Simulation of an indoor radio channel at mm-wave frequencies

Melters, M.A.A.

Award date:
1990

FACULTY OF ELECTRICAL ENGINEERING
EINDHOVEN UNIVERSITY OF TECHNOLOGY
TELECOMMUNICATIONS DIVISION

SIMULATION OF AN INDOOR RADIO CHANNEL AT mm-WAVE FREQUENCIES.

BY

M.A.A. MELTERS

GRADUATION THESIS.

NOVEMBER 1989 - AUGUST 1990.

MENTORS:	IR. P.F.M. SMULDERS
	DR. IR. M.H.A.J. HERBEN.
SUPERVISOR:	PROF. DR. IR. G. BRUSSAARD.

SUMMARY.

The intention of this thesis is to implement a radio channel into a computer model and obtain information about the channel. Geometrical Optics is chosen as a model for the radio channel, because the radio system has to operate at mm-wave frequencies (around 60 GHz).
Attention is paid to the accuracy of the model for the frequencies under consideration.

The simulation programme calculates rays, which arrive at the receiver. These include the direct ray and rays which reflect 1, 2, 3 and 4 times against walls or objects within the room.
In the simulation it is possible to use some kind of radiation pattern for the transmitting and receiving antenna and different kinds of polarizations. It is also possible to neglect the polarization of the radio waves.

In characterizing the radio channel two parameters are looked at. Firstly the power delay spread as a measure of the width of the pulse response is considered. This power spread causes symbols to overlap at the receiver. This limits the maximum bit rate possible in the channel.
The second parameter is the total power at the receiver as a function of distance between transmitter and receiver. This parameter can be used as an indication of the power that needs to be transmitted.
Furthermore, the parameters calculated in case the polarization of the radio waves is considered, are compared with the parameters calculated when the polarization is not taken into account.

The results of the computer simulations are compared with measurements performed by the Telecommunications Laboratory of the Technical Research Centre of Finland. The comparisons show that although the frequencies used in the measurements were much lower than the frequencies considered in the simulation, the results are in good agreement with each other.

The results obtained by simulation show that the values for the delay spread and the rate of decay of received power with distance depend strongly on the room configuration and the antenna pattern used. The results do not differ much in case the polarization is considered or not.

Recommended is some kind of omnidirectional or horizontal radiating antenna and linear polarization in order to minimize the transmitted power to cover the whole room. The maximum achievable bit rate with these choices and without equalization of the radio channel is about 5 MBit/s for a room with dimensions of $40*60*4$ m^3. If the transmitted power is not restricted a vertical radiating source can be used. Using this antenna, the maximum achievable bit rate becomes 10 MBit/s.

CONTENTS.

1. Introduction.

Since 1988 the research programme of the Telecommunications Division of Eindhoven University of Technology contains a project in which various items in the field of inhouse radio are studied. The objective of this project is to examine the feasibility of a high-capacity indoor wireless network.
The project is conducted by Peter Smulders. Research is carried out in cooperation with many institutes by means of participation within COST (European Cooperation in the field of Scientific and Technical Research), i.e. COST 231/WG3.

The use of radio networks within an office building, a factory, a warehouse, a hospital etc. has several advantages over cable-based networks. It would free the users from cords tying them to particular locations. In addition the mobility would offer the flexibility of changing or creating various communication services without the need of expensive rewiring.

The present generation of indoor radio systems is primarily designed for speech and low data services, and cannot satisfy the demands in future broadband services. The appearance of powerful note-book laptop computers for video, voice and data for instance, leads to an anticipation of the need of flexible broadband radio interfaces in the near future. Therefore current research and development activities in the field of indoor radio are directed towards a large increase of capacity per user.

In working group 3 of COST 231, rooms are defined to which further investigations are directed. Also is agreement on communication between mobiles through intervention by a base station, positioned for instance in the middle of the room, close to the ceiling. The mobile antenna is thought to be 1.5 meters above the floor. The frequencies that are considered, lie above 30 GHz (especially around 60 GHz). This range results from the necessary bandwidth (approximately 2 GHz) of the system, which is not available below 30 GHz.

The main interest is characterizing the indoor radio channel. This is possible with a measurement set up in a room. However, measured channel characteristics are valid for a particular room configuration, namely the configuration of the room under consideration.

To be able to predict an indoor radio channel in general, one should undertake extensive measurements in several different rooms and even in several different buildings.
It would be a lot easier if computer simulations could replace the measurements. All possible room configurations could be looked at with almost no extra effort. The result could be statements about the maximum achievable data rate in the room and the effect of the room configuration on the maximum data rate.

The subject of this report is a description of a model for the indoor radio environment and the implementation of it in a computer programme. With this programme it should be possible to predict the radio characteristics of a particular indoor radio link.

The following subjects are discussed. Firstly the method of modelling the indoor radio link between a base station and a mobile is discussed and some simplifications made in the process. Then the concept of the computer programme is described with subjects like reflection, polarization and antenna gain function. The results of some simulations are shown after remarks about the underlying signal theory are made. Finally, conclusions are drawn.

In Appendix G a description of the computer programmes is given.

2. Model for an indoor radio link.

2.1. Introduction.

In an indoor radio system, with both transmitter and receiver antennae positioned in the same room, electromagnetic power will travel directly from transmitter to receiver, but also via reflections against walls and objects within the room. The room constitutes in this case a multipath channel for a radio link.

In choosing a model for implementation in a simulation programme, there are some restrictions. First of all the model has to be accurate and at the same time easy to implement in a computer programme. Furthermore computertime is expensive, so the programme must be well written and the model chosen, must make it possible to quickly simulate a radio link for some room configuration.
The most important feature is, however, that the model must give some physical insight in the most important parameters that constitute the characteristics of the radio link. For instance, it must be relatively easy to predict the changes in the radio link when an object is moved from one place to another, or when a wall is coated with another material, changing the reflectivity.

These restrictions make the choice of a model much easier. Solving the Maxwell equations for a particular room configuration for instance, is very difficult and hard to implement. So this option is not considered here. An alternative is making use of Physical Optics (PO) to describe the reflections against walls and objects, as is commonly done for reflector antenna systems. This method includes an integration of the incident electromagnetic fields over all surfaces which are illuminated by the transmitter radiation and radiation reflected by one or more surfaces. If the room configuration becomes complicated, however, the computation of the integrals is very time consuming. This means that PO is also not suited as a model for the indoor radio link.

For the frequencies under consideration there are some methods known, based on rays. These are Geometrical Optics (GO), Geometrical Theory of Diffraction (GTD), Uniform geometrical Theory of Diffraction (UTD) and Uniform Asymptotic Theory of diffraction (UAT). GO describes the field received directly from the transmitter and via reflections. This method is based on the assumption that the wavelength approaches zero. The other techniques assume that the wavelength is small compared with the dimensions of the objects the electromagnetic wave interacts with.

GO predicts zero fields in shadow regions as shown in Fig. 1.

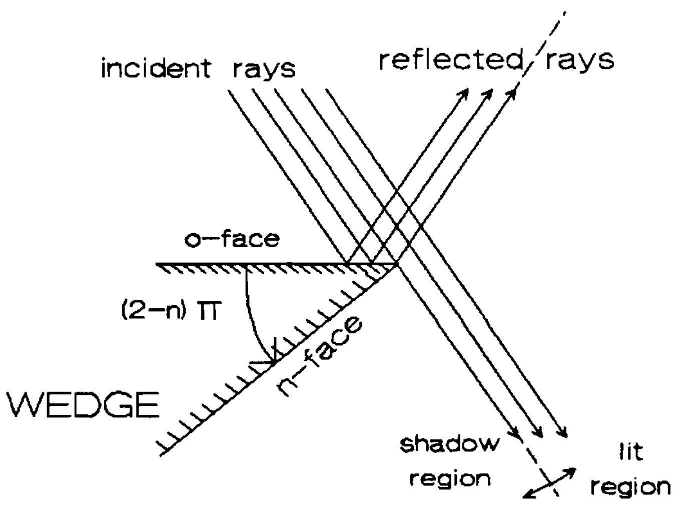


Fig. 1: Configuration of a wedge illuminated by a plane wave (GO).

GTD supplies the GO with rays diffracted by edges or corners of objects. The major limitation of GTD is its failure at reflection and shadow boundaries, where it predicts infinite fields. UTD is in this case the wanted extension. It predicts smooth continuous fields at the boundaries and approaches GTD as the observation point is removed from these boundaries.
UAT also predicts finite fields for the directions mentioned. Not by adjusting the GTD coefficients, but by changing the GO field to infinite fields compensating the GTD infinite fields. This gives less physical insight and also will cause problems when computing the infinite fields. For these reasons and because UTD is very well described in literature [1,2,3,4], UTD seems to be the most appropriate method to compute the fields. Therefore, UTD will be treated in more detail.

2.2. Description of UTD.

2.2.1. UTD as an extension for GO.

Before considering the accuracy of UTD the development from GO to UTD will be described. The time dependence exp(jωt) is suppressed throughout.

The total GO field in an observation point P is given by [5]

$$E^{go}(P) = E^{d}(P) + E^{r}(P), \tag{1a}$$

in which the GO direct and reflected rays E^d and E^r exist only in the lit regions.

Let $E^i_\parallel(Q_R)$ be the component of the incident field in Q_R in the direction of $\hat{e}^i_\parallel$ in Fig. 2. Likewise, let $E^r_\parallel(P)$ be the component of the reflected field in P in the direction of $\hat{e}^r_\parallel$, and let $E^i_\perp(Q_R)$ and $E^r_\perp(P)$ be the components of the incident field in Q_R and reflected field in P respectively, perpendicular to the plane of incidence defined by $\hat{n}$ and $\hat{s}^i$ in Fig.2. In case of reflections at a perfectly conducting surface, the reflected field is described by

$$\begin{bmatrix} E^r_\parallel(P) \\ E^r_\perp(P) \end{bmatrix} = \begin{bmatrix} E^i_\parallel(Q_R) \\ -E^i_\perp(Q_R) \end{bmatrix} \sqrt{\frac{\rho^r_1 \rho^r_2}{(\rho^r_1+s^r)(\rho^r_2+s^r)}}\; e^{-jks^r}. \qquad (1b)$$

The distances ρ^r_1, ρ^r_2 and s^r are associated with the reflected ray from Q_R to P as shown in Fig. 2. The wave number is denoted by k and the square root results from the conservation of energy in a tube. Expression (1b) must be extended by a dyadic surface reflection coefficient if the reflecting surface is not perfectly conducting.

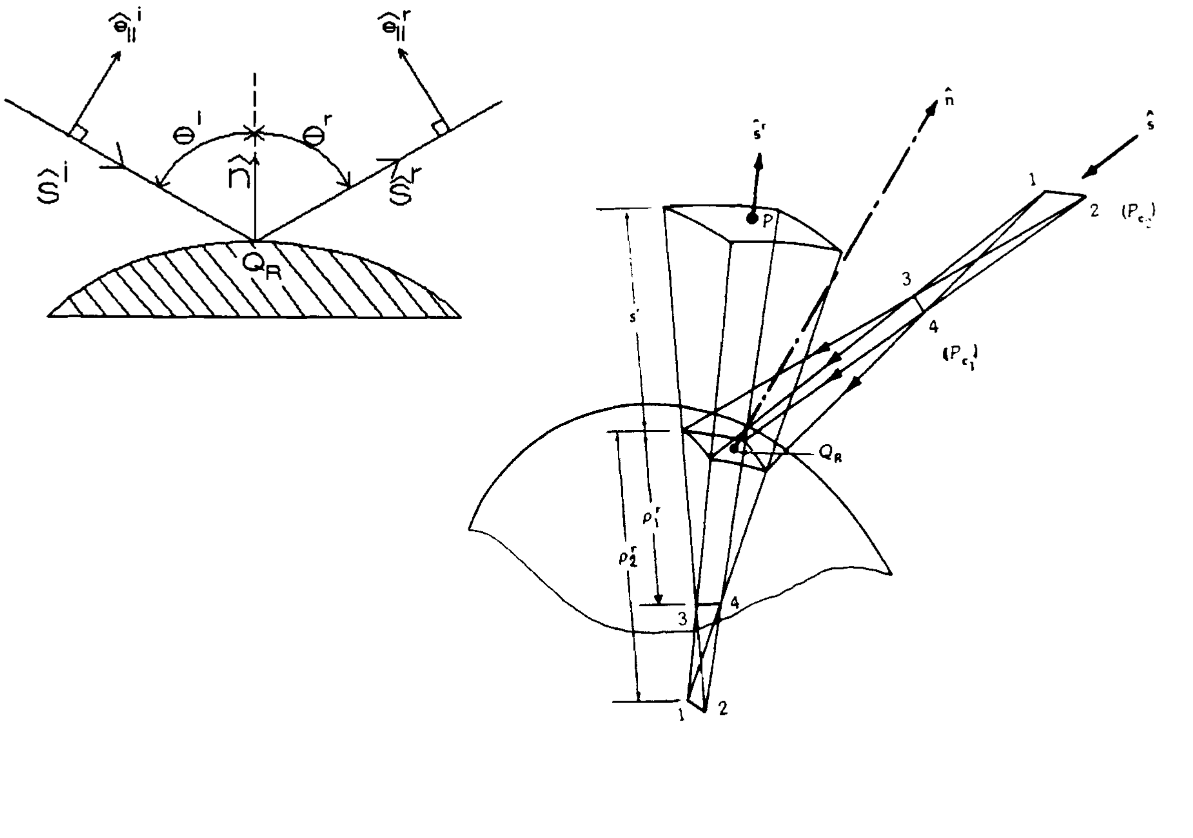


Fig. 2: The reflected GO field.

Diffraction at the wedge is shown in Fig. 3.

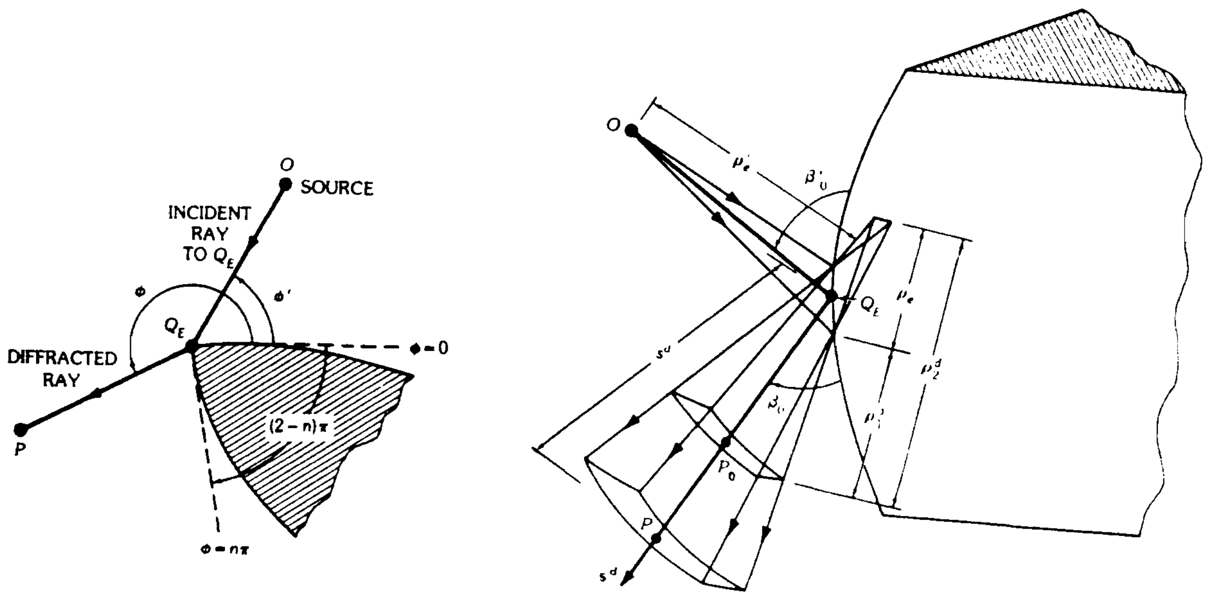


Fig. 3: Diffraction at the wedge.

The GTD field consists of a diffracted ray E_k^d added to the GO rays. Considering the situation in Fig. 3, the diffracted field is described by

$$\begin{bmatrix} E^d_{\beta_0}(P) \\ E^d_{\phi}(P) \end{bmatrix} = \begin{bmatrix} -D^k_{es} E^i_{\beta'_0}(Q_E) \\ -D^k_{eh} E^i_{\phi'}(Q_E) \end{bmatrix} \sqrt{\frac{\rho_e}{s^d(\rho_e + s^d)}} \; e^{-jks^r}. \tag{2a}$$

The soft and hard diffraction coefficients D^k_{es} en D^k_{eh} are functions of ϕ, ϕ' and β_0, and can be written as

$$D^k_{es,eh} = \frac{e^{-j\pi/4}\sin(\pi/n)}{n\sqrt{2\pi k}\,\sin\beta_0} \left[\frac{1}{\cos(\pi/n) - \cos[(\phi-\phi')/n]} \mp \frac{1}{\cos(\pi/n) - \cos[(\phi+\phi')/n]} \right], \tag{2b}$$

where n defines the exterior wedge angle as in Fig. 3; hence n=2 for a half-plane and n=3/2 for an exterior right angle, etc.
One observes that $D^k_{es,eh}$ becomes infinite for $\phi - \phi' = \pi$ and $\phi + \phi' = \pi$. These are the shadow and reflection boundary, respectively.

UTD supplies for these directions the correct coefficients by replacing $D^k_{es,eh}$ by

$$\begin{aligned} D_{es,eh} = \frac{e^{-j\pi/4}}{2n\sqrt{2\pi k}\,\sin\beta_0} & \Bigg[\cot\left(\frac{\pi + (\phi - \phi')}{2n}\right) F\,[kL^i a^+(\phi - \phi')] \\ & + \cot\left(\frac{\pi - (\phi - \phi')}{2n}\right) F\,[kL^i a^-(\phi - \phi')] \\ & \mp \Bigg\{ \cot\left(\frac{\pi + (\phi + \phi')}{2n}\right) F\,[kL^{rn} a^+(\phi + \phi')] \\ & + \cot\left(\frac{\pi - (\phi + \phi')}{2n}\right) F\,[kL^{ro} a^-(\phi + \phi')] \Bigg\} \Bigg] \end{aligned} \tag{3a}$$

The function F contains a Fresnel integral and is defined by

$$F(x) = 2j\sqrt{x}\,e^{jx} \int_{\sqrt{x}}^{\infty} e^{-j\tau^2}\, d\tau, \tag{3b}$$

where the parameter kL is required to be sufficiently large (generally greater than 3) and

$$a^{\pm}(\beta) = 2\cos^2\left(\frac{2n\pi N^{\pm} - \beta}{2}\right), \tag{3c}$$

in which $N^{\pm}$ are the integers which most nearly satisfy the equation

$$2n\pi N^{\pm} - \beta = \pm\pi, \tag{3d}$$

with

$$\beta = \phi \pm \phi'. \tag{3e}$$

For a straight wedge with planar faces, illuminated by a point source does

$$L^{ro} = L^{rn} = L^i = \frac{s^i s^d}{s^i + s^d} \sin^2 \beta_0, \tag{3f}$$

in which s^i and s^d are the distances from the point of the edge diffraction at Q_E to the source and observation points, respectively.

Although the UTD diffraction coefficient has cotangent terms which become singular for shadow and reflection boundaries, the combination of the cotangent term and the Fresnel integral remains finite. The result is a step in $D_{es,eh}$, which nicely compensates the discontinuity in the GO field, giving a smooth continuous field across the boundaries.

The above coefficients are derived for perfectly conducting surfaces. However coefficients for non-perfectly conducting surfaces are also available. They are described in articles of Tiberio and Griesser [3,4].

2.2. Accuracy of UTD.

On modelling a real situation, one has always to consider the limitations of the model. Aas [6] has compared the UTD solution with the exact solution for a plane wave incident on a perfectly conducting 90° exterior wedge (Fig. 4).

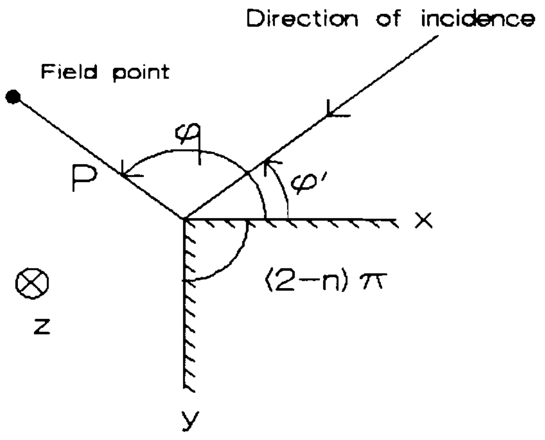


Fig. 4: Plane wave incident on exterior perfectly conducting wedge.

He studied the accuracy of the theory when the field point (or, by reciprocity, the source point) approaches the edge.

The results show the field errors $|\Delta E_z| = \dfrac{|E_z^{utd} - E_z^{exact}|}{|E_z^{exact}|}$ and $|\Delta H_z| = \dfrac{|H_z^{utd} - H_z^{exact}|}{|H_z^{exact}|}$,

for three different angles and four different distances (Fig.5).

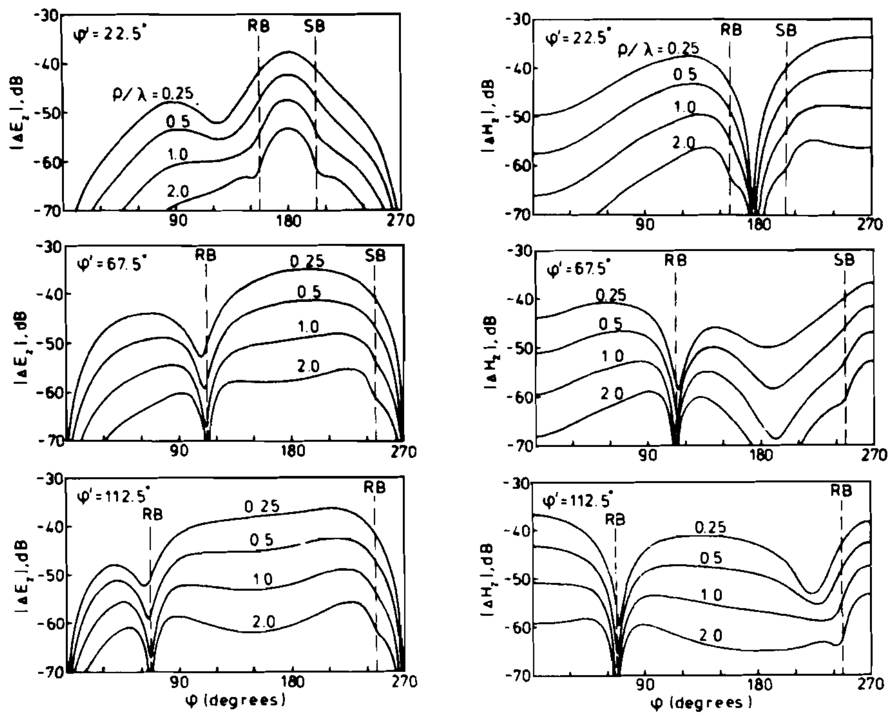


Fig. 5: Accuracy of UTD for a plane wave on perfectly conducting wedge.

In Fig. 5 one can see that the UTD solution remains accurate for points 0.25λ removed from the wedge. The remaining field components E_x, E_y, H_x and H_y have to be calculated directly from the Maxwell equations, because the errors in these components will be large.

The accuracy of the UTD for a non-perfectly conducting wedge is presented in [4]. Again a situation is considered where a plane wave is incident on the wedge, or by reciprocity a line source is located in the interior wedge. The angle of incidence is 30°. The distance from the field point/line source to the wedge is 1.6λ (kρ=10). See Fig. 6.

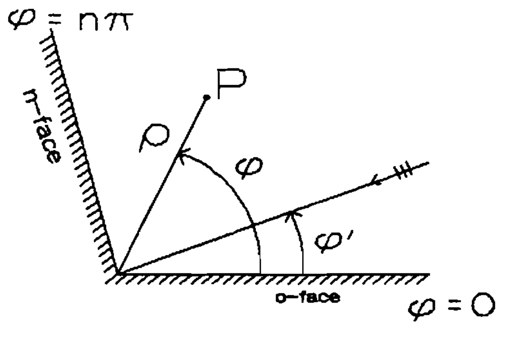


Fig. 6: Line source in a non-perfectly conducting wedge.

In Fig. 7 the UTD solution is compared with the exact solution for both polarizations. The total field in the direction of ϕ is normalized to a line source field of unity.

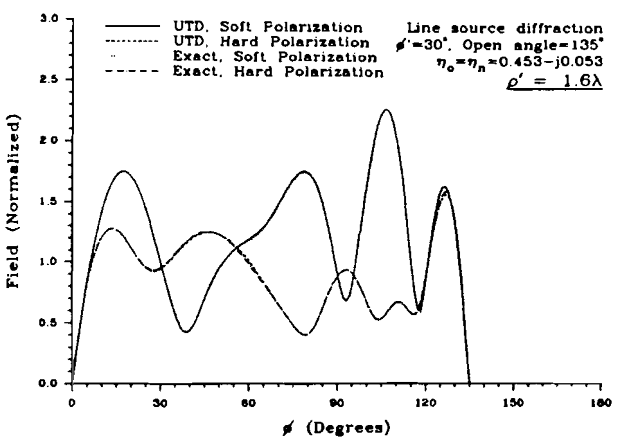


Fig. 7: Accuracy of UTD for a non-perfectly conducting wedge.

As one can see the solutions show excellent agreement with each other. Only a slight difference occurs for the hard polarization at $\phi = 125°$. This means that UTD is very accurate down to 1.6λ away from the wedge.

2.3. Contribution of diffracted rays to total field.

To calculate the total field in the near vicinity of the wedge, one has to add a surface wave and a surface wave transition field. The two are necessary to get a continuous total field. However, these two components decay typically exponentially away from the face of the wedge, so they will not be considered here.

Of special interest is the contribution of the diffracted rays to the total field in a particular field point. This is illustrated in Fig. 8 [4], where the individual field components are compared with each other. The results are calculated for an incident plane wave on a non-perfectly conducting 85° interior wedge. The reason why this is important results from implementation problems. If the diffracted rays do not form a significant contribution to the total field, it is not necessary to include them in the simulation programme. This means that only the geometrical rays have to be implemented, which is a lot easier.

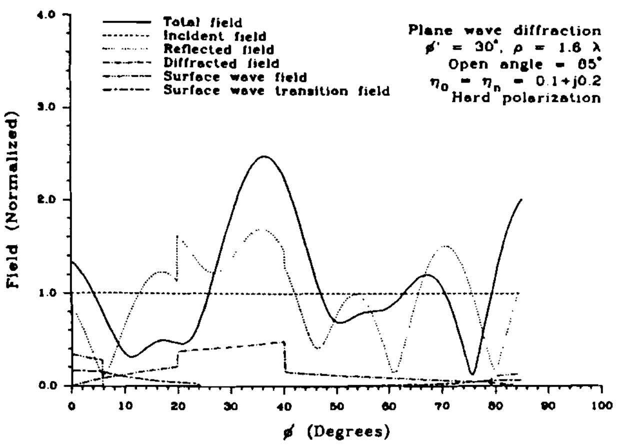


Fig 8: Contribution of field components to total field.

As one could expect, the absolute value of the reflected field is discontinuous at $\phi=20°$ and $\phi=40°$, corresponding to shadow boundaries of the double reflected fields. The discontinuities, added by the diffracted field, exactly compensate the Geometrical Optics discontinuities.

The conclusion is that the diffracted rays are important only for the directions of the shadow and reflection boundaries and close to these boundaries. The question is then, how broad are these areas. To draw conclusions about this out of Fig. 8 is very tricky. Therefore the power of the diffracted rays with respect to the reflected rays is calculated for a 90° interior and exterior wedge. This work is done by Peter Smulders. For the configuration see Fig. 9.

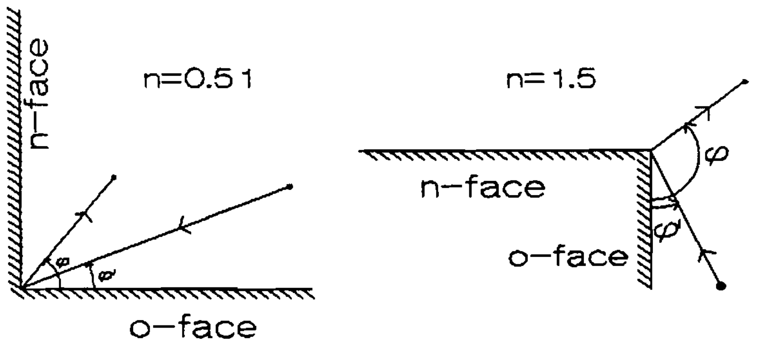
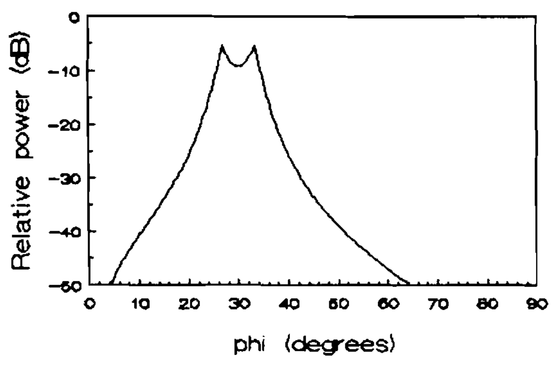


Fig. 9: Configuration for calculating the strength of diffraction rays.

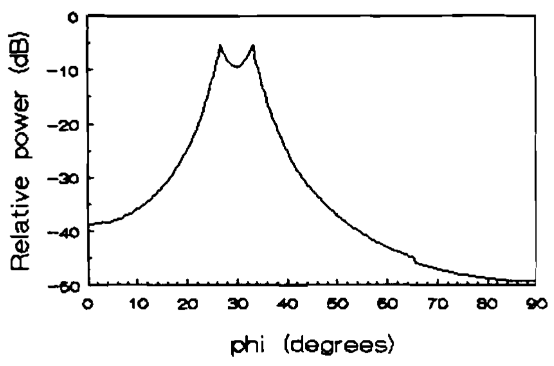
The transmitter and receiver are point sources at a distance of 10m and 5m from the edge, respectively and the angle of incidence $\phi' = 30°$.

The power of the diffracted rays is calculated using the diffraction coefficients of UTD for a perfectly conducting wedge as given in equations (3a) to (3f). It is normalized with respect to the received power of the reflected ray in case the receiver is located at the reflection boundary. Fig. 10 shows the results for the interior wedge (n=0.51). The reason for the choice of n=0.51 is that for n=0.50 the calculation of the Fresnel integrals becomes inaccurate.

In Fig. 11 the results for the exterior wedge (n=3/2) are shown.

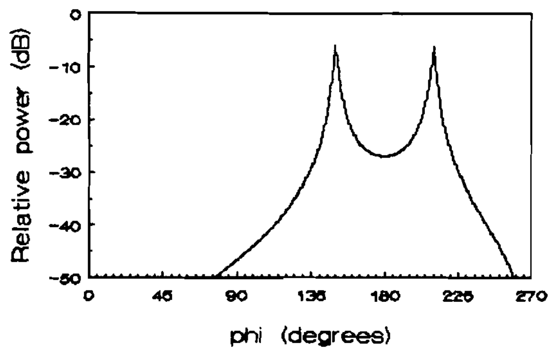


β-component.

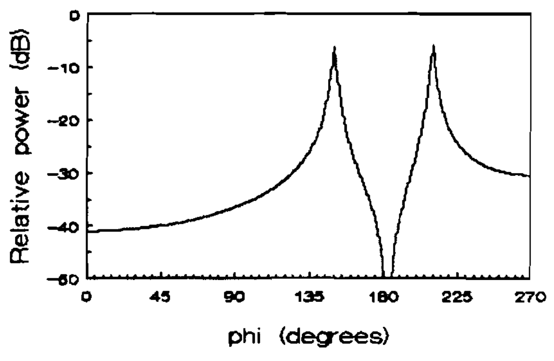


ϕ-component.

Fig. 10: Comparison between power of diffracted and reflected rays (n=0.51).



β-component.



ϕ-component.

Fig. 11: Comparison between power of diffracted and reflected rays (n=1.5).

These figures show that the power of the diffracted rays remains relatively important over broad observation angles around the shadow and reflection boundaries. This implies that diffracted rays even at frequencies of 60GHz may contribute significantly to the total received power.

However, since the available amount of time for this research is limited, a first impression of the indoor radio channel is obtained by considering Geometrical Optic rays only. The contribution of diffracted rays to the total received power is therefore subject to further investigations.

3. Fundamentals of the simulation programme.

3.1. Introduction.

In chapter two it was decided that the simulation programme would be based on Geometrical Optics only. Using GO means that rays have to be defined along which electromagnetic power travels from transmitter to receiver. Reflection coefficients have to be applied and electromagnetic field polarization states have to be looked at.

In the simulation only the reflectivity of objects and walls are considered and not the transmissivity. Thus the walls of the room are assumed infinitely thick and the objects within the room do not allow an electromagnetic wave to travel through them.

Further assumptions, made in the process of writing the programme, are explained in this chapter. Furthermore, the techniques used for defining rays and applying reflection coefficients, are discussed.

3.2. Geometrical Optic rays.

The principal way of defining a reflected ray, is making use of the mirror technique. This technique is illustrated for a two dimensional case in Fig. 12a. A line from point T perpendicular to AA', is drawn to point M. M is at precisely the same distance from AA' as T is. Then the line MR is drawn and the reflected ray TCR is constructed.

If we want to use this method in a three dimensional room, all surfaces are described by vectors. One vector to describe the position of a corner of the surface and two vectors to describe the direction. See Fig. 12b.

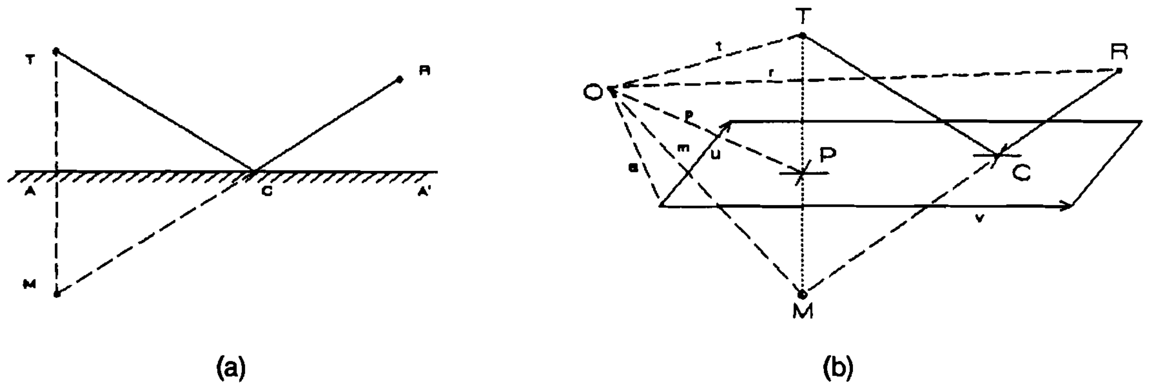


Fig. 12: Mirror technique for two resp. three dimensional case.

To calculate M for the three dimensional situation, an origin is chosen. The position vector of the surface U is $\underline{a}$, and the direction vectors are $\underline{v}$ and $\underline{u}$. The mirror point M of T is to be calculated. P is the crossing point of the line TM perpendicular to the surface.

The equation that defines the surface is

$$U: \underline{x} = \underline{a} + \lambda\underline{u} + \mu\underline{v}. \tag{6}$$

with $\underline{x}$ = (x,y,z) and λ and μ between 0 and 1.

The equations that hold are

$$\begin{aligned} &\underline{t} - \underline{p} = \gamma_0\,(\underline{u} \times \underline{v}) \\ &\quad \underline{p} = \underline{a} + \lambda_0\underline{u} + \mu_0\underline{v}\,\cdot \end{aligned} \tag{7}$$

Combining these two gives

$$\underline{t} - \underline{a} = \lambda_0\underline{u} + \mu_0\underline{v} = \gamma_0(\underline{u} \times \underline{v}). \tag{8}$$

Multiplying with $\underline{u}$ and $\underline{v}$, respectively, results in

$$\begin{aligned} (\underline{t}-\underline{a})\cdot\underline{u} &= \lambda_0|\underline{u}|^2 + \mu_0(\underline{v}\cdot\underline{u}) \\ (\underline{t}-\underline{a})\cdot\underline{v} &= \mu_0|\underline{v}|^2 + \lambda_0(\underline{u}\cdot\underline{v}). \end{aligned} \tag{9}$$

Because of the fact that $\underline{u}$ and $\underline{v}$ are defined in such a way, that they are perpendicular to each other, the inner product $(\underline{u}.\underline{v}) = 0$.
This gives

$$\begin{aligned} \lambda_0 &= \frac{((\underline{t}-\underline{a})\cdot\underline{u})}{|\underline{u}|^2} \\ \mu_0 &= \frac{((\underline{t}-\underline{a})\cdot\underline{v})}{|\underline{v}|^2}. \end{aligned} \tag{10}$$

From these equations $\underline{p}$ can be calculated and M becomes

$$M: \ \underline{m} = \underline{t} + 2(\underline{p}-\underline{t}). \tag{11}$$

Now the crossing point C of the line MR with the surface has to be looked at. For reference see Fig. 12b. The equation for the line MR is

$$MR : \underline{x} = \underline{r} + \gamma(\underline{r}-\underline{m}). \tag{12}$$

Combining with the equation for U, gives

$$\lambda_0\underline{u} + \mu_0\underline{v} + \gamma_0(\underline{r}-\underline{m}) = \underline{r}-\underline{a}. \tag{13}$$

This is a combination of three equations, which can be solved for γ_0, μ_0 and λ_0. The crossing point C follows from (12). In this way the reflected ray TCR is constructed.
For multiple reflected rays, other surfaces have to act as a mirror for M (Fig.13).

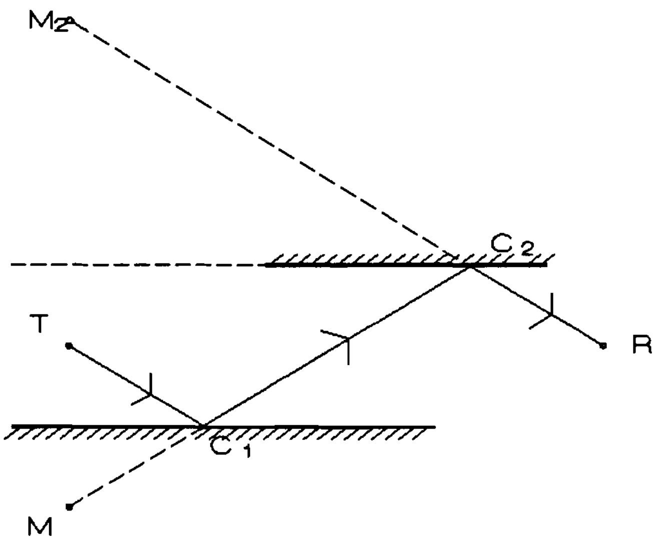


Fig.13: Multiple reflected rays.

The crossing points of the lines M_2R and MC_2 with the two reflecting surfaces, give the two times reflected ray TC_1C_2R. Rays which reflect three or four times are constructed in a similar manner.

3.3. Reflection coefficients.

3.3.1. Parallel and perpendicular polarization.

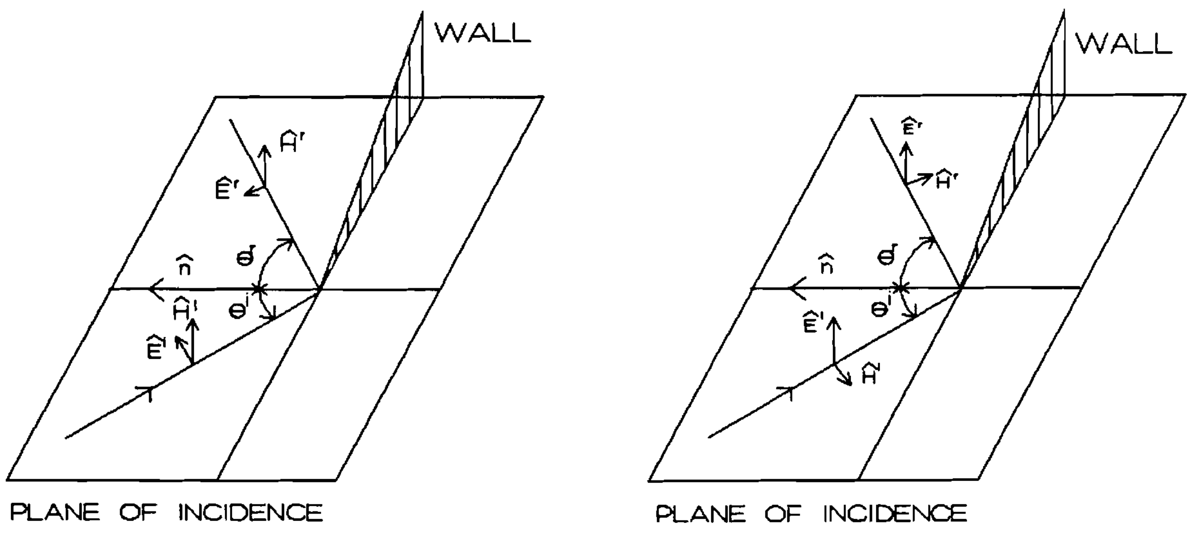
The Fresnel reflection coefficients for an infinite surface and an incident plane wave, are easily extracted from the boundary conditions for the electric and magnetic field and the material properties ϵ, μ, and σ, respectively the dielectric permittivity, the magnetic permeability and the conductance. For the parallel and perpendicular electric field components of an incident plane wave (see Fig. 14) the reflection coefficients are defined by [7]

$$R_\perp = \frac{E^r_\perp}{E^i_\perp} = \frac{\cos\theta_i - \frac{\mu_1}{\mu_2}\sqrt{\frac{\mu_2}{\mu_1}n^2 - \sin^2\theta_i}}{\cos\theta_i + \frac{\mu_1}{\mu_2}\sqrt{\frac{\mu_2}{\mu_1}n^2 - \sin^2\theta_i}}$$

$$R_\parallel = \frac{E^r_\parallel}{E^i_\parallel} = \frac{n^2\cos\theta_i - \sqrt{\frac{\mu_2}{\mu_1}n^2 - \sin^2\theta_i}}{n^2\cos\theta_i + \sqrt{\frac{\mu_2}{\mu_1}n^2 - \sin^2\theta_i}} , \qquad (14)$$

respectively, with

$$n^2 = \epsilon_{r_2} - j\frac{\sigma_2}{\omega\epsilon_0} . \qquad (15)$$



(a) (b)

Fig. 14: Parallel (a) and perpendicular (b) polarization.

As stated these coefficients hold for infinite surfaces and an incident plane wave. Using these coefficients in the simulation programme implies that small errors will be made in certain situations.
In general it is allowed to use the coefficients for finite objects and walls if half of the first Fresnel zone is smaller than the dimensions of the objects and walls, since this area effectively contributes to the reflection of the wave. The first Fresnel zone is considered in the next paragraph.

3.3.2. Fresnel zone.

The first Fresnel zone is described by the points along which the distance from transmitter to receiver is $\lambda/2$ longer than the Geometrical Optics reflected ray. With an intervention of a reflecting surface, the points of the first Fresnel zone on the surface are drawn in Fig. 15.

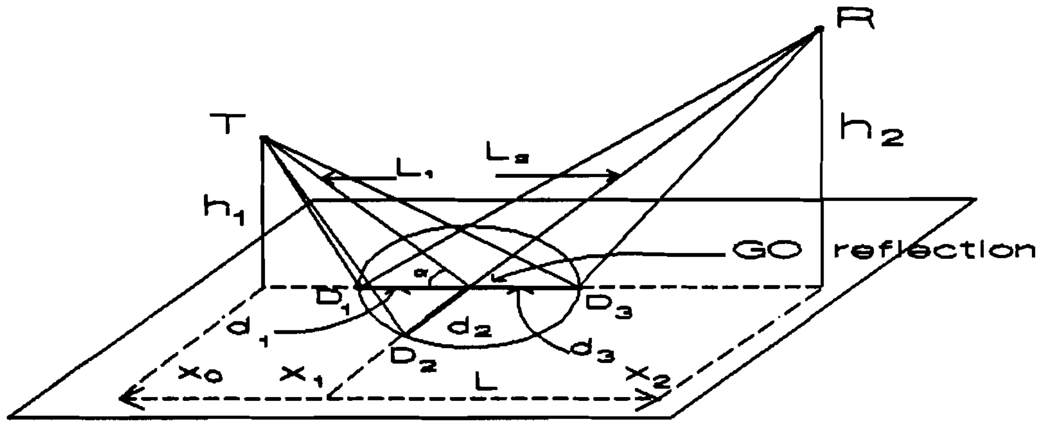


Fig. 15: First Fresnel zone on reflecting surface.

The length of the Geometrical Optics reflected ray is

$$L^d = L_1 + L_2 = \sqrt{h_1^2+(x_1-x_0)^2} + \sqrt{h_2^2+(x_2-x_1)^2}\ . \tag{16}$$

The ray from transmitter to receiver via D_1 resp. D_2 have lengths

$$\begin{aligned} TD_1R&: L_1 + L_2 + \frac{\lambda}{2} = \sqrt{h_1^2+(x_1-x_0-d_1)^2} + \sqrt{h_2^2+(x_2-x_1+d_1)^2} \\ TD_2R&: L_1 + L_2 + \frac{\lambda}{2} = \sqrt{L_1^2+d_2^2} + \sqrt{L_2^2+d_2^2} \end{aligned} \tag{17}$$

The ray via D_3 can be found by replacing d_1 by d_3 in (17).

These equations can be solved for d_1, d_2 and d_3, if h_1, h_2 and $L = x_2\text{-}x_0$ are given.
The parameter h_1 and h_2 are different for each reflecting surface, even when the transmitter and receiver are stationary. They are also dependent on the order of reflection. See Fig. 16.

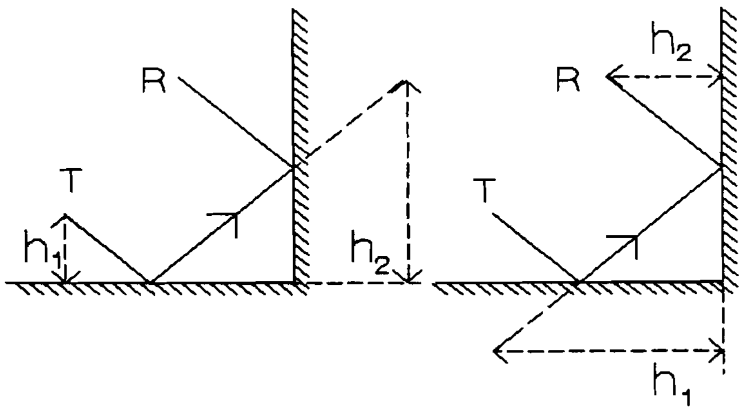


Fig. 16: Definition of parameters for a two times reflected ray.

An estimation of h_1, h_2 and L depends also on the room dimensions. The rooms defined in COST have dimensions of 10*10*3 m^3, 60*40*4 m^3 and 75*4*3 m^3. This means that the values of h_1, h_2 and L extend from 1 to over 100 meters. For some values of the parameters h_1, h_2 and L the first Fresnel zone characterized by d_1, d_2 and d_3 is calculated numerically. The results are listed in Table 1. The wave length is 0.005m.

Table 1: First Fresnel zone, defined by d_1, d_2 and d_3 [m].

α(°)	h_1[m]	h_2[m]	L[m]	d_1	d_2	d_3
45.0	0.5	0.5	1.0	0.059	0.042	0.059
11.3	0.5	0.5	5.0	0.402	0.080	0.402
5.71	0.5	0.5	10.0	1.099	0.112	1.099
2.86	0.5	0.5	20.0	3.020	0.158	3.020
1.91	0.5	0.5	30.0	5.421	0.194	5.421
26.6	0.5	2.0	5.0	0.143	0.067	0.156
14.0	0.5	2.0	10.0	0.349	0.091	0.398
7.13	0.5	2.0	20.0	0.925	0.130	1.115
4.76	0.5	2.0	30.0	1.647	0.155	2.070
14.0	0.5	4.5	20.0	0.365	0.096	0.431
9.46	0.5	4.5	30.0	0.641	0.117	0.786
7.13	0.5	4.5	40.0	0.961	0.135	1.217
40.4	0.5	8.0	10.0	0.089	0.060	0.097
23.0	0.5	8.0	20.0	0.185	0.078	0.212
12.0	0.5	8.0	40.0	0.464	0.106	0.564
38.7	2.0	2.0	5.0	0.143	0.090	0.143
21.8	2.0	2.0	10.0	0.312	0.116	0.312
11.3	2.0	2.0	20.0	0.812	0.160	0.812
5.71	2.0	2.0	40.0	2.239	0.224	2.239
28.1	8.0	8.0	30.0	0.438	0.206	0.438
21.8	8.0	8.0	40.0	0.625	0.232	0.625

From Table 1 it is clear that d_2 does not place a restriction on the dimensions of the objects since it is always smaller than d_1 and d_3 or smaller than 0.25m.

Looking at the values of d_1 and d_3 in Table 1 and choosing $2*2m^2$ as a lower limit on the dimensions of the walls and surfaces of the objects in the rooms then only in case α is small, errors are made in using the Fresnel reflection coefficients. However small α implies generally large L and this means that only for rays which have to travel far and thus have large L a small error is made by using incorrect reflection coefficients. Since the power carried by these rays is small compared to the direct ray and the reflected rays with small L the error introduced by the incorrect reflection coefficient will be negligable. Therefore a lower limit of $2*2m^2$ for the dimensions of the objects is justifiable.

Furthermore this choice takes into account the error made in the reflection coefficients when a ray reflects close to the edge of a surface, since generally half of the first Fresnel zone is smaller then approximately $0.25*0.50m^2$ at 60GHz.

3.3.3. Reflection coefficients.

In the equations for the reflection coefficients (12), some simplifications, taking into account the material properties, can be made. The material properties for the frequencies under consideration, known so far, are summarized in Table 2.

Table 2: Material properties at high frequencies known so far.

Material	ϵ_r	μ_r	σ	Reflectivity %	Source
Plasterboard (1cm,60GHz)				2	[8]
Chipboard (1.9cm,60GHz)				20	[8]
Autoclaved aerated concrete blocks (10cm,60GHz)				16	[8]
Aluminium (1mm,60GHz)				>99	[9]
Wood (20mm,60GHz)				2	[9]
Glass (3mm,60GHz)				16	[9]
Homo Sapiens (60GHz)				<0.06	[9]
Building brick					
(Humidity 0%, 1GHz)	3		0		[10]
(Humidity 0%, 10GHz)	3		0		[10]
(Humidity 5%, 1GHz)	5.5		2.2E-2		[10]
(Humidity 5%, 10GHz)	5		4.1E-1		[10]
Glass (10GHz)	5-10				[10]
Wood (10GHz)	3-7				[10]
Perspex (10GHz)	3				[10]
Concrete					
(Humidity 0%, 0.14GHz)	4.5		1.4E-3		[10]
(Humidity 5%, 0.14GHz)	6.5		5.6E-3		[10]
(Humidity 15%, 0.14GHz)	14		2.8E-2		[10]
Aluminium (0.14GHz)	1	1	3.7E7		[10]
Iron (0.14GHz)	1	1E4-1E5	9.5E6		[10]

For metal (no iron), the following values for ϵ, σ and μ can be substituted. The frequency is 60 GHz.

$$\begin{aligned}
\epsilon_{r_2} &= 1-10 \\
\epsilon_0 &= \frac{1}{36\ \pi} * 10^{-9}\ \frac{F}{m} \\
\sigma_2 &= 3.3*10^7\ \ (\Omega m)^{-1} \\
\mu_{r_2} &= 1\ \ \frac{H}{m}\ .
\end{aligned} \tag{18}$$

The square root in the equations for the reflection coefficients becomes

$$\sqrt{\epsilon_{r_2}\ -j*10^7-\sin^2\theta_i} = 3.16\ *10^3\ e^{j\frac{3\pi}{4}}\ . \tag{19}$$

Substituting (19) into (14) gives

$$\begin{aligned}
R_{\perp} &= -1 \\
R_{\parallel} &= +1.
\end{aligned} \tag{20}$$

For iron the substitutions are

$$\begin{aligned}
\mu_{r_2} &= 10^4\ -\ 10^5 \\
\epsilon_{r_2} &= 1 \\
\sigma_2 &= 9.5*10^6\ \ (\Omega m)^{-1}\ ,
\end{aligned} \tag{21}$$

and the square root term becomes

$$\begin{aligned}
\frac{\mu_1}{\mu_2}\sqrt{\frac{\mu_2}{\mu_1}\,(\epsilon_{r_2}\ -j\frac{\sigma}{\omega\ \epsilon_0})\ -\ \sin^2\theta_i} &= \frac{1}{10^4}\sqrt{10^4(1-j1.7*10^8)\ -\ \sin^2\theta_i} \\
&= \frac{1}{10^4}\ 1.3*10^2\ e^{j\frac{3\pi}{4}}\quad .
\end{aligned} \tag{22}$$

The coefficients are now

$$\begin{aligned}
R_{\perp} &= -1 \\
R_{\parallel} &= +1,
\end{aligned} \tag{23}$$

which is the same as for other conductors (20).

For non-conducting materials, for instance concrete, the substitutions are

$$\begin{aligned} \epsilon_{r_2} &= 1-10 \\ \mu_{r_2} &= 1 \\ \sigma_2 &= 10^{-2} - 10^{-3} \ (\Omega\ \mathrm{m})^{-1}. \end{aligned} \qquad (24)$$

The square of the refractive index n becomes

$$n^2 = \epsilon_{r_2} - j\frac{\sigma_2}{\omega \epsilon_0} = \epsilon_{r_2} - j\ 0.001 \approx \epsilon_{r_2}\,, \qquad (25)$$

and the reflection coefficients

$$\begin{aligned} R_\perp &= \frac{\cos\theta_i - \sqrt{\epsilon_{r_2} - \sin^2\theta_i}}{\cos\theta_i + \sqrt{\epsilon_{r_2} - \sin^2\theta_i}} \\ R_\parallel &= \frac{\epsilon_{r_2}\cos\theta_i - \sqrt{\epsilon_{r_2} - \sin^2\theta_i}}{\epsilon_{r_2}\cos\theta_i + \sqrt{\epsilon_{r_2} - \sin^2\theta_i}}\,. \end{aligned} \qquad (26)$$

3.4. Implementation of polarization-dependent reflection coefficients.

If a ray is reflected by a wall or object and $R_\parallel$ and $R_\perp$ are to be used, the amplitude of the electric field parallel and perpendicular to the plane of incidence, have to be calculated.

Before this can be done, the polarization of the incoming ray has to be defined. This definition is shown in Fig. 17. Reference is the floor of the room, lying in the xy plane.

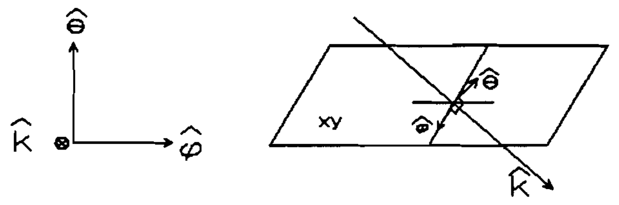


Fig. 17: Definition of polarization state of rays.

The ϕ polarization state of the electric field of the ray is the direction perpendicular to the ray propagation and parallel to the xy plane. It is defined positive if it points to the right, when looking in the the direction of propagation. The θ polarization is perpendicular to the direction of propagation and the ϕ polarization. It is positive if it has a positive z-component (see Fig. 18).

This definition is chosen, because in this way it is only necessary to rotate the two polarization components if a ray reflects against a wall of the room. The angle by which the polarization components have to be rotated, has to be calculated. In Fig.18 the situation is shown.

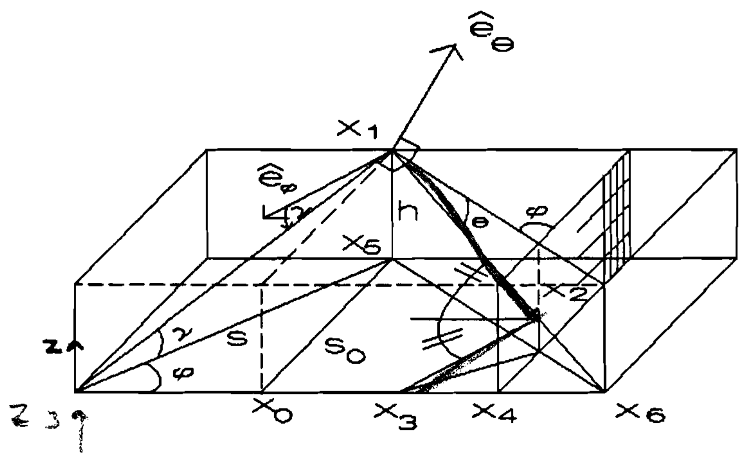


Fig. 18: Reflection against wall.

The points known in this situation are $\underline{x}_1$, $\underline{x}_2$ and $\underline{x}_3$, standing for the position of the transmitter, reflection point and receiver or second reflection point, respectively. The rays $\underline{x}_1$-$\underline{x}_2$ and $\underline{x}_3$-$\underline{x}_2$ define the plane of incidence. The angle over which the ϕ and θ have to be rotated is γ, with

$$\tan\gamma = \frac{h}{s} = \frac{h\sin\phi}{s_0} = \frac{z_1-z_5}{|\underline{x}_0-\underline{x}_5|}\,\frac{|\underline{x}_0-\underline{x}_6|}{|\underline{x}_5-\underline{x}_6|}\,. \tag{27}$$

The equations giving $\underline{x}_5$ and $\underline{x}_6$ are

$$\begin{aligned} x_5 &= x_1 \\ y_5 &= y_1 \\ z_5 &= z_1-z_3 \\ x_6 &= x_2 + \frac{L_2}{L_1}(x_2-x_1) \\ y_6 &= y_2 + \frac{L_2}{L_1}(y_2-y_1) \\ z_6 &= z_5\ , \end{aligned} \tag{28}$$

and the equations for $\underline{x}_4$ and $\underline{x}_0$

$$\underline{x}_4 = \frac{\underline{x}_3 + \underline{x}_8}{2} \qquad (29)$$
$$\underline{x}_0 = \underline{x}_4 + \frac{L_1}{L_2}(\underline{x}_4 - \underline{x}_8) \, .$$

The lengths L_1 and L_2 used above, are defined as

$$L_1 = |\underline{x}_1 - \underline{x}_2| \qquad (30)$$
$$L_2 = |\underline{x}_3 - \underline{x}_2| \, .$$

To be complete, the rotation of the two polarization components gives (Fig. 19)

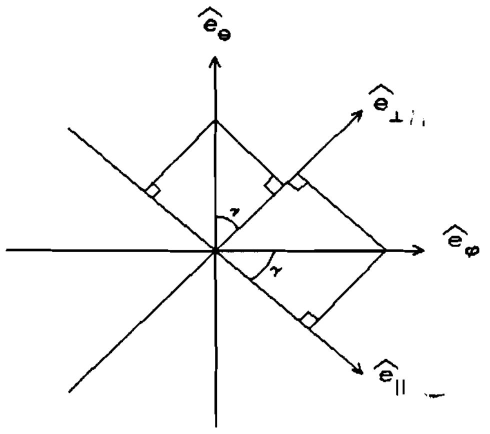


Fig. 19: Rotation of polarization components.

$$\begin{bmatrix} E_\perp \\ E_\parallel \end{bmatrix} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \begin{bmatrix} E_\theta \\ E_\phi \end{bmatrix} , \qquad (29)$$

and the angle of incidence θ, between the normal on the reflecting surface and the incident or reflected ray is

$$\tan\theta_i = \frac{|\underline{x}_2 - \underline{x}_4|}{|\underline{x}_3 - \underline{x}_4|} = \frac{\sqrt{(x_2-x_4)^2 + (y_2-y_4)^2 + (z_2-z_4)^2}}{\sqrt{(x_3-x_4)^2 + (y_3-y_4)^2}} \, . \qquad (32)$$

3.5. Received power.

3.5.1. Introduction.

Depending on the type of transmitter antenna, the polarization state of the electromagnetic wave can be linear, circular or elliptical. When some polarization state is transmitted, it will be affected by the multiple reflections, since the reflection coefficients for parallel and perpendicular polarization components differ.

This paragraph describes how the received power is calculated from the amplitudes of the arriving electric field components of the rays. This in case a single ray arrives at the receiver and in case multiple rays arrive at almost the same time. In this process the receiving antenna will play a major role.

3.5.2. Linear polarization.

A linearly polarized wave sent by the transmitter can be decomposed into a θ and ϕ component which have in general different amplitudes and zero phase difference. The total field can be described by

$$E_T - E_{\theta_T}\hat{e}_\theta + E_{\phi_T}\hat{e}_\phi \ , \tag{33}$$

with the time dependence $e^{j\omega t}$ suppressed.

The θ and ϕ components will be transformed by the programme, giving a field at the position of the receiver of

$$E_R - E_{\theta_R}e^{-jkL}\hat{e}_\theta + E_{\phi_R}e^{-jkL}\hat{e}_\phi \ , \tag{34}$$

where $k=2\pi/\lambda$ is the wavenumber and L is the length of the ray. The ray will arrive at some angle at the receiver. If the sensitivity of the receiver antenna for this direction is defined by F_θ and F_ϕ, which are the electric field components of the receiving antenna, the received power is

$$P \sim | F_\theta E_{\theta_R} - F_\phi E_{\phi_R} |^2 \ . \tag{35}$$

The - sign results from the difference of the ϕ direction of the antenna and the arriving ray.

3.5.3. Multiple linearly polarized waves.

If multiple rays arrive at the receiver at almost the same time, the received power can be calculated in the same way as in case two rays arrive. The two rays, which arrive from different directions, constitute an electric field

$$\begin{aligned} E &= E_1 + E_2 \\ &= \left(E_{\theta_1}\hat{e}_{\theta_1} + E_{\phi_1}\hat{e}_{\phi_1} + E_{\theta_2}e^{j\chi}\hat{e}_{\theta_2} + E_{\phi_2}e^{j\chi}\hat{e}_{\phi_2} \right) e^{-jkL_1}, \end{aligned} \tag{36}$$

with

$$\chi = kL_1 - kL_2 \tag{37}$$

and L_1 and L_2 the lengths of ray 1 and 2 respectively.

The four different components of E have to be multiplied with different factors because the two rays arrive from different directions and for each direction the θ and ϕ sensitivities, in general, are different.

The received power becomes

$$P \sim | (F_{\theta_1}E_{\theta_1} - F_{\phi_1}E_{\phi_1}) + (F_{\theta_2}E_{\theta_2} - F_{\phi_2}E_{\phi_2}) \; e^{j\chi} |^2 . \tag{38}$$

3.5.4. Elliptical Polarization.

In general a electromagnetic wave is elliptically polarized. If the transmitter transmits a wave with an E_θ and an E_ϕ component with a phase difference of δ, the total field for a single arriving ray at the receiver is

$$E = E_\theta e^{-jkL}\hat{e}_\theta + E_\phi e^{-jkL}e^{j\delta}\hat{e}_\phi \, . \tag{39}$$

Since the sensitivity of an antenna is described for left and right hand circular polarization, the arriving field has to be rewritten in a combination of these components. In Fig. 20 the components E_θ and E_ϕ are sketched.

If the θ and ϕ components are rotated and written in x and y components, the electric field becomes

$$\begin{aligned} E &= (E_\theta \sin\alpha + E_\phi \cos\alpha\cos\delta + jE_\phi\cos\alpha\sin\delta)\hat{e}_x \\ &\quad + (-E_\theta\cos\alpha + E_\phi\sin\alpha\cos\delta + jE_\phi\sin\alpha\sin\delta)\hat{e}_y \\ &= (E_1 + jE_2)\hat{e}_x + (E_3 + jE_4)\hat{e}_y \\ &= \sqrt{E_1^2 + E_2^2}\, e^{j\alpha_1}\hat{e}_x + \sqrt{E_3^2 + E_4^2}\, e^{j\alpha_2}\hat{e}_y . \end{aligned} \tag{40}$$

In (40) the term $e^{j\omega t - jkL}$ is suppressed.

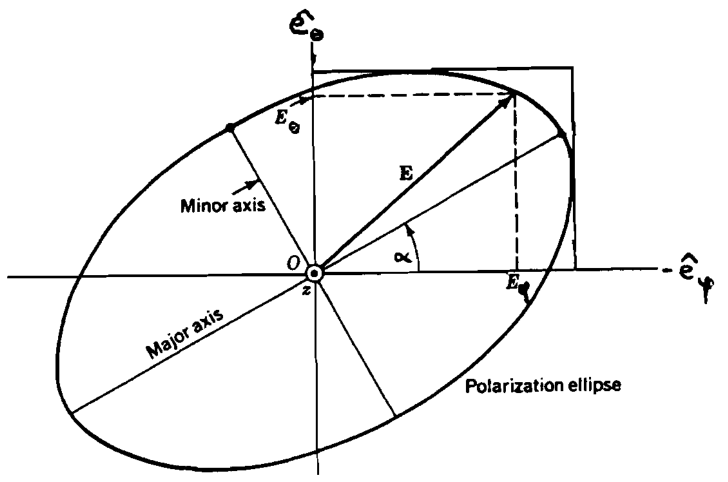


Fig. 20: Rotation of polarization components.

The x and y components have a phase difference of π/2 [11], if

$$\tan 2\alpha = \frac{2E_\phi E_\theta \cos\delta}{E_\phi^2 - E_\theta^2} . \tag{41}$$

If $\alpha_1 > \alpha_2$ then E is rewritten in

$$E = \sqrt{E_1^2 + E_2^2}\, e^{-jkL}\hat{e}_x + \sqrt{E_3^2 + E_4^2}\, e^{-jkL - j\frac{\pi}{2}}\hat{e}_y . \tag{42}$$

In this representation the phase α_1 of the θ component is neglected compared to kL.

The real electric field becomes now

$$E = E_x \cos(\omega t - kL)\hat{e}_x + E_y \sin(\omega t - kL)\hat{e}_y , \tag{43}$$

where

$$E_x = \sqrt{E_1^2 + E_2^2}\,,\qquad E_y = \sqrt{E_3^2 + E_4^2}\,. \tag{44}$$

This is a combination of a right E_R and left E_L circularly polarized wave with

$$\begin{aligned} E &= E_R + E_L\,,\\ E_R &= \tfrac{1}{2}(E_x + E_y)\cos(\omega t-kL)\hat{e}_x + \tfrac{1}{2}(E_x + E_y)\sin(\omega t-kL)\hat{e}_y\,,\\ E_L &= \tfrac{1}{2}(E_x - E_y)\cos(\omega t-kL)\hat{e}_x - \tfrac{1}{2}(E_x - E_y)\sin(\omega t-kL)\hat{e}_y \end{aligned} \tag{46}$$

If $\alpha_1 < \alpha_2$ then the amplitudes of the left and right circularly polarized wave are reversed.

The received power for a single arriving ray is now calculated by, first rewriting the E_θ and E_ϕ of the arriving field at the position of the receiver and the sensitivity of the receiving antenna in x and y components. And second, combining these components to get

$$P \sim |\,F_L E_L\,|^2 + |\,F_R E_R\,|^2\,, \tag{46}$$

where F_L and F_R are the sensitivities of the receiving antenna for left and right circularly polarized waves respectively.

3.5.5. Multiple elliptically polarized waves.

When two elliptically polarized waves arrive at almost the same time, the matter is more complicated. The electric fields of the waves at the position of the receiver are

$$\begin{aligned} E_1 &= E_{\theta_1} e^{-jkL_1}\hat{e}_{\theta_1} + E_{\phi_1} e^{-jkL_1} e^{j\delta_1}\hat{e}_{\phi_1}\,,\\ E_2 &= E_{\theta_2} e^{-jkL_2}\hat{e}_{\theta_2} + E_{\phi_2} e^{-jkL_2} e^{j\delta_2}\hat{e}_{\phi_2}\,. \end{aligned} \tag{47}$$

Both these components are rewritten in circular components. The x and y axis of these coordinate transformations, however, are not aligned. This causes a phase difference between the right circular component of E_1 and the right circular component of E_2. This difference is small and therefore neglected compared with kL_2-kL_1.

Thus the phase difference between the rays is

$$\Delta\phi = kL_2 - kL_1 \; . \tag{48}$$

Now the sensitivity of the receiver for left and right circular polarizations for the angles at which the rays arrive are calculated. These are F_{R_1}, F_{R_2}, F_{L_1} and F_{L_2} respectively, and the received power becomes

$$P_r = | \, F_{R_1} E_{R_1} + F_{R_2} E_{R_2} \, e^{j\Delta\phi} \, |^2 + | \, F_{L_1} E_{L_1} + F_{L_2} E_{L_2} \, e^{j\Delta\phi} \, |^2 \; . \tag{49}$$

3.6. Summary and further assumptions.

Summarizing the choices made for the simulation of the indoor radio channel gives

+ Geometrical Optics is used as a model for the indoor radio environment. This involves a direct ray and multiple reflected rays.
+ At every reflection the difference in reflection coefficients for both electric field polarization components is considered.
  It is also possible to define all kinds of polarization states at the transmitter (linear, circular or elliptical).

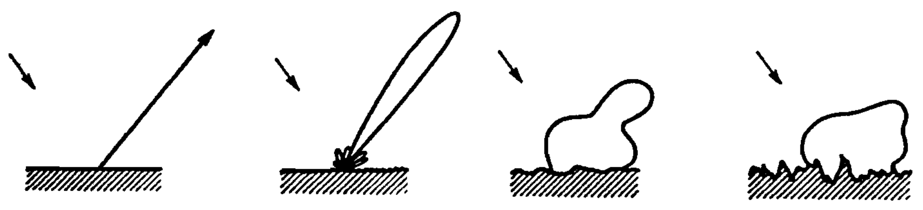
Now some further assumptions have to be made. First of all the transmitted electromagnetic wave will be spherical. Because probably some kind of electric dipole or open waveguide will be used for measurements and not a reflector antenna system.

It is allowed to use the Fresnel reflection coefficients for the spherical wave reflections, since half of the first Fresnel zone, which effectively contributes to the reflection, is very small. For this small angle of propagation the spherical wave will be considered locally flat.

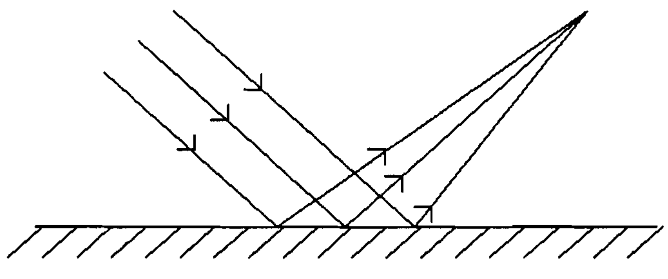
The objects will be considered to have no transmissivity. For metal objects this is obvious, but the walls, in general, will let an electromagnetic wave through. However the attenuation will be so large that the results of the simulation will not be affected by them.

The assumption that the walls do not let a electromagnetic wave through, implies that the indoor radio channel can only be simulated for a situation where transmitter and receiver are positioned in the same room.

Furthermore the walls will be assumed electromagnetically flat. This means that a plane wave incident on the wall at an angle θ, causes a reflected wave to travel away from the reflecting surface at the same angle. So rough surfaces will not be considered. The reason for this is that the whole concept of Geometrical Optics in this case does not hold anymore. An illustration is given in Fig. 21a, where a plane wave is incident on a rough surface. The reflected wave will have less power in the Snellius direction than in case the wave reflects against a flat surface. The remaining incident power will travel in other directions [12].



(a)



(b)

Fig. 21: Reflection against rough surfaces.

This behaviour can be accounted for by multiplying the ordinary reflection coefficient with some function. But if one ray incident on a surface is transformed into several rays with different directions, a ray incident on the surface at some other place, will also be able to reach the receiver (Fig. 21b).
Thus if the reflection coefficient is altered for the geometrical ray, other rays (not GO) have to be calculated also. For this reason the simulation is restricted to flat surfaces.

4. Multipath channel model.

4.1. Introduction.

Now that it is possible to calculate rays from transmitter to receiver, the characteristics of the channel have to be described in some well defined parameters. Before this can be done, the fundamentals of multipath channel theory underlying these parameters, have to be looked at. This signal theory is described in this chapter.

4.2. Multipath channel theory.

4.2.1. Introduction.

In a room, a radio channel between transmitter and receiver is constituted by several rays. These rays form a multipath channel. The response to a transmitted pulse is a train of pulses, as shown in Fig. 22. This is the first characteristic of a multipath channel.

The second characteristic of a multipath channel is the change in received pulse trains, caused by movement in the room. Although these movements will be slow, they can block for instance the direct ray from transmitter to receiver and in doing so, reduce the received power dramatically.

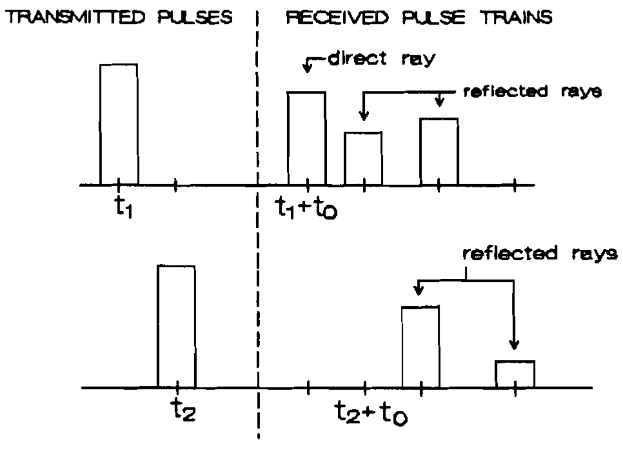


Fig. 22: Multipath character of radio channel.

The channel will now be described quantitatively [13].

4.2.2. Bandpass signal.

In the indoor radio system, digital information bearing signals are transmitted by some type of carrier modulation. The channel, through which the signal is transmitted, is limited in bandwidth to an interval of frequencies centred around the carrier. The signal transmitted through the channel is called a bandpass signal.
The representation of bandpass signals and systems is usually done in terms of equivalent low-pass waveforms.

A real valued signal s(t) with a frequency content concentrated in a band of frequencies in the vicinity of f_c, can be expressed in the form

$$s(t) = a(t) \cos [2\pi f_c t + \theta(t)], \tag{50}$$

with a(t) the envelope of s(t) and having a frequency content around f=0, f_c is the carrier frequency, and $\theta(t)$ the phase of s(t).

Now define

$$u(t) = a(t)\, e^{j\theta(t)}. \tag{51}$$

Combining (51) with (50) gives

$$s(t) = \mathrm{Re}\left[\, u(t)\, e^{j2\pi f_c t} \,\right]. \tag{52}$$

u(t) has a frequency content around f=0. It is called the equivalent low-pass signal of s(t). Furthermore the Fourier transform of s(t) is

$$\begin{aligned} S(f) &= \tfrac{1}{2} \int_{-\infty}^{\infty} [\, u(t)\, e^{j2\pi f_c t} + u^*(t) e^{-j2\pi f_c t} \,]\, e^{-j2\pi f t}\, dt \\ &= \tfrac{1}{2} [\, U(f-f_c) + U^*(-f-f_c) \,] \,. \end{aligned} \tag{53}$$

4.2.3. Linear bandpass system.

A linear system may be described by its impulse response h(t) or by its frequency response H(f), which is the Fourier transform of h(t).

Since h(t) is real

$$H^*(-f) = H(f) \ . \tag{54}$$

Define

$$C(f-f_c) = \begin{cases} H(f) & f>0 \\ 0 & f<0 \end{cases} , \tag{55}$$

and

$$C^*(-f-f_c) = \begin{cases} 0 & f>0 \\ \overset{}{\underset{H^*(-f)}{H(f)}} & f<0 \end{cases} \ . \tag{56}$$

Combining (54), (55) and (56) results in

$$H(f) = C(f-f_c) + \overset{C^*(-f+f_c)}{C^*(-f-f_c)} \ . \quad C(f \cdot {}^{f} c) \tag{57}$$

The inverse transform of H(f) yields h(t)

$$\begin{aligned} h(t) &= c(t)\, e^{j2\pi f_c t} + c^*(t)\, e^{-j2\pi f_c t} \\ &= 2 \operatorname{Re}\left[c(t)\, e^{j2\pi f_c t}\right] , \end{aligned} \tag{58}$$

where c(t) is the inverse Fourier transform of C(f). In general c(t) is complex-valued. It is called the impulse response of the equivalent low-pass channel.

4.2.4. Response of a bandpass system to a bandpass signal.

Suppose s(t), with its equivalent low-pass signal u(t), excites a system with impulse response h(t). The received signal is also a bandpass signal and therefore can be expressed in the form

$$r(t) = \operatorname{Re}\left[v(t)\, e^{j2\pi f_c t}\right] . \tag{59}$$

r(t) is related to s(t) and R(f) to S(f), by

$$\begin{aligned} r(t) &= \int_{-\infty}^{\infty} s(\tau)\, h(t-\tau)\, d\tau \\ R(f) &= S(f)\, H(f) \ . \end{aligned} \tag{60}$$

Substituting (53) for S(f) and (57) for H(f) results in

$$R(f) = \frac{1}{2} [U(f-f_c) + U^* (-f-f_c)] [C(f-f_c) + C^* (-f-f_c)]$$
$$= \frac{1}{2} [V(f-f_c) + V^* (-f-f_c)] , \quad (61)$$

since $U(f-f_c)=0$ and $C(f-f_c)=0$ for $f<0$ because s(t) is a narrowband signal and h(t) is the impulse response of a narrowband system, and where

$$V(f) = U(f)C(f). \quad (62)$$

The conclusion is that the indoor radio system can be considered as a baseband channel, if the equivalent low-pass signals are used.

4.3. Characterization of the indoor radio channel.

In a multipath channel there are multiple propagation paths. Associated with each path is a propagation delay and an attenuation factor. If the transmitted pulse is represented by s(t) then the received signal is

$$x(t) = \sum_{n=1}^{N} \alpha_n \, s(t-\tau_n) , \quad (63)$$

where α_n is the attenuation factor for the n-th path and τ_n is the propagation delay. These factors can change in time if the transmitter or receiver or a person moves within the room.

Substitution for s(t) and switching to equivalent low-pass signals results in the received signal

$$r(t) = \sum_{n=1}^{N} \alpha_n \, e^{-j2\pi f_c \tau_n} \, u[t-\tau_n] , \quad (64)$$

and the equivalent low-pass impulse response

$$c(t) = \sum_{n=1}^{N} \alpha_n \, e^{-j2\pi f_c \tau_n} \, \delta(t-\tau_n) . \quad (65)$$

In a measurement set-up, it is not possible to send a δ-pulse. In case of a transmitted pulse

$$b(t) = \begin{cases} 1 & 0 \le t \le \tau_p \\ 0 & \text{elsewhere} \end{cases} , \quad (66)$$

the received equivalent low-pass signal is

$$\begin{aligned} b(t) * c(t) &= \int_{-\infty}^{\infty} b(\tau)\, h(t-\tau)\,d\tau \\ &= \int_{-\infty}^{\infty} b(\tau) \sum_{n=1}^{N} \alpha_n\, e^{-j2\pi f_c \tau_n}\, \delta(t-\tau_n-\tau)\, d\tau \\ &= \sum_{n=1}^{N} \alpha_n\, e^{-j2\pi f_c \tau_n}\, b(t-\tau_n) \ . \end{aligned} \tag{67}$$

From (67), the very important power delay profile can be derived. It is defined as

$$p(t) = \left| \sum_{n=1}^{N} \alpha_n\, e^{-j2\pi f_c \tau_n}\, b(t-\tau_n) \right|^2 \tag{68}$$

in case of overlapping pulses. If the room is larger or equivalently the pulse shorter, the chance of two pulses overlapping becomes less and (49) can be simplified to

68

$$p(t) = \sum_{n=1}^{N} \alpha_n^2\ b^2(t-\tau_n)\ . \tag{69}$$

Two parameters derived from the power delay profile characterize the radio channel. First there is the average mean delay $\bar{\tau}$, as a measure of the delay between the transmitted pulse and the centre of received power. It is defined for non-overlapping pulses as

$$\bar{\tau} = \frac{\sum_{n=1}^{N} \alpha_n^2\, \tau_n}{\sum_{n=1}^{N} \alpha_n^2} \tag{70}$$

and in case of overlapping pulses as

$$\bar{\tau} = \frac{\int_{-\infty}^{\infty} \tau\, p(\tau)\, d\tau}{\int_{-\infty}^{\infty} p(\tau)\, d\tau}\ , \tag{71}$$

The other parameter is the average delay spread σ. This parameter is a measure for the width of the received pulse train. The definition of the average delay spread is

$$\sigma^2 = \overline{\tau^2} - \overline{\tau}^2 , \tag{72}$$

where

$$\overline{\tau^2} = \frac{\sum_{n=1}^{N} \alpha_n^2 \tau_n^2}{\sum_{n=1}^{N} \alpha_n^2} \tag{73}$$

for non-overlapping pulses and

$$\overline{\tau^2} = \frac{\int_{-\infty}^{\infty} \tau^2 p(\tau)\, d\tau}{\int_{-\infty}^{\infty} p(\tau)\, d\tau} \tag{74}$$

in case of overlapping pulses.

Furthermore a parameter can be defined to give an impression of the decay of received power with distance, compared to open space. In open space the received power is proportional to $1/r^m$, with $m=2$. In the indoor radio environment m can be calculated from

$$G = \frac{1}{r^m} , \tag{75}$$

where [14]

$$G = \sum_{n=1}^{N} \alpha_n^2 \tag{76}$$

with no overlap and

$$G = \frac{\int_{-\infty}^{\infty} p(\tau)\, d\tau}{\int_{-\infty}^{\infty} b^2(\tau)\, d\tau} \tag{77}$$

for overlapping pulses.

(76) and (77) imply an integration of the power delay profile to define G. Note that this is different from measurements in the mobile communications area where G is measured in case an unmodulated carrier is transmitted.

The parameter m becomes

$$m = {}^{r}\log\left(\frac{1}{G}\right) = -\frac{{}^{10}\log G}{{}^{10}\log r}\ . \tag{78}$$

To define m, G is calculated for several positions of the receiver with respect to the transmitter. Then G is plotted as a function of the distance between transmitter and receiver. Line fitting gives m for the particular room under consideration. m can be used as an indication of the power needed to be transmitted.

4.4. Implications of power delay spread.

It is clear that the delay spread causes symbols to overlap at the receiver, introducing intersymbol interference. In effect, the delay spread sets a limit on the transmission symbol rate in a digital mobile radio channel.

The result of [15] highlights the importance of the ratio σ/T_s, where T_s is the symbol time. It is shown that the delay spread shape is of no importance for $\sigma/T_s<0.2$, but can have a profound influence for $\sigma/T_s>0.3$. With this information the upper limit of the bit rate is derived as follows.

Define

$$\frac{\sigma}{T_s} = 0.2\ ,\quad T_s = \frac{1}{r_s}\ , \tag{79}$$

with r_s the symbol rate.

The bit rate is then kr_s, where k is a constant expressed in bit/symbol. The upper limit of the bit rate becomes

$$r_{b,max} = \frac{0.2\ k}{\sigma}\ . \tag{80}$$

This equation states that high bit rates can be achieved by either using a modulation method with a high value of k or keeping the delay spread σ low.

5. Accuracy simulations.

5.1. Introduction.

In writing software for some particular purpose, one has always to consider the possibility of making mistakes. These can be found by extensively running the programme over and over again. But even then, one cannot be sure that the programme is one hundred percent correct.
For these reasons, a few special situations are considered, and it is assumed that if the results for these situations are correct, the programme works properly in all situations alike the test situation.

Testing a programme, however, does not give any proof for the correctness of the simulation. Therefore simulation results are compared with measurements. The results of these measurements were accessible via the COST cooperation. The comparisons will show that the simulation gives a relatively accurate impression of the real situation.

Four programmes have been written. STRAAL3D for calculating the rays which travel from transmitter to receiver and applying the reflection coefficients. It can be used for calculating rays neglecting the polarization state of the waves and in case the polarization state is considered. In the first case the reflection coefficients have a constant value, independent of the angle of incidence of the waves. In the second case the reflection coefficients depend on the angle of incidence and on the polarization component.

The next programme is POWER. It is used to calculate the power delay profile of a particular situation and the parameters power gain and delay spread. It is used when the polarization is neglected.
If the polarization is considered POLPOWER is the programme which calculates the power delay profile and the parameters. Both POWER and POLPOWER use the output of STRAAL3D.
When several power delay profiles are calculated, CUMPOWER is used to combine these profiles to an averaged power delay profile.

The calculated power delay profiles are shifted to the origin of the time axis, since the receiver is not aware of the delay of the direct ray. Furthermore the profiles are weighted with the maximum value of the power delay profile. CUMPOWER therefore does not take into account the total received power, but only the shape of the profile.

There are a few items in the programmes that can be tested readily. First of all the calculations of the rays from transmitter to receiver can be checked by printing all coordinates of the reflection points on paper and check them by hand.

One can also define a room with the transmitter and receiver placed in such positions that the room is completely symmetrical. The calculated rays will reflect at places, which can be easily verified by hand. Another option is drawing the rays on the computer screen and check the symmetry of the rays. This works if the programme does not make the same mistake over and over again.

Another feature of the programmes has to be checked. This is the handling of the polarization components generated by the transmitting antenna at a reflection against a wall.
In this case nothing else helps than calculation by hand. However, there are a few possibilities to check the calculation of the handling of circularly polarized waves. In case such a wave reflects perpendicular against a wall, the polarization state is reversed.
One can also send a right circular polarized wave on a receiving antenna, which is only sensitive for right or left hand polarized waves. In the first case all arriving power is received and the received pulse is of the same shape as the one send. But in the second case no power is received what so ever.

The testing of the programmes is explained in this chapter. Not only the calculation of rays or the handling of polarization components are considered but also results of simulations are compared with measurements.

5.2. Checking rays.

Checking rays with the help of reflection points.

The checking of rays by hand is very time consuming and tiresome if the rays are multiple reflected. Therefore, tables of reflection coefficients are only given for the single and double reflected rays. The rays which reflect three and four times are here only checked by symmetry, since the programme just calls the same procedures for the calculation of these rays as for the calculation of the single and double reflected rays.

First an empty room is considered. It is symmetrical because only rectangular rooms are considered. After putting in the transmitter and receiver, the programme calculates the direct and single reflected rays as shown in Fig. 23.

The reflection points for the single reflected rays, together with the positions of the transmitter and receiver as well as the room configuration, are listed in Table 3.

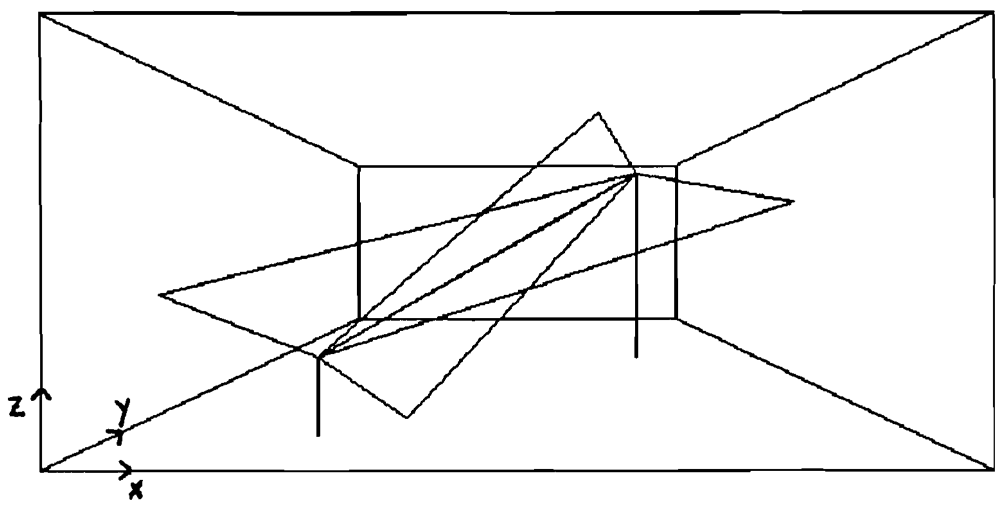


Fig. 23: Single reflected rays in empty room.

Table 3: Room configuration and coordinates of reflection points of single reflected rays.

coordinates transmitter	: 150 50 20.
coordinates receiver	: 450 150 80.
x-y coordinates room	: (0,0), (600,0), (600,200), (0,200).
height room	: 100.

coordinates of reflection points of single reflected rays:

no.	x	y	z
1	222.9	0.0	34.7
2	600.0	125.6	65.1
3	377.0	200.0	65.3
4	0.0	74.3	34.9
5	209.4	69.4	0.0
6	390.6	130.6	100.0

Since the rays are three dimensional, one has to look carefully at which wall the rays reflect. But it is clear that the single reflected rays are calculated correctly.

The double reflected rays are shown in Fig. 24 and the reflection points are listed in Table 4.

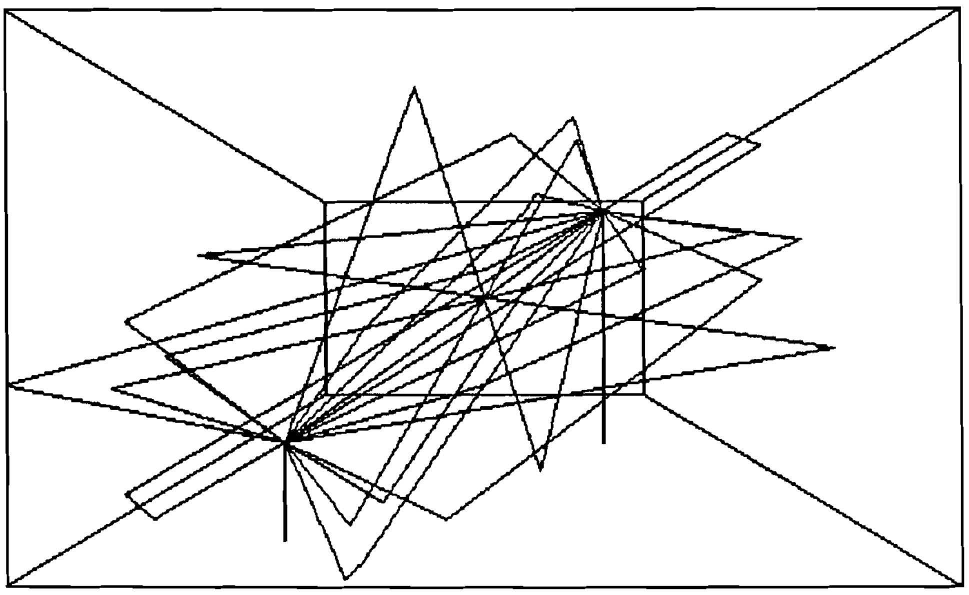


Fig. 24 Double reflected rays in empty room.

Table 4: Coordinates reflection points double reflected rays.

no.	x	y	z	x	y	z
1	296.0	0.0	34.7	600.0	101.3	65.1
2	198.7	0.0	29.9	401.3	200.0	70.1
3	2.0	0.0	34.7	0.0	0.6	~~7~~34.9
4	222.9	0.0	44.5	390.6	111.0	100.0
5	600.0	199.3	65.1	598.0	200.0	65.3
6	600.0	79.62	38.0	0.0	120.37	61.9
7	600.0	125.6	95.1	571.0	130.60	100.0
8	239.8	200.0	38.0	360.2	0.0	61.9
9	377.0	200.0	95.5	390.6	191.00	100.0
10	0.0	65.9	29.9	600.0	134.07	70.0
11	0.0	98.6	34.9	304.0	200.00	65.3
12	0.0	74.3	44.8	331.0	130.60	100.0
13	209.4	8.9	0.0	222.9	0.0	4.5
14	269.0	69.4	0.0	600.0	125.67	55.1
15	209.4	89.0	0.0	377.0	200.00	55.5
16	29.00	69.4	0.0	0.0	74.33	4.8
17	192.1	63.5	0.0	407.8	136.42	100.0
18	241.9	80.3	100.0	358.0	119.61	0.0

From Fig. 24 it appears to get difficult to follow the rays on the computer screen. Therefore the rays have been checked with the help of the coordinates of the reflection points listed in Table 4.

Checking rays by symmetry.

Although the symmetry technique, as mentioned before is not one hundred percent accurate, it gives a strong impression of the correctness of the programme. Therefore all reflection points of the three and four times reflected rays are checked by hand.
The transmitter and receiver are now in (151,99,49) and (451,101,51). See Fig. 25 and 26.

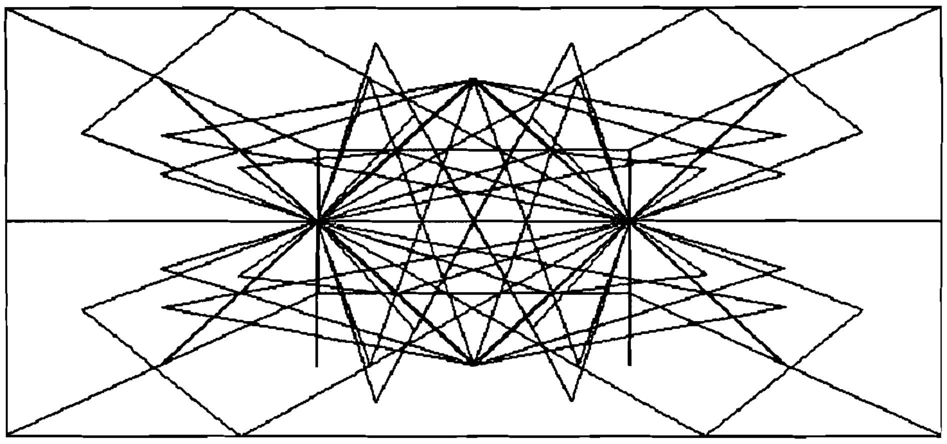


Fig. 25: Three times reflected rays in empty room.

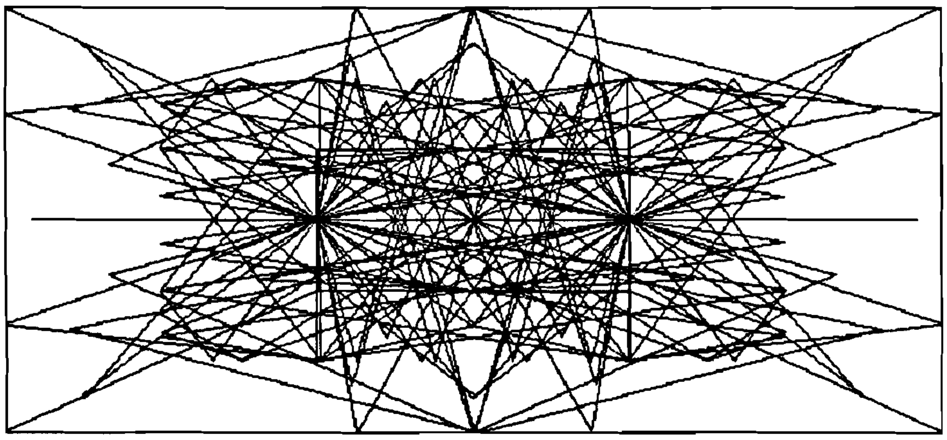


Fig. 26: Four times reflected rays in empty room.

The coordinates of the transmitter and the receiver are not chosen to be (150,100,50) and (450,100,50) respectively, because some rays would reflect exactly in the corner of two walls and the programme would not accept them as valid rays.

The symmetry is seen in Figures 25 and 26, but the accuracy of the calculation can be made more acceptable if an object is placed in the room declining the amount of rays. This is done in Fig. 27. The number of three times reflected rays is now reduced to 4 and the number of four times reflected rays is now 6.

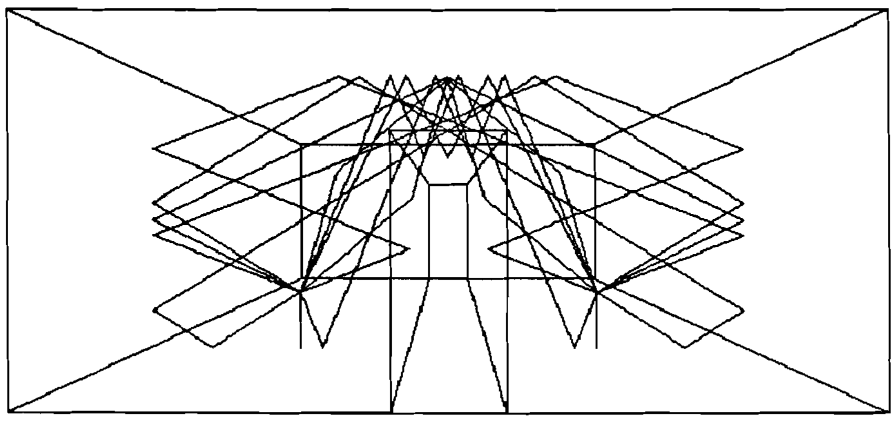


Fig. 27: Three and four times reflected rays in room with object.

Fig. 27 shows that the program correctly calculates the three and four times reflected rays. Furthermore is shown that the presence of an object is managed well by the programme.

5.3. Polarization handling of the programmes.

The polarization handling capabilities of the programme is tested in a situation where both transmitter and receiver are placed between two walls. For convenience the direct ray is not calculated and the transmitter and receiver are positioned at the same coordinates. The configuration is shown in Fig. 28.

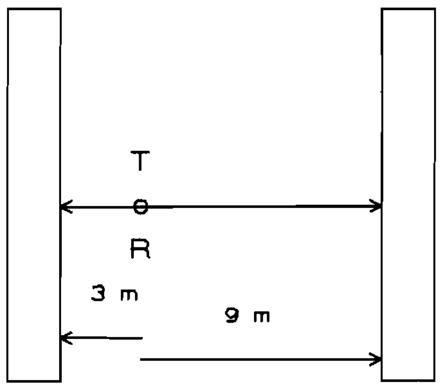


Fig. 28: Test configuration for polarization handling.

When a left-hand circular polarized wave is incident perpendicular on a wall, the reflected wave will be right-hand circularly polarized. This follows directly from the reflection coefficients for the two polarization components and the definition of the direction of the incident and reflected field components. This implies that the receiver cannot receive a single or three times reflected ray. Only the two and four times reflected rays are detected.

For the configuration in Fig. 28 the instants at which the rays arrive are listed below.

1 reflection;	right circular polarized :	20, 60 ns.
2 reflections;	left circular polarized :	80, 80 ns.
3 reflections;	right circular polarized :	100, 140 ns.
4 reflections;	left circular polarized :	160, 160 ns.

Due to the limitations of the computer the receiver will detect the single reflected ray, since two linear components are considered separately. The power carried by this ray is negligible, but a ray is still detected at t=20 ns. This point in time is shifted to t=0 ns in the power delay profile. The result of this shift is that the power carried by the two and four times reflected rays appear in the power delay profile at t=60 and t=140 ns respectively.

The calculated power delay profile is shown in Fig. 29.

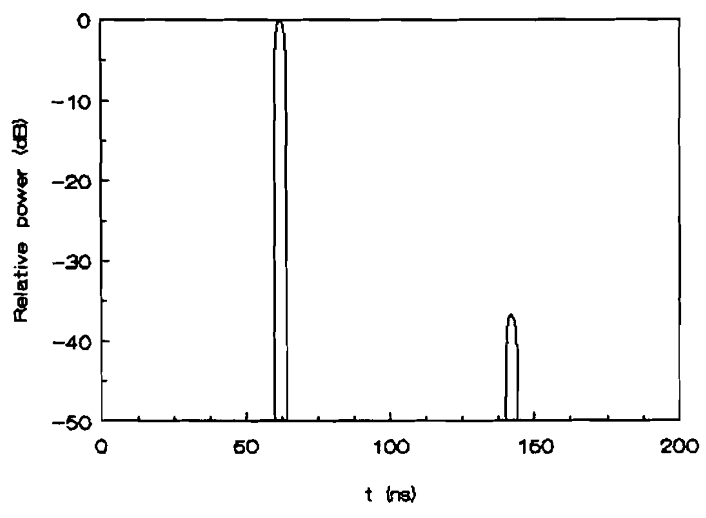


Fig. 29: Calculated power delay profile for test configuration.

The received power from the first right circular polarized circular wave is not shown in Fig. 29 since it is 200 dB down with respect to the received power at t=60 ns.

Although this test gives little proof for the correctness in other room configurations, it gives an example of the capabilities of the programmes.

5.4. Comparison simulation results with measurements.

The measurements available are conducted in a laboratory. The frequency used is 1.8 GHz. This means that any conclusion must be weighted with the discrepancies between the measurement conditions and the fundamentals of the computer programme. First of all there is the lower frequency. Geometrical Optics is valid for very high frequencies and the lower the frequency gets the more do diffraction rays add to the received power. Furthermore there is the problem of modelling the environment. What values of the dielectric permittivity has to be chosen for the different walls, and which wall is not considered in the input file because it is very thin.

A big problem is the position of the receiver in the configuration. See for a map of the measurement environment Appendix B. In some measurements set ups the receiver was placed in another room or very far away in a corridor. It is clear that the programme is in this case not able to predict the power delay profile accurately. Only in a room configuration where multiple rays reach the receiver, the calculated power delay profile can be relatively accurate.

In Fig. 30 the positions of the transmitter and receiver are shown with respect to the room configuration. The positions RX1, RX2 and RX3 are considered for the position of the receiver. The walls numbered 0 to 9 are the objects put into the input file.

A Horizontal source (Appendix C) performs the part of transmitting and receiving antenna. The pulse send is 8 ns wide and has a cosine shape. The polarization is linear. For detailed information about the polarization and radiation pattern see chapter 6.

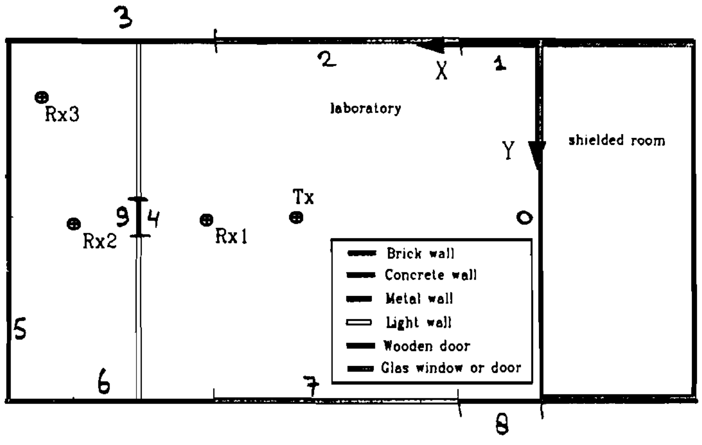
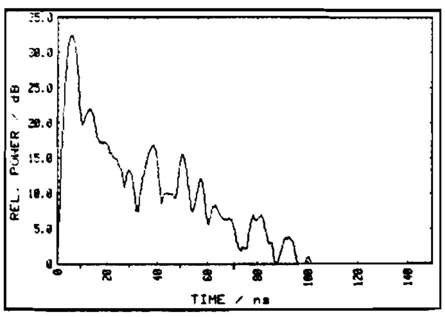
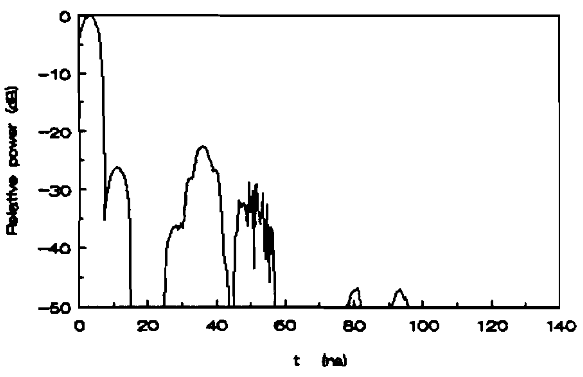


Fig. 30: Measurement environment.

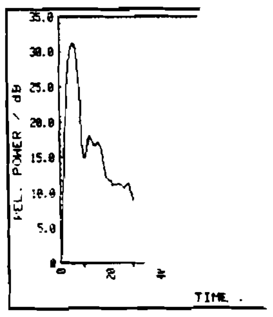
The calculated and measured power delay profile at the receiver in positions RX1, RX2 and RX3 are respectively shown in Fig. 31a, b and c.

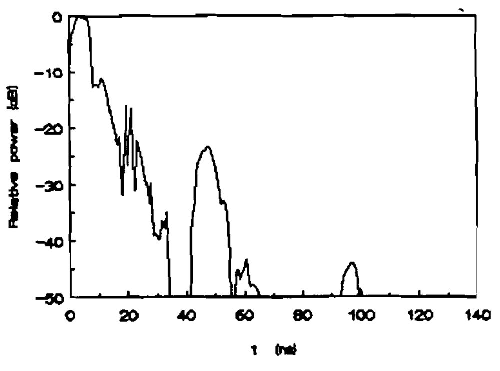




Measured profile. Calculated profile.

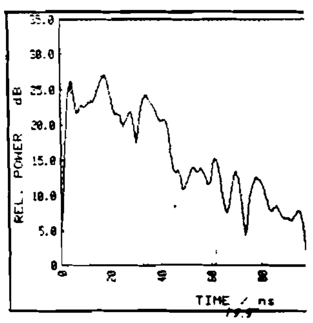
Fig. 31a: Power delay profiles for position RX1.

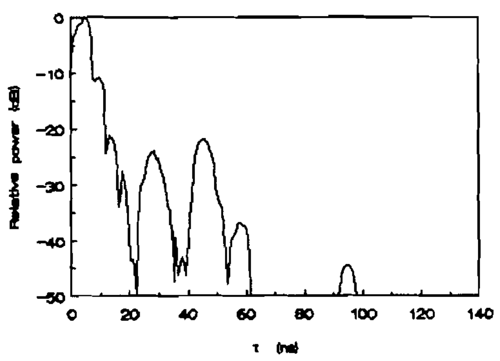




Measured profile. Calculated profile.

Fig. 31b: Power delay profiles for position RX2.





Measured profile. Calculated profile.

Fig. 31c: Power delay profile for position RX3.

In Fig. 31a, b and c the similarities between the measured and calculated power delay profiles are the occurrence of the peaks in the profile at the same point in time and the general shape of the profile. A difference is seen in the continuity of the measured profile and discontinuity of the calculated one. This is caused by diffraction and the presence of objects in the measurement set up, which are absent in the simulation. But all things considered, it is clear that the programme gives accurate information about the arrival of the mean peaks in the profile and the overall shape of the profile. The restriction of the programme is that the antennas must be placed in the same room and that there are multiple rays present.

6. Results of simulations.

6.1. Introduction.

Since the amount of simulations blows up with every new parameter introduced, the simulations are first focused on the 40*60*4 m^3 room defined in COST. This room is chosen as the radio system is most likely to be used in a large room. In the simulations empty rooms and rooms with objects within are considered.
Very important is the fact that not a single position of the receiver with respect to the transmitter is considered, but the power delay profile averaged over various positions of the receiver in a room, since the main interest lies in the profile, which is typical for the room in a real situation. Therefore plots are made of density functions of the delay spread and scatter plots of the received power versus distance.

Another aspect of the simulation is the difference of the power delay profiles calculated with and without consideration of polarization.

Furthermore cross-polarization effects for different types of antennas are considered.

6.2. Simulations without polarization consideration.

First the averaged power delay profile is calculated for the room mentioned above. The polarization of the waves is neglected. The transmitter is located 10 cm below the ceiling, in the middle of the room and the receiver is placed on 25 positions in the room at a height of 1.5m.

The programme uses an omnidirectional, horizontal or vertical radiating source (see Appendix C) for the transmitting and receiving antenna. The pulse width is chosen to be 5 ns, with a cosine shape. This width is chosen with the goal of a data rate of 150 MBit/s in mind.

The choice of reflection coefficients is very difficult. Values up to 0.2 (Table 2) are possible for non conductors. Therefore the values 0.01, 0.2 and 0.4 are used to obtain information over a larger area.

In Fig. 32 the averaged power delay profile is shown for omnidirectional transmitter and receiver antennas and reflection coefficients of 0.2 for the walls.

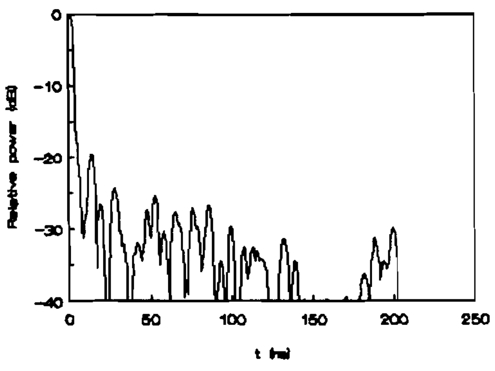


Fig. 32: Average power delay profile (R=0.2; Omnidirectional antennas).

From each individual profile the delay spread σ and the total received power is calculated. The total received power is weighted with a constant and the transmitted power. In Fig. 33 a plot of the probability function of the delay spread and a scatter plot of the received power versus distance is given. The total received power decay is a measure of the power that needs to be transmitted and the delay spread sets a limit on the maximum bit rate.

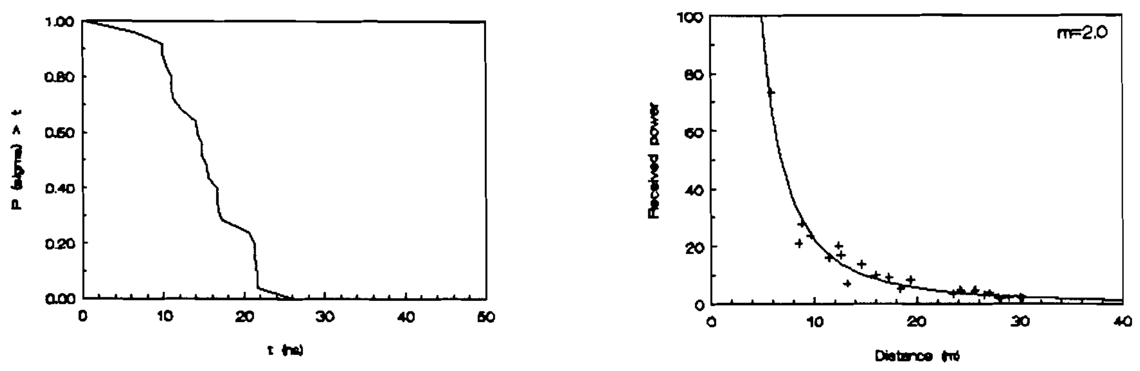


Fig. 33: Probability function σ and received power versus distance.

The parameter m, describing the decay $1/r^m$ of the received power with distance r between the transmitter and receiver is 2.0. This is the same as in an open space environment with only a direct ray. One could expect that m should be smaller than 2 because besides the direct ray also reflected rays are present. Since this is not the case the contribution of the reflected rays to the received power is negligible compared with the direct ray.

The calculated parameters σ and m depend on the reflection coefficients of the walls. Therefore the same calculation of the parameters is made for R=0.01 and R=0.4. For the averaged power delay profiles and plots of the probability of σ and the received power versus distance, see Appendix D. The results are listed in Table 5. The parameters σ_{50} and σ_{MAX} are respectively the values of σ which in 50% of the situations is exceeded, and the maximum value.

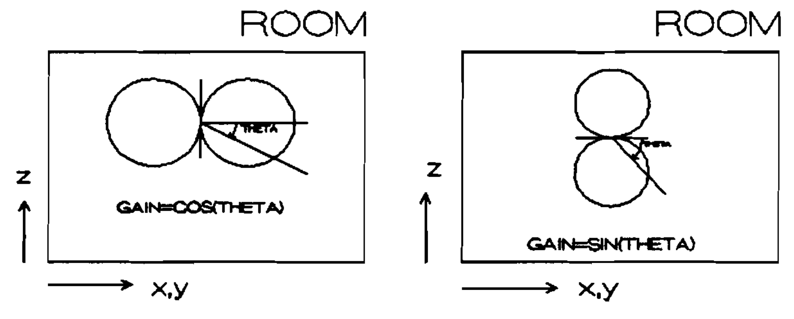
Table 5: Parameters σ and m as a function of the reflection coefficients of the walls.

R	σ_{50}	σ_{MAX}	m
0.01	1.3	1.5	2.0
0.2	15	26	2.0
0.4	34	57	1.9

An interesting result is that the delay spread is very much affected by the change in reflection coefficient of the walls and m is not. This can be explained by the presence of reflected rays which are weighted by their time delays for the determination of the power delay spread. This causes σ to increase. However the reflected rays do not carry enough power compared with the direct ray to cause m to decrease.

Applying the simulations for the same room configuration but different radiation patterns of the transmitter and receiver antenna results in values for σ and m as listed in Table 6. See Appendix D for the power delay profiles, plots of the probability function of σ and a plot of the received power versus distance. The reflection coefficients for the walls are taken to be 0.2.

The different shapes of the radiation pattern of the antennas and their orientations are shown in Fig. 34. The Horizontal source has a cos(θ) shape and the Vertical source a sin(θ) shape. θ is the angle between a ray and the xy-plane.



Horizontal radiating source. Vertical radiating source.

Fig. 34: Radiation patterns and their orientation.

Table 6: Parameters σ and m when different radiation patterns are used.

Pattern	σ_{50}	σ_{MAX}	m
Omnidirectional.	15	26	2.0
Horz. source.	16	26	1.8
Vert. source.	5	14	4.2

When Horizontal sources are used for transmitter and receiver antenna the parameter m becomes 1.8. This is caused by the radiation pattern of the antenna. The gain of this antenna has zeros in the directions perpendicular to floor and ceiling. If the receiver is placed nearly below the transmitter, the gain of the antennas for the direct ray will be low. When the receiver is placed further away the gain for the direct ray improves. This causes the received power to decay less with distance between the transmitter and receiver than in case an omnidirectional antenna is used.

In case a Vertical source is used, the gain of the antenna is maximum for the directions perpendicular to the floor and ceiling. If the receiver is moved away from the transmitter the gain of the antennas for the direct ray becomes less, resulting m to become larger than 2.
Furthermore the delay spread is very low. The reason for this is that rays which have to travel far and thus arrive relatively late will approach the receiver at angles for which the gain of the antenna is low. Therefore they are insignificant in the power delay profile. So the pulse broadening and the delay spread are small.

6.3. Simulations without polarization consideration.

When the polarization of the transmitted wave is considered, the reflectivity of walls has to be defined differently. In chapter 3 the coefficients are given for some different materials. Fig. 35 shows the absolute values of these reflection coefficients as a function of the dielectric permittivity ϵ and the angle of incidence θ_i between the normal on the reflecting surface and the incident ray.

The averaged power delay profile for the 40*60*4 m^3 room is calculated using linear polarization. The transmitter radiates an electric field with only a θ component. The direction of this component is defined in Fig. 17 of chapter 3. The antennas are first chosen to be omnidirectional and $\epsilon=2$.

The co-polarization at the receiver is the θ component of the electric field at the receiver. The power in this component constitutes the power delay profile. The power in the ϕ component of the electric field constitutes the cross-polarization. The pulse width is 5 ns.

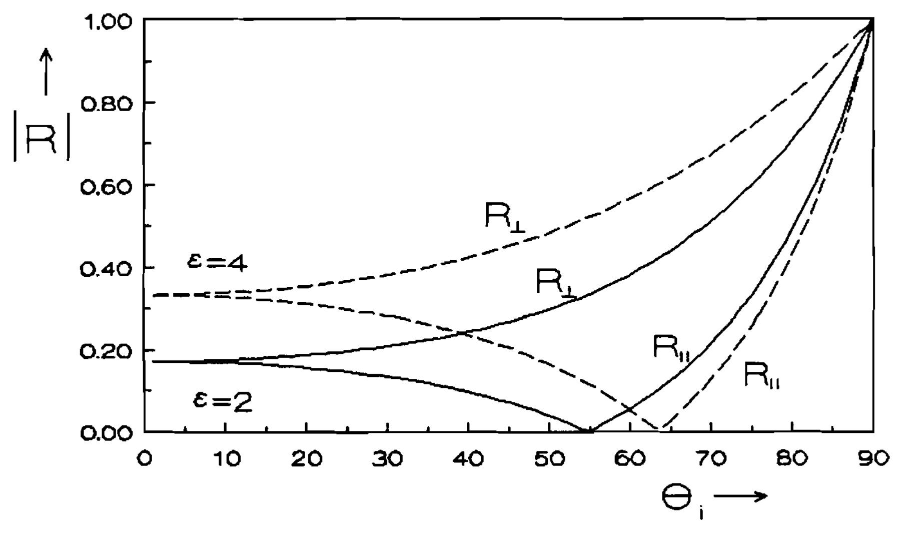
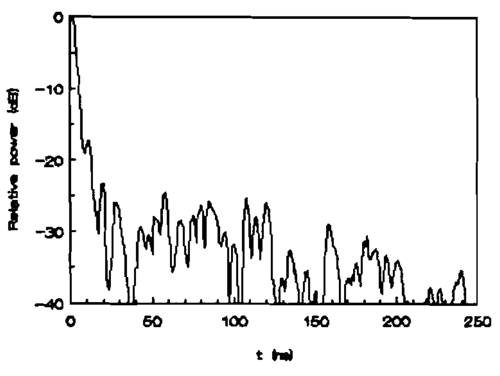


Fig. 35: Reflection coefficient as a function of ϵ and θ_i.

In Fig. 36 the averaged power delay profile for omnidirectional antennas is shown. The cross-polarization is shown in Fig. 37.



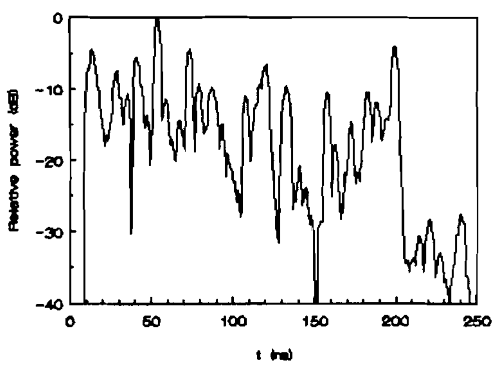


Fig. 36: Co-polarization power delay profile. Fig. 37: Cross-polarization power delay profile.

The averaged power delay profile of the co-polarization is very much the same as for the non-polarization calculations. The cross-polarization power delay profile is not comparable with earlier calculations. Furthermore the received power in the cross-polarization profile does not decay during the first 200 ns after the arrival of the first ray. This is a result of the increasing cross-polarization with number of reflections and thus distance and time.

This compensates for the decrease of total power carried by the multiple reflected rays. The result is a more or less constant cross-polarization power.
The average total power of the cross-polarization power delay profiles is 20.0 dB down with respect to the total co-polarization power.

A plot of the probability function of the power delay spread and the plot of the received power versus distance between transmitter and receiver are shown in Fig. 38.

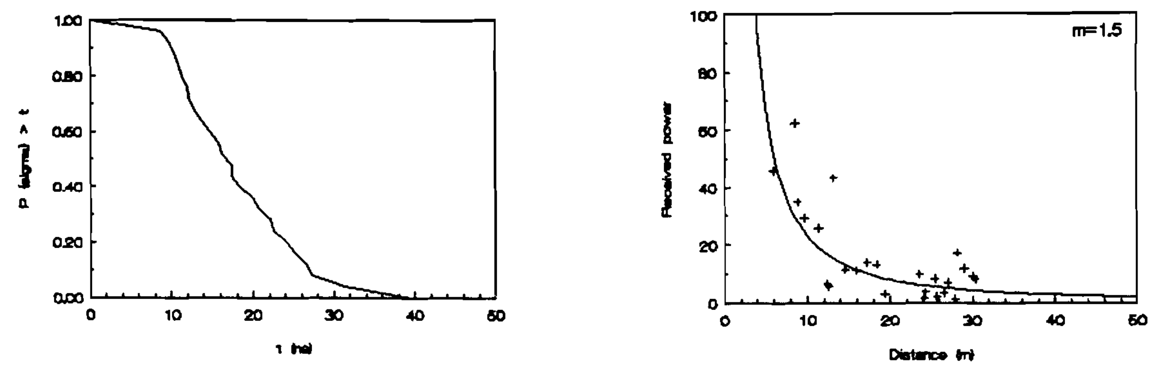


Fig. 38: Probability function of power delay spread and received power versus distance.

The values of σ_{50} in Fig. 38 is comparable with the values out of the non-polarization simulations, although exact comparisons can not be made because the reflection coefficient for the two polarization components of the electric field differ and are dependent of the angle of incidence. Furthermore the value of m must not be taken strictly because not all points in the scatter plot lie closely to the fitted line. The general result is that, if omnidirectional antennas are used, the values of σ and m do not differ much from the values calculated by the non-polarization programme.

The next step is the calculation of the averaged power delay profiles for other antenna radiation patterns and circular polarization. The power delay profiles of these simulations together with plots of the received power versus distance and probability function plots are shown in Appendix E.

The parameters σ_{50}, σ_{90}, σ_{MAX} and m as well as the average attenuation between the co- and cross-polarization of these simulations are listed in Table 7. σ_{90} is the value of σ which only in 10% of the situations is exceeded.

The definition used for cross-polarization is very straight forward. If the transmitting antenna radiates only a θ component and the receiving antenna is only sensitive for the same θ component, then all power arriving at the receiver in the ϕ component is cross-polarization power. The same goes for circular polarization. If transmitting and receiving antenna are only sensitive for left circularly polarized waves, the cross-polarization power is formed by the right circular waves arriving at the receiver.

For the Horizontal radiating source it is possible to have only a θ component of the electric field, but for a real Vertical radiating source this is not the case. However, the antenna patterns and polarization states chosen are used for convenient defining the cross-polarization power. Furthermore these choices are made for obtaining an impression of the physical characteristics of the radio channel. In a later stage different kinds of antenna radiation patterns can be defined to simulate a real configuration.

Table 7: Parameters σ_{50}, σ_{90}, σ_{MAX}, m and attenuation between co- (C) and cross- (X) polarization.

Radiation patterns	Polarization	σ_{50}	σ_{90}	σ_{MAX}	m	X/C (dB)
Omnidirectional	linear	16	27	39	1.5	-20.0
Omnidirectional	circular	12	26	43	2.0	-11.9
Horz. source	linear	17	27	39	1.3	-19.8
Horz. source	circular	12	26	43	1.7	-12.5
Vert. source	linear	8	14	23	2.9	-24.0
Vert. source	circular	6	12	15	3.2	-9.4

The values of σ in Table 7 for Omnidirectional and Horizontal sources do not differ much. The values of m are also similar for these radiators. These values are also similar with the ones calculated by the non-polarization programme.

The same conclusions can be drawn from the values of σ for the Vertical source. These are also comparable with the results of the non-polarization programme. Even the higher value of m for the Vertical source is confirmed.

There are however some slight differences between the values of m calculated by the two programmes and between m for linear and circular polarization. The values of m calculated by the polarization considering programme are smaller than the values of m calculated by the programme which neglects the polarization. This is caused by the difference in reflection coefficients of the walls.

If the receiver is placed further away from the transmitter, the angles between the rays and the normal on the reflecting surfaces approach 90°. For these angles the reflection coefficients become close to 1. Therefore the decay of received power with distance is less than in case the polarization is neglected and an absolute reflection coefficient of $R=0.2$ is used.

The difference between the values of m for linear polarization and circular polarization is caused by the fact that circular polarization is constituted by two linear polarizations. These two components have to be of the same amplitude to give only right or only left circularly polarized waves. However at every reflection the two linear components have different reflection coefficients causing the amplitudes to differ.

This implies that circular polarization not only looses power due to reflections but also due to amplitude differences between the components. A higher value of m is the result. This also explains why circular polarization is more sensitive to cross-polarization than linear polarization.
The difference in power loss of the co-polarization component due to cross polarization is only 0.4 dB.

A first conclusion can be drawn from the figures in Table 7. If a high bit rate is desired and the transmitted power is not a limiting factor, the antenna should be a Vertical source. This implies, however, that although the received power at short distances is similar for both antennas, the power needed to cover the whole room will be higher for a Vertical source then in case a Horizontal source is used. Therefore a more omnidirectional radiation pattern must be chosen. This is like a trade off between transmitted power and the achievable bit rate. High bit rates mean more power and less power means lower bit rates.

6.4. Effect of room configuration on power delay spread.

The use of an antenna with a radiation pattern like a Vertical source allows a higher bit rate, since the delay spread is less than in case a Horizontal or omnidirectional source is used. The disadvantage, however, is the power needed to be transmitted. If this is not a problem the use of a antenna with a pattern like the Vertical source is recommended.
In case the transmitted power is a problem, an omnidirectional radiating or Horizontal source must be used. For this situation the effect of the room configuration on the power delay spread is calculated. Only an omnidirectional source is considered since the results for this antenna are similar to the results for the Horizontal source.
The delay spread is calculated with the programme which neglects the polarization since these values do not differ much with the values of the delay spread calculated with the polarization considering programme.

The probability functions of the delay spread are calculated for three different rooms. The rooms are the ones defined in COST. The dimensions of the rooms are 40*60*4, 10*10*3 and 75*4*3 m^3. The effect of different reflection coefficients of the walls and the presence of metal objects within the room on the power delay spread are considered. The room configurations are shown in Fig. 39.

As mentioned before the antenna is omnidirectional. The pulse width is 1 ns and the pulse is rectangular for convenience. The results are listed in Table 8, 9 and 10. The probability functions are shown in Appendix F.

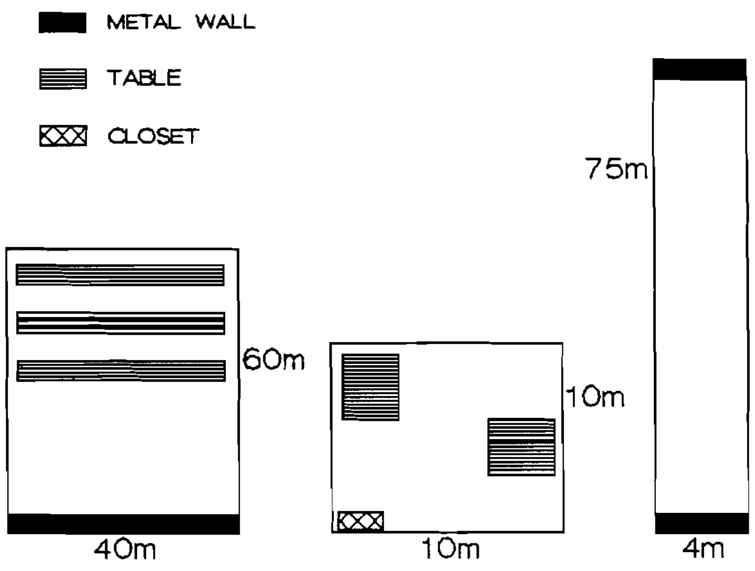


Fig. 39: Room configuration defined in COST.

Table 8: Power delay spread for 40*60*4 room.

Configuration	R_{walls}	R_{obj}	σ_{50}	σ_{MAX}
empty	0.2	-	16	19
empty	0.4	-	37	42
empty	0.6	-	64	69
empty	1.0	-	106	110
objects	0.2	1.0	37	62

Table 9: Power delay spread for 10*10*3 room.

Configuration	R_{walls}	R_{obj}	σ_{50}	σ_{MAX}
empty	0.2	-	3.6	3.7
objects	0.2	1.0	3.3	4.8

Table 10: Power delay spread for 75*4*3 room.

Configuration	R_{walls}	R_{obj}	σ_{50}	σ_{MAX}
empty	0.2	-	13	17
objects	0.2	1.0	122	156

The effect of the reflection coefficients on the power delay spread is easily extracted from Table 8. As expected the power delay spread increases with the reflection coefficients, since the reflected rays loose less power at every reflection.
The choice for reflection coefficients R = 1 is done for simulating a factory. In this case the delay spread becomes very large.

Objects within a room also cause the delay spread to increase. The values are more than a factor 2 greater. The reason for this is the same as mentioned above. The metal objects reflect rays with no loss of power. The power delay profile is broadened and the delay spread increased. This effect is also seen for the 10*10*3 m^3 room but in a very less dramatically way.

The room with dimensions of 75*4*3 m^3 is chosen to simulate the effect a corridor has on the power delay profile. The objects placed in it perform the role of metal doors at the end of the corridor. They cause a tremendous pulse broadening at the receiver resulting in a large value for the delay spread.

6.5. Analyzing the results.

First of all the power delay spread calculated by the polarization considering programmes and the programmes which neglect the polarization are similar.

Secondly the power delay profile is affected by the choice of the antenna used. A Vertical source clearly gives the lowest value for the delay spread. However this choice implies that more power needs to be send to cover the whole room. If this is a problem a more omnidirectional source must be chosen. The delay spread will be greater but the transmitted power will be less.

Using the relation between the delay spread and the bit rate of chapter 4, the achievable bit rate without equalization will be about 5 MBit/s for a Horizontal source in a room with dimensions of 40*60*4 m^3. If a Vertical source is used, the achievable bit rate becomes about 10 MBit/s. For smaller rooms the maximum bit rate increases. The room of 10*10*3 m^3 allows a bit rate of about 50 MBit/s.

The presence of metal objects in the room cause the delay spread to increase. This implies lower bit rates. For the room of 40*60*4m^3 the maximum achievable bit rate decreases to about 3 MBit/s.

7. Conclusions and recommendations.

Conclusions.

For mm-wave frequencies (around 60 GHz) a first impression of an indoor radio channel can be obtained by using Geometrical Optics as a model. Comparing results with measurements show that the simulation is a powerful tool to predict the characteristics of an indoor radio channel.
The most important parameters for characterizing an indoor radio channel are the delay spread and the rate of decay of received power with distance. The lower the delay spread gets, the higher the achievable bit rate will be.
The delay spread and the rate of decay of received power depend strongly on the room configuration and the antenna pattern of transmitter and receiver. The values of these parameters, calculated independently by a polarization considering programme and a programme which does not take the polarization into account, are similar.
If a vertical radiating source is used, the maximum achievable bit rate for a room with dimensions of $40*60*4$ m^3 is about 10 MBit/s. If the transmitted power is restricted, an omnidirectional or horizontal radiating source must be used since the transmitted power for these antennas will be less. The maximum achievable bit rate decreases in this case to 5 MBit/s.
Cross-polarization effects also occur. If a linearly polarized wave is transmitted then the power in the cross-polarization at the receiver is approximately 20 dB down with respect to the co-polarization power. In case circular polarization is used this figure is 10 dB. These values were calculated for a room with dimensions of $40*60*4$ m^3.

Recommendations.

Although expected to be small, the contribution of diffracted rays to the power delay profile and the power delay spread has to be looked at more carefully.
Furthermore, simulations have to be carried out for a room in which the walls and objects have different reflection coefficients.
Because the reflectivity of the walls and objects have significant influence on the delay spread, some measurements must be undertaken to obtain the reflection coefficients of different materials.
The positions of the transmitter and receiver and different antenna radiation patterns have to be looked at also. What, for instance, are the results for the power delay spread in case the transmitter is placed in an upper corner of the room?

8. Literature.

[1] R.G. Kouyoumjian, P.H. Pathak,
"A uniform GTD for an edge in a perfectly conducting surface",
Proc. IEEE, Vol 62, no 11, pp 1448-1461, November 1974.

[2] T. Griesser, C.A. Balanis,
"Dihedral corner reflector backscatter using higher order reflections and diffractions",
IEEE Tr. A&P., Vol 35, no 11, pp 1235-1247, November 1987.

[3] R. Tiberio, G. Pelosi, G. Manara,
"A uniform GTD formulation for the diffraction by a wedge with impedance faces",
IEEE Tr. A&P., Vol 33, no 8, pp 867-873, August 1985.

[4] T. Griesser, C.A. Balanis,
"Reflections, diffractions, and surface waves for an interior impedance wedge of arbitrary angle",
IEEE Tr. A&P., Vol 37, no 7, pp 927-935, July 1989.

[5] Y.T. Lo, S.W. Lee,
Antenna Handbook.
New York: Van Nostrand Reinhold Company, 1988.

[6] J.A. Aas,
"On the accuracy of the uniform GTD close to a 90° edge",
IEEE Tr. A&P., Vol 27, no 5, pp 704-705, September 1979.

[7] J.C. Arnbak, J. Dijk, M.H.A.J. Herben, J.T.A. Neessen,
"Radio en Radar",
College dictaat 5.653.0, Eindhoven University of Technology, 2nd edition, September 1987.

[8] S.E Alexander, G. Pugliese,
"Cordless communication within buildings: results of measurements at 600 MHz and 60 GHz",
Br. Telecom. Tech. J., Vol 1, pp 99-105, July 1983.

[9] S.T.S. Chia, D.R. Greenwood, D.C. Rickard, C.R. Shephard, R. Steele,
"Propagation studies for a point to point 60 GHz microcellular system for urban environments",
IEE Conf. on Comm., Birmingham, pp 28-32, April 1986.

[10] J. Dijk, A.C.A. van der Vorst, L.J.M. Wijdemans, F.M. Boumans, W.T.E. Vaessen,
"Microgolfdemping bij bouwmaterialen t.b.v. het stoorniveau en computerruimtes",
Telecommunication Division of Eindhoven University of Technology, Januari 1988.

[11] E. Hecht, A. Zajac,
Optics.
Addison Wesley Publishing Company, Menlo Park, California, U.S.A., 1974.

[12] P. Beckmann, A. Spizzichino,
The scattering of electromagnetic waves from rough surfaces.
Pergamon Press, London 1963.

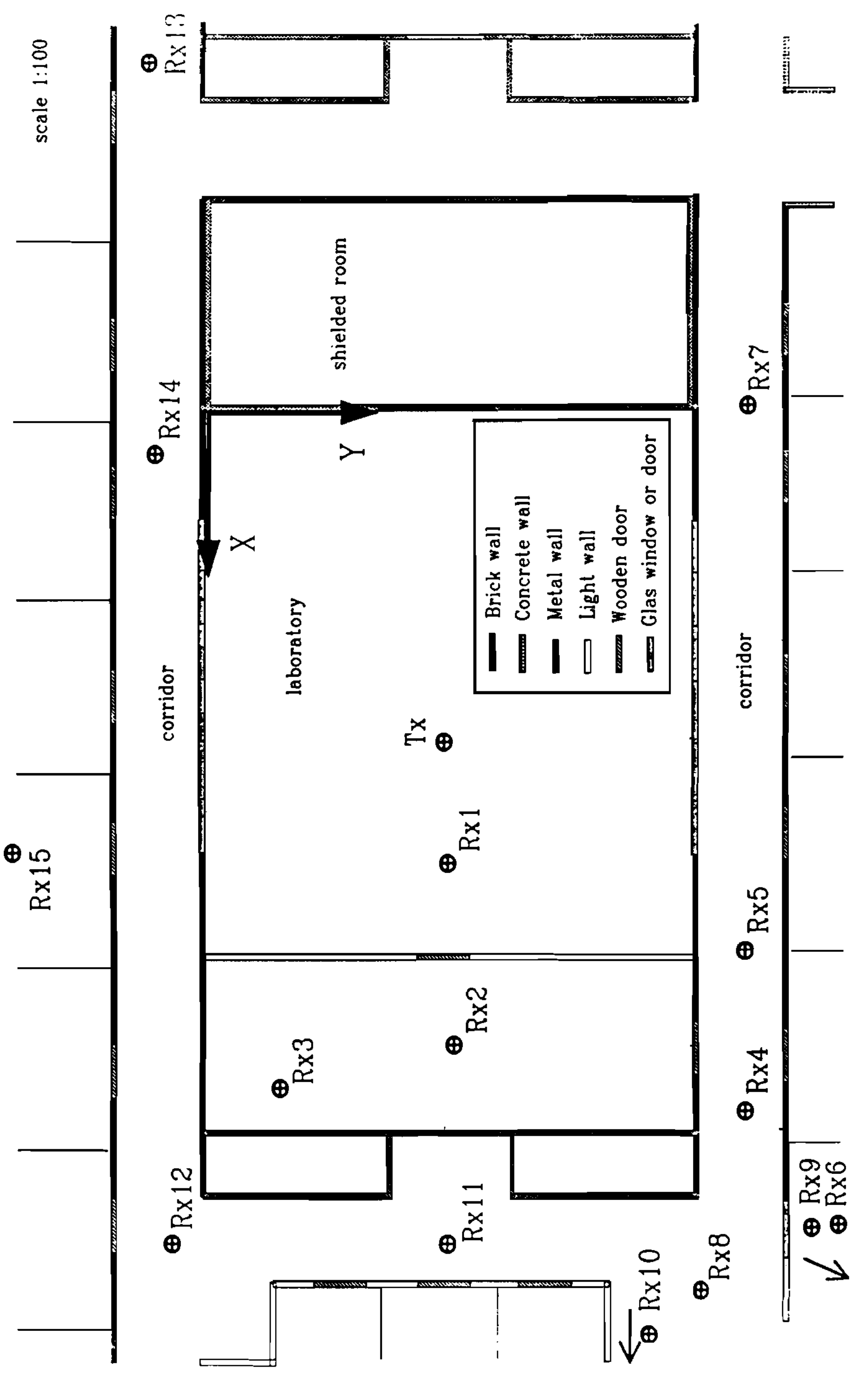
[13] J.G. Proakis,
Digital Communications.
McGraw-Hill, Singapore 1983.

[14] A.A.M. Saleh, R.A. Valenzuela,
"A statistical model for indoor multipath propagation",
IEEE J. on Sel. Areas in Comm., Vol SAC-5, no 2, pp 128-137, Februari 1987.

[15] B. Glance, L.J. Greenstein,
"Frequency selective fading effects in digital mobile radio with diversity combining",
IEEE Tr. on Comm, Vol COM-31, no 9, pp 1085-1094, September 1983.

Appendix A. Contents Appendix.

Appendix B. Measurement set-up for measuring power delay profiles within a laboratory.



Appendix C. Radiation patterns of Horizontal and Vertical sources.

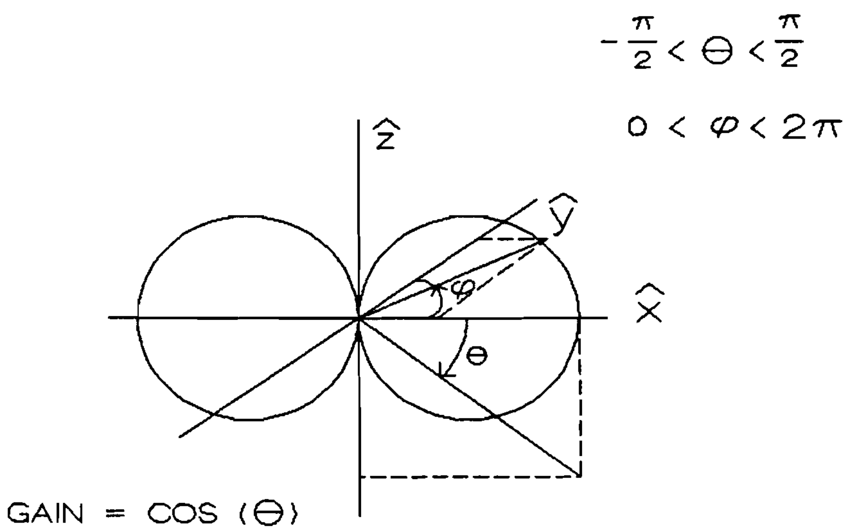


Fig. C1: Horizontal radiating source.

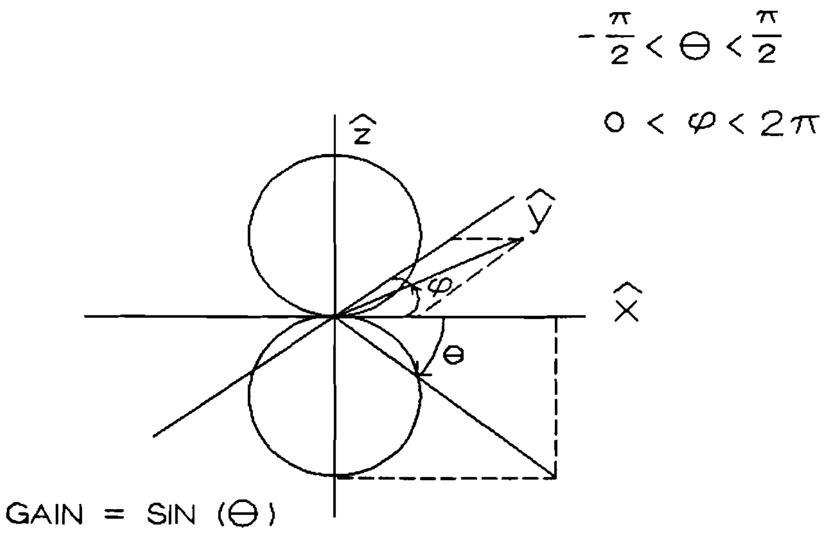


Fig. C2: Vertical radiating source.

Appendix D.

Results of simulations with non polarization programme.

Power delay profile calculated for empty room of 40*60*4 m^3, omnidirectional radiators and reflection coefficients of walls R=0.01.

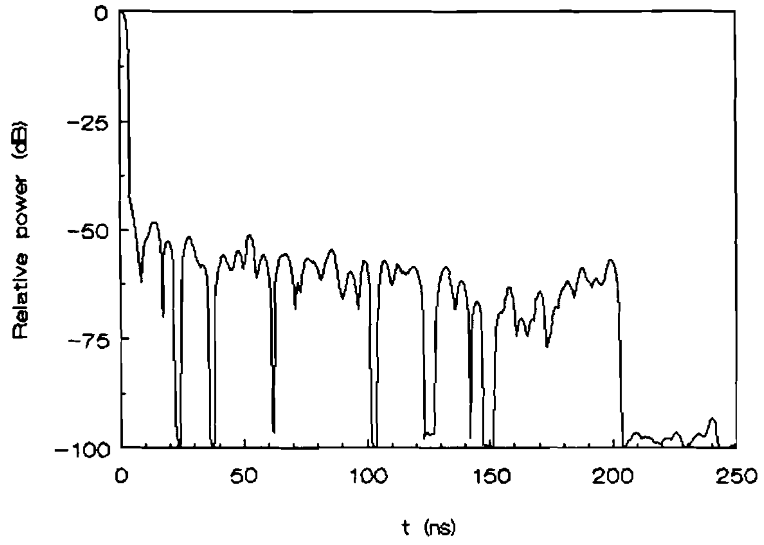
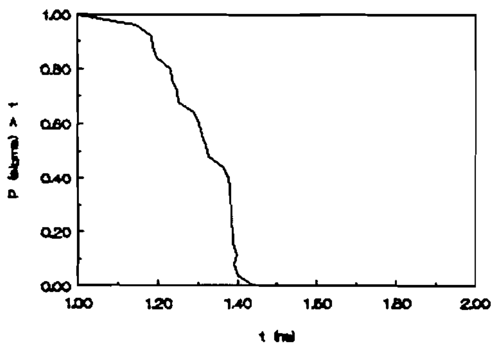


Fig. D1: Power delay profile: Omnidirectional antennas, R=0.01.

Power delay spread probability function and plot of received power versus distance.



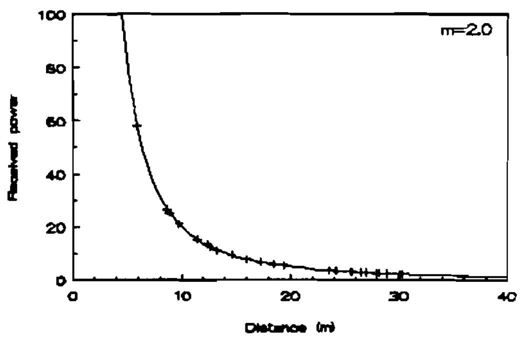


Fig. D2: Probability function of σ.

Fig. D3: Received power versus distance.

Power delay profile calculated for room of 40*60*4 m^3, omnidirectional radiators and reflection coefficients of walls R=0.4.

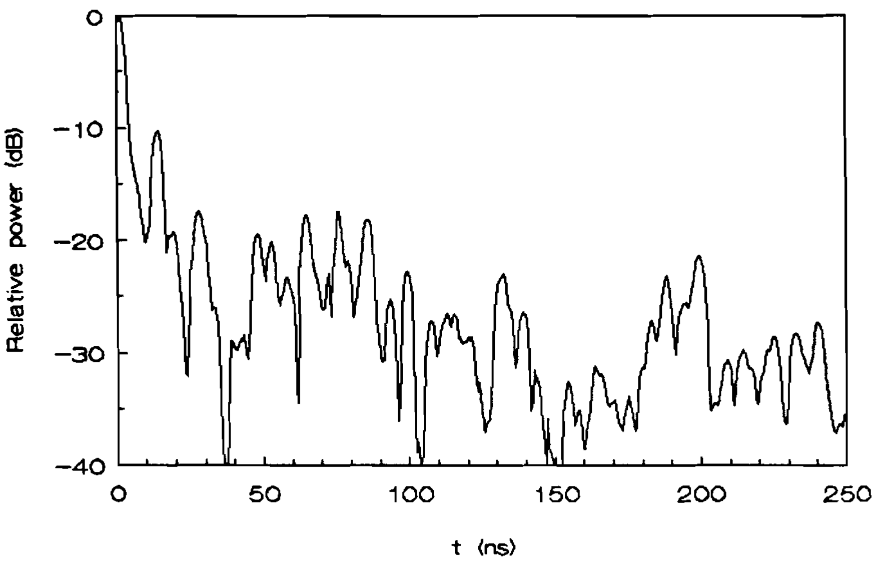
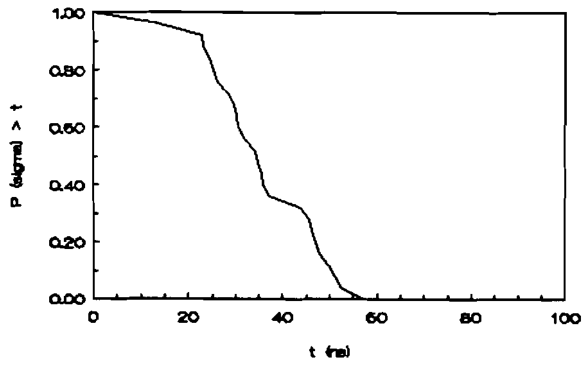


Fig. D4: Power delay profile: Omnidirectional antennas, R=0.4.

Power delay spread probability function and plot of received power versus distance.



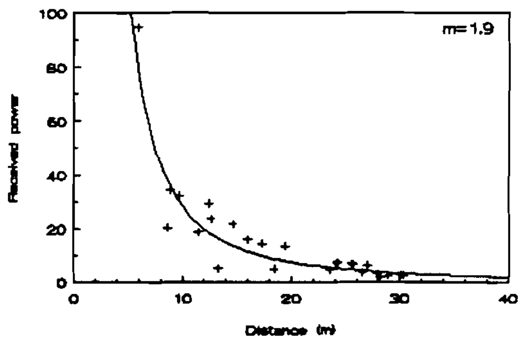


Fig. D5: Probability function of σ.

Fig. D6: Received power versus distance.

Power delay profile calculated for room of 40*60*4 m^3, Horizontal sources and reflection coefficients of walls R=0.2.

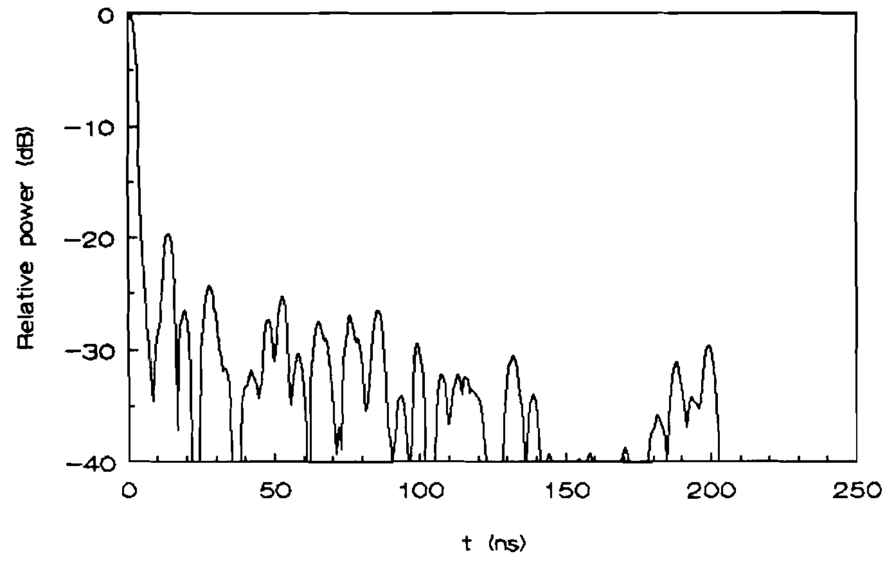
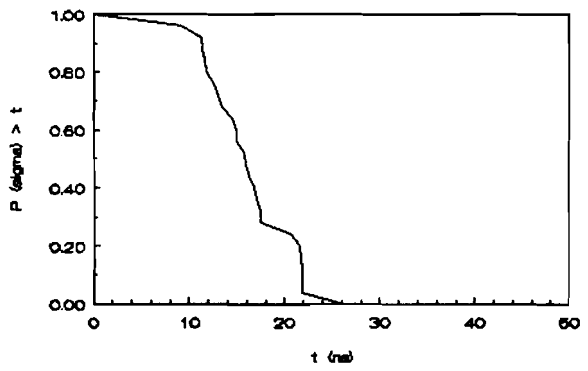


Fig. D7: Power delay profile; Horizontal sources, R=0.2.

Power delay spread probability function and plot of received power versus distance.



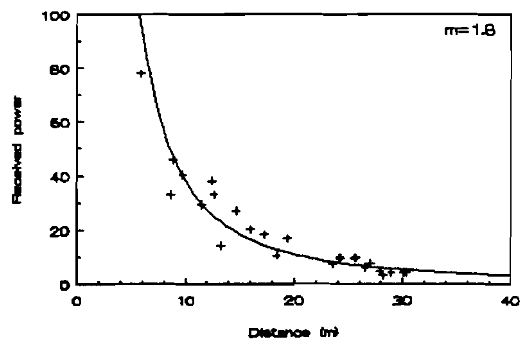


Fig.D8: Probability function of σ.

Fig. D8: received power versus distance.

Power delay profile calculated for room of 40*60*4 m^3, Vertical sources and reflection coefficients of walls R=0.2.

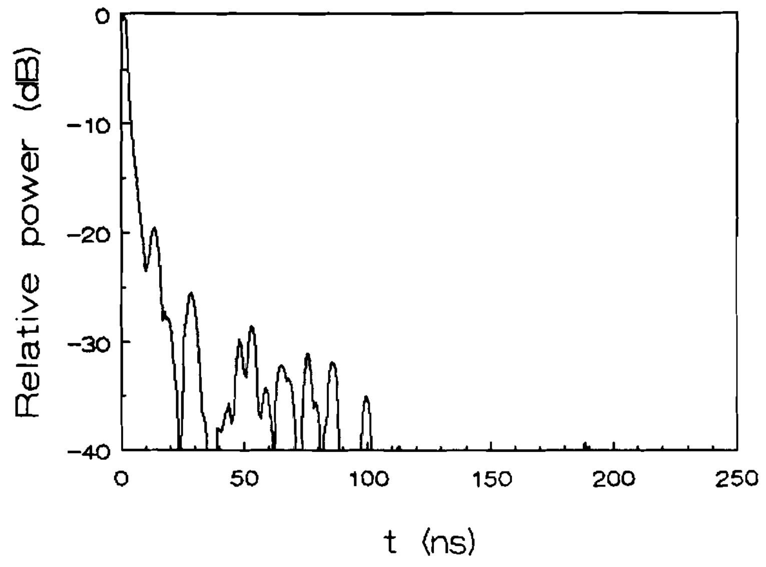
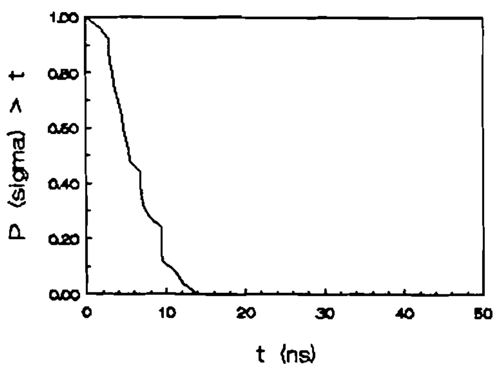


Fig. D10: Power delay profile; Vertical source, R=0.2.

Power delay spread probability function and plot of received power versus distance.



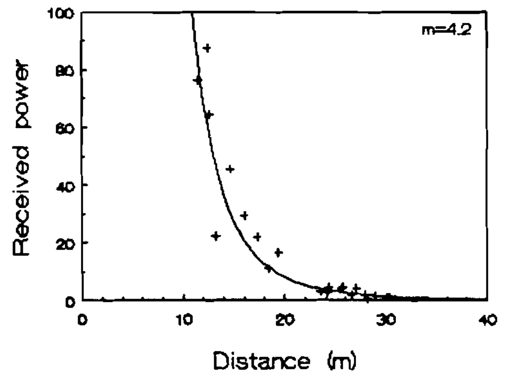


Fig. D11: Probability function of σ.

Fig. D12: Received power versus distance.

Appendix E. Results of simulations with polarization considering programme.

Power delay profile, probability function of σ and received power versus distance for empty room of 40*60*4 m^3. Results for circular polarization and omnidirectional antennas.

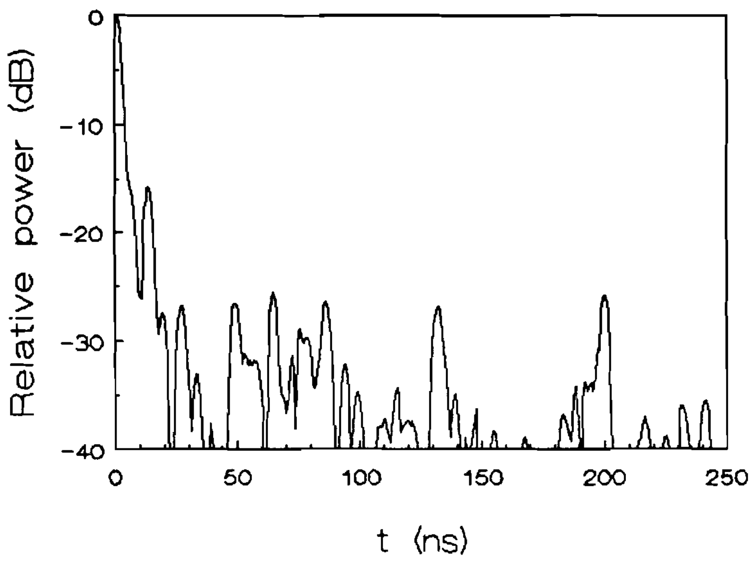


Fig. E1: Power delay profile co-polarization.

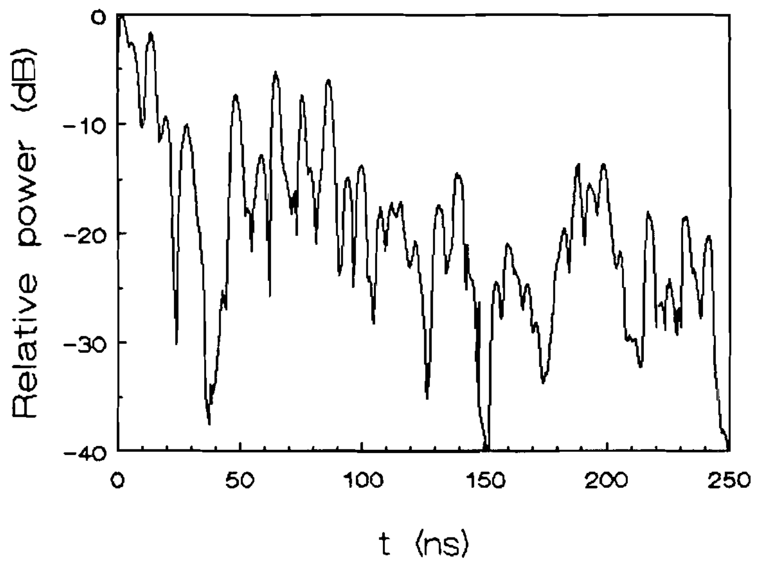


Fig. E2: Power delay profile cross-polarization.

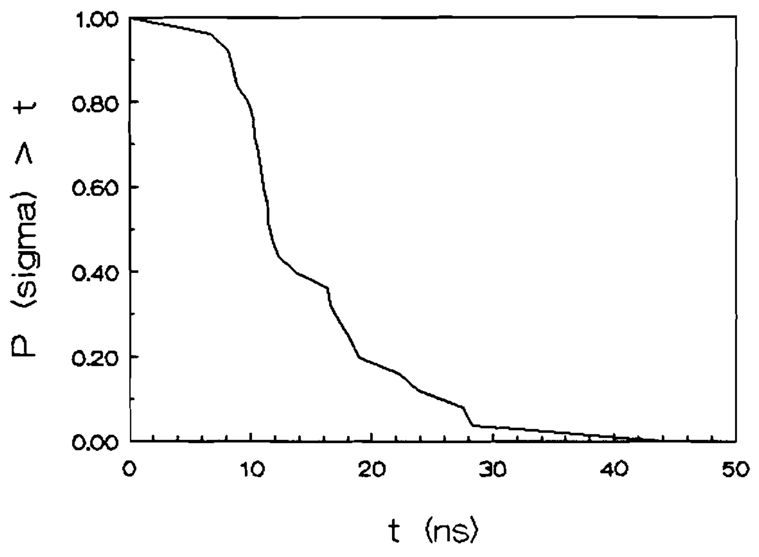


Fig. E3: Probability function of σ of co-polarization.

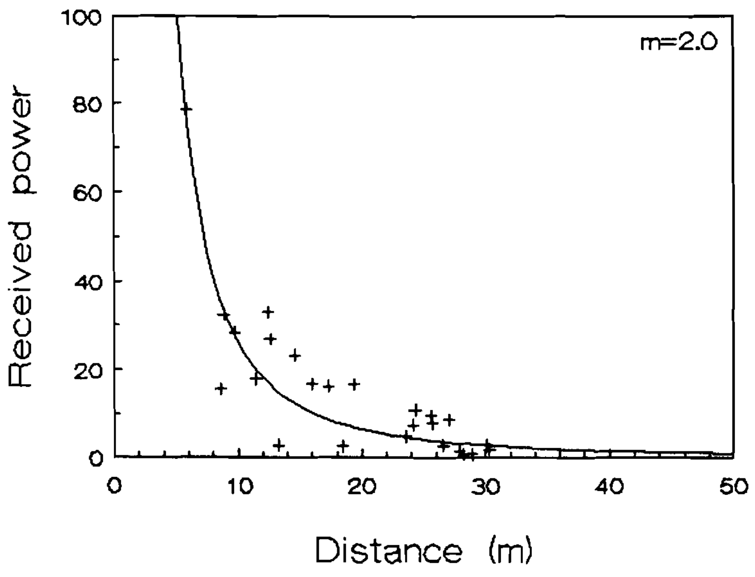


Fig. E4: Received power versus distance of co-polarization.

Results for linear polarization and Horizontal sources.

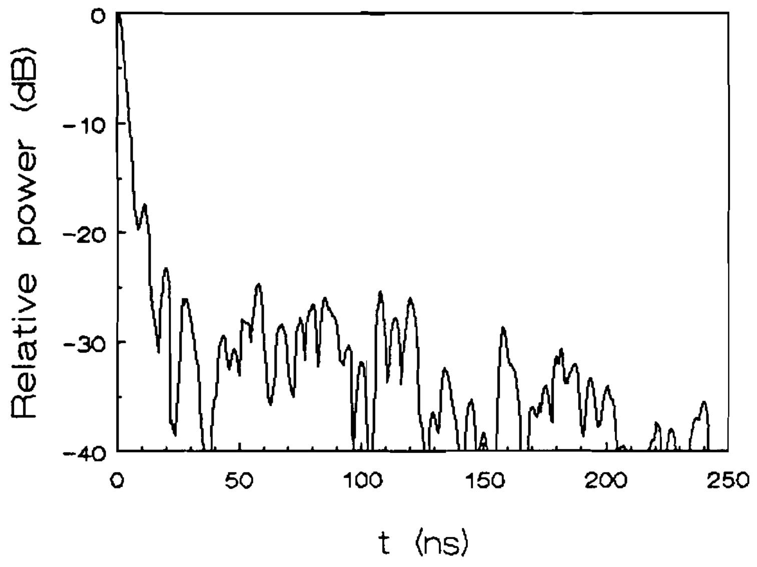


Fig. E5: Power delay profile co-polarization.

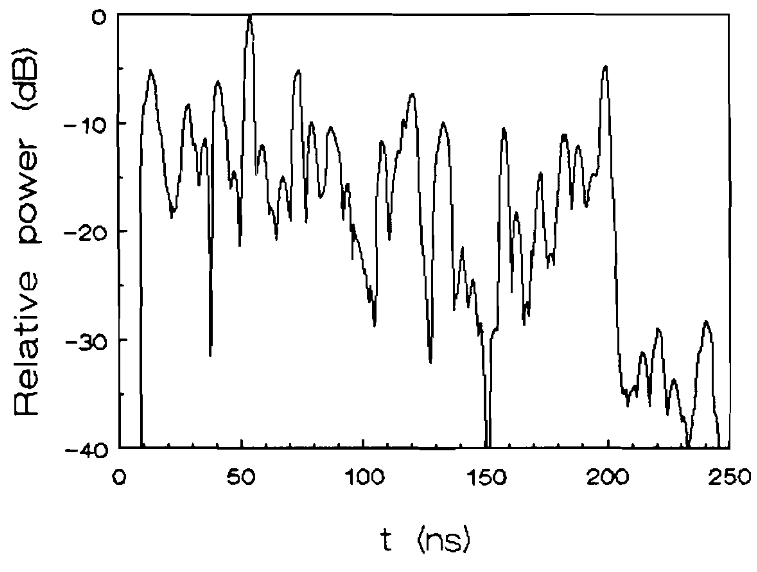


Fig. E6: Power delay profile cross-polarization.

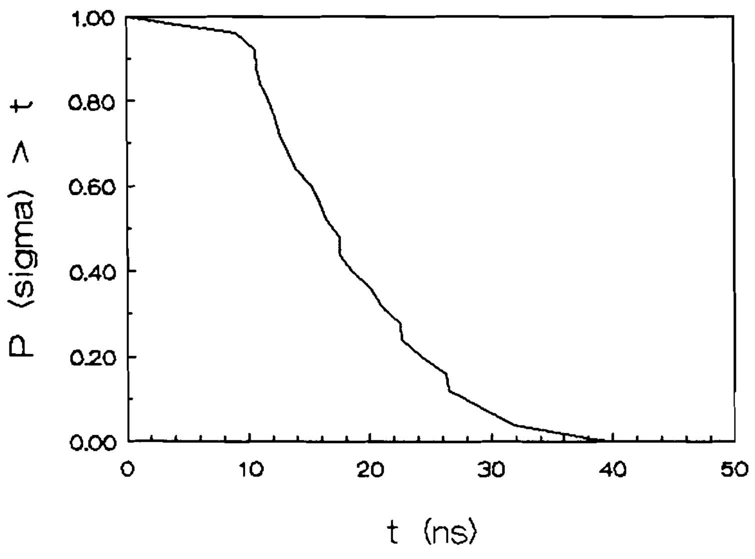


Fig. E7: Probability function of σ of co-polarization.

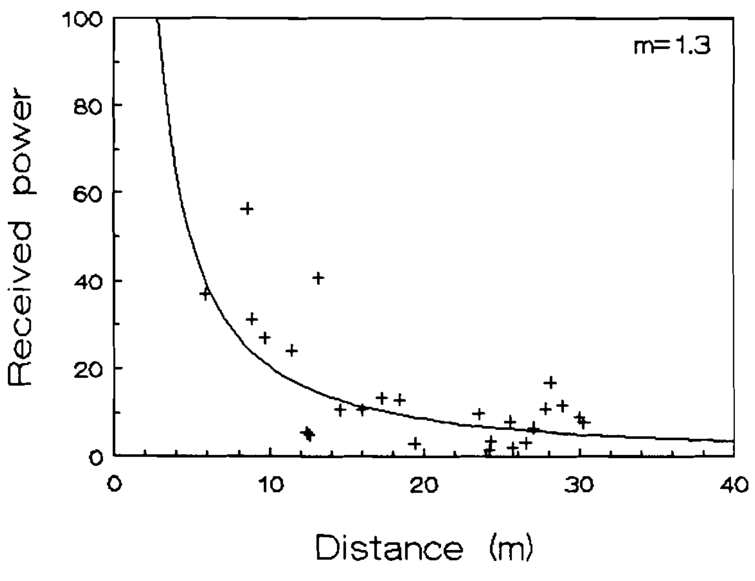


Fig. E8: Received power versus distance of co-polarization.

Results for circular polarization and Horizontal sources.

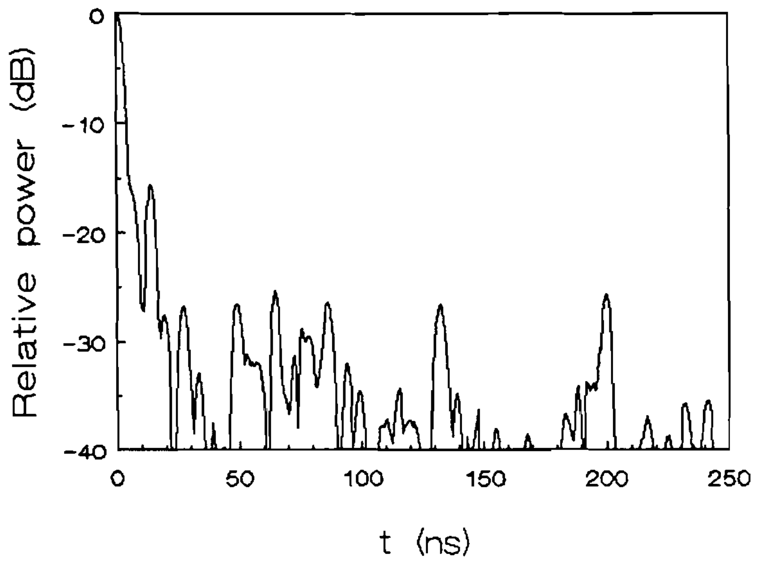


Fig. E9: Power delay profile co-polarization.

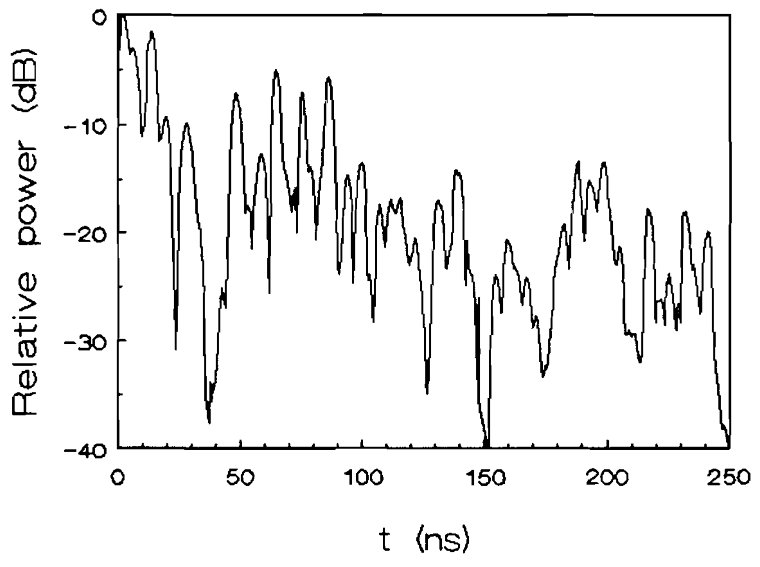


Fig. E10: Power delay profile cross-polarization.

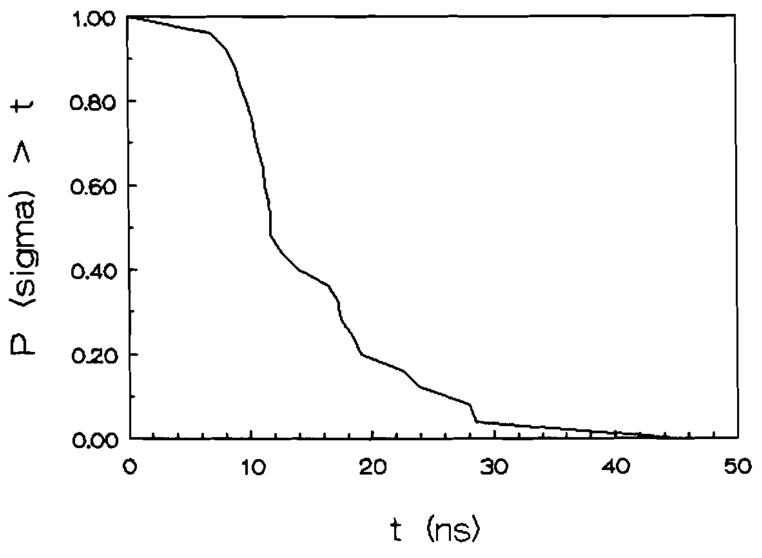


Fig. E11: Probability function of σ of co-polarization.

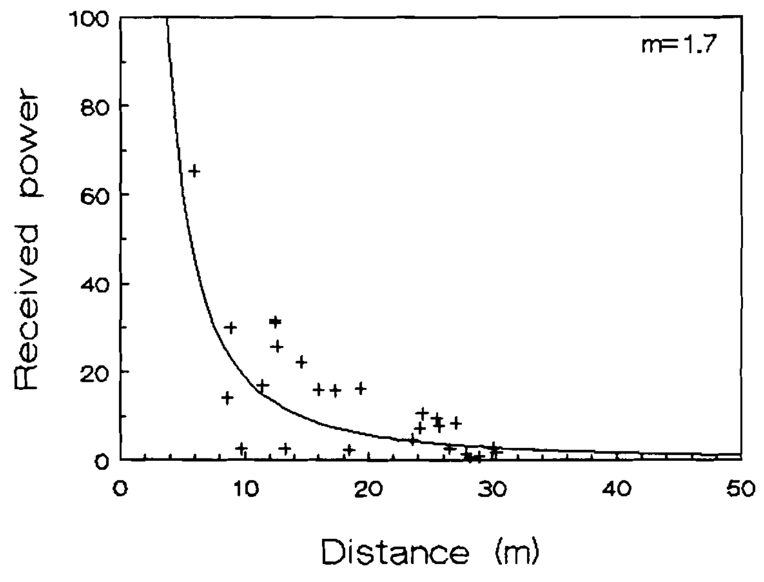


Fig. E12: Received power versus distance of co-polarization.

Results for linear polarization and Vertical sources.

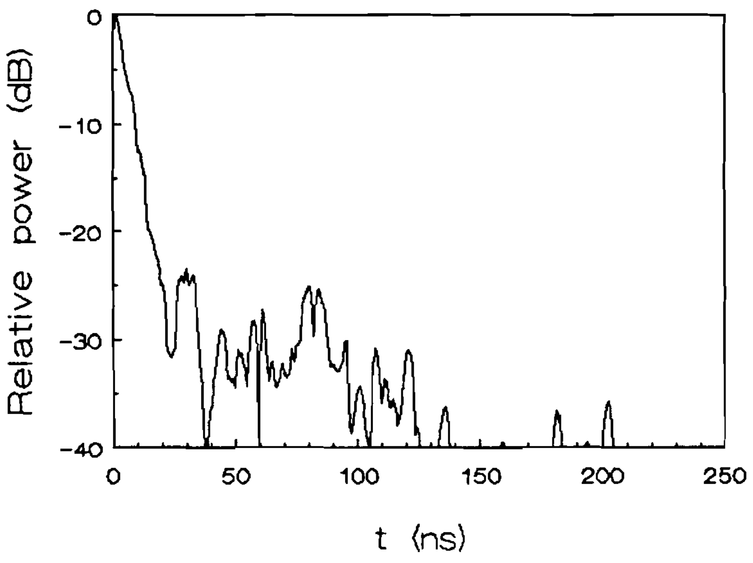


Fig. E13: Power delay profile co-polarization.

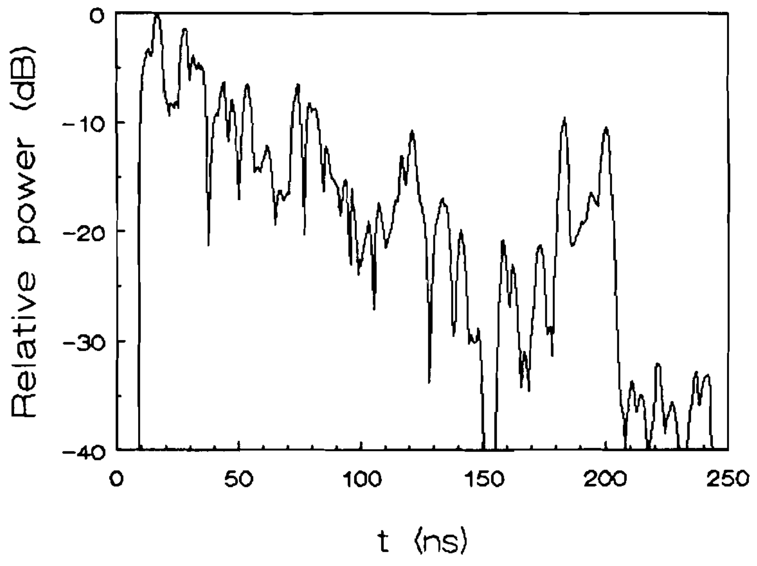


Fig. E14: Power delay profile cross-polarization.

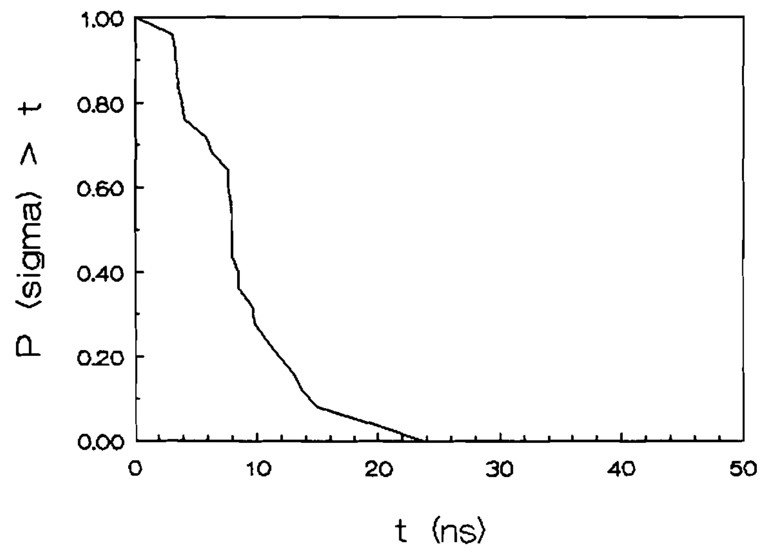


Fig. E15: Probability function of σ of co-polarization.

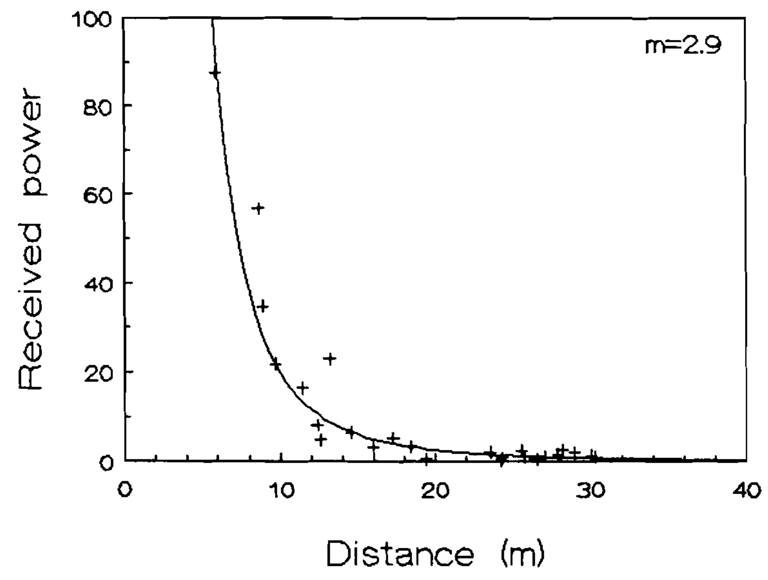


Fig. E16: Received power versus distance of co-polarization.

Results for circular polarization and Vertical sources.

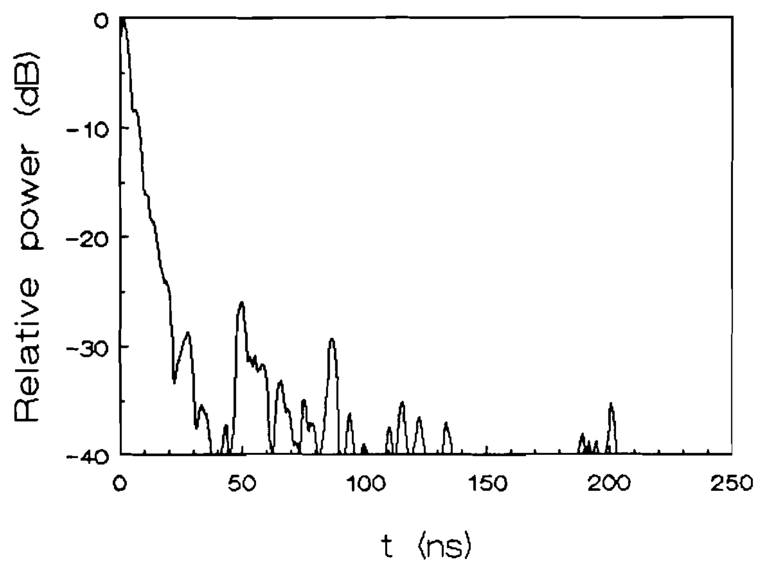


Fig. E17: Power delay profile co-polarization.

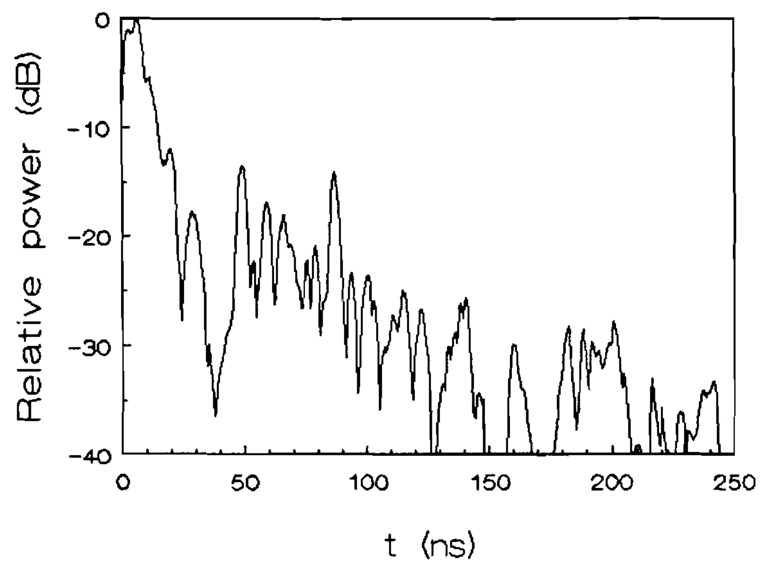


Fig. E18: Power delay profile cross-polarization.

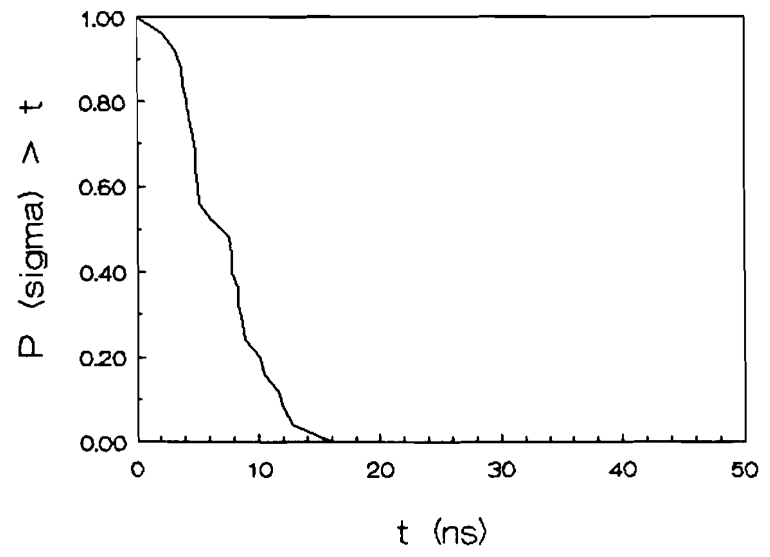


Fig. E19: Probability function of σ of co-polarization.

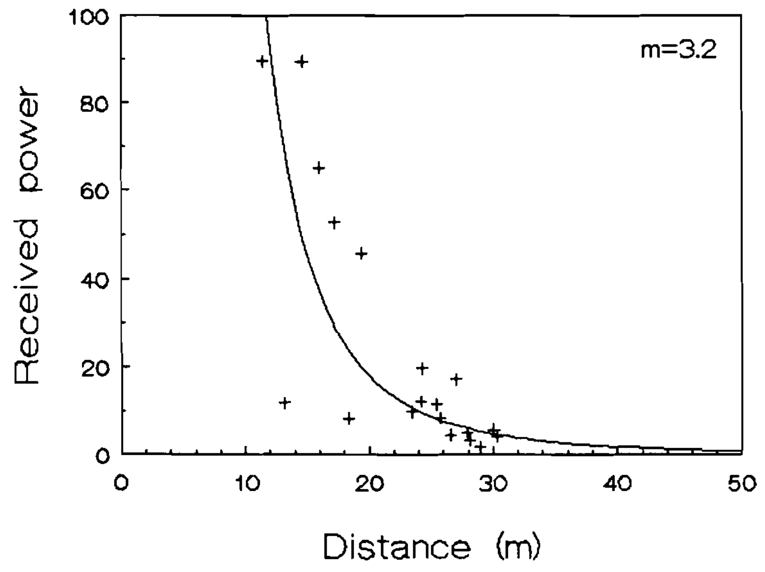


Fig. E20: Received power versus distance of co-polarization.

Appendix F. Probability function of power delay spread σ to characterize the effect of objects in a room.

Probability function of σ for empty room with dimensions 40*60*4 m^3 and R=0.2, 0.4 and 0.6.

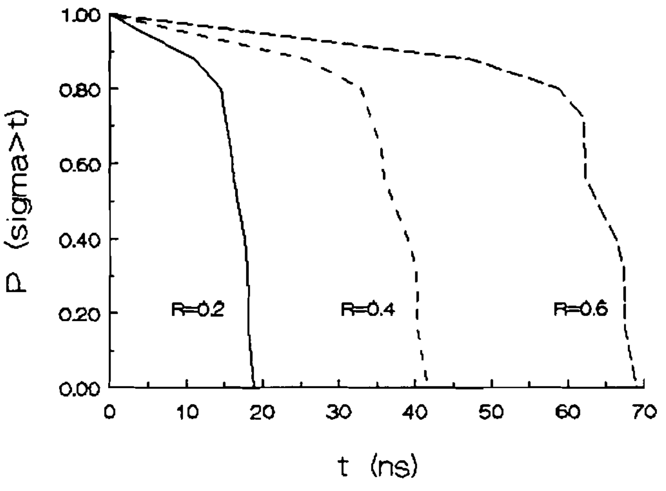


Fig. F1: Probability functions of σ for room 40*60*4, R=0.2, 0.4 and 0.6.

Probability function of σ for the same room with R=1 for simulating a factory.

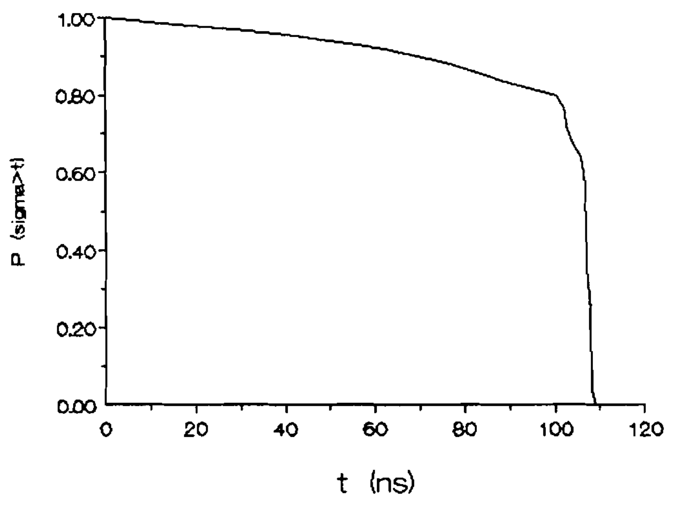


Fig. F2: Probability function of σ for room 40*60*4 and R=1.0.

Probability function of σ for room of 40*60*4 m^3 with objects. $R_{walls}=0.2$ and $R_{objects}=1.0$.

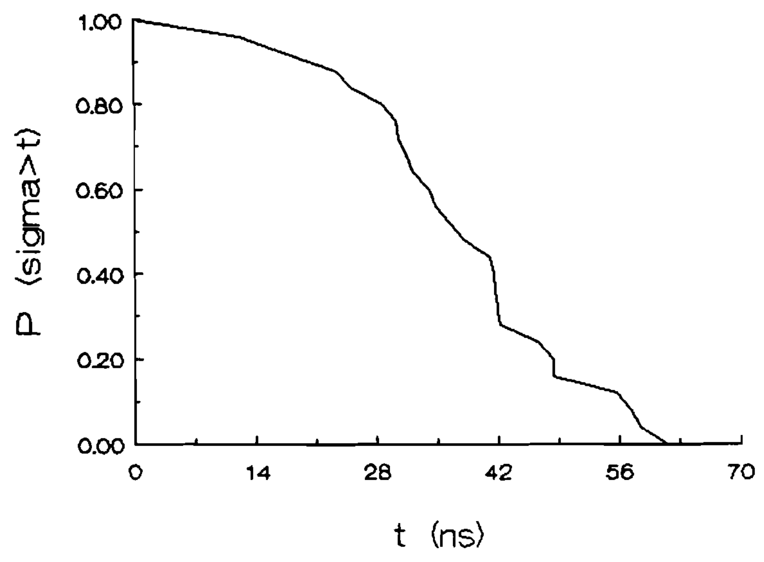


Fig. F3: Probability function of σ for room 40*60*4 m^3 with objects.

Probability function of σ for empty room of 10*10*3 m^3, $R_{walls}=0.2$.

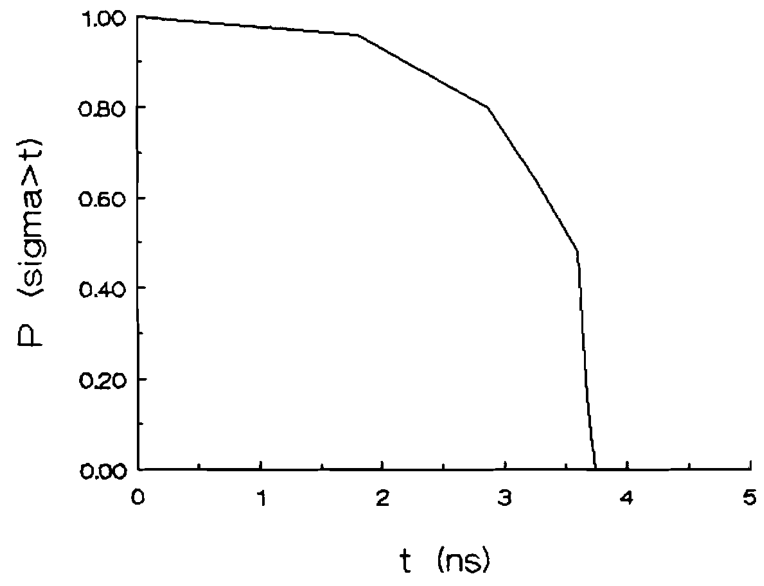


Fig. F4: Probability function of σ for empty room of 10*10*3 m^3.

Probability function of σ for room of 10*10*3 m^3 with objects. $R_{walls}=0.2$ and $R_{objects}=1.0$.

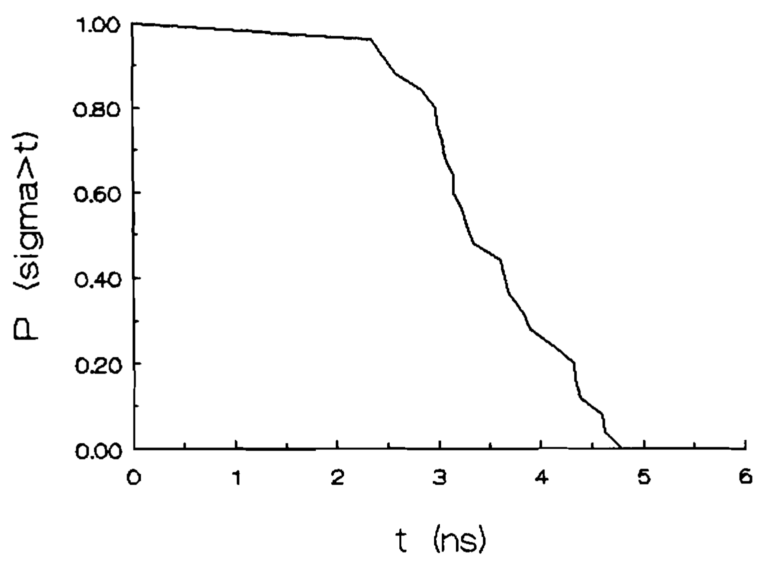


Fig. F5: Probability function of σ for room of 10*10*3 m^3 with objects.

Probability function of σ for empty room of 75*4*3 m^3, $R_{walls}=0.2$.

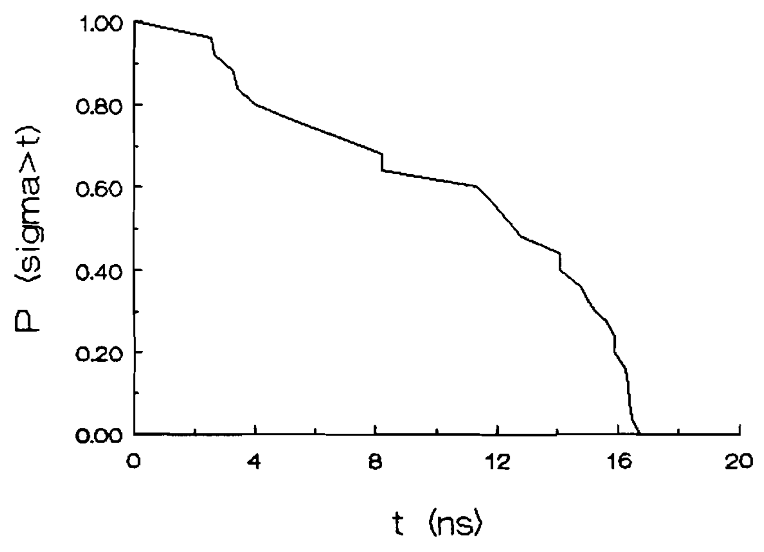


Fig. F5: Probability function of σ for empty room of 75*4*3 m^3.

Probability function of σ for room of 75*4*3 with 2 metal walls.

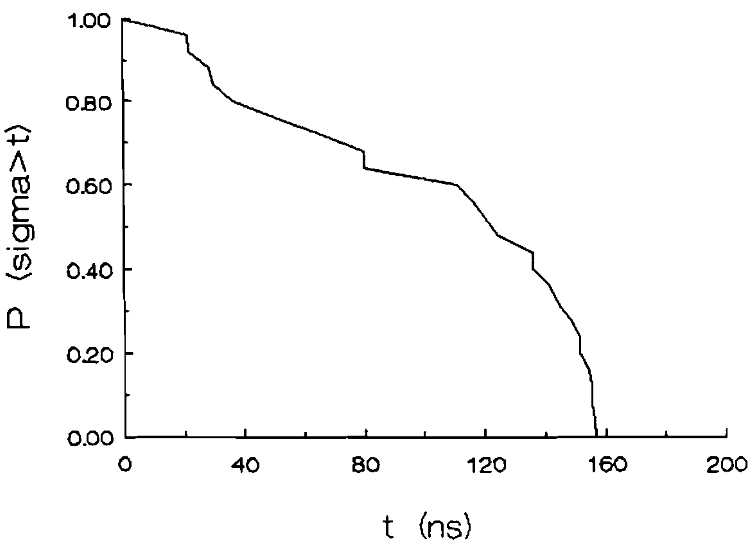


Fig. F5: Probability function of σ for room of 75*4*3 m^3 with objects.

Appendix G. software description.

G.1. Functional description programmes.

G.1.1. Introduction.

The programmes STRAAL3D and POLPOWER calculate for a given room configuration the power delay profile, average mean delay, average delay spread and power gain.

Since Geometrical Optics is used as a model for the indoor radio link, rays have to be defined which travel from transmitter to receiver via multiple reflections against walls and objects. Each ray represents an electromagnetic wave with some kind of polarization state. The polarization state can be described by two perpendicular field components and a phase difference. These components in general, have different reflection coefficients when reflecting against a surface. So the polarization state of the wave is changed. This means that for each ray the two components have to be considered separately. Furthermore the antenna gain has to be considered, because the rays leave the transmitter and approach the receiver at different angles. If the polarization is not considered, the combination of the programmes STRAAL3D and POWER calculate the power delay profile. The difference between the input and output of the polarization considering programmes and the non-polarization programmes is explained below.

G.1.2. STRAAL3D.

The programme STRAAL3D calculates for a given room configuration all rays which arrive at the receiver. Of each ray the polarization components, the phase difference, the two angles of arrival and the length of the ray are written in an output file. The room configuration is stored in an input file. Its format is explained in the next section. A block diagram of the programme STRAAL3D is given in Fig. G1.

Running STRAAL3D.

When the programme STRAAL3D is executed, it asks for the names of the input and output file. The input file is the file where the room configuration is stored in.

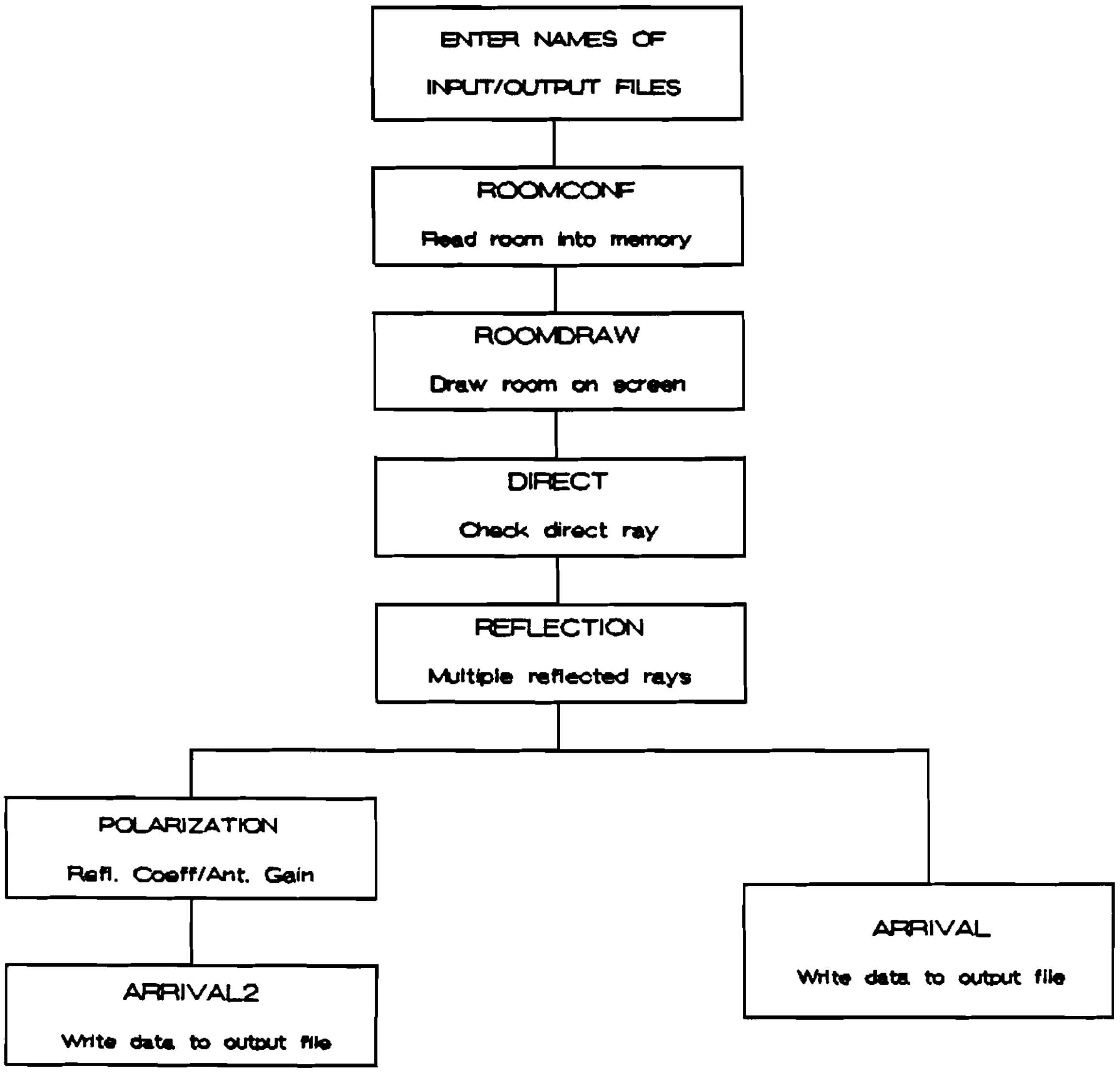


Fig. G1: Block diagram of STRAAL3D.

Input file.

The input file has the format as shown in Fig. G2. It is an ordinary text file with just numbers separated by blanks and consisting of several lines. The maximum supported room dimensions are unlimited. However a room with x coordinates over 600 meters or y coordinates over 200 meters or z coordinates over 100 meters will be distorted when drawn on the screen. The coordinates have to be entered in meters.

Another restriction is the fact that all walls or surfaces of objects have to be rectangular and stand perpendicular to each other. This is a consequence of the data representation of the room configuration.

line 1:	Tx Ty Tz Rx Ry Rz	Coordinates transmitter (T), receiver (R).
line 2:	N	Number of objects in room.
line 3:	x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4 z_0 z_B	Coordinates room.
line 4:	ϵ_1 0 ϵ_2 0 ϵ_3 0 ϵ_4 0 ϵ_0 0 ϵ_B 0	Refl. coeff. or permittivity of room surfaces.
line 5:	x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4 z_0 z_B	Coordinates first object.
line 6:	ϵ_1 0 ϵ_2 0 ϵ_3 0 ϵ_4 0 ϵ_0 0 ϵ_B 0	Refl. coeff. or permittivity first object.

Fig. G2: Example of input file.

The first line of the input file consists of six numbers which represent the x, y and z coordinates of the transmitter (T) and the receiver (R), respectively. The second line gives the number (N) of objects within the room. The walls of the room do not form an object within the room.

The third line consists of the coordinates of the walls of the room. First the coordinates of the ground plane are given and then the z coordinate of the ground plane and the z coordinate of the ceiling. The order of the x and y coordinates is shown in Fig. G3.

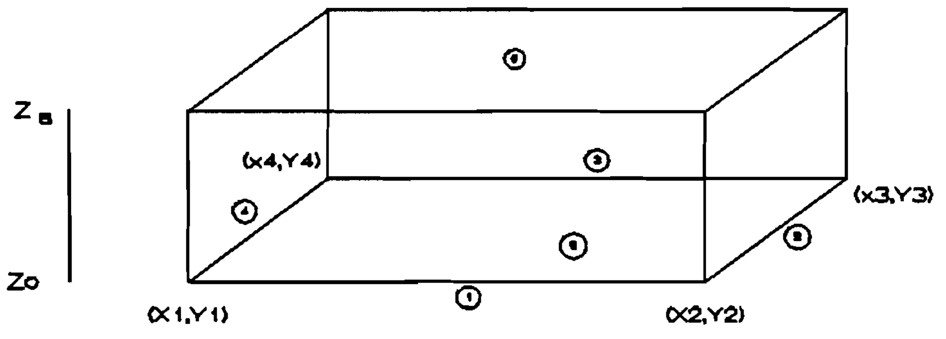


Fig. G3: Order of x and y coordinates.

The fourth line gives the reflection coefficient or the dielectric permittivity of the walls, separated by zeros. The first entry is the permittivity of wall no. 1. The number of the walls is given in Fig. G3. The ninth entry in this line is the permittivity of the floor and the eleventh one the permittivity of the ceiling. The next two lines are the same as lines 3 and 4, but now for object no. 1. The order is the same as for the walls of the room.

Retrieve room configuration.

In the procedures ROOMCONF and ROOMDRAW is the room loaded into memory, and drawn on the screen, respectively. In the procedure ROOMCONF the data of the room configuration is put into an array of records. This data storage is explained extensively in the technical description of the program. The procedure ROOMDRAW uses the procedure SCREENLINE to draw lines on the screen. This procedure performs a coordinate transformation to draw two dimensional lines from three dimensional coordinates.

Direct ray.

The procedure DIRECT checks if the direct ray from transmitter to receiver exists. This is done by calling the procedure CROSSPLANE. If the direct ray exists the procedure DIRECT returns a boolean to the main programme with the value true.
CROSSPLANE looks if the line from transmitter to receiver, represented by the coordinates of the transmitter and receiver, crosses a plane of any object within the room. If the room is empty, the direct ray automatically exists, since the room is rectangular.

Reflected rays.

The procedures REFLECTION calculate the rays which reflect 1, 2, 3 and 4 times against walls and objects. The procedure REFLECTION1 calls a procedure MIRROR, which returns the mirror point of the transmitter, mirrored in a particular plane. Then the procedure REFLECTION1 calls the procedure CROSSPLANE, which checks if the line from the receiver to the mirror point of the transmitter, crosses the plane used as a mirror. If so, the procedure CROSSPLANE returns the crossing point of the line and a boolean valued true.

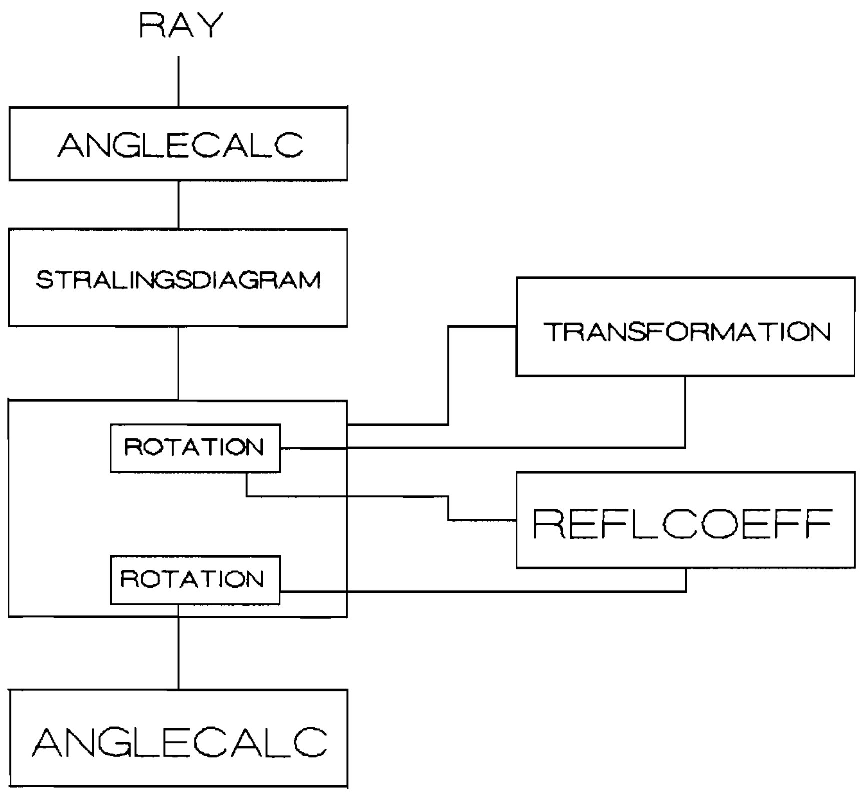


Fig. G6: Procedure POLARIZATION.

TRANSFORMATION calculates the angle of rotation to rewrite the θ and ϕ components of the electric field to parallel and perpendicular components with respect to the plane of incidence. The angle of rotation is γ. TRANSFORMATION also defines the angle of incidence in the plane of incidence.
After this the procedure PLANEREFLECTION takes over again and rotates the electric field components.

The reflection coefficients are applied in the procedure REFLCOEFF and the procedure PLANEREFLECTION rotates the components back to the original directions.

After the last reflection of each ray is considered, the angle of arrival at the receiver is calculated. Furthermore the length of each ray is calculated and the strength of each field component is altered. The procedure POLARIZATION controls this process and after all rays are considered, the main program calls the procedure ARRIVAL2 to write the data to the output file.

Output file.

In each line of the output file the data of one ray is listed. The order is, length of the ray in meters, amplitude of the θ component, amplitude of the ϕ component, the phase difference between the two field components, the θ angle of the arriving ray and the ϕ angle of the arriving ray, respectively. An example of the output file is shown in Fig. G7. The dots between the lines indicate that the files usually are longer. For an empty room there are around 130 lines

```
2.4270146E+00  5.9772999E-01  0.0000000E+00  0.0000000E+00  1.8230075E-01  3.2170791E+00
1.5187421E+01  2.3454007E-02  2.3052775E-02  0.0000000E+00  2.8953580E-02  6.2713284E+00
1.2828813E+01  2.7738646E-02  2.7866085E-02  0.0000000E+00  3.4345274E-02  3.1558082E+00
      ..             ..             ..             ..             ..             ..
      ..             ..             ..             ..             ..             ..
4.0247123E+00 -3.0080704E-02  0.0000000E+00  0.0000000E+00  9.3588902E-01  3.2177919E+00
3.6489024E+00 -4.3525639E-02  0.0000000E+00  0.0000000E+00 -8.5773157E-01  3.2170791E+00
9.9101090E+00 -2.5449683E-02  9.0142505E-03  0.0000000E+00  4.4413744E-02  4.4696516E+00
      ..             ..             ..             ..             ..             ..
      ..             ..             ..             ..             ..             ..
9.8711703E+00 -2.5537172E-02  9.04462843E-03  0.0000000E+00  4.4588351E-02  1.8144531E+00
1.7974407E+01  1.1668662E-02 -6.50191716E-04  0.0000000E+00  2.4478440E-02  5.7183330E+00
3.0387193E+01  5.4832854E-03 -9.41156692E-07  0.0000000E+00  1.4481959E-02  3.1451599E+00
```

Fig. G7: Example of output file of STRAAL3D with polarization consideration.

No polarization.

There is a possibility to neglect the polarization dependency at each reflection and only use an absolute reflection coefficient. If this option is chosen, the dielectric permittivity of the walls in the input file have to be changed to the absolute reflectivity of the wall. The rest of the input file stays the same. Furthermore the procedures POLARIZATION and ARRIVAL2 must not be called by the main program, but the procedure ARRIVAL must be called in stead. The output file changes too. Each line states now the time of arrival of one ray, followed by the strength of the field and the two angles of arrival at the receiver. An Example of such a file is given in Fig. G8.

```
1.00209E+02  9.3986566439E-03 -7.9918456183E-02  5.8344619632E+00
1.98432E+02  6.7458610309E-04 -4.0327083161E-02  5.1835552447E+00
     ..              ..                ..                  ..
7.12303E+02  7.9349985522E-07 -1.1231416637E-02  6.0325313361E+00
3.57261E+02  3.3508683070E-06  5.0404735227E-02  3.6595800198E+00
     ..              ..                ..                  ..
1.09695E+02  3.1255764423E-05  4.2604092073E-01  5.8344619631E+00
1.32093E+02  2.0882067581E-05 -2.7078633478E-01  7.8539816337E-01
```

Fig. G8: Output file for non-polarization consideration.

G.1.3. POLPOWER.

The procedure POLPOWER represents the receiving stage of the process of calculating the power delay profile with consideration of the polarization dependant reflection coefficients. The program starts with asking for the name of the input file. This is the file where STRAAL3D has written its data in. Then the name of the output file and the pulse width is asked.
Now the procedure STRALINGRECEIVER is called. It reads all the data in the file into memory and into the same records as they were in the program STRAAL3D.

The following stage is the procedure POLPOWER (Fig. G7). It is called from the main program and it looks for all the rays, which overlap at the receiver. It scans the time axis and at every instant, during which one or more rays are present at the receiver it calls RECEIVEDPOWERLIN or RECEIVEDPOWERELL, for respectively a linear polarized receiving antenna or a circular polarized receiving antenna.

The procedure RECEIVEDPOWERLIN first calls the procedure STRALINGSDIAGRAM, where the sensitivity of the receiving antenna is calculated. Back in RECEIVEDPOWERLIN, the electric field components are shaped in a cosine form and multiplied with the receiver sensitivity. In case there are multiple arriving rays, each component of the rays are added while considering its phase and then the received power is calculated. This received power is valid for one moment in time. To calculate the power received later, control is given back to POLPOWER, which checks the next moment for arriving rays.

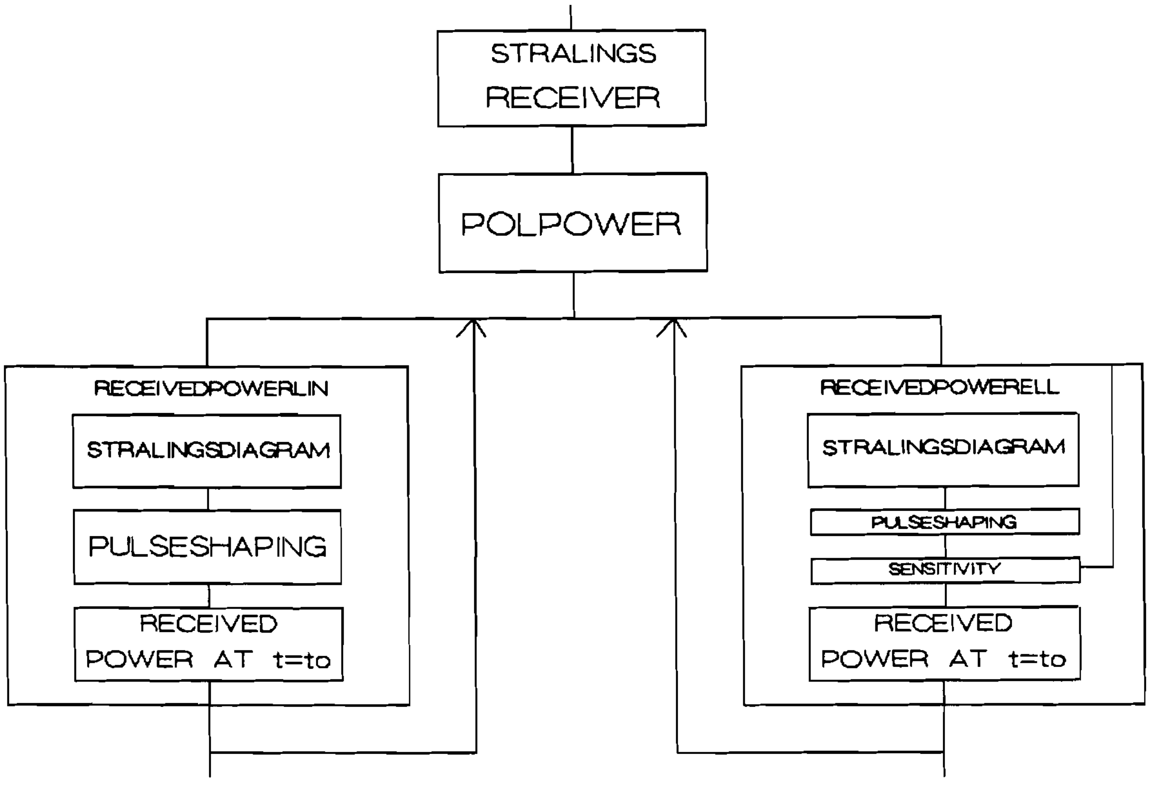


Fig. G9: Program POLPOWER.

In case a circular polarized antenna is used, the process is somewhat more complicated. Again POLPOWER checks for arriving rays, but now RECEIVEDPOWERELL is called. This procedure calls STRALINGSDIAGRAM, where the sensitivity of the receiving antenna is defined and then calls the procedure SENSITIVITY twice. The first time for rewriting the receiving antenna θ and ϕ components in left and right hand circular waves and then for the same operation for the electric field components arriving at the receiver.

RECEIVEDPOWERELL calls STRALINGSDIAGRAM and SENSITIVITY for each ray that arrives at the same moment. For this moment RECEIVEDPOWERELL calculates the received power from the left and right hand circular components of the arriving rays. After this POLPOWER takes over and checks for the next arriving rays.

Statistics.

The programme POLPOWER also calculates the parameters average mean delay, average delay spread and the power gain. This is done by simply calling the procedure STATISTICS from the main programme. The results are written to the screen of the computer.

Output file.

The output file of POLPOWER is a file containing the power delay profile of the indoor radio link. The power delay profile is therefore positioned direct after t=0. This means that the arrival time of the direct ray is zero.
Each line of the output file consists of an integer stating the time (x 0.5 ns) and the power received at that instant. An example is given in Fig. G8. Only the first 5 ns are given of the profile. The whole profile consists of 500ns.

```
0.000E+00    -4.9015672971E+00
5.000E-01    -2.9904913392E+00
1.000E+00    -1.7927844644E+00
1.500E+00    -9.9160012372E-01
2.000E+00    -4.6068661495E-01
2.500E+00    -1.3997703022E-01
3.000E+00     0.0000000000E+00
3.500E+00    -2.9249934271E-02
4.000E+00    -2.3006768684E-01
4.500E+00    -6.1944257638E-01
5.000E+00    -1.2353880296E+00
```

Fig. G10: Example of an output file of POLPOWER.

A.1.4. POWER.

In case the polarization is not considered, the programme POWER calculates the power delay profile and the parameters. The parameters are written to the monitor of the computer.
The input file of the programme is the output file of STRAAL3D in case no polarization is considered. The output file is the same as for POLPOWER.

When executed, the programme asks for the name of the input file, the output file and the pulse width. After this the received power is calculated for every 0.5ns after the pulse is sent. For calculation of the parameters average mean delay, delay spread and power gain the procedure STATISTICS is called. This is the same procedure as in the programme POLPOWER. Also the same as in the programme POLPOWER is the procedure STOREPDP. It is used to write de power delay profile to the output file.

G.1.5. CUMPOWER.

Last of the programmes is CUMPOWER. It is used to calculate a cumulative power delay profile of several profiles. The names of the different files where the individual profiles are stored in, must be written in the file LIST.PDP. This list is read by the programme CUMPOWER.

After the programme is executed, it starts with asking where the cumulative power delay profile must be written to. It also asks for the pulse width used to calculate the individual profiles. This information is necessary while the average mean delay and delay spread of the cumulative profile is calculated also (STATISTICS). The procedure STOREPDP is used to write the cumulative profile to disk.

NOTE: Special attention must be paid to the lines in the main programmes where the directories of the input and output files are specified. These statements usually follow the the statements

```
WRITELN ('GEEF DE INVOERFILE');
READLN (INVOER);
```

The statement specifying the directory of the input file is

```
INVOER:=CONCAT ('C:\TP50\MARC\',INVOER);
```

In this statement the operation CONCAT puts the strings C:\TP50\MARC\ and INVOER together. The string C:\TP50\MARC specifies the directory of the input file. In the CONCAT statement the third back slash \ must not be forgotten.

G.2. Technical description programmes.

G.2.1. STRAAL3D.

G.2.1.1. Data structure.

Room configuration.

The key of the data structure is the storage of the different surfaces of the room and objects. first of all, all objects and the room are stored in a different record. These records are the items of an array. This array stores the whole room configuration. The data type of the array is CONFIGURATION, an array from 0 to 10 of a record called object. Thus the maximum number of objects within the room is 10.
The record OBJECT stores all parameters of one object. It consists of 4 arrays of 18 items and the type definition is shown below.

```
OBJECT = RECORD
   X: ARRAY[1..18] OF REAL;
   U: ARRAY[1..18] OF REAL;
   V: ARRAY[1..18] OF REAL;
   R: ARRAY[1..18] OF REAL;
  END;
CONFIGURATION = ARRAY[0..10] OF OBJECT;
```

The array X stores the coordinates of one corner point of each of the six surfaces of an object. The side surfaces of an object are numbered 1 to 4, the ground plane has number 5 and the top number 6. So X[1], X[2], X[3] are the x, y, z coordinates of the corner point of surface number 1.
Next are the arrays U and V. These are vectors which constitute the surfaces of an object. For example, the third surface of an object is described by

$$\underline{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} X[7] \\ X[8] \\ X[9] \end{pmatrix} + \lambda \begin{pmatrix} U[7] \\ U[8] \\ U[9] \end{pmatrix} + \mu \begin{pmatrix} V[7] \\ V[8] \\ V[9] \end{pmatrix},$$

with λ and μ between 0 and 1.

This leaves the array R. In this array are the reflection coefficients or the dielectric permittivity stored. Thus R[9] is the data of surface number 4 and indeed R[10] and R[11] are not used.
So if CONF is the variable of type CONFIGURATION in which the room configuration is stored, CONF[0] represents all data of the walls of the room and CONF[1] all the data of the surfaces of object 1 in the room. Furthermore the corner point of surface number 6 (=top surface) of object number 3, is therefore assigned by CONF[3].X[16], CONF[3].X[17] and CONF[3].X[18].

This way of storage is useful, while all surfaces of the room or object are addressable by just increasing a dummy variable with 1. Each surface is handled the same, without knowledge of the nature of it, i.e. a side surface or top or bottom plane.

Storage of reflection points.

Of each ray the coordinates of the reflection points have to be remembered. Since each ray starts at the transmitter and ends at the receiver, these coordinates are stored separately in an array T of six real variables.

The reflection points of the single reflected rays are stored in an array RAY1[I,J]. The I is the number of the ray and J denotes the coordinates numbers. So the x, y, z of the third single reflected ray is denoted by respectively, RAY1[3,1], RAY1[3,2] and RAY1[3,3]. The data about the reflectivity of the reflecting surface is stored in RAY1[3,4].

The storage of the reflection points of the multiple reflected rays is done in a similar manner. If XC1, YC1 and ZC1 represent the x, y, and z coordinate of the first reflection point, respectively, and the reflection information is denoted by R1, then the storage of the points is

RAY2[I,XC1]	RAY3[I,XC1]	RAY4[I,XC1]
RAY2[I,YC1]	RAY3[I,YC1]	RAY4[I,YC1]
RAY2[I,ZC1]	RAY3[I,ZC1]	RAY4[I,ZC1]
RAY2[I,XC2]	RAY3[I,XC2]	RAY4[I,XC2]
RAY2[I,YC2]	RAY3[I,YC2]	RAY4[I,YC2]
RAY2[I,ZC2]	RAY3[I,ZC2]	RAY4[I,ZC2]
RAY2[I,R1]	RAY3[I,XC3]	RAY4[I,XC3]
RAY2[I,R2]	RAY3[I,YC3]	RAY4[I,YC3]
	RAY3[I,ZC3]	RAY4[I,ZC3]
	RAY3[I,R1]	RAY4[I,XC4]
	RAY3[I,R2]	RAY4[I,YC4]
	RAY3[I,R3]	RAY4[I,ZC4]
		RAY4[I,R1]
		RAY4[I,R2]
		RAY4[I,R3]
		RAY4[I,R4]

and I is number of the ray.

Storage of field at the receiver.

Before the data about the electromagnetic fields at the receiver are written to disk, they are stored in records. For each ray there is a record DETECTRECORD to store its data. These records are held together by an array DETECTARR. The type definitions of the record and array are

```
DETECTRECORD = RECORD
  LENGTH,
  ATHETA,APHI,
  PHASE,
  THETA,PHI   : REAL;
END;

DETECTARR = ARRAY[1..200] OF DETECTRECORD;
```

The fields of the record represent respectively, the LENGTH of the ray, the amplitude ATHETA of the θ component of the electromagnetic field, the amplitude APHI, the phase difference between ATHETA and APHI (positive if APHI leads), and the THETA and PHI angle of arrival at the receiver.
The array DETECTARR is only 200 deep, but this has proved to be enough.

G.2.1.2. Variables in the main program.

All the variables used in the main program will now be explained. For this purpose the main programme is listed below. The line numbers are just for the help of describing the programme.

```
1       BEGIN
2               WRITELN ('GEEF DE INPUTFILE');
3               READLN(INVOER);
4               INVOER:=CONCAT('C:\TP50\MARC\ROOMCONF\',INVOER);
5               WRITELN ('GEEF DE OUTPUTFILE');
6               READLN (UITVOER);
7               UITVOER:=CONCAT('C:\TP50\MARC\UIT_STR3\',UITVOER);
8               ROOMCONF (CONF,T,AANTAL,INVOER);
9               DETECTGRAPH (GD,GM);
10              INITGRAPH (GD,GM,'C:\TP50');
11              ROOMDRAW (CONF,T,AANTAL);
12              DIRECT (CONF,T,AANTAL,EXIST0);
13              REFLECTION1 (CONF,T,AANTAL,RAY1,AANTAL1,PL1);
14              REFLECTION2 (CONF,T,AANTAL,RAY2,AANTAL2,PL2);
15              REFLECTION3 (CONF,T,AANTAL,RAY3,AANTAL3,PL3);
16              REFLECTION4 (CONF,T,AANTAL,RAY4,AANTAL4,PL4);
17
18              ARRIVAL (UITVOER,T,RAY1,RAY2,RAY3,RAY4,EXIST0,
19                      AANTAL1,AANTAL2,AANTAL3,AANTAL4);
20
21              POLARIZATION (EXIST0,T,RAY1,RAY2,RAY3,RAY4,
22                      AANTAL1,AANTAL2,AANTAL3,AANTAL4,
23                      PL1,PL2,PL3,PL4,DETECTREC,TOTAL);
24
25              ARRIVAL2 (TOTAL,DETECTREC,UITVOER);
26              SETTEXTSTYLE (0,0,2);
27              OUTTEXTXY (0,0,'EINDE');
28              WHILE NOT KEYPRESSED DO;
29              CLOSEGRAPH;
30      END.
```

Line 1 to 7 speak for themselves. INVOER and UITVOER are strings, to which a directory path is assigned by the concat statement.

In line 8 a call is made to procedure ROOMCONF, to retrieve the room configuration from disk. The variable CONF is the array, in which the configuration is stored. T represents the coordinates of the transmitter and receiver. Furthermore denotes AANTAL the number of objects within the room and is INVOER the file where the room configuration is stored in.

Line 9 and 10 are for initializing the graphics system of the computer.

The next new variable is EXIST0, it is a boolean, which is true if the direct ray exists and false if not. The amount of rays which reflect 1, 2, 3 and 4 times are remembered by the integers AANTAL1, AANTAL2, AANTAL3 and AANTAL4 respectively.

The variables PL1 to PL4 need more explanation. In the situation where a ray reflects against a ground or top plane of an object, the θ and ϕ components need not be rotated. Therefore it is remembered if a ray reflects against such a plane or not. PL1[i] is an array of characters, where the i denotes the ray number of the single reflected rays. If PL1[i]='Z', the reflection takes place against a side plane of an object and PL1[i]='U' denotes a reflection against a top or bottom plane.
PL2[i,j], PL3[i,j] and PL4[i,j] are also arrays of characters. The i stands for the number of the ray and the j for the reflection number, for instance 2 for the second reflection point.
The last variable is TOTAL. It is an integer denoting the total amount of rays.

Line 26 till 29 are for drawing the word 'EINDE' on the computer screen as an indication that the calculations are ready, and for closing the graphics system of the computer.

G.2.1.3. Description of procedures in STRAAL3D.

```
PROCEDURE ROOMCONF (   VAR C          : CONFIGURATION;
                       VAR T          : ARRAY6;
                       VAR AANTAL     : INTEGER;
                       INVOER         : STRING);
```

ROOMCONF retrieves the data of the room configuration from disk. The name of the input file together with the directory path is contained in INVOER.
The variable C will contain the whole configuration. The array T contains the coordinates of the transmitter and receiver, respectively. AANTAL denotes the number of objects in the room.

In the procedure the coordinates of the objects are read and converted into vectors by subtracting then from each other. The vectors are written in the variable C.

```
PROCEDURE SCREENLINE (X1,Y1,Z1,X2,Y2,Z2: REAL);
```

SCREENLINE is the procedure that performs a coordinate transformation. It transforms the three dimensional coordinates (X1,Y1,Z1) and (X2,Y2,Z2) into two dimensional screen coordinates and draws a between them on the screen.

```
PROCEDURE ROOMDRAW (  C             : CONFIGURATION;
                      T             : ARRAY6;
                      AANTAL        : INTEGER);
```

ROOMDRAW draws the room configuration stored in C on the screen. It arranges the three dimensional coordinates of the room and calls the procedure SCREENLINE to draw the lines on the screen.
T is the array where the coordinates of the transmitter and receiver are listed in. It is an array of 6 real variables. AANTAL is the number of objects in the room.

```
PROCEDURE MIRROR (    QX,QY,QZ,
                      AX,AY,AZ,
                      VX,VY,VZ,
                      UX,UY,UZ             : REAL;
                      VAR MX,MY,MZ         : REAL);
```

MIRROR performs the calculation of the mirror point of point (QX,QY,QZ) mirrored in the plane described by the direction vectors (VX,VY,VZ) and (UX,UY,UZ) and the position vector (AX,AY,AZ). The mirror point is (MX,MY,MZ).

```
PROCEDURE CROSSPLANE ( QX,QY,QZ,
                       MX,MY,MZ,
                       AX,AY,AZ,
                       UX,UY,UZ,
                       VX,VY,VZ             : REAL;
                       VAR PX,PY,PZ         : REAL;
                       VAR CROSSING         : BOOLEAN);
```

CROSSPLANE checks if the line from (QX,QY,QZ) to (MX,MY,MZ) crosses the plane described by the direction vectors (UX,UY,UZ) and (VX,VY,VZ) and the position vector (AX,AY,AZ). If so the boolean CROSSING will be true and the crossing point of the line with the plane is (PX,PY,PZ).

```
PROCEDURE DIRECT (         C            : CONFIGURATION;
                           T            : ARRAY6;
                           AANTAL       : INTEGER;
                           VAR EXIST    : BOOLEAN);
```

DIRECT looks if the line from transmitter to receiver (stored in T) crosses an object of the room configuration C. AANTAL is the number of objects in the room. If the line does not cross any object, the boolean EXIST will be true.
The calculation performed, calls for each surface of every object in the room and of the room itself the procedure CROSSPLANE.

```
PROCEDURE CROSSLINES1 (  XC,YC,ZC          : REAL;
                         T                 : ARRAY6;
                         C                 : CONFIGURATION;
                         A                 : INTEGER;
                         VAR CROSSING      : BOOLEAN);
```

CROSSLINES1 checks if the ray from transmitter (T[1],T[2],T[3]) via reflection point (XC,YC,ZC) to the receiver (T[4],T[5],T[6]) crosses an object in the room. If not the boolean CROSSING will be false, so the ray exists.
C is the room configuration and A is the number of objects in the room.
The calculation is the same as in DIRECT. For each surface in the room and the two lines, the procedure CROSSPLANE is called.

```
PROCEDURE REFLECTION1 (  C            : CONFIGURATION;
                         T            : ARRAY6;
                         A            : INTEGER;
                         VAR RAY      : ARRAY1_5;
                         VAR Z        : INTEGER;
                         VAR PLANE    : PLANE1);
```

REFLECTION1 calculates all single reflected rays possible in the room defined by C. T represents the coordinates of transmitter and receiver, and A is the number of objects in the room. RAY is the array in which the reflection points and the reflection coefficient of the rays is stored. PLANE is an array in which the nature of the reflecting wall is stored, 'Z' for side planes of objects an 'U' for top or bottom planes. Z denotes the number of calculated single reflected rays.
The calculation is explained in Appendix A.1.

```
PROCEDURE CROSSLINES2 ( XC1,YC1,ZC1,
                        XC2,YC2,ZC2        : REAL;
                        T                  : ARRAY6;
                        A                  : INTEGER;
                        C                  : CONFIGURATION;
                        VAR CROSSING       : BOOLEAN);
```

CROSSLINES2 performs the same action as CROSSLINES1, but now for the two times reflected rays three lines. So for each line all surfaces are checked by CROSSPLANE.
The coordinates of transmitter and receiver are stored in T, the points (XC1,YC1,ZC1) and (XC2,YC2,ZC2) are the reflection points of the ray and C is the room configuration with A objects in the room. If the lines do not cross any plane (CROSSPLANE), the boolean CROSSING will be false and the two times reflected ray exists.

```
PROCEDURE REFLECTION2 ( C          : CONFIGURATION;
                        T          : ARRAY6;
                        A          : INTEGER;
                        VAR RAY    : ARRAY2_10;
                        VAR Z      : INTEGER;
                        VAR PLANE  : PLANE2);
```

REFLECTION2 calculates all rays which arrive at the receiver after two reflections. The room configuration is stored in C, the positions of the transmitter and receiver in T and A denotes the number of objects in the room.
RAY is the array where the reflection points and reflection coefficient are stored in, Z is in this case the number of calculated rays. Furthermore PLANE is the array which remembers the nature of the reflecting surface, 'Z' for side planes and 'U' for top and bottom planes.
The rays are found in a similar way as in REFLECTION1. The transmitter is mirrored two times in different planes. If the constructed ray is found, CROSSLINES2 checks if the ray does not cross any object.

```
PROCEDURE CROSSLINES3 ( XC1,YC1,ZC1,
                        XC2,YC2,ZC2,
                        XC3,YC3,ZC3        : REAL;
                        T                  : ARRAY6;
                        A                  : INTEGER;
                        C                  : CONFIGURATION;
                        VAR CROSSING       : BOOLEAN);
```

```
PROCEDURE REFLECTION3 ( C          : CONFIGURATION;
                        T          : ARRAY6;
                        A          : INTEGER;
                        VAR RAY    : ARRAY3_15;
                        VAR Z      : INTEGER;
                        VAR PLANE  : PLANE3);
```

The combination of CROSSLINES3 and REFLECTION3 calculate all three times reflected rays. REFLECTION3 calculates a ray and CROSSLINES3 checks if the line does not cross any object in the room.
In both procedures does C represent the room configuration, T the position of transmitter and receiver and A the number of objects in the room. In CROSSLINES3 are (XC1,YC1,ZC1), (XC2,YC2,ZC2) and (XC3,YC3,ZC3) the reflection points of the ray. If the ray does not cross another object in the room, the boolean CROSSING will be false and the ray exists.
Furthermore in REFLECTION3 is RAY the array where the reflection points are stored in, Z the total number of rays and PLANE the array, where the nature of the reflecting surfaces is remembered.

```
PROCEDURE CROSSLINES4 ( XC1,YC1,ZC1,XC2,YC2,ZC2,
                        XC3,YC3,ZC3,XC4,YC4,ZC4     : REAL;
                        T                           : ARRAY6;
                        A                           : INTEGER;
                        C                           : CONFIGURATION;
                        VAR CROSSING                : BOOLEAN);
```

```
PROCEDURE REFLECTION4 ( C           : CONFIGURATION;
                        T           : ARRAY6;
                        A           : INTEGER;
                        VAR RAY     : ARRAY4_20;
                        VAR Z       : INTEGER;
                        VAR PLANE   : PLANE4);
```

CROSSLINES4 and REFLECTION4 are the same kind of procedures as CROSSLINES3 and REFLECTION4, respectively. Now the four times reflected rays are calculated.
All the variables are exactly the same as the ones for the three times reflected rays.

```
PROCEDURE ARRIVAL (     UIT          : STRING;
                        T            : ARRAY6;
                        RAYA         : ARRAY1_5;
                        RAYB         : ARRAY2_10;
                        RAYC         : ARRAY3_15;
                        RAYD         : ARRAY4_20;
                        EXIST        : BOOLEAN;
                        A1,A2,A3,A4  : INTEGER);
```

ARRIVAL performs the writing of the strength of the rays and the arrival moments to disk. UIT contains the name and the directory path of the output file. T is the array with the coordinates of transmitter and receiver and in the arrays RAYA, RAYB, RAYC and RAYD the reflection points of the rays are stored in. These are necessary for calculating the length L of the rays. Furthermore does EXIST tell the procedure if the direct ray exists. A1, A2, A3 and A4 denote the number of 1, 2, 3 and 4 times reflected rays.

The calculation performed is multiplying 1/L with the reflection coefficients at every reflection of each ray. Furthermore ARRIVAL calls STRALINGSDIAGRAM for every ray which leaves the transmitter. ARRIVAL also calls the procedure ANGLECALC for calculation of the angles at which the rays leave the transmitter and the angles at which the rays arrive at the receiver. ARRIVAL also performs the writing of the data to the output file on disk in the case where no polarization effects are considered.

```
PROCEDURE ANGLECALC (   XT,YT,ZT,
                        XC1,YC1,ZC1             : REAL;
                        VAR THETA,PHI,PHIREC    : REAL);
```

ANGLECALC calculates the angles THETA and PHI between the points (XT,YT,ZT) and (XC1,YC1,ZC1) as defined earlier. In the case that the angles are to calculated at the receiver, the angle PHIREC is the desired angle. The output THETA has to be multiplied with -1.

```
PROCEDURE REFLCOEFF (   PERP,PAR,THETA,EPS  : REAL;
                        VAR PERPUIT,PARUIT  : REAL);
```

REFLCOEFF calculates the amplitude of the reflected wave components PERPUIT and PARUIT in case of an incident wave at the angle THETA. EPS is the dielectric permittivity of the reflecting surface. The amplitude of the perpendicular and parallel components of the incident wave are PERP and PAR, respectively.

```
PROCEDURE TRANSFORMATION (   X1,Y1,Z1,
                             X2,Y2,Z2,
                             X3,Y3,Z3               : REAL;
                             VAR GAMMA,THETAINV     : REAL);
```

TRANFORMATION calculates the angle GAMMA over which the incident wave components have to be rotated to become the parallel and perpendicular components at the reflection surface. Furthermore the angle of incidence THETAINV in the plane of incidence is calculated.
The input variables are the points (X1,Y1,Z1), (X2,Y2,Z2) and (X3,Y3,Z3). The second point is the reflection point and the first and third point are points on the incoming and reflected ray, respectively. Usually these points are other reflection points or the positions of transmitter and receiver.

```
PROCEDURE STRALINGSDIAGRAMHUYGENS (       THETA,PHI                : REAL;
                                          VAR ETHETA,EPHI,FASE     : REAL);


PROCEDURE STRALINGSDIAGRAMHUYGENSCIRC (   THETA,PHI                : REAL;
                                          VAR ETHETA,EPHI,FASE     : REAL);
```

```
PROCEDURE STRALINGSDIAGRAMCORRHL ( THETA,PHI                 : REAL;
                                   VAR ETHETA,EPHI,FASE      : REAL);


PROCEDURE STRALINGSDIAGRAMCORRHC ( THETA,PHI                 : REAL;
                                   VAR ETHETA,EPHI,FASE      : REAL);
```

The procedures STRALINGSDIAGRAM calculate for the input angles THETA and PHI as defined earlier, the output electric field components ETHETA and EPHI. PHASE is the difference in phase between these components. It is positive if EPHI leads.
STRALINGSDIAGRAMHUYGENS and STRALINGSDIAGRAMHUYGENSCIRC are for a linear polarized and a circular polarized wave Huygens radiator. The procedures STRALINGDIAGRAMCORRHL and STRALINGSDIAGRAMCORRHC are for a linear and circular polarized corrugated horn.

```
PROCEDURE PLANEREFLECTION (   SIDEPLANE                : STRING;
                              X1,Y1,Z1,
                              X2,Y2,Z2,
                              X3,Y3,Z3,
                              ETHETA,EPHI,EPSILON      : REAL;
                              VAR ETHETAUIT,EPHIUIT    : REAL);
```

PLANEREFLECTION is called whenever reflection against a surface is considered. It calculates the electric field components ETHETAUIT and EPHIUIT from the incident components ETHETA and EPHI. EPSILON represents the dielectric permittivity of a reflecting surface.
Furthermore SIDEPLANE is the string which tells the procedure if the reflecting surface is a sideplane or a top or bottom plane.
The procedure PLANEREFLECTION controls the whole procedure at a reflection. If the reflection takes place against a sideplane, ETHETA and EPHI are rotated over an angle calculated in TRANSFORMATION, then the reflection coefficient are applied in a call to REFLCOEFF and after this the components are rotated back to the definitions of the directions of θ and ϕ components.

```
PROCEDURE POLARIZATION ( EXIST                : BOOLEAN;
                         T                    : ARRAY6;
                         RAY1                 : ARRAY1_5;
                         RAY2                 : ARRAY2_10;
                         RAY3                 : ARRAY3_15;
                         RAY4                 : ARRAY4_20;
                         AANTAL1,AANTAL2,
                         AANTAL3,AANTAL4      : INTEGER;
                         PL1                  : PLANE1;
                         PL2                  : PLANE2;
                         PL3                  : PLANE3;
                         PL4                  : PLANE4;
                         VAR DETECTION        : DETECTARR;
                         VAR TOTAL            : INTEGER);
```

POLARIZATION is called from the main program to calculate all data which are to be put in the outputfile of the whole programme. The input EXIST is true if the direct ray exists, T contains the position of the transmitter and receiver. RAY1, RAY2, RAY3 and RAY4 contain the reflection points of the rays and the dielectric permittivities of the reflecting surfaces. The number of rays are denoted by AANTAL1, AANTAL2, AANTAL3 and AANTAL4, respectively. The nature of the reflecting surface is stored in the arrays PL1, PL2, PL3 and PL4.
The output DETECTION is the array of records where the length of the rays, the θ and ϕ components, the phase difference between these components and the arriving angles at the receiver are listed in.
TOTAL is the number of records in DETECTION.
POLARIZATION calculates all the lengths of the rays, calls for every reflection the procedure PLANEREFLECTION and stores all data of the rays in the array DETECTION.

```
PROCEDURE ARRIVAL2 (     TOTAL          : INTEGER;
                         DETECTION      : DETECTARR;
                         UITVOER        : STRING);
```

ARRIVAL2 writes the data contained in DETECTION to disk. The name of the output file and directory path are contained by UITVOER. TOTAL is the number of records in DETECTION.
This procedure is called when the polarization components are considered separately at every reflection.

G.2.2. POLPOWER.

G.2.2.1. Data structure.

The data types used in POLPOWER are very much the same as in STRAAL3D. For instance the type DETECTARR is the same. New types are

```
        ARRAY30 = ARRAY[1..30] OF INTEGER;
        ARRAY999 = ARRAY[0..999] OF REAL;
```

These are used for storing numbers of rays arriving at the same time and the power delay profile, respectively. This will become clear later.

G.2.2.2. Variables in the main programme.

For describing the variables in the main program of POLPOWER, it is listed below.

```
1       BEGIN
2               WRITELN ('GEEF DE INVOERFILE');
3               READLN (INVOER);
4               INVOER:=CONCAT('C:\TP50\MARC\ROOMCONF\',INVOER);
5               WRITELN ('GEEF DE UITVOERFILE');
6               READLN (UITVOER);
7               UITVOER:=CONCAT ('C:\TP50\MARC\POL_PDP\',UITVOER);
8               STRALINGRECEIVER (INVOER,TOTAL,DETECTION);
9               WRITELN ('GEEF DE PULSLENGTE');
10              READLN (TAUP);
11              POLPOWER (DETECTION,TOTAL,TAUP,PDP);
12              STATISTICS (PDP,TAUP,AMD,ADS,GAIN,PDPN);
13              WRITELN ('AVERAGE MEAN DELAY   = ',AMD,' ns');
14              WRITELN ('AVERAGE DELAY SPREAD = ',SQRT(ADS),' ns');
15              WRITELN ('POWER GAIN           = ',GAIN);
16              STOREPDP (UITVOER,PDPN);
17      END.
```

Lines 2 to 7 are for telling the programme where the input file is stored and where the output file must be written to. INVOER and UITVOER will hold this information. Lines 9 and 10 are for obtaining the pulse width (TAUP).
The procedure STRALINGRECEIVER reads the data from the input file on disk. The data is stored in DETECTION, a variable of the type DETECTARR. TOTAL denotes the number of records DETECTION holds.

The calculation of the power delay profile is controlled by the procedure POLPOWER. The power delay profile will be stored in PDP, an array of 1000 real variables representing the received power every 0.5ns.

The procedure STATISTICS calculates the parameters average mean delay (AMD), average delay spread (ADS) and the power gain (GAIN). These are all real variables.
The last output of the procedure is the variable PDPN of the type ARRAY999. It holds the normalized power delay profile, which is the calculated power delay profile divided by its maximum value and shifted to t=0. Furthermore the logarithm is taken.
In lines 13 to 15 the values of the parameters are written to the monitor.
Finally STOREPDP writes the normalized power delay profile PDPN to disk in a file represented by UITVOER.

G.2.2.3. Description of procedures.

```
PROCEDURE STRALINGRECEIVER (  INVOER          : STRING;
                              VAR TOTAL       : INTEGER;
                              VAR DETECTION   : DETECTARR);
```

STRALINGRECEIVER retrieves the data written to disk by STRAAL3D into the array of records DETECTION. The name of the input file is INVOER. This string holds the name and the directory path of the input file. TOTAL denotes the total number of records in DETECTION.

```
PROCEDURE STRALINGSDIAGRAMHUYGENS (       THETA,PHI                  : REAL;
                                          VAR ETHETA,EPHI,FASE       : REAL);

PROCEDURE STRALINGSDIAGRAMHUYGENSCIRC (   THETA,PHI                  : REAL;
                                          VAR ETHETA,EPHI,FASE       : REAL);

PROCEDURE STRALINGSDIAGRAMCORRHL (        THETA,PHI                  : REAL;
                                          VAR ETHETA,EPHI,FASE       : REAL);

PROCEDURE STRALINGSDIAGRAMCORRHC (        THETA,PHI                  : REAL;
                                          VAR ETHETA,EPHI,FASE       : REAL);
```

The procedures STRALINGSDIAGRAM do not need further explanation. They are exactly the same as in STRAAL3D.

```
PROCEDURE SENSITIVITY (   ETHETA,EPHI,FASE    : REAL;
                          VAR RSENSE,LSENSE   : REAL);
```

SENSITIVITY rewrites the electric field components ETHETA and EPHI with a phase difference of FASE into a combination of left and right circular components, LSENSE and RSENSE respectively.
The calculation contains a rotation of the ETHETA and EPHI components over an angle, which follows from the amplitudes of ETHETA and EPHI and their phase difference FASE.

```
PROCEDURE RECEIVEDPOWERLIN (  Z           : INTEGER;
                              NUMBER      : ARRAY30;
                              T           : INTEGER;
                              TAUP        : REAL;
                              DETECTION   : DETECTARR;
                              VAR PDP     : REAL);

PROCEDURE RECEIVEDPOWERELL (  Z           : INTEGER;
                              NUMBER      : ARRAY30;
                              T           : INTEGER;
                              TAUP        : REAL;
                              DETECTION   : DETECTARR;
                              VAR PDP     : REAL);
```

RECEIVEDPOWERLIN and RECEIVEDPOWERELL calculate the receive power at moment T. The numbers of the records of the rays which arrive at this moment are listed in the array NUMBER. The information about the rays is stored in DETECTION. TAUP is the pulse width. The received power at moment T is PDP.

RECEIVEDPOWERLIN calculates the received power for a linear polarized receiving antenna. It calls the procedure STRALINGSDIAGRAM to define the sensitivity of the antenna for the different arriving rays. Furthermore RECEIVEDPOWERLIN shapes the pulses in a cosine shape.

RECEIVEDPOWERELL does very much the same for a circular polarized antenna. It also shapes the arriving pulses and calls STRALINGSDIAGRAM for defining the sensitivity of the antenna for the arriving rays. But now this sensitivity is rewritten to a sensitivity for circular polarized waves. This is done by the procedure SENSITIVITY. The same action is taken for the arriving rays. The results are combined to the received power at moment T.

```
PROCEDURE POLPOWER (    DETECTION     : DETECTARR;
                        TOTAL         : INTEGER;
                        TAUP          : REAL;
                        VAR PDP       : ARRAY999);
```

POLPOWER is called from the main program and it looks for all rays arriving at the same moment. Its output is the power delay profile, stored in PDP. The arrival data of the rays is stored in DETECTION. Again TOTAL is the number of records in DETECTION and TAUP is the pulse width.

POLPOWER looks for every ray arriving at the receiver close to a moment T. When a ray is found, the number of the record of DETECTION containing the information of the ray is stored. If all the rays for a particular moment are found POLPOWER calls RECEIVEDPOWERLIN or RECEIVEDPOWERELL, depending on the type of receiving antenna. The value of the received power is stored in PDP[T]

```
PROCEDURE STOREPDP (    UITVOER       : STRING;
                        PDP           : ARRAY999);
```

As explained earlier STOREPDP writes the normalized power delay profile to the file contained in the string UITVOER.

```
PROCEDURE STATISTICS (  PDP                   : ARRAY999;
                        TAUP                  : REAL;
                        VAR AMD,ADS,GAIN      : REAL;
                        VAR PDPN              : ARRAY999);
```

Also explained earlier the procedure STATISTICS calculates the parameters average mean delay (AMD), average delay spread (ADS) and the power gain (GAIN) from the received power stored in PDP. GAIN is calculated by integrating the power delay profile. TAUP is the pulse width.

Furthermore is the power delay profile normalized to PDPN.

G.2.3. POWER.

In POWER there are no new data types, which are not already explained earlier in STRAAL3D or POLPOWER.

In the main programme of POWER the same statements for obtaining the information about the input and output files come back. These will not be described again.

Furthermore the part in which the rays arriving at the same time is now kept in the main programme. It is the same as in POLPOWER. Also the part in which the pulses are shaped, is kept in the main programme. This because the programme is very short.

After the power delay profile is calculated, the procedure STATISTICS is called. This procedure calculates the parameters from the power delay profile and normalizes it. Of course it is the same procedure as in POLPOWER. The results are written to the monitor and the normalized power delay profile to the output file specified at the start of the programme. This is done by the procedure STOREPDP.

G.2.4. CUMPOWER.

The main programme of CUMPOWER is listed below.

```
BEGIN
        WRITELN ('GEEF DE UITVOERFILE');
        READLN (UITVOER);
        WRITELN ('GEEF DE GEBRUIKTE PULSLENGTE');
        READLN (TAUP);
        UITVOER:=CONCAT('C:\TP50\MARC\CUM_PDP\',UITVOER);
        ASSIGN (BATCH,'C:\TP50\MARC\PDP\LIST.PDP'); RESET (BATCH);
        FOR I:=0 TO 999 DO PDPTOT[I]:=0;
        WHILE NOT EOF (BATCH) DO
        BEGIN
                READLN (BATCH,INVOER);
                INVOER:=CONCAT('C:\TP50\MARC\PDP\',INVOER);
                ASSIGN (INV,INVOER); RESET (INV);
                FOR I:=0 TO 999 DO
                BEGIN
                        READLN(INV,X,PD);
                        PD:=EXP(PD*LN(10)/10);
                        PDPTOT[I]:=PDPTOT[I]+PD;
                END;
                CLOSE (INV);
        END;
        CLOSE (BATCH);
```

```
23          STATISTICS (PDPTOT,TAUP,AMD,ADS,GAIN,PDPN);
24          WRITELN ('AMD = ',AMD,' ns');
25          WRITELN ('ADS = ',SQRT(ADS),' ns');
26          WRITELN ('PG  = ',GAIN);
27          STOREPDP (UITVOER,PDPN);
28      END.
```

Now only the output file and the pulse width have to be specified. The input files are listed in the text file LIST.PDP. In line 7 this file is opened and in line 11 the names of the input files are read from LIST.PDP.

In lines 16 to 17 the logarithmic values of the power delay profiles in the input files are converted to linear ones. These values are then added to PDPTOT. This is done for all input files.
STATISTICS is the called to calculate the parameters AMD, ADS and GAIN and to normalize the POWER delay profile. The parameter are written to the monitor and STOREPDP is used to write the normalized cumulative power delay profile to the output file.

G.3. Manual software.

G.3.1. STRAAL3D.

Before STRAAL3D can be executed, some actions have to be taken. In the main program there are some statements putting strings together. These are the concat statements. The string in the concat statements with 'C:\' must be altered to the desired directory path of the input or output files. The back slash \ after the name of the directory where the files are stored must not be forgotten.
Furthermore does the string in the statement INITGRAPH have to point to the directory where the TurboPascal files are stored in.
If the polarization dependency of the rays is not considered, the calls to the procedures POLARIZATION and ARRIVAL2 must be between braces, { and }. These braces must include the parameter list.
If the polarization dependency is considered, the procedure call ARRIVAL must be between braces.
The last action one has to take before running the programme is checking if the procedure POLARIZATION calls the right STRALINGSDIAGRAM procedure for defining the transmitter antenna. This call occurs five times in POLARIZATION. Of coarse this is not necessary in the case where the polarization is not considered.
The frequency has to be defined in the programmes POLPOWER and POWER only. It has to be done in the procedures RECEIVEDPOWERLIN (Listing page 151) and RECEIVEDPOWERELL (Listing page 152) of POLPOWER and in the main programme of POWER (Listing page 159).

G.3.2. POLPOWER.

Also in POLPOWER one has to define the directory paths in the concat statements of the main programme.

Furthermore one has to check if the call in the procedure POLPOWER to RECEIVEDPOWERELL or RECEIVEDPOWERLIN is correct. RECEIVEDPOWERLIN is for a linear polarized receiving antenna and RECEIVEDPOWERELL for a circular polarized antenna. This call occurs one time at the end of the procedure POLPOWER.

Also in the procedures RECEIVEDPOWERLIN and RECEIVEDPOWERELL the call to the procedures STRALINGSDIAGRAM for the right receiving antenna have to be checked. This is also just one call, now at the beginning of the procedures RECEIVEDPOWER...

G.3.3. POWER.

In this programme only the directory paths in the concat statements in the main programme have to be checked.

G.3.4. CUMPOWER.

Again the directory paths in the main programme have to be checked, but also the directory path of the assign statement to open the file LIST.PDP. This file must be present in the directory specified and it must contain the names of the input files. An example of LIST.PDP is given below.

```
RC1.A
RC1.B
RC1.C
RC1.D
RC1.E
RC1.F
RC1.G
RC1.H
RC1.I
RC1.J
```

GOOD LUCK !

G.5. Listing software.

G.5.1. STRAAL3D.

```
PROGRAM STRAAL3D (INPUT,OUTPUT);

{$M 32768,0,655360}                                         {increasing stack}

USES CRT,GRAPH;

TYPE ARRAY6 = ARRAY[1..6] OF REAL;
     OBJECT = RECORD
          X: ARRAY[1..18] OF REAL;
          U: ARRAY[1..18] OF REAL;
          V: ARRAY[1..18] OF REAL;
          R: ARRAY[1..18] OF REAL;
     END;
     CONFIGURATION =  ARRAY[0..10] OF OBJECT;
     ARRAY1_5      =  ARRAY[1..20,1..5] OF REAL;
     ARRAY2_10     =  ARRAY[1..50,1..10] OF REAL;
     ARRAY3_15     =  ARRAY[1..50,1..15] OF REAL;
     ARRAY4_20     =  ARRAY[1..100,1..20] OF REAL;
     PLANE1        =  ARRAY[1..20] OF CHAR;
     PLANE2        =  ARRAY[1..50,1..2] OF CHAR;
     PLANE3        =  ARRAY[1..50,1..3] OF CHAR;
     PLANE4        =  ARRAY[1..100,1..4] OF CHAR;
     DETECTRECORD = RECORD
          LENGTH,
          ATHETA,APHI,
          PHASE,
          THETA,PHI   : REAL;
     END;
     DETECTARR     =  ARRAY [1..200] OF DETECTRECORD;


VAR  AANTAL,AANTAL1,
     AANTAL2,AANTAL3,
     AANTAL4,
     I,J,GM,GD,TOTAL  :  INTEGER;
     T                :  ARRAY6;
     CONF             :  CONFIGURATION;
     INVOER,UITVOER   :  STRING;
     EXISTO           :  BOOLEAN;
     RAY1             :  ARRAY1_5;
     RAY2             :  ARRAY2_10;
     RAY3             :  ARRAY3_15;
     RAY4             :  ARRAY4_20;
     UIT              :  TEXT;
     DETECTREC        :  DETECTARR;
     PL1                :    PLANE1;
     PL2                :    PLANE2;
     PL3                :    PLANE3;
     PL4                :    PLANE4;


PROCEDURE ROOMCONF (  VAR C          : CONFIGURATION;
                      VAR T          : ARRAY6;
                      VAR AANTAL     : INTEGER;
                      INVOER         : STRING);

     VAR I,J    : INTEGER;
         INP    : TEXT;
         XY     : ARRAY[1..10] OF REAL;

BEGIN
     ASSIGN (INP,INVOER); RESET(INP);                          {opening input file}
     FOR I:=0 TO 10 DO                                                 {initialize}
     BEGIN
          WITH C[I] DO
          BEGIN
               FOR J:=1 TO 18 DO
               BEGIN
                    U[J]:=0; V[J]:=0; X[J]:=0; R[J]:=0;
               END;
          END;
     END;
     FOR I:=1 TO 6 DO READ (INP,T[I]);        {read position transmitter and revceiver}
     READLN (INP,AANTAL);                      {read total amount of objects in room}
     FOR I:=0 TO AANTAL DO
     BEGIN
          FOR J:=1 TO 10 DO READ (INP,XY[J]);             {read coordinates of objects}
          WITH C[I] DO               {calculate corner points and vectors to define surfaces}
          BEGIN
               X[1]:=XY[1];       U[1]:=XY[3]-XY[1];      V[1]:=0;
               X[2]:=XY[2];       U[2]:=XY[4]-XY[2];      V[2]:=0;
               X[3]:=XY[9];       U[3]:=0;                V[3]:=XY[10]-XY[9];
               X[4]:=XY[3];       U[4]:=XY[5]-XY[3];      V[4]:=0;
               X[5]:=XY[4];       U[5]:=XY[6]-XY[4];      V[5]:=0;
               X[6]:=XY[9];       U[6]:=0;                V[6]:=XY[10]-XY[9];
               X[7]:=XY[5];       U[7]:=XY[7]-XY[5];      V[7]:=0;
               X[8]:=XY[6];       U[8]:=XY[8]-XY[6];      V[8]:=0;
               X[9]:=XY[9];       U[9]:=0;                V[9]:=XY[10]-XY[9];
               X[10]:=XY[7];      U[10]:=XY[1]-XY[7];     V[10]:=0;
               X[11]:=XY[8];      U[11]:=XY[2]-XY[8];     V[11]:=0;
               X[12]:=XY[9];      U[12]:=0;               V[12]:=XY[10]-XY[9];
               X[13]:=XY[1];      U[13]:=XY[3]-XY[1];     V[13]:=XY[7]-XY[1];
               X[14]:=XY[2];      U[14]:=XY[4]-XY[2];     V[14]:=XY[8]-XY[2];
               X[15]:=XY[9];      U[15]:=0;               V[15]:=0;
               X[16]:=XY[1];      U[16]:=XY[3]-XY[1];     V[16]:=V[13];
               X[17]:=XY[2];      U[17]:=XY[4]-XY[2];     V[17]:=V[14];
               X[18]:=XY[10];     U[18]:=0;               V[18]:=0;
          END;
```

```
        J:=1;
        WHILE J<18 DO
        BEGIN                                             {re-initialize index variables}
              IF J MOD 3 = 0 THEN J:=J+1;
              READ (INP,C[I].R[J]); J:=J+1;{read dielectric permittivity or reflection coefficient}
        END;
    END;
    CLOSE (INP);
END;



PROCEDURE SCREENLINE (X1,Y1,Z1,X2,Y2,Z2: REAL);

VAR  XS1,YS1,XS2,YS2: REAL;

BEGIN
    XS1:=X1+(300-X1)*(Y1/300);                      {perform coordinate transformation}
    YS1:=0.5*Y1+(300-Y1)*(Z1/100);
    XS2:=X2+(300-X2)*(Y2/300);
    YS2:=0.5*Y2+(300-Y2)*(Z2/100);
    LINE (ROUND(XS1),ROUND(320-YS1),                              {draw line on screen}
        ROUND(XS2),ROUND(320-YS2));
END;



PROCEDURE ROOMDRAW (  C             : CONFIGURATION;
                      T             : ARRAY6;
                      AANTAL        : INTEGER);

    VAR  I,J     : INTEGER;
         Z1,Z2   : REAL;

    BEGIN                                          {draw room configuration on screen}
        FOR I:=0 TO AANTAL DO
        BEGIN
            WITH C[I] DO
            BEGIN
                J:=1;
                WHILE J<=9 DO
                BEGIN
                    SCREENLINE (X[J],X[J+1],X[3],X[J+3],X[J+4],X[3]);
                    J:=J+3;
                END;
                SCREENLINE (X[1],X[2],X[3],X[10],X[11],X[3]);
                J:=1;
                WHILE J<=9 DO
                BEGIN
                    SCREENLINE (X[J],X[J+1],X[18],X[J+3],X[J+4],X[18]);
                    J:=J+3;
                END;
```

```
                    SCREENLINE (X[1],X[2],X[18],X[10],X[11],X[18]);
                    J:=1;
                    WHILE J<=9 DO
                    BEGIN
                         SCREENLINE (X[J],X[J+1],X[3],X[J],X[J+1],X[18]);
                         J:=J+3;
                    END;
                    SCREENLINE (X[10],X[11],X[3],X[10],X[11],X[18]);
              END;
        END;
        SCREENLINE (T[1],T[2],T[3],T[1],T[2],0);
        SCREENLINE (T[4],T[5],T[6],T[4],T[5],0);
   END;




PROCEDURE MIRROR ( QX,QY,QZ,
                   AX,AY,AZ,
                   VX,VY,VZ,
                   UX,UY,UZ          : REAL;
                   VAR MX,MY,MZ      : REAL);

VAR    PX,PY,PZ,
       LAMBDA,MU,
       LU,LV,
       BX,BY,BZ    : REAL;

BEGIN
       LU:=SQR(UX)+SQR(UY)+SQR(UZ);
       LV:=SQR(VX)+SQR(VY)+SQR(VZ);
       BX:=QX-AX;
       BY:=QY-AY;
       BZ:=QZ-AZ;
       LAMBDA:=(BX*UX+BY*UY+BZ*UZ)/LU;
       MU:=(BX*VX+BY*VY+BZ*VZ)/LV;
       PX:=AX+LAMBDA*UX+MU*VX;                          {crossing point with objects}
       PY:=AY+LAMBDA*UY+MU*VY;
       PZ:=AZ+LAMBDA*UZ+MU*VZ;
       MX:=QX+2*(PX-QX);                                            {mirror point}
       MY:=QY+2*(PY-QY);
       MZ:=QZ+2*(PZ-QZ);
END;
```

```
PROCEDURE CROSSPLANE ( QX,QY,QZ,
                       MX,MY,MZ,
                       AX,AY,AZ,
                       UX,UY,UZ,
                       VX,VY,VZ             : REAL;
                       VAR PX,PY,PZ         : REAL;
                       VAR CROSSING         : BOOLEAN);

VAR   A1,A2,A3,A4,A5,A6,
      A7,A8,A9,A10,A11,A12,
      MU,LAMBDA,GAMMA      : REAL;

BEGIN                                      {calculating three variables from three equations}
      CROSSING:=FALSE;
      A1:=UX;
      A5:=UY;
      A9:=UZ;
      A2:=VX;
      A6:=VY;
      A10:=VZ;
      A3:=QX-MX;
      A7:=QY-MY;
      A11:=QZ-MZ;
      A4:=QX-AX;
      A8:=QY-AY;
      A12:=QZ-AZ;
      IF A9<>0 THEN
      BEGIN
            A2:=A2*A9-A10*A1;
            A3:=A3*A9-A11*A1;
            A4:=A4*A9-A12*A1;
            A6:=A6*A9-A10*A5;
            A7:=A7*A9-A11*A5;
            A8:=A8*A9-A12*A5;
            IF A2<>0 THEN
            BEGIN
                  A7:=A7*A2-A3*A6;
                  A8:=A8*A2-A4*A6;
                  A9:=A9*A2;
                  A11:=A11*A2-A3*A10;
                  A12:=A12*A2-A4*A10;
                  IF A7=0 THEN GAMMA:=10 ELSE GAMMA:=A8/A7;
                  MU:=(A4-A3*GAMMA)/A2;
                  LAMBDA:=(A12-A11*GAMMA)/A9;
            END ELSE
            BEGIN
                  IF A3=0 THEN GAMMA:=10 ELSE GAMMA:=A4/A3;
                  IF A6=0 THEN GAMMA:=10 ELSE MU:=(A8-A7*GAMMA)/A6;
                  LAMBDA:=(A12-A10*MU-A11*GAMMA)/A9;
            END
      END
```

```
    ELSE IF A5<>0 THEN
    BEGIN
        A2:=A2*A5-A6*A1;
        A3:=A3*A5-A7*A1;
        A4:=A4*A5-A8*A1;
        IF A2<>0 THEN
        BEGIN
            A5:=A5*A2;
            A7:=A7*A2-A3*A6;
            A8:=A8*A2-A4*A6;
            A11:=A11*A2-A3*A10;
            A12:=A12*A2-A4*A10;
            IF A11=0 THEN GAMMA:=10 ELSE GAMMA:=A12/A11;
            MU:=(A4-A3*GAMMA)/A2;
            LAMBDA:=(A8-A7*GAMMA)/A5;
        END ELSE
        BEGIN
            IF A3=0 THEN GAMMA:=10 ELSE GAMMA:=A4/A3;
            IF A10=0 THEN GAMMA:=10 ELSE MU:=(A12-A11*GAMMA)/A10;
            LAMBDA:=(A8-A7*GAMMA-A6*MU)/A5;
        END;
    END
    ELSE
   BEGIN
        IF A6<>0 THEN
        BEGIN
            A1:=A1*A6;
            A3:=A3*A6-A7*A2;
            A4:=A4*A6-A8*A2;
            A2:=0;
            A11:=A11*A6-A7*A10;
            A12:=A12*A6-A8*A10;
            A10:=0;
            IF A11<>0 THEN GAMMA:=A12/A11 ELSE GAMMA:=10;
            IF A6<>0 THEN MU:=(A8-A7*GAMMA)/A6 ELSE GAMMA:=10;
            IF A1<>0 THEN LAMBDA:=(A4-A3*GAMMA)/A1 ELSE GAMMA:=10;
        END
        ELSE
        IF A10<>0 THEN
        BEGIN
            IF A7<>0 THEN GAMMA:=A8/A7 ELSE GAMMA:=10;
            IF A10<>0 THEN MU:=(A12-A11*GAMMA)/A10 ELSE GAMMA:=0;
            IF A1<>0 THEN LAMBDA:=(A4-A3*GAMMA-A2*MU)/A1 ELSE
                                        GAMMA:=10;
        END;
    END;
    IF ((LAMBDA>=0.000001) AND (LAMBDA<=0.999999)               {if conditions are true}
        AND (MU>=0.000001) AND (MU<=0.999999)                 {the reflection takes place}
        AND (GAMMA>=0.000001) AND (GAMMA<=0.999999)) THEN CROSSING:=TRUE;
    PX:=AX+LAMBDA*UX+MU*VX;
    PX:=AY+LAMBDA*UY+MU*VY;
    PZ:=AZ+LAMBDA*UZ+MU*VZ;
END;
```

```
PROCEDURE DIRECT ( C          : CONFIGURATION;
                   T          : ARRAY6;
                   AANTAL     : INTEGER;
                   VAR EXIST  : BOOLEAN);

VAR  I,J        : INTEGER;
     PX,PY,PZ   : REAL;
     CROSS      : BOOLEAN;

BEGIN
     EXIST:=TRUE;
     CROSS:=FALSE;
     I:=0;
     WHILE ((I<=AANTAL) AND (CROSS=FALSE)) DO                    {for all objects do}
     BEGIN
          J:=1;
          WHILE ((J<=18) AND (CROSS=FALSE)) DO                   {for all surfaces do}
          BEGIN
               WITH C[I] DO
               BEGIN
                    CROSSPLANE (  T[1],T[2],T[3],T[4],T[5],T[6],       {check if line}
                                  X[J],X[J+1],X[J+2],                {crosses surface}
                                  U[J],U[J+1],U[J+2],
                                  V[J],V[J+1],V[J+2],
                                  PX,PY,PZ, CROSS);
                    J:=J+3;
               END;
          END;
          I:=I+1;
     END;
     IF CROSS=TRUE THEN EXIST:=FALSE
     ELSE
          SCREENLINE (T[1],T[2],T[3],T[4],T[5],T[6]);
END;
```

```
PROCEDURE CROSSLINES1 (      XC,YC,ZC          : REAL;
                             T                 : ARRAY6;
                             C                 : CONFIGURATION;
                             A                 : INTEGER;
                             VAR CROSSING      : BOOLEAN);

VAR  I,J                : INTEGER;
     CROSS1,CROSS2      : BOOLEAN;
     PX,PY,PZ           : REAL;

BEGIN
     CROSSING:=FALSE;
     CROSS1:=FALSE;
     CROSS2:=FALSE;
     I:=0;
     WHILE ((I<=AANTAL) AND (CROSSING=FALSE)) DO                    {for all objects}
     BEGIN
          J:=1;
          WHILE ((J<=18) AND (CROSSING=FALSE)) DO                   {for all surfaces}
          BEGIN
               WITH C[I] DO
               BEGIN
                    CROSSPLANE (  T[1],T[2],T[3],             {does line from transmitter}
                                  XC,YC,ZC,                   {to reflection point crosses}
                                  X[J],X[J+1],X[J+2],                       {the surface}
                                  U[J],U[J+1],U[J+2],
                                  V[J],V[J+1],V[J+2],
                                  PX,PY,PZ,CROSS1);
                    IF CROSS1=FALSE THEN
                    CROSSPLANE (  XC,YC,ZC,
                                  T[4],T[5],T[6],
                                  X[J],X[J+1],X[J+2],
                                  U[J],U[J+1],U[J+2],
                                  V[J],V[J+1],V[J+2],
                                  PX,PY,PZ,CROSS2);
                    IF ((CROSS1=TRUE) OR (CROSS2=TRUE)) THEN
                    CROSSING:=TRUE;
                    J:=J+3;
               END;
          END;
          I:=I+1;
     END;
END;
```

```
PROCEDURE REFLECTION1 ( C            : CONFIGURATION;
                        T            : ARRAY6;
                        A            : INTEGER;
                        VAR RAY      : ARRAY1_5;
                        VAR Z        : INTEGER;
                        VAR PLANE    : PLANE1);

VAR  I,J                : INTEGER;
     MX,MY,MZ,
     XC,YC,ZC,
     PX,PY,PZ           : REAL;
     CROSSING,CROSS     : BOOLEAN;

BEGIN
     Z:=0;
     CROSS:=FALSE;
     CROSSING:=TRUE;
     I:=0;
     WHILE I<=A DO                                                      {for all objects}
     BEGIN
          J:=1;
          WHILE J<=18 DO                                                {for all surfaces}
          BEGIN
               WITH C[I] DO
               BEGIN
                    MIRROR ( T[1],T[2],T[3],X[J],X[J+1],X[J+2],         {mirror transmitter}
                             U[J],U[J+1],U[J+2],V[J],V[J+1],V[J+2],            {in surface}
                             MX,MY,MZ);
                    CROSSPLANE ( T[4],T[5],T[6],MX,MY,MZ,         {check if line from receiver}
                                 X[J],X[J+1],X[J+2],                     {to mirror point of}
                                 U[J],U[J+1],U[J+2],                  {transmitter crosses}
                                 V[J],V[J+1],V[J+2],            {surface used for mirroring}
                                 XC,YC,ZC,CROSS);                        {the transmitter}
                    IF CROSS=TRUE THEN CROSSLINES1 ( XC,YC,ZC,          {if so, check if}
                                                     T,C,A,CROSSING);   {no other}
                    IF CROSSING=FALSE THEN                             {oblect is crossed}
                    BEGIN
                         Z:=Z+1;                               {storing reflection point}
                         RAY[Z,1]:=XC;                            {and reflection data}
                         RAY[Z,2]:=YC;
                         RAY[Z,3]:=ZC;
                         RAY[Z,4]:=R[J];
                         RAY[Z,5]:=R[J+1];
                         IF ((J=1) OR (J=4) OR (J=7) OR (J=10)) THEN        {side plane}
                              PLANE[Z]:='Z' ELSE PLANE[Z]:='U';          {or upper plane}
                         SCREENLINE (T[1],T[2],T[3],XC,YC,ZC);
                         SCREENLINE (XC,YC,ZC,T[4],T[5],T[6]);
                         CROSSING:=TRUE;
                    END;
               END;
               J:=J+3;
          END;
          I:=I+1;
     END;
END;
```

```
PROCEDURE CROSSLINES2 (       XC1,YC1,ZC1,
                              XC2,YC2,ZC2       : REAL;
                              T                 : ARRAY6;
                              A                 : INTEGER;
                              C                 : CONFIGURATION;
                              VAR CROSSING      : BOOLEAN);

VAR   XX,Y,Z                  : REAL;
      I,J                     : INTEGER;
      CROSS1,CROSS2,CROSS3    : BOOLEAN;

BEGIN
      CROSSING:=FALSE;
      I:=0;
      WHILE ((I<=A) AND (CROSSING=FALSE)) DO                    {for all objects}
      BEGIN
           J:=1;
           WHILE ((J<=18) AND (CROSSING=FALSE)) DO               {for all surfaces}
           BEGIN
                WITH C[I] DO
                BEGIN
                     CROSS3:=TRUE;
                     CROSSPLANE (   XC1,YC1,ZC1,           {does line crosses a surface}
                                    XC2,YC2,ZC2,
                                    X[J],X[J+1],X[J+2],
                                    U[J],U[J+1],U[J+2],
                                    V[J],V[J+1],V[J+2],
                                    XX,Y,Z,CROSS1);
                     IF CROSS1=FALSE THEN
                     BEGIN
                          CROSSPLANE (   T[1],T[2],T[3],
                                         XC1,YC1,ZC1,
                                         X[J],X[J+1],X[J+2],
                                         U[J],U[J+1],U[J+2],
                                         V[J],V[J+1],V[J+2],
                                         XX,Y,Z,CROSS2);
                          IF CROSS2=FALSE THEN
                          CROSSPLANE (   T[4],T[5],T[6],
                                         XC2,YC2,ZC2,
                                         X[J],X[J+1],X[J+2],
                                         U[J],U[J+1],U[J+2],
                                         V[J],V[J+1],V[J+2],
                                         XX,Y,Z,CROSS3);
                     END ELSE CROSSING:=TRUE;
                     IF CROSS3=FALSE THEN J:=J+3 ELSE
                                    CROSSING:=TRUE;
                END;
           END;
           I:=I+1;
      END;
END;
```

```
PROCEDURE REFLECTION2 ( C            : CONFIGURATION;
                        T            : ARRAY6;
                        A            : INTEGER;
                        VAR RAY      : ARRAY2_10;
                        VAR Z        : INTEGER;
                        VAR PLANE    : PLANE2);

VAR  I,J,K,L                         : INTEGER;
     XM1,YM1,ZM1,
     XM2,YM2,ZM2,
     XC1,YC1,ZC1,
     XC2,YC2,ZC2                     : REAL;
     CROSS1,CROSS2,CROSSING          : BOOLEAN;

BEGIN                                                  {same structure as REFLECTION1}
     Z:=0;                                               {but now with more mirroring}
     CROSSING:=TRUE;
     I:=0;
     WHILE I<=A DO
     BEGIN
          J:=1;
          WHILE J<=18 DO
          BEGIN
               MIRROR (  T[1],T[2],T[3],
                         C[I].X[J],C[I].X[J+1],C[I].X[J+2],
                         C[I].U[J],C[I].U[J+1],C[I].U[J+2],
                         C[I].V[J],C[I].V[J+1],C[I].V[J+2],
                         XM1,YM1,ZM1);
               K:=0;
               WHILE K<=A DO
               BEGIN
                    L:=1;
                    WHILE L<=18 DO
                    BEGIN
                         IF ((L=J) AND (K=I)) THEN ELSE
                         BEGIN
                              CROSS1:=FALSE;
                              CROSS2:=FALSE;
                              MIRROR (  XM1,YM1,ZM1,
                                        C[K].X[L],C[K].X[L+1],C[K].X[L+2],
                                        C[K].U[L],C[K].U[L+1],C[K].U[L+2],
                                        C[K].V[L],C[K].V[L+1],C[K].V[L+2],
                                        XM2,YM2,ZM2);
                              CROSSPLANE (XM2,YM2,ZM2,T[4],T[5],T[6],
                                        C[K].X[L],C[K].X[L+1],C[K].X[L+2],
                                        C[K].U[L],C[K].U[L+1],C[K].U[L+2],
                                        C[K].V[L],C[K].V[L+1],C[K].V[L+2],
                                        XC2,YC2,ZC2,CROSS1);
                              IF CROSS1=TRUE THEN
                              CROSSPLANE (XC2,YC2,ZC2,XM1,YM1,ZM1,
                                        C[I].X[J],C[I].X[J+1],C[I].X[J+2],
                                        C[I].U[J],C[I].U[J+1],C[I].U[J+2],
                                        C[I].V[J],C[I].V[J+1],C[I].V[J+2],
                                        XC1,YC1,ZC1,CROSS2);
                        IF CROSS2=TRUE THEN
                        CROSSLINES2 (  XC1,YC1,ZC1,XC2,YC2,ZC2,
                                       T,A,C,CROSSING);
                        IF CROSSING=FALSE THEN
                        BEGIN
                            Z:=Z+1;
                            RAY[Z,1]:=XC1;
                            RAY[Z,2]:=YC1;
                            RAY[Z,3]:=ZC1;
                            RAY[Z,4]:=XC2;
                            RAY[Z,5]:=YC2;
                            RAY[Z,6]:=ZC2;
                            RAY[Z,7]:=C[I].R[J];
                            RAY[Z,8]:=C[I].R[J+1];
                            RAY[Z,9]:=C[K].R[L];
                            RAY[Z,10]:=C[K].R[L+1];
                            IF ((J=1) OR (J=4) OR (J=7) OR (J=10))
                                THEN PLANE[Z,1]:='Z'
                                ELSE PLANE[Z,1]:='U';
                            IF ((L=1) OR (L=4) OR (L=7) OR (L=10))
                                THEN PLANE[Z,2]:='Z'
                                ELSE PLANE[Z,2]:='U';
                            SCREENLINE (T[1],T[2],T[3],XC1,YC1,ZC1);
                            SCREENLINE (XC1,YC1,ZC1,XC2,YC2,ZC2);
                            SCREENLINE (XC2,YC2,ZC2,T[4],T[5],T[6]);
                            CROSSING:=TRUE;
                        END;
                    END;
                    L:=L+3;
                END;
                K:=K+1;
            END;
            J:=J+3;
        END;
        I:=I+1;
    END;
END;
```

```
PROCEDURE CROSSLINES3 (     XC1,YC1,ZC1,XC2,YC2,ZC2,
                            XC3,YC3,ZC3          : REAL;
                            T                    : ARRAY6;
                            A                    : INTEGER;
                            C                    : CONFIGURATION;
                            VAR CROSSING         : BOOLEAN);

VAR  XX,Y,Z                           : REAL;
     I,J                              : INTEGER;
     CROSS1,CROSS2,CROSS3,CROSS4      : BOOLEAN;

BEGIN
     CROSSING:=FALSE;
     I:=0;
     WHILE ((I<=A) AND (CROSSING=FALSE)) DO
     BEGIN
          J:=1;
          WHILE ((J<=18) AND (CROSSING=FALSE)) DO
          BEGIN
               WITH C[I] DO
               BEGIN
                    CROSS4:=TRUE;
                    CROSSPLANE (  XC1,YC1,ZC1,XC2,YC2,ZC2,
                                  X[J],X[J+1],X[J+2],
                                  U[J],U[J+1],U[J+2],
                                  V[J],V[J+1],V[J+2],
                                  XX,Y,Z,CROSS1);
                    IF CROSS1=FALSE THEN
                    BEGIN
                         CROSSPLANE (  T[1],T[2],T[3],XC1,YC1,ZC1,
                                       X[J],X[J+1],X[J+2],
                                       U[J],U[J+1],U[J+2],
                                       V[J],V[J+1],V[J+2],
                                       XX,Y,Z,CROSS2);
                         IF CROSS2=FALSE THEN
                         BEGIN
                              CROSSPLANE (XC3,YC3,ZC3,XC2,YC2,ZC2,
                                       X[J],X[J+1],X[J+2],
                                       U[J],U[J+1],U[J+2],
                                       V[J],V[J+1],V[J+2],
                                       XX,Y,Z,CROSS3);
                              IF CROSS3=FALSE THEN
                              CROSSPLANE (T[4],T[5],T[6],XC3,YC3,ZC3,
                                       X[J],X[J+1],X[J+2],
                                       U[J],U[J+1],U[J+2],
                                       V[J],V[J+1],V[J+2],
                                       XX,Y,Z,CROSS4);
                         END ELSE CROSSING:=TRUE;
                    END ELSE CROSSING:=TRUE;
                    IF CROSS4=FALSE THEN J:=J+3 ELSE CROSSING:=TRUE;
               END;
          END;
          I:=I+1;
     END;
END;
```

```
PROCEDURE REFLECTION3 ( C           : CONFIGURATION;
                        T           : ARRAY6;
                        A           : INTEGER;
                        VAR RAY     : ARRAY3_15;
                        VAR Z       : INTEGER;
                        VAR PLANE   : PLANE3);

VAR  I,J,K,L,M,N                            : INTEGER;
     XM1,YM1,ZM1,
     XM2,YM2,ZM2,
     XC1,YC1,ZC1,
     XC2,YC2,ZC2,
     XC3,YC3,ZC3,
     XM3,YM3,ZM3                            : REAL;
     CROSS1,CROSS2,CROSS3,CROSSING          : BOOLEAN;

BEGIN
     Z:=0;
     CROSSING:=TRUE;
     I:=0;
     WHILE I<=A DO
     BEGIN
          J:=1;
          WHILE J<=18 DO
          BEGIN
               MIRROR (  T[1],T[2],T[3],
                         C[I].X[J],C[I].X[J+1],C[I].X[J+2],
                         C[I].U[J],C[I].U[J+1],C[I].U[J+2],
                         C[I].V[J],C[I].V[J+1],C[I].V[J+2],
                         XM1,YM1,ZM1);
               K:=0;
               WHILE K<=A DO
               BEGIN
                    L:=1;
                    WHILE L<=18 DO
                    BEGIN
                         IF ((L=J) AND (I=K)) THEN ELSE
                         BEGIN
                              MIRROR (  XM1,YM1,ZM1,
                                        C[K].X[L],C[K].X[L+1],C[K].X[L+2],
                                        C[K].U[L],C[K].U[L+1],C[K].U[L+2],
                                        C[K].V[L],C[K].V[L+1],C[K].V[L+2],
                                        XM2,YM2,ZM2);
                              M:=0;
                              WHILE M<=A DO
                              BEGIN
                                   N:=1;
                                   WHILE N<=18 DO
                                   BEGIN
                                        CROSS1:=FALSE;
                                        CROSS2:=FALSE;
                                        CROSS3:=FALSE;
```

```
MIRROR (T[4],T[5],T[6],
        C[M].X[N],C[M].X[N+1],C[M].X[N+2],
        C[M].U[N],C[M].U[N+1],C[M].U[N+2],
        C[M].V[N],C[M].V[N+1],C[M].V[N+2],
        XM3,YM3,ZM3);
CROSSPLANE (XM2,YM2,ZM2,
        XM3,YM3,ZM3,
        C[M].X[N],C[M].X[N+1],C[M].X[N+2],
        C[M].U[N],C[M].U[N+1],C[M].U[N+2],
        C[M].V[N],C[M].V[N+1],C[M].V[N+2],
        XC3,YC3,ZC3,
        CROSS1);
IF CROSS1=TRUE THEN
CROSSPLANE (XM2,YM2,ZM2,
        XC3,YC3,ZC3,
        C[K].X[L],C[K].X[L+1],C[K].X[L+2],
        C[K].U[L],C[K].U[L+1],C[K].U[L+2],
        C[K].V[L],C[K].V[L+1],C[K].V[L+2],
        XC2,YC2,ZC2,
        CROSS2);
IF CROSS2=TRUE THEN
CROSSPLANE (XC2,YC2,ZC2,
        XM1,YM1,ZM1,
        C[I].X[J],C[I].X[J+1],C[I].X[J+2],
        C[I].U[J],C[I].U[J+1],C[I].U[J+2],
        C[I].V[J],C[I].V[J+1],C[I].V[J+2],
        XC1,YC1,ZC1,
        CROSS3);
IF CROSS3=TRUE THEN
CROSSLINES3 (XC1,YC1,ZC1,
        XC2,YC2,ZC2,
        XC3,YC3,ZC3,
        T,A,C,CROSSING);
IF CROSSING=FALSE THEN
BEGIN
        Z:=Z+1;
        RAY[Z,1]:=XC1;
        RAY[Z,2]:=YC1;
        RAY[Z,3]:=ZC1;
        RAY[Z,4]:=XC2;
        RAY[Z,5]:=YC2;
        RAY[Z,6]:=ZC2;
        RAY[Z,7]:=XC3;
        RAY[Z,8]:=YC3;
        RAY[Z,9]:=ZC3;

        RAY[Z,10]:=C[I].R[J];
        RAY[Z,11]:=C[I].R[J+1];
        RAY[Z,12]:=C[K].R[L];
        RAY[Z,13]:=C[K].R[L+1];
        RAY[Z,14]:=C[M].R[N];
        RAY[Z,15]:=C[M].R[N+1];
```

```
                                            IF ((J=1) OR (J=4) OR
                                            (J=7) OR (J=10))
                                                THEN PLANE[Z,1]:='Z'
                                                ELSE PLANE[Z,1]:='U';
                                            IF ((L=1) OR (L=4) OR
                                            (L=7) OR (L=10))
                                                THEN PLANE[Z,2]:='Z'
                                                ELSE PLANE[Z,2]:='U';
                                            IF ((N=1) OR (N=4) OR
                                            (N=7) OR (N=10))
                                                THEN PLANE[Z,3]:='Z'
                                                ELSE PLANE[Z,3]:='U';
                                            SCREENLINE
                                                (T[1],T[2],T[3],XC1,YC1,ZC1);
                                            SCREENLINE
                                                (XC1,YC1,ZC1,XC2,YC2,ZC2);
                                            SCREENLINE
                                                (XC2,YC2,ZC2,XC3,YC3,ZC3);
                                            SCREENLINE
                                                (XC3,YC3,ZC3,T[4],T[5],T[6]);
                                            CROSSING:=TRUE;
                                        END;
                                        N:=N+3;
                                    END;
                                    M:=M+1;
                                END;
                            END;
                            L:=L+3;
                        END;
                        K:=K+1;
                    END;
                    J:=J+3;
                END;
                I:=I+1;
            END;
        END;
```

```
PROCEDURE CROSSLINES4 (      XC1,YC1,ZC1,XC2,YC2,ZC2,
                             XC3,YC3,ZC3,XC4,YC4,ZC4      : REAL;
                             T                            : ARRAY6;
                             A                            : INTEGER;
                             C                            : CONFIGURATION;
                             VAR CROSSING                 : BOOLEAN);

VAR  XX,Y,Z                  : REAL;
     I,J                     : INTEGER;
     CROSS1,CROSS2,
     CROSS3,CROSS4,CROSS5    : BOOLEAN;

BEGIN
     CROSSING:=FALSE;
     I:=0;
     WHILE ((I<=A) AND (CROSSING=FALSE)) DO
     BEGIN
          J:=1;
          WHILE ((J<=18) AND (CROSSING=FALSE)) DO
          BEGIN
               WITH C[I] DO
               BEGIN
                    CROSS5:=TRUE;
                    CROSSPLANE (  T[1],T[2],T[3],
                                  XC1,YC1,ZC1,
                                  X[J],X[J+1],X[J+2],
                                  U[J],U[J+1],U[J+2],
                                  V[J],V[J+1],V[J+2],
                                  XX,Y,Z,CROSS1);
                    IF CROSS1=FALSE THEN
                    BEGIN
                         CROSSPLANE (  XC1,YC1,ZC1,
                                       XC2,YC2,ZC2,
                                       X[J],X[J+1],X[J+2],
                                       U[J],U[J+1],U[J+2],
                                       V[J],V[J+1],V[J+2],
                                       XX,Y,Z,CROSS2);
                         IF CROSS2=FALSE THEN
                         BEGIN
                              CROSSPLANE (  XC2,YC2,ZC2,
                                            XC3,YC3,ZC3,
                                            X[J],X[J+1],X[J+2],
                                            U[J],U[J+1],U[J+2],
                                            V[J],V[J+1],V[J+2],
                                            XX,Y,Z,CROSS3);
                              IF CROSS3=FALSE THEN
                              BEGIN
                                   CROSSPLANE (  XC3,YC3,ZC3,
                                                 XC4,YC4,ZC4,
                                                 X[J],X[J+1],X[J+2],
                                                 U[J],U[J+1],U[J+2],
                                                 V[J],V[J+1],V[J+2],
                                                 XX,Y,Z,CROSS4);
```

```
                                        IF CROSS4=FALSE THEN
                                        CROSSPLANE (   XC4,YC4,ZC4,
                                                       T[4],T[5],T[6],
                                                       X[J],X[J+1],X[J+2],
                                                       U[J],U[J+1],U[J+2],
                                                       V[J],V[J+1],V[J+2],
                                                       XX,Y,Z,CROSS5);
                                    END ELSE CROSSING:=TRUE;
                               END ELSE CROSSING:=TRUE;
                           END ELSE CROSSING:=TRUE;
                           IF CROSS5=FALSE THEN J:=J+3 ELSE CROSSING:=TRUE;
                      END;
                 END;
                 I:=I+1;
            END;
END;
```

```
PROCEDURE REFLECTION4 ( C            : CONFIGURATION;
                        T            : ARRAY6;
                        A            : INTEGER;
                        VAR RAY      : ARRAY4_20;
                        VAR Z        : INTEGER;
                        VAR PLANE    : PLANE4);

VAR    I,J,K,L,M,N,P,Q              : INTEGER;
       XM1,YM1,ZM1,
       XM2,YM2,ZM2,
       XC1,YC1,ZC1,
       XC2,YC2,ZC2,
       XC3,YC3,ZC3,
       XM3,YM3,ZM3,
       XC4,YC4,ZC4,
       XM4,YM4,ZM4                  : REAL;
       CROSS1,CROSS2,CROSS3,
       CROSS4,CROSSING              : BOOLEAN;

BEGIN
       Z:=0;
       CROSSING:=TRUE;
       I:=0;
       WHILE I<=A DO
       BEGIN
            J:=1;
            WHILE J<=18 DO
            BEGIN
                 MIRROR (  T[1],T[2],T[3],
                           C[I].X[J],C[I].X[J+1],C[I].X[J+2],
                           C[I].U[J],C[I].U[J+1],C[I].U[J+2],
                           C[I].V[J],C[I].V[J+1],C[I].V[J+2],
                           XM1,YM1,ZM1);
```

```
K:=0;
WHILE K<=A DO
BEGIN
     L:=1;
     WHILE L<=18 DO
     BEGIN
          IF ((L=J) AND (K=I)) THEN ELSE
          BEGIN
               MIRROR (  XM1,YM1,ZM1,
                         C[K].X[L],C[K].X[L+1],C[K].X[L+2],
                         C[K].U[L],C[K].U[L+1],C[K].U[L+2],
                         C[K].V[L],C[K].V[L+1],C[K].V[L+2],
                         XM2,YM2,ZM2);
               M:=0;
               WHILE M<=A DO
               BEGIN
                    N:=1;
                    WHILE N<=18 DO
                    BEGIN
                         MIRROR (  T[4],T[5],T[6],
                                   C[M].X[N],C[M].X[N+1],
                                   C[M].X[N+2],
                                   C[M].U[N],C[M].U[N+1],
                                   C[M].U[N+2],
                                   C[M].V[N],C[M].V[N+1],
                                   C[M].V[N+2],XM4,YM4,ZM4);
                         P:=0;
                         WHILE P<=A DO
                         BEGIN
                              Q:=1;
                              WHILE Q<=18 DO
                              BEGIN
                                   IF ((P=M) AND (N=Q)) THEN ELSE
                                   BEGIN
                                        CROSS1:=FALSE;
                                        CROSS2:=FALSE;
                                        CROSS3:=FALSE;
                                        CROSS4:=FALSE;
                                        MIRROR (XM4,YM4,ZM4,
                                             C[P].X[Q],C[P].X[Q+1],
                                             C[P].X[Q+2],
                                             C[P].U[Q],C[P].U[Q+1],
                                             C[P].U[Q+2],
                                             C[P].V[Q],C[P].V[Q+1],
                                             C[P].V[Q+2],
                                             XM3,YM3,ZM3);
                                        CROSSPLANE (
                                             XM2,YM2,ZM2,
                                             XM3,YM3,ZM3,
                                             C[P].X[Q],C[P].X[Q+1],
                                             C[P].X[Q+2],
                                             C[P].U[Q],C[P].U[Q+1],
                                             C[P].U[Q+2],
                                             C[P].V[Q],C[P].V[Q+1],
                                             C[P].V[Q+2],
                                             XC3,YC3,ZC3,CROSS1);
```

```
IF CROSS1=TRUE THEN
CROSSPLANE (
      XM2,YM2,ZM2,
      XC3,YC3,ZC3,
      C[K].X[L],C[K].X[L+1],
      C[K].X[L+2],
      C[K].U[L],C[K].U[L+1],
      C[K].U[L+2],
      C[K].V[L],C[K].V[L+1],
      C[K].V[L+2],
      XC2,YC2,ZC2,CROSS2);
IF CROSS2=TRUE THEN
CROSSPLANE (
      XM1,YM1,ZM1,
      XC2,YC2,ZC2,
      C[I].X[J],C[I].X[J+1],
      C[I].X[J+2],
      C[I].U[J],C[I].U[J+1],
      C[I].U[J+2],
      C[I].V[J],C[I].V[J+1],
      C[I].V[J+2],
      XC1,YC1,ZC1,CROSS3);
IF CROSS3=TRUE THEN
CROSSPLANE (
      XC3,YC3,ZC3,
      XM4,YM4,ZM4,
      C[M].X[N],C[M].X[N+1],
      C[M].X[N+2],
      C[M].U[N],C[M].U[N+1],
      C[M].U[N+2],
      C[M].V[N],C[M].V[N+1],
      C[M].V[N+2],
      XC4,YC4,ZC4,CROSS4);
IF CROSS4=TRUE THEN
CROSSLINES4 (
      XC1,YC1,ZC1,
      XC2,YC2,ZC2,
      XC3,YC3,ZC3,
      XC4,YC4,ZC4,
      T,A,C,CROSSING);
IF CROSSING=FALSE THEN
BEGIN
      Z:=Z+1;
      RAY[Z,1]:=XC1;
      RAY[Z,2]:=YC1;
      RAY[Z,3]:=ZC1;
      RAY[Z,4]:=XC2;
      RAY[Z,5]:=YC2;
      RAY[Z,6]:=ZC2;
      RAY[Z,7]:=XC3;
      RAY[Z,8]:=YC3;
      RAY[Z,9]:=ZC3;
      RAY[Z,10]:=XC4;
      RAY[Z,11]:=YC4;
      RAY[Z,12]:=ZC4;
      RAY[Z,13]:=C[I].R[J];
```

```
                                                                                RAY[Z,14]:=C[I].R[J+1];
                                                                                RAY[Z,15]:=C[K].R[L];
                                                                                RAY[Z,16]:=
                                                                                        C[K].R[L+1];
                                                                                RAY[Z,17]:=C[P].R[Q];
                                                                                RAY[Z,18]:=
                                                                                        C[P].R[Q+1];
                                                                                RAY[Z,19]:=C[M].R[N];
                                                                                RAY[Z,20]:=
                                                                                        C[M].R[N+1];
                                                                                IF ((J=1) OR (J=4) OR
                                                                                        (J=7) OR (J=10))
                                                                                  THEN PLANE[Z,1]:='Z'
                                                                                  ELSE PLANE[Z,1]:='U';
                                                                                IF ((L=1) OR (L=4)
                                                                                OR (L=7) OR (L=10))
                                                                                  THEN PLANE[Z,2]:='Z'
                                                                                  ELSE PLANE[Z,2]:='U';
                                                                                IF ((Q=1) OR (Q=4)
                                                                                OR (Q=7) OR (Q=10))
                                                                                  THEN PLANE[Z,3]:='Z'
                                                                                  ELSE PLANE[Z,3]:='U';
                                                                                IF ((N=1) OR (N=4)
                                                                                OR (N=7) OR (N=10))
                                                                                  THEN PLANE[Z,4]:='Z'
                                                                                  ELSE PLANE[Z,4]:='U';
                                                                                SCREENLINE (
                                                                                        T[1],T[2],T[3],
                                                                                        XC1,YC1,ZC1);
                                                                                SCREENLINE (
                                                                                        XC1,YC1,ZC1,
                                                                                        XC2,YC2,ZC2);
                                                                                SCREENLINE (
                                                                                        XC2,YC2,ZC2,
                                                                                        XC3,YC3,ZC3);
                                                                                SCREENLINE (
                                                                                        XC3,YC3,ZC3,
                                                                                        XC4,YC4,ZC4);
                                                                                SCREENLINE (
                                                                                        XC4,YC4,ZC4,
                                                                                        T[4],T[5],T[6]);
                                                                                CROSSING:=TRUE;
                                                                          END;
                                                                   END; Q:=Q+3;
                                                            END; P:=P+1;
                                                      END; N:=N+3;
                                                END; M:=M+1;
                                          END;
                                   END; L:=L+3;
                             END; K:=K+1;
                       END; J:=J+3;
                 END; I:=I+1;
           END;
     END;
```

```
PROCEDURE ANGLECALC (  XT,YT,ZT,
                       XC1,YC1,ZC1                : REAL;
                       VAR THETA,PHI,PHIREC       : REAL);

VAR   PHIHELP,THETAHELP:  REAL;

BEGIN
      IF XT<>XC1 THEN
            PHIHELP:=ARCTAN ((YC1-YT)/(XC1-XT));
      THETAHELP:=SQRT(SQR(XT-XC1)+SQR(YT-YC1));
      IF THETAHELP=0 THEN THETA:=0.5*PI*(ZT-ZC1)/ABS(ZT-ZC1)
                  ELSE THETA:=ARCTAN ((ZT-ZC1)/THETAHELP);
      IF (XT=XC1) THEN
      BEGIN
            IF (YT<YC1) THEN
            BEGIN
                  PHI:=PI/2;
                  PHIREC:=3*PI/2
            END
            ELSE IF YT>YC1 THEN
            BEGIN
                  PHI:=3*PI/2;
                  PHIREC:=PI/2
            END
            ELSE
            BEGIN
                  PHI:=0;
                  PHIREC:=0;
            END;
      END
      ELSE
            IF ((XC1>XT) AND (YC1>=YT)) THEN
            BEGIN
                  PHI:=PHIHELP;
                  PHIREC:=PI+PHI;
            END
            ELSE IF ((XC1<XT) AND (YC1>=YT)) THEN
            BEGIN
                  PHI:=PI+PHIHELP;
                  PHIREC:=PI+PHI;
            END
            ELSE IF ((XC1<XT) AND (YC1<YT)) THEN
            BEGIN
                  PHI:=PI+PHIHELP;
                  PHIREC:=PHI-PI;
            END
            ELSE IF ((XC1>XT) AND (YC1<YT)) THEN
            BEGIN
                  PHI:=2*PI+PHIHELP;
                  PHIREC:=PHI-PI;
            END;
END;
```

```
PROCEDURE REFLCOEFF (  PERP,PAR,THETA,EPS   : REAL;
                       VAR PERPUIT,PARUIT   : REAL);

VAR   RPAR1,RPAR2,
      RPERP1,RPERP2  : REAL;

BEGIN
      IF EPS=0 THEN
      BEGIN
           PERPUIT:=-PERP;
           PARUIT:=PAR;
      END
      ELSE IF EPS=1 THEN
      BEGIN
           PERPUIT:=0;
           PARUIT:=0
      END ELSE
      BEGIN
           RPERP1:=COS(THETA)-SQRT(EPS-SQR(SIN(THETA)));
           RPERP2:=COS(THETA)+SQRT(EPS-SQR(SIN(THETA)));
           RPAR1:=EPS*COS(THETA)-SQRT(EPS-SQR(SIN(THETA)));
           RPAR2:=EPS*COS(THETA)+SQRT(EPS-SQR(SIN(THETA)));
           PERPUIT:=PERP*RPERP1/RPERP2;
           PARUIT:=PAR*RPAR1/RPAR2;
      END;
END;



PROCEDURE TRANSFORMATION (  X1,Y1,Z1,
                            X2,Y2,Z2,
                            X3,Y3,Z3                : REAL;
                            VAR GAMMA,THETAINV      : REAL);

VAR   L1,L2,
      X5,Y5,Z5,
      X6,Y6,Z6,
      X4,Y4,Z4,
      X0,Y0,Z0,
      HULP,
      X15,X05,X06,X56   : REAL;

BEGIN
      L1:=SQRT(SQR(X1-X2)+SQR(Y1-Y2)+SQR(Z1-Z2));
      L2:=SQRT(SQR(X3-X2)+SQR(Y3-Y2)+SQR(Z3-Z2));
      X5:=X1;
      Y5:=Y1;
      Z5:=Z3;

      X6:=X2+L2*(X2-X1)/L1;
      Y6:=Y2+L2*(Y2-Y1)/L1;
      Z6:=Z5;
      X4:=0.5*(X3+X6);
      Y4:=0.5*(Y3+Y6);
      Z4:=0.5*(Z3+Z6);
```

```
        X0:=X4+L1*(X4-X6)/L2;
        Y0:=Y4+L1*(Y4-Y6)/L2;
        Z0:=Z4+L1*(Z4-Z6)/L2;

        X15:=Z1-Z5;
        X05:=SQRT(SQR(X0-X5)+SQR(Y0-Y5)+SQR(Z0-Z5));
        X06:=SQRT(SQR(X0-X6)+SQR(Y0-Y6));
        X56:=SQRT(SQR(X05)+SQR(X06));

        HULP:=X05*X56;
        IF HULP<>0 THEN GAMMA:=ARCTAN(X15*X06/HULP) ELSE GAMMA:=PI/2;
        THETAINV:=ARCTAN(SQRT(SQR(X2-X4)+SQR(Y2-Y4)+SQR(Z2-Z4))/
                                      SQRT(SQR(X3-X4)+SQR(Y3-Y4)));
END;



PROCEDURE STRALINGSDIAGRAMHUYGENS (  THETA,PHI                  : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

BEGIN
        THETA:=PI/2-ABS(THETA);
        ETHETA:=1.5*SQR(SIN(THETA));
        EPHI:=0;FASE:=0;
END;



PROCEDURE STRALINGSDIAGRAMHUYGENSCIRC (  THETA,PHI                  : REAL;
                                         VAR ETHETA,EPHI,FASE  : REAL);

BEGIN
        THETA:=PI/2-ABS(THETA);
        ETHETA:=1.5*SQR(SIN(THETA));
        EPHI:=ETHETA;
        FASE:=PI/2;
END;
```

```
PROCEDURE STRALINGSDIAGRAMCORRHL (    THETA,PHI                 : REAL;
                                      VAR ETHETA,EPHI,FASE : REAL);

VAR   N,I          : INTEGER;
      TH,GAIN      : REAL;

BEGIN
      IF THETA<0 THEN
      BEGIN
            EPHI:=0;
            ETHETA:=0;
            FASE:=0;
      END
      ELSE
      BEGIN
            TH:=PI/2-THETA;
            N:=2;
            GAIN:=1;
            FOR I:=1 TO N DO GAIN:=GAIN*COS(TH);
            GAIN:=2*(N+1)*GAIN;
            ETHETA:=GAIN*COS(PHI);
            EPHI:=-GAIN*SIN(PHI);
            FASE:=0;
      END;
END;



PROCEDURE STRALINGSDIAGRAMCORRHC (    THETA,PHI                 : REAL;
                                      VAR ETHETA,EPHI,FASE : REAL);

VAR   N                         : INTEGER;
      TH,
      PHIX,PHIY,
      GAIN,
      ETHX,ETHY,
      EPHIX,EPHIY,
      ALPHA1,ALPHA2             : REAL;

BEGIN
      IF THETA<0 THEN
      BEGIN
            EPHI:=0;
            ETHETA:=0;
            FASE:=0;
      END
      ELSE
      BEGIN
            TH:=PI/2-THETA;
            N:=2;
            GAIN:=2*(N+1)*EXP(N*LN(COS(TH)));
            PHIX:=-PHI;
            PHIY:=-PHI-PI/2;
            ETHX:=GAIN*COS(PHIX);
            ETHY:=GAIN*COS(PHIY);
            EPHIX:=-GAIN*SIN(PHIX);
            EPHIY:=-GAIN*SIN(PHIY);
            ETHETA:=SQRT(SQR(ETHX)+SQR(ETHY));
            EPHI:=SQRT(SQR(EPHIX)+SQR(EPHIY));
            IF ETHY=0 THEN ALPHA1:=PI/2 ELSE
            BEGIN
                  ALPHA1:=ARCTAN(ABS(ETHX/ETHY));
                  IF ((ETHX>0) AND (ETHY<0)) THEN ALPHA1:=PI-ALPHA1;
                  IF ((ETHX<0) AND (ETHY<0)) THEN ALPHA1:=PI+ALPHA1;
                  IF ((ETHX<0) AND (ETHY>0)) THEN ALPHA1:=-ALPHA1;
            END;
            IF EPHIY=0 THEN ALPHA2:=PI/2 ELSE
            BEGIN
                  ALPHA2:=ARCTAN(ABS(EPHIX/EPHIY));
                  IF ((EPHIX>0) AND (EPHIY<0)) THEN ALPHA2:=PI-ALPHA2;
                  IF ((EPHIX<0) AND (EPHIY<0)) THEN ALPHA2:=PI+ALPHA2;
                  IF ((EPHIX<0) AND (EPHIY>0)) THEN ALPHA2:=-ALPHA2;
            END;
            FASE:=ALPHA2-ALPHA1;
            FASE>PI THEN FASE:=FASE-2*PI;
            IF FASE<-PI THEN FASE:=FASE+2*PI;
      END;
END;



PROCEDURE PLANEREFLECTION ( SIDEPLANE                 : STRING;
                            X1,Y1,Z1,
                            X2,Y2,Z2,
                            X3,Y3,Z3,
                            ETHETA,EPHI,EPSILON       : REAL;
                            VAR ETHETAUIT,EPHIUIT     : REAL);

VAR   GAMMA,
      THETAINVAL,
      APERP,APAR,
      APERPUIT,APARUIT      : REAL;
```

```
BEGIN
     IF SIDEPLANE='Z' THEN
     BEGIN
          TRANSFORMATION (X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,GAMMA,THETAINVAL);

          APERP:=COS(GAMMA)*ETHETA+SIN(GAMMA)*EPHI;
          APAR:=-SIN(GAMMA)*ETHETA+COS(GAMMA)*EPHI;

          REFLCOEFF (     APERP,APAR,THETAINVAL,
                          EPSILON,
                          APERPUIT,APARUIT);

          ETHETAUIT:=COS(GAMMA)*APERPUIT-SIN(GAMMA)*APARUIT;
          EPHIUIT:=SIN(GAMMA)*APERPUIT+COS(GAMMA)*APARUIT;
     END
     ELSE
     BEGIN
          THETAINVAL:=ARCTAN((SQRT(SQR(X1-X2)+SQR(Y1-Y2))/ABS(Z1-Z2)));

          REFLCOEFF (     ETHETA,EPHI,THETAINVAL,
                          EPSILON,
                          ETHETAUIT,EPHIUIT);
     END;
END;




PROCEDURE POLARISATIE ( EXIST                  : BOOLEAN;
                        T                      : ARRAY6;
                        RAY1                   : ARRAY1_5;
                        RAY2                   : ARRAY2_10;
                        RAY3                   : ARRAY3_15;
                        RAY4                   : ARRAY4_20;
                        AANTAL1,AANTAL2,
                        AANTAL3,AANTAL4        : INTEGER;
                        PL1                    : PLANE1;
                        PL2                    : PLANE2;
                        PL3                    : PLANE3;
                        PL4                    : PLANE4;
                        VAR DETECTION          : DETECTARR;
                        VAR TOTAL              : INTEGER);

VAR  E0,I,J                 : INTEGER;
     L,X,
     THETAINVAL,
     THETA,PHI,PHIREC,
     ETHETA,EPHI,FASE,
     ETHETAUIT,EPHIUIT      : REAL;
```

```
BEGIN
        IF EXIST=TRUE THEN E0:=1 ELSE E0:=0;
        TOTAL:=E0+AANTAL1+AANTAL2+AANTAL3+AANTAL4;
        IF E0=1 THEN
        BEGIN
                ANGLECALC (T[1],T[2],T[3],T[4],T[5],T[6],THETA,PHI,PHIREC);
                STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
                L:=SQRT(SQR(T[1]-T[4])+SQR(T[2]-T[5])+SQR(T[3]-T[6]));
                ETHETA:=ETHETA/L;
                EPHI:=EPHI/L;
                DETECTION[1].LENGTH:=L;
                DETECTION[1].ATHETA:=ETHETA;
                DETECTION[1].APHI:=EPHI;
                DETECTION[1].PHASE:=FASE;
                DETECTION[1].THETA:=-THETA;
                DETECTION[1].PHI:=PHIREC;
        END;
        J:=E0;


        FOR I:=1 TO AANTAL1 DO
        BEGIN
                ANGLECALC (      T[1],T[2],T[3],RAY1[I,1],RAY1[I,2],RAY1[I,3],
                                 THETA,PHI,X);
                STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
                ETHETAUIT:=0;
                EPHIUIT:=0;
                L:=     SQRT(SQR(T[1]-RAY1[I,1])+SQR(T[2]-RAY1[I,2])+SQR(T[3]-RAY1[I,3]))+
                        SQRT(SQR(T[4]-RAY1[I,1])+SQR(T[5]-RAY1[I,2])+SQR(T[6]-RAY1[I,3]));

                PLANEREFLECTION (        PL1[I],T[1],T[2],T[3],
                                         RAY1[I,1],RAY1[I,2],RAY1[I,3],
                                         T[4],T[5],T[6],ETHETA,EPHI,
                                         RAY1[I,4],
                                         ETHETAUIT,EPHIUIT);

                ANGLECALC (      RAY1[I,1],RAY1[I,2],RAY1[I,3],T[4],T[5],T[6],
                                 THETA,PHI,PHIREC);
                DETECTION[I+J].LENGTH:=L;
                DETECTION[I+J].ATHETA:=ETHETAUIT/L;
                DETECTION[I+J].APHI:=EPHIUIT/L;
                DETECTION[I+J].PHASE:=FASE;
                DETECTION[I+J].THETA:=-THETA;
                DETECTION[I+J].PHI:=PHIREC;
        END;


        J:=J+AANTAL1;
        FOR I:=1 TO AANTAL2 DO
        BEGIN
                ANGLECALC (      T[1],T[2],T[3],RAY2[I,1],RAY2[I,2],RAY2[I,3],
                                 THETA,PHI,X);
                STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
                ETHETAUIT:=0;
                EPHIUIT:=0;
```

```
     L:=    SQRT(SQR(T[1]-RAY2[I,1])+SQR(T[2]-RAY2[I,2])+
                 SQR(T[3]-RAY2[I,3]))+
            SQRT(SQR(RAY2[I,1]-RAY2[I,4])+SQR(RAY2[I,2]-RAY2[I,5])+
                 SQR(RAY2[I,3]-RAY2[I,6]))+
            SQRT(SQR(T[4]-RAY2[I,4])+SQR(T[5]-RAY2[I,5])+
                 SQR(T[6]-RAY2[I,6]));

     PLANEREFLECTION (   PL2[I,1],T[1],T[2],T[3],
                         RAY2[I,1],RAY2[I,2],RAY2[I,3],
                         RAY2[I,4],RAY2[I,5],RAY2[I,6],
                         ETHETA,EPHI,
                         RAY2[I,7],
                         ETHETAUIT,EPHIUIT);
     ETHETA:=ETHETAUIT;
     EPHI:=EPHIUIT;
     PLANEREFLECTION (   PL2[I,2],RAY2[I,1],RAY2[I,2],RAY2[I,3],
                         RAY2[I,4],RAY2[I,5],RAY2[I,6],
                         T[4],T[5],T[6],
                         ETHETA,EPHI,
                         RAY2[I,9],
                         ETHETAUIT,EPHIUIT);
     ETHETA:=ETHETAUIT;
     EPHI:=EPHIUIT;
     ANGLECALC (     RAY2[I,4],RAY2[I,5],RAY2[I,6],T[4],T[5],T[6],
                     THETA,PHI,PHIREC);
     DETECTION[I+J].LENGTH:=L;
     DETECTION[I+J].ATHETA:=ETHETA/L;
     DETECTION[I+J].APHI:=EPHI/L;
     DETECTION[I+J].PHASE:=FASE;
     DETECTION[I+J].THETA:=-THETA;
     DETECTION[I+J].PHI:=PHIREC;
END;



J:=J+AANTAL2;
FOR I:=1 TO AANTAL3 DO
BEGIN
     ANGLECALC (     T[1],T[2],T[3],RAY3[I,1],RAY3[I,2],RAY3[I,3],
                     THETA,PHI,X);
     STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
     ETHETAUIT:=0;
     EPHIUIT:=0;
     L:=    SQRT(SQR(T[1]-RAY3[I,1])+SQR(T[2]-RAY3[I,2])+
                 SQR(T[3]-RAY3[I,3]))+
            SQRT(SQR(RAY3[I,1]-RAY3[I,4])+SQR(RAY3[I,2]-RAY3[I,5])+
                 SQR(RAY3[I,3]-RAY3[I,6]))+
            SQRT(SQR(RAY3[I,4]-RAY3[I,7])+SQR(RAY3[I,5]-RAY3[I,8])+
                 SQR(RAY3[I,6]-RAY3[I,9]))+
            SQRT(SQR(T[4]-RAY3[I,7])+SQR(T[5]-RAY3[I,8])+
                 SQR(T[6]-RAY3[I,9]));
```

```
        PLANEREFLECTION (    PL3[I,1],T[1],T[2],T[3],
                             RAY3[I,1],RAY3[I,2],RAY3[I,3],
                             RAY3[I,4],RAY3[I,5],RAY3[I,6],
                             ETHETA,EPHI,
                             RAY3[I,10],
                             ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        PLANEREFLECTION (    PL3[I,2],RAY3[I,1],RAY3[I,2],RAY3[I,3],
                             RAY3[I,4],RAY3[I,5],RAY3[I,6],
                             RAY3[I,7],RAY3[I,8],RAY3[I,9],
                             ETHETA,EPHI,
                             RAY3[I,12],
                             ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        PLANEREFLECTION (    PL3[I,3],RAY3[I,4],RAY3[I,5],RAY3[I,6],
                             RAY3[I,7],RAY3[I,8],RAY3[I,9],
                             T[4],T[5],T[6],
                             ETHETA,EPHI,
                             RAY3[I,14],
                             ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;

        ANGLECALC (     RAY3[I,7],RAY3[I,8],RAY3[I,9],T[4],T[5],T[6],
                        THETA,PHI,PHIREC);
        DETECTION[I+J].LENGTH:=L;
        DETECTION[I+J].ATHETA:=ETHETA/L;
        DETECTION[I+J].APHI:=EPHI/L;
        DETECTION[I+J].PHASE:=FASE;
        DETECTION[I+J].THETA:=-THETA;
        DETECTION[I+J].PHI:=PHIREC;
    END;


    J:=J+AANTAL3;
    FOR I:=1 TO AANTAL4 DO
    BEGIN
        ANGLECALC (     T[1],T[2],T[3],RAY4[I,1],RAY4[I,2],RAY4[I,3],
                        THETA,PHI,X);
        STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
        ETHETAUIT:=0;
        EPHIUIT:=0;
        L:=   SQRT(SQR(T[1]-RAY4[I,1])+SQR(T[2]-RAY4[I,2])+
                    SQR(T[3]-RAY4[I,3]))+
              SQRT(SQR(RAY4[I,1]-RAY4[I,4])+SQR(RAY4[I,2]-RAY4[I,5])+
                    SQR(RAY4[I,3]-RAY4[I,6]))+
              SQRT(SQR(RAY4[I,4]-RAY4[I,7])+SQR(RAY4[I,5]-RAY4[I,8])+
                    SQR(RAY4[I,6]-RAY4[I,9]))+
              SQRT(SQR(RAY4[I,7]-RAY4[I,10])+SQR(RAY4[I,8]-RAY4[I,11])+
                    SQR(RAY4[I,9]-RAY4[I,12]))+
              SQRT(SQR(T[4]-RAY4[I,10])+SQR(T[5]-RAY4[I,11])+
                    SQR(T[6]-RAY4[I,12]));
        PLANEREFLECTION (   PL4[I,1],T[1],T[2],T[3],
                            RAY4[I,1],RAY4[I,2],RAY4[I,3],
                            RAY4[I,4],RAY4[I,5],RAY4[I,6],
                            ETHETA,EPHI, RAY4[I,13],
                            ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        PLANEREFLECTION (   PL4[I,2],RAY4[I,1],RAY4[I,2],RAY4[I,3],
                            RAY4[I,4],RAY4[I,5],RAY4[I,6],
                            RAY4[I,7],RAY4[I,8],RAY4[I,9],
                            ETHETA,EPHI,RAY4[I,15],
                            ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        PLANEREFLECTION (   PL4[I,3],RAY4[I,4],RAY4[I,5],RAY4[I,6],
                            RAY4[I,7],RAY4[I,8],RAY4[I,9],
                            RAY4[I,10],RAY4[I,11],RAY4[I,12],
                            ETHETA,EPHI,RAY4[I,17],
                            ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        PLANEREFLECTION (   PL4[I,4],RAY4[I,7],RAY4[I,8],RAY4[I,9],
                            RAY4[I,10],RAY4[I,11],RAY4[I,12],
                            T[4],T[5],T[6],
                            ETHETA,EPHI,RAY4[I,19],
                            ETHETAUIT,EPHIUIT);
        ETHETA:=ETHETAUIT;
        EPHI:=EPHIUIT;
        ANGLECALC (     RAY4[I,10],RAY4[I,11],RAY4[I,12],T[4],T[5],T[6],
                        THETA,PHI,PHIREC);
        DETECTION[I+J].LENGTH:=L;
        DETECTION[I+J].ATHETA:=ETHETA/L;
        DETECTION[I+J].APHI:=EPHI/L;
        DETECTION[I+J].PHASE:=FASE;
        DETECTION[I+J].THETA:=-THETA;
        DETECTION[I+J].PHI:=PHIREC;
    END;
END;



PROCEDURE ARRIVAL ( UIT             : STRING;
                    T               : ARRAY6;
                    RAYA            : ARRAY1_5;
                    RAYB            : ARRAY2_10;
                    RAYC            : ARRAY3_15;
                    RAYD            : ARRAY4_20;
                    EXIST           : BOOLEAN;
                    A1,A2,A3,A4     : INTEGER);

VAR  I,J                    : INTEGER;
     L,THETAT,PHIT,
     THETAR,PHIR,PHIREC,
     ETHETA,EPHI,FASE       : REAL;
     ARR                    : TEXT;
```

```
BEGIN
      ASSIGN (ARR,UIT);REWRITE(ARR);
      L:=SQRT(SQR(T[1]-T[4])+SQR(T[2]-T[5])+SQR(T[3]-T[6]));
      IF EXIST=TRUE THEN
      BEGIN
           ANGLECALC (T[1],T[2],T[3],T[4],T[5],T[6],THETAT,PHIT,PHIREC);
           STRALINGSDIAGRAMHUYGENS (THETAT,PHIT,ETHETA,EPHI,FASE);
           WRITELN(ARR,10*L/3:12,' ',ETHETA/L,' ',-THETAT,' ',PHIREC);
      END;
      FOR I:=1 TO A1 DO
      BEGIN
           ANGLECALC (      T[1],T[2],T[3],RAYA[I,1],RAYA[I,2],RAYA[I,3],
                            THETAT,PHIT,PHIREC);
           STRALINGSDIAGRAMHUYGENS (THETAT,PHIT,ETHETA,EPHI,FASE);
           ANGLECALC (      RAYA[I,1],RAYA[I,2],RAYA[I,3],T[4],T[5],T[6],
                            THETAR,PHIR,PHIREC);
           L:=    SQRT(SQR(T[1]-RAYA[I,1])+SQR(T[2]-RAYA[I,2])+
                        SQR(T[3]-RAYA[I,3]));
           L:=    L+SQRT(SQR(T[4]-RAYA[I,1])+SQR(T[5]-RAYA[I,2])+
                        SQR(T[6]-RAY1[I,3]));
           WRITELN (ARR,    10*L/3:12,' ',ETHETA*RAYA[I,4]/L,' ',-THETAR,' ',PHIREC);
      END;
      FOR I:=1 TO A2 DO
      BEGIN
           ANGLECALC (      T[1],T[2],T[3],RAYB[I,1],RAYB[I,2],RAYB[I,3],
                            THETAT,PHIT,PHIREC);
           STRALINGSDIAGRAMHUYGENS (THETAT,PHIT,ETHETA,EPHI,FASE);
           ANGLECALC (      RAYB[I,4],RAYB[I,5],RAYB[I,6],T[4],T[5],T[6],
                            THETAR,PHIR,PHIREC);
           L:=    SQRT(SQR(T[1]-RAYB[I,1])+SQR(T[2]-RAYB[I,2])+
                        SQR(T[3]-RAYB[I,3]));
           L:=    L+SQRT(SQR(RAYB[I,1]-RAYB[I,4])+SQR(RAYB[I,2]-RAYB[I,5])+
                        SQR(RAYB[I,3]-RAYB[I,6]));
           L:=    L+SQRT(SQR(T[4]-RAYB[I,4])+SQR(T[5]-RAYB[I,5])+
                        SQR(T[6]-RAYB[I,6]));
           WRITELN (ARR,    10*L/3:12,' ',ETHETA*RAYB[I,7]*RAYB[I,9]/L,' ',
                            -THETAR,' ',PHIREC);
      END;
      FOR I:=1 TO A3 DO
      BEGIN
           ANGLECALC (      T[1],T[2],T[3],RAYC[I,1],RAYC[I,2],RAYC[I,3],
                            THETAT,PHIT,PHIREC);
           STRALINGSDIAGRAMHUYGENS (THETAT,PHIT,ETHETA,EPHI,FASE);
           ANGLECALC (      RAYC[I,7],RAYC[I,8],RAYC[I,9],T[4],T[5],T[6],
                            THETAR,PHIR,PHIREC);
           L:=    SQRT(SQR(T[1]-RAYC[I,1])+SQR(T[2]-RAYC[I,2])+
                        SQR(T[3]-RAYC[I,3]));
                  L:=L+SQRT(SQR(RAYC[I,1]-RAYC[I,4])+SQR(RAYC[I,2]-RAYC[I,5])+
                        SQR(RAYC[I,3]-RAYC[I,6]));
                  L:=L+SQRT(SQR(RAYC[I,4]-RAYC[I,7])+SQR(RAYC[I,5]-RAYC[I,8])+
                        SQR(RAYC[I,6]-RAYC[I,9]));
                  L:=L+SQRT(SQR(T[4]-RAYC[I,7])+SQR(T[5]-RAYC[I,8])+
                        SQR(T[6]-RAYC[I,9]));
           WRITELN (ARR,    10*L/3:12,' ',ETHETA*RAYC[I,10]*RAYC[I,12]*
                            RAYC[I,14]/L,' ',-THETAR,' ',PHIREC);
      END;
```

```
        FOR I:=1 TO A4 DO
        BEGIN
            ANGLECALC (      T[1],T[2],T[3],RAYD[I,1],RAYD[I,2],RAYD[I,3],
                             THETAT,PHIT,PHIREC);
            STRALINGSDIAGRAMHUYGENS (THETAT,PHIT,ETHETA,EPHI,FASE);
            ANGLECALC (      RAYD[I,10],RAYD[I,11],RAYD[I,12],T[4],T[5],T[6],
                             THETAR,PHIR,PHIREC);
            L:=    SQRT(SQR(T[1]-RAYD[I,1])+SQR(T[2]-RAYD[I,2])+
                        SQR(T[3]-RAYD[I,3]));
            L:=    L+SQRT(SQR(RAYD[I,1]-RAYD[I,4])+SQR(RAYD[I,2]-RAYD[I,5])+
                        SQR(RAYD[I,3]-RAYD[I,6]));
            L:=    L+SQRT(SQR(RAYD[I,4]-RAYD[I,7])+SQR(RAYD[I,5]-RAYD[I,8])+
                        SQR(RAYD[I,6]-RAYD[I,9]));
            L:=    L+SQRT(SQR(RAYD[I,10]-RAYD[I,7])+SQR(RAYD[I,11]-RAYD[I,8])+
                        SQR(RAYD[I,12]-RAYD[I,9]));
            L:=    L+SQRT(SQR(T[4]-RAYD[I,10])+SQR(T[5]-RAYD[I,11])+
                        SQR(T[6]-RAYD[I,12]));
            WRITELN (ARR,    10*L/3:12,' ',ETHETA*RAYD[I,13]*RAYD[I,15]*
                             RAYD[I,17]*RAYD[I,19]/L,' ',-THETAR,' ',PHIREC);
        END;
        CLOSE(ARR);
END;



PROCEDURE ARRIVAL2 (     TOTAL        : INTEGER;
                         DETECTION : DETECTARR;
                         UITVOER      : STRING);

VAR   I      : INTEGER;

BEGIN
        ASSIGN (UIT,UITVOER); REWRITE (UIT);
        FOR I:=1 TO TOTAL DO
        BEGIN
            WITH DETECTION[I] DO
            BEGIN
                IF ((ATHETA<>0) OR (APHI<>0)) THEN
                BEGIN
                    WRITELN (UIT,     LENGTH,' ',ATHETA,' ',APHI,' ',
                                      PHASE,' ',THETA,' ',PHI);
                END;
            END;
        END;
        CLOSE (UIT);
END;
```

```
{MAIN PROGRAM}


BEGIN
        WRITELN ('GEEF DE INPUTFILE');
        READLN(INVOER);
        INVOER:=CONCAT('C:\TP50\MARC\WORK\',INVOER);
        WRITELN ('GEEF DE OUTPUTFILE');
        READLN (UITVOER);
        UITVOER:=CONCAT('C:\TP50\MARC\WORK\',UITVOER);
        ROOMCONF (CONF,T,AANTAL,INVOER);
        DETECTGRAPH (GD,GM);
        INITGRAPH (GD,GM,'C:\TP50');
        ROOMDRAW (CONF,T,AANTAL);

        DIRECT (CONF,T,AANTAL,EXIST0);
        REFLECTION1 (CONF,T,AANTAL,RAY1,AANTAL1,PL1);
        REFLECTION2 (CONF,T,AANTAL,RAY2,AANTAL2,PL2);
        REFLECTION3 (CONF,T,AANTAL,RAY3,AANTAL3,PL3);
        REFLECTION4 (CONF,T,AANTAL,RAY4,AANTAL4,PL4);

        ARRIVAL (  UITVOER,T,RAY1,RAY2,RAY3,RAY4,EXIST0,
                   AANTAL1,AANTAL2,AANTAL3,AANTAL4);

        POLARISATIE (   EXIST0,T,RAY1,RAY2,RAY3,RAY4,
                        AANTAL1,AANTAL2,AANTAL3,AANTAL4,
                        PL1,PL2,PL3,PL4,DETECTREC,TOTAL);

        ARRIVAL2 (TOTAL,DETECTREC,UITVOER);

        SETTEXTSTYLE (0,0,2);
        OUTTEXTXY (0,0,'EINDE');
        WHILE NOT KEYPRESSED DO;
        CLOSEGRAPH;
END.
```

G.4.2. POLPOWER.

```
PROGRAM POLARISATION_POWER_DELAY_PROFILE_WITH_STATISTICS (INPUT,OUTPUT);

{$M 32000,0,655360}

TYPE DETECTRECORD  = RECORD
           LENGTH,
           ATHETA,APHI,
           PHASE,
           THETA,PHI   : REAL;
     END;
     DETECTARR     = ARRAY [1..200] OF DETECTRECORD;
     ARRAY30       = ARRAY[1..30] OF INTEGER;
     ARRAY999      = ARRAY[0..999] OF REAL;


VAR  DETECTION     : DETECTARR;
     PDP,PDPN      : ARRAY999;
     TAUP          : REAL;
     TOTAL         : INTEGER;
     INVOER,
     UITVOER,IN1   : STRING;
     AMD,ADS,GAIN  : REAL;



PROCEDURE STRALINGRECEIVER ( INVOER          : STRING;
                             VAR TOTAL       : INTEGER;
                             VAR DETECTION   : DETECTARR);

VAR  INV  : TEXT;
     I    : INTEGER;

BEGIN
     ASSIGN (INV,INVOER);
     RESET (INV);
     TOTAL:=0;
     I:=1;
     WHILE NOT EOF(INV) DO
     BEGIN
          WITH DETECTION[I] DO
          BEGIN
                TOTAL:=TOTAL+1;
                READLN (INV,LENGTH,ATHETA,APHI,PHASE,THETA,PHI);
                I:=I+1;
          END;
     END;
     CLOSE (INV);
END;
```

```
PROCEDURE STRALINGSDIAGRAMHUYGENS (  THETA,PHI              : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

BEGIN
      THETA:=PI/2-ABS(THETA);
      ETHETA:=SQR(SIN(THETA));
      EPHI:=0;
      FASE:=0;
END;



PROCEDURE STRALINGSDIAGRAMHUYGENSCIRC (  THETA,PHI              : REAL;
                                         VAR ETHETA,EPHI,FASE  : REAL);

BEGIN
      THETA:=PI/2-ABS(THETA);
      ETHETA:=-SQR(SIN(THETA));
      EPHI:=SQR(SIN(PHI));
      FASE:=-PI/2;
END;



PROCEDURE STRALINGSDIAGRAMCORRHL (   THETA,PHI              : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

VAR   N,I          : INTEGER;
      TH,GAIN      : REAL;

BEGIN
      IF THETA>0 THEN
      BEGIN
           EPHI:=0;
           ETHETA:=0;
           FASE:=0;
      END
      ELSE
      BEGIN
           TH:=PI/2+THETA;
           N:=2;
           GAIN:=1;
           FOR I:=1 TO N DO GAIN:=GAIN*COS(TH);
           GAIN:=2*(N+1)*GAIN;
           ETHETA:=-GAIN*COS(PHI);
           EPHI:= +GAIN*SIN(PHI);
           FASE:=0;
      END;
END;
```

```
PROCEDURE STRALINGSDIAGRAMCORRHC (   THETA,PHI               : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

VAR   N,I                          : INTEGER;
      TH,
      PHIX,PHIY,
      GAIN,
      ETHX,ETHY,EPHIX,EPHIY,
      ALPHA1,ALPHA2               : REAL;

BEGIN
      IF THETA>0 THEN
      BEGIN
            EPHI:=0;
            ETHETA:=0;
            FASE:=0;
      END
      ELSE
      BEGIN
            TH:=PI/2+THETA;
            N:=2;
            GAIN:=1;
            FOR I:=1 TO N DO GAIN:=GAIN*COS(TH);
            GAIN:=GAIN*2*(N+1);
            PHIX:=PHI;
            PHIY:=PHI-PI/2;
            ETHX:=-GAIN*COS(PHIX);
            ETHY:=-GAIN*COS(PHIY);
            EPHIX:=+GAIN*SIN(PHIX);
            EPHIY:=+GAIN*SIN(PHIY);
            ETHETA:=SQRT(SQR(ETHX)+SQR(ETHY));
            EPHI:=SQRT(SQR(EPHIX)+SQR(EPHIY));
            IF ETHY=0 THEN ALPHA1:=PI/2 ELSE
            BEGIN
                  ALPHA1:=ARCTAN(ABS(ETHX/ETHY));
                  IF ((ETHX>0) AND (ETHY<0)) THEN ALPHA1:=PI-ALPHA1;
                  IF ((ETHX<0) AND (ETHY<0)) THEN ALPHA1:=PI+ALPHA1;
                  IF ((ETHX<0) AND (ETHY>0)) THEN ALPHA1:=-ALPHA1;
            END;
            IF EPHIY=0 THEN ALPHA2:=PI/2 ELSE
            BEGIN
                  ALPHA2:=ARCTAN(ABS(EPHIX/EPHIY));
                  IF ((EPHIX>0) AND (EPHIY<0)) THEN ALPHA2:=PI-ALPHA2;
                  IF ((EPHIX<0) AND (EPHIY<0)) THEN ALPHA2:=PI+ALPHA2;
                  IF ((EPHIX<0) AND (EPHIY>0)) THEN ALPHA2:=-ALPHA2;
            END;
            FASE:=ALPHA2-ALPHA1;
            IF FASE>PI THEN FASE:=FASE-2*PI;
            IF FASE<-PI THEN FASE:=FASE+2*PI;
      END;
END;
```

```
PROCEDURE SENSITIVITY (  ETHETA,EPHI,FASE      : REAL;
                         VAR RSENSE,LSENSE     : REAL);

VAR  ALPHA            : REAL;
     E1,E2,E3,E4      : REAL;
     AX,AY,A1,A2,
     DELTA            : REAL;

BEGIN
     A1:=0;A2:=0;
     DELTA:=0;
     IF ((ABS(FASE-PI/2)<0.00001) OR (ABS(FASE+PI/2)<0.00001)) THEN ALPHA:=PI/2
     ELSE
     BEGIN
          IF ETHETA=EPHI THEN
          BEGIN
               IF COS(FASE)=0 THEN ALPHA:=PI/2
               ELSE
               ALPHA:=PI/4
          END
          ELSE
          BEGIN
               AX:=SQR(EPHI)-SQR(ETHETA);
               AY:=2*ETHETA*EPHI*COS(FASE);
               ALPHA:=0.5*ARCTAN (AY/AX);
          END;
     END;
     E1:=ETHETA*SIN(ALPHA)+EPHI*COS(ALPHA)*COS(FASE);
     E2:=EPHI*COS(ALPHA)*SIN(FASE);
     E3:=-ETHETA*COS(ALPHA)+EPHI*SIN(ALPHA)*COS(FASE);
     E4:=EPHI*SIN(ALPHA)*SIN(FASE);
     AX:=SQRT(SQR(E1)+SQR(E2));
     AY:=SQRT(SQR(E3)+SQR(E4));

     IF E1<>0 THEN A1:=ARCTAN (ABS(E2/E1))
     ELSE
     BEGIN
          IF E2>=0 THEN A1:=PI/2
          ELSE A1:=-PI/2;
     END;
     IF ((E1<0) AND (E2>=0)) THEN A1:=PI-A1;
     IF ((E1<0) AND (E2<0))  THEN A1:=-PI+A1;
     IF ((E1>0) AND (E2<0)) THEN A1:=-A1;
     IF E3<>0 THEN A2:=ARCTAN (ABS(E4/E3))
     ELSE
     BEGIN
          IF E4>=0 THEN A2:=PI/2
          ELSE A2:=-PI/2;
     END;
     IF ((E3<0) AND (E4>=0)) THEN A2:=PI-A2;
     IF ((E3<0) AND (E4<0))  THEN A2:=-PI+A2;
     IF ((E3>0) AND (E4<0)) THEN A2:=-A2;
```

```
          DELTA:=A2-A1;
          IF (((DELTA>0) AND (DELTA <PI)) OR (DELTA<-PI)) THEN
          BEGIN
               LSENSE:=0.5*(AX+AY);
               RSENSE:=0.5*(AX-AY);
          END
          ELSE
          BEGIN
               RSENSE:=0.5*(AX+AY);
               LSENSE:=0.5*(AX-AY);
          END;
     END;



PROCEDURE RECEIVEDPOWERLIN (      Z                  : INTEGER;
                                  NUMBER             : ARRAY30;
                                  T                  : INTEGER;
                                  TAUP               : REAL;
                                  DETECTION          : DETECTARR;
                                  VAR PDP            : REAL);

VAR   ETHETA,EPHI,FASE            : ARRAY[1..30] OF REAL;
      I,J                         : INTEGER;
      PR,PI,THELP                 : REAL;

BEGIN
      {SENSITIVITY RECEIVER IN DIRECTION THETA,PHI}
      FOR I:=1 TO Z DO
      BEGIN
           STRALINGSDIAGRAMHUYGENS (   DETECTION[NUMBER[I]].THETA,
                                       DETECTION[NUMBER[I]].PHI,
                                       ETHETA[I],EPHI[I],FASE[I]);
           ETHETA[I]:=ETHETA[I]*DETECTION[NUMBER[I]].ATHETA;
           EPHI[I]:=EPHI[I]*DETECTION[NUMBER[I]].APHI;
      END;
      {PULSESHAPING}
      FOR I:=1 TO Z DO
      BEGIN
           THELP:=T/2-10*DETECTION[NUMBER[I]].LENGTH/3;
           ETHETA[I]:=ETHETA[I]*SQRT(COS(PI*THELP/TAUP));
           EPHI[I]:=EPHI[I]*SQRT(COS(PI*THELP/TAUP));
      END;
      PR:=0;
      PI:=0;
      FOR I:=1 TO Z DO
      BEGIN
           PR:=SQRT(SQR(ETHETA[I])+SQR(EPHI[I]))*
                 COS(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005)+PR; {0.005 IS THE WAVE-}
           PI:=SQRT(SQR(ETHETA[I]+SQR(EPHI[I]))*                   {LENGTH IN METERS}
                 SIN(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005)+PI;
      END;
      PDP:=SQR(PR)+SQR(PI);
END;
```

```
PROCEDURE RECEIVEDPOWERELL (      Z               : INTEGER;
                                  NUMBER          : ARRAY30;
                                  T               : INTEGER;
                                  TAUP            : REAL;
                                  DETECTION       : DETECTARR;
                                  VAR PDP         : REAL);


VAR   I,J                              : INTEGER;
      PRR,PRI,PLR,PLI,THELP            : REAL;
      ETHETA,EPHI,FASE,
      SENSER,SENSEL,RIGHT,LEFT : ARRAY[1..30] OF REAL;


BEGIN
      FOR I:=1 TO Z DO
      BEGIN
          STRALINGSDIAGRAMHUYGENSCIRC (      DETECTION[NUMBER[I]].THETA,
                                             DETECTION[NUMBER[I]].PHI,
                                             ETHETA[I],EPHI[I],FASE[I]);

          {PULSESHAPING}
          THELP:=T/2-10*DETECTION[NUMBER[I]].LENGTH/3;
          DETECTION[NUMBER[I]].ATHETA:=DETECTION[NUMBER[I]].ATHETA
                                                  *SQRT(COS(PI*THELP/TAUP));
          DETECTION[NUMBER[I]].APHI:=DETECTION[NUMBER[I]].APHI
                                                  *SQRT(COS(PI*THELP/TAUP));

          SENSITIVITY (     ETHETA[I],EPHI[I],FASE[I],
                            SENSER[I],SENSEL[I]);
          SENSITIVITY (     DETECTION[NUMBER[I]].ATHETA,
                            DETECTION[NUMBER[I]].APHI,
                            DETECTION[NUMBER[I]].PHASE,
                            RIGHT[I],LEFT[I]);
      END;
      PRR:=0;
      PRI:=0;
      PLR:=0;
      PLI:=0;
      FOR I:=1 TO Z DO
      BEGIN
          PRR:=PRR+SENSER[I]*RIGHT[I]*
                      COS(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005);        {0.005 is the}
          PRI:=PRI+SENSER[I]*RIGHT[I]*                                     {wavelength}
                      SIN(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005);
          PLR:=PLR+SENSEL[I]*LEFT[I]*
                      COS(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005);
          PLI:=PLI+SENSEL[I]*LEFT[I]*
                      SIN(2*PI*DETECTION[NUMBER[I]].LENGTH/0.005);
      END;
      PDP:=SQR(PRR)+SQR(PRI)+SQR(PLR)+SQR(PLI);
END;
```

```
PROCEDURE POLPOWER (  DETECTION         : DETECTARR;
                      TOTAL             : INTEGER;
                      TAUP              : REAL;
                      VAR PDP           : ARRAY999);

VAR  NUMBER   : ARRAY30;
     I,J,T,Z  : INTEGER;

BEGIN
     FOR I:=0 TO 999 DO PDP[I]:=0;
     FOR T:=0 TO 999 DO
     BEGIN
          Z:=0;
          FOR J:=1 TO 30 DO NUMBER[J]:=0;
          FOR I:=1 TO TOTAL DO
          BEGIN
               IF (((DETECTION[I].LENGTH*10/3) >= (T/2-TAUP/2)) AND
                    ((DETECTION[I].LENGTH*10/3) <= (T/2+TAUP/2))) THEN
               BEGIN
                    Z:=Z+1;
                    NUMBER[Z]:=I;
               END;
          END;
          IF Z<>0 THEN
          RECEIVEDPOWERLIN (Z,NUMBER,T,TAUP,DETECTION,PDP[T]);
     END;
END;


PROCEDURE STOREPDP (  UITVOER     : STRING;
                      PDP         : ARRAY999);

VAR  I     : INTEGER;
     UIT   : TEXT;


BEGIN
     ASSIGN (UIT,UITVOER); REWRITE (UIT);
     FOR I:=0 TO 999 DO
     BEGIN
          WRITELN (UIT,I/2:10,'  ',PDP[I]);
     END;
     CLOSE (UIT);
END;
```

```
PROCEDURE STATISTICS (  PDP                    : ARRAY999;
                        TAUP                   : REAL;
                        VAR AMD,ADS,GAIN       : REAL;
                        VAR PDPN               : ARRAY999);

VAR  I,J,AFR      : INTEGER;
     OK           : BOOLEAN;
     MAX,HULP     : REAL;

BEGIN
     {ARRIVAL FIRST RAY}
     OK:=FALSE;
     I:=1;
     WHILE OK=FALSE DO
     BEGIN
          IF PDP[I]<>0 THEN
          BEGIN
               AFR:=ROUND(I/2);
               OK:=TRUE;
          END
          ELSE I:=I+1;
     END;
     ADS:=0;
     AMD:=0;
     HULP:=0;
     GAIN:=0;
     MAX:=0;
     FOR I:=0 TO 999 DO
     BEGIN
          IF PDP[I]<>0 THEN
          BEGIN
               IF PDP[I]>MAX THEN MAX:=PDP[I];
               GAIN:=PDP[I]+GAIN;
               AMD:=I*PDP[I]+AMD;
               HULP:=I*PDP[I];
               ADS:=I*HULP+ADS;
          END;
     END;

     AMD:=AMD/(2*GAIN);
     ADS:=ADS/(4*GAIN);
     ADS:=ADS-SQR(AMD);
     AMD:=AMD-AFR;
     GAIN:=GAIN/(2*TAUP/PI);
     {VERSCHUIVEN VAN PDP}
     FOR I:=0 TO 999 DO PDPN[I]:=-100;
     FOR I:=0 TO (999-AFR*2) DO
     BEGIN
          IF PDP[I+AFR*2]<>0 THEN
          BEGIN
               PDPN[I]:=LN(PDP[I+AFR*2]/MAX);
               PDPN[I]:=10*PDPN[I]/LN(10);
          END;
     END;
END;
```

```
{MAIN PROGRAM}


BEGIN
        WRITELN ('GEEF DE INVOERFILE');
        READLN (INVOER);
        INVOER:=CONCAT('C:\TP50\MARC\WORK\',INVOER);
        WRITELN ('GEEF DE UITVOERFILE');
        READLN (UITVOER);
        UITVOER:=CONCAT ('C:\TP50\MARC\WORK\',UITVOER);
        STRALINGRECEIVER (INVOER,TOTAL,DETECTION);
        WRITELN ('GEEF DE PULSLENGTE');
        READLN (TAUP);
        POLPOWER (DETECTION,TOTAL,TAUP,PDP);
        STATISTICS (PDP,TAUP,AMD,ADS,GAIN,PDPN);
        WRITELN ('AVERAGE MEAN DELAY   = ',AMD,' ns');
        WRITELN ('AVERAGE DELAY SPREAD = ',SQRT(ADS),' ns');
        WRITELN ('POWER GAIN           = ',GAIN);
        STOREPDP (UITVOER,PDPN);
END.
```

G.4.3. POWER.

```
PROGRAM POWER_ZONDER_POLARISATIE (INPUT,OUTPUT);

TYPE ARRAY999 = ARRAY[0..999] OF REAL;

VAR   J,TAUP                  : INTEGER;
      TAU,AMD,ADS,GAIN,
      THELP,ALFA,ALFA1,
      THETA,PHI,
      ETHETA,EPHI,FASE        : REAL;
      INVOER,UITVOER,IN1      : STRING;
      INV,UIT                 : TEXT;
      PDP,R,IM,PDPN           : ARRAY999;
      OK                      : BOOLEAN;


PROCEDURE STRALINGSDIAGRAMHUYGENS (  THETA,PHI               : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

BEGIN
      THETA:=PI/2+THETA;
      ETHETA:=-SQR(SIN(THETA));
      EPHI:=0;
      FASE:=0;
END;


PROCEDURE STRALINGSDIAGRAMCORRHL (   THETA,PHI               : REAL;
                                     VAR ETHETA,EPHI,FASE : REAL);

VAR   N,I          : INTEGER;
      TH,GAIN      : REAL;

BEGIN
      IF THETA>0 THEN
      BEGIN
            EPHI:=0;
            ETHETA:=0;
            FASE:=0;
      END
      ELSE
      BEGIN
            TH:=PI/2+THETA;
            N:=2; GAIN:=1;
            FOR I:=1 TO N DO GAIN:=GAIN*COS(TH);
            GAIN:=2*(N+1)*GAIN;
            ETHETA:=-GAIN*COS(PHI);
            EPHI:=+GAIN*SIN(PHI);
            FASE:=0;
      END;
END;
```

```
PROCEDURE STOREPDP (    UITVOER      : STRING;
                        PDP          : ARRAY999);

VAR   I     : INTEGER;
      UIT   : TEXT;

BEGIN
      ASSIGN (UIT,UITVOER); REWRITE (UIT);
      FOR I:=0 TO 999 DO
      BEGIN
            WRITELN (UIT,I/2:10,'  ',PDP[I]);
      END;
      CLOSE (UIT);
END;



PROCEDURE STATISTICS (   PDPX                  : ARRAY999;
                         TAUP                  : REAL;
                         VAR AMD,ADS,GAIN      : REAL;
                         VAR PDPNX             : ARRAY999);

VAR   I,J,AFR      : INTEGER;
      OK           : BOOLEAN;
      MAX,HULP     : REAL;

BEGIN
      {ARRIVAL FIRST RAY}
      OK:=FALSE;
      I:=1;
      WHILE OK=FALSE DO
      BEGIN
            IF PDPX[I]<>0 THEN
            BEGIN
                  AFR:=ROUND(I/2);
                  OK:=TRUE;
            END
            ELSE I:=I+1;
      END;
      ADS:=0;
      AMD:=0;
      HULP:=0;
      GAIN:=0;
      MAX:=0;
      FOR I:=0 TO 999 DO
      BEGIN
            IF PDPX[I]<>0 THEN
            BEGIN
                  IF PDPX[I]>MAX THEN MAX:=PDPX[I];
                  GAIN:=PDPX[I]+GAIN;
                  AMD:=I*PDPX[I]+AMD;
                  HULP:=I*PDPX[I];
                  ADS:=I*HULP+ADS;
            END;
      END;
```

```
        AMD:=AMD/(2*GAIN);
        ADS:=ADS/(4*GAIN);
        ADS:=ADS-SQR(AMD);
        AMD:=AMD-AFR;
        GAIN:=GAIN/(2*TAUP/PI);

        {VERSCHUIVEN VAN PDP}
        FOR I:=0 TO 999 DO PDPNX[I]:=-100;
        FOR I:=0 TO (999-AFR*2) DO
        BEGIN
            IF PDPX[I+AFR*2]<>0 THEN
            BEGIN
                PDPNX[I]:=LN(PDPX[I+AFR*2]/MAX);
                PDPNX[I]:=10*PDPNX[I]/LN(10);
             END;
        END;
END;
```

```
{MAIN PROGRAM}

BEGIN
    FOR J:=0 TO 999 DO
    BEGIN
        PDP[J]:=0;
        R[J]:=0;
        IM[J]:=0;
    END;
    WRITELN ('GEEF DE INVOERFILE');
    READLN (INVOER);
    INVOER:=CONCAT('C:\TP50\MARC\WORK\',INVOER);
    WRITELN ('GEEF DE UITVOERFILE');
    READLN (UITVOER);
    WRITELN ('GEEF DE PULSLENGTE');
    READLN (TAUP);
    UITVOER:=CONCAT('C:\TP50\MARC\WORK\',UITVOER);
    ASSIGN (INV,INVOER); RESET (INV);
    ASSIGN (UIT,UITVOER); REWRITE (UIT);

    WHILE NOT EOF(INV) DO
    BEGIN
        READLN (INV,TAU,ALFA,THETA,PHI);
        STRALINGSDIAGRAMHUYGENS (THETA,PHI,ETHETA,EPHI,FASE);
        OK:=FALSE;
        J:=2*ROUND(TAU-8);
        WHILE ((J<=999) AND (OK=FALSE)) DO
        BEGIN
            IF ((TAU>=J/2-TAUP/2) AND (TAU<=J/2+TAUP/2)) THEN
            BEGIN
                THELP:=J/2-TAU;
                ALFA1:=ALFA*SQRT(COS(PI*THELP/TAUP));
                R[J]:=R[J]+ALFA1*COS(2*PI*TAU*0.005/3E8){0.005 IS THE WAVELENGTH}
                IM[J]:=IM[J]+ALFA1*SIN(2*PI*TAU*0.005/3E8);
            END ELSE IF J/2>TAU+TAUP/2 THEN OK:=TRUE;
            J:=J+1;
        END;
    END;

    FOR J:=0 TO 999 DO
    BEGIN
        PDP[J]:=SQR(R[J])+SQR(IM[J]);
    END;

    STATISTICS(PDP,TAUP,AMD,ADS,GAIN,PDPN);
    WRITELN ('AMD = ',AMD,' ns');
    WRITELN ('ADS = ',SQRT(ADS),' ns');
    WRITELN ('PG  = ',GAIN);
    STOREPDP (UITVOER,PDPN);
END.
```

G.4.4. CUMPOWER.

```
PROGRAM CUMULATIVE_POWER_DELAY_PROFILE (INPUT,OUTPUT);

TYPE ARRAY999 = ARRAY[0..999] OF REAL;

VAR  I                : INTEGER;
     INVOER,
     UITVOER          : STRING;
     INV,BATCH        : TEXT;
     PD,AMD,ADS,
     GAIN,TAUP,X      : REAL;
     PDPTOT,
     PDPN             : ARRAY999;




PROCEDURE STOREPDP (    UITVOER     : STRING;
                        PDP         : ARRAY999);

VAR  I     : INTEGER;
     UIT   : TEXT;

BEGIN
     ASSIGN (UIT,UITVOER); REWRITE (UIT);
     FOR I:=0 TO 999 DO
     BEGIN
          WRITELN (UIT,I/2:10,'  ',PDP[I]);
     END;
     CLOSE (UIT);
END;




PROCEDURE STATISTICS (   PDP                  : ARRAY999;
                         TAUP                 : REAL;
                         VAR AMD,ADS,GAIN     : REAL;
                         VAR PDPNX            : ARRAY999);

VAR  I,J,
     AFR        : INTEGER;
     OK         : BOOLEAN;
     MAX,
     HULP       : REAL;
```

```
BEGIN
      {ARRIVAL FIRST RAY}
      OK:=FALSE;
      I:=1;
      WHILE OK=FALSE DO
      BEGIN
            IF PDP[I]<>0 THEN
            BEGIN
                  AFR:=ROUND(I/2);
                  OK:=TRUE;
            END
            ELSE I:=I+1;
      END;
      ADS:=0;
      AMD:=0;
      HULP:=0;
      GAIN:=0;
      MAX:=0;
      FOR I:=0 TO 999 DO
      BEGIN
            IF PDP[I]<>0 THEN
            BEGIN
                  IF PDP[I]>MAX THEN MAX:=PDP[I];
                  GAIN:=PDP[I]+GAIN;
                  AMD:=I*PDP[I]+AMD;
                  HULP:=I*PDP[I];
                  ADS:=I*HULP+ADS;
            END;
      END;
      AMD:=AMD/(2*GAIN);
      ADS:=ADS/(4*GAIN);
      ADS:=ADS-SQR(AMD);
      AMD:=AMD-AFR;
      GAIN:=GAIN/(2*TAUP/PI);

      {VERSCHUIVEN VAN PDP}
      FOR I:=0 TO 999 DO PDPNX[I]:=-100;
      FOR I:=0 TO (999-AFR*2) DO
      BEGIN
            IF PDP[I+AFR*2]<>0 THEN
            BEGIN
                  PDPNX[I]:=LN(PDP[I+AFR*2]/MAX);
                  PDPNX[I]:=10*PDPNX[I]/LN(10);
            END;
      END;
END;
```

```
BEGIN
        WRITELN ('GEEF DE UITVOERFILE');
        READLN (UITVOER);
        WRITELN ('GEEF DE GEBRUIKTE PULSLENGTE');
        READLN (TAUP);
        UITVOER:=CONCAT('C:\TP50\MARC\WORK\',UITVOER);
        ASSIGN (BATCH,'C:\TP50\MARC\WORK\LIST.PDP'); RESET (BATCH);
        FOR I:=0 TO 999 DO PDPTOT[I]:=0;
        WHILE NOT EOF (BATCH) DO
        BEGIN
              READLN (BATCH,INVOER);
              INVOER:=CONCAT('C:\TP50\MARC\WORK\',INVOER);
              ASSIGN (INV,INVOER); RESET (INV);
              FOR I:=0 TO 999 DO
              BEGIN
                     READLN(INV,X,PD);
                     PD:=EXP(PD*LN(10)/10);
                     PDPTOT[I]:=PDPTOT[I]+PD;
              END;
              CLOSE (INV);
        END; CLOSE (BATCH);

        STATISTICS (PDPTOT,TAUP,AMD,ADS,GAIN,PDPN);
        WRITELN ('ADS = ',SQRT(ADS),' ns');
        STOREPDP (UITVOER,PDPN);
END.
```