User's manual

MOGA dynamic aperture optimization tool

Version 1.0

April, 2016

Tool programmed by M. Ehrlichman

Manual written by M. Aiba

1. Introduction

   A dynamic aperture optimisation tool is developed by M. Ehrlichman for SLS2 upgrade project. It has been successfully applied to the SLS2 lattice under development. This manual includes the following sections:

   Section 2 - how the tool works such that users may have an idea for the parameters in the input files

   Section 3 – description of the input files

   Section 4 - how to use the tool particularly in Merlin cluster at PSI

   Section 5 - how to extract the optimisation result (output files)

2. How it works

   Optimization of dynamic aperture is simply a multi-variable optimisation from mathematical point of view. The knobs to be optimised are mainly the sextupole moments in the lattice. Higher order moments may be included if found in the lattice.

   Multi objective genetic algorithm (MOGA) is employed for the optimization, which is implemented in **PISA** framework (actual version is **aPISA**). For ranking the population in MOGA, **PISA** utilises a separate routine, namely **SPEA2** in this tool.

   The optimisation requires particle tracking to evaluate the dynamic aperture. It is performed by using **bmad** code, and the lattice to be optimised is therefore defined in the bmad format.

   The objectives are the on-momentum and off-momentum dynamic apertures, and the momentum deviation for the off-momentum aperture can be specified by user.

   Furthermore, several constraints can be set by user in order to obtain “reasonable” result: the parameters to be constrained are the horizontal and vertical chromaticity and their higher order terms (chromatic tune footprint), higher order dispersion and sextupole and (if used as knob) higher order multipole strengths. These constraint values are also computed with **bmad**.

   For more details of the tool, see E. Ehrlichman, “A Genetic Algorithm for Chromaticity correction in Diffraction Limited Storage Rings.”, PRAB, to be published, and the references therein.

3. Input files

   a. Main input file (*common.in*)

   The main input file contains the parameter groups, general, da (dynamic aperture), moga for performing the optimisation. Also there are the groups, fp (footprint), touschek, adts (amplitude dependent tune shift), da_raster and evaluator for post processing. The file is prepared in the following format (the descriptions after ! character are comments):

```
&general
  lat_file = 'dc01a.bmad' ! Lattice file in bmad format
  use_hybrid = .true. ! Concatinate the linear elements between non-linear elements
  periodicity = 3 ! Super periodicity of the lattice
  use_line = 'per'  ! Beam line used for DA computation, defined in the bmad lattice file
/
&da
  tracking_method = 1 ! Standard bmad tracking algorithm (see bmad manual)
  n_turn   = 200 ! The number of turns to be tracked to evaluate dynamic aperture
  n_adts   = 200 ! The number of turns to be tracked to evaluate adts. (-1 to disable)
  n_angle = 7 ! The number of directions of DA search in x-y plane
  dE(1) = 0.0 ! Energy deviation for DA evaluation
  dE(2) = -0.03
  dE(3) =  0.03
  track_dims = 4 ! Tracking dimension
  init_len = 0.002 ! Initial step for DA search
  adts_x_min = 37.0 ! Constraint for the amplitude dependent tune shift
  adts_x_max = 37.5
  adts_y_min = 10.0
  adts_y_max = 10.5
/
&moga
  generate_feasible_seeds_only = -1    ! -1 to disable. If 1. Simulator stops when population is
feasible.
  moga_output_file = 'moga_results.out' ! Output filename containing the optimization result
  initial_pop = 'random' ! The last generation of the above output can be in here to continue.
  seed = 4,5 ! Random seed for MOGA optimisation (Fortran routine used requires 2 seeds…)
  max_gen = 900 ! Maximum generations for MOGA optimisation

  set_chrom_x = 0.0 ! Set chromaticity
  set_chrom_y = 0.0

  !breeder parameters !
  breeder_params%cross_p = 0.8 ! Pc in Ehrlichman's paper
  breeder_params%mutate_p = 0.0435 ! Pm (1/Number_of_knobs typically)
  breeder_params%eta = 0.8 ! kappa=1/(1+eta)

  !constraints
  linear_vec_cutoff = 0.0020  ! LA smaller than this is perfectly bad objective value.
  co_limit = 0.0040 ! Constraint for the off energy closed orbit
  fp_dE_neg = -0.05 ! Negative momentum boundary for constraining chromatic tune shift
  fp_dE_pos = 0.05 ! Positive momentum boundary for constraining chromatic tune shift
```

```
  n_fp_steps = 20 ! Number of points to compute chromatic tune shift
  x_fp_min = 37.0 ! Constraint boundary for chromatic tune shift
  x_fp_max = 37.5
  y_fp_min = 10.0
  y_fp_max = 10.5

  !variables
  ! Varialbles (Knobs) in MOGA : ‘c’ for sextupole in dispersive section, ‘h’ for harmonic sext
  !          type   name      l cons    u cons   l init   u init   mutate width
  mags_in(1)  = 'c',   'sd',     -1000.0,  1000.0,  -1000.0, 1000.0,  300.0
          …
          …
  mags_in(23) = 'h',   'oxx_l',  -3000.0,  3000.0,  -3000.0, 3000.0,  600.0
/
&fp ! Parameters to compute chromatic tune sshift for post processing
  pz_min = -0.05
  pz_max =  0.05
  n_pz = 31
/
&touschek ! For post processing to check if Touschek lifetime after the optimisation
  tracking_method = 1 ! bmad standard method
  rf_bucket = 0.05 ! Assumed bucket height for 4D tracking
  n_turn = 100 !  The number of turns to evaluate Toushcek lifetime
  dims = 4 ! 4D or 6D
  current = 0.001 ! Beam current (A)
  bunch_length = 0.00261 ! Bunch length (m)
  horizontal_emittance = -1 ! Horizontal emittance (-1 = computed from lattice)
  vertical_emittance = 10.0e-12 ! Vertical emittance (with coupling assumed)
  stepping = 'by_n_steps' ! by_ix, by_n_steps or by_file
  n_ma_locs = 1000 ! Number of steps
/
&adts ! Parameters to compute amplitude dependent tune shift for post processing
 use_bounds_file = .true. ! Get ADTS in x and y from DA program
 x_min = -0.01
 x_max =  0.01
 y_min =  0.00
 y_max =  0.01
 n_steps = 60
/
&da_raster ! Parameters to visualise the dynamic aperture for post processing, “survival plot”
 x_min = -0.003
 x_max =  0.003
```

```
  y_min = 0.0
  y_max = 0.01
  nx = 51
  ny = 25
  calc_tunes = 'no'
  linear_bounds_file = '../00da_linear/linear_boundary.dat'
/
 &evaluator ! Switches for post processing (read by dynap_pisa_evaluator.py)
   makeLats = .true.
   plotOnly = .false.
   doNDP = .false.
   doLA = .true.
   doDA = .true.
   doRA = .false.
   doTL = .false.
   doFP = .true.
   doADTS = .true.
/
```

b. Lattice file

Lattice file is described in the bmad format. OPA is capable to output it. However, the aperture parameters have to be added to OPA output (if not included) for the MOGA optimisation since the objectives (dynamic aperture) is evaluated with respect to the linear aperture.

```
*[aperture_type]=elliptical
*[aperture]=0.01
```

Also rf setting to get stable 6*6 matrix is required even when the tracking for the dynamic aperture optimisation is in transverse only. This could be avoided by a dedicated coding (in bmad?) but adding an rf cavity to the lattice file is easy fix.

```
rfcav: rfcavity, voltage = -1.41 * 1e6, harmon = 480, l = 0.0    ! 500 MHz RF, 5% bucket
```

Step size for tracking and fringe field type also added to perform reasonable tracking.

```
sbend[ds_step] = 0.001
sbend[fringe_type] = linear_edge
```

See the example file described in Section 4 for more details.

c. **PISA** parameter file

```
initial_population_size 300
```

parent_set_size 150
offspring_set_size 150
objectives 3
constraints 5

initial_population_size is the number of population in the first generation, which is actually constant in the later generations. *parent_set_size* is the number of individuals to be left for the next generation, and *offspring_set_size* is the number of new individuals in the next generation. These should be half of *initial_population_size*. *objective* is the number of objectives (3 is for the dynamic apertures for the nominal momentum, positive and negative momentum offsets). *contraints* is the number of constraints, which should be always 5 unless the code is expanded to include more constraints.

d. **SPEA2** parameter file

There are four parameters to be given for SPEA2 ranking algorithm.

seed 10
tournament 4
k_neighbor SQRT
verbose YES

seed is the random seed for ranking process. *tournament* is the number of random numbers to be drawn. *k_neighbor* specifies how to compute the distance between two individuals. If *verbose* is YES, spea2_diag.log file is output, reporting the number of good individuals at each generation.

4. How to run in Merlin cluster

First of all, it is obvious that you need an account on merlin cluster to use the tool.
Merlin cluster is dedicated for time-consuming computations, and it is mandatory to login to the login cluster, merlinl01.psi.ch and not to the computation node.

To get started, copy the example situated in the slsbd project directory by typing
> cp -r /afs/psi.ch/project/slsbd/public/MOGA/Example/ Example
and
> cd Example
In that directory, all the necessary input files are contained.

Then, to start MOGA optimisation on the lattice in Example directory (*da01a.bmad*), type

> qsub sge.dynap_pisa

The optimisation job is terminated after ~24 hours and if necessary, continue next day with the last generation. The last generation should be extracted from moga_resuts.out to a file. The filename containing the last generation needs to be put into the main input file (*common.in*).

5. Output files and post processing

During or after MOGA optimisation, one can see the progress by taking a look at the file *spea2_diag.log*. An example of this file is:

…
…
(12.03.2016 08:17:39) spea2: #423 1st front size (118)
(12.03.2016 08:19:40) spea2: #424 1st front size (118)
(12.03.2016 08:23:25) spea2: #425 1st front size (105)
(12.03.2016 08:27:00) spea2: #426 1st front size (97)
(12.03.2016 08:30:12) spea2: #427 1st front size (99)
(12.03.2016 08:33:28) spea2: #428 1st front size (101)
(12.03.2016 08:37:00) spea2: #429 1st front size (102)
(12.03.2016 08:39:45) spea2: #430 1st front size (103)
(12.03.2016 08:42:45) spea2: #431 1st front size (104)
(12.03.2016 08:46:27) spea2: #432 1st front size (107)
(12.03.2016 08:50:05) spea2: #433 1st front size (111)
(12.03.2016 08:54:03) spea2: #434 1st front size (113)
(12.03.2016 08:57:53) spea2: #435 1st front size (116)
(12.03.2016 09:01:09) spea2: #436 1st front size (118)

#423 etc. are the number of generations and the 1st front size(118) is the number of individuals that are not dominated by any other individual. It is seen that the front sizes decreases at 426th generation. This indicates that a superior individual is spawn and the Pareto front is pushed towards. When the front size steadily approaches the population size, it is an indication that the optimisation is converging.

The optimisation job can be cancelled whenever necessary (any mistake found or it has converged)

> qdel <job_ID>

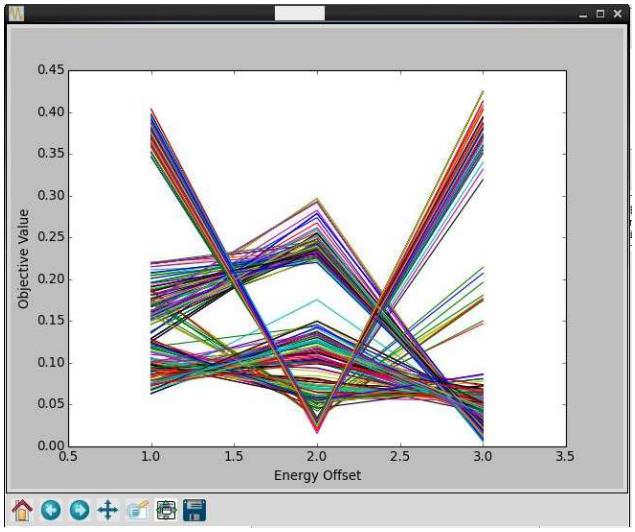
Job_ID can be found by

> qstat

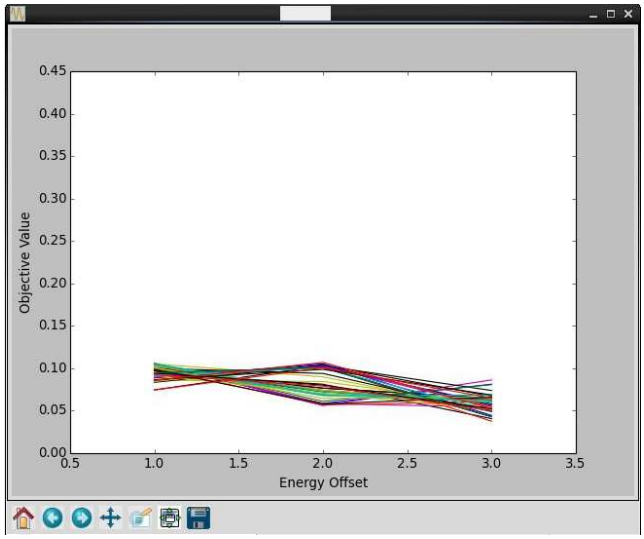
To perform post processing, go to the directory *Example/report/* and type

> /afs/psi.ch/project/slsbd/public/MOGA/ebin/moga_objs.py

Then a matplotlib window pops up



The horizontal axis represents the three energy offsets, that is, 1 = the nominal energy. 2 = negative energy offset and 3 = positive energy offset. The vertical axis corresponds to the dynamic aperture for these energy offsets. N.B. the smaller objective value is the better dynamic aperture. In the above window, one can exclude relatively bad individuals by bringing the mouse pointer close to the line to be excluded and click right button. After the selection, it will be like the next figure.



Then, by hitting 'w' key, *moga_picked.dat* file is created that contains only the individuals left. They are to be examined in the next step.

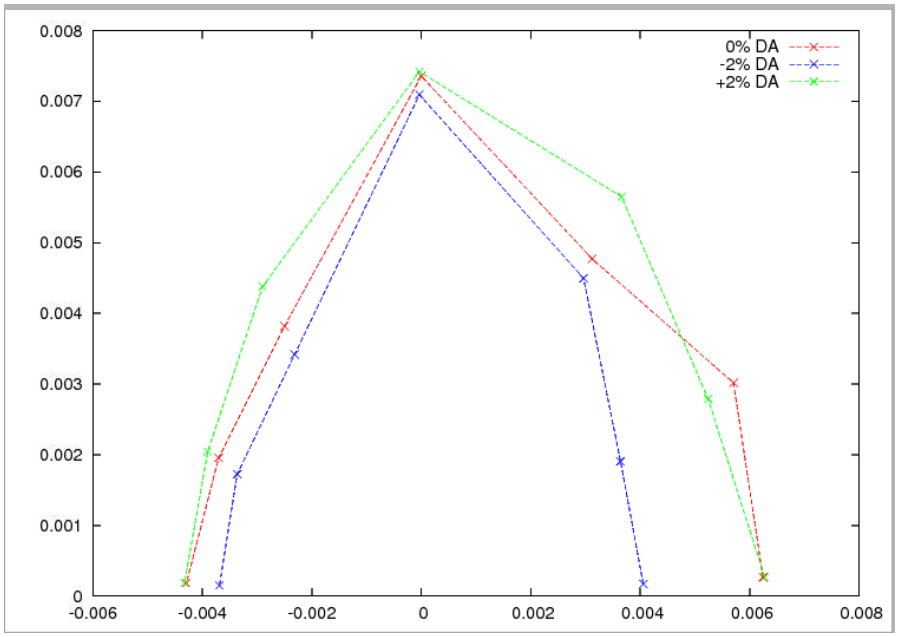
Next step is to evaluate or visualise various properties of the individuals (post processing) such as the dynamic apertures and amplitude dependent tune shifts:

> qsub sge.dynap_pisa_evaluator

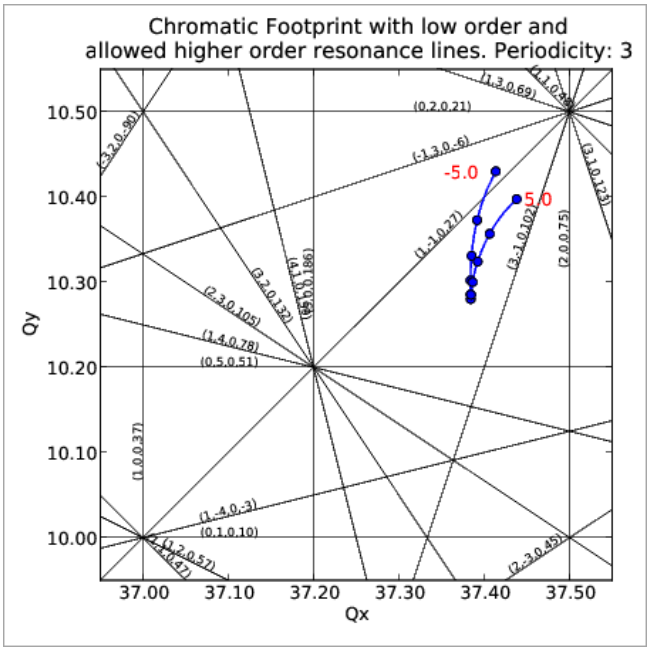
in Example/report directory.

The results of the post processing are stored in automatically generated directories for each individual. It is noted that the post processing may take long time if many individual are examined and/or Touschek lifetime evaluation is included.

A couple of figures created by the post processing are shown below:



Dynamic aperture output (set doDA = .true. in common.in)



Dynamic aperture output (set doFP = .true. in common.in)

The objectives and the variables are also visualized by:

> gnuplot /afs/psi.ch/project/slsbd/public/MOGA/ebin/moga_objs.gp

> gnuplot /afs/psi.ch/project/slsbd/public/MOGA/ebin/moga_var.gp

in */Example/report* directory.

These gnuplot scripts generate report_objs.eps and report_vars.eps.

A full set of analysis files is found in:

/gpfs/home/ehrlichman_m/SLS2/dynap_pisa/dc01a/pisa_1_cont/report/seed_20520