EMDA

Release 1.1.3

Rangana Warshamanage, Garib N. Murshudov

Jan 06, 2021

CONTENTS:

CHAPTER

ONE

EMDA METHODS MODULE

Author: "Rangana Warshamanage, Garib N. Murshudov" MRC Laboratory of Molecular Biology

This software is released under the Mozilla Public License, version 2.0; see LICENSE.

emda_methods.**apply_bfactor_to_map**(*mapname*, *bf_arr*, *mapout*)

Applies an array of B-factors on the map.

Arguments:

Inputs:

mapname: string Map file name.

bf_arr: float, 1D array An array/list of B-factors.

mapout: bool If True, map for each B-factor will be output.

Outputs:

all_mapout: complex, ndarray 4D array containing Fourier coefficients of all maps. e.g. all_mapout[:,:,:,i], where i represents map number corresponding to the B-factor in bf_arr.

emda_methods.**applymask**(*mapname*, *maskname*, *outname*)

emda_methods.**average_maps**(*maplist*, *rot=0.0*, *ncy=5*, *res=6*, *interp='linear'*, *fit=True*, *tra=None*, *axr=None*, *fobj=None*, *masklist=None*)

Calculates the best average maps using Bayesian principles.

Calculates the best average map using Bayesian principles. This is done in two steps; 1. Parameter estimation using a likelihood function, 2. Best map calculation. Parameter estimation is similar to map overlay where each map is brought onto static map by maximizing the overlap. The best maps are calculated using superimposed maps.

Arguments:

Inputs:

maplist: list List of half maps to average.

masklist: list, optional List of masks to apply on maps. len(masklist) == len(maplist) // 2

rot: float, optional Initial rotation in degrees. Default is 0.0.

axr: list, optional Rotation axis. Default is [1, 0, 0].

tra: list, optional Translation vector in fractional units. Default is [0.0, 0.0, 0.0]

res: float, optional Fit start resolution in Angstrom units. Default is 6.0 Angstrom.

ncy: integer, optional Number of fitting cycles. Default is 5.

interp: string, optional Interpolation type either "linear" or "cubic". Default is linear.

fobj: string File object for logging. If None given, EMDA_average.txt will be output.

fit: bool, optional If True, map fitting will be carried out before average map calculation. Default is True.

Outputs: Outputs a series of overlaid maps (**fitted_map_**?.mrc). Also, outputs a series of average maps (**avgmap_**?.mrc)

emda_methods.**balbes_data**(*map1name*, *map2name*, *fsccutoff=0.5*, *mode='half'*)

Returns data required for Balbes pipeline.

Required data is output with their references in EMDA.xml.

Arguments:

Inputs:

map1name: string Name of the map 1.

map2name: string Name of the map 2.

fsccutoff: float, optional FSC of desired resolution. Defualt is 0.5

mode: string Mode can be either 'half' or 'any'. If the input maps are the half maps, mode should be 'half'. Otherwise, mode should be 'any'. Default mode is half.

Outputs: Outputs EMDA.xml containing data and references to other data. No return variables.

emda_methods.**bestmap**(*hf1name*, *hf2name*, *outfile*, *mode=1*, *knl=5*, *mask=None*)

emda_methods.**center_of_mass_density**(*arr*)

Returns the center of mass of 3D density array.

This function accepts density as 3D numpy array and caclulates the center-of-mass.

Arguments:

Inputs: arr: density as 3D numpy array

Outputs: com: tuple, center-of-mass (x, y, z)

emda_methods.**compositemap**(*maps*, *masks*)

emda_methods.**difference_map**(*maplist*, *smax=0.0*, *mode='ampli'*, *masklist=None*)

Calculates difference map.

Arguments:

Inputs:

maplist: string List of map names to calculate difference maps.

masklist: string, optional List of masks to apply on maps.

smax: float, optional Resolution to which difference map be calculated.

mode: string, optional Different modes to scale maps. Three difference modes are supported. 'ampli' - scale between maps is based on amplitudes [Default]. 'power' - scale between maps is based on powers (intensities). 'norm' - normalized maps are used to calculate difference map.

Outputs: Outputs diffence maps and initial maps after scaling in MRC format.

emda_methods.**estimate_map_resol**(*hfmap1name*, *hfmap2name*)

Estimates map resolution.

Arguments:

Inputs:

hfmap1name: string Halfmap 1 name.

hfmap2name: string Halfmap 2 name.

Outputs:

map_resol: float Map resolution determined by the halfmap FSC.

emda_methods.**fetch_data**(*emdbidlist*, *alldata=False*)

emda_methods.**fouriersp_correlation**(*half1_map*, *half2_map*, *kernel_size=5*, *mask=None*)

Calculates Fourier space local correlation using half maps.

Arguments:

Inputs:

half1_map: string Name of half map 1.

half2_map: string Name of half map 2.

kernel_size: integer, optional Radius of integration kernal. Default is 5.

Outputs: Following maps are written out: fouriercorr3d_halfmaps.mrc - local correlation in half maps. fouriercorr3d_fullmap.mrc - local correlation in full map

using the formula 2 x FSC(half) / (1 + FSC(half)).

fouriercorr3d_truemap.mrc - local correlation in true map. Useful for validation purpose.

emda_methods.**get_biso_from_map**(*halfmap1*, *halfmap2*)

Calculates the isotropic B-value of the map.

Arguments:

Inputs:

halfmap1: string Halfmap 1 file name.

halfmap2: string Halfmap 2 file name.

Outputs:

biso: float Map B-iso value.

emda_methods.**get_biso_from_model**(*mmcif_file*)

Calculates the isotropic B-value of the model.

Arguments:

Inputs:

mmcif_file: string mmCIF file name.

Outputs:

biso: float Model B-iso value.

emda_methods.**get_data**(*struct*, *resol=5.0*, *uc=None*, *dim=None*, *maporigin=None*)

Returns data of a map or a model into an ndarray.

Reads map data into an ndarray, or if the structure input is an atomic model, it calculates the map from the model and returns as an ndarray.

Arguments:

Inputs:

struct: string CCP4/MRC map file name or PDB/ENT/CIF file resol: float, optional

resolution to calculates map from model. Default is 5.0 A.

uc: float, 1D array Parameter for modelmap generation. If absent, this will be determined by dim parameter.

dim: sequence (integers), optional Parameter for modelmap generation. If absent, this will be determined from the size of the molecule.

maporigin: sequence (integers), optional Parameter for modelmap generation. If present, the calculated map will be shifted according to this information. If absent, this parameter is taken as [0, 0, 0].

Outputs:

uc: float, 1D array Unit cell

arr: float, 3D array Map values as Numpy array

origin: list Map origin list

emda_methods.**get_dim**(*model*, *shiftmodel='new1.cif'*)

Returns the box dimension to put the modelmap in.

Determines the dimension of the box for the model based map.

Arguments:

Inputs: model: atomic model as .pdb/.cif shiftmodel: name for COM shifted model, optional.

Default name - new1.cif.

Outputs: dim: integer, dimension of the box.

emda_methods.**get_fsc**(*arr1*, *arr2*, *uc*)

Returns FSC as a function of resolution

Arguments:

Inputs:

arr1: float, ndarray Density array 1.

arr2: float, ndarray Density array 2.

uc: float, 1D array Unit cell

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array Linear array of FSC in each resolution bin.

emda_methods.**get_map_power**(*mapname*)

Calculates the map power spectrum.

Arguments:

Inputs:

mapname: string Map name.

Outputs:

res_arr: float Resolution array.

power_spectrum: float Map power spectrum.

emda_methods.**get_variance**(*half1name*, *half2name*, *filename=None*, *maskname=None*)
Returns noise and signal variances of half maps.

Returns noise and signal variances of half maps. Return values are not corrected for full map.

Arguments:

Inputs:

half1name: string Name of the half map 1.

half2name: string Name of the half map 2.

filename: string If present, statistics will be printed into this file.

maskname: String If present, input maps will be masked before computing variances.

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

noisevar: float, 1D array Linear array of noise variance in each resolution bin.

signalvar: float, 1D array Linear array of signal variance in each resolution bin.

emda_methods.**half2full**(*half1name*, *half2name*, *outfile='fullmap.mrc'*)
Combines half maps to generate full map.

Arguments:

Inputs:

half1name: string Name of half map 1.

half2name: string name of half map 2.

outfile: string, optional Name of the output file. Default is fullmap.mrc

Outputs:

fullmap: float, 3D array 3D Numpy array of floats.

emda_methods.**halfmap_fsc**(*half1name*, *half2name*, *filename=None*, *maskname=None*)
Computes Fourier Shell Correlation (FSC) using half maps.

Computes Fourier Shell Correlation (FSC) using half maps. FSC is not corrected for mask effect in this implementation.

Arguments:

Inputs:

half1name: string Name of the half map 1.

half2name: string Name of the half map 2.

filename: string If present, statistics will be printed into this file.

maskname: String If present, input maps will be masked before computing FSC.

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array Linear array of FSC in each resolution bin.

emda_methods.**halfmap_fsc_ph**(*half1name*, *half2name*, *filename='halffsc.txt'*, *maskname=None*)
Computes Fourier Shell Correlation (FSC) using half maps.

Computes Fourier Shell Correlation (FSC) using half maps. FSC is not corrected for mask effect in this implementation.

Arguments:

Inputs:

half1name: string Name of the half map 1.

half2name: string Name of the half map 2.

filename: string If present, statistics will be printed into this file.

maskname: String If present, input maps will be masked before computing FSC.

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array Linear array of FSC in each resolution bin.

emda_methods.**lowpass_map**(*uc*, *arr1*, *resol*, *filter='ideal'*, *order=4*)

Lowpass filters a map to a specified resolution.

This function applies a lowpass filter on a map to a specified resolution. This operations is carried out in the Fourier space. Note that lowpass map will have the same shape as input data.

Arguments:

Inputs:

uc: float, 1D array Unit cell params.

arr1: float, 3D array 3D Numpy array containing map values.

resol: float Resolution cutoff for lowpass filtering in Angstrom units.

filter: string, optional Fiter type to use in truncating Fourier coefficients. Currently, only 'ideal' or 'butterworth' filters can be employed. Default type is ideal.

order: integer, optional Order of the Butterworth filter. Default is 4.

Outputs:

fmap1: complex, 3D array Lowpass filtered Fourier coefficeints.

map1: float, 3D array Lowpass filtered map in image/real space

emda_methods.**map2mtz**(*mapname*, *mtzname='map2mtz.mtz'*)

Converts a map into MTZ format.

Arguments:

Inputs:

mapname: string Map file name.

mtzname: string Output MTZ file name. Default is map2mtz.mtz

Outputs: Outputs MTZ file.

emda_methods.**map2mtzfull**(*uc*, *arr1*, *arr2*, *mtzname='halfnfull.mtz'*)

Writes several 3D Numpy arrays into an MTZ file.

This function accepts densities of two half maps as 3D numpy arrays and outputs an MTZ file containing amplitudes of half1, half2 and full map. The outfile data labels are H, K, L, Fout1, Fout2, Foutf, Poutf. The last four labels correspond to amplitude of halfmap1, amplitudes of halfmap2, amplitudes of fullmap and the phase values of fullmap, respectively.

Arguments:

Inputs:

uc: float, 1D array Unit cell params.

arr1: float, 3D array Half1 map values.

arr2: float, 3D array Half2 map values.

mtzname: string, optional Output MTZ file name. Default is halfnfull.mtz

Outputs: Outputs an MTZ file containing amplitudes of half maps and full map.

emda_methods.**map_model_validate**(*half1map*, *half2map*, *modelfpdb*, *bfac=0.0*, *lig=True*, *model1pdb=None*, *mask=None*, *modelresol=None*, *lgf=None*)

Calculates various FSCs for maps and model validation.

Arguments:

Inputs:

half1map: string Name of half map 1.

half2map: string Name of half map 2.

modelfpdb: string Name of the model refined against full map in cif/pdb/ent formats.

model1pdb: string, optional Name of the model refined against one half map in cif/pdb/ent formats. If included, FSC between that and half maps will be calculated.

mask: string, optional Name of the mask file. It will apply on half maps before computing FSC. If not included, a correlation based masked will employed.

modelresol: float, optional An argument for model based map calculation using REFMAC. Resolution to calculate model based map. If not specified, an FSC based cutoff will be used.

bfac: float, optional An overall B-factor for model map. Default is 0.0

lig: bool, optional An argument for model based map calculation using REFMAC. Set True, if there is a ligand in the model, but no description. Default is True.

lgf: string, optional An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs:

fsc_list: list List of FSCs is returned. If len(fsc_list) is 4, FSC lables are as follows: 0 - half maps FSC 1 - half1map - model1 FSC 2 - half2map - model1 FSC 3 - fullmap-fullmodel FSC If len(fsc_list) is 2, only 0 and 3 contains.

Outputs FSCs in allmap_fsc_modelvsmap.eps

emda_methods.**map_transform**(*mapname*, *tra*, *rot*, *axr*, *outname='transformed.mrc'*)

Imposes a transformation on a map.

Imposes a transformation (i.e. translation and rotation) on a map and returns the transformed map.

Arguments:

Inputs:

mapname: string Name of the input map.

tra: list of three floats values Translation vector as a list in Angstrom units.

rot: float Rotation to apply in degrees.

axr: list of three integers Axis to rotation. e.g [1, 0, 0]

outname: string, optional Name of the transformed map. Default is transformed.mrc.

Outputs:

transformed_map: float, 3D array 3D Numpy array of floats.

emda_methods.**mapmagnification**(*maplist*, *rmap*)

emda_methods.**mapmodel_fsc**(*map1*, *model*, *fobj*, *bfac=0.0*, *modelresol=5.0*, *lig=True*, *phaserand=False*, *mask=None*, *lgf=None*)

emda_methods.**mask_from_halfmaps**(*uc*, *half1*, *half2*, *radius=9*, *norm=False*, *iter=1*, *thresh=0.5*)

Generates a mask from half maps.

Generates a mask from half maps based on real space local correlation.

Arguments:

Inputs:

uc: float, 1D array Unit cell parameters.

half1: float, 3D array Half map 1 data.

half2: float, 3D array Half map 2 data.

radius: integer, optional Radius of integrating kernel in voxels. Default is 9.

norm: bool, optional If true, normalized maps will be used to generate correlation mask. Default is False.

iter: integer,optional Number of dilation cycles. Default is 1 cycle.

thresh: float, optional Correlation cutoff for mask generation. Program automatically decides the best value, however, user can overwrite this.

Outputs:

mask: float, 3D array 3D Numpy array of correlation mask.

emda_methods.**mask_from_map**(*uc*, *arr*, *kern=5*, *resol=15*, *filter='butterworth'*, *order=1*, *prob=0.99*, *itr=3*, *orig=None*)

Generates a mask from a map.

Generates a mask from a map.

Arguments:

Inputs:

uc: float, 1D array Unit cell parameters.

arr: float, 3D array Map data.

half2: float, 3D array Half map 2 data.

kern: integer, optional Radius of integrating kernel in voxels. Default is 5.

resol: float, optional Resolution cutoff for lowpass filtering in Angstrom units. Default is 15 Angstrom.

filter: string,optional Filter type to use with lowpass filtering. Default is butterworth.

order: integer, optional Butterworth filter order. Default is 1.

prob: float, optional Cumulative probability cutoff to decide the density threshold. Default value is 0.99.

itr: integer, optional Number of dilation cycles. Default is 3 cycles.

orig: list of three integer values. Map origin. e.g. [0, 0, 0]

Outputs:

mask: float, 3D array 3D Numpy array of the mask.

Outputs lowpass.mrc and mapmask.mrc files.

emda_methods.**mirror_map**(*mapname*)

emda_methods.**model2map**(*modelxyz*, *dim*, *resol*, *cell*, *bfac=0.0*, *lig=True*, *maporigin=None*, *ligfile=None*)

emda_methods.**mtz2map**(*mtzname*, *map_size*)

Converts an MTZ file into MRC format.

This function converts data in an MTZ file into a 3D Numpy array. It combines amplitudes and phases with "Fout0" and "Pout0" labels to form Fourier coefficients. If the MTZ contains several aplitude columns, only the one corresponding to "Fout0" will be used.

Arguments:

Inputs:

mtzname: string MTZ file name.

map_size: list Shape of output 3D Numpy array as a list of three integers.

Outputs: outarr: float 3D Numpy array of map values.

emda_methods.**overall_cc**(*map1name*, *map2name*, *space='real'*, *resol=5*, *maskname=None*)

emda_methods.**overlay_maps**(*maplist*, *rot=0.0*, *ncy=5*, *res=6*, *interp='linear'*, *hfm=False*, *masklist=None*, *tra=None*, *axr=None*, *fobj=None*, *usemodel=False*, *fitres=None*)

Superimposes several maps.

Superimposes several maps using a likelihood function. All maps are overlaid on the first map.

Arguments:

Inputs:

maplist: list List of maps to overlay.

masklist: list List of masks to apply on maps.

rot: float, optional Initial rotation in degrees. Default is 0.0.

axr: list, optional Rotation axis. Default is [1, 0, 0].

tra: list, optional Translation vector in fractional units. Default is [0.0, 0.0, 0.0]

res: float, optional Fit start resolution in Angstrom units. Default is 6.0 Angstrom.

ncy: integer, optional Number of fitting cycles. Default is 5.

interp: string, optional Interpolation type either "linear" or "cubic". Default is linear.

hfm: bool, optional If True, overlay will be carried out on half maps. In this case, maplist will contain half maps. e.g. [map1_half1.mrc, map1_half2.mrc, map2_half1.mrc, map2_half2.mrc, ...]. masklist will contain masks for each map. e.g. [map1_mask.mrc, map2_mask.mrc]. The length of masklist should be equal to half the length of maplist. If False, uses full maps for overlay. Default is False.

fobj: string File object for logging. If None given, EMDA_overlay.txt will be output.

Outputs: Outputs a series of overlaid maps (**fitted_map_**?.mrc).

emda_methods.**predict_fsc**(*hf1name*, *hf2name*, *nparticles=None*, *bfac=None*, *mask=None*)

emda_methods.**prepare_refmac_data**(*hf1name*, *hf2name*, *outfile='fullmap.mtz'*, *bfac=None*, *maskname=None*, *xmlobj=None*, *fsccutoff=None*)

emda_methods.**read_atomsf**(*atom*, *fpath=None*)

emda_methods.**read_map**(*mapname*)

Reads CCP4 type map (.map) or MRC type map.

Arguments:

Inputs:

mapname: string CCP4/MRC map file name

Outputs:

uc: float, 1D array Unit cell

arr: float, 3D array Map values as Numpy array

origin: list Map origin list

emda_methods.**read_mtz**(*mtzfile*)

Reads mtzfile and returns unit_cell and data in Pandas DataFrame.

Arguments:

Inputs:

mtzfile: string MTZ file name

Outputs:

uc: float, 1D array Unit cell

df: Pandas data frame Map values in Pandas Dataframe

emda_methods.**realsp_correlation**(*half1map*, *half2map*, *kernel_size=5*, *norm=False*, *lig=True*, *model=None*, *model_resol=None*, *mask_map=None*, *lgf=None*)

Calculates local correlation in real/image space.

Arguments:

Inputs:

half1map: string Name of half map 1.

half1map: string Name of half map 2.

kernel_size: integer, optional Radius of integration kernal in pixels. Default is 5.

norm: bool, optional If True, correlation will be carried out on normalized maps. Default is False.

model: string, optional An argument for model based map calculation using REFMAC. Name of model file (cif/pdb). If present, map-model local correlation will be calculated.

model_resol: float, optional An argument for model based map calculation using REFMAC. Resolution to calculate model based map. If absent, FSC based resolution cutoff will be employed.

mask_map: string, optional Mask file to apply on correlation maps. If not given, correlation based mask will be employed.

lig: bool, optional An argument for model based map calculation using REFMAC. Set True, if there is a ligand in the model, but no description. Default is True.

lgf: string, optional An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs: Following maps are written out: rcc_halfmap_smax?.mrc - reals space half map local correlation. rcc_fullmap_smax?.mrc - correlation map corrected to full map

using the formula 2 x FSC(half) / (1 + FSC(half)).

If a model included, then rcc_mapmodel_smax?.mrc - local correlation map between model and

full map.

rcc_truemapmodel_smax?.mrc - truemap-model correaltion map for validation purpose.

emda_methods.**realsp_correlation_mapmodel**(*fullmap*, *model*, *resol*, *kernel_size=5*, *lig=True*, *norm=False*, *nomask=False*, *mask_map=None*, *lgf=None*)

Calculates real space local correlation between map and model.

Arguments:

Inputs:

fullmap: string Name of the map.

model: string An argument for model based map calculation using REFMAC. Name of model file (cif/pdb/ent/mtz/mrc).

resol: float An argument for model based map calculation using REFMAC. Resolution to calculate model based map.

kernel_size: integer, optional Radius of integration kernal in pixels. Default is 5.

mask_map: string, optional Mask file to apply on correlation maps.

nomask: bool, optional If True, correlation maps are not masked. Otherwise, internally calculated mask is used, if a mask is not supplied.

norm: bool, optional If True, correlation will be carried out on normalized maps. Default is False.

lig: bool, optional An argument for model based map calculation using REFMAC. Set True, if there is a ligand in the model, but no description. Default is True.

lgf: string, optional An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs: Following maps are written out: modelmap.mrc - model based map. rcc_mapmodel.mrc - real space local correlation map.

emda_methods.**resample_data**(*curnt_pix*, *targt_pix*, *targt_dim*, *arr*)

Resamples a 3D array.

Arguments:

Inputs:

curnt_pix: float Current pixel size.

targt_pix: float Target pixel size.

targt_dim: list List of three integer values.

arr: float 3D array of map values.

Outputs:

new_arr: float, 3D array Resampled 3D array.

emda_methods.**rotate_density**(*arr*, *rotmat*, *interp='linear'*)
Returns a rotated array of density

Rotates the array of density using inperpolation.

Arguments:

Inputs: arr: density as 3D numpy array rotmat: 3 x 3 rotation matrix as 2D numpy array. interp: string.

Type of interpolation to use: cubic or linear. Default is linear

Outputs: rotated_arr: ndarray. Rotated array.

emda_methods.**scale_map2map**(*staticmap*, *map2scale*, *outfile*)

emda_methods.**set_dim_equal**(*x*)
Sets all dimentions equal and even

This function accepts 3D numpy array and sets its all 3 dims even and equal

Arguments:

Inputs: x: 3D numpy array

Outputs: x: 3D numpy array with all dims are even and equal

emda_methods.**set_dim_even**(*x*)
Sets all dimentions even

This function accepts 3D numpy array and sets its all 3 dims even

Arguments:

Inputs: x: 3D numpy array

Outputs: x: 3D numpy array with all dims are even

emda_methods.**shift_density**(*arr*, *shift*)
Returns a shifted copy of the input array.

Shift the array using spline interpolation (order=3). Same as Scipy implementation.

Arguments:

Inputs: arr: density as 3D numpy array shift: sequence. The shifts along the axes.

Outputs: shifted_arr: ndarray. Shifted array

emda_methods.**singlemap_fsc**(*map1name*, *knl=3*)
Returns Fourier Shell Correlation (FSC) of a map.

Computes Fourier Shell Correlation (FSC) between a map and its reconstituted other half from neighbough Fourier coefficients. This method can be used to estimate FSC based resolution. However, results seem to be reliable when an unfiltered map is used.

Arguments:

Inputs:

map1name: string Name of the map.

knl: integer, optional Radius of the integrating kernel.

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array Linear array of FSC in each resolution bin.

Outputs reconstituted map as 'fakehalf.mrc'

emda_methods.**sphere_kernel_softedge**(*radius=5*)
Generates a soft-edged spherical kernel.

Arguments:

Inputs:

radius: integer, optional Radius of integrating kernel in voxels. Default is 5.

Outputs:

kernel: float, 3D array 3D Numpy array of spherical kernel.

emda_methods.**symaxis_refine**(*maplist*, *mapoutvar=False*, *emdbidlist=None*)

emda_methods.**twomap_fsc**(*map1name*, *map2name*, *fobj=None*, *xmlobj=None*)
Returns Fourier Shell Correlation (FSC) between any two maps.

Computes Fourier Shell Correlation (FSC) using any two maps.

Arguments:

Inputs:

map1name: string Name of the map 1.

map2name: string Name of the map 2.

fobj: file object for logging If present, statistics will be printed into this file.

xmlobj: xml object If present, statistics will be printed into an XML file.

Outputs:

res_arr: float, 1D array Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array Linear array of FSC in each resolution bin.

emda_methods.**write_mrc**(*mapdata*, *filename*, *unit_cell*, *map_origin=None*)
Writes 3D Numpy array into MRC file.

Arguments:

Inputs:

mapdata: float, 3D array Map values to write

filename: string Output file name

unit_cell: float, 1D array Unit cell params

map_origin: list, optional map origin. Default is [0.0, 0.0, 0.0]

Outputs: Output MRC file

emda_methods.**write_mtz**(*uc*, *arr*, *outfile='map2mtz.mtz'*)
Writes 3D Numpy array into MTZ file.

Arguments:

Inputs:

uc: float, 1D array Unit cell params.

arr: complex, 3D array Map values to write.

Outputs: outfile: string Output file name. Default is map2mtz.mtz.

CHAPTER

TWO

EMDA COMMAND LINE MODULE

```
usage: emda [command] [arguments]
```

2.1 Positional Arguments

command Possible choices: info, halffsc, fsc, singlemapfsc, ccmask, mapmask, lowpass, power, bfac, resol, half2full, map2mtz, map2mtzfull, transform, mtz2map, resample, resamplemap2map, rcc, mmcc, fcc, mapmodelvalidate, mapmodelfsc, overlay, average, diffmap, applymask, scalemap, bestmap, predfsc, refmac, occ, mirror, model2map, composite, magref, com, fetch, symref

2.2 Named Arguments

--version show program's version number and exit

2.3 Sub-commands:

2.3.1 info

Output basic information about the map.

```
emda info [-h] --map MAP
```

Named Arguments

--map input map

2.3.2 halffsc

Calculates FSC between half-maps.

```
emda halffsc [-h] --h1 H1 --h2 H2 [--msk MSK] [--out OUT]
```

Named Arguments

--h1	input map1 map
--h2	input map2 map
--msk	input mask (mrc/map)
--out	output data table Default: "table_variances.txt"

2.3.3 fsc

Calculates FSC between two maps.

```
emda fsc [-h] --map1 MAP1 --map2 MAP2
```

Named Arguments

--map1	input map1 map
--map2	input map2 map

2.3.4 singlemapfsc

Calculates FSC using neighbour average.

```
emda singlemapfsc [-h] --h1 H1 [--knl KNL]
```

Named Arguments

--h1	input map1 map
--knl	kernel radius in voxels Default: 3

2.3.5 ccmask

Generates mask based on halfmaps correlation.

```
emda ccmask [-h] --h1 H1 --h2 H2 [--knl KNL] [--nrm] [--itr ITR] [--thr THR]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--knl	kernel radius in voxels
	Default: 10
--nrm	if True use normalized maps
	Default: False
--itr	number of dilation cycles
	Default: 1
--thr	cc threshold
	Default: 0.5

2.3.6 mapmask

Generate a mask from a map.

```
emda mapmask [-h] --map MAP [--knl KNL] [--prb PRB] [--itr ITR] [--res RES]
             [--fil FIL]
```

Named Arguments

--map	input map
--knl	kernel radius in voxels
	Default: 5
--prb	density cutoff probability in cumulative density function
	Default: 0.99
--itr	number of dilate iterations
	Default: 3
--res	lowpass resolution in Angstroms
	Default: 15.0
--fil	filter type to use: ideal or butterworth
	Default: “butterworth”

2.3.7 lowpass

Lowpass filter to specified resolution.

```
emda lowpass [-h] --map MAP --res RES [--fil FIL]
```

Named Arguments

--map	input map (mrc/map)
--res	lowpass resolution in Angstrom
--fil	filter type to use: ideal or butterworth Default: "ideal"

2.3.8 power

Calculates power spectrum.

```
emda power [-h] --map MAP
```

Named Arguments

--map	input map (mrc/map)

2.3.9 bfac

Apply a B-factor on the map.

```
emda bfac [-h] --map MAP --bfc BFC [BFC ...] [--out]
```

Named Arguments

--map	input map (mrc/map)
--bfc	bfactor(s) to apply
--out	if use, writes out map Default: False

2.3.10 resol

Estimates map resolution based on FSC.

```
emda resol [-h] --h1 H1 --h2 H2
```

Named Arguments

--h1 input halfmap1 map

--h2 input halfmap2 map

2.3.11 half2full

Combine two halfmaps to make the fullmap.

```
emda half2full [-h] --h1 H1 --h2 H2 [--out OUT]
```

Named Arguments

--h1 input halfmap1 map

--h2 input halfmap2 map

--out output map (mrc/map)
Default: "fullmap.mrc"

2.3.12 map2mtz

Convert MRC/MAP to MTZ.

```
emda map2mtz [-h] --map MAP [--out OUT]
```

Named Arguments

--map input map (mrc/map)

--out output map (mtz)
Default: "map2mtz.mtz"

2.3.13 map2mtzfull

Convert MRC/MAP to MTZ using half maps.

```
emda map2mtzfull [-h] --h1 H1 --h2 H2 [--out OUT]
```

Named Arguments

--h1 input hfmap1 (mrc/map)

--h2 input hfmap2 (mrc/map)

--out output map (mtz)
Default: "map2mtzfull.mtz"

2.3.14 transform

Apply a transformation on the map.

```
emda transform [-h] --map MAP [--tra TRA [TRA ...]] [--rot ROT]
               [--axr AXR [AXR ...]] [--out OUT]
```

Named Arguments

--map	input map (mrc/map)
--tra	translation vec. in Angstrom. eg 1.0 0.0 0.0
	Default: [0.0, 0.0, 0.0]
--rot	rotation in degree
	Default: 0.0
--axr	rotation axis
	Default: [1, 0, 0]
--out	output map (mrc/map)
	Default: "transformed.mrc"

2.3.15 mtz2map

Convert MTZ to MRC/MAP.

```
emda mtz2map [-h] --mtz MTZ --map MAP --out OUT
```

Named Arguments

--mtz	input map (mtz)
--map	input map (mrc/map)
--out	output map (mrc/map)

2.3.16 resample

Resample a map in Fourier space.

```
emda resample [-h] --map MAP --pix PIX [--dim DIM [DIM ...]]
              [--cel CEL [CEL ...]] [--out OUT]
```

Named Arguments

--map	input map (mrc/map)
--pix	target pixel size (A)
--dim	target map dimensions. e.g. 100 100 100
--cel	target cell. e.g. a b c 90 90 90
--out	output map name Default: "resampled.mrc"

2.3.17 resamplemap2map

Resample map2 on map1.

```
emda resamplemap2map [-h] --map1 MAP1 --map2 MAP2 [--out OUT]
```

Named Arguments

--map1	static map (mrc/map)
--map2	map to resample (mrc/map)
--out	output map name Default: "resampled2staticmap.mrc"

2.3.18 rcc

real space correlation

```
emda rcc [-h] --h1 H1 --h2 H2 [--mdl MDL] [--res RES] [--msk MSK] [--nrm]
         [--knl KNL] [--lgf LGF]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--mdl	Input model (cif/pdb)
--res	Resolution (A)
--msk	input mask (mrc/map)
--nrm	if True use normalized maps Default: False
--knl	Kernel size (pixels) Default: 5
--lgf	ligand description file

2.3.19 mmcc

real space correlation

```
emda mmcc [-h] --map MAP --mdl MDL --res RES [--nrm] [--msk MSK] [--knl KNL]
          [--nomask] [--lgf LGF]
```

Named Arguments

--map	input full/deposited map
--mdl	Input model (cif/pdb)
--res	Resolution (A)
--nrm	if use, normalized maps are used
	Default: False
--msk	input mask (mrc/map)
--knl	Kernel size (pixels)
	Default: 5
--nomask	if use, correlation maps are not masked
	Default: False
--lgf	ligand description file

2.3.20 fcc

Fourier space correlation

```
emda fcc [-h] --h1 H1 --h2 H2 [--knl KNL] [--msk MSK]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--knl	Kernel size (pixels)
	Default: 5
--msk	input mask (mrc/map)

2.3.21 mapmodelvalidate

map-model validation using FSC

```
emda mapmodelvalidate [-h] --h1 H1 --h2 H2 --mdf MDF [--md1 MD1] [--msk MSK]
                      [--res RES] [--bfc BFC] [--lgf LGF]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--mdf	input full atomic model
--md1	input halfmap1 atomic model
--msk	input mask (mrc/map)
--res	Resolution (A)
--bfc	Overall B factor for model. default=0.0 Default: 0.0
--lgf	ligand description file

2.3.22 mapmodelfsc

map-model FSC

```
emda mapmodelfsc [-h] --map MAP --mdl MDL [--msk MSK] --res RES [--bfc BFC]
                 [--lgf LGF] [--phaserand]
```

Named Arguments

--map	input map
--mdl	input atomic model
--msk	input mask (mrc/map)
--res	Resolution (A)
--bfc	Overall B factor for model. default=0.0 ignored by REFMAC Default: 0.0
--lgf	ligand description file
--phaserand	use if phase randomized FSC is calculated Default: False

2.3.23 overlay

overlay maps

```
emda overlay [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
             [--tra TRA [TRA ...]] [--rot ROT] [--axr AXR [AXR ...]]
             [--ncy NCY] [--res RES] [--fitres FITRES] [--int INT] [--hfm]
             [--mod]
```

Named Arguments

--map maplist for overlay

--msk masklist for overlay

--tra translation vector. default=[0.0, 0.0, 0.0]

Default: [0.0, 0.0, 0.0]

--rot rotation in deg. default=0.0

Default: 0.0

--axr rotation axis. default=[1,0,0]

Default: [1, 0, 0]

--ncy number of fitting cycles. default=5

Default: 5

--res starting fit resol. (A). default= 6 A

Default: 6

--fitres final fit resol. (A). default= 0.0 A

Default: 0.0

--int interpolation method (linear/cubic). default= linear

Default: “linear”

--hfm if use employ half maps

Default: False

--mod if use calls model overlay

Default: False

2.3.24 average

weighted average of several maps

```
emda average [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
             [--tra TRA [TRA ...]] [--rot ROT] [--axr AXR [AXR ...]]
             [--ncy NCY] [--res RES] [--int INT]
```

Named Arguments

--map maplist to average

--msk masklist for maps

--tra translation vec.

Default: [0.0, 0.0, 0.0]

--rot rotation in deg

Default: 0.0

--axr rotation axis

Default: [1, 0, 0]

--ncy number of fitting cycles

Default: 10

--res starting fit resol. (A)

Default: 6

--int interpolation method ([linear]/cubic)

Default: "linear"

2.3.25 diffmap

difference map using average maps

```
emda diffmap [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]] [--res RES]
             [--mod MOD]
```

Named Arguments

--map maplist to diffmap

--msk masklist for maps

--res diffmap resol. (A) [default=0.0 A]

Default: 0.0

--mod Choose from ampli, [norm], power

Default: "norm"

2.3.26 applymask

apply mask on the map

```
emda applymask [-h] --map MAP --msk MSK [--out OUT]
```

Named Arguments

--map	map to be masked
--msk	mask to be applied
--out	output map name Default: "mapmasked.mrc"

2.3.27 scalemap

scale onemap to another

```
emda scalemap [-h] --m1 M1 --m2 M2 [--out OUT]
```

Named Arguments

--m1	input map
--m2	map to be scaled
--out	output map name Default: "scaledmap.mrc"

2.3.28 bestmap

calculate bestmap

```
emda bestmap [-h] --h1 H1 --h2 H2 [--msk MSK] [--knl KNL] [--mod MOD]
             [--out OUT]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--msk	mask to be applied
--knl	kernel radius (pixels) Default: 5
--mod	fsc type (1-resol bins, 2-local) Default: 1

--out output map name
Default: "bestmap.mrc"

2.3.29 predfsc

predict FSC based on # particles

```
emda predfsc [-h] --h1 H1 --h2 H2 [--msk MSK] [--npa NPA [NPA ...]]
             [--bfc BFC [BFC ...]]
```

Named Arguments

--h1 input halfmap1 map
--h2 input halfmap2 map
--msk mask map
--npa n fold of particles
--bfc list of B factors

2.3.30 refmac

prepare data for refmac refinement

```
emda refmac [-h] --h1 H1 --h2 H2 [--msk MSK] [--bfc BFC [BFC ...]] [--out OUT]
```

Named Arguments

--h1 input halfmap1 map
--h2 input halfmap2 map
--msk mask map
--bfc b-factor list
--out output mtz file name
Default: "output.mtz"

2.3.31 occ

overall correlation in real space

```
emda occ [-h] --m1 M1 --m2 M2 [--msk MSK] [--spc SPC]
```

Named Arguments

--m1	input map1 map
--m2	input map2 map
--msk	mask map
--spc	space (real/fourier) for CC calculation Default: "real"

2.3.32 mirror

mirror the map

```
emda mirror [-h] --map MAP
```

Named Arguments

--map	input map

2.3.33 model2map

calculate model based map

```
emda model2map [-h] --mdl MDL --res RES --dim DIM [DIM ...] --cel CEL
               [CEL ...] [--bfc BFC] [--lgf LGF] [--org ORG [ORG ...]]
```

Named Arguments

--mdl	input atomic model
--res	Resolution (A)
--dim	map dim
--cel	cell parameters
--bfc	overall b-factor Default: 0.0
--lgf	ligand description file
--org	map origin

2.3.34 composite

make composite map

```
emda composite [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
```

Named Arguments

--map	maplist to combine
--msk	masklist for maps

2.3.35 magref

magnification refinement

```
emda magref [-h] --map MAP [MAP ...] --ref REF
```

Named Arguments

--map	maplist to correct for magnification [.mrc/.map]
--ref	reference map [.mrc/.map]

2.3.36 com

center of mass

```
emda com [-h] --map MAP [--msk MSK]
```

Named Arguments

--map	input map (MRC/MAP)
--msk	mask to apply on the map

2.3.37 fetch

fetch EMmap and model

```
emda fetch [-h] --emd EMD [EMD ...] [--all]
```

Named Arguments

--emd	list of EMD entries. e.g. 3651
--all	Use to download all data (mask, map, halfdata, model)
	Default: False

2.3.38 symref

refine symmetry axis of a group

```
emda symref [-h] --map MAP [MAP ...] [--emd EMD [EMD ...]] [--mapout]
```

Named Arguments

--map	list of maps to find symmetry axes
--emd	list of emdbid of maps
--mapout	Use to output maps
	Default: False

PYTHON MODULE INDEX

e

INDEX

S

T

W