Master thesis

An adaptive time integration method for more efficient simulation of particle accelerators

Matthias Toggweiler

2011

Department of Computer Science
ETH Zurich

Supervised by

Peter Arbenz
ETHZ

Andreas Adelmann
PSI

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Abstract

An important part of electromagnetic particle-in-cell (PIC) simulations is the time integration scheme. The large number of particles used in plasma simulation impose the usage of an efficient method with low memory consumption. A popular integrator is the Boris-Buneman method, also utilized in the Object Oriented Parallel Accelerator Library (OPAL) at the Paul Scherrer Institut. Until now, constant time steps are applied to carry the simulation forward in time. In this thesis we investigate whether adapting solely the global time step, used for moving all particles, can provide enhanced efficiency. We make an excursion into the world of adaptive methods and propose an adaptive variant of the Boris-Buneman method. First tests with a prototype implementation in OPAL show promising results, despite the rather simple way how the time step is adapted.

Contents

1. Introduction

The kinetic simulation of plasma using particles is a computationally intensive task. Depending on the application area, the number of real particles, like electrons, protons or ions, is too large such that we can simulate them all. Then macro-particles are used, where each of them stands for many real particles. It may be possible that the number of real particles is feasible to simulate without introducing macro-particles[1]. In either way, to accurately model a particle beam, a large number of simulation particles have to be used. Therefore, serious simulations have high memory and CPU demands and are run in parallel nowadays.

At the Paul Scherrer Institut (PSI, [1]), a library called OPAL (Object Oriented Parallel Accelerator Library, [2]) has been developed by the Accelerator Modelling and Advanced Simulations group (AMAS, [3]). OPAL is used for the simulation of existing accelerators but also in the design of the new SwissFEL project [4]. In this thesis we consider only OPAL-T, where the beam in a linear accelerator is simulated by moving particles forward in time. OPAL supports also enveloppe tracking and cyclotron simulation.

Different methods are conceivable to evolve the simulation forward in time. Since the exact movement of a particle is described as an initial value problem for a differential equation, such methods are called (time) integrators. What is common to them is that they create a discrete trajectory that approximates the solution. How they transport a given state from time t_n to t_{n+1} is cruical for accuracy and computational effort.

Currently, a time integrator that uses constant time steps $t_{n+1} - t_n = \Delta t$ is utilized. The goal of this master thesis is to investigate whether an integrator that uses variable time steps can provide enhanced efficiency. Of course, the variability in the steps must be chosen such that the resulting integrator benefits from it. Such a variability can then be called *adaptivity*. This thesis is not about any kind of spatial adaptivity. At the moment only the global time step, shared among all particles, should be adapted.

This report is organized as follows. In Section 2, more details about the simulation and the equations that need to be solved are presented. In Section 3,

[1] For an OPAL simulation like in the results section, the beam consists of more than 10^9 real particles.

the currently used integration method is explained. Section 4 describes the construction and implementation of the adaptive method. The evaluation that led to the choice among other approaches for adaptivity is included there. Results of first tests are reported in Section 5. Finally, in Section 6, we discuss the usefulness of the modified scheme and conclude with suggestions for future work.

I would like to thank Peter Arbenz, Andreas Adelmann and Christof Kraus for the mentoring during the master thesis.

2. Problem statement

2.1. High level view of the simulation procedure

The theory needed for understanding the time integration scheme is not that complex. First we do not model any collision between particles. We assume further that the beam consists of only one species of particles, e.g. only electrons. The motion of particles is affected by external electromagnetic fields and repulsive inter-particle Coulomb forces. External fields are used to accelerate and guide the beam, while the repulsive forces are a consequence of particles carrying electric charge.

Whereas the effect of external fields acts individually on each particle, the calculation of the self-induced field (in the following also called internal field) of the beam requires a computation that involves the particles as a collective. The calculation of the internal field is typically the most costly part of a time step, since it requires the invocation of a field solver, which in turn requires having particles synchronized at the same time and communication between parallel compute nodes.

Once the time integrator has obtained the values of the total electric and magnetic field for a particle, it can calculate the force (Lorentz force) that acts on the particle. Having the force and hereby the acceleration, the velocity can be updated (called "kick"). With the velocity, the position can be updated (called "push"). In which sequence these substeps are done and how often makes the difference between integrators. In summary, we have four building blocks for a time step:

- push,
- kick,
- obtain external fields,
- obtain internal field.

2.2. Governing equations

We consider N particles. Each carries the same mass m and electric charge q. The position vector of particle i is $\mathbf{x}_i \in \mathbb{R}^3$, the velocity vector $\mathbf{v}_i \in \mathbb{R}^3$. In a nonrelativistic model, the movement of a particle over time t is given as

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{v}_i, \tag{2.1}$$

$$m\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = q(\mathbf{e}_i + \mathbf{v}_i \times \mathbf{b}_i), \tag{2.2}$$

where $\mathbf{e}_i$ is the electric field at the particle's position and $\mathbf{b}_i$ is the magnetic field. The right hand side of (2.2) is called Lorentz force. The symbol $\times$ stands for the cross product and is defined as

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} v_y b_z - v_z b_y \\ v_z b_x - v_x b_z \\ v_x b_y - v_y b_x \end{pmatrix}. \tag{2.3}$$

The fields can be decomposed in external and internal contributions:

$$\mathbf{e}_i = \mathbf{e}^{\mathrm{ext}}(\mathbf{x}_i, t) + \mathbf{e}^{\mathrm{int}}(i, \mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N), \tag{2.4}$$

$$\mathbf{b}_i = \mathbf{b}^{\mathrm{ext}}(\mathbf{x}_i, t) + \mathbf{b}^{\mathrm{int}}(i, \mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N). \tag{2.5}$$

The arguments make clear that we need the position and velocity of every particle to calculate the self-field that acts on particle i, while the value of the external field is independent of other particles. Often external fields are not stationary but change over time, so we make this dependence explicit. Since a moving charge induces a magnetic field, the internal field also contributes to $\mathbf{b}_i$.

2.3. Particle-in-cell method

How do we calculate the self-induced field? Coulomb's law states that the repulsive force between two point charges is proportional to the inverse square of the distance. For equal charges q and distance r this means that each point charge feels a force with magnitude

$$f = k\frac{q^2}{r^2} \tag{2.6}$$

where k is the Coulomb constant. In other words, the electric field at position $\mathbf{x}_i$ generated from a point charge at position $\mathbf{x}_j$ is

$$\mathbf{e}_{i,j} = k\frac{q}{\|\mathbf{x}_i - \mathbf{x}_j\|^2}\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} = k\frac{q}{\|\mathbf{x}_i - \mathbf{x}_j\|^3}(\mathbf{x}_i - \mathbf{x}_j). \tag{2.7}$$

One way to simulate internal field effects would be to calculate the superposition of each particle pair Coulomb interaction. Then the internal field for particle i would read

$$\mathbf{e}^{\text{int}}(i, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{j=1, j\neq i}^{N} \mathbf{e}_{i,j}. \tag{2.8}$$

Such a solution would be inefficient since the operation count is proportional to N^2, hence making simulations with millions of particles impractical.

Although the cost of a N-body pair interaction problem could be reduced drastically from quadratic to linear by the fast multipole method [5], a different approach is taken. A concept called space charge ([6] or others) is introduced, where electric charge is understood as a continuum rather than concentrated at discrete positions. The associated method for obtaining the internal field is the particle-in-cell (PIC) method. Here we give only a rough idea, an in-depth description can be found in [6]. For an illustrative article about the method see [7].

In the PIC method, we think in continuous fields. From the charge density $\rho(\mathbf{x})$, the electrostatic potential $\phi(\mathbf{x})$ is calculated as the solution of the Poisson problem (with suitable boundary conditions),

$$\Delta\phi(\mathbf{x}) = -\frac{\rho(\mathbf{x})}{\epsilon_0}, \tag{2.9}$$

where ϵ_0 is the electric constant related to the Coulomb constant with $k = 1/(4\pi\epsilon_0)$. The electric field is then obtained as the gradient of the potential:

$$\mathbf{e}(\mathbf{x}) = -\nabla\phi(\mathbf{x}). \tag{2.10}$$

Our PIC approach introduces a rectangular grid around all particles to do these field calculations. Each particle contributes to the charge density of the corners of its containing cell. Having obtained ρ at the grid points, the Poisson problem is transformed to a discrete problem by finite differences. The solution of this equation is discussed in [8]. After the electric field is calculated at the grid points, it is interpolated to the particle positions. The most widely used

choice of scatter and interpolation, namely a linear weighting, is also called Cloud-in-cell (CIC).

The grid based PIC solution is not chosen alone to calculate the interaction field more efficiently. Birdsall and Langdon write in [6, p. 9]:

> "However, the grid provides a *smoothing effect* by not resolving spatial fluctuations that are smaller than the grid size; an exact field calculation would keep everything, which is usually more than we want."

So the grid has even a positive effect on the solution, but another numerical parameter, the grid size, has to be chosen.

What we have excluded so far is how the magnetic field, induced from moving charges, is calculated. The above explanations hold for the electrostatic case, where charges do not move. The real solver performs the calculation in a reference frame moving with the particles, such that the electrostatic case holds approximatively. Then, a Lorentz transformation back gives the electric and magnetic field one observes in the laboratory reference frame. But for the rest of this work we do not need a detailed understanding of the internals of the space charge solver. We can just think of a black-box that gives us $\mathbf{e}^{\text{int}}$ and $\mathbf{b}^{\text{int}}$ when invoked.

2.4. Quality measure

How do we judge if a new method is better than the currently used one? Roughly speaking, the requirement for a new method is that it achieves at least the same accuracy as the existing method with less computational effort. Or, the other way round, for the same computational effort it gives a better accuracy. Accuracy is measured using the error respective to a reference solution. For which testcase and at which positions this error should be measured was not defined a priori.

3. Constant step size integration method

3.1. One-step methods

The state of each particle (for a specific time) is represented as a point in a 6-dimensional Euclidean space. We have 3 dimensions for the position and 3 for the velocity. The state of the whole N-particle simulation can therefore be seen as an element of a $6N$-dimensional space. Since every possible state of the system can be represented with a single point in it, it is called *phase space*. From now on, we denote the phase space state by $\mathbf{Z}$[1]

$$\mathbf{Z} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{pmatrix}. \tag{3.1}$$

Given an initial condition $\mathbf{Z}_0$ for $t = t_0$ (and a specification of the fields), the differential equations (2.1) and (2.2) taken for $i = 1, \ldots, N$ have an unique solution function $\mathbf{Z}(t)$. Except for some special cases, an analytic solution is beyond question and we have to resort to a numerical method.

We look at methods which, given the state at time t_0, produce a sequence of approximations for $\mathbf{Z}(t)$ at discrete times $\{t_1, t_2, t_3, \ldots\}$. Let Ψ be such a method. Given a step size h and the initial condition $\mathbf{Z}_0 = \mathbf{Z}(t_0)$, the method outputs a new state

$$\mathbf{Z}_1 = \Psi^h(\mathbf{Z}_0) \tag{3.2}$$

such that it can be understood as an approximation

$$\mathbf{Z}_1 \approx \mathbf{Z}(t_0 + h) = \mathbf{Z}(t_1). \tag{3.3}$$

[1] We use small letters for 3-dimensional vectors, and capital letters for $3N$- and $6N$-dimensional vectors.

Now, out of $\mathbf{Z}_1$, the repeated application of the one-step method moves the simulation forward in time:

$$\mathbf{Z}_{n+1} = \Psi^h(\mathbf{Z}_n) \approx \mathbf{Z}(t_n + h) = \mathbf{Z}(t_{n+1}). \tag{3.4}$$

Each step can introduce a deviation from the exact solution. Since future steps are built upon previous approximations, the error can accumulate. A method is called consistent if the generated approximation curve approaches the solution as the step size h goes towards zero. Every reasonable method has this property. Depending how fast the error decreases when h is made smaller, methods are categorized with respect to different orders. A more accurate solution can be obtained in two ways, either by reducing the step size h or by using a method of higher order. Of course, higher accuracy implies more computational effort to arrive at a solution, because more steps are used or the computational cost per step is higher.

Depending on the application, purely small local error (high order) is not the only criterion if a method is the best choice in a global view. Other properties, such as energy conservation or number of force evaluations per step, may play an important role when deciding for an integrator.

3.2. Euler method

As a simple example to get used to the notation, we write down how the well-known Euler method would look for our case.

To avoid particle number subscripts, we gather the individual position and velocity vectors into two big vectors:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} \qquad \mathbf{V} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{pmatrix} \tag{3.5}$$

This makes sense since the transformations are always done for all particles before the integrators proceeds to the next substep. The field vectors are rewritten as big vectors too.

The Euler method for the differential equation

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t} = \mathbf{F}(\mathbf{Z}, t) \tag{3.6}$$

reads

$$\frac{\mathbf{Z}_{n+1} - \mathbf{Z}_n}{h} = \mathbf{F}(\mathbf{Z}_n, t_n), \tag{3.7}$$

where the subscript denotes the step number. Note that the left hand side is a finite difference approximation for the derivative. Written for our specific case (see (2.1) – (2.2)), the method is characterized by the equations

$$\frac{\mathbf{X}_{n+1} - \mathbf{X}_n}{h} = \mathbf{V}_n, \tag{3.8}$$

$$\frac{\mathbf{V}_{n+1} - \mathbf{V}_n}{h} = \frac{q}{m}(\mathbf{E}_n + \mathbf{V}_n \times \mathbf{B}_n). \tag{3.9}$$

Here the $\times$-symbol is understood as component-wise cross product. $\mathbf{E}_n$ is a shorthand for $\mathbf{E}(\mathbf{X}_n, \mathbf{V}_n, t_n)$ (analogously for $\mathbf{B}_n$). These equations can be solved without problems for $\mathbf{X}_{n+1}$ and $\mathbf{V}_{n+1}$, respectively. Algorithm 1 gives an idea how this method would be implemented in our context[2]. The actual state is used to calculate the forces, then the positions are advanced and finally the velocity is updated. Algorithm 2 sketches how the method could be used inside a simulation loop to reach a specified end time exactly[3].

Algorithm 1 Euler$(\mathbf{X}, \mathbf{V}, t, h)$

1: $\mathbf{E} \leftarrow$ QueryExternalEField$(\mathbf{X}, t)$ + CalculateSpaceChargeE$(\mathbf{X}, \mathbf{V})$
2: $\mathbf{B} \leftarrow$ QueryExternalBField$(\mathbf{X}, t)$ + CalculateSpaceChargeB$(\mathbf{X}, \mathbf{V})$
3: **for** $i = 1$ to N **do**
4: $\quad \mathbf{x}_i \leftarrow \mathbf{x}_i + h\mathbf{v}_i$
5: $\quad \mathbf{v}_i \leftarrow \mathbf{v}_i + hq(\mathbf{e}_i + \mathbf{v}_i \times \mathbf{b}_i)/m$
6: **end for**
7: $t \leftarrow t + h$
8: **return** $(\mathbf{X}, \mathbf{V}, t)$

The Euler method is an explicit method, meaning that no equations have to be solved numerically while running it. The number of force evaluations (or equivalent, field evaluations) is one. No memory is required in addition to the phase space and field vectors. This makes it a very inexpensive method. However, the method is not very accurate (of order 1) and suffers from stability problems. We can do better with only little effort.

[2] Because the external fields can have an explicit time dependence, we pass the time to the integration method too.

[3] To be consistent with the adaptive method presented later, the time t is updated by the method and not by the simulation loop.

Algorithm 2 Simulation($\mathbf{X}_0, \mathbf{V}_0, t_{\text{end}}, h$)

$\mathbf{X} \leftarrow \mathbf{X}_0$
$\mathbf{V} \leftarrow \mathbf{V}_0$
$t \leftarrow 0$
while $t < t_{\text{end}}$ **do**
  if $t_{\text{end}} - t < h$ **then**
    $h \leftarrow t_{\text{end}} - t$
  end if
  $(\mathbf{X}, \mathbf{V}, t) \leftarrow$Euler($\mathbf{X}, \mathbf{V}, t, h$)
  WriteStatistics($t, \mathbf{X}, \mathbf{V}$)
end while
return $(\mathbf{X}, \mathbf{V})$

3.3. Boris-Buneman method

3.3.1. Derivation

The method that is used for time integration in OPAL is the Boris-Buneman method, that will be explained now. Often it is just called the Boris method. This is a summary of a more detailed presentation in [6, pages 58 – 63].

A downside of the Euler method is its asymmetry. The update step is derived from using a forward difference for the derivative at time t_n. An intuitively better choice seems to be a centered difference. The leapfrog integration achieves this by offsetting position and velocity by half a time step. One quantity is updated a full step while the other quantity is "in the middle":

$$\frac{\mathbf{X}_{n+1/2} - \mathbf{X}_{n-1/2}}{h} = \mathbf{F}_X(\mathbf{V}_n, t_n), \tag{3.10}$$

$$\frac{\mathbf{V}_{n+1} - \mathbf{V}_n}{h} = \mathbf{F}_V(\mathbf{X}_{n+1/2}, t_{n+1/2}). \tag{3.11}$$

The leapfrog method is as cheap as the Euler method, but more accurate (has order 2 instead of 1). Symplecticity and greater stability make it superior to the Euler method. It can be implemented in an explicit way, too. For an in-depth discussion of properties of one-step methods, see for example [9].

However, in our case we have a problem. The velocity derivative $\mathbf{F}_V$ depends on the velocity itself if a magnetic field is present (see (3.9)), so we would have to write $\mathbf{F}_V(\mathbf{X}_{n+1/2}, \mathbf{V}_{n+1/2}, t_{n+1/2})$ in (3.11) instead. If we introduce a new unknown $\mathbf{V}_{n+1/2}$, we cannot calculate $\mathbf{V}_{n+1}$ out of known values. Using $\mathbf{F}_V(\mathbf{X}_{n+1/2}, \mathbf{V}_n, t_{n+1/2})$ is a possibility, but then the symmetry is lost. The idea

to maintain centering by using $\mathbf{F}_V(\mathbf{X}_{n+1/2}, (\mathbf{V}_n+\mathbf{V}_{n+1})/2, t_{n+1/2})$ is attributed to Buneman in [6]. The discretized equations become now

$$\frac{\mathbf{X}_{n+1/2} - \mathbf{X}_{n-1/2}}{h} = \mathbf{V}_n, \tag{3.12}$$

$$\frac{\mathbf{V}_{n+1} - \mathbf{V}_n}{h} = \frac{q}{m}\left(\mathbf{E}_{n+1/2} + \frac{\mathbf{V}_n + \mathbf{V}_{n+1}}{2} \times \mathbf{B}_{n+1/2}\right). \tag{3.13}$$

Assuming we are given the field values at half steps $(\mathbf{E}_{n+1/2}, \mathbf{B}_{n+1/2})$, the method is perfectly symmetric[4].

Equation (3.13) is an implicit definition for the velocity at the new time step. But it can be solved for $\mathbf{V}_{n+1}$ analytically, permitting an *explicit* implementation. How it is solved in an elegant way is due to Boris.

First, two auxiliary variables are introduced (switching to the single particle vectors),

$$\mathbf{v}_* = \mathbf{v}_n + \frac{h}{2}\frac{q}{m}\mathbf{e}, \tag{3.14}$$

$$\mathbf{v}_{**} = \mathbf{v}_{n+1} - \frac{h}{2}\frac{q}{m}\mathbf{e}, \tag{3.15}$$

which absorb the electric field of the implicit equation (3.13), giving

$$\frac{\mathbf{v}_{**} - \mathbf{v}_*}{h} = \frac{q}{m}\frac{\mathbf{v}_* + \mathbf{v}_{**}}{2} \times \mathbf{b}. \tag{3.16}$$

This system of 3 scalar equations can be solved purely analytically, but a geometric interpretation gives a solution already simplified to few arithmetic operations. By observing that this is a rotation in a plane perpendicular to $\mathbf{b}$ which leaves the length of the velocity vector invariant, Boris derived that the following steps determine $\mathbf{v}_{**}$ that satisfies (3.16):

$$\mathbf{t} = \frac{hq}{2m}\mathbf{b}, \tag{3.17}$$

$$\mathbf{w} = \mathbf{v}_* + \mathbf{v}_* \times \mathbf{t}, \tag{3.18}$$

$$\mathbf{s} = \frac{2}{1 + \|\mathbf{t}\|^2}\mathbf{t}, \tag{3.19}$$

$$\mathbf{v}_{**} = \mathbf{v}_* + \mathbf{w} \times \mathbf{s}. \tag{3.20}$$

[4] In fully electromagnetic codes, the electric and magnetic fields are usually stored on staggered grids, offset by half a time step of each other. Similar to the averaging of velocity, one field has to be averaged to get a time-centered value. For more information see [6]. In our quasi-electrostatic application, the method is not strictly symmetric since we use $\mathbf{E}_{n+1/2} = \mathbf{E}(\mathbf{X}_{n+1/2}, \mathbf{V}_n, t_{n+1/2})$ and $\mathbf{B}_{n+1/2} = \mathbf{B}(\mathbf{X}_{n+1/2}, \mathbf{V}_n, t_{n+1/2})$. The mean (longitudinal) particle velocity used for doing the Lorentz transformation in the internal field calculation is not time-centered.

The implementation is sketched in Algorithm 3. Equation (3.12) is split between a step and the previous step such that positions are defined at integral times and not only at half times.

Algorithm 3 BorisBuneman($\mathbf{X}, \mathbf{V}, t, h$)

$(\mathbf{X}, t) \leftarrow \text{BBPush}(\mathbf{X}, \mathbf{V}, t, h/2)$
$\mathbf{E} \leftarrow \text{QueryExternalEField}(\mathbf{X}, t) + \text{CalculateSpaceChargeE}(\mathbf{X}, \mathbf{V})$
$\mathbf{B} \leftarrow \text{QueryExternalBField}(\mathbf{X}, t) + \text{CalculateSpaceChargeB}(\mathbf{X}, \mathbf{V})$
$\mathbf{V} \leftarrow \text{BBKick}(\mathbf{V}, \mathbf{E}, \mathbf{B}, h)$
$(\mathbf{X}, t) \leftarrow \text{BBPush}(\mathbf{X}, \mathbf{V}, t, h/2)$
return $(\mathbf{X}, \mathbf{V}, t)$

Algorithm 4 BBPush($\mathbf{X}, \mathbf{V}, t, h$)

for $i = 1$ to N **do**
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + h\mathbf{v}_i$
end for
$t \leftarrow t + h$
return $(\mathbf{X}, t)$

Algorithm 5 BBKick($\mathbf{V}, \mathbf{E}, \mathbf{B}, h$)

for $i = 1$ to N **do**
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + hq\mathbf{e}_i/(2m)$
    $\mathbf{t} \leftarrow hq\mathbf{b}_i/(2m)$
    $\mathbf{w} \leftarrow \mathbf{v}_i + \mathbf{v}_i \times \mathbf{t}$
    $\mathbf{s} \leftarrow 2\mathbf{t}/(1 + \mathbf{t}^{\mathrm{T}}\mathbf{t})$
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \mathbf{w} \times \mathbf{s}$
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + hq\mathbf{e}_i/(2m)$
end for
return $\mathbf{V}$

The Boris-Buneman method is popular in PIC codes due to its simple (explicit) implementation. The method is of order 2 while requiring only one field evaluation. Good energy behaviour is also attributed to it, see e.g. [10]. The most important point, however, is the low memory requirement. While other methods of higher order may need fewer steps to gain the same accuracy, they need more vectors to store intermediate values, which is a serious drawback when dealing with a large number of particles.

3.3.2. Relativistic generalization

The speeds of particles compared to the speed of light c are not neglible and make a relativistic treatment of the problem necessary. Fortunately, the method of Boris is easy to adapt. The relativistic variant is described in [6, pages 356 – 357], but can be simply derived from the nonrelativistic version by replacing the mass m by γm_0, where γ is the Lorentz factor and m_0 the rest mass of the particle.

OPAL uses as momentum variable

$$\mathbf{p} = \gamma \frac{\mathbf{v}}{c} = \gamma \boldsymbol{\beta} \tag{3.21}$$

instead of the plain velocity. With this choice, the Lorentz factor is calculated as

$$\gamma = \sqrt{1 + \|\mathbf{p}\|^2}. \tag{3.22}$$

Of course the transformed units have to be taken into account when writing down the relativistic variant, but no other changes in the integrator are required.

4. Adaptive step size integration method

4.1. Conventional step size control

This explanation is based on [11, Chapter VIII, Introduction]. Conventional adaptivity changes the time step in a way that an estimate for the local error is below a tolerance e_{tol}. The local error is assumed to be proportional to h^r, where r depends on the order of the method. Based on the estimate e_n for the current local error, a new step size is predicted by a formula like

$$h_{n+1} = 0.85 h_n \left(\frac{e_{\text{tol}}}{e_n} \right)^{1/r}, \tag{4.1}$$

where 0.85 is a safety factor. Then, a step with h_{n+1} is applied and accepted if $e_{n+1} \leq e_{\text{tol}}$ and repeated otherwise with a smaller step size obtained from using e_{n+1} in (4.1). If we have the exact solution the error estimate is trivial to obtain, but this is rarely the case in real applications. In general, the difference between two solutions of different accuracy is used to arrive at an error estimate. One choice is to compare the result of a big step h with the result of two fine steps $h/2$. However, often one uses a method with order k which at the same time (with the same force evaluations) contains an *embedded* method of lower order $k' < k$.

4.2. Adaptive geometric integration

Geometric integration has the goal to reflect certain properties of the physical system within the numerical method. For example, if the total energy in a system is conserved, one tries to construct an integration scheme that also conserves the total energy. Methods that have purely small local error are not necessarily superior to geometric integrators when looking at the simulation for a longer series of steps. A more profound introduction can be found in [9] or in [11].

One import class consists of systems which have a reversibility property. In the simplest case (for conservative mechanical systems) this means that the

solution curve obtained when inverting the direction of the velocity vector is the same, just traversed in the opposite direction. In other words, if $(\mathbf{x}(t+h), \mathbf{v}(t+h))$ is a point of the solution with initial conditions $(\mathbf{x}(t), \mathbf{v}(t))$, then the solution obtained from the initial conditions $(\mathbf{x}(t+h), -\mathbf{v}(t+h))$ goes again through $(\mathbf{x}(t), \mathbf{v}(t))$. Our system (2.1) – (2.2) has a similar reversibility property, but in addition to inverting the velocity vector the magnetic field vector must be inverted too. A one-step method which satisfies

$$\Psi^{-h}(\Psi^h(y_n)) = y_n \tag{4.2}$$

is called symmetric or time-reversible. Such one-step methods reproduce the reversibility of the system within the numerical flow. One has observed that respecting the reversibility leads to surprisingly good long-time behaviour. For the theory, see [11].

The concept of adaptive geometric integration is to look at a transformed system that evolves in a fictitious time τ. The relation that connects the real time to it is called a Sundman transformation:

$$\frac{\mathrm{d}t}{\mathrm{d}\tau} = g(\mathbf{Z}). \tag{4.3}$$

The function g determines the time rescaling and depends on the state of the system. It has to be chosen depending on the problem. The idea is that the rescaling stretches the time axis where the dynamics require a higher step resolution and compresses it where larger steps can be taken. When g is large, $t(\tau)$ increases rapidly and we take large steps in real time. When g is small, $t(\tau)$ increases slowly and we take small steps.

Where the evolution of the system over t was described with

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t} = \mathbf{F}(\mathbf{Z}, t), \tag{4.4}$$

the transformed equation with independent variable τ becomes, by the chain rule of differentiation,

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}\tau} = \frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t}g(\mathbf{Z}) = \mathbf{F}(\mathbf{Z}, t)g(\mathbf{Z}). \tag{4.5}$$

The curve $\mathbf{Z}(\tau)$ is identical to $\mathbf{Z}(t)$, but the trajectories are traversed at different speeds with respect to the independent variable. Constant steps taken in the fictitious time τ correspond to variable steps in the real time t, excluding the uninteresting case of a constant g. Multiple methods exist to solve such a transformed system. It is not straightforward since a separable system[1] like (3.10) - (3.11) can become nonseparable.

[1] Separable means that the position derivative depends only on the velocity, and the velocity derivative only on the position.

4.3. Evaluation of alternatives

This subsection makes a detour explaining how the author arrived at the final choice of a method, but is not essential for understanding the rest of the thesis.

4.3.1. Voting out of conventional step size control

Conventional step size control was not considered further in this work, since it was unclear how to arrive reasonably at an error estimate. Using an embedded method for this would have certainly led to an increased memory demand of the integrator. Doing a step with h and $h/2$ and then deciding for a step size did not seem like a cheap solution either. And then there were these countless warnings in geometric integration literature that conventional adaptivity destroys the good behaviour of constant step size variants, see e.g. [11, Chapter 8, Introduction]:

> "In the previous chapters we have studied symmetric and symplectic integrators, and we have seen an enormous progress in long-time integrations of various problems. Decades ago, a similar enormous progress was the introduction of algorithms with automatic step size control. Naively, one would expect that the blind combination of both techniques leads to even better performances. We shall see by a numerical experiment that this is not the case, a phenomenon observed by Gladman, Duncan & Candy (1991) and Calvo & Sanz-Serna (1992)."

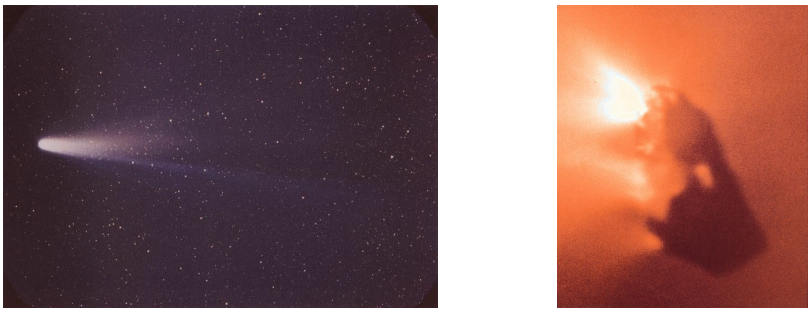
In retrospect, the author of this master thesis maybe took such statements too serious. Often, in such literature few-body problems over a long time interval are studied which is not exactly the case what we do in OPAL. A comparison with a "traditional" approach to adaptivity would have been certainly interesting.

4.3.2. Kepler test case

Three different attempts to adaptive geometric integration were compared on a standard test problem, the Kepler problem. To make it a little less abstract, suppose we want to integrate the motion of Halley's comet (Figure 4.1, [12]) which returns to the inner solar system every 75 – 76 years. It possesses, in good approximation, an elliptical orbit with a high eccentricity of $e \approx 0.967$ and a semi-major axis of $a \approx 17.8$ AU[2], where the sun is at one of the foci

[2] Astronomical unit, average distance Earth-Sun

Figure 4.1.: Photo of Halley's comet taken from earth (left) and from spacecraft (right), during close encounter in 1986. Source: [13]



of the ellipse. Although this case has an analytical solution (the ellipse) and integrators specially tuned to deal with such problems exist, it serves as a showcase for general purpose adaptive geometric integration.

We model the problem in two dimensions, in the plane of the orbit. The reference frame is chosen such that the sun is at position $(0,0)^{\mathrm{T}}$. Since the comet mass m is much smaller than the sun mass M, the center of mass of the two-body system, which defines the focus point of the orbit ellipse, stays practically at the sun. In other words, we can neglect the acceleration that acts on the sun. The acceleration that pulls the coment with position $\mathbf{x} = (x,y)^{\mathrm{T}}$ towards the origin is

$$\mathbf{a}(\mathbf{x}) = -\frac{GM}{\|\mathbf{x}\|^2}\frac{\mathbf{x}}{\|\mathbf{x}\|} = -\frac{GM}{(x^2+y^2)^{3/2}}\mathbf{x} \tag{4.6}$$

where G is the gravitational constant. Thus the system of equations we want to solve is

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}, \tag{4.7}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \mathbf{a}(\mathbf{x}). \tag{4.8}$$

As initial condition the comet is chosen to be in the pericentre with

$$\mathbf{x}_0 = (a(1-e),0)^{\mathrm{T}}, \tag{4.9}$$

$$\mathbf{v}_0 = \left(0, \sqrt{\frac{GM(1+e)}{a(1-e)}}\right)^{\mathrm{T}}. \tag{4.10}$$

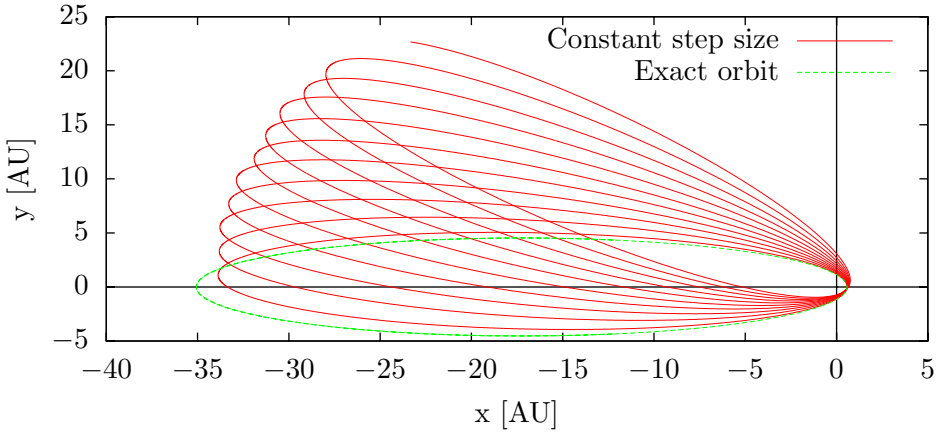
This leads to an elliptical curve with semi-major axis a and eccentricity e.

Now, the goal was to integrate 10 periods of the orbit, which is a time span of roughly 750 years. Kepler's third law lets us calculate the period with

$$T = 2\pi\sqrt{\frac{a^3}{GM}}. \tag{4.11}$$

The constant step size leapfrog method (like in (3.10) – (3.11)), used with $20'000$ time steps (one step is a bit less than half a month), has serious problems to reproduce the correct solution, see Figure 4.2. At the end time $10T$, the comet should be again in the pericentre, but is far away from it with this method. However, the elliptical shape of the orbit is preserved.

Figure 4.2.: Constant step size integration fails to reproduce orbit



The adaptive strategy that leads to success in this case is to use a time transformation (see (4.3)) with

$$g(\mathbf{x}) = \|\mathbf{x}\|^{3/2}. \tag{4.12}$$

The variable time step

$$t_{n+1} - t_n \approx \int_{\tau_n}^{\tau_n+\Delta\tau} g(\mathbf{x}(\tau))\mathrm{d}\tau \tag{4.13}$$

is at its lowest value when the comet is at the point of closest approach and at its highest value when the comet is the farthest away[3]. The particular choice

[3]The approximation sign in (4.13) is used because methods reconstruct original time not exactly.

of the exponent 3/2 can be justified by a scaling invariance property of the system, see [9, Section 9.3.2] or [14].

The transformed system (see (4.5)) is for this case

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = \mathbf{v}g(\mathbf{x}), \tag{4.14}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\tau} = \mathbf{a}(\mathbf{x})g(\mathbf{x}), \tag{4.15}$$

where the coupling in the right hand side of (4.14) does not allow a straightforward symmetric implementation with the standard leapfrog method. One solution is to look at implicit methods, but for this evaluation only explicit methods were considered. The essence is presented below, for details consult the respective papers ([15], [16], [17]).

A symmetric method for adaptive step size integration was published 2005 [15]. Instead of the time scaling g the inverse concept of a step density objective $Q = 1/g$ is used. The variable ρ, the current step density, is updated along with the variables of the original system, in our case $\mathbf{x}$ and $\mathbf{v}$. A differential equation defines the change of the step density with a control function G

$$\frac{\mathrm{d}\rho}{\mathrm{d}\tau} = G(\mathbf{x}, \mathbf{v}), \tag{4.16}$$

where the control function is calculated from the derivative of the step density objective along the solution, which gives in our case

$$G(\mathbf{x}, \mathbf{v}) = \frac{\nabla Q(\mathbf{x})^{\mathrm{T}}\mathbf{v}}{Q(\mathbf{x})} = -\frac{3}{2}\frac{\mathbf{v}^{\mathrm{T}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}. \tag{4.17}$$

The method is generic as any symmetric one-step method Ψ for the original system (in our case leapfrog) can now be used to give an adaptive variant which is characterized by

$$\frac{\rho_{n+1/2} - \rho_{n-1/2}}{\Delta\tau} = G(\mathbf{x}_n, \mathbf{v}_n), \tag{4.18}$$

$$(\mathbf{x}_{n+1}, \mathbf{v}_{n+1}) = \Psi^{\Delta\tau/\rho_{n+1/2}}(\mathbf{x}_n, \mathbf{v}_n), \tag{4.19}$$

$$\frac{t_{n+1} - t_n}{\Delta\tau} = 1/\rho_{n+1/2}, \tag{4.20}$$

with initial step density $\rho_0 = 1$.

A more recent scheme from 2008 (first method in [16]) has been examined. It is symplectic[4] and the scaling function may depend on potential energy

[4]Symplecticity is a stronger property than symmetry, see e.g. [9].

$V(\mathbf{x}) = -GMm/\|\mathbf{x}\|$ and kinetic energy $T(\mathbf{v}) = m\|\mathbf{v}\|^2/2$ (m is the comet's mass). We arrive at our choice of g if we use[5]

$$g(T,V) = \left(\frac{-GMm}{V}\right)^{3/2} = g(V). \tag{4.21}$$

The method works only for conservative systems where the total energy $H = T + V$ stays constant. This is exploited to make the system (4.14) – (4.15) separable:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = \mathbf{v}g(V(\mathbf{x})) = \mathbf{v}g(H - T(\mathbf{v})), \tag{4.22}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\tau} = \mathbf{a}(\mathbf{x})g(V(\mathbf{x})). \tag{4.23}$$

The total energy H is determined by the initial $\mathbf{x}$ and $\mathbf{v}$. We integrate this system with the leapfrog method (step size $\Delta\tau$ and in the form (3.10) – (3.11)), where for time reconstruction

$$\frac{t_{n+1/2} - t_{n-1/2}}{\Delta\tau} = g(H - T(\mathbf{v}_n)) \tag{4.24}$$

is used.

The third and last candidate was the adaptive Verlet method, see [9, Chapter 9.2.4] or [17]. The transformed system (4.14) – (4.15) is rewritten as a differential system of equations with an algebraic constraint:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = \mathbf{v}\lambda, \tag{4.25}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\tau} = \mathbf{a}(\mathbf{x})\lambda, \tag{4.26}$$

$$\lambda = g(\mathbf{x}). \tag{4.27}$$

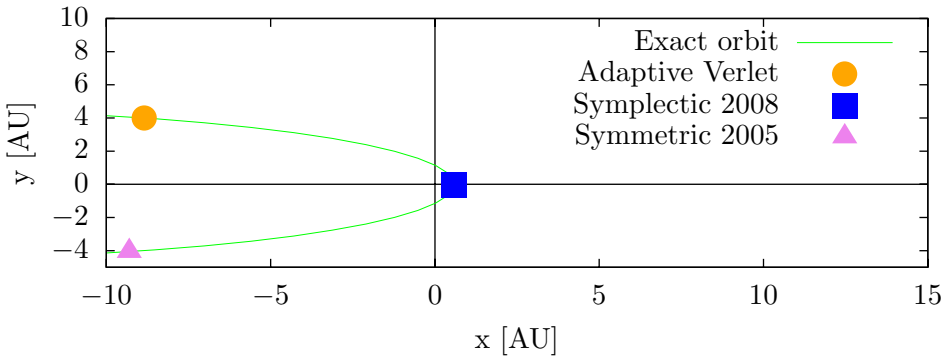
The new variable λ, like ρ before, is transported together with the other space space variables. The discretization is

$$\frac{\mathbf{x}_{n+1/2} - \mathbf{x}_{n-1/2}}{\Delta\tau} = \mathbf{v}_n\lambda_n, \tag{4.28}$$

$$\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta\tau} = \mathbf{a}(\mathbf{x}_{n+1/2})\frac{\lambda_n + \lambda_{n+1}}{2}, \tag{4.29}$$

$$\frac{\lambda_n + \lambda_{n+1}}{2} = g(\mathbf{x}_{n+1/2}), \tag{4.30}$$

Figure 4.3.: Phase error of adaptive methods



where the last equation defines a (symmetric) update rule[6] for the rescaling variable λ.

We integrated the problem again until the end time $10T$. All three adaptive methods (parameter $\Delta\tau$ chosen such that roughly 9000 steps were used) reproduced the orbit with no visible deviation when looking at the ellipse as a whole. The adaptive choice of the time step concentrates the computational effort on the critical region, thus leading to a much more efficient integration. While both symmetric-only methods have a notable phase error, the symplectic method is very close to the pericentre after the integration, see Figure 4.3.

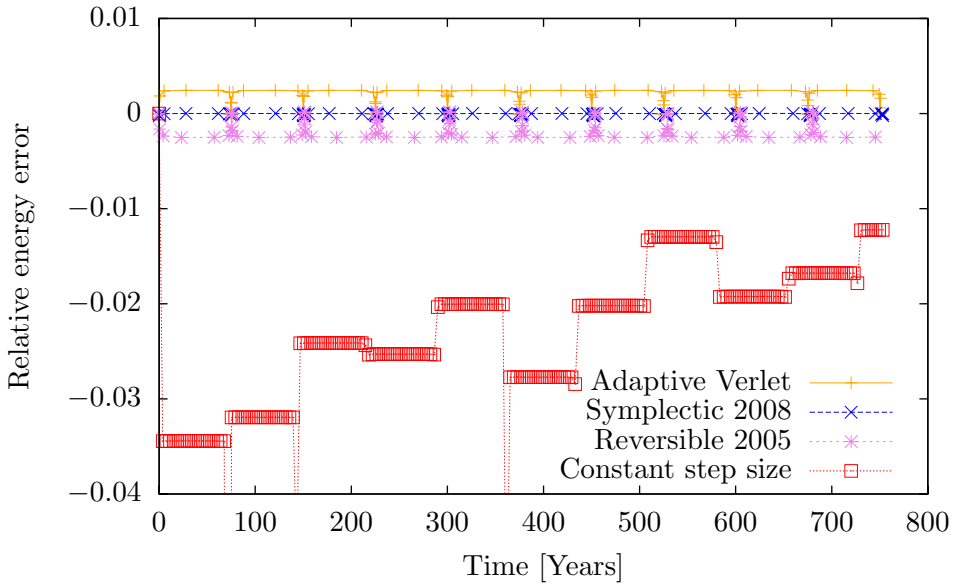
The constant step size method and all three adaptive methods show no drift in total energy, meaning that the energy error stays bounded for long times, see Figure 4.4. The error is smaller for the adaptive methods. For the symplectic method the energy error is especially small.

As an additional experiment, an asymmetric adaptive Verlet integrator was applied to the problem. The step size is chosen before the step as $h = g(\mathbf{x}_n)\Delta\tau$ and then the system is moved forward with a full step $\mathbf{x}_{n+1} = \Psi^h(\mathbf{x}_n)$. One can immediately see that the method is asymmetric because the step size $t_{n+1} - t_n$ depends only on information of time t_n. This eccentric problem reacts very sensitive to this asymmetry, see Figure 4.5. The numerical trajectory collapses, producing inferior results than the constant step size method. The reason is that the energy error grows very rapidly due to the asymmetry.

Now we asked ourselves which adaptive scheme was suited best for the Boris-Buneman integrator. The symplectic method from 2008, although giving the best results in the Kepler case, is not general enough because it requires that

[5] One could take the constants out of g because only the proportionality is important.
[6] Other choices of the symmetric update are possible, see [9].

Figure 4.4.: No energy drift for symmetric methods



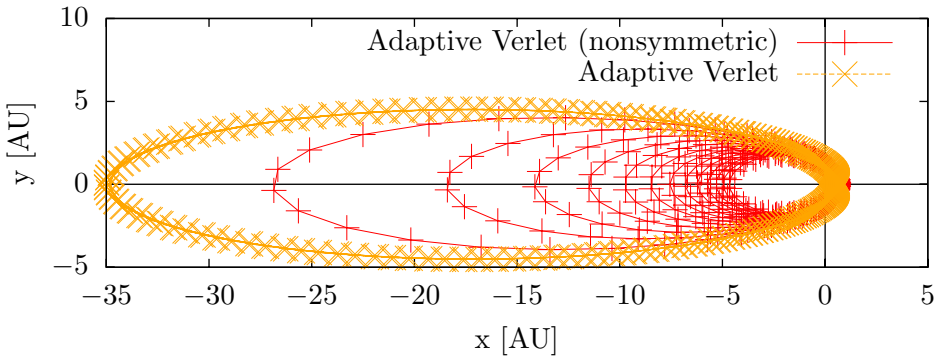
the total energy of the system is conserved. A second reason of the nonapplicability is that it requires a separable system. Our system (2.1) – (2.2) becomes nonseparable as soon as there is a magnetic field. The difficulty in the symmetric method from 2005 is the calculation of the control function G. While in this single-object case we have an analytic derivative of the density objective Q, there seems to be no analytic derivative available if we choose, e.g., Q as the volume of the particle beam. One could approximate the derivative numerically by using a finite difference, but this would, by the author's current understanding, mean to move all particles a little bit forth and back in time every step, increasing the needed computation considerably. The adaptive Verlet method is therefore the only candidate we examine further.

4.4. Adaptive Boris-Buneman method

4.4.1. Derivation

The adaptive Verlet method ([9, Chapter 9.2.4] or [17]) won our evaluation due to its simplicity. Like in the Kepler case we are satisfied with a rescaling function g that depends only on positions $\mathbf{X}$. Since the goal of rescaling in our case is to adapt the frequency of internal field solves, and the force of the

Figure 4.5.: Asymmetry in method breaks it down



internal field depends mainly[7] on positions, this should be good enough[8].

The Boris-Buneman method is very similar to the leapfrog/Verlet method, so it is not difficult to copy the way how variable step sizes are incorporated. The system in τ,

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\tau} = \mathbf{V}g(\mathbf{X}), \tag{4.31}$$

$$\frac{\mathrm{d}\mathbf{V}}{\mathrm{d}\tau} = \mathbf{F}_V(\mathbf{X}, \mathbf{V}, t)g(\mathbf{X}), \tag{4.32}$$

is rewritten as a differential system with an algebraic constraint,

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\tau} = \mathbf{V}\lambda, \tag{4.33}$$

$$\frac{\mathrm{d}\mathbf{V}}{\mathrm{d}\tau} = \mathbf{F}_V(\mathbf{X}, \mathbf{V}, t)\lambda, \tag{4.34}$$

$$\lambda = g(\mathbf{X}), \tag{4.35}$$

where a new variable λ is introduced that will be propagated together with the other phase space variables. In analogy to (3.12) – (3.13), the discretized

[7] The velocity has an influence too (change of reference frame during the calculation of the internal field). But as long we do not know how this should influence the rescaling, we do not use a more complicated version.

[8] As a side node, it is possible to use the approach in [9, Chapter 9.5] to construct an adaptive Boris-Buneman method with rescaling based on both positions and velocities, because Boris-Buneman can be interpreted as the concatenation of a first order method and its adjoint.

equations which lead to the method are

$$\frac{\mathbf{X}_{n+1/2} - \mathbf{X}_{n-1/2}}{\Delta\tau} = \mathbf{V}_n \lambda_n, \tag{4.36}$$

$$\frac{\mathbf{V}_{n+1} - \mathbf{V}_n}{\Delta\tau} = \mathbf{F}_V(\mathbf{X}_{n+1/2}, (\mathbf{V}_n + \mathbf{V}_{n+1})/2, t_{n+1/2}) \frac{\lambda_n + \lambda_{n+1}}{2}, \tag{4.37}$$

$$\frac{\lambda_n + \lambda_{n+1}}{2} = g(\mathbf{X}_{n+1/2}), \tag{4.38}$$

where equation (4.38) is new for the adaptive variant and defines the time step adaption[9]. Note that it is chosen in a symmetric way too, and therefore the method is symmetric and time-reversible.

What is not included in these equations is that the adaptive method also has to reconstruct the real time t[10]. For this we get together with the position update:

$$\frac{t_{n+1/2} - t_{n-1/2}}{\Delta\tau} = \lambda_n. \tag{4.39}$$

Summarizing, one step of the algorithm is presented in Algorithm 6. We can reuse the subroutines of the constant step size integrator. One additional scalar variable, λ, has to be stored, and updated in line 5. Accuracy is controlled with $\Delta\tau$, which is the constant step size in the transformed time. Using $t_n = (t_{n+1/2} + t_{n-1/2})/2$ in addition to (4.39) to have t defined at full steps, we can say that one step moves the real time forward by $t_{n+1} - t_n = (\lambda_n + \lambda_{n+1})\Delta\tau/2 = g(\mathbf{X}_{n+1/2})\Delta\tau$.

Algorithm 6 AdaptiveBorisBuneman($\mathbf{X}, \mathbf{V}, t, \lambda, \Delta\tau$)

1: $(\mathbf{X}, t) \leftarrow$ BBPush($\mathbf{X}, \mathbf{V}, t, \lambda\Delta\tau/2$)
2: $\mathbf{E} \leftarrow$ QueryExternalEField($\mathbf{X}, t$) + CalculateSpaceChargeE($\mathbf{X}, \mathbf{V}$)
3: $\mathbf{B} \leftarrow$ QueryExternalBField($\mathbf{X}, t$) + CalculateSpaceChargeB($\mathbf{X}, \mathbf{V}$)
4: $g \leftarrow g(\mathbf{X})$
5: $\lambda \leftarrow 2g - \lambda$
6: $\mathbf{V} \leftarrow$ BBKick($\mathbf{V}, \mathbf{E}, \mathbf{B}, g\Delta\tau$)
7: $(\mathbf{X}, t) \leftarrow$ BBPush($\mathbf{X}, \mathbf{V}, t, \lambda\Delta\tau/2$)
8: **return** $(\mathbf{X}, \mathbf{V}, t, \lambda)$

[9] [18] suggests to use another symmetric relation. However, we could not observe any drawbacks of this simplest choice in our application.

[10] "Reconstructing time" had to be done already for the constant step size methods, but there it was simply $t_{n+1} - t_n = h$

4.4.2. Choice of time rescaling

In order to apply the method, we have to specify $g(\mathbf{X})$. The intuition is that the self-field has to be computed more frequently when the charge density is higher (beam volume is smaller), and less frequently when the charge density is lower (beam volume is higher). Given this simple but plausible argument, we use

$$g(\mathbf{X}) = (\min\{\Delta x, \Delta y, \Delta z\})^{3/2}, \tag{4.40}$$

where $\Delta x, \Delta y, \Delta z$ are the maximum beam extents in the respective axis directions, so for example

$$\Delta x = \max_i\{x_{i,x}\} - \min_i\{x_{i,x}\}. \tag{4.41}$$

This is essentially a scaling that depends on the beam width, as the length Δz is usually larger. The exponent was chosen as $3/2$ since the scaling based on the distance in the Kepler case used the same exponent, where it is justified by a scaling invariance argument (see [9, section 9.3.2] and [14]). It is not clear whether the same argument can be made here, but the type of force (magnitude proportional to the inverse of squared distance) is at least the same.

A systematic comparison of different choices has not been done yet. As an alternative to account better for local effects inside the beam, we tried to use a scaling that depends on the maximum magnitude of the electric field vectors on the grid, but this led to a very non-smooth behaviour of g and did not give good results at all. So, for the moment, we continue with this simple choice, not excluding that there could be a better solution.

4.4.3. Minimum and maximum step size setting

Often one wants to limit the stepsize $\Delta t = \Delta\tau g(\mathbf{X}_{n+1/2})$ such that

$$\Delta t_{\min} \leq \Delta t \leq \Delta t_{\max}. \tag{4.42}$$

This can be achieved by limiting g between $a = \Delta t_{\min}/\Delta\tau$ and $b = \Delta t_{\max}/\Delta\tau$. A hard limit

$$\hat{g} = \begin{cases} a & \text{if } g \leq a \\ g & \text{if } a < g < b \\ b & \text{if } g \geq b \end{cases} \tag{4.43}$$

provokes oscillation of the rescaling variable λ once a bound is reached. Therefore the bounds are enforced in a smoother way with

$$\hat{g} = \begin{cases} \frac{(c-a)g}{c} + a & \text{if } g < c \\ g & \text{if } c \leq g \leq d \\ \frac{(d-b)d}{g} + b & \text{if } g > d \end{cases} \qquad c = \frac{5}{4}a, \quad d = \frac{4}{5}b \tag{4.44}$$

Of course other choices are possible. A simpler approach (proposed in [9, page 248]) is to use $\hat{g} = b\frac{g+a}{g+b}$. Our choice leaves g far away from the bounds invariant.

4.4.4. Overhead of adaptivity

How much does a step of the adaptive scheme cost compared to the old scheme? In the current form where g is chosen as a function of the maximum beam extents (in axis directions), it can be implemented extremly lightweight such that no significant overhead is added. The maximum beam extents have to be calculated by the space charge solver for repositioning of the grid anyway, so we can avoid recalculating these numbers. The only additional requirements are storage of the rescaling variable λ and its updating, which can be neglected compared to the other work done per step. For other choices of g communication can become necessary, but we can line up messages right after the communication of the space charge solver. So, no other costly synchronization point for communication is needed.

4.4.5. Implementation in OPAL

This part is probably only relevant for OPAL developers.

In the current implementation, the integration logic (Boris-Buneman) is split across the classes `ParallelTTracker` and `PartPusher`. The latter has methods to do a substep (push or kick) for a single particle, and the former contains the big picture of the integration logic by deciding when to use which substep. See Figure 4.6 for a simplified view of the interaction. The tracker class contains the outer simulation loop as well as the inner loops over the particles.

The first work in the implementation phase was to extract the integration logic into a separate class. The idea is depicted in Figure 4.7. The method `move` is invoked once per step by the tracker class, telling the integrator that one full step should be integrated. Of course, the integrator is not completely autonomous, for example the field evaluation requires a *callback* to the tracker, which in turn is responsible for updating the field vectors. For this rewriting

of code some simplifications have been made (like assuming no boundary conditions are present), which would have to be incorporated again for obtaining the original functionality.

Having the integration logic at a glance, the adaptive method was easy to implement. The adaptive integrator is just a new subclass of the `Integrator` class, see Figure 4.8 for a class diagram. If the new method tag `PARALLEL-TA` is used in the input file, a flag gets passed to the constructor of `ParallelTTracker` to trigger instantiation of the subclass `BorisAdaptiveIntegrator`. Up to now, the minimum/maximum step size setting is hardcoded in the integrator, but this could be changed to allow customization via the input file.

The emission of particles uses a logic where variable steps cannot yet be applied, therefore the first iterations of the simulation loop are left untouched. As soon as the emission is finished, the integration method is switched.

Figure 4.6.: Sequence diagram showing current organization of OPAL time integration

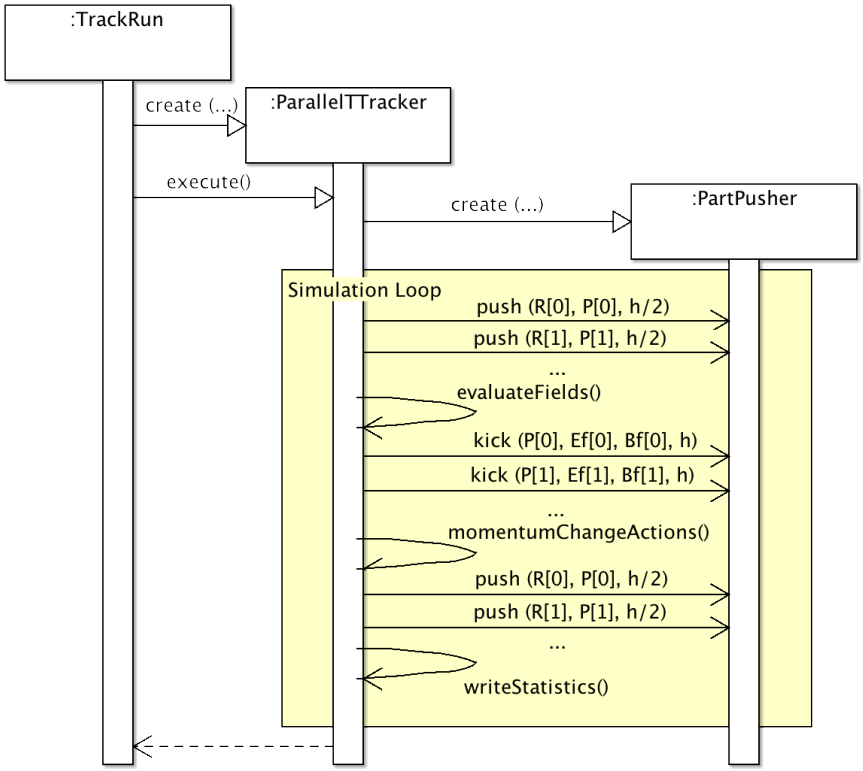


Figure 4.7.: Sequence diagram showing proposed organization of OPAL time integration

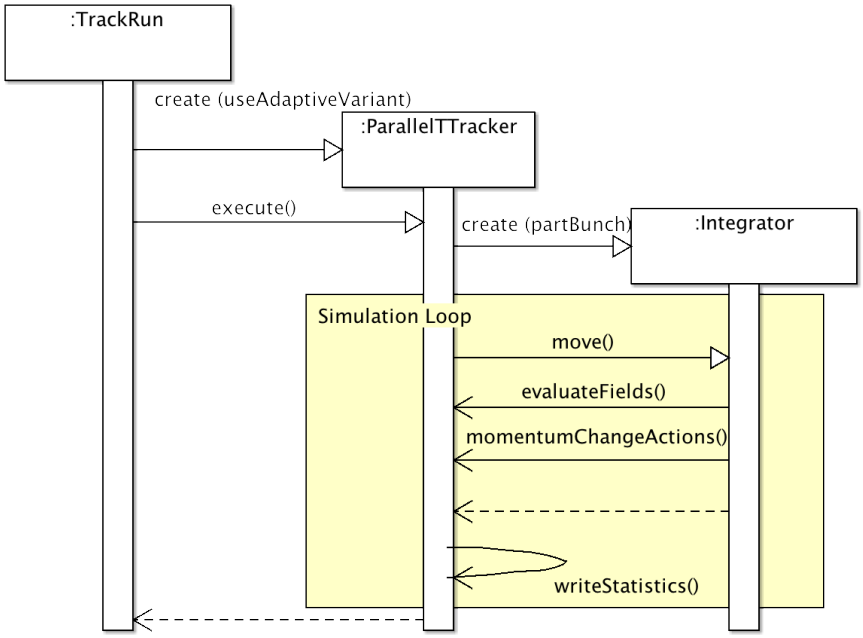
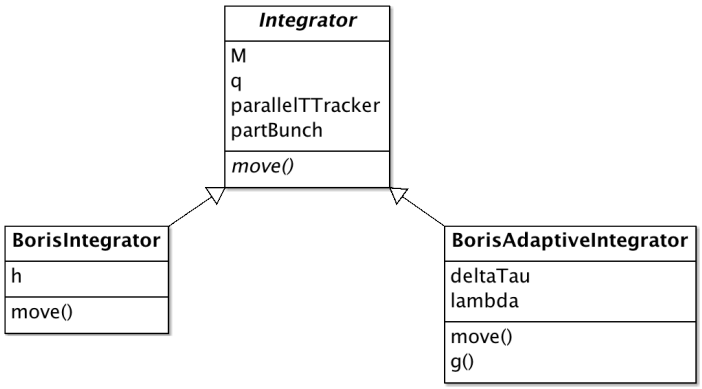


Figure 4.8.: Class diagram: Time integration logic as subclass of an abstract integrator



4.5. Multiple time stepping extension

The adaptive strategy presented so far does not account for any time step requirements of external fields, except that a maximum step size can be chosen. The method worked well on the tested examples, but there could arise a constraint that the external fields have to be sampled with a certain frequency which is higher than the frequency that one wants to use for the space charge solver. Such a constraint would limit the efficiency of the presented method. An idea is to evaluate the external fields more often than to calculate the internal field. In the current method these two things are linked together, but a multiple time stepping approach (see for example in [11]) could make sense. While for a local single-computer simulation this may be not a huge improvement, for a massively parallel simulation it must be noted that only internal field evaluations require communication, so a few external field substeps might be run without communication. Normally multiple time stepping is applied to constant step size integrators, but subcycling small steps for the external fields could also be integrated into the adaptive integrator. The implementation itself would not be a big challenge, but one has first to come up with a specification of the constraints.

5. Results

5.1. N-body prototype

During the thesis, a two-dimensional N-body prototype (for low $N < 500$) was programmed to understand the implementation of the Boris-Buneman integrator and to try out adaptive variants. It also helped to understand the notion of emittance. However, no reasonable benchmark results could be obtained, as often the constant step size method did not converge at all and hence no reference solution was available to measure against. Most probably this was an effect of the pairwise force calculation[1].

5.2. Benchmarks

5.2.1. Setup

The presented method was tested on a laptop computer on two real world-simulations of OPAL, with a low mesh resolution of $8 \times 8 \times 8$ and with only $20'000$ particles.

The reference solutions were calculated with the Boris-Buneman integrator using a small constant step size of $\Delta t_{\text{ref}} = 10^{-13}$. The adaptive method was run without a maximum step size setting, but with a minimum stepsize of Δt_{ref}.

Several runs were performed with the constant step size method and with the adaptive method. Different values for the parameters Δt and $\Delta\tau$ were chosen such that the resulting number of steps covered the same range[2]. The accuracy of a run is judged with respect to three scalar statistical measures.

We observe the measures **x-emittance** and **z-emittance**. Beam emittance is an important quantity in the design of particle accelerators. It is a measure of how "compressed" the beam is, not only in position but also in momentum.

[1] An attempt to regularize close approaches has been made, but this did not help so the author proceeded with building the OPAL prototype.

[2] The relation between $\Delta\tau$ and number of steps depends on g and hence on the problem itself.

Up to constants, the emittance is the phase space volume of the beam[3]. x-emittance is related to the beam volume in the phase space projected onto the transverse x-axis, and z-emittance is the same for the longitudinal z-axis. OPAL gives us the *normalized emittance* in units of m.

The third measure we look at is the mean kinetic **energy** of a particle in the beam, measured in MeV.

Since a step of the adaptive method is about equally expensive as a step for the existing scheme, we measure computational effort in number of steps.

The following procedure is used to determine the error $\mathcal{E}$ of one run with respect to one measure. In the attempt to better judge the quality of the produced curve than comparing solely the values at the end, we compare the run's output at W equidistant points $z_1, z_2, \ldots, z_W$ along the longitudinal axis (z-axis) of the accelerator[4]. Then the maximum relative error among these individual errors counts as overall error. Suppose we have reference values $\{E_1, E_2, \ldots, E_W\}$ for the energy at $\{z_1, z_2, \ldots, z_W\}$, the energy error for a run is defined as

$$\mathcal{E}_{\text{Energy}}(\hat{E}_1, \hat{E}_2, \ldots, \hat{E}_W) = \max_{k=1,\ldots,W} \left\{ \left| \frac{\hat{E}_k - E_k}{E_k} \right| \right\}, \tag{5.1}$$

where the $\hat{E}_k$ are the statistical values obtained from the run.

The OPAL input files for the scenarios can be found in the appendix.

5.2.2. Scenario EGun-CTF3

The first test case is "EGun-CTF3", integrated up to $z = 1.0$ meters. For purpose of the simulation, we cite [20]:

> "The Paul Scherrer Institut (PSI) has a project to build a compact, high brightness free electron laser. For this purpose a new 2.5-cell RF gun has been designed at PSI and is now ready to be manufactured. The RF gun plays an important role in preserving beam emittance and hence to deliver a high quality beam to the injector."

For plots of emittance and energy of the scenario, see Figures 5.1 – 5.3. The choice of the step sizes for one run of the adaptive method is depicted in Figure 5.4.

[3] Of course, "volume" would have to be made more precise, since a collection of points does not have a volume itself. For a rigorous introduction, see e.g. [19]

[4] The integration procedure has been modified so that the z_{stop} of a track is reached pretty exactly, by taking a smaller step if the last step would overshoot the end-coordinate. Of course another method would be to use interpolation in the post processing of the statistics file.

Figure 5.1.: x-emittance of scenario EGun-CTF3

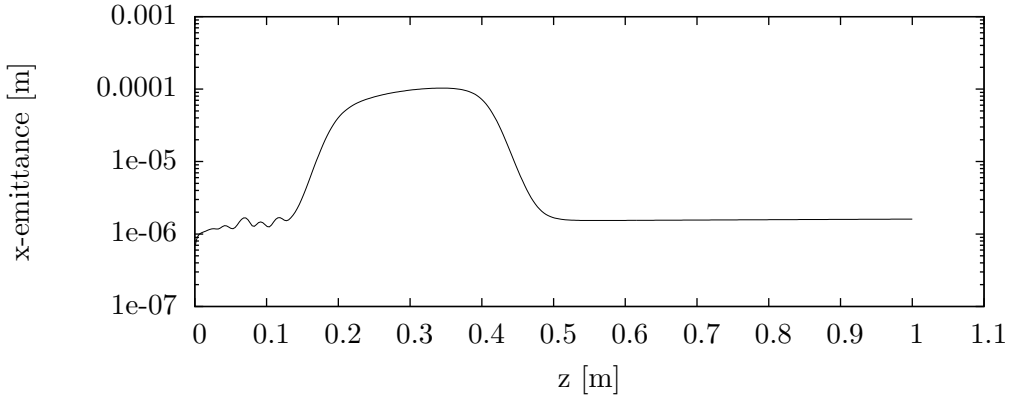


Figure 5.2.: z-emittance of scenario EGun-CTF3

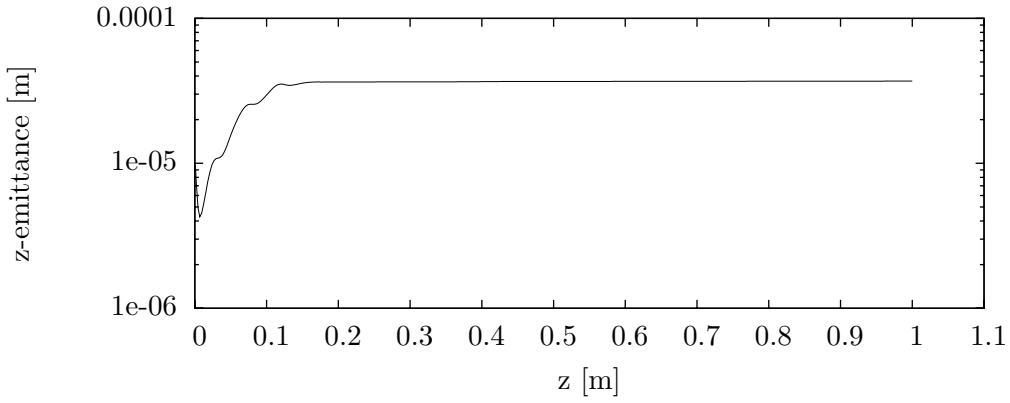


Figure 5.3.: Energy of scenario EGun-CTF3

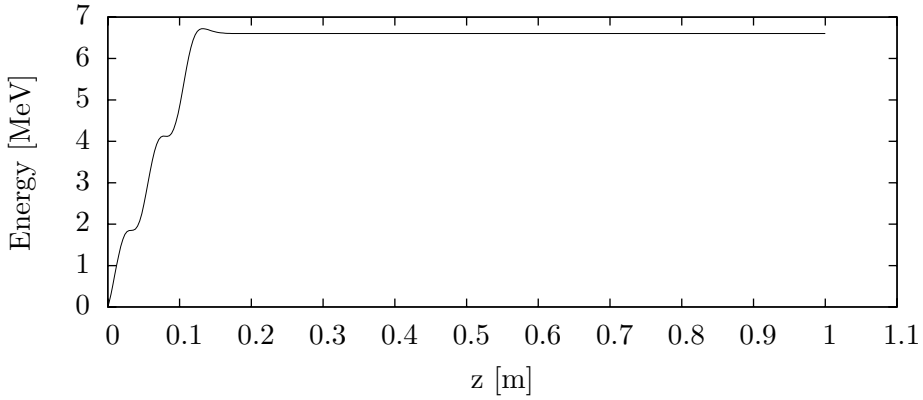
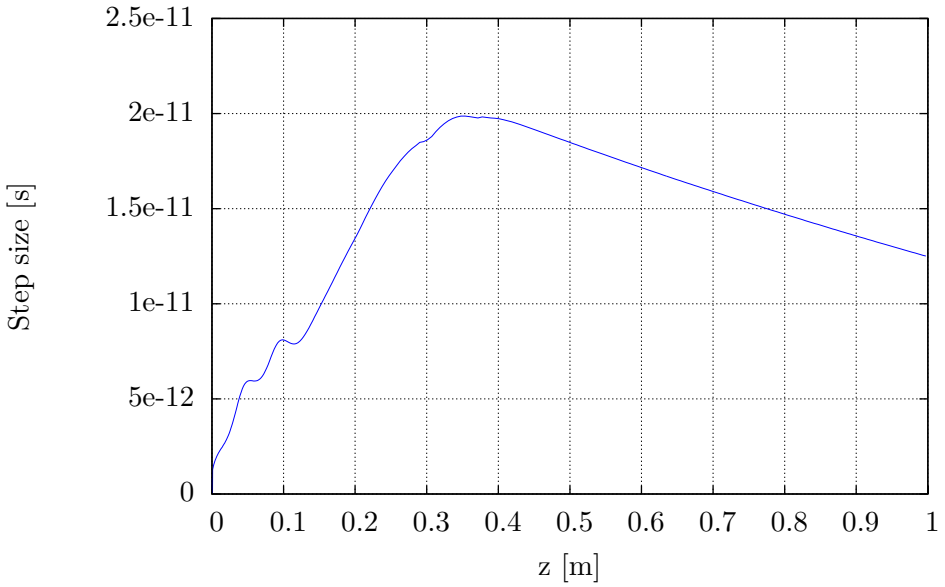


Figure 5.4.: Step size choice in scenario EGun-CTF3



Every 0.1 meters, the difference to the reference solution was measured. The reference solution took $33'572$ steps to finish. Figure 5.5 shows $\mathcal{E}_{\text{x-emittance}}$ for several runs of both methods. We can see that the adaptive method achieves the same accuracy with fewer steps, where the difference becomes notable when a relative error of less than 2% is required. Table 5.1 gives examples of how many steps are required to reach a prescribed error in x-emittance. For convergence plots of z-emittance and energy, see Figures 5.6 and 5.7.

Figure 5.5.: Convergence of x-emittance, scenario EGun-CTF3

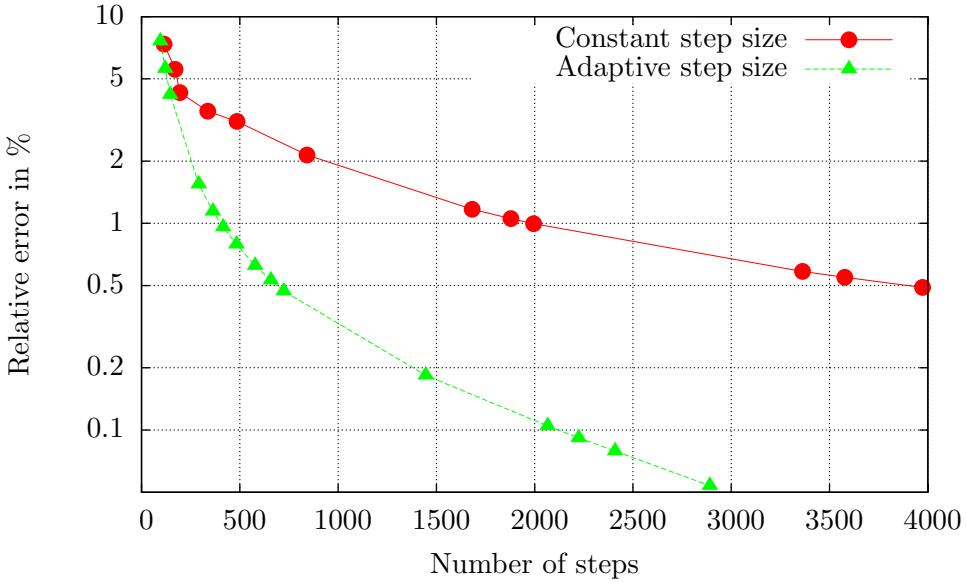


Table 5.1.: Number of steps to achieve a prescribed relative error in x-emittance, scenario EGun-CTF3. A step number of $m \pm \delta$ means that the error of one run with $m - \delta$ steps was above the tolerance, and that the errors of all runs with $m + \delta$ steps or more were below the tolerance.

Tolerated $\mathcal{E}$ in %	#Steps Boris-Buneman	#Steps Adaptive Boris-Buneman
5.0	183 ± 12	134 ± 13
1.0	1936 ± 58	389 ± 26
0.5	3775 ± 198	691 ± 33
0.1	$> 10'000$	2145 ± 79

Figure 5.6.: Convergence of z-emittance, scenario EGun-CTF3

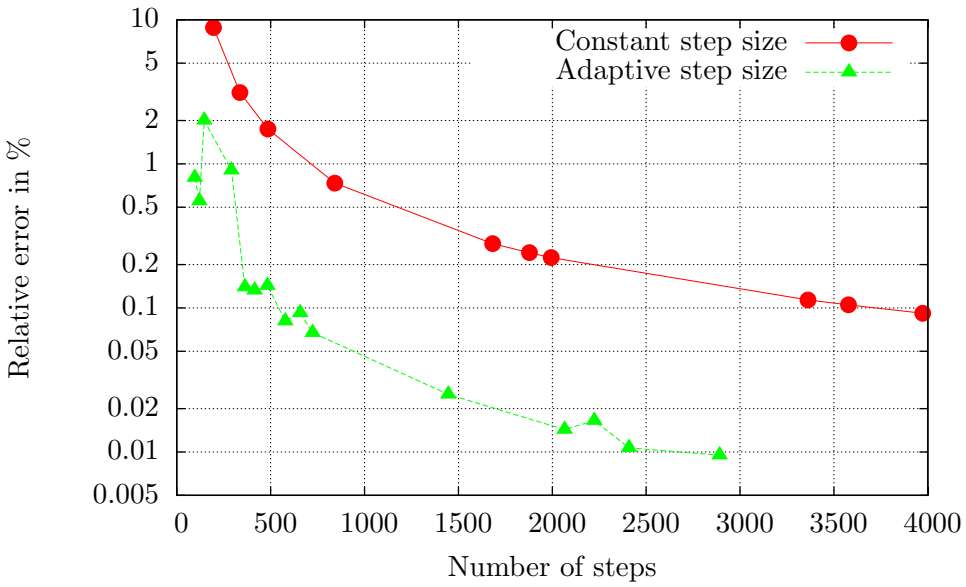


Figure 5.7.: Convergence of energy, scenario EGun-CTF3

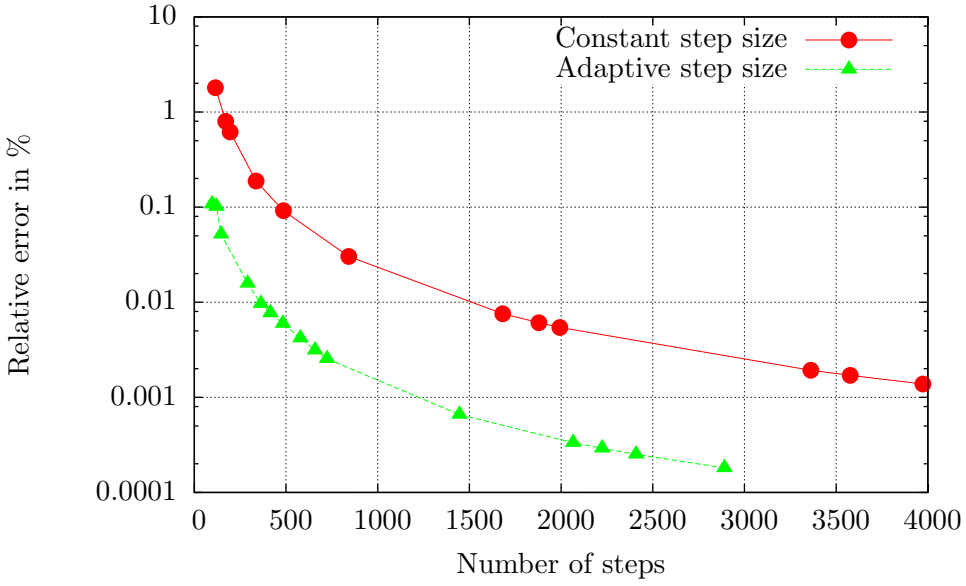
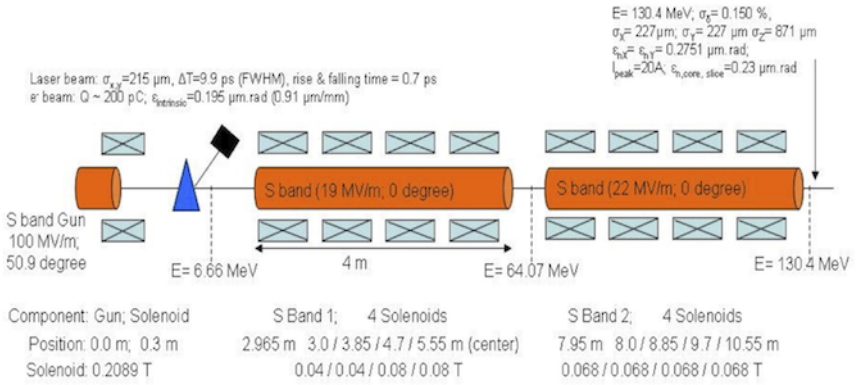


Figure 5.8.: SwissFEL injector. Source: [20]



5.2.3. Scenario FinPhase3

The second test case is "FinPhase3", integrated up to $z = 7.5$ meters. For context of the simulation, we cite again [20]:

> "The first part of the SwissFEL injector test facility consists of a 2.5 cell RF gun, a solenoid for emittance compensation and invariant envelope matching, a triplet of quadrupoles for emittance measurements and two TW S-Band structures."

See also Figure 5.8 to get an impression of the structure. Since we stop at $z = 7.5$ m, we encounter only the first part of the accelerator elements. For plots of emittance and energy of the scenario, see Figures 5.9 – 5.11. The choice of the step sizes for one run of the adaptive method is depicted in Figure 5.12.

Every 0.5 meters, the difference to the reference solution was measured. The reference solution took 250′704 steps to finish. In analogy to the first test case, we give convergence plots for all three measures in the Figures 5.13 – 5.15. In Table 5.2, we compare the effort needed to reach a certain accuracy in x-emittance. The difference is not that stunning like in the first test case, but still, almost only half of the steps are needed with the adaptive scheme to reach the same tolerance. The energy convergence is outstandingly good for this case.

Figure 5.9.: x-emittance of scenario FinPhase3

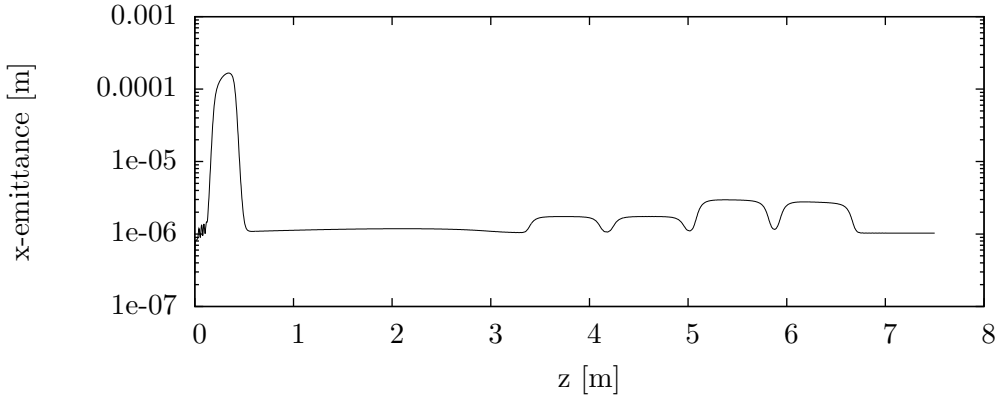


Figure 5.10.: z-emittance of scenario FinPhase3

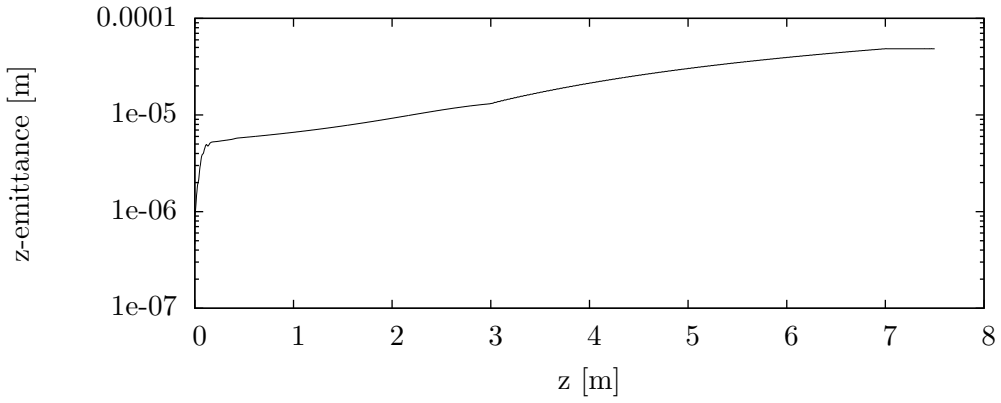


Figure 5.11.: Energy of scenario FinPhase3

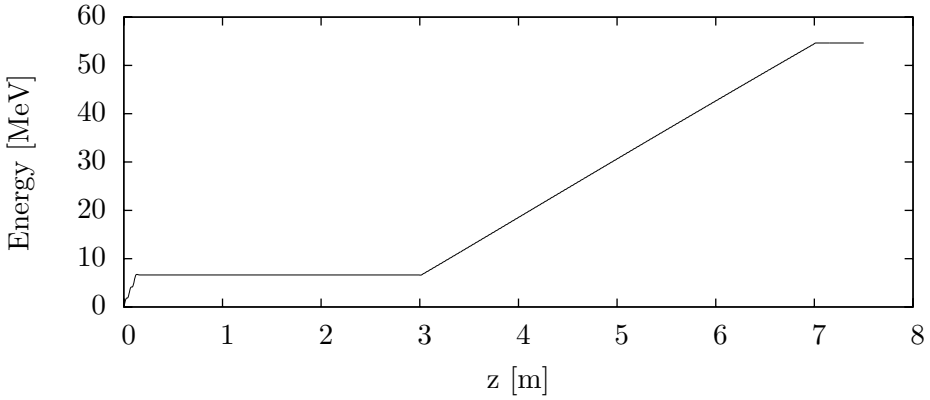


Figure 5.12.: Step size choice in scenario FinPhase3

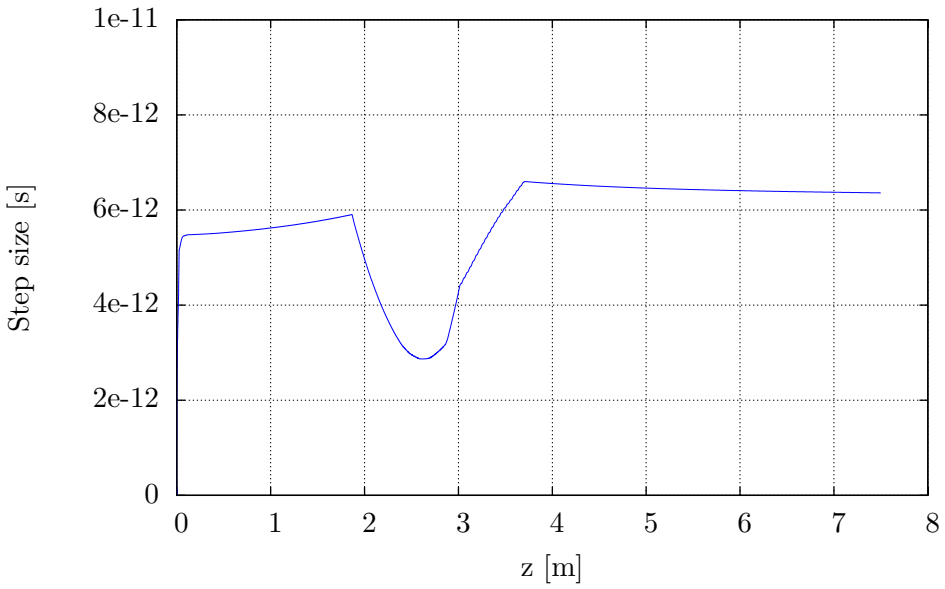


Figure 5.13.: Convergence of x-emittance, scenario FinPhase3

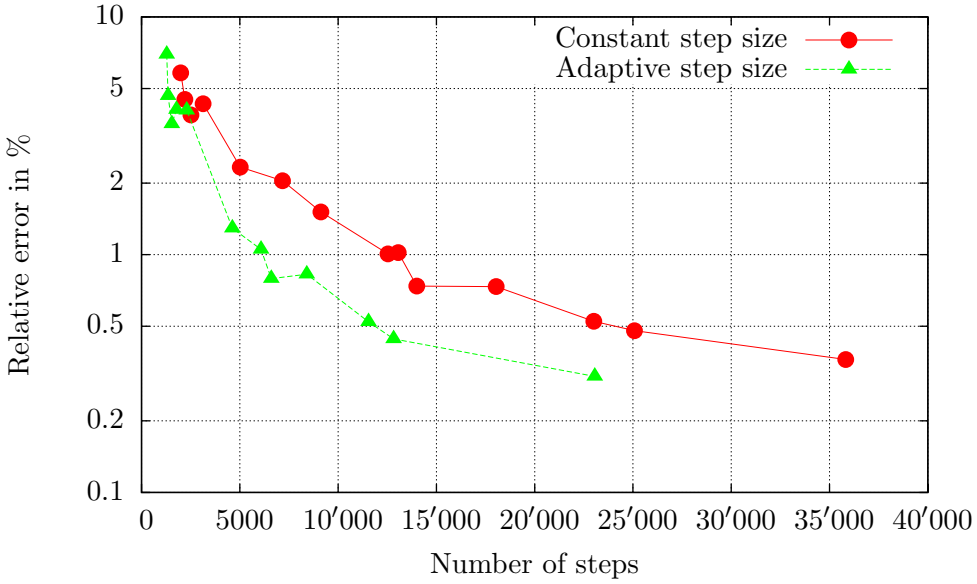


Table 5.2.: Number of steps to achieve a prescribed relative error in x-emittance, scenario FinPhase3. A step number of $m \pm \delta$ means that the error of one run with $m - \delta$ steps was above the tolerance, and that the errors of all runs with $m + \delta$ steps or more were below the tolerance.

Tolerated $\mathcal{E}$ in %	#Steps Boris-Buneman	#Steps Adaptive Boris-Buneman
5.0	2101 ± 106	1315 ± 35
1.0	13′530 ± 470	6342 ± 260
0.5	24′044 ± 1033	12′189 ± 639

Figure 5.14.: Convergence of z-emittance, Scenario FinPhase3

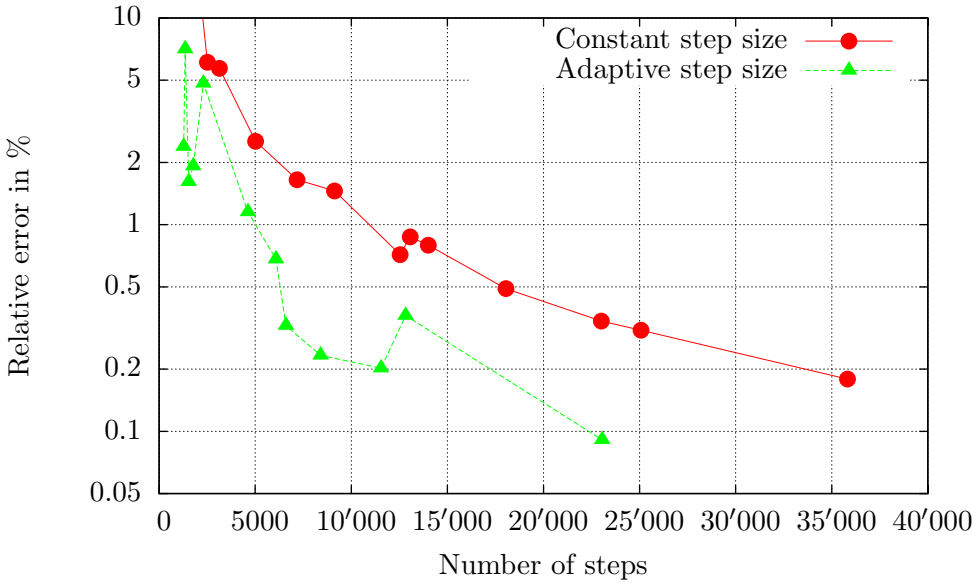
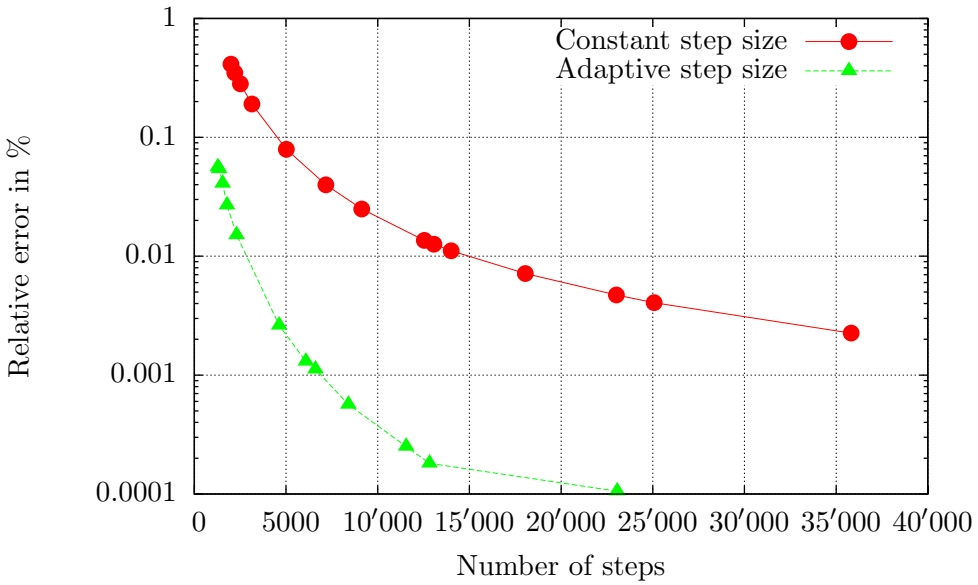


Figure 5.15.: Convergence of energy, scenario FinPhase3



5.3. Other results

The tests were repeated with a non-symmetrized version of the method, where λ was just set to the current value of g once in a step. Unlike for the Kepler test problem where this asymmetry led to a fatal breakdown, we did not observe negative consequences for these two test cases. But since the symmetry comes for free it surely makes sense to preserve it. Maybe it matters if one needs good energy conservation in a long term integration.

6. Discussion

The adaptive Boris-Buneman method is a modification of the adaptive Verlet method and rather a recombination of existing techniques than a new method. However, while adaptive mesh refinement techniques exist, we are not aware of an attempt that applies temporal adaptivity in this way to a PIC code.

We can say that adaptive time stepping leads to a clearly more efficient integration than the constant step size method for the tested examples. We expect that the results carry over to simulations with a higher mesh resolution and more particles. However, it must be noted that in both scenarios the method profits from the great variations in beam size right after emission. The gain in computation time will most probably not be that high when looking at a less interesting interval, e. g., at scenario FinPhase3 for $z \in [10\text{ m}, 20\text{ m}]$. But as soon as the beam width decreases again, like at quadrupole elements that would come later on in scenario FinPhase3 (at $z \approx 23$ m), we expect the adaptive method to show its advantages.

The step size can already now be adapted in a coarse way by manually splitting the simulation into different tracks and setting individual constant time steps for the respective z-intervals. In this way a good chunk of computational effort can be saved. However, this thesis has made the first step towards an automated step size control.

The implementation is still in the prototype-phase. Some bugfixes and minor changes are necessary to integrate the method fully into OPAL. Of course the method must be tested on more scenarios. It could be investigated whether a multiple time stepping scheme further enhances efficiency. Other future work could encompass an evaluation of different choices of the time rescaling. After all, there is a good chance that time to solution in OPAL simulations can be reduced with such an approach.

Bibliography

[1] Paul Scherrer Institut. http://www.psi.ch/.

[2] A. Adelmann, Ch. Kraus, Y. Ineichen, S. Russel, Yuanjie Bi, and J. Yang. The OPAL (Object Oriented Parallel Accelerator Library) Framework . Technical Report PSI-PR-08-02, Paul Scherrer Institut, 2008.

[3] The AMAS group. http://amas.web.psi.ch/.

[4] SwissFEL project. http://www.psi.ch/media/swissfel-the-future-project.

[5] Wikipedia. Fast multipole method, 2010. http://en.wikipedia.org/w/index.php?title=Fast_multipole_method&oldid=403908463.

[6] Ch. K. Birdsall and A. B. Langdon. *Plasma physics via computer simulation*. McGraw-Hill, 1985.

[7] The Electrostatic Particle In Cell (ES-PIC) Method. http://www.particleincell.com/2010/es-pic-method/.

[8] A. Adelmann, P. Arbenz, and Y. Ineichen. A fast parallel Poisson solver on irregular domains applied to beam dynamics simulations. *J. Comput. Phys.*, 229:4554–4566, 2010.

[9] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2004.

[10] L. Patacchini and I. H. Hutchinson. Explicit time-reversible orbit integration in particle in cell codes with static homogeneous magnetic field. *J. Comput. Phys.*, 228:2604–2615, 2009.

[11] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations (Springer Series in Computational Mathematics)*. Springer, 2006.

[12] Wikipedia. Halley's comet, 2011. http://en.wikipedia.org/w/index.php?title=Halley%27s_Comet&oldid=448827409.

[13] National Space Science Data Center, Comet Photo Gallery. `http://nssdc.gsfc.nasa.gov/photo_gallery/photogallery-comets.html`.

[14] C. J. Budd, B. Leimkuhler, and M. D. Piggott. Scaling invariance and adaptivity. *Appl. Numer. Math.*, 39:261–288, December 2001.

[15] E. Hairer and G. Söderlind. Explicit, time reversible, adaptive step size control. *SIAM Journal on Scientific Computing*, 26(6):1838–1851, 2005.

[16] K. Modin. On explicit adaptive symplectic integration of separable hamiltonian systems. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 222(4):289–300, 12 2008.

[17] W. Huang and B. Leimkuhler. The adaptive Verlet method. *SIAM J. Sci. Comput*, 18:239–256, 1997.

[18] S. Cirilli, E. Hairer, and B. Leimkuhler. Asymptotic error analysis of the adaptive Verlet method. *BIT*, 39:25–33, 1998.

[19] K. Wille. *Physik der Teilchenbeschleuniger und Synchrotronstrahlungsquellen.* Teubner, 1996.

[20] A. Falone, A. Adelmann, J.-Y. Raguin, and L. Stingelin. RF GUN STUDIES FOR THE SWISSFEL INJECTOR. In *International Particle Accelerator Conference (IPAC11) San Sebastian, Spain*, Accelerator Concept Division, Paul Scherrer Institut, 5232-Villigen, Switzerland, 2011.

A. Appendix

A.1. OPAL input files

A.1.1. EGun-CTF3.in

```
Option, AUTOPHASE = 4;
Title, string = "Phase 1 of PSI FEL 250 Injector with CTF3 RF Photoinjector";
QB = 0.2e-9;
BF = 2998.0;
BC = QB*BF;
FINSS_RGUN: RFCavity, L = 0.34986, VOLT = 100.0, FMAPFN = "CTF3_Ez_ASTRA.opal",
    ELEMEDGE = 0.0, TYPE = "STANDING", FREQ = BF,
    LAG = -3.5/180.0*PI;
FIND1_MSOL10: Solenoid, L = 0.6, KS = 0.206, FMAPFN = "NEW_SINGLE_SOL_NOFRINGE_ASTRA.opal",
    ELEMEDGE = 0.3;
FIND1_MQ10: Quadrupole, L = 0.1, K1 = 0.0, ELEMEDGE = 1.38;
FIND1_MQ20: Quadrupole, L = 0.1, K1 = 0.0, ELEMEDGE = 1.56;
FIND1_MQ30: Quadrupole, L = 0.1, K1 = 0.0, ELEMEDGE = 1.74;
Injector: Line = (FINSS_RGUN, FIND1_MSOL10, FIND1_MQ10, FIND1_MQ20, FIND1_MQ30);
Fs1:FIELDSOLVER, FSTYPE = FFT, MX = 8, MY = 8, MT = 8,
    PARFFTX = true, PARFFTY = true, PARFFTT = true,
    BCFFTX = open, BCFFTY = open, BCFFTT = open,
    BBOXINCR = 1, GREENSF = INTEGRATED;
Dist1:DISTRIBUTION, DISTRIBUTION = "GUNGAUSSFLATTOPTH",
    sigmax = 0.000395, sigmapx = 0.0, corrx = 0.0,
    sigmay = 0.000395, sigmapy = 0.0, corry = 0.0,
    sigmat = 0.0, pt = 0.0, sigmapt = 0.0, corrt = 0.0, tRise = 7.45e-12, tFall = 7.45e-12,
    tPulseFWHM = 10.4e-12, ekin = 0.4, NBIN = 1, DEBIN = 30;
beam1: BEAM, PARTICLE = ELECTRON, pc = P0, NPART = 20000, BFREQ = BF , BCURRENT = BC,
    CHARGE = -1;
Select, Line = Injector;
// 100 steps for emission
track, line = Injector, beam = beam1, MAXSTEPS = 100, DT = 1.0e-13, ZSTOP = 1;
    run, method = "PARALLEL-T", beam = beam1, fieldsolver = Fs1, distribution = Dist1;
endtrack;
// Followup track to compare adaptive method
track, line = Injector, beam = beam1, MAXSTEPS = 1000000000, DT = 1e-12, ZSTOP = 1.0;
    run, method = "PARALLEL-T", beam = beam1, fieldsolver = Fs1;
endtrack;
QUIT;
```

A.1.2. FinPhase3.in

Nonrelevant elements (after $z = 7.5$ m) excluded from file.

```
TITLE, STRING = "SwissFEL Injector, Phase 3 (January 2011) 1D TWS fieldmap";
QB = 2e-10;
BFREQ = 2997.912*1e6;
BCURRENT = QB*BFREQ;
TWFMAPSHIFT = 0.075;
TWSOLSHIFT = -0.375;
OPTION, AUTOPHASE = 4;
FINSS_RGUN_dphi = -3.30/180.0*PI;
FINSB01_RACC_dphi = 0.0/180.0*PI;
FIND1_RTDC_dphi = 0.0/180.0*PI;
FINSS_RGUN: RFCAVITY, L = 0.25, VOLT = 100.0, FMAPFN = "FINSS-RGUN.dat", ELEMEDGE = 0.0,
    TYPE = "STANDING", FREQ = 2997.912, LAG = FINSS_RGUN_dphi;
FINSB01_RACC: TRAVELINGWAVE, L = 4.15, VOLT = 19, FMAPFN = "TWS_PSI_Sband_ASTRA.dat",
    ELEMEDGE = 2.95+TWFMAPSHIFT, NUMCELLS = 120, MODE = 1/3, ACCURACY = 39, FREQ = 2998.0,
    LAG = FINSB01_RACC_dphi;
FIND1_RTDC: RFCavity, TYPE = NOAP, L = 0.1, VOLT = 0.0, FMAPFN = "FIND1_RTDC.h5part",
    ELEMEDGE = 0.9665, TYPE = "STANDING", FREQ = 3003.13, LAG = FIND1_RTDC_dphi;
FIND1_MSOL10: SOLENOID, L = 0.26, KS = 116.5*0.38704/220+0.0022,
    FMAPFN = "NEW_SINGLE_SOL_NOFRINGE_OPAL.dat", ELEMEDGE = 0.3;
FINSB01_MSOL10: SOLENOID, L = 0.75, KS = 0.04, FMAPFN = "INSB_MSLAC_Bz_NOFRINGE_OPAL.dat",
    ELEMEDGE = 3.375+TWSOLSHIFT;
FINSB01_MSOL20: SOLENOID, L = 0.75, KS = 0.04, FMAPFN = "INSB_MSLAC_Bz_NOFRINGE_OPAL.dat",
    ELEMEDGE = 4.225+TWSOLSHIFT;
FINSB01_MSOL30: SOLENOID, L = 0.75, KS = 0.08, FMAPFN = "INSB_MSLAC_Bz_NOFRINGE_OPAL.dat",
    ELEMEDGE = 5.075+TWSOLSHIFT;
FINSB01_MSOL40: SOLENOID, L = 0.75, KS = 0.08, FMAPFN = "INSB_MSLAC_Bz_NOFRINGE_OPAL.dat",
    ELEMEDGE = 5.925+TWSOLSHIFT;
END: SOLENOID, L = 0.01, KS = 0.0, ELEMEDGE = 70., FMAPFN = "INSB_MSLAC_Bz_NOFRINGE_OPAL.dat";
InjectorPhase3: Line = (FINSS_RGUN, FIND1_MSOL10, FINSB01_RACC, FINSB01_MSOL10, FINSB01_MSOL20,
    FINSB01_MSOL30, FINSB01_MSOL40, END);
Dist1: DISTRIBUTION, DISTRIBUTION = "GUNGAUSSFLATTOPTH", SIGMAX = 2*0.000275, SIGMAPX = 0.0,
    CORRX = 0.0, SIGMAY = 2*0.000275, SIGMAPY = 0.0, CORRY = 0.0, SIGMAT = 0.0, PT = 0.0,
    SIGMAPT = 0.0, CORRT = 0.0, TRISE = 4.32e-13, TFALL = 4.32e-13, TPULSEFWHM = 6.03e-12,
    EKIN = 0.63, NBIN = 1, DEBIN = 80;
Fs1: FIELDSOLVER, FSTYPE = FFT, MX = 8, MY = 8, MT = 8,
    PARFFTX = true, PARFFTY = true, PARFFTT = true,
    BCFFTX = open, BCFFTY = open, BCFFTT = open, BBOXINCR = 1, GREENSF = INTEGRATED;
beam1: BEAM, PARTICLE = ELECTRON, pc = P0, NPART = 20000, BFREQ = BFREQ, BCURRENT = BCURRENT,
    CHARGE = -1;
SELECT, LINE = InjectorPhase3;
// 100 steps for emission
TRACK, LINE = InjectorPhase3, BEAM = beam1, MAXSTEPS = 100, DT = 1e-13, ZSTOP = 0.2;
    RUN, METHOD = "PARALLEL-T", BEAM = beam1, FIELDSOLVER = Fs1, DISTRIBUTION = Dist1;
ENDTRACK;
// Followup track to compare adaptive method
TRACK, LINE = InjectorPhase3, BEAM = beam1, MAXSTEPS = 100000000, DT = 1e-12, ZSTOP = 7.5;
    RUN, METHOD = "PARALLEL-T", BEAM = beam1, FIELDSOLVER = Fs1;
ENDTRACK;
QUIT;
```