Dies ist Titel der Masterarbeit
This is the title of the thesis

David Symhoven

München 2017

Dies ist Titel der Masterarbeit
This is the title of the thesis

David Symhoven

Masterarbeit
am Lehrstuhl Computational & Plasma Physics
der Ludwig–Maximilians–Universität
München

vorgelegt von
David Symhoven
aus Witten

München, den 30.06.2017

Erstgutachter: Prof.Dr.Hartmut Ruhl
Zweitgutachter: Zweitgutachter

Abstract:
Blah Blah Blah Mr. Freeman

Contents

1 Introduction

Now it's going loose ...[1]

2

Fundamentals

2.1 Liénard-Wiechert Potentials

$$\begin{aligned} E &= E \\ B &= B \end{aligned} \tag{2.1}$$

2.2 Numerics

The following section deals with the numeric aspects of this thesis. We explain the underlying equation of motions and their history. After that, we go into several methods with which we can solve differential equations. Over the course of the last decades numerous methods were invented each of which has its own strength and weaknesses. Some of them are very easy to implement, which in turn usually leads to unprecise results. Others are quite complicated and sophisticated to implement, but very accurate. Therefore one should always consider which method is best for the problem and what it is one want to achieve.
Following this, we also want to define and calculate the numerical complexity of some chosen algorithms.

2.2.1 Equations of motion

As we know from mechanics the dynamic of a particle is determined by the forces acting on it. In our case there is a force due to electro-magnetic fields. That can be external fields, but also fields due to moving particles, as we explained in section 2.1.
The dynamics of our system is described by the *Lorentz-Newton* equation

$$\begin{aligned} \frac{\mathrm{d}x^\mu}{\mathrm{d}\tau} &= u^\mu \\ \frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} &= F^\mu_{\ \nu}\, u^\nu + g^\mu, \end{aligned} \tag{2.2}$$

which derivation is quite longish, why we want to refer to literature.
The term $F^{\mu}_{\ \nu}$ describes the electromagnetic field strength tensor

$$F^{\mu}_{\ \nu} = \begin{pmatrix} 0 & E_x & E_y & E_z \\ E_x & 0 & B_z & -B_y \\ E_y & -B_z & 0 & B_x \\ E_z & B_y & -B_x & 0 \end{pmatrix}. \tag{2.3}$$

The damping term g^{μ} considers the fact that charged particles radiate fields when they are moving which leads to a loss in their kinetic energy. Within the context of classical electrodynamics Max *Abraham* and Hendrick *Lorentz* discussed radiation damping in their same-named equation first. In 1938 *Dirac* generalized the equation whilst taking special relativity into account.
We now want to deal with how to solve the Lorentz-Newton equation (2.2) numerically.

2.2.2 Euler-Scheme

The most simple method is the explicit *Euler*-Method. It's easy to implement but not very accurate, as we shall see later. But before we go into the details of the explicit Euler-Scheme we need to address some prerequisites all following methods will have in common.
Starting point will always be a first order system of the kind

$$\begin{aligned} \frac{\mathrm{d}x^{\mu}}{\mathrm{d}\tau} &= u^{\mu} \\ \frac{\mathrm{d}u^{\mu}}{\mathrm{d}\tau} &= f^{\mu}(x^{\nu}, u^{\nu}) \\ x^{\mu}(\tau_0) &= x_0^{\mu} \\ u^{\mu}(\tau_0) &= u_0^{\mu}. \end{aligned} \tag{2.4}$$

Systems of higher order can always be reduced to a first order system.
In order to solve the equation of motion numerically the domain needs to be discretized. Therefore we divide the time interval into N equidistant partial intervals h, by defining

$$h := \Delta\tau = \tau_{i+1} - \tau_i.$$

The idea is to calculate each point along the trajectory $x_i^{\mu} = x^{\mu}(\tau_i)$ iteratively, starting from the initial values x_0^{μ} and u_0^{μ}. But to calculate these points all differential operators in (2.4) need to be discretized as well. That is where all methods differ. Each method has its own way to discretize the differential operators.

The basis of the Euler-Scheme is a first order Taylor expansion of the integration variable x^μ in τ around τ_i

$$x^\mu(\tau_{i+1}) = x^\mu(\tau_i) + \frac{\mathrm{d}x^\mu}{\mathrm{d}\tau}\Big|_{\tau=\tau_i} \underbrace{(\tau_{i+1} - \tau_i)}_{=h} + \mathcal{O}(h^2). \tag{2.5}$$

Analogously for u^μ and solving for $\frac{\mathrm{d}x^\mu}{\mathrm{d}\tau}$ and $\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau}$ respectively yields

$$\begin{aligned} \frac{x^\mu_{i+1} - x^\mu_i}{h} &= u^\mu_i \\ \frac{u^\mu_{i+1} - u^\mu_i}{h} &= f^\mu(x^\nu_i, u^\nu_i). \end{aligned} \tag{2.6}$$

This way of discretizing allows a very easy calculation of x^μ_i according to

$$\begin{aligned} x^\mu_{i+1} &= x^\mu_i + h\ u^\mu_i \\ u^\mu_{i+1} &= u^\mu_i + h\ f^\mu(x^\nu_i, u^\nu_i). \end{aligned} \tag{2.7}$$

In order for us to calculate the goodness of this approximation we need to introduce the *Procedural Error* and the *Order of Consistency*.[4]

2.2.2.1 Procedural Error and Order of Consistency

Definition 2.2.1 (Procedural Error and Order of Consistency) *Let $I \subseteq \mathbb{R}$ be a interval, $f : I \times \mathbb{R}^d \to \mathbb{R}^d$, $y : I \to \mathbb{R}^d$ a solution of the initial value problem*

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} y(\tau) &= f(\tau, y(\tau)), \\ y(\tau_0) &= y_0. \end{aligned} \tag{2.8}$$

(a) *The term*

$$\eta(\tau, h) := y(\tau) + hf(\tau, y(\tau)) - y(\tau + h) \quad \text{for } \tau \in I,\ 0 < h \leq b - \tau \tag{2.9}$$

is called local Procedural Error of the One-Step-Scheme at τ for the increment h.

(b) *The One-Step-Scheme has an Order of Consistency $p \geq 1$, if the local Procedural Error fulfils*

$$||\eta(\tau, h)|| \leq Ch^{p+1} \quad \text{for } \tau \in I,\ 0 < h \leq b - \tau, \tag{2.10}$$

with a constant $C \geq 0$, which is independent of τ and h.

Descriptively the Procedural Error is the difference between the exact solution $y(\tau + h)$ and the result, which we get from the One-Step-Scheme starting from the exact solution at the earlier time step $y(\tau)$. Figure 2.1 illustrates the situation.

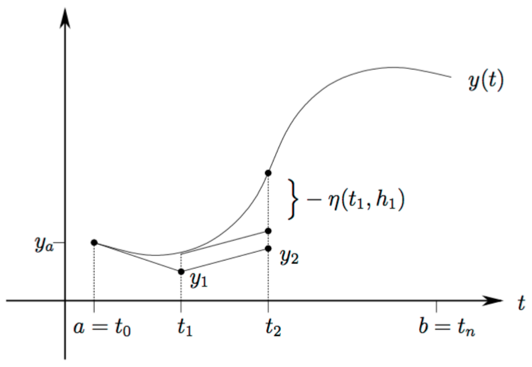

Figure 2.1: Illustration of the Procedural Errors of an One-Step-Scheme. [4]

We now want to use the definitions 2.2.1 to calculate the Order of Consistency of the Euler-Scheme.
Starting point is the system (2.2). Thereby we focus on the equation for u^μ, since x^μ can be easily integrated from u^μ. Following Definition 2.2.1 we have

$$y = u^\mu. \tag{2.11}$$

We get

$$\eta(\tau, h) = u^\mu(\tau_i) + hf^\mu(x^\nu, u^\nu) - u^\mu(\tau_{i+1}). \tag{2.12}$$

The last term can be calculated with a Taylor-expansion analogously to (2.5).

$$u^\mu(\tau_{i+1}) = u^\mu(\tau_i) + h\left.\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau}\right|_{s=\tau_i} + h^2\left.\frac{\mathrm{d}^2u^\mu}{\mathrm{d}\tau^2}\right|_{s=\tau_i}. \tag{2.13}$$

Plugging in (2.13) in (2.12) yields

$$\begin{aligned} &\eta(\tau,h) = u^\mu(\tau_i) + h\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} - u^\mu(\tau_{i+1}) \\ \overset{(2.13)}{\Longrightarrow}\ &\eta(\tau,h) = u^\mu(\tau_i) + h\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} - u^\mu(\tau_i) - h\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} - h^2\frac{\mathrm{d}^2u^\mu}{\mathrm{d}\tau^2} \\ \Longleftrightarrow\ &\eta(\tau,h) = \frac{\mathrm{d}^2u^\mu}{\mathrm{d}\tau^2}\ h^2, \end{aligned} \tag{2.14}$$

since $\frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} = f^\mu(x^\nu, u^\nu)$ holds for the Euler-Scheme. Thus

$$|\eta(\tau,h)| \leq Ch^2 \quad \text{with } C := \frac{1}{2}\max_{\tau\in\mathcal{D}(u^\mu)}\left|\frac{\mathrm{d}^2u^\mu}{\mathrm{d}\tau^2}\right|. \tag{2.15}$$

$\mathcal{D}(u^\mu)$ denotes the domain of u^μ. Therefore, the Euler-Scheme has an Order of Consistency of one.

2.2.3 Leap-Frog-Scheme

A definitely better method is the so called *Leap-Frog*-Scheme. One can easily proof that it has an Order of Consistency of two.
In contrast to the explicit Euler-Scheme this method has several advantages. For one it is time reversible, i.e. it is possible to reach any previous point in time from every point later in the trajectory. On the other hand the Leap-Frog-Scheme is symplectic, meaning it conserves the phase space volume from which energy and momentum conservation follows.
However, one disadvantage is that it's only suited for systems in which the acting force exclusively depends on the current position, but not on the velocity of the particle. This would lead to an implicit equation system which is numerically way more expensive to solve.
Thus the differential equation should be of the form

$$\frac{\mathrm{d}^2x^\mu}{\mathrm{d}\tau^2} = \frac{\mathrm{d}u^\mu}{\mathrm{d}\tau} = f^\mu(x^\nu). \tag{2.16}$$

As we already mentioned, the various methods discretize the differential operators differently The Leap-Frog-Scheme uses

$$\begin{aligned} \frac{x^\mu_{i+1} - x^\mu_i}{h} &= u^\mu_{i+\frac{1}{2}} \\ \frac{u^\mu_{i+\frac{1}{2}} - u^\mu_{i-\frac{1}{2}}}{h} &= f^\mu(x^\nu_i). \end{aligned} \tag{2.17}$$

Solving for the new time step yields

$$\begin{aligned} x^\mu_{i+1} &= x^\mu_i + h u^\mu_{i+\frac{1}{2}} \\ u^\mu_{i+\frac{1}{2}} &= u^\mu_{i-\frac{1}{2}} + h f^\mu(x^\nu_i). \end{aligned} \tag{2.18}$$

As we can see, position and velocity are calculated at different times. They are shifted against each other in time by $h = \frac{1}{2}$.
But what if we have a system in which the force depends on the velocity ? Are we stuck with expensive implicit methods ? Fortunately not. We can use the *Boris* - Method.

2.2.4 Boris-Method

This method was invented in 1970 by J.P. Boris[2] and is the standard method for pushing particles in plasma simulations today. We want to solve the Lorentz-Newton equation

$$\frac{v_{i+\frac{1}{2}} - v_{i-\frac{1}{2}}}{h} = \frac{q}{m}\left(\vec{E} + \frac{v_{i+\frac{1}{2}} + v_{i-\frac{1}{2}}}{2} \times \vec{B}\right) \tag{2.19}$$

Boris noticed, that upon defining

$$\begin{aligned} \vec{v}_- &:= v_{i-\frac{1}{2}} + \frac{h}{2}\frac{q\vec{E}}{m} \quad \text{and} \\ \vec{v}_+ &:= v_{i+\frac{1}{2}} + \frac{h}{2}\frac{q\vec{E}}{m}, \end{aligned} \tag{2.20}$$

one can eliminate the electric field. Plugging in (2.20) into (2.19) yields

$$\frac{\vec{v}_+ - \vec{v}_-}{h} = \frac{q}{m}(\vec{v}_+ + \vec{v}_-) \times \vec{B}. \tag{2.21}$$

Following the geometrical analysis in [3] we find

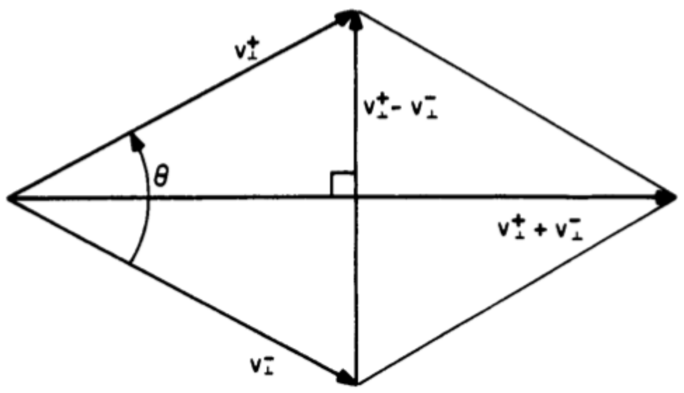

Figure 2.2: Blah Blah

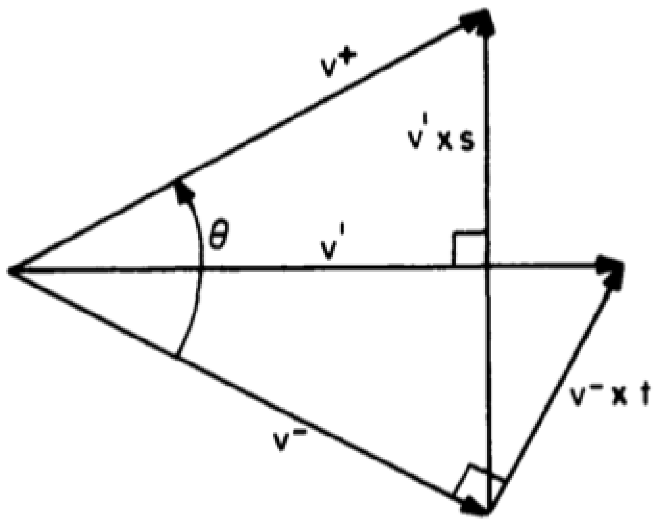

Figure 2.3: Blah Blah

2.2.5 Interpolation of Trajectories

The previously presented methods calculate the particle trajectory solely at discrete points in time $x_i^\mu(\tau)$. Calculating Liénard-Wiechert fields according to equation (2.1) however, requires the intersection point of the trajectory with the backward lightcone of the observation point. In most cases the calculated points of the trajectory are not lying directly on the lightcone, so we need a procedure to calculate the intersection point exactly.
The simplest solution is a linear interpolation between the last point inside and the first point outside the lightcone. Figure 2.4 illustrates the situation.

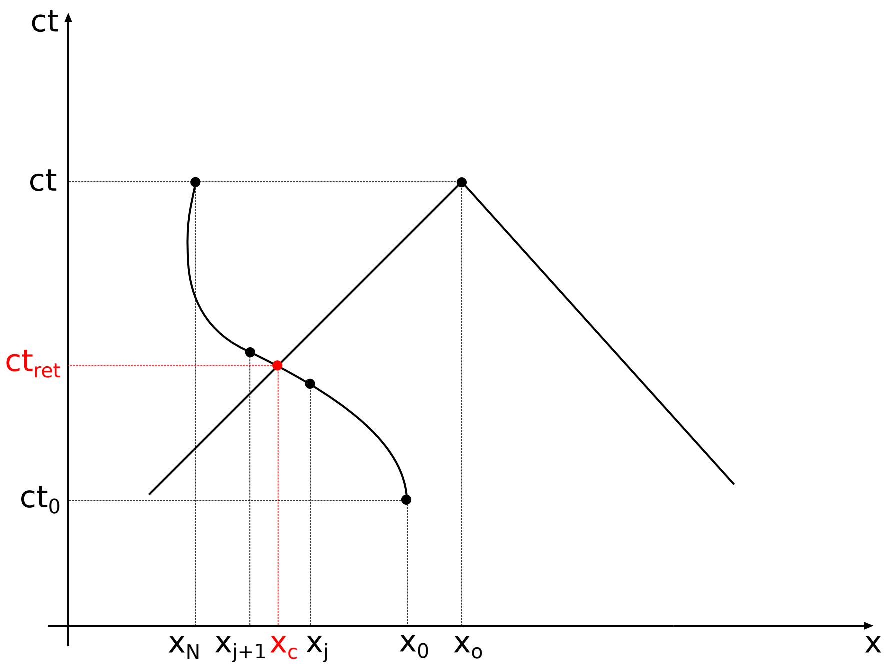

Figure 2.4: Minkowski space showing the particle trajectory with starting point $x_0^\mu(t_0)$ and last point $x_N^\mu(t)$. The observation point $x_o^\mu(t)$ with its backward lightcone is also shown. If we want to calculate the Liénard-Wiechert fields at the observation point $x_o^\mu(t)$, we need the intersection point $x_c^\mu(t_{ret})$ of the trajectory with the backward lightcone.

Thereto let $x_j^\mu \in \mathbb{R}^{3+1}$ be the last point inside and $x_{j+1}^\mu \in \mathbb{R}^{3+1}$ the first point outside the lightcone. Further let $x_c^\mu \in \mathbb{R}^{3+1}$ be the intersection point of interest then we get

$$x_c^\mu = x_j^\mu + \lambda \left(x_{j+1}^\mu - x_j^\mu \right), \tag{2.22}$$

where $\lambda \in [0, 1]$. Due to the finite speed of light the intersection point x_c^μ needs to fulfill

$$|\vec{x}_o(t) - \vec{x}_c(t_{ret})| = c\ (t - t_{ret}) \iff (x_o - x_c)_\mu (x_o - x_c)^\mu = 0. \tag{2.23}$$

Thereby $x_o^\mu \in \mathbb{R}^{3+1}$ denotes the observation point where the fields shall be calculated. Note, that on the left hand side of (2.23) only spatial components of the respective four vectors are used.
Plugging in (2.22) in (2.23) yields

$$\begin{aligned}\lambda^2 (x_{j+1} - x_j)_\mu (x_{j+1} - x_j)^\mu + \lambda\ 2(x_{j+1} - x_j)_\mu (x_j - x_o)^\mu + (x_j)_\mu (x_j)^\mu \\ + (x_o)_\mu (x_o)^\mu - 2(x_j)_\mu (x_o)^\mu = 0.\end{aligned} \tag{2.24}$$

We define

$$\begin{aligned} a &:= (x_{j+1} - x_j)_\mu (x_{j+1} - x_j)^\mu \\ b &:= 2(x_{j+1} - x_j)_\mu (x_j - x_o)^\mu \\ c &:= (x_j)_\mu (x_j)^\mu + (x_o)_\mu (x_o)^\mu - 2(x_j)_\mu (x_o)^\mu. \end{aligned}$$

In general the quadratic equation (2.24) in λ has two solutions

$$\lambda_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

One denotes the intersection point with the backward lightcone, the other one with the forward lightcone. Since $\lambda \in [0, 1]$ we are only interested in the larger one.

$$\lambda_{1/2} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.$$

Plugging in λ in (2.22) gives the desired intersection point.

2.3 Hybrid Fields

In this section we introduce the concept of hybrid fields and how this approach reduces the numerical complexity from N^2 to N.
Usually the numerical complexity of multi particle simulation is N^2, due to the interaction

between each particle. In our case we need N push operations for the particles. One push for each particle. In order to do that, however, we need to solve equation (2.2) and therefore all Liénard-Wiechert fields from the other $N-1$ particles are needed. This results in $N(N-1)$ calculations for each time step, period. If we want to calculate the Liénard-Wiechert fields, we also need to store all positions from all particles, as explained in section 2.2.5. For a few particles such simulations are effortlessly feasible. But with increasing particle numbers such simulations may not just require more and more memory capacity but also become so time consuming that at some point we simply can not do them anylonger. That's where the hybrid field model comes in. Instead of calculating the Liénard-Wiechert fields at every time step for all particles at all grid points we save them onto the grid and propagate them through the grid using Maxwell equations. How that works in detail is explained in the following sections.

2.3.1 Maxwell-Equations

The Maxwell equations are the foundation of the classical electromagnetism and describe how the electric field $\vec{E} \in \mathbb{R}^3$ and the magnetic field $\vec{H} \in \mathbb{R}^3$ are generated by charges and currents respectively and how they evolve over time in space in presence of one another. In homogenous and isotropic media the Maxwell Equations read

$$\begin{aligned}
\epsilon_0\epsilon_r \vec{\nabla}\vec{E} &= \rho \\
\mu_0\mu_r \vec{\nabla}\vec{B} &= 0 \\
\vec{\nabla} \times \vec{E} &= -\mu_0\mu_r \frac{\partial \vec{B}}{\partial t} \\
\vec{\nabla} \times \vec{H} &= \epsilon_0\epsilon_r \frac{\partial \vec{E}}{\partial t} + \sigma \vec{E},
\end{aligned} \tag{2.25}$$

where ϵ_0 and ϵ_r are the vacuum and the relative electric permeability respectively. Same holds for μ_0 and μ_r for magnetic materials. ρ denotes the current density of the electric source and σ the electric conductivity.
To solve the Maxwell equations numerically we need to discretize them first. The most robust and reliable way of doing this is with the *Yee-Algorithm*.

2.3.2 The Yee-Scheme

In this section we talk about how to solve the Maxwell Equations (2.25) and how to propagate the fields on a numerical grid. We thereby focus on the Maxwell Equations in vacuum, i.e. $\rho = \sigma = 0$ and $\epsilon_r = \mu_r = 1$. As can be seen in the Liénard-Wiechert formula (2.1) there is a singularity for the field values at the particle position itself. Thus, we just want to propagate fields far away from the source, which is explained in more detail later.

To push the fields on the grid. i.e solving for the fields at the next time step, we use the Yee - Scheme introduced by *Kane Yee* in 1966.[6]
Figure 2.5 illustrates a so called Yee-Box and the components we need to solve the curl equations of (2.25).

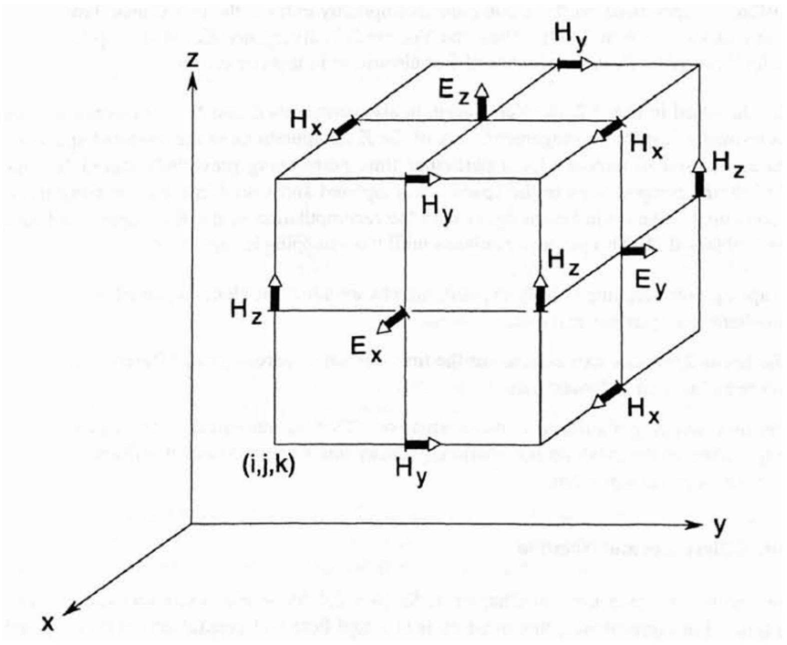

Figure 2.5: An illustration of a Yee-Box which is used to solve the curl Maxwell Equations. Shown are the positions of the electric and magnetic field vector components about a cubic unit cell of the Yee space lattice.The Yee algorithm centers its E and H components in three- dimensional space so that every E component is surrounded by four circulating H components, and every H component is surrounded by four circulating E components.[5]

2.3.3 Near-and Farfields

2.4 Uniaxial Perfectly Matched Layer

List of Figures

Bibliography

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, New York, NY, USA, 1989.

[2] J. P. Boris. Relativistic plasma simulation-optimization of a hybrid code. *Proceeding of Fourth Conference on Numerical Simulations of Plasmas*, November 1970.

[3] A.Bruce Langdon Charles K. Birdsall. *Plasma physics via computer simulation*. IOP, 1991.

[4] Christian Kanzow. *Numerische Mathematik II*. 2005. [Online; accessed 13-Oktober-2016].

[5] A. Taflove and S.C. Hagness. *Computational Electrodynamics: The Finite-difference Time-domain Method*. Artech House antennas and propagation library. Artech House, 2005.

[6] Kane S. Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Trans. Antennas and Propagation*, pages 302–307, 1966.

Declaration

Erklärung:

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

München, Datum der Abgabe

München, 31.03.2017, David Symhoven