Basic Installation

Download from https://www.enthought.com/products/canopy either the free or academic version of Enthought Canopy. Your Python version must be 2.7 or newer.

Method 1:

- Install by running `git clone https://github.com/samedling/MCSAS.git` to download the entire repository.
- For future updates, run `git pull origin master` from the MCSAS folder. You may also need to manually remove your old fastmath.so file.

Method 2:

- Go to https://github.com/samedling/MCSAS and click Download ZIP in the lower right to install. Update the same way.

There are two optional but recommended ways of speeding up the code:

1. (Strongly recommended:) Install PyOpenCL following the directions at http://wiki.tiker.net/PyOpenCL/Installation and then the first time your run it, it will ask you which platform and device you want to use. Try the GPU first; if it doesn't work, use the CPU. On my dual core CPU, I obtained a 25x speedup (and during fitting enabled another 5x speedup for a total of ~125x); quad core CPUs should be nearly twice as fast and GPUs should be even faster!
2. F2Py acceleration should work automatically on OS X or Linux. If not, or if you want to make sure you get every bit of performance you can and you have gfortran installed, run `make` to compile the Fortran code (or if you have ifort installed, edit the makefile before running `make`). I've found this useful even if you are running OpenCL on the CPU (it might be unnecessary if running OpenCL on the GPU). On my dual core CPU, I obtained an 8x speedup (and during fitting another 10x speedup for ~80x); quad core CPUs likely not much faster. There is a small benefit to having this in addition to OpenCL.

Run `python newgui.py` on the command line or open it in Canopy and click run. (Note: you may discover running `nice python newgui.py` results in your system being a lot more responsive.)

OS X Fortran Installation

Apple doesn't provide a recent version of gfortran, but you can download one from http://hpc.sourceforge.net

If you also don't have Xcode Tools, follow the directions at https://wiki.helsinki.fi/display/HUGG/Installing+the+GNU+compilers+on+Mac+OS+X

1. Install XCode Tools from the App Store.
2. Install the Command Line Tools by running xcode-select --install
3. Download the latest stable gfortran version from http://hpc.sourceforge.net

Tested on OS X 10.10 "Yosemite".

Some Troubleshooting

If things ran fine before, but after updating returns KeyError, remove the file called default.txt.

OS X and Windows: You need to close all the old plots before you can run things again. Otherwise, it's otherwise unresponsive for some reason.

OS X/EPD/Tkinter: Make sure you have Canopy. EPD might tell you it's updated everything, but it's still not the same as Canopy.

If you receive compiler OpenCL compiler warnings when starting the program it's probably due to your OpenCL device not supporting 64-bit floating point numbers; it should be fine, but if you get errors later, try using a different OpenCL device.

Linux/OpenCL: apt-get on Ubuntu wasn't helpful to me; follow the directions linked above for more success.

PIL: On older systems you may need to manually remove PIL and install Pillow (`sudo pip uninstall PIL` and `sudo pip install Pillow`); newer systems should simply come with Pillow. Otherwise Image won't be able to read the funny SAXS TIF files.

Scientific Linux/F2PY: Make sure you are using the version of F2PY which matches your version of Python (2.7+), otherwise just loading the Fortran module causes a Segmentation Fault.

Fortran/Hyperthreading: If you have a Core i7 processor, (or other CPU with hyperthreading) performance may be slightly improved by adding `export OMP_NUM_THREADS=<n>` to your .bash_profile (where n = the number of physical cores).

Due to the large number of shapes, it's possible some of the sped-up versions of these have bugs. If you find that the shapes don't look right, edit the global_vars.py file so `accelerate_points = False` to disable the erroneous speedup.

Running the Program

Individual Monte Carlo Calculations

'Real Space' will show you the points.

'Calculate Intensity' will show you the detector image.

To activate/inactivate the advanced options, toggle the Simple Options/Advanced Options

button at the top of the center column.

The Radial Symmetry and Small Angle Approx. checkboxes speed the program, so check them if appropriate.

Performing Fits

1. Input the name of the experimental data file to fit and click "Plot Exp Data". If "Center of Beamstop" is left blank ("0 0") then it will plot the original experimental data (which takes a minute). The lower bounds option in the center column is quite useful here. Try a value in the range 1e-8 to 1e-6. Then, move the mouse over the center of the beamstop and read the x,y-coordinates from the plot screen. Use these values and replot the experimental data. It will crop a sqaure around the center and downsample it so the side length is equal to the Pixels parameter.
2. Input known values, uncheck relevant parameter boxes, make a good guess of unknown parameters. To see how good your guess is, press "Plot Residuals".
3. When you have a satisfactory guess, click "Fit Exp Data". Make sure that the update interval isn't too small, or it will actually take longer and/or make no progress. Each iteration, it prints out the time and the sum of the residuals; be aware that it is normal for the sum of the residuals to go several iterations without changing significantly.
4. Read the fit results from the terminal. If you had a grid compression >1 (assuming you're using Fortran acceleration) and now you want more printable results, copy the fit results back into the GUI and Plot Residuals.

Some comments: * Grid compression only works with reliably with fortran; it works with OpenCL if the number of points/pixels does not exceed OpenCL's capabilities. * If the fit steps are each taking less than 10 seconds, there would probaly be very little additional time taken by increasing pixels by 40% or halving the grid compression or z_scaling.

Adding Models

First, make sure you have the most recent version.

To add a Monte Carlo model:

1. Open density_formula.py. At the very bottom of the file, create another "elif:" block like the ones above it. Remember the parameters your function uses. Remember the number you assigned. Save and close the file.
2. Open newgui.py and go to the line defining MC_num_and_name. After the last number (currently around line 80), add a line like the ones above it using the number from step 1.
3. Go to the beginning of the Fit_Parameter class definition In the elif block (currently around line 450), add a pair of lines with the number and the parameters from step 1. Save and close the file.

Do not add to the middle of the list as this will cause errors when Fortran or OpenCL are enabled.

To add an analytic model:

1. Open analytic_formula.py. Near the bottom of the file but above where theory_csv is defined, create another "elif:" block like the ones above it. Remember the number you assigned. Save and close the file.
2. Open newgui.py and go to the line defining Analytic_options. After the last number (currently around line 90), add a line like the ones above it using the number from step 1. Save and close the file.

Finally (after some testing), use git to add/commit/push (see below for details) or e-mail a collaborator to do it for you.

Uploading Changes with Git

Git Setup

Make sure you have git and a github account.

E-mail scott.medling@anu.edu.au with your username so I can add you as a collaborator.

Optionally, to configure your local git, run

```
git config --global user.name "<Name>"
git config --global user.email "<E-mail Address>"
git config --global color.ui auto       #Improves readability.
git config --global core.editor vim     #If you like vim.
```

To download the respository, run

```
git clone https://github.com/samedling/MCSAS.git
```

Basic Git Use

Every time you edit

```
git pull origin master    #Download latest changes.  Run every time you start.
```

After editing a file, or number of files

```
git add <filename>        #Adds a filename
git add <filename2>       #Adds another filename, etc.

git status                #Tells you which files have been changed/added.

git commit -m "<Insert short message here.>"  #Saves added changes locally.
git push -u origin master #Uploads committed changes to respository.
```

Other Useful Git Commands

```
git log --oneline     #Displays summary of each commit.
git log -<n>          #Displays last n commit details.
git log --after="<yyyy-mm-dd>"
```

Advanced Git Use

If there's a collision/conflict/whatever (usually at the push stage) because you and someone both editing the same part of the same file, you'll need to manually fix it, which sometimes sucks. You may need to separately run

```
git fetch origin master
git merge
```

If you want to go back just to look, make sure you've committed any changes and then run

```
git checkout <hex_number>
```

To create a new branch so you can make commits based on an older version (again, make sure you've committed any changes), run

```
git checkout -b <branch_name> <hex_number>
```

For potentially unstable changes, use the develop branch:

```
git checkout -b <branch_name>   #Creates branch from current commit.
git checkout develop        #Switch branches.
```

Make your changes, commits, etc. Then merge:

```
git checkout master         #Switch back to master.
git merge develop --no-ff   #Merge while preserving commit history.
git push
```

Copyright

Copyright 2015 Max Proft and Scott Medling, Australian National University. All rights reserved.