LONG-TERM SIMULATION OF BEAM-BEAM EFFECTS IN THE TEVATRON AT COLLISION ENERGY

A. Kabel*, Y. Cai, SLAC, Stanford, CA 94025, USA
T. Sen, FNAL, Batavia, IL 60510, USA

Abstract

The beam-beam effect is a significant source of nonlinearities in the Tevatron. We have developed a code which allows us to estimate its contribution to the finite lifetime of the anti-proton beam, both at collision and injection energy, by tracking realistic particle distribution for a high number of terms and extrapolating from the particle loss rate. We describe the physical modeling underlying the code and give benchmarking results.

INTRODUCTION

In its current operating mode, the Tevatron operates with 36+36 (anti-)proton bunches circulating in the same beampipe on separated helices and being brought to collision in two interaction points, B0 and D0 at an collision energy of 980GeV.

Thus, each bunch will experience 2 design and 70 parasitic collision with an antagonist bunch. The pattern of parasitic crossings experienced by the weak bunch will depend on the cogging stage and the number of the bunch in its train; a three-fold symmetry introduces equivalence classes of bunches, reducing the number of discernible bunches to 12.

In the injection stage with an energy of 150GeV, the helices are completely separated, leading to 72 parasitic collisions. In both cases, the beam-beam effect will exert strongly non-linear forces on the anti-proton bunch, possibly leading to particle loss due to diffusion or incoherent resonances. In the past, we have been able to predict particle loss rates and extrapolate lifetime signatures for different operating parameters with long-term tracking studies using a crude approximation of the lattice dynamics of the Tevatron. In this paper, we study a more careful implementation aimed at preserving the more important non-linear effects of the lattice, in particular chromaticity, while retaining the high tracking speeds necessary. We can not expect absolutely to predict beam lifetimes (at injection and collision, the typical timescales are $10h$ and $100h$; requiring about 10^{12} and 10^{14} particles·turns simulation effort, resp.), but attempt to establish signatures of beam loss rates for different operational parameters.

PHYSICAL MODELLING

The goal of our tracking studies is to obtain insight into beam loss rates and lifetimes by tracking macro-particles for very long times and recording their amplitude vs. time behavior, thus going beyond the more conventional dynamic aperture calculation which only provide a limited insight about the machine behavior.

Clearly, a full simulation of the machine–full number of particles, full lattice, full machine-time simulation–is beyond today's computational resources. We have to restrict ourselves to

- a simplified model of the lattice. While retaining all beam-beam interactions exactly, the rest of the lattice can be lumped together as either linear elements; chromatic elements (see below), or can be treated element-by-element, where multipoles are handled to the required order, other elements symplectically with up-to-cubic hamiltonians.
- a restriction of the number of turns and the number of particles; the maximum $N_{Turns}N_{Particles}$ is limited by the speed and number of CPUs available; a rough estimates shows that we require $N_{Turns}N_{Particles} \geq 10^{10}$ to predict lifetimes in the range of hours.
- the restriction of the number of turns forces us to extrapolate from the loss rate vs. turn number to its long term behavior. Solutions for the diffusion equation with absorbing boundary conditions (provided by the scraping aperture) behave as $\propto e^{-\sqrt{t/T}}$ and $\propto e^{-t/T}$ for small and large apertures, resp.; we use Ts obtained by a fitting to both asymptotic behaviors to characterize loss rates.

THE TRACKING CODE `dumbbb`

To address the issues laid out in the previous sections, a code, '`dumbbb`', was developed. Great care was taken to yield maximum speed for raw tracking while retaining the flexibility of a general-purpose tracking code; i. e., easy set-up and manipulation of beamlines and parameters as well as flexibility in the choice of accuracy of the physical model and the resulting speed trade-off.

The structure of the code and the programming techniques used as well as a beam dynamics C++ library used as a programming environment are described in more detail in [5].

Lattice Manipulation

`dumbbb` will read a MAD 8.x conformant input file. As the MAD syntax seems not to be formally defined, the

* akabel@stanford.edu

0-7803-8859-3/05/$20.00 ©2005 IEEE

parser is restricted to the well-defined subset of files generated by MAD's `SAVE` command.

The read file is converted to an internal representation, allowing for manipulation of elements and beamlines. In particular, a MAD definition can be expanded into a flat lists of beamline elements; flat lists, in turn, can be subject to insertions at arbitrary longitudinal positions, including within elements, which are split in two sections. This is used for inserting beam-beam elements at positions according to the currently examined bunch and cogging stage.

Elements

The flat list of elements acts as a 'class factory' for tracking elements proper. Each MAD element will generate a finite number of tracking elements, acting on particles' phase space vectors. The currently implemented elements and their implementation methods are:

- *Drift Spaces.* Either linear or first-order chromatic.
- *Quadrupoles.* Either linear or first-order chromatic. First-order chromaticity is implemented either by the exact solution of the 3rd-order Hamiltonian or by a sequence of thick-lens matrices and chromatic thin-lens kicks.
- *Sector Bends.* Either linear or first-order chromatic. Chromaticity is implemented as a sequence of thick-lens matrices and kicks according to the 3rd-order Hamiltonian.
- *Multipoles, RF Cavities, Electrostatic Separators* Implemented by a sequence of, possibly chromatic, drift spaces and thin-lens kicks.
- *Beam-Beam Elements, Tune Prints, Apertures, Counters.* See below.

Lumping and Optimization

The generated beamline and its elements can act either on raw floating-point phasespace vectors $\boldsymbol{x}$ or on differential-algebraic (DA) objects $\boldsymbol{x}+\alpha_1 \mathrm{d}\boldsymbol{x}+\alpha_2 \mathrm{d}\boldsymbol{x}\vee\mathrm{d}\boldsymbol{x}+\ldots$. At program startup, the closed orbits of both rings are determined (by Newton iteration) and differential-algebraic maps around it are constructed.

We then find the (3 each) eigendistributions' correlation matrices of the proton beam, use the emittances to construct beam matrices which, together with the orbit offsets with respect to the anti-proton orbit, are used to construct beam-beam elements at the 72 crossing points. Strictly speaking, this procedure should be iterated until convergent.

The interjacent elements can then be lumped together in a symplectic fashion, either by using a chromatic map (see below) or just a linear map. If no lumping is used, an optimization steps joins elements whose adjacent maps can be simplified, e.g., a sequence of two drift spaces, linear transfer elements, or thin-lens multipoles.

Using the full Tevatron lattice as of March, 2005, which, in our representation, comprises 12902 tracking elements, we obtain the tracking speeds given in 1. It should be noted that, given the estimated turn number from above, tracking element-by-element to obtain absolute lifetimes is not without the realm of the possible.

Table 1: `dumbbb` Tracking Speeds for Different Physical Models (on a 1.8GHz Xeon Processor

Model	**Speed[turns/s]**
200 Sextupoles + Chroma lumping	5700
9 6D slices + 70 4D parasitics	6200
+ 200 Sextupoles + Chroma lumping	3200
+ elment-by-element	610

Code Validation

We have carefully checked the validity of the tracking part of our code by comparing to established tools such as MAD 8.x. Optical functions on the zero orbit generally agree to CPU precision, on the helices, deviations of 10^{-5} can be observed, which can be traced back to the thin-lens approximation for the electrostatic separator in `dumbbb`.

Element-by-element tune prints (calculated by the Laskar method in `dumbbb`) show excellent agreement. Symplecticity of all element and lump maps is checked at program startup by tracking DA vectors; usually, the quantity $\sqrt{\sum_{ik}(M^\top JM-J)^2_{ik}} < 10^{-12}$.

A Symplectic Prescription for Chromatic Lumping

One of the dilemmas of high-order 'lumped' tracking is the non-symplecticity of a truncated power series expansions of a section map. This can be remedied by very high-order expansion, which is computationally expensive, or by any of a number of symplectification approaches.

For the problem at hand, chromatic effects seem to be of significant importance. When handling sextupoles in dispersive sections correctly, the chromaticity of the interjacent section also has to be implemented correctly to obtain the correct net chromaticity.

This can be achieved by writing the total map of a section as $\tilde{M} = M(1-p_tG)+\mathrm{O}(p_t^2) = Me^{-p_tG}+\mathrm{O}(p_t^2)$, where M is the linear part of the map and G is a symplectic generator acting on the transverse subspace; it is obtained by evaluating the third-order map around the dispersion orbit. e^{-p_tG} is symplectic by construction. By writing $H=-\frac{1}{2}JG$ and expressing H in an eigenbasis of G, the Hamiltonian reduces to any of three forms:

- $H=\alpha pq$, a rescaling operation
- $H=\frac{\beta}{2}(p^2+q^2)$, a harmonic oscillator
- $H=\frac{\gamma}{2}(p^2-q^2+P^2-Q^2)+\delta(qP-Pq)$, an isotropic repulsive oscillator in a rotating frame of reference.

All these case can readily be evaluated either in closed form or as a sequence of 3 thin-lens kicks by determining the eigenvectors of JG and constructing a canonical basis in which it has harmonic oscillator form, reducing the problem to at most two quadrupole transformations.

Beam-Beam Effects

At present, the Tevatron is operated in the weak-strong regime of the beam-beam interaction. Assuming gaussian distributions, the effect on the antiproton bunch can then be calculated using the Bassetti-Erskine formula[3], which involves the evaluation of the complex error function.

While not the dominant part of CPU time for some of the more extensive beamline models, care was taken to find an efficient implementation. We benchmarked several numeric implementation and found the prescrition given in [4] to be fastest. Other approaches have been used in similar codes, such as Pade approximations and grid interpolations.

When $\beta_{x,y} \leq c\sigma_t$, the hourglass effect becomes important. We then have to slice the strong bunch and to include the full 6d dynamics of the weak bunch; the symplectic prescription used is given in [2]. We use several optimized functions, distinuishing the cases of

- Four-dimensional:
  - Coupling: tilted beam
  - No coupling: upright beam
- Six-dimensional:
  - Coupling: slice with position-dependent tilt
  - Coupling: beam with constant tilt
  - No coupling: upright beam

Weak Bunch Population

Lifetime calculations are based on particle migration rates across a boundary in configuration space. It is safe to assume that, for realistic distances of that boundary, most of the particles in that bunch will never cross that boundary within the simulated time; these stable particles will be close (in action space) to the closed orbit. A previous version of `dumbbb` consequently provided a 'de-coring' option, restricting bunch population to particles far (in action space) away from the closed orbit. Because of the high dimensionality of phasespace, the action radius of the omitted core has to be chosen quite large to result in an appreciable reduction of the particle number. This is unsatisfactory, however, as for high actions the deformation of surfaces of equal action due to the nonlinear dynamics becomes significant. The present version of `dumbbb` thus uses a weighted-macroparticle approach: The weak bunch is represented by particles equidistributed in radial and angular space in six-dimensional spherical coordinates; particles carry a

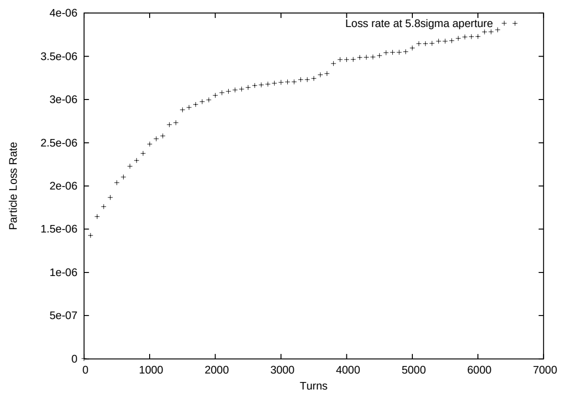

Figure 1: Typical Particle Loss Rate from Simulation Run

gaussian weight $\exp -r^2/2r^5 \mathrm{d}r\mathrm{d}\mathrm{d}\vartheta_4 \sin^4 \vartheta_1 \ldots \vartheta_5$. The cutoff radius of the distribution can be chosen freely. The radial and angular distribution is generated from an unique integer particle tag by means of a Halton pseudorandom sequence over the first 6 primes, thus reducing charge distribution noise and facilitating parallelization and reproducibility of simulation runs. It should be noted that, by weighting particles and recording their migration rates, *p*ost mortem emittance parameter scans for loss rates are possible by reassigning weights according to different emittances.

Simulation Runs

We have ported the code to the DOE's NERSC facility SP computers and performed runs on 256 nodes each, varying chromaticity in steps of 5 from 0 to 20. An evaluation of the results will be the subject of a forthcoming paper. A typical particle loss rate is shown in figure (chromaticity 5, aperture 5.8σ).

ACKNOWLEDGMENTS

Work supported in part by the U. S. Department of Energy under contract number DE-AC02-76SF00515.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U. S. Department of Energy under Contract No. DE-AC03-76SF00098.

REFERENCES

[1] A. Kabel, Y. Cai, B. Erdelyi, T. Sen, M. Xiao, Proceedings of the 2003 IEEE Particle Accelerator Conference (2003)

[2] K. Hirata, H, Moshammer, F Ruggiero, Part. Acc. **40**,205 (1993)

[3] M. Bassetti, G Erskine, CERN ISR TH80-06 (1980)

[4] F. Matta and A. Reichel, Math. Comp. **25**, 339 (1971)

[5] A. Kabel, A C++ Framework for High-Speed, Long-Term Particle Trackign Studies, This Conference.