TNO Physics and Electronics Laboratory

TNO Defence Research

TD 94 -

0

Phone +31 70 326 42 21

TNO-report FEL-93-B366

copy no. 5

title

A theoretical study on the performance of the FDTD code on a massively parallel computer

[illegible]

TD 94

TNO Defence Research

TN(
Lat TD '14 0766

Oude Waalsdorperweg 63
2597 AK The Hague
P.O. Box 96864
2509 JG The Hague
The Netherlands

Fax +31 70 328 09 61
Phone +31 70 326 42 21

0

TNO-report
FEL-93-B366

copy no.
5

title

A theoretical study on the performance of the FDTD code on a massively parallel computer

author(s):

M.G.E. Brand
L.J. van Ewijk

date:

April 1994

classification

classified by	:	G.A. van der Spek
classification date	:	April 1, 1994

A-1

title	:	Ongerubriceerd
managementuittreksel	:	Ongerubriceerd
abstract	:	Ongerubriceerd
report text	:	Ongerubriceerd

94-31722

no. of copies	:	27
no. of pages	:	28 (excluding RDP and distribution list)
no. of appendices	:	-

All information which is classified according to Dutch regulations shall be treated by the recipient in the same way as classified information of corresponding value in his own country. No part of this information will be disclosed to any party.

The classification designation ONGERUBRICEERD is equivalent to UNCLASSIFIED.

MANAGEMENTUITTREKSEL

Titel	: Een theoretische studie naar de prestaties van de FDTD code op een massief parallelle computer
Auteur(s)	: dr. M.G.E. Brand, ir. L.J. van Ewijk
Datum	: april 1994
IWP-nr.	: 760.3
Rapportnr.	: FEL-93-B366

Voor veel toepassingen in de elektrotechniek is het van groot belang om zeer nauwkeurig de interaktie van elektromagnetische golven met gecompliceerde strukturen te kunnen modelleren. Dit heeft geleid tot de ontwikkeling van een groot aantal modelleringsmethoden en computerprogramma's. De meeste van deze methoden hebben een beperkt toepassingsgebied (door de gekozen fysische benadering) en zijn zeer rekenintensief. De methode die in de literatuur bekend staat als de "Finite Difference Time Domain" (FDTD) methode lijkt beide bezwaren te ondervangen. Bij het Fysisch en Elektronisch Laboratorium TNO is een 3-dimensionaal (3D) FDTD programma ontwikkeld voor de berekening van de radardoorsnede (RCS) van objekten. Behalve voor RCS berekeningen is de methode ook ingezet voor andere elektromagnetische problemen. Hoewel de FDTD methode een lagere complexiteit heeft dan alle andere exacte modelleringstechnieken beperken de vereiste rekentijd en geheugencapaciteit toch drastisch het toepassingsgebied. De methode is echter bij uitstek geschikt om te implementeren op supercomputers met zeer veel parallelle processoren.

In dit rapport presenteren wij een theoretische studie naar de prestaties van een implementatie van de FDTD code op een massief parallelle computer. De hier gepresenteerde complexiteitsanalyse is uitgevoerd voor RCS predictie, maar kan met enige kleine wijzigingen gebruikt worden voor andere elektromagnetische problemen. Bij de analyse is uitgegaan van een data decompositie algoritme.

De analyse is toegepast op een mogelijke implementatie op de Parsytec GCel-3/512 van het "Interdisciplinary Center for Computer based Complex systems research Amsterdam (IC^{3}A)". De Parsytec is geconfigureerd als een twee-dimensionaal rooster van 512 T805 transputers. Als vergelijkingsmateriaal worden de prestaties van een implementatie van de code op een conventionele supercomputer, de Convex C230 van het Fysisch en Elektronisch Laboratorium TNO, gepresenteerd.

Uit deze theoretische studie is gebleken dat de methode zich uitstekend laat parallelliseren met parallelle efficienties van 95%-99%. Dit betekent dat de methode zeer goed schaalbaar is: hoe meer processoren hoe sneller de code. Op de Parsytec zal de code 6 tot 10 keer sneller lopen dan op de Convex, hetgeen de oplossing van veel grotere problemen toelaat in dezelfde rekentijd.
Er wordt aanbevolen om een parallelle versie van het FDTD programma te ontwikkelen en te implementeren op de Parsytec GCel-3/512 van het IC^{3}A, met de bedoeling de hier gepresenteerde analyse te valideren en om te beschikken over een krachtige code voor de oplossing van elektromagnetische problemen. Een dergelijke implementatie kan op eenvoudige wijze overgezet worden op de volgende generatie parallelle computers die rekensnelheden van 100 Gflops tot 1 Tflop binnen bereik zullen brengen voor het jaar 2000.

CONTENTS

1 SCOPE AND OUTLINE

Accurate numerical modelling of full-vector electromagnetic wave interactions with arbitrary structures is difficult. Typical structures of engineering interest have complicated shapes, apertures, cavities and material compositions or surface loadings. Often these features prohibit the use of approximate solution techniques and make it necessary to solve the governing electromagnetic wave equations, the Maxwell equations, exactly. This poses a major computational challenge, because in solving the Maxwell equations one typically needs to sample the electromagnetic waves with ten points per wavelength or more. For real life applications this leads to problems with a tremendous number of unknowns.
Traditionally, these computations were done in the frequency domain. Until recently the primary computational approach for detailed modelling of electromagnetic wave scattering involved frequency-domain integral equations. From a supercomputing perspective this involved the need to achieve efficient solutions of dense complex valued systems of tens of thousands of linear equations. The solution effort and memory requirements involved with these methods has put a severe limitation on their applicability.
Directly solving the Maxwell equations in the time-domain is relatively new in the electromagnetic community but is becoming increasingly popular. It allows supercomputing solutions of tens of millions of electromagnetic field unknowns without any matrix manipulations. One of the most popular approaches to time domain modelling is the Finite Difference Time Domain (FDTD) method. The FDTD method is very simple in concept and execution. However, it is remarkably robust, providing highly accurate modelling predictions for a wide variety of electromagnetic wave interaction problems.
At the Physics and Electronics Laboratory a FDTD code has been developed to predict the Radar Cross Section (RCS) of arbitrary complex objects. This code was implemented on a Convex C230, which is a conventional supercomputer with three vector processors. With this computer scattering from objects with maximum dimensions of only 5 wavelengths can be analyzed.
However, the FDTD method seems naturally suited for large scale processing on massively parallel processors (MPP's).
The advent of these MPP's promises computational speeds up to 1 Tflop before the end of the century. This should permit FDTD and similar approaches to model the dynamics of hundreds of millions to billions of field unknowns, allowing for instance the calculation of the RCS of entire F-16 size jet fighters up to 10 GHz, the calculation of the radiation pattern of a realistic phased array antenna or the fields induced in a patient during a hyperthermia treatment.

In this report we present a theoretical study of the performance of an implementation of the FDTD code on a MPP. The MPP we have in mind is the Parsytec GCel-3/512 of the "Interdisciplinary Center for Computer based Complex systems research Amsterdam (IC^{3}A)", which is configured as a two-dimensional grid of 512 T805 transputers. The theoretical analysis presented in this report is, however, completely general and can be used for every MPP. The analysis is carried out for RCS prediction, but can, with minor adaptions, be used for other electromagnetic problems.
In chapter 2 we discuss the FDTD method, presenting all the issues which have to be adressed in a parallel implementation. In chapter 3 a time-complexity analysis of the FDTD method on a MPP is discussed and applied to an implementation on the Parsytec GCel-3/512. In chapter 4 a comparison is presented between run-times on the Parsytec and the Convex. Chapter 5 consists of conclusions and recommendations.

We would like to acknowledge A.Hoekstra of the IC^{3}A for introducing us to the concepts of time-complexity analysis and the IC^{3}A for providing us with an account on the Parsytec.

2 DESCRIPTION OF THE FDTD METHOD FOR RCS PREDICTION

2.1 Introduction

The Radar Cross Section of an object is defined as:

$$\mathrm{RCS} = \lim_{r\to\infty} 4\pi r^2 \frac{|\underline{E}^s|^2}{|\underline{E}^i|^2} = \lim_{r\to\infty} 4\pi r^2 \frac{|\underline{H}^s|^2}{|\underline{H}^i|^2} \tag{2.1}$$

where r is the distance from object to observer, $\underline{E}^i$ and $\underline{H}^i$ are the electric and magnetic fields incident at the object, and $\underline{E}^s$ and $\underline{H}^s$ are the observed scattered electric and magnetic fields. Throughout this report we will use the convention that underlined quantities are vector quantities. To calculate the RCS of an object one has to calculate the fields scattered by that object when illuminated by an incident plane wave. The equations which govern these fields are the Maxwell curl equations:

$$\begin{aligned}\mu(\underline{x})\partial_t \underline{H}(\underline{x},t) &= -\nabla\times\underline{E}(\underline{x},t),\\ \varepsilon(\underline{x})\partial_t \underline{E}(\underline{x},t) &= \nabla\times\underline{H}(\underline{x},t)-\sigma(\underline{x})\underline{E}(\underline{x},t),\end{aligned} \tag{2.2}$$

which have to be solved subject to the usual boundary conditions at medium interfaces. In eqs.(2.2a) and (2.2b) ε is the electrical permittivity, σ the electrical conductivity, μ the magnetic permeability and ∂_t denotes partial differentiation with respect to t.
Eqs.(2.2) constitute a system of six coupled scalar equations with six unknowns : E_x, E_y, E_z, H_x, H_y, H_z. In this report we will discuss the Finite Difference Time Domain (FDTD) method for the direct numerical solution of these equations.

2.2 The FDTD algorithm

The FDTD method was first presented by Yee[1] and developed further by Taflove[2]. In order to solve the system of equations (2.2) Yee introduced a set of finite difference equations. Following Yee's notation, we denote a space point in a rectangular lattice as

$$(i,j,k) = (i\delta,j\delta,k\delta) \tag{2.3}$$

and any function of space and time as

$$F^n(i,j,k) = F(i\delta,j\delta,k\delta,n\Delta t) \tag{2.4}$$

where $F^n(i,j,k)$ is a field sample at position $(i\delta,j\delta,k\delta,n\Delta\tau)$, δ is the lattice space increment and Δt is the time increment. Yee used centered difference approximations for the time and space derivatives that are both simply programmed and second order accurate (the local truncation error is of order 2) in the space and time increments respectively:

$$\partial_x F^n(i,j,k)= \frac{F^n(i+½,j,k)-F^n(i-½,j,k)}{\delta} +O(\delta^2) \tag{2.5a}$$

$$\partial_t F^n(i,j,k)= \frac{F^{n+1/2}(i,j,k)-F^{n-1/2}(i,j,k)}{\Delta t} +O(\Delta t^2) \tag{2.5b}$$

To achieve the accuracy of (2.5a) and to realize all of the required space derivatives of (2.2), the components of $\underline{E}$ and $\underline{H}$ are positioned about a unit cell of the lattice as shown in Fig. 2.1.

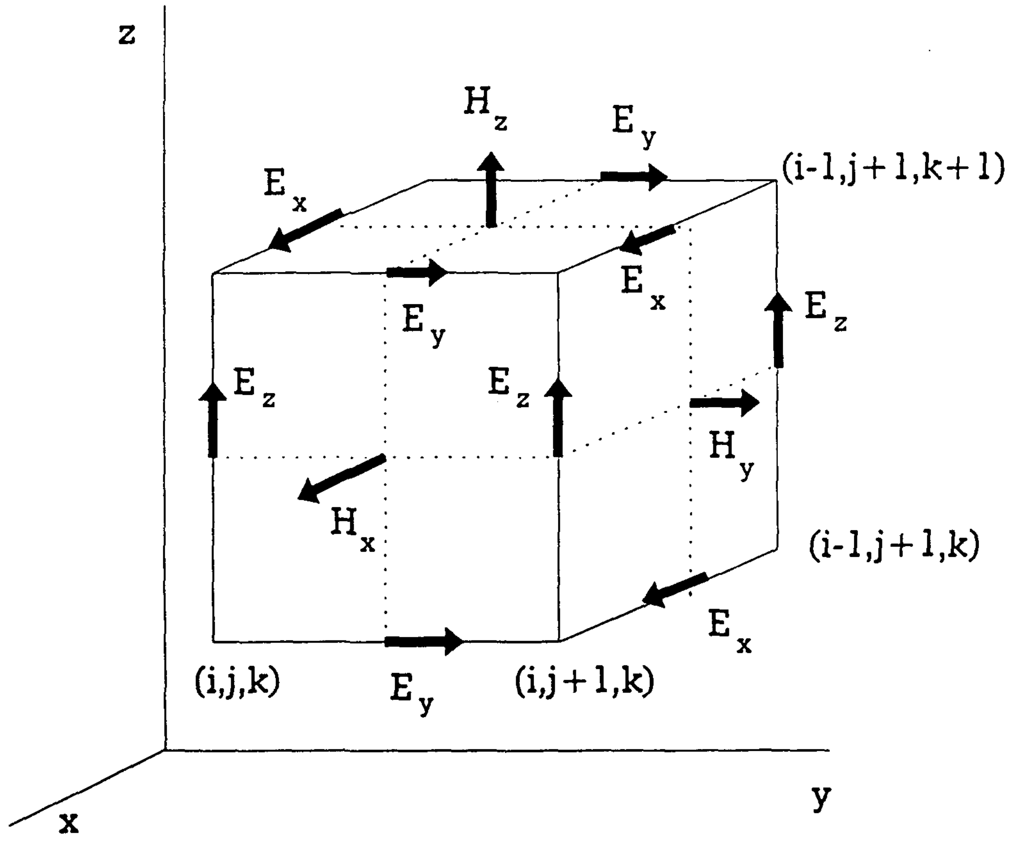


Fig. 2.1 A Finite Difference Time Domain unit lattice cell

This leads to a so-called "staggered grid" in which the E and H grids are interleaved in space. To achieve the accuracy of (2.5b) the E and H fields are evaluated at alternative half time-steps. This is the so called "leap-frog" scheme.
Substituting the expressions (2.5a and b) into (2.2) one obtains a finite-difference analog of the Maxwell equations. The following are sample finite difference time-stepping relations for a magnetic and an electric field component:

$$\begin{aligned} E_x^{n+1}(i+½,j,k) &= C_1(m)\, E_x^n(i+½,j,k) \\ &+ C_2(m)\{ H_z^{n+1/2}(i+½,j+½,k) - H_z^{n+1/2}(i+½,j-½,k) \\ &\quad -H_y^{n+1/2}(i+½,j,k+½) + H_y^{n+1/2}(i+½,j,k-½) \} \end{aligned} \qquad (2.6a)$$

$$\begin{aligned} H_x^{n+1/2}(i,j+½,k+½) &= H_x^{n-1/2}(i,j+½,k+½) \\ &+ C_3(m)\{ E_y^n(i,j+½,k+1) - E_y^n(i,j+½,k) \\ &\quad -E_z^n(i,j+1,k+½) + E_z^n(i,j,k+½) \} \end{aligned} \qquad (2.6b)$$

with

$$C_1(m) =(2\varepsilon(m)-\Delta t\sigma(m)) / (2\varepsilon(m)+\Delta t\sigma(m))$$
$$C_2(m) = 2\Delta t / (2\delta\varepsilon(m)+\delta\Delta t\sigma(m))$$
$$C_3(m) = \Delta t / (\delta\mu(m))$$

in which m = media(i,j,k) is the type of medium at a field component location. To ensure stability of the time stepping algorithm, Δt is chosen to satisfy the inequality [2]

$$\Delta t \leq \delta/(c_{max}\sqrt{3})$$

where c_{max} is the maximum wave phase velocity within this model. To obtain accurate results δ needs to be chosen small compared to the wavelength, usually $\delta \leq \lambda/10$. Also δ has to be small compared to the dimensions of the object, because curved boundaries are transformed into a staircase approximation.

At each time step the system of equations to update the field components is fully explicit and very simple. The new value of a E-field vector component at any lattice point depends only on its previous value and on the previous values of the surrounding H-field components and vice versa.

2.3 The computational domain

The FDTD model is by necessity formulated as a boundary and initial value problem on a finite portion of space: the computational domain, usually a rectangular parallelepiped (brick) with dimensions $0 \le x \le a\delta$, $0 \le y \le b\delta$ and $0 \le z \le c\delta$ (a,b and c integers). Therefore the solution of scattering and radiation problems in unbounded (open) spatial regions requires a mechanism for coupling the closed region numerical solution to exterior space. This mechanism serves to enforce the radiation condition which prescribes the behaviour of electromagnetic fields at infinity. In general the computational domain consists of an arbitrary scattering target surrounded by a finite portion of free space enclosed by the surface on which the radiation boundary condition is enforced.

2.3.1 The Absorbing Boundary Condition

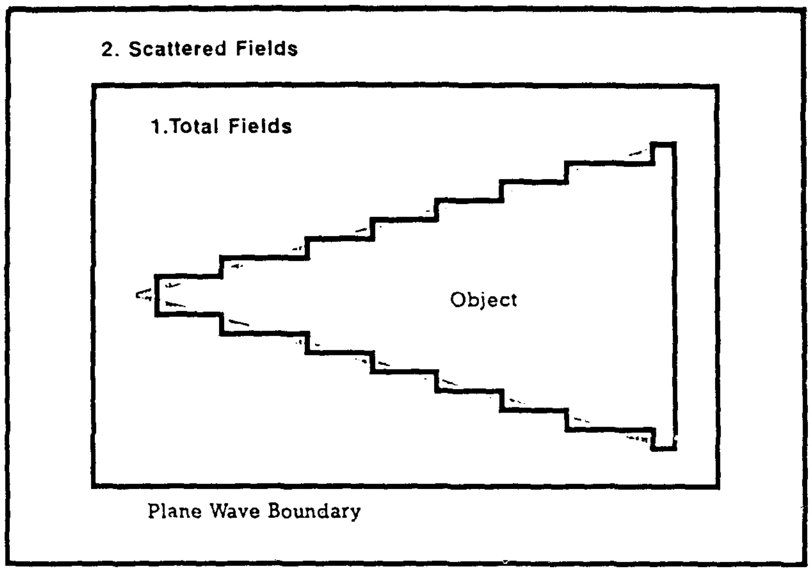
The absorbing boundary condition or radiation boundary condition which is implemented is based upon the so called "one-way wave equation". This type of boundary conditions was introduced by Enquist and Majda[3]. The one way wave equation is a partial differential equation which permits wave propagation only in certain directions. Enforcing a numerical approximation of this "one-way wave equation" at the boundary of the computational domain produces a highly accurate absorbing boundary. In the current FDTD code a second-order finite difference approximation of the one way wave equation which was proposed by Mur[4] is implemented.
A sample expression for a field component at the boundary x=0 is given in (2.7).

$$\begin{aligned}
U^{n+1}(0,j,k) = & -U^{n-1}(1,j,k) + A1\,(U^{n+1}(1,j,k) + U^{n-1}(0,j,k)) \\
& + A2\,(U^{n}(0,j,k) + U^{n}(1,j,k)) \\
& + A3\,(U^{n}(0,j+1,k) + U^{n}(0,j-1,k) + U^{n}(1,j+1,k) + U^{n}(1,j-1,k) \\
& \quad + U^{n}(0,j,k+1) + U^{n}(0,j,k-1) + U^{n}(1,j,k+1) + U^{n}(1,j,k-1) \\
& \quad - 4\,U^{n}(0,j,k) - 4\,U^{n}(1,j,k))
\end{aligned} \tag{2.7}$$

in which A1,A2 and A3 are simple constants determined by δ and Δt.

2.3.2 Lattice regions and the plane wave source condition

As shown in fig.2.2 the computational domain is divided into two distinct regions, separated by a rectangular surface which serves to connect the fields in each region.



Absorbing Boundary Condition

Fig. 2.2 The global geometry of the FDTD lattice

Region 1 of the lattice is denoted the total-field region, region 2 the scattered-field region. The object is positioned in region 1 where all field quantities are comprised of the sum of the incident wave and the scattered field. In region 2 all field quantities are comprised only of the scattered field.

The rectangular planes that form the boundary between regions 1 and 2 contain E and H field components. These field components are updated according to equations (2.6) and subsequently corrected to maintain the two distinct regions. A Typical FDTD computation at these boundary points is:

$$E_x^{n+1}(i+½,j_o,k) = E_x^{n+1}(i+½,j_o,k)\big|_{eqn(2.6a)} + C_2(m)\, H_z^{i,n+1/2}(i+½,j_o-½,k) \qquad (2.8)$$

Here $E_x^{n+1}(i+½,j_o,k)$ is the usual FD-TD value of the total Ex component evaluated at point $(i+½,j_o,k)$ and time step n+1. The superscript "i" denotes the known incident field component value. These computations assure consistency of the subtraction operations of field components across the Region 1 / Region 2 boundary. In effect, total-field quantities are always subtracted from similar total-field quantities; the same goes for scattered-field quantities. This enforcement of consistency serves to precisely connect the two regions. Further, the inclusion of arbitrary values of Ei and Hi in the consistency relations permits the specification of any desired plane wave of arbitrary angle of incidence and arbitrary polarisation.

2.3.3 The RCS calculation

Time stepping is continued until the desired sinusoidal steady state behaviour is achieved. The time needed to reach this state is mainly depending on the object's electrical size. The total number of time steps needed for a wave travelling at the speed of light to make two complete front-to-back-to-front traverses is normally sufficient to reach the steady state. Typical times can be found in [5]. After reaching the sinusoidal steady state the magnitude and phase of the components of the near-fields have to be determined. Due to the fact that the FD-TD method causes some fields to have a nonphysical dc offset, an algorithm is used that determines the maximum and minimum of a field component. The magnitude is calculated as (max-min)/2 and the phase is the phase of the maximum minus an arbitrary fixed phase. To achieve a greater accuracy multiple cycles can be treated this way and an average result can be calculated.
The far field and RCS data can be obtained, in principle, by solving an integral equation for the induced currents on the surface of the object. For complex objects this can be very difficult. A good alternative is to set up an equivalent problem. The object is surrounded with an arbitrary, closed, virtual boundary, the surface Sa, on which the near-field data is obtained via the FD-TD method. To obtain the RCS these near fields are transformed into far fields using the Chu-Stratton equations:

$$\underline{E}^s(\underline{x}) = 1/4\pi\int_{S_a} ds\ \{i\omega\mu(\underline{n}'\times\underline{H})g_0+\underline{n}'\times\underline{E}\times\underline{\nabla}'g_0+(\underline{n}'\cdot\underline{E})\underline{\nabla}'g_0\} \qquad (2.9a)$$

$$\underline{H}^s(\underline{x}) = -1/4\pi\int_{S_a} ds\ \{i\omega\varepsilon(\underline{n}'\times\underline{E})g_0-\underline{n}'\times\underline{H}\times\underline{\nabla}'g_0-(\underline{n}'\cdot\underline{H})\underline{\nabla}'g_0\} \qquad (2.9b)$$

in which $\underline{n}'$ is an outward pointing normal and g_0 the greens function of free space. The far fields are obtained by taking the observation point $\underline{x}$ to infinity. The RCS is calculated from these farfields by applying (2.1)

3 TIME COMPLEXITY ANALYSIS OF THE FDTD CODE

In chapter 2 the description of the FDTD code is given.
In this chapter we will perform a time complexity analysis of this code when implemented on a parallel machine. The purpose of such an analysis is to assess the effectiveness of using a parallel machine, expressed as a numerical efficiency defined as:

$$\varepsilon = \frac{T_{seq}}{p\, T_{par}} \qquad (3.1)$$

In this equation T_{seq} represents the time needed by the program when run on one processor, T_{par} the time when run on a parallel machine and p the number of processors used. The numerical efficiency reflects the speed-up that can be achieved by increasing the number of processors.
The analysis given here will be generally applicable to any parallel machine, but we restrict ourselves to an implementation on the Parsytec GCel-3/512 when specific numbers are necessary. Furthermore data decomposition is used, see Fox [7] and the 3-D computational domain will be mapped onto the 2-D processor network by assigning all gridcells in one of the cartesian dimensions to the same processor.

3.1 The FDTD code on a parallel machine

From a sample equation of the computations that are performed with the FDTD technique we can find that 6 floating operations are needed to update one field component.
In every time step we need to compute several new field components in every gridcell. Figure 3.1 shows the way the field components are distributed over each gridcell. It is clear that it is not necessary to compute all values needed for a complete update of a gridcell in that specific cell itself. Many values are shared between more than one cell.

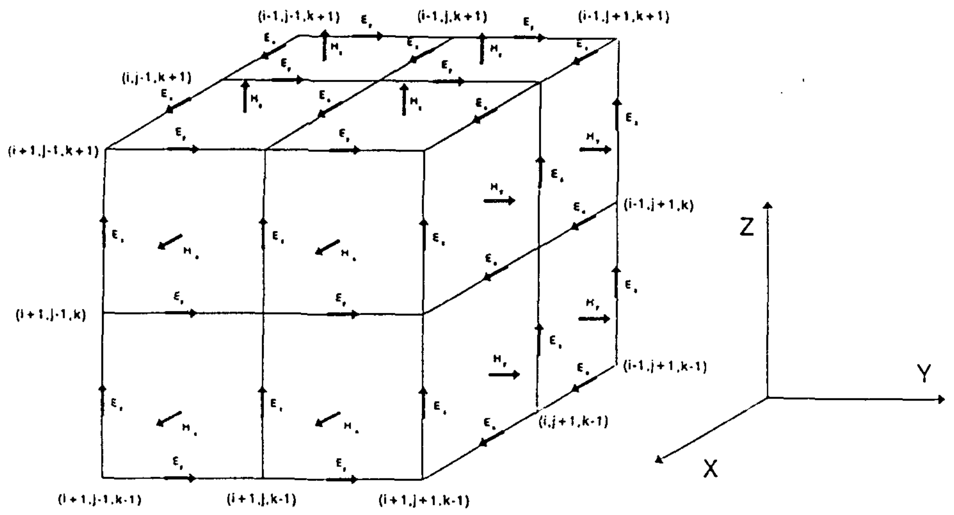


Fig. 3.1 Distribution of field components over FDTD gridcells

In figure 3.2 we can see that the minimum number of field components that must be computed in a gridcell equals 6. This implies that the update of a gridcell costs 36 floating operations.

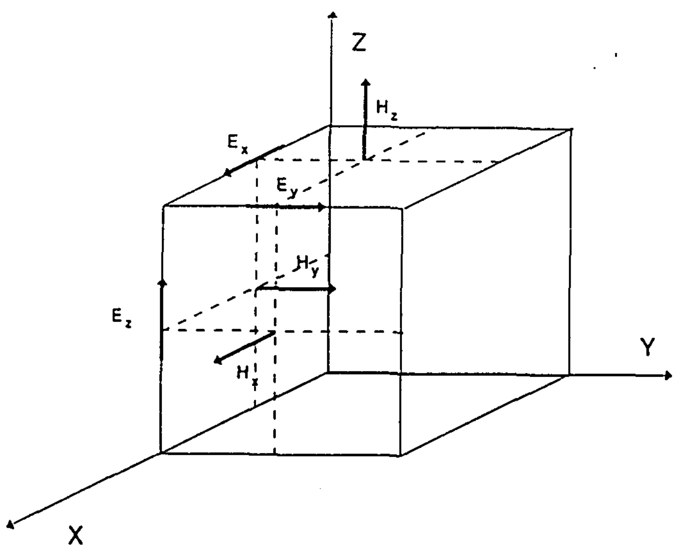


Fig. 3.2 The field components that must be updated in a gridcell

In the following we use the same notation as is used in Hoekstra [6].

The computational work to be done consists of a number of cycles, or time steps as they are called. Each cycle can be divided into two parts:

- computation of the field values in all gridcells,
- exchange of data between processors.

After each cycle the processors are synchronized and the next cycle is started.

The time needed by the parallel program depends on the number of processors, the number of floating operations to be performed and on a number of system parameters. These system parameters are:

-	τ_{calc}	time needed for one floating operation,
-	τ_{comm}	time needed to transfer one real number to a neighbouring processor,
-	$\tau_{startup}$	time needed to establish a communication link,
-	topology	the order of the processors in the network.

In order to find an expression for the total time needed for the computation we start with the time needed for the update of the field components. We use a grid of a x b x c gridcells divided over p_1 x p_2 processors, as shown in figure 3.3.

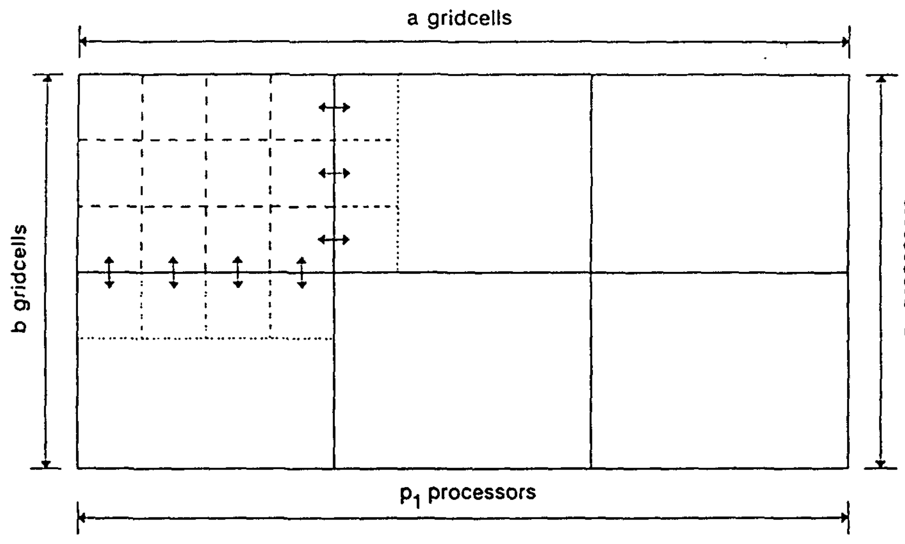


Fig. 3.3 **The FDTD grid with a x b x c gridcells divided over p_1 x p_2 processors. The c direction is the direction out of the paper**

This means that every processor ideally computes the values for $a/p_1 \times b/p_2 \times c$ gridcells. It is, however, implementation dependent whether a/p_1 and b/p_2 are integer numbers. If this is not the case we must consider the worst case situation where some processors compute the values for more gridcells than other processors. This can be written as in the following equation:

$$T_{par} = 36 \left\lceil \frac{a}{p_1} \right\rceil \left\lceil \frac{b}{p_2} \right\rceil c\, \tau_{calc} \tag{3.2}$$

in which T_{par} is the time needed by the parallel program and $\lceil x \rceil$ denotes the ceiling function of x. By using T_{seq}, as defined earlier, equation (3.2) can be written as:

$$T_{par} = \frac{T_{seq}}{p} + 36 \left(\left\lceil \frac{a}{p_1} \right\rceil \left\lceil \frac{b}{p_2} \right\rceil - \frac{ab}{p_1 p_2} \right) c\, \tau_{calc} \tag{3.3}$$

in which $p_1 x p_2$ equals the total number of processors p and T_{seq} can be expressed as 36 abc τ_{calc}. The second term in equation (3.3) shows the effect of dividing the gridcells over the available processors. If this can't be done evenly some processors have to handle more gridcells than other processors, which is called load imbalance.

The load imbalance can be avoided by choosing the correct implementation, but in general this term is present.

3.2 Non-parallel parts of the computation

Some parts of the computation cannot be performed completely in parallel. In the grid three boundaries can be pointed out at which some extra computations are necessary (see section 2.3). These boundaries are:

- the outer perimeter of the grid,
- the source boundary,
- the monitor boundary.

At the outer perimeter of the grid one has to take care that the fields comply with the radiation condition in order to avoid nonphysical reflections of the fields, see section 2.3.1.

At the source boundary the incident fields are initialised as a plane wave, see section 2.3.2.

The monitor boundary is used to determine the final results, after all time steps are done. During the last 5 steps the fields on this boundary must be monitored to determine amplitude and phase of the scattered field, see section 2.3.3.

The extra work that must be done at these boundaries is:

- 20 floating operations at the outer perimeter,
- 20 floating operations at the source boundary,
- 3 floating operations at the monitor boundary.

It must be noted that the extra work at the monitor boundary is only necessary during the last 5 time steps, but in this analysis it is supposed to be done during all steps. This is only a small deviation from reality because the number of operations needed at this boundary is small compared to the other boundaries. The computations at these boundaries are taken care of by the appropriate processors and are done in parallel by those processors. The other processors, however, must wait until this work is done. That is the reason that this part of the update is not done completely in parallel.

The time needed for these computations can be expressed as:

$$T_{np} = 43 \left(\left\lceil \frac{a}{p_1} \right\rceil \left\lceil \frac{b}{p_2} \right\rceil \right) c \, \tau_{calc} \qquad (3.4)$$

in which we assume that all boundaries have the same number of gridcells. Because the source and monitor boundary are only slightly smaller than the outer boundary the time for the above mentioned non-parallel computation will be a little smaller than given by equation (3.4).

The time needed for all computations in one time step can now be found by summing equation (3.3) and (3.4).

3.3 Time needed for communication

When defining the computational domain of the problem one must be careful that this grid can be mapped onto the processor network of the parallel machine that is used. The Parsytec at the University of Amsterdam is equipped with 16 x 32 processors in a rectangular grid. One natural choice for mapping the computational domain onto this network is to assign all gridcells in one cartesian direction of the grid to the same processor, as shown in fig 3.3. With this mapping it is only necessary to exchange data between two adjoining processors, as is clear from the equation for the update of the fields and fig 3.3.

Data transfer is necessary after each update and for every gridcell at a boundary of two processors two field components must be imported from the next processor. The communication is done for the complete processor network in one direction at the time. For the directions we use the terms North, East, South and West, denoting the four sides of the rectangular network. First all data is transferred from East to West in the processor network, then from West to East, from North to

South and finally from South to North. The reason for this kind of communication is that it is easy to implement in Parix, the operating system that is used.
For the communication from East to West we know that there are p_2 processors that must send and receive data. Each processor sends (and receives) $2\lceil b/p_2 \rceil$ field components of 4 bytes. In an equation this reads:

$$T_{sendE\text{-}W} = \tau_{startup} + 2 . \lceil \frac{b}{p_2} \rceil 4 . c . \tau_{comm} \tag{3.5}$$

where $\tau_{startup}$ is the time to set up a communication link between two adjacent processors and τ_{comm} the time needed to transfer 1 byte over this link. Every processor, except those at the edges of the computational domain, must send data from East to West and from West to East. Furthermore the same time is needed to receive data from the East and from the West. So for the East-West communication the total time will be 4 times the time given in eq (3.5) (we don't consider the fact that processors at the edges have to do less communication), resulting in:

$$T_{commE\text{-}W} = 4 \{ \tau_{startup} + 2 . \lceil \frac{b}{p_2} \rceil 4 . c . \tau_{comm} \} \tag{3.6}$$

For the North-South communication we follow the same reasoning and arrive at:

$$T_{commN\text{-}S} = 4 \{ \tau_{startup} + 2 . \lceil \frac{a}{p_1} \rceil 4 . c . \tau_{comm} \} \tag{3.7}$$

3.4 Total time needed

By summing eq (3.3), (3.4), (3.6) and (3.7) we arrive at an expression for the total time that is needed for one time step. This equation then reads:

$$\begin{aligned} T_{par} = {} & 36 \lceil \frac{a}{p_1} \rceil \lceil \frac{b}{p_2} \rceil c\, \tau_{calc} + \\ & 43 (\lceil \frac{a}{p_1} \rceil + \lceil \frac{b}{p_2} \rceil) c\, \tau_{calc} \\ & 8\, t_{startup} + 32 (\lceil \frac{a}{p_1} \rceil + \lceil \frac{b}{p_2} \rceil) c\, \tau_{comm} \end{aligned} \tag{3.8}$$

Equation (3.8) represents a simple model for the implementation of the FDTD method on a parallel machine, so this equation can merely be used to obtain an impression of the computational cost of the method. For a more exact estimate a more extensive analysis must be done.

We can substitute the expression for T_{seq}, as given earlier, and for T_{par} from equation (3.8) in equation (3.1) and compute the numerical efficiency for various parameters. To do this it is also necessary to supply some values for the system parameters of the machine we are using. At the Parsytec GCell-3/512 those values are approximately [9]:

- τ_{calc} $= 1 \cdot 10^{-6}$ s (for 1 floating operation),
- τ_{comm} $= 1 \cdot 10^{-6}$ s (for the communication of 1 byte),
- $\tau_{startup}$ $= 1.5 \cdot 10^{-4}$ s (to startup a link for communication).

The numerical efficiency for various numbers of gridcells and processors is given in fig 3.4. It is clear that the efficiency of this method is very close to unity as long as the number of gridcells to be handled by each processor, also called the grainsize, is large. Of course this increases the total computational time as well, this will be discussed in chapter 4.

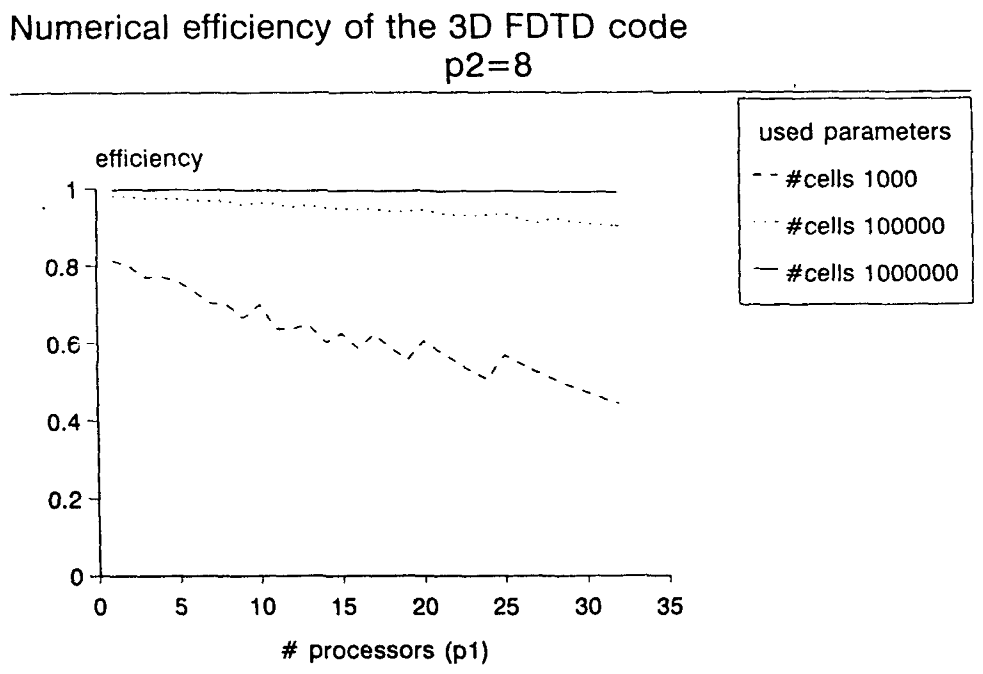


Fig. 3.4 Numerical efficiency for various numbers of gridcells and processors

4 COMPUTATION TIME AND MEMORY USAGE

4.1 Memory usage of the FDTD code

When implementing the FDTD method on any computer one has to consider memory requirements because the number of gridcells grows rapidly with increasing size of the computational domain. Besides, a substantial amount of values must be known in each gridcell to perform the update. In each gridcell the values of 3 components of the electric field, 3 components of the magnetic field and 4 material properties (permittivity, permeability, electric and magnetic conductivity) must be known. Furthermore, each processor must also store the values from communication with the neighbouring processors. This amounts to three field components for each plane of the gridcell for which the processor doesn't store those values itself, see fig 3.2. It must be noted that at the borders of the computational domain these values do not originate from communication with other processors. Nevertheless the same amount of storage space must be available.

Summarizing the memory requirement for each gridcell is:

- 10 numbers of 4 bytes,
- 3 numbers of 4 bytes for each plane at the border of the subdomain of the processor.

Besides, each processor must also store the executable code and reserve some memory for the operating system. This is estimated to be 800 kB.

This results in:

$$M_p = 40\left\lceil\frac{a}{p_1}\right\rceil\left\lceil\frac{b}{p_2}\right\rceil c + 12\left\{\left\lceil\frac{a}{p_1}\right\rceil c + \left\lceil\frac{b}{p_2}\right\rceil c + \left\lceil\frac{a}{p_1}\right\rceil\left\lceil\frac{b}{p_2}\right\rceil\right\} + 8 \cdot 10^5 \qquad (4.1)$$

With eq (4.1) it is possible to compute the number of bytes of memory needed by each processor, but also to determine the minimum amount of memory needed by the FDTD code on a computer with only one processor. (This remark holds for parallel computers as well, as long as task decomposition is used.)

The total amount of memory can easily be found by multiplying the result of eq (4.1) with the number of processors. This latter number is, however, of little interest. One could define for instance the memory efficiency as the minimum amount of memory divided by the amount of memory used by the parallel computer, but this is only relevant when the efficient use of memory is critical.

The memory requirement is computed for a gridsize of 300x300x400 cells as a function of the number of processors. The result is shown in fig 4.1 with the drawn line the memory per processor in megabytes. When the full network of available processors is used, 32x16 processors, the required memory is 3.9 MB per processor. Theoretically this is the maximum possible computational domain that can be defined at this machine.

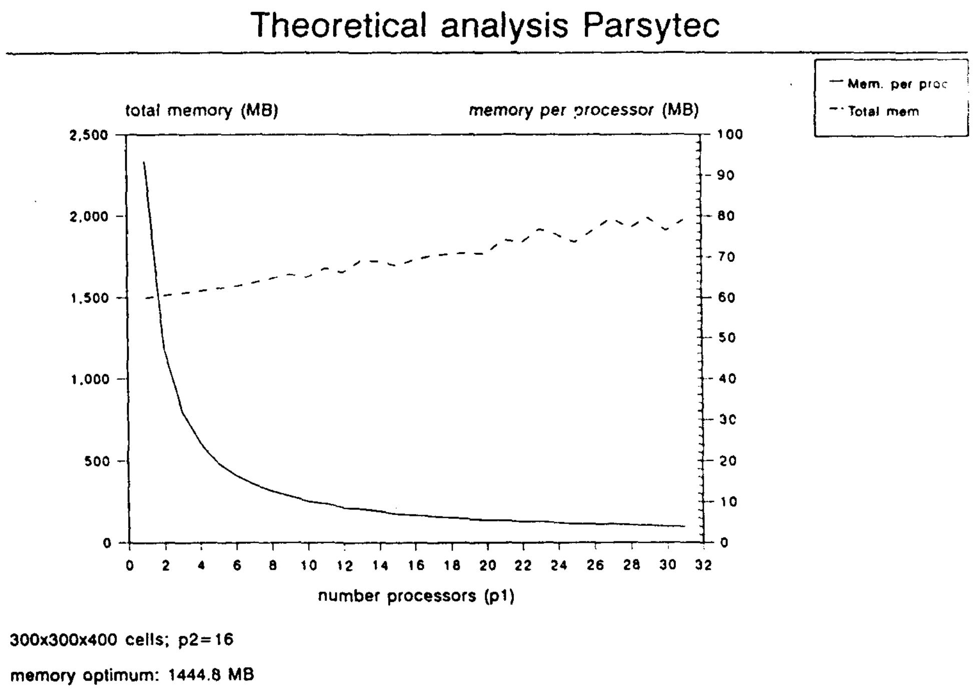


Fig. 4.1 The memory requirement computed for a gridsize of 300 x 300 x 400 cells

4.2 Runtime of the FDTD code

This section will give a comparison of the runtime needed by the FDTD code, as elaborated upon in this report, when this code is implemented on the Parsytec GCel-3/512 and on the Convex C-230. For this comparison we will use eq (3.8) to estimate the runtime needed for one time step by the Parsytec. For the time needed by the Convex we will use the following expression, empirically determined by v. Gennip [8]:

$$T = 1.3*(1.7 \cdot 10^{-6}\, d^3 + 2.7 \cdot 10^{-5}\, d^2 + 2.9 \cdot 10^{-3}\, d) \qquad (4.2)$$

where d represents the cube root of the number of gridcells.

In order to make a thorough comparison we will use three gridsizes. The first grid will be 30x30x40 cells, making it possible to define objects with a size of about one wavelength in all dimensions. These kind of objects are mostly interesting from scientific point of view. The next grid will be 120x120x200 cells, which is the maximum possible grid at the Convex at this moment restricted by the amount of available memory, see section 4.1. The third grid under consideration will be 300x300x400 cells, making it possible to define objects with a size of about 10 wavelengths in all dimensions. Objects of this size are considered quite large in the field of RCS computations with the finite difference method.

When applying eq (3.7) we can vary for instance the number of processors used. Furthermore we must set values for various system times. We use the same values as we did in the assessment of the numerical efficiency in chapter 3:

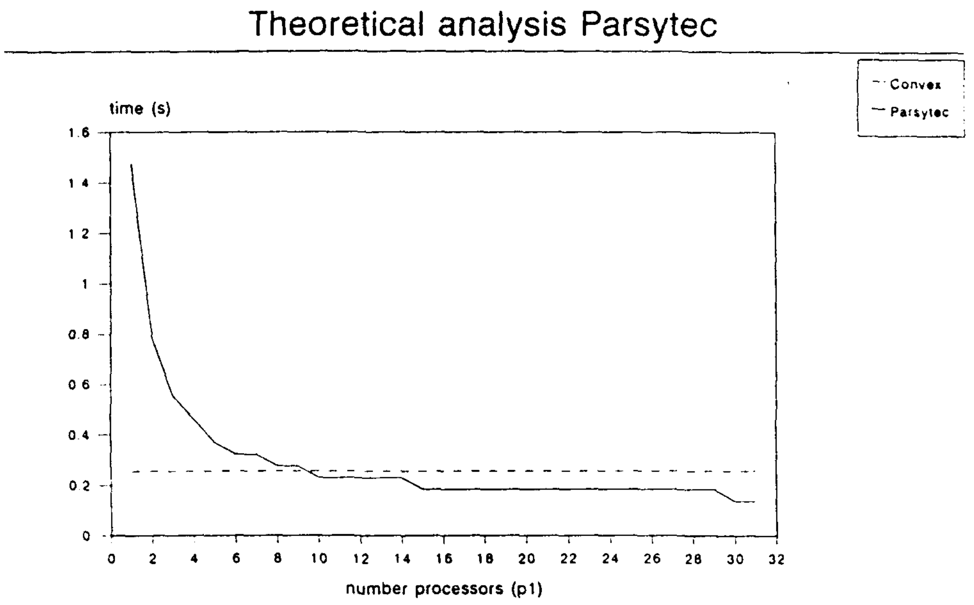
- τ_{calc} = 1.10^{-6} s (for 1 floating operation),
- τ_{comm} = 1.10^{-6} s (for the communication of 1 byte),
- $\tau_{startup}$ = $1.5.10^{-4}$ s (to startup a link for communication).

The expected runtimes are estimated for various sizes of the processor network used. In all cases the computations are done as function of the number of processors p_1, see fig 3.3, and for p_2 equal to 1, 4, 8 and 16. A representative collection of the results is given in fig 4.2 to 4.5. In all graphs the vertical axis displays the time needed to compute all the fields in the grid after one time step, including the communication with the neighbouring processors. The horizontal axis displays the number of processors (p_1) used by the Parsytec. A footnote states the size of the grid and the number of processors p_2.

Fig 4.2 shows that for a grid of 30x30x40 cells the Convex needs 0.25 s for each update. The Parsytec is faster when 10 processors in a row are used. Also the effect of load imbalance is very clear from this graph. If this computation is done with 20 processors instead of 10 (in a row) one only gains a few tens of seconds per update.

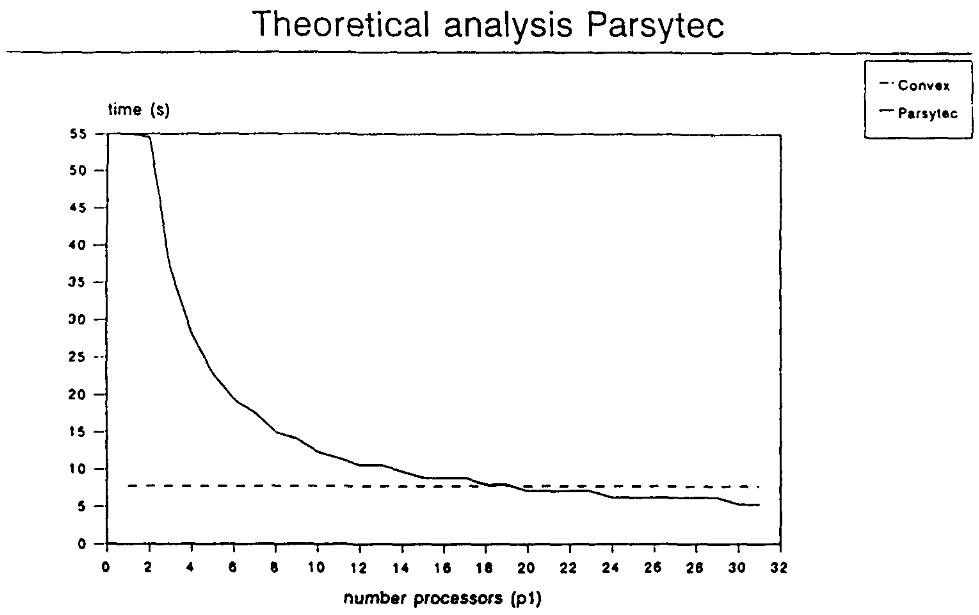
Fig 4.3 and 4.4 give the estimates for the maximum grid size possible on the Convex. The time on the Convex is 7.7 s. The Parsytec needs 18x1 processors (fig 4.3) or 2x8 processors (fig 4.4) to perform an update in the same time. This obviously shows that using a processor network as in fig 4.4 is more efficient than the one in fig 4.3.

Fig 4.5 finally is the result for the 300x300x400 grid. From the memory analysis in section 4.1 we know that it is only possible to use this grid on the maximum available number of processors. The time needed for one update is slightly less than 4 s, which is not yet unacceptably high.



30x30x40 cells; p2=1
memory Convex : 5
memory optimum: 2.3

Fig. 4.2 Time needed for one update for a grid of 30 x 30 x 40 cells as a function of the number of processors p_1; $p_2=1$



120x120x200 cells; p2=1
memory Convex : 238.8 MB
memory optimum: 116.7 MB

Fig. 4.3 Time needed for one update for a grid of 120 x 120 x 200 cells as a function of the number of processors p_1; $p_2=1$

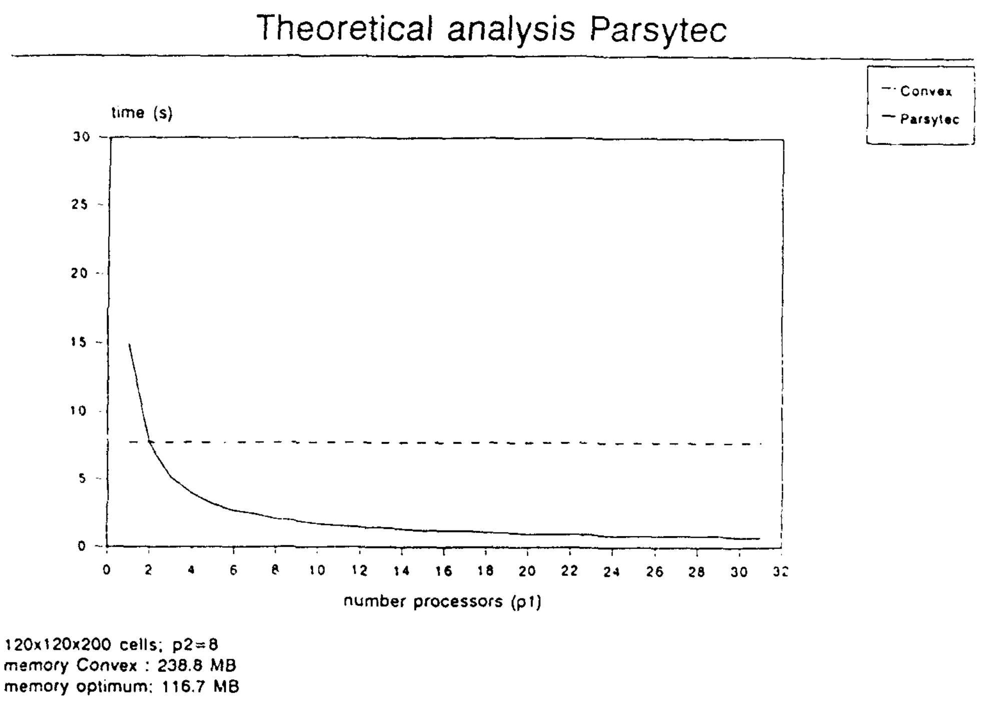


Fig. 4.4 Time needed for one update for a grid of 120 x 120 x 200 cells as a function of the number of processors p_1; $p_2=8$

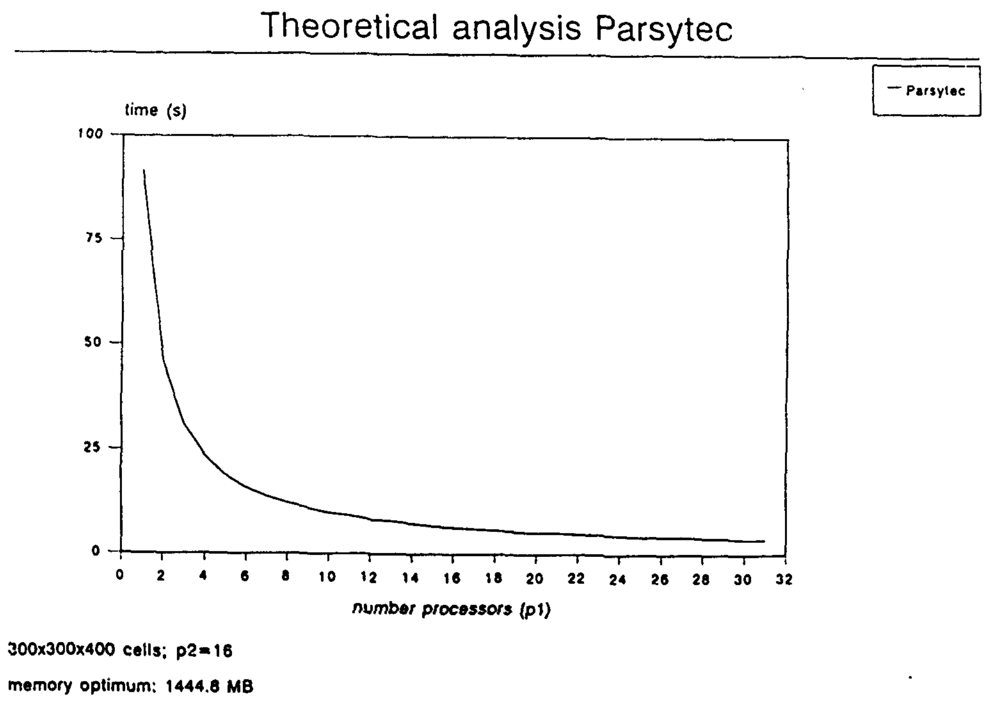


Fig 4.5 Time needed for one update for a grid of 300 x 300 x 400 cells as a function of the number of processors p_1; $p_2=16$

5 SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

In this report we have presented a theoretical study on the performance of an implementation of the FDTD code on a massively parallel computer. An expression was derived for the time needed to run this code on a parallel machine using a data decomposition algorithm. In this algorithm the computational domain is divided into columns that are assigned to the individual processors. For every time step each processor updates all field values in the column and then exchanges data at the borders of the columns with adjacent processors. This type of data-decomposition is a natural choice for a parallel computer which is configured as a two-dimensional grid of processors. The derived expression can be adapted in a straightforward manner to parallel computers configured as a three-dimensional grid of processors.
Run times for the code were determined for the Parsytec GCel-3/512 of the "Interdisciplinary Center for Computer based Complex systems research Amsterdam (IC^{3}A)", which is configured as a two-dimensional grid of 512 (16x32) T805 transputers. This was done under the assumption of asynchronous communication. Comparisons with the performance of an implementation on a conventional supercomputer, the Convex C230 of the Physics and Electronics Laboratory TNO, were presented.

5.2 Conclusions

From the analysis presented in this report it is concluded that the FDTD method is naturally suited for large scale processing on massively parallel processors (MPP). The numerical efficiency of a parallel implementation is expected to be 0.95 to 0.99, provided the grainsize is large. This means that the speed-up is directly proportional to the number of processors used.
On the Convex C230 the maximum domain size which can be handled comprises of 120x120x200 grid cells. For this problem the estimated execution time on the Parsytec is roughly 10 times smaller than on the Convex.
On the Parsytec the maximum domain size is 300x300x400 grid cells. The time needed for one time-step is slightly less than 4 s, which is not yet unacceptably high.

5.3 Recommendation

As a direction of future research we recommend to implement the FDTD code on the Parsytec GCel-3/512 at the IC^{3}A, in order to obtain a fast and powerful code for electromagnetic computation and to check the analysis presented here. Such a code can easily be ported to the next generation of parallel computers, which will make available computational speeds from 100 Gflops to 1 Tflop before the end of the century.

REFERENCES

[1] K.S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media", IEEE Trans. antennas and propagation, Vol. AP-14, No. 3, May 1966, pp.302-307.

[2] A. Taflove and M.E. Brodwin, "Numerical solution of steady state electromagnetic scattering problems using the time dependent Maxwell's equations", IEEE Trans. microwave theory tech., Vol. MTT-23, Aug. 1975, pp.623-630.

[3] B. Engquist and A. Majda, "Absorbing boundary conditions for the numerical solution of waves", Math. comp. 31, 1977, pp.629-651.

[4] G. Mur, "Absorbing boundary conditions for the finite difference approximation of the time domain electromagnetic field equations", IEEE Trans. electromagn. compat., EMC-23, 1981, pp.377-382.

[5] A. Taflove and K.R. Umashankar, "Review of FD-TD numerical modeling of electromagnetic wave scattering and radar cross section", Proceedings of the IEEE, Vol. 77, No. 5, May 1989, pp.682-699.

[6] A.Hoekstra, "Time complexity of a parallel conjugate gradient solver for light scattering simulations: Theory and SPMD implementation", Department of Computer Systems, Faculty of Mathematics and Computer Sciences, University of Amsterdam, Technical report CS-92-06, June 1992.

[7] G.Fox et al, "Solving problems on concurrent processors, Volume 1", Prentice Hills, Englewood Cliffs, New Jersey, 1988.

[8] G.v.Gennip, "Theory and applications of the 3-dimensional finite-difference time-domain method", TNO report FEL-92-B190, June 1992.

[9] A.Hoekstra, private communication.

[illegible]

G.A. van der Spek
(Group leader)

[illegible]

M.G.E. Brand
(Project leader/Author)

[illegible]

L.J. van Ewijk
(Author)

REPORT DOCUMENTATION PAGE (MOD-NL)

1. DEFENSE REPORT NUMBER (MOD-NL) TD94-0166	2. RECIPIENT'S ACCESSION NUMBER	3. PERFORMING ORGANIZATION REPORT NUMBER FEL-93-B366
4. PROJECT/TASK/WORK UNIT NO. 20369	5. CONTRACT NUMBER -	6. REPORT DATE APRIL 1994
7. NUMBER OF PAGES 28 (EXCL. RDP & DISTRIBUTION LIST)	8. NUMBER OF REFERENCES 9	9. TYPE OF REPORT AND DATES COVERED

10. TITLE AND SUBTITLE
A THEORETICAL STUDY ON THE PERFORMANCE OF THE FDTD CODE ON MASSIVELY PARALLEL COMPUTER

11. AUTHOR(S)
M.G.E. BRAND
L.J. VAN EWIJK

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TNO PHYSICS AND ELECTRONICS LABORATORY, P.O. BOX 96864, 2509 JG THE HAGUE
OUDE WAALSDORPERWEG 63, THE HAGUE, THE NETHERLANDS

13. SPONSORING/MONITORING AGENCY NAME(S)
TNO PHYSICS AND ELECTRONICS LABORATORY, P.O. BOX 96864, 2509 JG THE HAGUE
OUDE WAALSDORPERWEG 63, THE HAGUE, THE NETHERLANDS

14. SUPPLEMENTARY NOTES
THE CLASSIFICATION DESIGNATION ONGERUBRICEERD IS EQUIVALENT TO UNCLASSIFIED.

15. ABSTRACT (MAXIMUM 200 WORDS, 1044 POSITIONS)
IN THIS REPORT WE PRESENT A THEORETICAL STUDY ON THE PERFORMANCE OF AN IMPLEMENTATION OF THE FDTD CODE ON A MASSIVELY PARALLEL COMPUTER. THE FINITE DIFFERENCE TIME DOMAIN (FDTD) METHOD IS ONE OF THE MOST POPULAR APPROACHES FOR THE NUMERICAL MODELLING OF A WIDE VARIETY OF ELECTROMAGNETIC WAVE INTERACTION PROBLEMS. THE ANALYSIS PRESENTED IS CARRIED OUT FOR RADAR CROSS SECTION (RCS) PREDICTION BUT CAN, WITH MINOR ADAPTATIONS, BE USED FOR OTHER ELECTROMAGNETIC PROBLEMS. THE ANALYSIS IS APPLIED TO A POSSIBLE IMPLEMENTATION ON THE PARSYTEC GCEL-3/512 OF THE "INTERDISCIPLINARY CENTER FOR COMPUTER BASED COMPLEX SYSTEMS RESEARCH AMSTERDAM (IC3A)", WHICH IS CONFIGURED AS A TWO-DIMENSIONAL GRID OF 512 T805 TRANSPUTERS. COMPARISONS WITH THE PERFORMANCE OF AN IMPLEMENTATION ON A CONVENTIONAL SUPERCOMPUTER, THE CONVEX C230 OF THE PHYSICS AND ELECTRONICS LABORATORY TNO, ARE PRESENTED.

16. DESCRIPTORS	IDENTIFIERS
SCATTERING OF ELECTROMAGNETIC WAVES	MAXWELL EQUATIONS
RADAR CROSS SECTION	FINITE DIFFERENCE METHOD
NUMERICAL COMPUTATIONS	TIME DOMAIN METHOD
PARALLEL PROCESSING	
PERFORMANCE	

17a. SECURITY CLASSIFICATION (OF REPORT) ONGERUBRICEERD	17b. SECURITY CLASSIFICATION (OF PAGE) ONGERUBRICEERD	17c. SECURITY CLASSIFICATION (OF ABSTRACT) ONGERUBRICEERD
18. DISTRIBUTION/AVAILABILITY STATEMENT UNLIMITED		17d. SECURITY CLASSIFICATION (OF TITLES) ONGERUBRICEERD

Distributielijst

1.	Bureau TNO-Defensieonderzoek
2.	Directeur Wetenschappelijk Onderzoek en Ontwikkeling*)
3.	HWO-KL*)
4.	HWO-KLu*)
5.	HWO-KM*)
6 t/m 8.	Hoofd TDCK
9.	IC^{3}A, t.a.v. prof. dr. L.O.Hertzberger
10.	IC^{3}A, t.a.v. dr. P.M.A. Sloot
11.	IC^{3}A, t.a.v. dr. A.G.Hoekstra
12.	TNO-TM, Bibliotheek
13.	TNO-PML, Bibliotheek
14.	M&P-TNO, Marketing Communicatie
15.	Directie TNO-FEL, t.a.v. dr. J.W. Maas
16.	Directie TNO-FEL, t.a.v. ir. J.A. Vogel, daarna reserve
17.	Archief TNO-FEL, in bruikleen aan ir. G.H. Heebels
18.	Archief TNO-FEL, in bruikleen aan ir. H.R. van Es
19.	Archief TNO-FEL, in bruikleen aan ir. G.A. van der Spek
20.	Archief TNO-FEL, in bruikleen aan ir. H.J.M. Heemskerk
21.	Archief TNO-FEL, in bruikleen aan ir. L.J. van Ewijk
22.	Archief TNO-FEL, in bruikleen aan dr. M.G.E. Brand
23.	Archief TNO-FEL, in bruikleen aan ir. P.L.J. van Lieshout
24.	Archief TNO-FEL, in bruikleen aan ir. W. Huiskamp
25.	Archief TNO-FEL, in bruikleen aan ir. M.H. Smit
26.	Documentatie TNO-FEL
27.	Reserve

Indien binnen de krijgsmacht extra exemplaren van dit rapport worden gewenst door personen of instanties die niet op de verzendlijst voorkomen, dan dienen deze aangevraagd te worden bij het betreffende Hoofd Wetenschappelijk Onderzoek of, indien het een K-opdracht betreft, bij de Directeur Wetenschappelijk Onderzoek en Ontwikkeling.

*) Beperkt rapport (titelblad, managementuittreksel, RDP en distributielijst)