A Survey of the Quantum Singular Value Transformation (QSVT)

1 Introduction

QSVT is an algorithmic framework that, given a block encoding of a matrix, applies polynomial transform p to singular values while preserving its singular vectors.

Various quantum algorithms build upon this idea, like Grover's search and Shor's algorithm. QSVT is also applied to several Quantum Machine Learning tasks, including the construction of support matrix machines.

Here, we seek to explore the setup, implementation details and computational advantages of QSVT in a quantum inspired molecular analysis context.

add more stuff here

2 Mathematical Algorithm

Input: A matrix $\boldsymbol{X}$ (n × n) whose singular value decomposition is $\mathbf{X} = \mathrm{W}\Sigma\mathrm{V}\dagger$, where Σ is a diagonal matrix with singular values of $\boldsymbol{X}$.

Input: A polynomial $\boldsymbol{P}$

Output: A unitary matrix whose top block equals $\boldsymbol{P}(\boldsymbol{X})$, or $\begin{bmatrix} WP(\Sigma)V^{\dagger} & . \\ . & . \end{bmatrix}$

1. Prepare a unitary matrix $\boldsymbol{U}$ (2n × 2n) that encodes $\boldsymbol{X}$ onto the top left entry of $\boldsymbol{U}$ s.t. we have a block encoding of $\boldsymbol{X}$:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{X} & . \\ . & . \end{bmatrix}$$

$A = (\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I) \Longleftrightarrow U = \begin{bmatrix} A & . \\ . & . \end{bmatrix}$ *

2. Initialize an *n* cubit state $|0\rangle^{\otimes n}$

3. If polynomial $\boldsymbol{P}$ is even ($\boldsymbol{P}(-\mathrm{x}) = \boldsymbol{P}(\mathrm{x})$ for all x), then apply the following to $|0\rangle^{\otimes n}$

$$\tilde{\Pi}_{\phi_1} U \prod_{k=1}^{\frac{d-1}{2}} \Pi_{\phi_{2k}} U^{\dagger} \tilde{\Pi}_{\phi_{2k+1}} U$$

4. If polynomial $\boldsymbol{P}$ is odd ($\boldsymbol{P}(-\mathrm{x}) = -\boldsymbol{P}(\mathrm{x})$ for all x), then apply the following to $|0\rangle^{\otimes n}$

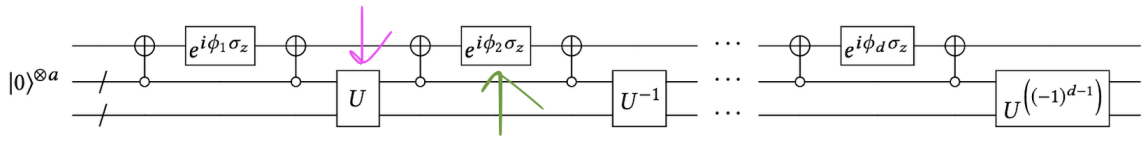
$$\prod_{k=1}^{\frac{d}{2}} \Pi_{\phi_{2k}-1} U^{\dagger} \tilde{\Pi}_{\phi_{2k}} U$$

Definitions:

Π	Projector	Fixed projector that selects the "reference" subspace, usually the block-encoding ancilla state.
$\tilde{\Pi}$	Projector	Conjugated version of Π under U. Projects onto the image of the Π subspace under $U^\dagger$.
Π_ϕ	Controlled phase unitary	Phase rotation conditioned on being in the range of Π.
$\tilde{\Pi}_\phi$	Controlled phase unitary	Phase rotation conditioned on being in the range of $\tilde{\Pi}$.

The tilde (~) is used in the projector-controlled phase gates to distinguish whether they act on the column or row subspaces.

3 Circuit Representation



The **Unitary Operator** transforms entries to a polynomial of a single higher degree. The **Signal Processing Operator** allows the fine tuning of the coefficients.

The following is an example, where *a* is a constant that lies between -1 and 1 (doesn't have to be a matrix). U(a) is a block-encoded unitary with *a* in the top left. S(...) is the Signal Processing Operator.

$$S(-\pi/2)U(a)S(\pi/2)U(a)S(0) = \begin{pmatrix} 2a^2-1 & 2a\sqrt{1-a^2} \\ -2a\sqrt{1-a^2} & 2a^2-1 \end{pmatrix}.$$

4 Quantum-Enhanced Molecular Feature Extraction

In this application, we provide a Hamiltonian matrix $\boldsymbol{H}$ as the input matrix (via the `build_pi_huckel_hamiltonian(SMILE string)` function, which produces the concrete matrix that plays the role of $\boldsymbol{X}$. This application attempts to apply a quantum-inspired polynomial filter to molecular Hamiltonians to extract rich structural features that can improve property prediction and chemical insight. We run this against a classical Chebyshev filter to benchmark runtime performances and quality of information extracted

1) Normalize $\boldsymbol{H}$: `normalize_hamiltonian(H)`
   This ensures that the block-encoded matrix has singular values in $[-1,1]$. $\boldsymbol{H_{norm}}$ is our new hamiltonian matrix.

2) ~~We find polynomial by $\boldsymbol{P}$ by `compute_phase_angles(degree, method="sym_qsp")`~~
   Target polynomial is a Chebyshev polynomial of degree `degree`. What we get returned is an array *p* of length d (index i represents the degree), where *p*[i] is the coefficient.

3) We construct a unitary $\boldsymbol{U}$ with $\boldsymbol{H_{norm}}$ in the top left (aka block encoding) by running:
   `block_encoding = qml.BlockEncode(H_norm, wires=range(required_qubits))`

4) Step 3), in addition, sets up $|0\rangle^{\otimes n}$, where *n* = `required_qubits` is $\lceil \log_2(2m) \rceil$, given $\boldsymbol{H_{norm}}$ is a m x m matrix.

5) `compute_phase_angles` takes $\boldsymbol{P}$ and uses the QSP solver to find a sequence of phases $\{\phi_k\}_{k=0}^{K-1}$ such that the QSVT product evaluates to $\boldsymbol{P}$

6) The list `projectors = [qml.PCPhase(phi_k, ...)]` is the code-level handle that QSVT uses to instantiate and place those projector-controlled phase gates in the right order.

7) The full alternating product is represented by the single `qml.QSVT` call, which builds:

$$\prod_k \left(\tilde{\Pi}_{\phi_k} U^\dagger \Pi_{\phi_{k-1}} U \right)$$

5 References

format properly later

Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics

https://dl.acm.org/doi/pdf/10.1145/3313276.3316366

Intro to QSVT

https://pennylane.ai/qml/demos/tutorial_intro_qsvt