Simò N-Body Simulation Codes Manual

September 1, 2021

1 Introduction

This is a guide for using the N-body simulations of the electrostatic N-body. The files included "mpi-nbody.fox" and "nbody.fox" are scripts for COSY INFINITY (maintained at Michigan State University), and each requires the file "COSY.bin" to be in the same directory. "COSY.bin" can be created by by running the "cosy.fox" script with COSY. The codes use the Picard iteration method to generate Taylor expansions of the system's ODEs (equation 1). The system consists of N charged particles that can be relativistic as well as non-relativistic, interacting via Coulomb's force while their motion could be subjected to external electromagnetic field. The equations of motion with our choice of scaling in (2) are

$$\frac{d\hat{Y}_i}{d\hat{t}} = \begin{bmatrix} \dfrac{\hat{p}_{x_i}}{\sqrt{f_i^2 + \hat{p}_{x_i}^2 + \hat{p}_{y_i}^2 + \hat{p}_{z_i}^2}} \\ \dfrac{\hat{p}_{y_i}}{\sqrt{f_i^2 + \hat{p}_{x_i}^2 + \hat{p}_{y_i}^2 + \hat{p}_{z_i}^2}} \\ \dfrac{\hat{p}_{z_i}}{\sqrt{f_i^2 + \hat{p}_{x_i}^2 + \hat{p}_{y_i}^2 + \hat{p}_{z_i}^2}} \\ \dfrac{q\,n_i}{m\,c^2}\left[\dfrac{q}{4\pi\epsilon_0}\displaystyle\sum_{\substack{j=1\\ j\neq i}}^{N} \dfrac{n_j\,(x_i - x_j)}{\gamma\left[(x_i-x_j)^2 + (y_i-y_j)^2 + \gamma^2\,(z_i-z_j)^2\right]^{3/2}} + E_{x_i} + c\,(\hat{v}_{y_i}B_{z_i} - \hat{v}_{z_i}B_{y_i})\right] \\ \dfrac{q\,n_i}{m\,c^2}\left[\dfrac{q}{4\pi\epsilon_0}\displaystyle\sum_{\substack{j=1\\ j\neq i}}^{N} \dfrac{n_j\,(y_i - y_j)}{\gamma\left[(x_i-x_j)^2 + (y_i-y_j)^2 + \gamma^2\,(z_i-z_j)^2\right]^{3/2}} + E_{y_i} + c\,(\hat{v}_{z_i}B_{x_i} - \hat{v}_{x_i}B_{z_i})\right] \\ \dfrac{q\,n_i}{m\,c^2}\left[\dfrac{q}{4\pi\epsilon_0}\displaystyle\sum_{\substack{j=1\\ j\neq i}}^{N} \dfrac{n_j\gamma\,(z_i - z_j)}{\left[(x_i-x_j)^2 + (y_i-y_j)^2 + \gamma^2\,(z_i-z_j)^2\right]^{3/2}} + E_{z_i} + c\,(\hat{v}_{x_i}B_{y_i} - \hat{v}_{y_i}B_{x_i})\right] \end{bmatrix}. \tag{1}$$

2 Codes Description

The integrator automatically selects an optimal time *stepsize* and an optimal *order* for each particle i. Once all particles functions are expanded, the particles are distributed over a number of bins in order for them to be propagated as needed (only particles in the fist bin are propagated at each time step) until the end of the simulation time. The user can use the type of time-bins from the two available options: bins of equal time-widths, and bins of equal number of particles. In addition, the initial and final momentum and total energy of the system are calculated. For more information about the Simò N-body integrator, we refer the reader to the Simò N Body Integrator publication.

Part of the main inputs in the codes are the initial conditions: positions, scaled momenta, charges' and masses' factors. Each particle i is associated with an array $\hat{Y}_i$ as follow:

$$\hat{Y}_i = (x_i, y_i, z_i, \hat{p}_{x_i}, \hat{p}_{y_i}, \hat{p}_{z_i}, f_i, n_i)\,,$$

where variables are scaled according to

$$\hat{p}_{x_i} = \frac{p_{x_i}}{mc}, \quad \hat{p}_{y_i} = \frac{p_{y_i}}{mc}, \quad \hat{p}_{z_i} = \frac{p_{z_i}}{mc}, \quad f_i = \frac{m_i}{m}, \quad n_i = \frac{q_i}{q}. \tag{2}$$

Here, c is the speed of light, m and q are the proton's mass and charge, respectively. By (1) each particle will have 6 ODEs

$$\frac{d\hat{Y}_i}{d\hat{t}} = \left(\frac{dx_i}{d\hat{t}}, \frac{dy_i}{d\hat{t}}, \frac{dz_i}{d\hat{t}}, \frac{d\hat{p}_{x_i}}{d\hat{t}}, \frac{d\hat{p}_{y_i}}{d\hat{t}}, \frac{d\hat{p}_{z_i}}{d\hat{t}}\right),$$

where $\hat{t} = t\,c$.

2.1 The Parallel Simò Integrator Code

The file "mpi-nbody.fox" is a parallelized version of the Simò integrator code "nbody.fox". The initial conditions can be set manually in the code, or can be read from a file. The other main inputs are:

NP: Number of particles
ORDER: Maximum allowed *order* of Picard iterations
ACCURACY: Required accuracy of the integrator
BINS: Number of time bins
TINITIAL: Initial time of the simulation
TFINAL: Final time of the simulation
DELTLIMIT: Minimum limit for a time *stepsizes*
BINNINGTYPE: Type of time bins: 1 for equal time-widths, 0 for equal number of particles

OUTPUT_STEP: Increment of the simulation time for output
NUM_STEPS: Number of outputs
BOUNDFLAG: Boundaries flag: 1 includes boundaries, 0 no boundaries
XMIN: Minimum limit in x-direction
XMAX: Maximum limit in x-direction
YMIN: Minimum limit in y-direction
YMAX: Maximum limit in y-direction
P: Number of processors

To consider only classical effects, the minimum limit for a time *stepsizes* DELTLIMIT is always set to the diameter of a proton. When the BOUNDFLAG is set to 1, the code will remove the particles that go beyond the x or y limits: XMIN, XMAX, YMIN, and YMAX. The number of processors P is provided using the intrinsic procedure PNPRO, but it can also be set manually in the code. When running the code with a large number of particles, the length of vectors is increased by calling the intrinsic procedure SCRLEN before calling the procedure that contains these vectors.

The main outputs of the code are the particles' positions and scaled momenta at TFINAL (the end of the simulation) where NUM_STEPS should be set to 1 in this case. If desired to get multiple output files at different times, the NUM_STEPS can be set to the number of required output files at increments of the simulation time OUTPUT_STEP. In this case, both TINITIAL and TFINAL can be equal. The final simulation time will be NUM_STEPS×OUTPUT_STEP. The user can set the code to output any other desired information such as optimal *orders* and optimal time *stepsizes*.

Running the parallel code requires using the MPI version of COSY. For instruction on how to get the MPI COSY as a copy or building it from the MPI source files, check the NIU Beam Physics Guide. Depending on the used MPI COSY build, you might get error messages requiring the increase of LMEM or LVAR values. This can be done by increasing these values in the MPI source files of MPI COSY and compile them to get another MPI COSY build of larger memory. The LMEM and LVAR values can be changed manually at each instant of each source file, or can be changed using the cosyresize utility as explained in the NIU Beam Physics Guide.

To run the code, use the following command:

```
$ mpirun -np <number of processors> <name of MPI COSY build> mpi-nbody.fox
```

Currently, the code can simulate a few hundred thousands of particles in a reasonable amount of time. When NP $\geq$ 50000, we suggest to use about 100 bins to speed up the runs. For more efficient runs, we suggest to use equal time-widths bins if the initial distribution is uniform, and equal number of particles bins if the initial distribution is Gaussian.

2.2 The Serial Simò Integrator Code

The serial version of the Simò integrator code is “nbody.fox”. Much like the mpi code, the particles initial conditions can be set manually in the code, or they can be read from a file. The user will also need the following inputs:

NP: Number of particles
ORDER: Maximum allowed *order* of Picard iterations
ACCURACY: Required accuracy of the integrator
BINS: Number of time bins
TINITIAL: Initial time of the simulation
TFINAL: Final time of the simulation
DELTLIMIT: Minimum limit for a time *stepsizes*
BINNINGTYPE: Type of time bins: 1 for equal time-widths, 0 for equal number of particles
B: Integer used to write output at some steps

The code will generate some output files that contain data which could be used for analysis and visualization. Main output files are:

"Step-dt.dat" Step number, particle number, time *stepsize* used for propagation
"Step-order.dat" Step number, maximum required optimal *order* at the step
"Particle-Order-dt.dat" Step number, particle number, optimal *order*, optimal time *stepsize*
"Non-Propagated.dat" Non-propagated particles: $i, x_i, y_i, z_i, \hat{p}_{x_i}, \hat{p}_{y_i}, \hat{p}_{z_i}$
"Propagated.dat" Propagated particles: $i, x_i, y_i, z_i, \hat{p}_{x_i}, \hat{p}_{y_i}, \hat{p}_{z_i}$
"Bins.dat" Information about time-bins and particles in each of them
"NPBIN.dat" Number of particles in each bin
"Finaldistribution.dat" Final: $i, x_i, y_i, z_i, \hat{p}_{x_i}, \hat{p}_{y_i}, \hat{p}_{z_i}$