Custom MCCE Runs and submit_shell

Default MCCE runs (run_mcce4)

By default, each step of MCCE has certain parameters set to it. For example, step 3 defaults to the Gunner Lab's in-house Poisson-Boltzmann solver, NGPB, though Delphi and ZAP are available to those with the respective OpenEye licenses. But what if the user wants non-default settings?

By default, run_mcce4 uses the following settings:

```
step1.py {input_pdb} -d 4 --dry
step2.py -l 1 -d 4
step3.py -d 4
step4.py --xts -i 7 -n 1
```

Each of these lines is a shell command. If you were to copy and paste these lines (after changing {input_pdb} to a local PDB file), MCCE would run just as though "run_mcce4" had been the command. To see more of our options, let's look at the output of "step1.py -h":

```
step1.py -h
usage: step1.py [-h] [--norun] [--noter] [-e /path/to/mcce] [-u Key=Value] [-d epsilon] [-
load_runprm prm_file] [--dry] pdb

Run mcce step 1, premcce to format PDB file to MCCE PDB format.

positional arguments:
  pdb

options:
  -h, --help            show this help message and exit
  --norun               Create run.prm but do not run step 1; default: False.
  --noter               Do not label terminal residues (for making ftpl); default: False.
  -e /path/to/mcce      mcce executable location; default: mcce.
```

```
  -u Key=Value          User customized variables; default: .
  -d epsilon            protein dielectric constant for PBE solvers; default: 4.0.
  -load_runprm prm_file
                        Load additional run.prm file, overwrite default values.
  --dry                 Delete all water molecules; default: False.
```

So, if we wanted to start step 1 with an alternate version of MCCE, a dielectric constant of 8, and keeping explicit water molecules, we could change the command to:

```
step1.py {input_file.pdb} -e /home/other_mcce -d 8
```

Following a similar process for each step, we can create a full custom run to our liking.

submit_mcce4.sh

However, it's tedious to submit these instructions over and over. In MCCE4-Alpha/schedulers, a file named "submit_mcce4.sh" can be found. We recommend copying "submit_mcce4.sh" to the local directory and customizing it to fit the desired job. Let's look at the script:

```
# Input and Output:
input_pdb="prot.pdb"    # (INPDB)

# Set MCCE4 Parameters
MCCE_HOME="/home/MCCE4-Alpha"
USER_PARAM="./user_param"
EXTRA="./user_param/extra.tpl"
TMP="/tmp"
CPUS=1

# Step control flags
step1="t"               # STEP1: pre-run, pdb-> mcce pdb  (DO_PREMCCE)
step2="t"               # STEP2: make rotamers            (DO_ROTAMERS)
step3="t"               # STEP3: Energy calculations      (DO_ENERGY)
step4="t"               # STEP4: Monte Carlo Sampling     (DO_MONTE)
step_clean="t"          # Clean PBE data                  (BACKUP CLEAN) : Set to f if step3
--debug option is used

# Optional step controls
stepM="f"               # Generate Partial Membranes                    : If true, user MUST
satisfy condidtions of stepM.sh, which can be be obtained on MCCE4/inhouse/stepM.sh
```

```
stepA="f"               # Run a custom script between step1 and step2   : If true, user MUST
satisfy condidtions of their custom script
stepB="f"               # Run a custom script between step2 and step3   : If true, user MUST
satisfy condidtions of their custom script
stepC="f"               # Run a custom script between step3 and step4   : If true, user MUST
satisfy condidtions of their custom script

# MCCE Simulation
STEP1="step1.py -d 4 --dry"
STEP2="step2.py -d 4 -l 1"
STEP3="step3.py -d 4 -s delphi -p $CPUS -t \$TMP"
STEP4="step4.py --xts -i 7 -n 1"

# Optional MCCE script locations
STEPM="/path/to/stepM.sh"          # Optional StepM: Bash script
STEPA="/path/to/stepA_script.py"   # Optional StepA: Python script to run between step1 and
step2.
STEPB="/path/to/stepB_script.py"   # Optional StepB: Python script to run between step2 and
step3.
STEPC="/path/to/stepC_script.py"   # Optional StepC: Python script to run between step3 and
step4.

# NOTE: User is responsible to precheck if custom scripts work properly and efficiently
```

- "Set MCCE4 Parameters" controls which version of MCCE is being used.
- "Step control flags" allow for steps to be turned on and off between runs.
- "Optional step controls" allow for custom scripts to be run between steps, and partial membrane generation
- "MCCE Simulation" is where each step is customized- use "step[1-4].py -h" to see options for each step
- Lastly, the paths to any custom scripts are referenced

Then, submit_mcce4.sh can be executed:

```
chmod +x submit_mcce4.sh # make this file an executable
./submit_mcce4.sh # execute the submit shell
```

We recommend using custom submit shells in conjunction with p_batch/pro_batch. These programs included with MCCE4-Alpha are able to accept a -custom flag and a local shell script, to execute it across all relevant jobs. Learn about this functionality here.

Revision #4
Created 11 August 2025 20:06:52 by Jared

Updated 12 August 2025 19:31:59 by Jared