Development of methods for characterizing plant and stand architectures and for model comparisons

Dissertation

zur Erlangung des Doktorgrades der Fakultät für Forstwissenschaften und
Waldökologie der Georg-August-Universität Göttingen

vorgelegt von **Helge Dzierzon**

geboren in Osterode am Harz

Göttingen 2003

D7

Erster Berichterstatter:	Prof. Dr. Winfried Kurth
Zweiter Berichterstatter:	Prof. Dr. Dr. h.c. Branislav Sloboda
Dritter Berichterstatter:	Prof. Dr. Risto Sievänen
Tag der mündlichen Prüfung:	7.11.2003

Diese Arbeit wurde durch die Niedersächsische Staats- und Universitätsbibliothek, SUB-Göttingen, unter folgender Internetadresse veröffentlicht:

http://webdoc.sub.gwdg.de/diss/2003/dzierzon/

Contents

[*] Dzierzon et al. (2003) slightly changed
[†] Dzierzon et al. (2003) slightly changed

[‡] Dzierzon et al. (2003) slightly changed

List of Figures

List of Tables

1 Introduction

Numerous models for the growth of trees and collectives of trees are existing. Classical approaches like Prodan (1965) or Kramer (1984) use only single variables like height or basal area and their distribution within a stand. Those approaches have the main disadvantage of basically disregarding competition between individuals. Furthermore, they yield implausible values when environmental change occurs. Advanced approaches (Sloboda and Pfreundt (1989), Pfreundt (1988), Pfreundt and Sloboda (1996), Pretzsch (1992a), Nagel et al. (2000), Guericke (2001)) try to emphasize the individual. The reaction to environmental changes and competition is improved.

But before a more intensive view of the state of the art is given, the terms *model*, *reconstruction* and *simulation* have to be clarified. The term *model* is used in various ways in the literature. We will try to be somewhat more precise here. Kurth (1999) defines a model as *an image of reality*. A model itself can be a system of submodels. A reconstruction is also an image of reality, hence a model in the broader sense. However, this term narrows to images with less demand on abstraction and generality. A *reconstruction* always describes a unique situation like the reconstruction of the development of the tree heights within a stand. A *simulation* is the algorithmic implementation of a model/reconstruction on a computer. It is not possible to distinguish sharply between these notions. For instance, implementation limits on the computer can have backward effects to the model or a reconstruction may be part of a more complex model.

To be able to characterize models - especially the complex ones - and/or its parts it is useful to categorize them by their motivation and design. Kurth and Bredemeyer (1996) give two main motivations for building models:

- building models as an instrument in science,
- building models as a planning and decision aid.

In science, models can be an aid for integrating research results, their explanation, their application for test planning purposes and hypothesis testing, and for building up theories (cf. Kurth 1999).

In the second category, models are used as a prediction instrument. Prediction includes a temporal or spatial extrapolation (Kurth 1999). An example is the prediction of reactions of ecosystems to potential climate change. But prediction is also related with the practical, everyday use of models. In forestry for instance a various number of models are used for management purposes.

The design of models can be categorized in different ways. Kurth (1999) gives an overview about the classification of models. One possibility is to differentiate between quantitative and qualitative models. A quantitative model predicts numerical values. Yield tables for instance are purely quantitative. They calculate the amount of stand growth within a time period. Qualitative models do not treat numerical values, or if so, they do it in an unprecise way.

Empirical models are directly based on measured values. Again, yield tables and regressions are good examples. Theoretical approaches are flowing into conceptional models – for instance, thermodynamics is a conceptional approach. The borders between both types of models are fluent. E.g., theoretical approaches are often tested empirically, whereas a linear regression contains the theoretical concept of linearity (cf. Kurth 1999).

Kurth (1994) gives a more textual classification method based on the hierarchy theory of O'Neill et al. (1986). It is a classification by the scale or hierarchy level at which a model operates (Kurth 1999). Kurth (1994) complemented the classification by a second direction which represents the level of structure. E.g., models with a focus on physiological parameters usually do not treat structures like tree segments. The result is a triangle of model classification which can be seen as a model continuum. Figure 1 shows that triangle.

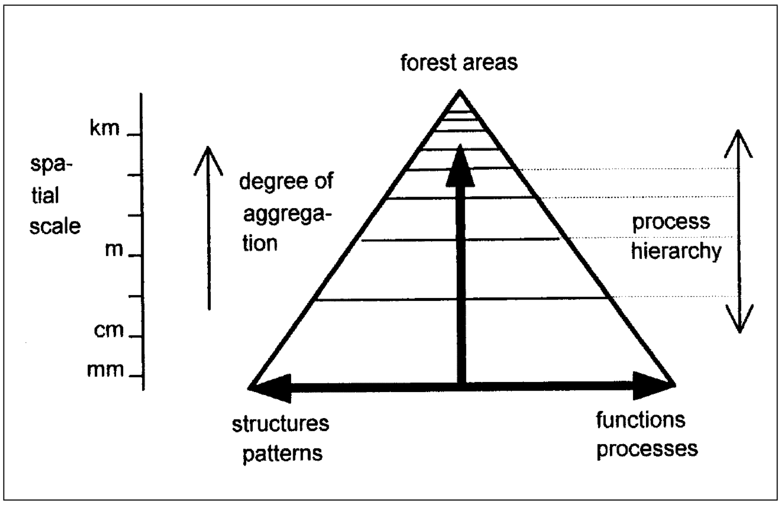


Figure 1: A triangle of plant models (from: Kurth (1999))

This work has a main focus on tree stand models and their description and analysis. Tree stands and their virtual counterparts are analysed and compared by their structure. In general, structure is the relationship between parts of an entity, that is, the relationship between parts of a system. Structure can have numerous shapes. It can be chemical, biological or physical. In our case, structure is seen as tree and stand architecture, unless interaction is meant. Architecture can have influence on interaction but contains no interaction. The term architecture means not only the geometrical structure – Godin (2000) defines plant architecture as follows:

Plant architecture is any individual description based on decomposition of the plant into components, specifying their biological type and/or their shape and/or their location/orientation in space and/or how these components are physically related to each other. Godin (2000) also shows the necessity to develop representations of information to describe architecture. He sees three types of information:

- decomposition information,
- geometrical information and
- topological information.

Architecture distinguishes between the shapes of considered types of objects. The demands addressed to a description of tree structure are different to those regarding entire tree stands. But the necessity to gain information to determine structure does not differ, hence the differentiation of the above types of information is also useful in stand simulation. The neighbourhood relationship, e.g., can be seen as topology (Albert 1999). In stand simulation, the representation of tree crowns is often simplified. Nevertheless even simplified virtual trees have an architecture.

To continue the description of the state of the art, an overview about single tree models is given. Single tree models without an explicit interaction between trees are usually more detailed, either concerning structure or concerning the level of process. The given overview is devoted to the classification of process hierarchy and level of structure.

Detailed models treat morphological units like tree segments as a part of architectural structure. Other models use single tree models as building blocks to model a collection of trees. The characteristics of a single tree control its response to interaction. The results can be emergent structures at stand level. Within the scope of individual based models, it is popular to use aggregated forms of tree architecture. The crowns are often represented by geometrical bodies like irregular pyramids or spheres. Also the stem is seen as an entity and is represented by cylinders or cones. Stand models and emergent structure are discussed later on.

The architectural structure of the stem taper is described by Sloboda (1985), Gaffrey et al. (1998) and Sloboda et al. (1998). They tried to express the shape of the stem taper by curves which result from linear models or spline function. The target was to replace unprecise predictors for the volume of a stem like the form factor by more precise functions with a minimum number of input parameters.

Architectural structure of single trees can be expressed in different ways. The oldest description dates back to Leonardo da Vinci (*Leonardo rule*). He postulated a preservation law for cross sectional area. The sum of cross sectional areas of all shoots emerging from a branching point is equal to the cross sectional area of the mother shoot. Another relationship involving diameters of woody axes in trees was proposed by Chiba (1990, 2000) on the base of the pipe model theory of trees (Shinozaki et al. 1964). He hypothesised that the total weight distal to a position should be directly supported by stem biomass per length at that position.

Architectural structure is expressed in a statical way by McMahon and Kronauer (1976). They argued that a branch follows the deflection laws of a cantilever beam and that the ratio between deflection and length remains constant when the branch grows. The length of the corresponding cantilever is estimated by the mean path length, where a path is the unique connection going in distal direction from a given segment of the tree to a given branch tip supported by that segment. Under specified conditions this can indeed lead to self-similar crown forms.

All the structural laws discussed so far are supposed to hold independently from the position of a segment within a tree. But the position can have influence on the length of tree segments. Length and diameter are usually correlated, hence the position has also influence on the structure formed by diameter patterns. The most popular positional structures are acrotony, basitony and mesotony. They are genetically fixed and determine the shape of a tree to a great degree. But also amphitony and the axis trend are important positional parameters in the architecture of trees (cf. Goebel 1928, Lück et al. 1990).

An important aspect of three-dimensional structure is the structure of space exploitation. One important type of description of three-dimensional tree structure is given by the formalism of turtle geometry (Abelson and diSessa 1982) which is used in growth grammars (cf. Kurth 1999). It is a method for encoding branched structures which are assembled of one-dimensional segments (cf. Kurth 1999). The segments are generated by a virtual *turtle*. The movement of the turtle is controlled by commands. In the scope of growth grammars the commands are the result of an interpretation process of a string which is explained in a more detailed way later on.

The spatial distribution of trees in space can be approximated by three-dimensional grids (voxel spaces). Parameters like the length of segments or their diameter are discretized into voxels. Oppelt et al. (2000), for instance, used a voxel space to investigate the spatial structure of the root systems of the tree species *Grevia flava, Strychnos cocculoides, Vangueria infausta, Strychnos spinosa* from Botswana, Africa.

Fractal analysis is another application of voxel spaces. The spatial structure of trees under different scales of resolution is investigated. The result is the fractal dimension (meaning a possibly non-integral number between 0 and 3). The estimation of the fractal dimension of a natural object can be considered as a way to quantify how intensely the object fills the space in which it is embedded (Mandelbrot 1982, Voss 1988). One method of estimating fractal dimension by measuring the occupancy of a voxel space is called box counting method. Other definitions and methods to determine the fractal dimension are existing, but the box counting method is the most commonly used variant in the case of botanical objects (cf. Oppelt et al. 2000).

According to the model triangle (Kurth 1994) (cf. figure 1), models which treat only structural parameters are at the bottom left corner. They do not have any process information. One example is Kellomäki and Kurttio (1991). They implemented an empirical dynamic crown model for Scots pines (*Pinus sylvestris L.*). It describes the behaviour of individual shooting and is in fact a purely empirical model. Its input are structural parameters like the length of growth units, the angle of branching and others. These parameters are seen as dependent upon the properties of mother shoot, order and age of a daughter shoot.

Similar to Kellomäki and Kurttio (1991) is the approach of Kurth and Anzola Jürgenson (1997). The differences are primarily the species and the implementation: Here we have a model for *Picea abies* Karst. which is also empirical. The main difference to that of Kellomäki and Kurttio is the fact that it is implemented by using a formal language. In this case the *GROGRA* L-system specification (Kurth 1999) is used.

Functional structural tree models (*FSTM*) treat morphological entities of a tree as interacting units, each equipped with its own geometrical, physical and physiological characteristics. *FSTMs* are becoming increasingly popular as research tools in botany, agronomy and forest science. The representation of tree architecture in *FSTMs* requires (at least implicitly) some mathematical concepts for handling branched, multiscaled structures (e.g. list representations, L-systems, graph theory (cf. Godin and Caraglio 1998)). *FSTMs* are models which combine architectural structure and processes. Remembering the triangle of models they are situated at the middle of the bottom line.

Many of the *FSTM* combine structures and functions at the morphological level of growth units in relation to the local light conditions. Kellomäki and Strandman (1995) developed a model for the structural growth of young Scots pine crowns based on light interception of shoots. The light model and interception is due to the work and field data of Oker-Blom and Smolander (1988). The structure is taken from Kellomäki and Kurttio (1991). The aim is to predict timber quality (Kellomäki et al. 1999a,b) and to represent the linkage between stem properties and properties of sawn timber through structural growth (Ikonen et al. in press).

The model *LIGNUM* (Perttunen et al. 1996, 1998) is another example for an *FSTM*. The object oriented model *LIGNUM* treats a tree as a collection of a large number of simple units that correspond to the organs of a tree (Perttunen et al. 1996). *LIGNUM* simulates the interception of photosynthetically-active radiation (PAR) in the tree crown, the processes of photosynthesis and respiration and the allocation of carbon among the tree segments. The light interception of shoots is modelled much like Kellomäki and Strandman (1995). But Kellomäki and Strandman took no carbon allocation processes into account. Only the shoot length depends on light interception. Perttunen et al. tried to explain the carbon allocation processes within the tree. For carbon allocation they used the pipe model of Shinozaki et al. (1964), and new growth is possible if the photosynthetic production exceeds the respiration demands of the foliage, sapwood and roots (Perttunen et al. 1998). The model is implemented for *Pinus sylvestris* L. (Perttunen et al. 1996, 1998), *Acer saccharum* Marsh (Perttunen et al. 2001) and *Pinus banksiana* Lambert (Lo et al. 2001).

The local light environment is also subject of Takenaka (1994). Takenaka approximated the tree by a modular structure of linear stems. The stem is referred to as *BU* (**B**ranch **U**nit). The growth of *BUs* and the creation of new ones is a response to local light conditions. The light model is characterized by internal shading calculation and a standard overcast sky as the light donator (cf. Moon and Spencer 1942).

The compound model *GroCOM* (**Gro**wth grammar based **COM**pound model) seizes the ideas of *LIGNUM*. Carbon balance and diameter growth are handled in the same way as in *LIGNUM* (cf. Kurth 1999). But the morphological rules are implemented by using L-systems where the carbon balance is calculated directly by a reimplementation of the *LIGNUM* kernel. *GroCOM* offers the possibility to use either the light model of Takenaka (1994) or that of *LIGNUM* in the version of Perttunen et al. (1996).

The light acclimation of *Alnus glutinosa* L. and the competition effects are imaged by Eschenbach (2000) as an application of the model *ALMIS*. *ALMIS* is an object oriented model. It discretizes incoming light with the help of a voxel space. The amount of incoming light is determined empirically. Contrary to the mentioned above models, carbon allocation is modelled without using the pipe model (Shinozaki et al. 1964). Carbon is allocated by *directed* diffusion.

A more comprehensive approach is that of Lanwert et al. (1998), Anzola Jürgenson (2002). They investigated the possibilities which may arise by merging models and with this aim they tried to build up a complex data interface structure between different simulation softwares to enhance the possibilities of simulations. They suggested to combine process oriented and structural models of the *AMAP* family (*MIR*, *MUSC*, *AMAPsim*) with the water flow model *HYDRA* (Früh and Kurth 1999). The surface-environment relations are proposed to be simulated by *HYDRO*. For a more intensive view I refer to Lanwert et al. (1998), Dauzat et al. (2001), Anzola Jürgenson (2002).

An overview of the scope of single tree models is given, now the spatial scale becomes larger. Trees form collectives, which are called tree stands, and which are also imaged by studious scientists. The classical approaches of tree stand models have been already mentioned in the beginning. They arose out of the requirement to estimate the amount of yield and its development within a forest planning period. The models were made for practical use and have an economical focus. The use in practice is one reason why they are held as simple as possible. Approaching the task in such a simplified way was possible because the considered tree stands had a simple structure. Planted by foresters, trees within a stand were and still often are of the same species and age. On the base of such tree stands the yield tables developed by Kramer (1984), Prodan (1965), Assmann (1961), Schober (1975) and others had been highly precise in the past.

Changes in the forest management and in global climate - whatever direction it will take - destroyed the precision of such approaches. In order to solve this problem, building up models with competition and interaction between trees of different ages and species is seen to be necessary (cf. Pretzsch 1997). The reason is that the classical approaches handle no information about mechanisms. The processes which are taking place in trees or interactions with their effect on stand structure are unregarded.

One approach for the solution of such problems is the individual based modelling of tree stands. The stand is decomposed into interacting individuals. Such stand models can also handle process or structural information. The processes have different scales. Usually ecophysiological processes are considered which are added to stand models.

Tree stand models which do not treat ecophysiological processes usually image trees by simplified tree architecture. The design of the single tree submodels was already mentioned above. The interaction of the submodels causes emergent architectural structure at stand level. The different forms of tree stand structure shall be reviewed here.

Tree stands consist of single trees. But stand structure can be expressed by parameters which aggregate the single tree information like classical approaches do. Distributions of diameters or basal areas are typical structures (cf. Schober 1975).

The decomposition of a stand into trees is represented in a somewhat more explicit way by indices which characterize the point position distributions. The target of those indices is similar: The index shall express whether the distribution is random or not (Hopkins 1954, Clark and Evans 1951). Those approaches neglect attributes of the individuals. Gadow et al. (1998) see two other parameters beside point distributions: the species mingling and differentiation. In Gadow et al. (1998) the neighbourhood pattern is proposed as a means to describe stand structure. It takes the species and dimension of the next neighbours into account. Füldner (1995) also proposed indices for structure description: the structural quadruple to estimate the aggregation, segregation and dimensional differentiation of a stand.

An index describes stand structure in a very aggregated way. Degenhardt (1999), Degenhardt and Pommerening (2000) consequently investigated the possibilities to reproduce the structure out of the indices. The reproduced structures have the same characteristics according to the index but cannot reconstruct explicitly the positions of the trees.

Positional distribution can be seen as part of the topological structure. Geometrical structure can be expressed by the extension of tree crowns. Duchiron (2003), Stüber and Staupendahl (1996), Nagel et al. (2000) investigated the influence of crown overlaps on structure and growth of single trees. Nagel et al. (2000) proposes the *C66* index to depict the light regime under a roof and the effects on the growth of regeneration.

Other approaches for describing the aspect of horizontal structure were given by Degenhardt (1999) as well as Degenhardt and Pommerening (2000). They discussed the possibilities which result from Gibbs processes as a description task.

A step forward to stand models with interaction is the stand model *FOREST* (Ek and Monserud (1974) and Monserud (1997)). The authors consider the influences of mixed stands and competition on a growth model. This model does not treat the architecture of single trees, but it contains some stand architecture information like the position within a sample plot. The diameter and height of single trees are simulated using yield tables. The approach contains also a mortality ratio which is calculated from the diameter and the predicted diameter increment. *FOREST* is fitted to *Pinus resinosa* Ait. in Wisconsin, *Picea mariana* Mill. in northern Ontario and *Populus* 'Tristis' in northern Wisconsin. Sterba (1990) fitted the model to a German pine spruce mixed stand (*Pinus sylvestris L.*, *Picea abies L.*).

The forest growth prediction software *SILVA* introduced the individual based models at stand level. The structure of the stand is seen in a simplified way. The crowns of individuals are represented by simple geometric bodies like irregular triangles. Pretzsch (1990a,b, 1992a,b, 2001), Kahn and Pretzsch (1997) adapted the model to spruce and beech stands in southern Germany. Interaction is taking place by distance sensitivities. The response of the crowns has impact on the tree structure expressed in diameter and height. The response does also determine the stand structure.

Nagel et al. (2000) developed an approach similar to *SILVA*. Their tool *BWIN* also uses distance sensitivities and simple tree architecture. But the *C66* index is used for simulating spruce regeneration.

Kurth (1999), Kurth and Sloboda (1999a,b) give simple examples for simulating tree stand architecture using a rule based language. They principally showed the advantages of such an approach and reimplemented the structural spruce crown model of Pretzsch (1992a,b). The approach is simplified to two-dimensional space.

The amount of information which is needed to extend the description of processes to stand level results in a problem of calculation time of the simulation runs. Nevertheless many models are existing at stand level which include process information. However, the architecture of trees is often simplified like in purely structural models. Kurth (1999) calls such models *pseudo FSTMs* because real *FSTMs* are describing morphological units. A simplification aggregates morphological units into virtual units at a higher level, which have no botanical counterpart.

As in single tree models local light conditions are also related to processes which are taken into account at stand level. Ottorini (1991) for instance adds a light model to his tree stand image. But the trees are simplified to stylized crown shapes. The light income and the adapted carbon acquisition and allocation are also part of *MADEIRA* (List et al. 1994). The architecture of the crown is simplified by a voxelspace to reduce computation time.

Light conditions, but also environmental parameters like temperature and humidity and the response of the tree are investigated by Grote (2002a,b), Grote and Pretzsch (2002). The authors added some processes which are taking place to the model *SILVA* and modelled biomass growth of tree stands. The crown of the tree is divided into levels of different height. Within those levels the physiological information is aggregated.

Too complex to describe in detail is the spruce tree stand model of Pfreundt (1988), Pfreundt and Sloboda (1996). In the simulation an enormous amount of functional information is regarded. The stand is seen as an open system. The exchange parameters are the silvicultural treatment, imission of nutritients and water. Within the stand the needle area distribution along the stem is calculated. Stand structure is seen as needle distribution. For more details I refer to Pfreundt (1988).

Roots and the uptake of nutrients are part of the model *TRAGIC* (Hauhs et al. 1995, 2003). The development of roots within the soil is modelled by a random walk within the soil followed by diffusion-limited aggregation. However, the authors see the simulator also as an editor for the forester, analogous to a flight simulator.

Middelhoff and Breckling (1998) also took the roots of trees into account. The effects of nutrient uptake by the roots on the structural growth of trees are modelled in a rather detailed way. The tree architecture is rather simple. The structure is represented by a *shoot* which contains four *plant parts* (leaves, branches, stem and root stock). These parts contain the nutrients. The *shoot* has no explicitly modelled structure but a certain structural aspect is contained in biomass-surface relations (Middelhoff and Breckling 1998).

The model *TREEDYN* (Bossel 1996) is an example for an aggregated process oriented tree stand model. It simulates on the stand level physiological processes like carbon fluxes. Sonntag (1998) used *TREEDYN* to estimate the influences of the climate changes on tree stands.

As already mentioned, our work puts emphasis on tree stand models. A modelling process should have four steps:

- analysis of reality,
- creation of virtual reality (model),
- simulation,
- analysis of simulation results (validation).

Out of that structure of creating and validating models two main target scopes of this work arise. First the analysis of real or virtual architectural structure. The analysis can be applied either to static or dynamic structures. The second target is the implementation of models on the computer.

The result of a simulation run with a tree or tree stand model is usually a large 3D structure with numerous components, each of them characterized by many values of variables. If one wants to compare such a virtual tree structure with a real tree or with the result of another model, some tools for analysing tree representations should be at hand (Dzierzon et al. 2003).

The analysis of real or virtual structures is necessary to create a model or to assess the results of simulation runs. The assessment of simulation runs requires the comparison between reality and virtual reality. The analysis of architecture also requires the discretisation of the structures. Without a special discretisation tool, various different software systems would be necessary for the assessment. Up to now only the French tool *AMAPmod* (Godin et al. 1998) is existing which is specialized on analysing tree architectures for model creation purposes. The architecture is represented in a multiscaled data format (Godin and Caraglio 1998).

To give options for the analysis and discretisation of tree stand architecture and to show how architectural information can be informatically treated, the tool *GroDisc* was created (**Gro**Gra-related **Disc**retisation tool) (Dzierzon et al. 2003). The data structure of *GroDisc* has some similarity to that of Geographic Information Systems (*GIS*). The algorithms for the analysis were designed using the Standard Template Library *STL* (cf. Breymann 1998). A focus during implementation was the possibility of extending the software. It is easy for a programmer to extend the discretisation module. *GroDisc* additionally integrates *GROGRA*, *LIGNUM* (cf. Dzierzon et al. 2003), *GroFant* (Kurth 1999) and the statistical software *R*. A data interface to *AMAPmod* is also existing. In this work it is shown how such interfaces are able to enhance the analysis tree and stand structure. *GroDisc* has a graphical user interface which is implemented using the *C++* library *Qt* version 3.1. *GroDisc* is programmed using the object oriented paradigm. The language is *C++*.

Scientific results should be reproducible at every point in the world and by every scientist. That requirement does also hold for models and their simulation with scientific purpose. Models with practical purpose should be reproducible, too, especially if they are published with scientific ambitions. The implementation of models on the computer lets arise some questions regarding the transparency of models. The reason is that models are usually implemented using the object oriented or procedural programming paradigm. In principle those paradigms are useful in many cases as far as the model structure is concerned (cf. Breckling 1997). However, both paradigms do not offer the possibility to separate sharply between the model and the technical necessities (cf. Kurth 1999). It seems that the rule based paradigm in the shape of string rewriting systems is a good possibility for separating the model and technical parts in a very strict way.

To give alternatives to procedural or object oriented paradigm, stand models are created and implemented on the computer by using L-systems as an example for a rule based language. The L-systems are already used to implement *FSTMs* at tree level, but simplified tree stand models are up to now implemented only within the procedural or object oriented paradigm.

One of the models introduced in this work is an empirical stand growth reconstruction for Scots pine (*Pinus sylvestris* L.). It is abutted on the model of Pretzsch (1992a), hence the crown of individuals is represented by an irregular pyramid with eight fixed corners.

The interaction between trees is simulated by distance sensitivity. The collection of data has been done by Duchiron (2003) in 1995 and has been repeated in 2001 by the author.

All model approaches like Pfreundt (1988), Pretzsch (1992a), Nagel et al. (2000), Guericke (2001) have the disadvantage of having a large demand of empirical data collection. The question arose if aggregation of *FSTMs* could be a base for stand modelling and thus help to reduce the amount of expensive empirical work. Due to the cooperation with the Finnish forest research institute (*METLA*) the model *LIGNUM* adapted for Scots pine was chosen for that purpose. A *LIGNUM* tree is grown under a specified light regime as it is ruled by a gap. Those trees are grown to an age of eleven years. The tree is aggregated into an irregular pyramid as in the case where the real stand was measured. The behaviour of this simplified model is investigated, particularly concerning growth dependent on distance to the edge of the gap.

Validation of models is a quite ambitious target. Validation has to be done in several steps. As already mentioned in the first step, the classification of a model and its parts because validation implies analysis. And for analysis purposes it is necessary to know what kind of information the model handles, e.g. if it is quantitative or if it has qualitative parts. The second step is comparison. The results of the model *LIGNUM* are compared with real Scots pines to see if the results are realistic (cf. Dzierzon et al. 2003).

2 Lindenmayer systems

One main target of this work is to model tree stand growth using L-systems as an alternative to procedural or object oriented programming languages. Hence this chapter pays attention to the concept of L-systems as a rule based language and the differences to the object oriented and procedural paradigm. The models which are introduced within this work are implemented by an extension of L-systems. These so-called growth grammars (Kurth and Sloboda 1999a,b, Kurth 2002) are interpreted by the software *GROGRA* (Kurth 1994, 1999). Thus the language specifications of the growth grammars and the interpreter are characterized.

2.1 Programming languages in general

A language in general provides the transportation and processing of information. The need of information presumes uncertainty. The elimination of uncertainty means to solve problems. The reduction of uncertainty depends on the possibilities the solution of a problem has. In computer science the number of possibilities is defined as information. The amount is even calculable. The logarithm on the base of two of the number of possibilities is the amount of information and is measured in *bit* (cf. equation 1).

$$Information = \log_2 \; number\ of\ possibilities \tag{1}$$

Tree stand modelling and algorithms for analysing architecture are information problems. The processing of information is done with the aid of computers. Computers have to be programmed hence a programming language is needed. Every programming language follows a programming paradigm. Within the scope of tree and tree stand modelling mainly three paradigms appear (cf. Kurth 2002):

- the procedural paradigm (PP),
- the object oriented paradigm (OOP),
- the rule based paradigm (RBP).

Other paradigms like the logical (*PROLOG*) or functional (*Lisp*) are existing, but they do not have much importance in ecological modelling.

The PP is the oldest. Languages like *C*, *Pascal* or *Fortran* are examples. A program based on the PP is seen as a sequence of commands (Kurth 2002). Statements like *if*, *do* or *while* modify the linear order of commands. The commands can have nested structure (Kurth 2002) which follows the concept of subroutines. Procedural languages are very useful to solve large and complex numerical problems. Mathematicians often use purposely *Fortran* because of large numerical libraries which the language provides.

The object oriented paradigm (OOP) inherits the PP. The main idea of OOP are encapsulated objects which are created, modified and deleted during execution. The objects are described by abstract classes. Inheritance allows to create an abstract class hierarchy. The objects can represent “real world” entities (Kurth 2002). In ecological modelling the OOP is quite popular (cf. Saarenmaa 1992, Salminen et al. 1994, Breckling 1997, Pertunen et al. 1996, 1998, Eschenbach 2000). Some OOP languages like *SIMULA*, *C++* or *Object Pascal* still allow procedural programming. Other languages like *SMALLTALK* are stricter. They do not allow procedural parts.

Lindenmayer systems (L-systems, after Lindenmayer (1968), see Prusinkiewicz and Lindenmayer (1990)) are examples for the rule based paradigm. Extended versions of L-systems are existing. The sensitive and non-sensitive growth grammars of Kurth (1994), Kurth and Sloboda (1999a,b) are such extensions of L-systems. Here, a program consists of several rules of the form *left-hand side* → *right-hand side*, some of which are applicable in a certain situation (Kurth 2002). If a rule is applied the left-hand side is replaced by the right-hand side. Rewriting processes emerge if the application of the rules is repeated.

Mixtures between the paradigms are existing. Prusinkiewicz et al. (1999) developed the language *L* which is such a combination. The language *L* allows to combine *C* and L-system code which makes it a mixture between PP and RBP.

2.2 The concept of L-systems

The central concept of L-systems is that of rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of *rewriting rules* or *productions* (Prusinkiewicz and Lindenmayer 1990). The idea of using rewriting systems as a description of plant growth is based on the observation that many plants display some degree of self similarity. For instance the shapes of *Brassicaceae* often seem to be self-similar. Self similarity can be created by rewriting processes. The only thing one needs is information about an *initior*, a *generator* and the replacement rules.

Based on the idea of using character strings for rewriting systems, Lindenmayer (1968) introduced that formalism in a biological context. He described the interaction of cells with string rewriting systems. Kurth (1994, 1999), Kurth and Sloboda (1999a,b) extended the L-systems by global sensitivity and interpretive rules (cf. figure 3).

Figure 2 and listing 1 show an example to depict the botanical significance of an L-system. Here the "bud" with the symbol a is created in the first step. In the second step the resulting string gets more complex. The a is replaced — like the second rule denotes — by $F[RU45b]a$. The first shoot is created and a second bud depicted by the symbol b appeared. The brackets let a branch emerge after a rotation with an angle of 45 degrees. In the third step the resulting string is $F[RU45b]F[RU45b]a$ because the second and third rules have to be applied. A second branch is created now. By repeating the rewriting process, the tree becomes larger and more buds and shoots emerge.

Listing 1: An L-system example in the syntax used by *GROGRA*.

```
* → a ,
a → F [ RU(45) b ] a ,
b → F b ,
```

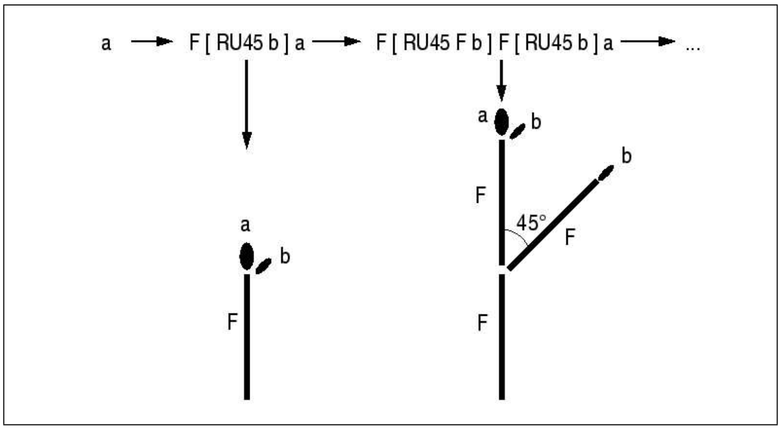


Figure 2: Example of an L-system, indicating the botanical significance of an L-system (from: Kurth (1999)).

2.3 Language specification of *GROGRA* L-systems

The detailed description of the whole syntax and semantics is also explained in Kurth (1994) and Kurth (1999). The L-system specification of *GROGRA* is used for stand simulation purposes in this work. For a better understanding of the rule based computer code, the used keywords, turtle commands and sensitive functions are demonstrated.

2.3.1 Turtle commands

The three-dimensional structure in *GROGRA* is represented by turtle geometry (Abelson and diSessa 1982). The turtle geometry is created by a virtual turtle which executes commands. The commands are specified in the growth grammar and appear in the resulting string. Table 1 shows all commands which are used in this work. The table shows also the meaning of the command and the local or persistent effect of attributes. The most important commands are F and RH, RU, RL. F means “forward” and orders the turtle to create an elementary unit of cylindrical shape which may be part of a tree. If the turtle shall step forward without creating a unit the lower case f is used. RH, RU, RL let the turtle change the direction (cf. figure 9) using rotation vectors. During the creation of a unit attributes are given to the unit. The values of the attributes can also be set by turtle commands. The attributed turtle commands can have local or persistent effect. All commands with an additional lower case l letter have local effect. “Local” means that the value is only valid for the next following F symbol and not persistent for all following symbols. Brackets denote the beginning and the end of branching (cf. listing 1). The repetition operator tells the turtle that she has to repeat the following commands. The scope of repetitions is denoted by “lower than” ($<$) and “greater than” ($>$) symbols. The command T was added for special purposes in stand modelling and is used in the L-systems which are created here. The chapter 2.4 is dedicated to the extensions which are made for the special purposes of this work. Hence the command T is explained there.

2.3.2 Variables

The growth grammar specification allows to declare variables and constants in the heading of the L-system. Four different types of variables are used:

- stochastic variables,
- sensitive variables,
- local register variables,
- index variables.

Stochastic variables allow to specify numbers which are generated at runtime by a pseudo-random number algorithm and which follow a certain distribution. Different types of stochastic distributions are available. Here the uniform and normal distribution are applied. Sensitivity is explained further on. It is one of the main extensions in the growth

Command	**Meaning**	**Effect**
F	forward by generating a unit	-
f	forward without generating a unit	-
S	store position of turtle	-
T	generate triangle	-
C	connect points	-
RU	rotate around up axis	-
RL	rotate around left axis	-
RH	rotate around head axis	-
[	begin branch	-
]	end branch	-
D	set diameter of unit	persistent
Dl	set diameter of unit	local
L	set length of unit	persistent
Ll	set length of unit	local
N	set the needle parameter	persistent
Nl	set the needle parameter	local
P	set colour of unit	persistent
Pl	set colour of unit	local
K	create local variable	local
A	set value of local variable	local
&	repetition	-

Table 1: Turtle commands. For more commands see Kurth (1999)

grammars, compared with classical L-systems. Sensitivity returns information about the structure to the L-system. Functions are existing which query the structure (see chapter 2.3.3). But also the length or diameter of elementary units can to be stored in a variable. Kurth (1998) introduced the local register. It is attached to an elementary unit and hence local. It is possible to alter the value of this local variable with the turtle command *A*. An index is a counter which increases its value automatically with each repetition within the scope of repetition operators. Listing 2 shows examples for the declaration. In line one a constant *con* with the value 8.0 is declared. Line two declares a uniformly distributed stochastic variable with values from the interval [0; 100]. The keyword *length* in line three indicates, that the length of units is stored in the variable. And in the last line a local register variable *loc* is declared.

Listing 2: The declaration of constants and different forms of variables in growth grammars.

```
\const con 8.0,
\var   uni uniform 0 100,
\var   l length,
\var   loc local,
```

2.3.3 Growth grammar extensions

One main extension of the *GROGRA* specification are globally sensitive functions (cf. Kurth and Sloboda 1999b). During the interpretation process in insensitive grammars it is not possible that a generated structure has backwards effect on the application of the rules. Shading calculation for instance is such a problem, where existing structure exerts influence to structural growth in the following steps. The sensitivity allows to take such processes into account. The user can specify functions which are able to investigate the structure. The function returns values which can be used within the grammar. Figure 3 shows the flow of control which is taking place by applying sensitive rules. This figure additionally shows the functionality of interpretive rules. They are also an extension of the classical L-system approach. Interpretive rules have no influence on future structures. Primary the generative rules are applied which create the new string. By application of the interpretive rules the structure is created. Interpretive rules are optional, but they make sense in certain situations.

Table 2 presents the sensitive functions which are used in the stand modelling L-systems. The functions 14 and 15 are created for the needs of tree stand modelling and are described in detail in the next chapter.

Function 8 (f_8) calculates the degree of occlusion of the sky hemisphere. It serves to simulate competition for light. The competition for light is assumed to result from local shading processes. The hemisphere is discretized after den Dulk (1989). He divided the hemisphere into sky segments which donate light. The segments are represented as directions. The sensitive function f_8 investigates from the point of view of a reference unit each sky segment and counts if it is touched by another unit of the structure. The returned value is the ratio between touched and non touched segments. Function 15 (f_{15}) does essentially the same. The only difference is that each touched sky segment is weighted with the sine of its inclination angle. Hence sky segments near to the horizon have much lower significance compared with segments near the zenith.

sum is not a sensitive function but requires sensitivity. It is not used in this work, but shows that also sensitive operators are existing. *sum* is an arithmetical operator which accumulates attribute values of tree units in the branching system which emerges distal to a reference unit. The operator gets two parameters. The first one is a condition and is optional. With this condition is it for instance possible to exclude all units with an empty needle parameter from calculation. The second parameter is essential and specifies the type of information which shall be accumulated. The type is characterized by a sensitive variable like the length or diameter of tree units.

Function	Meaning
$f_8(col)$	Occlusion of sky. The sky hemisphere is discretized after den Dulk (1989) into segments. A segment is considered to be covered if from the viewpoint of a reference unit a tree unit *touches* the hemispherical segment. The variable *col* specifies units which shall be excepted by colour.
$f_{14}(col)$	Calculates the distance between the end point of a tree unit and the surface of the next triangle in a certain direction. The triangle has to be generated with command $T(e_1, e_2, e_3)$ in the L-System. The variable *col* specifies units which shall be excepted by colour.
$f_{15}(col)$	Like function 8, but each touched sky segment is weighted with the sine of the inclination angle of its corresponding direction.
sum$(cond,var)$	The arithmetical operator *sum* accumulates the sensitive variable *var* over a branching system distal to the reference unit.

Table 2: Used sensitive functions (f_n) and the operator *sum*; see Kurth (1999). *col* denotes the colour of an element and e_n are the nodes of a triangle.

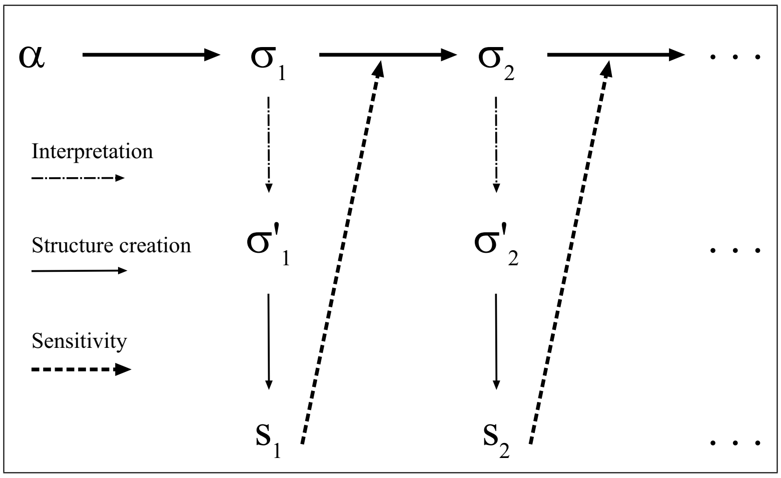


Figure 3: The structure creation steps during application of a sensitive growth grammar. α, σ_i, $\sigma_i^{'}$ denote strings, S_i geometrical structures. The interpretation and sensitivity step is optional (cf. Kurth and Sloboda 1999a,b).

2.4 Extensions for simulating simplified crown structures

A simple structure of tree crowns is used in modelling tree stands. The crowns are represented by an irregular pyramid with eight corners. The main stem is a cylinder. The degree of competition between individuals is determined by the distance of crowns to the crown of the next neighbour. The distance is defined as the distance between a corner of the irregular pyramid and the surface of the crown of the next neighbour. The direction is determined by crown structure (cf. figure 6). An overlapping of crowns results in negative distances (cf. figure 7). Figure 4 and figure 5 show the crown structure in side and top view (cf. Pretzsch 1992a).

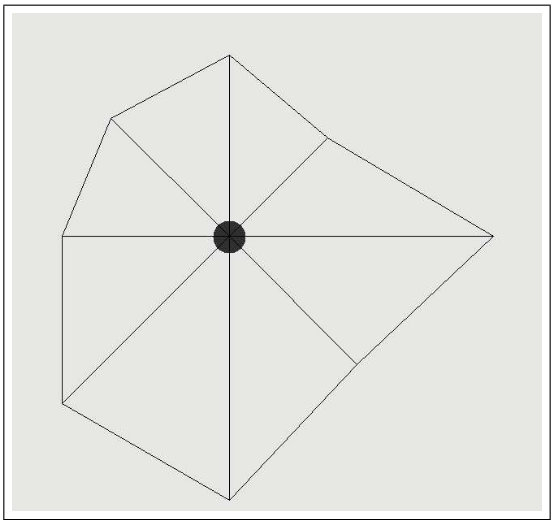


Figure 4: Top view of simple crown structure.

The L-system specification had to be changed for this purpose in two points: Firstly it is ineffective to represent an irregular pyramid only cylindrical elementary units. It is more reasonable to represent an irregular pyramid by triangles than by elementary units which are simple lines. For that purpose a new command is created. The command $T(e_1, e_2, e_3)$ creates a triangle with the nodes e_1, e_2 and e_3. The nodes are designated by an index which refers to a point. The index is constituted by the command $S(e_n)$.

Imaging the pyramid one will see that it has nine corners: the top corner which represents the top of crown and the eight nodes of the basic area (basic nodes). Calculating the distance to the crown surface of the next neighbour shall be done with the coordinates of the basal nodes. The direction is determined by the vector from the stem in the height of the crown base and a basal node. This line is the basic unit of the pyramid. It is

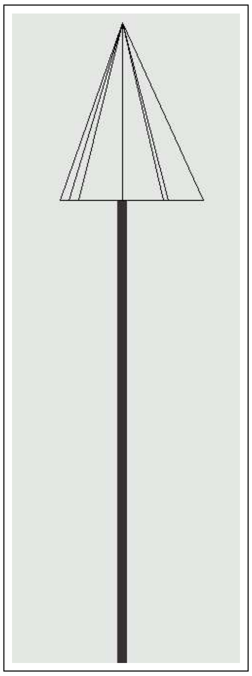


Figure 5: Side view of simple crown structure.

stored as an elementary unit. Using start and endpoint of it, it is possible to calculate the direction of the basic unit. The representation of this basic unit as elementary unit allows to store additional information into this unit. That helps to differentiate between interesting and uninteresting units.

The second change concerns sensitive functions. The calculation of distance between a basic unit of a pyramid and the surface of a triangle has to be done by a new sensitive function. So a new function was added to the collection. Listing 3 shows the declaration of this C-function (for the complete code see appendix on page 155). The function gets the current unit as its argument. It calculates the minimum distance to each triangle in the global triangle list. The minimum distance can be negative if the corresponding triangle is "behind" the unit. Negative values indicate overlapping. The function returns the signed distance. It is possible to use this distance within an L-system, for instance to implement a factor of reduction depending on crown distance.

f_{15} which is similar to f_{14} has been described in the last chapter. To complete the description listing 4 gives analogously to listing 3 the declaration of the function (for the complete code also see appendix on page 155). The parameter co (colour) has the same functionality as in f_8.

Listing 3: Declaration of function 14 (Calculation of distance between a tree unit and the surface of a triangle (see appendix on page 155 for definition).

```
float function14(int ib, struct spross hu *current);
```

Listing 4: Declaration of function 15 (Calculation of occlusion of the sky) (see appendix on page 155 for definition).

```
float function15(int ib, struct spross hu *current, int co);
```

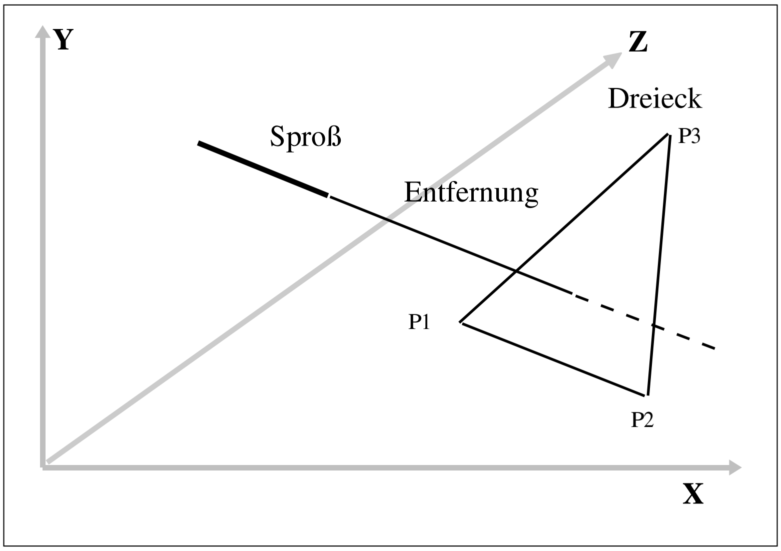


Figure 6: Distance between elementary unit (shoot) and triangle, which represents a part of tree crown.

2.5 The interpreter *GROGRA*

The software *GROGRA* ("**Gro**wth **Gra**mmar interpreter"; Kurth (1994, 1999)) was primarily designed as a system for the interpretation of extended L-systems. *GROGRA* was specifically tailored to the simulation of forest trees in ecophysiological applications. (There exist other L-system softwares, e.g., L-Studio/cpfg; Prusinkiewicz et al. (2000)). Beyond the pure grammar interpretation, *GROGRA* provides additional features: a data filter to represent measured trees, interfaces linking it to process-oriented simulation tools and to statistical software (figure 8), and analysis tools. An overview of analysis options available with *GROGRA* is given in table 3.

This design of *GROGRA* allows the representation of L-system-generated and manually-measured trees in the form of the same data structure and their comparison with the same

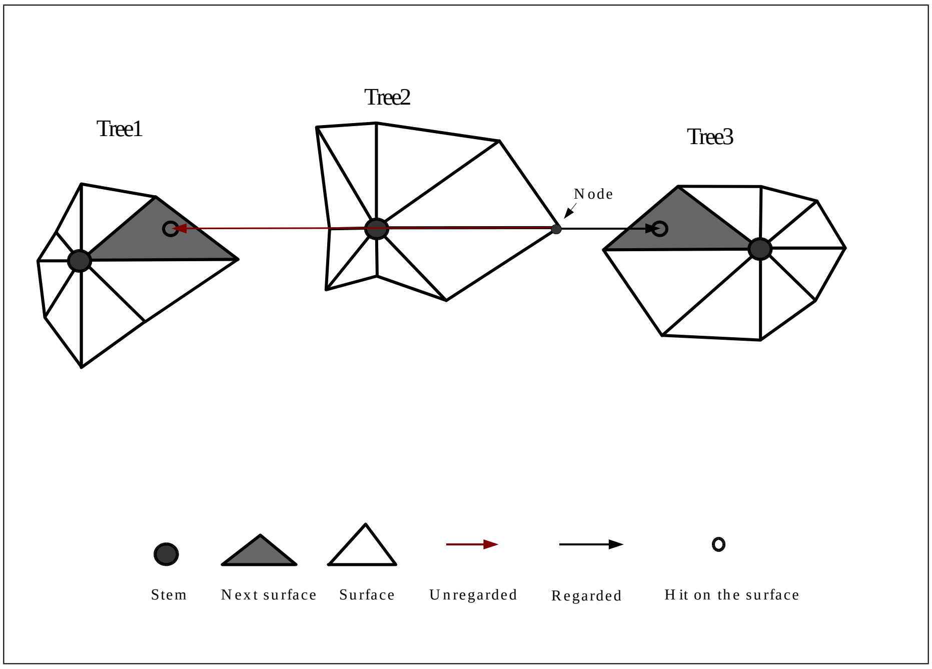


Figure 7: A negative and a positive distance. The gray arrow denotes the minimum distance but is "behind" the stem and stays unregarded. The other distance is regarded.

analysis tools (see Kurth and Sloboda (1999a) for examples of Norway spruce trees). This internal structure is a linked tree of records (cf. Godin 2000), which is not very different from *LIGNUM*'s internal tree representation.

For a more detailed description of *GROGRA* it is referred to Kurth (1999) and Kurth (1994). The last one is the reference manual and is available via Internet at: http://www.uni-forst.gwdg.de/∼wkurth/grogra.html.

Elementary	gives fundamental parameters: number of elementary units, number of terminal units, total length, volume, surface area and others — available for the simulated structure as a whole and for individual plants
Distribution	frequency tables: elementary units, compound units, axes, el. units per compound unit etc., in total and for each branching order
Table	for each elementary unit (shoot): identifier, id. of mother unit, number of daughter units, length, diameter, branching angle, squared sum of daughter diameters and other parameters, suitable for processing with statistical software
Topological	number of components, number of (graph-theoretical) links, maximal and average topological depth and other topological parameters, see Oppelt et al. (2001)
Pathlength	table enabling analysis of diameter-pathlength relation according to McMahon and Kronauer (1976) and other approaches for tapering
Cubic Cells	rasterization of leaf area suitable for interfacing with turbid-medium radiation simulators (see Kurth and Sloboda (1999b) for details)
Fractal	rasterization with varying grid resolution, yields data file for further processing with box-counting analysis software (cf. Oppelt et al. 2000)
Bole	table with discretized description of the main stem
Level	total length and leaf area included in horizontal layers with specified thickness
Branch Positions	for each elementary unit: number of daughter units and their positions (for further processing)
Axes	parameters for each axis, including average interbranch distance and its standard deviation
Population Dynamics	time series of frequencies of elementary units of different types
HYDRA	rediscretization interface for processing with water flow simulator *HYDRA* (Früh and Kurth 1999)

Table 3: Analysis options of *GROGRA* (from: Dzierzon et al. 2003)

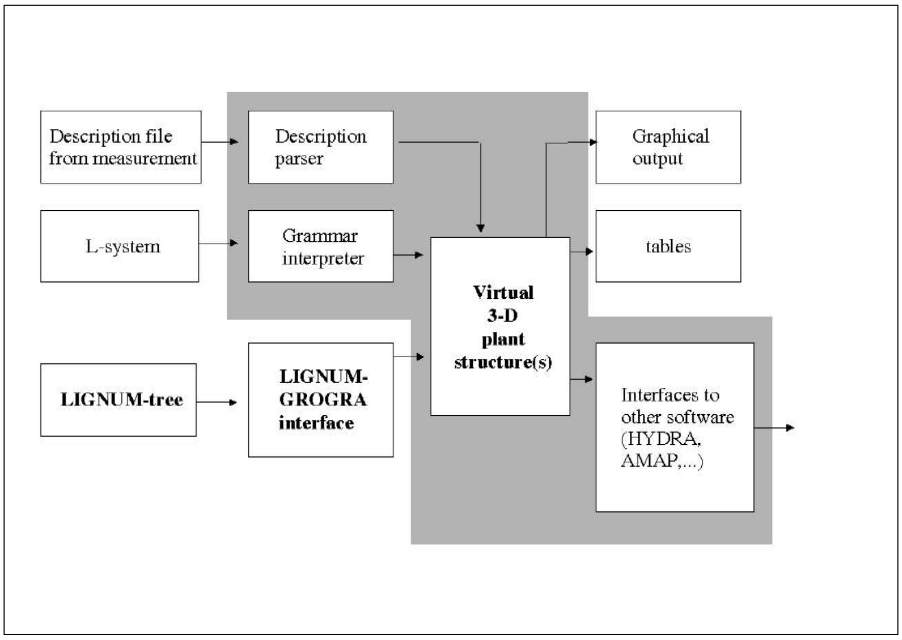

Description file from measurement
Description parser
Graphical output
L-system
Grammar interpreter
tables
Virtual 3-D plant structure(s)
LIGNUM-tree
LIGNUM-GROGRA interface
Interfaces to other software (HYDRA, AMAP,...)

Figure 8: Structure of the software *GROGRA*. The grey zone denotes the *GROGRA* kernel (from: Dzierzon et al. 2003).

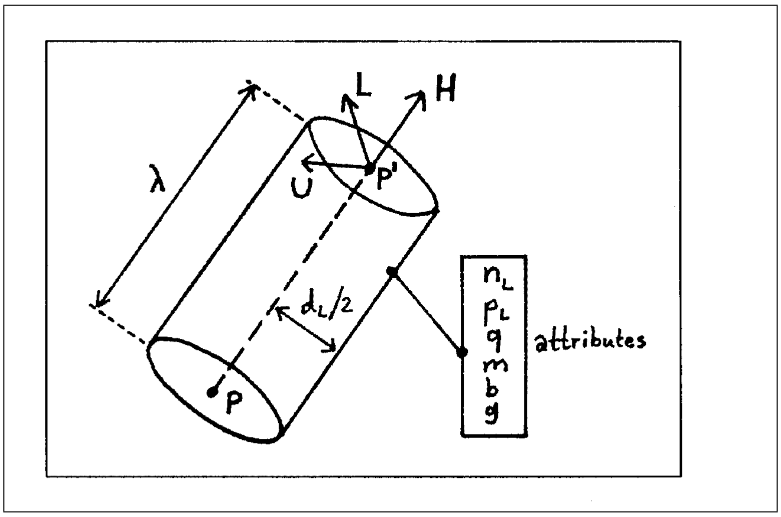


Figure 9: The rotation vectors ($\vec{U}$, $\vec{L}$, $\vec{H}$) used by RU, RL and RH commands and other attributes of an elementary unit in *GROGRA* (from: Kurth 1994).

3 Tools

Extending possibilities by integration and interfacing different tools for modelling and analysing architecture in the first place requires the description of these tools. Especially interfacing requires the clarification of the internal data structures of the tools. Details about the process of interfacing and integrating are part of the *GroDisc* description further on.

As simulation tools, *LIGNUM* and *GROGRA* are used. *GROGRA* has already been described above, but the functionality and the data structure of *LIGNUM* have to be clarified. Within the scope of analysis and model integration three tools are used: *AMAPmod*, the statistical software *R*, *SAS* and *STATISTICA*. The integration is achieved by implementing a new software, *GroDisc*, which is also characterized here. The chapters *LIGNUM* and *AMAPmod* are slightly changed citations from Dzierzon and Kurth (2002), Dzierzon et al. (2003).

3.1 *LIGNUM*

The structural representation of a tree with *LIGNUM* can be formally defined as follows (cf. figure 10): The model tree of *LIGNUM* consists of one axis. An axis is a possibly empty sequence of tree segments (TS), branching points (BP) and exactly one bud (B) terminating the sequence. Each tree segment must be followed by exactly one branching point. A branching point is a set of zero or more axes.

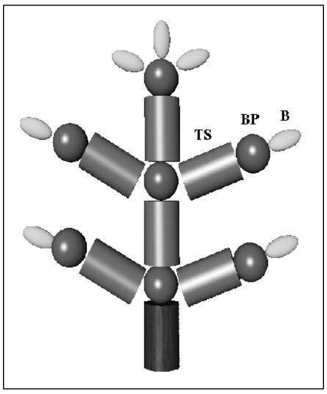


Figure 10: The structure of a *LIGNUM* tree. TS = Tree segment, BP = Branching point, B = Bud (from: Dzierzon et al. 2003)

The definition does not imply how *LIGNUM* should be implemented. The current implementation as a list of lists is one possibility. However, the definition sets the requirement that the units appear in the axis in a certain order: Tree segments and branching points occur alternatingly and the bud is the rearmost unit. There can also be an axis containing only one bud. The definition of the branching point simply captures the branched structure of the tree.

A tree segment corresponds to a piece (segment) of a woody axis, containing heartwood, sapwood, bark and possibly foliage. Coniferous and deciduous trees are distinguished. Tree segments of conifers contain a cylinder of needles and have buds at the end. The segments of deciduous trees have individual leaves and buds. In both cases an axis is a sequence of segments and branching points terminating in a bud. Axes correspond to the stems and branches of real trees. Branching points can be thought of as points that connect branches (axes) to the stem or higher order branches in real trees. The tree itself is thus an aggregation of its axes (Perttunen et al. 1996). The roots are considered only in terms of their mass. The metabolic functioning (e.g. photosynthesis and respiration) and the physiological state of the units are directly associated to the roots.

LIGNUM simulates the interception of photosynthetically-active radiation (PAR) in the tree crown, the processes of photosynthesis and respiration and the allocation of carbon among the tree segments. Secondary growth of the woody axes is mainly controlled by the necessity to maintain sufficient sapwood area for hydraulic supply of the supported foliage. The number and length of new shoots growing each year is determined by the available amount of carbon and by species-specific morphogenetic patterns.

In this project the tool *LIGNUM* is used to simulate Scots pine (*Pinus sylvestris* L.) trees, i.e. conifers. Hence a tree segment contains sap- and heartwood, bark and needles. The relative thickness growth of the woody parts of branches is based on the pipe model (Shinozaki et al. 1964). Used parameters are listed in table 4. The parameters are the same as in Perttunen et al. (1998) except a new component within the light model, which mimics a surrounding tree stand.

The radiation model is a central part of *LIGNUM* and it treats attenuation of solar radiation in the crown of the tree itself, i.e. self shading. The amount of light which reaches a tree segment consists of two components in the calculation. First, the sky is divided into sectors and the amount of radiation coming from each sector during the growing period is assumed to be known (Perttunen et al. 1998). The amount of radiation coming from each sector is calculated from zonal brightness of standard overcast sky (Ross 1981). In the simulations, the total incoming radiation was 1200 MJ/m^2 of photosynthetically-active radiation during growing season corresponding to conditions in southern Finland

(Stenberg 1996). Second, the beam of light coming from a sky sector may go through foliage or hit the woody part of a segment in the tree crown. If it hits the woody part, it is blocked. The attenuation caused by foliage is calculated according to Oker-Blom and Smolander (1988), see also Kellomäki and Strandman (1995). For further information it is referred to Perttunen et al. (1998).

Parameter	Meaning	Unit	Value
a_f	Needle mass / tree segment area relationship	$kg \cdot m^{-2}$	1.3
f_c	The coverage of needles	$kg \cdot m^{-2}$	14
a_r	Needle / root relationship	$kg \cdot kg^{-1}$	0.5
l_R	Length/radius for a new tree segment	-	100
m_f	Maintenance respiration rate of needles	$kgC \cdot kgC^{-1} \cdot a^{-1}$	0.2
m_r	Maintenance respiration rate of roots	$kgC \cdot kgC^{-1} \cdot a^{-1}$	0.24
m_s	Maintenance respiration rate of sapwood	$kgC \cdot kgC^{-1} \cdot a^{-1}$	0.024
q	Tree segment shortening factor	-	0.1
S_r	Senescence rate of roots	a^{-1}	0.33
S_s	Senescence rate of sapwood	a^{-1}	0.07
ρ_w	Density of wood	$kg \cdot m^{-3}$	400
ζ	Fraction of heartwood in new tree segments	-	0.6
P_r	Proportion of intercepted solar radiation that is used in photosynthesis	-	0.001
d	Density of surrounding stand	ha^{-1}	10000
k_e	Extinction coefficient (for total needle area) of radiation in tree stand	-	0.14

Table 4: *LIGNUM* parameters for simulated Scots pine (from: Dzierzon et al. 2003).

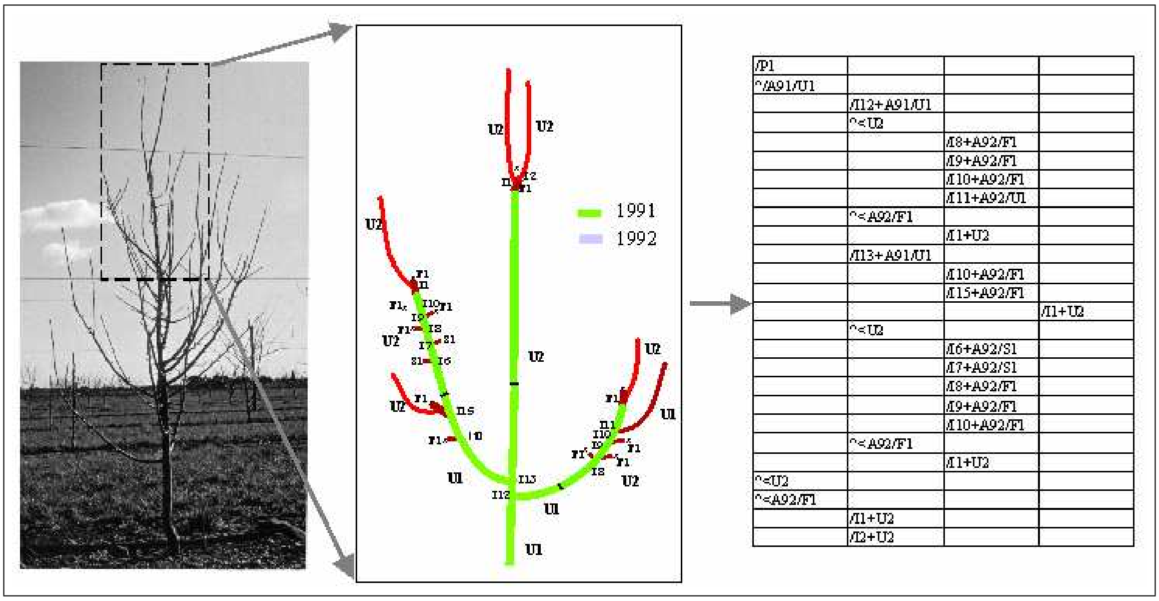
3.2 *AMAPmod*

AMAPmod is a tool to support the work of plant modelling. *AMAPmod* provides users with a methodology and corresponding tools to measure plants, create databases and analyse information extracted from these databases (Godin et al. 1998). That means that *AMAPmod* introduces two components: a database and a query-language (*AMAPmod* querying language (*AML*)) to study the database in a way that suits plant modelling. For this work it is important to describe the database used in *AMAPmod* and the way of analysing the tree structures. *AMAPmod* uses so called multi scaled tree graphs (*MTG*) (Godin and Caraglio 1998). The idea of multi scaled tree graphs will be illustrated by an example. Observing a tree (e.g. a Scots pine) standing a kilometre away, one can perceive a crown and a stem. But nothing can be told about individual leaves or even shoots. The smaller the distance between observer and object, the more is visible. This means that it depends on the scale of view how many details can be seen. In general scaling information is not represented in data structures. One of the frequent consequences can be very slow graphical output. Hence it makes sense to store information about the membership of objects to others with a higher level of scale. To use the example, a shoot is member of a branch and a branch is member of a crown. Godin and Caraglio (1998) developed a mathematical formalism based on graph theory designed to represent such information in a database. The formalism also has impact on visualization of trees and on data exploration. The relevance for visualization originates of the observation that it does not make sense to build up polygons on a screen without seeing them because they are too small or irrelevant at the given scale. The importance for data exploration follows from the fact that botanists often want to know e.g. how many internodes are inside a shoot.

Figure 11 shows an example of such a *MTG*. On the left side we see a measured tree, in this case an apple tree. This is described in three different levels of resolution in the central picture. **I** stands for internodes, **U** for growth units and **P** for the whole plant. On the very right side the table representation of the *MTG* file is shown. The indentation of a column represents the order of a unit. When a database is created, data have to be extracted and analysed. With *AMAPmod* a tool is available to do such work. From a practical point of view, *AMAPmod* is an interactive interpreter that processes user's commands one after another (Godin et al. 1999). In a more technical view, *AMAPmod* contains three main modules. Every module contains special functions that can be used to build and manage dedicated data types (cf. Godin et al. 1998):

- the kernel,
- the *MTG* module and
- the *STAT* module.

The kernel module gives basic functions and operators like addition and multiplication or sine and cosine. With the *MTG* module, one obtains possibilities to interpret the *MTG* structure. It is for instance possible to calculate the order of a given shoot or internode. In the *STAT* module all statistical functions are given. If one wants to create a histogram about the distribution of diameters or of any other attribute of a tree element one can do it with *AMAPmod*, using the query language *AML*.



/P1			
^/A91/U1			
	/I12+A91/U1		
	^<U2		
		/I8+A92/F1	
		/I9+A92/F1	
		/I10+A92/F1	
		/I11+A92/U1	
	^<A92/F1		
		/I1+U2	
	/I13+A91/U1		
		/I10+A92/F1	
		/I15+A92/F1	
			/I1+U2
	^<U2		
		/I6+A92/S1	
		/I7+A92/S1	
		/I8+A92/F1	
		/I9+A92/F1	
		/I10+A92/F1	
	^<A92/F1		
		/I1+U2	
^<U2			
^<A92/F1			
	/I1+U2		
	/I2+U2		

Figure 11: The representation of a measured tree in a multiscaled tree graph. (From: http://www.cirad.fr/presentation/programmes/amap/themes/math2.shtml).

3.3 *GroDisc*

3.3.1 Motivation and requirement

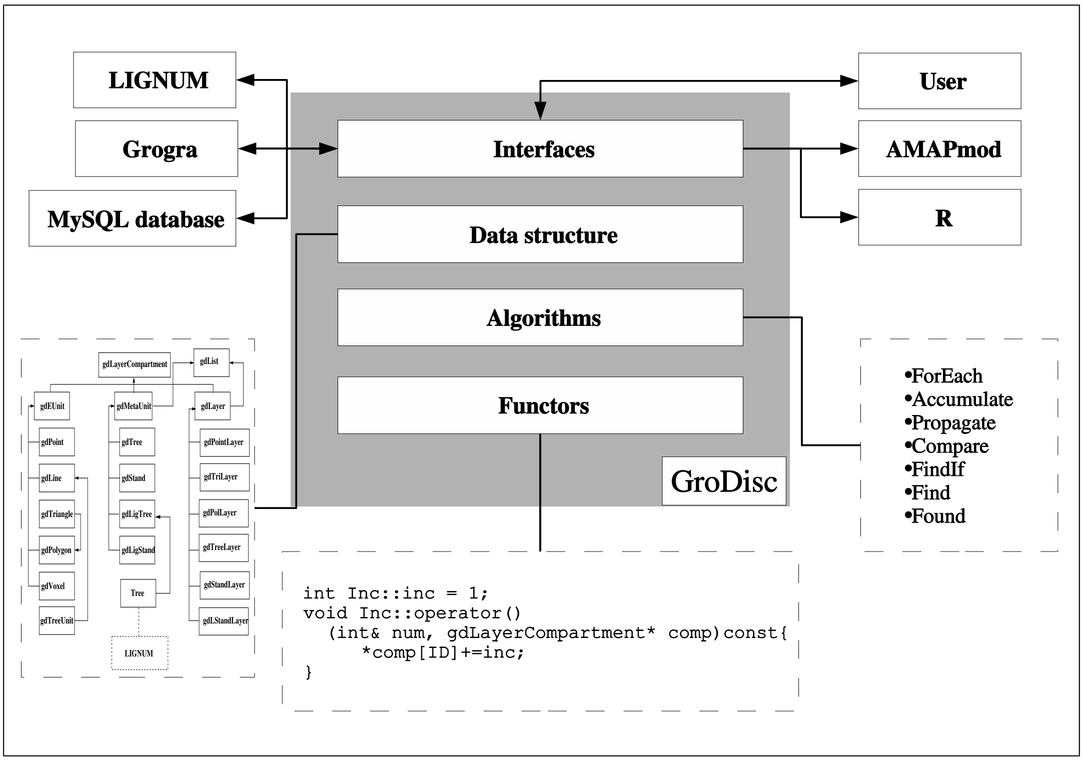
The introduction has already focused on the different steps of reasonable modelling: analysis of reality, creation of virtual reality (the model), simulation and analysis of simulation results (validation). This work emphasizes the individual based modelling of tree stands. The submodels for the trees are simplified. A tool should be at hand which assists the analysing part of the modelling process. In this chapter a tool is introduced which handles architectural 3D information and provides analysis algorithms for modelling and validation purposes. The tool has to fulfil some principles and guidelines. The principles are modularity, generality and simplicity. The fulfilment of these principles shall serve the transparency of the software and of the computer code. Out of these principles and from the existing necessities for architectural representation and analysis some guidelines arose.

The modular design of a software is indispensable for the transparency of the code. The modular design of *GroDisc* (Kurth 1999) is shown in figure 12. It consists of the proper data structure which is a *C++* class hierarchy, the interfaces to other tools and the algorithmic part. The algorithms are divided into the algorithms which provide the movement through the structure (cf. Breymann 1998, Stroustrup 1998) and the instructions which are applied to an object (functors).

Reprogramming statistical analysis tools is inefficient. The integration of existing tools is one main guideline. Tree architecture can be analysed by the tool *AMAPmod* (Godin et al. 1998) with a multiscaled data representation (Godin and Caraglio 1998). More basic statistics are provided by the software *R*.

The integration of tools and the comprehension of simulation results presupposes data interfacing. In addition to the statistical tools, data interfaces to *LIGNUM* (Perttunen et al. 1996, 1998) and *GROGRA* are needed which are used for modelling trees.

Validation and analysis of three-dimensional structure requires the querying of data out of the structure. Some data are not explicitely available. They have to be calculated out of the structure. Such processes occur usually along with discretization and are the main purpose of *GroDisc*.



```
int Inc::inc = 1;
void Inc::operator()
   (int& num, gdLayerCompartment* comp)const{
       *comp[ID]+=inc;
}
```

Figure 12: The modular structure of *GroDisc*. For the data structure also see figure 14 on page 34

.

The following sections give a description of the software and its modules. Firstly the internal data structure is introduced which is based on the list concept of the *C++* standard template library (*STL*). The data structure tries to handle architectural information in a general way. The design of algorithms and functors which are due to the *STL* are described later in this text. Interfaces are implemented for this software. Figure 12 shows the interfaces. A description is also given here. The data format of *GROGRA* is also used in *GroDisc* in the form of the so called *DTG* ASCII files, hence the interface is directly part of the data structure of *GroDisc* and characterized there.

3.3.2 Data structure

The internal data structure of *GroDisc* has to image architectural structures and combine them with discrete data. The need for architectural information encompasses decomposition, geometry and topology (cf. Godin 2000). *GroDisc* is implemented in OOP using *C++*, hence the first step is indeed to create a suggestive class hierarchy. The problem was to create a structure which is maximally general.

The geometrical part of the data structure is abutted to Geographical Information systems (*GIS*). The reason is that the data models of *GIS* still handle geometrical information in a general way. To characterize the data model of *GroDisc* a short expedition to *GIS* is necessary. In *GIS* some terms are used in a specific way. A real entity is called an "object". The computational implementation is the "representation" of the object. In the scope of *GIS* the main data models are

- geometrical representations,
- regular networks (grids),
- irregular networks (like *TINs*) and
- images (bitmaps).

Regular networks in the 3D version are part of *GroDisc* and used in Oppelt et al. (2000). Not used are irregular networks like *TINs* and Bitmaps. Pfund (2002) describes the possibilities for extending *GIS* to the third dimension. Up to now the implementations of 3D *GIS* are in a preliminary stadium which gives another reason for building up one's own software. For geometrical objects Requita (1980), Streilein (1999) define guidelines for the representation of the objects:

- definition range: quantity of the object's attributes, which can be represented.
- completeness: geometrical quality (accuracy, level of detail),
- unambiguousness: an object is unique if there exists to each object exactly one representation and to each representation exactly one object,

- compactness: storage space needed,
- efficiency: computing time for creation, analyse and processing.

The design of a geometrical representation has to optimise these guidelines. Three different approaches are existing to image a real world entity (Pfund 2002): spatial enumeration, constructive solid geometry (*CSG*) and boundary representation (*B-Rep*).

Figure 13 illustrates these three possibilities. Spatial enumeration means that a real world entity is approximated by the filled cells of a 3D grid. Constructive solid geometry is the representation by set theoretical combinations (union, intersection, difference) of primitives. Primitives are simple geometrical bodies like points, cubes or spheres (cf. Pfund 2002). The most common way of representation is boundary representation: The entity is described by surfaces, nodes and edges. In *GIS*, boundary relationships are very important. The B-rep simplifies the data handling of border relations.

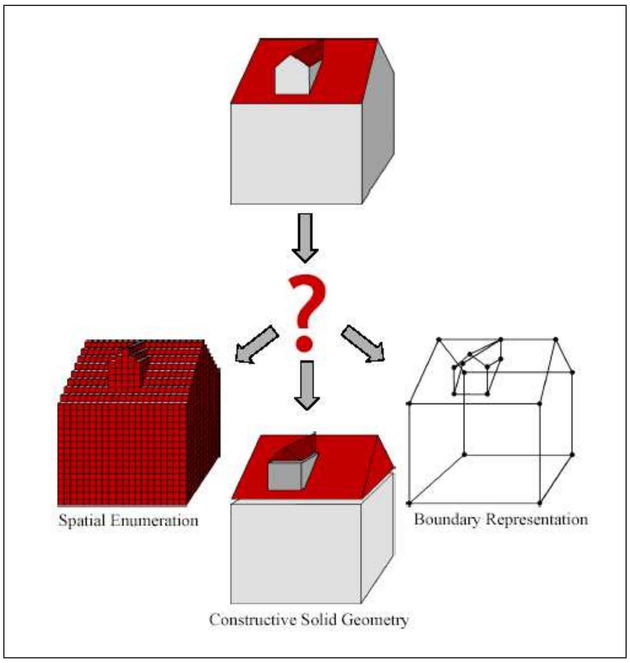


Figure 13: Basic geometric modelling concepts for 3D-*GIS*: spatial Enumeration, *CSG* and *B-Rep*(from: Pfund 2002).

Boundary relations are not important for this work, hence the representation method *CSG* is chosen (but using only the "union" operation as the set theoretical operation). The boundary representation would contradicted the guidelines of compactness and efficiency. The representation of a tree with a large number of segments using nodes, edges and surfaces would have meant a quite large storage need. Apart from this the *CSG* is the most direct way to represent real entities. Representing tree and tree stand architecture by *CSG* fulfils the criteria of completeness and definition range.

Voxel spaces are seen as a collection of the primitives *voxel*. In principle voxels correspond to regular networks (grids). Nevertheless, one can still represent a voxel as a primitive. Two different types are existing: cubical and cylindric. They are described in connection with the class hierarchy.

The primitives have to be represented by a class hierarchy. The class hierarchy of *GroDisc* is shown in figure 14.

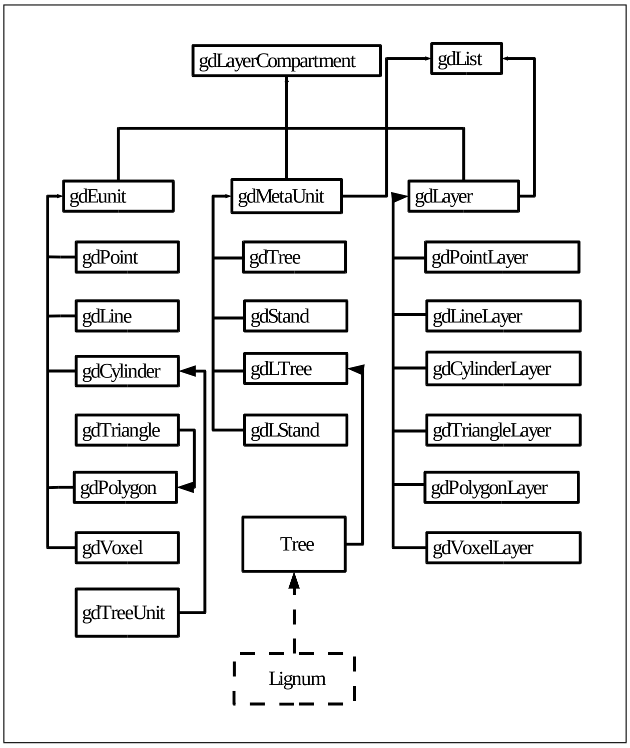


Figure 14: The class hierarchy of *GroDisc*.

Primitives are simple geometrical bodies which image real world entities. They can not be divided into smaller elements. A cylinder for instance can represent a tree shoot. In ecological modelling such representations are called elementary units (EU). In order to show the association of *GroDisc* with ecological modelling this convention is adapted. Elementary units (primitives) are:

- points,

- lines,
- cylinders,
- triangles,
- polygons and
- voxels.

These elementary units are implemented as classes in the class hierarchy. All the units inherit the abstract class *gdEUnit* which represents the elementary units. An assembly of elementary units is kept in a layer (*gdLayer*). In order to keep the overall view only consistent, units are pooled in a layer. This means that a layer stores only geometrical information of one type. For this reason, the class hierarchy has a layer class for every type of elementary unit.

GroDisc has its application in analysing model results of tree and tree stand models. It is indeed possible to represent a tree by cylindrical units, but a tree itself can be also seen as a unit. Because of the fact that a tree is a collection of elementary units, a tree itself can not be treated like an *EU*. To solve this problem, a meta unit is introduced into the class hierarchy. A *gdTree* is a unit which is part of a layer but contains cylinders as members. A *gdTree* is not directly a list of cylinders. The class *gdTreeUnit* is a cylinder, but it contains some information which are typical for tree units like a parameter about leaf information.

The design concepts of the classes *gdLTree* and *gdLStand* are identical to *gdTree* and *gdStand*. The *L* stands for *LIGNUM*. Those classes use the *LIGNUM* class hierarchy (cf. figure 15) and are useful for the interface between *GroDisc* and *LIGNUM*. Those *LIGNUM* units are assembled within the class *gdLStandLayer*.

Each class inherits *gdLayerCompartment*. And each class is a compartment of a layer, even the layer itself. This abstract class stores the attributive information. Attributes are stored within maps of information with the same data type. These attributes are technically addressed by a double bracket operator. The design is similar to the field concept in data bases. The argument of the first operator is the information about the type of the field. The second operator has an argument which contains the name of the field as a string. Listing 5 exemplifies the bracket operators. A *gdLayerCompartment com* is declared and the identifier, which is an integer, is stored in the variable *num*. The example also shows the possibility to store a value to the attribute of a layer compartment. Here the attribute gets the value one.

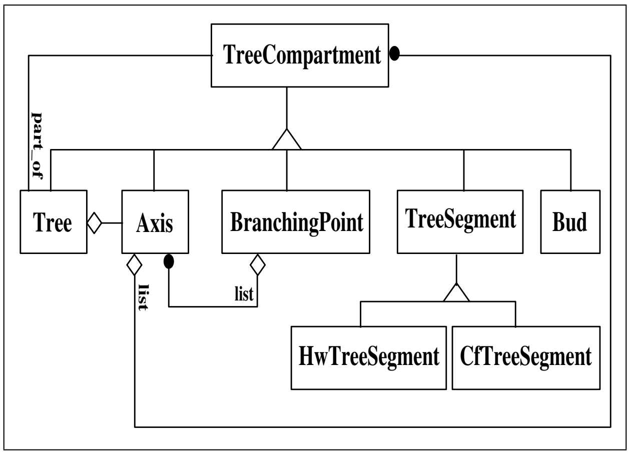


Figure 15: The class hierarchy of *LIGNUM*.

Listing 5: Example for storing attibutal information.

```
gdLayerCompartment com;
int num = com[INT]["ID"];
com[INT]["ID"] = 1;
```

The decomposition information is already represented by the list structure. For instance a tree is decomposed by shoots which is illustrated by the fact that a *gdTree* has *gdTreeUnits* as list members.

In general list structure topology can be represented in an implicit or explicit way. Implicit means that the relation to neighbours is expressed by the design of lists. An example for such a representation can be found in Perttunen et al. (1996, 1998) (cf. figure 15). This list structure constructs a definite topological structure. The main disadvantage of such an approach is that the neighbours have to be calculated out of the structure and that the process to build up such a structure is rather complex. To save calculation time and to keep the structure more simple, in *GroDisc* the topological information (neighbourhood) is expressed explicitly by special lists as members of the class *gdLayerCompartment*. Existing lists are:

- *_neighbours*,
- *_mothers*,
- *_daughters*.

The list *_neighbours* contains pointers to geometrical units which are defined as neighbours if the relationship is symmetric. However in real trees the relationship between the tree units is directed. In that case it is reasonable to differentiate between units before and after the respective unit. A tree unit has a mother and a defined amount of daughters. The mother would be part of the list *_mothers* and the daughters of the list *_daughters*.

3.3.3 Algorithms

In general, every terminating programme implemented in a convenient language realizes an algorithm. The term can be defined as a logical arithmetical or computational procedure that, if correctly applied, ensures the solution of a problem (after www.wordreference.com). The problem here is to query or manipulate architecture. The architecture is represented in a list structure. A general algorithm has to apply rules to the elements of the list structure. This chapter introduces functions which offer a general way to apply rules to list elements. The functions originate from the algorithm concept of the *STL*. The proper rules are arguments of the functions and are described in the next chapter.

Listing 6 exemplifies an *STL* algorithm which sends a signal through a simple list. The signal or rule is represented by an operator *op*. The list members are represented by iterators. Technically these iterators are pointers to the corresponding element of the list.

Listing 6: The for_each algorithm of the *STL*

```
template <class It ,class Op>
void for_each(It first , It last , Op op){
  while(first!=last) fun(*first++);
}
```

GroDisc offers algorithms which are similar to the *STL* algorithms. However, the design is different. The reason for this is the list structure which contains parallel lists. It is not possible to pass through such a list directly in a sequential way. In addition *GroDisc* algorithms do pass the lists as such to the algorithm and not the iterators. It is more reasonable to pass the layer compartments directly to the algorithm. Within the function it is tested what kind of compartment is available. The rule is then applied to the compartment. If the compartment itself is a list, the function is applied to that compartment again. Different types of algorithms are existing: Table 5 gives an overview about the existing functions. The functions *ForEach*, *Accumulate*, *FindIf* and *FoundIf* are equivalent to their *STL* counterparts. *ForEach* sends a signal through the list structure, *Accumulate* does exactly the same but returns a value. This function is used to query data out of layer compartments. *FindIf* searches for layer compartments applying a rule which has to be implemented by an operator which is applied to the list members. *FindIf* returns zero if nothing is found. *FoundIf* does exactly the same but only returns the information whether a compartment is found or not.

The algorithm functions of *GroDisc* have additional arguments. The *GroDisc* data structure has three levels: layer, meta unit and elementary unit. The argument *level* allows to specify levels which shall be skipped. The value of this argument is specified by integer variables which can be combined by the bitwise operator *OR* ('|').

In some situations the mentioned algorithms are insufficient. One example is the calculation of the smallest distance within a tree stand. The solution can not be solved by accumulating each tree. All possible tree pairs have to be taken into account. This problem can be calculated in an easy way if an algorithm is at hand which provides the application of rules to pair relations. Two algorithms are at hand: *ForEachPair* and *AccumulatePairs*. Two lists are given to these algorithms as arguments: a reference list and a second list which contains the objects for the comparison process. The pairs are constituted from both lists. The algorithms pass every possible pair relation between these list structures to a rule. The algorithms grap every member of a list and compare this member with each member of the comparing list. This design allows to store intermediate results to the reference units. To give an example: If the distance to the next neighbours of each tree within a tree stand has to be calculated it is possible to store the result as an attribute to each reference tree. Sometimes – like in the example – both lists are identical. Just for convenience an overloaded function is created which has only one layer compartment as an argument. Within the function the compartment is passed twice to the proper compare function. Table 5 gives the algorithms and their arguments. The code of all functions is given in the appendix (see page 140).

algorithm	**return value**	**RLC**	**CLC**	**init. value**	**operator**	**level RLC**	**level CLC**
ForEach	void	yes	-	-	functor	yes	-
Accumulate	void	yes	-	yes	functor	yes	-
ForEachPair	void	yes	yes	-	functor	yes	yes
ForEachPair	void	yes	-	-	functor	yes	yes
AccumulatePairs	void	yes	yes	yes	functor	yes	yes
AccumulatePairs	void	yes	-	yes	functor	yes	yes
FindIf	LC	yes	-	-	predicate	yes	-
FoundIf	bool	yes	-	-	predicate	yes	-

Table 5: The algorithms and their arguments. RLC = reference layer compartment, CLC = compare layer compartment, LC=layer compartment, init. = initial.

3.3.4 Functors

The concept of the *STL* requires the passing of operators which are applied to the list elements to the algorithms. Usually these operators represent functions. In the find algorithms the operators are called predicates. The functor section characterizes the way these operators are implemented in the *GroDisc* structure.

The declaration of the function argument as a template allows different concepts. The main disadvantage of taking functions as arguments is that the number of parameter values is fixed. The call of the operator within the function with a fixed number of arguments does not allow parameterizing complex operations. To give an example: If a layer compartment shall be rotated in 3D space, the values for the rotation angles are needed. However the call of the operator does not allows three arguments. The solution is to follow the concept of so called *functors*. *Functors* are bracket operators within a structure or alternatively within a class. The use of static variables allow to parameterize the rule. The origin of the term *functor* is Stroustrup (1998).

Listing 7 illustrates the way functors work. Each layer compartment has an identifier in the form of a number. As an example, the functor *Inc* increases the identifier by *inc*. The variable *inc* is set to the value one by default. The *Accumulate* algorithm is exemplified by listing 8. This algorithm counts the number of compartments within a structure. The result is stored in the parameter *num* which in this case is also a return value. Listing 9 again shows a functor which is used in the *Compare* algorithm. Here the arguments are different. Two layer compartments are passed to the functor. The example calculates the minimum distance between two layer compartments within a structure. These can for instance be trees. The first parameter is the return value to which the minimum distance is passed. The first compartment is the reference unit. The second compartment is then the neighbour tree to which the distance is calculated. For that purpose the logical operator "||" is abused. It is an virtual operator which represents the calculation of the distance between two layer compartments. It can be reimplemented in each subclass of *gdLayerCompartment*. Here it is used as a distance between the origins of trees.

In certain situations it is necessary to check which data type the most recently used layer compartment belongs to. For instance the information about the midpoint of a voxel is not available in a layer compartment. The method for returning the midpoint is implemented in the class *gdVoxel*. In such a situation the data structure allows the layer compartment to be dynamically casted to a voxel because a voxel inherits *gdLayerCompartment*.

Listing 7: ForEach: example of a functor which increases the identifier of layer compartments (declaration and definition).

```
struct Inc{
  static int inc;
  void operator()(gdLayerCompartment* comp)const;
};

int Inc::inc = 1;
void Inc::operator()(gdLayerCompartment* comp)const{
  *comp[INT]["ID"]+=inc;
}
```

Listing 8: Accumulate: example of a functor which counts layer compartments (declaration and definition).

```
struct Count{
  void operator()(int& num, gdLayerCompartment* comp)const;
};

void Count::operator()(int& num, gdLayerCompartment* comp)const{
  num++;
}
```

Listing 9: ForEachPair: example of a functor which calculates minimum distance between two layer compartments (declaration and definition).

```
struct LowestDistance{
  void operator()(double& dist, gdLayerCompartment* ref,
                  gdLayerCompartment* comp)const;
};

void LowestDistance::operator()(double& dist, gdLayerCompartment* ref,
                                gdLayerCompartment* comp)const{
  double dummy = *ref||*comp;
  dist = (dummy<dist?dummy:dist);
}
```

3.3.5 Interface to *LIGNUM*

With the simulation of *LIGNUM* trees the necessity to analyse the simulation results arises. The analysis of the results is supposed to be done by the *GroDisc* software, hence a data interface is needed. Just for completeness the interface is implemented bidirectionally. The interface is partly described in Dzierzon et al. (2003). Originally it was applied as a *LIGNUM GROGRA* interface. This chapter describes this interface and, in addition, a direct interface to the *GroDisc* data structure.

The interface can take two routes: The first is a file connection. The *DTG* file format of *GROGRA* is used. This file format can be used either in *GroDisc* or in *GROGRA*. The other way is a direct translation to the *GroDisc* data structure.

Firstly the translation of *LIGNUM* trees to *DTG* files is described. The *DTG* file images the internal data structure of *GROGRA*, hence the interface is primary an interface between *LIGNUM* and *GROGRA*. In *LIGNUM* as well as in *GROGRA* the information of interest in a simulated tree is stored in a data type which basically represents the organ "tree segment". The task was to collect the tree segments out of a *LIGNUM* tree and to filter the attribute information out of the segment. In order to translate trees from *LIGNUM* into the *DTG* format it is necessary to consider two items: the information stored in tree segments itself, and how those segments are linked together, i.e. the representation of topology. The representation of topology is different in both tools. For this reason a translation process is needed. Focusing on a segment, *GROGRA* stores information in the data structure of the segment about the location of this segment within the tree architecture. Every segment "knows" its predecessor, that is mother segment. All of the segments are chained in a list. The list itself does not represent the context of tree segments, but the "knowledge" of mother segments satisfies the minimum demands posed by representing topology.

The data structure of *LIGNUM* represents topology in a more implicit way. As already described, tree segments are merged in lists which describe axes in the systematic of Gravelius order (cf. figure 10). Querying such a data structure is a rather complex process. In this case the mother segments for each segment have to be calculated. If one scanned tree segment by tree segment through this list of lists, it would be not guaranteed that the predecessor element is the mother segment - usually it is not. *LIGNUM* offers algorithms to solve such problems. One of them allows querying the structure considering the topology. At a branching point the algorithm allows to treat daughter segments in a quasi simultaneous way. This allows to pass the information "who is the mother" to the daughter segments.

The information of a tree segment can be subdivided into three parts: identification, geometrical and physiological information. In the filtering process information can take three different ways: direct adoption, transformation, and sometimes information has to be calculated newly out of existing data.

Table 6 shows the physiological information which *LIGNUM* holds in a segment. The *DTG* format provides space for five unspecified variables which can be used for whatever information is required. Table 6 shows the physiological information which is stored in a *LIGNUM* tree segment. These five attributes are adapted in the translation process.

Part of physiological information stored in a *LIGNUM* tree segment
Foliage mass
Sapwood mass
Dry weight
Respiration rate
Radius of heartwood

Table 6: Selection of physiological information of a *LIGNUM* tree segment

The converter translates the *LIGNUM* tree directly into a *GROGRA* 3D structure in the form of a *DTG* ASCII file. Technically the converter uses the *LIGNUM* libraries to build up a *LIGNUM* tree. The *DTG* file is readable by *GroDisc*. That opens a data connection from *LIGNUM* to *GroDisc*.

However, it is possible to convert a *LIGNUM* tree directly into *GroDisc* tree. If this happens all physiological information is stored into the *GroDisc* tree segments and can be processed during algorithmic analysis. The advantage of direct translation is that all attributive information can be stored into the *GroDisc* structure. The appendix on page 146 shows the functor which is used for that purpose. The way is the same as in the *LIGNUM DTG* connection. The only difference is that the argument of the functor is not the file but a *GroDisc* tree.

The translation process of *GroDisc* to *LIGNUM* trees is more complex. The topological information is stored very differently. The main problem is to calculate the topology out of the *GroDisc* or *DTG* structure, respectively. It is possible to take advantage of the fact that both the *LIGNUM* tree segment and the *GroDisc* unit have an identifier in the form of an integer variable. Figure 16 illustrates the translation process of the topology on a simple virtual example.

If the origin of data is a *DTG* file, the data are read into a *GroDisc* tree. The *GroDisc* tree is then accumulated. The *GroDisc* tree units are "collected" and put into a *LIGNUM* tree. The problem was to identify the position of a segment within the *LIGNUM* tree. Two functors are necessary for that purpose: one which sends the signal through the *GroDisc* tree and one for the *LIGNUM* tree. The functors are given in the appendix on page 146. The identification of the position is done by the mother segment concept. In a first step, the first segment at the base of the tree is inserted to the *LIGNUM* tree. During the following accumulate process through the *LIGNUM* tree, the mother segment of the reference unit is looked for. If the mother segment is found, the position of the *GroDisc* tree unit within the *LIGNUM* tree is almost known. At this moment, however, it is not clear whether the segment follows the axis or whether a new axis has to be created. The Gravelius orders of the mother and reference unit answer that question. If the orders are identical the reference unit is a unit within the same axis. In that case a *LIGNUM* branching point and a tree segment is built and inserted into the axis list. If the orders are different a new axis has to be created. The axis is inserted into the corresponding branching point of the mother unit. The new axis also gets inserted the daughter segment. If a new axis is inserted to a branching point, *LIGNUM* makes sure that a terminating bud is put to the end of a list (using the function InsertTerminatingBud(BranchingPoint<TS,BUD>& bp)).

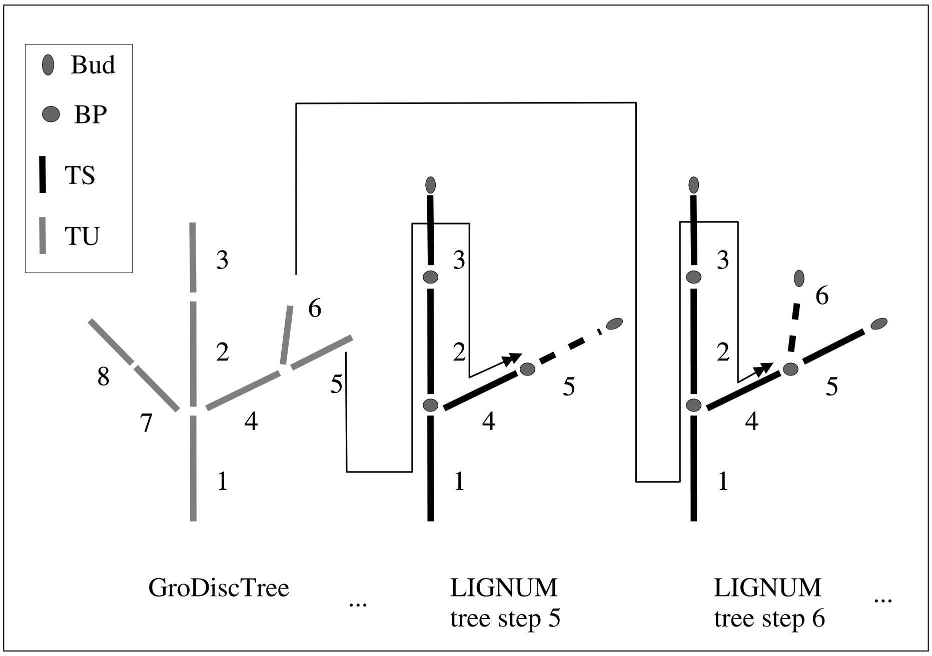


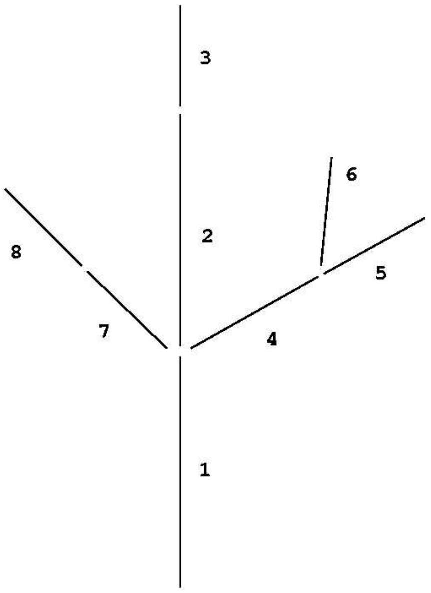
Figure 16: The translation process of the *GroDisc LIGNUM* interface. TU = *GroDisc* tree unit, TS = *LIGNUM* tree segment, BP = branching point.

3.3.6 Interface to *AMAPmod*

One of the objectives of this work has been to analyse virtual Scots pine crowns simulated by the *LIGNUM* software. *LIGNUM* itself does not have any statistical analysis tools.

There are two ways of solving this problem: either programming statistical software and integrating it into *LIGNUM* or using existing software. With *GROGRA* (Kurth 1994) and *AMAPmod* (Godin et al. 1998) software is available that provides useful tools for analysing tree structures. For this reason it would not have been efficient to develop a new program code. However, to take *GROGRA* and *AMAPmod* into use for that purpose, it was necessary to create possibilities to exchange data.

This chapter characterizes an interface between *GroDisc* and the *MTG* multiscaled architecture description of *AMAPmod*. This interface is used as an opportunity to show the difference between the procedural way of programming translation algorithms and the algorithm concept of the *STL*. Two implementations are existing. One is a simple function which is adapted from the *GROGRA* software and the other is a functor which is applied by an accumulate algorithm. In the second implementation the *LIGNUM* interface is applied. The procedural function was used in Dzierzon et al. (2003). The text is cited in a slightly changed version.



Order1	**Order2**	**Order3**	**Number**
/P1			
ˆ<U1/I1			1
	+U1/I1		4
		+U1/I1	6
	ˆ<U2/I1		5
	+U2/I1		7
	ˆ<U3/I1		8
ˆ<U2/I1			2
ˆ<U3/I1			3

Figure 17: A symbolic tree (right side) and its translation to an *MTG* structure (left side).

In order to automatically generate an *MTG* description of a tree in the *AMAPmod* coding language of Godin et al. (1998), the tree is traversed in depth-first order (cf. Grimaldi 1989). Figure 17 shows a simple example of a virtual tree and its translation to a *MTG* structure. The example tree has three different Gravelius orders (0 to 2). In the associated table each column represents a Gravelius order. An elementary unit is denoted by an **U** followed by an identifying number. Every unit is encoded by a string. Taking unit six as an example the string code is: ”/P1<U1/I1+U1/I1+U1/I1”, for the unit four the complete code is: ”/P1<U1/I1+U1/I1”. Comparing both examples one can easily see that some code is repeated. To avoid this repetition the symbol ”ˆ”is used. The symbol ’<’ signifies that the axis is continued, whereby branching is represented by ’+’.

The procedural algorithm which translates the *GroDisc* tree into an *MTG* description is adapted from the *GROGRA* software. In *GROGRA* a similar algorithm was already used to encode a tree in a format suitable for numerical simulation of water flow in the branching system of the crown (Früh and Kurth 1999). Each tree segment met during this search is marked to prevent double counting and the corresponding data are written into the *MTG* code file, including the required labels (indices) of growth units and internodes and their spatial coordinates. Currently the data interface works with only three levels of resolution (internode, growth unit and whole plant), but for most practical applications concerning forest trees this number of scales has turned out to be sufficient. For trees generated by *LIGNUM*, the number of scales is actually reduced to two, because each segment is treated as one growth unit consisting of one single internode. The notions ”growth unit” and ”internode” have lost their botanical meaning in this connection, because *LIGNUM* itself does not construct botanical internodes. The resulting code file contains the complete topological and geometrical information about all tree segments (cf. Dzierzon et al. 2003).

The last remarks concerning the translated scale information are also valid for the *C++* solution of the interface. The reimplementation can be seen as a direct interface between a *LIGNUM* tree and the *MTG* file description. A *LIGNUM* tree is used to take advantage of the *LIGNUM* data structure. A functor for a *LIGNUM* accumulate algorithm is created which is given in the appendix on page 146. The result of the process is written to a string. Important for the writing process is the correct order of the units. Figure 17 shows the correct sequence by number (1,4,6,5,7,8,2,3). That is exactly the way how a *LIGNUM* accumulate process handles the order. What remains to be done by the functor is to query the order, the number and the attributes out of a reference tree segment and write it to the resulting string. However, before the writing process starts, the identifiers have to be calculated. Identifiers are ascending numbers within an axis. The start number depends on the amount of emerging axes at a branching point. The result is returned to a string which is written to an *MTG* file.

Figure 18 illustrates the interface. The shown tree is a measured Scots pine from the region of Göttingen, Germany. The tree is translated from a *DTG* to an *MTG* tree using the *GroDisc* connection and visualized by the *AMAPmod* software.

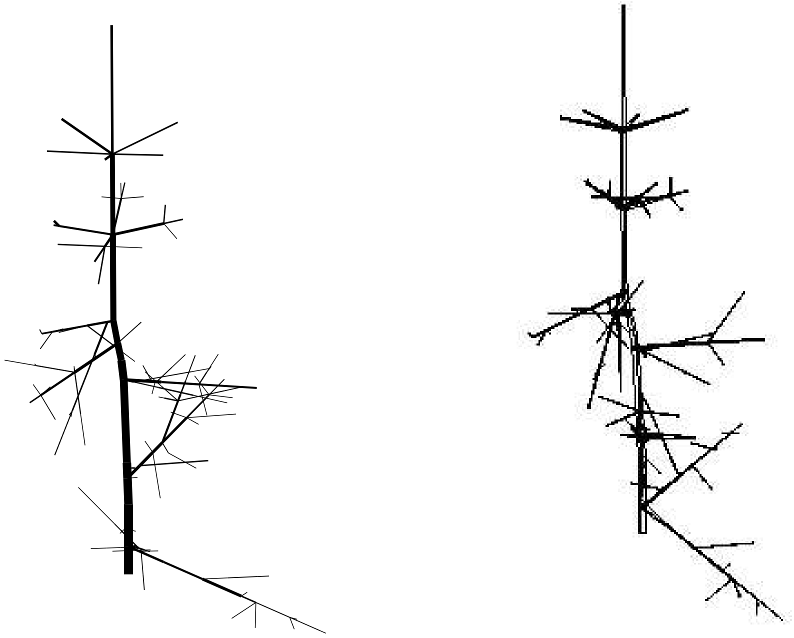


Figure 18: An example for the *GroDisc AMAPmod* interface. Left: the original tree. Right: the *MTG* equivalent visualized by *AMAPmod*.

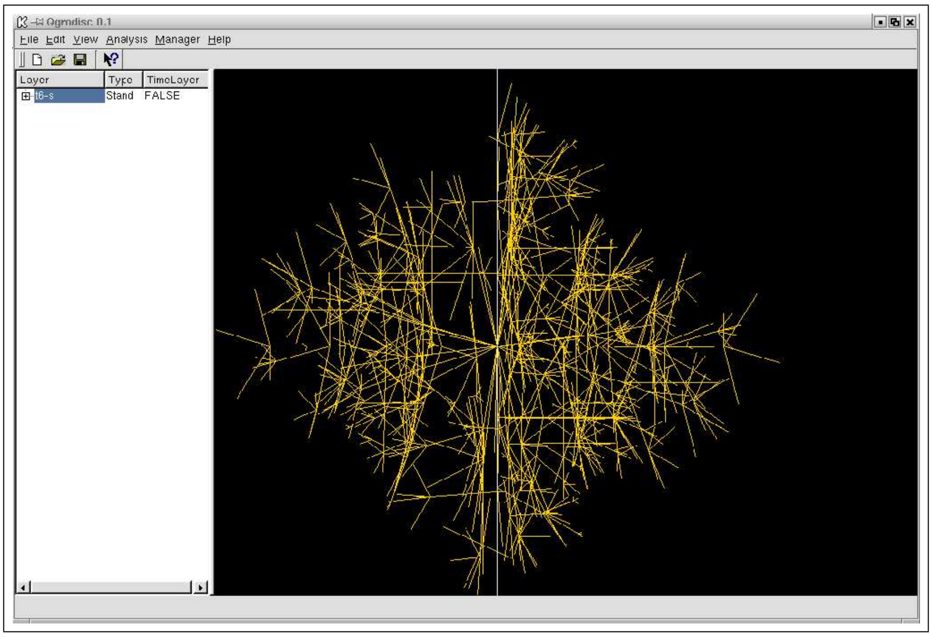
3.3.7 Interface to *R*

Another, more general statistical tool is the software *R*. It is a command line tool with a large statistical library and data management. The interface is aided by a data base connection. A direct translation would be possible by integrating the software *GroDisc* directly into *R*. However, the data interfacing would be extraordinarily complex and storage consuming. The piping of data through a data base was the better solution. *GroDisc* transports selected layer compartments and their attributes to a *MySQL* data base. The *RMySQL* extension of *R* then allows to import the data and opens the possibility to process the data statistically. Examples are given below in the analysis chapters.

3.3.8 User interface

GroDisc is a program which is as well used in batch mode as with a user interface. For the interaction mode an interface to the user is needed. The guidelines for a user interface are always similar. The working group has to be able to use the program without going to a *GroDisc* seminar. A graphical user interface *GUI* is helpful to accomplish simple use. Another guideline was to create software which is at least theoretically platform independent. Several tools are existing for programming platform independent *GUIs*.

The decision was in favour of the *Qt* libraries developed by *TROLLTECH*. The *Qt* Free Edition (*X11*) is provided under both the Q Public License ("QPL") and the *GPL*.



Layer	Type	TimeLayer
t6-s	Stand	FALSE

Figure 19: Screenshot of the graphical user interface of *GroDisc*.

It is possible to access the tools *R* and *AMAPmod* directly through the user interface. An editor is existing which allows to edit *R* and *AMAPmod* script code. *GroDisc* starts the corresponding process and pipes the code to the software by a file connection. Figure 19 shows a screen shot of the user interface. *GroDisc* is at the moment implemented on a Linux SuSE 7.0 and 7.2. The window classes are created with the *Qt* version 3.1. The graphics are implemented using *OpenGL*® .

4 Motivation for the reconstruction and models

The following chapters are dedicated to the reconstruction of an aged Scots pine (*Pinus sylvestris* L.) stand in Syke near Bremen and to a model for juvenile Scots pines growing in the understorey of a gap as an aggregation of the *FSTM LIGNUM*.

In general the task was to show how L-systems can be used in the scope of individual based tree stand modelling like they are used in *SILVA* (Pretzsch 1992a, 2001) or *BWIN* (Nagel et al. 2000). Within these models the crowns of the individuals are represented in a simplified manner by simple geometrical bodies like e.g. spheres or irregular pyramids. Those models are usually implemented by using the procedural or object oriented programming paradigm. In this work it is not tried to implement a comprehensive model like *SILVA* or *BWIN*. The task is to exemplify the use of rule based language in this scope.

The reconstruction of the Scots pine stand in particular seizes an approach of Pretzsch (1992a, 2001) to exemplify the use of L-systems in the scope of individual based modelling with simplified crown structures. He postulated a distance sensitivity between Norway spruce crowns. The reconstruction of the Scots pine stand will show how the Scots pine crowns behave and how the concept of distance sensitivity can be integrated into an L-system.

The creation of a model for juvenile Scots pines also exemplifies the use of L-systems in the scope of individual based tree stand modelling. Here, the focus lies on the question if it is possible to aggregate an *FSTM* to simulate juvenile trees and thus to obtain a model which is much more efficient in calculation time and at the same time more transparent than the original *FSTM*. The creation of juvenile tree stand models as a purely empirical approach is very difficult. A reduction of a model which tries to consider causalities might result in a less precise but more general model of the growth of juvenile Scots pines. Because of the complexity and computer resource consuming nature of *LIGNUM* we tried to aggregate the behaviour of *LIGNUM* trees which are growing in the understorey of a gap. For this reason the light model of *LIGNUM* had to be extended. The further implementation of this model is achieved by using an L-system. The use of the L-system was expected to show how approaches like the use of a light model to control growth can be integrated into an L-system. Characteristic here is the question how to represent a canopy which is formed by an aged tree stand and which is disturbed by a gap. The other question was how a canopy reduces the available light. The motivation was to show that components like light models or canopies can be expressed by rule based languages.

5 Reconstruction of a Scots pine stand

The following section describes the reconstruction of a Scots pine stand near Bremen. The motivation for the reconstruction has been given in the last chapter. The reconstruction is based on the hypothesis that the parameters height, diameter and height of the crown base are related to the crown extension, whereas the crown extension depends on the distance sensitivity towards its next neighbour. This hypothesis will be investigated by correlation and regression analysis. The results of the reconstruction will be integrated into an L-system. This L-system contains a sensitive function which represents distance sensitivity between crowns (function 14, see chapter 2.4). The results of further simulation runs are analysed and compared with the real data to assess the quality of the simulation run.

5.1 Material and methods

The following chapters are dedicated to the description of the measured tree stand and the methods which were used for collecting data. The targets of the measurement are characterized. In order to get a better impression of the conditions of the environment in which the stand grows a brief description of the site is given.

5.1.1 Study object

For the collection of the data required to examine the hypothesis a Scots pine tree stand in Syke next to Bremen (northern Germany) was chosen. The main advantage of this stand was that it has been previously measured by Duchiron (2003) in 1995. Figure 20 shows on a map where the site is located and a position plot of the sample trees. The sample plot is partly surrounded by the appropriate stand. The western side borders to a road, hence it is an open side.

The stand was established in 1950 using classical planting methods, hence the age of the stand was 51 and 57 years in 1995 and 2001. The stand is a pure Scots pine stand. It was treated in classical way during the grow up. However, the exact development from 1950 to 1995 is unclear. During the period from 1995 to 2001 the stand was thinned in 1998. The fact that the stand was thinned does indeed complicate the interpretation of the analysis of growth which took place during that period.

Within this work models are used which are developed for Scots pines of southern Finland, hence for the better interpretation of the growth behaviour it is useful to describe the climate conditions in which the trees grew. The stand is located within the growth zone of “mean and western lowlands of lower saxony”. Table 7 gives the main climate characteristics of the growth region of “mean Geest area” which is part of the mentioned growth zone. The mean Geest area is mainly influenced by the Atlantic climate. The region is near the coastal area of northern Germany but the influence of wind and reduced precipitation compared to the coastal area characterizes this area. Distinctive early and late frosts are typical for this region. The soil is mainly influenced by the glacier period in northern Germany. The soil constitution is prevailed by sand, hence the types of soils are mostly pure soils from the podsol range.

abs. altitude	**mean air temperature**		**mean precipitation**		**mean draught index**
[m]	year [$°C$]	May-Sept. [$°C$]	year [mm]	May-Sept. [mm]	year
10-70	8.3	14.2	730	350	36

Table 7: Climate characteristics in the area of the sample plot (Arbeitskreis Standortskartierung 1985).

5.1.2 Study parameters and measurement methods

For measuring stand and tree parameters many methods are existing. Table 8 gives an overview about the measured parameters and their precision. Breast height diameters (DBH) are classically measured with a calliper. The DBH is defined as the diameter of the tree main stem in a height of 1.3 meters. The crown base and the height of the trees are measured by a “Blume-Leis” tool. The crown base is defined as the height of the first living branch. A branch is supposed to be alive if it possesses green foliage (needles) (cf. Pretzsch 1992a). Some problems arose during the measurement of the heights. Chapter 5.1.3 discusses these problems considering the measurement results. The crown extension is measured using a so called “crown mirror”. This tool allows to plumb the extension of the crown. The extensions are measured in eight fixed directions distributed regularly over the orientations. Duchiron (2003) used a “Top con” to determine the positions. She marked the trees with paper board signs. These signs had unfortunately been removed by playing children, hence a reconstruction of the positions was necessary. The reconstruction was supported by the *GIS ArcView* 3.2a and *GroDisc*. *GroDisc* calculated the distances and azimuths of the next neighbours of each tree. The *GIS* produced a positional plot of the trees. The calculated data from *GroDisc* allowed to verify the reconstruction.

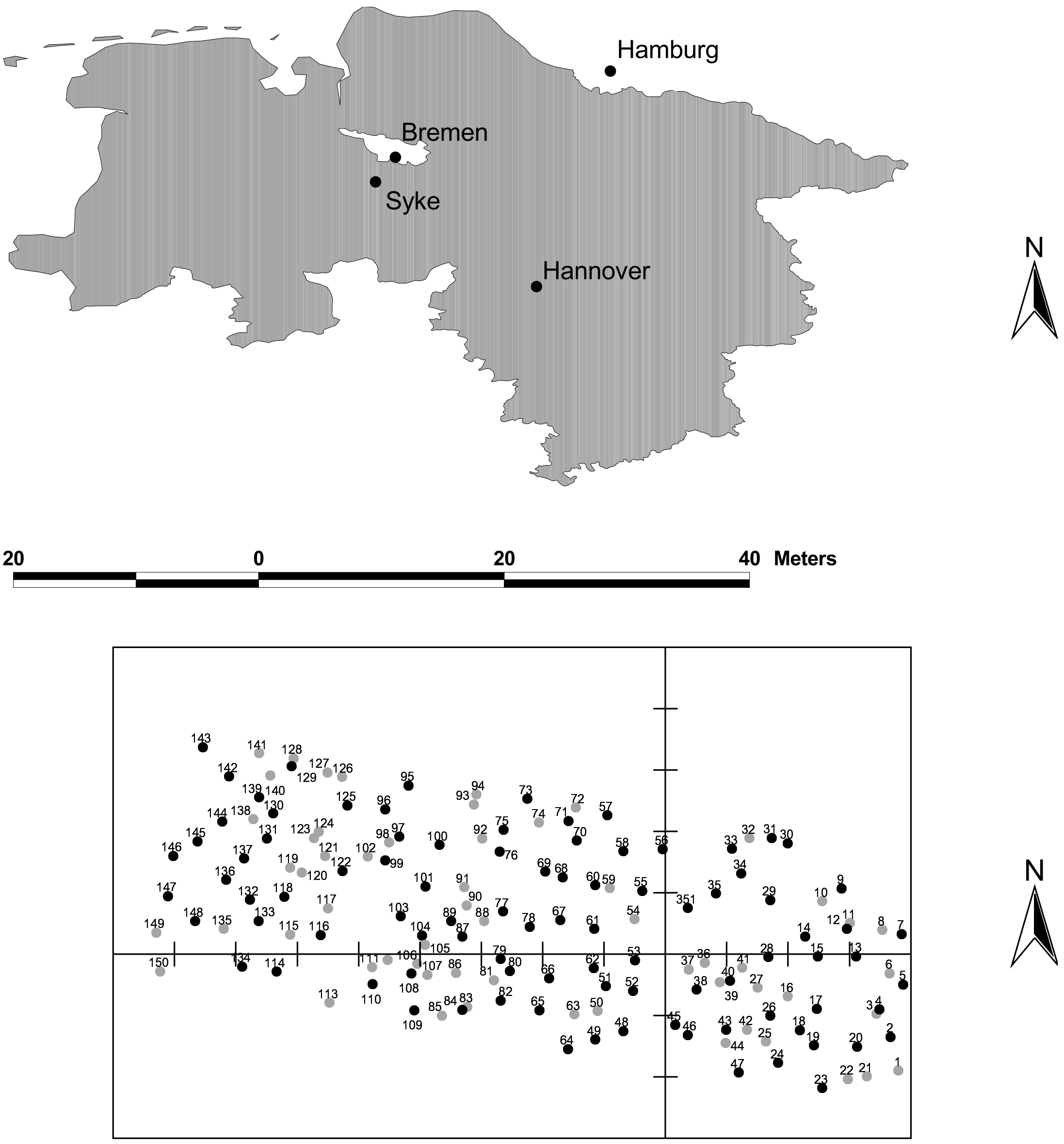


Figure 20: The location of the stand in northern Germany (above) and the positions of the trees (below). The black points denote living, the gray points logged trees.

Parameter	**Unit**	**Precision**
DBH	cm	1 mm
Height (H)	m	1 cm
Crown base (CB)	m	1 cm
Crown extension (CE)	m	1 cm
Position	m	1 cm

Table 8: Measured parameters in 1995 and 2001. The positions had to be reconstructed in 2001.

5.1.3 General statistics

The purpose of measuring a complete tree stand was to get an idea of the growth behaviour of Scots pine trees within a stand. This chapter summarizes the results of these measurements. The presentation of these results also serves to create a better idea of the stand. Tables 9, 10 and 11 give an overview about the results of measuring. Some parameters were not measured but calculated: $\overline{G}$ is the mean basal area of the stand. D_g is the quadratic mean diameter (cf. equation 2), the diameter which results from the mean basal area. D_h also is a quadratic diameter but is calculated for a subset of the trees. Only the strongest trees are taken into account. The strongest trees are defined as upper height quintile of the set of trees. The values for $\overline{CE}$ are displayed in an aggregated form. The values are averaged over the eight directions.

$$D_g = \sqrt{\frac{4}{\pi} \cdot \overline{G}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{2}$$

Variable	**Unit**	**N**	**Mean**	**Std**	**Min**	**Max**
$\overline{D}$	$[cm]$	151	18.61	4.15	10.3	31.3
D_g	$[cm]$	151	19.07	-	-	-
D_h	$[cm]$	151	20.61	-	-	-
$\overline{G}$	$[cm^2]$	151	285.58	127.80	83.32	769.45
$\overline{H}$	$[m]$	151	17.99	2.05	11.50	24.00
$\overline{CB}$	$[m]$	151	12.26	1.53	0.00	15.00
$\overline{CE}$	$[m]$	151	1.51	0.56	0.00	2.90

Table 9: Characteristics of the stand in 1995. Crown extensions are averaged per tree. $\overline{D}$ = mean DBH, D_g = quadratic mean diameter, $\overline{G}$ = man basal area, $\overline{CE}$ = mean crown extension, $\overline{CB}$ = mean crown base, $\overline{H}$ = mean height.

The values for the situations in 1995 and 2001 are common values for a Scots pine stand. The data show that the tree stand is very homogeneous. The diameters and heights are normally distributed. This fact also shows the homogeneity. A so called "Plenter forest" for example does not follow normal distribution. The distributions are shown in figure 21. The only exceptions are the increments of the crown bases: These cannot be seen as a normal distribution.

Variable	**Unit**	**N**	**Mean**	**Std**	**Min**	**Max**
$\overline{D}$	$[cm]$	94	23.34	4.28	14.80	36.00
D_g	$[cm]$	94	23.73	-	-	-
D_h	$[cm]$	94	25.83	-	-	-
$\overline{G}$	$[cm^2]$	94	442.16	163.19	1017.88	172.03
$\overline{H}$	$[m]$	94	20.04	1.88	16.00	24.50
$\overline{CB}$	$[m]$	94	13.86	1.40	10.50	17.50
$\overline{CE}$	$[m]$	94	2.28	0.60	1.20	4.70

Table 10: Characteristics of the stand in 2001. Crown extensions are averaged per tree. $\overline{D}$ = mean DBH, D_g = quadratic mean diameter, $\overline{G}$ = man basal area, $\overline{CE}$ = mean crown extension, $\overline{CB}$ = mean crown base, $\overline{H}$ = mean height.

Variable	**Unit**	**N**	**Mean**	**Std**	**Min**	**Max**
$\overline{D}$	$[cm]$	94	2.87	1.13	0.60	7.00
D_g	$[cm]$	94	4.66	-	-	-
D_h	$[cm]$	94	4.35	-	-	-
$\overline{G}$	$[cm^2]$	94	2.26	0.88	0.47	5.50
$\overline{H}$	$[m]$	94	1.73	1.72	-2.50	6.00
$\overline{CB}$	$[m]$	94	1.52	1.29	0.00	6.50
$\overline{CE}$	$[m]$	94	0.53	0.57	1.20	4.70

Table 11: Characteristics of the stand of the increments. Crown extensions are averaged per tree. $\overline{D}$ = mean DBH, D_g = quadratic mean diameter, $\overline{G}$ = man basal area, $\overline{CE}$ = mean crown extension, $\overline{CB}$ = mean crown base, $\overline{H}$ = mean height.

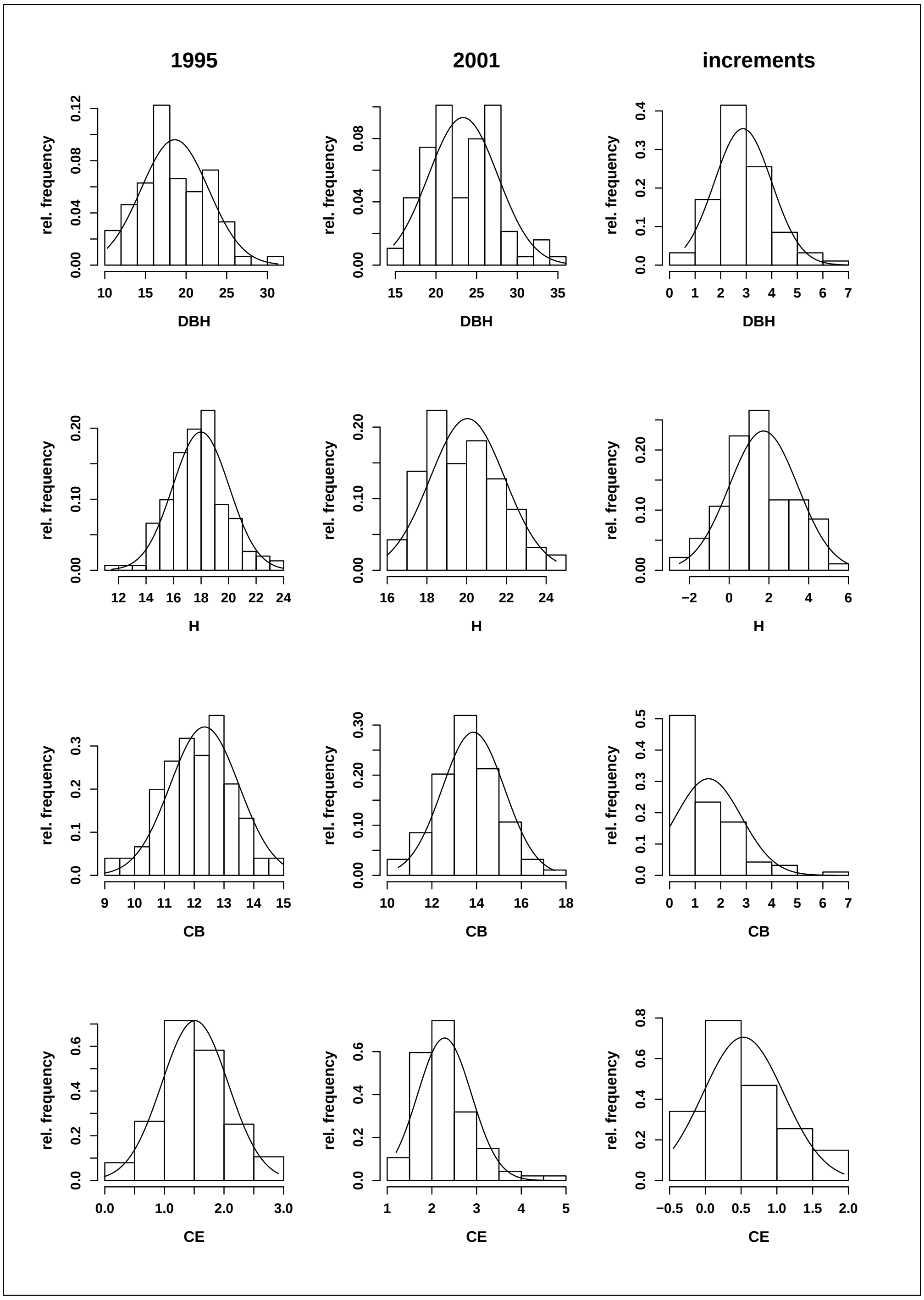


Figure 21: The distribution plots. CE = mean crown extension, CB = crown base, H = height. Smooth lines = best-fit normal distributions.

However, the calculated increments which are also shown in figure 21 have to be commented. It meets the eye that the precision of the increments is low. The standard deviations are larger than in the static situations. The increments of tree heights even show abnormal minimum values: some are negative. It is not possible to explain these values in a certain way. However, there are some ideas about it. The first assumption is that some tree crowns were damaged during thinning in 1998. This at least is the impression we had when the stand was firstly examined in 2001. These damages could indeed explain some negative increments. Another possible explanation could lie in the method of measuring. A typical but intractable error is to mislocate the top of the crown. The repetition of height measurements is also susceptible to misdetermination caused by different positions of the assistant during the measurement process. The data of Duchiron (2003) do not contain any information about these positions. In the author's opinion this could be one of the favoured reasons why the precision of height measurements was affected.

5.1.4 Correlations and regressions

To repeat the hypothesis: The assumption was that the diameter of a tree depends on the extension of the crown. The parameters height and crown base can then be seen as depending on the DBH. Furthermore the assumption includes that these statistical relationships are also valid for the growth, that is the increment during the period that is considered.

As a first step for proving this hypothesis correlation analyses are performed. In an advanced step a regression analysis is accomplished. Tables 12, 13 and 14 show the correlation matrices between the measured values. The correlations show that the assumption concerning the situations in 1995 and 2001 can be confirmed except the relation between the diameter and crown base. Especially the values for 2001 do not show any significance.

However, the increments do not show any reasonable coherences. The regression analysis shows this fact even clearer. The static situations in 1995 and 2001 show the expected coherences whereas the regression analyses concerning the increments do not show any significance. Table 15 and figure 22 illustrates the correlations and regressions.

Based on those results it is not possible to reconstruct the growth of the trees using the crown extensions as the calculation basis for all other parameters. An alternative possibility is to use the initial situation of 1995 to calculate the mean annual increment for each parameter. This approach is less general but usually generates reasonable results. Table 16 and figure 23 show this regression analysis. All regressions are significant but the coefficient of determination (R^2) does not exceed 0.17.

1995	Unit	DBH	H	CB	CE
DBH	$[cm]$	1.00	0.41	0.23	0.70
H	$[m]$	0.41	1.00	0.49	0.27
CB	$[m]$	0.23	0.49	1.00	0.10
CE	$[m]$	0.70	0.27	0.10	1.00

Table 12: Correlation matrix for the year 1995. CE = mean crown extension, CB = crown base, H = height.

2001	Unit	DBH	H	CB	CE
DBH	$[cm]$	1.00	0.48	-0.03	0.66
H	$[m]$	0.48	1.00	0.34	0.19
CB	$[m]$	-0.03	0.34	1.00	-0.06
CE	$[m]$	0.66	0.19	-0.06	1.00

Table 13: Correlation matrix for the year 2001. CE = mean crown extension, CB = crown base, H = height.

inc.	Unit	DBH	H	CB	CE
DBH	$[cm]$	1.00	0.13	-0.27	0.00
H	$[m]$	0.13	1.00	0.24	0.21
CB	$[m]$	-0.27	0.24	1.00	0.15
CE	$[m]$	0.00	0.21	0.15	1.00

Table 14: Correlation matrix for the increments. CE = mean crown extension, CB = crown base, H = height.

Variable			**1995**			**2001**			**inc.**		
			N	R^2	p-value	N	R^2	p-value	N	R^2	p-value
CE $[m]$	vs.	DBH $[cm]$	151	0.4855	<0.0001	94	0.4362	<0.0001	94	0.0000	0.9968
DBH $[cm]$	vs.	H $[m]$	151	0.1641	<0.0001	94	0.2278	<0.0001	94	0.0520	0.2249
DBH $[cm]$	vs.	CB $[m]$	151	0.0314	0.02957	94	0.0008	0.7898	94	0.0732	0.0084
H $[m]$	vs.	CB $[m]$	151	0.2764	<0.0001	94	0.1152	0.0008	94	0.0588	0.0186

Table 15: Linear regressions and the significance of the slope (p-value). This table corresponds to figure 22. CE = mean crown extension, CB = crown base, H = height.

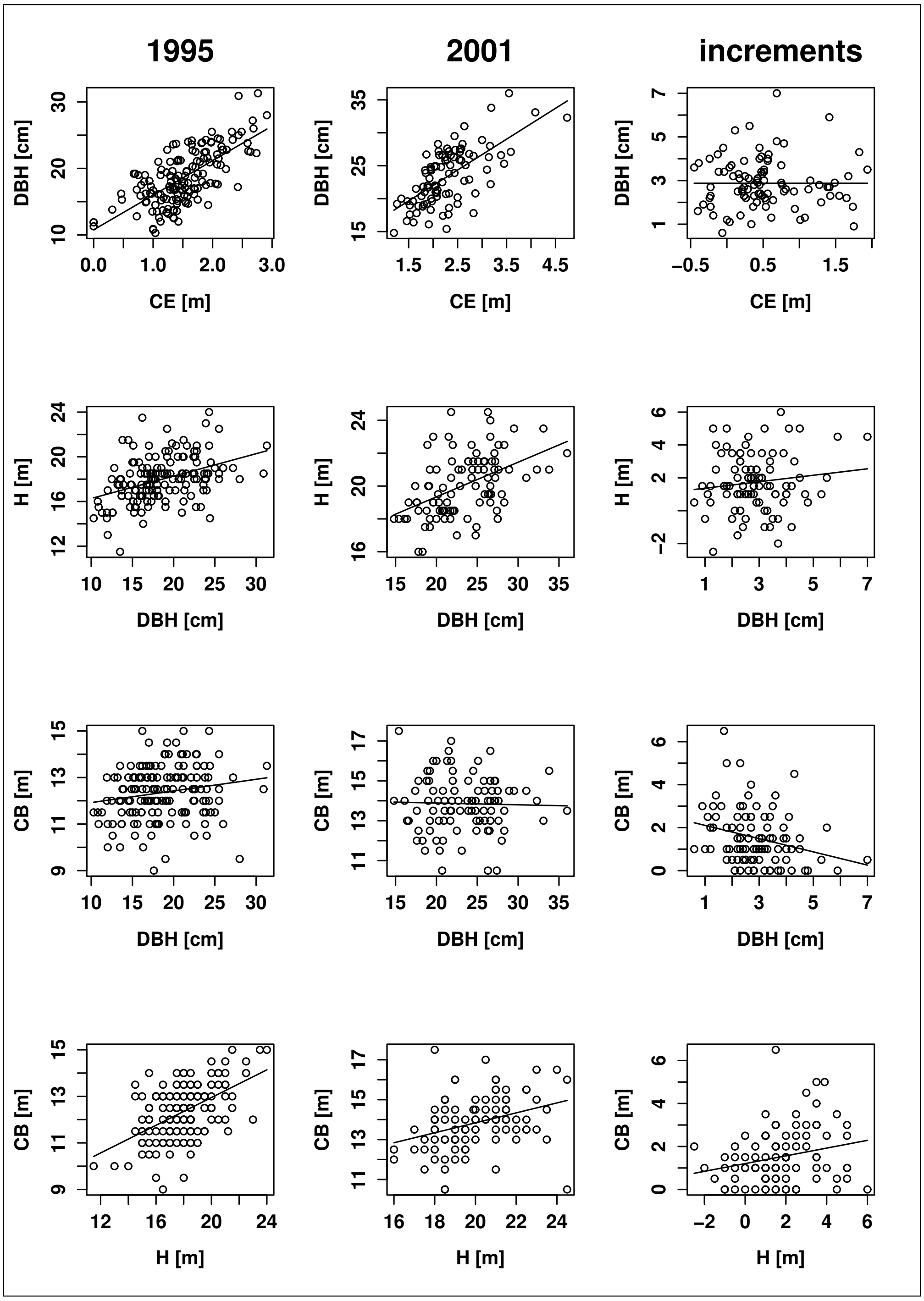


Figure 22: The correlation plots. CE = mean crown extension, CB = crown base, H = height. Smooth line = best-fit linear regression.

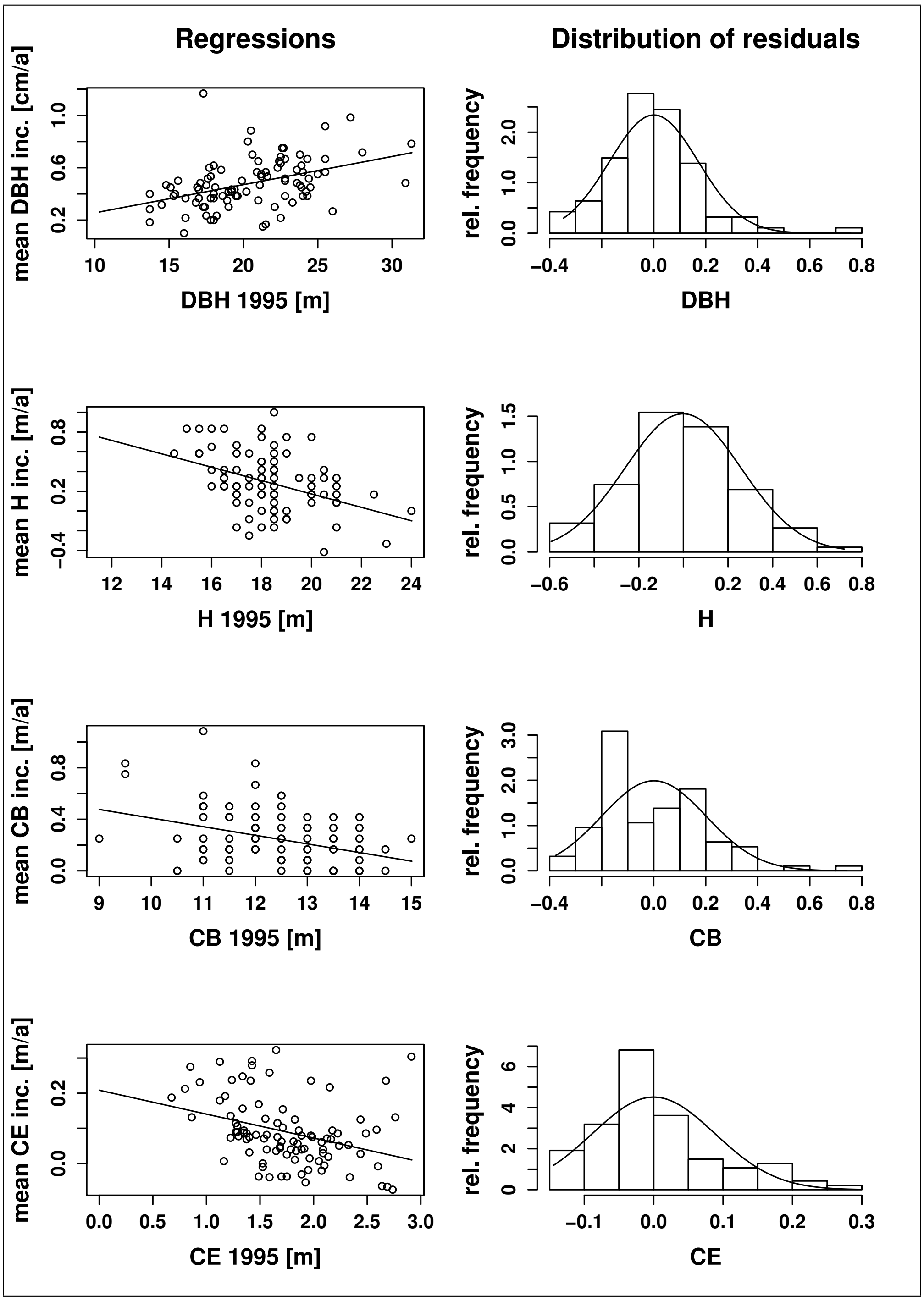


Figure 23: The correlation and regression plots of 1995, 2001 and the increments. CE = mean crown extension, CB = crown base, H = height. Smooth lines left plots = best-fit linear regressions. Smooth lines right plots = best-fit normal distribution.

Variable			N	Intercept	Slope	R^2	p-value	sd_{res}
DBH $[cm]$	vs.	ΔDBH $[cm/a]$	94	0.041	0.021	0.18	<0.0001	0.170
H $[m]$	vs.	ΔH $[m/a]$	94	1.528	-0.068	0.17	<0.0001	0.261
CB $[m]$	vs.	ΔCB $[m/a]$	94	1.077	-0.067	0.13	0.0003	0.200
CE $[m]$	vs.	ΔCE $[m/a]$	94	0.208	-0.068	0.12	0.0006	0.088

Table 16: Linear regressions and significance of the slope (p-value). This table corresponds to figure 23. DBH = breast height diameter, H = height, CB = crown base and CE = mean crown extension.

5.1.5 Analysis of distance sensitivity

One part of the reconstruction was the analysis of distance sensitivity of Scots pine crowns. The term distance is used analogously to the distance calculation of the sensitive L-system function 14 of *GROGRA* (cf. chapter 2.4). It means the distance of a node of the base unit of the irregular pyramid to the surface of the next neighbour. These distances are calculated by *GroDisc* for each tree and direction. The idea behind is to use this data to adapt the model of Pretzsch (1992a) to the Scots pine stand. Table 17 gives the results of the calculations for the years 1995 and 2001. The values are analogously to the crown radii averaged over the eight directions, analogously to the crown radii. The Student t-Test shows that these averages do not significantly differ between 1995 and 2001 (cf. table 18). Figure 24 additionally shows the distributions of the distances.

Pretzsch's approach is based on a correlation between the crown radii and the mean annual increment of those crowns which are not hindered by their neighbours (potential radius increment). A tree crown is not hindered if the distance to the next neighbour is greater than one meter (cf. equation 3) (Pretzsch 1992a). This behaviour could be reconstructed for the mean distances as the upper plot in figure 25 illustrates. The correlation and regression respectively is significant. The second part of Pretzsch's approach contains a coherence between the distances and the relative crown radii increments (cf. equation 4). In the end one obtains a reduction factor of the potential radii increment. This regression is nonlinear and results in decrements of the radii if crowns overlap. This regression could not be reproduced. The lower plot in figure 25 and table 19 show the best-fit function and the proper scatter plot. A more intensive statistical analysis was not carried out because the plot further shows that the data do not allow to infer any reduction behaviour in case of overlapping at all.

Nevertheless, the distances which are presented in table 17 show that the trees at least stop growing if the degree of overlapping exceeds a certain distance. Regarding the combined data (2001 and 1995) the trees show a maximum degree of overlapping of -0.86 meters if one sees the minimum distance as the mean value minus the standard deviation.

$$zkr_{pot} = a \cdot zkr + b, \ \forall \ zkr \text{ where } dist > 1m \tag{3}$$

$$zkr_{rel} = 1 - e^{-a \cdot (dist+b)} \tag{4}$$

a, b = regression parameters,

zkr = crown radius,

zkr_{pot} = potential crown radii increment ($dist > 1m$),

zkr_{rel} = crown radii increment relative to zkr_{pot},

$dist$ = distance to surface of next neighbour.

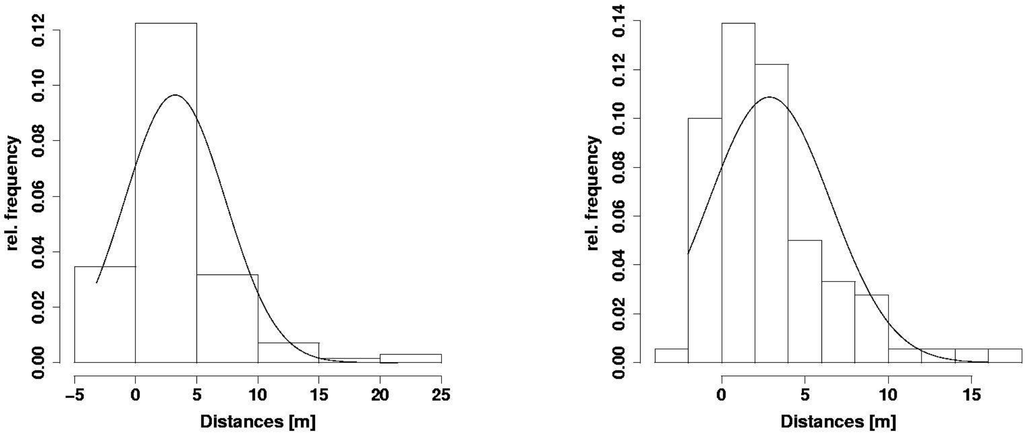


Figure 24: Results of the calculation of distances in 1995 (left side) and 2001 (right side). Smooth line = best-fit of normal distribution.

5.2 Reconstruction of the stand

The statistical results have shown no relation which would be usable within a model for a single tree. Only a simple form of interaction could be shown. Thus the growth of trees is depicted by a simple reconstruction of the mean growth during the five years period between 1995 and 2001. This chapter shows how this reconstruction is implemented.

The model for a single tree is due to the design of Pretzsch (1992a) for modelling Norway spruce (*Picea abies* (L.)) within stands. He designed the crown shape in a very

Year	**Unit**	**N**	**Mean**	**Std**	**Min**	**Max**
1995	$[m]$	115	3.23	4.13	-3.2	24.6
2001	$[m]$	71	2.88	3.67	-2.0	17.8
comb.	$[m]$	286	3.09	3.95	-3.2	25.6

Table 17: Results of the calculation of distances in 1995 and 2001 and both combined.

Variable	**Mean 1995**	**Mean 2001**	**t**	**df**	**p-value**
DIST	3.23	2.88	0.6699	205.747	0.5037

Table 18: Results of the t-Test between the distances in 1995 and 2001. DIST = Distance to the surface of the next crown.

Function	**Unit**	**a**	**b**	**R^2**	**p-value**
$zkr_{pot} = a \cdot zkr + b$	$[m]$	-0.079	0.223	0.183	0.0005
$zkr_{rel} = 1 - e^{-a \cdot (dist+b)}$	$[m]$	2.22	0.99	-	-

Table 19: Results of the regression after Pretzsch (1992a). The notation is the same as in the publication of Pretzsch. zkr = crown radius, zkr_{pot} = potential crown radii increment, zkr_{rel} = relative crown radii increment to zkr_{pot}, $dist$ = distance to surface of next neighbour and a and b are regression parameters.

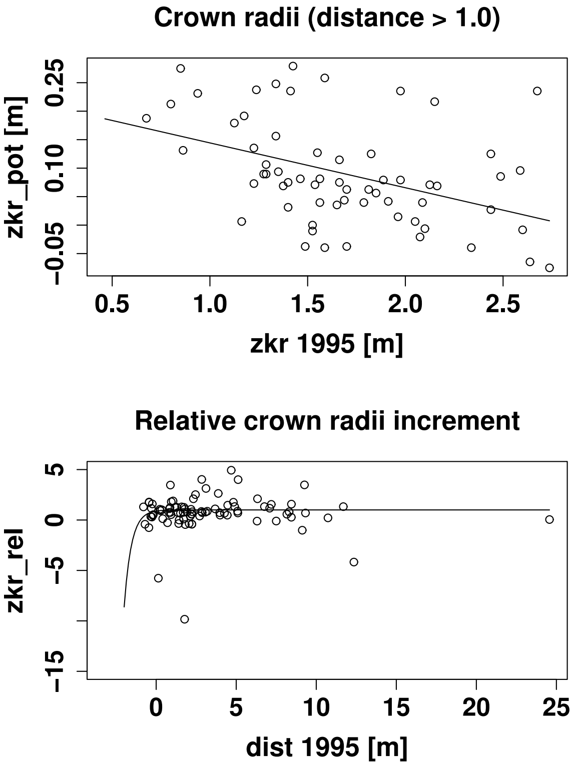


Figure 25: Results of the regression after Pretzsch (1992a). The notation is the same as in the publication of Pretzsch. zkr = crown radius, zkr_{pot} = potential crown radii increment, zkr_{rel} = relative crown radii increment to zkr_{pot} and $dist$ = distance to surface of next neighbour. Smooth lines upper plot = best-fit linear regression. Smooth line lower plot = best-fit of function $zkr_{rel} = 1 - e^{-a \cdot (dist+b)}$.

simple way: The stem is a cylinder and the crown is represented by an irregular pyramid. The base of the pyramid has eight corners. We call the horizontal lines from the stem to the corners the base elements of the crown. Their orientation is fixed. One base element is oriented exactly towards the north. The others align in angles of 45 degrees respectively (cf. figure 4 and figure 5). An elementary unit is existing which start point is at the crown base and the end point at the top of the crown. This unit is called central unit of the crown and denoted the crown height.

In the model, each base element is able to grow, which affects the extension of the crown in its pyramidal representation. The state at the moment t is based on the coherence between an initial situation (t_0) and the mean annual increment (t_i). The dispersion around the regression line is quite large. The scattering is thus included in the simulation. The residuals of the dispersion are normally distributed. Equation 5 and equation 6 give an example, determine the diameter at timestep t. The mean annual diameter increment (d_i) is calculated from the initial diameter (d_0) and a correcting summand. The correcting summand represents scattering with the given standard deviation.

$$\begin{aligned} d_i &= 0.021 * d_0 + 0.041 + d_r && (5) \\ d_t &= d_0 + t * d_i && (6) \end{aligned}$$

d_i	=	diameter increment $[cm/a]$,
d_0	=	initial diameter in 1995 $[cm]$,
d_r	=	residuals $[cm/a]$, random numbers,
t	=	time $[a]$,
d_t	=	diameter at timestep t $[cm]$.

The reconstruction is implemented using an L-system. The complete L-system is presented in the appendix. Here only the most important components are explained. The global and local variables, the global constants and the symbols are shown in the Tables 20 to 23. Two extensions are used here: The turtle command T and the sensitive function 14. Additional functionality of *GROGRA* is used: array variables, sensitivity and stochastic modelling (Kurth and Anzola Jürgenson (1997), Kurth and Sloboda (1999a), Kurth and Sloboda (1999b)). First the initial situation has to be set. This is done by an array variable. Listing 10 gives its declaration. The meaning is: A variable *data* exists which is an array. Data are stored in the file *system*.a0**1**. The array has two dimensions and has 13 columns and 151 rows. The product of the number of columns and rows has to be equal to the file size.

Listing 10: Declaration of the array variable for data input

```
\var data array 1 2 13 151,
```

To calculate the distance to the crown surface of the next neighbour we need sensitivity. Listing 11 shows the declaration for function 14 in the L-System. This function achieves the calculation. The sensitive functions are implemented in *C*. The variable *dtt* (**d**istance **t**o **t**riangle) gets the value of the distance and can be used on the left side of the rule e.g within a condition. Function 14 has no arguments, hence the fourth part in the declaration equals zero. This value could be one or two if the function was able to handle arguments.

Listing 11: Declaration of the sensitive function which calculates the distance to the surface of the crown of the next neighbour.

```
\var dtt function 14 0,
```

For the reconstruction of the scattering around the linear regression a stochastic variable is needed. Listing 12 exemplifies the declaration of a stochastic variable. Here, the normally distributed residuals (*dres*) of the diameter regression are declared. *dres* will be the correction term in the simulation of mean annual increments of the diameter. The residuals are standardized to the mean value zero and the variance of 0.0017. When a rule containing *dres* in its right-hand side is applied, a pseudorandom number following this distribution is generated.

Listing 12: Declaration of a normally distributed stochastic variable

```
\var dres  normal 0 0.0017,
```

Listing 13 shows how the data of the initial situation in 1995 is added to the L-System. The declaration of the array variable *data* has already been described. Here, it is used by calling the variable. The first argument defines the row and the second the column of the array. The turtle gets the content of the variable for a forward movement without generating a unit. In this way the turtle is ordered to move to a specific tree position. The symbol *cut* is implemented to simulate the thinning in 1998. The first argument initializes the time step. The value of the second argument contains the information if the tree was harvested in 1998 or not. If yes, the symbol will be replaced by the *cut* operator % in another rule. The symbol *pinus* stands for the actual tree. X, Y and HVAR are constants which denote the columns in the array. X and Y are the position coordinates and the column of HVAR holds information about the trees which are harvested in 1998.

Listing 13: The generation of the initial positions in 1995.

```
stand → &(trees) < [ RU(90) f(data(i,X)) RL(−90) f(data(i,Y)) RU(−90)
   cut(0,data(i,HARV)) pinus(i) ] >  ,
```

The tree is decomposed into two elements: the main stem and the crown. Within the L-System (cf. listing 14) it is useful to represent these parts by symbols. The symbol *stem* stands for the main stem of the Scots pine. The representation of the Scots pine crown is more complex because a geometrical body of an irregular pyramid is not at hand in the *GROGRA* specification. Two symbols are existing which are part of the representation the crown: *crbase* (to represent the base units of the crown) and *crtop* which is a representation of the crown height (central unit of the crown). The rule which describes the *pinus* symbol contains a shortcut here ("..."). This part contains *C*-commands (cf. Kurth 1999) which are only significant for visualization. After the visualization part the triangles are created using the new turtle command *T*. This command uses the indices which are created by the command *S*. The triangles are needed for the representation of the irregular pyramid.

Listing 14: The generation of the initial trees in 1995. The complete rule is given in the appendix.

```
pinus(ind) → stem(data(ind,BHD),dres,0,data(ind,CB),cbres,0,0)
                  RH(135) RU−90 &(8) < [ RL(i*(−45))
                   crext(data(ind,KR+i),ceres,0,0)  S(i+9*ind) ] > RU(90)
                   crtop(data(ind,H)−data(ind,CB),chres,0,0) S(8+ind*9)
                   ...
                   &(7) < T(i+ind*9,(i+1)+ind*9, 8+ind*9) >
                  T(7+ind*9,ind*9,8+ind*9),
```

Symbol	**Meaning**
*	the axiom
stand	stand of reconstructed trees
pinus	representation of a single tree
stem	the main stem of a tree
crext	the crown base units of a tree (eight pieces)
crtop	the centre unit of the crown

Table 20: The symbols of the reconstruction implementation.

The geometrical implementation of these rules is split into two parts: generative rules for the calculation of the regression functions and the time and interpretive rules in which the turtle commands are implemented. Interpretive rules are distinguished from generative by the arrow symbol. Generative rules only have a single arrow (→) interpretive a double arrow (⇒). The crown extension rule contains a condition. If the function 14

name	**unit**	**spec.**	**meaning**
dtt	$[m]$	function	sensitive function which calculates the distance of a unit to the surface of a triangle
i	-	index	index variable for use within the scope of repetition operator
data	-	array	array for the input of the attributes of the initial situation in 1995
dres	$[m]$	normal	reconstruction of the normally distributed residuals of the diameters
ceres	$[m]$	normal	reconstruction of the normally distributed residuals of the mean crown extensions
chres	$[m]$	normal	reconstruction of the normally distributed residuals of the crown heights
ceres	$[m]$	normal	reconstruction of the normally distributed residuals of the crown extensions

Table 21: Global variables of the L-system implementation of the reconstruction of the Scots pine stand.

name	**unit**	**symbols**	**meaning**
ind	-	pinus	number of tree
t	$[a]$	stem, crbase, crtop	actual step of time
h	$[m]$	stem	actual height of tree
hr	$[m]$	stem	correction factor ($chres$)
hi	$[m/a]$	stem	annual increment of h
d	$[m]$	stem	actual diameter (d1.3) of tree
dr	$[m]$	stem	correction factor ($dres$)
di	$[m/a]$	stem	annual increment of d
cre	$[m]$	urstem, stem, crext, crtop	actual crown extension of the tree
crer	$[m]$	stem	correction factor ($creres$)
crei	$[m/a]$	stem	annual increment of cre
crt	$[m]$	urstem, stem, crext, crtop	actual crown height of the
crtr	$[m]$	stem	correction factor ($dres$)
crti	$[m/a]$	stem	annual increment of d

Table 22: Local variables used in the context of symbols within the L-system of the reconstruction of the Scots pine stand.

name	**unit**	**meaning**
trees	-	number of trees of the stand
X	-	column in *data* which contains the x-coordinates
Y	-	column in *data* which contains the y-coordinates
DBH	-	column in *data* which contains the DBHs
CB	-	column in *data* which contains the crown bases
CE	-	column in *data* which contains the crown extensions
DIES	-	column in *data* which contains if the tree is harvested in 1998
dslo	$[m/a]$	annual diameter increment
dint	$[m]$	intercept of regression
cbslo	$[m/a]$	annual increment of the crown base
cbint	$[m]$	intercept of regression
chslo	$[m/a]$	annual increment of the crown height
chint	$[m]$	intercept of regression
ceslo	$[m/a]$	annual increment of the crown extension
ceint	$[m]$	intercept of regression

Table 23: Global constants used in the context of symbols within the L-system of the reconstruction of the Scots pine stand.

(*dtt*) returns a value which exceeds the maximal degree of overlapping, rule two is applied which creates growth otherwise rule three will be evoked which stops the growth. The maximal degree of overlapping is adapted from the results of the distance sensitivity analysis: It is the mean distance minus the standard deviation of all distances.

The interpretive rules (cf. listing 16) create the geometrical structure which results from the regression parameters. Additionally the colour, diameter and needle attributes of a turtle step are set for the symbols *crext*, *crbase* and *crtop*. Three parameters for the turtle step are set: the colour, the needle parameter and the diameter. The colour and the needle parameter only have auxiliary function. These parameters are used in *GroDisc* to filter the crown base units out of the structure.

Listing 15: The generative rules for the growth of main stem and crown.

```
/*rule 1*/ stem(d,dr,di,cb,cbr,cbi,t) →
        stem(d,dr,(d*dslo+dint+dr),cb,cbr,(cb*cbslo+cbint+cbr),t+1),

/*rule 2*/ (dtt>(-0.86)) crext(ce,cer,cei,t) →
                        crext(ce,cer,(ce*ceslo+ceint+cer),t+1),

/*rule 3*/                crext(ce,cer,cei,t) →
                        crext(ce,cer,cei,t+1),
/*rule 4*/              crtop(ch,chr,chi,t)  →
                           crtop(ch,chr,chslo*ch+chint+chr,t+1),
```

Listing 16: The interpretative rules for the growth of main stem and crown.

```
/*rule 1*/  stem(d,dr,di,cb,cbr,cbi,t) ⇒
                                 P(7)  Dl(d+t*di) N(0) F(cb+t*cbi),
/*rule 2*/  crext(ce,cer,cei,t)          ⇒
                                 P(10) Dl(0.01)   N(1) F(ce+t*cei),
/*rule 3*/  crtop(ch,chr,chi,t)          ⇒
                                 P(11) Dl(0.01)   N(1) F(ch+t*chi),
```

5.3 Results

The simulation of the model covers a period of six years. This chapter shows the results of the simulation and the results of the comparative analysis between real and virtual stand in 2001.

Figure 26 illustrates the simulated stand in 2001. Equivalently to the presentation of the measurements the results of the simulation are given in table 24. Figure 27 graphically shows the distributions of the parameters. The table and figure do not directly answer the question if the reconstruction was successful. Therefore table 25 gives the results of a t-Test which was performed in order to analyse whether the mean values are significantly different or not. All simulated mean values are not significantly different.

Variable	Unit	N	Mean	Std	Min	Max
$\overline{D}$	$[cm]$	94	23.10	4.14	15.44	34.86
$\overline{G}$	$[cm^2]$	94	433.84	156.89	187.88	957.64
D_g	$[cm]$	94	23.46	-	-	-
D_h	$[cm]$	94	28.58	-	-	-
$\overline{H}$	$[m]$	94	19.89	4.64	11.48	30.37
$\overline{CB}$	$[m]$	94	14.03	3.15	6.97	23.92
$\overline{CE}$	$[m]$	93	2.38	0.58	1.15	3.86

Table 24: Characteristics of the simulated stand in 2001. Crown extensions and distances are average values.

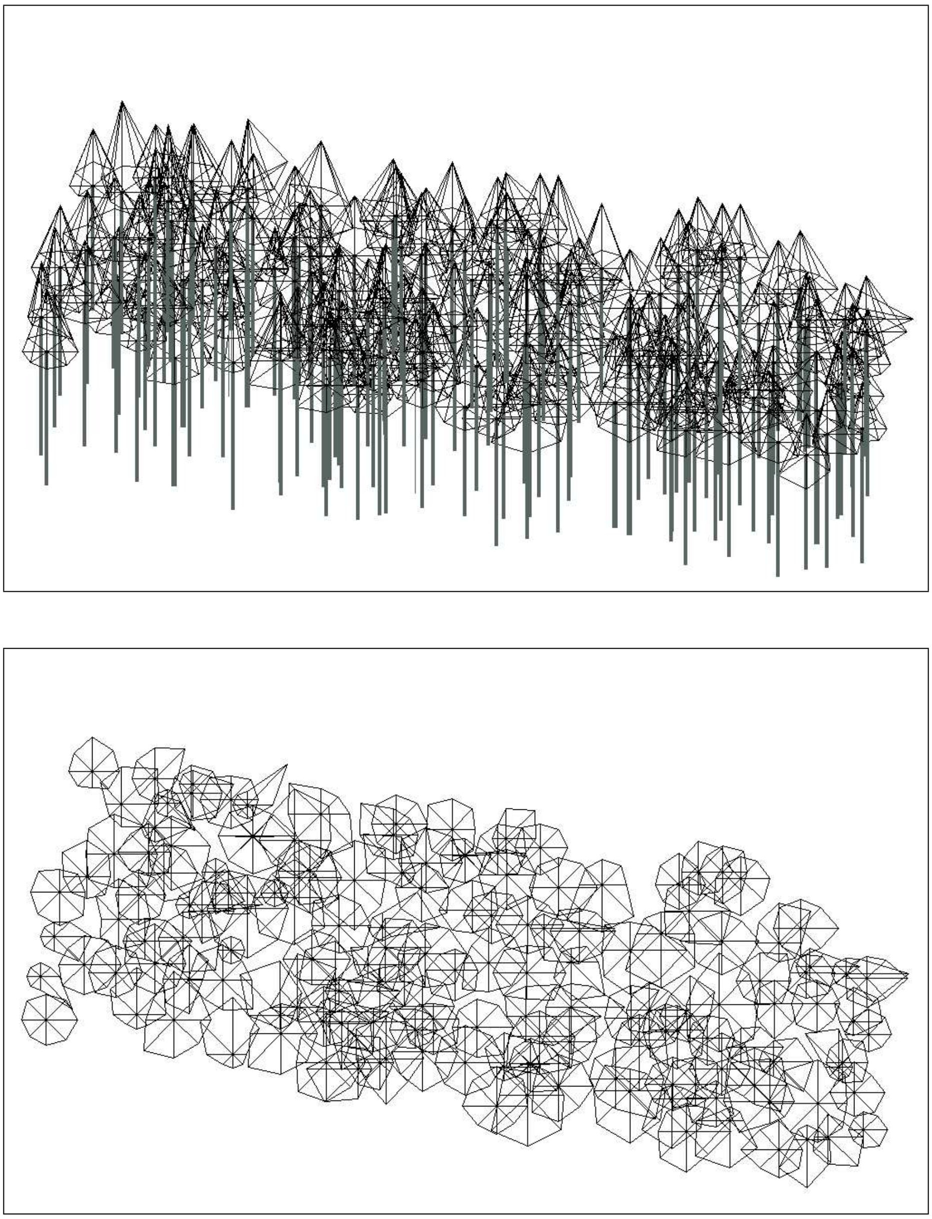


Figure 26: The simulated stand after a period of 6 years (2001 eq.). Figure 53 on page 169 shows the stand in 1995 visualized by *AMAP*.

Variable	Mean sim	Mean 2001	t	df	p-value
DBH	23.10	23.34	0.394	185.814	0.694
H	19.89	20.04	0.292	122.875	0.771
CB	14.03	13.86	-0.497	128.235	0.619
CE	2.36	2.28	-0.920	185.034	0.356

Table 25: Results of the two sided t-Test between simulation result and measurement in 2001.

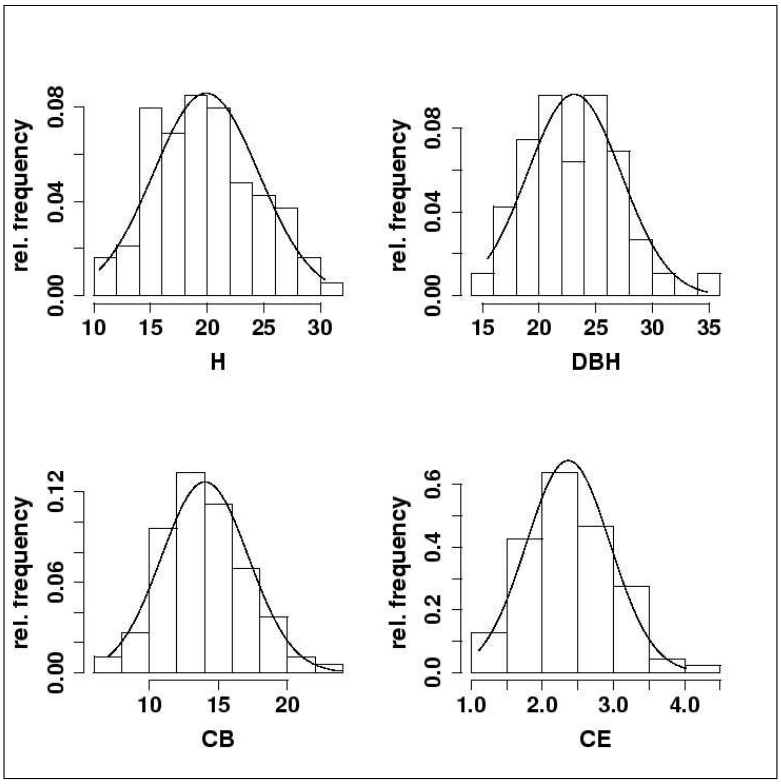


Figure 27: Characteristics of the simulated stand in 2001. H = height, CB = crown base and CE = crown extension. Smooth line = best-fit of normal distribution.

6 Extensions of *LIGNUM*

The last chapters were dedicated to the reconstruction of an aged Scots pine stand. In chapter 4 it has been already mentioned that the use of L-systems demonstrated on a model for juvenile Scots pines as an aggregation of *LIGNUM*. The model *LIGNUM* contains an extension in the light model *LIGNUM-forest*. For the modelling of Scots pine which grow in the understorey of a gap the light model extension *LIGNUM-forest* had to be modified. The model *LIGNUM* (Version 3.7 which is used in Perttunen et al. (1998)) simulates the growth of Scots pine trees on the base of the radiation regime under which the trees grow. In *LIGNUM* two aspects of the radiation are considered: the total amount of photosynthically-active radiation (PAR) reaching the segment from the sky and the reduction by tree internal shading. Competition between trees would occur by simultaneously simulating several trees. Unfortunately the *LIGNUM* version used here was not able to perform this. However, in *LIGNUM* an extension of the light model is available which allows to calculate the reduction of the radiation caused by a surrounding stand in a simplified manner. The tree stand around the tree of interest is seen as a collective of identical trees. This extension in the light model of *LIGNUM* is denoted as *LIGNUM-forest*. In this extension the reference tree is seen as a member of the surrounding stand. To simulate juvenile trees under a roof the reference tree has to be detached from the collective of the surrounding stand. This modification results in some differences in the calculation and additional requirements in the parameterization process. The extension which accomplishes this calculation was implemented by the author and is in the following called *LIGNUM-canopy*.

6.1 *LIGNUM-forest*

To simulate trees under different light regimes within a stand of homogeneous trees *LIGNUM* offers an extension to the light model (*LIGNUM-forest*). Within the simulation *LIGNUM-forest* calculates the radiation reaching a tree segment within a tree that is growing in a stand among identical trees. The idea is that a tree grows in a “hole” in a stand (Sievänen). The “hole” is represented by an empty cylinder around the tree. The incoming radiation is reduced by the surrounding stand. The reduction is expressed by the reduction factor q_r. This factor depends on the distance a light beam travels through the canopy of the surrounding stand on its way from a sky sector to the tree segment. The reduction of the incoming light additionally depends on the needle mass (q_{needle}) and the extinction coefficient (k_e) of the surrounding crowns. Equations 7 to 12 show the calculations which are carried out during the simulation. The radius (R) of the “hole” depends on the stand density (in *trees/ha*). Calculation of the travel distance (D_{tr}) through the crown canopy requires the height of intersection between the light beam and the cylinder (H_h), the height of the canopy (H) and the distance from the segment to the edge of the hole (D_0). The distance D_0 is not calculated explicitly but approximated by a mean distance ($\overline{D_0}$). The height of intersection is also influenced by the height of the reference tree segment (Z_0). Figure 28 illustrates the light model of *LIGNUM-forest.* The notation corresponds to the equations 7 to 12. M is the centre of the cylinder which represents the "hole". Here, M is the origin of the coordinate system.

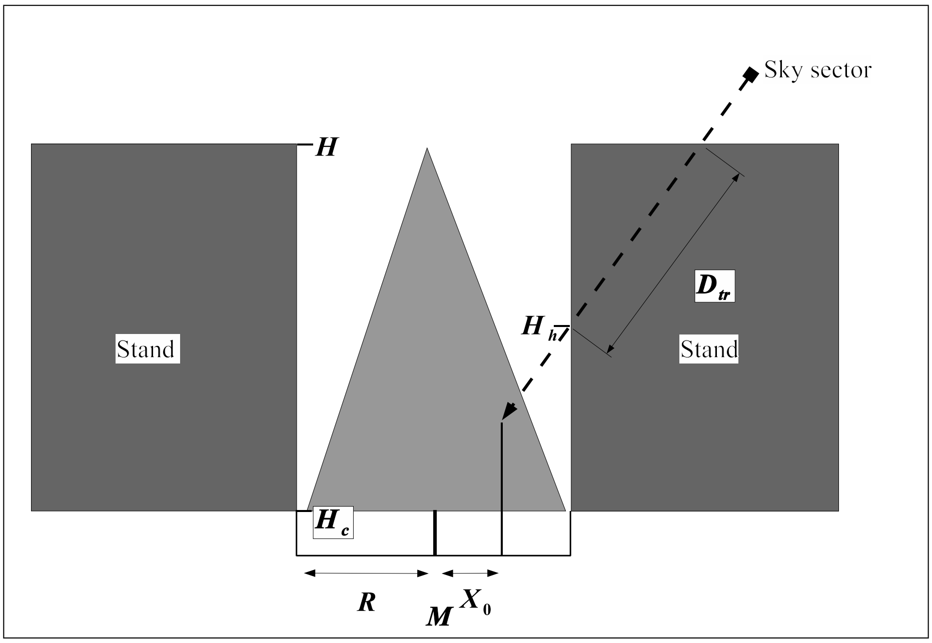


Figure 28: Light model extension *LIGNUM-forest.* H = height of stand, H_c = crown base, H_h = height of intersection, D_{tr} = travel distance, R = radius of the gap, M = centre of the gap (position of the tree), X_0 = horizontal distance of the reference segment to the centre M.

$$
\begin{aligned}
R &= \sqrt{\frac{(10000.0/Q_{tree})}{\pi}} && (7)\\
\overline{D_0} &= R \cdot \left(1 - 0.35 \cdot (X_0/R)^{2.5}\right) && (8)\\
H_h &= Z_0 + \tan(\phi) \cdot D_0 && (9)\\
D_{tr} &= \frac{H - H_h}{\sin(\phi)} && (10)\\
q_{needle} &= Q_{tree} \cdot \frac{la/10000.0}{H - H_c} && (11)\\
q_r &= e^{-k_e \cdot D_{tr} \cdot q_{needle}} && (12)
\end{aligned}
$$

q_r	=	reduction factor,
k_e	=	extinction coefficient,
$la\ [m^2]$	=	leaf (needle) area,
$R\ [m]$	=	radius of hole,
$H_h\ [m]$	=	height of the surrounding stand,
$H_h\ [m]$	=	height of intersection,
$H_c\ [m]$	=	height of the crown base,
$Z_0\ [m]$	=	height of segment,
Q_{tree}	=	density of surrounding stand,
$X_0\ [m]$	=	distance to centre of hole,
$\overline{D_0}\ [m]$	=	mean distance to edge of hole,
$D_{tr}\ [m]$	=	travel distance through canopy,
q_{needle}	=	needle density of canopy,
ϕ	=	inclination angle.

The input parameter values of the central *LIGNUM* model have already been given in table 4 in chapter 3.1. In the calculation some exceptional situations have to be considered. Equation 8 is only valid if X_0 is positive and $\leq R$. If X_0 is negative the value is reset to zero. If the value exceeds the radius X_0 is analogously reset to the radius. In similar situations the height of intersection (H_h) can be lower than the canopy crown base. In such a situation the calculated travel distance would be too high. In order to calculate the correct distance one has to make sure that the height of the intersection (H_h) is not allowed to be lower than the crown base (H_c) (cf. figure 28). Finally the reduction factor is calculated with equation 12. This formula is an exponential function which contains the extinction coefficient, the needle density and the travel distance through the canopy.

6.2 *LIGNUM-canopy*

LIGNUM-forest simulates a tree which grows under a light regime of a surrounding stand. The reference tree is a member of this collective. During this work the requirement arose to investigate the behaviour of a *LIGNUM* tree under the light condition under a roof which is "disturbed" by a gap. In this case the reference tree can not be part of the collective of the surrounding stand. However, the ideas of light reduction calculation of *LIGNUM-forest* can be adapted. The light model is illustrated in figure 29. The fact that the surrounding stand belongs to a different collective of trees creates some additional requirements to parameterization. The calculation is also affected. Concerning parameterization the values for leaf area la, height and height of crown base of the surrounding stand collective have to be provided. Furthermore the radius of the gap has to be set. The extension of the gap is not affected by the density of the stand which forms the canopy. Table 26 gives an example for the input parameters. Technically the parameters are provided by a file. Within this example additional parameters appear. The density of the canopy is indeed given as in *LIGNUM-forest*. INCL, AZIM and RAD allow to parameterize the light conditions of the hemisphere. RAD is the total photosynthetically-active radiation (PAR) per growing season in southern Finland (Perttunen et al. 1996, 1998). INCL and AZIM give the number of sectors of the hemisphere "turtle", INCL in inclination and AZIM in azimuth orientation. The values for H, H_c and LA are measured values from a site in southern Finland.

Parameter	**Value**	**Unit**	**Meaning**
INCL	10.0	-	number of inclination zones
AZIM	12.0	-	number of azimuth zones
RAD	1200.0	$[MJ]$	total annual radiation
DENS	1000.0	$[trees/ha]$	stand density
LA	53.3	$[m^2/tree]$	leaf area per tree
K	0.14	-	extinction coefficient of the canopy
H	19.7	$[m]$	height of canopy forming stand
HC	13.5	$[m]$	height of the crown base of the canopy forming stand
R	10.0	$[m]$	radius of gap
X0	0.0	$[m]$	distance of tree base to the centre of gap
Z0	0.0	$[m]$	elevation of reference point

Table 26: Parameters of the canopy extension of the *LIGNUM* light model with exemplary values.

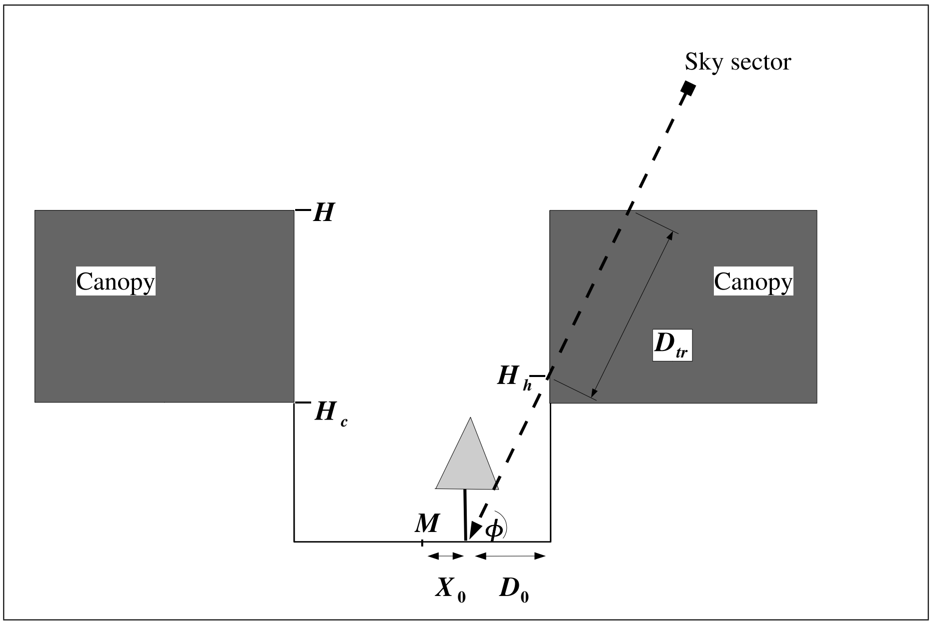


Figure 29: Light model extension *LIGNUM-canopy*. H = height of stand, H_c = crown base, H_h = height of intersection, D_{tr} = travel distance, R = radius of the gap, M = centre of the gap (position of the tree), X_0 = horizontal distance of the reference segment to the centre M.

The remaining parameters are input values for the radiation calculation. Equations 13 and 14 show the differences in the calculation to equations 7 to 12 in *LIGNUM-forest.* The distance of the tree from the edge of the gap is calculated in a more explicit way because this distances can vary from zero to twice the radius R depending on the direction of the light. This is specifically the case if the radius of the gap is high as e.g. in the example where the radius is 21 meters (cf. equation 14). The calculation of the height of the intersection point between the light beam and the gap cylinder is given by equation 14. It is identical to equation 10 except that here Z_0 is given by the modeller.

$$D_0 = \sqrt{R^2 - \sin^2(\theta) \cdot X_0^2} - \cos(\theta) \cdot X_0 \quad (13)$$

$$H_h = Z_0 + D_0 \cdot \tan(\phi) \quad (14)$$

D_0	=	distance of tree to edge of gap,
R	=	radius of gap,
X_0	=	distance of tree to centre of gap,
θ	=	horizontal angle of light beam.
H_h	=	height of intersection light beam / edge of gap,
Z_0	=	altitude of the reference point within the gap,
ϕ	=	inclination angle of the light beam.

In contrast to *LIGNUM-forest* this calculation is not applied for each tree segment and year. It is only determined once in the beginning of the simulation and only for one reference point. X_0 is the distance from the reference point to the centre of the gap. Z_0 denotes the elevation of that point. X_0 is also the position of the tree within the gap. This assumption causes an error in the calculation of D_0 and H_h for each tree segment. This differences can be expressed by

$$D'_0 = \sqrt{R^2 - \sin^2(\theta) \cdot (X_0 - x)^2} - \cos(\theta) \cdot (X_0 - x) \qquad (15)$$

$$H'_h = (Z_0 + h) + D_0 \cdot \tan(\phi) \qquad (16)$$

D'_0 = real distance to the edge of the gap,
H'_h = real height of the intersection,
x = distance between growth unit and centre of the tree,
h = height of the growth unit,

where h varies from zero to the height of the tree and x between plus/minus the distance between the endpoint of the tree segment and the stem in horizontal direction. The function for H'_h is not linear but depends on ϕ and on the result of the calculation for the distance of the edge of the gap. The values for $\tan(\phi)$ vary from 0 to 1 because the angle of ϕ varies from 0 to 45 degrees. In the following calculations the radius R has a value of 21 meters and the tree a crown extension of around 0.7 meters, hence in the horizontal direction ΔD_0 varies between zero and 1.4 meters. In the vertical direction the maximum error is exactly the height of the tree which has a maximal value of 2.7 meters.

7 Plausibility and sensitivity of *LIGNUM* †

The tests comprise the simulation of a tree over ten years under a strongly reduced light regime of 10000 trees/ha. The results are compared with real trees. Three types of comparison are applied: the fractal analysis, the diameter analysis after Chiba (1990, 2000) and a simulation of water flow with *HYDRA* (Früh 1995, Früh and Kurth 1999). This chapter is part of a publication of Dzierzon et al. (2003).

7.1 Sample trees

As reference trees which can be compared with simulated ones, three Scots pine trees (*Pinus sylvestris* L.) were investigated. The trees were 8, 10 and 11 years old; they were grown on a poor sandy (medium new red sandstone) soil in a wide-spaced stand together with some aged pine trees, located in Reinhausen near Göttingen (Germany). The trunk and the complete above-ground branching system of each tree was mapped; lengths, diameters, angles and positions of insertion nodes of each growth unit were manually measured and recorded in a DTD file (Kurth 1994), together with the topological information necessary to reconstruct the structure of the tree crown. As we have used these trees only as examples and not to deduce general statements about Scots pine, we will present only results obtained from one of them in the following sections. The results from the other two trees were similar to the presented ones in all cases.

The appearance of the analysed trees is shown in figure 30. The real pine (left side) is the 11 years old measured one, hence we simulated a pine with a (fictitious) age of 11 years (right side). As the measured trees grew within a stand, we adopted in the simulations the light regime corresponding to the concept of *LIGNUM-forest*. Table 27 gives some characteristics of the analysed trees. Trees marked with bold font are used for analysis.

Parameter	**Unit**	**Real pine 3**	**Real pine 2**	**Real pine 1**	**Simulated pine**
Age	[*years*]	10	8	**11**	**11**
Height	[*m*]	1.380	2.034	**1.155**	**1.706**
Diameter of first element	[*cm*]	3.12	1.91	**1.80**	**1.74**
Volume of stem	[cm^3]	438.58	371.58	**149.75**	**158.47**

Table 27: Some characteristics of the analysed trees.

[†] Dzierzon et al. (2003) slightly changed

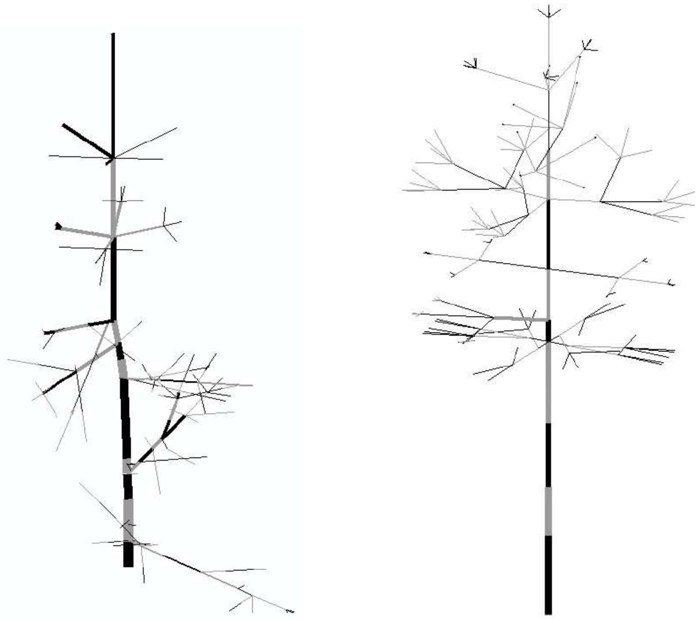


Figure 30: The simulated (left side) and the real tree (right side) which are compared.

7.2 Fractal analysis

One of our example methods is fractal analysis of branching systems. The estimation of the fractal dimension (meaning a possibly non-integral number between 0 and 3) of a natural object can be considered as a way to quantify how intensely the object fills the space in which it is embedded (Mandelbrot 1982, Voss 1988). The fractal dimension of a living organism is regarded to stand in connection with its gas exchange, biomechanical and transport characteristics (West et al. 1997, 1999). Various definitions of "dimension" and "fractal" are used in the mathematical literature (see Edgar 1990, Peitgen et al. 1992). Here we restrict ourselves to box counting dimension as the most commonly used variant for botanical objects (Stoll 1995, Berntson and Stoll 1997, Oppelt et al. 2000). Basically, it describes the relationship between the number of equally-sized cubic cells or boxes needed to cover the whole object when it is embedded in a 3-D grid and the resolution at which the object is observed, i.e. the side length of the boxes. When resolution varies between specific upper and lower bounds (outer and inner cutoff, cf. Berntson and Stoll 1997), a linear regression between log(resolution) and log(box count) can be established in many cases. The negative slope of this regression line estimates the fractal dimension of the object under consideration. The closer this value comes to 3, the stronger the space-filling tendency of the object. To get a visual impression of this approach, we refer to Kurth (1999) and to the examples given below in figure 31. However, box-counting dimension concentrates structural properties of a tree in only one value, disregarding architectural differences e.g. between branches of different age and order; its descriptive power is therefore limited. Other methods of analysis, yielding different information, will be explained below.

Box counting dimension was estimated for the skeletons of the tree crowns, using a fixed set of grid resolutions: 200, 180, 160, 140, 120, 100, 80, 60, 50, 40, 30 and 20 mm, which is identical with the set used by Petermann (1999) for young Norway spruce trees. This set of resolutions covers approximately the range of size relevant for the proper branching systems of the trees. Particularly, structures below 20 mm (”inner cutoff”), like, e.g., surface features of single shoots, are not considered. The existence of outer and inner cutoff distinguishes fractal analysis of real-world structures from ideal mathematical fractals which typically exhibit self-similarity at arbitrary scales (Berntson and Stoll 1997). Estimation of box counting dimension was carried out by simple linear regression analysis of logarithmic resolution and frequency values obtained from *GROGRA* analysis data (regression calculations done with the software tool *GroDisc*, cf. Dzierzon and Kurth 2002).

7.3 Analysis of diameters

A path is the unique connection going in distal direction from a given segment of the tree to a given branch tip supported by that segment. Usually a tree segment belongs to several paths, each ending in a different branch tip. McMahon and Kronauer (1976), Chiba (2000) postulated relationships between a tree segment and accumulated values of paths emerging from this tree segment.

McMahon and Kronauer (1976) argued that a branch follows the deflection laws of a cantilever beam (a solid with conical shape and rectangular cross-section) and that the ratio between deflection and length remains constant when the branch grows. The length of a cantilever beam is correlated with its basal diameter. McMahon and Kronauer took the mean path length of the branching system as the length of the corresponding cantilever, so that the basal diameter d of a branch scales as:

$$d = c \cdot (\bar{l} + E)^{\beta}, \tag{17}$$

where $\bar{l}$ is mean path-length and d the diameter of the basal segment. E, c and are parameters. McMahon and Kronauer (1976) postulated that is 3/2 if the structure is indeed elastically self similar. In our analysis we tried to find out if our Scots pine trees - real or simulated - have such a behaviour or not. *GROGRA* calculates the mean path length starting from each segment and gives an output as an ASCII-file. The fitting of the curve is made by STATISTICA using the iterative quasi-Newton method for non linear regression (StatSoft 1999).

Another relationship involving diameters of woody axes in trees was proposed by Chiba (1990, 2000) on the basis of the pipe model theory of trees (Shinozaki et al. 1964). He hypothesized that the total weight distal to a position z, T, should be directly supported by stem biomass per length, S, at that position. This relationship is supposed to have a linear form. Hence we get approximatively:

$$T(z) = b \cdot S(z), \tag{18}$$

where b is a proportionality constant (cf. Chiba 2000). In the examples we inserted for S the cross-sectional area of a segment and for T the accumulated weight of all tree segments (including leaves) distal to the considered one.

7.4 Simulation of water flow

In contrast to the statistical methods of analysis mentioned above, the simulation of tree-internal water flow with the software *HYDRA* (Früh 1995, Früh and Kurth 1999) follows a physical, mechanistic approach. The numerical algorithms in *HYDRA* are based on a discretized initial boundary value problem (cf. Douglas and Jones 1963) combining Darcy flow in the branched network of woody axes, water storage and conductivity losses due to cavitation events. By a sound mathematical derivation and by model tests, a consistent translation of the physiological assumptions into the computational kernel of *HYDRA* was ensured (Früh and Kurth 1999). The necessary structural information (topology and geometry of branches, leaf distribution) is taken from *GROGRA*, using a data filter which ensures the fulfilment of numerical requirements imposed upon the spatial discretization of the tree crown by performing a fusion of closely neighbouring branching nodes and by insertion of additional, intermediate nodes (see Kurth 1994, Früh 1995; for details). Time series of transpiration and soil water potential are also input of *HYDRA* and can be taken from measurements or from separate models.

In contrast to other models of the tree as a hydraulic system (e.g. Tyree and Sperry 1988, Rapidel 1995), *HYDRA* does not only calculate steady-state distributions of water potential and flows, but allows short-term dynamic studies concerning the response of the system to sudden changes in the transpiration rate (Früh and Kurth 1999). However, here we will restrict ourselves to simulated steady-state profiles of water potential along selected paths in the crown. This sort of output allows to find out in which branches the lowest potentials occur, and to assess the hydraulic significance of crown architecture by comparing profiles obtained from systematically varied branching systems.

Current limitations of *HYDRA* are the assumption of a uniform transpiration rate throughout the whole crown, the missing connection to leaf energy balance and stomatal regulation, and the lack of a feedback to a model of water transport in the soil. All these issues are currently being addressed (Lanwert et al. 1998, Schulte 2002).

7.5 Results of the fractal analysis

Figure 31 shows the results of fractal analysis. Each of the trees shows fractal behaviour. The real Scots pine has a dimension of around 1.24. That of the simulated one is practically the same (1.27). The low values (< 2) result from our exclusion of foliage and from our choice of outer and inner cutoff. So the two branching systems are "not far" from linear structures (dimension 1) in the considered range of grid resolutions.

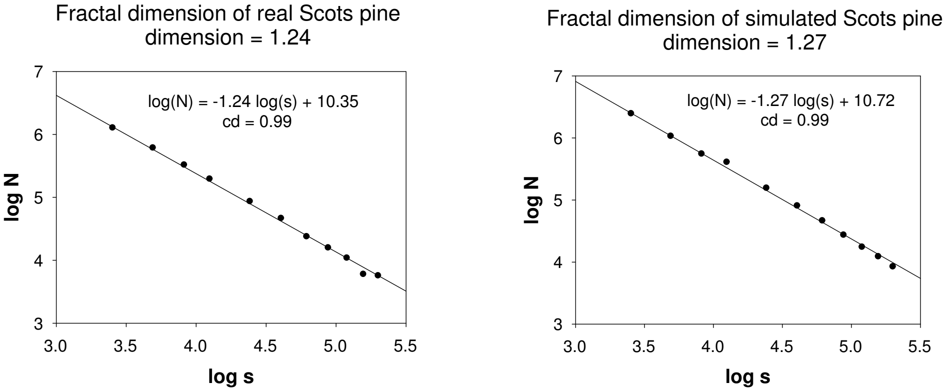


Figure 31: Results of fractal analysis. Left: real Scots pine. Right: simulated Scots pine. s = grid resolution, N = number of occupied grid cells.

7.6 Results of the analysis of diameters

The results of path length analysis are shown in figure 32 and Table 28. In contrast to McMahon and Kronauer (1976) our trees do not have a regression exponent (β) of 3/2. But this result should be interpreted with caution. As McMahon and Kronauer (1976) have already observed, the optimum in the iterative minimizing process is very flat. Hence changes in β do not result in significant changes of the coefficient of determination. In fact, it is possible to create a regression for the real Scots pine with a fixed β of 3/2 with a coefficient of determination of 0.645. This is nearly the same coefficient as before (0.648).

Parameter estimations: MacMahon & Kronauer function				
	C	**E**	β	**cd**
Real tree	0.0046	108.3396	1.15494	0.65
Simulated tree	0.0137	-0.9305	0.9491	0.90

Table 28: Estimation of parameters c, E and β from McMahon and Kronauer's equation. cd = coefficient of determination

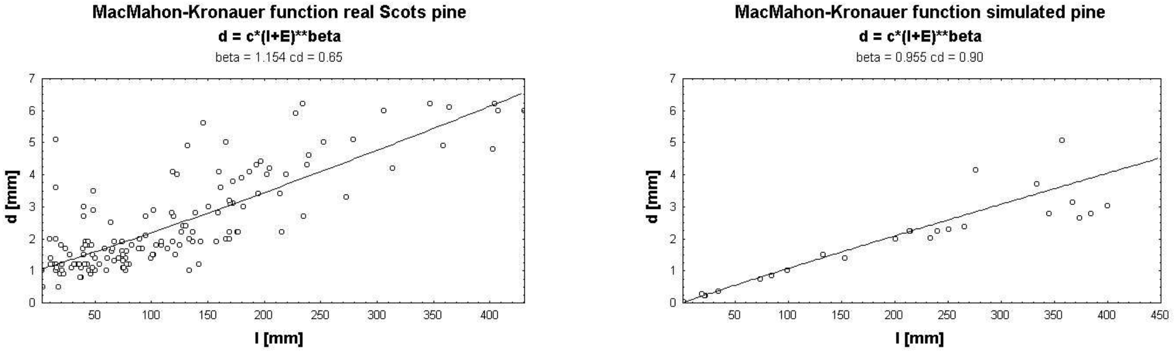


Figure 32: Results of path length analysis. Left: real Scots pine. Right: simulated Scots pine. d = segment diameter, l = mean distal path length; c, E and β are fitted parameters; cd = coefficient of determination

Several data points of the simulated pine in figure 32 coincide. The simulated tree has indeed more segments than the scatterplot implies. Because of its symmetry, several segments have the same relationship between diameter and mean path length and are therefore plotted only once.

Figure 33 shows the results of analysis of supplied mass of a tree segment. S is the basal area of a segment, T the accumulated mass of the segments supplied by the considered segment (cf. Chiba 2000). The range of S and T is nearly the same in both cases but the shape of the scatterplot is different. The relationship was supposed to have a linear form, but the simulated pine seems to exhibit a nonlinear shape which we fitted tentatively by a quadratic polynomial.

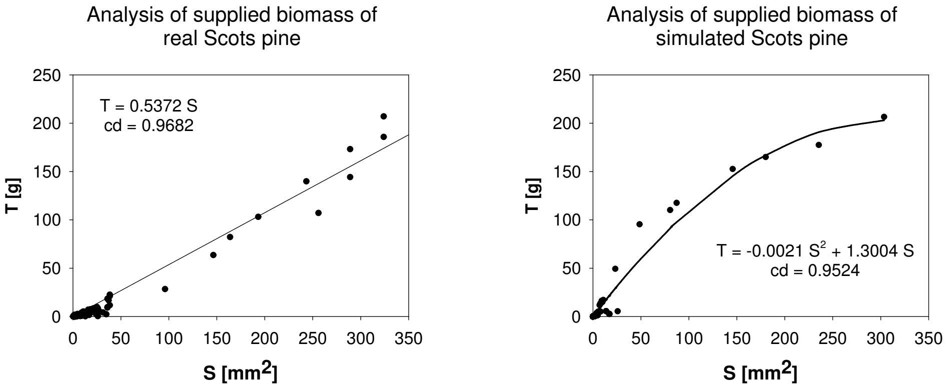


Figure 33: Results of analysis of supplied biomass. Left: real Scots pine. Right: simulated Scots pine. S = cross section area of segment, T = accumulated biomass of all segments distal to the considered one, cd = coefficient of determination.

7.7 Results of the simulation of water flow

Figure 34 shows the results of simulation of water flow. In this simulation we used an empirical relationship between the diameter d of a segment and its hydraulic conductivity HC (cf. Cochard 1992). The relationship is

$$HC = 3.751 \cdot d^{2.41} \tag{19}$$

for Scots pine. *HYDRA* (Früh 1995) gives a diagram of water potential $[MPa]$ versus path length $[m]$ as possible output. Before starting the simulation, paths have to be selected. *HYDRA* calculates among others the water potential within all segments. The graphical output uses only preselected paths to prevent confusion. To give an overview about simulated waterflow within the trees, paths were selected which represent all branching orders in the tree. Another term of selection was the location within the tree. We tried to select paths which represent different locations. The longest path is the main stem (branching order 0). Along a path, the order usually changes. Hence the marked orders in figure 34 are the orders of the last segment.

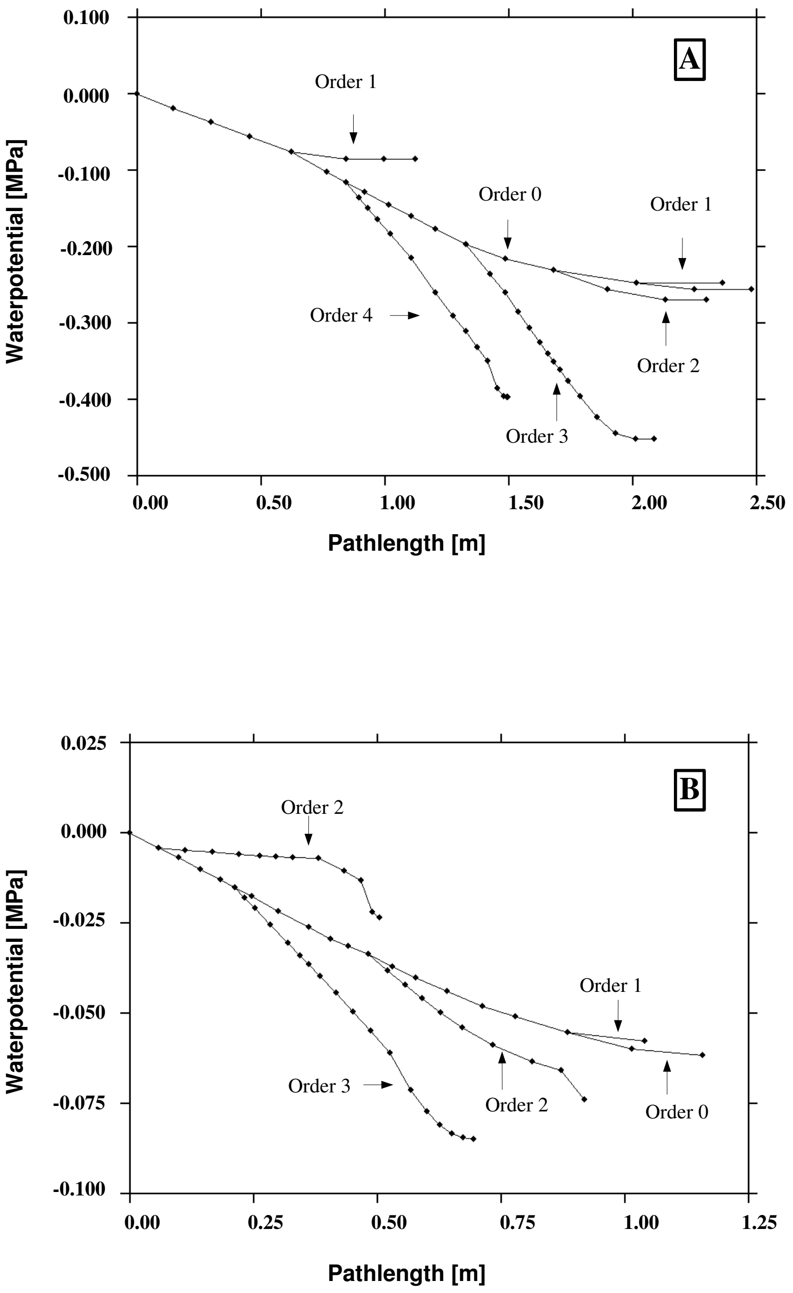


Figure 34: Water potential versus distance from base of the tree along preselected paths with different values of branching order, in the real (A) and simulated (B) tree. The order of a path is determined by the order of its last segment.

7.8 Sensitivity of *LIGNUM-canopy*

In the last chapters *LIGNUM* has been tested under dominant light regimes. Dominant means that the stand density of the sorrounding stand was extraordinary high (10000 trees/ha). In the further steps, however, the canopy version instead of the forest version is used. The sensitivity tests with *LIGNUM-forest* illustrated the behaviour of *LIGNUM* in extreme situations. Therefore repeating the same tests with the canopy version seemed to be inefficient because one would expect similar results. Nevertheless, tests with *LIGNUM-canopy* showed that the *LIGNUM* model is sensitive concerning the parameters a_f (needle mass - tree segment area relationship) and f_c (coverage of needles: 1 kg of needles will cover f_c m^2). In addition, the influence which the the radius of the gap (R) in the canopy and the distance of the reference point (X_0) to the centre of the gap may have on the simulation results has to be tested before the model is applied. Two simulation runs over a time period of 11 years are applied: one with $a_f = 1.3$ and $f_c = 14$ (variant 1) and one with $a_f = 0.6$ and $f_c = 28$ (variant 2).

To contrast the differences between a varying distance to the centre of the gap and a varying radius of the gap both possibilities are illustrated. The upper mentioned variants are implemented using a varying radius and a constant X_0.

7.8.1 Results

The results illustrate two aspects of the simulation with *LIGNUM-canopy*. The first is the difference between a varying radius R and distance to the centre of the gap (X_0). The calculation of PAR is independent of from a_f and f_c because a_f and f_c are internal parameters for the reference tree and do not affect the calculation of the transmittance of radiation through the canopy. The second aspect are the differences of results according to the internal parameter a_f and f_c (variant 1 and 2).

Figure 35 shows the total incoming PAR in relation to R and X_0. The range of both curves is of interest. A closed canopy ($R = 0.0$) has a large influence on the total incoming radiation reaching the bottom of the gap, hence it constitutes a minimum PAR of 583 MJ. The position of a reference point relative to the edge of the gap has less influence (PAR=910 MJ at $D_0 = 21.0$). The reason for this is that the reference point at the edge of a gap with a radius of 21 meter has several sectors which are not hindered by the canopy, hence the light from these sectors is not reduced. If the gap radius is zero no “free” sector is existing. The maximum values in both cases (varying radius and distance) are equivalent (PAR = 1166 MJ). The reason is simple: In both cases X_0 equals zero and R 21.0 meters.

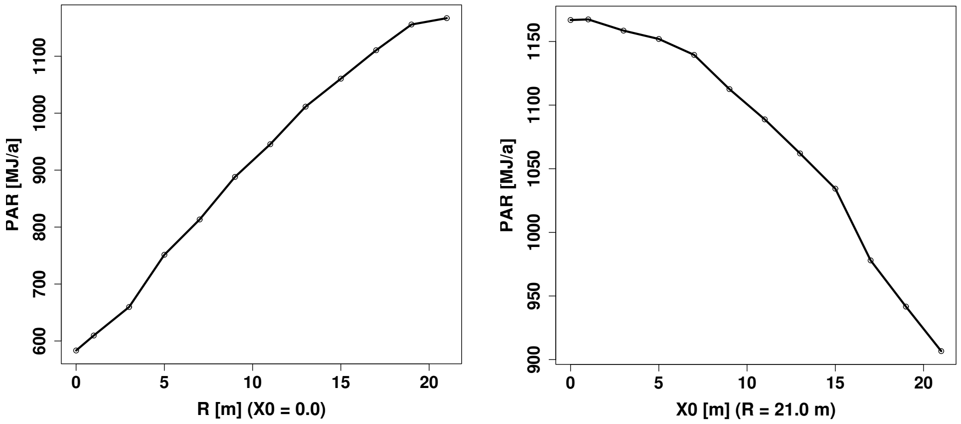
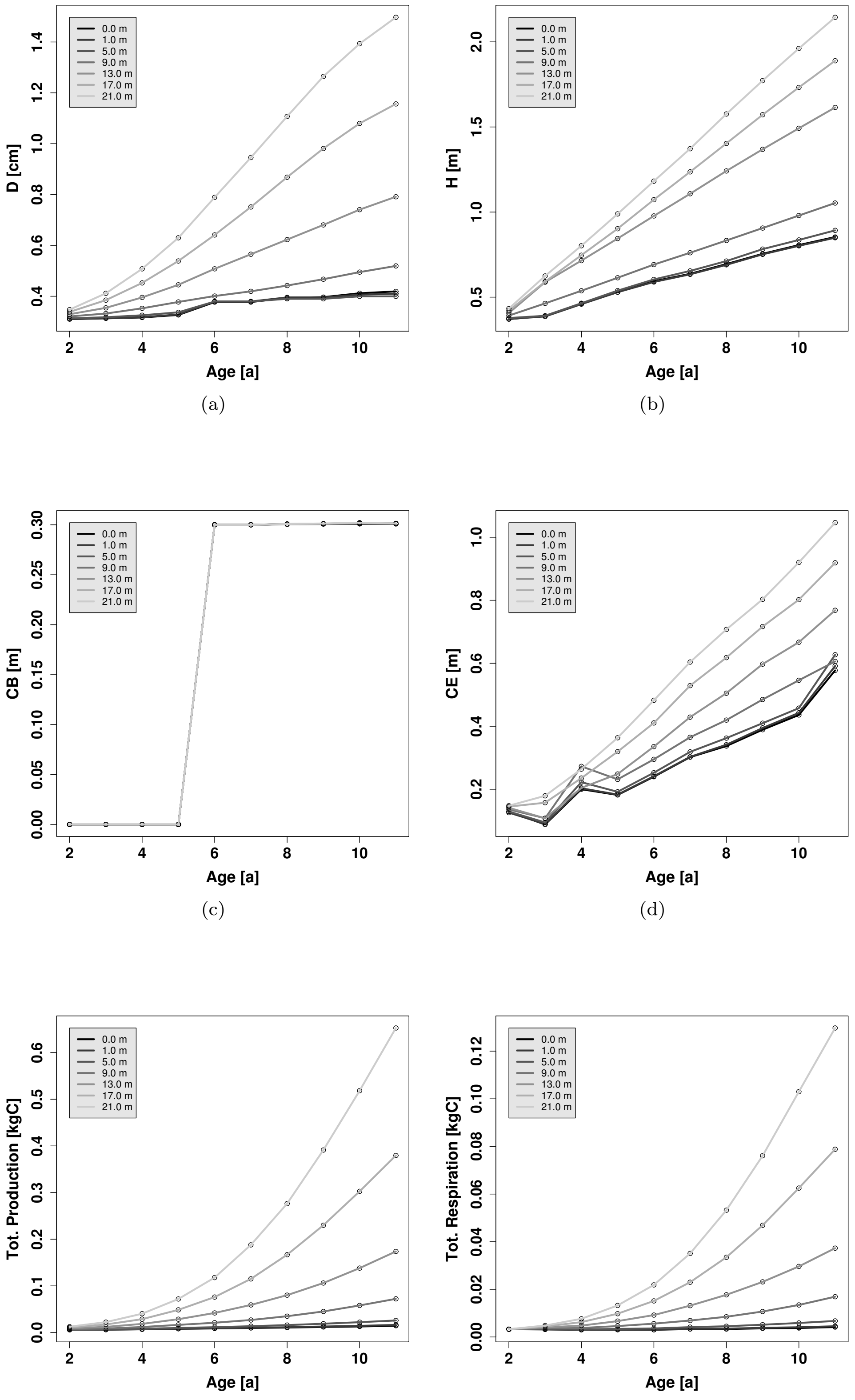


Figure 35: Total incoming photosynthetically-active radiation in relation to the radius of the gap (left) and the distance of the tree to the centre of the gap (right).

Figure 36, 37 and figure 38 show the simulation results of the two alternatives. These figures illustrate the development of the geometrical parameters height (H), diameter (D0.0), crown base (CB) and the crown extension (CE) in time under different radiation regimes. In addition to these parameters the total production of carbon and the total respiration are shown. These values are interesting in relation to the gap radius because a tree are supposed to die if the incoming radiation is too small.

The pictures exemplify the sensitivity of the model to both variants. A high coverage of needles and a low needle mass - tree segment relationship continuously increase the dimensions of the resulting values. In general the different radii have the expected influence on the development of the parameter over time. The smaller the radius the greater the increment in time. The total production and total respiration even approach zero if the radius also converges to zero. The height of the crown base suddenly increases in both cases at an age of 6 years. The simulations result in an enormous number of segments.



(a)

(b)

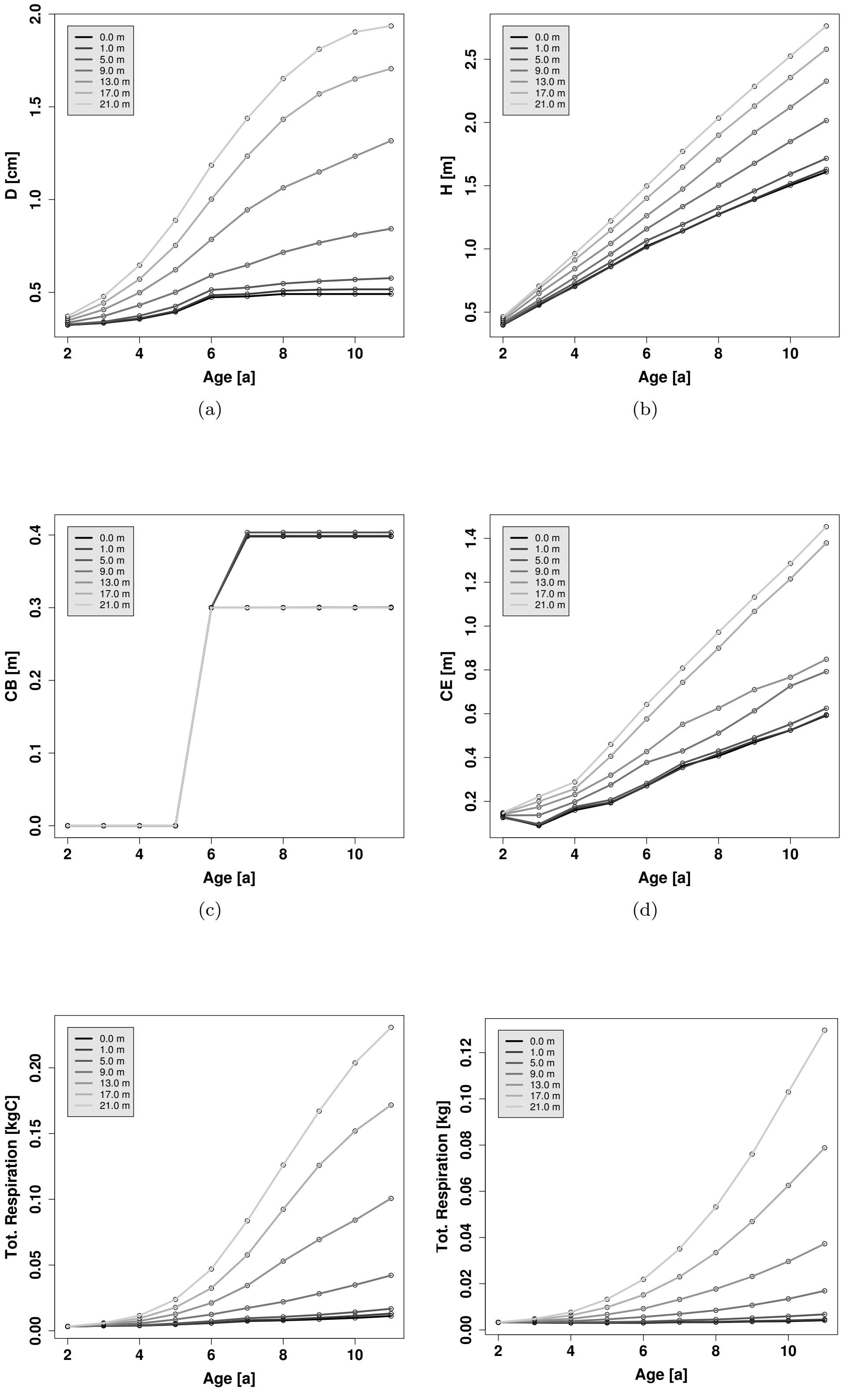
(c)

(d)

(e)

(f)

Figure 36: The development of diameter (D0.0) (a), height (H) (b), crown base (CB) (c) and crown extension (CE) (d) with: $a_f = 1.3$ and $f_c = 14$. Image (e) is the total photosynthetic production and (f) the total respiration. The different colours of the lines denote different radii of the gap.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 37: The development of diameter (D0.0) (a), height (H) (b), crown base (CB) (c) and crown extension (CE) (d) with: $a_f = 0.6$ and $f_c = 28$. Image (e) is the total photosynthetic production and (f) the total respiration. The different colours of the lines denote different radii of the gap.

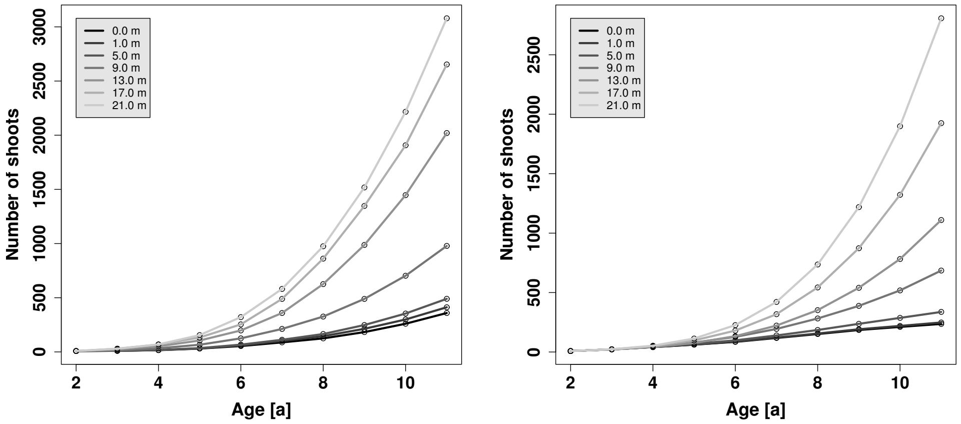


Figure 38: The development of the number of shoots with $a_f = 1.3$ and $f_c = 14$ (left) and $a_f = 0.6$ and $f_c = 28$ (right).

8 Model aggregation of *LIGNUM* Scots pines

8.1 Motivation

The idea was to create an individual based model which simulates the growth of trees in the understorey of a gap as an aggregation of *LIGNUM*. The model shall be implemented with an L-system. The aggregation is obtained by using individuals with simplified crown structures as in Pretzsch (1992a). The *LIGNUM-canopy* version is then used in simulation runs for trees in the understorey of a gap. The results of the simulation of juvenile trees are aggregated and integrated into the new model which is implemented using L-systems. The aggregated version is then used for further simulation runs. The results of this simulation results are compared with the original data from *LIGNUM-canopy*.

8.2 A model for a juvenile Scots pine as an aggregation of *LIGNUM-canopy*

In the scope of ecology Franc and Picard (2001) noticed that often descriptions of individuals in a population are aggregated into one global variable, such as mean size. In our case, the spatial scale is different. However, if one sees the tree as a collection of tree segments, this view of aggregation can be maintained. The descriptions of all tree segments as a population shall be aggregated to a reduced number of variables such as height, crown extension and diameter of a tree.

LIGNUM-canopy simulates a juvenile tree and its behaviour under a light regime which is influenced by a gap. Within the gap the incoming radiation depends on the distance of the tree to the centre of the gap. The closer the tree is to the edge the smaller is the incoming radiation. The parameters height, height of the crown base, diameter and mean crown extension are investigated in relation to that distance. It will be shown that — except the crown base — all parameters can be approximated using linear regressions with time and distance as independent variables. These results are integrated into an L-system. The L-system has three components: a model for the individual tree, a representation of the canopy and a light model. The results of the L-system simulation runs are presented and compared with the original data. To get an impression of the differences between the sensitive functions 15 and 8, both functions are used for simulation runs. This application of either function 8 and 15 can be seen both as a model comparison and a sensitivity test.

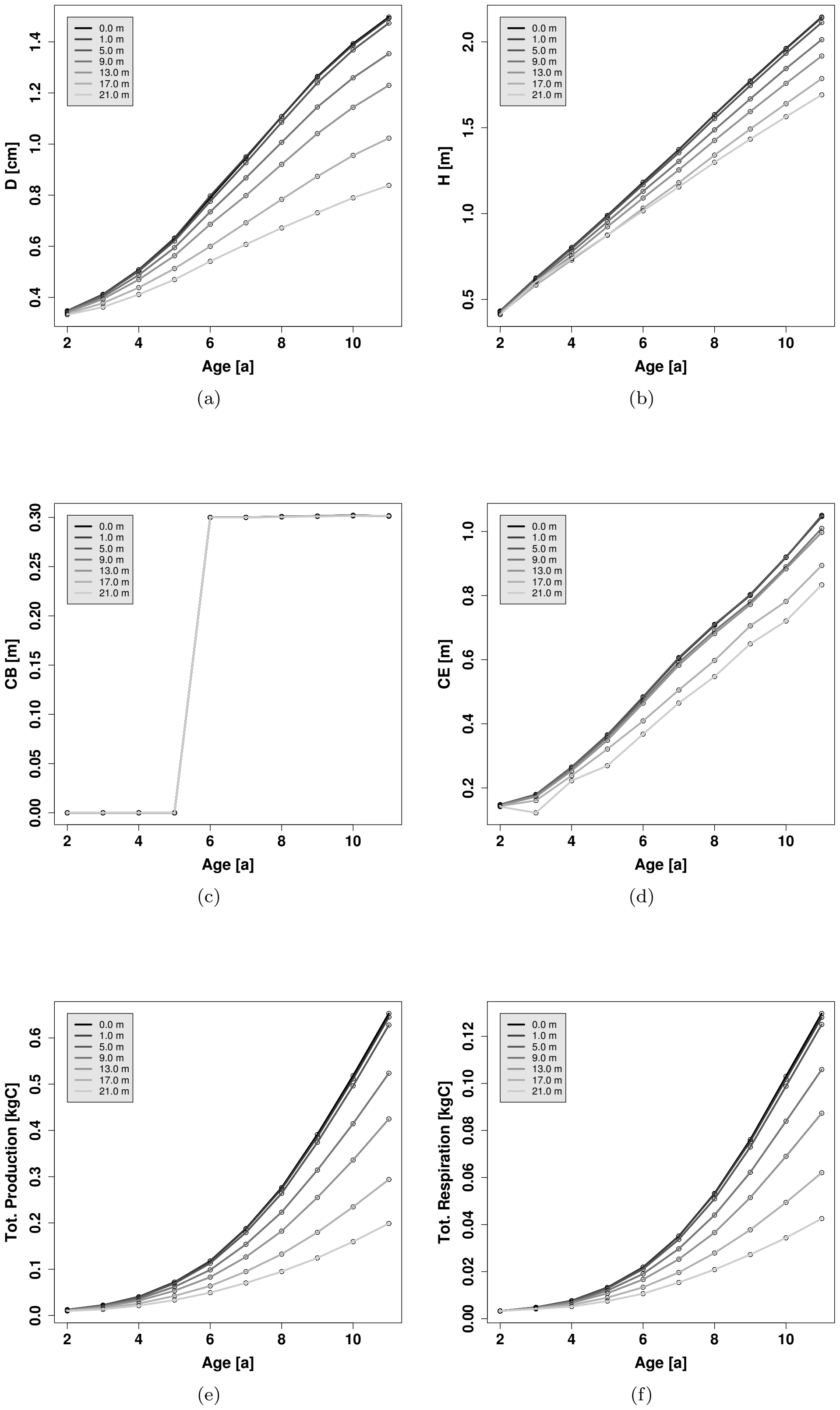
8.2.1 Simulation runs with *LIGNUM-canopy*

The sensitivity tests were done with a varying radius of the gap. The simulation runs concentrated on a varying distance (X_0) of the tree to the gap edge. The previous sensitivity tests gave no precise answer according to the parameter a_f and f_c. The results from both parameter settings are reasonable. In Perttunen et al. (1998) the parameter combination was: $a_f = 1.3$ and $f_c = 14$. For that reason this combination is also used within the following simulation runs. Figure 39 illustrates the results of the simulation run with *LIGNUM-canopy* with a varying X_0. These results are similar to the sensitivity tests in chapter 7.8. Figure 35 has already shown the differences between the radiation profiles resulting from a varying radius of the gap and from a varying distance of the tree to the gap centre. A greater radiation income with a constant radius results in greater trees. The minimum height for instance is 1.69 meters with a varying X_0 and 0.75 meters with a varying radius.

Figure 40 completes the results by showing the number of shoots the *LIGNUM* tree has. Figure 41 exemplifies the shapes of the trees within the gap. Three trees from three different distances are chosen: one from the centre ($X_0 = 0.0\ m$), one from an intermediate position ($X_0 = 11.0\ m$) and one from the edge of the gap ($X_0 = 21.0\ m$). The figure shows how the shape of the tree changes along the radiation gradient. The more dominant the light regime the less plenteous the tree. The images were obtained with *GROGRA* using the *LIGNUM-GROGRA* interface without depicting the needles.

8.2.2 Aggregation of the simulation results

The aggregation was implemented as a regression between the light conditions and the growth parameters height, diameter, crown extension and height of the crown base. The results were integrated into an L-system in the *GROGRA* specification. *GROGRA* offers sensitive functions which calculate the light conditions for a tree segment. The concept differs from that of *LIGNUM*. *GROGRA* uses a hemispherical covering ratio between the maximum number of sky sectors and the actual number of sky sectors which are not shaded by another segment (CR, function 8, see chapter 2.3.3). Function 15 additionally weights the shaded sector with $\sin(\phi)$ where ϕ is the inclination angle of the incoming radiation (WCR) (cf. equations 20 to 22).



(a)

(b)

(c)

(d)

(e)

(f)

Figure 39: The development of diameter (D0.0) (a), height (H) (b), crown base (CB) (c) and crown extension (CE) (d) with: $a_f = 1.3$ and $f_c = 14$. Image (e) is the total photosynthetic production and (f) the total respiration. The different colours of the lines denote different distances of the reference tree to the centre of the gap.

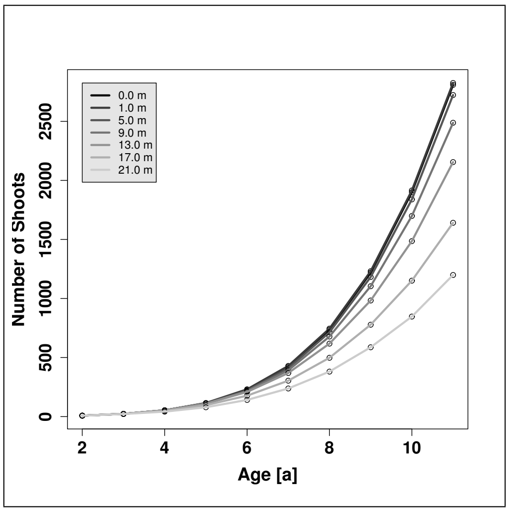


Figure 40: The development of the number of shoots ($a_f = 1.3$, $f_c = 14$). The different colours of solid lines denote the different distances to the centre of the gap.

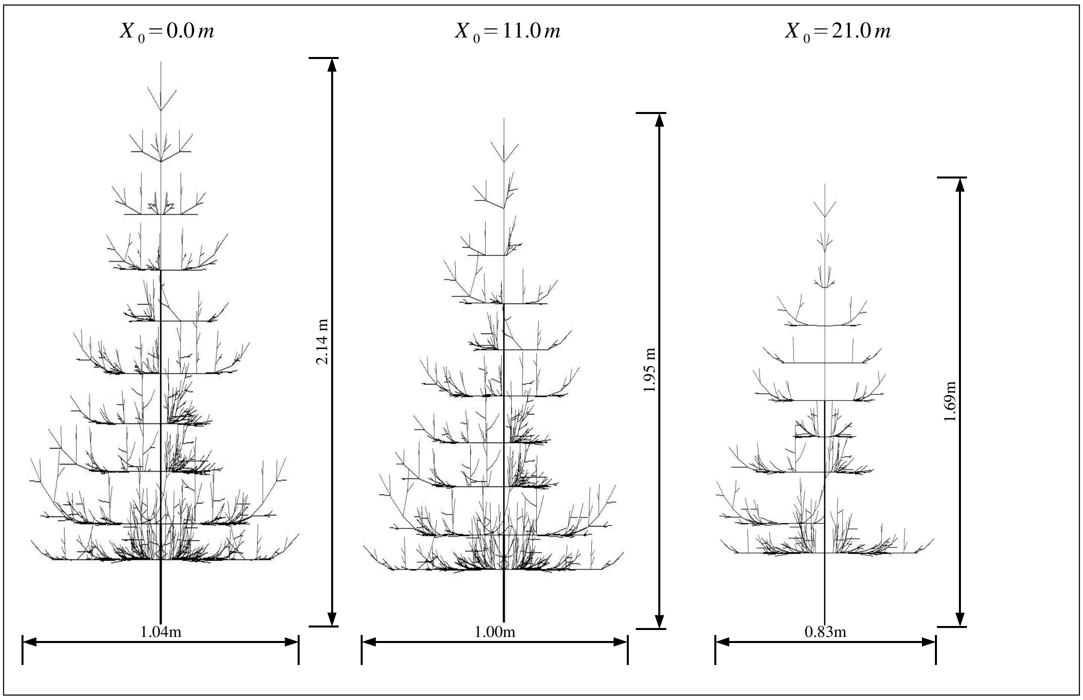


Figure 41: The shape of the trees simulated with *LIGNUM-canopy*. The trees are depicted using *GROGRA* without depicting the needles. The shapes are exemplified by three positions: one in the centre ($X_0 = 0.0\ m$), on at an intermediate position ($X_0 = 11.0\ m$) and one which is located at the edge of the gap ($X_0 = 21.0\ m$). The size of the tree is given by the height (H) and crown extension (CE).

$$\begin{aligned} RAD &= \frac{\sum PAR_i}{PAR_{tot}} && (20) \\ CR &= 1 - \frac{\sum CR_i}{CR_{tot}} && (21) \\ WCR &= 1 - \frac{\sum \sin(\phi_i) \cdot CR_i}{\sum \sin(\phi_i)} && (22) \\ & && (23) \end{aligned}$$

PAR_i = reduced radiation income from sky sector i, according to the *LIGNUM* radiation model.

PAR_{tot} = total radiation, according to the *LIGNUM* radiation model.

CR_i = shaded light beam from sky sector i. Values can be 0 for not shaded and 1 for shaded.

CR_{tot} = total number of sky sectors.

ϕ = inclination angle of light beam

Figure 42 gives the quantitative differences between the different concepts. The values for RAD, CR and WCR are calculated during the simulation process of *LIGNUM-canopy*. Table 29 additionally contains the corresponding values for the height, crown extension and diameter. These parameters are expressed as annual increments. Based on this data it is possible to calculate a regression between the light situation and the annual increments as input for the L-system model. The WCR values will be used as input values for the regression. Figure 43 illustrates these relations and table 30 shows the results of the regression analysis. The coefficient of determination (R^2) exceeds values of 0.8 and all slopes are significantly different from zero (p-value).

8.2.3 Implementation of the aggregation as an L-System

The L-system for the implementation of the model for juvenile Scots pines is due to the ideas which were presented in chapter 5.2. The tree is seen as decomposed into the compartments “main stem” and “crown”. The main stem is represented by an empty cylinder and the crown as a irregular pyramid (Pretzsch 1992a). *GROGRA* does not offer an irregular pyramid as a geometrical body. Thus the crown is divided into the surface represented by triangles, the base units from the crown base to the nodes of the pyramid and a central unit from the crown base to the top of the crown. The shape of the L-system is then very similar to the implementation of the reconstruction of the stand in Syke (cf. chapter 5.2) but contains a model for the incoming radiation. The L-system

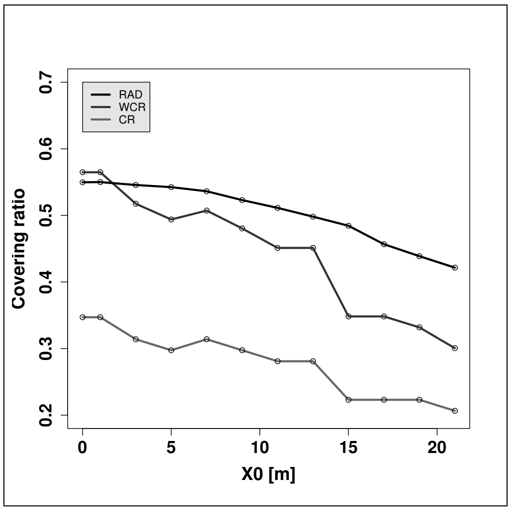
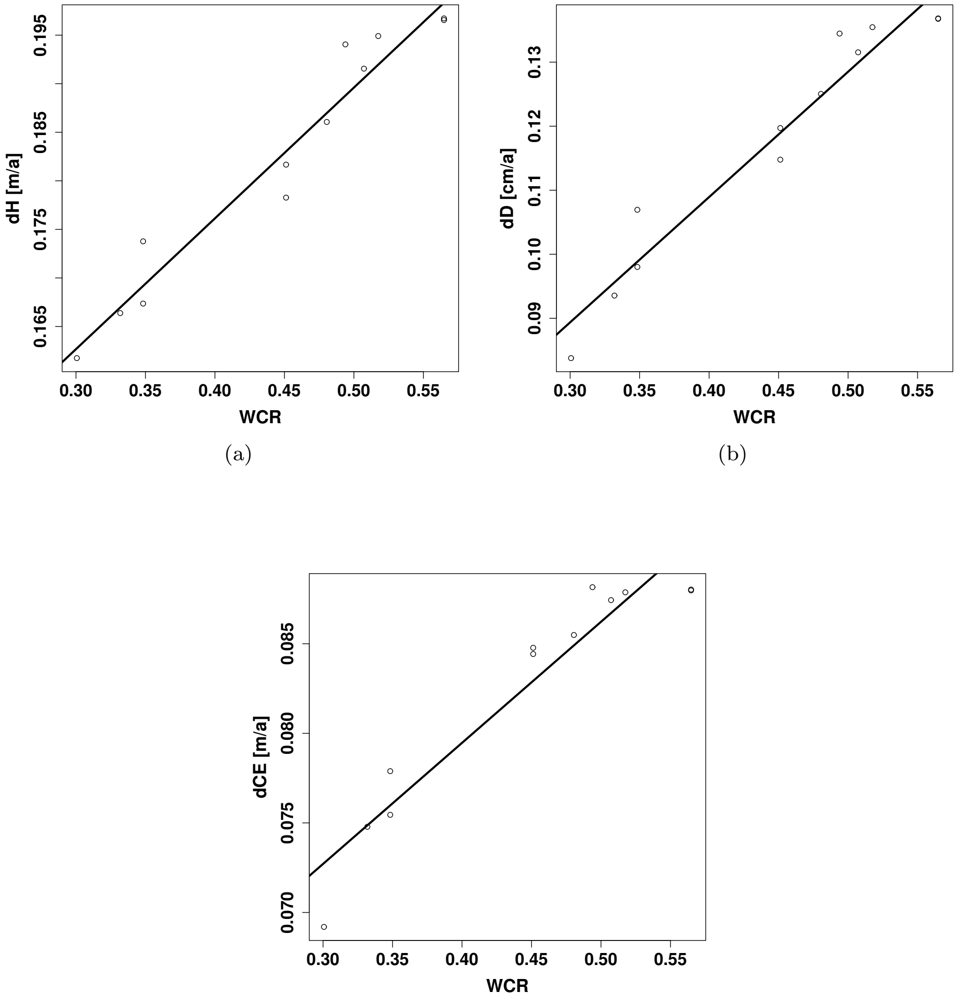


Figure 42: The covering ratios depending on the distance of the reference tree to the centre of the gap (X0). RAD = ratio between total incoming radiation to potential total radiation according to *LIGNUM*, CR = ratio between the number of non shaded sectors to the total number of sectors (equivalent to function 8 of *GROGRA*) and WCR = the same as CR but the number is weighted by the sine of the inclination angle (equivalent to function 15 of *GROGRA*).

Distance [m]	**RAD**	**CR**	**WCR**	**ΔH**[m/a]	**ΔD**[cm/a]	**ΔCE**[m/a]
0	0.5498	0.3471	0.5648	0.1967	0.1368	0.0880
1	0.5500	0.3471	0.5648	0.1966	0.1368	0.0880
3	0.5457	0.3140	0.5175	0.1949	0.1355	0.0879
5	0.5424	0.2975	0.4939	0.1941	0.1345	0.0882
7	0.5363	0.3140	0.5072	0.1916	0.1315	0.0874
9	0.5230	0.2975	0.4805	0.1861	0.1250	0.0855
11	0.5113	0.2810	0.4512	0.1817	0.1197	0.0848
13	0.4981	0.2810	0.4512	0.1783	0.1148	0.0844
15	0.4845	0.2231	0.3483	0.1738	0.1069	0.0779
17	0.4567	0.2231	0.3483	0.1674	0.0980	0.0755
19	0.4388	0.2231	0.3319	0.1664	0.0935	0.0748
21	0.4216	0.2066	0.3006	0.1617	0.0838	0.0692

Table 29: The increment rates of H (height), D0.0 and CE (crown extension). They are results of linear regressions to RAD (covering ratio). The values for CR and WCR are given for comparison purposes. RAD = covering ratio PAR_{red}/PAR_{tot} where red = reduced light income and tot = total incoming radiation, CR = ratio of non shaded sectors to total number of sky sectors.



(a)

(b)

(c)

Figure 43: The development of the investigated parameters height (H) (a), diameter (D0.0) (b) and mean crown extension (CE) (c) in relation to the weighted covering ratio WCR.

Variable	**Unit**	**Int.**	**Slope**	$\mathbf{R^2}$	**p-value**
ΔH	$[m/a]$	0.1223	0.1347	0.9475	<0.0001
ΔD	$[cm/a]$	0.0307	0.1957	0.9407	<0.0001
ΔCE	$[m/a]$	0.0524	0.0676	0.9100	<0.0001

Table 30: Linear regressions between increments of height (H), diameter (D) at the bottom and crown extension (CE) to the weighted covering ratio WCR. The p-value is the result of the statistical test whether the slope is not zero. This table corresponds to figure 43.

has two sections: one creates the juvenile stand and one a representation of the aged, surrounding stand. In the first rule (see rule 0 in listing 17) two symbols are established which represent both compartments of the stand: $juven$ and $roof$.

Symbol	**Meaning**
*	the axiom
juven	juvenile trees (could be a set of only one tree)
roof	the canopy represented by palisades arranged in a spiral
pinus	representation of a single tree
urstem	intermediate step needed by light calculation
stem	the main stem of a tree
crext	the crown base units of a tree (eight pieces)
crtop	the central unit of the crown

Table 31: The symbols of the implementation of juvenile Scots pine trees.

The model represents an aggregation of the *LIGNUM-canopy* extension. In *LIGNUM-canopy* the canopy is simplified by a homogeneous roof. In a geometrical view the roof is a box with a specified crown base and height. This box can be disturbed by a regular gap which is represented by an empty cylinder. Within the L-system the representation of the roof has to be done by a trick because the L-system specification of *GROGRA* does not offer the geometrical representations which are needed here. The roof is simulated by virtual palisades (cylindrical elements). The palisades are positioned in a spiral which fills the space in an optimal way (analogously to the spirals of phyllotaxis). These positions result from a rotation around 137.5 degrees. The length of the forward step is then $3.16 \cdot \sqrt{i}$ where i is an index number (cf. listing 17 and figure 45).

The symbol $roof$ (rule 1) is replaced by a sequence of commands which creates the positions of the palisades and the symbol *palis*. Two rules are dedicated to the symbol *palis*: one generative (rule 2) and one interpretive (rule 3). In rule 2 the distance of a palisade to the centre of the gap is calculated. This value affects the interpretation. If the distance is greater than the radius of the gap the palisade is created, otherwise it is not. The variables *c_x* and *c_y* are sensitive variables which return the position of the turtle. *gap_x* and *gap_y* are the coordinates of the centre of the gap.

The last question of the implementation of the roof was part of the parameterization: How many palisades of the canopy representation are reasonable? The smaller the number of palisades the greater is the transmittance through the canopy. The target was to stop

Name	Spec.	Meaning
w	uniform	random angle for spatial distribution of juvenile trees
c_x	xcoordinate	x-coordinate of a palisade
c_y	ycoordinate	y-coordinate of a palisade
i	index	index variable for use within the scope of repetition operator
sky	function	sensitive function 15 or 8 for the calculation of WCR/CR
corr_cr	const	correction factor for WCR/CR

Table 32: Global variables of the L-system implementation for simulating juvenile Scots pine trees.

Name	Unit	Used in symbols	Meaning
h	$[m]$	urstem, stem	actual height of tree
d	$[m]$	urstem, stem	actual diameter of tree
t	$[a]$	urstem, stem, crtop	actual time step
cre	$[m]$	urstem, stem, crext, crtop	actual crown extension of the tree
crt	$[m]$	urstem, stem, crext, crtop	actual crown height of the tree
dtg	$[m]$	palis	distance of a palisade to the centre of the gap

Table 33: Local variables used in the context of symbols within the L-system of the simulations of juvenile Scots pine trees.

name	unit	meaning
r	$[m]$	distance of reference tree to centre of gap (varied in different simulation runs)
trees	-	number of juvenile trees
npalis	-	number of palisades for canopy representation
colpalis	-	colour of palisades (for visualization)
h_stand	$[m]$	height of canopy
cb_stand	$[m]$	crown base of canopy
rad_gap	$[m]$	radius of gap
gap_x	$[m]$	x-coordinate of the centre of the gap
gap_y	$[m]$	y-coordinate of the centre of the gap
h_int	$[m/a]$	intercept of the height increment vs. weighted covering ratio regression
h_slo	$[m/a]$	slope of the height increment vs. weighted covering ratio regression
d_int	$[m/a]$	intercept of the diameter increment vs. weighted covering ratio regression
d_slo	$[m/a]$	slope of the diameter increment vs. weighted covering ratio regression
cre_int	$[m/a]$	intercept of the crown extension increment vs. weighted covering ratio regression
cre_slo	$[m/a]$	slope of the crown extension increment vs. weighted covering ratio regression

Table 34: Global constants in the L-system of the simulations of juvenile Scots pine trees.

any transmittance caused by an insufficient number of palisades. Figure 44 exemplifies what happens with the results of the sensitive function 15 if the number of palisades varies. If the number of palisades exceeds 400 the covering ratio stays constant, hence this value was chosen for the further simulation runs.

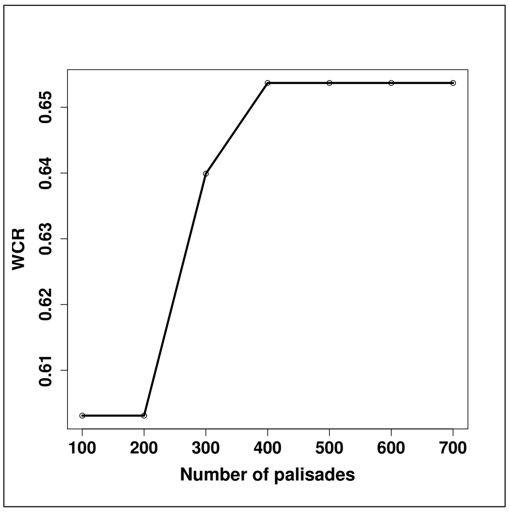


Figure 44: The covering ratio WCR in relation to the number of palisades of the canopy representation within the L-system.

Listing 17: Implementation of the canopy.

```
/*rule 0*/  * → [ roof ] [ juven ],

/*rule 1*/  roof → RL−90 f(y_c) RU(90) f(x_c) &(npalis)
        <RU(137.5) [ f(3.16*sqrt(i)) RL(90) F(0) f(ka_stand) palis(0) ]  >,

/*rule 2*/  palis(disttg) → palis(sqrt((gap_x−c_x)^2+(gap_y−c_y)^2)),

/*rule 3*/  (disttg>gap_r) palis(disttg) ⇒ P(cpalis) D(1.3) F(h_stand),
                           palis(disttg) ⇒ ,
```

The implementation of the juvenile stand (listing 18) is similar to the reconstruction of the real stand in chapter 5.2. In rule 1 the turtle moves to the centre of the gap. The tree itself is separated from the others using the bracket symbols which denote a branching. Within the “branch” the turtle rotates randomly and steps forward by the value r which is set by the user. The variable r can be a constant if only one tree is simulated and the position of the tree within the gap has to be controlled. If the gap shall be filled with various trees the constant radius can be replaced by a stochastic variable with e.g. a uniform distribution. If the turtle arrives at the position of the tree it has to erect. This

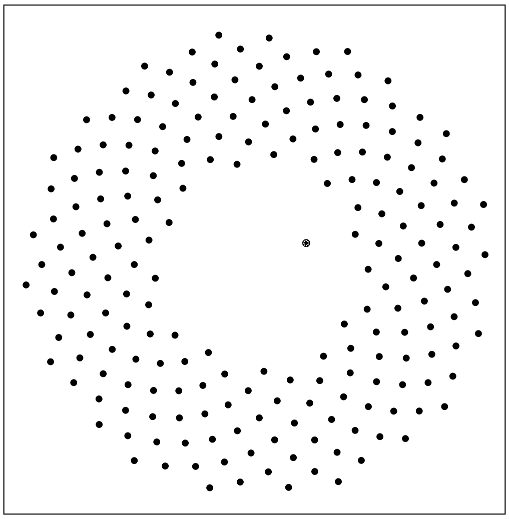


Figure 45: Result (top view) of the interpretation of the symbol *roof* as palisades (analogous to the spirals of phyllotaxis). The little dot in the middle is a Scots pine tree after 24 time steps (11 years) with a distance of 11 meters from the centre of the gap.

erection is expressed by a "rotate up" command. After rotating up the symbol *pinus* is introduced which represents the tree. In rule 2 the symbol *pinus* is replaced by *urstem* (see listing 18). It symbolizes an intermediate step during the growth of the trees. The reason is the implementation of the function *sky*. This function shall return information about the covering ratio and has to be applied using a reference symbol.

In rule 3 the tree is assembled. This rule is very similar to listing 14 in chapter 5.2. The symbol *stem* is followed by a sequence of commands which creates the main stem and the triangle representation. The three dots denote a sequence of commands for visualization purposes and are not of interest here. Rule 4 is the rule where the calculation of the covering ratio is taking place. The turtle is at the top of the crown. This is a difference to the *LIGNUM-canopy* simulation runs where the calculation is done at the bottom of the tree. In rule 5 the sensitive function 15 or 8 respectively is applied. The symbol *stem* is replaced if the function returns values greater than zero. The results are used to set the variables of *urstem*. The values are adapted from the behaviour of the *LIGNUM-canopy* tree (cf. table 30). The value has to be inverted because the regressions are based on the ratio of the open sky whereas here the value for the shaded sky is calculated. The variable *sky* contains the value of this ratio. To access this value repeatedly it is necessary to use the operator "_" (cf. Kurth 1999) to prevent a time consuming recalculation. The symbols *crext* and *crtop* have to be erased after every second step. Otherwise the symbols would be interpreted as a geometrical structure which would double the reference tree.

Listing 18: Implementation of the juvenile trees (generative rules).

```
/*rule 1*/ juven →
        RL−90 f(y_c) RU(90) f(x_c) [ RU(w) f(r) RL(90) pinus(i) ],

/*rule 2*/ pinus(ind) →
        urstem(ind,0,0,0,0,0),

/*rule 3*/ urstem(ind,h,d,cre,crt,t) →
        stem(ind,h,d,cre,crt,t) S(8+ind*9) crtop(crt,t)
        RU−90 RL(starta)
        &(8) < [ RL(i*45) crext(cre) S(i+9*ind) ] > RU(90)
        ...
        &(7) < T(i+ind*9,(i+1)+ind*9, 8+ind*9) >
        T(7+ind*9,ind*9,8+ind*9),

/*rule 4*/ (sky>0)  stem(ind,h,d,cre,crt,t) → urstem(ind,
                                     h   + (h_slo   * corr_cr*(1−sky_) + h_int),
                                     d   + (d_slo   * corr_cr*(1−sky_) + d_int),
                                     cre + (cre_slo * corr_cr*(1−sky_) + cre_int),
                                     h   + (h_slo   * corr_cr*(1−sky_) + h_int),
                                     t+1),

/*rule 5*/ crext(cre)    → ,
/*rule 6*/ crtop(crt,t)  → ,
```

Listing 19 shows the interpretive rules which create the geometrical structure of a tree. Rule 1 creates the stem with a diameter of *d* and a height of *h*. Rule 2 creates a base unit of the crown and rules 3 and 4 the central unit of the crown. The crown base has no continuous growth behaviour like e.g. the height. It suddenly increases after seven years from a height of zero to 0.3 meters. For this reason these rules are needed: Rule 3 is applied if the time step is lower than seven, otherwise rule 4 is utilised. The forward step is negative because the position of the turtle is at the top of the crown when this rule is applied, hence the turtle has to move back to the crown base. The colours are set to enable the transfer of the resulting data of the simulation results to *GroDisc*. The needle parameter can also be used for interfacing purposes.

Listing 19: Implementation of the juvenile trees (interpretive rules).

```
/*rule 1*/ stem(ind,h,d,cre,crt,t) ⇒ Dl(d)     P(7)  N(0) F(h),
/*rule 2*/ crext(cre)              ⇒ Dl(0.01) P(10) N(1) F(cre),
/*rule 3*/ (t<7) crtop(crt,t)      ⇒ Dl(0.01) P(11) N(0) F(−crt),
/*rule 4*/       crtop(crt,t)      ⇒ Dl(0.01) P(11) N(0) F(−crt+0.3),
```

8.2.4 Results of the L-System simulation

The following chapter presents the results of the simulation of juvenile Scots pines under a gap. The parameters for the simulation have already been presented in the previous chapter. Furthermore the L-system was introduced. The simulation is done twice using two different sensitive functions for the calculation of the covering ratio. Both functions are adapted by introducing a correction factor to the WCR of the *LIGNUM* calculation. It was of interest if function 8 and 15 have different influence to the simulation results.

In the beginning the results of the calculations of the sensitive functions 8 and 15 are given. Figure 46 (left side) gives the radiation ratios calculated by the sensitive functions 8 and 15 depending on the distance (X_0) of the tree to the centre of the gap. This figure corresponds to figure 42 in chapter 8.2.2. What this figure shows is that the palisade concept is less exact. The incoming radiation has its maximum not at the centre of the gap but at a distance of seven meters from the centre. The dimensions of the parameters also differs from the original data. The values of WCR as well as of CR are lower than the corresponding results of *LIGNUM-canopy* which are presented in figure 42. Only the relation between both covering ratios is similar. Both in the results of *LIGNUM-canopy* and in the results of the sensitive functions 8 and 15 the covering ratio CR has persistently lower values than WCR. The right side of figure 46 shows the adapted version of the covering ratios calculated by the sensitive functions. The results of these functions are multiplied by a constant factor (1.44 for WCR and 2.69 for CR). The factor results from the differences between WCR calculated by *LIGNUM-canopy* and WCR calculated by *GROGRA*. The resulting values are presented in figure 46 on the right side.

Table 35 gives the results of a regression analysis as in chapter 8.2.2. The results of this regression analysis are also illustrated in the appendix (figure 54 on page 170). Both serve the purpose of showing the quantitative differences between the two model concepts. The L-system model is parameterized by aggregating the results of *LIGNUM-canopy*. However, the results are different. The comparison of these values with table 30 in chapter 8.2.2 shows that all regression coefficients of the L-system simulation are lower than in the *LIGNUM-canopy* simulation runs. These differences and their causalities are discussed later on in the discussion section. Figure 47 gives the temporal development of the parameters height, height of crown base, diameter and crown extension. The development in time is linear. This behaviour is due to the aggregation as linear regressions. The position of the linear curves regarding the different light regimes differ from the original data from *LIGNUM-canopy*. Here, the tree in the centre of the gap is not the tree with the highest increment. The reason is the light model which is implemented in the sensitive

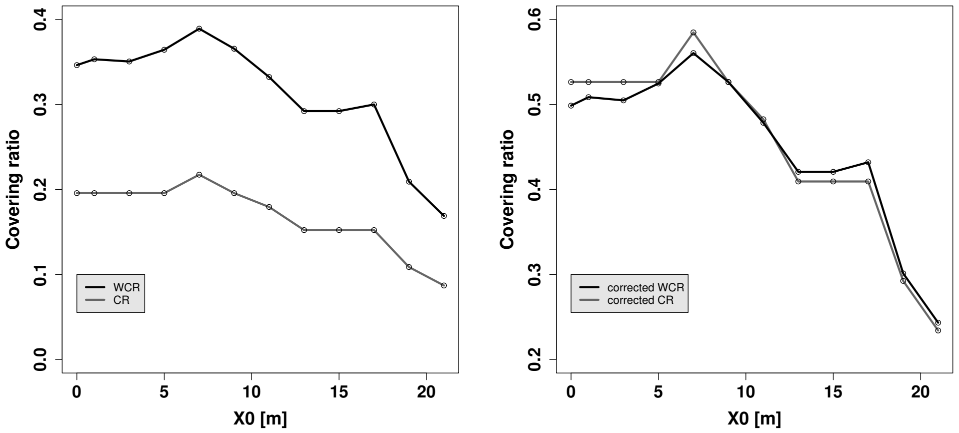


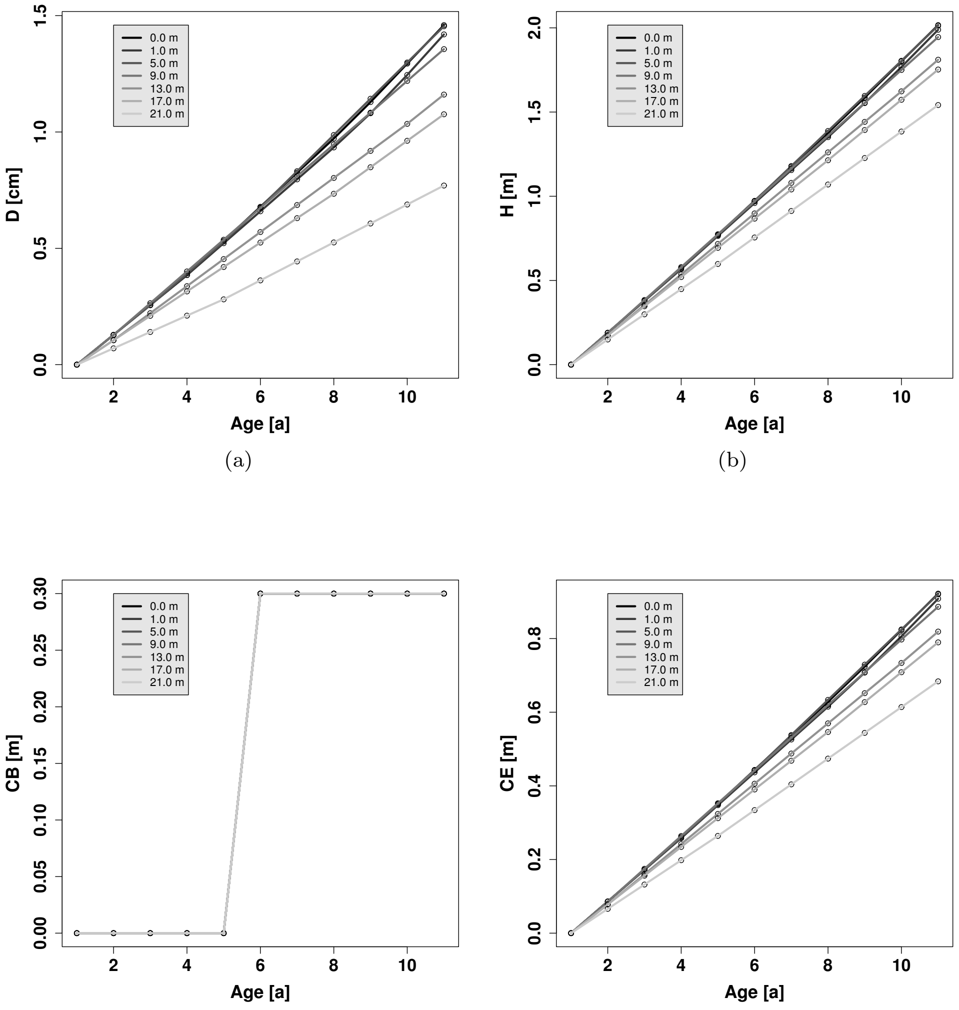
Figure 46: The covering ratios depending on the distance of the reference tree to the centre of the gap (X0) in the *GROGRA* simulation run. CR = ratio between the number of non shaded sectors to the total number of sectors (inverted result of sensitive function 8) and WCR = the same as CR but the number is weighted by the sine of the inclination angle (inverted result of sensitive function 15) (left side). The right image illustrates the results of WCR and CR corrected. The correction tries to adjust the covering ratios to the results of *LIGNUM-canopy* (see figure 42) (factors 2.69 for CR and 1.44 for WCR).

function 15. Figure 46 has already shown that the light gradient along the radius of the gap is not decreasing in a linear fashion. The maximum is at a distance of seven meters from the centre, hence the tree at that position has the largest increment for height, diameter and crown extension. The behaviour of this model changes if the position of the tree is moved to the edge of the gap. The development of the height of the crown bases differs. This is a result of the behaviour of *LIGNUM*. The crown base has no linear increment but suddenly changes the height after an age of 5 years. Figure 48 illustrates the quality of the simulation run. It shows the relation between the "predicted" results of the L-system simulation and the original data which resulted from the *LIGNUM-canopy* simulation run. The quality of the simulation results can be assessed by the shape of the scatter plots. Especially regarding the diameter, they show deviations in the lower values. These values result from growth conditions with less radiation income which occur for to trees at the edge of the gap. The reason is a difference in the profile of the WCR curves of *LIGNUM-canopy* and the sensitive function 15. The gradient of the *GROGRA* version is steeper, hence the light income is simulated as smaller than in the original data.

The table 36 and figure 50 have the same target as the previous figures and tables. They show the results of simulation runs with the aggregated version of *LIGNUM-canopy*. The results of the regression analysis are additionally illustrated in figure 55 on page 171.

Variable	Unit	Int.	Slope	R^2	p-value
ΔH	$[m/a]$	0.0957	0.1404	0.9332	<0.0001
ΔD	$[cm/a]$	0.0113	0.2040	0.9332	<0.0001
ΔCE	$[m/a]$	0.0402	0.0705	0.9332	<0.0001

Table 35: Linear regressions between height (ΔH), diameter (ΔD) at the bottom and crown extension (ΔCE) to the weighted covering ratio WCR calculated by the sensitive function 15. The p-value denotes the test whether the slope is not null.



(a) (b)

(c) (d)

Figure 47: The development of diameter (D0.0) (a), height (H) (b), crown base (CB) (c) and crown extension (CE) (d). The different colours of the lines denote different distances of the reference tree to the centre of the gap. The light regime (WCR) is calculated by the sensitive function 15.

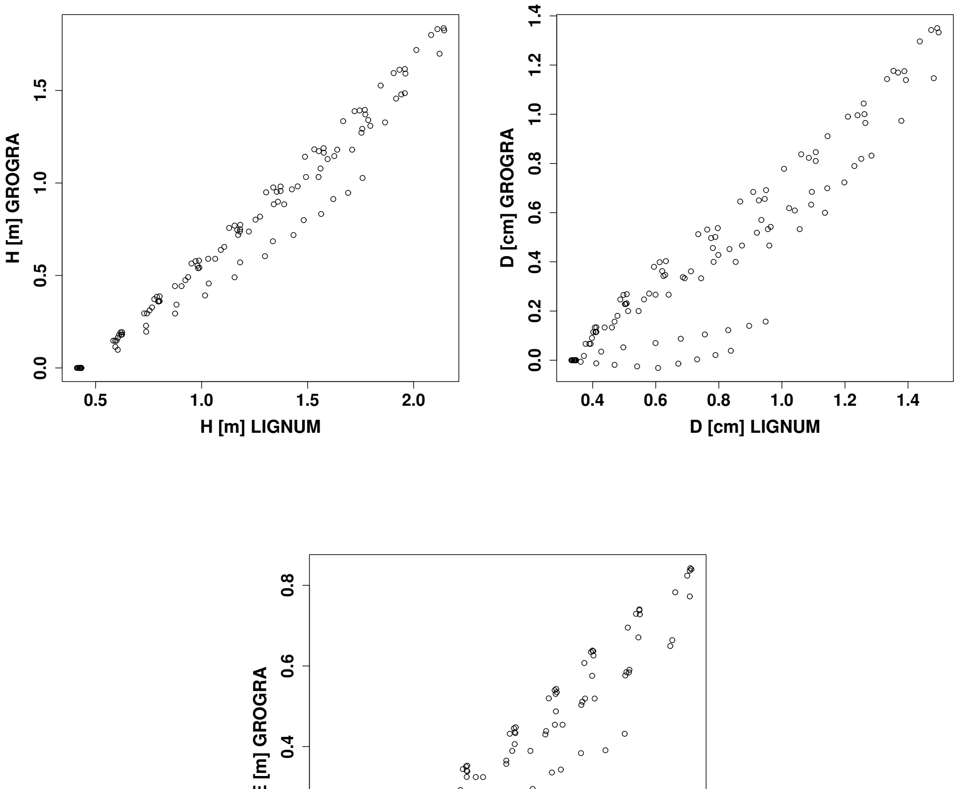


Figure 48: The results of the L-system simulation versus the original data of *LIGNUM*. The light conditions are calculated by sensitive function 15 (WCR).

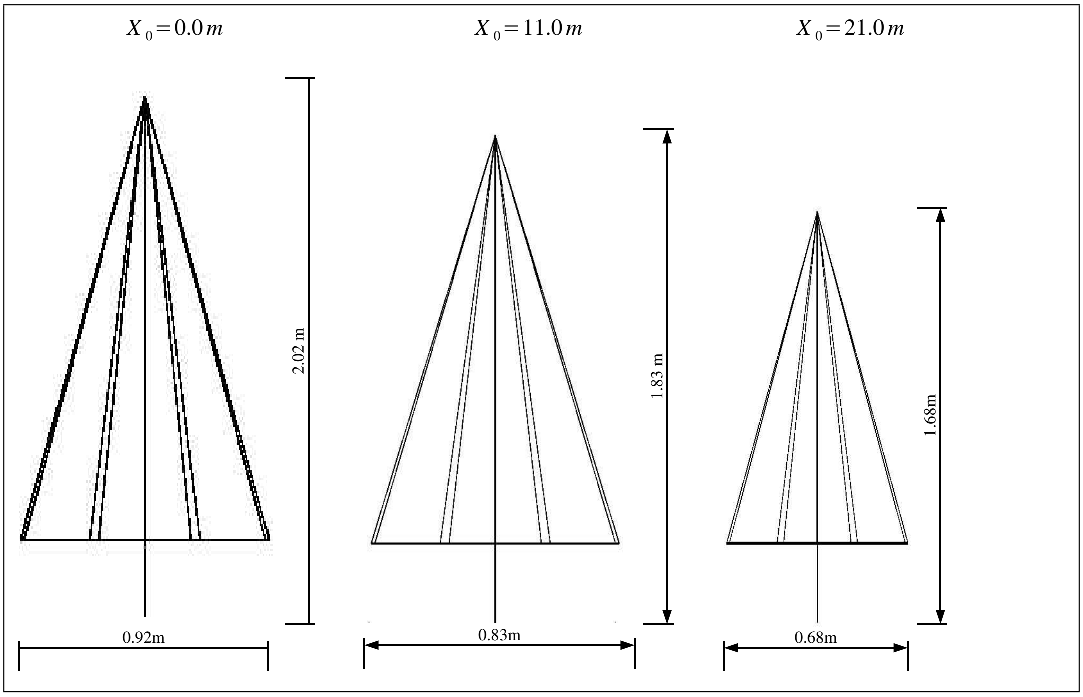
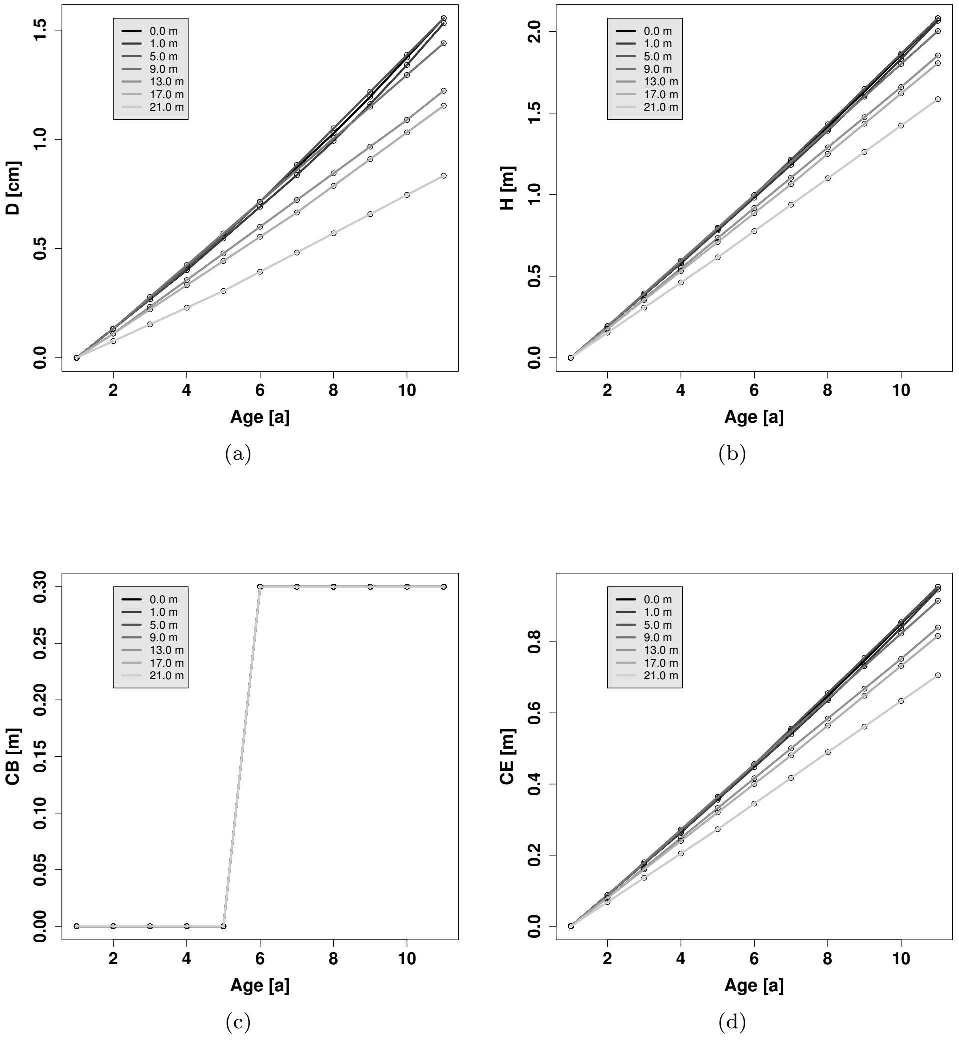


Figure 49: The shape of the resulting trees. Three trees from different distances are chosen: one from the centre ($X_0 = 0.0\ m$), one from an intermediate position ($X_0 = 11.0\ m$) and one from the edge of the gap ($X_0 = 21.0\ m$).

The difference here is that the simulation runs are applied using the sensitive function 8 corrected with the factor 2.69. It was of interest how the reaction of the model would be using CR instead of WCR. One would indeed expect that using the correction factor would lead to results similar to those of WCR (function 15). The results do indeed show such a behaviour for every parameter. This can be seen as a sensitivity analysis and gives information about the differences of both concepts. Also the scatter plots which image the quality of the simulation (figure 51) are similar to the WCR version of the simulation.

Variable	**Unit**	**Int.**	**Slope**	$\mathbf{R^2}$	**p-value**
ΔH	$[m/a]$	0.1033	0.1327	0.9502	<0.0001
ΔD	$[cm/a]$	0.0223	0.1929	0.9502	<0.0001
ΔCE	$[m/a]$	0.0440	0.0666	0.9502	<0.0001

Table 36: Linear regressions between height (ΔH), diameter (ΔD) at the bottom and crown extension (ΔCE) to the covering ratio CR calculated by the sensitive function 8. The p-value denotes the test whether the slope is not zero.



(a) (b)

(c) (d)

Figure 50: The development of diameter (D0.0) (a), height (H) (b), crown base (CB) (c) and crown extension (CE) (d). The different colours of the lines denote different distances of the reference tree to the centre of the gap. The light regime (CR) is calculated by sensitive function 8.

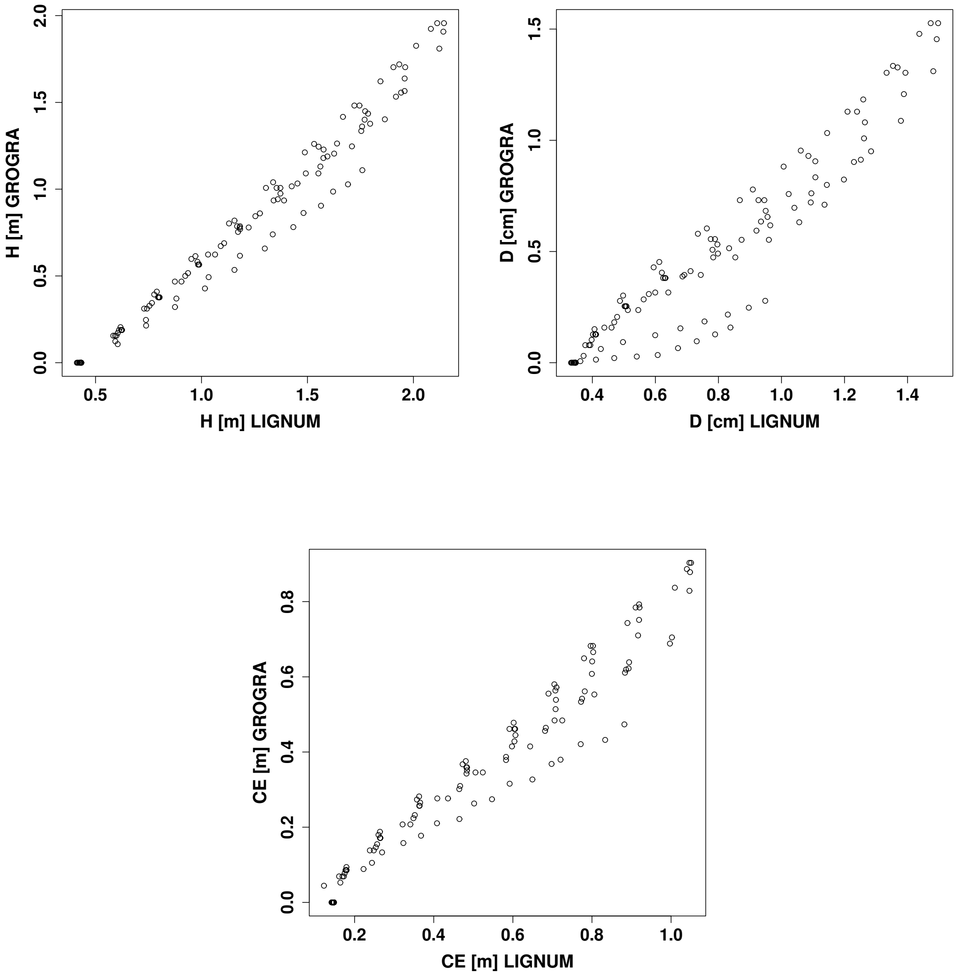


Figure 51: The results of the L-system simulation versus the original data of *LIGNUM*. The light conditions are calculated by sensitive function 8 (CR).

8.2.5 How the gap fills with trees: a model experiment

The following figures show the gap filled with 50 trees. The trees are distributed over the gap in a random manner. The turtle moves to the centre of the gap and rotates. The angle is chosen from a uniform random distribution. Also randomly distributed are the values for X_0 which result from the forward movement of the turtle (variable r in listing 18). Figure 52 illustrates the results. It is a top view of the gap after 24 time steps. 24 time steps correspond to an age of eleven years. Table 37 shows the mean values of the height, diameter, crown extension and the height of the crown base after 11 years. The height of the crown base is fixed at 0.3 meters. This is the reason why the standard deviation is zero and the minimum value equals the maximum. At this point it would be interesting to assess how realistic these values are. In comparison with the measured trees in chapter 5 the values are on the whole correct. Unfortunately not enough measured values are available for verification. Therefore a greater collection of trees would be necessary than only three randomly chosen trees without any information about the light conditions in which the trees grew up.

Variable	Unit	N	Mean	Std	Min	Max
H	$[m]$	50	1.86	0.133	1.57	2.05
D	$[cm]$	50	1.23	0.194	0.81	1.51
CE	$[m]$	50	0.84	0.067	0.70	0.94
CB	$[m]$	50	0.30	0.000	0.30	0.30

Table 37: Results of the simulation run with 50 juvenile trees in the understorey of a gap.

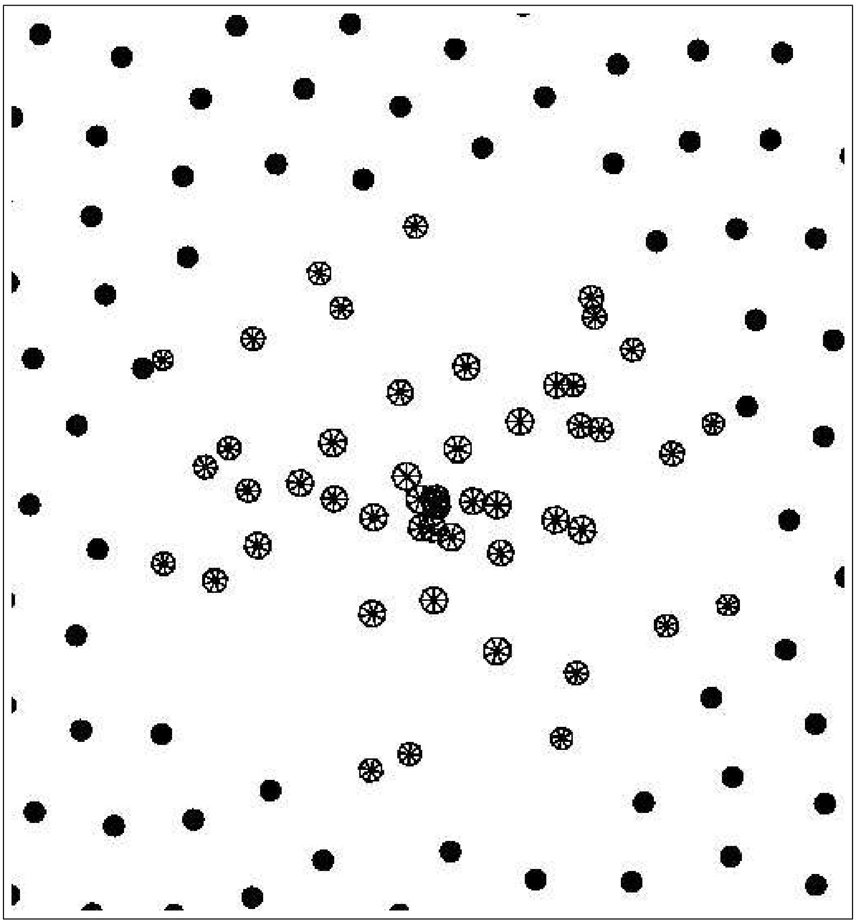


Figure 52: The simulated juvenile stand within a gap (top view). The trees have gray colour. The positions are randomly distributed over the gap. The black points which are arranged as a spiral are the palisades which represent the aged canopy.

9 Discussion

In the scope of ecological modelling many models are existing which try to image the growth of trees and tree stands. The concepts of these models follow different aspects of information which they try to process. Kurth (1999) suggested a so-called model triangle to characterize the different models with the type of information the models process. It differentiates models by the level of aggregation and structural and functional information. The focus of this work lies on individual-based tree stand models as they are used in economically motivated tools like *SILVA* (Kahn and Pretzsch 1997, Pretzsch 2001) or *BWIN* (Nagel et al. 2000). These models represent tree crowns by simple geometrical bodies like e.g. irregular pyramids or spheres. The functional information of these models is low, hence they are located in the triangle of the models more at the left base (cf. figure 1). This thesis tries to introduce L-systems in the scope of simplified individual-based tree stand modelling. Usually such models are implemented using the procedural or object oriented programming paradigm. The main disadvantage of these paradigms is less scientific transparency. L-systems introduce an alternative to these paradigms. The use of L-systems is shown on two examples: a reconstruction of a Scots pine tree stand in Syke near Bremen and a model for juvenile Scots pines which is due to an aggregation of the model *LIGNUM* (Perttunen et al. 1996, 1998). For the analysis of a part of reality as well as for the simulation results a tool in form of a software must be at hand. In this work the tool *GroDisc* is developed which aids this work. The problem was to handle, analyse and discretize 3D architecture. The following chapter discusses the structure and assesses the functionality of *GroDisc*.

9.1 *GroDisc*

The software *GroDisc* was implemented as an aid in modelling work. To get a retrospective view: the work of modelling comprehends four different steps — analysis of reality, creation of the model, implementation of the model and analysis of the simulation results. *GroDisc* was created to assist the analysis part of the modelling work. Thus *GroDisc* analyses reality and virtual reality. The importance of creating a tool which handles real 3D data will be discussed. Furthermore *GroDisc* was very helpful to transport data between different simulation and analysis tools, hence an assessment of these interface modules is given in this chapter.

9.1.1 The data structure

The data structure of *GroDisc* had to represent 3D architecture. Architecture consists of three different types of information: decomposition, geometry and topology (Godin 2000). In principle, the problem of representing architecture has similarities to the concept of *GIS*. The main aspect of developing one's own software instead of using existing *GIS* is the representation of real 3D structures. There is no sufficient *GIS* available which provides the handling of real 3D information. The development of real 3D *GIS* is in a preliminary stadium (cf. Pfund 2002). Thus the data structure of *GroDisc* presents 3D architecture but is abutted to the data structure of *GIS* (see figure 14 on page 34).

The data structure of *GroDisc* depicts the decomposition, geometrical and topological information in a transparent and useful way. The only disadvantage is the strict thematic detachment of the primitives. A tree with simplified crown structures like that of Pretzsch (1992a) is represented by two different types of geometrical bodies (one cylinder for the main stem and an irregular pyramid which can be represented by triangles). It is not possible to represent a tree which is decomposed into a cylinder and triangles in *GroDisc*. The information of such an entity has to be kept within two different layers which causes the necessity of intersection operations. Very useful was the detachment of attribute from the architecture information. The implementation of the attribute table as an *STL* map increased the flexibility in the programming work. It was very easy to add new attribute fields without changing the class structure itself by adding variables. This concept of detaching the attribute information technically facilitated the outsourcing of the attributes to a data base. The data base concept of the attribute information is in a preliminary state. A *MySQL* data base is existing which currently serves as the interface between *GroDisc* and the statistical software *R*. The enhancement of the *GroDisc* data base concept will be the next developmental step in the future.

9.1.2 The algorithm concept

Creating algorithms for analysing a tree stand structure requires the traversing through the structure and the application of instructions to list elements. In *GroDisc* the traversing is provided by functions. This concept is also due to the *STL*. Functions like "ForEach" or "Accumulate" traverse through the structure and apply functions or functors to the elements of the list structure. The instructions are passed to these traversing functions. The shape of these instructions can be either a function or more complex figures like functors. Functors can be either classes or structures.

The disadvantage of such a concept is that it takes some time for the programmer to get used to it, especially if the basic list structure is complex. A good example for such a complex structure is the *STL* version of *LIGNUM.* In this version, for instance, traversing algorithms are provided which traverse quasi simultaneously through axes with the same Gravelius order ("Propagate"). Such functions are rather useful but difficult to understand. Nevertheless, once the programmer got used to it the time requirement for developing analysis algorithms of 3D structures significantly decreases. The functor concept also aids the scientific requirement to be transparent. The aspect of transparency is very important within working groups. This is the reason why Perttunen (oral communication) reimplemented *LIGNUM.* Such a concept may avoid the separation of software within a working group. This experience which arose in the plant modelling group of Sievänen and Perttunen let arise the idea for implementing the analysis tool *GroDisc* using a similar concept of programming. This concept is illustrated in chapter 3.3.3. In the *GroDisc* concept some differences to *LIGNUM* are existing. *LIGNUM* additionally offers traversing functions like "PropagateUp" which traverse through the axis structure in a quasi simultaneous way. Such functions are not available in *GroDisc* because they are adapted to the special data structure of *LIGNUM.* However, *GroDisc* provides functions which allow to apply functors to pairs of compartments ("AccumulatePairs", "ForEachPair"). These functions are very useful for calculating minimum distances. An example is the calculation of a minimum distance between a tree unit and a Triangle analogous to function 14 of *GROGRA* (for the code see appendix). It is applied using the traversing function "AccumulatePairs". This functor is a good example for the comparison between the procedural concept of *GROGRA* and the object oriented concept of *GroDisc.* The calculation of the distance itself is indeed the same (function 14 of *GROGRA* and function "Distance" in *GroDisc*). Only the syntax is a slightly different. However, the *GroDisc* notation additionally shows the functor which is applied by the traversing function "ForEachPair". This functor demonstrates how elegant the shape of such a functor can be. Another example regards the translation of a *GroDisc* tree into an *MTG* structure. That functor exemplifies the data interface to *LIGNUM* and the way a functor works. The functor accomplishes two tasks: the calculation of an identifier and the translation to an *MTG.* The calculation of the identifier shows the limits of the *STL* concept. The reason is that *LIGNUM* as well as *GroDisc* does not offer functions which traverse the structure in the right way for the special calculation. More simple is then the translation process itself. At the moment the functor is applied to a tree segment all necessary information is available. The programmer is not forced to deal with code concerning traversing.

To give a conclusion: The *STL* concept of traversing through list structures needs the precognition of at least a procedural programming language or better an object oriented language. Nevertheless, the programming code attains clearness, hence transparency. Such a transparency is very important if a software is used or developed by several persons like in working groups.

9.1.3 Interfaces [†]

Interfaces between software tools for creating, analysing and manipulating virtual plants are useful for comparison of model results with field data and to test theoretical analysis approaches. An exchange of data between software systems broadens the spectrum of available methods for validation and comparison between models. Because different teams of researchers have their specific views and approaches, an exchange of data and the application of tools developed by another team can help to detect weaknesses of specific models which are easily overlooked by one team alone. The accessibility of a larger database of plant architectures (measured and simulated) will also provide better possibilities to validate functional-structural tree models. The chapter about the *GROGRA-LIGNUM* interface has described how an interface between different tools can be constructed. Exemplarily some of the possibilities of data analysis have been shown, but also some of the difficulties, which can arise when different research tools for plant structures are interfaced, could be pinpointed.

The main technical problem was to specify an appropriate filter which makes data from *LIGNUM* accessible for *GROGRA*. Because *GROGRA* provides less physiological information than *LIGNUM*, there is no one-to-one correspondence between the elementary data structures of both softwares although they are very similar in their way to represent tree architecture. However, the unavoidable loss of information did neither affect the application of analysis methods included in the *GROGRA* system, nor did it interfere with the hydraulic simulation performed with *HYDRA* on the basis of *GROGRA* structures.

9.2 Reconstruction of the Scots pine stand

The reconstruction of the Scots pine stand which is located in Syke near Bremen aimed at showing how such a reconstruction can be implemented using an L-system. It was intended to show that models like that of Pretzsch (1992a) are able to be implemented by L-systems. This idea was motivated by the observation that models like Pretzsch (1992a) or Nagel et al. (2000) are excellent tools for forecasting the growth of trees especially under

[†] Dzierzon et al. (2003) slightly changed

changed environmental conditions and in mixed forests. However, the use of procedural or object oriented programming languages in the implementation of models decreases the transparency. In this work an alternative which offers a probable solution is introduced.

9.2.1 Correlations and regressions

Before a reconstruction could be implemented the reality had to be analysed. The analysis was performed by a correlation and regression analysis. The correlation analysis should confirm the hypothesis that the parameters height, crown base and diameter are directly or at least indirectly depending on the crown extension. Such relations should open the possibility to calculate the tree dimensions due to crown distance sensitivity. From the beginning the target was not to create a comprehensive and general model like *SILVA* (Pretzsch 2001) or *BWIN* (Nagel 1999). Such concepts can not be implemented due to the lacking data base.

The hypothesis could be confirmed for the static situations in 1995 and 2001, as table 15 on page 56 shows. The only exception is the relation between the diameter and height of crown base, but the height of crown base can be explained by the tree height. However, the relation which is due to the hypothesis vanishes if the increments of the parameters are considered. It was clear from the beginning that the relation of the parameter increments is less concise than that of the static situations in 1995 and 2001. The reason is that an increment is a combination of two random variables, which extends the dispersion. Nevertheless, the complete lack of precision was surprising. The reason for that might be precision problems of the measurement as they have been discussed in chapter 5. However, even ignoring the question of the precision of measurement, it seems that it is not possible to calculate relative increments without considering an initial situation. This knowledge is not an original output of this work: it has already been mentioned by Nagel (1999) and Pretzsch (2001). The advanced correlation and regression analysis additionally confirmed that fact (see table 16 on page 59). It was possible to calculate the annual increment for each parameter from the initial situation in 1995. However, the target of this reconstruction was not to create a new approach for simulating Scots pines but to give alternatives for the implementation of such models.

9.2.2 Distance sensitivity

Part of the investigation of the Scots pine tree stand was the distance sensitivity between Scots pine crowns. Pretzsch (1992a, 2001) has postulated such sensitivity for Norway spruces. He hypothesized that Norway spruce trees reduce their crown extension if the crowns overlap with their neighbours.

In this work it was tried to reproduce this model for the Scots pine stand in Syke near Bremen. It was not possible to verify this model for the Scots pine species. Only the regression of the potential crown increment showed significant results with a coefficient of determination of 0.18 (see table 19 on page 61). The nonlinear fitting for the relative crown radii increment failed. Figure 25 on page 62 shows that the shape of this curve is not reasonable. The function was developed for the retreatment of crowns. The data do not show such a retreatment, hence the curve is not confirmed by the data. The data only show a normal distribution of the distances in 1995 and 2001. The t-test suggests that the origin of both distributions is the same population. The consideration of all distances showed a distance sensitivity which can be expressed as a maximum degree of overlapping such that all growth stops if it is exceeded. The minimum distance is seen as the mean value minus the standard deviation of the overall distance distribution.

The reason for the missing retreatment of Scots pine crowns if the crowns overlap can not be discussed considering only one reason. The first point is that it is not possible to compare the results with the publication of Pretzsch (1992a) because he did not specify the quality of the regression which describes the relative increment of the crown radii. But beside this there are some possible explanations. The first is that both approaches differ concerning the species and the crown radii. It is indeed possible that Scots pines simply do not have the behaviour of reducing their radii in the case of overlapping. The practical impression is that Scots pine crowns are not that dynamic. In his regressions Pretzsch (1992a) explicitely used the values for the distances in each direction. The regression which is applied here used averaged values over the eight directions. Another reason might be edge effects. The plot is located within a surrounding tree stand. Distances in the direction of the neighbour trees which are not part of the plot might disturb the calculation of regressions.

The interpretation of these results has to be done carefully. The data base is very small and the temporal distance between the measurements (1995 to 2001 = 6 years) is also small. Nevertheless, the reconstruction of the Scots pine trees was successful. The result of the simulation was not essentially different from the measurements in 2001. The simulation has additionally shown that L-systems are useful to implement even reconstructions which contain distance sensitivity like the approach of Pretzsch (1992a,b, 2001). The implementation of such approaches as L-systems has a special importance in this work, hence a special chapter is dedicated to this aspect and will be discussed together with the L-systems which were implemented for the *LIGNUM* model aggregation.

9.3 Plausibility and sensitivity of *LIGNUM* ‡

Results were presented which imply a comparison between a simulated and a real Scots pine tree. These comparisons were not designed to verify or falsify the *LIGNUM* model for Scots pine. The measured sample trees were from a site in Germany, whereas *LIGNUM* was parameterized for trees grown in Finland, i.e. under distinctly different climatic conditions. Furthermore, our comparisons were not systematical enough: A larger number of sample trees would have been necessary to yield statistically secure results. This part of our work was mainly meant to demonstrate some of the possibilities which are opened by interfacing different software tools. Other interfaces, e.g. between *GROGRA* and *AMAPmod* (Godin et al. 1998) provide even more possibilities to analyse *LIGNUM*-generated trees (Dzierzon and Kurth 2002). Zeide and Pfeifer (1991) found values of fractal dimension greater than two and lower than three measured at Rocky Mountains conifers in South Carolina (USA). Their values are thus much larger than ours. Unfortunately it is not possible to compare these results directly. Zeide and Pfeifer (1991) used a so called "two surface method" to calculate what they denoted as "fractal dimension". This method differs strongly from the box counting method and uses differences in the sizes of the given trees as the only source of variation of scale. Another difference is that we only calculated the dimension of the tree skeleton without leaves (needles). Furthermore our investigated trees are much smaller. Zeide's and Pfeifer's trees have diameters greater than 5 cm compared to ours with a maximum diameter of 3.12 cm (see table 27 on page 77). However, our values of fractal dimension, obtained with the box counting method only, showed good coincidence between simulated and measured trees. In simulation-derived water potential profiles, there was again a high degree of qualitative similarity between simulated and architecturally measured trees (we emphasize that we did not measure the water potentials directly). Quantitative differences were, like in diameter analysis, mainly due to differences in the distribution of diameters along the woody axes: The tree simulated by *LIGNUM* had a relatively weak secondary growth. This also explains the low simulated water potential values: *HYDRA* uses an empirically derived diameter-conductivity relationship where the diameter of a segment appears in the formula with an exponent of 2.41 (from Cochard 1992). Therefore, a small reduction of diameter can already cause a considerable reduction of axial hydraulic conductivity, which will normally lead to lower potentials during simulation. In Chiba's (1990, 2000) cross-section area vs. biomass relationship test (see figure 33 on page 83), the results of measured and simulated trees differed. The difference between the shapes of the curves

‡Dzierzon et al. (2003) slightly changed

was mainly caused by segments belonging to the main stem of the tree. Nevertheless, this difference is remarkable, since Chiba's approach and the mechanistic growth model of *LIGNUM* (see Perttunen et al. 1996) were both inspired by the "pipe model" of Shinozaki et al. (1964). In the analysis of Chiba, the total cross-sectional area of stem and branches were used. *LIGNUM* employs a stipulation of the pipe model (Nikinmaa 1992) in which the cross-sectional area of sapwood instead of total cross-sectional area is used. According to Figure 33 on page 83, Chiba's interpretation of the pipe model seems to correspond to the data of this study better than that used in *LIGNUM*. This analysis shows that caution has to be applied when general ideas like the "pipe model" are used in specific models. A specification of complex models, like models of carbon allocation in trees, must therefore be precise and should not rely on ambiguous terms like "pipe model". Such a term can be understood and applied in many ways (e.g. for a tree as a whole or for junctions of tree parts as in *LIGNUM*). It should also be kept in mind that the observed crown level or branch level pipe model relationships are also affected by the rate of senescence of the foliage and by heartwood formation. The effects of heartwood formation have been studied in the framework of *LIGNUM* (Sievänen et al. 1997).

We compared results of *HYDRA* visually, i.e. in a qualitative and limited way of comparison. To go one step further would mean to quantify the differences, that is, to calculate distances between two simulations of water potential in tree crowns. It is not an easy task to define such distance measures. A purely graph-theoretical distance like that proposed by Ferraro and Godin (2000) for plant architecture is not sufficient because the sizes of the water potential gradients play also an important role in our case. Perhaps a combination of graph-theoretical and physical distances would have to be defined and applied here.

Concerning interfaces between software tools, some general remarks can be made. Bridges between different software systems can help to spare much time which is normally necessary to implement a simulation model or generic analysis tools. Systems like *AMAPmod*, *LIGNUM* or *HYDRA* have cost many man-years to develop them, and sharing their use is therefore a question of economy in research. On the other hand, it is also clear that learning and understanding a specific method can be much more thorough if a research team manages to create its own software tool incorporating the method and experiences the intrinsic difficulties of the method during the process of software development and debugging. Here, a balance between both requirements — economy and "learning by doing" — has to be found by each team.

9.4 Sensitivity of *LIGNUM-canopy*

LIGNUM-canopy is used to simulate Scots pine trees in the understorey of forest gaps. It is not the first time that *LIGNUM* is used for such a purpose. Perttunen et al. (2001) simulated sugar maple trees under the radiation conditions of a gap. The approaches in the simulations with *LIGNUM-canopy* and in the investigations by Perttunen et al. (2001) are quite different, not only regarding the species. Perttunen et al. established variants of light conditions in the understorey of a gap. These variants of light conditions are due to fish eye photos, hence purely empirical. *LIGNUM-canopy* explicitly calculates the incoming radiation by the travel distance through the surrounding canopy. In Perttunen et al. total incoming radiation reaching the understorey varied from 5.6 to 26.6 %. *LIGNUM-canopy* calculated lower percentages of light reduction (55 to 45%) which are equivalent to 45 to 55% of light which reaches the understorey. The conditions in Perttunen et al. (2001) are probably quite different. Especially the radius of the gap in the *LIGNUM-canopy* simulations is very big.

A varying radius can affect the light regime particularly when the radius gets very small. In both variants the tree will nearly die if the radius of the gap converges to zero. The tree still grows but the total production of carbon converges to zero. This fact supports the plausibility of the model. Scots pines are shade intolerant, hence it was expected that the tree stops growing at least under a closed canopy. That is the reason why such a large radius of the gap was chosen during the further simulation runs with *LIGNUM-canopy*.

During the sensitivity tests with *LIGNUM-canopy* the parameters a_f and f_c were varied. The tests should show the influence of these parameters on the simulation results. Figure 36 on page 87 shows the results of variant 1 and figure 37 on page 88 those of variant 2. The variant 2 resulted in greater trees. This conclusion is valid for each parameter which is illustrated. It seems that a reduced needle mass/tree segment area relationship is more than compensated by an increased needle coverage. The differences between both variants are not very large, hence it was difficult to decide which variant was the most reasonable for further simulations. The parameter combination of variant 1 was used in Perttunen et al. (1998), hence that one was chosen. In such situations it would be useful to have measured data available for comparison.

Very noticeable in both variants is the number of shoots of the resulting trees. This number is enormous compared with the real measured German trees. In principle this behaviour is controllable by an input function which controls the number of new buds in relation to the foliage mass of a mother shoot. However, this relation is due to empirical values (see Perttunen et al. 1998). In a future step it would be necessary to adjust this input function.

9.5 Modelling of juvenile Scots pine trees as an aggregation of *LIGNUM*

A rule based model is introduced which is intended to aggregate results of the model *LIGNUM*. The trees are grown under the light conditions in the understorey of a gap. For that reason the light model extension *LIGNUM-canopy* is used. The creation of this model has been done in four steps: aggregation of the light model, aggregation of the growth of the juvenile tree, implementation of the aggregated version as an L-system and at last the comparison of the simulation with the original data.

9.5.1 Aggregation of *LIGNUM-canopy*

The main target of the aggregation was a new simplified model with special aspects of the *LIGNUM* growth behaviour. Aggregation always means the loss of information. In the scope of the model triangle of Kurth (1999), the lost of information means an adjustment. Here the physiological information which is simulated by *LIGNUM* is abstracted in the aggregation process. Thus in the model triangle the new model moves to the left side upward. The newly created model contains almost only architectural information. The only functionality which is held is that the architectural structure depends on the radiation income depicted by a hemispherical covering ratio.

The assessment of these results has several aspects. The first regards the simulation results of *LIGNUM-canopy*. Some aspects have already been discussed in the section about the sensitivity tests of *LIGNUM-canopy*. The simulations for instance also result in an enormous number of segments (see figure 40 on page 93). The dimensions of the other parameters (H, D, CE and CB) compared to the real measured trees in chapter 7 turned out to be reasonable. Nevertheless, to validate the growing behaviour of the simulated Scots pines, more real data, especially in combination with light measurements, would be necessary. One observation is indeed that the simulated trees react in a manner proportional to the reduction of the radiation income. The ratio RAD varied from 0.55 in the centre of the gap to 0.42 at its edge which is a relative reduction of 23.7%. The height of the tree is reduced by 21.1%, whereas the diameter reacted very sensitive to the light conditions with a reduction of 43.6%.

The other assessment aspects regard the aggregation process. This process had two tasks: the aggregation of the light model of *LIGNUM* and the aggregation of the growth of the *LIGNUM* tree. The light model is simplified by using covering ratios. The ratio expresses the ratio of incoming radiation to the total radiation emerging from the hemisphere. This ratio was calculated at a reference point whose distance to the centre of the gap could be varied. It could be shown that this ratio decreases in relation to the distance of the reference point to the centre of the gap. The ratio RAD which expresses the ratio of incoming light calculated by *LIGNUM* showed that using a radius of the gap of 21 meters results in a light reduction of 23.7%. The other more simple ratios reflected this reduction in a stronger way. The ratio CR which only counts the non shaded sectors has a similar range but persistently calculates a very low radiation income. The ratio WCR which weights the non shaded sectors by their inclination angle corrects the level of light income near the centre. The ratio has a greater range and results in a lower radiation at the edge of the gap than the ratio RAD. The reason for such a behaviour is the counting method used in the ratios CR and WCR. In the calculation of RAD the canopy does not reduce the light of a shaded sector to zero which happens with the counting method of the ratios CR and WCR.

To aggregate the growth behaviour of *LIGNUM* Scots pines the relation between the light conditions and the annual growth was investigated. It was very easy to constitute a linear regression with high coefficients of determination between both. For measuring the light income the ratio WCR is used. The sense of using this ratio instead of RAD is that the aggregation should reduce the information about the amount of light which emerges from a sector. RAD includes more of such information. Additionally, this ratio reduces the light income from lower inclination angles by weighting the sector with the inclination angle which seemed to be reasonable.

9.5.2 The implementation

The aggregation of *LIGNUM-canopy* resulted in a new model. This new model is implemented using an L-system. Several problems had to be solved for this implementation: the representation of the light model, the representation of the canopy and the representation of the individual juvenile tree.

The light conditions are calculated by the sensitive function 15. This function returns a ratio which is equivalent to the inverted version of WCR. The used model of the hemisphere is due to den Dulk (1989). Figure 46 on page 103 gives the results. This figure also shows the results of the sensitive function 8 which returns the inverted equivalent to CR. The comparison with the calculations of *LIGNUM-canopy* showed strong differences: The *LIGNUM-canopy* calculations were always higher. The reason for that behaviour may lie in the differences of the light models. Both *LIGNUM* and the approach of den

Dulk divided the hemisphere into sectors and tried to keep the area of all sectors more or less evenly sized (turtle approach). The main difference between both approaches is the number of sectors. *GROGRA* uses only 46 sectors. The firmament of *LIGNUM* is in principle variable but was parameterized with 120 sectors. This very discrete way of parameterizing the sky may cause differences in the calculation of the ratios.

The canopy is represented by a spiral of palisades. This representation was implemented only as an improvization because no geometrical body like a box was available in *GROGRA* which would represent the canopy more reasonably. The problem of this representation was that one could not guarantee that some light beams would not "cheat" through the canopy. Such light beams might be the reason for the peculiar maximum of CR as well as WCR at a distance of seven meters from the centre of the gap of the *GROGRA* version (see figure 46 on page 103).

The representation of the individual juvenile tree was very similar to the trees of the reconstruction of the tree stand in Syke. The only difference is that the juvenile trees do not use distance sensitivity. In contra to the reconstruction, the shapes of the *LIGNUM* trees were imaged very well by the irregular pyramids. Here, no maximum height of the crown extension would bring any advantages in the representation.

9.5.3 Results

The aggregation of the *LIGNUM-canopy* Scots pines should show that it is possible to implement such a model as an L-system, which is discussed in the last chapter below. Nevertheless, another target was to show that is is possible to simulate juvenile Scots pines by aggregating an existing *FSTM* instead of creating a purely empirical model. The validation of such approaches is very important as for instance Sterba (1990), Pretzsch (2001), Nagel et al. (2000) and Pretzsch and Ďurský (2001) noticed. A final assessment of the question if the simulation of juvenile trees was successful is not possible because the data for a quantitative validation are missing. However, first the model results have to be compared. Figure 48 on page 105 illustrates the differences between the *LIGNUM-canopy* results and the resulting values of the aggregated version. In the whole view the aggregation was successful. The scatterplot shows indeed some parts where the aggregation differs from the original model. Especially the diameters show different values if the incoming radiation is strongly reduced. The explanation for this lies in the step from the *LIGNUM-canopy* calculation of the ratio WCR to that of *GROGRA*. The shape of both curves is different even in the *GROGRA* version with a correction factor. The gradient of the *GROGRA* version of WCR is steeper. This results in smaller dimensions of all parameters in the aggregated version of *LIGNUM-canopy*. A solution would be the integration of the *LIGNUM* light model into the L-system specification of *GROGRA*. Such an approach would not significantly increase the demand on computer resources.

Within the simulation runs function 8 was used to clarify the difference between the use of CR and WCR in the aggregated version. The correction factor for CR adapted the curve almost completely to the WCR curve (see figure 46 on page 103). The simulation results by applying the sensitive function 8 were also very similar. The conclusion here is that the use of WCR does not enhance the light model calculation. The light model could be even more reduced to CR.

9.6 The use of L-systems to simulate simplified crown structures

The L-systems of this work have the characteristic that the resulting string describing the stand keeps its length after the virtual trees are established (after three steps in the reconstruction and four steps in the model for juvenile Scots pines). Usually the process of rule application lets increase the length of the string. This behaviour is caused by the concept of the crown representation. Growth means here not the increment of the structure by adding new elements, but the increment of dimension of the existing geometrical elements. The constant string length has the advantage to save computer resources, hence calculation time. The simulation of the juvenile Scots pine tree stand with 50 trees, for instance, consumed only 10 seconds over a period of 11 time steps.

The L-system specification of *GROGRA* allowed in a very flexible way to create the geometrical structure of the individual tree as a combination of elementary units and triangles. The triangles represent the irregular pyramid. The elementary units have different tasks. One represents the main stem of the tree, others are part of the crown. Eight elementary units for instance build the crown base units and one the central unit of the crown. This representation was able to create a complete tree stand structure. A greater number of geometrical bodies (primitives) would increase the flexibility of the L-systems. If, for instance, the crown should be represented by a sphere, this would not be possible at the current developmental state of the L-system specification. However, the L-system specification is currently being extended (Kurth oral communication).

The model for juvenile trees and the reconstruction of a tree stand use the concept of sensitive functions within the L-system. The reconstruction contains a distance sensitivity between different crowns and the model for juvenile Scots pine trees comprehends a light model which images the light conditions of the understorey of a gap. Basically these sensitive functions contradict the rule based paradigm because they are implemented using the procedural language *C* . Nevertheless, these functions are directly implemented into

GROGRA and differ from the modular environment functions of L-studio (Prusinkiewicz et al. 2000). The use of these sensitive functions within the L-system is easy and intuitive.

A special reason for using L-systems lies on the aspect of transparency of the implementation of models and reconstructions. This work additionally focuses on models which depict the reality of tree stand growth by individual based models where the tree crowns are represented by simple geometrical bodies. Procedural and object oriented implementations of models have an enormous amount of computer code lines. It is not known how many lines tools like *SILVA* or *BWIN* have but they are assumed to be enormous. The L-system for the reconstruction needed 13 rules and 62 lines of L-system code; the L-system for juvenile trees needed only slightly more, namely, 16 rules and 73 lines. This very condensed notation allows to publish the model at least in a thesis like this. It needs certainly time to get used to the concept of L-systems for implementing models. But to repeat it again: It makes sense to increase the transparency in the scope of modelling trees and tree stands, and L-systems are a good possibility to aid this transparency.

It shall be admitted that models like *LIGNUM* are not easy to implement using L-systems. Perttunen (oral communication) tried to implement the C-allocation algorithm using the concept of L-system. In principle this has worked but was rather inefficient. Prusinkiewicz et al. (1999) propose the language *L* to open the possibility to combine L-systems and *C*/*C++* code. This approach shows how complex the world of L-systems is and how many possibilities they open.

10 Summary

A large number of models are existing which try to image the growth of trees and tree stands. In classical forestry very aggregated models exist which forecast the development of economical parameters like the basal area or the resulting volume of a stand. These models work highly precise if the structure of a tree stand is simple, i.e. if the stand has a homogenous age and is monospecific. Other models try to describe the growth of tree stands with a higher level of structure by individual-based modelling. These models often do not comprehend functional information like light reception, carbon allocation or water flow. Models which consider such functions are existing in the scope of functional structural models (FSM) which contain functional as well as structural information.

The models mentioned above have different focusing regarding the level of aggregation and information about structure and functionality. They can be characterized by the model triangle (Kurth 1999). The modelling work itself comprehends four steps. Two of those steps regard analysis of reality or of virtual reality. A tool was designed and implemented which aids the work of the analysis part (*GroDisc*). *GroDisc* was developed to represent real 3D structure. It contains several analysis algorithms to discretize 3D architecture. The term architecture is defined after Godin (2000) and contains three components: geometry, decomposition and topology. Existing software like *AMAPmod* can provide excellent tools in the scope of analysing 3D architecture. However, the focus of *AMAPmod* lies more on tree than on tree stand architecture. Nevertheless, one demand was to integrate existing software into the tool *GroDisc*. It would have been a pity not to use the strength of tools like *AMAPmod* or *R*. Thus *AMAPmod* and the statistical software *R* can be directly linked to *GroDisc* by a data interface. *GroDisc* also contains two interfaces to other software for the enhancement of the analysis work. Mainly used in this work were the interfaces to *GROGRA* and *LIGNUM*.

A main focus of this work was to introduce rule based languages into the scope of individual-based tree stand modelling with simplified crown structures like those utilized by *SILVA* or *BWIN*. The architecture of such models contains simplified representations of the crown structure like spheres or irregular pyramids. It was shown here that such structures can be expressed by L-systems. The advantages of the use of L-systems also apply on the stand level. It could be shown that the condensed notation of this rule based language has a positive effect on the transparency. The reason for that is the high-level character of the rule language which reduces the technical contention with computer code.

Before the L-system could be used, the *GROGRA* specification had to be extended. Two sensitive functions were added. One achieves the calculation of the minimum distance of an elementary unit to a triangle. This function is used for calculating distance sensitivity between tree crowns. For that purpose the turtle command T is introduced which creates a triangle as a geometrical entity. The other function is an extension of the available light model within the L-system specification. It calculates a covering ratio in a way analogous to an already established function which simply returns the ratio between the numbers of shaded and non-shaded sectors of the hemisphere. The new function weights the shaded sectors by their inclination angle. The use of L-systems in this scope is exemplified on two examples: a reconstruction of a tree stand in Syke near Bremen and a model for a juvenile stand in the understorey of a gap. Within the reconstruction of the tree stand near Bremen it is tried to reproduce a model for distance sensitivity like Pretzsch (1992a,b, 2001) postulated for Norway spruce. This approach assumes a retreat of the crown extension if the crowns overlap. Such a behaviour could not be confirmed for the Scots pine tree stand. The crowns only stop the extension of the crown radii at a certain degree of overlapping. The reconstruction is attended by a correlation analysis for the parameters height, height of the crown base and diameter. It was expected that the parameters height, height of the crown base and diameter would be directly or indirectly dependent on the crown extension. This relation could be confirmed for the static situations in 1995 and 2001 but not for the increments. The growth of these parameters is then reconstructed by easily calculating the mean annual increment for each parameter from the initial situation in 1995. This regression shows coefficients of determination of around 0.18. For that reason the distributions of the residuals are also reconstructed. This reconstruction is implemented by an L-system. The L-system contains 13 rules and 62 lines of programming code. The irregular pyramids of the crown model are put together from triangles. The calculation of distance sensitivity is realized by the sensitive function which is created by the author.

The use of L-systems in the scope of individual based models is also illustrated by a model for a juvenile Scots pine stand in the understorey of a gap. The behaviour of the individuals results from aggregating the behaviour of *LIGNUM* trees. *LIGNUM* contains a model for the growth of Scots pines in a very detailed manner and a light model. The light model is aggregated by using a covering ratio. Two variants are existing: one is the ratio between the number of shaded and total number of sectors of the hemisphere and one is basically the same but weights the shaded vectors by the inclination angle. The growth of the tree is expressed as a regression between the annual increments of the parameters height, height of the crown base and crown extension and the light conditions expressed by the weighted covering ratio. The resulting L-system is very similar to that of the reconstruction of a Scots pine tree stand. However, it additionally contains a representation of the aged canopy surrounding the gap. The canopy is represented by "palisades" (cylindrical elements) which are arranged as a spiral similar to phyllotaxis. The calculation of the covering ratio is accomplished by two sensitive functions of *GROGRA*. The tree crown is again represented by triangles. The L-system contains 16 rules and 73 programming code lines. The quality of the aggregation was tested by simulating different trees in different distances from the centre of the gap. The simulation results were compared with the original data from the *LIGNUM* simulation. The comparison showed that the sensitive functions supply values different from the *LIGNUM* lightmodel calculation. The reason for that lies in the differences between the light models which are the base for each calculation.

Some conclusions: The tool *GroDisc* was an excellent and very flexible tool for analysing 3D architecture. The data structure represented the 3D architecture in a transparent way. The concept considerably increased the transparency of the algorithmic programming code. The integration of *AMAPmod* and *R* also considerably enhanced the possibilities for analysing tree and tree stand structure. Especially the use of the software *R* helped to avoid reimplementing well-known statistical algorithms.

The use of a rule based programming language in the form of the *GROGRA* L-systems specification extremely increased the transparency of the implementation work of modelling tree stands. It provides a transparency which cannot be provided by procedural or object oriented programming languages. It is easily possible to publish the L-system at least in a thesis like this. To remind of the complexity of the developed L-systems: The L-system for the model of juvenile Scots pines contains only 16 rules. It shall be admitted that it takes time to get acquainted with L-system programming or to understand the code completely. However, comprehending for instance C-code written by a foreign programmer can be much more laborious.

Finally the overall perception is that L-systems are a very useful tool even in individual based tree stand modelling. Simple crown structures as they are proposed by Pretzsch or Nagel are easy to implement. The use of L-systems significantly enhances the scientific transparency of the models and reconstructions.

Acknowledgement

First of all I want to thank Prof. Dr. Winfried Kurth for giving me the opportunity to work in his group and for the constant and supportive guidance. Absolute reliability is one of the major characteristics of my supervisor. To know Winfried was and still is an enrichment.

I'm very grateful to Prof. Dr. Dr. h.c. Sloboda and his institute (Institute for Forest Biometry and Applied Computer Science of the University of Göttingen). He and other members of the institute — especially Ilona — provided not only the necessary "infrastructure" but also a familiar atmosphere.

The author and his work were located at the Plant Modelling Group at the institute mentioned above, hence many thanks once more to Winfried and also to Michael, Ulrike, Gustavo and Dirk for the pleasant company. The working group is always a good and helpful discussion forum.

Part of this work evolved from a cooperation with the Metsäntutkimoslaitos Vantaan Tutkimuskeskus (METLA), Finland. I thank the people there for the assistance and friendship during my stay in 2000, namely Risto, Jari and Minna. This stay was financed by a grant from the DAAD. The application was supported by the Consulate General of Finland in Hannover. Therefore many thanks to Mr. Batsch and Mrs. Gimbel. The stay was additionally financed by my aunt Elisabeth. For this generosity I would especially like to thank her.

Many grateful thanks to Dr. Marie Stella Duchiron alias Schtella for allowing me to use the Scots pine data which are part of several simulation runs.

The project of this work has been financed by the DFG (German Research Foundation) (Az. KU-847/3-1,2).

And the most important at the end. I thank Katrin for to the English corrections and the encouragement. Without her the amount of reasonable text would be less and the state of the author might be bad today.

References

H. Abelson and A.A. diSessa. *Turtle geometry.* MIT Press, Cambridge, 1982.

M. Albert. *Analyse der eingriffsbedingten Strukturveränderung und Durchforstungsmodellierung in Mischbeständen*, volume 6. Hainholzverlag, Hainholz Forstwissenschaften, Göttingen, 1999.

G.A. Anzola Jürgenson. *Linking structural and process-oriented models of plant growth.* PhD thesis, University of Göttingen, 2002.

Arbeitskreis Standortskartierung. *Forstliche Wuchsgebiete und Wuchsbezirke in der Bundesrepublik Deutschland.* Landwirtschaftsverlag GmbH, Münster–Hiltrup, 1985.

E. Assmann. *Waldertragskunde.* BLV Verlagsgesellschaft, München–Bonn–Wien, 1961.

G.M. Berntson and P. Stoll. Correcting for finite spatial scales of self-similarity when calculating the fractal dimensions of real-world structures. In *Proceedings of the Royal Society London*, volume 264 of *B*, pages 1531–1537, 1997.

H. Bossel. TREEDYN3 forest simulation model. *Ecological modelling*, 90:187–227, 1996.

B. Breckling. An object oriented modelling strategy to depict activity pattern of organisms in heterogeneous environments. *Environmental Modellig and Assessment*, 2:95–104, 1997.

U. Breymann. *Designing Components with the C++ STL. A new approach to programming.* Addison-Wesley, 1998.

Y. Chiba. Plant form analysis based on the pipe model theory. I. A statical model within the crown. *Ecological Research*, pages 207–220, 1990.

Y. Chiba. Mathematical model of stem formation and tree architectural development. In H-C. Spatz and Th. Speck, editors, *Plant Biomechanics 2000. Proceedings of the 3rd Plant Biomechanics Conference Freiburg Badenweiler 27. 8.-2. 9. 2000*, pages 606–612, Stuttgart, 2000. Thieme.

P.J. Clark and F.C. Evans. Distance to nearest neighbour as a measure of spatial relationships in populations. *Ecology*, 35:445–453, 1951.

H. Cochard. Vulnerability of several conifers to air embolism. *Tree Physiology*, 11:73–83, 1992.

J. Dauzat, B. Rapidel, and A. Berger. Simulation of leaf transpiration and sap flow in virtual plants: model description and application to a coffee plantation in Costa Rica. *Agricultural and Forest Meteorology*, 109:143–160, 2001.

A. Degenhardt. Simulative Erzeugung von Waldstrukturen auf der Grundlage von Inventurdaten. In *Beiträge zur 11. Tagung der Sektion Forstliche Biometrie und Informatik des Deutschen Verbandes Forstlicher Forschungsanstalten, Ljubljana*, pages 58–68, 1999.

A. Degenhardt and A. Pommerening. Erzeugung von Bestandesstrukturen auf der Grundlage von Probekreisdaten. In *Beiträge zur 12. Tagung der Sektion Forstliche Biometrie und Informatik des Deutschen Verbandes Forstlicher Forschungsanstalten, Ljubljana*, pages 297–311, 2000.

J.A. den Dulk. *The interpretation of remote sensing, a feasability study.* PhD thesis, University of Wageningen, Netherlands, 1989.

J. Douglas and B.F. Jones. On predictor-corrector methods for nonlinear parabolic differential equations. *Journal of the Society for Indusrial and Applied Mathemathics*, 11: 195–204, 1963.

M.-S. Duchiron. *Structures Forestiéres. Dynamique—Modélisation—Consequénces pour l'écosystème.* Édition par l'auteur, Paris, 2003.

H. Dzierzon and W. Kurth. LIGNUM: A Finnish tree growth model and its interface to the French AMAPmod database. In F. Hölker, editor, *Scales, hierarchies and emergent properties in ecological models*, volume 6 of *Theorie in der Ökologie*, pages 95–104, Frankfurt a. M, 2002. Peter Lang Verlag.

H. Dzierzon, J. Perttunen, W. Kurth, R. Sievänen, and B. Sloboda. Enhanced possibilities for analyzing tree structure as provided by an interface between different modelling systems. *Silva Fennica*, 37(1):31–44, 2003.

G. Edgar. *Measures, Topology and Fractal Geometry.* Springer-Verlag, New York-Heidelberg-Berlin, 1990.

A.R. Ek and R.A. Monserud. Trials with program FOREST. Growth and reproduction simulation for mixed species even- or uneven-aged forest stands. In J. Fries, editor, *Proceeding of meetings in 1973 Royal College of Forestry. Growth models for tree and stand simulation.*, volume 30, pages 56–73, 1974.

Ch. Eschenbach. The effect of light acclimation of single leaves on whole tree growth and competition – an application of the tree growth model ALMIS. *Annals of Forest Science*, 57:599–609, 2000.

P. Ferraro and C. Godin. A distance measure between plant architectures. *Annals of Forest Science*, 57:445–461, 2000.

A. Franc and N. Picard. Scaling and aggregation of variables. I - Link with measurement. franc, 2001.

Th. Früh. Entwicklung eines Simulationsmodells zur Untersuchung des Wasserflusses in der verzweigten Baumarchitektur. In *Berichte des Forschungszentrums Waldökosysteme*, volume B 131, page 192, 1995.

Th. Früh and W. Kurth. The hydraulic system of trees: Theoretical framework and numerical simulation. *Journal of Theoretical Biology*, 201:251–270, 1999.

K. Füldner. *Strukturbeschreibung von Buchen-Edellaubholz-Mischwäldern*, volume 146. Cuvillier, Göttingen, 1995.

K.v. Gadow, G.Y. Hui, and M. Albert. The neighbouhoodpattern — a new parameter for describing forest structures. *Forstwissenschaftliches Centralblatt*, 115(1):1–10, 1998.

D. Gaffrey, B. Sloboda, and N. Matsumura. Representation of tree stem taper curves and their dynamic, using a linear model and the centroaffine transformation. *Journal of Forest Research*, (3):67–74, 1998.

C. Godin. Representing and encoding plant architecture. A review. *Annals of Forest Science*, 57:413–438, 2000.

C. Godin and Y. Caraglio. A multiscale model of plant topological structures. *Journal of Theoretical Biology*, 191:1–46, 1998.

C. Godin, E. Costes, and H. Sinoquet. A method for describing plant architecture which integrates topology and geometry. *Annals of Botany*, 84:343–357, 1999.

C. Godin, Y. Guédon, and E. Costes. Exploration of a plant architecture database with the AMAPmod software illustrated on an apple tree hybrid family. *Agronomie*, 19: 163–184, 1998.

K. Goebel. *Organographie der Pflanzen*. G. Fischer Verlag, 2nd edition, 1928.

R.P. Grimaldi. *Discrete and Combinatorial Mathematics.* Addison-Wesley, Reading, second edition, 1989.

R. Grote. Foliage and branch biomass estimation of coniferous and deciduous tree species. *Silva Fennica*, 36(4):779–788, 2002a.

R. Grote. Von der Baumdimension zur Biomasse und wieder zurück — Ein neuer Ansatz zur dynamischen Modellierung von Baum- und Bestandesbiomassen. In H-P. Dietrich et al., editor, *Inventur von Biomasse- und Nährstoffvorräten. Forstliche Forschungsberichte München*, pages 129–138, 2002b.

R. Grote and H. Pretzsch. A model for individual tree development based on physiological processes. *Plant Biology*, 4:167–180, 2002.

M. Guericke. *Untersuchungen zur Wachstumsdynamik von Mischbeständen aus Buche und Europ. Lärche (Larix decidua, Mill.) als Grundlage für ein abstandsabhängiges Einzelbaumwachstumsmodell.* PhD thesis, Universität Göttingen, 2001.

M. Hauhs, A. Kastner-Maresch, and K. Rost-Siebert. A model relating forest growth to ecosystem-scale budgets of energy and nutrients. *Ecological Modelling*, 83:229–243, 1995.

M. Hauhs, F.-J. Knauft, and H. Lange. Algorithmic and interactive approaches to stand growth modelling. In A. Amaro, D. Reed, and P. Soares, editors, *Modelling Forest Systems*, Wallingford, UK, 2003. CABI Publishing.

B. Hopkins. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18:213–227, 1954.

V.-P. Ikonen, S. Kellomäki, and H. Peltola. Linking tree stem properties of Scots pine to the properties of sawn timber through simulated sawing. In *Workshop Proceedings for the Fourth Workshop on Connection between Forest Resources and Wood Quality: Modelling Approaches and Simulation Software, Harrison, British Columbia, Canada, September 8-15, 2002*, in press.

M. Kahn and H. Pretzsch. Das Wuchsmodell SILVA - Parametrisierung der Version 2.1 für Rein- und Mischbestände aus Fichte und Buche. *Allgemeine Forst- und Jagdzeitung*, 168(6/7):115–123, 1997.

S. Kellomäki, V.-P. Ikonen, H. Peltola, and T. Kolström. Modelling the structural growth of Scots pine with implications for wood quality. *Ecological Modelling*, 122:117–134, 1999a.

S. Kellomäki and O. Kurttio. A model for the structural development of a Scots pine crown based on modular growth. *Forest Ecology and Management*, 43:103–123, 1991.

S. Kellomäki, H. Peltola, and V.-P. Ikonen. Modelling the structural growth and timber quality of Scots pine related to silvicultural management. In G. Nepveu, editor, *Workhop Proceedings for the Third Workhop on "Connection between silviculture and wood quality through modelling approaches and simulation softwares", La Londe-les-Maures, France, September 5-12, 1999*, pages 565–570, 1999b.

S. Kellomäki and H. Strandman. A model for structural growth of young Scots pine crowns based on light interception by shoots. *Ecological Modelling*, 80:237–250, 1995.

H. Kramer. *Grundlagen zur forstlichen Ertragskunde.* Published by editor, third edition, 1984.

W. Kurth. *Growth grammar interpreter GROGRA 2.4. - A software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling.*, volume B 38 of *Berichte des Forschungszentrums Waldökosysteme.* Göttingen, 1994.

W. Kurth. Some new formalisms for modelling the interactions between plant architecture, competition and carbon allocation. *Bayreuther Forum Ökologie*, 52:53–98, 1998.

W. Kurth. *Die Simulation der Baumarchitektur mit Wachstumsgrammatiken.* Wissenschaftlicher Verlag, Berlin, 1999.

W. Kurth. Spatial structure, sensitivity and communication in rule-based models. In F. Hölker and B. Breckling, editors, *Scales, hierarchies and emergent properties in ecological models*, volume 6 of *Theorie in der Ökologie*, pages 29–46, Frankfurt a. M, 2002. Peter Lang Verlag.

W. Kurth and G.A. Anzola Jürgenson. Triebwachstum und Verzweigung junger Fichten in Abhängigkeit von den beiden Einflußgrößen Beschattung und Wuchsdichte: Datenaufbereitung und -analyse mit GROGRA. In D.R. Pelz, editor, *Deutscher Verband Forstlicher Forschungsanstalten, Sektion Forstliche Biometrie und Informatik, 10. Tagung, Freiburg i.Br., 24.–28.9.1995, Ljubilana, Biotechnische Fakultät*, pages 89–108, 1997.

W. Kurth and M. Bredemeyer. Modellkonzeption und -vernetzung am Forschungszentrum Waldökosysteme (Göttingen). In K. Mathes, B. Breckling, and K. Ekschmitt, editors, *Systemtheorie in der Ökologie. Beiträge zum Gfö-Workshop in Rauischholzhausen am 15.3.1996*, pages 67–72, Landsberg, 1996. Ecomed-Verlag.

W. Kurth and B. Sloboda. Tree and stand architecture and growth described by formal grammars — I. Non-sensitive trees. *Journal of Forest Science*, 45:16–30, 1999a.

W. Kurth and B. Sloboda. Tree and stand architecture and growth described by formal grammars — II. Sensitive trees and competition. *Journal of Forest Science*, 45:53–63, 1999b.

D. Lanwert, J. Dauzat, and T. Früh. Water use of forest trees: a possibility of combining structures and functional models. *Bayreuther Forum Ökologie*, 52:117–128, 1998.

A. Lindenmayer. Mathematical models for cellular interaction in development, parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

R. List, M. Küppers, and F. Schröder. Madeira – ein Programm zur Simulation des Wachstums (Biomasse und Architektur) von Holzgewächsen. In *Verhandlungen der Gesellschaft für Ökologie, 23*, pages 39–48, Innsbruck, 1994.

E. Lo, Z.M. Wang, M. Lechowicz, C. Messier, E. Nikinmaa, R. Sievänen, and J. Perttunen. Adaptation of the LIGNUM model for simulations of growth and light response in jack pine. *Forest Ecology and Management*, 150(3):279–291, 2001.

J. Lück, H.B. Lück, and M. Bakkali. A comprehensive model for acrotonic, mesotonic and basitonic branches in plants. *Acta Biotheoretica*, 38:257–288, 1990.

B.B. Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman, New York, 1982.

T.A. McMahon and E. Kronauer. Tree structure: deducing the principle of mechanical design. *Journal of Theoretical Biology*, 59:443–466, 1976.

U. Middelhoff and B. Breckling. Nutrient uptake and growth of an Alnus glutinosa stand: an individual based model about the interaction of plant and soil. *Bayreuther Forum Ökologie*, 52:189–203, 1998.

R.A. Monserud. *Methodology for simulating Wisconsin northern hardwood stand dynamics.* PhD thesis, University of Wisconsin-Madison, 1997.

P. Moon and D.E. Spencer. Illumination from a non-uniform sky. *Illumination Engineering*, 37:707–726, 1942.

J. Nagel. *Konzeptionelle Überlegungen zum schrittweisen Aufbau eines waldwachstumkundlichen Simulationssystems für Nordwestdeutschland.* J.D. Sauerländers's Verlag, Frankfurt am Main, 1999.

J. Nagel, M. Albert, and M. Schmidt. *BWINPro Programm zur Bestandesanalyse und Prognose. Handbuch zur Version 5.1.* Niedersächsische Forstliche Versuchsanstalt, Göttingen, 2000.

E. Nikinmaa. Analysis of the growth of Scots pine; matching structure with function. *Acta Forestalia Fennica*, 235, 1992.

P. Oker-Blom and H. Smolander. The ratio of shoot silhouette area to total needle area in Scots pine. *Forest Science*, 34(4):894–906, 1988.

R.V. O'Neill, D.L. de Angelis, J.B. Weide, and T.F.H. Allen. A hierachical concept of ecosystems. *Princeton University Press*, 1986.

A.L. Oppelt, W. Kurth, H. Dzierzon, G. Jentschke, and D.L. Godbold. Structure and fractal dimensions of root systems of four co-occuring fruit tree species from Botswana. *Annals of Forest Science*, 57(5/6):463–475, 2000.

A.L. Oppelt, W. Kurth, and D.L. Godbold. Topology, scaling relations and Leonardo's rule in root systems from African tree species. *Tree Physiology*, 21:117–128, 2001.

J.M. Ottorini. Growth and development of individual douglas fir in stands for applications to simulation in silviculture. *Annales des Sciences Forestières*, 48:651–666, 1991.

H-O. Peitgen, H. Jürgens, and D. Saupe. *Bausteine des Chaos – Fraktale.* Springer / Klett-Cotta, Berlin–Stuttgard, 1992.

J. Perttunen, E. Nikinmaa, M.J. Lechowicz, R. Sievänen, and C. Messier. Application of the functional-structural tree model LIGNUM to sugar maple saplings (*Acer saccharum* Marsh) growing in forest gaps. *Annals of Botany*, 88(3):471–481, 2001.

J. Perttunen, R. Sievänen, and E. Nikinmaa. LIGNUM. A model combining the structure and the functioning of trees. *Ecological Modelling*, 108:189–198, 1998.

J. Perttunen, R. Sievänen, E. Nikinmaa, H. Salminen, H. Saarenmaa, and J. Väkevä. LIGNUM: a tree model based on simple structural units. *Annals of Botany*, 77:87–98, 1996.

C. Petermann. Baumstrukturen und Fraktale. Master's thesis, Westfälische Wilhelms-Univ. Münster, 1999.

J. Pfreundt. Modellierung der räumlichen Verteilung von Strahlung, Photosynthesekapazität und Produktion in einem Fichtenbestand und ihrer Beziehung zur Bestandesstruktur. In *Berichte des Forschungszentrums Waldökosysteme*, volume A 39, Göttingen, 1988.

J. Pfreundt and B. Sloboda. The relation of local stand structure to photosynthetic capacity in a spruce stand: a model calculation. *Lesnictví-Forestry*, 42:149–160, 1996.

M. Pfund. A topological data structure 3D GIS architecture. *GIM International, February 2002*, 16(2), 2002.

H. Pretzsch. Analyse der Bestandesstruktur als Grundlage für die Entwicklung eines Wachstumssimulators für Mischbestände. In *Tagungsberichte der Arbeitsgruppe Biometrie in der Ökologie*, volume 1, pages 51–71, 1990a.

H. Pretzsch. Struktur zur Steuerung von Wuchsmodellen für Rein- und Mischbestände. In *Deutscher Verband Forstlicher Forschungsanstalten, Sektion Ertagskunde. Jahrestagung Verden/Aller 28.-30.5.1990*, pages 80–100, 1990b.

H. Pretzsch. Kronenformen und ihre Veränderung unter Konkurrenz. In *Deutscher Verband Forstlicher Forschungsanstalten, Sektion Ertagskunde. Jahrestagung Grillenburg 1992*, pages 40–61, 1992a.

H. Pretzsch. Modellierung der Kronenkonkurrenz von Fichte und Buche in Rein- und Mischbeständen. *Allgemeine Forst- und Jagdzeitung*, 163(11/12):203–213, 1992b.

H. Pretzsch. Wo steht die Waldwachstumsforschung heute? Denkmuster - Methoden - Feedback. *Allgemeine Forst- und Jagdzeitung*, 168:98–102, 1997.

H. Pretzsch. *Modellierung des Waldwachstums.* Parey Buchverlag, Berlin, 2001.

H. Pretzsch and J. Ďurský. Evaluierung von Waldwachstumssimulatoren auf Baum- und Bestandesebene. *Allgemeine Forst- und Jagdzeitung*, 172(8/9):146–150, 2001.

M. Prodan. *Holzmesslehre.* J.D. Sauerländer's, Frankfurt, 1965.

P. Prusinkiewicz, R. Karwowski, R. Mech, and J. Hanan. L-studio/cpfg: a software system for modeling plants. *Lecture Notes in Computer Science*, 1779:457–464, 2000.

P. Prusinkiewicz, R. Karwowski, J. Perttunen, and Sievänen R. Specification of L - a plant modelling language based on Lindenmayer systems. Version 0.5. *University of Calgary, Department of Computer Science, Research note*, 1999.

P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants.* Springer, New York, 1990.

B. Rapidel. *Etude expérimentale et simulation des transferts hydriques dans les plantes individuelles. Application au caféier (Coffea arabica L.).* PhD thesis, Université Montpellier II, Montpellier, 1995.

A. Requita. Representation of rigid solids: Theory, methods and systems. *Computing Surveys*, 12(4):87–93, 1980.

J. Ross. *The radiation regime and architecture of plant stands.* W. Junk, The Hague-Boston-London, 1981.

H. Saarenmaa. Object-oriented design of topological tree and forest models. In A. Teller, P. Mathy, and J.N.R. Jeffers, editors, *Responses of forest ecosystems to environmental changes*, pages 957–958, London, 1992.

H. Salminen, H. Saarenmaa, J. Perttunen, R. Sievänen, J. Väkevä, and E. Nikinmaa. Modelling trees using an object oriented scheme. *Mathematical and Computer Modelling*, 20(8):49–64, 1994.

R. Schober. *Ertragstafeln wichtiger Baumarten.* J.D. Sauerländer's Verlag, Frankfurt a.M., 1975.

M. Schulte. Final report of the project: Numerische Simulation des hydraulischen Systems Baum-Boden bei der Traubeneiche (Quercus petraea, (Matt.) Liebl.) granted by the DFG: Sl 11/8-3. Technical report, DFG, 2002.

K. Shinozaki, K. Yoda, K. Hozumi, and T. Kira. A quantitative analysis of plant form — the pipe model theory. I. Basic analysis. *Japanese Journal of Ecology*, 14(3):97–105, 1964.

R. Sievänen. Class Firmament in LIGNUM. Technical description in the computer code for the method "diffuseForestRegionRadiationSum" in Firmament.cc.

R. Sievänen, E. Nikinmaa, and J. Perttunen. Evaluation of importance of sapwood senescence on tree growth using the model LIGNUM. *Silva Fennica*, 31(3):329–340, 1997.

B. Sloboda. Bestandesindividuelles biometrisches Schaftformmodell zur Darstellung und zum Vergleich von Formigkeit und Sortimentsausbeute sowie Inventur. In *Inventoring and Monitoring Endangered Forests. IUFRO Conference Zürich 1985.*, pages 345–353, Birmensdorf, 1985. Eidg. Anstalt für das forstliche Versuchswesen.

B. Sloboda, D. Gaffrey, and N. Matsumura. Erfassung individueller Baumschaftformen und ihrer Dynamik durch Spline-Funktionen und Verallgemeinerung durch lineare Schaftformmodelle. *Allgemeine Forst und Jagdzeitung*, 169(2):29–39, 1998.

B. Sloboda and J. Pfreundt. Tree and stand growth. A system analytical spatial model with consequences for test planning for thinning and single tree development. In *Artificial intelligence and growth models for forest management decisions, Publication No. FWS-1-89*, Blacksburg, Virginia, 1989.

M. Sonntag. *Klimavernderungen und Waldwacshtum: TREEDYN3-Simulationen mit einer Analyse modellstruktureller Unsicherheiten.* Verlag Mainz, Wissenschaftsverlag, Aachen, 1998.

StatSoft. *STATISTICA for Windows (Computer program manual).* Tulsa, 1999.

P. Stenberg. Metsikön rakenne, saäteilyolot ja tuotos. (In Finnish). *University of Helsinki, Department of Forest Ecology, Publications No. 15*, 1996.

H. Sterba. Valitätsprüfung des Einzelbaumsimulators für Fichten-Kiefern-Mischbestände. In *Deutscher Verband Forstlicher Forschungsanstalten, Sektion Ertagskunde. Jahrestagung Verden/Aller 28.-30.5.1990*, pages 101–117, 1990.

P. Stoll. *Modular growth and foraging strategies in rhizome systems of Solidago altissima L. and branches of Pinus sylvestris L.* PhD thesis, University of Zurich, 1995.

A. Streilein. Digitale Photogrammetrie und CAAD. *Mitteilung Nr. 68, Institut für Geodäsie und Photogrammetrie, ETH Zürich*, 1999.

B. Stroustrup. *Die C++ Programmiersprache.* Addison-Wesley, Bonn, third edition, 1998.

V. Stüber and K. Staupendahl. Modellierung der Kronenschirmflächen am Beispiel eines Douglasienbestandes. In K. v. Gadow and Th. Beisch, editors, *Beiträge zur Waldinventur, Festschrift A. Akça*, pages 117–130, Göttingen, 1996.

A. Takenaka. A simulation model of tree architecture development based on growth response to local light environment. *Journal of Plant Research*, 107:321–330, 1994.

M.T. Tyree and J.S. Sperry. Do woody plants operate near the point of catastrophic xylem dysfunction caused by dynamic water stress? Answers from a model. *Plant Physiology*, 88:574–580, 1988.

R.F. Voss. Fractals in nature: from characterization to simulation. In H.-O. Peitgen and D. Saupe, editors, *The science of fractal images*, pages 21–70, New York-Heidelberg-Berlin, 1988. Springer-Verlag.

G. West, J.H. Brown, and B.J. Enquist. A general model for the origin of allometric scaling laws in biology. *Science*, 276:122–126, 1997.

G. West, J.H. Brown, and B.J. Enquist. A general model for the structure and allometry of plant vascular systems. *Nature*, 400:664–667, 1999.

B. Zeide and P. Pfeifer. A method for estimation of fractal dimension of tree crowns. *Forest Science*, 37(5):1253–1265, 1991.

Appendix 1: Algorithms in *GroDisc*

ForEach

```
// Declaration

template <class Op>
void ForEach(gdLayerCompartment& comp, Op op, int level = 0);

// Definition

template <class Op>
void ForEach(gdLayerCompartment& comp, Op op, int level){
  if(gdLayer* layer = dynamic_cast<gdLayer*>(&comp)){
    if((level&GD_LAYER)==0)
      op(layer);
    if(gdEUnit* unit = dynamic_cast<gdEUnit*>(*(layer->begin()))){
      if((level&GD_EUNIT) !=0)
        return ;
    }
    list<gdLayerCompartment*>::iterator next = layer->begin();
    list<gdLayerCompartment*>::iterator last = layer->end();
    while(next!=last)
      ForEach(**next++,op,level);
  }
  else if(gdMetaUnit* unit = dynamic_cast<gdMetaUnit*>(&comp)){
    if((level&GD_MUNIT)==0)
      op(unit);
    list<gdLayerCompartment*>::iterator next = unit->begin();
    list<gdLayerCompartment*>::iterator last = unit->end();
    while(next!=last)
      ForEach(**next++,op,level);
  }
  else if(gdEUnit* unit = dynamic_cast<gdEUnit*>(&comp))
    op(unit);
}
```

Accumulate

```
// Declaration

template <class T, class Op>
void Accumulate(gdLayerCompartment& comp, T& init , Op op, int level = 0);

// Definition

template <class T, class Op>
void Accumulate(gdLayerCompartment& comp, T& init , Op op, int level){
  if(gdLayer* layer = dynamic_cast<gdLayer*>(&comp)){
    if((level&GD_LAYER)==0)
    op(init ,layer);

    list<gdLayerCompartment*>::iterator next = layer->begin();
    list<gdLayerCompartment*>::iterator last = layer->end();
    while(next!=last)
      Accumulate(**next++,init ,op,level);
  }
  else if(gdMetaUnit* unit = dynamic_cast<gdMetaUnit*>(&comp)){
    if((level&GD_MUNIT)==0)
    op(init ,unit);
    list<gdLayerCompartment*>::iterator next = unit->begin();
    list<gdLayerCompartment*>::iterator last = unit->end();
    while(next!=last)
      Accumulate(**next++,init ,op,level);
  }
  else if(gdEUnit* unit = dynamic_cast<gdEUnit*>(&comp)){
    if((level&GD_EUNIT) ==0)
    op(init ,unit);
  }
}
```

ForEachPair

```
// Declarations

template <class Op>
void ForEachPair(gdLayerCompartment& ref, gdLayerCompartment& comp,
                 Op op, int level1 = 0, int level2 = 0);


template <class Op>
void ForEachPair(gdLayerCompartment& comp1, Op op,
                 int level1 = 0, int level2 = 0);

template <class Op>
class ForEachPairOp{
public:
  ForEachPairOp(const Op op, int level2);
  void operator () (gdLayerCompartment& ref, gdLayerCompartment* comp);
protected:
  Op  _op;
  int _level2;
};
```

```
// Implementations

template <class Op>
ForEachPairOp<Op>::ForEachPairOp(const Op op, int level2)
        :_op(op),_level2(level2)
{
}

template <class Op>
void ForEachPairOp<Op>::operator ()
        (gdLayerCompartment& comp, gdLayerCompartment* ref){
  Accumulate(comp,ref,_op,_level2);
}

template <class Op>
void ForEachPair(gdLayerCompartment& ref, gdLayerCompartment& comp,
                 Op op, int level1, int level2){
  ForEachPairOp<Op> op1(op,level2);
  Accumulate(ref,comp,op1,level1);
}

template <class Op>
void ForEachPair(gdLayerCompartment& comp1,
                 Op op, int level1 = 1000, int level2 = 1000){
  ForEachPair(comp1,comp1,op,level1,level2);
}
```

AccumulatePairs

```
// Declarations

template <class Op, class T>
void AccumulatePairs(gdLayerCompartment& comp1, gdLayerCompartment& comp2,
                     T& init, Op op, int level1 = 0, int level2 = 0);

template <class Op, class T>
void AccumulatePairs(gdLayerCompartment& comp1, T& init,
                     Op op, int level1 = 0, int level2 = 0);



template <class Op, class T>
class AccumulatePairsOp1{
public:
  AccumulatePairsOp1(const Op op, T& init, int level2);
  gdLayerCompartment& operator()
             (gdLayerCompartment& comp1, gdLayerCompartment* comp2);
protected:
  Op  _op;
  T*  _init;
  int _level2;
};

template <class Op, class T>
class AccumulatePairsOp2{
public:
  AccumulatePairsOp2(const Op op, T& init);
  gdLayerCompartment*  operator()
             (gdLayerCompartment* comp1, gdLayerCompartment* comp2);
protected:
  Op  _op;
  T*  _init;
};
```

```
// Implementations

template <class Op, class T>
AccumulatePairsOp1<Op,T>::AccumulatePairsOp1(const Op op, T& init, int level2)
        :_op(op),_init(&init),_level2(level2)
{
}

template <class Op, class T>
gdLayerCompartment& AccumulatePairsOp1<Op,T>::operator ()
          (gdLayerCompartment& comp1, gdLayerCompartment* comp2){
  AccumulatePairsOp2<Op,T> op2(_op,*_init);
  Accumulate(comp1,comp2,op2,_level2);
  return comp1;
}

template <class Op, class T>
AccumulatePairsOp2<Op,T>::AccumulatePairsOp2(const Op op, T& init)
        :_op(op),_init(&init)
{
}

template <class Op, class T>
gdLayerCompartment* AccumulatePairsOp2<Op,T>::operator ()
          (gdLayerCompartment* comp1, gdLayerCompartment* comp2){
  _op(comp1,comp2,*_init);
  return comp1;
}

template <class Op, class T>
void AccumulatePairs(gdLayerCompartment& comp1, gdLayerCompartment& comp2,
                     T& init, Op op, int level1, int level2){
  AccumulatePairsOp1<Op,T> op1(op,init,level2);
  Accumulate(comp1,comp2,op1,level1);
}

template <class Op, class T>
void AccumulatePairs(gdLayerCompartment& comp1, T& init,
                     Op op, int level1 = 1000, int level2 = 1000){
  AccumulatePairs(comp1,comp1,init,op,level1,level2);
}
```

Appendix 2: Functors of *GroDisc*

MTG interface functor of *GroDisc*

```
// Declaration of the functor which provides the interface
// to AMAPmod. The functor is applied by Accumulate within the
// scope of a LIGNUM tree.

template <class TS, class BUD=DefaultBud<TS> >
struct CreateMTG{
  static int tree_nr;
  static int max_ord;
  QString& operator()(QString& res, TreeCompartment<TS,BUD>* tc)const;
};


// Implementation of the functor which provides the interface
// to AMAPmod.



// Values for parameterization of the functor. tree_nr is the number
// of the first tree. max_ord is the maximal order of the tree and is
// required at the beginnig of the translation process. It affects the
// number of TABS within the MTG-file.

template <class TS, class BUD>
int CreateMTG<TS,BUD>::tree_nr = 1;
template <class TS, class BUD>
int CreateMTG<TS,BUD>::max_ord = 0;

// The implementation of the bracket operator which is applied by the
// algorithm Accumulate.

template <class TS, class BUD>
QString& CreateMTG<TS,BUD>::operator()
          (QString& res,TreeCompartment<TS,BUD>* tc)const{
  // The current BranchingPoint is used to calculate the correct
  // identifiers of the TreeSegments.
  if(BranchingPoint<TS,BUD> *bp =
      dynamic_cast<BranchingPoint<TS,BUD>*>(tc)){
    int axis_counter = 1;
    list<Axis<TS,BUD>*> a_list = GetAxisList(*bp);
    list<Axis<TS,BUD>*>::iterator nextAxis = a_list.begin();
    list<Axis<TS,BUD>*>::iterator lastAxis = a_list.end();
    while(nextAxis!=lastAxis){ // while 1
      int id = axis_counter;
      bool start=true;
      list<TreeCompartment<TS,BUD>*> lc_list =
         GetTreeCompartmentList(**nextAxis);
      list<TreeCompartment<TS,BUD>*>::iterator nextTC =
         lc_list.begin();
      list<TreeCompartment<TS,BUD>*>::iterator lastTC =
         lc_list.end();
      while(nextTC!=lastTC){ // while 2
        if(TS* segment = dynamic_cast<TS*>(*nextTC)){
          segment->id = id++;
          if(start){
            segment->id*=-1;
            start = false;
          }
        }
        nextTC++;
      } // close while 2
      axis_counter++;
      nextAxis++;
    } // close while 1
  }

  // The current TreeSegment is written to the MTG-file.
  else if(TS* segment = dynamic_cast<TS*>(tc)){
    int omeg = (int) GetValue(*segment,omega);
    int c1,c2;
    QString buf;
    for(c1=0;c1<omeg;c1++)
      res+="\t";
    if(segment->id<0){
      buf.sprintf("+U%d/I1",segment->id*-1);
      res+=buf;
    }
    else{
      buf.sprintf("^<U%d/I1",segment->id);
      res+=buf;
    }
    for(c2=0;c2<max_ord-c1+1;c2++)
      res.append("\t");
    Point pend = GetEndPoint(*segment);
    buf.sprintf(”%.6f\t%.6f\t%.6f\t%.6f\t%.6f\n”,pend.getX(),pend.getY(),
                   pend.getZ()/1000.,GetValue(*segment,Lignum::R),
                   GetValue(*segment,Lignum::Wf));
    res+=buf;
  }
  return res;
}
```

Distance to triangle

```
// Declaration of the functor which provides the calculation of the
// distance between a gdTreeUnit to the surface of a gdTriangle. The
// functor is applied by the algorithm ForEachPair.

struct DistUnitTri{
  void operator ()(gdLayerCompartment* c1, gdLayerCompartment* c2);
}

// Implementation of the bracket operator which appiel the
// calculation. The distance is calculated by the function
// ``Distance'' which is identic to function 14 of \grogra.

void DistUnitTri::operator ()(gdLayerCompartment* c1, gdLayerCompartment* c2){
  if(gdTreeUnit* unit = dynamic_cast<gdTreeUnit*>(c1)){
    if((int) (*unit)[INT][”COLOR”]!=10) return;
    if(gdTriangle *tri = dynamic_cast<gdTriangle*>(c2)){
      (*unit)[DOUBLE][”HVAR1”] = Distance(*unit,*tri);
    }
  }
}
```

LIGNUM interface functor of *GroDisc*

```
// Declaration of the structure which contains the functors which
// handles the interface between LIGNUM and GroDisc. Both directions
// are implemented here. The overloaded operator 1 controls the
// GroDisc LIGNUM data interface whereas the oprator 2 and 3
// translates the LIGNUM into a GroDisc tree. The translation process
// is started by the overloaded function ConvertTree.

template <class TS, class BUD>
struct ConvertUnits{
  static LGMAD       lig_val_type;
  static string      gd_val_type;
  static gdTree*     tree;
  // Operator 1
  TreeCompartment<TS,BUD>* operator()
    (ConvertUnitsData& data, TreeCompartment<TS,BUD>* tc)const;
  // Operator 2
  void operator ()
    (Tree<TS,BUD>& tree, gdLayerCompartment* lc) const;
  // Operator 3
  gdTreeUnit&  operator()
    (gdTreeUnit& unit, TreeCompartment<TS,BUD>* tc)        const;
};

// The parameter lig_val_type and gd_val_type control the type of
// information which is translated. The resulting tree is stored into
// the variable tree.

template <class TS, class BUD>
LGMAD       ConvertUnits<TS,BUD>::lig_val_type   = Wf;
template <class TS, class BUD>
string      ConvertUnits<TS,BUD>::gd_val_type    = "NAD";
template <class TS, class BUD>
gdTree*     ConvertUnits<TS,BUD>::tree           = new gdTree();

template <class TS, class BUD>
TreeCompartment<TS,BUD>* Lignum::ConvertUnits<TS,BUD>::operator()
  (ConvertUnitsData& data, TreeCompartment<TS,BUD>* tc)const{
  if(Axis<TS,BUD>* axis = dynamic_cast<Axis<TS,BUD>*>(tc)){
    data.order++;
  }
  else if(TS* seg = dynamic_cast<TS*>(tc)){
    gdTreeUnit *new_unit = new gdTreeUnit();
    // Setting mother
    new_unit->parent(data.mother);
    data.mother = new_unit;
    // Color is set after conversion
    // Setting Nad
    (*new_unit)[DOUBLE][gd_val_type.c_str()] = GetValue(*seg,lig_val_type);
    // Setting adur and edur
    (*new_unit)[DOUBLE]["ADUR"] = GetValue(*seg,R);
    (*new_unit)[DOUBLE]["EDUR"] = GetValue(*seg,R);
    // setting length
    (*new_unit)[DOUBLE]["LENGTH"] = GetValue(*seg,L);
    // Setting order
    (*new_unit)[INT]["ORDER"] = data.order;
    // Setting q
    (*new_unit)[DOUBLE]["Q"]      = 0.;
    // Setting scale
    (*new_unit)[INT]["SCALE"] = 1;
    // Setting gen
    (*new_unit)[INT]["GEN"]   = (int)
      (GetValue(GetTree(*seg),age) - GetValue(*seg,age));

    PositionVector sh = GetDirection(*seg);
    Point psh = (Point) sh;
    new_unit->sh(psh);

    // Setting panf and pend
    Point panf = GetPoint(*seg);
    double length = GetValue(*seg,L);
    Point pend = (length*(Point) sh) + panf;
    new_unit->panf(panf);
    new_unit->pend(pend);
    tree->pushBack(new_unit);
  }
  return tc;
};
```

```
// This operator is called by operator 2 of ConvertUnits<TS,BUD>.
// It ``physically'' adds the GroDisc tree segment to the LIGNUM tree.

template <class TS, class BUD>
gdTreeUnit& Lignum::ConvertUnits<TS,BUD>::operator()
  (gdTreeUnit& unit, TreeCompartment<TS,BUD>* tc)const
{
  if(Axis<TS,BUD>* axis    = dynamic_cast<Axis<TS,BUD>*> (tc)){
    Point panf     = unit.panf();
    Point pend     = unit.pend();
    Point sh       = unit.sh();
    PositionVector dir(sh.getX(),sh.getY(),sh.getZ());
    double order   = (double) unit[INT]["ORDER"];
    double length = unit[DOUBLE]["LENGTH"];
    double radius = unit[DOUBLE]["ADUR"];
    int     nr     = unit[INT]["NR"];
    double nad     = unit[DOUBLE]["NAD"];

    Tree<TS,BUD>& tree = GetTree(*axis);
    list<TreeCompartment<TS,BUD>*>& tc_list = GetTreeCompartmentList(*axis);
    list<TreeCompartment<TS,BUD>*>::iterator next = tc_list.begin();
    list<TreeCompartment<TS,BUD>*>::iterator last = tc_list.end();
    //  list<TreeCompartment<TS,BUD>*>::iterator it_term_bud = --last;

    while(next!=last){
      if(TS* seg = dynamic_cast<TS*>(*next)){
        if(seg->id==unit[INT]["MNR"] &&
           (int) GetValue(*seg,omega)==unit[INT]["ORDER"]){
          TS* new_seg = new TS(panf,dir,order,length,radius,radius,&tree);
          SetValue(*new_seg,lig_val_type,nad);
          new_seg->id = nr;
          BranchingPoint<TS,BUD> *bp =
            new BranchingPoint<TS,BUD>(panf,dir,&tree);
          InsertTreeCompartmentSecondLast(*axis,new_seg);
          InsertTreeCompartmentSecondLast(*axis,bp);
          break;
        }
        if(seg->id==unit[INT]["MNR"] &&
           (int) GetValue(*seg,omega)<unit[INT]["ORDER"]){
          TS *new_seg = new TS(panf,dir,order,length,radius,radius,&tree);
          new_seg->id=nr;
          SetValue(*new_seg,Wf,nad);
          BranchingPoint<TS,BUD> *bp      =
            new BranchingPoint<TS,BUD>(pend,dir,&tree);
          Bud<TS,BUD>               *bud     =
            new Bud<TS,BUD>(pend,dir,unit[INT]["ORDER"],&tree);
          Axis<TS,BUD>              *n_axis =
            new Axis<TS,BUD>(pend,dir,&tree);

          InsertTreeCompartment(*n_axis,new_seg);
          InsertTreeCompartment(*n_axis,bp);
          InsertTreeCompartment(*n_axis,bud);

          next++;
          if(BranchingPoint<TS,BUD>* bp_insert =
             dynamic_cast<BranchingPoint<TS,BUD>*>(*next))
            InsertAxis(*bp_insert,n_axis);
          break;
        }
      }
      next++;
    }
  }
  return unit;
};

// This operator adds each tree segment from the gdTree to the LIGNUM
// tree by invoking operator 3 of ConvertUnits<TS,BUD>.
void ConvertUnits<TS,BUD>::operator ()
   (Tree<TS,BUD>& tree, gdLayerCompartment* lc)const{
   if(gdTreeUnit* unit = dynamic_cast<gdTreeUnit*>(lc){
      Accumulate(*tree,*unit, ConvertUnits<TS,BUD> ());
   }
}

// This function starts the converting process from a LIGNUM tree to
// a GroDisc tree.

template <class TS, class BUD>
gdTree& ConvertTree(Tree<TS,BUD>& lig_tree,gdTree& gd_tree){
  ConvertUnitsData data;
  data.order  = -1;
  data.mother =  0;
  ConvertUnits<TS,BUD>::tree = &gd_tree;
  // This PropagateUp translates the tree. The following ForEaches
  // set the identifier and the mothenumbers.
  PropagateUp(lig_tree,data, ConvertUnits<TS,BUD> ());
  ForEach(gd_tree,CreateNumbers ());
  ForEach(gd_tree,CreateMotherNrs ());
  return gd_tree;
}

// This function start the converting process from a GroDisc tree to
// a LIGNUM tree. The LIGNUM tree has to be initialized by adding an
// Axis to the tree which contains the first element.

template <class TS, class BUD>
void ConvertTree(gdTree& gd_tree, Tree<TS,BUD>& lig_tree){
  if(!gd_tree.isEmpty()){
    if(gdTreeUnit *unit = dynamic_cast<gdTreeUnit*>(gd_tree.begin())){
      Point panf = unit->panf();
      Point pend = unit->pend();
      Point sh   = unit->sh();
      double order  = (double) (*unit)[INT]["ORDER"];
      double length = (*unit)[DOUBLE]["LENGTH"];
      double radius = (*unit)[DOUBLE]["ADUR"];
      double nad    = (*unit)[DOUBLE]["NAD"];
      PositionVector dir(sh);

      Axis<TS,BUD>& axis = GetAxis(lig_tree);
      TS *segment =
                  new TS(panf,dir,order,length,radius,radius,&lig_tree);
      segment->id = (*unit)[INT]["NR"];
      SetValue(*segment,ConvertUnits<TS,BUD>::lig_val_type,nad);
      BranchingPoint<TS,BUD> *bp =
                  new BranchingPoint<TS,BUD>(pend,dir,&lig_tree);
      Bud<TS,BUD>*  bud  = new Bud<TS,BUD>(pend,dir,0,&lig_tree);

      InsertTreeCompartment(axis,segment);
      InsertTreeCompartment(axis,bp);
      InsertTreeCompartment(axis,bud);
      Accumulate(gd_tree,lig_tree,ConvertUnits<TS, BUD> ());
    }
  }
  else
    cout<<"The␣gdTree␣seems␣to␣contain␣weird␣members."<<endl;
}
```

Appendix 3: Sensitive functions of *GROGRA*

Function 14

```
float function14(int ib, struct spross hu *current)
{
  /* Basicially this function calculates distances between pend of
  'spross' and all triangles generated by the T(a,b,c)
  turtle command. This function is firstly used to calculate
  distances between tree crowns represented by unregular
  pyramides.  */

  float result;
  float detM;                /* Determinant of coefficent matrix M.      */
  struct vektor n;           /* Right side of equation system.           */
  struct vektor x;           /* Solution of equation system.             */
  struct vektor bvec, cvec;  /* Vectors of plane (Add.
  lauftr->node[0]).z.ss */
  struct triangle *lauftr;   /* Variable for stepping through
  triangle list. */
  float det1, det2, det3;

  result = F_LARGE;
  for (lauftr = firsttriangle; lauftr != NULL; lauftr = lauftr->next)
  {
    /**** Calculating avec and bvec from nodes of triangle ****/
    bvec = vediff((lauftr->node[1])->z.ss,
    (lauftr->node[0])->z.ss); /* B-A */
    cvec = vediff((lauftr->node[2])->z.ss,
    (lauftr->node[0])->z.ss); /* C-A */

    /**** Calculating right side of system ****/
    n = vediff(current->pend, (lauftr->node[0])->z.ss);

    /**** Calculating the determinant of M ****/
    detM = determ(current->sh, bvec,cvec);

    if (detM > epsilon || detM < -epsilon)
    {
      /**** Calculating solution using det of 'Cramer matrices' ****/
      det1 = determ(n, bvec, cvec);
      det2 = determ(current->sh, n, cvec);
      det3 = determ(current->sh, bvec, n);
      x.x  = det1 / detM;
      x.y  = det2 / detM;
      x.z  = det3 / detM;
      x.x *= -1;

      /**** If para_result.y and para_result.z is positive and the sum
      is between 0 and 1 then point inside triangle. ****/
      if ((x.y>0 && x.z>0) && (x.y+x.z)<1.)
      {
        /**** We are looking for the lowest value. To avoid
        calculating distances very behind point, we assume
        that no intersection
        is existing 'behind' startpoint of spross. ****/
        if(x.x < result && x.x >= -(current->laenge))
        result = x.x;
      }
    }
  }
  return result;
}
```

Function 15

```
float function15(int ib, struct spross hu *current, int co)

/* like function8, but each touched sky segment is weighted
with the sine of the height angle of its corresponding
direction. The result is again normalized, i.e. divided
by the weighted sum of all possible sky segments. */
{
  short touchedsegs[nbskyseg];
  int segindx, ij;
  float currz, weight, sumtouch, sumall, result;
  struct spross *sh;

  if (current == NULL)
  {
    sprintf(s, "\nAbnormal␣situation␣in␣function␣15␣(current=NULL)!\n"); w();
    return 0.;
  }

  for (ij=0; ij < nbskyseg; ij++)   /* Initialisation */
  touchedsegs[ij] = 0;
  currz = (current->pend).z;
  for (sh = basis[ib]; sh != NULL; sh = sh->nachf)
  {
    if (((sh->panf).z > currz) && (sh != current)
    && ((co < 0) || ((sh->farbe) == co)))
    {
      segindx = skyseg(vediff(sh->panf, current->pend));
      if ((segindx >= 0) && (segindx < nbskyseg))
      touchedsegs[segindx] = 1;
    }
    if (((sh->pend).z > currz) && (sh != current)
    && ((co < 0) || ((sh->farbe) == co)))
    {
      segindx = skyseg(vediff(sh->pend, current->pend));
      if ((segindx >= 0) && (segindx < nbskyseg))
      touchedsegs[segindx] = 1;
    }
  }
  sumtouch = sumall = 0.;
  for (ij = 0; ij < nbskyseg; ij++)
  {
    weight = turtsky[ij].z / norm(turtsky[ij]);
    /* = sin(height angle) of turtsky */
    sumall += weight;
    if (touchedsegs[ij] > 0)
    sumtouch += weight;
  }
  result = sumtouch / sumall;
  return result;
}
```

MTG interface function of *Grogra*

```
void writemtghead(FILE *mtgf)
{
  fprintf(mtgf, ”# mtg file created by GROGRA\n\n”);
  fprintf(mtgf, ”CODE:\tFORM-A\n\n”);
  fprintf(mtgf, ”CLASSES:\n”);
  fprintf(mtgf, ”SYMBOL\tSCALE\tDECOMPOSITION\tINDEXATION\tDEFINITION\n”);
  fprintf(mtgf, ”$\t0\tFREE\tFREE\tIMPLICIT\n”);
  fprintf(mtgf, ”P\t1\tCONNECTED\tFREE\tEXPLICIT\n”);
  fprintf(mtgf, ”U\t2\t<-LINEAR\tFREE\tIMPLICIT\n”);
  fprintf(mtgf, ”I\t3\t<-LINEAR\tFREE\tEXPLICIT\n\n”);
  fprintf(mtgf, ”DESCRIPTION:\n”);
  fprintf(mtgf, ”LEFT\tRIGHT\tRELTYPE\tMAX\n”);
  fprintf(mtgf, ”U\tU\t<\t1\n”);
  fprintf(mtgf, ”U\tU\t+\t?\n”);
  fprintf(mtgf, ”I\tI\t<\t1\n”);
  fprintf(mtgf, ”I\tI\t+\t?\n\n”);
  fprintf(mtgf, ”FEATURES:\n”);
  fprintf(mtgf, ”NAME\tTYPE\n”);
  fprintf(mtgf, ”XX\tREAL\n”);
  fprintf(mtgf, ”YY\tREAL\n”);
  fprintf(mtgf, ”ZZ\tREAL\n”);
  fprintf(mtgf, ”Dia\tREAL\n”);
  fprintf(mtgf, ”Leafval\tREAL\n\n”);
}


void mtgcodedfs(struct spross *bs, FILE *mtgf)
{
  struct spross *nac;  /* potential successor shoot in dfs order */
  struct spross *nace; /* true successor shoot in dfs order */
  struct spross *nacs; /* order-preserving successor shoot in dfs order */
  struct spross *bb;   /* running thru structure */
  long lnr, durchl, plnr;  /* plnr = number of plant */
  int erf, indx, undx, ii, abzw, fskip, newpl, fstop, maxord, minord;
  if (bs == NULL)
    {
      sprintf(s, ”\nExceptional situation in mtgcodedfs (bs == NULL).\n”);
      w();
    }
  else
```

```
{
  lnr = (long) 0;
  maxord = 0;
  minord = 10000;
  for (bb = bs; bb != NULL; bb = bb->nachf)
    {
      lnr++;
      bb->yke = 0.;  /* overwritten here */
      bb->yka = 0.;
      bb->xke = 0.;
      if (bb->or > maxord)
        maxord = bb->or;
      if ((bb->or >= 0) && (bb->or < minord))
        minord = bb->or;
    }
  /* lnr is now the total number of el. units, including bs.
     maxord is the maximal order, minord the min. order >= 0. */
  maxord++;       /* one extra TAB necessary */
  sprintf(s, "\nTotal number of shoots: %ld\n", lnr); w();
  fprintf(mtgf, "MTG:\nENTITY-CODE");
  for (ii=1; ii <= maxord-minord; ii++)
    fprintf(mtgf, "\t");  /* fill with TABs */
  fprintf(mtgf, "XX\tYY\tZZ\tDia\tLeafval\n\n");

  bb = bs;  /* now the proper dfs run */
  plnr = durchl = (long) 0;
  while (bb != NULL)  /* main loop */
    {
      durchl++;
      if (durchl % 100 == 0)
        {
          if (langu == 0)
            sprintf(s, "  ** Bitte warten. Schleife von mtgcodedfs,"
                    "Durchlauf %ld\n", durchl);
          else
            sprintf(s, "  ** Please wait. Loop of mtgcodedfs,"
                   "cycle %ld\n", durchl);
        w();
        }
     newpl = 0;
     if ((bb->mutter == NULL) && (bb->yke < .2))  /* new plant */
        {
        plnr++;
```

```
   newpl = 1;
   fprintf(mtgf, "/P%ld", plnr);
   for (ii = 1; ii <= maxord-minord; ii++)
      fprintf(mtgf, "\t");  /* fill with TABs */
   fprintf(mtgf, "%f\t%f\t%f\t0\t0\n",
      (bb->panf).x, (bb->panf).y, (bb->panf).z);
   }
/* mark: */
abzw = 0;
fskip = 0;
if (bb->yke < .2)  /* shoot not yet considered */
   {
   if (bb->mutter != NULL)
      {
      if (bb->or == (bb->mutter)->or)
         {
         if (bb->vorw == (bb->mutter)->vorw) /* inside same U */
            {
            bb->yke = (bb->mutter)->yke + 1.;
            bb->xke = (bb->mutter)->xke;
            }
         else
            {
            bb->yke = 1.;  /* first I of new U */
            bb->xke = (bb->mutter)->xke + 1.;
            }
         }
      else
         {
         (bb->mutter)->yka += 1.;
         bb->xke = (bb->mutter)->yka;  /* index of U */
         bb->yke = 1.;                 /* index of I */
         abzw = 1;
         }
      }
   else
      {
      if ((bb->farbe == unsichtb) && (bb->laenge < epsilon))
         {
         bb->yke = bb->xke = .3;
         fskip = 1;
         }
      else
```

```
            bb->yke = bb->xke = 1.;
        abzw = 0;
        }
    if (!fskip)
        {
        /* write the corresponding line in mtg file: */
        if (bb->mutter != NULL)
            {  /* this is an ugly exception for the dummy shoot */
            if (((bb->mutter)->farbe == unsichtb) &&
                 ((bb->mutter)->laenge < epsilon))
                newpl = 1;
            }
        indx = integ(bb->yke);
        undx = integ(bb->xke);
        for (ii = 1; ii <= (bb->or)-minord; ii++)
            fprintf(mtgf, "\t");
        if (newpl)
            fprintf(mtgf, "^/");
        else
            {
            if (!abzw)
                fprintf(mtgf, "^<");
            else
                fprintf(mtgf, "+");
            }
        if (bb->mutter == NULL)
            fprintf(mtgf, "U%d/I1", undx);
        else
            {
            if (bb->vorw != (bb->mutter)->vorw)
                fprintf(mtgf, "U%d/I1", undx);
            else
                fprintf(mtgf, "I%d", indx);
            }
        for (ii = (bb->or)+1; ii <= maxord; ii++)
            fprintf(mtgf, "\t");
        fprintf(mtgf, "%f\t%f\t%f\t%f\t%f\n",
            (bb->pend).x, (bb->pend).y, (bb->pend).z, bb->edur,
            bb->nad);
        }
    }
/* now look for daughter shoots of bb to proceed with dfs */
erf = 0;
```

```
nacs = (struct spross *) NULL;
nac = bs;
while ((nac != NULL) && (erf == 0))
   {
   if ((nac->mutter == bb) && (nac->yke < .5))
      { /* appropriate daughter found */
      if (nac->or == bb->or)
         nacs = nac;
      else
         {
         nace = nac;
         erf = 1;
         }
      }
   nac = nac->nachf;
   }
if (!erf && (nacs != NULL))
   {
   nace = nacs;
   erf = 1;
   }
if (!erf)
   {
   if (bb->mutter != NULL)  /* go down */
      bb = bb->mutter;
   else  /* look for next plant, go to basis of next plant */
      {
      fstop = 0;
      while ((bb != NULL) && (!fstop))
         {
         if (bb->yke < .2)
            fstop = 1;
         else
            bb = bb->nachf;
         }
      if (bb != NULL)
         {
         while (bb->mutter != NULL) /* go to basis of next plant */
            bb = bb->mutter;
         if (bb->yke > .5)
            {
            sprintf(s, ”\nWarning: Inconsistency in mtgcodedfs\n”);
            w();
```

```
                        bb = (struct spross *) NULL;
                        }
                    }
                }
            }
        else
            bb = nace;
        }  /* while */
    }  /* else */
}
```

Appendix 4: L-systems

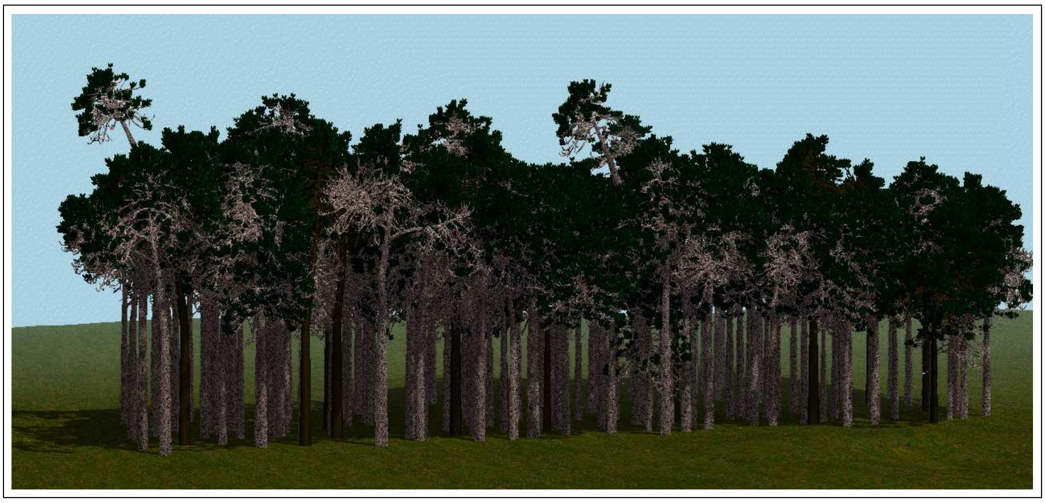
Reconstruction

```
\var i   index,
\var dtt function 14 0,

\var data array 1 2 16 151,

\const anzw    8,
\const trees 150,
\const X       2,
\const Y       3,
\const BHD     4,
\const H       5,
\const CB      6,
\const KR      7,
\const DIES   15,

\var eva function 30 3,

\var dres  normal 0 0.0017,
\const dslo 0.0210,
\const dint 0.0041,

\var cbres normal 0 0.3,
\const cbslo -0.067,
\const cbint  1.077,

\var chres normal 0 0.2887,
\const chslo -0.0783,
\const chint  0.5026,

\var ceres normal 0 0.0880,
\const ceslo  -0.068,
\const ceint   0.208,

\axiom stand 1-8,

stand # &(trees) < [ RU(90) f(data(i,X)) RL-90 f(data(i,Y)) RU-90
  cut(0,data(i,DIES)) pinus(i) ] >   ,
pinus(ind) #  stem(data(ind,BHD),dres,0,data(ind,CB),cbres,0,0) RH135 RU-90
  &(anzw) < [ RL(i*(-(360/anzw))) crext(data(ind,KR+i),ceres,0,0)
  S(i+(anzw+1)*ind) ] > RU(90)
  crtop(data(ind,H)-data(ind,CB),chres,0,0) S(anzw+ind*(anzw+1))
  &(anzw) < C(i+ind*(anzw+1),anzw+ind*(anzw+1)) >
  &(anzw-1) < C(i+ind*(anzw+1),(i+1)+ind*(anzw+1)) >
  C((anzw-1)+ind*(anzw+1),ind*(anzw+1))
  &(anzw-1) < T(i+ind*(anzw+1),(i+1)+ind*(anzw+1), anzw+ind*(anzw+1)) >
  T((anzw-1)+ind*(anzw+1),ind*(anzw+1),anzw+ind*(anzw+1)),

stem(d,dr,di,cb,cbr,cbi,t) # stem(d,dr,(d*dslo+dint+dr),cb,
                                   cbr,(cb*cbslo+cbint+cbr),t+1) ,

(dtt>(-0.86)) crext(ce,cer,cei,t) # crext(ce,cer,(ce*ceslo+ceint+cer),t+1) ,
              crext(ce,cer,cei,t) # crext(ce,cer,cei,t+1),

crsurf(ind) # crsurf(ind),

crtop(ch,chr,chi,t)  #  crtop(ch,chr,chslo*ch+chint+chr,t+1) ,

(t>1 && d=0) cut(t,d) # % ,
             cut(t,d) # cut(t+1,d),


stem(d,dr,di,cb,cbr,cbi,t) ## P(7)  Dl((d+t*di)/100) N(0) F(cb+t*cbi) ,
crext(ce,cer,cei,t)        ## P(10) Dl(0.01)          N(1) F(ce+t*cei) ,
crtop(ch,chr,chi,t)        ## P(11) Dl(0.01)          N(1) F(ch+t*chi) ,
```

Juvenile Scots pines in a gap

```
\const trees  1,
\const npalis 250,
\const colpalis 9,

\var w   uniform  0 360,
/*\var r   uniform  0  21,*/
\const r 21.0,

\const h_stand  19.7,
\const cb_stand 0.0,
\const rad_gap  21,
\const x_c      50,
\const y_c      50,

\var c_x xcoordinate,
\var c_y ycoordinate,

\var i index,
\var sky function 8 0,

\const anzw   8,

\const h_int   0.1231
\const h_slo   0.1241
\const d_slo   0.1809
\const d_int   0.0317
\const cre_int 0.0627
\const cre_slo 0.0527


\axiom * 1-25,

* # [ roof ] [ stand ],

roof # RL-90 f(y_c) RU(90) f(x_c) &(npalis) < RU(137.5) [ f(3.16*sqrt(i))
         RL(90) f(cb_stand) palis(0) ]  > ,

palis(disttg) # palis(sqrt((x_c-c_x)^2+(y_c-c_y)^2)),
stand # RL-90 f(y_c) RU(90) f(x_c) &(trees) < [ RU(w) f(r) RL(90) pinus(i) ] >,

pinus(ind) #  urstem(ind,0,0,0,0,0),

urstem(ind,h,d,cre,crt,t) # stem(ind,h,d,cre,crt,t)
  S(anzw+ind*(anzw+1)) crtop(crt,t) RU-90
  &(anzw) < [ RL(i*360/anzw) crext(cre) S(i+(anzw+1)*ind) ] > RU(90)
  &(anzw)   < C(i+ind*(anzw+1),anzw+ind*(anzw+1)) >
  &(anzw-1) < C(i+ind*(anzw+1),(i+1)+ind*(anzw+1)) >
  C((anzw-1)+ind*(anzw+1),ind*(anzw+1))
  &(anzw-1) < T(i+ind*(anzw+1),(i+1)+ind*(anzw+1), anzw+ind*(anzw+1)) >
  T((anzw-1)+ind*(anzw+1),ind*(anzw+1),anzw+ind*(anzw+1)),


(sky>0)  stem(ind,h,d,cre,crt,t) # urstem(ind,
                                           h   + (h_slo  *(1-sky_) + h_int),
                                           d   + (d_slo  *(1-sky_) + d_int),
                                           cre + (cre_slo*(1-sky_) + cre_int),
                                           h   + (h_slo  *(1-sky_) + h_int),t+1),
crext(cre)     # ,
crtop(crt,t)   # ,


(disttg>rad_gap) palis(disttg) ## P(colpalis) D(1.3) F(h_stand),
                 palis(disttg) ## ,

stem(ind,h,d,cre,crt,t) ## Dl(d)    P(7)  N(0) F(h)       ,
crext(cre)              ## Dl(0.01) P(10) N(1) F(cre)     ,
(t<5) crtop(crt,t)      ## Dl(0.01) P(11) N(0) F(-crt)    ,
      crtop(crt,t)      ## Dl(0.01) P(11) N(0) F(-crt+0.3) ,
```

Appendix 5: *AMAP* pictures of Warwe



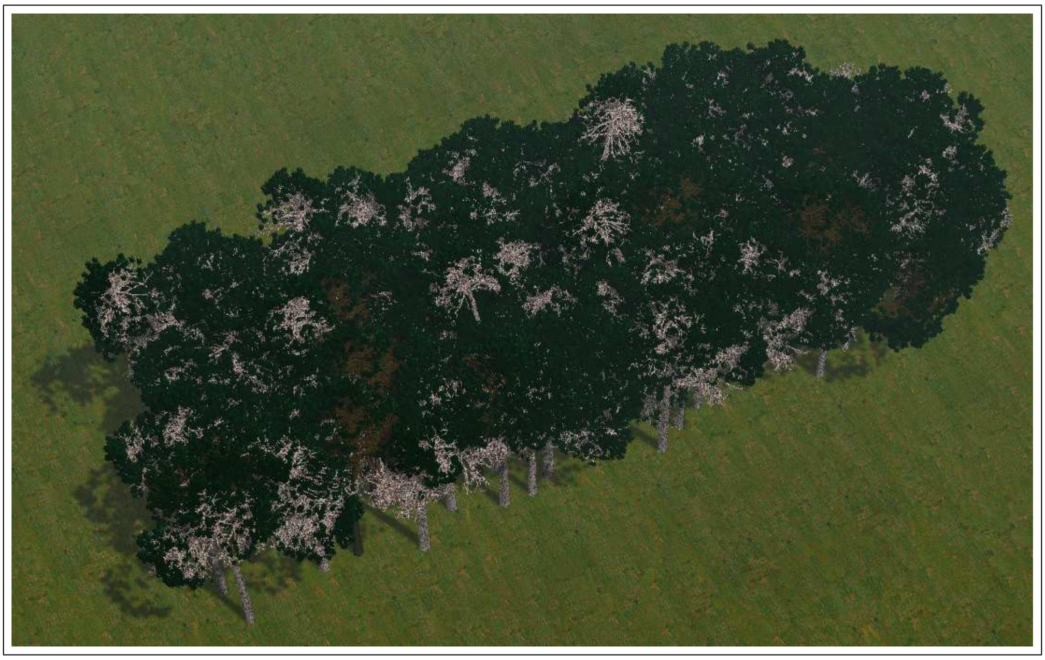
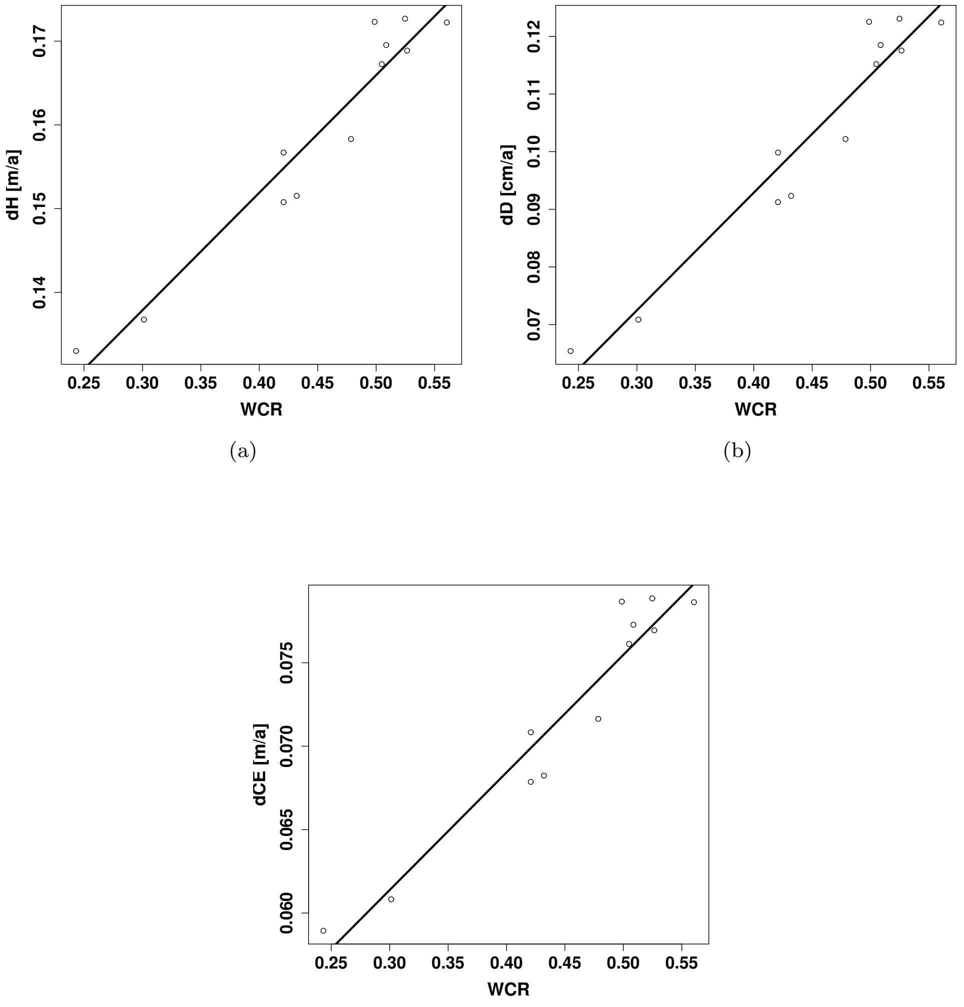


Figure 53: The Scots pine trees of Warwe in 1995, side view (Visualized by *AMAP*) (see Duchiron (2003)).

Appendix 6: Aggregation of *LIGNUM-canopy*

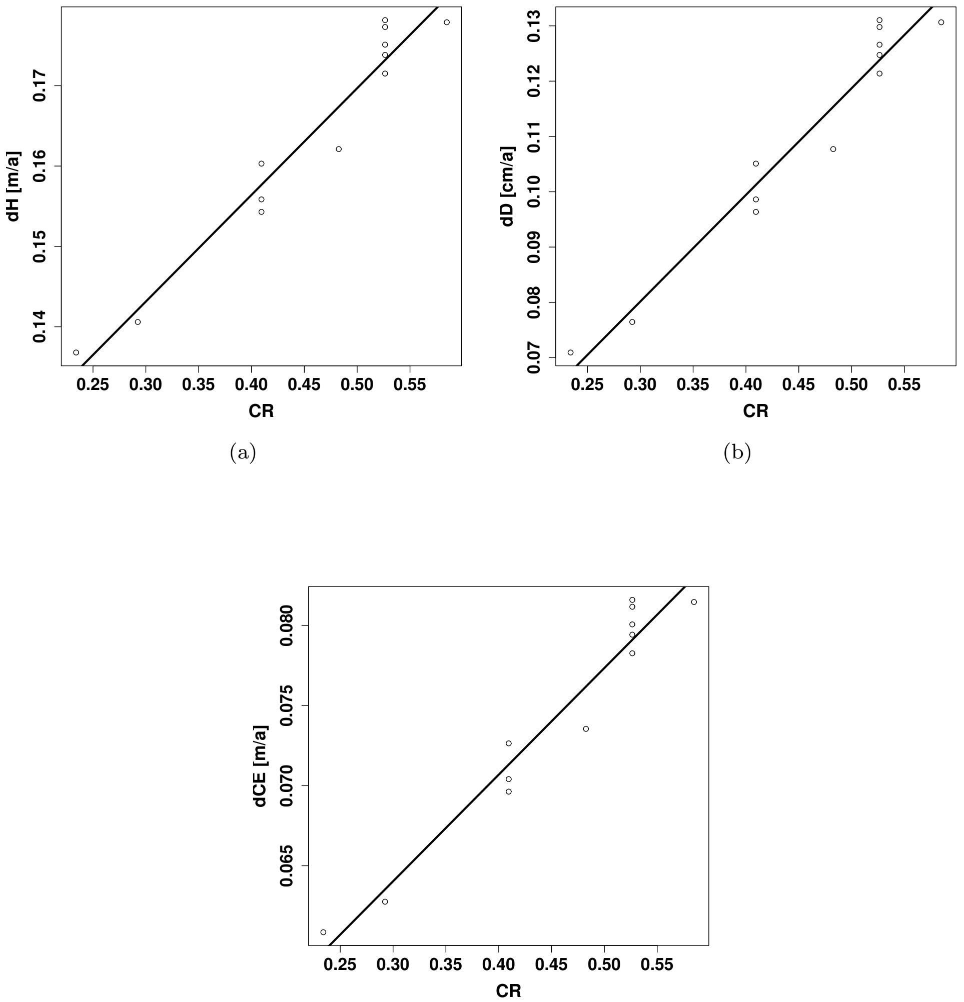


(a)

(b)

(c)

Figure 54: The development of the investigated parameters height (dH) (a), diameter (dD) (b) and mean crown extension (dCE) (c) in relation to the weighted covering ratio WCR calculated by the sensitive function 15.



(a)

(b)

(c)

Figure 55: The development of the investigated parameters height (H) (a), diameter (D0.0) (b) and mean crown extension (CE) (c) in relation to the covering ration CR calculated by sensitive function 8.

Wilamowitzweg 18
D-37085 Göttingen
GERMANY

+49 551 399715
hdzierz@gwdg.de

Helge Dzierzon

Personal Information

Marital status: unmarried

Nationality: German

Age: 33

Place of birth: Osterode, Germany

Working Experience

1997

Niehaus Umwelt Consulting, Hann Münden, Germany
Independent collaborator

- Data processing and map creation from forest management data.

1997

Research Centre of Lower Saxony, Göttingen
Independent collaborator

- Department of environmental controls.
- Data collection and statistical analysis.

1998

Institute for Forest Biometry and applied Computer Science, University of Göttingen

Independent collaborator

- Planning of (a) stratified sampling for the estimation of data relevant to management of a 17700 ha forestry district.

1998

Institute for Forest Biometry and applied Computer Science, University of Göttingen

Independent collaborator

- Different tasks.

1998–1999

Institute for Forest Biometry and applied Computer Science, University of Göttingen

Employee

- Automating of map creation processes using ArcView and its internal language Avenue in a dendrochronological project.

1999–2000

Finnish Forest Research Institute Vantaa, Finland

Volunteer

- DAAD scholarship.
- Simulating virtual Scots pine trees under different light regimes caused by a virtual surrounding stand.

2000-2003

Institute for Forest Biometry and applied Computer Science, University of Göttingen

Employee

- Person in charge for the project: “Development of methods for characterizing plant and stand architectures and for model comparisons.”

2003

Institute for Forest Biometry and applied Computer Science, University of Göttingen

Employee

- Member of a GIS subproject of the project: “Indicators and strategies for a multi functional and sustainable forest use. A Case study of the forest landscape Solling Mountains.”

Education

1991-1993	Undergraduate Student	University of Göttingen

- Forest Science

1993-1997	Graduate Student	University of Göttingen

- Forest Science

2000-2003	Doctoral Student	University of Göttingen

- Forest Science

Summary of Qualifications

Diplomforstwirt (Master of Forest Science eq.)

Dr. forest. disp. (PhD eq.)

Languages

English, fluent

Computational Skills

Programming languages: C, C++, Avenue, SAS, R, L-Systems

GIS applications: ArcView, Arc Info

Statistical applications: SAS, R, STATISTICA

Modelling applications: GROGRA, Stella

Interests and Activities

Swimming, cross country skiing, music (piano)

Community Activities

Member of the town council of Göttingen

Publications

Dzierzon, H. and Kurth, W.: LIGNUM: A Finnish tree growth model and its interface to the French AMAPmod database. **In**: F. Hölker (editor) Scales, hierarchies and emergent properties in ecological models. Theorie in der Ökologie 6: pages 95–104. Peter Lang Verlag, Frankfurt/M., 2002.

Dzierzon, H., Perttunen, J., Kurth, W., Sievänen, R. and Sloboda,B.: Enhanced possibilities for analyzing tree structure as provided by an interface between different modelling systems. Silva Fennica, 37(1): pages 31–44, 2003.

Oppelt, A. L., Kurth, W., Dzierzon, H., Jentschke, G. and Godbold, D. L.: Structure and fractal dimensions of root systems of four co-occurring fruit tree species from Botswana. Ann. For. Sci. 57(5/6): pages 463–475, 2002.