Agentic automation of end-to-end quantum resource estimation for fault-tolerant quantum chemistry

Mark Jack, Vincent Russo, Michael Souffrant, Yudong Cao
Zapata Quantum

***Abstract*—Determining whether a quantum computer could ever be useful for a given chemistry problem is one of the most expensive questions in the field to answer. A fault-tolerant resource estimate for a single molecule is a months-to-years study for a team of specialists, and the cost is dominated not by computation but by human coordination. That coordination splits into two kinds of toil. The first is *operational*: reconciling mismatched software versions, restarting runs that failed for a triageable reason, and checking that intermediate calculations converged. The second is *algorithmic*: recognizing when a completed result is physically wrong or out of specification, and knowing the prescribed recourse. The DARPA Quantum Benchmarking program, the public reference for such studies, spent roughly three years of analyst effort on a handful of catalyst molecules, most of it on exactly this toil.**

We present a multi-agent pipeline that absorbs both. Specialized agents, backed by validators that encode the domain checks a human analyst would perform by hand, are designed to handle operational failures automatically and to route out-of-specification results to codified algorithmic recourse, under a supervisor that decides when a result is trustworthy enough to proceed. With it we independently reproduce the DARPA benchmarking suite's fault-tolerant resource-estimation workflow end-to-end, from molecular geometry to physical-qubit budget, in hours per molecule with no human in the loop. This changes the economics of the question: evaluating one more candidate molecule costs hours of commodity compute rather than months of scarce specialist time, freeing those specialists for the judgment that agents cannot yet make and making bulk screening feasible. Because the agents encode judgment about the workflow rather than any particular method or machine, the pipeline is designed to extend across classical simulation methods, quantum algorithms, and hardware architectures.

***Index Terms*—quantum resource estimation, fault-tolerant quantum computing, multi-agent systems, workflow automation, agentic orchestration, quantum chemistry, DMRG, benchmarking, cloud deployment**

I. Introduction

Estimating the quantum resources required to simulate a chemically relevant molecule to chemical accuracy is, in current practice, a manual end-to-end study that can take a small team months-to-years per catalyst instance [1], [2]. The DARPA Quantum Benchmarking program produced fault-tolerant resource estimates for ten catalyst instances over roughly three years of analyst effort, with a reference DMRG classical cost on the order of 4×10^5 CPU-hours across the suite [3]. As candidate catalysts and target architectures multiply, this manual path does not scale: each new molecule, each new classical baseline (DMRG, CCSD(T), DFT-driven active spaces), each new fault-tolerant scheme (alternate QPE variants, alternate qubit modalities, alternate quantum error-correction codes) currently demands a fresh study. Most of the cost of each study is coordination rather than computation, and it divides into *operational* toil (environment and dependency failures, triaged reruns) and *algorithmic* toil (recognizing and remediating results that complete but are out of specification); the pipeline is built to absorb both (Section IV).

We present an automated multi-agent pipeline that replicates the 13-stage methodology of [3] end-to-end, from molecular XYZ geometry to the physical-qubit budget required to compute the ground-state energy on a fault-tolerant architecture with a selected QPE variant. A graph-based supervisor routes work to four ReAct [4] sub-agents for molecular structure, active-space selection, DMRG quantum chemistry, and resource estimation. Each sub-agent is backed by stage-specific tool servers that expose PySCF, OpenFermion, Block2 DMRG, and QuantumMAMBO.jl primitives, with extractor nodes parsing tool messages into a typed pipeline state and validator nodes enforcing per-stage domain checks. The pipeline is deployed on a managed compute cluster with parallel-node autoscaling across compute tiers sized per molecule (Section V).

We report end-to-end wall-clock times against the DARPA Quantum Benchmarking catalyst suite. The deployment sizes each molecule to a compute tier by active-space size and atom count (Section V), and end-to-end times range from roughly 1 to 24 hours per molecule; the representative $TS_{1/2}$ transition-state instance (31 orbitals, 44 electrons) finishes in about 72 minutes on a single node (Table I).

The stage decomposition is independent of the method, algorithm, and hardware model at each stage, so the same orchestration is designed to extend to other classical methods, QPE variants, qubit modalities, and error-correction codes (Section VII).

II. Background

A 13-stage workflow [3] takes a molecule from a starting geometry to a fault-tolerant physical-qubit budget. The stages, in execution order, are: (1) obtain geometry, (2) select an active space by automated valence active space (AVAS) [5] on top of a closed-shell or restricted open-shell Hartree-Fock reference computed with PySCF [6], (3) construct the second-quantized Hamiltonian, (4) double-factorize the two-electron integrals to produce a low-rank tensor representation suitable for qubitized phase estimation [7], [8], (5) optionally apply LP-BLISS symmetry shifts to reduce the Hamiltonian 1-norm λ [9], as implemented in QuantumMAMBO.jl [10],

(6) compute a DMRG ground-state energy in the active space with Block2 [11], (7) extract the dominant configuration state function (CSF) overlap $|\gamma|$ as a gate for the QPE shot count, (8) select a QPE variant, (9) set energy accuracy and failure tolerance, (10) establish an error model across hardware, phase estimation, truncation, and rotation synthesis channels, (11) optimize the QPE thresholds by grid search to minimize Toffoli count, (12) compute exact logical resources (Toffoli count, logical qubit count) using OpenFermion's [12] double-factorized QPE cost model, and (13) compute physical resources (physical qubit count, code distance, fault-tolerant runtime) under a surface-code error model [13].

The program applied this workflow to a public suite of ten catalyst instances across three families (Schrock molybdenum nitride, bridged dimolybdenum nitride, and molybdenum-pincer intermediates and transition states).

III. System architecture

The pipeline is built as an asynchronous state graph with a hub-and-spoke supervisor topology (Figure 1). The supervisor is a deterministic Python node, not an LLM call: it reads a `completed_steps` field on the typed pipeline state, computes the next canonical stage from the 13-stage execution order described in Section II, and routes to the corresponding sub-agent, or terminates when the final stage is complete. This keeps the routing layer auditable and cheap; every LLM call sits inside a sub-agent that owns one or two adjacent stages.

The four ReAct sub-agents own contiguous stage blocks (Figure 1): molecular structure (Stage 1), quantum chemistry (Stages 2–5: AVAS, Hamiltonian construction, double factorization, optional LP-BLISS), DMRG (Stages 6–7, with SU(2) symmetry and CSF-overlap extraction), and resource estimation (Stages 8–13). A stronger LLM handles sub-agent reasoning and a lighter one handles per-tool reasoning.

The tool servers wrap PySCF, OpenFermion, Quantum-MAMBO.jl, and Block2, plus a BenchQ-equivalent [14] physical-resource estimator and an AutoCCZ [15] factory sweep at Stage 13.

Each stage writes typed fields onto a shared state through extractor nodes, and validators enforce domain checks (Pauli exclusion, finite-and-negative energies, the CSF-overlap gate on the shot count, surface-code odd-distance parity). An optional human-in-the-loop interrupt after Stage 1 lets an operator override the inferred charge, spin, active space, and basis set; the reported runs proceed autonomously. Agent observability and guardrails are built in.

IV. Two classes of failure

The pipeline handles two qualitatively different failure modes. *Operational* failures are infrastructural: a tool server raises, a pinned dependency resolves to the wrong version, or an SCF cycle is left at too coarse a convergence tolerance. They surface as tool-error envelopes or validator violations, and the owning sub-agent is designed to repair them in place without human intervention, retrying with corrected parameters (for example tightening `pyscf_conv_tol`), falling back to an alternate tool path, or flagging a version mismatch. *Algorithmic* failures are subtler: every infrastructural check passes, yet the completed result is scientifically out of specification. The canonical case is a dominant CSF overlap $|\gamma|$ below the threshold at which the qubitized-QPE shot count stays tractable, signaling genuine multi-reference character rather than a bug. A retry is useless; the recourse is a *prescription*, a codified remediation sequence walked in order: bond-dimension escalation, active-space revision, then a non-QPE estimator or an inflated but flagged shot count. The same mechanism reconciles a disagreement with a published reference by attributing it to an accountable cause (an active-space or version difference) rather than reporting it silently.

V. Deployment

The pipeline ships as a layered container image: a heavy base carrying the chemistry and estimation toolchain plus the Julia runtime, and a thin app layer for the pipeline source, rebuilt through CI and stored in a container registry. Runs are dispatched as batch jobs on a managed compute cluster, one worker per molecule; a deployment-service agent auto-derives the cluster tier from active-space size and atom count, mapping five tiers onto four machine types spanning roughly 16 to 80 vCPUs (up to 540 GiB memory, with in-core double factorization on the largest). The cluster autoscales on demand.

VI. Results

Table I reports end-to-end results for eight of the ten DARPA QB catalyst instances. The four single-reference instances ($|\gamma| \approx 0.92$) complete the full 13-stage pipeline; the four multi-reference instances ($|\gamma|$ between 0.14 and 0.56) are gated at Stage 7 and routed to the algorithmic recourse of Section IV: an operational fix (tightening `pyscf_conv_tol` to 10^{-9}) cleared the initial terminations, but the residual low overlap reflects genuine multi-reference character, which inflates the shot count by over an order of magnitude and cannot be cleared by a retry; their rows report estimates under that accepted penalty.

For the representative $TS_{1/2}$ run the algorithm failure probability is 0.027%; the 1.20-hour wall clock is dominated by integral construction and the DMRG sweep, with the closed-form Stage 8–13 block finishing in seconds. As a direct reproduction check, the $TS_{1/2}$ DMRG energy matches the DARPA reference [3] to within 2 mHa (-2928.211 versus -2928.213 Ha) under a matched bond-dimension schedule; absolute resource counts differ because active spaces are selected automatically by AVAS rather than by hand, which the validators flag as an accountable preprocessing choice. Across the suite, Toffoli counts span 2.5×10^{10} to 2.4×10^{13}, logical qubits 749 to 3565, and physical-qubit budgets 7.5×10^6 to 2.78×10^7 at code distances 41–47 (Figure 2); the spread within this set is a shot-configuration artifact, not chemistry.

VII. Extensions

Three extensions are in flight: (i) repackaging each stage as an agent-callable service with a typed request/result contract

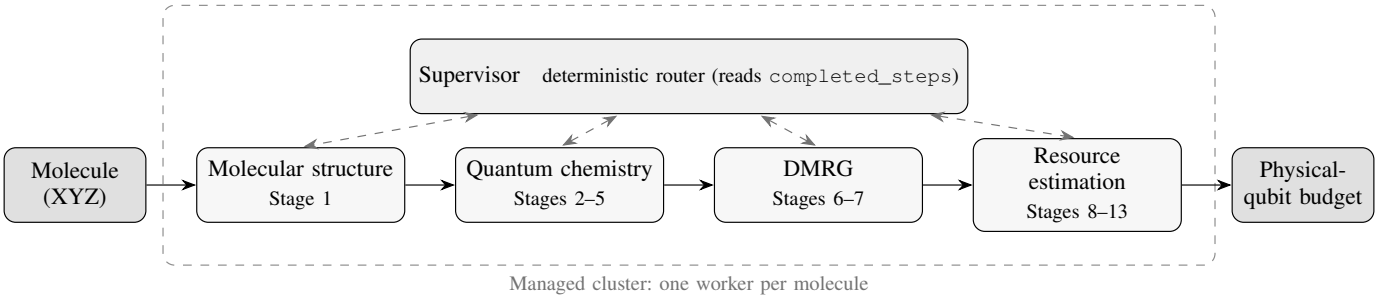

Fig. 1. Overall workflow. A deterministic supervisor routes each molecule through four ReAct sub-agents that own contiguous stage blocks and drive tool servers (PySCF, OpenFermion, QuantumMAMBO.jl, Block2, and a BenchQ-style Stage-13 estimator). Extractor and validator nodes carry a typed state along the pipeline, which runs as one worker per molecule on a managed cluster.

Molecule	(orbitals, electrons)	λ_{tot}	Toffoli	Logical	Physical	d	Wall (hr)	QPE (hr)
2	(31, 44)	218.3	2.50×10^{10}	994	8.41×10^{6}	41	15.23	967.0
PC	(31, 44)	216.5	1.32×10^{11}	1056	9.85×10^{6}	45	24.40	5115.9
RC	(31, 44)	210.4	1.25×10^{11}	1054	9.84×10^{6}	45	4.52	4849.5
$TS_{1/2}$	(31, 44)	216.8	1.34×10^{11}	1055	9.84×10^{6}	45	1.20	5175.9
4a	(21, 31)	98.4	3.67×10^{10}	749	7.50×10^{6}	43	2.54	1423.3
PC^{-}	(49, 67)	410.4	2.43×10^{13}	2091	1.76×10^{7}	47	6.40	942944.7
I	(49, 67)	483.6	5.69×10^{11}	3171	2.51×10^{7}	47	8.02	22040.5
$TS_{1/4a}$	(49, 67)	462.3	4.04×10^{12}	3565	2.78×10^{7}	47	34.43	156471.0

TABLE I

END-TO-END RESULTS ON THE DARPA QB CATALYST SUITE. UPPER BLOCK: SINGLE-REFERENCE INSTANCES ($|\gamma| \approx 0.92$). LOWER BLOCK: MULTI-REFERENCE INSTANCES GATED AT STAGE 7 (LOW $|\gamma|$: 4A 0.36, PC^{-} 0.14, I 0.56, $TS_{1/4a}$ 0.27). λ_{TOT} IS PRE-LP-BLISS; WALL IS THE PIPELINE WALL-CLOCK AND QPE THE FAULT-TOLERANT ALGORITHM RUNTIME.

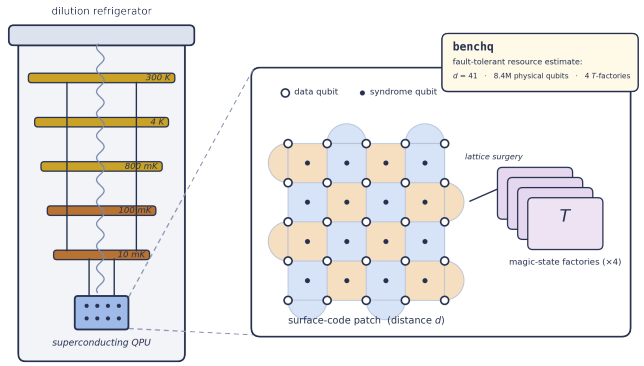

Fig. 2. Physical realization of a fault-tolerant estimate. The logical algorithm runs on a surface-code patch of distance d (data and syndrome qubits) on a superconducting QPU in a dilution refrigerator, with magic-state (T) factories supplying non-Clifford operations through lattice surgery. The annotated budget ($d = 41$, 8.4×10^{6} physical qubits, four factories) is the Stage-13 estimate for instance 2 (Table I).

and a common error envelope, bounding the supervisor's API surface; (ii) a Bayesian empirical-hardness model (regressions for Toffoli, logical- and physical-qubit counts over roughly one hundred published estimates), a separate artifact that may later couple back as a triage layer; and (iii) parameterising Stage 13 by qubit modality (ion-trap, neutral-atom, photonic), code distance, and error-correction code (toric, QLDPC, tile codes), so that a prompt such as "estimate $TS_{1/2}$ on an ion-trap architecture with a QLDPC code" routes through the same orchestration that produces the surface-code numbers of Section VI.

VIII. RELATED WORK

LLM agents that drive scientific workflows through external tools are now well established: ChemCrow [16] couples a language model to curated chemistry tools, and Coscientist [17] autonomously designs and runs laboratory experiments. Closest in spirit are El Agente Q [18] and its successor El Agente Cuántico [19], LLM-based multi-agent systems that generate and execute quantum-chemistry workflows from natural language, the latter extending to resource estimation. The present paper occupies the complementary slice: a single, fully automated implementation of the 13-stage workflow of [3], deployed against the public DARPA QB suite in unattended wall-clock time.

Several open-source packages estimate fault-tolerant resources for a *specified* algorithm or circuit, and are natural cross-checks for Stages 8–13. The Azure Quantum Resource Estimator [20] and Qualtran [21] standardize logical-to-physical cost modeling; OpenFermion's resource-estimates module [2], [8] provides the double-factorized QPE costings our Stage 12 calls directly; BenchQ [14], whose surface-code model Stage 13 reimplements, and pyLIQTR [22], TFermion [23], and QREChem [24] produce logical-level (Toffoli/T) estimates for chemistry from circuit- or Trotter-level inputs. Because Stages 12–13 drive the same or equiv-

alent cost models, per-stage outputs are directly comparable, and any residual deviation is attributable to upstream *automated* choices (active space, thresholds) rather than the cost model. For neutral-atom hardware, Infleqtion's resource-superstaq [25] compiles Cirq circuits to movement, syndrome-extraction, and magic-state-cultivation primitives with per-primitive costs, including correlated decoding; methodologically complementary to our formula-driven Stage 13, it is the planned cross-validation anchor for the neutral-atom extension (Section VII), with agreement bands on code distance, qubit count, and runtime. None of these tools automates the geometry-to-budget study itself; our pipeline consumes them as interchangeable back ends.

IX. Conclusion

The 13-stage resource-estimation workflow [3] is end-to-end automatable and deployable to a managed compute cluster, collapsing the human coordination that dominates the manual DARPA QB study from a multi-year calendar effort to unattended machine wall-clock while preserving its stage structure and per-stage domain checks. Absorbing both operational and algorithmic toil moves the marginal cost of evaluating a molecule from specialist-months to commodity-compute-hours and reallocates scarce expertise toward the judgment agents cannot yet make. Because the stage decomposition is agnostic to method, algorithm, qubit modality, and code family, the same orchestration extends to — and can be quantitatively cross-validated, via the estimators of Section VIII, against — the diversifying fault-tolerant landscape.

References

[1] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proceedings of the National Academy of Sciences*, vol. 114, no. 29, pp. 7555–7560, 2017.

[2] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, "Quantum computing enhanced computational catalysis," *Physical Review Research*, vol. 3, p. 033055, 2021.

[3] N. Bellonzi, A. Kunitsa, J. T. Cantin, J. A. Campos-Gonzalez-Angulo, M. D. Radin, Y. Zhou, P. D. Johnson, L. A. Martínez-Martínez, M. R. Jangrouei, A. S. Brahmachari, L. Wang, S. Patel, M. Kodrycka, I. Loaiza, R. A. Lang, A. Aspuru-Guzik, A. F. Izmaylov, J. R. Fontalvo, and Y. Cao, "Feasibility of accelerating homogeneous catalyst discovery with fault-tolerant quantum computers," 2024. [Online]. Available: https://arxiv.org/abs/2406.06335

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," 2022. [Online]. Available: https://arxiv.org/abs/2210.03629

[5] E. R. Sayfutyarova, Q. Sun, G. K.-L. Chan, and G. Knizia, "Automated construction of molecular active spaces from atomic valence orbitals," *Journal of Chemical Theory and Computation*, vol. 13, no. 9, pp. 4063–4078, 2017.

[6] Q. Sun, X. Zhang, S. Banerjee, P. Bao, M. Barbry, N. S. Blunt, N. A. Bogdanov, G. H. Booth, J. Chen, Z.-H. Cui, J. J. Eriksen, Y. Gao, S. Guo, J. Hermann, M. R. Hermes, K. Koh, P. Koval, S. Lehtola, Z. Li, J. Liu, N. Mardirossian, J. D. McClain, M. Motta, B. Mussard, H. Q. Pham, A. Pulkin, W. Purwanto, P. J. Robinson, E. Ronca, E. R. Sayfutyarova, M. Scheurer, H. F. Schurkus, J. E. T. Smith, C. Sun, S.-N. Sun, S. Upadhyay, L. K. Wagner, X. Wang, A. White, J. D. Whitfield, M. J. Williamson, S. Wouters, J. Yang, J. M. Yu, T. Zhu, T. C. Berkelbach, S. Sharma, A. Y. Sokolov, and G. K.-L. Chan, "Recent developments in the PySCF program package," *The Journal of Chemical Physics*, vol. 153, no. 2, p. 024109, 2020.

[7] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, "Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization," *Quantum*, vol. 3, p. 208, 2019.

[8] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, "Even more efficient quantum computations of chemistry through tensor hypercontraction," *PRX Quantum*, vol. 2, no. 3, p. 030305, 2021.

[9] I. Loaiza and A. F. Izmaylov, "Block-invariant symmetry shift: Preprocessing technique for second-quantized hamiltonians to improve their decompositions to linear combination of unitaries," 2023. [Online]. Available: https://arxiv.org/abs/2304.13772

[10] I. Loaiza, A. M. Khah, N. Wiebe, and A. F. Izmaylov, "Reducing molecular electronic hamiltonian simulation cost for linear combination of unitaries approaches," *Quantum Science and Technology*, vol. 8, no. 3, p. 035019, 2023.

[11] H. Zhai and G. K.-L. Chan, "Low communication high performance ab initio density matrix renormalization group algorithms," *The Journal of Chemical Physics*, vol. 154, no. 22, p. 224116, 2021.

[12] J. R. McClean, K. J. Sung, I. D. Kivlichan, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. Rubin, N. P. D. Sawaya, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush, "OpenFermion: The electronic structure package for quantum computers," 2017. [Online]. Available: https://arxiv.org/abs/1710.07629

[13] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, 2012.

[14] BenchQ developers, "BenchQ: Quantum resource estimation software," GitHub repository, 2023. [Online]. Available: https://github.com/zapatacomputing/benchq

[15] D. Litinski, "A game of surface codes: Large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, p. 128, 2019.

[16] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Augmenting large language models with chemistry tools," *Nature Machine Intelligence*, vol. 6, pp. 525–535, 2024.

[17] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, "Autonomous chemical research with large language models," *Nature*, vol. 624, pp. 570–578, 2023.

[18] Y. Zou, A. H. Cheng, A. Aldossary, J. Bai, S. X. Leong, J. A. Campos-Gonzalez-Angulo, C. Choi, C. T. Ser, G. Tom, A. Wang, Z. Zhang, I. Yakavets, H. Hao, C. Crebolder, V. Bernales, and A. Aspuru-Guzik, "El agente: An autonomous agent for quantum chemistry," 2025. [Online]. Available: https://arxiv.org/abs/2505.02484

[19] I. Gustin, L. M. Calderón, J. B. Pérez-Sánchez, J. F. Gonthier, Y. Nakamura, K. Panicker, M. Ramprasad, Z. Zhang, Y. Zou, V. Bernales, and A. Aspuru-Guzik, "El agente cuántico: Automating quantum simulations," 2025. [Online]. Available: https://arxiv.org/abs/2512.18847

[20] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, "Assessing requirements to scale to practical quantum advantage," 2022. [Online]. Available: https://arxiv.org/abs/2211.07629

[21] M. P. Harrigan, T. Khattar, C. Yuan, A. Peduri, N. Yosri, F. D. Malone, R. Babbush, and N. C. Rubin, "Expressing and analyzing quantum algorithms with Qualtran," 2024. [Online]. Available: https://arxiv.org/abs/2409.04643

[22] K. Obenland, J. Elenewski, A. Kurlej, J. Belarge, J. Blue, and R. Rood, "pyLIQTR: A quantum circuit synthesis and resource estimation library," Zenodo, doi:10.5281/zenodo.7221272, 2023, code: github.com/isi-usc-edu/pyLIQTR.

[23] P. A. M. Casares, R. Campos, and M. A. Martin-Delgado, "TFermion: A non-Clifford gate cost assessment library of quantum phase estimation algorithms for quantum chemistry," *Quantum*, vol. 6, p. 768, 2022. [Online]. Available: https://arxiv.org/abs/2110.05899

[24] M. Otten, B. Kang, D. Fedorov, A. Benali, S. Habib, Y. Alexeev, and S. K. Gray, "QREChem: Quantum resource estimation software for chemistry applications," 2024. [Online]. Available: https://arxiv.org/abs/2404.16351

[25] C. Campbell, R. Rines, V. Omole, T. Oberoi, P. Goiporia, R. Roy, R. P. Cline, E. B. Jones, and T. Tomesh, "Resource estimation via efficient compilation of key quantum primitives," 2026, code: github.com/Infleqtion/resource-superstaq. [Online]. Available: https://arxiv.org/abs/2604.01376