3

MODELLING MOISTURE MOVEMENT

The reason for modelling

Computer models of moisture movement are used in three different contexts in this thesis. A simple isothermal model is used as a companion to the chamber experiments, allowing testing of theory against measurement, and extrapolation of short experiments to longer times. A more complicated model, which includes heat transfer, is used to predict the performance of alternative air conditioning systems that co-operate with climate buffering materials. This model is also used to explain the idea that moisture movement through outer walls is driven mainly by the relative humidity gradient.

There are several published programs for modelling heat and moisture movement through walls. Inexplicably, none of them include the indoor climate in the calculation. The person running the program sets the indoor temperature and RH as boundary conditions for what happens in the wall. Since my purpose is to investigate how walls influence the indoor climate I had to write the programs myself.

Another reason for modelling as well as measuring is that there is some doubt about the correctness of predicting rapidly changing conditions using absorption and permeability properties of materials obtained under equilibrium or steady state measurements. The results from a computer model which is fed the static properties of the materials can be compared with the dynamic experimental results. Alternatively, the dynamic material properties can be obtained by optimising the fit between theory and experiment.

The description of the programs begins with the isothermal model and then explains the extensions needed to cope with a temperature gradient.

Basic concepts

The computer simulation uses the general method known as Finite Element Analysis. The experimental system: weather, room, and wall, is divided into slices. Each slice is regarded as homogeneous. Water is transferred between the slices according to set rules for a brief time. After each cycle of transfers the state of each slice is updated and then the process is repeated. The division into slices is made so fine grained that the step in water content from one slice to the next is small compared with the total variation over all the slices. The time interval between iterations must also be small, to ensure that the slice which is subjected to the greatest change in water content is not forced to an extreme water content where the rules of exchange between slices are no longer valid.

The exchange of water between slices is based on Fick's first law: that the flux of moisture through a porous material is proportional to the concentration gradient. As the moisture penetrates the pores under this driving force it is absorbed into the surrounding material according to the equilibrium described by the sorption curve.

In this simple version of the model, the equilibration of water with the material is assumed to be instantaneous in each slice, so the dispersion of water into the heart of the material is limited by the resistance to diffusion. There is some evidence that this is a

naive assumption. Wood, in particular, is thought to absorb water rather slowly from the vapour diffusing through the cell cavities (14).

The model includes the inside climate as an active participant in the game. The inside climate can generate a moisture flux but not a relative humidity. That is forbidden because the porous wall contributes to the moisture content of the room, so the RH cannot be a quantity defined by the person running the program. In this way the model follows the design of the climate chamber described in the previous chapter.

The published descriptions of models that have been used as inspiration and guidance for this, much simplified version, are listed in references (15,16,20).

The choice of language and programming style

The computer model is intended to provide insight into the physical processes at work and to allow "what if ..." calculations with different material constants, as well as speculative changes to the algorithms to see if one can achieve a better fit to the chamber results. It is not meant to be a tool for designing walls, in competition with established heavyweight programs such as MATCH and WUFI.

It is important that such a program should be designed from the start to be naturally immune to side effects whereby an alteration to one action has unintended, and maybe never discovered, effects on other calculations. One way of improving the resistance to side effects is to use simple algorithms, as close to the underlying physics as possible. Another technique for achieving natural immunity to side effects is object oriented programming. This is a concept which will not be discussed in more detail here, except to explain that the underlying idea is to encapsulate the processes and data for each layer in the model wall within an object which cannot directly be altered from other parts of the program. Objects can only influence each other by sending messages through the filter of an interface. There is no single array of data values which can be freely updated by various procedures scattered around in the code.

These considerations led to the choice of Java as the programming language.

The program is kept simple and easy to change as ideas develop. This means that there is no elegant mathematics to speed up the calculations, and no convergence tests. The process plods along at the simplest possible level, made possible by the speed of modern computers. The check against wild values forming transiently in a slice because of too long a time between iterations or too thin a slice is left to the person running the program: the slice dimensions and iteration time are varied and the results are checked for consistency over a range of values.

This means making many trial runs. For this reason, and also because of the many graphs of experimental runs, a graphing program was written specially for this project.

The basic physics of the model

The numerical ingredients in the model are the diffusion coefficient used in Fick's first law and some approximation to the slope of the sorption curve, for calculating the water absorbed at each point through the material. Both of these constants are known to vary with water content. However, the main purpose of this program is to model the interior walls of houses, which are usually operating in the middle of the RH range. I have therefore assumed constant coefficients.

The question of what really is the potential which drives moisture diffusion is ducked by making the whole process isothermal, so any unit will do.

Finally, one more constant is necessary. This controls the rate of entry of water vapour into the wall. The diffusion rate through the wall surface is calculated from a fixed surface resistance.

A systematic description of the program

The wall

The wall is divided into slices parallel to the surface. The movement of water between the slices is controlled by the concentration difference (the vapour pressure difference is the quantity actually used) between adjacent slices operating against the resistance to diffusion of the material. This movement is summed over a short time, just a few seconds. Water enters the first slice from the room at a rate determined by the surface diffusion constant. At the same time some water is being lost to the next slice, according to the concentration difference between the two slices. The action then moves down through the layers to the back of the wall. Then the state of each slice is updated by taking the net gain of water, adding it to the water already present and equilibrating this water with the material, according to the water capacity. This generates a new equilibrium RH which, together with the temperature, defines a new water vapour concentration which is used in the next iteration.

That is all there is to the physics. The program structure is also simple. Each slice is an object which contains its own constants (diffusion and sorption coefficients, density) and state (water content) in variables which, once the starting state has been constructed, can only be changed by that object's own functions.

There are just two important functions belonging to each slice. One looks at the state of the adjacent slice 'upstream' (towards the room) and the state of its own slice and calculates how much water will move across the boundary between them during the specified time interval. It then sends back to the adjacent slice a message stating the amount of water it has taken. The other function receives this message from the next slice downstream and then updates its own slice's state.

The transport of water between the slices is repeatedly calculated, using only two constants: the diffusion constant through the material and the water capacity of the material (which is just the absorption isotherm simplified to a straight line).

The wall is divided into as many slices as are needed to give a stable calculation. The calculation always goes through the wall from the inside towards the outside.

The weather outside and the climate in the room

The weather and the room are represented by specialised slices. Two further constants are needed here: the leak rate of outside air into the room and the surface resistance between room and wall. The surface wall slice notices that the upstream slice is the room and uses the surface resistance, which is a property of the room, to calculate the water flow, rather than its own material diffusion constant. It is merely a container for water diffusing in through the surface resistance. The function that updates the state of the wall slice does, however, use the static properties of the wall material, such as its water capacity. This anomalous surface slice is made thin, so that the distribution of water with depth is hardly distorted by the odd behaviour of the surface slice. The diffusion coefficient of the wall material is first used when moving water from the surface slice to the next slice.

The order of the calculations

In this cascade of calculations each object looks 'upstream' at the previous object to decide how much water will stream into itself. Alternatively, each object could look downstream at the next object and decide how much water to pump out into its neighbour. The two treatments are, I think, equivalent. I have chosen the first because the message about how much water is received by each object is sent back after the focus has passed from the upstream object. It can then update itself without risk that the result can affect this round of the calculation.

The cascade starts again with the room leaking in some weather. The weather then updates itself by looking at the time and deciding where it is in its endless cycle. It could also easily be programmed to look up its values in a file, such as a test reference year.

The room has an extra function which allows it to add water to itself, or take some away, according to a user set cycle time and amplitude, precisely as in the experimental chamber.

The program controller and the starting data

The controlling object calls on each slice in turn to activate its diffusion calculation (described mathematically later). The last action of this diffusion function is to send back to the previous slice the amount of water taken, thus causing the previous slice to update itself. The program controller takes care of the return to the top of the cascade. It knows the calculation time interval, when to write intermediate results to a file and when to stop.

The user loads the material data into the wall slices and instructs the calculation controlling object how frequently to run. This is achieved through a user interface which is also programmed in Java, using a RAD (Rapid Application Development) program called JBuilder (Inprise corp., formerly Borland). This part of the program manages a database of starting data which is loaded into the objects before the controller is let loose on them. It is a complicated and uninteresting part of the program which is not described here.

The advantage of this way of programming is that one can make a new type of object called wood, for example, which appears to its neighbours just like the wall slices described above but has a completely different way of updating itself.

The program in more detail

This section deals with the next level of detail in the programming: the class definitions and the calculation algorithms. The program code is listed in an appendix. The scientifically important code, which is very brief, is marked by bold type. A narrative explanation of these short code sections is given below. The formulæ are written out without use of symbols. This is partly because the expressions are not manipulated further, so there is no real need for symbols. Furthermore, the program code uses names rather than single character symbols, so the equations listed here are easily identified in the program code.

The class *Slice* defines variables common to all types of slice: weather, room and wall. This class also declares, but does not define, the two calculating functions, called *diffuse* and *update*. *Slice* is an abstract class which cannot be used directly but can only serve as the base for deriving more specialised classes. The various classes derived from *Slice* are described later.

The starting states of the slices of the model building are constructed from a file of starting data. The program then calls the *update* function for the weather. Then it calls the *diffuse* function of each slice in turn, passing as parameters the address of the previous slice and the time interval. The *diffuse* function works out the water movement into the slice from the previous slice. When it has finished it calls the *update* function of the previous slice, informing it how much water it has taken in.

After the last slice the program starts again with the room slice and cascades down through the wall slices. The weather is included at the end, in the case of an outside wall. This cycling continues for the preset number of hours. The state of the slices is written to a file at an adjustable interval, typically every hour.

The calculations

The description starts with the behaviour of a typical wall slice and then describes the special behaviour of the weather and room classes.

Diffuse, for a wall slice, calculates the flux streaming in from the previous slice. To do this it takes the vapour pressure of the previous slice and its own vapour pressure, thickness and diffusion coefficient. These values are inserted into Fick's first law to derive the flux into this slice. Flux into the wall from the chamber is regarded as positive.

flux = time × diffusion coefficient ×
(vapour pressure of previous slice - vapour pressure in this slice) / thickness

Diffuse finally calls the previous slice's *update* function with the flux as parameter.

The *update* function for this slice, which is called by the next slice downstream, adds to the pre-existing water content the flux which the slice received from its previous slice

and subtracts the flux lost to the slice downstream, which it has just been told about. The slice then recalculates its RH, using the water capacity:

New RH = water content/(water capacity × density × thickness).

The water capacity is in units of kg/kg/(100%RH).

The new vapour pressure, and other climatic variables, are calculated from the RH by standard climate calculations, available as static functions in the class *Cmath*. (A static function can be regarded as a utility, available to any other function in the program) and listed in normal mathematical symbols in an appendix.

This calculation set: -{ flux into this slice, update previous slice }- , cascades down from slice to slice.

This is the basic pattern. The object oriented approach ensures that the physical state of each slice can only be altered by that slice's own *update* function. Its current vapour pressure is only *shown* to the next slice and the amount of water taken from it is *reported* in a message from the next slice.

The driving force in this model is vapour pressure difference. This is not controversial because the model is isothermal. It makes no difference how the water content is expressed, because all forms: vapour pressure, concentration, relative humidity, are equivalent.

The specialised derived classes

There are different *diffuse* and *update* functions for the various sub-classes. These are explained below.

The weather slice (class *Weather*) is the start of the cascade. Its *update* function calculates the weather according to a pre-set cycle. It ignores the flux returned as the parameter. Its *diffuse* function is similar to that of the wall slice, with the surface resistance instead of the material resistance.

The room slice (class *Room*) has a *diffuse* function that calculates a constant leak rate of air from the previous slice, which is usually the weather.

water in = leakrate × iteration time × volume ×
vapour pressure recalculated to kg per cubic metre

'Water out' is calculated similarly, using the room vapour pressure. Finally:

flux = water in - water out

A further peculiarity of the room's *diffuse* function is that the room itself is equipped to generate a water vapour flux, independent of the outside weather, to imitate man-made water sources. This can be a constant or cyclic flux and is calculated much like the weather.

The wall slice (class *Wall*) has a *diffuse* function that first enquires about the type of the upstream slice. If it is of type *Room* it uses this calculation:

flux = time × vapour pressure difference × room diffusion coefficient

Notice that the path length is not used in this formula. It is built into the room slice's 'diffusion coefficient'. This flux is therefore the water passing just into the surface of the first wall slice, taking no account of slice thickness. The wall's *update* function does however use the thickness and so the water is correctly 'distributed' through the slice. (This treatment of the surface is a bit clumsy. Maybe there should be a separate *Surface* class characterised by zero water capacity, but this complicates the programming considerably and breaks the uniformity of the present structure).

The next wall slice notices that its upstream neighbour is of its own type and uses the standard calculation in the *diffuse* function:

flux = time × vapour pressure difference × diffusion coefficient / thickness

The first wall slice should be thin, because it does not contribute to the resistance of the wall.

A rapid iteration time, typically ten to thirty seconds, is needed to prevent the thin surface wall slice from acquiring too large a change of RH in one iteration. The program does not check that each movement of water is infinitesimal, so each construction must be checked for stability by running it with different time intervals. The simplicity of the program is compensated by the speed of modern computers.

The program that models combined heat and moisture transport

A computer model that includes heat flow is used in chapter 6 to predict the performance of a museum storeroom. Heat moves through the wall in an exactly analagous way to moisture. Heat moves at a rate equal to the temperature gradient multiplied by a diffusion coefficient, called the thermal conductivity. After the heat has moved into a slice it reacts with the material according to the heat capacity, to cause a rise in temperature. This is programmed in a very similar way to that described earlier for moisture movement.

There is a link between heat and moisture movement. This is illustrated in figure 3.1.

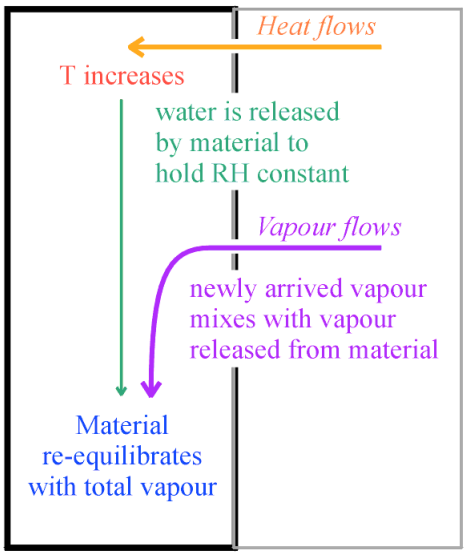

Figure 3.1 One slice of the wall, showing both heat and water vapour moving into the slice from the slice to the right. After the heat flow has raised the temperature the material releases water, to maintain a constant RH (because the RH equilibrium of absorbent materials is nearly independent of temperature, depending only on the water content). Water also moves into the slice by diffusion from the right. These two sources of water are added together and then re-equilibrated with the material to give a new value for the water content of the material, and a new value for the vapour pressure in the pores.

The explanation given here is strictly formal. There is no attempt to give a microscopic description of what is happening. The space available for mixing the vapour is the whole volume of the slice, which is also occupied by the material. This is unrealistic but makes no significant quantitative difference with absorbent materials. As explained in chapter 1, the water content of the material entirely dominates the space around it and asserts almost the same equilibrium RH over a large range of ratio of pore space to solid. This trick eliminates the need to know about the pore volume and has the programming advantage that the large mixing space reduces the chance of the RH going to extreme values.

The situation within mineral insulation, such as glass fibre, can also be modelled safely using this assumption. The ratio of absorbent material to surrounding space is now much smaller but the model is closer to the physical reality.

This model is also different from the isothermal model in other ways. It is used to simulate buildings in the landscape, so the outer surface of the wall reaches close to 100% RH. The constant water capacity has to be replaced by a proper sorption curve. The diffusion coefficient is also strongly affected by the RH at the high end. The diffusion through the surface is affected by wind speed, though this can be averaged. These complications led to restriction of the program to working with wood, with the isotherm and moisture coefficient derived from functions hard coded into the program. It was not therefore possible to use the program for the varied materials and laminates

investigated in the climate chamber. It was in fact written several years ago, using a core of methods developed by Poul Jensen (21) for investigating the drying of timber. His model had been tested against the measured drying of wood so I have re-used it, only adding routine functions for calculating the effects of leakage and of mechanical air conditioning processes.

Capillary movement of water

The two commercial programs mentioned earlier, MATCH and WUFI, include capillary movement of liquid water as a transport mechanism which dominates at high RH. The program just described does not treat capillary movement separately. It includes the measured variation with RH of the water vapour permeability in wood as an alternative way of treating the matter.

My general approach has been to use measured parameters, without enquiring at all deeply into their physical meaning. The model programs are not the main subject in this thesis. I have used them to extrapolate experimental results and to test alternative interpretations of measurements in buildings. The programs are just intended as an aid to designing further experiments and as a guide for developing measurement strategies in real buildings.