ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

NEW CERN PROTON SYNCHROTRON BEAM OPTIMIZATION TOOL

E. Piselli, A. Akroh
CERN, Geneva, Switzerland

Abstract

This paper describes a new software tool recently developed at CERN called "New CPS Beam Optimizer". This application allows the automatic optimization of beam properties using a statistical method, which has been modified to suit the purpose. Tuning beams is laborious and time-consuming, therefore, to gain operational efficiency, this new method to perform an intelligent automatic scan sequence has been implemented.

The application, written in JavaFX, uses CERN control group standard libraries and is quite simple. The GUI is user-friendly and allows operators to configure different optimisation processes in a dynamic and easy way.

Different measurements, complemented by simulations, have therefore been performed to try and understand the response of the algorithm. These results are presented here, along with the modifications still needed in the original mathematical libraries.

INTRODUCTION

The accelerator complex at CERN is a succession of machines that accelerate particles to increasingly higher energies. Sequentially, each machine increases the energy of a beam of particles, before injecting into the next one. The CERN Proton Synchrotron (CPS) complex is the first group of accelerators serving as injectors for the Large Hadronic Collider (LHC). It comprises of one Linear Accelerator (Linac2) and two different synchrotron rings, the Proton Synchrotron Booster (PSB) and the Proton Synchrotron (PS), as well as several beam transfer lines.

Beam tuning is the process where operators change accelerator beam parameters in order to minimize or maximize beam observables.

Unfortunately, irregular beamline designs together with misalignment of the equipment challenge any simulation to achieve the best results. Simulating in advance the final accelerator tune is often very slow and not always efficient. In addition, some devices like quadrupoles, correction deflectors and kickers are very laborious to simulate with high certainty.

Very often, during beams setting-up or machine development, operators may have the challenge to find optimum settings, especially when the phase space of available parameters is large.

The problem of tuning different parameters versus one or more detectors can be compared to a numerical analysis optimization concept. In fact, in this field of mathematics, optimization is devoted to the study of the theory and methods to search the smallest or largest value of a function:

$$\min_{x \epsilon X} f(x) \quad \text{or} \quad \max_{x \epsilon X} f(x)$$

where:

- $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is the multivariable function
- $X \subseteq \mathbb{R}^n$ is the set of possible solutions

When the problem is looking for minimum or maximum of a function, most of the known algorithms are based on the concept of the derivative and on the gradient information. In general it is not always possible to have an analytical expression of the function (which is abstract) and, as a consequence, the derivatives cannot be calculated. For this use cases one has to look into algorithms that use some function samples in a defined set of points in order to calculate different iterations. Direct-search methods are used for both deterministic and stochastic applications and, since they are effective techniques in deterministic applications especially when derivatives are unavailable, they have been targeted as primary choice for the development of the tool.

These methods are generally robust with respect to small perturbations in the function's values and therefore, are used often in applications where noise is present, which is often the situation faced in operations.

NELDER MEAD ALGORITHM

In the group of direct-search methods, the most popular one is called Nelder-Mead algorithm [1].

The algorithm uses a regular simplex, which is a polytope in n-dimensional space with $n + 1$ vertices, each of which are connected to all other vertices (e.g. a triangle in $\mathbb{R}^2$, a tetrahedron in $\mathbb{R}^3$, etc.).

In order to perform an optimization, the algorithm begins with the function's values on a set of $n + 1$ points in the parameter space of n variables (simplex) and it moves across the surface to be analysed in the direction of steepest ascent (for maximization) or steepest descent (for minimization) by replacing the worst vertex in the simplex with its "mirror image" across the face formed by the remaining vertices. The algorithm, while running, can change in five different ways during an iteration, as illustrated in Fig. 1 in two dimensions.

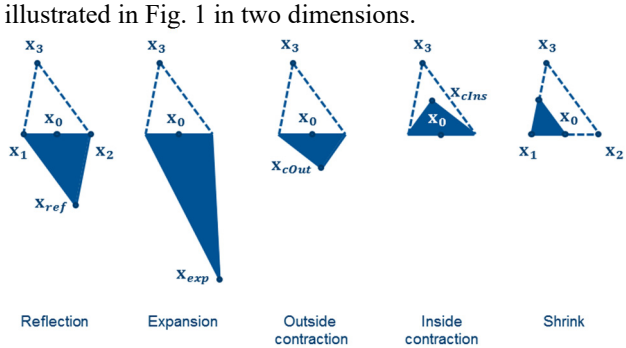

Figure 1: Nelder Mead iterations.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

An outline of the different steps rules for the Nelder-Mead simplex algorithm for function minimization with n parameters is as follows [1][2][3]:

1- **Initialization**. An initial simplex with n+1 points is chosen and the function is evaluated at each point. Rank order is generated in ascending order of the function at each point.
If $f(x_1) < f(x_2) < \cdots < f(x_{n+1})$ the ordered set is $\{x_1, x_2, \ldots, x_{n+1}\}$. We denote with S^0 this initial simplex.

2- **Simplex iteration**. Considering this to be the iteration k, the algorithm starts removing the worst point $f(x_{n+1})$ from S^{k-1} and adding the new point x_{ref} calculated reflecting x_{n+1} as:

$$x_{ref} = (1 + \alpha)x_0 - \alpha x_{n+1}$$

Where α, called reflection coefficient, is normally equal to 1 and x_0 is calculated from the remaining n points of S^{k-1} as:

$$x_0 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Then depending on the value of the function in x_{ref} among the points in S^{k-1}, one of the following operation will be performed:

- Expansion: if $f(x_{ref}) < f(x_1)$, a new point is calculated:

$$x_{exp} = x_0 + \gamma(x_{ref} - x_0)$$

Where γ, called expansion coefficient, is usually equal to 2.
From its estimation $f(x_{exp})$ we can have:
  - if $f(x_{exp}) < f(x_1)$ then x_{exp} replaces x_{n+1} in the simplex
  - else x_{ref} replaces x_{n+1} in the simplex

- Reflection: if $f(x_1) \le f(x_{ref}) \le f(x_n)$ x_{ref} replaces x_{n+1}

- Contraction: if $f(x_{ref}) \ge f(x_n)$ we can have 2 possible contractions:
  - Outside contraction:
  if $f(x_n) \le f(x_{ref}) < f(x_{n+1})$ then calculate:
  $$x_{cOut} = x_0 + \beta(x_{ref} - x_0)$$
  if $f(x_{cOut}) \le f(x_{ref})$ then replace x_{n+1} with x_{cOut} else go to the following point (shrink);

  - Inside contraction:
  if $f(x_{ref}) \ge f(x_{n+1})$ then calculate:
  $$x_{cIns} = x_0 - \beta(x_0 - x_{n+1})$$
  if $f(x_{cIns}) < f(x_{n+1})$ then replace x_{n+1} with x_{cIns}, otherwise go to the following point (shrink);

  Where β is the contraction coefficient usually equal to $\frac{1}{2}$.

- Shrink: all the points in the simplex are shrunk towards x_1 by replacing each points as:

$$x_i = x_1 + \delta(x_i - x_1) \qquad \forall i \in [2, n+1]$$

Where δ is the shrinkage coefficient usually equal to $\frac{1}{2}$.

3- **Next iteration or terminate**. At this stage either a new point has replaced x_{n+1} through expansion, reflection or contraction or a set of new points $\{x_2, x_3, \ldots, x_{n+1}\}$ have been calculated with a shrink. The new set of points S^k is reordered based on their respective function values. If the stopping conditions are satisfied the algorithm terminates, otherwise another iteration is needed.

SOLUTION PROPOSED

Following preliminary investigation and studies done on the Nelder Mead algorithm, the method is believed to properly fit the needs of beam operations at CERN.

For this purpose the point x_i (in $\{x_1, x_2, \ldots, x_{n+1}\}$ is replaced by a set of values of the n beam parameters $\{x_{i,1}, x_{i,2}, \ldots, x_{i,n+1}\}$ to be optimized and the function by the beam observable:

$$x_1 = \{x_{1,1}, x_{1,2}, \ldots, x_{1,n+1}\}$$
$$x_2 = \{x_{2,1}, x_{2,2}, \ldots, x_{2,n+1}\}$$
$$\ldots$$
$$x_n = \{x_{n,1}, x_{n,2}, \ldots, x_{n,n+1}\}$$

In order to fulfil our needs, some modifications with respect to the original method have been adopted.

The first modification was to add constraints with upper and lower bounds for all beam parameters $x_{i,j}$, ($\forall j$ and $1 \le i \le n$). This is an essential condition due to:
- Hardware limits in the different devices imposed by the power supplies' working range.
- Software limits given by different possible instabilities and safety problems that could lead to beam losses.

The second modification adopted was to add specific convergence criteria's in order to avoid the possibility to end up in a local minimum/maximum. For example, a way of restarting the algorithm after a certain number of useless attempts has been added.

The development of the tool is based on the java class "NelderMeadSimplex" in the library Apache Commons Math3. Before starting the GUI development application, simulations have been made, successfully checking the

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

behaviour of a routine implemented for the maximization of a double Gaussian function in $\mathbb{R}^2$, see Fig.2 and Fig.3.

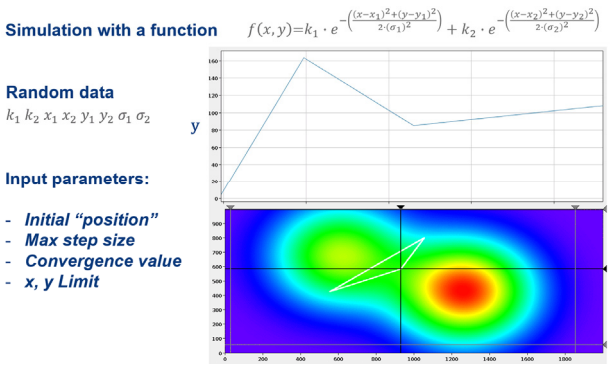

Figure 2: Simulation, initial iteration.

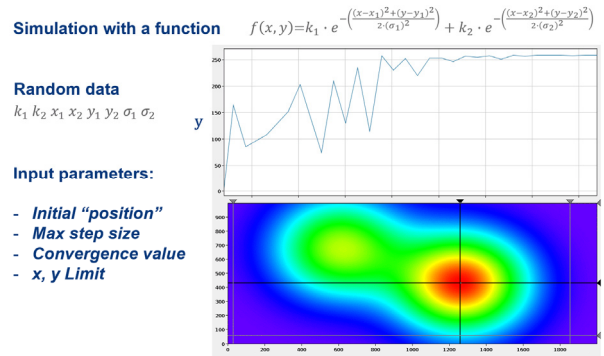

Figure 3: Simulation, last iteration.

SOFTWARE APPLICATION

The application program has been developed using JavaFX toolkit. The flexibility and the easy use of this programming language allowed the creation of a very robust and reliable application, respecting a very simple logic composed with a model, a view (GUI) part and a controller part. An example is sketched in Fig. 4.

This logic allowed us to design a block diagram taking into account all the specifications that we needed (Fig. 5).

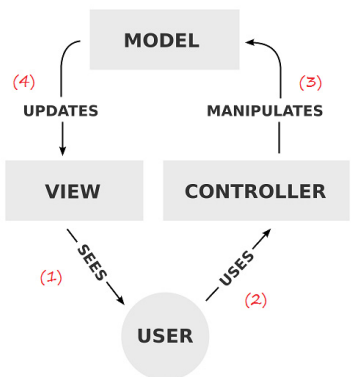

Figure 4: Application logic.

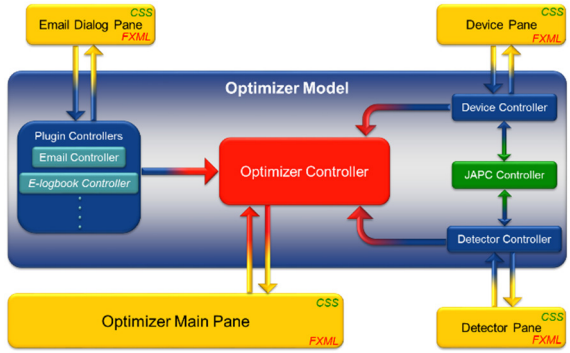

Figure 5: Application GUI mock-up.

Figure 6 shows the GUI which is currently released and in the Control Console Manager (CCM) of the PS Booster and available for operations in the CERN Control Center (CCC).

In the GUI one can identify different areas:

1- All the devices to scan have been grouped according to their physical location in the machine. With these two menus, it is possible to select different machine zones and the devices to optimize.

2- Once the devices have been selected, these panels show the actual values and acquisition data. For each device it is possible to select either hardware or software limits to be used during the scan.

3- In this part of the GUI it is possible to select which beam observable to monitor. Most of the time it is a Beam Current Transformer (BCT). It is even possible to select two observables in order to perform the optimization of ratio of beam observables.

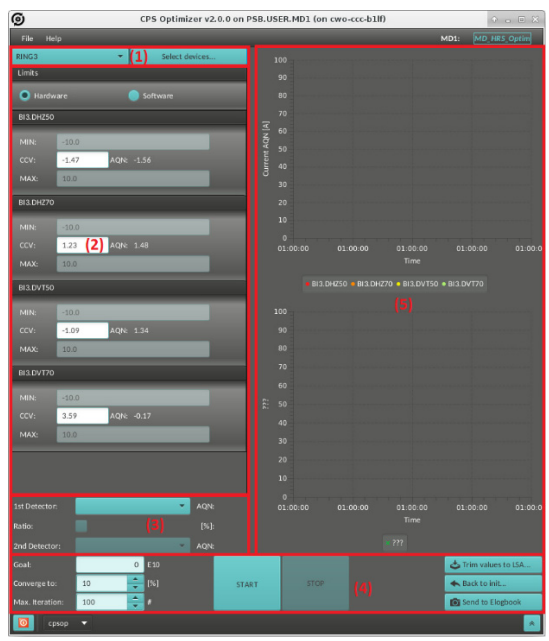

Figure 6: Application GUI screenshot.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

4- This is the central part of the application. Here it is possible to set the constraints for the algorithm:
   - "Goal": the value which one aims to reach in order to stop the optimization.
   - "Converge to": once the "Goal" value is reached by the optimization, the reading of the beam observable(s) should be within a range fixed around this value as percentage of the goal value reached. In this case the process stops and the optimization is finished successfully.
   - "Maximum Iterations": the maximum number of iterations of the algorithm before converging. If this number is exceeded, the optimization process finishes and the application shows a failure message.
   - Buttons: used to control the process. At any time, even when the algorithm is still running, one can stop the optimization either keeping the last values or returning to the initial values. It is also possible to send images and optimization information to the eLogbook and save all the devices' values to the Injector Control Architecture (InCA) database.
5- These two charts display the evolution of the optimization process while scanning devices: the top chart shows acquisition data from the selected devices and the bottom one the acquisition data from the selected observables (or their ratio).

MEASUREMENTS

During this year different campaigns of measurements have been carried out in the PSB.

In the PSB, the horizontal and vertical tune of the machine (Q_h and Q_h) – the number of betatronic oscillation for one turn in horizontal and vertical planes - are defined by the main quadrupoles. At low energy, when the proton density is too high in a small space the particles reject themselves. This effect is called "space-charge" [4]. Space-charge effect can alter the Q_h and Q_h of the machine, inducting tune spread and leading to instabilities or undesired blow-up of important beam parameters, such as the transverse emittance. The smaller the emittance, the higher the brightness of the beam. To compensate for these effects, one has to balance the crossing of resonances with proper multipole setting. The control of the tune is flexible in the PSB and can be adapted for specific beam types using additional and independent magnet trims called Q-Strips. These will bring a small correction called $\Delta Q_h$and $\Delta Q_h$to the tune of the machine.

With the new optimization tool, several tests have been performed using the Q-Strips, in order to maximize the extracted intensity of specific beams.

Figure 7 shows a screenshot of the improvements of the extracted beam from the first ring of the PSB followed after tuning the Q-Strips. It is very interesting to see in Fig. 8 and Fig. 9 the detailed evolution of the tuned parameters and of the beam observable during the iterations.

Figure 7: Q-Strips tuning

Using the tool, the beam performance could be increased from 370E10ppp to 430E10ppp, which corresponds to +16% in only 40 steps.

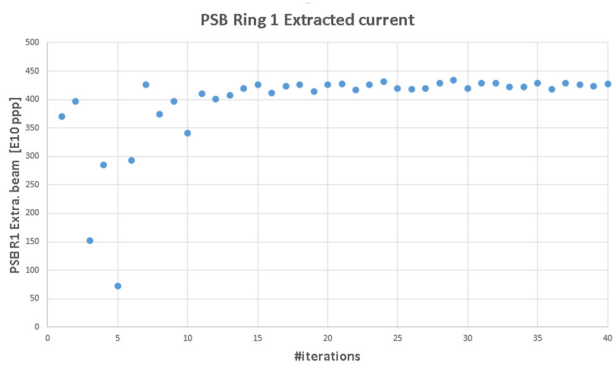

Figure 8: Beam observable evolution vs iterations.

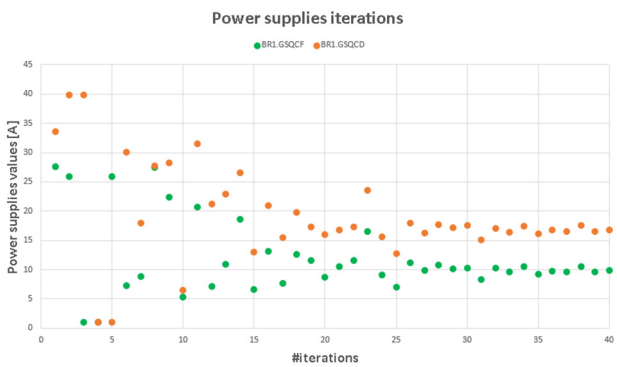

Figure 9: Beam parameters tune evolution vs iterations.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA120

Figure 10: Optimization of 4 devices starting with no beam.

Another type of measurements to test more intensively the robustness and efficiency of the algorithm was performed too. In fact, in order to maximize the transmission of the injected intensity in PSB from the Linac2, it is always required to tune all the injection horizontal and vertical deflectors. These deflectors are used to optimize the angle and the position of the beam.
Fig. 10 shows a screenshot of the application at the end of the optimization. To be noted that 4 devices were used at the same time and that the process started from some wrong values of these devices.
Even in this case, the algorithm has converged to an optimum value after 46 iterations and changing drastically the 4 devices values.

CONCLUSIONS AND PERSPECTIVES

A software tool has been developed targeting the operation of the CPS complex at CERN. This application allows the automatic optimization of beam properties using a statistical method. Preliminary measurements with beam in the PSB showed a fast response of the algorithm and all the tests done, agreed with expectations. Although initial measurements concur with expectations, there remains a need to implement a minimization optimization functionality and furthermore to improve the "fine tuning" of the convergence parameters in order to increase the speed of the optimization during machine operation.
From experience with this method, it is understood that the beam tuning task could often be automated and that many parameters could be auto tuned with results similar to an experienced human operator.

The plan for the future is to have a general tool which can be used across accelerators for the current and future CERN operations.

ACKNOWLEDGMENT

We gratefully acknowledge the support of the PSB-PS operators and section leaders and G. Kruk for his advice during our Java code development.

Special thanks to G.P. Di Giovanni for his suggestions and essential help during the measurements.

REFERENCES

[1] Margaret H. Wright, "Nelder, Mead, and the Other Simplex Method", Documenta Mathematica - Extra Volume ISMP (2012) 271–276
[2] J. J. Tomick, "On Convergence of the Nelder-Mead algorithm for unconstrained stochastic optimization", PhD Thesis (1995)
[3] G. B. Dantzig, "Linear Programming and Extension", Princeton University Press (1963)
[4] K. Schindl. "Space Charge", CAS (CERN Accelerator School), Basic Course on General Accelerator Physics (2000)