Basic of Q-chem

Q-Chem is a general-purpose electronic-structure package maintained and distributed by Q-Chem, Inc., headquartered in Pleasanton, California (USA). Founded in 1993, it is developed by a broad community of contributors. Within the CCATS group, Q-Chem serves as the primary platform for method development and production calculations.

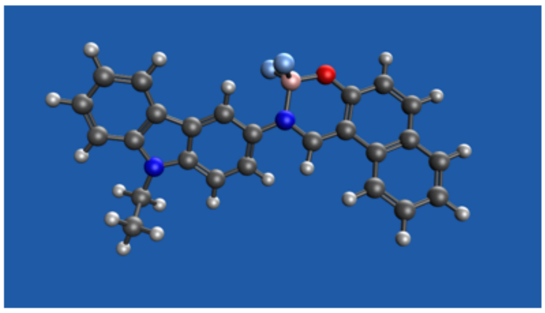
1.Q-Chem overview

Q-Chem is a general-purpose electronic-structure package offering broad coverage from HF/DFT to advanced correlated and multireference methods. As of **August 23, 2025**, the latest release is **Q-Chem 6.3**.

Core capabilities (selected)

1. **Ground-state SCF (HF/ROHF/UHF):** analytic gradients/Hessians, robust optimizers, and property analysis.
2. **Density Functional Theory:** semilocal → double-hybrid functionals; energies, geometry optimizations, and vibrational analyses; TDDFT including spin-flip variants for challenging excitations.
3. **Post-HF correlation:** MP2 family (incl. OO/RI/dual-basis, MP3), coupled-cluster (CCSD, CCSD(T)) and a wide EOM-CC suite (EE/SF/IP/EA), plus ADC methods.
4. **Strong correlation & active-space options:** CASSCF/CAS-CI, RAS-CI/RAS-SF, CCVB/PP, and spin-flip DFT.
5. **Excited states (broad toolbox):** CIS, CIS(D), TDDFT/SF-TDDFT, ΔSCF via MOM/SGM, NOCI/NOCIS, EOM-CC, ADC; support for core-level excitations (CVS).
6. **Embedding & extended systems:** stand-alone **QM/MM**, CHARMM interface, **QM/EFP**, and modern dielectric/embedding models.
7. **PES exploration & optimization:** constrained and unconstrained optimizations (minima/TS), IRC paths, and relaxed scans.

Performance & parallelism
Q-Chem supports OpenMP threading, MPI for distributed-memory runs. Usually one should run with 2^n numbers of cores. I ran some benchmark of the parallel jobs:


Test molecule, 50 atoms& 214 electrons in total

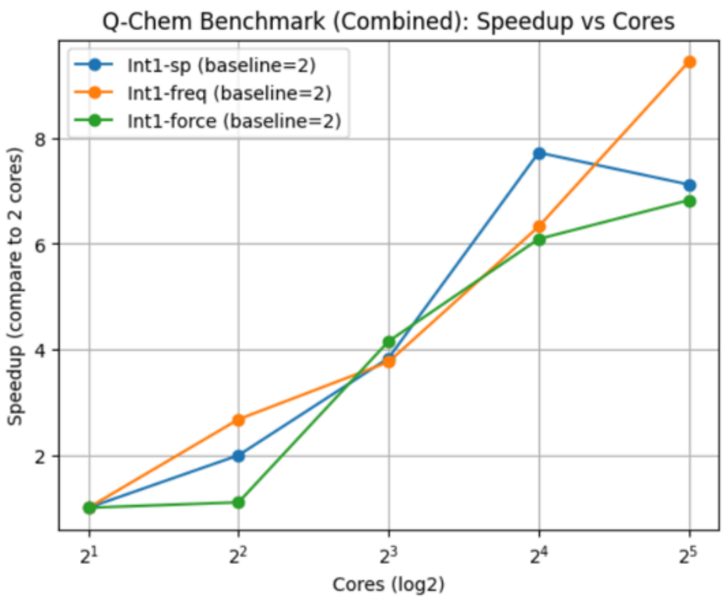


Figure 1. Parallel benchmarking of Q-Chem on the test molecule. Three tasks(single-point (SP), force, and vibrational frequency) were run with 2, 4, 8, 16, and 32 CPU cores on PETE. Speedup is computed from wall time relative to the 2-core run. According to the Figure, adding more cores won't give proportional speeds.

2. Install & Compile Q-chem

Requirements:

- An access to a Linux Computer
- Q-chem package
- Basic knowledge of the Linux shell & Slurm

Optional:

- Pete supercomputer (used as an example in this tutorial)
- Q-chem developer access

If you plan to develop new features in Q-Chem (e.g., new functions, special requirements), or if you simply want your own installation, you'll need to download and compile Q-Chem in your home environment.

2.1: Download Q-Chem

Select a suitable directory for Q-Chem that has enough space (at least 10 GB) and will not be automatically cleaned up by system administrators.

On the Pete supercomputer, a recommended location is:

```
/scratch/$user/software/
```

Replace `$user` with your own username. If the `software` directory does not exist, create it with:

```
cd /scratch/$user
mkdir software
cd software
```

You can use `pwd` to verify that you are inside `/scratch/$user/software`.

For Q-Chem developers, you can check out the latest source code via **SVN**:

```
svn --username=$qchem-developer-id checkout https://jubilee.q-chem.com/svnroot/qchem/trunk
$qchem-directory-name
```

Replace `$qchem-developer-id` with your developer ID and `$qchem-directory-name` with a custom name for your Q-Chem installation.
If you just want a standard installation, you can simply use `qchem` as the folder name:

```
svn --username=$qchem-developer-id checkout https://jubilee.q-chem.com/svnroot/qchem/trunk
qchem
```

The download may take some time. After it finishes, you should see a new folder with the name you specified. Change into this directory, and prepare to configure the build.

2.2 Compile Q-Chem

On the Pete supercomputer, you can write the following script (`compiler.sh`) to set up the environment and configure Q-Chem:

```
#!/bin/bash
#SBATCH --partition=batch
#SBATCH --ntasks=16
#SBATCH --tasks-per-node=16
#SBATCH --output=./logs/%J_stdout.txt
#SBATCH --error=./logs/%J_stderr.txt
#SBATCH --time=1-00:00:00
#SBATCH --job-name=comp


mkdir logs

module purge
module load cmake3/3.24.3
module load impi/2021.2.0
module load intel/2021.2.0

export QC=/scratch/$USER/software/qchem
source $QC/bin/qchem.setup.sh
export QCSCRATCH=/scratch/$USER


cd $QC
./configure intel release

make -j 16 qcprog.exe > mall.log
```

Source the script to start configuration:

```
source ./compiler.sh
```

The terminal should display progress messages. If everything is set up correctly, you will see:

> Configureing done
>
> Generating done
>
> Build files have been written to: /scratch/$USER/software/qchem/build

This indicates that the **build directory** has been created successfully.

If you encounter errors, check the output and carefully read the error messages. In most cases, problems can be resolved by installing the required module, searching for the error message on Google or by asking ChatGPT for help.

Copy the compiler.sh file into build directory

```
cp ./compiler.sh ./build/compiler.sh
```

Edit the copied script by adding one more line at the end:

```
make -j 16 qcprog.exe > mall.log
```

This will build Q-Chem with 16 parallel threads and save the compilation output into `mall.log`.

Use the following command to submit the compile command.

```
sbatch compiler.sh
```

The first compilation can usually take more than **4 hours**.

To check whether the job is still running or has finished (successfully or with errors), use:

```
squeue -u $USER
```

You can also check the build logs in:

```
/scratch/$USER/software/qchem/build/logs
```

When compilation finishes successfully, a file named `qcprog.exe` will appear in either the `build` directory or the `exe` directory.

If the executable is located in the `build` directory, copy it to your designated `exe` folder, for example:

```
cp ./build/qcprog.exe /scratch/$USER/software/qchem/exe/
```

At this point, you have successfully compiled Q-Chem.

3. File structure of Q-chem input file

Q-Chem input files are divided into several **modules**, each enclosed by `$…` and `$end`.
Below is an example input file, followed by explanations of the most commonly used sections:

```
$molecule
0 1
O         -3.13270       -0.56690        1.47260
H         -3.45320        0.33800        1.47260
O         -1.81270       -0.56690        1.47260
H         -1.49230       -1.21460        2.10460
$end

$rem
BASIS  =  6-31+G*
```

```
JOB_TYPE  = opt
METHOD = HF
SOLVENT_METHOD = PCM
$end

$solvent
Dielectric 78.39
$end
```

3.1 $molecule

The **molecule block** specifies the atomic structure, charge, and spin multiplicity of the system:

```
$molecule
0 1
O         -3.13270       -0.56690        1.47260
H         -3.45320        0.33800        1.47260
O         -1.81270       -0.56690        1.47260
H         -1.49230       -1.21460        2.10460
$end
```

The first line after `$molecule` contains:

- **Charge** (here `0`, meaning neutral molecule)
- **Multiplicity** (here `1`, meaning a singlet state, in which all electrons are paired)

The following lines list each atom:

- **Element symbol** (e.g., O, H)
- **Cartesian coordinates** in Ångström (x, y, z).

This block defines both the electronic state and the geometry of your system.

3.2 `$rem`

The **$rem block** (short for *run-time environment module*) controls most of the calculation settings in Q-Chem.

```
$rem
BASIS          = 6-31+G*
JOB_TYPE       = opt
METHOD         = HF
SOLVENT_METHOD = PCM
$end
```

Here's what each line means:

- `BASIS = 6-31+G\*`
    - Specifies the basis set.

    - `6-31+G*` is a split-valence basis set with diffuse (`+`) and polarization (`*`) functions.
    - You can change this to other basis sets (e.g., `cc-pVDZ`, `def2-TZVP`).
- `JOB_TYPE = opt`
    - Defines the type of calculation.
    - Common options:
        - `sp` → single-point energy
        - `opt` → geometry optimization
        - `freq` → vibrational frequency analysis
        - `force` → analytic gradient
- `METHOD = HF`
    - Specifies the electronic structure method.
    - Examples:
        - `HF` (Hartree–Fock)
        - `DFT functionals` (e.g., `B3LYP`, `PBE0`)
        - `MP2`, `CCSD`, `ADC`, etc.
- `SOLVENT_METHOD = PCM`
    - Tells Q-Chem to include solvent effects using the **Polarizable Continuum Model (PCM)**.
    - Requires an additional `$solvent` block. It specific which solvent you are using by their dielectric value.

The `$rem` section is the heart of the input file, it controls what calculation is being done and how. A significant thing is that, whenever you copy an input file from others, and you don't know exactly each line means.

4. Run a Q-Chem job

Once you have Q-Chem installed, run a few small jobs to test it. (You can also borrow a working Q-Chem install on a shared server to practice job submission.)

We usually keep each project in its own folder—don’t mix everything together (unless you’re a T-800).

```
cd /scratch/$USER
mkdir -p qchem_calculation/tutorial_practices
cd qchem_calculation/tutorial_practices
```

Tip: press `Tab` for filename autocompletion.

You can either copy the example input file in section 3, or create your own one. I give the example input name as `H2O2.inp`. Just use the following command

```
vi H2O2.inp
```

In `vi`: press `a` to enter insert mode, paste example input from Section 3, press `Esc`, then type `:wq` to save and quit.

usually we use bash to submit jobs. Still create a file named `sub.sh`. You can just use the following bash first:

```
#!/bin/bash
#SBATCH -J qchem_test              # Job name
#SBATCH -p express                 # Partition/queue
#SBATCH --time=01:00:00            # Wall time limit
#SBATCH --nodes=1
#SBATCH --ntasks=1                 # 1 MPI task (Q-Chem will use threads)
#SBATCH --cpus-per-task=16         # Threads for OpenMP (-nt)
#SBATCH -o ./logs/OUTPUT_%J.log
#SBATCH -e ./logs/ERRORS_%J.log


# --- Q-Chem environment ---
export QC=/scratch/$USER/software/qchem          # Q-Chem install root
# export QCAUX=/path/to/qcaux                    # Optional: EFP bases, etc.
source "$QC/bin/qchem.setup.sh"                  # Or qchem.setup / qchem_setup
# Use node-local scratch if available; else fall back to a per-user scratch
export QCSCRATCH="${SLURM_TMPDIR:-/scratch/$USER/qcscratch/$SLURM_JOB_ID}"
mkdir -p "$QCSCRATCH"

# Compiler/runtime modules (adjust to your cluster)
module purge
module load intel/2021.2.0

# Make OpenMP threads match Slurm allocation
export OMP_NUM_THREADS="${SLURM_CPUS_PER_TASK}"


infile="$1"
# Output name: replace .inp with .out (or just append .out if no .inp)
outfile="${infile%.inp}.out"

echo "Running Q-Chem on $infile with ${OMP_NUM_THREADS} threads…"
qchem -nt "${OMP_NUM_THREADS}" "$infile" "$outfile"

echo "Done. Output: $outfile"
```

Why these settings?

- `--ntasks=1` + `--cpus-per-task=16` is the correct layout for **threaded** (OpenMP) Q-Chem runs.
- `qchem -nt N` tells Q-Chem to use **N OpenMP threads**.

Submit the job (from the folder containing `H2O2.inp` and `sub.sh`):

```
sbatch sub.sh H2O2.inp
```

Useful Slurm command:

```
squeue -u $USER           # Check job status
```

5. View Q-Chem Results with IQmol