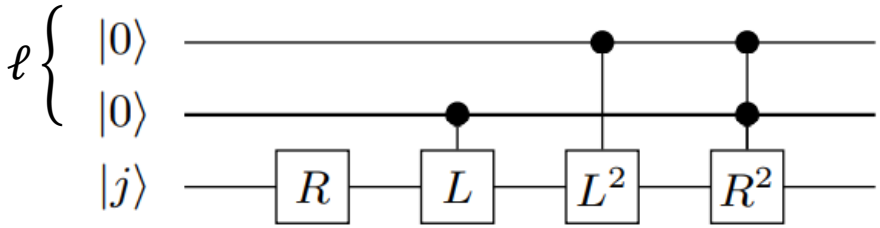
- $c(j,\ell) = \mathrm{mod}(j + \ell - 1, N)$ for $\ell = 0,1,2$
- Construct O_c to map $|j\rangle$ to $|\mathrm{mod}(j-1,N)\rangle$, $|j\rangle$ or $|\mathrm{mod}(j+1,N)\rangle$ depending on the value of ℓ
- $\mathrm{mod}(j-1,N)$ and $\mathrm{mod}(j+1,N)$ can be represented by shift permutations

$$X = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix}$$

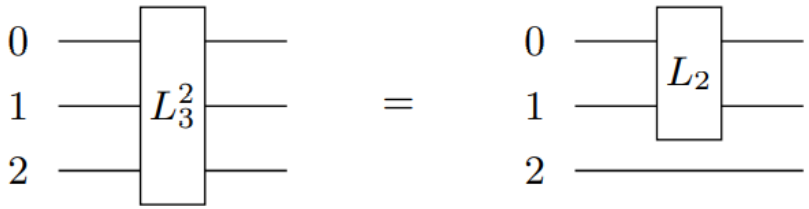
$$R = \begin{pmatrix} 0 & 1 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & 0 & \cdots & \ddots & 0 \end{pmatrix}, \text{ and } L = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 1 \\ 1 & 0 & 0 & \ddots & 0 \\ \vdots & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$



An O_C circuit



- $s = 3$, $\ell = 0,1,2$
- Apply R-shift (maps j to $\mathrm{mod}(j-1, N)$) for all ℓ. This is exactly what need to be done for the sup-diagonal (ℓ=0)
- For $\ell = 1$ (diagonal), apply L-shift to offset the R-shift (maps j to j)
- For $\ell = 2$ (sub-diagonal), apply L^2-shift to achieve an overall L-shift (maps j to $j+1$)
- For $\ell = 3$ (outside of the nonzero band), apply R^2-shift to offset L^2-shift

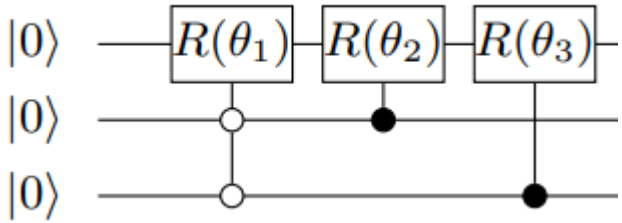


O_A circuit

- Controlled rotations
- In general, the rotation is determined (controlled) by both $|\ell\rangle$ and $|j\rangle$ qubits. But for circulant matrix, the rotation is independent of $|j\rangle$
- Use an alternative definition of $c(j,\ell)$:

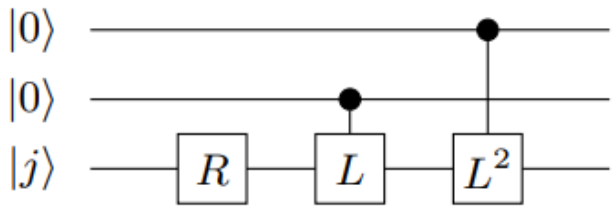
$$c(j,\ell) = \begin{cases} j & \text{if } \ell = 0 \text{ (diagonal) or } 3, \\ j+1 & \text{if } \ell = 1 \text{ (subdiagonal)}, \\ j-1 & \text{if } \ell = 2 \text{ (supdiagonal)}. \end{cases}$$

- Apply $R(\theta_1)$ with $\theta_1 = \cos^{-1}(\alpha - 1)$ when $\ell = 0$
  $(R(\theta_1)|0\rangle|00\rangle + |0\rangle|11\rangle + |0\rangle[|01\rangle + |10\rangle])|j\rangle$
- Apply $R(\theta_2)$ with $\theta_2 = \cos^{-1}\beta$ when $\ell = 1$ (subdiagonal)
- Apply $R(\theta_3)$ with $\theta_3 = \cos^{-1}\gamma$ when $\ell = 2$ (supdiagonal)

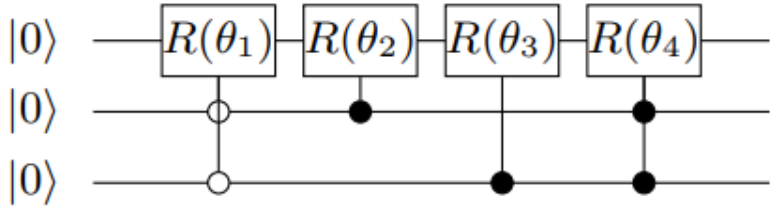


Alternative O_C and O_A circuit

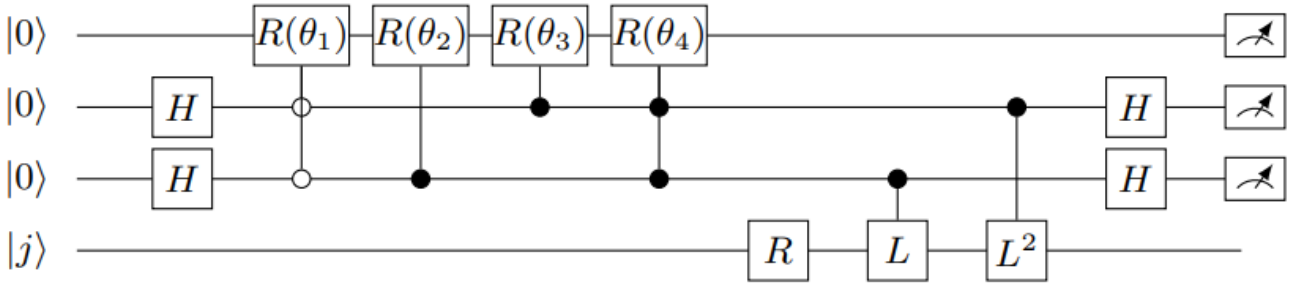
- O_C and O_A are not unique
- Alternative O_C



- Alternative O_A



- Complete circuit



Total number of controlled rotations and permutations depend only on s.

Each controlled operation can be implemented with polylog(N) gates

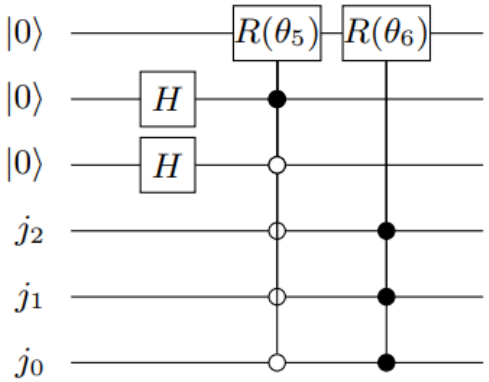
Banded Toeplitz matrix

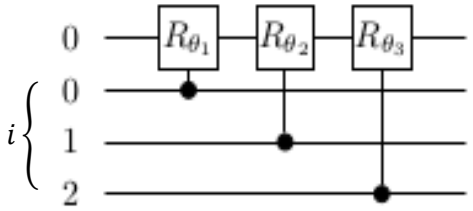
- Tridiagonal

$$A = \begin{pmatrix} \alpha & \gamma & 0 & \cdots & 0 \\ \beta & \alpha & \ddots & \ddots & 0 \\ 0 & \beta & \ddots & \gamma & \vdots \\ \vdots & \ddots & \ddots & \alpha & \gamma \\ 0 & 0 & \cdots & \beta & \alpha \end{pmatrix}$$

- Additional circuit components to zero out γ and β at the lower left and upper right corner of the previous A

$\theta_5 = \cos^{-1}\beta$
$\theta_6 = \cos^{-1}\gamma$

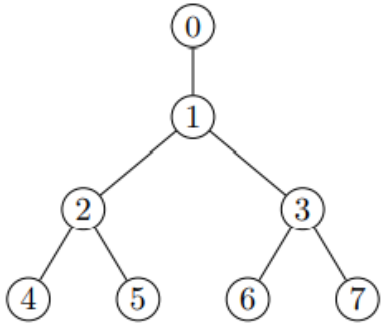




- Can be further modified to block encode non-Toeplitz banded matrices

Example 2: extended binary tree

$$A = \begin{pmatrix} \gamma & \beta & & & & & & \\ \beta & \alpha & \beta & \beta & & & & \\ & \beta & \alpha & & \beta & \beta & & \\ & \beta & & \alpha & & & \beta & \beta \\ & & \beta & & \gamma & & & \\ & & \beta & & & \gamma & & \\ & & & \beta & & & \gamma & \\ & & & \beta & & & & \gamma \end{pmatrix}$$



- Positions and values of nonzero both depend on row and column indices
- Each column has at most $s = 4$ nonzeros

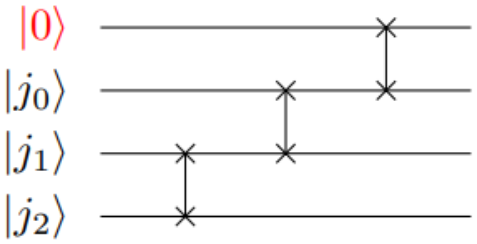
$$c(j,\ell) = \begin{cases} j & \text{if } \ell = 0 \text{ (diagonal);} \\ 2j & \text{if } \ell = 1 \text{ (left child);} \\ 2j+1 & \text{if } \ell = 2 \text{ (right child);} \\ \lfloor j/2 \rfloor & \text{if } \ell = 3 \text{ (parent);} \end{cases}$$

The O_C circuit

- $c(j,\ell)$ uses multiplication and division by 2 in addition to increment by 1

$$c(j,\ell) = \begin{cases} j & \text{if } \ell = 0 \text{ (diagonal);} \\ 2j & \text{if } \ell = 1 \text{ (left child);} \\ 2j+1 & \text{if } \ell = 2 \text{ (right child);} \\ \lfloor j/2 \rfloor & \text{if } \ell = 3 \text{ (parent);} \end{cases}$$

- Multiplication/division implemented by a SWAP circuit



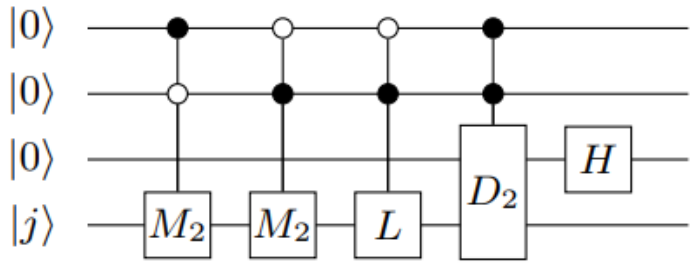
$\times 2$:

$1 = |001\rangle \rightarrow |001\rangle \rightarrow |010\rangle$

$2 = |010\rangle \rightarrow |100\rangle \rightarrow |100\rangle$

$5 = |0101\rangle \rightarrow |1001\rangle \rightarrow |1001\rangle \rightarrow |1010\rangle$

- O_C:



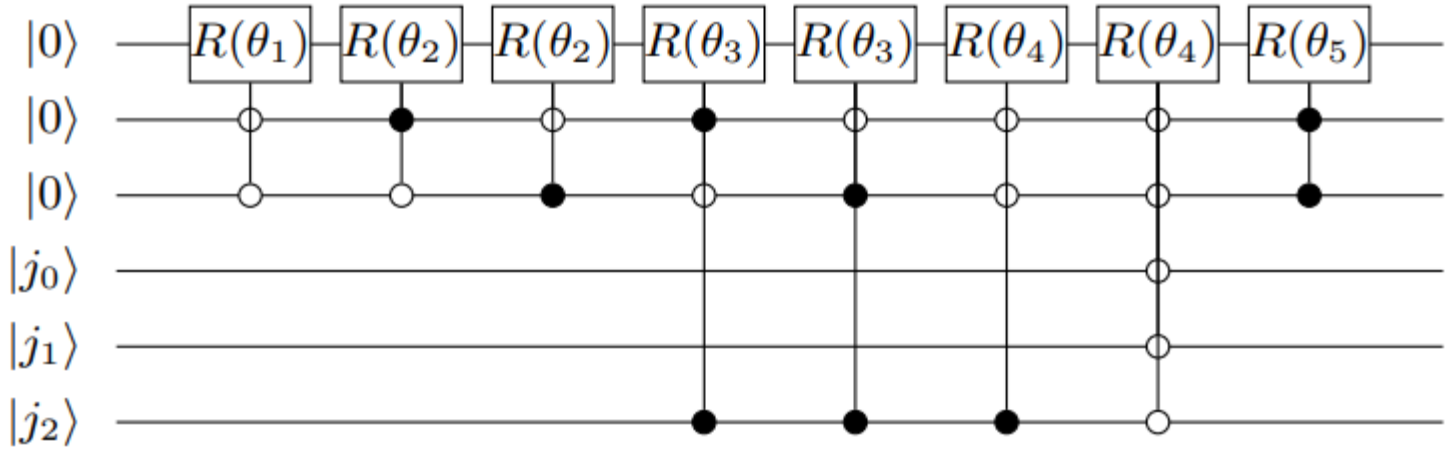
$\div 2$:

$5 = |0101\rangle \rightarrow |0110\rangle \rightarrow |1010\rangle$

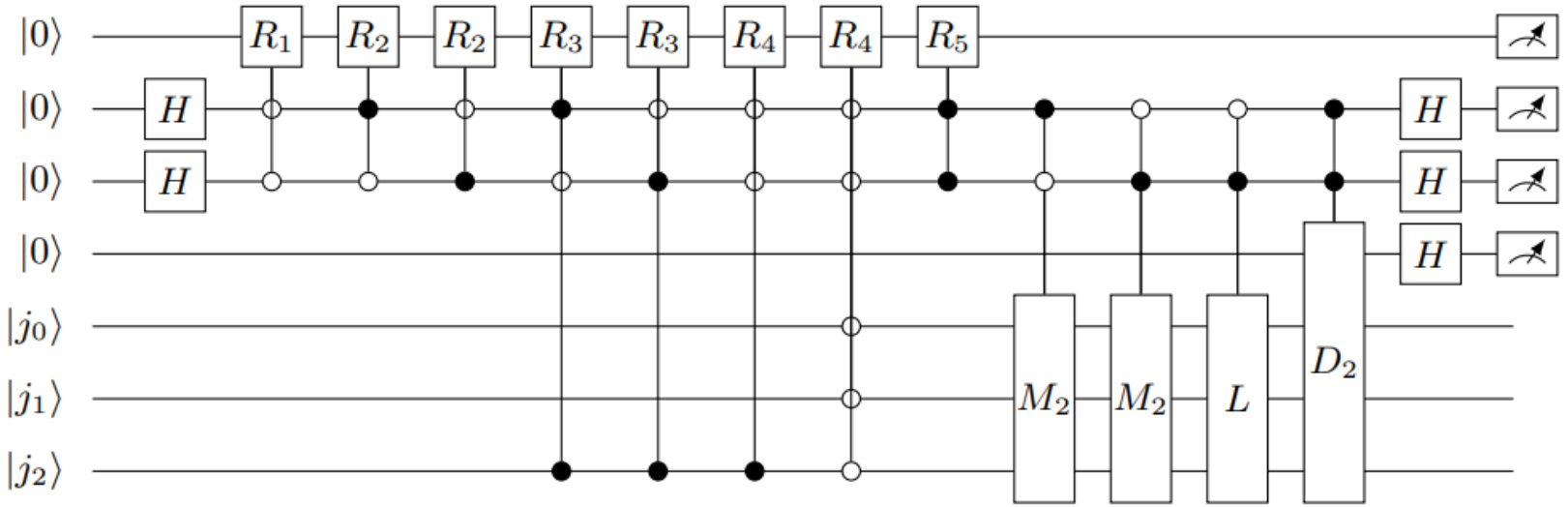
$6 = |0110\rangle \rightarrow |0101\rangle \rightarrow |0011\rangle$

The O_A circuit

- Use $R(\theta_1)$, $\theta_1 = \cos^{-1}\alpha$ to place α on the diagonal ($\ell = 0$)
- Use $R(\theta_2)$, $\theta_2 = \cos^{-1}\beta$ to place β at $U_{i,j}$ for $i = 2j$ and $i = 2j$+1 ($\ell = 1,2$)
- Use $R(\theta_3)$, $\theta_3 = \frac{\pi}{2} - \theta_2$ to zero out $U_{2j,j}$ and $U_{2j+1,j}$ for leaf j
- Use $R(\theta_4)$, $\theta_4 = \cos^{-1}\gamma - \theta_1$ to adjust the diagonal elements associated with root and leaf columns
- Use $R(\theta_5)$, $\theta_5 = \cos^{-1}\beta - \theta_1$ to place β at $U_{\left\lfloor \frac{j}{2} \right\rfloor, j}$



The complete circuit



- The number of controlled rotations depends only on s, independent of matrix dimension
- The number of gates used implement M_2, D_2, L, and each controlled rotation is a polynomial of n

General circuit construction strategy

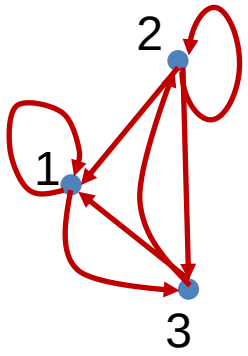
- O_C for nonzero structure, O_A for numerical value
- The constructions of O_C and O_A are intertwined
- Identify the most general case of $c(j, \ell)$ that can be efficiently implemented in O_C (only requires control from ℓ)
  - "Efficient" means constant number of controlled gates, possible if $s = \max(\ell)$ is a small constant
- Encode all other (constant number of) special cases in O_A (requires control from both j and ℓ)
  - Banded circulant matrix: the second subdiagonal set to 0
  - Extended binary tree: root, leaves

Gate complexity: polylog(N)

Efficient circuit for sparse quantum walks

- Block encode random Markov chain (doubly stochastic) matrix $P = P^T$

$$P_{i,j} \geq 0, \sum_i P_{i,j} = 1$$



- Random walk $w = P^k v$
- Quantum walk: block encode $T_k(P)$, where $T_k(t)$ is the kth degree of Chebyshev polynomial

$$w = U_{T_k(P)}(|v\rangle|0\rangle) = |T_k(P)v\rangle|0\rangle + |*\rangle|0\rangle$$

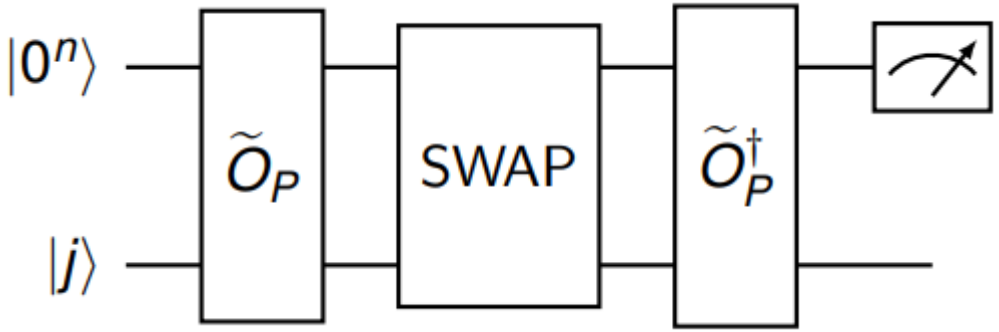
- Quantum (query) efficiency: $O(N)$ vs $O(\sqrt{N})$

Block encode s-sparse P

- Previous techniques yield a block encoding circuit for A/s
- For many applications, working with A/s is just as good as working with A (can always rescale the solution afterwards)
- For quantum walks, we really need to block encode $T_k(P)$ instead of $T_k(P/s)$
- One remedy is to express $T_k(P)$ as a linear combination of $T_j(P/s)$ for $j = 0,1,\ldots,k$, but that is not efficient!
- An alterative is to construct the block encoding of P (instead of P/s) and $T_k(P)$ directly.

Circuit for the block encoding of P

- General structure of U_P



- O_P transformation

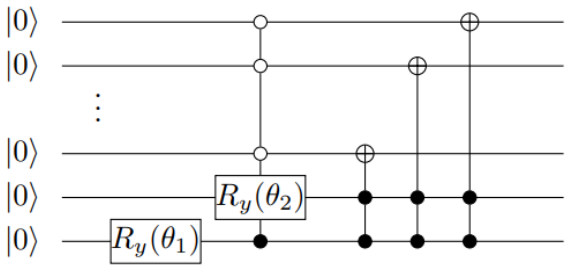
$$O_P|0^n\rangle|j\rangle = \sum_k \sqrt{P_{jk}}|k\rangle|j\rangle$$

- SWAP $|i\rangle|j\rangle = |j\rangle|i\rangle$
- $U_{T_k(P)} = (Z_\Pi U_P)^k$, where $Z_\Pi = 2|0^n\rangle\langle 0^n| \otimes I_n - I$

Circuit for block encoding of P (walk on cyclic graph)

- $0 < \alpha, \beta < 1, \ \alpha + 2\beta = 1$
- O_P circuit constructed in 2 steps
  - Construct

$$K|0^n\rangle = \begin{pmatrix} \sqrt{\alpha} & \sqrt{\beta} & \cdots & \sqrt{\beta} \end{pmatrix}^T$$



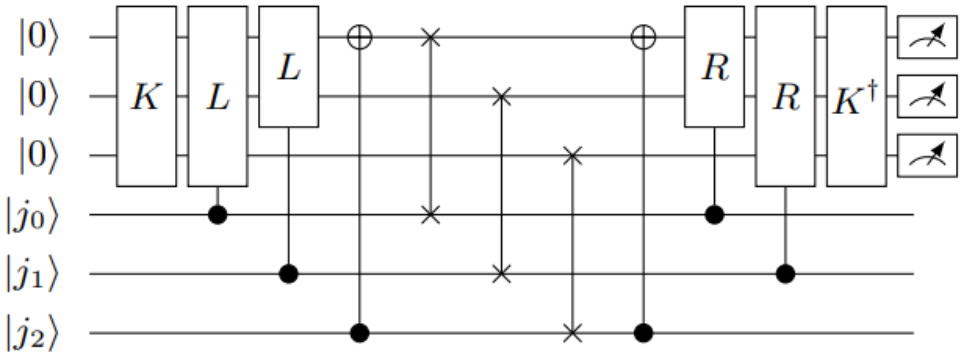
  - Construct permutation using powers of L shift

$$L_n^j|Pe_1\rangle|j\rangle = |Pe_j\rangle|j\rangle$$

$$L_n^j = \left(L_n^{2^{n-1}}\right)^{j_1}\left(L_n^{2^{n-2}}\right)^{j_2}\cdots\left(L_n^{2^0}\right)^{j_n}$$

$$j_1, j_2, \ldots, j_n \in \{0,1\}, \ L_n^2 = L_{n-1} \otimes I_2$$

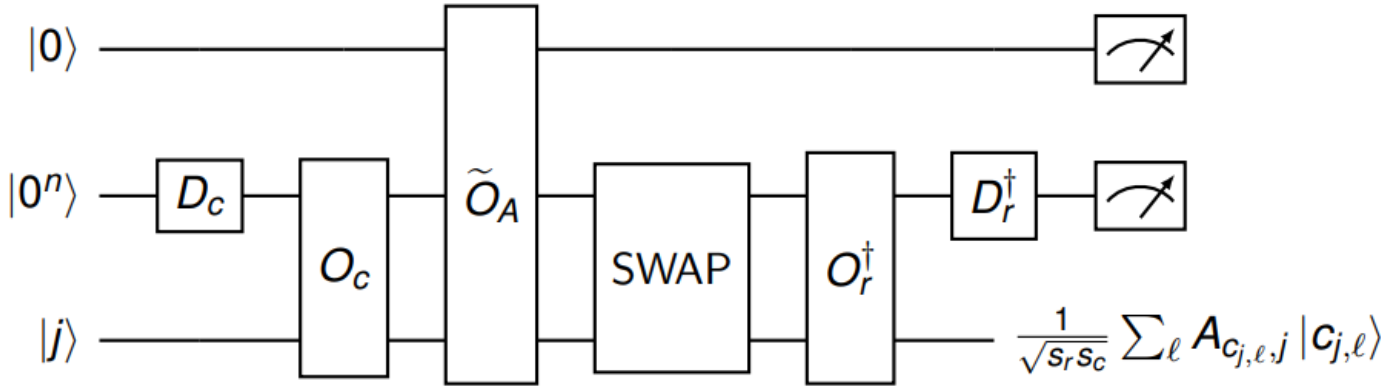
$$P = \begin{pmatrix} \alpha & \beta & 0 & \cdots & \beta \\ \beta & \alpha & \ddots & \ddots & 0 \\ 0 & \beta & \ddots & \beta & \vdots \\ \vdots & \ddots & \ddots & \alpha & \beta \\ \beta & 0 & \cdots & \beta & \alpha \end{pmatrix}$$



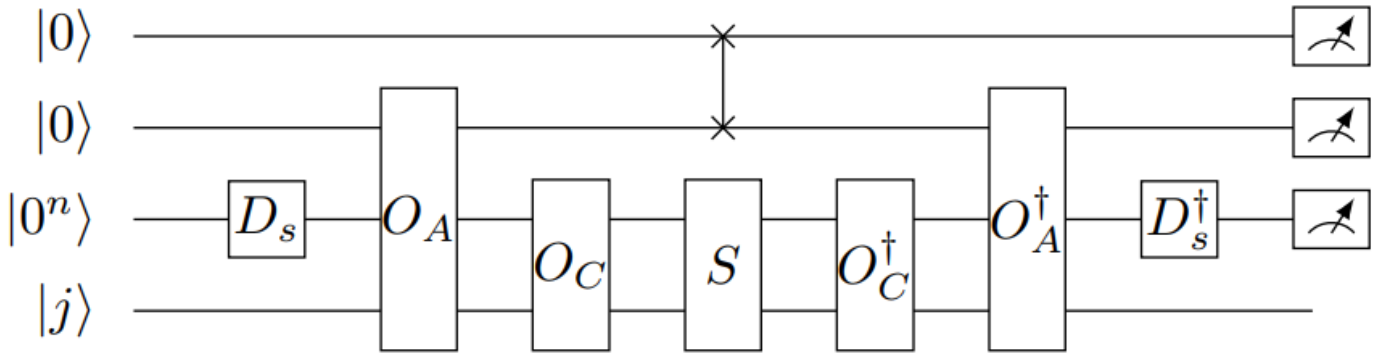
Loke and Wang, Ann. Physics, 2017

Other block encoding circuit

- Sparsity specified by both row and column indices



- Hermitian block encoding of Hermitian matrices



Concluding remarks

- Block encoding based quantum (sparse) linear algebra is practical only if the block encoded (sparse) matrix can be efficiently decomposed into a circuit with poly(log(N))gates
- The decomposition is non-trivial for even well structure sparse matrices
- Efficient decomposition requires
  - efficient O_c to encode the position of the non-zero matrix elements
  - efficient O_A to encode the numerical values (through controlled rotations)
  - The construction of O_c and O_A are closely coupled in general
- Block encoding random walk requires a different strategy
- Many other issues to consider in practice