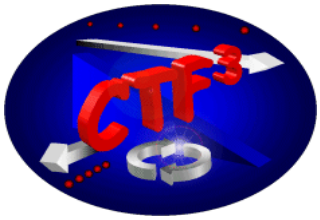
CERN – European Organization for Nuclear Research

European Laboratory for Particle Physics



CTF3 Note 068 (Tech.)
CARE/ELAN Document-2005-002
(Beam Steering, Linac)

THE AUTOMATIC STEERING SYSTEM IN CTF3

R. Lifshitz[1], D. Schulte[2]

Abstract

An automatic system for steering-response measurement has been implemented at the CLIC Test Facility (CTF3) at CERN. This system integrates between the controls environment in the PS complex and the context of a scripting environment, which also serves as a platform for the machine simulation. Automatic trajectory correction is made possible thanks to the measured response parameters. The steering system is used as a machine commissioning tool, as well as a measurement tool for benchmarking the simulation code. This document introduces the system, describes it's main components, and gives a practical guide for potential users.

[1] Technion - Israel Institute of Technology, Haifa, Israel
[2] CERN, Geneva, Switzerland

Geneva, Switzerland
24/02/2005

The Automatic Steering System in CTF3

R.Lifshitz[1], D. Schulte[2]

1) Technion - Israel Institute of Technology, Haifa, Israel
2) CERN, Geneva, Switzerland

CTF3-Note-068(TECH)

Abstract

An automatic system for steering-response measurement has been implemented at the CLIC Test Facility (CTF3) at CERN. This system integrates between the controls environment in the PS complex and the context of a scripting environment, which also serves as a platform for the machine simulation. Automatic trajectory correction is made possible thanks to the measured response parameters. The steering system is used as a machine commissioning tool, as well as a measurement tool for benchmarking the simulation code. This document introduces the system, describes it's main components, and gives a practical guide for potential users.

Introduction

An automatic trajectory correction system has been implemented in CTF3 during the 2004 machine runs. The goals of this system are to supply a useful tool for machine commissioning and to serve as a benchmark for the linac simulation tool PLACET [1]. A steering-response matrix is measured from each deflecting magnet to all the position monitors. It reflects both free-space and focusing regions in the linac. This response matrix can be used for automatic trajectory correction, which can steer the beam through the linac in an optimal way.

Using a good-enough description of the physical system, it might be possible to produce these coefficients using the simulation, without measurement. For this reason, the benchmarking of the simulation code PLACET [1] is considered as one of the primary goals of this system. This will allow faster operation of the steering system as well as it will approve the simulation tool for the future of CTF3 and CLIC.

This document introduces the steering system in CTF3. Sections 1 and 2 give a conceptual overview of the software and the way it operates. The different steering scripts are described in section 3, and a practical user's guide is presented in section 4.

1. The response matrix and the correction term

The beam trajectory can be modified by using dedicated deflector magnets (dipoles), which are installed along the linac. In order to incorporate trajectory correction functionality, the software must know the reaction of the beam to changes of current in these magnets. This data corresponds to response coefficients and summarized in a response matrix, assuming linearity. Expression 2.1 shows this relation, where $\delta\vec{k}$ is the vector of changes in deflector currents, $\delta\vec{b}$ is the corresponding changes vector of the BPM readings, and $\hat{R}$ is the response matrix.

$$\delta\vec{b} = \hat{R}\delta\vec{k} \qquad (2.1)$$

The response matrix is obtained by measuring the beam position monitor (BPM) signal responses to changes in the applied currents to each deflecting magnet. Changing the current of a given dipole and sampling all position monitors gives the response coefficients for all BPMs, with respect to this dipole. In fact, the current of each dipole is scanned in a number of steps, giving the opportunity to estimate both the response and its linearity.

Having a measured response matrix, various algorithms may be used to alter the setting of the deflecting magnets, so that the trajectory fits better to the requirement. The currently implemented tool is based on the minimization of a term similar to the one shown in 2.2, where x is the minimized function.

$$x = \sum_i \left(\frac{b^{init}{}_i - b^{final}{}_i}{\sigma_i} \right)^2$$
$$= \sum_i \frac{(b^{init}{}_i + \delta b_i - b^{final}{}_i)^2}{(\sigma_i)^2} = \sum_i \frac{(b^{init}{}_i - b^{final}{}_i + \sum_j \hat{R}_{ij} \delta k_j)^2}{(\sigma_i)^2} \qquad (2.2)$$

For each BPM (i), b^{init} is the initial reading and b^{final} is the value requested by the operator. In practice, b^{final} has always been set to zero. The correction procedure minimizes the function x with respect to the deflector magnet changes vector $\delta\vec{k}$. The resulting $\delta\vec{k}$ is thus the system's suggestion for a change in the deflecting magnet settings, which should yield the required trajectory.

By applying only a given fraction of the suggested change, the system's operation becomes iterative. The signals from all devices are re-read before each iteration, so that effects such as device and beam slow jitters can be averaged out.

2. Software layout

The steering system provides a tool which is useful both for trajectory optimization and for the characterization of the machine, as a part of the simulation code benchmarking. Given that the simulated model and measurement fit together nicely, it is foreseen to use the model as a way to estimate the response matrix, instead of re-measuring it after any change in the optics. This is a crucial feature, since the response measurement is, naturally, very time consuming.

It was thus decided to design the whole system from within the context of the simulation tool PLACET [1]. This allows a direct integration of the machine simulation into the steering system. In fact, this has already been done, although not yet used.

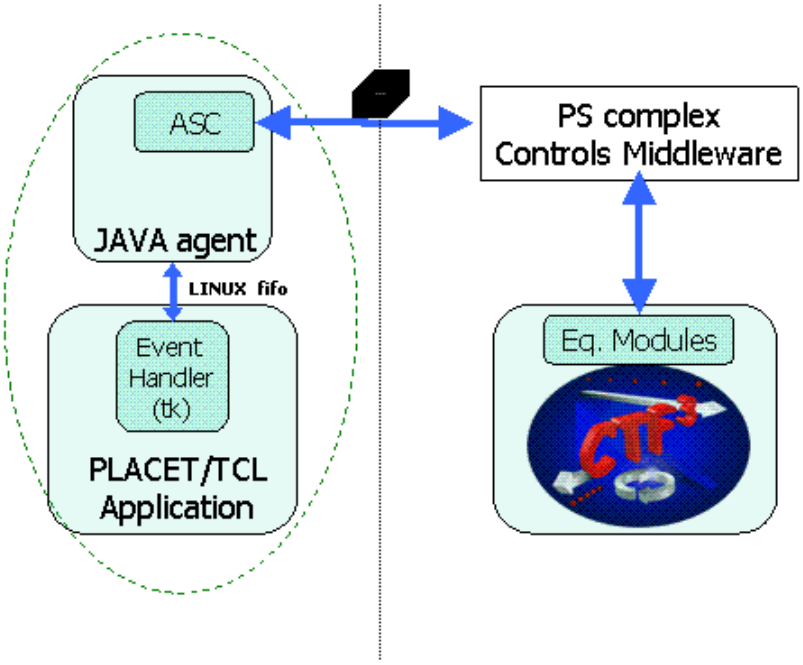


Figure 1: The CTF3 steering software layout

PLACET is based on the standard TCL interpreter [1]. The steering software is a collection of scripts written in PLACET, just as the machine simulation is. The basic hardware access functionality was implemented in JAVA, using the standard ASC [2] toolbox for accessing machine control equipment in the PS complex. A corresponding hardware-access toolbox was written in TCL. This toolbox launches the above-mentioned JAVA agent, enabling PLACET/TCL applications to access the hardware. This set of tools includes some safety features, to prevent unauthorized access to hardware.

3. The steering software scripts

The steering application is arranged in the form of a master module and sub scripts, all written in PLACET/TCL. The master script (namely "master.tcl") is responsible for loading all the necessary functionalities and launching the user interfaces. As was mentioned above, this software is ready for integration with a simulated model of the machine. However, at the current stage of development, this feature is less useful for operation.

The steering scripts are a group of standalone programs, which the user can launch from within the steering software. Since they are re-loaded on each execution, they can be modified between runs, with no need to restart the system for the changes to take effect. They all are located under a "*scripts*" subdirectory in the steering software path, and they all match the naming pattern "*do_*.tcl*"

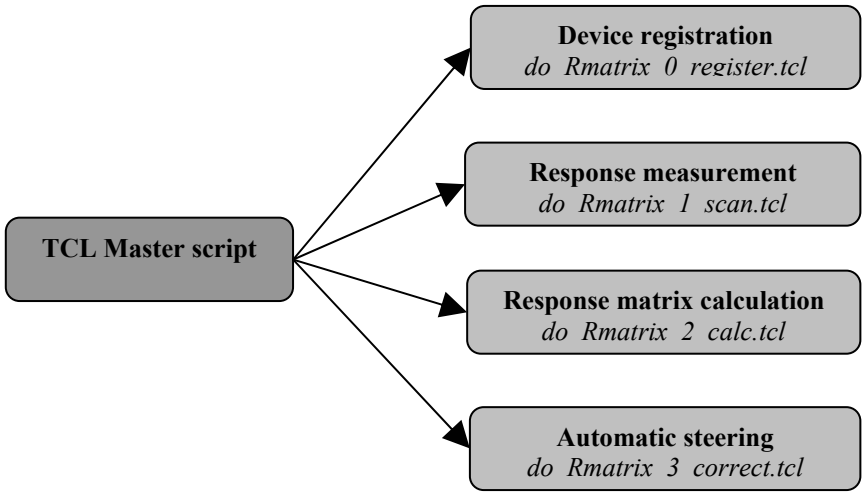


Figure 2: The steering scripts

Most of the scripts have *configuration blocks* within them, which encapsulate all properties the user might want to access. These blocks are written in a self-explanatory way, for users to be able to edit them manually and modify the parameters according to the needs. An interactive access interface to these parameters is a useful addition to the software, which has not been implemented yet.

3.1 The device registration script

(file name: *do_Rmatrix_0_register.tcl*)

As its name suggests, this script is responsible for the registration of all devices in the system. In the PS controls environment, all machine variables are described as ASC parameters [2], which require registration. After this is done, each parameter can be queried for data or even synchronously refreshed in a time or event driven way. Note that the actual registration script is "*do_machine_devRegister.tcl*", which is called from within the above mentioned file.

The registration script adds the names of all relevant devices to a number of lists, according to the device types:

- *Beam Position Monitors*
  All beam position monitors, implemented using the treated signals "xx.SVBPMyyyy-C".
- *Dipoles*
- *Quadrupoles*
  The quadrupole magnets in CTF3 use the same DC power supply modules as the dipoles. For this reason, quadrupoles and dipoles are differenly named instances of the same object class.
- *BPM-like devices*
  This type includes RF power measurement equipment in the CK line.
  These devices share the same type of sampling ADC modules as BPMs, thus their implementation in the software is almost the same, hence the name.
- *Other devices*
  This list is left empty for future use.

After filling these lists, the script loops over the devices and uses a TCL hardware-access call to register the different devices in the system – both in the TCL and ASC (JAVA) contexts. While the ASC registration is required for communication with the control system, the TCL registration initializes a number of local user-transparent variables, which allow the later use of features such as waiting for devices to stabilize, setting device groups, etc.

3.2 Response measurement

(file name: *do_Rmatrix_1_scan.tcl*)

This script performs the actual acquisition of the data which is necessary for calculating the response matrix. In order to be able to reproduce all measurement conditions, the concept of this script is to log all possible data items (both relevant and irrelevant). A separate script uses this data to calculate the response matrix.

This current of each dipole is modified in a number of small steps. All steps are relative to it's nominal setting, to which the dipole is set back after the scan is finished. The different values of current for a dipole scan are created on-the-fly using a set of parameters, which are defined for each dipole within the *configuration block* of the script. Default parameters exist in case of missing definitions. The nature of the dipole scan may be simple "low-high" or a cyclic "low-

high-low" scan, for hysteresis estimation. In any case, only a subset of the sampled points is used for the response coefficient calculation. In the log file, the boundaries of this range are marked as "*mark1*" and "*mark2*".

Note that, in principle, the user might not want to use all dipoles or BPMs for the response measurement. For this purpose, *participating devices* are defined within the configuration block of this script. While these devices should be a subgroup of the lists defined in the registration script (see 3.1), the user is likely to make a mistake at this point. For this reason, the scanning script automatically registers the devices which were not properly listed in the registration script.

The execution of the response measurement script is the most time-consuming part of the steering procedure. A faster and more efficient solution would be to use a simulated model of the machine to produce this data. The benchmarking of a reliable and consistent simulation for this purpose is currently under study.

The log file, which is created by this script, is digested by the calculation script (see 3.3 below) to produce the actual response coefficients and matrix.

3.3 Response matrix calculation

(file name: *do_Rmatrix_2_calc.tcl*)

The above mentioned scanning script produces a log file, which is used by the this script for calculating actual values for the response coefficients. The calculation script extracts the relevant data for each dipole and BPM and uses only what comes between the "*mark1*" and "*mark2*" points, as explained above. The linear fit is done by GNUPLOT [3], which is called from within this script.

The calculation script produces an output file named "*revive_Rmeas.tcl.out*", namely the *revive-file*. After it's creation, the revive-file is a standalone TCL script which reproduces all necessary variables for using the calculated response matrix for trajectory correction, with no need to re-measure or calculate.

Note that the response coefficients are calculated from each participating deflector to all position monitors. This implies that the produced matrix will include meaningless entries, such as responses of BPMs to downstream dipoles. While these entries are theoretically zero, beam jitter may introduce nonzero entries which might spoil the correction capability of the matrix. This effect may be avoided by requiring causality in the matrix, which is one of the configuration switches found in the *configuration block* of the calculation/correction scripts.

3.4 Trajectory correction

(file name: *do_Rmatrix_3_correct.tcl*)

The correction script uses the revive-file (see 3.3 above) to define a response matrix to work with. This script iteratively queries the machine for the dipole settings and BPM readings, and then it implements a correction algorithm (see 1 above) to suggest changes for the deflecting

dipoles. During the process, the data is logged and, by default, user approval is required before each change is implemented on the hardware.

Figure 3 shows the horizontal BPM signals convergence, while using the trajectory correction script, in the CTF3 PETS line.

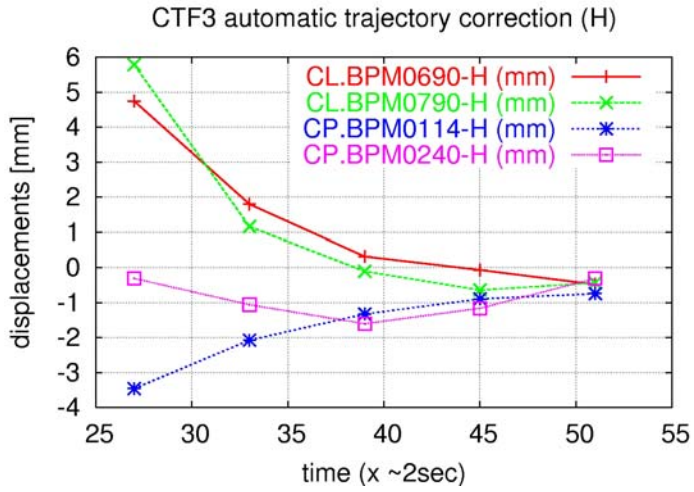


Figure 3: Trajectory correction in the CTF3 PETS line (horizontal)

Since the revive-file is used, there is no need to re-scan the machine to change between different response-matrices. Given that the system is online and the registration script was properly executed, assuming different parameters are required, the user needs only to replace the revive-file and to modify the configuration block inside the correction script.

Note that, since manual user-approval is required by default, the user might be prompted to react to this script. These prompts will appear in the terminal from which the steering software was launched.

4. Practical instructions

This section is intended for real-time users who want to get results quickly out of this software. This section stands for itself (no need to read the above) and is by no means a full description of the steering software.

4.1 Summary of facts

The application-core is called "*master.tcl*". It is a script written in PLACET/TCL, located under "~/beysic/ctf3_2004/master", when "~" is the ctf3op account home directory.

This script invokes two JAVA-based sub-processes:

(1) a hardware-access console ("R_Placet2Ctf") and
(2) a GUI ("CTF Beam Dyamics GUI").

All the hardware-access features are based on standard PS JAVA-ASC [2] tools, which

are implemented in the JAVA hardware-access console.

After starting the application (see below), the interesting stuff is done by running scripts from within this application. These are (independent) TCL scripts, located in "~/beysic/ctf3_2004/scripts". Their names answer to a common naming template: "do_*.tcl"

Although the scripts are executed from within the GUI (see below), there is no friendly interface for their internal configuration - such as setting the lists of devices which participate in a response-matrix measurement, etc. For this reason, one must manually edit any of these scripts and browse through the "*configuration block*" in the beginning of each of them.

Since these scripts are re-loaded each time one executes them (from the GUI),
The user can change the settings of a script while the whole application is running, and re-run a script.
However – this is NOT TRUE for the "device registration" script (see below).

4.2 How to

4.2.1 Start the steering system

- **start the application:**

  (from ctf3op home-directory:)
  >cd ~/beysic/ctf3_2004/master
  >go
  (two panels will show: "CTF Beam Dynamics" and "R_Placet2Ctf")

- **authorize hardware-access for this software**

  In order to permit this application to actually modify hardware properties (essentially - setting dipoles):

  (in the "R_Placet2Ctf" panel)
  go to the "user" tab
  enter the obvious password (at the bottom-right box)
  press ENTER
  check the box labeled "Allow HW setting".

- **make the application "register" the appropriate devices in the control system**

  (in the "CTF-Beam-Dynamics" panel:)
  go to the "Scripts" tab
  run "do_Rmatrix_0_register.tcl"
  (the upper textbox shows all available scripts, to execute each - **copy it's name to**

the lower box, and press the "GO" button.)

- **wait for it to complete**

  messages will appear in the "messages" tab of the "R_Placet2Ctf" panel. wait until it stops updating.
  The last message should look something like:
  "java got : DO;EVENTS_STOP;0;0"

- **setup the trigger (e.g. make beam-events seen by this application)**

  (in the "R_Placet2Ctf" panel:)
  go to the "user" tab
  enter a reasonable number (N= 500? 1000?) in the text-field next to the "Auto every (ms)" checkbox, and then - **check this box!**

4.2.2 Run the scanning script (collect data for a response measurement)

- **run the scanning script:**

  (in the "CTF-Beam-Dynamics" panel:)
  go to the "Scripts" tab
  run "do_Rmatrix_1_scan.tcl" (see 3 above)

  This can take lots of time, and indications about what's going on can be viewed in the "messages" area of the "R_Placet2Ctf" panel. However, watching the standard output of the host terminal might be more useful.

  After this is finished, the log-file "~/beysic/ctf3_2004/scripts/output/Rmatrix.log" containing all the data. This log file will be used to calculate a response matrix.

4.2.3 Make a response matrix out of the data

- **Run the calculation script**

  (in the "CTF-Beam-Dynamics" panel:)
  go to the "Scripts" tab
  run "do_Rmatrix_2_calc.tcl" (see 3 above)

  This will show lots of information in the terminal window, and create the file:
  "~/beysic/ctf3_2004/scripts/output/revive_Rmeas.tcl.out"

  The created file is an independent TCL script, which recreates the measured matrix as a regular TCL variable, to be used by the correction script.

IMPORTANT
If one intends to re-use this matrix (e.g. this setup will be used again) - consider putting the "revive_..." file in a well known place, so that it can be used again!

4.2.4 Run the steering correction algorithm

- **Run the correction script:**

  (in the "CTF-Beam-Dynamics" panel:)
  go to the "Scripts" tab
  run "do_Rmatrix_3_correct.tcl"

  this script loads the (above mentioned) matrix, and uses it to minimize the beam steering deviations-from-zero in the participating BPMs, using the participating steering dipoles.

  By default, the script **will halt** before implementing any correction on the hardware, and the user will be prompted for authorization in the host terminal.

  Pressing ENTER will allow the script to manifest the changes and to move to the next iteration, pressing "q" will quit.

Acknowledgements

We acknowledge the support of the European Community-Research Infrastructure Activity under the FP6 “Structuring the European Research Area” programme (CARE, contract number RII3-CT-2003-506395)

References

1. **PLACET** - A program to simulate drive beams / D. Schulte
   Proceedings of EPAC 2000, Vienna, Austria
   available at: http://dschulte.home.cern.ch/dschulte/placet.html

2. **ASC** – Applications Services and Components,
   a JAVA package for controlling accelerators in the PS complex,
   provided by CERN AB/PS-CO-AP
   available at: http://ab-dep-co-ap.web.cern.ch/ab-dep-co-ap/dev/index.html

3. **GNUPLOT** - An interactive plotting program / T.Williams & C.Kelly
   available at: http://www.gnuplot.info/docs/gnuplot.html