A FRAMEWORK FOR THE DESIGN OF SIMULATION-BASED GREENHOUSE CONTROL

By

René Lacroix

A Thesis submitted to the Faculty of
Graduate Studies and Research in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

Department of Agricultural Engineering
McGill University, Montréal
© René Lacroix, March 1994

ISBN 0-315-94653-9

ABSTRACT

René Lacroix Ph.D. (Agric. Eng.)

SIMULATION-BASED GREENHOUSE CONTROL

The main objectives were: 1) to develop tools to aid in the design of enclosed agro-ecosystems, and 2) to use these tools to develop a prototype simulation-based control system. Three tools were developed: 1) a conceptual framework, 2) a (simulated) greenhouse system and 3) a simulation approach within OS/2.

Part of the conceptual framework was dedicated to "conscious control", defined as a form of control practised by an entity that uses models of itself in its decision-making processes. The greenhouse system was composed of six modules (a simulation manager, a weather generator, a greenhouse model, a crop model, a Pavlovian controller and a cognitive controller), which were implemented under OS/2 as separate processes.

The greenhouse system was used to develop a prototype simulation-based controller. Primarily, the role of the controller was to determine temperature setpoints that would minimize the heating load. The simulation model used by the controller was an artificial neural network. The controller adapted temperature setpoints to anticipated meteorological conditions and reduced greenhouse energy consumption, in comparison with a more traditional controller.

Generally, the results showed the feasibility and illustrated some of the advantages of using simulation-based control. The research resulted in the definition of elements that will allow the creation of a methodological framework for the design of simulation-based control and, eventually, a theory of conscious control.

RESUME

René Lacroix Ph.D. (Génie Agr.)

LE CONTROLE DES SERRES BASE SUR LA SIMULATION

Les principaux objectifs étaient: 1) d'élaborer un certain nombre d'outils pouvant aider au design d'agro-écosystèmes clos et 2) d'utiliser ces outils pour développer un prototype de système de contrôle ayant recours à la simulation dans ses prises de décision. Trois outils ont été développés: 1) un cadre conceptuel, 2) un système de simulation de serre et 3) une approche de simulation sous OS/2.

Une partie du cadre conceptuel a été dédiée au "contrôle conscient", défini comme une forme de contrôle pratiqué par une entité qui a recours à des modèles d'elle-même dans ses prises de décision. Le système de simulation de serre a été composé de six modules (un gestionnaire de simulation, un générateur de données météorologiques, un modèle de serre, un modèle de croissance de plantes, un contrôleur Pavlovien et un contrôleur cognitif) qui ont été implantés dans des processus informatiques différents sous OS/2.

Le système de simulation de serre a été utilisé pour développer le prototype d'un contrôleur ayant recours à la simulation dans ses décisions. Le rôle du contrôleur consistait principalement à déterminer des consignes de température minimisant les coûts de chauffage. Le modèle utilisé par le contrôleur était construit avec un réseau de neurones artificiels. Le contrôleur a adapté les consignes aux prévisions météorologiques et a réussi à diminuer les coûts énergétiques par rapport à un contrôleur plus traditionnel.

Les résultats ont démontré la faisabilité et illustré certains des avantages du contrôle basé sur la simulation. Les recherches ont aussi permis de déterminer des éléments qui permettront d'élaborer un cadre méthodologique pour le design de systèmes de contrôle basé sur la simulation et, éventuellement, une théorie du contrôle conscient.

ACKNOWLEDGEMENTS

I would first like to express my most profound thanks to my research director, Dr. Robert Kok, for his confidence, his encouragement and his support during my doctoral studies. His intellectual rigor and his search of innovation were, for me, important sources of inspiration and stimulation.

I would also like to thank the other professors of the Department of Agricultural Engineering for their advice and help at various moments during the last years. Especially, I would like to thank Dr. Eric Norris, a member of my doctoral committee and chairman of the department, whose help was precious on many occasions. Also, I would like to thank Dr. Shiv Prasher and Dr. John Sheppard for their participation on my doctoral committee.

I would like to express my deep gratitude to Dr. Laurent Gauthier, from the Department of Agricultural Engineering, Laval University, whose participation on my doctoral committee was greatly appreciated. I would like to thank him also for his support during my doctoral studies.

I sincerely acknowledge Darwin Lyew, student confrère, for having done the arduous task of reviewing my thesis. His comments were greatly appreciated. I would also like to extend my acknowledgements to the members of the administrative and technical staff of the Department of Agricultural Engineering. Their help was invaluable.

I thank Jean-Pierre Desjardins for having transmitted the meteorological data used in this research. I also thank Dr Jean Claude Zanghi, from UQAM (Université du Québec à Montréal), for his encouragement during all my graduate studies.

I would like to acknowledge my deep gratitude to the various organisms that supported this research and permitted its realisation. I thank the Graduate Faculty of McGill University, CORPAQ (Conseil de Recherches en Pêcheries et Agriculture du Québec), FCAR (Fonds pour la Formation de Chercheurs et l'Aide à la Recherche) and NSERC (Natural Sciences and Engineering Research Council of Canada).

Finally, I would like to express special thanks to my spouse, Ginette, for her understanding, her support and her encouragements during all of my graduate studies. I thank her enormously for her patience during all those years. I would like also to address special thanks to our sons, Marc-Olivier, who so often helped me to forget the "tracas" related to the realisation of a doctorate, and Louis Gabriel, our newborn.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

I.1 FROM NATURAL ECOSYSTEMS TO ENCLOSED AGRO-ECOSYSTEMS

Natural ecosystems are generally characterized by an inherent stability and by a capacity to adapt to changes. These attributes are largely due to the bio-diversity of the ecosystems and their complexity. Human beings learned early how to alter natural ecosystems to optimize some of the natural processes so as to grow larger quantities of some specific foods and fibres. Historically, one of the most important interventions has been the removal, either direct or indirect, from the production space, of species that were considered useless. This resulted in the breaking of the food chains and the disappearing of some of the prey-predator interactions, that play an important role in the maintenance of the population balance. For example, if trees are removed from a certain region, some bird species can no longer inhabit this zone and this can lead to an increase in the population of some insects, which in turn may result in considerable defoliation of the cultivated crop. Consequently, to avoid any instability caused by modifications of the natural ecosystems, humans early realized that some control action was necessary to compensate for the removal of inherent regulatory mechanisms and buffering capacities. For instance, the search for control mechanisms has culminated in the discovery of pesticides, that have been used over many decades to prevent crop devastation by insects. Also, nutrients have been applied in massive quantities to compensate for the lack of natural contribution of organic matter to maintain soil fertility.

Over the centuries, humans have started to construct envelopes to grow some species so as to optimize further some agricultural processes. Barns for animals have been used for a long time and, during the last century, glasshouses have been constructed for growing crops. The greenhouse industry has grown rapidly over the last few decades around the world and, in some countries such as in Japan, research is presently being done to construct "plant factories" (Takakura, 1991). In a few decades, there may be even a need to construct such production units in extraterrestrial regions such as on large-scale space stations and moon bases (Fortson, 1992; Salisbury, 1991; Mendell, 1985). By enclosing species of interest within envelopes, the control of the production environment is facilitated. This creates a large potential for the improvement of the growing conditions. Conversely, conditions are also created that favour the growth of some undesired species. For example, diseases can spread rapidly in a barn, transported by flies that multiply at a rhythm that could never be reached under natural conditions. In greenhouses, the high moisture content in some zones can contribute to the development of fungi. Within envelopes, the biodiversity and the buffering capacity are diminished to very low levels, resulting in potential instabilities that could be dramatic. Moreover, within enclosed environments, many abiotic factors rely exclusively on artificial mechanisms, e.g., water supply depends essentially on irrigation in greenhouses and barns. Thus, the whole population of species grown within an envelope can be eliminated rapidly if management practices are poor. However, in enclosed agro-ecosystems, it is possible to obtain growth rates that could never be obtained under natural circumstances. For this to occur, an adequate control is required, which in turn demands that the state of the system be constantly monitored and analyzed.

For many millennia, humans have been able to conceive and maintain viable agricultural systems. Now, it is even possible to construct enclosed agro-ecosystems that are stable. The required control actions have been developed through a long learning process, punctuated by much experimentation. This development has been the direct result of the human capacity to observe, analyze and understand physical processes, and their ability to synthesize new procedures based on this experience and knowledge.

I.2 SYSTEM AUTONOMY

To increase productivity while reducing the amount of tedious work and repetitive tasks, process automatization has been a priority for a long time. Like in many other sectors, mechanization in agriculture has been progressing rapidly since the industrial revolution started in the nineteenth century. With the advent of electronics, and mainly after the Second World War, complex control structures have been developed that give the possibility of conferring more autonomy to production processes. This autonomy is attractive, because it reduces the need for human effort, while improving the quality of control. Also, machines can work under conditions that are bad or dangerous to human beings. Now, with the rapid evolution of computer technology, it is even possible to conceive systems that are able to replace humans in some of their management tasks. For example, it is now feasible to create systems that mimic humans in their decision-making activities with respect to well defined problems. With the research that is being pursued in artificial intelligence, the systems that will be constructed in the future will be more able to work within complex contexts that are characterized by uncertainty, contradiction and partial information. System autonomy is desired not only at the physical level but also at the virtual level, because it releases humans from repetitive reasoning tasks. It also creates some new possibilities, such as the treatment of large amounts of information.

In enclosed agro-ecosystems, autonomy is also desired. This will be especially true in extraterrestrial stations, where there is a requirement to maintain a minimum number of staff members whose attention must be dedicated to the execution of high skill tasks. For an enclosed agro-ecosystem to be autonomous and, at the same time, stable, its control system should possess functionalities that mimic human cognitive faculties such as observation, reasoning, analysis and learning, which have permitted humans to develop relatively stable agro-ecosystems in the past. Artificial implementation of cognitive faculties can be achieved by creating control systems with sufficient intricacy, comprised of computers, where some would be able to process high-level knowledge. These computers would perform what is called "cognitive control".

An interesting aspect in the development of cognitive control systems is the design of mechanisms that possess a certain cognizance of the overall enclosed agro-ecosystem, including the controlling system itself. Such a control system would perform what is defined as "conscious control", which is considered to be a subset of cognitive control. One approach to constructing such a control system is to give it both the access to models of the overall system (including itself) and the capacity to use these models for simulation. One of the advantages of a system possessing a simulation capacity is that it would be able to predict its own response to some control scenarios and to anticipated disturbances. It would also be able to revise its past decisions based on the actual response of the system. It could be aware of its physical weaknesses, and even of its reasoning limitations.

I.3 RESEARCH OBJECTIVES

The design of autonomous and enclosed agro-ecosystems (AEAE) in which conscious control is practised presents enormous challenges. Research in this area has been initiated only recently and many aspects must be investigated before a physical implementation will be possible. For instance, factors to be considered for the design of such complex systems need to be determined and guidelines must be elaborated. The overall objective in this research was to make a contribution in this area. This was to be done by 1) developing some analytical tools that will help to the creation of complex AEAE's and 2) using these tools to construct a prototype controller that has recourse to simulation in its decision-making processes. The analytical tools to be developed were to be comprised of a conceptual framework, a (simulated) greenhouse system and a simulation approach within OS/2. The greenhouse was used as the primary reference system in this research, but the intent was to develop, as much as possible, tools that would be suitable for any AEAE.

The approach in this research was as follows. A conceptual framework for the design of enclosed agro-ecosystems was first developed. Such a framework is essential for establishing a basis for a systematic analysis and construction of "intelligent" agro-ecosystems, as has been done in a more general fashion for intelligent manufacturing systems (Kim, 1990). The framework was to be complete, coherent and integrated. It also had to be general enough for the concepts to be applicable to different AEAE's. The framework that was developed was divided into four major sections which are related respectively to: 1) the overall agro-ecosystem, 2) the controlling subsystem, 3) cognitive control and 4) conscious control. This conceptual framework is presented in Chapter III.

Following the conceptual framework, a simulation structure was constructed for the development and testing of cognitive control systems. The simulated system was comprised of a greenhouse, crops and an extrinsic control system, as described in Chapter IV of this thesis. The adoption of a simulated environment rather than a physical environment to develop cognitive systems is justified by many reasons. The most obvious one is that it decreases the resource requirements and the associated costs (e.g., greenhouse structures, their equipment and operation, data acquisition system, crop production). Moreover, it diminishes the number of physical constraints and hardware complications such as electrical and electronic connections, information transfers, equipment breakage and interferences. All the functions involved in physical control and that require a lot of processing do not need to be incorporated in the simulated system. Also, machines that possess a large computing capacity, but do not yet exist, can be simulated. Experiments can be completed more rapidly than in a physical milieu and without meteorological constraints. Inversely, many experiments can be done with the same set of meteorological conditions, which increases the number of configurations that can be compared. Any engineered system must pass through a physical validation phase before its implementation in a production context, but a lot of research must be done before reaching this stage. Simulation can potentially narrow the domain of physical experimentation to be done and, therefore, it facilitates a more rapid elaboration of design criteria and constraints. This is particularly true when information treatment mechanisms are developed, because many contexts can be artificially generated and presented to the mechanisms to test how they react.

In this research, the cognitive controller that was conceived was relatively simple, but the objective was to develop a simulation tool that would be suitable for more complex ones. Also, the simulation structure that was developed was comprised of many components that differ in both their role and nature. Thus, another important objective of this research was to investigate the possibility of implementing the simulation structure within the multi-tasking operating system OS/2. This would allow the implementation of the different components as distinct, interacting processes running concurrently. One advantage of this approach is that complex simulation structures could be implemented on only one computer. The approach that was developed to implement simulation structures within OS/2 will be explained in Chapter V.

Once the simulation structure was implemented, it was used to develop the prototype of a simulation-based cognitive controller. To avoid confusion in the text below, the qualifier "functional" will be utilised to refer to the simulation of the overall greenhouse system and the term "subjective" will be used to qualify simulations done by the cognitive controller. Primarily, the role of the controller was to determine temperature setpoints that minimize the heating load of the greenhouse, in consideration of the weather conditions anticipated for the following hours. The strategy was based on the assumption that crops possess a "temperature integration capacity", by which crops respond to an average temperature rather than to the absolute temperature path. The model used by the controller for its simulations was constructed with an artificial neural network. From the analysis of the behavior of the greenhouse system in the presence of this controller, the intent was to determine some constraints and requirements related to the construction of simulation-based control systems. This would form the basis for the development of a methodology for the design of simulation-based control of enclosed agro-ecosystems. The structure of the prototype simulation-based controller, and the approach used to develop it, will be presented in Chapter VI. Its effects on the behavior of the greenhouse system will be analyzed in Chapter VII.

To summarize, the objectives of this research were to:

1) develop a conceptual framework for the design of autonomous agro-ecosystems, using the greenhouse as a reference system,
2) conceive a simulated greenhouse system to help in the development of cognitive control systems,
3) implement the simulated greenhouse system within the multi-tasking operating system OS/2,
4) develop a cognitive controller prototype which has recourse to simulation in its decision-making activities,
5) evaluate the behavior of the (simulated) greenhouse system in the presence of the cognitive controller prototype and,
6) determine a set of requirements, constraints and factors to be considered for the design of simulation-based control systems.

II. LITERATURE REVIEW

II.1 INTRODUCTION

The literature review is divided into three major blocks. In the first part, greenhouse control is discussed in general terms to give an idea of its complexity, and to establish a basis for the development of a conceptual framework. The second part is devoted to the subject of greenhouse climate control and will furnish some guidance for the understanding of the control strategies used by the cognitive controller in this research. In the third part, the use of simulation as a tool for control purposes will be presented and the approach for directly integrating it in the control system will be introduced.

II.2 GREENHOUSE CONTROL SYSTEMS

Major developments have occurred in the greenhouse industry over the last few decades. New production techniques have been implemented (e.g., artificial substrates or nutrient film transient culture) in concert with innovative equipment (e.g., thermal screens, carbon dioxide generators, artificial lights). Greenhouse systems have become very complex and require equally complex control structures to maintain optimal growth conditions and to justify the significant investments involved. Fortunately, the cost of digital computers have been continuously decreasing and they can now be used widely for control purposes. Computers can provide the necessary flexibility and power, and will soon be an essential component of any greenhouse control system.

In parallel with the increasing complexity of the greenhouse systems, researchers have developed simple theoretical frameworks to help orient their research. Very early on, different levels of control activities were recognized. Udink ten Cate *et al.* (1978) proposed a hierarchy with three levels of decision. The first and lowest level corresponded to short term regulation, with time periods varying from seconds to minutes. Examples of metabolic functions considered at the lowest control level were photosynthesis, CO_2 absorption, translocation and transpiration. The realisation of setpoints for climate and root environment variables is one of the important functions at this level. The second level involved decisions for the present and the following few days. Crop growth and development are events occurring in this time frame. The determination of dynamic setpoints is realized at this level. At the third level, decisions are made for an overall growth season and take into account all crop development phases (e.g., seeding, flowering, fruiting, harvesting). Similar frameworks have been discussed by other authors (Bailey and Chalabi, 1991; Tantau, 1989; Kok and Desmarais, 1987; Challa, 1985) and, based on these frameworks, distributed control systems have been implemented in greenhouse research facilities (Jones *et al.*, 1989; Bakker *et al.*, 1988; Kok and Desmarais, 1985; Hoshi and Kozai, 1984). In these installations, low-level processors regulated climate and root environment in one or more growing zones according to setpoints fixed by high-level computers. The latter were more powerful than low-level computers and were used to execute complex functions that required extensive information treatment such as optimizations and consultation of expert systems. This brief description indicates the kind of systems that will be installed soon in some large greenhouse enterprises and crop factories. This also shows that there is a trend towards integrating into the same structure the different levels of control activity ranging from regulation to management.

Computer technologies have evolved to the point where computers are now able to participate in decision-making processes and replace the managers in some of their routine functions. For this purpose, recourse to simulated cognitive functions such as reasoning, learning, pattern recognition and prediction is essential and many researchers have worked in this direction for greenhouse control. Takakura *et al.* (1984) discussed the idea of implementing the expertise of the best growers in knowledge-based control systems. Kurata (1988) constructed an algorithm that learned control rules used by growers. Kano and Shimaji (1988) coupled an expert system with a crop model to control the concentration of nutrients in solutions. The expert system was based on the observations of growers. Jackson *et al.* (1989) conceived an expert system that chose setpoints for the frequency and duration of misting events. The expert system rules were acquired from an experienced person in this field. Similar applications have been discussed for greenhouse control (Boulard *et al.*, 1991; Fynn *et al.*, 1991; Gauthier and Guay, 1990; Jones *et al.*, 1989; Harazano *et al.*, 1988; Hirafuji *et al.*, 1988), for the aerospatial sector (Ihara *et al.*, 1989; Lum and Heer, 1988) and for the industrial world, globally (Antsaklis *et al.*, 1991; Stengel, 1991; Shoureshi, 1991; Berger and Loparo, 1989; Wright *et al.*, 1986).

Computing technologies will continue to evolve rapidly and, to exploit their full potential, it is necessary to elaborate on reference frameworks. Such frameworks do not exist in the greenhouse control field, despite the existence of a large quantity of knowledge around the world. Their integration into a single framework would furnish a powerful tool for orienting research and determining long term objectives. The necessity to develop such a conceptual tool arises from the need to be able to 1) cope with the increasing complexity of the greenhouse production systems and to 2) conceive the appropriate control systems that will allow the integration of cognitive aspects.

II.3 GREENHOUSE CLIMATE CONTROL

II.3.1 Setpoint determination vs realisation

Two control steps have generally been recognized in greenhouse climate control: 1) the determination of setpoints and 2) their realisation (Lessard, 1989; Seginer, 1980). In the following discussion, these two topics are presented in the reverse order, so as to follow the chronological development of the importance attributed to each aspect. Strategies and actions that allow a control system to realize desired setpoint values have been extensively studied in classical control theory that generally deals with predetermined objectives to be reached. The role of classical control systems is mainly to counteract the undesired effects of disturbances on the system behavior with the help of feedback, anticipatory or adaptive control algorithms. The level of sophistication of the algorithms depends on the equipment used to regulate the processes. In the case of greenhouses, many control elements possess only one or two power levels (e.g., ventilators, gas burning CO_2 injectors, artificial lights) and, therefore, are subjugated to ON-OFF control. More sophisticated controls like P, PI or PID have been used with systems where power can be modulated, e.g., water distributed heating systems (Hooper, 1988; Valentin and Van Zealand, 1980), liquid CO_2 generator (Bakker *et al.*, 1988) and ventilation systems (Hooper, 1988). In the same vein, Hooper and Davis (1985) used adaptive control to determine the degree of opening of vents. Anticipatory control has been used for CO_2, where crop and ventilation models were used to predict CO_2 needs for the following minutes (Jones *et al.*, 1989; Matthews *et al.*, 1987; Bakker, 1985; Montheith *et al.*, 1983). Because of the specificities of the variables and the equipment involved in greenhouse production, original strategies have also been developed for the realization of setpoints, such as knowledge-based control of natural ventilation (Sase and Nara, 1985).

Setpoint determination constitutes the other important aspect of greenhouse control and, in fact, seems to have become a priority in this field of research. The first phase of greenhouse climate automation corresponded to the introduction of analog devices and was mostly dedicated to regulation, i.e., to setpoint realization. With the introduction of computers for control, setpoints have been progressively allowed to fluctuate with time. This has opened a wide range of control possibilities, which can be exploited by the specific demands of the greenhouse production processes, where the global optimum for temperature, carbon dioxide, luminosity and nutrients continuously varies as a function of the stage of crop growth, the meteorological conditions and the economic context. The present trend in greenhouse control research is to develop methods that continuously find the setpoints that are the most appropriate to each context. Berg and Lentz (1989) concluded that temperature setpoints can be matched very accurately and that the remaining problem was to find optimal setpoints; this could be true for all controlled variables.

In this regard, optimisation algorithms have been used to determine optimal temperature setpoints (Seginer *et al.*, 1991; Schapendonk *et al.*, 1984; Challa *et al.*, 1980) and CO_2 concentration (Critten, 1991; Bakker *et al.*, 1988; Nederhoff, 1988; Challa and Schapendonk, 1986). Seginer (1980) suggested that this approach could be used with any controlled variables. In general, the approach consisted of maximizing the difference between operation costs and income due to crop growth. Crop models were used in these algorithms to predict growth and development, and greenhouse models were necessary to establish the heat and mass transfer balances.

II.3.2 Temperature determination

Concurrent to the development of optimal control, interesting work has been done on temperature setpoint determination, with the objective of reducing the energy consumption while maintaining high production rates. Many authors have shown that crops respond to an average temperature more than to the absolute pattern (De Koning, 1988a; Langhans *et al.*, 1981; Krug and Liebig, 1980;). These authors applied different temperature regimes, keeping the same average temperature; the development rates and yields were found to be similar with tomato, lettuce, cucumber and chrysanthemum for all regimes tested, even when night temperatures were higher than during the day. This was referred as the "temperature integration effect". Some authors reported that the integration period, i.e., the period within which the temperature integral has to be re-established without affecting growth and development, could be as long as one week for certain crops such as tomato (De Koning, 1990; De Koning, 1988c; Liebig, 1988; Cockshull, 1988; Hurd and Graves, 1984). De Koning (1990; 1988c) concluded that this was true for mature tomato plants, even with large temperature amplitude, e.g., with temperatures fluctuating between 14°C and 26°C. Aikman and Picken (1989) explained that photosynthesis is driven by light, but the conversion of carbohydrate into plant structure is temperature dependent. For this reason, plants grown under different regimes but with a similar long term average temperature could have the same rates of development and growth.

Many authors have suggested that the temperature integration effect could be exploited to develop control strategies for determining greenhouse temperature setpoints in a way so as to reduce energy consumption without affecting crop production (Bailey and Chalabi, 1991; Hooper and Davis, 1988; Bailey, 1985; Miller *et al.*, 1985; Saffell and Marshall, 1983). This exploitation can be made possible with the use of computer, with which the temperature setpoints no longer need to remain fixed. Within this approach, a possible strategy is to let the inside temperature increase when solar radiation intensity is high enough so as to avoid heating, and to make a maximal use of this "free" energy. The idea is to increase the temperature setpoint for ventilation and compensate by lowering the temperature during other periods, such as during the following night (O'Flaherty, 1989). This allows for diminution in the energy consumption (O'Flaherty, 1989), extended periods of CO_2 fertilization (Cockshull, 1985; Miller et al., 1985) and the passive raising of temperature during midday for pollination, using solar energy (Hurd and Graves, 1984).

Another possible strategy using the temperature integrating effect is to switch heating from periods when heat loss factors are high to periods with milder conditions. Many authors have stressed the effect of the wind speed on greenhouse heat losses. These authors discussed the possible advantages of reducing the temperature when the wind speed is large and restoring the temperature integral by raising the temperature above normal when air speed is reduced (Aikman and Picken, 1989; Cockshull, 1985; Bailey, 1985). Reductions of energy consumption varying between 3 and 19% were reported when using wind-related setpoint regimes (Aikman and Picken, 1989; Bailey and Seginer, 1988; Bailey, 1985; Hurd and Graves, 1984). It was suggested that more benefits could be possible by adjusting the setpoints on the basis of weather forecasts (Bailey and Chalabi, 1991; Aikman and Picken, 1989; Bailey and Seginer, 1988).

Another approach to reducing energy consumption consists of delaying heating during the night for greenhouses equipped with a thermal screen (Bailey, 1988; Krug and Liebig,. 1980). The energy saved depends on the nature of the thermal screen, and the setpoints. Bailey (1988) calculated the potential annual savings with a transparent thermal screen and an aluminized one as being respectively 20.7 and 32.6%. By reducing the day heating setpoint by 2°C, he calculated additional savings of respectively 2.5 and 3.9%; with a reduction of 4°C, these values were increased to 4.8 and 7.4%. Using a similar approach within an optimal control strategy, and assuming a 50% reduction in heat loss coefficients with the thermal screen and a minimum temperature setpoint of 15°C, Bailey and Seginer (1988) calculated that potential savings up to 15% were possible.

II.3.3 Perspectives

The review on greenhouse climate control gives an indication of the different aspects that can be investigated. The determination of setpoints is challenging and judicious choices could permit a reduction of the resource requirements during a production process and an increase in the crop yields. Determining appropriate temperature setpoints offers interesting avenues in cold climates such as those prevailing in some regions of Canada, where energy costs constitute at least 25% of the total production costs (Lessard and Boudreau, 1992). This may be done advantageously by using the temperature integration effect. Furthermore, the use of weather forecasts may bring some additional benefits to climate control. There has not been any suggestion about this potential use in all papers consulted, with the exception of Bailey and Chalabi (1991) who mentioned the eventual consideration of using the forecasted wind speed in their control strategy. However, Fynn *et al.* (1991) used projected values of solar radiation, based on weather forecasts, to determine the crop requirements for nutrient control. Such an approach could be used for climate control, for example by starting the ventilation earlier during days when high solar radiation intensity is anticipated. It could also possibly permit the maintenance of high humidity levels at certain moments, if conditions are expected to be more favourable in the near future, e.g., when a sunny day is anticipated after a rainy day. Methods to integrate weather forecasts in control strategies must then be developed and tested to evaluate the potential benefits for greenhouses.

II.4 SIMULATION IN CONTROL

II.4.1 Simulation for control design

Many greenhouse control strategies and tactics can be conceived, where each one possesses a limited domain of applicability and is more appropriate for certain conditions. Physical experiments permit the determination of these ranges, but it is a laborious and costly task to test all control strategies within all possible contexts. Simulation constitutes a cheaper alternative in this regard, and has been used since the beginning of the computer era for this purpose. In the greenhouse control domain, this approach has been often adopted. For example, Lessard (1989) developed with simulation a control algorithm to fix temperature setpoints as a function of the season, the hour, the temperature and the intensity of solar radiation. Bailey (1988) used simulation to compare five control strategies for thermal screens. Bailey (1985) tested by simulation the variation of temperature setpoints as a function of the wind velocity. Others interfaced simulators with different digital controllers to compare their performance (Kozai *et al.*, 1985).

However, even with simulation, it is impossible to determine and test in advance all the possible combinations of crop characteristics and climatic conditions that can occur on a specific site. A possible solution to this problem is that the controller itself uses simulation to test different control scenarios and chooses the most appropriate one as a function of the occurring context. Studies related to the use of simulation during on-line greenhouse control were not found. Models have often been used in optimisation algorithms to determine optimal setpoints or for setpoint realization as within the anticipatory control approach, but not for simulation. This idea has, however, been suggested by researchers in other domains (Berger and Loparo, 1989; Lum and Heer, 1988) and the construction of simulation-based control systems for greenhouses must be investigated. This question will be approached from another perspective in the conceptual framework described in Chapter III, by relating it to conscious control, in the more global context of cognitive control.

II.4.2 Simulation and artificial intelligence

Simulation has been widely exploited for many decades, but its usage has mostly been limited to some specialists. In order to make this tool more accessible, many authors have coupled simulation and artificial intelligence tools. These technologies are complementary, each one contributing in its own fashion to solve a problem. Reddy (1987) discussed knowledge-based simulation systems, showing that expert systems (ES's) prescribe while simulation models describe. ES's can suggest actions and simulations will predict the consequences of these actions. In the same vein, Thangavadivelu and Colvin (1989) argued that mathematical models are used to predict system behavior, and ES's provide reasoning ability, helping in activities such as diagnosing, selecting alternatives and planning.

Many authors have constructed systems where an ES serves as an intelligent interface between simulation and users, helping in modelling, experimenting, analyzing and interpreting simulation results (O'Keefe, 1986). For instance, Moser (1986) mentioned that modelling and validation were causing problems for many people. Expert simulation systems can help a user to choose or construct a model from a base of modules, that can be assembled according to its objectives and to the characteristics of the modelled system (Deng and Jenkins, 1989; Murray and Sheppard, 1988; Reddy, 1987; O'Keefe, 1986; Shannon *et al.*, 1985). They can even generate an appropriate computer program (Haddock, 1987). Some authors showed that, by delimiting the required domain of scenarios and the necessary outputs to solve specific problems, ES's can choose the experiments to be done with models (Reddy, 1987; Shannon *et al.*, 1985). By doing a preliminary elimination of possible alternatives and reducing the number of simulations to be done, the decision processes can be accelerated (Broner *et al.*, 1990). Lastly, ES's can help in analyzing and interpreting simulation results, since results interpretation is habitually done by experts. ES's would allow for a more optimal use of simulation as a decision tool, while possessing the capacity to explain how they arrive at their conclusions (McClendon *et al.*, 1989; Reddy, 1987; Haddock, 1987; O'Keefe, 1986). Moser (1986) showed the advantage of using an ES to assist in decision making that utilizes simulation. Lal and Peart (1989) stated that simulations do not give optimal solutions and that an expert is required to analyze results and choose the best option.

The literature of the last decade is very rich in papers on techniques for coupling artificial intelligence with simulation. In general, researchers tried to find approaches to couple technologies issuing from the two fields so as to automate simulation analysis and to facilitate its usage by more people. This approach could be also valid to facilitate the implementation of simulation in control systems. For example, knowledge-based techniques could help a control system to do efficient simulation analysis by delimiting the investigation domain and taking into account the underlying assumptions and limitations of the models for results analysis and interpretation.

II.4.3 Use of neural models

Over the past thirty years, many greenhouse climate models have been constructed. Lacroix and Zanghi (1990) analyzed 33 of these. Generally, greenhouse models are composed of a certain number of mathematical relations establishing the energy balance for one or many components of the greenhouse such as interior air, cover, crop and soil. Most of these models have been deduced from basic physical concepts, while certain aspects have been left to some empirical adjustments, e.g., for natural ventilation and outside convection. These models need to be calibrated and validated on the specific site where they are used. Because of the number of variables involved in large models such as the multicomponent dynamic ones, the process of validation becomes a very complex and time-consuming task.

Since they possess some learning capacity, artificial neural networks constitute an interesting alternative to traditional modelling methods (Bullock *et al.*, 1992; Padgett and Roppel, 1992; Tang *et al.*, 1991; Wilderberger, 1990). Kok *et al.* (1991) reported results from experiments showing how well the behavior of a traditionally simulated greenhouse was mimicked by a neural network. Through a learning process, neural networks approximate the implicit relationships existing between sets of inputs and outputs. They, therefore, potentially offer an easy way to model some systems such as the greenhouse climate, and could be advantageous for constructing simulation-based controllers. Moreover, they allow automation of the modelling process, leading to the construction of *in situ* self-adjusting simulation machines. Also, since the same neural structure can be used to model different types of system, it is possible to implement it in hardware (neuro-chips), leading to very fast execution time for the simulations compared to that of traditional simulation processes (NeuralWare, 1991).

Artificial neural networks have been used in agriculture for many purposes such as in fermentation process control (Zhang *et al.*, 1992), crop phenology prediction (Elizondo *et al.*, 1992), apple classification (Ben-Hanan *et al.*, 1991), pH control of hydroponic solution (Morimoto and Hashimoto, 1991), meat quality control (Whittaker *et al.*, 1991), construction of an expert system for herbicide prescription (Zhuang and Engel, 1990), etc. However, the usage of neural networks for anticipatory control seems to have been investigated only in fields other than agriculture (e.g., McCullough, 1992). Such an application for greenhouse climate control would certainly be interesting for the reasons described above.

II.5 CONCLUSION

The literature review has shown that the complexity of greenhouse production systems has increased considerably over the last few decades and that the present trend is towards an integration of regulation and management activities into the same control structure. However, high level decision-making activities still have to be done by human beings who learn by experience how to cope with complex factors such as crop production requirements and socio-economic constraints. Thus, to increase control system autonomy, it is necessary to progressively withdraw the human manager from the control loops. This requires the construction of control systems that possess some cognitive functionalities to obtain an equivalent decision-making capacity. To orient the research necessary to conceive such complex systems, conceptual frameworks must be developed. They force an integration and a classification of already existing knowledge, and furnish common terminology and reference schemes for specific research activities.

The inclusion of simulation capacity in control systems constitutes a particularly interesting field of research. To make a contribution in this specific domain, a (simulated) greenhouse system comprised of a simple simulation-based controller was constructed and its behavior analyzed. The controller was responsible for determining the most appropriate temperature setpoints, with the help of simulations and consideration of weather forecasts. One of the tested control strategies consisted of shifting the heating to periods when heat loss factors were smaller, assuming that crops are good "temperature integrators". The model used for simulation by the controller was constructed with an artificial neural network. Before presenting the whole greenhouse system and the results that were obtained, the conceptual framework elaborated in this research will be first presented.

III. CONCEPTUAL FRAMEWORK

III.1 INTRODUCTION

In this chapter, the intent is to develop a conceptual framework that will help in the design of autonomous and enclosed agro-ecosystems (AEAE). Greenhouses, which form a specific instance of AEAE's, will be used as the reference system for the development of the concepts and their illustration. The conceptual framework to be developed should be complete and general enough to be adapted to different AEAE's. The framework should also be coherent and all its components integrated.

The conceptual framework will be presented in four parts, passing from general to specific considerations. In the first part, a general model of an enclosed agro-ecosystem will be presented. In the second part, the structure and the functions of the associated control system will be described. The third part will deal with the cognitive level of this control system. Finally, the fourth part will be specific to simulation-based cognitive control, or what is called "conscious control".

III.2 THE ENCLOSED AGRO-ECOSYSTEM: A GENERAL MODEL

III.2.1 Overall description

A general model of an enclosed agro-ecosystem was described in detail by Kok and Lacroix (1993) and the main concepts will be introduced in this section. This model is a representation of a system which is essentially production oriented. The human consumers are, therefore, not included in its internal components. Managers have also been considered as external agents, who inject some control information in the system. The only humans who are part of this system are the operators who work at the production level.

Figure III.1 shows the model for a greenhouse system with its inputs and outputs that can be either virtual or physical. The major components found inside the system are the production setting, the crop and two extrinsic controllers (Pavlovian and cognitive). Only some components need to be changed for this model to become a valid representation of any enclosed agro-ecosystem. For instance, to obtain a model of an aquacultural production system, the crop could be replaced by a fish population and the greenhouse envelope by a pond.

III.2.2 Physical inputs

All external agents, information, signals and fluxes that directly affect the behavior of the system are considered as inputs. They are separated into physical and virtual inputs. The physical inputs are those that influence the material aspects of the system. The virtual inputs are essentially information-based and can be treated only by some of the internal components.

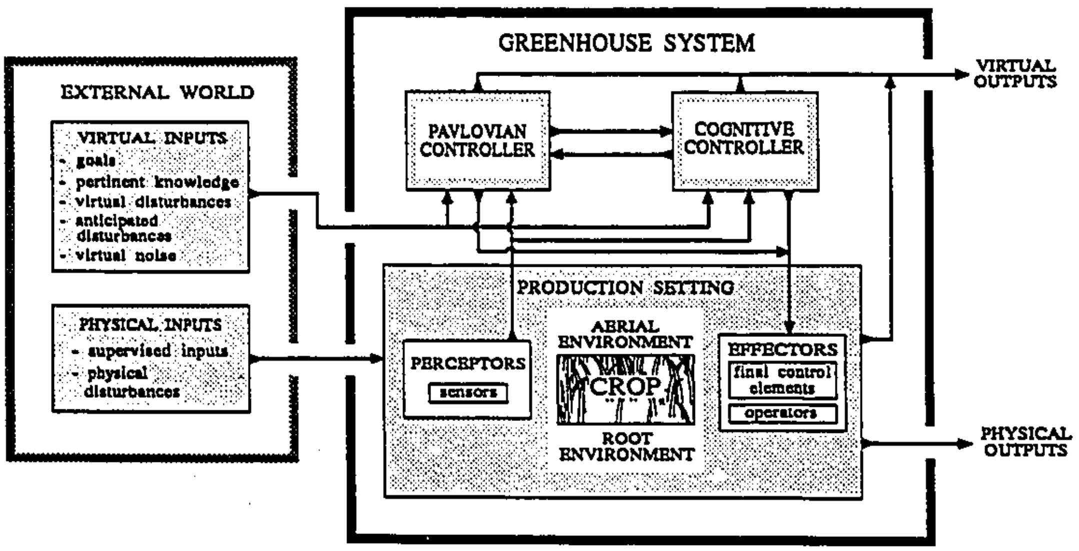

Figure III.1 Model of an enclosed agro-ecosystem.

Physical inputs can be classified as either supervised inputs or disturbances. Supervised inputs include all physical entities that are required in the production processes and that are deliberately inserted into the system (e.g., propane gas, liquid CO_2, seeds, nutrients). The disturbances are defined as all entities affecting the system without being deliberately imported (e.g., solar radiation, external air, insects).

The flow rates of the supervised inputs are determined by internal system components according to some control objectives. Supervised inputs are often injected into the system to compensate for some undesirable effects of the disturbances. They include all goods that are exchanged against money in a market context. The goods can be fixed (e.g., structural components and equipment) or variable (e.g., sources of energy, seeds, nutrients, water, carbon dioxide and labor). Their availability can be influenced by external agents. For example, a power failure will affect the supply and consumption of electricity, and underground pollution by pesticide may affect the availability of irrigation water.

Disturbances are all other physical inputs affecting the system. Their presence is not determined by the needs of the system. They are simply part of the surrounding environment. In some cases, their input flow rate can be controlled from inside the system (e.g., the unfolding of a shading curtain will reduce solar radiation intensity). In a market context, disturbances will habitually not be exchanged for money. In agricultural production, the external climate forms an important set in this category. Another important group is composed of biological agents such as microbes, fungi, insects and other animals. Certain disturbances are desired, such as solar energy for a greenhouse, while other are undesirable because they decrease system productivity.

III.2.3 Virtual inputs

The virtual inputs are information-based and can only be treated by humans or advanced electronic devices such as computers. They directly affect the Pavlovian and cognitive processes in the extrinsic controllers. Virtual inputs are divided into five categories: 1) goals, 2) pertinent knowledge, 3) virtual disturbances, 4) anticipated disturbances and 5) virtual noise. Goals have a determining influence on the general organization of the production system and on all the control decisions. Examples of goals are the quality of the food to be produced, the species to be grown and the rate of production. Depending on the degree of autonomy a system is to have, more or less specific goals will be supplied to it from external managers.

To obtain and maintain high production levels, farmers must make important decisions at different control levels, varying from management to process regulation. These decisions require a large amount of information and knowledge. An autonomous system requires similar information, which would include among others, management practices, the required growing conditions for specific species and the availability of some supervised physical inputs. A large part of the pertinent knowledge will come from sources such as growers' associations, research institutions and government agencies. If the system is not totally autonomous, it will also continue to include input from growers in its decision processes.

Part of the external virtual variables can be considered as disturbances if they affect the system behavior without being intentionally imported. Many economic, political and social factors can be included in this category. Characteristics of the market, consumer's habits, insurances, governmental subsidies, taxes, inflation, prices of the physical inputs and interest rates can all be considered as virtual disturbances for a greenhouse enterprise. A sophisticated system must obviously consider those variables to function optimally.

Knowledge of future disturbances helps in the making of optimal control decisions. The future can never be known perfectly, but it is, however, possible to obtain an estimate of what will happen since the probability of the occurrence of certain events is higher than for others. Anticipations based on prediction models and consideration of present and past states of variables can be used. The results of these calculations are called anticipated disturbances. Anticipated values for both virtual and physical disturbances can be obtained. Examples for virtual inputs are projections of prices and market characteristics, and other economic forecasts habitually used by enterprises. Examples for physical disturbances are weather forecasts and predictions for the increase of the population of certain insects, which can be of help to devise optimal management practices.

In each of the above categories, information flows into the system and affects the decision-making processes. It is also probable that false information will come in and that may lead to sub-optimal decisions. The term "virtual noise" is used to categorize such information that includes, for example, erroneous knowledge and wrong predictions. The way a system can deal with virtual noise will certainly affect its productivity levels and even its survival.

III.2.4 Outputs

System outputs have also been separated into virtual and physical categories. An agro-ecosystem's products form the most important physical output. Physical outputs also include any by-products such as wastewater or parts of the plants not fit for consumption. Virtual outputs consist of any kind of information for external agents, such as service requests or data about production levels. Virtual outputs incorporate information about the state of the internal components, for use by the external managers.

III.2.5 Internal components

The internal components of the production system are divided into three groups: 1) the production objects, 2) the production setting and 3) the extrinsic controllers. In an agricultural enterprise, the production objects may be either plants or animals, usually processed into food, feed or fibre. Especially in greenhouses, ornamental species can also be produced. The configuration of the production setting depends largely on the choice of the production object. For non-autonomous systems, this choice is a goal decided upon by an external agent.

III.2.5.1 Production setting

The production setting is composed of everything that directly affects the growth and yield of the production objects, such as operators, climatic environment, equipment and other structural components. The production setting is subdivided into four categories: a) physical structure, b) the encompassment of the production object, c) perceptors and d) effectors. The physical structure contains the enclosed agro-ecosystem and defines its physical boundaries. It incorporates many components such as equipment and the service subsystems. Tables, benches and moving platforms are examples of equipment installed in greenhouses. The service subsystems supply consumable inputs such as electric power, water and carbon dioxide. A large number of control mechanisms are inherent in this production setting group, e.g., to regulate pressure or limit movement.

For a plant culture, the encompassment of the production object is comprised of the climatic and root environments. Similarly, in an aquacultural system, water would form the major part of the encompassment. The climatic environment of a crop includes the air surrounding it, together with the impinging radiation. In greenhouse production, air is often characterized by its temperature, velocity, humidity and carbon dioxide concentration. Oxygen, ethylene, pesticides and other contaminants could be added to this list. The radiation impinging on crops is often divided into two sets: visible and far infrared. The radiation beneficial for biomass production is mostly in the visible part of the spectrum, whereas infrared radiation is often used to heat the production objects. The production object itself will also influence the climate. For example, crops can influence the air moisture content by transpiration and plants can absorb a large amount of carbon dioxide during photosynthesis, thereby affecting its concentration in the air. The root environment includes the substrate to which the roots are exposed. In greenhouse production, the root environment is often characterized by its temperature, pH and concentration of nutrients, and, in the case of solid substrates, by its moisture content.

The perceptors that are part of the production setting are sensors used to collect information from the internal components of the system and from the external world. Many sensors are already used in greenhouses for measuring temperature, radiation, humidity, gas composition, wind velocity, pressure, etc. Signals from the sensors can be used directly by low-level control components integrated in the physical setting, or sent to the extrinsic controllers for analysis, leading to the formulation of higher-level control decision. Other perceptors that deal essentially with information, reside in the extrinsic controllers. Their role is to select pertinent information from the external world. In the future, robots may be used as perceptors with olfactory and tactile sensors and artificial vision.

The effectors are the components that carry out control decisions within the production setting. They include the final control elements and the operators. The final control elements are habitually used for the regulation of the production object's encompassment, the goal being to achieve and maintain some desired state. For example, in greenhouses, effectors influencing the aerial environment of the crops can include radiant or convection heating systems, artificial lighting, carbon dioxide injectors, heat pumps, ventilation systems, cooling pads, water sprayers, heat exchangers, solar collectors and thermal screens. The effectors affecting the root environment will be mostly the heating, irrigation and fertilizing systems.

Operators, constituting the second type of effector, can modify some aspects of the production setting or the production objects themselves. These actions require analysis of information and decision-making. The operators are therefore intelligent agents, which possess their own control system, but are subjugated to higher control so to assure the achievement of the general goals of the system. The operators can be either human or robotic. In greenhouses, the activities performed by operators include, among others, transplanting, pollination, harvesting and cleaning, as well as the displacement, modification and guidance of equipment.

III.2.5.2 Extrinsic controllers

The term "extrinsic control" is used to distinguish it from "intrinsic control" which is inherent to the nature and structure of physical objects. In this framework, extrinsic control is performed with computers and the task has been divided between two types of machine: Pavlovian and cognitive. Each machine can, in turn, be distributed over many processors. The Pavlovian controller is dedicated mostly to regulation, i.e., it has to make the appropriate control decisions to assure the realization of setpoints. It is also responsible for the collection and compilation of measured values. Globally, its role could be compared to that played by commercial digital controllers presently installed in greenhouses. It operates with reflex actions, without analyzing the consequences of its decisions. Its control activities are determined largely by control routines and parameters dictated by higher authority. In the greenhouse industry, control strategies are encoded in software provided by companies that sell control devices, and setpoints are set manually by growers. However, if a system is to be autonomous, this has to be done by an artificially intelligent entity. In the present framework, this is done by the cognitive machine.

The cognitive controller possesses functionalities generally associated with intelligence such as learning, anticipation and planning, and is capable of reasoning and decision-making. This confers on it the capacity to replace the managers, in making some decisions that require complex reasoning processes. It, therefore, allows the withdrawal of the human from the control loop and an increase of system autonomy. All control processes that are unusual or not routine are executed by the cognitive machine. This controller is responsible for continuously deciding and transmitting to the Pavlovian controller the control actions and the associated parameters that are the most appropriate to the external context (e.g., meteorological conditions) and to the crop characteristics (e.g., cultivars, growth stage). The differences between the Pavlovian and cognitive control levels are more extensively discussed in Section III.3.4.3.

III.3 THE CONTROL SYSTEM

III.3.1 The importance of control

As stated in the introduction of the thesis, there will be a limited number of species within an enclosed agro-ecosystem. In this context, many functions that are habitually assumed by other species in natural environments must be compensated for by artificial mechanisms. For example, for fruits to be produced, pollination must be done by operators. Also, many abiotic factors necessary for growth are absent in enclosed agro-ecosystems, and equipment must be added to supply them (e.g., irrigation system). Since many physical control mechanisms and the inherent buffering capacities habitually found in nature are not existent in such a system, instability can easily occur. The only way to maintain the stability is with a complete and fully integrated control system. As has been shown through the evolution of agriculture, for control to be efficient, part of the control system must possess some cognizance of the controlled system and must be capable of intelligent decision-making. It must, therefore, include both physical and virtual control mechanisms. Depending on the nature of the controlled system and the level of knowledge about it, there could be a large presence of virtual mechanisms in the complete system.

Because of the importance of the control system for the viability of an enclosed agro-ecosystem and the optimization of its processes, it is necessary for emphasis to be put on the control aspect in the elaboration of a conceptual framework. While being kept simple, the framework should allow for the construction of complex systems. It should also be conceived in a way that will integrate the virtual control mechanisms. The part of the framework related to the control system was presented in detail by Kok and Lacroix (1993) and some major aspects are reported in the following sections.

III.3.2 Goal orientation

In agriculture, control is carried out primarily for the achievement of some specific production goals, while maintaining the stability and the sustainability of the total production system. In this instance, control includes all traditional activities found in agricultural production enterprises and ranges from management to regulation. It implies many decision-making and decision implementation activities (Kok and Desmarais, 1987). Antsaklis (1990) gave a similar definition of control for industrial processes. An illustration of hierarchical control by Doebelin (1985) showed comparable activities.

A basic aspect of the framework is, therefore, that the complete control system functions to achieve a set of goals. Main goals can be decomposed into subgoals, which can then be further subdivided, etc. The relationships between goals can be described by a goal tree, as illustrated in Figure III.2. The subgoals directly associated with one goal are called child goals. The achievement of a specific goal requires the attainment of one or more child goals.

In the design of a control system, creation of the goal tree should be one of the first steps, since it will determine its general structure. The development of the goal tree helps to place into perspective all the interactions between the components and to construct a coherent system. The simple goal tree illustrated in Figure III.2 is for a greenhouse operating under the constraints of a market economy and, in this example, the main goal is profit maximization. However, in different circumstances, e.g., in a space station, the goal might be to maintain a specific rate of food production. For profit maximization, two subgoals have been defined in Figure III.2: 1) minimization of energy costs and 2) maximization of biomass production. These goals are then further subdivided into goals for optimal control of temperature, humidity and CO_2, etc. It should be noted that other subgoals could have been defined at any level and that a complete tree can rapidly become very complex. However, at the beginning of a design project, it is necessary that the decomposition be done in as much detail as possible to assure a coherent control structure.

III.3.3 The network of control mechanisms

In an enclosed agro-ecosystem, as in many other production units, it is difficult to establish a clear barrier between the controlling system and the controlled system. Some components could be classified in both categories. For example, some control mechanisms are integrated in the physical system, e.g., a gas pressure regulator. Other mechanisms, such as those habitually implemented in computers, are clearly part of the controlling system. In the framework, all mechanisms that are deliberately added to achieve the goals of the enclosed agro-ecosystem are considered as part of the control system.

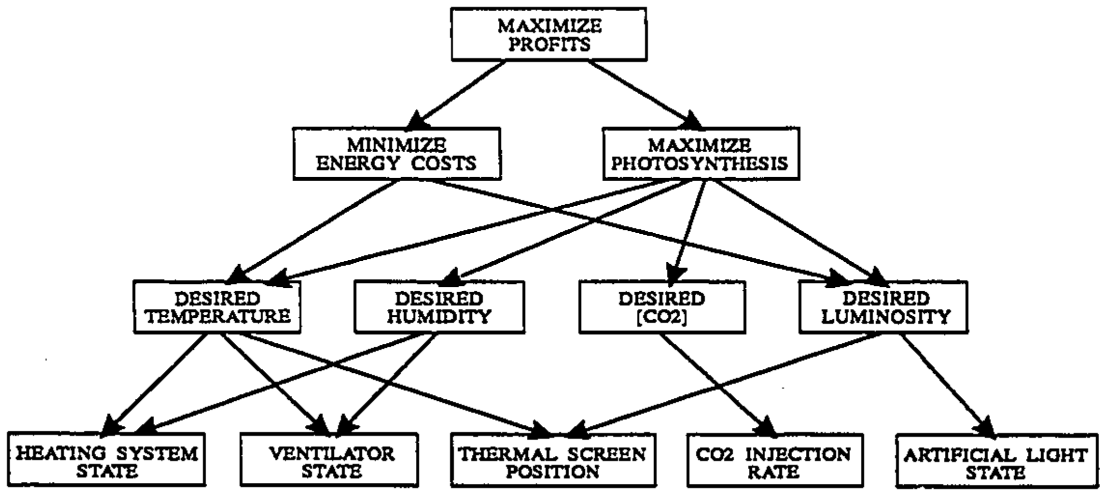

Figure III.2 A goal tree.

The control system is, therefore, conceived as a network of interrelated mechanisms functioning together to achieve major production goals (Figure III.3). The mechanisms work in parallel and they input and output to a common pool of variables. This does not imply that each mechanism has access to all variables; rather it will usually have the capacity to access only a limited number of variables. Each mechanism is responsible for making the appropriate decisions to reach a certain subgoal, either by dictating the realization of final control actions or by the activation of other control mechanisms. A final control action may consist of either an instruction to an effector, such as for the activation of a final control element, or the execution of a task by an operator. It might also be the execution of a specific computing activity by an extrinsic controller, e.g., indexing of a data base.

In the framework, the achievement of a goal corresponds to the attainment of a target value for a controlled variable, a controlled variable being associated with each control mechanism. A control mechanism decides what actions should be taken to attain the desired value. The actions consist of manipulating other variables that must themselves reach certain target values for the parent goal to be satisfied. The realizations of the

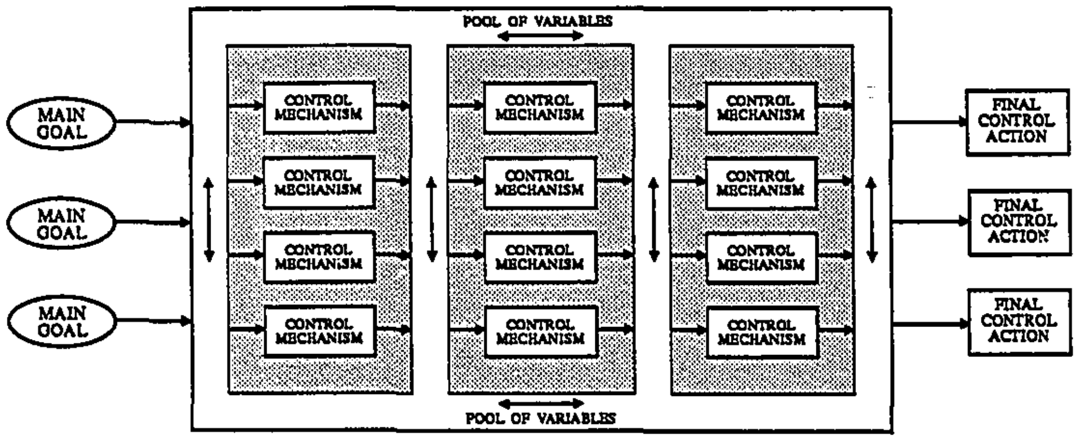

Figure III.3 The control mechanism network.

desired values for the manipulated variables are then subgoals. In this case, the manipulated variables, themselves, temporarily become controlled variables and must resort to their own child mechanisms for the target values to be achieved. At the end of the chain of control mechanisms, there will be one mechanism that attempts to carry out the required final control action.

The basic unit of the control structure is the "control mechanism" and it was attempted to develop a general model for this. The intent was to make the model sufficiently versatile to accommodate control at any level, for both virtual and physical mechanisms. Also, the model must allow for full system functionality. This model is illustrated in Figure III.4. It is characterized by a series of attributes and functions. These attributes include goal type (prevention, assurance and performance), activity class (regulation, operation and management) and implementation level (physical, instinctive, Pavlovian and cognitive). They are discussed in the next section (Section III.3.4). The mechanism functions include decision-making, conflict treatment and activation and are discussed in Section III.3.5.

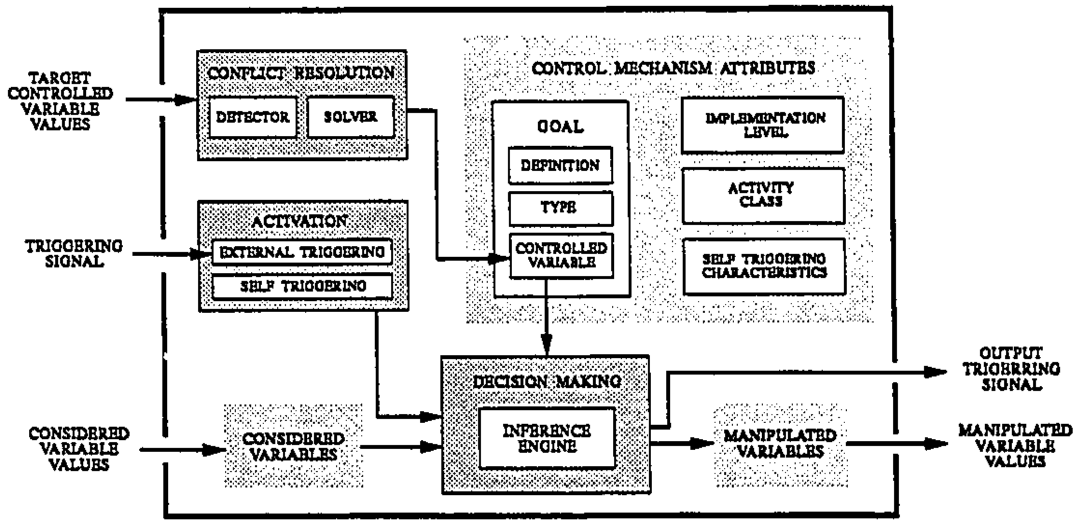

Figure III.4 The general control mechanism model.

III.3.4 Control mechanism attributes

III.3.4.1 Goal types

In the framework, three goal types are recognized: 1) prevention, 2) assurance and 3) performance. Some actions or some states must be prevented (e.g., sub-zero temperatures must not occur to avoid crop destruction) and others must be assured (e.g., irrigation water must be supplied to plants in sufficient quantity). It should be noted that prevention and assurance are complementary: preventing a situation from happening is similar to assuring that it does not occur. However, in a design project, it may be preferable to conceptually consider separately what must be prevented, and what should absolutely happen. The last goal type, performance, is wider in scope. All goals that must be achieved as much as possible are of the performance type.

The goal type will have some implications on its role in an overall control system. Prevention and assurance goals must absolutely be achieved and a highest priority will often be associated to the control mechanisms associated with these goals. They impose some constraints on the realization of other goals. If prevention or assurance goals are not achieved, the performance of the controlled system behavior may be severely compromised. In severe cases, the destruction of the whole system or of one of its components might result. The control system will do its best to achieve the realization of the performance type goals, but under the constraints imposed by the prevention and assurance goals. Achievement of the performance type goals is not necessary for the survival of the system, but is required for its optimal behavior and the achievement of the main goals.

It is important to determine the type of each goal in the preliminary phases of a design project, since it can affect other considerations such as the implementation level of the associated mechanisms. For example, it could be decided that a prevention type mechanism possessing a high priority level (e.g., prevention of low temperatures) be implemented with the help of a physical device (e.g., a thermostat), instead of relying on a virtual mechanism in a computer.

III.3.4.2 Activity classes

In the framework, three activity classes described by Kok and Desmarais (1987) and Gauthier and Kok (1989) were retained: regulation, operation and management. Generally, the complexity of the decision-making processes and the quantity of information treated increases when passing from regulation to management, whereas the frequency of control actions decreases. Also, the time interval considered in the decision-making processes of mechanisms usually increases. Management is oriented towards the formulation of long-term strategies. Operation is concerned with hourly and daily control decisions and implementation. Regulation is related to minute-to-minute control of variables like temperature and CO_2 concentration. Similar activity levels have been recognized by many researchers in the field of greenhouse control, and have been integrated into hierarchical frameworks (Udink ten Cate *et al.*, 1978).

Management activities include the definition of medium and long term strategies. They correspond to production planning, investment, infrastructure development and all other management decisions habitually made by farmers. The application of long term strategies on a daily basis forms the operational level of activities. In greenhouses, they correspond mostly to actions performed by operators, such as crop transplanting, pollination, harvesting, and all other "manual" operations. The application of long term strategies might also correspond to medium term decisions at the operational level. For example, static temperature setpoints are determined at the management level, after taking into consideration energy costs, average meteorological conditions, crop species, etc. Dynamic setpoints are defined at the operational level, with the objective of optimizing the short term processes as a function of the actual meteorological conditions, crop growth stage, etc. Regulation is related to classical control activities, such as feedback, feedforward and adaptive control. Regulation is mostly concerned with the realization of some predetermined setpoints. Mechanical and analog devices have long been used to regulate processes with P, PI and PID algorithms.

III.3.4.3 Implementation level

In the framework, four major structural levels are recognized for the implementation of the control mechanisms: physical, instinctive, Pavlovian and cognitive. This is done to imitate the way control is perceived to be organized in biological entities (Kok and Desmarais, 1985). This is similar in many ways to the distributed control approach described by several authors (Hoshi and Kozai, 1984; Bakker *et al.*, 1988; Roy and Jones, 1988). There are no absolute boundaries among levels. Instead, a gradual variation of many factors occurs from one level to the other. When passing from the physical to the cognitive level, the quantity of information treated by the mechanisms and the complexity of the treatment increases. Also, the implementation of the decisions generally shifts from physical to virtual actions.

Control mechanisms at the physical and instinctive levels are intrinsic to the production setting. They react only to physical inputs and the implementation of their decisions involves physical actions. Physical level mechanisms often lie in the fuzzy area between the controlling and the controlled system (e.g., circuit breakers, pressure regulators). Pavlovian and cognitive control mechanisms are extrinsic, i.e., they are implemented on machines that are separate from the production setting. These mechanisms are exclusively of a virtual nature and form the extrinsic controllers. The execution of their decisions can result in either a virtual or a physical action. Some extrinsic mechanisms are responsible exclusively for the management of the controller activities and their storage capacity. The virtual inputs of the system (e.g., disturbance anticipation) can be treated only by the extrinsic controllers.

Physical control mechanisms are integrated into the components of the production setting. The most basic mechanisms included in this category would be, for example, a large thermal mass added to passively control temperature, or a self-darkening cover material. More active mechanisms would include pressure regulators, circuit breakers and thermostats. At the physical level, the sensing, decision making and decision implementation are completely intermingled in the control mechanisms. Physical control is fully automatic and the decisions are impossible to thwart by mechanisms at other implementation levels.

At the instinctive level, the control mechanisms are still part of the production setting, but data sensing, information treatment and decision implementation processes are more distinct than for the physical level. Decisions are very difficult, if not impossible, to override by mechanisms at the extrinsic levels. Also, the consequences of the decisions are not analyzed beforehand. Parts of the alarm system are examples of instinctive mechanisms.

The goals of intrinsic control mechanisms are often of the prevention or assurance type. These mechanisms are associated with mechanical or electronic devices added to the controlled system to increase the reliability in the whole system. Other mechanisms at this level are associated with actuators and final control actions, and are of the performance goal type. Most intrinsic control mechanisms belong to the regulation activity class.

Pavlovian control is based on strategies and tactics that are learned. Its mechanisms carry out routine control actions in a reflex manner. Pavlovian control mechanisms vary greatly in their role and nature and, together, form the Pavlovian machine. Some mechanisms will be directly related to physical actions that must be carried out. This type of control involves data acquisition and treatment and can result in the manipulation of final control elements. Most of the microprocessor-based controllers presently installed in greenhouses perform this type of control, in which routines and parameters are furnished by humans. Other Pavlovian mechanisms will be more complex in nature, and will possess some learning capacity. These mechanisms will behave at a more abstract level and they will be able to recognize patterns in a large amount of inputs. Some will dictate parameters to mechanisms at lower levels. In Pavlovian control, the consequences of the actions emerging from decision-making processes are not analyzed beforehand. Pavlovian control mechanisms can be associated with the reflex mechanisms explained by Handelman *et al.* (1990). These mechanisms are automatic and require little or no reasoning. Their composition is determined by outside agents such as system designers and external managers, by the cognitive machine, or through learning processes. In this framework, the cognitive machine can download both control routines and control parameters to the Pavlovian mechanisms.

At the cognitive level, many functions related to human intelligence are imitated. The autonomy of a system will depend largely upon such cognitive mechanisms since they are designed to partially or wholly replace human managers. Cognitive mechanisms execute activities such as reasoning, pattern recognition and learning. Past and present events are continuously analyzed, interpreted and organized so that principles and rules can be extracted, and models can be created. Past conditions are investigated to validate and refine the models and control strategies. Models of the enclosed system and of the exterior world are intensively used to test different alternatives and choose an optimal course of action. In this sense, conscious control is an important part of the cognitive control activities. Control strategies are elaborated at the cognitive level and implemented, with the appropriate parameters, at the Pavlovian level. Cognitive decisions can also result in the activation of some final control elements. The cognitive control mechanisms form together the cognitive machine, which will be presented in more detail in Section III.4.

In Figure III.5, a set of control mechanisms and their implementation levels are shown. For example, in the upper left corner, a cognitive control mechanism is responsible for finding the optimal static temperature setpoint, according to market and economic conditions, cultivars' requirements and the average climate of the growth season. The

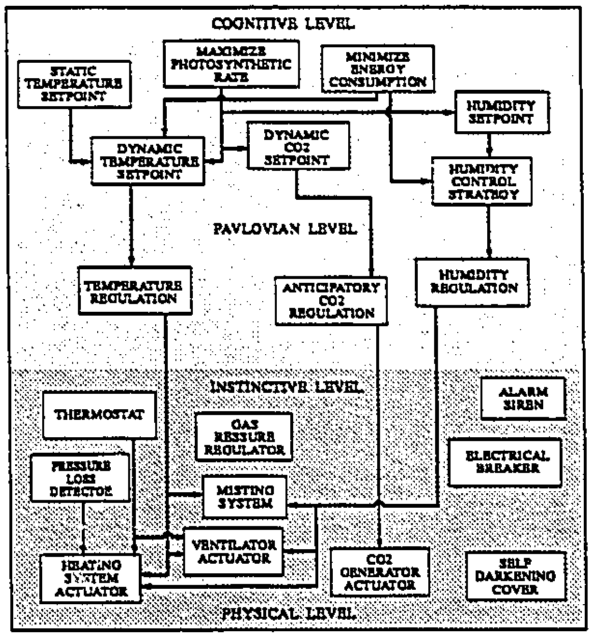

Figure III.5 Implementation levels of mechanisms.

realization of this static setpoint is the responsibility of another cognitive control mechanism, which continuously finds the optimal dynamic setpoint by considering the average meteorological conditions over a short period (e.g., hours) and the stage of crop development. A control mechanism installed at the Pavlovian level achieves the realization of these optimal setpoints, taking into account the actual external climate and the state of many final control elements (e.g., carbon dioxide enrichment vs. ventilation). The end result can be the activation of a ventilator, a misting system or a heating system under the control of mechanisms implemented at the intrinsic levels, i.e., at the instinctive and physical levels.

It should be noted that similar control effects can be achieved with different sets of mechanisms installed at various implementation levels. For example, on a greenhouse, a self-shading cover might be used to regulate solar radiation intensity, but this could also be done with a curtain. Each approach has advantages and disadvantages. More extensive decision-making is required to decide on when to open or close the curtain, but the use of a curtain gives more flexibility to the system. The choice of the implementation level will certainly be influenced by many factors such as cost, availability of materials, technical feasibility and system security. In the design of a control system, a balance must be maintained between the mechanisms at the four levels of implementation. The mechanisms must be arranged to function in parallel, not hierarchically. Mechanisms at the intrinsic levels are essential for the survival of the whole system and must not be overridden by the mechanisms at the extrinsic level. The mechanisms at the extrinsic levels potentially increase the optimal performance of the system. However, mistakes and poor reasoning at the extrinsic level could lead to decisions that, if carried out, would cause severe problems for the system.

III.3.5 Control mechanism functions

In the framework, the role of a control mechanism is to make decisions about the actions that should be taken for the required value of its associated controlled variable to be attained. The decision-making process is, therefore, a central activity of a control mechanism, and consists of determining the appropriate values for the variables that are to be manipulated. The inference engine is the unit responsible for decision-making in each mechanism (Figure III.4). The decision-making process can be very simple; it can even be completely intermingled with the implementation process, such as in a thermostat or a pressure regulator. At the other extreme, it can be extensive, involving long reasoning sequences, simulations, planning and other complex activities. In many cases, there will be many variables that will be considered in the decision-making process. These can be virtual inputs such as weather predictions, the conclusions of a decision-process by another control mechanism or sensor measurements. Note that the classification of the variables into "controlled", "manipulated" and "considered" is from the perspective of the mechanism under consideration only; the manipulated variable of one mechanism may be a controlled or considered variable of other mechanisms.

When more than one mechanism attempts to manipulate the same variable, a conflict occurs if they want to manipulate it differently. Since the final control elements and the operators can have only one state at a time, inconsistent instructions to them must be reconciled before implementation. Conflict resolution is, therefore, an important function of a control mechanism. At the extrinsic level, conflicts can be handled in a manner approaching that used by human, e.g., with a treatment based on fuzzy logic. When designing a control structure, it is important to detect which controlled variables are affected by the manipulation of variables from other control mechanisms. This helps in finding potential sources of conflict.

A control mechanism has to be activated for a decision-making process to be initiated. This can be done from inside the mechanism or from outside. The triggering can be done via several means: by an internal timer, by a "watchdog" mechanism which checks for a particular situation to occur, or by order from other control mechanisms.

III.4 FUNCTIONS AND STRUCTURE OF THE COGNITIVE CONTROLLER

III.4.1 Cognitive functions

Until lately, most control functions of modern controllers had been implemented at a primitive Pavlovian level. Presently, research for the implementation of mechanisms at the cognitive level is being pursued in many fields. In greenhouse research, authors like Hoshi and Kozai (1984), Takakura *et al.* (1984), Kozai (1985), Harazano *et al.* (1988), Kurata (1988) and Gauthier and Guay (1990) have worked on designing various components of knowledge-based control systems. Lum and Heer (1988) and Ihara *et al.* (1989) have also developed conceptual structures to design control systems based on artificial intelligence technologies, for use in autonomous systems in outer space. Wright *et al.* (1986) combined conventional and expert system controllers for military and advanced industrial applications. Berger and Loparo (1989) have presented a framework for the design of hierarchical intelligent control. Functions described in these papers include reasoning, decision-making, learning, information decoding and organizing, pattern recognition and predicting. In this framework, such functions are called "cognitive functions" and are performed by the cognitive controller. Figure III.6 contains a more complete list of cognitive functions. The performance of these functions would allow the controller to replace human managers and to act at the executive decision-making level for the agro-ecosystem. This would give the system a certain amount of autonomy.

In the framework described in this thesis, cognitive functions are performed by subsets of control mechanisms. These functions can be carried out more or less intelligently, and more or less "consciously". The degree of intelligence depends on the complexity of the information that is treated and on the depth of the treatment. The degree of "consciousness" or "self-awareness" is a function of the extent to which the mechanisms refer to models of the agro-ecosystem and its environment in their decision-making processes.

COGNITIVE FUNCTIONS

Observation Learning Memorization and recall
Pattern recognition Problem-formulating and solving
Rule following Model composition, testing and correction
Model formulation Extraction of rules and principles
Analysis Simulation and prediction
Abstraction Imagination Expression

Figure III.6 List of mind functions.

Concepts related to the structure and functioning of the cognitive controller have been described in detail by Lacroix and Kok (1991a). In this section, the elements concerning the proposition for a structure are reported. This proposition was developed as a conceptual basis for future development of complex cognitive controller; for this thesis, it was not the intent to implement such a structure in its entirety.

III.4.2 Proposition for a structure

III.4.2.1 Memory and control activities

The cognitive controller is formed by a network of control mechanisms that act in parallel and that access a common pool of variables called "memory", as illustrated in Figure III.7. A wide definition has been given to the term "variable". A variable can represent a very simple entity, such as the state of an actuator. It can also represent an agglomerate entity formed by complex composites of knowledge such as rules, models and meta-knowledge. These variables are used by the control mechanisms in their decision-making processes and are under the control of the inference engine, which is responsible for rule chaining, simulation and any other information treatment required to reach some decision. The source of the memory content includes any information collected by perceptors (e.g., data coming from the measurement network and knowledge imported from the external world) and any new knowledge synthesized by the controller itself, by decoding and reorganizing all available information. All facts, such as measured data, are included in data bases. Knowledge bases can store more abstract information and meta-knowledge in many forms such as rules, frames or models.

The cognitive control mechanisms can be segregated in many ways according to various characteristics such as the complexity of their reasoning processes and the type of cognitive functions they participate in. Also, control mechanisms act together to perform control activities to meet specific responsibilities and they can be classified according to

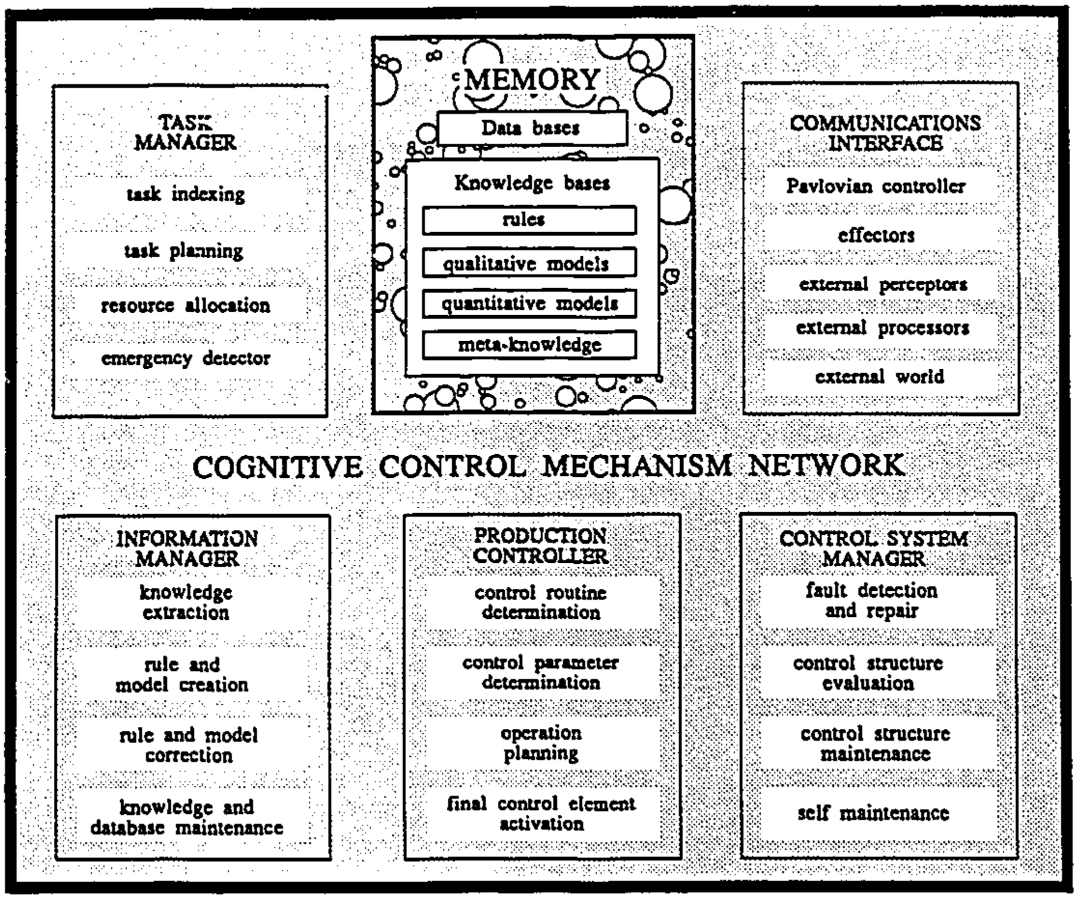

Figure III.7 Structure of a cognitive controller.

the type of activity they contribute to. Five major "responsibility groups" have been set up: 1) task management, 2) information management, 3) production control, 4) control system management and 5) communications. In Figure III.7, the cognitive controller has been structured according to these main groups. Each group includes mechanisms that differ greatly in complexity. For example, some of the mechanisms in the task manager will behave "consciously", while others will not. Also, the mechanisms of a specific group will participate in the performance of different cognitive functions. For example, some mechanisms will perform pattern recognition, while others will do problem-solving.

No clear distinction exists between the different responsibility groups and in a real cognitive controller they will overlap considerably. Here, the number of groups is limited, and others might have been conceived. Certain responsibilities are specific to the management of the cognitive controller itself, while others are more related to the functioning of the overall controlling and controlled systems. The task manager acts as the principal coordinator of all cognitive activities. The information manager is responsible for the organization and the maintenance of the memory, and for the processing of any information entering the system. The production controller consists of a set of mechanisms dedicated to the management, operation and regulation of the production processes. The control system manager is responsible for fault detection and for continuously evaluating the performance of the control system. Thus, it must also suggest modifications to the structure when required. Finally, a component interfaces with external agents such as effectors, Pavlovian controller and external data bases.

III.4.2.2 The task manager

The task manager is responsible for planning the activities of the cognitive controller according to what must be done, the time and resources available and the task priorities. This is important in time-critical systems where computational abilities are limited and the computing resources must be allocated optimally. To do so, the task manager indexes all required computing tasks, and it associates with each one a priority level and a maximum time delay for their execution. It also estimates the resources (time and computing) required to accomplish each task. This last aspect is particularly complex because, for each decision process, the controller must reason about the number of options that must be investigated and the depth of the deliberation. Such meta-reasoning can be done in parallel with the decision process of concern, by comparing the estimated costs associated to the pursuit of the deliberation (e.g., consequences of delays that are imposed, additional computational resource requirements) with the estimated benefits of a deeper investigation (Dean, 1991). The cognitive controller must therefore possess some cognizance about the computing capacities it can access. This can be the machine(s) on which it resides and any other external processors. The cognizance might consist of models of the complete physical support available for computations, and models of the resident software capacities. In parallel to this meta-reasoning, a priority level must be assigned to each task; if emergencies occur somewhere in the controlled or the controlling systems, a new attribution of the resources might be required, which will affect the previously established schedules.

III.4.2.3 The information manager

The information manager consists of all mechanisms that are directly assigned to the management of the memory pool. Its activities include the treatment of information arriving from the external environment (e.g., pattern detection in the information flows) and the creation of new knowledge. The information manager checks the contents of the data and knowledge bases for pertinence, coherence and accuracy. It also compiles and indexes this content. It also does a regular garbage collection to prevent cluttering of the memory. It studies the content of the memory to extract new rules and to detect patterns, so as to create new models. As a complement, the information manager is also responsible for the calibration, adjustment and adaptation of the models already contained in memory and participates in the revision of the control strategies.

III.4.2.4 The production controller

The production controller is directly responsible for the management, operation and regulation of the production processes. It decides which control routines and parameters are to be transmitted to the Pavlovian controller. It participates in the planning of the work done by the operators and in the activation of some final control elements. For example, a cognitive control mechanism of the production control group can be responsible for finding the optimal static setpoint for a certain variable, with respect to market and economic conditions, the type of cultivar and the average climate of the growth season. The realization of this static setpoint can be under the responsibility of another cognitive control mechanism, which continuously finds the optimal dynamic setpoints, considering the average meteorological conditions over a short period (e.g., hours) and the crop development stage. One or more control mechanisms installed on both Pavlovian and cognitive machines will share the responsibility of achieving the realization of these optimal setpoints, considering the actual external climate and the states of many of the final control elements (e.g., carbon dioxide enrichment vs. ventilation).

III.4.2.5 The control system manager

The activities of the production controller partially overlap the activities of the control system manager, which is responsible for a continuous evaluation of the overall controlling system and for the detection of faults and hardware failures. This aspect demands considerable attention when intelligent autonomous control systems are designed (Antsaklis *et al.*, 1991). Another aspect consists of the reevaluation of the control structure and suggestions of modifications, such as the transfer of an activity from conscious to reflex level (e.g., from cognitive to Pavlovian), or from extrinsic to intrinsic level. For this evaluation, a cognitive controller must possess models of the complete controlling system. It must know about the control mechanisms installed at the physical and instinctive levels so as to take them into account in its decisions. The controller must know the Pavlovian structure to be able to modify this structure and to send the correct routines and parameters. For the purposes of self-organization and self-repair, the cognitive controller should also possess models of itself, or at least of certain components.

III.4.2.6 The communications interface

Communication with its environment forms another major activity of a cognitive machine. The cognitive machine communicates with the Pavlovian machine to transmit to it control routines and parameters, and to obtain information. It also directly monitors the behavior of the controlled system via the perceptors, and transmits orders to the effectors. If necessary, it also allocates computing tasks to other processors. Finally, the cognitive machine is linked to external sources of information (e.g., via electronic mail).

III.4.2.7 Implementation of the structure

As was pointed out previously, it was not the intent of this project to proceed with the implementation of the entire structure proposed for the cognitive controller. However, it can be expected that the construction of such cognitive controllers will require complex combinations of traditional and novel hardware and software. They will most probably consist of hybrid multiprocessor machines, where different chips designed for specific purposes will be combined, e.g., neuro-chips and symbolic processors. Memory requirements will be extreme, both in term of speed of access and volume. At the software level, many technologies will interact, where each will play a role where they are most appropriate, such as artificial neural networks for pattern extraction and learning, genetic algorithms for optimization, Fortran programs for simulation, rule-based experts systems for reasoning, relational databases for simple data organization, and object-oriented programming for complex data representation and manipulation.

III.5 CONSCIOUS CONTROL

III.5.1 Cognition and consciousness

Many definitions exist for the term "consciousness". For example consciousness was defined by Dworetzky (1982) as "a state of awareness of the external environment and of internal events such as thought". In the present conceptual framework, a similar definition is used, except that the concept of "consciousness" is not considered as a faculty specific to biological entities. Consciousness is considered here as a faculty that can be implemented in an artificial system and, in this sense, its use is similar to that of Coles (1993) who worked on the engineering of machine (robot) consciousness. The term consciousness is used here in the sense of self-awareness, i.e., an entity is conscious if it is aware of its existence and its behavior within an environment. A conscious entity possesses the capacity to reason about itself, which is allowed by a cognizance of itself and its interactions with the environment. In this sense, consciousness can be considered as a subset of the overall cognition of a system. The knowledge about itself and its environment is organized into more or less complex and coherent composite structures that are called models. When a conscious entity wants to investigate how it would behave under specific circumstances, it can use these models to simulate some scenarios and make the appropriate analysis based on the simulation results. In this way, consciousness potentially gives intelligent entities the possibility of behaving optimally under specific conditions and to increase their survival capacity in comparison with non-conscious entities. A "conscious" control system, i.e., a system in which the characteristics of a conscious being are mimicked, could, therefore, also possess some advantages over more traditional control systems.

III.5.2 Artificial consciousness

The implementation of elements of "consciousness" or "self-awareness" into a control system to create an artificial consciousness was discussed by Lacroix and Kok (1991a). They mentioned many aspects that a designer should consider, such as the degree and the depth of self-awareness, and the extent to which models are used. Many levels of self-awareness were recognized. At a primary level, an entity will be aware of its physical functioning, but will possess a limited cognizance about its information processing capacity. At secondary and higher levels, the entity will start to be aware of its reasoning capacity and of its information treatment ability. Self-awareness at a secondary level might be the minimum requirement for the task manager of a cognitive controller to be able to optimally allocate the computing resources to different reasoning processes (see Section III.4.2.2).

The only aspects that are considered in this thesis are those related to the implementation of rudiments of self-awareness at a primary level. As previously discussed, one way to achieve this is to give the system access to models of itself, so that it can evaluate the consequences of its decisions by simulating its own behavior for many sets of disturbances and for possible, alternate control actions. This should help the system to choose actions that are the most appropriate for both the control objectives and the external conditions. Thus, an enclosed agro-ecosystem could exhibit some "self-awareness" if certain of its control mechanisms had recourse to simulation during their reasoning processes. These mechanisms would be distributed among all responsibility groups of the cognitive controller (task manager, production controller, etc.) and would act together to perform "conscious control" at a primary level.

III.5.3 Simulation-based control mechanism

For the conceptual framework to be coherent, the general model of the control mechanism that was presented in Section III.3 must be used as a basis to develop the architecture of a simulation-based control mechanism. In this model, the reasoning processes of each control mechanism are directed by an inference engine (Figure III.4), which acts according to considered knowledge and chaining rules. The role of the inference engine is to reach a decision about the control actions that must be taken to satisfy the particular goal associated with that control mechanism. Thus, a simulation-based decision-making process will be executed under the control of the inference engine in a simulation-based control mechanism.

As a basis for the development of a prototype simulation-based controller in the present research, it was decided to develop a mechanism in which the simulation-based decision-making process would be similar to that generally followed by human simulationists. Figure III.8 illustrates the flow of information in the inference engine of this simulation-based control mechanism. Four main phases are represented: 1) simulation design, 2) simulation supervision, 3) evaluation of results and 4) final decision-making.

In the simulation design phase, the domain of the simulation is determined. This includes the choice of the appropriate predictive model(s), the variables to be investigated, the model parameters, the input sequences, the control alternatives to be tested and the simulation parameters. Simulation domains can vary from simple to very complex, depending on the degree of cognition about a system and its boundary conditions.

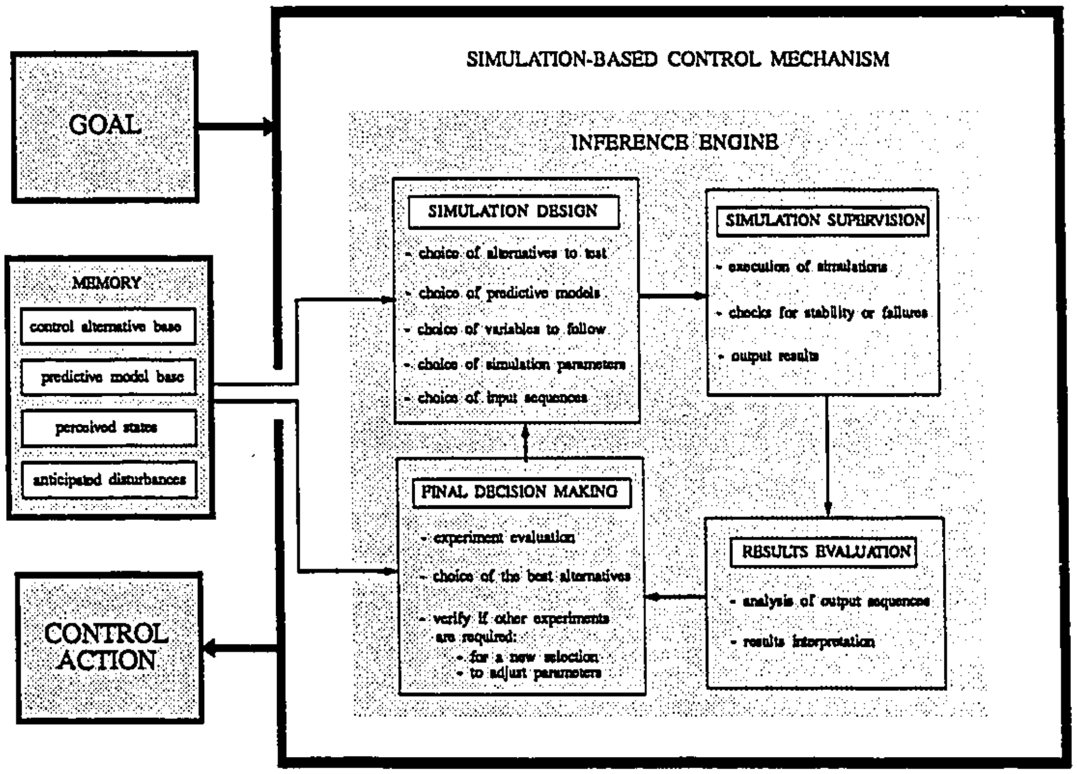
SIMULATION-BASED CONTROL MECHANISM
GOAL
INFERENCE ENGINE
SIMULATION DESIGN
- choice of alternatives to test
- choice of predictive models
- choice of variables to follow
- choice of simulation parameters
- choice of input sequences
SIMULATION SUPERVISION
- execution of simulations
- checks for stability or failures
- output results
MEMORY
control alternative base
predictive model base
perceived states
anticipated disturbances
FINAL DECISION MAKING
- experiment evaluation
- choice of the best alternatives
- verify if other experiments are required:
- for a new selection
- to adjust parameters
RESULTS EVALUATION
- analysis of output sequences
- results interpretation
CONTROL ACTION

Figure III.8 Flow of information in a simulation-based control mechanism inference engine.

When the simulation domain has been established, the simulations are executed. During this phase, the stability of the numerical calculations has to be continuously verified and simulation failures must be detected. The results for the variables that are monitored are put in queues.

When the simulations are completed, the output sequences are analyzed and interpreted. The goal of the analysis is to make recommendations about the modelled system and the alternatives tested. The analysis has to take into account the limitations of the model(s) and the underlying assumptions.

Once the analysis and interpretation of the results are completed, the entire experiment is evaluated, to verify if it is conclusive. If so, the best alternative is chosen. If contradictory results are obtained, or some doubt remains, and if more than one strategy is selected, a conflict resolution must be performed (e.g., by running another set of simulations). Finally, supplementary simulations may be required to adjust some parameters related to the chosen alternative (e.g., fine-tuning of gains after having chosen a PI algorithm).

III.6 CONCLUSION

The conceptual framework that was developed to help in the design of enclosed agro-ecosystems was presented in this chapter. The general model of a greenhouse was first introduced and the control system was presented as a network of control mechanisms, accessing a common pool of variables. The cognitive controller, composed of the control mechanisms implemented at the cognitive level, was then discussed. These mechanisms were described as acting together to perform some cognitive functions (e.g., pattern recognition, problem solving), while being part of responsibility groups (e.g., task manager, production control). Finally, conscious control was defined as a subset of cognitive control, and as a form of control achieved by a system that has recourse to models of itself in its decisions. An approach for designing mechanisms that can perform some conscious control at a primary level was then developed. The approach consists of giving the mechanisms the possibility to use simulation during their decision-making phase.

The design and implementation of control systems possessing the characteristics described in the conceptual framework constitutes a complex task and simulation systems will be helpful towards the achievement of that. This is why one objective of this research was the development of a simulated greenhouse system. A related objective was to investigate how a multitasking operating such as OS/2 can be exploited to implement complex simulation structures in an innovative way. Once created and implemented within OS/2, the simulated greenhouse system was to be used to develop the prototype of a simulation-based cognitive controller. The objective in developing this prototype was to determine a series of factors and constraints to be considered in the design of such a system, for the eventual development of a methodology for the design of controllers performing conscious control, and the eventual elaboration of a theory on conscious control.

IV. DESCRIPTION OF THE SIMULATED GREENHOUSE SYSTEM

IV.1 INTRODUCTION

One objective of this research was to construct a greenhouse simulation system which can be used to develop and test complex cognitive controllers. The intent was to obtain a simulator that gives a representation of a general greenhouse, and not to simulate a particular installation. Greenhouse and crop modelling has been an active field of research over the last 30 years and the models that were developed are able to generate results that represent many aspects of physical reality quite well. The use of such models is sufficient when the objective is to construct simulators to help in the development of concepts, principles and methods applicable to a large class of systems.

The greenhouse system and the control situation used in this thesis are illustrated in Figure IV.1. The production setting is composed of a greenhouse model which is largely based on GGDM2 (Gembloux Greenhouse Dynamic Model; de Halleux, 1989). The crops are represented by a crop model that includes some of the functions defined in SUCROS87 (Simple Universal CROp Simulator; Spitters *et al.*, 1989). The extrinsic control subsystem, whose scope in this thesis was limited to managing the greenhouse climate, consists of two units: the cognitive and the Pavlovian controllers. In this case, the role of the cognitive controller consists of feeding temperature setpoints to the Pavlovian controller. In the deliberations of the cognitive controller, various scenarios are simulated in response to weather forecasts and the most appropriate setpoints are then chosen. The model it uses for its simulations is an artificial neural network. The main activity of the Pavlovian controller is to regulate the greenhouse temperature according to the setpoints determined by the cognitive controller. The effectors used for this are the heating and ventilation subsystems.

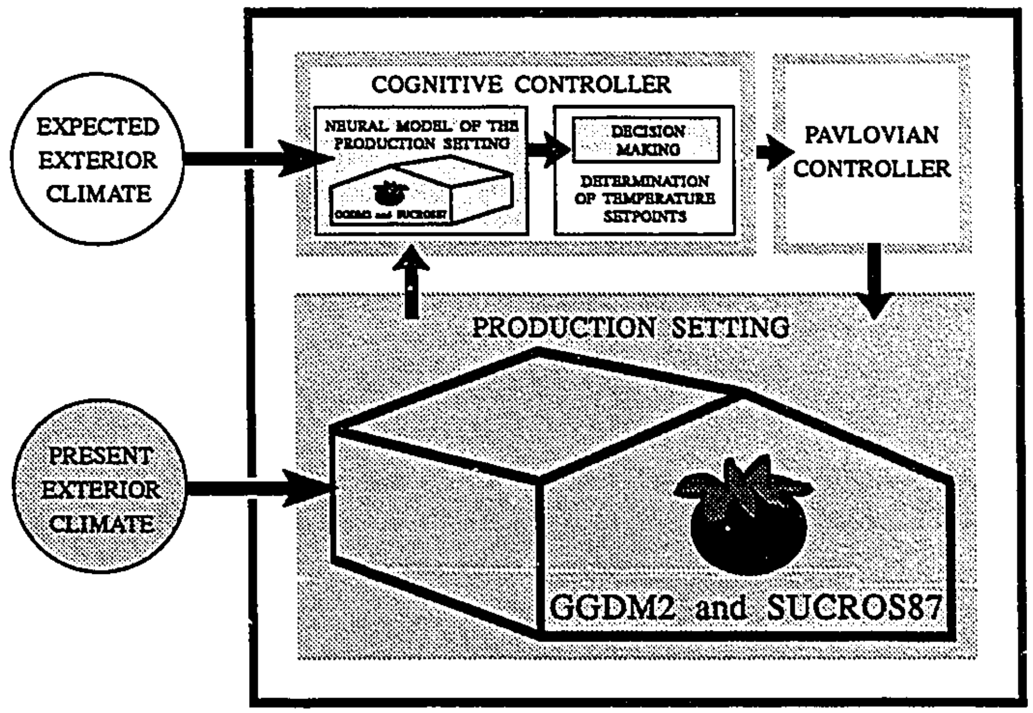

Figure IV.1 The simulated greenhouse system.

In the situation illustrated in Figure IV.1, two levels of simulation coexist. On the one hand, there is a simulated greenhouse containing simulated crops. On the other hand, the cognitive controller possesses a neural model of the production setting to run simulations on which to base its decisions. To avoid confusion in the following discussion, it is important to define some terminology. The simulation of the complete greenhouse system, comprising the controlled and the controlling subsystems, is referred to as the "functional simulation", whereas the term "subjective simulation" is used to refer to the simulations executed by the cognitive controller as part of its decision-making activities. Simulations at both levels are time based, but each refers to a specific and separate time frame. For the functional simulation this is called "functional time" whereas for the subjective simulations it is called "subjective time".

The main components of the functional greenhouse system include a greenhouse model, a crop model, a Pavlovian controller, a cognitive controller and a weather generator. These modules are respectively referred to as GHOUSE, CROP, PAVLOV, COGNITI and WEATHER. In this chapter, the components are described separately. The implementation of the functional simulation structure under the multitasking operating system OS/2 is described in Chapter V.

IV.2 THE GREENHOUSE

IV.2.1 Greenhouse models

Many greenhouse models have been constructed over the last 30 years. Lacroix and Zanghi (1990) analyzed and compared the structure of 33 such models. They classified the models into four categories: 1) static and unicomponent, 2) static and multicomponent, 3) dynamic and unicomponent and 4) dynamic and multicomponent. Unicomponent models establish the energy and mass balances only for the internal air, while in multicomponent models, balances are also established for the covers, the crop and the soil. When one wants to study the behavior of a particular system component such as crops, a multicomponent model is necessary. Static models are composed of algebraic equations and can be used only to study the average behavior of greenhouses over relatively long time periods, i.e., over at least one hour. These models do not take into account the energy storage in the various components of a greenhouse. Dynamic models, i.e., models composed of sets of differential equations, are necessary to study the short term behavior of a greenhouse.

In this project, it was required that the energy balance of the crop be calculated since the leaf temperature is an input to the crop model. Also, it was desirable that the simulated greenhouse be equipped with a thermal screen. Such a configuration is necessary to reduce the energy consumption in cold climates. At the same time, it leads to complex, non-linear behavior, which increases the challenges in the development of a simulation-based cognitive controller. GGDM2 was chosen because it is a multicomponent model, the energy balance of the crop is calculated, and it allows the simulation of a greenhouse equipped with a thermal screen.

IV.2.2 The model GGDM2

GGDM2 was described in detail by de Halleux (1989), who validated this model in Belgium and in Canada. GGDM2 is based on a set of 11 differential equations that describe the energy balances for two covers, the air volume between the covers ("Air 2" or "gas"), the air between the bottom cover and the crop ("Air 1" or "inside air"), the crop, and four soil layers. As well, it performs moisture balances for the two air volumes (Figure IV.2). The behavior of a greenhouse simulated with GGDM2 depends strongly on many model parameters (e.g., emissivity of covers), which can be modified to simulate different greenhouses. By choosing appropriate values, the model can be configured as two transparent covers or as a combination of a thermal screen and a single transparent cover. The latter combination was used in this research. The inputs to GGDM2 consist of date, time and meteorological conditions. The meteorological conditions include the global and diffuse components of solar radiation, dry-bulb temperature, humidity, wind speed and "sky temperature".

In GGDM2, all energy transfers, except those involved in ventilation, occur between horizontal, lumped strata. The properties of each stratum are thus assumed to be homogeneous. There is one stratum for each component previously listed (cover, thermal screen, air volumes, etc.). This kind of model, defined as the "quasi-one-dimensional type" by Seginer and Levav (1971), approximates the situation existing at the center of a large multispan greenhouse (Lacroix and Zanghi, 1990). This approach has been used by most researchers who have constructed multicomponent, dynamic models (e.g., Van Bavel and Sadler, 1979; Kindelan, 1980; Cooper and Fuller, 1983).

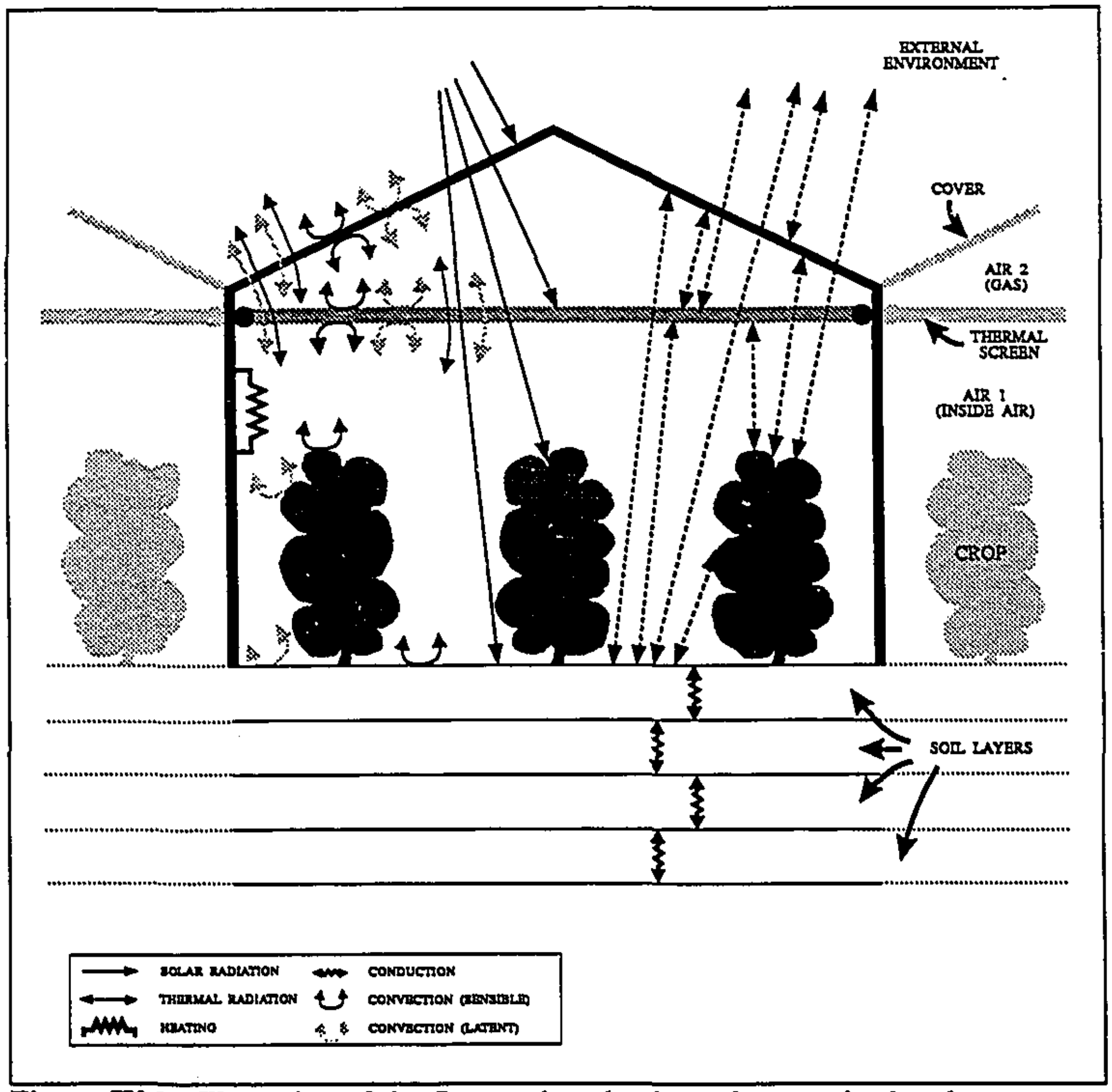

Figure IV.2 Illustration of the fluxes when the thermal screen is closed.

A total of 37 fluxes are involved in the energy and mass balances, when the thermal screen is closed. These fluxes are enumerated in Appendix B.1 and are illustrated in Figure IV.2. There are energy transfers by convection (latent and sensible heat - 18 fluxes), by thermal radiation (10 fluxes) and by conduction (4 fluxes). The energy balances of the soil surface, cover, thermal screen and crops are also affected by solar radiation (4 fluxes), for which the diffuse and direct components are treated separately. Part of the solar energy received by each component is reflected and the reflected radiation contributes to the diffuse radiation received by the other components. The model does not consider multiple reflections, since they contribute less than 5% of the total solar energy received by the greenhouse (de Halleux, 1989). Reflections are not illustrated in Figure IV.2. The 37th flux is sensible heat injected by the heating system. The various energy transfer mechanisms will not be described here. That information can be found in de Halleux (1989).

IV.2.3 The GGDM2 simulation program

GGDM2 was originally implemented as a computer program written in FORTRAN. A copy of this program was obtained from its author (de Halleux, 1989) and adapted for the needs of this research. Major modifications were done to allow for the control of the greenhouse temperature, and to permit for the opening and the closing of the thermal screen. Models of a heating system, a ventilation system, crop transpiration and air infiltration were also added. GGDM2 was also restructured to leave in the main program only operations such as data input and output, summations and averaging, etc. The main program was rewritten in BASIC to ease the implementation of the module GHOUSE in the functional simulation structure. All other operations were moved to subroutines which were left in FORTRAN (callable from BASIC).

IV.2.4 Modifications to GGDM2

IV.2.4.1 Opening and closing of the thermal screen

In its original form, the GGDM2 simulation program permitted only the simulation of a greenhouse with a screen installed which was permanently shut. Modifications were made to the program to allow for the simulation of the greenhouse for when the thermal screen is opened and for the simulation of the opening of the screen at sunrise and its closing at sunset.

The approach that was adopted to simulate the greenhouse when the thermal screen is opened at the beginning of the day is as follows: 1) all radiative and convective fluxes between the thermal screen and the other components of the greenhouse model are set to zero; 2) the two air volumes Air 1 and Air 2 are maintained as two distinct entities, but the air flow rate between the two volumes is set at a higher value than during the night; 3) the transmissivity of the thermal screen to solar radiation and to thermal radiation is set to 100%. These conditions are then maintained during the rest of the day. When night comes again, all radiative and convective fluxes between the thermal screen and the other greenhouse components are allowed to have a value different from zero, i.e., they are calculated with the mechanism originally implemented in GGDM2. Also, the air flow rate between Air 1 and Air 2 is decreased to a low value and the transmittance of the screen to thermal radiation takes its default (night) value.

The approach implies that there is no energy exchange between the screen and the other components during the day. The daytime energy fluxes are illustrated in Figure IV.3. In comparison to night, nine fluxes are absent (see Figure IV.2): convective exchanges (latent and sensible) between the two air volumes and the screen, solar gains by the screen and radiative exchanges between the screen on one side and the crop, the soil, the cover and the sky on the other side.

The approach used to simulate screen opening and closing generally worked well. However, it was observed in many experiments that numerical instability can occur at the moment when the thermal screen is opened. Such instability occurs when the outside temperature is low during the night. This creates a large difference in temperature between Air 1 and Air 2 which, in turn, creates large convective fluxes at screen opening. If the time increment used in the functional simulation is too large, this leads to numerical instability. Numerical stability can be maintained by reducing the time increment, or by gradually shifting the air flow rate between Air 1 and Air 2 from its night to its daytime value during a transitional period. The use of small time increments is not desirable,

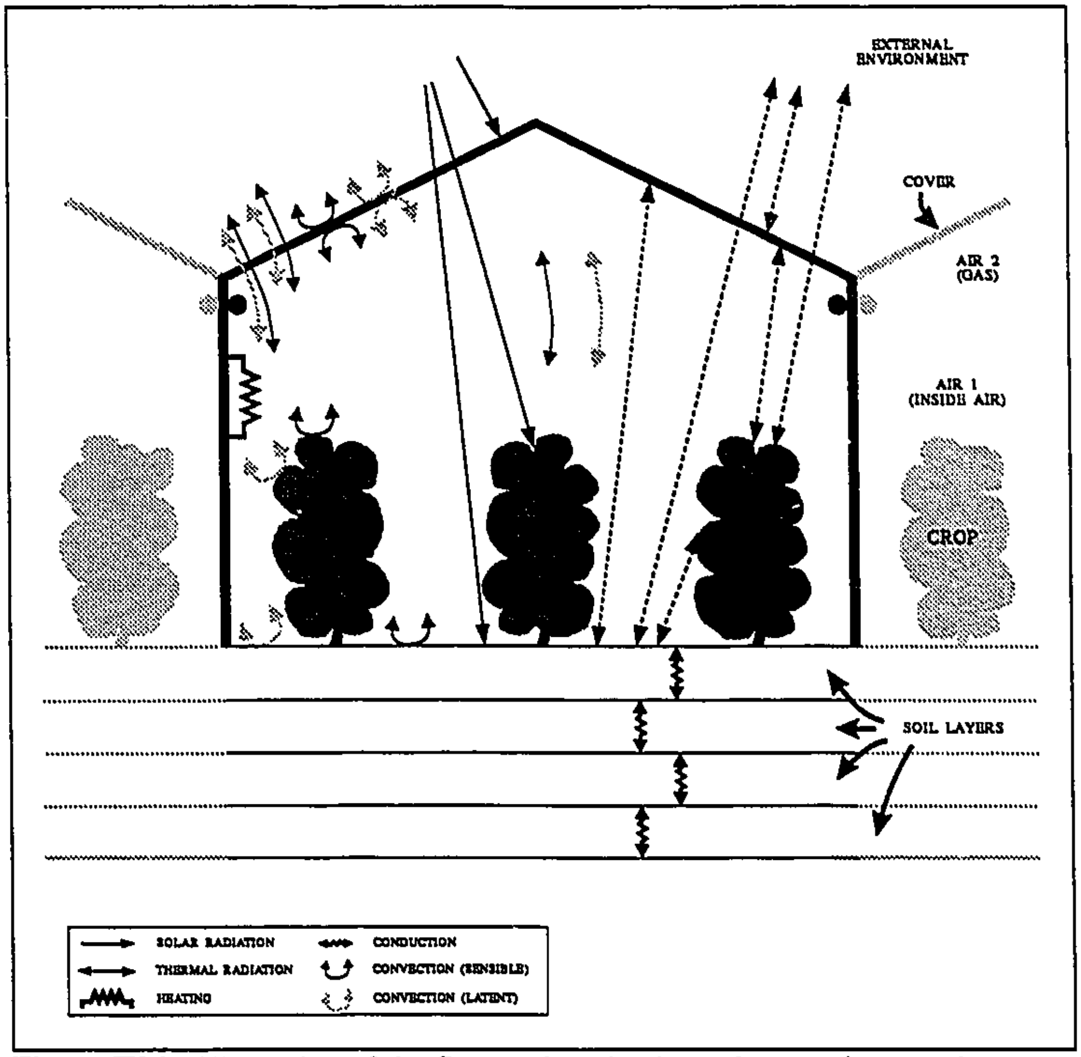

Figure IV.3 Illustration of the fluxes when the thermal screen is opened.

because it leads to long physical periods to run simulations. A variable time step could be used, but this would increase the complexity of the simulation. It was, therefore, decided to simulate a gradual opening of the thermal screen. In reality, in the greenhouse industry, thermal screens are often opened gradually too. The gradual opening is simulated by increasing slowly the flow rate from its night value to its day value. However, it is assumed that, during this transition, all other fluxes (convection, thermal radiation and solar radiation) occur as if the screen were fully opened. With the functional time increment and the air flow rate values used in this research, it was established that a transition period of 30 simulated minutes was adequate. It should be noted that, since the temperature of the two internal air volumes is similar during the day, the use of such a transition is not necessary to assure the numerical stability of the simulations when the screen is being shut.

For this research, the opening of the screen in the morning started when the solar radiation intensity outside the greenhouse was greater than 10 W/m^2. The screen was closed in the evening when the light intensity was lower than the same threshold value. In terms of the terminology developed for the conceptual framework, the thermal screen was controlled intrinsically, i.e., its state was not determined by extrinsic controllers.

IV.2.4.2 Control of the inside temperature

In the original GGDM2 program, the temperature of inside air (Air 1) was fixed at a specific value and the ventilation rate or heating load necessary to maintain this value were calculated. For this research, the subroutines were modified to allow for the internal temperature to fluctuate and to permit its control by means of the heating and ventilation systems. To simplify the calculations, it was assumed that heating and ventilation directly affect only Air 1. This represents the situation where the heat distribution system and the ventilation equipment are installed below the thermal screen.

Since GGDM2 did not contain models of heating and ventilation, these were added. The heating system was of the ON/OFF type; an effective power of 300 W/m^2 was found to be sufficient under most meteorological conditions with which the glasshouse simulated in this project was tested. The ventilation system was used to avoid overheating of the greenhouse when solar radiation intensity and/or outside temperature were too high. The model imitated a description of an experimental greenhouse used by Lessard (1989). It was composed of subsystems for small ventilators and large two-speed fans. The subsystem for the small fan ventilators was assigned a total capacity of 3.5 air renewals per hour (in terms of volume of Air 1). The subsystem consisting of the large fan ventilators was assigned capacities of 22.5 and 34 air renewals per hour (in terms of volume of Air 1), respectively for the low and the high speeds. The models of the heating and ventilation systems were kept rather simple, i.e., when ventilation or heating was necessary, the systems operated instantaneously at their full capacity.

IV.2.4.3 Crop transpiration

Crop transpiration makes an important contribution to humidity in greenhouses. It also affects the energy balance of plants since heat is necessary for evaporation. To date, many transpiration models have been constructed (Yang *et al.*, 1987; Willits *et al.*, 1981; Seginer, 1984) that use a common procedure. The transpiration rate is considered to be a function of the difference between the absolute humidity at saturation for the leaf temperature and the absolute humidity of the air. The rate is weighted by a total resistance, which is often considered as the sum of stomatal resistance and the resistance associated with the boundary layer at the leaf surface. These resistances are in series. For hypostomatous crops, such as tomatoes, the stomatal resistances of the lower and the upper leaf surfaces differ (Maher and O'Flaherty, 1973). In absence of any criteria for making an optimal choice, it was decided to adopt the model used by Maher and O'Flaherty (1973). The transpiration rate, which is assumed to directly affect only the crop and Air 1, is calculated by equation IV.1:

$$E_t = LAI * \left[\frac{1}{R_{t,l}} + \frac{1}{R_{t,u}}\right] * [W_{l,s} - W_a] \qquad \textbf{(IV.1)}$$

where:

E_t	: transpiration rate	[kg/m^2·s]
LAI	: Leaf area index	[m^2/m^2]
$R_{t,l}$	: Total resistance to water transfer from lower leaf surface to air	[s/m]

$R_{t,u}$	: Total resistance to water transfer from upper leaf surface to air	[s/m]
$W_{l,s}$	: Absolute humidity at saturation at the leaf temperature	[kg/m^3]
W_a	: Absolute humidity of the air (Air 1)	[kg/m^3]

The resistances for the lower and upper leaf surfaces are calculated with equations IV.2 and IV.3:

$$R_{t,l} = R_a + R_{s,l} \tag{IV.2}$$

$$R_{t,u} = R_a + R_{s,u} \tag{IV.3}$$

where:

R_a	: Aerodynamic resistance	[s/m]
$R_{s,l}$	: Stomatal resistance of the lower surface	[s/m]
$R_{s,u}$	: Stomatal resistance of the upper surface	[s/m]

The aerodynamic resistance is calculated with equation IV.4 (Maher and O'Flaherty, 1973):

$$R_a = 480 * (T_l - T_a)^{-0.25} \tag{IV.4}$$

where:

T_l	: Leaf temperature	[°C]
T_a	: Air temperature (Air 1)	[°C]

Different values have been suggested in the literature for the stomatal resistance (Bellamy and Kimball, 1986; Yang *et al.*, 1987; Johnstone and Ben Abdallah, 1989). These values vary greatly and are sometimes considered to be a function of the incoming solar radiation. In the absence of a consensus and criteria to choose a particular set of values, the stomatal resistance values suggested by Maher and O'Flaherty (1973) were used:

$$R_{s,l} = 60 \text{ s/m}$$
$$R_{s,u} = 3000 \text{ s/m}$$

IV.2.4.4 Infiltration rate

Originally, the infiltration rate was kept constant in GGDM2, however, it was shown to be dependent on wind speed (Townsend *et al.*, 1978; Short and Bauerle 1977; Okada and Takakura, 1973; Whittle and Lawrence, 1960). For example, Monteil (1985) reported infiltration rates between 0.5 air renewals per hour (AR/h) and 3 AR/h for a wind speed varying between 0 m/s and 10 m/s.

The sensitivity of the greenhouse model to wind speed was originally very low in comparison to what is described in the literature (see Appendix B). It was, therefore, decided to consider the infiltration rate as a function of wind speed. Many relationships between infiltration and wind speed have been used in different simulation models (Bellamy and Kimball, 1986; Chiapale *et al.*, 1983; Rotz, 1977). It was arbitrarily decided to use the relationship developed by Okada and Takakura (1973) for a glasshouse, which was described by equation IV.5:

$$V_g = 0.44 * u + 0.14 * (T_i - T_o)^{0.5} \qquad \textbf{(IV.5)}$$

where

V_g:	air renewal rate	$[m^3/m^2 \cdot h]$
u :	wind speed	[m/s]
T_i:	inside temperature	[°C]
T_o:	cutside temperature	[°C]

IV.2.5 Model parameters

As previously mentioned, the behavior of the greenhouse model depends strongly on the value of the parameters. The values chosen for this research are enumerated and briefly discussed in Appendix B. Most values came from de Halleux (1989). The transparent cover material was ordinary horticultural glass with a thickness of 4 mm. The thermal screen was made of aluminized polyester, completely opaque to far infrared radiation. The crop parameters such as the leaf area index and the vegetal mass remained constant during simulations, which were all done over a maximum period of one month. The soil surface was white, as if it was covered with a white polyethylene. This is often the case in greenhouses with hydroponic culture. The configuration of the simulated greenhouse setting is illustrated in Figure IV.4, where the dimensions are given for one span. A complete greenhouse system would be formed by many of these spans side by side.

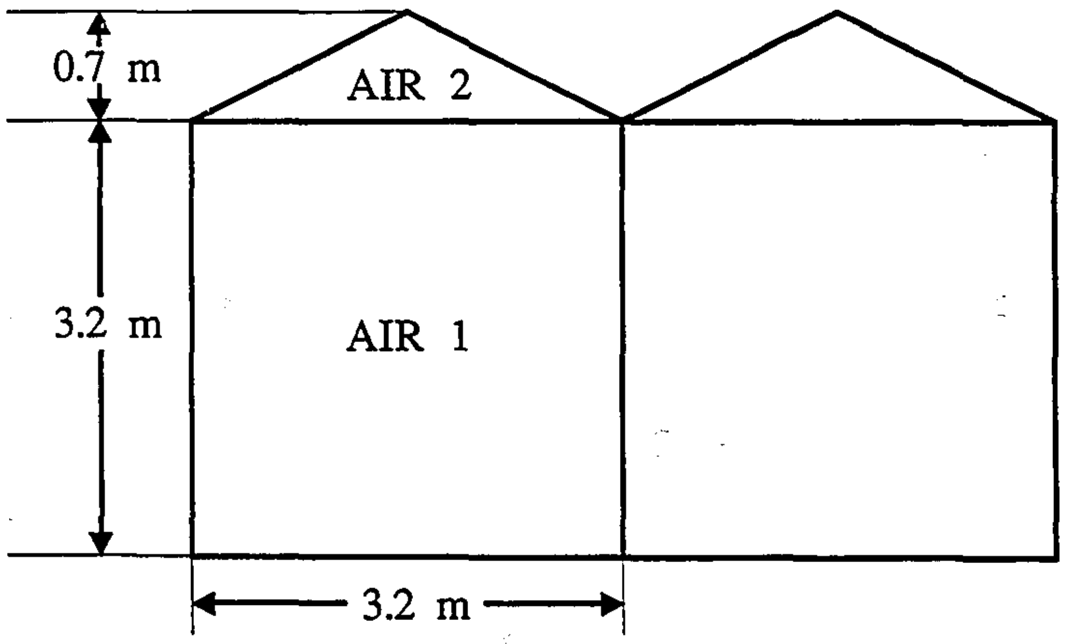

Figure IV.4 Configuration of the simulated greenhouse.

IV.2.6 Steady-state analysis of the greenhouse model

A series of simulations were executed to analyse the steady-state response of the greenhouse model under various meteorological conditions. This was done to verify that the simulation results are reasonable and free of aberrations, and compare the results with values found in the literature. More specifically, the analysis consisted mostly of studying the effects of the outside temperature, wind speed, solar radiation and cloudiness on the heating and ventilation requirements, and on the temperature of the diverse greenhouse components. The simulations and the results are described in details in Appendix B.3. The steady-state response of the greenhouse model, as it was configured, was generally found to be satisfactory and adequate for the present study.

IV.3 THE CROP

IV.3.1 Crop modelling

To predict crop yield, three aspects must be considered: growth, partitioning and development. Many models have been created for each aspect. The processes represented in crop growth models are generally photosynthesis and respiration. Photosynthesis has been modelled with different approaches, varying from simple empirical relationships to more complex relations based on biophysical processes. In the models, the main factors affecting photosynthesis are light intensity, temperature and CO_2 concentration, whereas respiration is considered to be essentially a function of the temperature (Curry, 1971; Van Bavel, 1975; Enoch, 1978; Seginer *et al.*, 1986; Gary, 1988). Two major simplifications are generally encountered in growth models. First, only potential growth (i.e., without deficit of nutrient or water) is modelled. Thus, the processes occurring at the roots are not considered. Second, crops are generally considered as lumped systems and the vertical gradients existing in the concentration of CO_2, the temperature of the air, and in the characteristics of the crop, are not considered.

Partitioning consists of distributing the photosynthetic products to the different parts of the crops (e.g., fruits, leaves, stems, roots). France and Thornley (1984) presented several approaches to modelling partitioning. The simplest is empirical and is based on "partitioning fractions". The partitioning fractions, which consist of the fractions of photosynthetic products allocated to each part of the plant, are considered to vary from one stage of crop development to another.

Development consists of the passage of the plant from one stage to another (e.g., germination and emergence, floral initiation, fruit appearing). A simple approach to modelling crop development is based on the degree-day accumulation (France and Thornley, 1984; van Keulen and Wolf, 1986).

IV.3.2 Choice of model

In this research, the main criterion for choosing a crop model was the ability of the model to predict yield in response to greenhouse temperature. It was also desirable that the model permit the calculation of the instantaneous rates of photosynthesis and respiration, for the eventual calculation of a CO_2 balance for the air in the greenhouse system in future research projects. The model SUCROS87 satisfied these criteria and was then chosen for this research. SUCROS87 was written in the Netherlands and was described by Spitters *et al.* (1989). It generates potential crop growth, i.e., growth under conditions of sufficient supply of water and nutrients in a pest-free environment. In this model, photosynthates are allocated using the partitioning factor method.

In this research, SUCROS87 was used to simulate the growth of tomatoes. Since simulations were always done for periods less than one month (in the functional time frame), it was assumed that the development stage of the crops did not vary. Thus, it was not necessary to implement a development model in the simulation structure. Also, for the same reason, it was assumed that all crop parameters, such as the leaf area index, did not vary during the simulations.

IV.3.3 The simulation program SUCROS87

The version of SUCROS87 that was obtained had been conceived to predict growth rate inside a greenhouse over a long period of time (e.g., one year). The program included subroutines that were not used in this project (e.g., to estimate the diurnal trends of the direct and diffuse components of the solar radiation inside a greenhouse). For this research, only the subroutines required to calculate the instantaneous canopy assimilation rate were conserved for implementation in the crop module (CROP). The leaf temperature and the solar radiation above the canopy were calculated by the greenhouse model (i.e., in the module GHOUSE) and were transmitted to the module CROP to be used as input by the crop model.

The program SUCROS87 had been written in FORTRAN. A procedure similar to that used with the greenhouse model GGDM2 was used for the implementation of the crop model in the functional simulation. The required subroutines were extracted and adapted to the needs of this research. They stayed in FORTRAN and were made callable from BASIC. A main program was written in BASIC to serve as the entry port in the simulation. The FORTRAN subroutines and the main program written in BASIC together formed the crop module.

IV.3.4 Crop parameters

The functioning of the crop model depends on parameters. The parameter values used in this research are listed in Appendix C. They are the same as those included in the original program SUCROS87, excepting the weight of each component of the plant. The weights were adjusted so that the total weight of the crop would be the same as the value used by the greenhouse model. As previously mentioned, all crop parameters were kept constant during functional simulations.

IV.3.5 Input variables

The input variables for the crop model are the leaf temperature, CO_2 concentration and incoming direct and diffuse components of the solar radiation. In the functional simulation structure, the values for all these variables are furnished by the module GHOUSE. Only the fraction of photosynthetically active radiation (PAR) in solar radiation is required by the crop model. The PAR fraction was considered to be 50% for both the direct and diffuse components (Cooper and Fuller, 1983; van Heemst, 1986; Spitters *et al.*, 1989). Since the CO_2 balances in this project were not established in the module GHOUSE, a constant value of 330 ppm was assumed for the CO_2 concentration.

IV.3.6 Crop model analysis

A series of calculations was done with the crop model to see how the net CO_2 assimilation rate would vary with different sets of input values. Specifically, the intent was to investigate the effect of a variation of the leaf temperature in combination with variations in solar radiation intensity and CO_2 concentration on the model. This was pursued because, in this research, the main goal of the cognitive controller was to control air temperature, which plays a large role in the determination of the leaf temperature.

The calculations performed helped the author to become familiar with some of the possible responses of the crop model under various conditions. The model was shown to be sensitive to the three input variables. A particularly interesting point for this research is that the net assimilation rate varied with the leaf temperature for a given solar radiation intensity and CO_2 concentration. Thus, different temperature control strategies may have different impacts on crop production. The variations on the assimilation rate induced by different CO_2 concentrations was not a specific concern of the present research. However, this will become an important aspect with the eventual addition of CO_2 balances and CO_2 control in the functional simulation.

IV.4 THE PAVLOVIAN CONTROLLER

The main function of the Pavlovian controller (module PAVLOV) in this project was the regulation of the greenhouse temperature. The decision-making capacity of the Pavlovian controller was low, but was sufficiently developed to allow control without the help of the cognitive machine. The Pavlovian controller set the functioning state of the heater and the ventilators. In this project, the humidity of the greenhouse air was not controlled. In the future, it will be relatively easy to add mechanisms for humidity control with the simulation structure that was developed. For the same reason, it will be also easy to add CO_2 control.

The decision-making logic of the Pavlovian controller for this project is illustrated in Figure IV.5. In this scheme, there are two temperature setpoints: the heating setpoint, SETPOINT(H), and the ventilation setpoint, SETPOINT(V). The heating system is controlled with SETPOINT(H) and the ventilators are controlled with SETPOINT(V). The values for these two setpoints are determined by the cognitive controller and their realisation is assured by the Pavlovian controller. For example, when the air temperature of the greenhouse falls by more than 0.5°C below SETPOINT(H), the Pavlovian controller turns ON the heater; it turns OFF the heater when SETPOINT(H) is overshot by more than 0.5°C. When SETPOINT(V) is overshot by 1.0°C, the small ventilators are started; they are stopped when the temperature has returned below SETPOINT(V). The first and second levels of ventilation using the large fans are started when SETPOINT(V) is respectively overshot by 2 and 4°C. These values are similar to those used by Lessard (1989) who, however, did not use different ventilation and heating setpoints.

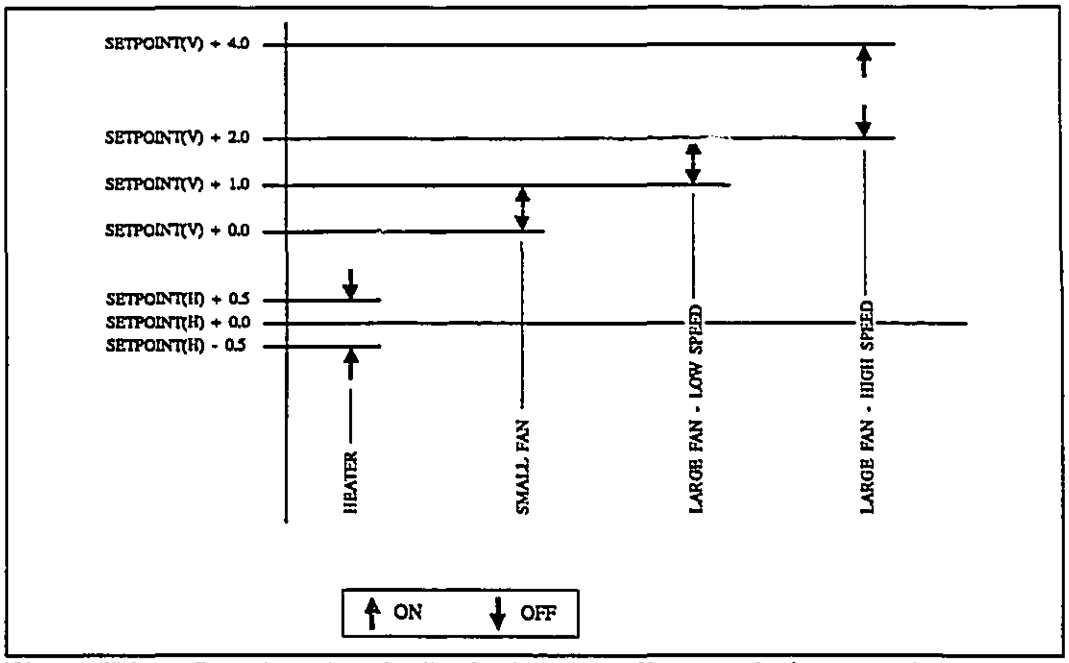

Figure IV.5 Setpoints that the Pavlovian controller uses in its control decisions.

IV.5 THE SIMULATION-BASED COGNITIVE CONTROLLER

IV.5.1 Role of the cognitive controller

In the present investigation phase, the function of the cognitive controller (module COGNITI) was limited. It concerned only the determination of the heating setpoint under various meteorological conditions. The role of the cognitive controller was to consider the anticipated meteorological conditions for the coming 24-hour period and to calculate a path of hourly setpoints so as to optimize the value of a certain variable. To make a "conscious" choice, the controller generated various setpoint paths, evaluated these by simulating the system response to them, and then picked the best one.

Various variables can be optimized when determining temperature setpoints. For this research, the emphasis was on the minimization of the amount of energy required for heating. Also, in some experiments, the variable to be optimized was the "net income", considered as the difference between the income from fruit sales and the expense due to heating. The approach used by the cognitive controller for the achievement of its goal is presented in more detail in Chapter VI.

IV.5.2 The simulation-based control mechanism

Within the conceptual framework, as described in Section III.4.2, a cognitive controller is composed of a set of control mechanisms. In this research, the cognitive controller consisted of a single mechanism that was responsible for determining the temperature setpoints. This mechanism exercised a certain degree of conscious control since, during its decision-making processes, it used a model of the controlled system to run simulations and to choose the most appropriate control action. Figure IV.6 illustrates the general architecture that was used as the basis for the conception of this simulation-based control mechanism. In this, each time the mechanism is activated, it 1) determines the domain of simulation, 2) runs the required simulations, 3) analyses the simulation results, and 4) finds the most appropriate setpoint path. The domain of the simulation consists of the most probable weather scenario, together with various possible temperature setpoint paths for the following 24-hour period. The most probable weather scenario consists of a sequence of hourly values for outside temperature, global solar radiation and wind. These are the most important variables to consider since they play a major role in the energy balance of the greenhouse; also, they have an impact on crop growth. In a physical context, the mechanism should access weather forecast bulletins and generate some weather scenarios according to these bulletins. However, in this research, it was assumed that the incoming meteorological conditions were always perfectly known, i.e., that the forecasts were ideal.

The system response model used by the cognitive controller in its simulations was a back-propagation artificial neural network (ANN). The ANN had to be trained so as to approximate the relationship existing between, on one side, the weather conditions and

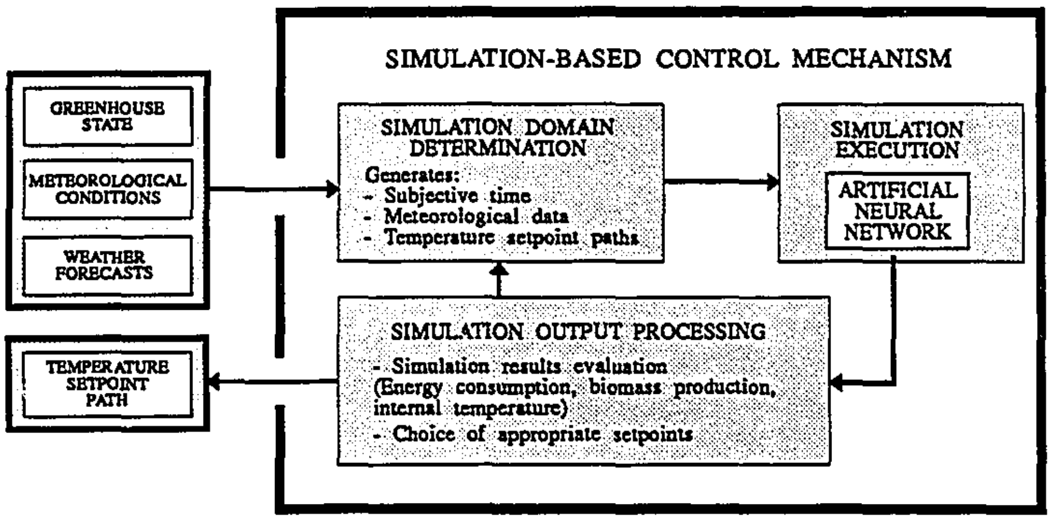
SIMULATION-BASED CONTROL MECHANISM
GREENHOUSE STATE
METEOROLOGICAL CONDITIONS
WEATHER FORECASTS
TEMPERATURE SETPOINT PATH
SIMULATION DOMAIN DETERMINATION
Generates:
- Subjective time
- Meteorological data
- Temperature setpoint paths
SIMULATION EXECUTION
ARTIFICIAL NEURAL NETWORK
SIMULATION OUTPUT PROCESSING
- Simulation results evaluation
(Energy consumption, biomass production, internal temperature)
- Choice of appropriate setpoints

Figure IV.6 General architecture of the prototype cognitive controller.

the temperature setpoint, and, on the other side, the state of the greenhouse and the crop. As shown in Figure IV.7, the inputs to the ANN were: day of the year and time, global solar radiation and outside temperature with lagged values, wind speed, lagged inside temperature and the present temperature setpoint. The day and time informed the ANN about the position of the sun, which affects the optical characteristics of the greenhouse cover. The lagged values for internal and external temperature and solar radiation gave the ANN the capacity to behave dynamically like the modelled system. The outputs were: inside temperature, heating load and crop growth.

As pointed out, the ANN had to be trained before being used by the cognitive controller. The data sets for the training were created with simulations done with the functional simulation structure for various sequences of temperature setpoints and meteorological data. The procedure used is presented in Chapter VI, together with the details for the architecture and the implementation of the simulation-based cognitive controller.

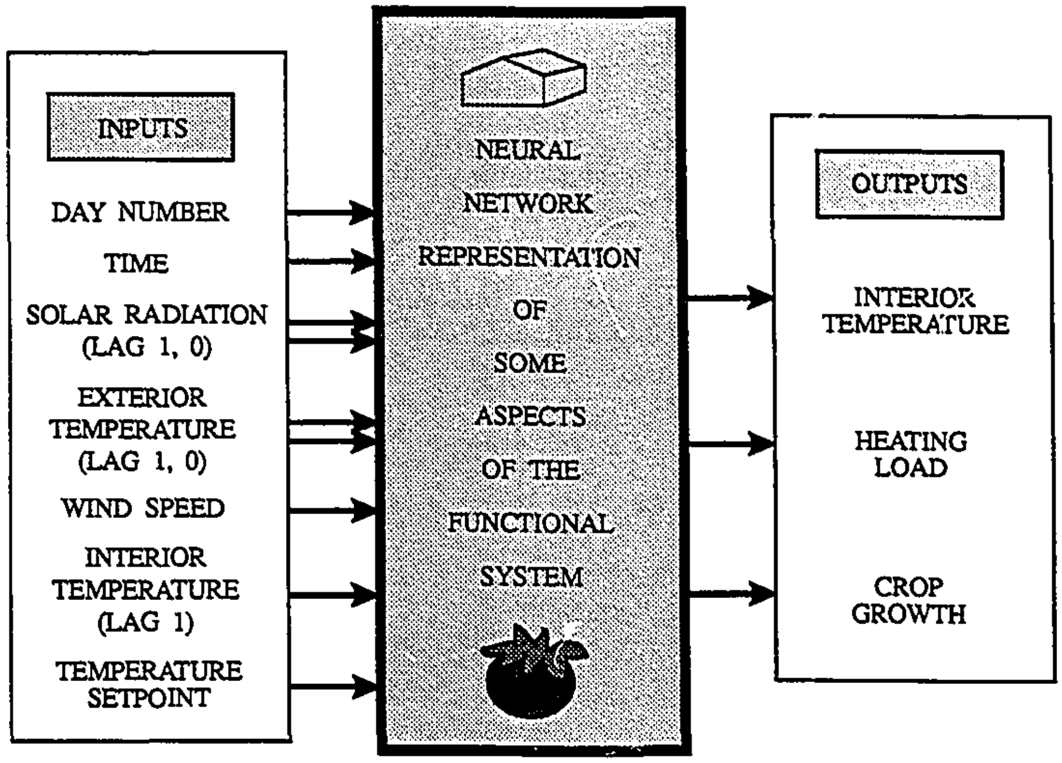
INPUTS
DAY NUMBER
TIME
SOLAR RADIATION (LAG 1, 0)
EXTERIOR TEMPERATURE (LAG 1, 0)
WIND SPEED
INTERIOR TEMPERATURE (LAG 1)
TEMPERATURE SETPOINT
NEURAL NETWORK REPRESENTATION OF SOME ASPECTS OF THE FUNCTIONAL SYSTEM
OUTPUTS
INTERIOR TEMPERATURE
HEATING LOAD
CROP GROWTH

Figure IV.7 Inputs and outputs of the neural model.

IV.6 WEATHER INPUTS

IV.6.1 Simulation requirements

The meteorological input variables for the functional simulation were determined by the requirements of the greenhouse model. They included the global and diffuse components of solar radiation, dry-bulb temperature, humidity, wind speed and sky temperature. The values for these variables were produced by the module WEATHER. At least two approaches existed for the production of meteorological data by WEATHER. The first approach consisted of using weather models. The second approach consisted of using a "data table" approach and supplying data values from files. For this research, the second approach was used for all variables, except sky temperature. The data tables contained hourly climatological data measured at Montréal and obtained from the Atmospheric

Environment Service (AES - Environment Canada). The sky temperature was not available from Atmospheric Environment Service (AES) and a model was used for its calculation.

To calculate the sky temperature, the equation proposed by Dogniaux and Lemoine (1984) was chosen. This model, whose usage has been shown to be valid in Canada (Gueymard, 1988), possesses the advantage that it takes into account the cloudiness. Contrary to clear-sky models, it can then furnish the sky temperature under any condition. In this model, the sky temperature is a function of ambient temperature, vapor pressure and cloudiness, as shown in equation IV.6. These variables are all available from AES.

$$T_s = T_o * [0.904 - 0.005 * e_v^{0.5} - (0.304 - 0.061 * e_v^{0.5}) * (10 - C)]^{0.25} \qquad \textbf{(IV.6)}$$

where:

T_s	: Radiative sky temperature	[K]
T_o	: Ambient temperature	[K]
e_v	: Vapor pressure	[hPa]
C	: Cloudiness	[tenths]

IV.6.2 Format of AES data

Atmospheric Environment Service data is recorded on an hourly basis. For all variables except solar radiation, the values contained in data files supplied by AES represent punctual measurements taken at the beginning of each hour from 00:00 to 23:00 h. This time is "local standard time" (Atmospheric Environment Service, 1983; Chatigny *et al.*, 1981). For solar radiation, the value represents the irradiation integrated over one hour. In this case, the time indicated is "true solar time". Local standard time is the conventional time used everywhere within any given time zone, and is equal to the "mean solar time" at the meridian of reference for the given time zone. The mean solar time is an approximation of the true solar time, adjusted to give days of equal length (i.e., 24 hours). There is, therefore, a difference between the time indicated for solar radiation and the one indicated for all other variables. The magnitude of this difference oscillates during a year. However, as shown in Appendix A, the magnitude of the difference varies between approximately -9 and + 21 minutes during the year for Montréal. Since the time difference is small, it was neglected and it was assumed that the time indicated for solar radiation in the AES records is local standard.

IV.6.3 Meteorological data disaggregation

For simulations related to greenhouse control, it is necessary to follow the dynamic behavior of a system over small time steps, e.g., over periods smaller than one minute. The instantaneous values of the meteorological conditions are therefore needed, which implies a disaggregation of the hourly values taken from AES. For variables other than solar radiation, this does not cause too much of a problem, and it was assumed that their values vary linearly during the hour from one data point to the other. For solar radiation, the instantaneous flux is needed rather than the total energy received hourly. In natural environments, the solar radiation intensity can vary widely during one hour and complex models can be developed to generate instantaneous solar fluxes. However, a simple approach was adopted in this project. The average flux was calculated from the total energy received for each hour, and this average was assumed to occur in the middle of the hour, i.e., 30 minutes before the end of the hour. Then, in a simulation, the instantaneous flux was found by linear interpolation between hourly averages. This approach is valid for all hours during the day, except in the hours when sunrise and sunset occur. For these two hours, the flux was set to zero at the sunrise and the sunset. The flux at each instant during the sunrise period was found by linearly interpolating between zero and the average value during the following hour. At sunset, it was interpolated between the average value at the preceding hour and zero. The same approach was used for both the direct and the diffuse components of solar radiation.

For the development of the input data files, sunrise and sunset hours were calculated using standard formulae in solar engineering. Sunrise and sunset hours are a function of the latitude of the station and the solar declination (Kreith and Kreider, 1978). A Fourier series was used to estimate the solar declination as a function of the day angle (MER, 1983).

IV.6.4 The meteorological data files

Using the procedure described in section IV.6.3, data files were prepared for three years, from 1982 to 1984. In the original AES files, some data were missing. The missing values were either for the global or diffuse components of the solar radiation. The number of missing values from 1982 to 1984 were respectively 98, 148 and 23, and were respectively distributed over 17, 18 and 5 days. Thus, during the worst year (1983), less than 1% of the solar radiation data was missing. During the same year, 60% of this data was for consecutive days and for the diffuse component. Many models exist to estimate the incoming solar radiation as a function of the other meteorological variables. One such model was described by Won (1977). However, instead of using a model, a simpler approach was adopted. For each day where data was missing, the values during the preceding and following days were visually analyzed, and the cloudiness was considered so as to generate values for these days. A more accurate method must be adopted for analysis of meteorological data (e.g., to do correlation studies or to validate meteorological models). However, the approach used was sufficient to fulfil the objectives of this research.

A sample of the meteorological data files that were created is shown in Figure IV.8. The chosen arrangement simplifies the treatment to be performed by the module WEATHER, which generates the meteorological conditions during functional simulations. For all variables except solar radiation (both the direct and the diffuse components), the value occurs at the hour indicated at the beginning of each record. This means that the values

Hour	Temperature	Relative humidity	Wind speed	Atmospheric pressure	Global solar radiation	Diffuse solar radiation	Cloudiness
1	-10.6	82.6	10.3	100.35	0	0	10
2	-9.4	84.0	8.6	100.15	0	0	10
3	-9.5	83.9	9.2	100.12	0	0	10
4	-9.3	84.1	8.3	99.91	0	0	10
5	-8.8	81.6	8.3	99.85	0	0	10
6	-10.1	80.0	7.2	99.85	0	0	10
7	-10.1	79.9	5.6	99.95	-12	-12	10
8	-10.2	79.8	3.1	100.15	5	5	10
9	-10.3	83.0	4.2	100.39	23	23	10
10	-9.5	83.9	4.2	100.49	79	79	10
11	-8.0	79.5	6.1	100.69	137	137	10
12	-8.1	79.4	11.4	100.86	218	218	10
13	-9.2	80.9	10.3	101.13	169	169	9
14	-9.4	77.4	10.3	101.43	233	233	8
15	-9.5	74.0	8.3	101.77	141	141	0
16	-10.0	69.7	8.3	102.01	49	49	0
17	-10.4	72.3	7.2	102.31	11	11	0
18	-10.3	69.0	7.2	102.41	-26	-25	0
19	-10.5	68.6	6.7	102.62	0	0	0
20	-10.6	68.3	6.7	102.78	0	0	0
21	-11.2	66.9	4.7	102.89	0	0	0
22	-10.5	68.5	4.2	103.02	0	0	0
23	-11.8	73.1	4.2	103.09	0	0	0
24	-12.1	73.0	4.1	103.10	0	0	0

Figure IV.8 Sample of a meteorological data file.

found in the original AES files were shifted negatively by one hour for all variables, except solar radiation. For solar radiation, the values represent the flux occurring 30 minutes before the hour indicated at the beginning of each record. For convenience, negative values for solar radiation are used at the beginning and the end of each day. At sunrise, the negative value occurs in the hour preceding the sunrise hour when sunrise occurs in the first 30 minutes of the hour. The negative value is in the sunrise hour when sunrise occurs in the last 30 minutes of the hour. The module WEATHER simply interpolates between this value and the following one. The zero value is reached exactly at the sunrise, which is 7:11 AM on February 2nd (see Figure IV.8). When the flux is negative, WEATHER sets the value to zero. The same approach is used during the sunset period, except that the procedure is inverted.

To verify that the adopted approach does not lead to large errors for solar radiation, the total energy received monthly in simulations was computed for the year 1982 and compared to the values calculated with AES data. Results are shown in Table IV.1. It can be observed that the adopted approach generally produces an underestimation and the difference varies between 0.3 and 1.4%. The daily pattern is also changed very slightly, as illustrated in Figure IV.9, which is representative of the pattern generally observed. Due to the interpolations, there is a certain flattening of the curve, e.g., the peaks are slightly lower. The differences at both ends of the day are small since irradiation at those moments is very small. It can be concluded that the adopted approach gives errors that are within an acceptable range and that the approach was appropriate for the needs of this research.

Table IV.1 Total solar energy received monthly with AES and simulated data.

Month	With AES data [MJ/m^2]	With simulated data [MJ/m^2]	Difference [%]
January	187.2	185.2	1.1
February	270.2	267.9	0.9
March	430.2	427.9	0.5
April	475.9	474.2	0.4
May	650.2	648.0	0.3
June	588.5	586.9	0.3
July	733.7	731.3	0.3
August	539.8	537.7	0.4
September	368.6	366.1	0.7
October	288.9	286.3	0.9
November	131.0	129.6	1.1
December	119.4	117.7	1.4

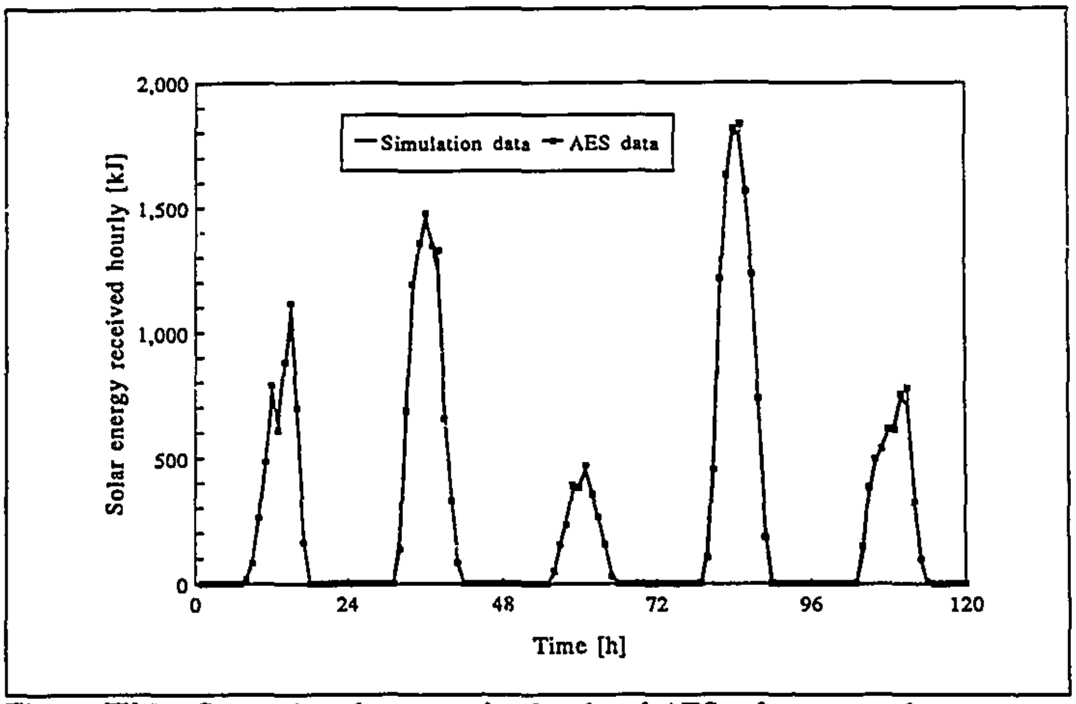

Figure IV.9 Comparison between simulated and AES solar energy data.

IV.7 DYNAMIC ANALYSIS OF THE FUNCTIONAL GREENHOUSE SYSTEM

Once all modules were developed, they were integrated in a common simulation structure within OS/2. Simulation experiments were performed with this structure to study the behavior of the complete functional greenhouse system, in absence of the cognitive controller. The specific objectives were 1) to verify that the simulation results are reasonable and free of aberrations, 2) to compare the results with values found in the literature and 3) for the author to become familiar with the system behavior under various meteorological conditions. A number of variables were considered in this analysis and particular attention was paid to the inside air temperature (Air 1), the biomass production rate and the heating load. In all simulations, the day temperature setpoint was maintained constant at 21°C, while the night temperature setpoint was 17°C; these values correspond to those used in a production context (CPVQ, 1984). The setpoint for ventilation was kept constant at 25°C (see Figure IV.5). The simulations were done for all months of 1982 using meteorological data from the Canadian Atmospheric Environment Service (AES) as weather input. The procedure used and the results are presented in detail in Appendix D.

In general, it was concluded that the models and the functional simulation furnished results that were reasonable and realistic. The functional system responded well to disturbances. However, future improvements could certainly be made at many levels, principally in the calculation of moisture balances, of the infiltration rates and of the sky temperature. Some mechanisms for the regulation of the humidity might also be added to the control system, because of the influence of humidity on disease occurrence and nutrient assimilation. However, it was concluded that the functional system as it was conceived and constructed, constituted an excellent tool for the development of complex control systems.

IV.8 CONCLUSION

The functional greenhouse system was described as being formed by four subsystems: a greenhouse, a crop, a Pavlovian controller and a cognitive controller. Inputs to the system are mainly related to meteorological conditions. The system constitutes a deterministic representation of some greenhouse. A prototype cognitive controller was developed for it. The different modules were installed in a common simulation structure under OS/2, which is discussed in Chapter V.

V. THE SIMULATION STRUCTURE AND ITS FUNCTIONING UNDER OS/2

V.1 INTRODUCTION

In Chapter IV, the modules of the functional greenhouse system which were to be included in the simulation structure were described. In physical reality, the processes that these modules represent would all function in parallel. For example, the controllers would continuously do calculations and symbolic treatment, while interacting with the greenhouse environment. Also, the environment and the crop will constantly influence each other. However, if the simulation of these processes is to be implemented on only one microcomputer, the modules cannot be executed simultaneously. It is, nevertheless, possible to use some features of a multi-tasking operating system such as OS/2 to obtain a good representation of the parallelism inherent in physical reality. OS/2 allows the concurrent execution of many processes, while permitting inter-process communication and synchronization. It therefore constitutes a convenient environment for the simulation of a complex arrangement of controlled and controlling components.

One of the objectives of this research was to investigate how a multitasking operating system can be exploited to implement a simulation structure comprised of independent components, which differ in their roles and in the computer languages in which they are written. The use of OS/2 to implement the functional simulation will be discussed in this chapter. The content of several sections of this chapter has been reported in a paper by Lacroix and Kok (1991b).

V.2 SIMULATION CHARACTERISTICS

V.2.1 Simulation modules

The inputs and greenhouse components that were modelled and described in Chapter IV were combined into a single system called the "functional simulation structure". The functional simulation structure consists of six inter-linked modules (Figure V.1): MANAGER, WEATHER, GHOUSE, CROP, PAVLOV and COGNITI. The simulation manager (MANAGER), which has not been described previously, coordinates the activities of the functional simulation and produces functional time. The WEATHER module generates the present meteorological conditions. GHOUSE contains the greenhouse model and CROP contains the crop model. PAVLOV and COGNITI contain the Pavlovian and cognitive controllers respectively.

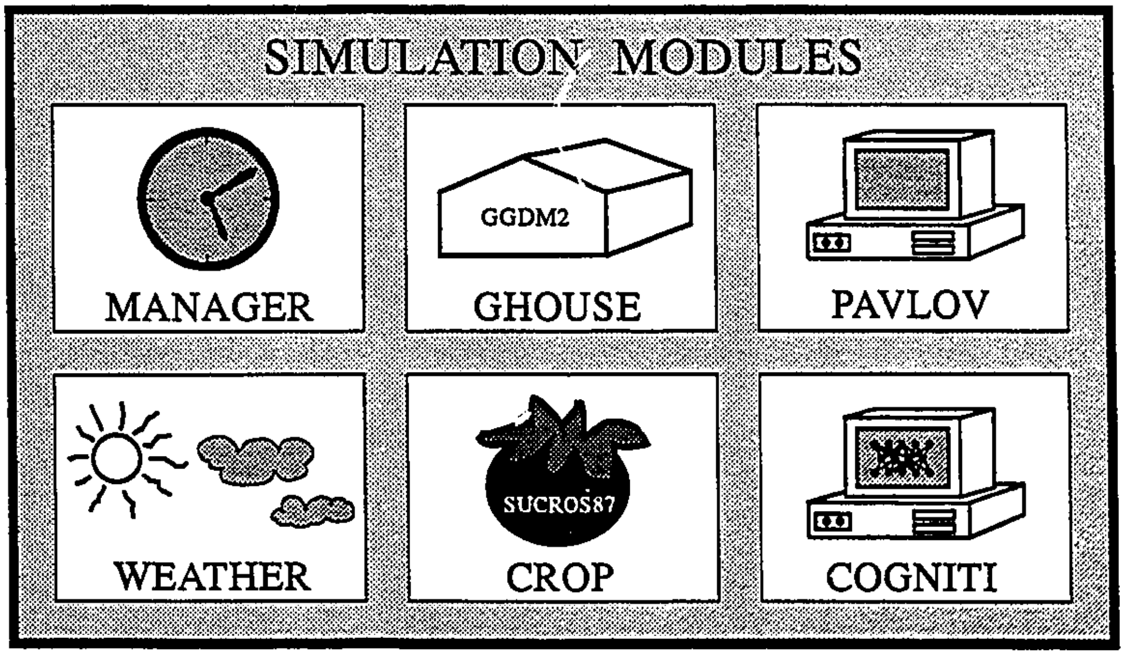

Figure V.1 Modules composing the functional simulation structure.

MANAGER, PAVLOV and WEATHER were all written in Microsoft BASIC 7.0 (Microsoft Corporation, 1989). The sections of interest of GGDM2 and SUCROS87 had been written in FORTRAN. They were maintained in this language, but restructured as subroutines callable from BASIC 7.0. Then, the main programs for GHOUSE and CROP were written in BASIC 7.0 and formed the entry ports into GHOUSE and CROP which called these subroutines. The FORTRAN subroutines were compiled with Microsoft FORTRAN 5.1 (Microsoft Corporation, 1991), and linked to the main programs using mixed-language programming features (Microsoft Corporation, 1989). The cognitive controller in COGNITI was written using GURU 3.0 (Micro Data Base Systems, 1991). GURU is an integrated product in which are combined several software components which can share data. An expert system shell, a relational data base, a procedural language, a text processor and a spreadsheet are available. The subjective-level (neural) model was constructed with NeuralWorks Professional II/Plus (NeuralWare, 1991). Once the neural network was adequately trained, it was saved as a C source file which was then compiled and linked with Microsoft C 6.0 (Microsoft Corporation, 1990) as a function callable from GURU 3.0.

V.2.2 Simulation implementation alternatives

The functional simulation structure contains modules which are diverse both in their roles and the software with which they were constructed. Several alternatives existed for implementing these modules. A first possibility was to install the modules on separate computers. Kozai *et al.* (1985) adopted this approach to test greenhouse control computers by interfacing them with a computer running a greenhouse climate simulation. A second method consisted of installing all modules on one computer, within one process, as has been done traditionally in simulation. A third approach was to implement the modules as different processes on one computer.

For a physical implementation of the control approach described in the conceptual framework, one would probably install the Pavlovian and cognitive controllers on separate computers. The latter might even need to be installed in a distributed fashion on a number of specialized hardware components. However, the installation of the entire set of modules on a single computer had the major advantage that the communication channels and protocols between elements were relatively simple to establish and that inter-module communication was fast. Also, with this approach, system failures due to hardware problems were less likely to occur than with a set of cabled computers. Also, all modules could be controlled from a single workstation, i.e., with one keyboard, mouse and screen. At the same time, this approach reduced the required amount of equipment. For these reasons, this approach was selected.

Although the entire simulation was to be installed on one computer, it was desired to preserve as much modularity as possible. Generally, this simplified the development of individual modules and allowed for the substitution and testing of different programs within each module, e.g., to test and compare various versions of the cognitive controller. It was also necessary to adopt an approach permitting the use of different programming languages within the same structure. For these reasons, it was decided to explore the potential of implementing the simulation modules as different processes running concurrently under OS/2.

V.3 TIME CONSIDERATIONS

V.3.1 Physical, functional and subjective time

Within the simulation structure, data has to be exchanged between processes. Because this data must refer to the same time frame (in this instance, functional time), interprocess synchronization must be maintained. To fulfil this condition, it is essential to first define and examine some concepts related to time. Time perceived by humans within their physical reality is referred to as "physical time". When operating in a simulated framework, a different type of time exists: this is simulated time. In the case of the functional simulation, it is called "functional time" and it is generated by the MANAGER module. A third type of time is associated with the cognitive controller, which, as part of its activities, has recourse to simulation. This time is called "subjective time".

Functional time can be defined with respect to physical time in two ways. The first reference is static and chronological; it is the physical period of time for which the simulation is run. One advantage of simulation is that this period can be situated anywhere along the physical time axis: in the past, present or future. The second reference is dynamic and is the rate at which functional time flows with respect to physical time. This is called the functional time flow rate. An advantage of simulation is that this rate can be adjusted. Traditionally, simulation has been used to examine rapidly how a system would behave over a long period. This is achieved by maintaining a high functional time flow rate, which is possible because of the simplified representation of the system in a limited model. For example, by using simple models, it is possible to simulate, in a few days, the behavior of the earth's atmosphere during many years. Conversely, simulation can allow the slowing down of the functional time flow in relation to physical time, which can be advantageous on some occasions. For example, this approach can be used to simulate physical processes that happen very rapidly in physical reality, allowing close observation of some of the reactions involved (e.g., collisions between two bodies).

The slowing down of functional time flow can be useful in the simulation of control systems for physical greenhouses, or for more complex agro-ecosystems. If such control systems are to be effective, they must be able to make and carry out decisions rapidly. Although the computational requirements for real-time artificially cognitive control are unknown, they are certainly enormous and beyond the abilities of today's hardware. In a simulated framework, one can effectively multiply the computing capacity of the controller by slowing down functional time, i.e., by taking longer in physical time than what is being simulated. The rate of functional time flow can also vary during a simulation and can even be stopped completely, yielding a computing machine of infinite capacity. Such an approach can be used to simulate cognitive control systems where the required computing activity is tremendous.

Subjective time is related to functional time in a manner similar to that in which functional time is related to physical time. The same considerations therefore apply. Subjective simulations can be started anywhere in the past or future of functional time and can flow at any rate. One must remember, however, that this flow rate strongly influences the computing demand of the simulated controller, which will in turn affect the computing demand of the functional simulation.

V.3.2 Functional time flow rate

In simulation on a digital computer, functional time must flow in discrete increments. The size of these increments, together with the complexity of the simulation modules and the capacity of the physical machine, determines the functional time flow rate. The use of large increments is desirable because it allows simulation over a substantial period to be done in a short duration of physical time. However, the increments should be small enough to ensure that the simulation is adequately representative of the aspects of reality that are modelled. Not all simulation modules have to function with the same time increment but, to keep the modules synchronized, all increments should be integer multiples of the smallest one.

The greenhouse model contained in the GHOUSE module consists of a set of differential equations. During a simulation, these equations are numerically integrated with respect to time. An increment of 15 seconds is small enough to avoid overshooting in the heating and ventilation systems and to assure numerical stability with this model. Because the greenhouse reacts directly to physical inputs and effector settings, the weather generator module (WEATHER) and the Pavlovian controller module (PAVLOV) need to operate with the same increment. For the crop model, for average climatic conditions, it might be sufficient to calculate changes in the biomass every hour. To allow the concentration of CO_2 inside the greenhouse to be controlled, CO_2 balances will eventually be implemented in the GHOUSE module. In this case, since the resident mass of gaseous CO_2 in the greenhouse atmosphere is fairly small and can change rather quickly, the rate of CO_2 consumption or production will have to be calculated about as often as the greenhouse climate. As for the cognitive controller, its role is to generate temperature setpoints for the regulation performed by the Pavlovian controller. The time increment for the COGNITI module can therefore be larger than that of the PAVLOV module.

From the previous considerations, it was concluded that a functional time increment of 15 seconds was sufficient, and MANAGER generated time on this basis. All activities done during each time increment are referred to as forming a "simulation cycle". Four modules had to communicate during each simulation cycle to exchange data, i.e., WEATHER, GHOUSE, CROP and PAVLOV. The size of the increment for the COGNITI module was established separately.

V.3.3 The cognitive controller and time

A cognitive controller of the type defined in the conceptual framework (Chapter III), would be running many complex decision-making activities in parallel and would require enormous computing capacity. Unlike the Pavlovian controller, the behavior of such a controller would be unpredictable and its activities would not follow a cycle. Many reasoning processes would run in parallel, some would come to an end while others would continue. At the same time, new processes would continuously be initiated. Theoretically, some deliberations could have limitless duration. This would produce the impression of a multiple stream sequence of decision-making that could, theoretically, continue for infinity. Such a machine could be simulated with the "infinite computing capacity" approach, by which functional time would stop to let the cognitive activities execute. However, for investigations to be done over a functional period of a sufficient length (e.g., over a growing season) and within reasonable physical time duration, one cannot stop functional time indefinitely. One way to maintain the effect of an ongoing decision-making sequence while letting functional time flow is to: 1) suspend the cognitive activities at some point in each functional simulation cycle, next, 2) increment functional time and execute the other modules and then 3) come back to the cognitive activities and pursue these for a further (physical) duration. The implementation of such functionality requires special procedures and an approach is suggested for this in Chapter XI.

The cognitive controller that was implemented in this research was less complex than that described in the conceptual framework, and thus required a similarly less complex simulation structure. Its functioning was based on sequences of decision-making steps of finite size. Whenever it was activated, a temperature setpoint was generated for the Pavlovian controller. Decision-making activities of the cognitive controller varied over a functional day from basic interpolations to more complex simulation-based determinations, but when the COGNITI process was activated, all its operations were completed first before the functional simulation passed to a new time increment. This was, therefore, based on the "infinite computing capacity" approach. With this approach, the increment for the COGNITI process must be kept sufficiently large because, when it is executed, subjective time flows while functional time stops. The smaller the increment for the cognitive controller, the more often functional time would be stopped and the longer the period of physical time that the experimenter must wait for results. For the control strategy used in this research, an increment of five minutes was found to be adequate, i.e., COGNITI was activated every five minutes of functional time.

Figure V.2 illustrates how time evolves according to the three different frames, that are orthogonal to each other. Functional time is shown to be increasing in equal increments,

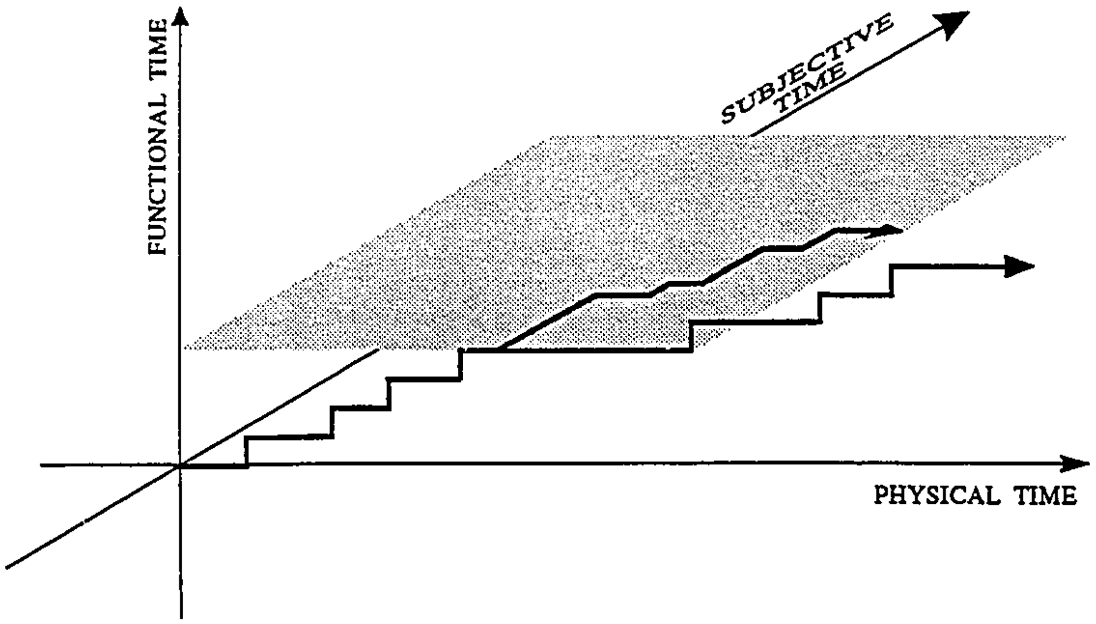

Figure V.2 Representation of the different time flows.

but the physical time duration required to deal with each increment is not necessarily the same. Rather, it depends on the amount of work to be done by the various modules. The flow of subjective time is illustrated for one occurrence. This happens only when the cognitive controller is activated, which occurred maximally with a frequency of once every five minutes in this research (or once per 20 functional time increments).

V.4 IMPLEMENTATION UNDER OS/2

V.4.1 Functioning of OS/2

The multitasking operating system OS/2 permits the concurrent execution of many programs. In this context, there exist many possibilities for the implementation of the functional simulation structure. In this research, it was desired to develop an approach that would be as flexible as possible, so as to allow for the eventuality of more complex situations and for further elaboration. To be able to achieve this goal, it was important to have a basic understanding of how OS/2 works.

V.4.1.1 Threads, processes and sessions

The functioning of OS/2 is based on threads, processes and sessions. A "thread" is defined as a series of instructions to be executed by the microprocessor (CPU). Threads are grouped into units called "processes", which are the owners of the system resources such as memory, files, semaphores, etc. The processes are further grouped into "sessions", in which they share a virtual console composed of a keyboard, a mouse and a screen (Dror, 1988). Many sessions can be created at the same time, each consisting of one or more processes. Within each process, one or more threads can in turn be created for processing by the CPU. When a user initiates the concurrent execution of many programs within OS/2, the operating system creates different processes each comprising at least one thread. The processes can be part of one session or of separate sessions. To simplify the discussion below, a distinction will not be made between "process" and "thread". It will be assumed that there is only one thread per process, and only the term "process" will be used.

In a single-tasking operating system, only one process exists and is executed at a time. In a multi-tasking system such as OS/2, many processes can be executed concurrently. Since OS/2 must presently be installed on machines possessing only one physical processor, the CPU must be switched from one process to the other in order for all of the concurrent programs to progress. In OS/2 terminology, all processes that are waiting to be executed are said to be "dispatchable", while the process that is currently being executed is said to be "dispatched".

The switching of the CPU between the processes is controlled by a component called the "scheduler". The scheduler determines which of the dispatchable processes will actually be dispatched. This is done with a "priority, round-robin" algorithm (Dror, 1988). Each process running under OS/2 is assigned a dispatching priority. There are three classes of dispatching priority: idle-time class, regular class and time-critical class. These classes are each subdivided into 32 levels. The processes that are part of the lowest priority class (idle-time class) are dispatched only when the processor has nothing else to do. In contrast, processes in the time-critical class will always be dispatched first. This includes many processes that belong to the operating system or that control the user interface. At an intermediate level, there is the regular class. When a user executes many programs concurrently, most processes that are created are put into the regular priority class.

Within each class, each process is dispatched for only a short physical duration called a "time slice". The processes are dispatched one after the other. The order in which the processes are dispatched and the length of the time slice are determined by the scheduler, which dynamically changes the priority level of all processes in consideration of many factors, such as the total memory used by each process, the number of I/O operations the processes require, the number of processes that are dispatchable at the same time, etc. A process running in the foreground (i.e., a process that accesses the physical console) will usually have its priority level increased, compared to processes running in the background. Also, if a process has not been dispatched for some time, OS/2 will temporarily increase its priority level. The minimum and maximum length of the time slices can be adjusted by a user with the system variable TIMESLICE in the system configuration file (CONFIG.SYS). The maximum duration for which processes must wait before being dispatched is controlled with the system variable MAXWAIT. Finally, dynamic priority changes by OS/2 can be enabled or disabled by modifying the value of the system variable PRIORITY.

V.4.1.2 OS/2 Application Program Interfaces

OS/2 version 1.3 was used for this research. This version contains functions that allow a user to request some services from OS/2 and to keep some control over how it operates. These functions are called "Application Program Interfaces (API)". More than 200 API's are provided which can be used to create multitasking programs, manage memory and files, and control input/output operations (e.g., screen, mouse, keyboard). For the functional simulation, API's were used to create child processes, for interprocess communication and synchronization, signal-handling, stopping processes and for doing minor tasks such as sound generation (beep signals). These functions can be called, with a list of arguments, in traditional computer languages such as C or BASIC. In the following discussion, the arguments accompanying the API calls will generally not be given. To obtain more information on how to use OS/2 API's, one may refer to Dror (1988) or Lacobucci (1988).

V.4.2 Process control in the functional simulation

Within a session a process can create and control other processes. The process that creates other processes is called the "parent" process; the created processes are called "child" processes. It is also possible for a process of one session to create other sessions comprising their own processes. However, the process of the original session is not the parent of the processes in the newly created sessions, and processes that are part of different sessions are more difficult to synchronize. For this reason, it was decided to use only one session for the execution of the functional simulation. This had an additional advantage in that the evolution of all modules during simulations could be followed on one screen in a full-screen mode. Each simulation module was constructed as a different process, and the process containing MANAGER was the parent process of all the others. Even if all processes were part of the same session, the modules appeared to each other to be running on separate machines.

To keep the various processes synchronized, MANAGER must govern which of the modules are to be executed during a simulation cycle and how the processes that contain them are executed. Two major approaches to process execution exist: parallel and sequential. If all processes eligible for execution during a cycle are executed in parallel (parallel approach), the OS/2 operating system must decide when each should be dispatched by means of its round-robin algorithm. In this case, if the elements are to communicate during the time increment, process synchronization can become rather complex. On the other hand, if MANAGER governs the sequence of execution (sequential approach), then synchronization can be more easily realised and interprocess communication is simplified.

For the implementation of the simulation, the sequential approach was chosen. The functional time increment was kept constant throughout the simulation and the four modules WEATHER, GHOUSE, CROP and PAVLOV were executed sequentially every cycle by MANAGER. The COGNITI process was executed when necessary; for convenience, the dispatch of its process was governed by PAVLOV. The execution flow for the overall simulation is illustrated in Figure V.3. The MANAGER module, which acts as the coordinator of the overall structure, first increments the functional time. Then, WEATHER generates the meteorological conditions prevailing during this time step. Next, GHOUSE calculates the greenhouse state at the end of the increment, and the crop growth is established from these conditions by CROP. The PAVLOV module reacts to the internal conditions and fixes the status of the heater and the ventilators for the next time increment. COGNITI is executed once every five (functional) minutes, to establish temperature setpoints.

V.4.3 Interprocess communication

Many tools, such as queues, pipes and shared memory, are available for interprocess communication within OS/2. Several of these tools were considered, but the use of "shared memory segments" (SMS's) was found to be the most convenient for the

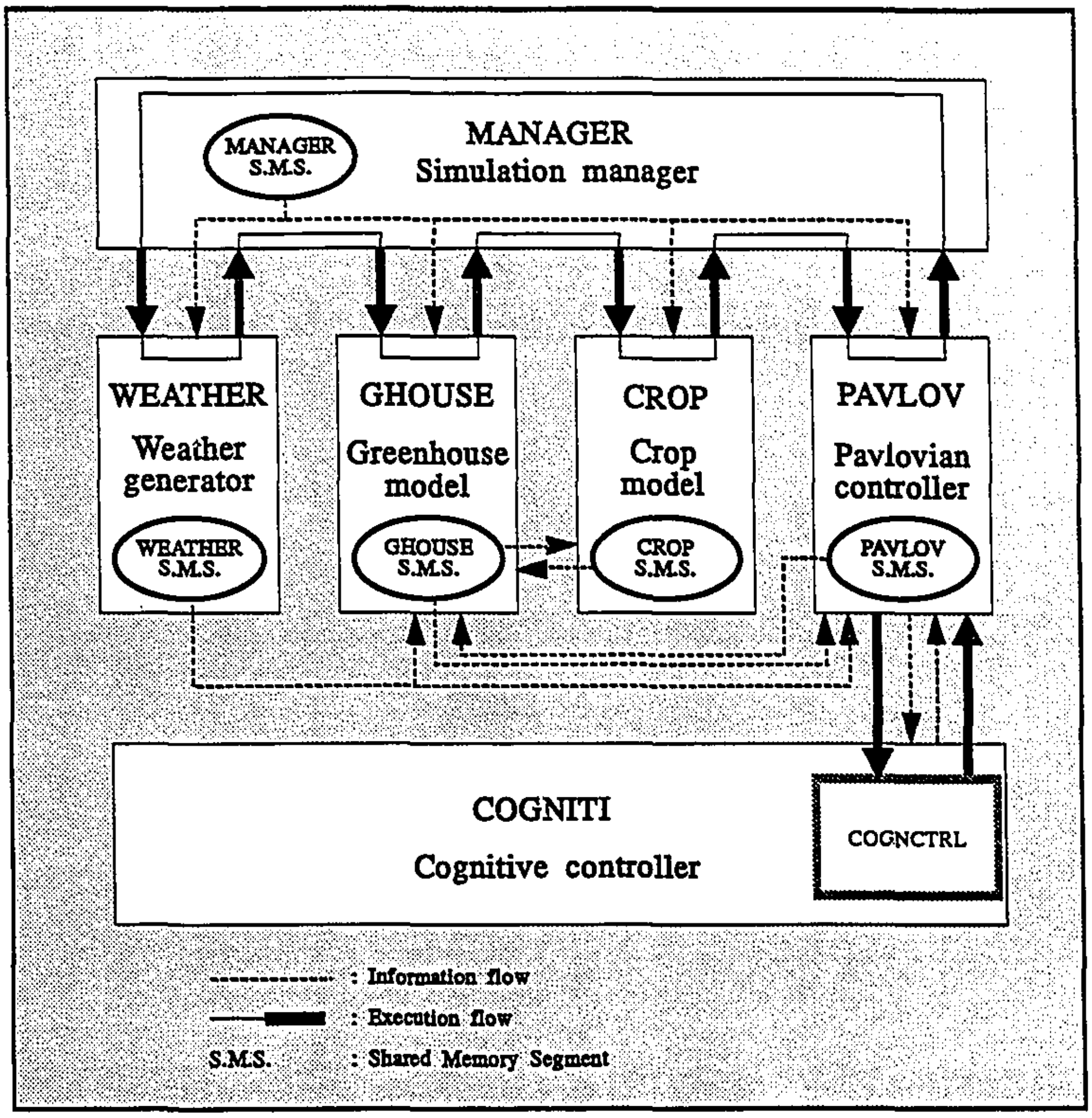

Figure V.3 Execution and information flow in the functional simulation.

functional simulation. The routes for inter-process communication occurring through SMS's are shown in Figure V.3. Five SMS's are used, one per module, with the exception of COGNITI. COGNITI cannot directly access SMS's because the programming language integrated in GURU 3.0 cannot be used to call OS/2 API's. Communication between PAVLOV and this module therefore takes place via text files on a virtual disk.

Each SMS is defined by the parent process (MANAGER) in a call to the API DosAllocShrSeg(), during which the size of the SMS is specified. The number of bytes depends on the number of values that must be transmitted to the other modules. Each child process can access SMS's with the API DosGetShrSeg(). When a process first gains access to an SMS, it obtains from OS/2 a segment selector that can be used to address the SMS in subsequent operations. The process can then write or read a series of bytes that starts at the memory address corresponding to the segment selector. In BASIC, reading and writing are done respectively with the PEEK and POKE statements.

In the functional simulation, values of both numerical and string variables are transferred from one process to the other via SMS's. Since only series of bytes can be read or written, a mechanism must be used to transform numerical values into series of bytes and vice-versa. For this, a simple approach was used, in which numerical values are first transformed into strings, and the characters of the strings are then written one after the other one into the memory segment. The reverse procedure is used to read data from an SMS. Each string written in an SMS is preceded by a byte which indicates the number of bytes the string occupies. When a process wants to read a certain string in an SMS, it then knows how many bytes it has to peek for each variable. This procedure is necessary, because the transformation of a numerical value into a string does not always lead to the same string length in BASIC.

The adopted procedure is not very efficient because, when a numerical value is transformed into a string, a larger memory space is required to contain it. For example, in BASIC, a single precision real number occupies four bytes in memory, but, as a string, it may need up to nine bytes. As well, when large amounts of data must be accessed by the different processes, the speed of transfer is decreased. However, due to the relatively small amount of data transferred between modules in the present simulation, this simple procedure was considered to be sufficient. Note that the management of the shared memory segments is facilitated by the fact that the processes are executed sequentially.

In this case, one process completes its cycle and then transfers all the necessary information to the other processes using shared memory. There is no need to tell the other processes that the sequence of information at a certain address is being changed and must not be read at that particular moment, as would be the case if the processes were executed in parallel.

The contents of the shared memory segments used in this project are illustrated in Table V.1. For example, MANAGER writes values for two variables into its SMS, ActualDate$ and ActualTime$, that respectively represent the current functional date and the current functional time. The size allocated for each SMS is between 50 and 200 bytes. These sizes can be modified easily if more variables have to be transferred between the modules in the future. The contents of the data files required for intercommunication between PAVLOV and COGNITI are shown in Table V.2. The name of the file created by PAVLOV is called "COGNITI.IN"; the name "COGNITI.OUT" is used for the file created by COGNITI. The values stored in COGNITI.IN represent the average conditions that occurred between calls to COGNITI, i.e., in this investigation the average conditions during five minutes.

V.4.4 Interprocess synchronization

As previously explained, it was decided that the modules should be activated sequentially by MANAGER, i.e., in a functional simulation cycle, each module must execute all its activities before the next is activated. At the same time, it was advantageous to keep the processes memory-resident in order to avoid having to re-execute at each cycle all shutdown and start-up operations (e.g., variables saving and loading, initial calculations). It was therefore decided that the child processes would be executed *asynchronously* (in OS/2 terminology) by MANAGER, i.e., when MANAGER initiated the execution of a child process, it would not wait for the child process to finish its execution before pursuing other activities. This implies that all processes would be concurrently dispatchable, and that a mechanism would be necessary for MANAGER to keep control of the dispatch of the various processes. This mechanism would be able to suspend the activities of a process, activate one or more other processes, and then come back to the initial process and reactivate it. Without such a mechanism, the dispatch of the processes would be entirely under the control of OS/2.

Table V.1 Content of the shared memory segments.

PROCESS	VARIABLE NAME	VARIABLE DESCRIPTION
MANAGER	ActualDate$	Current functional date
	ActualTime$	Current functional time
WEATHER	TempOut	Outside temperature
	RelHum	Outside relative humidity
	WindVel	Wind speed
	AtmPress	Atmospheric pressure
	GlobRad	Global solar flux
	DiffRad	Diffuse solar flux
	CloudOpac	Cloud opacity
	TSky	Sky temperature
GHOUSE	TempIn	Inside air temperature
	RelHumIn	Inside relative humidity
	CO2In	Inside CO_2 concentration
	TempLeaf	Leaf temperature
	TrDirRad	Transmitted direct solar radiation
	TrDiffRad	Transmitted diffuse solar radiation
	HauSol	Solar height
CROP	NetCO2Uptake	Net CO_2 uptake by crops
PAVLOV	FanJet	State of the small fan ventilator (OFF/ON)
	Ventilator	State of the large fan ventilator (OFF/LOW/HIGH)
	Heater	State of the heater (OFF/ON)

Table V.2 Content of the data files COGNITI.IN and COGNITI.OUT.

PROCESS	VARIABLE NAME	VARIABLE DESCRIPTION
PAVLOV	ActualDate$	Current functional date
	HourActTime$	Current functional time
	AvgTempOut	Average outside temperature
	AvgWindVel	Average wind speed
	AvgGlobRad	Average global solar flux
	AvgTempIn	Average inside air temperature
COGNITI	HeatSpt	Temperature setpoint for the heating system
	VentSpt	Temperature setpoint for the ventilation system

Special tools are available within OS/2 that allow one to keep a certain amount of control over the dispatch of processes. For example, it is possible to disable the capacity of OS/2 to dynamically change the priority levels of processes; with API's it is then possible to control from one process the priority levels of various other processes, and to continuously decide which one will be dispatched. Other tools in OS/2 allow one to suspend the execution of a process until a certain event happens. This is called putting a process in a "nondispatchable state". For example, the process can be made to wait until a semaphore is cleared before continuing its execution. A process can also be made to wait for a certain time interval before its execution is pursued. Various options were examined, and, in this research, semaphores were found to be preferable to other mechanisms.

The general procedure that was used to sequentially activate the processes is illustrated in Figure V.4. The approach is based on the transfer of the processing activity between different processes during a simulation cycle. In this example, the processing activity is transferred between only two processes (PARENT and CHILD), where CHILD is the child process of PARENT. The same method can be used for three or more processes.

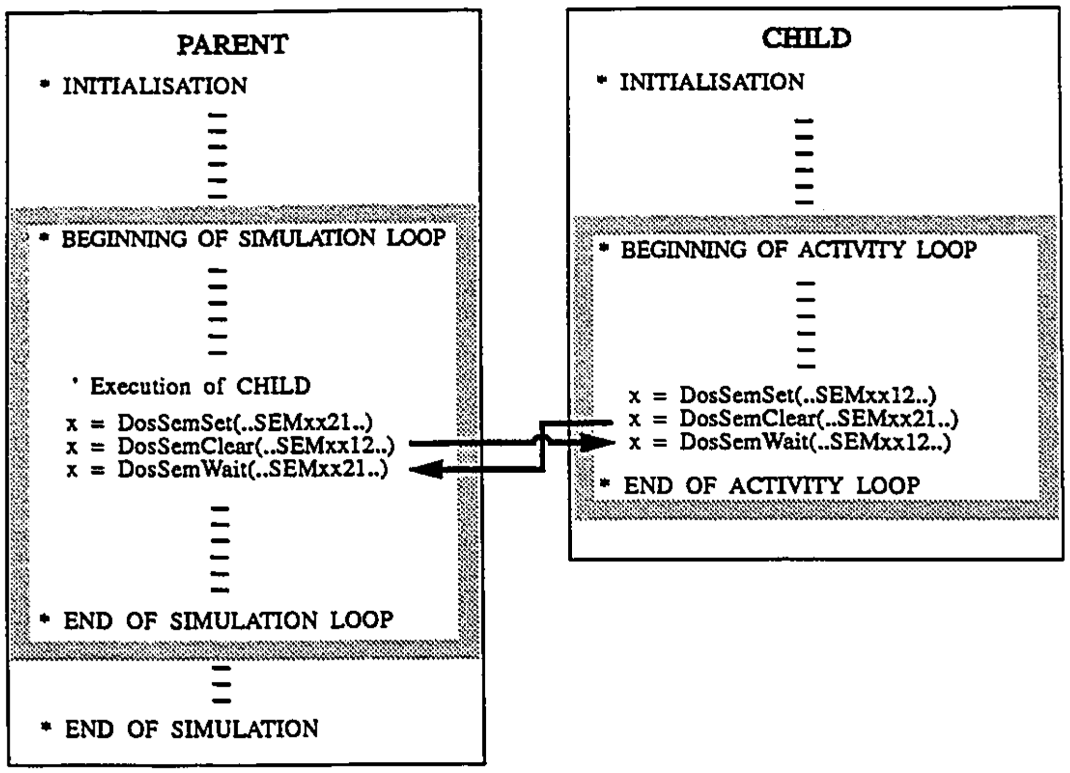
PARENT
* INITIALISATION
* BEGINNING OF SIMULATION LOOP
* Execution of CHILD
x = DosSemSet(..SEMxx21..)
x = DosSemClear(..SEMxx12..)
x = DosSemWait(..SEMxx21..)
* END OF SIMULATION LOOP
* END OF SIMULATION
CHILD
* INITIALISATION
* BEGINNING OF ACTIVITY LOOP
x = DosSemSet(..SEMxx12..)
x = DosSemClear(..SEMxx21..)
x = DosSemWait(..SEMxx12..)
* END OF ACTIVITY LOOP

Figure V.4 Approach used for the execution of child processes in the functional simulation.

The processes PARENT and CHILD are being executed in the regular priority class, and OS/2 allocates the computing resources using the round-robin algorithm, as was explained in Section V.4.1.1. However, for the simulation, the control of the activity transfer is maintained with the use of two semaphores for each child process. In Figure V.4, the semaphores are called SEMxx12 and SEMxx21.

Suppose that PARENT is in the simulation loop (within the gray limits in Figure V.4). Suppose also that CHILD is in the activity loop, i.e., in the loop of activities that are executed during each functional simulation cycle. Suppose also that only the process PARENT is presently dispatchable. Its current dispatch is, therefore, under the control of OS/2. During the same time, CHILD is in a non-dispatchable state. The semaphore

SEMxx12 is 'set' and CHILD waits for it to become 'cleared' before continuing its execution; it is waiting at the instruction:

```
X = DosSemWait(..SEMxx12..)
```

After a certain time, PARENT completes a simulation loop and comes back to the point where it must activate CHILD, which will in turn complete an activity loop. PARENT sets the semaphore SEMxx21, clears SEMxx12 and begins its wait for SEMxx21 to be cleared before continuing. This is done via the following series of calls:

```
X = DosSemSet(..SEMxx21..)
X = DosSemClear(..SEMxx12..)
X = DosSemWait(..SEMxx21..)
```

PARENT is now in a non-dispatchable state and, since SEMxx12 has been cleared, CHILD becomes dispatchable. Thus, the execution of CHILD comes under the control of OS/2. When it is eventually dispatched, it pursues its activities and completes a loop, after which control has to be transferred back to PARENT. The transfer is accomplished using the same approach as before, i.e., at the end of its loop, CHILD sets SEMxx12, clears SEMxx21 and begins its wait for SEMxx12 to become cleared before continuing. As a result, PARENT becomes dispatchable again and completes another loop, and so on. The whole simulation progresses, during which processing activity is continuously transferred from one process to the other using semaphores.

The adopted procedure is safe and robust, even if, for the very short period during which the transfer occurs between the two processes, both are in a dispatchable state and their dispatch is under the exclusive control of the operating system. For example, in Figure V.4, it is theoretically possible that CHILD be dispatched by OS/2 between the calls by PARENT to DosSemClear(..SEMxx12..) and DosSemWait(..SEMxx21..). In such a case, the semaphore SEMxx21 will be cleared by CHILD and the call to DosSemWait(..SEMxx21..) will have no effect. PARENT will simply continue its execution. A similar scenario might occur when control is transferred from CHILD to PARENT.

Note that for the chosen approach to work safely during a simulation cycle, the parent process must use the complete sequence DosSemSet(..SEMxx21..), DosSem-Clear(..SEMxx12..) and DosSemWait(..SEMxx21..), and the child process must use the equivalent sequence. For example, one might be tempted to use the API DosSemSetWait to skip one step. This API is used by a process to set a semaphore and then to begin a wait for it to be cleared by another process. For example, in Figure V.5, PARENT makes

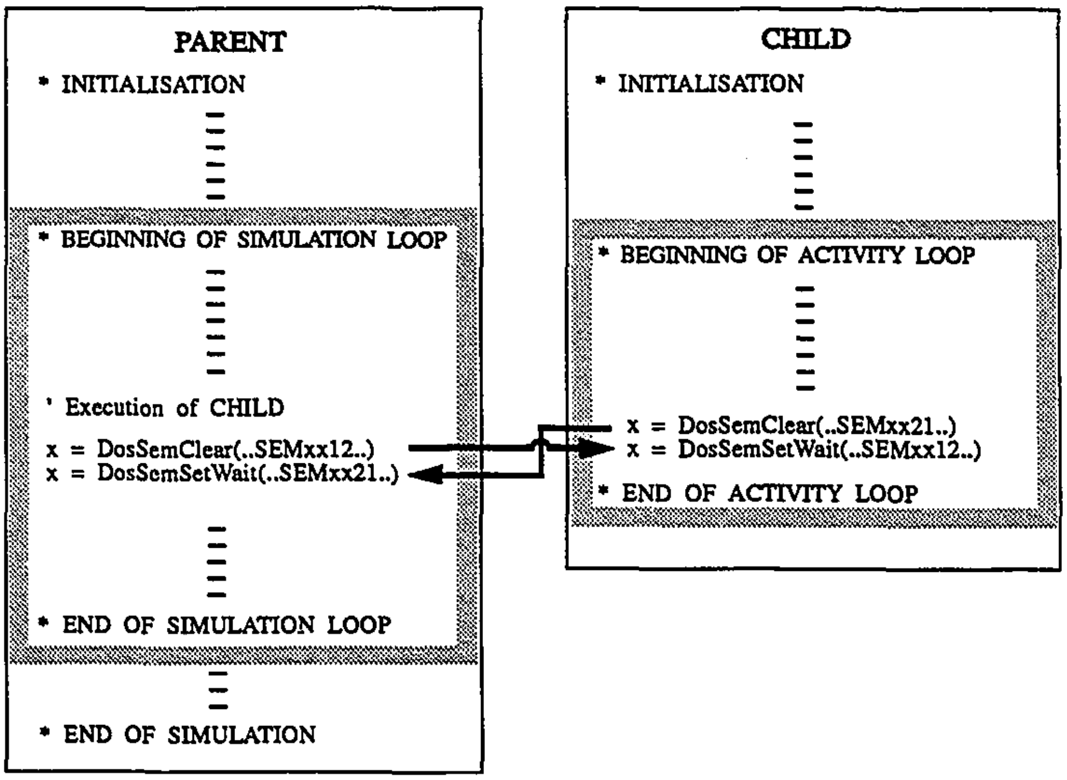
PARENT
* INITIALISATION
* BEGINNING OF SIMULATION LOOP
* Execution of CHILD
x = DosSemClear(..SEMxx12..)
x = DosSemSetWait(..SEMxx21..)
* END OF SIMULATION LOOP
* END OF SIMULATION
CHILD
* INITIALISATION
* BEGINNING OF ACTIVITY LOOP
x = DosSemClear(..SEMxx21..)
x = DosSemSetWait(..SEMxx12..)
* END OF ACTIVITY LOOP

Figure V.5 Execution of child processes using DosSemSetWait.

a sequential call to DosSemClear(..SEMxx12..) and DosSemSetWait(..SEMxx21..). During these two function calls, both processes are simultaneously in a dispatchable state. Thus, the child process might possibly be dispatched by OS/2 during the two calls because of the way the round-robin algorithm works. In this case, the child would execute a complete simulation loop and come back to the control sequence; it would then clear the semaphore SEMxx21, and set the semaphore SEMxx12 and wait for it to be cleared before continuing (using DosSemSetWait). The operating system would, at some later point, dispatch the parent process, which would set the semaphore SEMxx21 and wait for it to be cleared before continuing (also using DosSetSemWait). This would then result in the freezing of the simulation, where each process would wait indefinitely for the other process to clear the semaphores SEMxx21 and SEMxx12. A similar situation might occur when control is transferred in the other way, i.e., from CHILD to PARENT.

For the functional simulation, the same procedure as used above with PARENT and CHILD was used to synchronize the activities between MANAGER and all its direct child processes. Figure V.6 illustrates the arrangement that was adopted. Four sets of semaphores are defined (two for each of the four direct child processes) and the program sections involving their use are shown for MANAGER and WEATHER; the correspon-ding sections in GHOUSE, CROP and PAVLOV were organized in a similar fashion. In this scheme, PAVLOV runs COGNITI as a child process; it keeps it under control by using the procedure shown in Figure V.7, via an executable file written in BASIC (COGNCTRL.EXE) and a fifth set of semaphores. This special procedure is necessary, because, as previously mentioned, GURU's programming language cannot be used to call API's. It can however be used to execute an external file which, in this case, manipulates the semaphores.

V.5 EXECUTION OF THE FUNCTIONAL SIMULATION

V.5.1 Initialization phase

A functional simulation starts in the MANAGER process with an initialization phase, at the beginning of which the simulation parameters are read (e.g., beginning and end times of the simulation, functional time increment, report frequency). Next, the SMS's and semaphores are defined using the API's DosAllocShrSeg() and DosCreateSem(). The four child processes, WEATHER, CROP, GHOUSE and PAVLOV, are then created in turn by MANAGER, and each passes through an initialization phase. The sequence of API calls used for this is shown for WEATHER in Figure V.6. The procedure used is similar to that described in section V.4.4. MANAGER first sets the semaphore SEMMW21 via DosSemSet(). Using a call to DosExecPgm(), it then executes WEATHER *asynchronously*, i.e., the new process is concurrently dispatchable with the parent process. MANAGER continues and executes the statement DosSemWait() which is the instruction to wait for the semaphore SEMMW21 to be cleared before continuing. Concurrently, WEATHER executes its initialization phase, during which it reads its initialization parameters and gains access to both its required SMS's via DosGetShrSeg() and to the required system semaphores via DosOpenSem(). Then it sets the semaphore SEMMW12, clears the semaphore SEMMW21 using DosSemClear() and begins its wait for the semaphore SEMMW12 to be cleared before continuing. Because the semaphore SEMMW21 has been cleared, the MANAGER process becomes dispatchable again. The same procedure is then applied sequentially to GHOUSE, CROP and PAVLOV.

During its initialization phase, the PAVLOV process also creates COGNITI and a new set of semaphores (SEMPC12 and SEMPC21) which are used to keep control of COGNITI (Figure V.7). The execution of COGNITI is initiated by executing GURU together with a start-up procedural file that contains the commands specific to a simulation. Since one objective of the simulations is to compare the behavior of a

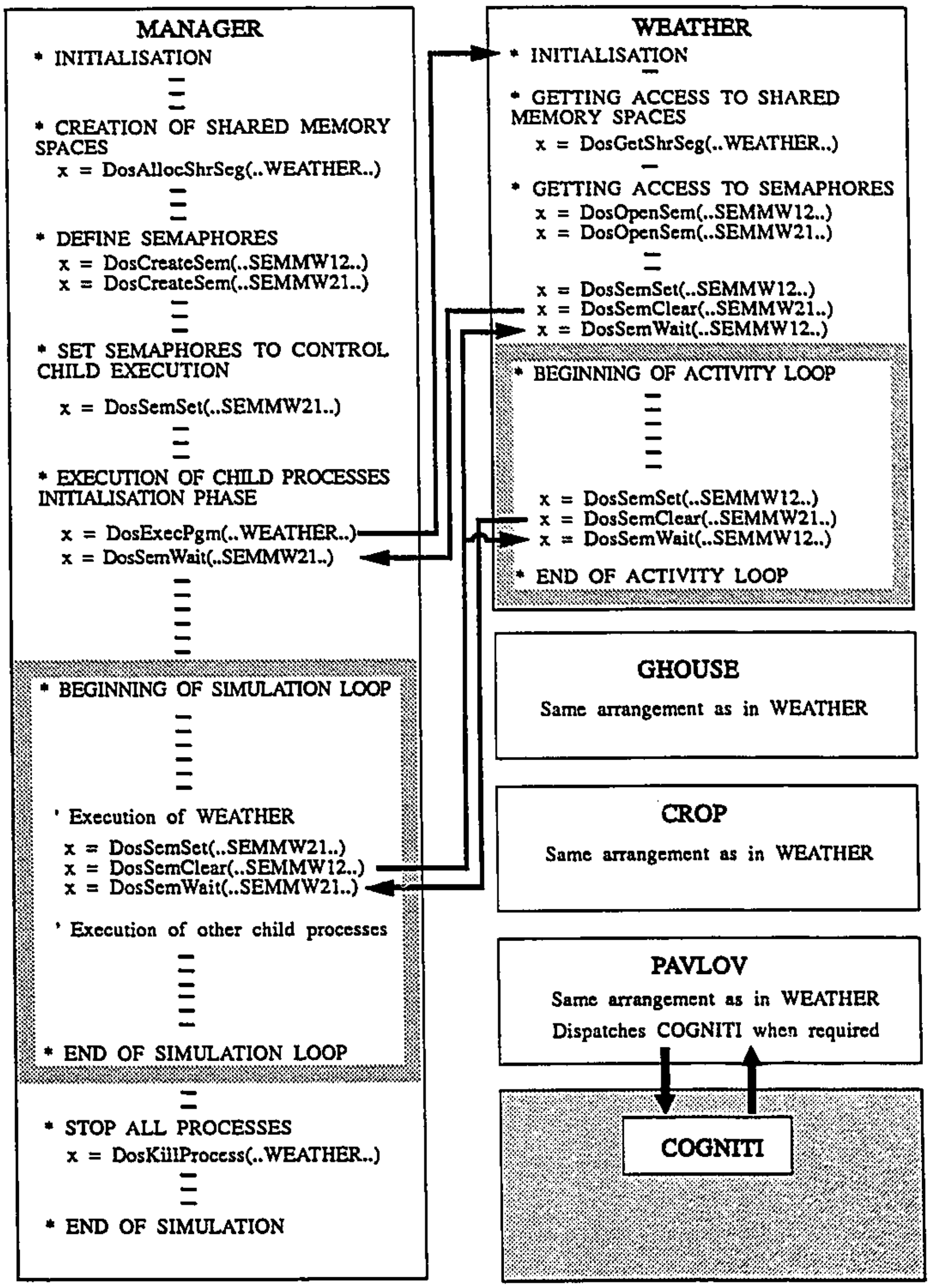
MANAGER
* INITIALISATION
* CREATION OF SHARED MEMORY SPACES
x = DosAllocShrSeg(..WEATHER..)
* DEFINE SEMAPHORES
x = DosCreateSem(..SEMMW12..)
x = DosCreateSem(..SEMMW21..)
* SET SEMAPHORES TO CONTROL CHILD EXECUTION
x = DosSemSet(..SEMMW21..)
* EXECUTION OF CHILD PROCESSES INITIALISATION PHASE
x = DosExecPgm(..WEATHER..)
x = DosSemWait(..SEMMW21..)
* BEGINNING OF SIMULATION LOOP
' Execution of WEATHER
x = DosSemSet(..SEMMW21..)
x = DosSemClear(..SEMMW12..)
x = DosSemWait(..SEMMW21..)
' Execution of other child processes
* END OF SIMULATION LOOP
* STOP ALL PROCESSES
x = DosKillProcess(..WEATHER..)
* END OF SIMULATION
WEATHER
* INITIALISATION
* GETTING ACCESS TO SHARED MEMORY SPACES
x = DosGetShrSeg(..WEATHER..)
* GETTING ACCESS TO SEMAPHORES
x = DosOpenSem(..SEMMW12..)
x = DosOpenSem(..SEMMW21..)
x = DosSemSet(..SEMMW12..)
x = DosSemClear(..SEMMW21..)
x = DosSemWait(..SEMMW12..)
* BEGINNING OF ACTIVITY LOOP
x = DosSemSet(..SEMMW12..)
x = DosSemClear(..SEMMW21..)
x = DosSemWait(..SEMMW12..)
* END OF ACTIVITY LOOP
GHOUSE
Same arrangement as in WEATHER
CROP
Same arrangement as in WEATHER
PAVLOV
Same arrangement as in WEATHER
Dispatches COGNITI when required
COGNITI

Figure V.6 Synchronization of processes in the functional simulation.

greenhouse when it is subjected to different control approaches, a different start-up file is created for each approach. The maintenance of control over COGNITI by PAVLOV is done with an approach slightly different from that used between MANAGER and its

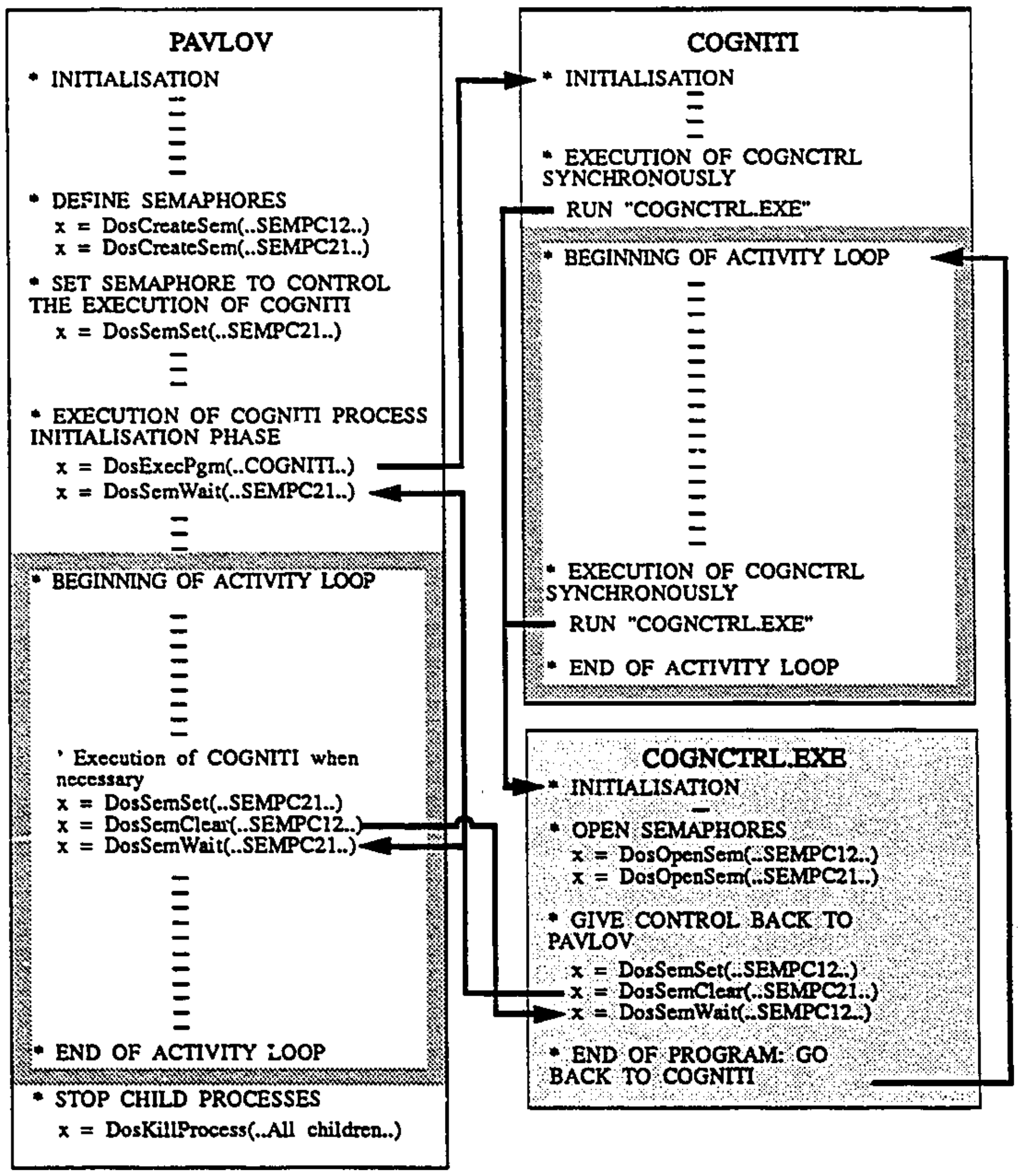
PAVLOV
* INITIALISATION
* DEFINE SEMAPHORES
x = DosCreateSem(..SEMPC12..)
x = DosCreateSem(..SEMPC21..)
* SET SEMAPHORE TO CONTROL THE EXECUTION OF COGNITI
x = DosSemSet(..SEMPC21..)
* EXECUTION OF COGNITI PROCESS INITIALISATION PHASE
x = DosExecPgm(..COGNITI..)
x = DosSemWait(..SEMPC21..)
* BEGINNING OF ACTIVITY LOOP
' Execution of COGNITI when necessary
x = DosSemSet(..SEMPC21..)
x = DosSemClear(..SEMPC12..)
x = DosSemWait(..SEMPC21..)
* END OF ACTIVITY LOOP
* STOP CHILD PROCESSES
x = DosKillProcess(..All children..)
COGNITI
* INITIALISATION
* EXECUTION OF COGNCTRL SYNCHRONOUSLY
RUN "COGNCTRL.EXE"
* BEGINNING OF ACTIVITY LOOP
* EXECUTION OF COGNCTRL SYNCHRONOUSLY
RUN "COGNCTRL.EXE"
* END OF ACTIVITY LOOP
COGNCTRL.EXE
* INITIALISATION
* OPEN SEMAPHORES
x = DosOpenSem(..SEMPC12..)
x = DosOpenSem(..SEMPC21..)
* GIVE CONTROL BACK TO PAVLOV
x = DosSemSet(..SEMPC12..)
x = DosSemClear(..SEMPC21..)
x = DosSemWait(..SEMPC12..)
* END OF PROGRAM: GO BACK TO COGNITI

Figure V.7 Execution of COGNITI process.

child processes, because COGNITI cannot directly access semaphores. This is because COGNITI is written in GURU 3.0, with cannot make calls to OS/2 API's. The approach that was used is as described below. PAVLOV first sets the semaphore SEMPC21, next starts the execution of COGNITI and begins its wait for the semaphore SEMPC21 to be cleared before continuing. The execution of COGNITI begins with an initialization phase. COGNITI then *synchronously* executes the external program COGNCTRL.EXE, i.e., COGNITI waits for COGNCTRL.EXE to terminate before continuing. COGNCTRL.EXE gets access to the semaphores defined in PAVLOV (SEMPC12 and SEMPC21), gives control back to PAVLOV and then becomes dormant. This is accomplished using the same procedure as that employed for MANAGER and its child processes. Because COGNITI waits for COGNCTRL.EXE to terminate, it also remains dormant.

To keep control of COGNITI, the simplest approach would have been to let PAVLOV execute GURU 3.0 each time COGNITI had to be executed. However, this would have required considerable resources in term of (physical) time and hard disk interactions. The approach adopted has the advantage of keeping in memory the values of all the variables created. Note that a more elegant procedure would have been to let MANAGER directly control COGNITI, but the present approach reduced the number of operations performed during a simulation and was quite sufficient for current needs.

V.5.2 Simulation loop

After all the children have been created, the functional simulation loop can start (Figure V.6). During a simulation cycle, MANAGER increments the functional time and writes the date and time values into its SMS. The child processes are then activated one after the other by use of the semaphore API's DosSemSet(..SEMxx21..), DosSem-Clear(..SEMxx12..) and DosSemWait(..SEMxx21..), in a way similar to that described in Section V.4.4. Each time a child is activated, the MANAGER process becomes dormant.

When the child completes an activity loop, it gives control back to MANAGER and becomes dormant.

The WEATHER process is the first child to be executed. WEATHER reads the date and time in MANAGER's SMS, determines the values of the meteorological variables by means of a look-up in a database and by interpolation, and writes them into its own SMS. Upon activation, the main program in the GHOUSE process (written in BASIC) then reads these values, as well as the values in the SMS's of MANAGER, CROP and PAVLOV. Next, it calls the FORTRAN subroutines that constitute the greenhouse model. These then establish the greenhouse moisture and energy balances. Finally, the main program writes the result values into its own SMS to make them available to the other processes. Next, the CROP process is activated. Again, there is a main program written in BASIC that handles the information flow and calls FORTRAN subroutines (the crop model) for the calculations of the crop growth. Again, the main program writes values in CROP's SMS.

The last child process to be activated in the cycle is PAVLOV. PAVLOV communicates with MANAGER, WEATHER and GHOUSE, reading from their SMS's and writing its decisions into its own SMS. It also collects data and carries out a preliminary treatment for COGNITI, and it activates COGNITI every five minutes. This is done by clearing the semaphore SEMPC12 (Figure V.7). When this semaphore is cleared, COGNCTRL.EXE terminates and control returns to COGNITI, which executes a complete loop of activities. These activities can include, for example, simple procedural treatments, operations on databases, calls to external functions, simulations or expert system consultations. When the loop is completed, COGNITI again executes COGNCTRL.EXE, which returns control to PAVLOV and becomes dormant until the next cycle during which COGNITI has to be executed.

V.6 CONCLUSION

The functional simulation structure was installed under OS/2. The approach that was adopted to synchronize the processes and to exchange data worked well. The simulation structure constitutes a good tool for the development of cognitive controllers, and it was used to develop the prototype of a simulation-based cognitive controller. The funtioning and the structure of this prototype are described in the following Chapter.

VI. DEVELOPMENT OF A PROTOTYPE SIMULATION-BASED CONTROLLER

VI.1 INTRODUCTION

A variety of control systems might be developed and tested with the help of the functional greenhouse system implemented within OS/2. In this project, the prototype of a greenhouse climate controller which has access to the use of simulation in its decision-making processes was developed. Various versions were developed and tested in this research, but the emphasis was on the development of a version in which the goal was to minimize the energy consumed for heating. In this case, the strategy used by the simulation-based controller (SBC) was based on the assumption that crops possess a temperature integration capacity, as was discussed in Chapter II (literature review). A few experiments were also done in which the role of the SBC was to maximize the net income (i.e., income from fruit sale minus cost for heating). These experiments were executed to illustrate how the functional simulation structure can be used to develop and test various types of cognitive controllers.

In the first of the following four sections in this chapter, the functioning of the SBC and its components will be described. This will include a description of the various approaches used by the SBC to determine the temperature setpoints. In the second and third sections, the specific role of the SBC in the various experiments will be presented. The reference controller, the controller with which the SBC is compared in the various experiments, will also be described in the second section. In the fourth section, the development of the neural model used for simulation by the controller will be discussed. Results of simulations executed with the functional greenhouse system in which the reference controller and the various versions of the SBC were implemented will be presented in Chapter VII.

VI.2 THE SIMULATION-BASED CONTROLLER

VI.2.1 General functioning and structure

A general description of the functioning and structure of the simulation-based controller (SBC) were given in Chapter IV. The development of the controller was a gradual process and only the final configuration is presented here. In this project, the SBC was involved in five main activities: 1) data input, 2) data treatment, 3) simulation-based determination of hourly temperature setpoints, 4) generation of "current" setpoints and 5) data output. These activities were not all carried out every time COGNITI was activated. The structure of the controller was algorithmic and, due to the relatively small number of activities, there was no need for a task manager of the complexity discussed in Chapter III (conceptual framework). For this research, there was a set of rules determining the activities to be done as a function of the current functional time. An infinite computing capacity was assumed, i.e., every time the module COGNITI was activated, all activities it had to perform were terminated before control was given back to the module PAVLOV.

The flow of information in COGNITI is shown in Figure VI.1. The first activity of the SBC was to input data every time COGNITI was activated (i.e., every five functional minutes in this research). This data came from the module PAVLOV and included the date, time, meteorological conditions (solar radiation, temperature and wind speed) and the internal air temperature (Air 1). The second activity was data treatment, which included summation of values every time COGNITI was activated. Also, at the end of each functional hour, averages were calculated and data stored in data bases. The third activity consisted of a simulation-based determination of hourly setpoints for the near future. In this research, setpoints were determined once a day (functional day) for the incoming 24-hour period. This was performed at the end of the hour during which sunset occurred. This process will be presented in more detail in Section VI.2.3. The last activity consisted of generating a "current" setpoint each time COGNITI was activated; this

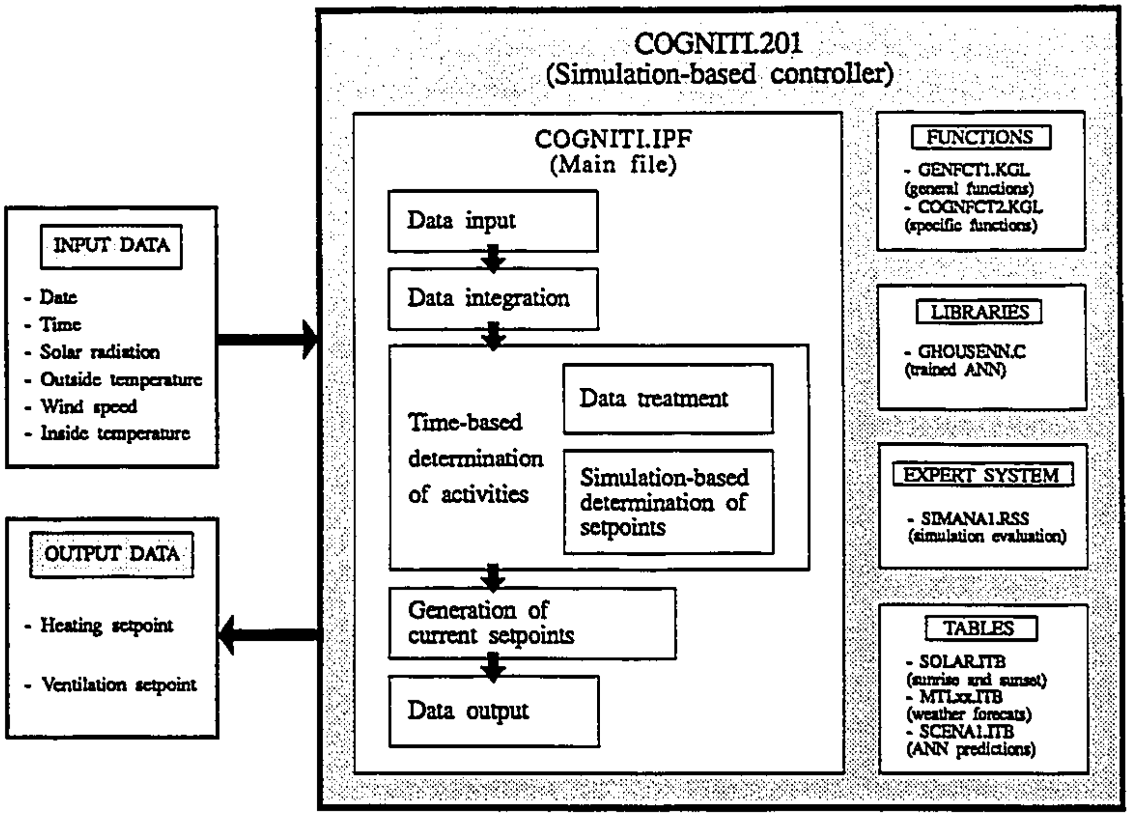

Figure VI.1 Structure of the prototype simulation-based controller (COGNITI.201).

setpoint was then written into a data file to be read and used by the module PAVLOV during the following five functional minutes.

VI.2.2 Software implementation

The module COGNITI was written in GURU version 3.0. All the files that made up the module COGNITI are shown in Figure VI.1. The listings of these files are presented in Appendices G.11 to G.15. The file COGNITI.IPF constituted the main program, which controlled the overall data treatment in the module COGNITI. This file was written in GURU 3.0 Procedural Language. The file GENFCT1.KGL, which was written in KnowledgeMan GURU Language, contained many general functions used by the main program for various treatments (e.g., summation of the elements of an array). The file COGNFCT2.KGL, also written in KnowledgeMan GURU Language, included more specific functions that were used during the simulation-based decision-making process. The trained neural network used in the subjective simulations was contained in the file GHOUSENN.C. This was written in the C language and also contained code for interfacing with GURU 3.0; it was compiled as a Dynamic-Link Library under OS/2 and was used within GURU as a CLINK function. The evaluations of the subjective simulations were done with a set of rules contained in the file SIMANA1.RSS, which also contained instructions to control the inference processes. This file was written in the language specific to expert system applications (Rule Set Source) within GURU 3.0. Finally, some data bases were accessed by COGNITI. The first data base was located in the file SOLAR.ITB which included fields for sunrise and sunset hours for each day of the year. The second data base, located in SCENA1.ITB, contained fields in which the predictions of the neural network and the path of temperature setpoints chosen by the controller were stored. The other data bases (MTL82.ITB, MTL83.ITB and MTL84.ITB) contained weather forecasts and were used during the subjective simulations. There was one of these databases for each of the three years used in the functional simulations. For this research, the incoming meteorological conditions were taken to be perfectly known by the controller; the data contained in the weather forecast databases were identical to those used by the functional simulation.

VI.2.3 Simulation-based determination of setpoints

A simulation-based control approach allows a controller to predict how the controlled system will react according to the anticipated meteorological conditions. The controller can then choose the setpoint trajectory that will allow it to best fulfil its control objective(s). An anticipation period (i.e., the period for which conditions are predicted and analyzed) of many days gives a lot of flexibility to such a controller which has the choice between a large number of possible temperature trajectories. The longer the period that is considered in the decisions, the more flexible the control can be and, consequently, the more optimal it can be made. However, the longer the period that is considered, the more numerous are the possible setpoint trajectories and, consequently, the more complex are the data treatments and the required decision-making processes. For this research, a simple case was considered, to help determine the constraints and requirements for the future consideration of more complex cases.

The chosen approach consisted of an anticipation period of 24 hours. The goal of the controller was to 1) generate a series of scenarios that were expected to lead to a required 24-hour temperature integral, 2) run simulations to estimate energy requirements and/or net crop assimilation for each scenario and 3) depending on the experiment, choose the setpoint trajectory for which the energy requirement was minimum or for which the net income was maximum. This simulation-based determination of setpoints was done once per 24-hour period, at the beginning of each night. Specifically, it was done at the end of the hour during which sunset occurred. For most experiments, the 24-hour temperature integral to be achieved was the same as that furnished by a reference controller, i.e., the controller with which the SBC is compared. In two experiments, this constraint was relaxed slightly, and the controller had the choice between scenarios leading to various temperature integrals; this will be explained in Section VI.4.

The simulation-based setpoint determination process was divided into four phases, in a way similar to that defined in the conceptual framework (Section III.5.3). This process is illustrated in Figure VI.2. The four phases were: 1) determination of the simulation domain, 2) execution of simulations, 3) evaluation of the simulation results and 4) final decision-making.

The determination of the domain of the simulation was comprised of three aspects: 1) generation of subjective time, 2) retrieval of anticipated meteorological conditions, and 3) generation of a sequence of temperature setpoints to be tested. The first aspect, related

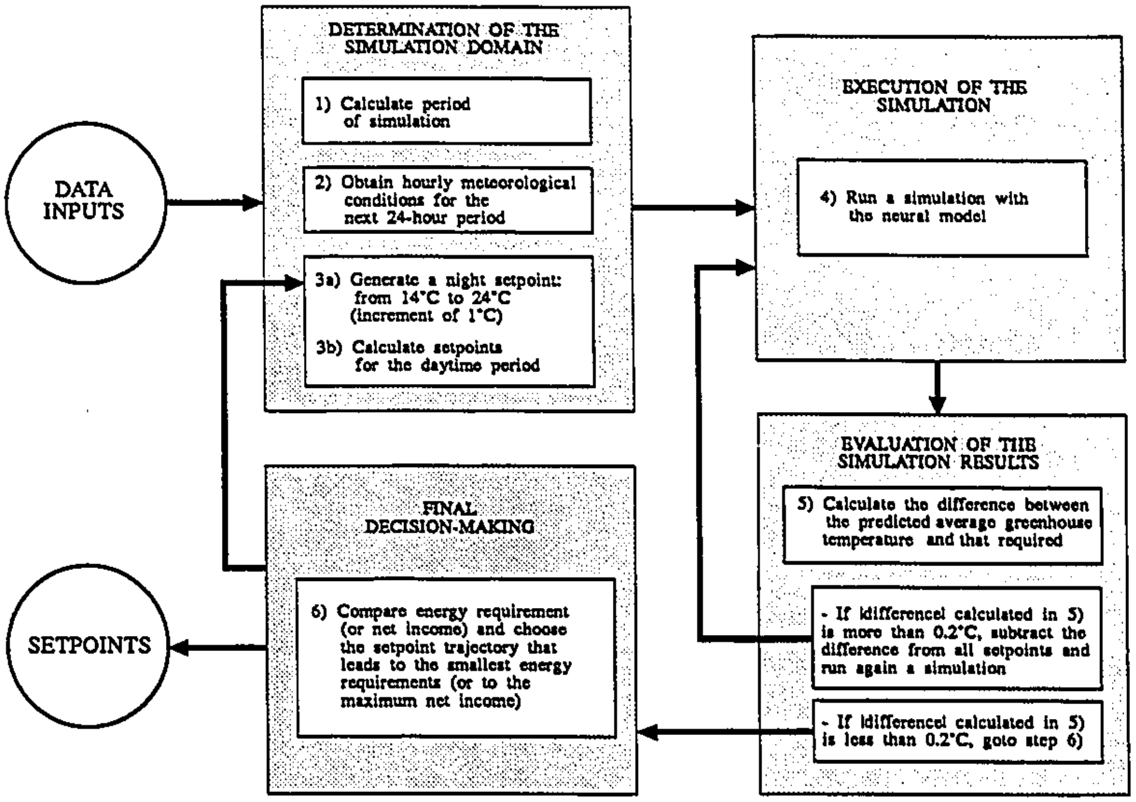
DATA INPUTS
DETERMINATION OF THE SIMULATION DOMAIN
1) Calculate period of simulation
2) Obtain hourly meteorological conditions for the next 24-hour period
3a) Generate a night setpoint: from 14°C to 24°C (increment of 1°C)
3b) Calculate setpoints for the daytime period
EXECUTION OF THE SIMULATION
4) Run a simulation with the neural model
EVALUATION OF THE SIMULATION RESULTS
5) Calculate the difference between the predicted average greenhouse temperature and that required
- If |difference| calculated in 5) is more than 0.2°C, subtract the difference from all setpoints and run again a simulation
- If |difference| calculated in 5) is less than 0.2°C, goto step 6)
FINAL DECISION-MAKING
6) Compare energy requirement (or net income) and choose the setpoint trajectory that leads to the smallest energy requirements (or to the maximum net income)
SETPOINTS

Figure VI.2 Simulation-based determination of the setpoints by the prototype controller (COGNITI.201).

to subjective time, consisted of determining the duration of the simulation, which goes up to the end of the hour during which the sunset will occur on the following day. The information contained in the data base SOLAR.ITB was used in this calculation. The second aspect consisted of the retrieval of the anticipated meteorological conditions from the appropriate databases (e.g., in the data base MTL83.ITB for 1983), for the subjective time period generated in the first step. The third aspect consisted of the generation of hourly temperature setpoints to be tested for the subjective period previously determined. For most experiments in this research, the same procedure was used to accomplish this: a setpoint between 15°C and 24°C (with increments of 1°C) was first chosen for the night and then the corresponding setpoint for the day was calculated so as to maintain a 24-hour temperature integral equal to a target value. The relationship used is given in

Equation VI.1, in which the temperature integral is expressed in terms of a 24-hour average temperature. For most experiments, the 24-hour average temperature was kept constant for a specific month and was the same as the monthly average temperature obtained with the reference controller.

$$\mathrm{DaySpt} = \frac{24 * \mathrm{DAvgTemp} - \mathrm{NightSpt} * \mathrm{NightLen}}{\mathrm{DayLen}} \qquad \text{(VI.1)}$$

where:

DaySpt:	Hourly setpoint for the day	[°C]
DAvgTemp:	Average temperature to be maintained over 24 hours	[°C]
NightSpt:	Hourly setpoint for the night	[°C]
NightLen:	Duration of the night	[h]
DayLen:	Duration of the day	[h]

Small variations of this approach were tried also. For example, the use of an increment of 0.1°C was tried. As well, in one experiment, the daily setpoints had the possibility of following a sine curve. In another experiment, as previously mentioned, the controller had the possibility to use various 24-hour temperature integral values. The specific approaches used for the various experiments are described in Sections VI.3 and VI.4.

Once the domain was determined, the simulation was executed for the 24-hour period. After that, the resulting 24-hour average temperature (as predicted by the ANN model of the greenhouse) was examined to see if it was sufficiently close to what was required. For instance, it was possible that the predicted greenhouse temperature was higher than the setpoint because of overheating. If the difference between the predicted average temperature and the target average temperature was less than 0.2°C, the results were used in the fourth phase (final decision-making). If the difference was larger than 0.2°C, all setpoints were adjusted according to this difference and a new simulation was run. During the adjustment of the setpoints, some rules were used to verify that the new setpoints were both lower than the desired maximum (24°C) and higher than the desired minimum (15°C). The process of adjusting the setpoints and repeating the simulation was done for a maximum of 30 times. If a significant difference still existed after 30 times, the results were used as is.

After the results were accepted in the third phase, they were analyzed by a rule set (SIMANA1.RSS) in the fourth phase. The goal of the rule set was to compare the actual results with those obtained with a previous "winner" setpoint path. The rules contained in SIMANA1.RSS were developed and used to evaluate the simulation results with respect to one of two goals: 1) minimization of the energy requirements for heating or 2) maximization of net income. In this thesis, emphasis was placed on the first aspect, i.e., on the minimization of the energy requirements for heating but, for illustrative purposes, a few experiments were done where the goal was the maximization of the net income. For the minimization of energy, the rule set compared the energy requirements for the current path of temperature setpoints to the energy requirement with the previous winner path. The path with the lowest overall energy requirement was then selected. Equivalent rules were used when the goal was the maximization of the net income.

The entire process described above (comprising the four phases) was repeated for the next value of night setpoint. In the case of the experiments where scenarios associated with different temperature integrals were analyzed by the SBC, the entire process was also repeated for each of the various temperature integral values. When the entire nest of loops was completed, the final chosen path of setpoints and the corresponding predicted values of heating, greenhouse temperature and crop assimilation were written into data base SCENA1.ITB. The setpoint values in SCENA1.ITB were then used during the following functional 24-hour period to generate a setpoint every five minutes for the functional simulation.

VI.3 GOAL 1 OF THE SBC: MINIMIZATION OF ENERGY COSTS

VI.3.1 The control strategy

As discussed in the literature review (Chapter II), the use of a heating schedule based on the assumption that crops possess a temperature integration capacity could lead to more efficient control of the greenhouse temperature than when traditional procedures are used. Potential benefits include a reduction of the heating load 1) by better usage of the free solar energy and 2) by shifting heating to periods when heat loss coefficients are smaller. In the literature review, it was pointed out that a reduction of the heating load from 3% to more than 15% was possible, just by manipulating heating setpoints, so as to transfer heating from periods when the overall heat loss coefficient is high to periods with lower coefficients. Specifically, it was suggested that the heating load could be reduced substantially by transferring heating from a windy day to a calm day. A reduction could also be obtained by transferring heating from the day to the night for a greenhouse equipped with a thermal screen.

Before a strategy was developed to be used by the SBC, some calculations were done to obtain an idea of the benefits that might be expected by shifting heating towards periods when heat loss coefficients are low. The method used and the results are described in Appendix E. These results indicate that shifting the heating to periods when heat loss coefficients are lower could lead to substantial reductions in energy consumption, depending on factors such as the difference between inside and outside temperatures, the variation of the overall heat transfer coefficient of the greenhouse, etc. For example, the opening of the thermal screen in the functional greenhouse system increased the overall heat loss coefficient by an amount varying between 30% and 50% (see Appendices B and D). In this case, as is shown in Appendix E, if the temperature difference between the inside and outside were 10°C and if the inside temperature during the night were 4°C higher than during the day instead of being equal, energy consumption might be reduced by 6 to 7%. Greenhouse heat loss coefficients also vary as a function of meteorological conditions such as wind speed. For example, in Appendix B, the overall heat loss coefficient was observed to increase by as much as 80% when the wind speed increased from 0 m/s to 10 m/s. In this case, for a difference of temperature of 10°C between inside and outside, a transfer of heating from a period of very high wind speed to a period of very low wind speed might also lead to a reduction in energy requirements between 6 and 7%, if the inside temperature difference between the two periods is 6°C.

In this context, a simulation-based control approach is attractive because it allows a controller to predict how the greenhouse system will react according to the anticipated meteorological conditions. The controller can then choose the setpoint trajectory that will lead to lower energy consumption. As previously discussed, an anticipation period that can be as long as one week gives a lot of flexibility to such a controller. However, at the same time, this approach involves more complex data treatment and decision-making processes. The adopted approach consisted, therefore, of a prediction period of 24 hours, within which the temperature integral had to be established. The goal of the controller (referred to below as COGNITI.201) was to 1) generate a series of scenarios that were expected to lead to the required temperature integral, 2) run simulations to estimate energy requirements for each scenario, and 3) choose the setpoint trajectory for which the energy requirement was minimum. This simulation-based decision-making process was done as described in Section VI.2.3. It was done once per 24-hour period, at the beginning of each night. The 24-hour temperature integral to be maintained was to be similar to that of a reference controller, which will be described in Section VI.3.2.

VI.3.2 The reference controller

The effects of simulation-based control on the greenhouse system were to be compared with those of a reference controller. The reference controller might have been the controller COGNITI.101, that was used in Appendix D for the dynamic evaluation of the functional greenhouse system. It maintained a fixed day/night temperature regime (i.e., the day and night setpoints were kept constant respectively at 21°C and 17°C). However, it was decided to create another version of the controller in which the 21°C and 17°C setpoints would be used, except that for each night following a day during which there was overheating, the setpoints would be modified so as to maintain a relatively constant 24-hour temperature integral. This was done following a procedure similar to that suggested by de Koning (1988b). There were two main justifications for this choice. First, it was desirable to compare the effects of two different regimes that lead to approximately the same 24-hour temperature integral. Second, it was interesting to see the effects of adding the "compensation effect" to the more traditional temperature regime used in Appendix D. The reference controller will be referred to below as COGNITI.102. The basic principles of the algorithm used by COGNITI.102 are explained in this section. More details are given in the source code of the file COGNITI.102 listed in Appendix G.8.

The strategy used by the reference controller to generate setpoints was to force the 24-hour temperature integral to a given target value. The target value of the integral (for a given day in the year) was the value that would be obtained if the day temperature was constant at 21°C and the night temperature was 17°C. In its strategy, the reference controller kept the daytime setpoint constant at 21°C. Then, during the night, it calculated a new setpoint every five minutes, so as to achieve the target 24-hour temperature integral. These setpoints were a function of the temperature integral achieved up until then (calculated starting from the beginning of the previous daytime period), the length of the remaining night period and the target 24-hour temperature integral (de Koning, 1988b).

The target 24-hour temperature integral was calculated with equation VI.2:

$$\text{TargInt} = (\text{DayLen} * 21) + (\text{NightLen} * 17) \qquad \textbf{(VI.2)}$$

where:

TargInt:	Target value of the 24-hour temperature integral	[°C·m]
DayLen:	Length of the day	[m]
NightLen:	Length of the night	[m]

The length of the day and of the night were calculated with standard equations used in solar engineering. The nighttime setpoints were calculated every five minutes with equation VI.3:

$$\text{NightSpt} = \frac{(\text{TargInt} - \text{RealInt})}{\text{RemNight}} \qquad \textbf{(VI.3)}$$

where:

NightSpt:	Nighttime setpoint	[°C]
RealInt:	Temperature integral achieved since the beginning of the 24-hour period	[°C·m]
RemNight:	Length of the remaining night period	[m]

The temperature integral achieved since the beginning of the 24-hour period was calculated using the average greenhouse temperature realised during each interval of five functional minutes (i.e., between two calls to the module COGNITI).

During the initial development of the reference controller, no minimum temperature setpoint had been initially given. This led to very low temperatures during nights following days with high intensity of solar radiation (e.g., a minimum greenhouse temperature of 9.4°C was reached in March 1982). A fixed minimum setpoint of 12°C was subsequently used in all simulations.

VI.3.3 Development and evaluation procedure

Since the length of the day varies throughout the year and the day and night setpoints were not identical, the 24-hour temperature integral maintained by the reference controller varied slightly from one 24-hour period to the next. For this reason, it was decided that, for each day, the SBC must achieve a 24-hour average temperature equal to the monthly average temperature achieved with the reference controller. The main difference between the reference controller and the SBC was that more flexible temperature regimes could be generated and maintained with the latter.

The reference controller (COGNITI.102) and the SBC (COGNITI.201) were installed one after the other in the module COGNITI of the functional simulation structure. Parameters values as defined in Appendices B and C were used. The ventilation setpoint was kept constant at 25°C for both controllers. Simulations were executed with AES data for one month periods between January and May for 1982, 1983 and 1984. The period between January and May was chosen because it involves heating and large meteorological variations from day to day. In January, it is generally cold and solar radiation intensity is low, while in May it can be either hot or cold, with either high or low solar radiation intensity. The simulation results will be presented in Chapter VII.

VI.4 GOAL 2 OF THE SBC: MAXIMIZATION OF NET INCOME

A second series of experiments was executed to illustrate how the functional greenhouse system can be used to develop and test various control strategies. In this second series, the goal of the SBC was to maximize the net income, defined as the difference between income from fruit sales and expenses due to heating. Generally, an approach similar to that described in Section VI.2.3 was used.

The predicted income from fruit sale was calculated as a function of the predicted daily fruit production, the price of fruit and an attenuation factor. The attenuation factor was used since SUCROS87 calculated potential growth, i.e., growth under ideal conditions. The income was then calculated using equation VI.4:

$$\text{Income} = \text{FrEff} * \text{FrProd} * \text{FrPrice} \qquad \text{(VI.4)}$$

where:

Income:	income from fruit sale	[$\$/m^2$]
FrEff:	attenuation factor	[%]
FrProd:	amount of fruit produced	[kg/m^2]
FrPrice:	price of fruit	[$/kg]

As described in Chapter IV, an oil furnace was modelled in this research. The predicted expense related to heating was therefore a function of the predicted daily energy consumed for heating, the furnace efficiency and the price of oil. The oil was assumed to have a calorific value of 38850 kJ/l. The expense for heating was calculated with equation VI.5:

$$\text{Expense} = \frac{\text{Energy}}{\text{FurnEff} * 38850} * \text{EnPrice} \qquad \text{(VI.5)}$$

where:

Expense:	cost of heating	[$\$/m^2$]
Energy:	amount of energy consumed	[kJ/m^2]
FurnEff:	furnace efficiency	[%]
EnPrice:	price of oil	[$/l]

Four experiments were executed using this approach for the Month of March 1983. In the first experiment, the simulation-based determination of setpoints by the SBC was as described in Section VI.2.3. In the second experiment, the SBC had the possibility of maintaining temperature setpoints as low as 12°C (instead of 15°C). In the third experiment, the SBC had the possibility of choosing among various values of the 24-hour temperature integral. In the fourth experiment, the SBC had the possibility of forcing the daily setpoints to follow a sine curve. In all cases, the results were compared with the results obtained with the reference controller (COGNITI.102). The following values were used in all experiments: the fruit attenuation factor was 75%, the price of fruits was 2.20\$/kg (CPVQ, 1984), the furnace efficiency was 75% and the price of oil was 0.329\$/l (CREAQ, 1985). The results of the experiments will be presented in Chapter VII.

VI.5 DEVELOPMENT OF THE NEURAL MODEL

VI.5.1 Overall procedure

As was described in Chapter IV, the model of the greenhouse system used by the SBC was an artificial neural network (ANN) with 9 inputs and 3 outputs. Before being used for simulation, the ANN needed to be configured, trained and tested for it to constitute a valid representation of the functional greenhouse system. Many factors must be considered in the development of an ANN for a specific application. The first aspect is related to the data needed to train and test the ANN. The second aspect involves the choice of an appropriate configuration for the ANN to be able to learn the patterns existing within the training data set. A third aspect concerns the depth of the learning process, measured in terms of numbers of learning cycles. The training of an ANN can be a long process of trial and error, where the complexity of the patterns implicitly present in the data affects the choice of configuration, which, in turn, affects the number of learning cycles.

For this project, a feed-forward ANN architecture of the back-propagation type was adopted. The main factors to consider with this type of ANN are the number of hidden layers, the number of processing elements (PE's) per hidden layer, the transfer function used in the PE's, the learning rule and the learning schedule. Various combinations of these factors were investigated. The overall process for configuring, training and testing these ANN's is illustrated in Figure VI.3. First, training and testing data were produced by running simulations with the functional greenhouse system. Next, the ANN's were trained for a pre-determined number of learning cycles using the training data set. Then, the trained ANN's were used in recall mode with inputs from the testing data set. The results were compared with the desired outputs included in the testing data set. Then, the ANN with the best performance was chosen. The results of several comparisons are reported in this thesis to illustrate how different factors can influence the performance of an ANN. However, the internal functioning of ANN's will not be discussed here; detailed explanations can be found in various publications (Lacroix and Kok, 1992; NeuralWare, 1991; Zhuang and Engel, 1990).

VI.5.2 Production of training and testing data sets

ANN's must be trained with data that are representative of all situations which they might face in recall mode. For this project, the ANN had to be trained so as to be able to react to various combinations of meteorological conditions and setpoints. Because the whole greenhouse system was simulated, it was possible to generate appropriate training data sets. However, at the time when the data sets were produced, the exact structure and functioning of the SBC was not yet known. Thus, it was decided to produce data that would include a wide spectrum of situations with which the ANN might have to deal. The training data was therefore produced by running functional simulations with three different COGNITI setpoint generators. The first one was the reference controller, COGNITI.102. The other two produced setpoints randomly, as will be explained below. The file name of these other two setpoint generators were COGNITI.103 and COGNITI.104. They are listed in Appendices G.9 and G.10.

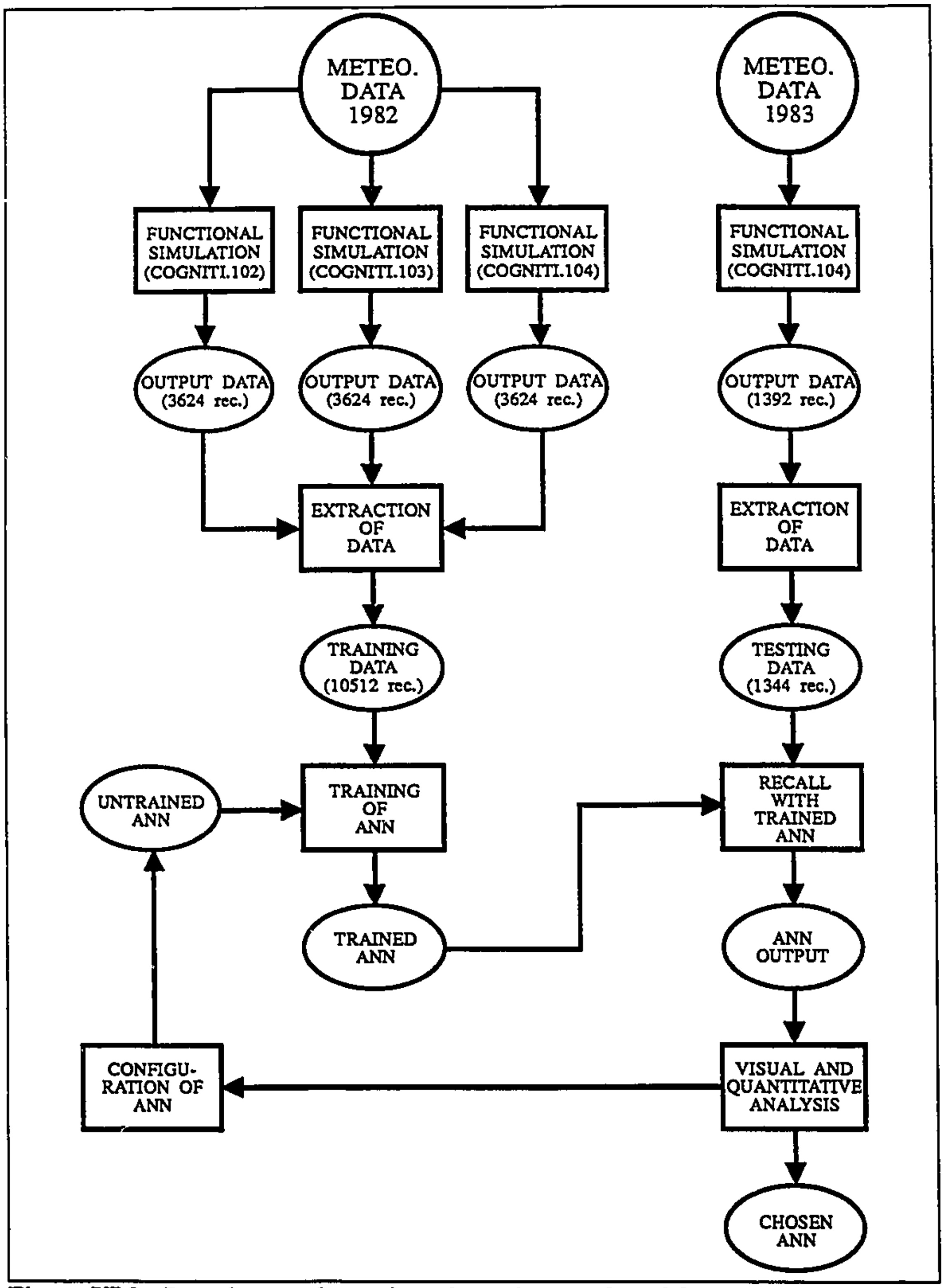

Figure VI.3 Procedure used to train and test the neural models.

In COGNITI.104, the procedure to generate setpoints was similar to that used in the reference controller, except that, at the beginning of each day, the day setpoint was chosen randomly between 12.0°C and 24.5°C. The night setpoints were then calculated so as to achieve the same monthly average temperature as with the reference controller. In COGNITI.103, at the end of a given time period, the duration of the next period was determined randomly between 0 and 86400 seconds (uniformly distributed). Then, the type of variation of the setpoint for the period was chosen randomly from two possibilities. These were: 1) the setpoint could be constant or 2) it could follow a sine curve. A probability of 50% was associated with each option. Next, a random value was chosen between 14°C and 24.5°C and this was used as either the setpoint constant or as the average of the setpoint sinusoid. For the sinusoid, an amplitude was also chosen, randomly, between 0°C and 2.5°C. For both setpoint generators COGNITI.103 and COGNITI.104, the minimum and maximum temperatures were fixed at 12.0°C and 24.5°C respectively. The later was just slightly lower than the ventilation setpoint, which was 25°C.

To produce the training data set, simulations were done with the functional simulation set-up for the months between January and May 1982, using all three versions of COGNITI (i.e., COGNITI.102, COGNITI.103 and COGNITI.104) to generate the setpoints (Figure VI.3). To produce data to test the ANN's, simulations were done with COGNITI.104, for the months of February and April 1983. For every month of simulation, an output file was created and from each of these the records of the first 24 hours were removed so as to keep only data that were independent from the initial conditions. The 15 training data sets that were generated (three COGNITI versions times five months) were combined to form a single training set of 10512 records. Using the same approach, the two testing data sets were merged into one file that contained 1344 records.

VI.5.3 The number of learning cycles

Before comparing different ANN configurations, the number of learning cycles to be applied for their training was determined. The appropriate number of learning cycles is specific to any application and is determined by an iterative process. It is a function of the complexity of the patterns to be internalized as well as the ANN configuration. For this research, the approach that was adopted consisted of using one of the tools included in NeuralWorks software (NeuralWare, 1991) that allows one to follow the degree of learning of an ANN during a training process.

The tool that was used calculates and plots the variation of a root mean square (RMS) error value as a function of the number of learning cycles. The RMS value indicates the degree of error between the values an ANN is being trained to predict and those that it actually calculates. During a training process, the RMS value should move towards zero. The RMS error value during the training of a sample ANN is shown in Figure VI.4. The RMS value had become relatively stable at 100000 learning cycles. For the data used, a value of 100000 learning cycles was found to be adequate for all tested ANN's to converge to a stable state. A value of 100000 was also found to be adequate for the training of ANN's with similar data in a previous study (Kok *et al.*, 1993). Thus, all configurations that were compared in this thesis were trained for 100000 cycles.

VI.5.4 Comparison of different configurations

For the determination of an appropriate ANN configuration, many trials were done, some of which led to very poor results. To give an impression of how the different factors influenced the learning capacity of the ANN's, the results of two series of experiments are reported here. The first series is related to the architecture of ANN's, i.e., to the number of hidden layers and to the number of processing elements (PE's) in each hidden layer. In this series, the internal characteristics were the same for all ANN's that were

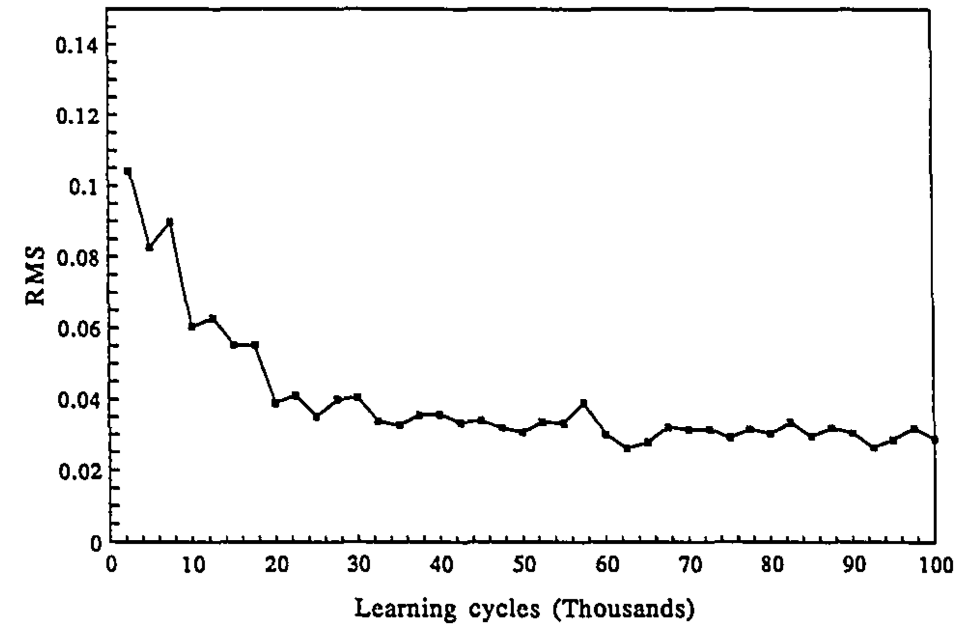

Figure VI.4 Illustration of the variation of the RMS error value during the training of an ANN.

compared. The second series of experiments is concerned with internal characteristics such as the type of transfer function and the learning rule. All ANN's that were compared in this second series had the same architecture.

To compare the various ANN's, both qualitative and quantitative analysis were used. The qualitative analysis consisted of comparing graphs of the predicted and the desired values. For the quantitative analysis, three mathematical tools were used. The two first were the mean and standard deviation of the differences between the predicted results and those desired. The third tool consisted of a score value (called SCORE hereafter) calculated using a procedure described by Kok *et al.* (1993). This procedure consists of filtering the data and calculating a weighted root-mean square. The advantage of this method is that the value produced is unitless; it is, therefore, useful for comparing results for different variables. The values for SCORE may vary between 0 and 100. A value of 0 means extremely poor predictive capacity, while a value of 100 would correspond to perfect predictive ability for the ANN in question. The procedure for calculating a SCORE value is described in Appendix F.

VI.5.4.1 Architecture

In the first of the two series of experiments, the ANN's differed only in their architecture, i.e., in the number of hidden layers and in the number of PE's per hidden layer. The ANN's were constructed using a NeuralWorks standard procedure to build back-propagation ANN's. The delta-rule learning rule was used for training and the sigmoid function was used as the transfer function in the PE's. Experiments were done with many different architectures, but results are reported here only for four of these (Table VI.1).

Table VI.1 Description of four different neural network architectures.

	Number of hidden layers	Number of PE's per hidden layer	Total number of PE's	Total number of connections
CONF1	2	19	50	9747
CONF2	1	19	31	513
CONF3	1	10	22	270
CONF4	2	5	22	675

The four ANN's were each trained for 100000 learning cycles and tested in accordance with the procedure shown in Figure VI.3. The results of the testing process for the three ANN output variables are shown in Table VI.2. The SCORE values are listed together with the means and the standard deviations of the differences between the values calculated by the ANN's and the target values (i.e., the testing data). The SCORE values and the standard deviations of the differences are inversely related and lead to similar conclusions. For example, the higher the SCORE value is, the lower is the standard deviation of the differences for a given output variable, and the better are the predictions.

The results varied considerably between architectures (Table VI.2). For the given number of learning cycles, the ANN's with one hidden layer learned the patterns better than those with two hidden layers. Different learning schedules and a larger number of learning cycles might have improved the results obtained with the ANN's with two hidden layers. Also, for a given number of hidden layers, the number of processing elements affected the learning of the ANN. Thus, for a given number of learning cycles and a given set of internal characteristics, the architecture of an ANN seems to affect its capacity to approximate the relationships that exist between the inputs and outputs. Finding an adequate structure requires many trials.

Table VI.2 Means (Avg.) and standard deviations (Std.) of the differences between target and calculated values, and SCORE values (Score) for the first set of experiments.

	Temperature [°C]			Heating load [W/m^2]			Net assimilation rate [g(CO$_2$)/m^2·h]		
	Avg. [°C]	Std. [°C]	Score [%]	Avg. [°C]	Std. [°C]	Score [%]	Avg. [°C]	Std. [°C]	Score [%]
CONF1	-0.4	2.5	71	-12.3	37.5	59	.061	0.46	75
CONF2	-0.2	1.0	90	-1.1	23.3	80	.056	0.33	83
CONF3	-0.3	1.0	90	0.3	23.6	79	.054	0.34	82
CONF4	-0.5	2.5	71	-14.2	44.4	51	-.060	0.84	54

VI.5.4.2 Internal characteristics

In the second series of experiments, the objective was to show how, for a given architecture and a given number of learning cycles, the degree of learning of an ANN can be influenced by internal characteristics such as the learning rule, the transfer function in the processing elements, etc. This series of experiments was done with the ANN that produced the most successful results during the investigation to find the most appropriate configuration for this research. The configuration of the ANN used was similar to that developed by NeuralWare (1991) to predict stock prices. It differed from the ANN used in the first set of experiments in many aspects such as the learning rule, the transfer function and the architecture. However, in the second series, the same architecture was conserved for all experiments (i.e., they all possessed the same number of hidden layers and PE's). Only internal characteristics were modified.

The architecture of the ANN's used in this series is illustrated in Figure VI.5. All ANN's contained three hidden layers with nine PE's each. Two separate hidden layers (HIDDEN 1A and HIDDEN 1B) were attached to the input layer. These two layers were attached to the third hidden layer (HIDDEN 2) which was attached to the output layer. All layers were fully connected although all the connections are not shown in Figure VI.5.

For this series of experiments, the procedure consisted of starting with a set of internal characteristics similar to that one used in the first series (e.g., sigmoid function, delta-rule learning rule), changing one factor at a time incrementally until the final configuration (the most successful) was reached. The results obtained during this series of experiments

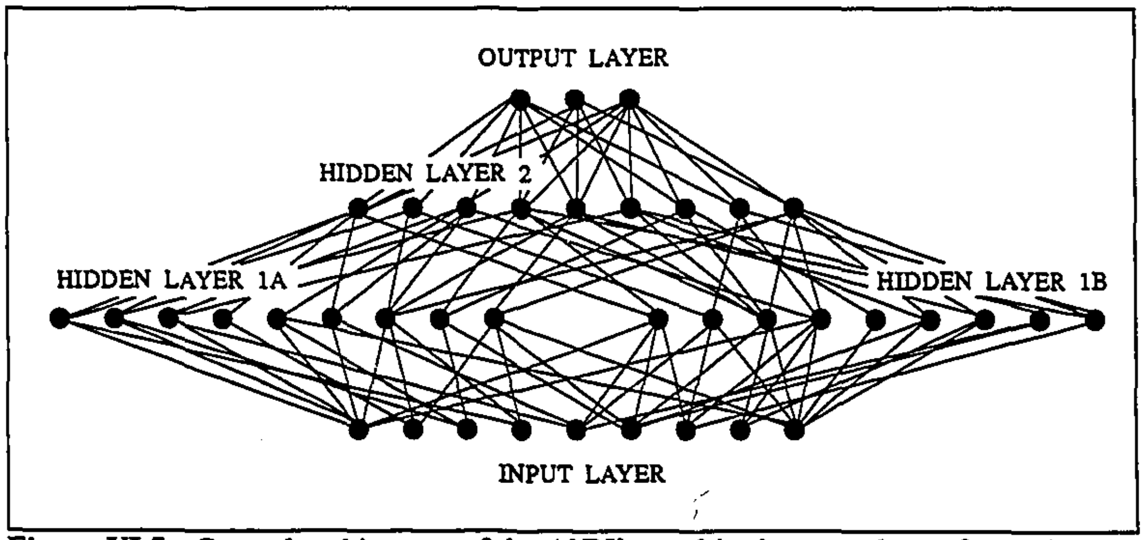

Figure VI.5 General architecture of the ANN's used in the second set of experiments.

are reported in Table VI.3. The differences in the configurations are presented below, together with the results. Many terms specific to ANN internal characteristics are used without being defined; more information can be obtained in NeuralWare (1991) or in any textbook on artificial neural networks.

Except for the architecture, CONFI1 possessed the same configuration as that in the first set of experiment (Section VI.5.4.1). The results obtained with CONFI1 were comparable to those obtained with CONF1 and CONF4 (Table VI.2). In CONFI2, the cumulative delta-rule was used for learning rather than the delta-rule, as before. The results were similar to those obtained with CONFI1. The configuration of CONFI3 was the same as for CONFI2, except that the transfer function for the layer HIDDEN 1B was changed from sigmoid to sinusoid. This led to a considerable improvement of the learning of the ANN, especially with respect to heating. In CONFI4, the size of the epoch (used by the

Table VI.3 Means (Avg.) and standard deviations (Std.) of the differences between target and calculated values, and SCORE values (Score) for different ANN configurations.

	Temperature [°C]			Heating load [W/m^2]			Net assimilation rate [g(CO$_2$)/m^2·h]		
	Avg. [°C]	Std. [°C]	Score [%]	Avg. [°C]	Std. [°C]	Score [%]	Avg. [°C]	Std. [°C]	Score [%]
CONFI1	-0.43	2.61	70	-13.3	45.5	50	0.057	0.46	75
CONFI2	-0.47	2.65	69	-13.8	44.5	51	0.061	0.46	75
CONFI3	-0.42	1.87	79	2.0	20.7	82	0.096	0.39	78
CONFI4	-0.37	1.48	83	-1.4	20.4	82	0.103	0.37	80
CONFI5	-0.34	2.39	72	-0.7	20.6	82	0.080	0.42	78
CONFI6	0.01	0.67	94	1.1	19.5	83	0.040	0.22	89
CONFI7	0.05	0.70	94	1.9	18.2	84	0.076	0.23	88
CONFI8	0.56	0.63	94	-1.2	16.1	86	0.032	0.20	90

cumulative delta-rule) was changed from 16 to 100 records. With this change, the learning slightly improved in comparison with CONFI3. The difference between CONFI5 and CONFI4 was in the learning schedule (used by NeuralWorks during a training process to change learning coefficient values, e.g., the momentum term). In CONFI5 the learning schedule was the same as that used by NeuralWare to predict stock prices. This change induced a small decrease in the learning of the ANN. In CONFI6, the transfer function used by the elements of the output layer in CONFI5 was changed from sigmoid to linear, which led to a large improvement, especially for temperature and crop assimilation. In CONFI7, the ANN bias was disconnected from the elements of the output layer. This did not improve the learning of the ANN. Finally, in CONFI8, the low value of the bias was changed, which led to the final configuration. This configuration furnished the best results of the eight ANN's compared in Table VI.3. This was the configuration chosen for the SBC model.

The testing data and the predictions of three ANN configurations (CONFI1, CONFI4 and CONFI8) are shown for the three ANN output variables in Figures VI.6, VI.7 and VI.8 respectively, for the first 15 days of February 1983. In general, the differences between the predictions and the testing data were large with CONFI1, especially for the greenhouse temperature and the heating load. CONFI8 generally produced the smallest differences and the results with CONFI4 were intermediary to CONFI1 and CONFI8. It can be noted that what is illustrated in these figures corresponds to the results that were expressed by the SCORE values.

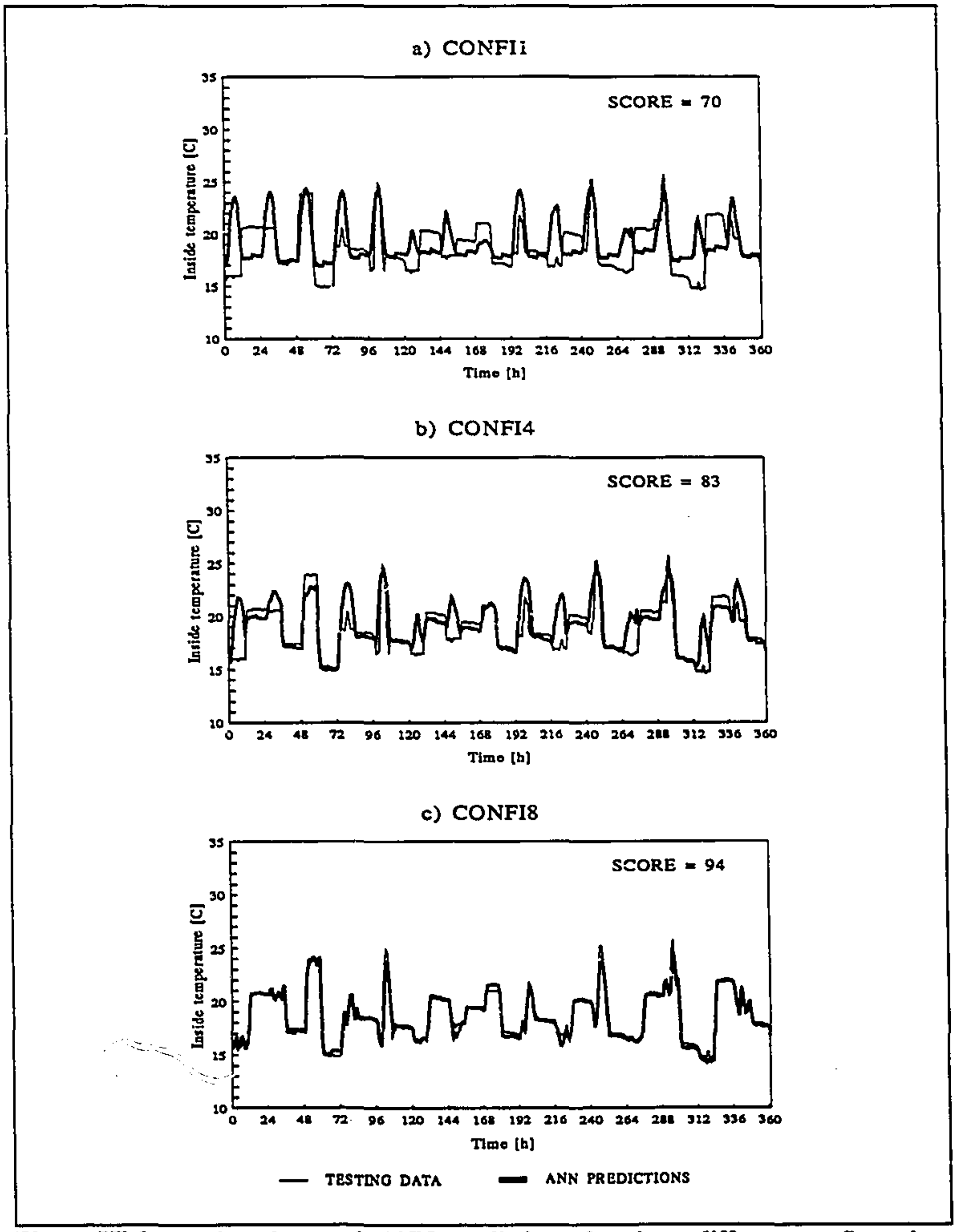

Figure VI.6 Testing data and ANN predictions for three different configurations (CONFI1, CONFI4 and CONFI8) for internal air temperature during the first 15 days of February 1983.

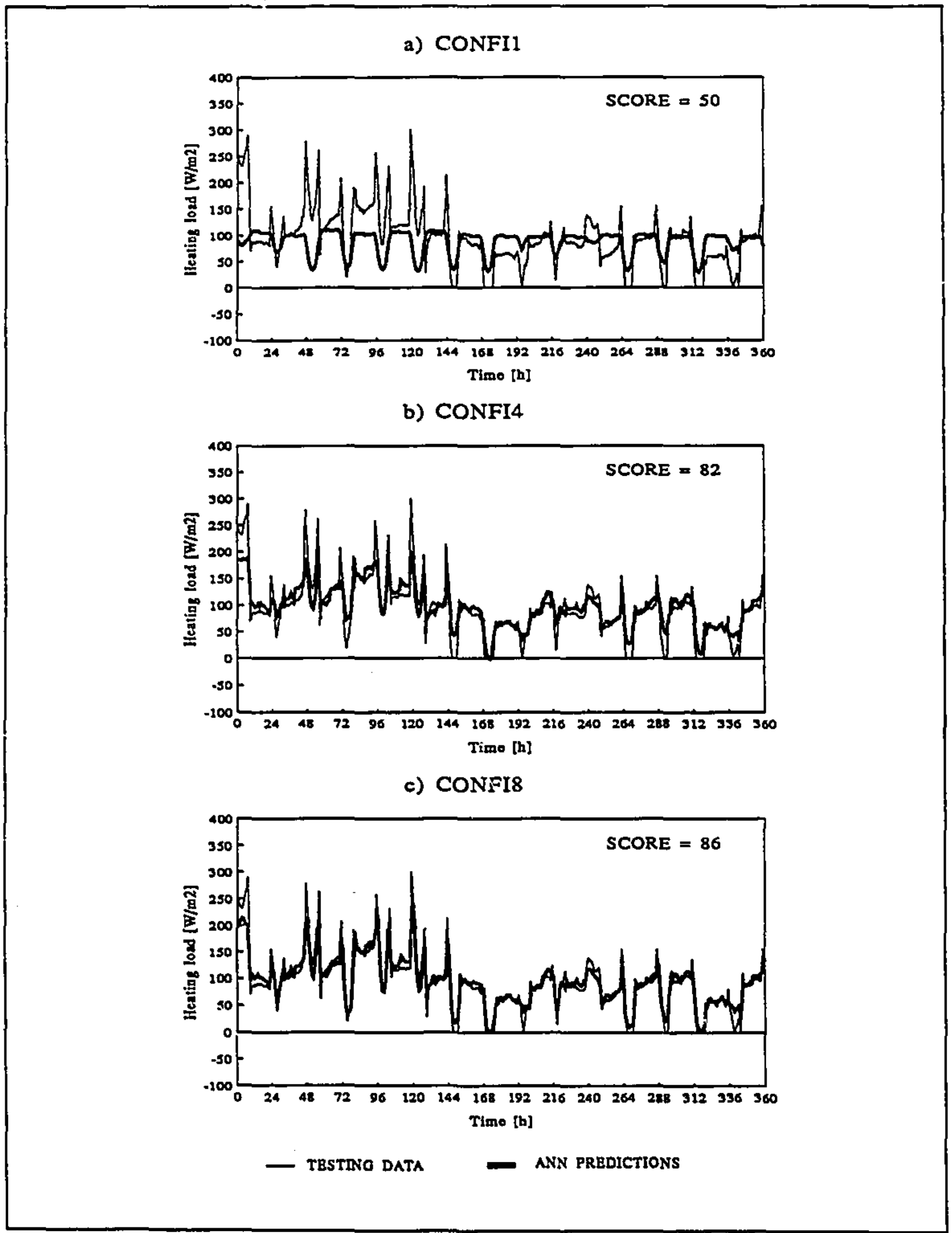

Figure VI.7 Testing data and ANN predictions for three different configurations (CONFI1, CONFI4 and CONFI8) for heating load during the first 15 days of February 1983.

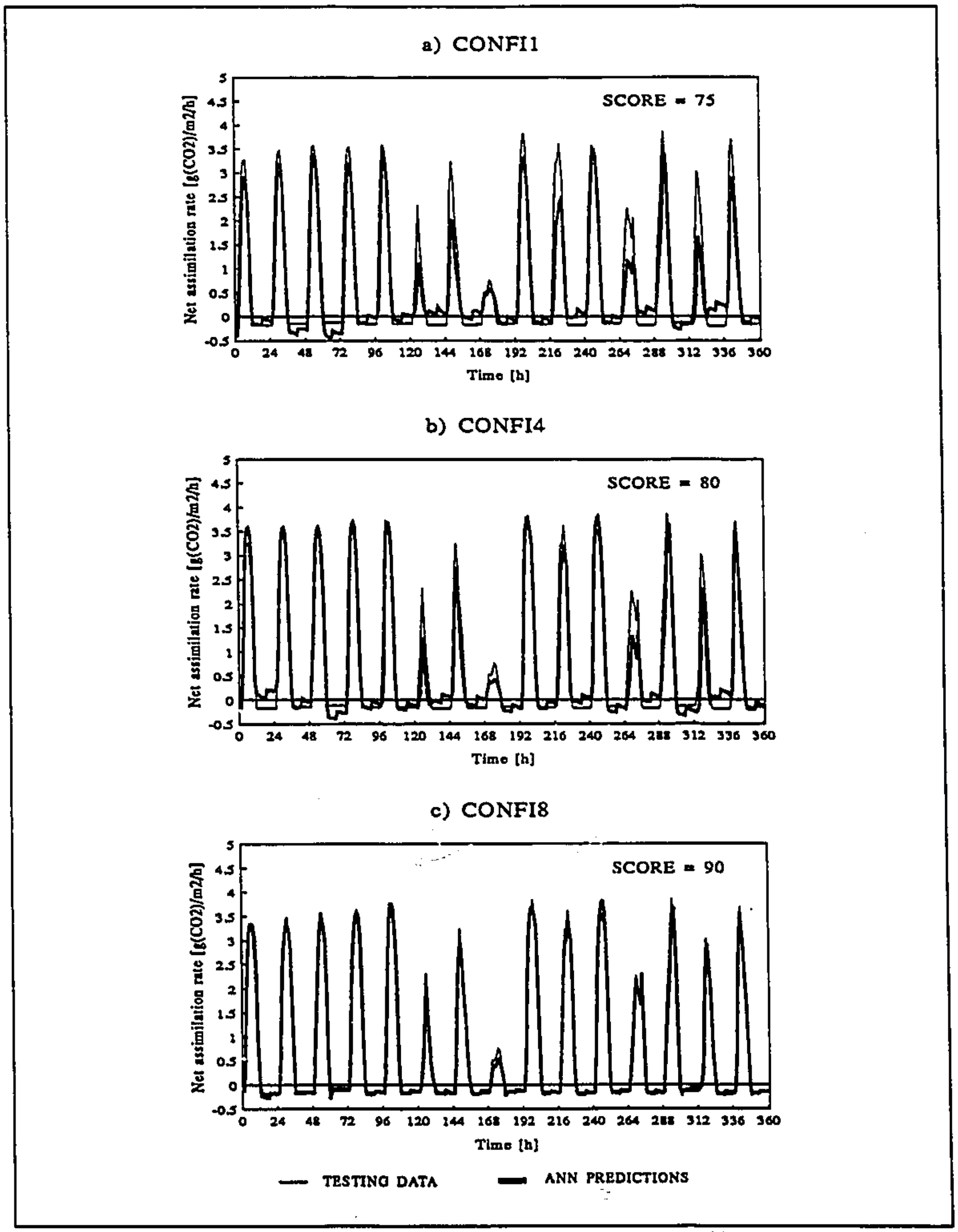

Figure VI.8 Testing data and ANN predictions for three different configurations (CONFI1, CONFI4 and CONFI8) for the net assimilation rate during the first 15 days of February 1983.

VI.6 CONCLUSION

The general structure and the functioning of the SBC were first described in this chapter. The phases of the simulation-based process for the determination of setpoints were outlined. Setpoint determination by the SBC can be accomplished according to one or several goals. Two goals were presented: the minimization of the energy requirements for heating and the maximization of the net income. With regard to the first goal, a strategy was conceived based on the assumption that crops possess a temperature integration capacity. The strategy consists of shifting heating towards periods where greenhouse heat loss coefficients are lower. For this research, the SBC used this strategy to decide, on the basis of prediction and analysis, what setpoints to maintain so as to reduce energy consumption while achieving a specific 24-hour temperature integral.

The last part of the chapter concerned the construction of the neural model for the SBC. The objective was to find an ANN configuration that would be appropriate for the model. The investigation about the configuration of ANN's was limited, but it illustrated that many factors will affect the learning of ANN's and that appropriate choices must be made to allow an ANN to detect the patterns that are present in the data sets. The configuration found in CONFI8 furnished the best results of all ANN's that were tested. The experiments where the role of the SBC was to minimize energy consumption were done with this model. The same model was used in the illustrative experiments on the maximization of net income. The results of all the experiments are presented in Chapter VII.

VII. RESULTS OBTAINED WITH THE VARIOUS CONTROL STRATEGIES

VII.1 INTRODUCTION

This chapter is divided into three parts. In the first part, the simulation results obtained with the reference controller (COGNITI.102) will be presented. They will be compared with the results obtained with the controller used for the evaluation of the functional greenhouse system described in Appendix D (COGNITI.101). In the second part, the simulation results obtained with the simulation-based controller (COGNITI.201), for which the goal was to minimize the energy needed for heating, will be presented. The predictions of the neural network will be first compared with what occurred during the functional simulations. Then, the results obtained with COGNITI.201 will be compared with that obtained for the same periods with COGNITI.102. Comparisons will also be made with the results obtained with COGNITI.101. The third part of the chapter will be concerned with a brief analysis of the results obtained with the SBC when its goal was to maximize net income.

VII.2 THE REFERENCE CONTROLLER

The internal air (Air 1) temperatures obtained with the setpoints generated by the reference controller (COGNITI.102) for the first 15 days of the months of January, March and May 1982 are shown in Figure VII.1. In January, the temperature generally fluctuated between the base setpoints, i.e., between 17°C and 21°C. In March and May, periods of overheating occurred frequently. During the nights following these overheating periods, COGNITI.102 generated setpoints that were low enough to achieve the required 24-hour temperature integral (i.e., between 12°C and 15°C).

Results of simulations executed with the reference temperature regime (COGNITI.102) and with the fixed day/night temperature regime (COGNITI.101) used in Appendix D for

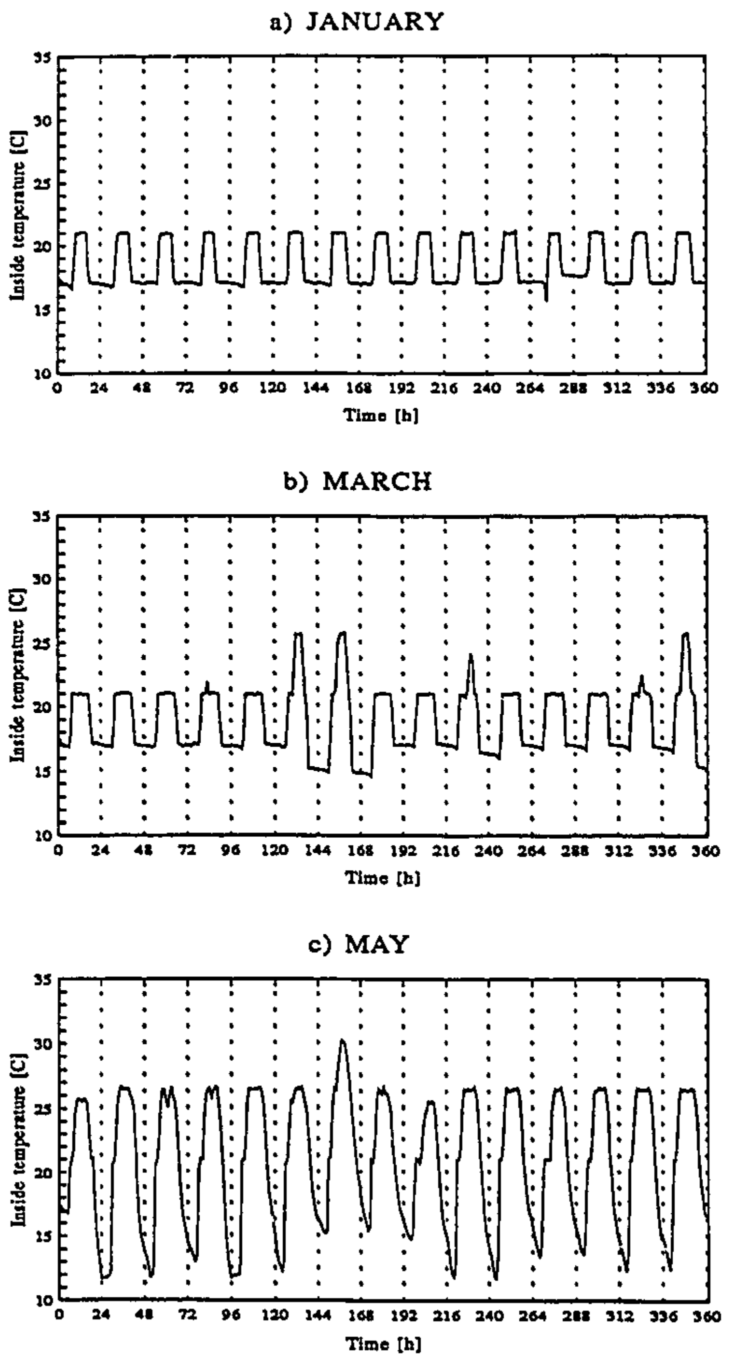

Figure VII.1 Internal air temperature with setpoints generated by the reference controller during the first 15 days of January, March and May 1982.

Table VII.1 Internal air temperatures obtained with controllers COGNITI.101 and COGNITI.102.

	COGNITI.101			COGNITI.102		
	Average temp. [°C]	Minimum temp. [°C]	Maximum temp. [°C]	Average temp. [°C]	Minimum temp. [°C]	Maximum temp. [°C]
January	18.4	15.6	21.2	18.4	15.6	21.2
February	18.8	16.9	25.7	18.7	15.3	25.8
March	19.5	16.9	26.8	18.9	13.1	26.8
April	20.2	16.6	26.8	19.5	11.7	26.8
May	22.0	16.6	30.2	21.5	11.7	30.2

the evaluation of the functional greenhouse system are shown in Tables VII.1 and VII.2. The results, which are for the months from January to May 1982, give an idea of the effect of adding a compensation for overheating to a fixed day/night temperature regime. The difference between the monthly average temperature maintained by each regime increased from January to May (Table VII.1). The difference was negligible in January, which implies that there was probably no need for temperature compensation during this

Table VII.2 Energy requirements for heating with controllers COGNITI.101 and COGNITI.102.

	Energy for heating COGNITI.101 [MJ/m^2]	Energy for heating COGNITI.102 [MJ/m^2]	Difference in percentage [%]
January	388.2	389.1	0.2
February	256.2	254.9	-0.5
March	202.0	193.6	-4.2
April	142.8	131.9	-7.6
May	33.1	23.6	-28.7
All five months	1022.3	993.1	-2.8

month. With COGNITI.102, the minimum temperature decreased from January to May, resulting in temperatures slightly lower than the minimum allowable setpoint value in April and May (i.e., 11.7°C). The maximum temperatures were similar for both regimes since they were essentially a function of the ventilation for which the setpoint was constant and identical in both regimes. For both regimes, the monthly average greenhouse temperature increased between January and May. This was due to two main factors. The first factor was that the length of the day increased from January to May; therefore, the day temperature setpoint of 21°C was applied over longer periods. The second factor was that the frequency of overheating increased from January to May. This reason is valid even with COGNITI.102 since overheating was not totally compensated for, due to the limitation imposed by the minimum allowed setpoint. The energy requirements for heating for the two temperature regimes are listed for each month in Table VII.2. The requirements with COGNITI.102 were slightly lower than with COGNITI.101 between March and May, while the difference was negligible for January and February.

The monthly average greenhouse temperatures maintained with COGNITI.102 for the first five months of the years 1982, 1983 and 1984 are shown in Table VII.3. The monthly average temperature varied slightly from one year to the other in April and May. This was

Table VII.3 Monthly average internal air temperatures obtained with COGNITI.102 in 1982, 1983 and 1984.

	Average temperature for 1982 [°C]	Average temperature for 1983 [°C]	Average temperature for 1984 [°C]
January	18.4	18.4	18.4
February	18.7	18.7	18.6
March	18.9	18.9	18.9
April	19.5	19.2	19.5
May	21.5	20.1	20.3

mostly due to the variation in total solar energy received during a specific month from year to year. Consequently, it was decided that, for a given year, the simulation-based controller (SBC) should achieve the same daily average temperature as the monthly average temperature obtained with the reference controller during that year. The results of the simulations executed with the SBC trying to minimize the amount of energy consumed for heating are presented in the following section.

VII.3 MINIMIZATION OF THE ENERGY REQUIREMENTS (SBC)

VII.3.1 Predictions of the neural model

In the main set of experiments, the goal of the simulation-based controller (SBC) was to minimize the energy requirements for heating. The results obtained with the SBC will be compared to the results obtained with the reference controller, but the predictions of the neural model included in the SBC and the situations that occurred during the functional simulations will first be compared. For this comparison, the SCORE value method will be used again (see Appendix F). The two output variables of interest for the SBC in its simulations were the internal air (Air 1) temperature and the heating load. The SCORE values for 1982, 1983 and 1984 are listed in Table VII.4.

For the internal air temperature, the SCORE values were generally high for all three years (i.e., in the range of 85% to 90%). Except for May 1983, the predictions for 1983 and 1984 were slightly better than in 1982. The predicted (ANN) and obtained (functional simulation) values for the internal air temperature for the 2nd to the 16th day of January, March and May 1983 are shown in Figure VII.2. Generally, the predictions were worst at the low and high temperatures, mostly during the nights in January and during the days in March, when the solar radiation intensity was high enough to lead to overheating. In May, the deviations between predicted and obtained values occurred during both the days and the nights. Generally, the ANN was able to predict the pattern of the maximum daily

Table VII.4 Values of SCORE for the two output variables of interest of the ANN during 1982, 1983 and 1984.

	SCORE for Temperature [%]		
	1982	1983	1984
January	86	90	90
February	87	88	87
March	86	90	86
April	88	89	88
May	88	87	89
	SCORE for Heating load [%]		
	1982	1983	1984
January	87	87	87
February	87	89	87
March	86	86	87
April	83	83	79
May	63	75	72

temperature (around 27°C), which was largely dependent on the ventilation setpoint.

The SCORE values for the heating load are also shown in Table VII.4. In all three years, the quality of these predictions by the ANN decreased from January to May. For May, the SCORE value was always below 75%. However, it can be seen in Figure VII.3 that the predictions of the ANN for heating followed the general pattern rather well, but differed in the details. Since in May the amounts of energy were quite low, the predictions of the ANN were considered as acceptable.

The ANN's training was based entirely on 1982 data. However, the SCORE values obtained during the years 1983 and 1984 were generally at least as high as the SCORE values for 1982. This indicates that the ANN was probably not overtrained and that it had

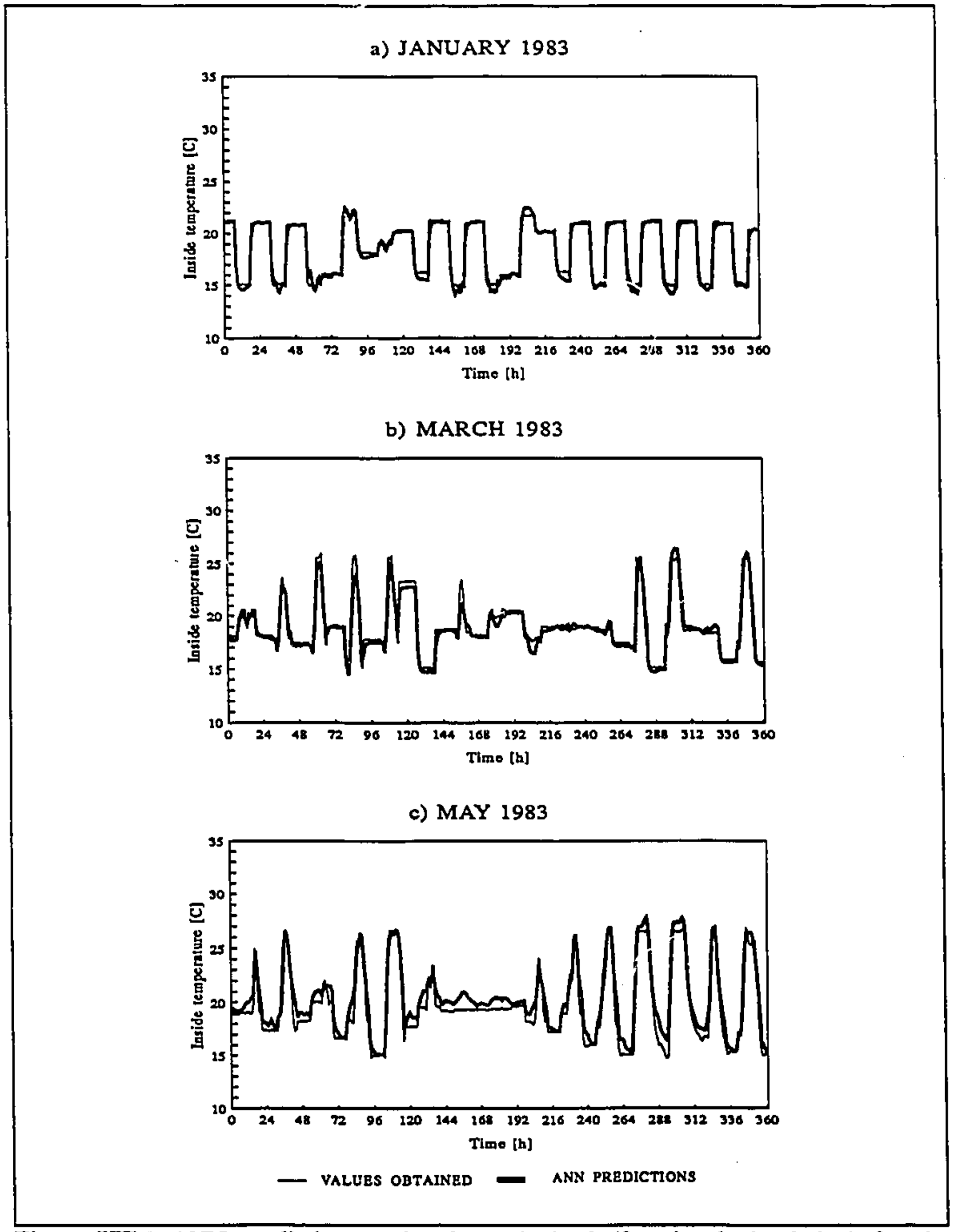

Figure VII.2 ANN predictions and values obtained (functional simulation) for the internal air temperature, from the 2nd to the 16th of January, March and May 1983.

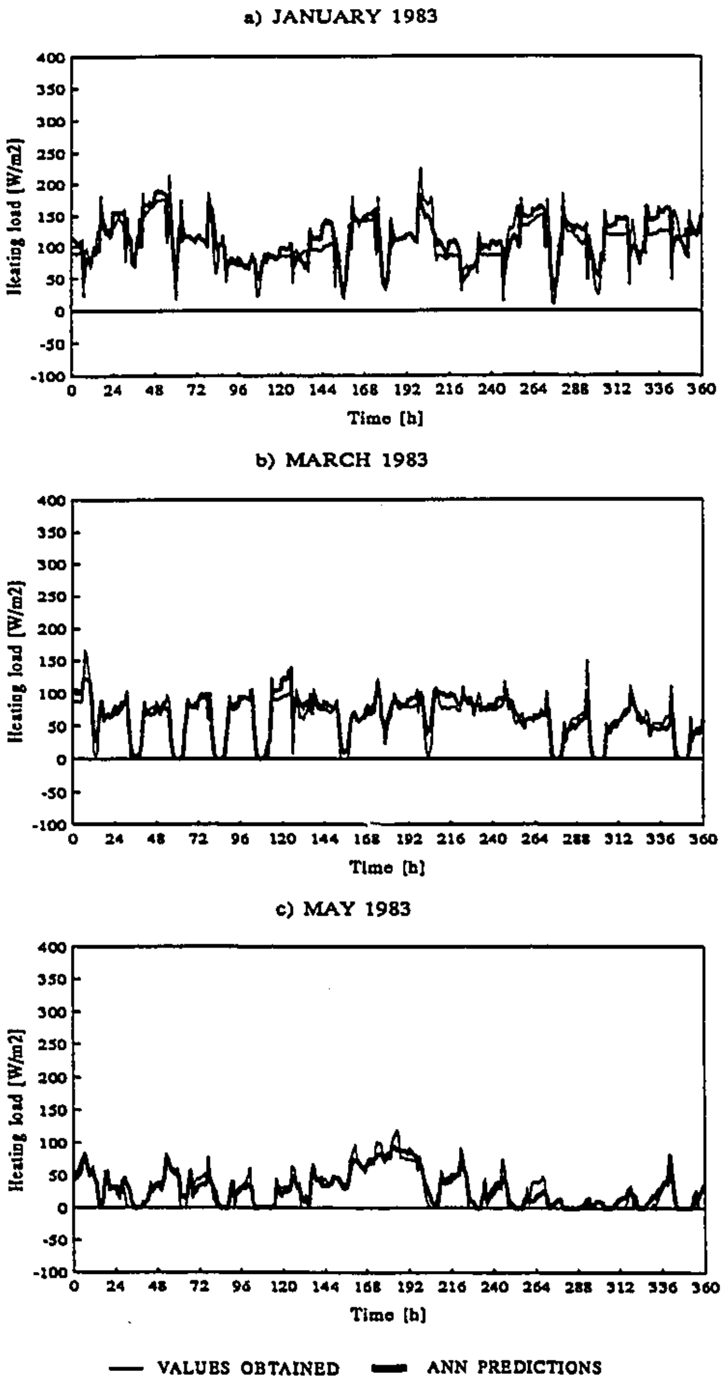

Figure VII.3 ANN predictions and values obtained (functional simulation) for the heating load, from the 2nd to the 16th of January, March and May 1983.

internalized only the general relationships existing between the inputs and the outputs in the training data set. Thus, its predictive capacity was not constrained to be only applicable to contexts that occurred in 1982.

VII.3.2 Comparison of the SBC with the reference controller

The temperature setpoints generated by both COGNITI.102 and COGNITI.201 during the first 15 days of January 1982 are plotted in Figure VII.4a. The COGNITI.201 setpoint sequence is shown together with the resultant internal air temperature in Figure VII.4b. The outside temperature and solar radiation intensity during these periods are plotted in Figure VII.4c. The same information for the first 15 days of January 1983 and 1984 is given in Figures VII.5 and VII.6. This information was also generated for the months of March and May 1982, 1983 and 1984 and is presented in Figures VII.7 to VII.12.

In Figures VII.4a, the temperature regime generated by the SBC (COGNITI.201) was observed to be inverted for most of January 1982, i.e., high setpoints were generated for the nights and low setpoints were generated for the days. The SBC maintained similar temperature regimes for January 1983 and 1984. The night setpoint was often close to 21°C. The SBC, therefore, estimated with the help of its simulations that it was generally more beneficial in January to maintain higher temperatures during the night, when the thermal screen was closed. There were a few exceptions to this, for example during the 4th night or at 96 h in January 1983, when the day setpoint was higher than the night setpoint. These exceptions often coincided with periods during which the outside temperature increased rapidly during the night and/or at the beginning of the following day. For example, during the 5th day of January 1983 (i.e., between 96 h and 120 h), the outside temperature increased from -16°C in the night to about 0°C in the middle of the day (Figure VII.4c). A similar situation occurred between 216 h and 240 h. These rapid changes of the outside temperature appear to have influenced the decisions of the controller. As can be observed in Figure VII.4b, generally, the greenhouse temperature

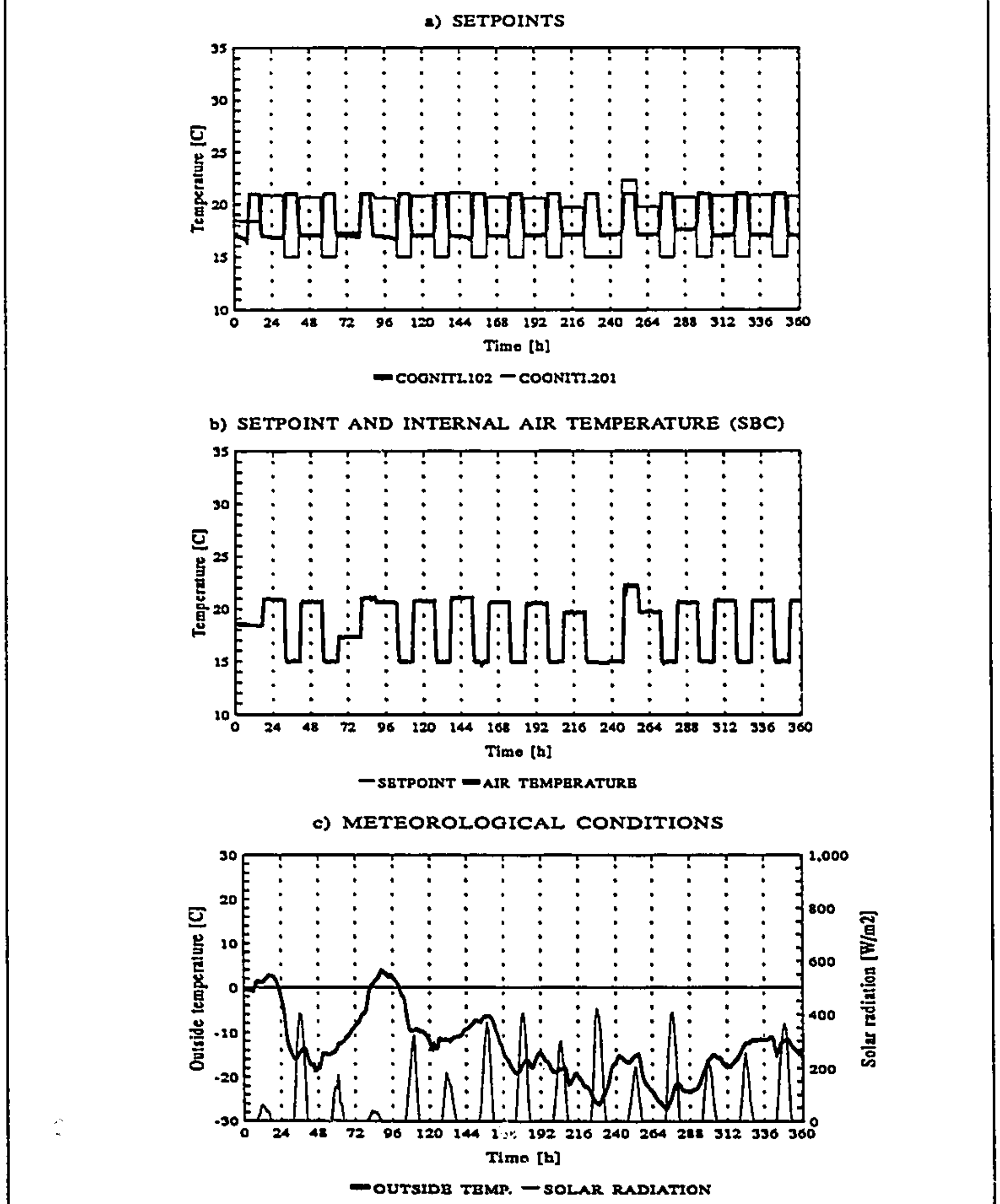

Figure VII.4 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of January 1982.

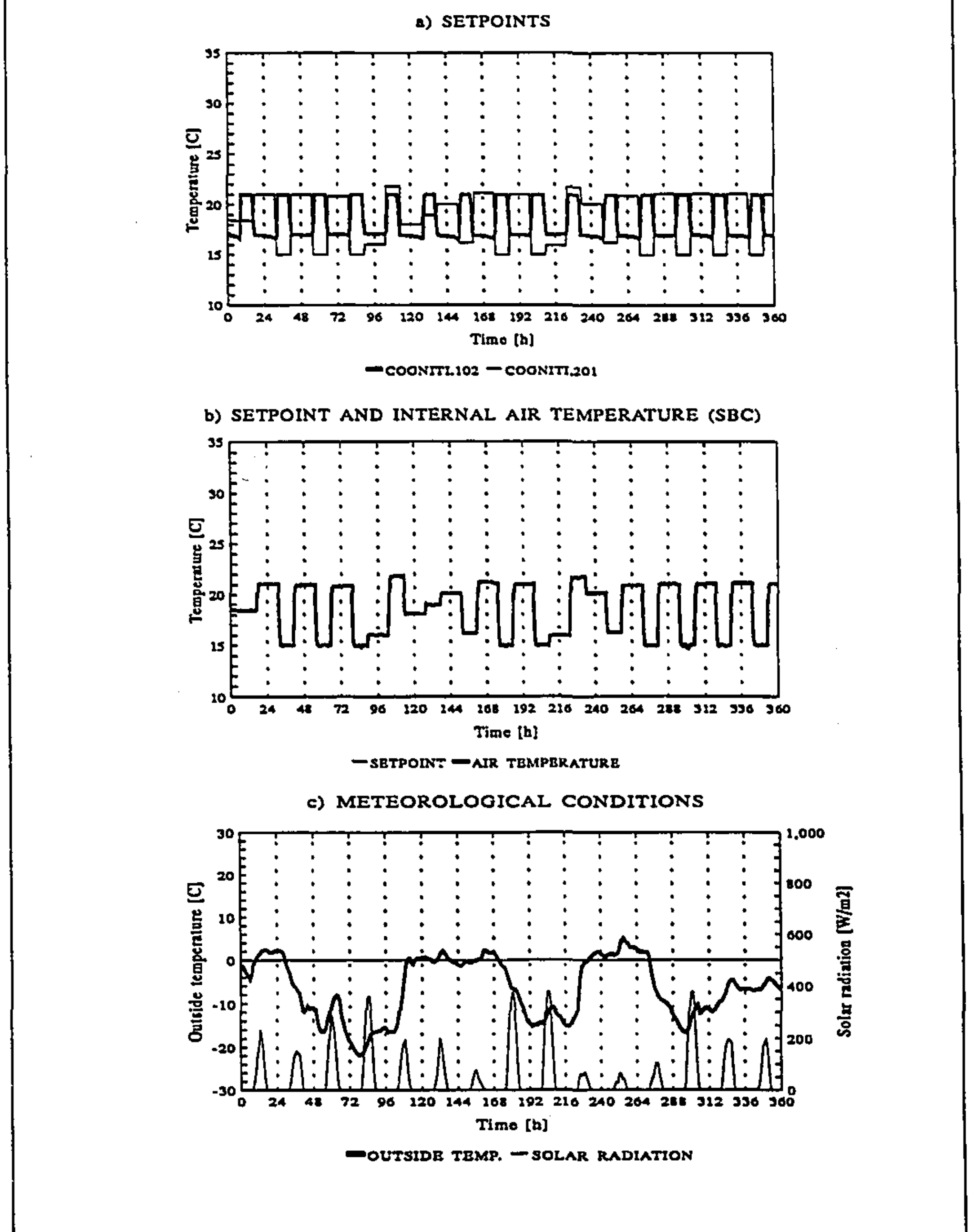

Figure VII.5 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of January 1983.

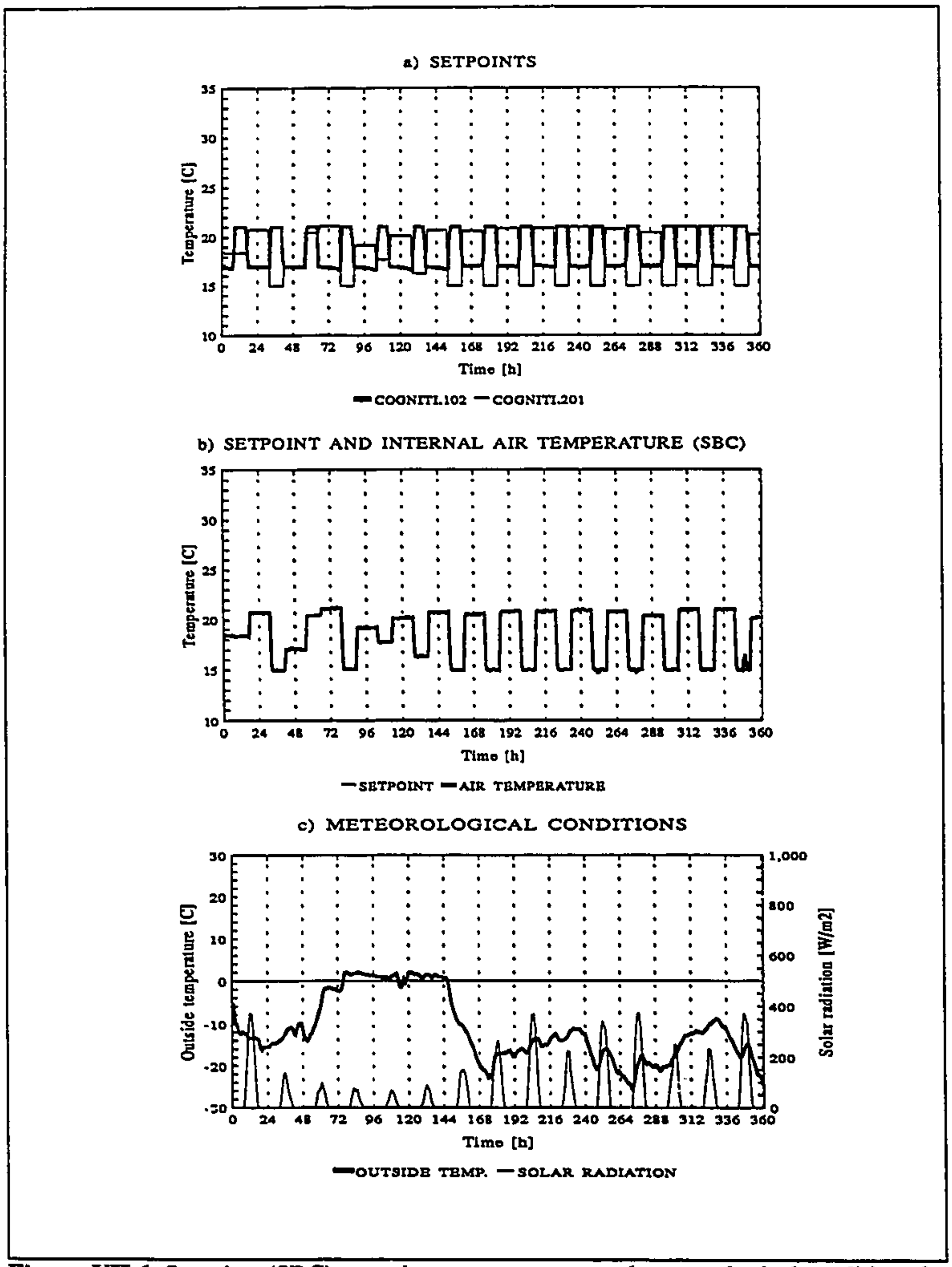

Figure VII.6 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of January 1984.

followed the setpoints generated for January 1982 and no overheating occurred. This was also true for 1983 and 1984 (Figures VII.5b and VII.6b).

As compared to January, the SBC setpoints varied more during the first 15 days of March (Figures VII.7, VII.8 and VII.9). The daily fluctuations of the setpoints generated by the SBC were frequently smaller than in January, but the day-to-day differences were larger. The difference between night and day setpoints was as small as 1°C and as large as 8°C. Overheating during the day occurred frequently (Figures VII.7b, VII.8b and VII.9b). The SBC anticipated overheating (with the help of its simulations) and compensated by picking a low setpoint for the night preceeding overheating. For the three years studied, it was observed that the SBC maintained very high setpoints during the night (e.g., 23°C) when the outside temperature was low and solar radiation intensity during the following day was also relatively low. This was often observed in March 1984, when overheating was often negligible (Figure VII.9b).

During the first 15 days of May 1982, the temperature setpoints were often lower during the night than during the day (Figure VII.10). This was true also for May of 1983 and 1984 (Figures VII.11 and VII.12). In this way, in comparison with the regime maintained in January, the temperature regime was similar to that traditionally maintained in greenhouses. Overheating occurred almost every day due to the relatively high solar radiation intensity and outside temperature. Since the ANN had predicted this overheating, the controller maintained low temperatures during the preceeding night to reach the required 24-hour temperature integral. In a few instances, it was observed that the setpoint was maintained approximately constant during successive 24-hour periods. When this constant temperature was low (e.g., 15°C, between 288 h and 336 h in May 1983), this corresponded to very high levels of overheating during the day (Figure VII.11). Inversely, when the temperature setpoint was at a constant high value (e.g., 20°C, between 150 h and 220 h in May 1983), this corresponded to days when there was no overheating during the day. Finally, it was observed that with the SBC, the daily minimum values for the

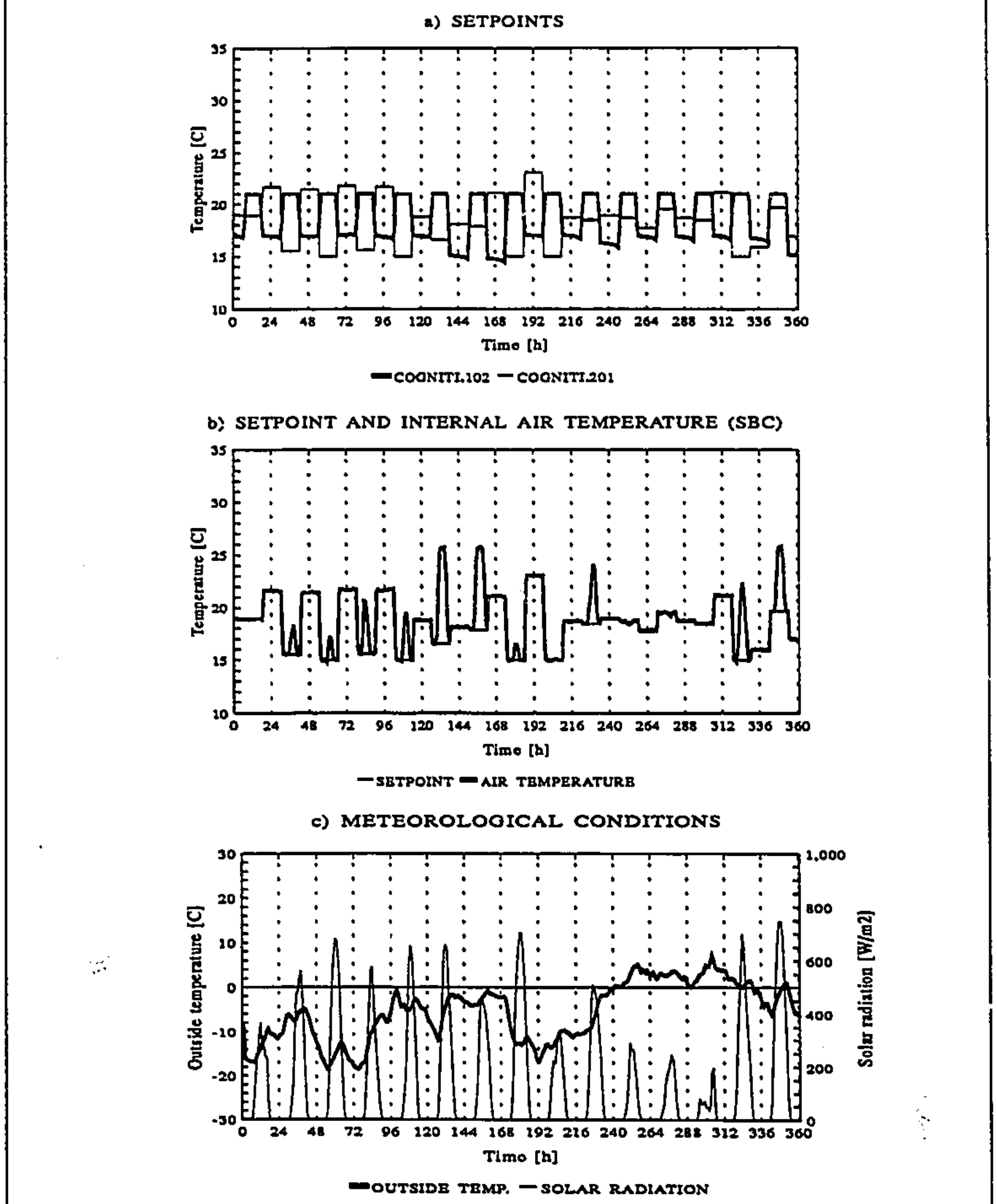

Figure VII.7 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of March 1982.

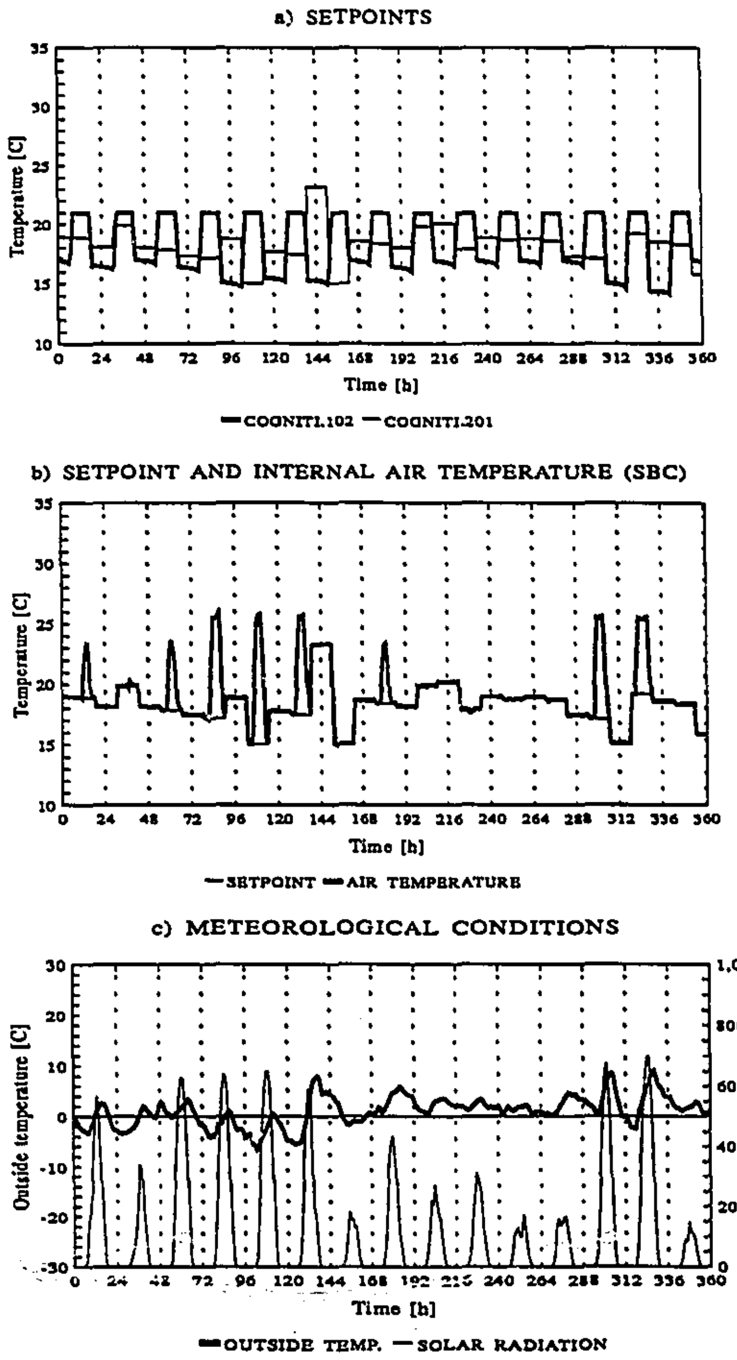

Figure VII.8 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of March 1983.

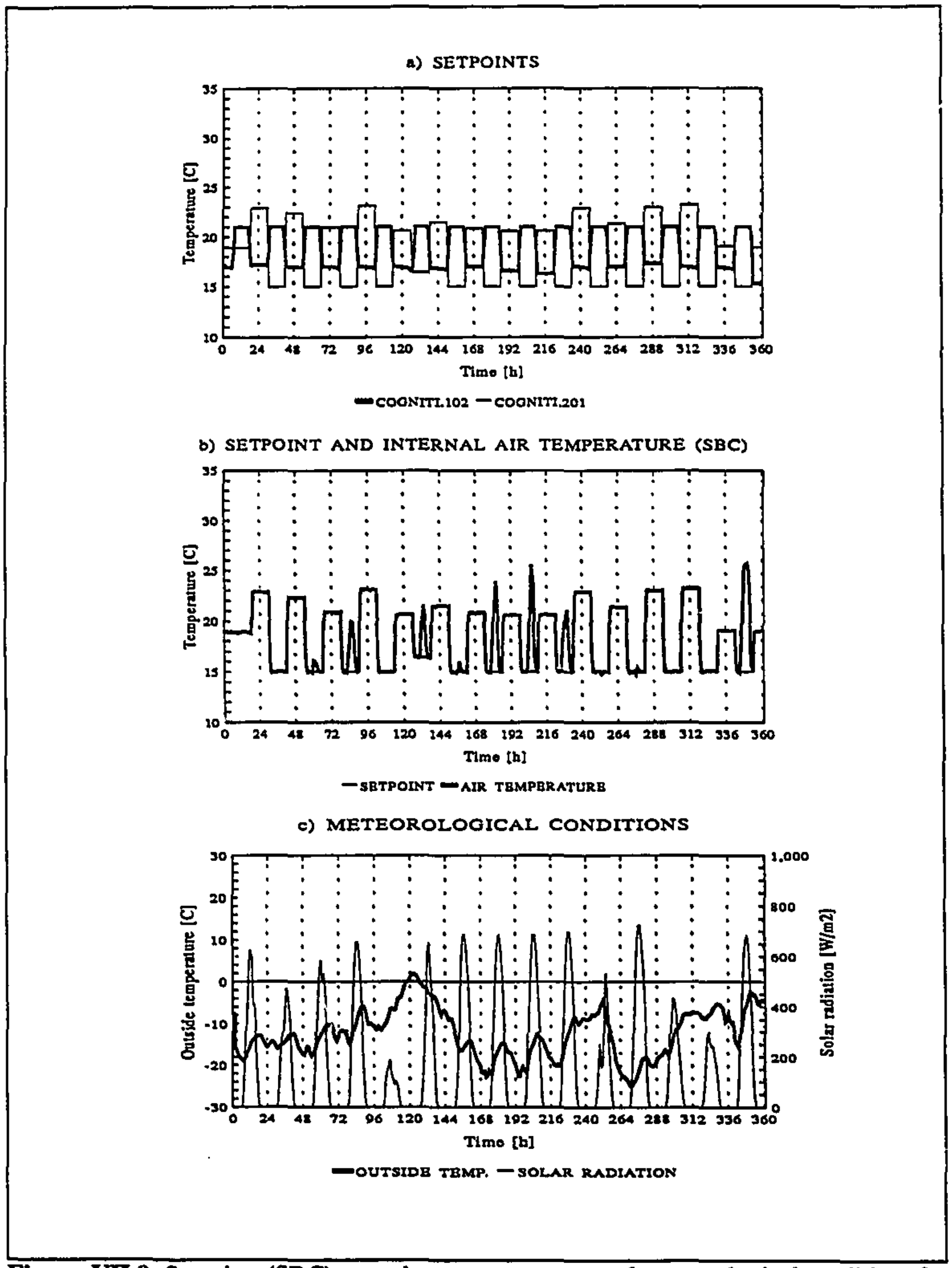

Figure VII.9 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of March 1984.

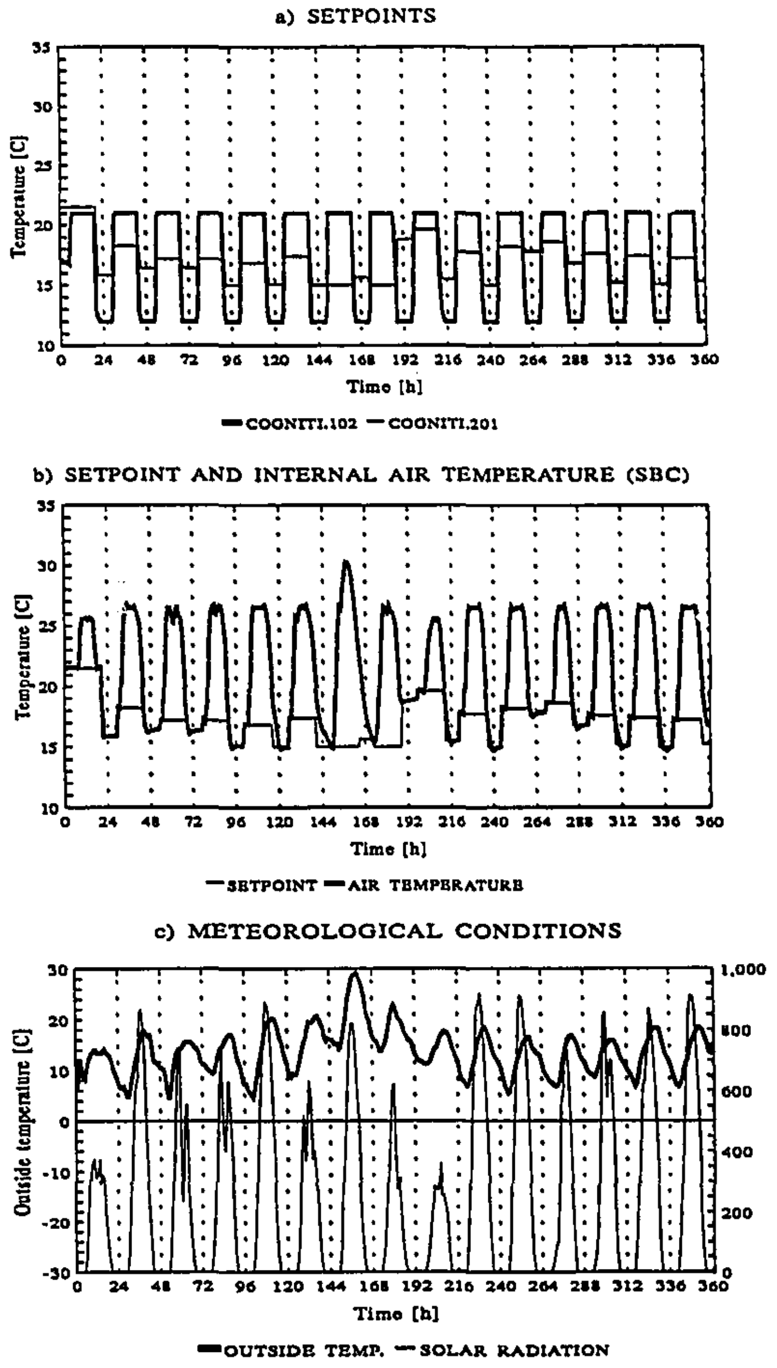

Figure VII.10 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of May 1982.

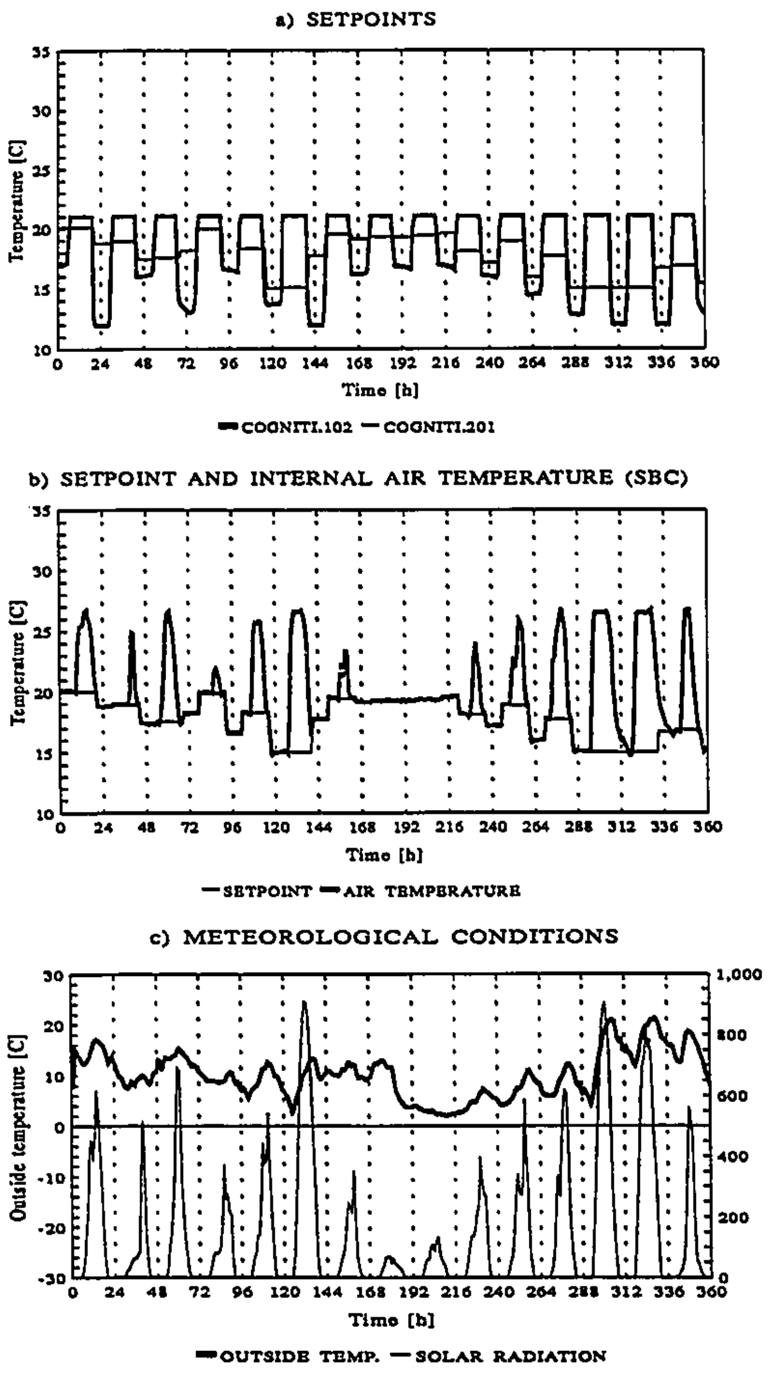

Figure VII.11 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of May 1983.

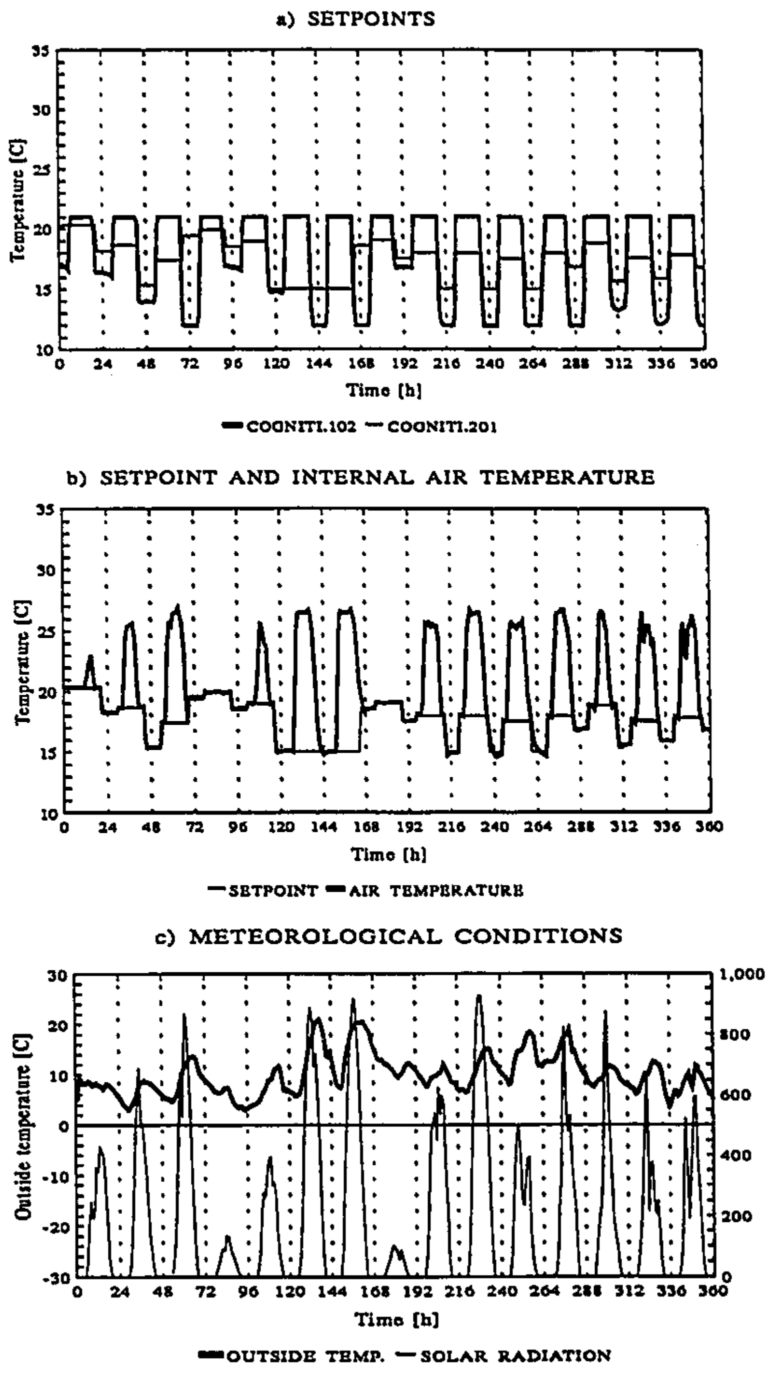

Figure VII.12 Setpoints (SBC), greenhouse temperature and meteorological conditions for the first 15 days of May 1984.

setpoints were generally higher than with COGNITI.102 (Figures VII.10a, VII.11a and VII.12a), even if a similar monthly average temperature was maintained by both controllers. This will be discussed below.

The monthly minimum internal air temperature, the monthly average internal air temperature and the total energy consumed for heating with the SBC and with the reference controller for 1982, 1983 and 1984 are compared in Table VII.5. The maximum temperatures are not reported here. They were similar for the two controllers since they were dependent on the ventilation, which was itself dependent on the ventilation setpoint and this was the same for both cases. Generally, the two controllers achieved similar average temperatures (and, therefore, the same temperature integral), but the SBC required 5.7% to 14.3% less energy to accomplish this. The total energy requirement over the five month simulation period was reduced by 7.1%, 7.1% and 7.7%, for the years 1982, 1983 and 1984 respectively. These values correspond in magnitude to values reported in the literature and to what was calculated in Appendix E. Thus, the SBC was able to adapt the temperature regime to the anticipated meteorological conditions so as to decrease the energy consumption. If the energy requirements of the SBC are compared with those of the traditional controller used in Appendix D (COGNITI.101) and listed in Table VII.2, the reductions for the months of March, April and May 1982 are 12.0%, 13.7% and 35.3% respectively. In this comparison, for the five months from January to May, the total reduction of the energy requirement is 9.8%.

The temperature regime maintained with the SBC led to the lowest energy requirements, which was the desired goal. This was done based on the assumption that growth would not be different for different temperature regimes that lead to the same temperature integral. However, it seems that, with the functional greenhouse system as installed and used, the growth rate as calculated by the crop model (based on SUCROS87) was slightly influenced by the nature of the temperature regime. Monthly averages of daily photosynthetis and daily respiration for January to May 1982, 1983 and 1984 are shown

Table VII.5 Results obtained with the simulation-based controller (COGNITI.201) and the reference controller (COGNITI.102) for 1982, 1983 and 1984.

	COGNITI.102			COGNITI.201			Diff. Energy for heating [%]
	T_{min} [°C]	T_{avg} [°C]	Energy [MJ/m^2]	T_{min} [°C]	T_{avg} [°C]	Energy [MJ/m^2]	
1982							
January	15.6	18.4	389.1	14.6	18.3	365.7	6.0
February	15.3	18.7	255.0	14.7	18.8	234.4	8.1
March	13.1	18.9	193.6	14.8	19.1	177.8	8.2
April	11.7	19.5	131.9	14.7	19.6	123.2	6.6
May	11.7	21.5	23.6	14.7	21.5	21.4	9.3
Total			993.1			922.5	7.1
1983							
January	16.3	18.4	304.5	14.7	18.6	285.9	6.1
February	15.4	18.7	230.3	14.7	18.9	215.3	6.5
March	14.0	18.9	183.5	14.8	19.1	170.6	7.0
April	11.6	19.2	126.9	14.6	19.2	117.7	7.2
May	11.6	20.1	58.9	14.7	20.0	50.5	14.3
Total			904.2			840.0	7.1
1984							
January	16.0	18.4	334.0	14.7	18.5	314.8	5.7
February	14.8	18.6	216.8	14.8	18.8	201.8	6.9
March	13.8	18.9	250.4	14.8	19.0	227.2	9.3
April	11.7	19.5	89.9	14.7	19.6	85.4	5.0
May	11.7	20.3	53.7	14.6	20.3	48.5	9.7
Total			944.9			877.7	7.7

in Table VII.6. While the differences in photosynthetic rates were null or very small between March and May for all three years, COGNITI.102 generally led to higher photosynthetic rates than COGNITI.201 in January and February. The respiration rates were similar for both temperature regimes during all months for the three years considered. Different photosynthetic rates in combination with similar respiration rates led to slightly different growth rates with the two temperature regimes even though the same

Table VII.6 Results obtained for crop in 1982, 1983 and 1984.

	COGNITI.102		COGNITI.201	
	Photosynthesis [g(CO$_2$/m^2·d]	Respiration [g(CO$_2$/m^2·d]	Photosynthesis [g(CO$_2$/m^2·d]	Respiration [g(CO$_2$/m^2·d]
1982				
January	10.3	2.6	9.6	2.5
February	15.1	2.7	14.5	2.7
March	20.4	2.8	20.0	2.8
April	23.2	2.9	23.1	3.0
May	28.8	3.5	28.8	3.5
1983				
January	9.2	2.6	8.8	2.6
February	14.3	2.7	13.9	2.7
March	18.5	2.8	18.3	2.8
April	18.5	2.9	18.5	2.8
May	23.5	3.1	23.6	3.1
1984				
January	9.6	2.6	9.1	2.6
February	12.9	2.7	12.6	2.7
March	20.3	2.8	19.8	2.8
April	22.3	3.0	22.3	3.0
May	24.9	3.2	24.9	3.2

temperature integral was achieved. This was due to the crop growth model used which may or may not reflect physical reality accurately.

VII.4 MAXIMIZATION OF NET INCOME

A series of four experiments in which the goal of the SBC was to maximize the net income was also performed. The objective was to illustrate how the SBC might be modified in the future to pursue research on simulation-based control. The simulations were done for the month of March 1983. The net income obtained with the reference controller and with the various versions of the SBC are listed in Table VII.7. These values were calculated as described in Section VI.4.

In the first simulation, the SBC followed a procedure similar to that used during the minimization of energy requirements. At the beginning of each night, the SBC generated various sequences of setpoints that led to a 24-hour temperature integral similar to that produced by COGNITI.102. Then, it chose the scenario that would furnish the largest net income. This controller is referred to below as COGNITI.202. In its decisions, the SBC had to take into account the predicted net assimilation rates for the 24-hour period being considered. The ANN predictions and the values that occurred during the first two weeks of March 1983 are shown together in Figure VII.13. The ANN predictions were generally very good, except during days when very high photosynthetic rates occurred. The setpoints chosen by COGNITI.202 and the internal air temperature (Air 1) are shown in Figure VII.14 for the first 15 days of March 1983. The fluctuations of the setpoints generated by COGNITI.202 were generally larger than with COGNITI.201 (Figure VII.8). When solar radiation intensity was high enough to create overheating, the setpoint chosen by COGNITI.202 was higher during the daytime periods than during the nighttime period. For the days with relatively low solar radiation intensity, the SBC maintained a higher setpoint during the nights. It can be seen in Table VII.7 that the net income with COGNITI.202 was increased slightly compared to COGNITI.102.

Table VII.7 Net income values obtained with the reference controller and the various versions of the SBC.

	Cost for heating [$/m^2]	Income from fruit sale [$/m^2]	Net income [$/m^2]
COGNITI.102	2.07	11.97	9.90
COGNITI.202	1.93	11.86	9.93
COGNITI.203	1.91	11.87	9.96
COGNITI.204	1.92	11.84	9.92
COGNITI.205	1.69	12.04	10.35

In the second experiment, the procedure used by the SBC (COGNITI.203) was the same as that used by COGNITI.202, except that the minimum allowable setpoint was lowered from 15°C to 12°C. It can be seen in Figure VII.15 that a setpoint of 12°C was generally chosen by the SBC during the nights preceding periods of high overheating. This led to a further small increase in the net income (Table VII.7).

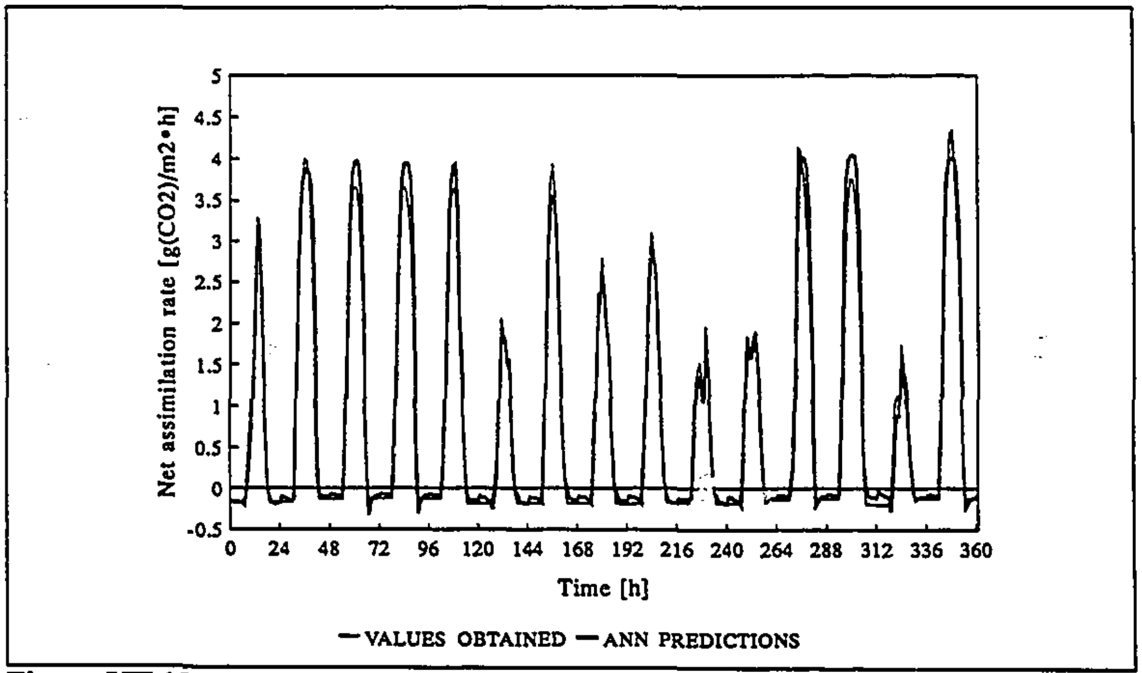

Figure VII.13 ANN predictions and values obtained (functional simulation) for the net assimilation rate, from the 2nd to the 16th of March 1983.

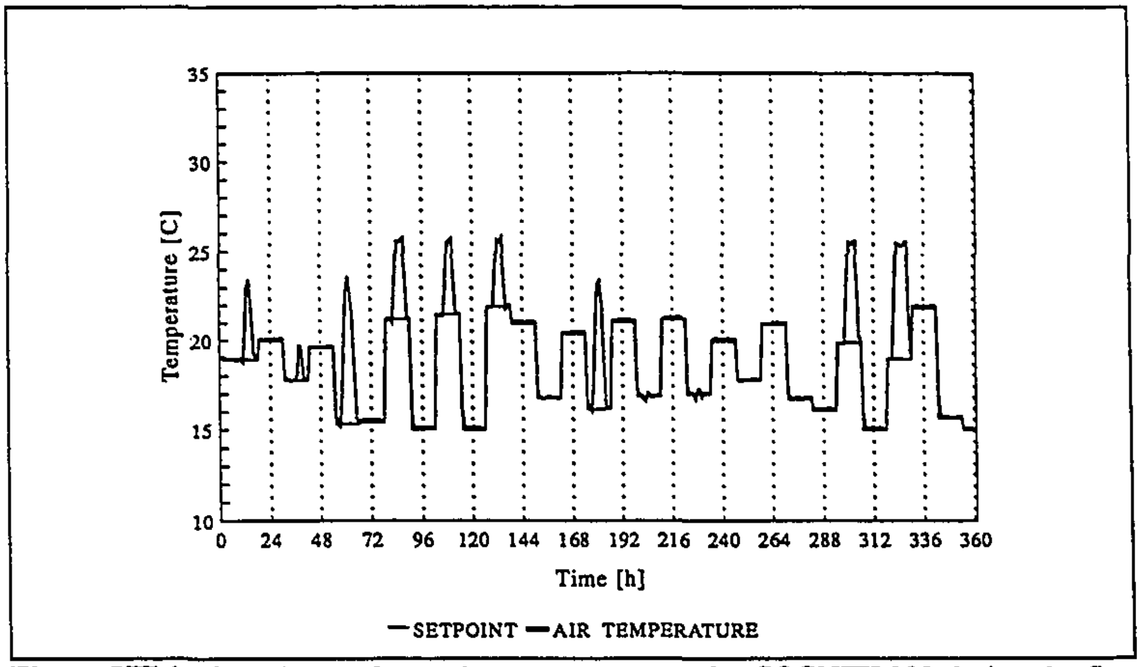

Figure VII.14 Setpoints and greenhouse temperature for COGNITI.202 during the first 15 days of March 1983.

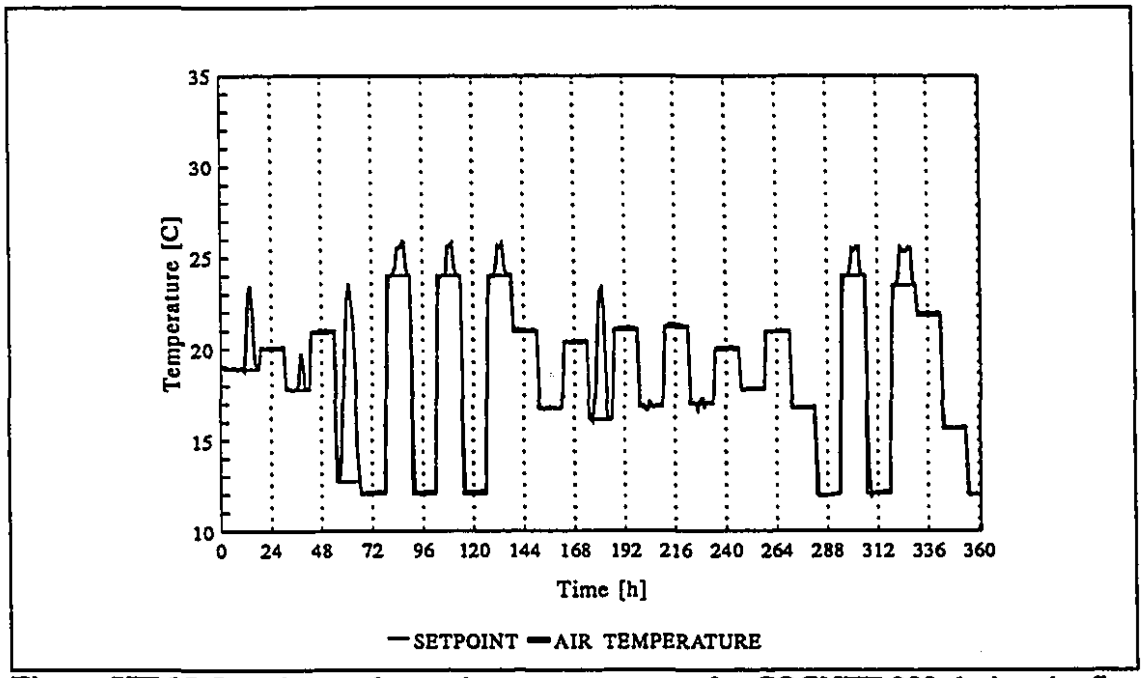

Figure VII.15 Setpoints and greenhouse temperature for COGNITI.203 during the first 15 days of March 1983.

The third experiment was executed with COGNITI.204, which had the choice, for the daytime periods, between a setpoint that either was constant or followed a sine curve. When the SBC chose to follow a sine curve, this was generally done for a period following a night with a high setpoint (Figure VII.16). Also, the peaks of the sine curves always corresponded to the minimum setpoint of the 24-hour period under consideration by the SBC. Although the SBC had predicted that the chosen setpoints would lead to a larger net income than with COGNITI.202 (half of the regimes tested by COGNITI.204 were identical to those tested by COGNITI.202), the resulting net income was finally slightly lower than with COGNITI.202 (Table VII.7).

In the fourth experiment, the constraint on the SBC to achieve one specific value for the 24-hour temperature integral was relaxed. The SBC had the possibility of choosing

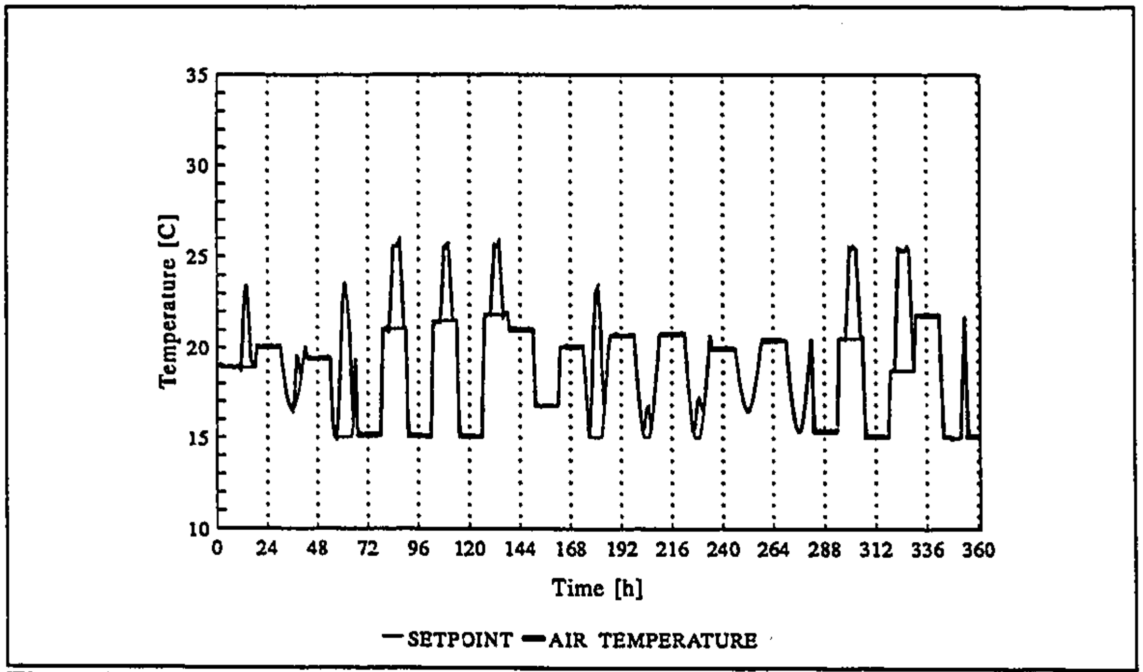

Figure VII.16 Setpoints and greenhouse temperature for COGNITI.204 during the first 15 days of March 1983.

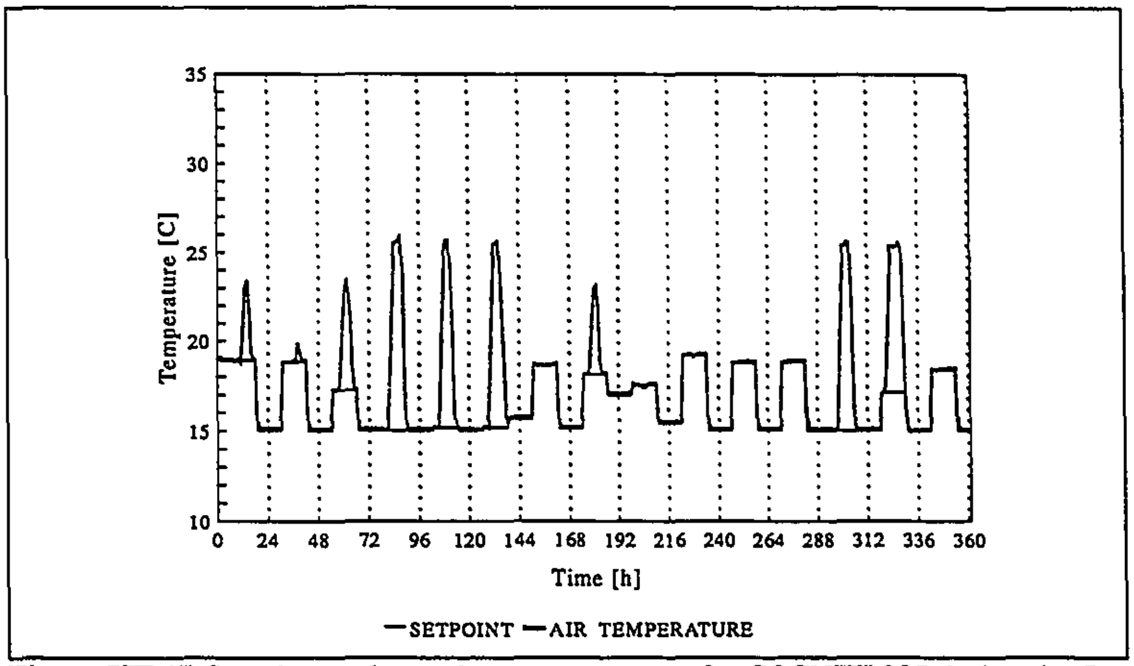

Figure VII.17 Setpoints and greenhouse temperature for COGNITI.205 during the first 15 days of March 1983.

amongst various values for the integral. The maximum deviation of these values from the 24-hour average of the temperature integral of the reference controller was ±2°C. As is shown in Figure VII.17, the allowable minimum setpoint value of 15°C was frequently chosen by COGNITI.205. This temperature regime led to the largest net income of all regimes tested, due to both a decrease of the heating cost and an increase of the income from fruit sale (Table VII.7).

Although the various versions of the SBC did not lead to very large increases in the net income compared to the reference controller, some valuable information was obtained from the experiments. It was observed that the optimization of a variable such as "net income" was very context sensitive, because such a variable is itself a function of many other variables (energy consumed, cost of energy, amount of fruit produced, etc.). Also, it was observed that the results depend largely on the precision of the predictions of the models used in the simulation-based decision-making processes. The research also indicated that setpoints might be generated in many possible ways, and that more flexibility (e.g., temperature integral, allowable minimum setpoint) might increase the benefits of such control strategies. Also, it was noted that the investigation and analysis of many scenarios by a SBC could be done rapidly with the use of a neural model. In some experiments, the SBC was executing and analyzing at least 120 simulations each day in a short physical time period.

VII.5 CONCLUSION

Two series of experiments were done with the prototype of a simulation-based controller (SBC). In the main series, the role of the controller prototype was to generate temperature setpoints so as to reduce the energy required for heating, while maintaining a specific average temperature. In the second series, which was carried out mainly to illustrate the possible use of various versions of the SBC, the goal was the maximization of the net income. In all experiments, the strategy used by the controller was based on the maintenance of different setpoints during the night and the day. At the beginning of each night, the controller generated a series of setpoints for the next 24 hours and then simulated the greenhouse response using a neural model, taking into account the anticipated meteorological conditions. The controller chose the scenario that led to the smallest amount of energy required for heating (or to the largest net income). In these simulations, the controller had ideal predictions of the meteorological conditions that would occur during the period under consideration.

Simulation-based control allowed the greenhouse system to adapt itself to the anticipated disturbances and to behave optimally in terms of energy requirements. This control approach reduced the energy consumption of the greenhouse system by an amount varying from 6% to 14%, thereby achieving its goal. These results indicate that there is a potential for saving energy in the greenhouse industry by manipulating the temperature regimes. However, further investigations must be undertaken to estimate the magnitude of the benefits that could actually be realized in less ideal contexts. First, the approach should be tested with smaller amplitudes of oscillations in the daily setpoint and with thermal screens of lower quality. Second, the use of smaller and more realistic data sets for the development of neural models should be investigated. Third, experiments must be done with uncertain weather forecasts. Fourth, experiments should be carried within physical greenhouses to validate the approach. Finally, the capacity of crops to integrate temperature must be further investigated and the crop models should be modified appropriately to reflect this capacity.

VIII. GENERAL DISCUSSION

VIII.1 THE DESIGN OF ENCLOSED AGRO-ECOSYSTEMS

In this research, the objectives were to 1) develop tools to aid in the design of enclosed agro-ecosystems and 2) use these tools to construct and test the prototype of a simulation-based controller. The greenhouse was used as the reference for the development of the tools, but the intent was that the different concepts and results emanating from this research could apply to any type of enclosed ecosystem. Three major tools were developed: a conceptual framework, a simulated greenhouse system and a simulation approach under OS/2. These tools are discussed in this section.

VIII.1.1 The conceptual framework

The conceptual framework that was developed comprises a general model of an enclosed agro-ecosystem with its inputs and its outputs, and a description of its control system. In this way, it constitutes a relevant tool for describing and analyzing enclosed agro-ecosystems and for developing appropriate control systems. The conceptual framework can be useful at many levels. First, it can be used in the initial phase of a project for the design of entirely new enclosed ecosystems. Second, it can serve to improve existing systems. Third, it can be employed to construct a simulation model to be used either during the preliminary phases of a design project or by a cognitive control system that would perform conscious control.

For the design of an entirely new system, the conceptual framework can help to establish the structure and the functionality of the system to be created. For example, one could first classify the inputs and the outputs into the categories that were defined (e.g., virtual disturbances, physical disturbances, supervised inputs). Then, the internal system components such as perceptors, effectors, subcomponents of the production encompassment, etc, could be determined as a function of the requirements of the production object. The goal tree could be elaborated to determine the characteristics of the associated control mechanisms such as their implementation level and their activity class. This would help in the design of an appropriate control structure.

The framework can also be useful in the analysis of existing systems, e.g., to improve their associated control system. The framework furnishes a method to decompose a system into its internal components and identify its inputs and outputs. It can also help in the analysis of the intrinsic characteristics of the system, such as its inherent regulation processes, which will, in turn, aid to create measures for the improvement of the system. For example, the framework could be employed to define the control mechanisms to be added. A similar analysis could help a designer to conceive a model of the overall system for simulation executed by the designer, or by the control system.

The overall framework was developed in reference to greenhouses, but it can be adapted to any enclosed agro-ecosystems by substituting some elements (e.g., the production object) and putting emphasis on some particular aspects. Because the concepts that were elaborated are general, it is also possible for them to be applied to open-field agro-ecosystems and to ecosystems in general. They could be particularly useful for the development and improvement of resource management systems (e.g., for animal population control), and this aspect deserves some consideration. Thus, the adaptation and the application of the conceptual framework to different types of ecosystem (i.e., agricultural and non-agricultural, enclosed and not enclosed) is expected to be an interesting research topic.

The framework that was developed is obviously incomplete and each aspect can be further investigated. It will serve as a basis for future discussion and development, and will be improved according to results obtained in future research. The aspects related to cognitive and conscious control will need special consideration and the implementation of complex controllers in both simulated and physical environments will be of help at this level. Also, the overall conceptual framework will improve by applying it within diverse contexts and, especially, to different ecosystems. This will help to detect lacunae, and to further generalize the concepts, thereby creating a more complete tool that will be better integrated and more coherent.

VIII.1.2 The functional greenhouse system

Simulation has been used for many decades in the preliminary phases of many design projects. It is an inexpensive approach allowing the reduction of investigation costs in comparison with physical experimentation. This is particularly true for the design of information-based systems, which are well-suited for development by simulation due to their virtual nature. Thus, simulation is very useful as an aid in the design of control systems in which decisions are based on extensive treatment of information and knowledge. For this reason, one objective of this research was to develop a simulated greenhouse system that would permit the development and analysis of various control approaches, during this project and after.

In the way it was developed and implemented on OS/2, the functional greenhouse system constitutes an excellent tool with which to develop control methods. In this project, it was used to develop the prototype of a simulation-based control system. It allowed the incremental development of the control routines and the study, in isolation, of their effects on the controlled subsystem. Such an incremental development would have been difficult, if not impossible, in a physical milieu. Also, simulation allowed the detection of many aberrations in the control procedures during their development, which would have been costly if they had been implemented directly in a physical environment. Simulation also permitted a narrowing of the domain of investigation for physical experimentation in relation to the development of optimal temperature control strategies. Simulation helped to determine factors to be considered during the development of neural models. The functional greenhouse system furnished an environment which was free of many on-line control constraints habitually found in a physical environment. For example, many operations that would contribute substantially to the total computing load were not required here (e.g., A/D and D/A conversions and communications with humans). As a result, simulation allowed the concentration on the research objective which was the development of a simulation-based controller. Finally, simulation can allow the testing of controllers of virtually any computing capacity. It will be possible to use the functional greenhouse system in the future development of more complex cognitive controllers.

As with any tool, the functional greenhouse system can be improved in many ways. One of the shortcomings of this system was the lack of humidity control, which should be changed early in future investigations. Also, the possibility to control and maintain high concentrations of CO_2 inside the greenhouse should be added to the system. High concentrations of CO_2 increase crop yields considerably, depending on factors such as climatic conditions. CO_2 control requires that a mass balance of its fluxes in the air of the greenhouse be established. This will constitute a relatively simple task, and a proposition for the balances to be added can be found in Chapter XI.

Many other aspects of the greenhouse system can be improved. For example, a crop development model should be added to the system to allow for functional simulations to be executed over a complete growing season. A method of simulating the opening of the thermal screen could be investigated further. As for research more specific to climate regulation, some attention should be given to the modelling of control equipment like heaters and ventilators. Also, the modelling of the flux involved in the moisture balances could be improved, especially for crop transpiration and condensation on greenhouse covers.

Finally, a need exists for the development of a methodological framework to design experiments by simulation. Such frameworks have existed for a long time for physical experimentations, e.g., to elaborate experimental protocols. These methodological frameworks are well documented and are taught in universities. A similar methodological framework should be developed for simulation. Such a framework may include hints about how to choose the required tools, how to limit the complexity of the models, when and how to do sensitivity analysis, how to interpret simulation results, etc.

VIII.1.3 Simulation under OS/2

One objective of this research was to create an approach for implementing a multi-module simulation structure within a multi-tasking operating system. The OS/2 operating system has proven to be quite satisfactory for the installation of a simulation structure which consists of several elements written in different languages and that need to inter-communicate and to remain synchronized. A number of tools were available in OS/2 that allowed processes to interact and these have proven to be useful for the purposes of this research. The synchronization approach that was developed made use of semaphores and allowed the complete control of the dispatch of the different processes. Named shared memory was a good mechanism for transferring data between modules.

The implemented simulation structure was modular, which eased the editing, compilation and linking of each component. Each component could independently access resources, such as memory, as they were required. It was possible to modify, with only minor operations, many aspects of the simulation, such as the inter-communication characteristics. Editing of each module was rapidly and easily performed. For example, it was very simple to interchange the many versions of the programs used in the module COGNITI. This permitted the simulation and comparison of different greenhouse control strategies. It was also easy to change the models used in the functional simulation. It was simple to shift from the model of the glasshouse equipped with a thermal screen to that of the greenhouse with two transparent covers. In future investigation phases, it will be easy to modify some modules for other variables to be controlled. For example, only minor modifications will need to be done to allow the control of the CO_2 concentration and the position of the thermal screen by the Pavlovian controller instead of by the module GHOUSE.

In this research, the role of the cognitive controller was restrained to the derivation of set-point paths, by simulation, for the Pavlovian machine. In a more sophisticated arrangement, other mechanisms can be added to the cognitive controller. These mechanisms will act in parallel inside independent processes and many will execute "continuously". All these decision-making processes, executed in parallel, will create the impression of an ongoing multiple processing stream. With such a controller it will not be possible to adopt the approach used for this project. In this project, functional time flow was simply stopped when module COGNITI was executed and all its activities had to be finished for the functional simulation to be pursued. In this sense, "infinite computing capacity" was assumed. However, for the functional structure to include a more complex controller, characterized by an ongoing processing activity, a different approach is required to allow for functional time to flow in parallel. This can be done by adding a mechanism that will simply "suspend" the cognitive information processing during a small physical period at each functional time increment, to allow for the other modules to be executed. This mechanism can be implemented by a "priority adjustment" method under OS/2, as will be discussed in Chapter XI. Due to its modularity and the way it was conceived, it is expected that only minor modifications will be necessary to implement such a method into the simulation structure that was implemented.

The procedure used to implement the simulation into OS/2 was acceptable, but improvements are required for further investigations. An important adjustment will be to allow the MANAGER module to directly control the execution of the COGNITI module. This will require the construction of a library of external C routines that will make calls to OS/2 API's from within GURU 3.0. COGNITI will then gain direct access to system semaphores and shared memory segments.

VIII.2 IMPLEMENTATION OF CONSCIOUS CONTROL

In the conceptual framework, conscious control was defined as a subcomponent of cognitive control and as being performed by a system that refers to representations of itself in its decision-making processes. The suggested approach for a rudimentary implementation of conscious control was to give a system the capacity to access models of itself to run simulations.

One objective of this research was to investigate the use of simulation by a greenhouse climate control system. The specific goals of this part of the investigation were to: 1) analyze the behavior of the greenhouse when subjected to this form of control, and 2) determine elements of a design methodology for such control systems. Depending on the experiment, the role of the simulation-based controller (SBC) was to determine temperature setpoints that would minimize the energy requirements for greenhouse heating or that would maximize net income. The setpoints were determined for a period of up to 24 hours prior to their use for regulation. Thus, the SBC was practising, in an extended fashion, a combination of anticipatory control and optimal control. This approach allowed the functional greenhouse system to adapt itself to anticipated disturbances and to behave in a more desirable manner than with a controller using a traditional approach.

VIII.2.1 Potential advantages of simulation-based control

A simulation-based controller gives a system the capacity to prepare itself for future disturbances, when it has a notion of what is anticipated. The system can investigate the effects of different control alternatives on the behavior of the controlled system prior to their implementation. This allows the controller to choose the actions it considers would best fulfil its goals. A simulation-based control system also has the capacity to continuously revise its past decisions by executing simulations based on conditions that have occurred. In this sense, such a system can be self-adjusting. Thus, the implementation of a simulation capacity in a controlling system potentially gives it many advantages over conventional ones, by conferring upon it the capacity to adapt to new situations. Because it increases the adaptive capacity of a system, the implementation of simulation-based control in other types of enclosed ecosystems is promising.

Another advantage of a simulation-based control approach is that it allows a control system designer to let a computer do the simulation experiments, rather than doing it himself/herself. With the traditional approach, a designer must anticipate and simulate a whole range of possible sets of disturbances and system reactions in order to develop the appropriate control algorithms. For many complex production operations, the anticipation of all conditions that can occur can rapidly become a tedious task, if not an impossible one. For example, such a situation was experienced in this project when many simulation experiments were done to try to conceive strategies for the determination of setpoints that would minimize energy requirements for heating. At the beginning, the appropriate conditions for obtaining benefits with various approaches were unknown. Many simulation experiments were done, which was time consuming due to the large number of manipulations that were necessary (e.g., choice of meteorological sequences, preparation of files, analysis of results). In such a context, it can be advantageous to let a controller run the simulation experiments itself, as was finally done in this project. A controller can even work continuously during nights and week-ends. Simulation-based control may also be advantageous when some components of the controlled systems are modified or when new knowledge is acquired about some control aspects. The control system may itself adapt to the new context without the necessity for a designer to test and implement new control routines.

In this project, the simulation-based control approach was used for the determination of setpoints, but a similar form of control will certainly be useful for setpoint realization or regulation. For example, it would be possible to use the anticipatory capacity of simulation-based control, on a short term basis, to complement the PID control of hot water-distributed heating systems or for the operation of ventilation systems. This might be done by simulating the behavior of the greenhouse according to different control actions and to the meteorological conditions expected during the following period, e.g., during the following 15 minutes. The role of the SBC was also restrained in this project to the control of only one variable, the temperature. However, the use of a similar approach will certainly help to control other factors such as CO_2 concentration, artificial light and hydroponic solution composition. The simulation-based approach could be particularly helpful in multivariable context, i.e., for the simultaneous control of many variables.

In the conceptual framework, the addition of simulation capacity to a control system was considered as an approach to implement conscious control at a primary level, in a rudimentary fashion. Even if it is possible to develop a whole theory which is exclusive to "simulation-based control", there are advantages to trying to put it in a more global conceptual context. By inscribing it into the context of cognitive and conscious control, simulation-based control can be placed in the perspective of other functions such as learning, reasoning and problem-solving. Also, the development of a conceptual framework based on human mind functions and other attributes related to intelligence is very powerful. It may not be possible to recreate the exact same faculties as those exhibited by humans, but trying to mimic those faculties helps to explore new possibilities for system design and to engineer new procedures and devices. This can potentially expand the present capacities of existing systems. For example, artificial neural networks (ANN's) were created when researchers tried to mimic the structure of the human nervous system. The structure of ANN's is very simple and in this sense, it is not comparable with the human neural network. However, ANN's enable one to do things that were not previously possible and to create systems that are able to adjust themselves to new situations. A similar approach should be adopted to develop a theory on conscious control that is based on perceptions of the characteristics of human consciousness.

VIII.2.2 Multivariable conscious control

Many variables need to be considered simultaneously in the control of enclosed ecosystems. This is important since the global optimum continuously varies in such systems. For example, in a greenhouse, the global optimum varies as a function of the meteorological conditions and the characteristics of the crop (cultivar, growth phase, etc). Multivariable control is, therefore, an important aspect to be considered in these systems. In this research, the simulation-based approach was used only for the control of the temperature, but the method that was developed can be extended to a multivariable situation.

In this project, the SBC looped through different temperature regimes and chose an 'optimal' regime. In this process, the SBC was doing some form of numerical optimisation. A similar approach can be extended to more than one variable by nesting a new loop for each variable considered. For example, if CO_2 concentration was to be considered with temperature, different trajectories of CO_2 setpoints should be tested in combination with each possible temperature setpoint trajectory. However, it must be considered that the number of simulations increases rapidly for each added variable. For example, if ten temperature setpoint trajectories are tested in combination with ten CO_2 setpoint trajectories, the total number of simulations will be 100. With a third variable that can also follow ten different trajectories, the number of simulations would increase to 1000. With a large number of variables, the number of simulations to be executed will be tremendous. However, the power and capacity of computers are increasing rapidly. Also, neural models allow the rapid execution of a large number of simulations because calculations are performed very fast with ANN's. Finally, it is possible to use artificial intelligence techniques (e.g., rule-based expert systems) to constrain or limit the domain of the investigation to be done by simulation, as will be discussed in Section VIII.2.3. These aspects all need to be investigated further in order to develop a theory on multivariable conscious control.

VIII.2.3 Simulation-based control mechanisms

A controller performs simulation-based control when it can execute simulation-based decision-making processes (SBDMP's). A certain number of elements are required to allow for the execution of SBDMP's. First, the goal of the decision-making process must be clearly defined. Second, different control actions must exist, to be tested by simulation. Third, a model of the controlled system is required. Fourth, the disturbances affecting the controlled system during the simulated period must be known, or must be at least estimated. Fifth, a mechanism must exist to implement the simulation-based decision-making process. In the conceptual framework, such a mechanism was called a "simulation-based control mechanism". The decision-making process in a simulation-based control mechanism was determined to be composed of four main phases: 1) determination of the domain of simulation, 2) execution of the simulations, 3) evaluation of the simulation results and 4) final decision-making.

A SBDMP would be relatively straightforward if the model of the controlled system constituted a perfect representation of this system, and if the disturbances were perfectly known. However, uncertainty always resides, up to a certain degree, at these two levels. A good approach for dealing with this uncertainty is to integrate it into the SBDMP, instead of assuming that it does not exist. Treatment of the uncertainty can be done during the four phases of a SBDMP, as will be seen below.

VIII.2.3.1 Determination of the domain of simulation

The first phase of a SBDMP consists of the determination of the simulation domain, which is itself related to all the "materials" used in a simulation experiment, according to the specific objective. Decisions must be made for the choice of the control alternatives to be tested, the appropriate predictive model(s), the model parameters, the initial conditions, the variables to be observed, the simulation parameters, and the input sequences. Simulation domains can vary from simple to very complex, depending on the degree of cognition about the modelled system, and the disturbances that affect it. A model is never more than a representation of a system, and not the system itself. Some uncertainty always remains after a modelling process.

Depending on the objective of a study, it might be sufficient to assume that there is no uncertainty associated with a model, mostly if the modelled system is fairly well known. If the disturbances are also well known, different control alternatives can then be tested by simulation with the model and the set of known inputs. However, in many cases, the exact behavior of the system, with respect to a specific set of known disturbances, is quite uncertain. There exist many possibilities for the consideration of the model uncertainty in a SBDMP. One approach consists of taking into account in the other phases of the SBDMP, the limits of the model (e.g., during the evaluation of the simulation results). Another approach consists of investigating the various behaviors that a system could possibly exhibit with respect to the set of disturbances. For example, it is possible to use stochastic models and randomly generate many possible behaviors in response to disturbances. It is also possible to use a similar approach with many different (deterministic) models of the system. The same approach can be adopted when the initial state of the system is not-well known (e.g., due to bad measurements), which can have great consequences on the course of the simulations, especially with chaotic models.

Independent of the uncertainty associated with the model used in a SBDMP, it is possible that the disturbances are not well known. Such a situation occurs when decisions are made regarding a future context, for which the exact disturbances are unknown. This can also be due to limitations in the perception network that lead to poor representations of the system's environment. A valid approach for these cases consists of generating a whole set of possible disturbance paths, and using them as inputs for a set of simulations.

If it is possible for the model to react in different ways to a given set of disturbances, and if different sets of disturbances might possibly occur, the number of simulations can increase rapidly. For example, if a model can react in ten different ways to one set of disturbances, and if ten disturbance scenarios are possible, this leads to 100 possible behaviors. When the objective of the simulation process is to compare the response to different control actions, the number of simulations can become enormous. For example, in a multivariable context similar to that presented in Section VIII.2.2, where three variables can each follow 10 different trajectories, 100000 simulations (10^5) would need to be executed and analyzed for an investigation of all possible combinations of setpoints, system behaviors and disturbances.

With ANN's, it is possible to run a large number of simulations in a reasonably short time. Moreover, a possible avenue is to develop mechanisms that would narrow the domain of simulation. One of these is the possibility of using expert systems, as discussed in Chapter II. The conception of these expert systems would be based on experience and would help, for example, to eliminate combinations of scenarios that are less plausible. Rule sets could also make associations between control actions that must be tested according to anticipated disturbances, which would also diminish the number of scenarios to be tested. It is also possible to investigate another avenue, whereby artificial neural networks would be trained to recognize situations and would help to narrow the domain of investigation. In this sense, ANN's could help to transfer some of the decisions from a "conscious" level to a "reflex" level (e.g., from cognitive to Pavlovian).

VIII.2.3.2 Execution of the simulations

Once a simulation domain is established, the simulations can be executed. During simulations, the predictions of the model used are not always good. For example, during the development of the SBC prototype, it was observed that the neural model was sometimes predicting a negative heating load. At least two alternatives exist to deal with such awkward predictions: 1) recalibrate the model (e.g., retrain an ANN) or 2) develop rule sets to adjust the predictions during simulations. With each alternative are associated a series of costs and advantages that need to be evaluated in order to make a decision regarding an option. For example, the choice can depend on the availability of training data and the complexity of the model. For the situation described above, it was only required that a rule be added to adjust the prediction of the neural model during subjective simulations. This rule simply specified that, when the heating load was predicted to be negative, it was set equal to zero. This improvement was simple and was certainly advantageous in comparison with retraining the neural model. Many other rules could also have been added. For example, when the outside temperature was very low, the opening of the thermal screen was increasing the heating load drastically, thereby creating a peak. The neural model had problems predicting such peaks. Some rules could have been added to improve these predictions. One such rule could have been, in fuzzy terms:

> IF outside temperature is very low and tne thermal screen is being opened
> THEN increase considerably the predicted heating load.

The situations presented here are simple. Nevertheless, they suggest that, in some cases, rule sets, and even complete expert systems, could become an important complement to models to adjust predictions during simulations. Such rule sets can be conceived when it is known where and how models fail. This must be learned by experience.

VIII.2.3.3 Evaluation of the simulation results

When simulations have been completed, the output sequences can be analyzed and interpreted. The goal of the analysis can be to verify that simulation results fulfil some requirements. The evaluation can also serve to prepare the output data for use in the final phase of decision-making. The analysis must ideally take into account the limits of the model(s) and the underlying assumptions. It may also consider the uncertainty associated with the results, which may be due to the model itself and/or the disturbances.

In this project, each time a simulation was finished, it was evaluated to see if the temperature integral had been achieved with the setpoint trajectory that was tested. If the simulation did not lead to the required integral, the setpoints were adjusted and the simulation was repeated. Rules limited the new setpoints between minimum and maximum values. Other rules could have been added to do a more in-depth analysis of the results and, consequently, produce a more optimal approach for adjusting the setpoints. For example, when the temperature integral was predicted to not have been achieved for the following 24-hour period due to overheating, only the night setpoints could have been adjusted before rerunning the simulation. In a different context, rules could be conceived that would analyze the predicted humidity or crop thermal stresses, if the neural model was supposed to predict values for these variables. Once again, the development of complete expert systems could be justified in some contexts for the evaluation of the simulation results.

VIII.2.3.4 Final decision-making

Once the analysis and interpretation of the results are completed, the whole simulation experiment is evaluated to verify which control alternative should be adopted. In the primary experiments for this research, a simple expert system chose the scenario where the energy requirement was minimum. In subsequent research, rules may be added to do more complex analysis. For example, some of these rules may evaluate the impact of the temperature trajectories on crops. The analysis could be further complicated if market conditions and a multitude of other factors, such as humidity, are considered in the final decision. Rules that take into consideration uncertainty associated with models and disturbances could also be added at this level of analysis.

VIII.2.4 Partial conclusion

The insertion of expert systems can be done at many places within a SBDMP to improve the final decisions. This is in agreement with the conclusions of many authors, as was seen in the literature review (Chapter II). In this project, only simple rule sets were implemented, but they indicated a potential for the implementation of larger rule sets in more complex applications.

VIII.3 ARTIFICIAL NEURAL NETWORKS

VIII.3.1 Advantages of ANN's

In this project, ANN's have proven to be an appropriate technology with which to model a system and for implementation into a simulation-based controller. The simulation of the greenhouse system behavior over 24-hour periods was very fast in comparison with the time that would have been required if a traditional procedural model had been used. The rapidity of the calculations with ANN's suggests that they can be very useful for numerical optimization, since they allow rapid looping through diverse contexts.

An important advantage of ANN's is that they possess learning capacity. Therefore, they constitute a technology particularly well suited for the implementation of conscious control. First, ANN's can be used to analyze and revise past decisions and past contexts, a process by which they learn how to best behave when they will again face similar conditions. Second, neural models used in the simulation-based decision-making processes can be retrained continuously when new data is acquired, so as to be adapted to new situations. In this case, the approach with ANN's can be advantageous compared to traditional least-square adjustments of model parameters. With the latter, all data must be fed simultaneously, while with ANN's the new data can be simply added and used for further training.

A further advantage of ANN's is that they are able to approximate not only the physical reactions of a system but also the control decisions (extrinsic or intrinsic) that are practised in this system. This was observed in this project, in which the ANN was trained to internalise the greenhouse behavior, which was affected by the opening and closing of the thermal screen; thus, the ANN internalised implicitly the decisions concerning the thermal screen.

VIII.3.2 Development of a neural model

As seen in the present project, the development of a neural model requires the consideration of many aspects. First, there is a need for data with which to train the ANN. Second, an adequate configuration must be determined. Third, an appropriate number of learning cycles has to be applied during the training process. Concerning the first aspect, ANN's need to be trained with data that cover the whole spectrum of situations which they may face in a recalling mode. This is necessary since ANN's perform better in interpolation. The training data must, therefore, vary considerably so as to cover as many combinations as possible of the different variables involved.

With regards to the second aspect, concerning the configuration of ANN's, many factors need to be considered, such as the input variables, the number of processing elements, the learning paradigm, etc. The input variables of the ANN must be carefully chosen as a function of the output variables. The inputs must contain the information that will adequately inform an ANN about specific needs. For example, in this project the lagged values of certain variables were used as inputs, to allow the ANN to behave dynamically. If the neural model were to be used for subjective simulations with smaller subjective time steps, supplementary inputs could be necessary. For example, the state of the thermal screen may need to be fed to the ANN (e.g., opened, closed, half-closed). For a study with variable ventilation setpoints, the ventilation setpoints must be part of the inputs. While it is necessary to feed appropriate inputs to the network, redundancy should also be avoided to facilitate the pattern extraction process. Also, useless input variables must be avoided for the same reason. To help in the choice of the appropriate inputs, a method of sensitivity analysis with neural models must be developed.

Concerning the other factors to be considered in the determination of an appropriate ANN configuration, it was seen in this project that many of them will affect the learning capacity of the ANN. For example, the number of processing elements per hidden layer and the number of hidden layers affected the ANN learning capacity. The capacity to learn was also observed to be affected by a whole series of associated factors, such as the learning paradigm, the learning rules, the transfer function in the processing elements, the learning schedules and the learning parameters.

The number of learning cycles is another important aspect in the development of a neural model. On one hand, the number of learning cycles must be sufficient to allow the ANN to internalise the general patterns in the data sets. On the other hand, the ANN should not be overtrained, so as to avoid internalising the patterns that are specific to the training data set. The ANN should be trained so as to detect the patterns that are inherent in the "population" from which the training data set was extracted, and not the patterns that are specific to the sample. In ANN terminology, an ANN should *learn* during a training process; it should not *memorize* (Caudill, 1991). To help in the determination of the appropriate number of learning cycles, one approach consists of following the degree of learning during a training process. Many tools exist for this. In this project, an RMS value was used.

Many choices need to be made during the development of a neural model. The determination of the most appropriate combination can become a tedious trial-and-error process. It is, therefore, imperative that techniques and software be developed to facilitate this process.

VIII.3.3 Disadvantages

ANN's have a certain number of disadvantages. One of these is that a lot of data is required to train an ANN. This can be a limiting factor for the construction of neural models in a physical context. A second disadvantage of ANN's is that they are good only for recalling in interpolated conditions. A new training phase is necessary after the controlled entity has been modified. In this sense, ANN's can be compared to empirical models. Like empirical models, they give accurate results under the conditions in which they are developed, but it can be dangerous to use them under other conditions.

One approach to avoid retraining ANN's each time modifications are made to the system could be to create many specific ANN's and couple them together. The ANN's would form together a network where inputs for one ANN would be the outputs of other ANN's. For example, there could be separate ANN's to model different entities such as crop transpiration, the heating system and the greenhouse climate. In this case, a modification of the crop, for example, would require the retraining of the ANN that models crop transpiration. All the other ANN's that are part of the same network would not need to be retrained.

VIII.4 GREENHOUSE TEMPERATURE CONTROL

VIII.4.1 Potential benefits

The control strategy used by the simulation-based controller (SBC) to determine setpoints that minimize the energy required for heating was based on the assumption that crops possess a "temperature integration capacity". The strategy consisted of shifting heating towards periods when heat loss coefficients were smaller. At the beginning of each night, the controller determined setpoints for the coming 24-hour period. This was done with simulations based on weather forecasts. This control strategy resulted in a decrease in the energy consumption of the functional greenhouse system by a substantial amount. The results indicated the existence of potential savings for greenhouse enterprises. This justifies pursueing the investigation further. Especially, it should be determined if energy savings would be greater if 1) the set of setpoint trajectories available to the controller for its decisions was larger than the set available in this project, and 2) the decision period was extended to a period longer than 24 hours.

The advantages of using a simulated environment to develop the prototype controller were numerous, as were previously discussed (Section VIII.1.2). However, as with any simulation study, the effects of the control approach used in this research on physical greenhouses can be verified only through experiments in a physical environment. The effects of the control strategy on crops should be further investigated in both simulated and physical environments. Also, in this project, perfect weather forecasts were used to allow for the development, in isolation, of the algorithm used by the controller in its simulation-based decision-making processes. Future research must be done to determine how simulation-based controllers can deal with uncertainty such as that contained in weather forecast bulletins emitted by meteorological centers.

VIII.4.2 Extension to longer periods

The temperature integration period, i.e., the period within which the temperature integral must be achieved, can be as long as one week for mature tomato plants (see Chapter II). It could, therefore, be possible to extend the anticipation period, which was restrained to 24 hours in this project. The anticipation period could be extended to five days since weather forecast bulletins are currently available for this period. This would confer more flexibility for temperature control and would potentially increase the benefits.

Two aspects need to be considered in the elaboration of control strategies applied over many days: the length of the anticipation period and the length of the temperature integration period. In this project, both periods were the same: the anticipation period was 24 hours and the temperature integral had to be established within this period. This was the simplest case that could be considered. When the anticipation period is extended to 48 hours or more, the temperature integral can be either established or not established within the anticipation period. The simplest case is when the two periods are the same. However, by changing from 24 to 48 hours, the complexity of the processes increases rapidly, since the number of possible meteorological sequences and, consequently, the number of possible temperature regimes increases tremendously. For example, if for each slice of 24 hours, five meteorological scenarios are possible, this leads to 25 scenarios for 48 hours. If the anticipation period is extended up to five days, the controller will have 3125 (5^5) scenarios to investigate. Also, over a longer period, the weather forecasts become less reliable and more weather scenarios have to be investigated. This illustrates that there is a combinatorial expansion of the number of scenarios to be investigated by the controller when the anticipation period is extended over many days. Also, if the integration period is extended to seven days, the complexity of the decision-making process will increase. Extending the anticipation and integration periods can, therefore, potentially lead to additional benefits, but the management of such a control procedure would also be more complex. Research is, therefore, required to develop strategies able to deal with long anticipation periods. As was discussed in Section VIII.2.3, such strategies might be based on mechanisms that would narrow the domain of simulation.

VIII.4.3 Advantages of the simulation-based approach

The simulation-based approach seems to be advantageous in comparison with other approaches suggested in the literature for the implementation of control strategies based on the assumption that crop possess a temperature integration capacity. With regard to wind dependent temperature control, a procedure presented in the literature consisted of calculating the temperature setpoints as a function of the difference between the present and average wind speeds for a season. Equation VIII.1 was used by Bailey (1985) and Hurd and Graves (1984):

$$T_{spt} = T_b + (u_{avg} - u) * dT/du \qquad \text{(VIII.1)}$$

where:

T_{spt}	: Temperature setpoint	[°C]
T_b	: Base temperature	[°C]
u_{avg}	: Average wind speed	[m/s]
u	: Present wind speed	[m/s]
dT/du	: Constant of proportionality	[°C.s/m]

A control strategy using equation VIII.1 allows a certain average internal temperature to be obtained by decreasing the setpoint if wind speed is above the seasonal average speed, and increasing it otherwise. One of the problems with this approach is that the average wind speed can vary greatly from one year to the other, leading to sub-optimal average temperatures. Also, there is no guarantee that the temperature integral will be retrieved within a few days, and a mechanism is required to adjust the setpoints according to the departure between the desired and the realised temperature integral (Hurd and Graves, 1984). Also, the use of equation VIII.1 is not easily applicable to a greenhouse equipped with a thermal screen, for two related reasons. First, the effect of wind speed on heat losses is not the same when the screen is closed and when it is open. This requires that the temperature setpoints be calculated differently during the night and the day. Second, the length of the day varies during a year, which implies that the day and night temperature regimes must vary throughout the year. This is probably why wind-related algorithms have not been investigated for greenhouses equipped with thermal screens. A simulation-based approach can be advantageous in this instance to determine appropriate setpoints, because it allows the consideration of many factors at the same time. Apart from wind speed and thermal screen position it will also consider, implicitly, any other factor that affect heat loss factors (e.g., rain). The decision-making processes become complex when many factors are considered, and can hardly be based on simple control algorithms. A simulation-based controller is attractive for dealing with this situation. Such a controller would generate many setpoint sequences for the near future, and choose the most appropriate one by simulating the behavior of the greenhouse in all cases, and for the anticipated meteorological conditions.

Other advantages of the simulation-based control approach include the possibility of maintaining optimal temperatures. This is not the case when the temperatures are fixed and the temperature regime is simply inverted in greenhouses equipped with a thermal screen. Also, by allowing the consideration of meteorological conditions over many days, the simulation-based approach can help obtain a better distribution of the temperatures over many days, and establish the required temperature integral within the required period. Finally, a simulation-based approach can lead to more appropriate results than a controller using a procedure similar to that used by the reference controller in this project, because it does not allow such extreme temperatures.

VIII.4.4 The effect on crops

The approach used by the SBC to minimize energy requirements was based on the assumption that crops possess a temperature integration capacity. However, the growth rate as calculated by the crop model (SUCROS87) was not entirely independent of the nature of temperature regimes (even when the temperature integral was kept constant). This characteristic of the crop model had already become apparent in Appendix C when, during its evaluation, it was found that the net assimilation rate varied as a function of the leaf temperature for a given intensity of solar radiation and a given CO_2 concentration. This is in contradiction to the research results presented in the literature by many authors who have proposed that the specifics of a temperature path do not significantly affect yield, as long as the temperature integral attains a given value (de Koning, 1990; Hurd and Graves, 1984). From the results obtained in this research, it can be concluded that this capacity of crops to integrate the temperature was not reflected by the crop model used in the functional simulation system. Thus, research will be needed to obtain more detailed knowledge about the temperature integration capacity of crops and to appropriately modify the crop model.

In future developments, the consideration of crop growth will require that more complex temperature trajectories be tested. This will be especially true if high CO_2 concentrations are maintained, since the optimal leaf temperature might vary with CO_2 concentration. Finally, further research will be required to study the effects of different temperature regimes on crops. All patterns are certainly not good for all crops (Miller *et al.*, 1985). Suitability depends on many factors such as the cultivar, the development phase, etc. Research must be done to determine the limits within which the temperature is allowed to fluctuate and the maximum time that can elapse before reestablishing the temperature integral (Cockshull, 1988; 1985).

IX. SUMMARY AND CONCLUSIONS

Three tools were conceived to aid in the design of autonomous, enclosed agro-ecosystems: 1) a conceptual framework, 2) a (simulated) greenhouse system and 3) a simulation approach within OS/2. These tools were used as the basis for the development of a prototype simulation-based controller. The conceptual component furnished a frame for the development of simulation-based control, which was perceived as a specific instance of conscious control. The greenhouse system implemented under OS/2 was used to develop and test the prototype controller.

In the conceptual framework, a general model of an enclosed agro-ecosystem was first described. A theoretical structure was then suggested for its control system. For the general model of the agro-ecosystem, different categories of virtual and physical inputs and outputs were defined. The internal components of the system were subdivided into the production object, the production setting and the extrinsic controllers. The control system was described as a network of interacting control mechanisms, operating to achieve main production goals. A general model of a control mechanism was conceived and functions and attributes were defined. The functions comprised conflict treatment, decision-making and activation. The attributes included goal type (prevention, assurance, performance), activity level (regulation, operation, management) and implementation level (physical, instinctive, Pavlovian, cognitive).

Control mechanisms implemented at the cognitive level were recognized as performing "cognitive control". These mechanisms were also defined as being part of various responsibility groups (task management, information management, production control, control structure management and communications interface). A subset of cognitive control mechanisms were also recognized as performing "conscious control" when they were using, in their decision-making processes, models of the system of which they were part. Globally, conscious control was defined as the type of control performed by an entity accessing and using an internal representation of itself. It was suggested that conscious control could be implemented in a system by giving it the capacity to simulate. The expectation was that conscious control can potentially furnish an ecosystem with an adaptive capacity. One possibility was for the system to analyze the consequences of its control actions prior to their implementation.

To be able to further develop concepts related to cognitive control, and especially those related to conscious control, a (simulated) greenhouse system was conceived and implemented under OS/2. The objective was to construct a tool to be used in this project as well as in future investigations. The overall greenhouse system was composed of six modules: a simulation manager, a weather generator, a greenhouse model, a crop model, a Pavlovian controller and a cognitive controller. The modules were implemented as different processes in a common simulation structure under OS/2. Semaphores were used for interprocess synchronization and "shared memory segments" were used for interprocess exchange of data. The approach for simulation under OS/2 has proven to be quite satisfactory. For example, it allowed for the integration within the same structure, of programs that were written in different computer languages. The structure that was conceived was modular, which simplified the development of the various components. In the future, it will be relatively easy to adapt the structure for the simulation of more complex control situations.

Once implemented within OS/2, the greenhouse system was used to develop a prototype simulation-based controller. A series of experiments in which the role of the controller was to determine temperature setpoints that would minimize energy consumption for heating were executed. The control approach was based on the assumption that crops possess a temperature integration capacity. The approach consisted of shifting heating from periods with large heat loss coefficients to periods with small heat loss coefficients. At the beginning of each night, the controller produced and tested, by simulation, setpoints for the next 24-hour period. In this process, the controller had exact knowledge of the meteorological conditions that were to occur. The daily average temperature to be attained was the same as the monthly average temperature achieved with a reference controller. The model used by the controller was developed using artificial neural network technology.

The simulation-based control approach allowed the greenhouse system to adjust to predicted changes in meteorological conditions. High setpoints were maintained during the nights when outside temperature and solar radiation intensity were anticipated to be low during the following day. Inversely, low setpoints were maintained during the nights preceding days during which temperature and solar radiation intensity were anticipated to be high. This control resulted in a decrease of the heating load by an amount varying from 5% to 14% in comparison with a reference controller.

The results obtained indicated the presence of potential savings for the operation of greenhouses and further investigation is required to have a better estimation of these benefits. Specifically, experiments should be carried out with smaller temperature oscillations and with thermal screens of a lower quality. Also, research must be done with uncertain weather forecasts such as those available from meteorological centers. As well, the impact of control strategies on crops, such as that developed in this project, must be investigated in order to determine the limits of their applicability. Finally, investigations should be performed in physical greenhouses in order to obtain a better estimate of the benefits of these control strategies. Physical experiments will allow the analysis of how physical data will affect the learning capacity of artificial neural networks. This will also permit observations of how decisions made during simulation-based processes will be affected by the uncertainty attributed to anticipated disturbances and neural models.

The development of a prototype simulation-based controller and its implementation into a greenhouse system have shown the feasibility and usefulness of simulation-based control. They have allowed the determination of a set of constraints, requirements and factors to be considered for the design of simulation-based control systems. For example, requirements to extend the approach over a long anticipation period were defined. Also, a set of factors to be considered for the development of neural models became evident. The results obtained and the rapidity of calculations with artificial neural networks suggest that the simulation-based control approach should be extended into a multivariable control context. They also suggest that research could be carried out to apply this approach to the control of other types of enclosed agro-ecosystems, and of ecosystems in general.

X. CONTRIBUTIONS TO KNOWLEDGE

The following were original contributions to knowledge:

1. The development of a conceptual framework to aid in the design of enclosed agro-ecosystems; this included a general model of enclosed agro-ecosystems and a proposal for the functioning and the structure of the associated control systems;

2. The development of concepts related to cognitive and conscious control;

3. The creation of a functional greenhouse system which:
   - was composed of many interrelated controlled and controlling components,
   - allowed the simulation of the opening and closing of a thermal screen;

4. The development of a method for the simulation of complex systems within a multitasking operating system;

5. The development of a simulation-based control approach for enclosed agro-ecosystems;

6. The use of artificial neural networks for system modelling and simulation-based control;

7. The development of a temperature control approach for a simulation-based controller;

8. The development of elements of methodology for the implementation of conscious control, and participation in the creation of a base for the emergence of a theory on conscious control.

XI. RECOMMENDATIONS FOR FUTURE RESEARCH

XI.1 GENERAL RECOMMENDATIONS

Many aspects of the tools that were created in this project can be improved. Many suggestions were made in Chapter VIII for the improvement of the conceptual framework, the functional greenhouse system and the simulation approach under OS/2.

Another area of research will be the application of the conceptual framework to design control and management systems for other types of ecosystems. This will help to improve the framework and, at the same time, the framework should help the development of appropriate control systems. It will be possible to apply the framework to other types of enclosed ecosystems, to other agricultural production processes such as open field agriculture, to other food production processes such as aquacultural systems, and, finally, to ecosystems in general.

The creation of self-adjusting controllers, with the use of artificial neural networks, will constitute another interesting field of research. It will be possible to investigate self-adjustment at two levels: 1) for the improvement of models used by controllers, and 2) for the improvement of control strategies.

Three other subjects of investigation must be mentioned: 1) the addition of CO_2 control in the functional greenhouse system, 2) the development of an approach for the implementation of complex controllers under OS/2 and 3) the elaboration of methods to integrate uncertain weather forecasts in control decisions. Suggestions related to these three aspects are given in the following three sections.

XI.2 ADDITION OF CO_2 BALANCES TO THE FUNCTIONAL GREENHOUSE SYSTEM

The greenhouse system will be considerably improved by adding the possibility of maintaining high concentrations of CO_2. For this, it is required that CO_2 balances be incorporated into the greenhouse model. The greenhouse model contains two air volumes, and a CO_2 balance must be established simultaneously for each volume. The following is a suggestion for creating these two balances.

CO_2 balances can be easily added in the greenhouse model since it is coupled with a crop model. The CO_2 balance in a greenhouse is principally affected by crop, ventilation with external air and CO_2 generators. The crop consumes large quantities of CO_2 during photosynthesis. It also releases smaller quantities by respiration. In the greenhouse model, it might be assumed that only Air 1 is directly affected by crops and CO_2 generators, and that there is no production of CO_2 by organic matter in the soil. In this case, the CO_2 balance for Air 1 could be calculated by equation XI.1:

$$\frac{dC_{a,1}}{dt} = \frac{A_s}{V_1} * (C_c + C_{v,1} + C_x + C_i) \qquad \textbf{(XI.1)}$$

where:

$C_{a,1}$	: concentration of CO_2 in Air 1	[g/m^3]
t	: time	[s]
A_s	: soil surface	[m^2]
V_1	: volume of Air 1	[m^3]
C_c	: CO_2 added or removed by crop	[g/m^2.s]
$C_{v,1}$	: CO_2 added or removed by ventilation	[g/m^2.s]
C_x	: CO_2 added by CO_2 generator	[g/m^2.s]
C_i	: CO_2 transferred between Air 1 and Air 2	[g/m^2.s]

For Air 2, the CO_2 balance might be affected only by ventilation and by exchanges with Air 1. This CO_2 balance will be calculated by equation XI.2:

$$\frac{dC_{a,2}}{dt} = \frac{A_s}{V_2} * (C_{v,2} - C_i) \qquad \text{(XI.2)}$$

where:

$C_{a,2}$	: concentration of CO_2 in Air 2	[g/m^3]
t	: time	[s]
A_s	: soil surface	[m^2]
V_2	: volume of Air 2	[m^3]
$C_{v,2}$	: CO_2 added or removed by ventilation	[g/m^2.s]
C_i	: CO_2 transferred between Air 1 and Air 2	[g/m^2.s]

XI.3 SIMULATION OF COMPLEX COGNITIVE CONTROLLERS

As discussed in Chapter V, a cognitive controller of the type defined in the conceptual framework might be running many complex decision-making activities in parallel. This would produce the impression of a multiple stream sequence of decision-making that could continue, theoretically, infinitely. One way to maintain the effect of an ongoing decision-making sequence, while letting functional time flow during a simulation, is to: 1) suspend the cognitive activities at some point in each simulation cycle, 2) increment functional time and execute the other modules, and 3) return to the cognitive activities and pursue these for a certain (physical) time duration. The implementation of such a functionality requires a special procedure, and a suggestion is made here to develop such a procedure.

A simulation structure supporting the above mentioned characteristic can be implemented using some features inherent to multitasking operating systems such as OS/2. Two mechanisms are necessary: one to determine the length of the period during which COGNITI executes at each cycle, and a second one to suspend its execution. The duration of the execution period will depend on many factors, such as the simulated cognitive computing capacity, and the capacity of the machine on which the functional simulation is installed. Once the length of the execution period is determined, its actual duration can be controlled during a simulation by using one of the timing functions available under OS/2 via API's (e.g., DosTimerAsync, DosTimerStart).

The mechanism to suspend the execution of a process, without resuming it, can be created by using the following procedure under OS/2:
1- Set PRIORITY to ABSOLUTE in CONFIG.SYS.
2- Let MANAGER start the execution of the process COGNITI at a lower priority level than the other processes; COGNITI will then be activated only when no other processes possessing a higher priority will execute, since PRIORITY is set to ABSOLUTE.
3- At each time increment, MANAGER will execute one after the other the processes other than COGNITI. Then, it will execute the API DosSleep(), which will force MANAGER to sleep during a predetermined period, which will, in turn, force COGNITI to run during this period.

XI.4 WEATHER FORECAST AND ANTICIPATORY CONTROL

In this project, the controller had perfect knowledge of future meteorological conditions. In a future version, it will have to access information similar to that found in the public bulletins currently emitted by the meteorological centres. The variables of interest are outside temperature, solar radiation and wind speed. Since only limited information is emitted about the values of these variables, models are required to generate more complete paths, which are required for simulation (e.g., sequences of hourly values). A method is suggested here to design models to generate values for temperature and solar radiation intensity.

The weather forecast bulletins that are routinely available contain only minimum and maximum temperatures expected for each of the five following days. To generate hourly temperature values, a simple procedure can be adopted, in which it is assumed that the minimum and maximum occur respectively around the sunrise hour and in the middle of the afternoon (Gueymard, 1988). Then, the temperature can be assumed to vary sinusoidally during a day as suggested by Staley *et al.* (1986) and France and Thornley (1984). Using this approach, the controller can interpolate intermediate values using a sinusoidal variation between the extrema.

The weather forecast bulletins do not directly refer to the expected solar radiation levels, but contain qualitative information about the state of the sky (e.g., cloudy with sunny periods, overcast). In this case, global radiation sequences can be generated by first calculating the clear sky values for each hour using a standard solar radiation model such as that proposed by Won (1977) and then attenuating these according to the anticipated state of the sky. If global solar radiation needs to be decomposed into its direct and diffuse components, a standard model, such as those conceived by Collares-Pereira and Rabl (1979) or Liu and Jordan (1960), can be used.

REFERENCES

Aikman, D. P. and A. J. F. Picken. 1989. Wind-related temperature setting in glasshouses. J. of Hort. Sc. 64(6):649-654.

Amsen, M. G. 1986. Thermal screens in greenhouses: diurnal variations in heat consumption. J. Agric. Engng. Res. 33:79-89.

Antsaklis, P. J. 1990. Neural networks in control systems. IEEE Control Systems Magazine 10(3):3-5.

Antsaklis, P. J., K. M. Passino and S. J. Wang. 1991. An introduction to autonomous control systems. IEEE Control Systems Magazine 11(4):5-13.

Atmospheric Environment Service. 1983. Documentation for the digital archive of Canadian climatological data identified by element. Atmospheric Environment Service, Canadian Climate Center. Canada. 15 pp.

Bailey, B. J. 1988. Improved control strategies for greenhouse thermal screens. Acta Hort. 230:485-492.

Bailey, B. J. 1985. Wind dependent control of greenhouse temperature. Acta Hort. 174:381-386.

Bailey, B. J. 1978. Heat conservation in glasshouse with aluminized thermal screens. Acta Hort. 76:275-278.

Bailey, B. J. and Z. S. Chalabi. 1991. Optimisation of controlled plant environments. pp. 478-487. In: Proceedings of the 1991 Symposium on Automated Agriculture for the 21st Century, ASAE, Chicago, USA, December 1991, 540 pp.

Bailey, B. J. and I. Seginer. 1988. Optimum control of greenhouse heating. Presented at the ISHS Symposium Engineering and Economic Aspects of Energy Saving in Protected Cultivation, Cambridge, September 4-8th, 1988.

Bakker, J. C. 1985. A CO_2 control algorithm based on simulated photosynthesis and ventilation rate. Acta Hort. 174:387-392.

Bakker, J. C., L. van den Bos, A. J. Arendzen and L. Spaans. 1988. A distributed system for glasshouse climate control data acquisition and analysis. Comp. and Electro. in Agric. 3(1):1-7.

Bellamy, L. A. and B. A. Kimball. 1986. CO_2 enrichment duration and heating credit as determined by climate. In: Carbon dioxyde enrichment of greenhouse crops. Vol. II, Physiology, Yield, and Economics. Eds, H. Z. Enoch and B. A. Kimball. CRC Press, Inc., Boca Raton, Florida, USA. 230 pp.

Ben-Hanan, U., P.-O. Gutman and K. Peleg. 1991. Classification of apples with a neural network based classifier. In: Mathematical and control applications in agriculture and horticulture, eds Y. Hashimoto and W. Day, IFAC, Pergamon Press, Oxford, 165-169.

Berg, E. and W. Lentz. 1989. Dynamic optimal control of plant production. Acta Hort. 248:223-241.

Berger, K. M. and K. A. Loparo. 1989. A hierarchical framework for learning control. In: Artificial intelligence, simulation, and modelling. Eds: Widman L. E., K. A. Loparo and N. R. Nielsen. John Wiley & Sons, N.Y. 556 pp.

Boulard, T., B. Jeannequin and R. Martin-Clouaire. 1991. Analysis of knowledge involved in greenhouse climate management - Application to the determination of daily setpoints for a tomato crop. In: Mathematical and control applications in agriculture and horticulture, eds Y. Hashimoto and W. Day, IFAC, Pergamon Press, Oxford, 265-270.

Broner, I., C. S. Comstock and A. C. Parente. 1990. Intelligent optimization for a barley management expert system. ASAE Paper no. 90-7013.

Bullock, D., D. Whittaker, J. Brown and D. Cook. 1992. Neural networks for your toolbox. Agric. Engng., July 1992.

Caudill, M. 1991. Neural network training tips and techniques. AI Expert 6(1):56-61.

Chalabi, Z. S. and B. J. Bailey. 1991. Sensitivity of a non-steady state model of the greenhouse microclimate. Agr. and Forest Meteor. 56:11-127.

Challa, H. 1985. Report of the working party "Crop growth models". Acta Hort. 174:169-175.

Challa, H., J. C. Bakker, G. P. A. Bot, A. J. Udink ten Cate and J. van de Vooren. 1980. Economical optimization of energy consumption in an early cucumber. Acta Hort. 118:191-199.

Challa, H. and A. H. C. M. Schapendonk. 1986. Dynamic optimalization of CO_2 concentration in relation to climate control in greenhouses. In: Carbon dioxyde enrichment of greenhouse crops. Volume I, Status and CO_2 sources. Eds: H. Z. Enoch and B. A. Kimball. CRC Press Inc. Boca Raton, Florida, 181 pp.

Chatigny, R., N. Galanis, C. Gueymard and D. Labelle. 1981. Mise sous forme accessible des fichiers de données atmosphériques horaires en vue de leur utilisation efficace sur ordinateur. Report presented to the Direction des Energies Nouvelles du Ministère de l'Energie et des Ressources du Québec. Québec.

Cheston, T. S. and D. L. Winter. 1980. Human factors of outer space production. AAAS Selected Symposium, Westview Press, Inc. Boulder, Colorado, USA, 206 pp.

Chiapale, J.-P., C. H. M. van Bavel and E. J. Sadler. 1983. Comparison of calculated and measured performance of a fluid-roof and a standard greenhouse. Energy in Agric. 2:75-89.

Cockshull, K. E. 1988. The integration of plant physiology with physical changes in the greenhouse climate. Acta Hort. 229:113-123.

Cockshull, K. E. 1985. Greenhouse climate and crop response. Acta Hort. 174:285-292.

Coles, L.S. 1993. Engineering machine consciousness. AI Expert 8(5):34-41.

Collares-Pereira, N. and A. Rabl. 1979. The average distribution of solar radiation - correlations between diffuse and hemispherical and between daily and hourly insolation values. Solar Energy. 22(2).

Cooper, P. I. and R. J. Fuller. 1983. A transient model of the interaction between crop, environment and greenhouse structure for predicting crop yield and energy consumption. J. Agric. Engng. Res. 28:401-417.

C.P.V.Q. 1984. Légumes de serre, Culture. AGDEX 290/20. Ministère de l'Agriculture, des Pêcheries et de l'Alimentation du Québec. 156pp.

CREAQ. 1985. Sources d'énergie. Le Comité de Références Economiques en Agriculture du Québec, Ministère de l'Agriculture, de Pêcheries et de l'Alimentation du Québec. Agdex 760, Février 1985.

Critten, D. L. 1991. Optimization of CO_2 concentration in greenhouses: a modelling analysis for the lettuce crop. J. of Agr. Eng. Res. 48:261-271.

Curry, R. B. 1971. Dynamic simulation of plant growth - Part I. Development of a model. Trans. of the ASAE 14(5):946-949,959.

Dean, T. 1991. Decision-theoretic control of inference for time-critical applications. Int. J. of Intelligent Systems 6(4):417-441.

de Halleux, D., J. Nijskens and J. Deltour. 1991. Adjustment and validation of a greenhouse climate dynamic model. Bull. Rech. Agron. Gembloux. 26(4):429-453.

de Halleux, D. 1989. Modèle dynamique des échanges énergétiques des serres: Etude théorique et expérimentale. Thèse de doctorat, Université de Gembloux, Belgique. 284 pp.

de Koning, A. N. M. 1990. Long-term temperature integration of tomato. Growth and development under alternating temperature regimes. Scientia Hort. 45:117-127.

de Koning, A. N. M. 1988a. The effect of different day/night temperature regimes on growth, development and yield of glasshouse tomatoes. J. of Hort. Sc. 63(3):465-471.

de Koning, A. N. M. 1988b. An algorithm for controlling the average 24-hour temperature in glasshouses. J. of Hort. Sc. 63(3):473-477.

de Koning, A. N. M. 1988c. More efficient use of base load heating with a temperature integrating control programme. Effect on development, growth and production of tomato. Acta Hort. 229:233-237.

Deng, D. and J. O. Jenkins. 1989. Artificial intelligence validation of simulation models. Advances in AI and simulation. The Society for Computer Simulation International 20(4):6-11.

Doebelin, E. O. 1985. Control system principles and design. John Wiley & Sons. N.Y., USA. 577 pp.

Dogniaux, R. and M. Lemoine. 1984. Valeurs horaires moyennes mensuelles du bilan radiatif et de ses composantes à Uccle. Rep. A113, IRM, Brussels.

Dror, A. 1988. The Waite Group's OS/2 Programmer's Reference. Howard W. Sams & Company, Indianapolis, Indiana, USA. 621pp.

Dworetzky, J. P. 1982. Psychology. West Publishing, St-Paul, Minnesota, USA, 693 pp.

Elizondo, D. A., R. W. McClendon and G. Hoogenboom. 1992. Neural network models for predicting crop phenology. ASAE Paper no. 92-3596.

Enoch, H. Z. 1978. A theory for optimalization of primary production in protected cultivation: II. Primary plant production under different outdoor light regimes. Acta Hort. 76:45-57.

Fortson, R. 1992. Cultivating the high frontier. Agric. Engng. 73(6):20-23.

France, J. and J. H. M. Thornley. 1984. Mathematicals models in agriculture. Butterworths and Co. England. 335 pp.

Fynn, R. P., W. L. Bauerle and W. L. Roller. 1991. Expert system model for nutrient selection in a drip irrigation system. pp. 488-496. Proceedings of the 1991 Symposium on Automated Agriculture for the 21st Century, ASAE, Chicago, USA, December 1991, 540 pp.

Gary, C. 1988. A simple carbon balance model simulating the short-term responses of young vegetative tomato plants to light, CO_2 and temperature. Acta Hort. 229:245-250.

Gauthier, L. and R. Guay. 1990. An object-oriented design for a greenhouse climate control system. Trans. of the ASAE 33(3):999-1004.

Gauthier, L. and R. Kok. 1989. Integrated farm control software: I. Functional requirements and basic design criteria. AI Application in Natural Res. Mngt. 3(1):27-37

Gueymard, C. 1988. A study of the hourly distribution of the monthly-average ambient temperature and sky temperature in Canada. Presented at "Energy Solutions for Today", SESCI Conference, Ottawa, June 1988.

Haddock, J. 1987. An expert system framework based on a simulator generator. Simulation 48(2):45-53.

Hamel, A. 1992. "Aquaserre" de changer si ce n'est que pour s'améliorer. Option Serre 5(4):4-8.

Handelman, D. A., S. H. Lane and J. J. Gelfand. 1990. Integrating neural networks and knowledge-based systems for intelligent robotic control. IEEE Control Systems Magazine 10(3):77-87.

Harazano, Y., H. Kamiya and K. Yabuki. 1988. A control method based on an artificial intelligence technique and its application for controlling plant environment. Acta Hort. 230:209-214.

Hellickson, M. A. and J. N. Walker. 1983. Ventilation of agricultural structures. ASAE Monograph, American Society of Agricultural Engineers, St. Joseph, Michigan.

Hirafuji, M., Y. Ono and K. Kobayashi. 1988. A knowledgebase system and a neuro-computer system for agricultural information processing. Acta Hort. 230:253-259.

Hooper, A. W. 1988. Computer control of the environment in greenhouses. Comp. and Electro. in Agric. 3(1):11-27.

Hooper, A. W. and P. F. Davis. 1988. An algorithm for temperature compensation in a heated greenhouse. Comp. and Electro. in Agric. (2):251-262.

Hooper, A. W. and P. F. Davis. 1985. Control of greenhouse air temperature with an adaptive control algorithm. Acta Hort. 174:407-412.

Hoshi, T. and T. Kozai. 1984. Knowledge-based and hierarchically distributed online control system for greenhouse management. Acta Hort. 148:301-308.

Hurd, R. G. and C. J. Graves. 1984. The influence of different temperature patterns having the same integral on the earliness and yield of tomatoes. Acta Hort. 148:547-554.

Ihara, H., S. Mohri and S. Kawai. 1989. On dependable intelligent systems in space. Space Technol. 9(3):233-240.

Jackson, B. K., P. H. Jones, J. W. Jones and J. A. Paramore. 1989. Real-time greenhouse monitoring and control with an expert system. Comp. and Electro. in Agric. 3(4):273-285.

Johnstone, H. J. and N. Ben Abdallah. 1989. Prediction of potential latent heat recovery in greenhouses. CSAE/ASAE paper no. 89-4013.

Jones, P., Y. Hwang, J. W. Jones and R. A. Bucklin. 1988. Greenhouse environmental control using a tomato model. ASAE Paper no. 88-4058.

Jones, P., B. L. Roy and J. W. Jones. 1989. Coupling expert systems and models for the real-time control of plant environments. Acta Hort. 248:445-452.

Kano, A. and H. Shimaji. 1988. Greenhouse environmental control system with a crop model and an expert system. Acta Hort. 230:229-236.

Kim, S.H. 1990. Designing intelligence. Oxford University Press, Inc., New York, USA. 273 pp.

Kimball, B. A. 1986. A modular energy balance program including subroutines for greenhouses and other latent heat devices. Agricultural Research Service. United States Department of Agriculture. ARS 33. 356pp.

Kindelan, M. 1980. Dynamic modeling of greenhouse environment. Transactions of the ASAE 23(6):1232-1239.

Kok, R. and G. Desmarais. 1987. Structure d'un système de contrôle pour une serre intelligente. CSAE Paper no. 87-103.

Kok, R. and G. Desmarais. 1985. An integrated hierarchical control system for an intelligent greenhouse. ASAE Paper no. NAR85-403.

Kok, R. and R. Lacroix. 1993. An analytical framework for the design of autonomous, enclosed agro-ecosystems. Agric. Systems 43:235-260.

Kok, R., R. Lacroix and E. Taillefer. 1993. Imitation of a procedural greenhouse model with an artificial neural network. Submitted to Can. Agr. Engng.

Kok, R., R. Lacroix and E. Taillefer. 1991. Greenhouse climate modelling with an artificial neural network. CSAE Paper no. 91-212.

Kozai, T. 1985. Ideas of greenhouse climate control based on knowledge engineering techniques. Acta Hort. 174:365-373.

Kozai, T., M. J. le Mahieu, K. Kurata and T. Takakura. 1985. A greenhouse climate simulator for testing greenhouse computers. Part 1: Operation test of ventilation control. Acta Hort. 174:413-418.

Kreith, F. and J. F. Kreider. 1978. Principles of solar engineering. McGraw-Hill Book Company, New York. 778 pp.

Krug, H., H. P. Liebig. 1980. Diurnal thermoperiodism of the cucumber. Acta Hort. 118:83-94.

Kurata, K. 1988. Greenhouse control by machine learning. Acta Hort. 230:195-200.

Lacobucci, E. 1988. OS/2 Programmer's guide. Osborne McGraw-Hill, Berkeley, California, USA. 1100 pp.

Lacroix, R. and R. Kok. 1992. Les réseaux de neurones artificiels: une technologie à exploiter. Proceedings of: 16ième conférence de génie rural; Les nouvelles technologies informatiques en agro-alimentaire. Laval University, October 1992, Québec.

Lacroix, R. and R. Kok. 1991a. A cognitive controller with recourse to simulation. Presented at the Automated Agriculture for the 21st Century ASAE Symposium, Chicago, USA, December 1991.

Lacroix, R. and R. Kok. 1991b. Greenhouse simulation in a multitasking environment using OS/2. Presented at the 1991 Annual Meeting of NABEC, Ste-Anne de Bellevue, Canada, July 1991.

Lacroix, R. and J. C. Zanghi. 1990. Etude comparative de la structure des modèles de transfert d'énergie et de masse dans les serres. Can. Agric. Eng. 32(2):269-284.

Lal, H. and R. M. Peart. 1989. PROLOG for an integrated simulation and expert system. Advances in AI and simulation. The Society for Computer Simulation International 20(4):6-11.

Langhans, R. W., M. Wolfe and L. D. Albright. 1981. Use of average night temperatures for plant growth for potential energy savings. Acta Hort. 115:31-37.

Lessard, J.-P. 1989. Etude et développement de stratégies de contrôle informatisées de la température dans les serres. Mémoire de maîtrise, Université Laval, Québec.

Lessard, J.-P. and J.-M. Boudreau. 1992. Procédure de choix d'un système énergétique pour une entreprise serricole: Guide d'utilisation. Institut de Technologie Agro-Alimentaire, St-Hyacinthe, Québec.

Liebig, H. P. 1988. Temperature integration by Kohlrabi growth. Acta Hort. 230:371-380.

Liu, B. Y. H. and R. C. Jordan. 1960. The interrelationship and characteristic distribution of direct, diffuse and total solar radiation. Solar Energy. 4:1-19.

Lum, H. and E. Heer. 1988. Progress towards autonomous, intelligent systems. Space Technol. 8(4):337-349.

Maher, M. J. and T. O'Flaherty. 1973. An analysis of greenhouse climate. J. Agric. Eng. Res. 18:197-203.

Matthews, R. B., B. Marshall, R. A. Saffell and D. Harris. 1987. Computer control of carbon dioxyde concentration in experimental glasshouses and its use to estimate net canopy photosynthesis. Agr. and Forest Met. 40:279-292.

McClendon, R. W., W. D. Batchelor, D. B. Adams, J. W. Jones and J. E. Hook. 1989. Incorporating a crop growth simulation model in an expert system. Advances in AI and simulation. The Society for Computer Simulation International 20(4):176-179.

McCullough, C.L. 1992. An anticipatory fuzzy logic controller utilizing neural net prediction. Simulation 58(5):327-332.

Meijer, J. 1980. Reduction of heat losses from greenhouses by means of internal blinds with low thermal emissivity. J. of Agric. Engng. Res. 25:381-390.

Mendell, W. W. 1985. Lunar bases and space activities for the 21st century. Lunar and Planetary Institute, NASA, USA, 865 pp.

MER. 1983. Inventaire et validation des modèles de calcul du rayonnement solaire au Québec en vue des applications énergétiques. Collection Etudes et Recherches, Ministère de l'Energie et des Ressources du Québec, Gouvernement du Québec. 207 pp.

Micro Data Base Systems. 1991. GURU - Integrated components manual. Micro Data Base Systems, Inc., Lafayette, Indiana, USA.

Microsoft Corporation. 1991. Microsoft FORTRAN; Reference. Microsoft Corporation, USA, 534 pp.

Microsoft Corporation. 1990. Microsoft C; Advanced Programming Techniques. Microsoft Corporation, USA, 477 pp.

Microsoft Corporation. 1989. Microsoft BASIC; Programmer's guide. Microsoft Corporation, USA, 782 pp.

Miller, W. B., R. W. Langhans and L. D. Albright. 1985. Plant growth under averaged day/night temperatures. Acta Hort. 174:313-320.

Monteil, C. 1985. Contribution informatique à l'analyse énergétique des serres agricoles. Thèse de doctorat, Institut National Polytechnique de Toulouse, France, 168 pp.

Montheith, J. L., B. Marshall, R. A. Saffell, D. Clarke, J. N. Gallagher, P. J. Gregory, C. K. Ong, G. R. Squire and A. Terry. 1983. Environmental control of a glasshouse suite for crop physiology. J. of Exp. Botany. 34(140):309-321.

Morimoto, T. and Y. Hashimoto. 1991. Application of fuzzy logic and neural network to the process control of solution pH in deep hydroponic culture. In: Mathematical and control applications in agriculture and horticulture, eds Y. Hashimoto and W. Day, IFAC, Pergamon Press, Oxford, 147-152.

Moser, J. G. 1986. Integration of artificial intelligence and simulation in a comprehensive decision-support system. Simulation 47(6):223-229.

Murray, K. J. and S. V. Sheppard. 1988. Knowledge-based simulation model specification. Simulation 50(3):112-119.

Nederhoff, E. M. 1988. Dynamic optimization of the CO_2 concentration in greenhouses: an experiment with cucumber (Cucumis Sativus L.). Acta Hort. 229:341-348.

NeuralWare. 1991. Neural computing - NeuralWorks Professional II/PLUS and NeuralWorks Explorer. NeuralWare, Inc. Pittsburgh, USA. 360 pp.

O'Flaherty, T. 1989. Microcomputers in energy-saving greenhouses. Acta Hort. 245:416-423.

Okada, M. and T. Takakura. 1973. Guide and data for greenhouse air conditioning. J. of Agr. Met. 28(4):11-18.

O'keefe, R. 1986. Simulation and expert systems - A taxonomy and some examples. Simulation 46(1):10-16.

Padgett, M. L. and T. A. Roppel. 1992. Neural networks and simulation: modeling for applications. Simulation 58(5):295-305.

Patterson, G. N. 1989. Priorities in geolunar space. Institute for Aerospace Studies University of Toronto, Downsview, Ontario. 121pp.

Rabbinge, R., S. A. Ward and H. H. van Laar. 1989. Simulation and system management in crop protection. Simulation Monographs; 32. Pudoc, Wageningen, The Netherlands. 420 pp.

Reddy, R. 1987. Epistemology of knowledge based simulation. Simulation 48(4):162-166.

Rotz, C. A. 1977. Computer simulation to predict energy use and system costs for greenhouse environmental control. Ph.D. thesis, Pennsylvania State University. 132pp.

Roy, B. L. and P. Jones. 1988. A dynamic multi-tasking environmental control system for plant growth chambers. ASAE Paper no. 88-4019.

Saffell, R. A. and B. Marshall. 1983. Computer control of air temperature in a glasshouse. J. Agr. Engng. Res. 28:469-477.

Salisbury, F.B. 1991. Lunar farming: Achieving maximum yield for the exploration of space. HortScience 26(7):827-833.

Sase, S. and M. Nara. 1985. A control algorithm for natural ventilation based on wind tunnel testing. Acta Hort. 174:75-80.

Schapendonk, A. H. C. M., H. Challa, P. W. Broekharst and A. J. Udink ten Cate. 1984. Dynamic climate control: an optimization study for earliness of cucumber production. Scientia Hort. 23:137-150.

Seginer, I. 1984. On the night transpiration of greenhouse roses under glass or plastic cover. Agr. Met. 30:257-268.

Seginer, I. 1980. Optimizing greenhouse operation for best aerial environment. Acta Hort. 106:169-178.

Seginer, I., A. Angel, S. Gal and D. Kantz. 1986. Optimal CO_2 enrichment strategy for greenhouses; a simulation study. J. Agric. Engng. Res. 34:285-304.

Seginer, I. and N. Levav. 1971. Models as tools in greenhouse climate design. Agricultural Engineering Faculty. Technion-Israel Institute of Technology. 80pp.

Seginer, I., G. Shina, L. D. Albright and L. S. Marsh. 1991. Optimal temperature setpoints for greenhouse lettuce. J. of Agr. Eng Res. 49:209-226.

Shannon, R. E., R. Mayer and H. H. Adelsberger. 1985. Expert systems and simulation. Simulation 44(6):275-284.

Short, T.H. and W.L. Bauerle. 1977. A double-plastic heat conservation system for glass greenhouses. ASAE Paper no. 77-4528.

Shoureshi, R. 1991. Learning and decision making for intelligent control systems. IEEE Control Systems 11(1):34-37.

Silveston, P. L., W. D. Costigane, H. Tiessen and R. R. Hudgins. 1980. Energy conservation through control of greenhouse humidity. I. Condensation heat losses. Can. Agr. Eng. 22(2):125-132.

Spitters, C. J. T. 1986. Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis. Part II. Calculation of canopy photosynthesis. Agr. and Forest Met. 38:231-242.

Spitters, C. J. T., H. v. Keulen and D. W. G. v. Kraalingen. 1989. A simple and universal crop growth simulator: SUCROS87. In: Simulation and system management in crop protection. Simulation Monographs; 32. Eds: R. Rabbinge, S. A. Ward and H. H. van Laar. Pudoc, Wageningen, The Netherlands. 420 pp.

Spitters, C. J. T., H. A. J. M. Toussaint and J. Goudriaan. 1986. Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis. Part I. Components of incoming radiation. Agr. and Forest Met. 38:217-229.

Staley, L. M., G. J. Monk and J. M. Molnar. 1986. The influence of thermal curtains on energy utilization in glass greenhouses. J. Agr. Engng. Res. 33:127-139.

Stengel, R. F. 1991. Intelligent failure-tolerant control. IEEE Control Systems 11(4):14-23.

Takakura, T. 1991. Greenhouse production in engineering aspects. In: Mathematical and control applications in agriculture and horticulture, eds Y. Hashimoto and W. Day, IFAC, Pergamon Press, Oxford. pp 19-21.

Takakura, T., K. A. Jordan and L. L. Boyd. 1971. Dynamic simulation of plant growth and environment in the greenhouse. Trans. of the ASAE 14(5):964-971.

Takakura, T., H. Shono and T. Hojo. 1984. Crop management by intelligent computer systems. Acta Hort. 148:317-318.

Tang, Z., C. de Almeida and P. A. Fishwick. 1991. Time series forecasting using neural networks vs. Box-Jenkins methodology. Simulation 57(5):303-310.

Tantau, H.-J. 1989. On-line climate control. Acta Hort. 248:217-222.

Tantau, H.-J. 1985. Analysis and synthesis of climate control algorithms. Acta Hort. 174:375-380.

Thangavadivelu, S. and T. S. Colvin. 1989. Decision support for crop production operations scheduling. Advances in AI and simulation. The Society for Computer Simulation International 20(4):6-11.

Townsend, J. S., M. G. Britton and M. Rafiq-ur-Rehman. 1978. Infiltration and exfiltration rates in plastic-covered greenhouses. CSAE Paper no. 78-409.

Treu, S. 1988. Designing a "Cognizant Interface" between the user and the simulation software. Simulation 51(6):227-234.

Udink ten Cate, A. J. 1983. Modeling and (adaptive) control of greenhouse climates. Thesis, Agricultural University of Wageningen, The Netherlands, 159pp.

Udink ten Cate, A. J., G. P. A. Bot and J. J. van Dixhoorn. 1978. Computer control of greenhouse climates. Acta Hort. 87:265-272.

Valentin, J. and J. van Zealand. 1980. Adaptive split-range control of a glasshouse heating system. Acta Hort. 106:109-115.

Van Bavel, C. H. M. 1975. A behavioral equation for leaf carbon dioxyde assimilation and a test of its validity. Photosynthetica 9(2):165-175.

Van Bavel, C. H. M. and E. J. Sadler. 1979. SG79 - a computer simulation program for analyzing energy transformations in a solar greenhouse. Mimeographed report. Texas A&M University. 75 pp.

Van Heemst, H.D.J. 1986. Potential crop production. in Modelling of agricultural production: weather, soils and crops. H. van Keulen and J. Wolf (Eds.). Simulation Monographs. Centre for Agricultural Publishing and Documentation. Wageningen. the Netherlands. 479 pp.

Van Keulen, H. and J. Wolf. 1986. Modelling of agricultural production: weather, soils and crops. H. van Keulen and J. Wolf (Eds.). Simulation Monographs. Centre for Agricultural Publishing and Documentation. Wageningen. the Netherlands. 479 pp.

Whittaker, A. D., B. S. Park, J. D. McCauley and Y. Huang. 1991. Ultrasonic signal classification for beef quality grading through neural networks. pp 116-125. In: Proceedings of the 1991 Symposium, Automated Agriculture for the 21st Century, Chicago, December 1991. 540 pp.

Whittle, R. M. and W. J. C. Lawrence. 1960. The climatology of glasshouse: II. Ventilation. J. of Agr. Eng. Res 5(1):36-41.

Wildberger, A. M. 1990. Neural networks as a modeling tool. In: AI and simulation 22(3), ed. SCSI, San Diego, CA, 65-74.

Willits, D. H., M. M. Peet and P. Chandra. 1981. The effect of transpiration on the solar collection potential of a greenhouse. ASAE Paper no. 81-4041.

Won, T. 1977. The simulation of hourly global radiation from hourly reported meteorological parameters - Canadian Prairie area. Paper presented at the 3rd conference of the Canadian Solar Energy Society Inc., Edmonton, Alberta, 23 pp.

Wright, M. L., M. W. Green, G. Fiegl and P. F. Cross. 1986. An expert system for real-time control. IEEE Software 3(2):16-24.

Yang, X., T. H. Short, R. D. Fox and W. L. Bauerle. 1987. An experimental study of a greenhouse cucumber crop microclimate. ASAE Paper no. 87-4547.

Zeigler, B. P. 1990. Object-oriented simulation with hierarchical, modular models: Intelligent agents and endomorphic systems. Academic Press, Inc., San Diego, USA, 395pp.

Zhang, Q., J. F. Reid, J. B. Litchfield, J. Ren and S.-W. Chang. 1992. A prototype neural network supervised control system for *Bacillus Thuringiensis* fermentations. ASAE Paper no. 92-3592.

Zhuang, X. and B. A. Engel. 1990. Neural network for applications in agriculture. ASAE Paper no. 90-7024.

APPENDIX A. METEOROLOGICAL DATA

Atmospheric Environment Service (AES) data is recorded on an hourly basis. For all variables except solar radiation, the values contained in data files supplied by AES represent punctual measurements taken at the beginning of each hour from 00:00 to 23:00 h. This time is "local standard time" (Atmospheric Environment Service, 1983; Chatigny *et al.*, 1981). For solar radiation, the value represents the irradiation integrated over one hour. In this case, the time indicated is "true solar time". Local standard time is the conventional time used everywhere within any given time zone, and is equal to the "mean solar time" at the meridian of reference for the given time zone. The mean solar time is an approximation of the true solar time, adjusted to give days of equal length (i.e., 24 hours). There is, therefore, a difference between the time indicated for solar radiation and the one indicated for all other variables. The magnitude of this difference oscillates during a year as shown below.

The difference between the mean solar time (MST) and the true solar time (TST) is given by the "equation of time" (ET), as illustrated in equation A.1.

$$TST = MST + ET \tag{A.1}$$

where, for convenience, the units are in minutes:

TST	: True solar time	[Minutes]
MST	: Mean solar time	[Minutes]
ET	: Equation of time	[Minutes]

The equation of time can be approximated, e.g., by a Fourier series. Values of ET have been plotted in many textbooks and oscillations between approximately -16 minutes and +14 minutes can be observed (Kreith and Kreider, 1978). Thus, the relationship between MST and TST can be approximated by equation A.2.

$$TST = MST \pm 16 \text{ minutes} \tag{A.2}$$

The mean solar time at a specific longitude is given by equation A.3.

$$MST = ST + 4 * (\text{Reference meridian} - \text{Longitude}) \qquad \text{(A.3)}$$

where ST is local standard time. In equation A.3, the meridian of reference and the longitude are given in degrees. The meridian of reference and the longitude for Montréal are respectively 75° and 73.75°. When this information is used in equations A.2 and A.3, the relationship between TST and ST at the meteorological station at Dorval Airport (Montréal) is (equations A.4 and A.5):

$$TST = ST + 4 * (75 - 73.75) \pm 16 \text{ minutes} \qquad \text{(A.4)}$$

$$TST = ST + 5 \pm 16 \text{ minutes} \qquad \text{(A.5)}$$

The magnitude of the difference, therefore, varies between approximately -9 and + 21 minutes during the year for Montréal.

APPENDIX B. THE GREENHOUSE MODEL

B.1 ENERGY FLUXES IN GGDM2

CONVECTION

FL(1)	: Convection Soil surface - Air 1
FL(2)	: Convection Air 1 - Thermal screen
FL(3)	: Convection Thermal screen - Air 2
FL(4)	: Convection Air 2 - Cover
FL(5)	: Convection Cover - Outside air
FL(6)	: Convection Crop - Air 1
FL(7)	: Convection Air 1 - Air 2
FL(8)	: Convection Air 2 - Outside air
FL(9)	: Convection Air 1 - Outside air

EVAPORATION

FL(11)	: Evaporation Soil surface - Air 1
FL(12)	: Evaporation Air 1 - Thermal screen
FL(13)	: Evaporation Thermal screen - Air 2
FL(14)	: Evaporation Air 2 - Cover
FL(15)	: Evaporation Cover - Outside air
FL(16)	: Evaporation Crop - Air 1
FL(17)	: Evaporation Air 1 - Air 2
FL(18)	: Evaporation Air 2 - Outside air
FL(19)	: Evaporation Air 1 - Outside air

INFRARED RADIATION

FL(21) :	Radiation Soil susrface - Crop
FL(22) :	Radiation Soil susrface - Thermal screen
FL(23) :	Radiation Soil surface - Cover
FL(24) :	Radiation Soil surface - Sky
FL(25) :	Radiation Crop - Thermal screen
FL(26) :	Radiation Crop - Cover
FL(27) :	Radiation Crop - Sky
FL(28) :	Radiation Thermal screen - Cover
FL(29) :	Radiation Thermal screen - Sky
FL(30) :	Radiation Cover - Sky

SOLAR RADIATION

FL(31) :	Solar radiation on cover
FL(32) :	Solar radiation on thermal screen
FL(33) :	Solar radiation on soil surface
FL(34) :	Solar radiation on crop

SOIL CONDUCTION

FL(36) :	Conduction Soil surface - Soil 1
FL(37) :	Conduction Soil 1 - Soil 2
FL(38) :	Conduction Soil 2 - Soil 3
FL(39) :	Conduction Soil 3 - Deep soil

HEATING

FL(40) : Heating

B.2 MODEL PARAMETER VALUES

B.2.1 Cover

For this research, the transparent cover material was assumed to be ordinary horticultural glass with a thickness of 4 mm. All parameter values were as described by de Halleux (1989). The absorptivity, reflectivity and transmissivity to the direct component of solar radiation were a function of the incident angle of the radiation. During a simulation, the values for transmissivity and reflectivity were found from those in Table B.1 by linear interpolation, and the absorptivity was calculated from the two other values. The other parameter values for the cover are presented in Table B.2.

B.2.2 Thermal screen

The parameter values of the thermal screen are listed in Table B.3. The screen was made of aluminized polyester completely opaque to far infrared radiation. All parameter values except the thermal transmissivity were as reported by de Halleux (1989).

Table B.1 Cover transmittivity and reflectivity to direct solar radiation (from de Halleux, 1989).

Incidence angle	0°	15°	30°	45°	60°	75°	90°
Transmissivity [%]	89	88	86	84	74	37	0
Reflectivity [%]	10	10	11	14	22	61	100

Table B.2 Cover properties (from de Halleux, 1989).

Thermal emissivity [%]	90
Transmissivity to thermal radiation [%]	0
Absorptivity to diffuse solar radiation [%]	15
Transmissivity to diffuse solar radiation [%]	75
Surface thermal capacity [J/°K·m^2]	8000.0

Table B.3 Thermal screen properties.

Thermal emissivity [%]	35
Transmissivity to thermal radiation [%]	0
Surface thermal capacity [J/°K·m^2]	630.0

B.2.3 Crop

The crop parameters used by the greenhouse model are shown in Table B.4. The leaf area index and the vegetal mass are also parameters of the crop model and, ideally, their values should vary during a simulation over a long period (e.g., over a growing season). However, the crop parameters were assumed to remain constant during simulations, which were all done over a maximum period of one month. The total vegetal mass used by the greenhouse model corresponds to the weight that can be calculated from the percentage of water in the plant and the dry weight values used in the crop model (Section C.1).

Table B.4 Crop parameters for the greenhouse model.

Thermal emissivity [%]	70
Reflectivity to global solar radiation [%]	15
Absorptivity to global solar radiation [%]	70
Transmissivity to global solar radiation [%]	15
Specific heat [J/kg·°C]	4180
Leaf area index [m^2/m^2]	3
Total vegetal mass [kg/m^2]	6
Proportion of the soil covered by crops [%]	80

B.2.4 Soil

The thickness, thermal conductivity, density and specific heat of the soil layers are shown in Table B.5. All these values, except the layer thicknesses, are as reported by de Halleux (1989), and are typical for a silty-sandy soil.

Table B.5 Soil parameters.

Soil layer number	1	2	3	4
Thickness [m]	0.05	0.10	0.50	2.00
Density [kg/m^3]	1500	1700	1900	2100
Thermal conductivity [W/m·°C]	0.6	1.1	1.4	1.6
Specific heat [J/kg·°C]	900	1100	1300	1500

The soil surface was assumed to be white, as if it were covered with white polyethylene, as is often the case in greenhouses with hydroponic culture. Thus, the solar reflectivity of the soil surface and its emissivity were set respectively at 70% and 30%. The solar absorptivity was assumed to be the same as the emissivity.

To calculate heat transfer by conduction in the soil, the temperature in the deep soil (i.e., at 2 m) was required. Some models exist to generate such a temperature for open field soil, but these models are not valid when the soil is covered by a building such as a greenhouse, which affects the soil temperature (Kindelan, 1980). In the absence of criteria to establish the deep soil temperature, a constant value of 15°C was assumed.

B.2.5 Convective parameters

Several parameters related to convective transfer needed to be determined. The speed of the air in both Air 1 and Air 2 was given a value of 0.6 m/s, which is the average of the range suggested to avoid crop stresses, i.e., from 0.2 to 1 m/s (CPVQ, 1984; Hellickson and Walker, 1983). A second parameter is the infiltration rate of air into the greenhouse, for which a value of 1.0 air renewal per hour (de Halleux, 1989) was used initially. After a series of experiments, a relationship was added to the greenhouse model to calculate the infiltration rate as a function of the wind speed (see equation IV.5). This is discussed in more detail in Section B.3.4.

Two other important convective parameters are related to the air flow rate between Air 1 and Air 2. As discussed in Section IV.2.4.1, this air flow rate is different when the thermal screen is closed from when it is opened. The scarcity of information in the literature would suggest that not much research has been done about the air transfer rates when thermal screens are shut. The air flow rates depend on the installation, on the type of thermal screen and on the air tightness of all the joints. The rates probably vary a lot from one greenhouse to the other. Meijer (1980) reported values varying between 10^{-3} and $5*10^{-3}$ m^3/m^2s and, for this research, a value of $2.4*10^{-3}$ m^3/m^2s was arbitrarily chosen. For the air flow rate when the thermal screen is opened, it was simply considered that it was 20 times larger than when the thermal screen was closed. During simulations, it was observed that the performance of the greenhouse model was quite sensitive to these values and, therefore, these values should be determined carefully when simulating the behavior of a specific greenhouse. However, the values chosen here were appropriate for the present study and simply represented the situation found in some greenhouse.

B.3 STEADY-STATE ANALYSIS OF THE GREENHOUSE MODEL

B.3.1 Introduction

The steady-state analysis of the greenhouse model consisted mostly of studying the effects of the outside temperature, wind speed, solar radiation and cloudiness on the heating and ventilation requirements, and on the temperatures of the greenhouse components. In each simulation, all input values were kept constant. Also, all greenhouse parameters were fixed at the values determined in Section B.2. The analysis was done in three series of experiments. In each series, simulations were executed for various values of one meteorological variable.

In the first series, the outside temperature ranged between -30°C and 30°C, and the effect of this variation was studied for two states of the thermal screen (open and closed) and for two levels of cloudiness (0 tenths and 10 tenths). In this, the wind speed and solar radiation intensity were kept constant. In the second series, the impact of the wind speed on the greenhouse was evaluated. Simulations were done for wind speeds ranging from 0.0 m/s to 13.3 m/s. The effect was studied again for the two states of the thermal screen and for two cloudiness levels, with a constant outside temperature and solar radiation intensity. Two different methods of calculating the infiltration rate were also tested. In the third series, the effect of the solar radiation intensity on the system was studied. It was especially analyzed how the solar intensity affects the temperature of the components, the heating load and the ventilation load. In this, the wind speed and outside temperature were kept constant.

In all simulations, the outside relative humidity and the atmospheric pressure were maintained constant at 80% and 101.3 kPa respectively. The dead band for the ON-OFF heating system was 1°C and the power of the furnace was 300 W/m^2. The heating setpoint was kept constant at 20°C.

B.3.2 Attainment of steady-state conditions

Since the greenhouse model is dynamic (i.e., composed of differential equations), all simulations were done over a long period to obtain steady-state conditions. A period of five days was sufficient to obtain quasi steady-state conditions. This is illustrated in Figures B.1a and B.1b. For these, the outside temperature was constant at -20°C, and the wind speed and the solar radiation intensity were zero. It can be noted that the temperature of all components became relatively constant after a few hours for both states of the thermal screen; only the soil surface temperature continued to vary slowly. For both states of the thermal screen, the heating load was also relatively stable after few hours, as shown in Figure B.1c. In all analyses, only the average results of the last 24 hours were considered.

B.3.3 Variation of the outside temperature

In the first set of experiments, the effect of the outside temperature on the greenhouse behavior was studied. The following conditions were maintained throughout all experiments: constant heating setpoint (20°C), wind speed of zero, no solar radiation. The outside temperature was also kept constant in each simulation, and simulations were executed for outside temperature values ranging from -30°C to +30°C, with increments of 10°C. The simulations were done for both states of the thermal screen (closed and open) and for two cloudiness levels (0 and 10).

The variation of the temperature of the greenhouse components as the outside temperature varied from +30 to -30°C is shown in Figure B.2, when the thermal screen was either closed or open and the cloudiness was either 0 tenths or 10 tenths. The temperature of the cover and the Air 2 (referred below as "gas") were lower when the thermal screen was closed. The low temperature of the cover was due to the lack of heat gain by infrared radiation from the soil and crop, and to the low heat gains by convection from the gas.

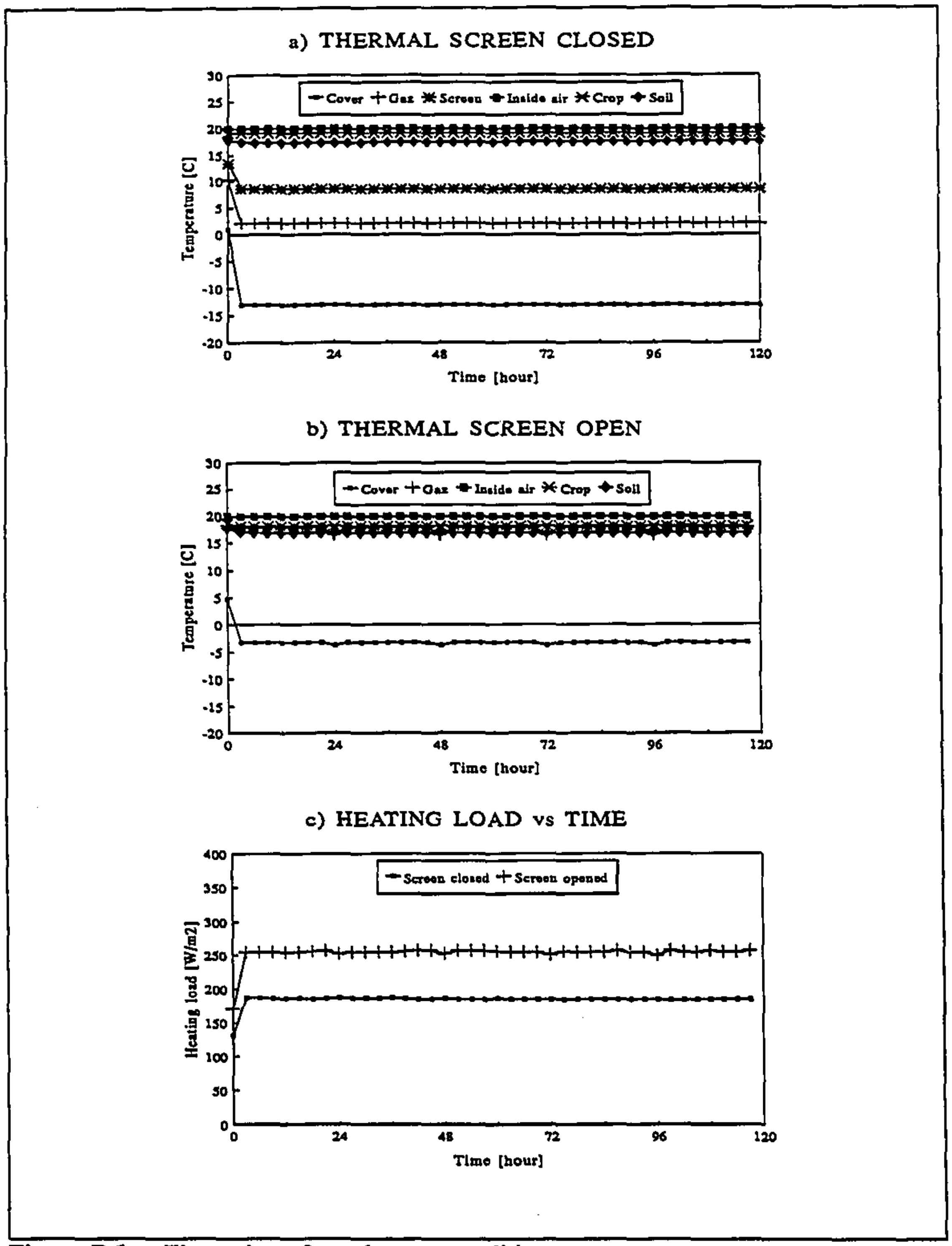

Figure B.1 Illustration of steady-state conditions.

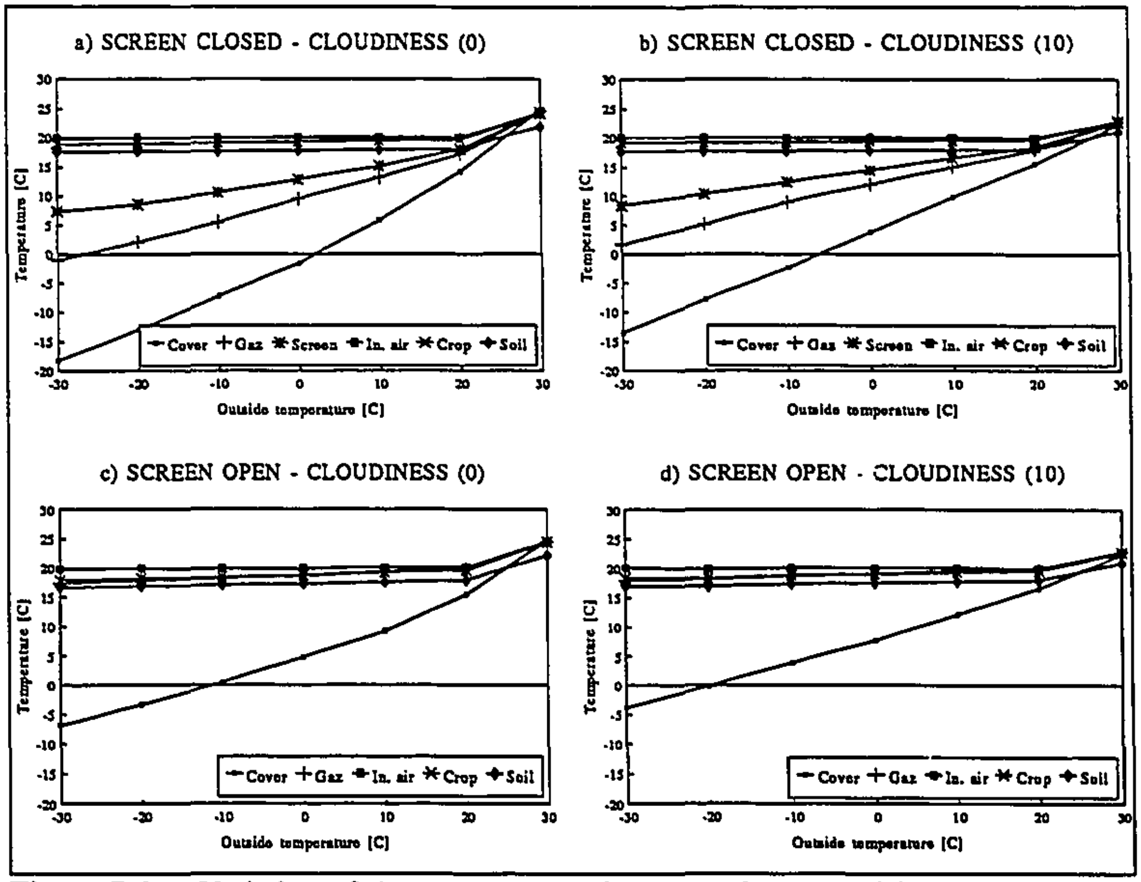

Figure B.2 Variation of the temperature of the greenhouse model components with outside temperature.

The temperature of the gas was maintained low due to the relatively slow rate of air transfer between the gas and Air 1 (referred below as "inside air" or simply "air").

When the outside temperature was higher than a certain value, the temperature of the cover was lower than the outside temperature (e.g., for outside temperature equal to or higher than 0°C when the thermal screen was closed and the cloudiness was 0 - Figure B.2a). This was mainly caused by the large amount of energy transferred by radiation between the cover and the sky vault, whose temperature was very low compared to the outside air temperature. This is discussed in more detail in Section B.3.4.

The temperature of the gas was lower than that of the inside air even when the screen was open (Figure B.2c and B.2d); the difference between them was 3°C when the lowest outside temperature was used. This is contrary to what is usually found in physical greenhouses where the air temperature is generally stratified in the opposite direction. This is due to the model, in which the air layers are considered as homogeneous blocks; also the direction of heat transfer is from the bottom of the greenhouse to the top.

The inside temperature closely followed the setpoint (20°C) when the outside temperature was equal to or less than 20°C (Figure B.2). Generally, the crop temperature decreased with the outside temperature, mainly because the radiative temperature of the sky and of the cover (or of the thermal screen when it was closed) decreased equally.

The steady-state heating load as a function of the outside temperature is shown in Figure B.3. The variations are inversely proportional and the relationships are not linear. It can be observed that there was heating even if the outside temperature was the same as the setpoint (20°C), mainly because of large radiative losses. The overall heat transfer coefficient can be calculated by dividing the heating load by the difference in temperature between the greenhouse air and outside air. The values vary from 5.5 $W/m^2 \cdot °C$ to 9.8 $W/m^2 \cdot °C$ without a thermal screen, with an average of 7.65 $W/m^2 \cdot °C$. The value reported by CPVQ (1984) for glass is given in terms of surface cover unit area and is 6.46 $W/m_c^2/°C$. This value can be transformed in terms of soil surface unit area by multiplying it by the cover to soil surface ratio. This ratio was 1.2 for the greenhouse configuration used in this project, which would yield a value of 7.75 $W/m^2 \cdot °C$. This is close to the average of the values that were obtained in the simulations (i.e., 7.65 $W/m^2 \cdot °C$).

The decrease of the heating load when the thermal screen was closed varied between 20 and 40% (Figure B.3), depending on the outside temperature. The benefits were slightly larger when the cloudiness was 0. These values were lower than the 60% suggested by Bailey (1988) for this type of thermal screen. This can be due to a number of factors such

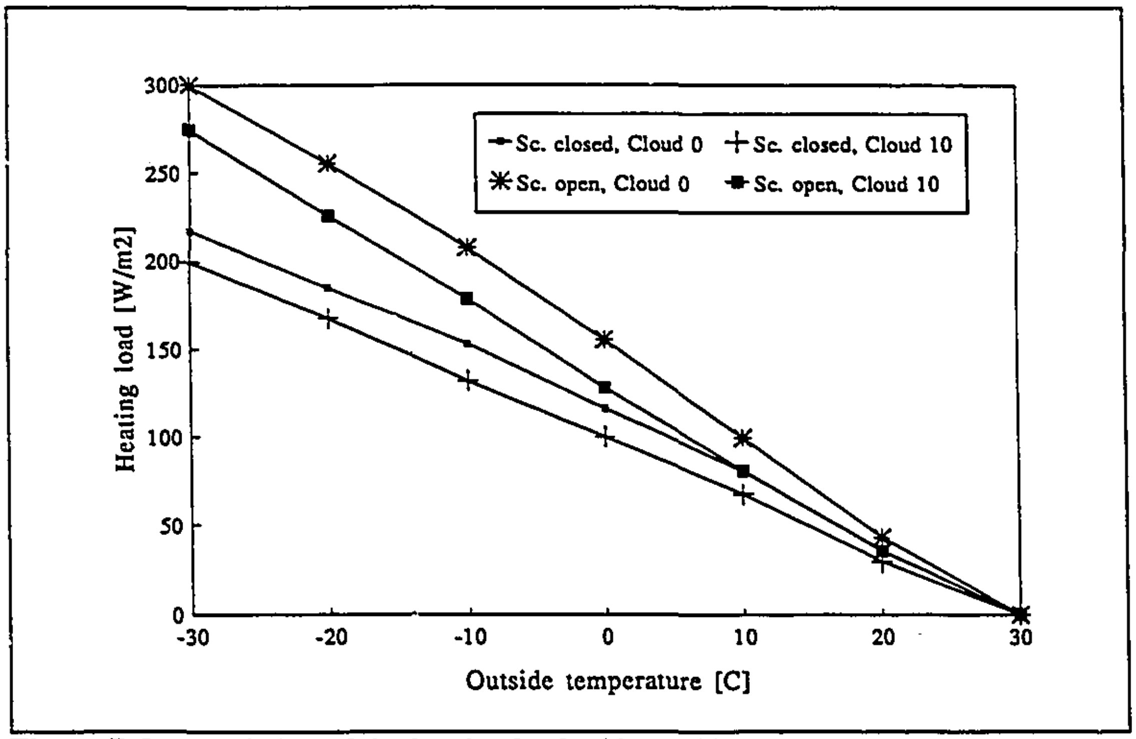

Figure B.3 Variation of the heating load with outside temperature.

as the absence of wind and a constant infiltration rate. This will be discussed in Section B.3.4. Figure B.3 also illustrates that a variation of the cloudiness from 10 to 0 caused an increase in the heating load between 8 and 23%.

The variation of fluxes that play an important role in energy losses from the greenhouse is shown in Figure B.4. The following fluxes were considered: convective losses to soil, infiltration losses and convective and radiative losses from the cover. These fluxes can be used to calculate the energy balance for the overall greenhouse. The radiative losses from the internal components (i.e., soil, crop and thermal screen) to the sky were not considered because they are null here, the glass being opaque to infrared radiation. The variations of the contribution (in percentage) of each flux in the greenhouse heat losses, for both cloudiness levels and for the two states of the thermal screen, are shown in Figure B.5.

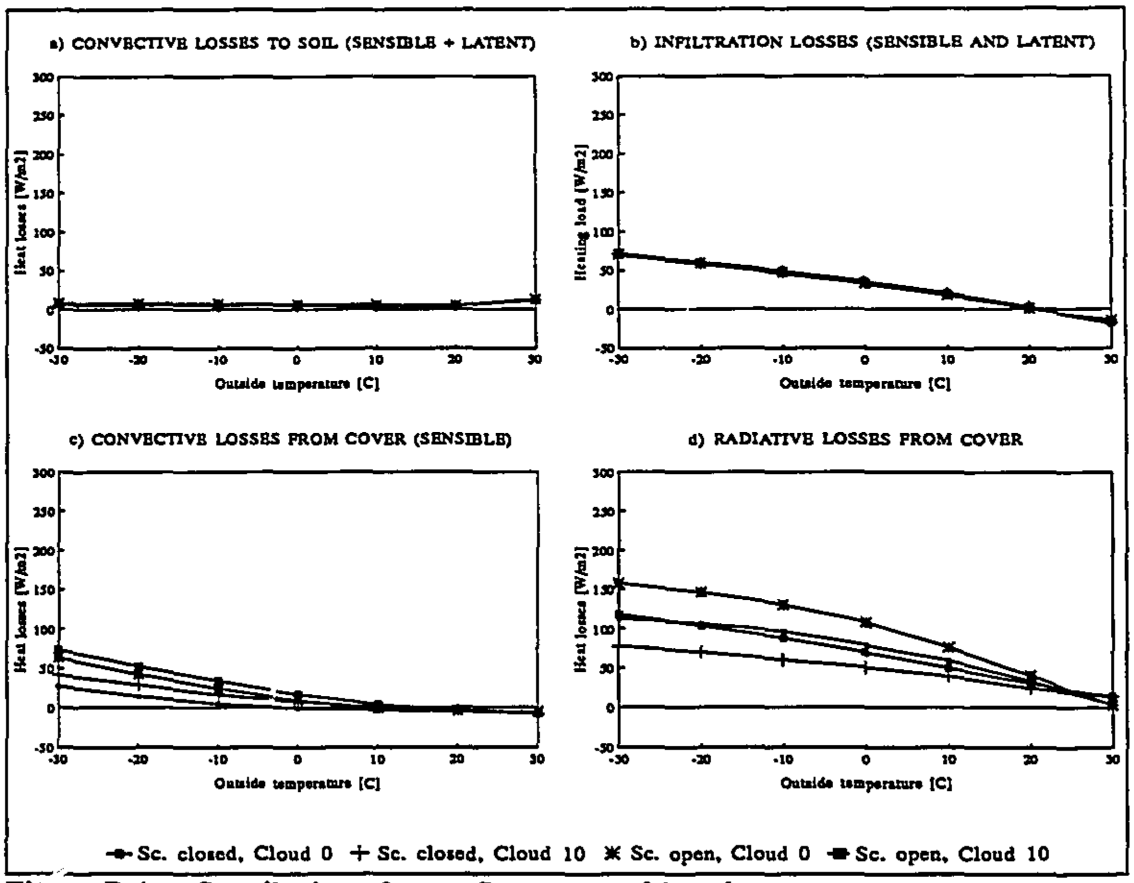

Figure B.4 Contribution of some fluxes to total heat loss.

It can be observed in Figure B.4a that the convective heat losses to the soil (sensible and convective combined) did not vary greatly and were always less than 10 W/m^2. They counted generally for less than 10% of total greenhouse losses (Figure B.5). The losses (sensible and latent) due to air infiltration, that were calculated assuming a constant rate of 1.0 air renewal per hour for both the gas and the internal air, are shown in Figure B.4b. The infiltration losses contributed for up to 38% of the total losses when the thermal screen was closed; with no thermal screen, they contributed for up to 28% of the total (Figure B.5). The convective losses by the cover were often not very important but went up to 70 W/m^2 (Figure B.4c) and contributed for up to 26% of the total losses. For an outside temperature as low as 0°C, there was a heat gain by the cover by convection. This means that the temperature of the cover was lower than that of the outside

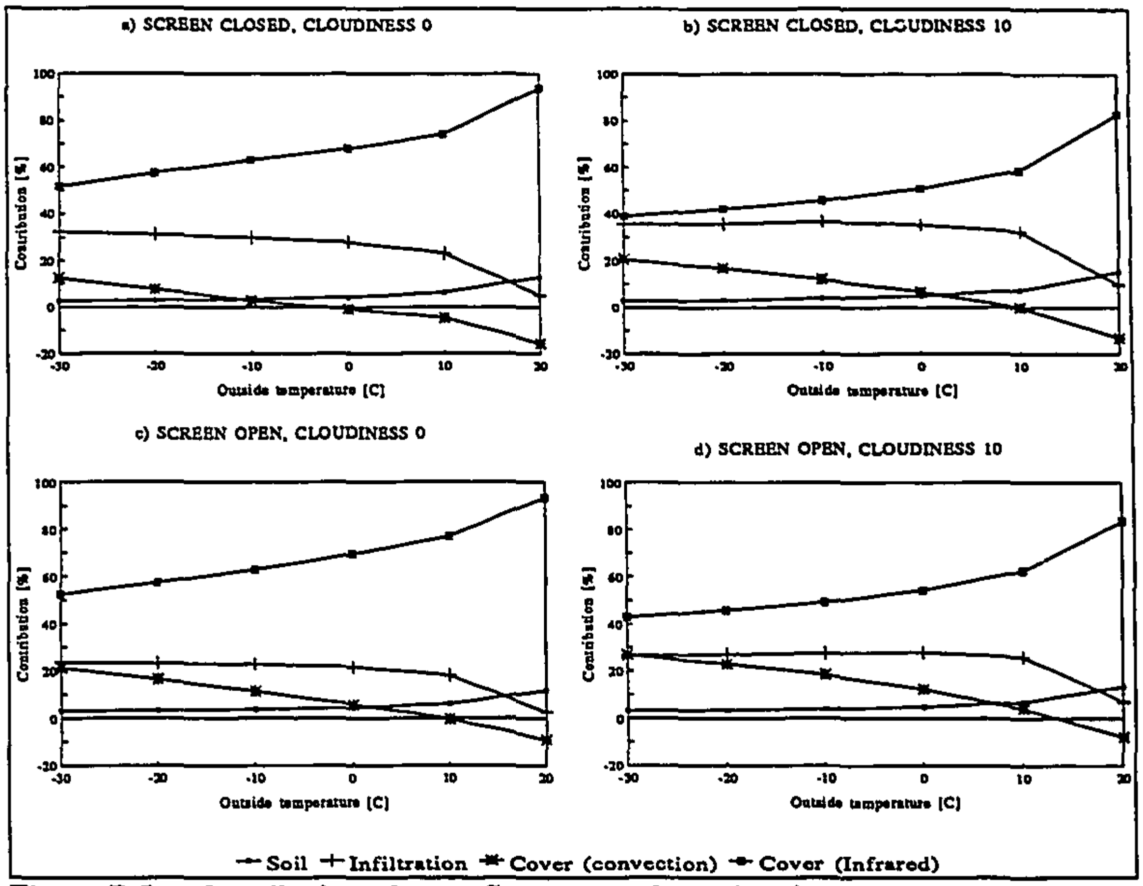

Figure B.5 Contribution of some fluxes to total heat loss in percentage.

temperature, probably due to the importance of the radiative losses from the cover. This phenomena was accentuated when the thermal screen was closed, for the reasons previously given. Figure B.4d shows the radiative losses by the cover. Their contribution to the total heat losses varied between 40 and 95% (Figure B.5).

B.3.4 Variation of the wind speed

In the second series of experiments for the steady-state analysis, the effect of the wind speed on greenhouse behavior was studied. The following conditions were maintained in all these experiments: constant temperature setpoint (20°C), constant exterior temperature (0°C) and no solar radiation. Simulations were done with wind speed fixed at seven levels (i.e., between 0.0 m/s and 13.3 m/s, in increments of 2.22 m/s), for both states of the thermal screen (closed and open) and for two levels of cloudiness (0 tenths and 10 tenths). This led to 28 simulations.

The energy consumption for different wind speeds is shown in Figure B.6. An increase in heating load with increasing wind speed was significant only when the thermal screen was open. This is in agreement with results obtained by Bailey (1978) who found that the addition of an aluminized thermal screen to a glasshouse reduced the effect of the wind speed on the heat losses to a non-significant level. With a cloudiness of 0 tenths, there was even a slight decrease in the heat losses when the wind speed increased (Figure B.6). This is once again due to the fact that the cover temperature was slightly lower than the outside air (because of the large amount of radiative losses) and that heat gains occurred, which increased as the wind speed increased. This can be seen in Figure B.7 in which it is shown how the convective heat losses by the cover varied with the wind speed.

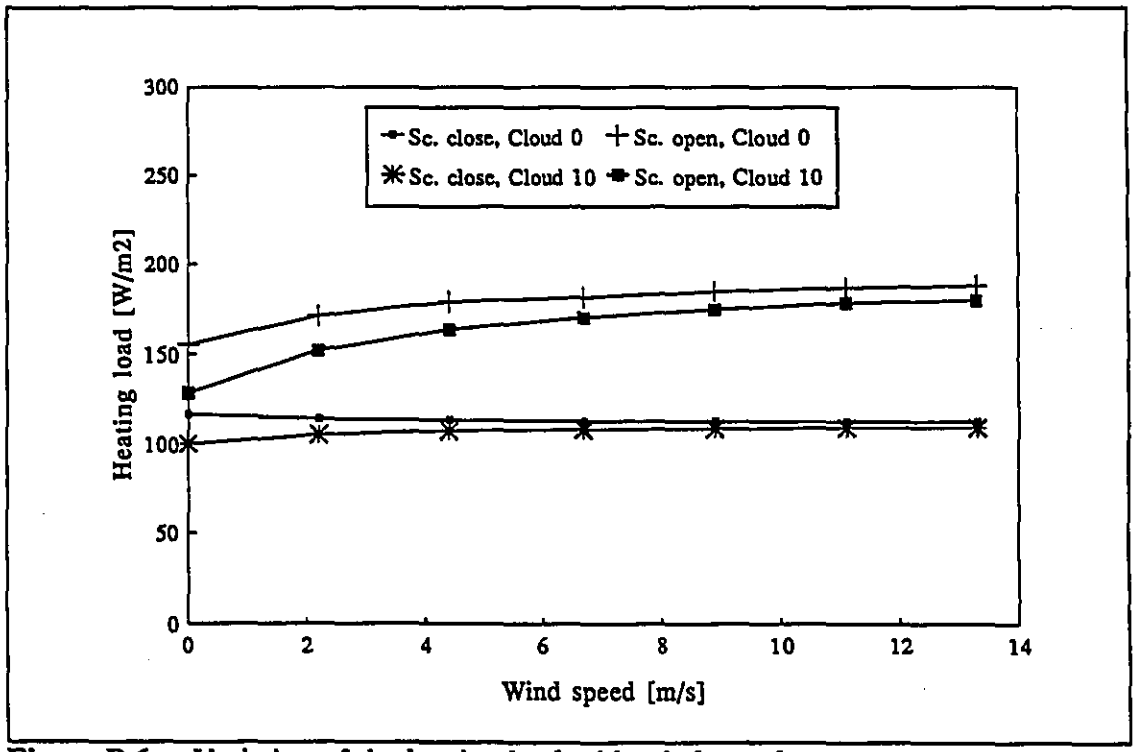

Figure B.6 Variation of the heating load with wind speed.

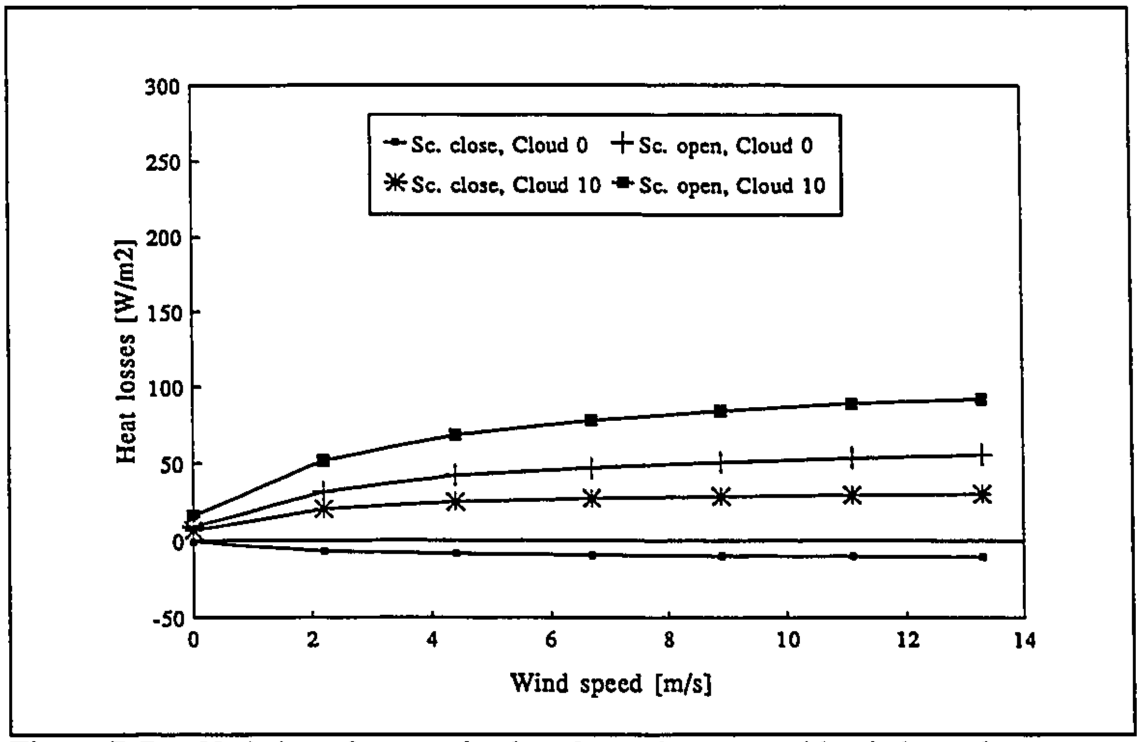

Figure B.7 Variation of convective heat losses by cover with wind speed.

Due to its configuration, the greenhouse model was not as sensitive to the wind speed as has been generally reported in the literature. A linear function is often used to relate the overall heat transfer coefficient to the wind speed (equation B.1):

$$U = a + b * u \tag{B.1}$$

where

U:	overall heat transfer coefficient	[W/m^2·°C]
u:	wind speed	[m/s]
a and *b*:	coefficients depending on the greenhouse.	

According to Bailey (1988), values reported in the literature vary between 5.2 W/m^2·°C and 6.5 W/m^2·°C for coefficient *a*, and between 0.3 J/m^3·°C and 0.6 J/m^3·°C for coefficient *b*. Aikman and Picken (1989) used a value of 5.2 W/m^2·°C for coefficient *a* and a value of 0.7 J/m^3·°C for *b*. With these values, if the wind speed changes from 0 m/s to 10 m/s, the overall heat transfer coefficient will increase between 60% and 135%. In Figure B.6, the largest heating load increase was for when there was no thermal screen and the cloudiness was 10 tenths. In this case, the heat loss increased from 130 to 180 W/m^2. This corresponds to an increase of the overall coefficient of about 35%, which is rather small compared to the values found in the literature.

Two main reasons can be suggested to explain the relatively low sensitivity of the greenhouse model to the wind speed. The first one is related to the radiative heat losses by the cover, which could have been overestimated (due to the underestimation of the sky temperature by the model of Dogniaux and Lemoine, 1984), thereby leading to an underestimation of the cover temperature (de Halleux *et al.*, 1991). This may have reduced the impact of the convective transfers between the cover and the outside air in the energy balance. This mechanism was the only one which was a function of the wind speed, which could account for the very low dependence of the overall transfer coefficient to this meteorological factor.

A second reason for the low variability of the heat losses with respect to the wind speed could be related to the fact that the infiltration rate was maintained constant throughout the simulations while it might have been dependent on wind speed (Townsend *et al.*, 1978; Whittle and Lawrence, 1960; Okada and Takakura, 1973). Infiltration rates can vary greatly from one greenhouse to another, depending on the leakage characteristics, which itself depends on the type of construction, the age of the structure and meteorological factors other than wind speed. Monteil (1985) reported infiltration rates increasing from 0.5 air renewals per hour (AR/h) to 3 AR/h for a wind speed increasing from 0 m/s to 10 m/s. Short and Bauerle (1977) noticed a doubling of the infiltration rate with a wind speed increasing from 0 km/h to 24 km/h. Infiltration levels of up to 3 AR/h were reported by Silveston *et al.* (1980). According to Hellickson and Walker (1983), infiltration rates can vary between 0.75 AR/h and 1.5 AR/h for a new glass construction and can go up to 4 AR/h for an old construction in poor conditions.

Many relationships between infiltration and wind speed have been used in different simulation models (Chiapale *et al.*, 1983; Rotz, 1977). Bellamy and Kimball (1986) and Kimball (1986) used the relationship developed by Okada and Takakura (1973) for a glasshouse, which was described by equation B.2:

$$V_g = 0.44 * u + 0.14 * (T_i - T_o)^{0.5} \qquad \textbf{(B.2)}$$

where

V_g:	air renewal rate	$[m^3/m^2 \cdot h]$
u :	wind speed	[m/s]
T_i:	inside temperature	[°C]
T_o:	outside temperature	[°C]

For instance, an infiltration rate of 2.25 AR/h can be calculated with equation B.2 for the greenhouse used in this project, a temperature difference of 20°C between the inside and outside and a wind speed of 13.3 m/s. With no wind, the infiltration rate is reduced to about 0.2 AR/h.

Two new sets of experiments were done with equation B.2 integrated into the greenhouse model. In the first set, the infiltration rate was calculated with equation B.2 but, as suggested by Kimball (1986) and Rotz (1977), it was cut by half when the thermal screen was closed. In the second set, the infiltration rate was calculated with the equation B.2, but was assumed to affect only Air 2 (gas). This approach corresponds better to the structure of the greenhouse model, in which all fluxes (except ventilation) are vertical. It represents what happens at the centre of a large greenhouse, where there are no walls. With this approach, there is no need to reduce the infiltration rate when the thermal screen is closed.

The results of the two sets of simulations are reported in Figures B.8a to B.8c. The influence of the wind speed on the heating load can be seen to be larger in general than with a constant infiltration rate. Without the thermal screen, the heating load increased between 55% and 80% when the wind speed increased from 0 m/s to 10 m/s, depending on the cloudiness level. These values are more close to those calculated with equation B.1 and with values of *a* and *b* found in the literature. However, there were no appreciable differences among the two sets.

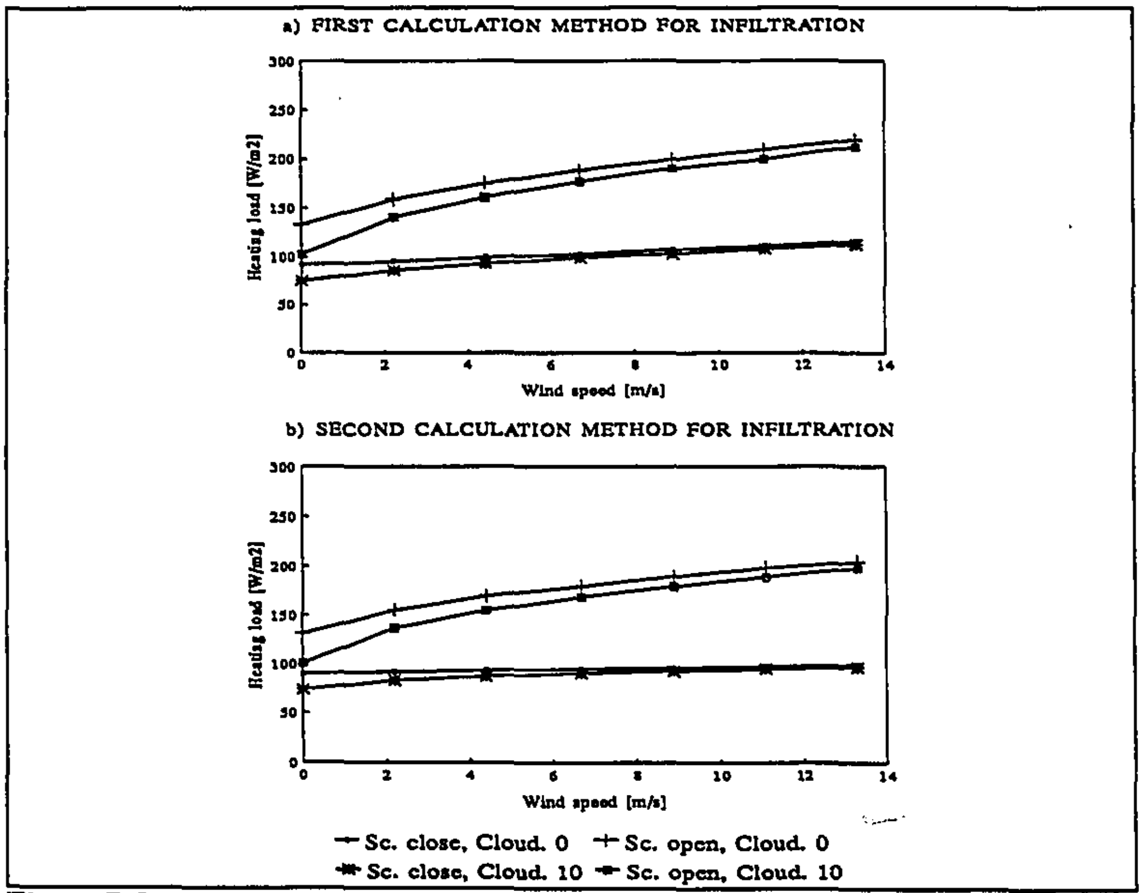

Figure B.8 Variation of the heating load with wind speed for different ways of calculating the infiltration rates.

Modification of the infiltration rate calculation affected the heating load considerably. This implies that the greenhouse model is very sensitive to the air renewal rate. This was also reported by De Halleux (1989). Chalabi and Bailey (1991) also noticed the importance of the leakage rate in a sensitivity study of a glasshouse model. Such leakages are reduced in new types of construction, but they can still be important in some installations submitted to sub-zero temperatures, where structures are stressed due to successive periods of frost and thaw. Also, many greenhouse are equipped with vents for natural ventilation, which are not always tightly closed. The infiltration variable should be carefully calibrated for simulation of the behavior of a specific installation.

For the functional simulation, the equation proposed by Okada and Takakura (1973) was adopted to calculate the infiltration rates from the gas volume. The choice is arbitrary, but this equation furnished results that were more comparable to what was observed in greenhouses by several researchers. Also, it added some interesting variability to the greenhouse model behavior.

B.3.5 Variation of the solar radiation

A third series of experiments was done to investigate the effect of solar radiation on inside air and crop temperatures, heating load and ventilation load. In all experiments, the following conditions were maintained: outside temperature of 0°C, wind speed of zero, cloudiness of 0 tenths and thermal screen open. The diffuse component of the solar radiation was considered to account for 20% of the global radiation. The hour and the date were maintained constant at respectively 12:00 am and March 10th; this implied that the transmittivity, reflectivity and absorptivity of the cover for the direct component of solar radiation were maintained constant during the simulations, allowing steady-state conditions to be reached.

The experiments were divided in two subsets. In the first subset, the effect on ventilation was studied. Seven levels of global radiation intensity from 0 to 900 W/m^2 were considered. The steady-state inside air temperature (Air 1), crop temperature and ventilation load are shown in Figure B.9 for the different radiation levels. The ventilation was observed to start at a solar intensity of about 300 W/m^2 for an outside temperature of 0°C. The crop temperature was also observed to be generally higher than the air temperature, the difference being as great as 4°C at high radiation intensities. From the simulation outputs, it was calculated that the soil and the cover each absorbed less than 10% of the incoming (outside) solar radiation. The crop absorbed about 50% of the solar radiation arriving outside the greenhouse.

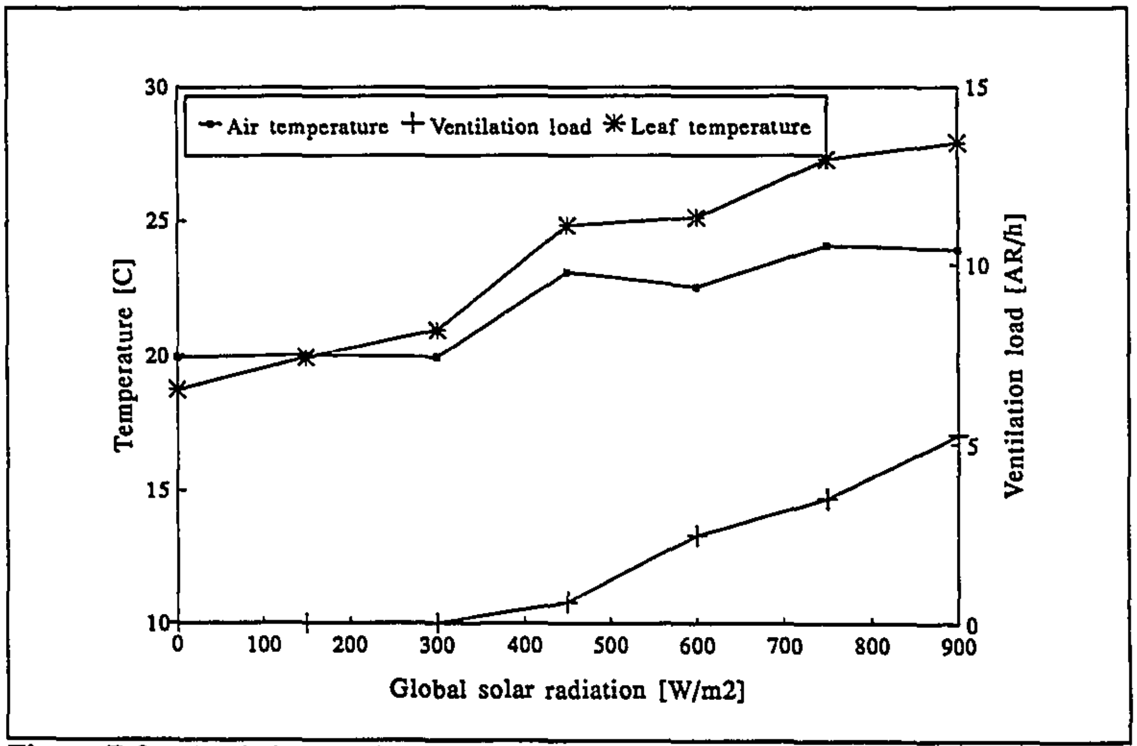

Figure B.9 Variation of air temperature, crop temperature and ventilation with solar radiation intensity.

In the second subset of experiments, the maximum solar radiation intensity was set to 550 W/m^2, the number of intensity levels was increased to 12 and there was no ventilation. This allowed the impact of solar radiation intensity on the heating load and on the inside air and crop temperatures to be studied. As shown in Figure B.10, the heating load became 0 W/m^2 when the radiation intensity was 350 W/m^2. At higher intensities, the air and crop temperatures started to increase. For solar radiation intensities less than 150 W/m^2, the crop temperature was less than the air temperature; the situation was inverted when the solar radiation intensity was more than 150 W/m^2.

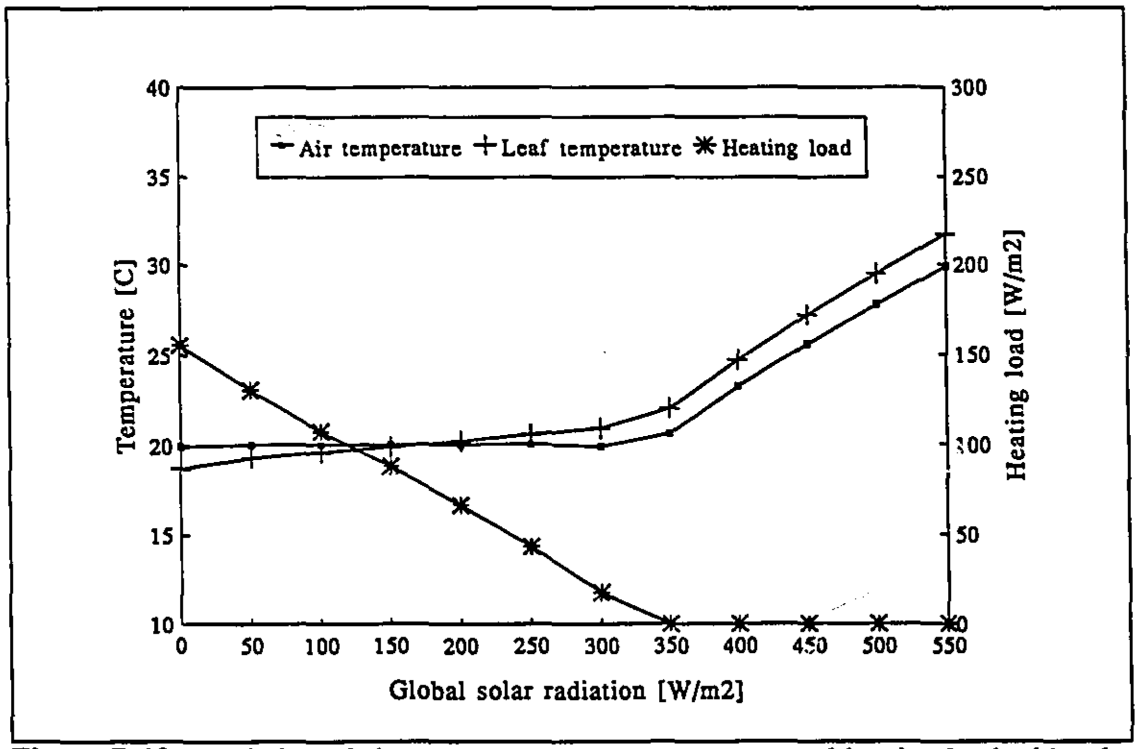

Figure B.10 Variation of air temperature, crop temperature and heating load with solar radiation intensity.

B.3.6 Conclusion

The steady-state response of the greenhouse model, as it was configured, was generally adequate. Many aspects of the model could certainly be improved, such as the calculation of the infiltration rates and crop transpiration. Ideally, a sensitivity analysis should be carried to determine the priority in which the parameters should be adjusted during a calibration process if the simulation had to represent a specific greenhouse system. However, the greenhouse model as it was configured was appropriate for the present study.

APPENDIX C. THE CROP MODEL

C.1 CROP PARAMETER VALUES

The parameters used in this research for the crop model and their given values are listed in Table C.1. These values are the same as those included in the original program SUCROS87, excepting the weight of each component of the plant. The weights of the components were adjusted so that the total weight would correspond to the value used in the greenhouse model. It can be noted that with a moisture content of 3.5%, the total fresh weight is 6 kg/m^2. All parameters were assumed to remain constant during functional simulations, which were done for periods less than one month in this project.

Table C.1 Crop parameters for the crop model.

Diffuse light extinction coefficient [%]	80
Scattering coefficient of leaves for PAR* [%]	20
Leaf area index [m^2/m^2]	3
Percentage of dry weight [%]	3.5
Reference temperature for maintenance respiration [°C]	25
Q_{10}	2
Maintenance coefficients:	
- leaves	0.03
- stems	0.015
- roots	0.01
- fruits	0.01
Assimilate requirements [g CH_2O/g dry matter]:	
- leaves	1.37
- stems	1.45
- roots	1.39
- fruits	1.37
Dry matter partition factor [%]:	
- leaves	17
- stems	8
- roots	3
- fruits	72
Dry weight [x 10^{-3} kg]:	
- leaves	94.5
- stems	18.9
- roots	44.1
- fruits	52.5

* Photosynthetically active radiation

C.2 ANALYSIS OF THE CROP MODEL

C.2.1 Introduction

The crop model is composed of algebraic equations. Therefore, its instantaneous response can be obtained directly from one set of input values without it being affected by the inputs at previous times. The main input variables for the crop model are the leaf temperature, photosynthetically active radiation (PAR) and the CO_2 concentration. They are used in the calculation of photosynthesis and respiration and, consequently, net CO_2 assimilation rate. Three sets of calculations were done with the crop model to see how the net CO_2 assimilation rate will vary with different sets of input values. The intent was to investigate the effect of a variation of the leaf temperature in combination with variations in solar radiation intensity and CO_2 concentration on the model. This was pursued because, in this research, the main goal of the cognitive controller was to control air temperature, which plays a large role in the determination of the leaf temperature. The crop parameter values used for the calculations were those listed in Section C.1.

In most calculations, the leaf temperature ranged between 15°C and 37°C (in increment of 2°C), to simulate the possible range of values that can be found in greenhouses. The solar radiation levels chosen for the calculations were arrived at by the following reasoning: 1) the solar radiation levels can reach 1000 W/m^2 in summer (total visible spectrum) at Montréal, Canada, 2) a maximum of 80%, or 800 W/m^2, is transmitted through the cover and 3) only 50% of this is PAR, giving a final value of 400 W/m^2. In all calculations, the solar height (which is also an input to calculate the photosynthesis) was kept constant at 45°. The ratio of diffuse to global solar radiation was kept constant at 0.4.

C.2.2 First set of calculations

In the first set of calculations, the PAR intensity ranged between 25 W/m^2 and 150 W/m^2 in increment of 25 W/m^2. The value of 150 W/m^2 corresponds approximately to an outside level of 375 W/m^2 (total visible spectrum), which in turn corresponds to the magnitude of intensities often observed during winter. The CO_2 concentration ranged between 330 ppm and 990 ppm. The net assimilation rate varied with the leaf temperatures and solar radiation intensities for each CO_2 concentration, as shown in Figure C.1. An increase in crop productivity was observed as CO_2 concentration increased. At a given CO_2 concentration, it was observed that the leaf temperature at

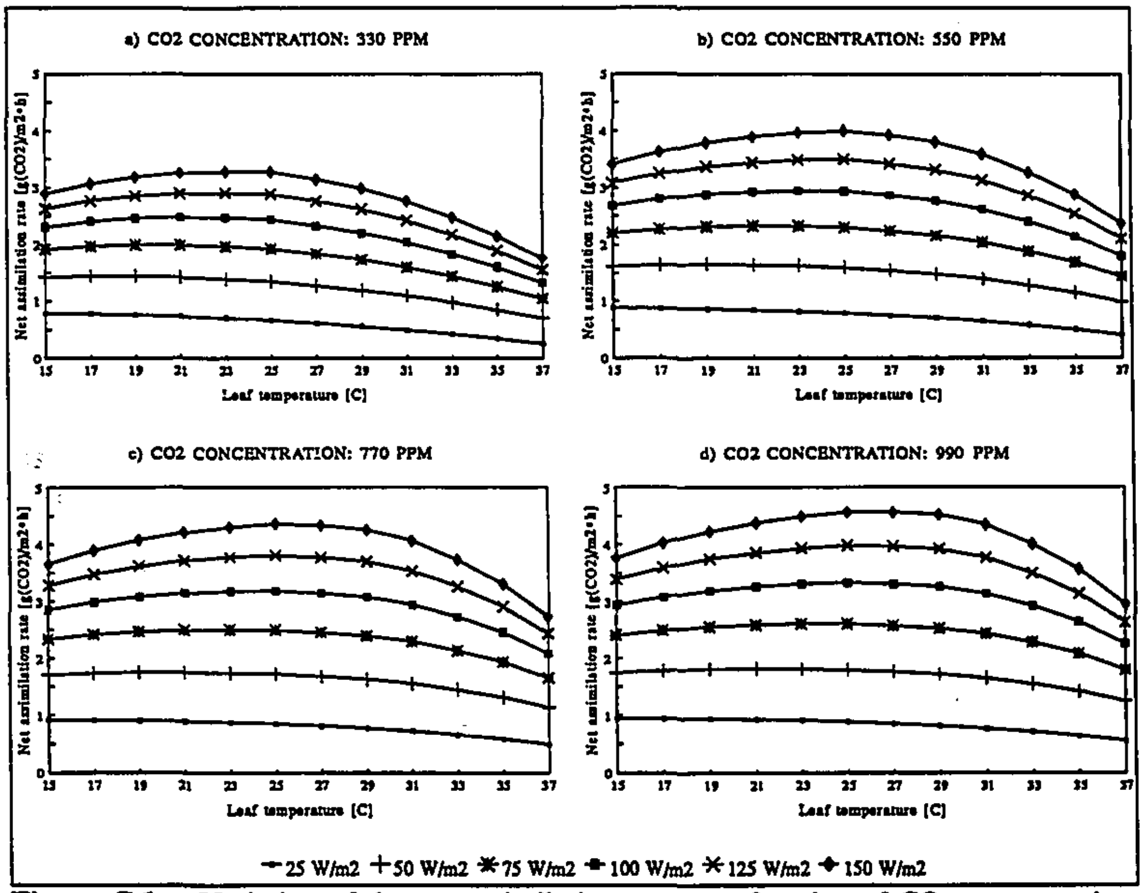

Figure C.1 Variation of the net assimilation rate as a function of CO_2 concentration, leaf temperature and solar radiation intensity (less than 150 W/m^2).

which the maximum net assimilation rate occurred gradually shifted from about 15-17°C to 23-27°C when the solar radiation increased from 25 to 150 W/m^2. The shift was larger at higher CO_2 concentrations. At a CO_2 concentration of 990 ppm and a PAR intensity of 150 W/m^2, a shift of the temperature from 15°C to 25°C increased the biomass production rate by about 20%.

C.2.3 Second set of calculations

In the second set of calculations, the analysis was extended over a larger range of PAR intensities (up to 400 W/m^2 PAR), which corresponds to values that can be found in the summer time. This was done for three CO_2 concentrations: 330, 660 and 990 ppm (Figures C.2). The same conclusions as before were reached. For a given CO_2 concentration, the optimal leaf temperature, i.e., the temperature corresponding to the maximum assimilation rate, increased as the PAR intensity increased. For example, at a PAR intensity of 400 W/m^2 and a CO_2 concentration of 990 ppm (Figure C.2c), a change of the leaf temperature from 15°C to 29°C increased the biomass production rate by more than 40%. For a given PAR intensity, the optimal temperature increased with the CO_2 concentration.

C.2.4 Third set of calculations

In the third series of calculations, the intent was to see how the net assimilation rate varies with both the PAR intensity and the CO_2 concentration, the leaf temperature being kept constant at 21°C. The net growth rate increased with both input variables (Figures C.3a and C.3b). The only difference between the two graphs is that, for Figure C.3b, the range of CO_2 concentrations is larger than for Figure C.3a. In general, the marginal benefit (i.e., the increase of biomass production rate per unit of increase of CO_2 concentration) decreased when the CO_2 concentration increased. The benefit increases were very high for CO_2 concentration between 300 and 900 ppm, but they become

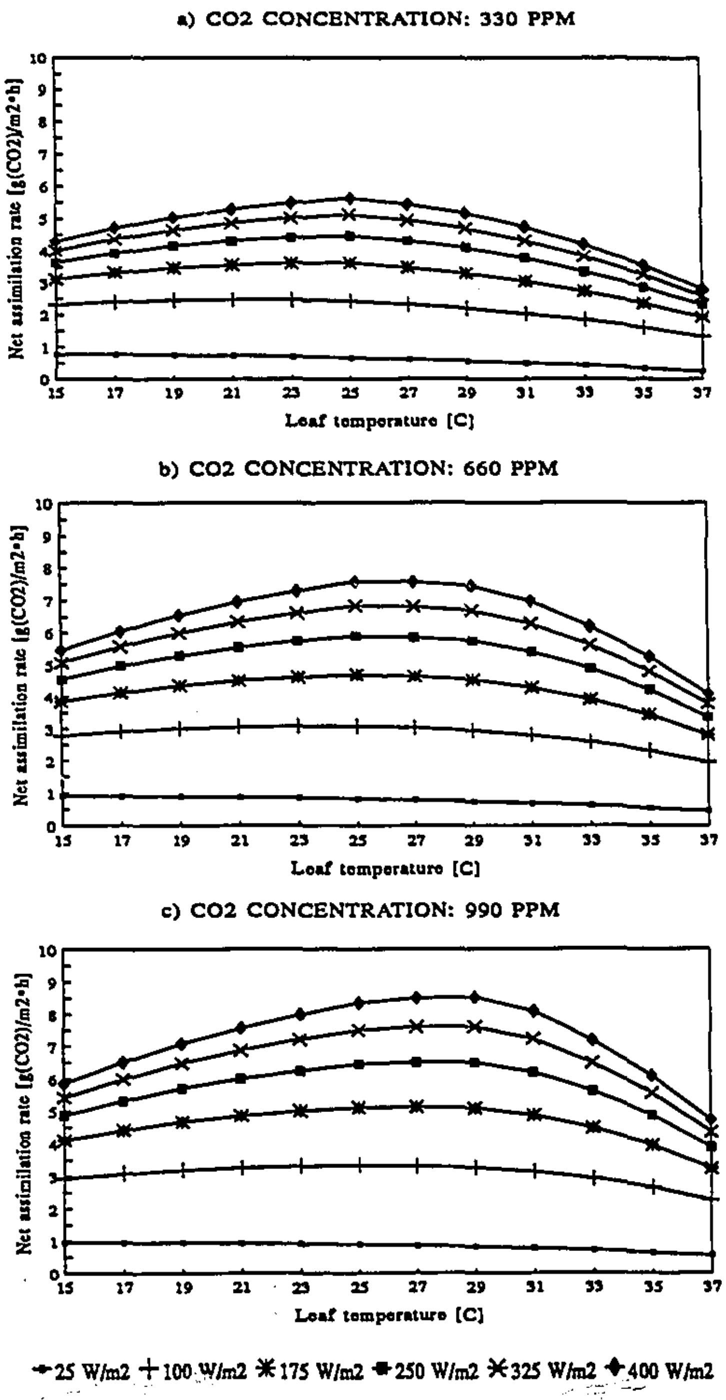

Figure C.2 Variation of the net assimilation rate as a function of CO_2 concentration, leaf temperature and solar radiation intensity (less than 400 W/m^2).

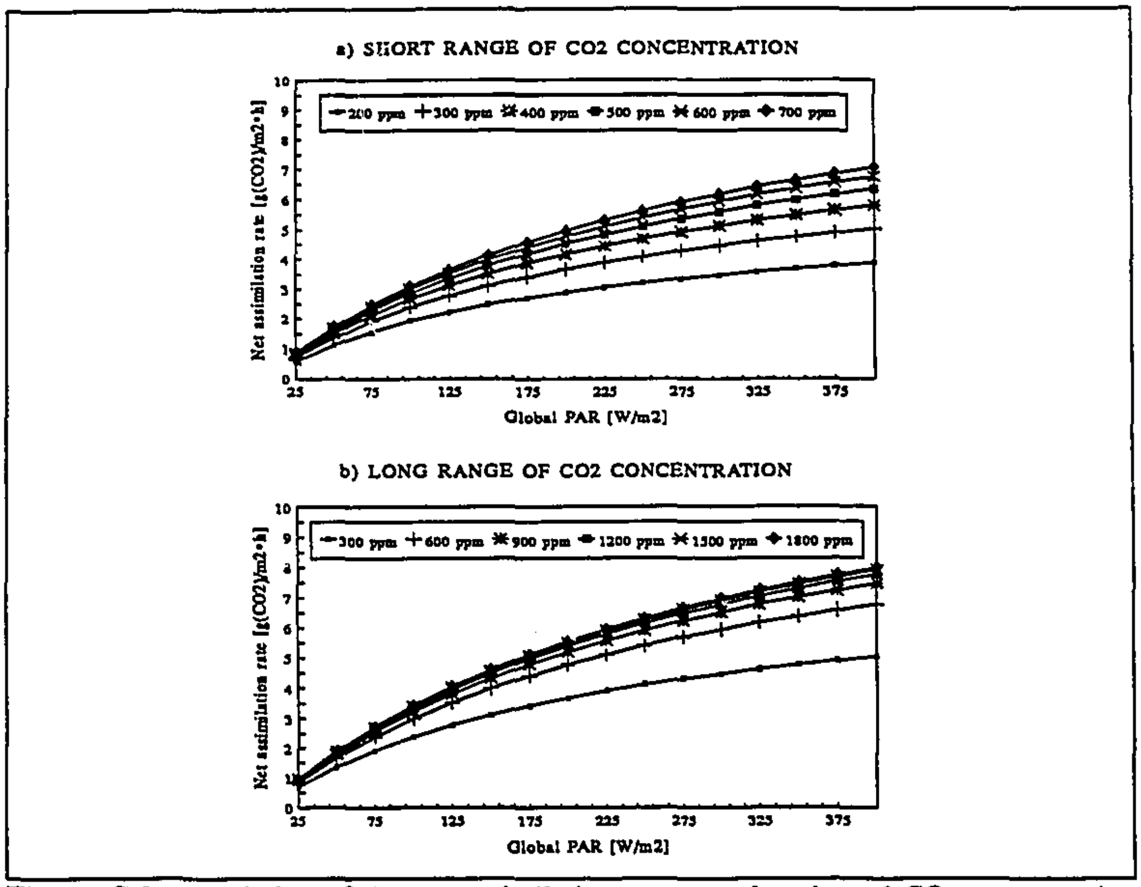

Figure C.3 Variation of the net assimilation rate as a function of CO_2 concentration and solar radiation intensity (leaf temperature held constant at 21°C).

negligible at high CO_2 concentrations such as 1500 ppm (Figure C.3b).

C.2.5 Conclusion

Some calculations were performed to become familiar with some of the possible responses of the crop model under various conditions. The model showed to be sensitive to the three main input variables (i.e., leaf temperature, PAR, CO_2 concentration). A particularly interesting point for this research is that the net assimilation rate varied with the leaf temperature for a given solar radiation intensity and CO_2 concentration. Thus, different temperature control strategies may have a different impact on crop production. The variations induced on the net assimilation rate by different CO_2 concentrations did not concern specifically the present research. However, this will become an important aspect in the eventual addition of CO_2 balances and CO_2 control in the functional simulation.

APPENDIX D. DYNAMIC BEHAVIOR OF THE GREENHOUSE SYSTEM

D.1 INTRODUCTION

Simulation experiments were performed to study the behavior of the complete functional greenhouse system in absence of the cognitive controller. The specific objectives were to 1) verify that the simulation results are reasonable and free of aberrations, 2) compare the results with values found in the literature and 3) become familiar with the system behavior in different situations, especially under various meteorological conditions. Many variables were considered in this analysis and particular attention was given to the inside air temperature (Air 1), the heating load and the biomass production rate.

In all simulations, the day temperature setpoint was maintained constant at 21°C, while the night temperature setpoint was 17°C; these values correspond to those used in a production context (CPVQ, 1984). The file COGNITI.101, listed in Appendix G.7, was used by the module COGNITI for this purpose. The setpoint for ventilation was kept constant at 25°C (see Figure IV.5). The simulations were done for all months of the year 1982 using meteorological data from the Canadian Atmospheric Environment Service (AES) as weather input.

From the simulation, a large number of output variables is available for the analysis of the greenhouse system behavior. For example, all fluxes, temperatures and humidities of the greenhouse model constitute a set of about 60 variables. Also, values for about 10 variables are produced by the crop model. Only some of these values produced during the simulations are reported here.

D.2 GENERAL BEHAVIOR OF THE GREENHOUSE SYSTEM

Figure D.1 illustrates the variation of the inside temperature and the heating load during the first 15 days of three months: January, March and May. Figure D.2 shows the outside temperature and solar radiation intensities during these periods. In January, the internal

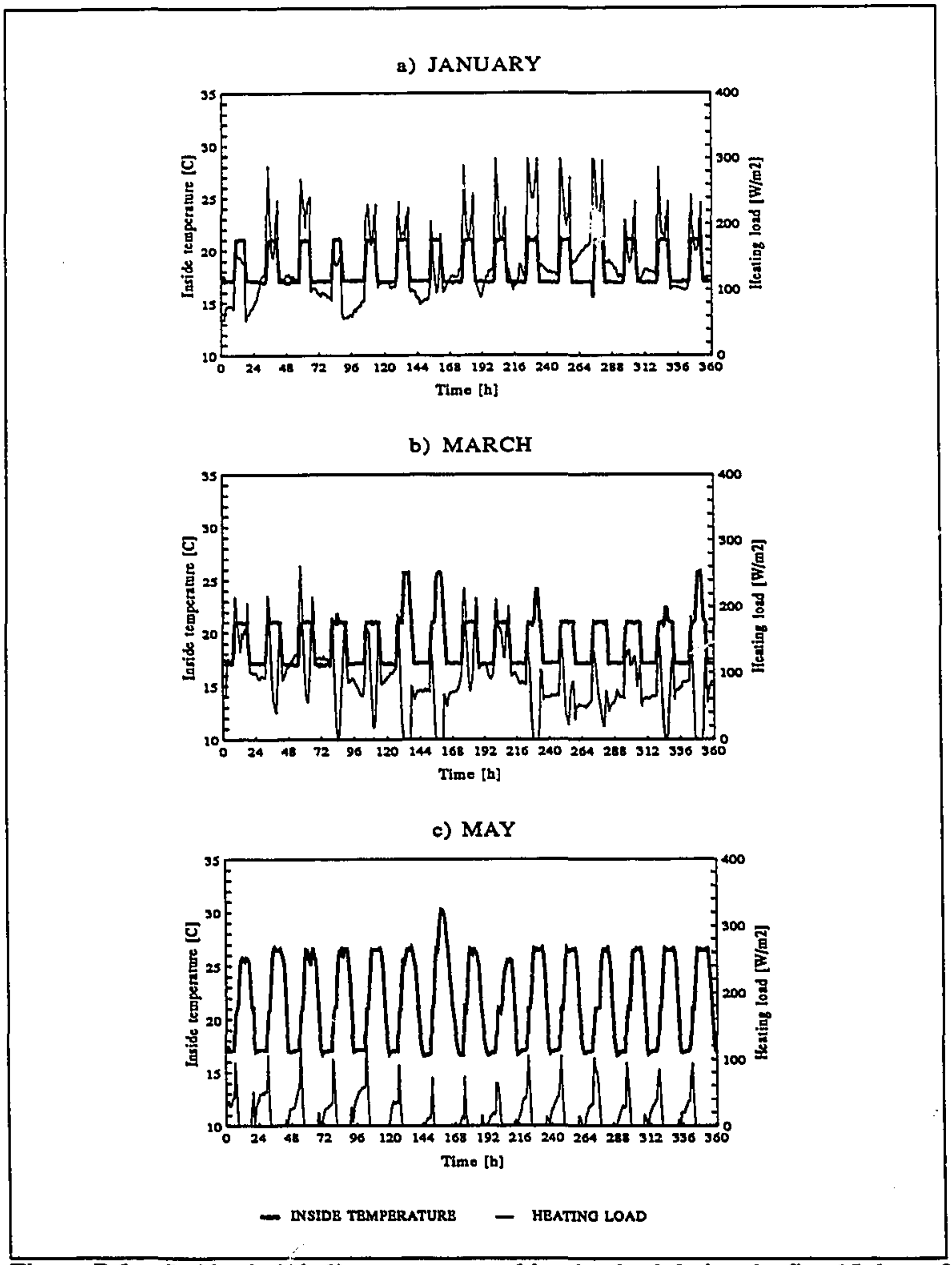

Figure D.1 Inside air (Air 1) temperature and heating load during the first 15 days of January, March and May 1982.

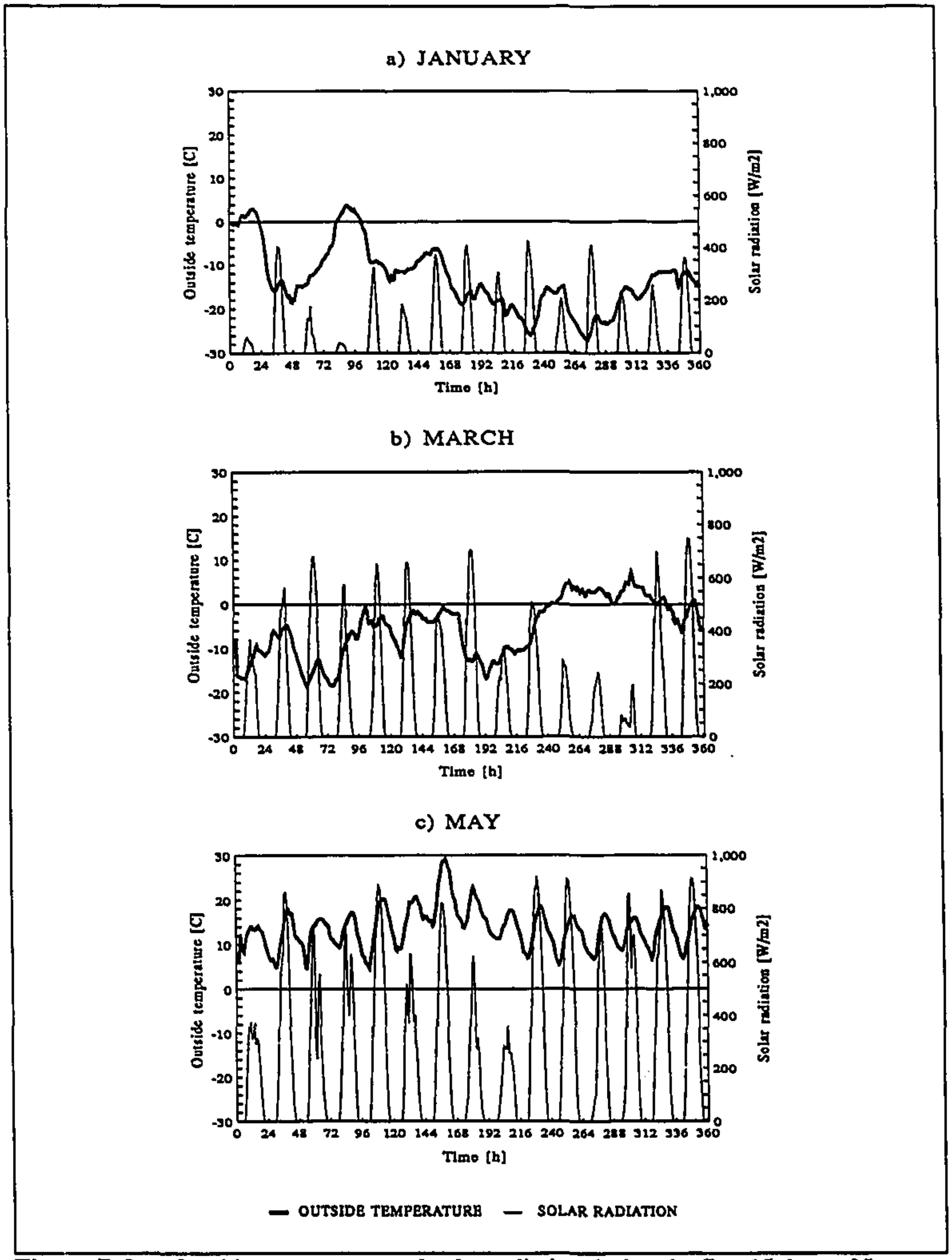

Figure D.2 Outside temperature and solar radiation during the first 15 days of January, March and May 1982.

temperature generally followed the setpoints (Figure D.1a), while in March the temperature occasionally rose above the setpoints (Figure D.1b). In May, such rises above the setpoint (these will be called "overheating" hereafter) happened every day. Overheating occurs when solar gains become larger than heat losses. With respect to heating, it can be seen in Figure D.1 that the heating requirements were often larger during the day than during the night in January. This is because, during the day, higher temperatures were maintained and the thermal screen was open, which increased the overall heat transfer coefficient. The peaks in heating happened at the beginning and the end of the day, while the thermal screen was open but the solar radiation intensity was low. Similar peaks were observed by Amsen (1986). In May, heating occurred mainly during the night; the peaks occurred when the thermal screen was opened and the temperature setpoint for heating was increased to its day value (both occurred about at the same time).

Figure D.3 shows the temperature for both greenhouse air volumes, i.e., for Air 1 (inside air) and Air 2 (gas). It may be noted that the temperature difference between the two volumes was large during the nights, when the thermal screen was closed. The difference decreased when the outside temperature increased (Figures D.3a to D.3c). During the day, often there was also a temperature difference between the air volumes, as was previously noted during the steady-state analysis of the greenhouse model (Section B.3.3). This difference was as large as 4°C during January (Figure D.3a), while it was almost negligible during May (Figure D.3c).

In Figure D.4, the difference between the crop and the inside air temperature (Air 1) is illustrated. Generally, during the night, the crop temperature was lower than the air temperature. The difference decreased from January to May. During the day, when the solar radiation intensity was high enough, the crop temperature was higher than the air temperature. In May, this difference could be as large as 4°C (Figure D.4c). These results correspond to those presented by other authors (Takakura *et al.*, 1971).

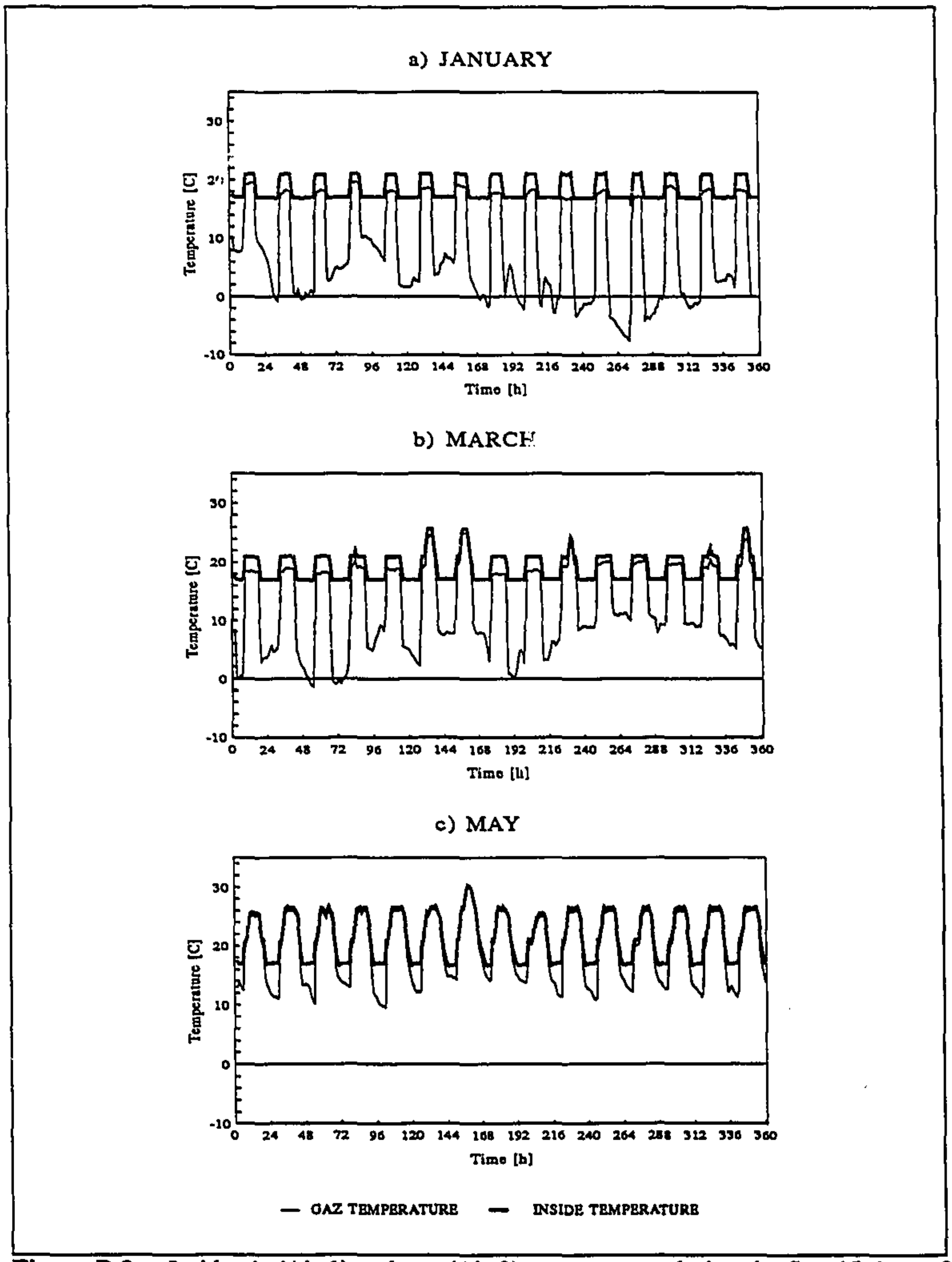

Figure D.3 Inside air (Air 1) and gas (Air 2) temperatures during the first 15 days of January, March and May 1982.

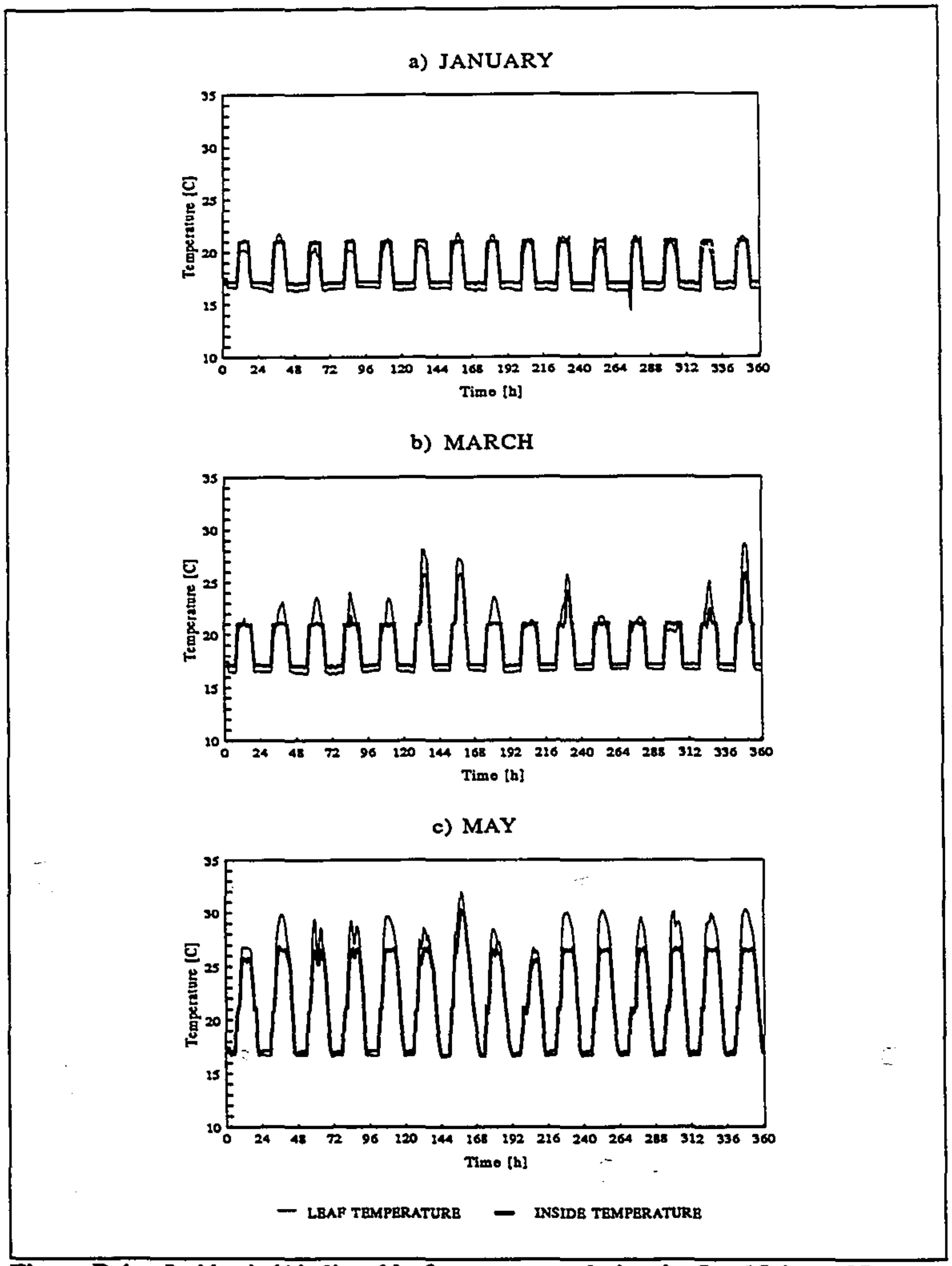

Figure D.4 Inside air (Air 1) and leaf temperatures during the first 15 days of January, March and May 1982.

The ventilation rate and the internal air temperature during the first 15 days of March, April and May 1982 are shown in Figure D.5. The graph for January is not shown here since no ventilation occurred during this month. Ventilation occurred very infrequently during March, but in May, ventilation occurred almost every day. In general, the ventilation system kept the inside temperature to values between 26°C and 27°C.

Some simulation results for each of the 12 months of 1982 are summarized in Table D.1. The two first columns are respectively the monthly average outside temperatures and the monthly total solar energy received. The third column is the average inside air temperatures (Air 1), which were higher during the warm season. This is due to two

Table D.1 Monthly results for Montréal, 1982.

	Outside temperature [°C] (1)	Solar radiation [MJ/m^2] (2)	Inside temperature [°C] (3)	Leaf temperature [°C] (4)	Heating load [MJ/m^2] (5)	Ventilation load [AR/h] (6)
January	-15.0	185.2	18.4	17.8	388.2	0.0
February	-9.0	267.9	18.8	18.5	256.2	0.0
March	-2.5	427.9	19.5	19.6	202.0	0.1
April	4.3	474.2	20.2	20.4	142.8	0.8
May	15.3	648.0	22.0	22.6	33.1	3.8
June	17.0	586.9	21.9	22.6	21.6	4.3
July	21.2	731.3	23.9	24.6	11.6	11.0
August	17.6	537.7	21.9	22.5	22.2	4.3
September	14.9	366.1	20.6	20.9	40.6	2.0
October	9.0	286.3	19.6	19.6	93.3	0.3
November	3.6	129.6	18.6	18.3	170.2	0.0
December	-2.7	117.7	18.4	17.9	251.2	0.0

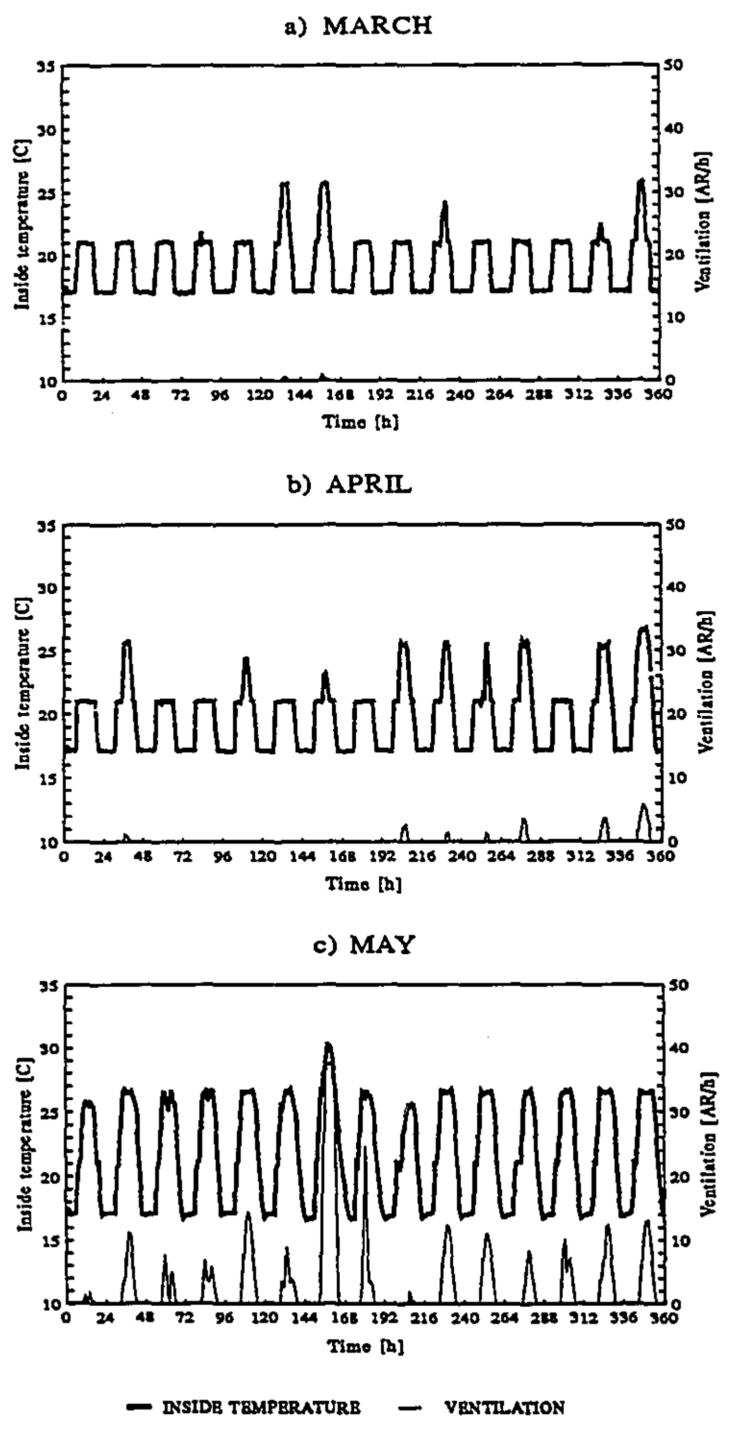

Figure D.5 Inside air (Air 1) temperature and ventilation during the first 15 days of March, April and May 1982.

major reasons. First, the length of the day increases when passing from the winter to the summer months, which implies that the higher temperature setpoint (21°C) was maintained over longer periods during summer days. Second, overheating occurred more frequently during the summer. The monthly average leaf temperatures are shown in the fourth column. During winter, these were lower than the inside air temperature, and higher during the period May to September. The two last columns are, respectively, the total energy required for heating and the monthly average ventilation rate. Heating occurred even during the warmest months of the summer, due to the configuration of the heating control subsystem and important radiative losses.

D.3 HEATING LOAD

A set of simulations was performed to analyze the effect of the absence of the thermal screen on the heating load of the greenhouse. In these simulations, the thermal screen was maintained open during both the night and the day. The results were than compared with the results obtained when the thermal screen was open during the day and closed during the night. Figure D.6 shows the results obtained for the first 10 days of January 1982. Without the thermal screen, nighttime energy consumption was 30 to 50% greater. A similar difference was observed for the months between February and April (not shown here). This is slightly higher than what was reported in Section B.3.3 (i.e., during the steady-state analysis of the greenhouse model), where values between 20% and 40% were found.

The total heating loads with and without the thermal screen are shown in Table D.2 for all months of the year except July and August (Columns 1 and 2). The difference in percentage is shown in column 3. The difference varied between 28.4 and 32.1%. The overall annual difference was 30.6%. This corresponds to the values generally found in the literature (CPVQ, 1984). Bailey (1988) calculated an annual saving of 32.6% when an aluminized thermal screen was added to a glasshouse.

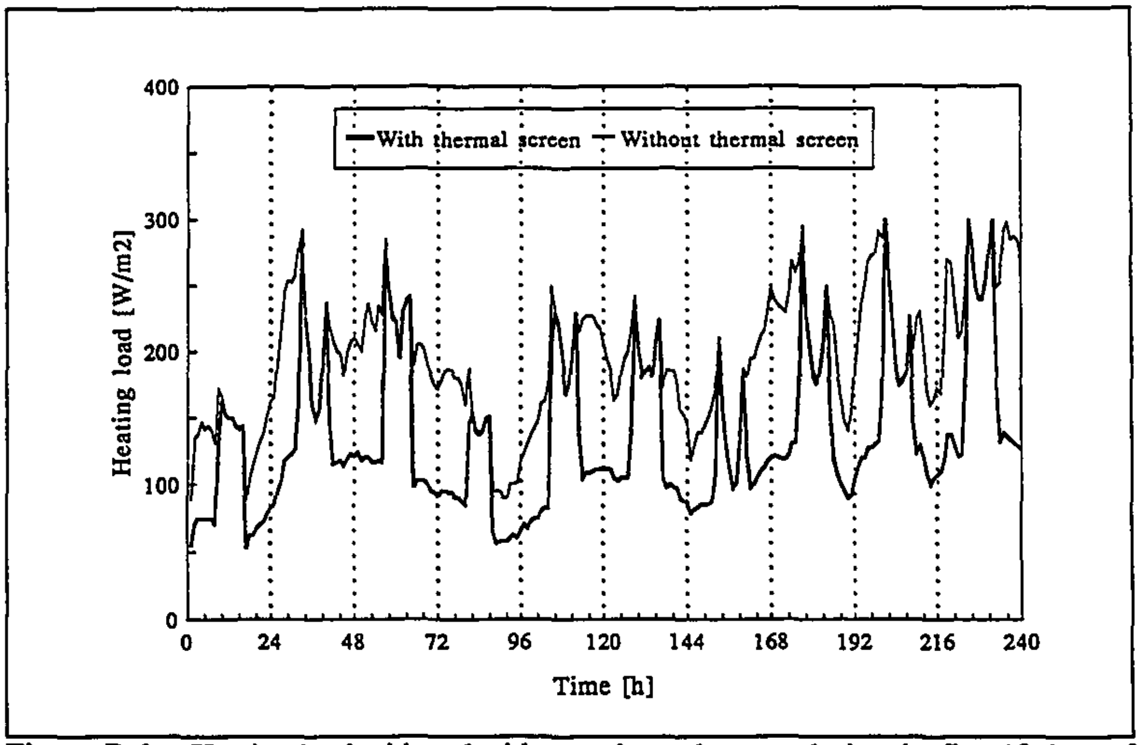

Figure D.6 Heating load with and without a thermal screen during the first 10 days of January 1982.

In Table D.3, the results obtained with the thermal screen are compared with some values found in CPVQ (1984). The first two columns are respectively the energy required to heat the simulated greenhouse and the number of degree-days during each month of 1982 (AES data used in the simulation). The degree-days were calculated on a basis of 18°C. Column 3 contains the heating requirements for a double-layer polyethylene multispan greenhouse in the region of Montréal. These values were calculated from the oil consumption values reported by CPVQ (1984), considering a furnace efficiency of 75% and a calorific value of 38850 kj/L of oil. Column 4 is the number of degree-days corresponding to the oil consumption values reported by CPVQ (1984). Considering the number of heating degree-days for both situations (columns 2 and 4), a certain correspondence can be observed (in magnitude) between the consumption of the glasshouse equipped with the thermal screen and one with double-polyethylene (columns

Table D.2 Difference in heating load between the greenhouse with and without a thermal screen.

	Heating load (no thermal screen) [MJ/m^2] (1)	Heating load (with thermal screen) [MJ/m^2] (2)	Difference [%] (3)
January	569.8	388.2	31.9
February	373.3	256.2	31.4
March	288.5	202.0	30.0
April	201.5	142.8	29.1
May	48.5	33.1	31.8
June	31.8	21.6	32.1
September	58.0	40.6	30.0
October	131.7	93.3	29.2
November	237.6	170.2	28.4
December	362.6	251.2	30.7
All months	2303.3	1599.2	30.6

1 and 3). The overall heat loss coefficient of a glasshouse is generally considered to be 6.46 W/m^2_c·°C (CPVQ, 1984) and if it is reduced on average by about 30% when a thermal screen is added, an overall transfer coefficient of 4.5 W/m^2_c·°C is obtained. Considering that, for a double-polyethylene cover, the overall heat transfer coefficient is 4.0 W/m^2_c·°C (CPVQ, 1984), which is relatively close to the previously calculated value (4.5 W/m^2_c·°C), it can be concluded that the simulated greenhouse furnished results that are fairly realistic.

Table D.3 Difference in heating load between simulation and CPVQ (1984).

	Heating load (simulation) [MJ/m^2] (1)	Degree-days (AES, 1982) (2)	Heating load (CPVQ, 1984) [MJ/m^2] (3)	Degree days (CPVQ, 1984) (4)
January	388.2	1023	384.6	861
February	256.2	755	276.8	770
March	202.0	636	204.0	643
April	142.8	412	136.9	392
May	33.1	103	84.5	169
June	21.6	45	14.6	27
July	11.6	13	----	5
August	22.2	46	5.8	23
September	40.6	108	61.2	100
October	93.3	279	116.6	296
November	170.2	432	192.3	497
December	251.2	641	358.4	770

D.4 BIOMASS PRODUCTION

The hourly average net crop assimilation rate (gross photosynthesis rate minus maintenance respiration rate) during the first 15 days of the months of January, March and May 1982 is shown in Figure D.7. The respiration rates did not vary considerably from night to night, compared to the photosynthetic rates during the day. The respiration rates increased only slightly for warmer temperatures. The photosynthetic rates were quite low in January, about 40% less than those in May. The daily respiration, photosynthesis and fresh fruit production rates for the months of January, March and May 1982 are shown in Figure D.8. For several days in January, the photosynthetic rate equalled the respiration rate, leading to no net production of crop. The total growth sometimes varied widely from one day to the next.

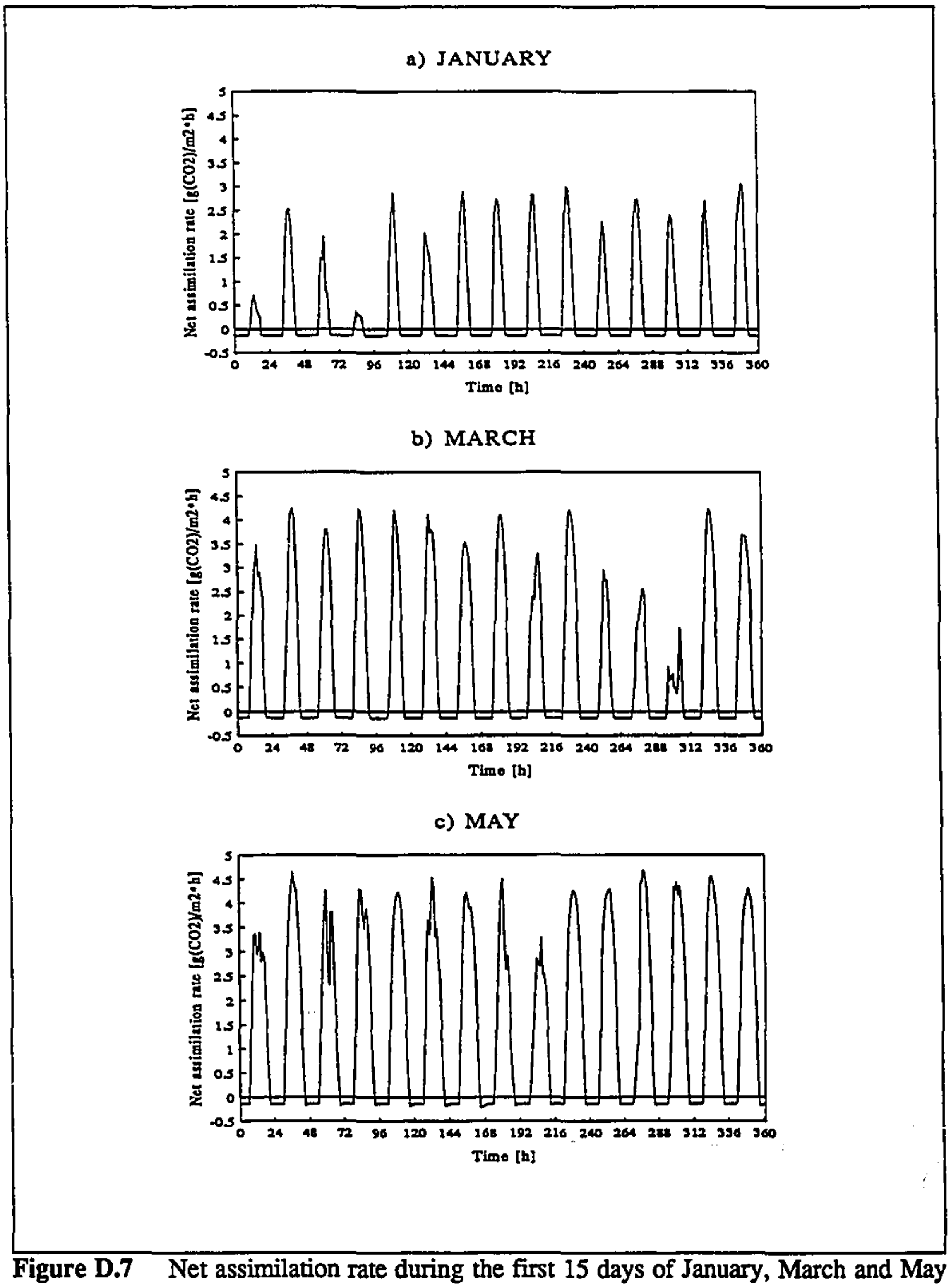

Figure D.7 Net assimilation rate during the first 15 days of January, March and May 1982.

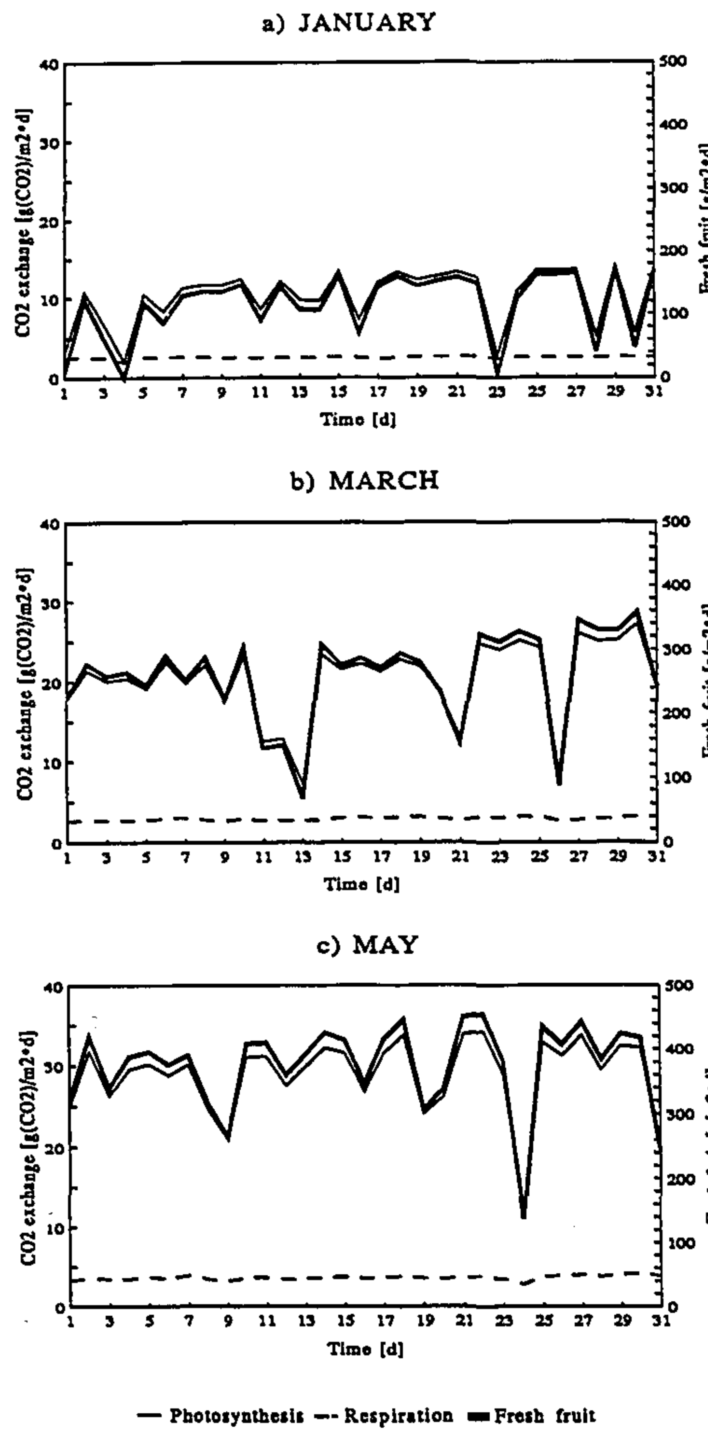

Figure D.8 Daily photosynthetic rates, respiration rates and fruit production rates during the months of January, March and May 1982.

Simulation results for each month of the year are shown in Table D.4. The monthly average leaf temperature was higher in summer than in winter by as much as 6.8°C (column 1). The total solar energy (PAR) reaching the crops is shown in column 2. If these values are compared with the solar radiation reaching the outside of the greenhouse (Table D.1), it can be computed that between 75 and 80% of the total solar radiation was transmitted by the cover. In columns 3 and 4 of the Table D.4, it can be seen that the maintenance respiration was as high as 25% of the gross photosynthesis during January; this fraction decreased to about 12% in May. The higher crop production in May compared to June was probably due to the higher total solar energy received during the May. The total fresh fruit production varied from 0.8 $kg/m^2 \cdot wk$ in January to 2.7 $kg/m^2 \cdot wk$ in July. The yearly average was 1.7 $kg/m^2 \cdot wk$. It is not easy to compare these results with those from other authors, because there is variability in so many factors like temperature, solar radiation intensity, cultivar and soil conditions. However, the levels of production that were obtained here correspond to values reported in the literature. For example, Cooper and Fuller (1983) mentioned yields of 0.71 $kg/m^2 \cdot wk$ during winter, while Hamel (1992) mentioned yields of 1.5 $kg/m^2 \cdot wk$ as a yearly average.

D.5 CONCLUSION

In general, it can be concluded that the models and the functional simulation furnished results that are reasonable and realistic. The functional system responded well to disturbances. However, some future improvements could certainly be done at many levels, principally in moisture balances and in the calculation of the infiltration rates and of the sky temperature. Some mechanisms for the regulation of the humidity should also be added to the control system, because of the influence of humidity on disease occurrence and nutrient assimilation. However, it is concluded that the functional system as it was conceived and constructed constitutes an excellent tool for the development of complex control systems.

Table D.4 Simulation results for crop.

	Leaf temperature [°C] (1)	PAR [MJ/m^2] (2)	Gross photosynthesis [g(CH$_2$O)/m^2·d] (3)	Maintenance respiration [g(CH$_2$O)/m^2·d] (4)	Fresh fruit [g/m^2·d] (5)
January	17.8	72.9	10.3	2.5	115.3
February	18.5	100.9	15.1	2.7	185.3
March	19.6	159.1	20.4	2.9	260.5
April	20.4	183.0	23.2	3.1	299.4
May	22.7	254.9	28.8	3.6	375.7
June	22.6	232.7	27.2	3.6	352.5
July	24.6	290.8	29.6	4.1	379.9
August	22.5	208.5	25.2	3.5	322.1
September	20.9	138.0	19.1	3.2	237.3
October	19.6	107.3	15.2	2.9	182.9
November	18.3	50.4	8.5	2.6	88.0
December	17.9	46.3	7.6	2.5	76.8

APPENDIX E. SHIFT OF THE HEATING PERIODS

The variation of the heat losses with time for a greenhouse is illustrated in Figure E.1. Periods 1 and 2 are of the same duration ($t_1 = t_2$) and the difference in temperature between the inside and outside is positive, constant and equal during both periods ($\Delta T_1 = \Delta T_2 = \Delta T$). The difference in heat loss is essentially due to a change in the value of the overall heat loss coefficient, which has a lower value during period 1 than period 2 ($U_1 < U_2$). This reflects the situation when a thermal screen is closed during period 1 and is opened during period 2. If heat losses during a period vary linearly with the difference in temperature between inside and outside, the heat loss rate for each period can be expressed by equations E.1 and E.2:

$$Q_1 = U_1 * \Delta T \tag{E.1}$$

$$Q_2 = U_2 * \Delta T \tag{E.2}$$

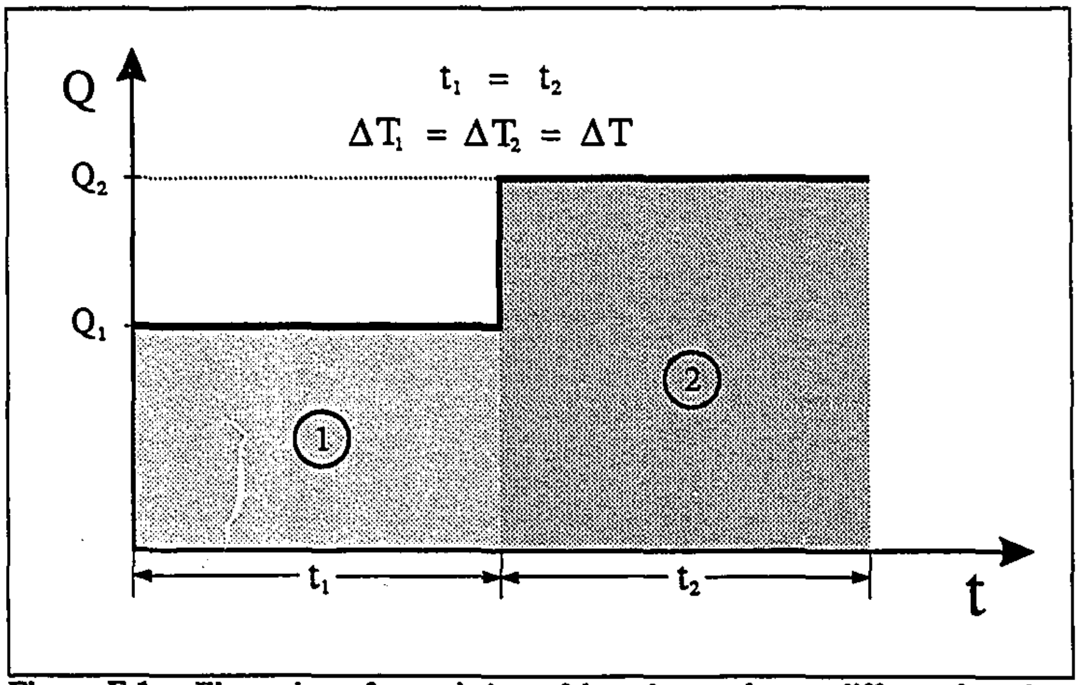

Figure E.1 Illustration of a variation of heat losses due to different heat loss coefficients.

where:

Q_1:	Heat loss rate during period 1	[W/m^2]
Q_2:	Heat loss rate during period 2	[W/m^2]
U_1:	Overall heat loss coefficient during period 1	[W/m^2·°C]
U_2:	Overall heat loss coefficient during period 2	[W/m^2·°C]
ΔT:	Difference of temperature between inside and outside	[°C]

Since periods 1 and 2 are of the same duration, the average heat loss rate for both periods can be calculated by equation E.3:

$$Q_a = \frac{(U_1 * \Delta T + U_2 * \Delta T)}{2} \tag{E.3}$$

where:

Q_a:	Average heat loss rate during both periods	[W/m^2]

If x is the ratio of the overall heat loss coefficients as in equation E.4:

$$x = \frac{U_2}{U_1} \tag{E.4}$$

then, by substituting equation E.4 into equation E.3, equations E.5 and E.6 are obtained:

$$Q_a = \frac{(U_1 * \Delta T) + (x * U_1 * \Delta T)}{2} \tag{E.5}$$

$$Q_a = \frac{(1 + x) * (U_1 * \Delta T)}{2} \tag{E.6}$$

Now, if the temperature of each of the two periods is adjusted by y°C, in opposite directions (i.e., increase by y°C the temperature of period 1 and decrease by y°C the temperature of period 2), the same average temperature as previously will be obtained.

Thus, the same temperature integral will be obtained. The new average heat loss rate (Q_b) will be:

$$Q_b = \frac{U_1 * (\Delta T + y) + U_2 * (\Delta T - y)}{2} \qquad \textbf{(E.7)}$$

Substituting equation E.4 into E.7 gives equations E.8 and E.9.

$$Q_b = \frac{U_1 * (\Delta T + y) + x * U_1 * (\Delta T - y)}{2} \qquad \textbf{(E.8)}$$

$$Q_b = \frac{(1 + x) * (U_1 * \Delta T) + (1 - x) * y * U_1}{2} \qquad \textbf{(E.9)}$$

The decrease of the heat loss rate when changing the inside temperature by y°C can be calculated as a percentage by equation E.10:

$$Q_{\%} = \frac{Q_a - Q_b}{Q_a} * 100 \qquad \textbf{(E.10)}$$

By substituting for Q_a and Q_b in equation E.10 with equations E.6 and E.9 respectively, equation E.11 is obtained:

$$Q_{\%} = \frac{(x - 1) * y}{(x + 1) * \Delta T} * 100 \qquad \textbf{(E.11)}$$

Equation E.11 indicates what percentage of energy can be saved when heating is shifted from a period where the heat loss coefficient is large towards a period where the heat loss coefficient is smaller. Several conclusions can be drawn from this equation. First, the percentage of energy saved varies with the ratio of the overall heat loss coefficients (x). Second, it increases with the magnitude of the temperature change y. Third, it decreases as the difference of temperature between inside and outside increases. This means that the lower the outside temperature is, the less will be the relative effect of shifting temperature setpoints in percentage. By subtracting equation E.9 from equation E.6, it could be observed that the difference of temperature between inside and outside does not affect the energy savings in absolute terms. It should be noted that equation E.11 is valid only when ΔT is equal in both periods initially and when the duration of both periods is the same.

The percentage of decrease of the average heat loss rate was calculated for several values of x, y and ΔT with equation E.11. The results are listed in Table E.1. The choice of the values for the x variable was justified by the values obtained in Appendix D. It was seen in Appendix D that the addition of an aluminized thermal screen to the functional greenhouse model reduced the overall heat loss coefficient by as much as 50%; such a value has also been reported in the literature. The application of this change in the heat loss coefficient to the situation illustrated in Figure E.1 corresponds to an x value of 2. It was also seen in Appendix B that, with the functional greenhouse model in which the infiltration rate was calculated with the model of Okada and Takakura (1973), the overall heat loss coefficient increased by as much as 80% when wind speed was increased from 0 m/s to 10 m/s. In the literature, values up to 135% were reported for the same conditions. Increases of 80% and 135% when passing from period 1 to 2 in Figure E.1 correspond to x values of 1.8 and 2.35 respectively.

It is shown in Table E.1 that, in the given configuration of the functional greenhouse model, a transfer of heating from day to night could possibly lead to a decrease of the energy consumption compared to a situation where the temperature would be kept the same for both periods. For example, if the temperature difference between the inside and outside were constant at 10°C and the inside temperature during the night were 4°C higher than during the day (y = 2°C), energy consumption might be reduced by 6.7%. If the difference of temperature between inside and outside was 20°C, this reduction would be 3.3%. Some reduction of the heating load could also be possible by transferring heating towards periods when the wind speed is low. For instance, a transfer of heating from a period of very high wind speed to a period of very low wind speed (x = 2.5), while maintaining an inside temperature difference of 6°C between the two periods (y = 3°C),

Table E.1 Percentage of decrease of the average heat loss for different values of *x*, *y* and ΔT.

x (1)	y [°C] (2)	ΔT [°C] (3)	$Q_{\%}$ [%] (4)	x (5)	y [°C] (6)	ΔT [°C] (7)	$Q_{\%}$ [%] (8)
1.5	1	10	2.0	1.5	1	20	1.0
2.0	1	10	3.3	2.0	1	20	1.7
2.5	1	10	4.3	2.5	1	20	2.1
1.5	2	10	4.0	1.5	2	20	2.0
2.0	2	10	6.7	2.0	2	20	3.3
2.5	2	10	8.6	2.5	2	20	4.3
1.5	3	10	6.0	1.5	3	20	3.0
2.0	3	10	10.0	2.0	3	20	5.0
2.5	3	10	12.9	2.5	3	20	6.4
1.5	4	10	8.0	1.5	4	20	4.0
2.0	4	10	13.3	2.0	4	20	6.7
2.5	4	10	17.1	2.5	4	20	8.6

could lead to a reduction in energy requirements of 12.9% for a ΔT of 10°C.

The values listed in Table E.1 indicate that shifting the heating to periods where heat loss coefficients are lower could lead to reductions in energy consumption. This seems to justify the development and the testing of control strategies based on this approach, to obtain a better indication of the potential in energy savings for greenhouses.

APPENDIX F. CALCULATION OF SCORE VALUES

[1]To judge the performance of a neural net its recall output must be compared to some testing values and this may be done in any of a number of ways. For example, the testing data and the neural net outputs for a given variable might be presented graphically and the comparison done visually by human judges. The detection of similarity (or difference) in pattern will then depend on the processing ability of the visual cortex. This approach, however, is qualitative and is practical only for a very limited amount of data. Moreover, it cannot be relied upon to be neutral, i.e., to repeatedly yield the same result for the same data, either with the same judge or with different judges. To compare the data from a substantial number of experiments it is preferable to rely on a numerical, quantitative method. In many ways such an approach is more reliable and more neutral although it is also less comprehensive.

A study was done to develop a neutral evaluation method to compare the recall output of a trained neural network with testing data. The intent was that the method should yield a single score for each variable, indicative of the extent of agreement between the two outputs. Furthermore this score should correspond in some direct manner to the visual judgment of humans and also, it would be preferable if the score were fixed in range between 0 and 100. In judging agreement between two data sets visually, at least two separate but interacting phenomena are taken into account by humans and these should be reflected in the evaluation method: a) similarity (or difference) in the magnitudes of the signals and b) in their shapes.

To calculate the score for two sets of values (each of length N) of an output variable the following is done: a) the testing data is called SET_A and the output from the neural network is called SET_B; b) SET_A is regarded as the reference set and its minimum value is found, LOW; c) the reference value LOW is then subtracted from each of the values in SET_A and SET_B to yield SET_C and SET_D:

[1] The following text has been extracted from Kok *et al.* (1993) and adapted to the present context.

$$SET_C(I) = SET_A(I) - LOW \qquad 1 \leq I \leq N \tag{1}$$

$$SET_D(I) = SET_B(I) - LOW \qquad 1 \leq I \leq N \tag{2}$$

and next; d) the root mean square value of SET_C is found:

$$RMS = \sqrt{\frac{\sum_{I=1}^{N} SET_C(I)^2}{N}} \tag{3}$$

which is used later for scaling and; e) using equation 4 below, both SET_C and SET_D are filtered to remove high frequency components, resulting in SET_E and SET_F, each containing (N - 4) values:

$$X(I) = X(I-2) + 2 * X(I-1) + 3 * X(I) + 2 * X(I+1) + X(I+2) \qquad 3 \leq I \leq N-2 \tag{4}$$

Then, f) the absolute differences between SET_E and SET_F are calculated:

$$SET_G(I) = ABS\ [SET_F(I) - SET_E(I)] \qquad 3 \leq I \leq N-2 \tag{5}$$

and; g) the root mean square value of the absolute differences is found with:

$$RMSDIFF = \sqrt{\frac{\sum_{I=3}^{N-2} SET_G(I)^2}{N-4}} \tag{6}$$

so that; h) the final score can be calculated with:

$$\text{SCORE} = 100 * \left[1 - \frac{\text{RMSDIFF}}{\text{RMS}}\right] \tag{7}$$

Except for extreme circumstances the method will always yield a number between 0 and 100. The magnitude of SCORE is directly related to the overall difference between the two data sets, a value of 100 indicating a perfect match between the two signals and a value of 0 meaning there is little or no likeness between them. The combination of formulas used in the method were the result of a large number of trials and were found to yield results which agreed well with the visual judgment of the authors.

APPENDIX G. COMPUTER PROGRAMS

G.1. File GHOUSE.PAR

This file contains the input data for the greenhouse model contained in the module GHOUSE

```
'************************************************************************
'*************************** GHOUSE.PAR ********************************

SOIL PARAMETERS
===============
CLAMDA  :  0.6      1.1      1.4      1.6
ELI     :  0.05     0.1      0.5      2.0
RHOI    : 1500.0   1700.00  1900.00  2100.00
CSMSI   :  900.0   1100.0   1300.0   1500.0
RHOSL   :  0.70
ABSSL   :  0.30
ESOL    :  0.30

GREENHOUSE PARAMETERS
=====================
LATIT.(d):  45.0
LONG SER :  96.00
LARG SER :  96.00
PIED DROI:  3.20
FAITE    :  3.90
LARG CHAP:  3.20

COVER PARAMETERS
================
ECOUV 11 :  0.90
ECOUV 12 :  0.90
TAUCOU   :  0.0
 89.0   88.0   86.0   84.0   74.0   37.0   00.0
 10.0   10.0   11.0   14.0   22.0   61.0   100.0
AB.CO.DIF:  15.0
TR.CO.DIF:  75.0
CS SUR CO: 8000.0
ORIENT;W :  0.0

SCREEN PARAMETERS
=================
EECR21   :  0.35
EECR22   :  0.35
TAUECR   :  0.0
CS SUR EC: 630.0
EPGAZ    :  0
ITOIEC   :  0
```

CONVECTIVE PARAMETERS

LAM-TURB :	0.63
VIT AIRIN:	0.60
V AIR GAZ:	0.60
TAUEG :	1.0

CROP PARAMETERS

PROP VEG :	0.8000
RHOVG :	0.15
TAUVG :	0.15
DV :	0.10
EVEG :	0.70
CSMVEG :	4180.00

G.2. File MANAGER5.BAS

This file contains the code for the module MANAGER

```
DECLARE SUB WriteShrDateTime (SelMemDateTime%, ActualDate$, ActualTime$)
DECLARE SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading, Reading())
DECLARE SUB OutputDate (DaySim#, ActualDate$)
DECLARE SUB OutputTime (HourSim%, MinuteSim%, SecondSim%, ActualTime$)

'           ********** MANAGER5.BAS **********

' This program executes WEATHER, GHOUSE, CROP and PAVLOV.

'$INCLUDE: 'E:\BC7\DATIM.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPE.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPC.BI'

CLS


'*******************************************************************************
'**************************** INITIALISATION *******************************
'*******************************************************************************

'* Description of variables and dimensioning *

DEFINT A-Z
DIM DayStart AS DOUBLE
DIM DayEnd AS DOUBLE
DIM DaySim AS DOUBLE
DIM LengthOfSim AS DOUBLE, TimeBegin AS DOUBLE, TimeEnd AS DOUBLE
DIM GhClimate(10) AS SINGLE, ActuatorVal(10) AS SINGLE
DIM CropState(10) AS SINGLE, Manager(10) AS SINGLE
DIM Reading(10) AS SINGLE
DIM SemHanMW1 AS LONG, SemHanMW2 AS LONG
DIM SemHanMG1 AS LONG, SemHanMG2 AS LONG
DIM SemHanMC1 AS LONG, SemHanMC2 AS LONG
DIM SemHanMP1 AS LONG, SemHanMP2 AS LONG
DIM WeatherResult AS RESULTCODES
DIM GhouseResult AS RESULTCODES
DIM CropResult AS RESULTCODES
DIM PavlovResult AS RESULTCODES
DIM ActualProcess AS PIDINFO
DIM SelMemDateTime AS INTEGER
DIM SelMemGlobal AS INTEGER
DIM SelMemWeather AS INTEGER
DIM SelMemGhouse AS INTEGER
DIM SelMemCrop AS INTEGER
DIM SelMemPavlov AS INTEGER

TimeBegin = TIMER

'* Definition of shared memory spaces

MemDateTime$ = "\SHAREMEM\DATETIME.MEM" + CHR$(0) 'Simulation time and date
MemGlobal$ = "\SHAREMEM\GLOBSIM.MEM" + CHR$(0)  'MANAGER output
MemWeather$ = "\SHAREMEM\WEATHER.MEM" + CHR$(0) 'WEATHER output
MemGhouse$ = "\SHAREMEM\GHOUSE.MEM" + CHR$(0)   'GHOUSE output
MemCrop$ = "\SHAREMEM\CROP.MEM" + CHR$(0)       'CROP output
MemPavlov$ = "\SHAREMEM\PAVLOV.MEM" + CHR$(0)   'PAVLOV output
```

```
'* Allocation of shared memory spaces

X = DosAllocShrSeg%(50, VARSEG(MemDateTime$), SADD(MemDateTime$), SelMemDateTime)
X = DosAllocShrSeg%(100, VARSEG(MemGlobal$), SADD(MemGlobal$), SelMemGlobal)
X = DosAllocShrSeg%(100, VARSEG(MemWeather$), SADD(MemWeather$), SelMemWeather)
X = DosAllocShrSeg%(200, VARSEG(MemGhouse$), SADD(MemGhouse$), SelMemGhouse)
X = DosAllocShrSeg%(100, VARSEG(MemCrop$), SADD(MemCrop$), SelMemCrop)
X = DosAllocShrSeg%(100, VARSEG(MemPavlov$), SADD(MemPavlov$), SelMemPavlov)


'* Definition of semaphores

SemMW1$ = "\SEM\SEMMW1" + CHR$(0)
SemMW2$ = "\SEM\SEMMW2" + CHR$(0)
SemMG1$ = "\SEM\SEMMG1" + CHR$(0)
SemMG2$ = "\SEM\SEMMG2" + CHR$(0)
SemMC1$ = "\SEM\SEMMC1" + CHR$(0)
SemMC2$ = "\SEM\SEMMC2" + CHR$(0)
SemMP1$ = "\SEM\SEMMP1" + CHR$(0)
SemMP2$ = "\SEM\SEMMP2" + CHR$(0)


'* Creation of semaphores

X = DosCreateSem%(1, SemHanMW1, VARSEG(SemMW1$), SADD(SemMW1$))
X = DosCreateSem%(1, SemHanMW2, VARSEG(SemMW2$), SADD(SemMW2$))
X = DosCreateSem%(1, SemHanMG1, VARSEG(SemMG1$), SADD(SemMG1$))
X = DosCreateSem%(1, SemHanMG2, VARSEG(SemMG2$), SADD(SemMG2$))
X = DosCreateSem%(1, SemHanMC1, VARSEG(SemMC1$), SADD(SemMC1$))
X = DosCreateSem%(1, SemHanMC2, VARSEG(SemMC2$), SADD(SemMC2$))
X = DosCreateSem%(1, SemHanMP1, VARSEG(SemMP1$), SADD(SemMP1$))
X = DosCreateSem%(1, SemHanMP2, VARSEG(SemMP2$), SADD(SemMP2$))


'* Define childs

ChildWeather$ = "G:\SIMULA\WEATHER5.EXE" + CHR$(0)
ChildGhouse$ = "G:\SIMULA\GHOUSE5.EXE" + CHR$(0)
ChildCrop$ = "G:\SIMULA\CROP5.EXE" + CHR$(0)
ChildPavlov$ = "G:\SIMULA\PAVLOV5.EXE" + CHR$(0)


'* Variable initialisation *

A$ = "0"
B$ = "00" + CHR$(0)


'* Read the simulation parameters *

OPEN "G:\simula\simul.par" FOR INPUT AS #1
INPUT #1, IniDate$, EndDate$, IniTime$, EndTime$
FOR I = 1 TO 3
        INPUT #1, Manager(I)
NEXT I
CLOSE #1
```

```
SecondStep = Manager(1)
FlagPrint = Manager(2)
LengthOfSleep = Manager(3)


'* Write in shared memory initial data for other processes

CALL WriteShrMem(SelMemGlobal, 2, Manager())
CALL WriteShrDateTime(SelMemDateTime, IniDate$, IniTime$ + "0000")


'** Calculate nth day for the first and last day of simulation *

IniYearOfSim = VAL(MID$(IniDate$, 1, 2))
EndYearOfSim = VAL(MID$(EndDate$, 1, 2))
IniMonthOfSim = VAL(MID$(IniDate$, 3, 2))
EndMonthOfSim = VAL(MID$(EndDate$, 3, 2))
IniDayOfSim = VAL(MID$(IniDate$, 5, 2))
EndDayOfSim = VAL(MID$(EndDate$, 5, 2))
IniHourOfSim = VAL(MID$(IniTime$, 1, 2))
EndHourOfSim = VAL(MID$(EndTime$, 1, 2))
IniMinuteOfSim = VAL(MID$(IniTime$, 3, 2))
EndMinuteOfSim = VAL(MID$(EndTime$, 3, 2))


' Based on hour varying from 0 to 23, and minute varying from 0 to 59 for
' time steps inferior or equal to 60 seconds. For larger time steps, minutes
' vary from 1 to 60. If the time step is 3600 seconds, hour varies from 1 to 24.
' Large time steps are allowed only for debugging. Time steps should be divi-
' dable by 60. This simplifies the procedures. For time step of 3600 seconds,
' start and end simulation at fixed hour (1:00 or 15:00).
' Combinations of start, end and timestep of simulation must always be
' configured to force passing through the fixed hours; if not, there will
' be a bug at the end of the first day.

DayStart = DateSerial#(IniYearOfSim, IniMonthOfSim, IniDayOfSim)
DayEnd = DateSerial#(EndYearOfSim, EndMonthOfSim, EndDayOfSim)
HourStart = IniHourOfSim
HourEnd = 23
IF SecondStep <= 60 THEN
        IF 60 MOD SecondStep <> 0 THEN
                PRINT "Error: time step must be a divider of 60": END
                END IF
        MinuteStart = IniMinuteOfSim
        MinuteEnd = 59
        MinuteStep = 1
        SecondStart = SecondStep
        SecondEnd = 60
        END IF
IF SecondStep > 60 THEN                'To be able to debug with large time step
        IF SecondStep MOD 60 <> 0 OR SecondStep > 3600 THEN
                PRINT "Error: verify input 'secondstep'"
                END IF
        MinuteStep = SecondStep / 60
        MinuteStart = IniMinuteOfSim + MinuteStep
        MinuteEnd = 60
        SecondStep = 60     'Can be anything <> 0; this forces simul. to pass
        SecondStart = 0     'through the seconds loop.
        SecondEnd = 0
        END IF
IF MinuteStep = 60 THEN
        HourEnd = 24
        HourStart = IniHourOfSim + 1
        MinuteStart = 0
        MinuteEnd = 0
        END IF


'* Execute initialisation phase of childs

X = DosSemSet%(SemHanMW1)
X = DosExecPgm%(0, 0, 0, 2, 0, 0, 0, 0, WeatherResult, VARSEG(ChildWeather$), SADD(ChildWeather$))
IF (X) THEN PRINT "Error in trying to execute WEATHER..."
X = DosSemWait%(SemHanMW1, -1)

X = DosSemSet%(SemHanMG1)
X = DosExecPgm%(0, 0, 0, 2, 0, 0, 0, 0, GhouseResult, VARSEG(ChildGhouse$), SADD(ChildGhouse$))
IF (X) THEN PRINT "Error in trying to execute GHOUSE..."
X = DosSemWait%(SemHanMG1, -1)

X = DosSemSet%(SemHanMC1)
X = DosExecPgm%(0, 0, 0, 2, 0, 0, 0, 0, CropResult, VARSEG(ChildCrop$), SADD(ChildCrop$))
IF (X) THEN PRINT "Error in trying to execute CROP..."
X = DosSemWait%(SemHanMC1, -1)

X = DosSemSet%(SemHanMP1)
X = DosExecPgm%(0, 0, 0, 2, 0, 0, 0, 0, PavlovResult, VARSEG(ChildPavlov$), SADD(ChildPavlov$))
IF (X) THEN PRINT "Error in trying to execute PAVLOV..."
X = DosSemWait%(SemHanMP1, -1)

'* Get actual process identification PID to be able to kill it with its children

X = DosGetPid%(ActualProcess)
IF (X) THEN PRINT "Error in getting PID"

LOCATE 1, 55: PRINT "(Press 's' to stop)"


'*****************************************************************************
'************************* BEGINNING OF SIMULATION *************************
'*****************************************************************************


'**** Loop for days ****

FOR DaySim = DayStart TO DayEnd
        CALL OutputDate(DaySim, ActualDate$)
        IF DaySim = DayEnd THEN HourEnd = EndHourOfSim


'  *** Loop for hours ***
```

```
        FOR HourSim = HourStart TO HourEnd

                IF FlagPrint = 0 THEN
                        LOCATE 1, 1: PRINT ActualDate$, HourSim
                END IF

                IF DaySim = DayEnd AND HourSim = EndHourOfSim THEN
                        IF MinuteStep < 60 THEN MinuteEnd = EndMinuteOfSim
                        IF MinuteStep = 1 THEN MinuteEnd = EndMinuteOfSim - MinuteStep
                END IF

'    ** Loop for minutes **

                FOR MinuteSim = MinuteStart TO MinuteEnd STEP MinuteStep

'       * Loop for seconds *
                        FOR SecondSim = SecondStart TO SecondEnd STEP SecondStep
                                CALL OutputTime(HourSim, MinuteSim, SecondSim, ActualTime$)

                                IF FlagPrint = 1 THEN
                                        LOCATE 1, 20
                                        PRINT ActualDate$, ActualTime$
                                END IF

                                '* Change the date and time in global shared memory

                                CALL WriteShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)

                                '* Execute one cycle for WEATHER by clearing a semaphore
                                X = DosSemSet%(SemHanMW1)
                                X = DosSemClear%(SemHanMW2)
                                X = DosSemWait%(SemHanMW1, -1)

                                '* Execute one cycle for GHOUSE by clearing a semaphore
                                X = DosSemSet%(SemHanMG1)
                                X = DosSemClear%(SemHanMG2)
                                X = DosSemWait%(SemHanMG1, -1)

                                '* Execute one cycle for CROP by clearing a semaphore
                                X = DosSemSet%(SemHanMC1)
                                X = DosSemClear%(SemHanMC2)
                                X = DosSemWait%(SemHanMC1, -1)

                                '* Execute one cycle for PAVLOV by clearing a semaphore
                                X = DosSemSet%(SemHanMP1)
                                X = DosSemClear%(SemHanMP2)
                                X = DosSemWait%(SemHanMP1, -1)

                                IF INKEY$ = "s" THEN GOTO EndOfSimulation
                                IF LengthOfSleep > 0 THEN SLEEP LengthOfSleep

                        NEXT SecondSim
'    * End of loop for seconds

                NEXT MinuteSim
'    ** End of loop for minutes
```

```
                IF MinuteStep > 1 AND MinuteStep < 60 THEN
                        MinuteStart = MinuteStep
                        ELSE MinuteStart = 0
                        END IF
        NEXT HourSim
' *** End of loop for hours

        IF MinuteStep = 60 THEN
                HourStart = 1
                ELSE HourStart = 0
                END IF
NEXT DaySim
'**** End of loop for days

'*****************************************************************************
'************************* END OF MAIN LOOP ******************************
'*****************************************************************************


EndOfSimulation:

CLS
PRINT : PRINT

X = DosSemSet%(SemHanMW1)
X = DosFlagProcess%(WeatherResult.codeTerminate, 0, 0, SigArg)
X = DosSemWait%(SemHanMW1, -1)
'X = DosKillProcess%(1, WeatherResult.codeTerminate)

X = DosSemSet%(SemHanMG1)
X = DosFlagProcess%(GhouseResult.codeTerminate, 0, 0, SigArg)
X = DosSemWait%(SemHanMG1, -1)
'X = DosKillProcess%(1, GhouseResult.codeTerminate)

X = DosSemSet%(SemHanMC1)
X = DosFlagProcess%(CropResult.codeTerminate, 0, 0, SigArg)
X = DosSemWait%(SemHanMC1, -1)
'X = DosKillProcess%(1, CropResult.codeTerminate)

X = DosSemSet%(SemHanMP1)
X = DosFlagProcess%(PavlovResult.codeTerminate, 0, 0, SigArg)
X = DosSemWait%(SemHanMP1, -1)
'X = DosKillProcess%(1, PavlovResult.codeTerminate)

X = DosFreeSeg%(SelMemDateTime)
X = DosFreeSeg%(SelMemGlobal)
X = DosFreeSeg%(SelMemWeather)
X = DosFreeSeg%(SelMemGhouse)
X = DosFreeSeg%(SelMemCrop)
X = DosFreeSeg%(SelMemPavlov)

TimeEnd = TIMER
IF TimeBegin > TimeEnd THEN
        TimeBegin = 86400 - TimeBegin
END IF
LengthOfSim = TimeEnd - TimeBegin
```

```
PRINT "Simulation time: "; FIX(LengthOfSim), " seconds  (assuming < 24 hrs...)"

END
'*****************************************************************************

'*****************************************************************************
SUB OutputDate (DaySim#, ActualDate$)

' Calculates ACTUAL DATE

YEAROUTPUT$ = MID$(STR$(year&(DaySim#)), 4, 2)
MONTHOUT = month&(DaySim#)
IF MONTHOUT < 10 THEN
        MONTHOUTPUT$ = "0" + MID$(STR$(MONTHOUT), 2, 1)
        ELSE MONTHOUTPUT$ = MID$(STR$(MONTHOUT), 2, 2)
        END IF
DayOFMONTH = day&(DaySim#)
IF DayOFMONTH < 10 THEN
        DayOUTPUT$ = "0" + MID$(STR$(DayOFMONTH), 2, 1)
        ELSE DayOUTPUT$ = MID$(STR$(DayOFMONTH), 2, 2)
        END IF
ActualDate$ = YEAROUTPUT$ + MONTHOUTPUT$ + DayOUTPUT$

END SUB
'*****************************************************************************

'*****************************************************************************
SUB OutputTime (HourSim, MinuteSim, SecondSim, ActualTime$)

' Calculates ACTUAL TIME

HourOUT = HourSim
MinuteOUT = MinuteSim
SECONDOUT = SecondSim

IF SECONDOUT = 60 THEN
        SECONDOUT = SECONDOUT - 60
        MinuteOUT = MinuteOUT + 1
        END IF
IF MinuteOUT = 60 THEN
        MinuteOUT = MinuteOUT - 60
        HourOUT = HourOUT + 1
        END IF

IF HourOUT < 10 THEN
                HourOUT$ = "0" + MID$(STR$(HourOUT), 2, 1)
                ELSE
                HourOUT$ = MID$(STR$(HourOUT), 2, 2)
                END IF
IF MinuteOUT < 10 THEN
                MinuteOUT$ = "0" + MID$(STR$(MinuteOUT), 2, 1)
                ELSE
                MinuteOUT$ = MID$(STR$(MinuteOUT), 2, 2)
                END IF
IF SECONDOUT < 10 THEN
                SECONDOUT$ = "0" + MID$(STR$(SECONDOUT), 2, 1)
```

```
                    ELSE
                    SECONDOUT$ = MID$(STR$(SECONDOUT), 2, 2)
                    END IF
ActualTime$ = HourOUT$ + ":" + MinuteOUT$ + ":" + SECONDOUT$

END SUB
'****************************************************************************

'****************************************************************************
SUB WriteShrDateTime (SelMemDateTime, ActualDate$, ActualTime$)

' Writes ACTUAL DATE and ACTUAL TIME in shared memory

DEF SEG = SelMemDateTime
FOR I = 1 TO 6
        POKE I, ASC(MID$(ActualDate$, I, 1))
NEXT I
FOR I = 1 TO 8
        POKE I + 6, ASC(MID$(ActualTime$, I, 1))
NEXT I
DEF SEG

END SUB
'****************************************************************************

'****************************************************************************
DEFSNG A-Z
SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading, Reading())

' Writes some data in shared memory

DIM SREADINGS$(NumOfReading)
DIM NUMOFCHR(NumOfReading + 1)

FOR I = 1 TO NumOfReading
        SREADINGS$(I) = STR$(Reading(I))
        NUMOFCHR(I) = LEN(SREADINGS$(I))
        IF NUMOFCHR(I) >= 255 THEN
                PRINT "ERROR in transferring data through shared memory..."
                END
        END IF
NEXT I

DEF SEG = SelMemName
FOR I = 1 TO NumOfReading
        POKE I, NUMOFCHR(I)
NEXT I

STARTWRITE = NumOfReading + 1
EndWRITE = NumOfReading + NUMOFCHR(1)
FOR I = 1 TO NumOfReading
        DEF SEG = SelMemName
        FOR J = STARTWRITE TO EndWRITE
                POKE J, ASC(MID$(SREADINGS$(I), J - STARTWRITE + 1, 1))
        NEXT J
        STARTWRITE = EndWRITE + 1
        EndWRITE = EndWRITE + NUMOFCHR(I + 1)
NEXT I
DEF SEG

END SUB
```

G.3. File WEATHER5.BAS

This file contains the code for the module WEATHER

```
DECLARE FUNCTION TSkyDogLem! (DryBulb!, VapPress!, CloudOpac!)
DECLARE FUNCTION SatPress! (Temp!)
DECLARE FUNCTION CalcTSky! (Temp!)
DECLARE SUB ReadShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)

'                ********** WEATHER5.BAS **********

'$INCLUDE: 'E:\BC7\BSEDOSPE.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPC.BI'

ON SIGNAL(5) GOSUB SignalADetected
SIGNAL(5) ON

'*****************************************************************************
'*************************** INITIALISATION **********************************
'*****************************************************************************

'* Description of variables and dimensioning *

DIM SemHanMW1 AS LONG
DIM SemHanMW2 AS LONG
DIM ValEnd(1 TO 8, 0 TO 24) AS SINGLE
DIM ValSlope(1 TO 8, 0 TO 23) AS SINGLE
DIM Weather(10) AS SINGLE, Manager(10) AS SINGLE
DIM SelMemGlobal AS INTEGER, SelMemWeather AS INTEGER
DIM SelMemDateTime AS INTEGER

'* Definition of shared memory spaces

MemDateTime$ = "\SHAREMEM\DATETIME.MEM" + CHR$(0)  'Simulation time and date
MemGlobal$ = "\SHAREMEM\GLOBSIM.MEM" + CHR$(0)     'Manager output
MemWeather$ = "\SHAREMEM\WEATHER.MEM" + CHR$(0)    'WEATHER output


'* Getting allocated shared memory spaces

X = DosGetShrSeg%(VARSEG(MemDateTime$), SADD(MemDateTime$), SelMemDateTime)
X = DosGetShrSeg%(VARSEG(MemGlobal$), SADD(MemGlobal$), SelMemGlobal)
X = DosGetShrSeg%(VARSEG(MemWeather$), SADD(MemWeather$), SelMemWeather)


'* Definition of semaphores

SemMW1$ = "\SEM\SEMMW1" + CHR$(0)
SemMW2$ = "\SEM\SEMMW2" + CHR$(0)

'* Opening the semaphores

X = DosOpenSem%(SemHanMW1, VARSEG(SemMW1$), SADD(SemMW1$))
X = DosOpenSem%(SemHanMW2, VARSEG(SemMW2$), SADD(SemMW2$))
```

```
'* Read initial values

CALL ReadShrMem(SelMemGlobal, 2, Manager())
SecondStep = Manager(1)
FlagPrint = Manager(2)

CALL ReadShrDateTime(SelMemDateTime, IniDate$, IniTime$)
IniDay = VAL(MID$(IniDate$, 5, 2))
YeMoSim$ = MID$(IniDate$, 1, 4)
MeteoFile$ = "J:\MTLDATA\MTLADJ\MTL" + YeMoSim$ + ".DAT"
OPEN MeteoFile$ FOR INPUT AS #1


'* Search for first day of simulation

IF IniDay > 1 THEN

        FOR I = 1 TO IniDay - 1
                LINE INPUT #1, A$
                FOR J = 1 TO 24
                        IF I = IniDay - 1 AND J = 24 THEN
                                INPUT #1, A
                                FOR K = 1 TO 7
                                        INPUT #1, ValEnd(K, 24)
                                NEXT K
                                ValEnd(3, 24) = ValEnd(3, 24)
                        ELSE
                                LINE INPUT #1, A$
                        END IF
                NEXT J
        NEXT I

        VapPress = ValEnd(2, 24) / 100 * SatPress(ValEnd(1, 24))
        ValEnd(8, 24) = TSkyDogLem(ValEnd(1, 24), VapPress, ValEnd(7, 24) / 10)

END IF

X = DosSemClear%(SemHanMW1)
X = DosSemSetWait%(SemHanMW2, -1)

'********************************************************************************
'************************** BEGINNING OF LOOP ******************************
'********************************************************************************

LOOPBEGIN:

'* Read date and time in global shared memory

CALL ReadShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)

DaySim = VAL(MID$(ActualDate$, 5, 2))
HourSim = VAL(MID$(ActualTime$, 1, 2))
MinuteSim = VAL(MID$(ActualTime$, 4, 2))
SecondSim = VAL(MID$(ActualTime$, 7, 2))
ActHour = HourSim + MinuteSim / 60 + SecondSim / 3600
```

```
'*** VERIFY IF DAY HAS CHANGED ***

IF DaySim <> PrecDay THEN        'Day has changed

        '* Verify that meteo. data file has to be changed
        PreYeMoSimS = YeMoSimS
        YeMoSimS = MID$(ActualDateS, 1, 4)
        IF YeMoSimS <> PreYeMoSimS THEN
                CLOSE #1
                MeteoFileS = "J:\MTLDATA\MTLADJ\MTL" + YeMoSimS + ".DAT"
                OPEN MeteoFileS FOR INPUT AS #1
        END IF

        '* Read new meteo. data for the actual day

        FOR I = 1 TO 8
                ValEnd(I, 0) = ValEnd(I, 24)
        NEXT I

        LINE INPUT #1, AS
        FOR J = 1 TO 24
                INPUT #1, A
                FOR I = 1 TO 7
                        INPUT #1, ValEnd(I, J)
                NEXT I
                ValEnd(3, J) = ValEnd(3, J)
                VapPress = ValEnd(2, J) / 100 * SatPress(ValEnd(1, J))
                ValEnd(8, J) = TSkyDogLem(ValEnd(1, J), VapPress, ValEnd(7, J) / 10)
        NEXT J

        '* Calculate the slopes

        FOR J = 0 TO 23
                FOR I = 1 TO 8
                        ValSlope(I, J) = (ValEnd(I, J + 1) - ValEnd(I, J)) / 3600
                NEXT I
        NEXT J

END IF

'**********************************

PrecDay = DaySim

'* Calculate instantaneous value

FOR I = 1 TO 8
        IF I < 5 OR I > 6 THEN
                Weather(I) = ValEnd(I, HourSim) + ValSlope(I, HourSim) * (MinuteSim * 60 + SecondSim)
        ELSE
                IF ActHour > 3 AND ActHour < 22 THEN
                        NetHour = FIX(ActHour - .5)
                        TimeDiff = (ActHour - (NetHour + .5)) * 3600
                        Weather(5) = ValEnd(5, NetHour + 1) + ValSlope(5, NetHour + 1) * TimeDiff
                        Weather(6) = ValEnd(6, NetHour + 1) + ValSlope(6, NetHour + 1) * TimeDiff
                        IF Weather(5) < 0 THEN Weather(5) = 0
                IF Weather(6) < 0 THEN Weather(6) = 0
                IF Weather(6) > Weather(5) THEN Weather(6) = Weather(5)
            ELSE
                Weather(5) = 0
                Weather(6) = 0
            END IF
        END IF
NEXT I

'* Print values on screen if required

IF FlagPrint = 1 THEN
LOCATE 3, 15: PRINT ActualDate$, ActualTime$
LOCATE 3, 45: PRINT "Temp. out :"; : PRINT USING "####.#"; Weather(1)
LOCATE 4, 15: PRINT "Rel. hum. :"; : PRINT USING "####.#"; Weather(2)
LOCATE 4, 35: PRINT "Wind (m/s):"; : PRINT USING "####.#"; Weather(3)
LOCATE 4, 55: PRINT "Glob. rad.:"; : PRINT USING "######"; Weather(5)
LOCATE 5, 15: PRINT "Diff. rad.:"; : PRINT USING "######"; Weather(6)
LOCATE 5, 35: PRINT "Sky Temp. :"; : PRINT USING "####.#"; Weather(8)
LOCATE 5, 55: PRINT "Cloud Op. :"; : PRINT USING "####.#"; Weather(7)
END IF

'* Write Weather values in shared memory

CALL WriteShrMem(SelMemWeather, 8, Weather())

'* Stop temporarily and continue MANAGER process

X = DosSemSet%(SemHanMW2)
X = DosSemClear%(SemHanMW1)
X = DosSemWait%(SemHanMW2, -1)

GOTO LOOPBEGIN

'****************************************************************************
'********************************* END OF LOOP ******************************
'****************************************************************************

END

'****************************************************************************
SignalADetected:

'* Subroutine executed when Signal A is detected

X = DosCloseSem%(SemHanMW2)

X = DosSemClear%(SemHanMW1)
X = DosCloseSem%(SemHanMW1)

END

RETURN
'****************************************************************************
```

```
'*****************************************************************************
FUNCTION CalcTSky (TempOut)

'*** Calculates Radiative sky temperature ***

CalcTSky = (.0552 * (TempOut + 273.15) ^ 1.5) - 273.15

END FUNCTION
'*****************************************************************************

'*****************************************************************************
SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)

DEF SEG = SelMemName
DateRead$ = ""
FOR I = 1 TO 6
        DateRead$ = DateRead$ + CHR$(PEEK(I))
NEXT I
TimeRead$ = ""
FOR I = 7 TO 14
        TimeRead$ = TimeRead$ + CHR$(PEEK(I))
NEXT I
DEF SEG

END SUB
'*****************************************************************************

'*****************************************************************************
SUB ReadShrMem (SelMemName AS INTEGER, NumOfReading, Reading())

DIM NUMOFCHR(NumOfReading + 1)

DEF SEG = SelMemName
FOR I = 1 TO NumOfReading
        NUMOFCHR(I) = PEEK(I)
NEXT I

STARTREAD = NumOfReading + 1
ENDREAD = NumOfReading + NUMOFCHR(1)
FOR I = 1 TO NumOfReading
        A$ = ""
        DEF SEG = SelMemName
        FOR J = STARTREAD TO ENDREAD
                A$ = A$ + CHR$(PEEK(J))
        NEXT J
        Reading(I) = VAL(A$)
        STARTREAD = ENDREAD + 1
        ENDREAD = ENDREAD + NUMOFCHR(I + 1)
NEXT I
DEF SEG

END SUB
'*****************************************************************************

'*****************************************************************************
FUNCTION SatPress (Temp)
```

```
' *SOUS-ROUTINE: CALCUL DE LA PRESSION A SATURATION*

' T = [C]
' SatPress = [Pa]


T = Temp + 273.16

IF T >= 273.16 THEN

        R# = 22105649.25#: A# = -27405.526#
        B# = 97.5413: C# = -.146244
        D# = .00012558#: E# = -.000000048502#
        F# = 4.34903: G# = .0039381#

        Z# = A# + B# * T + C# * T ^ 2 + D# * T ^ 3 + E# * T ^ 4
        Y# = F# * T - G# * T ^ 2
        X# = Z# / Y#:
        SatPress = R# * CDBL(EXP(X#))

        ELSE

        SatPress = EXP(31.9602 - 6270.3605# / T - .4605 * LOG(T))

END IF

END FUNCTION
'*******************************************************************************

'*******************************************************************************
FUNCTION TSkyDogLem (DryBulb, VapPress, CloudOpac)

EO = VapPress / 100
Intl = (.304 - .061 * EO ^ .5) * (1 - CloudOpac)

TSkyDogLem = ((DryBulb + 273.15) * (.904 - .005 * EO ^ .5 - Intl) ^ .25) - 273.15

END FUNCTION
```

G.4. File GHOUSE5.BAS

This file contains the code for the module GHOUSE

```
DECLARE SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)
DECLARE SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB ReadShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB SolDec (IMois!, IJour!, Mois!(), SolarDeclin!)
DECLARE SUB Verung (A!, A1!, B!, C!, D!, E!, F!, G!, H!, I!, J!, K!, L!, M!, N!, P!, BYVAL Q%, R!, S!, BYVAL
T%, BYVAL U%, V!, W!, X!)
DECLARE FUNCTION SatVap! (TSUR!)

'                ********** GHOUSES5.BAS **********


'$INCLUDE: 'E:\BC7\BSEDOSPE.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPC.BI'

DIM Heap%(2048)
COMMON SHARED /NMALLOC/ Heap%()

ON SIGNAL(5) GOSUB SignalADetected
SIGNAL(5) ON

'*****************************************************************************
'**************************** INITIALISATION ********************************
'*****************************************************************************

'* Description of variables and dimensioning *

DIM SemHanMG1 AS LONG
DIM SemHanMG2 AS LONG
DIM SelMemDateTime AS INTEGER
DIM SelMemGlobal AS INTEGER
DIM SelMemWeather AS INTEGER
DIM SelMemGhouse AS INTEGER
DIM SelMemCrop AS INTEGER
DIM SelMemPavlov AS INTEGER
DIM Manager(10) AS SINGLE
DIM Weather(10) AS SINGLE
DIM GhClimate(10) AS SINGLE
DIM ActuatorVal(10) AS SINGLE
DIM CropState(10) AS SINGLE

DIM FL(40), SumFL(40)
DIM Mois(12), Temp(10), SumTemp(10), AvgTemp(10), TCV(7)
DIM IDRAP(10) AS INTEGER


FOR I = 1 TO 12
        READ Mois(I)
NEXT I
DATA 0,31,28,31,30,31,30,31,31,30,31,30

CONST PI = 3.141592654#

CO2Out = 330
TotTime = 0
```

```
' Initialisation of greenhouse parameters

TempSS = 15!
AMassVeg = 6!
LAI = 3

QEMAX = 300!
TAUIGCL = 25
TAUIGOP = 500
InfiltRate = 0!
ThermalSc = 1

CALL Initial(SurfSol, VolSer)

FOR I = 1 TO 9
        READ Temp(I)
NEXT I
READ HII, HGI, CO2In
DATA 12.0, 13.0, 15.0, 17.0, 18.0, 18.0, 17.0, 16.0, 15.0
DATA 70.0, 80.0, 800

EGAZ = HGI / 100 * SatVap(Temp(2))
EI = HII / 100 * SatVap(Temp(4))

TCV(1) = Temp(1) - .01
TCV(2) = Temp(2) - .01
TCV(3) = Temp(3) - .01
TCV(4) = Temp(4) - .01
TCV(5) = Temp(5) - .01
TCV(6) = Temp(6) - .01
TCV(7) = Temp(1) - .01


'* Open files to write output data

OPEN "k:GHTEMP.DAT" FOR OUTPUT AS #3
OPEN "k:GHFLUX.DAT" FOR OUTPUT AS #10


'* Definition of shared memory spaces

MemDateTime$ = "\SHAREMEM\DATETIME.MEM" + CHR$(0)  'Simulation time and date
MemGlobal$ = "\SHAREMEM\GLOBSIM.MEM" + CHR$(0)      'MANAGER output
MemWeather$ = "\SHAREMEM\WEATHER.MEM" + CHR$(0)     'WEATHER output data
MemGhouse$ = "\SHAREMEM\GHOUSE.MEM" + CHR$(0)       'GHOUSE output data
MemCrop$ = "\SHAREMEM\CROP.MEM" + CHR$(0)           'CROP output data
MemPavlov$ = "\SHAREMEM\PAVLOV.MEM" + CHR$(0)       'PAVLOV output data

'* Allocation of shared memory spaces

X = DosGetShrSeg%(VARSEG(MemDateTime$), SADD(MemDateTime$), SelMemDateTime)
X = DosGetShrSeg%(VARSEG(MemGlobal$), SADD(MemGlobal$), SelMemGlobal)
X = DosGetShrSeg%(VARSEG(MemWeather$), SADD(MemWeather$), SelMemWeather)
X = DosGetShrSeg%(VARSEG(MemGhouse$), SADD(MemGhouse$), SelMemGhouse)
X = DosGetShrSeg%(VARSEG(MemCrop$), SADD(MemCrop$), SelMemCrop)
X = DosGetShrSeg%(VARSEG(MemPavlov$), SADD(MemPavlov$), SelMemPavlov)
```

```
'* Definition of semaphores

SemMG1$ = "\SEM\SEMMG1" + CHR$(0)
SemMG2$ = "\SEM\SEMMG2" + CHR$(0)

'* Opening the semaphores

X = DosOpenSem%(SemHanMG1, VARSEG(SemMG1$), SADD(SemMG1$))
X = DosOpenSem%(SemHanMG2, VARSEG(SemMG2$), SADD(SemMG2$))

'* Initial readings

CALL ReadShrMem(SelMemGlobal, 2, Manager())
SecondStep = Manager(1)
FlagPrint = Manager(2)


X = DosSemClear%(SemHanMG1)
X = DosSemSetWait%(SemHanMG2, -1)

'*****************************************************************************
'********************* BEGINNING OF MAIN LOOP ******************************
'*****************************************************************************

LOOPBEGIN:

PrecDate$ = ActualDate$

'* Read the date and time in global shared memory

CALL ReadShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)

HourSim = VAL(MID$(ActualTime$, 1, 2))
MinuteSim = VAL(MID$(ActualTime$, 4, 2))
SecondSim = VAL(MID$(ActualTime$, 7, 2))
ActTime = HourSim * 3600 + MinuteSim * 60 + SecondSim

IF PrecDate$ <> ActualDate$ THEN
        MonthSim = VAL(MID$(ActualDate$, 3, 2))
        DaySim = VAL(MID$(ActualDate$, 5, 2))
        CALL SolDec(MonthSim, DaySim, Mois(), SolarDeclin)
        CALL ReadShrMem(SelMemCrop, 1, CropState())
END IF

'* Read Weather data and Actuator state in shared memory

CALL ReadShrMem(SelMemWeather, 8, Weather())
TempOut = Weather(1): SumTempOut = SumTempOut + TempOut
RelHum = Weather(2): SumRelHum = SumRelHum + RelHum
WindVel = Weather(3): SumWindVel = SumWindVel + WindVel
GlobRad = Weather(5): SumGlobRad = SumGlobRad + GlobRad
DiffRad = Weather(6): SumDiffRad = SumDiffRad + DiffRad
TSky = Weather(8): SumTSky = SumTSky + TSky

EEX = RelHum / 100! * SatVap(TempOut)
```

```
CALL ReadShrMem(SelMemCrop, 1, CropState())
NetCO2Uptake = CropState(1)

CALL ReadShrMem(SelMemPavlov, 3, ActuatorVal())
FanJet = ActuatorVal(1)
Ventilator = ActuatorVal(2)
Heating = ActuatorVal(3)


'* Calculate flux induced by final control elements

IF FanJet = 1 THEN
          VentilRate = 3.5
          ELSE
          VentilRate = 0!
END IF
IF Ventilator = 1 THEN VentilRate = VentilRate + 22.5
IF Ventilator = 2 THEN VentilRate = VentilRate + 34
TAU = InfiltRate + VentilRate
SumVentil = SumVentil + VentilRate

IF Heating = 1 THEN
          FL(40) = QEMAX
          ELSE
          FL(40) = 0
END IF
IF (FL(40) = QEMAX) THEN ISumChauf = ISumChauf + 1


'* Regulation of thermal screen

IF GlobRad < 10 THEN
          'The screen is closed
          ThermalSc = 1
          TAUIG = TAUIGCL
ELSE
          'The screen is opened
          IF ThermalSc = 1 THEN OpenTime = ActTime
          ElapsTime = ActTime - OpenTime
          IF ElapsTime < 1800 THEN
                    TAUIG = TAUIGCL + (ElapsTime / 1800 * (TAUIGOP - TAUIGCL))
          ELSE
                    TAUIG = TAUIGOP
          END IF
          ThermalSc = 0
END IF


'***********     Calculate FLUX and STATE variables     **************

CALL Verung(SecondStep, ActTime, SolarDeclin, TempOut, EEX, WindVel, GlobRad, DiffRad, TSky, TempSS,
ThermalSc, TAUIG, TAU, LAI, AMassVeg, chlmp, VARPTR(Temp(1)), EGAZ, EI, VARPTR(TCV(1)),
VARPTR(FL(1)), HauSol, TrDirRad, TrDiffRad)

'**********************************************************************
```

```
RelHumIn = EI / SatVap(Temp(4)) * 100
HGAZ = EGAZ / SatVap(Temp(2)) * 100!
TempIn = Temp(4)
TempLeaf = Temp(5)
TrGlobRad = TrDirRad + TrDiffRad


'* Calculate CO2 balance

' To be added


'*****   Calculate Sums for statistics   *****

TotTime = TotTime + SecondStep

FOR I = 1 TO 9
        SumTemp(I) = SumTemp(I) + Temp(I)
NEXT I

SumEI = SumEI + EI
SumEGAZ = SumEGAZ + EGAZ
SumCO2In = SumCO2In + CO2In

FOR I = 1 TO 40
        SumFL(I) = SumFL(I) + FL(I)
NEXT I


'*****   Print values   *****

IF FlagPrint = 1 THEN
'LOCATE 11, 1: PRINT ActualDate$
'LOCATE 12, 1: PRINT ActualTime$
LOCATE 7, 15: PRINT "Temp. out:"; : PRINT USING "#####.#"; TempOut
LOCATE 7, 35: PRINT "Glob rad.:"; : PRINT USING "#######"; GlobRad
LOCATE 7, 55: PRINT "In. sol. :"; : PRINT USING "#######"; TrGlobRad
LOCATE 8, 15: PRINT "T. air   :"; : PRINT USING "#####.#"; TempIn
LOCATE 8, 35: PRINT "T. gaz   :"; : PRINT USING "#####.#"; Temp(2)
LOCATE 8, 55: PRINT "T. leaf  :"; : PRINT USING "#####.#"; TempLeaf
LOCATE 9, 15: PRINT "R.H. In. :"; : PRINT USING "#####.#"; RelHumIn
LOCATE 9, 35: PRINT "In. CO2  :"; : PRINT USING "#######"; CO2In
LOCATE 10, 15: PRINT "Ventil. :"; : PRINT USING "###.###"; VentilRate
LOCATE 10, 35: PRINT "Heat    :"; : PRINT USING "#######"; FL(40)
LOCATE 10, 55: PRINT "Th. scr.:"; : PRINT USING "#######"; ThermalSc
END IF

' Print values in shared memory

GhClimate(1) = TempIn
GhClimate(2) = RelHumIn
GhClimate(3) = CO2In
GhClimate(4) = TempLeaf
GhClimate(5) = TrDirRad
GhClimate(6) = TrDiffRad
GhClimate(7) = HauSol
```

```
CALL WriteShrMem(SelMemGhouse, 7, GhClimate())

'* Ecriture des resultats horaires (ou moins)

IF TotTime >= 3600 THEN

NumOfCount = TotTime / SecondStep: TotTime = 0

AvgChauf = QEMAX * ISumChauf / NumOfCount: ISumChauf = 0
AvgVentil = SumVentil / NumOfCount: SumVentil = 0!

FOR I = 1 TO 9: AvgTemp(I) = SumTemp(I) / NumOfCount: NEXT I
FOR I = 1 TO 9: SumTemp(I) = 0: NEXT I

AvgEI = SumEI / NumOfCount: SumEI = 0
AvgEGAZ = SumEGAZ / NumOfCount: SumEGAZ = 0
AvgRelHumIN = AvgEI / SatVap(AvgTemp(4)) * 100
AvgHGAZ = AvgEGAZ / SatVap(AvgTemp(2)) * 100!

AvgCO2In = SumCO2In / NumOfCount: SumCO2In = 0

AvgTempOut = SumTempOut / NumOfCount: SumTempOut = 0
AvgRelHum = SumRelHum / NumOfCount: SumRelHum = 0
AvgWindVel = SumWindVel / NumOfCount: SumWindVel = 0
AvgGlobRad = SumGlobRad / NumOfCount: SumGlobRad = 0
AvgDiffRad = SumDiffRad / NumOfCount: SumDiffRad = 0
AvgTSky = SumTSky / NumOfCount: SumTSky = 0

PRINT #3, USING "#######.##"; MonthSim; DaySim; ActTime / 3600;
PRINT #3, USING "#######.##"; AvgTempOut; AvgRelHum; AvgWindVel; AvgGlobRad;
PRINT #3, USING "#######.##"; AvgDiffRad; AvgTSky; TempSS; AvgRelHumIN; AvgHGAZ;
FOR I = 1 TO 9: PRINT #3, USING "#######.##"; AvgTemp(I); : NEXT I
PRINT #3, USING "##########"; AvgCO2In;
PRINT #3, USING "#######.##"; AvgChauf; AvgVentil

PRINT #10, USING " ###.#"; MonthSim; DaySim; ActTime / 3600
FOR I = 1 TO 40
                SumFL(I) = SumFL(I) / NumOfCount
                PRINT #10, USING "#######.##"; SumFL(I);
NEXT I
PRINT #10,
FOR I = 1 TO 40: SumFL(I) = 0: NEXT I

END IF


X = DosSemSet%(SemHanMG2)
X = DosSemClear%(SemHanMG1)
X = DosSemWait%(SemHanMG2, -1)

GOTO LOOPBEGIN
'*****************************************************************************
'**************************** END OF MAIN LOOP ******************************
'*****************************************************************************
```

```
ENDOFSIMULATION:

END


SignalADetected:

'Subroutine executed when Signal A is detected

CLOSE #3
CLOSE #10

X = DosSemClear%(SemHanMG1)
X = DosCloseSem%(SemHanMG1)

X = DosCloseSem%(SemHanMG2)

END

RETURN
'*****************************************************************************

'*****************************************************************************
SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)

DEF SEG = SelMemName
DateRead$ = ""
FOR I = 1 TO 6
        DateRead$ = DateRead$ + CHR$(PEEK(I))
NEXT I
TimeRead$ = ""
FOR I = 7 TO 14
        TimeRead$ = TimeRead$ + CHR$(PEEK(I))
NEXT I
DEF SEG

END SUB
'*****************************************************************************

'*****************************************************************************
FUNCTION SatVap (TSUR)

'***********************************************************************
' SatVap
' ==========================================================================
' SatVap  : FONCTION PERMETTANT LE CALCUL DE LA TENEUR EN EAU A SATUR
'           PARTIR DE LA TEMPERATURE
'***********************************************************************

AA = 17.2693882#
BB = 237.3
IF TSUR > 0! GOTO 800

AA = 21.8745584#
BB = 265.5
```

```
800 : SatVap = 1.323 * EXP((AA * TSUR) / (BB + TSUR)) / (273.16 + TSUR)

END FUNCTION
'******************************************************************************

'******************************************************************************
SUB SolDec (IMois, IJour, Mois(), SolarDeclin)

ISMois = 0
FOR KLM = 1 TO IMois
        ISMois = ISMois + Mois(KLM)
NEXT KLM

DayNumber = ISMois + IJour
DayAngle = 2! * PI * DayNumber / 365!

SolarDeclin = .006918 - .399912 * COS(DayAngle) + .070257 * SIN(DayAngle)
SolarDeclin = SolarDeclin - .006759 * COS(2! * DayAngle) + .000907 * SIN(2! * DayAngle)
SolarDeclin = SolarDeclin - .002697 * COS(3! * DayAngle) + .00148 * SIN(3! * DayAngle)


END SUB
```

G.5. File CROP5.BAS

This file contains the code for the module CROP

```
DECLARE SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)
DECLARE SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB ReadShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB LPHCUR (TempLeaf, CO2In, EFF, FGMAX)
DECLARE SUB ASSIM (SCV, FGMAX, EFF, KDIF, LAI, SINB, PARDIR, PARDIF, FGROS)

'                 ********** CROP5.BAS **********


'$INCLUDE: 'E:\BC7\BSEDOSPE.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPC.BI'

DIM Heap%(2048)
COMMON SHARED /NMALLOC/ Heap%()

ON SIGNAL(5) GOSUB SignalADetected
SIGNAL(5) ON

'*****************************************************************************
'************************* INITIALISATION ************************************
'*****************************************************************************

'* Description of variables and dimensioning *

DIM SemHanMC1 AS LONG
DIM SemHanMC2 AS LONG
DIM SelMemDateTime AS INTEGER
DIM SelMemGlobal AS INTEGER
DIM SelMemWeather AS INTEGER
DIM SelMemGhouse AS INTEGER
DIM SelMemCrop AS INTEGER
DIM SelMemPavlov AS INTEGER
DIM Manager(10) AS SINGLE
DIM Weather(10) AS SINGLE
DIM GhClimate(10) AS SINGLE
DIM ActuatorVal(10) AS SINGLE
DIM CropState(10) AS SINGLE


'* Initialisation of variables

AvgTempLeaf = 20
DTGA = 0
MAINTS = 4.084
PI = 3.141593


'*********           Crop parameters              *******

'* Diffuse light extinction coefficient  (0.8 for horizontal leaves)
                    CONST KDIF = .8

'* Scattering coefficient of leaves for PAR (400 - 700 nm)
                    CONST SCV = .2

'* Leaf Area Index
```

```
                    CONST LAI = 3!

'* Percentage dry weight
                    CONST PERCDW = 3.5

'* Maintenance coefficients
                    CONST MAINLV = .03
                    CONST MAINST = .015
                    CONST MAINSO = .01
                    CONST MAINRT = .01
                    CONST REFTMP = 25!
                    CONST Q10 = 2!

'* Assimilate requirements of organs [ g CH2O/g dry matter of organ ]
                    CONST ASRQSO = 1.37
                    CONST ASRQLV = 1.39
                    CONST ASRQST = 1.45
                    CONST ASRQRT = 1.39

'* Dry matter partitioning
                    CONST FLV = .17
                    CONST FST = .08
                    CONST FSO = .72
                    CONST FRT = .03

'* Dry weights of leaves, stems, roots and storage organs [g m-2]
'* To obtain 6 kg total fresh weight
                    CONST WLV = 94.5
                    CONST WRT = 18.9
                    CONST WST = 44.1
                    CONST WSO = 52.5


'* Initialisation of outputfile

OPEN "k:CRHOUR.DAT" FOR OUTPUT AS #1
OPEN "k:CRDAY.DAT" FOR OUTPUT AS #2


'* Definition of shared memory spaces

MemDateTime$ = "\SHAREMEM\DATETIME.MEM" + CHR$(0)  'Simulation time and date
MemGlobal$ = "\SHAREMEM\GLOBSIM.MEM" + CHR$(0)      'MANAGER output
MemWeather$ = "\SHAREMEM\WEATHER.MEM" + CHR$(0)     'WEATHER output data
MemGhouse$ = "\SHAREMEM\GHOUSE.MEM" + CHR$(0)       'GHOUSE output data
MemCrop$ = "\SHAREMEM\CROP.MEM" + CHR$(0)           'CROP output data
MemPavlov$ = "\SHAREMEM\PAVLOV.MEM" + CHR$(0)       'PAVLOV output data

'* Allocation of shared memory spaces

X = DosGetShrSeg%(VARSEG(MemDateTime$), SADD(MemDateTime$), SelMemDateTime)
X = DosGetShrSeg%(VARSEG(MemGlobal$), SADD(MemGlobal$), SelMemGlobal)
X = DosGetShrSeg%(VARSEG(MemWeather$), SADD(MemWeather$), SelMemWeather)
X = DosGetShrSeg%(VARSEG(MemGhouse$), SADD(MemGhouse$), SelMemGhouse)
X = DosGetShrSeg%(VARSEG(MemCrop$), SADD(MemCrop$), SelMemCrop)
X = DosGetShrSeg%(VARSEG(MemPavlov$), SADD(MemPavlov$), SelMemPavlov)
```

```
'* Definition of semaphores

SemMC1$ = "\SEM\SEMMC1" + CHR$(0)
SemMC2$ = "\SEM\SEMMC2" + CHR$(0)

'* Opening the semaphores

X = DosOpenSem%(SemHanMC1, VARSEG(SemMC1$), SADD(SemMC1$))
X = DosOpenSem%(SemHanMC2, VARSEG(SemMC2$), SADD(SemMC2$))


'* Initial writings and readings

CALL ReadShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)
MonthSim = VAL(MID$(ActualDate$, 3, 2))
DaySim = VAL(MID$(ActualDate$, 5, 2))

CALL WriteShrMem(SelMemCrop, 1, CropState())

CALL ReadShrMem(SelMemGlobal, 2, Manager())
SecondStep = Manager(1)
FlagPrint = Manager(2)

NumOfCount = 3600 / SecondStep

X = DosSemClear%(SemHanMC1)
X = DosSemSetWait%(SemHanMC2, -1)


'*****************************************************************************
'*********************** BEGINNING OF MAIN LOOP *****************************
'*****************************************************************************

LOOPBEGIN:

'* Read values in shared memory spaces

PrecDate$ = ActualDate$

CALL ReadShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)

HourSim = VAL(MID$(ActualTime$, 1, 2))
MinuteSim = VAL(MID$(ActualTime$, 4, 2))
SecondSim = VAL(MID$(ActualTime$, 7, 2))
ActTime = HourSim * 3600 + MinuteSim * 60 + SecondSim

IF ActualDate$ <> PrecDate$ THEN
        MonthSim = VAL(MID$(ActualDate$, 3, 2))
        DaySim = VAL(MID$(ActualDate$, 5, 2))
        GOSUB DailyCalculations
END IF

CALL ReadShrMem(SelMemGhouse, 7, GhClimate())
TempIn = GhClimate(1)
CO2In = GhClimate(3)
TempLeaf = GhClimate(4)
```

```
PARDIR = .5 * GhClimate(5)
PARDIF = .5 * GhClimate(6)
HauSol = GhClimate(7)

IF PARDIR = 0 AND PARDIF = 0 THEN
        FGROS = 0!
        GOTO NoPhotosynthesis
END IF


'* Calculate crop growth with Sucros model

'*********************************************************************************
'****************** CALCULATE CROP GROWTH WITH SUCROS MODEL *************

'*------ Photosynthesis

' SCV     Scattering coefficient of leaves for visible         *
'         radiation (PAR)                       -      I   *
' FGMAX   Assimilation rate at light saturation    mg CO2/  I   *
'                                          m2 leaf/s      *
' EFF     Initial light use efficiency          mg CO2/J  I   *
' KDIF    Extinction coefficient for diffuse light          I   *
' LAI     Leaf area index                       m2/m2   I   *
' SINB    Sine of solar height                      -    I   *
' PARDIR  Instantaneous flux of direct radiation (PAR) W/m2   I   *
' PARDIF  Instantaneous flux of diffuse radiation(PAR) W/m2   I   *
'       -> seems to be on a horizontal surface (Rene Lacroix)
' FGROS   Instantaneous assimilation rate of        mg CO2/  O   *
'         whole canopy                         m2 soil/s     *

' TempLeaf:    leaf temperature             [oC]
' CO2In  :     CO2-concentration            [vpm]
' EFF    :    initial light use efficiency  [mg CO2 J-1]
' FGMAX  :      gross assimilation at light saturation  [mg CO2 m-2 s-1]


'*------ Determine EFF and FGMAX value of leaf photosynthesis-light response
'*       curve from leaf temperature and CO2 concentration

CALL LPHCUR(TempLeaf, CO2In, EFF, FGMAX)

'*------Instantaneous assimilation

IF HauSol < 0 THEN
        BEEP: PRINT "Error in solar height (CROP module)...": END
END IF

SINB = SIN(HauSol)

CALL ASSIM(SCV, FGMAX, EFF, KDIF, LAI, SINB, PARDIR, PARDIF, FGROS)

'*-------Integration of assimilation rate to a daily total (DTGA)
DTGA = DTGA + FGROS * SecondStep

'*-------Instantaneous respiration rate (InstRespRate) in mg CO2/m2/s
```

```
NoPhotosynthesis:

InstMAINTS = MAINTS / 86400 * 1000    'MAINTS is calculated in g CH2O/m2/day
InstTeff = Q10 ^ (((TempLeaf - REFTMP) / 10!))
InstRespRate = InstMAINTS * InstTeff * 44 / 30

'*------Calculate net CO2 uptake in mg CO2/m2/s
NetCO2Uptake = FGROS - InstRespRate

'*------Prepare for statistics

SumFGROS = SumFGROS + FGROS
SumInstRespRate = SumInstRespRate + InstRespRate
SumTrPARRad = SumTrPARRad + PARDIR + PARDIF
SumTempLeafD = SumTempLeafD + TempLeaf
SumTempLeafH = SumTempLeafH + TempLeaf
SumCO2In = SumCO2In + CO2In
Counter = Counter + 1

'****************************************************************************


'* Ecriture des resultats horaires

IF Counter = NumOfCount THEN

AvgFGROS = SumFGROS / Counter
AvgInstRespRate = SumInstRespRate / Counter
AvgTrPARRad = SumTrPARRad / Counter
AvgTempLeafH = SumTempLeafH / Counter
AvgCO2In = SumCO2In / Counter
CumCarbon = CumCarbon + (AvgFGROS - AvgInstRespRate) * 3.6

PRINT #1, USING "####"; MonthSim; DaySim;
PRINT #1, USING " #####.#"; ActTime / 3600; AvgTempLeafH; AvgTrPARRad; AvgCO2In;
PRINT #1, USING "    ##.##^^^^"; AvgFGROS; AvgInstRespRate; CumCarbon

Counter = 0
SumFGROS = 0
SumInstRespRate = 0
SumTrPARRad = 0
SumTempLeafH = 0
SumCO2In = 0

END IF


'* Print on screen is required

IF FlagPrint = 1 THEN
LOCATE 20, 20: PRINT ActualDate$, ActualTime$
LOCATE 21, 15: PRINT "Leaf temp.:"; : PRINT USING "####.#"; TempLeaf
LOCATE 21, 40: PRINT "PAR Dir:"; : PRINT USING "#####.#"; PARDIR
LOCATE 21, 60: PRINT "PAR Dif:"; : PRINT USING "#####.#"; PARDIF
LOCATE 22, 15: PRINT "Phot.[mg/m2/s]: "; : PRINT USING "##.###^^^^"; FGROS
LOCATE 22, 45: PRINT "Resp.[mg/m2/s]: "; : PRINT USING "##.###^^^^"; InstRespRate
LOCATE 23, 15: PRINT "Net CO2 Uptake [mg/m2/s]: "; : PRINT USING "##.##^^^^"; NetCO2Uptake
LOCATE 23, 55: PRINT "Sol. h.: "; : PRINT USING "###.#"; HauSol * 180 / PI
'LOCATE 23, 45: PRINT USING "###.##^^^^"; InstMAINTS; InstTeff
END IF


'* Print values in shared memory

CropState(1) = NetCO2Uptake
CALL WriteShrMem(SelMemCrop, 1, CropState())


X = DosSemSet%(SemHanMC2)
X = DosSemClear%(SemHanMC1)
X = DosSemWait%(SemHanMC2, -1)

GOTO LOOPBEGIN
'**************************** End of loop **************************************

ENDOFSIMULATION:

END


DailyCalculations:
' ****************** Calculations for DAILY values ***************************

AvgTempLeaf = SumTempLeafD * SecondStep / 86400

'*------ DTGA    Daily total gross assimilation [mg CO2/m2/d]
'*------ GPHOT   Gross photosynthesis [g CH2O m-2 day-1]

GPHOT = DTGA * 30! / 44! / 1000

'*------ Maintenance respiration [g CH2O m-2 day-1]

MAINTS = WLV * MAINLV + WST * MAINST + WSO * MAINSO + WRT * MAINRT
TEFF = Q10 ^ (((AvgTempLeaf - REFTMP) / 10!))
PRODMAINTEFF = MAINTS * TEFF
IF GPHOT < PRODMAINTEFF THEN
        MAINT = GPHOT
        ELSE
        MAINT = PRODMAINTEFF
END IF

'*------ Assimilate requirements for dry matter conversion
'*------ [g CH2O/g dry matter]

ASRQ = FLV * ASRQLV + FST * ASRQST + FSO * ASRQSO + FRT * ASRQRT

'*------Rate of growth  [g d.w. m-2 day-1]

GTW = (GPHOT - MAINT) / ASRQ
GLV = GTW * FLV
GST = GTW * FST
GSO = GTW * FSO
```

```
GRT = GTW * FRT

'*----- Fresh weight of fruits [g m-2]
'*----- PERCDW = Percentage dry weight of fruits

FGSO = GSO * (100! / PERCDW)

PRINT #2, USING "  ###"; VAL(MID$(PrecDate$, 3, 2)); VAL(MID$(PrecDate$, 5, 2));
PRINT #2, USING "#####.##"; AvgTempLeaf; GPHOT; MAINT; GTW; GSO; FGSO

'*------ Reset values to 0

DTGA = 0
SumTempLeafD = 0

RETURN
'*******************************************************************************

SignalADetected:

'* Subroutine executed when Signal A is detected

GOSUB DailyCalculations

CLOSE #1
CLOSE #2

X = DosCloseSem%(SemHanMC2)

X = DosSemClear%(SemHanMC1)
X = DosCloseSem%(SemHanMC1)

END

RETURN
'*******************************************************************************

'*******************************************************************************
SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)

DEF SEG = SelMemName
DateRead$ = ""
FOR I = 1 TO 6
        DateRead$ = DateRead$ + CHR$(PEEK(I))
NEXT I
TimeRead$ = ""
FOR I = 7 TO 14
        TimeRead$ = TimeRead$ + CHR$(PEEK(I))
NEXT I
DEF SEG

END SUB
```

G.6. File PAVLOV5.BAS

This file contains the code for the module PAVLOV

```
DECLARE SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)
DECLARE SUB WriteShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())
DECLARE SUB ReadShrMem (SelMemName AS INTEGER, NumOfReading!, Reading!())

'               ********** PAVLOV5.BAS **********


'$INCLUDE: 'E:\BC7\BSEDOSPE.BI'
'$INCLUDE: 'E:\BC7\BSEDOSPC.BI'


ON SIGNAL(5) GOSUB SignalADetected
SIGNAL(5) ON


'********************************************************************************
'************************** INITIALISATION ***********************************
'********************************************************************************

'* Description of variables and dimensioning *

DIM SemHanMP1 AS LONG, SemHanMP2 AS LONG
DIM SemHanPC1 AS LONG, SemHanPC2 AS LONG

DIM SelMemDateTime AS INTEGER
DIM SelMemGlobal AS INTEGER
DIM SelMemWeather AS INTEGER
DIM SelMemGhouse AS INTEGER
DIM SelMemCrop AS INTEGER
DIM SelMemPavlov AS INTEGER

DIM Manager(10) AS SINGLE
DIM Weather(10) AS SINGLE
DIM GhClimate(10) AS SINGLE
DIM ActuatorVal(10) AS SINGLE
DIM CropState(10) AS SINGLE

DIM GURUResult AS RESULTCODES
DIM ActualProcess AS PIDINFO


'* Initialisation of variables

TempIn = 20
HeatSpt = 20
VentSpt = 25


'* Open file to output data

OPEN "k:PAVLOV.DAT" FOR OUTPUT AS #1


'* Definition of shared memory spaces

MemDateTime$ = "\SHAREMEM\DATETIME.MEM" + CHR$(0)  'Simulation time and date
```

```
MemGlobal$ = "\SHAREMEM\GLOBSIM.MEM" + CHR$(0)      'MANAGER output
MemWeather$ = "\SHAREMEM\WEATHER.MEM" + CHR$(0)    'GHOUSE-PAVLOV data ex.
MemGhouse$ = "\SHAREMEM\GHOUSE.MEM" + CHR$(0)      'GHOUSE output data
MemCrop$ = "\SHAREMEM\CROP.MEM" + CHR$(0)          'CROP output data
MemPavlov$ = "\SHAREMEM\PAVLOV.MEM" + CHR$(0)      'PAVLOV output data


'* Allocation of shared memory spaces

X = DosGetShrSeg%(VARSEG(MemDateTime$), SADD(MemDateTime$), SelMemDateTime)
X = DosGetShrSeg%(VARSEG(MemGlobal$), SADD(MemGlobal$), SelMemGlobal)
X = DosGetShrSeg%(VARSEG(MemWeather$), SADD(MemWeather$), SelMemWeather)
X = DosGetShrSeg%(VARSEG(MemGhouse$), SADD(MemGhouse$), SelMemGhouse)
X = DosGetShrSeg%(VARSEG(MemCrop$), SADD(MemCrop$), SelMemCrop)
X = DosGetShrSeg%(VARSEG(MemPavlov$), SADD(MemPavlov$), SelMemPavlov)


'* Definition of semaphores

SemMP1$ = "\SEM\SEMMP1" + CHR$(0)
SemMP2$ = "\SEM\SEMMP2" + CHR$(0)
SemPC1$ = "\SEM\SEMPC1" + CHR$(0)
SemPC2$ = "\SEM\SEMPC2" + CHR$(0)


'* Opening the semaphores

X = DosOpenSem%(SemHanMP1, VARSEG(SemMP1$), SADD(SemMP1$))
X = DosOpenSem%(SemHanMP2, VARSEG(SemMP2$), SADD(SemMP2$))
X = DosCreateSem%(1, SemHanPC1, VARSEG(SemPC1$), SADD(SemPC1$))
X = DosCreateSem%(1, SemHanPC2, VARSEG(SemPC2$), SADD(SemPC2$))


'* Initial readings

CALL ReadShrMem(SelMemGlobal, 2, Manager())
SecondStep = Manager(1)
FlagPrint = Manager(2)

CALL ReadShrDateTime(SelMemDateTime, IniDate$, IniTime$)
IniDay = VAL(MID$(IniDate$, 5, 2))
YeMoSim$ = MID$(IniDate$, 1, 4)
HourIniTime = VAL(MID$(IniTime$, 1, 2)) + VAL(MID$(IniTime$, 4, 2)) / 60


'*** Execute initialisation phase of COGNITI ***

COGNITIIN = FREEFILE
OPEN "k:COGNITI.IN" FOR OUTPUT AS #COGNITIIN
PRINT #COGNITIIN, IniDate$
PRINT #COGNITIIN, HourIniTime
PRINT #COGNITIIN, AvgTempOut
PRINT #COGNITIIN, AvgWindVel
PRINT #COGNITIIN, AvgGlobRad
PRINT #COGNITIIN, AvgTempIn
CLOSE #COGNITIIN
```

```
X = DosSemSet%(SemHanPC1)

SHELL "copy cognctrl.exe k:"

PRINT : PRINT "TRY TO CREATE GURU PROCESS...": PRINT
CHILD1$ = "e:\guru30\GURU.EXE" + CHR$(0)
C$ = " " + CHR$(0) + "-G -M g:\simula\COGNITI.IPF" + CHR$(0) + CHR$(0)
X = DosExecPgm%(0, 0, 0, 2, VARSEG(C$), SADD(C$), 0, 0, GURUResult, VARSEG(CHILD1$), SADD(CHILD1$))
IF (X) THEN PRINT "Error no.: "; X

X = DosSemWait%(SemHanPC1, -1)


' *** End of initialisation phase ***

X = DosSemClear%(SemHanMP1)
X = DosSemSetWait%(SemHanMP2, -1)
CLS


'*****************************************************************************
'********************** BEGINNING OF MAIN LOOP *****************************
'*****************************************************************************

LOOPBEGIN:

PrecDate$ = ActualDate$

'* Read values in shared memory

CALL ReadShrDateTime(SelMemDateTime, ActualDate$, ActualTime$)

'IF PrecDate$ <> ActualDate$ THEN
'   DaySim = VAL(MID$(ActualDate$, 5, 2))
'END IF

CALL ReadShrMem(SelMemWeather, 8, Weather())
TempOut = Weather(1)
'RelHum = Weather(2)
GlobRad = Weather(5)
WindVel = Weather(3)

CALL ReadShrMem(SelMemGhouse, 7, GhClimate())
TempIn = GhClimate(1)
RelHumIn = GhClimate(2)
'CO2In = GhClimate(3)

'CALL ReadShrMem(SelMemCrop, 1, CropState())
'NetCO2Uptake = CropState(1)


' Prepare data for averages

Sum1TempIn = Sum1TempIn + TempIn
Sum1TempOut = Sum1TempOut + TempOut
Sum1GlobRad = Sum1GlobRad + GlobRad
```

```
Sum1WindVel = Sum1WindVel + WindVel
Totl1Time = Totl1Time + SecondStep

Sum2TempIn = Sum2TempIn + TempIn
Sum2TempOut = Sum2TempOut + TempOut
Sum2GlobRad = Sum2GlobRad + GlobRad
Sum2WindVel = Sum2WindVel + WindVel
Tot2Time = Tot2Time + SecondStep

SumHeatSpt = SumHeatSpt + HeatSpt
IF Heating = 1 THEN SumChauf = SumChauf + 1


' *****   Call COGNITI every five minutes   *****

IF Totl1Time >= 300 THEN

        NumOfCount = Totl1Time / SecondStep: Totl1Time = 0

        AvgTempIn = Sum1TempIn / NumOfCount: Sum1TempIn = 0
        AvgTempOut = Sum1TempOut / NumOfCount: Sum1TempOut = 0
        AvgGlobRad = Sum1GlobRad / NumOfCount: Sum1GlobRad = 0
        AvgWindVel = Sum1WindVel / NumOfCount: Sum1WindVel = 0

        HourActTime = VAL(MID$(ActualTime$, 1, 2)) + VAL(MID$(ActualTime$, 4, 2)) / 60

        GOSUB CallCogniti

        IF FlagPrint = 0 THEN
                LOCATE 1, 1: PRINT ActualDate$ + "   "; ActualTime$
                PRINT HeatSpt, VentSpt
        END IF

END IF


' *****    CONTROL SECTION     *****

'* Control of ventilators
IF (TempIn > VentSpt + 1) THEN FanJet = 1
IF (TempIn < VentSpt + 01) THEN FanJet = 0
IF (Ventilator = 0 AND TempIn > VentSpt + 2) THEN Ventilator = 1
IF (TempIn > VentSpt + 41) THEN Ventilator = 2
IF (Ventilator = 2 AND TempIn < VentSpt + 2) THEN Ventilator = 1
IF (TempIn < VentSpt + 11) THEN Ventilator = 0

'* Control of heating system
IF (TempIn < HeatSpt - .5) THEN Heating = 1
IF (TempIn > HeatSpt + .5) THEN Heating = 0


'* Print values in shared memory

ActuatorVal(1) = FanJet
ActuatorVal(2) = Ventilator
ActuatorVal(3) = Heating
```

```
CALL WriteShrMem(SelMemPavlov, 3, ActuatorVal())

'* Print values on screen if required

IF FlagPrint = 1 THEN

LOCATE 1, 1: PRINT "MANAGER:"
LOCATE 1, 55: PRINT "(Press 's' to stop)"
LOCATE 3, 1: PRINT "WEATHER:": PRINT "------"
LOCATE 7, 1: PRINT "GHOUSE:": PRINT "-----"
LOCATE 20, 1: PRINT "CROP:": PRINT "----"
LOCATE 14, 1: PRINT "PAVLOV:": PRINT "-----"

LOCATE 14, 20: PRINT ActualDate$, ActualTime$
LOCATE 15, 15: PRINT "Glob. sol. rad.:"; : PRINT USING "#####.##"; GlobRad
LOCATE 15, 45: PRINT "Outside temp.  :"; : PRINT USING "#####.##"; TempOut
LOCATE 16, 15: PRINT "Temp. In.:"; : PRINT USING " ###.#"; TempIn
LOCATE 16, 35: PRINT "Heat Spt.:"; : PRINT USING " ###.#"; HeatSpt
LOCATE 16, 55: PRINT "Vent Spt.:"; : PRINT USING " ###.##"; VentSpt
LOCATE 17, 15: PRINT "FanJet state   :"; : PRINT USING "########"; FanJet
LOCATE 17, 45: PRINT "Ventil. state  :"; : PRINT USING "########"; Ventilator
LOCATE 18, 15: PRINT "Heating system :"; : PRINT USING "########"; Heating

END IF


'* Output data in files every hour

IF Tot2Time >= 3600 THEN

        NumOfCount = Tot2Time / SecondStep: Tot2Time = 0

        AvgTempIn = Sum2TempIn / NumOfCount: Sum2TempIn = 0
        AvgTempOut = Sum2TempOut / NumOfCount: Sum2TempOut = 0
        AvgGlobRad = Sum2GlobRad / NumOfCount: Sum2GlobRad = 0
        AvgWindVel = Sum2WindVel / NumOfCount: Sum2WindVel = 0
        AvgChauf = SumChauf * 300 / NumOfCount: SumChauf = 0

        AvgHeatSpt = SumHeatSpt / NumOfCount: SumHeatSpt = 0

        PRINT #1, USING " ###"; VAL(MID$(ActualDate$, 3, 2)); VAL(MID$(ActualDate$, 5, 2));
        PRINT #1, USING " #####.##"; HourActTime; AvgTempOut; AvgGlobRad; AvgWindVel; AvgHeatSpt;
AvgTempIn; AvgChauf

END IF


X = DosSemSet%(SemHanMP2)
X = DosSemClear%(SemHanMP1)
X = DosSemWait%(SemHanMP2, -1)

GOTO LOOPBEGIN
'*************************** End of loop ************************************

ENDOFSIMULATION:
```

```
END

'********************************************************************************
'****************** Call to COGNITI process when necessary *******************
'********************************************************************************
CallCogniti:

COGNITIIN = FREEFILE
OPEN "k:COGNITI.IN" FOR OUTPUT AS #COGNITIIN
PRINT #COGNITIIN, ActualDate$
PRINT #COGNITIIN, HourActTime
PRINT #COGNITIIN, AvgTempOut
PRINT #COGNITIIN, AvgWindVel
PRINT #COGNITIIN, AvgGlobRad
PRINT #COGNITIIN, AvgTempIn
CLOSE #COGNITIIN


'* Execute one cycle for COGNITI by clearing a semaphore

X = DosSemSet%(SemHanPC1)
X = DosSemClear%(SemHanPC2)
X = DosSemWait%(SemHanPC1, -1)


' Read control values from COGNITI in k:COGNITI.OUT

COGNITIOUT = FREEFILE
OPEN "K:COGNITI.OUT" FOR INPUT AS #COGNITIOUT
INPUT #COGNITIOUT, HeatSpt, VentSpt
CLOSE #COGNITIOUT

CLS
RETURN
'********************** End of call to COGNITI process ***********************


'***** Subroutine executed when Signal A is detected *****

SignalADetected:

CLOSE #1

X = DosKillProcess%(0, GURUResult.codeTerminate)
IF (X) THEN
        PRINT "Error in closing Child"
ELSE
        PRINT "GURU child is closed"
END IF

'* Clear semaphore of COGNCTRL.EXE
X = DosSemClear%(SemHanPC2)

'* Close all semaphores
```

```
X = DosCloseSem%(SemHanMP2)

X = DosSemClear%(SemHanMP1)
X = DosCloseSem%(SemHanMP1)

END

RETURN
'*******************************************************************************

'*******************************************************************************
SUB ReadShrDateTime (SelMemName AS INTEGER, DateRead$, TimeRead$)

DEF SEG = SelMemName
DateRead$ = ""
FOR I = 1 TO 6
        DateRead$ = DateRead$ + CHR$(PEEK(I))
NEXT I
TimeRead$ = ""
FOR I = 7 TO 14
        TimeRead$ = TimeRead$ + CHR$(PEEK(I))
NEXT I

DEF SEG

END SUB
```

G.7. File COGNITI.101

```
/***************************************************************************/
/******              COGNITI.101                         **************/
/***************************************************************************/
/* Day/night (21/17 C) regime */

E.WFU = false
E.CF = 2
PI := 3.14159
LoopFlag = 99999


run "K:COGNCTRL.EXE"

/***                    ***/
WHILE LoopFlag = 99999 DO

InFile = fopen("k:cogniti.in", "r")


IF InFile <> -1 THEN
  ActDate = fgetl(InFile)
  ActTime = tonum(fgetl(InFile))
  TempOut = tonum(fgetl(InFile))
  WindVel = tonum(fgetl(InFile))
  GlobRad = tonum(fgetl(InFile))
  TempIn = tonum(fgetl(InFile))
  /* output ActDate, ActTime, TempOut, GlobRad, TempIn */
  fclose(InFile)
ELSE
  output "Error opening file (read)..."
  fclose(InFile)
ENDIF

IF GlobRad < 10 THEN
  HeatSpt = 17
ELSE
  HeatSpt = 21
ENDIF

VentSpt = 25
/* OUTPUT HeatSpt, VentSpt */

OutFile = fopen("k:cogniti.out", "w")


IF OutFile <> -1 THEN
  fputl(tostr(HeatSpt,10,3), OutFile)
  fputl(tostr(VentSpt,10,3), OutFile)
ELSE
  output "Error opening file (write)..."
ENDIF
fclose(OutFile)

RUN "K:COGNCTRL.EXE"

ENDWHILE
```

G.8. File COGNITI.102

```
/****************************************************************************/
/************************** COGNITI.102 **************************************/
/****************************************************************************/
/* Day night regime (21/17 C) with compensation */


E.WFU = FALSE
E.SUPD = TRUE
PI = 3.14159
LoopFlag = 99999

TimeStep = 300
RNLENGTH = 0
DRTSum = 0
NRTSum = 0
DayCtr = 0
DayPeriod = FALSE
NDTAvg = 17
DayHeatSpt = 21
MinHeatSpt = 12


/*************************** INITIALISATION ********************************/

USE "g:\\gurufile\\itb\\SOLAR.ITB"
LOAD UDF "g:\\gurufile\\kgl\\GENFCT1.KGB"

/* Input data from Pavlov module */

InFile = FOPEN("k:cogniti.in", "r")
IF InFile <> -1 THEN
   ActDate = fgetl(InFile)
   ActTime = tonum(fgetl(InFile))
   /* output ActDate, ActTime */
ELSE
   output "Error opening file (read)..."
ENDIF
FCLOSE(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))
OBTAIN FOR SDayNumber = NTHDAY (Month, Day)

IF SunRise < ActTime THEN
   RNLength = (SunRise-ActTime) * 3600 + 86400
ELSE
   RNLength = (SunRise-ActTime) * 3600
ENDIF
NDTSum = RNLength * NDTAvg
/* OUTPUT RNLENGTH, ActTime, SunRise */


RUN "K:COGNCTRL.EXE"


/*************************** MAIN LOOP ********************************/
```

```
WHILE LoopFlag = 99999 DO

InFile = fopen("k:cogniti.in", "r")
IF InFile <> -1 THEN
  ActDate = fgetl(InFile)
  ActTime = tonum(fgetl(InFile))
  TempOut = tonum(fgetl(InFile))
  WindVel = tonum(fgetl(InFile))
  GlobRad = tonum(fgetl(InFile))
  TempIn = tonum(fgetl(InFile))
  /* output ActDate, ActTime, TempOut, GlobRad, TempIn */
ELSE
  output "Error opening file (read)..."
ENDIF
fclose(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))


/* Calculate HeatSpt */

IF GlobRad > 10 THEN
   DRTSum = DRTSum + (TempIn * TimeStep)
   DayCtr = DayCtr + 1
   HeatSpt = DayHeatSpt
   DayPeriod = TRUE
ELSE
   IF DayPeriod = TRUE THEN
         OBTAIN FOR SDayNumber = NTHDAY (Month, Day) + 1
         RNLength = (SunRise-ActTime) * 3600 + 86400
         DTSurplus = DRTSum - (DayCtr * DayHeatSpt * TimeStep)
         NDTSum = RNLength * NDTAvg - DTSurplus
         NRTSum = 0
         DayCtr = 0
         DRTSum = 0
         DayPeriod = FALSE
   ENDIF
   NRTSum = NRTSum + (TempIn * TimeStep)
   RNLength = RNLength - TimeStep
   IF RNLength > 0 THEN
         HeatSpt = (NDTSum - NRTSum) / RNLength
   ELSE
         HeatSpt = NDTAvg
   ENDIF
ENDIF

IF HeatSpt < MinHeatSpt THEN HeatSpt = MinHeatSpt; ENDIF

VentSpt = 25

/* OUTPUT HeatSpt, VentSpt; OUTPUT */

/* Output results for Pavlov module */

OutFile = FOPEN("k:cogniti.out", "w")
```

```
IF OutFile <> -1 THEN
  fputl(tostr(HeatSpt,10,3), OutFile)
  fputl(tostr(VentSpt,10,3), OutFile)
ELSE
  output "Error opening file (write)..."
ENDIF
FCLOSE(OutFile)


RUN "K:COGNCTRL.EXE"

ENDWHILE
```

G.9. File COGNITI.103

```
/********************************************************************************/
/************************** COGNITI.103 *************************************/
/********************************************************************************/
/* File to produce data for NN learning - random FLAT and SINE              */

E.WFU = FALSE
E.SUPD = TRUE
PI = 3.14159
LoopFlag = 99999

TimeStep = 300
MaxHeatSpt = 24.5
MinHeatSpt = 14
HeatSpt = 20


/**************************** INITIALISATION **********************************/

SEED = TONUM(SUBSTR(time(),1,2))*3600 + TONUM(SUBSTR(time(),4,2))*60
SEED = SEED + TONUM(SUBSTR(time(),7,2))
RAND(SEED)

run "K:COGNCTRL.EXE"


/****************************  MAIN LOOP   ***********************************/

WHILE LoopFlag = 99999 DO

/*** New series ***/
TOLast = HeatSpt
LOPeriod = RAND(0) * 86400
NOIncr = LOPeriod / TimeStep

IF RAND(0) < 0.5 THEN
  FcType = "FLAT"
  CtSpt = MinHeatSpt + RAND(0) * (MaxHeatSpt - MinHeatSpt)
ELSE
  FcType = "SINE"
  Ampl = RAND(0) * (MaxHeatSpt - MinHeatSpt) / 4
ENDIF


/* Loop for series */

CtrPer = 1

WHILE CtrPer <= NOIncr DO

  IF FcType = "FLAT" THEN
    HeatSpt = CtSpt
  ELSE
    HeatSpt = TOLast + Ampl + Ampl * Sin(2*PI*CtrPer/NOIncr-PI/2)
  ENDIF

  IF HeatSpt > MaxHeatSpt THEN HeatSpt = MaxHeatSpt; ENDIF
```

```
    IF HeatSpt < MinHeatSpt THEN HeatSpt = MinHeatSpt; ENDIF

    VentSpt = 25

    OutFile = FOPEN("k:cogniti.out", "w")
    IF OutFile <> -1 THEN
      fputl(tostr(HeatSpt,10,3), OutFile)
      fputl(tostr(VentSpt,10,3), OutFile)
    ELSE
      output "Error opening file (write)..."
    ENDIF
    FCLOSE(OutFile)

    RUN "K:COGNCTRL.EXE"

    CtrPer = CtrPer + 1

  ENDWHILE
  /* End of series loop */


  ENDWHILE
```

G.10. File COGNITI.104

```
/*****************************************************************************/
/************************** COGNITI.104 **************************************/
/*****************************************************************************/
/* File to produce data for NN learning - random FLAT and compensation      */

E.WFU = FALSE
E.SUPD = TRUE
PI = 3.14159
LoopFlag = 99999

TimeStep = 300
NightBefore = FALSE
RTSum = 0
MinHeatSpt = 12
MaxHeatSpt = 24.5

DIM DesAvgTemp(12)
DesAvgTemp(1) = 18.4
DesAvgTemp(2) = 18.7
DesAvgTemp(3) = 18.9
DesAvgTemp(4) = 19.5
DesAvgTemp(5) = 21.5


/*************************** INITIALISATION *********************************/

SEED = TONUM(SUBSTR(time(),1,2))*3600 + TONUM(SUBSTR(time(),4,2))*60
SEED = SEED + TONUM(SUBSTR(time(),7,2))
RAND(SEED)

USE "g:\\gurufile\\itb\\SOLAR.ITB"
LOAD UDF "g:\\gurufile\\kgl\\GENFCT1.KGB"

/* Input data from Pavlov module */

InFile = FOPEN("k:cogniti.in", "r")
IF InFile <> -1 THEN
   ActDate = fgetl(InFile)
   ActTime = tonum(fgetl(InFile))
   /* output ActDate, ActTime */
ELSE
   output "Error opening file (read)..."
ENDIF
FCLOSE(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))
OBTAIN FOR SDayNumber = NTHDAY (Month, Day)
IF SunRise < ActTime THEN
   RNLENGTH = (SunRise-ActTime) * 3600 + 86400
ELSE
   RNLENGTH = (SunRise-ActTime) * 3600
ENDIF
DTSum = RNLength * DesAvgTemp(Month)
/* OUTPUT RNLENGTH, ActTime, SunRise */
```

```
DayHeatSpt = RAND(0) * (24.5 - 14) + 14

RUN "K:COGNCTRL.EXE"

/***************************  MAIN LOOP  *********************************/

WHILE LoopFlag = 99999 DO

InFile = fopen("k:cogniti.in", "r")
IF InFile <> -1 THEN
  ActDate = fgetl(InFile)
  ActTime = tonum(fgetl(InFile))
  TempOut = tonum(fgetl(InFile))
  WindVel = tonum(fgetl(InFile))
  GlobRad = tonum(fgetl(InFile))
  TempIn = tonum(fgetl(InFile))
  /* output ActDate, ActTime, TempOut, GlobRad, TempIn */
ELSE
  output "Error opening file (read)..."
ENDIF
fclose(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))

RTSum = RTSum + TempIn * TimeStep
RNLENGTH = RNLENGTH - TimeStep


/* Verify if a new period of 24 hours starts */

IF RNLENGTH <= 0 THEN
  RTSum = 0
  OBTAIN FOR SDayNumber = NTHDAY (Month, Day) + 1
  RNLength = (SunRise-ActTime) * 3600 + 86400
  DTSum = RNLength * DesAvgTemp(Month)
  DayHeatSpt = RAND(0) * (24.5 - 14) + 14
  NightBefore = TRUE
  /* OUTPUT RNLENGTH, ActTime, SunRise, DayHeatSpt */
ENDIF

/* Calculate HeatSpt */

IF GlobRad > 10 THEN
   NightBefore = FALSE
   HeatSpt = DayHeatSpt
ELSE
   IF NightBefore = TRUE THEN
        HeatSpt = DayHeatSpt
   ELSE
        HeatSpt = (DTSum - RTSum) / RNLength
   ENDIF
ENDIF
```

```
IF HeatSpt < MinHeatSpt THEN HeatSpt = MinHeatSpt; ENDIF
IF HeatSpt > MaxHeatSpt THEN HeatSpt = MaxHeatSpt; ENDIF

VentSpt = 25
/* OUTPUT HeatSpt, VentSpt; OUTPUT */

/* Output results for Pavlov module */

OutFile = FOPEN("k:cogniti.out", "w")
IF OutFile <> -1 THEN
   fputl(tostr(HeatSpt,10,3), OutFile)
   fputl(tostr(VentSpt,10,3), OutFile)
ELSE
   output "Error opening file (write)..."
ENDIF
FCLOSE(OutFile)


RUN "K:COGNCTRL.EXE"

ENDWHILE
```

G.11. File COGNITI.201

```
/*****************************************************************************/
/************************* COGNITI201 *************************************/
/*****************************************************************************/
/* Simulation-based cognitive controller */


/************************** INITIALISATION ********************************/

E.WFU  = FALSE
E.SUPD = TRUE
E.TRAC = "N"

DIM TestSpt(24);
DIM DAvgTemp(12);
DIM SimIn(25,6);
DIM ChosSpt(24);

GLOBAL DIM SimOut(24,3);
GLOBAL DIM PredOut(24,3);
GLOBAL DIM Yin(9)
GLOBAL DIM Yout(3)

GLOBAL NightLen, DayLen, DMMethod


/*** Values to be modified for each experiment ***/

/* 1982: 18.4, 18.7, 18.9, 19.5, 21.5 */
/* 1983: 18.4, 18.7, 18.9, 19.2, 20.1 */

DAvgTemp(1) = 18.4
DAvgTemp(2) = 18.7
DAvgTemp(3) = 18.9
DAvgTemp(4) = 19.2
DAvgTemp(5) = 20.1

FrstTab = "g:\\gurufile\\itb\\MTL83.ITB"

DMMethod = "PrHeOnly"

FrEff = 0.75
FrPrice = 2.00
FurnEff = .75
RatioJS = 38850
EnerPrice = 0.0976

/**************************************************/


/* Open files, tables, etc. */

USE "g:\\gurufile\\itb\\SOLAR.ITB"
USE FrstTab AS FrstTab

USE ScenaX
ERASE "K:Scena1.itb"
```

```
IMPRESS ScenaX TO Scena1 WITH "K:Scena1.itb"
FINISH ScenaX

LOAD UDF "g:\\gurufile\\kgl\\COGNFCT2.KGB"
LOAD UDF "g:\\gurufile\\kgl\\GENFCT1.KGB"
LOAD FUNCTION "e:\\GURU30\\GHOUSENN.DLL"


/* Initialisation of variables */

PI        = 3.14159

LoopFlag = 99999
TimeStep = 300

SumTO = 0; SumWV = 0; SumGR = 0; SumTI = 0
Avg2TO = 0; Avg2WV = 0; Avg2GR = 0; Avg2TI = 0
HourCount = 0
RTSum = 0

MinHeatSpt = 15
MaxHeatSpt = 24


/* Input data from Pavlov module */

InFile = FOPEN("k:cogniti.in", "r")
IF InFile <> -1 THEN
  ActDate = fgetl(InFile)
  ActTime = tonum(fgetl(InFile))
  output ActDate, ActTime
ELSE
  output "Error opening file (read)..."
ENDIF
FCLOSE(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))

OBTAIN FROM SOLAR FOR SDayNumber = NTHDAY(Month, Day)
EndOfDay = TRUNC(Solar.Sunset) + 1

IF EndOfDay < ActTime THEN
  RDLength = (EndOfDay - ActTime) * 3600 + 86400
ELSE
  RDLength = (EndOfDay - ActTime) * 3600
ENDIF

OUTPUT ActTime, EndOfDay, RDLength

NumOfRec = TRUNC(RDLength / 3600)
ATTACH NumOfRec TO Scena1
XXHour = TRUNC(ActTime)
XXDay = NTHDAY(Month, Day)

I = 1; WHILE I <= NumOfRec DO
```

```
  OBTAIN I FROM Scena1;
  XXHour = XXHour + 1
  IF XXHour > 24 THEN
    XXHour = XXHour - 24
    XXDay = NTHDAY(Month, Day) + 1
  ENDIF
  Scena1.CDayNumber := XXDay
  Scena1.CHour := XXHour
  Scena1.CChosSpt := DAvgTemp(Month)
  Scena1.CPTemp := 0
  Scena1.CPHeat := 0
  Scena1.CPCO2 := 0
  I = I + 1
ENDWHILE

DTSum = RDLength * DAvgTemp(Month)


RUN "K:COGNCTRL.EXE"

/**************************** MAIN LOOP ********************************/

WHILE LoopFlag = 99999 DO

InFile = fopen("k:cogniti.in", "r")
IF InFile <> -1 THEN
  ActDate = fgetl(InFile)
  ActTime = tonum(fgetl(InFile))
  TempOut = tonum(fgetl(InFile))
  WindVel = tonum(fgetl(InFile))
  GlobRad = tonum(fgetl(InFile))
  TempIn = tonum(fgetl(InFile))
  output ActDate, ActTime, TempOut, GlobRad, TempIn
ELSE
  output "Error opening file (read)..."
ENDIF
fclose(InFile)

Month = TONUM(SUBSTR(ActDate,3,2))
Day = TONUM(SUBSTR(ActDate,5,2))

SumTO = SumTO + TempOut
SumWV = SumWV + WindVel
SumGR = SumGR + GlobRad
SumTI = SumTI + TempIn

RTSum = RTSum + TempIn * TimeStep
RDLength = RDLength - TimeStep


/* Every hour, compile statistics */

HourCount = HourCount + TimeStep

IF HourCount >= 3600 THEN
```

```
  Avg1TO = Avg2TO
  Avg1WV = Avg2WV
  Avg1GR = Avg2GR
  Avg1TI = Avg2TI

  Avg2TO = SumTO / (HourCount/TimeStep)
  Avg2WV = SumWV / (HourCount/TimeStep)
  Avg2GR = SumGR / (HourCount/TimeStep)
  Avg2TI = SumTI / (HourCount/TimeStep)

  output Avg2TO,Avg2WV,Avg2GR,Avg2TI

  SumTO = 0
  SumWV = 0
  SumGR = 0
  SumTI = 0
  HourCount = 0

ENDIF


/* Every 24 hours, start a new period and generate 24 hourly setpoints */

IF RDLength <= 0 THEN

  DayNumber = NTHDAY(Month, Day)

  OBTAIN FROM SOLAR FOR SDayNumber = DayNumber + 1
  EndOfDay = TRUNC(Solar.Sunset) + 1
  RDLength = (EndOfDay - ActTime) * 3600 + 86400
  OUTPUT ActTime, EndOfDay, RDLength

  DTSum = RDLength * DAvgTemp(Month)
  RTSum = 0


  /******************** Generate 24 hourly setpoints ********************/

  LastTemp := Avg2TI

  /* Obtain meteorological conditions */

  ObtMet(DayNumber, LastTemp, SimIn);


  /***** Loop for different temp. setpoints *****/

  NightSpt := 14
  FirstEva := TRUE
  WHILE NightSpt <= 24 DO

    /* Determine setpoints */

    SptGen1(NightSpt, DAvgTemp(Month), TestSpt);
    /* Execute simulations */

    DoSim1(SimIn, TestSpt, DAvgTemp(Month), MinHeatSpt, MaxHeatSpt, SimOut);


    /* Calculate statistics */

    PAvgTemp := AVGARR2(SimOut,      ., 24);
    PAvgHeat := AVGARR2(SimOut, 2, 1, 24);
    PSumCO2 := SUMARR2(SimOut, 3, 1, 24);
    PFrFruit := CalWFruit(PSumCO2);

    ? TestSpt(1), PAvgTemp, PAvgHeat, PSumCO2, PFrFruit;


    /* Make comparison and decide on one scenario */

    IF FirstEva THEN
          FirstEva = FALSE
          ChSptArr(TestSpt, SimOut, ChosSpt, PredOut);
    ELSE
          CONSULT "G:\\gurufile\\es\\SimAnal.rsc" TO SEEK
          IF TYPE(Scenario) = "UNKNOWN" THEN
            ? "Problem in consultation..."
            WAIT; STOP
          ENDIF
          IF Scenario = 2 THEN
            ChSptArr(TestSpt, SimOut, ChosSpt, PredOut);
          ENDIF
    ENDIF

    NightSpt = NightSpt + 1

  ENDWHILE
  /***** Loop NightSpt *****/


  /* Put Results in table */

  FillTab(Scena1, SimIn, ChosSpt, PredOut);


  /* Output results on screen */

  PAvgTemp := AVGARR2(PredOut, 1, 1, 24);
  MaxSpt := MAXARR1(ChosSpt, 1, 24);
  PAvgHeat := AVGARR2(PredOut, 2, 1, 24);
  PSumCO2 := SUMARR2(PredOut, 3, 1, 24);

  ?; ? "For Day number: ", DayNumber, " Night temp. chosen is: ", ChosSpt(1);
  ? PAvgTemp, PAvgHeat, PSumCO2;
  ? "Max. setpoint: ", MaxSpt; ?;

  /**********************************************************************/

ENDIF
```

```
/* Generate instantaneous setpoint for heating from array ChosSpt */

DayNumber = NTHDAY(Month, Day)
OBTAIN FROM Scenal FOR CDayNumber = DayNumber AND CHour = (TRUNC(ActTime)+1)
HeatSpt = Scenal.CChosSpt


/* Output setpoint */

VentSpt = 25
OUTPUT HeatSpt, VentSpt; OUTPUT

OutFile = FOPEN("k:cogniti.out", "w")
IF OutFile <> -1 THEN
  fputl(tostr(HeatSpt,10,3), OutFile)
  fputl(tostr(VentSpt,10,3), OutFile)
ELSE
  output "Error opening file (write)..."
ENDIF
FCLOSE(OutFile)


RUN "K:COGNCTRL.EXE"



ENDWHILE
/************************** END OF MAIN LOOP *******************************/
```

G.12. File GENFCT1.KGL

This file is written in GURU User Defined Functions (UDF), which are used by COGNITI.102, COGNITI.104 and COGNITI.201.

```
/****************************************************************************/
/**************      GENFCT1.KGL              *******************/

/****************************************************************************/
/* - SUMARR2 FUNCTION                             *******************/
/* Calculates sum of elements of an array 2-D                      */
/****************************************************************************/

FUNCTION SUMARR2(ARRAY Arr, Col, AStart, AEnd)

LOCAL I, SumArr;

SumArr := 0;
FOR I:= AStart TO AEnd DO
  SumArr := SumArr + Arr(I,Col);
ENDFOR

RETURN(SumArr);

ENDFUNCTION


/****************************************************************************/
/* - AVGARR2 FUNCTION                             *******************/
/* Calculates average of elements of an array 2-D                  */
/****************************************************************************/

FUNCTION AVGARR2(ARRAY Arr, Col, AStart, AEnd)

LOCAL AvgArr;

AvgArr := SUMARR2(Arr, Col, AStart, AEnd) / (AEnd-AStart+1);

RETURN(AvgArr);

ENDFUNCTION


/****************************************************************************/
/* - MAXARR1 FUNCTION                             *******************/
/* Finds the maximum value within an array 1-D                     */
/****************************************************************************/

FUNCTION MAXARR1(ARRAY Arr, AStart, AEnd)

LOCAL I, MaxArr;

MaxArr := -99999;

FOR I:= AStart TO AEnd DO
  MaxArr := MAX(Arr(I), MaxArr);
ENDFOR

RETURN(MaxArr);

ENDFUNCTION
```

```
/****************************************************************************/
/* - NTHDAY FUNCTION                              *******************/
/* Calculates day of the year from day and month                    */
/****************************************************************************/

FUNCTION NTHDAY(Month, Day)

LOCAL NTH;

NTH := Day + 61 * Month / 2 - 30;
IF Month > 2 THEN NTH := NTH - 2; ENDIF
IF Month = 1 OR Month = 3 OR Month = 5 OR Month = 7 THEN NTH := NTH - .5;ENDIF
IF Month = 9 OR Month = 11 THEN NTH := NTH + .5; ENDIF

RETURN(NTH);

ENDFUNCTION
/****************************************************************************/
```

G.13. File COGNFCT2.KGL

This file is written in GURU User Defined Functions (UDF), which are used by COGNITI.201.

```
/****************************************************************************/
/**************      COGNFCT2.KGL          *******************/
/****************************************************************************/


/****************************************************************************/
/* - ObtMet FUNCTION                                   ********************/
/* Obtain expected meteorological conditions                          */
/****************************************************************************/

FUNCTION ObtMet(DayNumber, LastTemp, ARRAY SimIn)


/* Initialisation */

LOCAL I, RecNum1, DayBegin, FlagA;

GLOBAL Solar;
GLOBAL FrstTab;

GLOBAL DayLen, NightLen;

FlagA := FALSE;


/**********************        Main          *****************************/

/* RecNum1 is the number of the physical record containing data for          */
/* the hour during which SUNSET occured                                      */

OBTAIN DayNumber FROM Solar;
RecNum1 := (DayNumber - 1) * 24 + TRUNC(Solar.SunSet) + 1;

OBTAIN RecNum1 FROM FrstTab;
SimIn(1,3) := FrstTab.WTempOut;
SimIn(1,4) := FrstTab.WGlobRad;
SimIn(1,6) := LastTemp;

FOR I := 2 TO 25 DO
  OBTAIN (RecNum1 - 1) + I FROM FrstTab;
  SimIn(I,1) := FrstTab.WDayNumber;
  SimIn(I,2) := FrstTab.WHour;
  SimIn(I,3) := FrstTab.WTempOut;
  SimIn(I,4) := FrstTab.WGlobRad;
  SimIn(I,5) := FrstTab.WWindVel;
  IF (FrstTab.WGlobRad > 1 AND FlagA = FALSE) THEN
    DayBegin := CURREC(FrstTab);
    FlagA := TRUE;
  ENDIF
ENDFOR


/* Calculate NightLen and DayLen */

NightLen := DayBegin - RecNum1 - 1;
DayLen := 24 - NightLen;
```

```
/******************************************************************************/

ENDFUNCTION

/******************************************************************************/
/* - FillTab FUNCTION                              ********************/
/* Put Results in table                                              */
/******************************************************************************/

FUNCTION FillTab(NamOfTab, ARRAY SimIn, ARRAY ChosSpt, ARRAY PredOut)


/* Initialisation */

LOCAL I, RecNum2;
LOCAL PreDay, PreHour, PreSpt, PreTemp, PreHeat, PreCO2;


/**********************          Main         ****************************/

OBTAIN LAST FROM NamOfTab;

IF NamOfTab.CHour < (SimIn(2,2) - 1) THEN

   PreDay := NamOfTab.CDayNumber;
   PreHour := NamOfTab.CHour;
   PreSpt := NamOfTab.CChosSpt;
   PreTemp := NamOfTab.CPTemp;
   PreHeat := NamOfTab.CPHeat;
   PreCO2 := NamOfTab.CPCO2;

   ATTACH 1 TO NamOfTab;
   OBTAIN LAST FROM NamOfTab;

   NamOfTab.CDayNumber := PreDay;
   NamOfTab.CHour := PreHour + 1;
   NamOfTab.CChosSpt := PreSpt;
   NamOfTab.CPTemp := PreTemp;
   NamOfTab.CPHeat := PreHeat;
   NamOfTab.CPCO2 := PreCO2;

ENDIF

RecNum2 := CURREC(NamOfTab);
ATTACH 24 TO NamOfTab;

FOR I := 1 TO 24 DO

   OBTAIN (RecNum2 + I) FROM NamOfTab;
   NamOfTab.CDayNumber := SimIn(I+1, 1);
   NamOfTab.CHour := SimIn(I+1, 2);
   NamOfTab.CChosSpt := ChosSpt(I);
   NamOfTab.CPTemp := PredOut(I, 1);
   NamOfTab.CPHeat := PredOut(I, 2);
```

```
  NamOfTab.CPCO2 := PredOut(I, 3);

ENDFOR

/****************************************************************************/

ENDFUNCTION  /* FillTab */



/****************************************************************************/
/* - SptGen1 FUNCTION                                  ********************/
/* Changes array with chosen setpoints                              */
/****************************************************************************/

FUNCTION SptGen1(NightSpt, DAvgTemp, ARRAY TestSpt)

/* Initialisation */

LOCAL I, DaySpt;
GLOBAL DayLen, NightLen;


/* Determine setpoints */

DaySpt := (24 * DAvgTemp - NightSpt * NightLen) / DayLen;

FOR I := 1 TO 24 DO
  IF I <= NightLen THEN
    TestSpt(I) := NightSpt;
  ELSE
    TestSpt(I) := DaySpt;
  ENDIF
ENDFOR

/****************************************************************************/

ENDFUNCTION  /* SptGen1 */



/****************************************************************************/
/* - ChSptArr FUNCTION                                 ********************/
/* Determines setpoints                                             */
/****************************************************************************/

FUNCTION ChSptArr(ARRAY TestSpt, ARRAY SimOut, ARRAY ChosSpt, ARRAY PredOut)


/* Initialisation */

LOCAL I, J;


/* Replace ChosSpt by TestSpt */
```

```
FOR I := 1 TO 24 DO
  ChosSpt(I) := TestSpt(I);
  FOR J := 1 TO 3 DO
    PredOut(I, J) := SimOut(I, J);
  ENDFOR
ENDFOR

/*****************************************************************************/

ENDFUNCTION  /* ChSptArr */



/*****************************************************************************/
/* - DoSim1 FUNCTION                                    ******************/
/* Runs simulations - checks if desired avg. temp. is realised          */
/* If not, it modifies the setpoints                                    */
/*****************************************************************************/

FUNCTION DoSim1(ARRAY SimIn, ARRAY TestSpt, DAvgTemp, MinHeatSpt, MaxHeatSpt, ARRAY SimOut)


/* Initialisation */

LOCAL I, SimulCont, Ctrl, PAvgTemp, AvgTDiff, MaxTDiff;

SimulCont := TRUE;
Ctrl := 0;
MaxTDiff := 0.2;


/***        Loop for simulations until temp. diff. is not too large      ***/

WHILE SimulCont = TRUE DO

  Ctrl := Ctrl + 1;
  IF Ctrl >= 30 THEN
    SimulCont := FALSE;
    ? "SimulCont becomes FALSE because of counter";
  ENDIF


  /* Execute simulation */

  SimulA(SimIn, TestSpt, SimOut);


  /* Calculate statistics */

  PAvgTemp := AVGARR2(SimOut, 1, 1, 24);


  /* Test to redo simulation */

  AvgTDiff := PAvgTemp - DAvgTemp;
  IF ABS(AvgTDiff) > MaxTDiff THEN
    FOR I := 1 TO 24 DO
          TestSpt(I) := TestSpt(I) - AvgTDiff;
          IF TestSpt(I) < MinHeatSpt THEN
            TestSpt(I) := MinHeatSpt;
          ENDIF
          IF TestSpt(I) > MaxHeatSpt THEN
            TestSpt(I) := MaxHeatSpt;
          ENDIF
    ENDFOR
  ELSE
    SimulCont := FALSE;
  ENDIF


ENDWHILE
/***                       Loop SimulCon                          ***/

? "Counter: ", Ctrl;

/****************************************************************************/


ENDFUNCTION



/***************************************************************************/
/* - SimulA FUNCTION                              *******************/
/* Simulates over 24 hours                                         */
/***************************************************************************/

FUNCTION SimulA(ARRAY SimIn, ARRAY TestSpt, ARRAY SimOut)

CLINK ghousenn;
LOCAL I, XXX;
GLOBAL ARRAY Yin;
GLOBAL ARRAY Yout;


/***************************************************************************/

FOR I := 1 TO 24 DO

  Yin(1) := SimIn(I+1,1);  /* WDayNumber      */
  Yin(2) := SimIn(I+1,2);  /* WHour           */
  Yin(3) := SimIn(I+1,3);  /* WTempOut (lag 0) */
  Yin(4) := SimIn(I,3);    /* WTempOut (lag 1) */
  Yin(5) := SimIn(I+1,4);  /* WGlobRad (lag 0) */
  Yin(6) := SimIn(I,4);    /* WGlobRad (lag 1) */
  Yin(7) := SimIn(I+1,5);  /* WWindVel        */
  Yin(8) := SimIn(I,6);    /* TempIn (lag 1)   */
  Yin(9) := TestSpt(I);

  XXX := ghousenn();
  /* Consult rule set to adjust predicted values */

  IF Yout(2) < 0 THEN
     Yout(2) := 0;
  ENDIF


  /* Fill array SimOUt */

  SimOut(I,1) := Yout(1);
  SimOut(I,2) := Yout(2);
  SimOut(I,3) := Yout(3);

/* Predicted temp. is used as an input for next hour */

  SimIn(I+1,6) := Yout(1);

ENDFOR

/****************************************************************************/


ENDFUNCTION



/****************************************************************************/
/* - CalWFruit FUNCTION                              ********************/
/* Calculates daily fresh fruit production in kg/m2                       */
/* Input: DTNA - Daily total net assimilation [g CO2 /m2 /d]              */
/****************************************************************************/

FUNCTION CalWFruit(DTNA)


/* Initialisation */

LOCAL PERCDW, ASRQSO, ASRQLV, ASRQST, ASRQRT;
LOCAL FLV, FST, FSO, FRT, WLV, WRT, WST, WSO;
LOCAL NPHOT, ASRQ, GTW, GSO, FrFruit;


/* Percentage dry weight */

PERCDW := 3.5;


/* Assimilate requirements of organs [ g CH2O/g dry matter of organ ] */

ASRQSO := 1.37;
ASRQLV := 1.39;
ASRQST := 1.45;
ASRQRT := 1.39;


/* Dry matter partitioning */
```

```
FLV := .17;
FST := .08;
FSO := .72;
FRT := .03;


/* Dry weights of leaves, stems, roots and storage organs [g m-2] */

WLV := 75;
WRT := 15;
WST := 35;
WSO := 40;



/***** Calculations for DAILY values *****/

/*------ DTNA       Daily total net assimilation [g CO2/m2/d]  */
/*------ NPHOT      Net photosynthesis [g CH2O m-2 day-1]       */

/* DTNA is sum of SimOut(I,3)                               */

NPHOT := DTNA * 30 / 44;


/*------ Assimilate requirements for dry matter conversion     */
/*------ [g CH2O/g dry matter]                                */

ASRQ := FLV * ASRQLV + FST * ASRQST + FSO * ASRQSO + FRT * ASRQRT;


/*------Rate of growth         [g d.w. m-2 day-1]             */

GTW := NPHOT / ASRQ;
GSO := GTW * FSO;


/*------ Fresh weight of fruits [kg m-2]                        */
/*------ PERCDW = Percentage dry weight of fruits               */

FrFruit := GSO * (100 / PERCDW) / 1000;


/* Must add a multiplicative factor to attenuate potential growth */

/*  */


RETURN(FrFruit);

/**********************************************************************************/


ENDFUNCTION
```

G.14. File SIMANA1.RSS

This file is written in GURU expert system language
and is used by COGNITI.201.

```
GOAL:  Scenario

INITIAL:

        LOCAL TotFr1, TotFr2, TotEner1, TotEner2;
        LOCAL Profit1, Profit2
        LOCAL LarProfi, SmalEner, LarFruit

        Profit1   = UNKNOWN
        Profit2   = UNKNOWN
        TotFr1    = UNKNOWN
        TotFr2    = UNKNOWN
        TotEner1  = UNKNOWN
        TotEner2  = UNKNOWN

        LarProfi  = UNKNOWN
        SmalEner  = UNKNOWN
        LarFruit  = UNKNOWN

        Scenario  = UNKNOWN


RULE:  RULE1
        IF:      DMMethod = "PrHeFr"
        THEN:    Scenario = LarProfi
        NEEDS:   DMMethod
                 LarProfi
        REASON: Scenario 'LarProfi' leads to the largest profit.

RULE:  RULE2
        IF:      DMMethod = "PrHeOnly"
        THEN:    Scenario = SmalEner
        NEEDS:   DMMethod
                 SmalEner
        REASON: The scenario 'SmalEner' leads to the smallest energy consumption.

RULE:  RULE3
        IF:      DMMethod = "PrFrOnly"
        THEN:    Scenario = LarFruit
        NEEDS:   DMMethod
                 LarFruit
        REASON: The scenario 'LarFruit has been chosen since it leads to the largest production of fruits.

RULE:  RULE4
        IF:      TotEner1 > TotEner2
        THEN:    SmalEner = 2
        NEEDS:   TotEner1
                 TotEner2
        REASON: Scenario 1 requires more energy than scenario 2.

RULE:  RULE5
        IF:      TotEner1 <= TotEner2
        THEN:    SmalEner = 1
        NEEDS:   TotEner1
                 TotEner2
        REASON: Scenario 1 requires less or as much energy as scenario 2.
```

```
RULE:   RULE6
        IF:      TotFr1 >= TotFr2
        THEN:    LarFruit = 1
        NEEDS:   TotFr1
                 TotFr2
        REASON: Scenario 1 produces more or as many fruits as scenario 1.

RULE:   RULE7
        IF:      TotFr1 < TotFr2
        THEN:    LarFruit = 2
        NEEDS:   TotFr1
                 TotFr2
        REASON: Scenario 2 produces more fruits than scenario 1.

RULE:   RULE8
        IF:      Profit1 >= Profit2
        THEN:    LarProfi = 1
        NEEDS:   Profit1
                 Profit2
        REASON: With scenario 1, profits are larger or equal to those obtained with scenario 2.

RULE:   RULE9
        IF:      Profit1 < Profit2
        THEN:    LarProfi = 2
        NEEDS:   Profit1
                 Profit2
        REASON: With scenario 1, profits are lower than with scenario 2.


RULE:   RULE10
        IF:      KNOWN("TotFr1") AND KNOWN("TotEner1")
        THEN:    Profit1 = (TotFr1 * FrEff * FrPrice) - (TotEner1 / (FurnEff * RatioJS) * EnerPrice)
        REASON: Since unknown, the day's profit with scenario 1 is calculated.


RULE:   RULE11
        IF:      KNOWN("TotFr2") AND KNOWN("TotEner2")
        THEN:    Profit2 = (TotFr2 * FrEff * FrPrice) - (TotEner2 / (FurnEff * RatioJS) * EnerPrice)
        REASON: Since unknown, the day's profit with scenario 2 is calculated.


VAR:    TotEner1
        FIND:    TotEner1 = SumArr2(PredOut, 2, 1, 24) * 3.6

VAR:    TotEner2
        FIND:    TotEner2 = SumArr2(SimOut, 2, 1, 24) * 3.6

VAR:    TotFr1
        FIND:    TotFr1 = CalWFruit(SumArr2(PredOut, 3, 1, 24))

VAR:    TotFr2
        FIND:    TotFr2 = CalWFruit(SumArr2(SimOut, 3, 1, 24))

END:
```

G.15 File GHOUSENN.C

This file contains the C code for the neural network used by COGNITI.201.
A large portion of the file was created by NeuralWorks Professional II/PLUS

```
#include <stdio.h>
#include <math.h>
#include "clink.h"


/********
 * File that contains code for trained neural network
 ********/

int NN_Recall(Yin, Yout)
float Yin[9], Yout[3];

{
   float  Xout[41]; /* work arrays */
   long   ICmpT; /* temp for comparisons */

   /* Read and scale input into network */
   Xout[2] = Yin[0] * (0.0067114094) + (-0.013422819);
   Xout[3] = Yin[1] * (0.043478261) + (-0.043478261);
   Xout[4] = Yin[2] * (0.017513135) + (0.48861645);
   Xout[5] = Yin[3] * (0.017513135) + (0.48861645);
   Xout[6] = Yin[4] * (0.0010720412);
   Xout[7] = Yin[5] * (0.0010720412);
   Xout[8] = Yin[6] * (0.072463767);
   Xout[9] = Yin[7] * (0.053763443) + (-0.62903227);
   Xout[10] = Yin[8] * (0.08) + (-0.96);
LAB110:

   /* Generating code for PE 0 in layer 6 */
   Xout[32]  = (float)(1.9891913) + (float)(0.25902966) * Xout[2] +
         (float)(-0.14866734) * Xout[3] + (float)(0.69369978) * Xout[4] +
         (float)(0.59796864) * Xout[5] + (float)(1.905352) * Xout[6] +
         (float)(0.59024686) * Xout[7] + (float)(-0.46074805) * Xout[8] +
         (float)(0.22325529) * Xout[9] + (float)(-0.50397146) * Xout[10];
   Xout[32] = sin( Xout[32] );

   /* Generating code for PE 1 in layer 6 */
   Xout[33]  = (float)(0.99351734) + (float)(0.0018794787) * Xout[2] +
         (float)(0.17599295) * Xout[3] + (float)(0.74349421) * Xout[4] +
         (float)(0.75331873) * Xout[5] + (float)(0.19267508) * Xout[6] +
         (float)(0.12440583) * Xout[7] + (float)(-0.68037164) * Xout[8] +
         (float)(1.7880155) * Xout[9] + (float)(1.0893797) * Xout[10];
   Xout[33] = sin( Xout[33] );

   /* Generating code for PE 2 in layer 6 */
   Xout[34]  = (float)(1.5333716) + (float)(0.012909059) * Xout[2] +
         (float)(-0.2015717) * Xout[3] + (float)(-0.14415716) * Xout[4] +
         (float)(-0.15190293) * Xout[5] + (float)(2.339062) * Xout[6] +
         (float)(1.0141227) * Xout[7] + (float)(0.31570694) * Xout[8] +
         (float)(0.23018885) * Xout[9] + (float)(0.89545405) * Xout[10];
   Xout[34] = sin( Xout[34] );

   /* Generating code for PE 3 in layer 6 */
   Xout[35]  = (float)(-2.7582524) + (float)(0.16090919) * Xout[2] +
         (float)(-0.031894822) * Xout[3] + (float)(-1.1356597) * Xout[4] +
         (float)(-1.1276008) * Xout[5] + (float)(-1.0797584) * Xout[6] +
        (float)(-0.20758329) * Xout[7] + (float)(0.75770807) * Xout[8] +
        (float)(-0.24672173) * Xout[9] + (float)(1.5730982) * Xout[10];
Xout[35] = sin( Xout[35] );

/* Generating code for PE 4 in layer 6 */
Xout[36]  = (float)(1.3704528) + (float)(0.29395503) * Xout[2] +
        (float)(0.048022617) * Xout[3] + (float)(0.026341241) * Xout[4] +
        (float)(-0.19751644) * Xout[5] + (float)(1.7419448) * Xout[6] +
        (float)(1.1130654) * Xout[7] + (float)(0.1947632) * Xout[8] +
        (float)(0.9747498) * Xout[9] + (float)(2.5286434) * Xout[10];
Xout[36] = sin( Xout[36] );

/* Generating code for PE 5 in layer 6 */
Xout[37]  = (float)(2.9007239) + (float)(-0.10008816) * Xout[2] +
        (float)(0.19985022) * Xout[3] + (float)(-0.42082521) * Xout[4] +
        (float)(-0.69367999) * Xout[5] + (float)(3.1515582) * Xout[6] +
        (float)(0.74488121) * Xout[7] + (float)(0.49066862) * Xout[8] +
        (float)(-0.67654276) * Xout[9] + (float)(0.59356135) * Xout[10];
Xout[37] = sin( Xout[37] );

/* Generating code for PE 6 in layer 6 */
Xout[38]  = (float)(-2.5495651) + (float)(0.36935723) * Xout[2] +
        (float)(0.29089579) * Xout[3] + (float)(0.18954857) * Xout[4] +
        (float)(0.55218583) * Xout[5] + (float)(-2.8681786) * Xout[6] +
        (float)(-0.69646275) * Xout[7] + (float)(-0.20402968) * Xout[8] +
        (float)(0.2989611) * Xout[9] + (float)(-0.46448216) * Xout[10];
Xout[38] = sin( Xout[38] );

/* Generating code for PE 7 in layer 6 */
Xout[39]  = (float)(3.0107756) + (float)(0.26883194) * Xout[2] +
        (float)(0.20622511) * Xout[3] + (float)(1.4737056) * Xout[4] +
        (float)(1.4811258) * Xout[5] + (float)(0.75309461) * Xout[6] +
        (float)(0.21919827) * Xout[7] + (float)(-0.99006659) * Xout[8] +
        (float)(0.20267667) * Xout[9] + (float)(-1.4848368) * Xout[10];
Xout[39] = sin( Xout[39] );

/* Generating code for PE 8 in layer 6 */
Xout[40]  = (float)(-1.7667688) + (float)(-0.23100586) * Xout[2] +
        (float)(0.087861478) * Xout[3] + (float)(-0.73744994) * Xout[4] +
        (float)(-0.57696509) * Xout[5] + (float)(-1.7044407) * Xout[6] +
        (float)(-0.76289737) * Xout[7] + (float)(0.50426245) * Xout[8] +
        (float)(-0.74255133) * Xout[9] + (float)(0.20470144) * Xout[10];
Xout[40] = sin( Xout[40] );

/* Generating code for PE 0 in layer 3 */
Xout[11]  = (float)(-0.59171832) + (float)(-0.25841382) * Xout[2] +
        (float)(-0.10193979) * Xout[3] + (float)(-0.38771304) * Xout[4] +
        (float)(-0.36103427) * Xout[5] + (float)(-0.14100474) * Xout[6] +
        (float)(-0.17154361) * Xout[7] + (float)(-0.083304361) * Xout[8] +
        (float)(-0.26697445) * Xout[9] + (float)(0.01363752) * Xout[10];
Xout[11] = 1.0 / (1.0 + exp( -Xout[11] ));

/* Generating code for PE 1 in layer 3 */
Xout[12]  = (float)(-0.59818435) + (float)(-0.31765902) * Xout[2] +
        (float)(-0.09643925) * Xout[3] + (float)(-0.39429769) * Xout[4] +
        (float)(-0.37043002) * Xout[5] + (float)(-0.2433968) * Xout[6] +
        (float)(-0.3060849) * Xout[7] + (float)(-0.085178822) * Xout[8] +
        (float)(-0.20705269) * Xout[9] + (float)(0.077033229) * Xout[10];
Xout[12] = 1.0 / (1.0 + exp( -Xout[12] ));

/* Generating code for PE 2 in layer 3 */
Xout[13]  = (float)(-0.54992479) + (float)(-0.33429259) * Xout[2] +
        (float)(-0.18581215) * Xout[3] + (float)(-0.34007192) * Xout[4] +
        (float)(-0.23614497) * Xout[5] + (float)(-0.32590267) * Xout[6] +
        (float)(-0.1999457) * Xout[7] + (float)(-0.1023683) * Xout[8] +
        (float)(-0.3822526) * Xout[9] + (float)(-0.27493697) * Xout[10];
Xout[13] = 1.0 / (1.0 + exp( -Xout[13] ));

/* Generating code for PE 3 in layer 3 */
Xout[14]  = (float)(-0.4233501) + (float)(-0.29527512) * Xout[2] +
        (float)(-0.10825488) * Xout[3] + (float)(-0.42777213) * Xout[4] +
        (float)(-0.38113615) * Xout[5] + (float)(-0.098492488) * Xout[6] +
        (float)(-0.13218151) * Xout[7] + (float)(-0.073445335) * Xout[8] +
        (float)(-0.38791505) * Xout[9] + (float)(-0.16651818) * Xout[10];
Xout[14] = 1.0 / (1.0 + exp( -Xout[14] ));

/* Generating code for PE 4 in layer 3 */
Xout[15]  = (float)(-0.56588918) + (float)(-0.34874058) * Xout[2] +
        (float)(-0.066539034) * Xout[3] + (float)(-0.2155617) * Xout[4] +
        (float)(-0.30413505) * Xout[5] + (float)(-0.33235115) * Xout[6] +
        (float)(-0.24272862) * Xout[7] + (float)(-0.11398592) * Xout[8] +
        (float)(-0.19391777) * Xout[9] + (float)(-0.22152297) * Xout[10];
Xout[15] = 1.0 / (1.0 + exp( -Xout[15] ));

/* Generating code for PE 5 in layer 3 */
Xout[16]  = (float)(-0.50571024) + (float)(-0.40245506) * Xout[2] +
        (float)(-0.19168212) * Xout[3] + (float)(-0.24933891) * Xout[4] +
        (float)(-0.38121784) * Xout[5] + (float)(-0.25221494) * Xout[6] +
        (float)(-0.29782993) * Xout[7] + (float)(-0.080991276) * Xout[8] +
        (float)(-0.19195782) * Xout[9] + (float)(-0.11386347) * Xout[10];
Xout[16] = 1.0 / (1.0 + exp( -Xout[16] ));

/* Generating code for PE 6 in layer 3 */
Xout[17]  = (float)(-0.66447902) + (float)(-0.28673106) * Xout[2] +
        (float)(-0.14274742) * Xout[3] + (float)(-0.35041946) * Xout[4] +
        (float)(-0.2129717) * Xout[5] + (float)(-0.16149865) * Xout[6] +
        (float)(-0.17189832) * Xout[7] + (float)(-0.28935298) * Xout[8] +
        (float)(-0.15965633) * Xout[9] + (float)(-0.21273263) * Xout[10];
Xout[17] = 1.0 / (1.0 + exp( -Xout[17] ));

/* Generating code for PE 7 in layer 3 */
Xout[18]  = (float)(-0.54722929) + (float)(-0.36162347) * Xout[2] +
        (float)(-0.077566765) * Xout[3] + (float)(-0.36210597) * Xout[4] +
        (float)(-0.24634342) * Xout[5] + (float)(-0.2961491) * Xout[6] +
        (float)(-0.28150618) * Xout[7] + (float)(-0.086343057) * Xout[8] +
        (float)(-0.34502378) * Xout[9] + (float)(-0.18249743) * Xout[10];
Xout[18] = 1.0 / (1.0 + exp( -Xout[18] ));

/* Generating code for PE 8 in layer 3 */
Xout[19]  = (float)(-0.56934851) + (float)(-0.25411004) * Xout[2] +
        (float)(-0.1108318) * Xout[3] + (float)(-0.23197328) * Xout[4] +
        (float)(-0.13497426) * Xout[5] + (float)(-0.31694621) * Xout[6] +
        (float)(-0.11645628) * Xout[7] + (float)(-0.17641704) * Xout[8] +
        (float)(-0.353908) * Xout[9] + (float)(-0.26638937) * Xout[10];
Xout[19] = 1.0 / (1.0 + exp( -Xout[19] ));

/* Generating code for PE 0 in layer 4 */
Xout[20]  = (float)(-1.3726151) + (float)(-0.41776484) * Xout[11] +
        (float)(-0.346385) * Xout[12] + (float)(-0.46596813) * Xout[13] +
        (float)(-0.39206481) * Xout[14] + (float)(-0.44167396) * Xout[15] +
        (float)(-0.51491529) * Xout[16] + (float)(-0.4681803) * Xout[17] +
        (float)(-0.37968048) * Xout[18] + (float)(-0.63580382) * Xout[19] +
        (float)(0.31559098) * Xout[32] + (float)(0.46526456) * Xout[33] +
        (float)(-0.28111437) * Xout[34] + (float)(-1.1874872) * Xout[35] +
        (float)(-0.29991758) * Xout[36] + (float)(-0.74366003) * Xout[37] +
        (float)(0.32427502) * Xout[38] + (float)(1.4315212) * Xout[39] +
        (float)(-0.3796894) * Xout[40];
Xout[20] = 1.0 / (1.0 + exp( -Xout[20] ));

/* Generating code for PE 1 in layer 4 */
Xout[21]  = (float)(-0.67152935) + (float)(-0.32002193) * Xout[11] +
        (float)(-0.37309644) * Xout[12] + (float)(-0.35036573) * Xout[13] +
        (float)(-0.37239575) * Xout[14] + (float)(-0.41986287) * Xout[15] +
        (float)(-0.39671329) * Xout[16] + (float)(-0.24793507) * Xout[17] +
        (float)(-0.47973272) * Xout[18] + (float)(-0.42119896) * Xout[19] +
        (float)(-0.48646638) * Xout[32] + (float)(-0.27633816) * Xout[33] +
        (float)(-0.24991515) * Xout[34] + (float)(0.34119672) * Xout[35] +
        (float)(0.34748143) * Xout[36] + (float)(-0.38430738) * Xout[37] +
        (float)(0.53581536) * Xout[38] + (float)(0.041380573) * Xout[39] +
        (float)(0.33520049) * Xout[40];
Xout[21] = 1.0 / (1.0 + exp( -Xout[21] ));

/* Generating code for PE 2 in layer 4 */
Xout[22]  = (float)(0.42855689) + (float)(-0.12155396) * Xout[11] +
        (float)(-0.11398869) * Xout[12] + (float)(-0.29105589) * Xout[13] +
        (float)(-0.075611442) * Xout[14] + (float)(-0.085338771) * Xout[15] +
        (float)(-0.18273053) * Xout[16] + (float)(-0.13361581) * Xout[17] +
        (float)(-0.12762055) * Xout[18] + (float)(-0.1904092) * Xout[19] +
        (float)(-0.72726929) * Xout[32] + (float)(0.20671731) * Xout[33] +
        (float)(-0.37889093) * Xout[34] + (float)(0.47247827) * Xout[35] +
        (float)(-0.34235227) * Xout[36] + (float)(-1.0097913) * Xout[37] +
        (float)(0.94047534) * Xout[38] + (float)(-0.61915809) * Xout[39] +
        (float)(0.48552111) * Xout[40];
Xout[22] = 1.0 / (1.0 + exp( -Xout[22] ));

/* Generating code for PE 3 in layer 4 */
Xout[23]  = (float)(0.5780493) + (float)(0.16924731) * Xout[11] +
        (float)(-0.010042203) * Xout[12] + (float)(0.12102904) * Xout[13] +
        (float)(0.25738901) * Xout[14] + (float)(0.022943784) * Xout[15] +
        (float)(0.060082499) * Xout[16] + (float)(0.062955208) * Xout[17] +
        (float)(0.10988119) * Xout[18] + (float)(0.1715506) * Xout[19] +
        (float)(-0.13189773) * Xout[32] + (float)(1.1378285) * Xout[33] +
        (float)(-0.41327384) * Xout[34] + (float)(0.17059267) * Xout[35] +
        (float)(0.20763989) * Xout[36] + (float)(-1.2418939) * Xout[37] +
        (float)(0.92584532) * Xout[38] + (float)(-0.30488595) * Xout[39] +
        (float)(-0.14476408) * Xout[40];
Xout[23] = 1.0 / (1.0 + exp( -Xout[23] ));
```

```
/* Generating code for PE 4 in layer 4 */
Xout[24]  = (float)(2.230948) + (float)(0.98530674) * Xout[11] +
        (float)(0.88652188) * Xout[12] + (float)(0.69120491) * Xout[13] +
        (float)(0.95617467) * Xout[14] + (float)(0.88430905) * Xout[15] +
        (float)(0.91747081) * Xout[16] + (float)(0.72866738) * Xout[17] +
        (float)(0.71221268) * Xout[18] + (float)(0.85560614) * Xout[19] +
        (float)(-1.1049176) * Xout[32] + (float)(-0.12195916) * Xout[33] +
        (float)(-1.0440729) * Xout[34] + (float)(0.92504197) * Xout[35] +
        (float)(-1.4652604) * Xout[36] + (float)(-1.7871473) * Xout[37] +
        (float)(1.5392815) * Xout[38] + (float)(-0.74341345) * Xout[39] +
        (float)(1.0199662) * Xout[40];
Xout[24] = 1.0 / (1.0 + exp( -Xout[24] ));

/* Generating code for PE 5 in layer 4 */
Xout[25]  = (float)(-0.27201569) + (float)(-0.40155745) * Xout[11] +
        (float)(-0.39630798) * Xout[12] + (float)(-0.28361714) * Xout[13] +
        (float)(-0.3759959) * Xout[14] + (float)(-0.40662542) * Xout[15] +
        (float)(-0.40751967) * Xout[16] + (float)(-0.28550768) * Xout[17] +
        (float)(-0.45065045) * Xout[18] + (float)(-0.33002353) * Xout[19] +
        (float)(-0.36811116) * Xout[32] + (float)(-1.0287485) * Xout[33] +
        (float)(-0.22275332) * Xout[34] + (float)(0.5022704) * Xout[35] +
        (float)(-0.22328319) * Xout[36] + (float)(-0.078291751) * Xout[37] +
        (float)(0.21420877) * Xout[38] + (float)(-0.14072652) * Xout[39] +
        (float)(0.37697724) * Xout[40];
Xout[25] = 1.0 / (1.0 + exp( -Xout[25] ));

/* Generating code for PE 6 in layer 4 */
Xout[26]  = (float)(-0.5362792) + (float)(-0.33348036) * Xout[11] +
        (float)(-0.36085957) * Xout[12] + (float)(-0.34967053) * Xout[13] +
        (float)(-0.43955398) * Xout[14] + (float)(-0.38215598) * Xout[15] +
        (float)(-0.33711249) * Xout[16] + (float)(-0.27981967) * Xout[17] +
        (float)(-0.39319891) * Xout[18] + (float)(-0.34826592) * Xout[19] +
        (float)(-0.64270186) * Xout[32] + (float)(-0.19941714) * Xout[33] +
        (float)(-0.41529059) * Xout[34] + (float)(0.65331835) * Xout[35] +
        (float)(0.26489902) * Xout[36] + (float)(-0.46446413) * Xout[37] +
        (float)(0.75526738) * Xout[38] + (float)(-0.18345118) * Xout[39] +
        (float)(0.59931636) * Xout[40];
Xout[26] = 1.0 / (1.0 + exp( -Xout[26] ));

/* Generating code for PE 7 in layer 4 */
Xout[27]  = (float)(0.79525441) + (float)(0.19139498) * Xout[11] +
        (float)(0.25460431) * Xout[12] + (float)(0.00027791786) * Xout[13] +
        (float)(0.0069038277) * Xout[14] + (float)(0.12903735) * Xout[15] +
        (float)(0.15468669) * Xout[16] + (float)(0.19565026) * Xout[17] +
        (float)(0.094200522) * Xout[18] + (float)(0.11223574) * Xout[19] +
        (float)(0.042638183) * Xout[32] + (float)(-1.0494556) * Xout[33] +
        (float)(0.12813216) * Xout[34] + (float)(0.11258129) * Xout[35] +
        (float)(-0.30966258) * Xout[36] + (float)(0.58917385) * Xout[37] +
        (float)(-0.36976752) * Xout[38] + (float)(0.49818456) * Xout[39] +
        (float)(0.29090342) * Xout[40];
Xout[27] = 1.0 / (1.0 + exp( -Xout[27] ));

/* Generating code for PE 8 in layer 4 */
Xout[28]  = (float)(-0.29091597) + (float)(-0.043789942) * Xout[11] +
        (float)(0.032273758) * Xout[12] + (float)(-0.058469154) * Xout[13] +
        (float)(-0.076893799) * Xout[14] + (float)(-0.16643593) * Xout[15] +
          (float)(-0.01500199) * Xout[16] + (float)(-0.11099851) * Xout[17] +
          (float)(-0.046452753) * Xout[18] + (float)(-0.18354578) * Xout[19] +
          (float)(0.73873228) * Xout[32] + (float)(0.32451302) * Xout[33] +
          (float)(0.18016939) * Xout[34] + (float)(-1.5481416) * Xout[35] +
          (float)(0.17768244) * Xout[36] + (float)(0.53756446) * Xout[37] +
          (float)(-0.62132692) * Xout[38] + (float)(1.1890496) * Xout[39] +
          (float)(-0.70827347) * Xout[40];
   Xout[28] = 1.0 / (1.0 + exp( -Xout[28] ));

   /* Generating code for PE 0 in layer 5 */
   Xout[29]  = (float)(0.086300224) * Xout[20] +
          (float)(0.13250527) * Xout[21] + (float)(0.040108185) * Xout[22] +
          (float)(-0.088430524) * Xout[23] + (float)(0.11007355) * Xout[24] +
          (float)(0.1762621) * Xout[25] + (float)(0.14621736) * Xout[26] +
          (float)(0.31158632) * Xout[27] + (float)(0.18325169) * Xout[28];

   /* Generating code for PE 1 in layer 5 */
   Xout[30]  = (float)(0.21551111) * Xout[20] +
          (float)(-0.013122663) * Xout[21] + (float)(-0.042349834) * Xout[22] +
          (float)(0.10091929) * Xout[23] + (float)(0.14890747) * Xout[24] +
          (float)(-0.018832628) * Xout[25] + (float)(-0.028144082) * Xout[26] +
          (float)(0.062060647) * Xout[27] + (float)(0.28739673) * Xout[28];

   /* Generating code for PE 2 in layer 5 */
   Xout[31]  = (float)(-0.042698815) * Xout[20] +
          (float)(0.069258742) * Xout[21] + (float)(0.21870285) * Xout[22] +
          (float)(0.17737268) * Xout[23] + (float)(0.15863417) * Xout[24] +
          (float)(-0.0071050669) * Xout[25] + (float)(0.11469036) * Xout[26] +
          (float)(0.025598887) * Xout[27] + (float)(0.053718165) * Xout[28];

   /* De-scale and write output from network */
   Yout[0] = Xout[29] * (30.999999) + (5.5);
   Yout[1] = Xout[30] * (499.99999) + (-100);
   Yout[2] = Xout[31] * (8.4133335) + (-1.9236667);
   return( 0 );
}

/********
 * Function ghousenn
 ********/

void /* cdecl */ ghousenn(argc, argv)
int               argc;           /* input argument count          */
ITEM              argv[];   /* input argument data vector          */
{

          ITEM    Item1[1];
          int     Ctr;
          float   Yin[9], Yout[3];

          if (argc != 1)
                  /* ERROR !!!  incorrect number of arguments      */
                  ;
          else
                  for (Ctr=1; Ctr<=9; ++Ctr)
                    {
```

```
                getarray("Yin", Ctr, 1, &Item1[0]);
                if (Item1[0].type == DINT)
                        Yin[Ctr-1] = Item1[0].data.intt;
                if (Item1[0].type == DNUM)
                        Yin[Ctr-1] = Item1[0].data.num;
                }

        /* Make call to Neural network subroutine */

        NN_Recall(Yin, Yout);

        /*                                          */

        for (Ctr=1; Ctr<=3; ++Ctr)
                {
                Item1[0].data.num = Yout[Ctr-1];
                Item1[0].type = DNUM;
                setarray("Yout", Ctr, 1, &Item1[0]);
                }
}


/********
 * Entry for GURU
 ********/

int /* cdecl */ centry(argc, argv)
int        argc;
ITEM       argv[];
{
        int        strcmp();

        /* Check for the GHOUSENN function                          */
        if (strcmp(argv[0].data.str, "GHOUSENN") == 0)
                ghousenn(argc, argv);

        else    /* unrecognized function name ... return ERROR !!!   */
                return (-1);

        return (0);        /* return the sucessful response          */
}
```

G.16. File COGNCTRL.BAS

File used to control the execution of the module COGNITI

```
'*** PROGRAM COGNCTRL.BAS ***

' This program is used to control execution of GURU files

DECLARE FUNCTION DosSemClear% (BYVAL P1 AS LONG)
DECLARE FUNCTION DosSemSet% (BYVAL P1 AS LONG)
DECLARE FUNCTION DosSemWait% (BYVAL P1 AS LONG, BYVAL P2 AS LONG)
DECLARE FUNCTION DosOpenSem% (SEG P1 AS LONG, BYVAL P2s AS INTEGER, BYVAL P2o AS INTEGER)
DECLARE FUNCTION DosCloseSem% (BYVAL P1 AS LONG)


DEFINT A-Z

CLS

ON SIGNAL(3) GOSUB SignalSIGTERMDetected
SIGNAL(3) ON


DIM SemHanPC1 AS LONG, SemHanPC2 AS LONG

SemPC1$ = "\SEM\SEMPC1" + CHR$(0)
SemPC2$ = "\SEM\SEMPC2" + CHR$(0)

x = DosOpenSem%(SemHanPC1, VARSEG(SemPC1$), SADD(SemPC1$))
IF x THEN PRINT "Error in COGNCTRL"
x = DosOpenSem%(SemHanPC2, VARSEG(SemPC2$), SADD(SemPC2$))
IF x THEN PRINT "Error in COGNCTRL"
x = DosSemSet%(SemHanPC2)
IF x THEN PRINT "Error in COGNCTRL"
x = DosSemClear%(SemHanPC1)
IF x THEN PRINT "Error in COGNCTRL"
x = DosSemWait%(SemHanPC2, -1)
IF x THEN PRINT "Error in COGNCTRL"

'**********

x = DosCloseSem%(SemHanPC1)
x = DosCloseSem%(SemHanPC2)


END


SignalSIGTERMDetected:
'* Subroutine executed when COGNCTRL process is killed by PAVLOV

x = DosSemClear%(SemHanPC1)
x = DosCloseSem%(SemHanPC1)
x = DosCloseSem%(SemHanPC2)

END

RETURN
```