LA-UR-01-3305
Approved for public release;
distribution is unlimited.

Title: **A LAYER-BASED OBJECT-ORIENTED PARALLEL FRAMEWORK FOR BEAM DYNAMICS STUDIES**

Author(s): J. Qiang and R. D. Ryne

Submitted to:

http://lib-www.lanl.gov/la-pubs/00796399.pdf

A LAYER-BASED OBJECT-ORIENTED PARALLEL FRAMEWORK FOR BEAM DYNAMICS STUDIES

J. Qiang*, LANL, Los Alamos, NM 87545, USA
R. D. Ryne, LBL, Berkeley, CA 94720, USA

Abstract

A three-dimensional time-dependent parallel particle-in-cell framework has been developed to model complex accelerator systems. This framework has been designed based on object-oriented methodology using a layered structure. The layer-based object-oriented software design helps to encapsulate both the details of the physical application and its parallel implementation and gives the program good maintainability and extensibility. The new framework is currently being applied to the study of the LEDA beam halo experiment at the Los Alamos National Laboratory. Using the new framework running on a parallel supercomputer we can simulate, with high resolution, multiple bunches propagating and merging through the LEDA system, including the effects of interbunch and intrabunch 3D space-charge forces. Such high resolution multi-bunch simulation is beyond the capability of current serial beam dynamics codes.

1 INTRODUCTION

Macroparticle simulation plays an invaluable role in the study of charged-particle beams transporting through accelerators. A number of computer programs using macroparticle simulation have been developed during the last few decades [1, 2, 3, 4, 5]. However, as far as we know, at present, there is not a totally object-oriented program doing three-dimensional macroparticle simulation of intense beams using parallel computers in the time domain in the accelerator community. With growing interest in high resolution large scale simulation on parallel computers, a computer program based on object-oriented software design will have better maintainability, reusability and extensibility, resulting in a longer lifetime. In this paper, we have developed a multi-layer based object-oriented software framework for accelerator beam dynamics system study.

2 PHYSICAL MODEL AND COMPUTATION METHODS

We will consider charged particles moving in an accelerator that can be described by the Poisson-Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{r}' \cdot \frac{\partial f}{\partial \mathbf{r}} + \mathbf{p}' \cdot \frac{\partial f}{\partial \mathbf{p}} = 0 \qquad (1)$$

where f denotes the distribution function of particles, a superscript prime denotes the derivative with respect to time, $\mathbf{r}$ is the spatial coordinate, $\mathbf{p}$ is the conjugate momentum with $\mathbf{p}' = \mathbf{F}$, where $\mathbf{F}$ is the force including the contributions from both the external applied field F_{ext} and the self (space-charge) force F_{self}. The space-charge force in this equation is a mean-field approximation of the N-body micro-particle Coulomb force. In the beam frame, the space-charge force can be obtained from the solution of Poisson's equation

$$\nabla^2 \phi(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0} \qquad (2)$$

and

$$\mathbf{F}_{self} = -q\nabla\phi \qquad (3)$$

where ϕ is the electrostatic potential in the beam frame, ρ is the particle spatial charge density, and ϵ_0 is the vacuum permittivity. The solution of the Poisson equation can be written as

$$\phi(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}')\rho(\mathbf{r}')d\mathbf{r}' \qquad (4)$$

where G is the Green's function of the Poisson's equation. For three-dimensional open boundary conditions, the Green's function can be written as:

$$G(\mathbf{r}, \mathbf{r}') = \frac{1}{4\pi\epsilon_0|\mathbf{r} - \mathbf{r}'|} \qquad (5)$$

The Poisson-Vlasov equations are solved using the particle-in-cell approach. Here, macroparticles are generated with the same charge to mass ratio as the real particles in the beam bunch. The equations of motion for the particles are integrated using a second order leap frog algorithm. Within each step, the particles' spatial coordinates are advanced a half step using their present velocities. Then the particles are deposited onto a three-dimensional spatial grid to obtain the charge density distribution. Using the charge density distribution on the grid, the convolution Eq. 4 can be calculated using a FFT based algorithm [6]. The electric fields on the grid are calculated from the potential using a central finite difference scheme. The fields on the grid are reinterpolated back to the particles to obtain the space-charge force on the particles. The particles' momenta are advanced for one step using both the external applied force and the space-charge force. Finally, the particles' coordinates are advanced for another half step using the updated velocities to complete a full step.

* jiqiang@lanl.gov

3 OBJECT-ORIENTED MULTI-LAYER SOFTWARE DESIGN FOR BEAM DYNAMICS SYSTEM SIMULATIONS

A multi-layer based object-oriented software design is a system method of organizing the subsystems into an ordered set of "virtual worlds" [7]. In this design, objects identified from the analysis are organized into different physical modules. The physical modules are built into an ordered layer structure. The objects in the lower layer provide the service for the objects in the upper layer. Usually, the upper layer is related to the problem domain and the lower layer is related to the available resources. In applying this design to beam dynamics system simulations, we have defined a four-layer framework. These four layers are the data structure layer, function layer, application layer and control layer. A schematic plot of these four layers together with physical modules in each layer is given in Fig. 1. Here,

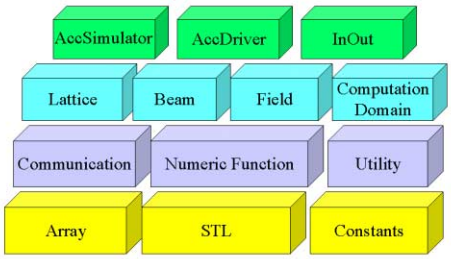

Figure 1: A schematic plot of the multi-layer structure for beam dynamics system simulations.

the data structure layer contains an array module, standard template library module and constant module. Each module may consist of one or more class objects working together to fulfill some given functions. The classes in this layer are very generic and can be reused for different problem domains. The function layer contains a communication module, numeric function module and utility module. The communication module is a group of classes handling explicit communication among different processors in the parallel implementation. The numeric function module provides all numeric function libraries used in the simulation. The utility module provides auxiliary functions for the simulation, e.g. a sorting function. The modules in this layer are relatively independent of the details of beam physics and could be reused in other system simulations. The application layer contains a beam bunch module, beam line element lattice module, field module and computation domain module. The modules in this layer are directly associated with the beam dynamics system. The beam bunch module defines the class information of a charged particle beam in the accelerator. The lattice module consists of classes defining external focusing and acceleration elements. Run-time polymorphism is used in the implementation so that a single operation using the function of the beam line element base class can automatically select the appropriate function from different concrete external beam line element objects to execute. The field module defines the classes for dealing with electromagnetic fields generated by the moving beam bunch. The computational domain module contains the classes describing the global and local geometry domain in the simulation. Since the modules in this layer are directly related to the beam physics, they might not be reusable in the other fields of study. The control layer contains a simulator module, driver module and input/output module. The simulator module contains function classes to set up the simulation environment, e.g. linac simulator, and do the simulation. The driver module provides a driver needed to run the simulation. The input/output module contains classes providing input and output functions. The object-oriented multi-layer structure gives the program good resuability, maintainability and extensibility. The classes in the lower layers can be reused in different applications. A class object is clearly defined in association with a given module and layer. New function modules and class objects can be added to different layers without affecting the other modules or classes. A software system with good maintainability and extensibility could have a longer life in principle.

4 PARALLEL IMPLEMENTATION AND APPLICATION

The computational model described above is implemented on high performance parallel multiprocessor computers using a message passing programming paradigm. A two-dimensional domain decomposition approach has been employed in the parallel implementation. The physical processors are mapped onto a two-dimensional logical processor grid. Each processor has a unique identification number and contains a local computation domain. The particles with their spatial coordinates inside the local computation domain are assigned to that processor. Computation is done simultaneously on all processors with local data. When particles move to a different computation domain, communication will be used to send these particles to the corresponding domain. In the Poisson solver and field calculation, when the information of more than a local processor is required, communication will be used to transfer the data to the local processor. A detailed discussion of the parallel implementation can be found in reference [5].

As an application, we have applied this new framework to the study of the LEDA halo experiment at the Los Alamos National Laboratory [8]. In this experiment, a mismatched beam is transported through a periodic focusing system. The system consists of 52 alternating-focusing quadrupole magnets with a focusing period of 41.96 cm. The gradients of the first four quadrupole magnets can be adjusted to create a mismatch that excites the breathing mode or the quadrupole mode. Since there is no longi-

tudinal focusing, the bunched beam out of the RFQ will gradually debunch and merge longitudinally through the system. Fig. 3 shows the $X - Z$ plot of three bunches

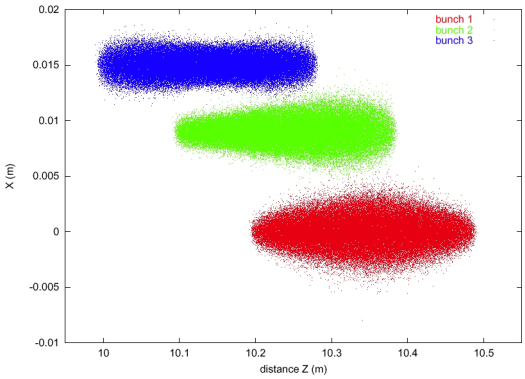

Figure 2: $X - Z$ distribution of three beam bunches near the end of halo channel.

near the end of transport system from a three-bunch simulation. It is seen that there is significant overlapping among the three bunches from the debunching of the beam. Only the middle bunch will be used as a comparison with experimental data since it has the correct boundary conditions. A longitudinal periodic boundary condition is applied to the whole computation domain containing the three bunches. Fig. 3 and Fig. 4 give the simulation results of the transverse beam rms size and maximum amplitude for the breathing mode and the quadrupole mode, plotted at the center of the drift spaces between quadrupole magnets. The physical parameters for the simulation are $I = 75mA$, $E = 6.7MeV$ and $f = 350MHz$. From Fig. 3, the two transverse components of the breathing mode are in phase, while the quadrupole mode in Fig. 4 has the two components out of phase. Work is now underway to compare the simulation results with experimental measurements.

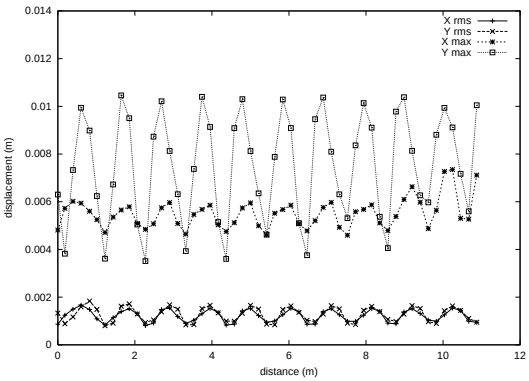

Figure 3: Transverse beam size as a function of distance for the breathing mode in the LEDA halo experiment.

5 ACKNOWLEDGMENTS

We would like to thank Drs. K. Crandall and T. Wangler for suggesting using multiple bunch simulation for the LEDA experiment. This work was performed on the Cray T3E and IBM SP at the National Energy Research Scientific Computing Center located at Lawrence Berkeley National Laboratory, and the SGI Origin 2000 at the Advanced Computing Laboratory located at Los Alamos National Laboratory. This work was supported by the U.S. Department of Energy, Office of Science, Division of High Energy and Nuclear Physics, under the project, Advanced Computing for 21st Century Accelerator Science and Technology. This work was also supported by the Division of High Energy Physics through the Los Alamos Accelerator Code Group.

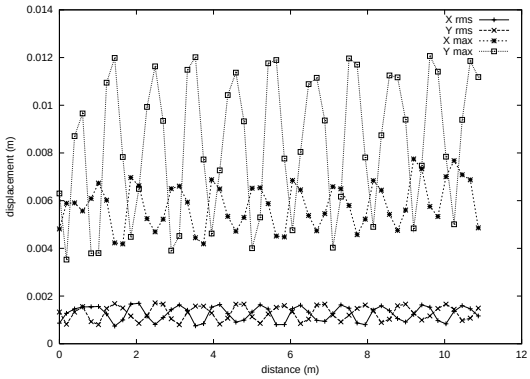

Figure 4: Transverse beam size as a function of distance for the quadrupole mode in the LEDA halo experiment.

6 REFERENCES

[1] M. E. Jones, B. E. Carlsten, M. J. Schmitt, C. A. Aldrich, an d E. L. Lindman, Nucl. Instr. and Meth. in Phys. Res. **A318**, 323 (1992).

[2] A. Friedman, D. P. Grote and I. Haber, Phys. Fluids **B 4**, 2203 (1992).

[3] R. Ryne and S. Habib, in: Computational Accelerator Physics, ed. J. J. Bisognano and A. A. Mondelli, AIP Conference Proceedings 391, Woodbury, p. 377, New York, 1997.

[4] J. Qiang, R. D. Ryne, S. Habib, Comput. Phys. Comm. **133**, 18 (2000).

[5] J. Qiang, R. D. Ryne, S. Habib, V. Decyk, J. Comput. Phys. **163**, 434 (2000).

[6] R. W. Hockney and J. W. Eastwood, Computer Simulation Using Particles, Adam Hilger, New York, 1988.

[7] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, New Jersey, 1991.

[8] T. P. Wangler, et al., "Experimental Study fo Proton-Beam Halo Induced by Beam Mismatch in LEDA," these proceedings.