Operator Decomposition via Lax dynamics and Cartan decomposition

Thomas Steckmann
Department of Mathematics, North Carolina State University and
Department of Physics, North Carolina State University

Moody T. Chu
Department of Mathematics, North Carolina State University

Digital quantum computing relies on compiling complex dynamics and operators to a limited number of local operations which can be physically implemented on quantum hardware. As such operators are contained in the Lie group $SU(2^n)$, one method of compiling arbitrary operators relies on a Cartan decomposition of the corresponding $\mathfrak{su}(2^n)$ Lie algebra. Here we present a method of numerically computing such a decomposition for a limited set of unitaries generated by a target Hamiltonian operator by mapping the parameter optimization problem to a parameter flow via Lax dynamics. We present a general formula for such an optimization, demonstrate the implementation, and present open questions and limitations of the proposed methodology.

I. INTRODUCTION

Research into quantum computing algorithms centers on developing and refining applications of quantum computers with exponential speed-ups over classical algorithms. While less powerful algorithms exist and may provide meaningful speedups to problems in the long term, the best chance at demonstrating the usefulness of a modern quantum computer centers on the types of computational problems which are the most challenging for classical systems. Among the class of problems which promise enormous speed-ups on quantum computers is Hamiltonian simulation, or studying the dynamics of a quantum system defined by some initial state and an evolution of that state according to a Hamiltonian operator. Representing the quantum state of many interacting particles using classical resources is exponentially hard, as is operating on the state. Quantum computers prepare and store a quantum state in a register of quantum bits (qubit), which can be operated on using a sequence of local operations called quantum gates. A key problem in Hamiltonian simulation is to define a discrete sequence of gates which evolve the register of quantum bits according the the Hamiltonian of the problem we wish to study. For modern quantum computers, in the so called Noisy Intermediate Scale Quantum (NISQ) era, small errors called noise are introduced in the run-time of the quantum computer - such as through interactions with the environment and imprecise implementations of the gate operations.[1] Implementing algorithms requires minimizing the number of gates used to construct the desired dynamics.

For near term quantum computers, fast-forwarding has emerged as a means of simulating quantum systems over very long time scales. Standard simulation methods, such as the Trotter-Suzuki formula, require gate sequences which have lengths increasing with the desired simulation time. Fast-forwarding algorithms allow for simulating a quantum system with, in most cases, a fixed number of gates to simulate for any target simulation time. However, the cost is that the fixed number of gates, in the general case, increases exponentially with the size of the system we aim to simulation. For near term algorithms on only a few qubits, these simulation methods are promising due to robustness to noise and the potential for approximate methods with relatively low gate overheads.[2–4]

Cartan decomposition within quantum computing originally investigated methods in control theory and general unitary matrix synthesis using single and two-qubit gates.[5–7] However, especially in the case of general unitary synthesis, the constructive algorithms are prohibitively expensive to implement on quantum hardware and the specific gate sequence in the decomposition is difficult to compute classically. As shown by Kökcü, et. al, Cartan decomposition can be applied to the problem of Hamiltonian simulation with a less-than-worst-case circuit depth such as polynomial scaling with number of qubits. In this work, we expand on work by Sá Earp and Pachos and Kökcü, et. al by developing an algorithm to find the specific decomposition by mapping the optimization problem to the equation for Lax dynamics.[8, 9] Rather than using a gradient descent based optimizer, we present a constructive algorithm to explicit build a system of first order differential equations which can be numerical integrated to solve for the specific rotation angles within the decomposition.

II. BACKGROUND

A. Group Notations

This work considers the compact semi-simple Lie group $SU(2^n)$ and the corresponding Lie algebra $\mathfrak{su}(2^n)$. We designate Lie algebras using lowercase fraktur font ($\mathfrak{g}$) and Lie groups using uppercase bold lettering ($\mathbf{G}$). Elements in the algebra are represented using lowercase lettering (g), and uppercase lettering represented group elements (G). Indexed algebraic elements define an ordered

basis (g_i). These relationships can be expressed as:

$$g \in \mathfrak{g},\ e^g = G \in \mathbf{G}$$

Additionally, it is useful here to define the elements of the Pauli Group on n qubits. Over the field of real numbers, we construct a basis g_i of $\mathfrak{g}$ using imaginary Pauli strings $i\mathcal{P}_j$, where $\mathcal{P}$ are Pauli strings composed of tensor products ($\otimes$) of the Pauli matrices and the identity matrix.

$$\sigma_0 = \sigma_I = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \sigma_1 = \sigma_x = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\sigma_2 = \sigma_y = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma_3 = \sigma_z = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

We additionally concatenate tensor products such as $\sigma_x \otimes \sigma_y \otimes \sigma_z \otimes \sigma_I$ as $X_0Y_1Z_2I_3$ where the index refers to the qubit index the elements acts on. Empty indices are assumed to be identity elements, so the above could be equivalently written as $X_0Y_1Z_2$.

We frequently make use of the commutation relations for the Pauli matrices, which are

$$[iX_j, iY_l] = -\delta_{j,l} 2iZ_j$$
$$[iY_j, iZ_l] = -\delta_{j,l} 2iX_j$$
$$[iZ_j, iX_l] = -\delta_{j,l} 2iY_j$$

A meaningful property of this formulation is that the exponential of an imaginary Pauli String can be efficiently implemented on a quantum computer.

B. Hamiltonian Dynamics

In general, a quantum mechanical system can be described using a Hamiltonian operator $\hat{H}$, which governs the behavior of the corresponding system $|\psi\rangle$ according to Schrodinger's equation

$$i\frac{d}{dt_s}|\psi\rangle = \hat{H}|\psi\rangle,$$

to which the solution is the unitary time evolution operator $U(t_s) = e^{it_s\hat{H}}$. We are concerned in this work with physical Hamiltonians which can be mapped to a sum of Pauli strings

$$\hat{H} = \sum_j \gamma_j \mathcal{P}_j$$

For a Hamiltonian on n qubits, $i\hat{H} \in \mathfrak{su}(2^n)$, and is written in term of the Pauli string basis elements $i\mathcal{P}_j$ as described in the Group Notation section.

C. Cartan decomposition

A Cartan decomposition on the Lie algebra $\mathfrak{g}$ is defined by an orthogonal split $\mathfrak{g} = \mathfrak{m} \oplus \mathfrak{k}$ satisfying the commutator relations

$$[\mathfrak{k}, \mathfrak{k}] \subseteq \mathfrak{k} \qquad [\mathfrak{m}, \mathfrak{m}] \subseteq \mathfrak{k} \qquad [\mathfrak{k}, \mathfrak{m}] \subseteq \mathfrak{m}. \tag{1}$$

Additionally, within the set $\mathfrak{m}$ we define a Cartan subalgebra $\mathfrak{h}$ which is a maximal Abelian algebra $\mathfrak{h} \in \mathfrak{m}$. As demonstrated by Sá Earp and Pachos, these sets allow for the following properties:[8]

- $\forall m \in \mathfrak{m}$, $\exists\ K \in e^{\mathfrak{k}}, h \in \mathfrak{h}$ such that $m = KhK^\dagger$
- $e^m = Ke^hK^\dagger$

As observed by Kökcü, et. al., many problems of interest in physics allow for a Cartan decomposition such that $i\hat{H} \in \mathfrak{m}$. Thus, $U(t_s) = e^{it_s\hat{H}} \in \mathbf{M}$, and it is possible to find a decomposition $U(t) = KH(t_s)K^\dagger$. We additionally define the notion of the Hamiltonian algebra $\mathfrak{g}(\hat{H})$ as the minimal algebra which contains the basis elements used to define the Hamiltonian. This is equivalently the algebra spanned by the unit basis which can be generated by the nest commutators of the Hamiltonian basis elements, and is outlined in more detail in reference [9]. It will furthermore be convenient to define an additional subspace $\tilde{\mathfrak{m}} = \text{span}\{m_j \mid \forall\ m_j \in \mathfrak{m} \text{ and } m_j \notin \mathfrak{h}\}$.

D. Lax dynamics

Through outlining the structure of the Lax dynamics problem, the comparison to the Cartan decomposition problem will become apparent. Lax dynamics concerns solutions to a differential equation of the form [10]

$$\begin{aligned} \frac{d}{dt}X(t) &= [X(t), b_1(X(t))] \\ X(0) &= X_0 \end{aligned} \tag{2}$$

where b_1 is some operator on $X(t)$, and there exists some complementary operator b_2 for which $b_1 + b_2 = I$. Through defining the operator b_1, we can tailor the Lax dynamics to our specific problem.

We are also interested in the equations for the dynamics of the parameter $g_1(t)$, which is given by

$$\begin{aligned} \frac{d}{dt}g_1(t) &= g_1(t)b_1(X(t)) \\ g_1(0) &= I \end{aligned} \tag{3}$$

Notably, the parameter $g_1(t)$ permits a similarity relationship between X_0 and $X(t)$:

$$X(t) = g_1(t)^{-1}X_0g_1(t) \tag{4}$$

In the case of Cartan decomposition, given $b_1(X(t)) \in \mathfrak{k}$ and $X_0 \in \mathfrak{m}$, Eq. 4, we have that $g_1(t) \in \mathbf{K}$ and $X(t) \in \mathfrak{m}$, giving a more familiar form of the equation that we are interested in solving:

$$m(t) = K^{-1}m_0K \tag{5}$$

The difference is only that we are interested in finding a solution such that $m(t)$ converges to an element in $\mathfrak{h}$: as we will demonstrate below, this can be achieved through careful choice of b_1.

III. RESULTS: ALGORITHM

A. Overview

The objective of the algorithm described below is to force the convergence of the lax dynamical flow into the $\mathfrak{h}$ Cartan subalgebra in a method that can be computed efficiently and is deterministic. The key steps require defining the operator $b_1(X(t))$ in the problem in order to find the equations for the Lax dynamics flow, and approximating the solution to equation 3 using a power series expansion to find the corresponding value of $g_1(t)$ to complete the decomposition. A preliminary implementation of the algorithm is presented in the Cartan Quantum Synthesizer Package.[11] Additional refinements of the above code are required to increase the efficiently and accuracy of the optimizer, as discussed in the conclusion.

B. Example Problem on Two Qubits

To illustrate the usage of Lax dynamics, we provide an example of constructing the Cartan decomposition of the time evolution operator for a simple example in physics called the transverse field Ising model. In this work is is sufficient to provide the Hamiltonian operator for the two-qubit case

$$i\hat{H} = \gamma_1 iZ_0Z_1 + \gamma_2 iI_0X_1 + \gamma_3 iX_0I_1. \tag{6}$$

From the Hamiltonian elements indexed as $\hat{H}_i$, we generate the Hamiltonian Algebra $\mathfrak{g}(\hat{H})$ using the following elements:

$$\begin{array}{lll} X_0I_1, & I_0X_1, & Z_0Y_1, \\ Z_0Z_1, & Y_0Y_1, & Y_0Z_1 \end{array}$$

which permits a Cartan decomposition and Cartan subalgebra spanned by

$$\begin{aligned} \mathfrak{m} &= \text{span}\{iZ_0Z_1, iY_0Y_1, iX_0I_1, iI_0X_1\} \\ &= \text{span}\{\tilde{m}_1, \tilde{m}_2, h_1, h_2\} \\ \mathfrak{h} &= \text{span}\{iX_0I_1, iI_0X_1\} = \text{span}\{h_1, h_2\} \\ \tilde{\mathfrak{m}} &= \text{span}\{iZ_0Z_1, iY_0Y_1\} = \text{span}\{\tilde{m}_1, \tilde{m}_2\} \\ \mathfrak{k} &= \text{span}\{iZ_0Y_1, iY_0Z_1\} = \text{span}\{k_1, k_2\} \end{aligned}$$

C. Step 1

Compute the commutator table for the basis elements $[\mathfrak{m}, \mathfrak{k}]$ We will reference this commutator table to construct the differential equations we use to solve the Lax dynamics, and additionally to construct the operator b_1 to ensure convergence of the solution. We represent the commutators using a table of structure constants which describe the result of the commutation $[A, B] = f^C_{A.B}C$. For example

$$\begin{aligned} [iX_0I_1, iZ_0Y_1] &= iX_0I_1 \cdot iZ_0Y_1 - iZ_0Y_1 \cdot iX_0I_1 \\ &= -iY_0Y_1 - iY_0Y_1 = -2iY_0Y_1. \end{aligned}$$

1. Example on Two Qubits

We describe the commutator as $[iX_0I_1, iZ_0Y_1] = f^{Y_0Y_1}_{X_0I_1, Z_0Y_1} iY_0Y_1 = -2iY_0Y_1$ or, equivalently, $[h_1, k_1] = f^{\tilde{m}_2}_{h_1,k_1}\tilde{m}_2 = -2\tilde{m}_2$. For every element $h_j, \tilde{m}_l, k_n$, this produces TABLE I.

D. Step 2

Construct the general operator b_1. For now, we use general coefficients κ_n, which we will later provide explicit forms for. The general definition of b_1 is constructed using $\Sigma_n\kappa_n k_n$ as follows:

$$b_1(X(t)) = \Sigma_n \kappa_n k_n \tag{7}$$

This satisfies $b_1(X(t)) \in \mathfrak{k}$, and thus we enforce $[X(t), b_1(X(t))] \in \mathfrak{m}$ and our solution for $X(t)$ will at all time remain in $\mathfrak{m}$ (following from the commutator properties of Cartan decomposition with an initial $X_0 = i\hat{H}$).

E. Step 3

Compute the full commutator $[X(t), b_1(X(t))]$. This step is easily performed using the commutator table for $[\mathfrak{m}, \mathfrak{k}]$ constructed above. The general form is as follows, and allows for enumerating the differential equation for the coefficient of each basis vector of $X(t)$ in $\mathfrak{m}$. We define the generic element $X(t)$ as a sum on the component in $\mathfrak{h}$ and in $\tilde{\mathfrak{m}}$ as $X(t) = \Sigma_j\alpha_j h_j + \Sigma_l\tilde{\beta}_l\tilde{m}_l$.

$$\begin{aligned} [X(t), b_1(X(t))] &= [\Sigma_j\alpha_j h_j + \Sigma_l\tilde{\beta}_l\tilde{m}_l, \Sigma_n\kappa_n k_n] \\ &= [\Sigma_j\alpha_j h_j, \Sigma_n\kappa_n k_n] + [\Sigma_l\tilde{\beta}_l\tilde{m}_l, \Sigma_n\kappa_n k_n] \\ &= \Sigma_j\Sigma_n\alpha_j\kappa_n[h_j, k_n] + \Sigma_l\Sigma_n\tilde{\beta}_l\kappa_n[\tilde{m}_l, k_n] \\ &= \Sigma_j\Sigma_n\alpha_j\kappa_n f^{\tilde{m}_p}_{h_j,k_n}\tilde{m}_p \\ &\quad + \Sigma_j\Sigma_n\tilde{\beta}_l\kappa_n f^{m_q}_{\tilde{m}_l,k_n} m_q \end{aligned} \tag{8}$$

We can now rearrange the terms in Eq. 8 to group similar basis elements. In the worst case, for each value of h_i and $\tilde{m}_i$, this provides equations of the form

$$\begin{aligned} \dot{X(t)} &= \Sigma_i(\Sigma_l\Sigma_n\tilde{\beta}_l\kappa_n f^{h_i}_{\tilde{m}_l,k_n})h_i \\ &\quad + \Sigma_i(\Sigma_l\Sigma_n\tilde{\beta}_l\kappa_n f^{\tilde{m}_i}_{\tilde{m}_l,k_n} + \Sigma_j\Sigma_n\alpha_j\kappa_n f^{\tilde{m}_i}_{h_j,k_n})\tilde{m}_i \end{aligned} \tag{9}$$

We note that, according to LEMMA 3.1, we constrain the commutators ranges in Eq. (8)

$\mathfrak{m}$ \ $\mathfrak{k}$	$k_1 = iZ_0Y_1$	$k_2 = iY_0Z_1$
$h_1 = iX_0I_1$	$f^{Y_0Y_1}_{X_0I_1,Z_0Y_1} iY_0Y_1 = f^{\tilde{m}_2}_{h_1,k_2}\tilde{m}_2 = -2\tilde{m}_2$	$2\tilde{m}_1$
$h_2 = iI_0X_1$	$2\tilde{m}_1$	$-2\tilde{m}_2$
$\tilde{m}_1 = iZ_0Z_1$	$-2h_2$	$2h_1$
$\tilde{m}_2 = iY_0Y_1$	$2h_1$	$-2h_2$

TABLE I. Full commutator table for the two qubit example

1. Example on Two Qubits

For out example problem, we fully define the differential equation by first enumerating the commutator and then grouping the resulting elements of the equation for $\dot{X(t)}$ by basis vector.

$$\begin{aligned}
\dot{X}(t) &= [\alpha_1 h_1 + \alpha_2 h_2 + \tilde{\beta}_1\tilde{m}_1 + \tilde{\beta}_2\tilde{m}_2, \kappa_1 k_1 + \kappa_2 k_2] \\
&= \alpha_1\kappa_1 f^{\tilde{m}_2}_{h_1,k_1}\tilde{m}_2 + \alpha_1\kappa_2 f^{\tilde{m}_1}_{h_1,k_2}\tilde{m}_1 + \alpha_2\kappa_1 f^{\tilde{m}_1}_{h_2,k_1}\tilde{m}_1 + \alpha_2\kappa_2 f^{\tilde{m}_2}_{h_2,k_2}\tilde{m}_2 \\
&+ \tilde{\beta}_1\kappa_1 f^{h_2}_{\tilde{m}_1,k_1}h_2 + \tilde{\beta}_1\kappa_2 f^{h_1}_{\tilde{m}_1,k_2}h_1 + \tilde{\beta}_2\kappa_1 f^{h_1}_{\tilde{m}_2,k_1}h_1 + \tilde{\beta}_2\kappa_2 f^{h_2}_{\tilde{m}_2,k_2}h_2 \\
&= (\alpha_2\kappa_1 f^{\tilde{m}_1}_{h_2,k_1}\tilde{m}_1 + \alpha_1\kappa_2 f^{\tilde{m}_1}_{h_1,k_2}\tilde{m}_1) + (\alpha_1\kappa_1 f^{\tilde{m}_2}_{h_1,k_1}\tilde{m}_2 + \alpha_2\kappa_2 f^{\tilde{m}_2}_{h_2,k_2}\tilde{m}_2) \\
&+ (\tilde{\beta}_1\kappa_2 f^{h_1}_{\tilde{m}_1,k_2}h_1 + \tilde{\beta}_2\kappa_1 f^{h_1}_{\tilde{m}_2,k_1}h_1) + (\tilde{\beta}_1\kappa_1 f^{h_2}_{\tilde{m}_1,k_1}h_2 + \tilde{\beta}_2\kappa_2 f^{h_2}_{\tilde{m}_2,k_2}h_2)
\end{aligned}$$

Given $X(t) = \Sigma_j\alpha_j h_j + \Sigma_l\tilde{\beta}_l\tilde{m}_l$ is an element of $\mathfrak{m}$, we instead write these equations as:

$$\dot{\alpha}_1 = \tilde{\beta}_2\kappa_1 f^{h_1}_{\tilde{m}_2,k_1} + \tilde{\beta}_1\kappa_2 f^{h_1}_{\tilde{m}_1,k_2} \tag{10}$$

$$\dot{\alpha}_2 = \tilde{\beta}_1\kappa_1 f^{h_2}_{\tilde{m}_1,k_1} + \tilde{\beta}_2\kappa_2 f^{h_2}_{\tilde{m}_2,k_2} \tag{11}$$

$$\dot{\tilde{\beta}}_1 = \alpha_1\kappa_2 f^{\tilde{m}_1}_{h_1,k_2} + \alpha_2\kappa_1 f^{\tilde{m}_1}_{h_2,k_1} \tag{12}$$

$$\dot{\tilde{\beta}}_2 = \alpha_1\kappa_1 f^{\tilde{m}_2}_{h_1,k_1} + \alpha_2\kappa_2 f^{\tilde{m}_2}_{h_2,k_2} \tag{13}$$

F. Step 4

Assign the values of κ_n to ensure certain $\dot{\alpha}_j$ are monotonic. Our aim is to ensure that the equations for certain $\dot{\alpha}_j$ are strictly increasing (or decreasing). We enforce this by carefully choosing values of κ_n to maintain a constant sign. The 0th index equation $\dot{\alpha}_0$ is set initially to be monotonic. This forces the value of α_0 to converge, after which the values of $\tilde{\beta}$ in the equation must converge to zero. However, there may exists other equations for $\dot{\alpha}_j$ that does not fully overlap with α_0, i.e. there are values of $\tilde{\beta}_l$ which are not included in the equation for $\dot{\alpha}_0$. In these case, we define the previously undefined values of κ_n to ensure that, after α_0 converges, $\dot{\alpha}_j$ is monotonic. Once all values of κ are defined, we have sufficiently constrained the equations to ensure all values of α converge to a constant and all values of $\tilde{\beta}$ converge to 0. The procedure is as follows:

1. Choose the equation for $\dot{\alpha}_0$. The exact ordering of the indices is not significant.
2. For each term $\tilde{\beta}_l\kappa_n f^{h_0}_{\tilde{m}_l,k_n}$, assign the value of $\kappa_n = \tilde{\beta}_l f^{h_0}_{\tilde{m}_l,k_n}$.
3. The equation for $\dot{\alpha}_0$ will now read $\dot{\alpha}_0 = \Sigma_l\Sigma_n|\tilde{\beta}_l|^2|f^{h_0}_{\tilde{m}_l,k_n}|^2$, which is strictly increasing.
4. Repeat steps 1-3 for h_i, $i = 1...|\mathfrak{h}|$ until all values κ are defined. Do not reassign any previously assigned values of κ_n.

1. Example on Two Qubits

Plugging in the values of the structure constants into eqs. (10 - 13) yields

$$\dot{\alpha}_1 = 2\tilde{\beta}_2\kappa_1 + 2\tilde{\beta}_1\kappa_2 \tag{14}$$

$$\dot{\alpha}_2 = -2\tilde{\beta}_1\kappa_1 + -2\tilde{\beta}_2\kappa_2 \tag{15}$$

$$\dot{\tilde{\beta}}_1 = 2\alpha_1\kappa_2 + 2\alpha_2\kappa_1 \tag{16}$$

$$\dot{\tilde{\beta}}_2 = -2\alpha_1\kappa_1 + -2\alpha_2\kappa_2 \tag{17}$$

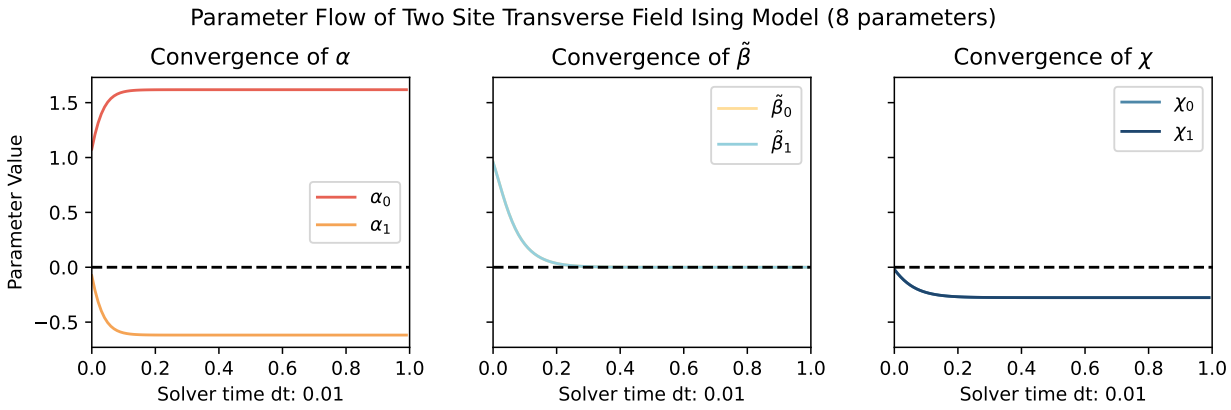

FIG. 1. Convergence behavior on the example two-qubit Transverse-Field Ising Model problem. Note that only one iteration of κ assignments are required. The values of $\tilde{\beta}$ and χ overlap.

And assigning the values of κ gives:

$$\dot{\alpha}_1 = 4|\tilde{\beta}_2|^2 + 4|\tilde{\beta}_1|^2 \tag{18}$$

$$\dot{\alpha}_2 = -4\tilde{\beta}_1\tilde{\beta}_2 + -4\tilde{\beta}_2\tilde{\beta}_1 \tag{19}$$

$$\dot{\tilde{\beta}}_1 = 4\alpha_1\tilde{\beta}_1 + 4\alpha_2\tilde{\beta}_2 \tag{20}$$

$$\dot{\tilde{\beta}}_2 = -4\alpha_1\tilde{\beta}_2 + -4\alpha_2\tilde{\beta}_1 \tag{21}$$

We note here that due the simplicity of the example problem, we only need to use the first 3 steps of the κ assignment algorithm.

G. Step 5

Define the differential equations which allow for solving the value of $g_1(t)$. The general form for the differential equation is

$$\frac{\mathrm{d}g_1(t)}{\mathrm{d}t} = g_1(t)b_1(X(t)) \tag{22}$$

Following from the equations for Lax dynamics, we additionally have $g_1(t) = I$. Thus, because $b_1(X(t)) \in \mathfrak{k}$, the parameter flow for $g_1(t)$ must also be contained within K. We then parameterize $g_1(t)$ using the exponential map of a continuous flow $\chi(t) = \Sigma_p \chi_p(t) k_p \in \mathfrak{k}$.

For $\mathfrak{k}$ which commutes, as it does in our example, the derivative of the exponential map of $exp(\Sigma_p\chi_p(t)k_p) = \Sigma_p\dot{\chi}_p(t)k_p$. However, in the general case, the derivative of the exponential $\frac{d}{dt}exp(\chi(t))$ is given by [12]

$$\frac{d}{dt}e^{\chi(t)} = e^{\chi(t)}\left(\frac{1-e^{-ad_{\chi(t)}}}{ad_{\chi(t)}}\right)\frac{d(\chi(t))}{dt} \tag{23}$$

We approximate the central term as a power series [12] up to some value $p = N$ for which the approximation is sufficiently close.

$$\left(\frac{1-e^{-ad_{\chi(t)}}}{ad_{\chi(t)}}\right) = \sum_{p=0}^{\infty}\frac{(-1)^k}{(k+1)!}(ad_{\chi(t)})^k \tag{24}$$

The necessary bounds on the value of N are not provided here, except that the coefficient of equation 24 scales as $2^p/(p+1)!$ which diminishes rapidly for sufficiently large $N > 10$.

Computing the numerical value of equation 24 requires computing the commutator table $[\mathfrak{k}, \mathfrak{k}]$ to find the corresponding structure constants. To operate efficiently on the operator $ad_{\chi(t)}$, we compute the adjoint representation R of $\Sigma_p\chi_p(t)k_p$ as $R_{j,l} = \chi_i f^l_{i,j}$. As the indices j, k uniquely determine i, we compute the structure constants $f^i_{j,l} = f^l_{i,k}$ (which are cyclic permutations). The resulting matrix R is $|\mathfrak{k}|\times|\mathfrak{k}|$ can be operated on using standard matrix operators. In particular, we compute the expanded form of the operator equation 24 as R^N. Combined with the power expansion, we now write equation 22 as

$$g_1(t)R^N(\Sigma_p\dot{\chi}_p(t)k_p) = g_1(t)(\Sigma_n\kappa_n k_n) \tag{25}$$

which can be solved using a linearly implicit ODE integrator.

1. *Example on Two Qubits*

The example chosen has a commuting set of basis vector in $\mathfrak{k}$, so does not provide a meaningful example of the implementation of the algorithm. Future work will implement a more general example to demonstrate this aspect.

H. Step 5

Solve the resulting differential equations. We have fully defined our first order equations for $\dot{\alpha}_j, \dot{\tilde{\beta}}_l$, and $\dot{\chi}_p$. This work does not explore the optimal solvers on the resulting problem. For the examples shown below, the dopri5 integrator in the `scipy.integrate`[13] is used along with matrix inversion of R^n.

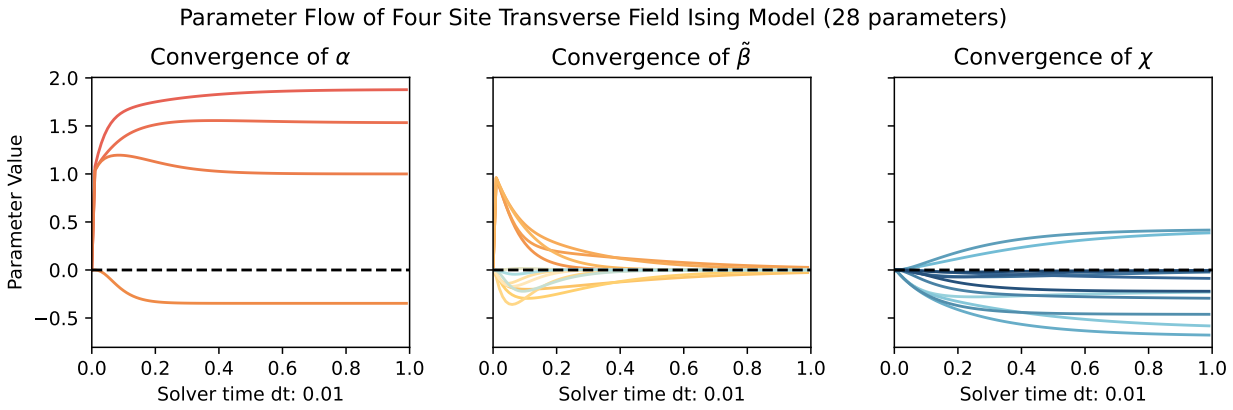

FIG. 2. Convergence behavior on the example four-qubit Transverse-Field Ising Model problem with 28 paramters. Multiple rounds of assignment for κ are needed, leading to α_0 converging before α_1. Darker red gradients correspond to the first assigned equations of $\dot{\alpha}_i$

1. *Example on Two Qubits*

For the provided example problem, only on iteration of the κ assignment protocol is required, and the $\mathfrak{k}$ basis elements commute. The results for the optimization routine with $N = 15$ and a stepsize of $dt = 0.01$ is shown in Figure 1. The convergence behavior can be clearly seen in the steps from monotonic increasing equation defined for α_0 and the decreasing function for α_1. In this case, the coefficients $\tilde{\beta}_0, \tilde{\beta}_1$, and χ_0, χ_1 are equal. To demonstrate the case with a non-commuting case, Figure 2 shows the convergence behavior over 28 parameters which requires multiple assignments of κ. In this case, the the darkest shades are assigned first, and the first assigned value of α_0 convergences rapidly while the remaining values only begin converging after the previously assigned values. Numerically verifying the computation demonstrates arbitrary accuracy for implementing the original exponential $e^{iHt_s} = U(t_s)$ through increasing N and reducing the step size dt.

IV. CONCLUSION

This work presents a proof of concept for the defined algorithm, which has been integrated into existing code to produce Cartan decompositions for test systems in condensed matter physics. Key remaining challenges include verifying the convergence accuracy of the differential equation solver, addressing observed numerical instability for long convergence times, eliminating the need to invert R^n, and reworking key components of the algorithm to reduce run-time, such as in calculating and storing commutator tables. Additionally, we hope this work leads to either of three key practical applications: optimizing over parameterized Hamiltonians (preserving variables in the integrator), implementing a general Cartan decomposition on a generic element $g \in \mathfrak{g}$ (rather than on $m \in \mathfrak{m}$, and solving for $g_1(t)$ in canonical coordinates of the second kind (rather than the exponential map which is the first kind). Missing these applications, the algorithm presents an application of Lax dynamics as a means of reducing an element into a corresponding minimal subspace without finding the corresponding eigenvalues of a high dimensional system. Refinements of the existing implementation will demonstrate if this method allows for understanding high dimensional systems with fewer resources than traditional diagonalization approaches.

V. LEMMAS

Lemma 3.1: $[h_i, k_j] \in \tilde{\mathfrak{m}}_\mathfrak{l}$**.** Assume $[h_i, k_j] \in \mathfrak{h}_\mathfrak{j}$. The first case $j = i$ implies $[h_i, h_i] \neq 0$ which is a contradiction. The second case $i \neq j$ implies $h_i k_j = f^l_{ij} h_j$. As the Lie algebra is semi-simple and compact, the structure constant are antisymmetric this implies $[h_i, h_j] \neq 0$ which contradicts the definition of $\mathfrak{h}$.

Lemma 4.1: X(t) is bounded. X(t) is similar to the Hamiltonian operator which, for a finite dimensions 2^n has bounded and finite eigenvalues. Thus, $|X(t)|$ is bounded. (This Lemma needs refinement).

[1] J. Preskill, en-GBQuantum computing in the nisq era and beyond, Quantum **2**, 79 (2018).
[2] T. Steckmann, T. Keen, A. F. Kemper, E. F. Dumitrescu, and Y. Wang, Simulating the mott transition on a noisy digital quantum computer via cartan-based fast-forwarding circuits, arXiv:2112.05688 [cond-mat, physics:quant-ph] (2021), arXiv: 2112.05688.
[3] J. Gibbs, Z. Holmes, M. C. Caro, N. Ezzell, H.-Y. Huang,

L. Cincio, A. T. Sornborger, and P. J. Coles, Dynamical simulation via quantum machine learning with provable generalization, arXiv:2204.10269 [quant-ph] (2022), arXiv: 2204.10269.

[4] J. Gibbs, K. Gili, Z. Holmes, B. Commeau, A. Arrasmith, L. Cincio, P. J. Coles, and A. Sornborger, Long-time simulations with high fidelity on quantum hardware, arXiv:2102.04313 [quant-ph, stat] (2021), arXiv: 2102.04313.

[5] B. Drury and P. Love, enConstructive quantum shannon decomposition from cartan involutions, Journal of Physics A: Mathematical and Theoretical **41**, 395305 (2008).

[6] F. Albertini and D. D'Alessandro, enControllability of symmetric spin networks, Journal of Mathematical Physics **59**, 052102 (2018), arXiv: 1803.06689.

[7] N. Khaneja and S. J. Glaser, enCartan decomposition of su(2n) and control of spin systems, Chemical Physics **267**, 11–23 (2001).

[8] H. N. S. Sá Earp and J. K. Pachos, enA constructive algorithm for the cartan decomposition of su(2n), Journal of Mathematical Physics **46**, 082108 (2005).

[9] E. Kökcü, T. Steckmann, J. K. Freericks, E. F. Dumitrescu, and A. F. Kemper, Fixed depth hamiltonian simulation via cartan decomposition, arXiv:2104.00728 [cond-mat, physics:quant-ph] (2021), arXiv: 2104.00728.

[10] M. T. Chu and L. K. Norris, Isospectral flows and abstract matrix factorizations, SIAM Journal on Numerical Analysis **25**, 1383–1391 (1988).

[11] T. Steckmann and E. Kökcü, *Cartan Quantum Synthesizer* (2021).

[12] W. Rossman, *Lie Groups - An Introduction Through Linear Groups*, Oxford Graduate Texts in Mathematics (Oxford Science Publications, 2002).

[13] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods **17**, 261 (2020).