

ASHRAE Research Project Report

RP-1052

Development of an Analytical Verification Test Suite for Whole Buildings Energy Simulation Programs- Building Fabric

Approval: April 2001

Contractor: Oklahoma State University

Principal Investigator: Jeffrey D. Spitler
Authors: Simon J. Rees, Xiao Dongyi

Author Affiliations, Oklahoma State University

Sponsoring Committee: TC 4.7, Energy Calculations

Co-Sponsoring Committee: N/A

Co-Sponsoring Organizations: N/A



Shaping Tomorrow's
Built Environment Today



**DEVELOPMENT OF AN ANALYTICAL VERIFICATION
TEST SUITE FOR WHOLE BUILDING ENERGY
SIMULATION PROGRAMS – BUILDING FABRIC**

ASHRAE 1052-RP

FINAL REPORT

April 2001

Jeffrey D. Spitler
Simon J. Rees
Xiao Dongyi

Oklahoma State University
School of Mechanical and Aerospace Engineering

**DEVELOPMENT OF AN ANALYTICAL VERIFICATION
TEST SUITE FOR WHOLE BUILDING ENERGY
SIMULATION PROGRAMS – BUILDING FABRIC**

ASHRAE 1052-RP

FINAL REPORT

April 2001

Jeffrey D. Spitler
Simon J. Rees
Xiao Dongyi

Oklahoma State University
School of Mechanical and Aerospace Engineering

Contents

Executive Summary	1
Introduction to the Test Documentation	5
The Test Documentation	
Test SSConv	12
Test SSCond	17
Test TC1	22
Test TC2	30
Test TC3	43
Test ExtSolRad	53
Test SolRadGlazing	65
Test SolRadShade	78
Test WinReveal	92
Test IntSolarDist	103
Test Infiltration-1	110
Test Infiltration-2	114
Test ExtLWRad	120
Test IntLWRad	126
Test IntHeatGain	142
Test GrdCouple	150
Introduction to the Test Suite Software	156
Verification of the Analytical Tests	159
Internal Testing of the Test Suite	161
External Testing of the Test Suite	214
Source Code Listings	227

Executive Summary

Introduction

Building energy analysis programs are, by the nature of the processes they seek to model, complex and require detailed input data involving many parameters. Demonstrating the validity of results, and diagnosing deficiencies in the calculation method or its computer implementation can be correspondingly difficult. The need for some form of validation of building energy analysis codes has been long recognized. A number of attempts have been made over the last two decades to identify suitable validation procedures by a number of organizations (e.g. NREL¹, NIST², BRE³, BEPAC⁴, IEA⁵) but no standard procedures have been universally accepted despite the proliferation and increased use of such programs.

Three types of validation methods can be identified (Judkoff, 1988, Bloomfield 1999) in which results from one code are compared with results from other sources. The sources of comparison data can come from:

- analytical models,
- inter-model comparisons
- empirical test results.

Of these approaches to validation the use of analytical tests are most abstracted from the full complexities of real building simulation problems but offer the most certain form of reference or ‘truth’ model with which comparisons can be made. Being slightly abstract in nature they may be of least interest to the end user of simulation codes but are arguably the most useful to the code developer in that they offer the clearest path to diagnosis of specific problems with the algorithms or their implementation.

It is the objective of this project to develop a suite of analytical tests to be used in testing the building fabric aspects of building energy analysis programs. Analytical solutions, in order to be tractable, have to be limited in scope and make a number of simplifying assumptions. Their purpose is to help in *verification* of the proper functioning of certain models and sub-models, rather than *validation* of the whole test program’s ability to model real situations. We make the distinction between code *verification* and *validation* following the definitions given by Boehm (1981) and Blotter (1990). They described verification as “solving the equations right” and validation as “solving the right equations”. Furthermore, we use the term *analytical* to mean a mathematical model of

¹ National Renewable Energy Laboratory, formerly Solar Energy Research Institute, Golden, CO

² National Institute for Standards and Technology, formerly the National Bureau of Standards, Gaithersburg, Maryland

³ Building Research Establishment, Watford, United Kingdom

⁴ Building Energy Performance Analysis Club, United Kingdom

⁵ International Energy Agency

reality that has an analytically determinable solution for a given set of parameters and boundary conditions.

Scope

In order to achieve this objective, the literature was reviewed for existing solutions; a test philosophy was formulated; a set of analytical tests was developed, documented, and implemented as Fortran subroutines; a user interface was written to allow the generation of analytical solutions and weather files for a range of parameters. Then, the test suite was “tested” in-house by comparing BLAST to the analytical solutions. Finally, the test suite was tested by a 3rd party, comparing Energy+ to the analytical solutions.

Test Philosophy

The tests developed for ASHRAE 1052-RP are designed so that, whenever possible, they may be useful to and used by program users, not only by developers. To this end, the test suite has the following features:

- When possible, the tests require only simple output typically available to program users. These include annual and monthly heating and cooling loads; annual and monthly peak heating and cooling loads. However, for many tests, hourly output is also required.
- Besides the unique weather files developed for this project, the tests, for the most part, require only simple input typically available to program users. This has been done by allowing as much flexibility as possible in the analytical solution. As an example, consider the exterior and interior convection coefficients. While it would be most convenient to have the user fix the convection coefficients, this may not be feasible. Therefore, where possible, the analytical solutions allow for temperature dependent exterior convection coefficients, and temperature dependent interior convection coefficients. (One partial exception to this is that the solar radiation tests require the presence of beam radiation only. Although the weather file formats allow this, the weather processor utilized by BLAST corrects this apparent deficiency. Therefore, in order to test BLAST against the analytical test suite, we had to modify and recompile the weather processor.)
- Test as few phenomena as possible with each test to expedite identification of specific problems.
- The tests are ordered to expedite identification of specific problems.

Results and Report Organization

The following analytical tests have been developed:

- Steady-state Convection

- Steady-state Conduction
- Exterior Radiation
- Exterior Solar Radiation
- Interior Radiation
- Transient conduction/ Step/Adiabatic
- Transient conduction/ Step/Fixed Temperature
- Transient conduction/ Sinusoidal Temperature
- Infiltration 1
- Infiltration 2
- Internal Convective Gains
- Internal Radiative Gains
- Ground Coupling
- Solar Transmission through Fenestration
- Internal Solar Distribution
- External Horizontal Shading
- External Vertical Shading
- Window Reveal

They are described in the section “The Test Documentation”. Each test description is self-contained, and gives the objective, zone description, test parameters, test output, test results, analytical solution, and references for the test. Each test gives both a general analytical solution and a specific analytical solution for a specific set of test parameters.

The general analytical solutions are implemented in Fortran, and a user interface written in Visual Basic allows users to conveniently generate both analytical solutions and a weather files for any of the tests with any set of parameters. This software is described below in the section “Introduction to the Test Suite Software”.

Our in-house test of the BLAST program against the test suite is described in the section “Internal Testing of the Test Suite”. Despite the fact that BLAST has been in existence for a number of years, several problems were found using the test suite. These include:

- BLAST calculates the density of air for infiltration calculations using the ideal gas law. The value of R is about 11% off. This results in an error in the zone load of about 11% when calculating the load due to infiltration.
- The zone load resulting from a step change in outside temperature was significantly in error when using the default convergence coefficients in BLAST. As far as we know, the convergence coefficient syntax has only ever been described in an unpublished report. Therefore, most users would not be capable of adjusting the coefficients.
- The syntax for describing vertical wings on windows is incorrectly documented in the manual. Therefore, two of the key dimensions are reversed if the wing is specified as described in the manual.

It might also be noted that the detailed infiltration test and the ground-coupling test cannot be replicated by BLAST because it simply doesn’t have the necessary internal algorithms

Third party testing of the test documentation and test suite software was undertaken by GARD Analytics as part of their effort in testing the EnergyPlus program. A number of detailed comments were provided relating to both the documentation and the software. The comments and our response is noted in section 6 of this report.

Conclusions

An analytical test suite for validation of the building fabric modeling in building energy analysis programs has been developed. In our in-house testing, it was very useful in uncovering some problems with a well-known building energy analysis program that has been in the public domain for more than twenty years. The documentation and software was tested by a third party, and refinements to the documentation suggested by the reviewers have been incorporated.

References

- Bloomfield, D. (1999). *An Overview of Validation Methods for Energy and Environmental Software*. ASHRAE Transactions. SE-99-6-1.
- Blottner, F.G. (1990). *Accurate Navier-Stokes Results for the Hypersonic Flow over a Spherical Nosedip*, AIAA Journal of Spacecraft and Rockets, Vol. 27, No. 2, March-April 1990, pp. 113-122.
- Boehm, B.W. (1981). *Software Engineering Economics*, Prentice-Hall, 1981.
- Judkoff, R. (1988). *Validation of Building Energy Analysis Simulation Programs at the Solar Energy Research Institute*. Energy and Buildings, Vol. 10, No. 3, p. 235. Lausanne, Switzerland: Elsevier Sequoia

Introduction to the Test Documentation

The Purpose of the Tests

A series of tests have been developed that are designed to help verify the ability of whole building energy simulation programs to model various aspects of heat transfer through the building fabric. These tests are each based on ‘analytical’ solutions. They have been funded by the American Society of Heating Refrigeration and Air-conditioning Engineers (ASHRAE) under research contract 1052-RP to Oklahoma State University. The tests are intended to be used in support of ASHRAE Standard 140 ‘Standard Method of Test for Building Energy Analysis Software’. These tests are for testing models relating to heat transfer through the building fabric, and not primary or secondary HVAC systems.

Use of analytical tests is only one of several methods that can be used to verify and validate programs for whole building energy simulation. The analytical solutions, in order to be tractable, have to be limited in scope and make a number of simplifying assumptions. As such, their purpose is to help in *verification* of the proper functioning of certain models and sub-models, rather than *validation* of the whole test program’s ability to model real situations. We make the distinction between the code *verification* and *validation* following the definitions given by Boehm (1981) and Blotter (1990). They described verification as “solving the equations right” and validation as “solving the right equations”. We use the term *analytical* to mean a mathematical model of reality that has an analytically determinable solution for a given set of parameters and boundary conditions.

The objective in each test is usually to test the ability of a building energy analysis program to model a particular heat transfer phenomena. This is done by comparing the test program output with the analytical solution for a special test zone. The data to be compared may be a single load or heat flux, or hourly loads over one or more days of output. In order to make each test specific and help diagnose problems it is necessary to minimize the number of heat transfer paths (and hence number of models involved). This requires the use of test zones that are rather different in their construction and specification than normal building zones. Considerable familiarity on the part of the tester with the operation and data requirements of the program to be tested is therefore required.

Method of Testing

The procedure by which each test is intended to be carried out can be summarized as follows:

1. Choose the test parameters and enter them into the test suite software
2. Run the test suite software and save the results and weather file produced
3. Set up the input for the program to be tested to comply with the test specification

4. Convert the weather file to the necessary format for input to the test program if required
5. Run the test program for the test zone with the test weather file
6. Compare and analyze the program results and analytical test results.

If it is convenient to run the test suite software the tests may be run with a variety of parameter values, otherwise the tests may be run using the parameter values and results given in the documentation.

Organization of the tests

The test suite consists of sixteen individual tests. The tests can be organized into groups relating to particular heat transfer phenomena as follows:

Group 1	-	Convection and Conduction
Group 2	-	Solar Gains and Shading
Group 3	-	Infiltration
Group 4	-	Long Wave Radiation
Group 5	-	Miscellaneous.

There is also some inter-relationship between some of the tests. In particular convection heat transfer is involved in nearly all the tests. It is generally not possible, for example, to remove convective resistances from a wall or window conductance, due to the organization of the input data structures of most programs. This makes it important that the steady state convection test is carried out first, and its results analyzed prior to the other tests.

The complete list of test is given in the Table Intro-1. Each test is given an abbreviation for the purposes of referencing tables, figures and equations.

Organization of the documentation

Each of the tests has its own section in the documentation that is designed to be self-contained. To this end the documentation for each test has the following sections:

- Objective
- Analytical Model
- Zone Description
- Test Parameters
- Test Output
- Test Results
- Analytical Solution
- References

Table Intro-1
Organization of the test suite

Test Group	Test Title	Abbreviation
Group 1	Steady-state Convection	SSConv
	Steady-state Conduction	SSCond
	Transient Conduction – Adiabatic Wall	TC1
	Transient Conduction – Step Response	TC2
	Transient Conduction – Sinusoidal Driving Temperature and Multi-layer Wall	TC3
Group 2	Exterior Solar Radiation – Opaque Surfaces	ExtSolRad
	Solar Radiation – Glazed Surfaces	SolRadGlazing
	Solar Radiation – Window Shading	SolRadShade
	Solar Radiation – Window Reveal Shading	WinReveal
	Solar Radiation – Internal Solar Distribution	IntSolarDist
Group 3	Infiltration – Fixed Infiltration Rate	Infiltration 1
	Infiltration – Stack Effect	Infiltration 2
Group 4	Interior Long Wave Radiation	IntRad
	External Long Wave Radiation	ExtLWRad
Group 5	Internal Heat Gains – Convective and Radiant	IntHeatGain
	Ground Coupling – Slab on Ground Floor	GrdCoup

The function of each section of the documentation can be summarized as follows.

Objective: This sets out the aim of the particular test in terms of which heat transfer path or sub model of the energy simulation program is to be tested. It also summarizes the basis for the test and any assumptions made.

Analytical Model: This is provided to summarize the basis of the analytical model and its assumptions. It also shows how the analytical model is being applied to represent the behavior of the test zone.

Zone Description: This sets out the definition of the test zone to be used. Particular geometric, construction and surface properties may be specified. Any heat transfer paths to be eliminated may also be specified.

Test Parameters: The particular parameters relevant to the test are given in this section of the documentation for each test. In addition the parameter values used to get the results published in the Results section are tabulated. The input screen from the test suite software is illustrated in this section.

Test Results: Results of applying the analytical solution for one or more sets of test parameters are given in tabular and/or graphical form. These have been generated using the test suite software.

Analytical Solution: The derivation of the analytical solution that forms the basis of the test is given, along with any assumptions and limitations.

References: References are given at the end of the documentation for each test. These are the references relating to either the analytical solution or parameter values that have been cited in the documentation for that test alone.

The Test Zone

The test series is based around a cube shaped of 3x3x3m internal size. Which surfaces are exposed or adiabatic vary from test to test. Only in two tests, IntRad and IntSolarDist, is the aspect ratio required to be varied. In some of the tests it is assumed that the zone load is instantaneously related to an internal surface temperature (e.g. tests TC1-3). In other words, the zone air is assumed to have no thermal mass. This assumption is often made in whole building energy simulation programs. If the zone is in fact modeled with thermal mass, it may be necessary for the user of the test to modify the zone geometry so that the air mass is minimized – i.e. the depth is made very small.

Convection Coefficients

An important issue is related to the existence of, or feasibility of “analytical” solutions for many phenomena. In the case of convection coefficients there is no one universally acceptable “right” coefficient or correlation. In the results published as part of the test documentation we have tried to use correlations that have been suggested in recent literature. However, by using the test suite software the user is able to specify the convection correlation and generate the results of a specific analytical solution. Most convection correlations used for exterior and interior building heat transfer should be reducible to the form (e.g. that given by Yazdanian and Klems 1994 at zero air speed),

$$h = A + C(T_s - T_\infty)^n \quad (\text{Intro-1})$$

where,

h = convection coefficient, W/(m²-K) or Btu/(hr-ft²-F)

A = constant, W/(m²-K) or Btu/(hr-ft²-F)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C or F

T_∞ = air temperature, C or F.

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations. For example, if the code uses the correlation given by Yazdanian and Klems (1994), which is,

$$h = \sqrt{\left[C_t (T_s - T_\infty)^{1/3} \right]^2 + \left[a V_o^b \right]^2} \quad (\text{Intro-2})$$

where $C_t = 0.84$ for SI units and constants a and b depend on wind direction. It is suggested that the wind velocity is zero so that the correlation reduces to

$$h = 0.84 (T_s - T_\infty)^{1/3} \quad (\text{Intro-3})$$

In this case A , C and n can be specified as 0.0, 0.84 and 0.333 respectively. For the analytical tests such as those involving transient conduction it is not practical (in the derivation the solution) to use a non-linear convection correlation and a constant convection coefficient must be set. This applies in the case of tests TC1, TC2, TC3 and IntHeatGain. Also, if the program to be tested uses only fixed coefficients then when it is necessary to specify A , C and n for the software A is set to the fixed value of h ; C and n are set to zero.

It is important that some effort is made to ensure that the convection correlation data input into the analytical solution correspond to the correlation known to be used in the test program. This is especially important as convection heat transfer is present in all of the tests. If the convection coefficients and their correlations aren't known from program documentation it may be necessary to run the analytical test and generate the weather file, then run the test program and find convection coefficients from the program output, and finally rerun the analytical test using the convection coefficients stated in the test program output.

If the test program does not allow specification or selection of fixed heat transfer coefficients then some difficulty arises in using the tests involving transient conduction mentioned above. The best approach is probably to find the average convection coefficients used by the test program for a particular set of test parameters and re-running the analytical calculation with these values. In this situation several iterations may be necessary before arriving at consistent results. A very close match between the analytical solution and the test program results can not be expected in these circumstances.

Surface Properties

In a number of the tests it is necessary to eliminate long-wave and/or solar radiation from the interior and/or exterior surface of the zone. This requires careful specification of the zone surface properties in the test program input data. It may be possible to achieve these goals in more than one way, depending on the exact input data requirements. A common approach would be to set solar absorptivity and long-wave emissivity for the relevant surfaces to zero. However, depending on the program care should be taken to avoid numerical errors (e.g. divide by zero errors) when setting some of these values to 0.0. It may be necessary to set a very small number (e.g. 0.00001).

Thermal Mass

Several of the tests rely on the fact that some or all of the zone fabric has zero thermal mass i.e. the product of the density and specific heat is zero. This is done so that certain heat gains can be assumed to be instantaneous. If program input does not allow specification of purely resistive walls it is important that the density and specific heat be set to zero, or as low as possible if for numerical reasons zero can not actually be specified.

Some of the tests also require that certain elements of the zone fabric be *heavyweight*. This means that the density and heat capacity of the fabric materials must be significant. This can be achieved using building material thermal properties within the normal range. For example, by using the properties of a single layer of dense concrete with density 2000 kg/m³, heat capacity 800 W/kg.K and thickness 0.15m.

Weather files

Whole building energy simulation programs generally require weather data input to run a simulation. To this end the test suite software will generate weather files corresponding to the test parameter values for each test. Weather parameters that do not require any variation as part of the test are kept fixed at default values (e.g. zero wind speed, fixed standard pressure) in the weather files. These files can be generated in either TMY2, WYEC2, or IWECC formats. Any conversion from these formats to another format required by the program under test is left to the user. It should be noted that discrepancies could arise as a result of weather file conversion. If it is possible the converted file should be checked against the original parameter values. Such discrepancies may arise due to error checking and automatic data repair during the conversion process. For example some conversion programs may attempt to repair solar data that appears as zeros in the input file, as is necessary in some of the tests.

Solar Data

Arriving at tests involving solar algorithms and data that can be considered as truly 'analytical' is problematic. This is mainly because of the empirical nature of all models of diffuse solar radiation. In view of this all the solar radiation tests in this series involve only direct beam solar radiation. In each of the solar radiation tests it is therefore necessary that diffuse radiation and radiation reflected from surroundings are eliminated. The weather files accordingly only contain data for direct normal radiation at ground level. Zeros are inserted for extraterrestrial and diffuse radiation.

Each of the solar radiation tests involves the testing of the program models associated with calculation of solar position. Rather than generate reference data by use of an algebraic model (of which there are several), reference data have been obtained for two sites and two dates from the U.S. Naval Observatory. This data and that reported in the test output and weather files are recorded at standard times for the time zone appropriate

to the location. Correction for daylight savings time should be disabled in the program to be tested, or the results should be shifted by one hour.

Test Program Initialization

It has been noted in trials of the tests that some test programs require careful initialization in the case of problems involving transient conduction (e.g. IntHeatGain) in order to achieve accurate surface temperature results. This is sometimes necessary because the zone has significant thermal mass but has no time/temperature history at the beginning of the simulation. Hence it becomes necessary to ‘warm up’ the zone model by applying a zero load for some time before the start of the real simulation. It is the testers responsibility to clarify whether this is necessary for the test program in question.

References

Blottner, F.G. (1990). *Accurate Navier-Stokes Results for the Hypersonic Flow over a Spherical Nosedip*, AIAA Journal of Spacecraft and Rockets, Vol. 27, No. 2, March-April 1990, pp. 113-122.

Boehm, B.W. (1981). *Software Engineering Economics*, Prentice-Hall, 1981.

Test SSConv: Steady-state Convection

Objective

The objective of this test is to find the response to a steady difference in dry bulb temperature between the inside and outside of the zone. The heat transfer is to be only by convection at the inside and outside surfaces and by conduction through one surface. To allow testing of the inside and outside convection models the conductive resistance is made negligible by suitable choice of thickness and thermal conductivity.

Analytical Model

The steady state heat transfer model consists of a model of one-dimensional steady state heat conduction through a homogeneous slab with convective boundary conditions. Convection is modeled as a resistance between the surface and dry bulb temperatures that is defined by a power law relationship with temperature difference.

Zone Description

The test zone geometry is cubic with internal dimensions 3×3×3m. No windows are present in this test. Only one surface is to be external and is to be a single homogeneous layer. The external wall corresponds to the homogeneous slab in the analytical model. The zone air temperature should be fixed. The test may be applied to any zone surface where the dry bulb temperature can be fixed, and the convection coefficients are known, on both sides of the surface. Transient effects need to be avoided by making the fabric heat capacity and/or density suitably low.

The effects of solar irradiation, long wave radiation, infiltration and internal gains must be eliminated in this test. This can normally be achieved by setting suitable surface properties and setting infiltration rate and internal gains to zero. Zone orientation and other location parameters are also irrelevant in this test.

Test Parameters

The driving temperature difference in this test is a constant difference between inside and outside drybulb temperatures. The user is able to choose the outside and inside temperatures when generating the weather files and the analytical results using the toolkit software. The same value of inside air temperature used in the calculation of the analytical solution must be used in the input data of the program to be tested.

The user, when calculating the response, is able to set the wall thermal conductivity in addition to the thickness. In order for the test to show sensitivity to the convection model

it is desirable for the conduction resistance to be very low. Accordingly, the user must set a suitably high thermal conductivity in relation to the thickness.

Convection coefficients have to be specified for the inside and external surfaces. Most convection correlations used for exterior and interior building heat transfer should be reducible to the form (e.g. that given by Yazdanian and Klems 1994 at zero air speed),

$$h = A + C(T_s - T_\infty)^n \quad (\text{SSConv-1})$$

where,

h = convection coefficient, W/(m²-K) or Btu/(hr-ft²-F)

A = constant, W/(m²-K) or Btu/(hr-ft²-F)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C or F

T_∞ = air temperature, C or F.

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations. The Test Suite software input screen that allows input of the parameters for this test is shown in Figure SSConv-1

Test Temperatures	
Inside temperature	10.0 (°C)
Outside temperature	40.0 (°C)

Convection Coefficients			
	Coefficient 'A'	Coefficient 'C'	Coefficient 'n'
Inside correlation	0.0	1.49	0.345 (W/m ² .K)
Outside correlation	0.0	0.84	0.333 (W/m ² .K)

Fabric Properties	
Conductivity	1.0 (W/m.K)
Thickness	0.1 (m)

Start Calculation

Status: TMY2 SI Units

Figure SSConv-1: Test Suite software parameter input screen for test SSConv.

Test Output

The principle data of interest in this test are the predicted inside and outside surface temperatures and the steady state zone load. These data are listed along with the input data in the output file, as well as the final values of the actual convection coefficients.

Test Results

The tabulated results included below in table SSConv-B have been produced with the following test parameters shown in Table SSConv-A. Inside and outside convection coefficient parameters are as given by the correlations of Awbi (1998) and Yazdanian and Klems (1994) respectively.

Table SSConv-A
Test parameters used in generating the tabulated results

Test Parameter	Value	Units
Thermal conductivity	1.0	W/m.K
Thickness	0.1	m
Inside temperature	10.0	°C
Outside Temperature	40.0	°C
Outside correlation coefficient 'A'	0.0	W/m ² .K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² .K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table SSConv-B
Tabulated results for the Steady-state convection test

Test Parameter	Value	Units
Heat Flux	34.4600	W/m ²
Zone Load	310.14	W
Inside Surface Temperature	20.3329	°C
Outside Surface Temperature	23.7789	°C
Inside convection coefficient	3.3349	W/m ² .K
Outside convection coefficient	2.1244	W/m ² .K

Analytical Solution

The analytical solution used in this test is based on the one-dimensional steady heat conduction equation with convective boundary conditions. The convention is made that a heat flux into the zone (cooling load) has a positive sign. The steady heat flux can be found from the total thermal resistance R_T and the overall temperature difference (Simpson 1975, HoF 1997: Chapter 3.),

$$q = \frac{(T_{a,o} - T_{a,i})}{R_T} \quad (\text{SSConv-2})$$

The total thermal resistance between the outside and inside air is the summation of the surface and fabric resistances so that,

$$R_T = R_{c,o} + R_f + R_{c,i} \quad (\text{SSConv-3})$$

where,

$$R_{c,o} = 1/h_{c,o} \quad (\text{SSConv-4})$$

$$R_f = L/k \quad (\text{SSConv-5})$$

$$R_{c,i} = 1/h_{c,i} \quad (\text{SSConv-6})$$

The convection coefficients being determined in this case from the correlations,

$$h_{c,o} = A_{c,o} + C_{c,o} \left(|T_{s,o} - T_{a,o}| \right)^{n_o} \quad (\text{SSConv-7})$$

$$h_{c,i} = A_{c,i} + C_{c,i} \left(|T_{s,i} - T_{a,i}| \right)^{n_i} \quad (\text{SSConv-8})$$

The zone load is simply the flux q multiplied by the relevant surface area (9.0 m²). The inside and outside surface temperatures can be finally calculated using the flux as follows,

$$T_{s,o} = T_{o,a} - \frac{q}{h_{c,o}} \quad (\text{SSConv-9})$$

and

$$T_{s,i} = T_{i,a} + \frac{q}{h_{c,i}} \quad (\text{SSConv-10})$$

References

Awbi, H.B., 1998. Calculation of convective heat transfer coefficients of room surfaces for natural convection. *Energy and Buildings*, 28(2): 219-227.

HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Atlanta, GA.

Simpson J.R., 1975. *Engineering Heat Transfer*. Macmillan: London. pp. 257.

Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test SSCond: Steady-state Conduction

Objective

The objective of this test is to find the response to a steady difference in temperature between the inside and outside of the zone. The heat transfer is to be only by convection at the inside and outside surfaces and by conduction through one surface. The test is very similar to that for steady-state convection. In this test however, it is intended that the conductive resistance is more significant than the convective resistances. Multi-layer fabrics are also intended to be used in this test.

Analytical Model

The steady state heat transfer model consists of a model of one-dimensional steady state heat conduction through a multi-layer slab with convective boundary conditions. Convection is modeled as a resistance between the surface and dry bulb temperatures that is defined by a power law relationship with temperature difference.

Zone Description

The test zone geometry is cubic with internal dimensions 3×3×3m. No windows are present in this test. Only one surface is to be external. The external wall corresponds to the homogeneous slab in the analytical model. The external surface to be tested can have a number of material layers. The zone air temperature should be fixed. Transient effects need to be avoided by making the fabric massless (see discussion in the introduction).

The effects of solar irradiation, long wave radiation, infiltration and internal gains must be eliminated in this test. This can normally be achieved by setting suitable surface properties and setting infiltration rate and internal gains to zero. Zone orientation and other location parameters are also irrelevant in this test.

Test Parameters

The driving temperature difference in this test is a constant difference between inside and outside drybulb temperatures. The user is able to choose the outside and inside temperatures when generating the weather files and the analytical results using the toolkit software. The same value of inside air temperature must also be used in the input data of the program to be tested.

The user, when calculating the response, has to specify the total number of fabric layers and the thermal conductivity and thickness of each layer. Convection coefficients have to be specified for the inside and external surfaces. Most convection correlations used for

exterior and interior building heat transfer should be reducible to the form (e.g. that given by Yazdanian and Klems 1994 at zero air speed),

$$h = A + C(T_s - T_\infty)^n \quad (\text{SSCond-1})$$

where,

h = convection coefficient, W/(m²-K) or Btu/(hr-ft²-F)

A = constant, W/(m²-K) or Btu/(hr-ft²-F)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C or F

T_∞ = air temperature, C or F.

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations. The Test Suite software input screen that allows input of the parameters for this test is shown in Figure SSCond-1

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCouple WinReveal

SSConv **SSCond** TC 1 TC 2 TC 3 ExtSolRad SolRadGlazing SolRadShade

Test Temperatures

Inside temperature: 10.0 (°C)

Outside temperature: 40.0 (°C)

Convection Coefficients

	Coefficient 'A'	Coefficient 'C'	Coefficient 'n'	Units
Inside correlation	0.0	1.49	0.345	(W/m ² .K)
Outside correlation	0.0	0.84	0.333	(W/m ² .K)

Fabric Properties

Layer Number	Thickness (m)	Conductivity (W/m.K)
1	0.1	0.1
2	0.05	0.05
3	0.05	0.25

Number of layers: 3

Start Calculation

Status: TMY2 SI Units

Figure SSCond-1: Test Suite software parameter input screen for test SSCond.

Test Output

The principle data of interest in this test are the predicted inside and outside surface temperatures and the steady state zone load. These data are listed along with the input data in the output file, and the final values of the actual convection coefficients.

Test Results

The tabulated results included below in table SSCond-B have been produced with the test parameters shown in Table SSCond-A. The ordering of the layers is unimportant in this steady-state test (the same result should be obtained with any ordering of the fabric layers). Inside and outside convection coefficient parameters are as given by the correlations of Awbi (1998) and Yazdanian and Klems (1994) respectively.

Table SSCond-A
Test parameters used in generating the tabulated results

Test Parameter	Value	Units
Number of fabric layers	3	-
Thermal conductivity: Layer 1	0.1	W/m.K
Thickness: Layer 1	0.1	m
Thermal conductivity: Layer 2	0.05	W/m.K
Thickness: Layer 2	0.05	m
Thermal conductivity: Layer 3	0.25	W/m.K
Thickness: Layer 3	0.05	m
Inside temperature	10.0	°C
Outside Temperature	40.0	°C
Outside correlation coefficient 'A'	0.0	W/m ² .K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² .K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table SSCond-B
Tabulated results for the Steady-state convection test

Test Parameter	Value	Units
Heat Flux	9.1554	W/m ²
Zone Load	82.40	W
Inside Surface Temperature	13.8568	°C
Outside Surface Temperature	33.9987	°C
Inside convection coefficient	2.3737	W/m ² .K
Outside convection coefficient	1.5255	W/m ² .K

Analytical Solution

The analytical solution used in this test is based on one-dimensional steady heat conduction. The convention is made that a heat flux into the zone (cooling load) has a positive sign. The steady heat flux can be found from the total thermal resistance R_T and the overall temperature difference (Simpson 1975, HoF 1997: Chapter 3.),

$$q = \frac{(T_{a,o} - T_{a,i})}{R_T} \quad (\text{SSCond-2})$$

The total thermal resistance between the outside and inside air is the summation of the surface and fabric resistances so that,

$$R_T = R_{c,o} + \sum_{j=1}^N R_{f,j} + R_{c,i} \quad (\text{SSCond-3})$$

where,

$$R_{c,o} = 1/h_{c,o} \quad (\text{SSCond-4})$$

$$R_{f,j} = L_j/k_j \quad j=1, N \quad \text{the number of fabric layers} \quad (\text{SSCond-5})$$

$$R_{c,i} = 1/h_{c,i} \quad (\text{SSCond-6})$$

The convection coefficients being determined, in this case, from the correlations,

$$h_{c,o} = A_{c,o} + C_{c,o} (T_{s,o} - T_{a,o})^{n_o} \quad (\text{SSCond-7})$$

$$h_{c,i} = A_{c,i} + C_{c,i} (T_{s,i} - T_{a,i})^{n_i} \quad (\text{SSCond-8})$$

The zone load is simply the flux q multiplied by the relevant surface area (9.0 m²). The inside and outside surface temperatures can be finally calculated using the flux as follows,

$$T_{s,o} = T_{a,o} - \frac{q}{h_{c,o}} \quad (\text{SSCond-9})$$

and

$$T_{s,i} = T_{a,i} + \frac{q}{h_{c,i}} \quad (\text{SSCond-10})$$

References

- Awbi, H.B., 1998. Calculation of convective heat transfer coefficients of room surfaces for natural convection.. *Energy and Buildings*, 28(2): 219-227.
- HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.
- Simpson J.R., 1975. *Engineering Heat Transfer*. Macmillan: London. pp. 257.

Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test TC1: Transient Conduction – Adiabatic Wall

Objective

The objective of this test is to find the response—in terms of wall surface temperatures and surface heat flux—to step changes in external dry bulb temperature when the inside surface of the test wall is adiabatic.

Analytical Model

The heat fluxes in this test are driven by step changes in dry bulb temperature through convective heat transfer. The conceptual model of the adiabatic surface is illustrated in Figure TC1-A(a). Heat transfer is assumed to be one-dimensional. The analytical solution is based on one-dimensional conduction through a homogeneous slab with a convective and an adiabatic boundary condition. Convection is modeled as a fixed resistance between the surface and the adjacent air. The model of the slab with these boundary conditions can be used to represent a zone wall in a number of configurations discussed below.

The response to the step changes should be seen in the external surface temperature and heat flux. After each step change in dry bulb temperature the surface temperature should show a first order response and return to equilibrium. The surface temperature at the adiabatic surface will also show a transient response and return to equilibrium with the dry bulb and exposed surface temperatures. The adiabatic surface temperature response lags that of the exposed surface temperature.

Zone Description

The test zone geometry is cubic with internal dimensions 3×3×3m. No windows are present in this test. Only one surface is to be external. This external surface may be selected to be any of the six surfaces of the zone, as any surface should have the same response. In other words the user may wish to test the response of the roof or a suspended floor in the same way as the walls. The external surface to be tested must have a single homogeneous layer (possibly multiple layers of the same material).

It is possible to apply this test to a number of zone configurations. The intended configuration with an external wall is illustrated in Figure TC1-A(b). In this case all the zone surfaces except the external test surface are adiabatic but remain convectively coupled. In this case the internal zone air temperature must float (no system) to reproduce the correct response at the inside surface of the test construction. In this case the inside dry bulb temperature is assumed to be equal to the adiabatic wall surface temperature. Other arrangements may be necessary if the air is given thermal mass, or the zone air temperature cannot be made to float.

The analytical solution for an adiabatic external wall can also be applied to two other cases. Firstly, with all the zone surfaces specified as adiabatic, a step change in internal dry bulb temperature should result in the same response given by the analytical solution. This arrangement is illustrated in Figure TC1-A(c). If the test program allows an adjacent adiabatic zone to be represented with the same temperatures applied to the wall in the adjacent zone, the same analytical solution applies for a wall that is twice the thickness specified in the analytical calculation. In this case the analytically predicted adiabatic (inside) surface temperature is equivalent to the temperature midway through the wall. This arrangement is shown in Figure TC1-A(d). Although it may be difficult to obtain the temperature midway through the wall, it should be possible in this case to compare both the inside and outside surface temperatures of the test zone with the analytical calculation of the external temperature.

The effects of solar irradiation, long wave radiation, infiltration and internal gains must be eliminated in this test. This can normally be achieved by setting suitable surface properties and setting infiltration rate and internal gains to zero. Zone orientation and other location parameters are irrelevant in this test.

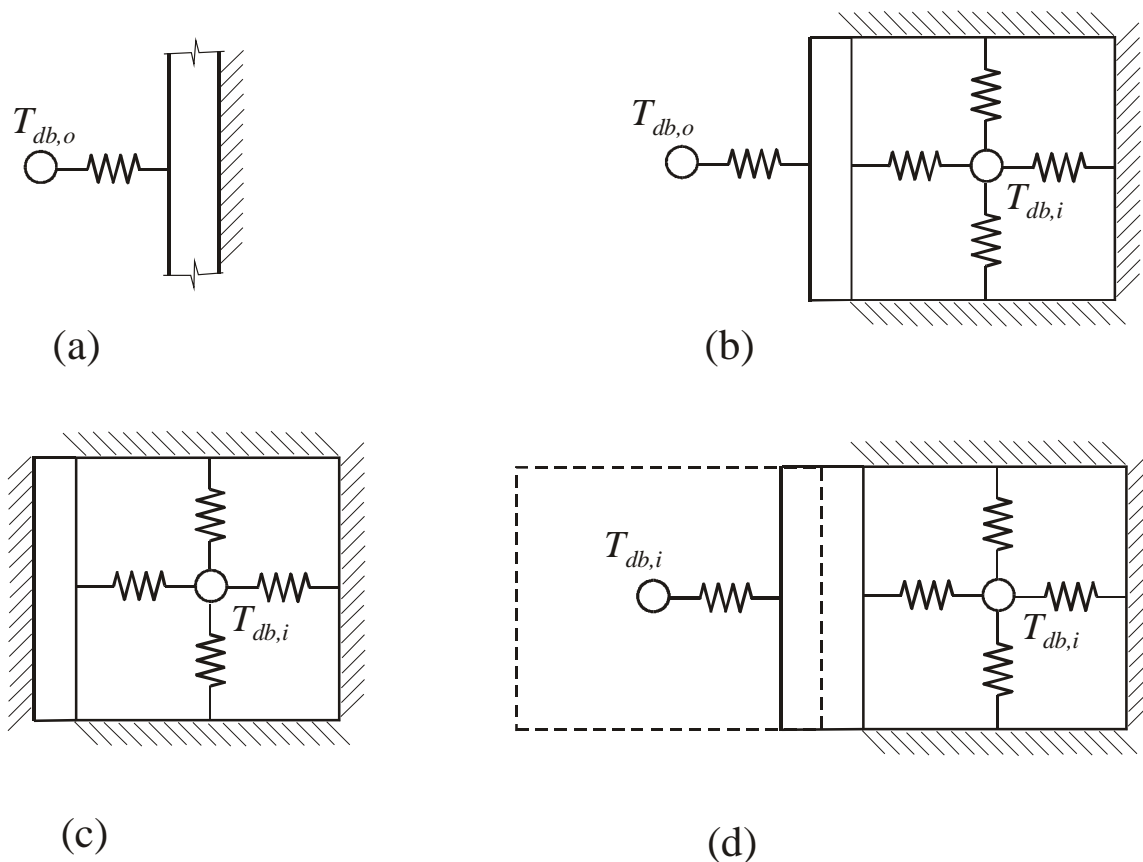


Figure TC1-A: (a) an illustration of the conceptual model of this test on an adiabatic wall, (b) a possible interpretation of the test for a zone with an external wall with an

effectively adiabatic inner surface, (c) application of the test to a zone with an adiabatic internal wall construction, (d) application to a zone with an inter-zone partition and same temperature on each side (the notional adjacent zone is also adiabatic).

Test Parameters

The driving external temperature in this test has a step up and a reverse step from a chosen datum level over a one-year period as shown in Figure TC1-B. The user is able to choose the datum temperature and size of the temperature step when generating the weather files and the analytical results using the toolkit software.

The user, when calculating the response, is able to set the thermal properties of the wall (conductivity, density and specific heat capacity) in addition to thickness. Convection coefficients have to be specified for the inside and external surfaces. These coefficients have to be taken as constant coefficients. This is a common limitation of analytical solutions to transient conduction problems. The Test Suite software input screen that allows input of the parameters for this test is shown in Figure TC1-C

It should be noted that if the thermal properties are chosen such that the response is of very long duration conditions might not have returned to equilibrium before the next step change occurs. In this case the analytical solution for the steps following the initial step (i.e. after 4299 hours) will be invalid and the analytical solution software will generate a warning message.

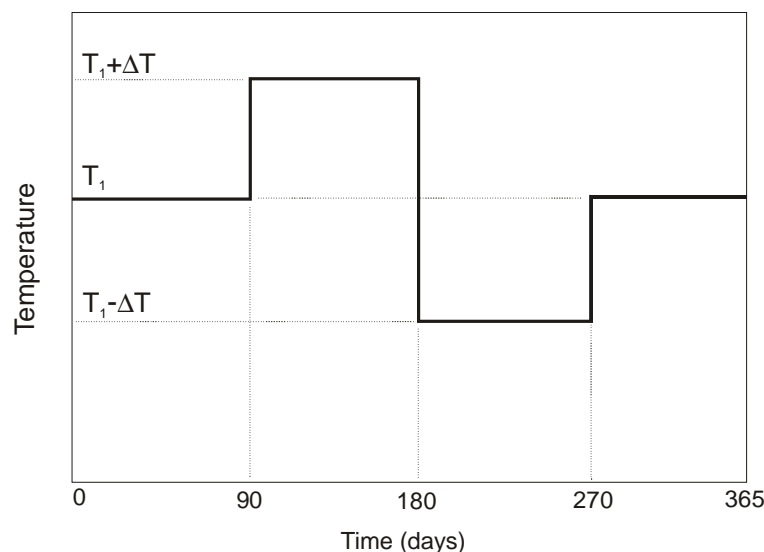


Figure TC1-B: Annual profile of the driving external dry bulb temperature

Figure TC1-C: Test Suite software parameter input screen for test TC1.

Test Output

The principle data of interest in this test are the predicted internal surface temperature and the external surface temperature and heat flux. If the zone temperature is allowed to float it should correspond to the internal surface temperature. This data is listed in the analytical response data file. This file consists of five columns of hourly values for the whole year as follows:

1. hour number,
2. external dry bulb temperature,
3. internal surface temperature,
4. external surface temperature,
5. external surface heat flux.

The time taken for the wall external surface to reach equilibrium with the external dry bulb temperature (and for the surface flux to return to zero) will vary depending on the chosen thermal properties. For most building materials the transient effects should be seen within 100 hours after each step in external temperature. Consequently the data of most interest from the annual hourly results will mostly be in the periods 2160-2260 hours, 4320-4420 hours and 6480-6580 hours.

Test Results

The tabulated results included below have been produced with the following test parameters shown in Table TC1-A. The parameter values are those used by Bland (1992). The analytical solution for these parameters is also plotted in Figure TC1-D.

Table TC1-A
Test parameters used in generating the tabulated results

Test Parameter	Value	Units
Thermal conductivity	0.14	W/m.K
Density	500	Kg/m ³
Specific heat capacity	2500	J/kg.K
Thickness	0.1	m
Initial temperature (T_0)	10.0	°C
Temperature step (ΔT)	30.0	°C
External convection coefficient	20.0	W/m ² .K
Internal convection coefficient	20.0	W/m ² .K

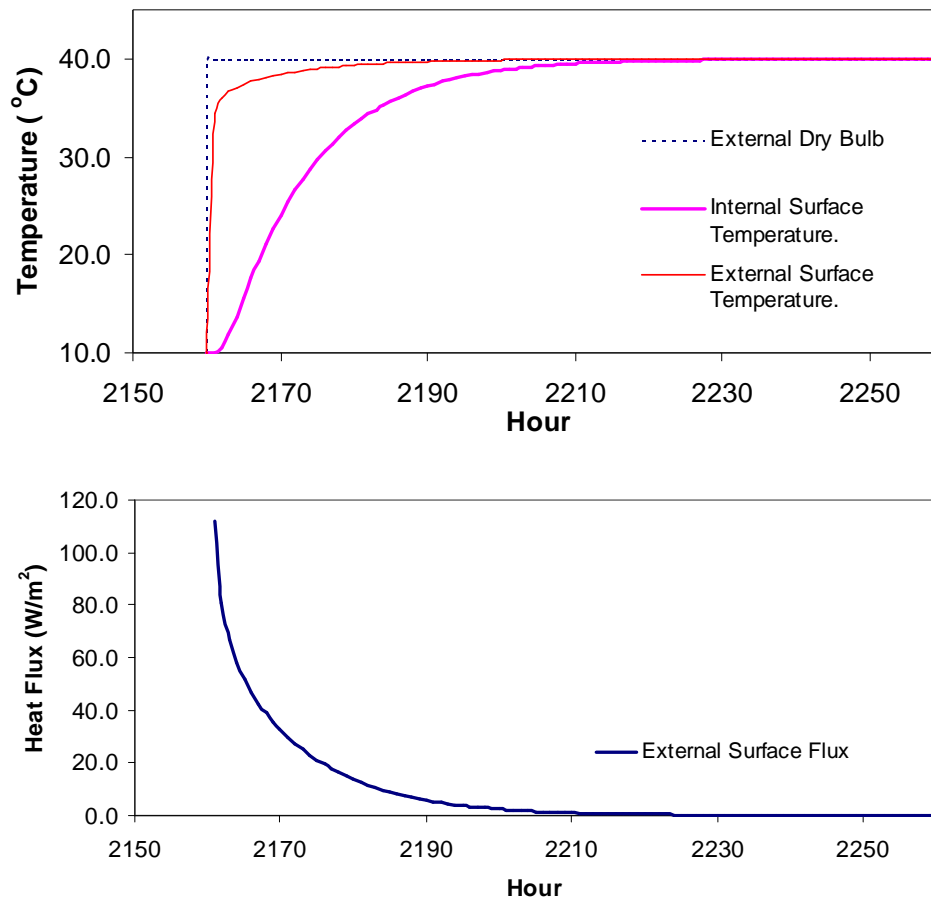


Figure TC1-D: The analytical solution for the parameters in Table TC1-A. The external surface temperature response is shown above and the surface heat flux below.

The tabulated results of the analytical solution for 100 hours after the first step in temperature (hours 2160-2260) are given below.

Table TC1-B
Results of the analytical calculation using the test parameters in Table TC1-A

Hour	Internal Surface Temperature (°C)	External Surface Temperature (°C)	Surface Flux (W/m ²)
2160	10.0000	10.0000	0.0000
2161	10.0131	34.4070	111.8601
2162	10.5080	35.9444	81.1112
2163	11.8613	36.6592	66.8150
2164	13.6783	37.0984	58.0321
2165	15.6274	37.4122	51.7560
2166	17.5458	37.6605	46.7907
2167	19.3648	37.8702	42.5960
2168	21.0599	38.0543	38.9131
2169	22.6264	38.2195	35.6109
2170	24.0682	38.3691	32.6175
2171	25.3925	38.5056	29.8888
2172	26.6078	38.6303	27.3943
2173	27.7224	38.7445	25.1106
2174	28.7444	38.8491	23.0186
2175	29.6815	38.9449	21.1015
2176	30.5406	39.0328	19.3443
2177	31.3282	39.1133	17.7335
2178	32.0502	39.1872	16.2569
2179	32.7122	39.2548	14.9033
2180	33.3190	39.3169	13.6624
2181	33.8752	39.3738	12.5248
2182	34.3852	39.4259	11.4820
2183	34.8527	39.4737	10.5259
2184	35.2813	39.5175	9.6495
2186	36.0344	39.5945	8.1095
2188	36.6672	39.6592	6.8153
2190	37.1991	39.7136	5.7276
2192	37.6461	39.7593	4.8135
2194	38.0218	39.7977	4.0453
2196	38.3375	39.8300	3.3997
2198	38.6028	39.8571	2.8572
2200	38.8258	39.8799	2.4012
2202	39.0132	39.8991	2.0180

2204	39.1707	39.9152	1.6959
2206	39.3030	39.9287	1.4253
2208	39.4143	39.9401	1.1978
2210	39.5077	39.9497	1.0066
2214	39.6523	39.9645	0.7110
2218	39.7544	39.9749	0.5021
2222	39.8266	39.9823	0.3547
2226	39.8775	39.9875	0.2505
2230	39.9135	39.9912	0.1769
2236	39.9486	39.9947	0.1050
2242	39.9695	39.9969	0.0623
2248	39.9819	39.9982	0.0370
2260	39.9936	39.9993	0.0130

Analytical Solution

The analytical solution used in this test is based on the one-dimensional unsteady heat conduction equation that can be expressed as,

$$\frac{dT}{dt} + \frac{k}{\rho C_p} \frac{d^2 T}{dx^2} = 0 \quad (\text{TC1-1})$$

where $T(x,t)$ is the temperature of the solid. The boundary conditions applied are

$$\begin{aligned} T(x,0) &= T_0 = T_1 & t &= 0 \\ T_o &= T_1 + \Delta T & t &> 0 \\ \frac{dT}{dx} &= 0 & x &= 0, t > 0 \\ -k \frac{dT}{dx} &= h_{c,o}(T_o - T) & x &= L, t > 0 \end{aligned}$$

The solution is given (Schneider 1955, Incropera 1990; pp240) as,

$$T = T_o + (T_0 - T_o) \sum_{n=1}^{\infty} C_n \exp(-\zeta_n \frac{\alpha t}{L^2}) \cos(\zeta_n \frac{x}{L}) \quad (\text{TC1-2})$$

where the coefficient C_n is given by

$$C_n = \frac{4 \sin \zeta_n}{2 \zeta_n + \sin(2 \zeta_n)} \quad (\text{TC1-3})$$

and the discrete values of ζ_n are positive roots of the transcendental equation

$$\zeta_n \tan \zeta_n = \frac{h_{c,o} L}{k} \quad (\text{TC1-4})$$

As time approaches infinity the whole of the solid approaches thermal equilibrium with the air temperature, so that $T(x, \infty) = T_o$. The surface flux correspondingly reduces to zero. This problem definition is the same as that suggested by Bland (1992) but the analytical solution is expressed slightly differently.

References

- Bland, B.H., 1992. Conduction in dynamic thermal models: Analytical tests for validation. *Building Services Engineering Research and Technology*. 13(4): 197-208.
- Incropera, F.P. and DeWitt, D.P., 1990. *Introduction to Heat Transfer*. 2nd Ed. New York: John Wiley.
- Schneider, P.J., 1955. *Conduction Heat Transfer*. Reading MA: Addison-Wesley.

Test TC2: Transient Conduction – Step Response

Objective

The objective of this test is to find the response to step changes in external dry bulb temperature when the inside air temperature is held constant. In this test convective heat transfer occurs at both internal and external surfaces of one wall. This test builds on the steady-state convection and conduction tests.

Analytical Model

The heat fluxes in this test are driven by step changes in external dry bulb only with the interior dry bulb remaining constant. The analytical solution is based on one-dimensional conduction through a homogeneous slab with convective boundary conditions on either side. Convection is modeled as a fixed resistance between the surface and the adjacent air. Time varying boundary conditions are applied over four time periods representing the sections of a stepped profile. This model of a single wall is used to represent one external wall of the test zone.

The response to the step changes should be seen in the internal and external surface temperature and heat fluxes as well as the internal load. After each step change in external temperature the external surface temperature should show a first order response and reach a steady state condition. Unlike the adiabatic transient test (TC1) the transient change at the start of each step is influenced by the previous step – the initial conditions at the start of each step being different. The analytical solution is correspondingly more complex.

Zone Description

The test zone geometry is cubic with internal dimensions 3×3×3m. No windows are present in this test. Only one surface is to be external. This external surface may be selected to be any of the six surfaces of the zone, as any surface should have the same response. In other words the user may wish to test the response of the roof or a suspended floor in the same way as the walls. The external surface to be tested must have a single homogeneous layer. The zone air temperature should be arranged to remain constant and at the same value as the initial outside dry bulb.

The effects of solar irradiation, long wave radiation, infiltration and internal gains must be eliminated in this test. This can normally be achieved by setting suitable surface properties and setting infiltration rate and internal gains to zero. Zone orientation and other location parameters are also irrelevant in this test.

Test Parameters

The driving external temperature in this test has a step up and a reverse step from a chosen datum level over a one-year period as shown in Figure TC2-A. The user is able to choose the datum temperature and size of the temperature step when generating the weather files and the analytical results using the toolkit software.

The user, when calculating the response, is able to set the thermal properties of the wall (conductivity, density and specific heat capacity) in addition to thickness. The wall consists of a single homogeneous layer. Convection coefficients have to be specified for the inside and external surfaces. These coefficients have to be taken as constant coefficients. This is a common limitation of analytical solutions to transient conduction problems. The Test Suite software input screen that allows input of the parameters for this test is shown in Figure TC2-B

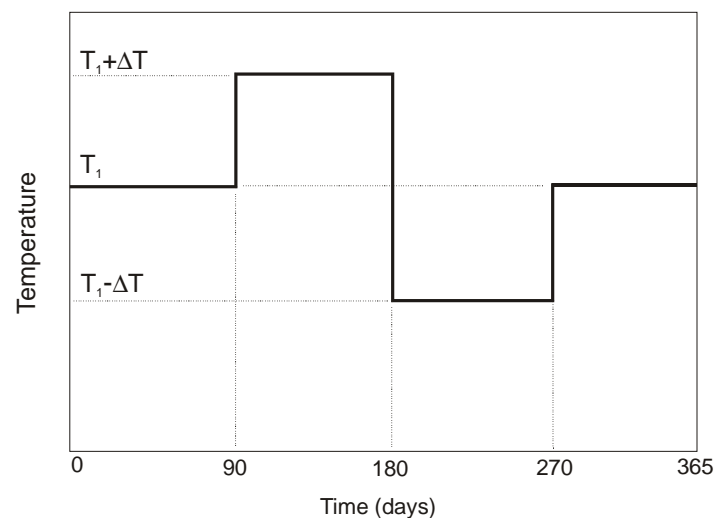


Figure TC2-A: Annual profile of the driving external dry bulb temperature

Figure TC2-B: Test Suite software parameter input screen for test TC2.

Test Output

The principle data of interest in this test are the predicted internal surface temperature and the external surface temperatures, heat fluxes and resultant internal load. This data is listed in the analytical response data file. This file consists of seven columns of hourly values for the whole year as follows:

1. hour number,
2. external dry bulb temperature,
3. internal surface temperature,
4. external surface temperature,
5. internal surface heat flux,
6. external surface heat flux,
7. zone internal load.

The time taken for the wall surface temperature and fluxes to reach steady state will vary depending on the chosen thermal properties. Most of the transient effects in the results will be seen in the periods 2160-2260 hours, 4320-4420 hours and 6480-6580 hours.

It should also be noted that the steady-state portion of the solution and results should correspond to that of the steady state tests when used with the same parameters. Both the

analytical solution and the test program results can be checked in this way. Furthermore, if this test is run with an inside convection coefficient of zero, the output should be equivalent to that of the adiabatic test TC1.

Test Results

The tabulated results included below have been produced with the following test parameters shown in Table TC2-A. The analytical solution for these parameters is plotted in Figures TC2-C and TC2-D. Tabulated results are presented in Tables TC2-B and TC2-C for the first and second dry bulb temperature steps.

Table TC2-A
Test parameters used in generating the tabulated results

Test Parameter	Value	Units
Thermal conductivity	0.14	W/m.K
Density	500	Kg/m ³
Specific heat capacity	2500	J/kg.K
Thickness	0.1	m
Initial temperature (T_0)	10.0	°C
Temperature step (ΔT)	30.0	°C
External convection coefficient	2.607	W/m ² .K
Internal convection coefficient	3.18	W/m ² .K

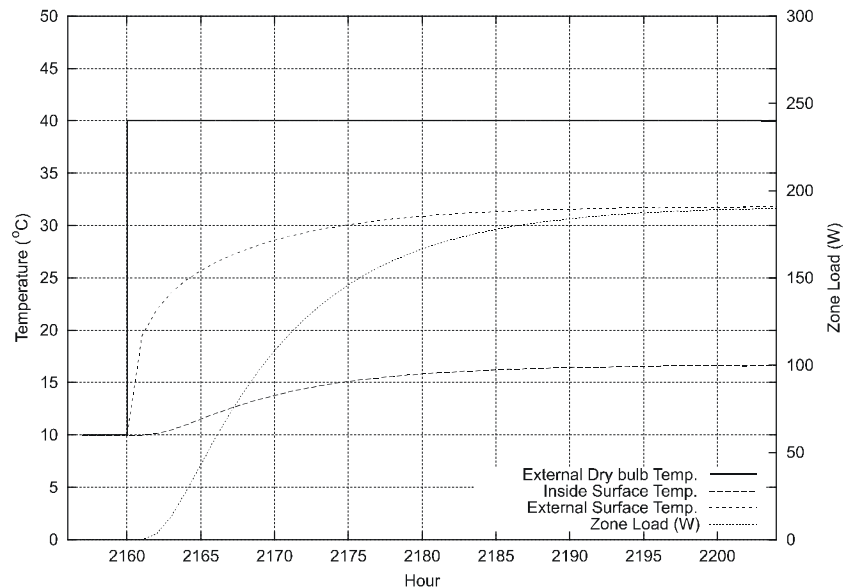


Figure TC2-C: The analytical solution for the parameters in Table TC2-A for the 48 hours around the first step change in outside dry bulb temperature.

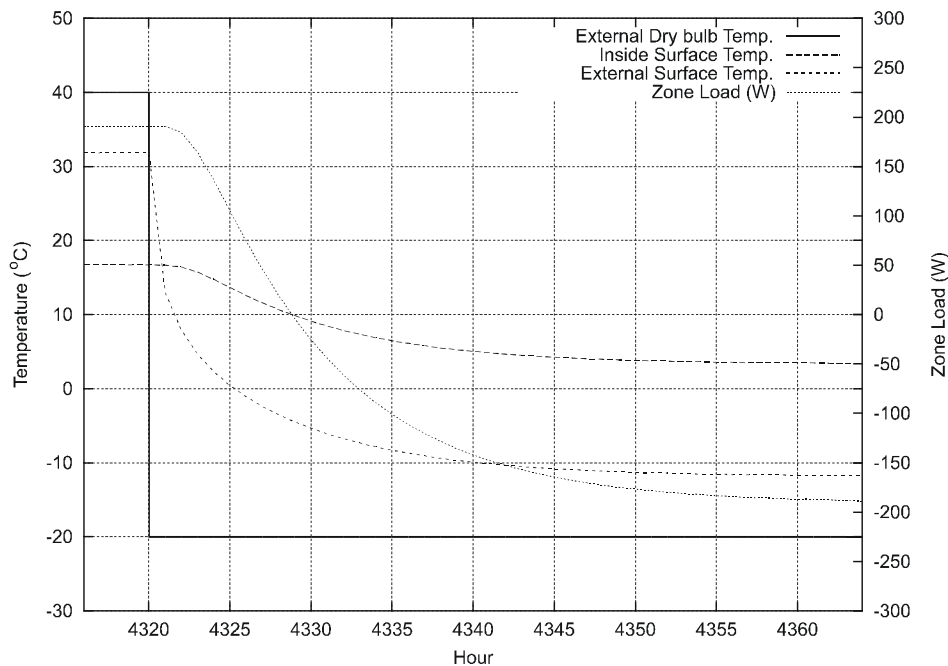


Figure TC2-D: The analytical solution for the parameters in Table TC2-A. for the 48 hours around the second step change in outside dry bulb temperature.

Table TC2-B

Tabulated values of the analytical solution for the first step in outside dry bulb temperature. This solution was computed using the parameters in Table TC2-A.

Hour	Internal Surface Temperature (°C)	External Surface Temperature (°C)	Internal Surface Flux (W/m ²)	External Surface Flux (W/m ²)	Internal Load (W)
2161	10.0026	19.4043	0.0083	53.6930	0.07
2162	10.1180	21.9634	0.3752	47.0214	3.38
2163	10.4665	23.5851	1.4835	42.7935	13.35
2164	10.9605	24.7701	3.0545	39.7043	27.49
2165	11.5028	25.7015	4.7788	37.2762	43.01
2166	12.0379	26.4674	6.4807	35.2794	58.33
2167	12.5402	27.1158	8.0778	33.5891	72.70
2168	12.9995	27.6746	9.5383	32.1323	85.84
2169	13.4135	28.1616	10.8550	30.8627	97.69
2170	13.7839	28.5888	12.0329	29.7489	108.30
2171	14.1138	28.9650	13.0820	28.7682	117.74
2172	14.4069	29.2971	14.0140	27.9025	126.13
2173	14.6669	29.5905	14.8408	27.1375	133.57
2174	14.8974	29.8501	15.5736	26.4609	140.16
2175	15.1015	30.0797	16.2229	25.8622	146.01

2176	15.2824	30.2830	16.7979	25.3323	151.18
2177	15.4425	30.4629	17.3072	24.8633	155.76
2178	15.5843	30.6221	17.7581	24.4481	159.82
2179	15.7099	30.7632	18.1574	24.0805	163.42
2180	15.8210	30.8880	18.5109	23.7550	166.60
2181	15.9195	30.9985	18.8239	23.4669	169.41
2182	16.0066	31.0964	19.1010	23.2118	171.91
2183	16.0838	31.1830	19.3464	22.9859	174.12
2184	16.1521	31.2597	19.5636	22.7859	176.07
2186	16.2661	31.3878	19.9262	22.4521	179.34
2188	16.3555	31.4881	20.2105	22.1905	181.89
2190	16.4256	31.5668	20.4333	21.9854	183.90
2192	16.4805	31.6285	20.6079	21.8246	185.47
2194	16.5235	31.6768	20.7449	21.6986	186.70
2196	16.5573	31.7147	20.8522	21.5998	187.67
2198	16.5837	31.7444	20.9363	21.5223	188.43
2200	16.6045	31.7677	21.0022	21.4616	189.02
2202	16.6207	31.7859	21.0539	21.4140	189.49
2204	16.6335	31.8003	21.0945	21.3767	189.85
2206	16.6435	31.8115	21.1262	21.3475	190.14
2208	16.6513	31.8203	21.1511	21.3246	190.36

Table TC2-C

Tabulated values of the analytical solution for the second step in outside dry bulb temperature.
This solution was computed using the parameters in Table TC2-A.

Hour	Internal Surface Temperature (°C)	External Surface Temperature (°C)	Internal Surface Flux (W/m ²)	External Surface Flux (W/m ²)	Internal Load (W)
4321	16.6745	13.0436	21.2248	-86.1446	191.02
4322	16.4437	7.9253	20.4910	-72.8013	184.42
4323	15.7467	4.6819	18.2744	-64.3456	164.47
4324	14.7586	2.3119	15.1325	-58.1671	136.19
4325	13.6742	0.4491	11.6839	-53.3109	105.16
4326	12.6038	-1.0827	8.2801	-49.3173	74.52
4327	11.5993	-2.3794	5.0858	-45.9368	45.77
4328	10.6808	-3.4971	2.1649	-43.0231	19.48
4329	9.8527	-4.4711	-0.4685	-40.4839	-4.22
4330	9.1118	-5.3255	-2.8244	-38.2564	-25.42
4331	8.4520	-6.0779	-4.9225	-36.2949	-44.30
4332	7.8659	-6.7420	-6.7865	-34.5637	-61.08
4333	7.3459	-7.3289	-8.4401	-33.0336	-75.96

4334	6.8850	-7.8480	-9.9058	-31.6804	-89.15
4335	6.4766	-8.3072	-11.2043	-30.4830	-100.84
4336	6.1150	-8.7138	-12.3544	-29.4232	-111.19
4337	5.7947	-9.0736	-13.3729	-28.4852	-120.36
4338	5.5111	-9.3921	-14.2748	-27.6547	-128.47
4339	5.2600	-9.6741	-15.0733	-26.9195	-135.66
4340	5.0376	-9.9238	-15.7803	-26.2686	-142.02
4341	4.8408	-10.1449	-16.4063	-25.6923	-147.66
4342	4.6665	-10.3406	-16.9606	-25.1821	-152.65
4343	4.5122	-10.5139	-17.4513	-24.7304	-157.06
4344	4.3755	-10.6673	-17.8858	-24.3304	-160.97
4346	4.1475	-10.9233	-18.6110	-23.6628	-167.50
4348	3.9687	-11.1241	-19.1795	-23.1395	-172.62
4350	3.8286	-11.2814	-19.6251	-22.7293	-176.63
4352	3.7187	-11.4048	-19.9744	-22.4077	-179.77
4354	3.6326	-11.5015	-20.2483	-22.1557	-182.23
4356	3.5651	-11.5773	-20.4629	-21.9581	-184.17
4358	3.5122	-11.6367	-20.6312	-21.8032	-185.68
4360	3.4707	-11.6832	-20.7631	-21.6818	-186.87
4362	3.4382	-11.7197	-20.8664	-21.5866	-187.80
4364	3.4127	-11.7484	-20.9475	-21.5120	-188.53
4366	3.3928	-11.7708	-21.0110	-21.4535	-189.10
4368	3.3771	-11.7884	-21.0608	-21.4077	-189.55

Analytical Solution

The analytical solution used in this test is based on the one-dimensional unsteady heat conduction equation that can be expressed as (Dougherty 1999),

$$\frac{dT}{dt} + \frac{k}{\rho C_p} \frac{d^2T}{dx^2} = 0 \quad (\text{TC2-1})$$

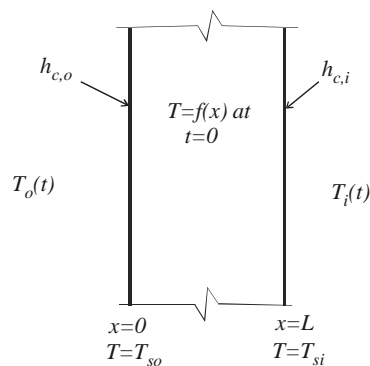


Figure TC2-E: Analytical Solution Boundary Conditions.

The boundary conditions are applied at the locations shown in Fig. TC2-E and are defined as,

$$T(x,0) = f(x) \quad (\text{TC2-2})$$

$$h_{c,o}[T_o(t) - T(x=0,t)] = -k \frac{\partial T(x=0,t)}{\partial x} \quad (\text{TC2-3})$$

$$h_{c,i}[T(x=L,t) - T_i(t)] = -k \frac{\partial T(x=L,t)}{\partial x} \quad (\text{TC2-4})$$

Or, Eqs. (TC2-3) and (TC2-4) can be written as:

$$\frac{\partial T(x=0,t)}{\partial x} - \frac{h_{c,o}}{k} T(x=0,t) = -\frac{h_{c,o}}{k} T_o(t) \quad (\text{TC2-5})$$

$$\frac{\partial T(x=L,t)}{\partial x} + \frac{h_{c,i}}{k} T(x=L,t) = \frac{h_{c,i}}{k} T_o(t) \quad (\text{TC2-6})$$

A non-dimensional temperature can be defined such that,

$$\theta(X,\tau) = \frac{T(x,t) - T_1}{T_0 - T_1}, \quad X = x / L, \quad \text{and} \quad \tau = \alpha t / L^2 \quad (\text{TC2-7})$$

where T_1 and T_0 are arbitrary constant temperatures. Then, Eqs (TC2-1), (TC2-2), (TC2-5) and (TC2-6) become:

$$\frac{\partial^2 \theta}{\partial X^2} = \frac{\partial \theta}{\partial \tau} \quad (\text{TC2-8})$$

$$\theta(X,0) = [f(XL) - T_1] / [T_0 - T_1] \quad (\text{TC2-9})$$

$$\frac{\partial \theta(X=0,\tau)}{\partial X} - \frac{h_{c,o}L}{k} \theta(X=0,\tau) = -\frac{h_{c,o}L}{k} \theta_o(\tau) \quad (\text{TC2-10})$$

$$\frac{\partial \theta(X=1,\tau)}{\partial X} + \frac{h_{c,i}L}{k} \theta(X=1,\tau) = \frac{h_{c,i}L}{k} \theta_i(\tau) \quad (\text{TC2-11})$$

where,

$$\theta_o(\tau) = \frac{T_o(\tau L^2 / \alpha) - T_1}{T_0 - T_1} \quad \text{and} \quad \theta_i(\tau) = \frac{T_i(\tau L^2 / \alpha) - T_1}{T_0 - T_1} \quad (\text{TC2-12})$$

From Özışık (p. 35), the eigenfunction $[R]$, eigenvalues $[\beta_m]$, and normalization integral $[N(\beta_m)]$ are:

$$R(X, \beta_m) = \beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X) \quad (\text{TC2-13})$$

$$\tan(\beta_m) = \frac{\beta_m \left(\frac{h_{c,i}L}{k} + \frac{h_{c,o}L}{k} \right)}{\beta_m^2 - \left(\frac{h_{c,i}L}{k} \right) \left(\frac{h_{c,o}L}{k} \right)} \quad (\beta_m \text{ are the roots of this equation.}) \quad (\text{TC2-14})$$

$$2[N(\beta_m)] = \left[\beta_m^2 + \left(\frac{h_{c,o}L}{k} \right)^2 \right] \left[1 + \frac{\frac{h_{c,i}L}{k}}{\beta_m^2 + \left(\frac{h_{c,i}L}{k} \right)^2} \right] + \frac{h_{c,o}L}{k} \quad (\text{TC2-15})$$

Using the eigenfunction to Fourier transform Eq. (TC2-8) gives:

$$\frac{1}{N(\beta_m)^{1/2}} \int_0^1 \frac{\partial^2 \theta}{\partial X^2} \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X) \right] dX = \frac{\partial \bar{\theta}(\tau)}{\partial \tau} \quad (\text{TC2-16})$$

where:

$$\bar{\theta}(\tau) = \frac{1}{N(\beta_m)^{1/2}} \int_0^1 \theta(X, \tau) \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X) \right] dX \quad (\text{TC2-17})$$

and the boundary condition is:

$$\bar{\theta}(0) = \frac{1}{N(\beta_m)^{1/2}} \int_0^1 \{ [f(XL) - T_i] / [T_o - T_i] \} \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X) \right] dX \quad (\text{TC2-18})$$

Integrating the left side of Eq. (TC2-16) by parts yields:

$$\begin{aligned} -\beta_m^2 \bar{\theta}(\tau) + \frac{1}{N(\beta_m)^{1/2}} \left\{ \frac{\partial \theta(1, \tau)}{\partial X} \left[\beta_m \cos(\beta_m) + \frac{h_{c,o}L}{k} \sin(\beta_m) \right] + \right. \\ \left. \theta(1, \tau) \left[\beta_m^2 \sin(\beta_m) - \beta_m \frac{h_{c,o}L}{k} \cos(\beta_m) \right] + \beta_m \left[-\frac{\partial \theta(0, \tau)}{\partial X} + \frac{h_{c,o}L}{k} \theta(0, \tau) \right] \right\} \end{aligned} \quad (\text{TC2-19})$$

From the eigenvalue equation [Eq. (TC2-14)], we find:

$$\beta_m^2 \sin(\beta_m) - \beta_m \frac{h_{c,o}L}{k} \cos(\beta_m) = \frac{h_{c,o}L}{k} \frac{h_{c,i}L}{k} \sin(\beta_m) + \beta_m \frac{h_{c,i}L}{k} \cos(\beta_m) \quad (\text{TC2-20})$$

which can be substituted into Eq. (TC2-19), along with substituting boundary condition Eq. (TC2-10) for the last term in Eq. (TC2-19), to yield:

$$\begin{aligned}
& -\beta_m^2 \bar{\theta}(\tau) + \frac{1}{N(\beta_m)^{1/2}} \left\{ \frac{\partial \theta(1, \tau)}{\partial x} \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] + \right. \\
& \left. \theta(1, \tau) \left[\frac{h_{c,o} L}{k} \sin(\beta_m) + \beta_m \cos(\beta_m) \right] \frac{h_{c,i} L}{k} + \beta_m \frac{h_{c,o} L}{k} \theta_o(\tau) \right\}
\end{aligned} \tag{TC2-21}$$

Equation (TC2-21) can be rearranged into the following form:

$$\begin{aligned}
& -\beta_m^2 \bar{\theta}(\tau) + \frac{1}{N(\beta_m)^{1/2}} \left\{ \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \left[\frac{\partial \theta(1, \tau)}{\partial X} + \frac{h_{c,i} L}{k} \theta(1, \tau) \right] \right. \\
& \left. + \beta_m \frac{h_{c,o} L}{k} \theta_o(\tau) \right\}
\end{aligned} \tag{TC2-22}$$

But from boundary condition Eq. (TC2-11), Eq. (TC2-22) can be reduced to:

$$\begin{aligned}
& -\beta_m^2 \bar{\theta}(\tau) + \frac{1}{N(\beta_m)^{1/2}} \left\{ \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \frac{h_{c,i} L}{k} \theta_{in}(\tau) + \beta_m \frac{h_{c,o} L}{k} \theta_o(\tau) \right\}
\end{aligned} \tag{TC2-23}$$

Equation (TC2-23) is a reduced version of the left side of Eq. (TC2-16). So, substituting Eq. (TC2-23) back into Eq. (TC2-16) gives:

$$\begin{aligned}
& \frac{\partial \bar{\theta}}{\partial \tau} + \beta_m^2 \bar{\theta}(\tau) = \frac{1}{N(\beta_m)^{1/2}} \left\{ \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \frac{h_{c,i} L}{k} \theta_i(\tau) \right. \\
& \left. + \beta_m \frac{h_{c,o} L}{k} \theta_o(\tau) \right\}
\end{aligned} \tag{TC2-24}$$

Multiplying Eq. (TC2-24) by $\exp[\beta_m^2 \tau]$, and integrating from 0 to τ gives:

$$\begin{aligned}
& \bar{\theta}(\tau) e^{\beta_m^2 \tau} - \bar{\theta}(0) = \frac{1}{N(\beta_m)^{1/2}} \left\{ \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \frac{h_{c,i} L}{k} \int_0^\tau \theta_i(\tau') e^{\beta_m^2 \tau'} d\tau' \right. \\
& \left. + \beta_m \frac{h_{c,o} L}{k} \int_0^\tau \theta_o(\tau') e^{\beta_m^2 \tau'} d\tau' \right\}
\end{aligned} \tag{TC2-25}$$

Substituting boundary condition Eq. (TC2-18) into Eq. (TC2-25), and multiplying the result by $\exp[-\beta_m^2 \tau]$ yields:

$$\begin{aligned}
\bar{\theta}(\tau) = & \frac{e^{-\beta_m^2 \tau}}{N(\beta_m)^{1/2}} \left[\left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \frac{h_{c,i} L}{k} \int_0^\tau \theta_i(\tau') e^{\beta_m^2 \tau'} d\tau' \right. \\
& + \beta_m \frac{h_{c,o} L}{k} \int_0^\tau \theta_{out}(\tau') e^{\beta_m^2 \tau'} d\tau' \} \\
& + \frac{e^{-\beta_m^2 \tau}}{N(\beta_m)^{1/2}} \int_0^1 \{ [f(XL) - T_1] / [T_0 - T_1] \} \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o} L}{k} \sin(\beta_m X) \right] dX
\end{aligned} \tag{TC2-26}$$

Inverting:

$$\theta(X, \tau) = \sum_{m=1}^{\infty} \frac{\bar{\theta}(\tau) \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o} L}{k} \sin(\beta_m X) \right]}{N(\beta_m)^{1/2}} \tag{TC2-27}$$

Substituting Eq. (TC2-26) into Eq. (TC2-27) gives:

$$\begin{aligned}
\theta(X, \tau) = & \sum_{m=1}^{\infty} \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o} L}{k} \sin(\beta_m X) \right] \frac{e^{-\beta_m^2 \tau}}{N(\beta_m)} \left\{ \left[\beta_m \cos(\beta_m) + \frac{h_{c,o} L}{k} \sin(\beta_m) \right] \right. \\
& \cdot \frac{h_{c,i} L}{k} \int_0^\tau \theta_{in}(\tau') e^{\beta_m^2 \tau'} d\tau' + \beta_m \frac{h_{c,o} L}{k} \int_0^\tau \theta_o(\tau') e^{\beta_m^2 \tau'} d\tau' \} \\
& + \frac{e^{-\beta_m^2 \tau}}{N(\beta_m)^{1/2}} \int_0^1 \{ [f(X'L) - T_1] / [T_0 - T_1] \} \left[\beta_m \cos(\beta_m X') + \frac{h_{c,o} L}{k} \sin(\beta_m X') \right] dX',
\end{aligned} \tag{TC2-28}$$

where β_m are the roots of Eq. (TC2-14). To apply a double step change in external temperature starting with zero initial temperatures, then $f(x) = 0$, $\theta_i(\tau) = 0$, $T_i = 0$, and we normalize the temperature using the size of the step DT (i.e. $T_1 = 0, T_0 = DT$), $\theta_o(\tau)$ equals the following function of time:

$$\begin{aligned}
\theta_o(\tau) = & \theta_1 = T_{o,1} / \Delta T \quad \text{for} \quad 0 \leq \tau \leq \tau_1 \\
\theta_o(\tau) = & \theta_2 = T_{o,2} / \Delta T \quad \text{for} \quad \tau_1 \leq \tau \leq \tau_2 \\
\theta_o(\tau) = & \theta_3 = T_{o,3} / \Delta T \quad \text{for} \quad \tau_2 \leq \tau
\end{aligned} \tag{TC2-29}$$

Equation (TC2-28) becomes:

(for $0 \leq \tau \leq \tau_1$)

$$\theta(X, \tau) = \frac{h_{c,o} L}{k} \sum_{m=1}^{\infty} \left[\beta_m \cos(\beta_m X) + \frac{h_{c,o} L}{k} \sin(\beta_m X) \right] \frac{(1 - e^{-\beta_m^2 \tau})}{\beta_m N(\beta_m)} \tag{TC2-30}$$

(for $\tau_1 \leq \tau \leq \tau_2$)

$$\theta(X,t) = \frac{h_{c,o}L}{k} \sum_{m=1}^{\infty} \frac{[\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X)]}{\beta_m N(\beta_m)} \quad (\text{TC2-31})$$

$$\bullet \{[(T_1 - T_2)/T_o]e^{-\beta_m^2(\tau - \tau_1)} - (T_1/T_o)e^{-\beta_m^2\tau} + T_2/T_o\}$$

(for $\tau_2 \leq \tau$)

$$\theta(X,\tau) = \frac{h_{c,o}L}{k} \sum_{m=1}^{\infty} \frac{[\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X)]}{\beta_m N(\beta_m)} \quad (\text{TC2-32})$$

$$\bullet \{[(T_1 - T_2)/T_o]e^{-\beta_m^2(\tau - \tau_1)} - (T_1/T_o)e^{-\beta_m^2\tau} + [(T_2 - T_3)/T_o]e^{-\beta_m^2(\tau - \tau_2)} + T_3/T_o\}$$

It was found that for proper numerical implementation of this solution it was better to consider each of Eq.s(TC2-30) (TC2-31) (TC2-32) as consisting of separate steady-state and transient parts. For example Eq. (TC2-30) can be separated out as:

$$\begin{aligned} \theta(X,\tau) = & \frac{h_{c,o}L}{k} \sum_{m=1}^{\infty} \left[\frac{\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X)}{\beta_m N(\beta_m)} \right] \\ & - \frac{h_{c,o}L}{k} \sum_{m=1}^{\infty} \left[\frac{\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X)}{\beta_m N(\beta_m)} \right] e^{-\beta_m^2\tau} \end{aligned} \quad (\text{TC2-33})$$

where the first summation term in Eq.(TC2-33) constitutes the steady-state solution. This term can in fact be directly replaced by the conventional steady-state conduction solution as follows,

$$\begin{aligned} \theta(X,\tau) = & \frac{T_{os} - T_i}{\Delta T} - \frac{T_{os} - T_{is} - T_i}{\Delta T} X \\ & - \frac{h_{c,o}L}{k} \sum_{m=1}^{\infty} \left[\frac{\beta_m \cos(\beta_m X) + \frac{h_{c,o}L}{k} \sin(\beta_m X)}{\beta_m N(\beta_m)} \right] e^{-\beta_m^2\tau} \end{aligned} \quad (\text{TC2-34})$$

where ΔT is the size of the temperature step, and T_{os} and T_{is} are the surface temperatures of the steady state solution given by,

$$T_{is} = \frac{T_o - T_i}{h_{c,i}(1/h_{c,i} + 1/k + 1/h_{c,o})} + T_i \quad (\text{TC2-35})$$

$$T_{os} = T_o - \frac{T_o - T_i}{h_{c,o}(1/h_{c,i} + 1/k + 1/h_{c,o})} \quad (\text{TC2-36})$$

Similar substitutions can be made for the other portions of the temperature step sequence.

Two useful properties of this analytical solution should be noted. Firstly, if the inside convection coefficient is set to zero (making the inside adiabatic) the solution will correspond to that given by test TC-1 for the same parameters. Secondly, after each step change in driving temperature the system will approach a steady-state condition similar to that given by test SSCond. This condition can also be easily checked by manual calculation.

References

Dougherty, R. 1999. Personal communication.

McQuiston, F.C., Parker, J.D., and Spitler, J.D. 2000. *Heating, Ventilating, and Air Conditioning Analysis and Design*, Fifth Edition. John Wiley and Sons, New York.

Özişik, M.N., 1980. *Heat Conduction*. New York: John Wiley.

Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test TC3: Transient Conduction – Sinusoidal Driving Temperature and Multi-layer Wall

Objective

The objective of this test is to find the response to sinusoidal changes in outside dry bulb temperature when the inside dry bulb temperature is held constant.

The heat fluxes in this test are driven by steady-periodic sinusoidal changes in external dry bulb only. The response to the driving temperature should be seen in the inside surface temperature and heat flux and hence zone load. The inside surface temperature should vary about the same mean as the outside but with reduced amplitude and a phase lag. The size of the reduction in amplitude and phase lag is dependent on the layer material properties. This test has the advantage of being able to deal with multi-layer walls.

Analytical Model

The analytical model for this test is based on the solution of the transient Fourier heat conduction equation for a multi-layer slab with convective boundary conditions. Conduction is assumed to be one-dimensional. Convection is assumed to be via a fixed resistance between the surface and adjacent air. The boundary conditions are such that the inside dry bulb temperature is fixed at the mean outside temperature. The outside temperature varies in a periodic sinusoidal pattern about its mean. Solution by complex representation of the temperature and use of matrix methods allows treatment of multiple homogeneous wall layers. This analytical model of a single wall is used to represent the external wall of the test zone.

Zone Description

The test zone geometry is cubic with internal dimensions 3×3×3m. No windows are present in this test. Only one surface is to be external. This external surface may be selected to be any of the six surfaces of the zone, as any surface should have the same response. In other words the user may wish to test the response of the roof or a suspended floor in the same way as the walls. The external surface to be tested may have multiple layers. The inside dry-bulb temperature should be fixed at the mean external temperature. (A difference in temperature between inside and the mean outside temperature can be dealt with by adding a steady state temperature difference and load to this analytical solution).

The effects of solar irradiation, long wave radiation, infiltration and internal gains must be eliminated in this test. This can normally be achieved by setting suitable surface properties and setting infiltration rate and internal gains to zero. Zone orientation and

other location parameters are also irrelevant in this test. The type of analytical solution used in this test requires that only fixed convection coefficients be used.

Test Parameters

The driving external temperature has a sinusoidal form of constant amplitude throughout the year. The user is able to choose the datum temperature, the amplitude and period of the outside dry bulb temperature cycle when generating the weather files and the analytical results using the toolkit software. The variation in outside dry bulb temperature can be expressed as,

$$T_o = T_1 + T_{amp} \sin\left(\frac{2\pi t}{P}\right) \quad (\text{TC3-1})$$

where T_1 and T_{amp} are the mean temperature and amplitude respectively, P is the period of the fluctuation (hours) and t is the time (hours)⁶. This function is illustrated in Figure TC3-1. The mean temperature and amplitude are to be specified by the user of the test along with the period of the fluctuations. The inside dry bulb temperature of the zone must be set to the same value as the mean outside dry bulb temperature. The controls should be set to keep the inside dry bulb temperature constant throughout the test.

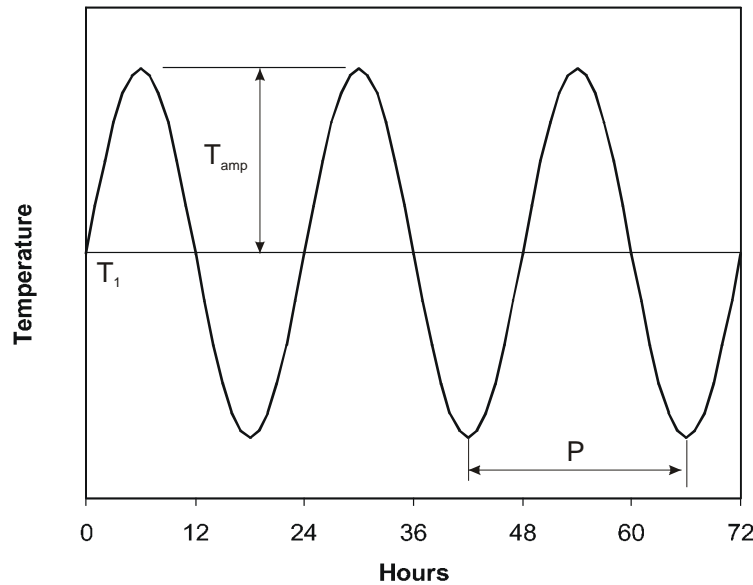


Figure TC3-1: Profile of the driving external dry bulb temperature

Although the analytical solution in principle is able to deal with multiple layers, the software interface allows the specification of three layers. The user, when calculating the response, is able to set the thermal properties of the wall layers (conductivity, density and

⁶ The period is normally taken as 24 hours for load calculation purposes.

specific heat capacity) in addition to thickness. Convection coefficients have to be specified for the inside and external surfaces. These coefficients have to be taken as constant coefficients. Although the user is able to specify any Period for the driving temperature fluctuation (not just integer values) it is probably not meaningful to use very short periods (e.g. six hours or less) as the hourly temperatures in the weather files will not meaningfully represent a sine wave pattern.

The input screen of the toolkit software is illustrated in Figure TC3-2 below.

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCoupl WinReveal
SSConv SSCond TC 1 TC 2 **TC 3** ExtSolRad SolRadGlazing SolRadShade

Test Temperatures

Mean temperature (°C)

Temperature amplitude (°C)

Period of fluctuation (hours)

Convection Coefficients

Outside coefficient (W/m².K)

Inside coefficient (W/m².K)

Fabric Properties

Number of layers	Layer Number	Conductivity (W/m.K)	Density (kg/m³)	Specific heat capacity (J/kg.K)	Thickness (m)
<input type="text" value="3"/>	1	0.14	700.0	500.0	0.1
	2	0.1	50.0	200.0	0.05
	3	0.2	500.0	800.0	0.1

Status: TMY2 SI Units

Figure TC3-2: The input screen for the toolkit software for test TC3. Note that Layer number one is inside.

Test Output

The principle data of interest in this test are the predicted internal load. These data are listed in the analytical response data file. This file consists of three columns of hourly values for the whole year as follows:

1. hour number,
2. external dry bulb temperature,
3. internal load

The fluctuation in internal load should show a steady periodic pattern. To this end the peak loads and temperatures should be the same for each month of the year. Presuming that steady periodic results are obtained, it should only be necessary to make hourly comparisons of the calculated loads and the analytical solution for one period. The error in the prediction of the phase lag between the outside dry bulb fluctuation and the internal load should be determined from the hourly results.

Test Results

The tabulated results included below have been produced with the following test parameters shown in Table TC3-A. The external convection coefficient is that given by the relationship published by Yazdanian and Klems (1994) with zero air speed and 10K temperature difference. The internal convection coefficient is that reported by McQuiston, Parker and Spitler (2000) Table 8-8. The analytical solution for these parameters is also plotted in Figure TC3-2 and TC3-3.

Table TC3-A
Test parameters used in generating the tabulated results
(Layer 1 is inside, layer 3 is outside)

Test Parameter	Value			Units
	Layer 1	Layer 2	Layer 3	
Thermal conductivity	0.14	0.1	0.2	W/m.K
Density	700	50	500	Kg/m ³
Specific heat capacity	500	200	800	J/kg.K
Thickness	0.1	0.05	0.1	m
Mean temperature (T_l)	20.0			°C
Temperature amplitude (T_{amp})	15.0			°C
External convection coefficient	1.81			W/m ² .K
Internal convection coefficient	3.18			W/m ² .K

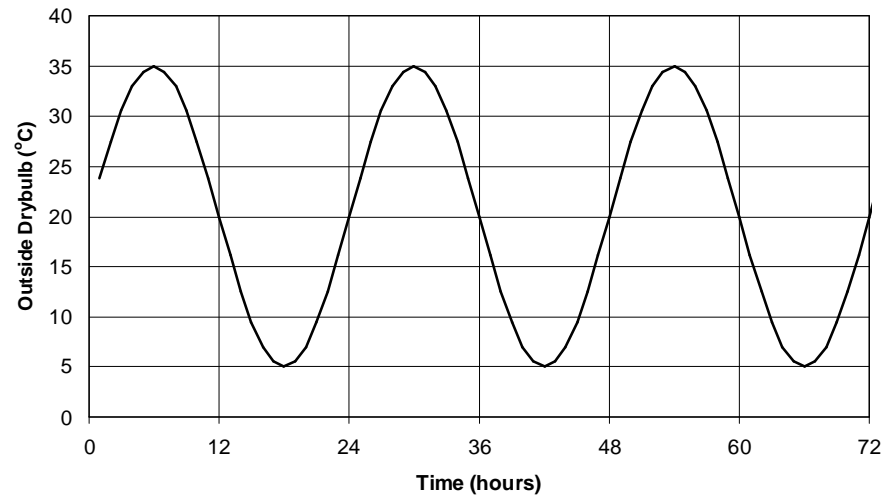


Figure TC3-2: The variation in external temperture for the parameters in Table TC3-A.

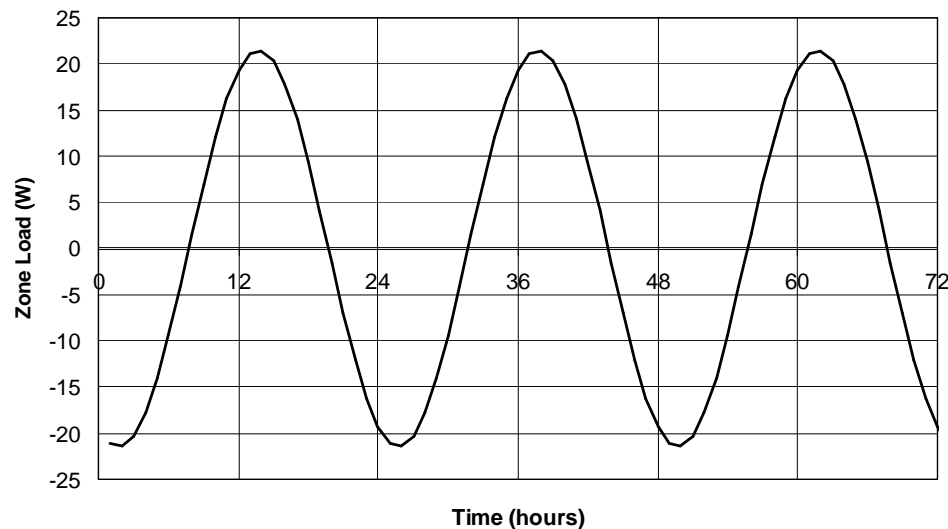


Figure TC3-3: The analytical solution of the zone load for the parameters in Table TC3-A.

The tabulated results of the analytical solution using the parameters in Table TC3-A for the first 36 hours are given below in Table TC3-B.

Table TC3-B

Tabulated results of the analytical solution calculated using the parameters shown in Table TC3-A

Hour	Outside Drybulb Temperature (°C)	Zone Load (W)
1	23.8823	-21.11
2	27.5000	-21.45
3	30.6066	-20.33
4	32.9904	-17.82
5	34.4889	-14.10
6	35.0000	-9.42
7	34.4889	-4.09
8	32.9904	1.51
9	30.6066	7.01
10	27.5000	12.03
11	23.8823	16.24
12	20.0000	19.33
13	16.1177	21.11
14	12.5000	21.45
15	9.3934	20.33
16	7.0096	17.82
17	5.5111	14.10
18	5.0000	9.42
19	5.5111	4.09
20	7.0096	-1.51
21	9.3934	-7.01
22	12.5000	-12.03
23	16.1177	-16.24
24	20.0000	-19.33
25	23.8823	-21.11
26	27.5000	-21.45
27	30.6066	-20.33
28	32.9904	-17.82
29	34.4889	-14.10
30	35.0000	-9.42
31	34.4889	-4.09
32	32.9904	1.51
33	30.6066	7.01
34	27.5000	12.03
35	23.8823	16.24
36	20.0000	19.33

Analytical Solution

The analytical solution used in this test is based on the one-dimensional unsteady heat conduction equation that can be expressed as,

$$\frac{d^2T(x,t)}{dx^2} = \frac{1}{\alpha} \frac{dT(x,t)}{dt} \quad (\text{TC3-1})$$

where the temperature T is a function of distance x and time t . Heat flux at any position x and time t is given by,

$$q(x,t) = -k \frac{dT(x,t)}{dx} \quad (\text{TC3-2})$$

The solution for a periodic driving function can be obtained by first taking the Laplace transforms of these equations, to give,

$$\frac{d^2T(x,s)}{dx^2} = \frac{1}{\alpha} sT(x,s) \quad (\text{TC3-3})$$

and,

$$q(x,s) = -k \frac{dT(x,s)}{dx} \quad (\text{TC3-4})$$

These equations have general solutions in the ‘s’ domain of,

$$T(x,s) = A \cosh(x\sqrt{s/\alpha}) + B \sinh(x\sqrt{s/\alpha}) \quad (\text{TC3-5})$$

$$q(x,s) = -k\sqrt{s/\alpha} A \cosh(x\sqrt{s/\alpha}) - k\sqrt{s/\alpha} B \sinh(x\sqrt{s/\alpha}) \quad (\text{TC3-6})$$

where A and B depend on the specific boundary conditions (Hittle 1992). If we are only interested in the surface temperatures and fluxes (T_1 and q_1 at $x=0$ and T_2 and q_2 at $x=l$) equations TC3-5 and TC3-6 are simplified to,

$$T_1(s) = \cosh(l\sqrt{s/\alpha})T_2(s) + \frac{1}{k\sqrt{s/\alpha}} \sinh(x\sqrt{s/\alpha})q_2(s) \quad (\text{TC3-7})$$

$$q_1(s) = k\sqrt{s/\alpha} \sinh(l\sqrt{s/\alpha})T_2(s) + \cosh(l\sqrt{s/\alpha})q_2(s) \quad (\text{TC3-8})$$

The solution to these equations for sinusoidal boundary conditions results in complex temperatures and fluxes in the real time domain. The boundary conditions are of the form $A \sin(\omega t)$ in the real domain and become $\text{Im}(Ae^{i\omega t})$ where ‘Im’ is the imaginary part (Pipes 1957). The equations for the complex temperatures (\hat{T}) and fluxes (\hat{q}) are then (Holmes and Wilson 1996),

$$\hat{T}_1(t) = \cosh(p + ip)\hat{T}_2(t) + \frac{l}{k(p + ip)}\sinh(p + ip)\hat{q}_2(t) \quad (\text{TC3-9})$$

$$\hat{q}_1(t) = \frac{k(p + ip)}{l}\sinh(p + ip)\hat{T}_2(t) + \cosh(p + ip)\hat{q}_2(t) \quad (\text{TC3-10})$$

where,

$$p = \left(\frac{\pi l^2}{P^* \alpha} \right)^{0.5} \quad (\text{TC3-11})$$

and P^* is the period in the appropriate units of time (i.e. seconds for SI, hours for IP). This can be simply expressed in matrix form as,

$$\begin{bmatrix} \hat{T}_1 \\ \hat{q}_1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_1 \end{bmatrix} \begin{bmatrix} \hat{T}_2 \\ \hat{q}_2 \end{bmatrix} \quad (\text{TC3-12})$$

The matrix of complex constants has some interesting properties. For an air gap or surface layer, where the diffusivity is high, the matrix elements become: $m_1=1$, $m_2=R_a$ or R_s and $m_3=0$. Also, for multiple layers \hat{T}_2 and \hat{q}_2 represent the interface temperatures and fluxes and matrices for each layer can be multiplied using the usual matrix rules so that for a multi-layer wall the solution from the inside air to outside air is,

$$\begin{bmatrix} \hat{T}_i \\ \hat{q}_i \end{bmatrix} = \begin{bmatrix} 1 & R_{Si} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} m_1^1 & m_2^1 \\ m_3^1 & m_1^1 \end{bmatrix} \begin{bmatrix} m_1^2 & m_2^2 \\ m_3^2 & m_1^2 \end{bmatrix} \dots \begin{bmatrix} m_1^n & m_2^n \\ m_3^n & m_1^n \end{bmatrix} \begin{bmatrix} 1 & R_{So} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{T}_o \\ \hat{q}_o \end{bmatrix} \quad (\text{TC3-13})$$

or, combining the constant matrices together,

$$\begin{bmatrix} \hat{T}_1 \\ \hat{q}_1 \end{bmatrix} = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \begin{bmatrix} \hat{T}_2 \\ \hat{q}_2 \end{bmatrix} \quad (\text{TC3-14})$$

For this analytic test we are concerned with the case where the inside temperature is zero and the outside varies sinusoidally ($T_o = \Delta T \sin(\omega t)$) and we desire to find $q_i(t)$. For a single layer under these conditions,

$$\begin{aligned} 0 &= m_1 \hat{T}_o + m_2 \hat{q}_2 \\ \hat{q}_1 &= m_3 \hat{T}_o + m_1 \hat{q}_2 \end{aligned} \quad (\text{TC3-15})$$

substitution gives,

$$\hat{q}_1 = \left(m_3 + \frac{m_1^2}{m_2} \right) \hat{T}_o \quad (\text{TC3-15})$$

using the definitions of m_1 – m_3 in equations TC3-9 and TC3-10 leads to some simplification,

$$\hat{q}_1(t) = \left[\frac{k(p+ip)}{l} \sinh(p+ip) - \frac{k(p+ip)}{l \sinh(p+ip)} \cosh^2(p+ip) \right] \hat{T}_2(t) \quad (\text{TC3-16})$$

$$\hat{q}_1(t) = \frac{k(p+ip)}{l \sinh(p+ip)} [\sinh^2(p+ip) - \cosh^2(p+ip)] \hat{T}_2(t) \quad (\text{TC3-17})$$

$$\hat{q}_1(t) = \frac{-k(p+ip)}{l \sinh(p+ip)} \hat{T}_2(t) \quad (\text{TC3-18})$$

which can be seen to be the same as

$$\hat{q}_1(t) = \frac{-l^2}{k^2} \frac{1}{m_2} \hat{T}_2(t) \quad (\text{TC3-19})$$

More generally for a multi layer wall,

$$\hat{q}_i(t) = U \frac{1}{UM_2} \hat{T}_o(t) \quad (\text{TC3-20})$$

To avoid complex quantities we take the modulus of $1/UM_2$ and introduce an associated time lag. This quantity is commonly known as the Decrement factor. The expression for the inside flux due to an external sinusoidal air temperature variation with zero inside temperature can then be expressed as,

$$q_i(t) = Uf\Delta T_o \sin(\omega(t - \phi)) \quad (\text{TC3-21})$$

where ΔT_o is the amplitude of the outside air temperature variation and the decrement factor f is given by,

$$f = \left| \frac{1}{UM_2} \right| \quad (\text{TC3-22})$$

and the time lag (hours) associated with the decrement factor for a period P of 24 hours is,

$$\phi = \frac{12}{\pi} \arctan \left(\frac{\text{Im}(1/UM_2)}{\text{Re}(1/UM_2)} \right) \quad (\text{TC3-23})$$

where the arctangent is evaluated in the range $-\pi$ to 0 .

References

Hittle, D.C. 1992. Response factors and conduction transfer functions. Lecture notes.

Holmes, M.J. and A. Wilson, 1996. Revised Sections A5, A8, A9 of the CIBSE Guide : Second Draft. London: Ove Arup and Partners, p. 148.

- Pipes, L.A. 1957. Matrix analysis of heat transfer problems. Journal of the Franklin Institute. Vol. 263, No. 3, pp. 195-206.
- Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.
- McQuiston, F.C., J.D., Parker, and J.D. Spitler, 2000. *Heating, Ventilating, and Air Conditioning Analysis and Design*, Fifth Edition. John Wiley and Sons, New York

Test ExtSolRad: Exterior Solar Radiation - Opaque Surfaces

Objective

The objective of this test is to find the response to different solar beam irradiation on opaque surfaces at different hours of a single day. The calculation of incidence angle and the treatment of solar absorption of beam irradiation on opaque surface for a given location and time are tested. The opaque surface has no thermal mass. The analytical solution of the heat flux (interior or exterior) or zone load is compared with the results of the program to be tested for the selected day.

Analytical Model

Incident direct solar fluxes are calculated using tabulated values of solar position and standard geometric relations. The exposed building fabric is assumed to have zero thermal mass so that an instantaneous heat balance calculation can be made to find the one-dimensional heat conduction and the convection at inside and outside surfaces. Long wave radiation is assumed to be zero at all surfaces.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. No windows are present in this test. Only one surface is to be external. This external surface may be selected to be any of the six surfaces of the zone except for the floor. The external surface to be tested can have a number of material layers which should have no thermal mass.

The effects of long wave radiation, infiltration and internal heat gains must be eliminated in this test. Except for the external surface, all other surfaces of the cube are adiabatic and have no thermal mass. The interior and exterior air temperature have to be taken as fixed. The interior and exterior convection coefficients may be correlations of surface and air temperature.

Test parameters

The different solar beam irradiation in this test is calculated with solar position data from the US Naval Observatory (USNO 2001). The weather files contain only direct normal radiation at ground level. The user is able to choose the location of the test zone (from a list of four: Atlanta (W 84° 25', N33° 46'), Boston (W 71° 05', N42° 19'), Chicago (W 87° 41', N41° 51') and Los Angeles (W118° 22', N34° 05')) and two test dates (June 21st, 1999 or August 21st, 1999) when generating the weather files using the toolkit software. The user is also able to specify the surface tilt angle, the surface azimuth and the solar

absorption of the surface when generating the analytical results. Different inside and outside dry bulb temperatures may also be set when using the software.

The user, when calculating the response, has to specify the total number of fabric layers and the thermal conductivity and thickness of each layer. The inside and outside air temperatures have to be specified and taken as fixed. Also, convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed) should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{ExtSolRad-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations. The test suite software input screen for this test is shown in Figure ExtSolRad-1.

Test output

The principle data of interest in this test are the predicted internal and external surface temperature, the predicted solar flux and the cooling load of the test zone. These data are listed in the output file along with the input data. It should be noted that the solar position data used to calculate the analytical solution is recorded at ten-minute intervals rather than hourly. This is reflected in the sun up and sundown times shown in some of the results.

Test results

Two sets of test results are provided below. These are for a south facing and west facing vertical surface. The tabulated results in table ExtSolRad-B have been produced with the following test parameters shown in table ExtSolRad-A where the exterior surface is south facing. Even though the surface has no thermal mass the zone load (internal flux) is not directly proportional to the incident flux. This can be seen at the beginning and end of the day in the results shown in Figures ExtSolRad-2 and ExtSolRad-3, and is due to the non-linear nature of the convection correlations.

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCoup
SSConv SSCond TC 1 TC 2 TC 3 ExtSolRad SolRadGlazing SolRadShade

Test Temperatures

Inside temperature 20.0 (°C)
Outside temperature 20.0 (°C)

Test Location Date

Location Atlanta
Date Aug 21st

Convection Coefficients

	Coefficient 'A'	Coefficient 'C'	Coefficient 'n'	
Inside correlation	0.0	1.49	0.345	(W/m ² .K)
Outside correlation	0.0	0.84	0.333	(W/m ² .K)

Fabric Properties

Layer Number	Thickness (m)	Conductivity (W/m.K)
1	0.1	1.15
2	0.1	1.05
3	0.1	1.15

Number of layers 3

Surface Properties

Surface tilt angle 90 (degree)
Surface azimuth 180 (degree)
Solar absorptivity 0.4

Write

Status: TMY2 SI Units

Figure ExtSolRad-1: Test suite software parameter input screen for test ExtSolRad.

Table ExtsolRad-A
Test parameters used in generating the tabulated results
(South facing surface)*

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	180	Degree
Solar absorption of the surface ' α '	0.4	-
Number of layers: ' N '	3	-
Thermal conductivity: layer 1 ' K_1 '	1.15	W/mK
Thickness: layer 1 ' L_1 '	0.1	M
Thermal conductivity: layer 2 ' K_2 '	1.05	W/mK
Thickness: layer 2 ' L_2 '	0.1	M
Thermal conductivity: layer 3 ' K_3 '	1.15	W/mK
Thickness: layer 3 ' L_3 '	0.1	M
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

* Ordering of the surfaces is nominally given by No. 1 being outside, but is unimportant in the analytical solution which is quasi-steadystate.

Table ExtsolRad-B
Tabulated results for the exterior solar radiation test (south facing surface)
*Zone load = (Internal surface heat flux) x (South facing surface area)

Time	Internal surface temperature (C)	External surface temperature (C)	Incident solar flux (W/m ²)	Internal surface heat flux (W/m ²)*
6:10	20.0000	20.0000	0.0000	0.0000
...				
7:50	20.0000	20.0000	0.0000	0.0000
8:00	20.9678	21.3516	6.7023	1.4258
8:10	22.3063	23.5403	22.7881	4.5847
8:20	23.3771	25.4381	39.2142	7.6573
8:30	24.3761	27.2965	56.8256	10.8504

8:40	25.2704	29.0207	74.2398	13.9338
8:50	26.0786	30.6222	91.1987	16.8811
9:00	26.9116	32.3119	109.8119	20.0641
9:10	27.6146	33.7663	126.3650	22.8560
9:20	28.3374	35.2871	144.1547	25.8209
9:30	28.9835	36.6672	160.6917	28.5477
9:40	29.5942	37.9885	176.8540	31.1880
9:50	30.1644	39.2365	192.3992	33.7063
10:00	30.7002	40.4213	207.3946	36.1175
10:10	31.2075	41.5534	221.9318	38.4392
10:20	31.6844	42.6268	235.8934	40.6552
10:30	32.1146	43.6023	248.7289	42.6812
10:40	32.5438	44.5822	261.7586	44.7274
10:50	32.8949	45.3888	272.5820	46.4195
11:00	33.2526	46.2147	283.7557	48.1593
11:10	33.5373	46.8753	292.7578	49.5559
11:20	33.8120	47.5153	301.5324	50.9131
11:30	34.0607	48.0971	309.5538	52.1504
11:40	34.2715	48.5916	316.4054	53.2046
11:50	34.4690	49.0561	322.8687	54.1969
12:00	34.6027	49.3714	327.2707	54.8716
12:10	34.7205	49.6497	331.1653	55.4677
12:20	34.8013	49.8410	333.8488	55.8781
12:30	34.8418	49.9367	335.1922	56.0834
12:40	34.8763	50.0184	336.3410	56.2589
12:50	34.8621	49.9849	335.8698	56.1869
13:00	34.8005	49.8389	333.8200	55.8737
13:10	34.7414	49.6992	331.8597	55.5740
13:20	34.6075	49.3828	327.4299	54.8960
13:30	34.4783	49.0781	323.1753	54.2439
13:40	34.3018	48.6628	317.3948	53.3566
13:50	34.1125	48.2184	311.2319	52.4088
14:00	33.8690	47.6485	303.3651	51.1961
14:10	33.6135	47.0527	295.1848	49.9317
14:20	33.3043	46.3345	285.3836	48.4121
14:30	32.9874	45.6019	275.4568	46.8677
14:40	32.6096	44.7329	263.7741	45.0430
14:50	32.2243	43.8521	252.0376	43.2018
15:00	31.7845	42.8531	238.8589	41.1242
15:10	31.3371	41.8442	225.6971	39.0382
15:20	30.8335	40.7179	211.1840	36.7242
15:30	30.2882	39.5093	195.8312	34.2597
15:40	29.7227	38.2685	180.3196	31.7511
15:50	29.1264	36.9749	164.4282	29.1602
16:00	28.4948	35.6217	148.1302	26.4789

16:10	27.7912	34.1355	130.6403	23.5717
16:20	27.0514	32.5991	113.0429	20.6118
16:30	26.2860	31.0393	95.7282	17.6603
16:40	25.4390	29.3516	77.6850	14.5366
16:50	24.5621	27.6506	60.3242	11.4751
17:00	23.6522	25.9422	43.8586	8.5080
17:10	22.5487	23.9603	26.2627	5.2444
17:20	21.2951	21.8629	10.0872	2.1098
17:30	20.0000	20.0000	0.0000	0.0000
...				
19:10	20.0000	20.0000	0.0000	0.0000

Figure ExtSolRad-2 shows the plotted results of the incident solar flux vs. time and the internal surface heat flux vs. time using the data listed in table ExtSolRad-B.

**Comparison of incident solar flux and internal surface heat flux:
South facing wall**

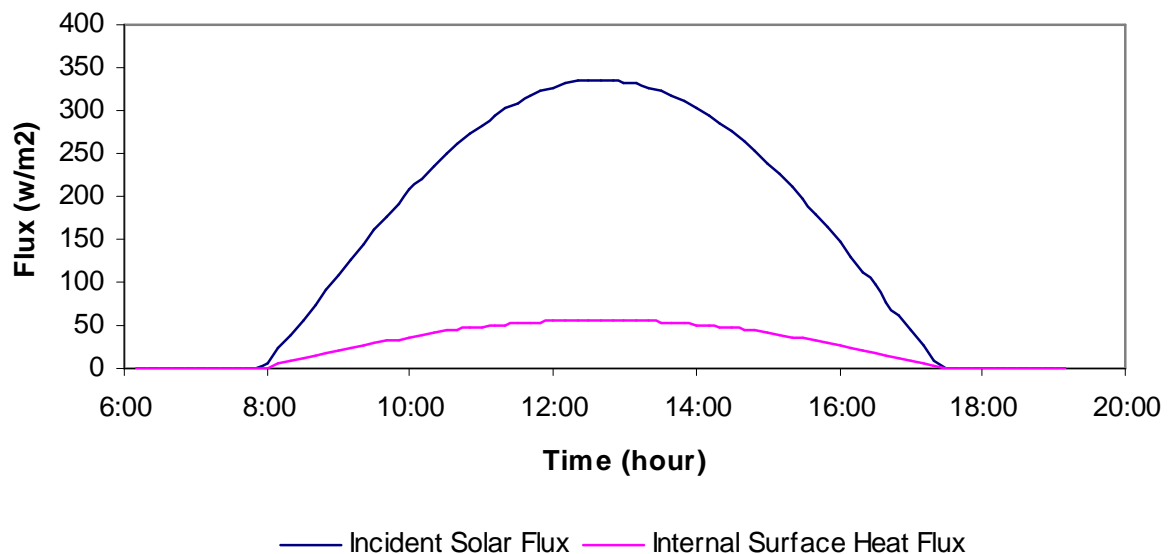


Figure ExtSolRad-2: Solar flux and internal heat flux calculated with parameters shown in Table ExtSolRad-A (south facing surface). Zone load = (Internal surface heat flux) x (South facing surface area).

The tabulated results included below in table ExtSolRad-D have been produced with the following test parameters shown in table ExtSolRad-C. In this case the surface is west facing and vertical.

Table ExtsolRad-C
Test parameters used in generating the tabulated results
(West facing surface)*

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	270	Degree
Solar absorption of the surface ' α '	0.4	-
Number of layers: ' N '	3	-
Thermal conductivity: layer 1 ' K_1 '	1.15	W/mK
Thickness: layer 1 ' L_1 '	0.1	m
Thermal conductivity: layer 2 ' K_2 '	1.05	W/mK
Thickness: layer 2 ' L_2 '	0.1	m
Thermal conductivity: layer 3 ' K_3 '	1.15	W/mK
Thickness: layer 3 ' L_3 '	0.1	m
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

* Ordering of the surfaces is nominally given by No. 1 being outside, but is unimportant in the analytical solution which is quasi-steadystate.

Table ExtsolRad-D
Tabulated results for the exterior solar radiation test (west facing surface)
*Zone load = (Internal surface heat flux) x (west facing surface area)

Time	Internal surface temperature (C)	External surface temperature (C)	Incident solar flux (W/m ²)	Internal surface heat flux (W/m ²)*
6:10	20.0000	20.0000	0.0000	0.0000
...				
12:40	20.0000	20.0000	0.0000	0.0000
12:50	23.1741	25.0703	35.8942	7.0448
13:00	25.2890	29.0571	74.6176	14.0000
13:10	27.0484	32.5930	112.9743	20.6002
13:20	28.5920	35.8287	150.6010	26.8871

13:30	30.0355	38.9533	188.8484	33.1328
13:40	31.3324	41.8337	225.5608	39.0165
13:50	32.5544	44.6064	262.0816	44.7780
14:00	33.7053	47.2666	298.1161	50.3852
14:10	34.7600	49.7431	332.4746	55.6680
14:20	35.7713	52.1514	366.5913	60.8583
14:30	36.7114	54.4180	399.2974	65.7868
14:40	37.5825	56.5416	430.4401	70.4401
14:50	38.3993	58.5524	460.3560	74.8763
15:00	39.1840	60.5014	489.7332	79.2024
15:10	39.8876	62.2632	516.6002	83.1342
15:20	40.5327	63.8899	541.6590	86.7810
15:30	41.1376	65.4250	565.5267	90.2373
15:40	41.6530	66.7402	586.1379	93.2088
15:50	42.1240	67.9480	605.1963	95.9462
16:00	42.5298	68.9930	621.7854	98.3209
16:10	42.8850	69.9109	636.4298	100.4114
16:20	43.1566	70.6147	647.7062	102.0173
16:30	43.3599	71.1427	656.1928	103.2239
16:40	43.4818	71.4599	661.3007	103.9492
16:50	43.5188	71.5561	662.8513	104.1693
17:00	43.4585	71.3992	660.3230	103.8104
17:10	43.3047	70.9992	653.8844	102.8959
17:20	43.0221	70.2660	642.1140	101.2213
17:30	42.5991	69.1717	624.6300	98.7274
17:40	42.0087	67.6520	600.5139	95.2745
17:50	41.2580	65.7317	570.3197	90.9293
18:00	40.2215	63.1039	529.5215	85.0170
18:10	38.8602	59.6952	477.5370	77.4099
18:20	37.1582	55.5043	415.1701	68.1631
18:30	34.7861	49.8049	333.3423	55.8007
18:40	31.7515	42.7785	237.8811	40.9697
18:50	27.6982	33.9409	128.3832	23.1941
19:00	23.0117	24.7786	33.3047	6.5644
19:10	20.0396	20.0448	0.0819	0.0194

Figure ExtSolRad-3 shows the plotted results of the incident solar flux vs. time and the internal surface heat flux vs. time using the data listed in table ExtSolRad-D.

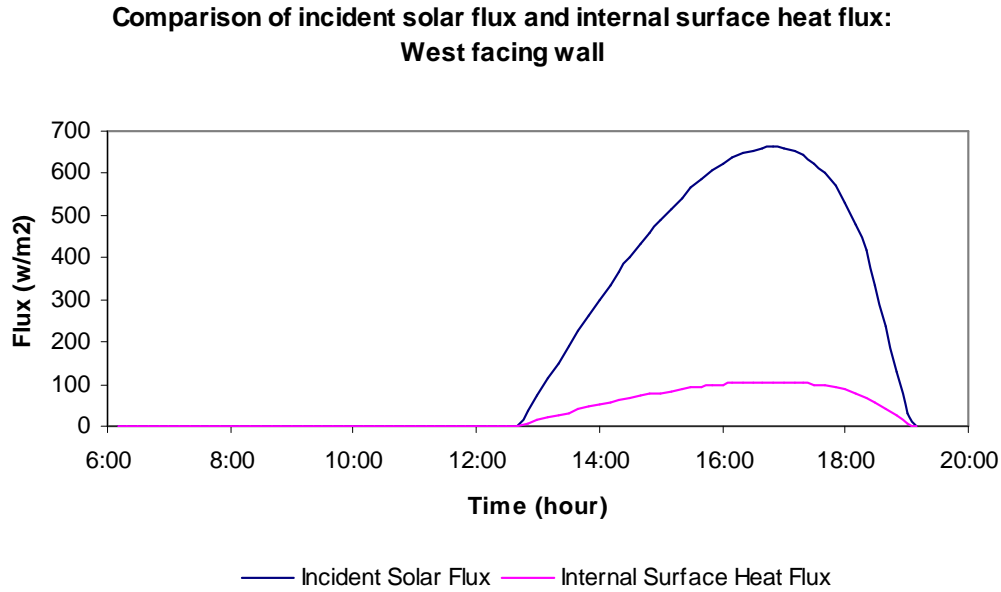


Figure ExtSolRad-3: Solar flux and internal heat flux calculated with parameters shown in Table ExtSolRad-C (west facing surface). Zone load = (Internal Surface Heat Flux) x (West Facing Surface Area).

Analytical solution

As the wall/roof construction has no thermal mass the analytical solution can be based on a steady state heat balance. The heat balance must include the direct solar irradiation but assumes no long wave radiation.

1. Solar beam irradiation on opaque surface

Given:

The solar beam irradiation at normal incidence I_n , W/m^2

Solar elevation (or altitude) β

Then the solar beam irradiation on the surface is:

$$I_{bs} = \text{Max}(\cos \theta, 0) I_n \quad (\text{ExtSolRad-2})$$

where,

I_{bs} = the beam radiation on the receiving plane of slope s , W/m^2

θ = the angle of incidence on the surface

while $\cos \theta$ is calculated (Fisher 1999) as:

$$\cos \theta = a_1 a_2 + b_1 b_2 + c_1 c_2 \quad (\text{ExtSolRad-3})$$

and

$$\begin{aligned} a_1 &= \sin \phi \cos \beta & b_1 &= \cos \phi \cos \beta & c_1 &= \sin \beta \\ a_2 &= \sin \psi \sin \varepsilon & b_2 &= \cos \psi \sin \varepsilon & c_2 &= \cos \varepsilon \end{aligned}$$

where,

ϕ = solar azimuth angle
 ψ = surface azimuth angle
 ε = the surface tilt angle

2. External surface heat balance

If the diffuse solar irradiation is zero, so that the total solar irradiation incident on the surface is due to incident beam irradiation $I_{b,s}$ (W/m²) and if the long wave radiation is also zero, then the exterior surface heat balance can be expressed as:

$$q''_{c,out} = \alpha I_{bs} + h_{c,o} (T_{a,o} - T_{s,o}) \quad (\text{ExtSolRad-4})$$

where,

$q''_{c,out}$ = external conduction heat flux, W/m²
 α = solar absorption of the surface, dimensionless
 I_{bs} = the beam radiation on the surface, W/m²
 $h_{c,o}$ = the exterior convection coefficient, W/m²K
 $T_{a,o}$ = external air temperature, C
 $T_{s,o}$ = external surface temperature, C

3. Internal surface heat balance

No window is present in the test zone and long wave radiation is eliminated, thus the internal heat balance is:

$$q''_{c,in} = h_{c,i} (T_{s,i} - T_{a,i}) \quad (\text{ExtSolRad-5})$$

where,

$q''_{c,in}$ = internal conduction heat flux, W/m²
 $h_{c,i}$ = interior convection coefficient, W/m²K
 $T_{s,i}$ = internal surface temperature, C
 $T_{a,i}$ = internal air temperature, C

4. The conduction heat transfer

As the wall construction has no thermal mass the conduction heat flux can be found from the total thermal resistance and the overall temperature difference of the fabric:

$$q_{c,out}'' = (T_{s,o} - T_{s,i}) / R_f \quad (\text{ExtSolRad-6})$$

where,

$q_{c,out}''$ = the conduction heat flux of the fabric, W/m²

$T_{s,o}$ = external surface temperature, C

$T_{s,i}$ = internal surface temperature, C

R_f = the total thermal resistance of the fabric, Km²/W

and

$$R_f = \sum_{j=1}^N L_j / K_j \quad (\text{ExtSolRad-7})$$

where,

L_j = the thickness of the jth layer, m

K_j = the thermal conductivity of the jth layer, W/m²K

N = the number of the layers

Due to the absence of thermal mass the conduction and surface fluxes can be equated, thus

$$q_c'' = q_{c,out}'' = q_{c,in}'' \quad (\text{ExtSolRad-8})$$

Using equations (ExtSolRad-5)~ (ExtSolRad-9) and solving for q_c'' ,

$$q_c'' = \frac{(T_{a,o} - T_{a,i}) + \alpha I_{bs} / h_{c,o}}{R_{c,o} + R_f + R_{c,i}} \quad (\text{ExtSolRad-9})$$

where,

$$R_{c,o} = 1 / h_{c,o} \quad (\text{ExtSolRad-10})$$

$$R_{c,i} = 1 / h_{c,i} \quad (\text{ExtSolRad-11})$$

The zone load is simply the heat flux q_c'' multiplied by the relevant surface area (9.0 m²). The external and internal surface temperature is finally calculated using the flux as follows:

$$T_{s,o} = T_{a,o} - (q_c'' - \alpha I_{bs}) / h_{c,o} \quad (\text{ExtSolRad-12})$$

$$T_{s,i} = T_{a,i} + q_c'' / h_{c,i} \quad (\text{ExtSolRad-13})$$

References

- Duffie J.A., W.A. Beckman 1991. *Solar Engineering of thermal Processes*. New York: Wiley.
- Fisher D. 1999. *ASHRAE Loads Toolkit Documentation (Draft)*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Atlanta, GA.
- McQuiston F.C., J.D. Parker, J.D. Spitler 2000. *Heating, Ventilating, and Air Conditioning: Analysis and Design*. New York: Wiley.
- USNO 2001. U.S. Naval Observatory: web site - <http://aa.usno.navy.mil/data/docs/AltAz.html>
- Walton, G. N. 1981. *Passive Solar Extension of the Building Loads Analysis and System Thermodynamics (BLAST) program*. Technical Report, United States Army Construction Engineering Research Laboratory, Champaign, IL.
- Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test SolRadGlazing: Solar Radiation – Glazed Surfaces

Objective

The objective of this test is to find the response to solar beam irradiation on glazed surfaces at different hours of a single day. The calculation of incidence angle, the treatment of solar absorption and transmission of beam irradiation on glazed surface for a given location and time are also tested. The analytical solution of the heat flux or zone load is compared with the results of the program to be tested for the selected day.

Analytical Model

Calculation of incident direct solar radiation is calculated from tabulated solar position data in the same way as test ExtSolRad. Transmission and absorption of the direct solar radiation is calculated using optical relations for a clear single pane of glazing. It is assumed that the glazed surface has no thermal mass and its conductivity is very high so that it will have a uniform temperature. Conduction through and convection from the window surface is calculated from an instantaneous heat balance. Diffuse radiation is eliminated from the test by setting it to zero in the weather file. It is assumed that there is no long wave radiation at any surface.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. Only one surface is to be external and is to be entirely glazed. This external glazed surface may be selected to be any of the six surfaces of the zone except for the floor. The window system is very simple, being of single pane construction without any frame or reveal. The test does not deal with glazings with special coatings of any kind.

The effects of long wave radiation, infiltration and internal heat gains must be eliminated in this test. Except for the external surface, all other surfaces of the cube are black (i.e. emissivity of one, to eliminate cavity albedo), adiabatic and have no thermal mass. The interior and exterior air temperature have to be taken as fixed. The interior and exterior convection coefficients can be specified in terms of a temperature dependent correlation.

Test parameters

The solar beam irradiation in this test is calculated with solar position data from the US Naval Observatory (USNO 2001) and the beam irradiation at normal incidence from the weather data. The user is able to choose the location of the test zone (from a list of four: Atlanta (W 84° 25', N33° 46'), Boston (W 71° 05', N42° 19'), Chicago (W 87° 41', N41° 51') and Los Angeles (W118° 22', N34° 05')) and two test dates (June 21st, 1999 or August 21st,

1999) when generating the weather files using the toolkit software. The other parameters to be chosen are the surface tilt angle, the surface azimuth, the thickness and extinction coefficient of the surface and the refractive index of the glazed surface. In programs that do not allow input of refractive index and extinction coefficient directly equivalent window model parameters must be selected. This can be done most easily for standard clear single pane glazing.

The inside and outside air temperatures have to be specified as the same and taken as fixed. The same value of the inside air temperature must also be used in the input data of the program to be tested. Also, convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed) should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{SolRadGlazing-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations.

Test output

The principle data of interest in this test are the predicted surface temperature, the predicted solar flux, the predicted total heat flux into the zone and the cooling load of the test zone. These data are listed in the output file along with the input data. It should be noted that the solar position data used to calculate the analytical solution is recorded at ten-minute intervals rather than hourly. This is reflected in the sun up and sundown times shown in some of the results. It should be noted that the weather file formats used by the test suite software record solar irradiances in terms of the accumulated flux received in the previous hour.

The screenshot shows the ASHRAE Analytical Test Toolkit (1052-RP) window. The 'SolRadGlazing' tab is selected. The interface includes the following sections:

- Test Temperatures:**
 - Inside temperature: 20.0 (°C)
 - Outside temperature: 20.0 (°C)
- Test Location Date:**
 - Location: Atlanta
 - Date: June 21st
- Convection Coefficients:**

	Coefficient 'A'	Coefficient 'C'	Coefficient 'h'	
Inside correlation	0.0	1.49	0.345	(W/m ² .K)
Outside correlation	0.0	0.84	0.333	(W/m ² .K)
- Surface Properties:**
 - Surface tilt angle: 90 (degree)
 - Surface azimuth: 180 (degree)
 - Thickness: 0.0023 (m)
 - Extinction coefficient: 10.0 (m⁻¹)
 - Refractive index: 1.526

A 'Start Calculation' button is located at the bottom center. The status bar at the bottom shows 'Status: TMY2 SI Units'.

Figure SolRadGlazing-1: Test suite software parameter input screen for test SolRadGlazing.

Test results

Two sets of test results are provided below. These are for a south facing and west facing glazed surface. The tabulated results included below in table SolRadGlazing-B have been produced with the following test parameters shown in table SolRadGlazing-A.

Table SolRadGlazing-A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ϵ '	90	Degree
Surface azimuth ' ψ '	180	Degree
Thickness of the surface ' L '	.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table SolRadGlazing-B

Tabulated results for the exterior solar radiation test

*Zone load = (Internal surface heat flux) x (south facing glazed surface area)

Time	Surface Temperature (C)	Incident Solar flux (W/m ²)	Internal Surface Heat flux (W/m ²)*
6:10	20.0000	0.0000	0.0000
...			
7:50	20.0000	0.0000	0.0000
8:00	20.1140	6.7023	0.2032
8:10	20.3549	22.7881	2.1470
8:20	20.5586	39.2142	6.0428
8:30	20.7530	56.8256	12.1716
8:40	20.9295	74.2398	19.9869
8:50	21.0907	91.1987	29.0976
9:00	21.2582	109.8119	40.7069
9:10	21.4003	126.3650	52.1476
9:20	21.5469	144.1547	65.6295
9:30	21.6783	160.6917	79.0276
9:40	21.8025	176.8540	92.8469
9:50	21.9186	192.3992	106.7169
10:00	22.0275	207.3946	120.5963
10:10	22.1305	221.9318	134.4017
10:20	22.2271	235.8934	148.0231

10:30	22.3141	248.7289	160.7168
10:40	22.4006	261.7586	173.8508
10:50	22.4712	272.5820	184.8207
11:00	22.5428	283.7557	196.3339
11:10	22.5997	292.7578	205.5899
11:20	22.6544	301.5324	214.7143
11:30	22.7039	309.5538	223.0987
11:40	22.7456	316.4054	230.2742
11:50	22.7846	322.8687	237.0943
12:00	22.8110	327.2707	241.7083
12:10	22.8342	331.1653	245.8158
12:20	22.8502	333.8489	248.6433
12:30	22.8581	335.1922	250.0446
12:40	22.8649	336.3410	251.2821
12:50	22.8621	335.8698	250.7839
13:00	22.8500	333.8201	248.5958
13:10	22.8384	331.8597	246.5562
13:20	22.8120	327.4299	241.8471
13:30	22.7865	323.1753	237.3916
13:40	22.7516	317.3948	231.3107
13:50	22.7141	311.2319	224.8770
14:00	22.6658	303.3651	216.6516
14:10	22.6149	295.1848	208.1566
14:20	22.5531	285.3836	198.0039
14:30	22.4897	275.4568	187.8268
14:40	22.4138	263.7741	175.8935
14:50	22.3362	252.0376	164.0785
15:00	22.2473	238.8589	150.9780
15:10	22.1567	225.6971	138.1362
15:20	22.0546	211.1840	124.2313
15:30	21.9437	195.8312	109.9319
15:40	21.8287	180.3196	95.9397
15:50	21.7073	164.4282	82.2153
16:00	21.5789	148.1302	68.8337
16:10	21.4361	130.6403	55.3732
16:20	21.2865	113.0429	42.8868
16:30	21.1324	95.7282	31.8493
16:40	20.9631	77.6850	21.7473
16:50	20.7896	60.3242	13.6246
17:00	20.6119	43.8586	7.5028
17:10	20.4006	26.2627	2.8233
17:20	20.1703	10.0872	0.4485
17:30	20.0000	0.0000	0.0000
...			
19:10	20.0000	0.0000	0.0000

Figure SolRadGlazing-2 shows the plotted results of the incident solar flux vs. time and the internal surface heat flux vs. time using the data listed in table ExtSolRad-B.

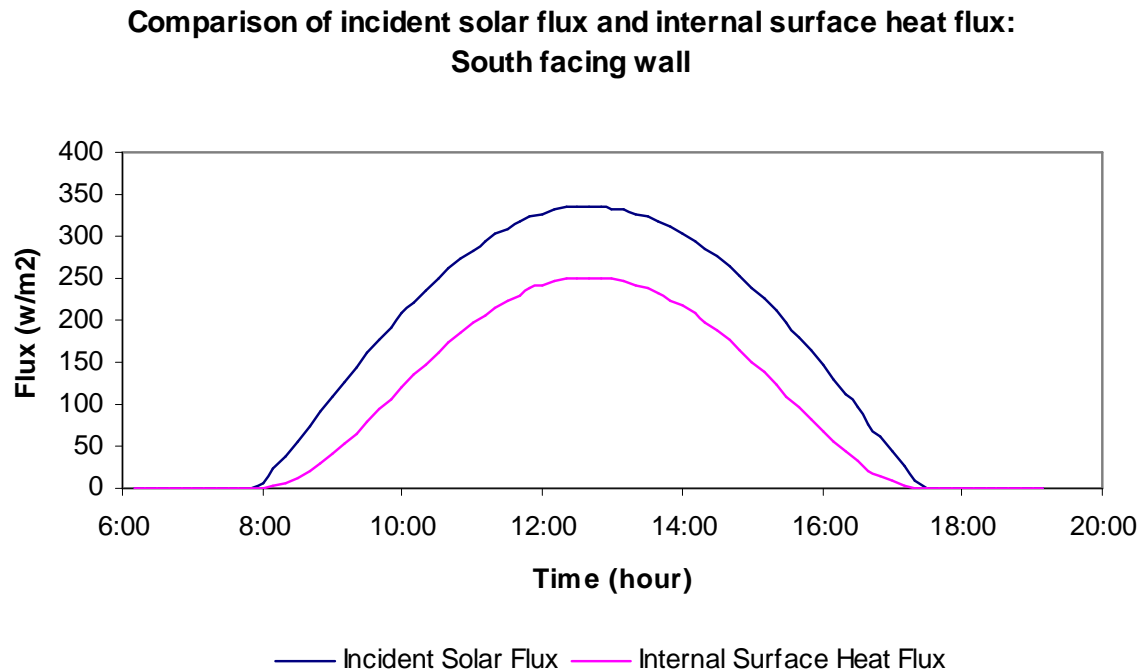


Figure SolRadGlazing-2: Incident solar flux on the glazing and internal surface heat flux calculated using the parameters in Table SolRadGalzing-A (south facing).

The tabulated results included below in table SolRadGlazing-D have been produced with the following test parameters shown in table SolRadGlazing-C for an west facing glazed surface.

Table SolRadGlazing-C
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ϵ '	90	Degree
Surface azimuth ' ψ '	270	Degree
Thickness of the surface ' L '	.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table SolRadGlazing-D
Tabulated results for the exterior solar radiation test

*Zone load = (Internal surface heat flux) x (west facing glazed surface area)

Time	Surface Temperature (C)	Incident Solar flux (W/m ²)	Internal Surface Heat flux (W/m ²)*
6:10	20.0000	0.0000	0.0000
...			
12:40	20.0000	0.0000	0.0000
12:50	20.5101	35.8942	4.1906
13:00	20.9271	74.6176	17.5082
13:10	21.2825	112.9743	38.7735
13:20	21.5976	150.6010	66.1773
13:30	21.8929	188.8484	99.2599
13:40	22.1572	225.5608	134.6603
13:50	22.4041	262.0816	172.3214
14:00	22.6334	298.1161	210.9840
14:10	22.8399	332.4746	248.6311
14:20	23.0339	366.5913	286.3199
14:30	23.2099	399.2974	322.4640

14:40	23.3686	430.4401	356.6925
14:50	23.5130	460.3560	389.3146
15:00	23.6472	489.7332	421.0258
15:10	23.7631	516.6002	449.7213
15:20	23.8652	541.6590	476.1958
15:30	23.9565	565.5267	501.1614
15:40	24.0303	586.1379	522.5217
15:50	24.0935	605.1963	542.1313
16:00	24.1435	621.7854	559.1012
16:10	24.1821	636.4298	574.0344
16:20	24.2061	647.7062	585.5403
16:30	24.2164	656.1928	594.2985
16:40	24.2130	661.3007	599.7361
16:50	24.1936	662.8513	601.7884
17:00	24.1571	660.3230	599.9775
17:10	24.1055	653.8845	594.4744
17:20	24.0305	642.1140	584.0370
17:30	23.9312	624.6300	568.3207
17:40	23.8029	600.5139	546.5062
17:50	23.6479	570.3197	519.1085
18:00	23.4422	529.5215	482.0270
18:10	23.1800	477.5370	434.7363
18:20	22.8606	415.1701	377.9736
18:30	22.4263	333.3423	303.4815
18:40	21.8858	237.8811	216.5700
18:50	21.1908	128.3832	116.8770
19:00	20.4356	33.3047	30.3171
19:10	20.0049	0.0819	0.0746

Figure SolRadGlazing-3 shows the plotted results of the incident solar flux vs. time and the internal surface heat flux vs. time using the data listed in table SolRadGlazing-D.

Comparison of incident solar flux and internal surface heat flux:
West facing wall

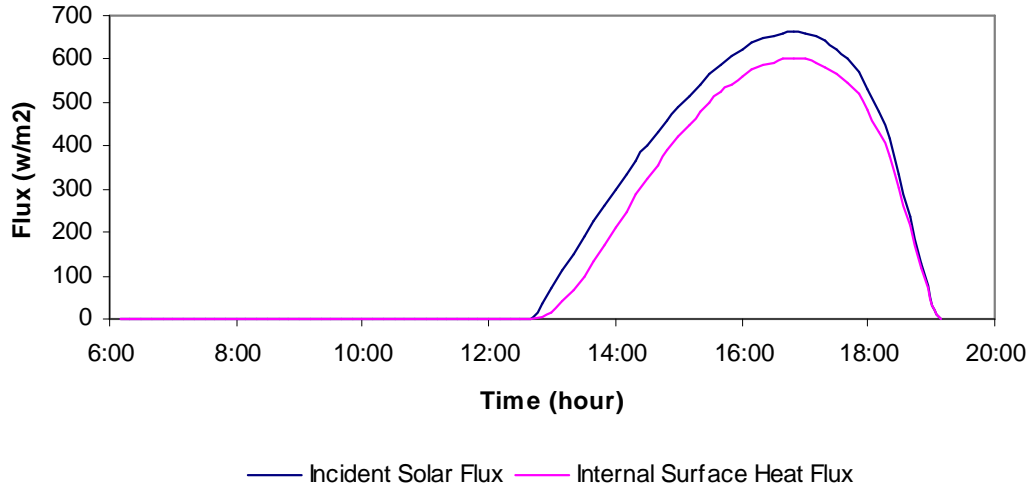


Figure SolRadGlazing-3: Incident solar flux on the glazing and internal surface heat flux calculated using the parameters in Table SolRadGalzing-C (west facing).

Analytical solution

Solar beam irradiation on glazed surface

Given:

The solar beam irradiation at normal incidence I_n , W/m^2

Solar elevation (or altitude) β , Radians

Then the solar beam irradiation on the surface is:

$$I_{bs} = \text{Max}(\cos \theta, 0) I_n \quad (\text{SolRadGlazing-2})$$

where,

I_{bs} = the beam radiation on the receiving plane of slope s , W/m^2

θ = the angle of incidence on the surface

while $\cos \theta$ is calculated (Fisher 1999) as:

$$\cos \theta = a_1 a_2 + b_1 b_2 + c_1 c_2 \quad (\text{SolRadGlazing -3})$$

and

$$a_1 = \sin \phi \cos \beta \quad b_1 = \cos \phi \cos \beta \quad c_1 = \sin \beta$$

$$a_2 = \sin \psi \sin \varepsilon \quad b_2 = \cos \psi \sin \varepsilon \quad c_2 = \cos \varepsilon$$

where,

ϕ = solar azimuth angle

ψ = surface azimuth angle

ε = the surface tilt angle

1. Absorption and transmittance of the surface

Suppose the external glazed surface is a single layer, the absorption and transmittance of the surface are calculated as follows (Duffie and Beckman 1991).

$$\alpha = (\alpha_{\perp} + \alpha_{\parallel})/2 \quad (\text{SolRadGlazing-4})$$

$$\tau = (\tau_{\perp} + \tau_{\parallel})/2 \quad (\text{SolRadGlazing-5})$$

where,

α = the absorption and of the surface, dimensionless

τ = the transmittance of the surface, dimensionless

α_{\perp} = the absorption for the perpendicular component of polarization,
dimensionless

α_{\parallel} = the absorption for the parallel component of polarization, dimensionless

τ_{\perp} = the transmittance for the perpendicular component of polarization,
dimensionless

τ_{\parallel} = the transmittance for the parallel component of polarization, dimensionless

and

$$\alpha_{\perp} = (1 - \tau_a) \cdot \frac{1 - r_{\perp}}{1 + r_{\perp} \tau_a} \quad (\text{SolRadGlazing-6})$$

$$\alpha_{\parallel} = (1 - \tau_a) \cdot \frac{1 - r_{\parallel}}{1 + r_{\parallel} \tau_a} \quad (\text{SolRadGlazing-7})$$

$$\tau_{\perp} = \frac{\tau_a (1 - r_{\perp})^2}{1 - (r_{\perp} \tau_a)^2} \quad (\text{SolRadGlazing-8})$$

$$\tau_{\parallel} = \frac{\tau_a (1 - r_{\parallel})^2}{1 - (r_{\parallel} \tau_a)^2} \quad (\text{SolRadGlazing-9})$$

Where,

τ_a = the transmittance when only absorption losses of the incident irradiation
have been considered, dimensionless

r_{\perp} = the reflectance for the perpendicular component of polarization,
dimensionless

r_{\parallel} = the reflectance for the parallel component of polarization, dimensionless

and (Duffie and Beckman 1991),

$$\tau_a = \exp\left(-\frac{KL}{\cos \theta_2}\right) \quad (\text{SolRadGlazing-10})$$

$$r_{\perp} = \frac{\sin^2(\theta_2 - \theta_1)}{\sin^2(\theta_2 + \theta_1)} \quad (\text{SolRadGlazing-11})$$

$$r_{//} = \frac{\tan^2(\theta_2 - \theta_1)}{\tan^2(\theta_2 + \theta_1)} \quad (\text{SolRadGlazing-12})$$

Where,

K = Extinction coefficient of the surface, m^{-1}

L = Thickness of the surface, m

θ_2 = Refraction angle of the surface, Radians

θ_1 = Incidence angle of the surface, Radians, $\theta_2 = \arcsin\left(\frac{n_a}{n_g} \sin \theta_1\right)$

and

n_a = Refractive index of the air, dimensionless

n_g = Refractive index of the surface, dimensionless

2. Heat balance

It is assumed that the glazing is sufficiently thin and of negligible thermal mass so that a heat balance can be calculated for the glazing at a single node. In this case the solar energy absorbed in the glass is convected to either inside or outside, and the heat balance can be written:

$$q''_{solar,absorbed} + q''_{convection,out} = q''_{convection,in} \quad (\text{SolRadGlazing-13})$$

The total energy transferred through the window due to solar radiation and via the convective resistance includes the transmitted component of the solar radiation and can be written:

$$q''_{total} = q''_{convection,in} + q''_{solar,transmitted} \quad (\text{SolRadGlazing-14})$$

Each component of the heat balance can be calculated as follows. First, given that only beam solar radiation I_{bs} (W/m^2) is incident upon the glazing, the solar beam irradiation absorbed by the surface is:

$$q''_{solar,absorbed} = \alpha I_{bs} \quad (\text{SolRadGlazing-15})$$

Where,

$q''_{solar,absorbed}$ = the solar beam irradiation absorbed by the surface, W/m^2 .

α = the solar absorption of the surface.

I_{bs} = the beam radiation on the surface, W/m^2

Given the conductivity of the surface is very high, the surface can be assumed to have a uniform temperature, so that the exterior convection heat flux is:

$$q''_{convection,out} = h_{c,o} (T_{a,o} - T_s) \quad (\text{SolRadGlazing-16})$$

where,

$q''_{convection,out}$ = external convection heat flux, W/m²

$h_{c,o}$ = the exterior convection coefficient, W/m²K

$T_{a,o}$ = external air temperature , C

T_s = surface (glass) temperature , C

The interior convection heat flux is:

$$q''_{convection,in} = h_{c,i} (T_s - T_{a,i}) \quad (\text{SolRadGlazing-17})$$

where,

$q''_{convection,in}$ = internal convection heat flux, W/m²

$h_{c,i}$ = interior convection coefficient, W/m²K

T_s = surface (glass) temperature , C

$T_{a,i}$ = internal air temperature , C

The heat balance on the glazing can then be written by substituting equations (SolRadGlazing-16~18) into (SolRadGlazing-14) such that,

$$\alpha I_{bs} + h_{c,o} (T_{a,o} - T_s) = h_{c,i} (T_s - T_{a,i}) \quad (\text{SolRadGlazing-18})$$

This heat balance equation can be rearranged to solve for T_s and substituted into equation (SolRadGlazing-18) to solve for $q''_{convection,in}$ giving,

$$q''_{convection,in} = \frac{(T_{a,o} - T_{a,i}) + \alpha I_{bs} \cdot R_{c,o}}{R_{c,o} + R_{c,i}} \quad (\text{SolRadGlazing-19})$$

where,

$$R_{c,o} = 1/h_{c,o} \quad (\text{SolRadGlazing-20})$$

$$R_{c,i} = 1/h_{c,i} \quad (\text{SolRadGlazing-21})$$

The solar beam irradiation transmitted into the zone is:

$$q''_{solar,transmitted} = \tau \cdot I_{bs} \quad (\text{SolRadGlazing-22})$$

Where,

$q''_{solar,transmitted}$ = the solar beam irradiation transmitted into the zone, W/m²

τ = the transmittance of the surface

I_{bs} = the beam radiation on the surface, W/m²

Finally, equation (SolRadGlazing-15) for the total heat flux transferred into the zone (W/m²) can be expanded to give,

$$q_{Total}'' = \tau \cdot I_{bs} + \frac{(T_{a,o} - T_{a,i}) + \alpha I_{bs} \cdot R_{c,o}}{R_{c,o} + R_{c,i}} \quad (\text{SolRadGlazing-23})$$

All internal surfaces are adiabatic and long wave radiation is eliminated, thus zone load is simply the summation of internal convection heat flux and the solar beam irradiation transmitted into the zone multiplied by the relevant glazing surface area (9.0 m²). Therefore,

$$\text{Zoneload} = A_g \cdot q_{total}'' \quad (\text{SolRadGlazing-24})$$

References

Duffie J.A., W.A. Beckman 1991. *Solar Engineering of thermal Processes*. New York: Wiley.

Fisher D. 1999. *ASHRAE Loads Toolkit Documentation (Draft)*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Atlanta, GA.

McQuiston F.C., J.D. Parker, J.D. Spitler 2000. *Heating, Ventilating, and Air Conditioning: Analysis and Design*. New York: Wiley.

USNO 2001. U.S. Naval Observatory: web site - <http://aa.usno.navy.mil/data/docs/AltAz.html>

Walton, G. N. 1981. *Passive solar extension of the Building Loads Analysis and System Thermodynamics (BLAST) program*. Technical Report, United States Army Construction Engineering Research Laboratory, Champaign, IL.

Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test SolRadShade: Solar Radiation – Window Shading

Objective

The objective of this test is to test the treatment of semi-infinite external shading over glazed surface at different hours of a single day for a selection of locations. Three types of window shading are considered: Semi-infinite horizontal fin, Semi-infinite vertical fin and a union of semi-infinite horizontal and vertical fin (similar to a window reveal)⁷. The shading combinations and geometry are illustrated in Figure SolRadShade-1.

Analytical Model

Calculation of incident direct solar radiation is calculated from tabulated solar position data in the same way as in test ExtSolRad. Transmission and absorption of solar radiation is modified from the SolRadGlazing test simply by the reduction in effective glazed area by the casting of shadows from the shading devices (there is no diffuse irradiation). Shadow geometry is calculated using published shading geometric relations (Rodriguez and Alvarez 1991). Shading devices are assumed to be completely opaque and non-reflecting. No radiation is exchanged between the glazing and shading devices.

Transmission and absorption of the direct solar radiation is calculated using optical relations for a clear single pane of glazing. It is assumed that the glazed surface has no thermal mass and its conductivity is very high so that it will have a uniform temperature. Conduction through and convection from the window surface is calculated from an instantaneous heat balance. Diffuse radiation is eliminated from the test by setting it to zero in the weather file. It is assumed that there is now long wave radiation at any surface.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. Only one surface is to be external and is to be entirely glazed. This external glazed surface may be selected to be any of the six surfaces of the zone except for the floor. The glazing consists of a single layer of plain glass without any type of coating. No other form of shading is present (e.g. additional window reveal). The interior and exterior air temperature have to be taken as constant.

⁷ The shading is said to be semi-infinite in that it has a finite depth (from the face of the zone exterior surface to its front edge) but is infinite in the direction parallel to the wall surface.

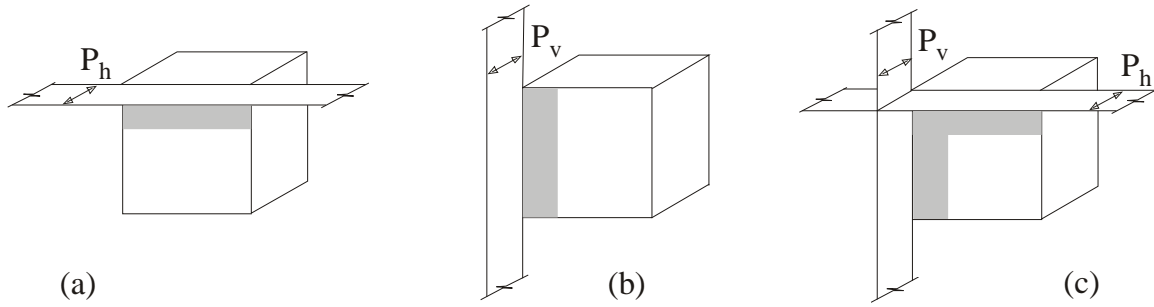


Figure SolRadGlazing-1: Zone and shading geometry showing (a) semi-infinite horizontal fin shading, (b) semi-infinite vertical fin shading, (c) combined semi-infinite horizontal and vertical fin shading. Vertical shading is indicated on the left by convention.

Transient effects due to thermal storage need to be avoided by making the fabric heat capacity and/or density suitably low. The effects long wave radiation, infiltration and internal gains must be eliminated in this test. The interior and exterior convection coefficients may be correlations of surface and air temperature.

Test parameters

The different solar beam irradiation in this test is calculated with solar position data from the US Naval Observatory (USNO 2001) and the beam irradiation at normal incidence from the weather data. The user is able to choose the location of the test zone (from a list of four: Atlanta (W 84° 25', N33° 46'), Boston (W 71° 05', N42° 19'), Chicago (W 87° 41', N41° 51') and Los Angeles (W118° 22', N34° 05')) and two test dates (June 21st, 1999 or August 21st, 1999) when generating the weather files using the toolkit software. The user must also choose the surface tilt angle, the surface azimuth, and the thickness and extinction coefficient of the surface. In addition, the refractive index of the glazed surface and the hanging length of external shading fin have to be specified. In case of semi-infinite vertical fin and the union of semi-infinite horizontal and vertical fin, the user also needs to specify on which side of the window is the vertical fin – left side or right side.

The inside and outside air temperatures have to be specified as the same and taken as fixed. The same value of the inside air temperature must also be used in the input data of the program to be tested. Also, convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed) should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{SolRadShade-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations. The test suite software input screen for this test is shown in Figure SolRadGlazing2 below.

	Coefficient 'A'	Coefficient 'C'	Coefficient 'n'	Units
Inside correlation	0.0	1.49	0.345	(W/m².K)
Outside correlation	0.0	0.84	0.333	(W/m².K)

Figure SolRadShade-2: Test suite software parameter input screen for test SolRadShade.

Test output

The principle data of interest in this test are the predicted sunlit area of the window, the predicted cooling load of the test zone with and without window shading. This data is listed in the output file along with the input data.

Test results

The tabulated results include the shaded areas and corresponding zone loads for south facing and west facing surfaces with three combinations of shading (i.e. six cases in total). The shading is applied as follows: (i) horizontal shading (overhang) only; (ii) a vertical shade (fin) only; (iii) both horizontal and vertical shading. The tabulated results for each combination of shading are included below in table SolRadShade-B. These results have been produced with the test parameters shown in table SolRadShade-A where the surface is south facing.

Table SolRadShade -A
Test parameters used in generating the tabulated results
(south facing surface)

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degrees
Surface azimuth ' ψ '	180	Degrees
Thickness of the surface ' L '	.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
Depth of the horizontal fin ' P_h '	0.6	m
Depth of the vertical fin ' P_v '	1.0	m
Vertical fin position*	Right side	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

* see figure SolRadGlazing-1 for positioning convention

Table SolRadShade –B
 Tabulated results for the window shading test (south facing surface)

Time	Sunlit Area (m ²)			Zone Load (W)			
	Horizontal & Vertical	Horizontal Only	Vertical Only	Unshaded	Horizontal & Vertical	Horizontal Only	Vertical Only
6:10	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00
...							
7:50	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00
8:00	0.0000	0.0000	0.0000	1.83	0.00	0.00	0.00
8:10	0.0000	0.0000	0.0000	19.32	0.00	0.00	0.00
8:20	0.0000	0.0000	0.0000	54.39	0.00	0.00	0.00
8:30	0.0000	0.0000	0.0000	109.54	0.00	0.00	0.00
8:40	0.0000	0.0000	0.0000	179.88	0.00	0.00	0.00
8:50	0.0000	0.3318	0.0000	261.88	0.00	9.66	0.00
9:00	0.0000	1.3012	0.0000	366.36	0.00	52.97	0.00
9:10	0.0000	1.8633	0.0000	469.33	0.00	97.17	0.00
9:20	0.0000	2.3804	0.0000	590.67	0.00	156.22	0.00
9:30	0.0000	2.7438	0.0000	711.25	0.00	216.84	0.00
9:40	0.0000	3.0350	0.0000	835.62	0.00	281.79	0.00
9:50	0.0497	3.2669	0.1369	960.45	5.30	348.64	14.61
10:00	0.4248	3.4683	1.1024	1085.37	51.23	418.26	132.95
10:10	0.7928	3.6283	1.9666	1209.62	106.56	487.65	264.31
10:20	1.1387	3.7812	2.7104	1332.21	168.55	559.70	401.20
10:30	1.4622	3.8916	3.3815	1446.45	234.99	625.45	543.46
10:40	1.7849	4.0088	4.0072	1564.66	310.30	696.93	696.65
10:50	2.0642	4.0810	4.5523	1663.39	381.51	754.25	841.36
11:00	2.3464	4.1718	5.0619	1767.00	460.67	819.06	993.83
11:10	2.5955	4.2190	5.5367	1850.31	533.61	867.39	1138.29
11:20	2.8404	4.2755	5.9790	1932.43	609.87	918.02	1283.77
11:30	3.0771	4.3262	6.4013	2007.89	686.49	965.17	1428.13
11:40	3.3002	4.3652	6.8043	2072.47	759.96	1005.19	1566.86
11:50	3.5221	4.4085	7.1903	2133.85	835.06	1045.23	1704.78
12:00	3.7174	4.4277	7.5562	2175.37	898.52	1070.20	1826.40
12:10	3.9150	4.4489	7.9199	2212.34	962.37	1093.61	1946.85
12:20	4.1022	4.4620	8.2742	2237.79	1019.98	1109.44	2057.33
12:30	4.2795	4.4649	8.6263	2250.40	1070.07	1116.43	2156.97
12:40	4.4636	4.4766	8.9738	2261.54	1121.62	1124.90	2254.96
12:50	4.4741	4.4741	9.0000	2257.05	1122.02	1122.02	2257.05
13:00	4.4577	4.4577	9.0000	2237.36	1108.18	1108.18	2237.36
13:10	4.4545	4.4545	9.0000	2219.01	1098.27	1098.27	2219.01
13:20	4.4216	4.4216	9.0000	2176.62	1069.34	1069.34	2176.62
13:30	4.4041	4.4041	9.0000	2136.52	1045.50	1045.50	2136.52
13:40	4.3703	4.3703	9.0000	2081.80	1010.90	1010.90	2081.80
13:50	4.3415	4.3415	9.0000	2023.89	976.30	976.30	2023.89
14:00	4.2934	4.2934	9.0000	1949.86	930.17	930.17	1949.86
14:10	4.2469	4.2469	9.0000	1873.41	884.01	884.01	1873.41

14:20	4.1806	4.1806	9.0000	1782.03	827.78	827.78	1782.03
14:30	4.1188	4.1188	9.0000	1690.44	773.62	773.62	1690.44
14:40	4.0248	4.0248	9.0000	1583.04	707.94	707.94	1583.04
14:50	3.9345	3.9345	9.0000	1476.71	645.57	645.57	1476.71
15:00	3.8194	3.8194	9.0000	1358.80	576.64	576.64	1358.80
15:10	3.6984	3.6984	9.0000	1243.23	510.88	510.88	1243.23
15:20	3.5351	3.5351	9.0000	1118.08	439.17	439.17	1118.08
15:30	3.3433	3.3433	9.0000	989.39	367.53	367.53	989.39
15:40	3.1090	3.1090	9.0000	863.46	298.28	298.28	863.46
15:50	2.8401	2.8401	9.0000	739.94	233.50	233.50	739.94
16:00	2.5057	2.5057	9.0000	619.50	172.48	172.48	619.50
16:10	2.0563	2.0563	9.0000	498.36	113.86	113.86	498.36
16:20	1.4492	1.4492	9.0000	385.98	62.15	62.15	385.98
16:30	0.6841	0.6841	9.0000	286.64	21.79	21.79	286.64
16:40	0.0000	0.0000	9.0000	195.73	0.00	0.00	195.73
16:50	0.0000	0.0000	9.0000	122.62	0.00	0.00	122.62
17:00	0.0000	0.0000	9.0000	67.53	0.00	0.00	67.53
17:10	0.0000	0.0000	9.0000	25.41	0.00	0.00	25.41
17:20	0.0000	0.0000	9.0000	4.04	0.00	0.00	4.04
17:30	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00
...							
19:10	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00

Figure SolRadShade-2, Figure SolRadShade-3 and Figure SolRadShade-4 show the comparison of the zone load with and without shading in the case of Semi-infinite horizontal fin, Semi-infinite vertical fin and a union of semi-infinite horizontal and vertical fin respectively, using the data listed in table SolRadShade-B for a south facing surface.

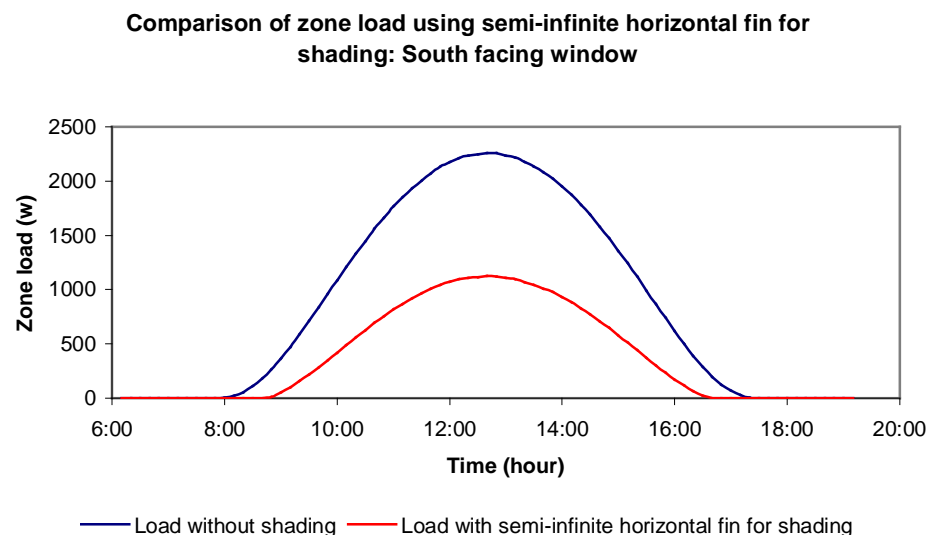


Figure SolRadShade-2: Analytical prediction of Zone Load with and without horizontal shading. South facing window.

**Comparison of zone load using semi-infinite vertical fin for shading:
South facing window**

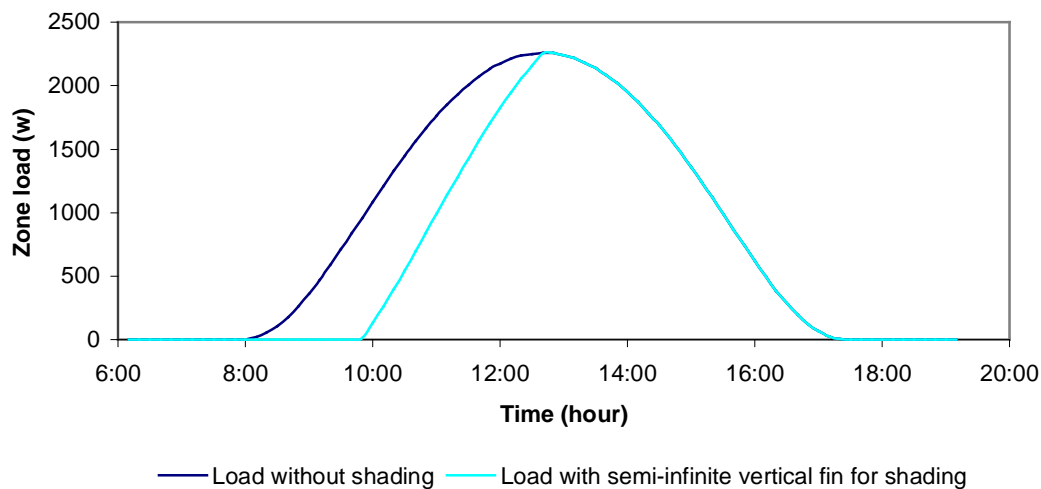


Figure SolRadShade-3: Analytical prediction of Zone Load with and without vertical shading. South facing window. The vertical shading is positioned on the right of the window

**Comparison of zone load using semi-infinite horizontal and vertical
fin for shading: South facing window**

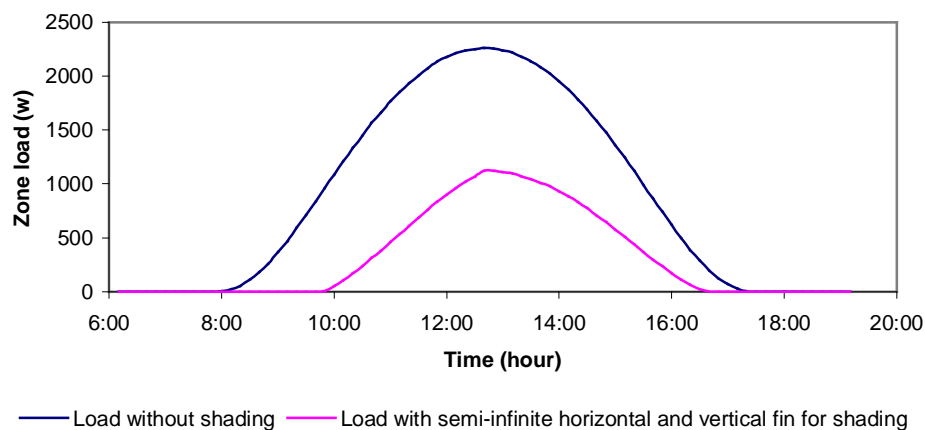


Figure SolRadShade-4: Analytical prediction of Zone Load with and without combined vertical and horizontal shading. South facing window. The vertical shading is positioned on the right of the window

The tabulated results included below in table SolRadShade-D have been produced with the following test parameters shown in table SolRadShade-C where the surface is west facing.

Table SolRadShade-C

Test parameters used in generating the tabulated results(west facing surface)

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degrees
Surface azimuth ' ψ '	270	Degrees
Thickness of the surface ' L '	.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
Depth of the horizontal fin ' P_h '	3.0	m
Depth of the vertical fin ' P_v '	3.0	m
Vertical fin position*	Right side	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

* see figure SolRadGlazing-1 for positioning convention

Table SolRadShade –D

Tabulated results for the window shading test (west facing surface)

Time	Sunlit Area (m ²)			Zone Load (W)			
	Horizontal & Vertical	Horizontal Only	Vertical Only	Unshaded	Horizontal & Vertical	Horizontal Only	Vertical Only
6:10	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00
...							
12:40	0.0000	0.0000	0.0000	0.00	0.00	0.00	0.00
12:50	0.0000	0.0000	0.0000	37.72	0.00	0.00	0.00
13:00	0.0000	0.0000	0.0000	157.57	0.00	0.00	0.00
13:10	0.0000	0.0000	0.0000	348.96	0.00	0.00	0.00
13:20	0.0000	0.0000	0.0000	595.60	0.00	0.00	0.00
13:30	0.0000	0.0000	0.0000	893.34	0.00	0.00	0.00
13:40	0.0000	0.0000	0.0000	1211.94	0.00	0.00	0.00
13:50	0.0000	0.0000	0.0000	1550.89	0.00	0.00	0.00
14:00	0.0000	0.0000	0.0000	1898.86	0.00	0.00	0.00
14:10	0.0000	0.0000	1.0094	2237.68	0.00	0.00	250.97
14:20	0.0000	0.0000	1.9937	2576.88	0.00	0.00	570.83
14:30	0.0000	0.0000	2.7913	2902.18	0.00	0.00	900.10
14:40	0.0000	0.0000	3.4848	3210.23	0.00	0.00	1243.00

14:50	0.0000	0.0000	4.0726	3503.83	0.00	0.00	1585.54
15:00	0.0000	0.0000	4.6104	3789.23	0.00	0.00	1941.10
15:10	0.0000	0.0000	5.0680	4047.49	0.00	0.00	2279.19
15:20	0.0000	0.0000	5.4910	4285.76	0.00	0.00	2614.81
15:30	0.0000	0.0000	5.8835	4510.45	0.00	0.00	2948.57
15:40	0.0000	0.0000	6.2312	4702.70	0.00	0.00	3255.96
15:50	0.4603	0.6320	6.5548	4879.18	249.55	342.64	3553.54
16:00	0.9631	1.2642	6.8559	5031.91	538.44	706.83	3833.14
16:10	1.4888	1.8733	7.1526	5166.31	854.61	1075.35	4105.82
16:20	1.9901	2.4109	7.4292	5269.86	1165.29	1411.67	4350.12
16:30	2.5062	2.9342	7.6870	5348.69	1489.41	1743.80	4568.40
16:40	3.0035	3.4033	7.9427	5397.62	1801.29	2041.06	4763.55
16:50	3.5143	3.8662	8.1809	5416.10	2114.87	2326.61	4923.19
17:00	4.0187	4.3046	8.4022	5399.80	2411.11	2582.65	5041.14
17:10	4.5148	4.7038	8.6385	5350.27	2683.95	2796.26	5135.38
17:20	5.0247	5.1049	8.8586	5256.33	2934.60	2981.43	5173.76
17:30	5.4909	5.4909	9.0000	5114.89	3120.60	3120.60	5114.89
17:40	5.8641	5.8641	9.0000	4918.56	3204.79	3204.79	4918.56
17:50	6.2097	6.2097	9.0000	4671.98	3223.50	3223.50	4671.98
18:00	6.5638	6.5638	9.0000	4338.24	3163.91	3163.91	4338.24
18:10	6.9111	6.9111	9.0000	3912.63	3004.51	3004.51	3912.63
18:20	7.2367	7.2367	9.0000	3401.76	2735.27	2735.27	3401.76
18:30	7.5746	7.5746	9.0000	2731.33	2298.75	2298.75	2731.33
18:40	7.8944	7.8944	9.0000	1949.13	1709.69	1709.69	1949.13
18:50	8.2132	8.2132	9.0000	1051.89	959.93	959.93	1051.89
19:00	8.5167	8.5167	9.0000	272.85	258.20	258.20	272.85
19:10	8.8219	8.8219	9.0000	0.67	0.66	0.66	0.67

Figure SolRadShade-5, Figure SolRadShade-6 and Figure SolRadShade-7 show the comparison of the zone load with and without shading in the case of Semi-infinite horizontal fin, Semi-infinite vertical fin and a union of semi-infinite horizontal and vertical fin respectively, using the data listed in table SolRadShade-D for an west facing surface.

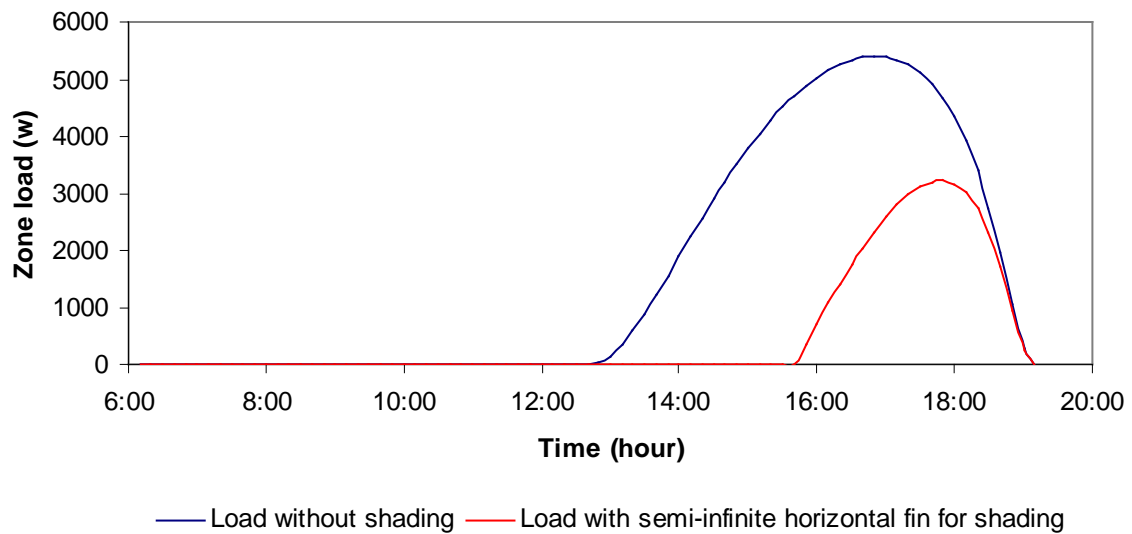
Comparison of zone load using semi-infinite horizontal fin for shading: West facing window

Figure SolRadShade-5: Analytical prediction of Zone Load with and without horizontal shading. West facing window.

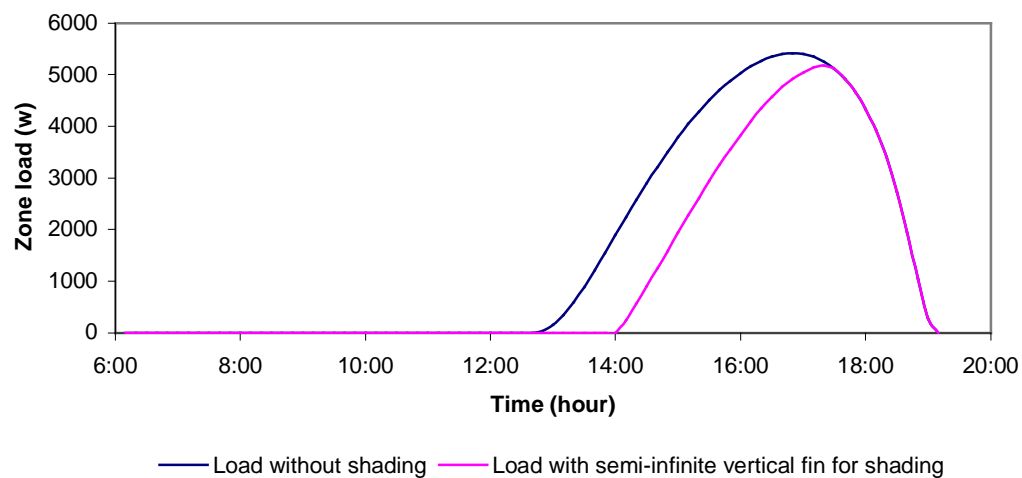
Comparison of zone load using semi-infinite vertical fin for shading: West facing window

Figure SolRadShade-6: Analytical prediction of Zone Load with and without vertical shading. West facing window. The vertical shading is positioned on the right of the window

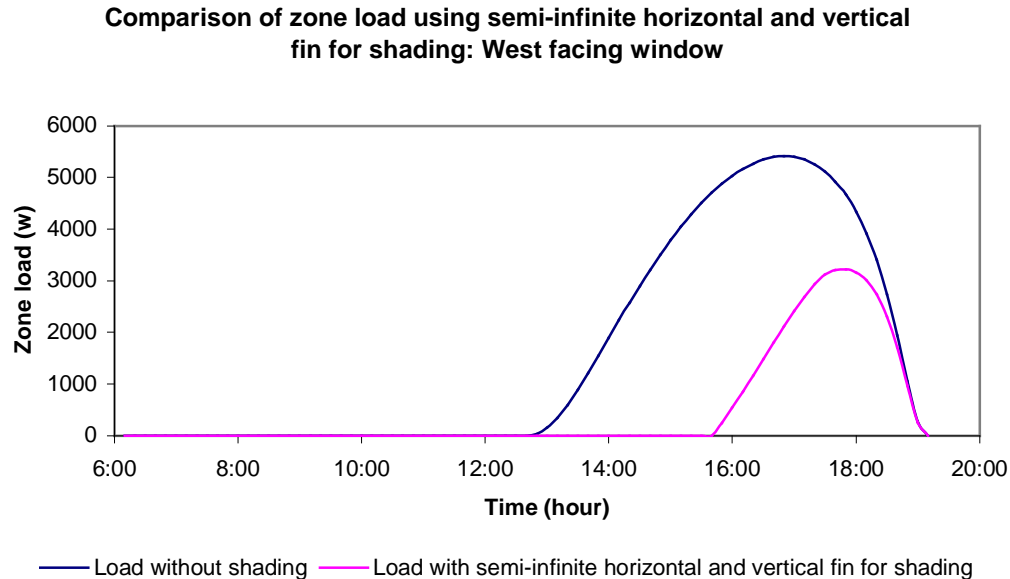


Figure SolRadShade-7: Analytical prediction of Zone Load with and without combined horizontal and vertical shading. The window is West facing with the vertical shading is positioned on the right.

Analytical solution

1. Solar beam irradiation on the window is the same as that in the test of Solar Radiation on Glazed Surface: SolRadGlazing.
2. Absorption and transmittance of the window are the same as that in the test of Solar Radiation on Glazed Surface: SolRadGlazing.
3. Heat balance of the window is the same as that in the test of Solar Radiation on Glazed Surface: SolRadGlazing.
4. External window shading

For a semi-infinite horizontal fin, where the fin is positioned directly at the top of the window, the sunlit area can be calculated (Rodriguez and Alvarez 1991, HoF 1997:Ch. 29) as follows:

$$H' = \max(0, H - P_h \tan \Omega) \quad (\text{SolRadShade-2})$$

$$\tan \Omega = \frac{\tan \beta}{\cos \gamma} \quad (\text{SolRadShade-3})$$

$$A_{\text{sunlit}} = W \times H' \quad (\text{SolRadShade-4})$$

Where,

W = window width, m

H = window height, m

P_h = depth of the horizontal fin, m

β = solar elevation (or altitude), Radians

Ω = profile angle, Radians

γ = wall solar azimuth angle, Radians, $\gamma = |\phi - \psi|$ and $\gamma \in [0, \pi/2]$

ϕ = solar azimuth, measured east from north, Radians

ψ = surface azimuth, measured east from north, Radians

H' = sunlit rectangle height, m

A_{sunlit} = sunlit rectangle area, m²

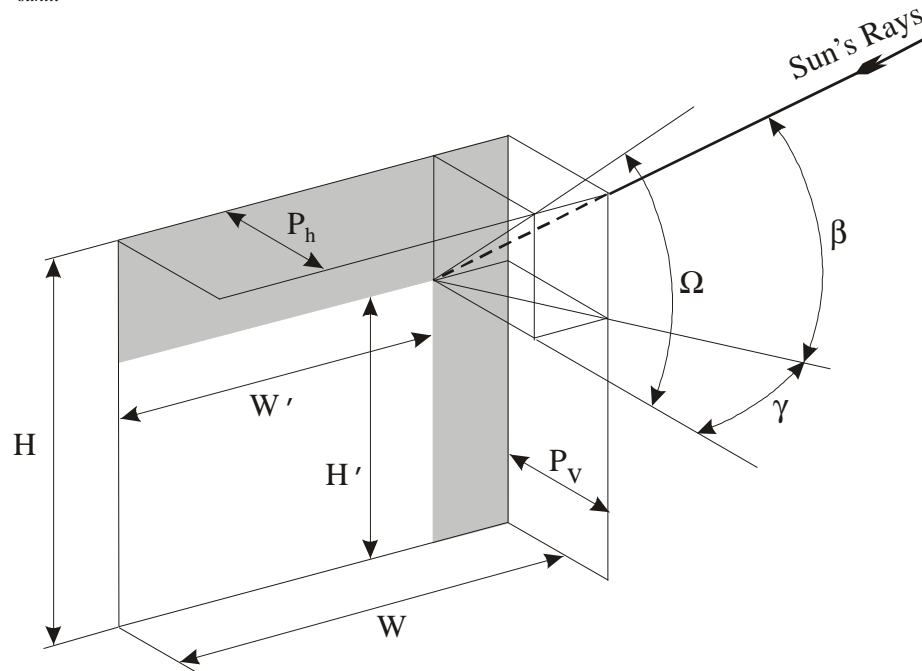


Figure SolRadShade-8: Window shading geometry and dimensions

For a semi-infinite vertical fin, where the fin is positioned directly at the side of the window, the sunlit area can be calculated (Rodriguez and Alvarez 1991, HoF 1997:Ch. 29) as follows:

$$W' = \max(0, W - P_v \tan \gamma) \quad (\text{SolRadShade-5})$$

$$A_{\text{sunlit}} = W' \times H \quad (\text{SolRadShade-6})$$

Where,

W = window length, m

H = window height, m

P_v = depth of the vertical fin, m

γ = wall solar azimuth angle, Radians, $\gamma = |\phi - \psi|$ and $\gamma \in [0, \pi/2]$

W' = sunlit rectangle width, m

A_{sunlit} = sunlit rectangle area, m²

For a union of semi-infinite horizontal and vertical fin (e.g. both vertical and horizontal fins or a window reveal), the new sunlit area can be calculated as follows:

$$W' = \max(0, W - P_v \tan \Omega) \quad (\text{SolRadShade-7})$$

$$H' = \max(0, H - P_h \tan \delta) \quad (\text{SolRadShade-8})$$

$$A_{sunlit} = W' \times H' \quad (\text{SolRadShade-9})$$

The new zone load is then the total heat flux multiplied by the new sunlit area:

$$\text{Zoneload}' = A_{sunlit} \cdot q_{total}'' \quad (\text{SolRadShade-10})$$

It should be noted that for a semi-infinite vertical fin, the sunlit area is also dependent on the angle between the solar azimuth ϕ and the fin azimuth φ . Only when this angle is greater than 90 degrees does the vertical fin have a shading effect on the window surface. Hence it is necessary to specify which side of the window the vertical fin is positioned – left or right when looking at the window from outside.

References

HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Atlanta, GA.

Rodriguez E. A., S. Alvarez 1991. Solar Shading Analytical Tests (I). IEA21RN136.

USNO 2001. U.S. Naval Observatory: web site - <http://aa.usno.navy.mil/data/docs/AltAz.html>

Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test WinReveal: Window Reveal

Objective

The objective of this test is to test the treatment of window reveal. At different hours of a single day for a given location, the test results of the response to different solar beam irradiation on glazed surfaces and the calculation of incidence angle, of solar absorption and transmission of beam irradiation on glazed surface from the previous tests are used. It is intended that the glazed surface has no thermal mass and its conductivity is very high so that it will have a uniform temperature. The analytical solution of the zone load is compared with the results of the program to be tested for the selected day.

Analytical Model

Calculation of incident direct solar radiation is calculated from tabulated solar position data in the same way as in test ExtSolRad. Transmission and absorption of solar radiation is modified from the SolRadGlazing test simply by the reduction in effective glazed area by the casting of shadows from the window reveal. Shadow geometry is calculated using the same published shading geometric relations as test SolRadShade. Transmission and absorption of the direct solar radiation is calculated using optical relations for a clear single pane of glazing. It is assumed that the glazed surface has no thermal mass and its conductivity is very high so that it will have a uniform temperature. Conduction through and convection from the window surface are calculated from an instantaneous heat balance. Diffuse radiation is eliminated from the test by setting it to zero in the weather file. It is assumed that there is no long wave radiation at any surface.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. Only one surface is to be external. This external surface may be selected to be any of the six surfaces of the zone except for the floor and must be a single layer of material. The external surface is supposed to be the window tested. Instead of the semi-infinite horizontal and vertical fin in the window-shading test, window reveal is present in this test.

The effects of long wave radiation, infiltration and internal heat gains must be eliminated in this test. The opaque part of the external surface should have zero solar absorption and be adiabatic, all internal surfaces of the cubic are black, adiabatic and have no thermal mass. The interior and exterior air temperature have to be taken as fixed. The interior and exterior convection coefficients may be correlations of surface and air temperature.

Test parameters

The different solar beam irradiation in this test is calculated with solar position data from the US Naval Observatory (USNO 2001) and the beam irradiation at normal incidence from the weather data. The user is able to choose the location of the test zone (from a list of four: Atlanta (W 84° 25', N33° 46'), Boston (W 71° 05', N42° 19'), Chicago (W 87° 41', N41° 51') and Los Angeles (W118° 22', N34° 05')) and two test dates (June 21st, 1999 or August 21st, 1999) when generating the weather files using the toolkit software. The user must also choose the surface tilt angle, the surface azimuth, and the thickness and extinction coefficient of the surface. In addition, the thickness, refractive index and extinction coefficient of the window surface, the window length, window height and the depth of the reveal have to be specified.

The inside and outside air temperatures have to be specified as the same and taken as fixed. The same value of the inside air temperature must also be used in the input data of the program to be tested. Also, convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{WinReveal-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations.

Test output

The principal data of interest in this test are the predicted sunlit area of the window and the predicted cooling load of the test zone with and without window reveal. This data is listed in the output file along with the input data. The test suite software input screen for this test is shown below in Figure WinReveal-1.

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

SSConv SSCond TC 1 TC 2 TC 3 ExtSolRad SolRadGlazing SolRadShade
IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCoup WinReveal

Test Temperatures

Inside temperature 20.0 (°C)

Outside temperature 20.0 (°C)

Test Location Date

Location Atlanta

Date June 21st

Convection Coefficients

	Coefficient 'A'	Coefficient 'C'	Coefficient 'h'	
Inside correlation	0.0	1.49	0.345	(W/m².K)
Outside correlation	0.0	0.84	0.333	(W/m².K)

Surface Properties

Surface tilt angle	90	(degree)
Surface azimuth	180	(degree)
Surface thickness	0.0023	(m)
Extinction coefficient	10.0	(m⁻¹)
Refractive index	1.526	
Window length	2.0	(m)
Window height	2.0	(m)
Depth of the reveal	0.3	(m)

Start Calculation

Status: TMY2 SI Units

Figure WinReveal-1: Test suite software parameter input screen for test WinReveal.

Test results

Test results are documented for two cases with the window surface facing due south and due west. The tabulated results included below in table WinReveal-B have been produced with the following test parameters shown in table WinReveal-A where the surface is south facing. The tabulated results include the shaded area and corresponding zone load.

Table WinReveal -A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degrees
Surface azimuth ' ψ '	180	Degrees
Thickness of the glass	.0023	m
Depth of the window reveal ' P '	0.3	m
Width of the window ' W '	2.0	m
Height of the window ' H '	2.0	m
Extinction coefficient of the window surface ' K '	10.0	m ⁻¹
Refractive index of the window surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table WinReveal -B
Tabulated results for the window reveal test

Time	Sunlit Area with Reveal (M ²)	Zone Load without Reveal (W)	Zone Load with Reveal (W)
6:10	0.0000	0.00	0.00
...			
7:50	0.0000	0.00	0.00
8:00	0.0000	0.51	0.00
8:10	0.0000	7.27	0.00
8:20	0.0000	21.90	0.00
8:30	0.0000	45.54	0.00
8:40	0.0000	76.07	0.00
8:50	0.0000	111.91	0.00
9:00	0.1622	157.82	6.40
9:10	0.3930	203.20	19.97
9:20	0.6282	256.81	40.33

9:30	0.8276	310.17	64.18
9:40	1.0059	365.29	91.86
9:50	1.1632	420.66	122.33
10:00	1.3047	476.11	155.29
10:10	1.4324	531.29	190.26
10:20	1.5495	585.77	226.92
10:30	1.6519	636.55	262.87
10:40	1.7530	689.12	302.01
10:50	1.8354	733.03	336.35
11:00	1.9199	779.12	373.95
11:10	1.9896	816.18	405.98
11:20	2.0588	852.72	438.90
11:30	2.1248	886.30	470.79
11:40	2.1855	915.03	499.96
11:50	2.2460	942.35	529.14
12:00	2.2971	960.83	551.79
12:10	2.3489	977.28	573.88
12:20	2.3971	988.61	592.44
12:30	2.4418	994.22	606.93
12:40	2.4889	999.18	621.73
12:50	2.4514	997.18	611.13
13:00	2.4026	988.42	593.68
13:10	2.3579	980.25	577.84
13:20	2.3032	961.39	553.56
13:30	2.2517	943.54	531.14
13:40	2.1949	919.19	504.38
13:50	2.1381	893.42	477.55
14:00	2.0728	860.48	445.89
14:10	2.0075	826.46	414.78
14:20	1.9323	785.81	379.61
14:30	1.8570	745.06	345.89
14:40	1.7684	697.29	308.28
14:50	1.6782	650.01	272.71
15:00	1.5740	597.59	235.16
15:10	1.4662	546.23	200.22
15:20	1.3403	490.64	164.40
15:30	1.1985	433.50	129.89
15:40	1.0435	377.63	98.51
15:50	0.8719	322.89	70.38
16:00	0.6797	269.57	45.81
16:10	0.4538	216.02	24.51
16:20	0.2084	166.46	8.67
16:30	0.0000	122.78	0.00
16:40	0.0000	82.98	0.00
16:50	0.0000	51.20	0.00

17:00	0.0000	27.50	0.00
17:10	0.0000	9.76	0.00
17:20	0.0000	1.28	0.00
17:30	0.0000	0.00	0.00
...			
19:10	0.0000	0.00	0.00

Figure WinReveal-2 shows the comparison of the zone load with and without reveal using the data listed in table WinReveal –B for a south-facing window.

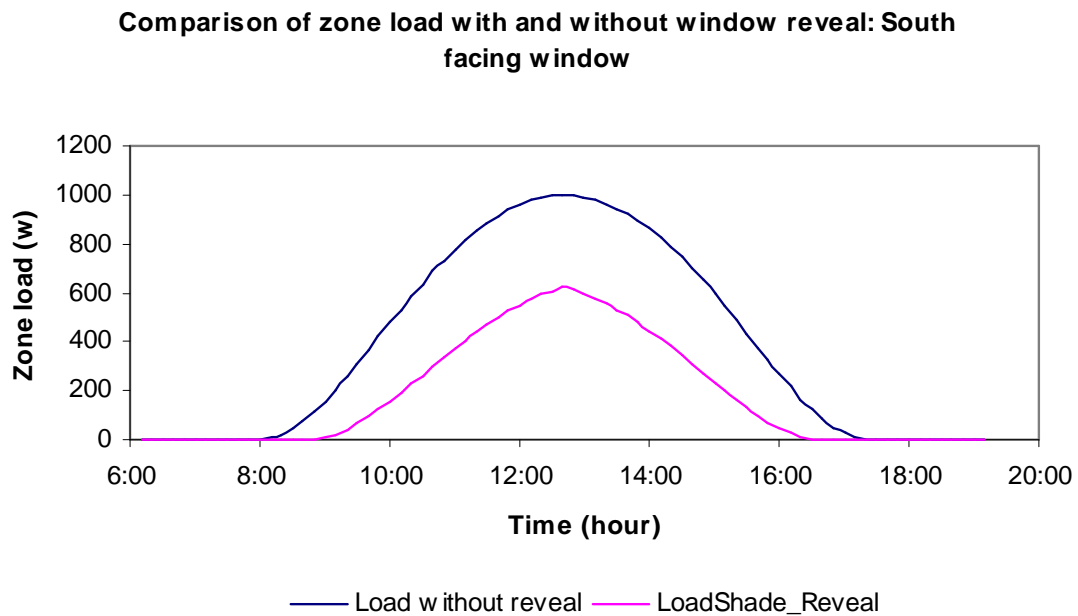


Figure WinReveal –2: Analytical prediction of Zone Load with and without window reveal shading for a south facing window.

The tabulated results included below in table WinReveal-D have been produced with the following test parameters shown in table WinReveal –C where the surface is west facing. The tabulated results include the shaded area and corresponding zone load.

Table WinReveal -C
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	270	Degree
Thickness of the glass	.0023	m
Width of the window reveal ' P '	0.4	m
Width of the window ' W '	2.0	m
Height of the window ' H '	2.0	m
Extinction coefficient of the window surface ' K '	10.0	m ⁻¹
Refractive index of the window surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table WinReveal -D
Tabulated results for the window reveal test

Time	Sunlit Area with Reveal (M ²)	Zone Load without reveal (W)	Zone Load with Reveal (W)
6:10	0.0000	0.00	0.00
...			
12:40	0.0000	0.00	0.00
12:50	0.0000	14.67	0.00
13:00	0.0000	65.98	0.00
13:10	0.0000	149.73	0.00
13:20	0.0000	258.57	0.00
13:30	0.3318	390.53	32.40
13:40	0.7938	532.08	105.58
13:50	1.1752	682.86	200.62
14:00	1.4905	837.77	312.17
14:10	1.7472	988.66	431.85
14:20	1.9694	1139.75	561.15

14:30	2.1580	1284.64	693.07
14:40	2.3208	1421.84	824.96
14:50	2.4644	1552.59	956.56
15:00	2.5964	1679.66	1090.25
15:10	2.7110	1794.63	1216.32
15:20	2.8150	1900.69	1337.59
15:30	2.9127	2000.68	1456.84
15:40	2.9980	2086.22	1563.61
15:50	3.0792	2164.74	1666.44
16:00	3.1546	2232.69	1760.79
16:10	3.2283	2292.49	1850.22
16:20	3.2951	2338.57	1926.46
16:30	3.3598	2373.65	1993.77
16:40	3.4202	2395.45	2048.24
16:50	3.4792	2403.72	2090.73
17:00	3.5350	2396.54	2117.97
17:10	3.5890	2374.61	2130.64
17:20	3.6423	2332.95	2124.32
17:30	3.6816	2270.20	2089.52
17:40	3.6966	2183.08	2017.48
17:50	3.7100	2073.66	1923.33
18:00	3.7226	1925.54	1791.99
18:10	3.7355	1736.64	1621.80
18:20	3.7462	1509.90	1414.08
18:30	3.7561	1212.32	1138.42
18:40	3.7655	865.14	814.43
18:50	3.7730	466.89	440.40
19:00	3.7801	121.11	114.45
19:10	3.7855	0.30	0.28

Figure WinReveal-3 shows the comparison of the zone load with and without reveal using the data listed in table WinReveal-D for a west-facing window.

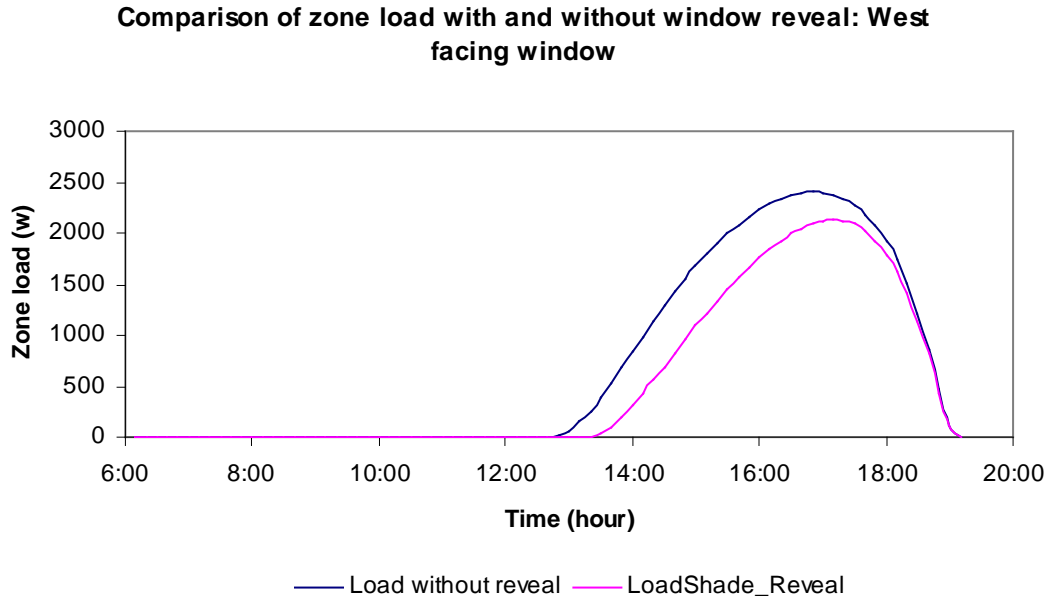


Figure WinReveal –3: Analytical prediction of Zone Load with and without window reveal shading for a west facing window.

Analytical solution

1. Solar beam irradiation on the window is the same as that in the test of Solar Radiation on Glazed Surface.
2. Absorption and transmittance of the window are the same as that in the test of Solar Radiation on Glazed Surface.
3. Heat balance of the window is the same as that in the test of Solar Radiation on Glazed Surface.
4. External window shading

For the case of a window reveal the sunlit area can be calculated the same as a combination of horizontal and vertical semi-infinite fins (Rodriguez and Alvarez 1991, HoF 1997:Ch. 29) as follows:

$$H' = \max(0, H - P \tan \Omega) \quad (\text{WinReveal -1})$$

$$W' = \max(0, W - P \tan \gamma) \quad (\text{WinReveal -2})$$

$$\tan \Omega = \frac{\tan \beta}{\cos \gamma} \quad (\text{WinReveal -3})$$

$$A_{\text{sunlit}} = W' \times H' \quad (\text{WinReveal -4})$$

Where,

W = window width, m

H = window height, m
 P = depth of the window reveal, m
 β = solar elevation (or altitude), Radians
 Ω = profile angle, Radians
 γ = wall solar azimuth angle, Radians, $\gamma = |\phi - \psi|$ and $\gamma \in [0, \pi/2]$
 ϕ = solar azimuth, measured east from north, Radians
 ψ = surface azimuth, measured east from north, Radians
 H' = sunlit rectangle height, m
 W' = sunlit rectangle width, m
 A_{sunlit} = sunlit rectangle area, m^2

The window reveal geometry and angles are shown in Figure WinReveal-4.

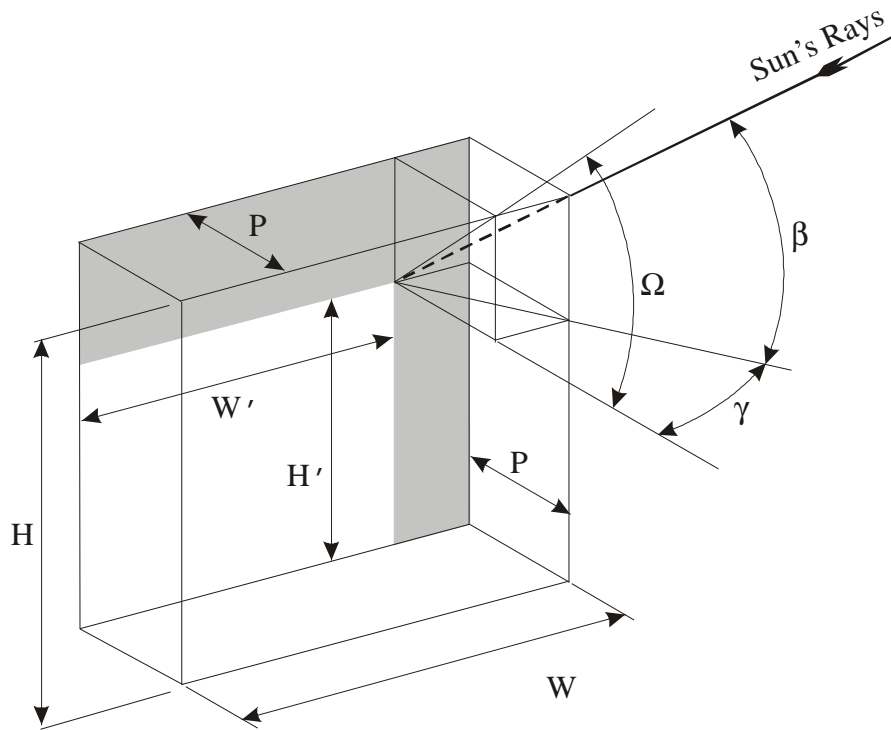


Figure WinReveal -4: Window reveal geometry and dimensions

The zone load is then the total heat flux multiplied by the new sunlit area:

$$Zoneload' = A_{sunlit} \cdot q_{total}'' \quad (\text{WinReveal -5})$$

References

- HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. Atlanta, GA.
- Rodriguez E. A., S. Alvarez 1991. Solar Shading Analytical Tests (I). IEA21RN136.
- USNO 2001. U.S. Naval Observatory: web site - <http://aa.usno.navy.mil/data/docs/AltAz.html>
- Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test IntSolarDist: Internal Solar Distribution

Objective

The objective of this test is to test the treatment of internal solar distribution and builds on the solar glazing and shading tests. A small window on one surface of the zone is used with both horizontal and vertical shading applied. The dimensions of the window and shade and the depth of the zone are arranged such that no beam solar radiation should fall on any surface other than the wall opposite the window at any time of the day. By making the wall opposite the window massless, and eliminating internal long-wave radiation, the zone load should correspond to the instantaneous solar gains through the glazing. By making all other surfaces in the zone heavyweight it should be possible to determine whether the test program is redistributing the solar gains to other surfaces or not. If solar gains are redistributed to the heavyweight surfaces the peak zone load will be correspondingly reduced and the energy redistributed over the day. Accordingly, when the test program is set to model *no* internal solar redistribution the program zone loads should match those calculated with this test solution.

Analytical Model

Calculation of incident direct solar radiation is calculated from tabulated solar position data in the same way as test ExtSolRad. Transmission and absorption of solar radiation is modified from the SolRadGlazing test simply by the reduction in effective glazed area by the casting of shadows from the shading devices. The effect of shading is calculated in the same way as the SolRadShade test. The geometry of the zone and shading is arranged so that it is not possible for direct (beam) solar radiation to fall on any surface other than the internal surface opposite the window. It is assumed that the surface opposite the window is *massless* and that *no* solar radiation is redistributed internally. Then when the exterior side of the interior surfaces are adiabatic, it is possible to assume the zone load is equal to the instantaneous gains from the glazing.

Zone description

The test zone geometry is cuboid with internal dimensions 3x0.5x3m. Only one surface is to be external. This external surface is selected to be one of the two walls with dimensions 3x3m. The zone is to be 0.5m deep on the walls adjacent the external wall (i.e. in the direction normal to the window surface). A 1.0x1.0m window with semi-infinite horizontal fin at the top and semi-infinite vertical fins on both sides is to be positioned in the middle of the external wall. The depth (perpendicular to the wall surface) of the shading fins is to be 0.5m. The zone and shading geometry is illustrated in Figure IntSolarDist-1. No additional window reveal is present in this test. The opaque part of the external surface is to be adiabatic.

The effects of conduction through opaque surfaces, long wave radiation, infiltration and internal heat gains must be eliminated in this test. The interior and exterior air temperature have to be taken as fixed. The interior and exterior convection coefficients also have to be taken as fixed.

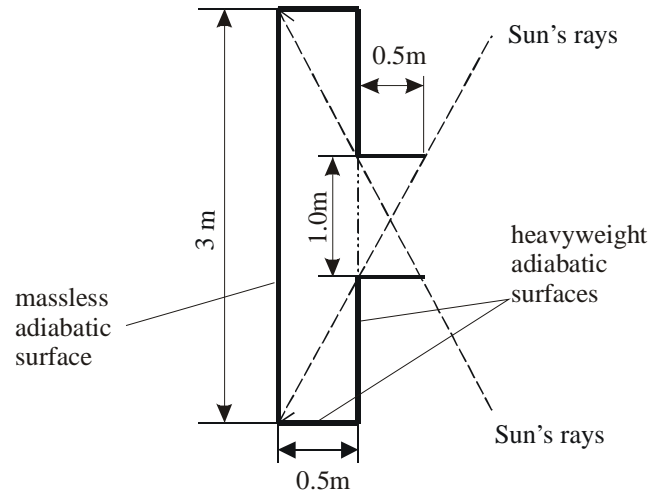


Figure IntSolarDist-1: Zone geometry (plan view) for test IntSolarDist.

Test parameters

The different solar beam irradiation in this test is calculated with solar position data from the US Naval Observatory (USNO 2001) and the beam irradiation at normal incidence from the weather data. The user is able to choose the location of the test zone (from a list of four: Atlanta (W 84° 25', N33° 46'), Boston (W 71° 05', N42° 19'), Chicago (W 87° 41', N41° 51') and Los Angeles (W118° 22', N34° 05')) and two test dates (June 21st, 1999 or August 21st, 1999) when generating the weather files using the toolkit software. The user must also choose the surface tilt angle, the surface azimuth, and the thickness and extinction coefficient and refractive index of the glazed surface.

The inside and outside air temperatures have to be specified as the same and taken as fixed. The same value of the inside air temperature must also be used in the input data of the program to be tested. Also, convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed), should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{IntSolarDist-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A, C and n that apply to both outside and inside convection correlations. The test suite software input screen for this test is shown in Figure IntSolarDist-2.

Figure IntSolarDist-2: Test suite software parameter input screen for test IntSolarDist.

Test output

The principle data of interest in this test are the predicted sunlit area of the window and the predicted cooling load of the test zone. This data is listed in the output file along with the input data.

Test results

The tabulated results included below in table IntSolarDist-B have been produced with the following test parameters shown in table IntSolarDist –A for an east facing external surface.

Table IntSolarDist -A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Location	Atlanta (W 84° 25', N33° 46')	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degrees
Surface azimuth ' ψ '	90	Degrees
Thickness of the window surface	.0023	m
Extinction coefficient of the window surface ' K '	10.0	m ⁻¹
Refractive index of the window surface ' n_g '	1.526	-
Solar absorption of the internal surfaces	1.0	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table IntSolarDist -B
Tabulated results for the internal solar distribution test

Time	Zone Load (W)
6:10	0.01
6:20	17.74
6:30	91.60
6:40	176.61
6:50	250.83
7:00	310.43
7:10	358.85
7:20	394.55
7:30	421.70

7:40	441.04
7:50	453.22
8:00	454.96
8:10	445.10
8:20	430.90
8:30	413.25
8:40	392.08
8:50	368.38
9:00	342.53
9:10	314.66
9:20	285.49
9:30	255.47
9:40	224.41
9:50	192.69
10:00	161.77
10:10	129.44
10:20	99.88
10:30	70.00
10:40	42.18
10:50	16.95
11:00	0.00
...	
19:10	0.00

Figure IntSolarDist–3 shows the zone load using the data listed in table IntSolarDist-B.

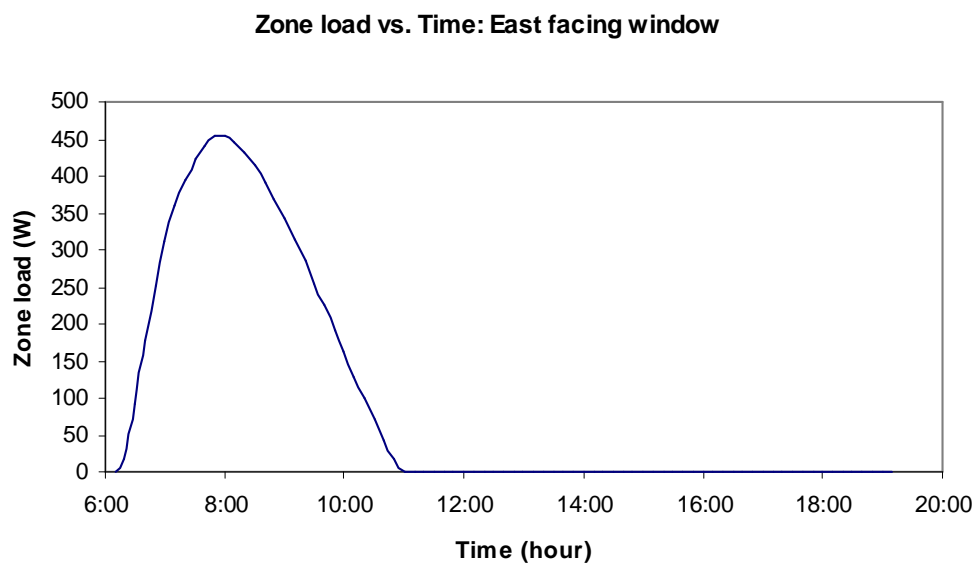


Figure IntSolarDist –3: Resulting zone loads when using the parameters shown in table IndSolarDist-B.

Analytical solution

Calculation of incident direct solar radiation, transmission and absorption of direct solar radiation and effect of shading is in the same way as the window reveal test WinReveal. The geometry of the zone and shading is arranged so that it is not possible for direct (beam) solar radiation to fall on any surface other than the internal surface opposite the window. It is assumed that the surface opposite the window is *massless* and that *no* solar radiation is redistributed internally. It is consequently possible to assume the zone load is equal to the instantaneous gains from the glazing.

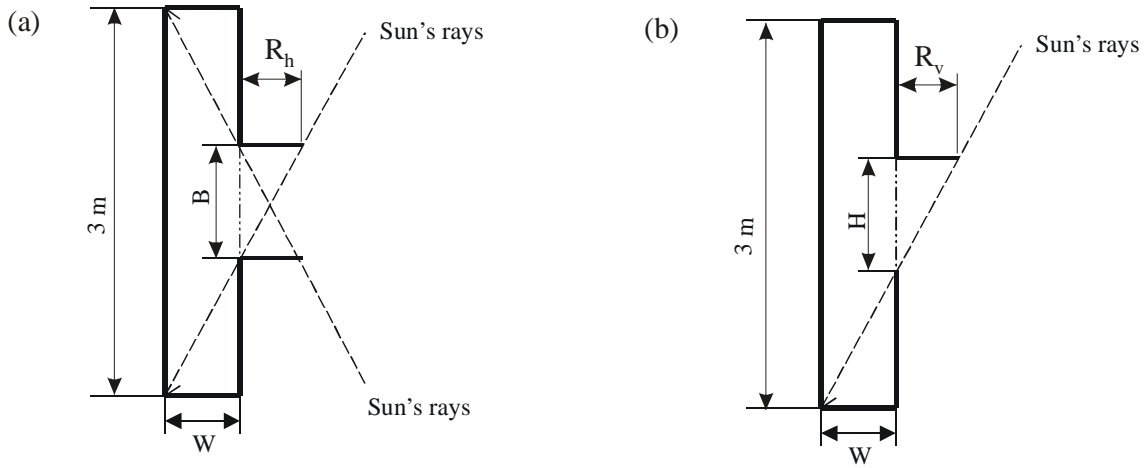


Figure IntSolarDist-4: Zone geometry (a) plan view showing two vertical fins (b) vertical view showing horizontal fin at the top of the window.

For the purpose of a solar distribution on the opposite wall of the external wall, the window shading strategy shown in Figure IntSolarDist-4 is used: an infinite vertical fin with hanging length of R_v on both sides and an infinite horizontal fin with hanging length of R_h at the top. Where R_h and R_v can be calculated as:

$$R_h = H \cdot \frac{2W}{3 - H} \quad (\text{IntSolarDist-2})$$

$$R_v = B \cdot \frac{2W}{3 - B} \quad (\text{IntSolarDist-3})$$

Where

R_v = hanging length of the vertical fin, m

R_h = hanging length of the horizontal fin, m

W = zone width, m

B = window length, m

H = window height, m

The zone geometry in this case is chosen to be 3m x 3m x 0.5m. This results in values for R_h and R_v of 0.5m. For the window shading strategy described above, the new sunlit area can be calculated as follows:

$$B' = \max(0, B - R_v \tan \gamma) \quad (\text{IntSolarDist-4})$$

$$H' = \max(0, H - R_h \tan \delta) \quad (\text{IntSolarDist-5})$$

$$\tan \delta = \frac{\tan \beta}{\cos \gamma} \quad (\text{IntSolarDist-6})$$

$$A_{new} = B' \times H' \quad (\text{IntSolarDist-7})$$

Where,

B = window length, m

H = window height, m

β = solar elevation (or altitude), Radians

γ = wall solar azimuth angle, Radians, $\gamma = |\phi - \psi|$ and $\gamma \in [0, \pi/2]$

ϕ = solar azimuth, measured east from north, Radians

ψ = surface azimuth, measured east from north, Radians

B' = sunlit rectangle length, m

H' = sunlit rectangle height, m

A_{new} = new sunlit rectangle area, m²

The new zone load is then the total heat flux multiplied by the new sunlit area:

$$\text{Zoneload}' = A_{new} \cdot q_{total}'' \quad (\text{IntSolarDist-8})$$

References

Rodriguez E. A., S. Alvarez, 1991. Solar Shading Analytical Tests (I). IEA21RN136.

USNO 2001. U.S. Naval Observatory: web site -

<http://aa.usno.navy.mil/data/docs/AltAz.html>

Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test Infiltration-1: Sensible Infiltration Load – Fixed Infiltration Rate

Objective

The objective of this test is to test the treatment of infiltration under fixed infiltration rate. The infiltration rate is defined in terms of a volumetric flow rate at outside conditions. The analytical solution of the zone load is compared with the results of the program to be tested.

Analytical Model

A steady state calculation is made using a fixed mass flow through the zone from outside. A specific driving force or orifice type and position is not defined. The zone loads are calculated from the mass flow rate and the entering and leaving air enthalpies. The moisture level is assumed constant and it is also assumed that the perfect gas law holds when calculating the air properties.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. At least one of the four walls is to be external. No windows are present in this test.

The effects of solar radiation, long wave radiation, internal heat gains and internal moisture gains must be eliminated in this test. All surfaces of the cube are adiabatic. The interior and exterior air temperature and relative humidity have to be taken as fixed.

Test parameters

The user is able to choose the inside and outside dry bulb temperatures, and outside humidity ratio when generating the weather files and the analytical results using the toolkit software. The outside and inside humidity ratio has to be the same. The same value of the inside air temperature and humidity ratio must also be used in the input data of the program to be tested. Also the volumetric infiltration rate must be specified and taken as fixed. Altitude is taken to be sea level and the atmospheric pressure to be 101325 Pa.

Test output

The principle data of interest in this test are the predicted outside air density, the predicted inside and outside air enthalpies, the predicted mass flow rate of the air and the predicted heating load of the test zone. These data are listed in the output file along with the input data.

Test results

The tabulated results included below in table Infiltration-1-B have been produced with the following test parameters shown in table Infiltration-1-A.

Figure Infiltration-1-1: The test suite software input screen for test Infiltration-1.

Table Infiltration-1-A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Outside air temperature	10	C
Inside air temperature	20	C
Outside humidity ratio	0.0046	-
Infiltration rate	0.5	m ³ /s

Table Infiltration-1-B
Tabulated results for the infiltration test

Test parameters	Value	Units
Zone load	6304.718	W
Outside air density	1.24316	Kg/m ³
Mass flow rate of the air	0.62158	Kg/s
Inside air enthalpy	31790.66	J/kg dry air
Outside air enthalpy	21647.63	J/kg dry air

Analytical solution

The analytical solution of infiltration under fixed infiltration rate is straightforward: Assume the volumetric infiltration rate based on outside air density is V , the mass flow rate of infiltration air is then:

$$\dot{m} = \rho_{o,m} V \quad (\text{Infiltration-1-1})$$

Where,

\dot{m} = mass flow rate of infiltration air, Kg/s

$\rho_{o,m}$ = outside moist air density, kg/m³

V = volumetric infiltration rate based on outside air density, m³/s

The outside moist air density is calculated as follows (Brandemuehl 1993, HoF 1997: Chapter 6),

$$\rho_{o,m} = \rho_{o,d} (1 + W) \quad (\text{Infiltration-1-2})$$

Where

$\rho_{o,d}$ = outside dry air density, kg/m³

W = humidity ratio of the outside air, dimensionless

While

$$\rho_{o,d} = \frac{P_a}{R_a (T_{o,a} + 273.15)} \quad (\text{Infiltration-1-3})$$

and

$$P_a = \frac{0.62198 P}{(0.62198 + W)} \quad (\text{Infiltration-1-4})$$

Where

P_a = partial pressure of dry air, Pa

P = atmospheric pressure, 101325 Pa

R_a = gas constant for air, 287.055 J/kgK

$T_{o,a}$ = dry bulb temperature of the outside air, C

With the mass flow rate known, the zone load can be solved out with the energy balance as follows:

$$\text{Zoneload} = \dot{m}(i_i - i_o) \quad (\text{Infiltration-1-5})$$

Where

i_i = enthalpy of the inside air, J/kg

i_o = enthalpy of the outside air, J/kg

Both the enthalpy of the inside and outside air are calculated as with the psychrometrics calculation routines taken from (Brandemuehl 1993, HoF 1997: Chapter 6),

$$i = i_{da} + i_{sv} W \quad (\text{Infiltration-1-6})$$

Where

i = enthalpy of the inside or outside air, J/kg

i_{da} = enthalpy of dry air, J/kg

i_{sv} = enthalpy of the saturated water vapor, J/kg

W = humidity ratio of the air, dimensionless

and

$$i_{da} = C_{p,a}T \quad (\text{Infiltration-1-7})$$

$$i_{sv} = H_{fg} + C_{p,v}T \quad (\text{Infiltration-1-8})$$

Where

$C_{p,a}$ = specific heat of air, 1006 J/kgK

$C_{p,v}$ = specific heat of water vapor, 1805 J/kgK

H_{fg} = reference heat of vaporization of water, 2501000 J/kg

T = dry bulb temperature of the air, C

References

- Brandemuehl, M.J., S. Gabel and I. Andresen, 1993. A Toolkit for Secondary HVAC for System Energy Calculations (629-RP). Atlanta, GA: ASHRAE.
- HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

Test Infiltration-2: Infiltration – Stack Effect

Objective

The objective of this test is to test the treatment of infiltration under the pressure difference due to density and height differences resulting from fixed openings in the building fabric. This is done for a single zone with openings at high and low level. The analytical solution of the zone load is compared with the results of the program to be tested.

Analytical Model

A steady state calculation is made of the mass flow through two simple orifice openings at high and low level in the zone. The flow is assumed to be driven by the difference in stack pressures due to the difference in height between the two openings and the difference in temperature inside and outside the zone. The zone loads are calculated from the mass flow rate and the entering and leaving air enthalpies. The humidity ratio is assumed to be the same inside as outside (i.e. no moisture is added or removed in the zone) and it is also assumed that the perfect gas law holds when calculating the air properties.

Zone description

The test zone geometry is a tall cubic with internal dimensions 3x3x10m. Two 0.5x0.2m openings are supposed to be symmetrically at the top and the bottom of the external wall. The external wall is however adiabatic. The geometrical arrangement is shown in Figure Infiltration-2-2. No other openings or source of infiltration are present in this test.

The effects of solar radiation, long wave radiation, internal heat gains and internal moisture gains must be eliminated in this test. All surfaces of the cube are adiabatic. The interior and exterior air temperature and relative humidity have to be specified as steady-state values.

Test parameters

The driving force of the infiltration in this test is the “stack pressure” due to the density difference caused by the inside and outside dry bulb temperature differences. The user is able to choose the inside and outside dry bulb temperature and the outside humidity ratio when generating the weather files and the analytical results using the toolkit software. The outside and inside humidity ratio has to be the same. The same value of the inside air temperature and humidity ratio must also be used in the input data of the program to be

tested. Altitude is taken to be sea level and the atmospheric pressure to be 101325 Pa. The input screen from the test suite software for this test is shown in Figure Infiltration-2-1 below.

The screenshot shows the 'ASHRAE Analytical Test Toolkit (1052-RP)' window. The 'Infiltration 2' tab is selected in the top menu bar. The 'Test Temperatures' section contains input fields for 'Inside temperature' (20.0 °C) and 'Outside temperature' (10.0 °C). The 'Humidity ratio' section contains an input field for 'Outside humidity ratio' (0.0046 Kg/kg dry air). A 'Start Calculation' button is located at the bottom center. The status bar at the bottom indicates 'Status: TMY2 SI Units'.

Figure Infiltration-2-1: Test suite software input screen for test Infiltration-2.

Test output

The principle data of interest in this test are the predicted height of neutral pressure level, the predicted mass flow rate and the predicted infiltration load of the test zone. These data are listed in the output file along with the input data.

Test results

The tabulated results included below in table Infiltration-2-B have been produced with the following test parameters shown in table Infiltration-2-A.

Table Infiltration-2-A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Outside air temperature	10	C
Inside air temperature	20	C
Outside air humidity ratio	0.0046	-
Discharge coefficient “ C_D ”	0.6	-
Flow exponent “ x ”	0.65	-

Table Infiltration-2-B
Tabulated results for the infiltration test

Test parameters	Value	Units
Height of neutral pressure level	4.9345	m
Mass flow rate	0.1489	kg/s
Zone load	1510.301	W

Analytical solution

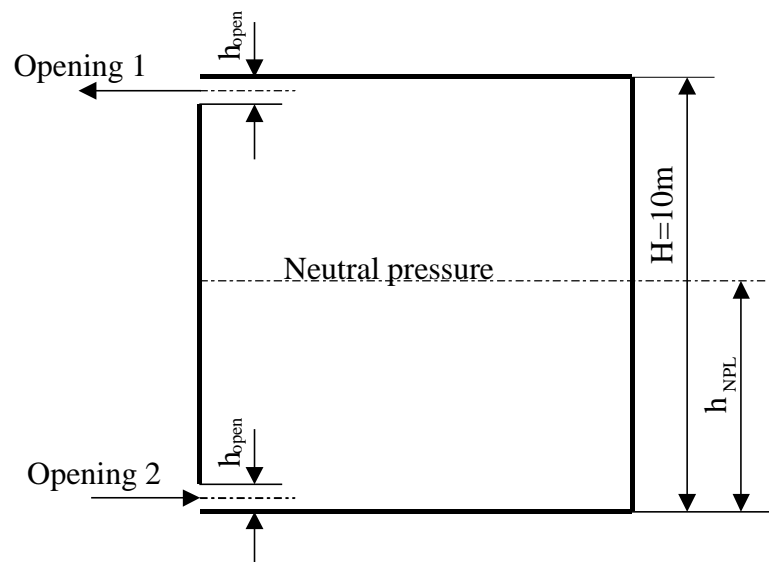


Figure Infiltration 2-2: Diagram of the test zone and openings.

As shown in Figure Infiltration-2-2, the “stack pressures” (between the neutral pressure level and the midpoint of the opening) at opening 1 and 2 are:

$$\Delta P_1 = (\rho_o - \rho_i)g(H - h_{NPL} - \frac{h_{open}}{2}) \quad (\text{Infiltration-2-1})$$

$$\Delta P_2 = (\rho_o - \rho_i)g(h_{NPL} - \frac{h_{open}}{2}) \quad (\text{Infiltration-2-2})$$

Where,

ΔP_1 = “stack pressure” at opening 1, pa

ΔP_2 = “stack pressure” at opening 2, pa

ρ_o = outside air density, kg/m³

ρ_i = inside air density, kg/m³

g = gravitational constant, =9.8 m/s²

H = wall height of the zone, m

h_{NPL} = height of the neutral pressure level, m

h_{open} = height of the opening, m

The mass flow rate through the openings can be approximated by the following power law correlation:

$$\dot{m}_1 = C\sqrt{\rho_i}(\Delta P_1)^x \quad (\text{Infiltration-2-3})$$

$$\dot{m}_2 = C\sqrt{\rho_o}(\Delta P_2)^x \quad (\text{Infiltration-2-4})$$

Where

\dot{m}_1 = mass flow rate through the opening 1, kg/s

\dot{m}_2 = mass flow rate through the opening 2, kg/s

x = flow exponent, its value theoretically should be between 0.5 and 1.0. Large openings are characterized by values very close to 0.5, while values are near 0.65 for very small openings (Walton 1989). The test case assumes this value of 0.65.

C = flow coefficient, $C = C_D A \sqrt{2}$

A = area of the opening, m²

C_D = discharge coefficient of the opening, a dimensionless number depends on the opening geometry and the Reynolds number of the flow. When the opening is small, C_D equals 0.6 for a wide range of Reynolds number (Walton 1982). The test case assumes this value of C_D .

There is an instantaneous mass balance over the zone so that,

$$\dot{m}_1 = \dot{m}_2 \quad (\text{Infiltration-2-5})$$

Assuming the opening 1 and 2 have the same flow coefficients and flow exponents, equation (Infiltration-2-3) and (Infiltration-2-4) can be substituted into (Infiltration-2-5):

$$C\sqrt{\rho_i}(\Delta P_1)^x = C\sqrt{\rho_o}(\Delta P_2)^x \quad (\text{Infiltration-2-6})$$

We can then substitute equation (Infiltration-2-1) and (Infiltration-2-2) into (Infiltration-2-6) and write (assuming that the air density is a function of temperature only):

$$C\sqrt{\rho_i}\left((\rho_o - \rho_i)g\left(H - h_{NPL} - \frac{h_{open}}{2}\right)\right)^x = C\sqrt{\rho_o}\left((\rho_o - \rho_i)g\left(h_{NPL} - \frac{h_{open}}{2}\right)\right)^x$$

(Infiltration-2-7)

Rearranging gives,

$$\left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2}} = \left(\frac{h_{NPL} - \frac{h_{open}}{2}}{H - h_{NPL} - \frac{h_{open}}{2}}\right)^x$$

(Infiltration-2-8)

which can be written as,

$$\left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}} = \frac{h_{NPL} - \frac{h_{open}}{2}}{H - h_{NPL} - \frac{h_{open}}{2}}$$

(Infiltration-2-9)

This can be rearranged to find the height of neutral pressure level:

$$h_{NPL} = \frac{\left(H - \frac{h_{open}}{2}\right) \cdot \left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}} + \frac{h_{open}}{2}}{1 + \left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}}}$$

(Infiltration-2-10)

The height of the neutral pressure level can then be substituted back into equation (Infiltration-2-1) and (Infiltration-2-2) to get the “stack pressures” at opening 1 and 2:

$$\Delta P_1 = (\rho_o - \rho_i)g \frac{(H - h_{open})}{1 + \left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}}}$$

(Infiltration-2-11)

$$\Delta P_2 = (\rho_o - \rho_i)g \frac{(H - h_{open}) \left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}}}{1 + \left(\frac{\rho_i}{\rho_o}\right)^{\frac{1}{2x}}}$$

(Infiltration-2-12)

The mass flow rate through the openings can be calculated by substituting the expressions of “stack pressure”, equation (Infiltration-2-11) and (Infiltration-2-12) into equation (Infiltration-2-3) and (Infiltration-2-4) respectively:

$$\dot{m}_1 = C \sqrt{\rho_i} \left((\rho_o - \rho_i) g \frac{(H - h_{open})}{1 + \left(\frac{\rho_i}{\rho_o} \right)^{\frac{1}{2x}}} \right)^x \quad (\text{Infiltration-2-13})$$

$$\dot{m}_2 = C \sqrt{\rho_o} \left((\rho_o - \rho_i) g \frac{(H - h_{open}) \left(\frac{\rho_i}{\rho_o} \right)^{\frac{1}{2x}}}{1 + \left(\frac{\rho_i}{\rho_o} \right)^{\frac{1}{2x}}} \right)^x \quad (\text{Infiltration-2-14})$$

With the mass flow rate known, the zone load can be solved out with the energy balance either as:

$$\text{Zoneload} = \dot{m}_1 (i_i - i_o) \quad (\text{Infiltration-2-15})$$

Or as:

$$\text{Zoneload} = \dot{m}_2 (i_i - i_o) \quad (\text{Infiltration-2-16})$$

Where

i_i = enthalpy of the inside air, J/kg dry air

i_o = enthalpy of the outside air, J/kg dry air

Enthalpies are calculated using the formulation given in test Infiltration-1 (Brandenmuehl 1993, HoF 1997:Chapter 6).

References

- Brandenmuehl, M.J., S. Gabel and I. Andresen, 1993. A Toolkit for Secondary HVAC for System Energy Calculations (629-RP). Atlanta, GA: ASHRAE.
- HoF, 1997. *1997 ASHRAE Handbook Fundamentals*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.
- Walton, G.N. 1989. Airflow Network Models for Element-Based Building Airflow Modeling. *ASHRAE Transactions*, 95(2): 611-620.
- Walton, G.N. 1982. Airflow and Multi-room Thermal Analysis. *ASHRAE Transactions*, 88(2): 78-91.

Test ExtLWRad: External Long Wave Radiation

Objective

The objective of this test is to test the treatment of external long wave radiation between a horizontal roof surface and the sky. It is intended that the external surface to be tested is massless surface with high conductivity and the test will build on the steady state conduction test. At the outer surface heat exchange with the environment is by convection and by long wave radiation to the sky.

Analytical Model

Heat transfer between the external surface and the environment is assumed to be by a combination of convection to the outside air and long wave radiation to the sky. A single sky temperature is used to model the long wave radiation using a single band gray surface analysis. The net steady state heat balance on both surfaces of the roof is calculated, assuming one-dimensional conduction through the roof, to find the surface temperatures and resultant zone load.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. No windows are present in this test. The external surface to be tested will be a horizontal roof, which has a single material layer.

The effects of solar irradiation, infiltration, internal long wave radiation, and internal heat gains must be eliminated in this test. Except for the external surface, all other surfaces of the cube are adiabatic and all surfaces have no thermal mass. The interior and exterior air temperature have to be taken as fixed.

Test parameters

When generating the weather files and the analytical results using the toolkit software, the sky temperature, the inside and outside air temperatures have to be specified and taken as fixed. The same value of the inside air temperature must also be used in the input data of the program to be tested. Also, the user has to specify the thermal conductivity, thickness and emissivity of the external surface. Convection coefficients have to be specified for the inside and outside surfaces. Most convection correlations used for exterior and interior building heat transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed), should be reducible to the form,

$$h = A + C \left(T_s - T_\infty \right)^n \quad (\text{ExtLWRad-1})$$

where,

h = convection coefficient, W/(m²K)

A = constant, W/(m²K)

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, C

T_∞ = air temperature, C

In the case of exterior convection correlations, this form may be obtained by setting the wind speed to zero. The user must provide the parameters A , C and n that apply to both outside and inside convection correlations.

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

SSCConv | SSCCond | TC 1 | TC 2 | TC 3 | ExtSolRad | SolRadGlazing | SolRadShade | IntSolarDist | **ExtLWRad** | IntLWRad | IntHeatGain | Infiltration 1 | Infiltration 2 | GrdCouple | WinReveal

Test Temperatures

Inside temperature: 20.0 (°C)

Outside temperature: 20.0 (°C)

Sky temperature: 5.0 (°C)

Convection Coefficients

	Coefficient 'A'	Coefficient 'C'	Coefficient 'n'	Units
Inside correlation	0.0	1.49	0.345	(W/m ² .K)
Outside correlation	0.0	0.84	0.333	(W/m ² .K)

Surface Properties

Surface emissivity: 0.9

Thermal conductivity: 1.0 (W/m.K)

Thickness: 0.1 (m)

Start Calculation

Status: TMY2 SI Units

Figure ExtLWRad-1: Test suite software parameter input screen for test ExtLWRad.

Test output

The principle data of interest in this test are the predicted internal, external surface temperature, and zone load. These data are listed in the output file along with the input data.

Test result

The tabulated results included below in table ExtLWRad-B have been produced with the following test parameters shown in table ExtLWRad-A.

Table ExtLWRad-A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Emissivity of the external surface	0.9	-
Thermal conductivity of the external surface	1.0	W/mK
Thickness of the external surface	0.1	m
Sky temperature	5.0	C
External air temperature	20	C
Internal air temperature	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.84	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.49	-
Inside correlation exponent 'n'	0.345	-

Table ExtLWRad-B
Tabulated results for the exterior solar radiation test

Test parameters	Value	Units
Heat flux	-17.7940	W/m ²
Zone load	-160.1457	W
Inside surface temperature	13.6787	C
Outside surface temperature	11.8993	C
Inside convection coefficient	2.8149	W/m ² K
Outside convection coefficient	1.6858	W/m ² K

Analytical solution

The analytical solution used in this test is based on the one-dimensional steady state heat conduction. We consider three components of the heat balance:

1. External surface heat balance

We assume there is no solar irradiation incident on the roof. The horizontal roof has a view factor between the roof and the ground of zero and between the roof and the sky of one. The sky is assumed to have an emissivity of one. Therefore the net long wavelength radiation flux into the roof can be found from (Incropera and DeWitt 1990):

$$q''_{\text{radiation,out}} = \varepsilon \sigma (T_{\text{sky}}^4 - T_{s,o}^4) \quad (\text{ExtLWRad} - 2)$$

Where

$q''_{\text{radiation,out}}$ = the external long wavelength radiation flux, W/m²

ε = the long wavelength emissivity of the roof, dimensionless

σ = Stefan-Boltzmann constant, W/m²K⁴

T_{sky} = the sky temperature, K

$T_{s,o}$ = the outside surface temperature, K

The external heat balance is then the external convection flux plus the external long wavelength radiation flux equals the external conduction flux :

$$q''_{c,out} = h_{c,o} (T_{a,o} - T_{s,o}) + \varepsilon \sigma (T_{\text{sky}}^4 - T_{s,o}^4) \quad (\text{ExtLWRad} - 3)$$

Where

$q''_{c,out}$ = external conduction heat flux, W/m²

$h_{c,o}$ = the external convection coefficient, W/m²K

$T_{a,o}$ = external air temperature , C

$T_{s,o}$ = external surface temperature , C

2. Internal surface heat balance

Since internal long wave radiation and internal heat gains are eliminated from the test zone the internal surface heat flux can be expressed as:

$$q''_{c,in} = h_{c,i} (T_{s,i} - T_{a,i}) \quad (\text{ExtLWRad-4})$$

where,

$q''_{c,in}$ = internal conduction heat flux, W/m²

$h_{c,i}$ = interior convection coefficient, W/m²K

$T_{s,i}$ = internal surface temperature , C

$T_{a,i}$ = internal air temperature , C

3. The steady state conduction heat transfer

The steady state conduction heat flux can be found from the thermal resistance and surface temperature difference of the fabric:

$$q_c'' = (T_{s,o} - T_{s,i}) / R_f \quad (\text{ExtLWRad-5})$$

where,

q_c'' = the conduction heat flux of the fabric, W/m²

$T_{s,o}$ = external surface temperature, C

$T_{s,i}$ = internal surface temperature, C

R_f = the thermal resistance of the fabric, Km²/W, $R_f = L/K$

L = the thickness of the fabric, m

K = the thermal conductivity of the fabric, W/m²K

In the steady state the conductive and convective fluxes are constant and equal so that,

$$q_c'' = q_{c,out}'' = q_{c,in}'' \quad (\text{ExtLWRad - 6})$$

Given this heat balance we can use equations (ExtLWRad-3) through (ExtLWRad-4) to write two further heat balance equations:

$$\varepsilon\sigma T_{s,o}^4 + \left(h_{c,o} + \frac{1}{R_f} \right) T_{s,o} - \frac{1}{R_f} T_{s,i} - \varepsilon\sigma T_{sky}^4 - h_{c,o} T_{a,o} = 0 \quad (\text{ExtLWRad - 7})$$

$$\frac{1}{R_f} T_{s,o} - \left(h_{c,i} + \frac{1}{R_f} \right) T_{s,i} + h_{c,i} T_{a,i} = 0 \quad (\text{ExtLWRad - 8})$$

Rearrange (ExtLWRad – 8) to get:

$$T_{s,i} = \frac{R_{c,i}}{R_f + R_{c,i}} T_{s,o} + \frac{R_f}{R_f + R_{c,i}} T_{a,i} \quad (\text{ExtLWRad - 9})$$

We substitute (ExtLWRad – 9) into (ExtLWRad – 7) to get:

$$\varepsilon\sigma T_{s,o}^4 + \frac{R_f + R_{c,i} + R_{c,o}}{R_f R_{c,o} + R_{c,i} R_{c,o}} T_{s,o} - \frac{T_{a,i} R_{c,o} + T_{a,o} R_{c,i} + T_{a,o} R_f}{R_f R_{c,o} + R_{c,i} R_{c,o}} - \varepsilon\sigma T_{sky}^4 = 0 \quad (\text{ExtLWRad - 10})$$

where,

$$R_f = L/K \quad (\text{ExtLWRad - 11})$$

$$R_{c,o} = 1/h_{c,o} \quad (\text{ExtLWRad - 12})$$

$$R_{c,i} = 1/h_{c,i} \quad (\text{ExtLWRad - 13})$$

The equation (ExtLWRad – 10) is nonlinear with respect to outside surface temperature and requires application of a non-linear equation solving technique such as the Newton-

Raphson method. With the external surface temperature known from (ExtLWRad – 10), we can substitute it into (ExtLWRad – 9) to get the internal surface temperature.

Knowing both the outside and inside surface temperature, we can either use (ExtLWRad – 4) or (ExtLWRad – 5) to compute the convective or conductive heat fluxes respectfully. The zone load is found from either of these fluxes multiplied by the relevant surface area (9.0 m^2).

References

- Incropera, F.P. and D.P. DeWitt, 1990. *Introduction to Heat Transfer*. 2nd Ed. New York: John Wiley.
- McQuiston F.C., J.D. Parker, J.D. Spitler 2000. *Heating, Ventilating, and Air Conditioning: Analysis and Design*. Wiley, New York.
- Yazdanian, M. and J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test IntLWRad: Interior Long Wave Radiation

Objective

The objective of this test is to test the treatment of long wave radiation within the zone using zone geometries of different aspect ratio. Energy enters the zone through one external surface. The heat flow is driven by a difference between the specified outside and inside dry bulb temperatures. The analytical solution of the steady state zone load is compared with the results of the program to be tested.

Analytical Model

Heat is exchanged between the zone and the outside environment by convection at the outside surface and conduction through the wall. Convection at the outside and inside surfaces is treated using a fixed convection coefficient. Conduction through the lightweight wall is treated by a one-dimensional steady-state analysis. All the inside surfaces exchange long wave radiation with each other and transfer heat by convection to the room air. The room air temperature is fixed. Radiation is treated in a single band with gray surfaces and the zone air is taken to be non-participating. The view factors between the internal surfaces in this test are calculated with an exact view factor based method.

Zone description

The test zone geometry is cuboid with any specified aspect ratio [3x3x(3x aspect ratio)]. No windows are present. Only one surface of the cuboid is selected to be exterior surface and all other surfaces are adiabatic. The exterior surface (and opposite surface) are square, with width and height 3m. Aspect ratio is defined as the width (or height) of the exterior wall to the length of the adjacent surfaces. Hence the adjacent surfaces are 3 x Aspect Ratio (m) long. The geometry is illustrated in Figure IntLWRad-3.

Both the inside and outside dry bulb temperatures and the convection coefficients have to be specified as constant values. A number of combinations of inside surface emissivities are to be tested. The effects of solar radiation, infiltration and internal heat gains must be eliminated in this test.

Test parameters

The user, when generating the analytical results using the toolkit software, is able to choose the aspect ratio of the cuboid. Three internal surface emissivities can be specified: that of the external wall, the opposite end wall, and that of the other surfaces (side walls, floor and ceiling). In addition, the outside air temperature, the zone air temperature, the external and internal convection coefficient have to be specified.

Test output

The principle data of interest in this test are the predicted view factors between the internal surfaces, the predicted temperature, radiative heat flux, radiative energy, convective heat flux on each surface and the predicted zone load. These data are listed in the output file along with the input data. The parameter input screen for this test is shown in Figure IntLWRad-1.⁸

Test results

The tabulated results included below in Tables IntLWRad-2 through IntLWRad-7 have been produced with the test parameters shown in table IntLWRad-1. The tabulated results include the surface view factors for different aspect ratios and combinations of surface emissivity. The external convection coefficient is that given by the relationship published by Yazdanian and Klems (1994) with 2 m/s air speed and 10K temperature difference. The internal convection coefficient is that reported by McQuiston *et al.* (2000) Table 8-8.

The screenshot shows the 'ASHRAE Analytical Test Toolkit (1052-RP)' window. The 'IntRad' tab is selected. The input fields are as follows:

Section	Parameter	Value	Units
Test Temperatures	Inside temperature	20.0	(°C)
	Outside temperature	40.0	(°C)
Convection Coefficients	Outside coefficient	21.26	(W/m².K)
	Inside coefficient	3.18	(W/m².K)
Surface emissivities	External surface	0.9	
	The surface opposite to the external	0.1	
	Other surfaces	0.3	
Aspect ratio	Zone aspect ratio	5.0	

At the bottom, there is a 'Start Calculation' button and a status bar showing 'Status: TMY2 SI Units'.

Figure IntLWRad-1: The test suite input screen for test IntLWRad.

⁸ This test requires longer run time than other tests (approx. one minute on a 800MHz PentiumIII PC). The progress bar may not move for much of this period but this is to be expected.

Table IntLWRad -1

Test parameters used in generating the tabulated results

Test parameters	Value	Units
Width of the cuboid	3.0	m
Outside air temperature	40	C
Inside air temperature	20	C
External convection coefficient	21.16	W/m ² K
Internal convection coefficient	3.18	W/m ² K

Table IntLWRad-2

Zone View Factors for Aspect Ratio = 1.0

Surface Index*	Surface Area (m ²)	1	2	3	4	5	6
1	9.0	0	0.199825	0.200044	0.200044	0.200044	0.200044
2	9.0	0.199825	0	0.200044	0.200044	0.200044	0.200044
3	9.0	0.200044	0.200044	0	0.199825	0.200044	0.200044
4	9.0	0.200044	0.200044	0.199825	0	0.200044	0.200044
5	9.0	0.200044	0.200044	0.200044	0.200044	0	0.199825
6	9.0	0.200044	0.200044	0.200044	0.200044	0.199825	0

Table IntLWRad-3

Zone View Factors for Aspect Ratio = 2.0

Surface Index*	Surface Area (m ²)	1	2	3	4	5	6
1	9.0	0	0.068590	0.232853	0.232853	0.232853	0.232853
2	9.0	0.068590	0	0.232853	0.232853	0.232853	0.232853
3	18.0	0.116426	0.116426	0	0.285875	0.240636	0.240636
4	18.0	0.116426	0.116426	0.285875	0	0.240636	0.240636
5	18.0	0.116426	0.116426	0.240636	0.240636	0	0.285875
6	18.0	0.116426	0.116426	0.240636	0.240636	0.285875	0

Table IntLWRad-4
Zone View Factors for Aspect Ratio = 5.0

Surface Index*	Surface Area (m ²)	1	2	3	4	5	6
1	9.0	0	0.012404	0.246899	0.246899	0.246899	0.246899
2	9.0	0.012404	0	0.246899	0.246899	0.246899	0.246899
3	45.0	0.049380	0.049380	0	0.359167	0.271037	0.271037
4	45.0	0.049380	0.049380	0.359167	0	0.271037	0.271037
5	45.0	0.049380	0.049380	0.271037	0.271037	0	0.359167
6	45.0	0.049380	0.049380	0.271037	0.271037	0.359167	0

Table IntLWRad-5
Zone View Factors for Aspect Ratio = 10.0

Surface Index*	Surface Area (m ²)	1	2	3	4	5	6
1	9.0	0	0.003162	0.249209	0.249209	0.249209	0.249209
2	9.0	0.003162	0	0.249209	0.249209	0.249209	0.249209
3	90.0	0.024921	0.024921	0	0.386383	0.281888	0.281888
4	90.0	0.024921	0.024921	0.386383	0	0.281888	0.281888
5	90.0	0.024921	0.024921	0.281888	0.281888	0	0.386383
6	90.0	0.024921	0.024921	0.281888	0.281888	0.386383	0

Table IntLWRad-6
Zone View Factors for Aspect Ratio = 20.0

Surface Index*	Surface Area (m ²)	1	2	3	4	5	6
1	9.0	0	0.000794	0.249801	0.249801	0.249801	0.249801
2	9.0	0.000794	0	0.249801	0.249801	0.249801	0.249801
3	180.0	0.012490	0.012490	0	0.400259	0.287381	0.287381
4	180.0	0.012490	0.012490	0.400259	0	0.287381	0.287381
5	180.0	0.012490	0.012490	0.287381	0.287381	0	0.400259
6	180.0	0.012490	0.012490	0.287381	0.287381	0.400259	0

* Surface indices are given as labels in Figure IntLWRad-3. Surface 1 is the external wall surface, surface 2 is opposite the external wall, 3 & 4 are the long side walls, 5 is the ceiling and 6 is the floor.

Three cases with different surface emissivities have been computed to show how the zone load is sensitive to internal long wave radiation and surface emissivities. The results are given in Table IntLWRad-7, and plotted in Figure IntLWRad-2.

Where

- For Case 1: emissivity of the exterior surface = 0.9
emissivity of the surface opposite to the external surfaces = 0.1
emissivity of other surfaces = 0.3
- For Case2: emissivity of the exterior surface = 0.9
emissivity of all other surfaces = 0.1
- For Case3: emissivities of all the surfaces = 0.9

Table IntLWRad-7
Zone Loads for the parameters shown in Table IntLWRad-1 and various Aspect Ratios. Case number indicates surface Emissivities.

Aspect Ratio	Zone Load (W)*		
	Case 1	Case 2	Case 3
1	860.15	737.42	977.11
2	949.87	826.27	1029.01
5	1033.21	945.07	1073.53
10	1069.52	1013.44	1091.71
20	1089.88	1057.83	1101.58

* Zone Load = The Sum of {(Convective Flux) x (Surface Area)} for the six surfaces

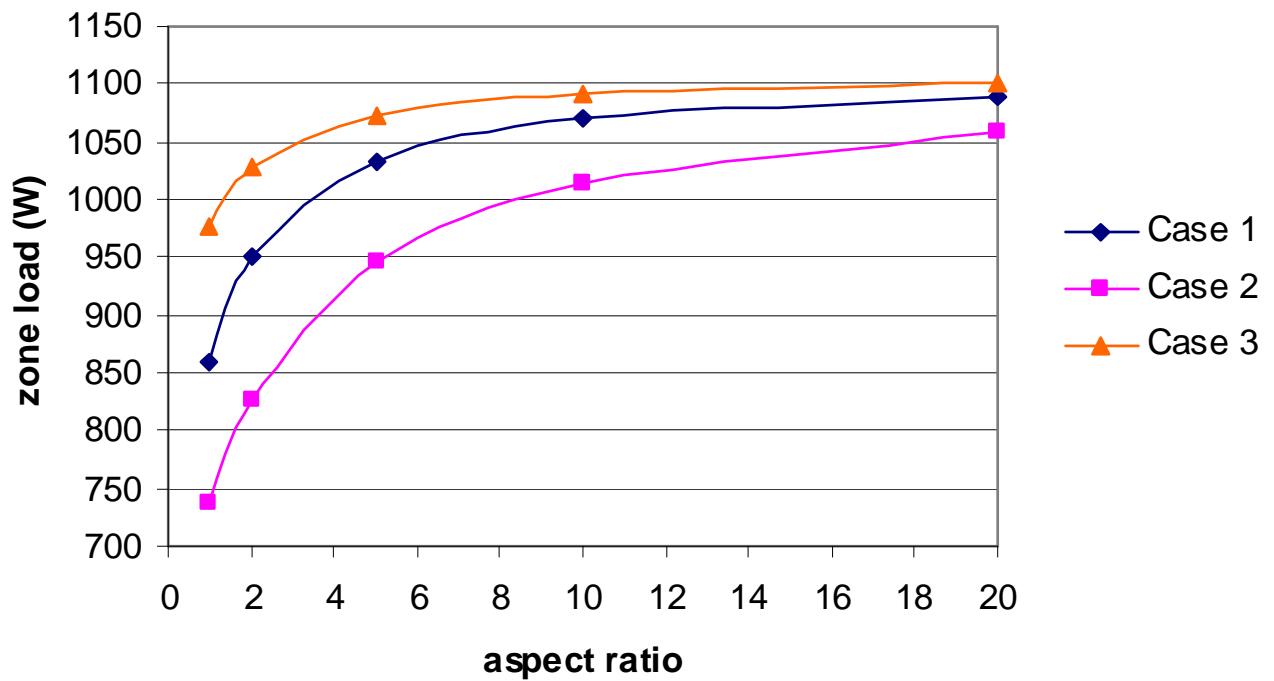


Figure IntLWRad-1: Variation of zone load with aspect ratio for different combinations of surface emissivity.

The detailed results of the temperature, radiative flux, radiative energy and convective flux on the surfaces for each case are showed in Tables IntLWRad-8 through IntLWRad-22.

Table IntLWRad-8
Detailed results for Emissivities as Case 1 and Aspect Ratio = 1.0

Surface Index*	Surface temperature	Radiative energy (W)	Radiative flux (W/m ²)	Convective Flux (W/m ²)
	(C)			
1	35.5046	416.40	46.2665	49.3048
2	21.4088	-40.32	-4.4799	4.4799
3	23.2852	-94.02	-10.4468	10.4468
4	23.2852	-94.02	-10.4468	10.4468
5	23.2852	-94.02	-10.4468	10.4468
6	23.2852	-94.02	-10.4468	10.4468

Table IntLWRad-9

Detailed results for Emissivities as Case 1 and Aspect Ratio = 2.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	35.0357	519.54	57.7267	47.8136
2	20.8653	-24.77	-2.7518	2.7517
3	22.1610	-123.70	-6.8720	6.8719
4	22.1610	-123.70	-6.8720	6.8719
5	22.1610	-123.70	-6.8720	6.8719
6	22.1610	-123.70	-6.8720	6.8719

Table IntLWRad-10

Detailed results for Emissivities as Case 1 and Aspect Ratio = 5.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.6002	615.35	68.3717	46.4286
2	20.3928	-11.24	-1.2492	1.2493
3	21.0554	-151.03	-3.3562	3.3561
4	21.0554	-151.03	-3.3562	3.3561
5	21.0554	-151.03	-3.3562	3.3561
6	21.0554	-151.03	-3.3562	3.3561

Table IntLWRad-11

Detailed results for Emissivities as Case 1 and Aspect Ratio = 10.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.4103	657.11	73.0117	45.8249
2	20.2055	-5.88	-0.6536	0.6536
3	20.5688	-162.80	-1.8089	1.8089
4	20.5688	-162.80	-1.8089	1.8089
5	20.5688	-162.80	-1.8089	1.8089
6	20.5688	-162.80	-1.8089	1.8089

Table IntLWRad-12

Detailed results for Emissivities as Case 1 and Aspect Ratio = 20.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.3039	680.51	75.6118	45.4866
2	20.1052	-3.01	-0.3347	0.3347
3	20.2959	-169.38	-0.9410	0.9410
4	20.2959	-169.38	-0.9410	0.9410
5	20.2959	-169.38	-0.9410	0.9410
6	20.2959	-169.38	-0.9410	0.9410

Table IntLWRad-13

Detailed results for Emissivities as Case 2 and Aspect Ratio = 1.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	36.1461	275.31	30.5899	51.3445
2	21.9238	-55.06	-6.1177	6.1178
3	21.9240	-55.06	-6.1182	6.1183
4	21.9240	-55.06	-6.1182	6.1183
5	21.9240	-55.06	-6.1182	6.1183
6	21.9240	-55.06	-6.1182	6.1183

Table IntLWRad-14

Detailed results for Emissivities as Case 2 and Aspect Ratio = 2.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	35.6817	377.45	41.9389	49.8679
2	21.4226	-40.72	-4.5240	4.5240
3	21.4707	-84.19	-4.6770	4.6769
4	21.4707	-84.19	-4.6770	4.6769
5	21.4707	-84.19	-4.6770	4.6769
6	21.4707	-84.19	-4.6769	4.6769

Table IntLWRad-15

Detailed results for Emissivities as Case 2 and Aspect Ratio = 5.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	35.0608	514.02	57.1129	47.8935
2	20.8040	-23.01	-2.5569	2.5569
3	20.8578	-122.75	-2.7278	2.7279
4	20.8578	-122.75	-2.7278	2.7279
5	20.8578	-122.75	-2.7278	2.7279
6	20.8578	-122.75	-2.7278	2.7279

Table IntLWRad-16

Detailed results for Emissivities as Case 2 and Aspect Ratio = 10.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.7034	592.64	65.8486	46.7568
2	20.4686	-13.41	-1.4903	1.4902
3	20.5059	-144.80	-1.6089	1.6089
4	20.5059	-144.80	-1.6089	1.6089
5	20.5059	-144.80	-1.6089	1.6089
6	20.5059	-144.80	-1.6089	1.6089

Table IntLWRad-17

Detailed results for Emissivities as Case 2 and Aspect Ratio = 20.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.4715	643.65	71.5167	46.0194
2	20.2559	-7.32	-0.8137	0.8137
3	20.2779	-159.09	-0.8838	0.8838
4	20.2779	-159.09	-0.8838	0.8838
5	20.2779	-159.09	-0.8838	0.8838
6	20.2779	-159.09	-0.8838	0.8838

Table IntLWRad-18

Detailed results for Emissivities as Case 3 and Aspect Ratio = 1.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.8934	550.86	61.2064	47.3609
2	23.8485	-110.14	-12.2382	12.2382
3	23.8497	-110.18	-12.2422	12.2421
4	23.8497	-110.18	-12.2422	12.2421
5	23.8497	-110.18	-12.2422	12.2421
6	23.8497	-110.18	-12.2422	12.2421

Table IntLWRad-19

Detailed results for Emissivities as Case 3 and Aspect Ratio = 2.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.6221	610.52	67.8354	46.4984
2	22.0852	-59.68	-6.6310	6.6310
3	22.4058	-137.71	-7.6507	7.6506
4	22.4058	-137.71	-7.6507	7.6506
5	22.4058	-137.71	-7.6506	7.6506
6	22.4058	-137.71	-7.6506	7.6506

Table IntLWRad-20

Detailed results for Emissivities as Case 3 and Aspect Ratio = 5.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.3895	661.69	73.5209	45.7586
2	20.8315	-23.80	-2.6442	2.6442
3	21.1144	-159.47	-3.5439	3.5439
4	21.1144	-159.47	-3.5439	3.5439
5	21.1144	-159.48	-3.5439	3.5439
6	21.1144	-159.48	-3.5439	3.5439

Table IntLWRad-21

Detailed results for Emissivities as Case 3 and Aspect Ratio = 10.0

Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.2942	682.63	75.8483	45.4557
2	20.4110	-11.76	-1.3068	1.3069
3	20.5860	-167.72	-1.8635	1.8635
4	20.5860	-167.72	-1.8635	1.8635
5	20.5860	-167.72	-1.8635	1.8635
6	20.5860	-167.72	-1.8635	1.8635

Table IntLWRad-22

Detailed results for Emissivities as Case 3 and Aspect Ratio = 20.0

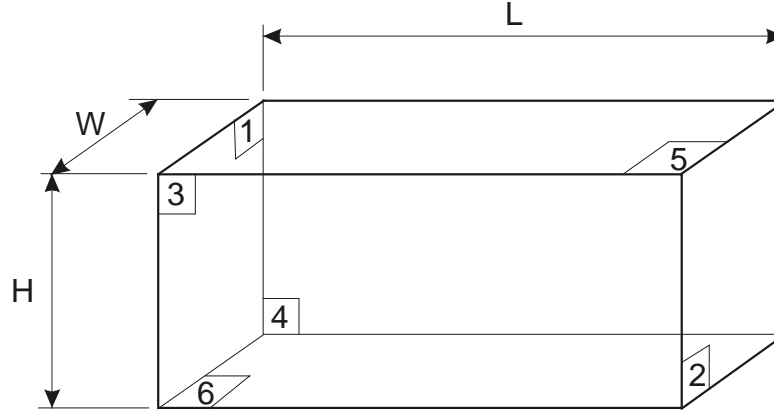
Surface Index*	Surface temperature	Radiative energy	Radiative flux	Convective Flux
	(C)	(W)	(W/m ²)	(W/m ²)
1	34.2428	693.95	77.1059	45.2921
2	20.2040	-5.84	-0.6485	0.6487
3	20.3005	-172.04	-0.9558	0.9557
4	20.3005	-172.04	-0.9558	0.9557
5	20.3005	-172.03	-0.9557	0.9557
6	20.3005	-172.03	-0.9558	0.9557

Analytical solution

Long wave radiation is treated in a single band with gray surfaces and the zone air is taken to be non-participating. Each of the zone surfaces is assumed to have a uniform temperature. The view factors between the internal surfaces are calculated with an exact view factor based method.

1. Calculation of the view factors

Suppose W , L , H is the width (m), length (m) and height (m) of the cuboid respectively. As showed in Figure IntLWRad –1, the areas (m²) of the interior surfaces are as follows:



$$\begin{aligned}
 A_1 &= A_2 = WH \quad (3 \times 3 \text{ m}^2) \\
 A_3 &= A_4 = HL \quad (3 \times 3 \times \text{AR} \text{ m}^2) \\
 A_5 &= A_6 = WL \quad (3 \times 3 \times \text{AR} \text{ m}^2)
 \end{aligned}$$

Figure IntRad-3: The zone geometry and convention used to label the faces of the zone. Surface 1 is the external wall of the zone.

The view factors between the internal surfaces are:

$$\begin{aligned}
 f_{12} = f_{21} = \frac{2}{\pi HW} & \left\{ H \sqrt{L^2 + W^2} \tan^{-1} \frac{H}{\sqrt{L^2 + W^2}} + W \sqrt{L^2 + H^2} \tan^{-1} \frac{W}{\sqrt{L^2 + H^2}} \right. \\
 & \left. - HL \tan^{-1}(H/L) - WL \tan^{-1}(W/L) - \frac{L^2}{2} \ln \left[\frac{(L^2 + W^2 + H^2)L^2}{(W^2 + L^2)(H^2 + L^2)} \right] \right\} \\
 & \quad \quad \quad (\text{IntLWRad-1})
 \end{aligned}$$

$$\begin{aligned}
 f_{13} = f_{14} = f_{23} = f_{24} = \frac{1}{\pi} & \left\{ \tan^{-1}(H/W) + (L/W) \tan^{-1}(H/L) - \frac{\sqrt{L^2 + W^2}}{W} \tan^{-1} \frac{H}{\sqrt{L^2 + W^2}} \right. \\
 & + \frac{W}{4H} \ln \left[\frac{(L^2 + H^2 + W^2)W^2}{(L^2 + W^2)(H^2 + W^2)} \right] + \frac{L^2}{4HW} \ln \left[\frac{(L^2 + H^2 + W^2)L^2}{(L^2 + H^2)(L^2 + W^2)} \right] \\
 & \left. - \frac{H}{4W} \ln \left[\frac{(L^2 + H^2 + W^2)H^2}{(L^2 + H^2)(H^2 + W^2)} \right] \right\} \\
 & \quad \quad \quad (\text{IntLWRad-2})
 \end{aligned}$$

$$f_{15} = f_{16} = f_{25} = f_{26} = \frac{1}{\pi} \left\{ \tan^{-1}(W/H) + (L/H) \tan^{-1}(W/L) - \frac{\sqrt{L^2 + H^2}}{H} \tan^{-1} \frac{W}{\sqrt{L^2 + H^2}} \right. \\ \left. + \frac{H}{4W} \ln \left[\frac{(L^2 + W^2 + H^2)H^2}{(L^2 + H^2)(W^2 + H^2)} \right] + \frac{L^2}{4WH} \ln \left[\frac{(L^2 + W^2 + H^2)L^2}{(L^2 + W^2)(L^2 + H^2)} \right] \right. \\ \left. - \frac{W}{4H} \ln \left[\frac{(L^2 + W^2 + H^2)W^2}{(L^2 + W^2)(W^2 + H^2)} \right] \right\}$$

(IntLWRad-3)

$$f_{34} = f_{43} = \frac{2}{\pi HL} \left\{ L \sqrt{W^2 + H^2} \tan^{-1} \frac{L}{\sqrt{H^2 + W^2}} + H \sqrt{W^2 + L^2} \tan^{-1} \frac{H}{\sqrt{L^2 + W^2}} \right. \\ \left. - LW \tan^{-1}(L/W) - HW \tan^{-1}(H/W) - \frac{W^2}{2} \ln \left[\frac{(W^2 + H^2 + L^2)W^2}{(H^2 + W^2)(L^2 + W^2)} \right] \right\}$$

(IntLWRad-4)

$$f_{31} = f_{41} = f_{32} = f_{42} = \frac{f_{13}A_1}{A_3} = \frac{f_{13}HW}{HL} = \frac{f_{13}W}{L}$$

(IntLWRad-5)

$$f_{35} = f_{36} = f_{45} = f_{46} = \frac{1}{\pi} \left\{ \tan^{-1}(L/H) + (W/H) \tan^{-1}(L/W) - \frac{\sqrt{W^2 + H^2}}{H} \tan^{-1} \frac{L}{\sqrt{W^2 + H^2}} \right. \\ \left. + \frac{H}{4L} \ln \left[\frac{(W^2 + L^2 + H^2)H^2}{(W^2 + H^2)(L^2 + H^2)} \right] + \frac{W^2}{4LH} \ln \left[\frac{(W^2 + L^2 + H^2)W^2}{(W^2 + L^2)(W^2 + H^2)} \right] \right. \\ \left. - \frac{L}{4H} \ln \left[\frac{(W^2 + L^2 + H^2)L^2}{(W^2 + L^2)(L^2 + H^2)} \right] \right\}$$

(IntLWRad-6)

$$f_{56} = f_{65} = \frac{2}{\pi WL} \left\{ L \sqrt{H^2 + W^2} \tan^{-1} \frac{L}{\sqrt{W^2 + H^2}} + W \sqrt{H^2 + L^2} \tan^{-1} \frac{W}{\sqrt{L^2 + H^2}} \right. \\ \left. - LH \tan^{-1}(L/H) - WH \tan^{-1}(W/H) - \frac{H^2}{2} \ln \left[\frac{(H^2 + W^2 + L^2)H^2}{(W^2 + H^2)(L^2 + H^2)} \right] \right\}$$

(IntLWRad-7)

$$f_{51} = f_{52} = f_{61} = f_{62} = \frac{f_{15}A_1}{A_5} = \frac{f_{15}HW}{WL} = \frac{f_{15}H}{L}$$

(IntLWRad-8)

$$f_{53} = f_{54} = f_{63} = f_{64} = \frac{f_{35} A_3}{A_5} = \frac{f_{35} HL}{WL} = \frac{f_{35} H}{W} \quad (\text{IntLWRad-9})$$

The view factor of one surface to itself is zero.

2. Calculation of the heat flux and zone load

With the view factors calculated above, the net radiative heat flux on the internal surfaces and the zone load can be calculated using the *net-radiation method* as follows (Stefanizzi *et al.* 1988, Hottel 1954, Siegel and Howell 1981).

The net radiative energy on the internal surfaces are:

$$[Q] = [D] \cdot [C] \quad (\text{IntLWRad-10})$$

Where

$[Q]$ = the column matrix Q_k , $k = 1, 2, \dots, 6$

Q_k = the net radiative energy on the k th internal surface, W

$$[D] = [B]^{-1}$$

$[B]$ = the square 6×6 matrix with coefficients B_{kj} given by

$$B_{kj} = \left[\frac{\delta_{kj}}{\epsilon_j} - \frac{f_{kj}(1 - \epsilon_j)}{\epsilon_j} \right] \frac{1}{A_j} \quad (\text{IntLWRad-11})$$

and

$$\delta_{kj} = \text{Kronecker's delta, } \delta_{kj} = \begin{cases} = 1, & \text{if } k = j \\ = 0, & \text{otherwise} \end{cases}$$

f_{kj} = view factors from the k th surface to the j th surface, dimensionless

ϵ_j = the emissivity of the j th surface, dimensionless

A_j = the area of the j th surface, m^2

$[C]$ = the column matrix C_k , $k = 1, 2, \dots, 6$, with C_k given by

$$C_k = \sum_{j=1}^6 (\delta_{kj} - f_{kj}) \sigma T_j^4 \quad (\text{IntLWRad-12})$$

where

σ = Stefan-Boltzmann constant, $\text{W/m}^2\text{K}^4$

T_j = the absolute temperature of the j th surface, K

The radiative heat flux (W/m^2) on each internal surface is the net radiative energy on the surface divided by the surface area:

$$q_{k,\text{radiative}}'' = \frac{Q_k}{A_k} \quad k = 1, 2, \dots, 6 \quad (\text{IntLWRad-13})$$

Substituting Q_k into (IntLWRad-13) in detail we get:

$$\begin{aligned}
 q_{k,radiative}'' &= \frac{1}{A_k} \sum_{i=1}^6 \left[D_{ki} \left(\sum_{j=1}^6 ((\delta_{ij} - f_{ij}) \sigma T_j^4) \right) \right] \\
 &= \frac{1}{A_k} \left\{ \sum_{i=1}^6 [D_{ki} (\delta_{i1} - f_{i1}) \sigma T_1^4] + \sum_{i=1}^6 [D_{ki} (\delta_{i2} - f_{i2}) \sigma T_2^4] + \cdots + \sum_{i=1}^6 [D_{ki} (\delta_{i6} - f_{i6}) \sigma T_6^4] \right\} \\
 &= \frac{\sigma}{A_k} \left\{ \sum_{i=1}^6 [D_{ki} (\delta_{i1} - f_{i1})] \right\} T_1^4 + \frac{\sigma}{A_k} \left\{ \sum_{i=1}^6 [D_{ki} (\delta_{i2} - f_{i2})] \right\} T_2^4 + \cdots + \frac{\sigma}{A_k} \left\{ \sum_{i=1}^6 [D_{ki} (\delta_{i6} - f_{i6})] \right\} T_6^4
 \end{aligned}$$

(IntLWRad-14)

The convective heat flux on each internal surface is:

$$q_{k,convective}'' = h_i (T_k - T_{a,i}) \quad k = 1, 2, \dots, 6 \quad (\text{IntLWRad-15})$$

Where

$q_{k,convective}''$ = the convective heat flux on the kth surface, W/m²

h_i = the internal convection coefficient, W/m²K

T_k = the absolute temperature of the kth surface, K

$T_{a,i}$ = the absolute zone air temperature, K

Given that the exterior surface is numbered surface (1) in figure IntLWRad-1, its heat balance can be expressed as:

$$h_o (T_{a,o} - T_1) = q_{1,radiative}'' + h_i (T_1 - T_{a,i}) \quad (\text{IntLWRad-16})$$

which can be rewritten as:

$$q_{1,radiative}'' + (h_i + h_o) T_1 - (h_i T_{a,i} + h_o T_{a,o}) = 0 \quad (\text{IntLWRad-17})$$

Where

h_i = interior convection coefficient, W/m²K

h_o = the exterior convection coefficient, W/m²K

$T_{a,o}$ = external absolute air temperature, K

For each of the other internal surfaces, say the kth surface ($k = 2, 3, \dots, 6$), the heat balance equation is:

$$q_{k,radiative}'' + h_i (T_k - T_{a,i}) = 0 \quad (\text{IntLWRad-18})$$

or,

$$q_{k,radiative}'' + h_i T_k - h_i T_{a,i} = 0 \quad (\text{IntLWRad-19})$$

Equations (IntLWRad-16) and (IntLWRad-18) are actually six nonlinear equations with six unknown surface temperatures. We solve for the surface temperatures using the Newton-Raphson method.

With the surface temperatures known, the convective heat flux on each internal surface can be computed directly with (IntLWRad-15). The zone load is then the summation of convective heat flux on each surface multiplied by surface area (W):

$$Zoneload = \sum_{k=1}^6 A_k q_{k,convective}'' \quad (\text{IntLWRad-20})$$

References

- Hottel, H.C. 1954. *Radiant Heat Transmission*, in William H. McAdams (ed.) “Heat Transmission” 3rd ed. Chap. 4. New York: McGraw Hill.
- McQuiston F.C., J.D. Parker, J.D. Spitler 2000. *Heating, Ventilating, and Air Conditioning: Analysis and Design*. New York: Wiley.
- Siegel, R., J.R. Howell. 1981. *Thermal Radiation Heat Transfer*. 2nd Ed. New York: McGraw Hill.
- Stefanizzi, P., A. Wilson and A. Pinney, 1988. The internal longwave radiation exchange in thermal models. *BRE/SERC Report An Investigation into Analytical and Empirical Validation Techniques for Dynamic Thermal Models of Buildings*. Vol. 2.
- Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

Test IntHeatGain: Internal Heat Gains – Convective and Radiative

Objective

The objective of this test is to find the response to a step change in internal convective and radiative heat gains when the walls of the zone are adiabatic. It is intended that the wall is thermal massive and the test will build on the transient conduction tests. The analytical solution of the zone load is compared with the results of the program to be tested.

Analytical Model

The zone air is assumed to have no thermal mass so that all convective gains from the internal load are added instantaneously to the zone load. The radiant gains are treated as a fixed radiant flux (they are not associated with any real surface) and so they can be equally distributed to all the room surfaces. The radiant fluxes at the walls are treated by calculating the transient conduction into the wall surface using a one dimensional heat conduction model with combined convection and constant flux boundary conditions. All the room surfaces are adiabatic and are assumed to have the same construction.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. No windows are present in this test. The zone fabric must consist of a single homogeneous layer and all the surfaces of the zone must have the same construction. Surface emissivity should have no effect, given the way that the radiant fluxes are imposed.

The internal air temperature and convection coefficient have to be taken as fixed. Zone orientation and other location parameters are also irrelevant in this test. Infiltration gains should be set to zero. There are to be no glazed surfaces.

Test parameters

The internal convective or radiative heat gain has a step change in this test of duration one week (168 hours). The user may choose any one-week period over which to make the test and compare results provided there is sufficient time for the load to return to zero after the gain is removed. The analytical solution is provided for 336 hours. The user is also able to choose the amplitude of the heat gain step when generating the analytical results using the toolkit software.

The user, when calculating the response, is able to set the thermal properties of the wall (conductivity, density and specific heat capacity) in addition to thickness. The internal convection coefficient has to be specified and taken as fixed. This is a common limitation of analytical solutions to transient conduction problems. Also the internal air temperature has to be specified and taken as fixed. The wall is set to be at an initial temperature the same as the internal air temperature. The test suite software input screen for this test is shown below in Figure IntHeatGain-1.

Test output

The principle data of interest in this test are the predicted internal surface temperature, the predicted internal heat flux and the cooling load of the test zone. Zone loads are listed for 100% convective gain and 100% radiative gain along with surface temperature and flux for 100% radiative gain. These data are listed in the output file along with the input data. *Note: no weather file is provided by the test software for this test (although a file could be generated using the test SSConv if necessary).*

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

SSConv SSCond TC 1 TC 2 TC 3 ExtSolRad SolRadGlazing SolRadShade
IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCouple WinReveal

Test Temperatures

Inside temperature 20.0 (°C)

Convection Coefficients

Inside coefficient 3.18 (W/m².K)

Fabric Properties

Conductivity 0.14 (W/m.K)
Density 500.0 (kg/m³)
Specific heat capacity 2500.0 (J/kg.K)
Thickness 0.1 (m)

Zone heat gain

Total heat load 3000.0 (W)
Radiative fraction 0.5 (0~1.0)

Start Calculation

Status: TMY2 SI Units

Figure IntHeatGain-1: Test suite software parameter input screen for test IntHeatGain.

Test result

The tabulated results included below in table IntHeatGain-B have been produced with the following test parameters shown in table IntHeatGain-A. The results include the internal surface temperatures and zone loads for three cases (i) 100% convective gain (ii) 50% radiant, 50% convective gain, and (iii) 100% radiant gain.

Table IntHeatGain -A
Test parameters used in generating the tabulated results

Test parameters	Value	Units
Thermal conductivity	0.14	W/mK
Thickness	0.1	m
Density	500	Kg/m ³
Specific heat capacity	2500	J/kgK
Internal air temperature	20.0	C
Internal convection coefficient	3.18	W/m ² K
Step size of the internal heat gain	3000	W

Table IntHeatGain -B
Tabulated results for the internal heat gain test

Time	Internal surface temperature with 50% radiative heat gain (C)	Internal surface temperature with 100% radiative heat gain (C)	Zone load with 50% radiative heat gain (W)	Zone load with 100% radiative heat gain (W)	Zone load with 100% convective heat gain (W)
0	20.0000	20.0000	1500.00	0.00	3000.00
1	23.1479	26.2994	2040.55	1081.73	3000.00
2	23.9396	27.8829	2176.51	1353.65	3000.00
3	24.4262	28.8560	2260.06	1520.75	3000.00
4	24.7748	29.5533	2319.93	1640.49	3000.00
5	25.0466	30.0969	2366.60	1733.84	3000.00
6	25.2717	30.5470	2405.25	1811.14	3000.00
7	25.4666	30.9369	2438.73	1878.09	3000.00
8	25.6410	31.2857	2468.67	1937.98	3000.00
9	25.8005	31.6047	2496.07	1992.77	3000.00
10	25.9486	31.9009	2521.49	2043.62	3000.00
11	26.0873	32.1783	2545.31	2091.26	3000.00
12	26.2180	32.4398	2567.76	2136.15	3000.00
13	26.3417	32.6870	2588.99	2178.61	3000.00
14	26.4588	32.9214	2609.11	2218.86	3000.00
15	26.5701	33.1438	2628.21	2257.06	3000.00

16	26.6757	33.3552	2646.36	2293.35	3000.00
17	26.7761	33.5560	2663.60	2327.83	3000.00
18	26.8716	33.7469	2679.99	2360.62	3000.00
19	26.9624	33.9285	2695.58	2391.80	3000.00
20	27.0487	34.1011	2710.40	2421.44	3000.00
21	27.1308	34.2653	2724.50	2449.64	3000.00
22	27.2089	34.4215	2737.91	2476.46	3000.00
23	27.2832	34.5700	2750.66	2501.96	3000.00
24	27.3538	34.7113	2762.79	2526.22	3000.00
26	27.4849	34.9734	2785.30	2571.24	3000.00
28	27.6035	35.2106	2805.67	2611.97	3000.00
30	27.7107	35.4252	2824.09	2648.81	3000.00
32	27.8078	35.6193	2840.75	2682.14	3000.00
34	27.8956	35.7949	2855.83	2712.30	3000.00
40	28.1119	36.2275	2892.97	2786.58	3000.00
46	28.2720	36.5478	2920.47	2841.58	3000.00
52	28.3906	36.7849	2940.84	2882.31	3000.00
58	28.4784	36.9605	2955.91	2912.46	3000.00
64	28.5434	37.0905	2967.08	2934.78	3000.00
76	28.6272	37.2581	2981.46	2963.55	3000.00
88	28.6731	37.3499	2989.35	2979.32	3000.00
100	28.6983	37.4002	2993.67	2987.97	3000.00
168	28.7278	37.4593	1498.74	2998.11	3000.00
169	25.5800	31.1600	958.20	1916.40	0.00
170	24.7883	29.5766	822.25	1644.50	0.00
171	24.3018	28.6036	738.70	1477.41	0.00
172	23.9532	27.9064	678.84	1357.69	0.00
173	23.6814	27.3628	632.17	1264.35	0.00
174	23.4564	26.9128	593.53	1187.06	0.00
175	23.2615	26.5230	560.07	1120.13	0.00
176	23.0871	26.1743	530.12	1060.25	0.00
177	22.9277	25.8553	502.74	1005.47	0.00
178	22.7796	25.5592	477.32	954.63	0.00
179	22.6409	25.2819	453.50	907.00	0.00
180	22.5102	25.0205	431.06	862.12	0.00
181	22.3866	24.7733	409.83	819.67	0.00
182	22.2695	24.5389	389.71	779.43	0.00
183	22.1583	24.3165	370.62	741.24	0.00
184	22.0526	24.1053	352.48	704.96	0.00
185	21.9522	23.9045	335.24	670.48	0.00
186	21.8568	23.7136	318.85	637.70	0.00
187	21.7660	23.5321	303.27	606.53	0.00
188	21.6797	23.3595	288.45	576.89	0.00
189	21.5977	23.1953	274.35	548.70	0.00
190	21.5196	23.0392	260.95	521.89	0.00
191	21.4453	22.8907	248.20	496.39	0.00
192	21.3747	22.7495	236.07	472.14	0.00
193	21.3076	22.6151	224.53	449.07	0.00

194	21.2437	22.4874	213.56	427.13	0.00
196	21.1251	22.2502	193.21	386.41	0.00
198	21.0179	22.0357	174.79	349.57	0.00
200	20.9208	21.8417	158.13	316.25	0.00
202	20.8330	21.6661	143.05	286.10	0.00
204	20.7536	21.5073	129.41	258.83	0.00
206	20.6818	21.3636	117.08	234.15	0.00
208	20.6168	21.2336	105.92	211.83	0.00
212	20.5048	21.0096	86.69	173.37	0.00
216	20.4131	20.8263	70.95	141.89	0.00
220	20.3381	20.6763	58.06	116.13	0.00
226	20.2504	20.5007	42.99	85.98	0.00
232	20.1854	20.3707	31.83	63.66	0.00
238	20.1372	20.2745	23.57	47.14	0.00
245	20.0967	20.1933	16.60	33.20	0.00
250	20.0752	20.1505	12.92	25.84	0.00
260	20.0456	20.0912	7.83	15.66	0.00
270	20.0276	20.0553	4.74	9.49	0.00
280	20.0167	20.0335	2.88	5.75	0.00
290	20.0101	20.0203	1.74	3.48	0.00
300	20.0061	20.0123	1.06	2.11	0.00
310	20.0037	20.0075	0.64	1.28	0.00
320	20.0023	20.0045	0.39	0.78	0.00
330	20.0014	20.0027	0.23	0.47	0.00
335	20.0011	20.0021	0.18	0.37	0.00
336	20.0010	20.0020	0.17	0.35	0.00

Figure IntHeatGain-2 shows the plotted result of the zone load vs. time using the data listed in table IntHeatGain - B.

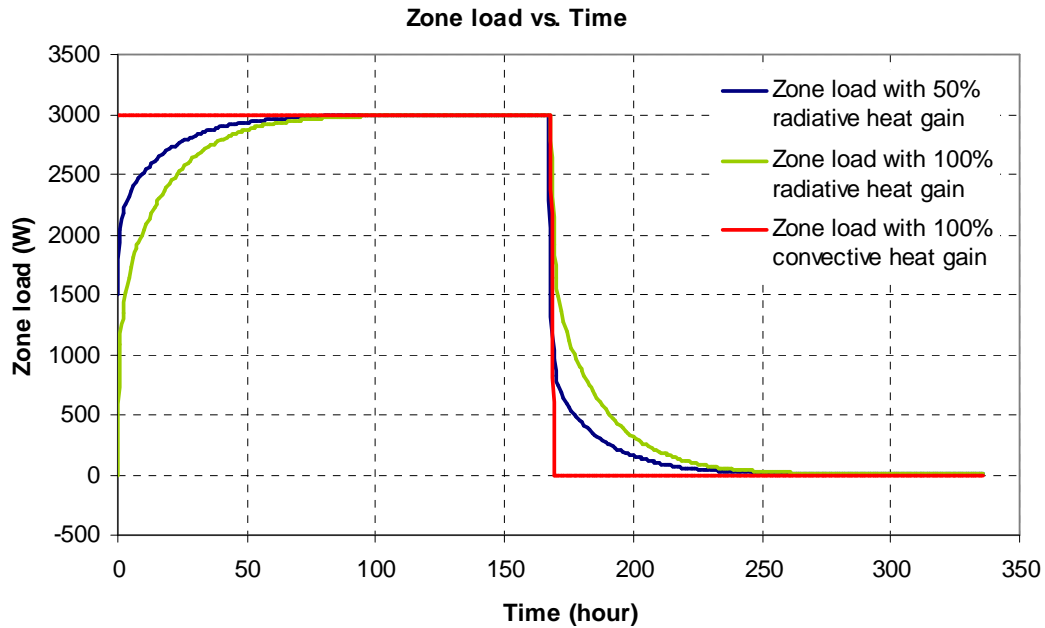


Figure IntHeatGain-2: Transient zone loads with different portions of radiant and convective gain.

Analytical solution

The radiant and convective components of the internal load can be treated separately and then summed for each hour to find the zone load. Convective internal heat gains contribute instantaneously to the zone load. Radiant gains are partly stored in the zone fabric and for adiabatic walls can be analyzed as follows.

For radiative internal heat gains, the analytical solution followed is based on the one-dimensional transient heat conduction. The partial differential equation for this conduction problem can be expressed as:

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha} \frac{\partial T}{\partial t} \quad (\text{IntHeatGain -1})$$

where,

$T = T(x,t)$ temperature distribution of the wall, C

α = heat diffusivity of the wall, m^2/s , $\alpha = \frac{k}{\rho c_p}$

k = conductivity of the wall, W/mK

ρ = density of the wall, Kg/m^3

c_p = specific heat capacity of the wall, J/kgK

t = time, s

The boundary conditions applied are:

$$-k \frac{\partial T(0,t)}{\partial x} = q_{radiative}'' + h_i(T_{a,i} - T(0,t)) \quad (\text{IntHeatGain -2})$$

$$\frac{\partial T(L,t)}{\partial x} = 0 \quad (\text{IntHeatGain -3})$$

Where,

$T_{a,i}$ = internal air temperature, C

$T(0,t)$ = temperature at $x = 0$, C

$T(L,t)$ = temperature at $x = L$, C

L = thickness of the wall, m

h_i = internal convection coefficient, W/m²K

$q_{radiative}''$ = internal radiative flux, W/m²

And,

$$q_{radiative}'' = \begin{cases} 0 & t < t_1 \\ \text{constant} & t_1 \leq t \leq t_2 \\ 0 & t > t_2 \end{cases} \quad (\text{IntHeatGain -4})$$

The initial conditions applied are:

$$T(x,0) = T_{a,i} \quad (\text{IntHeatGain -5})$$

Where,

$T(x,0)$ = temperature at $t = 0$, C

Use the spacial integral transform to solve this problem, the solution is:

$$T(x,t) = \begin{cases} T_{a,i} & t < t_1 \\ T_{a,i} + \sum_{n=1}^{\infty} \frac{A^2 \cos(\lambda_n L) q_{radiative}''}{k \lambda_n^2} [1 - e^{\alpha \lambda_n^2 (t_1 - t)}] \cos[\lambda_n (L - x)] & t_1 \leq t \leq t_2 \\ T_{a,i} + \sum_{n=1}^{\infty} \frac{A^2 \cos(\lambda_n L) q_{radiative}''}{k \lambda_n^2} [e^{\alpha \lambda_n^2 (t_2 - t)} - e^{\alpha \lambda_n^2 (t_1 - t)}] \cos[\lambda_n (L - x)] & t > t_2 \end{cases} \quad (\text{IntHeatGain -6})$$

Where,

λ_n = positive roots of: $\lambda_n \tan(\lambda_n L) = -H_1$, $H_1 = -h/k$

$$A = \sqrt{2\lambda_n} / [\lambda_n L + \sin(\lambda_n L) \cos(\lambda_n L)]^{1/2}$$

The internal surface temperature is then the temperature at $x = 0$, which is:

$$T(0, t) = \begin{cases} T_{a,i} & t < t_1 \\ T_{a,i} + \sum_{n=1}^{\infty} \frac{A^2 \cos^2(\lambda_n L) q_{radiative}''}{k \lambda_n^2} \left[1 - e^{-\alpha \lambda_n^2 (t_1 - t)} \right] & t_1 \leq t \leq t_2 \\ T_{a,i} + \sum_{n=1}^{\infty} \frac{A^2 \cos^2(\lambda_n L) q_{radiative}''}{k \lambda_n^2} \left[e^{-\alpha \lambda_n^2 (t_2 - t)} - e^{-\alpha \lambda_n^2 (t_1 - t)} \right] & t > t_2 \end{cases}$$

(IntHeatGain -7)

Thus the internal heat flux that contributes to the cooling load is:

$$q'' = h(T(0, t) - T_{a,i}) \quad \text{(IntHeatGain -8)}$$

The zone load due to the radiant portion of the internal gains is simply the heat flux q_c'' multiplied by the relevant surface area (9.0 m²). This is added to the convective gains to give the total hourly zone load.

References

Kakac S., Yener Y. 1993. Heat Conduction. Taylor & Francis

Özişik, M.N. 1968. Boundary Value Problems of Heat Conduction. International Textbook Co.

Test GrdCoup: Ground Coupling – Slab on Ground Floor

Objective

The objective of this test is to find the response to the three dimensional steady state heat loss/gain from a slab-on-ground floor.

Analytical Model

It is assumed that the thermal conductivities of the slab and the soil are equal so that the floor and the ground can be treated as a uniform semi-infinite solid. The floor is rectangular and is bounded (but not penetrated) on each side by four equal width external walls. It is assumed that the effect of the walls is to change the ground/slab surface temperature linearly over their thickness. The walls have a finite conductance but its actual value is unimportant.

The ground surface temperature is taken to be the same as the deep ground temperature. The inside air temperature is held constant and the inside floor surface temperature calculated. Heat flux to the floor is via a single convective resistance only. The analytical solution of the zone load is compared with the results of the program to be tested.

Zone description

The test zone geometry is cubic with internal dimensions 3x3x3m. No windows are present in this test. All surfaces of the zone are adiabatic except the floor. The effects of solar irradiation, long wave radiation and infiltration must be eliminated in this test. The inside air temperature, inside floor temperature and outside ground temperature have to be taken as fixed. Zone orientation and other location parameters are also irrelevant in this test.

Test parameters

The user is able to set the thickness and the thermal conductivity of the floor slab when generating the analytical results using the toolkit software. The inside air temperature and the ground temperature have to be specified as steady-state values. It is also assumed that the temperature falls linearly over the thickness of the wall. The same value of the inside air temperature and outside ground temperature must also be used in the input data of the program to be tested. Also, convection coefficients have to be specified for the floor. Most convection correlations used for exterior and interior building heat transfer (e.g. that given by Yazdanian and Klems 1994 at zero air speed) should be reducible to the form,

$$h = A + C(T_s - T_\infty)^n \quad (\text{GrdCoup-1})$$

where,

h = convection coefficient, $W/(m^2K)$

A = constant, $W/(m^2K)$

C = constant, units vary depending on n

n = exponent, non-dimensional

T_s = surface temperature, $^{\circ}C$

T_{∞} = air temperature, $^{\circ}C$

The user must provide the parameters A , C and n that apply to inside convection correlations. The input screen from the test suite software for this test is shown in Figure GrdCoup-1 below.

ASHRAE Analytical Test Toolkit (1052-RP)

File Options Help

SSConv SSCond TC 1 TC 2 TC 3 ExtSolRad SolRadGlazing SolRadShade
IntSolarDist ExtLWRad IntRad IntHeatGain Infiltration 1 Infiltration 2 GrdCoup WinReveal

Test Temperatures

Inside air temperature 25.0 ($^{\circ}C$)

Outside ground temperature 2.0 ($^{\circ}C$)

Convection Coefficients

Coefficient 'A' Coefficient 'C' Coefficient 'n'

Inside correlation 0.0 1.49 0.345 ($W/m^2.K$)

Fabric Properties

Slab floor Conductivity 1.0 ($W/m.K$)

Wall Thickness 0.24 (m)

Start Calculation

Status: TMY2 SI Units

Figure GrdCoup-1: Test suite software input screen for test GrdCoup.

Test output

The principle data of interest in this test are the predicted floor temperature, the predicted conduction heat flux through the floor and the heating or cooling load of the test zone. These data are listed in the output file along with the input data.

Test results

The tabulated results included below in table GrdCoup-B have been produced with the following test parameters shown in table GrdCoup-A.

Table GrdCoup-A Test parameters used in generating the tabulated results

Test parameters	Value	Units
Thermal conductivity of the slab	1.00	W/mK
Thickness of the wall	0.24	m
Inside air temperature	25	C
Ground temperature	2	C
Inside film correlation coefficient 'A'	0.0	W/m ² K
Inside film correlation coefficient 'C'	1.49	-
Inside film correlation exponent 'n'	0.345	-

Table GrdCoup-B Tabulated results for the Ground Coupling test

Test parameters	Value	Units
Inside floor temperature	18.1334	C
Conduction Heat flow through the slab	179.00	W
Zone Load	179.00	W

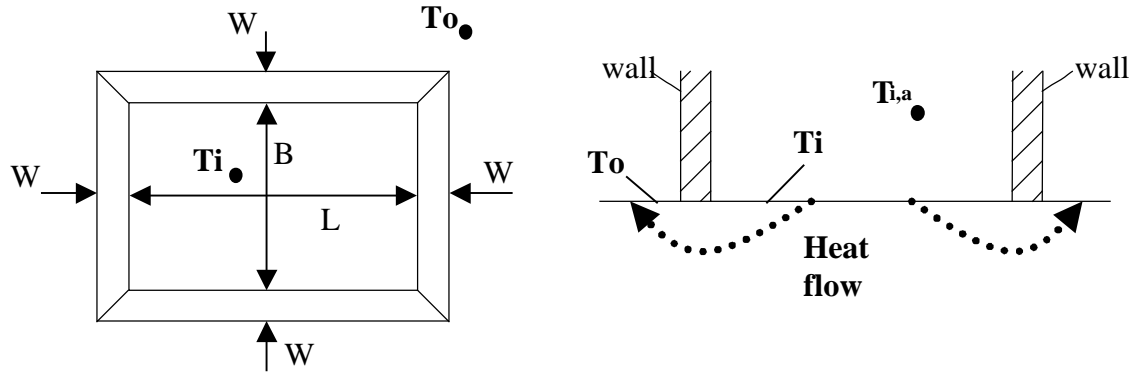


Figure GrdCoup-2: A plan view (left) and section (right) of the ground and wall geometry.

Analytical solution

As shown in Figure GrdCoup-1, given the slab of area $L \times B$ surrounded by a wall of thickness W , assume the surface temperature of the floor is T_i , and the temperature falls linearly from T_i to the ground temperature T_o , the exact solution for the problem under steady state three-dimensional conditions is given by Delsante *et al.* (1983). The total heat flow through the slab into the ground is:

$$Q = k(T_i - T_o) \frac{1}{\pi} F(L, B, W) \quad (\text{GrdCoup -2})$$

Where: T_i = surface temperature of the floor, C
 T_o = temperature of the outside ground, C
 k = conductivity of floor, W/mK

And $F(L, B, W)$ is a function of L , B and W with the units of length (m):

$$\begin{aligned}
F(L, B, W) = & \left(2 + \frac{L+B}{W}\right) \left[(L+W)^2 + (B+W)^2 \right]^{1/2} - \sqrt{2} \left(1 + \frac{L}{2W}\right) \left[L^2 + (L+2W)^2 \right]^{1/2} \\
& - \sqrt{2} \left(1 + \frac{B}{2W}\right) \left[B^2 + (B+2W)^2 \right]^{1/2} - \left(\frac{L+B}{W}\right) (L^2 + B^2)^{1/2} \\
& + \left(\frac{L^2 + B^2}{W}\right) \left[1 + \sqrt{2} \ln(\sqrt{2} - 1) \right] + 2W \left[\sqrt{2} + \ln(\sqrt{2} - 1) \right] \\
& - \frac{\sqrt{2}G^2}{W} \ln \left\{ \frac{[G^2 + (D+2W)^2]^{1/2} + D + 2W}{(G^2 + D^2)^{1/2} + D} \right\} \\
& + \left[\frac{G^2 - (B+W)^2}{W} \right] \times \ln \left\{ \frac{[(L+W)^2 + (B+W)^2]^{1/2} + B + W}{L + W} \right\} \\
& + \left[\frac{G^2 - (L+W)^2}{W} \right] \times \ln \left\{ \frac{[(L+W)^2 + (B+W)^2]^{1/2} + L + W}{B + W} \right\} \\
& + \left[\frac{L(2B - L)}{W} \right] \ln \left\{ \frac{(L^2 + B^2)^{1/2} + B}{L} \right\} + \left[\frac{B(2L - B)}{W} \right] \ln \left\{ \frac{(L^2 + B^2)^{1/2} + L}{B} \right\} \\
& - \left(\frac{L^2 - W^2}{W} \right) \ln \left\{ \frac{[W^2 + (L+W)^2]^{1/2} + W}{L + W} \right\} \\
& - \left(\frac{B^2 - W^2}{W} \right) \ln \left\{ \frac{[W^2 + (B+W)^2]^{1/2} + W}{B + W} \right\} \\
& + (2L + W) \ln \left\{ \frac{[W^2 + (L+W)^2]^{1/2} + L + W}{W} \right\} \\
& + (2B + W) \ln \left\{ \frac{[W^2 + (B+W)^2]^{1/2} + B + W}{W} \right\} \\
& + \frac{\sqrt{2}L^2}{W} \ln \left\{ \frac{[L^2 + (L+2W)^2]^{1/2} + L + 2W}{L} \right\} \\
& + \frac{\sqrt{2}B^2}{W} \ln \left\{ \frac{[B^2 + (B+2W)^2]^{1/2} + B + 2W}{B} \right\}
\end{aligned}$$

(GrdCoup -3)

Where $G = L - B$ and $D = L + B$

The convection heat flow at the floor is:

$$Q_{conv} = h_{c,floor} LB (T_{i,a} - T_i)$$

(GrdCoup -4)

Where

 $T_{i,a}$ = inside air temperature, C

The convection coefficient at the floor being determined from the correlation:

$$h_{c, floor} = A_{c,i} + C_{c,i} \left(T_i - T_{i,a} \right)^{n_i} \quad (\text{GrdCoup-5})$$

In steady state the convection heat flow at the floor equals the conduction heat flow through the floor. The zone load is then simply equal to the total heat flow through the floor.

References

- Davies M.G. 1993. Heat Loss from a Solid Floor: A New Formula. *Building Services Engineering Research Technology*, 14(2): 71-75.
- Delsante A.E., A.N. Stokes, P.J. Walsh, 1983. Application of Fourier Transforms to Periodic Heat Flow into the Ground under a Building. *International Journal of Heat Mass Transfer*, 26(1): 121-132.
- Yazdanian, M., J.H. Klems. 1994. Measurement of the exterior convective film coefficient for windows in low-rise buildings. *ASHRAE Transactions* 100(1): 1087-1096.

INTRODUCTION TO THE ANALYTICAL TEST SUITE SOFTWARE

Although each of the analytical solution can in principle be calculated ‘by hand’ (with a calculator) or with the aid of a spreadsheet the tests have been implemented in the form of a series of subroutines written in Fortran 90. These, in turn, have been compiled as ‘DLL’s that can be called from an interface. The interface allows users to easily enter values of the test parameters. The software basically takes the input test parameters and runs the analytical test and produces a result file along with an accompanying weather file.

Guidance on using the software is provided in the following sections:

INPUT OF THE TEST PARAMETERS

The test suite interface is arranged basically as a series of ‘tabbed’ panes – one for each test. On each tabbed pane there are a series of boxes for text input of the numeric parameter values. These are grouped by parameter type and are labeled along with an indication of the relevant units. Where the number of parameters varies a spreadsheet style grid is provided. An example parameter input screen (Test SSConv) showing the typical text input boxes and input grid is shown below.

The screenshot shows the ASHRAE Analytical Test Toolkit (1052-RP) interface. The title bar reads "ASHRAE Analytical Test Toolkit (1052-RP)". The menu bar includes "File", "Options", and "Help". Below the menu bar is a toolbar with icons for file operations and a help icon. The main window is divided into several tabs: "IntSolarDist", "ExtLWRad", "IntRad", "IntHeatGain", "Infiltration 1", "Infiltration 2", "GrdCoupl", "WinReveal", "SSConv", "SSCond", "TC 1", "TC 2", "TC 3", "ExtSolRad", "SolRadGlazing", and "SolRadShade". The "SSConv" tab is currently selected. The interface is organized into three main sections: "Test Temperatures", "Convection Coefficients", and "Fabric Properties".

Test Temperatures

Inside temperature: 10.0 (°C)
 Outside temperature: 40.0 (°C)

Convection Coefficients

	Coefficient 'A'	Coefficient 'C'	Coefficient 'h'	Units
Inside correlation	0.0	1.49	0.345	(W/m².K)
Outside correlation	0.0	0.84	0.333	(W/m².K)

Fabric Properties

Layer Number	Thickness (m)	Conductivity (W/m.K)
1	0.1	0.1
2	0.05	0.05
3	0.05	0.25

Number of layers: 3

Write

Status: TMY2 SI Units

TEST UNITS SYSTEM

The software allows either the SI or IP units system to be used for input and output of data. The output is always given in the same unit system as the input. The choice of units is determined by selecting either **SI Units** or **IP Units** from the **Options | Units** menu. The currently selected units system is indicated on the menu by a tick mark, and also in the right-hand pane of the status bar (at the bottom of the interface).

STARTING EACH TEST

The calculation of the analytical results for a given test is started using the **WRITE** button at the bottom of the interface. This will start the calculations for the analytical test currently shown in the tabbed pane. This will try and use the current parameters shown in the current tab of the interface. Any typographical errors or missing values in the input are checked at this stage.

When the **WRITE** button is pressed the user will be asked in a dialog for a filename where the test results will be saved. If appropriate to the test, the user will also be asked for a name for the weather data file. Progress in the calculation of the analytical solution and the writing of the weather file is shown (very approximately) in a progress bar. When the progress bar disappears the test calculation is complete.



TEST OUTPUT

Test output takes the form of a text file listing the analytical solution and a weather file in the selected format. When the user starts the test a dialogue box will appear asking for the names of the results file, and then again asking for the name of the weather file. In some cases (e.g. the steady state tests) the analytical result is in the form of single values. For the dynamic tests values are given at a series of time steps. Sometimes these are for each hour of the year, sometime for a single day. It should be reasonably easy to extract this data for plotting along with the results of the building energy simulation program. These output files have been arranged in comma separated variable format (CSV) for easy opening with spreadsheet tools.

TEST WEATHER FILES

Weather files can be output in WYEC2, IVEC, or TMY2 format. It is up to the user to either use these files directly, or convert them further, to suit the building energy program they are using. The choice of format is determined by selecting TMY2, WYEC2 or IVEC format from the Options | Weather File Format menu. The currently select file format is indicated in the center pane of the status bar (see example above).

SAVING TEST PARAMETERS

A facility is provided in the software for saving sets of test parameters. The parameters currently entered or shown in all the panels of the interface can be saved and subsequently reloaded. This can be done by selecting **Save** or **Save As** from the File menu. Previously stored sets of parameters are reloaded using the File | Open menu. The same functionality is provided on the toolbar using the  and  buttons.

Selecting New from the File menu enables the current parameters to be cleared. It is possible to either clear the parameters for all the tests, or only the current test (i.e. the one currently shown on the tabbed panes).

The sets of parameters are stored in a plain text file – normally with a ‘.par’ file extension. A file “Default.par” is provided that contains the default settings. This file is always read when the program starts. Default parameters can be modified by setting the desired parameters and saving using the “Default.par” file name.

Verification of the Analytical Tests

One of the concerns in the development of these analytical tests and the development of the associated software, is the verification of their derivation and implementation. The verification can be attempted by

- manual checking of derivation by other individuals
- checking against existing published solutions
- checking the computer implementation against published results
- cross checking one test implementation against another for similar boundary conditions and parameters
- checking against results from other computer implementations

A summary of how these methods have been used for certain tests is given below

SSConv and SSCond: There are many examples of solutions to this problem in engineering text books. The calculations are clearly easily possible to verify by manual calculation. This has been done for a wide range of parameter values.

TC1: Test TC1 is for the same adiabatic wall test as that described by Bland (1992) although the exact form of the solution is that given by Incropera and DeWitt (1990). Fortunately, Bland published actual values of fluxes and temperatures for one set of parameters in his paper to several significant figures. This implementation of the test has accordingly been crosschecked against this published data.

TC2: This transient solution is original in this particular form for a sequence of temperature steps. It was derived by Dr. Ron Dougherty at OSU and checked partly by being given as a problem in a graduate conduction course. This test has been tested against two other tests. Firstly towards the end of each time step the conduction approaches steady-state conditions. The predicted temperatures and fluxes have been both manually checked and against the predictions of test SSCond. Secondly, if the inside convection coefficient is given a value of zero the boundary conditions become the same as that of test TC1. It has thus been possible to check the predictions of this implementation of TC2 against that of TC1 and results published by Bland.

TC3: This test for periodic sinusoidal excitation share a theoretical basis with the so called 'Admittance Method'. In the first place it has been possible to check various stages of the derivation against a variety of published material. The predicted decrement factors and time lags have also been checked against those predicted by other software (BREADMIT) and published in tables (CIBSE Design Guide A). The test method has also been implemented as a MATLAB program and satisfactory comparison of the results made.

ExtLWRad: The test for exterior long wave radiation has been checked by making hand calculations for a specific hour, and also by implementing the solution in an equation solving package (EES).

ExtSolRad, *SolRadGlazing*, *SolRadShade*: Solar position data for one date (Atlanta 08/21/1999) and three times (7:00am, 12:00pm and 3:00pm) were selected and used to check the calculation of the incident solar radiation and the resultant heat balance by hand calculation. In view of the fact that the other tests involving shading, *IntSolDist* and *WinReveal* use exactly the same theory and implementations of the algorithms, they have not been separately checked.

Infiltration-1 and Infiltration-2: Hand calculations were made to check the psychrometric properties and resultant steady state zone load results of these tests.

IntLWRad: The values of view factors were checked against the tabulated values in *D.C. Hamilton and W.R. Morgan, Radiant Interchange Configuration Factors. NACA TN 2836 (1952)* for aspect ratio of 1, 2, 10, and 20. They are exactly the same to the 4th decimal number (NACA values have only 4 decimal numbers). The view factors were also checked for consistency. $\sum_{j=1}^6 F_{ij}$ ($i=1,...,6$) were calculated to be 1.000001 or 1.000002 or 0.999999 or 0.999998 (Theoretically should be summed up to exactly 1.0). Checks were also made for reciprocity $A_i F_{ij} = A_j F_{ji}$ and found to be satisfactory. With the view factors checked, a spreadsheet calculation was made to check the heat balance for each surface.

IntHeatGain: Firstly the roots of the transcendental equation associated with the transient conduction solution, and calculated by the software, were checked for one particular set of parameters. These roots were used in a spreadsheet calculation to find the temperature and flux response of the zone walls. These results were found to agree satisfactorily with the solution implemented in the toolkit software. No exact analytical solution for this test was found in the literature since our problem has adiabatic boundary condition outside, and both convection and radiative heat flux B.C. inside.

GrdCouple: Hand calculations were made to find the resulting steady state zone loads and check against the implementation in the toolkit software.

Internal Testing of the Test Suite

The analytical tests and their documentation have been initially tested by carrying out the whole test series with one particular whole building energy simulation program. This has been done within the School of Mechanical and Aerospace Engineering at OSU using the BLAST program. The findings of these tests are discussed in this section of the report. A brief summary of the test results follows along with a summary of the principal problems found. Results for each test completed are given in more detail at the end of this section of the report.

Summary of the BLAST Test Results

A brief description of the results of the comparisons for steady state conduction/convection, transient conduction, solar related cases, internal heat gains and internal long wave radiation are given below.

Steady state conduction and convection

The BLAST results match well with the analytical solution both for the Steady State Conduction case (with the maximum difference of 0.06% for zone load in one hour) and for the Steady State Convection case (with the maximum difference of 0.11% for zone load in one hour).

Transient conduction cases

The temperature and zone load responses predicted by BLAST were found to follow the profiles predicted by the analytical solutions. For the TC1 case (adiabatic wall), the maximum difference in internal surface temperature was about 0.5 °C. For the TC2 case (convection heat transfer at both internal and external surfaces), surface temperatures were predicted within 0.6 °C with a driving temperature difference of 30 °C. However, the percentage difference in the predicted zone load was significant for a few hours of each step change.

BLAST results show response to periodic changes in external dry bulb temperature approximately the same as the analytical solution. In the case of test TC3 (Sinusoidal Driving Temperature and Multi-layer Wall), the difference in the zone loads range from 28% to –64%. Efforts were made to improve match by selecting more precise convergence criteria, but without effect.

Solar related cases

In all the solar related tests, except for the problems that will be mentioned later, BLAST responds to exterior solar beam radiation approximately in the same way as analytical solution does and shows the same trend regarding the error analysis. Usually there will be one hour that BLAST doesn't show any load while there is a very small load for analytical solution. Most of the difference in the zone load is less than 3% as a proportion of the peak load, with a maximum difference ranges from 4.01% to 7.62%.

The following table shows the percentage differences in zone load for the solar related cases.

Test	Mean percentage difference in zone load (%)	Maximum percentage difference in zone load (%)
ExtSolRad	1.88	4.01
SolRadGlazing	1.85	4.23
SolRadShade	2.42	6.16
WinReveal	3.01	5.82
IntSolarDist	2.69	7.62

Internal heat gain

BLAST results show it responds to a step change in internal convective and radiative heat gain relatively slower than the analytical solution for the Internal Heat Gain test. The maximum difference is found in the first 40 hours (with differences of 10-37%) and after 168 hours when the load is switched off (with differences of 60-70%). During the period of 41-168 hours the results match well (with differences within $\pm 5\%$ and most less than 2%).

Internal long wave radiation

Three cases with various surface emissivity distribution were compared to BLAST results. It was found that in case 2, where the surface emissivities are all the same (0.9), the BLAST result matches with analytical solution better (with maximum difference of -1.72%) than in the other two cases (with maximum difference of -6.27% for case 1 and -4.33% for case 3) where differential surface emissivities were set. In addition, it was found that BLAST matches with analytical solution better with a larger aspect ratio than a smaller aspect ratio.

Internal Solar Distribution

The results of the IntSolarDist test showed that the different solar distribution options available in the BLAST program operated as expected. Only when the solar distribution was explicitly calculated did the BLAST results match with the analytical solution.

Problems found during Testing with BLAST

In carrying out the tests with the BLAST program, trials with several tests revealed a number of difficulties in using BLAST. These arose mostly from lack of documentation of some features, or erroneous documentation. In one case an error in one of BLAST's models is suspected. These findings may be of interest to general users of the program as well as developers and users of the test suite.

Convergence criteria

Poor results were initially obtained for the TC1 test. It was subsequently found that these could be improved considerably by changing the convergence criteria used by BLAST. The default convergence criteria of BLAST is described as the following in a draft technical report of BLAST (*George Walton 1981, Passive Solar Extension of the Building Loads Analysis and System Thermodynamics (BLAST) Program*):

```
"CONVERGENCE = (Loads, Temp, Iter);
```

Where

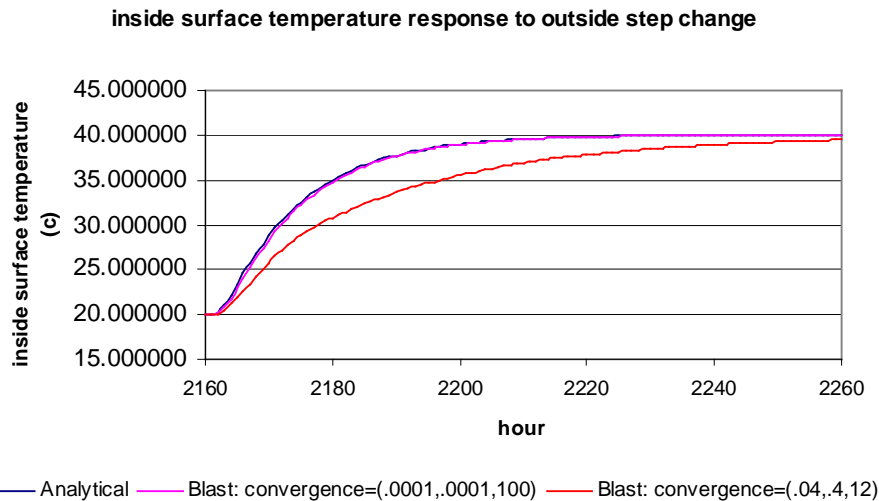
Loads - represents the number at which the loads balance must agree before convergence is reached. Default is 0.04.

Temp - represents the number at which the zone temperatures must agree before convergence is reached. Default is 0.4.

Iter - represents the number of iterations allowed in the heat balance convergence loop before it is decided that the zone will not converge for a particular hour. Default is 12.

Convergence is reached when either the loads or the temperature criterion is satisfied."

Using the default convergence criteria there is significant divergence between the results of BLAST and the analytical solution for test TC1. The following figure shows the difference resulting from various convergence criteria. Note that in the figure, the curve created by the default convergence criteria (.04, .4, 12) has a maximum difference in internal surface temperature of 4.33 °C, with 97 hours out of 100 hours having a difference of more than 0.5 °C. After selecting the convergence criteria of (.0001, .0001, 100), BLAST responds to step changes in external dry bulb temperature in approximately the same way as the analytical solution does. The maximum difference in internal surface temperature is about 0.5 °C, with 80 hours out of 100 hours having a difference within 0.2 °C.



Air density calculation

A significant difference was found between BLAST results and the analytical solution in the Infiltration-1 case where a fixed infiltration rate is specified. It appears this deviation comes from an error in the air density calculation. The following line appears in subroutine SIMZGD (rout33.f) of the BLAST source code, which has the effect of under-predicting the density by about 10%:

$$\text{AIRDEN} = (0.003235 * \text{OBP}(\text{STDTIM})) / (\text{ODB}(\text{STDTIM}) + 273.15)$$

The coefficient 0.003235 should correspond to the inverse of the gas constant ‘R’. The value 0.003235 implies a value for R of 309.1 J/kgK. The value of R should be about 287 J/kgK at room temperature. The value of specific heat Cp of air used by BLAST is 1004 J/kgK. Using the BLAST air density and specific heat Cp, hand calculation of the zone load gives 5809.9 Wh (the BLAST output gives 5810 Wh). Our analytical solution is 6304.7 Wh.

Vertical shading

Comparison of the BLAST results for case SolRadShade with the analytical solution initially showed poor results. It was found out that there appears to be an error in the BLAST manual regarding the syntax used to specify a vertical wing. Under “Wings and Overhangs” in BLAST manual, the following input syntax is given for a vertical wing

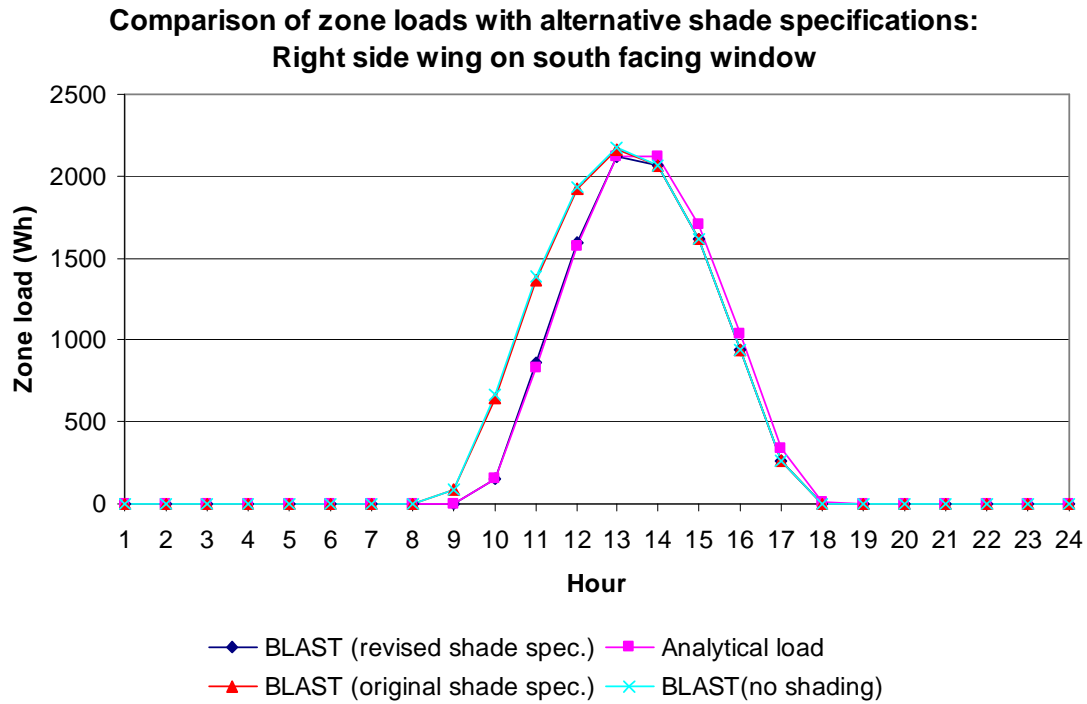
“WITH WINGS(usn1, usn2)”

Where ‘usn1’ is supposed to be the depth and ‘usn2’ the height of the vertical wing. The correct shading effect was only found with the arguments (or there definitions) reversed:

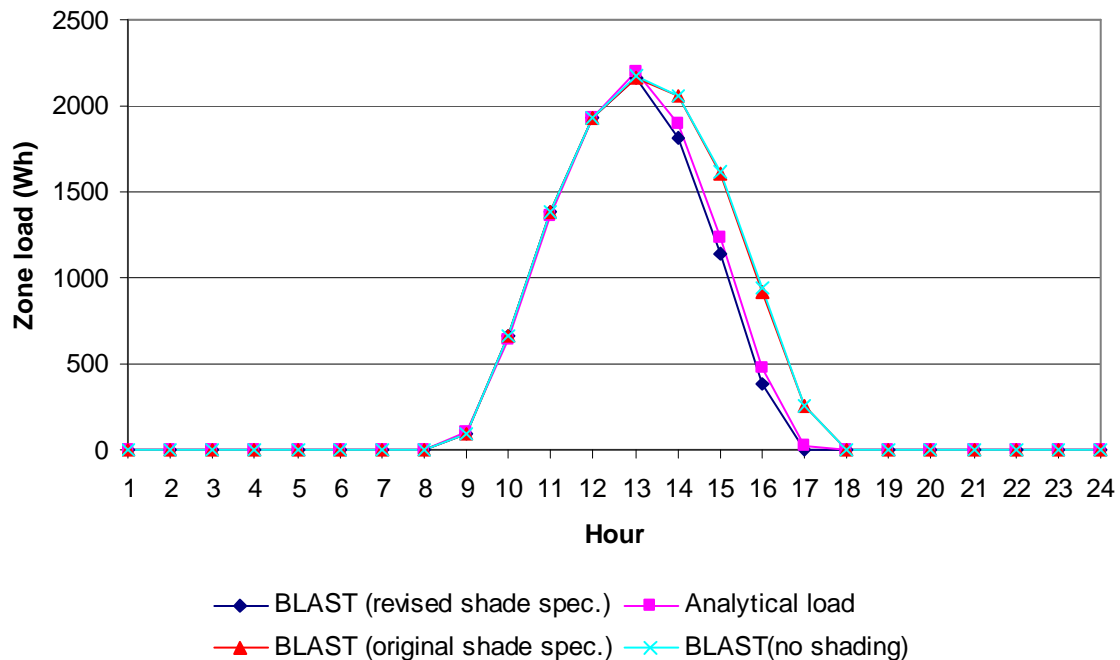
“WITH WINGS(usn2, usn1)”

The following two figures show the result of using the syntax as it appears in the BLAST Manual. In this case only a very small shading effect is observed. This might be expected with a wing that is only 0.6m high at the bottom of the 3x3m window. Using the

alternative syntax - Reversing the argument values - can be seen to agree well with the analytical solution.



Comparison of zone loads with alternative shade specifications: Left side wing on south facing window



Dependence on weather data period

It was noted that the BLAST results for the external solar radiation test ExtSolRad show good agreement with the analytical solution while specifying the weather tape for only a single day (August 21st) in BLAST input file. However, deviation does occur when a weather tape period of more than one week is specified. It is not apparent why this should be.

Weather data processing

In order to complete the tests involving solar irradiation it was necessary to modify the weather data processor (WIFE) used by BLAST. This was necessary to avoid diffuse irradiation being added into the processed weather file. Diffuse irradiation is set to zero in the test weather files so that loads are only induced by direct irradiation. It is acknowledged that this would be a problem for most users and may be an issue with other test programs.

Tests beyond the scope of BLAST

There are two tests from the test suite that BLAST does not have the capability to perform:

Infiltration-2

The Infiltration-2 test can't be performed in BLAST since the only way to introduce infiltration in BLAST input file is by specifying the infiltration rate, as that already done in Infiltration-1 test. No airflow network model is available.

GrdCoup

Although the GrdCoup test has been performed in BLAST, much difference was found between BLAST result and analytical solution. This was expected due to the simplified nature of the treatment of ground coupling in BLAST.

BLAST comparison results: Test SSCond**Test parameters**

Test Parameter	Value	Units
Number of fabric layers	3	-
Thermal conductivity: Layer 1	0.1	W/m.K
Thickness: Layer 1	0.1	m
Thermal conductivity: Layer 2	0.05	W/m.K
Thickness: Layer 2	0.05	m
Thermal conductivity: Layer 3	0.25	W/m.K
Thickness: Layer 3	0.01	m
Inside temperature	10.0	°C
Outside Temperature	40.0	°C
Outside correlation coefficient 'A'	12.49	W/m ² .K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.8	-
Inside correlation coefficient 'A'	3.076	W/m ² .K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.8	-

Tabulated results

Month	Blast result load in one hour	Analytical solution load in one hour
	wh	wh
1	110.4435484	110.379086
2	110.4464286	110.379086
3	110.4435484	110.379086
4	110.4444444	110.379086
5	110.4435484	110.379086
6	110.4444444	110.379086
7	110.4435484	110.379086
8	110.4435484	110.379086
9	110.4444444	110.379086
10	110.4435484	110.379086
11	110.4444444	110.379086
12	110.4435484	110.379086

Comment: The BLAST result matches well with the analytical solution for the Steady State Conduction case with the maximum difference of 0.061% for zone load in one hour.

BLAST comparison results: Test SSConv

Test parameters

Test Parameter	Value	Units
Thermal conductivity	1.0	W/m.K
Thickness	0.1	m
Inside temperature	10.0	°C
Outside Temperature	40.0	°C
Outside correlation coefficient 'A'	12.49	W/m ² .K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.8	-
Inside correlation coefficient 'A'	3.076	W/m ² .K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.8	-

Tabulated results

Month	Blast result load in one hour (wh)	Analytical solution load in one hour (wh)
1	535.080645	534.482452
2	534.970238	534.482452
3	535.080645	534.482452
4	535.000000	534.482452
5	535.080645	534.482452
6	535.000000	534.482452
7	535.080645	534.482452
8	535.080645	534.482452
9	535.000000	534.482452
10	535.080645	534.482452
11	535.000000	534.482452
12	535.080645	534.482452

Comment: The BLAST result matches well with the analytical solution for the Steady State Convection case with the maximum difference of 0.112% for zone load in one hour.

BLAST comparison results: Test Tc1**Test parameters**

Test Parameter	Value	Units
Thermal conductivity	0.14	W/m.K
Density	500	Kg/m ³
Specific heat capacity	2500	J/kg.K
Thickness	0.1	m
Initial temperature (T ₀)	20.0	°C
Temperature step (ΔT)	20.0	°C
External convection coefficient	12.49	W/m ² .K

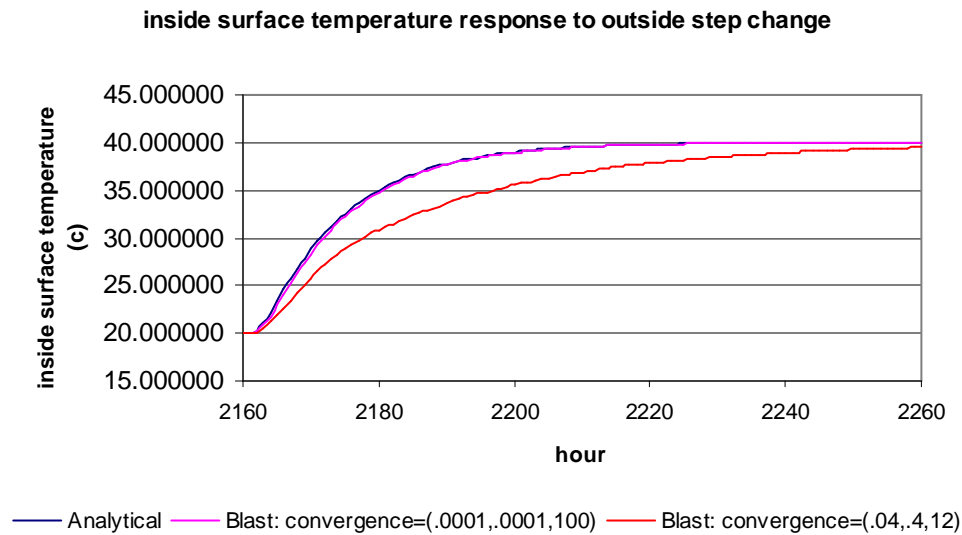
Tabulated results

Hour	Analytical inside surface temperature (C)	Blast inside surface temperature (C) convergence= (.0001, .0001,100)	Blast inside surface temperature (C) convergence= (.04, .4,12)
2160	20.000000	19.999989	19.999989
2161	20.006744	20.003082	20.002361
2162	20.280138	20.178837	20.128485
2163	21.066663	20.792765	20.555151
2164	22.159304	21.754086	21.190338
2165	23.359083	22.883829	21.903849
2166	24.559994	24.058771	22.622448
2167	25.713315	25.212294	23.433750
2168	26.799166	26.312470	24.245653
2169	27.811433	27.346106	25.033482
2170	28.750364	28.309422	25.789238
2171	29.619023	29.203352	26.510593
2172	30.421593	30.030903	27.086498
2173	31.162589	30.796047	27.717106
2174	31.846488	31.502956	28.226753
2175	32.477572	32.155842	28.798552
2176	33.059862	32.758675	29.258961
2177	33.597103	33.315254	29.673983
2178	34.092769	33.829067	30.061079
2179	34.550069	34.303387	30.427204
2180	34.971971	34.741268	30.775881
2181	35.361213	35.145474	31.109221
2182	35.720323	35.518620	31.428659
2183	36.051632	35.863064	31.735273

2184	36.357294	36.181019	32.029915
2185	36.639293	36.474548	32.313294
2186	36.899461	36.745495	32.586014
2187	37.139489	36.995628	32.848629
2188	37.360934	37.226513	33.101616
2189	37.565237	37.439663	33.345398
2190	37.753723	37.636410	33.580372
2191	37.927618	37.818020	33.806900
2192	38.088051	37.985691	34.025322
2193	38.236064	38.140450	34.235951
2194	38.372619	38.283333	34.439091
2195	38.498602	38.415215	34.635021
2196	38.614832	38.536949	34.824009
2197	38.722065	38.649342	35.006306
2198	38.820996	38.753075	35.182152
2199	38.912268	38.848854	35.351795
2200	38.996474	38.937256	35.515453
2201	39.074162	39.018879	35.673344
2202	39.145836	39.094208	35.825665
2203	39.211960	39.163738	35.972618
2204	39.272966	39.227936	36.114395
2205	39.329249	39.287186	36.251175
2206	39.381175	39.341900	36.383141
2207	39.429082	39.392391	36.510456
2208	39.473279	39.438992	36.633289
2209	39.514055	39.482029	36.751793
2210	39.551674	39.521740	36.866119
2211	39.586381	39.558418	36.976429
2212	39.618401	39.592262	37.082855
2213	39.647943	39.623520	37.185535
2214	39.675197	39.652359	37.284603
2215	39.700342	39.678970	37.380180
2216	39.723540	39.703556	37.472397
2217	39.744942	39.726238	37.561367
2218	39.764687	39.747192	37.647202
2219	39.782904	39.766522	37.730015
2220	39.799710	39.784382	37.809917
2221	39.815216	39.800850	37.887005
2222	39.829521	39.816044	37.961372
2223	39.842718	39.830090	38.033127
2224	39.854894	39.843041	38.102356
2225	39.866127	39.855015	38.169151
2226	39.876491	39.866051	38.233593
2227	39.886053	39.876255	38.295769
2228	39.894874	39.885658	38.355755
2229	39.903012	39.894360	38.413631
2230	39.910520	39.902374	38.469471
2231	39.917447	39.909790	38.523342

2232	39.923838	39.916622	38.575321
2233	39.929734	39.922913	38.625465
2234	39.935174	39.928745	38.673843
2235	39.940192	39.934113	38.720520
2236	39.944822	39.939083	38.765556
2237	39.949094	39.943653	38.809006
2238	39.953035	39.947891	38.850925
2239	39.956671	39.951782	38.891373
2240	39.960025	39.955391	38.930393
2241	39.963120	39.958710	38.968041
2242	39.965975	39.961784	39.004360
2243	39.968609	39.964607	39.039402
2244	39.971039	39.967228	39.073212
2245	39.973281	39.969631	39.105831
2246	39.975349	39.971867	39.137302
2247	39.977258	39.973942	39.167667
2248	39.979018	39.975834	39.196964
2249	39.980643	39.977596	39.225227
2250	39.982141	39.979206	39.252499
2251	39.983524	39.980705	39.278809
2252	39.984799	39.982071	39.304195
2253	39.985976	39.983349	39.328686
2254	39.987062	39.984539	39.352314
2255	39.988063	39.985619	39.375111
2256	39.988987	39.986626	39.397106
2257	39.989840	39.987564	39.418327
2258	39.990626	39.988411	39.438801
2259	39.991352	39.989201	39.458553
2260	39.992022	39.989910	39.477608

Plotted result



Comment: The principal findings of this test relate to the convergence criteria used by the BLAST program. The default convergence criteria used by BLAST are described in the draft technical report *Passive Solar Extension of the Building Loads Analysis and System Thermodynamics (BLAST) Program*, G.N. Walton 1981:

“CONVERGENCE = (Loads, Temp, Iter);

Where

Loads - represents the number at which the loads balance must agree before convergence is reached. Default is 0.04.

Temp - represents the number at which the zone temperatures must agree before convergence is reached. Default is 0.4.

Iter - represents the number of iterations allowed in the heat balance convergence loop before it is decided that the zone will not converge for a particular hour. Default is 12.

Convergence is reached when either the loads or the temperature criterion is satisfied.”

For the TC1 case poor results were found with the default convergence criteria. The figure above shows the difference in predicted inside surface temperature using various convergence criteria. Note that in the figure, the curve created by the default convergence criteria (.04, .4, 12) has a maximum difference in internal surface temperature of 4.33 °C, with 97 hours out of 100 hours having an difference more than 0.5 °C. After selecting the convergence criteria of (.0001, .0001, 100), BLAST responds to step changes in external dry bulb temperature in approximately the same way as the analytical solution. The maximum difference in internal surface temperature was about 0.5 °C, with 80 hours out of 100 hours having a difference within 0.2 °C. One can easily see why it’s important to specify the convergence criteria while doing BLAST comparison, although most users would have a difficult time finding documentation for this feature.

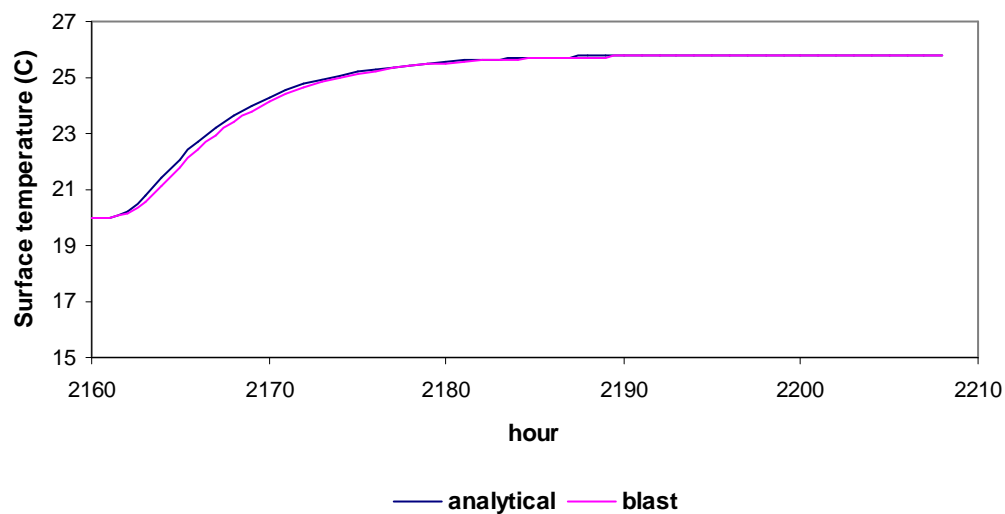
BLAST comparison results: Test Tc2

Test parameters

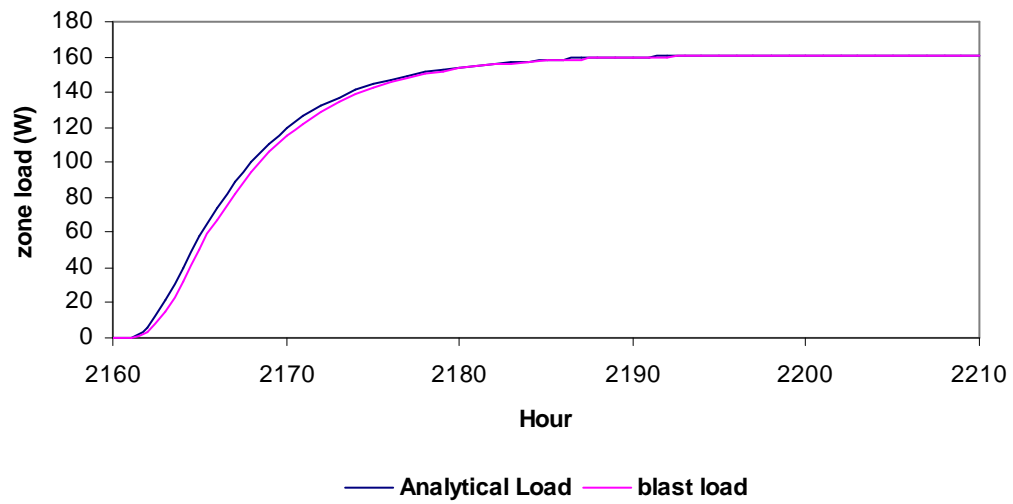
Test Parameter	Value	Units
Thermal conductivity	0.14	W/m.K
Density	500	Kg/m ³
Specific heat capacity	2500	J/kg.K
Thickness	0.1	m
Initial temperature (T_0)	20.0	°C
Temperature step (ΔT)	20.0	°C
External convection coefficient	12.49	W/m ² .K
Internal convection coefficient	3.076	W/m ² .K

Plotted results: First step change

Comparison of inside surface temperature: First step change in outside dry bulb temperature

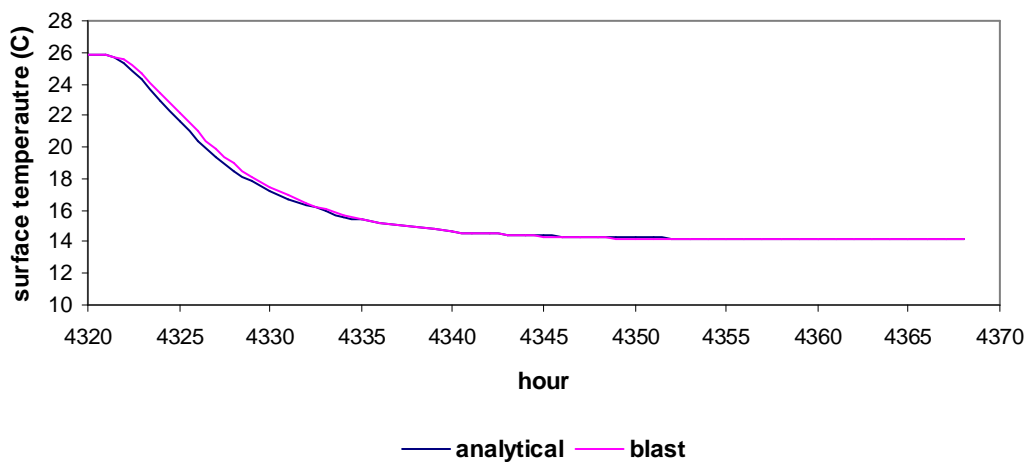


Comparison of zone load: First step change in outside dry bulb temperature

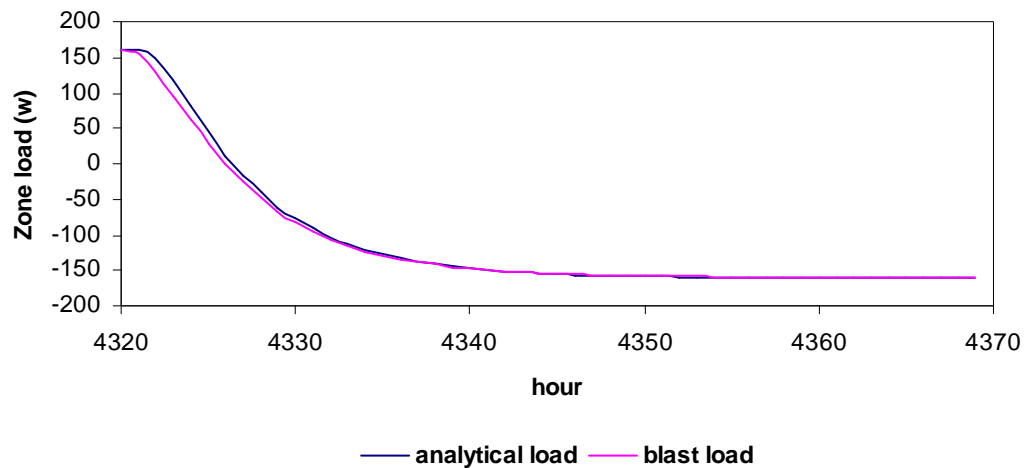


Plotted results: Second step change

Comparison of internal surface temperature: Second step change in outside dry bulb temperature



Comparison of zone load: Second step change in outside dry bulb temperature



Tabulated results: First step

Time	Analytical		Blast	
Hour	Inside surface temperature (C)	Zone load (W)	Inside surface temperature (C)	Zone load (W)
2160	20.000000	0.000000	19.999994	0.000000
2161	20.005854	0.162063	20.002367	0.000000
2162	20.219062	6.064499	20.128361	3.559815
2163	20.765785	21.199994	20.548534	15.210938
2164	21.440940	39.890981	21.154312	32.008301
2165	22.102179	58.196715	21.801237	49.946045
2166	22.694524	74.595208	22.409594	66.813965
2167	23.204977	88.726589	22.948643	81.760254
2168	23.637010	100.686980	23.412413	94.619873
2169	23.999516	110.722615	23.805433	105.517090
2170	24.302398	119.107598	24.135878	114.679443
2171	24.554933	126.098755	24.412550	122.350830
2172	24.765269	131.921715	24.643683	128.759522
2173	24.940369	136.769167	24.836559	134.103760
2174	25.086096	140.803495	24.997383	138.564453
2175	25.207364	144.160662	25.131443	142.281982
2176	25.308270	146.954149	25.243172	145.380615
2177	25.392231	149.278527	25.336281	147.962158
2178	25.462091	151.212541	25.413870	150.114014
2179	25.520219	152.821738	25.478521	151.906982
2180	25.568583	154.160665	25.532394	153.400879
2181	25.608825	155.274713	25.577284	154.645752
2182	25.642308	156.201651	25.614689	155.682861
2183	25.670167	156.972904	25.645857	156.547119

2184	25.693347	157.614620	25.671827	157.267578
2185	25.712634	158.148556	25.693468	157.867432
2186	25.728681	158.592814	25.711500	158.367432
2187	25.742034	158.962456	25.726524	158.784180
2188	25.753143	159.270014	25.739044	159.131592
2189	25.762387	159.525916	25.749477	159.420898
2190	25.770078	159.738837	25.758169	159.661865
2191	25.776477	159.915997	25.765413	159.862793
2192	25.781802	160.063402	25.771450	160.030273
2193	25.786232	160.186050	25.776480	160.169678
2194	25.789918	160.288097	25.780670	160.285889
2195	25.792985	160.373006	25.784163	160.383057
2196	25.795537	160.443653	25.787073	160.463135
2197	25.797661	160.502435	25.789497	160.530762
2198	25.799427	160.551344	25.791517	160.586426
2199	25.800897	160.592038	25.793200	160.633301
2200	25.802120	160.625898	25.794601	160.672119
2201	25.803138	160.654070	25.795769	160.704346
2202	25.803985	160.677511	25.796743	160.731690
2203	25.804689	160.697015	25.797554	160.754150
2204	25.805275	160.713243	25.798229	160.772705
2205	25.805763	160.726745	25.798792	160.788330
2206	25.806169	160.737980	25.799263	160.801514
2207	25.806507	160.747328	25.799654	160.812012
2208	25.806788	160.755105	25.799980	160.821289
2209	25.807021	160.761577	25.800253	160.828857
2210	25.807216	160.766961	25.800480	160.835205

Tabulated results: Second step

Time	Analytical		Blast	
Hou	inside surface temperature (C)	zone load (W)	inside surface temperature (C)	zone load (W)
4320	25.808179	160.793635	25.801603	160.735107
4321	25.796471	160.469510	25.796858	153.746338
4322	25.370056	148.664636	25.544872	130.442383
4323	24.276609	118.393647	24.704542	96.847412
4324	22.926299	81.011673	23.493010	60.971436
4325	21.603822	44.400206	22.199177	27.235840
4326	20.419131	11.603218	20.982472	0.000000
4327	19.398225	-16.659542	19.904369	-26.577637
4328	18.534160	-40.580324	18.951492	-48.669678
4329	17.809146	-60.651594	18.154772	-67.166260
4330	17.203382	-77.421561	17.487616	-82.626953
4331	16.698314	-91.403875	16.929974	-95.532471
4332	16.277641	-103.049795	16.464493	-106.296143
4333	15.927442	-112.744700	16.076250	-115.269775

4334	15.635986	-120.813355	15.752572	-122.749512
4335	15.393452	-127.527688	15.482786	-128.982910
4336	15.191639	-133.114663	15.257949	-134.174072
4337	15.023717	-137.763418	15.070573	-138.504395
4338	14.883996	-141.631446	14.914439	-142.111572
4339	14.767742	-144.849840	14.784337	-145.117432
4340	14.671012	-147.527695	14.675926	-147.622070
4341	14.590529	-149.755791	14.585592	-149.709229
4342	14.523564	-151.609666	14.510319	-151.448242
4343	14.467845	-153.152172	14.447597	-152.897461
4344	14.421485	-154.435604	14.395332	-154.104981
4345	14.382912	-155.503476	14.351783	-155.111084
4346	14.350817	-156.391992	14.315495	-155.949219
4347	14.324112	-157.131276	14.285257	-156.647949
4348	14.301893	-157.746393	14.260061	-157.229736
4349	14.283406	-158.258196	14.239066	-157.715088
4350	14.268023	-158.684040	14.221572	-158.119141
4351	14.255225	-159.038360	14.206994	-158.456299
4352	14.244576	-159.333169	14.194847	-158.736572
4353	14.235715	-159.578464	14.184725	-158.970703
4354	14.228343	-159.782560	14.176291	-159.165772
4355	14.222209	-159.952376	14.169263	-159.327637
4356	14.217105	-160.093671	14.163407	-159.463135
4357	14.212858	-160.211235	14.158527	-159.575684
4358	14.209325	-160.309052	14.154462	-159.669678
4359	14.206385	-160.390441	14.151074	-159.748047
4360	14.203939	-160.458160	14.148252	-159.813232
4361	14.201903	-160.514505	14.145900	-159.867432
4362	14.200210	-160.561387	14.143940	-159.912842
4363	14.198801	-160.600395	14.142306	-159.950928
4364	14.197629	-160.632851	14.140945	-159.982178
4365	14.196653	-160.659855	14.139811	-160.008301
4366	14.195841	-160.682325	14.138866	-160.030029
4367	14.195166	-160.701020	14.138078	-160.048096
4368	14.194604	-160.716575	14.137422	-160.063477
4369	14.194137	-160.729518	14.136875	-160.075928

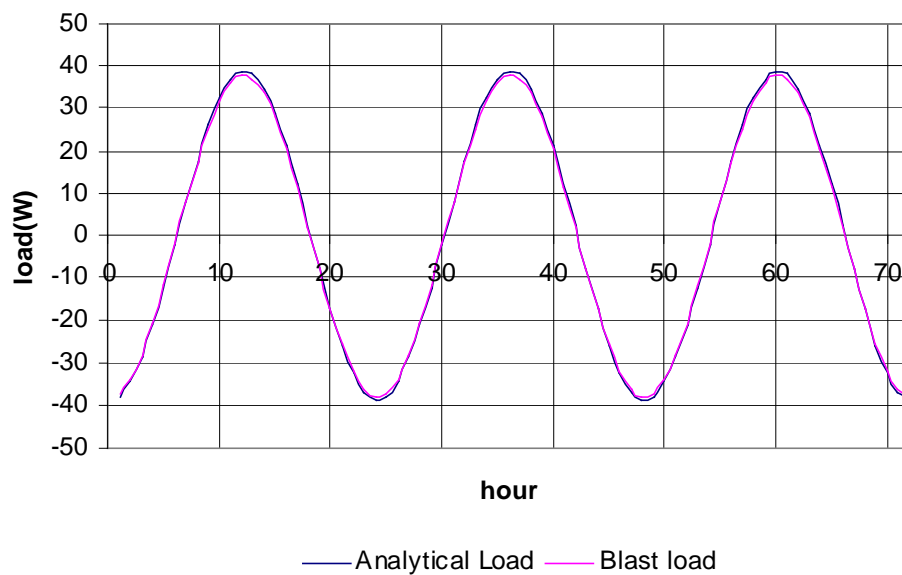
Comment: The BLAST result shows response to step changes in external dry bulb temperature in approximately the same way as the analytical solution. In the first step, the maximum difference in internal surface temperature is 0.3 °C, with 40 hours out of 51 hours have an difference within 0.1 °C. The difference in zone load is very large in the first ten hours, but it decreases to within 3% after that. In the second step, the maximum difference in internal surface temperature is -0.6 °C, with 37 hours out of 50 hours have an difference within ± 0.1 °C. Again the difference in zone load is very large in the first ten hours, but it decreases to within 5% after that.

BLAST comparison results: Test Tc3

Test parameters (Layer 1 is inside, layer 3 is outside)

Test Parameter	Value			Units
	Layer 1	Layer 2	Layer 3	
Thermal conductivity	0.14	0.1	0.2	W/m.K
Density	700	50	500	Kg/m ³
Specific heat capacity	500	200	800	J/kg.K
Thickness	0.1	0.05	0.1	m
Mean temperature (T_l)	20.0			°C
Temperature amplitude (T_{amp})	15.0			°C
Period of fluctuation	24			Hours
External convection coefficient	12.49			W/m ² .K
Internal convection coefficient	3.076			W/m ² .K

Plotted results



Tabulated results

Hour	Analytical Load (W)	Blast load (W)
1	-37.983979	-37.284180
2	-34.653140	-33.952881
3	-28.960746	-28.324463
4	-21.294726	-20.772461
5	-12.177505	-11.812744
6	-2.230408	-2.071533
7	7.868688	7.788818
8	17.431547	16.988037
9	25.806474	25.167236
10	32.422732	31.583252
11	36.829435	35.837891
12	38.726273	37.637451
13	37.983979	36.847168
14	34.653140	33.528320
15	28.960746	27.910156
16	21.294726	20.370117
17	12.177505	11.412109
18	2.230408	1.659424
19	-7.868688	-8.190186
20	-17.431547	-17.415283
21	-25.806474	-25.594238
22	-32.422732	-32.020996
23	-36.829435	-36.285400
24	-38.726273	-38.081787
25	-37.983979	-37.284180
26	-34.653140	-33.952881
27	-28.960746	-28.324463
28	-21.294726	-20.772461
29	-12.177505	-11.812744
30	-2.230408	-2.071533
31	7.868688	7.788818
32	17.431547	16.988037
33	25.806474	25.167236
34	32.422732	31.583252
35	36.829435	35.837891
36	38.726273	37.637451
37	37.983979	36.847168
38	34.653140	33.528320
39	28.960746	27.910156
40	21.294726	20.370117
41	12.177505	11.412109
42	2.230408	1.659424
43	-7.868688	-8.190186
44	-17.431547	-17.415283
45	-25.806474	-25.594238
46	-32.422732	-32.020996

47	-36.829435	-36.285400
48	-38.726273	-38.081787
49	-37.983979	-37.284180
50	-34.653140	-33.952881
51	-28.960746	-28.324463
52	-21.294726	-20.772461
53	-12.177505	-11.812744
54	-2.230408	-2.071533
55	7.868688	7.788818
56	17.431547	16.988037
57	25.806474	25.167236
58	32.422732	31.583252
59	36.829435	35.837891
60	38.726273	37.637451
61	37.983979	36.847168
62	34.653140	33.528320
63	28.960746	27.910156
64	21.294726	20.370117
65	12.177505	11.412109
66	2.230408	1.659424
67	-7.868688	-8.190186
68	-17.431547	-17.415283
69	-25.806474	-25.594238
70	-32.422732	-32.020996
71	-36.829435	-36.285400
72	-38.726273	-38.081787

Comment: The BLAST results show response to periodic changes in external dry bulb temperature in approximately the same way as analytical solution does in the TC3 case. The maximum difference in the zone load is 1.14W, which is 2.94% of the analytical peak load.

BLAST comparison results: Test ExtSolRad

Test parameters (Layer 1 is outside, layer 3 is inside)

Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	180	Degree
Solar absorption of the surface ' α '	0.9	-
Number of layers: ' N '	3	-
Thermal conductivity: layer 1 ' K_1 '	0.15	W/mK
Thickness: layer 1 ' L_1 '	0.1	m
Thermal conductivity: layer 2 ' K_2 '	0.05	W/mK
Thickness: layer 2 ' L_2 '	0.1	m
Thermal conductivity: layer 3 ' K_3 '	0.15	W/mK
Thickness: layer 3 ' L_3 '	0.1	m
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	12.49	W/m ² K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	3.076	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

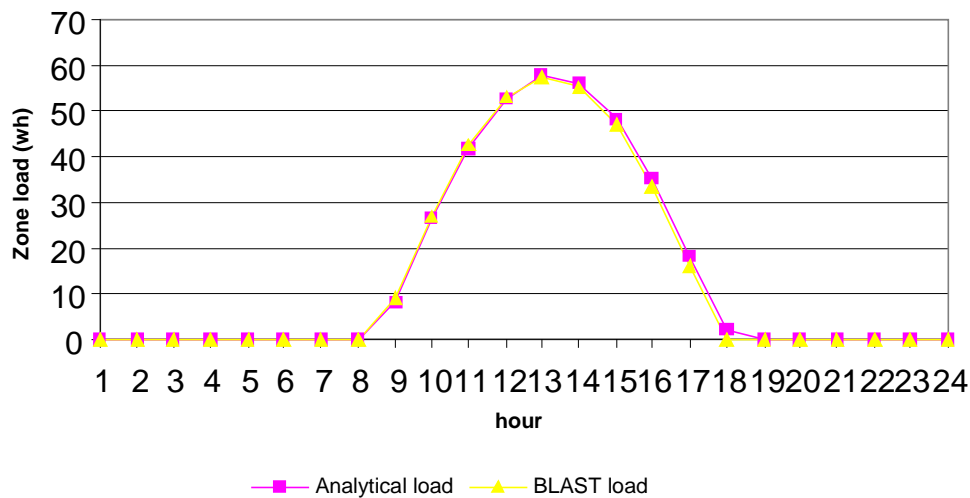
Tabulated results

hour	Analytical load (wh)	BLAST load 8/21_H000I0 (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000

9	8.4124	9.1853
10	26.3177	27.0967
11	41.8726	42.4409
12	52.8181	52.9443
13	57.8145	57.3872
14	56.2308	55.4155
15	48.4330	47.0940
16	35.1659	33.4402
17	18.0858	16.0771
18	2.3190	0.0000
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Plotted results

Zone load comparison of ExtSolRad: South facing wall



Comment: BLAST result shows it responds to exterior solar beam radiation in approximately the same way as analytical solution does in the condition of ExtSolRad case. The maximum difference between the loads calculated by BLAST and the analytical solution is 2.32 Wh, which is about 4% of the peak analytical load. This occurs when the analytical load is zero but BLAST predicts a load of 2.23W.

Note that it was necessary to modify the WIFE weather data processing program in order to obtain zero diffuse radiation in the final weather data file. Diffuse irradiation is set to zero in the test weather files so that loads are only induced by direct irradiation. It is acknowledged that this would be a problem for most users and may be an issue with other test programs.

BLAST comparison results: Test SolRadGlazing

Test parameters

Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	180	Degree
Thickness of the surface ' L '	0.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	8.23	W/m ² K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	3.076	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

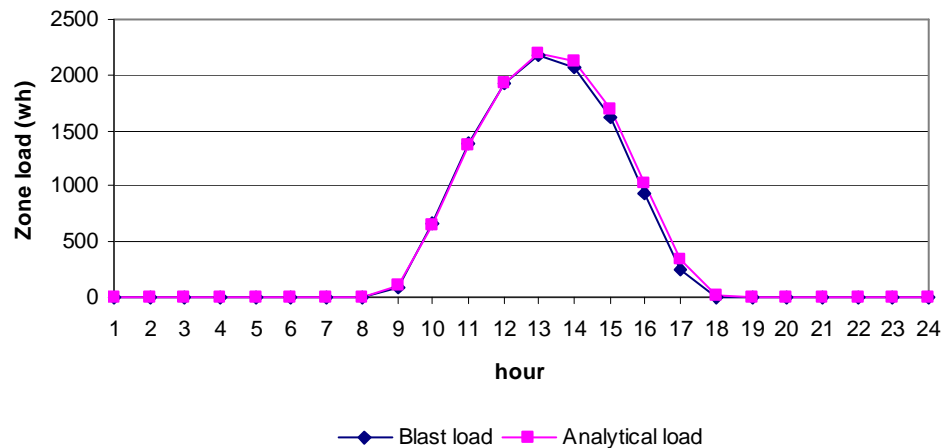
Tabulated results

hour	Blast load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	91.6245	100.0184
10	663.0557	641.0149
11	1384.1584	1360.4352
12	1932.5920	1931.6661
13	2170.9072	2200.8446
14	2063.0425	2115.1078
15	1620.8608	1699.2282
16	939.4871	1032.6289

17	258.5376	341.5271
18	0.0000	15.0076
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Plotted results

Zone load comparison for SolRadGlazing: South facing window



Comment: BLAST result shows response to exterior solar beam radiation in approximately the same way as the analytical solution. There is one hour that BLAST doesn't show any load while there is a very small load for analytical solution. At this hour the difference between predicted and analytical loads reaches a maximum of 93.14Wh, which is 4.23% of the analytical peak load.

BLAST comparison results: Test SolRadShade

Test parameters

Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	180	Degree
Thickness of the surface ' L '	0.0023	m
Extinction coefficient of the surface ' K '	10.0	m ⁻¹
Refractive index of the surface ' n_g '	1.526	-
Depth of the horizontal fin ' P_h '*	0.6	m
Depth of the vertical fin ' P_v '*	0.6	m
Vertical fin is on which side of the window*	Right/Left	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	8.23	W/m ² K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	3.076	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

* Combinations of these parameter values are used as noted in the results

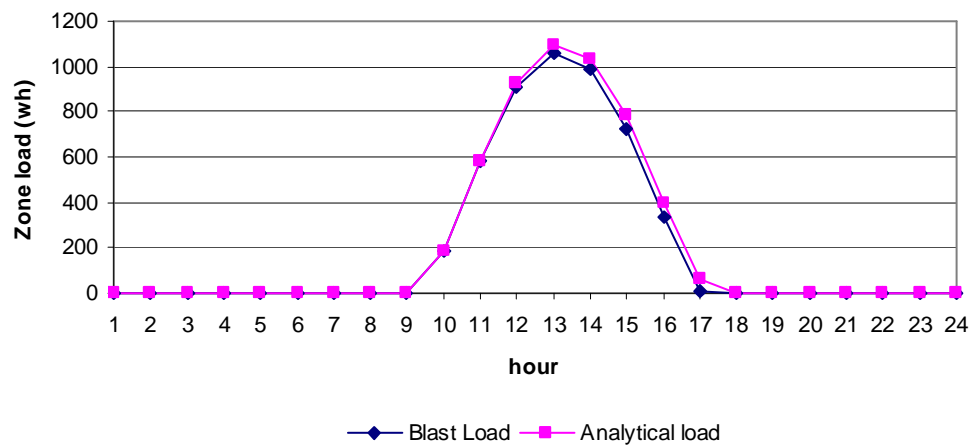
Tabulated and plotted results

1. South facing window with horizontal shade

hour	Blast Load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	0.0000	1.5560
10	188.2690	188.1192

11	583.7397	580.5106
12	911.2129	922.8312
13	1057.8550	1090.4571
14	990.6838	1036.3453
15	722.2458	782.5699
16	331.9292	396.8906
17	6.7747	60.1683
18	0.0000	0.0000
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Zone load comparison for SolRadShade_Hor

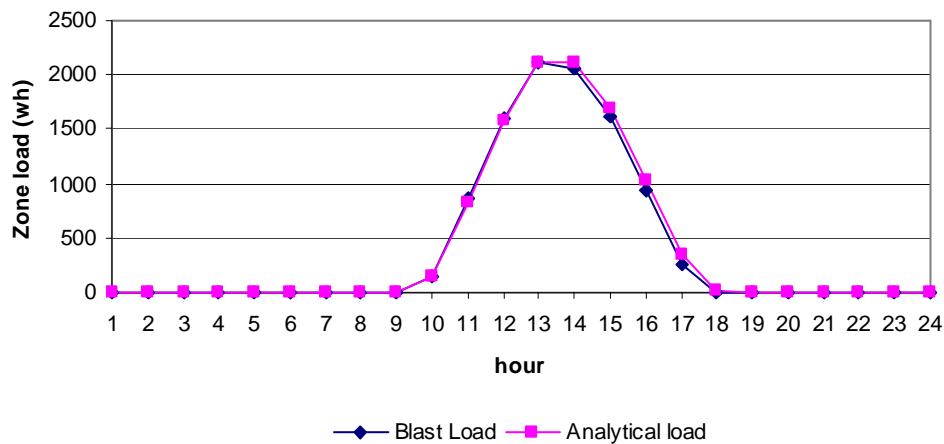


2. Vertical fin on right side of south facing window

hour	Blast Load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	0.0000	0.0000
10	148.5366	150.7421
11	857.4033	827.4395
12	1590.5742	1572.2631
13	2112.8013	2112.6204

14	2060.3738	2115.1078
15	1618.2712	1699.2282
16	937.4695	1032.6289
17	257.4417	341.5271
18	0.0000	15.0076
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Zone load comparison for SolRadShade_VerR

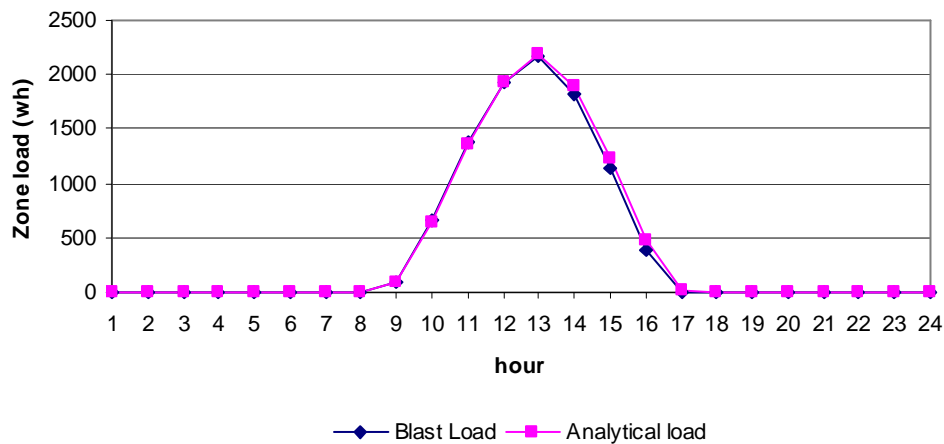


3. Vertical fin on left side of south facing window

Hour	Blast Load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	91.6782	100.0184
10	662.9302	641.0149
11	1383.1152	1360.4352
12	1930.5908	1931.6661
13	2168.3093	2192.9176
14	1817.4172	1895.3568
15	1141.4661	1238.3097
16	385.1753	480.3507

17	0.0000	18.2679
18	0.0000	0.0000
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Zone load comparison for SolRadShade_VerL

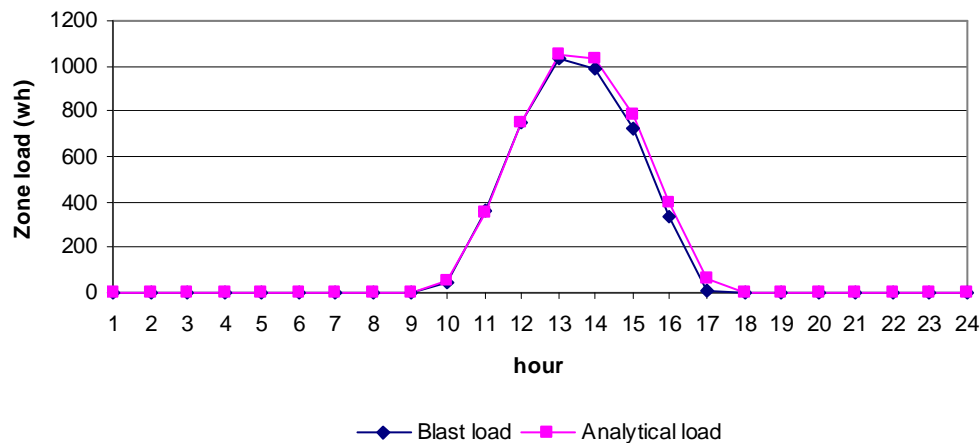


4. South facing window with horizontal shade and right side vertical fin

Hour	Blast load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	0.0000	0.0000
10	42.4111	50.6418
11	362.3796	355.4659
12	751.1267	751.9689
13	1030.9146	1046.8828
14	990.7393	1036.3453
15	722.3645	782.5699
16	332.3816	396.8906
17	7.6919	60.1683
18	0.0000	0.0000
19	0.0000	0.0000

20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Zone load comparison for SolRadShade_HorVerR

**Comment:**

The BLAST results show approximately the same shading effects for exterior solar beam radiation as that of the analytical solution.. The errors follow the same trend as in the SolRadGlazing case. The differences between predicted loads and the analytical solution are as follows:

- In the test case of south facing window with horizontal shade, the maximum zone load difference (64.96 Wh) is 5.96% of the analytical peak load.
- In the test case of south facing window with vertical fin on right side, the maximum zone load difference (95.16 Wh) is 4.50% of the analytical peak load.
- In the test case of south facing window with vertical fin on left side, the maximum zone load difference (96.84 Wh) is 4.42% of the analytical peak load.
- In the test case of south facing window with horizontal shade and right side vertical fin, the maximum zone load difference (64.51Wh) is 6.16% of the analytical peak load.

It was found out that there appears to be an error in the BLAST manual regarding the syntax used to specify a vertical wing. Under “Wings and Overhangs” in BLAST manual, the following input syntax is given for a vertical wing

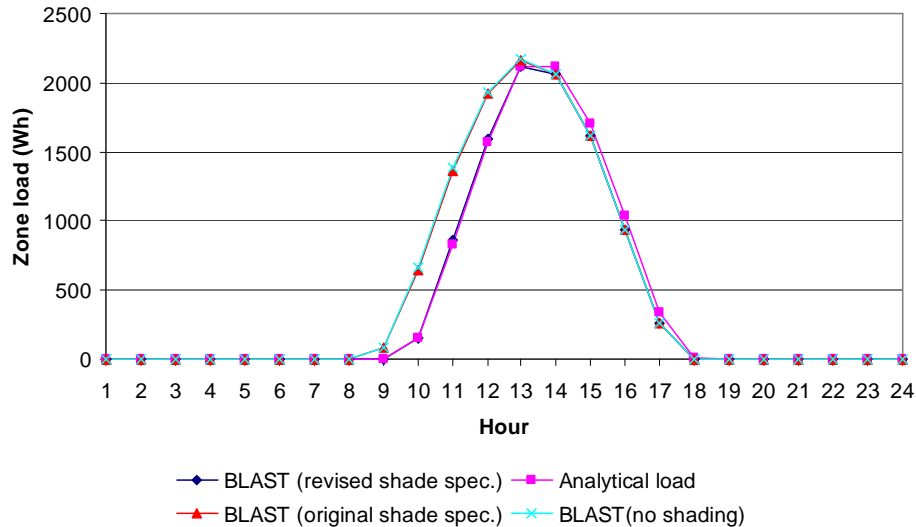
“WITH WINGS(usn1, usn2)”

Where ‘usn1’ is supposed to be the depth and ‘usn2’ the height of the vertical wing. The correct shading effect was only found with the arguments (or there definitions) reversed:

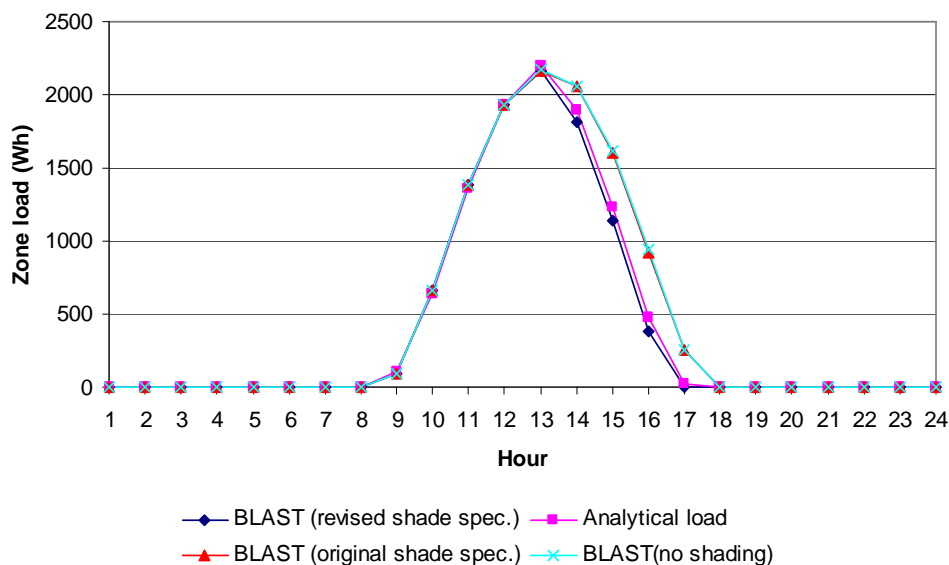
“WITH WINGS(usn2, usn1)”

The following two figures show the result of using the syntax as it appears in the BLAST Manual. In this case only a very small shading effect is observed. This might be expected with a wing that is only 0.6m high at the bottom of the 3x3m window. Using the alternative syntax - Reversing the argument values - can be seen to agree well with the analytical solution.

**Comparison of zone loads with alternative shade specifications:
Right side wing on south facing window**



Comparison of zone loads with alternative shade specifications: Left side wing on south facing window



BLAST comparison results: Test WinReveal

Test parameters

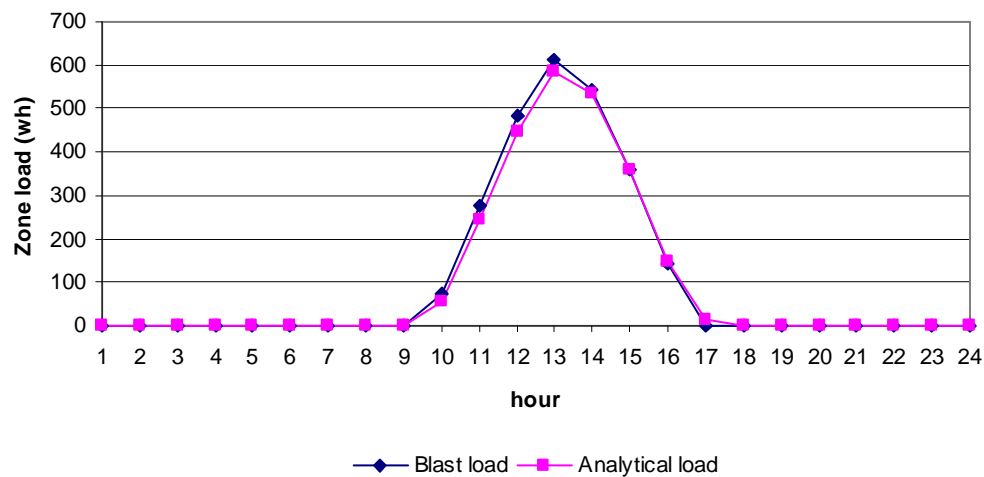
Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	180	Degree
Thickness of the window surface	0.0023	m
Depth of the window reveal ' R '	0.3	m
Length of the window ' B '	2.0	m
Height of the window ' H '	2.0	m
Extinction coefficient of the window surface ' K '	10.0	m ⁻¹
Refractive index of the window surface ' n_g '	1.526	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	8.23	W/m ² K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	3.076	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

Tabulated and plotted results

Hour	Blast load (wh)	Analytical load (wh)
1	0.0000	0.0000
2	0.0000	0.0000
3	0.0000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000
7	0.0000	0.0000
8	0.0000	0.0000
9	0.0000	0.0000
10	73.8672	56.8706
11	276.6445	242.9290
12	482.3848	448.2255

13	613.9238	586.6180
14	542.7305	533.8867
15	357.9856	357.2700
16	141.8394	148.1089
17	0.0000	13.0173
18	0.0000	0.0000
19	0.0000	0.0000
20	0.0000	0.0000
21	0.0000	0.0000
22	0.0000	0.0000
23	0.0000	0.0000
24	0.0000	0.0000

Zone load comparison of WinReveal--South facing window



Comment: BLAST result shows approximately the same shading effects for exterior solar beam irradiation as that of the analytical solution under the condition of WinReveal case. Error analysis shows that the maximum zone load difference (34.16 Wh) is 5.82% of the analytical peak load.

BLAST comparison results: Test IntSolarDist

Test parameters

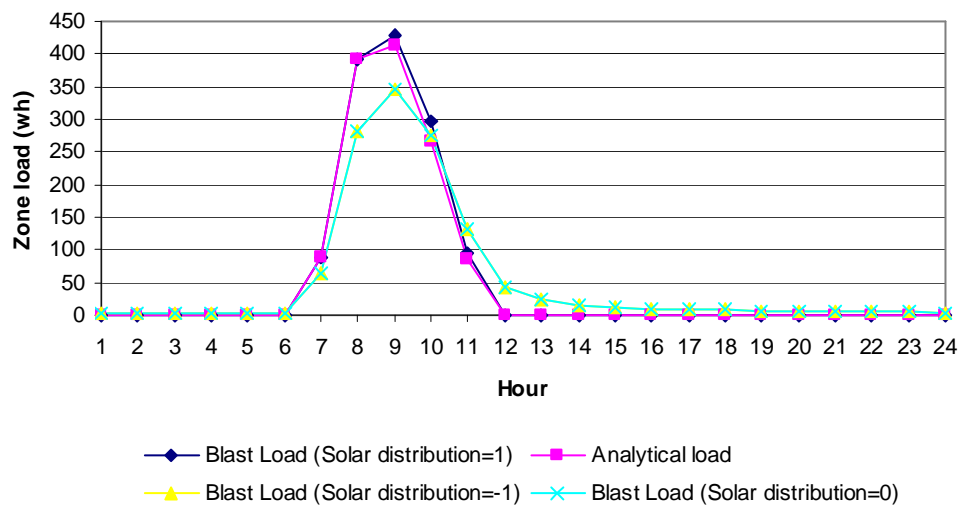
Test parameters	Value	Units
Location	Atlanta	-
Test date	08/21/1999	-
Surface tilt angle ' ε '	90	Degree
Surface azimuth ' ψ '	90	Degree
Thickness of the window surface	.0023	m
Depth of the horizontal fin ' P_h '	0.5	m
Depth of the vertical fin ' P_v '	0.5	m
Extinction coefficient of the window surface ' K '	10.0	m ⁻¹
Refractive index of the window surface ' n_g '	1.526	-
Solar absorption of the internal surfaces	1.0	-
External air temperature ' $T_{a,o}$ '	20	C
Internal air temperature ' $T_{a,i}$ '	20	C
Outside correlation coefficient 'A'	8.23	W/m ² K
Outside correlation coefficient 'C'	0.0	-
Outside correlation exponent 'n'	0.333	-
Inside correlation coefficient 'A'	3.076	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

Tabulated and plotted results

Hour	Blast Load (Solar distribution=1) (wh)	Blast Load (Solar distribution=0) (wh)	Blast Load (Solar distribution=-1) (wh)	Analytical load (wh)
1	0.0000	4.2429	4.2429	0.0000
2	0.0000	4.0212	4.0212	0.0000
3	0.0000	3.8259	3.8259	0.0000
4	0.0000	3.6506	3.6506	0.0000
5	0.0000	3.4917	3.4917	0.0000
6	0.0000	3.3452	3.3452	0.0000
7	88.6055	64.3809	64.3809	88.6987

8	391.7932	282.9180	282.9180	393.1928
9	427.3174	344.6870	344.6870	413.7012
10	298.1963	274.0200	274.0200	266.6587
11	94.8333	132.6670	132.6670	85.8311
12	0.0000	43.5574	43.5574	0.0000
13	0.0000	24.1409	24.1409	0.0000
14	0.0000	16.7078	16.7078	0.0000
15	0.0000	12.8271	12.8271	0.0000
16	0.0000	10.4370	10.4370	0.0000
17	0.0000	8.8181	8.8181	0.0000
18	0.0000	7.6536	7.6536	0.0000
19	0.0000	6.7805	6.7805	0.0000
20	0.0000	6.1067	6.1067	0.0000
21	0.0000	5.5735	5.5735	0.0000
22	0.0000	5.1426	5.1426	0.0000
23	0.0000	4.7876	4.7876	0.0000
24	0.0000	4.4895	4.4895	0.0000

Zone load comparison of IntSolarDist--East facing window



Comment: The IntSolarDist test is designed to be able to detect whether any solar energy is being redistributed within the zone. BLAST is able to redistribute the transmitted solar irradiation differently according to how a ‘solar distribution flag’ is set. According to the BLAST manual the meaning of the flags is as follows:

- Solar Distribution = -1: Detached shading devices and zone walls are not taken into account. All transmitted solar is assumed incident on the floor.
- Solar Distribution = 0: All transmitted solar is assumed incident on the floor.
- Solar Distribution = 1: Transmitted solar is distributed according to the position of the sun patch.

A value for the Solar Distribution flag of zero is the default in BLAST.

The resulting hourly zone loads (see plot above) with the different solar distribution options verify the operation of the solar distribution flag noted above. The results with flag values of 0 and -1, that assume transmitted solar incident on the floor (which is heavyweight), show a reduction in the peak load and storage of some of the energy until later hours. Only a flag value of 1, where the distribution is explicitly calculated and the energy is distributed to the massless rear wall of the zone, results in loads matching the analytical solution.

The differences between the predicted and analytical loads are as follows:

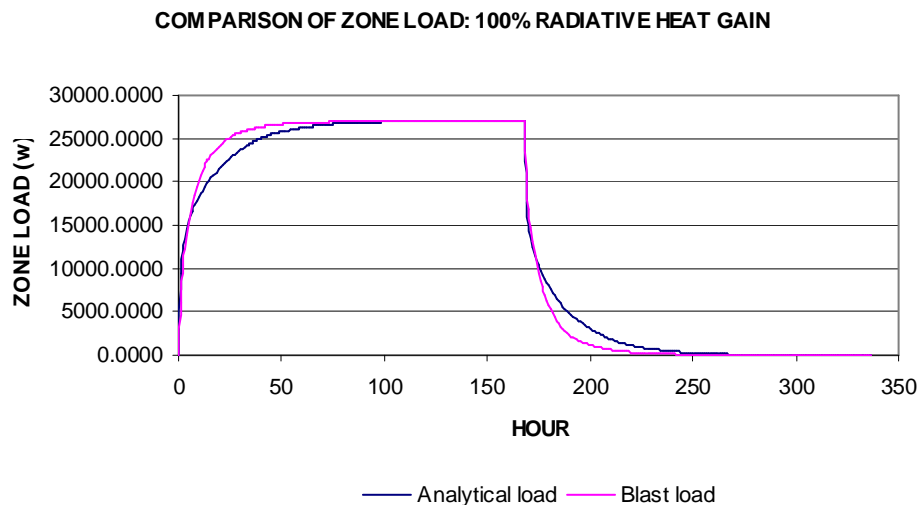
- When solar distribution was specified as 1, the maximum zone load difference (31.54 Wh) is 7.62% of the analytical peak load.
- When solar distribution was specified as 0 and -1, the maximum zone load difference (110.27 Wh) becomes 26.66% of the analytical peak load.

BLAST comparison results: Test IntHeatGain

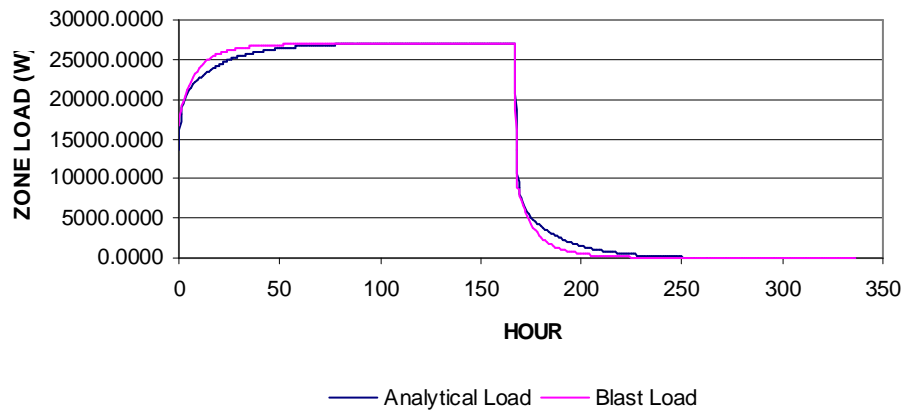
Test parameters

Test parameters	Value	Units
Thermal conductivity	0.14	W/mK
Thickness	0.1	m
Density	500	Kg/m ³
Specific heat capacity	2500	J/kgK
Internal air temperature	20.0	C
Internal convection coefficient	3.076	W/m ² K
Step size of the internal heat gain	27000	W
Radiative fraction of the internal heat gain	0.5 and 1.0	-

Plotted results: 100% & 50% radiative heat gain



**COMPARISON OF ZONE LOAD: 50% RADIATIVE HEAT GAIN, 50%
CONVECTIVE HEAT GAIN**



Note: Before hour zero, there is no internal heat gain

Tabulated results: 100% & 50% radiative heat gain

Time	100% RADIATIVE		50% RADIATIVE	
	Analytical load (wh)	Blast load (wh)	Analytical load (wh)	Blast load (wh)
0	0.0000	0.0000	13500.0000	16514.3945
1	9522.8852	6028.6504	18260.5181	18666.0156
2	11949.0243	10331.6172	19473.5877	19628.3574
3	13447.9799	12256.0283	20223.0655	20419.0098
4	14525.9269	13837.5986	20762.0390	21104.4785
5	15368.5200	15208.2969	21183.3355	21708.1035
6	16067.4556	16415.3164	21532.8033	22242.3496
7	16673.5904	17483.5586	21835.8707	22716.1582
8	17216.3958	18430.9023	22107.2734	23136.8926
9	17713.3261	19272.0801	22355.7385	23510.9023
10	18174.9199	20019.6875	22586.5355	23843.7305
11	18607.7008	20684.9980	22802.9259	24140.2305
12	19015.8478	21277.6484	23006.9994	24404.6035
13	19402.1631	21806.1484	23200.1571	24640.7637
14	19768.6333	22278.0547	23383.3922	24851.8711
15	20116.7547	22699.9004	23557.4529	25040.8418
16	20447.7234	23077.4883	23722.9372	25210.2227
17	20762.5453	23415.9043	23880.3482	25362.2500
18	21062.1017	23719.6289	24030.1264	25498.8965
19	21347.1869	23992.6035	24172.6689	25621.8965
20	21618.5312	24238.2949	24308.3411	25732.7754
21	21876.8153	24459.7617	24437.4832	25832.8809
22	22122.6782	24659.6953	24560.4146	25923.4004
23	22356.7233	24840.4727	24677.4372	26005.3789

24	22579.5224	25004.1895	24788.8367	26079.7051
25	22791.6177	25152.6777	24894.8844	26147.2930
26	22993.5247	25287.6094	24995.8379	26208.7871
27	23185.7335	25410.3984	25091.9423	26264.8398
28	23368.7104	25522.3301	25183.4307	26316.0176
29	23542.8989	25624.5313	25270.5250	26362.8223
30	23708.7213	25718.0059	25353.4362	26405.6992
31	23866.5796	25803.6387	25432.3653	26445.0391
32	24016.8562	25882.2188	25507.5036	26481.1934
33	24159.9153	25954.4453	25579.0332	26514.4727
34	24296.1036	26020.9375	25647.1273	26545.1543
35	24425.7511	26082.2480	25711.9511	26573.4824
36	24549.1719	26138.8691	25773.6615	26599.6582
37	24666.6651	26191.2305	25832.4080	26623.9297
38	24778.5153	26239.7461	25888.3332	26646.4297
39	24884.9936	26284.7480	25941.5723	26667.3242
40	24986.3580	26326.5527	25992.2545	26686.7539
41	25082.8542	26365.4395	26040.5026	26704.8438
42	25174.7158	26401.6602	26086.4334	26721.7090
43	25262.1655	26435.4395	26130.1583	26737.4512
44	25345.4152	26466.9824	26171.7831	26752.1602
45	25424.6666	26496.4688	26211.4088	26765.9180
46	25500.1118	26524.0645	26249.1314	26778.8027
47	25571.9335	26549.9180	26285.0423	26790.8809
48	25640.3057	26574.1641	26319.2284	26802.1992
49	25705.3942	26596.9180	26351.7726	26812.8457
50	25767.3567	26618.3066	26382.7539	26822.8477
51	25826.3432	26638.4160	26412.2471	26832.2559
52	25882.4968	26657.3438	26440.3239	26841.1094
53	25935.9534	26675.1699	26467.0522	26849.4492
54	25986.8426	26691.9707	26492.4968	26857.3125
55	26035.2877	26707.8203	26516.7194	26864.7266
56	26081.4061	26722.7773	26539.7786	26871.7246
57	26125.3096	26736.9023	26561.7303	26878.3340
58	26167.1045	26750.2500	26582.6277	26884.5762
59	26206.8920	26762.8691	26602.5215	26890.4766
60	26244.7687	26774.8047	26621.4599	26896.0488
61	26280.8262	26786.0977	26639.4886	26901.3320
62	26315.1520	26796.7910	26656.6515	26906.3281
63	26347.8291	26806.9180	26672.9901	26911.0586
64	26378.9369	26816.5137	26688.5440	26915.5391
65	26408.5506	26825.6094	26703.3508	26919.7813
66	26436.7421	26834.2305	26717.4466	26923.8047
67	26463.5796	26842.4082	26730.8653	26927.6172
68	26489.1281	26850.1660	26743.6396	26931.2305
69	26513.4496	26857.5273	26755.8003	26934.6582
70	26536.6030	26864.5137	26767.3770	26937.9102
71	26558.6444	26871.1465	26778.3977	26940.9961

72	26579.6272	26877.4434	26788.8891	26943.9199
73	26599.6022	26883.4238	26798.8766	26946.7012
74	26618.6179	26889.1055	26808.3845	26949.3398
75	26636.7203	26894.5039	26817.4357	26951.8438
76	26653.9533	26899.6348	26826.0522	26954.2227
77	26670.3586	26904.5098	26834.2548	26956.4805
78	26685.9760	26909.1445	26842.0635	26958.6250
79	26700.8434	26913.5488	26849.4972	26960.6641
80	26714.9967	26917.7363	26856.5738	26962.5996
81	26728.4702	26921.7168	26863.3106	26964.4395
82	26741.2966	26925.5039	26869.7238	26966.1875
83	26753.5071	26929.1055	26875.8290	26967.8477
84	26765.1310	26932.5273	26881.6410	26969.4258
85	26776.1968	26935.7852	26887.1739	26970.9258
86	26786.7310	26938.8828	26892.4410	26972.3516
87	26796.7593	26941.8301	26897.4552	26973.7070
88	26806.3060	26944.6348	26902.2285	26974.9961
89	26815.3942	26947.3008	26906.7726	26976.2207
90	26824.0458	26949.8398	26911.0984	26977.3848
91	26832.2820	26952.2539	26915.2165	26978.4922
92	26840.1226	26954.5527	26919.1368	26979.5449
93	26847.5866	26956.7383	26922.8688	26980.5449
94	26854.6922	26958.8164	26926.4216	26981.4961
95	26861.4565	26960.7969	26929.8038	26982.4004
96	26867.8959	26962.6816	26933.0235	26983.2598
97	26874.0261	26964.4746	26936.0885	26984.0781
98	26879.8618	26966.1816	26939.0064	26984.8555
99	26885.4172	26967.8047	26941.7841	26985.5957
100	26890.7059	26969.3496	26944.4285	26986.2988
101	26895.7405	26970.8203	26946.9458	26986.9668
102	26900.5333	26972.2207	26949.3422	26987.6035
103	26905.0960	26973.5527	26951.6235	26988.2090
104	26909.4395	26974.8223	26953.7953	26988.7852
105	26913.5744	26976.0293	26955.8627	26989.3320
106	26917.5107	26977.1797	26957.8309	26989.8516
107	26921.2580	26978.2734	26959.7045	26990.3477
108	26924.8253	26979.3145	26961.4881	26990.8184
109	26928.2212	26980.3066	26963.1861	26991.2656
110	26931.4541	26981.2520	26964.8026	26991.6914
111	26934.5317	26982.1504	26966.3414	26992.0957
112	26937.4615	26983.0059	26967.8062	26992.4824
113	26940.2505	26983.8203	26969.2008	26992.8477
114	26942.9057	26984.5957	26970.5283	26993.1973
115	26945.4333	26985.3340	26971.7921	26993.5293
116	26947.8395	26986.0352	26972.9952	26993.8438
117	26950.1301	26986.7051	26974.1406	26994.1426
118	26952.3107	26987.3398	26975.2309	26994.4297
119	26954.3866	26987.9473	26976.2688	26994.6992

120	26956.3628	26988.5234	26977.2569	26994.9570
121	26958.2441	26989.0723	26978.1976	26995.2031
122	26960.0350	26989.5957	26979.0930	26995.4355
123	26961.7399	26990.0938	26979.9455	26995.6582
124	26963.3630	26990.5684	26980.7570	26995.8691
125	26964.9081	26991.0195	26981.5295	26996.0703
126	26966.3789	26991.4473	26982.2650	26996.2617
127	26967.7792	26991.8574	26982.9651	26996.4434
128	26969.1121	26992.2480	26983.6316	26996.6152
129	26970.3811	26992.6172	26984.2661	26996.7793
130	26971.5891	26992.9707	26984.8701	26996.9375
131	26972.7391	26993.3066	26985.4451	26997.0840
132	26973.8339	26993.6250	26985.9925	26997.2266
133	26974.8761	26993.9297	26986.5136	26997.3594
134	26975.8682	26994.2188	26987.0096	26997.4883
135	26976.8127	26994.4961	26987.4819	26997.6094
136	26977.7118	26994.7578	26987.9314	26997.7266
137	26978.5678	26995.0098	26988.3594	26997.8359
138	26979.3826	26995.2461	26988.7668	26997.9414
139	26980.1583	26995.4727	26989.1547	26998.0410
140	26980.8967	26995.6875	26989.5239	26998.1348
141	26981.5997	26995.8926	26989.8754	26998.2266
142	26982.2689	26996.0879	26990.2100	26998.3125
143	26982.9060	26996.2734	26990.5285	26998.3926
144	26983.5125	26996.4512	26990.8317	26998.4707
145	26984.0898	26996.6191	26991.1204	26998.5449
146	26984.6394	26996.7813	26991.3952	26998.6152
147	26985.1626	26996.9336	26991.6568	26998.6816
148	26985.6607	26997.0801	26991.9059	26998.7461
149	26986.1349	26997.2188	26992.1430	26998.8047
150	26986.5863	26997.3516	26992.3687	26998.8633
151	26987.0160	26997.4746	26992.5835	26998.9180
152	26987.4251	26997.5957	26992.7881	26998.9688
153	26987.8145	26997.7090	26992.9828	26999.0195
154	26988.1853	26997.8184	26993.1681	26999.0664
155	26988.5382	26997.9219	26993.3446	26999.1094
156	26988.8742	26998.0215	26993.5126	26999.1523
157	26989.1940	26998.1152	26993.6725	26999.1934
158	26989.4985	26998.2031	26993.8248	26999.2324
159	26989.7883	26998.2891	26993.9697	26999.2695
160	26990.0643	26998.3691	26994.1076	26999.3027
161	26990.3269	26998.4453	26994.2390	26999.3359
162	26990.5770	26998.5195	26994.3640	26999.3672
163	26990.8151	26998.5898	26994.4830	26999.3984
164	26991.0417	26998.6543	26994.5964	26999.4258
165	26991.2574	26998.7168	26994.7042	26999.4531
166	26991.4628	26998.7793	26994.8069	26999.4785
167	26991.6583	26998.8359	26994.9047	26999.5039

168	26991.8444	26998.8887	13494.9977	9962.2295
169	17469.1364	19926.4180	8734.5682	8437.7168
170	15043.1660	16878.4355	7521.5830	7434.6416
171	13544.3709	14873.1992	6772.1855	6628.3545
172	12466.5768	13261.4912	6233.2884	5934.0088
173	11624.1292	11873.6182	5812.0646	5323.7188
174	10925.3322	10653.8008	5462.6661	4783.9766
175	10319.3292	9575.0186	5159.6646	4305.5161
176	9776.6494	8618.7393	4888.3247	3880.8242
177	9279.8386	7769.9312	4639.9193	3503.4524
178	8818.3585	7015.7002	4409.1793	3167.7769
179	8385.6860	6344.7998	4192.8430	2868.8704
180	7977.6421	5747.3770	3988.8210	2602.4080
181	7591.4249	5214.7852	3795.7124	2364.5923
182	7225.0482	4739.4326	3612.5241	2152.0867
183	6877.0157	4314.6494	3438.5078	1961.9587
184	6546.1317	3934.5723	3273.0659	1791.6301
185	6231.3904	3594.0498	3115.6952	1638.8333
186	5931.9108	3288.5488	2965.9554	1501.5728
187	5646.8987	3014.0825	2823.4493	1378.0918
188	5375.6238	2767.1401	2687.8119	1266.8433
189	5117.4060	2544.6301	2558.7030	1166.4631
190	4871.6061	2343.8301	2435.8031	1075.7510
191	4637.6210	2162.3381	2318.8105	993.6458
192	4414.8791	1998.0366	2207.4395	919.2122
193	4202.8381	1849.0579	2101.4191	851.6228
194	4000.9829	1713.7507	2000.4914	790.1497
195	3808.8233	1590.6575	1904.4117	734.1453
196	3625.8934	1478.4888	1812.9467	683.0396
197	3451.7495	1376.1045	1725.8748	636.3254
198	3285.9696	1282.4939	1642.9848	593.5557
199	3128.1518	1196.7625	1564.0759	554.3325
200	2977.9137	1118.1169	1488.9569	518.3027
201	2834.8913	1045.8521	1417.4456	485.1533
202	2698.7379	979.3425	1349.3689	454.6047
203	2569.1237	918.0315	1284.5618	426.4089
204	2445.7345	861.4231	1222.8673	400.3452
205	2328.2715	809.0769	1164.1357	376.2156
206	2216.4499	760.5989	1108.2249	353.8445
207	2109.9989	715.6360	1054.9994	333.0732
208	2008.6604	673.8750	1004.3302	313.7617
209	1912.1891	635.0344	956.0945	295.7837
210	1820.3510	598.8613	910.1755	279.0249
211	1732.9237	565.1296	866.4618	263.3838
212	1649.6953	533.6353	824.8477	248.7681
213	1570.4642	504.1958	785.2321	235.0959
214	1495.0384	476.6450	747.5192	222.2917
215	1423.2351	450.8345	711.6176	210.2888

216	1354.8804	426.6294	677.4402	199.0254
217	1289.8086	403.9075	644.9043	188.4458
218	1227.8620	382.5581	613.9310	178.5010
219	1168.8906	362.4807	584.4453	169.1438
220	1112.7515	343.5840	556.3757	160.3328
221	1059.3086	325.7842	529.6543	152.0308
222	1008.4324	309.0056	504.2162	144.2017
223	959.9997	293.1785	479.9998	136.8147
224	913.8931	278.2390	456.9465	129.8398
225	870.0009	264.1292	435.0004	123.2505
226	828.2167	250.7942	414.1084	117.0220
227	788.4394	238.1855	394.2197	111.1318
228	750.5724	226.2573	375.2862	105.5583
229	714.5242	214.9678	357.2621	100.2827
230	680.2072	204.2776	340.1036	95.2866
231	647.5384	194.1506	323.7692	90.5537
232	616.4386	184.5544	308.2193	86.0684
233	586.8325	175.4570	293.4162	81.8157
234	558.6482	166.8298	279.3241	77.7827
235	531.8176	158.6465	265.9088	73.9578
236	506.2757	150.8806	253.1378	70.3279
237	481.9604	143.5105	240.9802	66.8831
238	458.8129	136.5134	229.4065	63.6125
239	436.7772	129.8691	218.3886	60.5068
240	415.7998	123.5581	207.8999	57.5576
241	395.8299	117.5627	197.9149	54.7561
242	376.8191	111.8665	188.4095	52.0940
243	358.7213	106.4529	179.3607	49.5649
244	341.4927	101.3076	170.7464	47.1614
245	325.0916	96.4163	162.5458	44.8765
246	309.4782	91.7659	154.7391	42.7046
247	294.6147	87.3445	147.3073	40.6396
248	280.4650	83.1392	140.2325	38.6763
249	266.9949	79.1396	133.4975	36.8086
250	254.1718	75.3354	127.0859	35.0330
251	241.9645	71.7168	120.9822	33.3438
252	230.3435	68.2739	115.1717	31.7371
253	219.2806	64.9983	109.6403	30.2092
254	208.7491	61.8816	104.3745	28.7546
255	198.7233	58.9158	99.3617	27.3716
256	189.1791	56.0938	94.5895	26.0552
257	180.0933	53.4080	90.0466	24.8032
258	171.4438	50.8513	85.7219	23.6113
259	163.2097	48.4182	81.6049	22.4773
260	155.3711	46.1023	77.6856	21.3979
261	147.9090	43.8977	73.9545	20.3708
262	140.8053	41.7996	70.4026	19.3950
263	134.0427	39.8020	67.0214	18.4648

264	127.6050	37.9001	63.8025	17.5798
265	121.4764	36.0901	60.7382	16.7368
266	115.6422	34.3667	57.8211	15.9346
267	110.0881	32.7263	55.0441	15.1709
268	104.8008	31.1633	52.4004	14.4441
269	99.7675	29.6763	49.8838	13.7522
270	94.9759	28.2603	47.4879	13.0933
271	90.4144	26.9119	45.2072	12.4658
272	86.0720	25.6279	43.0360	11.8691
273	81.9382	24.4055	40.9691	11.3005
274	78.0029	23.2415	39.0014	10.7595
275	74.2566	22.1331	37.1283	10.2444
276	70.6902	21.0774	35.3451	9.7542
277	67.2951	20.0725	33.6476	9.2871
278	64.0631	19.1152	32.0315	8.8425
279	60.9863	18.2041	30.4931	8.4192
280	58.0572	17.3362	29.0286	8.0161
281	55.2689	16.5098	27.6344	7.6321
282	52.6145	15.7227	26.3072	7.2671
283	50.0875	14.9734	25.0438	6.9194
284	47.6819	14.2595	23.8410	6.5881
285	45.3919	13.5801	22.6959	6.2729
286	43.2118	12.9326	21.6059	5.9727
287	41.1364	12.3164	20.5682	5.6868
288	39.1607	11.7295	19.5804	5.4146
289	37.2799	11.1707	18.6400	5.1558
290	35.4895	10.6382	17.7447	4.9092
291	33.7850	10.1313	16.8925	4.6738
292	32.1624	9.6487	16.0812	4.4502
293	30.6177	9.1885	15.3088	4.2373
294	29.1472	8.7510	14.5736	4.0347
295	27.7473	8.3337	13.8737	3.8416
296	26.4147	7.9370	13.2073	3.6580
297	25.1460	7.5591	12.5730	3.4832
298	23.9383	7.1985	11.9692	3.3162
299	22.7886	6.8555	11.3943	3.1577
300	21.6941	6.5291	10.8471	3.0063
301	20.6522	6.2178	10.3261	2.8623
302	19.6603	5.9216	9.8302	2.7256
303	18.7161	5.6394	9.3581	2.5952
304	17.8172	5.3708	8.9086	2.4709
305	16.9615	5.1147	8.4807	2.3525
306	16.1469	4.8711	8.0734	2.2402
307	15.3714	4.6392	7.6857	2.1331
308	14.6331	4.4182	7.3166	2.0310
309	13.9303	4.2075	6.9652	1.9338
310	13.2613	4.0071	6.6306	1.8413
311	12.6244	3.8159	6.3122	1.7529

312	12.0181	3.6340	6.0090	1.6692
313	11.4409	3.4607	5.7204	1.5891
314	10.8914	3.2959	5.4457	1.5127
315	10.3683	3.1389	5.1841	1.4407
316	9.8703	2.9893	4.9352	1.3718
317	9.3963	2.8467	4.6981	1.3059
318	8.9450	2.7112	4.4725	1.2434
319	8.5154	2.5818	4.2577	1.1841
320	8.1064	2.4587	4.0532	1.1272
321	7.7171	2.3416	3.8585	1.0732
322	7.3464	2.2300	3.6732	1.0217
323	6.9936	2.1238	3.4968	0.0000
324	6.6577	2.0225	3.3289	0.0000
325	6.3380	1.9260	3.1690	0.0000
326	6.0336	1.8342	3.0168	0.0000
327	5.7438	1.7468	2.8719	0.0000
328	5.4679	1.6633	2.7340	0.0000
329	5.2053	1.5842	2.6027	0.0000
330	4.9553	1.5085	2.4777	0.0000
331	4.7173	1.4368	2.3587	0.0000
332	4.4908	1.3682	2.2454	0.0000
333	4.2751	1.3027	2.1375	0.0000
334	4.0698	1.2410	2.0349	0.0000
335	3.8743	1.1816	1.9372	0.0000
336	3.6882	1.1252	1.8441	0.0000

Comment: The BLAST results show a relatively slower response to a step change in internal convective and/or radiative heat gain in the Internal Heat Gain case. Most difference occurs in the first 40 hours (with error percentage up to 10-37%) and after 168 hours (with error percentage up to 60-70%, the internal heat gain occurs during the period of 0-168 hours). During the period of 41-168 hours the results matches well (with error percentage within $\pm 5\%$ and most less than 2%). In addition, the percentage error for the case of 50% radiative heat gain is relatively less than that for the case of 100% radiative heat gain, which might be expected.

BLAST comparison results: Test IntRad

Test parameters

Test parameters	Value	Units
Width of the cuboid	3.0	m
Outside air temperature	40	C
Inside air temperature	20	C
External convection coefficient	12.49	W/m ² K
Internal convection coefficient	3.076	W/m ² K

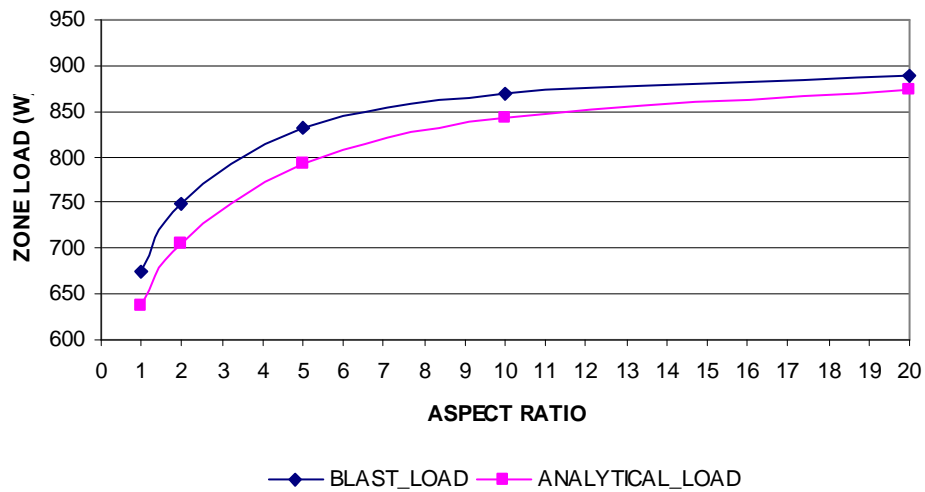
(Note: Emissivities of the surfaces as listed in different cases.)

Tabulated and plotted results:

Case 1: *EMISSIVITY of the EXTERIOR SURFACE =0.9*
 EMISSIVITY of OTHER SURFACES =0.1

ASPECT RATIO	BLAST LOAD (W)	ANALYTICAL LOAD (W)
1	673.899658	636.894897
2	749.285156	705.070923
5	832.828369	793.394104
10	869.833008	842.822205
20	889.138428	874.427429

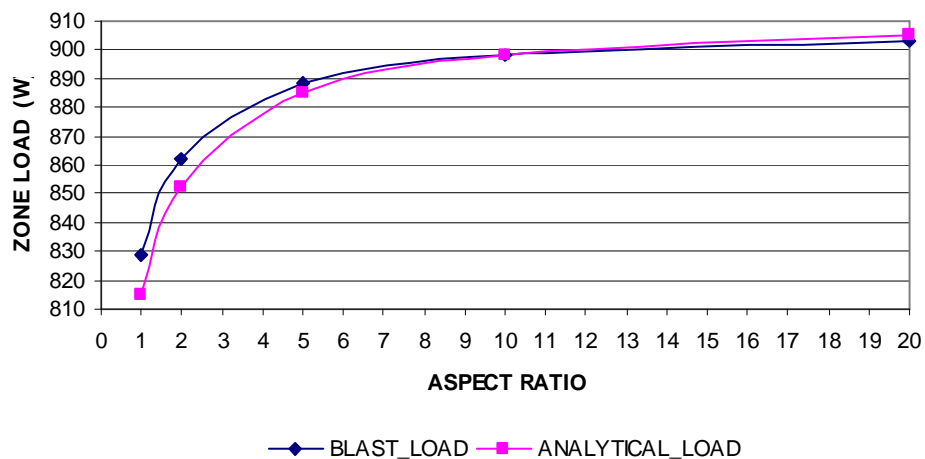
COMPARISON OF ZONE LOAD: CASE 1



Case 2: *EMISSIONITY of ALL THE SURFACES =0.9*

ASPECT RATIO	BLAST LOAD (W)	ANALYTICAL LOAD (W)
1	828.696045	814.712219
2	862.425781	852.588013
5	888.477539	884.776855
10	897.961914	897.870789
20	902.768311	904.924866

COMPARISON OF ZONE LOAD: CASE 2



Case 3:

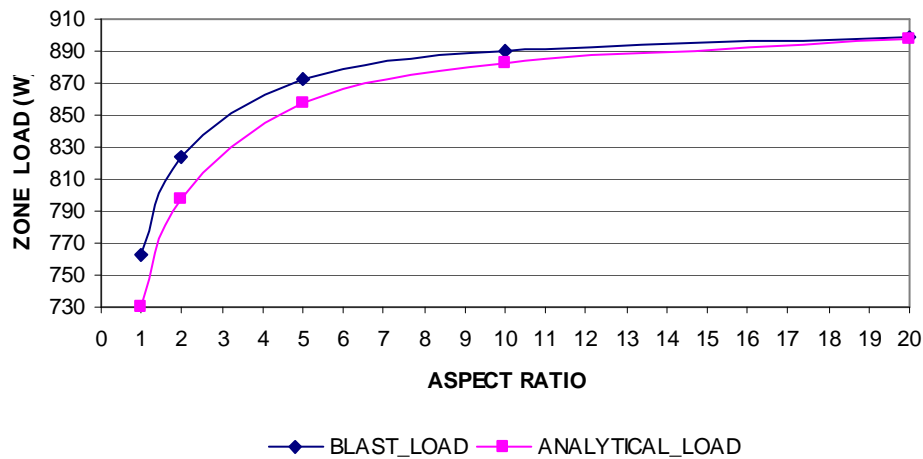
EMISSIONITY of the EXTERIOR SURFACE =0.9

EMISSIONITY of the SURFACE OPPOSITE to the EXTERIOR SURFACE =0.1

EMISSIONITY of OTHER SURFACES = 0.3

ASPECT RATIO	BLAST LOAD (w)	ANALYTICAL LOAD (w)
1	762.168213	730.5515
2	823.822022	796.9018
5	872.172607	857.1487
10	889.806641	883.0053
20	898.714356	897.3897

COMPARISON OF ZONE LOAD: CASE 3



Comment:

1. It was found that in case 2, where the surface emissivities are all the same, the BLAST results match with analytical solution better (with maximum error percentage of -1.72%) than in the other two cases (with maximum error percentage of -6.27% for case 1 and -4.33% for case 3) where various surface emissivities were set.
2. The BLAST results match with analytical solution better under larger aspect ratio than under smaller aspect ratio.

BLAST comparison results: Test Infiltration-1

Test parameters

Test parameters	Value	Units
Outside air temperature	10	C
Inside air temperature	20	C
Outside humidity ratio	0.0046	-
Infiltration rate	0.5	m ³ /s

Tabulated results:

Month	Blast Load in one hour (WH)	Analytical Load in one hour (WH)
1	5810.483871	6304.718103
2	5809.523810	6304.718103
3	5810.483871	6304.718103
4	5809.722222	6304.718103
5	5810.483871	6304.718103
6	5809.722222	6304.718103
7	5810.483871	6304.718103
8	5810.483871	6304.718103
9	5809.722222	6304.718103
10	5810.483871	6304.718103
11	5809.722222	6304.718103
12	5810.483871	6304.718103

Note: Atmospheric pressure = 101325 pa

Comment: A significant difference was found between the zone loads calculated by BLAST and the analytical solution in the Infiltration-1 case. It appears this deviation comes from an error in BLAST. The following line appears in the SIMZGD subroutine (rout33.f) of BLAST:

$$\text{AIRDEN} = (0.003235 * \text{OBP}(\text{STDTIM})) / (\text{ODB}(\text{STDTIM}) + 273.15)$$

(The coefficient 0.003235 corresponds to a value for $R = 309.1 \text{ J/kgK}$. The value of R should be about 287 at room temperature.) The infiltration load also depends on the value of specific heat. This is set at 1004 J/kgK set in BLOCK DATA SMZGBD:

DATA CPAIR / 1004. /

Using the BLAST air density and specific heat written in the source code, hand calculation gives a result of 5809.9 Wh. Our analytical solution is: 6304.7 Wh.

BLAST comparison results: Test GrdCoup

Test parameters

Test parameters	Value	Units
Thermal conductivity of the slab	0.15	W/mK
Thickness of the wall	0.24	m
Inside air temperature	25	C
Outside ground temperature	2	C
Inside correlation coefficient 'A'	0.94889	W/m ² K
Inside correlation coefficient 'C'	0.0	-
Inside correlation exponent 'n'	0.345	-

Tabulated results:

Month	Blast Load (WH)	Analytical Load (WH)
1	77.822581	32.034737
2	77.827381	32.034737
3	77.822581	32.034737
4	77.819444	32.034737
5	77.822581	32.034737
6	77.819444	32.034737
7	77.822581	32.034737
8	77.822581	32.034737
9	77.819444	32.034737
10	77.822581	32.034737
11	77.819444	32.034737
12	77.822581	32.034737

Comment: A significant difference was found between the zone loads calculated by BLAST and the analytical solution in the Ground Coupling case. This is to be expected as the ground coupling model implemented in BLAST is significantly different than that used by the analytical solution.

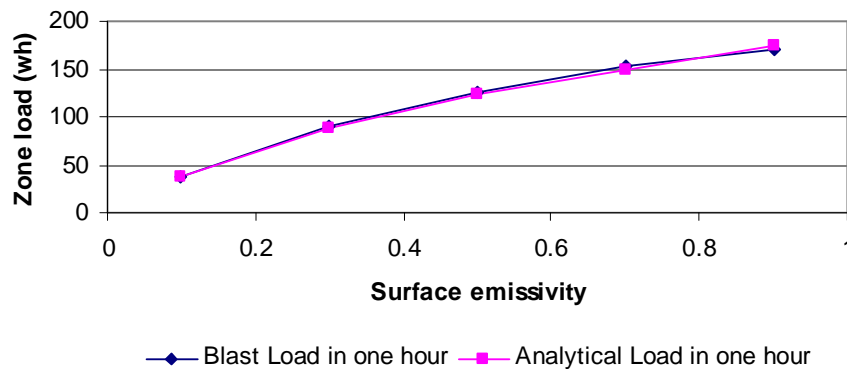
BLAST comparison results: Test ExtLWRad**Test parameters**

Test parameters	Value	Units
Emissivity of the external surface	0.1 - 0.9	-
Thermal conductivity of the external surface	1.0	W/mK
Thickness of the external surface	0.1	m
Sky temperature	4.32	C
External air temperature	20	C
Internal air temperature	20	C
Outside correlation coefficient 'A'	0.0	W/m ² K
Outside correlation coefficient 'C'	0.76	-
Outside correlation exponent 'n'	0.33	-
Inside correlation coefficient 'A'	0.0	W/m ² K
Inside correlation coefficient 'C'	1.51	-
Inside correlation exponent 'n'	0.33	-

Tabulated results:

Emissivity of the external surface	Blast Load in one hour (WH)	Analytical Load in one hour (WH)
0.1	37.5740	37.4920
0.3	90.3416	88.1097
0.5	126.2915	123.1708
0.7	153.2002	149.5015
0.9	170.1989	174.2754

Comparison of analytical and BLAST zone load under different surface emissivity



Comment: The BLAST results are approximately the same as analytical solution in the external long wave radiation case, with a maximum difference of 2.53% for the emissivities tested. It should be noted that the detailed convection algorithm was used while doing the BLAST comparison for this test. The BLAST external long wave radiant interchange algorithm is only used when the detailed convection model is specifically turned on - this is not the default. The detailed convection coefficient correlations are apparently only documented in a draft technical report (*George Walton 1981, Passive Solar Extension of the Building Loads Analysis and System Thermodynamics (BLAST) Program*).

External Testing of the Test Suite

External testing of the test suite has been undertaken by GARD Analytics using the EnergyPlus program. Comments were received in two phases and concerned both the test suite software and the documentation. The original comments are included below with the action taken in response given in *italics* after each point.

It should be pointed out that some of the things relating to the software are outside of the control of the software designers. Specifically, issues relating to the installation procedure are due to the design and operation of the MS Visual Basic software installer and the relevant operating system. Accordingly, it has not been possible to address these issues. In practice, we have not found any problems if the user just follows the instructions and keeps the most recent files on their system.

However, the user interface merely provides a convenient way for users to develop analytical solutions and weather files. In fact, we did not originally propose to develop the user interface as part of 1052-RP, and developing an advanced installer program is well beyond the scope of our project. Unfortunately, the sort of installation quirks that GARD Analytics experienced seem to be inevitable with the standard Visual Basic installation tools. FORTRAN source code is provided for each test that should allow any user to work around any installation problems by compiling on their specific system.

The actual results from EnergyPlus are not given in this report for the reason given by GARD Analytics in their introduction below.

Response to Comments received from GARD Analytics

(letters dated 1/15/01 and 2/7/01)

GARD Analytics has reviewed the draft report for ASHRAE 1052RP “ASHRAE Analytical Test Suite – Building Fabric.” EnergyPlus input files have been developed for all test cases except Infiltration-2. Our comments are listed below. Comments 1 – 17 were provided to the ASHRAE development contractor in mid-January in a document dated *January 15, 2001*. Comments 18 – 28 were subsequently developed. All tests were run using the original toolkit software (dated 12/11/00) provided to GARD. A revised software package dated 1/24/01, which corrects some of the problems identified in the early set of comments, was also tested. In general, the Analytical Test Suite is a very valuable testing tool which provides excellent benchmarks for testing fundamental algorithms. In the interest of thoroughness, we have listed most everything which came to mind as the tests were performed. Some of the comments may be redundant between tests. We realize that we are working from a draft, so we understand that many of these items may not apply to your current revisions. Comments are listed in the order they were encountered. Spreadsheet files have also been provided which show current EnergyPlus results compared with the toolkit results. These are first-pass results with little or no attempt to resolve the source of discrepancies. ***Please note that the EnergyPlus results files are not for public distribution.*** If you have any questions, please feel free to contact us.

General Comments

- 1) Need Table of Contents and page numbers along with Introduction section explaining the steps recommended in using the test suite and software

An introductory section has been added since the test documents were first reviewed.

- 2) Need discussion about toolkit software
 - a. Machine requirements
 - b. Input screens and what the “write” button does
 - c. What output files are generated and their format
 - d. Generation of a weather file and formats

An introductory section has been added since the test documents were first reviewed

- 3) Software does not run and gives error if output files are stored in a directory other than the root directory where the program resides

This problem was remedied in the Beta 2 release of the software.

- 4) Would be nice to have the numerical output file (.DAT) available in other formats such as .CSV or .XLS so it could be opened directly as an Excel spreadsheet for plotting of results.

This feature was added in the Beta 2 release of the software.

- 5) Would be nice if the tab key could be used to move from one input field to the next. Currently, the tab behavior is a bit random.

This problem was addressed in the Beta 2 release of the software.

- 6) Need to explain that use of the toolkit software may require an iterative approach:
 - a. You have to first run the toolkit software to get a weather file
 - b. Then you have to use the weather file to perform a simulation with the program being tested (DOE-2, BLAST, EnergyPlus, etc) to determine the interior and exterior film coefficients that were used. This is necessary for programs such as EnergyPlus which provide no user-specified film coefficients, but the film coefficients are reported as output variables.
 - c. Finally, you have to rerun the toolkit software using the new interior and exterior film coefficients to get a set of results that can now be used to do a comparison

Some discussion of this issue has been added to the introduction to the documentation.

- 7) Input screens are inconsistent when specifying film coefficient data. SSConv and SSCond ask for coefficients A, C and n. TC1 asks for the outside film coefficient only. TC2 and TC3 asks for outside and inside film coefficient. Probably should give user the option of either putting in film coefficients themselves or using the A, C and n coefficients

This is unavoidable and is simply due to the nature of the different analytical solutions used. (A possible alternative would be to not allow film correlation coefficients for any test, but that seems unnecessarily restrictive.)

- 8) Include an example in the documentation of how you take hourly interior and exterior film coefficient results from the program being tested and calculate the A, C and n coefficients in the h equation

This has been added to the introduction to the documentation.

- 9) Add comment about inside surface emissivities --- they should be set to a low number

Some discussion of this issue has been added to the introduction to the documentation.

Comments for Specific Tests

10) Test SSConv

- a. Comment in paragraph #2 about any surface of the cube can be selected as the test surface because they will give same results is not true in DOE-2 and EnergyPlus. These programs distinguish between the direction of heat flow when setting the film coefficients. Walls have horizontal heat flows; ceilings and floors have vertical heat flows. See Table 1, Chapter 24, 1997 ASHRAE Handbook of Fundamentals.

We have avoided comments about specific programs but our meaning was evidently unclear and the text has been modified.

11) Test SSCond

- a. In Table SSCond-A, need to specify the order of the layers. Is it from outside to inside or vice versa?

The order is in fact irrelevant (being a steady state test) and a note about this has been added to the documentation.

12) Test TC1

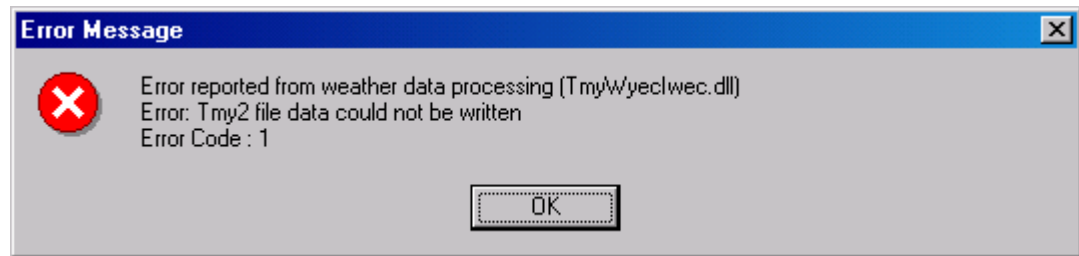
- a. Reduce the number of places after the decimal in table of results

The excessive decimal places were mainly for debugging/verification purposes. All the tables have now been amended.

- b. Input screen asks for outside film coefficient but output file (.DAT) echoes input as inside film coefficient

This has been corrected in the Beta 2 release of the software.

- c. The toolkit has been installed on two different Windows 2000 machines. Both machines have Office 2000 and Explorer 5.50 installed. Machine B also has Front Page 2000 and Compaq Visual Fortran 6.5 installed. There are also various sundry applications installed on both machines. On Machine A, many of the test runs bomb when trying to create the weather file.



But on Machine B, everything runs fine as long as the current working directory is always the directory where the toolkit software was installed (see comment #3 above).

This has been corrected in the Beta 2 release of the software.

- d. The TMY2 file created by the software for TC1, TC2 and TC3 shows LATITUDE=0, LONGITUDE=0 AND TIME ZONE=-6. This causes an error in EnergyPlus because the LAT and LONG do not correspond with the TZN.

This has been corrected in the Beta 2 release of the software.

- e. On TMY2 file created by toolkit, DEWPOINT=999.9 and RELHUM=0. This produces a problem when using weather processors to create an EnergyPlus compatible weather file.

This has been corrected in the Beta 2 release of the software.

13) Test TC2

- a. Column labeled "Internal Load" in Tables TC2-B and TC2-C should be labeled "Zone Load" as it is in the .DAT file.

This has been corrected in the documentation.

14) Test TC3

- a. Input screen has not been included in documentation

This has been added.

- b. Order of layers in Table TC3-A? Is Layer 1 the exterior layer?

A note about the order has been added to the table

- c. Reduce the number of places after the decimal in Table TC3-B

The decimal places shown have been modified.

- d. Column labels in .DAT file for hourly results should be revised to use same labels as were used in the .DAT file for TC2.

Less information is available from the TC3 solution and that is why the columns of data output are different.

15) Test ExtSolRad

- a. Order of layers in Table ExtSolRad-A and Table ExtSolRad-C?

A note has been added to the table.

- b. Last column in Table ExtSolRad-B and Table ExtSolRad-D should be changed to read “Internal Surface Heat Flux”

The table has been modified.

- c. In Figure ExtSolRad-2 and Figure ExtSolRad-3 the labels should be more explicit – “Solar_flux” should be changed to “Incident Solar Flux”; “Internal heat flux” should be changed to Internal Surface Heat Flux”

The table has been modified.

- d. Column labels in .DAT file for hourly results should be more explicit as to what they are – Ts_in, Ts_out, H_in, H_out, Solar_flux, Flux_in and Flux_out are not clear enough

This has been modified in the Beta2 release of the software.

- e. The ASHRAE and EnergyPlus results are out of phase by 30-60 minutes – not sure whose problem it is yet. Is the ASHRAE toolkit software recognizing daylight savings time? Is the solar radiation being calculated by the toolkit software for the beginning of the time step, middle of the time step or end of the time step? Is the data in the TMY2 file for the beginning, middle, or end of the hour?

The times used in the solar data are standard times for the appropriate time zone. No daylight savings time is applied. All the weather file formats used in the test suite software define the solar irradiation data in terms of the total energy received during the last hour.

16) Test SolRadGlazing

- a. Is the “standard clear single pane glazing” mentioned in the fourth paragraph of 1/8” thickness?

Yes - A note has been added to clarify this.

- b. Define what “black” surfaces means. I presume it means setting the internal surface emissivity=1.0?

Yes - A note has been added to clarify this.

- c. In Figure SolRadGlazing-2 and Figure SolRadGlazing-3 the labels should be more explicit – “Solar_flux” should be changed to “Incident Solar Flux”; “Internal heat flux into the zone” should be changed to Internal Surface Heat Flux”

The figure labels have been improved accordingly.

- d. Column labels in .DAT file for hourly results should be more explicit as to what they are – Ts, H_in, H_out, Solar_flux, Flux_tol and Load are not clear enough.

This has been modified in the Beta2 version of the software.

17) Test SolRadShade

- a. There is some terminology in the first paragraph that should be further defined. What does “semi-infinite” and “union of semi-infinite” mean? Also, what is “hanging length” in the 5th paragraph mean? Need to add diagrams describing the types of shading being discussed.

Some explanation of these terms has been added.

- b. Need diagrams for the cases being described in Table SolRadShade-A.

New diagrams have been added.

- c. Column labels in Table SolRadShade-B and Table SolRadShade-D are not clear – “HorVer”, “Hor” and “Ver” ???

Better labels have been added.

General Comments – Part 2

- 18) The WRITE button at the bottom of the input screen does not really write a file and the SAVE button does not really save the DAT file. If you get passed the “save” for the DAT file and bomb during the weather file creation (as has happened to us many times), no DAT file is created. Should probably rename the WRITE button “CALCULATE”

This is now shown as ‘Start Calculation’.

- 19) There are no engineering units shown for any of the output results in the DAT file.

This is a deliberate design decision. The analytical solutions themselves are independent of any unit system. So is the code (the Fortran90) that implements them. Hence, any consistent set of units used in the input will result in the same units system for the output (i.e. SI in gives SI out, IP in gives IP out).

Comments for Specific Tests – Part 2

20) Test IntSolarDist

- a. What is the type of construction for wall with window in it – heavyweight?

Heavyweight (see d below)

- b. What is the vertical length of vertical fins – cannot tell from Figure IntSolarDist-2? Is it the same as the height of the window?

0.5m (see d below)

- c. Not clear what the column titles in the DAT file mean

These have been simplified.

- d. Not clear what is meant by “Zone Load” in DAT file and in documentation. (This comment applies to other tests also) In EnergyPlus, the solar irradiation hitting a surface is absorbed and then shows up as a delayed cooling load in the zone some time later. It appears that the “Zone Load” talked about in 1052RP is the “solar load transmitted into the space,” but not sure.

As a number of issues in this test were unclear the documentation was given a more significant overhaul and re-reviewed.

21) Test Infiltration-1

- a. Need to say test is for sea level conditions so proper air density is used.

A comment about this has been added.

- b. When using the TM2 file created by 1052RP and converting to EnergyPlus format, our outdoor density and enthalpy differ from the 1052RP results because EnergyPlus is using drybulb and dewpoint temperatures to calculate the other psychrometric parameters.

It is not clear why the enthalpies would be different. It should be possible to find a drybulb/dewpoint specification that is consistent with the drybulb/humidity ratio definition used by the test suite.

22) Test IntHeatGain

- a. The two TXT output files generated are of no interest to the user and should not be saved

These are no longer used (they were for debugging).

- b. Should alert the user that he will not be prompted for a weather file name since no weather file is generated for this test.

A comment about this has been added in the documentation.

- c. This test runs take much more time to run than others --- others run in seconds while Test IntHeatGain runs more than 1 minute on our new 1200 MH machines. And during this entire time the Progress Chart bar never moves causing the user to think that the run has hung up. Progress bar should be moving to the right the whole time test is running.

We have looked into the run time issue. The main time consuming effort is in finding the roots of the transcendental equation. It appears that we need many roots to get the required accuracy – a minute doesn't seem to long!? It is not apparent how the progress bar can be improved. A note has been added about the long run time

- d. The DAT file shows time period of run in seconds – could be confusing. Should probably be changed to 1 hour increments to conform with what is being shown in tables and along X-axis of plots in documentation

We have just tried to be consistent in our use of units (SI input gives SI output).

- e. Need to tell reader that depending upon the simulation program being used, a warm-up period of one week or more with 0 W internal load may be required before the 27,000 W load is turned on to properly initialize surface temperatures

We have reduced the load to 3kW now and this may no longer be such a problem but we have added a note in the introduction about this issue.

- f. For programs like EnergyPlus and DOE-2 where the internal film coefficient is changing each hour and is different for walls, ceiling and floors, we used the average values for the 6 surfaces over the period of the simulation and used that for the toolkit input value. There's problem with toolkit asking for a constant interior film coefficient since many simulation programs do not allow user to set a constant inside film coefficients.

Analytical solutions for transient conduction problems are only feasible with constant convection coefficients. It is perhaps unfortunate that EnergyPlus does not allow the user to use constant coefficients.

- g. Input screen for this test allows user to only set inside film coefficient at a constant and not with coefficients like in other tests

See response to f. above.

- h. Surface temperature results are very sensitive to value set for film coefficient. We ended up setting value of film coefficient in toolkit run equal to EnergyPlus average value for the 6 surfaces over the 168 hour load-on period only.

See f. above.

23) Test IntRad

- a. Is the word “cuboid” in the documentation actually a word?

Yes!

- b. In Tables IntRad 2 through 6, the “Surface Index” column does not give any indication as to which wall is which is “external”, which wall is “opposite”, and which walls are “other”??

The tables have been overhauled and a note added about the numbering.

- c. Output report does not specify any engineering units for inputs or results. Column titles for output variables need to be better defined

The output has been simplified.

- d. For programs like EnergyPlus and DOE-2 where the internal film coefficient is changing each hour and is different for walls, ceiling and floors, we used the average values for the 6 surfaces over the period of the simulation and used that as the toolkit input value.

See 22) f. above.

- e. Aspect ratio seems to have very little impact on surface temperature in EnergyPlus. May be a bug in EnergyPlus or maybe not setting up EnergyPlus inputs properly.

This is an EnergyPlus issue. In the documented test results the zone load is more sensitive to aspect ratio than the surface temperatures.

24) Test ExtLWRad

- a. EnergyPlus calculates the sky temperature each hour using the outdoor relative humidity from the weather tape. At 10% RH, EnergyPlus gives 1 C sky temperature; at 50% RH, EnergyPlus gives 20 C. At these sky temperatures, EnergyPlus is calculating zone loads = 0.0. 1052RP toolkit

is calculating zone loads 0 to -1 W. The example in the documentation is giving much, much larger loads, >400 W, and a sky temperature of 7 C.

We have investigated this – results are documented elsewhere in the report.

- b. For programs like EnergyPlus and DOE-2 where the internal film coefficient is changing each hour and is different for walls, ceiling and floors, we used the average values for the 6 surfaces over the period of the simulation and used that for the toolkit input value.

See 22) f. above.

25) Test GrdCoup

- a. EnergyPlus does not yet account for edge losses for slab-on-grade floors.

Acknowledged. Ditto BLAST.

- b. Output report does not specify any engineering units for inputs or results
- c. Results are presented for “Conduction Heat flow through the slab” and “Convection Heat flow through the slab”. This is very confusing. Should probably be relabeled as “Convection Heat flow from the slab.”

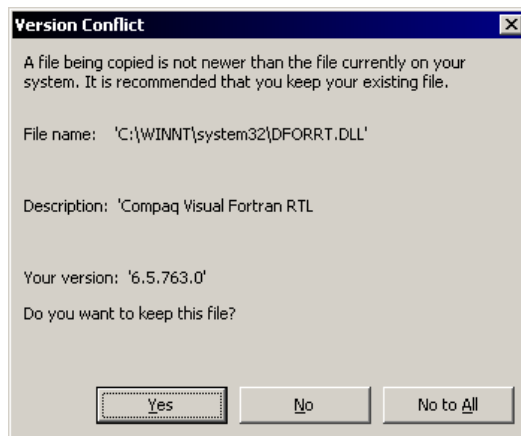
The output has been simplified and no longer shows these fields (these were really for debugging/verification).

- d. Should alert user that he will not be prompted for a weather file name since test does not create a weather file.

A note has been added about this in the documentation.

26) Installation on Windows 2000

Installation of the toolkit software cause several prompts regarding install files which were older than those present on the workstation. In all cases we kept the newer versions rather than overwriting with an older version. The installer should catch this so that a user who doesn't know any better cannot trash his system with old dlls. The files which produced these prompts are:



This is outside of our control. See introduction to this section.

27) Time Parameter in CSV File

Problems occur when copying data from the CSV file and pasting them into an Excel spreadsheet. Clock times with single digit hours (1:00AM through 9:50AM) are written in the CSV file with a leading blank space. Excel does not recognize these as being in time format but only as a string of digits. This causes problems when you try and create a chart with time as the x-axis. When the CSV file is created eliminate the leading space for times between 1:00 and 9:50.

This has been fixed by adjusting the output format.

Round #3 - General Comments

In March 2001 a Round #3 of simulations were run using ASHRAE 1052 Toolkit Software Version 2 and EnergyPlus Version 1 Build 02. The following highlights continuing problems and latest results.

- 28) When using the EnergyPlus Weather conversion routine to produce an EPW format file, the latitude and longitude get written as '.' with no numeric. This causes a fatal error in EnergyPlus. The 1052 Toolkit TM2 file is okay. The problem is with the EnergyPlus Weather converter.

As noted this is not a 1052-RP problem.

- 29) The 1052 Toolkit is still writing a DPT=0 and RH=100% on the TM2 weather file. This is causing psychrometric errors in EnergyPlus.

This has been fixed by setting the dry bulb and dpt the same.

- 30) The CSV file created by the 1052 Toolkit writes a series of rows at the top of the file which echos input data and reports some results. These rows need to have the text separated from the numeric value by a 'comma' so that the number can be linked to by an Excel spreadsheet.

Commas have now been added as requested

- 31) The problem described in comment #27 above still exists.

See reply to #27.

- 32) Still not indicating a version number in the help menu or on the CSV output files so never sure which version of the software was used for a set of runs.

A version number has been added in the about screen

- 33) When you save a set of input screens and then reopen, for those tests where a test day is specified, the day gets reset to June 21 even though you saved the input screens with the day set to August 21.

This feature now works as expected.

Source Code Listings

Each analytical test has been implemented in Fortran90/95. This has been done by writing the necessary subroutines and functions and making use of the ‘module’ concept in Fortran90. This code has been tested with driver programs also written in Fortran90 and has also been tested in the form of dynamic link libraries used with the user interface program (written in Visual Basic 6). The compiler used for development and testing was Compaq Visual Fortran 6.1.

Listings of the Fortran90 modules that are used to calculate the analytical solutions are given later in this section. The module and subroutine names, along with the tests in which they are used are indexed in the table below.

Module Name	Subroutines and Functions Contained	Analytical Tests
SSConv	SSConvection	SSConv SSCond
ConductUtil	FindTransRoots	TC1 TC2 ExtLWRad IntHeatGain
Adiabatic	AdiabaticStep; F1	TC1
HomoStep	HomogeneousStep; F2; Norm	TC2
Periodic	PeriodicCond	TC3
ExtSolRadiation	ExtSolRad; SolIrrad_ND; Heat_Balance; Incid_Angle	ExtSolRadiation
ExtSolRadiationGs	ExtSolRadiation_Gs; Incid_Angle; SolIrrad_ND; Absorp_Transmit; SunlitAreaHorVer; SunlitAreaHor; SunlitAreaVer; HeatBalance	SolRadGlazing SolRadShade
WindowReveal	WinReveal; Incid_Angle; SolIrrad_ND; Absorp_Transmit; SunlitAreaHorVer; SunlitAreaHor; SunlitAreaVer; HeatBalance	WinReveal
SolRadDistribution	SolRadDist; SolRadDist; SolIrrad_ND; Absorp_Transmit; SunlitAreaSetback; SunlitAreaHorVer; SunlitAreaHor; SunlitAreaVer; HeatBalance	SolRadDist
IntRad_1	IntRadiation; ViewFactor; JacobianFun; NRNonlinear; LUDecomp; LUSubstitution;	IntLWRad
MatrixInverse	SGECO; SGEFA; SGEDI; SAXPY; SSCAL; SDOT; SASUM; ISAMAX; SSWAP	IntLWRad
ExtLWRad	ExtLWRadiation; F1	ExtLWRad

Infilt_1	Infiltration_1	Infiltration 1
Infilt_2	Infiltration_2	Infiltration 2
PsychrometricsMod	TDB_H; TDB_RH; TDB_TW; TDB_W; DEWPOINT; DRYBULB; ENTHALPY; ENTHSAT; HUMRATIO; HUMTH; RELHUM; RHODRY; RHOMOIST; SATPRESS; SATTEMP; TAIRSAT; WETBULB; XITERATE	Infiltration 1 Infiltration 2
IntHeatGain	IntHeatGain_Rad; F1; Temperature	IntHeatGain
Grdcoup	GroundCoupling	GrdCoup
TMY2Util	SetNewTmy2; SetTmy2Header; SetTmy2DryBulb; SetTmy2DewPoint; SetTmy2Temps; SetTmy2Pressure; SetTmy2DirNormIrr; SetTmy2RelHumid; WriteTmy2	All Tests
WYEC2Util	SetNewWyec2; SetWyec2Header; SetWyec2DryBulb; SetWyec2DewPoint; SetWyec2Temps; SetWyec2Pressure; SetWyec2DirNormIrr; SetWyec2RelHumid; WriteWyec2	All Tests
IwecUtil	SetNewIwec; SetIwecHeader; SetIwecDryBulb; SetIwecDewPoint; SetIwecTemps; SetIwecPressure; SetIwecDirNormIrr; SetIwecRelHumid; WriteIwec	All Tests

Source Code: Module SSConv

MODULE SSConv

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to the steady-state convection
!               and conduction st cases.
!
! CREATED:      7.23.99      S.J.Rees
!
! MODIFIED:
!
PRIVATE
! make public main subroutine
PUBLIC :: SSConvection

CONTAINS

!-----

SUBROUTINE SSConvection(FileName, Cond, N_Cond, Thickness, &
                        N_thickness,T_outside, T_inside, Hin_A, Hin_C, &
                        Hin_n, Hout_A,Hout_C, Hout_n, Flux, Ts_in, &
                        Ts_out, H_in, H_out, ErrorFlag, ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SSConvection" :: SSConvection

! PURPOSE:      Produces weather files and analytical response for
!               the case of steady-state convection and conduction.
!               correlations inside and outside are specified as
!               The convection  $H = A + C \cdot (T_a - T_s)^n$ .
!               The steady state heat flux is found using a successive
!               substitution method to solve the non-linear equation.
!               A variable number of wall layers can be specified.
!               This enables a single high conductance wall to be
!               used to test the convection models, and more realistic

```

```

!           multi-layer walls to test steady state conduction.
!
!   CREATED:
!   7.23.99      S.J.Rees
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!
!   FileName.dat      -   Analytical response data
!
!   SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   Cond              -   array of wall conductivities
!   N_Conc            -   Number of conductivities in array
!   Thickness         -   array of wall Thicknesses
!   N_Thickness       -   Number of Thicknesses in array
!   T_outside         -   Outside air temperature
!   T_inside          -   Inside air temperature
!   Hin_A             -   Conv. correlation coefficient A for inside
!   Hin_C             -   Conv. correlation coefficient C for inside
!   Hin_n             -   Conv. correlation coefficient n for inside
!   Hout_A            -   Conv. correlation coefficient A for outside
!   Hout_C            -   Conv. correlation coefficient C for outside
!   Hout_n            -   Conv. correlation coefficient n for outside
!   Flux              -   Calculated steady state heat flux
!   Ts_in             -   Calculated inside surface temperature
!   Ts_out            -   Calculated outside surface temperature
!   H_in              -   Calculated inside conv. coefficient
!   H_out             -   Calculated outside conv. coefficient
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE
! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 1000
INTEGER, PARAMETER :: Results = 21
! unit number for analytical response output file
INTEGER, PARAMETER :: N_Hours=8760      ! No. of hours in year
REAL(prec2), PARAMETER :: Tolerance = 1.0E-010 ! tolerance

! arguments
INTEGER, INTENT(IN) :: N_Conc, N_thickness
CHARACTER *(*) , INTENT(IN) :: FileName
REAL(prec2), DIMENSION(N_Conc), INTENT(IN) :: Cond
REAL(prec2), DIMENSION(N_thickness), INTENT(IN) :: Thickness
REAL(prec2), INTENT(IN) :: T_outside, T_inside
REAL(prec2), INTENT(IN) :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(IN) :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(OUT) :: Flux
REAL(prec2), INTENT(OUT) :: Ts_in, Ts_out
REAL(prec2), INTENT(OUT) :: H_in, H_out

```

```

CHARACTER*(*), INTENT(OUT) :: ErrorMessage
INTEGER(2), INTENT(OUT) :: ErrorFlag ! two byte int for VB integer

! local variables
REAL(prec2) :: Fabric_R ! Slab thermal resistance (surface to surface)
REAL(prec2) :: R_total ! Overall thermal resistance (air to air)
REAL(prec2) :: Flux_inside ! Flux calculated between air and
                        ! surface inside
REAL(prec2) :: Flux_outside ! Flux calculated between air and
                        ! surface outside
REAL(prec2) :: Flux_fabric ! Flux calculated between outside
                        ! surface and inside surface
REAL(prec2) :: MinFlux, MaxFlux ! Min and max calculated flux values
INTEGER :: i, n, error
CHARACTER(LEN=132) :: dummy
INTEGER, PARAMETER :: DebugUnit = 20
LOGICAL, PARAMETER :: Debug = .false.
LOGICAL, PARAMETER :: Warning = .false.

! do some initial error checking
IF(N_Thickness /= N_Cond)THEN
    ErrorMessage = "Fabric data is inconsistant"
    ErrorFlag = 1
    RETURN
END IF
IF(ABS(T_outside - T_inside ) < Tolerance)THEN
    ErrorMessage = "Inside and outside temperatures &
                    appear to be the same ??"
    ErrorFlag = 1
    RETURN
END IF

! open a unit for debug output
IF(Debug .and. DebugUnit/=6)THEN
    OPEN (DebugUnit, file="SSConv.log", status='unknown', IOSTAT=error)
    IF(error/=0) THEN
        ErrorMessage="Debug output file could not be opened"
        ErrorFlag = 1
        RETURN
    END IF
    ! copy out arguments to debug file
    WRITE(DebugUnit, '("SSConv arguments:")')
    WRITE(DebugUnit, '("Inside convection correlation coefficient A =", &
                        F10.6)') Hin_A
    WRITE(DebugUnit, '("Inside convection correlation coefficient C =", &
                        F10.6)') Hin_C
    WRITE(DebugUnit, '("Inside convection correlation coefficient n =", &
                        F10.6)') Hin_n
    WRITE(DebugUnit, '("Outside convection correlation coefficient A =", &
                        F10.6)') Hout_A
    WRITE(DebugUnit, '("Outside convection correlation coefficient C =", &
                        F10.6)') Hout_C
    WRITE(DebugUnit, '("Outside convection correlation coefficient n =", &
                        F10.6)') Hout_n
    WRITE(DebugUnit, '("Outside air temperature                    =", &
                        F10.6)') T_outside
    WRITE(DebugUnit, '("Inside air temperature                    =", &

```

```

        F10.6)' ) T_inside
    DO i=1,N_Connd
        WRITE(DebugUnit,'("Fabric layer",I2," conductivity
        F10.6)' ) i, Cond(i)
        WRITE(DebugUnit,'("Fabric layer",I2," thickness
        F10.6)' ) i, Thickness(i)
    END DO
END IF

! set initial guess of surface temperatures
Ts_out = T_outside - 0.33*(T_outside - T_inside)
Ts_in = T_outside - 0.67*(T_outside - T_inside)
! calculate the resistance of the slab
Fabric_R = 0.0
DO i = 1, N_Connd
    IF(Cond(i)==0.0)THEN
        WRITE(ErrorMessage,'("Conductivity of layer ", I2, &
        " appears to be zero")' ) i
        ErrorFlag = 1
        RETURN
    END IF
    Fabric_R = Fabric_R + Thickness(i)/Cond(i)
END DO

H_in = Hin_A + Hin_C*ABS(T_inside - Ts_in)**Hin_n
H_out = Hout_A + Hout_C*ABS(T_outside - Ts_out)**Hout_n

! start loop to find Flux by successive substitution
n = 1
DO
    ! calculate total resistance
    IF(H_in==0.0 .or. H_out==0.0)THEN
        WRITE(ErrorMessage,'("Inside or outside convection coefficient &
        calculated to be zero", " - can not procede")' )
        ErrorFlag = 1
        RETURN
    END IF
    R_total = 1.0/H_in + Fabric_R + 1.0/H_out
    ! flux based on current value of total resistance
    Flux = (T_outside - T_inside) / R_total
    ! update surface temps based on flux
    Ts_out = T_outside - Flux/H_out
    Ts_in = T_inside + Flux/H_in
    ! re-calculate conv. coef using current temps
    H_in = Hin_A + Hin_C*(ABS(T_inside - Ts_in)**Hin_n)
    H_out = Hout_A + Hout_C*(ABS(T_outside - Ts_out)**Hout_n)

    ! calculate fluxes over different resistances for checking
    Flux_inside = H_in*(Ts_in - T_inside)
    Flux_outside = H_out*(T_outside - Ts_out)
    Flux_fabric = (Ts_out - Ts_in) / Fabric_R

    ! some debug output
    IF(Debug)THEN
        WRITE(DebugUnit,'("H_in = ",F8.5," H_out = ",F8.5," Ts_in = ", &

```

```

        F8.5," Ts_out = ",F8.5)')H_in, H_out, Ts_in, Ts_out
WRITE(DebugUnit,'("Q = ",F8.5," Qout = ",F8.5," Qin = ",F8.5,&
    " Qfab = ",F8.5)') Flux, Flux_outside, Flux_inside, Flux_fabric
END IF

! calc min and max fluxes
MinFlux = MIN(Flux, Flux_inside, Flux_outside, Flux_fabric)
MaxFlux = MAX(Flux, Flux_inside, Flux_outside, Flux_fabric)
! test for exit condition
IF(ABS(MaxFlux-MinFlux) < Tolerance .or. n > Max_Iterations) EXIT
n = n + 1
END DO

! set warning if max iterations exceded
IF(n > Max_Iterations)THEN
    ErrorMessage = "The steady-state heat flux could not be calculated &
        within the allowable number of iterations"
    ErrorFlag = -1
    RETURN
END IF

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results,'("# ASHRAE Analytical Test Suite - Results file")')
WRITE(Results,'("# Steady state convection/conduction test")')
WRITE(Results,'("#")')
WRITE(Results,'(("//"Test Parameters -")')

WRITE(Results,'("Inside convection correlation coefficient A =",",",&
    F15.4)') Hin_A
WRITE(Results,'("Inside convection correlation coefficient C =",",",&
    F15.4)') Hin_C
WRITE(Results,'("Inside convection correlation coefficient n =",",",&
    F15.4)') Hin_n
WRITE(Results,'("Outside convection correlation coefficient A =",",",&
    F15.4)') Hout_A
WRITE(Results,'("Outside convection correlation coefficient C =",",",&
    F15.4)') Hout_C
WRITE(Results,'("Outside convection correlation coefficient n =",",",&
    F15.4)') Hout_n

WRITE(Results,'("Outside air temperature =",",",F15.4)') T_outside
WRITE(Results,'("Inside air temperature =",",",F15.4)') T_inside

DO i=1,N_Con
    WRITE(Results,'("Fabric layer",I2," conductivity =",",", &
        F15.4)') i, Cond(i)
    WRITE(Results,'("Fabric layer",I2," thickness =", ",",&
        F15.4)') i, Thickness(i)
END DO

```

```
WRITE(Results,'(// "Test results -")')
WRITE(Results,'("Heat flux =",",",F15.4)') Flux
WRITE(Results,'("Inside surface temperature =",",",F15.4)') Ts_in
WRITE(Results,'("Outside surface temperature =",",",F15.4)') Ts_out
WRITE(Results,'("Inside convection coefficient =",",",F15.4)') H_in
WRITE(Results,'("Outside convection coefficient =",",",F15.4)') H_out

CLOSE(Results)
IF(Debug)THEN
    CLOSE(DebugUnit)
END IF

END SUBROUTINE SSConvection

END MODULE SSConv
```

Source Code: Module ConductUtil

MODULE ConductUtil

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to conduction test cases.
!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:
!

```

CONTAINS

```

!-----
SUBROUTINE FindTransRoots(f, Param, Alpha, N_roots, Debug)

```

```

! PURPOSE:      finds the roots of a given transcendental function using
!               a bisection method. Search starts near zero and proceeds
!               in +ve direction in steps X_Delta until N_roots are
!               found. The basic bisection method is modified so that
!               a root is not detected incorrectly at an asymptote
!               position. (Can occur with  $x \cdot \tan(x) = 0$ ) This is done by
!               testing for a high value of the function gradient as
!               well as the usual  $f(x_l) \cdot f(x_r) < 0$ .
!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
! n/a
!
! SUBROUTINE ARGUMENTS:
!   f             - name of transcendental function procedure
!   Param          - Array of parameters for transcendental

```

```

!           equation
!   Alpha   -   array of roots of the function
!   N_roots  -   number of roots to be searched for
!   Debug    -   Switch for printing debug info.

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)
! arguments
INTEGER, INTENT(IN) :: N_roots
REAL(prec), EXTERNAL :: f
REAL(prec), DIMENSION(N_roots), INTENT(INOUT) :: Alpha
REAL(prec), DIMENSION(:), INTENT(IN) :: Param
LOGICAL, INTENT(IN) :: Debug

! local variables
REAL(prec) :: Xl_Start, Xr_Start      ! X at left and right
                                           ! of initial interval
REAL(prec) :: Delta                  ! Current size of
                                           ! interval during search
REAL(prec) :: X_Left, X_Mid, X_Right ! X at left, middle and
                                           ! right in search
REAL(prec) :: V_Left, V_Mid, V_Right ! f(x) at left, middle and
                                           ! right in search
INTEGER :: N_bisecs                  ! search iteration counter
INTEGER :: n                         ! root counter
! parameters
INTEGER, PARAMETER :: Max_Iterations = 50 ! limit on No of iterations
REAL(prec), PARAMETER :: Tolerance = 1.0E-12 ! tolerance on x
REAL(prec), PARAMETER :: X_Delta = 0.001    ! increment for x

Xl_Start = 0.0001 ! starting at zero may cause FP errors with some f(x)
Xr_Start = Xl_Start+X_Delta
n=1
DO
  ! starting values of function
  V_Right = f(Xl_Start, Param)
  V_Left = f(Xr_Start, Param)
  IF(V_Left*V_Right >= 0.0.or.ABS((V_Left-V_Right)/X_Delta)>1.0E+04)THEN
    ! no root in this interval - increment interval
    Xl_Start = Xl_Start + X_Delta
    Xr_Start = Xr_Start + X_Delta
  ELSE ! root must be in this interval
    X_Left = Xl_Start
    X_Right = Xr_Start
    DO N_bisecs = 0, Max_Iterations ! start search for root
                                           ! in current interval
      V_Right = f(X_Right, Param)
      V_Left = f(X_Left, Param)
      Delta = 0.5 * (X_Right-X_Left)
      X_Mid = X_Left + Delta
      ! exit loop if root is found to tolerance
      IF(Delta < Tolerance)EXIT
      V_Mid = f(X_Mid, Param)
      IF(V_Left*V_Mid < 0.0)THEN
        ! a root lies in the left interval

```



```
        X_Right = X_Mid
        V_Right = V_Mid
    ELSE
        ! a root lies in the right interval
        X_Left = X_Mid
        V_Left = V_Mid
    END IF
END DO
! printout for debugging only
IF(Debug)THEN
    PRINT '(2I4," The root is ",2X,F16.12, " +/- ", E12.6)', &
        n,N_bisecs, X_Mid, Delta
END IF
! assign final value of X_Mid to array of roots
Alpha(n) = X_Mid
n=n+1
IF(n>N_roots)RETURN
! increment interval
Xl_Start = Xl_Start + X_Delta
Xr_Start = Xr_Start + X_Delta
END IF
END DO

END SUBROUTINE FindTransRoots

END MODULE ConductUtil
```

Source Code: Module Adiabatic

MODULE Adiabatic

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to conduction test case.
!               In this case the inside surface is adiabatic and the
!               flux is driven by a double step in outside temperature.
!
!
! CREATED:      5.20.99      S.J.Rees
!
! MODIFIED:
!
USE ConductUtil

PRIVATE
! make public main subroutine
PUBLIC :: AdiabaticStep

CONTAINS

!-----

SUBROUTINE AdiabaticStep(FileName, K, Rho, C_p, h_c, Thickness, &
                        T1, TempStep, TimeStep, Ext_temp, N_step, &
                        ErrorFlag, ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "AdiabaticStep" :: AdiabaticStep

! PURPOSE:      Produces weather files and analytical response for the
!               case of an step change in external air temperature and an
!               adiabatic internal wall. Step profile consists of:-
!               1. 90 days with external air temp at T1
!               2. 90 days with external air temp at T1+TempStep
!               3. 90 days with external air temp at T1-TempStep
!               4. 95 days with external air temp at T1

```

```

!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:
!   7.6.99      sjr -   added error flag and message to arguments
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
!
!   FileName.dat      -   Analytical response data
!   FileName.wy2      -   Weather file in WYEC2 format
!   FileName.tmy2     -   Weather file in TMY2 format
!
!
! SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   K                 -   slab conductivity
!   Rho               -   slab density
!   C_p              -   slab specific heat capacity
!   Thickness         -   slab thickness
!   Tl               -   Initial/final external air temperature
!   TempStep          -   Temperature Step
!   TimeStep          -   Time step is variable to deal with IP vs
!                       SI units. normally 1.0 or IP units,
!                       3600.0 for SI units for hourly time steps.
!   Ext_temp          -   External dry bulb temperature
!                       (for weather files)
!   N_step            -   Number of time steps (hours) - for VB calls
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE
! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: response=25 ! unit number for
                                ! analytical response output file
INTEGER, PARAMETER :: N_Hours=8760 ! No. of hours in year
INTEGER, PARAMETER :: N_roots=500 ! No. of roots of aux. equation
REAL(prec2), PARAMETER :: SS_Tolerance = 0.01 ! tolerance on steady
                                ! state temperature checking

! arguments
INTEGER, INTENT(IN) :: N_step
CHARACTER *(*) , INTENT(IN) :: FileName
REAL(prec2), INTENT(IN) :: K, Rho, C_p, h_c, Thickness
REAL(prec2), INTENT(IN) :: Tl, TempStep, TimeStep
REAL(prec1), DIMENSION(N_step), INTENT(OUT) :: Ext_temp
CHARACTER*(*) , INTENT(OUT) :: ErrorMessage
INTEGER(2), INTENT(OUT) :: ErrorFlag
                                !two byte int for VB integer

! local variables
REAL(prec2), DIMENSION(N_roots) :: Beta ! array of aux. eqn. roots

```

```

REAL(prec2), DIMENSION(3) :: Param      ! array of aux. eqn. parameters
REAL(prec2) :: Ext_Series, Int_Series    ! sum of exponential terms
REAL(prec2) :: time                      ! real variable for time
REAL(prec2) :: C_n                      ! coefficients in response series
REAL(prec2) :: N_Fourier                ! Fourier Number
REAL(prec2) :: Ext_Stemp                ! External Surface temperature
REAL(prec2) :: Int_Stemp                ! Internal Surface temperature
REAL(prec2) :: Alpha                   ! thermal diffusivity
INTEGER :: t,n,error                   ! loop counters, error flag
CHARACTER(LEN=132) :: dummy            ! dummy string
LOGICAL :: Debug = .false.             ! debug flag
LOGICAL :: Warning = .false.           ! warning flag

! calc thermal diffusivity
Alpha = K/(Rho*C_p)
! assign aux. eqn. parameters
Param(1) = h_c
Param(2) = K
Param(3) = Thickness

!open file for output of response
!attempt open - full path is passed as FileName by VB application
OPEN (response, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical response output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write some header information
WRITE(response, '( "# ASHRAE Analytical Test Suite - Results file") ')
WRITE(response, '( "# Transient conduction test TC1 - Adiabatic Wall") ')
WRITE(response, '( "#") ')
WRITE(response, '( "# Test Parameters -") ')
WRITE(response, '( "# Outside convection coefficient =",",",F15.4)') H_c
WRITE(response, '( "# Initial air temperature =",",",F15.4)') T1
WRITE(response, '( "# Outside air temperature Step =",",",F15.4)') TempStep
WRITE(response, '( "# Thermal conductivity =",",",F15.4)') K
WRITE(response, '( "# Wall material density =",",",F15.4)') Rho
WRITE(response, '( "# Specific heat capacity =",",",F15.4)') C_p
WRITE(response, '( "# Wall thickness =",",",F15.4)') Thickness
WRITE(response, '( "#",/, "# Test results -",/, "#") ')
WRITE(response, '( "# Time",",", "External Air Temp.",",", &
    "Internal Surface Temp.",",", "External Surface Temp.", &
    ",", "External Surface Flux") ')
!WRITE(response, '( "#",12X,"Air Temp.",2X,"Surface Temp.",2X, &
!    "Surface Temp.",3X,"Surface Flux",/)' )

! find roots of auxiliary equation
CALL FindTransRoots(f1, Param, Beta, N_roots, Debug)

! assign data for first 90 days
DO t=1, 90*24
    ! assign external temp array
    Ext_Temp(t) = T1

```

```

! write to response file
WRITE(response, '(1X,I6,"",F15.6,"",F15.6,"",F15.6,"",A15)') &
    t, T1, T1, T1, "0.000000"
END DO

! first step up after 90 days
DO t=90*24+1, 180*24
    time = REAL((t-90*24)*TimeStep)
    Ext_temp(t) = T1 + TempStep
    Ext_Series = 0.0
    Int_Series = 0.0
    DO n=1, N_Roots !calculate sum of exponential terms
        C_n = 4.0*DSIN(Beta(n))/(2.0*Beta(n)+DSIN(2.0*Beta(n)))
        N_Fourier = Alpha*time/(Thickness*Thickness)
        Ext_Series = Ext_Series + C_n*DEXP(-Beta(n)*Beta(n) &
            *N_Fourier)*DCOS(Beta(n))
        Int_Series = Int_Series + C_n*DEXP(-Beta(n)*Beta(n)*N_Fourier)
    END DO
    ! compute surface temp
    Ext_Stemp = T1 + TempStep - TempStep*Ext_Series
    Int_Stemp = T1 + TempStep - TempStep*Int_Series
    IF(Debug)THEN
        WRITE(*, '(1X,I6,4F15.6)') t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
            -h_c*(Ext_Stemp-Ext_temp(t))
    END IF
    ! write out response
    WRITE(response, '(1X,I6,"",F15.6,"",F15.6,"",F15.6,"",F15.6)') &
        t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
        -h_c*(Ext_Stemp-Ext_temp(t))
END DO

! check for steady state conditions
IF(ABS(Ext_temp(180*24)-Ext_Stemp)>SS_Tolerance)THEN
    Warning = .true.
END IF

! step down after 180 days
DO t=180*24+1, 270*24
    time = REAL((t-180*24)*TimeStep)
    Ext_temp(t) = T1 - TempStep
    Ext_Series = 0.0
    Int_Series = 0.0
    DO n=1, N_Roots !calculate sum of exponential terms
        C_n = 4.0*DSIN(Beta(n))/(2.0*Beta(n)+DSIN(2.0*Beta(n)))
        N_Fourier = Alpha*time/(Thickness*Thickness)
        Ext_Series = Ext_Series + C_n*DEXP(-Beta(n)*Beta(n) &
            *N_Fourier)*DCOS(Beta(n))
        Int_Series = Int_Series + C_n*DEXP(-Beta(n)*Beta(n)*N_Fourier)
    END DO
    ! compute surface temp
    Ext_Stemp = T1 - TempStep + 2.0*TempStep*Ext_Series
    Int_Stemp = T1 - TempStep + 2.0*TempStep*Int_Series
    IF(Debug)THEN
        WRITE(*, '(1X,I6,4F15.6)') t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
            -h_c*(Ext_Stemp-Ext_temp(t))
    END IF
    ! write out response
    WRITE(response, '(1X,I6,"",F15.6,"",F15.6,"",F15.6,"",F15.6)') &

```

```

        t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
        -h_c*(Ext_Stemp-Ext_temp(t))
    END DO
    ! check for steady state conditions
    IF(ABS(Ext_temp(270*24)-Ext_Stemp)>SS_Tolerance)THEN
        Warning = .true.
    END IF

    ! final step up after 270 days
    DO t=270*24+1, 365*24
        time = REAL((t-270*24)*TimeStep)
        Ext_temp(t) = T1
        Ext_Series = 0.0
        Int_Series = 0.0
        DO n=1, N_Roots !calculate sum of exponential terms
            C_n = 4.0*DSIN(Beta(n))/(2.0*Beta(n)+DSIN(2.0*Beta(n)))
            N_Fourier = Alpha*time/(Thickness*Thickness)
            Ext_Series = Ext_Series + C_n*DEXP(-Beta(n)*Beta(n) &
                *N_Fourier)*DCOS(Beta(n))
            Int_Series = Int_Series + C_n*DEXP(-Beta(n)*Beta(n)*N_Fourier)
        END DO
        ! compute surface temp
        Ext_Stemp = T1 - TempStep*Ext_Series
        Int_Stemp = T1 - TempStep*Int_Series
        IF(Debug)THEN
            WRITE(*, '(1X,I6,4F15.6)') t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
                -h_c*(Ext_Stemp-Ext_temp(t))
        END IF

        ! write out response
        WRITE(response, '(1X,I6, ",", F15.6, ", ", F15.6, ", ", F15.6, ", ", &
            F15.6)') t, Ext_temp(t), Int_Stemp, Ext_Stemp, &
            -h_c*(Ext_Stemp-Ext_temp(t))
    END DO
    ! check for steady state conditions
    IF(ABS(Ext_temp(365*24)-Ext_Stemp)>SS_Tolerance)THEN
        Warning = .true.
    END IF

    ! check if warning is needed
    IF(Warning)THEN
        IF(Debug)THEN
            WRITE(*, '(/"Warning: Steady state was not acheived at end &
                of each step.")')
            WRITE(*, '("      Comparison with analytical solution &
                may not be valid.")')
            WRITE(*, '("      Dry Bulb - Surface Temp. =",F12.8)') &
                Ext_temp(8760)-Ext_Stemp
            WRITE(*, '("      At the end of each step.")')
        END IF
        ErrorMessage = "Steady state was not acheived at end of each step &
            - results after the first step will not be valid!"
        ErrorFlag = -1
    END IF

    CLOSE(response)

```

```
END SUBROUTINE AdiabaticStep

!-----

FUNCTION f1(X, Param)

! PURPOSE:      Defines transcendental function used in analytical
!               solution to adiabatic wall test case with step in
!               air temperature as driving function
!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:
!

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)

REAL(prec) :: f1

REAL(prec), INTENT(IN) :: X
REAL(prec), DIMENSION(3) :: Param
REAL(prec) :: Biot_number, h_c, K, L

h_c = Param(1)
K = Param(2)
L = Param(3)

Biot_number = h_c*L/K

f1 = X*DTAN(X) - Biot_number

END FUNCTION f1

!-----

END MODULE Adiabatic
```

Source Code: Module HomoStep

MODULE HomoStep

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to conduction test case.
!               In this case the wall is homogeneous and has
!               convective boundary conditions inside and
!               out. The flux is driven by a double step in
!               outside air temperature.
!
! CREATED:      8.12.99      S.J.Rees
!
! MODIFIED:
!
USE ConductUtil

PRIVATE
! make public main subroutine
PUBLIC :: HomogeneousStep

CONTAINS

!-----

SUBROUTINE HomogeneousStep(FileName, K, Rho, C_p, h_in, h_out, Thickness, &
                           T_in, TempStep, TimeStep, T_out, N_step, ZoneArea, &
                           ErrorFlag, ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "HomogeneousStep" :: HomogeneousStep

! PURPOSE:      Produces weather files and analytical response for the
!               case of an step change in external air temperature and
!               fixed inside air temp. Step profile consists of:-
!               1. 90 days with external air temp at T_in
!               2. 90 days with external air temp at T_in+TempStep
!               3. 90 days with external air temp at T_in-TempStep

```



```

!           4. 95 days with external air temp at T_in
!
!   CREATED:
!   8.12.99      S.J.Rees
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!
!   FileName.dat      -   Analytical response data
!
!
!   SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   K                  -   slab conductivity
!   Rho                -   slab density
!   C_p               -   slab specific heat capacity
!   h_in              -   inside surface convection coefficient
!   h_out             -   outside surface convection coefficient
!   Thickness         -   slab thickness
!   T_in              -   Initial/final external air temperature
!   TempStep          -   Temperature Step
!   TimeStep          -   Time step is variable to deal with IP
!                       vs SI units. normally 1.0 or IP units,
!                       3600.0 for SI units for hourly time steps.
!   T_out             -   External dry bulb temperature
!                       (for weather files)
!   N_step            -   Number of time steps (hours) - for VB calls
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE
! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: response = 25 ! unit number for analytical
!response output file
INTEGER, PARAMETER :: N_Hours = 8760 ! No. of hours in year
INTEGER, PARAMETER :: N_roots = 500 ! No. of roots of aux. equation
!REAL, PARAMETER :: ZoneArea = 9.0 ! Zone Wall/floor area (m2)

! arguments
INTEGER, INTENT(IN) :: N_step
CHARACTER *(*) , INTENT(IN) :: FileName
REAL(prec2), INTENT(IN) :: K, Rho, C_p, h_in, h_out, Thickness
REAL(prec2), INTENT(IN) :: T_in, TempStep, TimeStep, ZoneArea
REAL(prec1), DIMENSION(N_step), INTENT(OUT) :: T_out
CHARACTER*(*) , INTENT(OUT) :: ErrorMessage
INTEGER(2), INTENT(OUT) :: ErrorFlag ! two byte int for VB integer

! local variables
REAL(prec2), DIMENSION(N_roots) :: Beta ! array of aux. eqn. roots
REAL(prec2), DIMENSION(N_roots) :: NormBeta

```

```

                                ! array of Norm function values
REAL(prec2), DIMENSION(4) :: Param      ! array of aux. eqn. parameters
REAL(prec2) :: Ext_Series, Int_Series    ! sum of exponential terms
REAL(prec2) :: time                      ! real variable for time
REAL(prec2) :: time1                    ! real variable for time at end of first step
REAL(prec2) :: time2                    ! real variable for time at end of second step
REAL(prec2) :: N_Fourier                 ! Fourier Number
REAL(prec2) :: Ext_Stemp                 ! External Surface temperature
REAL(prec2) :: Int_Stemp                 ! Internal Surface temperature
REAL(prec2) :: Alpha                    ! thermal diffusivity
REAL(prec2) :: Fourier, Fourier1, Fourier2 ! Fourier numbers
REAL(prec2) :: T_si, T_so                ! Steady-state Surface Temperatures
REAL(prec2) :: Theta_si, Theta_so       ! Non-dimensional surface Temperatures
REAL(prec2) :: Resistance                ! Overall thermal resistance
REAL(prec2) :: TransTerm                 ! Transient term in series solution
REAL(prec2), PARAMETER :: SMALL = 1.0E-30 ! utility parameter
REAL(prec2), PARAMETER :: BIG = 1.0E30    ! utility parameter
INTEGER :: t, n, error                   ! loop variables and error flag
LOGICAL :: Debug = .false.               ! debug flag

! calc thermal diffusivity
Alpha = K/(Rho*C_p)
! Set parameters for transcendental function
Param(1) = h_in
Param(2) = h_out
Param(3) = K
Param(4) = Thickness
! initialize T_out to T_in - for first period
T_out = T_in

IF(ABS(h_out)<SMALL)THEN
    ErrorMessage="Outside convection coefficient can not be zero"
    ErrorFlag = 1
    RETURN
END IF

!open file for output of response
!attempt open - full path is passed as FileName by VB application
OPEN (response, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical response output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write some header information
WRITE(response, '( "# ASHRAE Analytical Test Suite - Results file")')
WRITE(response, '( "# Transient conduction test - Homogeneous Wall")')
WRITE(response, '( "#")')
WRITE(response, '( "# Test Parameters -")')
WRITE(response, '( "# Inside convection coefficient = ", ",", ", F15.4)") H_in
WRITE(response, '( "# Outside convection coefficient = ", ",", ", F15.4)") H_out
WRITE(response, '( "# Outside air temperature Step = ", ",", ", F15.4)") TempStep
WRITE(response, '( "# Inside air temperature = ", ",", ", F15.4)") T_in
WRITE(response, '( "# Thermal conductivity = ", ",", ", F15.4)") K
WRITE(response, '( "# Wall material density = ", ",", ", F15.4)") Rho

```

```

WRITE(response, '("# Specific heat capacity =", "", F15.4)') C_p
WRITE(response, '("# Wall thickness =", "", F15.4)') Thickness
WRITE(response, '("#", "/", "# Test results -", "/", "#")')
WRITE(response, '("# Time", "", "External Air Temp.", "", &
    "Internal Surface Temp.", "", "External Surface Temp.", &
    "", "Internal Surface Flux", "", &
    "External Surface Flux", "", "Zone Load")')
!WRITE(response, '("#", 12X, "Air Temp.", 2X, "Surface Temp.", 2X, &
! "Surface Temp.", 3X, "Surface Flux", 3X, "Surface Flux", 7X "Load"/)')

! find roots of auxiliary equation
CALL FindTransRoots(f2, Param, Beta, N_roots, Debug)
! Find the norms and store in NormBeta array
DO n = 1, N_Roots
    NormBeta(n) = Norm(Beta(n), H_in, H_out, K, Thickness)
END DO

IF(Debug)THEN ! write out roots and norms
    DO n=1, N_roots
        Write(response, '("Beta = ", F20.12, "", " Norm(Beta) =", &
            F25.12)') Beta(n), NormBeta(n)
    END DO
END IF

! assign data for first 90 days
DO t=1, 90*24
    ! assign external temp array
    T_out(t) = T_in
    ! write to response file
    WRITE(response, '(1X, I6, "", F15.6, "", F15.6, "", F15.6, "", A15, &
        "", A15, "", A15)') t, T_in, T_in, T_in, "0.000000", &
        "0.000000", "0.000000"
END DO

! steady-state surface temps
IF(ABS(h_in)<SMALL)THEN ! to avoid div0 error when adiabatic
    Resistance = BIG
    T_si = T_in + Tempstep
ELSE
    Resistance = 1.0_8/h_in + Thickness/K + 1.0_8/h_out
    T_si = T_in + Tempstep/(h_in *Resistance)
END IF
T_so = T_in + Tempstep - Tempstep/(h_out *Resistance)

! first step up after 90 days
DO t=90*24+1, 180*24
    time = (t-90*24)*TimeStep
    T_out(t) = T_in + TempStep
    Ext_Series = 0.0_8
    Int_Series = 0.0_8
    Fourier = Alpha*time/(Thickness*Thickness)
    DO n= 1, N_Roots !,1,-1 !calculate sum of series terms
        TransTerm = EXP(-Beta(n)*Beta(n)*Fourier)/NormBeta(n)
        Ext_Series = Ext_Series + TransTerm
        Int_Series = Int_Series + (Beta(n)*DCOS(Beta(n)) + &

```

```

                (Thickness*h_out/K)*DSIN(Beta(n)))*TransTerm / Beta(n)
END DO
! compute surface temp
Theta_so = (T_so - T_in)/Tempstep - (Thickness*h_out/K)*Ext_Series
Theta_si = (T_si - T_in)/Tempstep - (Thickness*h_out/K)*Int_Series
Ext_Stemp = T_in + Theta_so*Tempstep
Int_Stemp = T_in + Theta_si*Tempstep

IF(Debug)THEN
    WRITE(*, '(1X,I6,5F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
        h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t))
END IF
! write out response
WRITE(response, '(1X,I6,"",F15.6,"",F15.6,"",F15.6,"",F15.6,"",&
    F15.6,"",F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
    h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t)), &
    ZoneArea*h_in*(Int_Stemp-T_in)
END DO

! step down after 180 days
! reset steady state terms for this step
T_si = 2*T_in - T_si
T_so = 2*T_in - T_so
timel = 90*24*TimeStep
DO t=180*24+1, 270*24
    time = (t-90*24)*TimeStep
    T_out(t) = T_in - TempStep
    Ext_Series = 0.0_8
    Int_Series = 0.0_8
    Fourier = Alpha*time/(Thickness*Thickness)
    Fourier1 = Alpha*(time-timel)/(Thickness*Thickness)
    DO n=1, N_Roots !calculate sum of series terms
        TransTerm = (2.0_8*DEXP(-Beta(n)*Beta(n)*Fourier1) - &
            DEXP(-Beta(n)*Beta(n)*Fourier)) / NormBeta(n)
        Ext_Series = Ext_Series + TransTerm
        Int_Series = Int_Series + (Beta(n)*DCOS(Beta(n)) + &
            (Thickness*h_out/K)*DSIN(Beta(n)))*TransTerm / Beta(n)
    END DO
    ! compute surface temp
    Theta_so = (T_so - T_in)/Tempstep + (Thickness*h_out/K)*Ext_Series
    Theta_si = (T_si - T_in)/Tempstep + (Thickness*h_out/K)*Int_Series
    Ext_Stemp = T_in + Theta_so*Tempstep
    Int_Stemp = T_in + Theta_si*Tempstep
    IF(Debug)THEN
        WRITE(*, '(1X,I6,5F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
            h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t))
    END IF
    ! write out response
    WRITE(response, '(1X,I6,"",F15.6,"",F15.6,"",F15.6,"",F15.6, &
        "",F15.6,"",F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
        h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t)), &
        ZoneArea*h_in*(Int_Stemp-T_in)
END DO

! final step up after 270 days

```

```

! reset steady state terms for this step
T_si = T_in
T_so = T_in
time2 = 180*24*TimeStep
DO t=270*24+1, 365*24
  time = (t-90*24)*TimeStep
  T_out(t) = T_in
  Ext_Series = 0.0_8
  Int_Series = 0.0_8
  Fourier = Alpha*time/(Thickness*Thickness)
  Fourier1 = Alpha*(time-time1)/(Thickness*Thickness)
  Fourier2 = Alpha*(time-time2)/(Thickness*Thickness)
  DO n=1, N_Roots !calculate sum of series terms
    TransTerm = (2.0_8*DEXP(-Beta(n)*Beta(n)*Fourier1) - &
      DEXP(-Beta(n)*Beta(n)*Fourier) - &
      DEXP(-Beta(n)*Beta(n)*Fourier2)) / NormBeta(n)
    Ext_Series = Ext_Series + TransTerm
    Int_Series = Int_Series + (Beta(n)*DCOS(Beta(n)) + &
      (Thickness*h_out/K)*DSIN(Beta(n)))*TransTerm / Beta(n)
  END DO
  ! compute surface temp
  Theta_so = (T_so - T_in)/Tempstep + (Thickness*h_out/K)*Ext_Series
  Theta_si = (T_si - T_in)/Tempstep + (Thickness*h_out/K)*Int_Series
  Ext_Stemp = T_in + Theta_so*Tempstep
  Int_Stemp = T_in + Theta_si*Tempstep
  IF(Debug)THEN
    WRITE(*, '(1X,I6,5F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
      h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t))
  END IF
  ! write out response
  WRITE(response, '(1X,I6,"",",F15.6,"",",F15.6,"",",F15.6,"",",F15.6,"",",&
    F15.6,"",",F15.6)') t, T_out(t), Int_Stemp, Ext_Stemp, &
    h_in*(Int_Stemp-T_in), -h_out*(Ext_Stemp-T_out(t)), &
    ZoneArea*h_in*(Int_Stemp-T_in)
END DO

CLOSE(response)

END SUBROUTINE HomogeneousStep

!-----

FUNCTION f2(X, Param)

! PURPOSE:      Defines transcendental function used in analytical
!               solution to adiabatic wall test case with step in
!               air temperature as driving function
!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)

```

```

! function type definition
REAL(prec) :: f2

! arguments
REAL(prec), INTENT(IN) :: X ! trial value of root
REAL(prec), DIMENSION(4), INTENT(IN) :: Param ! array of input parameters

!local variables
REAL(prec) :: H_in, H_out ! convection coefficients
REAL(prec) :: K, L ! conductivity and thickness
REAL(prec) :: H1, H2 ! parameter combinations

! assign paramters to local variables (see Ozisik)
H1 = Param(4)*Param(2)/Param(3)
H2 = Param(4)*Param(1)/Param(3)

! the transcendental function
f2 = DTAN(X) - X*(H1+H2)/(X*X - H1*H2)

END FUNCTION f2

!-----
FUNCTION Norm(Beta, H_in, H_out, K, L)

! PURPOSE: Defines normalization function used in analytical
! solution to homogeneous step test case with step
! in air temperature as driving function
!
! CREATED:
! 8.13.99 S.J.Rees
!
! MODIFIED:
!

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)
! function type definition
REAL(prec) :: Norm

! arguments
REAL(prec), INTENT(IN) :: Beta ! root of transcendental eqn.
REAL(prec), INTENT(IN) :: H_in, H_out ! convection coefficients
REAL(prec), INTENT(IN) :: K, L ! conductivity and thickness

!local variables
REAL(prec) :: H1, H2 ! parameter combinations

! assign parameter combinations (see Ozisik)
H1 = L*H_out/K
H2 = L*H_in/K

! the normalization function
Norm = 0.5_8*((Beta*Beta + H1*H1)*(1.0_8+H2/(Beta*Beta + H2*H2)) + H1)

END FUNCTION Norm

```

```
!-----  
END MODULE HomoStep
```

Source Code: Module Periodic

MODULE Periodic

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to conduction test case.
!               In this case a sinusoidal external air temperature
!               with internal air temperature held constant.
!               Three material layers are included with constant
!               convection coefficients inside and out.
!
! CREATED:
!   5.27.99      S.J.Rees
!
! MODIFIED:
!
!USE ConductUtil

PRIVATE
! make public main subroutine
PUBLIC :: PeriodicCond

CONTAINS

!-----

SUBROUTINE PeriodicCond(FileName, K, Rho, C_p, h_in, h_out, Thickness, &
                        N_layers, T_mean, T_amp, Period, Area, TimeStep, &
                        Ext_Temp, N_step, ErrorFlag, ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "PeriodicCond" :: PeriodicCond

! PURPOSE:      Produces weather files and analytical response for the
!               case of an sinusoidal external air temperature with
!               internal air temperature held constant. Three material
!               layers are included with constant convection coefficients
!               inside and out.
!
!

```



```

! CREATED:
!   5.27.99      S.J.Rees
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
!
!   FileName.dat      -   Analytical response data
!   FileName.wy2      -   Weather file in WYEC2 format
!   FileName.tmy2     -   Weather file in TMY2 format
!
!
! SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   K                  -   slab conductivity
!   Rho                -   slab density
!   C_p               -   slab specific heat capacity
!   h_in              -   internal convection coefficient
!   h_out              -   external convection coefficient
!   Thickness          -   slab thickness
!   N_layers           -   No of layers in wall
!   T_mean             -   Initial/final external dry bulb temperature
!   T_amp              -   amplitude of external dry bulb temperature
!   Period             -   Period of driving function ( hours)
!   Area               -   Area of external wall
!   TimeStep           -   Time step is variable to deal with
!                       -   IP vs SI units. normally 1.0 for IP units,
!                       -   3600.0 for SI units for hourly time steps.
!   Ext_Temp           -   External Air temperature

IMPLICIT NONE
! parameters
INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)
! data type parameter
INTEGER, PARAMETER :: response=27 ! unit number for output file
INTEGER, PARAMETER :: N_Hours=8760 ! No. of hours in year
LOGICAL :: Debug = .true. ! flag for debug output

! arguments
!CHARACTER(LEN=40), INTENT(IN) :: FileName
CHARACTER *(*), INTENT(in) :: FileName
INTEGER, INTENT(IN) :: N_step, N_layers
REAL(prec), DIMENSION(N_layers), INTENT(IN) :: K, Rho, C_p, Thickness
REAL(prec), INTENT(IN) :: h_in, h_out, Period, Area
REAL(prec), INTENT(IN) :: T_mean, T_amp, TimeStep
REAL(4), DIMENSION(N_step), INTENT(OUT) :: Ext_temp

CHARACTER*(*), INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out) :: ErrorFlag ! two byte int for VB integer

!local variables
INTEGER :: i, n, error ! counter and error variables
!INTEGER :: N_layers ! No of layers in wall

```

```

INTEGER :: t                                ! time counter
REAL(prec) :: Pi, omega, P                  ! Period variables
REAL(prec) :: time                          ! real value of time
REAL(prec) :: U_Value                       ! overall U value/conductance
REAL(prec) :: Dec_Factor                    ! decrement factor
REAL(prec) :: PhaseLag ! phase lag associated with dec. factor
REAL(prec), DIMENSION(N_Hours) :: Int_Load ! Internal Load
CHARACTER(LEN=36) :: dummy                  ! dummy string variable
! complex variables
COMPLEX(prec), DIMENSION(2,2) :: R_in       ! internal resistance matrix
COMPLEX(prec), DIMENSION(2,2) :: R_out      ! external resistance matrix
COMPLEX(prec), DIMENSION(2,2) :: M_tot      ! overall impedance matrix
COMPLEX(prec), DIMENSION(2,2) :: M_layer    ! fabric layer impedance matrix
COMPLEX(prec) :: Complex_coshP              ! complex value of cosh(P+iP)
COMPLEX(prec) :: Complex_sinhP              ! complex value of sinh(P+iP)
COMPLEX(prec) :: Complex_P                  ! complex value (P+iP)
COMPLEX(prec) :: Complex_F                  ! complex value of decrement factor

! get no. of layers from size of argument arrays
!N_layers = SIZE(K)
Pi = DASIN(1.0_8)*2.0

!open file for output of response
! process filename
! n = LEN_TRIM(FileName)
! dummy=FileName
! dummy(n+1:) = ".dat"
!attempt open
! OPEN (response, file=TRIM(dummy), status='unknown', IOSTAT=error)
! IF(error/=0) THEN
!     PRINT *, "Error: analytical response output file could not be opened"
!     STOP
! END IF

!attempt to open file - full path is passed as FileName by VB application
OPEN (response, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! set up inner and outer resistance matrices
R_in(1,1) = CMPLX(1.0,0.0,Prec)
R_in(1,2) = CMPLX(1.0/h_in,0.0,Prec)
R_in(2,1) = CMPLX(0.0,0.0,Prec)
R_in(2,2) = CMPLX(1.0,0.0,Prec)

R_out(1,1) = CMPLX(1.0,0.0,Prec)
R_out(1,2) = CMPLX(1.0/h_out,0.0,Prec)
R_out(2,1) = CMPLX(0.0,0.0,Prec)
R_out(2,2) = CMPLX(1.0,0.0,Prec)

! set M_tot to R_out to start
M_tot = R_out

```

```

! loop over fabric layers
DO i= N_layers, 1, -1
  !omega = Period*Pi/(24*24*3600)
  !omega = Pi/(Period*3600)
  omega = Pi/(Period*TimeStep)
  P = Thickness(i)*SQRT(omega*Rho(i)*C_p(i)/K(i))
  ! work out cosh(P+iP) and sinh(P+iP)
  Complex_P = CMPLX(P,P)
  Complex_coshP = 0.5*(CDEXP(Complex_P)+CDEXP(-Complex_P))
  Complex_sinhP = 0.5*(CDEXP(Complex_P)-CDEXP(-Complex_P))
  ! assemble matrix for this layer
  M_layer(1,1) = Complex_coshP
  M_layer(1,2) = Thickness(i)*Complex_sinhP/(K(i)*Complex_P)
  M_layer(2,1) = K(i)*Complex_P*Complex_sinhP/Thickness(i)
  M_layer(2,2) = M_layer(1,1)
  ! multiply matrices
  M_tot = MATMUL(M_layer,M_tot)
END DO

! Multiply by R_in
M_tot = MATMUL(R_in,M_tot)

! work out U-value
U_Value = 1.0/h_in + 1.0/h_out
DO i=1, N_layers
  U_Value = U_Value + Thickness(i)/K(i)
END DO
U_Value = 1.0/U_Value

! work out Decrement factor
Complex_F = 1.0/(U_Value*M_tot(1,2))
Dec_Factor= CDABS(Complex_F)
! lag on Decrement factor
PhaseLag = DATAN2(AIMAG(Complex_F), REAL(Complex_F))
! lag must be in the range -Pi to zero
IF(PhaseLag > 0.0)THEN
  PhaseLag = PhaseLag - 2.0*Pi
END IF
!PhaseLag = -12.0*PhaseLag*24.0/(Period*Pi)
PhaseLag = -PhaseLag*Period/(2.0*Pi)

! some debug output
!IF(Debug)THEN
!  WRITE(*, '( "Decrement Factor = ",F20.12)') Dec_Factor
!  WRITE(*, '( "Phase lag          = ",F20.12)') PhaseLag
!  WRITE(*, '( "U-Value          = ",F20.12)') U_Value
!END IF

! Start loop over each hour of the year to calc response
DO t=1, N_hours
  time = REAL(t)
  Ext_temp(t) = T_mean + T_amp*DSIN(2.0*time*Pi/Period)
  !Int_Load(t) = U_Value*Area*Dec_Factor* &
  !DSIN((2.0*time*Pi/Period)-PhaseLag)
  Int_Load(t) = U_Value*Area*Dec_Factor*T_amp* &
    DSIN(2.0*(time-PhaseLag)*Pi/Period)

```

```

        ! IF(Debug)THEN
        !   WRITE(*, '(1X,I6,2F20.12)') t, Ext_temp(t), Int_Load(t)
        !END IF
    END DO

    ! write header lines
    WRITE(response, '( "# ASHRAE Analytical Test Suite - Results file")')
    WRITE(response, '( "# Periodic conduction test")')
    WRITE(response, '( "#")')
    WRITE(response, '( "# Test Parameters -")')
    WRITE(response, '( "# Inside convection correlation coefficient =", &
        " ", F15.4)') h_in
    WRITE(response, '( "# Outside convection correlation coefficient =", &
        " ", F15.4)') h_out

    WRITE(response, '( "# Mean outside air temperature =", " ", &
        F15.4)') T_mean
    WRITE(response, '( "# Amplitude of outside air temperature =", &
        " ", F15.4)') T_amp

    DO i=1, N_layers
        WRITE(response, '( "# Fabric layer", I2, " conductivity =", " ", &
            F15.4)') i, K(i)
        WRITE(response, '( "# Fabric layer", I2, " density =", " ", &
            F15.4)') i, Rho(i)
        WRITE(response, '( "# Fabric layer", I2, " specific heat =", " ", &
            F15.4)') i, C_p(i)
        WRITE(response, '( "# Fabric layer", I2, " thickness =", " ", &
            F15.4)') i, Thickness(i)
    END DO

    WRITE(response, '( "#", "/", "# Test results -", "/", "#")')
    WRITE(response, '( "#", "/", "# Hour", " ", " ", "Outside Drybulb Temp.", &
        " ", " ", "Zone Load")')

    DO t=1, N_hours
        ! write out response
        WRITE(response, '(1X,I6," ", " ", F20.12, " ", " ", F20.12)') t, &
            Ext_temp(t), Int_Load(t)
    END DO

    CLOSE(response)

END SUBROUTINE PeriodicCond

END MODULE Periodic

```

Source Code: Module ExtSolRadiation

MODULE ExtSolRadiation

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to the exterior solar radiation
!               on opaque surfaces' test case.
!
! CREATED:      11.7.99      Dongyi Xiao
!
! MODIFIED:
!
PRIVATE
! make public main subroutine
PUBLIC :: ExtSolRad

CONTAINS

!-----

SUBROUTINE ExtSolRad(SolIrrad, Num_hour, Sol_Absorp, Sur_Azimuth, &
                    Sur_Tilt, File_in, FileName, Cond, N_Cond, Thickness, &
                    N_thickness, T_outside, T_inside, Hin_A, Hin_C, Hin_n, &
                    Hout_A, Hout_C, Hout_n, Sur_area, Coeff_A, Coeff_B, &
                    Time_interval, ErrorFlag, ErrorMessage, DirName, &
                    Length_DirName)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "ExtSolRad" :: ExtSolRad

! PURPOSE:      Produces analytical response for the case of exterior
!               solar radiation on opaque surfaces using the method of
!               heat balance. The convection correlations inside and
!               outside are specified as  $H = A + C \cdot (T_a - T_s)^n$ .
!               The steady state heat flux is found using a successive
!               substitution method to solve the non-linear equation.
!               A variable number of wall layers can be specified.

```

```

!
! CREATED:
!   11.7.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!
!   FileName.dat      -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   SolIrrad          -   Array of beam Irradiance in normal,
!                           for each hour
!   Num_hour           -   number of hours in one day
!   Sol_Absorp         -   The solar absorption of the surface
!   Sur_Azimuth        -   The surface azimuth
!   Sur_Tilt           -   The surface tilt angle
!   File_in            -   the index of solar position file
!   FileName           -   Root File Name for weather and data files
!   Cond               -   Array of wall conductivities
!   N_Conc             -   Number of conductivities in array
!   Thickness          -   Array of wall Thicknesses
!   N_Thickness        -   Number of Thicknesses in array
!   T_outside          -   Outside air temperature
!   T_inside           -   Inside air temperature
!   Hin_A              -   Conv. correlation coefficient A for inside
!   Hin_C              -   Conv. correlation coefficient C for inside
!   Hin_n              -   Conv. correlation coefficient n for inside
!   Hout_A             -   Conv. correlation coefficient A for outside
!   Hout_C             -   Conv. correlation coefficient C for outside
!   Hout_n             -   Conv. correlation coefficient n for outside
!   Sur_area           -   area of the opaque surface
!   Coeff_A            -   apparent solar irradiation
!                           at air mass equal zero
!   Coeff_B            -   atmospheric extinction coefficient
!   Time_interval      -   the time interval of the solar position data
!                           obtained from the US Naval Observatory
!                           (10 min)
!   ErrorFlag          -   for error checking
!   ErrorMessage       -   for error reporting
!   DirName            -   the directory of the VB interface executive
!                           file resides in.
!   Length_DirName     -   the length of the DirName character
IMPLICIT NONE

!parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: input=26 ! unit number for solar
                                !position input file
INTEGER, PARAMETER :: Results=27 ! unit number for
                                !analytical response output file
REAL(prec2), PARAMETER :: Pai=3.1415926

```

```

!arguments
REAL(prec2)           :: Sur_Azimuth
REAL(prec2)           :: Sur_Tilt
REAL(prec2)           :: Sol_Absorp, Sur_area
INTEGER, INTENT(in)    :: N_Conc, N_thickness, Num_hour
INTEGER, INTENT(in)    :: File_in
CHARACTER *(*) , INTENT(in) :: FileName

INTEGER, INTENT(in)    :: Length_DirName
CHARACTER (LEN=Length_DirName), INTENT(in) :: DirName

REAL(prec2), DIMENSION(N_Conc), INTENT(in) :: Cond
REAL(prec2), DIMENSION(N_thickness), INTENT(in) :: Thickness
REAL(prec2), INTENT(in) :: T_outside, T_inside
REAL(prec2), INTENT(in) :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in) :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(in) :: Coeff_A
REAL(prec2), INTENT(in) :: Coeff_B
REAL(prec2), INTENT(in) :: Time_interval

REAL(prec1), DIMENSION(Num_hour), INTENT(out) :: SolIrrad

CHARACTER*(*) , INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out) :: ErrorFlag ! two byte int for VB integer

!local variables
INTEGER :: i,n,error
INTEGER :: Num_Sol ! The number of data lines in the input file
INTEGER, DIMENSION(150) :: N_hour, M_minute
! Array of hour, minute element of the time
REAL(prec2), DIMENSION(150) :: BeamIrradS
! Beam irradiation on the receiving slope
REAL(prec2), DIMENSION(150) :: GeoFactor !Geometric factor used in
! calculating the Beam irradiation
REAL(prec2), DIMENSION(150) :: Cos_Theta
! Array of the cosine of the incident angle
REAL(prec2), DIMENSION(150) :: Altitude ! Array of solar altitude
REAL(prec2), DIMENSION(150) :: Sol_Zenith ! Array of solar zenith
REAL(prec2), DIMENSION(150) :: Sol_Azimuth ! Array of solar azimuth
REAL(prec2), DIMENSION(150) :: Flux_inside
! Array of calculated internal conduction heat flux
REAL(prec2), DIMENSION(150) :: Flux_outside
! Array of calculated external conduction heat flux
REAL(prec2), DIMENSION(150) :: Flux_fabric
! Array of calculated conduction heat flux through fabric
REAL(prec2), DIMENSION(150) :: BeamIrradN
! Array of beam Irradiance in normal
REAL(prec2), DIMENSION(150) :: Flux
! Array of calculated steady state heat flux
REAL(prec2), DIMENSION(150) :: Ts_in, Ts_out
! Array of calculated inside & outside surface temperature
REAL(prec2), DIMENSION(150) :: H_in, H_out
! Array of Calculated inside & outside conv. coefficient
REAL(prec2), DIMENSION(150) :: Load ! Array of the zone load
REAL(prec2) :: Peak_load ! The peak zone load in one day

CHARACTER(LEN=360) :: dummy ! dummy string variable

```

```

n=Length_DirName
dummy(1:n)=DirName(1:n)

! open the input file
IF(file_in==1) THEN
    dummy(n+1:)="\Atlanta_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==2) THEN
    dummy(n+1:) = "\Atlanta_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==3) THEN
    dummy(n+1:) = "\Boston_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==4) THEN
    dummy(n+1:) = "\Boston_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==5) THEN
    dummy(n+1:) = "\Chicago_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==6) THEN
    dummy(n+1:) = "\Chicago_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==7) THEN
    dummy(n+1:) = "\Los Angeles_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==8) THEN
    dummy(n+1:) = "\Los Angeles_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
END IF
IF(error/=0) THEN
    ErrorMessage="input file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! read the solar position data from the input file
READ(input,*) Num_sol
DO i=1, Num_Sol
    READ(input, *) N_hour(i), M_minute(i), Altitude(i), Sol_Azimuth(i)
END DO
CLOSE(input)

! Transfer the unit from degree to radian and
! calculate the solar zenith angle
DO i=1, Num_Sol
    Altitude(i)=Pai*Altitude(i)/180.
    Sol_Zenith(i)=Pai/2.-Altitude(i)
    Sol_Azimuth(i)=Pai*Sol_Azimuth(i)/180.
END DO

! Transfer the unit of surface tilt angle and surface
! azimuth from degree to radian
Sur_Azimuth=Sur_Azimuth*Pai/180
Sur_Tilt=Sur_Tilt*Pai/180

! calculate the cosine of the incident angle of the surface

```



```

CALL Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, &
                 Sol_Azimuth, Sur_Azimuth, Sur_Tilt)

! calculate the Beam Irradiance in normal
CALL SolIrrad_ND(file_in, Num_Sol, Altitude, N_hour, &
                 M_minute, BeamIrradN, SolIrrad, &
                 Coeff_A, Coeff_B, Time_interval)

! calculate analytical response
CALL Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN, Sol_Absorp, &
                 Cond, N_Cond, Thickness, N_thickness, Sur_area, &
                 T_outside, T_inside, Hin_A, Hin_C, Hin_n, Hout_A, &
                 Hout_C, Hout_n, Flux, Flux_inside, Flux_outside, &
                 Flux_fabric, Ts_in, Ts_out, H_in, H_out, ErrorFlag, &
                 ErrorMessage, BeamIrradS, GeoFactor, Load, Peak_load)

! Transfer the units of Sur_Azimuth, Sur_Tilt, Altitude,
! Sol_Zenith and Sol_Azimuth from radiance to degree for output
DO i=1, Num_Sol
    Altitude(i)=180.*Altitude(i)/Pai
    Sol_Zenith(i)=180.*Sol_Zenith(i)/Pai
    Sol_Azimuth(i)=180.*Sol_Azimuth(i)/Pai
END DO
Sur_Azimuth=Sur_Azimuth*180/Pai
Sur_Tilt=Sur_Tilt*180/Pai

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )')
WRITE(Results, '( "# Exterior Solar Radiation test" )')
WRITE(Results, '( //"Test Parameters -" )')

WRITE(Results, '( "Inside convection correlation coefficient A =", &
                 ", ", F15.4 )') Hin_A
WRITE(Results, '( "Inside convection correlation coefficient C =", &
                 ", ", F15.4 )') Hin_C
WRITE(Results, '( "Inside convection correlation coefficient n =", &
                 ", ", F15.4 )') Hin_n
WRITE(Results, '( "Outside convection correlation coefficient A =", &
                 ", ", F15.4 )') Hout_A
WRITE(Results, '( "Outside convection correlation coefficient C =", &
                 ", ", F15.4 )') Hout_C
WRITE(Results, '( "Outside convection correlation coefficient n =", &
                 ", ", F15.4 )') Hout_n

WRITE(Results, '( "Outside air temperature =", ", ", F15.4 )') T_outside
WRITE(Results, '( "Inside air temperature =", ", ", F15.4 )') T_inside

DO i=1, N_Cond
    WRITE(Results, '( "Fabric layer", I2, " conductivity =", &

```

```

        ",",F15.4)) i, Cond(i)
    WRITE(Results,'("Fabric layer",I2," thickness =", &
        ",",F15.4)) i, Thickness(i)
END DO
WRITE(Results,'("Surface Azimuth =",",",F15.4)) Sur_Azimuth
WRITE(Results,'("Surface Tilt angle =",",",F15.4)) Sur_Tilt
WRITE(Results,'("Solar Absorption of the surface =", &
    ",",F15.4)) Sol_Absorp
WRITE(Results,'("Surface Area =",",",F15.4)) Sur_area

WRITE(Results,'(// "Test results -")')
WRITE(Results,'(// "Peak Load =",",",F15.4)) Peak_load

WRITE(Results,'(// "Time",",", "Internal Surface Temp.",",", &
    "External surface Temp.",",", "Inside Convection Coeff.",",", &
    "Outside Convection Coeff.",",", "Incident Solar Flux", &
    ",", "Internal Surface Heat Flux", ",", &
    "External Surface Heat Flux",",", "Zone Load"))')

DO i=1, Num_Sol
    WRITE(Results,'(I2.2,":",I2.2, ",",F15.4, ",",F15.4, ",", &
        F15.4, ",",F15.4,"",F15.4, ",",F15.4, ",",F15.4, ",",F15.4, ",",F15.4)) &
        N_hour(i), M_minute(i),Ts_in(i), &
        Ts_out(i), H_in(i), H_out(i),BeamIrradS(i),&
        Flux_inside(i), Flux_outside(i), Load(i)
END DO

CLOSE(Results)

END SUBROUTINE ExtSolRad

!-----
SUBROUTINE Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, &
    Sur_Azimuth, Sur_Tilt)

! PURPOSE:      Calculate the cosine of the incident angle of the surface
!
! CREATED:
!   11.7.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Cos_Theta    -   Array of the cosine of the incident angle
!   Num_Sol       -   The number of data lines in the input file
!   Sol_Zenith    -   Array of solar zenith
!   Sol_Azimuth   -   Array of solar azimuth
!   Sur_Azimuth   -   The surface azimuth
!   Sur_Tilt      -   The surface tilt angle

```

```

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol) :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol) :: Sol_Zenith
REAL(prec2), DIMENSION(Num_Sol) :: Sol_Azimuth
REAL(prec2), INTENT(in)          :: Sur_Azimuth
REAL(prec2), INTENT(in)          :: Sur_Tilt

! local variables
REAL(prec2) :: SunDirCosA, SunDirCosB, SunDirCosC
              ! solar direction cosines
REAL(prec2) :: SurDirCosA, SurDirCosB, SurDirCosC
              ! surface direction cosines
INTEGER      :: i

DO i=1, Num_Sol

    ! calculate the solar direction cosines
    SunDirCosA=SIN(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosB=COS(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosC=COS(Sol_Zenith(i))

    ! calculate the surface direction cosines
    SurDirCosA=SIN(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosB=COS(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosC=COS(Sur_Tilt)

    ! calculate the cosine of incidence angle Theta
    Cos_Theta(i)=SunDirCosA*SurDirCosA+SunDirCosB*SurDirCosB &
                  +SunDirCosC*SurDirCosC

END DO

END SUBROUTINE Incid_Angle

!-----

SUBROUTINE SolIrrad_ND(File_in, Num_Sol, Altitude, N_hour, M_minute, &
                      BeamIrradN, SolIrrad, Coeff_A, Coeff_B, Time_interval)

! PURPOSE:      Calculate the Beam Irradiance in normal
!
! CREATED:
!   11.7.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:

```

```

!   n/a
!
! SUBROUTINE ARGUMENTS:
!   File_in           -   the index of solar position file
!   Num_Sol           -   The number of data lines in the input file
!   Altitude          -   Array of solar altitude
!   N_hour            -   Array of hour element of the time
!   M_minute          -   Array of minute element of the time
!   BeamIrradN        -   Array of Beam Irradiance in normal,
!                           for time intervals like 10 minutes
!   SolIrrad          -   Array of beam Irradiance in normal,
!                           for each hour
!   Coeff_A            -   apparent solar irradiation at air
!                           mass equal zero
!   Coeff_B            -   atmospheric extinction coefficient
!   Time_interval      -   the time interval of the solar position data
!                           obtained from the US Naval Observatory
!                           (10 min)
!
IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol, File_in
REAL(prec2), DIMENSION(Num_Sol) :: Altitude
REAL(prec2), DIMENSION(Num_Sol) :: BeamIrradN
INTEGER, DIMENSION(Num_Sol)      :: N_hour, M_minute
REAL(prec1), DIMENSION(24)       :: SolIrrad
REAL(prec2), INTENT(in)          :: Coeff_A
REAL(prec2), INTENT(in)          :: Coeff_B
REAL(prec2), INTENT(in)          :: Time_interval

! local variables
INTEGER                :: i, j

! initialize the array of Beam Irradiance in normal
DO i=1,24
    SolIrrad(i)=0.0
END DO

DO i=1, Num_Sol

    ! calculate the Beam Irradiation in normal
    ! (W/m2 for SI units or BTTU/(h.ft2) for IP units)
    BeamIrradN(i)=Coeff_A/EXP(Coeff_B/SIN(Altitude(i)))

    ! calculate the Beam Irradiance in normal in one hour
    ! (J/m2 for SI units or BTU/ft2 for IP units)
    ! (will be used for weather data)

    SolIrrad(N_hour(i)+1)=SolIrrad(N_hour(i)+1)+BeamIrradN(i)*Time_interval

END DO

```

```
END SUBROUTINE SolIrrad_ND
```

```
!-----
```

```
SUBROUTINE Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN, &
    Sol_Absorp, Cond, N_Conc, Thickness, N_thickness, &
    Sur_area, T_outside, T_inside, Hin_A, Hin_C, Hin_n, &
    Hout_A, Hout_C, Hout_n, Flux, Flux_inside, Flux_outside, &
    Flux_fabric, Ts_in, Ts_out, H_in, H_out, ErrorFlag, &
    ErrorMessage, BeamIrradS, GeoFactor, Load, Peak_load)
```

```
! PURPOSE:      Calculate analytical response for the case of exterior
!               solar radiation on opaque surfaces using the method of
!               heat balance. The convection correlations inside and
!               outside are specified as  $H = A + C \cdot (T_a - T_s)^n$ . The steady
!               state heat flux is found using a successive substitution
!               method to solve the non-linear equation.
!               A variable number of wall layers can be specified.
```

```
! CREATED:
!   11.7.99      Dongyi Xiao
```

```
! MODIFIED:
```

```
! INPUT FILES:
!   n/a
```

```
! OUTPUT FILES:
!   n/a
```

```
! SUBROUTINE ARGUMENTS:
```

```
!   Cos_Theta      -   Array of the cosine of the incident angle
!   Num_Sol         -   The number of data lines in the input file
!   Altitude        -   Array of solar altitude
!   BeamIrradN      -   Array of beam Irradiance in normal
!   Sol_Absorp      -   The solar absorption of the surface
!   Cond            -   Array of wall conductivities
!   N_Conc          -   Number of conductivities in array
!   Thickness       -   Array of wall Thicknesses
!   N_Thickness     -   Number of Thicknesses in array
!   Sur_area        -   The surface area
!   T_outside       -   Outside air temperature
!   T_inside        -   Inside air temperature
!   Hin_A           -   Conv. correlation coefficient A for inside
!   Hin_C           -   Conv. correlation coefficient C for inside
!   Hin_n           -   Conv. correlation coefficient n for inside
!   Hout_A          -   Conv. correlation coefficient A for outside
!   Hout_C          -   Conv. correlation coefficient C for outside
!   Hout_n          -   Conv. correlation coefficient n for outside
!   Flux            -   Array of calculated steady state heat flux
!   Flux_inside     -   Array of calculated internal conduction
!                       heat flux
!   Flux_outside    -   Array of calculated external conduction
!                       heat flux
!   Flux_fabric     -   Array of calculated conduction heat flux
!                       through the fabric
!   Ts_in           -   Array of Calculated inside surface
```

```

!           temperature
!   Ts_out      -   Array of Calculated outside surface
!           temperature
!   H_in        -   Array of Calculated inside conv. coefficient
!   H_out       -   Array of Calculated outside conv. coefficient
!   ErrorFlag   -   for error checking
!   ErrorMessage -   for error reporting
!   BeamIrradS  -   Array of beam irradiation on the receiving
!                   slope
!   GeoFactor   -   geometric factor used in calculating the Beam
!                   irradiation
!   Load       -   Array of the zone load
!   Peak_load   -   The peak zone load in one day

```

```

IMPLICIT NONE

```

```

! parameters

```

```

INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 500
REAL(prec2), PARAMETER :: Tolerance = 1.0E-06 ! tolerance
REAL(prec2), PARAMETER :: Delta_H = 1.0E-20
! a small number added to the initial value of convection
! coefficient to avoid zero convection coefficients

```

```

! arguments

```

```

INTEGER                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol)       :: BeamIrradN
REAL(prec2), INTENT(in)                :: Sol_Absorp
REAL(prec2), DIMENSION(Num_sol)       :: Altitude
REAL(prec2), DIMENSION(Num_Sol)       :: Cos_Theta
INTEGER, INTENT(in)                   :: N_Cond, N_thickness
REAL(prec2), DIMENSION(N_Cond), INTENT(in) :: Cond
REAL(prec2), DIMENSION(N_thickness), INTENT(in) :: Thickness
REAL(prec2), INTENT(in)               :: T_outside, T_inside
REAL(prec2), INTENT(in)               :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in)               :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(in)               :: Sur_area
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: BeamIrradS
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: GeoFactor
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_inside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_outside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_fabric

REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Ts_in, Ts_out
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: H_in, H_out
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Load
REAL(prec2), INTENT(out)                   :: Peak_load
CHARACTER*(*) , INTENT(out)                :: ErrorMessage
INTEGER(2), INTENT(out)                    :: ErrorFlag ! two byte int for VB integer

```

```

! local variables

```

```

REAL(prec2)                                :: Fabric_R
! Slab thermal resistance (surface to surface)

```

```

REAL(prec2),DIMENSION(Num_sol)    :: MinFlux, MaxFlux
                                   ! Min and max calculated flux values
INTEGER                           :: i, j,n, error
CHARACTER(LEN=132)                :: dummy
INTEGER, PARAMETER                :: DebugUnit = 20
LOGICAL, PARAMETER                :: Debug = .false.
LOGICAL, PARAMETER                :: Warning = .false.

! loop to calculate the geometric factor and
! the Beam irradiation on the receiving slope
DO j=1, Num_sol

    ! calculate the geometric factor used
    ! in calculating the Beam irradiation
    IF(SIN(Altitude(j))==0.0)THEN
        WRITE(ErrorMessage,'("The sine of the altitude(", I2, &
                                " )appears to be zero")) j
        ErrorFlag = 1
        RETURN
    END IF
    GeoFactor(j)=MAX(Cos_Theta(j)/SIN(Altitude(j)), 0.0)

    ! calculate the Beam irradiation on the receiving slope
    IF (Altitude(j)<0) THEN
        BeamIrradS(j)=0
    ELSE
        BeamIrradS(j)=GeoFactor(j)*BeamIrradN(j)*SIN(Altitude(j))
    END IF
END DO

! start loop to calculate the heat flux and internal &
! external surface temperature
DO j=1, Num_sol
    ! do some initial error checking
    IF(N_Thickness /= N_Cond)THEN
        ErrorMessage = "Fabric data is inconsistant"
        ErrorFlag = 1
        RETURN
    END IF

    ! open a unit for debug output
    IF(Debug .and. DebugUnit/=6)THEN
        OPEN (DebugUnit, file="ExtSolRad.log", &
              status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Debug output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF
        ! copy out arguments to debug file
        WRITE(DebugUnit,'("ExtSolRad arguments:")')
        WRITE(DebugUnit,'("Inside convection correlation coefficient A &
                            =",F10.6)') Hin_A
        WRITE(DebugUnit,'("Inside convection correlation coefficient C &
                            =",F10.6)') Hin_C
        WRITE(DebugUnit,'("Inside convection correlation coefficient n &
                            =",F10.6)') Hin_n
    
```

```

WRITE(DebugUnit, ' ("Outside convection correlation coefficient A &
      = ", F10.6) ') Hout_A
WRITE(DebugUnit, ' ("Outside convection correlation coefficient C &
      = ", F10.6) ') Hout_C
WRITE(DebugUnit, ' ("Outside convection correlation coefficient n &
      = ", F10.6) ') Hout_n
WRITE(DebugUnit, ' ("Outside air temperature
      = ", F10.6) ') &
      T_outside
WRITE(DebugUnit, ' ("Inside air temperature
      = ", F10.6) ') &
      T_inside
DO i=1,N_Cond
  WRITE(DebugUnit, ' ("Fabric layer", I2, " conductivity
      = ", &
      F10.6) ') i, Cond(i)
  WRITE(DebugUnit, ' ("Fabric layer", I2, " thickness
      = ", &
      F10.6) ') i, Thickness(i)
END DO
END IF

! set initial guess of surface temperatures
Ts_out(j) = T_outside - 0.33*(T_outside - T_inside)
Ts_in(j) = T_outside - 0.67*(T_outside - T_inside)

! calculate the resistance of the slab
Fabric_R = 0.0
DO i = 1, N_Cond
  IF(Cond(i)==0.0)THEN
    WRITE(ErrorMessage, ' ("Conductivity of layer ", &
      I2, " appears to be zero") ') i
    ErrorFlag = 1
    RETURN
  END IF
  Fabric_R = Fabric_R + Thickness(i)/Cond(i)
END DO

H_in(j) = Hin_A + Hin_C*ABS(T_inside - Ts_in(j))*Hin_n + Delta_H
H_out(j) = Hout_A + Hout_C*ABS(T_outside - Ts_out(j))*Hout_n &
      + Delta_H

! start loop to find Flux by successive substitution
n = 1
DO
  IF(H_in(j)==0.0 .or. H_out(j)==0.0)THEN
    WRITE(ErrorMessage, ' ("Inside or outside convection &
      coefficient calculated to be zero", " - can not procede") ')
    ErrorFlag = 1
    RETURN
  END IF

  ! flux based on current value of total resistance
  Flux(j)= ((T_outside - T_inside)+BeamIrradS(j)* &
      Sol_Absorp/H_out(j))/(1/H_in(j)+ &
      1/H_out(j)+Fabric_R)
  ! update surface temps based on flux
  Ts_out(j) = T_outside - (Flux(j)-BeamIrradS(j)* &
      Sol_Absorp)/H_out(j)

```



```

Ts_in(j) = T_inside + Flux(j)/H_in(j)
! re-calculate conv. coef using current temps
H_in(j) = Hin_A + Hin_C*(ABS(T_inside - Ts_in(j))*Hin_n)
H_out(j) = Hout_A + Hout_C*(ABS(T_outside - Ts_out(j))*Hout_n)

! calculate fluxes over different resistances for checking
Flux_inside(j) = H_in(j)*(Ts_in(j) - T_inside)
Flux_outside(j) = BeamIrradS(j)*Sol_Absorp+ H_out(j)* &
    (T_outside - Ts_out(j))
Flux_fabric(j)= (Ts_out(j) - Ts_in(j)) / Fabric_R

! some debug output
IF(Debug)THEN
    WRITE(DebugUnit,'("H_in = ",F8.5," H_out = ",F8.5, &
        " Ts_in = ",F8.5," Ts_out = ", F8.5)') H_in(j), &
        H_out(j), Ts_in(j), Ts_out(j)
    WRITE(DebugUnit,'("Q = ",F8.5," Qout = ",F8.5, &
        " Qin = ",F8.5," Qfab = ",F8.5)') Flux(j), &
        Flux_outside(j), Flux_inside(j), Flux_fabric(j)
END IF

! calc min and max fluxes
MinFlux(j) = MIN(Flux(j), Flux_inside(j), &
    Flux_outside(j), Flux_fabric(j))
MaxFlux(j) = MAX(Flux(j), Flux_inside(j), &
    Flux_outside(j), Flux_fabric(j))
! test for exit condition
IF(ABS(MaxFlux(j)-MinFlux(j)) < Tolerance .or. &
    n > Max_Iterations) EXIT
n = n + 1
END DO

! set warning if max iterations exceded
IF(n > Max_Iterations)THEN
    ErrorMessage = "The steady-state heat flux could not &
        be calculated within the allowable number of iterations"
    ErrorFlag = -1
    RETURN
END IF

IF(Debug)THEN
    CLOSE(DebugUnit)
END IF

! calculate the zone load
Load(j)=Sur_area*Flux(j)

! find out the peak zone load in one day
IF (j==2) THEN
    Peak_load=MAX(Load(j),Load(j-1))
END IF
IF (j>2)THEN
    Peak_load=MAX(Load(j),Peak_load)
END IF
END DO

END SUBROUTINE Heat_balance

```

```
END MODULE ExtSolRadiation
```

Source Code: Module ExtSolRadiationGs

MODULE ExtSolRadiationGs

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to the exterior solar radiation on
!               glazed surface and the window shading test cases.
!
! CREATED:      11.19.99      Dongyi Xiao
!
! MODIFIED:
!
PRIVATE

! make public main subroutine
PUBLIC :: ExtSolRad_Gs

CONTAINS

!-----

SUBROUTINE  ExtSolRad_Gs(SolIrrad, Num_hour, Sur_Azimuth, Sur_Tilt, &
                        FileName, File_in, Thickness, Extinc_Coeff, RefIndex_Sur, &
                        T_outside, T_inside, Hin_A, Hin_C, Hin_n, Hout_A, &
                        Hout_C, Hout_n, HorFin, VerFin, VerFin_Side, &
                        HangLength_Hor, HangLength_Ver, Win_Length, &
                        Win_height, Sur_area, Coeff_A, Coeff_B, &
                        Time_interval, ErrorFlag, ErrorMessage, &
                        DirName, Length_DirName)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "ExtSolRad_Gs" :: ExtSolRad_Gs

! PURPOSE:      Produces analytical response for the case of exterior
!               solar radiation on glazed surfaces and windowshading
!               using the method of heat balance. The convection

```

```

!           correlations inside and outside are specified as
!            $H = A + C \cdot (T_a - T_s)^n$ . The steady state heat flux is
!           found using a successive substitution method to solve
!           the non-linear equation.
!
!   CREATED:
!     12.10.99      Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!     n/a
!   OUTPUT FILES:
!
!     FileName.csv      -   Analytical response data
!
!   SUBROUTINE ARGUMENTS:
!     SolIrrad          -   Array of Beam Irradiance in normal,
!                           for each hour
!     Num_hour          -   number of hours in one day
!     Sur_Azimuth       -   The surface azimuth
!     Sur_Tilt          -   The surface tilt angle
!     FileName          -   Root File Name for weather and data files
!     File_in           -   the index of solar position file
!     Thickness         -   Thicknesses of the surface
!     Extinc_Coeff      -   Extinction coefficient of the surface
!     RefIndex_Sur      -   Refractive index of the surface
!     T_outside         -   Outside air temperature
!     T_inside         -   Inside air temperature
!     Hin_A             -   Conv. correlation coefficient A for inside
!     Hin_C             -   Conv. correlation coefficient C for inside
!     Hin_n             -   Conv. correlation coefficient n for inside
!     Hout_A            -   Conv. correlation coefficient A for outside
!     Hout_C            -   Conv. correlation coefficient C for outside
!     Hout_n            -   Conv. correlation coefficient n for outside
!     HorFin            -   Logical variable, is true if using
!                           semi-infinite horizontal fin
!     VerFin            -   Logical variable, is true if using
!                           semi-infinite vertical fin
!     VerFin_Side       -   For deciding the vertical fin is on
!                           which side of the window
!                           1--right side; 2--left side
!     HangLength_Hor    -   Hanging length of the horizontal fin
!     HangLength_Ver    -   Hanging length of the vertical fin
!     Win_Length        -   Window length
!     Win_height        -   Window height
!     Sur_area          -   The surface area
!     Coeff_A           -   apparent solar irradiation at air
!                           mass equal zero
!     Coeff_B           -   atmospheric extinction coefficient
!     Time_interval     -   the time interval of the solar position data
!                           obtained from the US Naval Observatory
!                           (10 min)
!     ErrorFlag         -   for error checking
!     ErrorMessage      -   for error reporting
!     DirName           -   the directory of the VB interface executive
!                           file resides in.
!

```

```

!   Length_DirName       -   the length of the DirName character

IMPLICIT NONE

!parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: input=26 ! unit number for input file
INTEGER, PARAMETER :: Results=27
                        ! unit number for analytical response output file
REAL(prec2), PARAMETER :: Pai=3.1415926
REAL(prec2), PARAMETER :: RefIndex_Air=1.0

!arguments
REAL(prec2)                :: Sur_Azimuth
REAL(prec2)                :: Sur_Tilt
REAL(prec2), INTENT(in)    :: Extinc_Coeff
REAL(prec2), INTENT(in)    :: RefIndex_Sur
REAL(prec2), INTENT(in)    :: Thickness
REAL(prec2), INTENT(in)    :: HangLength_Hor
REAL(prec2), INTENT(in)    :: HangLength_Ver
REAL(prec2), INTENT(in)    :: Win_Length
REAL(prec2), INTENT(in)    :: Win_height
REAL(prec2), INTENT(in)    :: Sur_area
REAL(prec2), INTENT(in)    :: Coeff_A
REAL(prec2), INTENT(in)    :: Coeff_B
REAL(prec2), INTENT(in)    :: Time_interval

INTEGER, INTENT(in)        :: Num_hour
REAL(prec1), DIMENSION(Num_hour), INTENT(out) :: SolIrrad

INTEGER, INTENT(in)        :: Length_DirName
CHARACTER (LEN=Length_DirName), INTENT(in)  :: DirName

CHARACTER *(*), INTENT(in) :: FileName
INTEGER, INTENT(in)        :: VerFin_Side
INTEGER, INTENT(in)        :: File_in

REAL(prec2), INTENT(in)    :: T_outside, T_inside
REAL(prec2), INTENT(in)    :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in)    :: Hout_A, Hout_C, Hout_n

CHARACTER*(*), INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out)    :: ErrorFlag ! two byte int for VB integer

LOGICAL, INTENT(in)        :: HorFin
LOGICAL, INTENT(in)        :: VerFin

!local variables
INTEGER    :: i,n,error
INTEGER    :: Num_Sol ! The number of data lines in the input file

INTEGER, DIMENSION(150) :: N_hour, M_minute
                        ! Array of hour, minute element of the time
REAL(prec2), DIMENSION(150) :: BeamIrradN
                        ! Array of beam Irradiance in normal

```

```

REAL(prec2), DIMENSION(150) :: BeamIrradS
! Beam irradiation on the receiving slope
REAL(prec2), DIMENSION(150) :: GeoFactor
! Geometric factor used in calculating
! the Beam irradiation
REAL(prec2), DIMENSION(150) :: Cos_Theta
! Array of the cosine of the incident angle
REAL(prec2), DIMENSION(150) :: Altitude ! Array of solar altitude
REAL(prec2), DIMENSION(150) :: Sol_Zenith ! Array of solar zenith
REAL(prec2), DIMENSION(150) :: Sol_Azimuth ! Array of solar azimuth
REAL(prec2), DIMENSION(150) :: Flux
! Array of flux transferred
! into the zone by convection
REAL(prec2), DIMENSION(150) :: Flux_inside
! Array of calculated internal
! convective heat flux
REAL(prec2), DIMENSION(150) :: Flux_outside
! Array of calculated external heat flux
! (=sum of absorbed solar flux and outside
! convective flux)
REAL(prec2), DIMENSION(150) :: Flux_Transmit
! Array of solar beam irradiation
! transmitted into the zone
REAL(prec2), DIMENSION(150) :: Absorp
! Array of absorption of the surface
REAL(prec2), DIMENSION(150) :: Transmit
! Array of transmittance of the surface

REAL(prec2), DIMENSION(150) :: Flux_Tol
! Array of Calculated total heat flux
REAL(prec2), DIMENSION(150) :: Ts
! Array of Calculated surface temperature
REAL(prec2), DIMENSION(150) :: H_in, H_out
! Array of Calculated inside &
! outside conv. coefficient
REAL(prec2), DIMENSION(150) :: Load ! Array of the zone load
REAL(prec2), DIMENSION(150) :: LoadShade_HorVer
! Array of zone load using semi-infinite
! horizontal and vertical fin
REAL(prec2), DIMENSION(150) :: LoadShade_Hor
! Array of zone load using semi-infinite
! horizontal fin
REAL(prec2), DIMENSION(150) :: LoadShade_Ver
! Array of zone load using
! semi-infinite vertical fin
REAL(prec2) :: Peak_load
! The peak zone load in one day
REAL(prec2), DIMENSION(150) :: SunlitArea_HorVer
! Sunlit area using semi-infinite horizontal
! and vertical fin
REAL(prec2), DIMENSION(150) :: SunlitArea_Hor
! Sunlit area using semi-infinite horizontal fin
REAL(prec2), DIMENSION(150) :: SunlitArea_Ver
! Sunlit area using semi-infinite vertical fin

CHARACTER(LEN=360) :: dummy ! dummy string variable

```

```

n=Length_DirName
dummy(1:n)=DirName(1:n)

! open the input file
IF(file_in==1) THEN
    dummy(n+1:)= "\Atlanta_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==2) THEN
    dummy(n+1:) = "\Atlanta_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==3) THEN
    dummy(n+1:) = "\Boston_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==4) THEN
    dummy(n+1:) = "\Boston_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==5) THEN
    dummy(n+1:) = "\Chicago_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==6) THEN
    dummy(n+1:) = "\Chicago_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==7) THEN
    dummy(n+1:) = "\Los Angeles_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==8) THEN
    dummy(n+1:) = "\Los Angeles_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
END IF
IF(error/=0) THEN
    ErrorMessage="input file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! read the solar position data from the input file
READ(input,*) Num_sol
DO i=1, Num_Sol
    READ(input, *) N_hour(i), M_minute(i), Altitude(i), Sol_Azimuth(i)
END DO
CLOSE(input)

! Transfer the unit from degree to radian and
! calculate the solar zenith angle
DO i=1, Num_Sol
    Altitude(i)=Pai*Altitude(i)/180.
    Sol_Zenith(i)=Pai/2.-Altitude(i)
    Sol_Azimuth(i)=Pai*Sol_Azimuth(i)/180.
END DO

! Transfer the unit of surface tilt angle and surface
! azimuth from degree to radian
Sur_Azimuth=Sur_Azimuth*Pai/180
Sur_Tilt=Sur_Tilt*Pai/180

! calculate the cosine of the incident angle of the surface

```

```

CALL Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, &
  Sur_Azimuth, Sur_Tilt)

! calculate the Beam Irradiance in normal
CALL SolIrrad_ND(file_in, Num_Sol, Altitude, N_hour, M_minute, &
  BeamIrradN, SolIrrad, Coeff_A, Coeff_B, Time_interval)

! calculate the absorption and transmittance of the surface
CALL Absorp_Transmit(Cos_Theta, Num_Sol, RefIndex_Air, RefIndex_Sur, &
  Extinc_Coeff, Thickness, Absorp, Transmit)

! calc analytical response
CALL Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN, Absorp, &
  Transmit, Sur_area, T_outside, T_inside, Hin_A, Hin_C, Hin_n, &
  Hout_A, Hout_C, Hout_n, Flux, Flux_inside, Flux_outside, &
  Flux_Transmit, Flux_Tol, Ts, H_in, H_out, ErrorFlag, &
  ErrorMessage, BeamIrradS, GeoFactor, Load, Peak_load)

! calculate the zone load under semi-infinite horizontal fin shading
IF (HorFin) THEN

  CALL SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
    Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
    SunlitArea_Hor)
  DO i=1, Num_Sol
    LoadShade_Hor(i)=SunlitArea_Hor(i)*Flux_Tol(i)
  END DO

ELSE

  DO i=1, Num_Sol
    SunlitArea_Hor(i)=0.0
    LoadShade_Hor(i)=0.0
  END DO

END IF

! calculate the zone load under semi-infinite vertical fin shading
IF (VerFin) THEN

  CALL SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
    Cos_Theta, Win_Length, Win_height, HangLength_Ver, &
    VerFin_Side, SunlitArea_Ver)
  DO i=1, Num_Sol
    LoadShade_Ver(i)=SunlitArea_Ver(i)*Flux_Tol(i)
  END DO

ELSE

  DO i=1, Num_Sol
    SunlitArea_Ver(i)=0.0
    LoadShade_Ver(i)=0.0
  END DO

END IF

! calculate the zone load under the combination of semi-infinite

```



```

! horizontal and semi-infinite vertical fin shading
IF (HorFin .AND. VerFin) THEN

    CALL SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
        Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
        HangLength_Ver, VerFin_Side, SunlitArea_HorVer)
    DO i=1, Num_Sol
        LoadShade_HorVer(i)=SunlitArea_HorVer(i)*Flux_Tol(i)
    END DO

ELSE

    DO i=1, Num_Sol
        SunlitArea_HorVer(i)=0.0
        LoadShade_HorVer(i)=0.0
    END DO

END IF

! Transfer the units of Sur_Azimuth, Sur_Tilt, Altitude,
! Sol_Zenith and Sol_Azimuth
! from radiance to degree for output
DO i=1, Num_Sol
    Altitude(i)=180.*Altitude(i)/Pai
    Sol_Zenith(i)=180.*Sol_Zenith(i)/Pai
    Sol_Azimuth(i)=180.*Sol_Azimuth(i)/Pai
END DO
Sur_Azimuth=Sur_Azimuth*180/Pai
Sur_Tilt=Sur_Tilt*180/Pai

!attempt to open file - full path is passed as FileName by VB application

IF (HorFin .OR. VerFin ) Then

    !open a result file for window shading
    OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
    IF(error/=0) THEN
        ErrorMessage="Shading Result file could not be opened"
        ErrorFlag = 1
        RETURN
    END IF

    ! write header lines
    WRITE(Results, '( "# ASHRAE Analytical Test Suite - &
        Window Shading Result file ")')
    WRITE(Results, '( "# Exterior Solar Radiation - Window shading test")')
    WRITE(Results, '( //"Test Parameters -")')

    WRITE(Results, '( "Inside convection correlation coefficient A =", &
        ", ", F15.4)') Hin_A
    WRITE(Results, '( "Inside convection correlation coefficient C =", &
        ", ", F15.4)') Hin_C
    WRITE(Results, '( "Inside convection correlation coefficient n =", &
        ", ", F15.4)') Hin_n
    WRITE(Results, '( "Outside convection correlation coefficient A =", &
        ", ", F15.4)') Hout_A

```

```

WRITE(Results, '("Outside convection correlation coefficient C =", &
    ", ", F15.4)') Hout_C
WRITE(Results, '("Outside convection correlation coefficient n =", &
    ", ", F15.4)') Hout_n

WRITE(Results, '("Outside air temperature =", ", ", F15.4)') T_outside
WRITE(Results, '("Inside air temperature =", ", ", F15.4)') T_inside

WRITE(Results, '("Fabric layer thickness =", ", ", F15.4)') Thickness
WRITE(Results, '("Surface Azimuth =", ", ", F15.4)') Sur_Azimuth
WRITE(Results, '("Surface Tilt angle =", ", ", F15.4)') Sur_Tilt
WRITE(Results, '("Extinction coefficient of the surface =", ", ", &
    F15.4)') Extinc_Coeff
WRITE(Results, '("Refractive index of the surface =", ", ", &
    F15.4)') RefIndex_Sur
WRITE(Results, '("Surface Area =", ", ", F15.4)') Sur_area
WRITE(Results, '("Hanging length of horizontal fin =", ", ", F15.4)') &
    HangLength_Hor
WRITE(Results, '("Hanging length of vertical fin =", ", ", F15.4)') &
    HangLength_Ver
WRITE(Results, '("Vertical fin side =", ", ", I2)') VerFin_Side
WRITE(Results, '("Window length =", ", ", F15.4)') Win_Length
WRITE(Results, '("Window height =", ", ", F15.4)') Win_height

WRITE(Results, '("//"Test results -")')
WRITE(Results, '("//"# Explanation of the column labels: ")')
WRITE(Results, '("# SunlitArea_HorVer - Sunlit area with &
    a union of semi-infinite horizontal and vertical fin ")')
WRITE(Results, '("# SunlitArea_Hor - Sunlit area with a &
    semi-infinite horizontal fin ")')
WRITE(Results, '("# SunlitArea_Ver - Sunlit area with a &
    semi-infinite vertical fin ")')
WRITE(Results, '("# Load - Zone load without shading ")')
WRITE(Results, '("# LoadShade_HorVer - Zone load with a &
    union of semi-infinite horizontal and vertical fin ")')
WRITE(Results, '("# LoadShade_Hor - Zone load with &
    a semi-infinite horizontal fin ")')
WRITE(Results, '("# LoadShade_Ver - Zone load with &
    a semi-infinite vertical fin ")')

WRITE(Results, '("//"Time", ", ", "SunlitArea_HorVer", ", ", &
    "SunlitArea_Hor", ", ", "SunlitArea_Ver", ", ", "Load", &
    ", ", "LoadShade_HorVer", ", ", "LoadShade_Hor", &
    ", ", "LoadShade_Ver")')

DO i=1, Num_Sol

    WRITE(Results, '(I2.2, ":", I2.2, ", ", F15.4, ", ", F15.4, ", ", &
        F15.4, ", ", F15.4, ", ", F15.4, ", ", F15.4, ", ", F15.4)') &
        N_hour(i), M_minute(i), SunlitArea_HorVer(i), &
        SunlitArea_Hor(i), SunlitArea_Ver(i), Load(i), &
        LoadShade_HorVer(i), LoadShade_Hor(i), LoadShade_Ver(i)

END DO
CLOSE (Results)
ELSE
    OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)

```

```

IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )' )
WRITE(Results, '( "# Exterior Solar Radiation test" )' )
WRITE(Results, '( "/"Test Parameters -" )' )

WRITE(Results, '( "Inside convection correlation coefficient A =", &
    ", ", F15.4 )' ) Hin_A
WRITE(Results, '( "Inside convection correlation coefficient C =", &
    ", ", F15.4 )' ) Hin_C
WRITE(Results, '( "Inside convection correlation coefficient n =", &
    ", ", F15.4 )' ) Hin_n
WRITE(Results, '( "Outside convection correlation coefficient A =", &
    ", ", F15.4 )' ) Hout_A
WRITE(Results, '( "Outside convection correlation coefficient C =", &
    ", ", F15.4 )' ) Hout_C
WRITE(Results, '( "Outside convection correlation coefficient n =", &
    ", ", F15.4 )' ) Hout_n

WRITE(Results, '( "Outside air temperature =", ", ", F15.4 )' ) T_outside
WRITE(Results, '( "Inside air temperature =", ", ", F15.4 )' ) T_inside
WRITE(Results, '( "Fabric layer thickness =", ", ", F15.4 )' ) Thickness
WRITE(Results, '( "Surface Azimuth =", ", ", F15.4 )' ) Sur_Azimuth
WRITE(Results, '( "Surface Tilt angle =", ", ", F15.4 )' ) Sur_Tilt
WRITE(Results, '( "Extinction coefficient of the surface =", &
    ", ", F15.4 )' ) Extinc_Coeff
WRITE(Results, '( "Refractive index of the surface =", &
    ", ", F15.4 )' ) RefIndex_Sur
WRITE(Results, '( "Surface Area =", ", ", F15.4 )' ) Sur_area

WRITE(Results, '( "/"Test results -" )' )
WRITE(Results, '( "/"Peak Load =", ", ", F15.4 )' ) Peak_load

WRITE(Results, '( "/"Time", ", ", "Surface Temp.", ", ", &
    "Inside Convection Coeff.", ", ", &
    "Outside Convection Coeff.", ", ", &
    "Incident Solar Flux", ", ", &
    "Internal Surface Heat Flux", ", ", "Zone Load" )' )

DO i=1, Num_Sol
    WRITE(Results, '( I2.2, ":", I2.2, ", ", F15.4, ", ", F15.4, ", ", &
        F15.4, ", ", F15.4, ", ", F15.4, ", ", F15.4 )' ) &
        N_hour(i), M_minute(i), Ts(i), H_in(i), &
        H_out(i), BeamIrradS(i), Flux_Tol(i), Load(i)
END DO

CLOSE(Results)

END IF

END SUBROUTINE ExtSolRad_Gs

```

```

!-----
SUBROUTINE Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, &
                      Sur_Azimuth, Sur_Tilt)

! PURPOSE:      Calculate the cosine of the incident angle of the surface
!
! CREATED:
!   11.7.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Cos_Theta      -   Array of the cosine of the incident angle
!   Num_Sol         -   The number of data lines in the input file
!   Sol_Zenith      -   Array of solar zenith
!   Sol_Azimuth     -   Array of solar azimuth
!   Sur_Azimuth     -   The surface azimuth
!   Sur_Tilt        -   The surface tilt angle

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol)       :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol),INTENT(in) :: Sol_Zenith
REAL(prec2), DIMENSION(Num_Sol),INTENT(in) :: Sol_Azimuth
REAL(prec2), INTENT(in)                :: Sur_Azimuth
REAL(prec2), INTENT(in)                :: Sur_Tilt

! local variables
REAL(prec2) :: SunDirCosA, SunDirCosB, SunDirCosC
              ! solar direction cosines
REAL(prec2) :: SurDirCosA, SurDirCosB, SurDirCosC
              ! surface direction cosines
INTEGER      :: i

DO i=1, Num_Sol

    ! calculate the solar direction cosines
    SunDirCosA=SIN(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosB=COS(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosC=COS(Sol_Zenith(i))

    ! calculate the surface direction cosines
    SurDirCosA=SIN(Sur_Azimuth)*SIN(Sur_Tilt)

```

```

    SurDirCosB=COS(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosC=COS(Sur_Tilt)

    ! calculate the cosine of incidence angle Theta
    Cos_Theta(i)=SunDirCosA*SurDirCosA+SunDirCosB* &
        SurDirCosB+SunDirCosC*SurDirCosC

END DO

END SUBROUTINE Incid_Angle
!-----

SUBROUTINE SolIrrad_ND(file_in, Num_Sol, Altitude, N_hour, M_minute, &
    BeamIrradN, SolIrrad, Coeff_A, Coeff_B, Time_interval)

    ! PURPOSE:      Calculate the Beam Irradiance in normal
    !
    ! CREATED:
    !   11.7.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   File_in      -   the index of solar position file
    !   Num_Sol      -   The number of data lines in the input file
    !   Altitude     -   Array of solar altitude
    !   N_hour       -   Array of hour element of the time
    !   M_minute     -   Array of minute element of the time
    !   BeamIrradN   -   Array of Beam Irradiance in normal,
    !                   for time intervals like 10 minutes
    !   SolIrrad     -   Array of Beam Irradiance in normal,
    !                   for each hour
    !   Coeff_A      -   apparent solar irradiation at air
    !                   mass equal zero
    !   Coeff_B      -   atmospheric extinction coefficient
    !   Time_interval -   the time interval of the solar
    !                   position data obtained from the US Naval
    !                   Observatory (10 min)

    IMPLICIT NONE

    ! parameters
    INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
    INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

    ! arguments
    integer              :: Num_Sol, file_in
    REAL(prec2), DIMENSION(Num_Sol) :: Altitude
    REAL(prec2), DIMENSION(Num_Sol) :: BeamIrradN
    INTEGER, DIMENSION(Num_Sol)    :: N_hour, M_minute
    REAL(prec1), DIMENSION(24)     :: SolIrrad

```

```

REAL(prec2), INTENT(in) :: Coeff_A
REAL(prec2), INTENT(in) :: Coeff_B
REAL(prec2),INTENT(in) :: Time_interval

! local variables
INTEGER      :: i,j

DO i=1,24
    SolIrrad(i)=0.0
END DO

DO i=1, Num_Sol

    ! calculate the Beam Irradiation in normal
    ! (W/m2 for SI units or BTU/(h.ft2) for IP units)
    BeamIrradN(i)=Coeff_A/EXP(Coeff_B/SIN(Altitude(i)))

    ! calculate the Beam Irradiance in normal in one hour
    ! (J/m2 for SI units or BTU/ft2 for IP units)
    ! (will be used for weather data)
    SolIrrad(N_hour(i)+1)=SolIrrad(N_hour(i)+1)+ &
        BeamIrradN(i)*Time_interval

END DO

END SUBROUTINE SolIrrad_ND

!-----
SUBROUTINE Absorp_Transmit(Cos_Theta, Num_Sol,RefIndex_Air,RefIndex_Sur,&
    Extinc_Coeff,Thickness,Absorp,Transmit)

! PURPOSE:      Calculate the absorption and transmittance of the surface
!
! CREATED:
!   11.19.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Cos_Theta      -   Array of the cosine of the incident angle
!   Num_Sol         -   The number of data lines in the input file
!   RefIndex_Air    -   Refractive index of the air
!   RefIndex_Sur    -   Refractive index of the surface
!   Extinc_Coeff    -   Extinction coefficient of the surface
!   Thickness       -   Thickness of the surface
!   Absorp          -   Array of absorption of the surface
!   Transmit        -   Array of transmittance of the surface

```

```

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol),INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol),INTENT(out) :: Absorp
REAL(prec2), DIMENSION(Num_Sol),INTENT(out) :: Transmit
REAL(prec2), INTENT(in)                    :: Extinc_Coeff
REAL(prec2), INTENT(in)                    :: Thickness
REAL(prec2), INTENT(in)                    :: RefIndex_Air
REAL(prec2), INTENT(in)                    :: RefIndex_Sur

! local variables
REAL(prec2) :: Theta_1      ! Incidence angle of the surface
REAL(prec2) :: Theta_2      ! Refractive angle of the surface
REAL(prec2) :: Transmit_A    ! Transmittance when only absorption
                                ! loss has been considered
REAL(prec2) :: Refl_Per      ! Reflectance for perpenticular
                                ! component of polarization
REAL(prec2) :: Refl_Par      ! Reflectance for parallel
                                ! component of polarization
REAL(prec2) :: Absorp_Per     ! Absorption for perpenticular
                                ! component of polarization
REAL(prec2) :: Absorp_Par     ! Absorption for parallel
                                ! component of polarization
REAL(prec2) :: Transmit_Per   ! Transmittance for perpenticular
                                ! component of polarization
REAL(prec2) :: Transmit_Par   ! Transmittance for parallel
                                ! component of polarization
INTEGER      :: i

DO i=1, Num_Sol

  IF (Cos_Theta(i)<0) THEN
    Absorp(i)=0
    Transmit(i)=0
  else

    Theta_1=ACOS(Cos_Theta(i))
    Theta_2=ASIN(RefIndex_Air/RefIndex_Sur*SIN(Theta_1))

    Transmit_A=EXP(-Extinc_Coeff*Thickness/COS(Theta_2))
    Refl_Per=(SIN(Theta_2-Theta_1))**2/(SIN(Theta_2+Theta_1))**2
    Refl_Par=(TAN(Theta_2-Theta_1))**2/(TAN(Theta_2+Theta_1))**2

    Absorp_Per=(1-Transmit_A)*(1-Refl_Per)/(1-Refl_Per*Transmit_A)
    Absorp_Par=(1-Transmit_A)*(1-Refl_Par)/(1-Refl_Par*Transmit_A)
    Transmit_Per=Transmit_A*(1-Refl_Per)**2/(1-
(Refl_Per*Transmit_A)**2)
    Transmit_Par=Transmit_A*(1-Refl_Par)**2/(1-
(Refl_Par*Transmit_A)**2)

    Absorp(i)=(Absorp_Per+Absorp_Par)/2.
  
```

```

        Transmit(i)=(Transmit_Per+Transmit_Par)/2.

    END IF
END DO

END SUBROUTINE Absorp_Transmit

!-----

SUBROUTINE SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
    Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
    HangLength_Ver, VerFin_Side, SunlitArea_HorVer)

    ! PURPOSE:      Calculate the new sunlit area in considering of
    !                the exteranl shading of the glazed surface
    !
    ! CREATED:
    !   12.10.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Num_Sol      -   The number of data lines in the input file
    !   Sol_Azimuth   -   Array of solar azimuth
    !   Sur_Azimuth   -   The surface azimuth
    !   Altitude      -   Array of solar altitude
    !   Cos_Theta     -   Array of the cosine of the incident angle
    !   Win_Length    -   Window length
    !   Win_height     -   Window height
    !   HangLength_Hor -   Hanging lenth of the horizontal fin
    !   HangLength_Ver -   Hanging lenth of the vertical fin
    !   SunlitArea_HorVer - Sunlit area using semi-infinite
    !                       horizontal and vertical fin
    !   VerFin_Side   -   For deciding the vertical fin
    !                       is on which side of the window
    !                       1--right side; 2--left side
    !
    IMPLICIT NONE

    ! parameters
    INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
    INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
    REAL(prec2), PARAMETER :: Pai=3.1415926

    ! arguments
    integer                                :: Num_Sol
    REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_HorVer
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth

```



```

REAL(prec2), INTENT(in)                :: Sur_Azimuth
REAL(prec2), INTENT(in)                :: Win_Length
REAL(prec2), INTENT(in)                :: Win_height
REAL(prec2), INTENT(in)                :: HangLength_Hor
REAL(prec2), INTENT(in)                :: HangLength_Ver
INTEGER, INTENT(in)                    :: VerFin_Side

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height
REAL(prec2)                :: VerFin_Azimuth ! The azimuth of the vertical fin
INTEGER                    :: i

IF (VerFin_Side .EQV. 1) THEN
    VerFin_Azimuth=Sur_Azimuth+1.5707963
ELSE IF (VerFin_Side .EQV. 2) THEN
    VerFin_Azimuth=Sur_Azimuth-1.5707963
END IF

DO i=1, Num_Sol
    IF (Cos_Theta(i)<0) THEN
        SunlitArea_HorVer(i)=0
    ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
        (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=Win_Length
        END IF
        Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
            TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

    ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
        (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=Win_Length
        ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
        END IF
        Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
            TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
        SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

    ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
        SunlitArea_HorVer(i)=0
    ELSE
        IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
        ELSE
            Sunlit_length(i)=Win_Length
        END IF
        Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
            TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    END IF
END DO

```

```

        SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)
    END IF
END DO

END SUBROUTINE SunlitAreaHorVer

!-----

SUBROUTINE SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
    Cos_Theta,Win_Length,Win_height,HangLength_Hor,SunlitArea_Hor)

    ! PURPOSE:      Calculate the new sunlit area in considering of the
    !                exteranl shading of the glazed surface
    !
    ! CREATED:
    !   12.10.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Num_Sol      -   The number of data lines in the input file
    !   Sol_Azimuth  -   Array of solar azimuth
    !   Sur_Azimuth  -   The surface azimuth
    !   Altitude     -   Array of solar altitude
    !   Cos_Theta    -   Array of the cosine of the incident angle
    !   Win_Length   -   Window length
    !   Win_height   -   Window height
    !   HangLength_Hor -   Hanging lenth of the horizontal fin
    !   SunlitArea_Hor -   Sunlit area using semi-infinite
    !                       horizontal fin

    IMPLICIT NONE

    ! parameters
    INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
    INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
    REAL(prec2), PARAMETER :: Pai=3.1415926

    ! arguments
    integer                                :: Num_Sol
    REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Hor
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
    REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
    REAL(prec2), INTENT(in)                      :: Sur_Azimuth
    REAL(prec2), INTENT(in)                      :: Win_Length
    REAL(prec2), INTENT(in)                      :: Win_height
    REAL(prec2), INTENT(in)                      :: HangLength_Hor

    ! local variables

```

```

REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height
INTEGER      :: i

DO i=1, Num_Sol
  IF (Cos_Theta(i)<0) THEN
    SunlitArea_Hor(i)=0
  ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
    (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
    (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)

  ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
    SunlitArea_Hor(i)=0
  ELSE
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  END IF
END DO

END SUBROUTINE SunlitAreaHor

!-----

SUBROUTINE SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
  Cos_Theta,Win_Length,Win_height,HangLength_Ver,&
  VerFin_Side,SunlitArea_Ver)

! PURPOSE:      Calculate the new sunlit area in considering of the
!               exteranl shading of the glazed surface
!
! CREATED:
!   12.10.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Sol_Azimuth  -   Array of solar azimuth
!   Sur_Azimuth  -   The surface azimuth
!   Altitude     -   Array of solar altitude
!   Cos_Theta    -   Array of the cosine of the incident angle
!   Win_Length   -   Window length
!   Win_height   -   Window height

```

```

!   HangLength_Ver      -   Hanging lenth of the vertical fin
!   VerFin_Side         -   For deciding the vertical fin is on which
!                           side of the window
!                           1--right side; 2--left side
!   SunlitArea_Ver      -   Sunlit area using semi-infinite vertical fin

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Ver
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
REAL(prec2), INTENT(in)                    :: Sur_Azimuth
REAL(prec2), INTENT(in)                    :: Win_Length
REAL(prec2), INTENT(in)                    :: Win_height
REAL(prec2), INTENT(in)                    :: HangLength_Ver
INTEGER, INTENT(in)                        :: VerFin_Side

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height
REAL(prec2)                      :: VerFin_Azimuth ! The azimuth of the vertical fin
INTEGER                          :: i

IF (VerFin_Side .EQV. 1) THEN
    VerFin_Azimuth=Sur_Azimuth+1.5707963
ELSE IF (VerFin_Side .EQV. 2) THEN
    VerFin_Azimuth=Sur_Azimuth-1.5707963
END IF

DO i=1, Num_Sol
    IF (Cos_Theta(i)<0) THEN
        SunlitArea_Ver(i)=0
    ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
        (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=Win_Length
        END IF
        SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

    ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
        (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=Win_Length
        ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &

```

```

                                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
      END IF
      SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
  SunlitArea_Ver(i)=0
ELSE
  IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
    Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                        TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
  ELSE
    Sunlit_length(i)=Win_Length
  END IF
  SunlitArea_Ver(i)=Sunlit_length(i)*Win_height
END IF
END DO

END SUBROUTINE SunlitAreaVer

!-----
SUBROUTINE Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN,Absorp,&
  Transmit,Sur_area,T_outside, T_inside, Hin_A, Hin_C,Hin_n,&
  Hout_A, Hout_C, Hout_n, Flux,Flux_inside, Flux_outside, &
  Flux_Transmit,Flux_Tol,Ts,H_in, H_out,ErrorFlag, &
  ErrorMessage, BeamIrradS,GeoFactor,Load,Peak_load)

! PURPOSE:      Calculate analytical response for the case of exterior
!               solar radiation on glazed surfaces using the method of
!               heat balance. The convection correlations inside
!               and outside are specified as  $H = A + C*(T_a - T_s)^n$ .
!               The steady state heat flux is found using a successive
!               substitution method to solve the non-linear equation.
!               A variable number of wall layers can be specified.
!
! CREATED:      11.27.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Cos_Theta    - Array of the cosine of the incident angle
!   Num_Sol       - The number of data lines in the input file
!   Altitude      - Array of solar altitude
!   BeamIrradN    - Array of beam Irradiance in normal
!   Absorp        - Array of absorption of the surface
!   Transmit      - Array of transmittance of the surface
!   Sur_area      - The surface area
!   T_outside     - Outside air temperature
!   T_inside      - Inside air temperature
!   Hin_A         - Conv. correlation coefficient A for inside

```

```

!   Hin_C           -   Conv. correlation coefficient C for inside
!   Hin_n           -   Conv. correlation coefficient n for inside
!   Hout_A          -   Conv. correlation coefficient A for outside
!   Hout_C          -   Conv. correlation coefficient C for outside
!   Hout_n          -   Conv. correlation coefficient n for outside
!   Flux            -   Array of calculated steady state heat flux
!   Flux_inside     -   Array of calculated internal conduction
!                   heat flux
!   Flux_outside    -   Array of calculated external conduction
!                   heat flux
!   Flux_Transmit    -   Array of solar beam irradiation transmitted
!                   into the zone
!   Flux_Tal        -   Array of Calculated total heat flux
!   Ts              -   Array of Calculated surface temperature
!   H_in            -   Array of Calculated inside conv. coefficient
!   H_out           -   Array of Calculated outside conv. coefficient
!   ErrorFlag       -   for error checking
!   ErrorMessage    -   for error reporting
!   BeamIrradS      -   Beam irradiation on the receiving slope
!   GeoFactor        -   geometric factor used in calculating the
!                   Beam irradiation
!   Load           -   Array of the zone load
!   Load_Shade      -   Array of the zone load in considering of
!                   exteranal shading
!   Peak_load       -   The peak zone load in one day

```

```

IMPLICIT NONE

```

```

! parameters

```

```

INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 500
REAL(prec2), PARAMETER :: Tolerance = 1.0E-06 ! tolerance
REAL(prec2), PARAMETER :: Delta_H = 1.0E-20
! a small number added to the initial value of convection
! coefficient to avoid zero convection coefficients

```

```

! arguments

```

```

INTEGER                                :: Num_Sol
REAL(prec2), DIMENSION(Num_sol), INTENT(in)  :: Absorp
REAL(prec2), DIMENSION(Num_sol), INTENT(in)  :: Transmit
REAL(prec2), DIMENSION(Num_sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), INTENT(in)                  :: T_outside, T_inside
REAL(prec2), INTENT(in)                  :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in)                  :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(in)                  :: Sur_area
REAL(prec2), DIMENSION(Num_sol), INTENT(in)  :: BeamIrradN
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: BeamIrradS
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: GeoFactor
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Tol
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_inside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_outside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Transmit

REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Ts

```

```

REAL(prec2),DIMENSION(Num_sol),INTENT(out)      :: H_in, H_out
REAL(prec2),DIMENSION(Num_sol),INTENT(out)      :: Load
REAL(prec2),INTENT(out)                         :: Peak_load
CHARACTER*(*),INTENT(out)                       :: ErrorMessage
INTEGER(2),INTENT(out)                         :: ErrorFlag ! two byte int for VB integer

! local variables
REAL(prec2),DIMENSION(Num_sol) :: MinFlux, MaxFlux
! Min and max calculated flux values
INTEGER                        :: i, j,n, error
CHARACTER(LEN=132)           :: dummy
INTEGER, PARAMETER           :: DebugUnit = 20
LOGICAL, PARAMETER           :: Debug = .false.
LOGICAL, PARAMETER           :: Warning = .false.

! loop to calculate the geometric factor and
! the Beam irradiation on the receiving slope
DO j=1, Num_sol

    ! calculate the geometric factor used
    !in calculating the Beam irradiation
    IF(SIN(Altitude(j))=0.0)THEN
        WRITE(ErrorMessage,'("The sine of the altitude(", &
            I2," )appears to be zero")) j
        ErrorFlag = 1
        RETURN
    END IF
    GeoFactor(j)=MAX(Cos_Theta(j)/SIN(Altitude(j)), 0.0)

    ! calculate the Beam irradiation on the receiving slope
    IF (Altitude(j)<0) THEN
        BeamIrradS(j)=0
    ELSE
        BeamIrradS(j)=GeoFactor(j)*BeamIrradN(j)*SIN(Altitude(j))
    END IF
END DO

! start loop to calculate the heat flux and internal &
!external surface temperature
DO j=1, Num_sol

    ! open a unit for debug output
    IF(Debug .and. DebugUnit/=6)THEN
        OPEN (DebugUnit, file="ExtSolRad.log", &
            status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Debug output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF
        ! copy out arguments to debug file
        WRITE(DebugUnit,'("ExtSolRad arguments:")')
        WRITE(DebugUnit,'("Inside convection correlation &
            coefficient A  =",F10.6)') Hin_A
        WRITE(DebugUnit,'("Inside convection correlation &

```

```

        coefficient C =",F10.6)') Hin_C
WRITE(DebugUnit, '("Inside convection correlation &
        coefficient n =",F10.6)') Hin_n
WRITE(DebugUnit, '("Outside convection correlation &
        coefficient A =",F10.6)') Hout_A
WRITE(DebugUnit, '("Outside convection correlation &
        coefficient C =",F10.6)') Hout_C
WRITE(DebugUnit, '("Outside convection correlation &
        coefficient n =",F10.6)') Hout_n
WRITE(DebugUnit, '("Outside air temperature =",F10.6)') &
        T_outside
WRITE(DebugUnit, '("Inside air temperature  =",F10.6)') &
        T_inside
END IF

! set initial guess of surface temperatures
Ts(j) = T_outside - 0.33*(T_outside - T_inside)

H_in(j) = Hin_A + Hin_C*ABS(T_inside - Ts(j))*Hin_n + Delta_H
H_out(j) = Hout_A + Hout_C*ABS(T_outside - Ts(j))*Hout_n + Delta_H

! start loop to find Flux by successive substitution
n = 1
DO
    IF(H_in(j)==0.0 .or. H_out(j)==0.0)THEN
        WRITE(ErrorMessage, '("Inside or outside convection coefficient
&
                                calculated to be zero", " - can not
procede")')
        ErrorFlag = 1
        RETURN
    END IF

    ! flux based on current value of total resistance
    Flux(j)= ((T_outside - T_inside)+BeamIrradS(j)* &
        Absorp(j)/H_out(j))/(1/H_in(j)+1/H_out(j))

    ! update surface temps based on flux
    Ts(j) = T_inside + Flux(j)/H_in(j)

    ! re-calculate conv. coef using current temps
    H_in(j) = Hin_A + Hin_C*(ABS(T_inside - Ts(j))*Hin_n)
    H_out(j) = Hout_A + Hout_C*(ABS(T_outside - Ts(j))*Hout_n)

    ! calculate fluxes over different resistances for checking
    Flux_inside(j) = H_in(j)*(Ts(j) - T_inside)
    Flux_outside(j) =BeamIrradS(j)*Absorp(j)+ &
        H_out(j)*(T_outside - Ts(j))

    ! some debug output
    IF(Debug)THEN
        WRITE(DebugUnit, '("H_in = ",F8.5," H_out = ",F8.5, &
            " Ts= ",F8.5)') H_in(j), H_out(j), Ts(j)
        WRITE(DebugUnit, '("Q = ",F8.5," Qout = ",F8.5, &
            " Qin =",F8.5)') Flux(j), Flux_outside(j), &
            Flux_inside(j)

```



```

        END IF

        ! calc min and max fluxes
        MinFlux(j) = MIN(Flux(j), Flux_inside(j), Flux_outside(j))
        MaxFlux(j) = MAX(Flux(j), Flux_inside(j), Flux_outside(j))
        ! test for exit condition
        IF(ABS(MaxFlux(j)-MinFlux(j)) < Tolerance &
        .or. n > Max_Iterations) EXIT
        n = n + 1
    END DO

    ! set warning if max iterations exceded
    IF(n > Max_Iterations)THEN
        ErrorMessage = "The steady-state heat flux could not  &
        be calculated within the allowable number of iterations"
        ErrorFlag = -1
        RETURN
    END IF

    IF(Debug)THEN
        CLOSE(DebugUnit)
    END IF

    ! Calculate the solar beam irradiation transmitted into the Zone
    Flux_Transmit(j)=Transmit(j)*BeamIrradS(j)

    !calculate the total flux transfered into the zone
    Flux_Tol(j)=Flux(j)+Flux_Transmit(j)

    ! calculate the zone load
    Load(j)=Sur_area*Flux_Tol(j)

    ! find out the peak zone load in one day
    IF (j==2) THEN
        Peak_load=MAX(Load(j),Load(j-1))
    END IF
    IF (j>2)THEN
        Peak_load=MAX(Load(j),Peak_load)
    END IF
END DO

END SUBROUTINE Heat_balance

END MODULE ExtSolRadiationGs

```

Source Code: Module WindowReveal

MODULE WindowReveal

```

!-----
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric
!               (1052-RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!-----
!
!
! PURPOSE:      Contains utility routines used in the calculation of
!               analytical solutions to the exterior solar radiation on
!               glazed surface and the window shading/window reveal
!               test case.
!
! CREATED:      11.30.2000      Dongyi Xiao
!
! MODIFIED:
!
PRIVATE

! make public main subroutine
PUBLIC :: WinReveal

CONTAINS

!-----

SUBROUTINE WinReveal(SolIrrad,Num_hour, Sur_Azimuth,Sur_Tilt,FileName,&
File_in,Thickness,Extinc_Coeff,RefIndex_Sur,T_outside,&
T_inside,Hin_A,Hin_C, Hin_n,Hout_A, Hout_C, Hout_n,&
HorFin,VerFin,Reveal,VerFin_Side,&
Reveal_width,Win_Length,Win_height,&
Coeff_A,Coeff_B,Time_interval,&
ErrorFlag, ErrorMessage,DirName, Length_DirName)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "WinReveal" :: WinReveal

! PURPOSE:      Produces weather files and analytical response for the
!               case of exterior solar radiation on glazed surfaces and
!               window shading using the method of heat balance.

```

```

!           The convection correlations inside and outside are
!           specified as  $H = A + C \cdot (T_a - T_s)^n$ . The steady state heat
!           flux is found using a successive substitution
!           method to solve the non-linear equation.
!
!   CREATED:
!     11.30.2000      Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!     n/a
!
!   OUTPUT FILES:
!
!     FileName.csv      -   Analytical response data
!
!   SUBROUTINE ARGUMENTS:
!     SolIrrad          -   Array of Beam Irradiance in normal,
!                           for each hour
!     Num_hour          -   number of hours in one day
!     Sur_Azimuth       -   The surface azimuth
!     Sur_Tilt          -   The surface tilt angle
!     FileName          -   Root File Name for weather and data files
!     File_in           -   the index of solar position file
!     Thickness         -   Thicknesses of the surface
!     Extinc_Coeff      -   Extinction coefficient of the surface
!     RefIndex_Sur      -   Refractive index of the surface
!     T_outside         -   Outside air temperature
!     T_inside          -   Inside air temperature
!     Hin_A             -   Conv. correlation coefficient A for inside
!     Hin_C             -   Conv. correlation coefficient C for inside
!     Hin_n             -   Conv. correlation coefficient n for inside
!     Hout_A            -   Conv. correlation coefficient A for outside
!     Hout_C            -   Conv. correlation coefficient C for outside
!     Hout_n            -   Conv. correlation coefficient n for outside
!     HorFin            -   Logical variable, it is true
!                           if using semi-infinite horizontal fin
!     VerFin            -   Logical variable
!                           it is true if using semi-infinite vertical fin
!     Reveal            -   Logical variable,
!                           it is true if using window reveal
!     VerFin_Side       -   For deciding the vertical fin is on
!                           which side of the window
!     Reveal_width      -   Width of the window reveal
!     Win_Length        -   Window length
!     Win_height        -   Window height
!     Coeff_A           -   apparent solar irradiation at
!                           air mass equal zero
!     Coeff_B           -   atmospheric extinction coefficient
!     Time_interval     -   the time interval of the solar position data
!                           obtained from the US Naval Observatory
!                           (10 min)
!     ErrorFlag         -   for error checking
!     ErrorMessage      -   for error reporting
!     DirName           -   the directory of the VB interface executive
!                           file resides in.
!

```

```

!   Length_DirName      -   the length of the DirName character

IMPLICIT NONE

!parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: input=5 ! unit number for input file
INTEGER, PARAMETER :: Results=7
      ! unit number for analytical response output file
REAL(prec2), PARAMETER :: Pai=3.1415926
REAL(prec2), PARAMETER :: RefIndex_Air=1.

!arguments
REAL(prec2)                :: Sur_Azimuth
REAL(prec2)                :: Sur_Tilt
REAL(prec2), INTENT(in)    :: Extinc_Coeff
REAL(prec2), INTENT(in)    :: Thickness

REAL(prec2), INTENT(in)    :: RefIndex_Sur
REAL(prec2), INTENT(in)    :: Win_Length
REAL(prec2), INTENT(in)    :: Win_height
REAL(prec2), INTENT(in)    :: Reveal_width

REAL(prec2), INTENT(in)    :: Coeff_A
REAL(prec2), INTENT(in)    :: Coeff_B
REAL(prec2), INTENT(in)    :: Time_interval

INTEGER, INTENT(in)        :: Length_DirName
CHARACTER (LEN=Length_DirName), INTENT(in) :: DirName

CHARACTER *(*), INTENT(in) :: FileName
INTEGER, INTENT(in)        :: VerFin_Side
INTEGER, INTENT(in)        :: File_in,Num_hour

REAL(prec2), INTENT(in)    :: T_outside, T_inside
REAL(prec2), INTENT(in)    :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in)    :: Hout_A, Hout_C, Hout_n

REAL(prec1), DIMENSION(Num_hour), INTENT(out) :: SolIrrad
CHARACTER*(*), INTENT(out)                    :: ErrorMessage
INTEGER(2), INTENT(out) :: ErrorFlag ! two byte int for VB integer

LOGICAL, INTENT(in)        :: HorFin
LOGICAL, INTENT(in)        :: VerFin
LOGICAL, INTENT(in)        :: Reveal

!local variables
INTEGER                :: i,n,error
INTEGER                :: Num_Sol
      ! The number of data lines in the input file
INTEGER, DIMENSION(150) :: N_hour, M_minute
      ! Array of hour, minute element of the time
REAL(prec2), DIMENSION(150) :: BeamIrradN

```

```

! Array of beam Irradiance in normal
REAL(prec2), DIMENSION(150)      :: BeamIrradS
! Beam irradiation on the receiving slope
REAL(prec2), DIMENSION(150)      :: GeoFactor ! Geometric factor used in
!calculating the Beam irradiation
REAL(prec2), DIMENSION(150)      :: Cos_Theta
! Array of the cosine of the incident angle
REAL(prec2), DIMENSION(150)      :: Altitude ! Array of solar altitude
REAL(prec2), DIMENSION(150)      :: Sol_Zenith ! Array of solar zenith
REAL(prec2), DIMENSION(150)      :: Sol_Azimuth ! Array of solar azimuth
REAL(prec2), DIMENSION(150)      :: Flux
! Array of flux transferred into the zone by convection
REAL(prec2), DIMENSION(150)      :: Flux_inside
! Array of calculated internal convective heat flux
REAL(prec2), DIMENSION(150)      :: Flux_outside
! Array of calculated external heat flux
REAL(prec2), DIMENSION(150)      :: Flux_Transmit
! Array of solar beam irradiation transmitted into the zone

REAL(prec2), DIMENSION(150)      :: Absorp
! Array of absorption of the surface
REAL(prec2), DIMENSION(150)      :: Transmit
! Array of transmittance of the surface
REAL(prec2), DIMENSION(150)      :: Load
REAL(prec2), DIMENSION(150)      :: LoadShade_HorVer
REAL(prec2), DIMENSION(150)      :: LoadShade_Hor
REAL(prec2), DIMENSION(150)      :: LoadShade_Ver
REAL(prec2), DIMENSION(150)      :: LoadShade_Reveal
REAL(prec2), DIMENSION(150)      :: Flux_Tol
REAL(prec2), DIMENSION(150)      :: Ts
REAL(prec2), DIMENSION(150)      :: H_in, H_out
REAL(prec2), DIMENSION(150)      :: SunlitArea_HorVer
REAL(prec2), DIMENSION(150)      :: SunlitArea_Hor
REAL(prec2), DIMENSION(150)      :: SunlitArea_Ver
REAL(prec2), DIMENSION(150)      :: SunlitArea_Reveal

REAL(prec2)                       :: Peak_load
REAL(prec2)                       :: Sur_area
REAL(prec2)                       :: HangLength_Hor
REAL(prec2)                       :: HangLength_Ver

CHARACTER(LEN=360) :: dummy          ! dummy string variable

n=Length_DirName
dummy(1:n)=DirName(1:n)

Sur_area=Win_Length*Win_height
HangLength_Ver=Reveal_width
HangLength_hor=Reveal_width

! open the input file
IF(file_in==1) THEN
  dummy(n+1:)="\Atlanta_June.txt"
  OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)

```

```

ELSE IF(file_in==2) THEN
    dummy(n+1:) = "\Atlanta_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==3) THEN
    dummy(n+1:) = "\Boston_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==4) THEN
    dummy(n+1:) = "\Boston_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==5) THEN
    dummy(n+1:) = "\Chicago_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==6) THEN
    dummy(n+1:) = "\Chicago_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==7) THEN
    dummy(n+1:) = "\Los Angeles_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==8) THEN
    dummy(n+1:) = "\Los Angeles_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
END IF
IF(error/=0) THEN
    ErrorMessage="input file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! read the solar position data from the input file
READ(input,*) Num_sol
DO i=1, Num_Sol
    READ(input, *) N_hour(i), M_minute(i), Altitude(i), Sol_Azimuth(i)
END DO
CLOSE(input)

! Transfer the unit from degree to radian
! and calculate the solar zenith angle
DO i=1, Num_Sol
    Altitude(i)=Pai*Altitude(i)/180.
    Sol_Zenith(i)=Pai/2.-Altitude(i)
    Sol_Azimuth(i)=Pai*Sol_Azimuth(i)/180.
END DO

! Transfer the unit of surface tilt angle and
! surface azimuth from degree to radian
Sur_Azimuth=Sur_Azimuth*Pai/180
Sur_Tilt=Sur_Tilt*Pai/180

! calculate the cosine of the incident angle of the surface
CALL Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, &
    Sur_Azimuth, Sur_Tilt)

! calculate the Beam Irradiance in normal
CALL SolIrrad_ND(file_in,Num_Sol,Altitude,N_hour, M_minute,&
    BeamIrradN, SolIrrad,Coeff_A,Coeff_B,Time_interval)

```

```

! calculate the absorption and transmittance of the surface
CALL Absorp_Transmit(Cos_Theta, Num_Sol, RefIndex_Air, RefIndex_Sur, &
                    Extinc_Coeff, Thickness, Absorp, Transmit)

! calc analytical response
CALL Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN, Absorp, &
                Transmit, Sur_area, T_outside, T_inside, Hin_A, Hin_C, Hin_n, &
                Hout_A, Hout_C, Hout_n, Flux, Flux_inside, Flux_outside, &
                Flux_Transmit, Flux_Tol, Ts, H_in, H_out, ErrorFlag, ErrorMessage, &
                BeamIrradS, GeoFactor, Load, Peak_load)

! calculate the zone load under semi-infinite horizontal fin shading
IF (HorFin) THEN

    CALL SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
                    Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
                    SunlitArea_Hor)
    DO i=1, Num_Sol
        LoadShade_Hor(i)=SunlitArea_Hor(i)*Flux_Tol(i)
    END DO

END IF

! calculate the zone load under semi-infinite vertical fin shading
IF (VerFin) THEN

    CALL SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
                    Cos_Theta, Win_Length, Win_height, HangLength_Ver, &
                    VerFin_Side, SunlitArea_Ver)
    DO i=1, Num_Sol
        LoadShade_Ver(i)=SunlitArea_Ver(i)*Flux_Tol(i)
    END DO

END IF

! calculate the zone load under the combination of
! semi-infinite horizontal and semi-infinite vertical fin shading
IF (HorFin .AND. VerFin) THEN

    CALL SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
                    Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
                    HangLength_Ver, VerFin_Side, SunlitArea_HorVer)
    DO i=1, Num_Sol
        LoadShade_HorVer(i)=SunlitArea_HorVer(i)*Flux_Tol(i)
    END DO

END IF

! calculate the zone load under the shading of window reveal
IF (Reveal) THEN

    CALL SunlitAreaReveal(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
                    Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
                    HangLength_Ver, SunlitArea_Reveal)
    DO i=1, Num_Sol
        LoadShade_Reveal(i)=SunlitArea_Reveal(i)*Flux_Tol(i)
    END DO

```

```

END DO

END IF

! Transfer the units of Sur_Tilt, Sur_Azimuth,Altitude,
! Sol_Zenith and Sol_Azimuth from radiance to degree for output
DO i=1, Num_Sol
  Altitude(i)=180.*Altitude(i)/Pai
  Sol_Zenith(i)=180.*Sol_Zenith(i)/Pai
  Sol_Azimuth(i)=180.*Sol_Azimuth(i)/Pai
END DO
Sur_Azimuth=Sur_Azimuth*180/Pai
Sur_Tilt=Sur_Tilt*180/Pai

IF (Reveal) Then

  !open a result file for window shading
  OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
  IF(error/=0) THEN
    ErrorMessage="Window reveal result file could not be opened"
    ErrorFlag = 1
    RETURN
  END IF

  ! write header lines
  WRITE(Results,'("# ASHRAE Analytical Test Suite - &
    Window Reveal Result file ")')
  WRITE(Results,'("# Exterior Solar Radiation - &
    Window Reveal test")')
  WRITE(Results,'("//Test Parameters -")')

  WRITE(Results,'("Inside convection correlation &
    coefficient A =",",",F15.4)') Hin_A
  WRITE(Results,'("Inside convection correlation &
    coefficient C =",",",F15.4)') Hin_C
  WRITE(Results,'("Inside convection correlation &
    coefficient n =",",",F15.4)') Hin_n
  WRITE(Results,'("Outside convection correlation &
    coefficient A =",",",F15.4)') Hout_A
  WRITE(Results,'("Outside convection correlation &
    coefficient C =",",",F15.4)') Hout_C
  WRITE(Results,'("Outside convection correlation &
    coefficient n =",",",F15.4)') Hout_n

  WRITE(Results,'("Outside air temperature =",",",F15.4)') T_outside
  WRITE(Results,'("Inside air temperature =",",",F15.4)') T_inside

  WRITE(Results,'("Fabric layer thickness =",",",F15.4)') Thickness
  WRITE(Results,'("Surface Azimuth =",",",F15.4)') Sur_Azimuth
  WRITE(Results,'("Surface Tilt angle =",",",F15.4)') Sur_Tilt
  WRITE(Results,'("Extinction coefficient of the surface =",&
    ",",F15.4)') &
    Extinc_Coeff
  WRITE(Results,'("Refractive index of the surface =",",",F15.4)') &
    RefIndex_Sur

```



```

WRITE(Results, '("Surface Area =", "", F15.4)') Sur_area
WRITE(Results, '("Width of the window reveal =", "", F15.4)') &
    Reveal_width

WRITE(Results, '("Window length =", "", F15.4)') Win_Length
WRITE(Results, '("Window height =", "", F15.4)') Win_height

WRITE(Results, '("//"Test results -")')

WRITE(Results, '("//"Time", "", "Sunlit Area with Window Reveal", "", &
    "Zone Load without Window Reveal", &
    "", "Zone Load with Window Reveal")')

DO i=1, Num_Sol

    WRITE(Results, '(I2.2, ":", I2.2, "", F15.4, "", F15.4, "", F15.4)') &
        N_hour(i), M_minute(i), SunlitArea_Reveal(i), Load(i), &
        LoadShade_Reveal(i)

END DO
CLOSE (Results)

ELSE IF (HorFin .OR. VerFin ) Then

    !open a result file for window shading
    OPEN (Results, file="SolRad_Shade.dat", &
        status='unknown', IOSTAT=error)
    IF(error/=0) THEN
        ErrorMessage="Shading Result file could not be opened"
        ErrorFlag = 1
        RETURN
    END IF

    ! write header lines
    WRITE(Results, '("# ASHRAE Analytical Test Suite - &
        Window Shading Result file ")')
    WRITE(Results, '("# Exterior Solar Radiation - &
        Window shading test")')
    WRITE(Results, '("//"Test Parameters -")')

    WRITE(Results, '("Inside convection correlation &
        coefficient A =", "", F15.4)') Hin_A
    WRITE(Results, '("Inside convection correlation &
        coefficient C =", "", F15.4)') Hin_C
    WRITE(Results, '("Inside convection correlation &
        coefficient n =", "", F15.4)') Hin_n
    WRITE(Results, '("Outside convection correlation &
        coefficient A =", "", F15.4)') Hout_A
    WRITE(Results, '("Outside convection correlation &
        coefficient C =", "", F15.4)') Hout_C
    WRITE(Results, '("Outside convection correlation &
        coefficient n =", "", F15.4)') Hout_n

    WRITE(Results, '("Outside air temperature =", "", F15.4)') T_outside
    WRITE(Results, '("Inside air temperature =", "", F15.4)') T_inside

```

```

WRITE(Results,'("Fabric layer thickness =",",",F15.4)') Thickness
WRITE(Results,'("Surface Azimuth =",",",F15.4)') Sur_Azimuth
WRITE(Results,'("Surface Tilt angle =",",",F15.4)') Sur_Tilt
WRITE(Results,'("Extinction coefficient of the surface =",&
    ",",F15.4)') Extinc_Coeff
WRITE(Results,'("Refractive index of the surface =",&
    ",",F15.4)') RefIndex_Sur

WRITE(Results,'("Surface Area =",",",F15.4)') Sur_area
WRITE(Results,'("Hanging length of horizontal fin =",",",F15.4)') &
    HangLength_Hor
WRITE(Results,'("Hanging length of vertical fin =",",",F15.4)') &
    HangLength_Ver
WRITE(Results,'("Vertical fin side =",",",I2)') VerFin_Side
WRITE(Results,'("Window length =",",",F15.4)') Win_Length
WRITE(Results,'("Window height =",",",F15.4)') Win_height

WRITE(Results,'(// "Test results -")')

WRITE(Results,'(// "# Explanation of the column labels: ")')
WRITE(Results,'("# SunlitArea_HorVer - Sunlit area with &
    a union of semi-infinite horizontal and vertical fin ")')
WRITE(Results,'("# SunlitArea_Hor - Sunlit area with a &
    semi-infinite horizontal fin ")')
WRITE(Results,'("# SunlitArea_Ver - Sunlit area with a &
    semi-infinite vertical fin ")')
WRITE(Results,'("# Load - Zone load without shading ")')
WRITE(Results,'("# LoadShade_HorVer - Zone load with a &
    union of semi-infinite &
    horizontal and vertical fin ")')
WRITE(Results,'("# LoadShade_Hor - Zone load with a &
    semi-infinite horizontal fin ")')
WRITE(Results,'("# LoadShade_Ver - Zone load with a &
    semi-infinite vertical fin ")')

WRITE(Results,'(// "Time",",",",SunlitArea_HorVer",",",&
    "SunlitArea_Hor",",",",SunlitArea_Ver",",",",Load",",", &
    "LoadShade_HorVer",",",",LoadShade_Hor", &
    ",",",LoadShade_Ver")')

DO i=1, Num_Sol

    WRITE(Results,'(I2.2,":",I2.2, ",",F15.4, ",",F15.4, ",", &
        F15.4, ",",F15.4,"",F15.4, ",",F15.4, ",",F15.4)') &
        N_hour(i), M_minute(i),SunlitArea_HorVer(i),&
        SunlitArea_Hor(i),SunlitArea_Ver(i),Load(i),&
        LoadShade_HorVer(i),LoadShade_Hor(i), LoadShade_Ver(i)

END DO
CLOSE (Results)
ELSE
    !attempt to open file - full path is
    !passed as FileName by VB application
    OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)

```

```

IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results,'("# ASHRAE Analytical Test Suite - Results file")')
WRITE(Results,'("# Exterior Solar Radiation test")')
WRITE(Results,'("//Test Parameters -")')

WRITE(Results,'("Inside convection correlation &
    coefficient A =",",",F15.4)') Hin_A
WRITE(Results,'("Inside convection correlation &
    coefficient C =",",",F15.4)') Hin_C
WRITE(Results,'("Inside convection correlation &
    coefficient n =",",",F15.4)') Hin_n
WRITE(Results,'("Outside convection correlation &
    coefficient A =",",",F15.4)') Hout_A
WRITE(Results,'("Outside convection correlation &
    coefficient C =",",",F15.4)') Hout_C
WRITE(Results,'("Outside convection correlation &
    coefficient n =",",",F15.4)') Hout_n

WRITE(Results,'("Outside air temperature =",",",F15.4)') T_outside
WRITE(Results,'("Inside air temperature =",",",F15.4)') T_inside

WRITE(Results,'("Fabric layer thickness =",",",F15.4)') Thickness
WRITE(Results,'("Surface Azimuth =",",",F15.4)') Sur_Azimuth
WRITE(Results,'("Surface Tilt angle =",",",F15.4)') Sur_Tilt

WRITE(Results,'("Extinction coefficient of the surface =",&
    ",",F15.4)') Extinc_Coeff
WRITE(Results,'("Refractive index of the surface =",&
    ",",F15.4)') RefIndex_Sur
WRITE(Results,'("Surface Area =",",",F15.4)') Sur_area

WRITE(Results,'("//Test results -")')
WRITE(Results,'("//Peak Load =",",",F15.4)') Peak_load

WRITE(Results,'("//Time",",",,"Surface Temp.",",",&
    "Inside Convection Coeff.",",", &
    "Outside Convection Coeff.",",",,"Incident Solar Flux",",", &
    "Internal Surface Heat Flux",",",,"Zone Load")')

DO i=1, Num_Sol
    WRITE(Results,'(I2.2,":",I2.2, ",",F15.4, ",",F15.4, &
        ",",F15.4, ",",F15.4, ",",F15.4,&
        ",",F15.4)') N_hour(i), M_minute(i),Ts(i),H_in(i), &
        H_out(i),BeamIrradS(i),Flux_Tol(i),Load(i)
END DO

CLOSE(Results)

END IF

```

```

END SUBROUTINE WinReveal

!-----

SUBROUTINE Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, &
                      Sur_Azimuth, Sur_Tilt)

! PURPOSE:      Calculate the cosine of the incident angle of the surface
!
! CREATED:
!   11.7.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Cos_Theta    -   Array of the cosine of the incident angle
!   Sol_Zenith   -   Array of solar zenith
!   Sol_Azimuth  -   Array of solar azimuth
!   Sur_Azimuth  -   The surface azimuth
!   Sur_Tilt     -   The surface tilt angle

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol) :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Sol_Zenith
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Sol_Azimuth
REAL(prec2), INTENT(in) :: Sur_Azimuth
REAL(prec2), INTENT(in) :: Sur_Tilt

! local variables
REAL(prec2) :: SunDirCosA, SunDirCosB, SunDirCosC
! solar direction cosines
REAL(prec2) :: SurDirCosA, SurDirCosB, SurDirCosC
! surface direction cosines
INTEGER      :: i

DO i=1, Num_Sol

! calculate the solar direction cosines
SunDirCosA=SIN(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
SunDirCosB=COS(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
SunDirCosC=COS(Sol_Zenith(i))

! calculate the surface direction cosines

```

```

    SurDirCosA=SIN(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosB=COS(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosC=COS(Sur_Tilt)

    ! calculate the cosine of incidence angle Theta
    Cos_Theta(i)=SunDirCosA*SurDirCosA+SunDirCosB*SurDirCosB &
        +SunDirCosC*SurDirCosC

END DO

END SUBROUTINE Incid_Angle

!-----
SUBROUTINE SolIrrad_ND(file_in,Num_Sol,Altitude,N_hour, M_minute, &
    BeamIrradN, SolIrrad,&
    Coeff_A,Coeff_B,Time_interval)

! PURPOSE:      Calculate the Beam Irradiance in normal
!
! CREATED:
!   11.7.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   File_in      -   the index of solar position file
!   Num_Sol      -   The number of data lines in the input file
!   Altitude     -   Array of solar altitude
!   N_hour       -   Array of hour element of the time
!   M_minute     -   Array of minute element of the time
!   BeamIrradN   -   Array of Beam Irradiance in normal,
!                   for time intervals like 10 minutes
!   SolIrrad     -   Array of Beam Irradiance in normal,
!                   for each hour
!   Coeff_A      -   apparent solar irradiation at air
!                   mass equal zero
!   Coeff_B      -   atmospheric extinction coefficient
!   Time_interval -   the time interval of the solar position data
!                   obtained from the US Naval Observatory
!                   (10 min)
!
IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol,file_in

```

```

REAL(prec2), DIMENSION(Num_Sol) :: Altitude
REAL(prec2), DIMENSION(Num_Sol) :: BeamIrradN
INTEGER, DIMENSION(Num_Sol)      :: N_hour, M_minute
REAL(prec1), DIMENSION(24)       :: SolIrrad

! local variables
REAL(prec2), INTENT(in) :: Coeff_A
REAL(prec2), INTENT(in) :: Coeff_B
REAL(prec2), INTENT(in) :: Time_interval

INTEGER      :: i,j

DO i=1,24
    SolIrrad(i)=0.0
END DO

DO i=1, Num_Sol

    ! calculate the Beam Irradiation in normal
    ! (W/m2 for SI units or BTTU/(h.ft2) for IP units)
    BeamIrradN(i)=Coeff_A/EXP(Coeff_B/SIN(Altitude(i)))
    ! calculate the Beam Irradiance in normal in one hour
    ! (J/m2 for SI units or BTU/ft2 for IP units)
    ! (will be used for weather data)
    SolIrrad(N_hour(i)+1)=SolIrrad(N_hour(i)+1)+ &
        BeamIrradN(i)*Time_interval

END DO

END SUBROUTINE SolIrrad_ND

!-----
SUBROUTINE Absorp_Transmit(Cos_Theta, Num_Sol, RefIndex_Air, RefIndex_Sur, &
    Extinc_Coeff, Thickness, Absorp, Transmit)

! PURPOSE:      Calculate the absorption and transmittance of the surface
!
! CREATED:
!   11.19.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Cos_Theta     -   Array of the cosine of the incident angle
!   RefIndex_Air  -   Refractive index of the air
!   RefIndex_Sur  -   Refractive index of the surface
!   Extinc_Coeff  -   Extinction coefficient of the surface
!   Thickness     -   Thickness of the surface
!   Absorp        -   Array of absorption of the surface

```

```

!   Transmit           -   Array of transmittance of the surface

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol),INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol),INTENT(out) :: Absorp
REAL(prec2), DIMENSION(Num_Sol),INTENT(out) :: Transmit
REAL(prec2), INTENT(in)                   :: Extinc_Coeff
REAL(prec2), INTENT(in)                   :: Thickness
REAL(prec2), INTENT(in)                   :: RefIndex_Air
REAL(prec2), INTENT(in)                   :: RefIndex_Sur

! local variables
REAL(prec2) :: Theta_1      ! Incidence angle of the surface
REAL(prec2) :: Theta_2      ! Refractive angle of the surface
REAL(prec2) :: Transmit_A   ! Transmittance when only
                                ! absorption loss has been considered
REAL(prec2) :: Refl_Per     ! Reflectance for perpenticular
                                ! component of polarization
REAL(prec2) :: Refl_Par     ! Reflectance for parallel
                                ! component of polarization
REAL(prec2) :: Absorp_Per   ! Absorption for perpenticular
                                ! component of polarization
REAL(prec2) :: Absorp_Par   ! Absorption for parallel
                                ! component of polarization
REAL(prec2) :: Transmit_Per ! Transmittance for perpenticular
                                ! component of polarization
REAL(prec2) :: Transmit_Par ! Transmittance for parallel
                                ! component of polarization
INTEGER      :: i

DO i=1, Num_Sol

    IF (Cos_Theta(i)<0) THEN
        Absorp(i)=0
        Transmit(i)=0
    else

        Theta_1=ACOS(Cos_Theta(i))
        Theta_2=ASIN(RefIndex_Air/RefIndex_Sur*SIN(Theta_1))

        Transmit_A=EXP(-Extinc_Coeff*Thickness/COS(Theta_2))
        Refl_Per=(SIN(Theta_2-Theta_1))**2/(SIN(Theta_2+Theta_1))**2
        Refl_Par=(TAN(Theta_2-Theta_1))**2/(TAN(Theta_2+Theta_1))**2

        Absorp_Per=(1-Transmit_A)*(1-Refl_Per)/(1+Refl_Per*Transmit_A)
        Absorp_Par=(1-Transmit_A)*(1-Refl_Par)/(1+Refl_Par*Transmit_A)
        Transmit_Per=Transmit_A*(1-Refl_Per)**2/(1-
(Refl_Per*Transmit_A)**2)

```

```

        Transmit_Par=Transmit_A*(1-Refl_Par)**2/(1-
(Refl_Par*Transmit_A)**2)

        Absorp(i)=(Absorp_Per+Absorp_Par)/2.
        Transmit(i)=(Transmit_Per+Transmit_Par)/2.

    END IF
END DO

END SUBROUTINE Absorp_Transmit

!-----
SUBROUTINE SunlitAreaReveal(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
                           Cos_Theta,Win_Length,Win_height,HangLength_Hor,&
                           HangLength_Ver,SunlitArea_Reveal)

! PURPOSE:      Calculate the new sunlit area in considering of the
!               exteranl shading ofthe glazed surface
!
! CREATED:
!   12.10.99    Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Sol_Azimuth   -   Array of solar azimuth
!   Sur_Azimuth   -   The surface azimuth
!   Altitude      -   Array of solar altitude
!   Cos_Theta     -   Array of the cosine of the incident angle
!   Win_Length    -   Window length
!   Win_height    -   Window height
!   HangLength_Hor -   Hanging lenth of the horizontal fin
!   HangLength_Ver -   Hanging lenth of the vertical fin
!   SunlitArea_Reveal -   Sunlit area using setback
!                               window shading strategy

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Reveal
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth

```



```

REAL(prec2), INTENT(in)                :: Sur_Azimuth
REAL(prec2), INTENT(in)                :: Win_Length
REAL(prec2), INTENT(in)                :: Win_height
REAL(prec2), INTENT(in)                :: HangLength_Hor
REAL(prec2), INTENT(in)                :: HangLength_Ver

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height
INTEGER                        :: i

DO i=1, Num_Sol
  IF (Cos_Theta(i)<0) THEN
    SunlitArea_Reveal(i)=0
  ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
    SunlitArea_Reveal(i)=0
  ELSE
    Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                        TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
                        TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    SunlitArea_Reveal(i)=Sunlit_length(i)*Sunlit_height(i)
  END IF
END DO

END SUBROUTINE SunlitAreaReveal

!-----
SUBROUTINE SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
                          Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
                          HangLength_Ver, VerFin_Side, SunlitArea_HorVer)

! PURPOSE:      Calculate the new sunlit area in considering of
!               the exteranl shading of the glazed surface
!
! CREATED:
!   12.10.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Sol_Azimuth   -   Array of solar azimuth
!   Sur_Azimuth   -   The surface azimuth
!   Altitude      -   Array of solar altitude
!   Cos_Theta     -   Array of the cosine of the incident angle
!   Win_Length    -   Window length
!   Win_height    -   Window height

```

```

!   HangLength_Hor       -   Hanging lenth of the horizontal fin
!   HangLength_Ver       -   Hanging lenth of the vertical fin
!   SunlitArea_HorVer    -   Sunlit area using semi-infinite horizontal
!                           and vertical fin
!   VerFin_Side          -   For deciding the vertical fin is on
!                           which side of the window
!                           1--right side; 2--left side

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_HorVer
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
REAL(prec2), INTENT(in)                     :: Sur_Azimuth
REAL(prec2), INTENT(in)                     :: Win_Length
REAL(prec2), INTENT(in)                     :: Win_height
REAL(prec2), INTENT(in)                     :: HangLength_Hor
REAL(prec2), INTENT(in)                     :: HangLength_Ver
INTEGER, INTENT(in)                         :: VerFin_Side

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
! Sunlit rectangle length and height
REAL(prec2) :: VerFin_Azimuth ! The azimuth of the vertical fin
INTEGER      :: i

IF (VerFin_Side .EQV. 1) THEN
    VerFin_Azimuth=Sur_Azimuth+1.5707963
ELSE IF (VerFin_Side .EQV. 2) THEN
    VerFin_Azimuth=Sur_Azimuth-1.5707963
END IF

DO i=1, Num_Sol
    IF (Cos_Theta(i)<0) THEN
        SunlitArea_HorVer(i)=0
    ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
        (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=Win_Length
        END IF
        Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
            TAN(Altitude(i)/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

    ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &

```

```

        (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
          IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=Win_Length
          ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                  TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
          END IF
          Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
                                TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
          SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

        ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
          SunlitArea_HorVer(i)=0
        ELSE
          IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                  TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
          ELSE
            Sunlit_length(i)=Win_Length
          END IF
          Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
                                TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
          SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)
        END IF
      END DO

END SUBROUTINE SunlitAreaHorVer

!-----
SUBROUTINE SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
                        Cos_Theta,Win_Length,Win_height,HangLength_Hor,SunlitArea_Hor)

! PURPOSE:      Calculate the new sunlit area in considering of
!               the exteranl shading of the glazed surface
!
! CREATED:
!   12.10.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      - The number of data lines in the input file
!   Sol_Azimuth   - Array of solar azimuth
!   Sur_Azimuth   - The surface azimuth
!   Altitude      - Array of solar altitude
!   Cos_Theta     - Array of the cosine of the incident angle
!   Win_Length    - Window length
!   Win_height    - Window height
!   HangLength_Hor - Hanging lenth of the horizontal fin
!   SunlitArea_Hor - Sunlit area using semi-infinite horizontal fin

```

```

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Hor
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
REAL(prec2), INTENT(in)                      :: Sur_Azimuth
REAL(prec2), INTENT(in)                      :: Win_Length
REAL(prec2), INTENT(in)                      :: Win_height
REAL(prec2), INTENT(in)                      :: HangLength_Hor

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height
INTEGER      :: i

DO i=1, Num_Sol
  IF (Cos_Theta(i)<0) THEN
    SunlitArea_Hor(i)=0
  ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
    (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
    (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)

  ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
    SunlitArea_Hor(i)=0
  ELSE
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  END IF
END DO

END SUBROUTINE SunlitAreaHor

!-----
-

SUBROUTINE SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude,&
  Cos_Theta,Win_Length,Win_height,HangLength_Ver,&
  VerFin_Side,SunlitArea_Ver)

```

```

! PURPOSE:      Calculate the new sunlit area in considering of
!                the exteranl shading of the glazed surface
!
! CREATED:
!   12.10.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Sol_Azimuth  -   Array of solar azimuth
!   Sur_Azimuth  -   The surface azimuth
!   Altitude     -   Array of solar altitude
!   Cos_Theta    -   Array of the cosine of the incident angle
!   Win_Length   -   Window length
!   Win_height   -   Window height
!   HangLength_Ver - Hanging lenth of the vertical fin
!   VerFin_Side  -   For deciding the vertical fin is
!                   on which side of the window
!                   1--right side; 2--left side
!   SunlitArea_Ver - Sunlit area using semi-infinite vertical fin

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Ver
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Sol_Azimuth
REAL(prec2), INTENT(in) :: Sur_Azimuth
REAL(prec2), INTENT(in) :: Win_Length
REAL(prec2), INTENT(in) :: Win_height
REAL(prec2), INTENT(in) :: HangLength_Ver
INTEGER, INTENT(in) :: VerFin_Side

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
! Sunlit rectangle length and height
REAL(prec2) :: VerFin_Azimuth ! The azimuth of the vertical fin
INTEGER :: i

IF (VerFin_Side .EQV. 1) THEN
    VerFin_Azimuth=Sur_Azimuth+1.5707963
ELSE IF (VerFin_Side .EQV. 2) THEN

```

```

        VerFin_Azimuth=Sur_Azimuth-1.5707963
    END IF

    DO i=1, Num_Sol
        IF (Cos_Theta(i)<0) THEN
            SunlitArea_Ver(i)=0
        ELSE IF ((Sur_Azimuth<Pai/2.0) .AND. &
            (Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
            IF (VerFin_Side .EQV. 1) THEN
                Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                    TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
            ELSE IF (VerFin_Side .EQV. 2) THEN
                Sunlit_length(i)=Win_Length
            END IF
            SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

        ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
            (Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
            IF (VerFin_Side .EQV. 1) THEN
                Sunlit_length(i)=Win_Length
            ELSE IF (VerFin_Side .EQV. 2) THEN
                Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                    TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
            END IF
            SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

        ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
            SunlitArea_Ver(i)=0
        ELSE
            IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
                Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                    TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
            ELSE
                Sunlit_length(i)=Win_Length
            END IF
            SunlitArea_Ver(i)=Sunlit_length(i)*Win_height
        END IF
    END DO

END SUBROUTINE SunlitAreaVer

!-----

SUBROUTINE Heat_balance(Cos_Theta, Num_Sol, Altitude, BeamIrradN,Absorp,&
    Transmit,Sur_area,T_outside, T_inside, Hin_A, &
    Hin_C,Hin_n,Hout_A, Hout_C, Hout_n, Flux, &
    Flux_inside, Flux_outside,Flux_Transmit, &
    Flux_Tol,Ts,H_in, H_out,ErrorFlag, ErrorMessage,&
    BeamIrradS,GeoFactor,Load,Peak_load)

    ! PURPOSE:      Calculate analytical response for the case of exterior
    !               solar radiation on glazed surfaces using the method of
    !               heat balance. The convection correlations inside and
    !               outside are specified as  $H = A + C*(T_a - T_s)^n$ .
    !               The steady state heat flux is found using a successive
    !               substitution method to solve the non-linear equation.
    !               A variable number of wall layers can be specified.

```

```

!
! CREATED:
!   11.27.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol      - The number of data lines in the input file
!   Altitude     - Array of solar altitude
!   Cos_Theta    - Array of the cosine of the incident angle
!   BeamIrradN   - Array of beam Irradiance in normal
!   BeamIrradS   - Array of beam irradiation on the
!                   receiving slope
!   GeoFactor     - geometric factor used in calculating
!                   the Beam irradiation
!   Absorp       - Array of absorption of the surface
!   Transmit     - Array of transmittance of the surface
!   Sur_area     - The surface area
!   T_outside    - Outside air temperature
!   T_inside     - Inside air temperature
!   Hin_A        - Conv. correlation coefficient A for inside
!   Hin_C        - Conv. correlation coefficient C for inside
!   Hin_n        - Conv. correlation coefficient n for inside
!   Hout_A       - Conv. correlation coefficient A for outside
!   Hout_C       - Conv. correlation coefficient C for outside
!   Hout_n       - Conv. correlation coefficient n for outside
!   Ts           - Array of Calculated surface temperature
!   Flux_Tal     - Array of Calculated total heat flux
!   Flux         - Array of calculated steady state heat flux
!   Flux_inside  - Array of calculated internal conduction
!                   heat flux
!   Flux_outside - Array of calculated external conduction
!                   heat flux
!   Flux_Transmit - Array of solar beam irradiation
!                   transmitted into the zone
!   Load        - Array of the zone load
!   Load_Shade   - Array of the zone load in considering
!                   of external shading
!   Peak_load    - The peak zone load in one day
!   H_in         - Array of Calculated inside conv. coefficient
!   H_out        - Array of Calculated outside conv. coefficient
!   ErrorFlag    - for error checking
!   ErrorMessage - for error reporting

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 500
REAL(prec2), PARAMETER :: Tolerance = 1.0E-06 ! tolerance

```

```

REAL(prec2), PARAMETER :: Delta_H = 1.0E-20
    ! a small number added to the initial value of convection
    ! coefficient to avoid zero convection coefficients

! arguments
INTEGER                                :: Num_Sol
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: BeamIrradN
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Absorp
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Transmit
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Cos_Theta
REAL(prec2), INTENT(in) :: T_outside, T_inside
REAL(prec2), INTENT(in) :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in) :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(in) :: Sur_area
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: BeamIrradS
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: GeoFactor
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Tol
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_inside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_outside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Transmit

REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Ts
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: H_in, H_out
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Load
REAL(prec2), INTENT(out) :: Peak_load
CHARACTER*(*) , INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out) :: ErrorFlag ! two byte int for VB integer

! local variables
REAL(prec2), DIMENSION(Num_sol) :: MinFlux, MaxFlux
    ! Min and max calculated flux values
INTEGER :: i, j, n, error
CHARACTER(LEN=132) :: dummy
INTEGER, PARAMETER :: DebugUnit = 20
LOGICAL, PARAMETER :: Debug = .false.
LOGICAL, PARAMETER :: Warning = .false.

! loop to calculate the geometric factor and
! the Beam irradiation on the receiving slope
DO j=1, Num_sol

    ! calculate the geometric factor used
    ! in calculating the Beam irradiation
    IF(SIN(Altitude(j))=0.0)THEN
        WRITE(ErrorMessage, '("The sine of the altitude(", &
            I2, " )appears to be zero")') j
        ErrorFlag = 1
        RETURN
    END IF
    GeoFactor(j)=MAX(Cos_Theta(j)/SIN(Altitude(j)), 0.0)

    ! calculate the Beam irradiation on the receiving slope
    IF (Altitude(j)<0) THEN
        BeamIrradS(j)=0
    
```



```

ELSE
    BeamIrradS(j)=GeoFactor(j)*BeamIrradN(j)*SIN(Altitude(j))
END IF
END DO

! start loop to calculate the heat flux and internal &
! external surface temperature
DO j=1, Num_sol

    ! open a unit for debug output
    IF(Debug .and. DebugUnit/=6)THEN
        OPEN (DebugUnit, file="ExtSolRad.log",&
            status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Debug output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF
        ! copy out arguments to debug file
        WRITE(DebugUnit,'("ExtSolRad arguments:")')
        WRITE(DebugUnit,'("Inside convection correlation &
            coefficient A  =",F10.6)') Hin_A
        WRITE(DebugUnit,'("Inside convection correlation &
            coefficient C  =",F10.6)') Hin_C
        WRITE(DebugUnit,'("Inside convection correlation &
            coefficient n  =",F10.6)') Hin_n
        WRITE(DebugUnit,'("Outside convection correlation &
            coefficient A  =",F10.6)') Hout_A
        WRITE(DebugUnit,'("Outside convection correlation &
            coefficient C  =",F10.6)') Hout_C
        WRITE(DebugUnit,'("Outside convection correlation &
            coefficient n  =",F10.6)') Hout_n
        WRITE(DebugUnit,'("Outside air temperature  =", &
            F10.6)') T_outside
        WRITE(DebugUnit,'("Inside air temperature   =", &
            F10.6)') T_inside
    END IF

    ! set initial guess of surface temperatures
    Ts(j) = T_outside - 0.33*(T_outside - T_inside)

    H_in(j) = Hin_A + Hin_C*ABS(T_inside - Ts(j))*Hin_n + Delta_H
    H_out(j) = Hout_A + Hout_C*ABS(T_outside - Ts(j))*Hout_n + Delta_H

    ! start loop to find Flux by successive substitution
    n = 1
    DO
        IF(H_in(j)==0.0 .or. H_out(j)==0.0)THEN
            WRITE(ErrorMessage,'("Inside or outside convection &
                coefficient calculated to be zero"," - can not procede")')
            ErrorFlag = 1
            RETURN
        END IF
    END DO

```

```

END IF

! flux based on current value of total resistance
Flux(j)= ((T_outside - T_inside)+BeamIrradS(j)*Absorp(j)/ &
          H_out(j))/(1/H_in(j)+1/H_out(j))

! update surface temps based on flux
Ts(j) = T_inside + Flux(j)/H_in(j)

! re-calculate conv. coef using current temps
H_in(j) = Hin_A + Hin_C*(ABS(T_inside - Ts(j))**Hin_n)
H_out(j) = Hout_A + Hout_C*(ABS(T_outside - Ts(j))**Hout_n)

! calculate fluxes over different resistances for checking
Flux_inside(j) = H_in(j)*(Ts(j) - T_inside)
Flux_outside(j) =BeamIrradS(j)*Absorp(j)+ &
                 H_out(j)*(T_outside - Ts(j))

! some debug output
IF(Debug)THEN
  WRITE(DebugUnit,'("H_in = ",F8.5," H_out = ",F8.5, &
    " Ts= ",F8.5)') H_in(j), H_out(j), Ts(j)
  WRITE(DebugUnit,'("Q = ",F8.5," Qout = ",F8.5, &
    " Qin = ",F8.5)') Flux(j), Flux_outside(j), &
    Flux_inside(j)
END IF

! calc min and max fluxes
MinFlux(j) = MIN(Flux(j), Flux_inside(j), Flux_outside(j))
MaxFlux(j) = MAX(Flux(j), Flux_inside(j), Flux_outside(j))
! test for exit condition
IF(ABS(MaxFlux(j)-MinFlux(j)) < Tolerance .or. &
  n > Max_Iterations) EXIT
n = n + 1
END DO

! set warning if max iterations exceded
IF(n > Max_Iterations)THEN
  ErrorMessage = "The steady-state heat flux could not &
    be calculated within the allowable number of iterations"
  ErrorFlag = -1
  RETURN
END IF

IF(Debug)THEN
  CLOSE(DebugUnit)
END IF

! Calculate the solar beam irradiation transmitted into the Zone
Flux_Transmit(j)=Transmit(j)*BeamIrradS(j)

!calculate the total flux transfered into the zone
Flux_Tol(j)=Flux(j)+Flux_Transmit(j)

! calculate the zone load
Load(j)=Sur_area*Flux_Tol(j)

```

```
        ! find out the peak zone load in one day
        IF (j==2) THEN
            Peak_load=MAX(Load(j),Load(j-1))
        END IF
        IF (j>2)THEN
            Peak_load=MAX(Load(j),Peak_load)
        END IF
    END DO

END SUBROUTINE Heat_balance

END MODULE WindowReveal
```

Source Code: Module SolRadDistribution

```

MODULE SolRadDistribution

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical      solutions to the internal solar distribution test case.
!
! CREATED:      11.19.99   Dongyi Xiao
!
! MODIFIED:
!

PRIVATE

! make public main subroutine
PUBLIC :: SolRadDist

CONTAINS

!-----
-

SUBROUTINE SolRadDist(SolIrrad,Num_hour,
Sur_Azimuth,Sur_Tilt,FileName,File_in,&
Thickness,Extinc_Coeff,RefIndex_Sur,T_outside,T_inside,&
Hin_A,Hin_C, Hin_n,Hout_A, Hout_C, Hout_n,&
HorFin,VerFin,Setback,VerFin_Side,&
Zone_width,Zone_length,Zone_height,&
Win_Length,Win_height,Coeff_A,Coeff_B,Time_interval,&
ErrorFlag, ErrorMessage,DirName, Length_DirName)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SolRadDist" :: SolRadDist

```

```

! PURPOSE:      Produces weather files and analytical response for the
case
!              of internal solar distribution using the method of heat
balance.
!              The convection correlations inside and outside are
specified as
!               $H = A + C \cdot (T_a - T_s)^n$ .
!              The steady state heat flux is found using a successive
substitution
!              method to solve the non-linear equation.
!
!
!  CREATED:
!    12.10.99    Dongyi Xiao
!
!  MODIFIED:
!
!  INPUT FILES:
!    n/a
!
!  OUTPUT FILES:
!
!    FileName.csv      -    Analytical response data
!
!  SUBROUTINE ARGUMENTS:
!    SolIrrad          -    Array of Beam Irradiance in normal, for each
hour
!    Num_hour          -    number of hours in one day
!    Sur_Azimuth       -    The surface azimuth
!    Sur_Tilt          -    The surface tilt angle
!    FileName         -    Root File Name for weather and data files
!    File_in           -    the index of solar position file
!    Thickness         -    Thicknesses of the surface
!    Extinc_Coeff      -    Extinction coefficient of the surface
!    RefIndex_Air      -    Refractive index of the air
!    RefIndex_Sur      -    Refractive index of the surface
!    T_outside         -    Outside air temperature
!    T_inside          -    Inside air temperature
!    Hin_A             -    Conv. correlation coefficient A for inside
!    Hin_C             -    Conv. correlation coefficient C for inside
!    Hin_n             -    Conv. correlation coefficient n for inside
!    Hout_A            -    Conv. correlation coefficient A for outside
!    Hout_C            -    Conv. correlation coefficient C for outside
!    Hout_n            -    Conv. correlation coefficient n for outside
!    HorFin            -    Logical variable
!                      it is true if using semi-infinite horizontal
fin
!    VerFin            -    Logical variable
!                      it is true if using semi-infinite vertical fin
!    Setback           -    Logical variable, it is true if using both
!                      semi-infinite horizontal and vertical fin
!    VerFin_Side       -    For deciding the vertical fin is on which side
of the window
!    Zone_width        -    Width of the test zone
!    Zone_length       -    Length of the test zone
!    Zone_height       -    Height of the test zone
!    Win_Length        -    Window length
!    Win_height        -    Window height

```

```

!   Coeff_A           -   apparent solar irradiation at air mass equal
zero
!   Coeff_B           -   atmospheric extinction coefficient
!   Time_interval     -   the time interval of the solar position data
!                       -   obtained from the US Naval Observatory (10
min)
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting
!   DirName           -   the directory of the VB interface executive
!                       -   file resides in.
!   Length_DirName    -   the length of the DirName character

```

```

IMPLICIT NONE

```

```

!parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: input=5 ! unit number for input file
INTEGER, PARAMETER :: Results=7 ! unit number for analytical response
output file

```

```

REAL(prec2), PARAMETER :: Pai=3.1415926
REAL(prec2), PARAMETER :: RefIndex_Air=1.

```

```

!arguments
REAL(prec2)           :: Sur_Azimuth
REAL(prec2)           :: Sur_Tilt
REAL(prec2), INTENT(in) :: Extinc_Coeff
REAL(prec2), INTENT(in) :: Thickness, Zone_width
REAL(prec2), INTENT(in) :: Zone_length, Zone_height

```

```

REAL(prec2), INTENT(in) :: RefIndex_Sur
REAL(prec2), INTENT(in) :: Win_Length
REAL(prec2), INTENT(in) :: Win_height
REAL(prec2), INTENT(in) :: Coeff_A
REAL(prec2), INTENT(in) :: Coeff_B
REAL(prec2), INTENT(in) :: Time_interval

```

```

INTEGER, INTENT(in)           :: Length_DirName
CHARACTER (LEN=Length_DirName), INTENT(in) :: DirName

```

```

CHARACTER *(*), INTENT(in) :: FileName
INTEGER, INTENT(in)        :: VerFin_Side
INTEGER, INTENT(in)        :: File_in, Num_hour

```

```

REAL(prec2), INTENT(in) :: T_outside, T_inside
REAL(prec2), INTENT(in) :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in) :: Hout_A, Hout_C, Hout_n

```

```

REAL(prec1), DIMENSION(Num_hour), INTENT(out) :: SolIrrad
CHARACTER*(*), INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out)    :: ErrorFlag ! two byte int for VB integer

```

```

LOGICAL, INTENT(in) :: HorFin
LOGICAL, INTENT(in) :: VerFin
LOGICAL, INTENT(in) :: Setback

```

```

!local variables
INTEGER                                :: i,n,error
INTEGER                                :: Num_Sol  ! The number of data lines in the
input file
INTEGER, DIMENSION(150)                :: N_hour, M_minute ! Array of hour, minute
element of the time
REAL(prec2), DIMENSION(150)            :: BeamIrradN      ! Array of beam Irradiance
in normal
REAL(prec2), DIMENSION(150)            :: BeamIrradS      ! Beam irradiaton on the
receiving slope
REAL(prec2), DIMENSION(150)            :: GeoFactor ! Geometric factor used in
calculating
                                           ! the Beam irradiation
REAL(prec2), DIMENSION(150)            :: Cos_Theta ! Array of the cosine of the
incident angle
REAL(prec2), DIMENSION(150)            :: Altitude      ! Array of solar altitude
REAL(prec2), DIMENSION(150)            :: Sol_Zenith     ! Array of solar zenith
REAL(prec2), DIMENSION(150)            :: Sol_Azimuth    ! Array of solar azimuth
REAL(prec2), DIMENSION(150)            :: Flux ! Array of flux transfered into the
zone by convection
REAL(prec2), DIMENSION(150)            :: Flux_inside ! Array of calculated internal
                                           ! convective heat flux
REAL(prec2), DIMENSION(150)            :: Flux_outside ! Array of calculated external
heat flux
REAL(prec2), DIMENSION(150)            :: Flux_Transmit ! Array of solar beam
irradiation
                                           ! transmitted into the zone
REAL(prec2), DIMENSION(150)            :: Absorp        ! Array of absorption of
the surface
REAL(prec2), DIMENSION(150)            :: Transmit      ! Array of transmittance
of the surface
REAL(prec2), DIMENSION(150)            :: Load
REAL(prec2), DIMENSION(150)            :: LoadShade_HorVer
REAL(prec2), DIMENSION(150)            :: LoadShade_Hor
REAL(prec2), DIMENSION(150)            :: LoadShade_Ver
REAL(prec2), DIMENSION(150)            :: LoadShade_setback
REAL(prec2), DIMENSION(150)            :: Flux_Tol
REAL(prec2), DIMENSION(150)            :: Ts
REAL(prec2), DIMENSION(150)            :: H_in, H_out
REAL(prec2), DIMENSION(150)            :: SunlitArea_HorVer
REAL(prec2), DIMENSION(150)            :: SunlitArea_Hor
REAL(prec2), DIMENSION(150)            :: SunlitArea_Ver
REAL(prec2), DIMENSION(150)            :: SunlitArea_setback

REAL(prec2)                            :: Peak_load
REAL(prec2)                            :: Sur_area
REAL(prec2)                            :: HangLength_Hor
REAL(prec2)                            :: HangLength_Ver

CHARACTER(LEN=360) :: dummy                ! dummy string variable

Sur_area=Win_Length*Win_height
HangLength_Ver=Win_Length*Zone_width*2/(Zone_length-Win_Length)
HangLength_hor=Win_height*Zone_width*2/(Zone_height-Win_height)

```

```

n=Length_DirName
dummy(1:n)=DirName(1:n)

! open the input file
IF(file_in==1) THEN
    dummy(n+1:)="\Atlanta_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==2) THEN
    dummy(n+1:) = "\Atlanta_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==3) THEN
    dummy(n+1:) = "\Boston_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==4) THEN
    dummy(n+1:) = "\Boston_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==5) THEN
    dummy(n+1:) = "\Chicago_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==6) THEN
    dummy(n+1:) = "\Chicago_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==7) THEN
    dummy(n+1:) = "\Los Angeles_June.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
ELSE IF(file_in==8) THEN
    dummy(n+1:) = "\Los Angeles_Aug.txt"
    OPEN (input, file=TRIM(dummy), status='unknown', IOSTAT=error)
END IF
IF(error/=0) THEN
    ErrorMessage="input file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! read the solar position data from the input file
READ(input,*) Num_sol
DO i=1, Num_Sol
    READ(input, *) N_hour(i), M_minute(i), Altitude(i), Sol_Azimuth(i)
END DO
CLOSE(input)

! Transfer the unit from degree to radian and calculate the solar zenith
angle
DO i=1, Num_Sol
    Altitude(i)=Pai*Altitude(i)/180.
    Sol_Zenith(i)=Pai/2.-Altitude(i)
    Sol_Azimuth(i)=Pai*Sol_Azimuth(i)/180.
END DO

! Transfer the unit of surface tilt angle and surface azimuth from degree
to radian
Sur_Azimuth=Sur_Azimuth*Pai/180
Sur_Tilt=Sur_Tilt*Pai/180

! calculate the cosine of the incident angle of the surface

```



```

CALL Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth, Sur_Azimuth,
Sur_Tilt)

! calculate the Beam Irradiance in normal
CALL SolIrrad_ND(file_in,Num_Sol,Altitude,N_hour, M_minute, BeamIrradN,
SolIrrad,&
                Coeff_A,Coeff_B,Time_interval)

! calculate the absorption and transmittance of the surface
CALL Absorp_Transmit(Cos_Theta,
Num_Sol,RefIndex_Air,RefIndex_Sur,Extinc_Coeff,&
                Thickness,Absorp,Transmit)

! calc analytical response
CALL Heat_balance(Cos_Theta, Num_Sol, Altitude,
BeamIrradN,Absorp,Transmit,&
                Sur_area,T_outside, T_inside, Hin_A, Hin_C,Hin_n,&
                Hout_A, Hout_C, Hout_n, Flux,Flux_inside,
Flux_outside, &
                Flux_Transmit,Flux_Tol,Ts,H_in,
H_out,ErrorFlag,ErrorMessage, &
                BeamIrradS,GeoFactor,Load,Peak_load)

! calculate the zone load under semi-infinite horizontal fin shading
IF (HorFin) THEN

    CALL SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth,
Altitude,Cos_Theta,&
                Win_Length,
Win_height,HangLength_Hor,SunlitArea_Hor)
    DO i=1, Num_Sol
        LoadShade_Hor(i)=SunlitArea_Hor(i)*Flux_Tol(i)
    END DO

END IF

! calculate the zone load under semi-infinite vertical fin shading
IF (VerFin) THEN

    CALL SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth,
Altitude,Cos_Theta, &
                Win_Length,
Win_height,HangLength_Ver,VerFin_Side,SunlitArea_Ver)
    DO i=1, Num_Sol
        LoadShade_Ver(i)=SunlitArea_Ver(i)*Flux_Tol(i)
    END DO

END IF

! calculate the zone load under the combination of
! semi-infinite horizontal and semi-infinite vertical fin shading
IF (HorFin .AND. VerFin) THEN

    CALL SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
                Cos_Theta,Win_Length,Win_height,HangLength_Hor, &
                HangLength_Ver,VerFin_Side,SunlitArea_HorVer)

```

```

      DO i=1, Num_Sol
        LoadShade_HorVer(i)=SunlitArea_HorVer(i)*Flux_Tol(i)
      END DO

    END IF

    ! calculate the zone load under the combination of semi-infinite
    ! horizontal and semi-infinite vertical fin shading
    IF (Setback) THEN

      CALL SunlitAreaSetback(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
        Cos_Theta, Win_Length, Win_height, HangLength_Hor, &
        HangLength_Ver, SunlitArea_setback)

      DO i=1, Num_Sol
        LoadShade_setback(i)=SunlitArea_setback(i)*Flux_Tol(i)
      END DO

    END IF

    ! Transfer the units of Sur_Tilt, Sur_Azimuth, Altitude, Sol_Zenith and
    Sol_Azimuth
    ! from radiance to degree for output
    DO i=1, Num_Sol
      Altitude(i)=180.*Altitude(i)/Pai
      Sol_Zenith(i)=180.*Sol_Zenith(i)/Pai
      Sol_Azimuth(i)=180.*Sol_Azimuth(i)/Pai
    END DO
    Sur_Azimuth=Sur_Azimuth*180/Pai
    Sur_Tilt=Sur_Tilt*180/Pai

    IF (Setback) Then

      ! open a result file for window shading
      OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
      IF(error/=0) THEN
        ErrorMessage="Solar distribution result file could not be opened"
        ErrorFlag = 1
        RETURN
      END IF

      ! write header lines
      WRITE(Results, '( "# ASHRAE Analytical Test Suite - Solar Distribution
Result file ")')
      WRITE(Results, '( "# Exterior Solar Radiation - Solar Distribution
test")')
      WRITE(Results, '( //"Test Parameters -")')

      WRITE(Results, '( "Inside convection correlation coefficient A
=", ", ", F15.4)') Hin_A
      WRITE(Results, '( "Inside convection correlation coefficient C
=", ", ", F15.4)') Hin_C
      WRITE(Results, '( "Inside convection correlation coefficient n
=", ", ", F15.4)') Hin_n
      WRITE(Results, '( "Outside convection correlation coefficient A
=", ", ", F15.4)') Hout_A

```

```

        WRITE(Results, '("Outside convection correlation coefficient C
=","",F15.4)') Hout_C
        WRITE(Results, '("Outside convection correlation coefficient n
=","",F15.4)') Hout_n

        WRITE(Results, '("Outside air temperature ="",F15.4)') T_outside
        WRITE(Results, '("Inside air temperature ="",F15.4)') T_inside

        WRITE(Results, '("Fabric layer thickness ="",F15.4)') Thickness
        WRITE(Results, '("Surface Azimuth ="",F15.4)') Sur_Azimuth
        WRITE(Results, '("Surface Tilt angle ="",F15.4)') Sur_Tilt
        WRITE(Results, '("Extinction coefficient of the surface ="",F15.4)')
Extinc_Coeff
        WRITE(Results, '("Refractive index of the surface ="",F15.4)')
RefIndex_Sur

        WRITE(Results, '("Surface Area ="",F15.4)') Sur_area
        WRITE(Results, '("Hanging length of horizontal fin ="",F15.4)') &
        HangLength_Hor
        WRITE(Results, '("Hanging length of vertical fin ="",F15.4)') &
        HangLength_Ver
        WRITE(Results, '("Window length ="",F15.4)') Win_Length
        WRITE(Results, '("Window height ="",F15.4)') Win_height

        WRITE(Results, '("//"Test results -")')

        WRITE(Results, '("//"Time", "", "Sunlit Area", "", "Zone Load")')

        DO i=1, Num_Sol

            WRITE(Results, '(I2.2,":",I2.2, "", F15.4, "", F15.4)')
N_hour(i), M_minute(i),&
                SunlitArea_setback(i), LoadShade_setback(i)

        END DO
        CLOSE (Results)

        ELSE IF (HorFin .OR. VerFin ) Then

            !open a result file for window shading
            OPEN (Results, file="SolRad_Shade.dat", status='unknown',
IOSTAT=error)
            IF(error/=0) THEN
                ErrorMessage="Shading Result file could not be opened"
                ErrorFlag = 1
                RETURN
            END IF

            ! write header lines
            WRITE(Results, '("# ASHRAE Analytical Test Suite - Window Shading
Result file ")')
            WRITE(Results, '("# Exterior Solar Radiation - Window shading test")')
            WRITE(Results, '("//"Test Parameters -")')

```

```

        WRITE(Results, '("Inside convection correlation coefficient A
=","",F15.4)') Hin_A
        WRITE(Results, '("Inside convection correlation coefficient C
=","",F15.4)') Hin_C
        WRITE(Results, '("Inside convection correlation coefficient n
=","",F15.4)') Hin_n
        WRITE(Results, '("Outside convection correlation coefficient A
=","",F15.4)') Hout_A
        WRITE(Results, '("Outside convection correlation coefficient C
=","",F15.4)') Hout_C
        WRITE(Results, '("Outside convection correlation coefficient n
=","",F15.4)') Hout_n

        WRITE(Results, '("Outside air temperature ="",F15.4)') T_outside
        WRITE(Results, '("Inside air temperature ="",F15.4)') T_inside

        WRITE(Results, '("Fabric layer thickness ="",F15.4)') Thickness
        WRITE(Results, '("Surface Azimuth ="",F15.4)') Sur_Azimuth
        WRITE(Results, '("Surface Tilt angle ="",F15.4)') Sur_Tilt
        WRITE(Results, '("Extinction coefficient of the surface ="",F15.4)')
Extinc_Coeff
        WRITE(Results, '("Refractive index of the surface ="",F15.4)')
RefIndex_Sur

!        WRITE(Results, '("Beam Irradiance in normal          ="",F10.2)')
BeamIrradN
        WRITE(Results, '("Surface Area ="",F15.4)') Sur_area
        WRITE(Results, '("Hanging length of horizontal fin ="",F15.4)') &
        HangLength_Hor
        WRITE(Results, '("Hanging length of vertical fin ="",F15.4)') &
        HangLength_Ver
        WRITE(Results, '("Vertical fin side ="",I2)') VerFin_Side
        WRITE(Results, '("Window length ="",F15.4)') Win_Length
        WRITE(Results, '("Window height ="",F15.4)') Win_height

        WRITE(Results, '("//"Test results -")')
        WRITE(Results, '("//"# Explanation of the column labels: ")')
        WRITE(Results, '("# SunlitArea_HorVer - Sunlit area with a union of
semi-infinite &
        horizontal and vertical fin ")')
        WRITE(Results, '("# SunlitArea_Hor - Sunlit area with a semi-infinite
horizontal fin ")')
        WRITE(Results, '("# SunlitArea_Ver - Sunlit area with a semi-infinite
vertical fin ")')
        WRITE(Results, '("# Load - Zone load without shading ")')
        WRITE(Results, '("# LoadShade_HorVer - Zone load with a union of semi-
infinite &
        horizontal and vertical fin ")')
        WRITE(Results, '("# LoadShade_Hor - Zone load with a semi-infinite
horizontal fin ")')
        WRITE(Results, '("# LoadShade_Ver - Zone load with a semi-infinite
vertical fin ")')

        WRITE(Results, '("//"Time", "", "SunlitArea_HorVer", "", "SunlitArea_Hor", "", &

```

```

"SunlitArea_Ver", ",", "Load", ",", "LoadShade_HorVer", ",", "LoadShade_Hor", &
    ",", "LoadShade_Ver")')

    DO i=1, Num_Sol

        WRITE(Results, '(I2.2, ":", I2.2, ", ", F15.4, ", ", F15.4, ", ", F15.4,
", ", F15.4, &
                                ", ", F15.4, ", ", F15.4, ", ", F15.4)') N_hour(i),
M_minute(i), &

SunlitArea_HorVer(i), SunlitArea_Hor(i), SunlitArea_Ver(i), Load(i), &
    LoadShade_HorVer(i), LoadShade_Hor(i),
LoadShade_Ver(i)

        END DO
        CLOSE (Results)

    ELSE
        ! attempt to open file - full path is passed as FileName by VB
        application
        OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Analytical results output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF

        ! write header lines
        WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )')
        WRITE(Results, '( "# Exterior Solar Radiation test" )')
        WRITE(Results, '( "/" Test Parameters -" )')

        WRITE(Results, '( "Inside convection correlation coefficient A
=", ",", F15.4)') Hin_A
        WRITE(Results, '( "Inside convection correlation coefficient C
=", ",", F15.4)') Hin_C
        WRITE(Results, '( "Inside convection correlation coefficient n
=", ",", F15.4)') Hin_n
        WRITE(Results, '( "Outside convection correlation coefficient A
=", ",", F15.4)') Hout_A
        WRITE(Results, '( "Outside convection correlation coefficient C
=", ",", F15.4)') Hout_C
        WRITE(Results, '( "Outside convection correlation coefficient n
=", ",", F15.4)') Hout_n

        WRITE(Results, '( "Outside air temperature =", ",", F15.4)') T_outside
        WRITE(Results, '( "Inside air temperature =", ",", F15.4)') T_inside

        WRITE(Results, '( "Fabric layer thickness =", ",", F15.4)') Thickness
        WRITE(Results, '( "Surface Azimuth =", ",", F15.4)') Sur_Azimuth
        WRITE(Results, '( "Surface Tilt angle =", ",", F15.4)') Sur_Tilt

        ! WRITE(Results, '( "Beam Irradiance in normal          =", F10.2)')
        BeamIrradN
        WRITE(Results, '( "Extinction coefficient of the surface =", ",", F15.4)')
        Extinc_Coeff

```

```

        WRITE(Results, '( "Refractive index of the surface =", "", "", F15.4) ')
RefIndex_Sur
        WRITE(Results, '( "Surface Area =", "", "", F15.4) ') Sur_area

        WRITE(Results, '( // "Test results -" ) ')
        WRITE(Results, '( // "Peak Load =", "", "", F15.4) ') Peak_load

        WRITE(Results, '( // "Time", "", "", "Surface Temp.", "", "", "Inside Convection
Coeff.", "", "", &
                                "Outside Convection Coeff.", "", "", "Incident Solar
Flux", "", "", &
                                "Internal Surface Heat Flux", "", "", "Zone Load" ) ')

        DO i=1, Num_Sol
            WRITE(Results, '( I2.2, ":", I2.2, " ", "", F15.4, " ", "", F15.4, " ", "", F15.4,
                                " ", "", F15.4, " ", "", F15.4) ') N_hour(i),
M_minute(i), Ts(i), H_in(i), &
H_out(i), BeamIrradS(i), Flux_Tol(i), Load(i)
        END DO

        CLOSE(Results)

    END IF

END SUBROUTINE SolRadDist

!-----
-

SUBROUTINE Incid_Angle(Cos_Theta, Num_Sol, Sol_Zenith, Sol_Azimuth,
Sur_Azimuth, Sur_Tilt)

    ! PURPOSE:      Calculate the cosine of the incident angle of the surface
    !
    ! CREATED:
    !   11.7.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Num_Sol      -   The number of data lines in the input file
    !   Cos_Theta    -   Array of the cosine of the incident angle
    !   Sol_Zenith   -   Array of solar zenith
    !   Sol_Azimuth  -   Array of solar azimuth
    !   Sur_Azimuth  -   The surface azimuth
    !   Sur_Tilt     -   The surface tilt angle

    IMPLICIT NONE

```

```

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol)       :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol),INTENT(in) :: Sol_Zenith
REAL(prec2), DIMENSION(Num_Sol),INTENT(in) :: Sol_Azimuth
REAL(prec2), INTENT(in)                :: Sur_Azimuth
REAL(prec2), INTENT(in)                :: Sur_Tilt

! local variables
REAL(prec2) :: SunDirCosA, SunDirCosB, SunDirCosC ! solar direction
cosines
REAL(prec2) :: SurDirCosA, SurDirCosB, SurDirCosC ! surface direction
cosines
INTEGER      :: i

DO i=1, Num_Sol

    ! calculate the solar direction cosines
    SunDirCosA=SIN(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosB=COS(Sol_Azimuth(i))*SIN(Sol_Zenith(i))
    SunDirCosC=COS(Sol_Zenith(i))

    ! calculate the surface direction cosines
    SurDirCosA=SIN(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosB=COS(Sur_Azimuth)*SIN(Sur_Tilt)
    SurDirCosC=COS(Sur_Tilt)

    ! calculate the cosine of incidence angle Theta
    Cos_Theta(i)=SunDirCosA*SurDirCosA+SunDirCosB*SurDirCosB+SunDirCosC*SurDirCosC

END DO

END SUBROUTINE Incid_Angle

!-----
-
SUBROUTINE SolIrrad_ND(file_in,Num_Sol,Altitude,N_hour, M_minute,
BeamIrradN, SolIrrad,&
                        Coeff_A,Coeff_B,Time_interval)

! PURPOSE:      Calculate the Beam Irradiance in normal
!
! CREATED:
!   11.7.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:

```

```

!   n/a
!
! SUBROUTINE ARGUMENTS:
!   File_in           -   the index of solar position file
!   Num_Sol           -   The number of data lines in the input file
!   Altitude          -   Array of solar altitude
!   N_hour            -   Array of hour element of the time
!   M_minute          -   Array of minute element of the time
!   BeamIrradN        -   Array of Beam Irradiance in normal,
!                           for time intervals like 10 minutes
!   SolIrrad          -   Array of Beam Irradiance in normal, for each
hour
!   Coeff_A           -   apparent solar irradiation at air mass equal
zero
!   Coeff_B           -   atmospheric extinction coefficient
!   Time_interval     -   the time interval of the solar position data
!                           obtained from the US Naval Observatory (10
min)

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol, file_in
REAL(prec2), DIMENSION(Num_Sol) :: Altitude
REAL(prec2), DIMENSION(Num_Sol) :: BeamIrradN
INTEGER, DIMENSION(Num_Sol)      :: N_hour, M_minute
REAL(prec1), DIMENSION(24)       :: SolIrrad

! local variables
REAL(prec2), INTENT(in) :: Coeff_A
REAL(prec2), INTENT(in) :: Coeff_B
REAL(prec2), INTENT(in) :: Time_interval

INTEGER      :: i, j

DO i=1, 24
    SolIrrad(i)=0.0
END DO

DO i=1, Num_Sol

    ! calculate the Beam Irradiation in normal
    ! (W/m2 for SI units or BTU/(h.ft2) for IP units)
    BeamIrradN(i)=Coeff_A/EXP(Coeff_B/SIN(Altitude(i)))
    ! calculate the Beam Irradiance in normal in one hour
    ! (J/m2 for SI units or BTU/ft2 for IP units) (will be used for
weather data)

    SolIrrad(N_hour(i)+1)=SolIrrad(N_hour(i)+1)+BeamIrradN(i)*Time_interval

END DO

```



```

END SUBROUTINE SolIrrad_ND

!-----
-

SUBROUTINE Absorp_Transmit(Cos_Theta,
Num_Sol,RefIndex_Air,RefIndex_Sur,Extinc_Coeff,&
                        Thickness,Absorp,Transmit)

! PURPOSE:      Calculate the absorption and transmittance of the surface
!
! CREATED:
!   11.19.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Cos_Theta      -   Array of the cosine of the incident angle
!   Num_Sol         -   The number of data lines in the input file
!   RefIndex_Air    -   Refractive index of the air
!   RefIndex_Sur    -   Refractive index of the surface
!   Extinc_Coeff    -   Extinction coefficient of the surface
!   Thickness       -   Thickness of the surface
!   Absorp          -   Array of absorption of the surface
!   Transmit        -   Array of transmittance of the surface

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol),INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol),INTENT(out)  :: Absorp
REAL(prec2), DIMENSION(Num_Sol),INTENT(out)  :: Transmit
REAL(prec2), INTENT(in)                      :: Extinc_Coeff
REAL(prec2), INTENT(in)                      :: Thickness
REAL(prec2), INTENT(in)                      :: RefIndex_Air
REAL(prec2), INTENT(in)                      :: RefIndex_Sur

! local variables
REAL(prec2) :: Theta_1      ! Incidence angle of the surface
REAL(prec2) :: Theta_2      ! Refractive angle of the surface
REAL(prec2) :: Transmit_A   !Transmittance when only absorption loss has
been considered
REAL(prec2) :: Refl_Per     ! Reflectance for perpenticular component of
polarization

```

```

      REAL(prec2) :: Refl_Par      ! Reflectance for parallel component of
polarization
      REAL(prec2) :: Absorp_Per    ! Absorption for perpendicular component of
polarization
      REAL(prec2) :: Absorp_Par    ! Absorption for parallel component of
polarization
      REAL(prec2) :: Transmit_Per ! Transmittance for perpendicular component of
polarization
      REAL(prec2) :: Transmit_Par ! Transmittance for parallel component of
polarization
      INTEGER      :: i

      DO i=1, Num_Sol

        IF (Cos_Theta(i)<0) THEN
          Absorp(i)=0
          Transmit(i)=0
        else

          Theta_1=ACOS(Cos_Theta(i))
          Theta_2=ASIN(RefIndex_Air/RefIndex_Sur*SIN(Theta_1))

          Transmit_A=EXP(-Extinc_Coeff*Thickness/COS(Theta_2))
          Refl_Per=(SIN(Theta_2-Theta_1))**2/(SIN(Theta_2+Theta_1))**2
          Refl_Par=(TAN(Theta_2-Theta_1))**2/(TAN(Theta_2+Theta_1))**2

          Absorp_Per=(1-Transmit_A)*(1-Refl_Per)/(1+Refl_Per*Transmit_A)
          Absorp_Par=(1-Transmit_A)*(1-Refl_Par)/(1+Refl_Par*Transmit_A)
          Transmit_Per=Transmit_A*(1-Refl_Per)**2/(1-
(Refl_Per*Transmit_A)**2)
          Transmit_Par=Transmit_A*(1-Refl_Par)**2/(1-
(Refl_Par*Transmit_A)**2)

          Absorp(i)=(Absorp_Per+Absorp_Par)/2.
          Transmit(i)=(Transmit_Per+Transmit_Par)/2.

        END IF
      END DO

      END SUBROUTINE Absorp_Transmit

!-----
-

      SUBROUTINE SunlitAreaSetback(Num_Sol, Sol_Azimuth, Sur_Azimuth, Altitude, &
          Cos_Theta,Win_Length, Win_height, HangLength_Hor, &
          HangLength_Ver,SunlitArea_setback)

      ! PURPOSE:      Calculate the new sunlit area in considering of the
exteranl shading of
      !                  the glazed surface
      !
      ! CREATED:
      !   12.10.99    Dongyi Xiao
      !
      ! MODIFIED:
      !

```

```

! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol           - The number of data lines in the input file
!   Sol_Azimuth       - Array of solar azimuth
!   Sur_Azimuth       - The surface azimuth
!   Altitude          - Array of solar altitude
!   Cos_Theta         - Array of the cosine of the incident angle
!   Win_Length        - Window length
!   Win_height        - Window height
!   HangLength_Hor     - Hanging length of the horizontal fin
!   HangLength_Ver     - Hanging length of the vertical fin
!   SunlitArea_setback - Sunlit area using setback window shading
strategy

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) ::
SunlitArea_setback
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)    :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)    :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)    :: Sol_Azimuth
REAL(prec2), INTENT(in)                       :: Sur_Azimuth
REAL(prec2), INTENT(in)                       :: Win_Length
REAL(prec2), INTENT(in)                       :: Win_height
REAL(prec2), INTENT(in)                       :: HangLength_Hor
REAL(prec2), INTENT(in)                       :: HangLength_Ver

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
! Sunlit rectangle length and height
INTEGER :: i

DO i=1, Num_Sol
  IF (Cos_Theta(i)<0) THEN
    SunlitArea_setback(i)=0
  ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
    SunlitArea_setback(i)=0
  ELSE
    Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
      TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
      TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-
Sur_Azimuth)))
    SunlitArea_setback(i)=Sunlit_length(i)*Sunlit_height(i)
  
```

```

        END IF
    END DO

END SUBROUTINE SunlitAreaSetback

!-----
-

SUBROUTINE SunlitAreaHorVer(Num_Sol, Sol_Azimuth, Sur_Azimuth,
Altitude,Cos_Theta, &
                        Win_Length,
Win_height, HangLength_Hor, HangLength_Ver, &
                        VerFin_Side, SunlitArea_HorVer)

    ! PURPOSE:      Calculate the new sunlit area in considering of the
    !               exteranl shading of
    !               the glazed surface
    !
    ! CREATED:
    !   12.10.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Num_Sol      -   The number of data lines in the input file
    !   Sol_Azimuth   -   Array of solar azimuth
    !   Sur_Azimuth   -   The surface azimuth
    !   Altitude      -   Array of solar altitude
    !   Cos_Theta     -   Array of the cosine of the incident angle
    !   Win_Length    -   Window length
    !   Win_height    -   Window height
    !   HangLength_Hor -   Hanging lenth of the horizontal fin
    !   HangLength_Ver -   Hanging lenth of the vertical fin
    !   SunlitArea_HorVer -   Sunlit area using semi-infinite horizontal and
    !               vertical fin
    !   VerFin_Side   -   For deciding the vertical fin is on which side
    !               of the window
    !               1--right side; 2--left side

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer :: Num_Sol

```

```

      REAL(prec2), DIMENSION(Num_Sol), INTENT(out)           ::
SunlitArea_HorVer
      REAL(prec2), DIMENSION(Num_Sol), INTENT(in)           :: Altitude
      REAL(prec2), DIMENSION(Num_Sol), INTENT(in)           :: Cos_Theta
      REAL(prec2), DIMENSION(Num_Sol), INTENT(in)           :: Sol_Azimuth
      REAL(prec2), INTENT(in)                                :: Sur_Azimuth
      REAL(prec2), INTENT(in)                                :: Win_Length
      REAL(prec2), INTENT(in)                                :: Win_height
      REAL(prec2), INTENT(in)                                :: HangLength_Hor
      REAL(prec2), INTENT(in)                                :: HangLength_Ver
      INTEGER, INTENT(in)                                     :: VerFin_Side

      ! local variables
      REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                     ! Sunlit rectangle length and height
      REAL(prec2)                       :: VerFin_Azimuth ! The azimuth of the
vertical fin
      INTEGER                           :: i

      IF (VerFin_Side .EQV. 1) THEN
        VerFin_Azimuth=Sur_Azimuth+1.5707963
      ELSE IF (VerFin_Side .EQV. 2) THEN
        VerFin_Azimuth=Sur_Azimuth-1.5707963
      END IF

      DO i=1, Num_Sol
        IF (Cos_Theta(i)<0) THEN
          SunlitArea_HorVer(i)=0
        ELSE IF ((Sur_Azimuth<Pai/2.0) .AND.
(Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
          IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
          ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=Win_Length
          END IF
          Sunlit_height(i)=MAX(0.0, Win_height-
HangLength_Hor*TAN(Altitude(i))/ &
                                Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
          SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

        ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
(Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
          IF (VerFin_Side .EQV. 1) THEN
            Sunlit_length(i)=Win_Length
          ELSE IF (VerFin_Side .EQV. 2) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
          END IF
          Sunlit_height(i)=MAX(0.0, Win_height-
HangLength_Hor*TAN(Altitude(i))/ &
                                Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
          SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)

        ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
          SunlitArea_HorVer(i)=0
        ELSE

```

```

        IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
            Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
        ELSE
            Sunlit_length(i)=Win_Length
        END IF
        Sunlit_height(i)=MAX(0.0, Win_height-HangLength_Hor* &
                             TAN(Altitude(i))/Cos(ABS(Sol_Azimuth(i)-
Sur_Azimuth)))
        SunlitArea_HorVer(i)=Sunlit_length(i)*Sunlit_height(i)
    END IF
END DO

END SUBROUTINE SunlitAreaHorVer

!-----
-

SUBROUTINE SunlitAreaHor(Num_Sol, Sol_Azimuth, Sur_Azimuth,
Altitude,Cos_Theta,Win_Length,&
                        Win_height,HangLength_Hor,SunlitArea_Hor)

    ! PURPOSE:      Calculate the new sunlit area in considering of the
exteranl shading of
    !                the glazed surface
    !
    ! CREATED:
    !   12.10.99      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Num_Sol       - The number of data lines in the input file
    !   Sol_Azimuth   - Array of solar azimuth
    !   Sur_Azimuth   - The surface azimuth
    !   Altitude      - Array of solar altitude
    !   Cos_Theta     - Array of the cosine of the incident angle
    !   Win_Length    - Window length
    !   Win_height    - Window height
    !   HangLength_Hor - Hanging lenth of the horizontal fin
    !   SunlitArea_Hor - Sunlit area using semi-infinite horizontal fin

    IMPLICIT NONE

    ! parameters
    INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
    INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
    REAL(prec2), PARAMETER :: Pai=3.1415926

    ! arguments

```

```

integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Hor
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
REAL(prec2), INTENT(in)                     :: Sur_Azimuth
REAL(prec2), INTENT(in)                     :: Win_Length
REAL(prec2), INTENT(in)                     :: Win_height
REAL(prec2), INTENT(in)                     :: HangLength_Hor

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
                                ! Sunlit rectangle length and height

INTEGER      :: i

DO i=1, Num_Sol
  IF (Cos_Theta(i)<0) THEN
    SunlitArea_Hor(i)=0
  ELSE IF ((Sur_Azimuth<Pai/2.0) .AND.
(Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-
HangLength_Hor*TAN(Altitude(i))/ &
                                Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
(Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
    Sunlit_height(i)=MAX(0.0, Win_height-
HangLength_Hor*TAN(Altitude(i))/ &
                                Cos(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)

  ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
    SunlitArea_Hor(i)=0
  ELSE
    Sunlit_height(i)=MAX(0.0, Win_height-
HangLength_Hor*TAN(Altitude(i))/ &
                                Cos(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
    SunlitArea_Hor(i)=Win_Length*Sunlit_height(i)
  END IF
END DO

END SUBROUTINE SunlitAreaHor

!-----
-

SUBROUTINE SunlitAreaVer(Num_Sol, Sol_Azimuth, Sur_Azimuth,
Altitude,Cos_Theta,Win_Length,&
Win_height, HangLength_Ver, VerFin_Side, SunlitArea_Ver)

! PURPOSE:      Calculate the new sunlit area in considering of the
exteranl shading of
!               the glazed surface
!
! CREATED:
!   12.10.99      Dongyi Xiao

```

```

!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Num_Sol          - The number of data lines in the input file
!   Sol_Azimuth      - Array of solar azimuth
!   Sur_Azimuth      - The surface azimuth
!   Altitude         - Array of solar altitude
!   Cos_Theta        - Array of the cosine of the incident angle
!   Win_Length       - Window length
!   Win_height       - Window height
!   HangLength_Ver   - Hanging length of the vertical fin
!   VerFin_Side      - For deciding the vertical fin is on which side
of the window
!                   1--right side; 2--left side
!   SunlitArea_Ver   - Sunlit area using semi-infinite vertical fin

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL(prec2), PARAMETER :: Pai=3.1415926

! arguments
integer                                :: Num_Sol
REAL(prec2), DIMENSION(Num_Sol), INTENT(out) :: SunlitArea_Ver
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Cos_Theta
REAL(prec2), DIMENSION(Num_Sol), INTENT(in)  :: Sol_Azimuth
REAL(prec2), INTENT(in)                     :: Sur_Azimuth
REAL(prec2), INTENT(in)                     :: Win_Length
REAL(prec2), INTENT(in)                     :: Win_height
REAL(prec2), INTENT(in)                     :: HangLength_Ver
INTEGER, INTENT(in)                         :: VerFin_Side

! local variables
REAL(prec2), DIMENSION(Num_Sol) :: Sunlit_length, Sunlit_height
! Sunlit rectangle length and height
REAL(prec2) :: VerFin_Azimuth ! The azimuth of the
vertical fin
INTEGER :: i

IF (VerFin_Side .EQV. 1) THEN
  VerFin_Azimuth=Sur_Azimuth+1.5707963
ELSE IF (VerFin_Side .EQV. 2) THEN
  VerFin_Azimuth=Sur_Azimuth-1.5707963
END IF

DO i=1, Num_Sol

```



```

      IF (Cos_Theta(i)<0) THEN
        SunlitArea_Ver(i)=0
      ELSE IF ((Sur_Azimuth<Pai/2.0) .AND.
(Sol_Azimuth(i)>(Sur_Azimuth+Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
          Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth+2*Pai))))
        ELSE IF (VerFin_Side .EQV. 2) THEN
          Sunlit_length(i)=Win_Length
        END IF
        SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

      ELSE IF ((Sur_Azimuth>Pai*3.0/2.0) .AND. &
(Sol_Azimuth(i)<(Sur_Azimuth-Pai*3.0/2.0))) THEN
        IF (VerFin_Side .EQV. 1) THEN
          Sunlit_length(i)=Win_Length
        ELSE IF (VerFin_Side .EQV. 2) THEN
          Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-(Sur_Azimuth-2*Pai))))
        END IF
        SunlitArea_Ver(i)=Sunlit_length(i)*Win_height

      ELSE IF (ABS(Sol_Azimuth(i)-Sur_Azimuth)>1.5707963) THEN
        SunlitArea_Ver(i)=0
      ELSE
        IF (ABS(VerFin_Azimuth-Sol_Azimuth(i))>1.5707963) THEN
          Sunlit_length(i)=MAX(0.0, Win_Length-HangLength_Ver* &
                                TAN(ABS(Sol_Azimuth(i)-Sur_Azimuth)))
        ELSE
          Sunlit_length(i)=Win_Length
        END IF
        SunlitArea_Ver(i)=Sunlit_length(i)*Win_height
      END IF
    END DO

  END SUBROUTINE SunlitAreaVer

!-----
-

  SUBROUTINE Heat_balance(Cos_Theta, Num_Sol, Altitude,
BeamIrradN,Absorp,Transmit,&
                        Sur_area,T_outside, T_inside, Hin_A, Hin_C,Hin_n,&
                        Hout_A, Hout_C, Hout_n, Flux,Flux_inside,
Flux_outside, &
                        Flux_Transmit,Flux_Tol,Ts,H_in, H_out,ErrorFlag,
ErrorMessage,&
                        BeamIrradS,GeoFactor,Load,Peak_load)

    ! PURPOSE:      Calculate analytical response for the case of exterior
solar
    !               radiaton on glazed surfaces using the method of heat
balance.
    !               The convection correlations inside and outside are
specified
    !               as  $H = A + C*(T_a - T_s)^n$ . The steady state heat flux is
found

```

```

!           using a successive substitution method to solve the non-
linear
!           equation. A variable number of wall layers can be
specified.
!
!   CREATED:
!   11.27.99      S.J.Rees, Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!   n/a
!
!   SUBROUTINE ARGUMENTS:
!   Num_Sol      -   The number of data lines in the input file
!   Altitude     -   Array of solar altitude
!   Cos_Theta    -   Array of the cosine of the incident angle
!   BeamIrradN   -   Array of beam Irradiance in normal
!   BeamIrradS   -   Array of beam irradiation on the receiving
slope
!   GeoFactor    -   geometric factor used in calculating the Beam
irradiation
!   Absorp      -   Array of absorption of the surface
!   Transmit    -   Array of transmittance of the surface
!   Sur_area    -   The surface area
!   T_outside   -   Outside air temperature
!   T_inside    -   Inside air temperature
!   Hin_A       -   Conv. correlation coefficient A for inside
!   Hin_C       -   Conv. correlation coefficient C for inside
!   Hin_n       -   Conv. correlation coefficient n for inside
!   Hout_A      -   Conv. correlation coefficient A for outside
!   Hout_C      -   Conv. correlation coefficient C for outside
!   Hout_n      -   Conv. correlation coefficient n for outside
!   Ts          -   Array of Calculated surface temperature
!   Flux_Tal    -   Array of Calculated total heat flux
!   Flux        -   Array of calculated steady state heat flux
!   Flux_inside -   Array of calculated internal conduction heat
flux
!   Flux_outside -   Array of calculated external conduction heat
flux
!   Flux_Transmit -   Array of solar beam irradiation transmitted
into the zone
!   Load       -   Array of the zone load
!   Load_Shade  -   Array of the zone load in considering of
externanl sahding
!   Peak_load   -   The peak zone load in one day
!   H_in        -   Array of Calculated inside conv. coefficient
!   H_out       -   Array of Calculated outside conv. coefficient
!   ErrorFlag   -   for error checking
!   ErrorMessage -   for error reporting

IMPLICIT NONE

! parameters

```

```

INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 500
REAL(prec2), PARAMETER :: Tolerance = 1.0E-06 ! tolerance
REAL(prec2), PARAMETER :: Delta_H = 1.0E-20
                                ! a small number added to the initial value of
convection
                                ! coefficient to avoid zero convection
coefficients

! arguments
INTEGER                                :: Num_Sol
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: BeamIrradN
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Absorp
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Transmit
REAL(prec2), DIMENSION(Num_sol), INTENT(in) :: Altitude
REAL(prec2), DIMENSION(Num_Sol), INTENT(in) :: Cos_Theta
REAL(prec2), INTENT(in) :: T_outside, T_inside
REAL(prec2), INTENT(in) :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(in) :: Hout_A, Hout_C, Hout_n
REAL(prec2), INTENT(in) :: Sur_area
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: BeamIrradS
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: GeoFactor
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Tol
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_inside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_outside
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Flux_Transmit

REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Ts
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: H_in, H_out
REAL(prec2), DIMENSION(Num_sol), INTENT(out) :: Load
REAL(prec2), INTENT(out) :: Peak_load
CHARACTER*(*) , INTENT(out) :: ErrorMessage
INTEGER(2), INTENT(out) :: ErrorFlag ! two byte int for VB integer

! local variables
REAL(prec2), DIMENSION(Num_sol) :: MinFlux, MaxFlux ! Min and max
calculated flux values
INTEGER                                :: i, j, n, error
CHARACTER(LEN=132) :: dummy
INTEGER, PARAMETER :: DebugUnit = 20
LOGICAL, PARAMETER :: Debug = .false.
LOGICAL, PARAMETER :: Warning = .false.

! loop to calculate the geometric factor and the Beam irradiation on the
receiving slope
DO j=1, Num_sol

! calculate the geometric factor used in calculating the Beam
irradiation
IF(SIN(Altitude(j))=0.0)THEN
WRITE(ErrorMessage, '( "The sine of the altitude(", I2, " )appears to
be zero")' ) j
ErrorFlag = 1

```

```

        RETURN
    END IF
    GeoFactor(j)=MAX(Cos_Theta(j)/SIN(Altitude(j)), 0.0)

    ! calculate the Beam irradiation on the receiving slope
    IF (Altitude(j)<0) THEN
        BeamIrradS(j)=0
    ELSE
        BeamIrradS(j)=GeoFactor(j)*BeamIrradN(j)*SIN(Altitude(j))
    END IF
END DO

! start loop to calculate the heat flux and internal & external surface
temperature
DO j=1, Num_sol

    ! open a unit for debug output
    IF(Debug .and. DebugUnit/=6)THEN
        OPEN (DebugUnit, file="ExtSolRad.log", status='unknown',
IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Debug output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF
        ! copy out arguments to debug file
        WRITE(DebugUnit,'("ExtSolRad arguments:")')
        WRITE(DebugUnit,'("Inside convection correlation coefficient A
=" ,F10.6)') Hin_A
        WRITE(DebugUnit,'("Inside convection correlation coefficient C
=" ,F10.6)') Hin_C
        WRITE(DebugUnit,'("Inside convection correlation coefficient n
=" ,F10.6)') Hin_n
        WRITE(DebugUnit,'("Outside convection correlation coefficient A
=" ,F10.6)') Hout_A
        WRITE(DebugUnit,'("Outside convection correlation coefficient C
=" ,F10.6)') Hout_C
        WRITE(DebugUnit,'("Outside convection correlation coefficient n
=" ,F10.6)') Hout_n
        WRITE(DebugUnit,'("Outside air temperature
=" , &
                                F10.6)') T_outside
        WRITE(DebugUnit,'("Inside air temperature
=" , &
                                F10.6)') T_inside
    END IF

    ! set initial guess of surface temperatures
    Ts(j) = T_outside - 0.33*(T_outside - T_inside)

    H_in(j) = Hin_A + Hin_C*ABS(T_inside - Ts(j))*Hin_n + Delta_H
    H_out(j) = Hout_A + Hout_C*ABS(T_outside - Ts(j))*Hout_n + Delta_H

    ! start loop to find Flux by successive substitution

```

```

n = 1
DO
  IF(H_in(j)==0.0 .or. H_out(j)==0.0)THEN
    WRITE(ErrorMessage,'("Inside or outside convection coefficient
&
                                calculated to be zero", " - can not
procede"))')
    ErrorFlag = 1
    RETURN
  END IF

  ! flux based on current value of total resistance
  Flux(j)= ((T_outside - T_inside)+BeamIrradS(j)*Absorp(j)/ &
    H_out(j))/(1/H_in(j)+1/H_out(j))

  ! update surface temps based on flux
  Ts(j) = T_inside + Flux(j)/H_in(j)

  ! re-calculate conv. coef using current temps
  H_in(j) = Hin_A + Hin_C*(ABS(T_inside - Ts(j))**Hin_n)
  H_out(j) = Hout_A + Hout_C*(ABS(T_outside - Ts(j))**Hout_n)

  ! calculate fluxes over different resistances for checking
  Flux_inside(j) = H_in(j)*(Ts(j) - T_inside)
  Flux_outside(j) =BeamIrradS(j)*Absorp(j)+ H_out(j)*(T_outside -
Ts(j))

  ! some debug output
  IF(Debug)THEN
    WRITE(DebugUnit,'("H_in = ",F8.5," H_out = ",F8.5," Ts=
",F8.5)') &
    H_in(j), H_out(j), Ts(j)
    WRITE(DebugUnit,'("Q = ",F8.5," Qout = ",F8.5," Qin =",F8.5)')
&
    Flux(j), Flux_outside(j), Flux_inside(j)
  END IF

  ! calc min and max fluxes
  MinFlux(j) = MIN(Flux(j), Flux_inside(j), Flux_outside(j))
  MaxFlux(j) = MAX(Flux(j), Flux_inside(j), Flux_outside(j))
  ! test for exit condition
  IF(ABS(MaxFlux(j)-MinFlux(j)) < Tolerance .or. n > Max_Iterations)
EXIT
    n = n + 1
  END DO

  ! set warning if max iterations exceded
  IF(n > Max_Iterations)THEN
    ErrorMessage = "The steady-state heat flux could not be calculated
&
                                within the allowable number of iterations"
    ErrorFlag = -1
    RETURN
  END IF

  IF(Debug)THEN
    CLOSE(DebugUnit)

```

```
END IF

! Calculate the solar beam irradiation transmitted into the Zone
Flux_Transmit(j)=Transmit(j)*BeamIrradS(j)

!calculate the total flux transfered into the zone
Flux_Tol(j)=Flux(j)+Flux_Transmit(j)

! calculate the zone load
Load(j)=Sur_area*Flux_Tol(j)

! find out the peak zone load in one day
IF (j==2) THEN
    Peak_load=MAX(Load(j),Load(j-1))
END IF
IF (j>2)THEN
    Peak_load=MAX(Load(j),Peak_load)
END IF
END DO

END SUBROUTINE Heat_balance

END MODULE SolRadDistribution
```

Source Code: Module IntRad_1

```

MODULE IntRad_1

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical      solutions to the internal long wave radiation test case.
!
! CREATED:      9.3.00      Dongyi Xiao
!
! MODIFIED:
!

USE MatrixInverse

PRIVATE

! make public main subroutine
PUBLIC :: IntRadiation

CONTAINS

!-----
-

SUBROUTINE  IntRadiation(FileName,Ratio_L,H_in,H_out,T_outside,
T_inside,Emissivity,&
                        Const_SB, Width, IPUnit, ErrorMessage, ErrorFlag)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "IntRadiation" :: IntRadiation

! PURPOSE:      Produces analytical response for the test case of interior
radiation
!               using exact method. A linpack subroutine is used to do
matrix inverse

```

```

!               and the nonlinear equations in the analytical solution are
solved by
!               Newton-Raphson method.
!
!   CREATED:
!   9.3.00      Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!
!   FileName.csv          -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   FileName              -   Root File Name for data files
!   Ratio_L               -   The aspect ratio of length to width
!   H_in                  -   Internal convection heat transfer coefficient
!   H_out                 -   External convection heat transfer coefficient
!   T_inside              -   Internal air temperature
!   T_outside             -   External air temperature
!   Emissivity            -   Array of surface emmisivity
!   Const_SB              -   Stefan-Boltzman constant,
!                           =5.6697*10.0**(-8) in SI; =0.1714*10.0**(-8)
in IP
!   Width                 -   The width of the cuboid zone
!   IPUnit                -   logical variable, is true if user specify IP
unit
!   ErrorFlag             -   for error checking
!   ErrorMessage          -   for error reporting

IMPLICIT NONE

!parameters
INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)

INTEGER, PARAMETER      :: Results=7      ! unit number for analytical
response output file

INTEGER, PARAMETER      :: NumVar=15      ! A parameter used in the
subroutine of solving                                     ! nonlinear equations, up to
NumVar variables

INTEGER, PARAMETER      :: N=6           ! Find a root in N dimention
INTEGER, PARAMETER      :: NStep=100
! Take NStep Newton-Raphson steps to improve
the root

REAL, PARAMETER         :: SUMX=1.0E-10 ! Summed variable increments
REAL, PARAMETER         :: SUMF=1.0E-10 ! Summed function increments

```



```

!Real(prec2),PARAMETER  :: Const_SB=5.6697*10.0**(-8) ! Stefan-Boltzman
constant

!arguments
REAL(prec1), INTENT(in) :: Const_SB
REAL(prec1), INTENT(in) :: Width

REAL(prec1), INTENT(in) :: Ratio_L
REAL(prec1), INTENT(in) :: H_in
REAL(prec1), INTENT(in) :: H_out
REAL(prec1), INTENT(in) :: T_outside
REAL(prec1), INTENT(in) :: T_inside
REAL(prec1), DIMENSION(6), INTENT(in) :: Emissivity

CHARACTER *(*), INTENT(in)  :: FileName
CHARACTER*(*), INTENT(out)  :: ErrorMessage
INTEGER(2), INTENT(out)     :: ErrorFlag ! two byte int for VB integer
LOGICAL, INTENT(in)        :: IPUnit

!local variables

! Variables used in linpack matrix inverse subroutine:
INTEGER                :: JOB
INTEGER, DIMENSION(6) :: IPVT
REAL, DIMENSION(2)    :: DET
REAL, DIMENSION(6)    :: WORK,Z
REAL                  :: RCOND

! Variables used in the internal radiation analytical solution:
INTEGER                :: i,k,j,error
REAL(prec1), DIMENSION(6,6) :: f ! The array of view
factors
REAL(prec1)                :: Length,Height ! The length and
height of the cuboid
REAL(prec1)                :: Toltal_Q
! The sum of radiative energy on the
internal surfaces
REAL(prec1), DIMENSION(6) :: Area ! The array of surface area
REAL(prec1), DIMENSION(6) :: Q ! The array of net radiative energy
!on the internal surfaces
REAL(prec1), DIMENSION(6) :: C ! The array used in calculation of
Q
REAL(prec1), DIMENSION(6,6) :: B ! The square 6X6 matrix used in
calculation of Q
REAL(prec1), DIMENSION(6,6) :: Delta ! Kronecker's delta,=1 if k=j, =0
otherwise
REAL(prec1), DIMENSION(NumVar) :: T
REAL(prec1)                    :: Ratio_H
REAL(prec1), DIMENSION(6)      :: Flux_conv
REAL(prec1), DIMENSION(6)      :: Flux_rad
REAL(prec1)                    :: Load

Ratio_H = 1.0

! set initial guess for surface temperatures
DO i=1,6

```

```

      T(i)=(T_outside+T_inside)/2.0
END DO

! Calculate the length, height of the cuiboid zone
Length=Width*Ratio_L
Height=Width*Ratio_H

! Calculate the surface area of the cuiboid zone
Area(1)=Height*Width
Area(2)=Area(1)
Area(3)=Height*Length
Area(4)=Area(3)
Area(5)=Width*Length
Area(6)=Area(5)

! Calculate the view factors
CALL Viewfactor(Width,Ratio_L,Ratio_H,f)

! Calculate the matrix B
DO k=1,6
  DO j=1,6
    IF (k==j) THEN
      Delta(k,j)=1.0
    ELSE
      Delta(k,j)=0.0
    END IF
    B(k,j)=(Delta(k,j)/Emissivity(j)-f(k,j)*(1-
Emissivity(j))/Emissivity(j))/Area(j)
  END DO
END DO

! Solve the inverse of matrix B
JOB=01
CALL SGECO(B,6,6,IPVT,RCOND,Z)
CALL SGEDI(B,6,6,IPVT,DET,WORK,JOB)

! Solve the non-linear equations to find surface temperature
CALL NRNonlinear(NStep,T,N,SUMX,SUMF,B,Area,Delta,f,H_in,H_out, &
      T_outside,T_inside,Const_SB)

! Calculate the matrix C
DO k=1,6
  C(k)=0.0
  DO j=1,6
    C(k)=C(k)+(Delta(k,j)-f(k,j))*Const_SB*T(j)**4
  END DO
END DO

! Calculate the radiative energy on each surface
Q=MATMUL(B,C)

! calculate the radiative and convective heat flux on each surface and the
zone load
Load=0.0

```

```

Toltal_Q=0.0

DO K=1,6

    Flux_rad(k)=Q(k)/Area(k)
    Flux_conv(k)=H_in*(T(k)-T_inside)
    Load=Load+Flux_conv(k)*Area(k)
    Toltal_Q=Toltal_Q+Q(k)

END DO

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )' )
WRITE(Results, '( "# Interior radiation test" )' )
WRITE(Results, '( // "Test Parameters -" )' )

WRITE(Results, '( "Inside convection coefficient =", ",", "&
    F15.4)" ) H_in
WRITE(Results, '( "Outside convection coefficient =", ",", "&
    F15.4)" ) H_out

! transfer the absolute temperature to C/F scale in output
IF (IPUnit) THEN
    WRITE(Results, '( "Inside air temperature =", ",", "&
        F15.4)" ) T_inside-459.67
    WRITE(Results, '( "Outside air temperature =", ",", "&
        F15.4)" ) T_outside-459.67
ELSE
    WRITE(Results, '( "Inside air temperature =", ",", "&
        F15.4)" ) T_inside-273.15
    WRITE(Results, '( "Outside air temperature =", ",", "&
        F15.4)" ) T_outside-273.15
END IF

WRITE(Results, '( "Width of the cuboid =", ",", "&
    F15.4)" ) Width
WRITE(Results, '( "Aspect ratio of length by width =", ",", "&
    F15.4)" ) Ratio_L
WRITE(Results, '( "Aspect ratio of Height by width =", ",", "&
    F15.4)" ) Ratio_H
WRITE(Results, '( "Emissivity of the surface =", ",", "&
    F15.4)" ) Emissivity

WRITE(Results, '( // "Test results -" )' )
WRITE(Results, '( // "Zone Load =", ",", "& F20.4)" ) load

WRITE(Results, '( // "Total radiative energy =", ",", "& F20.4)" ) Toltal_Q

```

```

WRITE(Results, '(// "Index", "", "Surface Temp.", "", "Surface Radiative
Energy", "", &
                "Surface Radiative Flux", "", "Surface Convective Flux")')

IF (IPUnit) THEN
    DO k = 1 , 6
        WRITE(Results, '(I6, "", F20.6, "", F20.6, "", F20.6, "",
F20.6)') &
                                K, T(k)-459.67,
Q(k), Flux_rad(k), Flux_conv(k)
        END DO
    ELSE
        DO k = 1 , 6
            WRITE(Results, '(I6, "", F20.6, "", F20.6, "", F20.6, "",
F20.6)') &
                                K, T(k)-273.15,
Q(k), Flux_rad(k), Flux_conv(k)
            END DO
        END IF

        ! write header lines
        WRITE(Results, '(// "# View factors")')

WRITE(Results, '(// "Index", "", "1", "", "2", "", "3", "", "4", "", "5", "", "6")')

    DO k=1, 6
        WRITE(Results, '(I6, "", F12.6, "", F12.6, "", F12.6, "", F12.6, "", &
F12.6, "", F12.6)') k, f(k,1), f(k,2), f(k,3), f(k,4), f(k,5),
f(k,6)
    END DO

    CLOSE(Results)

END SUBROUTINE IntRadiation

!-----
-----

SUBROUTINE Viewfactor(Width,Ratio_L,Ratio_H,f)

```

```

! PURPOSE:      caculate the view factors between the internal surfaces of
a cuboid
!              with specifed aspect ratio
!
! CREATED:
!   01.18.00    Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   Width      -   The width of the cuboid zone
!   Ratio_L    -   The aspect ratio of length to width
!   Ratio_H    -   The aspect ratio of height to width
!   f          -   The array of view factors

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
REAL,PARAMETER      :: PAI=3.1415926

! arguments
REAL(prec1), DIMENSION(6,6),INTENT(out)      :: f
REAL(prec1), INTENT(in)                      ::
Width
REAL(prec1), INTENT(in)                      ::
Ratio_L
REAL(prec1), INTENT(in)                      ::
Ratio_H

! local variables
REAL(prec1)      :: Length,Height           ! The length and Height of the
cuboid
INTEGER          :: i,j

! Calculate the length, height of the cuiboid zone
Length=Width*Ratio_L
Height=Width*Ratio_H

! Begin to calculate the viewfactors
Do i=1,6
  Do j=1,6
    IF (i==j) THEN
      f(i,j)=0.0
    END IF
  END DO
END DO

```

```

f(1,2)=2*(Height*SQRT(Length**2+Width**2)*ATAN(Height/SQRT(Length**2+Width**2)
)&

+Width*SQRT(Length**2+Height**2)*ATAN(Width/SQRT(Length**2+Height**2))&
-Height*Length*ATAN(Height/Length)-
Width*Length*ATAN(Width/Length)&
-
Length**2/2*LOG((Length**2+Width**2+Height**2)*Length**2/(Length**2+Width**2)&
/(Length**2+Height**2)))/(PAI*Height*Width)
f(2,1)=f(1,2)

f(1,3)=(ATAN(Height/Width)+Length/Width*ATAN(Height/Length)&
-
SQRT(Length**2+Width**2)/Width*ATAN(Height/SQRT(Length**2+Width**2))&
+Width/4/Height*LOG((Length**2+Width**2+Height**2)*Width**2/(Length**2+Width**
2))&
/(Width**2+Height**2))+Length**2/4/Height/Width*LOG((Length**2+Width**2+Height
**2)&
*Length**2/(Length**2+Width**2)/(Length**2+Height**2))-
Height/4/Width&
*LOG((Length**2+Width**2+Height**2)*Height**2/(Length**2+Height**2)/ &
(Width**2+Height**2)))/PAI
f(1,4)=f(1,3)
f(2,3)=f(1,3)
f(2,4)=f(1,3)

f(1,5)=(ATAN(Width/Height)+Length/Height*ATAN(Width/Length)&
-
SQRT(Length**2+Height**2)/Height*ATAN(Width/SQRT(Length**2+Height**2))&
+Height/4/Width*LOG((Length**2+Width**2+Height**2)*Height**2/(Length**2+Height
**2))&
/(Width**2+Height**2))+Length**2/4/Height/Width*LOG((Length**2+Width**2+Height
**2)&
*Length**2/(Length**2+Width**2)/(Length**2+Height**2))-
Width/4/Height&
*LOG((Length**2+Width**2+Height**2)*Width**2/(Length**2+Width**2)/
&
(Width**2+Height**2)))/PAI
f(1,6)=f(1,5)
f(2,5)=f(1,5)
f(2,6)=f(1,5)

f(3,4)=2*(Length*SQRT(Height**2+Width**2)*ATAN(Length/SQRT(Height**2+Width**2)
)&
+Height*SQRT(Length**2+Width**2)*ATAN(Height/SQRT(Length**2+Width**2))&
-Width*Length*ATAN(Length/Width)-Width*Height*ATAN(Height/Width)&
-
Width**2/2*LOG((Length**2+Width**2+Height**2)*Width**2/(Length**2+Width**2)&
/(Width**2+Height**2)))/(PAI*Height*Length)

```

```

f(4,3)=f(3,4)

f(3,1)=f(1,3)*Width/Length
f(3,2)=f(3,1)
f(4,1)=f(3,1)
f(4,2)=f(3,1)

f(3,5)=(ATAN(Length/Height)+Width/Height*ATAN(Length/Width)&
-
SQRT(Height**2+Width**2)/Height*ATAN(Length/SQRT(Height**2+Width**2))&
+Height/4/Length*LOG((Length**2+Width**2+Height**2)*Height**2/ &
(Length**2+Height**2)/(Width**2+Height**2))+Width**2/4/Height/ &
Length*LOG((Length**2+Width**2+Height**2)&
*Width**2/(Length**2+Width**2)/(Width**2+Height**2))-
Length/4/Height&
*LOG((Length**2+Width**2+Height**2)*Length**2/(Length**2+Height**2)/ &
(Width**2+Length**2)))/PAI
f(3,6)=f(3,5)
f(4,5)=f(3,5)
f(4,6)=f(3,5)

f(5,6)=2*(Length*SQRT(Height**2+Width**2)*ATAN(Length/SQRT(Height**2+Width**2)
)&
+Width*SQRT(Length**2+Height**2)*ATAN(Width/SQRT(Length**2+Height**2))&
-Height*Length*ATAN(Length/Height)-
Width*Height*ATAN(Width/Height)&
-
Height**2/2*LOG((Length**2+Width**2+Height**2)*Height**2/(Length**2+Height**2)
&
/(Width**2+Height**2)))/(PAI*Width*Length)
f(6,5)=f(5,6)

f(5,1)=f(1,5)*Height/Length
f(5,2)=f(5,1)
f(6,1)=f(5,1)
f(6,2)=f(5,1)

f(5,3)=f(3,5)*Height/Width
f(5,4)=f(5,3)
f(6,3)=f(5,3)
f(6,4)=f(5,3)

END SUBROUTINE Viewfactor

!-----
-

SUBROUTINE
JacobianFun(T,NumVar,D,Area,Delta,f,H_in,H_out,T_outside,T_inside, &
Alpha,Beta,Const_SB)

```

```

! PURPOSE:      caculate the Jacobian matrix and the set of Function
values
!              that will be used in solving the set nonlinear equations
with
!              Newton-Raphson Method
!
!
! CREATED:
!   01.18.00    Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   T           -   Array of surface temperature
!   NumVar       -   The number of variables can be solved
!   D           -   The square 6X6 matrix used in calculation of Q
!   Area        -   The array of surface area
!   Delta       -   Kronecker's delta,=1 if k=j, =0 otherwise
!   f           -   The array of view factors
!   H_in        -   Internal convection heat transfer coefficient
!   H_out       -   External convection heat transfer coefficient
!   T_inside    -   Internal air temperature
!   T_outside   -   External air temperature
!   Alpha       -   the Jacobian matrix of the set of nonlinear
equations
!   Beita       -   the function values of the set of nonlinear
equations
!   Const_SB    ::   Stefan-Boltzman constant

IMPLICIT NONE

!parameters
INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)

!arguments
INTEGER                  :: NumVar
REAL(prec1), INTENT(in)  :: H_in
REAL(prec1), INTENT(in)  :: H_out
REAL(prec1), INTENT(in)  :: T_outside
REAL(prec1), INTENT(in)  :: T_inside
REAL(prec1), INTENT(in)  :: Const_SB
REAL(prec1), DIMENSION(NumVar,NumVar)
                        :: Alpha
REAL(prec1), DIMENSION(6,6), INTENT(in)  :: D
REAL(prec1), DIMENSION(6,6), INTENT(in)  :: Delta
REAL(prec1), DIMENSION(6,6), INTENT(in)  :: f

REAL(prec1), DIMENSION(NumVar)             :: Beita
REAL(prec1), DIMENSION(NumVar), INTENT(in) :: T
REAL(prec1), DIMENSION(6), INTENT(in)      :: Area

!local variables

```



```

      INTEGER                                :: i,j,k
      REAL(prec2), DIMENSION(6,6)          :: Coef

      DO k=1,6

         Beita(k)=0.0
         DO j=1,6

            Coef(k,j)=0.0
            DO i=1,6
               Coef(k,j)=Coef(k,j)+Const_SB*(D(k,i)*(Delta(i,j)-
f(i,j)))/Area(k)
            END DO
            Beita(k)=Beita(k)+Coef(k,j)*T(j)**4
            Alpha(k,j)=4*Coef(k,j)*T(j)**3

         END DO

      END DO

      Beita(1)=Beita(1)+(H_in+H_out)*T(1)-(H_in*T_inside+H_out*T_outside)
      Alpha(1,1)=Alpha(1,1)+(H_in+H_out)
      DO k=2,6
         Beita(k)=Beita(k)+H_in*T(k)-H_in*T_inside
         Alpha(k,k)=Alpha(k,k)+H_in
      END DO

END SUBROUTINE JacobianFun

!-----
-

SUBROUTINE NRNonlinear(NStep,x,n,SUMX,SUMF,B,Area,Delta,f,H_in,H_out, &
                     T_outside,T_inside, Const_SB)

! PURPOSE:      Solving a set of nonlinear equations using the Newton-
Raphson method.
!              The algorithm is based on that described in Numerical
Recipes
!              (Press, Flannery, Teukolsky and Vetterling 1986)
!
! CREATED:
!   01.18.00    Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   NStep      -   Take NStep Newton-Raphson steps to improve the
root
!   x          -   Array of unknowns to be solved
!   n          -   number of unknowns to be solved

```

```

!    SUMX                -    Summed variable increments
!    SUMF                -    Summed function increments
!    B                  -    The square 6X6 matrix used in calculation of Q
!    Area               -    The array of surface area
!    Delta              -    Kronecker's delta,=1 if k=j, =0 otherwise
!    f                  -    The array of view factors
!    H_in               -    Internal convection heat transfer coefficient
!    H_out              -    External convection heat transfer coefficient
!    T_inside           -    Internal air temperature
!    T_outside          -    External air temperature
!    Const_SB           ::    Stefan-Boltzman constant

IMPLICIT NONE

!parameters
INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER      :: NumVar=15

!arguments
INTEGER, INTENT(in)      :: n,NStep
REAL(prec1), INTENT(in)  :: SUMF,SUMX
REAL(prec1), DIMENSION(n) :: x
REAL(prec1), INTENT(in)  :: H_in
REAL(prec1), INTENT(in)  :: H_out
REAL(prec1), INTENT(in)  :: T_outside
REAL(prec1), INTENT(in)  :: T_inside
REAL(prec1), INTENT(in)  :: Const_SB
REAL(prec1), DIMENSION(6,6), INTENT(in) :: B
REAL(prec1), DIMENSION(6,6), INTENT(in) :: Delta
REAL(prec1), DIMENSION(6,6), INTENT(in) :: f
REAL(prec1), DIMENSION(6), INTENT(in)   :: Area

!local variables
INTEGER                  :: i,k
INTEGER, DIMENSION(NumVar) :: Index
REAL(prec1)              ::
FlagD, FunError, XError
REAL(prec1), DIMENSION(NumVar,NumVar) :: Alpha
REAL(prec1), DIMENSION(NumVar)       :: Beita
REAL(prec1), DIMENSION(NumVar)       :: p

DO k=1,NStep
    CALL
    JacobianFun(x,NumVar,B,Area,Delta,f,H_in,H_out,T_outside,T_inside, &
                Alpha,Beita,Const_SB)

    FunError=0.
    DO i=1,n
        FunError=FunError+abs(Beita(i))
    END DO
    if(FunError.le.SUMF)return

    DO i=1,n
        p(i)=-Beita(i)
    END DO

```

```

      call LUDecomp(Alpha,n,NumVar,Index,FlagD)
      call LUSubstitution(Alpha,n,NumVar,Index,p)

      XError=0.

      DO i=1,n
        XError=XError+abs(p(i))
        x(i)=x(i)+p(i)
      END DO
      if(XError.le.SUMX)return

    END DO

  END SUBROUTINE NRNonlinear

!-----
-

SUBROUTINE LUDecomp(MatrixIn,n,NumVar,Index,FlagD)

  ! PURPOSE:      Execute the LU decomposition in solving nonlinear
equations.
  !              The algorithm is based on that described in Numerical
Recipes
  !              (Press, Flannery, Teukolsky and Vetterling 1986)
  ! CREATED:
  !   01.18.00    Dongyi Xiao
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   n/a
  !
  ! SUBROUTINE ARGUMENTS:
  !   MatrixIn    -   The input nxn matrix
  !   NumVar      -   The physical dimension of matrix MatrixIn
  !   Index       -   The output vector records the row permutation
  !   FlagD       -   partial pivoting
  !                   =1 if the number of row interchages is even,
  !                   =-1 if odd

  !parameters
  INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
  INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)
  INTEGER, PARAMETER      :: NMAX = 500
  REAL(prec1), PARAMETER  :: SNum=1.0e-20

  !arguments
  INTEGER, INTENT(in)      :: n,NumVar
  INTEGER, DIMENSION(n), INTENT(out) :: Index
  REAL(prec1), DIMENSION(NumVar,NumVar) :: MatrixIn

```

```

REAL(prec1),INTENT(out)                                :: FlagD

!local variables
INTEGER                                                :: i,imax,j,k
REAL(prec1)                                           :: aamax,dum,sum
REAL(prec1), DIMENSION(NMAX)                         :: vv

FlagD=1.
DO i=1,n

    aamax=0.

    DO j=1,n
        IF (abs(MatrixIn(i,j)).gt.aamax) THEN
            aamax=abs(MatrixIn(i,j))
        END IF
    END DO

    IF (aamax.eq.0.) THEN
        PAUSE 'singular matrix in LUDecomp'
    END IF
    vv(i)=1./aamax

END DO

DO j=1,n
    IF (j.GT.1) THEN
        DO i=1,j-1
            sum=MatrixIn(i,j)
            IF (i.GT.1) THEN
                DO k=1,i-1
                    sum=sum-MatrixIn(i,k)*MatrixIn(k,j)
                END DO
            MatrixIn(i,j)=sum
        ENDIF
    END DO
ENDIF
aamax=0.

DO i=j,n
    sum=MatrixIn(i,j)
    IF (j.GT.1) THEN
        DO k=1,j-1
            sum=sum-MatrixIn(i,k)*MatrixIn(k,j)
        END DO
    MatrixIn(i,j)=sum
ENDIF
dum=vv(i)*abs(sum)
IF (dum.ge.aamax) THEN
    imax=i
    aamax=dum
ENDIF
END DO

IF (j.NE.imax) THEN
    DO k=1,n
        dum=MatrixIn(imax,k)
    
```

```

        MatrixIn(imax,k)=MatrixIn(j,k)
        MatrixIn(j,k)=dum
    END DO
    FlagD=-FlagD
    vv(imax)=vv(j)
ENDIF

Index(j)=imax

IF (j.NE.n) THEN
    IF(MatrixIn(j,j) .EQ. 0.) THEN
        MatrixIn(j,j)=SNum
    END IF

    dum=1./MatrixIn(j,j)

    DO i=j+1,n
        MatrixIn(i,j)=MatrixIn(i,j)*dum
    END DO
ENDIF

END DO

IF(MatrixIn(n,n) .EQ. 0.) THEN
    MatrixIn(n,n)=SNum
END IF

END SUBROUTINE LUDecomp

!-----
-

SUBROUTINE LUSubstitution(MatrixIn,n,NumVar,Index,b)

    ! PURPOSE:      Execute the forward and backward substitution in solving
nonlinear
    !               equations.The algorithm is based on that described in
    !               Numerical Recipes (Press, Flannery, Teukolsky and
Vetterling 1986).
    !
    ! CREATED:
    !   01.18.00    Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   MatrixIn      -   The input nxn matrix
    !   NumVar        -   The physical dimension of matrix MatrixIn
    !   Index         -   The input vector records the row permutation
effected by the
    !               partial pivoting in LU decomposition

```

```

!      b                      -   The input right hand side vector

!parameters
INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)

!arguments
INTEGER, INTENT(in)      :: n, NumVar
INTEGER, DIMENSION(n), INTENT(in) :: Index
REAL(prec1), DIMENSION(NumVar, NumVar), INTENT(in) :: MatrixIn
REAL(prec1), DIMENSION(n) :: b

!local variables
INTEGER                  :: i, ii, j, ll
REAL(prec1)              :: sum

ii=0
DO i=1,n
    ll=Index(i)
    sum=b(ll)
    b(ll)=b(i)
    IF (ii.ne.0) THEN
        DO j=ii,i-1
            sum=sum-MatrixIn(i,j)*b(j)
        END DO
    ELSE IF (sum.ne.0.) THEN
        ii=i
    ENDIF
    b(i)=sum
END DO

DO i=n,1,-1
    sum=b(i)
    IF (i.lt.n) THEN
        DO j=i+1,n
            sum=sum-MatrixIn(i,j)*b(j)
        END DO
    ENDIF
    b(i)=sum/MatrixIn(i,i)
END DO

END SUBROUTINE LUSubstitution

END MODULE IntRad_1

```

Source Code: Module MatrixInverse

MODULE MatrixInverse

```

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
!
! PURPOSE:      Contains utility routines for matrix inverse used in the
!               calculation of analytical solutions to the internal long
!               wave radiation test case.
!
! CREATED:
!   9.3.00
!
!               The following Linpack subroutines are downloaded from
!               http://www.netlib.org/cgi-bin/search.pl. For purpose of
!               compiling in free format Fortran90, the comment style has
!               been changed from "c" to "!" These routines are put in this
!               module named"MatrixInverse" with the way of defining functions
!               and subroutines modified and are used to do matrix inverse
!               in the test of Interior Long Wave Radiation. -- D. Xiao
!
! MODIFIED:
!
CONTAINS
!-----
-

SUBROUTINE SGECO(A,LDA,N,IPVT,RCOND,Z)
  INTEGER LDA,N,IPVT(1)
  REAL A(LDA,1),Z(1)
  REAL RCOND
!
! SGECO FACTORS A REAL MATRIX BY GAUSSIAN ELIMINATION
! AND ESTIMATES THE CONDITION OF THE MATRIX.
!
! IF RCOND IS NOT NEEDED, SGEFA IS SLIGHTLY FASTER.
! TO SOLVE A*X = B , FOLLOW SGECO BY SGESL.
! TO COMPUTE INVERSE(A)*C , FOLLOW SGECO BY SGESL.
! TO COMPUTE DETERMINANT(A) , FOLLOW SGECO BY SGEDI.
! TO COMPUTE INVERSE(A) , FOLLOW SGECO BY SGEDI.
!
! ON ENTRY
!

```

```

!      A      REAL(LDA, N)
!              THE MATRIX TO BE FACTORED.
!
!      LDA      INTEGER
!              THE LEADING DIMENSION OF THE ARRAY  A  .
!
!      N      INTEGER
!              THE ORDER OF THE MATRIX  A  .
!
!      ON RETURN
!
!      A      AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
!              WHICH WERE USED TO OBTAIN IT.
!              THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
!              L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
!              TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
!
!      IPVT      INTEGER(N)
!              AN INTEGER VECTOR OF PIVOT INDICES.
!
!      RCOND      REAL
!              AN ESTIMATE OF THE RECIPROCAL CONDITION OF  A  .
!              FOR THE SYSTEM  A*X = B  , RELATIVE PERTURBATIONS
!              IN  A  AND  B  OF SIZE  EPSILON  MAY CAUSE
!              RELATIVE PERTURBATIONS IN  X  OF SIZE  EPSILON/RCOND  .
!              IF  RCOND  IS SO SMALL THAT THE LOGICAL EXPRESSION
!                  1.0 + RCOND .EQ. 1.0
!              IS TRUE, THEN  A  MAY BE SINGULAR TO WORKING
!              PRECISION.  IN PARTICULAR,  RCOND  IS ZERO IF
!              EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
!              UNDERFLOWS.
!
!      Z      REAL(N)
!              A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
!              IF  A  IS CLOSE TO A SINGULAR MATRIX, THEN  Z  IS
!              AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
!                  NORM(A*Z) = RCOND*NORM(A)*NORM(Z)  .
!
!      LINPACK. THIS VERSION DATED 08/14/78  .
!      CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
!
!      SUBROUTINES AND FUNCTIONS
!
!      LINPACK SGEFA
!      BLAS SAXPY,SDOT,SSCAL,SASUM
!      FORTRAN ABS,AMAX1,SIGN
!
!      INTERNAL VARIABLES
!
!      REAL SDOT,EK,T,WK,WKM
!      REAL EK,T,WK,WKM ! define of SDOT is excluded here --D. Xiao
!      REAL ANORM,S,SASUM,SM,YNORM
!      REAL ANORM,S,SM,YNORM ! define of SASUM is excluded here --D. Xiao
!      INTEGER INFO,J,K,KB,KP1,L
!
!      COMPUTE 1-NORM OF A

```



```

!
      ANORM = 0.0E0
      DO 10 J = 1, N
        ANORM = AMAX1(ANORM,SASUM(N,A(1,J),1))
10    CONTINUE
!
!      FACTOR
!
      CALL SGEFA(A,LDA,N,IPVT,INFO)
!
!      RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
!      ESTIMATE = NORM(Z)/NORM(Y) WHERE A*Z = Y AND TRANS(A)*Y = E .
!      TRANS(A) IS THE TRANSPOSE OF A . THE COMPONENTS OF E ARE
!      CHOSEN TO CAUSE MAXIMUM LOCAL GROWTH IN THE ELEMENTS OF W WHERE
!      TRANS(U)*W = E . THE VECTORS ARE FREQUENTLY RESCALED TO AVOID
!      OVERFLOW.
!
      SOLVE TRANS(U)*W = E
!
      EK = 1.0E0
      DO 20 J = 1, N
        Z(J) = 0.0E0
20    CONTINUE
      DO 100 K = 1, N
        IF (Z(K) .NE. 0.0E0) EK = SIGN(EK,-Z(K))
        IF (ABS(EK-Z(K)) .LE. ABS(A(K,K))) GO TO 30
        S = ABS(A(K,K))/ABS(EK-Z(K))
        CALL SSCAL(N,S,Z,1)
        EK = S*EK
30      CONTINUE
        WK = EK - Z(K)
        WKM = -EK - Z(K)
        S = ABS(WK)
        SM = ABS(WKM)
        IF (A(K,K) .EQ. 0.0E0) GO TO 40
        WK = WK/A(K,K)
        WKM = WKM/A(K,K)
        GO TO 50
40      CONTINUE
        WK = 1.0E0
        WKM = 1.0E0
50      CONTINUE
        KP1 = K + 1
        IF (KP1 .GT. N) GO TO 90
        DO 60 J = KP1, N
          SM = SM + ABS(Z(J)+WKM*A(K,J))
          Z(J) = Z(J) + WK*A(K,J)
          S = S + ABS(Z(J))
60      CONTINUE
        IF (S .GE. SM) GO TO 80
        T = WKM - WK
        WK = WKM
        DO 70 J = KP1, N
          Z(J) = Z(J) + T*A(K,J)
70      CONTINUE
80      CONTINUE
90      CONTINUE

```

```

      Z(K) = WK
100 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
!
!   SOLVE TRANS(L)*Y = W
!
      DO 120 KB = 1, N
        K = N + 1 - KB
        IF (K .LT. N) Z(K) = Z(K) + SDOT(N-K,A(K+1,K),1,Z(K+1),1)
        IF (ABS(Z(K)) .LE. 1.0E0) GO TO 110
        S = 1.0E0/ABS(Z(K))
        CALL SSCAL(N,S,Z,1)
110    CONTINUE
        L = IPVT(K)
        T = Z(L)
        Z(L) = Z(K)
        Z(K) = T
120 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
!
      YNORM = 1.0E0
!
!   SOLVE L*V = Y
!
      DO 140 K = 1, N
        L = IPVT(K)
        T = Z(L)
        Z(L) = Z(K)
        Z(K) = T
        IF (K .LT. N) CALL SAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
        IF (ABS(Z(K)) .LE. 1.0E0) GO TO 130
        S = 1.0E0/ABS(Z(K))
        CALL SSCAL(N,S,Z,1)
        YNORM = S*YNORM
130    CONTINUE
140 CONTINUE
      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
!
!   SOLVE U*Z = V
!
      DO 160 KB = 1, N
        K = N + 1 - KB
        IF (ABS(Z(K)) .LE. ABS(A(K,K))) GO TO 150
        S = ABS(A(K,K))/ABS(Z(K))
        CALL SSCAL(N,S,Z,1)
        YNORM = S*YNORM
150    CONTINUE
        IF (A(K,K) .NE. 0.0E0) Z(K) = Z(K)/A(K,K)
        IF (A(K,K) .EQ. 0.0E0) Z(K) = 1.0E0
        T = -Z(K)
        CALL SAXPY(K-1,T,A(1,K),1,Z(1),1)
160 CONTINUE
!   MAKE ZNORM = 1.0

```

```

      S = 1.0E0/SASUM(N,Z,1)
      CALL SSCAL(N,S,Z,1)
      YNORM = S*YNORM
!
      IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
      RETURN
      END SUBROUTINE SGECO

      SUBROUTINE SGEFA(A,LDA,N,IPVT,INFO)
      INTEGER LDA,N,IPVT(1),INFO
      REAL A(LDA,1)
!
!      SGEFA FACTORS A REAL MATRIX BY GAUSSIAN ELIMINATION.
!
!      SGEFA IS USUALLY CALLED BY SGECO, BUT IT CAN BE CALLED
!      DIRECTLY WITH A SAVING IN TIME IF RCOND IS NOT NEEDED.
!      (TIME FOR SGECO) = (1 + 9/N)*(TIME FOR SGEFA) .
!
!      ON ENTRY
!
!      A      REAL(LDA, N)
!             THE MATRIX TO BE FACTORED.
!
!      LDA    INTEGER
!             THE LEADING DIMENSION OF THE ARRAY A .
!
!      N      INTEGER
!             THE ORDER OF THE MATRIX A .
!
!      ON RETURN
!
!      A      AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
!             WHICH WERE USED TO OBTAIN IT.
!             THE FACTORIZATION CAN BE WRITTEN A = L*U WHERE
!             L IS A PRODUCT OF PERMUTATION AND UNIT LOWER
!             TRIANGULAR MATRICES AND U IS UPPER TRIANGULAR.
!
!      IPVT   INTEGER(N)
!             AN INTEGER VECTOR OF PIVOT INDICES.
!
!      INFO   INTEGER
!             = 0  NORMAL VALUE.
!             = K  IF U(K,K) .EQ. 0.0 . THIS IS NOT AN ERROR
!                 CONDITION FOR THIS SUBROUTINE, BUT IT DOES
!                 INDICATE THAT SGESL OR SGEDI WILL DIVIDE BY ZERO
!                 IF CALLED. USE RCOND IN SGECO FOR A RELIABLE
!                 INDICATION OF SINGULARITY.
!
!      LINPACK. THIS VERSION DATED 08/14/78 .
!      CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
!
!      SUBROUTINES AND FUNCTIONS
!
!      BLAS SAXPY,SSCAL,ISAMAX
!
!      INTERNAL VARIABLES

```

```

!
      REAL T
!      INTEGER ISAMAX,J,K,KP1,L,NM1
      INTEGER J,K,KP1,L,NM1 ! define of ISAMAX is excluded here --D. Xiao
!
!
!      GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
!
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
        KP1 = K + 1
!
!      FIND L = PIVOT INDEX
!
        L = ISAMAX(N-K+1,A(K,K),1) + K - 1
        IPVT(K) = L
!
!      ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
!
        IF (A(L,K) .EQ. 0.0E0) GO TO 40
!
!      INTERCHANGE IF NECESSARY
!
        IF (L .EQ. K) GO TO 10
        T = A(L,K)
        A(L,K) = A(K,K)
        A(K,K) = T
10      CONTINUE
!
!      COMPUTE MULTIPLIERS
!
        T = -1.0E0/A(K,K)
        CALL SSCAL(N-K,T,A(K+1,K),1)
!
!      ROW ELIMINATION WITH COLUMN INDEXING
!
        DO 30 J = KP1, N
          T = A(L,J)
          IF (L .EQ. K) GO TO 20
          A(L,J) = A(K,J)
          A(K,J) = T
20        CONTINUE
          CALL SAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
30        CONTINUE
          GO TO 50
40      CONTINUE
          INFO = K
50      CONTINUE
60      CONTINUE
70      CONTINUE
      IPVT(N) = N
      IF (A(N,N) .EQ. 0.0E0) INFO = N
      RETURN
      END SUBROUTINE SGEFA

```

```

SUBROUTINE SGEDI(A,LDA,N,IPVT,DET,WORK,JOB)
INTEGER LDA,N,IPVT(1),JOB
REAL A(LDA,1),DET(2),WORK(1)
!
! SGEDI COMPUTES THE DETERMINANT AND INVERSE OF A MATRIX
! USING THE FACTORS COMPUTED BY SGECO OR SGEFA.
!
! ON ENTRY
!
!     A      REAL(LDA, N)
!             THE OUTPUT FROM SGECO OR SGEFA.
!
!     LDA    INTEGER
!             THE LEADING DIMENSION OF THE ARRAY A .
!
!     N      INTEGER
!             THE ORDER OF THE MATRIX A .
!
!     IPVT   INTEGER(N)
!             THE PIVOT VECTOR FROM SGECO OR SGEFA.
!
!     WORK   REAL(N)
!             WORK VECTOR.  CONTENTS DESTROYED.
!
!     JOB    INTEGER
!             = 11  BOTH DETERMINANT AND INVERSE.
!             = 01  INVERSE ONLY.
!             = 10  DETERMINANT ONLY.
!
! ON RETURN
!
!     A      INVERSE OF ORIGINAL MATRIX IF REQUESTED.
!             OTHERWISE UNCHANGED.
!
!     DET    REAL(2)
!             DETERMINANT OF ORIGINAL MATRIX IF REQUESTED.
!             OTHERWISE NOT REFERENCED.
!             DETERMINANT = DET(1) * 10.0**DET(2)
!             WITH 1.0 .LE. ABS(DET(1)) .LT. 10.0
!             OR DET(1) .EQ. 0.0 .
!
! ERROR CONDITION
!
!     A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS
!     A ZERO ON THE DIAGONAL AND THE INVERSE IS REQUESTED.
!     IT WILL NOT OCCUR IF THE SUBROUTINES ARE CALLED CORRECTLY
!     AND IF SGECO HAS SET RCOND .GT. 0.0 OR SGEFA HAS SET
!     INFO .EQ. 0 .
!
! LINPACK. THIS VERSION DATED 08/14/78 .
! CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
!
! SUBROUTINES AND FUNCTIONS
!
! BLAS SAXPY,SSCAL,SSWAP
! FORTRAN ABS,MOD

```

```

!
!   INTERNAL VARIABLES
!
REAL T
REAL TEN
INTEGER I,J,K,KB,KP1,L,NM1
!
!
!   COMPUTE DETERMINANT
!
IF (JOB/10 .EQ. 0) GO TO 70
  DET(1) = 1.0E0
  DET(2) = 0.0E0
  TEN = 10.0E0
  DO 50 I = 1, N
    IF (IPVT(I) .NE. I) DET(1) = -DET(1)
    DET(1) = A(I,I)*DET(1)
!   ...EXIT
    IF (DET(1) .EQ. 0.0E0) GO TO 60
10    IF (ABS(DET(1)) .GE. 1.0E0) GO TO 20
      DET(1) = TEN*DET(1)
      DET(2) = DET(2) - 1.0E0
      GO TO 10
20    CONTINUE
30    IF (ABS(DET(1)) .LT. TEN) GO TO 40
      DET(1) = DET(1)/TEN
      DET(2) = DET(2) + 1.0E0
      GO TO 30
40    CONTINUE
50    CONTINUE
60    CONTINUE
70 CONTINUE
!
!   COMPUTE INVERSE(U)
!
IF (MOD(JOB,10) .EQ. 0) GO TO 150
DO 100 K = 1, N
  A(K,K) = 1.0E0/A(K,K)
  T = -A(K,K)
  CALL SSCAL(K-1,T,A(1,K),1)
  KP1 = K + 1
  IF (N .LT. KP1) GO TO 90
  DO 80 J = KP1, N
    T = A(K,J)
    A(K,J) = 0.0E0
    CALL SAXPY(K,T,A(1,K),1,A(1,J),1)
80    CONTINUE
90    CONTINUE
100   CONTINUE
!
!   FORM INVERSE(U)*INVERSE(L)
!
NM1 = N - 1
IF (NM1 .LT. 1) GO TO 140
DO 130 KB = 1, NM1
  K = N - KB
  KP1 = K + 1

```

```

        DO 110 I = KP1, N
            WORK(I) = A(I,K)
            A(I,K) = 0.0E0
110      CONTINUE
        DO 120 J = KP1, N
            T = WORK(J)
            CALL SAXPY(N,T,A(1,J),1,A(1,K),1)
120      CONTINUE
            L = IPVT(K)
            IF (L .NE. K) CALL SSWAP(N,A(1,K),1,A(1,L),1)
130      CONTINUE
140      CONTINUE
150      CONTINUE
        RETURN
        END SUBROUTINE SGEDI

        SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)
!
!   CONSTANT TIMES A VECTOR PLUS A VECTOR.
!   USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
!   JACK DONGARRA, LINPACK, 3/11/78.
!
        REAL SX(1),SY(1),SA
        INTEGER I,INCX,INCY,IX,IY,M,MP1,N
!
        IF(N.LE.0)RETURN
        IF (SA .EQ. 0.0E0) RETURN
        IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
!
!       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
!       NOT EQUAL TO 1
!
        IX = 1
        IY = 1
        IF(INCX.LT.0)IX = (-N+1)*INCX + 1
        IF(INCY.LT.0)IY = (-N+1)*INCY + 1
        DO 10 I = 1,N
            SY(IY) = SY(IY) + SA*SX(IX)
            IX = IX + INCX
            IY = IY + INCY
10      CONTINUE
        RETURN
!
!       CODE FOR BOTH INCREMENTS EQUAL TO 1
!
!       CLEAN-UP LOOP
!
        20 M = MOD(N,4)
        IF( M .EQ. 0 ) GO TO 40
        DO 30 I = 1,M
            SY(I) = SY(I) + SA*SX(I)
30      CONTINUE
        IF( N .LT. 4 ) RETURN
        40 MP1 = M + 1
        DO 50 I = MP1,N,4
            SY(I) = SY(I) + SA*SX(I)

```

```

        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)
50 CONTINUE
    RETURN
END SUBROUTINE SAXPY

FUNCTION SDOT(N,SX,INCX,SY,INCY)
!
!   FORMS THE DOT PRODUCT OF TWO VECTORS.
!   USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
!   JACK DONGARRA, LINPACK, 3/11/78.
!
    REAL SDOT
    REAL SX(1),SY(1),TEMP
    INTEGER I,INCX,INCY,IX,IY,M,MP1,N
!
    SDOT = 0.0E0
    TEMP = 0.0E0
    IF(N.LE.0)RETURN
    IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
!
!       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
!       NOT EQUAL TO 1
!
    IX = 1
    IY = 1
    IF(INCX.LT.0)IX = (-N+1)*INCX + 1
    IF(INCY.LT.0)IY = (-N+1)*INCY + 1
    DO 10 I = 1,N
        TEMP = TEMP + SX(IX)*SY(IY)
        IX = IX + INCX
        IY = IY + INCY
10 CONTINUE
    SDOT = TEMP
    RETURN
!
!       CODE FOR BOTH INCREMENTS EQUAL TO 1
!
!
!       CLEAN-UP LOOP
!
20 M = MOD(N,5)
    IF( M .EQ. 0 ) GO TO 40
    DO 30 I = 1,M
        TEMP = TEMP + SX(I)*SY(I)
30 CONTINUE
    IF( N .LT. 5 ) GO TO 60
40 MP1 = M + 1
    DO 50 I = MP1,N,5
        TEMP = TEMP + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) + SX(I + 2)*SY(I + 2)
+ SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4)
50 CONTINUE
60 SDOT = TEMP
    RETURN
END FUNCTION SDOT

```



```

      SUBROUTINE  SSCAL(N,DA,DX,INCX)
!
!   SCALES A VECTOR BY A CONSTANT.
!   USES UNROLLED LOOPS FOR INCREMENT EQUAL TO ONE.
!   JACK DONGARRA, LINPACK, 3/11/78.
!
      REAL DA,DX(1)
      INTEGER I,INCX,M,MP1,N,NINCX
!
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20
!
!       CODE FOR INCREMENT NOT EQUAL TO 1
!
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        DX(I) = DA*DX(I)
10  CONTINUE
      RETURN
!
!       CODE FOR INCREMENT EQUAL TO 1
!
!
!       CLEAN-UP LOOP
!
20  M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DX(I) = DA*DX(I)
30  CONTINUE
      IF( N .LT. 5 ) RETURN
40  MP1 = M + 1
      DO 50 I = MP1,N,5
        DX(I) = DA*DX(I)
        DX(I + 1) = DA*DX(I + 1)
        DX(I + 2) = DA*DX(I + 2)
        DX(I + 3) = DA*DX(I + 3)
        DX(I + 4) = DA*DX(I + 4)
50  CONTINUE
      RETURN
      END SUBROUTINE  SSCAL

      FUNCTION SASUM(N,X,INCX)
!
!   TAKES THE SUM OF THE ABSOLUTE VALUES.
!   JACK DONGARRA, LINPACK, 3/11/78.
!
      REAL SASUM
      REAL X(1),TEMP
      INTEGER I,INCX,M,MP1,N,NINCX
!
      SASUM = 0.0E0
      TEMP = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20

```

```

!
!      CODE FOR INCREMENT NOT EQUAL TO 1
!
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        TEMP = TEMP + ABS(X(I))
10    CONTINUE
      SASUM = TEMP
      RETURN

!
!      CODE FOR INCREMENT EQUAL TO 1
!
!
!      CLEAN-UP LOOP
!
20    M = MOD(N,6)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        TEMP = TEMP + ABS(X(I))
30    CONTINUE
      IF( N .LT. 6 ) GO TO 60
40    MP1 = M + 1
      DO 50 I = MP1,N,6
        TEMP = TEMP + ABS(X(I)) + ABS(X(I + 1)) + ABS(X(I + 2)) + ABS(X(I +
3)) + ABS(X(I + 4)) + ABS(X(I + 5))
50    CONTINUE
60    SASUM = TEMP
      RETURN
      END FUNCTION SASUM

      FUNCTION ISAMAX(N,SX,INCX)
!
!      FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
!      JACK DONGARRA, LINPACK, 3/11/78.
!
      INTEGER ISAMAX      !
      REAL SX(1),SMAX
      INTEGER I,INCX,IX,N
!
      ISAMAX = 0
      IF( N .LT. 1 ) RETURN
      ISAMAX = 1
      IF(N.EQ.1)RETURN
      IF(INCX.EQ.1)GO TO 20
!
!      CODE FOR INCREMENT NOT EQUAL TO 1
!
      IX = 1
      SMAX = ABS(SX(1))
      IX = IX + INCX
      DO 10 I = 2,N
        IF(ABS(SX(IX)).LE.SMAX) GO TO 5
        ISAMAX = I
        SMAX = ABS(SX(IX))
      5    IX = IX + INCX
10    CONTINUE
      RETURN

```

```

!
!      CODE FOR INCREMENT EQUAL TO 1
!
20 SMAX = ABS(SX(1))
   DO 30 I = 2,N
       IF(ABS(SX(I)).LE.SMAX) GO TO 30
       ISAMAX = I
       SMAX = ABS(SX(I))
30 CONTINUE
   RETURN
   END FUNCTION ISAMAX

SUBROUTINE SSWAP (N,SX,INCX,SY,INCY)
!
!      INTERCHANGES TWO VECTORS.
!      USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO 1.
!      JACK DONGARRA, LINPACK, 3/11/78.
!
   REAL SX(1),SY(1),STEMP
   INTEGER I,INCX,INCY,IX,IY,M,MP1,N
!
   IF(N.LE.0)RETURN
   IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
!
!      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL
!      TO 1
!
   IX = 1
   IY = 1
   IF(INCX.LT.0)IX = (-N+1)*INCX + 1
   IF(INCY.LT.0)IY = (-N+1)*INCY + 1
   DO 10 I = 1,N
       STEMP = SX(IX)
       SX(IX) = SY(IY)
       SY(IY) = STEMP
       IX = IX + INCX
       IY = IY + INCY
10 CONTINUE
   RETURN
!
!      CODE FOR BOTH INCREMENTS EQUAL TO 1
!
!
!      CLEAN-UP LOOP
!
20 M = MOD(N,3)
   IF( M .EQ. 0 ) GO TO 40
   DO 30 I = 1,M
       STEMP = SX(I)
       SX(I) = SY(I)
       SY(I) = STEMP
30 CONTINUE
   IF( N .LT. 3 ) RETURN
40 MP1 = M + 1
   DO 50 I = MP1,N,3
       STEMP = SX(I)
       SX(I) = SY(I)

```

```
        SY(I) = STEMP
        STEMP = SX(I + 1)
        SX(I + 1) = SY(I + 1)
        SY(I + 1) = STEMP
        STEMP = SX(I + 2)
        SX(I + 2) = SY(I + 2)
        SY(I + 2) = STEMP
50 CONTINUE
    RETURN
END SUBROUTINE SSWAP

END MODULE MatrixInverse
```

Source Code: Module ExtLWRad

```

MODULE ExtLWRad

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical      solutions to the external long wave radiation test case.
!
! CREATED:      7.7.2000      Dongyi Xiao
!
! MODIFIED:
!

USE ConductUtil

PRIVATE
! make public main subroutine
PUBLIC :: ExtLWRadiation

CONTAINS

!-----
-

SUBROUTINE ExtLWRadiation(FileName, Cond, N_Cond, Thickness, N_thickness,
Emissivity, &
                        T_outside, T_inside, T_sky, Hin_A, Hin_C, Hin_n,
Hout_A, &
                        Hout_C, Hout_n, Const_SB, Sur_area, IPUnit, ErrorFlag,
ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "ExtLWRadiation" :: ExtLWRadiation

! PURPOSE:      Produces weather files and analytical response for the
case

```

```

!           of external long wave radiation. Similar to the steady
state
!           convection test, the convection correlations inside and
!           outside are specified as  $H = A + C \cdot (T_a - T_s)^n$ .
!           The steady state heat flux is found using a successive
!           substitution method to solve the non-linear equation.
!           The outside surface temperature is found using the
bysection
!           method to solve the non-linear equation.
!
!
!   CREATED:
!     7.7.2000      S.J.Rees, Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!     n/a
!
!   OUTPUT FILES:
!
!     FileName.dat      -   Analytical response data
!
!   SUBROUTINE ARGUMENTS:
!     FileName          -   Root File Name for weather and data files
!     Cond               -   array of wall conductivities
!     N_Conc             -   Number of conductivities in array
!     Thickness          -   array of wall Thicknesses
!     N_Thickness        -   Number of Thicknesses in array
!     Emissivity         -   surface long wavelength emissivity
!     T_outside          -   Outside air temperature
!     T_inside           -   Inside air temperature
!     T_sky              -   Sky temperature
!     Hin_A              -   Conv. correlation coefficient A for inside
!     Hin_C              -   Conv. correlation coefficient C for inside
!     Hin_n              -   Conv. correlation coefficient n for inside
!     Hout_A             -   Conv. correlation coefficient A for outside
!     Hout_C             -   Conv. correlation coefficient C for outside
!     Hout_n             -   Conv. correlation coefficient n for outside
!     Const_SB           -   Stefan-Boltzman constant,
!                           =5.6697*10.0**(-8) in SI; =0.1714*10.0**(-8)
in IP
!     Sur_area          -   surface area
!     IPUnit            -   logical variable, is true if user specify IP
unit
!     ErrorFlag         -   for error checking
!     ErrorMessage      -   for error reporting

IMPLICIT NONE
! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Max_Iterations = 500
INTEGER, PARAMETER :: Results = 21 ! unit number for analytical response
output file
INTEGER, PARAMETER :: N_Hours=8760      ! No. of hours in year
REAL(prec2), PARAMETER :: Tolerance = 1.0E-06 ! tolerance
INTEGER, PARAMETER :: N_roots=1        ! No. of roots of aux. equation

```

```

!Real(prec2),PARAMETER  :: Const_SB=5.6697*10.0**(-8) ! Stefan-Boltzman
constant
  REAL(prec2), PARAMETER :: Delta_H = 1.0E-20
                                ! a small number added to the initial value of
convection
                                ! coefficient to avoid zero convection
coefficients

! arguments
INTEGER, INTENT(IN)          :: N_Cond, N_thickness
CHARACTER *(*), INTENT(IN)   :: FileName
REAL(prec2), DIMENSION(N_Cond), INTENT(IN)      :: Cond
REAL(prec2), DIMENSION(N_thickness), INTENT(IN) :: Thickness
REAL(prec2), INTENT(IN)      :: Emissivity, Sur_area, Const_SB
REAL(prec2), INTENT(IN)      :: T_outside, T_inside, T_sky
REAL(prec2), INTENT(IN)      :: Hin_A, Hin_C, Hin_n
REAL(prec2), INTENT(IN)      :: Hout_A, Hout_C, Hout_n
CHARACTER*(*), INTENT(OUT)    :: ErrorMessage
INTEGER(2), INTENT(OUT)       :: ErrorFlag ! two byte int for VB integer
LOGICAL, INTENT(in)          :: IPUnit

! local variables
REAL(prec2) :: Fabric_R ! Slab thermal resistance (surface to surface)
!REAL(prec2) :: R_total ! Overall thermal resistance (air to air)
REAL(prec2) :: Flux, Load
REAL(prec2) :: Ts_in, Ts_out
REAL(prec2) :: H_in, H_out
REAL(prec2) :: Flux_inside ! Flux calculated between air and surface
inside
  REAL(prec2) :: Flux_outside ! Flux calculated between outside surface and
air
  REAL(prec2) :: Flux_fabric ! Flux calculated between outside surface and
inside surface
  REAL(prec2) :: MinFlux, MaxFlux ! Min and max calculated flux
values
  INTEGER :: i, n, error
  CHARACTER(LEN=132) :: dummy
  INTEGER, PARAMETER :: DebugUnit = 20
  LOGICAL, PARAMETER :: Debug = .false.
  LOGICAL, PARAMETER :: Warning = .false.

REAL(prec2), DIMENSION(N_roots) :: Beta ! array of aux. eqn. roots
REAL(prec2), DIMENSION(8)       :: Param ! array of aux. eqn. parameters

!Sur_area=9.0
! do some initial error checking
IF(N_Thickness /= N_Cond)THEN
  ErrorMessage = "Fabric data is inconsistant"
  ErrorFlag = 1
  RETURN
END IF
!IF(ABS(T_outside - T_inside ) < Tolerance)THEN
  !ErrorMessage = "Inside and outside temperatures appear to be the same
??"
  !ErrorFlag = 1

```

```

        !RETURN
    !END IF

    ! open a unit for debug output
    IF(Debug .and. DebugUnit/=6)THEN
        OPEN (DebugUnit, file="ExtLWRad.log", status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrorMessage="Debug output file could not be opened"
            ErrorFlag = 1
            RETURN
        END IF
        ! copy out arguments to debug file
        WRITE(DebugUnit,'("ExtLWRad arguments:")')
        WRITE(DebugUnit,'("Inside convection correlation coefficient A =", &
            F10.6)') Hin_A
        WRITE(DebugUnit,'("Inside convection correlation coefficient C =", &
            F10.6)') Hin_C
        WRITE(DebugUnit,'("Inside convection correlation coefficient n =", &
            F10.6)') Hin_n
        WRITE(DebugUnit,'("Outside convection correlation coefficient A =", &
            F10.6)') Hout_A
        WRITE(DebugUnit,'("Outside convection correlation coefficient C =", &
            F10.6)') Hout_C
        WRITE(DebugUnit,'("Outside convection correlation coefficient n =", &
            F10.6)') Hout_n
        WRITE(DebugUnit,'("Outside air temperature                    =", &
            F10.6)') T_outside
        WRITE(DebugUnit,'("Inside air temperature                    =", &
            F10.6)') T_inside
        DO i=1,N_Conf
            WRITE(DebugUnit,'("Fabric layer",I2," conductivity
=>=", &
                F10.6)') i, Cond(i)
            WRITE(DebugUnit,'("Fabric layer",I2," thickness
=>=", &
                F10.6)') i, Thickness(i)
        END DO
    END IF

    ! set initial guess of surface temperatures
    Ts_out = T_outside - 0.33*(T_outside - T_inside)
    Ts_in  = T_outside - 0.67*(T_outside - T_inside)
    ! calculate the resistance of the slab
    Fabric_R = 0.0
    DO i = 1, N_Conf
        IF(Cond(i)==0.0)THEN
            WRITE(ErrorMessage,'("Conductivity of layer ", I2," appears to be
zero")') i
            ErrorFlag = 1
            RETURN
        END IF
        Fabric_R = Fabric_R + Thickness(i)/Cond(i)
    END DO

    H_in = Hin_A + Hin_C*ABS(T_inside - Ts_in)**Hin_n + Delta_H
    H_out = Hout_A + Hout_C*ABS(T_outside - Ts_out)**Hout_n + Delta_H

```



```

! assign aux. eqn. parameters
Param(1)=Emissivity
Param(2)=Const_SB
Param(5)=T_outside
Param(6)=T_inside
Param(7)=T_sky
Param(8)=Fabric_R

! start loop to find Flux by successive substitution
n = 1
DO
  ! calculate total resistance
  IF(H_in==0.0 .or. H_out==0.0)THEN
    WRITE(ErrorMessage, '("Inside or outside convection coefficient &
procede"))')
    ErrorFlag = 1
    RETURN
  END IF
  !R_total = 1.0/H_in + Fabric_R + 1.0/H_out

  ! assign aux. eqn. parameters
  Param(3)=h_out
  Param(4)=h_in

  ! find roots of auxiliary equation
  CALL FindTransRoots(f1, Param, Beta, N_roots, Debug)

  Ts_out=Beta(N_roots)

  ! flux based on current value of total resistance
  !Flux = (T_outside - T_inside) / R_total
  ! update surface temps based on flux
  !Ts_out = T_outside - Flux/H_out
  Ts_in =
Ts_out/(Fabric_R*h_in+1)+T_inside*Fabric_R*h_in/(Fabric_R*h_in+1)

  ! re-calculate conv. coef using current temps
  H_in = Hin_A + Hin_C*(ABS(T_inside - Ts_in)**Hin_n)
  H_out = Hout_A + Hout_C*(ABS(T_outside - Ts_out)**Hout_n)

  ! calculate fluxes over different resistances for checking
  Flux_inside = H_in*(Ts_in - T_inside)
  Flux_outside = H_out*(T_outside -
Ts_out)+Emissivity*Const_SB*(T_sky**4-Ts_out**4)
  Flux_fabric = (Ts_out - Ts_in) / Fabric_R

  ! some debug output
  IF(Debug)THEN
    WRITE(DebugUnit, '("H_in = ",F8.5," H_out = ",F8.5," Ts_in =
",F8.5, &
" Ts_out = ",F8.5)') H_in, H_out, Ts_in, Ts_out
    !WRITE(DebugUnit, '("Q = ",F8.5," Qout = ",F8.5," Qin =",F8.5,"
Qfab = ",F8.5)') &

```

```

        !Flux, Flux_outside, Flux_inside, Flux_fabric
        WRITE(DebugUnit,'(" Qout = ",F8.5," Qin = ",F8.5," Qfab = ",F8.5)')
&
        Flux_outside, Flux_inside, Flux_fabric
    END IF

    ! calc min and max fluxes
    !MinFlux = MIN(Flux, Flux_inside, Flux_outside, Flux_fabric)
    !MaxFlux = MAX(Flux, Flux_inside, Flux_outside, Flux_fabric)
    MinFlux = MIN(Flux_inside, Flux_outside, Flux_fabric)
    MaxFlux = MAX(Flux_inside, Flux_outside, Flux_fabric)

    ! test for exit condition
    IF(ABS(MaxFlux-MinFlux) < Tolerance .or. n > Max_Iterations) EXIT
    n = n + 1
END DO

! set warning if max iterations exceded
IF(n > Max_Iterations)THEN
    ErrorMessage = "The steady-state heat flux could not be calculated &
                    within the allowable number of iterations"
    ErrorFlag = -1
    RETURN
END IF

flux=flux_fabric
! calculate the zone load
Load=Sur_area*Flux

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results,'("# ASHRAE Analytical Test Suite - Results file")')
WRITE(Results,'("# External long wave radiation test")')
WRITE(Results,'("#")')
WRITE(Results,'(// "Test Parameters -")')

WRITE(Results,'("Inside convection correlation coefficient A
=","",F15.4)') Hin_A
WRITE(Results,'("Inside convection correlation coefficient C
=","",F15.4)') Hin_C
WRITE(Results,'("Inside convection correlation coefficient n
=","",F15.4)') Hin_n
WRITE(Results,'("Outside convection correlation coefficient A
=","",F15.4)') Hout_A
WRITE(Results,'("Outside convection correlation coefficient C
=","",F15.4)') Hout_C
WRITE(Results,'("Outside convection correlation coefficient n
=","",F15.4)') Hout_n

```

```

! transfer the absolute temperature to C/F scale in output
IF (IPUnit) THEN
  WRITE(Results, '( "Outside air temperature =", "", "", F15.4)') T_outside-
459.67
  WRITE(Results, '( "Inside air temperature =", "", "", F15.4)') T_inside-
459.67
  WRITE(Results, '( "Sky temperature =", "", "", F15.4)') T_sky-459.67
ELSE
  WRITE(Results, '( "Outside air temperature =", "", "", F15.4)') T_outside-
273.15
  WRITE(Results, '( "Inside air temperature =", "", "", F15.4)') T_inside-
273.15
  WRITE(Results, '( "Sky temperature =", "", "", F15.4)') T_sky-273.15
END IF

WRITE(Results, '( "Surface emissivity =", "", "", F15.4)') Emissivity
WRITE(Results, '( "Surface area =", "", "", F15.4)') Sur_area

DO i=1,N_Conc
  WRITE(Results, '( "Fabric layer", I2, " conductivity =", "", "", &
    F15.4)') i, Cond(i)
  WRITE(Results, '( "Fabric layer", I2, " thickness =", "", "", &
    F15.4)') i, Thickness(i)
END DO

WRITE(Results, '( //"Test results -")')
WRITE(Results, '( "Heat flux =", "", "", F15.4)') Flux
WRITE(Results, '( "Zone load =", "", "", F15.4)') Load
WRITE(Results, '( "Inside heat flux =", "", "", F15.4)') Flux_inside
WRITE(Results, '( "Outside heat flux =", "", "", F15.4)') Flux_outside
WRITE(Results, '( "Fabric heat flux =", "", "", F15.4)') Flux_fabric

! transfer the absolute temperature to C/F scale in output
IF (IPUnit) THEN
  WRITE(Results, '( //"Inside surface temperature =", "", "", F15.4)') Ts_in-
459.67
  WRITE(Results, '( "Outside surface temperature =", "", "", F15.4)') Ts_out-
459.67
ELSE
  WRITE(Results, '( //"Inside surface temperature =", "", "", F15.4)') Ts_in-
273.15
  WRITE(Results, '( "Outside surface temperature =", "", "", F15.4)') Ts_out-
273.15
END IF

WRITE(Results, '( "Inside convection coefficient =", "", "", F15.4)') H_in
WRITE(Results, '( "Outside convection coefficient =", "", "", F15.4)') H_out

CLOSE(Results)
IF(Debug)THEN
  CLOSE(DebugUnit)
END IF

END SUBROUTINE ExtLWRadiation
!-----
-

```

```

FUNCTION f1(X, Param)

! PURPOSE:      Defines transcendental function with respect to outside
!               surface temperature used in analytical solution
!               to external long wave radiation test case
!
! CREATED:
!   7.7.2000      Dongyi Xiao
!
! MODIFIED:
!

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)

REAL(prec) :: f1

REAL(prec), INTENT(IN) :: X
REAL(prec), DIMENSION(8) :: Param
REAL(prec) :: Emiss, Const_SB, hi, ho, Ta_o, Ta_i, Tsky, R_fabric

Emiss = Param(1)
Const_SB = Param(2)
ho = Param(3)
hi = Param(4)
Ta_o = Param(5)
Ta_i = Param(6)
Tsky = Param(7)
R_fabric = Param(8)

f1 = Emiss*Const_SB*X**4 + ((R_fabric*ho+1)/R_fabric - 1/R_fabric/ &
    (R_fabric*hi+1))*X - hi*Ta_i/(R_fabric*hi+1) - Emiss*Const_SB*Tsky**4 -
ho*Ta_o

END FUNCTION f1

END MODULE ExtLWRad

```

Source Code: Module Infilt_1

```

MODULE Infilt_1

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical      solutions to the infiltration test with fixed infiltration
!               rate.
!
! CREATED:      7.10.00      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!

USE PsychrometricsMod

PRIVATE
! make public main subroutine
PUBLIC :: Infiltration_1

CONTAINS

!-----
-

SUBROUTINE Infiltration_1(FileName, T_outside, T_inside, W_out, TDP_out,
RH_out, &
                        Volume_flow, IPUnit, ErrorFlag, ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "Infiltration_1" :: Infiltration_1

! PURPOSE:      Produces weather files and analytical response for the
case
!               of infiltration with fixed infiltration rate.
!

```

```

! CREATED:
!   7.10.00      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   N/A
!
! OUTPUT FILES:
!
!   FileName.csv      -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   T_outside          -   Outside air temperature
!   T_inside          -   Inside air temperature
!   W_out             -   humidity ratio of outside air
!   TDP_out           -   Dewpoint temperature of outside air
!   RH_out            -   outside relative humidity
!   Volume_flow       -   Volumetric infiltration flow rate
!   IPUnit            -   logical variable, is true if user specify IP
unit
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE

!parameters
INTEGER, PARAMETER    :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER    :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER    :: Results=3 ! unit number for analytical response
output file
REAL(prec2), PARAMETER :: Const_Grav=9.8 ! gravitational constant
Real(prec1), PARAMETER :: Patms=101325 ! atmospheric pressure

!arguments
REAL(prec1)           :: T_outside, T_inside
REAL(prec1), INTENT(in)      :: W_out
REAL(prec2)           :: Volume_flow
REAL(prec1), INTENT(out)     :: TDP_out, RH_out

CHARACTER*(*), INTENT(in)    :: FileName
CHARACTER*(*), INTENT(out)   :: ErrorMessage
INTEGER(2), INTENT(out)     :: ErrorFlag ! two byte int for VB
integer

LOGICAL, INTENT(in)         :: IPUnit

!local variables
INTEGER    :: i,error,ErrStat
REAL(prec1) :: Dens_out,Dens_in ! density of the inside and outside moist
air
REAL(prec1) :: Enthalpy_out,Enthalpy_in ! enthalpy of the inside and
outside air
REAL(prec1) :: W_in ! humidity ratio of inside air
REAL(prec1) :: TWB_out, TWB_in ! Wet bulb temperature of the air
REAL(prec1) :: TDP_in ! Dewpoint temperature of inside air

```

```

REAL(prec1) :: RhoD_out, RhoD_in ! Dry air density of the air
REAL(prec1) :: RH_in ! Relative humidity of the inside air
REAL(prec2) :: Mass_flow
REAL(prec2) :: Load

W_in=W_out
! Calc the air density and enthalpy by calling the Psychrometric routines
!CALL TDB_RH
(Patms,T_outside,W_out,RH_out,Enthalpy_out,TWB,TDP,RhoD,Dens_out,ErrStat)
!CALL TDB_RH
(Patms,T_inside,W_in,RH_in,Enthalpy_in,TWB,TDP,RhoD,Dens_in,ErrStat)
CALL TDB_W (Patms,T_outside,W_out,RH_out,Enthalpy_out,TWB_out,TDP_out, &
RhoD_out,Dens_out,ErrStat)
CALL TDB_W
(Patms,T_inside,W_in,RH_in,Enthalpy_in,TWB_in,TDP_in,RhoD_in,Dens_in,ErrStat)

! calc mass flow rate of the air
Mass_flow=Dens_out*Volume_flow

! calc the zone load
Load=Mass_flow*(Enthalpy_in-Enthalpy_out)

! transfer the outputs to IP unit if IP is specified
IF (IPUnit) THEN
  T_outside=T_outside*1.8+32.0
  T_inside=T_inside*1.8+32.0
  Volume_flow=Volume_flow*2119.0
  Dens_out=Dens_out*0.0625
  Enthalpy_out=Enthalpy_out/1000/2.33
  Enthalpy_in=Enthalpy_in/1000/2.33
  Mass_flow=Mass_flow*7937.0
  load=load*3.412
END IF

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
  ErrorMessage="Analytical results output file could not be opened"
  ErrorFlag = 1
  RETURN
END IF

! write header lines
WRITE(Results,'("# ASHRAE Analytical Test Suite - Results file")')
WRITE(Results,'("# Infiltration test case - fixed infiltration rate")')
WRITE(Results,'(// "Test Parameters -")')

WRITE(Results,'("Outside air temperature =",",",F15.4)') T_outside
WRITE(Results,'("Inside air temperature =",",",F15.4)') T_inside
WRITE(Results,'("Outside humidity ratio =",",",F15.4)') W_out
WRITE(Results,'("Volumetric infiltration flow rate =",",",F15.4)')
Volume_flow

WRITE(Results,'(// "Test results -")')
WRITE(Results,'("Outside air density =",",",F15.4)') Dens_out
WRITE(Results,'("Outside air enthalpy =",",",F15.4)') Enthalpy_out

```

```
WRITE(Results, '(' "Inside air enthalpy =", ", ", ", ", F15.4) ') Enthalpy_in
WRITE(Results, '(' "/" "Mass flow rate of the air =", ", ", ", ", F15.4) ') Mass_flow
WRITE(Results, '(' "/" "Zone Load =", ", ", ", ", F15.4) ') load

CLOSE(Results)

END SUBROUTINE Infiltration_1

END MODULE Infilt_1
```


Source Code: Module Infilt_2

```

MODULE Infilt_2

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical
!               solutions to the infiltration test case with fixed openings
!               (infiltration 2).
!
! CREATED:      2.4.00      Dongyi Xiao
!
! MODIFIED:
!

USE PsychrometricsMod

PRIVATE
! make public main subroutine
PUBLIC :: Infiltration_2

CONTAINS

!-----
-

SUBROUTINE Infiltration_2(FileName,T_outside,T_inside,W_out,TDP_out,
RH_out, &
                        H_wall,H_open,W_open,IPUnit, ErrorFlag,
ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "Infiltration_2" :: Infiltration_2

! PURPOSE:      Produces weather files and analytical response for the
case
!               of infiltration with fixed openings.

```

```

!
! CREATED:
!   4.9.00      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   N/A
!
! OUTPUT FILES:
!
!   FileName.csv      -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for weather and data files
!   T_outside         -   Outside air temperature
!   T_inside          -   Inside air temperature
!   W_out             -   Outside humidity ratio
!   TDP_out           -   Outside dew point temperature
!   RH_out            -   outside relative humidity
!   H_wall            -   Height of wall of the test zone
!   H_open            -   Height of the opening
!   W_open            -   Width of the opening
!   IPUnit            -   logical variable, is true if user specify IP
unit
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE

!parameters
INTEGER, PARAMETER    :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER    :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER    :: Results=3 ! unit number for analytical response
output file
REAL(prec2), PARAMETER :: Const_Grav=9.8 ! gravitational constant
Real(prec1), PARAMETER :: Patms=101325 ! atmospheric pressure
REAL(prec2), PARAMETER :: Coeff_D=0.6
REAL(prec2), PARAMETER :: Expon_flow=0.65

!arguments
REAL(prec1)           :: T_outside, T_inside
REAL(prec1), INTENT(in) :: W_out
REAL(prec2), INTENT(in) :: H_wall, H_open, W_open
REAL(prec1), INTENT(out) :: TDP_out, RH_out

CHARACTER*(*) , INTENT(in)    :: FileName
CHARACTER*(*) , INTENT(out)   :: ErrorMessage
INTEGER(2) , INTENT(out)      :: ErrorFlag ! two byte int for VB
integer

LOGICAL, INTENT(in)           :: IPUnit

!local variables
INTEGER      :: i,error,ErrStat
REAL(prec2) :: Area_open ! area of the opening

```

```

    REAL(prec2) :: Pressure_diff1, Pressure_diff2 ! pressure difference at
opening 1&2
    REAL(prec2) :: Coeff_flow ! flow coefficient of the opening
    REAL(prec1) :: Dens_out, Dens_in ! density of the inside and outside air
    REAL(prec1) :: Enthalpy_out, Enthalpy_in ! enthalpy of the inside and
outside air
    REAL(prec1) :: W_in ! humidity ratio of inside air
    REAL(prec1) :: RH_in ! relative humidity of the inside air
    REAL(prec1) :: TWB_out, TWB_in ! Wet bulb temperature of the air
    REAL(prec1) :: TDP_in ! Dewpoint temperature of inside air
    REAL(prec1) :: RhoD_out, RhoD_in ! Dry air density of the air
    REAL(prec2) :: H_NPL
    REAL(prec2) :: Mass_flow1, Mass_flow2
    REAL(prec2) :: Load

    ! Calc the area and flow coefficient of the opening
    Area_open=H_open*W_open
    Coeff_flow=Coeff_D*Area_open**2**0.5
    W_in=W_out

    ! Calc the air density and enthalpy by calling the Psychrometric routines
    CALL TDB_W (Patms, T_outside, W_out, RH_out, Enthalpy_out, TWB_out, &
                TDP_out, RhoD_out, Dens_out, ErrStat)
    CALL TDB_W (Patms, T_inside, W_in, RH_in, Enthalpy_in, TWB_in, &
                TDP_in, RhoD_in, Dens_in, ErrStat)

    ! calculate the height of neutral pressure level
    H_NPL=((H_wall-H_open/2)*(Dens_in/Dens_out)**(1/(2*Expon_flow)))+ &
          H_open/2)/(1+(Dens_in/Dens_out)**(1/(2*Expon_flow)))

    ! Calc the pressure difference at the opening
    Pressure_diff1=(Dens_out-Dens_in)*Const_Grav*(H_wall-H_NPL-H_open/2)
    Pressure_diff2=(Dens_out-Dens_in)*Const_Grav*(H_NPL-H_open/2)

    ! calc mass flow rate through the opening
    Mass_flow1=Coeff_flow*Dens_in**0.5*Pressure_diff1**Expon_flow
    Mass_flow2=Coeff_flow*Dens_out**0.5*Pressure_diff2**Expon_flow

    ! calc the zone load
    Load=Mass_flow1*(Enthalpy_in-Enthalpy_out)

    ! transfer the outputs to IP unit if IP is specified
    IF (IPUnit) THEN
        T_outside=T_outside*1.8+32.0
        T_inside=T_inside*1.8+32.0
        Mass_flow1=Mass_flow1*7937.0
        Mass_flow2=Mass_flow2*7937.0
        H_NPL=H_NPL/0.3048
        load=load*3.412
    END IF

    !attempt to open file - full path is passed as FileName by VB application
    OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
    IF(error/=0) THEN
        ErrorMessage="Analytical results output file could not be opened"
        ErrorFlag = 1
    
```

```
        RETURN
    END IF

    ! write header lines
    WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )')
    WRITE(Results, '( "# Infiltration test case" )')
    WRITE(Results, '( // "Test Parameters -" )')

    WRITE(Results, '( "Outside air temperature = ", " ", " ", F15.4 )') T_outside
    WRITE(Results, '( "Inside air temperature = ", " ", " ", F15.4 )') T_inside
    WRITE(Results, '( "Outside humidity ratio = ", " ", " ", F15.4 )') W_out

    WRITE(Results, '( "Discharge coefficient of the opening = ", " ", " ", F15.4 )')
    Coeff_D
    WRITE(Results, '( "Flow exponent = ", " ", " ", F15.4 )') Expon_flow

    WRITE(Results, '( // "Test results -" )')
    WRITE(Results, '( // "Mass flow through opening 1 = ", " ", " ", F15.4 )') Mass_flow1
    WRITE(Results, '( // "Mass flow through opening 2 = ", " ", " ", F15.4 )') Mass_flow2
    WRITE(Results, '( // "Height of neutral pressure level = ", " ", " ", F15.4 )') H_NPL
    WRITE(Results, '( // "Zone Load = ", " ", " ", F15.4 )') load

    CLOSE(Results)

END SUBROUTINE Infiltration_2

END MODULE Infilt_2
```

Source Code: Module PsychrometricsMod

```

MODULE PsychrometricsMod

! ** PURPOSE OF THIS MODULE:
!   Collection of psychrometric subroutines

! ** USE STATEMENTS:
!   none

! ** ACCESSIBLE SPECIFICATIONS OF MODULE SUBROUTINES OR FUNCTIONS:
PUBLIC   TDB_H      !Calculate props of moist air given dry bulb & enthalpy.
PUBLIC   TDB_RH     !Calculate props of moist air given dry bulb & Relative
Humidity
PUBLIC   TDB_TW     !Calculate props of moist air given dry bulb & Wet Bulb
PUBLIC   TDB_W      !Calculate props of moist air given dry bulb & Humidity
Ratio
PUBLIC   SATPRESS   !REAL FUNCTION SATPRESS (T) !Calculate saturation
pressure
                        ! of water vapor as a function of temperature
PUBLIC   DEWPOINT   !REAL FUNCTION DEWPOINT (W)
PRIVATE  DRYBULB    !REAL FUNCTION DRYBULB (H,W)
PRIVATE  ENTHALPY    !REAL FUNCTION ENTHALPY (TDB,W)
PRIVATE  ENTHSAT     !REAL FUNCTION ENTHSAT (TDB)
PRIVATE  HUMRATIO    !REAL FUNCTION HUMRATIO (Patm,Pw)
PRIVATE  HUMTH       !REAL FUNCTION HUMTH (TDB,H)
PRIVATE  RELHUM      !REAL FUNCTION RELHUM (Patm,Psat,HumRatio)
PRIVATE  RHODRY      !REAL FUNCTION RHODRY (TDB,W)
PRIVATE  RHOMOIST    !REAL FUNCTION RHOMOIST (RhoDry,W)
PRIVATE  SATTEMP     !REAL FUNCTION SATTEMP (P)
PRIVATE  TAIRSAT     !REAL FUNCTION TAIRSAT (HSat)
PRIVATE  WETBULB     !REAL FUNCTION WETBULB (TDB,W)
PRIVATE  XITERATE    !REAL FUNCTION XITERATE (X0,F0,X1,F1,X2,F2,ICount,ICvg)

! ** MODULE PARAMETER DEFINITIONS:
INTEGER, PRIVATE, PARAMETER :: Patm      = 1
INTEGER, PRIVATE, PARAMETER :: CpAir     = 2
INTEGER, PRIVATE, PARAMETER :: CpWat     = 3
INTEGER, PRIVATE, PARAMETER :: CpVap     = 4
INTEGER, PRIVATE, PARAMETER :: CpLiq     = 5
INTEGER, PRIVATE, PARAMETER :: DViscAir  = 6
INTEGER, PRIVATE, PARAMETER :: DViscLiq  = 7
INTEGER, PRIVATE, PARAMETER :: KAir      = 8
INTEGER, PRIVATE, PARAMETER :: KLiq      = 9
INTEGER, PRIVATE, PARAMETER :: RhoLiq    = 10
INTEGER, PRIVATE, PARAMETER :: Hfg       = 11
INTEGER, PRIVATE, PARAMETER :: RAir      = 12
INTEGER, PRIVATE, PARAMETER :: TKelMult  = 13
INTEGER, PRIVATE, PARAMETER :: TAbsAdd   = 14
INTEGER, PRIVATE, PARAMETER :: PaMult    = 15
INTEGER, PRIVATE, PARAMETER :: PAbsAdd   = 16

```

```

! ** MODULE VARIABLE DECLARATIONS:
  REAL, PRIVATE, Dimension(16) :: Prop
    DATA Prop(Patm) /101325./
    DATA Prop(CpAir) /1006./
    DATA Prop(CpWat) /4186./
    DATA Prop(CpVap) /1805./
    DATA Prop(CpLiq) /4186./
    DATA Prop(DViscAir) /0.0000182/
    DATA Prop(DViscLiq) /0.00144/
    DATA Prop(KAir) /0.0260/
    DATA Prop(KLiq) /0.604/
    DATA Prop(RhoLiq) /998./
    DATA Prop(Hfg) /2501000./
    DATA Prop(RAir) /287.055/
    DATA Prop(TKelMult) /1./
    DATA Prop(TAbsAdd) /273.15/
    DATA Prop(PaMult) /1./
    DATA Prop(PAbsAdd) /0./

! **** INTERNAL VARIABLE DICTIONARY:
!   Patm           Atmospheric pressure                (Pa)
!   CpAir          Specific heat of dry air             (J/kg C)
!   CpLiq          Specific heat of liquid water        (J/kg C)
!   CpVap          Specific heat of saturated water vapor (J/kg C)
!   DViscAir       Air dynamic viscosity               (kg/m s)
!   DViscLiq       Liquid dynamic viscosity            (kg/m s)
!   KAir           Air thermal conductivity            (W/m C)
!   KLiq           Liquid thermal conductivity          (W/m C)
!   RhoLiq         Liquid density                      (kg/m3)
!   Hfg            Latent heat of vaporization of water (J/kg)
!   RAir           Gas constant for air                (J/kg C)
!   TKelMult       Multiplying factor to convert user T to Kelvin
!   TAbsAdd        Additive factor to convert user P to Kelvin
!   tKel = Prop(TKelMult)*T + Prop(TKelAdd)
!   PaMult         Multiplying factor to convert user P to Pascals
!   PAbsAdd        Additive factor to convert user P to Pascals
!   Pa = Prop(PaMult)*P + Prop(PaAdd)
! *****

CONTAINS

! ** MODULE ELEMENTS SUBROUTINES or FUNCTIONS:

! *****
*

SUBROUTINE TDB_H ( Patms , TDB , W , RH , H , TWB , TDP , RhoD , RhoM , ErrStat )

! *****
! *   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
! *****
! *   SUBROUTINE:                TDB_H
! *
! *   LANGUAGE:                  FORTRAN 90
! *
! *   PURPOSE:                   Calculate psychrometric properties of

```

```

!*                               moist air given dry bulb temperature
!*                               and enthalpy.
!*****
!*   INPUT VARIABLES
!*   TDB           Dry bulb temperature           (C)
!*   H             Enthalpy of moist air          (J/kg)
!*
!*   OUTPUT VARIABLES
!*   W             Humidity ratio                 (-)
!*   RH            Relative humidity              (-)
!*   TWB           Wet bulb temperature          (C)
!*   TDP           Dewpoint temperature          (C)
!*   RhoD          Dry air density               (kg/m3)
!*   RhoM          Moist air density             (kg/m3)
!*   ErrStat       Error flag (0=ok, 1=error)     (-)
!*
!*   PROPERTIES
!*   Patm          Atmospheric pressure           (Pa)
!*   CpAir         Specific heat of air           (J/kg C)
!*   CpVap         Specific heat of water vapor   (J/kg C)
!*   Hfg           Reference heat of vaporization of water (J/kg)
!*****
!   MAJOR RESTRICTIONS:      Perfect gas relationships
!
!   DEVELOPER:               Shauna Gabel
!                           Michael J. Brandemuehl, PhD, PE
!                           University of Colorado at Boulder
!
!   DATE:                    January 1, 1992
!
!   SUBROUTINES CALLED:      None
!   FUNCTIONS CALLED:        SATPRESS
!                           RELHUM
!                           WETBULB
!                           DEWPOINT
!                           RHODRY
!                           RHOMOIST
!
!   REFERENCE:               1989 ASHRAE Handbook - Fundamentals
!*****

    INTEGER ErrStat
    Real Patms

    Prop(Patm) = Patms

    ErrStat = 0

!*** Calculate the saturation pressure at a given temperature.

    psat = SATPRESS (TDB)

!*** Calculate the humidity ratio as a function of dry bulb
!*** temperature and enthalpy.

    W = (H-Prop(CpAir)*TDB)/(Prop(Hfg)+Prop(CpVap)*TDB)

```

```

!*** Calculate the relative humidity as a function of partial water
!*** vapor pressure and saturation pressure.

      RH = RELHUM (Prop(Patm),psat,W)

!*** Calculate wet bulb temperature as a function of enthalpy,
!*** dry bulb temperature and humidity ratio.

      TWB = WETBULB (TDB,W)

!*** Calculate dewpoint temperature as a function of humidity ratio.

      TDP = DEWPOINT (W)

!*** Calculate dry and moist air density.

      RhoD = RHODRY (TDB,W)
      RhoM = RHOMOIST (RhoD,W)

      RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END SUBROUTINE TDB_H

!*****
*

SUBROUTINE TDB_RH (Patms,TDB,W,RH,H,TWB,TDP,RhoD,RhoM,ErrStat)

!*****
!*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!*   SUBROUTINE:                TDB_RH
!*
!*   LANGUAGE:                  FORTRAN 90
!*
!*   PURPOSE:                   Calculate psychrometric properties of
!*                               moist air with input of dry bulb
!*                               temperature and relative humidity.
!*****
!*   INPUT VARIABLES
!*   TDB                        Dry bulb temperature                (C)
!*   RH                        Relative humidity                    (-)
!*
!*   OUTPUT VARIABLES
!*   W                        Humidity ratio                        (-)
!*   H                        Enthalpy                            (J/kg)
!*   TWB                      Wet bulb temperature                (C)
!*   TDP                      Dewpoint temperature                (C)
!*   RhoD                     Dry air density                     (kg/m3)
!*   RhoM                     Moist air density                   (kg/m3)
!*   ErrStat                  Error flag (0=ok, 1=error)           (-)
!*
!*   PROPERTIES

```



```

!*      Patm              Atmospheric pressure                      (Pa)
!*****
!      MAJOR RESTRICTIONS:      None
!
!      DEVELOPER:                Shauna Gabel
!                                Michael J. Brandemuehl, PhD, PE
!                                University of Colorado at Boulder
!
!      DATE:                     January 1, 1992
!
!      SUBROUTINES CALLED:      None
!      FUNCTIONS CALLED:        SATPRESS
!                                HUMRATIO
!                                WETBULB
!                                ENTHALPY
!                                DEWPOINT
!                                RHODRY
!                                RHOMOIST
!
!      REVISION HISTORY:        None
!
!      REFERENCE:                1989 ASHRAE Handbook - Fundamentals
!*****
!      INTERNAL VARIABLES:
!      pw                      Partial water vapor pressure          (Pa)
!*****

      INTEGER ErrStat
      Real Patms

      Prop(Patm) = Patms

      ErrStat = 0

!*** Calculate the saturation pressure at a given temperature.

      psat = SATPRESS (TDB)

!*** Calculate the partial water vapor pressure as a function of the
!*** saturation pressure and relative humidity.

      pw = RH*psat

!*** Calculate the humidity ratio as a function of the partial water
!*** vapor pressure.

      W = HUMRATIO (Prop(Patm),Pw)

!*** Calculate the Halpy as a function of dry bulb temperature
!*** and humidity ratio.

      H = ENTHALPY (TDB,W)

!*** Calculate the wet bulb temperature as a function of dry bulb
!*** temperature and humidity ratio.

```

```

      TWB = WETBULB (TDB,W)

!*** Calculate the dewpoint temperature as a function of the humidity
!*** ratio.

      TDP = DEWPOINT (W)

!*** Calculate dry and moist air densities.

      RhoD = RHODRY (TDB,W)
      RhoM = RHOMOIST (RhoD,W)

      RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END SUBROUTINE TDB_RH

!*****
*

SUBROUTINE TDB_TW (Patms,TDB,W,RH,H,TWB,TDP,RhoD,RhoM,ErrStat)

!*****
!*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!*   SUBROUTINE:                TDB_TW
!*
!*   LANGUAGE:                  FORTRAN 90
!*
!*   PURPOSE:                   Calculate psychrometric properties of
!*                               moist air given dry bulb temperature
!*                               and wet bulb temperature.
!*****
!*   INPUT VARIABLES
!*   TDB                        Dry bulb temperature                (C)
!*   TWB                        Wet bulb temperature                (C)
!*
!*   OUTPUT VARIABLES
!*   W                          Humidity ratio                    (-)
!*   RH                         Relative humidity                  (-)
!*   H                          Enthalpy of air                    (J/kg)
!*   TDP                        Dewpoint temperature              (C)
!*   RhoD                       Dry air density                   (kg/m3)
!*   RhoM                       Moist air density                  (kg/m3)
!*   ErrStat                    Error flag (0=ok, 1=error)         (-)
!*
!*   PROPERTIES
!*   Patm                       Atmospheric pressure               (Pa)
!*   CpAir                      Specific heat of air                (J/kg C)
!*   CpVap                      Specific heat of water vapor        (J/kg C)
!*   CpWat                      Specific heat of liquid water        (J/kg C)
!*   Hfg                       Reference heat of vaporization of water (J/kg)
!*****
!*   MAJOR RESTRICTIONS:       None

```

```

!
!   DEVELOPER:           Shauna Gabel
!                       Michael J. Brandemuehl, PhD, PE
!                       University of Colorado at Boulder
!
!   DATE:                January 1, 1992
!
!   SUBROUTINES CALLED:  None
!   FUNCTIONS CALLED:    SATPRESS
!                       HUMRATIO
!                       RELHUM
!                       ENTHALPY
!                       DEWPOINT
!                       RHODRY
!                       RHOMOIST
!
!   REVISION HISTORY:    None
!
!   REFERENCE:           1989 ASHRAE Handbook - Fundamentals
! *****
!   INTERNAL VARIABLES:
!   psat                 Saturated water vapor pressure             (Pa)
!   wstar                Saturated humidity ratio at wet bulb temp. (-)
! *****

      INTEGER ErrStat
      Real Patms

      Prop(Patm) = Patms

      ErrStat = 0

!*** Calculate the saturation pressure for a given wet bulb
!*** temperature.

      psat = SATPRESS (TWB)

!*** Calculate the humidity ratio corresponding to saturation
!*** pressure for a given wet bulb temperature.

      wstar = HUMRATIO (Prop(Patm),psat)

!*** Calculate the humidity ratio as a function of dry bulb
!*** temperature, wet bulb temperature and the humidity ratio
!*** at saturation pressure for the given wet bulb temperature.

      W = ((Prop(Hfg)+(Prop(CpVap)-Prop(CpWat))*TWB)*wstar-
           Prop(CpAir)*(TDB-TWB))/
           (Prop(Hfg)+Prop(CpVap)*TDB-Prop(CpWat)*TWB)
&
&

!*** Calculate the saturation pressure for a given temperature.

      psat = SATPRESS (TDB)

!*** Calculate the relative humidity as a function of partial
!*** water vapor pressure and water vapor pressure at saturation.

```

```

      RH = RELHUM (Prop(Patm),psat,W)

!*** Calculate enthalpy as a function of dry bulb temperature
!*** and humidity ratio.

      H = ENTHALPY (TDB,W)

!*** Calculate the dewpoint temperature as a function of humidity
!*** ratio.

      TDP = DEWPOINT (W)

!*** Calculate dry and moist air densities.

      RhoD = RHODRY (TDB,W)
      RhoM = RHOMOIST (RhoD,W)

      RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END SUBROUTINE TDB_TWB

!*****
*

SUBROUTINE TDB_W (Patms,TDB,W,RH,H,TWB,TDP,RhoD,RhoM,ErrStat)

!*****
!*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!*   SUBROUTINE:                TDB_W
!*
!*   LANGUAGE:                  FORTRAN 90
!*
!*   PURPOSE:                   Calculate psychrometric properties of
!*                               moist air given dry bulb temperature
!*                               and humidity ratio.
!*****
!*   INPUT VARIABLES
!*   TDB                        Dry bulb temperature                (C)
!*   W                          Humidity ratio                    (-)
!*
!*   OUTPUT VARIABLES
!*   RH                         Relative humidity                  (-)
!*   H                          Enthalpy of moist air             (J/kg)
!*   TWB                        Wet bulb temperature              (C)
!*   TDP                        Dewpoint temperature              (C)
!*   RhoD                       Dry air density                  (kg/m3)
!*   RhoM                       Moist air density                 (kg/m3)
!*   ErrStat                    Error flag (0=ok, 1=error)         (-)
!*
!*   PROPERTIES
!*   Patm                       Atmospheric pressure              (Pa)

```

```

!*****
!      MAJOR RESTRICTIONS:      None
!
!      DEVELOPER:                Shauna Gabel
!                                Michael J. Brandemuehl, PhD, PE
!                                University of Colorado at Boulder
!
!      DATE:                     January 1, 1992
!
!      SUBROUTINES CALLED:       None
!      FUNCTIONS CALLED:         SATPRESS
!                                RELHUM
!                                WETBULB
!                                ENTHALPY
!                                DEWPOINT
!                                RHODRY
!                                RHOMOIST
!
!      REFERENCE:                1989 ASHRAE Handbook - Fundamentals
!*****

      INTEGER ErrStat
      Real Patms

      Prop(Patm) = Patms

      ErrStat = 0

!*** Calculate the saturation pressure at a given temperature.

      psat = SATPRESS (TDB)

!*** Calculate the relative humidity as a function of the partial
!*** pressure of water vapor and the saturation pressure of water vapor.

      RH = RELHUM (Prop(Patm),psat,W)

!*** Calculate enthalpy as a function of dry bulb temperature
!*** and humidity ratio.

      H = ENTHALPY (TDB,W)

!*** Calculate the wet bulb temperature as a function of enthalpy or
!*** dry bulb temperature and humidity ratio.

      TWB = WETBULB (TDB,W)

!*** Calculate the dewpoint temperature as a function of the humidity
!*** ratio.

      TDP = DEWPOINT (W)

!*** Calculate dry and moist air densities.

      RhoD = RHODRY (TDB,W)
      RhoM = RHOMOIST (RhoD,W)

```

```

      RETURN
!*****
!  EVOLUTIONARY HISTORY :
!  This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!  Energy Calculations which was originally written in FORTRAN 77.
!*****
END SUBROUTINE TDB_W

!*****
*

REAL FUNCTION DEWPOINT (W)

!*****
!*  Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!*  FUNCTION:                DEWPOINT
!*
!*  LANGUAGE:                FORTRAN 90
!*
!*  PURPOSE:                Calculate the dewpoint temperature given
!*                          humidity ratio
!*****
!*  INPUT VARIABLES
!*  W                        Humidity ratio                        (-)
!*
!*  OUTPUT VARIABLES
!*  DewPoint                Dew point temperature of air          (C)
!*
!*  PROPERTIES
!*  Patm                    Atmospheric pressure                  (Pa)
!*****
!  MAJOR RESTRICTIONS:      None
!
!  DEVELOPER:               Michael J. Brandemuehl, PhD, PE
!                          University of Colorado at Boulder
!
!  DATE:                   January 1, 1992
!
!  SUBROUTINES CALLED:      None
!  FUNCTIONS CALLED:        None
!
!  REVISION HISTORY:        None
!
!  REFERENCE:               1989 ASHRAE Handbook - Fundamentals
!*****
!  INTERNAL VARIABLES:
!  pw                      Partial water vapor pressure          (Pa)
!  small                    Small number
!*****

      DATA small/1.E-9/

!*** Test for "dry" air

```

```

      IF (W .LT. small) THEN
        DewPoint = -999
      ELSE

!*** Calculate the partial water vapor pressure as a function of
!*** humidity ratio.

        pw= Prop(Patm)*W/(.62198+W)

!*** Calculate dewpoint as saturation temperature at water vapor
!*** partial pressure

        DewPoint = SATTEMP(pw)

      ENDIF

999   RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION DEWPOINT

!*****
*

REAL FUNCTION DRYBULB (H,W)

!*****
!*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!*   FUNCTION:                DRYBULB
!*
!*   LANGUAGE:                FORTRAN 90
!*
!*   PURPOSE:                Calculate the dry bulb temperature of
!*                          moist air from enthalpy and humidity.
!*****
!*   INPUT VARIABLES:
!*   H                      Enthalpy                      (J/kg)
!*   W                      Humidity ratio                  (-)
!*
!*   OUTPUT VARIABLES:
!*   Drybulb                Dry bulb temperature          (C)
!*
!*   PROPERTIES:
!*   CpAir                  Specific heat of air           (J/kg C)
!*   CpVap                  Specific heat of water vapor   (J/kg C)
!*   Hfg                    Reference heat of vaporization of water (J/kg)
!*****
!   MAJOR RESTRICTIONS:    Uses perfect gas relationships
!                          Fit for enthalpy of saturated water vapor
!
!   DEVELOPER:             Shauna Gabel
!                          Michael J. Brandemuehl, PhD, PE
!                          University of Colorado at Boulder

```

```

!
!   DATE:                      January 1, 1992
!
!   SUBROUTINES CALLED:       None
!   FUNCTIONS CALLED:         None
!
!   REVISION HISTORY:         None
!
!   REFERENCE:                1989 ASHRAE Handbook - Fundamentals
! *****
!
!*** Calculate the dry bulb temperature as a function of enthalpy and
!*** humidity ratio.
!*** hDryAir = Prop(CpAir)*TDB
!*** hSatVap = Prop(Hfg) + Prop(CpVap)*TDB
!*** Enthalpy = hDryAir + W*hSatVap
!
!       Drybulb = (H-Prop(Hfg)*W)/(Prop(CpAir)+Prop(CpVap)*W)
!
!       RETURN
! *****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
! *****
END FUNCTION DRYBULB
! *****
*

REAL FUNCTION ENTHALPY (TDB,W)
! *****
!*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
! *****
!*   FUNCTION:                      ENTHALPY
!*
!*   LANGUAGE:                      FORTRAN 90
!*
!*   PURPOSE:                      Calculate the enthalpy of moist air.
! *****
!*   INPUT VARIABLES:
!*   TDB                          Dry bulb temperature                      (C)
!*   W                           Humidity ratio                          (-)
!*
!*   OUTPUT VARIABLES:
!*   Enthalpy                    Enthalpy of moist air                      (J/kg)
!*
!*   PROPERTIES:
!*   CpAir                      Specific heat of air                      (J/kg C)
!*   CpVap                      Specific heat of water vapor              (J/kg C)
!*   Hfg                       Reference heat of vaporization of water    (J/kg)
! *****
!   MAJOR RESTRICTIONS          Uses perfect gas relationships
!                               Fit for enthalpy of saturated water vapor
!

```



```

!      DEVELOPER:          Shauna Gabel
!                          Michael J. Brandemuehl, PhD, PE
!                          University of Colorado at Boulder
!
!      DATE:               January 1, 1992
!
!      SUBROUTINES CALLED:  None
!      FUNCTIONS CALLED:    None
!
!      REVISION HISTORY:    None
!
!      REFERENCE:           1989 ASHRAE Handbook - Fundamentals
!*****
!
!
!!*** Calculate the enthalpy as a function of dry bulb temperature and
!!*** humidity ratio.

      hDryAir = Prop(CpAir)*TDB
      hSatVap = Prop(Hfg) + Prop(CpVap)*TDB
      Enthalpy = hDryAir + W*hSatVap

      RETURN
!*****
!      EVOLUTIONARY HISTORY :
!      This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!      Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION ENTHALPY

!*****
*

REAL FUNCTION ENTHSAT (TDB)

!C*****
!C*      Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*      FUNCTION:          ENTHSAT
!C*
!C*      LANGUAGE:          FORTRAN 77
!C*
!C*      PURPOSE:           Calculate the enthalpy at saturation
!C*                        for given dry bulb temperature
!C*****
!C*      INPUT VARIABLES
!C*      TDB                Dry bulb temperature                      (C)
!C*
!C*      OUTPUT VARIABLES
!C*      EnthSat            Enthalpy at saturation                    (J/kg)
!C*
!C*      PROPERTIES
!C*      Patm              Atmospheric pressure                      (Pa)
!C*****
!C      MAJOR RESTRICTIONS:  None
!C
!C      DEVELOPER:          Shauna Gabel

```

```

!C                                     Michael J. Brandemuehl, PhD, PE
!C                                     University of Colorado at Boulder
!C
!C      DATE:                          January 1, 1992
!C
!C      SUBROUTINES CALLED:             None
!C      FUNCTIONS CALLED:               SATPRESS
!C                                     HUMRATIO
!C                                     ENTHALPY
!C
!C      REVISION HISTORY:               None
!C
!C      REFERENCE:                      1989 ASHRAE Handbook - Fundamentals
!C*****
!C      INTERNAL VARIABLES:
!C      psat          Saturated water vapor pressure          (Pa)
!C      w             Humidity ratio                          (-)
!C*****
!
!
!!*** Calculate the saturation pressure at the given temperature.

      psat = SATPRESS (TDB)

!!*** Calculate the humidity ratio from the saturation pressure

      w = HUMRATIO (Prop(Patm),psat)

!!*** Calculate the enthalpy as a function of dry bulb temperature
!!*** and humidity ratio.

      ENTHSAT = ENTHALPY (TDB,w)

      RETURN
!*****
!      EVOLUTIONARY HISTORY :
!      This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!      Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION ENTHSAT

!*****
*

REAL FUNCTION HUMRATIO (Patm,Pw)

!C*****
!C*      Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*      FUNCTION:                HUMRATIO
!C*
!C*      LANGUAGE:                FORTRAN 90
!C*
!C*      PURPOSE:                 Calculate the humidity ratio from water
!C*                             vapor pressure and atmospheric pressure
!C*****
!C*      INPUT VARIABLES

```

```

!C*      Patm          Atmospheric pressure              (Pa)
!C*      Pw           Partial water vapor pressure      (Pa)
!C*
!C*      OUTPUT VARIABLES
!C*      HumRatio      Humidity ratio                    (-)
!C*****
!C      MAJOR RESRICTIONS:      None
!C
!C      DEVELOPER:              Shauna Gabel
!C                              Michael J. Brandemuehl, PhD, PE
!C                              University of Colorado at Boulder
!C
!C      DATE:                  January 1, 1992
!C
!C      SUBROUTINES CALLED:      None
!C      FUNCTIONS CALLED:        None
!C
!C      REVISION HISTORY:        None
!C
!C      REFERENCE:              1989 ASHRAE Handbook - Fundamentals
!C*****

!!*** Calculate the humidity ratio.

      HumRatio = 0.62198*Pw/(Patm-Pw)

      RETURN
!*****
!      EVOLUTIONARY HISTORY :
!      This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!      Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION HUMRATIO

!*****
*

REAL FUNCTION HUMTH (TDB,H)

!C*****
!C*      Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*      FUNCTION:              HUMTH
!C*
!C*      LANGUAGE:              FORTRAN 90
!C*
!C*      PURPOSE:              Calculate the humidity ratio of moist air
!C*                          from dry bulb temperature and enthalpy.
!C*****
!C*      INPUT VARIABLES:
!C*      H                      Enthalpy                  (J/kg)
!C*      TDB                    Dry bulb temperature        (C)
!C*
!C*      OUTPUT VARIABLES:
!C*      HumTH                  Humidity ratio              (-)
!C*
!C*      PROPERTIES:

```

```

!C*      CpAir          Specific heat of air                      (J/kg C)
!C*      CpVap          Specific heat of water vapor              (J/kg C)
!C*      Hfg            Reference heat of vaporization of water    (J/kg)
!C*****
!C      MAJOR RESTRICTIONS:    Uses perfect gas relationships
!C                             Fit for enthalpy of saturated water vapor
!C
!C      DEVELOPER:            Shauna Gabel
!C                             Michael J. Brandemuehl, PhD, PE
!C                             University of Colorado at Boulder
!C
!C      DATE:                 January 1, 1992
!C
!C      SUBROUTINES CALLED:    None
!C      FUNCTIONS CALLED:      None
!C
!C      REVISION HISTORY:      None
!C
!C      REFERENCE:             1989 ASHRAE Handbook - Fundamentals
!C*****
!
!
!!*** Calculate humidity ratio from dry bulb temperature and enthalpy
!!*** hDryAir = Prop(CpAir)*TDB
!!*** hSatVap = Prop(Hfg) + Prop(CpVap)*TDB
!!*** Enthalpy = hDryAir + W*hSatVap

      HumTH = (H-Prop(CpAir)*TDB)/(Prop(Hfg)+Prop(CpVap)*TDB)

      RETURN
!*****
!      EVOLUTIONARY HISTORY :
!      This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!      Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION HUMTH

!*****
*

REAL FUNCTION RELHUM (Patm,Psat,HumRatio)

!C*****
!C*      Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*      FUNCTION:              RELHUM
!C*
!C*      LANGUAGE:              FORTRAN 90
!C*
!C*      PURPOSE:                Calculate the relative humidity from
!C*                             saturation and atmospheric pressures
!C*****
!C*      INPUT VARIABLES
!C*      Patm          Atmospheric pressure                      (Pa)
!C*      Psat          Saturation pressure                      (Pa)
!C*      HumRatio      Humidity ratio                          (-)
!C*

```

```

!C*      OUTPUT VARIABLES
!C*      RelHum      Relative humidity      (-)
!C*****
!C      MAJOR RESTRICTIONS:      None
!C
!C      DEVELOPER:      Shauna Gabel
!C                      Michael J. Brandemuehl, PhD, PE
!C                      University of Colorado at Boulder
!C
!C      DATE:      January 1, 1992
!C
!C      SUBROUTINES CALLED:      None
!C      FUNCTIONS CALLED:      None
!C
!C      REVISION HISTORY:      None
!C
!C      REFERENCE:      1989 ASHRAE Handbook - Fundamentals
!C*****
!C      INTERNAL VARIABLES:
!C      pw      Partial water vapor pressure      (Pa)
!C*****

!!*** Calculate the partial water vapor pressure as a function of
!!*** humidity ratio.

      pw = Patm*HumRatio/(.62198+HumRatio)

!!*** Calculate the relative humidity as a function of partial water
!!*** vapor pressure and water vapor pressure at saturation.

      RelHum = pw/Psat

      RETURN
!*****
!      EVOLUTIONARY HISTORY :
!      This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!      Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION RELHUM

!*****
*

REAL FUNCTION RHODRY (TDB,W)

!C*****
!C*      Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*      FUNCTION:      RHODRY
!C*
!C*      LANGUAGE:      FORTRAN 90
!C*
!C*      PURPOSE:      Calculate dry air density.
!C*****
!C*      INPUT VARIABLES
!C*      TDB      Dry bulb temperature      (C)
!C*      W      Humidity ratio      (-)

```

```

!C*
!C*   OUTPUT VARIABLES
!C*   RhoDry      Density of dry air                      (kg/m3)
!C*
!C*   PROPERTIES
!C*   Patm        Atmospheric pressure                    (Pa)
!C*   RAir        Gas constant for air                    (J/kg C)
!C*   TAbsAdd     Additive constant to convert user T to absolute T
!C*****
!C   MAJOR RESTRICTIONS:      Perfect gas relationships
!C
!C   DEVELOPER:              Shauna Gabel
!C                           Michael J. Brandemuehl, PhD, PE
!C                           University of Colorado at Boulder
!C
!C   DATE:                  January 1, 1992
!C
!C   SUBROUTINES CALLED:     None
!C   FUNCTIONS CALLED:       None
!C
!C   REVISION HISTORY:       None
!C
!C   REFERENCE:              1989 ASHRAE Handbook - Fundamentals
!C*****
!C   INTERNAL VARIABLES:
!C   pAir          Partial pressure of dry air              (Pa)
!C*****
!
!
!!*** Calculate the dry air density from perfect gas laws.

      pAir = 0.62198*Prop(Patm)/(0.62198+W)
      RhoDry = pAir/Prop(RAair)/(TDB+Prop(TAbsAdd))

      RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION RHODRY

!*****
*

REAL FUNCTION RHOMOIST (RhoDry,W)

!C*****
!C*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*   FUNCTION:              RHOMOIST
!C*
!C*   LANGUAGE:              FORTRAN 90
!C*
!C*   PURPOSE:              Calculate moist air density from dry air
!C*                          density and humidity ratio
!C*****

```

```

!C*    INPUT VARIABLES:
!C*    RhoDry          Dry air density          (kg/m3)
!C*    W              Humidity ratio           (-)
!C*
!C*    OUTPUT VARIABLES:
!C*    RhoMoist        Density of dry air       (kg/m3)
!C*****
!C    MAJOR RESTRICTIONS:    None
!C
!C    DEVELOPER:            Shauna Gabel
!C                          Michael J. Brandemuehl, PhD, PE
!C                          University of Colorado at Boulder
!C
!C    DATE:                January 1, 1992
!C
!C    SUBROUTINES CALLED:    None
!C    FUNCTIONS CALLED:      None
!C
!C    REVISION HISTORY:      None
!C
!C    REFERENCE:            1989 ASHRAE Handbook - Fundamentals
!C*****

!!*** Calculate the moist air density

      RhoMoist = RhoDry*(1.+W)

      RETURN
!*****
!    EVOLUTIONARY HISTORY :
!    This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!    Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION RHOMOIST

!*****
*

REAL FUNCTION SATPRESS (T)

!C*****
!C*    Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*    SUBROUTINE:            SATPRESS
!C*
!C*    LANGUAGE:              FORTRAN 90
!C*
!C*    PURPOSE:               Calculate saturation pressure of water
!C*                          vapor as a function of temperature
!C*****
!C*    INPUT VARIABLES
!C*    T                      Temperature          (C)
!C*
!C*    OUTPUT VARIABLES
!C*    SatPress               Saturation pressure   (Pa)
!C*
!C*    PROPERTIES

```

```

!C*      TKelMult      Multiplying factor to convert user T to Kelvin
!C*      TAbsAdd       Additive factor to convert user T to absolute T
!C*                      tKel = Prop(TKelMult) * (T + Prop(TAbsAdd))
!C*      PaMult        Multiplying factor to convert user P to Pascals
!C*      PAbsAdd       Additive factor to convert user P to absolute P
!C*                      Pa = Prop(PaMult) * (P + Prop(PAbsAdd))
!C*****
!C      MAJOR RESTRICTIONS:      173.16 K <= Temp <= 473.15 K
!C
!C      DEVELOPER:              Shauna Gabel
!C                              Michael J. Brandemuehl, PhD, PE
!C                              University of Colorado at Boulder
!C
!C      DATE:                   January 1, 1992
!C
!C      SUBROUTINES CALLED:      None
!C      FUNCTIONS CALLED:        None
!C
!C      REVISION HISTORY:        None
!C
!C      REFERENCE:               1989 ASHRAE Handbook - Fundamentals
!C
!C                              Hyland, R.W., and A. Wexler. 1983.
!C                              Formulations for the thermodynamic
!C                              properties of the saturated phases of H2O
!C                              from 173.15 K to 473.15 K. ASHRAE
!C                              Transactions, Vol. 89, No. 2A, pp. 500-519
!C*****
!C      INTERNAL VARIABLES:
!C      tKel              Temperature in Kelvin                      (K)
!C      pascals           Saturation pressure                       (Pa)
!C*****

DATA C1/-5674.5359/,C2/6.3925247/,C3/-0.9677843E-2/
DATA C4/0.62215701E-6/,C5/0.20747825E-8/,C6/-0.9484024E-12/
DATA C7/4.1635019/,C8/-5800.2206/,C9/1.3914993/,C10/-0.048640239/
DATA C11/0.41764768E-4/,C12/-0.14452093E-7/,C13/6.5459673/

!!*** Convert temperature from user units to Kelvin.
      tKel = Prop(TKelMult)*(T+Prop(TAbsAdd))

!!*** If below freezing, calculate saturation pressure over ice.

      IF (tKel .LT. 273.15) THEN

          pascals =
EXP(C1/tKel+C2+tKel*(C3+tKel*(C4+tKel*(C5+C6*tKel)))+C7*ALOG(tKel))

!!*** If above freezing, calculate saturation pressure over liquid water.

      ELSE IF (tKel .GE. 273.15) THEN

          pascals = EXP(C8/tKel+C9+tKel*(C10+tKel*(C11+tKel*C12))+C13*ALOG(tKel))

      ENDIF

```



```

!!*** Convert pressure from Pascals to user units

      SatPress = pascals/Prop(PaMult) - Prop(PAbsAdd)

      RETURN
!*****
!  EVOLUTIONARY HISTORY :
!  This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!  Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION SATPRESS

!*****
*

REAL FUNCTION SATTEMP (P)

!C*****
!C*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*   FUNCTION:                SATTEMP
!C*
!C*   LANGUAGE:                FORTRAN 90
!C*
!C*   PURPOSE:                 Calculate the saturation (boiling)
!C*                           temperature of water given pressure
!C*****
!C*   INPUT VARIABLES
!C*   P                        Pressure                        (Pa)
!C*
!C*   OUTPUT VARIABLES
!C*   SatTemp                 Saturation temperature of water vapor      (C)
!C*****
!C   MAJOR RESTRICTIONS:      None
!C
!C   DEVELOPER:               Shauna Gabel
!C                           Michael J. Brandemuehl, PhD, PE
!C                           University of Colorado at Boulder
!C
!C   DATE:                   January 1, 1992
!C
!C   SUBROUTINES CALLED:      None
!C   FUNCTIONS CALLED:        SATPRESS
!C                           XITERATE
!C
!C   REVISION HISTORY:        None
!C
!C   REFERENCE:               1989 ASHRAE Handbook - Fundamentals
!C*****
!C   INTERNAL VARIABLES:
!C   tSat                     Water temperature guess                (C)
!C   pSat                     Pressure corresponding to temp. guess    (Pa)
!C   error                    Deviation of dependent variable in iteration
!C   iter                     Iteration counter
!C   icvg                     Iteration convergence flag
!C   F1,F2                    Previous values of dependent variable in XITERATE
!C   X1,X2                    Previous values of independent variable in XITERATE

```

```

!C*****
!
      DATA itmax/200/

!!*** Use an iterative process to determine the saturation temperature
!!*** at a given pressure using a correlation of saturated water vapor
!!*** pressure as a function of temperature

!!*** Initial guess of boiling temperature

      tSat = 100.

!!*** Iterate to find the saturation temperature
!!*** of water given the total pressure

!!*** Set iteration loop parameters

      DO 100 iter = 1,itmax

!!*** Calculate saturation pressure for estimated boiling temperature

      pSat = SATPRESS(tSat)

!!*** Compare with specified pressure and update estimate of temperature

      error = P - pSat
      tSat = XITERATE (tSat,error,X1,F1,X2,F2,iter,icvg)

!!*** If converged leave loop iteration

      IF (icvg .EQ. 1) GO TO 110

!!*** Water temperature not converged, repeat calculations with new
!!*** estimate of water temperature

100    CONTINUE

!!*** Saturation temperature has not converged after maximum specified
!!*** iterations. Print error message, set return error flag, and RETURN

      WRITE(*,1001) itmax
1001  FORMAT(/1X,'*** ERROR IN FUNCTION SatTemp ***'/
           1X,'      Saturation temperature has not '
           'converged after ',I4,' iterations'/)

110    SatTemp = tSat

      RETURN
!*****
!  EVOLUTIONARY HISTORY :
!  This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!  Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION SATTEMP

!*****
*
```

```

REAL FUNCTION TAIRSAT (HSat)

!C*****
!C*   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*   FUNCTION:                ENTHSAT
!C*
!C*   LANGUAGE:                FORTRAN 90
!C*
!C*   PURPOSE:                 Calculate the dry bulb temperature given
!C*                           enthalpy at saturation.
!C*****
!C*   INPUT VARIABLES:
!C*   HSat                     Enthalpy at saturation                      (J/kg)
!C*
!C*   OUTPUT VARIABLES:
!C*   TAirsat                  Dry bulb temperature                      (C)
!C*****
!C   MAJOR RESTRICTIONS:      None
!C
!C   DEVELOPER:                Michael J. Brandemuehl, PhD, PE
!C                           University of Colorado at Boulder
!C
!C   DATE:                     January 1, 1992
!C
!C   SUBROUTINES CALLED:      None
!C   FUNCTIONS CALLED:        ENTHSAT
!C
!C   REVISION HISTORY:        None
!C
!C   REFERENCE:                1989 ASHRAE Handbook - Fundamentals
!C*****
!C   INTERNAL VARIABLES:
!C   error                     Deviation of dependent variable in iteration
!C   iter                      Iteration counter
!C   icvg                      Iteration convergence flag
!C   F1,F2                     Previous values of dependent variable in XITERATE
!C   X1,X2                     Previous values of independent variable in XITERATE
!C*****

      DATA itmax/20/,tSat/50./

!!*** Estimate saturation temperature if reasonable value not available

      IF(tSat .LT. -200. .OR. tSat .GT. 1000.) tSat = 50.

!!*** Calculate saturation temperature by iteration using function to
!!*** calculate saturation enthalpy from temperature

      DO 100 iter=1,itmax

          error = HSat - ENTHSAT(tSat)
          tSat = XITERATE(tSat,error,X1,F1,X2,F2,iter,icvg)

!!*** If converged, leave iteration loop.

```

```

        IF (icvg .EQ. 1) GO TO 110

!!*** Temperature not converged, repeat calculation with new
!!*** estimate of temperature.

100    CONTINUE

!!*** Temperature has not converged after maximum specified
!!*** iterations. Print error message and RETURN

        WRITE(*,1001) itmax
1001   FORMAT(/1X,'*** ERROR IN FUNCTION TAIRSAT ***'/
           1X,'      Temperature has not '
           'converged after ',I2,' iterations'/)

110    CONTINUE

        TAIRSat = tSat

        RETURN
!*****
!    EVOLUTIONARY HISTORY :
!    This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!    Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION TAIRSAT

!*****
*

REAL FUNCTION WETBULB (TDB,W)

!C*****
!C*    Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!C*****
!C*    FUNCTION:                WETBULB
!C*
!C*    LANGUAGE:                FORTRAN 90
!C*
!C*    PURPOSE:                Calculate wet bulb temperature from dry
!C*                          bulb temperature and humidity ratio
!C*****
!C*    INPUT VARIABLES
!C*    TDB                    Dry bulb temperature                (C)
!C*    W                      Humidity ratio of air              (-)
!C*
!C*    OUTPUT VARIABLES
!C*    WetBulb                Wet bulb temperature              (C)
!C*
!C*    PROPERTIES:
!C*    Patm                    Atmospheric pressure              (Pa)
!C*    Hfg                     Latent heat of vaporization of water (J/kg)
!C*    CpAir                   Specific heat of air              (J/kg C)
!C*    CpVap                   Specific heat of water vapor       (J/kg C)
!C*    CpWat                   Specific heat of water             (J/kg C)
!C*****
!C    MAJOR RESTRICTIONS:      None

```

```

!C
!C   DEVELOPER:           Shauna Gabel
!C                        Michael J. Brandemuehl, PhD, PE
!C                        University of Colorado at Boulder
!C
!C   DATE:                January 1, 1992
!C
!C   SUBROUTINES CALLED:  None
!C   FUNCTIONS CALLED:    SATPRESS
!C                        HUMRATIO
!C                        SATTEMP
!C                        XITERATE
!C
!C   REVISION HISTORY:    None
!C
!C   REFERENCE:           1989 ASHRAE Handbook - Fundamentals
!C*****
!C   INTERNAL VARIABLES:
!C   tBoil                Boiling temperature of water at given pressure    (C)
!C   psatStar             Saturation pressure at wet bulb temperature        (C)
!C   wStar                Humidity ratio as a function of PsatStar           (-)
!C   newW                 Humidity ratio calculated with wet bulb guess      (-)
!C   error                Deviation of dependent variable in iteration
!C   iter                 Iteration counter
!C   icvg                 Iteration convergence flag
!C   F1,F2                Previous values of dependent variable in XITERATE
!C   X1,X2                Previous values of independent variable in XITERATE
!C*****
!
      REAL newW
      DATA itmax/20/

!*** Initial temperature guess

      tBoil = SATTEMP (Prop(Patm))
      WetBulb = MAX( MIN(WetBulb,TDB,(tBoil-0.1)), 0.)

!*** Begin iteration loop

      DO 100 iter = 1,itmax

          IF (WetBulb .GE. (tBoil-0.09) ) WETBULB = tBoil-0.1

!*** Determine the saturation pressure for wet bulb temperature

          psatStar = SATPRESS (WetBulb)

!*** Determine humidity ratio for given saturation pressure

          wStar = HUMRATIO (Prop(Patm),psatStar)

!*** Calculate new humidity ratio and determine difference from known
!*** humidity ratio

          newW = ((Prop(Hfg)-(Prop(CpWat)-Prop(CpVap))*WetBulb)*wStar-
                  Prop(CpAir)*(TDB-WetBulb))/(Prop(Hfg)+Prop(CpVap)*TDB

```

```

      -Prop(CpWat)*WetBulb)

!*** Check error, if not satisfied, calculate new guess and iterate

      error = W-newW
      WetBulb = XITERATE(WetBulb,error,X1,F1,X2,F2,iter,icvg)

!*** If converged, leave iteration loop.

      IF (icvg .EQ. 1) GO TO 900

!*** Wet bulb temperature not converged, repeat calculation with new
!*** estimate of wet bulb temperature.

100   CONTINUE

!*** Wet bulb temperature has not converged after maximum specified
!*** iterations. Print error message, set return error flag, and RETURN

      WRITE(*,1009) itmax
1009  FORMAT(/1X,'*** ERROR IN FUNCTION WetBulb ***'/
           1X,'      Wet bulb temperature has not '
           'converged after ',I2,' iterations'//)

900   IF (WetBulb .GT. TDB) WetBulb = TDB

999   RETURN

!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****
END FUNCTION WETBULB

!*****
*

REAL FUNCTION XITERATE (X0,F0,X1,F1,X2,F2,ICount,ICvg)

!*****
!   Copyright ASHRAE.  Toolkit for HVAC System Energy Calculations
!*****
!
!   SUBROUTINE:           XITERATE
!
!   LANGUAGE:            FORTRAN 90
!
!   PURPOSE:             Iterately solves for the value of X which
!                       satisfies F(X)=0. Given Xi,F(Xi) pairs,
!                       the subroutine tests for convergence and
!                       provides a new guess for the value of the
!                       independent variable X.
!*****
!   INPUT VARIABLES
!   F0                   Current value of the function F(X)
!   X0                   Current value of X

```

```

!      F1,F2          Two previous values of F(Xi)
!      X1,X2          Two previous values of X
!
!      NOTE:          F1,X1,F2,X2 MUST BE STORED AND SAVED IN CALLING
!                      ROUTINE.  THEY NEED NO INITIALIZATION
!
!      ICount         Number of iterations
!
!      OUTPUT VARIABLES
!      XIterate        New estimate of X for F(X)=0
!      ICvg            Convergence flag  ICvg = 0:  Not converged
!                      ICvg = 1:  Converged
!*****
!      DEVELOPER:      Michael J. Brandemuehl, PhD, PE
!                      University of Colorado at Boulder
!
!      DATE:           January 1, 1992
!
!      INCLUDE FILES:  None
!      SUBROUTINES CALLED:  None
!      FUNCTIONS CALLED:  None
!
!      REFERENCE:      None
!*****
!      INTERNAL VARIABLES
!      small           Small number used in place of zero
!      mode            Number of points used in fit
!                      mode = 1:  Use XPerburb to get new X
!                      mode = 2:  Linear equation to get new X
!                      mode > 2:  Quadratic equation to get new X
!      coef(i)         Coefficients for quadratic fit
!                      F(X) = coef(1) + coef(2)*X + coef(3)*X*X
!      check           Term under radical in quadratic solution
!      FiQ,XiQ         Double precision values of Fi,Xi
!      slope           Slope for linear fit
!      tolRel          Relative error tolerance
!      xPerturb        Perturbation applied to X to initialize iteration
!*****

      DOUBLE PRECISION coef(3),check,F0Q,F1Q,F2Q,X0Q,X1Q,X2Q
      DATA tolRel/1.E-5/,xPerturb/0.1/,small/1.E-9/

!*** Check for convergence by comparing change in X

      IF ((ABS(X0-X1) .LT. tolRel*MAX(ABS(X0),small) .AND.
          ICount .NE. 1) .OR. F0 .EQ. 0.) THEN
          XIterate = X0
          ICvg=1
          RETURN
      ENDIF

!*** Not converged.
!*** If after the second iteration there are enough previous points to
!      fit a quadratic for the new X.  If the quadratic fit is not
!      applicable, mode will be set to 1 or 2 and a new X will be
!      determined by incrementing X from xPerturb or from a linear fit.

```

```

        ICvg=0
        mode=ICount
10    IF (mode .EQ. 1) THEN

!*** New guess is specified by xPerturb

        IF (ABS(X0) .GT. small) THEN
            XIterate = X0*(1.+xPerturb)
        ELSE
            XIterate = xPerturb
        ENDIF

        ELSEIF (mode .EQ. 2) THEN

!*** New guess calculated from LINEAR FIT of most recent two points

        SLOPE=(F1-F0)/(X1-X0)
        IF(slope.EQ.0) THEN
            mode=1
            GO TO 10
        ENDIF
        XIterate=X0-F0/SLOPE
    ELSE

!*** New guess calculated from QUADRATIC FIT

!*** If two Xi are equal, set mode for linear fit and return to top

        IF (X0 .EQ. X1) THEN
            X1=X2
            F1=F2
            mode=2
            GO TO 10
        ELSEIF (X0 .EQ. X2) THEN
            mode=2
            GO TO 10
        ENDIF

!*** Determine quadratic coefficients from the three data points
!*** using double precision.

        F2Q=F2
        F1Q=F1
        F0Q=F0
        X2Q=X2
        X1Q=X1
        X0Q=X0
        coef(3)=(F2Q-F0Q)/(X2Q-X0Q)-(F1Q-F0Q)/(X1Q-X0Q)/(X2Q-X1Q)
        coef(2)=(F1Q-F0Q)/(X1Q-X0Q)-(X1Q+X0Q)*coef(3)
        coef(1)=F0-(coef(2)+coef(3)*X0Q)*X0Q

!*** If points are colinear, set mode for linear fit and return to top

        IF (ABS(coef(3)) .LT. 1.D-10) THEN
            mode=2
            GO TO 10
        ENDIF

```



```
!*** Check for precision.  If the coefficients do not accurately
!*** predict the given data points due to round-off errors, set
!*** mode for a linear fit and return to top.

      IF (ABS((coef(1)+(coef(2)+coef(3)*X1Q)*X1Q-F1Q)/F1Q) .GT. 1.D-4) THEN
        mode=2
        GO TO 10
      ENDIF

!*** Check for imaginary roots.  If no real roots, set mode to
!*** estimate new X by simply incrementing by xPerturb

      check=coef(2)**2-4*coef(1)*coef(3)
      IF (check .LT. 0) THEN

!*** Imaginary roots -- go back to linear fit

        mode=2
        GO TO 10

      ELSEIF (check .GT. 0) THEN

!*** Real unequal roots -- determine root nearest to most recent guess

        XIterate=(-coef(2)+SQRT(check))/coef(3)/2
        xOther=-XIterate-coef(2)/coef(3)
        IF (ABS(XIterate-X0) .GT. ABS(xOther-X0)) XIterate=xOther
      ELSE

!*** Real Equal Roots -- one solution

        XIterate=-coef(2)/coef(3)/2
      ENDIF

    ENDIF

!*** Set previous variable values for the next iteration

    IF (mode .LT. 3) THEN

!*** No valid previous points to eliminate.

      X2=X1
      F2=F1
      X1=X0
      F1=F0

    ELSE

!*** Eliminate one previous point based on sign and magnitude of F(X)
!*** Keep the current point and eliminate one of the previous ones.

      IF (F1*F0 .GT. 0 .AND. F2*F0 .GT. 0) THEN

!*** All previous points of same sign.  Eliminate one with biggest F(X)
```

```

        IF (ABS(F2) .GT. ABS(F1)) THEN
            X2=X1
            F2=F1
        ENDIF
    ELSE

!*** Points of different sign.
!*** Eliminate the previous one with the same sign as current F(X).

        IF (F2*F0 .GT. 0) THEN
            X2=X1
            F2=F1
        ENDIF
    ENDIF
    X1=X0
    F1=F0
    ENDIF
    RETURN
!*****
!   EVOLUTIONARY HISTORY :
!   This subroutine is adopted from ASHRAE Toolkit for Secondary HVAC System
!   Energy Calculations which was originally written in FORTRAN 77.
!*****

END FUNCTION Xiterate

!   Notes:
!   Subroutines in this module are adopted from ASHRAE Toolkit for Secondary
!   HVAC System Energy Calculations which was originally written in FORTRAN
77,
!   and put into loads Toolkit FORTRAN 90 module by Leon Dong, 1998
!*****
*
!   Toolkit for Building Load Calculations in FORTRAN 90
!*****
*
!   OWNERSHIP:
!   Ownership of the module and contained subroutines in this toolkit
!   remain with the copyright owner. ASHRAE is granted a license to
!   distribute the module and contained subroutines in the toolkit
!   resulting from RP 987.
!*****
*

END MODULE PsychrometricsMod

```

Source Code: Module IntHeatGain

```

MODULE IntHeatGain

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical      solutions to the internal heat gain test case.
!
! CREATED:      12.22.99      Dongyi Xiao
!
! MODIFIED:
!
USE ConductUtil

PRIVATE

! make public main subroutine
PUBLIC :: IntHeatGain_Rad

CONTAINS

!-----
-

SUBROUTINE IntHeatGain_Rad(FileName,Thickness,H_in, Cond,den, cp,T_inside,&
                           Heatload,Percentage_rad,Sur_area,Time,Times,&
                           time_number,ErrorMessage,ErrorFlag)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "IntHeatGain_Rad" :: IntHeatGain_Rad

! PURPOSE:      Produces analytical response for the case of internal heat
!               gains based on transient conduction heat transfer.
!
! CREATED:

```

```

! 12.22.99      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
!
!   FileName.csv           -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   FileName               -   Root File Name for data files
!   Thickness               -   Thickness of the wall
!   H_in                   -   Internal convection heat transfer coefficient
!   Cond                   -   Conductivity of the wall
!   den                    -   Density of the wall
!   cp                     -   Specific heat capacity of the wall
!   T_inside               -   Internal air temperature
!   Heatload               -   Internal heat gain of the zone
!   Percentage_rad         -   The radiative percentage of the heat load
!   Sur_area               -   Surface area of the wall
!   Time                   -   The time period in which the step change
occurs
!   Times                  -   The array of times at which the surface
temperature
!                           will be solved
!   time_number            -   The number of times at which the surface
!                           temperature will be solved
!   ErrorFlag              -   for error checking
!   ErrorMessage           -   for error reporting

IMPLICIT NONE

!parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER :: Results=7 ! unit number for analytical response
output file
INTEGER, PARAMETER :: DebugFile_Temp=8 ! unit number for Debug file of
heat balance
INTEGER, PARAMETER :: DebugFile_Roots=9 ! unit number for window shading
result file

INTEGER, PARAMETER :: N_roots=1000          ! No. of roots of aux. equation

!arguments
REAL(prec2), INTENT(in)      :: Thickness
REAL(prec2), INTENT(in)      :: H_in
REAL(prec2), INTENT(in)      :: Cond
REAL(prec2), INTENT(in)      :: den
REAL(prec2), INTENT(in)      :: cp
REAL(prec2), INTENT(in)      :: T_inside
INTEGER, INTENT(in)          :: time_number

REAL(prec2), INTENT(in)      :: Heatload

```

```

REAL(prec2), INTENT(in)      :: Percentage_rad, Sur_area
REAL(prec2), DIMENSION(time_number), INTENT(in) :: Times
REAL(prec2), INTENT(in)      :: Time

CHARACTER *(*), INTENT(in)   :: FileName
CHARACTER *(*), INTENT(out)  :: ErrorMessage
INTEGER(2), INTENT(out)      :: ErrorFlag ! two byte int for VB integer

!local variables
INTEGER                :: i,j,error
INTEGER :: n ! The number of roots will be resolved
INTEGER :: location_number ! The number of locations at which
                        !the surface temperature will be solved
REAL(prec2)                :: Acc_temp
REAL(prec2), DIMENSION(350,1) :: Temp
REAL(prec2), DIMENSION(350,1) :: Flux
REAL(prec2), DIMENSION(350,1) :: Load
REAL(prec2)                :: Peak_load

INTEGER, DIMENSION(350,1)    :: Num_terms ! The number of terms needed
for convergence

                        ! while calculating the surface
temperature
REAL(prec2)                :: Flux_total
REAL(prec2)                :: Flux_radiative
REAL(prec2), DIMENSION(350) :: Flux_convective
REAL(prec2), DIMENSION(1)   :: Locations ! The array of locations at
which the
                        !surface temperature will
be solved
REAL(prec2), DIMENSION(350,1) :: accuracy ! Array of accuracy resulted
for
                        ! the surface temperature
REAL(prec2), DIMENSION(N_roots) :: realroots ! Array of realroots solved
out
REAL(prec2)                :: Time1 ! The beginning time of the step
change
REAL(prec2)                :: Time2 ! The end time of the step change

REAL(prec2), DIMENSION(N_roots) :: Beta ! array of aux. eqn. roots
REAL(prec2), DIMENSION(3)       :: Param ! array of aux. eqn. parameters
LOGICAL                        :: Debug = .false. ! debug flag

Acc_temp=0.0001

Locations(1)=0.0
location_number=1
Time1=0.0
Time2=Time

! assign aux. eqn. parameters
Param(1) = H_in
Param(2) = Cond
Param(3) = Thickness

! find roots of auxiliary equation

```

```

CALL FindTransRoots(f1, Param, Beta, N_roots, Debug)

realroots=Beta
Flux_total=Heatload/Sur_area
Flux_radiative=Flux_total*Percentage_rad
Flux_convective=0.0
DO i = 1 , (time_number-1)/2
    Flux_convective(i)=Flux_total*(1-Percentage_rad)
END DO

! calculate the internal surface temperature
CALL Temperature(N_roots,time_number, location_number, Locations, Times,
Temp,accuracy,&
                                Num_terms, realroots,Thickness,H_in, Cond, den, cp,
Acc_temp,&
                                T_inside,Flux_radiative,Time1,Time2)

! calculate the convection heat flux from the surface and the zone load
DO i = 1 , time_number

    DO j = 1 , location_number

        Flux(i,j)=H_in*(Temp(i,j)-T_inside)
        Load(i,j)=Sur_area*(Flux(i,j)+Flux_convective(i))

    END DO

    ! find out the peak zone load in one day
    IF (i==2) THEN

        Peak_load=MAX(Load(i,1),Load(i-1,1))

    END IF

    IF (i>2)THEN

        Peak_load=MAX(Load(i,1),Peak_load)

    END IF

END DO

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results,'("# ASHRAE Analytical Test Suite - Results file")')
WRITE(Results,'("# Internal heat gain test")')
WRITE(Results,'("//Test Parameters -")')

```

```

WRITE(Results, '("Inside convection coefficient =", "", F15.4)') H_in
WRITE(Results, '("Inside air temperature =", "", F15.4)') T_inside
WRITE(Results, '("Fabric layer thickness =", "", F15.4)') Thickness
WRITE(Results, '("Surface Area =", "", F15.4)') Sur_area
WRITE(Results, '("Conductivity of the wall =", "", F15.4)') Cond
WRITE(Results, '("Density of the wall =", "", F15.4)') den
WRITE(Results, '("Heat capacity of the wall =", "", F15.4)') cp
WRITE(Results, '("Accuracy used for solving surface temperature =", "", &
    F15.10)') Acc_temp
WRITE(Results, '("Internal heat load =", "", F15.4)') Heatload
WRITE(Results, '("Percentage of radiative heat load =", "", &
    F15.4)') Percentage_rad

WRITE(Results, '("Period of time solved for =", "", F15.4)') Time

WRITE(Results, '("//"Test results -")')
WRITE(Results, '("//"Peak Load =", "", F15.4)') Peak_load

WRITE(Results, '("//"Time", "", "Internal Surface Temp.", "", "Internal
Surface Heat Flux", &
    ", "", "Zone Load")')

DO i = 1 , time_number

    DO j = 1 , location_number

        WRITE(Results, '(F15.4, " ", F15.4, " ", F15.4, " ", F15.4)')
Times(i), Temp(i,j), &
        Flux(i,j), Load(i,j)

    END DO

END DO

CLOSE(Results)

!open a debug file for checking the roots
! OPEN (DebugFile_Roots, file="IntHeatGain_Roots.txt", status='unknown',
IOSTAT=error)
! IF(error/=0) THEN
!     ErrorMessage="Debug output file could not be opened"
!     ErrorFlag = 1
!     RETURN
! END IF

! write header lines
! WRITE(DebugFile_Roots, '("# ASHRAE Analytical Test Suite - &
!     DebugFile file for solving roots)')
! WRITE(DebugFile_Roots, '("# Internal heat gain test)')
! WRITE(DebugFile_Roots, '("//"Realroots)')

! DO i=1, N_roots

!     WRITE(DebugFile_Roots, '(F20.6)') realroots(i)

! END DO

```

```

!      CLOSE (DebugFile_Roots)

!open a debug file for checking the temperature results
!      OPEN (DebugFile_Temp, file="IntHeatGain_Temp.txt", status='unknown',
IOSTAT=error)
!      IF(error/=0) THEN
!          ErrorMessage="Debug output file could not be opened"
!          ErrorFlag = 1
!          RETURN
!      END IF

! write header lines
!      WRITE(DebugFile_Temp, '( "# ASHRAE Analytical Test Suite - DebugFile file
for &
!                               internal surface temperature" )' )
!      WRITE(DebugFile_Temp, '( "# Internal heat gain test" )' )
!
WRITE(DebugFile_Temp, '( //6X, "Times", 10X, "locations", 10X, "Ts", 10X, "Accuracy", &
!                               4X, "Num_terms" )' )

!      DO i = 1 , time_number

!          DO j = 1 , location_number

!              WRITE(DebugFile_Temp, '( 4F15.4, I10 )' ) Times(i), Locations(j),
Temp(i,j), &
!                               accuracy(i,j), Num_terms(i,j)

!          END DO

!      END DO

!      CLOSE (DebugFile_Temp)

END SUBROUTINE IntHeatGain_Rad

!-----
-

FUNCTION f1(X, Param)

! PURPOSE:      Defines transcendental function used in analytical
solution
!               to adiabatic wall test case with step in air temperature
as
!               driving function
!
! CREATED:
!   5.11.99      S.J.Rees
!
! MODIFIED:
!

```



```

IMPLICIT NONE

INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=12)

REAL(prec) :: f1

REAL(prec), INTENT(IN) :: X
REAL(prec), DIMENSION(3), INTENT(IN) :: Param
REAL(prec) :: h_c, K, L

h_c = Param(1)
K = Param(2)
L = Param(3)

f1 = X*TAN(X*L) - h_c/K

END FUNCTION f1

!-----
-----

SUBROUTINE Temperature(n,time_number, location_number, Locations, Times,
Temp,&
                                accuracy, Num_terms,realroots,Thickness,H_in, Cond,
den, &
                                cp, Acc_temp, T_inside,Flux_radiative,Time1,Time2)

    ! PURPOSE:      caculate the values of tempreture, accuracy and the number
of terms in      the series used to obtain the required accuracy at each
!                  time/location combinations
!
!   ! CREATED:      12.17.99   Dongyi Xiao
!
!   ! MODIFIED:
!
!   ! INPUT FILES:
!     n/a
!
!   ! OUTPUT FILES:
!     n/a
!
!   ! SUBROUTINE ARGUMENTS:
!     n              - The number of roots will be resolved
!     time_number    - The number of times at which the surface
!                      temperature will be solved
!     location_number - The number of locations at which the surface
!                      temperature will be solved
!     Locations      - The array of locations at which the surface
!                      temperature will be solved
!     Times          - The array of times at which the surface
!                      temperature will be solved
!     Temp           - Array of internal surface temperature

```

```

! accuracy          - Array of accuracy resulted for the surface
temperature
! Num_terms         - The number of terms in the series used to
obtain
!                   the required accuracy
! realroots          - Array of realroots solved out
! Thickness          - Thickness of the wall
! H_in              - Internal convection heat transfer coefficient
! Cond              - Conductivity of the wall
! den               - Density of the wall
! cp                - Specific heat capacity of the wall
! Acc_temp          - The required accuracy for solving the
temperature
! T_inside          - Internal air temperature
! Flux_radiative    - Internal radiative heat gain in terms of flux
! Time1             - The beginning time of the step change
! Time2             - The end time of the step change

IMPLICIT NONE

! parameters
INTEGER, PARAMETER :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: prec2 = SELECTED_REAL_KIND(p=12)

! arguments
INTEGER :: n
INTEGER, INTENT(in) ::
time_number
INTEGER, INTENT(in) ::
location_number
INTEGER, DIMENSION(time_number,location_number),INTENT(out) ::
Num_terms
REAL(prec2), DIMENSION(n) ::
realroots
REAL(prec2), DIMENSION(location_number),INTENT(in) ::
Locations
REAL(prec2), DIMENSION(time_number),INTENT(in) ::
Times
REAL(prec2), DIMENSION(time_number,location_number),INTENT(out) ::
accuracy
REAL(prec2), DIMENSION(time_number,location_number),INTENT(out) ::
Temp
REAL(prec2), INTENT(in) ::
Thickness
REAL(prec2), INTENT(in) ::
H_in
REAL(prec2), INTENT(in) ::
Cond
REAL(prec2), INTENT(in) ::
Acc_temp
REAL(prec2), INTENT(in) :: den
REAL(prec2), INTENT(in) :: cp
REAL(prec2), INTENT(in) ::
T_inside
REAL(prec2), INTENT(in) ::
Flux_radiative

```

```

      REAL(prec2), INTENT(in)                                ::
Time1
      REAL(prec2), INTENT(in)                                ::
Time2

      ! local variables
      REAL(prec2)      :: H1 ! An internal term used in solving the roots
      REAL(prec2)      :: func1, func2 , func3 , func4 ,func5
                           ! the corresponding iters used to calculate each
specific pre_remp
      REAL(prec2)      :: pre_temp !a variable used to store the value of Temp
temporarily
      REAL(prec2)      :: Alfa      !the difussivity of the slab (m2/s)
      INTEGER          :: i,j

      H1 = -H_in / Cond
      Alfa = Cond / den / cp

      DO i = 1,time_number

         DO j = 1,location_number

            accuracy(i, j) = 1
            Num_terms(i, j) = 0

         END DO

      END DO

      ! calculate the Temp until the required accuracy is arrived.
      DO i = 1,time_number
         DO j = 1,location_number

            IF (Times(i)<Time1) THEN
               Temp(i,j)=T_inside
               accuracy(i, j) = 0
               Num_terms(i, j) = 0
            ELSE
               !give necessary initial values to the corresponding variables.
               pre_temp = T_inside

               DO WHILE (ABS(accuracy(i, j)) > Acc_temp)

                  !let the Num_terms's value added by 1 at the begining of
each cycle
                  Num_terms(i, j) = Num_terms(i, j) + 1

                  !caculate a new value of the Temp and store it in
present_temp
                  func1 = 2* realroots(Num_terms(i, j))/
(realroots(Num_terms(i, j)) * &
                  Thickness+ SIN(realroots(Num_terms(i,
j))*Thickness)* &
                  COS(realroots(Num_terms(i, j))*Thickness))

```

```

                                func2 = COS(realroots(Num_terms(i, j)) * (Thickness-
Locations(j)))

                                IF (Times(i)<=Time2) THEN
                                    func3 = 1-EXP(Alfa * (realroots(Num_terms(i, j))) ** 2
&
                                        * (Time1-Times(i)))
                                ELSE
                                    func3 = EXP(Alfa * (realroots(Num_terms(i, j))) ** 2 *
&
                                        (Time2-Times(i)))-EXP(Alfa *
(realroots(Num_terms(i, j))) &
                                        ** 2 * (Time1-Times(i)))
                                END IF

                                func4 = COS(realroots(Num_terms(i,
j))*Thickness)*Flux_radiative/ &
                                    Cond/(realroots(Num_terms(i, j))) ** 2
                                func5 = func1 * func2 * func3*func4
                                pre_temp = pre_temp + func5

                                !caculate the value of the accuracy from the second cycle
                                IF (Num_terms(i, j) >= 2) THEN

                                    accuracy(i, j) = pre_temp - Temp(i, j)

                                END IF

                                !transfer the value of pre_temp to Temp
                                Temp(i, j) = pre_temp

                                END DO
                            END IF

                        END DO
                    END DO

                END SUBROUTINE Temperature
            END MODULE IntHeatGain

```

Source Code: Module Grdcoup

MODULE Grdcoup

```

!-----
-
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
-
!
!
! PURPOSE:      Contains utility routines used in the calculation of
analytical
!               solutions to the ground coupling test case.
!
! CREATED:
!   12.3.00     Dongyi Xiao
!
! MODIFIED:
!
PRIVATE

! make public main subroutine
PUBLIC :: GroundCoupling

CONTAINS

!-----
-

SUBROUTINE
GroundCoupling(FileName,Width,Length,Thickness,Conductivity,Tg_outside, &
               T_inside, Hc_A, Hc_C, Hc_n,ErrorFlag,ErrorMessage)

!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "GroundCoupling" :: GroundCoupling

! PURPOSE:      Produces analytical response for the test case of ground
coupling using
!               Delsante's steady state three dimensional slab on floor
conduction model.
!               Successive iteration is used to get the proper floor
temperature under

```

```

!           the convective boundary condition with the zone air
temperature.
!
!   CREATED:
!   12.3.00      Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!
!   FileName.csv      -   Analytical response data
!
! SUBROUTINE ARGUMENTS:
!   FileName          -   Root File Name for data files
!   Width              -   The width of the slab floor
!   Length             -   The length of the slab floor
!   Thickness          -   The thickness of the zone wall
!   Conductivity       -   Thermal conductivity of the slab floor
!   Tg_outside         -   Outside ground temperature
!   T_inside          -   Internal (zone) air temperature
!   Hc_A              -   Inside convection correlation coefficient "A"
!   Hc_C              -   Inside convection correlation coefficient "C"
!   Hc_n              -   Inside convection correlation coefficient "n"
!   ErrorFlag         -   for error checking
!   ErrorMessage      -   for error reporting

IMPLICIT NONE

!parameters
INTEGER, PARAMETER      :: prec1 = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER      :: prec2 = SELECTED_REAL_KIND(p=12)
INTEGER, PARAMETER      :: Results=7 ! unit number for analytical response
output file
INTEGER, PARAMETER      :: Max_Iterations = 5000
REAL(prec2), PARAMETER  :: Tolerance = 1.0E-06 ! tolerance
REAL(prec2),PARAMETER   :: Pai=3.1415926

INTEGER, PARAMETER      :: DebugUnit = 20
LOGICAL, PARAMETER      :: Debug = .false.

!arguments
REAL(prec2), INTENT(in)      :: Width,Length
REAL(prec2), INTENT(in)      :: Thickness
REAL(prec2), INTENT(in)      :: Conductivity

REAL(prec2), INTENT(in)      :: Tg_outside
REAL(prec2), INTENT(in)      :: T_inside
REAL(prec2), INTENT(in)      :: Hc_A, Hc_C, Hc_n

CHARACTER *(*) , INTENT(in)  :: FileName
CHARACTER*(*) , INTENT(out)  :: ErrorMessage

```

```

    INTEGER(2), INTENT(out)                :: ErrorFlag    ! two byte int for
VB integer

    !local variables
    INTEGER      :: n,error

    REAL(prec2) :: Q                ! The conduction heat flow through the floor
    REAL(prec2) :: Q_conv           ! The convection heat flow at the floor
    REAL(prec2) :: Area_floor       ! The area of the floor
    REAL(prec2) :: Hc_floor         ! The convection coefficient at the floor
    REAL(prec2) :: Q_F             ! The function used in calculation of Q
    REAL(prec2) :: G,D             ! The parameter used in calculation of Q
    REAL(prec2),dimension(17)      :: Part_F    ! The array used to store
different parts &
                                           ! of the function in calculation
of Q
    REAL(prec2)                :: Load
    REAL(prec2)                :: T_floor

    ! compute the function value of Q_F
    G=Length-Width
    D=Length+Width
    Area_floor=Length*Width

Part_F(1)=(2+(Length+Width)/Thickness)*((Length+Thickness)**2+(Width+Thickness)**2)**0.5

Part_F(2)=2**0.5*(1+Length/2/Thickness)*(Length**2+(Length+2*Thickness)**2)**0.5

Part_F(3)=2**0.5*(1+Width/2/Thickness)*(Width**2+(Width+2*Thickness)**2)**0.5

Part_F(4)=((Length+Width)/Thickness)*(Length**2+Width**2)**0.5

Part_F(5)=((Length**2+Width**2)/Thickness)*(1+2**0.5*LOG(2**0.5-1))

Part_F(6)=2*Thickness*(2**0.5+LOG(2**0.5-1))

Part_F(7)=2**0.5*G**2/Thickness*LOG(((G**2+(D+2*Thickness)**2)**0.5+ &
D+2*Thickness)/((G**2+D**2)**0.5+D))

Part_F(8)=(G**2-(Width+Thickness)**2)/Thickness&
*LOG(((Length+Thickness)**2+(Width+Thickness)**2)**0.5+ &
Width+Thickness)/(Length+Thickness))

Part_F(9)=(G**2-(Length+Thickness)**2)/Thickness&
*LOG(((Length+Thickness)**2+(Width+Thickness)**2)**0.5+ &
Length+Thickness)/(Width+Thickness))

Part_F(10)=Length*(2*Width-
Length)/Thickness*LOG(((Length**2+Width**2)**0.5+Width)/Length)

```

```

Part_F(11)=Width*(2*Length-
Width)/Thickness*LOG(((Length**2+Width**2)**0.5+Length)/Width)

Part_F(12)=(Length**2-Thickness**2)/Thickness&
*LOG(((Thickness**2+(Length+Thickness)**2)**0.5+Thickness)/(Length+Thickness))

Part_F(13)=(Width**2-Thickness**2)/Thickness&
*LOG(((Thickness**2+(Width+Thickness)**2)**0.5+Thickness)/(Width+Thickness))

Part_F(14)=(2*Length+Thickness)&
*LOG(((Thickness**2+(Length+Thickness)**2)**0.5+Length+Thickness)/Thickness)

Part_F(15)=(2*Width+Thickness)&
*LOG(((Thickness**2+(Width+Thickness)**2)**0.5+Width+Thickness)/Thickness)

Part_F(16)=2**0.5*Length**2/Thickness&
*LOG(((Length**2+(Length+2*Thickness)**2)**0.5+Length+2*Thickness)/Length)

Part_F(17)=2**0.5*Width**2/Thickness&
*LOG(((Width**2+(Width+2*Thickness)**2)**0.5+Width+2*Thickness)/Width)

Q_F=Part_F(1)-Part_F(2)-Part_F(3)-Part_F(4)+Part_F(5)+Part_F(6)- &
Part_F(7)+Part_F(8)+Part_F(9)&
+Part_F(10)+Part_F(11)-Part_F(12)-Part_F(13)+Part_F(14)+ &
Part_F(15)+Part_F(16)+Part_F(17)

! start loop to calculate the heat flow through the floor

! open a unit for debug output
IF(Debug .and. DebugUnit/=6)THEN
  OPEN (DebugUnit, file="GrdCoup.log", status='unknown', IOSTAT=error)
  IF(error/=0) THEN
    ErrorMessage="Debug output file could not be opened"
    ErrorFlag = 1
    RETURN
  END IF
  ! copy out arguments to debug file
  WRITE(DebugUnit,'("GrdCoup arguments:")')
  WRITE(DebugUnit,'("Inside convection correlation coefficient A
=" ,F15.4)') Hc_A
  WRITE(DebugUnit,'("Inside convection correlation coefficient C
=" ,F15.4)') Hc_C
  WRITE(DebugUnit,'("Inside convection correlation coefficient n
=" ,F15.4)') Hc_n
  WRITE(DebugUnit,'("Outside ground temperature
=" ,
&
F15.4)') Tg_outside
  WRITE(DebugUnit,'("Inside air temperature
=" , &
F15.4)') T_inside

```



```

END IF

! set initial guess of floor temperatures
T_floor = T_inside - 10.0

! compute the convection coefficient based on current floor temperature
Hc_floor = Hc_A + Hc_C*ABS(T_inside - T_floor)**Hc_n

! start loop to find heat flow by successive substitution
n = 1
DO
  IF(Hc_floor==0.0)THEN
    WRITE(ErrorMessage, '("Inside convection coefficient calculated to
be zero", &
                                " - can not procede"))' )
    ErrorFlag = 1
    RETURN
  END IF

  ! Convective heat flow based on current value of convection coef.
  Q_conv= (T_inside - T_floor)*Hc_floor*Area_floor

  ! calculate heat flow from analytical model for checking
  Q=Conductivity*(T_floor-Tg_outside)*Q_F/Pai

  ! update floor temps based on heat flow
  !T_floor = T_inside - Q/Hc_floor/Area_floor
  !try introducing underrelaxation
  T_floor = T_floor+0.1*(T_inside - Q/Hc_floor/Area_floor-T_floor)

  ! re-calculate conv. coef using current temps
  !Hc_floor = Hc_A + Hc_C*ABS(T_inside - T_floor)**Hc_n
  !try introducing underrelaxation
  Hc_floor = Hc_floor+0.1*(Hc_A + Hc_C*ABS(T_inside - T_floor)**Hc_n-
Hc_floor)

  ! some debug output
  IF(Debug)THEN
    WRITE(DebugUnit, '("Hc_floor = ",F8.5," T_floor= ",F8.5)')
Hc_floor, T_floor
    WRITE(DebugUnit, '("Q_conv = ",F8.5," Q = ",F8.5)') Q_conv, Q
  END IF

  ! test for exit condition
  IF(ABS(Q_conv-Q) < Tolerance .or. n > Max_Iterations) EXIT
  n = n + 1
END DO

! set warning if max iterations exceded
IF(n > Max_Iterations)THEN
  ErrorMessage = "The steady-state heat flux could not be calculated &
                  within the allowable number of iterations"
  ErrorFlag = -1
  RETURN
END IF

```

```

IF(Debug)THEN
    CLOSE(DebugUnit)
END IF

Load=Q

!attempt to open file - full path is passed as FileName by VB application
OPEN (Results, file=TRIM(FileName), status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrorMessage="Analytical results output file could not be opened"
    ErrorFlag = 1
    RETURN
END IF

! write header lines
WRITE(Results, '( "# ASHRAE Analytical Test Suite - Results file" )' )
WRITE(Results, '( "# Ground coupling test" )' )
WRITE(Results, '( // "Test Parameters -" )' )

WRITE(Results, '( "Inside air temperature =", " ", F15.4 )' ) T_inside
WRITE(Results, '( "Outside ground temperature =", " ", F15.4 )' ) Tg_outside
WRITE(Results, '( "Length of the slab floor =", " ", F15.4 )' ) Length
WRITE(Results, '( "Width of the slab floor =", " ", F15.4 )' ) Width
WRITE(Results, '( "Thickness of the wall =", " ", F15.4 )' ) Thickness
WRITE(Results, '( "Conductivity of the floor =", " ", F15.4 )' ) Conductivity
WRITE(Results, '( "Inside convection correlation coefficient A =", " ", &
    F15.4 )' ) Hc_A
WRITE(Results, '( "Inside convection correlation coefficient C =", " ", &
    F15.4 )' ) Hc_C
WRITE(Results, '( "Inside convection correlation coefficient n =", " ", &
    F15.4 )' ) Hc_n

WRITE(Results, '( // "Test results -" )' )
! WRITE(Results, '( // "Conduction Heat flow through the slab
=" , F20.6 )' ) Q
! WRITE(Results, '( // "Convection Heat flow through the slab      =", &
! F20.6 )' ) Q_conv
WRITE(Results, '( // "Convection coefficient of the floor =", " ", F20.6 )' )
Hc_floor
WRITE(Results, '( // "Inside floor temperature =", " ", F20.6 )' ) T_floor
WRITE(Results, '( // "Zone Load =", " ", F20.6 )' ) load

CLOSE(Results)

END SUBROUTINE GroundCoupling

END MODULE Grdcoup

```

Source Code: Module Tmy2Util

```

MODULE Tmy2Util

!-----
--
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
--
!
!
! PURPOSE:      Contains data structure definition for dealing with Tmy2
!               weather files and routines for manipulating certain fields
in
!               the data.
!
! CREATED:
!   11.7.99      S.J.Rees
!
! MODIFIED:
!   10.20.2000   Dongyi Xiao
!-----
--
IMPLICIT NONE
SAVE
!PRIVATE ! default is for everything to be private

! set specific procedures public
PUBLIC :: SetNewTmy2
PUBLIC :: SetTmy2Header

PUBLIC :: SetTmy2DryBulb
PUBLIC :: SetTmy2RelHumid

PUBLIC :: SetTmy2DirNormIrr
PUBLIC :: SetTmy2DewPoint
PUBLIC :: SetTmy2Pressure
PUBLIC :: SetTmy2Temps
PUBLIC :: WriteTmy2

! general prameters
INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: Tmy2out=20           ! unit number for user output file

```

```

INTEGER, PARAMETER :: N_Hours = 8760      ! Number of hours in year

!----- Tmy2 data structure -----
--
! variable elements
INTEGER, DIMENSION(N_Hours) :: DryBulb    ! Int value of Drybulb*10 (C)
INTEGER, DIMENSION(N_Hours) :: DewPoint   ! Dew point temp*10 (C)
INTEGER, DIMENSION(N_Hours) :: RelHumid   ! Relative Humidity (%)
INTEGER, DIMENSION(N_Hours) :: Pressure   ! pressure (Millibars)
INTEGER, DIMENSION(N_Hours) :: DirNormIrr ! Direct normal irradiance (Wh/m2)

INTEGER, DIMENSION(N_Hours) :: Month      ! Month of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Day        ! Day of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Hour       ! Hour of record - 2 digit

CHARACTER(LEN=22) :: City="ASHRAE Analytical test" ! City name - 22 char
CHARACTER(LEN=2)  :: State="OK" ! State
INTEGER           :: TimeZone= -6    ! Time zone (relative to UST/GMT) - 3
digit
                                ! -6 is CST
CHARACTER(LEN=1)  :: Lat_F="N" ! Latitude - N/S flag
INTEGER           :: Lat_Deg= 0   ! Latitude - Degrees component
INTEGER           :: Lat_Min= 0   ! Latitude - Minutes component
CHARACTER(LEN=1)  :: Long_F="W" ! Longitude - W/E flag
INTEGER           :: Long_Deg= 0  ! Longitude - Degrees component
INTEGER           :: Long_Min= 0  ! Longitude - Minutes component
INTEGER           :: Elevation = 0 ! Elevation A.S.L. (m)

! parameterized fields in order
! TMY2 header fields
INTEGER, PARAMETER :: WBAN          = 99999 ! weather station WBAN number - 5
digit

! Main data field parameters
INTEGER, PARAMETER :: Year          = 99    ! year of records - 2 digit

! solar irradiance data - note units are (Wh/m2)
INTEGER, PARAMETER :: ExtraHorIrr = 0
                                ! Extraterrestrial Horizontal irradiance - 4 digit
INTEGER, PARAMETER :: ExtraNormIrr = 0
                                ! Extraterrestrial direct normal irradiance - 4 digit
INTEGER, PARAMETER :: GlobHorIrr  = 0 ! Global Horizontal irradiance - 4
digit
CHARACTER(LEN=2), PARAMETER :: GlobHorIrr_F = "?0" ! Flag for above - 2 char
! INTEGER, PARAMETER :: DirNormIrr  = 0    ! Direct normal irradiance - 4
digit
! CHARACTER(LEN=2), PARAMETER :: DirNormIrr_F = "?0" ! Flag for above - 2
char
INTEGER, PARAMETER :: DiffHorIrr  = 0    ! Diffuse Horizontal irradiance -
4 digit
CHARACTER(LEN=2), PARAMETER :: DiffHorIrr_F = "?0" ! Flag for above - 2 char

! Illuminance data (lux/100)
INTEGER, PARAMETER :: GlobHorIll  = 0    ! Global Horizontal illuminance -
4 digit
CHARACTER(LEN=2), PARAMETER :: GlobHorIll_F = "?0" ! Flag for above - 2 char

```

```

    INTEGER, PARAMETER :: DirNormIll = 0      ! Direct normal illuminance - 4
digit
    CHARACTER(LEN=2), PARAMETER :: DirNormIll_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: DiffHorIll = 0      ! Diffuse Horizontal illuminance -
4 digit
    CHARACTER(LEN=2), PARAMETER :: DiffHorIll_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: ZenithIll = 0       ! Zenith illuminance - 4 digit
    CHARACTER(LEN=2), PARAMETER :: ZenithIll_F = "?0" ! Flag for above - 2 char

    ! sky conditions
    INTEGER, PARAMETER :: TotSky = 0          ! Total sky cover (tenths) - 2
digit
    CHARACTER(LEN=2), PARAMETER :: TotSky_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: OpaqSky = 0         ! Opaque sky cover (tenths) - 2
digit
    CHARACTER(LEN=2), PARAMETER :: OpaqSky_F = "?0" ! Flag for above - 2 char

    ! air conditions
    ! dry bulb is variable
    CHARACTER(LEN=2) :: DryBulb_F             ! Flag for DryBulb - 2 char
    ! dew point is variable
    CHARACTER(LEN=2) :: DewPoint_F            ! Flag for dew point - 2 char
    ! relative humidity is variable
    CHARACTER(LEN=2) :: RelHumid_F            ! Flag for relative humidity - 2
char
    ! pressure is variable
    CHARACTER(LEN=2) :: Pressure_F            ! Flag for pressure - 2 char
    ! DirNormIrr is variable
    CHARACTER(LEN=2) :: DirNormIrr_F          ! Flag for DirNormIrr - 2 char
    ! wind conditions
    INTEGER, PARAMETER :: WindDir = 0         ! Wind direction (0-360) N=0 or
360 - 3 digit
    CHARACTER(LEN=2), PARAMETER :: WindDir_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: WindSpeed = 0       ! Wind speed (m/s*10) - 4 digit
    CHARACTER(LEN=2), PARAMETER :: WindSpeed_F = "?0" ! Flag for above - 2 char

    INTEGER, PARAMETER :: Vis = 9999         ! Visability - 4 digit
    CHARACTER(LEN=2), PARAMETER :: Vis_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: CeilHeight = 9999 ! ceiling height, m/10 - 4 digit
    CHARACTER(LEN=2), PARAMETER :: CeilHeight_F = "?0" ! Flag for above - 2 char

    ! other useful stuff
    CHARACTER(LEN=10), PARAMETER :: Weather = "9999999999" ! weather
discription - 10 digit
    INTEGER, PARAMETER :: Precip = 0         ! Precipitation (mm) at the hour
- 3 digit
    CHARACTER(LEN=2), PARAMETER :: Precip_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: OptDepth = 0 ! Broadband Aerosol Optical depth,
thousandths - 3 digit
    CHARACTER(LEN=2), PARAMETER :: OptDepth_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: SnowDepth = 0      ! Snow depth (cm) on day
indicated - 3 digit
    CHARACTER(LEN=2), PARAMETER :: SnowDepth_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: DaysLastSnow = 88  ! Days since last snowfall (0 -
88) - 2 digit

```

```

    CHARACTER(LEN=2), PARAMETER :: DaysLastSnow_F = "?0" ! Flag for above - 2
    char

    CONTAINS

    !-----
    --

    SUBROUTINE SetNewTmy2(Filename, ErrFlag, ErrMess)
        !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetNewTmy2" :: SetNewTmy2

        ! PURPOSE:      Initializes the Tmy2 data structure and opens a file for
        !                  output.
        !
        ! CREATED:
        !   5.13.99      S.J.Rees
        !
        ! MODIFIED:
        !
        ! INPUT FILES:
        ! n/a
        !
        ! OUTPUT FILES:
        !   Filename      -   see arguments
        !
        ! SUBROUTINE ARGUMENTS:
        !   Filename      -   user defined name for creation of Tmy2 format file
        !   ErrFlag       -   for global error checking
        !   ErrMess       -   for error reporting

    IMPLICIT NONE

        ! arguments
        CHARACTER*(*) , INTENT(IN)          :: Filename
        CHARACTER*(*) , INTENT(OUT)         :: ErrMess
        INTEGER(2) , INTENT(OUT)            :: ErrFlag

        !local variables
        INTEGER :: i, j, error

        ErrFlag = 0
        ErrMess = ' '

        ! open file for output
        OPEN (Tmy2out, file=Filename, status='unknown', IOSTAT=error)
        IF(error/=0) THEN
            ErrMess = "TMY2 output file could not be opened"
            ErrFlag = 1
            RETURN
        END IF

        !initialize month, day and hour arrays
        DO i=1, N_Hours
            j = (i-1)/24 + 1 ! day no (1-365)
            Hour(i) = i - (j-1)*24 ! derive hour no (1-24)
            ! select month based on day no.
            SELECT CASE (j)

```

```

      CASE(1:31)           ! jan
        Month(i) = 1
        Day(i) = j
      CASE(32:59)         ! feb
        Month(i) = 2
        Day(i) = j - 31
      CASE(60:90)         ! march
        Month(i) = 3
        Day(i) = j - 59
      CASE(91:120)        ! april
        Month(i) = 4
        Day(i) = j - 90
      CASE(121:151)       ! may
        Month(i) = 5
        Day(i) = j - 120
      CASE(152:181)       ! june
        Month(i) = 6
        Day(i) = j - 151
      CASE(182:212)       ! july
        Month(i) = 7
        Day(i) = j - 181
      CASE(213:243)       ! aug
        Month(i) = 8
        Day(i) = j - 212
      CASE(244:273)       ! sept
        Month(i) = 9
        Day(i) = j - 243
      CASE(274:304)       ! oct
        Month(i) = 10
        Day(i) = j - 273
      CASE(305:334)       ! nov
        Month(i) = 11
        Day(i) = j - 304
      CASE(335:365)       ! dec
        Month(i) = 12
        Day(i) = j - 334
    END SELECT
  END DO

  ! specifically initialize other arrays
  DryBulb = 0
  RelHumid= 100
  DewPoint = 0
  !RelHumid= 0
  !DewPoint = 9999

  Pressure = 1000
  DirNormIrr= 0
  ! set temp flags to "?0" indicating "missing" by default
  DryBulb_F = "?0"
  DewPoint_F = "?0"
  Pressure_F = "?0"
  RelHumid_F = "?0"
  DirNormIrr_F= "?0"

  RETURN

```

```

END    SUBROUTINE SetNewTmy2

!-----
--
SUBROUTINE SetTmy2Header (Location, ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2Header" :: SetTmy2Header

  ! PURPOSE:      Used to set the header fields. REAL header fields
  !                is set differently according to the location.
  !
  ! CREATED:
  !   21.10.2000      Dongyi Xiao
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   n/a
  !
  ! SUBROUTINE ARGUMENTS:
  !   Location      -   index of location
  !   ErrFlag       -   for global error checking
  !   ErrMess       -   for error reporting

  IMPLICIT NONE

  INTEGER, INTENT(IN) :: Location
  CHARACTER*(*) , INTENT(OUT) :: ErrMess
  INTEGER(2), INTENT(OUT) :: ErrFlag
  INTEGER i,error

  ErrFlag = 0
  ErrMess = ' '

  ! TMY2 header fields
  IF (Location==0) THEN

    City="Atlanta" ! City name - 22 char
    State="GA" ! State
    TimeZone = -5      ! Time zone (relative to UST/GMT) - 3 digit
                        ! -6 is CST

    Lat_F ="N" ! Latitude - N/S flag
    Lat_Deg = 33      ! Latitude - Degrees component
    Lat_Min = 46      ! Latitude - Minutes component
    Long_F ="W"! Longitude - W/E flag
    Long_Deg = 84      ! Longitude - Degrees component
    Long_Min = 25      ! Longitude - Minutes component
    Elevation = 306    ! Elevation A.S.L. (m)
  ELSE IF (Location==1) THEN
    City="Boston" ! City name - 22 char
    State="MA" ! State
    TimeZone = -5      ! Time zone (relative to UST/GMT) - 3 digit
                        ! -6 is CST

    Lat_F ="N" ! Latitude - N/S flag
    Lat_Deg = 42      ! Latitude - Degrees component

```



```

    Lat_Min = 19          ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 71         ! Longitude - Degrees component
    Long_Min = 5          ! Longitude - Minutes component
    Elevation = 5         ! Elevation A.S.L. (m)
ELSE IF (Location==2) THEN
    City="Chicago" ! City name - 22 char
    State="IL" ! State
    TimeZone = -6       ! Time zone (relative to UST/GMT) - 3 digit
                        ! -6 is CST

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 41        ! Latitude - Degrees component
    Lat_Min = 51        ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 87        ! Longitude - Degrees component
    Long_Min = 41        ! Longitude - Minutes component
    Elevation = 186      ! Elevation A.S.L. (m)
ELSE IF (Location==3) THEN
    City="Los Angeles" ! City name - 22 char
    State="CA" ! State
    TimeZone = -8       ! Time zone (relative to UST/GMT) - 3 digit
                        ! -6 is CST

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 34        ! Latitude - Degrees component
    Lat_Min = 5         ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 118       ! Longitude - Degrees component
    Long_Min = 22        ! Longitude - Minutes component
    Elevation = 30       ! Elevation A.S.L. (m)
ELSE
    City="ASHRAE Analytical test" ! City name - 22 char
    State="OK" ! State
    TimeZone = 0         ! Time zone (relative to UST/GMT) - 3 digit
                        ! -6 is CST

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 0         ! Latitude - Degrees component
    Lat_Min = 0         ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 0         ! Longitude - Degrees component
    Long_Min = 0         ! Longitude - Minutes component
    Elevation = 0       ! Elevation A.S.L. (m)
ENDIF

END SUBROUTINE SetTmy2Header

!-----
--

SUBROUTINE SetTmy2DryBulb (RealDryBulb, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2DryBulb" :: SetTmy2DryBulb

    ! PURPOSE:      Used to set the drybulb temperatures. REAL dry bulb temp
array
    !              is converted to integer array for file format.
    !
    ! CREATED:
    !   5.13.99      S.J.Rees

```

```

!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
! n/a
!
! SUBROUTINE ARGUMENTS:
!   RealDryBulb      -   Array of real dry bulb temperatures
!   N_hrs            -   no. of hours (for VB dll calls)
!   ErrFlag          -   for global error checking
!   ErrMess          -   for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDryBulb
CHARACTER*(*) , INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag
INTEGER i,error

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDryBulb) /= N_Hours) THEN
    ErrMess = "Drybulb array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DryBulb = 10.0 * RealDryBulb
! set flag
!DryBulb_F = "B7"

END SUBROUTINE SetTmy2DryBulb

!-----
--

SUBROUTINE SetTmy2DewPoint (RealDewPoint, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2DewPoint" :: SetTmy2DewPoint

! PURPOSE:      Used to set the dew point temperatures. REAL dew point
temp array
!               is converted to integer array for file format.
!
! CREATED:
!   5.18.99      S.J.Rees
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!

```

```

! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   RealDewPoint   -   Array of real dew point temperatures (each hour of
year)
!   N_hrs          -   no. of hours (for VB dll calls)
!   ErrFlag        -   for global error checking
!   ErrMess        -   for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDewPoint
INTEGER i,error
CHARACTER*(*) , INTENT(OUT)      :: ErrMess
INTEGER(2), INTENT(OUT)          :: ErrFlag

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDewPoint) /= N_Hours) THEN
    ErrMess = "Error: dew point array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DewPoint = 10.0 * RealDewPoint
! set flag
!DewPoint_F = "E8"

END SUBROUTINE SetTmy2DewPoint

!-----
--

SUBROUTINE SetTmy2Temps (RealDryBulb, N1_hrs, RealDewPoint, N2_hrs, ErrFlag,
ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2Temps" :: SetTmy2Temps

! PURPOSE:      Used to set both the dry bulb and dew point temperatures.
REAL
!               arrays are converted to integer array for file format.
!
! CREATED:
!   5.18.99      S.J.Rees
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:

```

```

!   RealDryBulb      -   Array of real dry bulb temperatures (each hour of
year)
!   N1_hrs           -   no. of hours (for VB dll calls)
!   RealDewPoint     -   Array of real dew point temperatures (each hour
of year)
!   N2_hrs           -   no. of hours (for VB dll calls)
!   ErrFlag          -   for global error checking
!   ErrMess          -   for error reporting

```

```

IMPLICIT NONE

```

```

INTEGER, INTENT(IN) :: N1_hrs, N2_hrs
REAL(prec), DIMENSION(N1_hrs), INTENT(IN) :: RealDryBulb
REAL(prec), DIMENSION(N2_hrs), INTENT(IN) :: RealDewPoint
CHARACTER*(*), INTENT(OUT)      :: ErrMess
INTEGER(2), INTENT(OUT)         :: ErrFlag
INTEGER i,error

```

```

ErrFlag = 0
ErrMess = ' '

```

```

CALL SetTmy2DryBulb(RealDryBulb, N1_hrs, ErrFlag, ErrMess)
IF(ErrFlag/=0) RETURN
CALL SetTmy2DewPoint(RealDewPoint, N2_hrs, ErrFlag, ErrMess)
IF(ErrFlag/=0) RETURN

```

```

END SUBROUTINE SetTmy2Temps

```

```

!-----
--

```

```

SUBROUTINE SetTmy2Pressure (RealPressure, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2Pressure" :: SetTmy2Pressure

```

```

! PURPOSE:      Used to set the Pressure. REAL pressure array
!               is converted to integer array for file format.
!

```

```

! CREATED:
!   5.18.99      S.J.Rees
!

```

```

! MODIFIED:
!

```

```

! INPUT FILES:
!   n/a
!

```

```

! OUTPUT FILES:
!   n/a
!

```

```

! SUBROUTINE ARGUMENTS:

```

```

!   RealPressure    -   Array of real pressures (Kpa each hour of year)
!   N_hrs           -   no. of hours (for VB dll calls)
!   ErrFlag         -   for global error checking
!   ErrMess         -   for error reporting

```

```

IMPLICIT NONE

```

```

INTEGER, INTENT(IN) :: N_hrs

```

```

REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealPressure
CHARACTER*(*), INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealPressure) /= N_Hours) THEN
    ErrMess = "Error: Pressure array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

Pressure = 10.0 * RealPressure
! set flag
!Pressure_F = "A7"

END SUBROUTINE SetTmy2Pressure

!-----
--

SUBROUTINE SetTmy2DirNormIrr (RealDirNormIrr, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2DirNormIrr" ::
SetTmy2DirNormIrr

! PURPOSE:      Used to set the Direct normal irradiance. REAL Direct
normal
!               irradiance array is converted to integer array for file
format.
!
! CREATED:
! 11.7.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
! n/a
!
! SUBROUTINE ARGUMENTS:
! RealDirNormIrr - Array of real Direct normal irradiance (J/m2 each
hour of year)
! N_hrs          - no. of hours (for VB dll calls)
! ErrFlag        - for global error checking
! ErrMess        - for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDirNormIrr
INTEGER i,error
CHARACTER*(*), INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag

```

```

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDirNormIrr) /= N_Hours) THEN
    ErrMess = "Error: dew point array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DirNormIrr = RealDirNormIrr/3600.0
! set flag
!DewPoint_F = "E8"

END SUBROUTINE SetTmy2DirNormIrr

!-----
--

SUBROUTINE SetTmy2RelHumid (RealRelHumid, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetTmy2RelHumid" :: SetTmy2RelHumid

    ! PURPOSE:      Used to set the relative humidities. REAL relative
humidity array
    !              is converted to integer array for file format.
    !
    ! CREATED:
    !   5.13.99      S.J.Rees
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealRelHumid - Array of real relative humidities
    !   N_hrs        - no. of hours (for VB dll calls)
    !   ErrFlag      - for global error checking
    !   ErrMess      - for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N_hrs
    REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealRelHumid
    CHARACTER*(*) , INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag
    INTEGER i,error

    ErrFlag = 0
    ErrMess = ' '

    IF(SIZE(RealRelHumid) /= N_Hours) THEN
        ErrMess = "RealRelHumid array inconsistant size"

```

```

        ErrFlag = 1
        RETURN
    END IF

    RelHumid = 100.0 * RealRelHumid
    ! set flag
    !DryBulb_F = "B7"

END SUBROUTINE SetTmy2RelHumid

!-----
--

SUBROUTINE WriteTmy2(ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "WriteTmy2" :: WriteTmy2

    ! PURPOSE:      Writes out the Tmy2 data structure. This is to be called
    !                after initialization and setting of required fields.
    !
    ! CREATED:
    !   5.13.99      S.J.Rees
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   Filename      -   as defined in 'SetNewTmy2'
    !
    ! SUBROUTINE ARGUMENTS:
    !
    !   ErrFlag        -   for global error checking
    !   ErrMess        -   for error reporting

    IMPLICIT NONE

    CHARACTER*(*) , INTENT(OUT)      :: ErrMess
    INTEGER(2) , INTENT(OUT)         :: ErrFlag
    INTEGER i,error

    ErrFlag = 0
    ErrMess = ' '

    ! write header data
    WRITE(Tmy2out,90, IOSTAT=error) WBAN, City, State, TimeZone, Lat_F
,Lat_Deg, Lat_Min, &
                                Long_F, Long_Deg, Long_Min, Elevation
    IF(error/=0) THEN
        ErrMess = "Error: Tmy2 file header could not be written"
        ErrFlag = 1
        RETURN
    END IF

    ! write out hourly data
    DO i=1, N_Hours

```

```

        WRITE(Tmy2out, 100, IOSTAT=error) Year, Month(i), Day(i), Hour(i),
ExtraHorIrr, &
                                ExtraNormIrr, GlobHorIrr, GlobHorIrr_F,
DirNormIrr(i), &
                                DirNormIrr_F, DiffHorIrr, DiffHorIrr_F,
GlobHorIll, &
                                GlobHorIll_F, DirNormIll, DirNormIll_F,
DiffHorIll, &
                                DiffHorIll_F, ZenithIll, ZenithIll_F, &
                                TotSky, TotSky_F, OpaqSky, OpaqSky_F, DryBulb(i),
&
                                DryBulb_F, DewPoint(i), DewPoint_F, RelHumid(i),
&
                                RelHumid_F, Pressure(i), Pressure_F, WindDir,
WindDir_F, &
                                WindSpeed, WindSpeed_F, Vis, Vis_F, CeilHeight, &
                                CeilHeight_F, Weather, Precip, Precip_F,
OptDepth, &
                                OptDepth_F, SnowDepth, SnowDepth_F, DaysLastSnow,
&
                                DaysLastSnow_F

        IF(error/=0) THEN
            ErrMess = "Error: Tmy2 file data could not be written"
            ErrFlag = 1
            RETURN
        END IF

    END DO

    ! close weather file
    CLOSE(Tmy2out)

    ! header and main data formats - adapted from TMY2 manual
90  FORMAT( 1X,I5.5,1X,A22,1X,A2,1X
I3,1X,A1,1X,I2,1X,I2,1X,A1,1X,I3,1X,I2,2X,I4 )

100 FORMAT(1X, 4I2.2, 2I4.4, 7(I4.4, A2), 2(I2.2, A2), 2(I4.3, A2), 1(I3.3,
A2), &
            1(I4.4, A2), 2(I3.3, A2), 1(I4.4, A2), 1(I5.5, A2), A10, 3(I3.3,
A2), 1(I2.2, A2))

    END SUBROUTINE WriteTmy2

END MODULE Tmy2Util

```


Source Code: Module Wyec2Util

```

MODULE Wyec2Util

!-----
--
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
--
!
!
! PURPOSE:      Contains data structure definition for dealing with wyec2
!               weather files and routines for manipulating certain fields
in
!               the data.
!
! CREATED:
!   11.7.99      S.J.Rees, Dongyi Xiao
!
! MODIFIED:
!
!-----
--
IMPLICIT NONE
SAVE
PRIVATE ! default is for everything to be private

! set specific procedures public
PUBLIC :: SetNewWyec2
PUBLIC :: SetWyec2DryBulb
PUBLIC :: SetWyec2RelHumid

PUBLIC :: SetWyec2DewPoint
PUBLIC :: SetWyec2Pressure
PUBLIC :: SetWyec2Temps
PUBLIC :: SetWyec2DirNormIrr
PUBLIC :: WriteWyec2

! general parameters
INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: wyec2out=19      ! unit number for user output file
INTEGER, PARAMETER :: N_Hours = 8760    ! Number of hours in year

```

```

!----- WYEC2 data structure -----
--
! variable elements
INTEGER, DIMENSION(N_Hours) :: DryBulb      ! Int value of Drybulb*10 (C)
INTEGER, DIMENSION(N_Hours) :: DewPoint     ! Dew point temp*10 (C)
INTEGER, DIMENSION(N_Hours) :: Pressure     ! pressure (Kpa*100)
INTEGER, DIMENSION(N_Hours) :: DirNormIrr   ! Direct normal irradiance (KJ/m2)
INTEGER, DIMENSION(N_Hours) :: RelHumid     ! Relative humidity (%)

INTEGER, DIMENSION(N_Hours) :: Month        ! Month of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Day          ! Day of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Hour         ! Hour of record - 2 digit

! parameterized fields in order
INTEGER, PARAMETER :: WBAN = 0             ! weather station WBAN number - 5
digit
CHARACTER(LEN=1), PARAMETER :: Source="W"  ! source of file
INTEGER, PARAMETER :: Year = 99            ! year of records - 2 digit

! solar irradiance data (KJ/m2)
INTEGER, PARAMETER :: ExtraIrr = 0         ! Extraterrestrial irradiance - 4
digit
INTEGER, PARAMETER :: GlobHorIrr = 0       ! Global Horizontal irradiance - 4
digit
CHARACTER(LEN=2), PARAMETER :: GlobHorIrr_F = "A " ! Flag for above - 2 char
INTEGER, PARAMETER :: DiffHorIrr = 0       ! Diffuse Horizontal irradiance -
4 digit
CHARACTER(LEN=2), PARAMETER :: DiffHorIrr_F = "A " ! Flag for above - 2 char

! Illuminance data (lux/100)
INTEGER, PARAMETER :: GlobHorIll = 0       ! Global Horizontal illuminance -
4 digit
CHARACTER(LEN=1), PARAMETER :: GlobHorIll_F = "A" ! Flag for above - 1 char
INTEGER, PARAMETER :: DirNormIll = 0       ! Direct normal illuminance - 4
digit
CHARACTER(LEN=1), PARAMETER :: DirNormIll_F = "A" ! Flag for above - 1 char
INTEGER, PARAMETER :: DiffHorIll = 0       ! Diffuse Horizontal illuminance -
4 digit
CHARACTER(LEN=1), PARAMETER :: DiffHorIll_F = "A" ! Flag for above - 1 char
INTEGER, PARAMETER :: ZenithIll = 0        ! Zenith illuminance - 4 digit
CHARACTER(LEN=1), PARAMETER :: ZenithIll_F = "A" ! Flag for above - 1 char

! sky conditions
INTEGER, PARAMETER :: MinSun = 99          ! minutes of sunshine - 2 digit
CHARACTER(LEN=1), PARAMETER :: MinSun_F = "A" ! Flag for above - 1 char
INTEGER, PARAMETER :: CeilHeight = 9999    ! ceiling height, m/10 - 4 digit
CHARACTER(LEN=1), PARAMETER :: CeilHeight_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: SkyCond = 9999       ! Sky condition - 4 digit
CHARACTER(LEN=1), PARAMETER :: SkyCond_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: Vis = 9999           ! Visibility - 4 digit
CHARACTER(LEN=1), PARAMETER :: Vis_F = "9"! Flag for above - 1 char

! weather discription
INTEGER, PARAMETER :: Weather = 99999999  ! weather discription - 8 digit
CHARACTER(LEN=1), PARAMETER :: Weather_F = "9" ! Flag for above - 1
char

```

```

! other useful stuff
CHARACTER(LEN=1) :: Pressure_F           ! Flag for above - 1 char
! dry bulb is variable
CHARACTER(LEN=1) :: DryBulb_F           ! Flag for DryBulb - 1 char
! relative humidity is variable
CHARACTER(LEN=1) :: Relhumid_F          ! Flag for Relhumid - 1 char

! dew point is variable
CHARACTER(LEN=1) :: DewPoint_F          ! Flag for above - 1 char
! Direct normal irradiance is variable
CHARACTER(LEN=2) :: DirNormIrr_F        ! Flag for above - 2 char

INTEGER, PARAMETER :: WindDir = 0       ! Wind direction (0-359) - 3 digit
CHARACTER(LEN=1), PARAMETER :: WindDir_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: WindSpeed = 0     ! Wind speed (m/s*10) - 4 digit
CHARACTER(LEN=1), PARAMETER :: WindSpeed_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: TotSky = 0        ! Total sky cover (tenths) - 2
digit
CHARACTER(LEN=1), PARAMETER :: TotSky_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: OpaqSky = 0        ! Opaque sky cover (tenths) - 2
digit
CHARACTER(LEN=1), PARAMETER :: OpaqSky_F = "9" ! Flag for above - 1 char
INTEGER, PARAMETER :: SnowCover = 0      ! Snow cover (0 or 1) - 1 digit
CHARACTER(LEN=1), PARAMETER :: SnowCover_F = "9" ! Flag for above - 1 char

CONTAINS

!-----
--

SUBROUTINE SetNewWyec2(Filename, ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetNewWyec2" :: SetNewWyec2

  ! PURPOSE:      Initializes the WYEC2 data structure and opens a file for
  !               output.
  !
  ! CREATED:
  !   5.13.99      S.J.Rees
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   Filename      -   see arguments
  !
  ! SUBROUTINE ARGUMENTS:
  !   Filename      -   user defined name for creation of wyec2 format
file
  !   ErrFlag       -   for global error checking
  !   ErrMess       -   for error reporting

```

```

IMPLICIT NONE

! arguments
CHARACTER*(*) , INTENT(IN) :: Filename
CHARACTER*(*) , INTENT(OUT) :: ErrMess
INTEGER(2) , INTENT(OUT) :: ErrFlag

!local variables
INTEGER :: error, i, j

ErrFlag = 0
ErrMess = ' '

! open file for output
OPEN (Wyec2out, file=Filename, status='unknown', IOSTAT=error)

IF(error/=0) THEN
    ErrMess = "Error: WYEC2 output file could not be opened"
    ErrFlag = 1
    RETURN
END IF

!initialize month, day and hour arrays
DO i=1, N_Hours
    j = (i-1)/24 + 1 ! day no (1-365)
    Hour(i) = i - (j-1)*24 ! derive hour no (1-24)
    ! select month based on day no.
    SELECT CASE (j)
        CASE(1:31) ! jan
            Month(i) = 1
            Day(i) = j
        CASE(32:59) ! feb
            Month(i) = 2
            Day(i) = j - 31
        CASE(60:90) ! march
            Month(i) = 3
            Day(i) = j - 59
        CASE(91:120) ! april
            Month(i) = 4
            Day(i) = j - 90
        CASE(121:151) ! may
            Month(i) = 5
            Day(i) = j - 120
        CASE(152:181) ! june
            Month(i) = 6
            Day(i) = j - 151
        CASE(182:212) ! july
            Month(i) = 7
            Day(i) = j - 181
        CASE(213:243) ! aug
            Month(i) = 8
            Day(i) = j - 212
        CASE(244:273) ! sept
            Month(i) = 9
            Day(i) = j - 243
        CASE(274:304) ! oct
            Month(i) = 10
    
```

```

        Day(i) = j - 273
        CASE(305:334)      ! nov
        Month(i) = 11
        Day(i) = j - 304
        CASE(335:365)      ! dec
        Month(i) = 12
        Day(i) = j - 334
    END SELECT
END DO

! specifically initialize other arrays
DryBulb = 0
DewPoint = 0
Pressure = 10000
DirNormIrr = 0
Relhumid = 100

! set temp flags to "9" indicating "missing" by default
DryBulb_F = "9"
DewPoint_F = "9"
Pressure_F = "9"
DirNormIrr_F = "9"
Relhumid_F = "9"

RETURN

END    SUBROUTINE SetNewWyec2

!-----
--

SUBROUTINE SetWyec2DryBulb (RealDryBulb, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWyec2DryBulb" :: SetWyec2DryBulb

    ! PURPOSE:      Used to set the drybulb temperatures. REAL dry bulb temp
array    !          is converted to integer array for file format.
    !
    ! CREATED:
    !   5.13.99      S.J.Rees
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealDryBulb   -   Array of real dry bulb temperatures (each hour of
year)    !           N_hrs       -   no. of hours (for VB dll calls)
    !           ErrFlag    -   for global error checking
    !           ErrMess    -   for error reporting

    IMPLICIT NONE

```

```

    INTEGER, INTENT(IN) :: N_hrs
    REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDryBulb
    CHARACTER*(*), INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag
    INTEGER i,error
    ErrFlag = 0
    ErrMess = ' '

    IF(SIZE(RealDryBulb) /= N_Hours) THEN
        ErrMess = "Error: Drybulb array inconsistant size"
        ErrFlag = 1
        RETURN
    END IF

    DryBulb = 10.0 * RealDryBulb
    ! set flag
    DryBulb_F = " "

END SUBROUTINE SetWyec2DryBulb

!-----
--

SUBROUTINE SetWyec2DewPoint (RealDewPoint, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWyec2DewPoint" :: SetWyec2DewPoint

    ! PURPOSE:      Used to set the dew point temperatures. REAL dew point
temp array
    !              is converted to integer array for file format.
    !
    ! CREATED:
    !   5.18.99      S.J.Rees
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealDewPoint - Array of real dew point temperatures (each hour
of year)
    !   N_hrs        - no. of hours (for VB dll calls)
    !   ErrFlag      - for global error checking
    !   ErrMess      - for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N_hrs
    REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDewPoint
    CHARACTER*(*), INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag
    INTEGER i,error

```

```

    ErrFlag = 0
    ErrMess = ' '

    IF(SIZE(RealDewPoint) /= N_Hours) THEN
        ErrMess = "Error: dew point array inconsistant size"
        ErrFlag = 1
        RETURN
    END IF

    DewPoint = 10.0 * RealDewPoint
    ! set flag
    DewPoint_F = " "

END SUBROUTINE SetWyec2DewPoint

!-----
--

SUBROUTINE SetWyec2Temps (RealDryBulb, N1_hrs, RealDewPoint, N2_hrs,
ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWyec2Temps" :: SetWyec2Temps

    ! PURPOSE:      Used to set both the dry bulb and dew point temperatures.
REAL
    !              arrays are converted to integer array for file format.
    !
    ! CREATED:
    !   5.18.99      S.J.Rees
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealDryBulb   -   Array of real dry bulb temperatures (each hour of
year)
    !   N1_hrs        -   no. of hours (for VB dll calls)
    !   RealDewPoint  -   Array of real dew point temperatures (each hour
of year)
    !   N2_hrs        -   no. of hours (for VB dll calls)
    !   ErrFlag       -   for global error checking
    !   ErrMess       -   for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N1_hrs, N2_hrs
    REAL(prec), DIMENSION(N1_hrs), INTENT(IN) :: RealDryBulb
    REAL(prec), DIMENSION(N2_hrs), INTENT(IN) :: RealDewPoint
    CHARACTER*(*) , INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag
    INTEGER i,error

```

```

    ErrFlag = 0
    ErrMess = ' '

    CALL SetWynec2DryBulb(RealDryBulb, N1_hrs, ErrFlag, ErrMess)
    IF(ErrFlag/=0) RETURN
    CALL SetWynec2DewPoint(RealDewPoint, N2_hrs, ErrFlag, ErrMess)
    IF(ErrFlag/=0) RETURN

END SUBROUTINE SetWynec2Temps

!-----
--

SUBROUTINE SetWynec2Pressure (RealPressure, N_hrs, ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWynec2Pressure" :: SetWynec2Pressure

  ! PURPOSE:      Used to set the Pressure. REAL pressure array
  !                is converted to integer array for file format.
  !
  ! CREATED:
  !   5.18.99      S.J.Rees
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   n/a
  !
  ! SUBROUTINE ARGUMENTS:
  !   RealPressure   -   Array of real pressures (Kpa each hour of year)
  !   N_hrs          -   no. of hours (for VB dll calls)
  !   ErrFlag        -   for global error checking
  !   ErrMess        -   for error reporting

  IMPLICIT NONE

  INTEGER, INTENT(IN) :: N_hrs
  REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealPressure
  CHARACTER*(*), INTENT(OUT) :: ErrMess
  INTEGER(2), INTENT(OUT) :: ErrFlag
  INTEGER i,error
  ErrFlag = 0
  ErrMess = ' '

  IF(SIZE(RealPressure) /= N_Hours) THEN
    ErrMess = "Error: Pressure array inconsistant size"
    ErrFlag = 1
    RETURN
  END IF

  Pressure = 100.0 * RealPressure
  ! set flag
  Pressure_F = " "

```



```

END SUBROUTINE SetWyec2Pressure

!-----
--

SUBROUTINE SetWyec2DirNormIrr (RealDirNormIrr, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWyec2DirNormIrr" ::
SetWyec2DirNormIrr

! PURPOSE:      Used to set the direct normal irradiance. REAL direct
normal
!               irradiance array is converted to integer array for file
format.
!
!   CREATED:
!   11.7.99      S.J.Rees, Dongyi Xiao
!
!   MODIFIED:
!
!   INPUT FILES:
!   n/a
!
!   OUTPUT FILES:
!   n/a
!
!   SUBROUTINE ARGUMENTS:
!   RealDirNormIrr - Array of real Direct normal irradiance (J/m2 each
hour of year)
!   N_hrs          - no. of hours (for VB dll calls)
!   ErrFlag        - for global error checking
!   ErrMess        - for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDirNormIrr
CHARACTER*(*) , INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag
INTEGER i,error
ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDirNormIrr) /= N_Hours) THEN
    ErrMess = "Error:DirNormIrr array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DirNormIrr = RealDirNormIrr/1000.0
! set flag
DirNormIrr_F = " "

END SUBROUTINE SetWyec2DirNormIrr

!-----
--

```

```

SUBROUTINE SetWyec2RelHumid (RealRelhumid, N_hrs, ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetWyec2RelHumid" :: SetWyec2RelHumid

  ! PURPOSE:      Used to set the relative humidities. REAL relative
humidity array  ! is converted to integer array for file format.
  !
  ! CREATED:
  !   5.13.99      S.J.Rees
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   n/a
  !
  ! SUBROUTINE ARGUMENTS:
  !   RealRelhumid - Array of real relative humidities (each hour of
year)            !
  !   N_hrs        - no. of hours (for VB dll calls)
  !   ErrFlag      - for global error checking
  !   ErrMess      - for error reporting

  IMPLICIT NONE

  INTEGER, INTENT(IN) :: N_hrs
  REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealRelhumid
  CHARACTER*(*), INTENT(OUT) :: ErrMess
  INTEGER(2), INTENT(OUT) :: ErrFlag
  INTEGER i,error
  ErrFlag = 0
  ErrMess = ' '

  IF(SIZE(RealRelhumid) /= N_Hours) THEN
    ErrMess = "Error: Relhumid array inconsistant size"
    ErrFlag = 1
    RETURN
  END IF

  Relhumid = 100.0 * RealRelhumid
  ! set flag
  Relhumid_F = " "

END SUBROUTINE SetWyec2RelHumid

!-----
--

SUBROUTINE WriteWyec2(ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "WriteWyec2" :: WriteWyec2

  ! PURPOSE:      Writes out the WYEC2 data structure. This is to be called
  !               after initialization and setting of required fields.

```

```

!
! CREATED:
!   5.13.99      S.J.Rees
!
! MODIFIED:
!
! INPUT FILES:
! n/a
!
! OUTPUT FILES:
!   Filename      -   as defined in 'SetNewWyec2'
!
! SUBROUTINE ARGUMENTS:
!   ErrFlag       -   for global error checking
!   ErrMess       -   for error reporting

IMPLICIT NONE

CHARACTER*(*), INTENT(OUT)      :: ErrMess
INTEGER(2), INTENT(OUT)        :: ErrFlag

INTEGER i,error
ErrFlag = 0
ErrMess = ' '

DO i=1, N_Hours

    WRITE(Wyec2out, 100, IOSTAT=error) WBAN, Source, Year, Month(i),
Day(i), Hour(i), &
                                ExtraIrr, GlobHorIrr, GlobHorIrr_F,
DirNormIrr(i), &
                                DirNormIrr_F, DiffHorIrr, DiffHorIrr_F,
GlobHorIll, &
                                GlobHorIll_F, DirNormIll, DirNormIll_F,
DiffHorIll, &
                                DiffHorIll_F, ZenithIll, ZenithIll_F, MinSun, &
                                MinSun_F, CeilHeight, CeilHeight_F, SkyCond, &
                                SkyCond_F, Vis, Vis_F, Weather, Weather_F, &
                                Pressure(i), Pressure_F, DryBulb(i), DryBulb_F,
DewPoint(i), &
                                DewPoint_F, WindDir, WindDir_F, WindSpeed,
WindSpeed_F, &
                                TotSky, TotSky_F, OpaqSky, OpaqSky_F, SnowCover,
&
                                SnowCover_F

    IF(error/=0) THEN
        ErrMess = "Error: WYEC2 file data could not be written"
        ErrFlag = 1
        RETURN
    END IF

END DO

CLOSE(Wyec2out)

100 FORMAT(I5.5, A1, 4I2.2, &

```

```
      2I4, A2, I4, &  
      A2, I4, A2, I4, &  
      A1, I4, A1, I4, &  
      A1, I4, A1, I2.2, &  
      A1, I4.4, A1, I4.4, &  
      A1, I4.4, A1, I8.8, A1, &  
      I5, A1, I4, A1, I4, &  
      A1, I3, A1, I4, A1, &  
      I2, A1, I2, A1, I1, &  
      A1)  
  
      END SUBROUTINE WriteWyec2  
  
END MODULE Wyec2Util
```

Source Code: Module IwecUtil

```

MODULE IwecUtil

!-----
--
!
! PROJECT:      Development of an Analytical Verification Test Suite for
!               Whole Building Simulation Programs - Building Fabric (1052-
RP)
!
! CONTRACTOR:   Oklahoma State University,
!               School of Mechanical and Aerospace Engineering.
!
! COPYRIGHT:    The American Society of Heating, Refrigeration and Air
!               Conditioning Engineers Inc.
!
!-----
--
!
!
! PURPOSE:      Contains data structure definition for dealing with Iwec
!               weather files and routines for manipulating certain fields
in
!               the data.
!
! CREATED:
!   10.20.2000   Dongyi Xiao
!
! MODIFIED:
!
!-----
--
IMPLICIT NONE
SAVE
!PRIVATE ! default is for everything to be private

! set specific procedures public
PUBLIC :: SetNewIwec
PUBLIC :: SetIwecHeader

PUBLIC :: SetIwecDryBulb
PUBLIC :: SetIwecRelHumid

PUBLIC :: SetIwecDirNormIrr
PUBLIC :: SetIwecDewPoint
PUBLIC :: SetIwecPressure
PUBLIC :: SetIwecTemps
PUBLIC :: WriteIwec

! general prameters
INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=6)
INTEGER, PARAMETER :: Iwecout=20           ! unit number for user output file

```

```

INTEGER, PARAMETER :: N_Hours = 8760      ! Number of hours in year

!----- Iwec data structure -----
--
! variable elements
INTEGER, DIMENSION(N_Hours) :: DryBulb    ! Int value of Drybulb*10 (C)
INTEGER, DIMENSION(N_Hours) :: DewPoint   ! Dew point temp*10 (C)
INTEGER, DIMENSION(N_Hours) :: RelHumid   ! Relative Humidity (%)
INTEGER, DIMENSION(N_Hours) :: Pressure   ! pressure (Millibars)
INTEGER, DIMENSION(N_Hours) :: DirNormIrr ! Direct normal irradiance (Wh/m2)

INTEGER, DIMENSION(N_Hours) :: Month      ! Month of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Day        ! Day of record - 2 digit
INTEGER, DIMENSION(N_Hours) :: Hour       ! Hour of record - 2 digit

CHARACTER(LEN=22) :: City = "ASHRAE Analytical test" ! City name - 22 char
CHARACTER(LEN=2)  :: State = "OK" ! State - 2 char

CHARACTER(LEN=1)  :: TimeZone_F = "-" ! Time zone sign(relative to Universal
Time) - 1 digit
                                ! + if ahead of or equal to UT, - if
                                ! behind
INTEGER           :: TimeZone_hour = 06 ! Time zone - hours component - 2
digit
INTEGER           :: TimeZone_min = 00  ! Time zone - Minutes component -
2 digit

CHARACTER(LEN=1)  :: Lat_F="N" ! Latitude - N/S or 0 flag
INTEGER           :: Lat_Deg= 0    ! Latitude - Degrees component
INTEGER           :: Lat_Min= 0    ! Latitude - Minutes component
CHARACTER(LEN=1)  :: Long_F="W" ! Longitude - W/E or 0 flag
INTEGER           :: Long_Deg = 0   ! Longitude - Degrees component
INTEGER           :: Long_Min = 0   ! Longitude - Minutes component
CHARACTER(LEN=1)  :: Elevation_F = "+" ! Elevation sign - 1 char
                                ! + for above or at S.L., - for below
S.L.
INTEGER           :: Elevation = 0   ! Elevation (m) - 4 digit

! parameterized fields in order
! Iwec header fields
CHARACTER(LEN=4), PARAMETER :: FormDes="Iwec" ! the four characters "Iwec"
INTEGER, PARAMETER :: WMO_num = 999999 ! weather station WMO number -
6 digit
CHARACTER(LEN=3), PARAMETER :: Country="USA" ! Country - 3 char
INTEGER, PARAMETER :: WMO_regnum = 4 ! WMO region number - 1 digit
between 1 and 7

! Main data field parameters
INTEGER, PARAMETER :: Year = 1999 ! year of records - 4
digit(1982-1993) ?

! solar irradiance data - note units are (Wh/m2)
INTEGER, PARAMETER :: ExtraHorIrr = 0 ! Extraterrestrial Horizontal
irradiance - 4 digit

```

```

    INTEGER, PARAMETER :: ExtraNormIrr = 0 ! Extraterrestrial direct normal
irradiance - 4 digit
    INTEGER, PARAMETER :: GlobHorIrr = 0 ! Global Horizontal irradiance - 4
digit
    CHARACTER(LEN=2), PARAMETER :: GlobHorIrr_F = "?0" ! Flag for above - 2 char

! INTEGER, PARAMETER :: DirNormIrr = 0 ! Direct normal irradiance - 4
digit
! CHARACTER(LEN=2), PARAMETER :: DirNormIrr_F = "?0" ! Flag for above - 2
char
    INTEGER, PARAMETER :: DiffHorIrr = 0 ! Diffuse Horizontal irradiance -
4 digit
    CHARACTER(LEN=2), PARAMETER :: DiffHorIrr_F = "?0" ! Flag for above - 2 char

! Illuminance data (lux/100)
    INTEGER, PARAMETER :: GlobHorIll = 0 ! Global Horizontal illuminance -
4 digit
    CHARACTER(LEN=2), PARAMETER :: GlobHorIll_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: DirNormIll = 0 ! Direct normal illuminance - 4
digit
    CHARACTER(LEN=2), PARAMETER :: DirNormIll_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: DiffHorIll = 0 ! Diffuse Horizontal illuminance -
4 digit
    CHARACTER(LEN=2), PARAMETER :: DiffHorIll_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: ZenithIll = 0 ! Zenith illuminance - 4 digit
(Cd/m2/10)
    CHARACTER(LEN=2), PARAMETER :: ZenithIll_F = "?0" ! Flag for above - 2 char

! sky conditions
    INTEGER, PARAMETER :: TotSky = 0 ! Total sky cover (tenths) - 2
digit
    CHARACTER(LEN=2), PARAMETER :: TotSky_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: OpaqSky = 0 ! Opaque sky cover (tenths) - 2
digit
    CHARACTER(LEN=2), PARAMETER :: OpaqSky_F = "?0" ! Flag for above - 2 char

! air conditions
! dry bulb is variable
    CHARACTER(LEN=2) :: DryBulb_F ! Flag for DryBulb - 2 char
! dew point is variable
    CHARACTER(LEN=2) :: DewPoint_F ! Flag for dew point - 2 char
! relative humidity is variable
    CHARACTER(LEN=2) :: RelHumid_F ! Flag for relative humidity - 2
char
! pressure is variable
    CHARACTER(LEN=2) :: Pressure_F ! Flag for pressure - 2 char
! DirNormIrr is variable
    CHARACTER(LEN=2) :: DirNormIrr_F ! Flag for DirNormIrr - 2 char
! wind conditions
    INTEGER, PARAMETER :: WindDir = 0 ! Wind direction (0-360) N=0,
E=90,
! S=180, W=270, calm=0 - 3 digit
    CHARACTER(LEN=2), PARAMETER :: WindDir_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: WindSpeed = 0 ! Wind speed (m/s*10) - 3 digit
    CHARACTER(LEN=2), PARAMETER :: WindSpeed_F = "?0" ! Flag for above - 2 char

    INTEGER, PARAMETER :: Vis = 9999 ! Visibility - 4 digit (km*10)

```

```

    CHARACTER(LEN=2), PARAMETER :: Vis_F = "?0"! Flag for above - 2 char
    INTEGER, PARAMETER :: CeilHeight = 99999 ! ceiling height, m - 5 digit
    CHARACTER(LEN=2), PARAMETER :: CeilHeight_F = "?0" ! Flag for above - 2 char

    ! sky conditions
    INTEGER, PARAMETER :: SkyLayer = 9999 ! sky condition by layer - 4
digit
    CHARACTER(LEN=2), PARAMETER :: SkyLayer_F = "?0"! Flag for above - 2 char

    ! other useful stuff
    CHARACTER(LEN=10), PARAMETER :: Weather = "9999999999" ! weather
discription - 10 digit
    INTEGER, PARAMETER :: Precip = 0 ! Precipitation (mm) at the hour - 3
digit
    CHARACTER(LEN=2), PARAMETER :: Precip_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: OptDepth = 0 ! Broadband Aerosol Optical depth,
thousandths - 3 digit
    CHARACTER(LEN=2), PARAMETER :: OptDepth_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: SnowDepth = 0 ! Snow depth (cm) on day
indicated - 3 digit
    CHARACTER(LEN=2), PARAMETER :: SnowDepth_F = "?0" ! Flag for above - 2 char
    INTEGER, PARAMETER :: DaysLastSnow = 88 ! Days since last snowfall (0 -
88) - 2 digit
    CHARACTER(LEN=2), PARAMETER :: DaysLastSnow_F = "?0" ! Flag for above - 2
char

CONTAINS

!-----
--

SUBROUTINE SetNewIwec(Filename, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetNewIwec" :: SetNewIwec

    ! PURPOSE:      Initializes the Iwec data structure and opens a file for
    !               output.
    !
    ! CREATED:
    !   10.25.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   Filename        -   see arguments
    !
    ! SUBROUTINE ARGUMENTS:
    !   Filename        -   user defined name for creation of Iwec format file
    !   ErrFlag          -   for global error checking
    !   ErrMess          -   for error reporting

    IMPLICIT NONE

    ! arguments

```



```

CHARACTER*(*), INTENT(IN)      :: Filename
CHARACTER*(*), INTENT(OUT)     :: ErrMess
INTEGER(2), INTENT(OUT)        :: ErrFlag

!local variables
INTEGER :: i, j, error

ErrFlag = 0
ErrMess = ' '

! open file for output
OPEN (Iwecout, file=Filename, status='unknown', IOSTAT=error)
IF(error/=0) THEN
    ErrMess = "IWEC output file could not be opened"
    ErrFlag = 1
    RETURN
END IF

!initialize month, day and hour arrays
DO i=1, N_Hours
    j = (i-1)/24 + 1 ! day no (1-365)
    Hour(i) = i - (j-1)*24 ! derive hour no (1-24)
    ! select month based on day no.
    SELECT CASE (j)
        CASE(1:31)      ! jan
            Month(i) = 1
            Day(i) = j
        CASE(32:59)     ! feb
            Month(i) = 2
            Day(i) = j - 31
        CASE(60:90)     ! march
            Month(i) = 3
            Day(i) = j - 59
        CASE(91:120)    ! april
            Month(i) = 4
            Day(i) = j - 90
        CASE(121:151)   ! may
            Month(i) = 5
            Day(i) = j - 120
        CASE(152:181)   ! june
            Month(i) = 6
            Day(i) = j - 151
        CASE(182:212)   ! july
            Month(i) = 7
            Day(i) = j - 181
        CASE(213:243)   ! aug
            Month(i) = 8
            Day(i) = j - 212
        CASE(244:273)   ! sept
            Month(i) = 9
            Day(i) = j - 243
        CASE(274:304)   ! oct
            Month(i) = 10
            Day(i) = j - 273
        CASE(305:334)   ! nov
            Month(i) = 11
            Day(i) = j - 304
    END SELECT
END DO

```

```

        CASE(335:365)      ! dec
            Month(i) = 12
            Day(i) = j - 334
        END SELECT
    END DO

    ! specifically initialize other arrays
    DryBulb = 0
    RelHumid= 100
    DewPoint = 0
    Pressure = 1000
    DirNormIrr= 0
    ! set temp flags to "?0" indicating "missing" by default
    DryBulb_F = "?0"
    DewPoint_F = "?0"
    Pressure_F = "?0"
    RelHumid_F = "?0"
    DirNormIrr_F= "?0"

    RETURN

END    SUBROUTINE SetNewIwec

!-----
--
SUBROUTINE SetIwecHeader (Location, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecHeader" :: SetIwecHeader

    ! PURPOSE:      Used to set the header fields. REAL header fields
    !               is set differently according to the location.
    !
    ! CREATED:
    !   25.10.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   Location      -   index of location
    !   ErrFlag       -   for global error checking
    !   ErrMess       -   for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: Location
    CHARACTER*(*) , INTENT(OUT)      :: ErrMess
    INTEGER(2), INTENT(OUT)          :: ErrFlag
    INTEGER i,error

    ErrFlag = 0
    ErrMess = ' '

```

```

! IWEC header fields
IF (Location==0) THEN

    City="Atlanta" ! City name - 22 char
    State="GA" ! State
    TimeZone_F = "-" ! Time zone sign(relative to Universal Time) - 1 digit
                        ! + if ahead of or equal to UT, - if behind
    TimeZone_hour = 5 ! Time zone - hours component - 2 digit
    TimeZone_min = 0 ! Time zone - Minutes component - 2 digit

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 33 ! Latitude - Degrees component
    Lat_Min = 46 ! Latitude - Minutes component
    Long_F = "W" ! Longitude - W/E flag
    Long_Deg = 84 ! Longitude - Degrees component
    Long_Min = 25 ! Longitude - Minutes component
    Elevation_F = "+" ! Elevation sign - 1 char
                        ! + for above or at S.L., - for below S.L.
    Elevation = 306 ! Elevation A.S.L. (m)
ELSE IF (Location==1) THEN
    City="Boston" ! City name - 22 char
    State="MA" ! State
    TimeZone_F = "-" ! Time zone sign(relative to Universal Time) - 1 digit
                        ! + if ahead of or equal to UT, - if behind
    TimeZone_hour = 5 ! Time zone - hours component - 2 digit
    TimeZone_min = 0 ! Time zone - Minutes component - 2 digit

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 42 ! Latitude - Degrees component
    Lat_Min = 19 ! Latitude - Minutes component
    Long_F = "W" ! Longitude - W/E flag
    Long_Deg = 71 ! Longitude - Degrees component
    Long_Min = 5 ! Longitude - Minutes component
    Elevation_F = "+" ! Elevation sign - 1 char
                        ! + for above or at S.L., - for below S.L.
    Elevation = 5 ! Elevation A.S.L. (m)
ELSE IF (Location==2) THEN
    City="Chicago" ! City name - 22 char
    State="IL" ! State
    TimeZone_F = "-" ! Time zone sign(relative to Universal Time) - 1 digit
                        ! + if ahead of or equal to UT, - if behind
    TimeZone_hour = 6 ! Time zone - hours component - 2 digit
    TimeZone_min = 0 ! Time zone - Minutes component - 2 digit

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 41 ! Latitude - Degrees component
    Lat_Min = 51 ! Latitude - Minutes component
    Long_F = "W" ! Longitude - W/E flag
    Long_Deg = 87 ! Longitude - Degrees component
    Long_Min = 41 ! Longitude - Minutes component
    Elevation_F = "+" ! Elevation sign - 1 char
                        ! + for above or at S.L., - for below S.L.
    Elevation = 186 ! Elevation A.S.L. (m)
ELSE IF (Location==3) THEN
    City="Los Angeles" ! City name - 22 char
    State="CA" ! State

```

```

    TimeZone_F = "-"      ! Time zone sign(relative to Universal Time) - 1 digit
                        ! + if ahead of or equal to UT, - if behind
    TimeZone_hour = 8      ! Time zone - hours component - 2 digit
    TimeZone_min = 0       ! Time zone - Minutes component - 2 digit

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 34      ! Latitude - Degrees component
    Lat_Min = 5       ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 118     ! Longitude - Degrees component
    Long_Min = 22      ! Longitude - Minutes component
    Elevation_F = "+" ! Elevation sign - 1 char
                    ! + for above or at S.L., - for below S.L.
    Elevation = 30     ! Elevation A.S.L. (m)
ELSE
    City="ASHRAE Analytical test" ! City name - 22 char
    State="OK" ! State
    TimeZone_F = "+"      ! Time zone sign(relative to Universal Time) - 1 digit
                        ! + if ahead of or equal to UT, - if behind
    TimeZone_hour = 0      ! Time zone - hours component - 2 digit
    TimeZone_min = 0       ! Time zone - Minutes component - 2 digit

    Lat_F = "N" ! Latitude - N/S flag
    Lat_Deg = 0      ! Latitude - Degrees component
    Lat_Min = 0       ! Latitude - Minutes component
    Long_F = "W"! Longitude - W/E flag
    Long_Deg = 0      ! Longitude - Degrees component
    Long_Min = 0       ! Longitude - Minutes component
    Elevation_F = "+" ! Elevation sign - 1 char
                    ! + for above or at S.L., - for below S.L.
    Elevation = 0     ! Elevation A.S.L. (m)
ENDIF

END SUBROUTINE SetIwecHeader

!-----
--

SUBROUTINE SetIwecDryBulb (RealDryBulb, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecDryBulb" :: SetIwecDryBulb

    ! PURPOSE:      Used to set the drybulb temperatures. REAL dry bulb temp
array
    !               is converted to integer array for file format.
    !
    ! CREATED:
    !   25.10.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:

```

```

!   RealDryBulb      -   Array of real dry bulb temperatures
!   N_hrs            -   no. of hours (for VB dll calls)
!   ErrFlag          -   for global error checking
!   ErrMess          -   for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDryBulb
CHARACTER*(*), INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag
INTEGER i,error

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDryBulb) /= N_Hours) THEN
    ErrMess = "Drybulb array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DryBulb = 10.0 * RealDryBulb
! set flag
!DryBulb_F = "B7"

END SUBROUTINE SetIwecDryBulb

!-----
--

SUBROUTINE SetIwecDewPoint (RealDewPoint, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecDewPoint" :: SetIwecDewPoint

! PURPOSE:      Used to set the dew point temperatures. REAL dew point
temp array      is converted to integer array for file format.
!
!
! CREATED:
!   25.10.2000      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   RealDewPoint    -   Array of real dew point temperatures (each hour of
year)
!   N_hrs           -   no. of hours (for VB dll calls)
!   ErrFlag         -   for global error checking
!   ErrMess         -   for error reporting

```

```

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDewPoint
INTEGER i,error
CHARACTER*(*), INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealDewPoint) /= N_Hours) THEN
    ErrMess = "Error: dew point array inconsistant size"
    ErrFlag = 1
    RETURN
END IF

DewPoint = 10.0 * RealDewPoint
! set flag
!DewPoint_F = "E8"

END SUBROUTINE SetIwecDewPoint

!-----
--

SUBROUTINE SetIwecTemps (RealDryBulb, N1_hrs, RealDewPoint, N2_hrs, ErrFlag,
ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecTemps" :: SetIwecTemps

    ! PURPOSE:      Used to set both the dry bulb and dew point temperatures.
REAL
    !              arrays are converted to integer array for file format.
    !
    ! CREATED:
    !   25.10.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealDryBulb      -   Array of real dry bulb temperatures (each hour of
year)
    !   N1_hrs           -   no. of hours (for VB dll calls)
    !   RealDewPoint     -   Array of real dew point temperatures (each hour
of year)
    !   N2_hrs           -   no. of hours (for VB dll calls)
    !   ErrFlag          -   for global error checking
    !   ErrMess          -   for error reporting

IMPLICIT NONE

```

```

    INTEGER, INTENT(IN) :: N1_hrs, N2_hrs
    REAL(prec), DIMENSION(N1_hrs), INTENT(IN) :: RealDryBulb
    REAL(prec), DIMENSION(N2_hrs), INTENT(IN) :: RealDewPoint
    CHARACTER*(*), INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag
    INTEGER i,error

    ErrFlag = 0
    ErrMess = ' '

    CALL SetIwecDryBulb(RealDryBulb, N1_hrs, ErrFlag, ErrMess)
    IF(ErrFlag/=0) RETURN
    CALL SetIwecDewPoint(RealDewPoint, N2_hrs, ErrFlag, ErrMess)
    IF(ErrFlag/=0) RETURN

END SUBROUTINE SetIwecTemps

!-----
--

SUBROUTINE SetIwecPressure (RealPressure, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecPressure" :: SetIwecPressure

    ! PURPOSE:      Used to set the Pressure. REAL pressure array
    !                is converted to integer array for file format.
    !
    ! CREATED:
    !   25.10.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealPressure    -   Array of real pressures (Kpa each hour of year)
    !   N_hrs           -   no. of hours (for VB dll calls)
    !   ErrFlag         -   for global error checking
    !   ErrMess         -   for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N_hrs
    REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealPressure
    CHARACTER*(*), INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag

    ErrFlag = 0
    ErrMess = ' '

    IF(SIZE(RealPressure) /= N_Hours) THEN
        ErrMess = "Error: Pressure array inconsistant size"
        ErrFlag = 1
    
```

```

        RETURN
    END IF

    Pressure = 10.0 * RealPressure
    ! set flag
    !Pressure_F = "A7"

END SUBROUTINE SetIwecPressure

!-----
--

SUBROUTINE SetIwecDirNormIrr (RealDirNormIrr, N_hrs, ErrFlag, ErrMess)
    !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecDirNormIrr" ::
    SetIwecDirNormIrr

    ! PURPOSE:      Used to set the Direct normal irradiance. REAL Direct
normal
    !                irradiance array is converted to integer array for file
format.
    !
    ! CREATED:
    !   25.10.2000      Dongyi Xiao
    !
    ! MODIFIED:
    !
    ! INPUT FILES:
    !   n/a
    !
    ! OUTPUT FILES:
    !   n/a
    !
    ! SUBROUTINE ARGUMENTS:
    !   RealDirNormIrr - Array of real Direct normal irradiance (J/m2 each
hour of year)
    !   N_hrs          - no. of hours (for VB dll calls)
    !   ErrFlag        - for global error checking
    !   ErrMess        - for error reporting

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N_hrs
    REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealDirNormIrr
    INTEGER i,error
    CHARACTER*(*), INTENT(OUT) :: ErrMess
    INTEGER(2), INTENT(OUT) :: ErrFlag

    ErrFlag = 0
    ErrMess = ' '

    IF(SIZE(RealDirNormIrr) /= N_Hours) THEN
        ErrMess = "Error: dew point array inconsistant size"
        ErrFlag = 1
        RETURN
    END IF

```



```

DirNormIrr = RealDirNormIrr/3600.0
! set flag
!DewPoint_F = "E8"

END SUBROUTINE SetIwecDirNormIrr

!-----
--

SUBROUTINE SetIwecRelHumid (RealRelHumid, N_hrs, ErrFlag, ErrMess)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "SetIwecRelHumid" :: SetIwecRelHumid

! PURPOSE:      Used to set the relative humidities. REAL relative
humidity array
!               is converted to integer array for file format.
!
! CREATED:
!   25.10.2000      Dongyi Xiao
!
! MODIFIED:
!
! INPUT FILES:
!   n/a
!
! OUTPUT FILES:
!   n/a
!
! SUBROUTINE ARGUMENTS:
!   RealRelHumid    -   Array of real relative humidities
!   N_hrs           -   no. of hours (for VB dll calls)
!   ErrFlag         -   for global error checking
!   ErrMess         -   for error reporting

IMPLICIT NONE

INTEGER, INTENT(IN) :: N_hrs
REAL(prec), DIMENSION(N_hrs), INTENT(IN) :: RealRelHumid
CHARACTER*(*), INTENT(OUT) :: ErrMess
INTEGER(2), INTENT(OUT) :: ErrFlag
INTEGER i,error

ErrFlag = 0
ErrMess = ' '

IF(SIZE(RealRelHumid) /= N_Hours) THEN
  ErrMess = "RealRelHumid array inconsistant size"
  ErrFlag = 1
  RETURN
END IF

RelHumid = 100.0 * RealRelHumid
! set flag
!DryBulb_F = "B7"

END SUBROUTINE SetIwecRelHumid

```

```

!-----
--

SUBROUTINE WriteIwec(ErrFlag, ErrMess)
  !DEC$ ATTRIBUTES DLLEXPORT, ALIAS : "WriteIwec" :: WriteIwec

  ! PURPOSE:      Writes out the Iwec data structure. This is to be called
  !               after initialization and setting of required fields.
  !
  ! CREATED:
  !   25.10.2000      Dongyi Xiao
  !
  ! MODIFIED:
  !
  ! INPUT FILES:
  !   n/a
  !
  ! OUTPUT FILES:
  !   Filename        -   as defined in 'SetNewIwec'
  !
  ! SUBROUTINE ARGUMENTS:
  !
  !   ErrFlag          -   for global error checking
  !   ErrMess          -   for error reporting

  IMPLICIT NONE

  CHARACTER*(*) , INTENT(OUT)      :: ErrMess
  INTEGER(2) , INTENT(OUT)         :: ErrFlag
  INTEGER i,error

  ErrFlag = 0
  ErrMess = ' '

  ! write header data
  !WRITE(Iwecout,90, IOSTAT=error) WBAN, City, State, TimeZone, Lat_F
,Lat_Deg, Lat_Min, &
  !               Long_F, Long_Deg, Long_Min, Elevation
  WRITE(Iwecout,90, IOSTAT=error) FormDes, WMO_num, City, State, Country,
WMO_regnum, &
  !               TimeZone_F, TimeZone_hour, TimeZone_min, Lat_F, Lat_Deg,
Lat_Min, &
  !               Long_F, Long_Deg, Long_Min, Elevation_F, Elevation

  IF(error/=0) THEN
    ErrMess = "Error: Iwec file header could not be written"
    ErrFlag = 1
    RETURN
  END IF

  ! write out hourly data
  DO i=1, N_Hours

    WRITE(Iwecout, 100, IOSTAT=error) Year, Month(i), Day(i), Hour(i),
ExtraHorIrr, &

```

```

DirNormIrr(i), &
GlobHorIll, &
DiffHorIll, &
&
&
WindDir_F, &
CeilHeight_F, &
OptDepth, &
&
ExtraNormIrr, GlobHorIrr, GlobHorIrr_F,
DirNormIrr_F, DiffHorIrr, DiffHorIrr_F,
GlobHorIll_F, DirNormIll, DirNormIll_F,
DiffHorIll_F, ZenithIll, ZenithIll_F, &
TotSky, TotSky_F, OpaqSky, OpaqSky_F, DryBulb(i),
DryBulb_F, DewPoint(i), DewPoint_F, RelHumid(i),
RelHumid_F, Pressure(i), Pressure_F, WindDir,
WindSpeed, WindSpeed_F, Vis, Vis_F, CeilHeight,
SkyLayer, SkyLayer_F, Weather, Precip, Precip_F,
OptDepth_F, SnowDepth, SnowDepth_F, DaysLastSnow,
DaysLastSnow_F

    IF(error/=0) THEN
        ErrMess = "Error: Iwec file data could not be written"
        ErrFlag = 1
        RETURN
    END IF

END DO

! close weather file
CLOSE(Iwecout)

! header and main data formats - adapted from TMY2 manual
!90  FORMAT( 1X,I5.5,1X,A22,1X,A2,1X
I3,1X,A1,1X,I2,1X,I2,1X,A1,1X,I3,1X,I2,2X,I4 )
90  FORMAT( 1X,A4,1X,I6,1X,A22,1X
A2,1X,A3,1X,I1,1X,A1,1X,I2,1X,I2,1X,A1,1X,I2,1X,&
        I2,1X,A1,1X,I3,1X,I2,1X,A1,1X,I4 )

100 FORMAT(1X, I4.4, 3I2.2, 2I4.4, 7(I4.4, A2), 2(I2.2, A2), 2(I4.3, A2),
1(I3.3, A2), &
        1(I4.4, A2), 2(I3.3, A2), 1(I4.4, A2), 1(I5.5, A2), 1(I4.4, A2),
A10, &
        3(I3.3, A2), 1(I2.2, A2))

!100 FORMAT(1X, 4I2.2, 2I4.4, 7(I4.4, A2), 2(I2.2, A2), 2(I4.4, A2), 1(I3.3,
A2), &
!        1(I4.4, A2), 2(I3.3, A2), 1(I4.4, A2), 1(I5.5, A2), I10.10, 3(I3.3,
A2), 1(I2.2, A2))

END SUBROUTINE WriteIwec

END MODULE IwecUtil

```