A Demonstration of the Run-Time Coupling between ESP-r and TRNSYS

Francesca Macdonald[a], Romain Jost[b], Ian Beausoleil-Morrison[a],
Michaël Kummert[b], Timothy McDowell[c], and Alex Ferguson[d]

[a]Carleton University, Ottawa Canada, [b]École Polytechnique de Montréal, Montréal Canada,
[c]Thermal Energy System Specialists, LLC, Madison USA,
[d]Natural Resources Canada, Ottawa Canada

ABSTRACT

A run-time coupling between ESP-r and TRNSYS has been developed to offer the possibility of an integrated simulation environment that simultaneously manages detailed performance prediction of both building physics and energy systems. The co-simulator's design and implementation are described, together with a series of tests used in its verification. An example of the type of system which can be addressed by the coupling of these tools is demonstrated, and its computational efficiency is assessed.

INTRODUCTION

The topic of run-time coupling (also known as co-simulation) has been addressed by numerous authors within the past decade(Dorer and Weber, 1999; Huang et al., 1999; Janak, 1997; Djunaedy, 2005; Trčka et al., 2009; Wetter, 2010). This paper focuses upon the specific example of the run-time coupling between ESP-r and TRNSYS that is under development (Beausoleil-Morrison et al., 2011). The coupling of these complementary tools exploits the strengths of both simulation tools to facilitate the modelling of innovative building and energy system configurations more vigorously than either simulation program could achieve on its own. These new co-simulation capabilities can support the analysis of advanced systems such as Net Zero Energy Housing (NZEH). NZEH requires the rigorous treatment of a coupled simulation to accurately determine energy impacts of low-energy homes (load reduction/ energy efficiency) and renewable energy production strategies and systems.

This paper describes the design of the co-simulation environment in which ESP-r and TRNSYS iterate internally and between the two programs by exchanging data within the time-step. Each program iterates and converges internally, once converged, the harmonizer is called to ensure the other program has also converged. If both programs do not simultaneously converge the two programs are forced to continue iterating (with the last values exchanged) at the same time step until overall convergence is reached. The design is based on running both ESP-r and TRNSYS on separate threads that are managed by a middle-ware. The functionality and implementation of this middle-ware, referred to as the Harmonizer, is explained together with the modifications required to the ESP-r and TRNSYS source code to enable their cooperation with the Harmonizer. The main focus of the paper is to report the results obtained from a series of tests devised to verify the co-simulator, and to demonstrate its potential application. The computational efficiency is considered by comparing the simulation times of the co-simulator with ESP-r only and TRNSYS only simulations of the same systems.

DESIGN OF CO-SIMULATOR

The co-simulator is based on the use of shared libraries. Shared libraries are libraries of functions that are loaded into memory by executables at load time or run time, and can be shared with other executable files. A shared library is known as a dynamic link library (DLL) under the Windows operating system, which this co-simulator is currently limited to. Compiling the harmonizer, ESP-r and TRNSYS as shared libraries, allows each of the libraries to have access to the subroutines in the other libraries. The co-simulator comprises a launching executable which loads and initializes the Harmonizer shared library and executes the *main* function. This in turn loads and initializes the ESP-r and TRNSYS shared libraries, and executes the co-simulation. The design of the co-simulator is based on ESP-r and TRNSYS on separate threads to increase its computational efficiency, as this enables each program to perform its internal calculations independent of the other program. The Harmonizer manages the data exchange and evaluates the convergence between the two programs. The justification for a number of the software design decisions made regarding the co-simulator are discussed elsewhere (Beausoleil-Morrison et al., 2011).

The role of the Harmonizer is to establish and maintain the coupling between the ESP-r and TRNSYS shared libraries. It does this by performing the following functions:

- Loading the ESP-r and TRNSYS DLLs and executing the initialization processes (including parsing input files) required prior to simulation.
- Creating the threads to initiate the simulation time loop in each program.
- Synchronizing the data exchange between ESP-r and TRNSYS.
- Controlling the convergence between the two solvers.
- Acting as the simulation clock, controlling the marching forward through time.

Figure 1 describes the design of the Harmonizer. The Harmonizer receives its inputs from an input (text) file

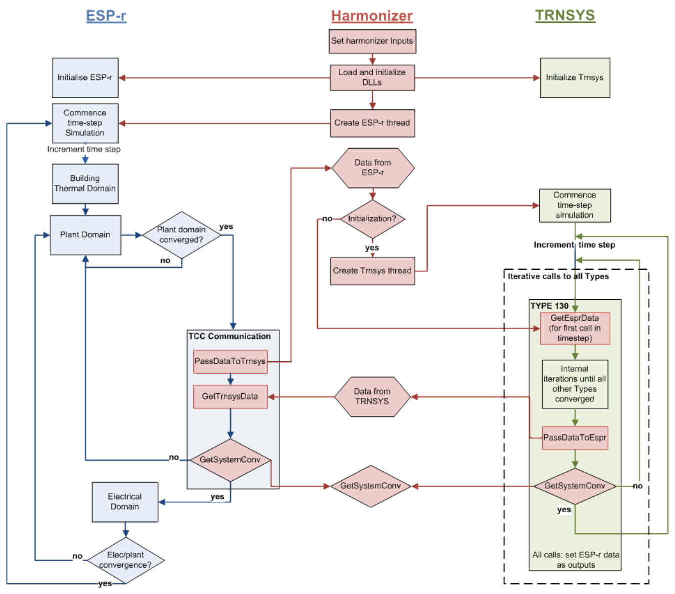


Figure 1: Harmonizer design

which stores the location of the ESP-r and TRNSYS DLLs as well as the location of the input files required by each program. Then each of the DLLs is loaded and ESP-r and TRNSYS are initialized. For the simulation to produce meaningful results both ESP-r and TRNSYS must run with the same weather file and with the same simulation period and number of time-steps. With the current design satisfying this condition is the user's responsibility. The ESP-r thread is created first which begins the simulation. For every time-step ESP-r first solves building domain, followed by the plant domain. Having achieved convergence in the plant domain, communication with the Harmonizer is initiated. The first call to the Harmonizer's PassDataToTrnsys function creates the TRNSYS thread and commences its simulation. The first call TRNSYS makes to the Harmonizer is the call to GetEsprData, to collect the boundary condition data that ESP-r passed to the Harmonizer. TRNSYS subsequently uses that data in its own internal iterations until convergence is reached and then passes the newly calculated boundary conditions back to the Harmonizer with PassDataToEspr. ESP-r then collects the new data with the call GetTrnsysData. Once the data exchange is complete, both programs call GetSystemConv to see if the whole system has converged; if it has then both programs may proceed to the next time-step and if not then another invocation of the current time-step is performed. The convergence criteria for ESP-r and TRNSYS solvers are set by the user, as are the tolerances in the Harmonizer.

The building-domain data that ESP-r passes to TRNSYS are calculated at the future time-row. They are the discrete values for the point in time at the end of the simulation time-step. As ESP-r solves the building and plant domains sequentially within the time-step, these are solved by ESP-r prior to the iteration between ESP-r's plant domain and TRNSYS for the given time-step.

The plant domain data passed by ESP-r are the discrete values it is solving for the future time-row. As there are multiple invocations per time-step, these values are evolving but converge to the future time-row once the Harmonizer concludes the system has converged for the time-step.

Because both ESP-r and TRNYSYS are loaded into the harmonizer as DLLS, they essentially become part of a single software program and can exchange data by passing variables through subroutines. This means they do not require any external means of data exchange such as shared memory, file I/O or sockets, but can pass variables directly. The required variables are passed via a shared derived data structure (DDS), which allows all the data that needs to be shared to be grouped (rather like an array made up of different data types) and facilitates the data dispatch. The order of the calls to the Harmonizer's data exchange functions is critical for ensuring synchronization, and is as follows:

1. PassDataToTrnsys waits for GetTrnsysData to complete (except in the first call).
2. GetEsprData waits for PassDataToTrnsys.
3. PassDataToEspr waits for GetEsprData.
4. GetTrnsysData waits for PassDataToEspr.

A number of flags are used to ensure that the ESP-r thread and the TRNSYS thread (which are otherwise operating independently) call these functions in the correct sequence.

PassDataToTrnsys is called by ESP-r to pass the DDS to the Harmonizer. Once all the data has been copied, the flag *EsprData.Ready* is set to true which allows GetEsprData to continue. GetEsprData is forced to wait for the data copying in PassDataToTrnsys to be complete before it can execute its commands. With the exception of the very first call, the bPassingDataToEspr flag is used to force PassDataToTrnsys to wait for data to be copied from TRNSYS to the Harmonizer and from the Harmonizer to ESP-r before ESP-r can pass its data into the Harmonizer.

GetEsprData is called by TRNSYS to retrieve the data passed by ESP-r and stored in the Harmonizer. The function will only return once the data has been retrieved which will only happen once *EsprData.Ready* is set true (i.e. once ESP-r has made that data available). After TRNSYS has received the data, the flag bPassingDataToTrnsys is set to allow PassDataToEspr to proceed.

PassDataToEspr is called by TRNSYS to pass data to the Harmonizer. As with PassDataToTrnsys, the DDS is copied before the *TrnsysData.Ready flag* is set to allow GetTrnsysData to continue. This ensures GetTrnsysData waits until the data copying in PassDataToEspr is complete. PassDataToEspr is in turn forced to wait (using the *bPassingDataToTrnsys* flag) for data to be copied from ESP-r to the Harmonizer and from the Harmonizer to TRNSYS before TRNSYS can pass its data into the Harmonizer.

GetTrnsysData is called by ESP-r to retrieve the data passed by TRNSYS and stored in the Harmonizer. The function only returns once the data has been retrieved, which only occurs once the flag *TrnsysData.Ready* is set to true (i.e. after TRNSYS has made that data available). Once ESP-r has received the data, the flag *bPassingDataToESP-r* is set to allow PassDataToTrnsys to proceed. The multi-threaded design of the cosimulator necessitates the use of multiple flags to force the coordination of the threads, otherwise data from the DDS may only be partially transferred when convergence is assessed.

GetSystemConv is called by ESP-r and TRNSYS to see if both programs have converged. Subsequent to ESP-r having satisfied convergence within its plant domain and TRNSYS having utilized its iterative solution method to converge a solution within the current time-step, the Harmonizer's convergence checker is invoked. It determines whether the two simulation programs have collectively converged by examining the state of the boundary conditions that are passed back and forth. If the Harmonizer concludes that the overall system has converged, each program is instructed to march forward in time to execute the simulation at the next time-step. If convergence of the boundary conditions is not attained, ESP-r and TRNSYS exchange boundary conditions via the Harmonizer and perform another invocation at the current time-step.

Under normal circumstances, the Harmonizer continues to invoke ESP-r's plant domain and TRNSYS until the values of the variables passed between the two simulators have stabilized. When there is no flow of water or air through the ports connecting the two simulators, ESP-r's plant domain and TRNSYS are essentially decoupled numerically. As such, to economize on computations the Harmonizer has been configured to minimize the invocations between the two solvers. As explained, ESP-r's plant domain and TRNSYS are invoked sequentially; on the first invocation at a given time-step, ESP-r's plant domain is passed data that TRNSYS solved at the previous time-step. As such, when flow through the ports becomes zero, a second invocation of ESP-r's plant domain is required to ensure an accurate simulation. Therefore, when the Harmonizer detects that the flow through the ports has ceased, it ensures there are two invocations of the solver before marching forward in time.

MODIFICATIONS TO ESP-r and TRNSYS

A number of additions were required in the ESP-r and TRNSYS source code to realize the coupling:

- ESP-r was modified such that data computed by a TRNSYS network could be imposed onto ESP-r's plant domain and such that flow and temperature data computed by ESP-r's plant domain could be imposed onto a TRNSYS network. This computed data comprises flow and temperature data for hydronic systems and includes additional moisture flow rates in the case of air-based systems.
- ESP-r was modified to enable temperature and humidity conditions determined by ESP-r's building domain to be imposed upon a TRNSYS network.
- ESP-r will be modified to enable heat transfer from a TRNSYS network to be imposed onto ESP-r building domain (i.e. internal gains).
- A new TRNSYS type (Type 130) was created to allow inputs and outputs to be passed to and from ESP-r modelling domains.
- A new category of TRNSYS type was created which is capable of preventing the TRNSYS engine from proceeding to the next time-step, which is required when the co-simulation fails to converge at a given time-step.

Two additional plant components have been added to the ESP-r code base to enable the interaction between ESP-r's plant domain and the Harmonizer. These new plant components have been generically termed TRNSYS coupling components (*TCC*), of which there are two types; the hydronic coupling component (*HCC*); and air-based coupling component (*ACC*). The ESP-r user is required to specify which real components in the plant network supply or receive a fluid stream (liquid or air). Coupling components can be either *sending* or *receiving*: a sending component indicates that ESP-r will send data to the Harmonizer; whereas a receiving component indicates that ESP-r will receive data from the Harmonizer.

A new subroutine, TCC_communication, handles all the data exchange between ESP-r and the Harmonizer.

It has been designed such that TRNSYS types can be contained within ESP-r building zones and thermal losses (e.g. from the skin of a thermal storage tank) can be injected into an ESP-r building zone. All the zone air-point temperatures are passed from ESP-r to the Harmonizer and thereby made available to Type 130. In TRNSYS the user needs to specify the containment of each type by mapping these zone air-point temperatures to the type's input. Similarly, the user also specifies the parasitic heat losses from types by mapping these outputs to Type 130, which will then be communicated via the Harmonizer to the ESP-r. In ESP-r these heat transfer rates will be injected as casual gains into the appropriate zones.

Type 130 has been created to receive data from each of ESP-r's sending HCCs and ACCs and dispatch data to each of ESP-r's receiving HCCs and ACCs. The standard TRNSYS procedure for connecting components is used to map ESP-r outputs (representing real components in the plant network) to Type 130 inputs and Type 130 outputs to the inputs of the other TRNSYS types. Likewise, the outputs of TRNSYS types are mapped to Type 130 inputs which are mapped to ESP-r inputs.

The main distinction between Type 130 and types representing real components is that Type 130 can prevent the TRNSYS engine from proceeding to the next time-step when TRNSYS has converged but there is no overall convergence for the system (TRNSYS and ESP-r). To implement this, a new category of type, the data exchanger type, was created. The data exchanger type is called directly after internal convergence to determine whether the simulation can proceed to the next time-step. Thus, during a co-simulation, Type 130 calls GetSystemConv in the Harmonizer and if overall convergence is reached TRNSYS calls after convergence types and then proceeds to the next time-step. If not, new iterations of the same time-step are made, after getting renewed data from ESP-r with a call to GetEsprData.

VERIFICATION

Three tests were used to verify the results obtained through the run time coupling of ESP-r and TRNSYS. The first test was to ensure data was not corrupted in its passage between the three programs. The two other tests were designed to ensure the results obtained in simulating first hydronic based and secondly air based systems were credible.

As previously described a DDS is used to transport data between ESP-r, TRNSYS, and the Harmonizer. TRNSYS uses double precision real numbers by default, whereas and ESP-r uses a mix of single precision, double precision, and integer variables. Therefore, data passed between them needs to be cast to its appropriate type in the Harmonizer. To prove the integrity of the data communications between ESP-r and TRNSYS via the Harmonizer the following test was used. An HCC sending component in ESP-r passed time-varying flow rates and temperatures to TRNSYS (via the Harmonizer) for every iteration through ESP-r's plant domain solver. The Harmonizer passed this onto Type 130 and TRNSYS incremented the value by two. This result was subsequently passed back to the Harmonizer and then to an HCC receiving component. Both the sent and returned values were made them available in ESP-r's reporting facility. The resulting .csv file was examined to ensure that the data were correctly passed through all the communications. This test demonstrated that transporting data between ESP-r and the Harmonizer and between TRNSYS and the Harmonizer was successful.

HYDRONIC SYSTEM TEST

A hydronic system test was devised to test a co-simulation in which ESP-r treats the building and the HVAC system's terminal device and in which TRNSYS treats the HVAC system's heating and circulation devices. This test case was based upon BESTEST case 600 (Judkoff and Neymark, 1995) and required a minimal plant network consisting of a sending and receiving HCC and a radiator in ESP-r. The HVAC system in TRNSYS comprised: a boiler and pump activated by a controller trying to maintain a constant room temperature. The test was further subdivided to investigate different control strategies using: an on/off controller to control the flow rate of a constant temperature stream of water (B1); an on/off controller to control the temperature of a constant flow rate of water (B2); and a PID controller used to control the flow rate of water at a constant temperature (B4). The implementation of this system is detailed in the schematic shown in Figure 2.

The results from these test cases are presented in the graphs in Figures 3, 4, and 5. Depicted in these graphs are the zone air temperature (sensed in ESP-r) and the controller set points used to control the plant network (in TRNSYS). In each case the air temperature can be seen to rise after supplying heat to the zone has ceased which reflects the thermal capacity of the water in the radiator. When the temperature reaches 20°C in the room, either the pump or boiler is turned off, but the residual heat in the radiator continues to heat the air of the zone for a few minutes. Figure 3 illustrates the case where the zone air temperature is regulated to 20°C by an on/off controller, controlling the pump.

The graph demonstrates that TRNSYS can sense the zone air-point in ESP-r, and correctly actuate the pump to respond to the heating demand and that ESP-r's air-point temperature responds to the transient behavior of the radiator. Figure 4 illustrates the case where the heater in TRNSYS is controlled rather than the flow rate. Again the graph demonstrates that TRNSYS can sense the zone air-point in ESP-r, and correctly controls the heater in response to the heating demand. The

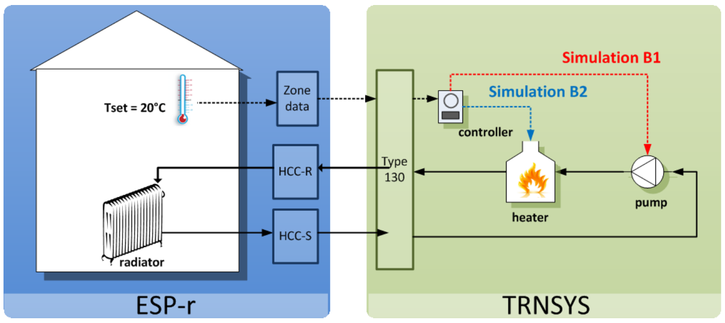


Figure 2: Co-simulation of a hydronic system

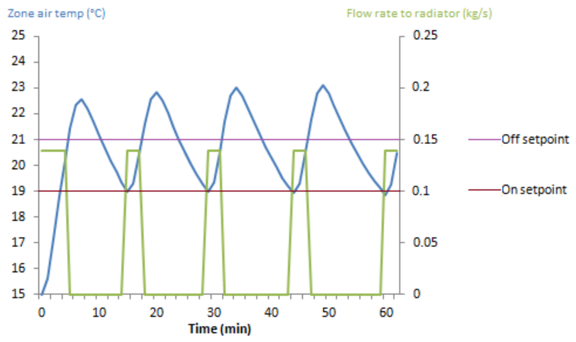


Figure 3: Results from co-simulation B1

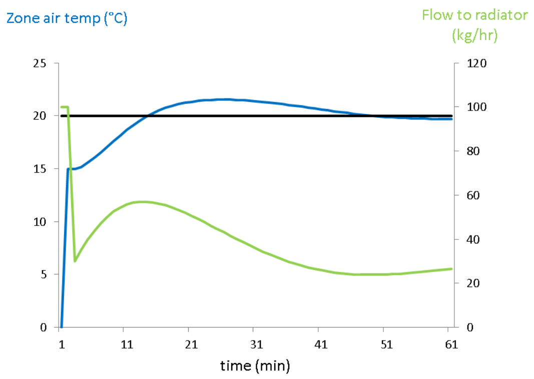


Figure 5: Results from co-simulation B4

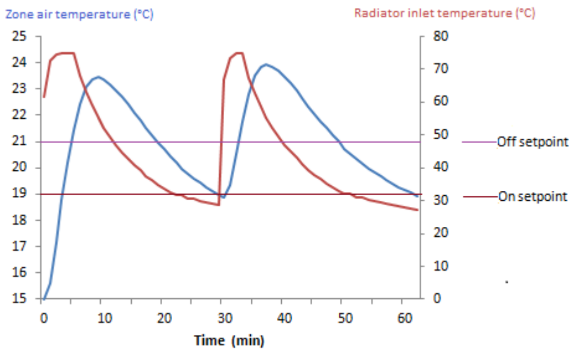
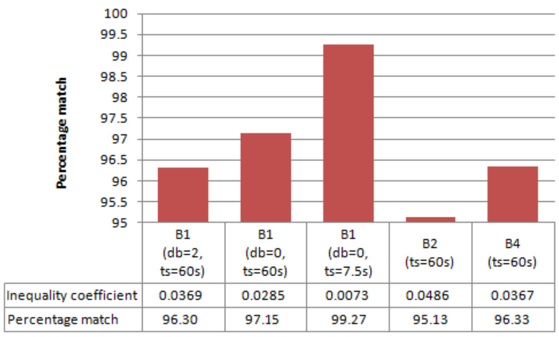


Figure 4: Results from co-simulation B2



	B1 (db=2, ts=60s)	B1 (db=0, ts=60s)	B1 (db=0, ts=7.5s)	B2 (ts=60s)	B4 (ts=60s)
Inequality coefficient	0.0369	0.0285	0.0073	0.0486	0.0367
Percentage match	96.30	97.15	99.27	95.13	96.33

Figure 6: Inequality coefficients and percentage match figures for different HCC test cases

heater is either off or maintaining a constant supply temperature of 75 °C ESP-r's air-point temperature responds to the constant flow of heated water with less cycling than in the case with the on/off controller controlling the pump. The effect of the PID controller can be seen in Figure 5 where the flow to the radiator can be seen to vary. These graphs attest the communication between the two programs is working correctly.

To verify the accuracy of the results obtained, the zone air temperature predicted was compared with that predicted by ESP-r using ideal control to control the air temperature to 20 °C. The Inequality Coefficient (Theil, 1961), describes the inequality in a time-series due to the mean, variance, and co-variance as described by equation 1. The resultant coefficient ranges in values between zero and one, with zero indicating a perfect match at every time-step and one denoting no match. This coefficient is converted to a *percentage match* by subtracting it from 1 and multiplying the result by 100. All the test presented in this paper are based on 1 minute time steps with convergence tolerance of 0.01 °C for temperatures and 0.1% for flowrates.

$$U = \frac{\sqrt{\frac{1}{n}\sum(X_t - Y_t)^2}}{\sqrt{\frac{1}{n}\sum X_t{}^2 + \frac{1}{n}\sum Y_t{}^2}} \qquad (1)$$

Where U is the inequality coefficient, X_t and Y_t are two time series, and n is the number of time-steps.

Figure 6 shows the inequality coefficients and the corresponding percentage matches obtained for a number of variations of the hydronic systems test. Test B1 utilizes on/off control of the pump with a dead-band of 2 and with a dead-band of 0, B2 controls the heater using on/off control and B4 controls the heater using PID control. The results coefficients obtained demonstrate that the co-simulation are in good agreement with the ideal control scenario, particularly in the case where the dead-band (db) of the on/off controller (in test B1) in TRNSYS is set to zero and the time-step (ts) is reduced to 7.5seconds.

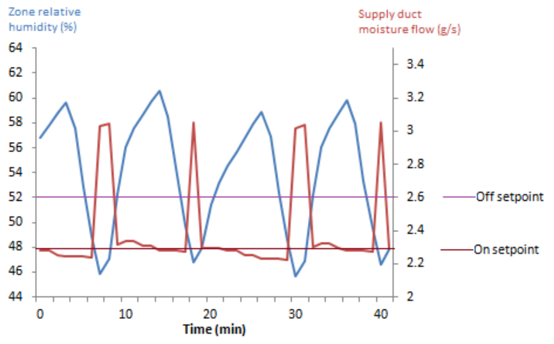


Figure 8: Results from air-based co-simulation

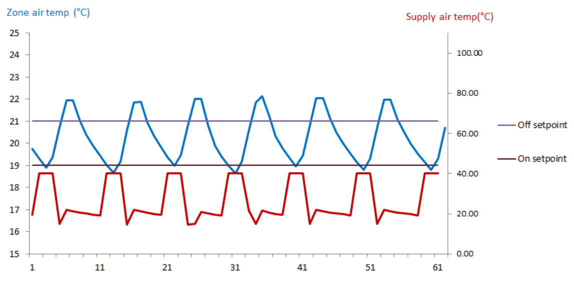


Figure 9: Results for humidity and moisture flow from air-based co-simulation

AIR-BASED SYSTEM TEST

The air-based system test was devised to test the ACC component and was similar to the previous test in that ESP-r treats the building and TRNSYS the HVAC system's heating and circulation devices. Again ESP-r required a minimal plant network consisting of a receiving and sending ACC and corresponding supply and return air ducts to the zone. The TRNSYS HVAC system comprised a heating coil, a fan, and a controller, as well as a humidifier with an on/off controller to maintain a relative humidity of 50%. Zone air temperature is regulated to 20°C by on/off control on the fan adding heated air (at 40°C) This test was used to test the behavior of the ACC components, but also to ensure zone air humidity and moisture flow data was handled correctly.

A set of results from the air-based test is illustrated in the graphs in Figures 8 and 9. The graph demonstrates that TRNSYS can sense the zone air-point in ESP-r, and correctly actuate the heating coil to respond to the heating demand and that ESP-r's air-point temperature responds to the temperature of the air steam from the supply duct. The plant network uses temperatures calculated in the building domain from the previous timestep which accounts for why the system supplies heat after the zone air has exceeded the off setpoint Figure 9 illustrates that TRNSYS can sense the zone humidity in ESP-r and correctly humidify the air stream when the zone humidity falls below 50%.

DEMONSTRATION

To demonstrate potential co-simulation applications, a house serviced by a DHW/space heating solar combi-system was simulated. The system used is described in

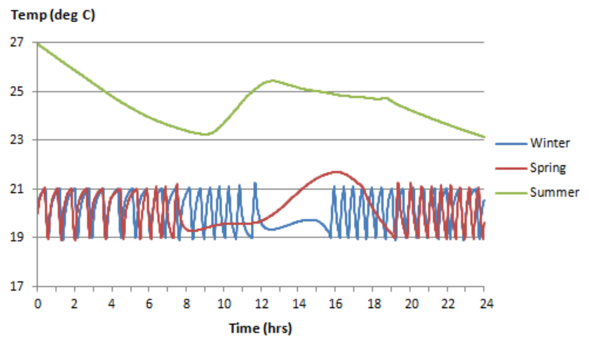


Figure 11: Predicted zone air temperatures heated by combi-system for different seasonal days

Figure 10, and demonstrates that the simulator could be used to support the analysis of NZEH.

Some results from this simulation are shown in Figures 11 and 12. The pump providing supply hot water to a radiator is commanded by an on/off controller to heat the zone to 20°C. The deadband is 1°C. Figure 11 illustrates the zone air temperatures predicted for a period of 24 hours in winter, spring and summer. The greatest degree of heating control is seen in the winter where heating is not required over a short period in the middle of the day (due to solar gains). The transitional day has a much longer period without heating and the solar gain is more significant than illustrated during the winter day. The summer day does not require any heating, the temperature curve reflects the free-float behaviour of the building as there is no cooling system in this simulation. In fact a period of night cooling can be seen to occur after the previous days overheating, before the temperatures in the zone rise again with the warm ambient day time temperatures characteristic of summer.

The graph in Figure 12 is taken from the transitional period and depicts the variations of the radiator water inlet temperature. The temperature of the water is seen to be higher after the solar heating period, as would be expected. It also shows that the pump is predominantly on during the early part of the day, off for the middle portion, and cycling on and off frequently in the late afternoon. During the early part of the day the supply temperature is lower and takes longer to heat the zone to the set point temperature, resulting in a predominantly on pump. When the solar gain is significant and no heating is required and the pump is off. With higher water supply temperatures from the solar combi-system, the pump cycles on and off frequently as the set point temperature in the zone is attained more rapidly than it was in the morning. This again demonstrates that the zone air temperature is being sensed in TRNSYS and the control on the pump is functioning correctly. However this test is being presented to demonstrate a potential application for the co-simulation and not for validation purposes.

COMPUTATIONAL EFFICIENCY

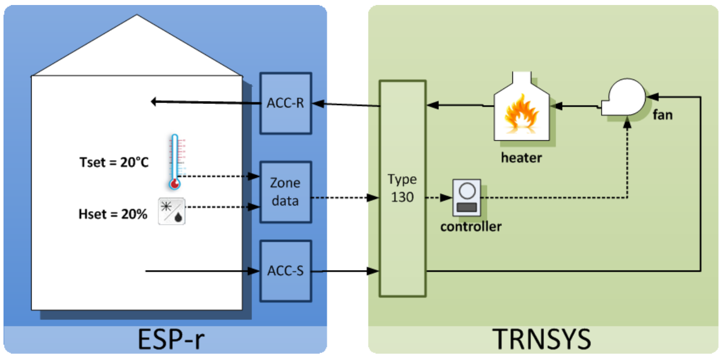


Figure 7: Co-simulation of an air-based system

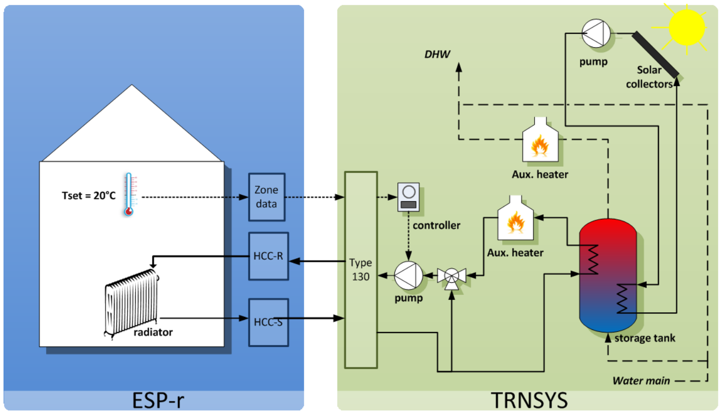


Figure 10: Co-simulation of a solar-combi system

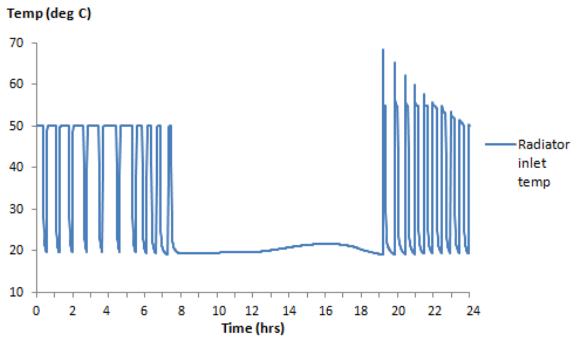


Figure 12: One day simulation results for combi-system

Time trials were conducted using the HCC and ACC tests. Both models were simulated for a period of 1 year using 1 minute time steps with: 1) ESP-r only; 2) TRNSYS only; and 3) using the co-simulator.

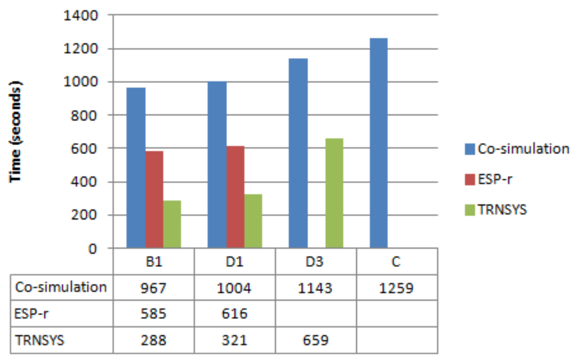
The time command runs a specified program command and reports on timing statistics about this program run. The times reported include: the *user* time, the actual CPU time spent running the process; the *real* time, the total real time for the process to run (including times where other processes were using the CPU); and the *sys* time, the time spent waiting for operating system calls.

The statistics reported in Figure 13 are of the elapsed real time. This figure is used because the time spent in the the actual launching executable is minimal (of the order of 0.015s) and the time spent waiting for operating system calls is also comparatively so small as to be insignificant. The times reported are never exactly replicated and will depend on other processes running on the machine during execution time, however variations are of the order of +/- 15 seconds. These results give an indication of the relative differences in time taken by the different simulation environments to simulate the different cases. Test B1 is the hydronic system test with an on/off controller on the pump. Test D1 is the air-based system test without humidity control and D3 is the same system as D1 with humidity control. Test C is the house serviced by a DHW/space heating solar combi-system.

It is interesting to note that the co-simulation takes 30% longer than ESP-r to execute the simple HCC and ACC tests and 65% longer than TRNSYS. This figure reduces to 30% longer than TRNSYS where humidity control is incorporated into the simulation. The result given for test C is presented to show how the simulation time of the Harmonizer increases with system complexity.

CONCLUDING REMARKS

This paper has described the design of the co-simulation environment using ESP-r to process the building domain, and potentially a portion of the mechanical energy systems; and TRNSYS to solve all or a portion of the mechanical energy systems. ESP-r and TRNSYS simulations are run as threads and their data exchange is managed by a middle-ware, known as the Harmonizer. The Harmonizer is also responsible for assessing



	B1	D1	D3	C
Co-simulation	967	1004	1143	1259
ESP-r	585	616		
TRNSYS	288	321	659	

Figure 13: Simulation times for different test cases

convergence and keeping the overall simulation time. The functionality and implementation of the Harmonizer have been described and the changes required in both ESP-r and TRNSYS to enable this co-simulation have been detailed.

The tests used to verify co-simulations involving both hydronic and air-based systems were described and their results reported. It has been shown that the co-simulations behave as one would expect and compare well with the 'ideal control' case. To further verify the co-simulation, it will be contrasted with an internal coupling method known as the "TRNSYS type wrapper" (Wang and Beausoleil-Morrison, 2009).

To exhibit the potential applications for the co-simulator, an example case was presented in which ESP-r treats the building domain and TRNSYS treats a solar combi-heating system. Further testing is required to verify cases in which ESP-r and TRNSYS collaborate in modelling the HVAC system.

The simulation times of the co-simulation were compared with those of mono-simulations. Further work in examining the issues around convergence stability and frequency of data transfer during time-step iterations is planned and may result in different co-simulation times.

The co-simulation capabilities described in this paper will be made available in the release versions of both ESP-r and TRNSYS in the near future. Future developments may include:

- Enabling heat transfer from a TRNSYS network to be imposed onto ESP-r building domain (i.e. internal gains);
- An extension to ESP-r's electrical domain to complete the coupling between the two programs;
- Provision of support for use of the co-simulation environment for users without a TRNSYS license. (The current co-simulator is only applicable for users who have both ESP-r and TRNSYS licenses)

REFERENCES

Beausoleil-Morrison, I., Macdonald, F., Kummert, M., McDowell, T., Jost, R., and Ferguson, A. (2011). The design of an ESP-r and TRNSYS co-simulator. In *Proc. Building Simulation 2011*, pages 2333–2340, Sydney, Australia.

Djunaedy, E. (2005). *External coupling between building energy simulation and computational fluid dynamics*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven Netherlands.

Dorer, V. and Weber, A. (1999). Air, contaminant and heat transport models: integration and application. *Energy and Buildings*, 30(1):97–104.

Huang, J., Winkelmann, F., and Buhl, F. (1999). Linking the COMIS multi-zone airflow model with the EnergyPlus building simulation program. In *Proc. Building Simulation '99*, pages 1065–1070, Kyoto Japan.

Janak, M. (1997). Coupling building energy and lighting simulation. In *Proc. Building Simulation '97*, volume II, pages 313–319.

Judkoff, R. and Neymark, J. (1995). International energy agency building energy simulation test (bestest) and diagnostic method.

Theil, H. (1961). Economic forecasts and policy. *Contributions to Economic Analysis*, XV(2).

Trčka, M., Hensen, J. L., and Wetter, M. (2009). Co-simulation of innovative integrated HVAC systems in buildings. *Journal of Building Performance Simulation*, 2(3):209–230.

Wang, W. and Beausoleil-Morrison, I. (2009). Integrated simulation through the source-code coupling of component models from a modular simulation environment into a comprehensive building performance simulation tool. *Journal of Building Performance Simulation*, 2(2):115–126.

Wetter, M. (2010). Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*. DOI: 10.1080/19401493.2010.518631.