

Thermal Modeling of Buildings

- This lecture deals with the model of a space heating system of a building by means of a passive solar system.
- The system is designed after a solar experimental building constructed in Tucson near the airport.
- The model is quite sophisticated. It models not only the physics of radiation through glassed windows, but also the weather patterns of Tucson.



Table of Contents

- Passive solar space heating
- Bond graph of a room
- Floor, windows, and walls
- The Dymola model
- Simulation results



Passive Solar Space Heating I



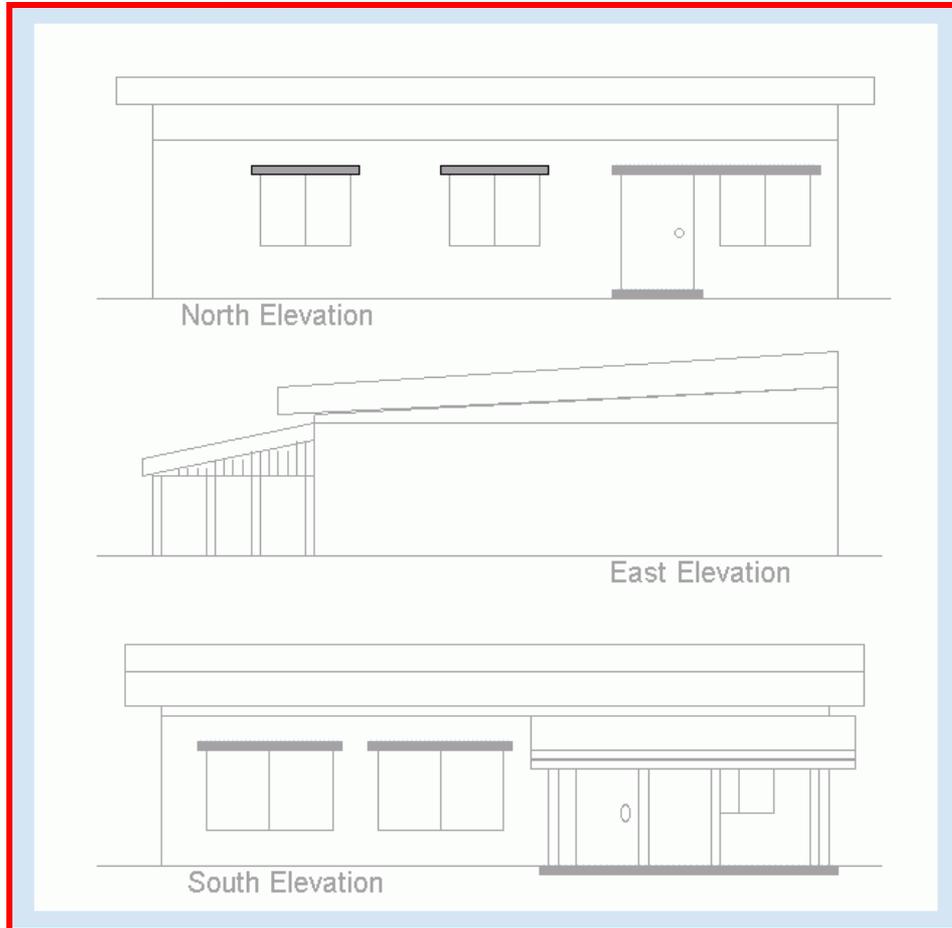
Southside view with (dismantled) sunspace.

Northside view.



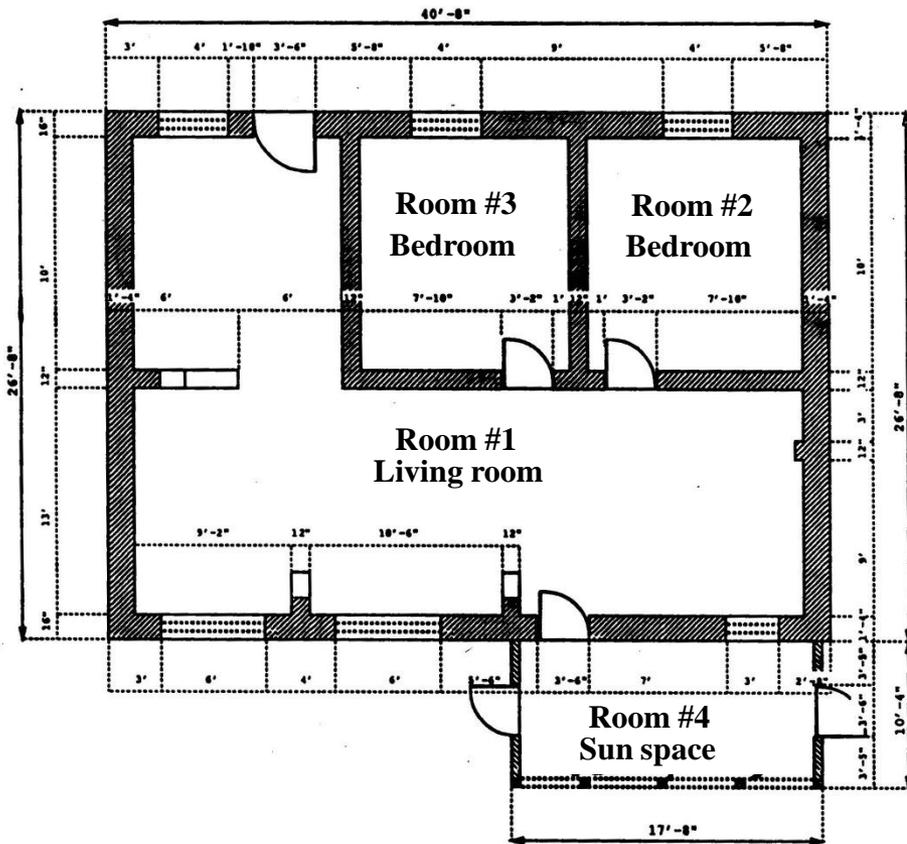
The house is constructed from Adobe brick. The photographs are rather recent. By the time they were taken, the house was no longer being used and had fallen a bit in disarray.

Passive Solar Space Heating II



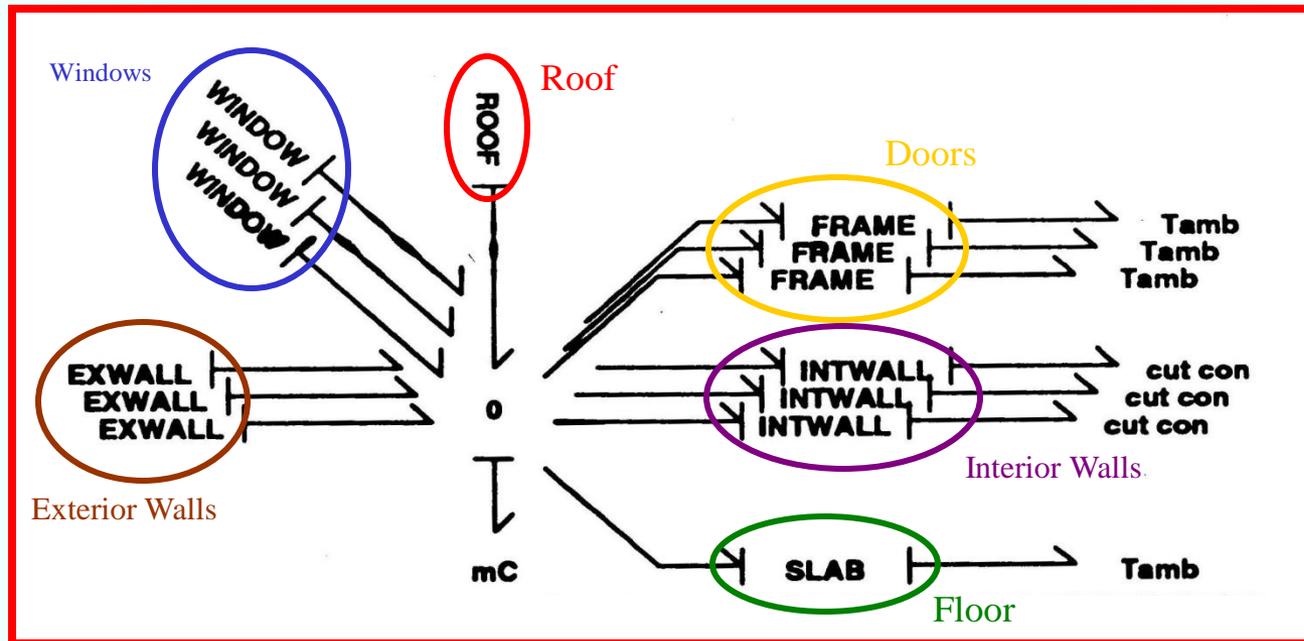
- The experimental solar building is shown here from three sides.
- Solar radiation through the walls, the windows, and the ceiling is to be modeled.
- Losses are also being modeled, including the losses through the slab.

Passive Solar Space Heating III



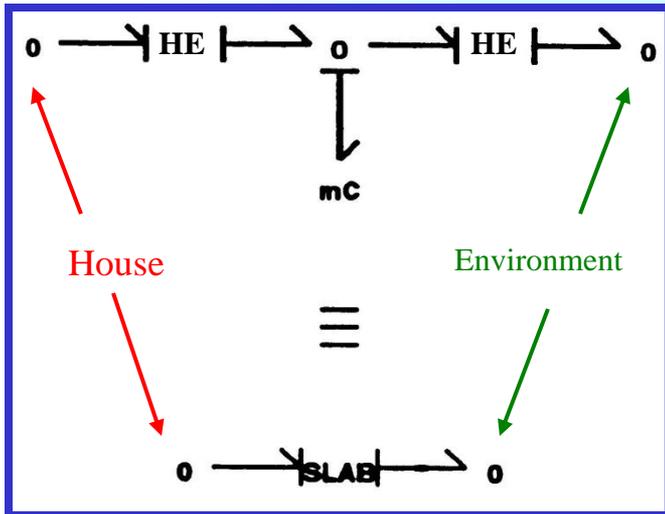
- The house has four rooms to be modeled: a living room, two bed rooms, and a sun space.
- It is assumed that the temperature within each room is constant, which makes it possible to model each room as a single 0-junction.
- ... This is clearly an experimental house, as there is neither a bathroom nor a kitchen.

The Bond Graph of a Room



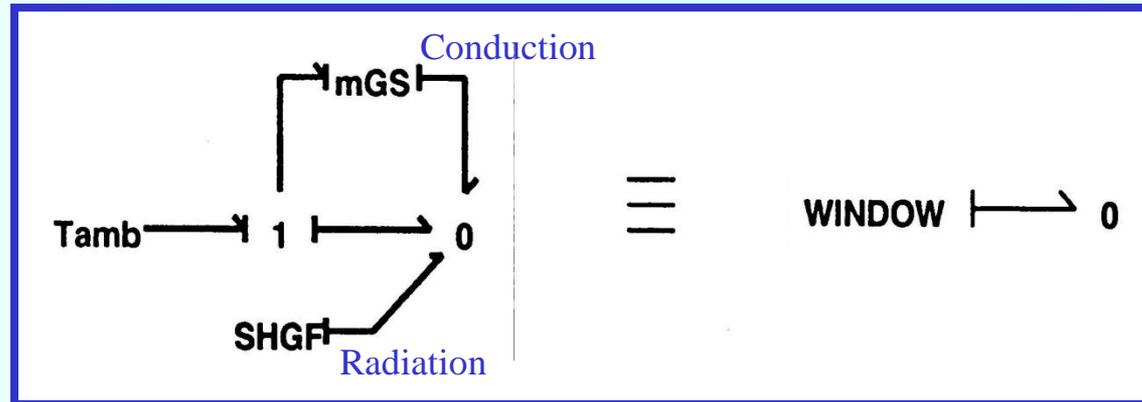
- Every room is modeled in approximately the same fashion. The model shows the heat capacity of the room as well as the interactions with the environment.

The Floor



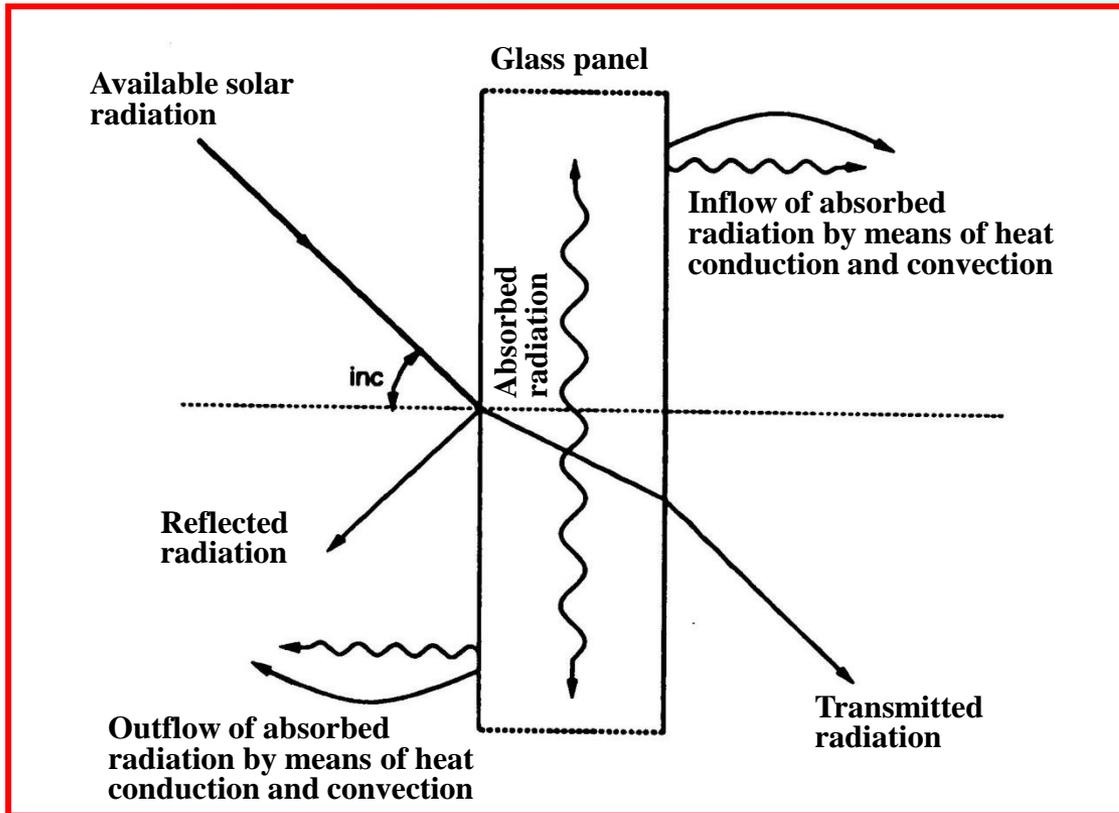
- The floor is modeled like a room.
 - It has its own heat capacity (the slab under the house consists of gravel).
 - It exchanges heat with the house.
 - It also exchanges heat with the environment.
- It is important, not to represent the exchange with the environment as a loss, since during the summer, heat is also entering the building through the slab.

The Windows I



- Heat transport across the windows occurs partly by means of *heat conduction*, and partly by means of *radiation*.

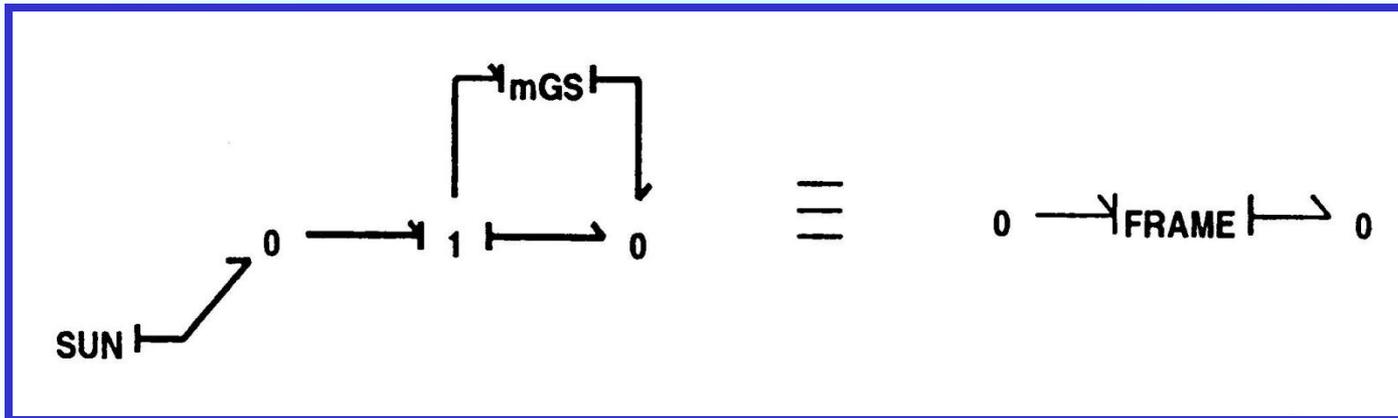
The Windows II



- Modeling the radiation accurately is not easy, since several different phenomena must be considered, and since the radiation is furthermore a function of the day of the year and the time of the day.

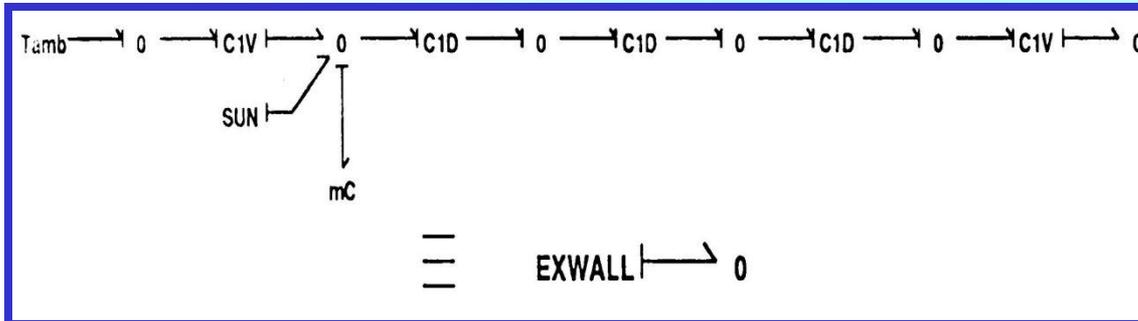
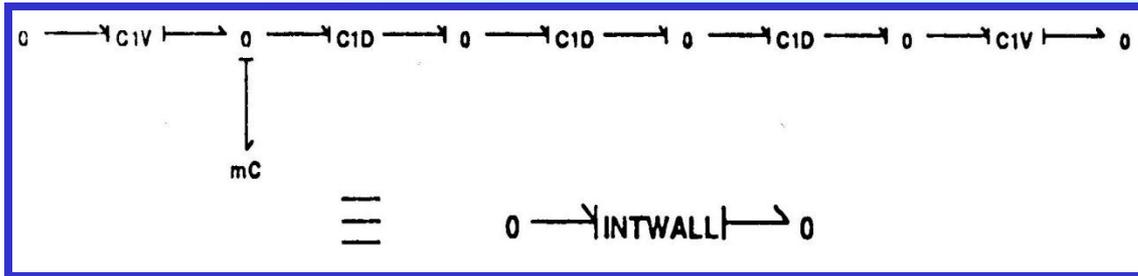
$$\equiv \text{SHGF} \rightarrow 0$$

The Doors



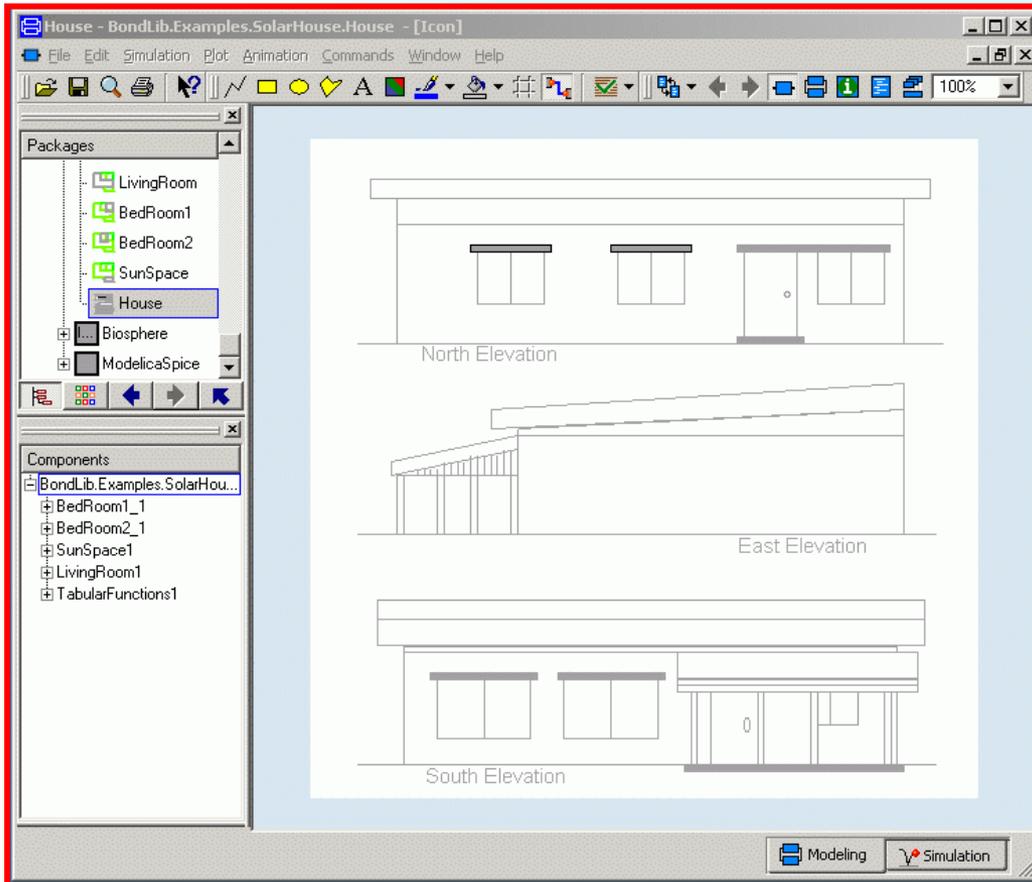
- The doors are modeled similarly to the windows, yet there is no glass, and there exists an additional heat conduction through the wood of the door.

The Walls



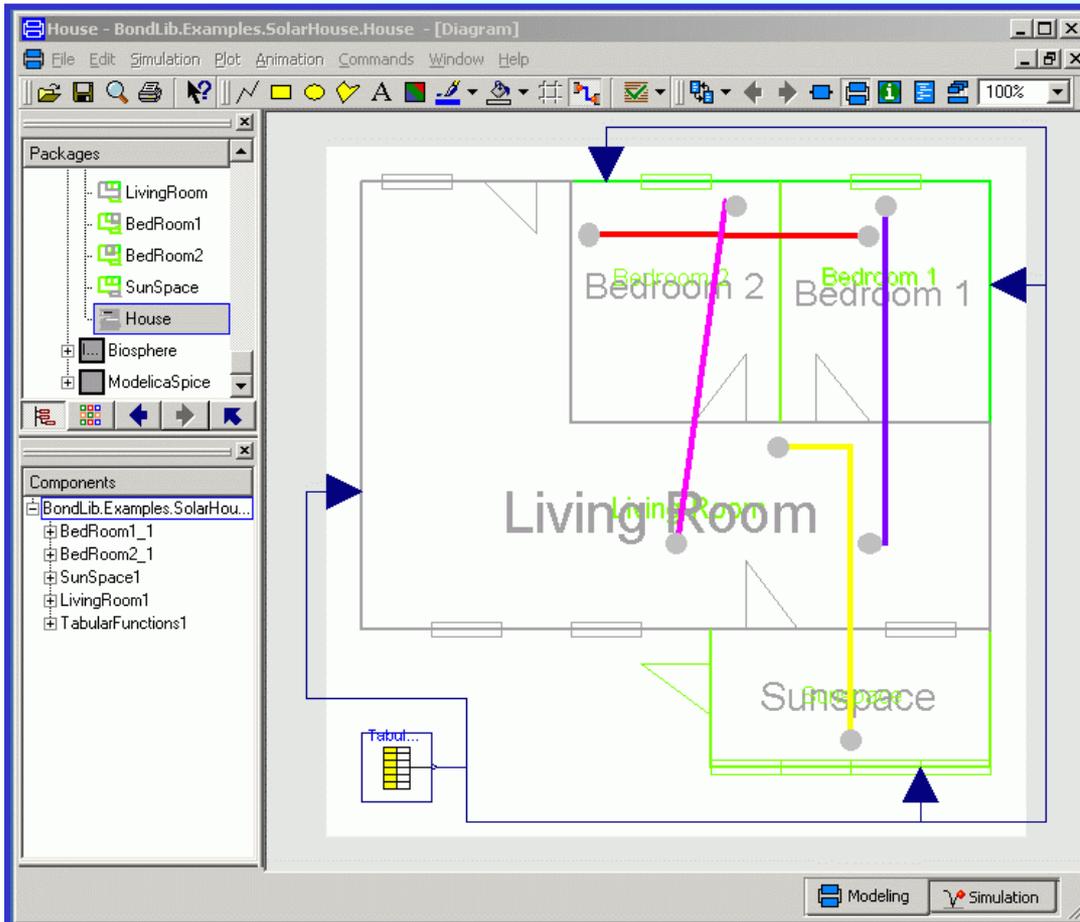
- Each wall is described by three heat conduction elements.
- At the two surfaces, there are additional convection elements modeling the transport of heat in the boundary layer.
- The exterior walls consider in addition the influence of solar radiation.
- In this program, the heat conduction elements *CID* contain on the right side a capacitor, whereas the convection elements *CIV* do not contain any capacity.

The *Dymola* Model I



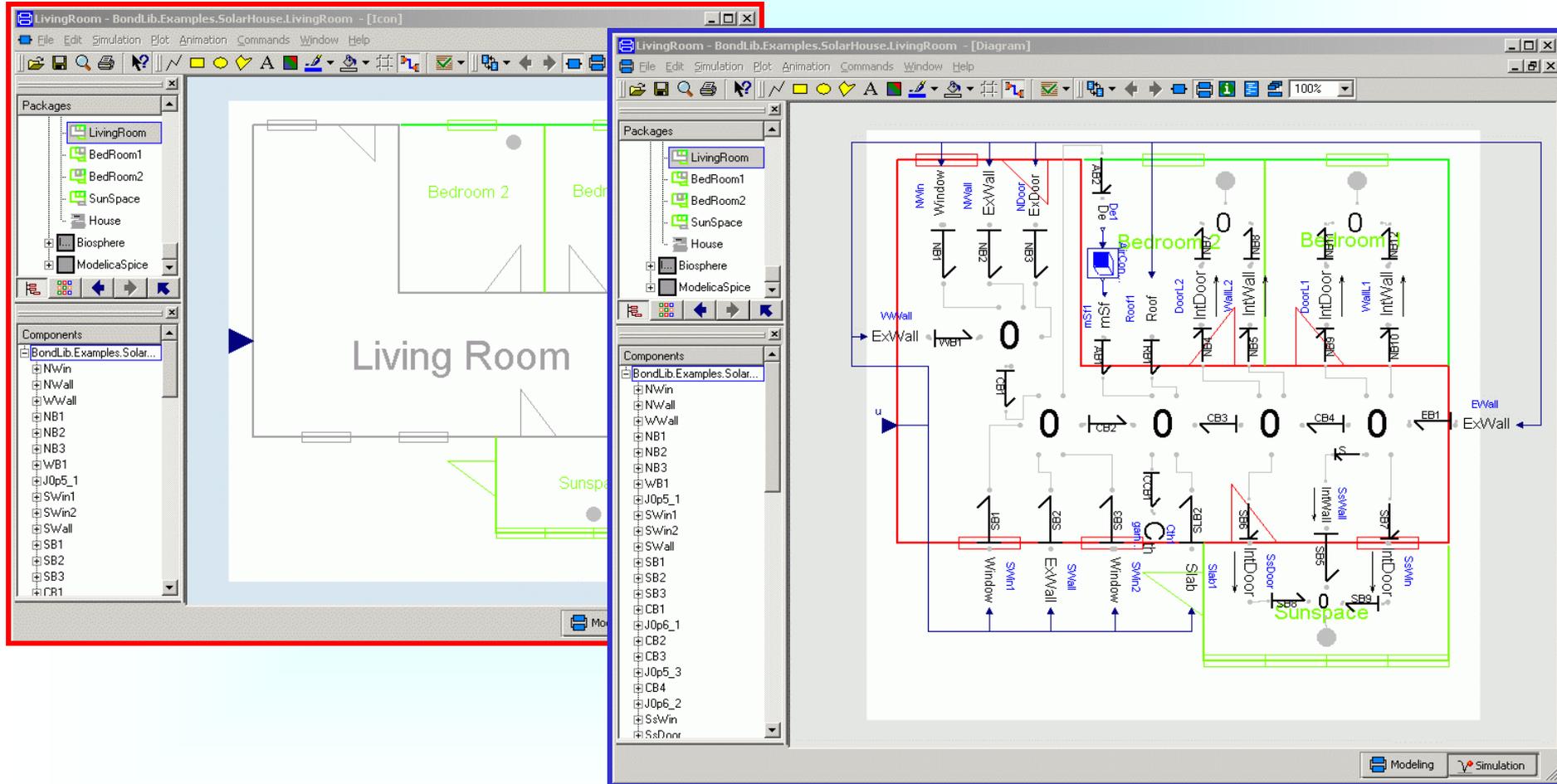
- The overall *Dymola* model is shown to the left.
- At least, the picture shown is the top-level icon window of the model.

The *Dymola* Model II

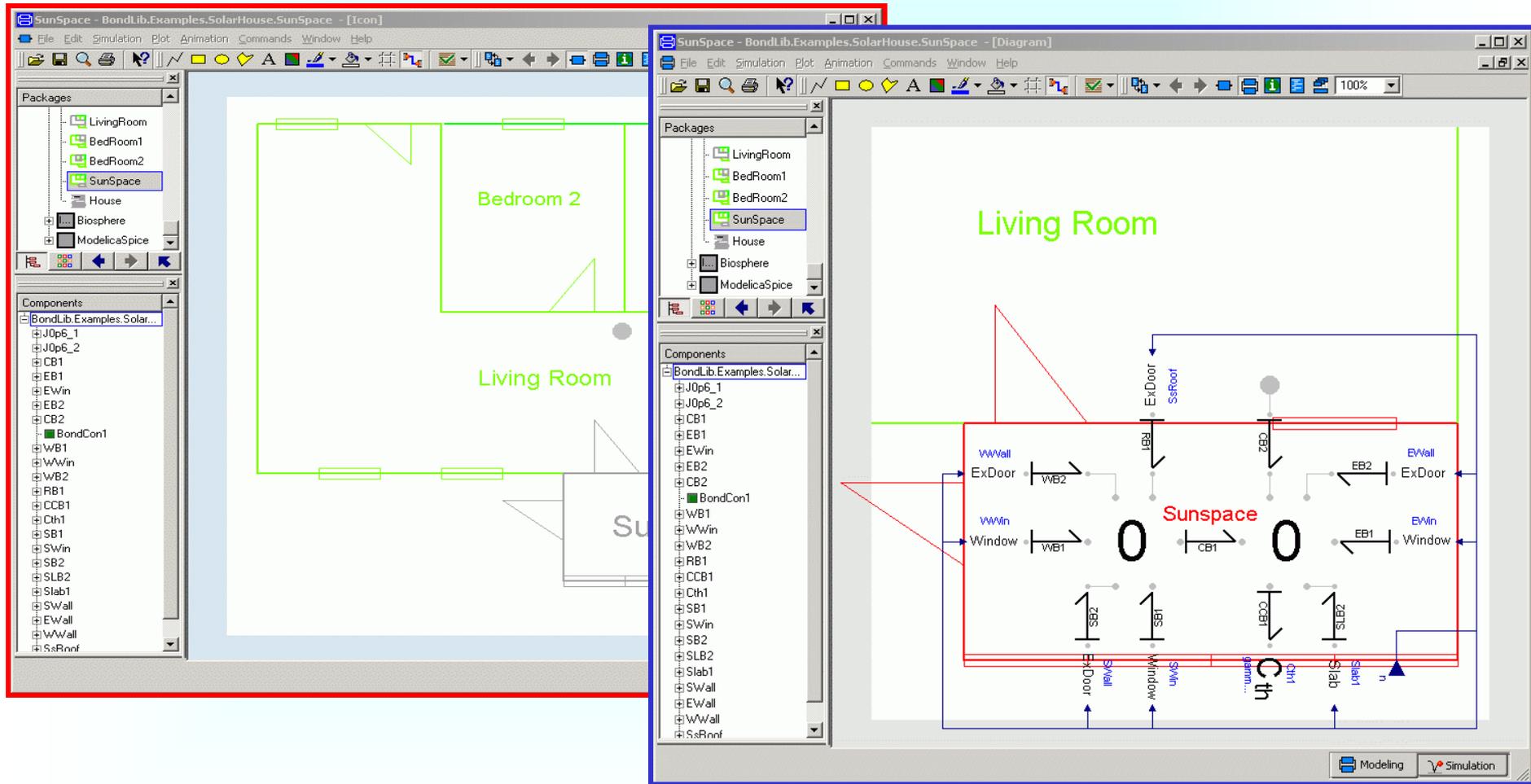


- Shown on the left side is the corresponding top-level diagram window.
- Each of the four rooms is a separate model.
- The four models are overlaid to each other.
- The bond graph connectors are graphically connected, connecting neighboring rooms to each other.

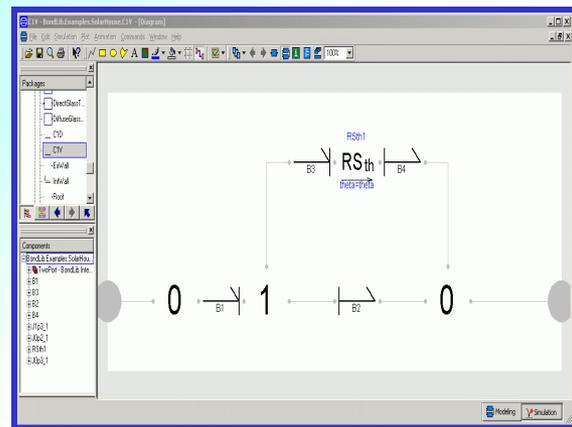
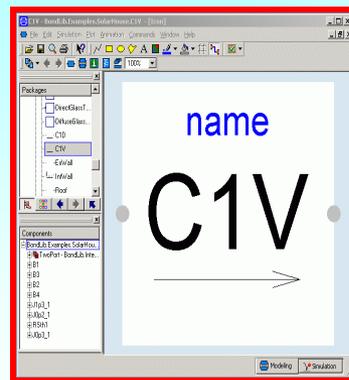
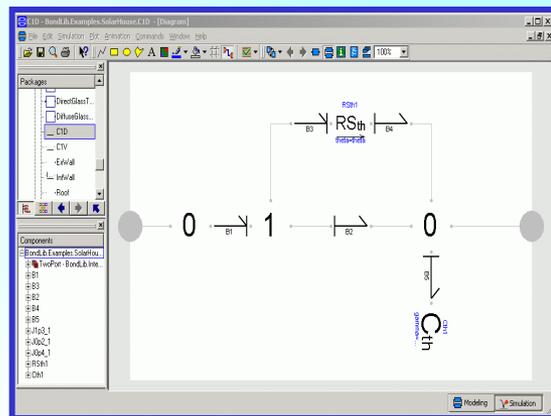
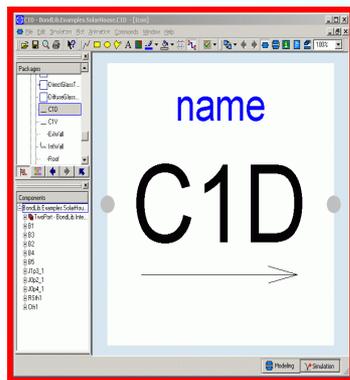
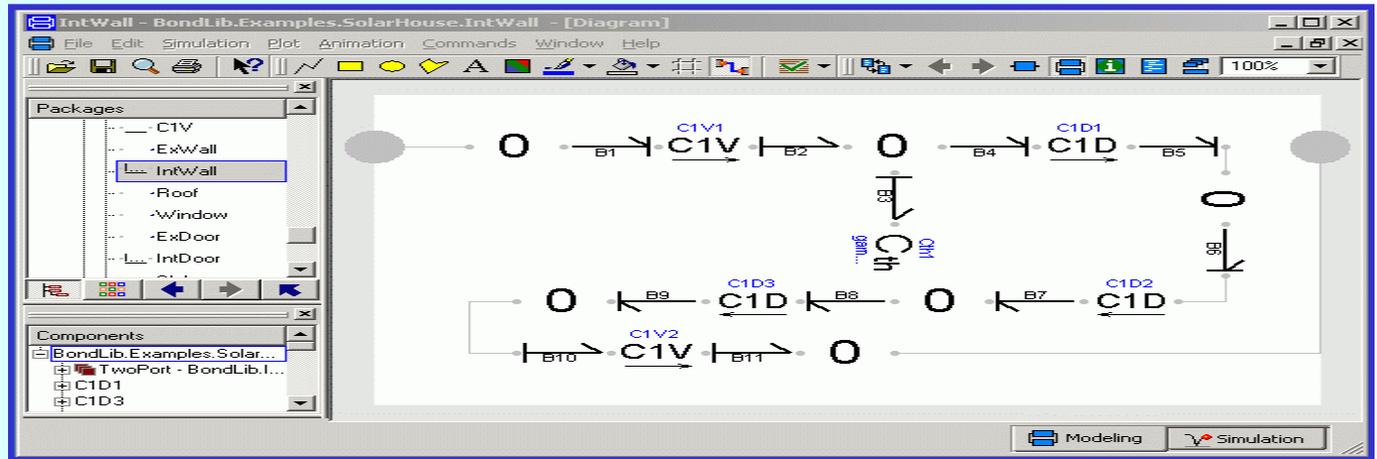
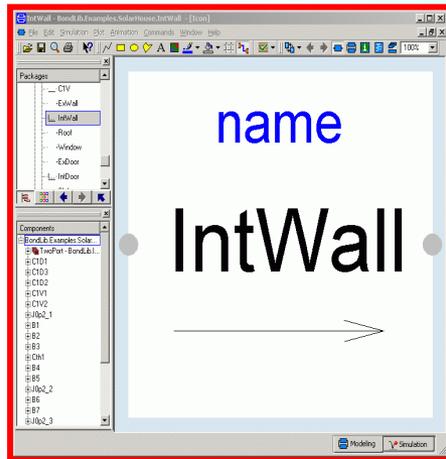
The Living Room



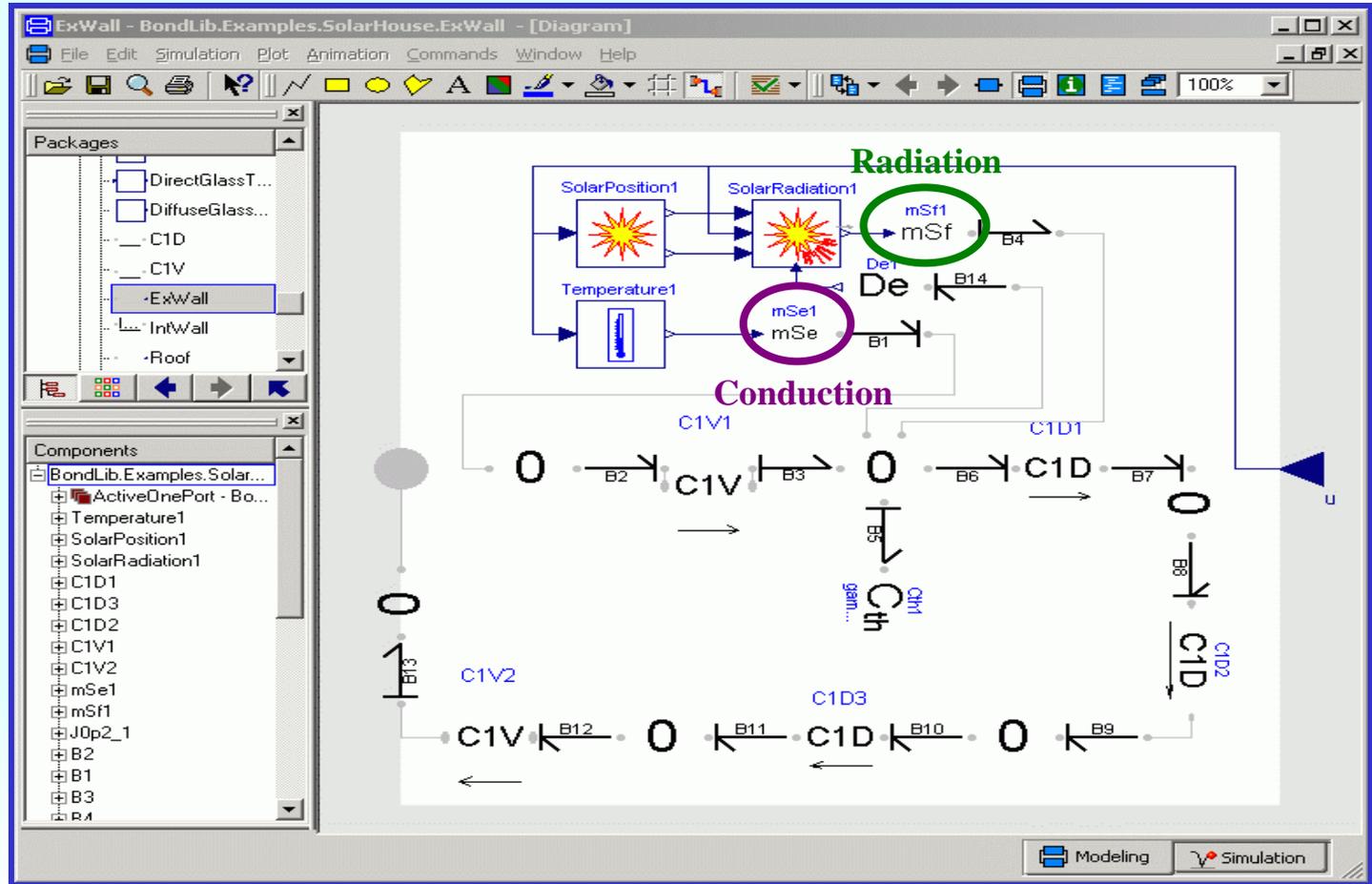
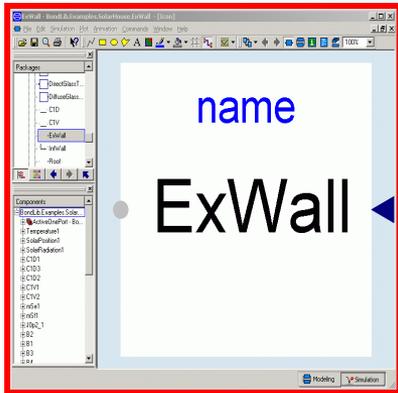
The Sunspace



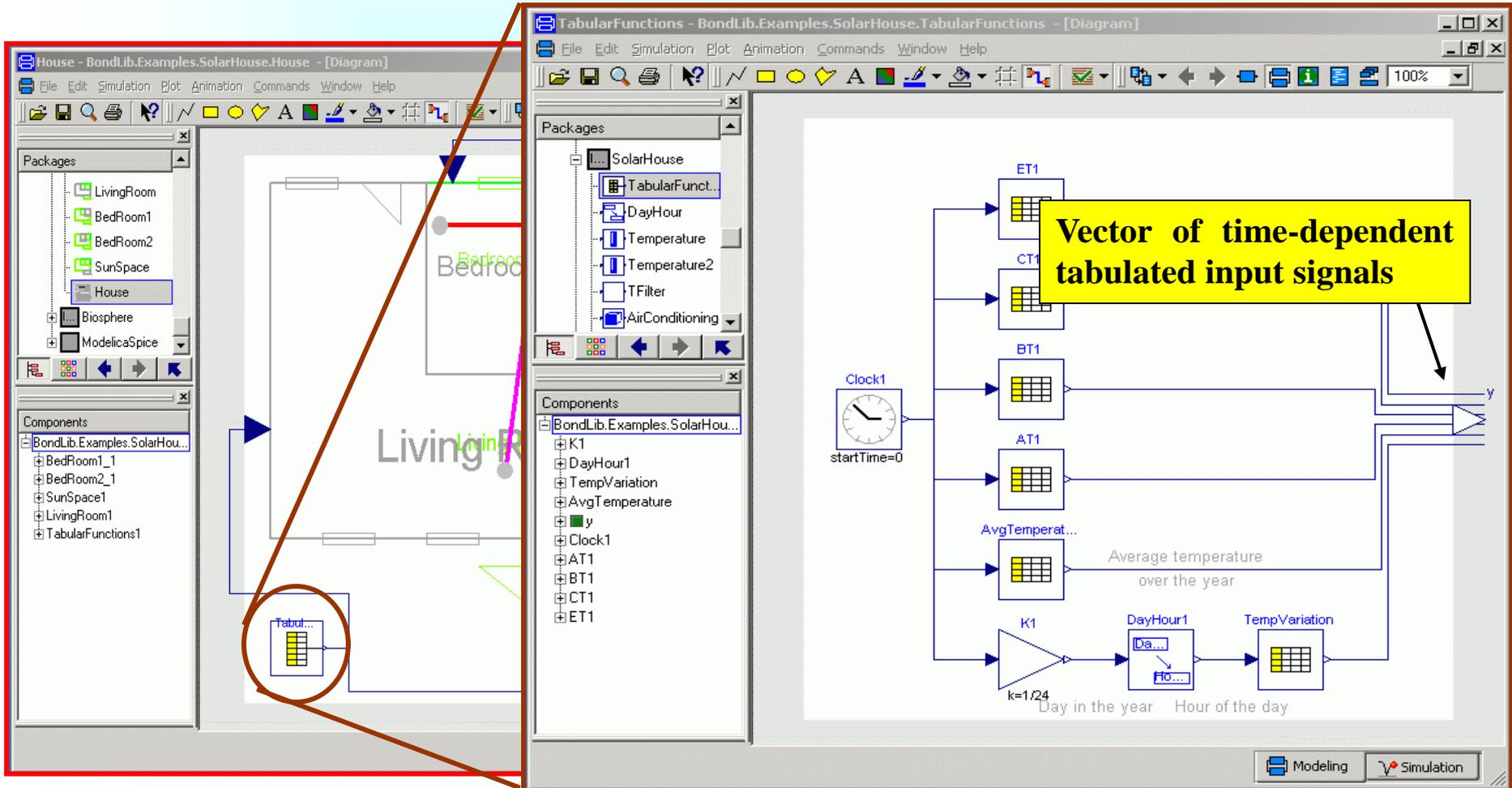
The Interior Wall



The Exterior Wall



The Tabular Functions



The Tabular Functions II

The image displays a software interface for modeling physical systems, specifically focusing on tabular functions. The main window shows a diagram of a solar house model with components like Clock1, AT1, BT1, CT1, and ET1. A green circle highlights the 'AvgTemperature' component in the diagram, which is linked to a configuration window.

The configuration window, titled 'AvgTemperature in BondLib.Examples.SolarHouse.TabularFunctions', shows the following settings:

- General** tab: Name: AvgTemperature, Comment: (empty)
- Model** tab: Path: Modelica.Blocks.Tables.CombiTable1Ds, Comment: Table look-up in one dimension (matrix/file) with one input and n outputs
- table data definition** tab:
 - tableOnFile: true (true, if table is defined on file or in function usertab)
 - table: fill(0.0, 0, 2) (table matrix (grid = first column))
 - tableName: "avgtemp_table" (table name on file or in function usertab (see docu))
 - fileName: "Solar_tables.mat" (file where matrix is stored)
- table data interpretation** tab:
 - columns: 2:size(table, 2) (columns of table to be interpolated)
 - smoothness: Blocks.Types.Smoothness.LinearSegments (smoothness of table interpolation)

Buttons at the bottom of the configuration window include OK, Info, and Cancel.

The Tabular Functions III

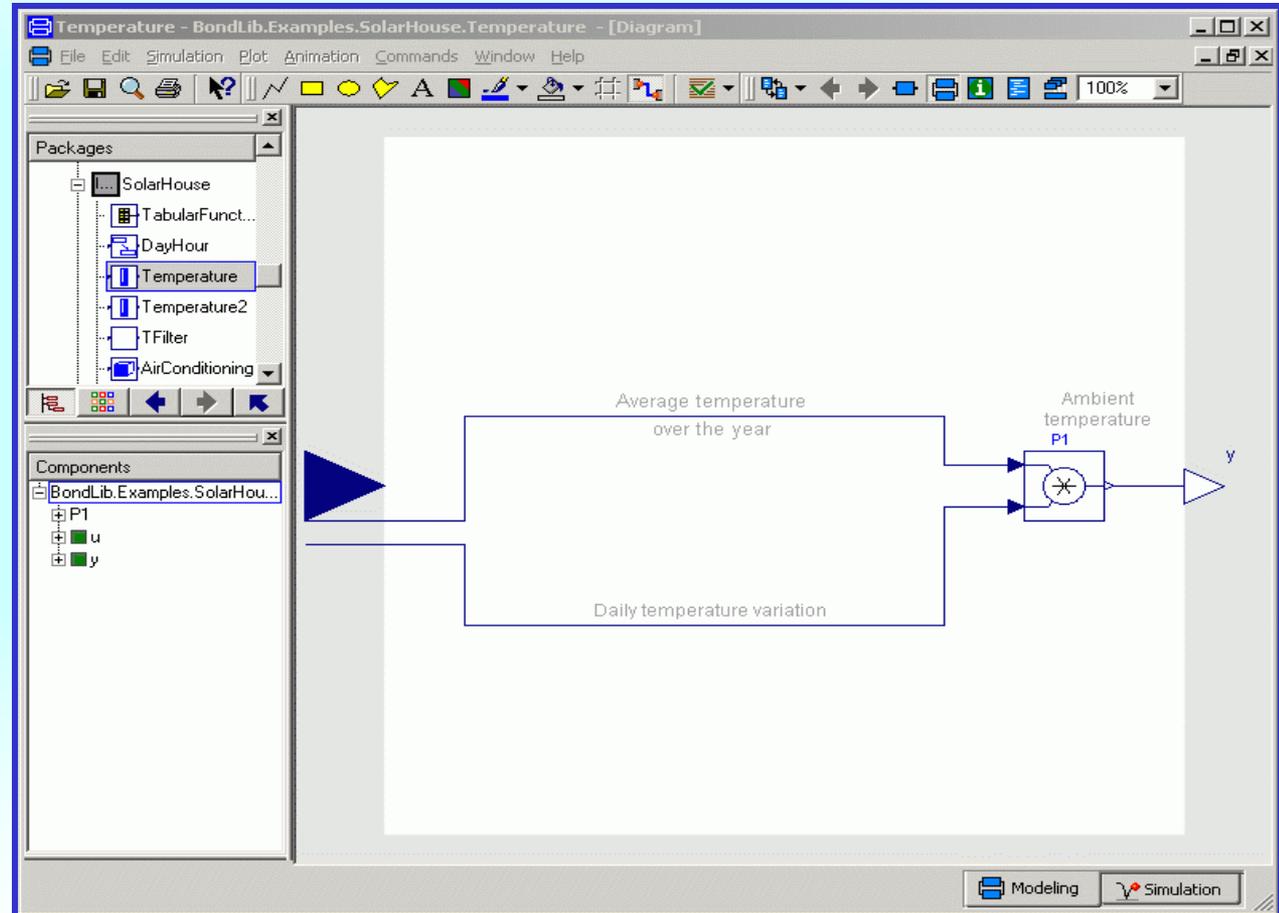
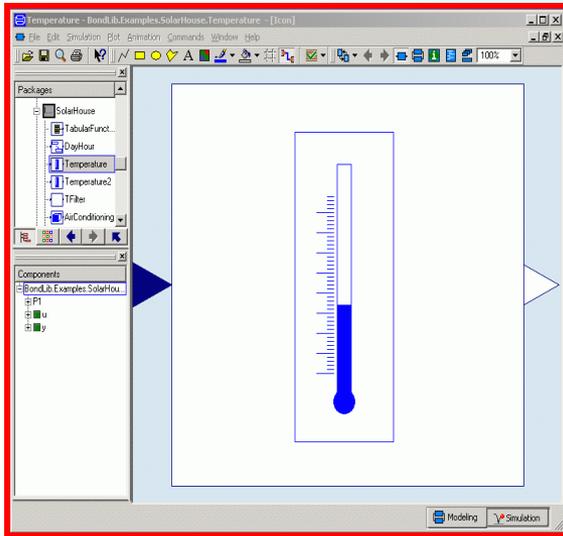
The image displays two windows from a software environment. The left window, titled "AvgTemperature in BondLib.Examples.SolarHouse", shows the configuration for a table function. The "table data definition" section is highlighted with a purple box, showing the following settings:

- tableOnFile: true
- table: fill(0,0,2)
- tableName: "avgtemp_table"
- fileName: "Solar_tables.mat"

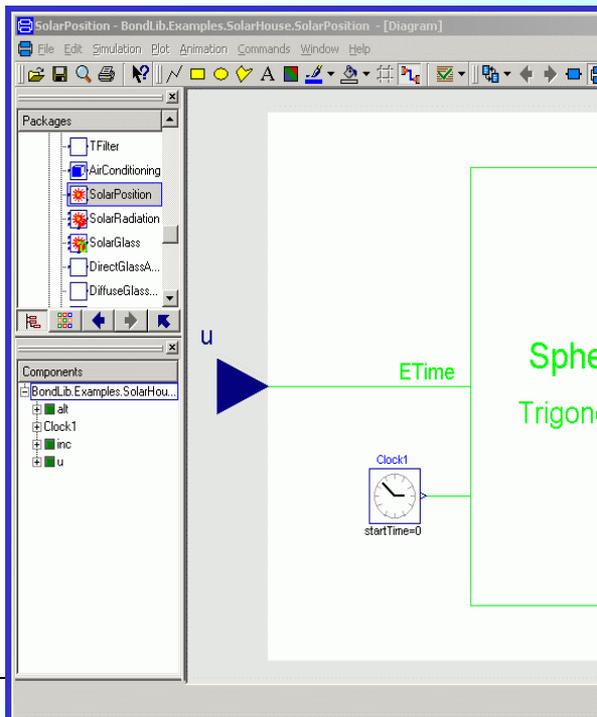
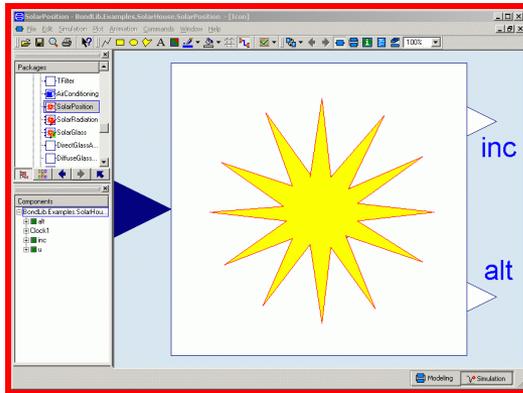
The right window, titled "C:\Cellier\Classes\Ece449\BondLib\Solar_convert.m", shows the MATLAB code for the script. The code defines the table data and saves it to a file:

```
1 % Convert the SolarHouse data to tables usable by Dymola
2 % -----
3 %
4 x_vals = [-1:33];
5 y_vals = [0.991,0.988,0.986,0.983,0.98,0.978,0.976,0.976,0.979,0.985,0.993,1.003,1.011,1.018,1.023,...
6           1.027,1.028,1.027,1.023,1.017,1.01,1.004,0.999,0.995,0.991,0.988,0.986,0.983,0.98,0.978,...
7           0.976,0.976,0.979,0.985,0.993];
8 tempvar_table = [ x_vals' , y_vals' ];
9 %
10 x_vals = [0,480,1224,1896,2640,3360,4104,4824,5568,6312,7032,7776,8496,8760];
11 y_vals = [284,283.71,284.26,288.71,292.59,296.48,302.59,302.59,302.04,299.81,294.26,287.59,284.26,284.0];
12 avgtemp_table = [ x_vals' , y_vals' ];
13 %
14 y_vals = [1232,1230,1214,1185,1135,1103,1088,1085,1107,1151,1192,1220,1233,1232];
15 AT1_table = [ x_vals' , y_vals' ];
16 %
17 y_vals = [0.14,0.142,0.144,0.156,0.18,0.196,0.205,0.207,0.201,0.177,0.16,0.149,0.142,0.14];
18 BT1_table = [ x_vals' , y_vals' ];
19 %
20 y_vals = [0.057,0.058,0.06,0.071,0.097,0.121,0.134,0.136,0.122,0.092,0.073,0.063,0.057,0.057];
21 CT1_table = [ x_vals' , y_vals' ];
22 %
23 y_vals = [-6.4,-11.2,-13.9,-7.5,1.1,3.3,-1.4,-6.2,-2.4,7.5,15.4,13.8,1.6,-6.4];
24 ET1_table = [ x_vals' , y_vals' ];
25 %
26 save Solar_tables.mat tempvar_table avgtemp_table AT1_table BT1_table CT1_table ET1_table -V4
27 %
28 return
29
```

The Temperature



The Solar Position



SolarPosition - BondLib.Examples.SolarHouse.SolarPosition - [Modelica Text]

```

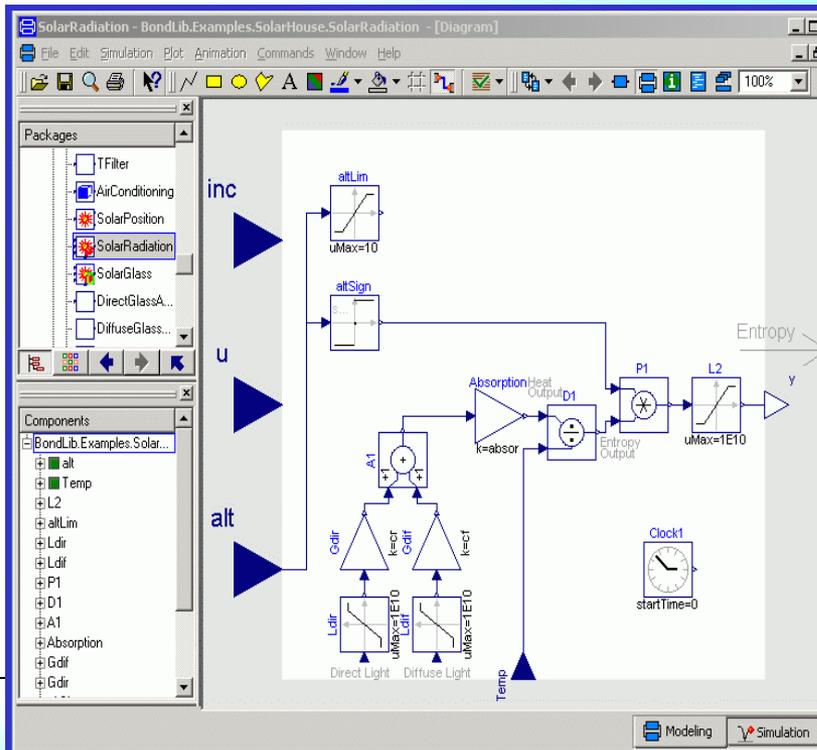
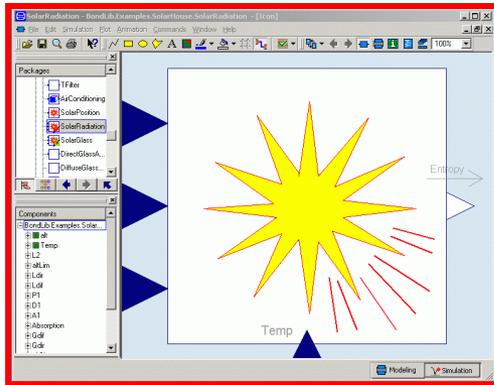
block SolarPosition "Solar position in the sky"
  constant Real pi=Modelica.Constants.pi;
  parameter Real lon(unit="deg") = 111 "Longitude";
  parameter Real lat(unit="deg") = 32 "Latitude";
  parameter Real lsm(unit="deg") = 105;
  parameter Real sfa(unit="deg") "Orientation of surface";
  parameter Real tilt(unit="deg") "Tilt of surface";
  parameter Real latr(unit="rad") = pi*lat/180 "Latitude";
  parameter Real sfar(unit="rad") = pi*sfa/180 "Orientation of surface";
  parameter Real tiltr(unit="rad") = pi*tilt/180 "Tilt of surface";
  output Real h "Height";
  output Real dec "Declination";
  output Real incl;
  output Real inc2;
  output Real inc3;
  output Real Time(unit="h");
  output Real ETime;
end SolarPosition;

equation
  Time = Clock1.y;
  ETime = u[6];
  h = (pi/12)*(Time - 12 + ETime/60 + (lsm - lon)/15);
  dec = (pi/180)*23.45*sin(2*pi*(284 + Time/24)/365);
  incl = sin(dec)*sin(latr)*cos(tiltr) - sin(dec)*cos(latr)*sin(tiltr)*cos(sfar);
  inc2 = cos(dec)*cos(latr)*cos(tiltr)*cos(h) + cos(dec)*sin(latr)*sin(tiltr)*cos(sfar)*cos(h);
  inc3 = cos(dec)*sin(tiltr)*sin(sfar)*sin(h);
  inc = incl + inc2 + inc3;
  alt = cos(latr)*cos(dec)*cos(h) + sin(latr)*sin(dec);
end SolarPosition;
  
```

Line: 1 Modeling Simulation



The Solar Radiation



```

SolarRadiation - BondLib.Examples.SolarHouse.SolarRadiation - [Modelica Text]
File Edit Simulation Plot Animation Commands Window Help
100%

Packages
├── TFilter
├── AirConditioning
├── SolarPosition
├── SolarRadiation
├── SolarGlass
├── DirectGlassA...
└── DiffuseGlass...

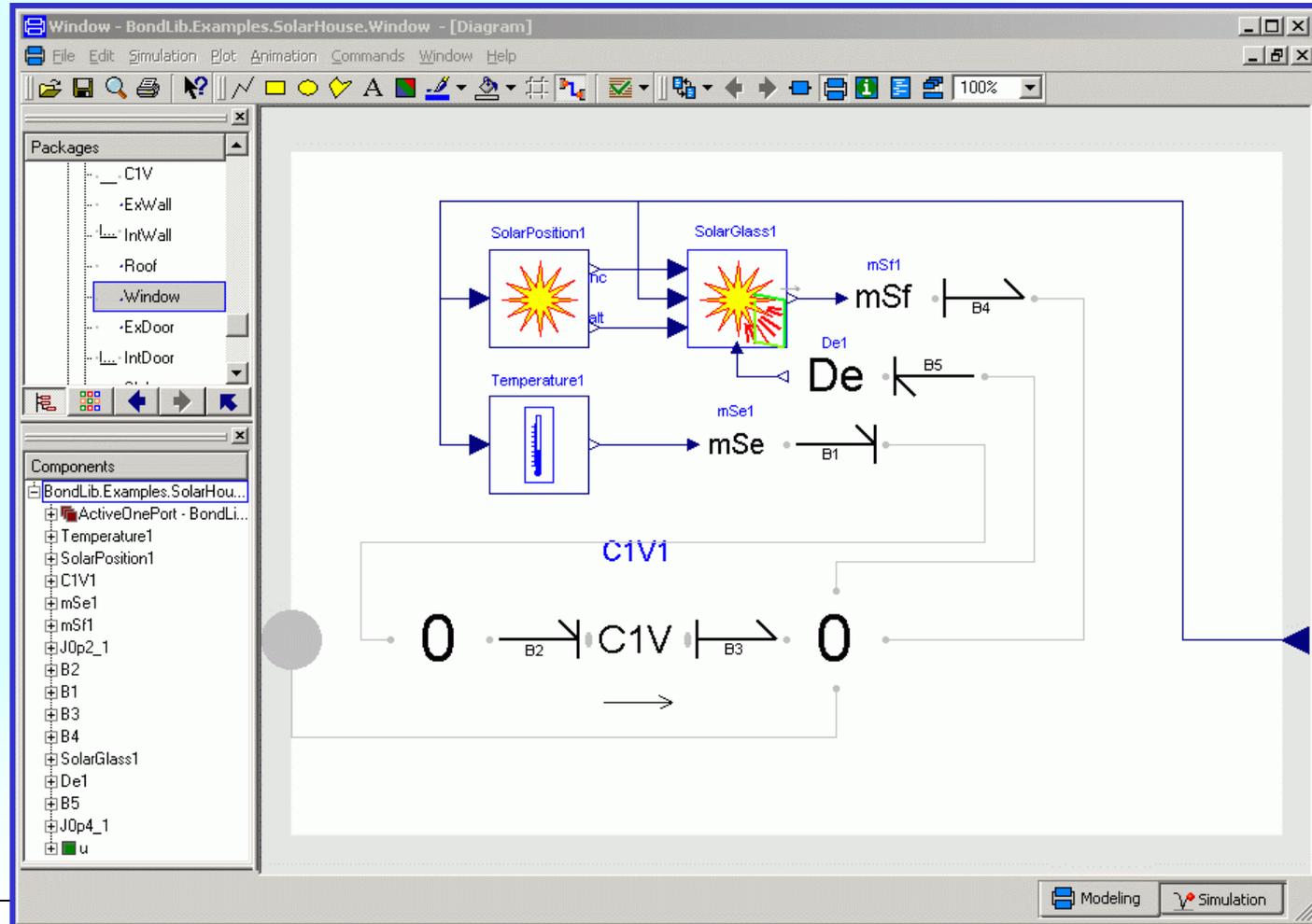
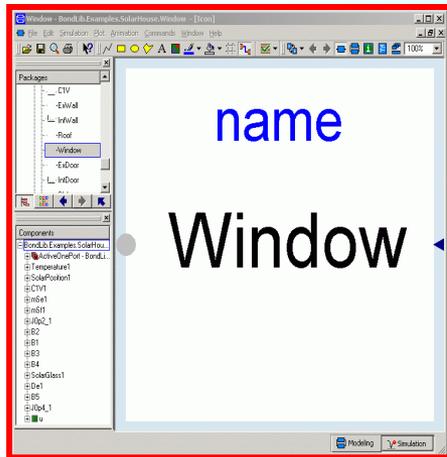
Components
├── BondLib.Examples.Solar...
├── alt
├── Temp
├── L2
├── altLim
├── Ldir
├── Ldif
├── P1
├── D1
├── A1
├── Absorption
├── Gdir
├── Gdif
└── u

block SolarRadiation "Solar radiation"
  constant Real pi=Modelica.Constants.pi;
  parameter Real sc=1353;
  parameter Real tilt(unit="deg") "Tilt of surface";
  parameter Real tiltr(unit="rad") = pi*tilt/180 "Tilt of surface";
  parameter Real ro=0.33;
  parameter Real cr=1 "Direct light gain factor";
  parameter Real cf=1 "Diffuse light gain factor";
  parameter Real absor "Absorption coefficient";
  parameter Modelica.SIunits.Area area "Area of exposed surface";
  parameter Real OH;
  parameter Real W;
  output Real altL "Limited altitude value";
  output Real sou;
  output Real idn;
  output Real dir "Direct light";
  output Real dif "Diffuse light";
  output Real Time(unit="h");
  output Real BTime;
  output Real CTime;

equation
  Time = Clock1.y;
  BTime = u[4];
  CTime = u[5];
  altL = altLim.y;
  sou = sc*(1 + 0.033*cos(2*pi*(Time/24)/365.));
  idn = sou/exp(BTime/altL);
  dir = idn*inc*(area - OH*W*(alt/sqrt(1 - alt*alt)));
  dif = idn*inc*(area - OH*W*(alt/sqrt(1 - alt*alt)));
  Ldir.u = dir;
  Ldif.u = dif;
end SolarRadiation;
Line: 1 Modeling Simulation
  
```



The Window



Translation and Simulation Logs

```

Messages - Dymola
Syntax Error  Translation  Dialog Error  Simulation

Translation of Bundle.Examples.SolarHouse.House:
DAE having 15853 scalar unknowns and 15853 scalar equations.

STATISTICS

Original Model
Number of components: 2071
Variables: 16042
  Constants: 51 (51 scalars)
  Parameters: 1793 (1787 scalars)
  Unknowns: 14198 (15859 scalars)
  Differentiated variables: 64 scalars
Equations: 11574
  Nontrivial: 7026

Translated Model
Constants: 3660 scalars
Free parameters: 908 scalars
Parameter depending: 761 scalars
Inputs: 0
Outputs: 16 scalars
Continuous time states: 64 scalars
Time-varying variables: 1329 scalars
Alias variables: 11039 scalars
Assumed default initial conditions: 64
  LogDefaultInitialConditions=true; gives more information
Number of mixed real/discrete systems of equations: 0
Sizes of linear systems of equations: {}
Sizes after manipulation of the linear systems: {}
Sizes of nonlinear systems of equations: {}
Sizes after manipulation of the nonlinear systems: {}
Number of numerical Jacobians: 0

Finished
// experiment StopTime=8600
Finished
  
```

```

Messages - Dymola
Syntax Error  Translation  Dialog Error  Simulation

Log-file of program ./dymosim
(generated: Thu Oct 25 14:52:37 2007)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "Solar_tables.mat" loading (tables for interpolation)
... "House.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslirt of Petzold modified by Dynasim))
The following error was detected at time: 401.6598918572815
Error: Singular inconsistent scalar system for LivingRoom1.SWin1.ClV1.RSthl.fl
Solver will attempt to handle this problem.

The following error was detected at time: 833.8987732753258
Error: Singular inconsistent scalar system for SunSpacel.RWin.ClV1.RSthl.fl =
Solver will attempt to handle this problem.

The following error was detected at time: 1002.000381787843
...
Solver will attempt to handle this problem.

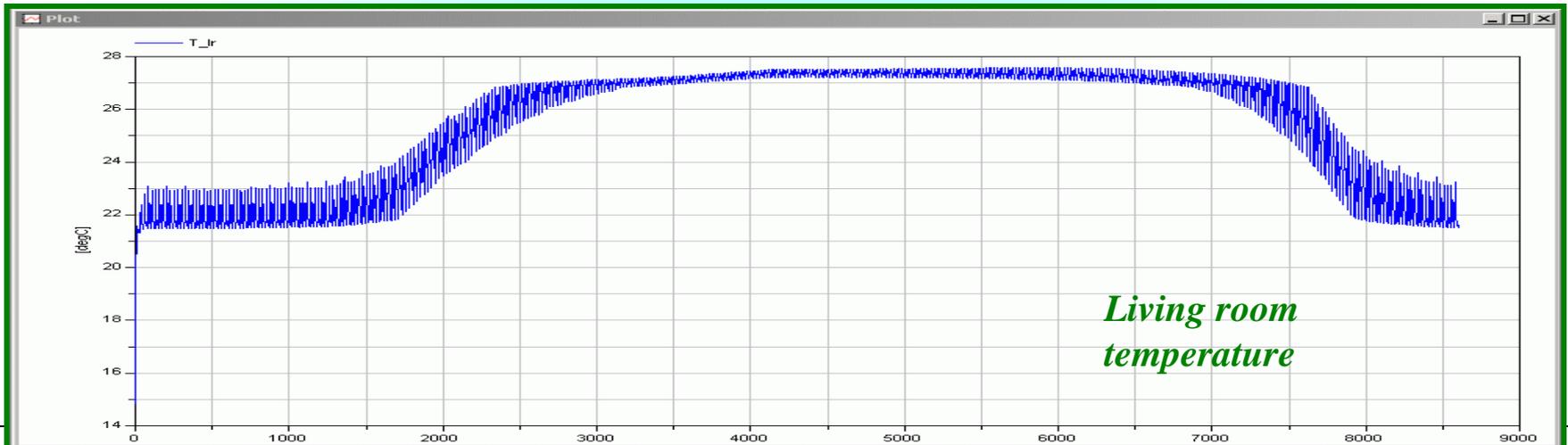
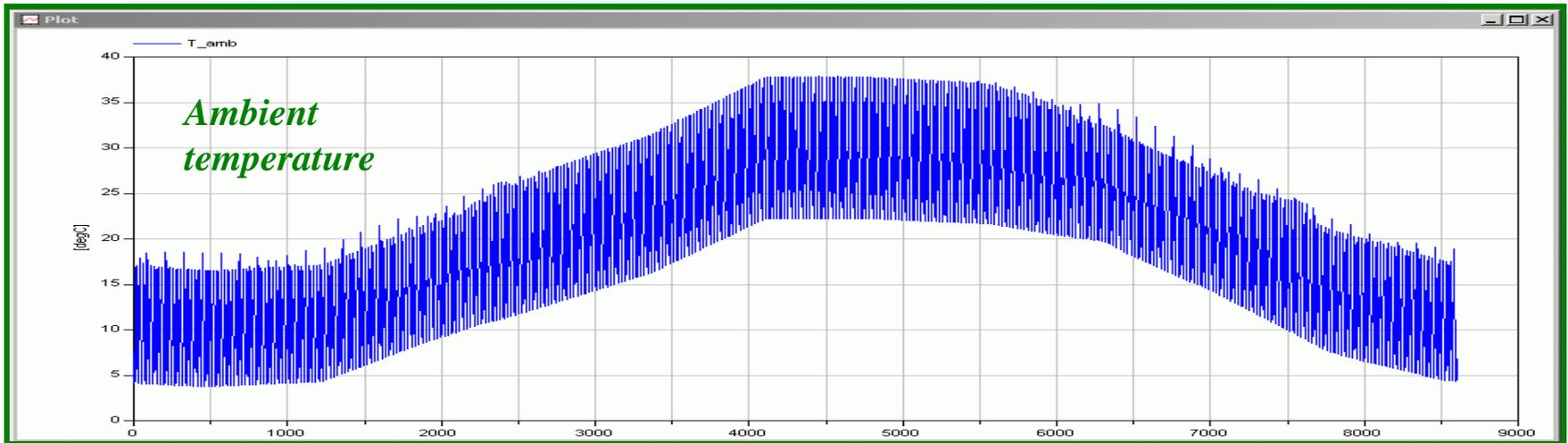
The following error was detected at time: 7529.398573012541
Error: Singular inconsistent scalar system for LivingRoom1.SWin1.ClV1.RSthl.fl
Solver will attempt to handle this problem.

Integration terminated successfully at T = 8600
WARNING: You have many state events. It might be due to chattering.
Enable logging of event in Simulation/Setup/Debug/Events during simulation
CPU-time for integration : 332 seconds
CPU-time for one GRID interval: 665 milli-seconds
Number of result points : 10000
Number of GRID points : 501
Number of (successful) steps : 60734
Number of F-evaluations : 2409197
Number of H-evaluations : 114000
Number of Jacobian-evaluations: 35910
Number of (model) time events : 0
Number of (U) time events : 0
Number of state events : 6527
Number of step events : 0
Minimum integration stepsize : 1.03e-006
Maximum integration stepsize : 1.51
Maximum integration order : 4

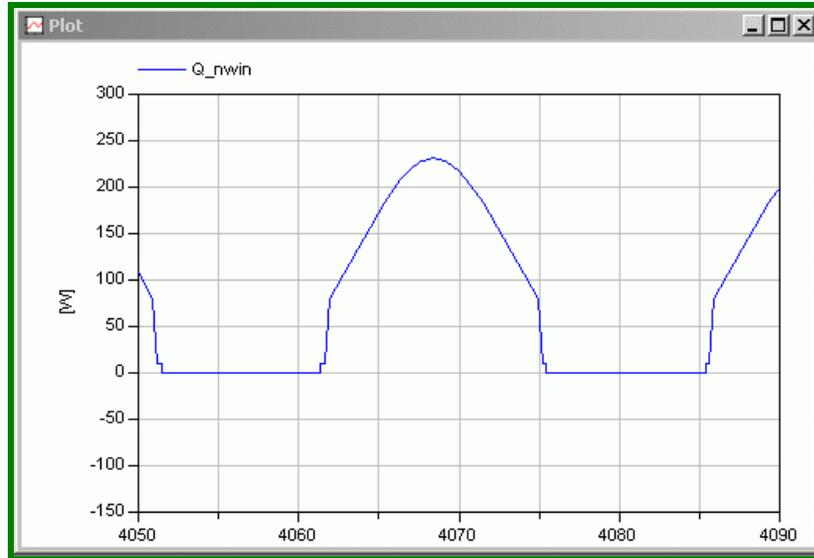
Calling terminal section
... "dsfinal.txt" creating (final states)
  
```



Simulation Results I

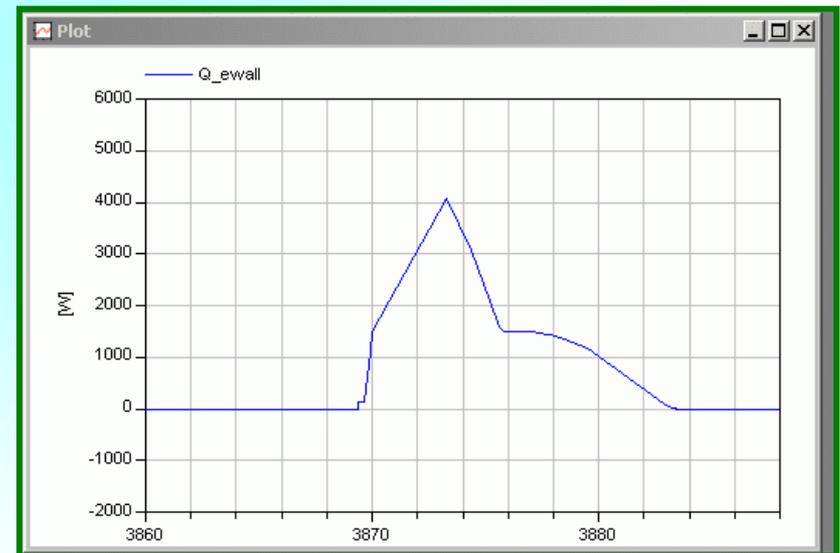


Simulation Results II

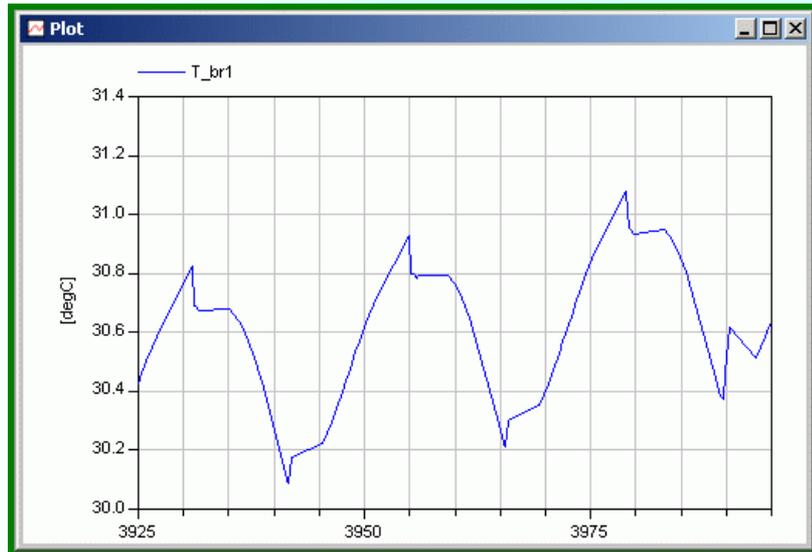


Radiation through North-exposed window

Radiation through East-exposed wall

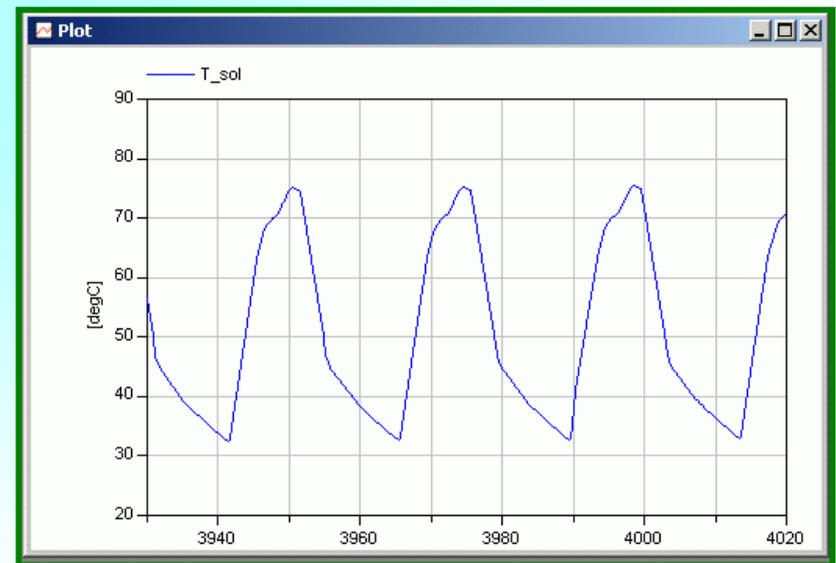


Simulation Results III



Temperature in bedroom #1

Temperature in sunspace

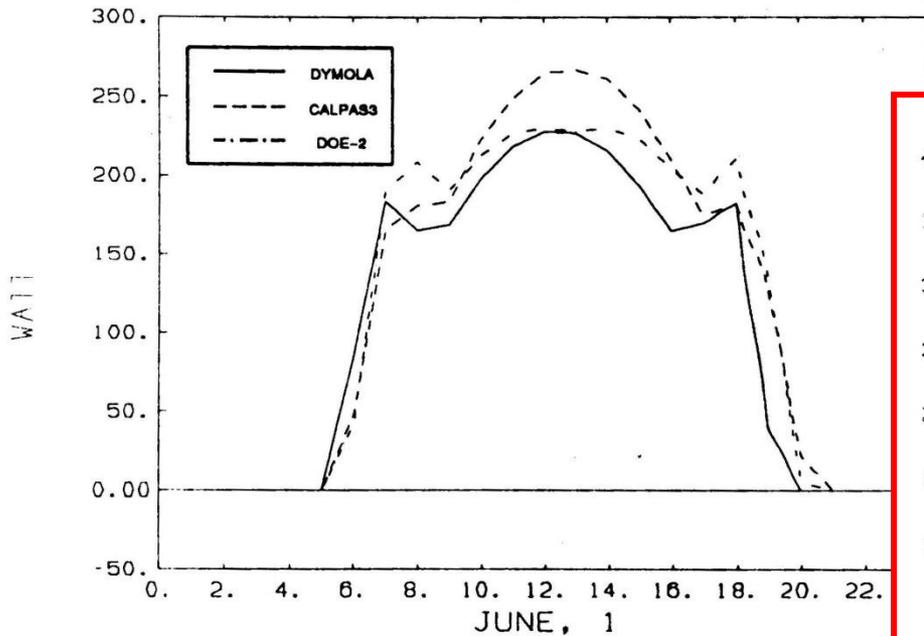


Passive Solar Space Heating III

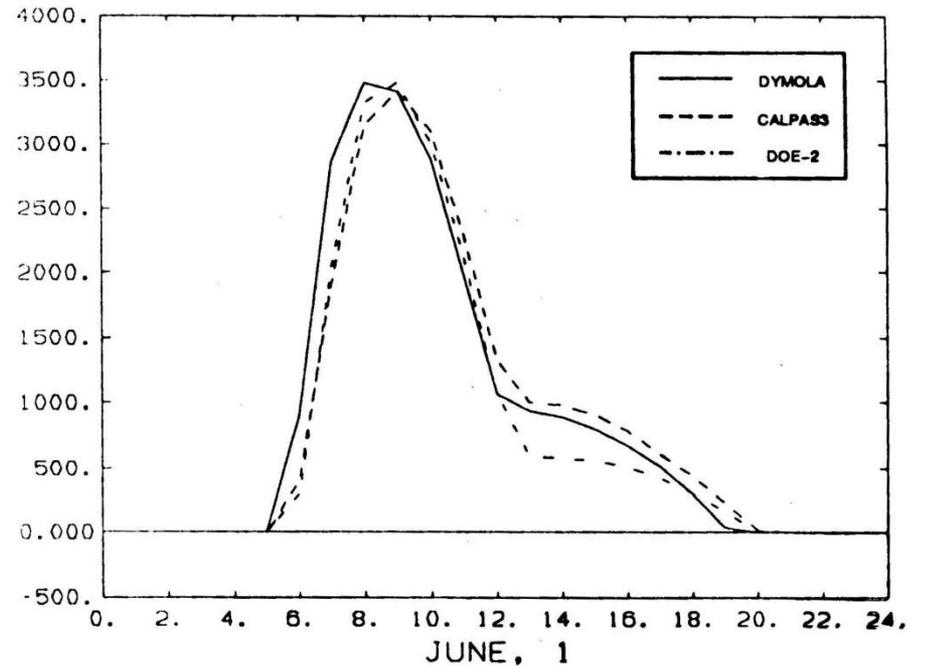
- The simulation results of three different programs were compared. These programs had been coded in *Dymola*, *Calpas 3*, and *DOE 2*.
- *Calpas 3* and *DOE 2* are commercial simulation programs specialized for space heating.
- *Calpas 3* is a fairly simple Program. It computes rapidly and is easy to use, as it offers only few parameters. However, the results aren't very precise.
- *DOE 2* is a much more accurate and rather expensive program. It computes slowly and is not easy to use, as it offers many parameters, for which the user must supply values.

Simulation Results IV

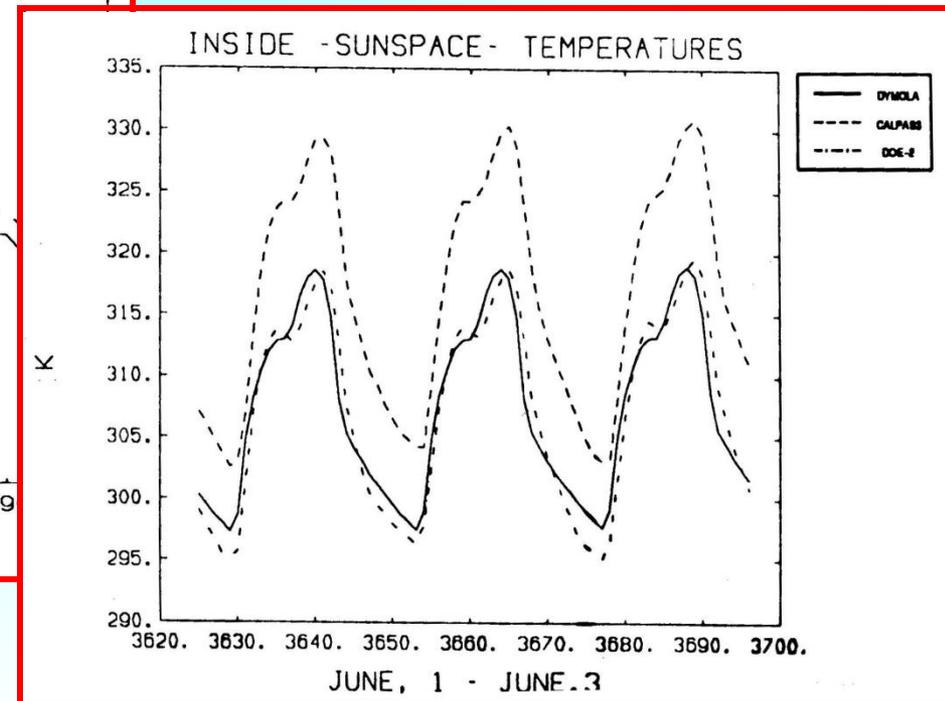
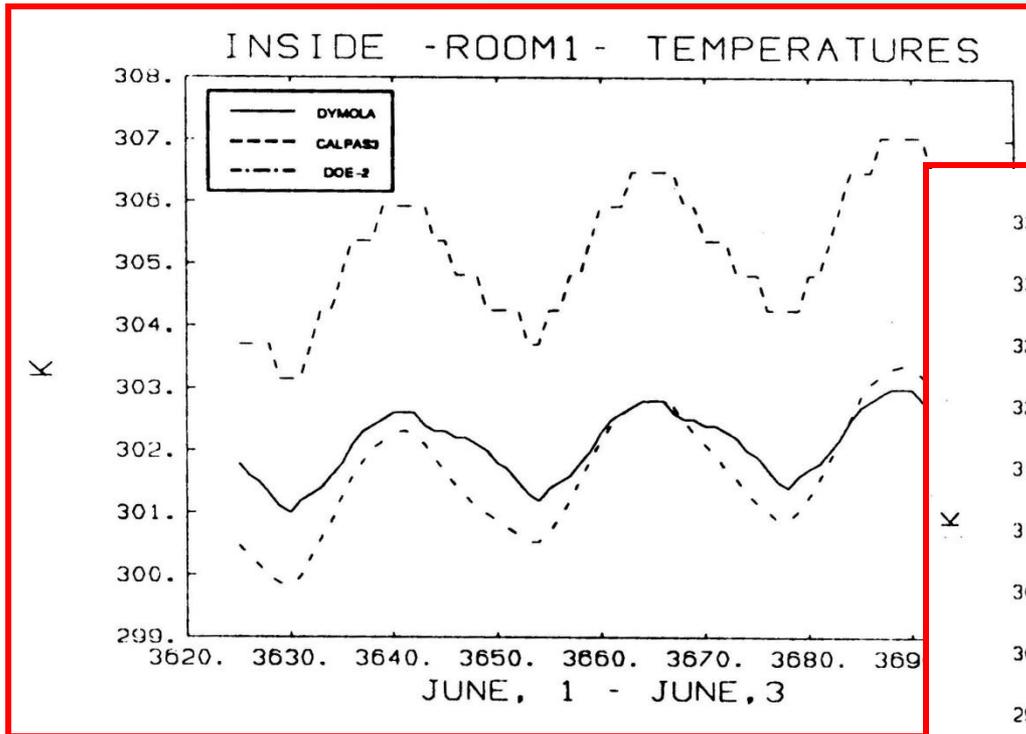
SOLAR RADIATION THROUGH A NORTH EXPOSED WINDOW



SOLAR RADIATION THROUGH A EAST EXPOSED WINDOW



Simulation Results V



Passive Solar Space Heating IV

- *Dymola* computes about as accurately as *DOE 2*. However, the time needed to complete a simulation run is shorter by about a factor of 50 in comparison with *DOE 2*.
- *Dymola* is much more flexible, as the program is not specialized for space heating simulations.
- The model assumptions, on which the simulation results are based, are clearly visible in the case of *Dymola*. This is not the case for either of the other two programs.

References

- Weiner, M. (1992), *Bond Graph Model of a Passive Solar Heating System*, MS Thesis, Dept. of Electr. & Comp. Engr., University of Arizona, Tucson, AZ.
- Weiner, M., and F.E. Cellier (1993), “Modeling and Simulation of a Solar Energy System by Use of Bond Graphs,” *Proc. SCS Intl. Conf. on Bond Graph Modeling*, San Diego, CA, pp.301-306.
- Cellier, F.E. (2007), *The Dymola Bond-Graph Library*, Version 2.3.