UNIVERSITE DE GENEVE

Centre universitaire d'étude des problèmes de l'énergie

ENET
97119507-1

TRNSYS compatible moist air hypocaust model

Final report

comissioned by the
Office fédéral de l'énergie, CH - 3003 Bern
(project 19507)

September 1998

Pierre HOLLMULLER
Bernard LACHAL

19, avenue de la Jonction - CH-1205 Genève - Tél 022 / 705 72 30 - Fax 022 / 705 72 00

TRNSYS compatible
moist air hypocaust model

Part 1 : Description and validation

Introduction

Preliminary work

History of this project draws back to a pilot project, also funded by the Federal Office of Energy, dealing with short term heat storage in agricultural greenhouses [4-6]. High involved humidities often induced condensation and evaporation within the hypocaust (pipe system). Sensitivity analysis therefore required a model that would take into account not only sensible but also latent heat exchanges, and allow for airflows in one or the other direction (for respect of thermal stratification during storage and withdrawal periods). Such a model had formerly been developped by Razafinjohany and Boulard [1] in the frame of a similar experience in Avignon, but happened not to be flexible in geometry outlay and input/output definition, nor to include possibilty of water infiltration - which in some cases will bee seen to play important role.

A first new development of the model was carried out within the mentioned project and allowed to reproduce general trends of condensation and evaporation patterns. A second validation work was done on a preheating system for fresh air in a residential building [8]. Monitoring could in this case be reproduced with an extremly good precision, but in absence of latent exchanges.

Objectives

Since this had not yet been the case, main purpose of this project was to render the module compatible with the TRNSYS environment commonly used for simulation of energy sytems, in particular the possibility to link back and forth surface conditions with other modules that include thermal capacities, like buildings, and therefore to stand timestep iteration (one of the main features the TRNSYS environment uses for system convergence). Besides, further validation of the model was also to be done on yet another winter preheating and summer cooling system in a commercial building [9].

Report structure

Body of this report will present essential features of the model (of which detailed description, including mathematical, is to be found in the user's manual). Focus will then be set on validation work for the three analysed systems.

Overview of the model

General features

The modeled hypocaust consists of a series of parallel tubes within a rectangular block of soil, swept by a flow of humid air. Flexible geometry description allows for inhomogenous soils, diverse border conditions, as well as use of symetries or pattern repetitions (run-time economy). Direction of the airflow can be controled (stratification in case of heat storage). Energy and mass balance within the tubes account for sensible as well as latent heat exchanges between airflux and tubes, frictional losses, diffusion into surrounding soil, as well as water infiltration and flow along the tubes.

Sensible and latent heat exchange

Heart of the model and worthwile mentioning here is the energy exchange between air and tube, while three dimensional diffusive heat transfer within soil is of classical, explicit type.

Sensible exchange depends on the air-tube temperature difference and the exchange coefficent, latter one depending in turn on flowrate. Cutting short on more sophisticated use of dimensionless analysis, we used for this coefficient a heuristical form of linear dependence on air velocity (see mathematical description in the user's manual), as developped for large plane exchange surfaces [2] and proved to be consistent in case of tubes [6].

Latent heat exchange is based on the Lewis approach [3], which considers preceeding sensible heat exchange to be an air mass exchange between the airflux and a superficial air layer on tube surface, at latters temperature and saturated in humidity. This exchange of moist air conveys a vapor transfer (condensation or evaporation, according to sign of transfer) which readily computes from difference in humidity ratios (see mathematical description in the user's manual).

Input/output configuration

Written as a TRNSYS compatible FORTRAN subroutine the model allows for modular use. Passed arguments include variable inputs (airflow, inlet temperature and humidity, air pressure, surface temperatures or heat gains) and fixed parameters that relate to either coupling with other modules (surface resistances and coupling modes) or simulation timestep and precision. Additional internal parameters (geometry, soil properties, initialisation, etc.) are furthermore provided by a parameter file. Retrieved arguments consist of basic outputs (outlet temperature and humidity, total sensible and latent heat rates, reciprocal border temperatures or heat gains) as well as a variety of optional outputs (including temperatures, energy rates and waterflows for specified node clusters).

Coupling to other modules

In the case of simple links, output of some other module is connected to input of this module (e.g. ambient temperature connected to surface temperature of hypocaust, or outlet fan temperature connected to inlet temperature of hypocaust), without evolution of other module having any influence on present one. In the case of reciprocal linking, a second link goes back from hypocaust to other module, so that evolution of each module affects evolution of the other one (eg. diffusing energy rate from building ground is connected to energy rate entering at hypocaust surface, and reciprocaly border temperature of hypocaust is connected back to outside surface temperature of building ground).

In latter case convergence method by timestep iteration integrated into TRNSYS will ensure correspondance of energy rates flowing out of one module and into the other one (at condition of forth and back coupling to be each one of different type : energy rate or temperature). In order to maintain possibilities as broad as possible, each one of the two linking modes are retained : 1) input to surface is energy rate, reciprocal output is temperature ; 2) input to surface is temperature, reciprocal output is energy rate.

Checking of latter linking method as well as of proper energy and mass balance calculation within the module was done on an academic example presented in the user's manual.

Validation on « Geoser » short term heat storage system in agricultural greenhouse

System description

In the « Geoser » experiment [4-6] excessive solar heat gains of two greenhouses (100 m2 ground surface each) in the Rhône valley were stored for reduction of fuel-consumption during heating period (usualy nightly). In the first case storage occured in a watertank (via an air/water heat exchanger), in the second case in the underlying and sidely isolated soil (via a burried pipe system), while a third greenhouse with mere fuel heating system served as a comparison.

The buried pipe heat storage system stands 80 cm underneath the greenhouse and consists of 24 tubes (external diameter : 16 cm ; length : 11 m ; mean axial distance : 42 cm ; total exchange surface, without distribution and collector pipes : 120 m^2) swept by a variable airflow (0 - 7000 m^3/h) runing in one or the other direction during charge and discharge.

Typical daily operation is shown on figure 1 : during lower setpoint period at night air from greenhouse is blown through pipes, takes up heat from soil and thus lowers demand to auxilary heating system. When difference between air and soil becomes too small for dischage (at setpoint change) system becomes inactive. As greenhouse temperature rises with solar radiation, air is blown again through pipes, this time for heat storage, in reversed direction, and until air-soil temperature difference becomes too small again. Same cycle repeats from sunset on.

During nightly discharge air heated along the tube evaporates free water in tubes, while storage period starts up with condensation of very moist greenhouse air, followed by evaporation again as air humidity drops down. Water balance shows a deficit (more evaporation than condensation) of 15 lit for this one day and of 800 lit over the 18 months of monitoring. This can most probably be explained by the use of a fog system at 2m height within the greenhouse, which sprays droplets of ~80 microns into air. One clearly observes effect of fog system on air humidity and temperature, especially at 1m height, when droplets reach wet bulb temperature and start to pump evaporation energy in air rather than in water and air (cf. detailed studies on atomization and spray phenomena [7,8]). According to data on sedimentation velocity and lifetime (~20 m/s, respectively 3-40 s depending on air humidity, ibid.) it is quite plausible that part of the droplets are swept by airflow into tubes.

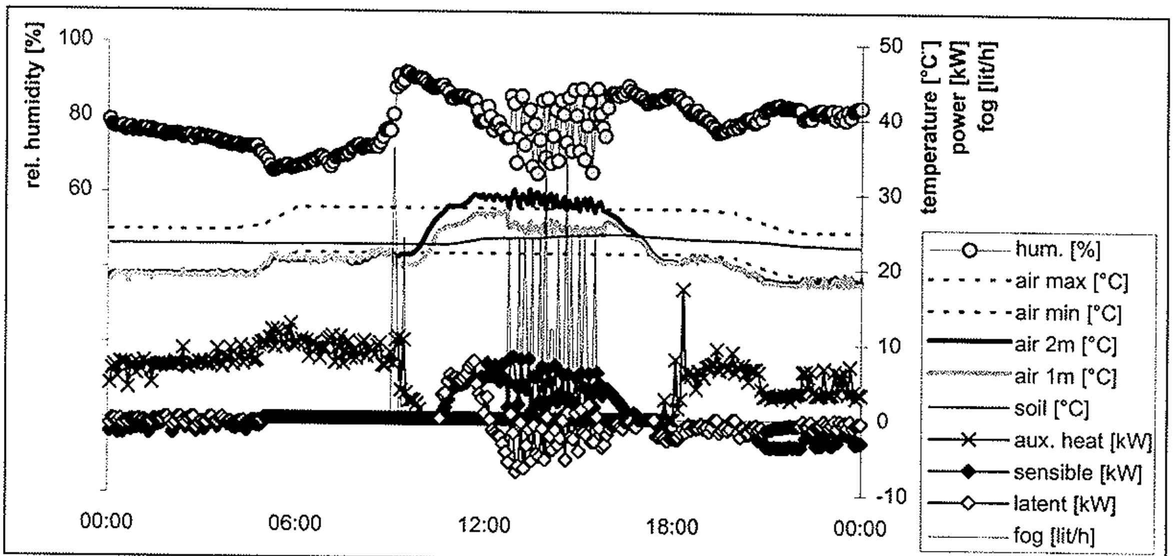


Fig 1 : Operation of « Geoser » short term heat storage system on May 9th 1994 : conditions in greenhouse (air temperature, setpoints and humidity) and in soil (temperature near pipe), energy furnished to soil (sensible and latent), auxilary heat demand and fog system.

Validation

Border and input conditions for modelling of this system are top and bottom temperatures (40 cm above and 70 cm beneath pipes) averaged over two monitoring points each, as well as measured inlet temperature and humidity. Initialisation is done on first six monts of monitoring and 12 following months are used for comparison of simulation and monitoring (1/4/94 - 30/3/95).

As shown on figures 2 to 5, a first simulation under preceeding conditions and without water infiltration very well reproduces sensible heat exchanges, as well in hourly as in integrated timestep. General trends of evaporation and condensation are beeing reproduced to some extent during first weeks (important measured latent exchanges, possibly due to use of only one air direction) but sytematically fall short in quantity and completely disappear at later periods. During first weeks one observes in particular (figure 3, broken line) that simulated latent exchange stops when condensed water has been evaporated again, which confirms the hypothesis of water infiltration during monitoring.

In a second simulation the measured daily water deficit is beeing infiltrated in constant hourly rate on the whole pipe surface, in which case evaporation (but not so condensation) is very well reproduced over the whole 12 month period.

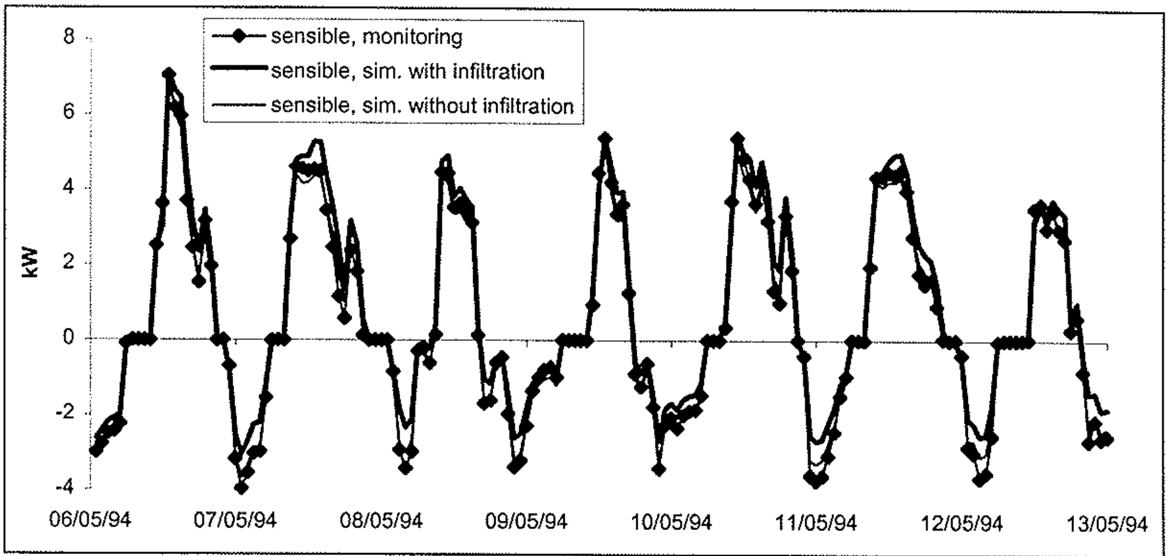


Fig 2 : Sensible energy rate (furnished to soil) for « Geoser » experiment : monitored and simulated data in hourly timestep from May 6th to 16th of 1994 (corresponding to 6th week of comparison period).

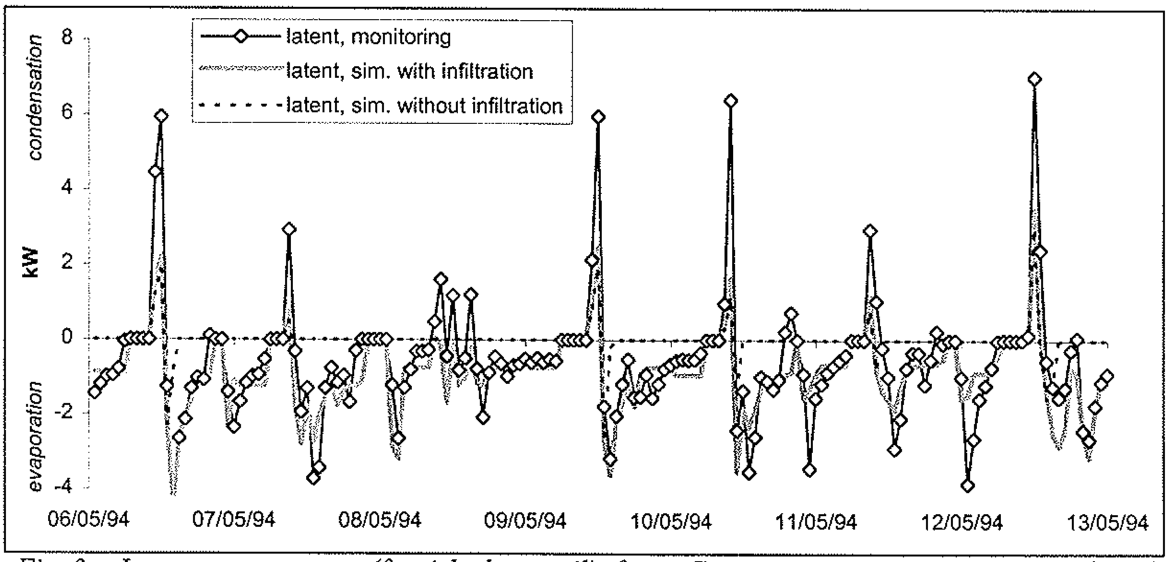


Fig 3 : Latent energy rate (furnished to soil) for « Geoser » experiment : monitored and simulated data in hourly timestep from May 6th to 16th of 1994 (corresponding to 6th week of comparison period).

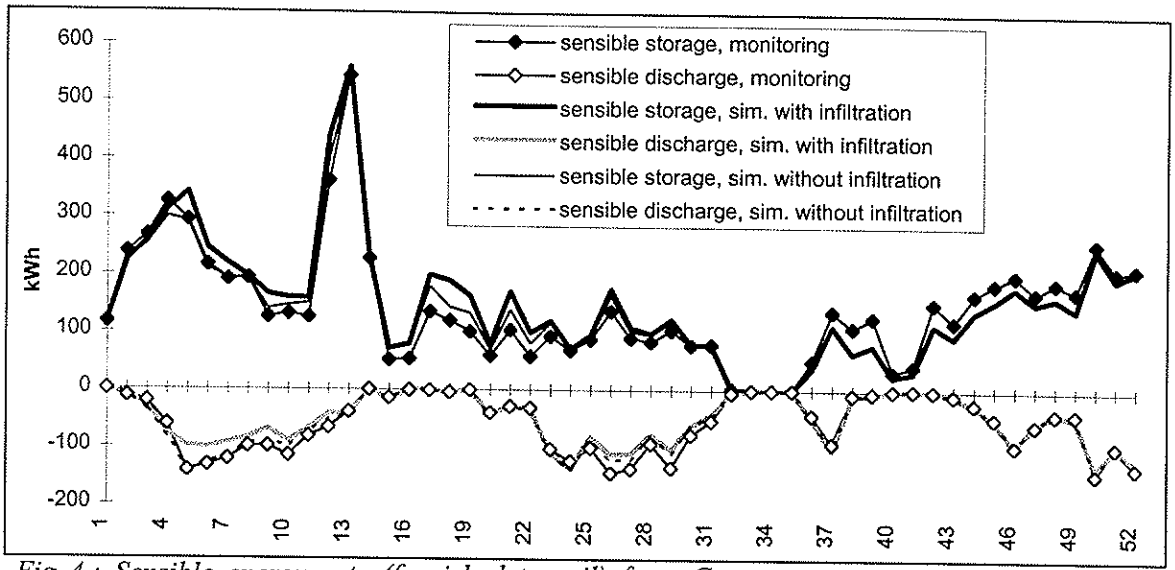


Fig 4 : Sensible energy rate (furnished to soil) for « Geoser » experiment : monitored and simulated data in weekly timestep over one year (April 1st to March 31st 1995).

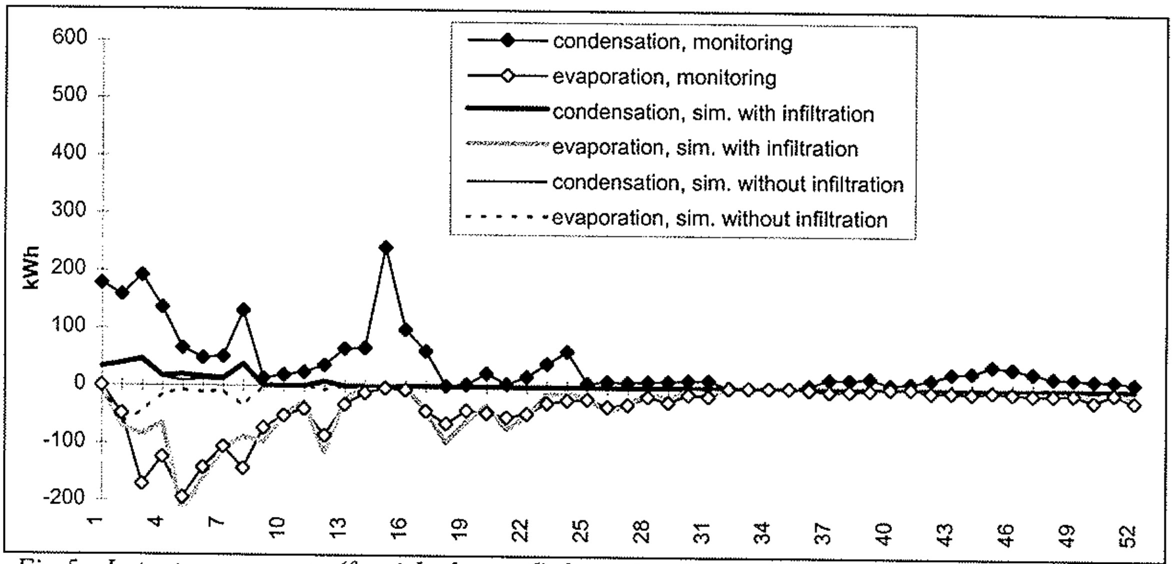


Fig 5 : Latent energy rate (furnished to soil) for « Geoser » experiment : monitored and simulated data in weekly timestep over one year (April 1st to March 31st 1995).

Validation on « Caroubier » fresh air preheating system in a residential home

System description

The « Caroubier » residential home in Geneva [9] is equiped with a complex preheating ventilation system in which, depending on solar radiation, fresh air passes either a solar roof or a burried pipe system before going through an exhaust air heat exchanger.

The hypocaust consists of 49 pipes (diameter : 12.5 cm ; lenghth : 50 m ; axial distance : 30 cm ; total pipe exchange surface : 980 m^2) divided into two similar branches that are running at 50 cm beneath an underground parking, approximatly 10cm above underground water level. Monitoring of the system over a 20 day winter period yields a two step airflow (1100 or 1400 m^3/h, for one of the two branches). No latent exchanges is detected over this period.

Validation

Simulation of the system was to be carried out on a one year period. Initialisation and input conditions therfore were taken from standard yearly meteorological data, combined with monitored values of Geneva area during 3 months preceding monitoring period. Border condition at top was air in the parking lot (supposed to fluctuate between 7°C at end of February and 23°C at end of August), while temperature 2.5 m beneath pipes was supposed to be constantly at 15°C because of moving underground water.

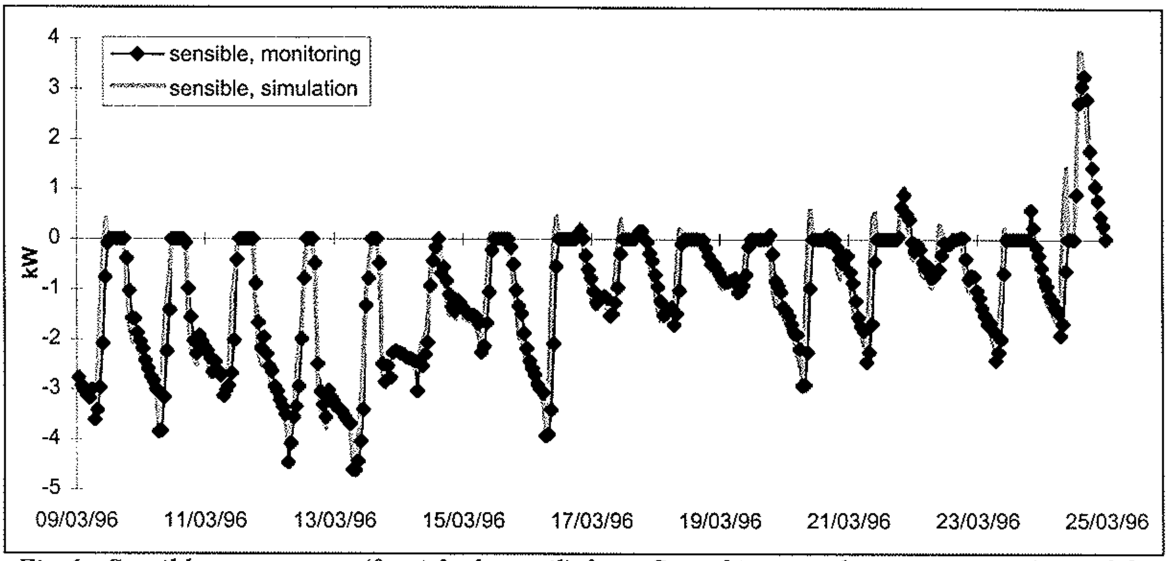


Fig 6 : Sensible energy rate (furnished to soil) for « Caroubier » preheating system (one of the two branches) : monitored and simulated data in hourly timestep (March 9th to 24th of 1996).

Correspondence of simulated and measured data over short monitoring period (figure 6) results to be excellent despite of little constraints coming from monitoring. This fact is probably not so much inherent to the absence of latent exchanges than to oversizing of the system (all possible energy is exchanged over the pipe distance) and the presence of a constant heat source immediately underneath the pipes.

Validation on « Schwerzenbacherhof » fresh air preheating and cooling system in a commercial and industrial building

System description

Further checking was done by validating the model against monitoring data from the hypocaust system of the « Schwerzenbacherhof » office and commercial building [10], of which a set of hourly data over a one year period was handed out by the Federal Office of Energy.

The burried pipe system consists of 43 pipes (external diameter : 25 cm ; length : 23 m ; mean axial distance : 116 cm ; total exchange surface, including distribution and collector pipes : 900 m^2) running at 75 cm beneath the second basement of the building (~6 m beneath ground surface). A varying airflux during office hours (6'000 - 12'000 m^3/h in winter, 18'000 m^3/h in summer) yields winter preheating as well as summer cooling of the building.

Although not expicitly mentioned in the handed out report, infiltration of underground water seems to be at work all along the year : comparaison of mesured enthalpy balance with sensible heat exchange yields evaporation within the tubes all over the year (cf. figure 8 and 9), without any water deposit by condensation ever. More detailed analysis of the hypocaust energy balance however shows some inconsitency with other from monitoring derived data, which are heat diffusion from building and towards deep ground[1], as well as capacitive heat gains and losses[2]. As a matter of fact one observes resulting balance default to have same magnitude and dynamic as latent exchanges . This very strong corelation most probably draws back on some systematic error of inlet/outlet humidity sensors and a priori invalidates any conclusion on water flow (infiltration and evaporation).

Validation

Border and input conditions for simulation of this system were upper soil temperature (75 cm above pipe bed, as averaged over 3 distinct monitoring points), lower soil temperature (600 cm beneath pipe bed, as given by a unique monitoring point), as well as inlet airflow, temperature and humidity. Initalisation was done by adding input data of last six months prior to effective yearly data.

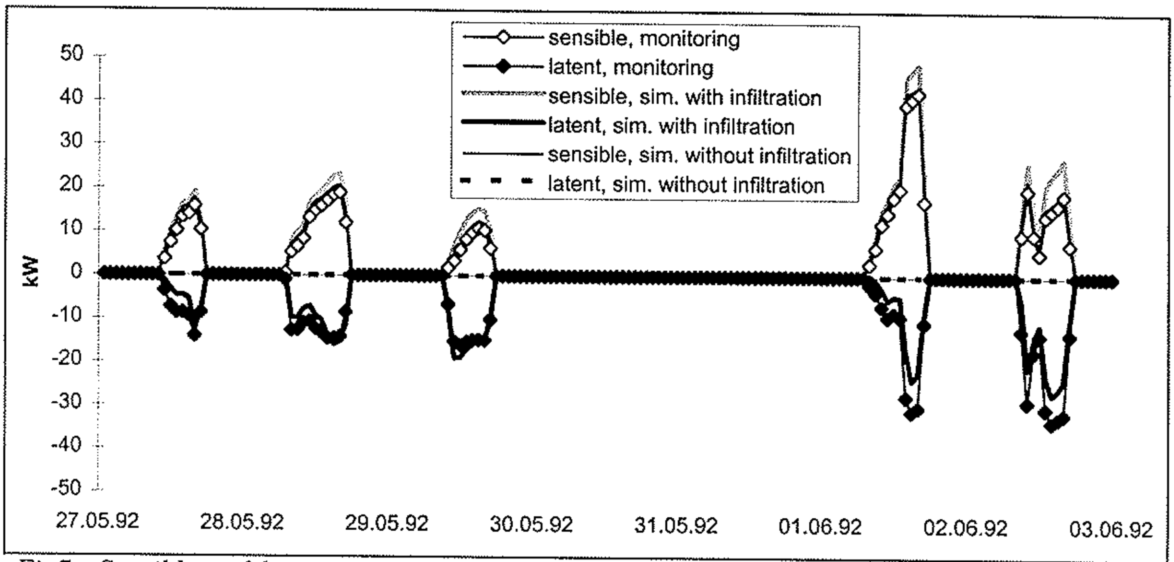


Fig7 : Sensible and latent energy rates furnished to soil in « Schwerzenbacherhof » hypocaust system : monitored and simulated data in hourly timestep (22nd week of 1992).

A first simulation uses an hourly water infiltration of exactly the same amount than seemingly evaporated during monitoring. General trend of resulting hourly, weekly and seasonal dynamic compares generally speaking quite well with data from monitoring (cf. figures 7, 8 and 9), in particular water flows, since almost all infiltrated water is beeing evaporated with expected dynamic (but with a slight summer « storage » of free water within tubes, corresponding to 4mm water on entire exchange surface, partly transfered on winter evaporation). Energy balance default from monitoring is mainly compensated by 1) increase in diffusion from building 2) decrease in diffusion to deep soil 3) lower winter preheating, respectively higher summer cooling (sensible exchange). Note that an altenative run with physicaly more plausible constant water infiltration yielded very similar results, not presented here.

A second simulation was done without water infiltration any (figures 7, 8 and 9). Except for evaporation which disapears completely, all other energy flows result to compare better to monitoring data than when infiltration is at work. This, as well as evolution of soil temperatures (figure 10) coroborates the hypothesis of inconsistency in monitoring of humidity.

[1] as evaluated by temperature gradient from 75 to 50 cm above as well as from 350 to 600 cm beneath pipes, with soil conductivities of 2.4, respectively 2.2 W/K m.

[2] as evaluated by temperature evolution at 50 cm above pipes, at interaxial position, as well as at 50, 150 and 350 cm beneath pipes, with soil capacity above pipes of 2.5 MJ/K m^3, beneath pipes of 2.3 MJ/K m^3.

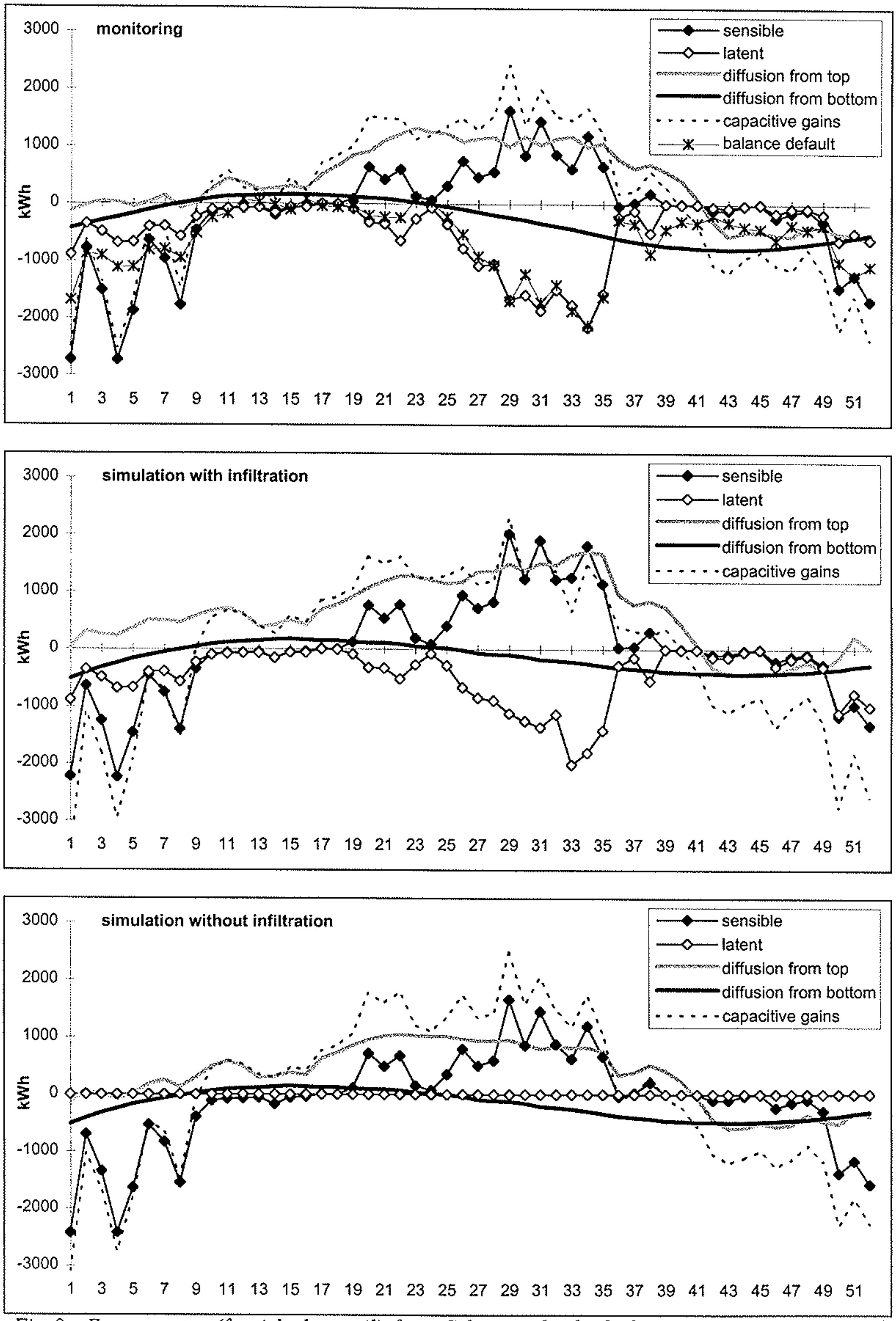


Fig 8 : Energy rates (furnished to soil) for « Schwerzenbacherhof » system : monitoring and simulation (with and without water infiltration) in weekly timestep over one year (1992).

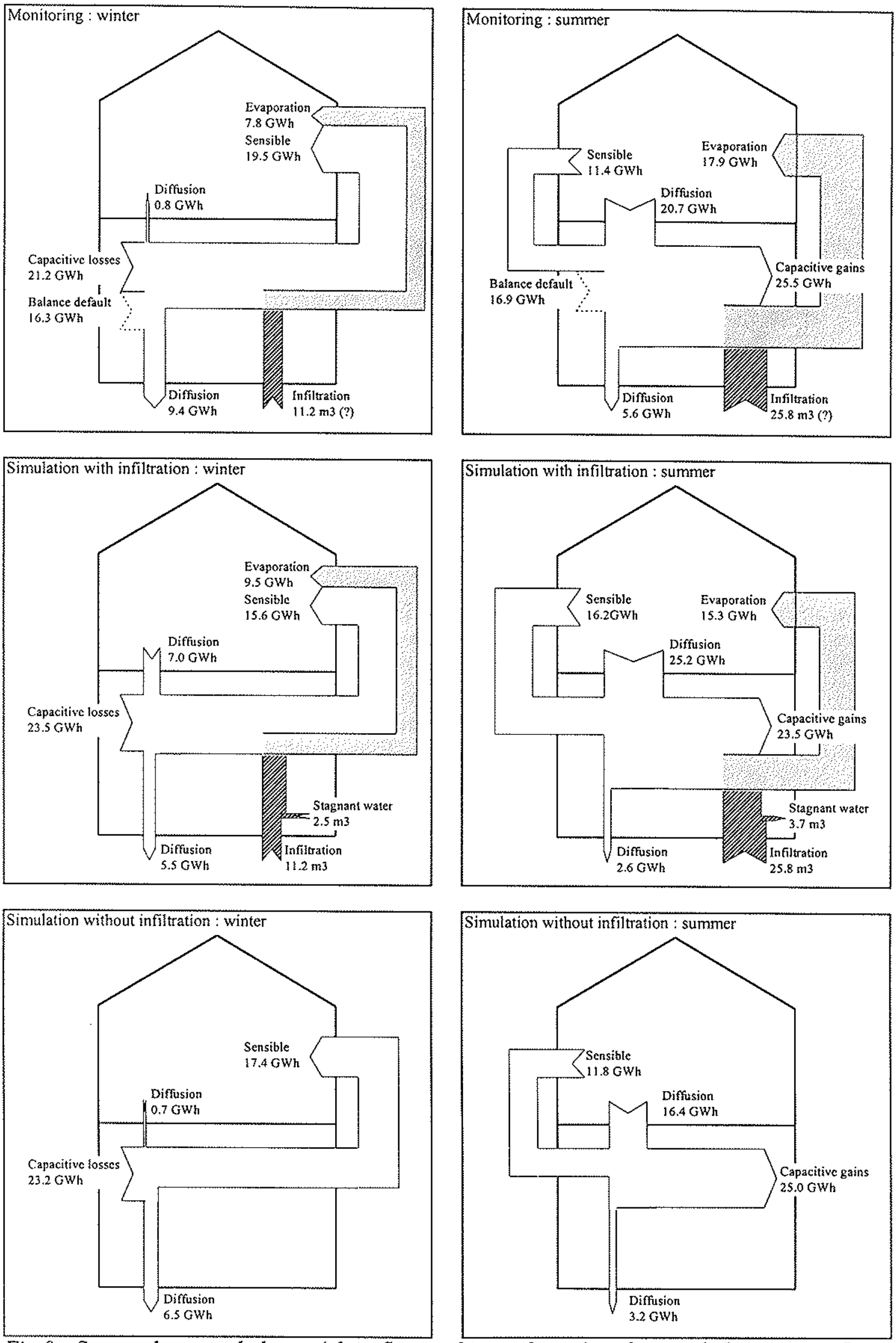


Fig 9 : Seasonal energy balance (plain flows, white and gray) and water balance (grey flows, plain and hatched) for « Schwerzenbacherhof » system : monitoring and simulation (with and without infiltration). Note that energy flow direction has nothing to do with air flow direction (summer sensible cooling power does not corespond to an airflow from building).

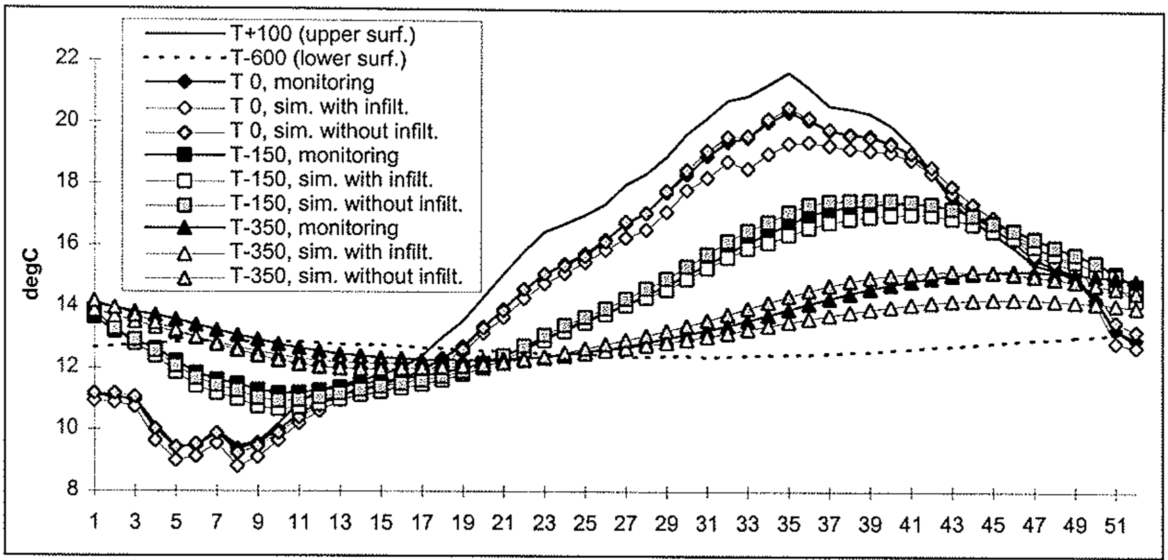


Fig 10 : « Schwerzenbacherhof » ground temperatures : monitoring and simulation (with and without water infiltration) in weekly timestep over one year (1992).

Whatever reality might have been, mere comparison between each of these two simulations gives good understanding of the role water infiltration can play in such a sytem. Hence one observes presence of water and subsequent necessary heat for evaporation to lower winter preheating and to rise summer cooling of air, but only to some extent (changes account respectively for 20 and 28% of seasonal latent heat). Main influence goes for increase in heat diffusion from building (which accounts for 66% of latent heat in winter, respectively 58% in summer). Taken together, changes in air sensible heat and ground diffusion have an important consequence on building energy balance though. In winter, presence of water globaly lowers hypocaust performance of 50% (15.6-7.0=8.6 instead of 17.4-0.7=16.7 GWh). In summer on the contrary, presence of water globaly rises hypocaust performance of 50% (25.2+16.2=41.4 instead of 16.4+11.8=28.2 GWh). This conclusion has nevertheless to be balanced by the fact that sensible heat carried by airflow can be distributed in a controled way, to the opposite of heat diffusion through ground.

Conclusions

Validation on monitored systems shows that the developped hypocaust model, quite well reproduces latent and sensible heat exchanges of buried pipe systems, under condition of taking into account eventual presence of water infiltration. From a monitoring point of view, problematic seems rather to be the evaluation of such flows, while from a project designer point of view it rather is how to control them (tightness of systems). In this frame of idea the model might further on help to evaluate the consequences (risks and benefits) of water infiltration, whose effect relate in analysed cases not so much with outlet temperatures than with surface diffusion from coupled building.

Strictly speaking, some of the validation could gain to be reevaluated in presence of a reciprocal linking with the building, as developped and tested within this work on an academic example. Such a validation exercise would suppose a precise knowledge of the buildings enveloppe, heating/cooling and internal gain structure though.

Another interesting feature (used in sensitivity analysis of the « Geoser » experiment but not presented here) is the flexible geometry, in particular the possibility to simulate soil adjacent to the hypocaust (lateral losses) as well as the optional output structure. In this sense it could be interesting to perform a quite easy extension of the model to water instead of moist air medium, as is more and more commonly used.

Acknoledgments

We wish to thank :

- Thierry Boulard (from the Institut national de recherches agronomiques of Avignon, France) for handing us over the original moist-air hypocaust routine developped in his institute.
- Antoine Reist (from the Station agronomique de Changins, Conthey), Pierre Jaboyedoff (from Sorane SA, Lausanne) and Javier Gil (from P&A Partners, Cochabamba, Bolivia) for their collaboration during the GEOSER experiment.
- Mark Zimmermann (from the Swiss federal laboratories for materials testing and research, Dübendorf) for handing us over the monitoring data of Schwerzenbacherhof.
- Markus Koschenz and Robert Weber (from the Swiss federal laboratories for materials testing and research, Dübendorf) for critical reading and valuable advices for the TRNSYS adaptation.
- The Office fédéral de l'énergie (Bern) and Office cantonal de l'énergie (Genève) for their financial support of the project.

References

1. Razafinjohany E., *Etude comparative dans les serres agricoles de deux systèmes de stockage de la chaleur influencé de l'humidité de l'air*, Thèse, 1989, Académie de Montpelier, Université de Perpignan.
2. Molineaux B., Lachal B., and Guisan O., *Thermal analysis of five outdoor swimming pools heated by unglazed solar collectors*, Solar Energy, Vol. 53, Nb. 1, July 1994, pp. 21-26.
3. Incropera F. P., De Witt D. P., *Fundamentals of heat and mass transfer*, John Wiley & Sons Inc., 1990.
4. Lachal B., Hollmuller P., Gil J., *Stockage de chaleur : résultats de GEOSER*, in *Stockage de chaleur, gestion de l'énergie et du climat dans les serres horticoles*, recueil des exposés du colloque GEOSER du 15 février 1996 au RAC Conthey, Office fédéral de l'énergie, 1996.
5. Lachal B., Hollmuller P., Gil J., *Ergebnisse des GEOSER-Projekts*, in *Energie-Management im Gartenbau*, Tagungsband zur Energiemanagement-Veranstaltung des 26. September 1996 an der ISW Wädenswil, Bundesamt für Energiewirtschaft, 1996.
6. Reist A., Lachal B., Jaboyedoff P., Hollmuller P., Gil J, *GEOSER : stockage temporaire de chaleur pour serres horticoles*, Rapport final (to be published).
7. Lefebvre A.H., *Atomization and sprays*, Hemisphere Publishing Corp., 1989.
8. Rodriguez E.A., Alvarez S., Martin R., *Water drop as a natural cooling ressource : physical principles*, PLEA Conference 1991, Kluver Academic Press, 1991, pp. 499-504.
9. Putallaz J., Lachal B., Hollmuller P., Pampaloni E, *Evaluation des performances du puits canadien de l'immeuble locatif du 19 rue des Caroubiers, 1227 Carouge*, expertise mandatée par l'Office cantonal de l'énergie de Genève, 1996.
10. Zimmermann M., *The Schwerzenbacherhof Office and Industrial Building, Switzerland. Ground Coupled Ventilation System*, IEA Low Energy Cooling Task, 1995.

TRNSYS compatible moist air hypocaust model

Part 2 : User manual

TYPE 61 : HYPOCAUST (AIR-TO-SOIL EXCHANGER)

General Description

This component models an air-to-soil heat exchanger. It accounts for sensible as well as latent exchanges between airflux and tubes, diffusion into surrounding soil, frictional losses and flow of condensed water along the tubes. Local heating from integrated fan motor can be taken into account at tube inlet or outlet. Direction of airflux can be controled (stratification in case of heat storage) and flexible geometry allows for inhomogenous soils as well as diverse border conditions.

Nomenclature

List hereafter covers all symbols used in the mathematical description of the model (other symbols are defined directly in the component configuration section). When as here, symbols in text account for currently described node and timestep, while subsripts are used to reference neighbor nodes or previous timestep.

ClatWat	Latent heat of water
CmAir	Mass-specific heat of air
CmVap	Mass-specific heat of vapor
CmWat	Volume-specific heat of water
CvSoil	Volume-specific heat of soil
CvTub	Volume-specific heat of tube
Ctub	Circumference of tube
Dt	Internal timestep
Dl	Node width (along x, y or z)
Dtub	Hydraulic diameter of tube
Fair	Airflow in tube
FairTot	Airflow, total (over tubes and modules)
Hrel	Relative humidity
Hrat	Absolute humidity (vapor pressure)
Hsat	Absolute humidity (vapor pressure) at saturation
Kair	Air/tube exchange coefficient
Kbord	Heat conduction coefficient of border (pondered, including *Rsurf*)
Ksoil	Heat conduction coefficient to neighbour node or surface (including *Rsurf*)
LamSoil	Heat conductivity of soil
LamTub	Heat conductivity of tube
MmolAir	Molar mass of air
MmolWat	Molar mass of water
Mair	Mass of air exchanged between airflow and tube superficial layer
Mwat	Mass of free water
MwatIn	Mass of water flowing into node
MwatInf	Mass of water infiltrating into node
MwatLat	Mass of water cond./evap.

MwatOut	Mass of water flowing or ejected out of node
Pfric	Energy rate of frictional losses
Pint	Energy rate of tube or soil internal gains
Plat	Energy rate of latent air-tube heat exchange
Psbl	Energy rate of sensible air-tube heat exchange
Psoil	Energy rate of heat diffused by neighbor nodes
Pwat	Energy rate of free water internal losses
PrAir	Pressure of air
Rsurf	Surface heat resistance
Rfric	Friction coefficient of tubes
RhoAir	Specific weight of air
Sair	Section of tube
Sbord	Area of border
Ssoil	Lateral area of soil node
Stub	Lateral area of tube node
Tair	Temperature of air
Tbord	Pondered temperature of border
Tsoil	Temperature of soil
Ttub	Temperature of tube
ThTub	Thickness of tube
Vair	Air velocity
Vwat	Velocity of water
VolSoil	Node volume
VolTub	Volume of tube node
Hrat	Humidity ratio

Mathematical Description

<u>Geometry</u>

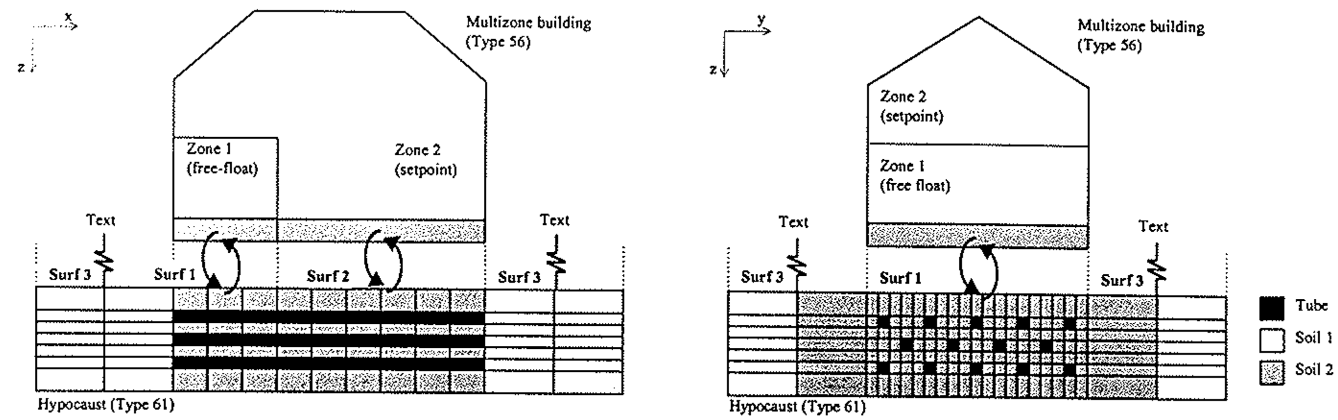
The model describes a block of rectangular soil nodes (which need not all share same physical properties), comprising parallel tubes that run along the x-axis (see figure 1). A correction factor allows to describe non-rectanguar tubes. If not adiabatic, surface conditions (which need not expand from edge to edge) can be given in terms of either inflowing energy rate or temperature. An additional surface resistance can be defined, especially for direct coupling with air temperature.

For matter of simplification and run time economy, symetries in the y-z plane can be used by describing only one module (=relevant part) and specifying the number of times it is used. In this case the symetry surface(s), which must be subject to adiabatic condition, may if necessary pass through the middle of some tubes (see figure 1).

Parametrisation of chosen geometry occurs in following way (of which best understanding can be taken from figure 1 and example at end) :

- define the occurence of typical cross-sections along the x-axis, with numbers that refer to them.
- define the typical cross-sections in the y-z plane, with numbers that refer to soil types, respectively surface conditions.
- define two additional cross-sections for frontal and rear surface conditions.

a) longitudinal section | **b) cross-section # 2 (through both zones)**



parametrisation :

1 1 2 2 2 3 3 3 3 3 3 1 1

where
1 : cross-section # 1 (through back and front)
2 : cross-section # 2 (through both zones)
3 : cross-section # 3 (through main zone)

parametrisation (by symetry only left half) :

```
  3 3 2 2 2 2 2 2 2 2 2 2
0 1 2 2 2 2 2 2 2 2 2 2 0
0 1 2 2 0 2 2 2 0 2 2 0 0
0 1 2 2 2 2 2 2 2 2 2 2 0
0 1 2 2 2 2 0 2 2 2 0 2 2 0
0 1 2 2 2 2 2 2 2 2 2 2 0
0 1 2 2 0 2 2 2 0 2 2 0 0
0 1 2 2 2 2 2 2 2 2 2 2 0
  0 0 0 0 0 0 0 0 0 0 0 0
```

where
0 : tube — 0 : adiabatic surface
1 : soil # 1 — 1 : surface condition # 1 (Zone 1)
2 : soil # 2 — 3 : surface condition # 3 (Ambient)

Fig.1 : Example of Type61 geometry and coupling to other Type.

<u>Linking</u>

In addition to airflow at inlet/outlet, surface conditions can also be coupled to other Types. One can therefore choose between following two modes :

- If output from other module (=input for Type 61) is the energy rate flowing into hypocaust, Type61 will return equivalent border temperature as output (=input for other Type). Latter is defined as the pondered average node temperature of all nodes comprised in that particular surface :

$$Tbord = \frac{\sum_{i \in bord} Ssoil_i \cdot Ksoil_i \cdot Tsoil_i}{Sbord \cdot Kbord} \tag{1}$$

with

$$Ksoil_i = \frac{1}{\frac{Dl_i/2}{LamSoil_i} + Rsurf}$$

$$Sbord = \sum_{i \in bord} Ssoil_i$$

$$Kbord = \frac{\sum_{i \in bord} Ssoil_i \cdot Ksoil_i}{Sbord}$$

One has to take care to use identical border area *Sbord* and heat conduction coefficient *Kbord* in other Type (check these calculated values in parameter control file). The timestep iteration procedure of TRNSYS then will guarantee for proper energy balance (energy rate flowing out of one module = energy rate flowing into other module), which can be checked by plotting inflowing energy rate as optional output of Type 61.

- If on the contrary output from other module is its equivalent border temperature, Type61 will return inflowing energy rate as output. Proper energy balance again is guaranteed by using identical border area and heat conduction coefficient in both Types.

Air flow

Air flow is either positive, negative or zero. If modelling a set of tubes of distinct cross sections, total flow is distributed among the tubes in following way :

$$Fair = FairTot \cdot \frac{Sair \cdot \sqrt{Dtub}}{\sum_{tubes}\left(Sair \cdot \sqrt{Dtub}\right)} \qquad (2)$$

so that according to form of pressure losses (see equation 12 further on) pressure equilibrium at output as well as power and flow integrals are respected.

Water flow

Apart from condensation of airflow (see air-tube heat exchange, further on), water can also enter tubes by infiltration (along part or all of the tube surface). Resultant free water either flows along the tubes or is directly ejected out of hypocaust (flow/ejection occurs in same direction than airflow, in positive direction when airflow is zero). Water flowing/ejected out of a tube node is :

$$MwatOut = \begin{cases} \left(Mwat_{t-1} + \Delta Mwat\right)\dfrac{Vwat \cdot Dt}{Dl} & \text{if water is flowing} \\ \left(Mwat_{t-1} + \Delta Mwat\right) & \text{if water is ejected} \end{cases} \qquad (3)$$

where

$$\Delta Mwat = MwatIn + MwatInf + MwatLat$$

while water flowing from preceeding node (i ±1, depending on flow direction) into actual one is :

$$MwatIn = \begin{cases} MwatOut_{i\pm1} & \text{if water is flowing} \\ 0 & \text{if water is ejected} \end{cases} \qquad (4)$$

Air-tube heat exchange

In each tube node, from inlet towards outlet, following heat exchanges are taken into account :

- **Sensible heat** is caracterised by a an exchange coefficient which depends on flowrate. Cutting short on dimensionless analysis, the model uses a linear dependence on air velocity (as derived from experiences on large plane surfaces [3] and confirmed by the author in the frame of an experience on a burried pipe system).

$$Kair = Kair0 + Kair1 \cdot Vair \tag{5}$$

so that

$$Psbl = Stub \cdot Kair \cdot (Tair - Ttub) \tag{6}$$

- **Latent heat** is determined by the Lewis approach [4] which considers preceeding sensible heat exchange to result from an air mass exchange between the airflux and a superficial air layer on the tube surface, at latters temperature and saturated in humidity. Analogy between heat and mass transfer readily give exchanged air mass during timestep *Dt* :

$$Mair = \frac{Psbl \cdot Dt}{CmAir \cdot (Tair - Ttub)} \quad ,$$

that is

$$Mair = \frac{Stub \cdot Kair \cdot Dt}{CmAir} \quad . \tag{7}$$

This air exchange conveys a vapor transfer, which is determined by the difference of humidity ratios of the airflux and the saturated layer :

$$\begin{aligned} MwatLat &= \left(Hrat(Tair, Hrel) - Hrat(Ttub, 100\%)\right) \cdot Mair \\ &= \left(Hrat(Tair, Hrel) - Hrat(Ttub, 100\%)\right) \cdot \frac{Stub \cdot Kair \cdot Dt}{CmAir} \end{aligned} \tag{8}$$

where, from equation of perfect gazes, humidity ratio computes as

$$Hrat(T, Hrel) = \frac{Hsat(Tair) \cdot MmolWat}{PrAir \cdot MmolAir} \tag{9}$$

According to its sign, this vapor transfer corresponds to condensation ($MwatLat > 0$) or evaporation ($MwatLat < 0$). In latter case *MwatLat* is furthermore limited by 1) available free water in node and 2) saturation pressure of air. Latent heat exchange is finally expressed as

$$Plat = Clat \cdot \frac{MwatLat}{Dt} \tag{10}$$

- **Diffused heat** from surrounding nodes (4 soil nodes, 2 tube nodes) is given by

$$Psoil = \sum_{i=1}^{6} Ssoil_i \cdot Ksoil_i \cdot \left(Tsoil_{i,t-1} - Ttub\right) \quad . \tag{11}$$

where

$$Ksoil_i = \begin{cases} \dfrac{1}{\dfrac{ThTub}{LamTub} + \dfrac{Dl_i/2}{LamSoil_i}} & \text{if neighbor is soil} \\ \dfrac{1}{\dfrac{Dl/2}{LamTub} + \dfrac{Dl_i/2}{LamTub}} & \text{if neighbor is tube} \end{cases}$$

- **Heat from frictional losses** relates to pressure drop along the tubes, which commonly writtes [5] as

$$\Delta PrAir = Rfric \cdot \frac{Dl}{Dtub} \cdot \frac{RhoAir \cdot Vair^2}{2}$$

or

$$\Delta PrAir = Rfric \cdot \frac{Dl \cdot RhoAir}{2} \cdot \frac{Fair^2}{Sair^2 \cdot Dtub} \tag{12}$$

where the friction coefficient *Rfric* is here considered to be independent of air velocity, and the hydraulic diameter of the tube writes as

$$Dtub = \frac{4 \cdot Sair}{Ctub} \tag{13}$$

Related energy rate then writes as

$$Pfric = Fair \cdot \Delta PrAir \tag{14}$$

and is supposed to be gained entirely by the airflow (see energy balance further on).

- **Heat lost by free water** computes as

$$Pwat = CmWat \cdot \frac{Mwat_{t-1} \cdot \left(Ttub_{t-1} - Ttub\right) + MwatIn \cdot \left(Ttub_{i \pm i} - Ttub\right)}{Dt} \tag{15}$$

- **Internal heat gain** is the heat gained by the tube :

$$Pint = \frac{CvTub \cdot VolTub \cdot \left(Ttub - Ttub_{t-1}\right)}{Dt} \tag{16}$$

Preceding energy rates allow to calculate new tube temperature and free water content of actual node, as well as air temperature and humidity ratio of next node. Since the saturated humidity in (9) is non-linear in terms of temperature, *Ttub* is determined by numerical resolution of the tube energy balance

$$Pint = Psbl + Plat + Psoil + Pwat \quad , \tag{17}$$

while water balance readily yields

$$Mwat = Mwat_{t-1} + MwatLat + MwatInf + MwatIn - MwatOut \quad . \tag{18}$$

Sensible energy and water balance on air finaly yield air conditions of next node ($i \pm 1$) :

$$Tair_{i\pm1} = Tair + \frac{Pfric - Psbl}{\left(CmAir + Hrat \cdot CmVap\right) \cdot RhoAir \cdot Sair \cdot Vair} \quad , \tag{19}$$

$$Hrat_{i\pm1} = Hrat - \frac{MwatLat}{RhoAir \cdot Sair \cdot Vair \cdot Dt} \quad , \tag{20}$$

where calculation can be pursued in same manner.

<u>Soil-soil, soil-tube and soil-surface exchanges</u>
Dynamic of soil nodes relies on diffusive heat from neighbor nodes :

$$Psoil = \sum_{i=1}^{6} Ssoil_i \cdot Ksoil_i \left(T_i - Tsoil_{t-1}\right) \quad , \tag{21}$$

where

$$T_i = \begin{cases} Tsoil_{i,t-1} & \text{if neighbor is soil} \\ Ttub_{i,t} & \text{if neighbor is tube} \\ Tsurf_{i,t} & \text{if neighbor is surface} \end{cases}$$

and

$$Ksoil_i = \begin{cases} \dfrac{1}{\dfrac{Dl/2}{LamSoil} + \dfrac{Dl_i/2}{LamSoil}} & \text{if neighbor is soil} \\ \dfrac{1}{\dfrac{Dl/2}{LamSoil} + \dfrac{ThTub}{LamTub}} & \text{if neighbor is tube} \\ \dfrac{1}{\dfrac{Dl/2}{LamSoil} + Rsurf} & \text{if neighbor is surface} \end{cases}$$

It allows to compute new soil temperature :

$$Tsoil = Tsoil_{t-1} + \frac{Psoil}{CvSoil \cdot VolSoil} \quad . \qquad (22)$$

Initialisation
Hypocaust is initialised with a common initial temperature for all nodes, as well as a common initial water thickness along all tubes. Optionaly one may define additional initial temperatures and water thicknesses for certain nodes or node clusters (see further on, definition of parameter file).

TRNSYS Component Configuration

Source code is separated into two files :

- **Type61.for** contains actual routine and is organised in different subroutines.
- **Type61.inc** is an include file used by the subroutines. It contains definition of variables and their organisation in common blocks, as well as definition of maximum allowed sizes, which are listed hereafter with their default values :

NIMax	max number of nodes along x	40
NJMax	max number of nodes along y (per module)	100 [1)]
NKMax	max number of nodes along z (per module)	20 [1)]
NtubMax	max number of tubes (per module)	20 [1)]
NsoilMax	max number of soiltypes	10
NsurfMax	max number of surfaces	6 [2)]
NoptMax	max number of optional outputs	100 [2)]
NiniMax	max number of initialisation conditions	20

1) module = relevant part in y-z plane (see further up, description of geometry).
2) Changing default values for maximum number of surfaces or maximum number of optional outputs will need renumeration of routine arguments (parameters, inputs and outputs) as defined in information flow diagram.

Input data is separated into three groups, of which two are passed as arguments, the last one read from a file :

- **Parameters** describe fixed data that deal with linking to other modules and with simulation deck.
- **Inputs** describe variable data.
- Parameters which are proper to the model are passed by means of a **Parameter definition file**, which is read by the routine at initialisation. While reading, the data is checked and rewritten to a control file (see below), so that eventual errors can be tracked.

Output data is separated into two groups, of which first one is returned as argument, second one written to a file :

- **Outputs** describe variable data, which can be linked to other modules.

- Parameters which are derived from supplied parameter file or from simulation deck are written to a **Parameter control file**, which can be used to check for proper definition. As pointed out, first part of this file is a formatted and commented copy of Parameter definition file (which it can substitute).

A synoptic view of these data groups is to be found in the information flow diagarm (next section), while this section presents each of them in a detailed table (with explanatory notes following last table).

Note, especially in case of debugging, that data is passed to/from the routine with TRNSYS compatible units as defined hereafter, where it is converted to standard SI units.

Parameters

Number	**Symbol**	**Definition and unit**	
1	*IparDef*	Logical unit of Parameter definition file [-]	1)
2	*IparCon*	Logical unit of Parameter control file [-]	1)
3	*Dt*	Internal timestep [hr]	2)
4	*FairMin*	Minimum airflow [m^3/hr]	3)
5	*DTtubTol*	Temperature tolerance for tube energy balance [K]	4)
6 - 11	*TypSurf*	Linking modes for surfaces 1 - 6 [-]	5)
12 - 17	*Rsurf*	Heat resistance at surfaces 1 - 6 [K m2 hr/kJ]	

Inputs

Number	**Symbol**	**Definition and unit**	
1	*FairTot*	Airflow, total over all modules [m3/hr]	6)
2	*TairIn*	Inlet temperature [degC]	
3	*HrelIn*	Inlet relative humidity [pcent]	
4	*PrAir*	Air pressure [bar]	7)
5	*FwatInfTot*	Water infiltration, total over all modules [m3/hr]	8)
6 - 11	*Xsurf*	Surface conditions for surfaces 1 - 6 [degC] or [kJ/hr]	5)

Parameter definition file
Each data set hereafter is written on one line (exception for *TypSoil* arrays, which take *NK* or *NK*+2 lines). Data within one dataset is separated by commas or blanks. Comments can be entered by using an asterix (*) in first column.

Symbol	**Definition and unit**	
Nmod,Nsec,Nsoil,Nsurf,NI,NJ, NK	Number of : modules, cross-sections, surfaces, nodes along x-axis, nodes along y-axis, nodes along z-axis [-]	
Dx (1:NI)	Node width along x-axis [m]	9)
Dy (1:NJ)	Node width along y-axis [m]	9)
Dz (1:NK)	Node width along z-axis [m]	9)
TypSec(1:NI)	Type of used cross-sections along x-axis [-]	10)
TypSoil (1:NJ,1:NK)	Type of surfaces on frontal cross-section [-]	11)
TypSoil (0:NJ+1,0:NK+1)	Type of soils/surfaces for typical cross-section in y-z plane [-]	12)

TypSoil (1:NJ,1:NK)	Type of surfaces on rear cross-section [-]	11)
PosInf	Position of water infiltration [-]	8)
Kair0, Kair1	Air-tube exchange coefficients [kJ/hr K m2] and [(kJ/hr K m2)/(m/s)]	13)
LamSoil, CvSoil	Soil conductivity [kJ/hr K m] and capacity [kJ/K m3]	14)
LamTub, CvTub	Tube conductivity [kJ/hr K m] and capacity [kJ/K m3]	
ThTub, CtubCor, Rfric	Tube thickness [m], circumference correction factor [-] and friction coefficient [-]	9) 15)
TypWatFlow (-1:1)	Type of water flow [-]	16)
Vwat (-1:1)	Velocity of water flow [m/hr]	16)
NiniSoil, NiniWat	Number of initial conditions (soil temperatures and waterthicknesses) [-]	17)
TiniSoil, PosIniSoil (1:6)	Initial temperature [degC] and corresponding node position [-]	17)
ThIniWat, PosIniWat (1:6)	Initial waterthickness [m] and corresponding node position [-]	17)
Nopt	Number of optional outputs	20)
TypOpt, PosOpt (1:6)	Type of optional output [-] and corresponding node position [-]	20)

Outputs

Number	**Symbol**	**Definition and unit**	
1	*TairOut*	Outlet temperature [degC]	
2	*HrelOut*	Outlet relative humidity [pcent]	
3	*PsblTot*	Sensible energy rate lost by airflow, total over tubes and modules [kJ/hr]	
4	*PlatTot*	Latent energy rate lost by airflow, total over tubes and modules [kJ/hr]	
5 - 10	*Xbord*	Equivalent border output for surfaces 1 - 6 [degC] or [kJ/hr]	5)
11 - 20	*Xopt*	Optional outputs	20)

Parameter control file

Data hereafter is written at end of file, after formatted copy of Parameter definition file.

Symbol	**Definition and unit**	
Ntub	Number of tubes (per module) [-]	
IflowIni	Node index of tube start along x-axis [-]	
IflowEnd	Node index of tube end along x-axis [-]	
PosTub(1:2)	Node index of tube position along y- and z-axis [-]	18)
Lx	Length of hypocaust [m]	
Ly	Width of hypocaust (total over modules) [m]	
Lz	Depth of hypocaust [m]	
Ltub	Length of tubes [m]	
SairTot	Tube cross-section area (total over all tubes and modules) [m2]	
StubTot	Tube surface (total over all tubes and modules) [m2]	

ZairTot	Normalisation factor for airflow distribution [m5/2]	
SinfTot	Water infiltration surface, total over all modules [m2]	8)
Sbord	Border area (total over all modules) [m2]	19)
Kbord	Equivalent border conduction coefficient [kJ/hr K m2]	19)
DtSoil	Maximum internal timestep for stability of soil temperature [hr]	
DtWat	Maximum internal timestep for consistency of water flow [hr]	
FairMinTub	Minimum air flow for stability of air temperature [m3/hr]	
Dt	Internal timestep effectively used in simulation [hr]	
FairMin	Minimum air effectively flow used in simulation [m3/hr]	

Explanatory notes for preceeding tables

1) Unless assigned in simulation deck with user-defined name, parameter definition and control files must by default be named ParamDef.txt and ParamCon.txt.
2) Since calculation of soil temperature is of explicit type, internal timestep should not exceed a maximum theoretical value *DtSoil*, which is proportional to smallest node volume of soil (problem of temperature oscillation). Consistency of water flow calculation (equation 3) also implies a maximum value *DtWat* for internal timestep, proportional to shortest tube node. Both of these computed values are written to the parameter control file. Type 61 usualy takes the smallest of these two values for the internal timestep (which happens by setting the 3rd routine parameter *Dt* to zero). The user may alternatively control soil temperature oscillation by defining a larger or smaller internal timestep himself (which happens by setting the 3rd routine parameter *Dt* to a positive value), in which case the value *DtWat* should not be exceeded though.
3) So as to avoid oscillations of air temperature along the tube, airflow should not exceed a theoretical minimum value *FairMinTub*, which is written to the parameter control file. Type 61 usualy takes this value as a lower limit to the airflow (which happens by setting the 4th routine parameter *FairMin* to zero). The user may alternatively control air temperature oscillation by defining a larger or smaller minimum airflow himself (which happens by setting the 4th routine parameter *FairMin* to a positive value). In both cases an airflow smaller than the minmum value will be set to zero (no air-tube exchange, only diffusion within soil).
4) Temperature tolerance (>0) sets precision of numerical resolution of energy balance in tube (equation 17).
5) For each surface, linking mode is one of the following :
   0 : corresponding input *Xsurf* is surface temperature, output *Xbord* is inflowing energy rate.
   1 : corresponding input *Xsurf* is is inflowing energy rate, output *Xbord* is equivalent border temperature.
6) Airflow direction along x-axis is carried by sign of airflow. If airflow is smaller (in absolute value) than minimum airflow *FairMin* (see parameter control file) it is considered as zero (no air-tube exchange, only diffusion within soil).
7) Air pressure is used to convert volume flow in mass flow as well as to determine humidity ratio from relative humidity (equation 9). In usual cases its dynamic is not known and it is suggested to take standard atmosferic pressure at local altitude, which can be approximated by :

$$PrAir = PrAir_0 \exp(-h/h_0) \text{ with } PrAir_0 = 1.01325 \text{ bar}, h_0 = 7656 \text{ m}.$$

8) Water infiltration is distributed on a certain tube area *SinfTot*, defined by the rectangular node cluster *PosInf* on which infiltration is to take place.
PosInf(1) and *PosInf(4)* are lower and upper node index along x-axis.
PosInf(2) and *PosInf(5)* are lower and upper node index along y-axis.
PosInf(3) and *PosInf(6)* are lower and upper node index along z-axis.
Only tube nodes within this cluster are considered for water infiltration.
9) Even for non-rectangular tubes, node width must be chosen so that cross-section area is given by *DyDz*. Cross-section perimetrer, exchange surfaces and hydraulic diameter will be corrected by tube circumference correction factor *CtubCor*. Latter is defined as the ratio between *real* tube perimeter and *rectangular* tube perimeter 2(*Dy* + *Dz*). For circular tubes node width has to be chosen so that $Dy = Dz = r\sqrt{\pi} \cong 1.772\,r$ and circumference correction factor becomes $\frac{1}{2}\sqrt{\pi} \cong 0.8862$. In case of a symetry plane passing in the middle of some tubes (tube node at hypocaust border, with latteral adiabatic condition) one furthermore has to divide Dy by half.
Generally speaking node widths *Dx*, *Dy* and *Dz* have to be chosen according to given problem, reminding that small soil volumes will lead to small internal timesteps and increase of runtime.Tube thickness *ThTub* may however be set to zero.
10) *TypSec(1:NI)* are positive integer numbers which refer to further on defined typical cross-sections along x-axis.
11) *TypSoil (1:NJ,1:NK)* are integer numbers which refer to given surface conditions for front and rear of hypocaust module (see example at end).
12) *TypSoil (0:NJ+1,0:NK+1)* are integer numbers which refer to further on defined soil types (bulk) or to given surface conditions (border). Exception are the 4 corners *TypSoil (0,0)* , *TypSoil (NJ+1,0)* , *TypSoil (0,NK+1)* , *TypSoil (NJ+1,NK+1)* which have no significance and are not defined (see figure 1 and example at end). This data set has to be repeated for the *Nsec* number of typical cross-sections.
13) Common values for air-tube exchange coefficients [3] are :
Kair0 : 7 - 11 [kJ/hr K m2]
Kair1 : 14 - 18 [(kJ/hr K m2)/(m/s)]
14) This line has to be repeated for the *Nsoil* number of soils.
15) Typical values for Friction coefficient are 0.01 - 0.02 [-].
16) Specification of water flow is given for all 3 airflow diections (negative, zero, positive). *TypWatFlow* indicates whether free water is to flow along the tubes (= 1) or to be ejected out (= 2). *Vwat* (≥ 0) specifies velocity of waterflow (if *TypWatFlow* = 1).
17) Initial temperatures are given for rectangular node clusters, defined by *PosIniSoil* :
PosIniSoil(1) and *PosIniSoil(4)* are lower and upper node index along x-axis,
PosIniSoil(2) and *PosIniSoil(5)* are lower and upper node index along y-axis,
PosIniSoil(3) and *PosIniSoil(6)* are lower and upper node index along z-axis,
except for first initial temperature which is applied to all nodes and thus does not need definition of *PosIniSoil* (see example at end).
Same structure accounts for initial water thicknesses. In this case only those nodes within the cluster which do effectively corespond to tube nodes are taken into account though.
18) This line is repeated for the *Ntub* number of tubes.
19) This line is repeated for the Nsurf number of surfaces.

20) *Nopt* defines the number of desired optional outputs. For each one of them *TypOpt* specifies the type of optional output and takes a value from one of the three following tables. *PosOpt* finally defines the rectangular node cluster for which the optional output is to be considered :
PosOpt(1) and *PosOpt(4)* are lower and upper node index along x-axis,
PosOpt(2) and *PosOpt(5)* are lower and upper node index along y-axis,
PosOpt(3) and *PosOpt(6)* are lower and upper node index along z-axis.
If *TypOpt* relates to tube/air nodes, only tube nodes within cluster will be considered.
If *TypOpt* relates to soil nodes, only soil nodes within cluster will be considered.
If *TypOpt* relates to miscelanous data, *PosOpt* is of no significance and should be set to 1.

Optional outputs for tube nodes :

Type	Symbol	Definition and unit	
1	*Tair*	Air temperature [degC]	*
2	*Hrel*	Air relative humidity [pcent]	*
3	*Habs*	Air absolute humidity [bar]	*
4	*Hrat*	Air humidity ratio [kg vapor/kg air]	*
5	*Mwat*	Free water in node [m3]	**
6	*MwatLat/Dt*	Water condensing (>0) or evaporating (<0) [m3/hr]	**
7	*MwatIn/Dt*	Water flowing into node [m3/hr]	**
8	*MwatInf/Dt*	Water infiltrating into node [m3/hr]	**
9	*MwatOut/Dt*	Water flowing out of node [m3/hr]	**
10	*Tsoil*	Tube temperature [degC]	**
11	*Psbl*	Sensible energy rate from air to tube [kJ/hr]	**
12	*Plat*	Latent energy rate from air to tube [kJ/hr]	**
13	*Pwat*	Energy rate lost by free water [kJ/hr]	**
14	*Pfric*	Energy rate from frictional losses [kJ/hr]	**
15	*Psoil(0)*	Energy rate diffused from all 6 neighbor nodes [kJ/hr]	**
16	*Psoil(1)*	Energy rate diffused from previous neighbor node along x-axis (from surface if border node) [kJ/hr]	**
17	*Psoil(2)*	Energy rate diffused from next neighbor node along x-axis (from surface if border node) [kJ/hr]	**
18	*Psoil(3)*	Energy rate diffused from previous neighbor node along y-axis (from surface if border node) [kJ/hr]	**
19	*Psoil(4)*	Energy rate diffused from next neighbor node along y-axis (from surface if border node) [kJ/hr]	**
20	*Psoil(5)*	Energy rate diffused from previous neighbor node along z-axis (from surface if border node) [kJ/hr]	**
21	*Psoil(6)*	Energy rate diffused from next neighbor node along z-axis (from surface if border node) [kJ/hr]	**
22	*Pint*	Energy rate of internal gains [kJ/hr]	**
23	*Fair*	Air flowrate [m3/hr]	*
24	*Vair*	Air velocity [m/s]	*

* averaged over node cluster
** integrated over node cluster and multiplied by number of modules

Optional outputs for soil nodes :

Type	Symbol	Definition and unit	
101	*Tsoil*	Soil temperature [degC]	*
102	*Psoil(0)*	Energy rate diffused from all 6 neighbor nodes [kJ/hr]	**
103	*Psoil(1)*	Energy rate diffused from previous neighbor node along x-axis (from surface if border node) [kJ/hr]	**
104	*Psoil(2)*	Energy rate diffused from next neighbor node along x-axis (from surface if border node) [kJ/hr]	**
105	*Psoil(3)*	Energy rate diffused from previous neighbor node along y-axis (from surface if border node) [kJ/hr]	**
106	*Psoil(4)*	Energy rate diffused from next neighbor node along y-axis (from surface if border node) [kJ/hr]	**
107	*Psoil(5)*	Energy rate diffused from previous neighbor node along z-axis (from surface if border node) [kJ/hr]	**
108	*Psoil(6)*	Energy rate diffused from next neighbor node along z-axis (from surface if border node) [kJ/hr]	**
109	*Pint*	Energy rate of internal gains [kJ/hr]	**

* averaged over node cluster

** integrated over node cluster and multiplied by number of modules

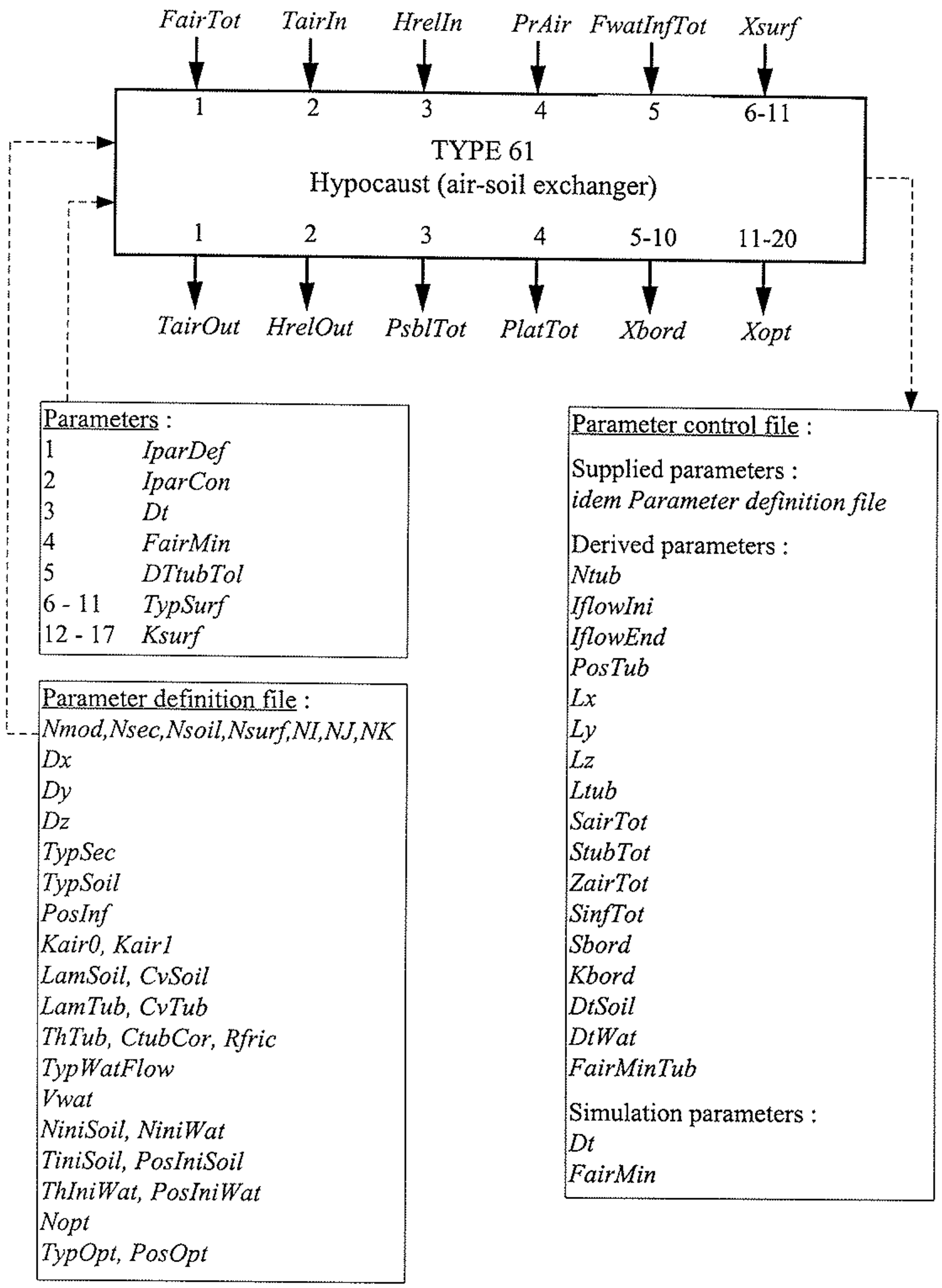
Miscellaneous data for optional output :

Type	Symbol	Definition and unit
201	*PsurfTot*	Total inflowing energy rate through surfaces (over all modules) [kJ/hr]
202	*PwatTot*	Total energy loss of free water (over all modules) [kJ/hr]
203	*PfricTot*	Total frictional losses (over all modules) [kJ/hr]
204	*PintTot*	Total tube and soil capacitive gains (over all modules) [kJ/hr]

Références

1. Hollmuller P., Lachal B., *TRNSYS compatible moist air hypocaust model : description and validation*, Centre universitaire d'études des problèmes de l'énergie, CH - 1205 Genève, 1998.
2. Razafinjohany E., *Etude comparative dans les serres agricoles de deux systèmes de stockage de la chaleur influencé de l'humidité de l'air*, Thèse, 1989, Académie de Montpelier, Université de Perpignan.
3. Molineaux B., Lachal B., and Guisan O., *Thermal analysis of five outdoor swimming pools heated by unglazed solar collectors*, Solar Energy, Vol. 53, Nb. 1, July 1994, pp. 21-26.
4. Incropera F. P., De Witt D. P., *Fundamentals of heat and mass transfer*, John Wiley & Sons Inc., 1990.
5. *1989 Ashrae Handbook, Fundamentals*, American society of heating, refrigerating and air conditioning engineers inc., 1791 Tullie Circle, N.E., Atlanta, GA 30329.

Information Flow Diagram



Example

Description

Example is the underground cooling system shown in Fig. 1. It is a mere case study ment to show the possibility of linking Type61 to the multizone building Type 56 and to check consistency of exchanged energy rates as well as of other internal variables. Hence following hypothesis are made :

- Ambient conditions are constant : temperature of 30°C, humidity of 50%, no solar radiation.
- Building is simplified to its uttermost : a first zone (8 m2, 16 m3) with simple brick wall is free-floating and adjoins a second zone (12 m2, 39 m3) with insulated brick wall and at fixed temperature (15°C). No windows are taken into account and no infiltration nor cross-ventilation is considered.
- Pipe system is underneath building and latteraly not insulated, wherefor lateral and from hypocaust distinct soil is taken into account.
- Airflow is constant (1000 m^3/hr) and is not injected into building but supposed to be used elsewhere.
- No water infiltration is considered, nor does free water flow along the tubes.
- Initial temperatures are 10°C for hypocaust, 15°C for surrounding soil and building.

Following variables are defined and analysed (some of which, for checking of proper energy and mass balance, are calculated by two alternative ways defined in the deck) :

Psbl	: senible energy lost by airflow
Plat	: latent energy lost by airflow
Pin	: internal gains of hypocaust and surrounding soil
PinG, PinG#	: internal gains of surrounding soil
PinH, PinH#	: internal gains of hypocaust
PinHt	: internal gains of hypocaust tubes
PinHs	: internal gains of hypocaust soil
Pfree, Pfree#	: energy diffused from free-floating zone into hypocaust
Pfix, Pfix#	: energy diffused from fixed setpoint zone into hypocaust
Pamb	: energy diffused from ambient into surrounding soil
Pfront	: energy diffused from surrounding soil front of the building into hypocaust
Pback	: energy diffused from surrounding soil back of the building into hypocaust
Pside	: energy diffused from surrounding soil to side of the building into hypocaust
Pwat	: energy diffused from free water into hypocaust
Pfric	: friction losses
T1-T4	: temperatures of airflow along tubes (mean value of all tubes)
Tout	: temperature of airflow at outlet
Tfree	: temperature of free-floating zone, air
TgFree	: temperature of free-floating zone, ground
Tfix	: temperature of fixed setpoint zone, air
TgFree	: temperature of fixed setpoint zone, ground
Mwat, Mwat#	: free water within tubes
dMlat	: condensation/evaporation within tubes
dMout	: total outflowing water.

Next pages show files for parametrisation of the system (parameter definition file for Type 61, building definition file for Type 56, simulation deck), after which corresponding simulation results are discussed.

Type61.par : parameter definition file (Type 61)

```
************************************************
* TYPE 61 SUPPLIED PARAMETERS
*===============================================
* Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK [-]:
  2    3    2     3     13 12  7

* DX [m]:
  1.0000E+00  1.0000E+00
  0.6666E+00  0.6666E+00  0.6666E+00
  0.6666E+00  0.6666E+00  0.6666E+00
  0.6666E+00  0.6666E+00  0.6666E+00
  1.0000E+00  1.0000E+00

* DY [m]:
  1.0000E+00  1.0000E+00  0.3000E+00  0.2000E+00
  0.2000E+00  0.2000E+00  0.2000E+00  0.2000E+00
  0.2000E+00  0.2000E+00  0.2000E+00  0.1000E+00

* DZ [m]:
  0.4000E+00  0.2000E+00  0.2000E+00  0.2000E+00
  0.2000E+00  0.2000E+00  0.4000E+00

* TypSec [-]:
   1   1   2   2   2   3   3   3   3   3   3   1   1

* TypSoil for front surface [-]:
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0

* TypSoil for sec# 1 (through ambient) [-]:
       3   3   3   3   3   3   3   3   3   3   3   3
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
   0   2   2   2   2   2   2   2   2   2   2   2   2   0
       0   0   0   0   0   0   0   0   0   0   0   0

* TypSoil for sec# 2 (through both zones) [-]:
       3   3   1   1   1   1   1   1   1   1   1   1
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   0   1   1   1   0   1   1   1   0   0
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   1   1   0   1   1   1   0   1   1   0
```

```
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   0   1   1   1   0   1   1   1   0   0
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
       0   0   0   0   0   0   0   0   0   0   0   0

* TypSoil for sec# 3 (through setpoint-zone only) [-]:
       3   3   2   2   2   2   2   2   2   2   2   2
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   0   1   1   1   0   1   1   1   0   0
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   1   1   0   1   1   1   0   1   1   0
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
   0   2   2   1   0   1   1   1   0   1   1   1   0   0
   0   2   2   1   1   1   1   1   1   1   1   1   1   0
       0   0   0   0   0   0   0   0   0   0   0   0

* TypSoil for rear surface [-]:
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0
       0   0   0   0   0   0   0   0   0   0   0   0

* PosInf [-]:
  1   1   1   9  12   7

* Kair0 [kJ/K m2] ,Kair1 [(kJ/K m2)/(m/s)]:
  0.1800E+02  0.1400E+02

* LamSoil [kJ/K m], CvSoil [kJ/K m3]:
  0.7200E+01  0.1000E+04
  0.5400E+01  0.1000E+04

* LamTub [kJ/K m], CvTub [kJ/K m3]:
  0.7200E+01  0.1000E+04

* ThTub [m], CtubCor [-], Cfric [-]:
  5.0000E-03  0.8862E+00  2.0000E-02

* TypWatFlow [-], Vwat [m/h]:
  1   1   1
  0.0000E+00  0.0000E+00  0.0000E+00

* NiniSoil,NiniWat [-]:
  2   1

* TiniSoil [degC], PosIniSoil [-]:
  0.1500E+02
  0.1000E+02  3   3   1  11  12   7

* ThIniWat [m], PosIniWat [-]:
  0.0000E+00

* Nopt [-]:
  17
```

```
* TypOpt [-], PosOpt [-]:
  107   3   3   1   5  12   1 !Pfree#
  107   6   3   1  11  12   1 !Pfix#
  103   3   3   1   3  12   7 !Pfront
  104  11   3   1  11  12   7 !Pback
  105   3   3   1  11   3   7 !Pside
  202   1   1   1   1   1   1 !Pwat
  203   1   1   1   1   1   1 !Pfric
  204   1   1   1   1   1   1 !Pin
   22   3   3   1  11  12   7 !PinHt
  109   3   3   1  11  12   7 !PinHs
    5   3   3   1  11  12   7 !Mwat
    6   3   3   1  11  12   7 !dMlat
    9  11   3   1  11  12   7 !dMout
    1   4   3   1   4  12   7 !T1
    1   6   3   1   6  12   7 !T2
    1   8   3   1   8  12   7 !T3
    1  10   3   1  10  12   7 !T4
*************************************************
```

Observations :

- Because of symetry in the y-z plane, only half of the hypocaust has to be simulated, cutting middle two pipes by half (*Nmod* = 2 and last node width *Dy* is half the width of other ones).
  3 cross-sections must be defined, one outside the building, two through the building (one cutting both zones, the other one through fixed zone only), as well as 3 surface conditions (ambient and floor of both zones).
- 2 temperature initialisation are used, for soils surrounding and beneath building respectively.

Building.bui : building definition file (Type 56)

```
**************************************************************************
* TYPE 56 DESCRIPTION                                                    *
**************************************************************************

PROPERTIES
**************************************************************************
DENSITY=1.204  : CAPACITY=1.012  : HVAPOR=2454  : SIGMA=2.041E-07
RTEMP =293.15

TYPES
**************************************************************************

*-- LAYERS ---------------------------------------------------------------
LAYER Brick30
THICKNESS=.30  : CONDUCTIVITY=3  : CAPACITY=1  : DENSITY=1800

LAYER Insul10
THICKNESS=.10  : CONDUCTIVITY=0.144  : CAPACITY=0.72  : DENSITY=90

LAYER Soil40
THICKNESS=.40  : CONDUCTIVITY=7.2  : CAPACITY=1  : DENSITY=1000

*-- INPUTS ---------------------------------------------------------------
INPUTS TgFree TgFix
```

```
*-- WALLS -----------------------------------------------------------
WALL Brick
LAYERS Brick30
ABS-FRONT=.8 : ABS-BACK=.8 : HFRONT=15 : HBACK=15

WALL Insul_Brick
LAYERS Insul10 Brick30
ABS-FRONT=.8 : ABS-BACK=.8 : HFRONT=15 : HBACK=15

WALL Soil
LAYERS Soil40
ABS-FRONT=.8 : ABS-BACK=0 : HFRONT=15 : HBACK=36
* rem : HBACK must be equal to Kbord from Type 61 *

WALL Insul_Soil
LAYERS Insul10 Soil40
ABS-FRONT=.8 : ABS-BACK=0 : HFRONT=15 : HBACK=36
* rem : HBACK must be equal to Kbord from Type 61 *

*-- COOLING ---------------------------------------------------------
COOLING CoolFix
ON=15 : POWER=1E6 : HUMIDITY=0

*-- ORIENTATIONS ----------------------------------------------------
ORIENTATIONS Ambient

*-- ZONES -----------------------------------------------------------
ZONES Free Fix

BUILDING
*********************************************************************

*-- ZONE Free -------------------------------------------------------
ZONE Free

WALL=Insul_Brick : AREA=16 : ADJACENT=Fix : BACK    : COUPLING=0
WALL=Brick       : AREA=16 : EXTERNAL : ORIENTATION=Ambient : FSKY=0.5
WALL=Soil        : AREA=8  : BOUNDARY=INPUT TgFree : COUPLING=0

REGIME
CAPACITANCE=1E+3 : VOLUME=16 : TINITIAL=15 : PHIINITIAL=50.0 : WCAPR=1

*-- ZONE Fix --------------------------------------------------------
ZONE Fix

WALL=Insul_Brick : AREA=16 : ADJACENT=Free : FRONT  : COUPLING=0
WALL=Insul_Brick : AREA=41 : EXTERNAL : ORIENTATION=Ambient : FSKY=0.5
WALL=Insul_Soil  : AREA=12 : BOUNDARY=INPUT TgFix  : COUPLING=0

REGIME
COOL=CoolFix
CAPACITANCE=1E+3 : VOLUME=39 : TINITIAL=15 : PHIINITIAL=50.0 : WCAPR=1

OUTPUTS
*********************************************************************

*-- TRANSFER --------------------------------------------------------
TRANSFER : TIMEBASE=1
```

```
*-- OUTPUTS --------------------------------------------------------------

ZONES=Free
NTYPES=1 20

ZONES=Fix
NTYPES=1 20

END
*************************************************************************
```

Observations :

- Preceding file must be processed by BID program before it can be used by Type 56 (for more details refer to Type56 component description).
- Note that for proper coupling with Type 61 *HBACK* of soil is set to identical value as *Kbord* from hypocaust and ground areas are identical to *Ssurf* from hypocaust (see Parameter control file to check this).

Type61.dck : simulation deck file

```
*****************************************************************
* SIMULATION:
*****************************************************************

*================================================================
  ASSIGN  Trnsys.txt      6
  ASSIGN  Out1.txt      101
  ASSIGN  Out2.txt      102
  ASSIGN  Out3.txt      103
  ASSIGN  Type61.par    200
  ASSIGN  Type61.con    201
  ASSIGN  Building.bld  300
  ASSIGN  Building.trn  301
  ASSIGN  Building.win  302
*================================================================

*================================================================
  EQUATIONS 37
*-------------------
  DtSim   = 1
  Tamb    = 30
  Hamb    = 50
  Aflow   = 1000
*-------------------
  Tfree = [1,1]
  Tfix  = [1,5]
  Pfree = -[1,4]
  Pfix  = -[1,8]
*-------------------
  Tout    = [2,1]
  Psbl    = [2,3]
  Plat    = [2,4]
  TgFree  = [2,5]
  TgFix   = [2,6]
```

```
  Pamb    = [2,7]
  Pfree#  = [2,11]
  Pfix#   = [2,12]
  Pfront  = [2,13]
  Pback   = [2,14]
  Pside   = [2,15]
  Pwat    = [2,16]
  Pfric   = [2,17]
  Pin     = [2,18]
  PinHt   = [2,19]
  PinHs   = [2,20]
  Mwat    = [2,21]*1000
  dMlat   = [2,22]*1000
  dMout   = [2,23]*1000
  T1      = [2,24]
  T2      = [2,25]
  T3      = [2,26]
  T4      = [2,27]
*--------------------
  PinH   = PinHt+PinHs
  PinG   = Pin-PinHt-PinHs
  PinH#  = Psbl+Plat+Pfree+Pfix+Pfront+Pback+Pside+Pwat
  PinG#  = Pamb-Pfront-Pback-Pside
  dMwat  = dMlat-dMout
  Mwat#  = GT(TIME,1)*[3,1]+LT(TIME,2)*dMwat
* Mwat#  = [3,1] replaced by preceding line because of bug in
*                integrator Type55
*=============================================================

*=============================================================
  SIMULATION    1            100          DtSim
  TOLERANCES    -0.0001      -0.0001
*=============================================================

**************************************************************
* COMPONENTS:
**************************************************************

*=============================================================
* Multizone Building
*-------------------------------------------------------------
UNIT 1   TYPE 56

PARAMETERS 5
*---------------------
* 01) Logical unit of building description file
* 02) Logical unit of transfer coefficient file
* 03) Logical unit of window library file
* 04) Mode of calculation for star network
* 05) Weighting factor for operative romm temperature

  300         301          302          0            0.5

INPUTS 8
*---------------------
* 01) Ambient temperature [degC]
* 02) Ambient humidity ratio [kg water / kg air]
* 03) Fictive sky temperature [degC]
```

```
* 04) Incident radiation for orientation ambient [kJ/hr]
* 05) Incident beam radiation for orientation ambient [kJ/hr]
* 06) Incident angle for orientation ambient [deg]
* 07) Ground temperature zone "Free" [deg C]
* 08) Ground temperature zone "Fix" [deg C]

  Tamb          0,0           Tamb          0,0           0,0
  0,0           TgFree        TgFix

  0.000E+00     0.000E+00     0.000E+00     0.000E+00     0.000E+00
  0.000E+00     0.100E+02     0.100E+02

* OUPUTS 8
*---------------------
* 01) Temperature of zone "Free" [degC]
* 02) Energy rate from zone "Free" to zone "Fix" [kJ/hr]
* 03) Energy rate from zone "Free" to "Ambient"  [kJ/hr]
* 04) Energy rate from zone "Free" to "Ground"  [kJ/hr]
* 05) Temperature of zone "Fix" [degC]
* 06) Energy rate from zone "Fix" to zone "Free" [kJ/hr]
* 07) Energy rate from zone "Fix" to "Ambient"  [kJ/hr]
* 08) Energy rate from zone "Fix" to "Ground"  [kJ/hr]
*==============================================================

*==============================================================
* Hypocaust
*--------------------------------------------------------------
UNIT 2   TYPE 61

PARAMETERS 17
*---------------------
* 01) Logical unit parameter definition file
* 02) Logical unit parameter control file
* 03) Internal timestep [hr]
* 04) Minimum airflow [m3/hr]
* 05) Tolerance on tube temperature [K]
* 06-11) Surface types
* 12-17) Resistance at surface [K m2 hr/kJ]

  2.000E+02     2.010E+02     0.000E+00     0.000E+00     1.000E-02
  1.000E+00     1.000E+00     0.000E+00     0.000E+00     0.000E+00
  0.000E+00     0.000E+00     0.000E+00     0.150E-01     0.000E+00
  0.000E+00     0.000E+00

INPUTS 11
*---------------------
* 01) Air flow [m3/h]
* 02) Air inlet temperature [degC]
* 03) Air inlet humidity [pcent]
* 04) Air pressure [Pa]
* 05) Water infiltration [m3/h]
* 06-11) Surface conditions [degC or W]

  Aflow         Tamb          Hamb          0,0           0,0
  Pfree         Pfix          Tamb          0,0           0,0
  0,0

  0.000E+00     0.000E+00     0.000E+00     1.000E+00     0.000E-03
```

```
  0.000E+00    0.000E+00    Tamb         0.000E+00    0.000E+00
  0.000E+00

* OUPUTS 30
*---------------------
* 01) Temperature of air outlet [degC]
* 02) Humidity of air outlet [pcent]
* 03) Sensible energy rate delivered to ground [kJ/hr]
* 04) Latent energy rate delivered to ground [kJ/hr]
* 05-10) Equivalent border conditions [degC or kJ/hr]
* 11-30) Optional outputs [fct of output type]
*==============================================================

*==============================================================
* Integrator
*--------------------------------------------------------------
  UNIT 3   TYPE 55
  PARAMETERS 7
  1  1  1  1  1E5  1  1E5
  INPUTS 1
  dMwat
  0
*==============================================================

*==============================================================
* Printers
*--------------------------------------------------------------
* PARAMETERS
*---------------------
* 01) Print time interval (>0=hours <0=months)
* 02) Time for start of printer (>0=hours <0=months)
* 03) Time for stop of printer (>0=hours <0=months)
* 04) Logical unit (<=0 for std Line Printer)
*--------------------------------------------------------------
* Printer 1
  UNIT 11   TYPE 25
*---------------------
  PARAMETERS 4
  1.000E+00    0.000E+00    1.000E+05    1.010E+02
  INPUTS 10
  Psbl         Plat         Pfree        Pfix         Pamb
  Pfront       Pback        Pside        Pwat         Pfric
  Psbl         Plat         Pfree        Pfix         Pamb
  Pfront       Pback        Pside        Pwat         Pfric
*--------------------------------------------------------------
* Printer 2
  UNIT 12   TYPE 25
*---------------------
  PARAMETERS 4
  1.000E+00    0.000E+00    1.000E+05    1.020E+02
  INPUTS 10
  PinH         PinG         PinH#        PinG#        Pfree#
  Pfix#        Mwat         Mwat#        dMlat        dMout
  PinH         PinG         PinH#        PinG#        Pfree#
  Pfix#        Mwat         Mwat#        dMlat        dMout
*--------------------------------------------------------------
* Printer 3
  UNIT 13   TYPE 25
```

```
*---------------------
  PARAMETERS 4
  1.000E+00     0.000E+00     1.000E+05     1.030E+02
  INPUTS 10
  Tfree         Tfix          TgFree        TgFix        Tamb
  T1            T2            T3            T4           Tout
  Tfree         Tfix          TgFree        TgFix        Tamb
  T1            T2            T3            T4           Tout
*=================================================================

*****************************************************************
  END
*****************************************************************
```

Observations :

- Linking is done by feeding upper hypocaust surfaces with outflowing energy rates (*Pfree* and *Pfix*) from the two zones and reciprocally feeding building with upper border temperatures (*Tfree* and *Tfix*) from hypocaust.
- Internal energy gains of hypocaust (*PinH, PinH#*) and surrounding ground (*PinG, PinG#*) are each defined by two alternative ways, so as to check for proper energy balance. Same is done for total free water within tubes (*Mwat, Mwat#*) and energy diffused from zones to hypocaust (*Pfree, Pfree#, Pfix, Pfix#*).

<u>Results of simulation</u>

Parameters defined further up and printed in output files are plotted hereafter and show following, expected dynamic :

- Airflow heats up hypocaust (see Fig. 3, *Psbl*). During first hours, energy diffuses from building and surrounding soil into colder hypocaust and as latter warms up diffusion reverses (see Fig. 4, *Pfront, Pback, Pside, Pfree, Pfix*).
- As airflow heats up hypocaust it cools down along the tubes (see Fig. 2, stratification of *Tamb, T1-T4, Tout*) and with time tends to reach equilibrium temperature.
- Warm and humid airflow condensates during first hours (see Fig. 3, *Plat* and Fig. 5, *dMlat, Mlat*). As ground temperature rises, all free water within tubes then evaporates again, after which no latent exchanges take place any more.
- Within Type 61 energy balance is correct (see Fig. 3, *PinH, PinH#, PinG, PinG#*), as is mass balance (see Fig. 5, *Mwat, Mwat#*). Consistency of energy flows between modules is also respected (see Fig. 4, *Pfree, Pfree#, Pfix, Pfix#*).

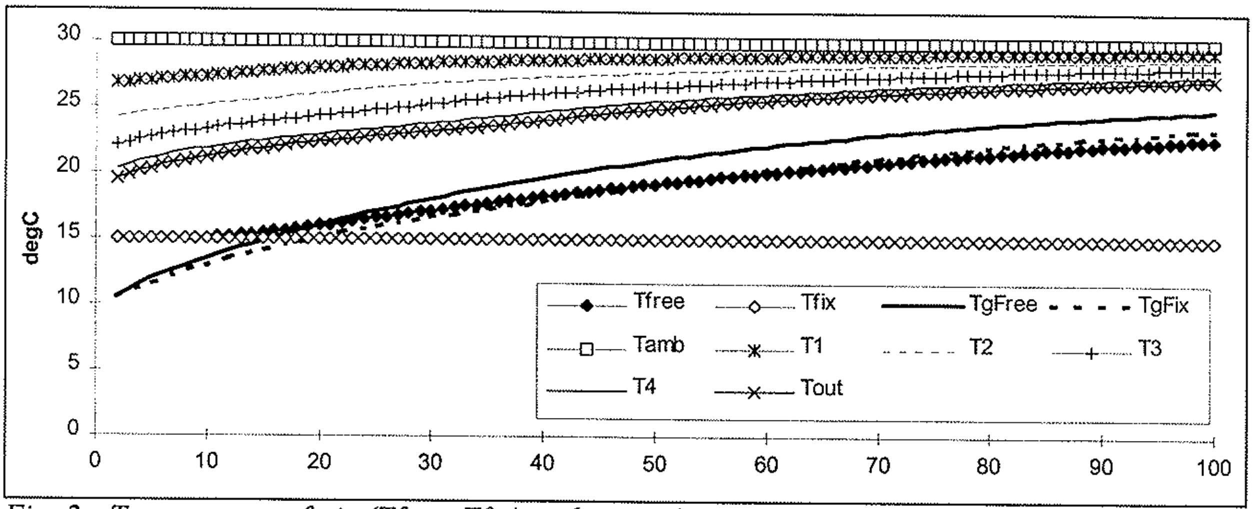


Fig. 2 : Temperature of air (Tfree, Tfix) and ground (TgFree, TgFix) of both zones as well as of airflow along the tubes (T1-T4) and at inlet and outlet (Tamb, Tout).

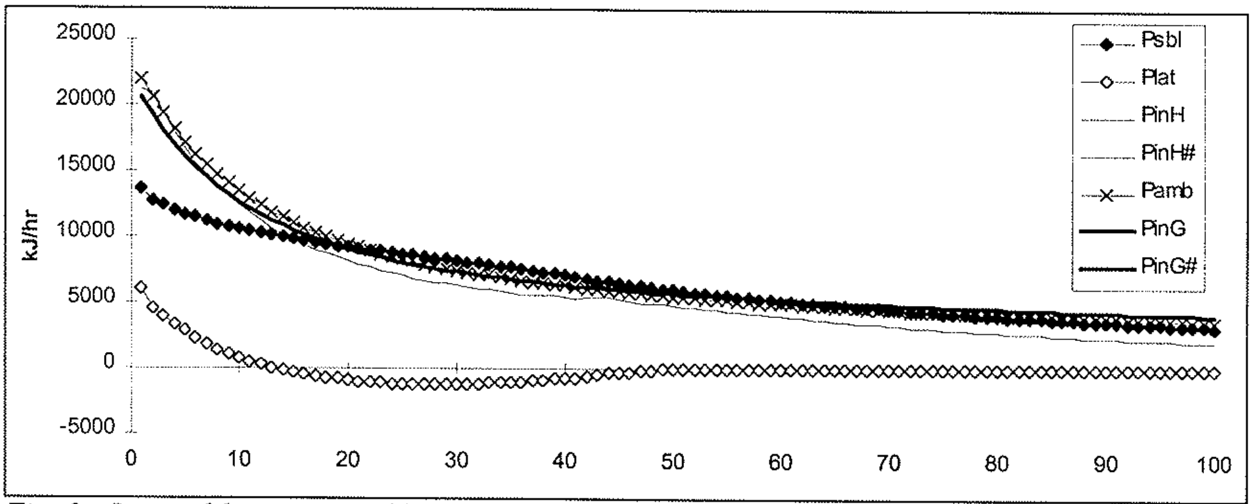


Fig. 3 : Internal heat gains of hypocaust (PinH, PinH#) and surrounding soil (PinG, PinG#), as well as energy entering hypocaust by airflow (Psbl, Plat) and diffused from ambient into surrounding soil (Pamb).

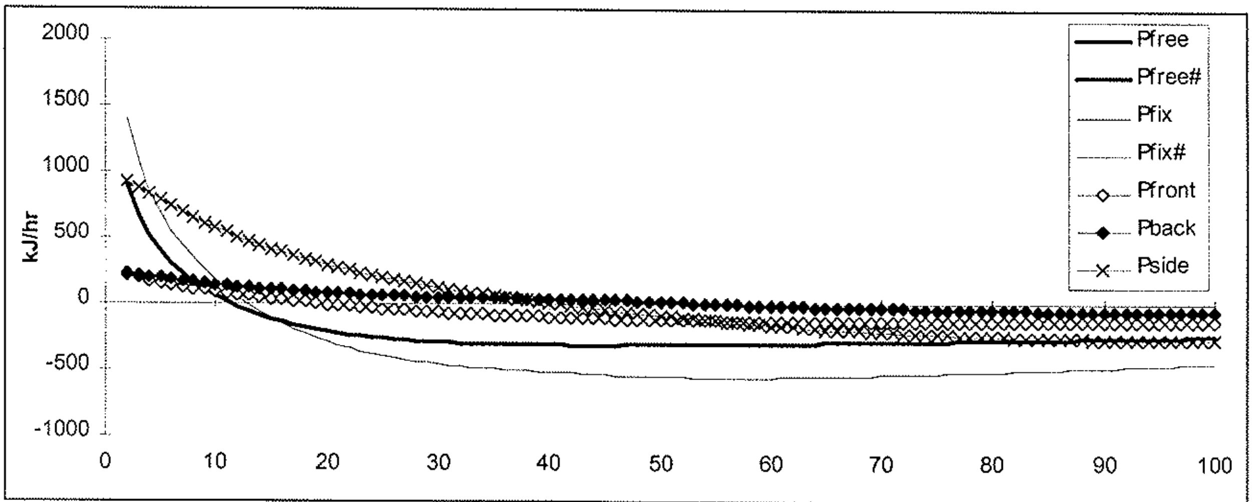


Fig. 4 : Energy entering hypocaust from building (Pfree, Pfree#, Pfix, Pfix#) and from surrounding soil (Pfront, Pback, Pside).

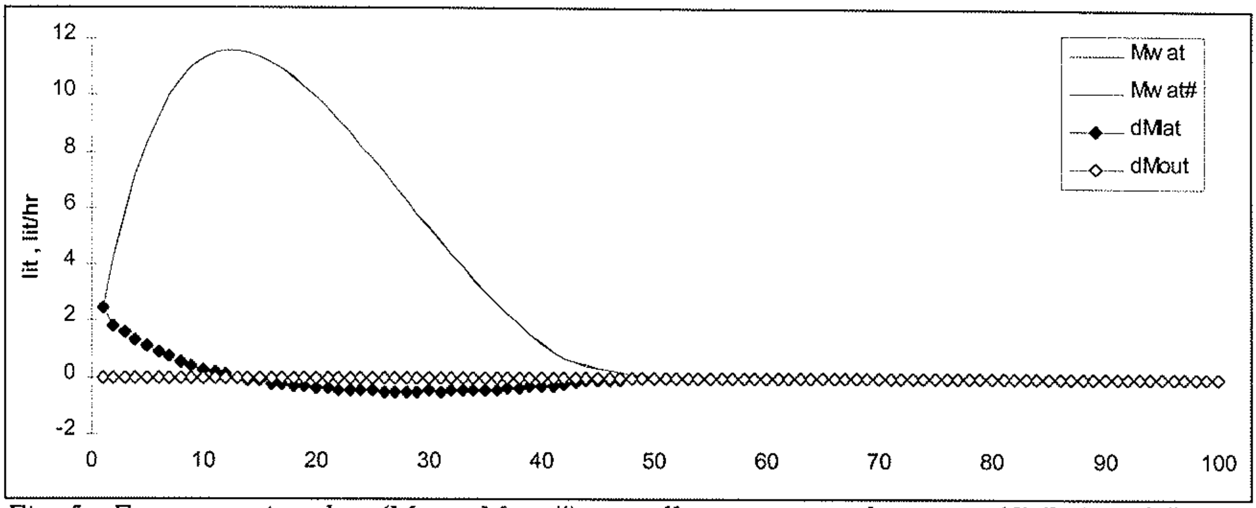


Fig. 5 : Free water in tubes (Mwat, Mwat#) as well as water condensation (dMlat) and flux out of tubes (dMout).

TRNSYS compatible moist air hypocaust model

Part 3 : Source code

```
      !MaxSizes
      !***************************************************************
      PARAMETER(NIMax=40) !max number of nodes along x [-]
      PARAMETER(NJMax=100) !max number of nodes along y (per module) [-]
      PARAMETER(NKMax=20) !max number of nodes along z (per module) [-]
      PARAMETER(NtubMax=20) !max number of tubes (per module) [-]
      PARAMETER(NsoilMax=10) !max number of soiltypes [-]
      PARAMETER(NsurfMax=6) !max number of surfaces [-]
      PARAMETER(NoptMax=100) !max number of optional outputs [-]
      PARAMETER(NiniMax=20) !max number of initialisation conditions [-]

      !PhysConst
      !***************************************************************
      REAL CmAir !specific heat of air [J/kg/K]
      REAL CmWat !specific heat of water [J/kg/K]
      REAL MmolAir !molar mass of air [kg/mol]
      REAL MmolWat !molar mass of water [kg/mol]
      REAL RhoWat !specific mass of water [kg/m3]
      REAL ClatWat !latent heat of water [J/kg]
      REAL CmVap ! specific heat of vapor [J/kg/K]
      REAL Rgas !gas constant for water [J/mol/K]

      COMMON/Type61PhysConst/
     &CmAir,CmWat,MmolAir,MmolWat,RhoWat,ClatWat,CmVap,Rgas

      !Files
      !***************************************************************
      INTEGER IparDef !unit number for parameter definition file [-]
      INTEGER IparCon !unit number for parameter control file [-]
      LOGICAL OparDef !initial status of parameter definition file [-]
      LOGICAL OparCon !initial status of parameter control file [-]

      COMMON/Type61Files/
     &IparDef,IparCon,OparDef,OparCon

      !Time
      !***************************************************************
      INTEGER Isim !general timestep number (Info(8)) [-]
      INTEGER Irep !general timestep repetition number (Info(7)) [-]
      REAL Dt !internal timestep [s]
      REAL DtSoil !max internal timestep for soil [s]
      REAL DtWat !max internal timestep for waterflow [s]
      INTEGER IDt !internal timestep number [-]
      INTEGER NDt !number of internal timesteps [-]

      COMMON/Type61Time/
     &Isim,Irep,Dt,DtSoil,DtWat,IDt,NDt

      !Geometry
      !***************************************************************
      INTEGER Nmod !number of modules [-]
      INTEGER Nsec !number of cross-sections [-]
      INTEGER TypSec(NIMax) !type of cross-sections along x [-]
      INTEGER NI !number of nodes along x [-]
      INTEGER NJ !number of nodes along y (per module) [-]
      INTEGER NK !number of nodes along z (per module) [-]
      REAL DX(0:NIMax+1) !node width along x [m]
      REAL DY(0:NJMax+1) !node width along y [m]
      REAL DZ(0:NKMax+1) !node width along z [m]
      INTEGER I !node index along x [-]
      INTEGER J !node index along y [-]
      INTEGER K !node index along z [-]
      REAL LX !length [m]
      REAL LY !lwidth [m]
      REAL LZ !height [m]
      INTEGER Idir !direction index (1-6) [-]
      COMMON/Type61Geometry/
     &Nmod,Nsec,TypSec,NI,NJ,NK,DX,DY,DZ,I,J,K,LX,LY,LZ,Idir

      !Tubes
      !***************************************************************
      INTEGER Ntub !number of tubes (per module) [-]
      INTEGER Itub !tube index [-]
      INTEGER PosTub(NtubMax,2) !tube position (J,K index) [-]
      REAL ThTub !tube thickness [m]
      REAL Ctub !tube circumference [m]
      REAL CtubCor !correction factor for tube circumference [-]
      REAL Dtub !tube hydraulic diameter [m]
      REAL VolTub !volume of tube node [m3]
      REAL LamTub !tube conductivity [W/m/K]
      REAL CvTub !tube specific heat [J/m3/K]
      REAL Ltub !tube length [m]
      REAL DTtubTol !tolerance on tube node temperature [K]
      REAL Sair !section of tube [m2]
      REAL SairTot !total section of tubes (over all modules) [m2]
      REAL Stub !surface of tube node [m2]
      REAL StubTot !total surface of tubes (over all modules) [m2]
      REAL Zair !airflow distribution factor (not normalised) [m5/2]
```

```
      REAL ZairTot !total of airflow distribution factors (over all modules) [m5/2]
      REAL Rfric !friction coefficent

      COMMON/Type61Tubes/
     &Ntub,Itub,PosTub,ThTub,Ctub,CtubCor,Dtub,VolTub,LamTub,CvTub,Ltub,
     &DTtubTol,Sair,SairTot,Stub,StubTot,Zair,ZairTot,Rfric

      !Surfaces
      !*****************************************************************
      INTEGER Nsurf !number of surfaces [-]
      REAL Xsurf(NsurfMax) !surface condition [degC or W]
      REAL Xbord(NsurfMax) !border condition [W or degC]
      REAL Kbord(NsurfMax) !border conduction coefficient [W/K/m2]
      REAL Sbord(NsurfMax) !total border area [m2]
      INTEGER TypSurf(NsurfMax) !type of surface condition [-]
      REAL Rsurf(NsurfMax) !surface exchange coefficient [K m2/W]

      COMMON/Type61Surfaces/
     &Nsurf,Xsurf,Xbord,Kbord,Sbord,TypSurf,Rsurf

      !Soil
      !*****************************************************************
      INTEGER Nsoil !number of soiltypes [-]
      INTEGER Isoil !soil index [-]
      REAL Tsoil0(NIMax,NJMax,NKMax)
           !initial soil temperature for TRNSYS timestep [degC]
      REAL Tsoil1(0:NIMax+1,0:NJMax+1,0:NKMax+1)
           !initial soil temperature for internal timestep [degC]
      REAL Tsoil2(NIMax,NJMax,NKMax)
           !final soil temperature for internal timestep [degC]
      INTEGER TypSoil(0:NIMax+1,0:NJMax+1,0:NKMax+1) !type of soil/surf. [-]
      REAL VolSoil !node volume [m3]
      REAL Ssoil(6) !lateral area of node [m2]
      REAL Ksoil(6) !conduction coefficient to neighbour node [W/m2/K]
      REAL LamSoil(NsoilMax) !soil conductivity [W/m/K]
      REAL CvSoil(NsoilMax) !soil specific heat [J/m3/K]

      COMMON/Type61Soil/
     &Nsoil,Isoil,Tsoil0,Tsoil1,Tsoil2,TypSoil,VolSoil,Ssoil,Ksoil,
     &LamSoil,CvSoil

      !Air
      !*****************************************************************
      INTEGER Iflow !airflow index [-]
      INTEGER IflowIni !I index of tube beginning [-]
      INTEGER IflowEnd !I index of tube end [-]
      INTEGER DirAir !direction of airflow [-]
      REAL Fair !tube airflow [m3/s]
      REAL FairTot !total airflow (over all modules) [m3/s]
      REAL FairMin !min airflow (over all modules) [m3/s]
      REAL FairMinTub !min airflow for stability [m3/s]
      REAL Vair !air velocity [m/s]
      REAL Kair !air/tube exchange coefficient [W/m2/K]
      REAL Kair0 !air/tube constant exchange coefficient [W/m2/K]
      REAL Kair1 !air/tube linear exchange coefficient [(W/m2/K)/(m/s)]
      REAL RhoAir !specific weight of air [kg/m3]
      REAL PrAir !air pressure [Pa]
      REAL DPrAir !total air pressure loss [Pa]
      REAL Tair !air temperature [degC]
      REAL TairIn !air temperature at hypocaust inlet [degC]
      REAL TairOut !air temperature at hypocaust outlet [degC]
      REAL Hrel !relative humidity [pcent]
      REAL HrelIn !relative humidity at inlet [pcent]
      REAL HrelOut !relative humidity at outlet [pcent]
      REAL Habs !absolute humidity [Pa]
      REAL HabsIn !absolute humidity at inlet [Pa]
      REAL HabsOut !absolute humidity at outlet [Pa]
      REAL Hrat !humidity ratio [kg vapor/kg air]
      REAL HratIn !humidity ratio at inlet [kg vapor/kg air]
      REAL HratOut !humidity ratio at outlet [kg vapor/kg air]
      REAL Hsat !saturating pressure [Pa]
      REAL HsatIn !saturating pressure at inlet [Pa]
      REAL HsatOut !saturating pressure at outlet [Pa]

      COMMON/Type61Air/
     &Iflow,IflowIni,IflowEnd,DirAir,Fair,FairTot,FairMin,FairMinTub,
     &Vair,Kair,Kair0,Kair1,RhoAir,PrAir,DPrAir,Tair,TairIn,TairOut,
     &Hrel,HrelIn,HrelOut,Habs,HabsIn,HabsOut,Hrat,HratIn,HratOut,
     &Hsat,HsatIn,HsatOut

      !Water
      !*****************************************************************
      REAL Vwat(-1:1) !velocity of waterflow (for each airflow type) [m/s]
      INTEGER TypWat(-1:1) !type of waterflow (for each airflow type) [-]
      REAL FwatInfTot !total water infiltration (over all modules) [m3/s]
      REAL SinfTot !total surface of water infiltration (over all modules) [m2]
      INTEGER PosInf(6) !position of water infiltration [-]
      REAL Mwat0(NIMax,NtubMax)
```

```
          !initial watermass for TRNSYS timestep [kg]
     REAL Mwat1(NIMax,NtubMax)
          !initial watermass for internal timestep [kg]
     REAL Mwat2(NIMax,NtubMax)
          !final watermass for internal timestep [kg]
     REAL MwatLat !mass of water cond./evap. in node [kg]
     REAL MwatIn !mass of water flowing into node [kg]
     REAL MwatInf !mass of water infiltrating into node [kg]
     REAL MwatOut !mass of water flowing out of node [kg]

     COMMON/Type61Water/
    &Vwat,TypWat,FwatInfTot,SinfTot,PosInf,Mwat0,Mwat1,Mwat2,
    &MwatLat,MwatIn,MwatInf,MwatOut

     !Energy
     !****************************************************************
     REAL Psbl !sensible power from airflow [W]
     REAL Plat !latent power from airflow [W]
     REAL Psoil(0:6) !diffusive power from neighbor nodes/surfaces [W]
     REAL Pwat !diffusive power from free water [W]
     REAL Pfric !power from frictional losses [W]
     REAL Pint !internal power [W]
     REAL PsblTot !total sensible power (over all modules) [W]
     REAL PlatTot !total latent power (over all modules) [W]
     REAL PsurfTot !total diffusive power from surfaces (over all modules) [W]
     REAL PwatTot !total diffusive power from free water (over all modules) [W]
     REAL PfricTot !total power from frictional losses (over all modules) [W]
     REAL PintTot !total internal power (over all modules) [W]

     COMMON/Type61Energy/
    &Psbl,Plat,Psoil,Pwat,Pfric,Pint,PsblTot,PlatTot,PsurfTot,PwatTot,
    &PfricTot,PintTot

     !Initialisation
     !****************************************************************
     INTEGER NiniSoil !number of initial temperatures [-]
     INTEGER NiniWat !number of initial water thicknesses [-]
     REAL TiniSoil(NiniMax) !initial temperatures [degC]
     REAL ThIniWat(NiniMax) !initial water thicknesses [m]
     INTEGER PosIniSoil(NiniMax,6) !position of initial temperatures [-]
     INTEGER PosIniWat(NiniMax,6) !position of initial water thicknesses [-]
     INTEGER Iini !initialisation index [-]

     COMMON/Type61Initialisation/
    &NiniSoil,NiniWat,TiniSoil,ThIniWat,PosIniSoil,PosIniWat,Iini

     !Optionals
     !****************************************************************
     INTEGER Nopt !number of optional outputs [-]
     INTEGER TypOpt(NoptMax) !type of optional output [-]
     INTEGER PosOpt(NoptMax,6) !position of optional output [-]
     REAL Opt(NoptMax) !optional output [fct of TypOpt]
     INTEGER Iopt !optionals index [-]

     COMMON/Type61Optionals/
    &Nopt,TypOpt,PosOpt,Opt,Iopt
```

```
C***********************************************************************
      SUBROUTINE Type61 (Time,Xin,Xout,T,DtDt,Par,Info,Icntrl,*)

      !VERSION : 1.0

      !DESCRIPTION :
      !Hypocaust model describing sensible and latent exchange between
      !a moist airflow and a buried pipe system, with possibility to
      !use both air directions.
      !
      !AUTHOR :
      !Pierre Hollmuller
      !Centre universitaire d'études des problèmes de l'énergie
      !Université de Genève
      !19, avenue de la Jonction
      !CH - 1205 Genève
      !e-mail: pierre.hollmuller@cuepe.unige.ch
      !-------------------------------------------------------------------

      !common variables:
      !------------------
      INCLUDE 'type61.inc'

      !local variables:
      !------------------
      DOUBLE PRECISION Xin(5+NsurfMax)
      DOUBLE PRECISION Xout(4+NsurfMax+NoptMax)
      REAL Time,T,DtDt
      REAL Par(5+2*NsurfMax)
      INTEGER*4 Info(15)
      INTEGER Icntrl
      !==================================================================

      !checking/conversion of arguments:
      !--------------------------------
      CALL CheckConnections(Info)
      CALL ConvertInput(Time,Xin,Par,Info)

      !initialisation / normal timestep:
      !--------------------------------
      IF (Isim.EQ.1) THEN !initialisation

         !set constants:
         CALL SetPhysicalConstants

         !set parameters:
         CALL OpenParameterFiles
         CALL ReadSuppliedParameters
         CALL SetDerivedParameters
         CALL SetSimulationParameters
         CALL CloseParameterFiles
         !initialise soil & water:
         CALL InitialiseSoil
         CALL InitialiseWater

      ELSE !normal timestep

         CALL ResetOutputs

         DO IDt=1,NDt

            !updates from previous timestep/surfaces:
            CALL UpdateSoil
            CALL UpdateWater
            CALL SetSurfaces

            !evolution of air/tube:
            DO Itub=1,Ntub
               CALL AirInput
               DO Iflow=IflowIni,IflowEnd
                  CALL TubeProperties
                  CALL TubeEvolution
                  CALL OptionalsTube
               END DO
               CALL AirOutput
            END DO

            !evolution of soil:
            DO I=1,NI
            DO J=1,NJ
            DO K=1,NK
            IF (TypSoil(I,J,K).NE.0) THEN
               CALL SoilProperties
               CALL SoilEvolution
               CALL OptionalsSoil
               CALL SetBorders
            END IF
            END DO
```

```
            END DO
            END DO

            !miscellaneous:
            CALL OptionalsMiscellaneous

         END DO

      END IF

      !checking/conversion of arguments:
      !--------------------------------
      CALL ConvertOutput(Xout)

      RETURN
      END
C*************************************************************************

C*************************************************************************
      SUBROUTINE CheckConnections(Info)

      !common variables:
      !-----------------
      INCLUDE 'type61.inc'

      !local variables:
      !----------------
      INTEGER*4 Info(15)
      CHARACTER*3 Ycheck(5+NsurfMax),Ocheck(4+NsurfMax+NoptMax)
      !=======================================================================

      !Units for optional outputs are read in parameter file during
      !1st simulation step, so checking can only be done at 2nd step:
      IF (Isim.EQ.2) THEN

         !Ycheck:
         !-------
         Ycheck(1)='VF1' !FairTot [m3/hr]
         Ycheck(2)='TE1' !TairIn [degC]
         Ycheck(3)='PC1' !HrelIn [pcent]
         Ycheck(4)='PR1' !PrAir [bar]
         Ycheck(5)='VF1' !FwatInfTot [m3/hr]
         DO Isurf=1,NsurfMax
            IF (TypSurf(Isurf).EQ.0) THEN
               Ycheck(5+Isurf)='TE1' !Xsurf [degC]
            ELSE
               Ycheck(5+Isurf)='PW1' !Xsurf [kJ/hr]
            END IF
         END DO

         !Ocheck:
         !-------
         Ocheck(1)='TE1' !TairOut [degC]
         Ocheck(2)='PC1' !HrelOut [pcent]
         Ocheck(3)='PW1' !PsblTot [kJ/hr]
         Ocheck(4)='PW1' !PlatTot [kJ/hr]
         DO Isurf=1,NsurfMax
            IF (TypSurf(Isurf).EQ.0) THEN
               Ocheck(5+Isurf)='PW1' !Xbord [kJ/hr]
            ELSE
               Ocheck(5+Isurf)='TE1' !Xbord [degC]
            END IF
         END DO
         DO Iopt=1,NoptMax
            !Tube:
            IF (TypOpt(Iopt).EQ.  1) Ocheck(Iopt)='TE1' !Tair [degC]
            IF (TypOpt(Iopt).EQ.  2) Ocheck(Iopt)='PC1' !Hrel [pcent]
            IF (TypOpt(Iopt).EQ.  3) Ocheck(Iopt)='PR1' !Habs [bar]
            IF (TypOpt(Iopt).EQ.  4) Ocheck(Iopt)='DM1' !Hrat [kg vapor/kg air]
            IF (TypOpt(Iopt).EQ.  5) Ocheck(Iopt)='VL1' !Mwat [m3]
            IF (TypOpt(Iopt).EQ.  6) Ocheck(Iopt)='VF1' !MwatLat [m3/hr]
            IF (TypOpt(Iopt).EQ.  7) Ocheck(Iopt)='VF1' !MwatIn [m3/hr]
            IF (TypOpt(Iopt).EQ.  8) Ocheck(Iopt)='VF1' !MwatInf [m3/hr]
            IF (TypOpt(Iopt).EQ.  9) Ocheck(Iopt)='VF1' !MwatOut [m3/hr]
            IF (TypOpt(Iopt).EQ. 10) Ocheck(Iopt)='TE1' !Tsoil [degC]
            IF (TypOpt(Iopt).EQ. 11) Ocheck(Iopt)='PW1' !Psbl [kJ/hr]
            IF (TypOpt(Iopt).EQ. 12) Ocheck(Iopt)='PW1' !Plat [kJ/hr]
            IF (TypOpt(Iopt).EQ. 13) Ocheck(Iopt)='PW1' !Pwat [kJ/hr]
            IF (TypOpt(Iopt).EQ. 14) Ocheck(Iopt)='PW1' !Pfric [kJ/hr]
            IF (TypOpt(Iopt).EQ. 15) Ocheck(Iopt)='PW1' !Psoil(0) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 16) Ocheck(Iopt)='PW1' !Psoil(1) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 17) Ocheck(Iopt)='PW1' !Psoil(2) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 18) Ocheck(Iopt)='PW1' !Psoil(3) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 19) Ocheck(Iopt)='PW1' !Psoil(4) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 20) Ocheck(Iopt)='PW1' !Psoil(5) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 21) Ocheck(Iopt)='PW1' !Psoil(6) [kJ/hr]
            IF (TypOpt(Iopt).EQ. 22) Ocheck(Iopt)='PW1' !Pint [kJ/hr]
            IF (TypOpt(Iopt).EQ. 23) Ocheck(Iopt)='VF1' !Fair [m3/hr]
```

```
            IF (TypOpt(Iopt).EQ. 24) Ocheck(Iopt)='VE1' !Vair [m/s]
            !Soil:
            IF (TypOpt(Iopt).EQ.101) Ocheck(Iopt)='TE1' !Tsoil [degC]
            IF (TypOpt(Iopt).EQ.102) Ocheck(Iopt)='PW1' !Psoil(0) [kJ/hr]
            IF (TypOpt(Iopt).EQ.103) Ocheck(Iopt)='PW1' !Psoil(1) [kJ/hr]
            IF (TypOpt(Iopt).EQ.104) Ocheck(Iopt)='PW1' !Psoil(2) [kJ/hr]
            IF (TypOpt(Iopt).EQ.105) Ocheck(Iopt)='PW1' !Psoil(3) [kJ/hr]
            IF (TypOpt(Iopt).EQ.106) Ocheck(Iopt)='PW1' !Psoil(4) [kJ/hr]
            IF (TypOpt(Iopt).EQ.107) Ocheck(Iopt)='PW1' !Psoil(5) [kJ/hr]
            IF (TypOpt(Iopt).EQ.108) Ocheck(Iopt)='PW1' !Psoil(6) [kJ/hr]
            IF (TypOpt(Iopt).EQ.109) Ocheck(Iopt)='PW1' !Pint [kJ/hr]
            !Miscelaneous:
            IF (TypOpt(Iopt).EQ.201) Ocheck(Iopt)='PW1' !PsurfTot [kJ/hr]
            IF (TypOpt(Iopt).EQ.202) Ocheck(Iopt)='PW1' !PwatTot [kJ/hr]
            IF (TypOpt(Iopt).EQ.203) Ocheck(Iopt)='PW1' !PfricTot [kJ/hr]
            IF (TypOpt(Iopt).EQ.204) Ocheck(Iopt)='PW1' !PintTot [kJ/hr]
         END DO

         CALL Typeck(1,Info,5+NsurfMax,5+2*NsurfMax,0)
         CALL Rcheck(Info,Ycheck,Ocheck)

      END IF

      RETURN
      END
C*******************************************************************

C*******************************************************************
      SUBROUTINE ConvertInput(Time,Xin,Par,Info)

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'

      !local variables:
      !-------------------
      DOUBLE PRECISION Xin(5+NsurfMax)
      REAL Par(5+2*NsurfMax)
      INTEGER*4 Info(15)
      CHARACTER*3 Ycheck(5+NsurfMax),Ocheck(4+NsurfMax+NoptMax)
      !============================================================

      !Info:
      !-----
      Isim=Info(8)
C     Irep=Info(7)
C     previous line is replaced by following ones because of a bug
C     in main TRNSYS program, version 14.1 and 14.2, which resets Info(7)
C     to 0 at the end of 1st timestep.
      IF (Isim.EQ.1) THEN
         Irep=-1
         TimePrev=Time
      ELSE IF (Time.EQ.TimePrev) THEN
         Irep=Irep+1
      ELSE
         Irep=0
         TimePrev=Time
      END IF
      Info(6)=5+NsurfMax+NoptMax

      !Parameters:
      !-----------
      IF (Isim.EQ.1) THEN
         IparDef=INT(Par(1)) ! [-]
         IparCon=INT(Par(2)) ! [-]
         Dt=Par(3)*3600 ! [s] <- [hr]
         FairMin=Par(4)/3600 ! [m3/s] <- [m3/hr]
         DTtubTol=Par(5) ! [K]
         DO Isurf=1,NsurfMax
            TypSurf(Isurf)=Par(5+Isurf) ! [-]
         END DO
         DO Isurf=1,NsurfMax
            Rsurf(Isurf)=Par(5+NsurfMax+Isurf)/3.6 ! [W/m2 K] <- [kJ/hr m2 K]
         END DO
      END IF

      !Inputs:
      !-------
      FairTot=0
      DirAir=0
      IF (Info(8).GT.1) THEN
      IF (ABS(Xin(1)/3600).GE.FairMin) THEN
         FairTot=ABS(Xin(1)/3600) ! [m3/s] <- [m3/hr]
         DirAir=INT(Xin(1)/ABS(Xin(1))) ! [-]
      END IF
      END IF

      TairIn=Xin(2) ! [degC]
```

```
      HrelIn=Xin(3) ! [pcent]
      PrAir=Xin(4)*1E5 ! [Pa] <- [bar]

      IF (Info(8).GT.1) THEN
         FwatInfTot=Xin(5)/3600 ! [m3/s] <- [m3/hr]
      ELSE
         FwatInfTot=0
      END IF

      DO Isurf=1,NsurfMax
         IF (TypSurf(Isurf).EQ.0) THEN
            Xsurf(Isurf)=Xin(5+Isurf) ! [degC] <- [degC]
         ELSE
            Xsurf(Isurf)=Xin(5+Isurf)/3.6 ! [W] <- [kJ/hr]
         END IF
      END DO

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE ConvertOutput(Xout)

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'

      !local variables:
      !-------------------
      DOUBLE PRECISION Xout(4+NsurfMax+NoptMax)
      !=================================================================

      Xout(1)=TairOut ! [degC]
      Xout(2)=HrelOut ! [pcent]
      Xout(3)=PsblTot*3.6 ! [kJ/hr] <- [W]
      Xout(4)=PlatTot*3.6 ! [kJ/hr] <- [W]
      DO Isurf=1,Nsurf
         IF (TypSurf(Isurf).EQ.0) THEN
            Xout(4+Isurf)=Xbord(Isurf)*3.6 ! [kJ/hr] <- [W]
         ELSE
            Xout(4+Isurf)=Xbord(Isurf) ! [degC]
         END IF
      END DO
      DO Iopt=1,Nopt
         Xout(4+NsurfMax+Iopt)=Opt(Iopt) ! cf. SetOptional for units
      END DO

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE ResetOutputs

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !=================================================================

      PsblTot=0
      PlatTot=0
      PsurfTot=0
      PwatTot=0
      PfricTot=0
      PintTot=0

      DO Isurf=1,Nsurf
         Xbord(Isurf)=0
      END DO

      DO Iopt=1,Nopt
         Opt(Iopt)=0
      END DO

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE UpdateSoil

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !=================================================================

      IF (IDt.EQ.1) THEN
```

```
            !reset:
            DO I=1,NI
            DO J=1,NJ
            DO K=1,NK
               IF (Irep.EQ.0) THEN
                  Tsoil0(I,J,K)=Tsoil2(I,J,K)
               END IF
               Tsoil1(I,J,K)=Tsoil0(I,J,K)
            END DO
            END DO
            END DO
         ELSE
            !update:
            DO I=1,NI
            DO J=1,NJ
            DO K=1,NK
               Tsoil1(I,J,K)=Tsoil2(I,J,K)
            END DO
            END DO
            END DO
         END IF

         RETURN
         END
C************************************************************************

C************************************************************************
         SUBROUTINE UpdateWater

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'
         !================================================================

         IF (IDt.EQ.1) THEN
            !reset:
            DO I=IflowIni,IflowEnd
            DO Itub=1,Ntub
               IF (Irep.EQ.0) THEN
                  Mwat0(I,Itub)=Mwat2(I,Itub)
                 END IF
               Mwat1(I,Itub)=Mwat0(I,Itub)
            END DO
            END DO
         ELSE
            !update:
            DO I=IflowIni,IflowEnd
            DO Itub=1,Ntub
               Mwat1(I,Itub)=Mwat2(I,Itub)
            END DO
            END DO
         END IF

         RETURN
         END
C************************************************************************

C************************************************************************
         SUBROUTINE SetSurfaces

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'

         !local variables:
         !-------------------
         INTEGER Isurf
         !================================================================

         DO I=1,NI
         DO J=1,NJ

            Isurf=TypSoil(I,J,0)
            IF (Isurf.GT.0) THEN
               IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
                  Tsoil1(I,J,0)=Xsurf(Isurf)
               ELSE !input power:
                  Tsoil1(I,J,0)=Tsoil1(I,J,1)
     &            +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
               END IF
            END IF

            Isurf=TypSoil(I,J,NK+1)
            IF (Isurf.GT.0) THEN
               IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
                  Tsoil1(I,J,NK+1)=Xsurf(Isurf)
               ELSE !input power:
                  Tsoil1(I,J,NK+1)=Tsoil1(I,J,NK)
```

```
     &          +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
            END IF
         END IF

      END DO
      END DO

      DO I=1,NI
      DO K=1,NK

         Isurf=TypSoil(I,0,K)
         IF (Isurf.GT.0) THEN
            IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
               Tsoil1(I,0,K)=Xsurf(Isurf)
            ELSE !input power:
               Tsoil1(I,0,K)=Tsoil1(I,1,K)
     &          +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
            END IF
         END IF

         Isurf=TypSoil(I,NJ+1,K)
         IF (Isurf.GT.0) THEN
            IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
               Tsoil1(I,NJ+1,K)=Xsurf(Isurf)
            ELSE !input power:
               Tsoil1(I,NJ+1,K)=Tsoil1(I,NJ,K)
     &          +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
            END IF
         END IF

      END DO
      END DO

      DO J=1,NJ
      DO K=1,NK

         Isurf=TypSoil(0,J,K)
         IF (Isurf.GT.0) THEN
            IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
               Tsoil1(0,J,K)=Xsurf(Isurf)
            ELSE !input power:
               Tsoil1(0,J,K)=Tsoil1(1,J,K)
     &          +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
            END IF
         END IF

         Isurf=TypSoil(NI+1,J,K)
         IF (Isurf.GT.0) THEN
            IF (TypSurf(Isurf).EQ.0) THEN !input temperature:
               Tsoil1(NI+1,J,K)=Xsurf(Isurf)
            ELSE !input power:
               Tsoil1(NI+1,J,K)=Tsoil1(NI,J,K)
     &          +Xsurf(Isurf)/Sbord(Isurf)/Kbord(Isurf)
            END IF
         END IF

      END DO
      END DO

      RETURN
      END
C*************************************************************************

C*************************************************************************
      SUBROUTINE AirInput

      !common variables:
      !--------------------
      INCLUDE 'type61.inc'

      !local variables:
      !-------------------
      REAL Hsat$,Hrat$,Habs$
      !=================================================================

      IF ((IDt.EQ.1).AND.(Itub.EQ.1)) THEN

         !psychometrics:
         HsatIn=Hsat$(TairIn)
         HabsIn=HrelIn*HsatIn/100
         HratIn=Hrat$(HabsIn,PrAir)

         !general flow variables:
         RhoAir=PrAir*MmolAir/Rgas/(Tair+273.15)
         DPrAir=Rfric*Ltub*RhoAir/2*(FairTot/ZairTot)**2
         PfricTot=FairTot*DPrAir

      END IF
```

```
      Tair=TairIn
      Hrel=HrelIn
      Hsat=HsatIn
      Habs=HabsIn
      Hrat=HratIn

      Iflow=IflowIni
      CALL TubeProperties
      Fair=FairTot*Zair/ZairTot
      Vair=Fair/Sair
      Kair=Kair0+Kair1*Vair

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE TubeProperties

      !common variables:
      !------------------
      INCLUDE 'type61.inc'
      !=========================================================

      IF (DirAir.GE.0) THEN
         I=Iflow
      ELSE
         I=IflowEnd+IflowIni-Iflow
      END IF
      J=PosTub(Itub,1)
      K=PosTub(Itub,2)

      IF ((J.EQ.1).OR.(J.EQ.NJ)) THEN
         Ssoil(1)=(2*DY(J)+DZ(K))*CtubCor*ThTub
         Ssoil(2)=(2*DY(J)+DZ(K))*CtubCor*ThTub
      ELSE
         Ssoil(1)=2*(DY(J)+DZ(K))*CtubCor*ThTub
         Ssoil(2)=2*(DY(J)+DZ(K))*CtubCor*ThTub
      END IF
      IF (I.EQ.IflowIni) THEN
         Ksoil(1)=0
      ELSE
         Ksoil(1)=2*LamTub/(DX(I-1)+DX(I))
      END IF
      IF (I.EQ.IflowEnd) THEN
         Ksoil(2)=0
      ELSE
         Ksoil(2)=2*LamTub/(DX(I+1)+DX(I))
      END IF

      IF (J.EQ.1) THEN
         Ssoil(3)=0
         Ksoil(3)=0
      ELSE
         Ssoil(3)=DX(I)*DZ(K)*CtubCor
         Ksoil(3)=1/(DY(J-1)/2/LamSoil(TypSoil(I,J-1,K))+ThTub/LamTub)
      END IF
      IF (J.EQ.NJ) THEN
         Ssoil(4)=0
         Ksoil(4)=0
      ELSE
         Ssoil(4)=DX(I)*DZ(K)*CtubCor
         Ksoil(4)=1/(DY(J+1)/2/LamSoil(TypSoil(I,J+1,K))+ThTub/LamTub)
      END IF

      Ssoil(5)=DX(I)*DY(J)*CtubCor
      Ksoil(5)=1/(DZ(K-1)/2/LamSoil(TypSoil(I,J,K-1))+ThTub/LamTub)
      Ssoil(6)=DX(I)*DY(J)*CtubCor
      Ksoil(6)=1/(DZ(K+1)/2/LamSoil(TypSoil(I,J,K+1))+ThTub/LamTub)

      Sair=DY(J)*DZ(K)
      VolTub=Ssoil(1)*DX(I)
      Stub=Ssoil(3)+Ssoil(4)+Ssoil(5)+Ssoil(6)
      Ctub=Stub/DX(I)
      Dtub=4*Sair/Ctub
      Zair=Sair*SQRT(Dtub)

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE TubeEvolution

      !common variables:
      !------------------
      INCLUDE 'type61.inc'
```

```

      !local variables:
      !-------------------
      REAL Hsat$,Hrat$,Habs$

      REAL MwatSat !mass of water necessary to satuarte node air [kg]
      REAL MwatLew !theoretical mass of water cond./evap. in node [kg]
      !================================================================

      !input water:
      !------------

      !flow from previous node:
      IF (Iflow.EQ.IflowIni) THEN
         MwatIn=0
      ELSE IF (TypWat(DirAir).EQ.1) THEN !flow
         MwatIn=MwatOut
      ELSE !ejection
         MwatIn=0
      END IF

      !infiltration
      MwatInf=0
      IF ((I.GE.PosInf(1)).AND.(I.LE.PosInf(4))) THEN
      IF ((J.GE.PosInf(2)).AND.(J.LE.PosInf(5))) THEN
      IF ((K.GE.PosInf(3)).AND.(K.LE.PosInf(6))) THEN
         MwatInf=FwatInfTot*Dt*RhoWat*Stub/SinfTot
      END IF
      END IF
      END IF

      !energy balance of tube :
      !------------------------
      DTairtub=MAX(ABS(Tair-Tsoil1(I,J,K)),0.5)
100   Tmax=Tair+DTairtub
      Tmin=Tair-DTairtub
      Ibal=0
      DO WHILE (Ibal.NE.2)

         CALL FindZero(Balance,Tsoil2(I,J,K),Tmin,Tmax,DTtubTol,Ibal)
         IF (Ibal.EQ.3) THEN
            DTairtub=2*DTairtub
            GOTO 100
         END IF

         !latent and sensible heat from air:
         IF (DirAir.NE.0) THEN
            HratSatTub=Hrat$(Hsat$(Tsoil2(I,J,K)),PrAir)
            HratSatAir=Hrat$(Hsat$(Tair),PrAir)
            IF ((Hrat.GT.HratSatTub).OR.(Hrat.GT.HratSatAir)) THEN
               !condensation:
               MwatLew=Dt*(Hrat-HratSatTub)*Kair*Stub/CmAir
               MwatSat=(Hrat-HratSatAir)*RhoAir*Sair*Vair*Dt
               MwatLat=MAX(MwatLew,MwatSat)
            ELSE
               !evaporation:
               MwatLew=Dt*(HratSatTub-Hrat)*Kair*Stub/CmAir
               MwatSat=(HratSatAir-Hrat)*RhoAir*Sair*Vair*Dt
               MwatLat=-MIN(MwatLew,MwatSat,
     &                      Mwat1(I,Itub)+MwatIn+MwatInf)
            END IF
            Plat=ClatWat*MwatLat/Dt
            Psbl=Stub*Kair*(Tair-Tsoil2(I,J,K))
         ELSE
            MwatLew=0
            MwatSat=0
            MwatLat=0
            Plat=0
            Psbl=0
         END IF

         !diffusive heat from water:
         Pwat=(Mwat1(I,Itub)*(Tsoil1(I,J,K)        -Tsoil2(I,J,K))
     &        +MwatIn      *(Tsoil2(I-DirAir,J,K)-Tsoil2(I,J,K)))
     &        *CmWat/Dt

         !diffusive heat from soil:
         Psoil(1)=Ksoil(1)*Ssoil(1)*(Tsoil1(I-1,J,K)-Tsoil2(I,J,K))
         Psoil(2)=Ksoil(2)*Ssoil(2)*(Tsoil1(I+1,J,K)-Tsoil2(I,J,K))
         Psoil(3)=Ksoil(3)*Ssoil(3)*(Tsoil1(I,J-1,K)-Tsoil2(I,J,K))
         Psoil(4)=Ksoil(4)*Ssoil(4)*(Tsoil1(I,J+1,K)-Tsoil2(I,J,K))
         Psoil(5)=Ksoil(5)*Ssoil(5)*(Tsoil1(I,J,K-1)-Tsoil2(I,J,K))
         Psoil(6)=Ksoil(6)*Ssoil(6)*(Tsoil1(I,J,K+1)-Tsoil2(I,J,K))
         Psoil(0)=Psoil(1)+Psoil(2)+Psoil(3)+Psoil(4)+Psoil(5)
     &           +Psoil(6)

         !internal heat:
         Pint=CvTub*VolTub*(Tsoil2(I,J,K)-Tsoil1(I,J,K))/Dt
```

```
            !energy balance:
            Balance=Pint-Plat-Psbl-Psoil(0)-Pwat

         END DO

         !update Tsoil1 (equivalence of Psoil in routines TubeEvolution and SoilEvolution):
         Tsoil1(I,J,K)=Tsoil2(I,J,K)

         !mass balance of water:
         !----------------------
         IF (TypWat(DirAir).EQ.1) THEN !flow
            MwatOut=(Mwat1(I,Itub)+MwatLat+MwatIn+MwatInf)
     &              *Vwat(DirAir)*Dt/DX(I)
         ELSE !ejection
            MwatOut=Mwat1(I,Itub)+MwatLat+MwatIn+MwatInf
         END IF
         Mwat2(I,Itub)=Mwat1(I,Itub)+MwatLat+MwatIn+MwatInf-MwatOut-MwatEje

         !energy balance of air :
         !-----------------------
         IF (DirAir.NE.0) THEN
            Pfric=PfricTot*(Fair/FairTot)*(DX(I)/Ltub)
            Tair=Tair+(Pfric-Psbl)/(CmAir+Hrat*CmVap)/(RhoAir*Sair*Vair)
            Hrat=Hrat-MwatLat/(RhoAir*Sair*Vair*Dt)
            Habs=Habs$(Hrat,PrAir)
            Hsat=Hsat$(Tair)
            Hrel=100*Habs/Hsat
         ELSE
            Pfric=0
            Tair=TairIn
            Hrat=HratIn
            Habs=HabsIn
            Hsat=HsatIn
            Hrel=HrelIn
         END IF

         !integrals :
         !-----------
         PsblTot=PsblTot+Psbl*Nmod/NDt
         PlatTot=PlatTot+Plat*Nmod/NDt
         PwatTot=PwatTot+Pwat*Nmod/NDt
         PintTot=PintTot+Pint*Nmod/NDt

         RETURN
         END
C*********************************************************************

C*********************************************************************
         SUBROUTINE AirOutput

         !common variables:
         !-----------------
         INCLUDE 'type61.inc'

         !local variables:
         !----------------
         REAL Hsat$,Hrat$,Habs$
         !============================================================

         IF ((IDt.EQ.1).AND.(Itub.EQ.1)) THEN
            TairOut=0
            HratOut=0
         END IF

         IF (DirAir.NE.0) THEN

            !air mix:
            TairOut=TairOut + Tair*(Nmod*Sair/SairTot)/NDt
            HratOut=HratOut + Hrat*(Nmod*Sair/SairTot)/NDt
            IF ((IDt.EQ.NDt).AND.(Itub.EQ.Ntub)) THEN
               !condensation of air mix:
               HratOutMax=Hrat$(Hsat$(TairOut),PrAir)
               HratOut=MIN(HratOut,HratOutMax)
               !other psychometric data:
               HabsOut=Habs$(HratOut,PrAir)
               HsatOut=Hsat$(TairOut)
               HrelOut=100*HabsOut/HsatOut
            END IF

         ELSE

            TairOut=TairIn
            HratOut=HratIn
            HabsOut=HabsIn
            HsatOut=HsatIn
            HrelOut=HrelIn

```

```
      END IF

      RETURN
      END
C**********************************************************************

C**********************************************************************
      SUBROUTINE SoilProperties

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !================================================================

      Ssoil(1)=DY(J)*DZ(K)
      Ssoil(2)=DY(J)*DZ(K)

      IF (I.EQ.1) THEN !border
         IF (TypSoil(0,J,K).EQ.0) THEN !adiabatic
            Ksoil(1)=0
         ELSE !surface condition
            Ksoil(1)=1/(DX(I)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(0,J,K)))
         END IF
      ELSE IF (TypSoil(I-1,J,K).EQ.0) THEN !soil-tube
         Ksoil(1)=0
      ELSE !soil-soil
         Ksoil(1)=2/(DX(I)/LamSoil(TypSoil(I,J,K))
     &             + DX(I-1)/LamSoil(TypSoil(I-1,J,K)))
      END IF

      IF (I.EQ.NI) THEN !border
         IF (TypSoil(NI+1,J,K).EQ.0) THEN !adiabatic
            Ksoil(2)=0
         ELSE !surface condition
            Ksoil(2)=1/(DX(I)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(NI+1,J,K)))
         END IF
      ELSE IF (TypSoil(I+1,J,K).EQ.0) THEN !soil-tube
         Ksoil(2)=0
      ELSE !soil-soil
         Ksoil(2)=2/(DX(I)/LamSoil(TypSoil(I,J,K))
     &             + DX(I+1)/LamSoil(TypSoil(I+1,J,K)))
      END IF

      IF (J.EQ.1) THEN !border
         Ssoil(3)=DX(I)*DZ(K)
         IF (TypSoil(I,0,K).EQ.0) THEN !adiabatic
            Ksoil(3)=0
         ELSE !surface condition
            Ksoil(3)=1/(DY(J)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(I,0,K)))
         END IF
      ELSE IF (TypSoil(I,J-1,K).EQ.0) THEN !soil-tube
         Ssoil(3)=DX(I)*DZ(K)*CtubCor
         Ksoil(3)=1/(DY(J)/2/LamSoil(TypSoil(I,J,K)) + ThTub/LamTub)
      ELSE !soil-soil
         Ssoil(3)=DX(I)*DZ(K)
         Ksoil(3)=2/(DY(J)/LamSoil(TypSoil(I,J,K))
     &             + DY(J-1)/LamSoil(TypSoil(I,J-1,K)))
      END IF

      IF (J.EQ.NJ) THEN !border
         Ssoil(4)=DX(I)*DZ(K)
         IF (TypSoil(I,NJ+1,K).EQ.0) THEN !adiabatic
            Ksoil(4)=0
         ELSE !surface condition
            Ksoil(4)=1/(DY(J)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(0,NJ+1,K)))
         END IF
      ELSE IF (TypSoil(I,J+1,K).EQ.0) THEN !soil-tube
         Ssoil(4)=DX(I)*DZ(K)*CtubCor
         Ksoil(4)=1/(DY(J)/2/LamSoil(TypSoil(I,J,K)) + ThTub/LamTub)
      ELSE !soil-soil
         Ssoil(4)=DX(I)*DZ(K)
         Ksoil(4)=2/(DY(J)/LamSoil(TypSoil(I,J,K))
     &             + DY(J+1)/LamSoil(TypSoil(I,J+1,K)))
      END IF

      IF (K.EQ.1) THEN !border
         Ssoil(5)=DX(I)*DY(J)
         IF (TypSoil(I,J,0).EQ.0) THEN !adiabatic
            Ksoil(5)=0
         ELSE !surface condition
            Ksoil(5)=1/(DZ(K)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(I,J,0)))
         END IF
      ELSE IF (TypSoil(I,J,K-1).EQ.0) THEN !soil-tube
```

```
            Ssoil(5)=DX(I)*DY(J)*CtubCor
            Ksoil(5)=1/(DZ(K)/2/LamSoil(TypSoil(I,J,K)) + ThTub/LamTub)
         ELSE !soil-soil
            Ssoil(5)=DX(I)*DY(J)
            Ksoil(5)=2/(DZ(K)/LamSoil(TypSoil(I,J,K))
     &              + DZ(K-1)/LamSoil(TypSoil(I,J,K-1)))
         END IF

         IF (K.EQ.NK) THEN !border
            Ssoil(6)=DX(I)*DY(J)
            IF (TypSoil(I,J,NK+1).EQ.0) THEN !adiabatic
               Ksoil(6)=0
            ELSE !surface condition
               Ksoil(6)=1/(DZ(K)/2/LamSoil(TypSoil(I,J,K))
     &                    + Rsurf(TypSoil(I,J,NK+1)))
            END IF
         ELSE IF (TypSoil(I,J,K+1).EQ.0) THEN !soil-tube
            Ssoil(6)=DX(I)*DY(J)*CtubCor
            Ksoil(6)=1/(DZ(K)/2/LamSoil(TypSoil(I,J,K)) + ThTub/LamTub)
         ELSE !soil-soil
            Ssoil(6)=DX(I)*DY(J)
            Ksoil(6)=2/(DZ(K)/LamSoil(TypSoil(I,J,K))
     &              + DZ(K+1)/LamSoil(TypSoil(I,J,K+1)))
         END IF

         VolSoil=DX(I)*DY(J)*DZ(K)

         RETURN
         END
C*************************************************************************

C*************************************************************************
         SUBROUTINE SoilEvolution

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'
         !===============================================================

         !energy balance :
         !----------------
         Psoil(1)=Ksoil(1)*Ssoil(1)*(Tsoil1(I-1,J,K)-Tsoil1(I,J,K))
         Psoil(2)=Ksoil(2)*Ssoil(2)*(Tsoil1(I+1,J,K)-Tsoil1(I,J,K))
         Psoil(3)=Ksoil(3)*Ssoil(3)*(Tsoil1(I,J-1,K)-Tsoil1(I,J,K))
         Psoil(4)=Ksoil(4)*Ssoil(4)*(Tsoil1(I,J+1,K)-Tsoil1(I,J,K))
         Psoil(5)=Ksoil(5)*Ssoil(5)*(Tsoil1(I,J,K-1)-Tsoil1(I,J,K))
         Psoil(6)=Ksoil(6)*Ssoil(6)*(Tsoil1(I,J,K+1)-Tsoil1(I,J,K))
         Psoil(0)=Psoil(1)+Psoil(2)+Psoil(3)+Psoil(4)+Psoil(5)+Psoil(6)

         Pint=Psoil(0)

         Tsoil2(I,J,K)=Tsoil1(I,J,K) + Pint*Dt
     &                         /(CvSoil(TypSoil(I,J,K))*VolSoil)

         !integrals :
         !-----------
         PintTot=PintTot+Pint*Nmod/NDt

         RETURN
         END
C*************************************************************************

C*************************************************************************
         SUBROUTINE SetBorders

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'

         !local variables:
         !-------------------
         INTEGER Isurf(6)
         !===============================================================

         IF (I.EQ.1) THEN
            Isurf(1)=TypSoil(0,J,K)
         ELSE
            Isurf(1)=0
         END IF
         IF (I.EQ.NI) THEN
            Isurf(2)=TypSoil(NI+1,J,K)
         ELSE
            Isurf(2)=0
         END IF
         IF (J.EQ.1) THEN
            Isurf(3)=TypSoil(I,0,K)
         ELSE
            Isurf(3)=0
```

```
      END IF
      IF (J.EQ.NJ) THEN
         Isurf(4)=TypSoil(I,NJ+1,K)
      ELSE
         Isurf(4)=0
      END IF
      IF (K.EQ.1) THEN
         Isurf(5)=TypSoil(I,J,0)
      ELSE
         Isurf(5)=0
      END IF
      IF (K.EQ.NK) THEN
         Isurf(6)=TypSoil(I,J,NK+1)
      ELSE
         Isurf(6)=0
      END IF

      DO Idir=1,6
      IF (Isurf(Idir).GT.0) THEN
         !set Xbord:
         IF (TypSurf(Isurf(Idir)).EQ.0) THEN !output power:
          Xbord(Isurf(Idir))=Xbord(Isurf(Idir)) + Nmod*Psoil(Idir)/NDt
         ELSE IF (IDt.EQ.NDt) THEN !output temperature:
          Xbord(Isurf(Idir))=Xbord(Isurf(Idir))
     &      + Tsoil2(I,J,K)*Nmod*Ssoil(Idir)*Ksoil(Idir)
     &        /Kbord(Isurf(Idir))/Sbord(Isurf(Idir))
         END IF
         !set integral:
         PsurfTot=PsurfTot+Psoil(Idir)*Nmod/NDt
      END IF
      END DO

      RETURN
      END
C*************************************************************************

C*************************************************************************
      SUBROUTINE SetPhysicalConstants

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !===========================================================

         CmAir=1000
         CmWat=4180
         MmolAir=0.0289645
         MmolWat=0.0180153
         RhoWat=998
         ClatWat=2501000
         CmVap=1805
         Rgas=8.3144

      RETURN
      END
C*************************************************************************

C*************************************************************************
      SUBROUTINE OpenParameterFiles

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !===========================================================

      INQUIRE (UNIT=IparDef,OPENED=OparDef)
      IF (.NOT.OparDef) THEN
         OPEN (UNIT=IparDef,FILE='paramdef.txt')
      END IF
      INQUIRE (UNIT=IparCon,OPENED=OparCon)
      IF (.NOT.OparCon) THEN
         OPEN (UNIT=IparCon,FILE='paramcon.txt')
      END IF

      RETURN
      END
C*************************************************************************

C*************************************************************************
      SUBROUTINE ReadSuppliedParameters

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !===========================================================

1     FORMAT (A)
100   FORMAT (12I4)
```

```
  200 FORMAT (4E12.4)

      WRITE(IparCon,1)'**********************************************'
      WRITE(IparCon,1)'* TYPE 61 SUPPLIED PARAMETERS                '
      WRITE(IparCon,1)'*=============================================='

      !Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK
      !----------------------------------------------------------------
      WRITE(IparCon,1) '* Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK
      WRITE(IparCon,100) Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK
      WRITE(IparCon,*)

      IF (Nmod.LT.1) CALL StopError(101)
      IF (Nsec.LT.1) CALL StopError(102)
      IF ((Nsoil.LT.1).OR.(Nsoil.GT.NsoilMax)) CALL StopError(103)
      IF ((Nsurf.LT.1).OR.(Nsurf.GT.NsurfMax)) CALL StopError(104)
      IF ((NI.LT.1).OR.(NI.GT.NIMax)) CALL StopError(105)
      IF ((NJ.LT.2).OR.(NJ.GT.NJMax)) CALL StopError(106)
      IF ((NK.LT.3).OR.(NK.GT.NKMax)) CALL StopError(107)

      !DX,DY,DZ
      !----------------------------------------------------------------
      WRITE(IparCon,1) '* DX [m]:'
      DX(0)=0
      DX(NI+1)=0
      CALL SkipComment(IparDef)
      READ(IparDef,*) (DX(I),I=1,NI)
      WRITE(IparCon,200) (DX(I),I=1,NI)
      WRITE(IparCon,*)

      WRITE(IparCon,1) '* DY [m]:'
      DY(0)=0
      DY(NJ+1)=0
      CALL SkipComment(IparDef)
      READ(IparDef,*) (DY(J),J=1,NJ)
      WRITE(IparCon,200) (DY(J),J=1,NJ)
      WRITE(IparCon,*)

      WRITE(IparCon,1) '* DZ [m]:'
      DZ(0)=0
      DZ(NK+1)=0
      CALL SkipComment(IparDef)
      READ(IparDef,*) (DZ(K),K=1,NK)
      WRITE(IparCon,200) (DZ(K),K=1,NK)
      WRITE(IparCon,*)

      DO I=1,NI
         IF (DX(I).LE.0) CALL StopError(201)
      END DO
      DO J=1,NJ
         IF (DY(J).LE.0) CALL StopError(202)
      END DO
      DO K=1,NK
         IF (DZ(K).LE.0) CALL StopError(203)
      END DO

      !TypSec
      !----------------------------------------------------------------
      WRITE(IparCon,1) '* TypSec [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) (TypSec(I),I=1,NI)
      WRITE(IparCon,100) (TypSec(I),I=1,NI)
      WRITE(IparCon,*)

      DO I=1,NI
         IF (TypSec(I).GT.Nsec) CALL StopError(301)
      END DO

      !TypSoil
      !----------------------------------------------------------------
      DO Isec=0,Nsec+1

         !initialise TypSoil:
         DO J=0,NJ+1
         DO K=0,NK+1
            TypSoil(NIMax,J,K)=0
         END DO
         END DO

         !read TypSoil:
         IF ((Isec.EQ.0).OR.(Isec.EQ.Nsec+1)) THEN
            IF (Isec.EQ.0) THEN
               WRITE(IparCon,FMT='(A32)')
     &         '* TypSoil for front surface [-]:'
            ELSE
               WRITE(IparCon,FMT='(A31)')
```

```
     &          '* TypSoil for rear surface [-]:'
              END IF
              DO K=1,NK
                 CALL SkipComment(IparDef)
                 READ(IparDef,*) (TypSoil(NIMax,J,K),J=1,NJ)
                 WRITE(IparCon,FMT='(4X,60I4)')
     &           (TypSoil(NIMax,J,K),J=1,NJ)
              END DO
              WRITE(IparCon,*)
           ELSE
              WRITE(IparCon,FMT='(A18,I2,A5)')
     &        '* TypSoil for sec#',Isec,' [-]:'
              CALL SkipComment(IparDef)
              READ(IparDef,*) (TypSoil(NIMax,J,0),J=1,NJ)
              WRITE(IparCon,FMT='(4X,60I4)') (TypSoil(NIMax,J,0),J=1,NJ)
              DO K=1,NK
                 CALL SkipComment(IparDef)
                 READ(IparDef,*) (TypSoil(NIMax,J,K),J=0,NJ+1)
                 WRITE(IparCon,FMT='(64I4)')
     &           (TypSoil(NIMax,J,K),J=0,NJ+1)
              END DO
              CALL SkipComment(IparDef)
              READ(IparDef,*) (TypSoil(NIMax,J,NK+1),J=1,NJ)
              WRITE(IparCon,FMT='(4X,60I4)')
     &           (TypSoil(NIMax,J,NK+1),J=1,NJ)
              WRITE(IparCon,*)
           END IF

           !test TypSoil:
           DO J=0,NJ+1
           DO K=0,NK+1
              IF ((J.GE.1).AND.(J.LE.NJ).AND.(K.GE.1).AND.(K.LE.NK)) THEN
                 IF (TypSoil(NIMax,J,K).LT.0) CALL StopError(401)
                 IF (TypSoil(NIMax,J,K).GT.Nsoil) CALL StopError(402)
              ELSE
                 IF (TypSoil(NIMax,J,K).LT.0) CALL StopError(403)
                 IF (TypSoil(NIMax,J,K).GT.Nsurf) CALL StopError(404)
              END IF
           END DO
           END DO

           !assign TypSoil:
           IF (Isec.EQ.0) THEN
              DO J=0,NJ+1
              DO K=0,NK+1
                 TypSoil(0,J,K)=TypSoil(NIMax,J,K)
              END DO
              END DO
           ELSE IF (Isec.EQ.Nsec+1) THEN
              DO J=0,NJ+1
              DO K=0,NK+1
                 TypSoil(NI+1,J,K)=TypSoil(NIMax,J,K)
              END DO
              END DO
           ELSE
              DO I=1,NI
              IF (TypSec(I).EQ.Isec) THEN
                 DO J=0,NJ+1
                 DO K=0,NK+1
                    TypSoil(I,J,K)=TypSoil(NIMax,J,K)
                 END DO
                 END DO
              END IF
              END DO
           END IF

        END DO

        !test front,back:
        DO J=1,NJ
        DO K=1,NK
           IF ((TypSoil(0,J,K).NE.0).AND.(TypSoil(1,J,K).EQ.0))
     &     CALL StopError(411)
           IF ((TypSoil(NI+1,J,K).NE.0).AND.(TypSoil(NI,J,K).EQ.0))
     &     CALL StopError(412)
        END DO
        END DO
        !test left,right:
        DO I=1,NI
        DO K=1,NK
           IF (Nmod.EQ.1) THEN
              IF (TypSoil(I,1,K).EQ.0) CALL StopError(421)
              IF (TypSoil(I,NJ,K).EQ.0) CALL StopError(422)
              IF (TypSoil(I,0,K).NE.0) CONTINUE !423
              IF (TypSoil(I,NJ+1,K).NE.0) CONTINUE !424
           ELSE IF (Nmod.EQ.2) THEN
              IF (TypSoil(I,1,K).EQ.0) CALL StopError(431)
              IF (TypSoil(I,NJ,K).EQ.0) CONTINUE !432
```

```
            IF (TypSoil(I,0,K).NE.0) CONTINUE !433
            IF (TypSoil(I,NJ+1,K).NE.0) CALL StopError(434)
         ELSE
            IF (TypSoil(I,1,K).EQ.0) CONTINUE !441
            IF (TypSoil(I,NJ,K).EQ.0) CONTINUE !442
            IF (TypSoil(I,0,K).NE.0) CALL StopError(443)
            IF (TypSoil(I,NJ+1,K).NE.0) CALL StopError(444)
         END IF
      END DO
      END DO
      !test top,bottom:
      DO I=1,NI
      DO J=1,NJ
         IF (TypSoil(I,J,1).EQ.0) CALL StopError(451)
         IF (TypSoil(I,J,NK).EQ.0) CALL StopError(452)
      END DO
      END DO

      !PosInf
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* PosInf [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) (PosInf(Ipos),Ipos=1,6)
      WRITE(IparCon,FMT='(6I4)') (PosInf(Ipos),Ipos=1,6)
      WRITE(IparCon,*)

      IF ((PosInf(1).LT.1).OR.(PosInf(1).GT.NI))
     &   CALL StopError(501)
      IF ((PosInf(2).LT.1).OR.(PosInf(2).GT.NJ))
     &   CALL StopError(502)
      IF ((PosInf(3).LT.1).OR.(PosInf(3).GT.NK))
     &   CALL StopError(503)
      IF ((PosInf(4).LT.PosInf(1)).OR.(PosInf(4).GT.NI))
     &   CALL StopError(504)
      IF ((PosInf(5).LT.PosInf(2)).OR.(PosInf(5).GT.NJ))
     &   CALL StopError(505)
      IF ((PosInf(6).LT.PosInf(3)).OR.(PosInf(6).GT.NK))
     &   CALL StopError(506)

      !Kair0,Kair1
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* Kair0 [kJ/hr K m2]'
     &                 ' ,Kair1 [(kJ/hr K m2)/(m/s)]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) Kair0,Kair1
      WRITE(IparCon,200) Kair0,Kair1
      WRITE(IparCon,*)

      IF ((Kair0.LT.0).OR.(Kair1.LT.0)) CALL StopError(601)
      Kair0=Kair0/3.6 ! [W/K m2] <- [kJ/hr K m2]
      Kair1=Kair1/3.6 ! [(W/K m2)/(m/s)] <- [(kJ/hr K m2)/(m/s)]

      !LamSoil,CvSoil
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* LamSoil [kJ/hr K m], CvSoil [kJ/K m3]:'
      DO Isoil=1,Nsoil
         CALL SkipComment(IparDef)
         READ(IparDef,*) LamSoil(Isoil),CvSoil(Isoil)
         WRITE(IparCon,200) LamSoil(Isoil),CvSoil(Isoil)
      END DO
      WRITE(IparCon,*)

      DO Isoil=1,Nsoil
         IF (LamSoil(Isoil).LE.0) CALL StopError(701)
         IF (CvSoil(Isoil).LE.0) CALL StopError(702)
         LamSoil(Isoil)=LamSoil(Isoil)/3.6 ! [W/K m] <- [kJ/hr K m]
         CvSoil(Isoil)=CvSoil(Isoil)*1000 ! [J/K m3] <- [kJ/K m3]
      END DO

      !LamTub,CvTub
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* LamTub [kJ/hr K m], CvTub [kJ/K m3]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) LamTub,CvTub
      WRITE(IparCon,200) LamTub,CvTub
      WRITE(IparCon,*)

      IF (LamTub.LE.0) CALL StopError(801)
      IF (CvTub.LE.0) CALL StopError(802)
      LamTub=LamTub/3.6 ! [W/K m] <- [kJ/hr K m]
      CvTub=CvTub*1000 ! [J/K m3] <- [kJ/K m3]

      !ThTub,CtubCor,Rfric
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* ThTub [m], CtubCor [-], Rfric [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) ThTub,CtubCor,Rfric
      WRITE(IparCon,200) ThTub,CtubCor,Rfric
```

```
      WRITE(IparCon,*)

      IF (ThTub.LT.0) CALL StopError(901)
      IF (CtubCor.LE.0) CALL StopError(902)
      IF (Rfric.LT.0) CALL StopError(903)

      !TypWat,Vwat
      !---------------------------------------------------------------
      WRITE(IparCon,1) '* TypWat [-], Vwat [m/hr]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) (TypWat(Idir),Idir=-1,1)
      WRITE(IparCon,100)  (TypWat(Idir),Idir=-1,1)
      CALL SkipComment(IparDef)
      READ(IparDef,*) (Vwat(Idir),Idir=-1,1)
      WRITE(IparCon,200)  (Vwat(Idir),Idir=-1,1)
      WRITE(IparCon,*)

      DO Idir=-1,1
         IF ((TypWat(Idir).LT.1).OR.(TypWat(Idir).GT.2))
     &   CALL StopError(1001)
         IF (Vwat(Idir).LT.0) CALL StopError(1002)
         Vwat(Idir)=Vwat(Idir)/3600 ! [m/s] <- [m/hr]
      END DO

      !NiniSoil,NiniWat
      !---------------------------------------------------------------
      WRITE(IparCon,1) '* NiniSoil,NiniWat [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) NiniSoil,NiniWat
      WRITE(IparCon,100) NiniSoil,NiniWat
      WRITE(IparCon,*)

      IF ((NiniSoil.LT.1).OR.(NiniSoil.GT.NiniMax)) CALL StopError(1101)
      IF ((NiniWat.LT.1).OR.(NiniWat.GT.NiniMax)) CALL StopError(1102)

      !TiniSoil,PosIniSoil
      !---------------------------------------------------------------
      WRITE(IparCon,1) '* TiniSoil [degC], PosIniSoil [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) TiniSoil(1)
      WRITE(IparCon,200) TiniSoil(1)
      DO Iini=2,NiniSoil
         CALL SkipComment(IparDef)
         READ(IparDef,*) TiniSoil(Iini),(PosIniSoil(Iini,Ipos),Ipos=1,6)
         WRITE(IparCon,FMT='(E12.4,6I4)')
     &                    TiniSoil(Iini),(PosIniSoil(Iini,Ipos),Ipos=1,6)
      END DO
      WRITE(IparCon,*)

      DO Iini=2,NiniSoil
         IF ((PosIniSoil(Iini,1).LT.1).OR.(PosIniSoil(Iini,1).GT.NI))
     &   CALL StopError(1201)
         IF ((PosIniSoil(Iini,2).LT.1).OR.(PosIniSoil(Iini,2).GT.NJ))
     &   CALL StopError(1202)
         IF ((PosIniSoil(Iini,3).LT.1).OR.(PosIniSoil(Iini,3).GT.NK))
     &   CALL StopError(1203)
         IF ((PosIniSoil(Iini,4).LT.PosIniSoil(Iini,1)).OR.
     &   (PosIniSoil(Iini,4).GT.NI)) CALL StopError(1204)
         IF ((PosIniSoil(Iini,5).LT.PosIniSoil(Iini,2)).OR.
     &   (PosIniSoil(Iini,5).GT.NJ)) CALL StopError(1205)
         IF ((PosIniSoil(Iini,6).LT.PosIniSoil(Iini,3)).OR.
     &   (PosIniSoil(Iini,6).GT.NK)) CALL StopError(1206)
      END DO

      !ThIniWat,PosIniWat
      !---------------------------------------------------------------
      WRITE(IparCon,1) '* ThIniWat [m], PosIniWat [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) ThIniWat(1)
      WRITE(IparCon,200) ThIniWat(1)
      DO Iini=2,NiniWat
         CALL SkipComment(IparDef)
         READ(IparDef,*) ThIniWat(Iini),(PosIniWat(Iini,Ipos),Ipos=1,6)
         WRITE(IparCon,FMT='(E12.4,6I4)')
     &                    ThIniWat(Iini),(PosIniWat(Iini,Ipos),Ipos=1,6)
      END DO
      WRITE(IparCon,*)

      DO Iini=2,NiniWat
         IF ((PosIniWat(Iini,1).LT.1).OR.(PosIniWat(Iini,1).GT.NI))
     &   CALL StopError(1301)
         IF ((PosIniWat(Iini,2).LT.1).OR.(PosIniWat(Iini,2).GT.NJ))
     &   CALL StopError(1302)
         IF ((PosIniWat(Iini,3).LT.1).OR.(PosIniWat(Iini,3).GT.NK))
     &   CALL StopError(1303)
         IF ((PosIniWat(Iini,4).LT.PosIniWat(Iini,1)).OR.
     &   (PosIniWat(Iini,4).GT.NI)) CALL StopError(1304)
         IF ((PosIniWat(Iini,5).LT.PosIniWat(Iini,2)).OR.
```

```
     &   (PosIniWat(Iini,5).GT.NJ)) CALL StopError(1305)
         IF ((PosIniWat(Iini,6).LT.PosIniWat(Iini,3)).OR.
     &   (PosIniWat(Iini,6).GT.NK)) CALL StopError(1306)
      END DO

      !Nopt
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* Nopt [-]:'
      CALL SkipComment(IparDef)
      READ(IparDef,*) Nopt
      WRITE(IparCon,100) Nopt
      WRITE(IparCon,*)

      IF ((Nopt.LT.0).OR.(Nopt.GT.NoptMax)) CALL StopError(1401)

      !TypOpt,PosOpt
      !-----------------------------------------------------------------
      WRITE(IparCon,1) '* TypOpt [-], PosOpt [-]:'
      DO Iopt=1,Nopt
         CALL SkipComment(IparDef)
         READ(IparDef,*) TypOpt(Iopt),(PosOpt(Iopt,Ipos),Ipos=1,6)
         WRITE(IparCon,100) TypOpt(Iopt),(PosOpt(Iopt,Ipos),Ipos=1,6)
      END DO
      IF (Nopt.LT.1) THEN
         WRITE(IparCon,*) '* none'
      END IF

      DO Iopt=2,Nopt
         IF ((PosOpt(Iopt,1).LT.1).OR.(PosOpt(Iopt,1).GT.NI))
     &   CALL StopError(1501)
         IF ((PosOpt(Iopt,2).LT.1).OR.(PosOpt(Iopt,2).GT.NJ))
     &   CALL StopError(1502)
         IF ((PosOpt(Iopt,3).LT.1).OR.(PosOpt(Iopt,3).GT.NK))
     &   CALL StopError(1503)
         IF ((PosOpt(Iopt,4).LT.PosOpt(Iopt,1)).OR.
     &   (PosOpt(Iopt,4).GT.NI)) CALL StopError(1504)
         IF ((PosOpt(Iopt,5).LT.PosOpt(Iopt,2)).OR.
     &   (PosOpt(Iopt,5).GT.NJ)) CALL StopError(1505)
         IF ((PosOpt(Iopt,6).LT.PosOpt(Iopt,3)).OR.
     &   (PosOpt(Iopt,6).GT.NK)) CALL StopError(1506)
      END DO

      WRITE(IparCon,1)'************************************************'
      WRITE(IparCon,*)

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE SetDerivedParameters

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'

      !local variables:
      !-------------------
      REAL DtIJK
      !=================================================================

1     FORMAT (A)

      WRITE(IparCon,1)'************************************************'
      WRITE(IparCon,1)'* TYPE 61 DERIVED PARAMETERS                      '
      WRITE(IparCon,1)'*==============================================='

      !Ntub,IflowIni,IflowEnd
      !-----------------------------------------------------------------
      Ntub=0
      IflowIni=NI+1
      IflowEnd=0
      DO I=1,NI

         Ncount=0
         DO J=1,NJ
         DO K=1,NK
            IF (TypSoil(I,J,K).EQ.0) THEN
               Ncount=Ncount+1
            END IF
         END DO
         END DO
         IF (Ncount.GT.Ntub) THEN
            Ntub=Ncount
         END IF

         IF (Ncount.GT.0) THEN
            IF (I.LT.IflowIni) THEN
```

```fortran
               IflowIni=I
            END IF
            IF (I.GT.IflowEnd) THEN
               IflowEnd=I
            END IF
         END IF

      END DO
      WRITE(IparCon,FMT='(A11,14X,I4)') '* Ntub [-]:',Ntub
      WRITE(IparCon,FMT='(A15,10X,I4)') '* IflowIni [-]:',IflowIni
      WRITE(IparCon,FMT='(A15,10X,I4)') '* IflowEnd [-]:',IflowEnd

      IF (Ntub.GT.NtubMax) CALL StopError(2101)

      !PosTub
      !------------------------------------------------------------------
      !test tube compatibility between sections:
      DO I=IflowIni+1,IflowEnd
      DO J=1,NJ
      DO K=1,NK
         IF(
     &   ((TypSoil(I,J,K).EQ.0).AND.(TypSoil(IflowIni,J,K).NE.0))
     &   .OR.
     &   ((TypSoil(I,J,K).NE.0).AND.(TypSoil(IflowIni,J,K).EQ.0))
     &   ) CALL StopError(2201)
      END DO
      END DO
      END DO

      !set PosTub:
      Itub=0
      DO J=1,NJ
      DO K=1,NK
         IF (TypSoil(IflowIni,J,K).EQ.0) THEN
            Itub=Itub+1
            PosTub(Itub,1)=J
            PosTub(Itub,2)=K
         END IF
      END DO
      END DO
      DO Itub=1,Ntub
         WRITE(IparCon,FMT='(A10,I2,A5,8X,2I4)')
     &   '* PosTub #',Itub,' [-]:',PosTub(Itub,1),PosTub(Itub,2)
      END DO

      !LX,LY,LZ,Ltub
      !------------------------------------------------------------------
      LX=0
      DO I=1,NI
         LX=LX+DX(I)
      END DO
      WRITE(IparCon,FMT='(A9,16X,E12.4)') '* LX [m]:',LX

      LY=0
      DO J=1,NJ
         LY=LY+DY(J)*Nmod
      END DO
      WRITE(IparCon,FMT='(A9,16X,E12.4)') '* LY [m]:',LY

      LZ=0
      DO K=1,NK
         LZ=LZ+DZ(K)
      END DO
      WRITE(IparCon,FMT='(A9,16X,E12.4)') '* LZ [m]:',LZ

      Ltub=0
      DO I=IflowIni,IflowEnd
         Ltub=Ltub+DX(I)
      END DO
      WRITE(IparCon,FMT='(A11,14X,E12.4)') '* Ltub [m]:',Ltub

      !SairTot,StubTot,ZairTot
      !------------------------------------------------------------------
      SairTot=0
      StubTot=0
      DO Itub=1,Ntub
         Iflow=IflowIni
         CALL TubeProperties
         SairTot=SairTot + Sair*Nmod
         StubTot=StubTot + Stub*Nmod*Ltub/DX(I)
         ZairTot=ZairTot + Zair*Nmod
      END DO
      WRITE(IparCon,FMT='(A15,10X,E12.4)') '* SairTot [m2]:',SairTot
      WRITE(IparCon,FMT='(A15,10X,E12.4)') '* StubTot [m2]:',StubTot
      WRITE(IparCon,FMT='(A17,8X,E12.4)') '* ZairTot [m5/2]:',ZairTot

      !SinfTot
      !------------------------------------------------------------------
```

```
      SinfTot=0
      DO Itub=1,Ntub
      DO I=IflowIni,IflowEnd
         CALL TubeProperties
         IF ((I.GE.PosInf(1)).AND.(I.LE.PosInf(4))) THEN
         IF ((J.GE.PosInf(2)).AND.(J.LE.PosInf(5))) THEN
         IF ((K.GE.PosInf(3)).AND.(K.LE.PosInf(6))) THEN
            SinfTot=SinfTot + Stub*Nmod
         END IF
         END IF
         END IF
      END DO
      END DO
      WRITE(IparCon,FMT='(A15,10X,E12.4)') '* SinfTot [m2]:',SinfTot

      !Sbord
      !-----------------------------------------------------------------
      !set Sbord:
      Sbord=0

      DO I=1,NI
      DO J=1,NJ
         Isurf=TypSoil(I,J,0)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DX(I)*DY(J)*Nmod
         Isurf=TypSoil(I,J,NK+1)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DX(I)*DY(J)*Nmod
      END DO
      END DO

      DO I=1,NI
      DO K=1,NK
         Isurf=TypSoil(I,0,K)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DX(I)*DZ(K)*Nmod
         Isurf=TypSoil(I,NJ+1,K)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DX(I)*DZ(K)*Nmod
      END DO
      END DO

      DO J=1,NJ
      DO K=1,NK
         Isurf=TypSoil(0,J,K)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DY(J)*DZ(K)*Nmod
         Isurf=TypSoil(NI+1,J,K)
         IF (Isurf.GT.0) Sbord(Isurf)=Sbord(Isurf)+DY(J)*DZ(K)*Nmod
      END DO
      END DO

      DO Isurf=1,Nsurf
         WRITE(IparCon,FMT='(A9,I2,A6,8X,E12.4)')
     &   '* Sbord #',Isurf,' [m2]:',Sbord(Isurf)
      END DO

      !Kbord
      !-----------------------------------------------------------------
      !set Kbord:
      Kbord=0

      DO I=1,NI
      DO J=1,NJ
         Isurf=TypSoil(I,J,0)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DX(I)*DY(J)*Nmod
     &                            *2*LamSoil(TypSoil(I,J,1))/DZ(1)
         Isurf=TypSoil(I,J,NK+1)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DX(I)*DY(J)*Nmod
     &                            *2*LamSoil(TypSoil(I,J,NK))/DZ(NK)
      END DO
      END DO

      DO I=1,NI
      DO K=1,NK
         Isurf=TypSoil(I,0,K)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DX(I)*DZ(K)*Nmod
     &                            *2*LamSoil(TypSoil(I,1,K))/DY(1)
         Isurf=TypSoil(I,NJ+1,K)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DX(I)*DZ(K)*Nmod
     &                           *2*LamSoil(TypSoil(I,NJ,K))/DY(NJ)
      END DO
      END DO

      DO J=1,NJ
      DO K=1,NK
         Isurf=TypSoil(0,J,K)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DY(J)*DZ(K)*Nmod
     &                            *2*LamSoil(TypSoil(1,J,K))/DX(1)
         Isurf=TypSoil(NI+1,J,K)
         IF (Isurf.GT.0) Kbord(Isurf)=Kbord(Isurf)+DY(J)*DZ(K)*Nmod
     &                            *2*LamSoil(TypSoil(NI,J,K))/DX(NI)
      END DO
```

```
      END DO

      DO Isurf=1,Nsurf
      IF (Sbord(Isurf).EQ.0) THEN
         Kbord(Isurf)=0
      ELSE
         Kbord(Isurf)=Kbord(Isurf)/Sbord(Isurf)
      END IF
      END DO

      DO Isurf=1,Nsurf
         WRITE(IparCon,FMT='(A9,I2,A12,2X,E12.4)')
     &   '* Kbord #',Isurf,' [kJ/hr K m2]:',Kbord(Isurf)*3.6 ! [kJ/hr K m2] <- [W/K m2]
      END DO

      !DtSoil,DtWat
      !-----------------------------------------------------------------
      !set DtSoil:
      Ncount=0
      DtSoil=0
      DO I=1,NI
      DO J=1,NJ
      DO K=1,NK
      IF (TypSoil(I,J,K).NE.0) THEN
         Ncount=Ncount+1
         CALL SoilProperties
         DtIJK=CvSoil(TypSoil(I,J,K))*VolSoil/2
     &   /(Ksoil(1)*Ssoil(1)+Ksoil(2)*Ssoil(2)+Ksoil(3)*Ssoil(3)
     &   +Ksoil(4)*Ssoil(4)+Ksoil(5)*Ssoil(5)+Ksoil(6)*Ssoil(6))
         IF (Ncount.EQ.1) THEN
            DtSoil=DtIJK
         ELSE
            DtSoil=MIN(DtSoil,DtIJK)
         END IF
      END IF
      END DO
      END DO
      END DO
      WRITE(IparCon,FMT='(A14,11X,E12.4)') '* DtSoil [hr]:',3600*DtSoil ! [hr] <- [s]

      !set DtWat:
      Ncount=0
      DtWat=0
      DO Idir=-1,1
      IF ((TypWat(Idir).EQ.1).AND.(Vwat(Idir).GT.0)) THEN
      DO I=IflowIni,IflowEnd
         Ncount=Ncount+1
         DtIJK=DX(I)/Vwat(Idir)
         IF (Ncount.EQ.1) THEN
            DtWat=DtIJK
         ELSE
            DtWat=MIN(DtWat,DtIJK)
         END IF
      END DO
      END IF
      END DO
      IF (DtWat.GT.0) THEN
         WRITE(IparCon,FMT='(A13,12X,E12.4)') '* DtWat [hr]:',3600*DtWat ! [hr] <- [s]
      ELSE
         WRITE(IparCon,FMT='(A13,14X,A4)') '* DtWat [hr]:','none'
      END IF

      !FairMinTub
      !-----------------------------------------------------------------
      !set FairMinTub:
      Ncount=0
      FairMinTub=0
      DO Itub=1,Ntub
      DO Iflow=IflowIni,IflowEnd
         Ncount=Ncount+1
         CALL TubeProperties
         FairMinIJK=2*SairTot*Stub*Kair0/(Sair*1290-2*Stub*Kair1)
         !(air capacity at normal condition: 1290 J/K/m3)
         IF (Ncount.EQ.1) THEN
            FairMinTub=FairMinIJK
         ELSE
            FairMinTub=MIN(FairMinIJK,FairMinTub)
         END IF
      END DO
      END DO
      WRITE(IparCon,FMT='(A21,4X,E12.4)')
     &'* FairMinTub [m3/hr]:',3600*FairMinTub ! [m3/hr] <- [m3/s]

      WRITE(IparCon,1)'**************************************************'
      WRITE(IparCon,*)

      RETURN
      END
```

```
C**********************************************************************

C**********************************************************************
      SUBROUTINE SetSimulationParameters

      !common variables:
      !-------------------
      REAL TIME0,TIMEF,DELT,IWARN
      COMMON /SIM/ TIME0,TIMEF,DELT,IWARN
      INCLUDE 'type61.inc'
      !================================================================

1      FORMAT (A)

      WRITE(IparCon,1)'**********************************************'
      WRITE(IparCon,1)'* TYPE 61 SIMULATION PARAMETERS               '
      WRITE(IparCon,1)'*=============================================='

      !simulation timestep:
      IF (Dt.EQ.0) THEN
         IF (DtWat.EQ.0) THEN
            NDt=MAX(IFIX(3600*DELT/DtSoil),1)
         ELSE
            NDt=MAX(IFIX(3600*DELT/MIN(DtSoil,DtWat)),1)
         END IF
      ELSE
         NDt=MAX(IFIX(3600*DELT/Dt),1)
      END IF
      Dt=3600*DELT/NDt
      WRITE(IparCon,FMT='(A10,15X,E12.4)') '* Dt [hr]:',Dt/3600 ! [hr] <- [s]

      !minimal airflow:
      IF (FairMin.EQ.0) THEN
         FairMin=FairMinTub
      END IF
      WRITE(IparCon,FMT='(A18,7X,E12.4)') '* FairMin [m3/hr]:',
     &                                     3600*FairMin ! [m3/hr] <- [m3/s]

      WRITE(IparCon,1)'**********************************************'
      WRITE(*,*)

      RETURN
      END
C**********************************************************************

C**********************************************************************
      SUBROUTINE CloseParameterFiles

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !================================================================

      IF (.NOT.OparDef) THEN
         CLOSE (IparDef)
      END IF
      IF (.NOT.OparCon) THEN
         CLOSE (IparCon)
      END IF

      RETURN
      END
C**********************************************************************

C**********************************************************************
      SUBROUTINE InitialiseSoil

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'
      !================================================================

      DO I=1,NI
      DO J=1,NJ
      DO K=1,NK
         Tsoil0(I,J,K)=TiniSoil(1)
         Tsoil1(I,J,K)=TiniSoil(1)
         Tsoil2(I,J,K)=TiniSoil(1)
      END DO
      END DO
      END DO

      DO Iini=2,NiniSoil
      DO I=PosIniSoil(Iini,1),PosIniSoil(Iini,4)
      DO J=PosIniSoil(Iini,2),PosIniSoil(Iini,5)
      DO K=PosIniSoil(Iini,3),PosIniSoil(Iini,6)
         Tsoil0(I,J,K)=TiniSoil(Iini)
         Tsoil1(I,J,K)=TiniSoil(Iini)
```

```
            Tsoil2(I,J,K)=TiniSoil(Iini)
         END DO
         END DO
         END DO
         END DO

         RETURN
         END
C***********************************************************************

C***********************************************************************
         SUBROUTINE InitialiseWater

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'
         !=================================================================

         DO Itub=1,Ntub
         DO Iflow=IflowIni,IflowEnd
            CALL TubeProperties
            Mwat0(Iflow,Itub)=Stub*ThIniWat(1)*RhoWat
            Mwat1(Iflow,Itub)=Stub*ThIniWat(1)*RhoWat
            Mwat2(Iflow,Itub)=Stub*ThIniWat(1)*RhoWat
         END DO
         END DO

         DO Iini=2,NiniWat
         DO Itub=1,Ntub
         DO Iflow=IflowIni,IflowEnd
            CALL TubeProperties
            IF ((I.GE.PosIniWat(Iini,1)).AND.(I.LE.PosIniWat(Iini,4))) THEN
            IF ((J.GE.PosIniWat(Iini,2)).AND.(J.LE.PosIniWat(Iini,5))) THEN
            IF ((K.GE.PosIniWat(Iini,3)).AND.(K.LE.PosIniWat(Iini,6))) THEN
               Mwat0(I,Itub)=Stub*ThIniWat(Iini)*RhoWat
               Mwat1(I,Itub)=Stub*ThIniWat(Iini)*RhoWat
               Mwat2(I,Itub)=Stub*ThIniWat(Iini)*RhoWat
            END IF
            END IF
            END IF
         END DO
         END DO
         END DO

         RETURN
         END
C***********************************************************************

C***********************************************************************
         SUBROUTINE OptionalsTube

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'

         !local variables:
         !-------------------
         INTEGER Ipos,Jpos,Kpos,NnodeOpt(NoptMax)
         !=================================================================

         !Initialise NnodeOpt:
         !--------------------
         IF (Isim.LT.3) THEN
         IF (IDt.EQ.1) THEN
         IF ((Itub.EQ.1).AND.(Iflow.EQ.IflowIni)) THEN
         DO Iopt=1,Nopt
            IF (INT(TypOpt(Iopt)/100).EQ.0) THEN
               NnodeOpt(Iopt)=0
               DO Ipos=PosOpt(Iopt,1),PosOpt(Iopt,4)
               DO Jpos=PosOpt(Iopt,2),PosOpt(Iopt,5)
               DO Kpos=PosOpt(Iopt,3),PosOpt(Iopt,6)
               IF (TypSoil(Ipos,Jpos,Kpos).EQ.0) THEN
                  NnodeOpt(Iopt)=NnodeOpt(Iopt)+1
               END IF
               END DO
               END DO
               END DO
            END IF
         END DO
         END IF
         END IF
         END IF

         !Update Opt:
         !--------------
         DO Iopt=1,Nopt
         IF (INT(TypOpt(Iopt)/100).EQ.0) THEN
            IF ((I.GE.PosOpt(Iopt,1)).AND.(I.LE.PosOpt(Iopt,4))) THEN
```

```
            IF ((J.GE.PosOpt(Iopt,2)).AND.(J.LE.PosOpt(Iopt,5))) THEN
            IF ((K.GE.PosOpt(Iopt,3)).AND.(K.LE.PosOpt(Iopt,6))) THEN

              GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
     &        20,21,22,23,24),TypOpt(Iopt)

              !Tair [degC]
 1            Opt(Iopt)=Opt(Iopt)+Tair/NnodeOpt(Iopt)/NDt
              GOTO 999

              !Hrel [pcent]
 2            Opt(Iopt)=Opt(Iopt)+Hrel/NnodeOpt(Iopt)/NDt
              GOTO 999

              !Habs [bar]
 3            Opt(Iopt)=Opt(Iopt)+Habs/1E5/NnodeOpt(Iopt)/NDt
              GOTO 999

              !Hrat [kg vap/kg air]
 4            Opt(Iopt)=Opt(Iopt)+Hrat/NnodeOpt(Iopt)/NDt
              GOTO 999

              !Mwat [m3]
 5            IF (IDt.EQ.NDt) THEN
                 Opt(Iopt)=Opt(Iopt)+Mwat2(I,Itub)/RhoWat*Nmod
              END IF
              GOTO 999

              !MwatLat [m3/hr]
 6            Opt(Iopt)=Opt(Iopt)+MwatLat/RhoWat/Dt*3600*Nmod/NDt
              GOTO 999

              !MwatIn [m3/hr]
 7            Opt(Iopt)=Opt(Iopt)+MwatIn/RhoWat/Dt*3600*Nmod/NDt
              GOTO 999

              !MwatInf [m3/hr]
 8            Opt(Iopt)=Opt(Iopt)+MwatInf/RhoWat/Dt*3600*Nmod/NDt
              GOTO 999

              !MwatOut [m3/hr]
 9            Opt(Iopt)=Opt(Iopt)+MwatOut/RhoWat/Dt*3600*Nmod/NDt
              GOTO 999

              !Tsoil [degC]
 10            Opt(Iopt)=Opt(Iopt)+Tsoil2(I,J,K)/NnodeOpt(Iopt)/NDt
              GOTO 999

              !Psbl [kJ/hr]
 11           Opt(Iopt)=Opt(Iopt)+Psbl*3.6*Nmod/NDt
              GOTO 999

              !Plat [kJ/hr]
 12           Opt(Iopt)=Opt(Iopt)+Plat*3.6*Nmod/NDt
              GOTO 999

              !Pwat [kJ/hr]
 13            Opt(Iopt)=Opt(Iopt)+Pwat*3.6*Nmod/NDt
              GOTO 999

              !Pfric [kJ/hr]
 14           Opt(Iopt)=Opt(Iopt)+Pfric*3.6*Nmod/NDt
              GOTO 999

              !Psoil(0) [kJ/hr]
 15           Opt(Iopt)=Opt(Iopt)+Psoil(0)*3.6*Nmod/NDt
              GOTO 999

              !Psoil(1) [kJ/hr]
 16           Opt(Iopt)=Opt(Iopt)+Psoil(1)*3.6*Nmod/NDt
              GOTO 999

              !Psoil(2) [kJ/hr]
 17           Opt(Iopt)=Opt(Iopt)+Psoil(2)*3.6*Nmod/NDt
              GOTO 999

              !Psoil(3) [kJ/hr]
 18           Opt(Iopt)=Opt(Iopt)+Psoil(3)*3.6*Nmod/NDt
              GOTO 999

              !Psoil(4) [kJ/hr]
 19           Opt(Iopt)=Opt(Iopt)+Psoil(4)*3.6*Nmod/NDt
              GOTO 999

              !Psoil(5) [kJ/hr]
 20           Opt(Iopt)=Opt(Iopt)+Psoil(5)*3.6*Nmod/NDt
              GOTO 999
```

```
                 !Psoil(6) [kJ/hr]
 21              Opt(Iopt)=Opt(Iopt)+Psoil(6)*3.6*Nmod/NDt
                 GOTO 999

                 !Pint [kJ/hr]
 22              Opt(Iopt)=Opt(Iopt)+Pint*3.6*Nmod/NDt
                 GOTO 999

                 !Fair [m3/hr]
 23              IF (Iflow.EQ.IflowIni)
     &              Opt(Iopt)=Opt(Iopt)+Fair*3600/NnodeOpt(Iopt)/NDt
                 GOTO 999

                 !Vair [m/s]
 24              IF (Iflow.EQ.IflowIni)
     &              Opt(Iopt)=Opt(Iopt)+Vair/NnodeOpt(Iopt)/NDt
                 GOTO 999

 999             CONTINUE

              END IF
              END IF
              END IF
          END IF
          END DO

          RETURN
          END
C**********************************************************************

C**********************************************************************
          SUBROUTINE OptionalsSoil

          !common variables:
          !--------------------
          INCLUDE 'type61.inc'

          !local variables:
          !--------------------
          INTEGER Ipos,Jpos,Kpos,NnodeOpt(NoptMax)
          !==============================================================

          !Initialise NnodeOpt:
          !---------------------
          IF (Isim.LT.3) THEN
          IF (IDt.EQ.1) THEN
          IF ((I.EQ.1).AND.(J.EQ.1).AND.(K.EQ.1)) THEN
          DO Iopt=1,Nopt
             IF (INT(TypOpt(Iopt)/100).EQ.1) THEN
                NnodeOpt(Iopt)=0
                DO Ipos=PosOpt(Iopt,1),PosOpt(Iopt,4)
                DO Jpos=PosOpt(Iopt,2),PosOpt(Iopt,5)
                DO Kpos=PosOpt(Iopt,3),PosOpt(Iopt,6)
                IF (TypSoil(Ipos,Jpos,Kpos).NE.0) THEN
                   NnodeOpt(Iopt)=NnodeOpt(Iopt)+1
                END IF
                END DO
                END DO
                END DO
             END IF
          END DO
          END IF
          END IF
          END IF

          !Update Opt:
          !--------------
          DO Iopt=1,Nopt
          IF (INT(TypOpt(Iopt)/100).EQ.1) THEN
             IF ((I.GE.PosOpt(Iopt,1)).AND.(I.LE.PosOpt(Iopt,4))) THEN
             IF ((J.GE.PosOpt(Iopt,2)).AND.(J.LE.PosOpt(Iopt,5))) THEN
             IF ((K.GE.PosOpt(Iopt,3)).AND.(K.LE.PosOpt(Iopt,6))) THEN

                GOTO (101,102,103,104,105,106,107,108,109),
     &          (TypOpt(Iopt)-100)

                !Tsoil [degC]
 101            Opt(Iopt)=Opt(Iopt)+Tsoil2(I,J,K)/NnodeOpt(Iopt)/NDt
                GOTO 999

                !Psoil(0) [kJ/hr]
 102            Opt(Iopt)=Opt(Iopt)+Psoil(0)*3.6*Nmod/NDt
                GOTO 999

                !Psoil(1) [kJ/hr]
 103            Opt(Iopt)=Opt(Iopt)+Psoil(1)*3.6*Nmod/NDt
                GOTO 999
```

```
               !Psoil(2) [kJ/hr]
104            Opt(Iopt)=Opt(Iopt)+Psoil(2)*3.6*Nmod/NDt
               GOTO 999

               !Psoil(3) [kJ/hr]
105            Opt(Iopt)=Opt(Iopt)+Psoil(3)*3.6*Nmod/NDt
               GOTO 999

               !Psoil(4) [kJ/hr]
106            Opt(Iopt)=Opt(Iopt)+Psoil(4)*3.6*Nmod/NDt
               GOTO 999

               !Psoil(5) [kJ/hr]
107            Opt(Iopt)=Opt(Iopt)+Psoil(5)*3.6*Nmod/NDt
               GOTO 999

               !Psoil(6) [kJ/hr]
108            Opt(Iopt)=Opt(Iopt)+Psoil(6)*3.6*Nmod/NDt
               GOTO 999

               !Pint [kJ/hr]
109            Opt(Iopt)=Opt(Iopt)+Pint*3.6*Nmod/NDt
               GOTO 999

999            CONTINUE

            END IF
            END IF
            END IF
         END IF
         END DO

         RETURN
         END
C**********************************************************************

C***********************************************************************
         SUBROUTINE OptionalsMiscellaneous

         !common variables:
         !-------------------
         INCLUDE 'type61.inc'
         !==================================================================

         !Update Opt:
         !--------------
         DO Iopt=1,Nopt
         IF (INT(TypOpt(Iopt)/100).EQ.2) THEN

            GOTO (201,202,203,204),(TypOpt(Iopt)-200)

            !PsurfTot [kJ/hr]
201         IF (IDT.EQ.NDT) Opt(Iopt)=PsurfTot*3.6
            GOTO 999

            !PwatTot [kJ/hr]
202         IF (IDT.EQ.NDT) Opt(Iopt)=PwatTot*3.6
            GOTO 999

            !PfricTot [kJ/hr]
203         IF (IDT.EQ.NDT) Opt(Iopt)=PfricTot*3.6
            GOTO 999

            !PintTot [kJ/hr]
204         IF (IDT.EQ.NDT) Opt(Iopt)=PintTot*3.6
999         CONTINUE

         END IF
         END DO

         RETURN
         END
C**********************************************************************

C***********************************************************************
         FUNCTION Hsat$(T)
         !saturating pressure [Pa]
         !(Ashrare Handbook 1989, Ch.6, for -100<T<200 [degC])

         !local variables:
         !-------------------
         REAL Hsat$,T,Tabs
         DATA Cn1,Cn2,Cn3,Cn4,Cn5,Cn6,Cn7 /
     &      -5.6745359  E+3,
     &       6.3925247,
     &      -9.677843   E-3,
     &       6.22115701 E-9,
     &       2.0747825  E-9,
```

```
     &      -9.484024   E-13,
     &       4.163501/
      DATA Cp1,Cp2,Cp3,Cp4,Cp5,Cp7 /
     &      -5.8002206  E+3,
     &       1.3914993,
     &      -4.8640239  E-2,
     &       4.1764768  E-5,
     &      -1.4452093  E-8,
     &       6.5459673/
      !=================================================================
      Tabs=MIN(MAX(T,-100.),200.)+273.15
      IF (T.LT.0) THEN
         Hsat$=EXP(Cn1/Tabs+Cn2+Cn3*Tabs+Cn4*Tabs**2+Cn5*Tabs**3
     &             +Cn6*Tabs**4+Cn7*LOG(Tabs))
      ELSE
         Hsat$=EXP(Cp1/Tabs+Cp2+Cp3*Tabs+Cp4*Tabs**2+Cp5*Tabs**3
     &                               +Cp7*LOG(Tabs))
      END IF
      END
C***********************************************************************

C***********************************************************************
      FUNCTION Hrat$(PrWat,PrAir)
      !humidity ratio [kg vapor/kg air]
      !(perfect gas equation)

      !common variables:
      !-------------------
      REAL CmAir,CmWat,MmolAir,MmolWat,RhoWat,ClatWat,CmVap,Rgas
      COMMON/Type61PhysConst/
     &CmAir,CmWat,MmolAir,MmolWat,RhoWat,ClatWat,CmVap,Rgas

      !local variables:
      !-------------------
      REAL Hrat$,PrWat,PrAir
      !=================================================================
      Hrat$=PrWat*MmolWat/(PrAir*MmolAir)
      END
C***********************************************************************

C***********************************************************************
      FUNCTION Habs$(Hrat,PrAir)
      !absolute humidity of moist air [Pa]
      !(perfect gas equation)

      !common variables:
      !-------------------
      REAL CmAir,CmWat,MmolAir,MmolWat,RhoWat,ClatWat,CmVap,Rgas
      COMMON/Type61PhysConst/
     &CmAir,CmWat,MmolAir,MmolWat,RhoWat,ClatWat,CmVap,Rgas

      !local variables:
      !-------------------
      REAL Habs$,Hrat,PrAir
      !=================================================================
      Habs$=Hrat*PrAir*MmolAir/MmolWat
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE FindZero(Y,X,AZ,BZ,E2,K)

      !original name: NB02A (from NAG library)
      !-----------------------------------------------------------------
      !determination of X +/- E2 so that Y(X)=0:
      !iterative calls to this routine allowsw to determinate X so that
      !
      !     AZ < X-E2 < X < X+E2 < BZ
      !
      !where
      !
      !     sgn(Y(AZ))=sgn(Y(X-E2)) <> sgn(Y(X+E2))=sgn(Y(BZ))

      !local variables:
      !-------------------
      REAL Y ! value of Y(X) (computed outside of the routine): input
      REAL X ! value of X: output
      REAL AZ ! inferior value of interval: input
      REAL BZ ! superior value of interval: input
      REAL E2 ! precision: input
      INTEGER K ! state of determination:
                ! 0: initialisation
                !  : input
                ! 1: iteration not completed
                !  : output
                ! 2: iteration completed
                !  : output
                ! 3: inappropriate interval: sgn(Y(AZ))=sgn(Y(BZ))
```

```
                    !  : output
      !=============================================================

      IF(K)125,10,50
C
C     CALCULATE Y(X) AT X=AZ.
   10 A = AZ
      B = BZ
      X = A
      J1 = 1
      IT = 1
      K = 1
      GO TO 20
C
C     BRANCH DEPENDING ON THE VALUE OF J1.
   50 GO TO (60,120,170,170,170),J1
C
C     CALCULATE Y(X) AT X=BZ.
   60 YA = Y
      X = B
      J1 = 2
      IT=2
      GO TO 20
C
C     ERROR RETURN BECAUSE NO BRACKET
   70 K=3
      GO TO 20
C
C     GET Y FOR X = X2
   90 IT = IT+1
      X = X2
      GO TO 20
C
C     SET THE FIRST BRACKET
  120 B = X
      YB = Y
C     MAKE TEST AT 200 FAIL IF YA OR YB ZERO
  125 Y=AMAX1(ABS(YA),ABS(YB))*2.0
      J1 = 3
      IF(YA*YB)126,200,70
  126 IF(K.GT.0)GO TO 100
      K=1
      IT=0
C
C     TEST WHETHER BRACKET IS SMALL ENOUGH
  100 IF(B-A.LE.E2)GO TO 200
      GO TO (10,10,130,150,160),J1
C
C     CALCULATE THE NEXT X BY THE SECANT METHOD BASED ON THE BRACKET.
  130 IF(ABS(YA) .LE.ABS(YB))GO TO 140
      X1 = A
      Y1 = YA
      X = B
      Y = YB
      GO TO 150
  140 X1 = B
      Y1 = YB
      X = A
      Y = YA
C
C     USE THE SECANT METHOD BASED ON THE FUNCTION VALUES Y1 AND Y.
  150 U=Y*(X-X1)/(Y-Y1)
  155 X2=X-U
      IF(X2.EQ.X)GO TO 110
      X1 = X
      Y1 = Y
      YTEST =.50*AMIN1(ABS(YA),ABS(YB))
C
C     CHECK THAT X2 IS INSIDE THE INTERVAL (A,B).
      IF((X2-A)*(X2-B) .LT. 0.0)GO TO 90
C
C     CALCULATE THE NEXT VALUE OF X BY BISECTION.
  160 X2 = 0.50*(A+B)
      YTEST = 0.0
C
C     CHECK WHETHER THE MAXIMUM ACCURACY HAS BEEN ACHIEVED.
      IF((X2-A)*(X2-B))90,200,200
C
C     MOVE AWAY FROM FIXED POINT TO GET CLOSE BRACKET
  110 IF(U.EQ.0.0)GO TO 160
      U=U+U
      GO TO 155
C
C     REVISE THE BRACKET (A,B).
  170 IF(Y.EQ.0.0)GO TO 195
      IF(YA*Y.GT.0.0)GO TO 180
      B = X
      YB = Y
```

```
      GO TO 190
  180 A = X
      YA = Y
C
C     USE YTEST TO DECIDE THE METHOD FOR THE NEXT VALUE OF X.
  190 J1=4
      IF(ABS(Y).GT.YTEST)J1=5
      IF(YTEST .LE.0.0)J1=3
      GO TO 100
C
C     Y = 0 - SET CLOSEST BRACKET
  195 IF(ABS(X-A).LT.ABS(X-B))GO TO 196
      A=X
      YA=Y
      GO TO 220
  196 B=X
      YB=Y
      GO TO 220
  200 IF(ABS(Y).LE.ABS(YB))GO TO 210
      X=B
      Y=YB
  210 IF(ABS(Y).LE.ABS(YA))GO TO 220
      X=A
      Y=YA
  220 K=2

   20 RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE SkipComment(FileUnit)

      !local variables:
      !-------------------
      INTEGER FileUnit,ICar
      CHARACTER*256 String
      LOGICAL IsData
      !=================================================================

100   FORMAT(A256)

      IsData=.FALSE.
      DO WHILE (.NOT.IsData)
         READ(FileUnit,100) String
         IF (String(1:1).NE.'*') THEN
            Icar=1
            DO WHILE ((.NOT.IsData).AND.(Icar.LE.256))
               IF (String(Icar:Icar).NE.' ') THEN
                  IsData=.TRUE.
               END IF
               Icar=Icar+1
            END DO
         END IF
      END DO
      BACKSPACE(FileUnit)

      RETURN
      END
C***********************************************************************

C***********************************************************************
      SUBROUTINE StopError(Ierr)

      !common variables:
      !-------------------
      INCLUDE 'type61.inc'

      !local variables:
      !-------------------
      INTEGER IerrList(65),Igoto
      DATA (IerrList(Igoto),Igoto=1,65)
     &     / 101, 102, 103, 104, 105, 106, 107, 201, 202, 203, 301, 401,
     &       402, 403, 404, 411, 412, 421, 422, 431, 434, 443, 444, 451,
     &       452, 501, 502, 503, 504, 505, 506, 601, 701, 702, 801, 802,
     &       901, 902,903, 1001,1002,1101,1102,1201,1202,1203,1204,1205,
     &      1206,1301,1302,1303,1304,1305,1306,1401,1501,1502,1503,1504,
     &      1505,1506,2101,2201,9999/
      !=================================================================

1     FORMAT(A,6I4)

      WRITE(IparCon,1)'*!!!!!!!!!!!!!! (not completed) !!!!!!!!!!!!!!!!'
      WRITE(IparCon,1)
      WRITE(IparCon,1)'* TYPE 61 ERROR:',Ierr
      WRITE(IparCon,1)

      Igoto=1
```

```
      DO WHILE ((Ierr.NE.IerrList(Igoto)).AND.(Igoto.LT.65))
         Igoto=Igoto+1
      END DO
      GOTO ( 101, 102, 103, 104, 105, 106, 107, 201, 202, 203, 301, 401,
     &        402, 403, 404, 411, 412, 421, 422, 431, 434, 443, 444, 451,
     &        452, 501, 502, 503, 504, 505, 506, 601, 701, 702, 801, 802,
     &        901, 902,903, 1001,1002,1101,1102,1201,1202,1203,1204,1205,
     &       1206,1301,1302,1303,1304,1305,1306,1401,1501,1502,1503,1504,
     &       1505,1506,2101,2201,9999)
     &       Igoto

      !Nmod,Nsec,Nsoil,Nsurf,NI,NJ,NK
      !-----------------------------------------------------------------
101   WRITE(IparCon,1) '* Nmod<1'
      GOTO 9999
102   WRITE(IparCon,1) '* Nsec<1'
      GOTO 9999
103   WRITE(IparCon,1) '* Nsoil<1 or Nsoil>NsoilMax'
      GOTO 9999
104   WRITE(IparCon,1) '* Nsurf<1 or Nsurf>NsurfMax'
      GOTO 9999
105   WRITE(IparCon,1) '* NI<1 or NI>NIMax'
      GOTO 9999
106   WRITE(IparCon,1) '* NJ<2 or NJ>NJMax'
      GOTO 9999
107   WRITE(IparCon,1) '* NK<3 or NK>NKMax'
      GOTO 9999

      !DX,DY,DZ
      !-----------------------------------------------------------------
201   WRITE(IparCon,1) '* DX(I)<0 at I:',I
      GOTO 9999
202   WRITE(IparCon,1) '* DY(J)<0 at J:',J
      GOTO 9999
203   WRITE(IparCon,1) '* DZ(K)<0 at K:',K
      GOTO 9999

      !TypSec
      !-----------------------------------------------------------------
301   WRITE(IparCon,1) '* TypSec(I)>Nsec at I:',I
      GOTO 9999

      !TypSoil
      !-----------------------------------------------------------------
401   WRITE(IparCon,1) '* TypSoil(J,K)<0 at J,K:',J,K
      GOTO 9999
402   WRITE(IparCon,1) '* TypSoil(J,K)>Nsoil at J,K:',J,K
      GOTO 9999
403   WRITE(IparCon,1) '* TypSoil(J,K)<0 at J,K:',J,K
      GOTO 9999
404   WRITE(IparCon,1) '* TypSoil(J,K)>Nsurf at J,K:',J,K
      GOTO 9999

      !front,back:
411   WRITE(IparCon,1) '* TypSoil(0,J,K)<>0 AND TypSoil(1,J,K)=0'
      WRITE(IparCon,1) '* at J,K:',J,K
      WRITE(IparCon,1) '* surface condition on tube must be adiabatic'
      GOTO 9999
412   WRITE(IparCon,1) '* TypSoil(NI,J,K)<>0 AND TypSoil(NI+11,J,K)=0'
      WRITE(IparCon,1) '* at J,K:',J,K
      WRITE(IparCon,1) '* surface condition on tube must be adiabatic'
      GOTO 9999
      !left,right:
421   WRITE(IparCon,1) '* TypSoil(I,1,K)=0 at I,K:',I,K
      WRITE(IparCon,1) '* no tube allowed on left surface when Nmod=1'
      GOTO 9999
422   WRITE(IparCon,1) '* TypSoil(I,NJ,K)=0 at I,K:',I,K
      WRITE(IparCon,1) '* no tube allowed on right surface when Nmod=1'
      GOTO 9999
431   WRITE(IparCon,1) '* TypSoil(I,1,K)=0 at I,K:',I,K
      WRITE(IparCon,1) '* no tube allowed on left surface when Nmod=2'
      GOTO 9999
434   WRITE(IparCon,1) '* TypSoil(I,NJ+1,K)<>0 at I,K:',I,K
      WRITE(IparCon,1) '* surface condition on right surface'
      WRITE(IparCon,1) '* must be adiabatic when Nmod=2'
      GOTO 9999
443   WRITE(IparCon,1) '* TypSoil(I,0,K)<>0 at I,K:',I,K
      WRITE(IparCon,1) '* surface condition on left surface'
      WRITE(IparCon,1) '* must be adiabatic when Nmod>2'
      GOTO 9999
444   WRITE(IparCon,1) '* TypSoil(I,NJ+1,K)<>0 at I,K:',I,K
      WRITE(IparCon,1) '* surface condition on right surface'
      WRITE(IparCon,1) '* must be adiabatic when Nmod>2'
      GOTO 9999
      !top,bottom:
451   WRITE(IparCon,1) '* TypSoil(I,J,1)=0 at I,J:',I,J
      WRITE(IparCon,1) '* no tubes allowed on upper surface'
      GOTO 9999
```

```
 452   WRITE(IparCon,1) '* TypSoil(I,J,NK)=0 at I,J:',I,J
       WRITE(IparCon,1) '* no tubes allowed on lower surface'
       GOTO 9999

       !PosInf
       !----------------------------------------------------------------
 501   WRITE(IparCon,1) '* PosInf(1)<1 OR PosInf(1)>NI'
       GOTO 9999
 502   WRITE(IparCon,1) '* PosInf(2)<1 OR PosInf(2)>NJ'
       GOTO 9999
 503   WRITE(IparCon,1) '* PosInf(3)<1 OR PosInf(3)>NK'
       GOTO 9999
 504   WRITE(IparCon,1) '* PosInf(4)<PosInf(1) OR PosInf(4)>NI'
       GOTO 9999
 505   WRITE(IparCon,1) '* PosInf(5)<PosInf(2) OR PosInf(5)>NJ'
       GOTO 9999
 506   WRITE(IparCon,1) '* PosInf(6)<PosInf(3) OR PosInf(6)>NK'
       GOTO 9999

       !Kair0,Kair1
       !----------------------------------------------------------------
 601   WRITE(IparCon,1) '* Kair0<0 OR Kair1<0'
       GOTO 9999

       !LamSoil,CvSoil
       !----------------------------------------------------------------
 701   WRITE(IparCon,1) '* LamSoil(Isoil)<=0 for Isoil:',Isoil
       GOTO 9999
 702   WRITE(IparCon,1) '* CvSoil(Isoil)<=0 for Isoil:',Isoil
       GOTO 9999

       !LamTub,CvTub
       !----------------------------------------------------------------
 801   WRITE(IparCon,1) '* LamTub<=0'
       GOTO 9999
 802   WRITE(IparCon,1) '* CvTub<=0'
       GOTO 9999

       !ThTub,CtubCor
       !----------------------------------------------------------------
 901   WRITE(IparCon,1) '* ThTub<=0'
       GOTO 9999
 902   WRITE(IparCon,1) '* CtubCor<=0'
       GOTO 9999
 903   WRITE(IparCon,1) '* Rfric<0'
       GOTO 9999

       !TypWat,Vwat
       !----------------------------------------------------------------
1001   WRITE(IparCon,1) '* TypWat(Idir)<1 OR TypWat(Idir)>2'
       WRITE(IparCon,1) '* for Idir:',Idir
       GOTO 9999
1002   WRITE(IparCon,1) '* Vwat(Idir)<0 for Idir:',Idir
       GOTO 9999

       !NiniSoil,NiniWat
       !----------------------------------------------------------------
1101   WRITE(IparCon,1) '* NiniSoil<1 OR NiniSoil>NiniMax'
       GOTO 9999
1102   WRITE(IparCon,1) '* NiniWat<1 OR NiniWat>NiniMax'
       GOTO 9999

       !TiniSoil,PosIniSoil
       !----------------------------------------------------------------
1201   WRITE(IparCon,1) '* PosIniSoil(Iini,1)<1'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,1)>NI'
       GOTO 1299
1202   WRITE(IparCon,1) '* PosIniSoil(Iini,2)<1'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,2)>NJ'
       GOTO 1299
1203   WRITE(IparCon,1) '* PosIniSoil(Iini,3)<1'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,3)>NK'
       GOTO 1299
1204   WRITE(IparCon,1) '* PosIniSoil(Iini,4)<PosIniSoil(Iini,1)'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,4)>NI'
       GOTO 1299
1205   WRITE(IparCon,1) '* PosIniSoil(Iini,5)<PosIniSoil(Iini,2)'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,5)>NJ'
       GOTO 1299
1206   WRITE(IparCon,1) '* PosIniSoil(Iini,6)<PosIniSoil(Iini,3)'
       WRITE(IparCon,1) '* OR'
       WRITE(IparCon,1) '* PosIniSoil(Iini,6)>NK'
       GOTO 1299
```

```
 1299 WRITE(IparCon,1) '* for Iini:',Iini
      GOTO 9999

      !ThIniWat,PosIniWat
      !---------------------------------------------------------------
 1301 WRITE(IparCon,1) '* PosIniWat(Iini,1)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,1)>NI'
      GOTO 1399
 1302 WRITE(IparCon,1) '* PosIniWat(Iini,2)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,2)>NJ'
      GOTO 1399
 1303 WRITE(IparCon,1) '* PosIniWat(Iini,3)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,3)>NK'
      GOTO 1399
 1304 WRITE(IparCon,1) '* PosIniWat(Iini,4)<PosIniWat(Iini,1)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,4)>NI'
      GOTO 1399
 1305 WRITE(IparCon,1) '* PosIniWat(Iini,5)<PosIniWat(Iini,2)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,5)>NJ'
      GOTO 1399
 1306 WRITE(IparCon,1) '* PosIniWat(Iini,6)<PosIniWat(Iini,3)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosIniWat(Iini,6)>NK'
      GOTO 1399
 1399 WRITE(IparCon,1) '* for Iini:',Iini
      GOTO 9999

      !Nopt
      !---------------------------------------------------------------
 1401 WRITE(IparCon,1) '* Nopt<0 or Nopt>NoptMax'
      GOTO 9999

      !TypOpt,PosOpt
      !---------------------------------------------------------------
 1501 WRITE(IparCon,1) '* PosOpt(Iopt,1)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,1)>NI'
      GOTO 1599
 1502 WRITE(IparCon,1) '* PosOpt(Iopt,2)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,2)>NJ'
      GOTO 1599
 1503 WRITE(IparCon,1) '* PosOpt(Iopt,3)<1'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,3)>NK'
      GOTO 1599
 1504 WRITE(IparCon,1) '* PosOpt(Iopt,4)<PosOpt(Iopt,1)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,4)>NI'
      GOTO 1599
 1505 WRITE(IparCon,1) '* PosOpt(Iopt,5)<PosOpt(Iopt,2)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,5)>NJ'
      GOTO 1599
 1506 WRITE(IparCon,1) '* PosOpt(Iopt,6)<PosOpt(Iopt,3)'
      WRITE(IparCon,1) '* OR'
      WRITE(IparCon,1) '* PosOpt(Iopt,6)>NK'
      GOTO 1599
 1599 WRITE(IparCon,1) '* for Iopt:',Iopt
      GOTO 9999

      !Ntub,IflowIni,IflowEnd
      !---------------------------------------------------------------
 2101 WRITE(IparCon,1) '* Ntub>NtubMax'
      GOTO 9999

      !PosTub
      !---------------------------------------------------------------
 2201 WRITE(IparCon,1) '* incompatibility of tube position'
      WRITE(IparCon,1) '* between cross-section at I:',I
      WRITE(IparCon,1) '* and cross-section at I',IflowIni
      WRITE(IparCon,1) '* at J,K:',J,K
      GOTO 9999

 9999 WRITE(IparCon,1)
      WRITE(IparCon,1)'*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'

      WRITE(*,1)
      WRITE(*,1) '!!! TYPE 61: error in parameter definition file !!!'
      WRITE(*,1) '!!!       => check in parameter control file    !!!'
      WRITE(*,1)

      STOP
```

```
      END
C***********************************************************************
```