SIMPLE 2017 Development Document

Responsibilities: Hans, Cyril, Dominika&Hans, Alex&Hans, Susan&Hans

* Restart mode for workflows, i.e. keep a status file that can replace the command line when given startit, should be possible to start from any previous iteration without having to give command line directives
* Continuum optimisation-based refinement in both Cartesian and Polar coordinates
* PRIME3D local refinement mode (dynamic continuous sampling around previous orientations using polar coordinates and the same weighting scheme as previously deployed)
* Outer product trick for Cartesian/Polar central section extraction and Cartesian 3D reconstruction + performance tests that are stringent enough that they can be described in a paper
* Preparing method paper for Protein Science: Improved ab initio 3D reconstruction from single-particle cryo-EM images with SIMPLE/PRIME
* Getting the masking routine in shape
* Help Marion interpret her SAGA maps
* Think about and develop a down-scaling strategy for prime2D/3D (inspired by cryo sparc?)
* Investigate the branch and bound approach used in cryo sparc to see if that is something we can use in SIMPLE
* Code a distributed workflow for multiptcl\_init (takes forever for large particle stacks)
* Pseudo symmetry refinement mode?
* when automsk is turned on in 2D we must force the next round to search all refs, or it will go to shit
* class for memory allocation simple\_alloc. Keep track of mem exceptions and profile memory (hash table).
* Go over all random number generations and see if we can use matrices instead
* Replace integer random number generator with intrinsic one (Guide to Fortran 2008)
* mailx -s "mail from prime2D" hans.elmlund@monash.edu < from\_prime2D
* need to break out if simple\_path in \*.env file and bash not identical
* chunk mode for prime2D

GPU

1. UNBLUR
2. PRIME2D

3) PRIME3D

Need support for the CTF formalism on GPU.

How do we deal with memorize sqsums with CTF on GPU?

How do we deal with leftover in chunk-based parallelisation?

Is chunk-based GPU execution the way forward?

If so, how does this splitter deal with nstates?

Tests

* Juha's code (icosahedrality of individual particle images)
* test that the Wiener filter works for initial model generation from particles
* evaluate stochastic momentum on the corr level
* evaluate deterministic momentum on the rec level. This is basically what Relion does with the slidebar where you select how much weight you want give to the reference vs. the data. This can be useful when you have “problematic data” (for example strong view preference)
* Need to test the knn structure 4 2D (million ribosomes)

Streaming

1. UNBLUR
2. PRIME2D

Ideas

* Create a class scoring function based on the sum of the integrated weights of the class they belong to (minimum population threshold must apply). Correlate with the visual assessment of the cavg quality.
* COMPICK: code the common lines based idea for particle picking
* COMSEL: code the common lines based idea for cavg selection for ab initio rec

New developments

* Probabilistic SO(3) scatter search for high-resolution refinement
* Quasi-continuous sorting of conformational states (affinity propagation, shell-weighting or whatever works)
* implement tilt test

Exception handling

* exception handling class that makes more sensible outputs (especially when running the code in distributed mode). We need to create a database of exceptions and then have ONE control point where we check the stack for errors and report what has failed (similar to the command line dictionary)
* refs should be included in the mixed formats check
* when nr of commands are not sufficient the error message should tell what is missing
* when eo=yes is set the program should ask for fsc-file if missing & instruct better

Refinement Considerations

Using the strategy pattern for the different modes of refinement?

When do we turn on neigh modes?

Should npeaks in refine=no/neigh modes always be at least 6?

Should we test qcont? My feeling is that it is too slow.

Known bugs

None at the moment

Compilation

* port to ifort (Intel compiler)
* port to PGI (Portland group, with CUDA-FORTAN)

Documentation

Need to get a html code doc generator in place and fix the doc of every class. Will FORD source code documentation provide the solution?

<http://fortranwiki.org/fortran/show/FORD>

<https://github.com/cmacmackin/ford>

<http://jacobwilliams.github.io/json-fortran/index.html>

Books/webpages

* Structured parallel programming
* Structure and Interpretation of Computer Programs
* J-P Morrison. Flow-Based Programming: A New Approach to Application Developments. CreateSpace, 2nd ed.
* Mathematical Foundations of Imaging, Tomography and Wavefield Inversion
* Geometric Algebra for Computer Science (Revised Edition): An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics) 1st Edition
* The Princeton Companion to Applied Mathematics
* Practical Machine Learning: http://www.computervisionmodels.com/
* Applied Stochastic Modelling, Second Edition (Chapman & Hall/CRC Texts in Statistical Science) 2nd Edition http://szeliski.org/Book/

Considerations

Should we implement a resolution update scheme for prime2D similar to that in the Unblur code?