La 5e conférence d'IBPSA - Canada
The 5th IBPSA - Canada Conference
Québec 21-22 mai

AN EXTENSIVE METHODOLOGY FOR COUPLING TRNSYS COMPONENTS TO ESP-R

Weimin Wang[1], Ian Beausoleil-Morrison[2]

[1]CANMET Energy Technology Centre, Natural Resources Canada, Ottawa, Canada
[2]Department of Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada

ABSTRACT

A methodology has been developed to enable the direct compilation and use of TRNSYS components within ESP-r. This "wrapper" is treated as a component in ESP-r's plant database. It differs from other conventional components in two aspects. First, this wrapper component has dummy values in the plant database in terms of the number of nodes, connections, and parameters while meaningful values are read in from a text file. Second, the TRNSYS component has all its self-coupling coefficients equal to one and it has no cross-coupling coefficients. The internal coupling strategy is followed to write the wrapper. However, efforts have been taken to minimize source code changes. In many cases, coupling a TRNSYS type is simplified to prepare an input text file. The use of this wrapper is demonstrated with a case study where TRNSYS type 60 couples to ESP-r.

INTRODUCTION

Simulation programs have been increasingly used in the practice of building design over the last decades. This is in part due to the enhanced ability of building simulation to model a large variety of system types and components in different domains. However, as buildings become more complex and more stringent design requirements are stipulated, it is often the case that no single simulation program is sufficient to solve all problems at hand. Rather, different simulations need to be integrated together in order to consider the interactions between building subsystems and to provide a holistic view of building performance in terms of functionality, occupant comfort, resource efficiency, and life-cycle cost. Integrated simulation is usually achieved with one of the following five approaches (Citherlet et al. 2001, Hensen et al. 2004):

Combination of stand-alone programs. With this approach, several simulation tools are used together but they are treated as separate, unrelated programs. For example, one simulation tool is employed for energy estimation while another is separately used for indoor illuminance calculation. There are several evident drawbacks with this stand-alone approach. First, without a unified product model, serious data redundancy may exist and the risk of inconsistent data across different simulation models is high. Second, the interactions between the involved systems cannot be captured (for example, the interaction between the lighting and HVAC systems). Due to these inherent limitations, the simple combination of different simulation programs is hard to be called "integrated simulation" in strict meaning.

Data and process model integration. This approach intends to provide a single program covering different domains. For example, ESP-r (Clarke 2001) is a widely used tool for the integrated simulation of the thermal, visual and acoustic performance of buildings. The data and process model integration approach has several major advantages. First, it addresses the problem of data redundancy and inconsistency associated with the stand-alone approach. Second, it avoids the switch between different simulation programs for a holistic view of building performance; therefore, the user needs to master the interface of one instead of multiple programs. Moreover, a single simulation environment facilitates the consideration of the dynamic interactions between building subsystems. However, the data and model integration approach normally leads to a monolithic program which might be difficult to maintain in the long term (Hensen et al. 2004). In addition, the user has to be restricted to the features available in the integrated program.

Data model interoperation. This approach works on the integration of data models for different simulation programs. Data interoperability can be achieved with either data sharing or data exchanging. With data sharing, a generic data model provides a common basis for domain-specific applications. A representative generic data model for buildings is the COMBINE project (Augenbroe 1992). With data exchanging, different applications exchange the information embedded in their data models through data translators. A normal procedure requires that a data translator

transform the data model underlying one application to a neutral file format. Then, the neutral format file is read in by another translator to generate a data model suitable for other applications. Data interoperation reduces the degree of data redundancy and inconsistency. However, data integration alone can hardly satisfy the need of integrated simulation which has a strong process focus rather than data focus (Augenbroe et al. 2004).

Process model interoperation. This approach intends to reuse the available component models for thermal, electrical and other physical processes. Process models can be reused via two strategies: internal coupling (Djunaedy et al. 2003) and neutral model format (Vuolle and Bring 1997). With internal coupling, a simulation program incorporates a component model from another program by converting it to a subroutine of the host simulation. Examples include the coupling between TRNSYS and COMIS (Dorer and Weber 1999), EnergyPlus and COMIS (Huang et al. 1999), and the incorporation of several TRNSYS components into ESP-r (Hensen 1991; Aasem 1993). The neutral model format works in a similar way as that for the data exchange in data model interoperation. A translator is required to transform a model in the neutral format to one in an understandable form for a specific simulation (Nataf 1995). Process model interoperation can save the resources used to enrich a simulation program. However, the source code of the coupled models may need to be modified.

Data and process model cooperation. This approach intends to link simulations and exchange the coupled data across them. Since the simulations work as separate and executable programs, the data and process model cooperation is also called external coupling (Djunaedy et al. 2003). Depending on the time of program linkage, there are two ways of external coupling: sequential coupling and run-time coupling (Trcka et al. 2006). In sequential coupling, a simulation runs first and after it finishes, some of its outputs will be used as the inputs to another program. Due to the lack of any feedback from the second simulation program, sequential coupling can be applied in limited cases when the second simulation has no impact on the coupled data. In run-time coupling, different simulations coexist and communicate at run time. Some critical run-time coupling issues include the inter-process communication mechanisms, the definition of exchanged data, time synchronization, and coupling strategies. Detailed discussions about these critical issues can be found in (Yahiaoui et al. 2003; Djunaedy et al. 2003; Trcka et al. 2006). Run-time coupling makes it possible to perform integrated simulation with different programs while considering the dynamic interactions between building subsystems. This is particularly useful when the combined simulations are powerful in different domains, e.g, the building and plant domains.

Following the internal coupling approach, a methodology was developed for coupling TRNSYS components to ESP-r. In contrast to previous work (Hensen 1991; Aasem 1993) that converted several TRNSYS types to ESP-r components, the current work produced a generic "wrapper" that can encapsulate any TRNSYS type representing a plant component. With the aid of this wrapper, the TRNSYS source code can be coupled into ESP-r and used within its plant domain. Moreover, the TRNSYS source code rarely needs to be changed if its syntax complies with the compiler used by ESP-r. This wrapper can be used in the following situations: 1) The model for a required component does not exist in ESP-r but is available in TRNSYS; and 2) The existing model lacks some features while they are available in a TRNSYS type.

MODELLING PLANT COMPONENTS IN ESP-R AND TRNSYS

In ESP-r (Clarke 2001), each plant component is represented by one or more discrete control volumes (nodes). For each node, the governing equations are established according to the principles of conservation for mass, energy and momentum. In the context of plant modelling, quantities such as enthalpy and fluid flow rate are of interest. Therefore, for most plant components, the modelling variables to represent a nodal state are temperature, first phase and second phase mass flow rates.

A component must be available in the plant component database of ESP-r for the user's selection. While a detailed description of the data entries is out of the scope of this paper, the groups of data fields for a plant component in the database are summarized below:

- General description. The plant component is described from its name, a brief description of its usage, the date of adding it to the database, the component type, and the component code.
- Nodal scheme. This group of data is essential to construct the matrix structure of a plant component and its connections to other components. The nodal scheme is described with the number of nodes, the number of self coefficients with non-zero value in the component matrix, the position for each non-zero self coefficient in the component matrix, the number of external (receiving) connections to each node, and a

variable type index for each node to determine its supported state variables and its nature (i.e., solid, water, air).

- Input data. This group of data includes the number of static data items (i.e, ADATA or BDATA), the number of control variables. Each static data item needs to have a description, default value and valid range. Each control variable has a brief description.

- Output data. This group contains the number of additional output parameters. Each additional output has a brief description and its type index (1-5).

In ESP-r, each plant component has a coefficient generator which generates the additional output values and all self, cross, and right hand coefficients. The calculation of these coefficients is based on the input data (ADATA or BDATA, which may not take default values), the control variables, and the nodal state variables determined from the previous step.

Based on the nodal scheme for each component and their connections, an overall matrix equation can be set up for a whole plant network, where all the matrix coefficients are contributed from the plant components' coefficient generators. The matrix equation is then solved to obtain the nodal state variables for the current step. Since the equation set is highly nonlinear, iteration is required to update and resolve the matrix until convergence is achieved.

In contrast to ESP-r using the simultaneous modelling techniques, TRNSYS uses the sequential modelling technique. This implies that for each simulation time step, the computation starts from a known boundary condition and work on the plant components one by one according to some prescribed path. The connections between plant components are represented as their input-output relationships. Given inputs, a TRNSYS component (i.e, Type) can determine its outputs, without the need of resolving a system matrix.

TRNSYS makes a distinction between inputs according to whether their values change with time. The time-dependent inputs are called INPUTS such as temperature and flow rate while the time-independent inputs are called PARAMETERS such as area and mass. During simulation, a TRNSYS component turns the current values of the INPUTS and PARAMETERS into OUTPUTS, some of which may be used as the inputs for the downstream components. For those components that solve differential equations numerically, DERIVATIVES are needed to specify the initial values, such as the initial temperatures of various nodes in a thermal storage tank or the initial zone temperatures in a multi zone building.

THE TRNSYS WRAPPER

In the current stage, the wrapper intends to support the coupling of advanced plant components in TRNSYS. Hence, the wrapper is treated as a plant component. In ESP-r, adding a plant component mainly involves two aspects of work: this component is defined and entered in the plant database; its coefficient generator is implemented and installed. These two aspects are discussed below.

As presented in the previous section, a component entry in ESP-r's plant database covers four groups of data fields: general description, nodal scheme, input, and output. Except for general description, data fields of all the other three groups cannot be defined in advance for the wrapper without knowing the TRNSYS types. Further, even for a specific TRNSYS type, it may be impossible to predefine many data fields in the plant database. For example, Type 60 indicates a thermal storage water tank with up to three immersed heat exchangers. Clearly, the number of heat exchangers affects how Type 60 is described as an ESP-r plant component in terms of its nodal scheme and the number of inputs. Therefore, an important task is to enable one wrapper component to represent multiple, unknown TRNSYS types.

The wrapper's polymorphic feature is achieved by assigning dummy values to all data fields related to nodal scheme, input, and output. Meaningful values are read in from a text file which will be elaborated in the next section. Thus, the user prepares a TRNSYS component via a text file instead of the database file. From the user's point of view, the text file is much easier to handle than the database file with a rigid format.

Like other plant components, the wrapper has a coefficient generator to prepare the required matrix coefficients. Since the state variables (i.e., temperature and flow rate) are known after calling the coupled TRNSYS type, the wrapper has the following distinguished characteristics:

- The number of its self-coupling coefficients is equal to the number of nodes defined for the coupled TRNSYS component. All its self-coupling coefficients are on diagonal positions and equal to one. Therefore, the wrapper has an entity matrix structure.

- It has no cross-coupling coefficients even if it has external receiving connections. External connections affect a TRNSYS component via the input variables.

- All its right hand coefficients can be obtained from the outputs or inputs of the coupled TRNSYS type.

Essentially, the wrapper's coefficient generator works in three steps: 1) Prepare the required inputs for a TRNSYS type; 2) Call that TRNSYS type; and 3) Process the wrapped component's outputs, including its right-hand coefficients and additional outputs.

The first and the third steps involve mapping the inputs and outputs between the coupled TRNSYS type and other ESP-r components in the plant network. Note that different dimension units may be used in TRNSYS and ESP-r. For example, the unit for mass flow rate is kg/s in ESP-r while it is kg/hr in TRNSYS. Hence, unit conversion needs to be considered in mapping the inputs and outputs between these two programs. As of the second step, while calling a TRNSYS type is straightforward, the wrapper has paid special attention to the following two points:

- For the first iteration within the first plant time step, the TRNSYS type is called four times, corresponding respectively to the version signing call, the initialization call, the TIME=TIME0 call, and the normal time step call in TRNSYS (Klein et al. 2004). This treatment is necessary to retrieve some INFO array values used in the interface of TRNSYS types and to check the validity of TRNSYS inputs.

- After the first iteration at the beginning of plant simulation, the TRNSYS type is called only once per iteration step to construct and solve the plant system matrix. As mentioned before, a TRNSYS call can solve all its nodal state variables. Therefore, the TRNSYS type should be called only once for each iteration; otherwise, it will be simulated increasingly ahead of the plant simulation step. Since the first phase mass balance is handled first by ESP-r, a TRNSYS call is needed only if the state variable indicates the first phase mass flow rate.

TRNSYS WRAPPER STEWARDSHIP

The TRNSYS wrapper can be applied in plant simulation according to the following procedure:

Step 1: add a placeholder in the wrapper's coefficient generator to accommodate the TRNSYS type to be coupled. This is a simple task because the user just needs to copy the snippet of code for an existing placeholder (e.g., for Type 60) and to change the type number accordingly.

Step 2: copy the TRNSYS source code for the coupled component to ESP-r. This is also simple because TRNSYS has a strong modular structure and the source code is available for end-users. Note that the source code of TRNSYS types is usually placed in the file `esruplt/trstyp.F`.

Step 3: install ESP-r and remove the coding bugs. Although the wrapper has been developed to facilitate the reuse of TRNSYS component models, the source code of the coupled TRNSYS type may need some changes. These changes are due to two major reasons. First, compilers or compiler settings may be different for ESP-r and TRNSYS. Second, the original connections between an individual Type and the whole TRNSYS simulation environment are cut off.

Incompatible compilers could cause the following coding problems:

- Syntax error. For example, TRNSYS uses the style of continuation tab format in many places, where a continuous line starts with a tab character followed by any digit except 0. However, this tab format is not allowed if ESP-r is installed with the GNU Fortran compiler (gfortran).

- Unsupported features. For example, Fortran 90 features such as modules and array operations as a whole are widely used throughout TRNSYS source code. However, they are not supported if ESP-r is installed with the GNU Fortran 77 compiler (g77).

- Unsupported functions. For example, `JFIX()` is a function frequently used in TRNSYS to round a value to its nearest integer. However, it is not supported by the g77 compiler.

Blocked connections are usually reported as missing the declaration of variables and functions. As mentioned before, TRNSYS types consistently use the module feature for clarity. All global constants and utility functions are put in two modules named respectively as `TrnsysConstants` and `TrnsysFunctions`. Thus, coding errors are reported whenever a TRNSYS type refers to a variable or function in those two modules.

Measures are taken to address some of the above issues that could occur in TRNSYS coupling. For example, based on our preliminary study, those global variables normally used in TRNSYS types were declared in the header file `trnsys.h`. In addition, the unsupported functions and several TRNSYS utility functions have been implemented in ESP-r (the file `trnsys_lib.F`). Therefore, in many cases, the code debugging process can be simplified if the user starts from adding a line to include the `trnsys.h` head file and commenting out the module use statements. Then, additional efforts are required to address specific problems. The time used in this step is TRNSYS type dependent. Many iterations

between ESP-r installation and code changing may be needed.

Step 4: prepare the TRNSYS input file. The input data file is named as `trnsysInput?.txt`, where the question mark indicates the serial number of a TRNSYS component in the plant network. For example, if a TRNSYS component is the third one in the plant network, the corresponding file name should be `trnsysInput3.txt`. Since each coupled TRNSYS component has one input file, there may have multiple TRNSYS input files. All these files are put in the folder `trnsys`, which locates in the upper directory of the ESP-r configuration file (.cfg).

Data fields in the TRNSYS input file can be separated by spaces, tabs, or commas. However, if a field is a phrase with blanks, it must be followed by a comma. The data fields in a TRNSYS input file can be grouped into five sections: component configuration, parameters, inputs, derivatives and outputs. These sections are presented below.

The section of component configuration gives the type number for the coupled TRNSYS component, the number of nodes for this component, the number of external (receiving) connections and the variable type index for each node. Detailed definitions about the number of external connections and the variable type index can be found in Hensen (1991).

The parameters section provides the number of TRNSYS parameters, and for each parameter, there is a brief description, a switch indicating whether this parameter needs to be changed in ESP-r, and some switch-dependent fields. Normally, a TRNSYS parameter does not need to change during the simulation. In this case, the user defines its value in the input file. A parameter could be considered for changing its value during the simulation in the following cases: 1) it indicates the thermal-physical property (e.g., specific heat and density) of a fluid; 2) it indicates the surrounding temperature for that component; and 3) its value cannot be defined in advance but can be obtained from the additional output of another component.

The inputs section provides the number of TRNSYS inputs, and for each input, there is a brief description, the input type, and some type-dependent fields. Currently, the wrapper supports the following input types: the outdoor temperature, a control variable, the surrounding temperature, a fixed value, the state variables of a receiving connection, the additional output of another plant component. Additional required information varies with the input type. For example, the serial number of a plant control loop is needed if the input indicates a control variable. On the other hand, the node and coupling are needed if the input indicates the state variable of a receiving connection. Nevertheless, the implemented input types may be insufficient and the user can easily enlarge this scope by referring to the code for available input types.

The derivatives section provides the number of derivatives, the initialization method, and the initial values if they are defined by the user. In the current stage, the wrapper assumes that the variables represented by these derivatives indicate the temperature. These variables can be initialized by either ESP-r or the user. In the former case, the temperature of each variable is assigned as 20 °C. In the latter case, the user needs to provide the initial temperature for each variable.

The outputs section establishes the mechanism to map ESP-r outputs from TRNSYS. This section includes the number of ESP-r outputs (i.e., right hand coefficients) and additional outputs. For each ESP-r output, the way mapping from TRNSYS to the three nodal state variables is defined. Because TRNSYS does not give special attention to the mass balance, the first and the second phase mass flow rate may not appear in its outputs. Therefore, besides TRNSYS outputs, the right hand coefficients can be mapped also from TRNSYS inputs or zeroized. For Type 60, for example, the flow rate of the immersed heat exchanged is not an output but it can be easily known from the input. In addition, because the second phase mass flow rate is not modelled, it does not appear in either TRNSYS inputs or outputs. Thus, the second phase mass flow rate can be zeroized to establish the plant matrix. As of the additional outputs, the mapping mechanism is simpler because this involves TRNSYS outputs only.

VALIDATION

A simple test for the stratified water tank is used here as a preliminary validation of the wrapper. The cylindrical tank has a volume of 324 L and its height is 1.8 m. The heat loss factor is 0.1 W/(m^2·°C). In this test, the surrounding temperature is kept at 5 °C. The tank is at an initial temperature of 30 °C. Hot water at 50 °C is added at the top of the tank at a rate of 36 L/h and the same amount of water is withdrawn at the bottom. At the same time, cold water at 10 °C is added at the bottom at a rate of 36 L/h and the same amount of water is withdrawn at the top.

TRNSYS type 60 was used to model the stratified water tank. This model was modelled by ESP-r via the wrapper and also by TRNSYS directly. In both cases,

the simulation duration is 2 hours with one minute time step; and the tank is modelled with 50 layers. Figure 1 shows the top, the average, and the bottom temperatures of the tank. The results simulated by ESP-r via the wrapper closely match those simulated directly by the TRNSYS program.

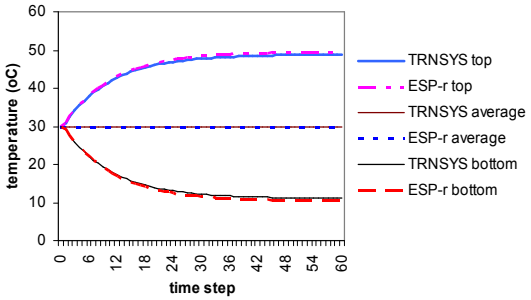

Figure 1: Comparison of simulation results for the flow-through test with and without the TRNSYS wrapper

CASE STUDY

A solar water heating system is employed here to demonstrate the use of the TRNSYS wrapper. As shown in Figure 2, the system consists of a solar collector, which is in turn connected to a solar tank with an immersed heat exchanger. A pump is used to circulate the fluid (50% propylene glycol) between the solar collector and the solar water tank. This pump is turned on when the temperature difference between the two connected components exceeds a certain value. There is an auxiliary water tank to supply the hot water for space heating. In case that the water from the solar tank cannot meet the heat demand, the electric heater within the auxiliary tank is used to heat the water until its temperature reaches the predefined setpoint. For the pump and the fan connected to the air heating coil, they are turned on when the space temperature drops below a setpoint.

The above system is used to provide space heating for the CCHT house (CCHT 2008), which has two above-grade storeys and a basement. This house was modelled with four thermal zones: the basement, the attic space, the attached garage, and the two storeys of living space. Only the living space was conditioned with a temperature setpoint of 20.5 °C and a deadband of 1 °C while the other three zones were "free floating", varying in response to the thermal contact with the other zones and the outdoors. More details about the building model can be found in Purdy and Beausoleil-Morrison (2001).

For the plant simulation, there are several components in ESP-r for water tanks with different features, but no component is currently available to be capable of modelling a stratified tank with either immersed heat exchangers or embedded heaters. Therefore, the effect of stratification would have to be ignored if this system were simulated with native ESP-r components. On the other hand, TRNSYS type 60 is an advanced plant component that can model a stratified water tank with up to two heaters and three immersed heat exchangers. However, in comparison with ESP-r, TRNSYS is weak in modelling the building. Therefore, it is ideal to couple TRNSYS type 60 in ESP-r plant simulation.

In order to investigate the impact of stratification on the system performance, two plant models were established. The first model consists of only native ESP-r components, of which the solar tank and the auxiliary tank were modelled respectively with component no 86 (storage tank with immersed coil) and component no 58 (electrically heated water tank with 2 connections). These two components were replaced in the second model with the wrapper coupling TRNSYS type 60 while all other components keep the same.

Most parameter values of the plant components are taken from a previous study (Haddad et al. 2007). However, since domestic hot water is not considered in this study, the temperature setpoint for the auxiliary tank was changed from 56 to 60 °C and the deadband was increased from 2 to 10 °C.

Figure 3 gives an example of the wrapper's input text file for the solar tank. Due to the space limitation, not all input data are presented there. The solar tank is represented with three nodes. The first node represents the water flowing to the auxiliary tank; the second node represents the fluid flowing out of the heat exchanger; while the third node represents the water close to the inlet of the heat exchanger. Since Type 60 reports all layer temperatures in its outputs, the user can easily change the number and the position of nodes as required. Therefore, the wrapper provides significant flexibility for component discretization.

Table 1: Energy consumption of the electrical heater in the auxiliary tank

simulation period	energy consumption (KJ)		difference (%)
	with wrapper	without wrapper	
Jan. 1-7	55114	51774	6.1
Mar. 1-7	45437	41412	8.9

The simulation was run for two periods: January 1-7 and March 1-7. Simulation results are presented in Table 1, where the numbers refers to the energy consumed by the heater in the auxiliary tank. The table shows that ignoring stratification underestimates energy consumption by 6-9%. This complies with a previous

study (Wang et al. 2007) which found that a fully-mixed model may lead to an underestimation of energy consumption by 8-15% for a gas-fired water tank. The degree of underestimation is lower here because an electrically heated tank has higher efficiency than a gas-fired water tank.

CONCLUSIONS

ESP-r and TRNSYS are two widely used simulation programs with different distinct features. Since neither of them can solve all problems, it may be desired to integrate them together for a particular task. The wrapper presented in this paper is useful for coupling TRNSYS types to ESP-r. With one component entry added to the ESP-r database, the user can couple multiple TRNSYS types and these coupled components can have different nodal schemes. The wrapper facilitates coupling TRNSYS types by reducing the potential work of code changes and operating on the input text files. The future work includes applying the wrapper for more robust validation and extending its scope from the plant domain to the electrical domain.

REFERENCES

Aasem E.O. (1993), Practical simulation of buildings and air-conditioning systems in the transient domain. Ph.D. Thesis, University of Strathclyde, Glasgow, UK.

Augenbroe G. (1992), 'Integrated building performance evaluation in the early design stages', *Building and Environment*, 27 149-161.

Augenbroe G. Wilde P. Moon H.J. and Malkawi A. (2004), 'An interoperability workbench for design analysis integration', *Energy and Buildings*, 36 737-748.

CCHT. 2008. http://www.ccht-cctr.gc.ca, accessed in January, 2008.

Citherlet S. Clarke J.A. and Hand J. (2001), 'Integration in building physics simulation', *Energy and Buildings*, 33 451-461.

Clarke J.A. (2001), *Energy Simulation in Building Design* (2nd edition), Butterworth-Heinemann.

Djunaedy E. Hensen J.L.M. and Loomans M.G.L.C. (2003), 'Towards external coupling of building energy and airflow modeling programs', *ASHRAE Transactions,* 109 771–787.

Dorer V. and Weber A. (1999), 'Air, contaminant and heat transport models: integration and application", *Energy and Buildings*, 30 97-104.

Haddad K. Purdy J. and Sibbitt B. (2007), 'Simulation of residential solar DHW systems in HOT3000 software', Proceeding of the 2nd Canadian Solar Buildings Conference, Calgary, Canada, 8 pages.

Hensen J.L.M. (1991), On the thermal interaction of building structure and heating and ventilating system. Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Hensen J. Djunaedy E. Radosevic M. and Yahiaoui A. (2004), 'Building performance simulation for better design: some issues and solutions', Proceeding of the 21th conference on Passive and Low Energy Architecture. Eindhoven, The Netherlands, pp.1185-1190.

Huang J. Winkelmann F. Buhl F. Pedersen C. Fisher D. Liesen R. Taylor R. Strand R. Crawley D. and Lawrie L. (1999), 'Linking the COMIS multi-zone airflow model with the EnergyPlus building simulation program', Proceeding of the 6th International IBPSA Conference, Kyoto, Japan, pp. 1065–1070.

Klein SA, et al. 2004 TRNSYS, version 16. Solar Energy Laboratory, University of Wisconsin.

Nataf J.M. (1995), 'A direct translator from neutral model format to the SPARK simulation environment', *Energy and Buildings*, 23 131-139.

Purdy J. and Beausoleil-Morrison I. (2001), 'The significant factors in modelling residential buildings', Proceeding of the 7th International IBPSA Conference, Rio de Janeiro, Brazil, pp. 207-214.

Trcka M. Hensen J.L.M. and Wijsman A.J.Th.M. (2006), 'Distributed building performance simulation – A novel approach to overcome legacy code limitations', *HVAC&R Research*, 12 621-639.

Yahiaoui A. Hensen J. and Soethout L. (2003), 'Integration of control and building performance simulation software by run-time coupling', Proceeding of the 8th International IBPSA Conference, Eindhoven, The Netherlands, pp. 1435-1441.

Vuolle M. and Bring A. (1997), 'An NMF based model library for building climate and energy simulation', Proceeding of the 6th International IBPSA Conference, Kyoto, Japan, 7 pages.

Wang W. Beausoleil-Morrison I. Thomas M. and Ferguson A. (2007), 'Validation of a fully-mixed model for simulating gas-fired water storage tanks', Proceeding of the 10th International IBPSA Conference, Beijing, China, 8 pages.

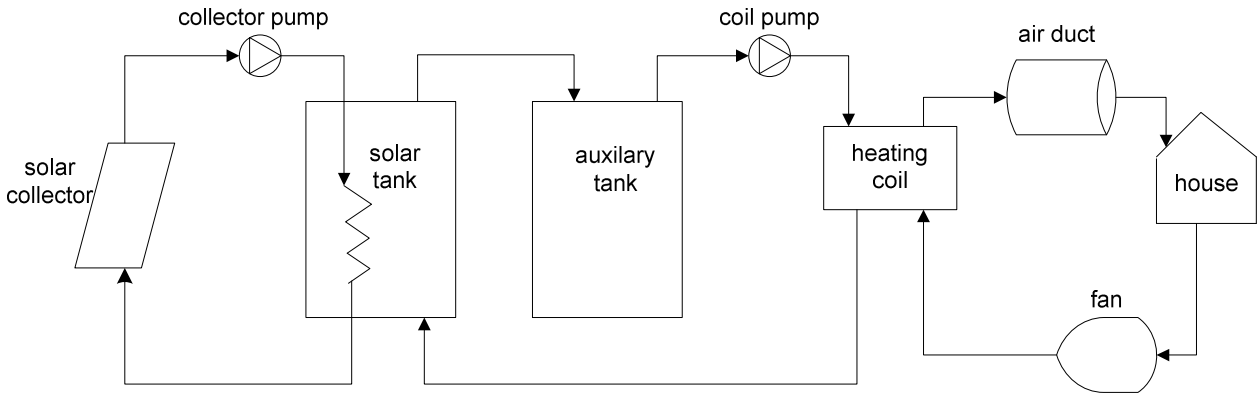

Figure 2: System diagram for the solar water heating

```
TRNSYS-Type,   60
Nodes-Number,   3
Node-Connection,   1, 1, 0    #0 means no external connections
Variable-Type,     20, 20, 20   #all three nodes represent water
Parameters,     44
  1,   User-specified inlet positions,  Y,     2  # Y: the parameter does not change
  2,   Tank volume,                     Y,     0.227
# ……
  5,   Height of flow inlet 1,          Y,     0.3
  6,   Height of flow outlet 1,         Y,     1.25
# ……
  9,   Fluid specific heat,             N,   3, -1, CP1  #CP1 is the unit code
# …… other parameters are omitted here
Inputs,   13
  1,   Flow rate at inlet 1,            2,   1,  1,  MF1 #1st coupling for 1st node
  2,   Flow rate at outlet 1,           0,  -2
# ……
  5,   Temperature at inlet 1,          1,   1,  1,  TE1
# ……
 10,   Flow rate for HX,                4,   2,   1,  MF1
# ……
 13,   Nusselt exponent for HX,         0,   0.25
Derivatives,  50,   1       # 1 means that the user defines initial values
  1,      30
# ……
Outputs,   3
# ID      temperature          1st phase mass          2nd phase mass
  1,      5,     TE1,           2,    MF1,             0,  MF1
  2,     23,     TE1,          -10,   MF1,             0,  MF1
  3,     55,     TE1,           0,    MF1,             0,  MF1
Additional-Outputs,  3
  1,  AVGTEMP,      17,   TE1  #average temperature
# ……
```

Figure 3: The input text file for the solar tank