ADVANCES IN MATCHING WITH MAD-X

Riccardo de Maria, Frank Schmidt and Piotr Krzysztof Skowronski, CERN, Geneva, Switzerland

Abstract

A new algorithm and mode has been developed for the matching in MAD-X [1]. The new algorithm, called JACOBIAN, is able to solve a generalized matching problem with an arbitrary number of variables and constraints. It aims at solving the corresponding least square problem. The new mode, called USE_MACRO, allows the user to construct his own macros and expressions for the definition of the constraints in a matching problem. The new algorithm combined with the macro constructs was successfully used for finding optic transitions and a non-linear chromaticity correction. This new approach can be seen as a major upgrade of the matching capabilities of MAD-X taking advantage of various modules like twiss, ptc, track, survey, aperture, etc.

INTRODUCTION

MAD-X provides several different matching algorithms. The most used are `LMDIF` and `SIMPLEX` (the names corresponds to statements used in the MAD-X language). The `LMDIF` implementation in MAD-X, is generally fast but limited to problems where the number of variables is not greater than the number of constraints. Also the constraints have to be differentiable in order to avoid numerical instabilities. `SIMPLEX` is suited for a more general class of problems where the constraints can be non-differentiable. It's slower then LMDIF and the control over the variable bounds is weak.

The new matching routine `JACOBIAN` has been developed in order to solve problems for an arbitrary number of variables and constraints. The variables can be with or without bounds. In particular, this method is very well suited for constrained problems where the variables are already close to the solution. Several options have been added in order to get a finer control on the variable bounds (see `SLOPE` option in `VARY` command [1]);

Until now MAD-X was able to match quantities provided by `TWISS` and stored in the `TWISS` or `SUMM` table. The new mode `USE_MACRO` has been developed to allow the user to define his own observables and allowing to use most of the MAD-X modules.

JACOBIAN

The algorithm is based on the Newton-Raphson method. A matching problem can be defined as

$$\boldsymbol{c} = \boldsymbol{f}(\boldsymbol{v})$$

where $\boldsymbol{c}$ is the vector of c constraints, $\boldsymbol{v}$ is the vector of v variables and $\boldsymbol{f}$ is the vector field representing the accelerator observables.

If $\boldsymbol{c}_0 = \boldsymbol{f}(\boldsymbol{v}_0)$ is a solution for $\boldsymbol{c}_0$ close to $\boldsymbol{c}$ then

$$\boldsymbol{c} = \boldsymbol{c}_0 + \frac{\mathrm{D}\boldsymbol{f}}{\mathrm{D}\boldsymbol{v}}(\boldsymbol{v}_0)\delta\boldsymbol{v} + O\left(|\delta\boldsymbol{v}|^2\right)$$

and the solution is found iteratively using

$$\boldsymbol{v} = \boldsymbol{v}_0 + \alpha_n\delta\boldsymbol{v} \qquad \frac{\mathrm{D}\boldsymbol{f}}{\mathrm{D}\boldsymbol{v}} = \boldsymbol{J} \qquad \delta\boldsymbol{v} = \boldsymbol{J}^{-1}(\boldsymbol{c} - \boldsymbol{c}_0).$$

where $\boldsymbol{J}$ is the Jacobian of the transformation, $\delta\boldsymbol{v}$ is the vector which points to the solution, α_n is the succession 2^{-n} and n is chosen such that the penalty function $|\boldsymbol{c} - \boldsymbol{c}_0|$ is smaller than the previous step (see the `BISEC` option in [1]).

The Jacobian of the transformation must be well conditioned and must have the maximum rank. This condition can be achieved by using the `WEIGHT` option and a proper choice of variables.

If $\boldsymbol{J}$ is square matrix ($c = v$) the system can be inverted exactly. In the case the matrix is rectangular ($v > c$ or $v < c$), the system is inverted by a QR or LQ decomposition yielding the minimization of $|\boldsymbol{c} - \boldsymbol{c}_0|$ or $|\delta\boldsymbol{v}|$ respectively (see [2]).

If a constraint is given in the form of an inequality, one has to distinguish the following cases. If the inequality is already fulfilled by the variables, the relative equation of the linear system is removed. Otherwise, if the inequality is not already fulfilled, it is treated as an equality.

If a variable exceeds its boundaries during each iteration and if $v > c$, the variable is excluded from the set and the linear system is solved again. Otherwise if $v = c$ or there were too many exclusions the variable is assigned to the limit of the boundary. This behavior can be controlled by the `STRATEGY` option (see [1]).

Before the matching process, the following transformations can be applied to the variable vector: a uniform random vector is added to the variables (see `RANDOM` in [1]) in order to avoid local minima; the values of the variables are moved towards a desired value in order to force the final solution to be close to those values (see `COOL`, `BALANCE`, `OPT` [1]).

Figure 1 shows the scheme of the algorithm.

USE_MACRO

The idea behind the new matching mode is to allow the constraints to be user defined expressions evaluated by macros.

MAD-X has the following features: the macros mechanism allows to group the action of the MAD-X modules,

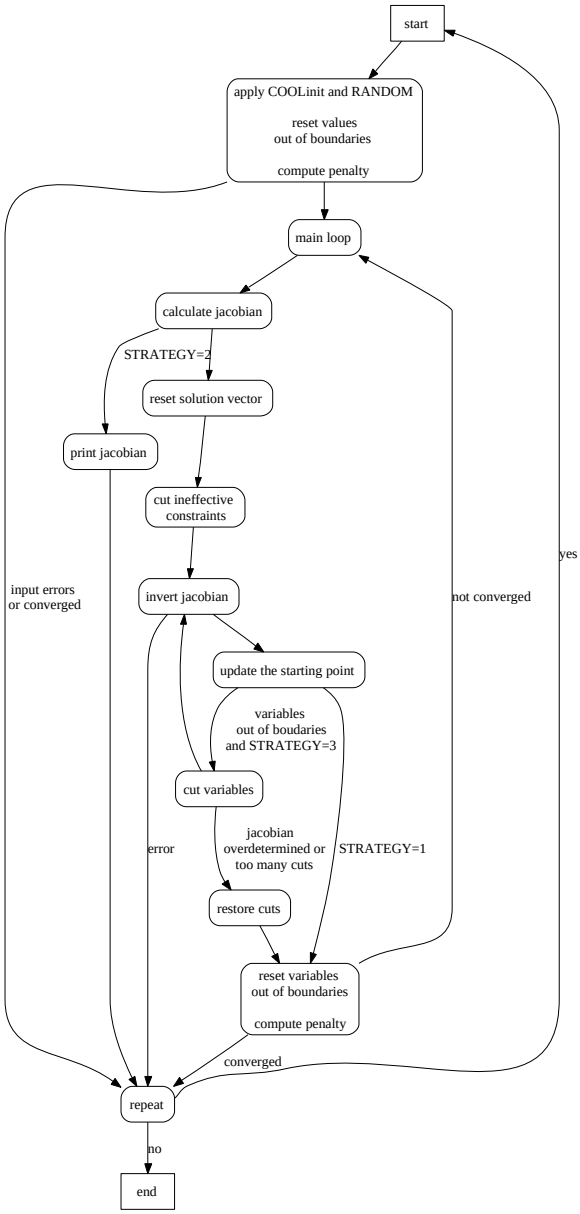

Figure 1: Algorithm of the JACOBIAN routine

the modules generally make accessible their results in tables and variables, the expressions allow to define functions of these quantities (see TABLE in [1]).

The USE_MACRO mode groups these functionalities for the definition of a matching problem.

The USE_MACRO mode is implemented using a syntax which is slightly different from the old style matching and there are two variants:

- using an already defined macro

```
m1: macro= {y=(x-3.5)*(x+2)*(x-4); };
match,use_macro;
  vary,name=x;
 use_macro: name= m1;
  constraint,expr=  y=0;
  jacobian,tolerance=1.e-24;
endmatch;
```

- or an implicit definition of the macro

```
match,use_macro;
  vary,name=x;
 m1: macro= {y=(x-3.5)*(x+2)*(x-4); };
  constraint,expr=  y=0;
  jacobian,tolerance=1.e-24;
endmatch;
```

For a full explanation refer to the MAD-X manual ([1]).

It's worth noting that more than one set of macro and constraints can be specified in sequence:

```
match,use_macro;
  vary,name=x;
 m1: macro= {y=(y-3.5)*(x+2)*(x-4); };
  constraint,expr=  y=0;
 m2: macro= {y=(y-3.5)*(x-1.4)*(x+3.5); };
  constraint,expr=  y=0;
  jacobian,tolerance=1.e-24;
endmatch;
```

This allows a great flexibility in defining complex constraints.

REAL LIFE EXAMPLE

The next example shows the correction of the first and second order chromaticity in the LHC using all the available sextupoles families independently.

A macro called madchrom can be defined to compute the first order chromaticity using the TWISS command and the second order using two times the TWISS command off-momentum and a finite difference.

```
madchrom: macro={
  twiss;
  qx0=table(summ,q1);
  qx1=table(summ,dq1);
  qy0=table(summ,q2);
  qy1=table(summ,dq2);
  dpp=.00001;
  twiss,deltap=dpp;
  qxpp=table(summ,q1);
  qypp=table(summ,q2);
  twiss,deltap=-dpp;
  qxmp=table(summ,q1);
  qymp=table(summ,q2);
  qx2=(qxpp-2*qx0+qxmp)/dpp^2;
  qy2=(qypp-2*qy0+qymp)/dpp^2;
};
```

This macro can be used for defining the matching problem by adding the variables and their dependent constraints.

```
use,sequence=lhcb1;
match,use_macro;
vary, name=ksd1.a12b1; vary, name=ksd1.a23b1;
vary, name=ksd1.a34b1; vary, name=ksd1.a45b1;
vary, name=ksd1.a56b1; vary, name=ksd1.a67b1;
vary, name=ksd1.a78b1; vary, name=ksd1.a81b1;
vary, name=ksd2.a12b1; vary, name=ksd2.a23b1;
vary, name=ksd2.a34b1; vary, name=ksd2.a45b1;
vary, name=ksd2.a56b1; vary, name=ksd2.a67b1;
vary, name=ksd2.a78b1; vary, name=ksd2.a81b1;
vary, name=ksf1.a12b1; vary, name=ksf1.a23b1;
vary, name=ksf1.a34b1; vary, name=ksf1.a45b1;
vary, name=ksf1.a56b1; vary, name=ksf1.a67b1;
vary, name=ksf1.a78b1; vary, name=ksf1.a81b1;
vary, name=ksf2.a12b1; vary, name=ksf2.a23b1;
vary, name=ksf2.a34b1; vary, name=ksf2.a45b1;
vary, name=ksf2.a56b1; vary, name=ksf2.a67b1;
vary, name=ksf2.a78b1; vary, name=ksf2.a81b1;
use_macro,name=madchrom;
constraint,expr= qx1=2;
constraint,expr= qy1=2;
constraint,expr= abs(qx2)<2;
constraint,expr= abs(qy2)<2;
jacobian,calls=10,bisec=3;
endmatch;
```

The initial value of all the variables are implicitly set to 0. In this way the algorithm will try to minimize the sum of the squares of the variables while fulfilling the constraints

An analysis of the different matching techniques can be summarized as follow:

`JACOBIAN` It solves problem in 5 iteration for a total of 160 calls of the macro. It exploits the larger number of variables in order to minimize the distance of the final solution from the starting value via the LQ decomposition.

`SIMPLEX` It doesn't converge to a solution.

`LMDIF` It can be used for solving a simplified version of the problem. The 32 variables has to be grouped in 4 set equal valued variables in order to make the number of constraints is equal to the the number of variables. The solution found requires 50% more sextupole strength than original solution provided by `JACOBIAN`.

CONCLUSION

The `JACOBIAN` procedure combined with the `USE_MACRO` construct has been successfully used for finding smooth optics transitions, estimations for the tunability of LHC insertion regions ([4]) and for non-linear chromaticity correction([3], [5],[6]).

The `JACOBIAN` algorithm performs better than the other methods for problems almost linear with differentiable constraints when the starting point is close to the solution and the problem has more variables than constraints.

The large flexibility allows the user to better adapt to the complexity of his matching problem.

The `USE_MACRO` mode extends the MAD-X matching procedure to include simultaneous constraints on a set of user defined observables.

Combing the `USE_MACRO` with MAD-X expressions allows to use of present and future MAD-X modules for matching modules.

REFERENCES

[1] http://cern.ch/mad/ MAD-X manual.

[2] Anderson, E. et al; LAPACK Users' Guide; Society for Industrial and Applied Mathematics 1999.

[3] Skowronski, P.; De Maria, R.; Schmidt, F.; Forest, E.; New Features of MAD-X Based on PTC; this conference.

[4] De Maria, R.; LHC IR Upgrade: a Dipole First Option. LHC-LUMI-05, Arcidosso, Italy, 31 Aug - 3 Sep 2005.

[5] De Maria, R.; Brüning, O.; Raimondi, P.; LHC IR Upgrade: A Dipole First Option with Local Chromaticity Correction EPAC'06, Edinburgh, Scotland, UK, 26 - 30 Jun 2006.

[6] De Maria, R; Brüning, O; A Low Gradient Triplet Quadrupole Layout Compatible with NbTi Magnet Technology and $\beta^* = 0.25$m EPAC'06, Edinburgh, Scotland, UK, 26 - 30 Jun 2006.