Synergia: An accelerator modeling tool with 3-D space charge

J. Amundson [a], P. Spentzouris [a,*], J. Qiang [b], R. Ryne [b]

[a] *Fermi National Accelerator Laboratory, Computing Division, CEPA/PSM, P.O. Box 500, Batavia, IL 60510, USA*
[b] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*

Abstract

High precision modeling of space-charge effects, together with accurate treatment of single-particle dynamics, is essential for designing future accelerators as well as optimizing the performance of existing machines. We describe Synergia, a high-fidelity parallel beam dynamics simulation package with fully three-dimensional space-charge capabilities and a higher order optics implementation. We describe the computational techniques, the advanced human interface, and the parallel performance obtained using large numbers of macroparticles. We also perform code benchmarks comparing to semi-analytic results and other codes. Finally, we present initial results on particle tune spread, beam halo creation, and emittance growth in the Fermilab Booster accelerator.
© 2005 Elsevier Inc. All rights reserved.

1. Introduction

In recent years, accurate modeling of beam dynamics in high-current low energy proton synchrotrons has become necessary because of new machines under consideration for future applications, such as the High Energy Physics neutrino program, and the need to optimize the performance of currently operating machines, such as the Spallation Neutron Source and the Fermilab Booster. These machines are characterized by high currents and require excellent control of beam losses, thus space-charge initiated halo formation is an essential component of their modeling. In order to obtain accurate predictions for realistic

---

conditions of operation, single-particle optics and self-consistent multi-particle effects must be combined in a single simulation code.

Several computer simulations of space-charge effects in circular accelerators using particle-in-cell techniques have been developed [1–3]. These simulations have emphasized the transverse dynamics while using a less rigorous approach for the longitudinal dynamics. Synergia [4] is a package for state-of-the-art simulation of linear and circular accelerators with a fully three-dimensional treatment of space charge, and the ability to use arbitrary order maps for the single-particle optics modeling.

Synergia is designed to be a general-purpose framework with an interface that is accessible to accelerator physicists who are not experts in simulation. Space-charge calculations are computationally intensive, typically requiring the use of parallel computers. The implementation of Synergia is fully parallel, including the particle tracking and space-charge modules. The code itself is a hybrid system based on previously developed accelerator physics codes. Synergia includes enhancements to these codes as well as new integration and interface modules. There is at least one other example of an accelerator framework which reuses existing codes [5]. Synergia distinguishes itself by being designed to provide a high level framework specifically for studying 3D multi-particle dynamics in a massively parallel computing environment.

Synergia was designed to be distributable to the particle accelerator community. Since compiling hybrid code can be a complicated task which is further complicated by the diverse set of existing parallel computing environments. Synergia includes a build system that allows it to be compiled and run on various platforms without requiring the user to modify the code and/or build system.

In this paper, we give a brief description of the components used in Synergia as well as the details involved in combining them into a single framework. We pay close attention to the build system, in keeping with the "distributable" goal mentioned above. We also describe how we have taken advantage of the Python scripting language [6] to give us a flexible human user interface with very little effort. In addition, we present a few Synergia applications. First, we compare with analytic calculations and predictions from other codes to verify the accuracy of our implementation. Then, in order to demonstrate the capabilities of the code in a realistic scenario, we present results from simulations of the Fermilab Booster [7].

2. Components

The two packages at the core of Synergia are IMPACT [10] and the *mxyzptlk/beamline* libraries [11]. We have added glue code and a human-interface wrapper to these packages, together with necessary extensions of their modules, to form the Synergia package.

2.1. IMPACT

Synergia uses IMPACT for its rf modeling and, most importantly, particle-in-cell (PIC) implementation of space-charge. IMPACT contains a suite of three-dimensional Poisson solvers that are invoked in the middle of each step of a split-operator-based model. In this model, the Hamiltonian governing single particle dynamics in an intense beam is written as

$$H = H_{ext} + H_{sc}, \tag{1}$$

where H_{ext} is the Hamiltonian in the presence of externally applied fields only (i.e. fields generated by beamline elements) and H_{sc} is the Hamiltonian which describes the effects of the space-charge fields. The electromagnetic potential associated with these fields is determined by ignoring the fine-grained texture caused by the discrete nature of the particles and treating the collective fields by a self-consistent average or mean field. We calculate the potential in the rest frame of the beam particles, where for most accelerator problems the motion of the particles with respect to each other is non-relativistic. In this case, the effect of the electric

and magnetic self-fields are both deduced from the scalar potential, which is related to the beam density, ρ, by Poisson's equation

$$\nabla^2 \phi = -\rho/\epsilon_0. \tag{2}$$

The H_{sc} term calculated with this procedure is simply proportional to the scalar potential, with a proportionality constant that varies as $1/\gamma^2$ to account for the azimuthal magnetic field associated with the longitudinal beam current. Since the scalar potential depends only on coordinates and not momenta, the effect of H_{sc} on a particle is a change in momentum, i.e. a space-charge kick, which we denote by $\mathscr{M}_{sc}$, and it is calculated using PIC techniques, as described in [10]. The effect of H_{ext} is expressed using the transfer map for the associated beamline element, $\mathscr{M}_{ext}$, which can be calculated very efficiently to arbitrary order, using Lie algebraic techniques (in the current Synergia implementation we only use up to second-order maps, see Section 2.2). Given $\mathscr{M}_{sc}$ and $\mathscr{M}_{ext}$, we use a second-order split-operator algorithm

$$\mathscr{M}(h) = \mathscr{M}_{ext}(h/2)\mathscr{M}_{sc}(h)\mathscr{M}_{ext}(h/2) + \mathcal{O}(h^3) \tag{3}$$

to propagate particles through a step in the independent variable. The above formula is accurate through second order in the step size h in approximating the effect of the full Hamiltonian, H, see [10]. The algorithm for the arbitrary order case is discussed in [12]. The problem of calculating beam propagation including space-charge effects therefore factorizes into the problem of calculating the two effects one at a time and combining them as above. A key advantage of this splitting, as opposed to one that separates the Hamiltonian into pieces involving only coordinates and only momenta, is that in our approach the rapid variation of the external fields is separated from the more slowly varying space-charge fields. A simulation step involves transport of a distribution of particles through half a step using $\mathscr{M}_{ext}$, solution of Eq. (2) using the new positions to determine $\mathscr{M}_{sc}$, application of $\mathscr{M}_{sc}$ which produces an instantaneous change in particles momenta, and finally transport through the remaining half step using $\mathscr{M}_{ext}$ and the new momenta. The choice of step size depends on the expected strength of the space-charge effects, allowing for performance optimization.

One subtle, but important, issue in the implementation of our algorithm is the use of the arc length s as the independent variable instead of the time t. This allows us to use the fully developed machinery of map-based methods for beam dynamics calculations [13]. The solution of the Poisson equation requires knowledge of the charge density at fixed time t, so during a simulation step particle coordinates are transformed from a fixed s to a fixed t and back to a fixed s representation. We implement this transformations using a *ballistic* approximation: during the transformation it is assumed that the particles move on predefined trajectories, namely straight lines with respect to the reference trajectory. The transformation between the phase-space variables (x^{in}, p_x, y^{in}, p_y, Δt, ΔE) and the fixed-time coordinates (x, y, z) is then given by

$$
\begin{aligned}
x &= x^{in} + \frac{p_x \Delta t}{(\Delta E + E_{ref})/c^2}, \\
y &= y^{in} + \frac{p_y \Delta t}{(\Delta E + E_{ref})/c^2}, \\
z &= z_{ref} + \frac{\Delta t/c}{(\Delta E + E_{ref})/c^2} \sqrt{(E + E_{ref})^2 - m^2 c^4 - c^2 p_x^2 - c^2 p_y^2},
\end{aligned}
\tag{4}
$$

where z_{ref} and E_{ref} are the longitudinal position and energy of the reference particle,[1] respectively. This version of the ballistic approximation neglects any curvature effects in the case of a circular accelerator, so it is valid for simulations of beam slices whose longitudinal extent is small compared to the radius of the accelerator.

---

[1] The reference particle is a hypothetical particle following the design trajectory of the accelerator.

We have extended the original IMPACT in several ways. IMPACT now includes an injection module, allowing multi-turn injection modeling. We have extended the beam generation module to include a six-dimensional Gaussian distribution with general correlations (see Section 3.2.4). We have also improved the memory management, allowing for an arbitrary number of beamline elements. Finally, we have enhanced the IMPACT particle data structure to allow following individual particles throughout the simulation and calculating particle tunes.

2.2. mxyzptlk/beamline libraries

The *mxyzptlk/beamline* package is a set of C++ libraries covering a wide range of accelerator physics computations. Even though the original code is over 10 years old, the libraries are written in a modern style, including real classes with encapsulation and well-considered interfaces. The package includes *basic_toolkit*, a set of useful utility classes such as Vector, Matrix, etc., *beamline*, classes for modeling elements of an accelerator beamline (the various magnets, rf cavities and other elements that make up a particle accelerator), *mxyzptlk*, classes for Lie algebraic accelerator physics calculations [13], and *physics_toolkit*, a set of classes for analysis and computation.

Synergia takes advantage of *mxyzptlk/beamline*'s arbitrary-order transfer maps, which are calculated using Lie algebraic techniques. The current Synergia implementation utilizes first- and second-order maps, but generalization to arbitrary orders is planned for the near future. Another desirable feature of the *mxyzptlk/beamline* package for our purposes is the ability to read accelerator descriptions in the MAD8 language [14]. The MAD8 parser in *beamline* is limited to processing accelerator lattice descriptions since the Synergia interface is much more flexible than the MAD8 command language. In a generic Synergia run lattice elements from MAD8 files can be combined in arbitrary ways and even mixed with native IMPACT/ Synergia elements. Note that accelerators are made of repetitive sequences of special magnet arrangements. Such a repetitive sequence is called the *lattice* and the special magnet arrangement a *lattice unit* or *lattice cell*.

3. Synergia

Synergia is the combination of IMPACT, the *mxyzptlk/beamline* libraries, glue code to get the two packages talking to each other, and a user-interface wrapper providing a straightforward, yet powerful, human interface. Fig. 1 shows the relationship between Synergia components, MAD8 files, and analysis tools.

3.1. Build system

Portability has been a major design concern in creating Synergia. We rely on multiple components written in multiple languages. While using multiple components allows us to quickly put together a powerful package, it also creates a configuration management problem. Multiple-language issues are particularly problematic because calling conventions vary from platform to platform. We solve the multiple language part of the problem by writing all of the inter-language wrapper code in terms of macros that can be redefined for various platforms. We solve configuration management problem by incorporating a modern build system based on the GNU Autotools [15] package for portable compilation to provide consistent builds on all platforms.

In principle, building Synergia is as simple as executing "`./configure && make && make install`" in the *mxyzptlk* directory followed by "`./configure && make`" in the Synergia directory. In practice, many options to configure are available. The two principles we have followed in constructing the build system are: (1) modifying the source (including Makefiles) should never be necessary, and (2) all options
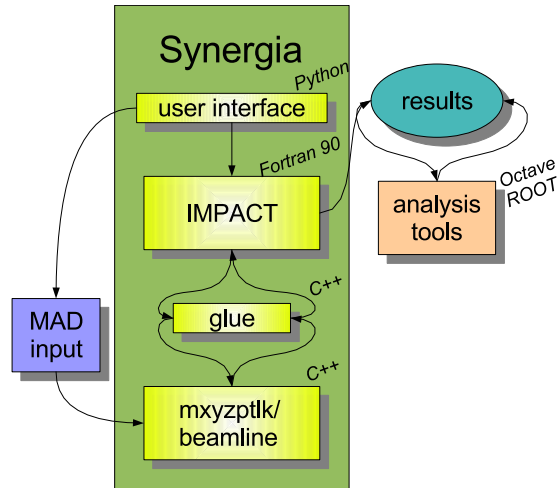
Fig. 1. Synergia components and their relation to outside inputs.

should come with reasonable defaults. To date, Synergia builds without modifications on Linux systems using either the Portland Group F90 compiler or the Intel F90 compiler, g++ or Intel CC, and either the MPICH or LAM implementations of the Message Passing Interface (MPI). Synergia also builds without modifications on AIX, using XL Fortran, Visual Age C++ and POE.

3.2. Human interface

The user-level interface to Synergia consists of a set of Python classes that wrap the low-level interfaces to the code. The Python interface generates an input file that is read by the simulation itself. The Python interpreter need not be present at run time. The Python interface can even generate a job to be automatically transferred and submitted to a remote machine where no Python interpreter is available.

To create a Synergia job, the user writes a short Python script utilizing these classes. An example script excerpt is shown in Fig. 2. The excerpt describes a FNAL Booster simulation which utilizes Synergia's matching module, the MAD parser, and demonstrates the use of simple Python syntax to run for multiple turns. The use of Python has several advantages: There is no application-specific syntax to learn. A user familiar with Python will be able to understand the entire interface. A user unfamiliar with Python will be able to copy an example script and modify it with little difficulty. Although most examples will only use Python trivially, the full power of the language is available should it be needed. Last, but not least, the use of an existing scripting language greatly simplifies the implementation, minimizing both the development time and the probability for introducing bugs.

3.2.1. Job description

Every Synergia job is a simple Python script. Synergia provides the class `Impact_parameters` as an interface to the internal parameters of IMPACT, including input beam, energy, space-charge parameters, etc. The accelerator lattice can be defined using elements from an external MAD8 file.

Synergia provides a basic matching module to generate matched beams, utilizing linear optics calculations from *mxyzptlk/beamline* for lattice function determination. We also provide an interface to our Octave utilities package that generates a matched beam in the presence of space charge by solving the rms envelope equations.

```
ip=impact_parameters.Impact_parameters()
ip.processors(4,16)
ip.space_charge_BC("transfinite,longperiodicround")
ip.input_distribution("6d gaussian")
mad_file = "booster.mad";mad_line = "booster"
(alpha_x,beta_x,alpha_y,beta_y)=\
        madcalc.twiss_initial(mad_file,mad_line)
# Set horizontal parameters based on beam width measurement
width_x = myopts.get_value("xwidth")
eps_x = width_x**2/beta_x
(width_xprime,r_x,emittance)=\
            matching.match_twiss_width(width_x,alpha_x,beta_x)
ip.x_params(sigma = width_x, lam = width_xprime * pz)
# Run for numturns turns; numturns is a command line parameter
numinjturns = 10
numturns =  myopts.get_value("numturns")
x_offset= 0; y_offset= 0; phase_offset=0
for turn in range(0,numturns):
    ip.add(impact_elements.External_element(
        kicks=96,steps=10,radius=0.04,
        mad_file_name=mad_file, beamline_name=mad_line))
    if turn< numinjturns:
        ip.add(impact_elements.Injection_element(2,ip.particles_val,
                            x_offset,y_offset,phase_offset))
my_synergia = synergia.Synergia(ip,sys.argv,synergia.options)
my_synergia.prepare_run(myopts.get_value("dirname"))
```

Fig. 2. Example excerpt of a Synergia Python script specifying a simulation with multiple injection turns utilizing basic Python. The lattice description is take from a MAD8 file; beam matching utilizes Synergia's matching module.

3.2.2. Job creation and submission

Synergia jobs can be arbitrarily complex. Typically, the user will want to run several different jobs varying only a few of the many input parameters. Synergia provides several facilities to assist the user in creating, submitting and managing simulation runs.

A Python module *options* provides a simple method to write scripts accepting command-line arguments for Synergia and user-defined parameters. When a Synergia job script is run, the command-line options for that job are automatically recorded in a manner so that they can be edited and/or reinvoked. Synergia also records all job parameters in a human-readable description file.

Synergia automatically generates batch system submission scripts based on a user-supplied template. Several example templates are provided, including templates for single-processor machines, multi-processor machines, the PBS batch system and more. Optionally, jobs can be defined to run on remote machines. Synergia generates scripts to export the input files to the remote machine, submit the job, and retrieve the files from the remote machine once the job is finished.

3.2.3. Diagnostics

A number of diagnostics are provided by default during the simulation run. In addition, we provide tools to allow users to analyze simulated data after a simulation has completed. The standard diagnostic utilities are evaluated at each split-operator step and include calculations of the second, third and fourth moments of all six degrees of freedom, two-, four-, and six-dimensional emittances, and all pairwise correlations for beam components.

For post-processing, we provide the ability to dump the entire beam, or a sampled subset of the beam, at any simulation step. Files can be dumped in plain text or HDF5 format [16]. Each particle is saved along
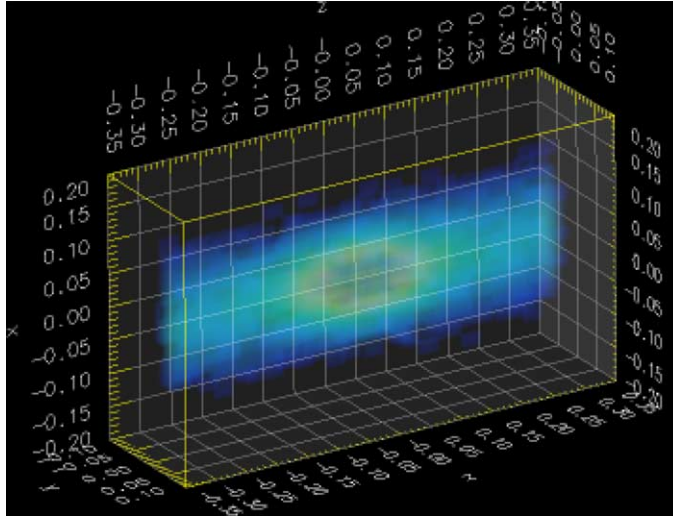
Fig. 3. OpenDX visualization of a three dimensional histogram of particle density of a FNAL Booster simulation.

with a unique tag so that individual particles can be tracked throughout the simulation. We provide tools for rearranging a series of particle dumps into individual tracks, both for diagnostic purposes and calculating particle tunes. The output format of the particle information dumps can be easily interfaced to visualization packages such as OpenDX [17]. An example of such visualization of a FNAL Booster simulation is shown in Fig. 3.

3.2.4. Distributions with general correlations

Synergia can generate Gaussian beams with arbitrary two-component correlations. Generation of random distributions with finite statistics leads to statistical errors in the moments of the generated distributions. We use, typically, $\mathcal{O}(10^6)$, macroparticles to simulate $\mathcal{O}(10^{12})$ real particles. The statistical errors in the simulation are therefore an artifact of the simulation only. Even a very small deviation in the correlation coefficients can make the difference between a matched beam and a beam that displays measurable phase-space oscillations. Some of the effects of space charge are very similar to the effects of a mismatched beam. In order to distinguish small space-charge effects from statistical fluctuations, it is advantageous to correct for the statistical fluctuations before the simulation starts. There are other cases, such as parameter scans, where running without space charge and a small number of macroparticles, say $\mathcal{O}(10^3)$, where the statistical fluctuations would be very important. We correct for these errors at the level of two-component correlations in the following procedure.

We want to generate a set of random vectors $\{r\}$ such that

$$\langle r_j \rangle = \bar{r}_j \tag{5}$$

and

$$\langle r_j r_k \rangle = C_{jk}, \tag{6}$$

where $\bar{r}_j$ and C_{jk} are given. A typical application would be a beam with no offset ($\bar{r}_j = 0$) and C_{jk} chosen to create a matched beam. We start by generating a finite set of (nominally) uncorrelated random vectors $\{\rho\}$. These vectors will have first and second moments

$$\bar{\rho}_j \equiv \langle \rho_j \rangle \tag{7}$$

and

$$X_{jk} \equiv \langle \rho_j \rho_k \rangle. \tag{8}$$

In the limit of infinite statistics, $X_{jk} \to \delta_{jk}$. Finite effects cause deviations from this limit, which will introduce small, but unphysical, deviations from the desired distribution. In order to correct for these effects, we use the transformation

$$r_j = A_{jk}(\rho_k - \bar{\rho}_k) + \bar{r}_j, \tag{9}$$

where

$$C = GG^{\mathrm{T}}, \tag{10}$$

$$X = HH^{\mathrm{T}} \tag{11}$$

and

$$A = GH^{-1}. \tag{12}$$

The resulting set of vectors $\{r\}$ will have the correct first and second moments, with no error contribution due to finite statistics.

3.3. Multi-turn injection

Synergia provides an injection module to allow modeling of multi-turn injection in a completely transparent manner. In multi-turn injection, particles are injected over a time period longer than it takes for the beam to travel around the ring. Newly injected particles have to be merged with the particles that have already propogated one or more times around the machine. Synergia models this process via an injection module that generates additional macro-particles according to a given distribution. The total beam current represented by these macro-particles is a parameter. An injection "element" can be placed anywhere between beamline elements in a given lattice, thus multi-turn injection can be simulated by simply including such an element in a loop (see Fig. 2). In our implementation, both the number of macro-particles and the beam current increase with any subsequent use of the injection module. To allow for injection painting, the injection module includes in its argument list vertical, horizontal, and longitudinal phase offsets for the center of the beam distribution.

4. Parallel performance

We have run benchmarks of our code on four different clusters under a variety of configurations. Our benchmark is a simulation of a single revolution of the FNAL Booster (see Section 5). The simulation included 2.7 million particles undergoing 100 space-charge kicks on a $65 \times 65 \times 65$ grid.

Three of the clusters are Linux clusters: lqcd [23], heimdall [24], and Alvarez [25]. Our benchmarks include a sampling of the range of currently available networking options for Linux: 100 Mbit Ethernet, Gigabit Ethernet and Myrinet 2000. We also compared the performance of the Intel fortran compiler (ifc) with the Portland Group fortran compiler (pgf90). For the former, the code was compiled with the optimization setting "-O2". For the latter, the code was compiled with the setting "-fast". The fourth cluster we used for benchmarking was Seaborg [26], the 6080-processor IBM SP at NERSC.

The propagation of particles by applying external maps is trivially parallelizable; the particles are independent of each other. In the space-charge calculation, however, each particle feels the effect of every other particle. Some global communication between processors in a parallel calculation scheme is therefore
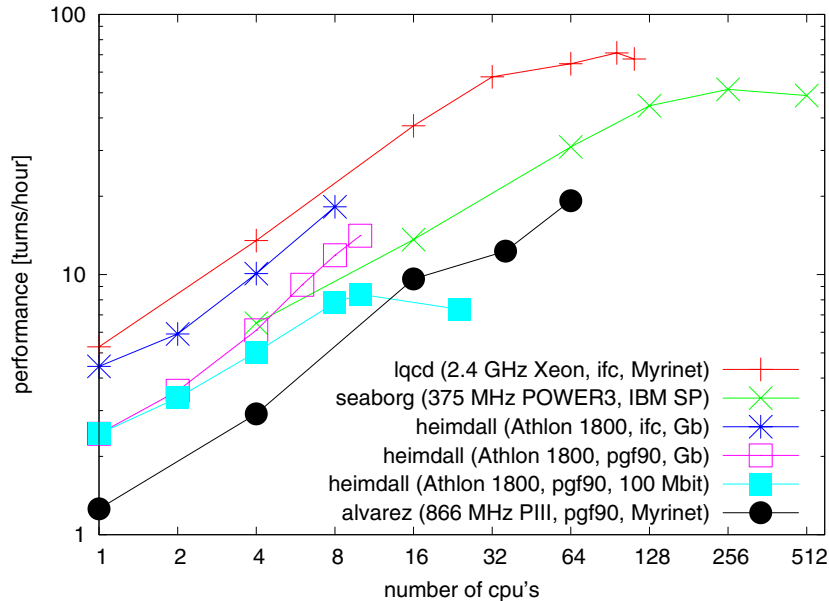
Fig. 4. Performance, defined as simulated Booster turns per hour, versus the number of processors used, on various parallel machines.

required. The scaling behavior of a parallel space-charge calculation must eventually be limited by networking performance.

The results of our benchmarks are displayed in Fig. 4. Overall, we find that Synergia scales very well up to a certain number of processors determined by the problem size and networking used in each case. The clear winner in scaling is the specialized configuration found in Seaborg. The fastest Linux clusters, however, showed overall superior performance. We can also see that Gigabit or Myrinet is necessary for a Linux cluster to effectively take advantage of more than a few processors. These tests were insufficient to distinguish between Gigabit and Myrinet.

5. Synergia tests and applications

In order to verify the accuracy of our simulation, we model several cases simple enough to perform comparisons with semi-analytic calculations. We start by comparing the evolution of a K-V beam distribution in an idealized FODO channel with the Synergia prediction. A K-V distribution is a beam distribution in the four-dimensional transverse phase-space which lies on a δ function shell. Its projections onto any transverse plane are uniform elliptical distributions with sharp boundaries [28]. A FODO channel is a periodic focusing structure composed of a sequence of focusing (F) and defocusing (D) quadrupoles separated by nonfocusing elements (O), such as a drift space. Then, we compare the Synergia prediction for the evolution of the second moments of a Gaussian beam distribution in a FODO channel to the solution of the envelope equations (13) and (14). We also compare FODO channel results from Synergia with another space-charge code. Finally, we compare the tune shifts predicted by Synergia to that of the Laslett tune shift formula [20].

The first realistic application of Synergia has been to model the FNAL Booster [7] during the first few hundred turns after injection. First, we study the incoherent tune shifts for different beam currents. Then,

we study patterns of halo formation quantitatively, as well as qualitatively. Finally, we examine emittance growth in various beam configurations.

5.1. Synergia benchmarking

For a K-V distribution, the charge density across the beam is constant and the forces associated with space charge vary linearly with the coordinates x and y. The evolution of the beam envelope can be calculated exactly by integrating the envelope equations [28]. As a first check, we compare the evolution of a K-V beam as predicted by Synergia to the solution of the envelope equations:

$$\sigma_x'' + K_x \sigma_x - \frac{\epsilon_{rms}^2}{\sigma_x^2} = \frac{\xi}{4(\sigma_x + \sigma_y)} \tag{13}$$

and

$$\sigma_y'' + K_y \sigma_y - \frac{\epsilon_{rms}^2}{\sigma_y^2} = \frac{\xi}{4(\sigma_x + \sigma_y)}, \tag{14}$$

where $\xi = 4Q^2 r_0 \lambda/(A\beta^2\gamma^3)$, with Q is the charge of a beam particle in units of e, r_0 is the classical proton radius, λ is the line charge density, A is the atomic number, $K_{x/y}$ are the focusing strengths, $\sigma_x = \langle x^2 \rangle^{1/2}$, $\sigma_y = \langle y^2 \rangle^{1/2}$, and ϵ_{rms} is the unnormalized rms emittance (a discussion on different emittance definitions and beam phase space can be found in [30]):

$$\epsilon_{rms} = \langle x^2 \rangle \langle x'^2 \rangle - \langle xx' \rangle^2 = \frac{\langle x^2 \rangle}{\beta_{Twiss}}. \tag{15}$$

Note that the rms value of x in a K-V beam of radius a is given by $\langle x^2 \rangle = a^2/4$.

In Fig. 5, we compare the numerical solution of Eqs. (13) and (14) to the Synergia result for the FODO channel defined by the following MAD8 [14] file:

```
drs: drift, l = 7.44d-2
drl: drift, l = 14.88d-2
qd7: quadrupole, l = 6.10d-2, kl = −103.11d0
qf7: quadrupole, l = 6.10d-2, kl = 103.11d0
channel: line = (drs, qd7, drl, qf7, drs)
```
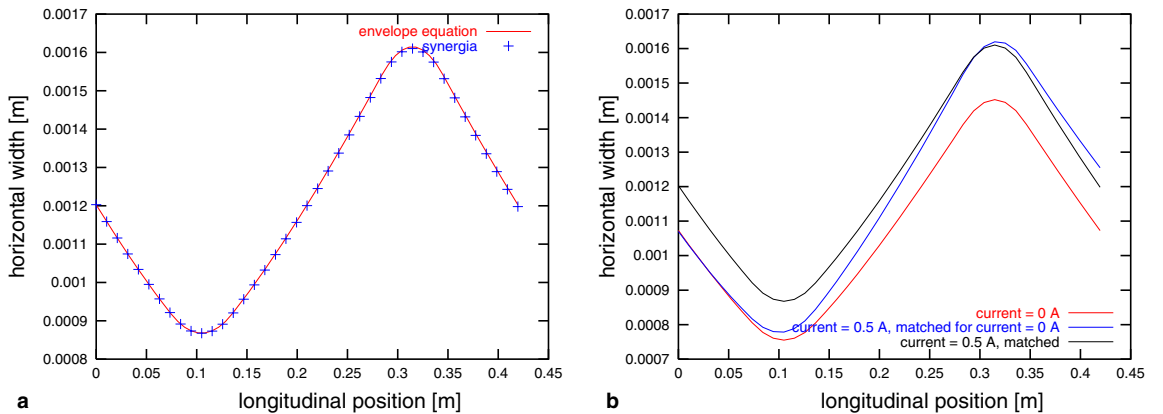

Fig. 5. (a) Comparison of the Synergia prediction for the evolution of an 0.5 A beam in the FODO lattice described in the text to the solution of the envelope equations. (b) Effect of including space charge in the matching condition, as calculated using Synergia.

The file describes a channel consisting of an empty tube, i.e., drift, (drs) followed by a quadrupole magnet (qd7), another drift (drl), another quadrupole (qf7) and a copy of the initial drift (drs). The lengths in meters are given by the l parameter. The parameter $k1$ describes the magnetic field gradient in units of meters^{-2} according to

$$k_1 = \frac{1}{(B\rho)} \frac{\partial B_y}{\partial x}, \tag{16}$$

where $B\rho$ is the ratio of the particle momentum to its charge.

For this comparison, we used a K-V beam with a kinetic energy 0.0067 GeV and two-dimensional transverse emittance 3.1×10^{-6} m rad in both the horizontal and vertical planes. Fig. 5(a) shows the comparison of the calculated horizontal beam width for a matched beam of 0.5 A. Fig. 5(b) shows the effects of taking into account space charge in the matching procedure in the evolution of the horizontal beam width. The Synergia prediction is consistent with the numerical solution of the envelope equations. The differences between the curves in Fig. 5(b) are a measure of the magnitude of the space-charge effect.

In the case of a more realistic beam distribution, such as a Gaussian distribution, the envelope equations can model the evolution of the second moments of the beam distribution under the assumption that the emittance evolution is known [28]. In the cases presented here we assume that the emittance remains constant. We compare the prediction of Synergia with the prediction of the envelope equations for the evolution of the width of a Gaussian beam in a lattice cell of the FNAL Booster [7]. Here, we use a beam that is Gaussian in the transverse coordinates and uniform in the longitudinal coordinate. The results are shown in Fig. 6(a). In Fig. 6(b), we show the effects of including space charge in the matching condition, as predicted by Synergia. The current used in this simulation is a typical operating current for the machine. In this case, the space-charge effect is small for the rms width change in one Booster cell. Traversing a single cell is a tiny fraction of the entire cycle in which the beam passes through 480,000 cells.

It is also important to cross-check the results from Synergia with other space-charge simulations. A benchmarking exercise comparing several codes, including Synergia, appears in [8]. As a simpler test, we include a consistency test comparing Synergia with the MaryLie/IMPACT (ML/I) code [9]. The comparison is done for two cases:
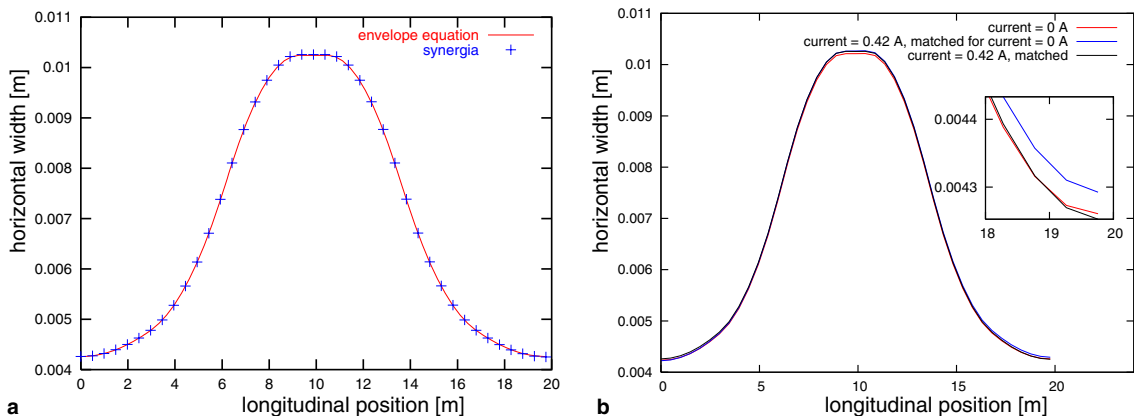

Fig. 6. (a) Comparison of the beam width evolution in a FNAL Booster cell as predicted by Synergia and the solution of the envelope equations. (b) Effects of space charge on matching, as calculated using Synergia. The insert shows the detail of these effects at the end of the cell.
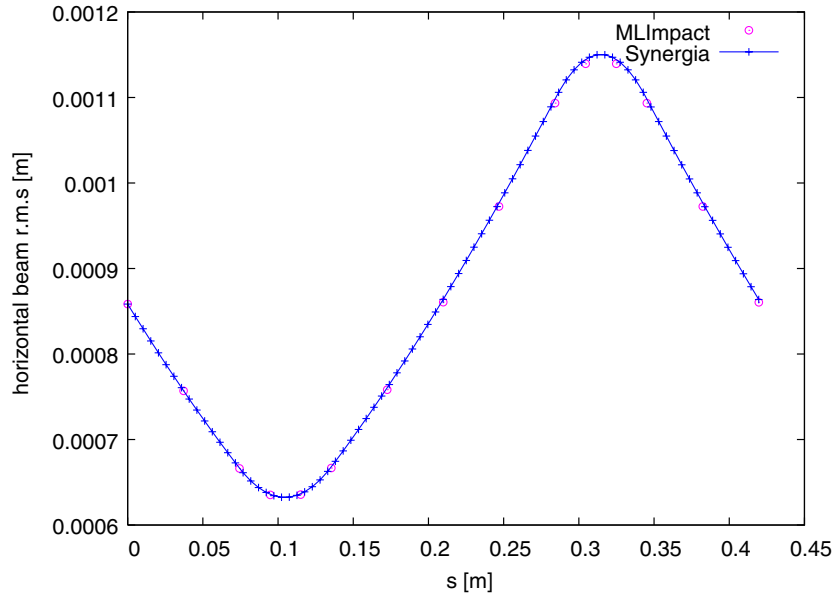
Fig. 7. Comparison of the Synergia and MaryLie/IMPACT predictions for the horizontal rms beam size of a K-V beam propagating in the FODO channel (case 1) described in the text as a function of s.

(1) The FODO channel described above, using a 0.5 A matched K-V beam with two-dimensional transverse emittance of 1.0×10^{-6} m rad in both planes.
(2) A 0.1 A cold proton beam in a FODO channel with rf cavities.

For each of these comparisons, we used a common input file of beam particles for both the Synergia and ML/I simulations. In Fig. 7, we show the comparison of the horizontal rms beam size predictions from the two codes for case 1. The agreement is very good. The difference between the prediction of the two simulations for the rms width of the beam at the end of the channel is 0.27%. This slight variation in the final answer is due to minor differences in the implementation of the Poisson solver and differences in the problem description in the simulation, such as the number of slices used in the split-operator particle advance algorithm.

In Fig. 8, we show the results for case 2. The agreement between the two codes is excellent. In this case, we model a cold, uniform density, 100 mA proton beam, with kinetic energy of 250 MeV, in a FODO channel with rf cavities. The channel consists of two 0.15 m focusing quadrupoles (fquad), with a gradient of 6 T/m, a 0.30 m defocusing quadrupole (dquad), with -6 T/m gradient, four 0.10 m drifts (dr), and two 1 m rf cavities (cav), with frequency 700 MHz. The rf cavities are treated by computing the linear transfer maps, including the effects of acceleration, and using numerical integration of the map coefficients. This requires a knowledge of the on-axis electric field and its derivative. For this example, the functional form of the field is given by $E(z) = E_0 \cos(\omega t + \phi)$. The beamline is arranged in the following way: (fquad dr cav dr dquad dr cav dr fquad). The cavity phases have been set so that the first cavity accelerates the beam and the second decelerates it by the same amount. Since the beam is cold, the rms equations describe the problem exactly, as long as the beam remains cold and uniform, so there is a matched condition where the final envelopes are identical to the initial values. We obtained the matched solution by solving the envelope
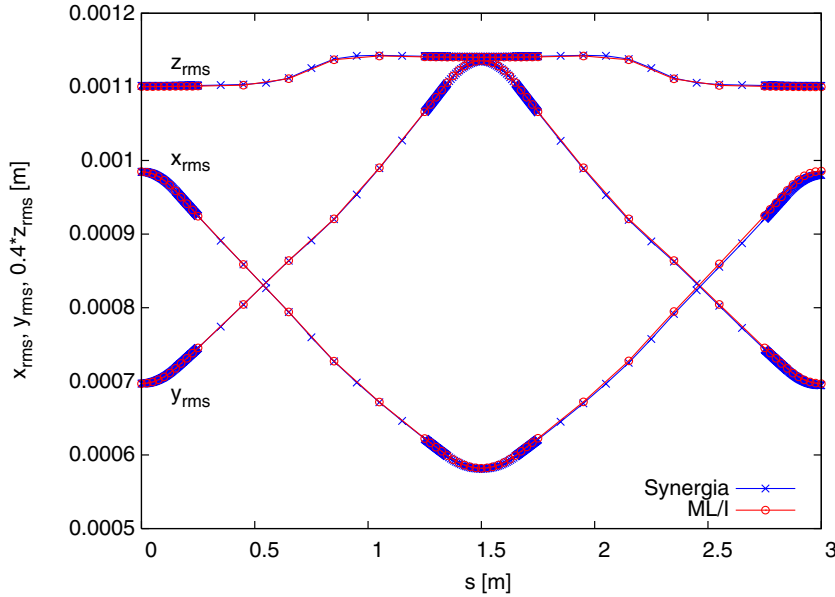
Fig. 8. Comparison of the Synergia and MaryLie/IMPACT predictions for the rms beam envelopes of a cold beam propagating in the FODO channel with rf cavities (case 2) described in the text.

equations in three dimensions [29]. The Synergia toolkit includes envelope equation solvers used to find matched beam parameters. We generated a numerical realization of the matched uniform distribution consisting of 100,000 particles. These particles were used as the input of both Synergia and ML/I.

The comparison between the Synergia and ML/I codes demonstrates that both implementations are consistent, and, most importantly, that they are in excellent agreement with the theoretical expectations for the test cases shown. In addition, the small differences in the obtained results demonstrate the level of uncertainty due to different choices of solvers and their parameters.

Another simple comparison we can make with analytic calculations is to compare the Laslett tune shift for a K-V beam with results from a Synergia simulation. We use the formula [20]

$$\Delta v = \frac{-N r_0}{8\pi \beta^2 \gamma^3 \epsilon_{\rm rms}}, \tag{17}$$

where N is the number of particles in the beam, r_0 is the classical proton radius and $\epsilon_{\rm rms}$ is the unnormalized rms emittance as defined in Eq. (15). The Synergia prediction for the tune shift is obtained by taking the peak of the Fourier transform of the horizontal and vertical position of individual particles, as a function of s, sampled each cell (24 times per turn) for 100 turns. Here, s is the coordinate along the path of the reference (or design) trajectory. By sampling each cell, we are able to extract the integer portion of the tune. Sampling once per turn is sufficient to extract the fractional tune. In Fig. 9, we show the comparison between the results from Eq. (17) and Synergia for the FNAL Booster ("bare" lattice, as described above) and for different beam currents. The agreement is very good; we thus conclude that Synergia can reliably reproduce analytical calculations of space-charge effects.

5.2. Application to the FNAL Booster

In this section, we present initial results from Synergia studies of beam behavior in the FNAL Booster during the first few hundred turns after injection.
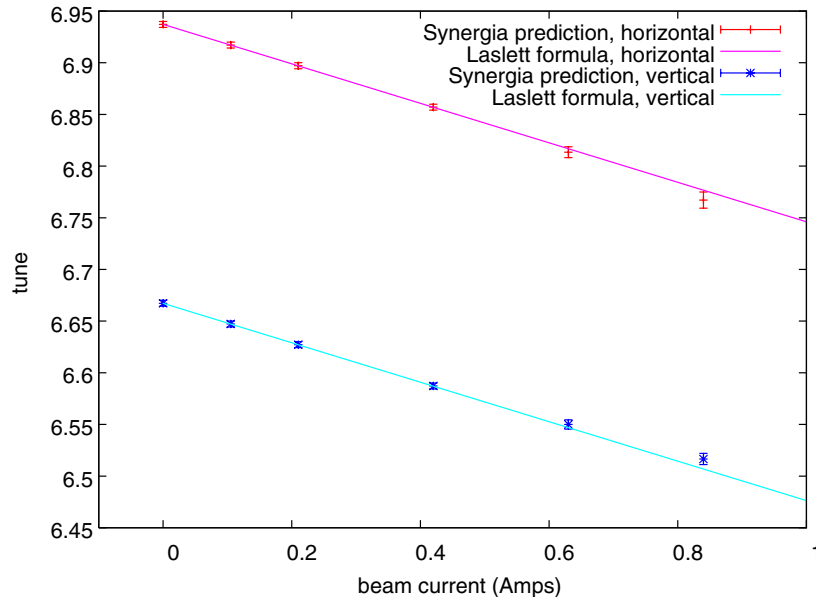
Fig. 9. Comparison of the horizontal and vertical tune shifts calculated using Synergia and using Eq. 17.

The Booster accelerator [7] is the first circular accelerator in FNAL's accelerator chain. It is a synchrotron, i.e. the field of its magnets changes with time, as the beam gets accelerated, in order to keep the beam radius constant. The Booster accelerates protons from a kinetic energy of 400 MeV to 8 GeV. It is a rapid-cycling machine, ramping the field of its magnets at 15 Hz. The Booster radius is 75.47 m and its lattice consists of 24 lattice units or cells. The main components of each cell are four combined function magnets, i.e. magnets which combine both quadrupole fields (for focusing) and dipole fields (for bending). The beam is accelerated by 17 rf cavities, with frequency that slews from 37.7 MHz at injection to 52.8 MHz at extraction. The nominal average current immediately after injection is ∼420 mA. Typically, the injection process lasts for 10 Booster turns. The beam is injected from the FNAL linear accelerator, the linac [18], and it is a stream of bunches equally spaced at the linac [18] RF frequency of 201.2 MHz.

There are many factors affecting the behavior of the Booster beam, including the energy spread and emittance of the injected beam, nonlinear field errors and space-charge effects. The space-charge effects have long been believed to be responsible for a significant fraction of the observed losses in the Booster [19] during the first 2 ms of the cycle (the injection, capture, and bunching phases). In this section we present a rudimentary study of these effects in an idealized Booster; we will examine these effects in greater detail in a subsequent paper. For all of the calculations in this paper we have used an idealized "bare" Booster lattice without any non-linear elements. We defer the inclusion of effects such as magnet offsets, correctors, etc., to a future study.

As our first example, we consider the transverse tune spread due to space charge, using a Booster lattice without nonlinear beamline elements and without the complications of multi-turn injection, but with realistic input beam parameters. The initial beam used in the simulation is a 6-dimensional Gaussian distribution, with the appropriate correlations to match it to the Booster lattice, accounting for space-charge effects. The horizontal and vertical rms emittance was 3.05×10^{-6} m rad. The full current was injected in a single turn. The single-particle optics calculations used transfer maps including both linear and quadratic terms (second-order maps). The momentum spread, $\Delta p/p$, was 0.0003. We used 96 space-charge kicks per turn, calculated on a $33 \times 33 \times 257$ computational grid with an average of four particles per grid cell. We followed the beam for 100 turns after injection, recording particle information once per space-charge kick.

In this comparison, we used currents of 0, 0.105, 0.210, 0.420, 0.630 and 0.840 Amps. As stated previously, the nominal Booster current is 0.420 Amps.

We compare the Synergia results with the results from the Laslett tune shift formula [20] for a Gaussian beam,

$$\Delta v = \frac{-N r_0}{4\pi \beta^2 \gamma^3 \epsilon_{\mathrm{rms}}}.$$
(18)

Note that the above tune shift for a Gaussian beam is a factor of two larger than the tune shift for a K-V beam in Eq. 17.

Tune shift distributions are frequently presented as scatter plots in two-dimensional transverse tune space. However, a scatter plot only gives a very qualitative picture of the distribution; the apparent shape is determined by the statistical outliers, while the internal density in the center is obscured by overlapping points. We propose the *generalized two-dimensional rms ellipse* as a quantitative measure of the spread of tune shifts in the two-dimensional transverse tune space. The ellipse is given by taking the covariance matrix

$$C = \begin{pmatrix} \langle x^2 \rangle - \langle x \rangle^2 & \langle xy \rangle - \langle x \rangle \langle y \rangle \\ \langle xy \rangle - \langle x \rangle \langle y \rangle & \langle y^2 \rangle - \langle y \rangle^2 \end{pmatrix},$$
(19)

decomposing into

$$C = R R^{\mathrm{T}},$$
(20)

where R is lower diagonal. The ellipse is then parameterized by

$$\begin{pmatrix} x \\ y \end{pmatrix} = R \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix} + \begin{pmatrix} \langle x \rangle \\ \langle y \rangle \end{pmatrix}.$$
(21)

For the case of tune spreads, we take x to be the horizontal tune and y to be the vertical tune. The resulting rms ellipse is a model-independent, statistically robust representation of the spread of the majority of the particle tunes.

The results for the different beam currents are summarized in Fig. 10, which shows the transverse tune spread together with the corresponding generalized two-dimensional rms ellipses. The filled squares in the figure correspond to the prediction from the Laslett formula for the corresponding beam current ("nominal" refers to the tune prediction for the zero current case).

Since the Laslett formula predicts tune shifts for particles at the core of a stationary beam its prediction is an upper limit of what we observe in the self-consistent particle simulation. Particles away from the center of the distribution experience smaller space-charge forces.

As a second example, we study the formation of halo in the case of mismatched beam. In Fig. 11 we plot the kurtosis[2]

$$k \equiv \frac{\left\langle (x - \langle x \rangle)^4 \right\rangle}{\left\langle (x - \langle x \rangle)^2 \right\rangle^2} - 3$$
(22)

of the beam distribution in each transverse plane as a function of s. The beam parameters are as described above, with a beam current of 0.420 Amps. We ran four different cases, varying the initial beam conditions: matched beam with and without momentum spread for single-turn injection, and 20% mismatched beam

---

[2] There are multiple possible definitions of kurtosis. Abramowitz and Stegun [21] refer to k defined above as the "kurtosis excess."
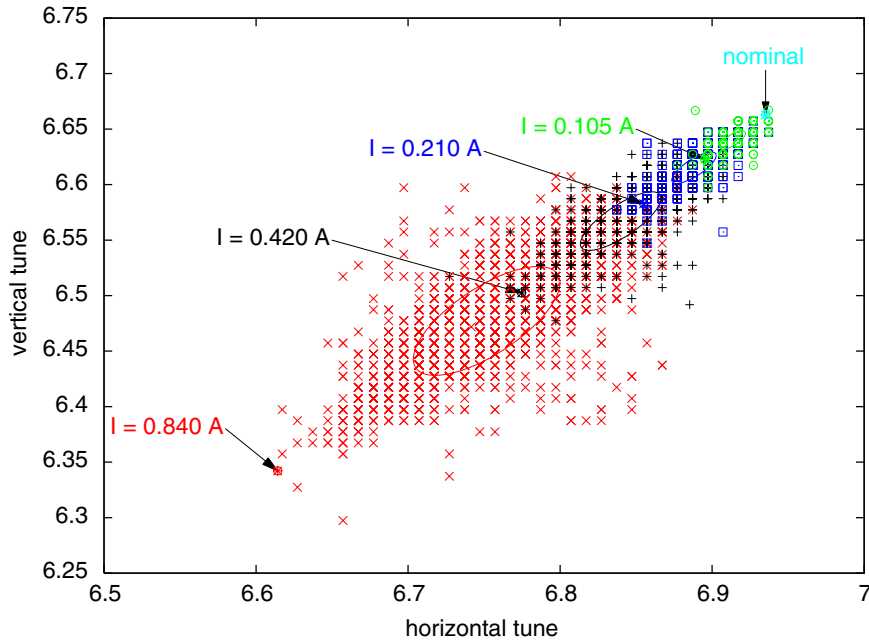
Fig. 10. Transverse tune spread calculated by Synergia, compared to the Laslett tune shift formula. The green circles correspond to a current of 0.105 A, the blue square symbols to 0.210 A, the black stars to 0.420 A and the red x's to 0.840 A. The definition of the ellipses is given in the text. The arrows point to the tune predicted by the Laslett formula for each current. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)
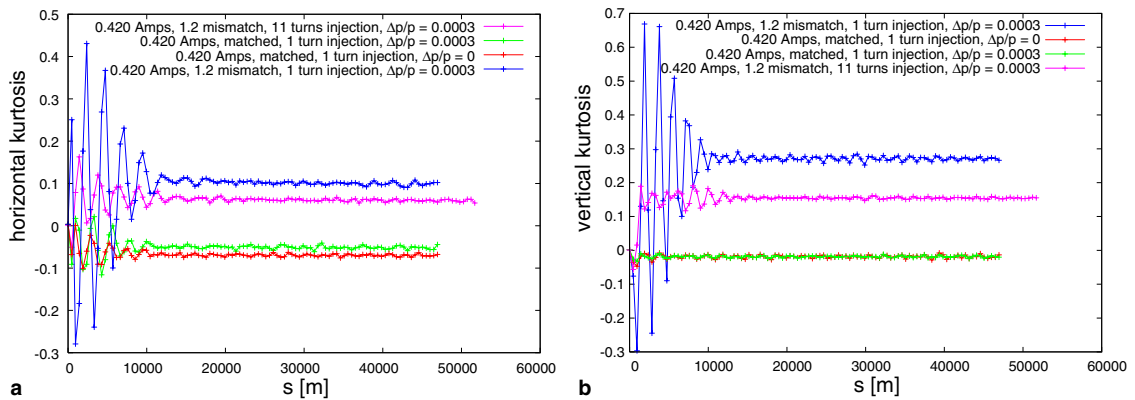

Fig. 11. Kurtosis in the horizontal (a) and vertical (b) planes, as a function of s. The beam current is 0.420 A, and the beam is matched with $\Delta p/p = 0$ (red), matched with $\Delta p/p = 0.0003$ (green), and mismatched with a 20% mismatch in both the horizontal and vertical widths, $\Delta p/p = 0.0003$, and for single turn (blue) and 11 turn (purple) injection. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

with momentum spread ($\Delta p/p = 0.0003$) for single- and multi-turn injection. For these simulations the lattice does not include any non-linear elements, but since we use second order transfer maps we expect chromatic effects to contribute to halo creation for non-zero momentum spread. In the cases with mismatch, the beam has been mismatched in both planes by stretching the width by a factor of $\mu = 1.2$ and adjusting the conjugate momentum distribution to maintain the original emittance.

We observe that in the matched beam cases, the kurtosis is close to zero and that non-zero momentum spread has a small effect in the horizontal and no effect in the vertical plane. The reason for the small difference in the horizontal is due to our matching procedure: we first match the beam correlations for the presence of dispersion, and then we match for the presence of space charge, neglecting any interaction between the two effects. This results in a small residual mismatch in the horizontal (non-zero dispersion) plane. Note that a Gaussian distribution has $k = 0$. A distribution with $k > 0$ is known as *leptokurtic*, while a distribution with $k < 0$ is known as *platykurtic*. In the case of the mismatched beam, the simulation quickly converges to a leptokurtic distribution, an indication of the halo formation. The multi-turn injection case shows a smaller increase in kurtosis than the single-turn case since the space-charge effects turn on gradually, resulting in a painting effect. We note that the authors of [22] use the same method to identify halo formation, except that they define a new parameter h, the "spatial-profile parameter." The spatial-profile parameter is related to kurtosis by $h = k + 1$.

In order to present a qualitative measure of the amount of halo created in the above simulations, we plot two dimensional phase space projections of the mismatched beam in both the transverse and longitudinal planes. Note that for the transverse phase space plots we use normalized coordinates, where x and y are scaled by $l = c/w$, with c the speed of light and w the angular frequency, and x' and y' are scaled by mc, where m is the proton mass. Figs. 12 and 13 show the evolution of the horizontal and vertical phase spaces, while Fig. 14 shows the evolution of a slice of the longitudinal phase space. The transverse phase space plots give a qualitative picture of halo formation. The longitudinal phase space plots show the transition from the bunched injected beam through debunching to a DC beam. The simulated injected beam models a realistic
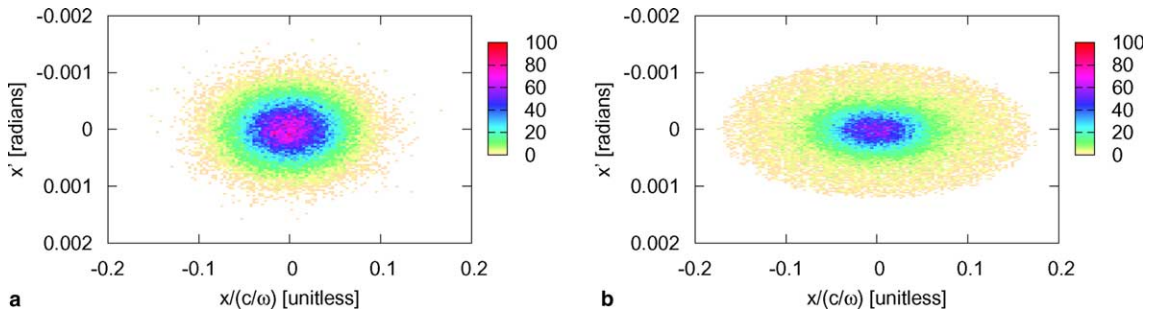

Fig. 12. Horizontal (x vs. x') phase-space plot for the mismatched beam case. (a) Beam in the beginning of the simulation, (b) after 100 turns.
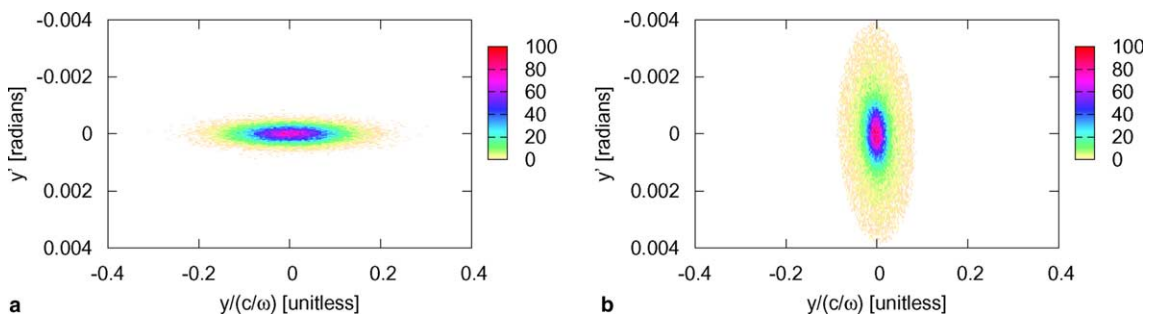

Fig. 13. Vertical (y vs. y') phase-space plot for the mismatched beam case. (a) Beam in the beginning of the simulation, (b) after 100 turns.
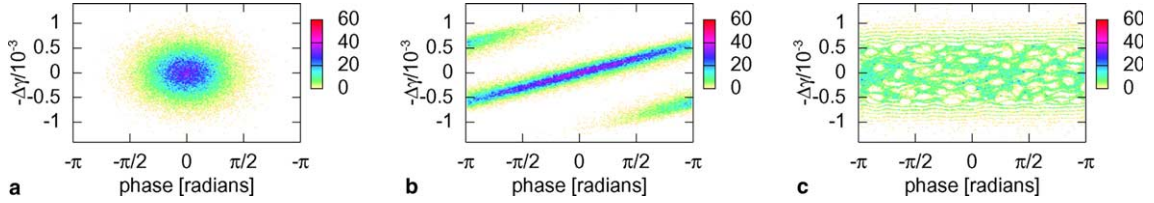
Fig. 14. Longitudinal (phase vs. $-\Delta\gamma$) phase-space plot for the mismatched beam case. (a) Beam in the beginning of the simulation, (b) after 3 turns, and (c) after 100 turns.
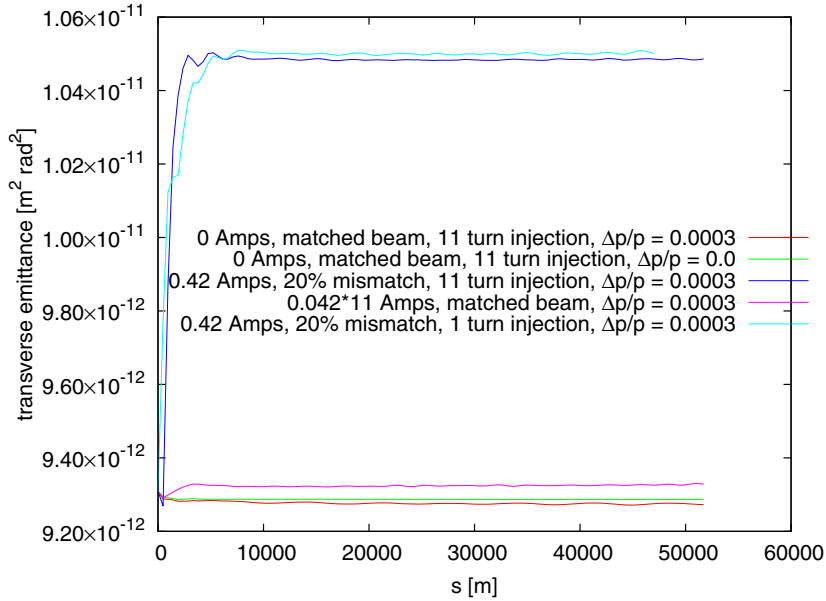

Fig. 15. Normalized 4-D transverse emittance in m^2 rad^2 for different initial conditions. The red and green curves correspond to a matched beam, with space-charge effects turned off (0 A) with and without a momentum spread of 0.0003, respectively. The purple and light blue curves correspond to a beam of 0.420 A total current and momentum spread of 0.0003, matched and mismatched respectively. Multi-turn injection of 11 turns is used in all of the above cases. The dark blue curve corresponds to a single turn injection simulation of a 0.420 A mismatched beam with 0.0003 momentum spread. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

Booster beam at injection which is bunched according to the 200 MHz Linac rf. The phase space slice in Fig. 14 corresponds to a full period of the 200 MHz rf.

Finally, we investigate how space-charge and chromatic effects affect the emittance of the Booster. In Fig. 15 we plot the normalized 4-D transverse emittance[3] for five different initial beam conditions, described in the caption of the figure. As expected, in the cases where the beam was matched there is no emittance growth. That is the case for both zero and non-zero momentum spread, and for space charge. (Our matching procedure takes into account space-charge effects on the second moments of the beam). In the mismatched cases we observe a 12% increase of the beam emittance during the first 10 to 15 turns after injection. The effect is a combination of chromatic and space-charge effects and it is very similar for both

---

[3] The 4-D emittance is the square root of the determinant of the covariance matrix of the transverse phase space.

the single- and multi-turn injection cases. The total current is the same, 0.420 Amps, in both cases. The emittance growth can be related to the conversion of beam free energy from mismatch oscillations into thermal energy of the beam, due to the effect of the non-linear space-charge forces [31]. We compare our result with the prediction of the free-energy model for the breathing mode case. In our case, where the space-charge tune shift divided by the tune is small ($\frac{\Delta v}{v} = -1.15\%$), the free-energy model prediction for emittance growth can be approximated by

$$\frac{\epsilon_f^T}{\epsilon_i^T} = 1 + 4\left[(\mu - 1)^2 - (\mu - 1)^3\right] + \mathcal{O}((\mu - 1)^4), \tag{23}$$

where μ is the mismatch parameter, and $\epsilon_{f,i}^T$ are the final (f) and initial (i) 4-D transverse emittances. With a mismatch parameter of 1.2, as in the case of our simulation, the model predicts a 4-D transverse emittance growth $\epsilon_f^T/\epsilon_i^T = 1.13$ to be compared with the 1.12 we obtained from the simulation.

6. Conclusions

In this paper we presented Synergia, a package for simulation of linear and circular accelerators with self-consistent treatment of space charge and the ability to use arbitrary order transfer maps for modeling single-particle optics. Synergia provides the tools necessary for non-expert users to easily port and use the package. The user interface takes advantage of the flexibility of Python to provide a complete and highly configurable system. The parallel implementation allows us to perform large scale simulations on modern supercomputers and clusters. We have verified the accuracy of our implementation by comparing Synergia to semi-analytic results and other codes.

In our initial application of Synergia, we studied beam behavior during the first hundred turns after injection in the Fermilab Booster. We calculated the particle tune spread and found that the majority of the particles experience a tune shift much smaller than that predicted by the Laslett formula. We also studied some mechanisms of beam halo creation and emittance growth. We used the kurtosis of the beam distribution as a quantitative measure of halo production. We found that if the injected beam is matched including space-charge effects there is no apparent emittance growth or halo creation. In the case of a mismatched beam, we found that both chromatic and space-charge effects are important in creating halo and emittance growth. A more detailed study of such effects and their dependence on initial conditions will follow in a future paper.

Acknowledgments

References

[1] F. Jones, G.H. Mackenzie, H. Schonauer, Particle Accelerators 31 (1990) 199.
[2] S. Machida, in: R. Ryne (Ed.), Computational Accelerator Physics, AIP Conf. Proc. No. 297, AIP, New York, 1994, p. 459.

 [3] J. Galambos, J. Holmes, D. Olsen, A. Luccio, J. Beebe-Wang, ORBIT User's Manual, Oak Ridge National Laboratory, SNS/ORNL/AP Technical Note No. 011, 1999.
 [4] http://cepa.fnal.gov/psm/aas/Advanced_Accelerator_Simulation.html.
 [5] N. Malitsky, R. TalmanAIP Conf. Proc. No 391, AIP, New York, 1997, p. 337.
 [6] http://www.python.org.
 [7] Booster Staff 1973, Booster Synchrotron, in: E.L. Hubbard (Ed.), *Fermi National Accelerator Laboratory Technical Memo TM-405*.
 [8] I. Hofmann et al., in: Proceedings of the PAC05 Particle Accelerator Conference, Knoxville, TN, May 16–20, 2005 (to appear). Available from: <http://www-wnt.gsi.de/ihofmann/Developer/Benchmarking/Montague.htm>.
 [9] http://scidac.nersc.gov/accelerator/mli/manual.pdf.
[10] J. Qiang, R.D. Ryne, S. Habib, V. Decyk, J. Comp. Phys. 163 (2000) 434.
[11] L. Michelotti, FERMILAB-CONF-91-159 Presented at 14th IEEE Particle Accelerator Conf., San Francisco, CA, May 6–9, 1991;
     L. Michelotti, FERMILAB-FN-535-REV;
     L. Michelotti. Published in Conference Proceedings: Automatic Differentiation of Algorithms: Theory, Implementation, and Application. Society for Industrial and Applied Mathematics. First International Workshop on Computational Differentiation. 1991;
     L. Michelotti. Published in Conference Proceedings: Advanced Beam Dynamics Workshop on Effects of Errors in Accelerators, their Diagnosis and Correction. Corpus Christi, Texas. October 3–8, 1991. American Institute of Physics: Proceedings No. 255. 1992.
[12] H. Yoshida, Phys. Lett. A 150 (1990) 262.
[13] A. Dragt, AIP Proc. 87, Phys. High Energy Accel., Fermilab, 1981, p. 147.
[14] F. Christoph Iselin, "The MAD program(Methodical Accelerator Design) Version 8.13/8", Physical Methods Manual, CERN/SL/92, 1992.
[15] G. Vaughan, B. Elliston, T. Tromey, I. Taylor (Eds.), GNU Autoconf, Automake and Libtool, Pearson Education, 2000.
[16] The Hierarchical Data Format, http://hdf.ncsa.uiuc.edu/.
[17] http://www.opendx.org.
[18] C. Ankenbrandt et al., in: Proceedings of the 11th International Conference on High-Energy Accelerators, 1980, p. 260.
[19] P. Popovic, C. Ankenbrandt, in: A.U. Luccio, W.T. Weng (Eds.), Workshop on Space Charge Physics in High Intensity Hadron Rings, AIP Conference Proceedings, Woodbury, New York, 1998, p. 128.
[20] L.J. Laslett, Proceedings of the 1963 Summer Study of Storage Rings, Accelerators and Experimentation at Super-High Energies, 1963, p. 324.
[21] M. Abramowitz, I.A. Stegun (Eds.), Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables, National Bureau of Standards Applied Mathematics Series No. 55.
[22] T.P. Wangler, K.R. Crandall, in: A.W. Chao (Ed.), Proceedings of the XX International Linac Conference, 2000, SLAC Report No. SLAC-R-561, eConf:C000821.
[23] http://lqcd.fnal.gov/.
[24] Linux cluster in the beams theory department at Fermilab.
[25] http://www.nersc.gov/alvarez/.
[26] http://hpcf.nersc.gov/computers/SP/.
[28] A.W. Chao, Physics of Collective Beam Instabilities in High Energy Accelerators, Wiley, New York, 1993; F. Sacherer, IEEE Trans. Nucl. Sci. NS-18, 1105 (1971).
[29] R. Ryne 1995, Los Alamos National Laboratory Report No. LA-UR-95-391, e-Print Archive: acc-phys/9502001.
[30] J. Buon, Beam phase space and emittance, in: S. Turner (Ed.), CERN Accelerator School – Fifth General Accelerator Physics Course, vol. I, p. 89, CERN 94 - 01, Geneva 1994.
[31] M. Reiser, Theory and Design of Charged Particle Beams, Wiley, New York, 1994, pp. 470–473;
     M. Reiser, J. Appl. Phys. 70 (1991) 1919.