Supplementary slides to ASR tutorial part 1, SWiMM 2021

Ask Hjorth Larsen

Department of Physics, DTU, Denmark

March 26, 2021

The ASR CLI interface

```
$ asr --help
Usage: asr [OPTIONS] COMMAND [ARGS]...

Options:
  --version   Show the version and exit.
  -h, --help  Show this message and exit.

Commands:
  cache      Inspect results.
  database   ASR material project database.
  find       Find result files.
  list       List and search for recipes.
  params     Compile a params.json file with all options and defaul
  results    Show results from records.
  run        Run recipe or python function in multiple folders.
```

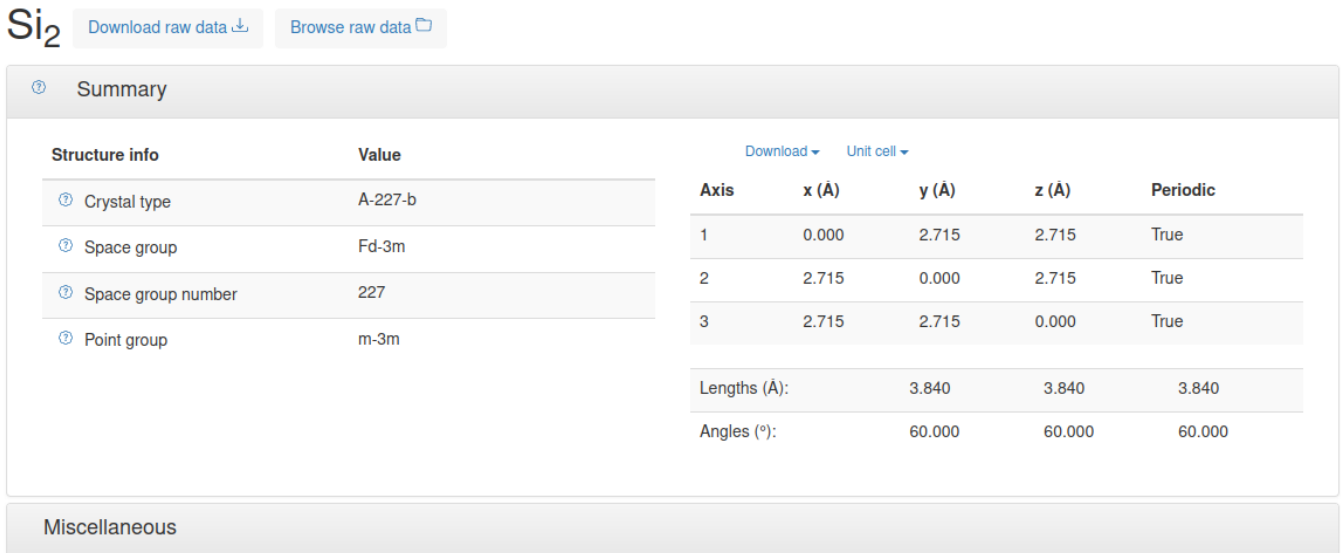
- See help for subcommands: `asr run --help`
- See help for recipe: `asr run "asr.structureinfo --help"`

Run a Recipe:

```
recipe:
  # Use ASE to build an example system (bulk Si)
  ase build -x diamond Si > structure.json

  # Inspect in ASE GUI:
  ase gui structure.json

  # Run our first recipe:
  asr run structureinfo

  # See what we have in front of us:
  asr results

  # Running same Recipe again reuses existing Record:
  asr run structureinfo

  # See what Records exist in this folder (also --help):
  asr cache ls
```

```
database:
  # Collect results into ASE database (database.db):
  asr database fromtree .

  # Inspect database using ASE tool:
  ase db database.db

  # Start webserver:
  asr database app database.db
```

View on local computer: 0.0.0.0:5000



Calculate a band structure (laptop-friendly parameters):

```
calculate:
  ase build -x diamond Si > structure.json
  # Be sure to check ase run "asr.bandstructure --help"
  # We will run with some very coarse parameters:
  asr run "asr.bandstructure --calculator \
      {'name':'gpaw','mode':'pw','kpts':(4,4,4)} \
      --npoints 60 --bscalculator {...,'txt':'-'}"

  # Remember: Use quotes "" to specify command correctly
  # Inherit Recipe defaults using ...: {...,'key':'value'}

  # The output will tell us that it runs both ground-state
  # and band structure calculation.

  # Let's see the results:
  asr results

  # Build database:
  asr database fromtree .
```

A look at the internals

```
askhl@erlkoenig:~/swimm-asr-demo/2-calculate$ tree .asr
.asr
├── asr.bandstructure:calculate-0069f58727
│   └── bs.gpw
├── asr.bandstructure:main-b91eb879df
├── asr.gs:calculate-e8c373a172
│   └── gs.gpw
├── asr.gs:main-ad21cbf526
├── asr.magnetic_anisotropy:main-1032a0522b
├── asr.magstate:main-190543f891
├── external_files
│   ├── 0069f58727-bs.gpw
│   └── e8c373a172-gs.gpw
├── records
│   ├── asr.bandstructure:calculate-0069f58727.json
│   ├── asr.bandstructure:main-b91eb879df.json
│   ├── asr.gs:calculate-e8c373a172.json
│   ├── asr.gs:main-ad21cbf526.json
│   ├── asr.magnetic_anisotropy:main-1032a0522b.json
│   └── asr.magstate:main-190543f891.json
├── record-table.json
└── work_dirs.json
```

Instructions are cached individually (“main” vs “calculate”)

Perform multiple calculations

```
bandstructures:
  # See contents of database:
  ase db inputs.db

  # Probe what would be unpacked from database:
  asr database main inputs.db --tree-structure tree/{formula}

  # Unpack database, after which we can inspect folder tree:
  asr database main inputs.db --tree-structure \
      tree/{formula} --run --atomsfile structure.json

  # Run band structure in each folder:
  asr run "asr.bandstructure --calculator \
      {'name':'gpaw','mode':'pw','kpts':(4,4,4)} \
      --bscalculator {...} --npoints 30" tree/* -j 4

  # Collect calculations into database.db:
  asr database fromtree tree/*
```

Write a recipe: An example

```
@command(module='asr.npt')
@option('--atoms', type=AtomsFile(),
        default='structure.json')
@option('--md', help='MD parameters', type=DictStr())
def main(atoms: Atoms,
         md: dict = {
             'timestep': 15.0,
             'temperature': 300.0,
             'nsteps': 100,
         }
) -> Result:
    """Toy recipe for running NPT dynamics with Asap3/EMT."""
    run_npt(atoms, **md)
    tnow = atoms.get_temperature()
    return Result.fromdata(atoms=atoms.copy(),
                           final_temperature=tnow,
                           volume=atoms.cell.volume,
                           **md)
```

- Detailed tutorial on Recipes by Joshua Elliott

On workflows

Towards high-throughput workflows with ASR and myqueue

- `https://myqueue.readthedocs.io/`

```
(venv-210302) [askhl@sylg tree-askhl]$ mq --help
usage: mq [-h] [-V]
          {help,list,ls,submit,resubmit,remove,rm,info,workflow,run,kick,modify,
init,sync,completion,config,daemon}
          ...

Frontend for SLURM/LSF/PBS.

Type "mq help <command>" for help. See https://myqueue.readthedocs.io/ for
more information.

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         Show version number

Commands:
  {help,list,ls,submit,resubmit,remove,rm,info,workflow,run,kick,modify,init,syn
c,completion,config,daemon}
    help                Show how to use this tool.
    list (ls)           List tasks in queue.
    submit              Submit task(s) to queue.
    resubmit            Resubmit failed or timed-out tasks.
    remove (rm)         Remove or cancel task(s).
    info                Show detailed information about task.
    workflow            Submit tasks from Python script or several scripts
                        matching pattern.
    run                 Run task(s) on local computer.
    kick                Restart T and M tasks (timed-out and out-of-memory).
    modify              Modify task(s).
    init                Initialize new queue.
    sync                Make sure SLURM/LSF/PBS and MyQueue are in sync.
    completion          Set up tab-completion for Bash.
    config              Create config.py file.
    daemon              Interact with the background process.
```

Example of workflow file (legacy syntax)

```
def create_tasks():
    """Generate all myqueue tasks for a specific material."""
    tasks = []

    def append(*args, **kwargs):
        tasks.append(task(*args, **kwargs))

    append('asr.setup.symmetrize',
           creates=['unrelaxed.json'],
           resources='1:10m')
    append('asr.relax --d3', resources='40:5h',
           restart=2,
           creates=['results-asr.relax.json'],
           deps='asr.setup.symmetrize')
    append('asr.database.material_fingerprint',
           creates=['results-asr.database.material_fingerprint.jso
           resources='1:10m',
           deps="asr.relax+--d3")
    ...
    return tasks
```

Workflow of ASR+Myqueue project

- Tell Myqueue about computational resources
- Define database of input geometries
- Split database into tree
- Define workflow file
- Run Myqueue on workflow and submit jobs
- Myqueue keeps track of failures, successes, etc.

Links

- Documentation `https://asr.readthedocs.io/`
- Development, bugs `https://gitlab.com/asr-dev/asr`
- Community support on `#asr` on Matrix: `https://app.element.io/#/room/#asr:matrix.org`
- Myqueue: `https://myqueue.readthedocs.io/`