Parametric matrix models

Morten Hjorth-Jensen

Department of Physics and Center for Computing in Science Education, University of Oslo, Norway

qGap Octover 7, 2025

Outline

What is this talk about?

Trotterization and time-evolution

The three steps of quantum control

1. **Quantum correlations:** Understanding and preparing an initial state
2. **Quantum dynamics:** Controlled evolution towards a desired final state
3. **Quantum measurements:** Measuring and characterizing the final state

Motivation: Non-commuting Exponentials

- In quantum mechanics and Lie theory, we often encounter operators X and Y that do not commute ($[X, Y] \neq 0$).
- We want to find an effective operator Z such that: e^X $e^Y =$ e^Z $, for$X,Y$inaLiealgebra.If$XandY$commute, thensimply$Z=X+Y$.Ifnot,$Z
- **BCH Formula:** $Z = \log(e^X e^Y)$ is given by an infinite series in X, Y and their commutators. It provides a systematic expansion to combine exponentials of non-commuting operators .
- **Use Cases:** Combines two small transformations into one. Fundamental in connecting Lie group multiplication with Lie algebra addition, time-evolution with split Hamiltonians, etc.

Recall: Commutators and Lie Algebra

- The **commutator** of two operators is $[X, Y] = XY - YX$. It measures the failure to commute.
- For a Lie algebra (e.g. operators in quantum mechanics), commutators of algebra elements remain in the algebra.
- The BCH formula asserts Z can be expressed entirely in terms of X, Y, and nested commutators like $[X, [X, Y]]$, $[Y, [X, Y]]$, etc. – no other independent products appear .
- Notation: It's useful to denote $\mathrm{ad}_X(Y) := [X, Y]$. Then nested commutators are iterated adjoint actions (e.g. $\mathrm{ad}_X^2(Y) = [X, [X, Y]]$, etc.).
- We assume familiarity with basic Lie algebra identities (Jacobi identity: $[X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0$) which will simplify nested commutators.

BCH Expansion: First Terms

For $Z = \log\left(e^X e^Y\right)$, the expansion begins:

- The series alternates between symmetric and antisymmetric nested commutators at higher orders .
- All higher-order terms involve nested commutators of X and Y only. No ordinary products without commutators appear (ensuring Z lies in the same Lie algebra) .
- The coefficients $1/2, 1/12, 1/24, \dots$ are fixed numerical values (involving Bernoulli numbers for higher terms). These were first worked out explicitly by Dynkin (1947) in general .

Series Characteristics

- The BCH series is generally infinite. In most cases, there is **no closed-form finite expression** for Z in terms of a finite number of terms .
- Each increasing order introduces more deeply nested commutators. For example:
    - 1st order: $X + Y$
    - 2nd order: $[X, Y]$
    - 3rd order: $[X, [X, Y]]$, $[Y, [X, Y]]$
    - 4th order: $[Y, [X, [X, Y]]]$, $[X, [Y, [Y, X]]]$, etc.
- The number of independent commutator terms grows rapidly with order. (All such terms up to 6th order are listed in the literature , but it becomes cumbersome beyond a few orders.)
- Fortunately, many practical scenarios require only the first few terms for approximation.
- If X and Y are “small” (e.g. small matrices or small time-step in evolution), the series converges and truncating after a few terms can give a good approximation .

Derivation: Outline (up to Third Order)

- **Method:** Compare power series of $e^X e^Y$ and e^Z and solve for Z order-by-order .
- Expand both sides: $e^X e^Y = I + X + Y + \frac{1}{2}(X^2 + XY + YX + Y^2) + \frac{1}{6}(X^3 + \cdots) + \cdots e^Z = I + Z + \frac{1}{2}Z^2 + \frac{1}{6}Z^3 + \cdots whereZ$ = X + Y + $A_2 + A_3 + \cdots$ (with A_n = terms of order n in X, Y).
- **First order:** Match linear terms: $Z^{(1)} = X + Y$. So far $Z = X + Y$.
- **Second order:** The $e^X e^Y$ expansion has $\frac{1}{2}(XY + YX)$ at order 2. Meanwhile e^Z gives $\frac{1}{2}(X + Y)^2 = \frac{1}{2}(X^2 + XY + YX + Y^2)$. The extra X^2 and Y^2 terms match on both sides, but $XY + YX$ vs $XY + YX$ is already present. However, note that $XY + YX$ cannot simplify to $2XY$ unless $XY = YX$. The discrepancy appears at this order .
- Thus, we postulate Z has a second-order correction $A_2 = \frac{1}{2}[X, Y]$ to account for the difference: XY + YX = $(X+Y)^2 - X^2 - Y^2 = XY + YX, butincluding A_2$ in Z yields new cross terms when squaring Z:

Special Case: Commutator is Central

- If $[X, Y]$ commutes with both X and Y (i.e. $[X, Y] = c, I$, a scalar multiple of the identity), **all higher-order commutators vanish.** In this case the BCH series *terminates* after the second term .
- Then the exact result is: Z = X + Y + $1\frac{}{2[X,Y]\,,and no further corrections are needed.}$
- This scenario occurs often in quantum mechanics when $[X, Y]$ is a c-number (for example, if X and Y are operators proportional to canonical variables p and q).
- **Example:** Position and momentum operators satisfy $[x, p] = i\hbar I$. Thus, $e^{\frac{i}{\hbar}ax}\, e^{\frac{i}{\hbar}bp} = \exp\left(\frac{i}{\hbar}(ax + bp) + \frac{i}{2\hbar}ab[x, p]\right) = e^{\frac{i}{\hbar}(ax+bp+\frac{1}{2}ab\, i\hbar)}$ *, yielding a phase factor* $e^{-iab/2}$ times $e^{\frac{i}{\hbar}(ax+bp)}$. (This is the basis of the Weyl representation in quantum mechanics.)
- Another example: For harmonic oscillator ladder operators $[a, a^\dagger] = 1$, the displacement operator factorization $e^{\alpha a} e^{-\alpha^* a^\dagger} = e^{-|\alpha|^2/2} e^{-\alpha^* a^\dagger + \alpha a}$ follows from BCH truncation.

Application: Lie Groups and Lie Algebras

- The BCH formula formalizes how group multiplication near the identity corresponds to addition in the Lie algebra plus commutator corrections .
- If X and Y are infinitesimal generators (Lie algebra elements), e^X and e^Y are group elements. Their product $e^X e^Y$ can be expressed as e^Z with Z in the Lie algebra, ensuring closure of the group-law in algebra terms.
- This underpins the Lie group–Lie algebra correspondence: the complicated group law (when the group is nonabelian) is captured by a formal power series in the algebra.
- **Example:** In $SO(3)$ (rotations), let X and Y be two small rotation generators (non-commuting). $e^X e^Y$ is a rotation whose generator Z is given by BCH. Thus, the axis and angle of the combined rotation can be found by computing Z. (In practice, one can compute up to a certain order if X, Y are small.)
- The BCH formula is used to prove properties like $\mathtt{tr}(\log(e^X e^Y)) = \mathtt{tr}(X) + \mathtt{tr}(Y)$ (since commutator contributions

Application: Quantum Time Evolution

- In quantum mechanics, if the Hamiltonian $H = H_1 + H_2$ (two parts that do not commute), the time-evolution operator is $U(t) = e^{-iHt}$. Directly computing $e^{-i(H_1+H_2)t}$ is hard if H_1 and H_2 don't commute.
- Using BCH, we can approximate: $\mathrm{e}^{-i(H_1+H_2)\Delta t} = \exp\Big(- iH_1\Delta t - iH_2\Delta t - \frac{1}{2}[H_1, H_2](\Delta t)^2 + \cdots \Big), sotofirstorderin\Delta t,$ $\mathrm{e}^{-i(H_1+H_2)\Delta t} \approx e^{-iH_1\Delta t}e^{-iH_2\Delta t}, withanerroroforder(\Delta t)^2$ governed by $\frac{-i}{2}[H_1, H_2](\Delta t)^2$.
- **Lie–Trotter Product Formula:** By taking n small time steps, $\Big(\mathrm{e}^{-iH_1t/n}e^{-iH_2t/n}\Big)^n = e^{-i(H_1+H_2)t+O(t^2/n)} \to e^{-i(H_1+H_2)t}$ as $n \to \infty. Inpractice, evenmodest\mathrm{n}yieldsagoodapproximation.$
- Higher-order splitting schemes (e.g. **Suzuki–Trotter decompositions**) use BCH terms cleverly to cancel lower-order errors. For example: $\mathrm{e}^{-i(H_1+H_2)\Delta t} = e^{-iH_1\Delta t/2}e^{-iH_2\Delta t}e^{-iH_1\Delta t/2} + O((\Delta t)^3), whicheliminatesthe\mathrm{O}((\Delta t)^2)$ error by symmetry. BCH provides the systematic way to analyze

Application: Quantum Computing (Hamiltonian Simulation)

- In quantum algorithms, especially for Hamiltonian simulation, we need to implement $U(t) = e^{-i(H_1+H_2+\cdots)t}$ via a sequence of quantum gates.
- The BCH formula underlies the **Trotter-Suzuki product formula** approach: $\mathrm{e}^{-i(H_1+H_2)t} \approx \left(e^{-iH_1 t/n} e^{-iH_2 t/n}\right)^n, whichbecomesexactas \mathrm{n} \to \infty$. For finite n, one incurs a small error.
- The leading error term is $\sim \frac{t^2}{2n}[H_1, H_2]$ from the BCH expansion . By increasing n (more, smaller time slices), the error can be made arbitrarily small, at the cost of more gates.
- Quantum computing implementations often use higher-order BCH-based formulas to reduce error. For instance, the second-order formula above, or higher-order Suzuki expansions, include additional exponentials to cancel out commutator errors up to higher orders.
- **Example:** To simulate $H = H_x + H_y + H_z$ (say parts of a

Symbolic Computation with Sympy

Using `Sympy`, we can manipulate non-commuting symbols and verify the BCH expansion:

```
from sympy.physics.quantum import Commutator, Operator
from sympy import Rational, expand

X, Y = Operator('X'), Operator('Y')



BCH series up to third order:



Z = X + Y
+ Rational(1,2)Commutator(X, Y)
+ Rational(1,12)(Commutator(X, Commutator(X,Y))
+ Commutator(Y, Commutator(Y,X)))

print(Z.expand(commutator=True))
```

Numerical Verification with Numpy

We can also numerically test how including commutator terms improves the approximation. Consider two small 2×2 matrices A and B:

```
import numpy as np
from numpy.linalg import norm
from scipy.linalg import expm  # matrix exponential

A = np.array([[0, 0.1],
[0, 0   ]])
B = np.array([[0,   0 ],
[0.1, 0 ]])



Compute exponentials:



U = expm(A) @ expm(B)              # e^A e^B
```

Worked Example: SU(2) Rotations

As an example in a physics context, consider spin-$\frac{1}{2}$ operators (Pauli matrices). Let $X = i\theta\sigma_x$ and $Y = i\phi\sigma_y$, which generate rotations about the *x*- and *y*-axes by angles θ and ϕ.

- We know $[\sigma_x, \sigma_y] = 2i, \sigma_z$. Thus, $[X, Y] = i^2\theta\phi[\sigma_x, \sigma_y] = -2\theta\phi, \sigma_z$.
- Since σ_z does not commute with σ_x or σ_y, higher commutators will appear (the algebra is nonabelian but finite-dimensional).
- Using the BCH formula up to second order: Z $\approx X + Y + \frac{1}{2}[X, Y] = i\theta\sigma_x + i\phi\sigma_y - \theta\phi\,\sigma_z\,.Thissuggestse^X e^Y \approx$ $\exp\big(i\theta\sigma_x + i\phi\sigma_y - \theta\phi, \sigma_z\big)$ for small angles.
- In fact, the exact combined rotation $e^{i\theta\sigma_x}e^{i\phi\sigma_y}$ equals a rotation about some axis in the *xy*-plane (at third order one would find adjustments to the axis angle). The BCH series can be resummed in this case to give a closed-form result (via $\mathrm{SO}(3)$ formulas for combining rotations).
- The key takeaway: BCH correctly identifies the σ_z component (proportional to $[X, Y]$) in the resultant rotation generator.

Exercises for Practice

1. Derive the BCH formula up to the third order term explicitly:
    1. Start from $\log\big(e^X e^Y\big) = Z = X + Y + A_2 + A_3 + \cdots$. Equate series coefficients to show $A_2 = \frac{1}{2}[X, Y]$ and $A_3 = \frac{1}{12}[X,[X,Y]] - \frac{1}{12}[Y,[X,Y]]$.
    2. (*Hint:* Use the expansion method or the identity $e^X Y e^{-X} = Y + [X,Y] + \frac{1}{2!}[X,[X,Y]] + \cdots$ to assist in the derivation.)
2. For operators A and B such that $[A,B] = cI$ (a central commutator), prove that $e^A e^B = \exp\big(A + B + \frac{1}{2}[A,B]\big)$ exactly. Verify this formula with a concrete example (e.g. 2×2 matrices or simple 2×2 block matrices).
3. Using the first-order Trotter approximation, show that $\mathrm{e}^{(H_1+H_2)\Delta t} = e^{H_1 \Delta t} e^{H_2 \Delta t} + O((\Delta t)^2)\,, anddeterminetheformofthe$O($\Delta t^2$) error term using the BCH expansion. What commutator appears?
4. Consider two 2×2 matrices (for example, Pauli matrices or random matrices) and numerically check the BCH formula:
    1. Compute $Z_{\mathrm{BCH}}^{(n)} = X + Y + \frac{1}{2}[X,Y] + \cdots$ up to nth order for your chosen X, Y.

Summary

- The Baker–Campbell–Hausdorff formula provides a powerful tool to combine exponentials of non-commuting operators into a single exponential. It expresses the result as an infinite series of nested commutators .
- In general, the series is infinite and has no closed form, but truncations are extremely useful for approximate calculations .
- The first few terms ($X + Y$, $\frac{1}{2}[X, Y]$, $\frac{1}{12}[X, [X, Y]]$, $\ldots$) often give insight into how non-commutativity affects combined operations.
- BCH is foundational in Lie theory (connecting local group structure to Lie algebra) and in practical computations in physics (quantum mechanics, quantum computing, optics, etc.) wherever splitting exponentials is needed .
- Through examples and exercises, we saw how BCH explains the error in splitting methods and how it can be checked with symbolic or numeric computation.
- Bottom line: Whenever you see $e^X e^Y$, remember the BCH formula allows you to rewrite it as e^Z with $Z = X + Y + \frac{1}{2}[X, Y] + \cdots$ .This

Quantum Hamiltonian Evolution

The time evolution operator for a quantum system is $U(t) = e^{-iHt}$, solving the Schrödinger equation $i, \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle$. Simulating $U(t)$ is essential in physics and chemistry . Many Hamiltonians are a sum of terms, $H = \sum_j H_j$. If all terms commute, time evolution factorizes exactly: e.g. for $H = H_1 + H_2$ with $[H_1, H_2] = 0$, we have $e^{-i(H_1+H_2)t} = e^{-iH_1 t}e^{-iH_2 t}$. In general H_j *do not* commute, so $e^{-i(H_1+H_2)t} \neq e^{-iH_1 t}e^{-iH_2 t}$. We need to approximate the evolution by alternating the non-commuting pieces in small time slices.

Trotter Product Formula

$\mathrm{e}^{-i(H_1+H_2)t} = \lim_{N\to\infty} \left(e^{-iH_1 \frac{t}{N}} \; e^{-iH_2 \frac{t}{N}} \right)^N .$

This is the basic Trotter-Suzuki decomposition (first-order splitting) . In the infinite step limit, it becomes exact (also known as the Lie product formula or Trotter formula). For finite N, $(e^{-iH_1 t/N} e^{-iH_2 t/N})^N$ approximates $e^{-i(H_1+H_2)t}$ with some error. Using a finite N steps is called Trotterization, and the approximation error can be bounded by a desired ϵ .

Higher-Order Suzuki Decompositions

By symmetrizing the sequence, we can cancel lower-order errors. For example, a second-order formula uses half-step kicks of H_1:$S_2(\Delta) \; = \; e^{-iH_1\Delta/2} \, e^{-iH_2\Delta} \, e^{-iH_1\Delta/2}\,, whichyieldse^{-i(H_1+H_2)\Delta}$ up to $O(\Delta^3)$ error . This symmetric Trotter-Suzuki formula eliminates the $O(\Delta^2)$ term. In general, there are higher even-order formulas (4th, 6th, $\ldots$) that achieve errors $O(\Delta^{p+1})$ for any desired order p . These higher-order decompositions (derived recursively by Suzuki) require more instances of the exponential operators (and sometimes negative-time coefficients) to cancel lower-order commutator errors.

First-Order Trotter Expansion (Derivation)

Using the Baker–Campbell–Hausdorff (BCH) formula, one finds:
$e^A e^B = \exp\Big(A + B + \frac{1}{2}[A, B] + \frac{1}{12}[A, [A, B]] - \frac{1}{12}[B, [A, B]] + \cdots\Big). For$ A=-iH$_1\Delta, ; B = -iH_2\Delta$:
$e^{-iH_1\Delta} e^{-iH_2\Delta} = \exp\Big(- i(H_1 + H_2)\Delta \ - \ \frac{1}{2}[H_1, H_2]\,\Delta^2 +$
$O(\Delta^3)\Big)\ . Thus, a single Trotter step incurs a local error term$-1 $\frac{}{2[H_1, H_2]\Delta^2}$.
The leading error scales as $O(\Delta^2)$, so after $N = t/\Delta$ steps the total error is $O(t, \Delta)$ (first order in Δ).

Second-Order Trotter Expansion (Insight)

In the symmetric product $S_2(\Delta) = e^{-iH_1\Delta/2}e^{-iH_2\Delta}e^{-iH_1\Delta/2}$, the first-order commutator terms cancel out. Intuitively, the $[H_1, H_2]$ error from the first half-step is negated by the second half-step. The leading error in S_2 involves double commutators like $[H_1, [H_1, H_2]]$ (and $[H_2, [H_1, H_2]]$) , which enter at order $O(\Delta^3)$. Thus the second-order scheme has local error $O(\Delta^3)$ (global error $O(\Delta^2)$), a significant improvement over first order.

Example: Single-Qubit $H = X + Z$

Consider a single qubit with Hamiltonian $H = \sigma_X + \sigma_Z$ (Pauli X and Z). Here $[X, Z] = 2iY \neq 0$, so X and Z do not commute . We cannot implement $e^{-i(X+Z)t}$ as one gate, but must Trotterize. Trotter strategy: alternate short rotations about the X-axis and Z-axis. For small Δt, $e^{-iX\Delta t}$ and $e^{-iZ\Delta t}$ are simpler rotations. Repeating them approximates the full evolution $e^{-i(X+Z)t}$. In this case, $e^{-iX\theta} = R_x(2\theta)$ and $e^{-iZ\theta} = R_z(2\theta)$, standard single-qubit rotations . Thus each Trotter step can be directly realized as two orthogonal axis rotations on the qubit.

Trotterization in Python (First-Order)

basicstyle=, `language=Python` import numpy as np from numpy.linalg import norm from scipy.linalg import expm

Define Pauli matrices

X = np.array([[0, 1], [1, 0]]) Z = np.array([[1, 0], [0,-1]]) H = X + Z

t = 1.0 N = 4 dt = t/N

First-order Trotter approximation

U$_t rot = np.eye(2) forkinrange(N) : U_t rot = expm(-1j * X * dt)$@$expm(-1j * Z * dt)$@$U_t rot$

Exact evolution

U$_e xact = expm(-1j * H * t) error = norm(U_t rot - U_e xact) print(error)$

Results: Trotter Approximation Error

With $N = 4$ time steps, the first-order Trotter approximation gives $|U_{\text{trot}} - U_{\text{exact}}| \approx 2.5 \times 10^{-1}$. Increasing to $N = 16$ steps reduces the error to $\sim 6 \times 10^{-2}$. Doubling N roughly halves the error, consistent with $O(1/N)$ convergence (global error $\sim O(t/N)$ for first order). A second-order Trotter scheme yields far smaller error for the same N. For example, at $N = 4$ steps, the symmetric formula gives error $\sim 2.4 \times 10^{-2}$ (about 10× smaller than first order). This faster convergence (error $\sim O(1/N^2)$) is evident in practice. In general, each $e^{-iH_j \Delta t}$ corresponds to a quantum gate implementing that term. In this 1-qubit example, $e^{-iX\Delta t}$ and $e^{-iZ\Delta t}$ are rotations about X and Z axes. Thus the Trotterized $e^{-i(X+Z)t}$ can be realized as a sequence of short rotations, which becomes exact in the limit of fine steps .

Error Scaling Comparison

Error norm versus number of Trotter steps N for first-order (Lie–Trotter) and second-order (symmetric) decomposition of $H = X + Z$. On a log–log plot, the first-order errors (yellow, circles) decrease linearly (slope -1), while second-order errors (red, squares) decrease with slope -2, confirming the $1/N$ vs $1/N^2$ scaling.

Scaling of Trotter Steps with Accuracy

The number of Trotter steps required grows as a function of the simulation time t and desired accuracy ϵ: First order: global error $\sim O(t^2/N)$, so to achieve error ϵ one needs $N = O(t^2/\epsilon)$ steps (gate operations) . Second order: global error $\sim O(t^3/N^2)$, so one needs $N = O!\big((t^3/\epsilon)^{1/2}\big) = O(t^{3/2}/\sqrt{\epsilon})$ steps for error ϵ.
Higher-order Suzuki formulas further reduce the scaling. In practice, there is a trade-off: higher order means more gates per step. One chooses an order that minimizes total error (Trotter error + hardware errors) for a given quantum hardware .

Exercises

1. Use the BCH expansion to show the leading correction term for $U_{\text{trot}}(\Delta) = e^{-iH_1\Delta}e^{-iH_2\Delta}$ is $-\frac{i}{2}[H_1, H_2]\Delta^2$. (Hint: Expand $e^{-iH_1\Delta}e^{-iH_2\Delta}$ to second order in Δ.)
2. Verify that in the second-order formula $S_2(\Delta) = e^{-iH_1\Delta/2}e^{-iH_2\Delta}e^{-iH_1\Delta/2}$, the $[H_1, H_2]$ term cancels out. What commutator(s) govern the leading error term of S_2?
3. Write a Python script (using NumPy) to simulate $U(t) = e^{-iHt}$ for a simple 2×2 Hamiltonian $H = H_1 + H_2$ with and without Trotterization. Compare the norm error $|U_{\text{trot}} - U|$ for different N and for first vs second-order Trotterization.