

# PARALLEL COMPUTATION OF BEAM-BEAM INTERACTIONS INCLUDING LONGITUDINAL MOTION

F.W. Jones, TRIUMF, 4004 Wesbrook Mall, Vancouver, Canada V6T 2A3  
W. Herr, CERN, CH-1211 Geneva 23, Switzerland

## Abstract

In beam-beam macroparticle simulations for collider rings, the accurate determination of the incoherent spectrum and potentially unstable coherent modes requires (1) large numbers of collisions, and (2) accurate electric field solutions at each collision. On a single processor, a self-consistent simulation typically uses a 2D model of the beam-beam interaction in order to achieve a reasonable computation time, however for the long ( $\sim 0.3\text{m}$ ) bunches in the LHC we wish to include the third dimension in order to account for effects such as longitudinal motion, crossing angle, and the beam size and density variations. We describe here a parallel algorithm, developed with MPI on a small commodity Linux cluster, to extend our simulation code BeamX from 2D to 3D using longitudinal subdivision (slicing) of the bunches. Although this paper concentrates on the computing methods, some performance trials and example results will also be shown.

## INTRODUCTION

In investigating coherent beam-beam effects in colliders, one goal of simulations is to identify potentially unstable modes and possible damping mechanisms. In this respect, useful qualitative descriptions can be obtained by simplifications such as rigid-bunch and soft-gaussian models, but herein we restrict ourselves to “self-consistent” simulations in which the electric fields are computed directly from the ensemble of macroparticles without assumptions as to the nature of their distribution.

For a 2D model of the beam-beam forces, such simulations are feasible on today’s desktop computers using conventional particle-mesh field solvers or, for parasitic collisions, grid-multipole[1] and shifted Green’s function methods[2]. However, the 2D treatments omit longitudinal effects such as: (1) longitudinal variation in transverse beam size (hourglass effect); (2) variation of longitudinal density (affects the strength of the beam-beam forces); (3) the effect of beams crossing at an angle instead of head-on; and (4) the coupling of longitudinal motion with these effects. It is therefore of interest for the LHC and other colliders to extend our simulations to 3D if it can be done without prohibitive computational cost.

## EXTENSION TO 3D

As in space-charge simulations, there are various numerical methods by which one can compute the bunch-to-bunch forces in 3D, and these methods are generally practi-

cal only for parallel computation. In our case, to show sufficient detail in the coherent frequency range requires  $\sim 10^5$  or more simulated collisions, suggesting that the problem lies in the supercomputer realm. However, the small transverse-longitudinal aspect ratio of the LHC bunches allows us to seek economies by using a rather coarse-grained subdivision of the solution domain in the longitudinal direction. In conjunction with a  $36 \times 36$  or more computational mesh in the transverse plane, we divide the beam longitudinally into  $\sim 10$  segments. This “bunch slicing” approach[3, 4] is applied to both beams, and the beam-beam collision is treated as a series of 2D slice-slice interactions (see Figure1).

For particle-mesh solvers it is natural to parallelize the field solver in the mesh computation stage. For the BeamX code, however, the fast-multipole solver in use does not lend itself readily to parallel decomposition because of its adaptive subdivision and hierarchical data structures. Hence, we have pursued a more fundamental parallelism, that of the pairwise slice interactions, which may be done independently on different processors provided the causal relationships are maintained. For  $N$  slices in each beam, the number of overlapping slices during the collision varies from 1 to  $N$  to 1, allowing a parallel speed-up of roughly  $N/2$  by the application of  $N$  processors.

## PARALLEL ALGORITHM

The design and implementation of the parallel version of BeamX were done using a small commodity Linux cluster, representing a low-cost resource which is able to handle small numbers of slices. Utilizing the MPI toolkit for inter-process communication, the fundamental division of labour is between a *master* process and several *slave* processes, as follows:

### Master:

- Filling of slice data structures (longitudinal binning)
- Dispatch of slice-data to Slaves
- Receipt of slice-data to Slaves and un-binning
- Longitudinal transport to next IP
- Compiling statistics and all program output

### Slave:

- Receipt of slice-data from Master
- Computation of electric fields
- Application of beam-beam forces via symplectic map
- Transport in transverse plane to next IP
- Dispatch of slice data to other Slaves and Master

As described in the next section, the Slaves require a nearly complete set of coordinates  $(x, x', y, y', \epsilon = \Delta E/E)$  for the macroparticles in a slice-pair to do their work, entailing  $\sim 0.5\text{MB}$  of data per 10k particles. With this communication overhead it is imperative to minimize the message volume between processes, and to this end a topology has been devised in which one beam's slices "stay at home", i.e. are resident in the Slave processes, and the other beam's slices "go visiting", i.e. are passed between Slave processes. This is illustrated schematically in Figure 1. Once a Slave has received its assigned Beam 1 slice from the Master, it is ready to receive Beam 2 slices to interact with it. The Master (process 0) sends Beam 1 slices  $1, \dots, N$  to Slave processes  $1, \dots, N$  respectively. It sends all the Beam 2 slices to Slave 1 in sequence. When Slave 1 receives a Beam 2 slice, it does the pair-interaction, updates the coordinates of each slice, and passes the Beam 2 slice on to Slave 2. Slave  $n, 1 < n \leq N$ , receives a Beam 2 slice from Slave  $n-1$ , does the pair-interaction, and passes it on to Slave  $n+1$ , or to the Master if  $n = N$ . Once a Slave has dealt with the last Beam 2 slice, it is finished with slice-interactions for this collision and can send its resident Beam 1 slice data to the Master.

In this scenario the Master does not have to perform any control or synchronization functions for the Slaves. All processes are essentially free-running and the data flow itself imposes and maintains the proper ordering of events.

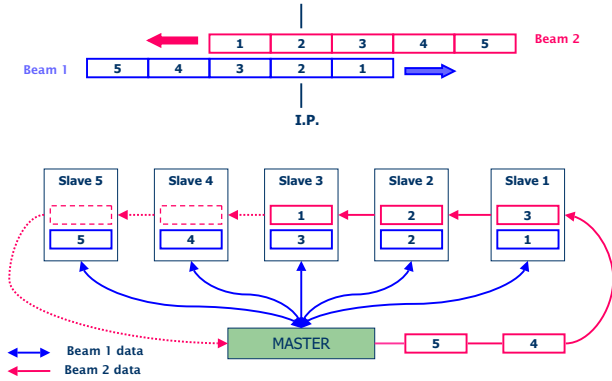


Figure 1: Inter-process communication scheme for parallel computation (case of 5 slices per bunch)

## DETAILS OF SLAVE WORK

The interaction of a slice-pair  $i-j$  occurs at the interaction point (IP) if  $i = j$ , but otherwise occurs at distance  $s = (j - i) * H/2$ , the collision point (CP), where  $H$  is the slice length. Since macroparticle transport through the rings is done by an IP-to-IP map, it is convenient to "drift" the particles forward or backward to the CP, evaluate and apply the beam-beam forces, and drift them to the IP again. In extending to 3D the angular kicks due to transverse forces are now accompanied by energy kicks due to longitudinal forces. To preserve symplecticity in the

6 phase-space variables the "synchro-beam" mapping[5] is employed:

$$\begin{aligned} x^{\text{new}} &= x - sF_x/2 \\ x'^{\text{new}} &= x' + F_x \\ y^{\text{new}} &= y - sF_y/2 \\ y'^{\text{new}} &= y' + F_y \\ \epsilon^{\text{new}} &= \epsilon + F_x(x' + F_x/2)/2 + F_y(y' + F_y/2)/2 \\ &\quad - s(\sigma_x^2 F_x + \sigma_y^2 F_y) \end{aligned}$$

where  $F_x$  and  $F_y$  are the angular deflections of a particle which has coordinates  $(x, y)$  when it passes the IP and encounters a given slice of the opposing beam at distance  $s$  from the IP, and  $\sigma_x^2$  and  $\sigma_y^2$  are the variances of the opposing beam.

The above mapping applies to beams moving on parallel trajectories. If the beams cross at an angle then a Lorentz transformation[6] is applied to each slice in turn so that it is oriented parallel to the opposing slice, after which the electric fields are computed, the synchro-beam mapping is done, and then the inverse transformation is performed.

After these iterated series of transformations for slice-pairs, all coordinates are transformed back to the IP in that their relative positions are consistent with the opposing bunches being in the middle of their collision, i.e. centered at the IP. It is then straightforward to transform all coordinates to the next IP by the conventional transfer matrix and longitudinal difference equations, according to the lattice optics and the applied RF voltage.

## PERFORMANCE

We performance-tested the parallel BeamX code on three different platforms:

1. Linux, Pentium IV 1.6 GHz, 100Mb Ethernet, MPICH toolkit
2. Linux, Pentium III 1.4 GHz, 1Gb Ethernet, LAM MPI toolkit
3. Linux, Pentium III 1.4 GHz, Dolphin Scalable Coherent Interconnect (SCI), SCALI MPI toolkit

On each platform, 1000 collisions were executed for varying numbers of slices  $N$ , where the number of parallel processes was  $N + 1$ . In order to estimate both the communication overhead and the parallel speedup, each job was run normally with each process running on a separate CPU, and with all processes running on a single CPU. As seen in Figure 2, there is significant communication overhead but the  $N^2$  complexity is reduced to nearly linear scaling by the parallel algorithm.

With conventional ethernet communications, increasing the bandwidth by a factor of 10 yielded a significant performance improvement for small numbers of slices, even with somewhat slower processors. For larger numbers of slices, the communication burden becomes proportionally smaller and the parallel efficiency becomes about the same for the three communication hardware/software configurations. In

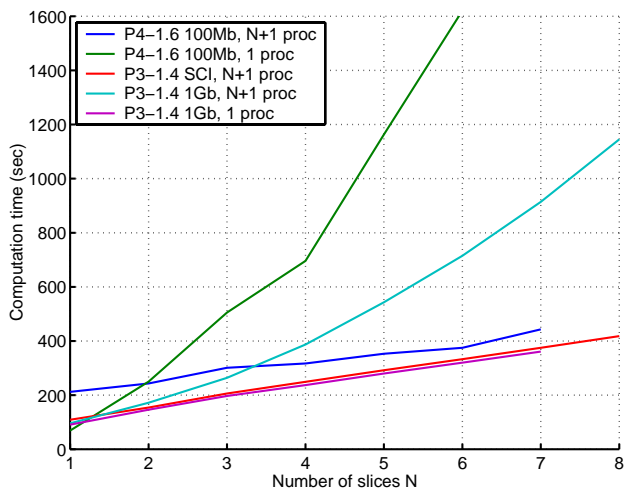


Figure 2: Parallel and single-node performance timings

particular the SCI showed no benefit over Ethernet in our trials. This may be a matter of tuning and remains to be investigated.

## EXAMPLES

The 3D-extended BeamX code has been applied to some test cases for the LHC with a single interaction point. For the LHC there is no dispersion in the interaction region and the variation of beam size along the bunch length (hour-glass effect) is quite small. The dominant longitudinal effects are due to the synchrotron motion and the crossing angle. Figure 3 shows the FFT spectra of the coherent frequencies (beam centroid motion) for runs of  $2^{17}=131072$  turns with 5 longitudinal slices and 50000 macroparticles, with an applied RF of 40 MV/turn (nominal synchrotron tune  $Q_s=0.00335$ ) and with crossing angles of 0, 150, and 300  $\mu$ radian. The crossing angle induces a relativistic projection of the beam-beam force and hence reduces the beam-beam tune shift as seen by the  $\pi$  mode frequency which shifts from its normal value of  $-1.21$  to  $-1.10$  and  $-0.92$ , respectively. With RF on, the synchrotron tune is comparable in size to the beam-beam tune shifts as revealed by the multiple sidebands.

As in soft-Gaussian simulations with bunch slicing[4] a relatively modest number of slices and macroparticles suffice to model these basic phenomena. Running the same cases with  $>5$  slices and  $>50000$  macroparticles showed little difference in the coherent spectra, with the relevant features being essentially unchanged.

## CONCLUSIONS

A three-dimensional self-consistent multiparticle beam-beam simulation has been developed using coarse-grained longitudinal subdivision and parallel programming techniques. The implementation via MPI, with a master-slave/slave-slave message-passing algorithm, reduces the computational cost from  $N^2$  to linear scaling with the number of slices  $N$  and makes it feasible to run simulations on

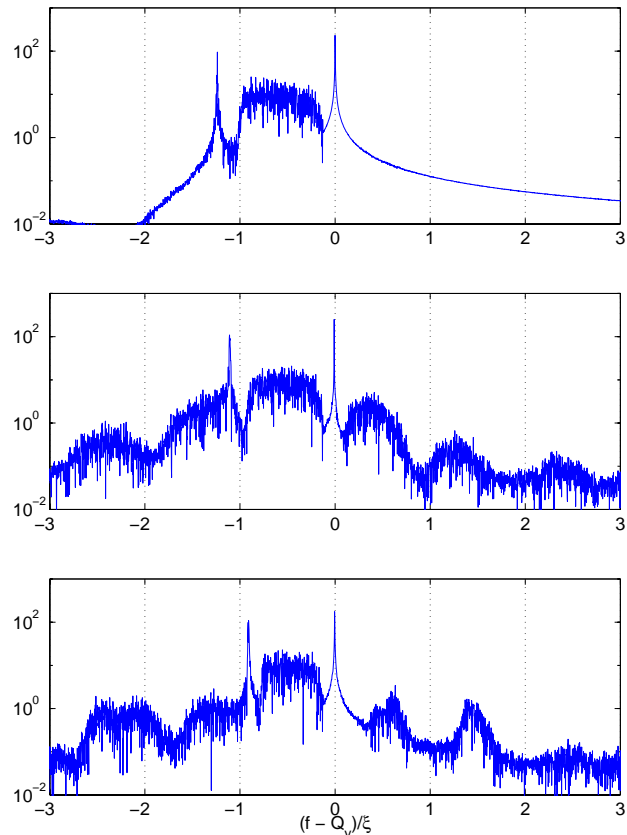


Figure 3: Spectra with 0, 100, and 300  $\mu$ r crossing angle

small low-cost Linux clusters with ethernet. To further decrease computation times, the use of more advanced MPI and/or SCI features may decrease communication overhead, and further parallelism can be sought, such as a parallelized multipole solver. In this work we have confined ourselves to commodity-based hardware/software solutions. The use of shared-memory parallel systems could yield greater parallel efficiency, although at considerably higher cost.

## ACKNOWLEDGEMENTS

We would like to thank Corrie Kost and Steve McDonald for establishing a test-bed Linux cluster at TRIUMF on which we could develop and test the code, and James Pinfold and Bryan Caron for arranging access to the THOR cluster at the University of Alberta Physics Department.

## REFERENCES

- [1] W. Herr, M.P. Zorzano and F. Jones, *Phys. Rev. S.T. Accel. Beams* **4**, 054402 (2001).
- [2] J. Quiang, M.A. Furman and R.D. Ryne, *Phys. Rev. S.T. Accel. Beams* **5**, 104402 (2002).
- [3] A. Jejcic et al., *XVII Int. Conf. on High Energy Accelerators*, Dubna 1998, 220.
- [4] W. Herr and R. Paparella, LHC Project Note 304, CERN 2002.
- [5] K. Hirata, H. Moshhammer and F. Ruggiero, *Part. Accel.* **40**, 205 (1993).
- [6] K. Hirata, *Phys. Rev. Lett.* **74**:12, 2228 (1995).