HybridQSE - Claude Build Instructions (Consolidated)

Version 1.0 • February 05, 2026

Purpose: give a second model (Claude) an unambiguous, engineering-grade plan for extending the Hybrid Quantum Simulation Engine from the current Hartree-Fock foundation into a validated, multi-method, multi-scale platform with optional ML acceleration.

Source map used: project docs (START_HERE/PROJECT_STATUS/DEVELOPMENT_GUIDE/ACHIEVEMENT_SUMMARY/PROJECT_TREE) [P-Docs], the two blueprint PDFs that define long-term architecture and multi-scale/ML features [BP, GQ], plus official library documentation [ExtDocs].

How to use this document

Read sections 1-3 once to internalize constraints. Then implement milestones M1->M4 in order. Every milestone has acceptance tests; do not merge work that fails them. This validate-first posture is a core blueprint requirement [BP].

Fast start (what exists now)

- Repo already contains: Molecule/Atom + XYZ IO, BasisSet/integrals (PySCF-backed), working RHF SCF solver, validation/benchmarks, example script, and unit tests [PROJECT_TREE, PROJECT_STATUS].
- Follow the development guide session ordering: DFT -> geometry optimization -> vibrational frequencies -> real-time dynamics [DEVELOPMENT_GUIDE].

Non-negotiables

- **Validation-first:** every new method must be verified against an external reference (PySCF or literature) before feature expansion [BP, PROJECT_STATUS].
- **Reproducibility:** deterministic runs when a seed is provided; log versions, inputs, and tolerances [BP].
- **Modularity:** new methods must not require core edits beyond registering a plugin/entry point [BP].
- **Units and provenance:** embed geometry units, energy units, method/basis identifiers, convergence criteria, and backend versions in results [BP].

1. Target capabilities and scope boundaries

The engine's truth chain is: (i) a system representation (atoms/fields + boundary conditions), (ii) a Hamiltonian/operator definition, (iii) a solver method (HF/DFT/post-HF/MC/MD/etc.), (iv) properties/spectra extraction, and (v) validation against references. This layered decomposition is explicit in the generalized engine design [GQ].

Scope boundaries for v1.x

- Primary: molecules and small clusters (non-relativistic electronic structure) with Gaussian basis sets via PySCF backend [PROJECT_STATUS].
- Secondary: classical nuclear dynamics on a quantum PES (Born-Oppenheimer MD) [DEVELOPMENT_GUIDE].
- Tertiary: vibrational analysis (Hessian -> normal modes -> IR) and spectrum simulation [GQ, DEVELOPMENT_GUIDE].
- Deferred: production-grade lattice QCD; treat as separate module with toy validation first [GQ].

What IR can and cannot do for geometry

IR spectra encode vibrational normal-mode frequencies and intensities near an equilibrium structure (normal-mode / mass-weighted Hessian picture) [GQ]. They constrain structure strongly, but the inverse problem IR -> unique 3D geometry is generally ill-posed. Implement model-based inversion: propose geometries, compute spectra, and fit/optimize candidates to match observed spectra while tracking uncertainty.

2. Baseline code you must preserve

Module	Status	Invariants
hybrid_qse/core/molecule.py	Implemented	Atom/Molecule; XYZ IO; nuclear repulsion; geometry ops
hybrid_qse/core/basis.py	Implemented	Integral API + PySCF interface; dipole integrals
hybrid_qse/methods/hf.py	Implemented	RHF SCF convergence; energy/orbitals/dipole outputs
hybrid_qse/validation/benchmarks.py	Implemented	Reference DB; error metrics; auto comparison

Citations: baseline inventory from PROJECT_TREE and PROJECT_STATUS [P-Docs].

Rule: any refactor must keep public imports stable (or provide deprecation shims). For each milestone, expand the validation database with at least one new reference point [BP].

3. Milestone plan (with acceptance tests)

M1 - DFT wrapper (RKS/UKS) with functional registry

- Implement methods/dft.py as a wrapper around PySCF DFT (RKS first; UKS later) [DEVELOPMENT_GUIDE; ExtDocs: PySCF].
- Expose functionals by name; map common aliases; reject unsupported functionals with clear errors [DEVELOPMENT_GUIDE].
- Return: total energy, convergence, MO coefficients/energies, density handle if available [DEVELOPMENT_GUIDE].
- **Acceptance:** DFT(H2O,B3LYP/6-31G) converges; energies match PySCF within tolerance; unit tests added [PROJECT_STATUS].

M2 - Geometry optimization with robust convergence

- Implement methods/optimizer.py. Start with SciPy BFGS using numerical gradients; later swap to analytic gradients via PySCF [DEVELOPMENT_GUIDE; ExtDocs: SciPy, PySCF].
- Add constraints support (freeze atoms, fix bond length/angle) [BP].
- **Acceptance:** Water converges from perturbed coordinates; optimized energy is lower; frequency analysis shows no imaginary modes for a true minimum [GQ].

M3 - Vibrations and IR spectrum simulation

- Implement properties/vibrations.py: numerical Hessian -> mass-weighted Hessian -> normal modes and frequencies (cm^-1) [GQ; DEVELOPMENT_GUIDE].
- Compute IR intensities via dipole-derivative finite differences and generate broadened spectrum [GQ].
- **Acceptance:** H2O frequencies are physically reasonable; peaks qualitatively align with reference spectra; unit tests cover Hessian symmetry and unit conversions [DEVELOPMENT_GUIDE].

M4 - Born-Oppenheimer MD (toy -> stable)

- Implement dynamics/md.py with velocity Verlet; forces via numerical gradient of energy (first pass) [DEVELOPMENT_GUIDE].
- Track total energy drift; implement thermostats only after NVE baseline is stable [BP].
- **Acceptance:** short NVE trajectory shows bounded energy drift; reproducible with fixed seed/initial velocities [BP].

4. Core representations to implement (make it extensible)

The blueprint emphasizes extensibility via plugins and a clean API [BP]. Implement minimal stable interfaces and keep everything else behind them.

4.1 Canonical objects

- **System**: Molecule now; later extend to PeriodicSystem or FieldSystem [GQ].
- **Method**: HF/DFT/MP2/CC/etc. implementing run(system, settings) -> Result [BP].
- **Result**: energy, convergence, orbitals/density handles, gradients (optional), provenance dict [BP].
- **Task**: explicit tasks (energy/gradient/optimize/vibrations/md) to compose workflows [BP].

4.2 Plugin registry

Implement a registry so new methods can be added without editing core code (entry points or internal registry) [BP].

5. Shortcuts that preserve truth

Predefine what we know corresponds to established acceleration approaches: symmetry, reduced representations, surrogate models, and error control [BP]. Implement them explicitly and quantify error/uncertainty.

Physics-first acceleration

- **Symmetry:** exploit point-group and spin symmetry where possible [BP].
- **Integral approximations:** density fitting / RI, Cholesky of ERIs; avoid full 4-index tensors [BP].
- **Pseudopotentials:** reduce core electrons for heavy atoms (later) [BP].
- **Adaptive basis:** automate basis refinement and basis-set convergence checks [BP].

ML acceleration (only with uncertainty control)

- **Delta-learning:** learn corrections to a cheap baseline toward a high-level reference [BP].
- **Surrogate potentials:** train ML force fields on ab initio forces; run MD with uncertainty-triggered retraining (active learning) [BP].
- **Neural wavefunctions:** optional neural-VMC module with physics constraints; defer until M1-M4 stable [BP].
- **Gate:** any ML mode must provide uncertainty and a fallback to ab initio when uncertainty is high [BP].

6. Comparing to verified sources (validation playbook)

- Expand BenchmarkDatabase: per-method and per-basis reference energies; later gradients/frequencies [PROJECT_STATUS].
- Validate against PySCF at identical settings first; then add literature/experimental references [PROJECT_STATUS, BP].
- For spectra: validate computed normal modes and IR intensities against curated reference spectra; document scaling factors [GQ].
- Automate: one command runs validations and prints a pass/fail summary [BP].

Minimum logging schema (structured output)

```
{
  "engine_version": "...",
  "backend": {"pyscf": "...", "numpy": "...", "scipy": "..."},
  "system": {"formula": "H2O", "charge": 0, "multiplicity": 1, "units": "angstrom"},
  "method": {"name": "DFT", "functional": "B3LYP", "basis": "6-31G"},
  "settings": {"scf_tol": 1e-9, "max_iter": 50},
  "results": {"energy_hartree": -76.4, "converged": true},
  "provenance": {"input_hash": "...", "seed": 1234}
}
```

7. Documentation and user friendliness

- Blueprint requirement: beginner-friendly Python interface, extensive tutorials, and actionable errors [BP].
- Keep a 5-minute path: pip install -> run example -> validated output [START_HERE].
- Write how-to guides for DFT/opt/vibrations/MD with expected outputs [BP].
- Support notebooks + a simple CLI input format (YAML/JSON) as a first-class interface [BP].

8. References

Internal project sources

- P-Docs: START_HERE.md, PROJECT_STATUS.md, DEVELOPMENT_GUIDE.md, ACHIEVEMENT_SUMMARY.md, PROJECT_TREE.txt (repo).
- BP: Blueprint for a Hybrid Multi-Scale Modeling Engine (PDF).
- GQ: Designing a Generalized Quantum Simulation Engine for Atoms, Molecules, and Subatomic Particles (PDF).

External documentation sources (ExtDocs)

- PySCF: pyscf.org
- Psi4: psicode.org/psi4manual/master/
- OpenFermion: quantumai.google/openfermion
- ASE (Atomic Simulation Environment): wiki.fysik.dtu.dk/ase/
- LAMMPS: lammps.org
- Quantum ESPRESSO: quantum-espresso.org
- JAX: jax.readthedocs.io
- QUDA: lattice.github.io/quda/

End.