Building Simulation

Seventh International IBPSA Conference
Rio de Janeiro, Brazil
August 13-15, 2001

BUILDINGS AIR-FLOW SIMULATIONS : AUTOMATICALLY-GENERATED ZONAL MODELS

Marjorie MUSY*, Etienne WURTZ** and Anne SERGENT**
*CERMA – Ecole d'Architecture de Nantes – 44319 Nantes
**LEPTAB – Université de La Rochelle – 17042 La Rochelle

ABSTRACT

We present an advanced formulation of zonal models for calculating indoor air temperature and flow distributions in buildings. Our modeling is based on modularity:
-the behavior of a room is represented by the connection of SPARK calculation modules
-the behavior of the building is obtained by the connection of its rooms calculation modules.
The modules represent sub-zones of the rooms or interfaces between sub-zones. There are different kinds of modules depending of the phenomena to be represented: walls, thermal plumes, jets… which are gathered in a models library. We developed a model-generating tool called GenSPARK that automatically assembles the appropriate modules to construct a zonal model of an entire building. Then, SPARK solves the resulting set of equations and the airflow and temperature distribution in the building are obtained. In this article, we describe our formulation of zonal models and show how GenSPARK works. We also give results that are compared to experimental ones.

INTRODUCTION

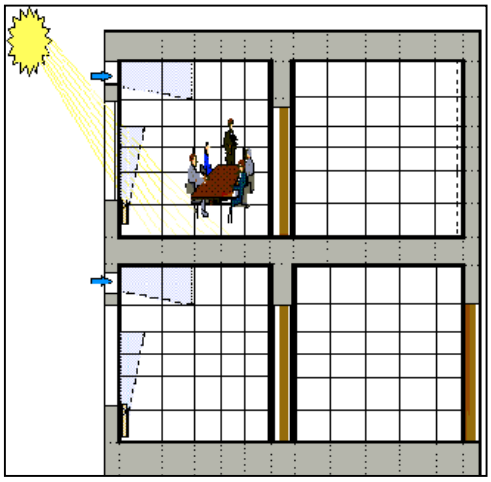
Zonal models are based on an approach that is intermediate between single-air-node models, which give no information about air flow patterns, and CFD models, which give detailed temperature and flow distributions but are extremely compute intensive. Such intermediate models execute much faster than CFD calculations but give more accurate heat transfers than the single-node approach and provide temperature and flow distributions that are detailed enough to predict thermal comfort. In zonal models, the inside of a room is divided into a small number of zones or "cells", which are usually rectangular parallelepipeds. Mass balance and heat balance equations are applied to the cells and the exchanges are calculated between them. The solution of the resulting set of coupled equations gives the airflow and temperature distribution in the room.
The first zonal models were based on fixed airflow directions and application of specific flow laws for plumes, jets and boundary layers. Allard and Inard [1] have reviewed them. Assuming fixed airflow directions obviously restricts the field of application of these models. Other models ([2], [3]) made the inter-cell airflow rates a function of the pressure distribution. It has been shown that this approach cannot correctly represent driving flows [4]. Later, hybrid models were formulated that applied specific laws for driving flows and used a power-law pressure distribution everywhere else [5]. However, these models are applicable only to a few simple configurations.
To extend the field of application of zonal models, we have reformulated them so that the behavior of a room is represented by the connection of calculation modules. We also have created modules to describe the building walls. All these modular elements form the *models library*. This work has been done in the SPARK modular environment ([6],[7]). By assembling the appropriate modules, a zonal model of an entire building can be constructed. We have developed a model-generating tool called GenSPARK that automates this process. Once the zonal model has been constructed, SPARK solves the resulting set of equations and the airflow and temperature distribution in the building is obtained.
In the following sections we first describe our formulation of zonal models. Then we show how GenSPARK works. Finally, we give examples of the kind of configurations we are able to analyze.

NEW FORMULATION OF THE ZONAL METHOD

Unlike the pressure-flow network models used to calculate airflow between rooms, the zonal method is based on heat and mass balance equations in macroscopic volumes called *cells*. Added to this is a relationship between inter-cell mass flow and pressure difference. A systematic attempt to use the zonal method with power law equations to describe the air flow for various configurations in two dimensions is described in [4]; in that work convergence problems were encountered and the results did not agree well with measurements. We retain the power-law formulation in the present work but apply it only to cells where no driven flow is expected to occur. In other cells, we use laws that are specific to the flow to be represented.
Our reformulation of zonal models is based on a new way of partitioning the rooms that accounts for different boundary conditions and the presence and location of devices such as heaters and diffusers. Associated with this partitioning are *cells* and *interfaces*. Cells are the parallelepiped-shaped subzones into which the room is divided. An interface is a rectangular surface that separates adjacent cells. There are different types of cells depending on the airflow within the cell and different

types of interfaces depending on the type of cells they separate. Cells are described by balance equations and state laws. Interfaces are described by heat and mass flow equations. The equations for different types of cells and interfaces form the models library.

Fig. 1: Partitioning of a building into cells



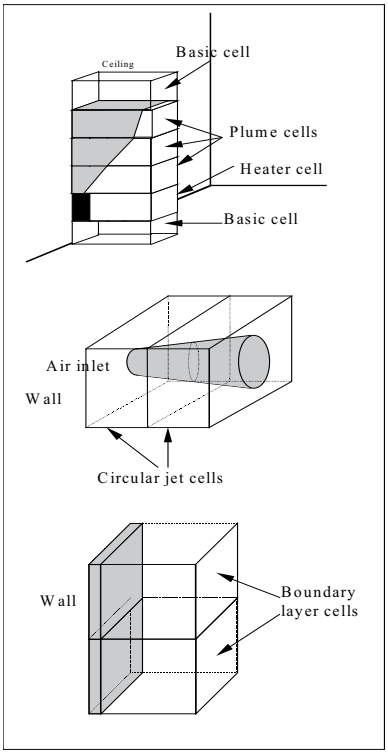
We do not allow the partitioning of rooms to change during a simulation. Therefore, the partitioning must be able to handle dynamic problems in which air flows, such as plumes, may appear and disappear as time progresses. We handle this by including intermittent flows of this type in one or more larger cells (fig. 1). For example, a *plume cell* contains two subcells, one containing air belonging to the plume itself and one containing air from the surroundings. This kind of partitioning is also done for *jet cells* and *boundary layer cells* (fig. 2). Note in this figure that the heater and its associated plume are contained in rectangular parallelepipeds of the same size and shape stacked on top of each other. This simplifies partitioning of a room that has different flow types since it makes it easy to "line up" cells.

Basic cells represent subzones where no driving flow occurs. As mentioned before, cells with driving flows are divided into two subcells. For a plume cell, for example, the basic cell equations are used for the subcell that contains the air entrained from the surroundings, whereas the air in the subcell that contains the plume itself is described using a different set of equations that are specific to the plume. This approach is also used in selecting the equations for the interfaces of cells that contain driving flows.

The equations for basic cells and basic interfaces are given in [8]. How plume and heated cells and interfaces are constructed starting from these basic elements is described too. Jets, boundary layers cells and interfaces are built following the same scheme: addition of specific equations to basic ones.

Note that it is not necessary to know *a priori* how high the plume will reach (or how long is the jet): the simulation will automatically convert plume cells (or jet cells) to basic cells beyond the dissipation point. Identically, it determine wether the boundary layers are developing up or down. Our model is also able to handle situations in which a boundary layer conflict occurs, for example when an upward airflow boundary layer meets a downward airflow boundary layer.

Fig. 2: Jets, plumes and boundary layers are included in several cells.



Wall conduction is represented by *wall interfaces* that contain a conduction model and a convection model (the convective heat transfer coefficient depends on the surface-to-air temperature difference). Walton's method [9] is used to calculate long- and short-wave radiant exchanges among the inside room surfaces. In this method each room surface is assumed to radiate to a fictitious surface whose area, emissivity and temperature gives about the same heat transfer from the room surface as in the actual multi-surface case. The advantage of the method is that it considerably reduces the number of interchange equations.

AUTOMATICALLY GENERATED SIMULATIONS

The Simulation Problem Analysis and Research Kernel (SPARK [6],[7]) is a modular environment that automates writing code for systems of non-linear equations. It was developed for building science but is applicable to other fields. It is related to simulation environments like TK!*Solver* [10], TRNSYS [11], CLIM2000 [12], IDA [13], and Allan [14].

Some key features of SPARK are:

- It has a front end that allows the user to build complex simulations by connecting smaller elements that are atomic classes (single equations) or macro-classes (equation subsystems).
- Using graph-theoretic techniques, it reduces the size of the equations system by automatically determining a small set of iteration variables for which the other unknowns can be solved. This step can be viewed as "smart" elimination of variables. SPARK's Newton-Raphson solver works on the reduced equation set and, after convergence, the remaining unknowns are solved for.
- Its output is a C++ program that is automatically compiled and executed. This program accepts user-specified input at run time and is calculationally efficient because it iterates on a reduced set of variables.
- Passing from a simulation problem to a design problem (i.e., having unknowns become inputs and inputs become unknowns) is simply a matter of keyword exchange in SPARK.
- It is possible to handle transient problems by adding time integrator classes.
- The use of a pre-processor allows generating automatically from equations expressed symbolically in Maple [7].

Implementation of zonal models in SPARK is straightforward. The main SPARK classes correspond to the different types of cells and interfaces that were described above. The work consists of creating atomic classes corresponding to each individual equation (this is done using the MAPLE symbolic processing program) and combining (usually by hand) them into macro-classes that correspond to cells and interfaces. Classes are stored in a model library for later use.

To build a zonal model for an entire room, the classes will be instantiated as many times as needed to define the simulation. For example, if a 3-D room is divided into 8 parallelepipeds (2 in each of the x, y and z directions), there will be 8 *cell* objects and 36 *interface* objects (12 zone-to-zone interfaces and 24 zone-to-surface interfaces). In the general case, if the x, y and z axes are divided into L, M and N sections, respectively, there will be $L\ M\ N$ zone objects and $3(L\ M + M\ N + L\ N)$ interface objects. After instantiation, the *cell* and *interface* objects are linked, i.e., the variables shared by objects are identified. This is done in a *room* object. Several *room* objects can be created and linked to build a simulation for a whole building. The *room* objects and their linked variables are stored in a file that specifies the overall problem and its inputs.

Manually connecting the cells and interfaces to create the room objects and then manually connecting the room objects to construct the whole-building simulation is laborious and error prone. Therefore, a tool called GenSPARK has been written that automates the connection process. GenSPARK uses a *knowledge base* and *rules*. The *knowledge base* contains the procedures for connecting each kind of cell and interface to their neighbors. It is directly linked to the model library. To introduce a new model in the library, the corresponding cells and interfaces have to be created and entries made in the *knowledge base*. The *rules* are guidelines for writing new cell and interface objects. They are needed so that GenSPARK understands how to handle new objects, which is key to practical use of zonal models.

The *knowledge base* is formed by observing how variables are shared between neighboring objects. An example is shown in fig. 3, where a basic cell and basic interface are connected. Variables are classified according to the way they are shared among objects (remember that objects are sets of equations). *Global variable,* as c_p, the heat capacity of air, have the same value in all the cells and interfaces. In fig. 3, the global variables are shown in the cross-hatched rectangles. *Local variables* take different values depending on the cell, the interface and the time step. In fig.3, we see that some variables, such as Tdot (temperature variation in the cell) and T_hist (temperature of the cell at previous time step) are linked to none of the neighboring elements. We call these *purely local variables*. Some cell variables, such as T, P and ρ (temperature, pression and air density in the cell) (fig. 3), are linked to all neighboring interfaces, and some interface variables, such as Q (energy flux) and M (mass flow rate) (fig. 3), are linked to both neighboring cells. These are called *octopus variables*. This family is divided in two groups: *perfect octopus variables* and *imperfect octopus variables*. Perfect octopus variables are linked to neighboring objects independently of the nature of these objects. An example is the basic-cell variable T (fig. 3.) Imperfect octopus variables do not link to certain types of objects. An example is basic-cell variable P. It cannot be linked to a wall interface, because the wall model does not use pressure. An *oriented variable* is a cell variable that links to only one of the neighboring interfaces, or an interface variable that links to only one of the neighboring cells. Examples in fig. 3 are cell variables Q_{Est} and M_{Est} and interface variables T_I and ρ_I. In general, oriented cell variables link to octopus interface variables and oriented interface variables link to octopus cell variables. The neighbor-only rule states that a cell variable can only be connected to the cell's neighboring interfaces, and an interface variable can only be connected to the cells adjacent to the interface. The names of octopus cell and interface variables must obey some rules. To understand these rules, we describe what GenSPARK does when it encounters an octopus variable. If it is a cell variable, called *var* for example, GenSPARK tries to link it to a variable called *var1* in the cell's east, top and north interfaces and to a variable called *var2* in the cell's west, bottom and south interfaces. This means that an

Fig. 3: Linkage basic cells and interfaces.

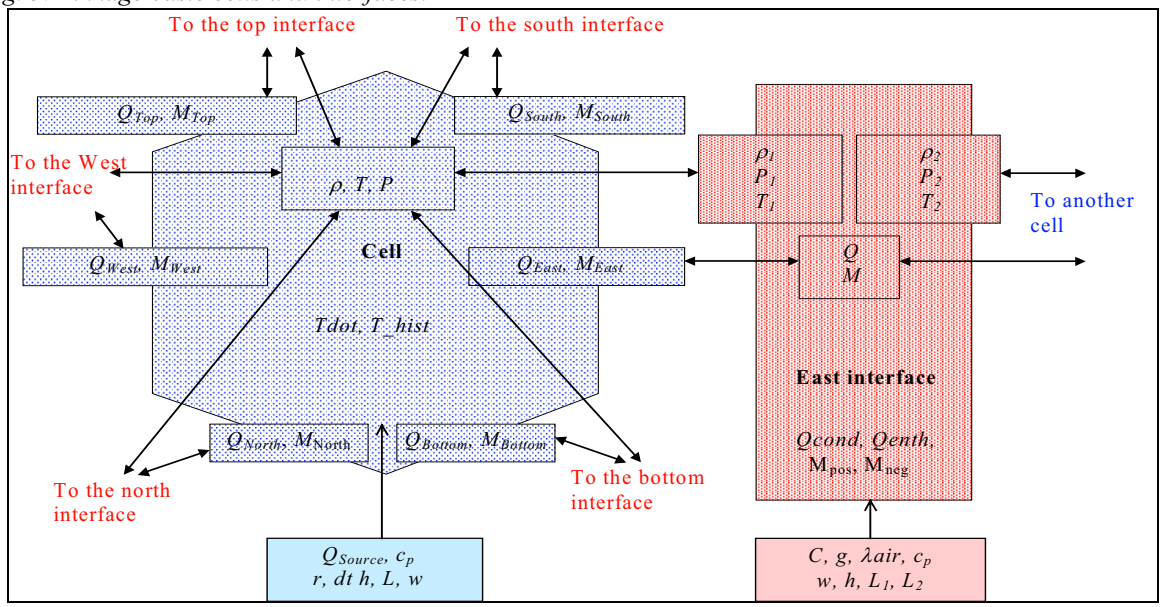


Fig. 4: Example of rooms' connection

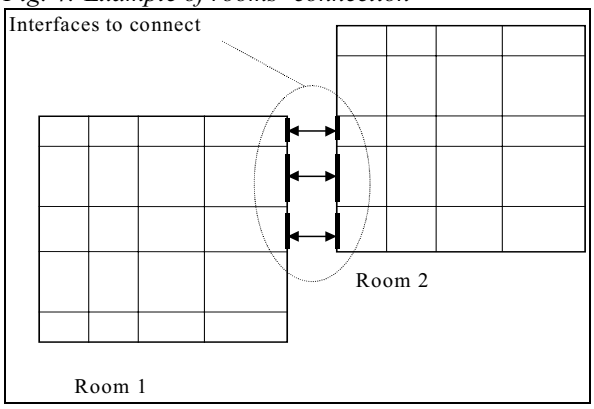
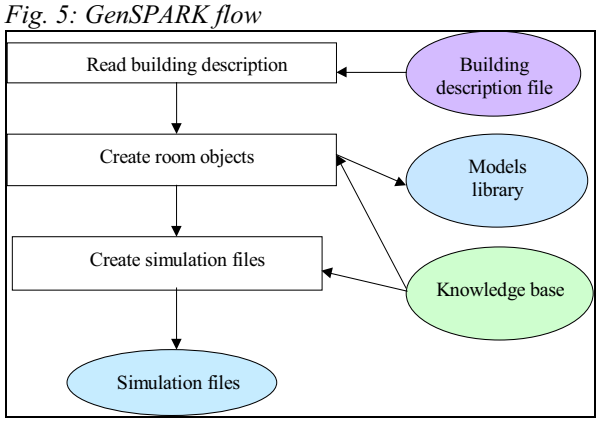


Fig. 5: GenSPARK flow



octopus cell variable, such as *T* (fig. 3), must have two corresponding interface variables, such as *T1* and *T2* that are *oriented variables*. Similarly, GenSPARK tries to link *octopus interface variables* to neighboring cell variables that have the same name followed by the extension East, West, North, South, Top or Bottom, depending of the orientation of the interface.

The user input to GenSPARK is a building description that includes the number of rooms. Then, rooms are described successively: the number of cells in each room, name of the *cells* and *interfaces* to use in the X, Y and Z directions are given. Lastly, in order to connect rooms to the neighboring ones, the interfaces to assembly must be specified (see fig. 4).
GenSPARK then creates room objects and the simulation files. The process is shown in fig. 5.
GenSPARK first creates a room object for each room based on the number of cells in each room and the names of the cell and interface objects. The cell and interface objects used in the room object are declared. Then the variables for each cell and interface are declared and linked to their parent objects, and, if they are octopus variables, they are linked to their neighbors. At this stage, GenSPARK uses the knowledge base to determine how to link the objects specified by the user. To create the simulation file GenSPARK needs to know how to connect the boundary interfaces between adjacent rooms; this information comes from the building description file. Partitioning is adjusted so that the boundary cells in a room have the same height and width as the neighboring boundary cells in the adjacent room. Finally, the user specifies any remaining input values and launches SPARK, which solves the set of equations created by GenSPARK.

VALIDATION OF THE MODELS

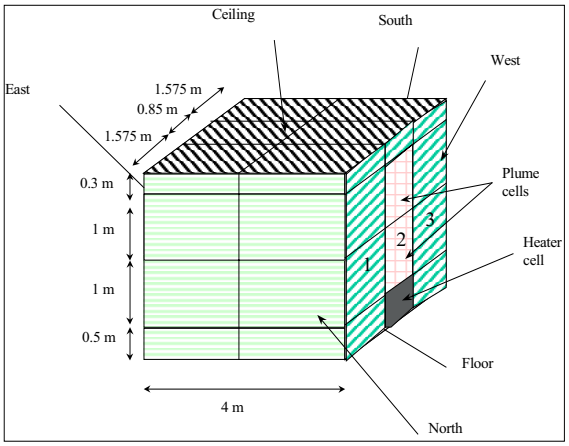
In this section we compare the plume, boundary layer and jet model results with measurements in different test rooms.

Plume model

The simulation results were compared with measurements from the CETIAT experimental room [15] equipped with an electrical heater. Fig.6 shows

the room geometry, the partitioning into cells, and the location of the heater and its associated plume cells. The inputs to the simulation are the heater power and the inside surface temperatures (Table 1).
Fig. 7 compares simulations done with and without explicit modeling of the plume caused by the heater. We note that when the plume is modeled there is a strong upward airflow along the right-hand wall and significant temperature stratification.

Fig. 6: The CETIAT experimental room partitioned into 24 cells.



When the plume is not modeled there is upward airflow in the right half of the room but it is not concentrated along the right-hand wall, and there is little temperature stratification.
For the case that the plume is explicitly modeled, the air temperature increases by about 7°C from floor to ceiling, indicating that there is significant stratification. Fig. 8 compares measured and simulated air temperature vs. height along the vertical centerline of the room. The simulation agrees with measurements to within 1°C in the three bottom layers and to within 1.3°C in the top layer. Note from this figure that the ceiling to floor *air* temperature difference is about 7°C whereas the ceiling to floor *surface* temperature difference is only 23.6-21.1 = 2.5°C. This example shows that it is important to model the plume explicitly.

Fig 8: Measured temperature along the vertical centerline of the room and results of the simulation when the plume is modeled. Calculated temperatures are assumed to be uniform in each cell and are shown as horizontal lines. Centerline temperatures have been calculated as the mean of the Sect. 1 and Sect. 2 temperatures (right-hand plot fig. 7).

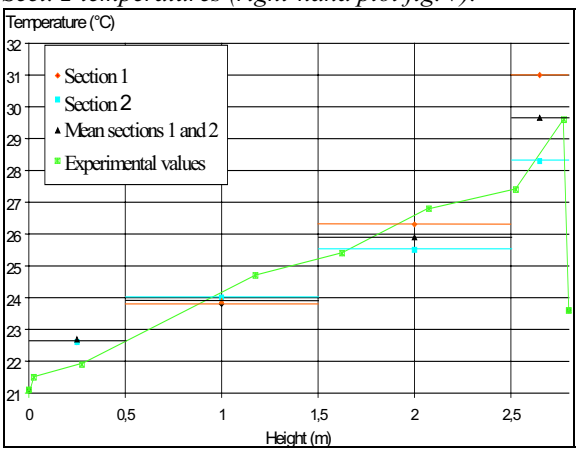
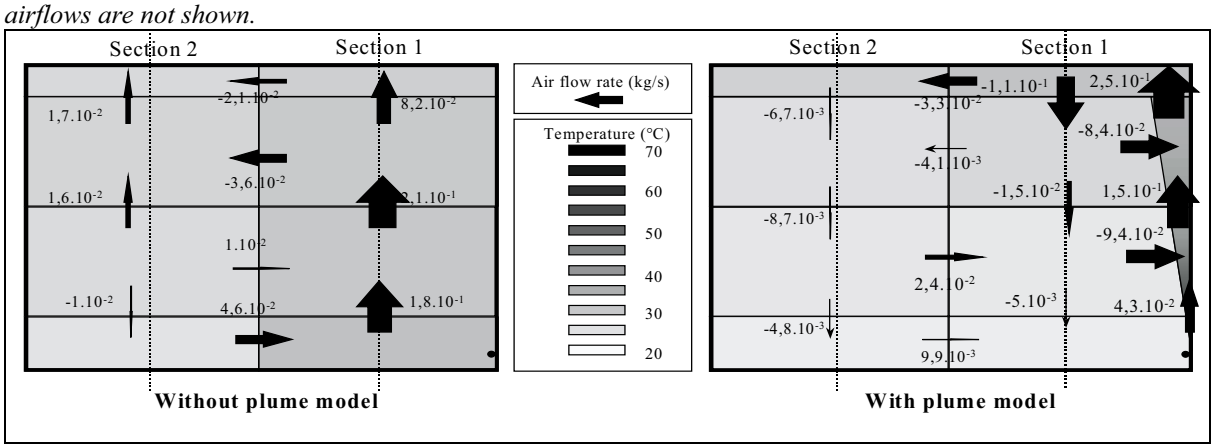


Table 1. Input values for inside surface temperature and heater power in the CETIAT experimental room.

	Temperature(°C)	Power (W)
Floor	21.1	
Ceiling	23.6	
East	21.0	
South	21.8	
North	22.3	
West_1	19.2	
West_2	27.2	
West_3	20	
Pelec		1500

Fig. 7: Calculated temperature and airflow values in the East-West vertical plane passing through the heater (fig. 6). Left-hand plot: results of the calculation done without modeling the plume produced by the heater. Right-hand plot: the plume has been explicitly modeled Shading shows the different temperature levels. Airflow rate is proportional to arrow thickness. Note that mass flows in some cells do not balance since the South-North airflows are not shown.



Boundary layer model

The simulation results were compared with measurements from the CSTB EREDIS experimental room, which is 3.6m square and 2.5m high. All surfaces have the same homogeneous temperature except the window, which has three different temperature points. Fig. 9 compares simulation results and measurements.

INCORPORERINCORPORERROn the vertical centerline of the room, calculated and measured temperatures agree to within less than 1°C. Simulations without modeling the boundary layer (not shown) show much poorer agreement (the temperature difference reaches 2.5°C in the center of the room), showing that it is important to properly model the boundary layer. On the contrary, we showed [8] that putting boundary layer models where wall to air temperature difference is too low can lead to a wrong air flow pattern in the room.

Jet model

We modeled the CETIAT experimental room, MINIBAT [16], which had a horizontal ventilation air jet just below the ceiling. Fig. 10 shows the room geometry and partitioning into cells, and the location of the ventilation air inlet and exhaust openings. Inputs to the simulation are the inside surface temperatures, the entering airflow rate, and a pollutant source (SF6) in the middle of the room (Table 3). Flow was modeled for a fluid consisting of two gaseous species, air and pollutant.

Fig. 11 compares the simulation results with measurements. Temperatures agree to within 1°C in the two middle layers of cells. There is even better agreement in the jet and in the bottom layer (fig. 12). The calculated pollutant concentration is too low except near the source. This may be due to the fact that the model assumes that the pollutant is distributed into the room only by molecular diffusion; it neglects turbulent diffusion, which in some locations in the room can be large compared to molecular diffusion.

Fig. 9: Boundary layer modeling: simulation results (left-hand plot) and comparison with measurements (right-hand plot) for the CSTB EREDIS test room.

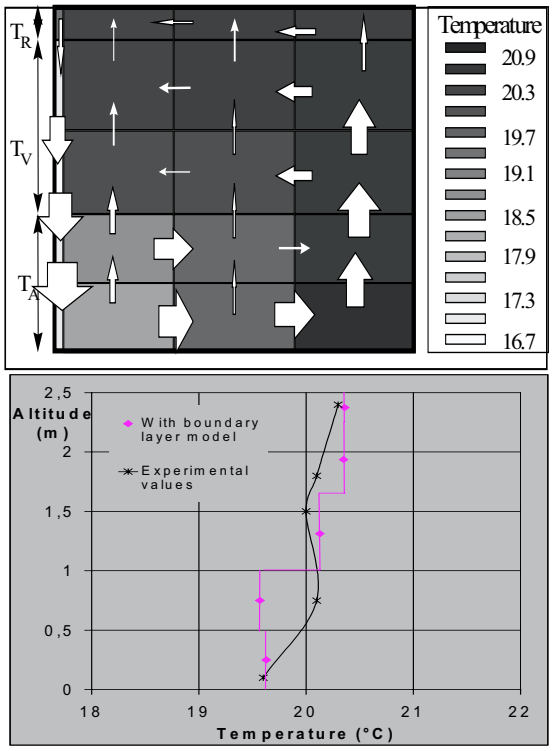


Table 3. Input values for inside surface temperatures, inlet air temperature and flow rate, and pollutant source strength for the MINIBAT test room.

	Temperature(°C)	Air flow rate (g/s)
Floor	19.4	
Ceiling	21.	
East	20.	
South	18.9	
North	20.	
West	19.9	
Inlet	33.5	8.32
Pollutant		0.002943

Figure 10: The MINIBAT test room partitioned into 4x3x4 = 48 cells. Three of these are jet cells.

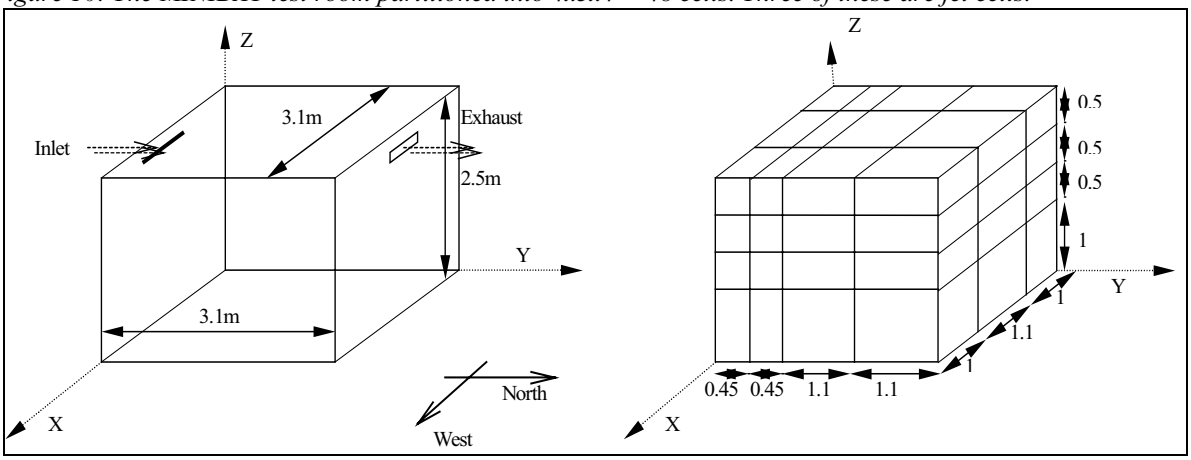


Figure 11: MINIBAT test room: calculated air temperature and air flow rate (left-hand) and pollutant concentration (right-hand) in the South-North vertical plane passing through the ventilation air jet. Shading shows air temperature (left-hand) or pollutant density (mg/m^3) (right-hand).

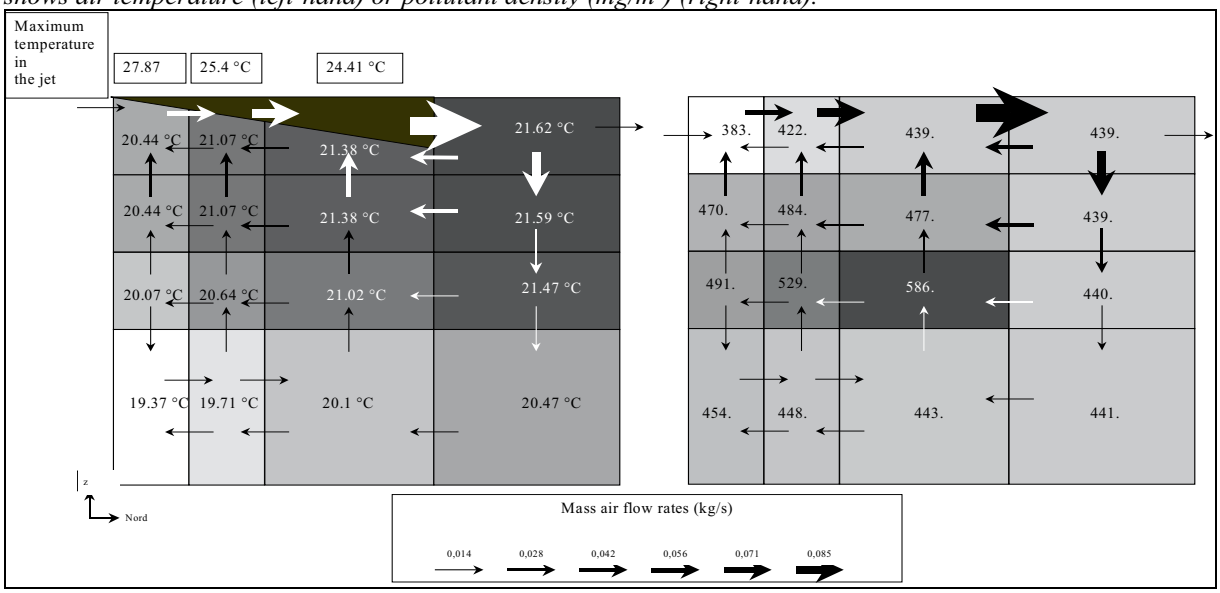
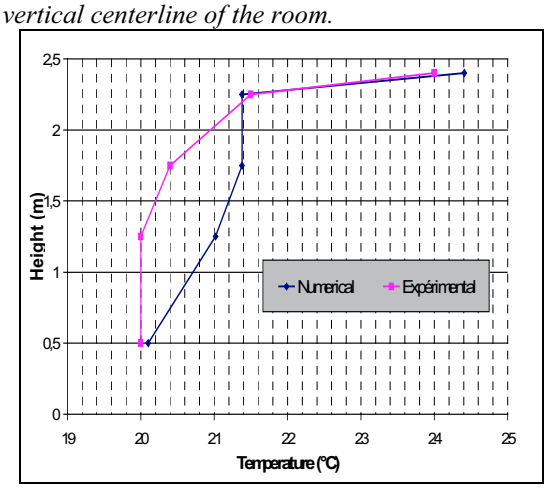


Figure 12: MINIBAT test room: calculated and measured air temperature vs. height along the vertical centerline of the room.



APPLICATIONS

To illustrate the possibilities offered by our new approach to zonal modelling, we used GenSPARK to automatically generate a SPARK model for a complex multiroom configuration. We simulated a four-room building whose vertical section is shown in fig. 13. Rooms 1, 2 and 3 are identical, have heaters and ventilation inlets, and are separated from Room 4, an atrium, by open doorways. The model contains a total of 108 cells (18 in each of Rooms 1, 2 and 3 and 54 in Room 4). Rooms 1, 2 and 3 have a heater cell and a plume cell. All other cells are basic cells. The inputs are the outside surface temperatures of the walls (15°C), the inlet air temperature (15°C), the power of the heaters (1500W), and inlet airflow rates (0.01kg/s). Fig. 13 shows the calculation results. The atrium has about 4°C of stratification. The airflow through the doorways is significant, being about 6 times higher than the ventilation air flow rate. Overall, the calculation appears to be qualitatively correct. Note that due to the height of the atrium there are certainly boundary layers along its walls, so the calculation could be improved by adding boundary layer cells.

CONCLUSIONS

We have shown how a zonal model can be used to calculate air temperature and flow distribution for single- and multi-room configurations. The results indicate that dividing room air volume into between 24 and 45 cells is sufficiently fine-grained to allow thermal comfort to be determined.

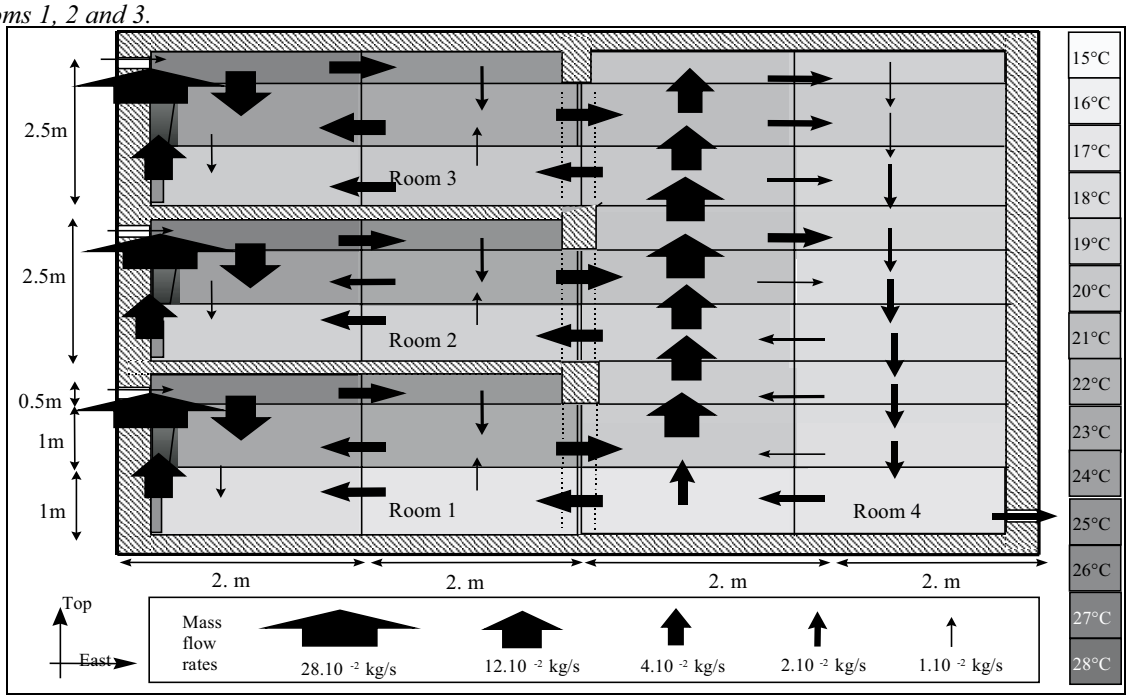
Different levels of automation were used to generate models in the SPARK simulation environment:

- Generating equation objects using symbolic processing,
- Automatic generation of SPARK simulation files for a whole building.

This makes it possible to generate a complete simulation of a building given only the building geometry and properties of the heating and cooling systems. This also allowed us to generate zonal models applied to rooms with geometry more complex than the examples considered in this paper [8]. The comparisons made so far with measurements have been encouraging.

It is important to note that the simulation approach that we have developed can be applied to all types of thermal and air-flow problems. It is not restricted to simple configurations nor is it required that the flow distribution be known *a priori*. We use a workstation for the calculation but this zonal model could be adapted to PC, using PC version of SPARK.

Figure 13: A four-room building, with atrium, that is 8 m wide, 7.5 m high and 3 m deep. Shown are calculated temperatures and airflow rates in the East-West vertical plane passing through the heaters at the bottom left of Rooms 1, 2 and 3.



REFERENCES

1. Allard F. and Inard, C., "Natural and mixed convection in rooms: prediction of thermal stratification and heat transfer by zonal models" Proceedings of International Symposium of Room Air Convection and Ventilation Effectiveness, pp. 335-342. Tokyo, 1992.

2. Bouia H., nJModélisation simplifiée d'écoulements de convection mixte internes: Application aux échanges thermo-aérauliques dans les locauxnj Ph.D. Thesis, University of Poitiers, France, 1993.

3. Wurtz E., Nataf J.M., Winkelmann F., njTwo- and three-dimensional natural and mixed convection simulation using modular zonal modelsnj IJHMT n°42, pp.923-940, 1998.

4. Wurtz E., Musy M., Allard F., njModélisation d'un panache d'émetteur de chaleur pour le logiciel de simulation énergétique des bâtiments orienté objet SPARKnj Int. J. of Th. Sc., Vol 39(3), pp. 433-441, 2000.

5. Bouia H., and Dalacieux P., njSimplified modeling of air movements inside dwelling roomnj Building Simulation'91, IBPSA, Sophia-Antipolis, France 1991.

6. Sowell E., Buhl F., Nataf J.M., njObject-oriented programming, equation-based submodels, and system reduction in SPARKnj Building Simulation'89, IBPSA, pp. 141-146, Vancouver, Canada, 1989.

7. Nataf J.M., Winkelmann F., njSymbolic modeling in building energy simulationnj Lawrence Berkeley National Laboratory report LBL-35439, Berkeley, CA 94720. (Apr. 1994).

8. Musy M., njGénération automatique de modèles zonaux pour l'étude du comportement thermo-aéraulique des bâtimentsnj Ph. D. Thesis, University of La Rochelle, France, 210p,1999.

9. Walton G.N., njA new algorithm for radiant exchange in room loads calculationsnj ASHRAE Transactions, vol 86, Part II, pp. 190-208, 1980.

10. Konopasek M., Jarayaman S., njConstraint and declarative languages for engineering applications: the tk!solver contributionnj Proceedings of the IEEE, 73(12), Dec. 1985.

11. Solar Energy Laboratory, njTRNSYS, a transient simulation programnj University of Wisconsin-Madison, 1988.

12. Bonneau D., Covalet D., Gautier D., Rongère F.X, njManuel de prise en main de CLIM2000, version 0.0nj 1989.

13. P. Sahlin, njIDA, A Modeling and Simulation Environment for Building Applicationsnj Institute of Applied Mathematics, Stockholm, Sweden, 1998.

14. Gaz de France, njManuel de référence, Allan 2.4.1.njGaz de France Technical Report, June 1992.

15. During H., njConsommation energetique et confort thermique des locaux chauffés: Approche par les modèles zonauxnj Ph.D. Thesis, INSA de Lyon, France, 1998.

16. Castanet S., njContribution à l'étude de la ventilation et de la qualité de l'air intérieur des locauxnj Ph.D. Thesis, INSA de Lyon, France 1998.