Running cohere phasing with command line user's scripts

Before using the cohere tools make sure that the installation of cohere package is completed, and that conda environment where it was installed is activated. Refer to https://cdi.readthedocs.io/en/latest/installation.html for installation instructions. The user's scripts should be installed.

The scripts use configuration files to control the flow of execution. Refer to https://cdi.readthedocs.io/en/latest/configuration.html for detailed description of configured parameters.

Below are instructions describing how to use the scripts from the command line. It is assumed the user's current directory is cohere-scripts (or cohere-scripts-main, if it was installed to this directory) that is parent to "scripts" directory. The scripts can be called from any directory, but the path must be modified to point to the scripts.

1. Create a new experiment

This script creates a new experiment directory in the working directory. The experiment directory is determined by <id> and <scan range> parameters, such as <id>_< scan range>. Under the experiment directory the script creates initial configuration in conf subdirectory with all configuration files. The files contain all supported parameters, but only the basic parameters are active, and the rest are commented out. One needs to edit the configuration files to change the functionality.

```
python scripts/create_experiment.py <id> <scan range> <working
directory> --specfile <specfile> -- beamline <beamline>
```

The parameters are as follows:

- id: an arbitrary literal value assign to this experiment
- scan range: scans that will be included in the data. This can be a single scan or range separated with "-"
- working_dir: a directory where the experiment will be created
- specfile: spec file recorded during experiment, optional, but typically used
- beamline: optional, specifies at which beamline the experiment was performed. If not configured, the preparation and visualization is not available.

example:

```
python scripts/create_experiment.py tst 54 workspace --specfile
example/example.spec –-beamline aps_34idc
```

This will create tst_54 directory in the workspace.

2. Another option to create new experiment is to base it on the configuration of existing experiment.

If there is already an experiment created with configuration files that you like and you wish to create a new experiment with a copy of the files, use this script. The experiment will be created in current directory. One can edit the configuration files to change the functionality.

```
python scripts/setup_34idc.py <id> <scan range> <conf dir> --specfile
<specfile> --copy_prep
```

The parameters are as follows:

- id: an arbitrary literal value assign to this experiment
- scan range: scans that will be included in the data. This can be a single scan or range separated with "-"
- conf_dir: a directory from which the configuration files will be copied
- specfile: optional, used when specfile configured in <conf_dir>/config file should be replaced by another specfile
- copy_prep: this is a switch parameter, set to true if the prep_data.tif file should be copied from experiment with the <conf_dir> into the prep directory of the newly created experiment

example:

```
python scripts/setup_34idc.py workspace/tst1 54 tst_54/conf --specfile
example/example.spec
```

This will create tst1_54 directory in the workspace directory. Note: the tst_54 experiment must exist.

3. Prepare the raw data for further processing and reconstruction.

This script reads raw data, applies correction based on physical properties of the detector, and optionally aligns and combines multiple scans. As a result, the prepared data file is stored in <experiment_dir>/prep/prep_data.tif file. This script uses main configuration file <experiment_dir>/conf/config and <experiment_dir>/conf/config_prep. Refer to https://cdi.readthedocs.io/en/latest/configuration.html for parameters description.

```
python scripts/run_prep.py <experiment_dir>
```

The parameters are as follows:

- experiment directory: directory of the experiment space

example:

```
python scripts/run_prep.py example/scan_54
```

This will create prepared data file example/scan_54/ prep/prep_data.tif.

4. Format data

This script reads the prepared data prep/prep_data.tif file and formats the data according to configured parameters. As a result, the prepared data file is stored in in <experiment_dir>/data/data.tif file. This script uses main configuration file <experiment_dir>/conf/config and <experiment_dir>/conf/config_data. Refer to https://cdi.readthedocs.io/en/latest/configuration.html for parameters description.

```
python scripts/format_data.py <experiment_dir>
```

The parameters are as follows:

- experiment directory: directory of the experiment space

example:

```
python scripts/format_data.py example/scan_54
```

This will create formatted data file example/scan_54/ data/data.tif.

5. Run reconstruction

This script reads the formatted data file and runs the reconstruction. The reconstruction results are saved in <experiment_dir>/results directory. Note: The results might be saved in different location in experiment directory space, depending on the use case. Refer to https://cdi.readthedocs.io/en/latest/experiment_space.html for details. This script uses main configuration file <experiment_dir>/conf/config and <experiment_dir>/conf/config_rec. Refer to https://cdi.readthedocs.io/en/latest/configuration.html for parameters description.

```
python scan/run_rec.py <processor> <experiment_dir> --rec_id <alternate
reconstruction id>
```

The parameters are as follows:

- processor: the library used when running reconstruction. Possible options:
  - cuda
  - opencl
  - cpu

The “cuda” and “opencl” options will invoke the processing on GPUs, and the “cpu” option on cpu. The best performance is achieved when running cuda library, followed by opencl, depending on the capabilities of your GPU card.

- experiment directory: directory of the experiment space.
- rec_id: optional parameter, when present, the alternate configuration will be used to run reconstruction.

example1:

```
python scripts/run_rec.py opencl example/scan_54
```

This will create reconstructed files: example/results/image.npy, example/results/support.npy, and example/results/coh.npy (if pcdi feature was used).

example2:

You need to have an alternate configuration file in the configuration directory. Copy the config_rec to alt_config_rec file, both in the conf directory, and start reconstruction with the alternate configuration.

```
cp example/scan_54/conf/config_rec example/scan_54/conf/alt_config_rec
python scripts/run_rec.py opencl example/scan_54 --rec_id alt
```

This will create reconstructed files: example/alt_results/image.npy, example/alt_results/support.npy, and example/alt_results/coh.npy (if pcdi feature was used).

6. Process visualization

This script reads the reconstructed files and processes them to create vts files that can be viewed with visualization tools such Paraview. The script will process “image.npy” files that are in the experiment space, or a given file, if –image_file option is used. If “resuls_dir” configuration parameter is included in the config_disp file, the specified directory will be searched for the “image.npy” files, instead of experiment directory, and the identified files will be processed for visualization.

```
python scripts/run_disp.py <experiment_dir> --image_file <image_file>
```

The parameters are as follows:

- experiment directory: directory of the experiment space
- image_file: optional parameter, if given, only this file will be processed

example:

```
python scripts/run_disp.py example/scan_54
```

This will create files ready for visual interpretation in vts format: example/scan_54/results/image.vts, example/scan_54/results/support.vts, and optionally example/scan_54/results/coh.vts.

7. Run all the parts with one script

The parameters are as follows:

- processor: the library used when running reconstruction. Possible options:
  - cuda (not available on Mac)
  - opencl
  - cpu

The “cuda” and “opencl” options will invoke the processing on GPUs, and the “cpu” option on cpu. The best performance is achieved when running cuda library, followed by opencl. Cuda option is not available on mac.

- experiment directory: directory of the experiment space
- rec_id: optional parameter, when present, the alternate configuration will be used to run reconstruction

```
python scripts/everything.py <processor> <experiment_dir> --rec_id
<alternate reconstruction id>
```

The parameters are as follows:

- processor: the library used when running reconstruction. Possible options:
  - cuda
  - opencl
  - cpu

The “cuda” and “opencl” options will invoke the processing on GPUs, and the “cpu” option on cpu. The best performance is achieved when running cuda library, followed by opencl.

- experiment directory: directory of the experiment space
- rec_id: optional parameter, when present, the alternate configuration will be used to run reconstruction

example:

```
python scripts/everything.py example/scan_54
```

Assuming the experiment scan_54 has been created, this will create files ready for visual interpretation in vts format: example/scan_54/results/image.vts, example/scan_54/results/support.vts,