Popolo Reference Manual

Contents

Chapter 1 Introduction

1.1 What is Popolo

Popolo is a collection of classes for calculating various heat transfer phenomena. The routines have been written from scratch in C#, and present a modern Applications Programming Interface (API) for .NET Framework programmers, allowing wrappers to be written for very high level languages. It contains classes to calculate solid conduction, convective heat transfer near wall surfaces, air ventilation, radiative heat balance of wall surfaces, transmitted solar radiation through a window, and so on. Users should build up these classes to simulate a whole complex building system. A sample source code to build test cases of BESTEST are provided. Since all the source code is distributed under the GNU General Public License, they can be freely downloaded from the Web site.

This manual describes how to use Popolo in your program. Some example codes are also provided.

1.2 Where to Download Popolo

The latest release of Popolo can be downloaded from website (http://gf.hvacsimulator.net). If you extract zipped file, you can find two dll files, Popolo.dll and GSLNET.dll. Popolo.dll is a main file and GSLNET.dll numerical library which is used in Popolo. GSLNET.dll is a wrapper library for GSL (GNU Scientific Library).

1.3 Making the first simple program

In order to use a “Popolo.dll” in your application, you must first add a reference to it. The procedure to make reference to dll files with Visual Studio 2008 is described below.

Figure 1.1 shows the start up window of Visual Studio. Selecting “File” - “New” - “Project”, you can open “New project” window as shown in Figure 1.2. Select “Visual C#” and “Console Application”, then click “OK” button.[†1)]

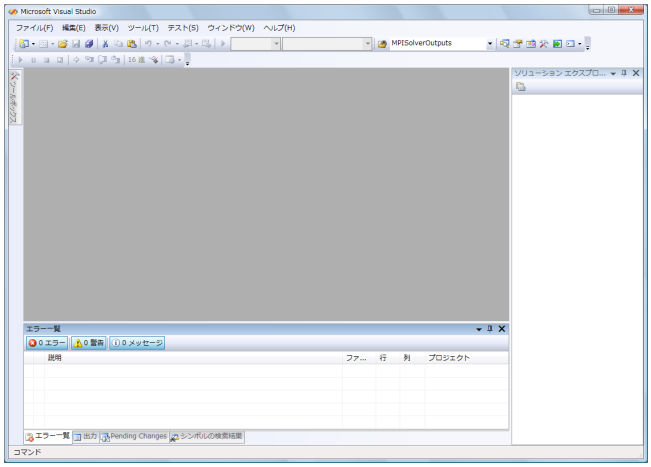

Fig.1.1 Start up window of Visual Studio

†1) You may also use some other languages which support .NET Framework such as C++.NET or Basic.NET.

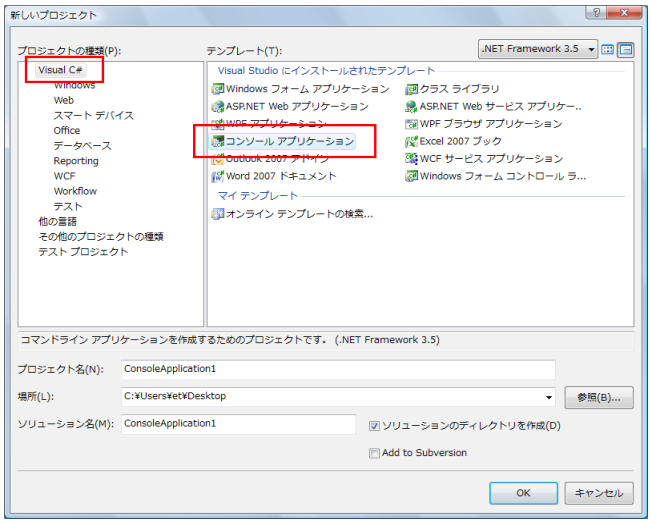

Fig.1.2 New Project Window

As shown in figure 1.3, a simple program is automatically generated by Visual Studio. In Solution Explorer, right-click on the project node and click Add Reference. In the Add Reference dialog box (Figure 1.4), select the "Browse" tab to browse for "Popolo.dll" in the file system (Figure 1.5). You can find that the reference to the "Popolo.dll" is added in the Solution Explorer (Figure 1.6).

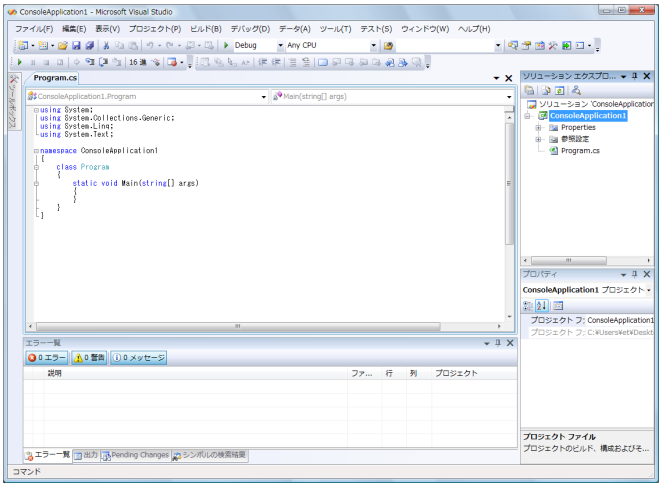

Fig.1.3 Console application project automatically generated by Visual Studio

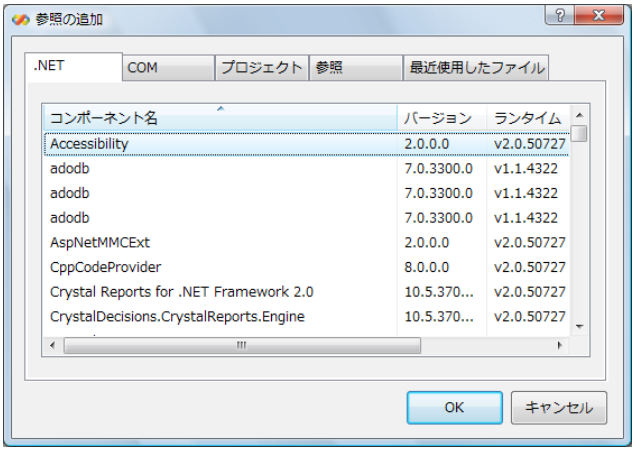

Fig.1.4 Add Reference dialog box 1

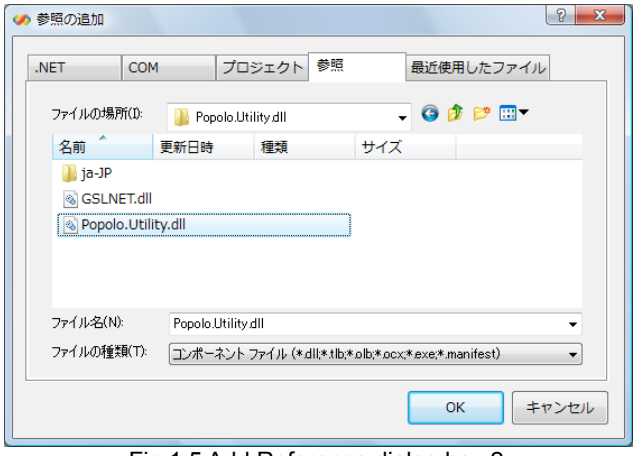

Fig.1.5 Add Reference dialog box 2

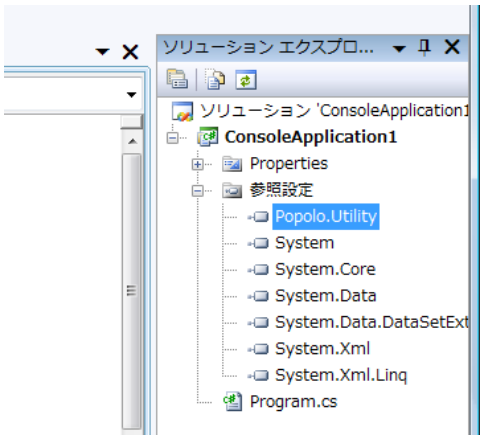

Fig.1.6 Reference to “Popolo.dll”

Popolo.dll includes various types of classes to execute building environmental simulation. They are divided into some name spaces. C# programs are organized using namespaces. “”using” directives are provided to facilitate the use of namespaces.

The following example (Figure 1.7) shows how to define a using directive. In line 3, “Popolo.ThermophysicalProperty” namespace is referenced. It is the namespace that provides classes to calculate thermo-physical property of water or moist air. In line 11, MoistAir class which belongs to “Popolo.ThermophysicalProperty” is called.

```
1  using System;
2
3  using Popolo.ThermophysicalProperty;
4
5  namespace ConsoleApplication
6  {
7     class Program
8     {
9        static void Main(string[] args)
10       {
11          double cpAir = MoistAir.GetSpecificHeat(0.018);
12          Console.WriteLine("Specific heat of moist air at absolute humidity of 0.018 kg/kg(DA) is" + cpAir.ToString(“F3”) + "kJ/K");
13          Console.Read();
14       }
15    }
16 }
```

Fig.1.7 Sample program using “Popolo.ThermophysicalProperty” namespace

Either by hitting F5 key or choosing "Start Debugging" from the menu, the executable will start. Figure 1.8 is the result of the program. You can find that the specific heat of moist air is successfully calculated

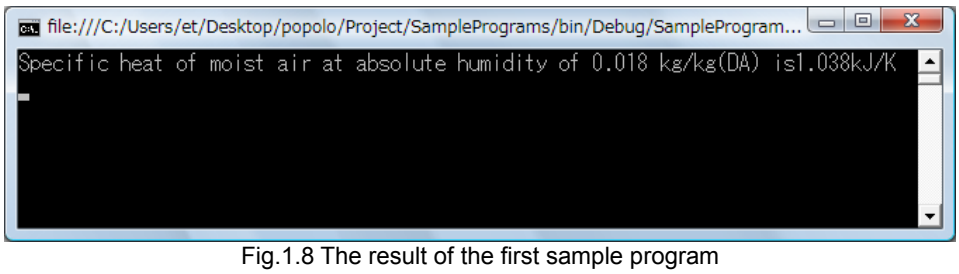

Fig.1.8 The result of the first sample program

Chapter 2 Calculating Thermodynamic Properties

2.1 Name space for calculating thermodynamic properties

The classes which calculate a thermodynamic properties belong to “Popolo.ThermophysicalProperty” namespace. Table 2.1 shows principal members defined in “Popolo.ThermophysicalProperty” namespace.

Table 2.1 Principal members defined in “Popolo.ThermophysicalProperty” namespace

Name	Type	General function
MoistAir	class	A class which express the moist air. The functions which calculates thermodynamic properties of the moist air is also defined in this class.
ImmutableMoistAir	interface	Read only interface for MoistAir class.
Water	static class	The functions which calculates thermodynamic properties of the water is defined in this class.

2.2 Calculating thermodynamic properties of moist air

The MoistAir class can be used to calculate thermodynamic properties of the moist air. Most of functions are transported from HVACSIM[1)]. and some of the functions are based on the Udagawa's program[2)].

2.2.1 The members of the MoistAir class

There are 7 primary variables which characterize a moist air, “Drybulb temperature”, “Wetbulb temperature”, “Enthalpy”, “Relative humidity”, “Absolute humidity”, “Specific volume” and “Atmospheric pressure”. These variables are defined as properties of the MoistAir class as shown in the table 2.2.

Table 2.2 Properties of the MoistAir class

Name	Mean	Has set accessor	Unit	Type
DryBulbTemperature	Drybulb temperature	yes	°C	double
WetBulbTemperature	Wetbulb temperature	yes	°C	double
Enthalpy	Enthalpy	yes	kJ/kg	double
RelativeHumidity	Relative humidity	yes	%	double
AbsoluteHumidity	Absolute humidity	yes	kg/kg(DA)	double
SpecificVolume	Specific volume	yes	m^3/kg	double
AtmosphericPressure	Atmospheric pressure	yes	kPa	double

Figure 2.1 is the sample code which edits the values of the properties of MoistAir object. In the line 11, an instance of the MoistAir class is created. From line 13 to line 20, values are set to the properties. From line 22 to line 29, the values of the MoistAir object is written to the standard output stream.

```
using System;
using Popolo.ThermophysicalProperty;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating instance of MoistAir class
            MoistAir mAir = new MoistAir();

            //Set values to property
            mAir.DryBulbTemperature = 25.6;
            mAir.AbsoluteHumidity = 0.018;
            mAir.RelativeHumidity = 50.0;
            mAir.WetBulbTemperature = 22;
            mAir.SpecificVolume = 0.86;
            mAir.Enthalpy = 58.0;
            mAir.AtmosphericPressure = 101.325;

            //Output values of properties
            Console.WriteLine("Drybulb Temperature:" + mAir.DryBulbTemperature);
            Console.WriteLine("Absolute Humidity:" + mAir.AbsoluteHumidity);
            Console.WriteLine("Relative Humidity:" + mAir.RelativeHumidity);
            Console.WriteLine("Wetbulb Temperature:" + mAir.WetBulbTemperature);
            Console.WriteLine("Specific Volume:" + mAir.SpecificVolume);
            Console.WriteLine("Enthalpy:" + mAir.Enthalpy);
            Console.WriteLine("Atmospheric Pressure:" + mAir.AtmosphericPressure);

            Console.Read();
        }
    }
}
```

Fig.2.1 The sample code to edit the values of the MoistAir property

Figure 2.2 shows the result of the sample code 2.1.

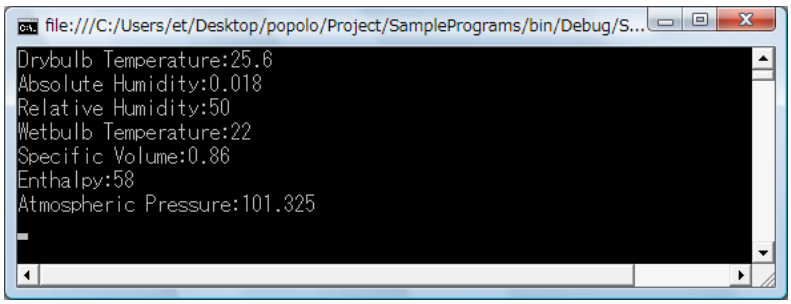

Fig.2.2 Result of the sample code 2.1

2.2.2 How to calculate thermodynamic properties of the moist air

An enumeration type “Property” which expresses the moist air properties is defined in MoistAir class. Table 2.3 shows a list of MoistAir.Property enumerator.

For each atmospheric pressure, two known properties allow determination of all other properties of the moist air. To calculate thermodynamic properties from two known properties, user can use static methods shown in table 2.4. The static methods are named as “GetAirStateFromXXYY”, where XX and YY represent two known properties. DB is DryBulb temperature, AH is Absolute Humidity, RH is Relative Humidity, EN is ENthalpy, WB is WetBulb temperature, SV is Specific Volume.

Table 2.3 A list of MoistAir.Property enumerator

Name	Meaning
DryBulbTemperature	Drybulb Temperature[°C]
WetBulbTemperature	Wetbulb Temperature [°C]
AbsoluteHumidity	Absolute Humidity [kg/kg(DA)]
RelativeHumidity	Relative Humidity [%]
Enthalpy	Enthalpy [kJ/kg]
WaterPartialPressure	Water Partial Pressure [kPa]
SpecificVolume	Specific Volume [m^3/kg]
SaturatedTemperature	Saturated Temperature [°C]

Table 2.4 Static methods to calculate thermodynamic properties of the moist air

Name	General function
GetAirStateFromAHEN	Calculate thermodynamic properties from absolute humidity [kg/kg] and enthalpy [kJ/kg]
GetAirStateFromAHRH	Calculate thermodynamic properties from absolute humidity [kg/kg] and relative humidity [%]
GetAirStateFromAHSV	Calculate thermodynamic properties from absolute humidity [kg/kg] and specific volume [m^3/kg]
GetAirStateFromDBAH	Calculate thermodynamic properties from dry bulb temperature [°C] and absolute humidity [kg/kg]
GetAirStateFromDBEN	Calculate thermodynamic properties from dry bulb temperature [°C] and enthalpy [kJ/kg]
GetAirStateFromDBRH	Calculate thermodynamic properties from dry bulb temperature [°C] and relative humidity [%]
GetAirStateFromDBSV	Calculate thermodynamic properties from dry bulb temperature [°C] and specific volume [m^3/kg]
GetAirStateFromDBWB	Calculate thermodynamic properties from dry bulb temperature [°C] and wet bulb temperature [°C]
GetAirStateFromRHEN	Calculate thermodynamic properties from relative humidity [%] and enthalpy [kJ/kg]
GetAirStateFromRHSV	Calculate thermodynamic properties from relative humidity [%] and specific volume [m^3/kg]
GetAirStateFromWBAH	Calculate thermodynamic properties from wet bulb temperature [°C] and absolute humidity [kg/kg]
GetAirStateFromWBEN	Calculate thermodynamic properties from wet bulb temperature [°C] and enthalpy [kJ/kg]
GetAirStateFromWBRH	Calculate thermodynamic properties from wet bulb temperature [°C] and relative humidity [%]
GetAirStateFromWBSV	Calculate thermodynamic properties from wet bulb temperature [°C] and specific volume [m^3/kg]

Each method is over loaded and have 4 combinations of parameters. Table 2.5 shows return value and 4 combinations of parameters.

The first and the second method calculates all the thermodynamic properties of the moist air from given two state. It returns MoistAir object whose Properties are correctly set upped. The second method request a value of atmospheric pressure as third parameter. By contrast, the first method suppose the value of the atmospheric pressure as 101.325 kPa.

The third and the fourth method calculates value of only one specific property of the moist air. They need shorter calculating time than the first and the second method.

Table 2.5 Return value and 4 combinations of parameters

No.	Return value	param 1	param 2	param 3	param 4
1	MoistAir object	value of specific property 1	value of specific property 2	N/A	N/A
2	MoistAir object	value of specific property 1	value of specific property 2	atmospheric pressure [kPa]	N/A
3	value of specific property (double)	value of specific property 1	value of specific property 2	MoistAir.Property	N/A
4	value of specific property (double)	value of specific property 1	value of specific property 2	MoistAir.Property	atmospheric pressure [kPa]

Figure 2.3 shows the sample code which calculates the values of the properties of MoistAir. In the line 14, using “GetAirStateFromDBAH” method, state of the moist air is calculated from the value of dry bulb temperature and absolute humidity. The value of the moist air are written to the standard output stream. In the line 27, “MoistAir.Property” enumerator is given as third parameter of the GetAirStateFromDBEN method. Therefore, only the value of absolute humidity is calculated.

Figure 2.4 shows the result.

```
1  using System;
2  using Popolo.ThermophysicalProperty;
3
4  namespace ConsoleApplication
5  {
6     class Program
7     {
8        static void Main(string[] args)
9        {
10          //Create an instance of the MoistAIr class
11          MoistAir mAir;
12
13          //Calculate state of the moist air from given two properties (DB 25 °C, AH 0.012 kg/kg)
14          mAir = MoistAir.GetAirStateFromDBAH(25, 0.012);
15
16          //Write value of the moist air to standard output stream.
17          Console.WriteLine("Dry bulb temperature:" + mAir.DryBulbTemperature.ToString("F1"));
18          Console.WriteLine("Absolute humidity:" + mAir.AbsoluteHumidity.ToString("F3"));
19          Console.WriteLine("Relative humidity:" + mAir.RelativeHumidity.ToString("F1"));
20          Console.WriteLine("Wet bulb temperature:" + mAir.WetBulbTemperature.ToString("F1"));
21          Console.WriteLine("Specific volume:" + mAir.SpecificVolume.ToString("F3"));
22          Console.WriteLine("Enthalpy:" + mAir.Enthalpy.ToString("F1"));
23          Console.WriteLine("Atmospheric pressure:" + mAir.AtmosphericPressure.ToString("F1"));
24          Console.WriteLine();
25
26          //Calculate relative humidity from dry bulb temperature and enthalpy.
27          double rHumid = MoistAir.GetAirStateFromDBEN(25, 58, MoistAir.Property.RelativeHumidity);
28          Console.WriteLine("Relative Humidity:" + rHumid.ToString("F1"));
29
30          Console.Read();
31       }
32    }
33 }
```

Fig.2.3 The sample code calculating the values of the properties of moist air

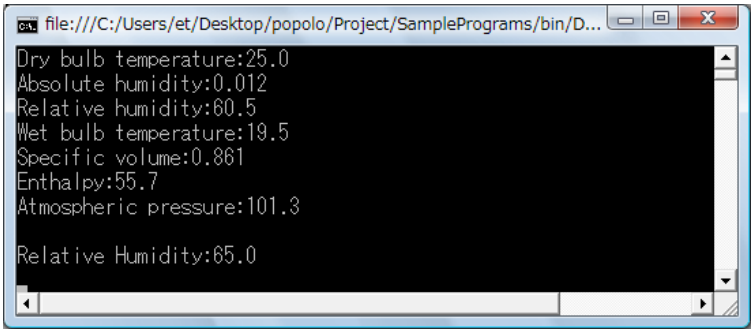

Fig.2.4 Result of the sample code

2.2.3 Calculation of saturated moist air

Table 2.6 shows the lists of static method to calculate saturated moist air state. They take two or three arguments. The first argument is value of a specific property of the moist air (ex. dry-bulb temperature, absolute humidity). The second argument is "MoistAir.Property", the enumerating object. The third object is value of atmospheric pressure.

Table 2.6 The lists of static method to calculate saturated moist air state

Name	General function
GetSaturatedDrybulbTemperature	Calculate saturated dry-bulb temperature from one of value of moist air property.
GetSaturatedAbsoluteHumidity	Calculate saturated absolute humidity from one of value of moist air property.
GetSaturatedEnthalpy	Calculate saturated enthalpy from one of value of moist air property.
GetSaturatedVaporPressure	Calculate saturated vapor pressure from one of value of moist air property.

2.2.4 Immutable interface for MoistAir class

As described above, MoistAir object has double type properties which contains values of the moist air states (ex. dry-bulb temperature, absolute humidity). Therfore, MoistAir object could be an argument or return value of some other method. For example, when we calculate an air handling unit, a cooling coil need state of inlet moist air which is outlet state of a fan. In this case, a MoistAir object will be given from a Fan object to a CoolingCoil object. Since the cooling coil has no ability to alter the value of inlet moist air state, the access for properties of MoistAir objects should be denied from the CoolingCoil object.

To enable this access control, ImmutableMoistAir interface is provided. ImmutableMoistAir interface has exactly same properties as MoistAir class, but only get accessors are defined. Therefore, we can't set values to ImmutableMoistAir properties. They are read-only properties. MoistAir object can be treated as ImmutableMoistAir object, since it implements ImmutableMoistAir interface. Figure 2.5 is the UML diagrams.

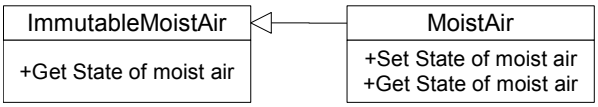

Fig.2.5 UML diagram of MoistAir class and ImmutableMoistAir interface

Figure 2.6 is a sample code which uses ImmutableMoistAir interface.

```
1  using System;
2
3  using Popolo.ThermophysicalProperty;
4
5  namespace ConsoleApplication
6  {
7     class Program
8     {
9        static void Main(string[] args)
10       {
11          //Create an instance of ImmutableMoistAir
12          ImmutableMoistAir mAir;
13
14          //Create MoistAir object and set it to ImmutableMoistAir object
15          mAir = MoistAir.GetAirStateFromDBAH(25, 0.012);
16
17          //Compile error occurs since ImmutableMoistAir interface doesn't have set accessors
18          //mAir.DryBulbTemperature = 27;
19          //mAir.Enthalpy = 56;
20
21       }
22    }
23 }
```

Fig. 2.6 Sample code which uses ImmutableMoistAir interface.

2.2.5 Other methods defined in MoistAIr class

1) BlendAir method

To calculate blended moist air state, BlendAir method is defined. Table 2.7 shows arguments of BlendAir method. Figure 2.7 is a sample code which uses BlendAir method. It creates two ImmutableMoistAir objects and blend them (mixing ratio is 3:7).

Table 2.7 Arguments of BlendAir method

No.	Argument 1	Argument 2	Argument 3	Argument 4
1	Blended moist air 1 (ImmutableMoistAir)	Blended moist air 2 (ImmutableMoistAir)	Blend rate of moist air 1 (double)	Blend rate of moist air 2 (double)
2	List of blended moist air (ImmutableMoistAir[ ])	Blend rate of moist air (double[ ])	-	-
3	Dry-bulb temperature list of blended moist air (double[ ])	Absolute humidity list of blended moist air (double[ ])	-	-

```
using System;

using Popolo.ThermophysicalProperty;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create an instance of MoistAir class
            ImmutableMoistAir mAir1, mAir2, blendAir;

            //Calculate moist air state (DB 25C, AH 0.012kg/kgDA)
            mAir1 = MoistAir.GetAirStateFromDBAH(25, 0.012);
            //Calculate moist air state (DB 30C, AH 0.014kg/kgDA)
            mAir2 = MoistAir.GetAirStateFromDBAH(30, 0.014);

            //Blend moist air. (Blend rate is 3:7)
            blendAir = MoistAir.BlendAir(mAir1, mAir2, 0.3, 0.7);

            Console.WriteLine("Dry bulb temperature:" + blendAir.DryBulbTemperature);
            Console.WriteLine("Absolute humidty:" + blendAir.AbsoluteHumidity);
            Console.WriteLine("Relative humidity:" + blendAir.RelativeHumidity);
            Console.WriteLine("Wet bulb temperature:" + blendAir.WetBulbTemperature);
            Console.WriteLine("Specific volume:" + blendAir.SpecificVolume);
            Console.WriteLine("Enthalpy:" + blendAir.Enthalpy);
        }
    }
}
```

Fig. 2.7 Sample code which uses BlendAir method

2) GetDynamicViscosity method

Method to calculate dynamic viscosity [m^2/s] of moist air. An argument is dry bulb temperature.

3) GetSpecificHeat method

Method to calculate specific heat [kJ/(kg K)] of moist air. An argument is absolute humidity.

4) GetThermalConductivity method

Method to calculate thermal conductivity [W/(m K)] of moist air. An argument is dry bulb temperature.

5) GetWaterVaporPressure method

Method to calculate water vapor pressure [kPa] of moist air. An argument is absolute humidity (and atmospheric pressure).

Chapter 3 Calculating Circuit Network

3.1 Name space for calculating circuit network

The classes which calculate a circuit network belong to “Popolo.CircuitNetwork” namespace. Table 3.1 shows principal members defined in “Popolo.CircuitNetwork” namespace.

Table 3.1 Principal members defined in “Popolo.CircuitNetwork” namespace

Name	Type	General function
Channel	class	A class which expresses a channel.
Circuit	class	A class which expresses a circuit. It consists of Channel and Node object.
CircuitSolver	class	A class which solve a Circuit object.
ImmutableChannel	interface	Read only interface for Channel class.
ImmutableCircuit	interface	Read only interface for Circuit class.
ImmutableNode	interface	Read only interface for Node class.
Node	class	A class which expresses a node.

A potential at each node in circuit network can be expressed in equation 3.1[3]. To solve a circuit network, user should make a Circuit object with a Channel and a Node object, which expresses equation 3.1. Then, solve the Circuit object with a CircuitSolver object.

$$m \cdot \frac{dp}{dt} = \sum_{i=0}^{N} \frac{1}{R_i} \cdot (p_i - p)^{\frac{1}{\eta}} + G \tag{3.1}$$

m : Capacity of node p : Potential of node N : Number of nodes which are connected to target node
R_i : Resistance between nodes G : Energy flow from the outside to node η : resistive index ($1 \leq \eta \leq 2$)

3.2 General descriptions of classes in CircuitNetwork namespace

3.2.1 Node class

Node is a class which expresses a node. A principal parameters of node are potential and capacity. A variable p and m in equation 3.1 represent potential and capacity. If the capacity is very small, value of the potential changes rapidly with the energy flow. User can initialize these value in constructor. For example, to make Node object whose capacity and potential are 5 and 10, user should write code like below. The first argument is a name of the node, the second is value of the capacity, and the third is value of the potential.

```
Node sampleNode = new Node("SampleNode", 5, 10)
```

To get the list of the channels which are connected to the node, use GetChannels method. To get the total energy flow to node, use GetTotalFlow method. If the returned value of GetTotalFlow method is less than 0, energy flows to outside of the node.

If the node is the boundary node (the node whose potential should be kept constant value), set value of the IsBoundaryNode property to true. To make constant energy flow to the node, set the energy flow to the ExternalFlow property. If the ExternalFlow is positive value, it means that energy flows from node to outside. If negative, energy flows from outside to node. A variable G in equation 3.1 represent the ExternalFlow.

These properties are used when user make boundary condition. Fig.3.1 shows how to set the boundary conditions.

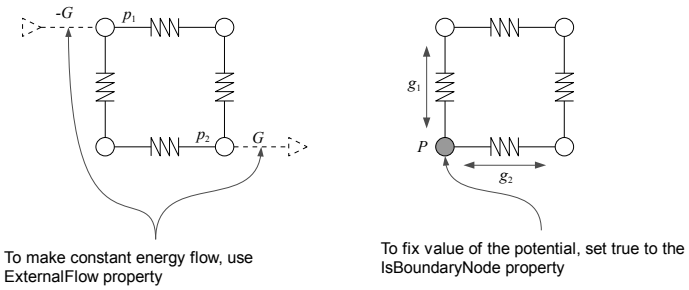

Figure 3.1 How to set the boundary conditions

As same as the MoistAir class, an immutable interface (ImmutableNode) for the Node class is defined. The Node class implements the ImmutableNode interface.

3.2.2 Channel class

Channel is a class which expresses a channel. It connects two node objects and represent their connection weight with resistance and resistive index. A variable R and η in equation 3.1 represent resistance and resistive index. User can initialize these value in constructor. For example, to make Channel object whose resistance and resistive index are 3 and 1.2, user should write code like below. The first argument is a name of the channel, the second is value of the resistance, and the third is value of the resistive index.

```
Channel sampleChannel = new Channel("SampleChannel", 3, 1.2);
```

To connect two nodes with Channel object, use Connect method.

```
sampleChannel.Connect(node1, node2);
```

The first and the second arguments are Node objects to connect.

To get the energy flow in the channel, use GetFlow method. Figure 3.2 is a sample program which calculate energy flow between two nodes. The potential of the first and the second node are 10 and 0. The resistance and resistive index of channel are 2 and 1.2. Energy flow of this channel is $1 / 2 \times (10 - 0)^{(1/1.2)} = 3.4$.

As same as the Node class, an immutable interface (ImmutableChannel) for the Channel class is defined. The Channel class implements the ImmutableChannel interface.

```
1  /// <summary>Circuit test 1</summary>
2  /// <remarks>Calculating energy flow between two nodes</remarks>
3  private static void circuitTest1()
4  {
5      //Create new instance of Node class.
6      Node node1 = new Node("SampleNode1", 0, 10);
7      Node node2 = new Node("SampleNode2", 0, 0);
8
9      //Create new instance of Channel class and connect nodes.
10     Channel channel = new Channel("SampleChannel", 2, 1.2);
11     channel.Connect(node1, node2);
12
13     //Calculate energy flow.
14     double flow = channel.GetFlow();
15
16     Console.WriteLine("Energy flow is : " + flow.ToString("F2"));
17     Console.Read();
18 }
```

Fig.3.2 A sample program which calculate energy flow between two nodes

3.2.3 Circuit class

A circuit consists of nodes and channels. To define the circuit, Add Node objects to Circuit object and connect them with Channel objects. To add Node objects to a Circuit object, use AddNode method.

```
ImmutableNode iNode = sampleCircuit.AddNode(sampleNode);
```

The return value of AddNode method is the ImmutableNode object which was added to the Circuit object by AddNode method. To remove Node objects from a Circuit object, use RemoveNode method.

```
sampleCircuit.RemoveNode(iNode);
```

The first argument is the removing ImmutableNode object.
To connect nodes, use ConnectNodes method.

```
ImmutableChannel iChannel = sampleCircuit.ConnectNodes(iNode1, iNode2, sampleChannel);
```

The first and the second arguments are ImmutableNode objects, and the third argument is a Channel object which connects two nodes. The two nodes should be added before calling ConnectNodes method. The returned value of ConnectNodes method is the ImmutableChannel object which connects two nodes.
To disconnect nodes, use DisconnectNodes method.

```
sampleCircuit.DisconnectNodes(iChannel);
```

The first argument is the Channel object which connects nodes.

The boundary conditions described at section 3.2.1 could be also set in Circuit class. Table 3.2 shows lists of the method to set boundary conditions which defined in Circuit class.

Table 3.2 The method to set boundary conditions

Name	Function	Argument 1	Argument 2
SetPotential	Set initial value of potential to node	Value of the potential (double)	Target node (ImmutableNode)
SetExternalFlow	Set energy flow from node to outside	Value of the energy flow (double)	Target node (ImmutableNode)
SetBoundaryNode	Set whether node is boundary node or not.	Whether node is boundary node or not (bool)	Target node (ImmutableNode)

3.2.4 CircuitSolver class

The CircuitSolver class has a function to solve a circuit network. It takes a Circuit object as an argument of constructor.

```
CircuitSolver cSolver = new CircuitSolver(sampleCircuit);
```

To solve a circuit network, use Solve method.

```
cSolver.Solve();
```

All the value of energy flows and potentials are updated when the Solve method is called. There are two kinds of circuit network, circuit network who has a capacity or not. The capacity is a variable m in equation 3.1. If all the capacities in a network are equal to 0, the network is a static network. In contrast, a network whose capacities takes positive value, is a dynamic network. If a network is a static network, the value of potentials or energy flows will not change unless boundary conditions change. If a network is a dynamic network, the value of potentials or energy flows depends on time. Therefore, user should set time step when a network is a dynamic network. To set time step, use TimeStep property as below. In this case, time step is set to 10 seconds.

```
cSolver.TimeStep = 10;
```

3.3 Sample programs calculating circuit networks

In this section, two sample programs which calculate circuit networks are given. One is a static network which represent a water pipe network. The other is a dynamic network which represent heat flow through a wall.

3.3.1 Calculating a water pipe network

Figure 3.3 shows a water pipe network to solve[†1)]. The network has three nodes (1, 2 and 3) and four channels (A, B, C and D). The resistance of each channel are shown in figure (R_A ~ R_D).

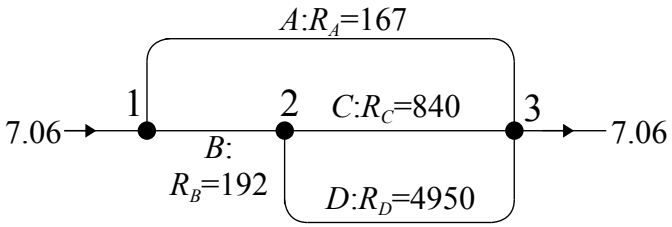

Fig.3.3 A water pipe network to solve

Figure 3.4 is a sample program which solve this network.

New instance of Circuit class is created in the line 5. Three nodes are added in the line 8~10. Since the capacities of these nodes are 0, this network is a static network. The node 1 and the node 2 is connected to outside and water flow rate is 7.06. This external water flow is set in the line 12 and 13.

The channels are created in the line 16 to 19. Since this is a water network, value of the resistive index is 2. In line 22 to 25, the nodes are connected by the channels.

In the line 29, created water flow network is solved by CircuitSolver object. All the potentials and water flow are calculated and written to the standard output stream. Figure 3.5 shows the result of the program.

```
1  /// <summary>Circuit test 2</summary>
2  /// <remarks>Calculating water pipe network</remarks>
3  private static void circuitTest2()
4  {
5      Circuit circuit = new Circuit("Circuit network of water pipe");
6
7      //Add nodes to circuit network
8      ImmutableNode node1 = circuit.AddNode(new Node("1", 0, 0));
9      ImmutableNode node2 = circuit.AddNode(new Node("2", 0, 0));
10     ImmutableNode node3 = circuit.AddNode(new Node("3", 0, 0));
11     //Set external water flow
12     circuit.SetExternalFlow(-7.06, node1);
13     circuit.SetExternalFlow(7.06, node3);
14
15     //Create channels
16     Channel chA = new Channel("A", 167, 2);
17     Channel chB = new Channel("B", 192, 2);
18     Channel chC = new Channel("C", 840, 2);
19     Channel chD = new Channel("D", 4950, 2);
20
21     //Connect nodes with channels
22     ImmutableChannel channelA = circuit.ConnectNodes(node1, node3, chA);
23     ImmutableChannel channelB = circuit.ConnectNodes(node1, node2, chB);
24     ImmutableChannel channelC = circuit.ConnectNodes(node2, node3, chC);
25     ImmutableChannel channelD = circuit.ConnectNodes(node2, node3, chD);
26
27     //Create solver
28     CircuitSolver cSolver = new CircuitSolver(circuit);
29     cSolver.Solve();
30     Console.WriteLine("Water flow A is " + channelA.GetFlow().ToString("F2"));
31     Console.WriteLine("Water flow B is " + channelB.GetFlow().ToString("F2"));
32     Console.WriteLine("Water flow C is " + channelC.GetFlow().ToString("F2"));
33     Console.WriteLine("Water flow D is " + channelD.GetFlow().ToString("F2"));
34     Console.Read();
35 }
```

Fig.3.4 A sample program which solve the water flow network

†1) 本問題の手計算による解法については建築設備基礎（著 木村建一）を参照して下さい。

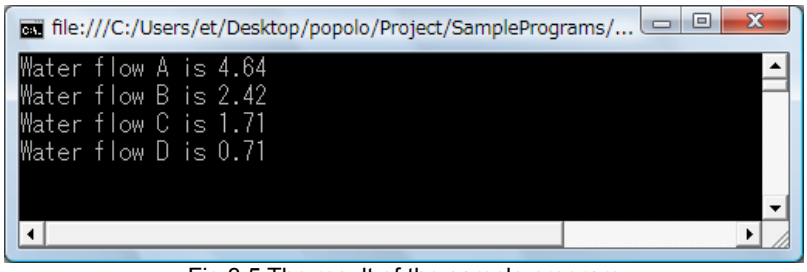

Fig.3.5 The result of the sample program

3.3.2 Calculating heat flow through a wall

Figure 3.6 is a wall to calculate a heat transfer. The wall has four layers; plywood, concrete, air gap and rock wool. The temperatures of Room 1 and Room 2 take constant value 20°C and 10°C. The temperatures of each layer are treated as potentials of node in this case.

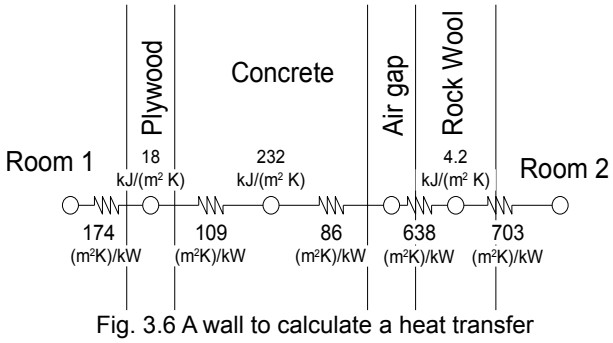

Fig. 3.6 A wall to calculate a heat transfer

Figure 3.7 is a sample program which solve the heat transfer of this wall.

In the line 8 to 14, new instances of Node class are created. In this case, each nodes has heat capacity. Since the room temperatures take constant value, edge nodes is set to boundary nodes in line 17 and 18.

As same as water pipe network, a CircuitSolver object is used to solve the network. In this case, time step should be set since the network has heat capacity and is a dynamic network. In the line 32, TimeStep is set to 3600 seconds (= 1 hour). Update method is called 24 times. The temperatures (potentials) of layers are written to the standard output stream in each iteration.

Figure 3.8 is a wall temperature distribution made from the result of the program.

```
/// <summary>Circuit test 3</summary>
/// <remarks>Calculating heat transfer through a wall</remarks>
private static void circuitTest3()
{
    Circuit circuit = new Circuit("Heat transfer network through wall");

    //Add nodes to circuit network
    ImmutableNode[] nodes = new ImmutableNode[6];
    nodes[0] = circuit.AddNode(new Node("Room 1", 0));
    nodes[1] = circuit.AddNode(new Node("Plywood", 17.9));
    nodes[2] = circuit.AddNode(new Node("Concrete", 232));
    nodes[3] = circuit.AddNode(new Node("Air gap", 0));
    nodes[4] = circuit.AddNode(new Node("Rock wool", 4.2));
    nodes[5] = circuit.AddNode(new Node("Room 2", 0));

    //Set boundary conditions (Room air temperatures).
    circuit.SetBoundaryNode(true, nodes[0]);
    circuit.SetBoundaryNode(true, nodes[5]);
    //Set air temperatures.
    circuit.SetPotential(20, nodes[0]);
    circuit.SetPotential(10, nodes[5]);
    for (int i = 1; i < 5; i++) circuit.SetPotential(10, nodes[i]); //Initialize wall temperatures to 10 C.

    //Connect nodes.
    ImmutableChannel channel01 = circuit.ConnectNodes(nodes[0], nodes[1], new Channel("Room 1-Plywood", 174, 1));
    ImmutableChannel channel12 = circuit.ConnectNodes(nodes[1], nodes[2], new Channel("Plywood-Concrete", 109, 1));
    ImmutableChannel channel34 = circuit.ConnectNodes(nodes[2], nodes[3], new Channel("Concrete-Air gap", 86, 1));
    ImmutableChannel channel45 = circuit.ConnectNodes(nodes[3], nodes[4], new Channel("Air gap-Rock wook", 638, 1));
    ImmutableChannel channel56 = circuit.ConnectNodes(nodes[4], nodes[5], new Channel("Rock wool-Room 2", 703, 1));

    CircuitSolver cSolver = new CircuitSolver(circuit);
    cSolver.TimeStep = 3600;

    for (int i = 0; i < nodes.Length; i++) Console.Write(nodes[i].Name + "  ");
    Console.WriteLine();
    for (int i = 0; i < 24; i++)
    {
        cSolver.Solve();
        Console.Write((i + 1) + "H : ");
        for (int j = 0; j < nodes.Length; j++) Console.Write(nodes[j].Potential.ToString("F1") + "  ");
        Console.WriteLine();
    }
    Console.Read();
}
```

Fig.3.7 A sample program which solve the heat transfer of the wall.

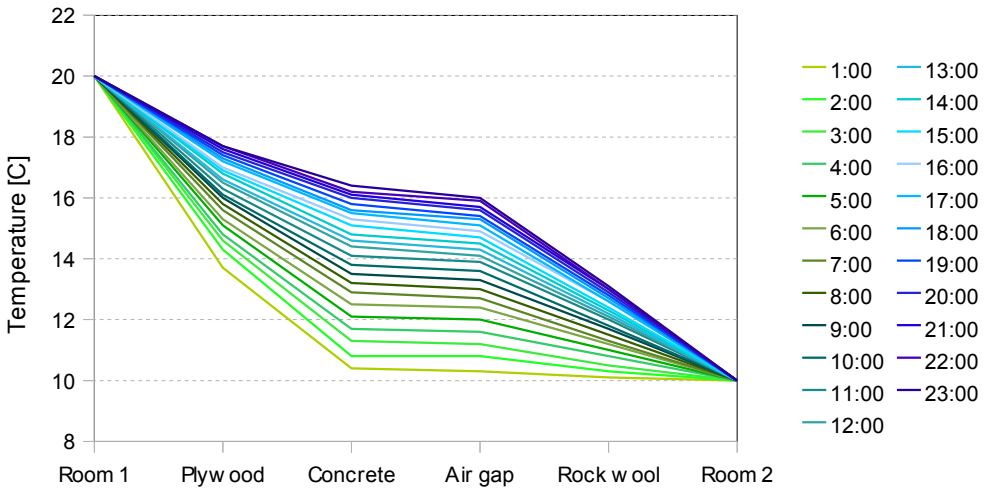

Fig.3.8 Wall temperature distribution

Chapter 4 Calculating Thermal Comfort

4.1 熱的快適性の計算に関するクラスが属する名前空間

熱的快適性に関する計算を行うためのクラスは「Popolo.Utility.ThermalConfort」名前空間に属しています。表 4.1 に Popolo.Utility.ThermalConfort 名前空間に属するメンバを示します。

表 4.1 Popolo.Utility.ThermalConfort 名前空間に属する主要メンバ

メンバ名称	メンバ種類	メンバの概要
PMVCalculator	static class	PMV 値を計算するためのクラス
SETStarCalculator	static class	SET*値を計算するためのクラス
HumanBody	class	非定常な人体内外の熱移動を表現するクラス
BodyPart	class	人体の部位を表現するクラス

4.2 各クラスの概要

4.2.1 PMVCalculator class

温熱環境に影響を与える 6 要素に基づいて、その条件で多くの人間が感じる温冷感を数値で表現したものが予想平均温冷感申告 (PMV : Predicted Mean Vote) です。また、PMV 値を利用して、予想される不満足者の割合である、予想不満足者率 (PPD : Predicted Percentage of Dissatisfied) を計算することが可能です。これらはいずれも P.O.Fanger 教授によって提唱された概念です。

PMV および PPD を計算するためには PMVCalculator クラスを利用します。PMVCalculator クラスの主要メンバを表 4.2 に示します。

表 4.2 PMVCalculator クラス主要メンバ

メンバ名称	メンバ種類	メンバの概要
Tasks	列挙型	仕事の種類を示す列挙型
GetMet	メソッド	仕事に対する代謝量を計算する
GetPMVFromPPD	メソッド	PPD 値をもとに PMV 値を計算する
GetPPDFromPMV	メソッド	PMV 値をもとに PPD 値を計算する
TryCalculateDryBulbTemperature	メソッド	PMV とその他の温熱条件から乾球温度を逆算する
TryCalculateHeatLossFromBody	メソッド	人体からの熱損失を計算する
TryCalculatePMV	メソッド	温熱 6 要素に基づいて PMV 値を計算する
TryCalculateRelativeHumidity	メソッド	PMV とその他の温熱条件から相対湿度を逆算する

1) Tasks

Tasks は各種の仕事を示す列挙型です。休息、オフィス作業、運動など、ASHRAE Standard 55 で例示された各種の仕事が定義されています。

2) GetMet

仕事に対する人間の代謝量[met]を返すメソッドです。列挙型の Tasks を引数にとります。

3) GetPMVFromPPD

PPD 値をもとに PMV 値を計算するメソッドです。PPD の曲線は図 4.1 に示すように左右に対称で、1 つの PPD 値に対して 2 つの PMV 値が考えられますが、本関数で計算されるのは正の値です。PPD 曲線は左右対称なので、符号を逆転させれば負の範囲の解が得られます。

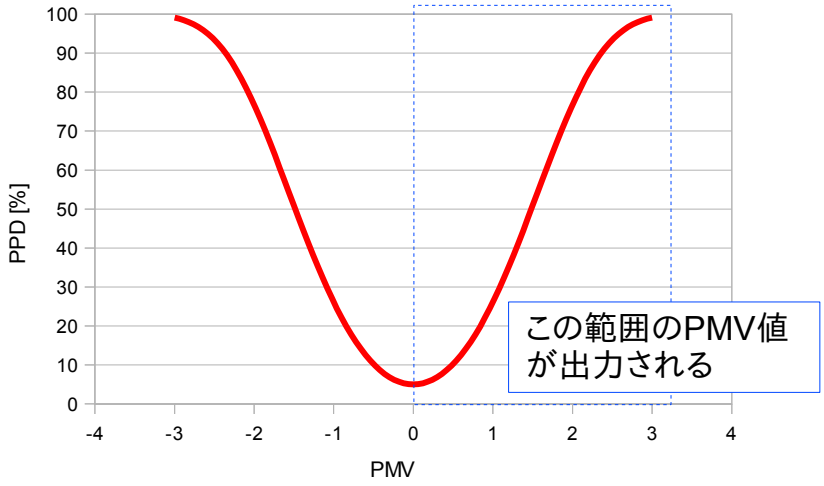

図 4.1　PMV-PPD 曲線

4) GetPPDFromPMV

PMV 値をもとに PPD 値を計算するメソッドです。引数は PMV 値のみです。

5) TryCalculateDryBulbTemperature

PMV とその他の温熱条件から乾球温度を逆算するメソッドです。表 4.3 に引数を示します。第 2~7 引数で与えた条件のもと、第 1 引数で与えた PMV 値を達成するために必要となる乾球温度を計算し、第 8 引数に出力します。プログラム例を図 4.2 に示します。出力は 29.2 °C 程度の値となるはずです。

表 4.3　TryCalculateDryBulbTemperature の引数一覧

番号	内容	番号	内容
1	PMV 値[-]	5	着衣量 [clo]
2	平均放射温度 [°C]	6	代謝量 [met]
3	気流速度 [m/s]	7	外部仕事量 [met]
4	相対湿度 [%]	8	出力：乾球温度 [°C]

```
/// <summary>PMVテスト1</summary>
private static void pmvTest1()
{
    double dbt;
    PMVCalculator.TryCalculateDryBulbTemperature(1.2, 25.6, 50, 0.1, 0.8, 1.2, 0, out dbt);
    Console.WriteLine(dbt);
    Console.Read();
}
```

図 4.2　PMV プログラム例 1

6) TryCalculateHeatLossFromBody

人体からの熱損失量を計算するメソッドです。表 4.4 に引数一覧を示します。第 1~7 引数に温熱 6 要素を設定すると、設定条件下における人体の熱損失量が第 8 引数に出力されます。この熱損失量は PMV 値を計算するための基礎となる値です。

表 4.4 TryCalculateHeatLossFromBody の引数一覧

番号	内容	番号	内容
1	乾球温度 [°C]	5	着衣量 [clo]
2	平均放射温度 [°C]	6	代謝量 [met]
3	気流速度 [m/s]	7	外部仕事量 [met]
4	相対湿度 [%]	8	出力：熱損失量 [W]

7) TryCalculatePMV

TryCalculatePMV は、温熱 6 要素に基づいて PMV 値を計算するメソッドです。表 4.5 に引数一覧を示します。オーバーロードされており、表 4.6 に示す引数でも呼び出すことが可能です。

表 4.5 TryCalculatePMV の引数一覧 1

番号	内容	番号	内容
1	乾球温度 [°C]	5	着衣量 [clo]
2	平均放射温度 [°C]	6	代謝量 [met]
3	気流速度 [m/s]	7	外部仕事量 [met]
4	相対湿度 [%]	8	出力：PMV [-]

表 4.6 TryCalculatePMV の引数一覧 2

番号	内容	番号	内容
1	代謝量 [met]	3	熱損失量 [W]
2	外部仕事量 [met]	4	出力：PMV [-]

4.2.2 SETStarCalculator クラス

ET*および SET*は Gagge らの理論に基づいて導出された体感温度です[4)]。

ET*および SET*を計算するためには SETStarCalculator クラスの TryCalculateSET メソッドを利用します。引数一覧を表 4.7 に示します。第 1 引数から第 10 引数までが入力であり、第 11 および第 12 引数は出力となります。

表 4.7 TryCalculateSET の引数一覧 1

番号	内容	番号	内容
1	乾球温度 [°C]	7	外部仕事量 [W/m2]
2	平均放射温度 [°C]	8	大気圧 [kPa]
3	気流速度 [m/s]	9	体重 [kg]
4	相対湿度 [%]	10	体表面積 [m2]
5	着衣量 [clo]	11	出力：ET* [-]
6	代謝量 [W/m2]	12	出力：SET* [-]

4.2.3 HumanBody クラス

PMV や SET*という指標は定常かつ均一な温熱環境に関して提案されたものですが、実際の温熱環境は往々にして非定常かつ不均一です。このような温熱環境下に置かれた人体の反応を知るためには、非定常な人体のモデルが必要となります。HumanBody クラスは、田辺ら[†1)]によって提案された人体 17 分割非定常モデルを計算するためのクラスです。また、HumanBody クラスに付随して、人体の一部のみを表現する BodyPart クラスが定義されています。図 4.3 に人体 17 分割非定常モデルの構成を示します。

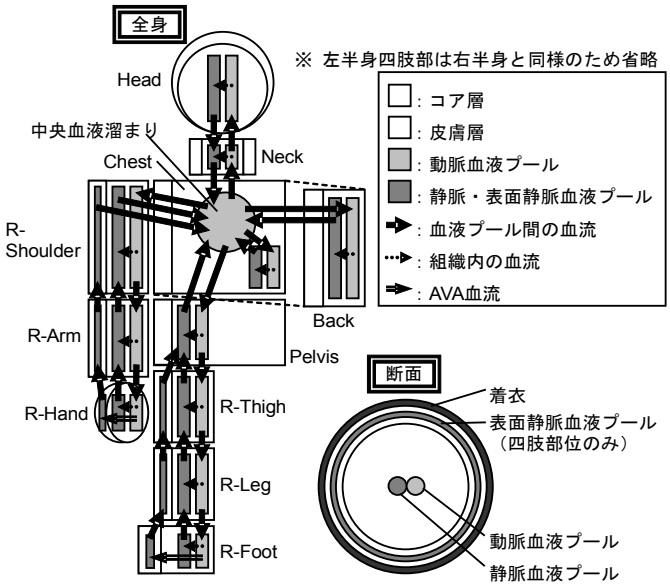

図 4.3　17 分割非定常モデルの構成

HumanBody クラスには、2 種類のコンストラクタが定義されています。引数を持たない場合、人体は標準体躯に初期化されます。標準体躯とは、体重=71.43 kg, 身長=1.72 m, 年齢=25, 心係数=2.58 L/(min m^2), 体脂肪率=15%の男性です。一方、以下に示すようにコンストラクタの引数で指定すれば、体重や身長を任意に設定することもできます。これらの情報はプロパティで取得することが可能です。

```
public HumanBody(double weight, double height, double age, bool isMale,
double cardiacIndexAtRest, double fatPercentage)
```

人体の特定の部位を指定するために、Nodes 列挙型が定義されています。表 4.8 に Nodes 列挙型のメンバ一覧を示します。

†1) S.Tanabe : Development of numerical thermo regulation model JOS for evaluation of thermal comfort, Building and environment に投稿中・・・・

表 4.8　Nodes 列挙型のメンバー一覧

メンバ名称	意味	メンバ名称	意味
Head	頭	LeftLeg	左ふくらはぎ
Neck	首	LeftFoot	左足
Chest	胸	RightShoulder	右肩
Back	背	RightArm	右腕
Pelvis	腰	RightHand	右手
LeftShoulder	左肩	RightThigh	右太股
LeftArm	左腕	RightLeg	右ふくらはぎ
LeftHand	左手	RightFoot	右足
LeftThigh	左太股	-	-

表 4.9 に HumanBody クラスの主要メソッドを示します。

まず、InitializeTemperature メソッドを使用することで、体の温度を初期化することができます。ただし、モデルの初期状態は、作用温度 28.8 °C の均一条件下における定常状態なので、必ずしも体温の初期化をする必要はありません。

Update メソッドを呼ぶと、引数で指定された秒数だけ時間を経過させ、モデルの状態を更新することができます。着衣量や部位ごとの温湿度などの条件設定については後述します。

GetBodyPart メソッドを使用すると、各部位を表す BodyPart オブジェクトを取得することができます。部位の内部の温度（コア層、筋肉層、脂肪層、皮膚層）などに関する情報は BodyPart オブジェクトを操作することで入手することができます。

表 4.9　HumanBody クラスの主要メソッド

名称	内容			
Update	概要	状態を更新する		
	戻り値	-		
	引数 1	経過させる秒数 [sec]	-	-
InitializeTemperature	概要	体の温度を初期化する		
	戻り値	-		
	引数 1	初期化する温度 [°C]	-	-
GetBodyPart	概要	体の部位を取得する		
	戻り値	体の部位 (BodyPart 型)		
	引数 1	体の部位 (Nodes 型)	-	-

表 4.10 に、境界条件設定に関するメソッドを示します。※がついているメソッドについては第 1 引数に Nodes 型を設定することで、部位ごとの指定が可能です。

表 4.10　HumanBody クラス：境界条件設定に関するメソッド

名称	内容			
SetCardiacIndex	概要	心係数[L/(min m^2)]を設定する		
	引数 1	心係数[L/(min m^2)] (double 型)	-	-
SetPosture	概要	姿勢を設定する		
	引数 1	姿勢 (BodyPosture 型)	-	-
SetWorkLoad	概要	仕事量[W/m^2]を設定する		
	引数 1	仕事量[W/m^2] (double 型)	-	-
※SetContactPortionRate	概要	接触面積割合[-]を設定する		
	引数 1	接触面積割合[-] (double 型)	-	-
SetHeatConductance ToMaterial	概要	物体への熱コンダクタンス[W/(m^2 K)]を設定する		
	引数 1	体の部位 (Nodes 型)	引数 2	物体への熱コンダクタンス[W/(m^2 K)]
※SetDrybulbTemperature	概要	周辺空気の乾球温度[°C]を設定する		
	引数 1	周辺空気の乾球温度[°C] (double 型)	-	-
※SetRelativeHumidity	概要	周辺空気の相対湿度[%]を設定する		
	引数 1	周辺空気の相対湿度[%] (double 型)	-	-
※SetMeanRadiant Temperature	概要	平均放射温度[°C]を設定する		
	引数 1	平均放射温度[°C] (double 型)	-	-
※SetMaterialTemperature	概要	接触物体の温度[°C]を設定する		
	引数 1	接触物体の温度[°C] (double 型)	-	-
※SetClothingIndex	概要	着衣量[clo]を設定する		
	引数 1	着衣量[clo] (double 型)	-	-
※SetVelocity	概要	気流速度[m/s]を設定する		
	引数 1	気流速度[m/s] (double 型)	-	-

表 4.11 に状態取得に関するメソッドを示します。これらのメソッドは全身を対象としたものです。部位ごとの詳細な情報を取得するためには GetBodyPart メソッドで部位を取得した後、BodyPart クラスに定義されたメソッドを使用します。

表 4.11　HumanBody クラス：状態取得に関するメソッド

名称	内容	
GetBloodFlow	概要	全血流[L/h]を取得する
	戻り値	全血流[L/h] (double 型)
GetSensibleHeatLossFromSkin	概要	顕熱損失[W]を計算する
	戻り値	顕熱損失[W] (double 型)
GetLatentHeatLossFromSkin	概要	潜熱損失[W]を計算する
	戻り値	潜熱損失[W] (double 型)
GetMetabolicRate	概要	代謝量[W]を取得する (基礎代謝・外部仕事・ふるえを考慮した値)
	戻り値	代謝量[W] (double 型)
GetAverageSkinTemperature	概要	全身の平均皮膚温[°C]を取得する
	戻り値	全身の平均皮膚温[°C] (double 型)

情報取得に関する BodyPart クラスのメソッド一覧を表 4.12 に示します。BodyPart クラス内部の細かな位置を指定するためには Segments 列挙型を使用します。表 4.13 に Segments 列挙型のメンバを示します。

表 4.12　BodyPart クラスの主要メソッド

名称	内容			
GetSensibleHeatLoss	概要	皮膚からの顕熱損失[W]を計算する		
	戻り値	皮膚からの顕熱損失[W] (double 型)		
	引数 1		-	-
GetHeatCapacity	概要	各部位の熱容量[Wh/K]を取得する		
	戻り値	各部位の熱容量[Wh/K] (double 型)		
	引数 1	部位 (Segments 型)	-	-
GetTemperature	概要	各部位の温度[°C]を取得する		
	戻り値	各部位の温度[°C] (double 型)		
	引数 1	部位 (Segments 型)	-	-
GetHeatConductance	概要	部位間の熱コンダクタンス[W/K]を取得する (接触がなければ 0)		
	戻り値	部位間の熱コンダクタンス[W/K]		
	引数 1	部位 1 (Segments 型)	引数 2	部位 2 (Segments 型)
GetMetabolicRate	概要	各部位の代謝量[W]を取得する		
	戻り値	各部位の代謝量[W] (double 型)		
	引数 1	部位 (Segments 型)	-	-
GetBloodFlow	概要	各部位の血流量[L/h]を取得する		
	戻り値	各部位の血流量[L/h] (double 型)		
	引数 1	部位 (Segments 型)	-	-
GetHeatTransfer	概要	部位 1 から部位 2 への熱移動量[W]を計算する		
	戻り値	部位 1 から部位 2 への熱移動量[W] (double 型)		
	引数 1	部位 1 (Segments 型)	引数 2	部位 2 (Segments 型)

表 4.13　Segments 列挙型のメンバー覧

メンバ名称	意味	メンバ名称	意味
Core	コア層	Artery	動脈
Muscle	筋肉層	SuperficialVein	表在静脈
Fat	脂肪層	DeepVein	深部動脈
Skin	皮膚層	AVA	AVA 血流

図 4.4 に人体モデルのプログラム例を示します。8 行目で人体モデルの初期化を行っています。ここでは体重 70kg、身長 1.6m、年齢 35 歳、心係数 2.58、体脂肪率 20%の女性としています。引数を設定しない場合には標準体躯の人体モデルとなります。

10~19 行目では境界条件を設定しています。これらは各部位の境界条件を一括して指定する場合の書き方で、特定の部位の境界条件のみを設定したい場合には、22 行目に示すように体の部位を第一引数で指定します。ここでは右手の先だけ 20°C に冷やしてみました。

26 行目以降で、時間を経過させています。28 行目の Update メソッドで 120 秒の時間を経過させています。16 回の繰り返しを行うため、全部で 30 分、時間を経過させることになります。もちろん、この間に境界条件に変化が合っても問題ありません。

29 および 30 行目では、左右の肩部を取得しています。32~35 行目では、左右の肩オブジェクトを利用して、各肩のコアの温度と皮膚の温度を取得し、コンソールに書き出しています。

プログラムを実行した結果を図 4.5 に示します。時間の経過とともに体の温度が上昇して往く様子が確認できます。また、右手は 20°C に冷やされているため、その影響で右肩も冷え、左肩よりもやや温度が低くなっていることが確認できます。

```
/// <summary>人体モデル計算の例</summary>
private static void humanBodyTest()
{
    //標準体躯の場合は引数不要
    //HumanBody body = new HumanBody();

    //人体モデルを作成:体重70kg,身長1.6m,年齢35歳,女性,心係数2.58,体脂肪率20%
    HumanBody body = new HumanBody(70, 1.6, 35, false, 2.58, 20);

    //着衣量[clo]を設定
    body.SetClothingIndex(0);
    //乾球温度[C]を設定
    body.SetDrybulbTemperature(42);
    //放射温度[C]を設定
    body.SetMeanRadiantTemperature(42);
    //気流速度[m/s]を設定
    body.SetVelocity(1.0);
    //相対湿度[%]を設定
    body.SetRelativeHumidity(50);

    //特定の部位の条件のみを設定したい場合（右手先のみ乾球温度20Cとした）
    body.SetDrybulbTemperature(HumanBody.Nodes.RightHand, 20);

    //時間を経過させ、状態を書き出す
    Console.WriteLine("時刻   | 右肩コア温度   | 右肩皮膚温度   | 左肩コア温度   | 左肩皮膚温度");
    for (int i = 0; i < 15; i++)
    {
        body.Update(120);
        ImmutableBodyPart rightShoulder = body.GetBodyPart(HumanBody.Nodes.RightShoulder);
        ImmutableBodyPart leftShoulder = body.GetBodyPart(HumanBody.Nodes.LeftShoulder);
        Console.Write(((i + 1)  * 120) + "sec | ");
        Console.Write(rightShoulder.GetTemperature(BodyPart.Segments.Core).ToString("F2") + " | ");
        Console.Write(rightShoulder.GetTemperature(BodyPart.Segments.Skin).ToString("F2") + " | ");
        Console.Write(leftShoulder.GetTemperature(BodyPart.Segments.Core).ToString("F2") + " | ");
        Console.Write(leftShoulder.GetTemperature(BodyPart.Segments.Skin).ToString("F2") + " | ");
        Console.WriteLine();
    }

    Console.Read();
}
```

図 4.4　人体モデルのプログラム例

```
時刻     |  右肩コア温度  |  右肩皮膚温度  |  左肩コア温度  |  左肩皮膚温度
120sec  | 36.38 | 36.15 | 36.38 | 36.22 |
240sec  | 36.44 | 36.61 | 36.45 | 36.79 |
360sec  | 36.50 | 36.63 | 36.52 | 36.86 |
480sec  | 36.56 | 36.55 | 36.59 | 36.80 |
600sec  | 36.61 | 36.46 | 36.66 | 36.70 |
720sec  | 36.66 | 36.42 | 36.71 | 36.65 |
840sec  | 36.70 | 36.37 | 36.76 | 36.60 |
960sec  | 36.73 | 36.34 | 36.81 | 36.56 |
1080sec | 36.77 | 36.32 | 36.85 | 36.54 |
1200sec | 36.80 | 36.31 | 36.89 | 36.52 |
1320sec | 36.83 | 36.30 | 36.92 | 36.51 |
1440sec | 36.85 | 36.30 | 36.95 | 36.50 |
1560sec | 36.88 | 36.29 | 36.98 | 36.49 |
1680sec | 36.90 | 36.29 | 37.01 | 36.48 |
1800sec | 36.92 | 36.29 | 37.03 | 36.48 |
```

図 4.5　プログラム実行結果

Chapter 5 Calculating Weather State

5.1 気象状態に関連する計算を行うクラスが属する名前空間

建物の熱負荷を計算するためのクラスは「Popolo.Utility.Weather」名前空間に属しています。表 5.1 に Popolo.Utility.Weather 名前空間に属する主要なメンバを示します。

表 5.1 Popolo.Utility.Weather 名前空間に属する主要なメンバ

クラス名称	内容
Incline	斜面を定義するためのクラス
Sky	気象全般に関する静的メソッドを提供するクラス
Sun	太陽を表現するためのクラス
WeatherData	気象データを保持するためのクラス
WeatherDataTable	気象データを保持するためのクラス（WeatherRecord の集合体）
WeatherRecord	気象データを保持するためのクラス（WeatherData の集合体）

5.2 各クラスの概要

5.2.1 Incline class

日射や夜間放射などの計算を行う際には、太陽や天空との位置関係を計算する必要があります。Incline は、ある平面の 3 次元的な傾斜を表現するためのクラスです。

傾斜面は、水平方向の向きである方位角と、垂直方向の向きである傾斜角を指定することで定めることが出来ます。Incline はコンストラクタで方位角と傾斜角を指定することができます。Incline のコンストラクタはオーバーロードされており、表 5.2 に示すように 3 種類の引数の組み合わせをとります。

表 5.2 Incline コンストラクタの引数一覧

No.	引数 1	引数 2
1	方位角：南を 0、東を負、西を正として radian で指定 (double 型)	傾斜角：水平面を 0、垂直面を 1/2π として radian で指定 (double 型)
2	方位 (Orientation 型)	傾斜角：水平面を 0、垂直面を 1/2π として radian で指定 (double 型)-
3	コピーする傾斜面オブジェクト (ImmutableIncline 型)	-

2 つ目のコンストラクタの第 1 引数である Orientation 型は、Incline クラスに定義された列挙型で、16 方位を示すことが可能です。

傾斜角は水平面を 0、垂直面を 1/2π としており、地面を向いた水平面は π となります。従って、例えば 45 度のオーバーハングとなっている傾斜面は 3/4π で表現することができます。

3 つ目のコンストラクタはコピーコンストラクタで、引数で渡された傾斜面オブジェクト（読み取り専用）と同じプロパティを持った傾斜面を作成します。

Incline クラスの主要なメソッドを表 5.3 に示します。

表 5.3 Incline クラスの主要メソッド

名称		内容		
GetDirectSolarRadiationRate	概要	傾斜面の法線に対する太陽光線入射角の余弦 cosθ[-]を計算する		
	戻り値	傾斜面の法線に対する太陽光線入射角の余弦 cosθ[-]		
	引数 1	太陽 (ImmutableSun 型)	-	-
GetTanPhiAndGamma	概要	プロファイル角および傾斜面の法線を基準とした太陽方位角の正接を求める		
	戻り値	-		
	引数 1	太陽 (ImmutableSun 型)	引数 2	出力：プロファイル角の正接
	引数 3	出力：傾斜面の法線を基準とした太陽方位角の正接		
Reverse	概要	向きを逆転させる		
	戻り値	-		

日射の計算を行うにあたっては、傾斜面と太陽との位置関係を知る必要があり、これは図 5.1 の様に示されます。ここで、γおよびϕは傾斜面の法線方向を基準とした太陽方位角と太陽高度で、特にϕは見かけの太陽高度と呼ばれます。

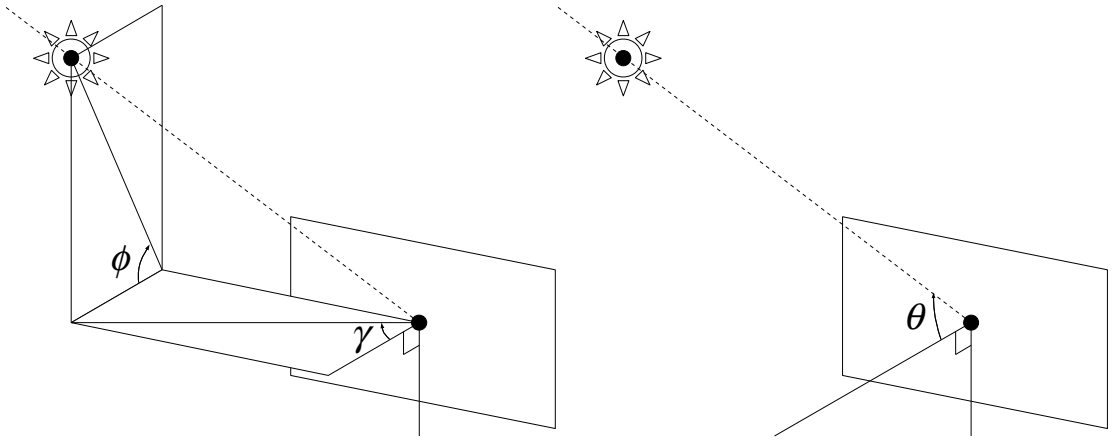

図 5.1 傾斜面と太陽との位置関係

このような傾斜面と太陽との位置関係を計算するために、GetTanPhiAndGamma メソッドが定義されています。第 1 引数は太陽を表すオブジェクトで 5.2.3 節で説明します。第 2 および第 3 引数は出力項目で、上記の見かけの太陽高度ϕおよびγの正接です。

また、図 5.1 右に示すように傾斜面の法線と太陽が形成する角度θを直接的に知りたい場合には GetDirectSolarRadiationRate メソッドを利用します。出力として cosθが計算されますが、この値は太陽の法線面直達日射について、傾斜面の法線方向への成分をとったものです。

Reverse メソッドは傾斜面を逆転させる場合に使用します。例えば東向き上 45°を向いている傾斜面について Reverse メソッドを呼ぶと、西向き下 45° (135°) を向いている傾斜面に変換されます。

夜間放射の計算のためには天空への形態係数を知る必要があります。この値は Incline クラスの GeometricFactor プロパティを参照することで得られます。

5.2.2 Sky class

Sky クラスは、天空に関する処理を提供する静的クラスです。Sky クラスの主要メソッドを表 5.4 に示します。

表 5.4 Sky クラスの主要メソッド

名称	内容			
DegreeToRadian	概要	角度[degree]をラジアン[radian]に変換する		
	戻り値	ラジアン[radian]		
	引数 1	角度[degree]	-	-
GetAtmosphericRadiation	概要	大気放射[W/m^2]を計算する		
	戻り値	大気放射[W/m^2]		
	引数 1	外気乾球温度[C]	引数 2	雲量[-] (0~1)
	引数 3	水蒸気分圧[kPa]		
GetNocturnalRadiation	概要	夜間放射[W/m^2]を計算する		
	戻り値	夜間放射[W/m^2]		
	引数 1	外気乾球温度[C]	引数 2	雲量[-] (0~1)
	引数 3	水蒸気分圧[kPa]		
RadianToDegree	概要	ラジアン[radian]を角度[degree]に変換する		
	戻り値	角度[degree]		
	引数 1	ラジアン[radian]	-	-

5.2.3 Sun class

1) コンストラクタ

Sun は太陽を表現するクラスです。地球上の特定の点を基準として太陽に関する計算を行うため、コンストラクタでは地球上の位置情報を与える必要があります。表 5.5 に Sun クラスのコンストラクタを示します。

表 5.5 Sun コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3
1	計算地点の緯度[degree] (double 型)	計算地点の経度[degree] (double 型)	標準時を規定する地点の経度[degree] (double 型)
2	基準となる都市 (City 型)	-	-
3	コピーする太陽オブジェクト (ImmutableSun 型)	-	-

2 つ目のコンストラクタの引数である City 型は、Sun クラスに定義された列挙型で、全世界の 110 程度の主要な都市を表現することができます。引数としてコンストラクタに渡すことで、その都市が位置している経度および緯度で Sun オブジェクトが初期化されます。次の例は東京で初期化を行った例です。

```
Sun sun = new Sun(Sun.City.Tokyo);
```

2) static メソッド

Sun クラスには、太陽位置や日射関連の基礎的計算を行うための static メソッドが定義されています。表 5.6 および表 5.7 に主要な static メソッドを示します。括弧書きで指定がない限り、引数の型は全て実数 (double) 型です。

表 5.6 Sun クラス：太陽位置に関する static メソッド

名称	内容			
GetEquationOfTime	概要	均時差を計算する。		
	戻り値	均時差		
	引数 1	日付 (DateTime 型)		
GetHourAngle	概要	均時差と地点情報に基づいて時角[°]を計算する。		
	戻り値	時角[°]		
	引数 1	均時差	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)
GetSunAltitude	概要	太陽高度[radian]を計算する		
	戻り値	太陽高度[radian]		
	引数 1	緯度[degree]	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)
GetSunAzimuth	概要	太陽方位角[radian]を計算する		
	戻り値	太陽方位角[radian]		
	引数 1	緯度[degree]	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)
GetSunDeclination	概要	太陽赤緯[degree]を計算する		
	戻り値	太陽赤緯[degree]		
	引数 1	計算を行う日時 (DateTime 型)		
GetSunPosition	概要	太陽高度[radian]および太陽方位[radian]を計算する		
	戻り値	-		
	引数 1	緯度[degree]	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)
	引数 5	出力：太陽高度[radian]	引数 6	出力：太陽方位角[radian]
GetSunRiseTime	概要	日の出の時刻を計算する		
	戻り値	日の出の時刻 (DateTime 型)		
	引数 1	緯度[degree]	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)
GetSunSetTime	概要	日の入の時刻を計算する		
	戻り値	日の入の時刻 (DateTime 型)		
	引数 1	緯度[degree]	引数 2	計算地点の経度[degree]
	引数 3	標準時を規定する地点の経度[degree]	引数 4	計算を行う日時 (DateTime 型)

表 5.7　Sun クラス：日射に関するメソッド

名称	内容			
GetExtraterrestrial Radiation	概要	大気圏外日射[W/m²]を計算する		
	戻り値	大気圏外日射[W/m²]		
	引数 1	通日 (int 型)	-	-
GetDirectNormalRadiation	概要	全天日射[W/m²]と天空日射[W/m²]から直達日射[W/m²]を求める		
	戻り値	直達日射[W/m²]		
	引数 1	全天日射[W/m²]	引数 2	天空日射[W/m²]
	引数 3	太陽高度[radian]	-	-
GetDiffuseHorizontal Radiation	概要	直達日射[W/m²]と全天日射[W/m²]から天空日射[W/m²]を求める		
	戻り値	天空日射[W/m²]		
	引数 1	直達日射[W/m²]	引数 2	全天日射[W/m²]
	引数 3	太陽高度[radian]	-	-
GetGlobalHorizontal Radiation	概要	天空日射[W/m²]と直達日射[W/m²]から全天日射[W/m²]を求める		
	戻り値	全天日射[W/m²]		
	引数 1	天空日射[W/m²]	引数 2	直達日射[W/m²]
	引数 3	太陽高度[radian]	-	-
EstimateDiffuseAnd DirectNormalRadiation	概要	水平面全天日射[W/m²]をもとに直散分離を行う		
	-	-		
	引数 1	水平面全天日射[W/m²]	引数 2	緯度[degree]
	引数 3	計算地点の経度[degree]	引数 4	標準時を規定する地点の経度[degree]
	引数 5	計算を行う日時 (DateTime 型)	引数 6	直散分離の手法
	引数 7	出力：法線面直達日射[W/m²]	引数 8	出力：天空日射[W/m²]

経度や緯度を引数とするメソッドがありますが、経度に関しては東を正、緯度に関しては北を正に取ることとします。例えば東京の場合では、東経 139.45°で北緯 35.4°ですから、139.45 と 35.4 を引数で与えます。一方、例えばブラジルのリオデジャネイロは西経 22.57°、南緯 43.12°にありますから-22.57 と-43.12 を引数で与えます。

標準時を規定する地点の経度とは、日本では明石市のある東経 135°のことです。

気象データによっては、水平面全天日射しか計測されていない場合があります。この場合に水平面全天日射を、法線面直達日射と水平面天空日射の成分に分割することを直散分離と呼びます。直散分離を実行するために、EstimateDiffuseAndDirectNormalRadiation メソッドが定義されています。

直散分離の手法としては、Berlage[5]、松尾[6]、永田[7]、Liu&Jordan[8]、宇田川・木村[9]、渡辺[10]、赤坂[11]、三木[12]の手法が利用可能で、第 6 引数で、DiffuseAndDirectNormalRadiationEstimatingMethod 列挙型を指定することで手法が選択できます。

3) 一般のプロパティ・メソッド

プロパティを利用して太陽に関する種々の情報を得ることができます。表 5.8 に Sun クラスの主要なプロパティを示します。

この内、天空放射量、法線面直達日射、水平面全天日射以外のプロパティは、地点と日時が確定すれば計算することができます。地点に関してはコンストラクタで既に指定しているため、日時情報を与える必要がありますが、これは Update メソッドを利用して実行することができます。引数は DateTime 型の変数です。Update メソッドが呼ばれる度に太陽位置等の情報は更新されます。

```
sun.Update(dateTime);
```

表 5.8 Sun クラスの主要プロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
Altitude	太陽高度	○	radian	double
CurrentDateTime	現在の日時	-	-	DateTime
DiffuseHorizontalRadiation	天空放射	○	W/m^2	double
DirectNormalRadiation	法線面直達日射	○	W/m^2	double
GlobalHorizontalRadiation	水平面全天日射	○	W/m^2	double
Latitude	計算地点の緯度	-	degree	double
Longitude	計算地点の経度	-	degree	double
Orientation	太陽方位角	-	radian	double
Revision	リビジョン	-	-	uint
StandardLongitude	標準時を規定する地点の経度	-	degree	double
SunRiseTime	日の出時刻	-	-	DateTime
SunSetTime	日の入時刻	-	-	DateTime

水平面天空日射量、法線面直達日射量、水平面全天日射量に関しては、いずれか 2 つの値が決まれば、太陽高度に基づいて残りの 1 つの値を求めることができます（図 5.2）。このため、Sun クラスには表 5.9 に示すメソッドが定義されています。

表 5.9 日射設定用メソッド

メソッド名称	引数 1	引数 2
SetDirectNormalRadiation	水平面全天日射	水平面天空日射
SetDiffuseHorizontalRadiation	法線面直達日射	水平面全天日射
SetGlobalHorizontalRadiation	水平面天空日射	法線面直達日射

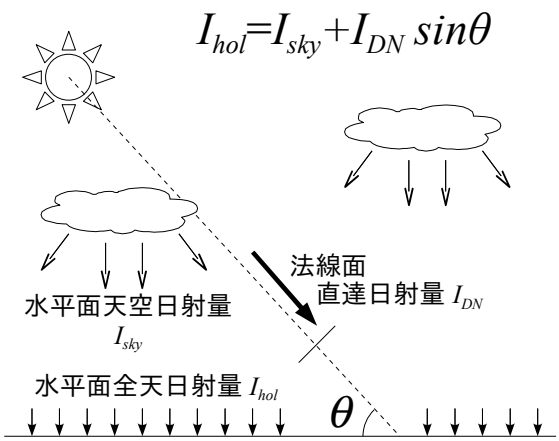

図 5.2　日射量の関係

直散分離のためのインスタンスメソッドも定義されており、Sun オブジェクトから直接に呼ぶことが可能です。この場合、太陽位置、日時などの情報は Sun オブジェクトの値が利用されるため、水平面全天日射量と直散分離の手法の 2 つが引数となります。推定された直達日射と天空日射は Sun オブジェクトに代入されます。

```
sun.EstimateDiffuseAndDirectNormalRadiation(globalHorizontalRadiation,
DiffuseAndDirectNormalRadiationEstimatingMethod.Udagawa);
```

5.3　気象状態の計算例

本節では、具体的な気象状態の計算例として太陽位置の計算と直散分離の計算法を示します。

5.3.1　太陽位置の計算

作成中

5.3.2 直散分離の計算

図 5.3 にプログラムを示します。観測された水平面全天日射量をもとに直散分離を行い、特定の傾斜面に入射する直達日射を計算します。図 5.4 は実行結果です。

```
/// <summary>気象状態計算の例</summary>
private static void w eatherTest()
{
    //東京における太陽を作成
    Sun sun = new Sun(Sun.City.Tokyo);

    //太陽位置を12月21日12時に調整
    DateTime dTime = new DateTime(1983, 12, 21, 12, 0, 0);
    sun.Update(dTime);

    //傾斜面を作成（南西の垂直面と東の45°傾斜面）
    Incline seInc = new Incline(Incline.Orientation.SE, 0.5 * Math.PI);
    Incline w Inc = new Incline(Incline.Orientation.W, 0.25 * Math.PI);

    //直散分離を実行して太陽に設定
    sun.EstimateDiffuseAndDirectNormalRadiation(467);

    //傾斜面へ入射する直達日射成分を計算する
    double cosThetaSE, cosThetaW;
    cosThetaSE = seInc.GetDirectSolarRadiationRate(sun);
    cosThetaW = w Inc.GetDirectSolarRadiationRate(sun);

    Console.WriteLine("東京の12月21日12時における");
    Console.WriteLine("太陽高度=" + Sky.RadianToDegree(sun.Altitude).ToString("F1") + " 度");
    Console.WriteLine("太陽方位=" + Sky.RadianToDegree(sun.Orientation).ToString("F1") + " 度");
    Console.WriteLine("法線面直達日射量=" + sun.DirectNormalRadiation.ToString("F1") + " W/m2");
    Console.WriteLine("水平面全天日射量=" + sun.GlobalHorizontalRadiation.ToString("F1") + " W/m2");
    Console.WriteLine("南西垂直面の直達日射量=" + (sun.DirectNormalRadiation * cosThetaSE).ToString("F1") + " W/m2");
    Console.WriteLine("東45度面の直達日射量=" + (sun.DirectNormalRadiation * cosThetaW).ToString("F1") + " W/m2");

    Console.Read();
}
```

図 5.3　気象状態計算の例

```
東京の12月21日12時における
太陽高度=31.0 度
太陽方位=5.1 度
法線面直達日射量=632.1 W/m2
水平面全天日射量=467.0 W/m2
南西の垂直面の直達日射量=347.4 W/m2
東の45度面の直達日射量=264.3 W/m2
```

図 5.4　実行結果

Chapter 6　Calculating Thermal Load of Building

6.1　建物熱負荷計算クラスが属する名前空間

建物の熱負荷を計算するためのクラスは「Popolo.Utility.LoadCalculation」名前空間に属しています。表 6.1 に Popolo.Utility.LoadCalculation 名前空間に属する主要なメンバを示します。図 6.1 に熱負荷計算に利用する各クラスの関係性を示します。

表 6.1　Popolo.Utility.LoadCalculation 名前空間に属する主要なメンバ

クラス名称	内容
IHeatGain	発熱要素を定義するための interface
Outdoor	外気状態、太陽位置などの外界条件を表現するクラス
SunShade	代表的な形状の日除けを表現するクラス
Wall	壁体を表現するクラス
WallLayers	壁体内部の壁層を表現するクラス
WallMaterial	壁体の材料を表現するクラス
WallSurface	短波長、長波長放射率や傾斜角など、壁体表面を表現するクラス
Window	窓を表現するクラス
WindowSurface	短波長、長波長放射率や傾斜角など、窓表面を表現するクラス
Zone	壁体、窓などを持つ熱負荷計算の対象となるゾーンを表現するクラス

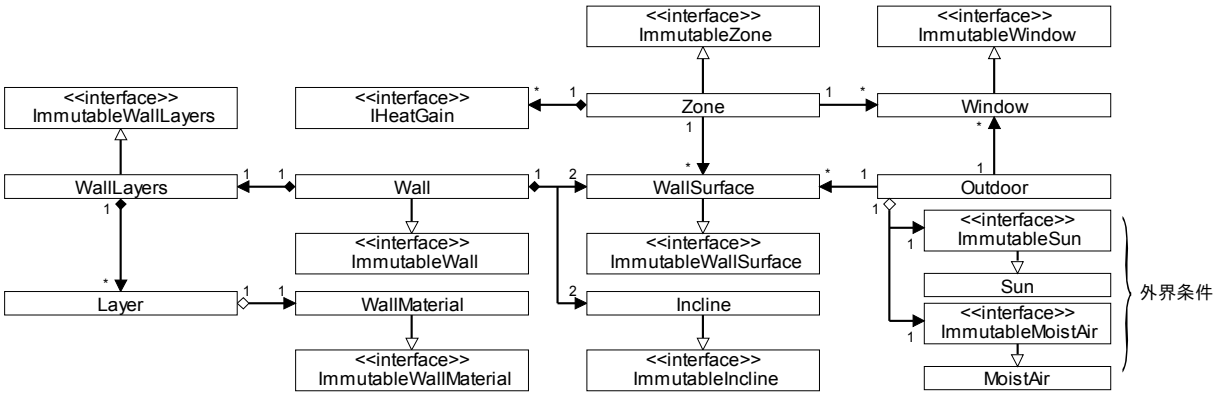

図 6.1　Popolo.Utility.LoadCalculation 名前空間に関連するクラスの関係

6.2　各クラスの概要

6.2.1　GlassPanes class

GlassPanes クラスは複数枚のガラス板で構成された層状のガラスを表現するクラスです。GlassPanes には異なる引数を持つ 2 種類のコンストラクタが定義されています。表 6.2 に GlassPanes クラスのコンストラクタを示します。

表 6.2　GlassPanes コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3
1	総合透過率[-] (double 型)	総合吸収率[-] (double 型)	熱貫流率[W/(m2-K)] (double 型)
2	層をなすガラス板のリスト (Pane 型配列)	-	-

ガラスからの日射熱取得を計算するためには、図 6.2 に示すように透過日射による熱取得、日射吸収による熱取得、貫流による熱取得を考える必要があります。1 つめのコンストラクタはこれらの量を規定する総合透過率、総合吸収率、熱貫流率を直接に与えるものです。

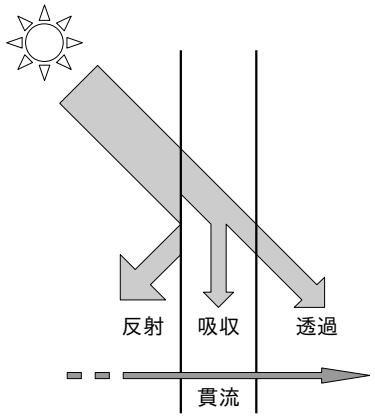

図 6.2　ガラスからの日射熱取得

ガラス層が 1 枚のガラス板のみで構成される場合、総合透過率と総合吸収率はガラス 1 枚の透過率および吸収率と一致します。一方、複数のガラスと空気層から構成される複層ガラスの場合、ガラス板の間で反射が繰り返されるため、計算は複雑になります。GlassPanes クラスの 2 つめのコンストラクタは、このような複数のガラス板から構成されるガラス層を取り扱うためのものです。引数である Pane クラスは 1 枚のガラス板を表現するクラスで、GlassPanes クラスのインナークラスとして定義されています。表 6.3 に GlassPanes.Pane クラスのコンストラクタを示します。

表 6.3　GlassPanes.Pane コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3	引数 4
1	透過率[-] (double 型)	吸収率[-] (double 型)	-	-
2	外側透過率[-] (double 型)	外側吸収率[-] (double 型)	内側透過率[-] (double 型)	内側吸収率[-] (double 型)
3	ガラス板の種類 (PredifinedGlassPane 型)	-	-	-

ガラス板によっては日射が入射する方向によって透過率および吸収率の値が異なるものがあります。どちらから入射しても同様の性質を示す場合には、1 つめのコンストラクタを使用します。入射方向によって特性が変化する場合には 2 つめのコンストラクタを使用し、内側および外側に応じて透過率と吸収率を設定します。また、いくつかの代表的なガラス板は定義済みで、これを利用して初期化を行う場合には 3 つめのコンストラクタを使用します。

Pane クラスを利用して個別のガラス板の透過率および吸収率を定義し、ガラス板の配列を GlassPanes クラスのコンストラクタで渡すと、GlassPanes クラス内部で複数のガラス板相互の反射等が計算され、総合透過率と総合吸収率が計算されます。ただし、配列の要素番号が小さい側が室内側となります。

表 6.4 に GlassPanes クラスのプロパティ一覧を示します。

表 6.4　GlassPanes クラスのプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
AngularDependenceCoefficients	入射角特性係数	○	-	double[]
ConvectiveRate	熱取得の対流成分の割合	○	-	double
HeatTransferCoefficientOfGlass	ガラス層のみの熱貫流率	-	W/(m²K)	double
LongWaveEmissivity	長波長放射率	○	-	double
OverallAbsorptance	吸収日射取得率	-	-	double
OverallHeatTransmissionCoefficient	熱貫流率	-	W/(m²K)	double
OverallTransmittance	日射透過率	-	-	double
RadiativeRate	熱取得の放射成分の割合	○	-	double

AngularDependenceCoefficients はガラスの入射角特性を計算するための係数です。日射が垂直に入射した場合の透過率を τ_N、入射角 θ の場合の透過率を τ_D として、入射角特性を式 6.1 で表現します。この時の A_i の実数値配列が AngularDependenceCoefficients です。デフォルトでは A_i = {3.4167, -4.389, 2.4948, -0.5224} と設定されており、図 6.3 に示す入射角特性となっています。通常はこの特性を利用して問題ありません。

$$\frac{\tau_D}{\tau_N} = \sum_{i=1}^{N} A_i \cos^i \theta \tag{6.1}$$

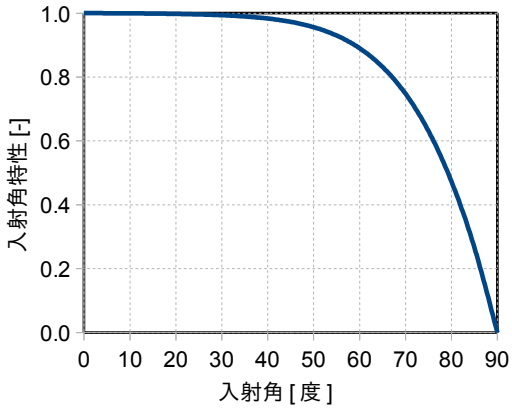

図 6.3　入射角特性

貫流熱と吸収日射熱取得はそれぞれ即時に熱負荷となる対流成分と、室の壁面に一旦吸収された後に時間遅れを伴って熱負荷となる放射成分とに分解することが出来ます。ConvectiveRate および RadiativeRate はそれぞれ対流成分および放射成分の割合を示しており、0~1 の範囲の実数で指定する必要があります。

LongWaveEmissivity は長波長放射率で、夜間にガラスから天空へ放射される熱量を計算する際に利用されます。

HeatTransferCoefficientOfGlass はガラスのみの熱貫流率で、これにガラス両端の空気の総合熱伝達率の影響を加えたものが窓全体の熱貫流率で OverallHeatTransmissionCoefficient です。

OverallAbsorptance と OverallTransmittance はそれぞれ総合吸収率と総合透過率を意味しており、吸収日射と透過日射による熱取得を計算する際に利用します。

表 6.5 に GlassPanes クラスの主要なメソッドを示します。

表 6.5　GlassPanes クラスの主要メソッド

名称	内容			
Copy	概要	GlassPanes オブジェクトをコピーする		
	戻り値	-		
	引数 1	コピーする GlassPanes (ImmutableGlassPanes 型)	-	-
GetStandardIncidentAngle Characteristic	概要	ガラスの標準入射角特性[-]を計算する		
	戻り値	ガラスの標準入射角特性[-]		
	引数 1	入射角の余弦（cosθ）	-	-
SetHeatTransfer CoefficientsOfAirGaps	概要	空気層の総合熱伝達率[W/(m^{2}K)]を設定する		
	戻り値	-		
	引数 1	空気層の番号 ：室内側から 0,1,2,3..	引数 2	空気層の総合熱伝達率[W/(m^{2}K)]
SetInsideOverallHeat TransferCoefficient	概要	内表面総合熱伝達率[W/(m^{2}K)]を設定する		
	戻り値	-		
	引数 1	内表面総合熱伝達率[W/(m^{2}K)]	-	-
SetOutsideOverallHeat TransferCoefficient	概要	外表面総合熱伝達率[W/(m^{2}K)]を設定する		
	戻り値	-		
	引数 1	外表面総合熱伝達率[W/(m^{2}K)]	-	-

複数のガラス板で構成されたガラス層の場合、各ガラス板の間にある空気層の総合熱伝達率を設定する必要があります。SetHeatTransferCoefficientsOfAirGaps メソッドを使用すると、ガラス板間の総合熱伝達率を設定することができます。

図 6.4 に複層ガラスの特性を計算するプログラムの例を示します。2 枚の板ガラスと中空層で構成された複層ガラスです。ガラス板は 6mm の透明ガラスと 6mm の熱戦吸収ガラスを使用しています。PredifinedGlassPane 列挙型を利用してガラス板を初期化しており、それぞれのガラスの透過率および吸収率をコンソールに出力しています。これらのガラス板の配列を GlassPanes クラスのコンストラクタに渡すことで、複層ガラスオブジェクトを作成し、総合透過率、総合吸収率、熱貫流率を出力しています。また、ガラス板の順序を入れ替えることで、外側に透明ガラスを配置した場合と内側に透明ガラスを配置した場合の違いを計算しています。

図 6.5 にプログラムの実行結果を示します。

```
private static void glassPanesTest()
    {
        //ガラス板を作成
        GlassPanes.Pane[] panes = new GlassPanes.Pane[2];

        //室内側は6mmの透明ガラス、室外側は6mmの熱線吸収ガラスの場合
        panes[0] = new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.TransparentGlass06mm);
        panes[1] = new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.HeatAbsorbingGlass06mm);

        //物性確認
        Console.WriteLine("透明ガラスの透過率=" + panes[0].InnerSideTransmittance.ToString("F2"));
        Console.WriteLine("透明ガラスの吸収率=" + panes[0].InnerSideAbsorptance.ToString("F2"));
        Console.WriteLine("熱線吸収ガラスの透過率=" + panes[1].InnerSideTransmittance.ToString("F2"));
        Console.WriteLine("熱線吸収ガラスの吸収率=" + panes[1].InnerSideAbsorptance.ToString("F2"));
        Console.WriteLine();

        //ガラス作成
        GlassPanes glass = new GlassPanes(panes);

        //空気層の総合熱伝達率[W/(m2-K)]を設定
        glass.SetHeatTransferCoefficientsOfAirGaps(0, 6d);

        Console.WriteLine("室内側：透明ガラス　室外側：熱線吸収ガラス");
        Console.WriteLine("総合透過率[-] = " + glass.OverallTransmittance.ToString("F3"));
        Console.WriteLine("総合吸収率[-] = " + glass.OverallAbsorptance.ToString("F3"));
        Console.WriteLine("熱貫流率[W/(m2-K)]" + glass.OverallHeatTransferCoefficient.ToString("F3"));
        Console.WriteLine();

        //室内側は6mmの熱線吸収ガラス、室外側は6mmの透明ガラスの場合
        panes[0] = new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.HeatAbsorbingGlass06mm);
        panes[1] = new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.TransparentGlass06mm);

        //ガラス作成
        glass = new GlassPanes(panes);

        //空気層の総合熱伝達率[W/(m2-K)]を設定
        glass.SetHeatTransferCoefficientsOfAirGaps(0, 6d);

        Console.WriteLine("室内側：熱線吸収ガラス　室外側：透明ガラス");
        Console.WriteLine("総合透過率[-] = " + glass.OverallTransmittance.ToString("F3"));
        Console.WriteLine("総合吸収率[-] = " + glass.OverallAbsorptance.ToString("F3"));
        Console.WriteLine("熱貫流率[W/(m2-K)]" + glass.OverallHeatTransferCoefficient.ToString("F3"));

        Console.Read();
    }
```

図 6.4　複層ガラスの特性計算の例

```
透明ガラスの透過率=0.79
透明ガラスの吸収率=0.14
熱線吸収ガラスの透過率=0.60
熱線吸収ガラスの吸収率=0.34

室内側：透明ガラス　室外側：熱線吸収ガラス
総合透過率[-] = 0.476
総合吸収率[-] = 0.104
熱貫流率[W/(m2-K)]3.148

室内側：熱線吸収ガラス　室外側：透明ガラス
総合透過率[-] = 0.476
総合吸収率[-] = 0.198
熱貫流率[W/(m2-K)]3.148
```

図 6.5　実行結果

6.2.2 Window class

Window クラスは建物の窓を表現するクラスです。6.2.1 節で解説した GlassPanes オブジェクトをコンストラクタで指定することで、GlassPanes オブジェクトの特性に従った窓を作成することができます。表 6.6 に Window クラスのコンストラクタの引数一覧を示します。

表 6.6 Window コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3
1	複層のガラス (ImmutableGlassPanes 型)	-	-
2	複層のガラス (ImmutableGlassPanes 型)	屋外側傾斜面 (ImmutableIncline 型)	-
3	複層のガラス (ImmutableGlassPanes 型)	屋外側傾斜面 (ImmutableIncline 型)	日除け (ImmutableSunShade 型)

第 2 引数は 5.2.1 節で解説した傾斜面オブジェクトで、窓の屋外側の傾斜の様子を表現します。また、第 3 引数は窓への日影の落ち方を計算するための日除けオブジェクトで、6.2.3 節で解説します。

表 6.7 に Window クラスの主要なプロパティを示します。

表 6.7 Window クラスの主要プロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
AbsorbedHeatGain	吸収日射による熱取得	-	W	double
Albedo	アルベド	○	-	double
ConvectiveHeatGain	対流熱取得	-	W	double
Glass	複層ガラス	-	-	ImmutableGlassPanes
IndoorDrybulbTemperature	屋内の乾球温度	○	°C	double
IndoorSurfaceTemperature	室内側表面温度[C]	-	°C	double
NocturnalRadiation	夜間放射	○	W/m^2	double
OutdoorDrybulbTemperature	外気乾球温度	○	°C	double
OutdoorSideIncline	屋外側傾斜面情報	-	-	ImmutableIncline
OutdoorSurfaceTemperature	室外側表面温度[C]	-	°C	double
RadiativeHeatGain	放射熱取得	-	W	double
Shade	日除け	○	-	ImmutableSunShade
ShadowRate	日影面積率	○	-	double
Sun	太陽	○	-	ImmutableSun
SurfaceArea	窓面積	○	m^2	double
TransferHeatGain	温度差による貫流熱取得	-	W	double
TransmissionHeatGain	透過日射による熱取得	-	W	double

窓を介して移動する熱としては、透過日射による熱移動、日射吸収による熱移動、温度差による貫流熱移動があります。これらの量を計算するためには、ガラスの特性に加え、いくつかの条件を与える必要があります。

地表面が日射を反射する大きさであるアルベドは、Albedo プロパティを利用して設定します。貫流熱移動に影響を与える室内外の乾球温度はそれぞれ、IndoorDrybulbTemperature と

OutdoorDrybulbTemperature プロパティを利用して設定します。夜間放射は天空への放射量を表しており、NocturnalRadiation プロパティで設定します。ShadowRate は日影となり直達日射が届かない面積割合を表す値ですが、直接に日影面積率を与えず、日除けオブジェクトを Shade プロパティで設定することもできます。法線面直達日射や天空放射などの値に関しては、Sun プロパティを利用して設定しますが、これは 5.2.3 節で解説した Sun 型のオブジェクトです。

以上の条件設定を適切に行うと、窓を介して移動する熱量を把握することができるようになります。AbsorbedHeatGain、TransferHeatGain、TransmissionHeatGain はそれぞれ、吸収日射による熱取得、温度差による貫流熱取得、透過日射による熱取得です。また、ConvectiveHeatGain と RadiativeHeatGain はそれぞれ対流および放射による熱取得を表しており、その比率は GlassPanes オブジェクトで設定した ConvectiveRate と RadiativeRate に従います。両者の和は AbsorbedHeatGain と TransferHeatGain の和に一致します。

図 6.6 に、Window クラスを利用した、窓からの熱取得計算プログラムの例を示します。

3~7 行目は気象データ配列です。1 日の直達日射、天空放射、乾球温度、夜間放射を実数値配列に設定しています。

9~12 行目では窓ガラスオブジェクトの作成を行っています。まず 10 行目で、6.2.1 節で解説した GlassPanes クラスを利用して 3mm のシングル透明ガラスを作成しています。この GlassPanes オブジェクトを Window クラスのコンストラクタで指定することで 3mm シングル透明ガラスを持った窓オブジェクトが作成されます。

15、16 行目では表面総合熱伝達率を設定しています。一般的に、屋外側は 23 W/(m^2-K)、屋内側 9.3 W/(m^2-K) 程度の値となります。

19、20 行目では、5.2.1 節で解説した Incline クラスを利用して窓の屋外側の傾斜面を設定しています。本例では傾斜面は南向きの垂直面としています。

23 行目では地表面反射率の設定を行っています。地面の状態に応じて地表面反射率は変化し、例えば新雪の場合などは 0.8 程度の値となりますが、通常の都市建築の場合、0.2 程度の値を設定します。

26 行目は日除けの設定ですが、本例の窓は日除けを持たないこととし、EmptySunShade を設定しておきます。詳細は 6.2.3 節で記します。

28~32 行目では 5.2.3 節で解説した太陽オブジェクトを作成しています。地点は東京とし、日付は 7 月 21 日に設定しています。計算開始時点の時刻は 0:00 としています。

41~58 の繰り返し文で、1 日 24 時間の日射熱取得計算を行っています。

まず、法線面直達日射量と天空放射量に基づいて Sun オブジェクトに日射量を設定しています。また、Window クラスの NocturnalRadiation プロパティと OutdoorDrybulbTemperature プロパティを利用して、夜間放射と外気乾球温度を設定しています。以上により外界条件の設定ができたため、Window クラスから熱取得に関する情報を取得できるようになります。51~53 行目では Window クラスから取得した情報をコンソールに書き出しています。最後に、56、57 行目で時刻を 1 時間進ませ、太陽位置を更新しています。以上を 24 回繰り返すことで終日の窓の熱取得計算が行われます。

図 6.7 はプログラムの実行結果です。

```
private static void windowTest1()
{
  //気象データ//直達日射,天空放射,乾球温度,夜間放射
  double[] wdIdn = new double[] { 0, 0, 0, 0, 0, 244, 517, 679, 774, 829, 856, 862, 847, 809, 739, 619, 415, 97, 0, 0, 0, 0, 0, 0 };
  double[] wdIsky = new double[] { 0, 0, 0, 0, 21, 85, 109, 116, 116, 113, 110, 109, 111, 114, 116, 114, 102, 63, 0, 0, 0, 0, 0, 0 };
  double[] wdDbt = new double[] { 27, 27, 27, 27, 27, 28, 29, 30, 31, 32, 32, 33, 33, 33, 34, 33, 32, 32, 31, 30, 29, 29, 28, 28 };
  double[] wdRN = new double[] { 24, 24, 24, 24, 24, 24, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 25, 25, 25, 25, 24, 24, 24 };

  //3mm透明ガラスの窓を作成
  GlassPanes.Pane pane = new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.TransparentGlass03mm);
  GlassPanes glassPane = new GlassPanes(pane);
  Window window = new Window(glassPane);

  //表面総合熱伝達率を設定
  window.SetOutsideOverallHeatTransferCoefficient(23d);
  window.SetInsideOverallHeatTransferCoefficient(9.3);

  //屋外面の傾斜を設定//南向き垂直壁
  Incline incline = new Incline(Incline.Orientation.S, 0.5 * Math.PI);
  window.OutdoorSideIncline = incline;

  //地表面反射率：アルベドを設定
  window.Albedo = 0.2;

  //日除けは無し
  window.Shade = SunShade.EmptySunShade;

  //7月21日0:00の東京の太陽を作成
  Sun sun = new Sun(Sun.City.Tokyo);
  DateTime dTime = new DateTime(2001, 7, 21, 0, 0, 0);
  sun.Update(dTime);
  window.Sun = sun;

  //室内側乾球温度は25度で一定とする
  window.IndoorDrybulbTemperature = 25;

  //計算結果タイトル行
  Console.WriteLine(" 時刻 | 透過日射[W] | 吸収日射[W] | 貫流熱[W] | 対流熱取得[W] | 放射熱取得[W]");

  //終日の計算実行
  for (int i = 0; i < 24; i++)
  {
//日射量設定
sun.SetGlobalHorizontalRadiation(wdIsky[i], wdIdn[i]);
//夜間放射設定
window.NocturnalRadiation = wdRN[i];
//外気乾球温度を設定
window.OutdoorDrybulbTemperature = wdDbt[i];

//計算結果書き出し
Console.WriteLine(dTime.ToShortTimeString().PadLeft(5) + " | " + window.TransmissionHeatGain.ToString("F1").PadLeft(11) + " | " +
  window.AbsorbedHeatGain.ToString("F1").PadLeft(11) + " | " +  window.TransferHeatGain.ToString("F1").PadLeft(9) + " | " +
  window.ConvectiveHeatGain.ToString("F1").PadLeft(13) + " | " + window.RadiativeHeatGain.ToString("F1").PadLeft(13));

//時刻更新
dTime = dTime.AddHours(1);
sun.Update(dTime);
  }

  Console.Read();
}
```

図 6.6　窓からの熱取得計算プログラムの例

時刻	透過日射[W]	吸収日射[W]	貫流熱[W]	対流熱取得[W]	放射熱取得[W]
0:00	0.0	0.0	12.8	5.8	7.1
1:00	0.0	0.0	10.8	4.9	6.0
2:00	0.0	0.0	8.8	4.0	4.9
3:00	0.0	0.0	6.9	3.1	3.8
4:00	9.7	0.3	9.8	4.5	5.5
5:00	40.2	1.1	15.9	7.6	9.3
6:00	60.2	1.6	23.6	11.3	13.9
7:00	76.8	2.1	30.6	14.7	18.0
8:00	91.1	2.4	37.6	18.0	22.0
9:00	131.7	3.5	43.4	21.1	25.8
10:00	186.8	5.0	48.7	24.2	29.5
11:00	225.7	6.0	53.0	26.6	32.5
12:00	233.1	6.2	55.2	27.7	33.8
13:00	206.0	5.5	56.5	27.9	34.1
14:00	153.4	4.1	57.1	27.5	33.7
15:00	97.7	2.6	52.9	25.0	30.6
16:00	65.0	1.7	47.4	22.1	27.0
17:00	31.9	0.9	40.7	18.7	22.9
18:00	0.0	0.0	33.9	15.2	18.6
19:00	0.0	0.0	28.6	12.9	15.7
20:00	0.0	0.0	23.9	10.8	13.2
21:00	0.0	0.0	20.1	9.0	11.1
22:00	0.0	0.0	17.5	7.9	9.6
23:00	0.0	0.0	14.8	6.7	8.1

図 6.7　実行結果

6.2.3　SunShade class

SunShade クラスは典型的な形状を持った日除けを表現するためのクラスです。図 6.8 に作成可能な日除けの種類を示します。これらの日除けを作成するために、いくつかの static メソッドが用意されています。表 6.9 に日除け作成のための static メソッドを示します。SunShade はコンストラクタを持たないため、SunShade 型のインスタンスを作成するためには、これらの static メソッドを利用する必要があります。また、特別な日除けとして、空の日除け（日除けが無く、直達日射が遮られない）が定義されており、static プロパティである EmptySunShade で取得できます。

表 6.8 に SunShade クラスの主要なプロパティを示します。多くのプロパティは読み取り専用で、窓各部位の長さに関する情報を持っています。

Incline は 5.2.1 節で解説した傾斜面オブジェクトで、適切に設定を行うことで、傾斜した壁に窓が設けられている場合などについても計算が可能です。デフォルトでは南面する垂直面としています。

IsReverse プロパティは日除けの位置を裏返すか否かで、このプロパティを true に設定すると、日除けが設定された位置のみから直達日射が届くとみなされます。例えばドライエリアの計算を行う場合などには、水平庇を定義して IsReverse プロパティを true に設定します。

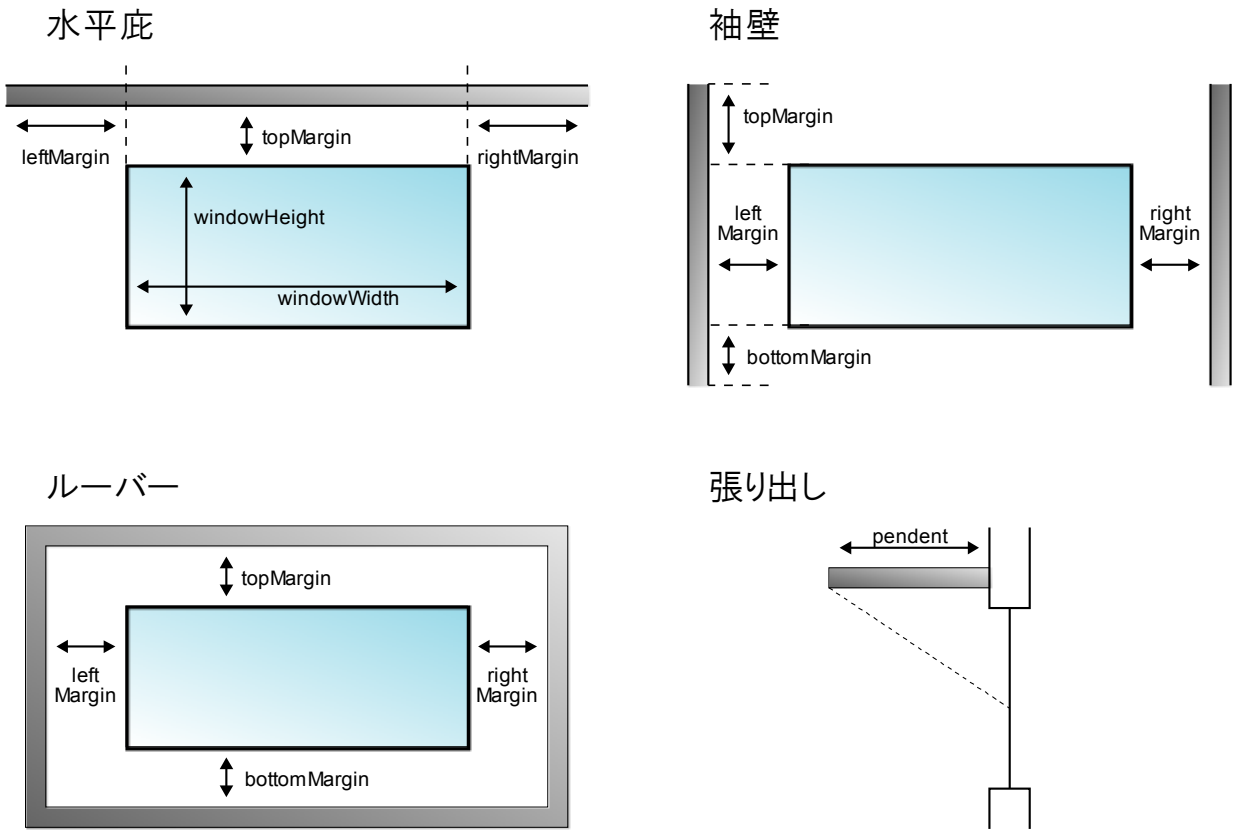

図 6.8　日除けの種類（英字は引数の変数名称）

表 6.8　SunShade クラスの主要プロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
BottomMargin	下部マージン	-	m	double
Incline	傾斜面	○	-	ImmutableIncline
IsReverse	裏返しか否か	○	-	bool
LeftMargin	左側マージン	-	m	double
Name	名称	○	-	string
Pendent	張り出し幅	-	m	double
RightMargin	右側マージン	-	m	double
SunShadeShape	日除けの種類	-	-	SunShade.Shape
TopMargin	上部マージン	-	m	double
WindowHeight	窓高さ	-	m	double
WindowWidth	窓幅	-	m	double

表 6.9　日除けを作成するための static メソッド

名称	内容			
MakeGridSunShade	概要	ルーバーを作成する		
	戻り値	ルーバー日除け		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	左側マージン[m]
	引数 5	右側マージン[m]	引数 6	上部マージン[m]
	引数 7	下部マージン[m]	-	-
MakeHorizontalSunShade	概要	水平庇を作成する		
	戻り値	水平庇		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	左側マージン[m]
	引数 5	右側マージン[m]	引数 6	上部マージン[m]
	引数 7	傾斜面	-	-
MakeHorizontalSunShade	概要	水平庇（無限長）を作成する		
	戻り値	水平庇（無限長）		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	上部マージン[m]
	引数 5	傾斜面	-	-
MakeVerticalSunShade	概要	袖壁を作成する		
	戻り値	袖壁		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	横側マージン[m]
	引数 5	左側か否か（右の場合は false）	引数 6	上部マージン[m]
	引数 7	下部マージン[m]	引数 8	傾斜面
MakeVerticalSunShade	概要	袖壁（無限長）を作成する		
	戻り値	袖壁（無限長）		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	横側マージン[m]
	引数 5	左側か否か（右の場合は false）	引数 6	傾斜面
MakeVerticalSunShade	概要	袖壁を作成する		
	戻り値	袖壁		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	横側マージン[m]
	引数 5	上部マージン[m]	引数 6	下部マージン[m]
	引数 7	傾斜面	-	-
MakeVerticalSunShade	概要	袖壁（無限長）を作成する		
	戻り値	袖壁（無限長）		
	引数 1	窓幅[m]	引数 2	窓高[m]
	引数 3	張り出し幅[m]	引数 4	横側マージン[m]
	引数 5	傾斜面	-	-

6.2.4 WallLayers class

WallLayers クラスは 1 以上の材料で構成される層状の壁体を表現するためのクラスです。WallLayers クラスのプロパティと主要なメソッドを表 6.10 と表 6.11 に示します。AddLayer や SetLayer メソッドを利用しながら層の構成を設定し、GetOverallHeatTransferCoefficient メソッドで全体の熱貫流率[W/(m²K)]を計算するという流れになります。なお、本クラスは読み取り専用の ImmutableWallLayers インターフェースを継承しています。

単一の壁層を表現するには、WallLayers クラスのインナークラスである Layer クラスを利用します。Layer クラスのコンストラクタを表 6.12 に示します。壁の材料と厚み、また、壁体が非常に厚い場合などには壁の分割数を設定します。

表 6.10 WallLayers クラスのプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
LayerNumber	壁層の数	-	-	uint
Name	名称	○	-	string

表 6.11 WallLayers クラスの主要メソッド

名称	内容			
GetOverallHeatTransfer Coefficient	概要	壁表面の総合熱伝達率 [W/(m²K)]を指定して熱貫流率 [W/(m²K)]を計算する		
	戻り値	熱貫流率[W/(m²K)]		
	引数 1	表面 1 の総合熱伝達率 [W/(m²K)]	引数 2	表面 2 の総合熱伝達率 [W/(m²K)]-
GetOverallHeatTransfer Coefficient	概要	熱貫流率[W/(m²K)]を計算する		
	戻り値	-		
	引数 1	-	引数 2	-
UsingMaterial	概要	壁素材を使っているか否かを返す		
	戻り値	壁素材を使っているか否か		
	引数 1	壁素材 (ImmutableWallMaterial 型)	-	-
AddLayer	概要	壁層を追加する		
	戻り値	-		
	引数 1	壁層 (Layer 型)	引数 2	-
GetLayer	概要	壁層配列を取得する		
	戻り値	壁層配列 (Layer 型)		
	引数 1	-	引数 2	-
GetLayer	概要	指定の位置の壁層を取得する		
	戻り値	壁層 (Layer 型)		
	引数 1	壁層番号	引数 2	-
RemoveLayer	概要	壁層を削除する		
	戻り値	-		
	引数 1	壁層番号	引数 2	-
SetLayer	概要	指定の位置に壁層を設定する		
	戻り値	-		
	引数 1	壁層番号	引数 2	壁層 (Layer 型)

表 6.12　WallLayers.Layer コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3
1	壁材料　(ImmutableWallMaterial 型)	壁の厚み　(double 型)	-
2	壁材料　(ImmutableWallMaterial 型)	壁の厚み　(double 型)	壁の分割数　(uint 型)

壁材料を表現するためには WallMaterial クラスを利用します。表 6.13 に WallMaterial コンストラクタの引数一覧を示します。素材の名称、熱伝導率[W/(m K)]、容積比熱[kJ/(m^3 K)]が主な情報です。2 番目のコンストラクタを利用することで、予め定義されているいくつかの主要な材料の物性で初期化することができます。表 6.15 に定義済の壁材料一覧を示します。3 番目のコンストラクタはコピーコンストラクタです。表 6.14 に WallMaterial クラスのプロパティ一覧を示します。

なお、空気層の場合には VolumetricSpecificHeat プロパティを 0 とし、ThermalConductivity プロパティに熱コンダクタンス[W/(m^2 K)]を設定します。

表 6.13　WallMaterial コンストラクタの引数一覧

No.	引数 1	引数 2	引数 3
1	素材名称　(string 型)	熱伝導率 [W/(m K)]　(double 型)	容積比熱 [kJ/(m^3 K)]　(double 型)
2	既定義の素材 (WallMaterial.PredifinedMaterials 列挙型)	-	-
3	壁材料　(ImmutableWallMaterial 型)	-	-

表 6.14　WallMaterial クラスのプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
Material	素材の種類	-	-	WallMaterial.PredifinedMaterials
Name	名称	○	-	string
ThermalConductivity	熱伝導率	○	W/(m K)	double
VolumetricSpecificHeat	容積比熱	○	kJ/(m^3 K)	double

以上に示したように、多層壁の材料の熱伝導率 λ と厚み l が与えられると、式 6.2 に従って、壁体全体の熱貫流率が計算できるようになります。ここで、α_1, α_2 は壁表面の総合熱伝達率 [W/(m^2 K)]、C_{air} は壁層に含まれる空気層の熱コンダクタンス [W/(m^2 K)]です。

$$K = 1/\left\{\frac{1}{\alpha_1} + \sum\frac{l}{\lambda} + \sum\frac{1}{C_{air}} + \frac{1}{\alpha_2}\right\} \tag{6.2}$$

図 6.9 に、WallLayers クラスと WallMaterial クラスを利用して、多層壁の熱貫流率を計算するプログラムの例を示します。

合板、非密閉の空気層、鉄筋コンクリート、漆喰の 4 層で構成される多層壁で、6~15 行目ではそれぞれの壁材料を作成しています。合板、非密閉の空気層、鉄筋コンクリートに関しては定義済の材料を利用して初期化を行い、漆喰については熱伝導率と容積比熱を直接に指定して作成しています。

17~25 行では、これらの壁材料を使って、多層壁を作成しています。それぞれの厚みは 20mm, 10mm, 150mm, 10mm です。

27~35 行で結果の書き出しを行っており、各層の材料の名称、厚みの他、全体の熱貫流率を算出しています。また、38 行目では鉄筋コンクリートを軽量コンクリートに入れ替えており、40~47 行で、その計算結果を書き出しています。図 6.10 に、計算結果を示します。

表 6.15　定義済の壁材料一覧

列挙型名称	壁材料	熱伝導率	容積比熱	列挙型名称	壁材料	熱伝導率	容積比熱
Mortar	セメント・モルタル	1.512	1591	SprayedRockWool	吹付けロックウール	0.047	167.9
ReinforcedConcrete	コンクリート	1.6	1896	BeadMethodPolystyreneFoam_S	ビーズ法ポリスチレンフォーム特号	0.034	33.9
LightweightAggregateConcrete1	軽量骨材コンクリート１種	0.81	1900	BeadMethodPolystyreneFoam_1	ビーズ法ポリスチレンフォーム１号	0.036	37.7
LightweightAggregateConcrete2	軽量骨材コンクリート２種	0.58	1599	BeadMethodPolystyreneFoam_2	ビーズ法ポリスチレンフォーム２号	0.037	31.4
AutomaticLevelControl	軽量気泡コンクリートパネル ALC パネル	0.17	661.4	BeadMethodPolystyreneFoam_3	ビーズ法ポリスチレンフォーム３号	0.04	25.1
Brick	普通れんが	0.62	1386	BeadMethodPolystyreneFoam_4	ビーズ法ポリスチレンフォーム４号	0.043	18.8
FireBrick	耐火れんが	0.99	1553	ExtrudedPolystyreneFoam_1	押出法ポリスチレンフォーム１種	0.04	25.1
Copper	銅	370.1	3144	ExtrudedPolystyreneFoam_2	押出法ポリスチレンフォーム２種	0.034	25.1
Aluminum	アルミニウム合金	200	2428	ExtrudedPolystyreneFoam_3	押出法ポリスチレンフォーム３種	0.028	25.1
Steel	鋼材	53.01	3759	RigidUrethaneFoam_1_1	硬質ウレタンフォーム保温版１種１号	0.024	56.1
Lead	鉛	35.01	1469	RigidUrethaneFoam_1_2	硬質ウレタンフォーム保温版１種２号	0.024	44
StainlessSteel	ステンレス鋼	15	3479	RigidUrethaneFoam_1_3	硬質ウレタンフォーム保温版１種３号	0.026	31.4
FloatGlass	フロートガラス	1	1914	RigidUrethaneFoam_2_1	硬質ウレタンフォーム保温版２種１号	0.023	56.1
PolyvinylChloride	PVC(塩化ビニル)	0.17	1023	RigidUrethaneFoam_2_2	硬質ウレタンフォーム保温版２種２号	0.023	44
Wood1	天然木材１類（桧、杉、えぞ松等）	0.12	519.1	RigidUrethaneFoam_2_3	硬質ウレタンフォーム保温版２種３号	0.024	31.4
Wood2	天然木材２類（松、ラワン等）	0.15	648.8	RigidUrethaneFoam_InSite	硬質ウレタンフォーム（現場発泡品）	0.026	49.8
Wood3	天然木材３類（ナラ、サクラ、ブナ等）	0.19	845.6	PolyethyleneFoam_A	ポリエチレンフォーム　A	0.038	62.8
Plywood	合板	0.19	716	PolyethyleneFoam_B	ポリエチレンフォーム　B	0.042	62.8
WoodWoolCement	断熱木毛セメント板	0.1	841.4	PhenolicFoam_1_1	フェノールフォーム保温版 １種１号	0.033	37.7
WoodChipCement	木片セメント板	0.17	1679	PhenolicFoam_1_2	フェノールフォーム保温版 １種２号	0.03	37.7
HardBoard	ハードボード	0.17	1233	PhenolicFoam_2_1	フェノールフォーム保温版 ２種１号	0.036	56.5
ParticleBoard	パーティクルボード	0.15	715.8	PhenolicFoam_2_2	フェノールフォーム保温版 ２種２号	0.034	56.5
PlasterBoard	せっこうボード	0.17	1030	InsulationBoard_A	A 級インシュレーションボード	0.049	324.8
GypsumPlaster	せっこうプラスター	0.6	1637	TatamiBoard	タタミボード	0.045	15.1
WhiteWash	漆喰	0.7	1093	SheathingInsulationBoard	シージングボード	0.052	390.1
SoilWall	土壁	0.69	1126	CelluloseFiberInsulation_1	吹込用セルローズファイバー断熱材１	0.04	37.7
FiberCoating	繊維質上塗材	0.12	4.2	CelluloseFiberInsulation_2	吹込用セルローズファイバー断熱材２	0.04	62.8
Tatami	畳床	0.11	527.4	Soil	土壌（ローム質）	1.047	3340
Tile	タイル	1.3	2018	ExpandedPolystyrene	EPS	0.035	300
PlasticTile	プラスチック（P）タイル	0.19	4.2	CoveringMaterial	外装材	0.14	1680
GlassWoolInsulation_10K	グラスウール断熱材 10K 相当	0.05	8.4	Linoleum	合成樹脂・リノリウム	0.19	1470
GlassWoolInsulation_16K	グラスウール断熱材 16K 相当	0.045	13.4	Carpet	カーペット	0.08	318
GlassWoolInsulation_24K	グラスウール断熱材 24K 相当	0.038	20.1	AsbestosPlate	石綿スレート	1.2	1820
GlassWoolInsulation_34K	グラスウール断熱材 32K 相当	0.036	26.8	SealedAirGap	密閉空気層	5.8	0
HighGradeGlassWoolInsulation_16K	高性能グラスウール断熱材 16K 相当	0.038	13.4	AirGap	非密閉空気層	11.6	0
HighGradeGlassWoolInsulation_24K	高性能グラスウール断熱材 24K 相当	0.036	20.1	PolystyreneFoam	ポリスチレンフォーム	0.035	80
BlowingGlassWoolInsulation_13K	吹込用グラスウール断熱材１種 13K 相当	0.052	10.9	StyreneFoam	スチレン発泡板	0.035	10
BlowingGlassWoolInsulation_18K	吹込用グラスウール断熱材２種 18K 相当	0.052	16.7	RubberTile	ゴムタイル	0.4	784
BlowingGlassWoolInsulation_30K	吹込用グラスウール断熱材２種 30K 相当	0.04	29.3	Kawara	瓦	1	1506
BlowingGlassWoolInsulation_35K	吹込用グラスウール断熱材２種 35K 相当	0.04	37.7	LightweightConcrete	軽量コンクリート	0.78	1607
RockWoolInsulationMat	住宅用ロックウール断熱材 マット	0.038	33.5	Asphalt	アスファルトルーフィング	0.11	920
RockWoolInsulationFelt	住宅用ロックウール断熱材 フェルト	0.038	41.9	FrexibleBoard	フレキシブルボード	0.35	1600
RockWoolInsulationBoard	住宅用ロックウール断熱材 ボード	0.036	58.6	CalciumSilicateBoard	珪酸カルシウム板	0.13	680
BlowingRockWoolInsulation_25K	吹込用ロックウール断熱材 25K	0.047	20.9	PhenolicFoam	高性能フェノールボード	0.02	37.7
BlowingRockWoolInsulation_35K	吹込用ロックウール断熱材 35K	0.051	29.3	GNT	花崗岩	4.3	2.9
RockWoolAcousticBoard	ロックウール化粧吸音板	0.058	293.9	AcrylicResin	アクリル樹脂	0.21	1666

```
private static void wallOverallHeatTransferCoefTest()
{
    //多層壁オブジェクトを作成
    WallLayers wLayers = new WallLayers("熱貫流率計算用多層壁");

    //壁層の素材を作成
    WallMaterial[] materials = new WallMaterial[4];
    //第1層：合板
    materials[0] = new WallMaterial(WallMaterial.PredefinedMaterials.Plywood);
    //第2層：非密閉空気層
    materials[1] = new WallMaterial(WallMaterial.PredefinedMaterials.AirGap);
    //第3層：鉄筋コンクリート
    materials[2] = new WallMaterial(WallMaterial.PredefinedMaterials.ReinforcedConcrete);
    //第4層：漆喰
    materials[3] = new WallMaterial("漆喰", 0.7, 1000);

    //壁の各層を作成
    //合板:20mm
    wLayers.AddLayer(new WallLayers.Layer(materials[0], 0.02));
    //空気層:厚みは関係なし
    wLayers.AddLayer(new WallLayers.Layer(materials[1], 0.01));
    //鉄筋コンクリート:150mm
    wLayers.AddLayer(new WallLayers.Layer(materials[2], 0.15));
    //漆喰:10mm
    wLayers.AddLayer(new WallLayers.Layer(materials[3], 0.01));

    //結果書き出し
    Console.WriteLine("壁層の構成");
    for (uint i = 0; i < wLayers.LayerNumber; i++)
    {
WallLayers.Layer layer = wLayers.GetLayer(i);
Console.WriteLine("第" + (i + 1) + "層：" + layer.Material.Name + "(" + layer.Thickness + "m)");
    }
    Console.WriteLine("熱貫流率=" + wLayers.GetOverallHeatTransferCoefficient().ToString("F1") + " W/(m2-K)");
    Console.WriteLine();

    //軽量コンクリートに変えてみる
    wLayers.SetLayer(2, new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.LightweightConcrete), 0.15));

    //結果書き出し
    Console.WriteLine("壁層の構成");
    for (uint i = 0; i < wLayers.LayerNumber; i++)
    {
WallLayers.Layer layer = wLayers.GetLayer(i);
Console.WriteLine("第" + (i + 1) + "層：" + layer.Material.Name + "(" + layer.Thickness + "m)");
    }
    Console.WriteLine("熱貫流率=" + wLayers.GetOverallHeatTransferCoefficient().ToString("F1") + " W/(m2-K)");

    Console.Read();
}

```

図 6.9　多層壁の熱貫流率計算テスト

```
壁層の構成
第 1 層:合板 (0.02m)
第 2 層:非密閉空気層 (0.01m)
第 3 層:コンクリート (0.15m)
第 4 層:漆喰 (0.01m)
熱貫流率 =3.3 W/(m2-K)

壁層の構成
第 1 層:合板 (0.02m)
第 2 層:非密閉空気層 (0.01m)
第 3 層:軽量コンクリート (0.15m)
第 4 層:漆喰 (0.01m)
熱貫流率 =2.5 W/(m2-K)
```

図 6.10　実行結果

6.2.5 Wall class

壁体の非定常熱伝導を計算するためには、Wall クラスを使用します。表 6.16 に Wall クラスのコンストラクタを示します。6.2.4 節で説明した WallLayer オブジェクトを与えることで壁層が初期化されます。

表 6.17 に Wall クラスの主要なプロパティを示します。この他、一般的な壁体の構築に関連する処理を表 6.18 に示します。

表 6.16　Wall コンストラクタの引数一覧

No.	引数 1	引数 2
1	壁の構成　(ImmutableWallLayers 型)	壁体名称　(string 型)
2	壁の構成　(ImmutableWallLayers 型)	-

表 6.17　Wall クラスの主要なプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
TimeStep	計算時間間隔	○	sec	double
SurfaceArea	壁表面積	○	m^2	double
Layers	壁層	-	-	-
AirTemperature1	1 側の乾球温度	○	°C	double
AirTemperature2	2 側の乾球温度	○	°C	double
Radiation1	1 側の放射量	○	W/m^2	double
Radiation2	2 側の放射量	○	W/m^2	double
Incline1	1 側の傾斜情報	-	-	-
Incline2	2 側の傾斜情報	-	-	-

1)　通常の熱伝導

壁体の熱伝導を解くにあたっては、壁体表面に入射する放射量と周辺の空気温度が必要となります。AirTemperature プロパティおよび Radiation プロパティを利用してこれらの情報を入力し、Update メソッドを呼ぶと TimeStep プロパティに設定された秒数が経過した状態を得ることができます。プログラム内部では、式 6.3 が後退差分によって解かれ、各壁層の温度が計算されます[2)]。壁体の質点は図 6.11 に示されるように各壁層の境界に設けられています。非定常熱伝導を解くプログラムの具体例を図 6.12 に示します。壁体の構成は表 6.19 の通りで、3.3.2 節で熱回路網を利用して解いたものと同一の構成です。プログラムの実行結果を図 6.13 と図 6.14 に示します。回路網を用いた場合とほぼ同様の結果が得られていることが確認できます。

$$\left.\frac{\partial T}{\partial t}\right|_m = \frac{1}{0.5(CAP_m + CAP_{m+1})}\left\{\frac{1}{R_{m+1}}(T_{m+1}-T_m)-\frac{1}{R_m}(T_m-T_{m-1})\right\} \tag{6.3}$$

T : 壁体温度[°C],　t : 時刻[sec],　M : 壁体分割数,
CAP : 単位面積あたりの熱容量 [J/(m²K)],　R : 節点間の熱容量 [(m²K)/W]

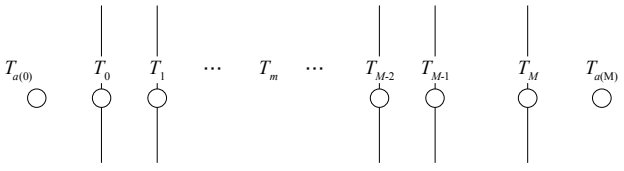

図 6.11　質点の位置

表 6.18　Wall クラスの主要メソッド

名称	内容			
Update	概要	壁体の状態を更新する		
	戻り値	-		
	引数 1	斜面情報 (ImmutableIncline 型)	引数 2	1 側か否か (bool 型)
SetIncline	概要	斜面情報を設定する。裏面は自動で設定される。		
	戻り値	-		
	引数 1	斜面情報 (ImmutableIncline 型)	引数 2	1 側か否か (bool 型)
GetSurface	概要	壁表面を取得する		
	戻り値	壁表面オブジェクト		
	引数 1	1 側か否か (bool 型)	引数 2	-
InitializeTemperature	概要	壁体内部の温度を初期化する		
	戻り値	-		
	引数 1	内部の温度 (double 型)	引数 2	-
SetOverallHeatTransfer Coefficient	概要	壁表面の総合熱伝達率 [W/(m^2 K)]を設定する		
	戻り値	-		
	引数 1	壁表面の総合熱伝達率 (double 型)	引数 2	1 側か否か
SetOverallHeatTransfer Coefficient	概要	壁表面の総合熱伝達率 [W/(m^2 K)]を設定する		
	戻り値	-		
	引数 1	壁表面 1 の総合熱伝達率 (double 型)	引数 2	壁表面 2 の総合熱伝達率 (double 型)
GetOverallHeatTransfer Coefficient	概要	壁表面の総合熱伝達率 [W/(m^2 K)]を取得する		
	戻り値	壁表面の総合熱伝達率 [W/(m^2 K)]		
	引数 1	1 側か否か	引数 2	-
SetConvectiveRate	概要	総合熱伝達率[W/m2-K]のうち、対流熱伝達の割合[-]を設定する		
	戻り値	-		
	引数 1	対流熱伝達の割合[-]	引数 2	-
SetRadiativeRate	概要	総合熱伝達率[W/m2-K]のうち、放射熱伝達の割合[-]を設定する		
	戻り値	-		
	引数 1	放射熱伝達の割合[-]	引数 2	-
GetConvectiveRate	概要	総合熱伝達率[W/m2-K]のうち、対流熱伝達の割合[-]を取得する		
	戻り値	対流熱伝達の割合[-]		
	引数 1	-	引数 2	-
GetRadiativeRate	概要	総合熱伝達率[W/m2-K]のうち、放射熱伝達の割合[-]を取得する		
	戻り値	放射熱伝達の割合[-]		
	引数 1	-	引数 2	-

表 6.19　計算例の壁構成

層番号	素材	熱伝導率 [W/(m K)]	容積比熱 [kJ/(m^3 K)]	厚み [mm]
1	合板	0.19	716	25
2	コンクリート	1.4	1934	120
3	空気層	熱抵抗 = 0.086 (m^2 K)/W		
4	ロックウール	0.042	84	50

```
/// <summary>壁熱貫流テスト</summary>
private static void wallHeatTransferTest()
{
    WallLayers layers = new WallLayers();
    WallLayers.Layer layer;
    layer = new WallLayers.Layer(new WallMaterial("合板", 0.19, 716), 0.025);
    layers.AddLayer(layer);
    layer = new WallLayers.Layer(new WallMaterial("コンクリート", 1.4, 1934), 0.120);
    layers.AddLayer(layer);
    layer = new WallLayers.Layer(new WallMaterial("空気層", 1d / 0.086, 0), 0.020);
    layers.AddLayer(layer);
    layer = new WallLayers.Layer(new WallMaterial("ロックウール", 0.042, 84), 0.050);
    layers.AddLayer(layer);
    Wall wall = new Wall(layers);

    wall.TimeStep = 3600;
    wall.AirTemperature1 = 20;
    wall.AirTemperature2 = 10;
    wall.InitializeTemperature(10); //壁体内温度は10℃均一とする
    wall.SurfaceArea = 1;

    Console.WriteLine("温度分布の推移");
    Console.WriteLine("合板, コンクリート, 空気層, ロックウール");
    double[] temps;
    for (int i = 0; i < 24; i++)
    {
wall.Update();
temps = wall.GetTemperatures();
Console.Write((i + 1).ToString("F0").PadLeft(2) + "時間後 | ");
for (int j = 0; j < temps.Length - 1; j++) Console.Write(((temps[j] + temps[j + 1]) / 2d).ToString("F1") + " | ");
Console.WriteLine();
    }

    //定常状態まで進める
    for (int i = 0; i < 1000; i++) wall.Update();
    Console.WriteLine();
    Console.WriteLine("定常状態の温度分布");
    temps = wall.GetTemperatures();
    for (int j = 0; j < temps.Length - 1; j++) Console.Write(((temps[j] + temps[j + 1]) / 2d).ToString("F1") + " | ");

    Console.WriteLine();
    Console.WriteLine("定常状態の熱流1: " + wall.GetHeatTransfer(true).ToString("F1"));
    Console.WriteLine("定常状態の熱流2: " + wall.GetHeatTransfer(false).ToString("F1"));
    Console.WriteLine("定常状態の熱流3: " + wall.GetStaticHeatTransfer().ToString("F1"));

    Console.Read();
}
```

図 6.12　多層壁の熱流計算プログラムの例

温度分布の推移
合板, コンクリート, 空気層, ロックウール
1 時間後 | 12.9 | 10.5 | 10.2 | 10.1 |
2 時間後 | 13.7 | 11.0 | 10.5 | 10.3 |
3 時間後 | 14.2 | 11.5 | 10.9 | 10.4 |
4 時間後 | 14.6 | 11.9 | 11.2 | 10.6 |
5 時間後 | 14.9 | 12.3 | 11.6 | 10.9 |
6 時間後 | 15.1 | 12.7 | 12.0 | 11.1 |
7 時間後 | 15.4 | 13.1 | 12.4 | 11.2 |
8 時間後 | 15.6 | 13.4 | 12.7 | 11.4 |
9 時間後 | 15.9 | 13.7 | 13.1 | 11.6 |
10 時間後 | 16.1 | 14.0 | 13.4 | 11.8 |
11 時間後 | 16.2 | 14.3 | 13.6 | 11.9 |
12 時間後 | 16.4 | 14.6 | 13.9 | 12.0 |
13 時間後 | 16.6 | 14.8 | 14.2 | 12.2 |
14 時間後 | 16.7 | 15.0 | 14.4 | 12.3 |
15 時間後 | 16.9 | 15.2 | 14.6 | 12.4 |
16 時間後 | 17.0 | 15.4 | 14.8 | 12.5 |
17 時間後 | 17.2 | 15.6 | 15.0 | 12.6 |
18 時間後 | 17.3 | 15.8 | 15.2 | 12.7 |
19 時間後 | 17.4 | 16.0 | 15.4 | 12.8 |
20 時間後 | 17.5 | 16.1 | 15.5 | 12.9 |
21 時間後 | 17.6 | 16.3 | 15.7 | 13.0 |
22 時間後 | 17.7 | 16.4 | 15.8 | 13.1 |
23 時間後 | 17.8 | 16.5 | 16.0 | 13.1 |
24 時間後 | 17.8 | 16.6 | 16.1 | 13.2 |

定常状態の温度分布
19.0 | 18.3 | 17.8 | 14.1 |
定常状態の熱流 1: 5.9
定常状態の熱流 2: -5.9
定常状態の熱流 3: 5.9

図 6.13　実行結果

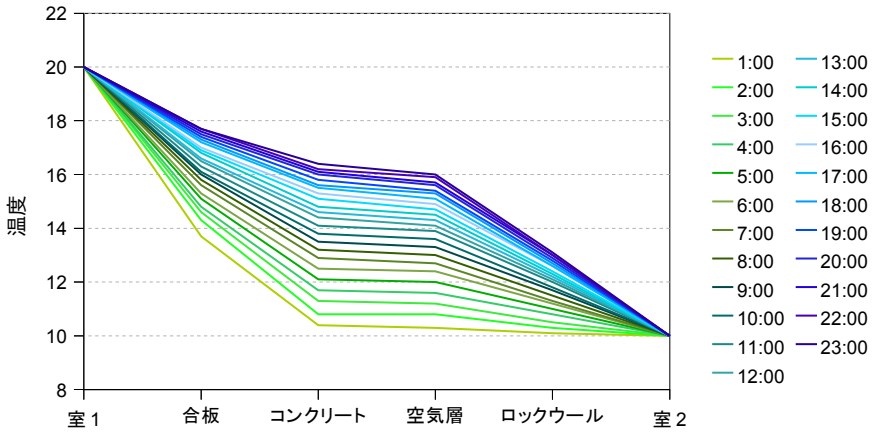

図 6.14　壁体内部の温度分布の推移

2) 冷温水配管が埋設されている場合の熱伝導

壁体内部に冷温水配管を埋設することで壁体を加熱あるいは冷却して、壁表面からの放射によって冷暖房を行う放射冷暖房システムが存在します。このようなシステムを計算する場合には壁体内部に埋め込まれる配管の仕様と、配管が埋設される位置を指定する必要があります。

配管を表現するクラスとして、Tube クラスが定義されています。Tube クラスのコンストラクタを以下に示します。引数はいずれも実数型で、それぞれ、冷温水管埋設層平均温度と冷温水温度との熱通過有効度[-]、フィン効率[-]、流体の比熱[J/(kg K)]を意味しています。これらの値は式 6.4~式 6.9 を利用して求めます。

Tube tube = new Tube(epsilon, finEfficiency, fluidSpecificHeat);

$$\frac{H_P}{A_F}=\frac{\varepsilon_{PNL}(c_w G_w)}{A_F}(T_{wi}-T_m) \tag{6.4}$$

$$\varepsilon_{PNL}=\frac{\varepsilon_{PX}}{1+(\varepsilon_{PX} c_w G_w / AC_f)(1/\eta_P-1)} \tag{6.5}$$

$$\varepsilon_{PX}=1-\exp\left(-\frac{K_P A_F}{c_w G_w}\right) \tag{6.6}$$

$$\eta_P=\frac{1}{w}\left\{D+(w-D)\frac{\tanh Z}{Z}\right\} \qquad Z=\frac{w-D}{2}\sqrt{\frac{C_f}{\lambda D}} \tag{6.7}$$

$$K_p=1/\left\{A_F/L\left(1/\pi D\alpha_w+R_b\right)\right\} \tag{6.8}$$

$$a_w=1057(1.352+0.0198\,T_m)v^{0.8}/D^{0.2} \tag{6.9}$$

H_P : 床からの発熱量[W], AF : 床面積[m^3], c_w : 水比熱[J/kg], G_w : 水流量[kg/s], T_{wi} : 冷温水温度[°C]
T_m : 冷温水管埋設層平均温度[°C], ε_{PNL} : 冷温水管埋設層平均温度と冷温水温度との熱通過有効度[-]
ε_{PX} : 管表面と冷温水温度との熱通過有効度[-], Cf : 冷温水管埋設層から隣接層への熱コンダクタンス[W/(m^2 K)]
η_P : フィン効率[-], K_P : 管表面から冷温水までの熱貫流率[W/(m^2 K)], w : 冷温水管設置間隔[m], D : 管径[m]
λ : 床スラブ熱伝導率[W/(m K)], L : 管の全長[m], α_w : 管内熱伝達率[W/(m^2 K)],
R_b : パネルと管の間の単位管長あたりの熱抵抗 [m K/W], T_m : 水温[°C], v : 管内流速 [m/s]

表 6.20 Tube クラスの主要メソッド

名称	内容			
SetFlowRate	概要	流体の流量[kg/s]を設定する		
	戻り値	-		
	引数 1	流体の流量[kg/s]	引数 2	-
GetOutletFluidTemperature	概要	チューブへの移動熱量[W]を指定して流体の出口温度[C]を計算する		
	戻り値	流体の出口温度[C]		
	引数 1	チューブへの移動熱量[W]	引数 2	-
GetOutletFluidTemperature	概要	流体の出口温度[C]を計算する		
	戻り値	流体の出口温度[C]		
	引数 1		引数 2	-

冷温水配管に関連する Wall クラスのメソッドを表 6.21 に示します。AddTube および RemoveTube メソッドで、作成した冷温水配管オブジェクトを壁体内に埋設します。また、Update メソッドを呼び出した後、GetHeatTransferToTube メソッドを使用することで、壁体から冷温水配管へ移動した熱量[W]を計算することができます。

表 6.21　Wall クラスの主要メソッド（冷温水配管関連）

名称	内容			
AddTube	概要	壁層に冷温水配管を埋め込む		
	戻り値	-		
	引数 1	冷温水配管 (Tube 型)	引数 2	壁層番号（壁体の分割数を加算した番号）
RemoveTube	概要	壁層から冷温水配管を取り外す		
	戻り値	-		
	引数 1	壁層番号	引数 2	-
GetHeatTransferToTube	概要	冷温水配管への熱移動量[W]を計算する		
	戻り値	冷温水配管への熱移動量[W]		
	引数 1	冷温水配管が埋設されている層の番号	引数 2	-

冷温水配管が埋設された床を計算するプログラムの例を図 6.16 に示します。図 6.15 に計算対象の床の構成と物性値を示します。熱容量の大きい水パックの間に冷温水配管が埋設されています。

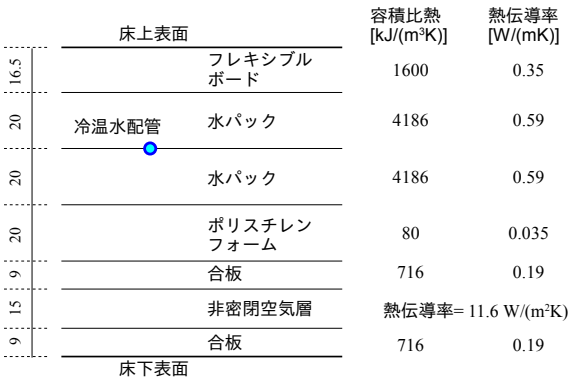

図 6.15　計算対象の床の構成と物性値

配管の素材は架橋ポリエチレンで、設置間隔は 267mm、外径は 3.4mm、内径は 2.3mm です。また、通過水量は 0.1L/min/本、加熱する床面積は 6.48m² です。フィン効率は以下のように計算できます。

$C_f = 1 / (0.02 / 0.59) + 1 / (0.02 / 0.59) = 59$ W/(m² K)

$Z = (0.0267 - 0.0034) \times (59 / (0.59 \times 0.0034))^{0.5} = 3.99$

$\eta_p = 1 / 0.0267 \times (0.0034 + (0.0267\text{-}0.0037)) \times \tanh(3.99) / 3.99 = 0.346$

また、架橋ポリエチレンの熱伝導率を 0.47 W/(m K)とすると、冷温水管埋設層平均温度と冷温水温度との熱通過有効度[-]は以下のように計算できます。

$v = (0.01 / 60 / 1000) / (3.14 \times (0.0034 / 2)^2) = 0.0184$ m/s

$\alpha_w = 10.57 \times (1.352 + 0.0198 \times 25) \times 0.0184^{0.8} \times 0.0034^{0.2} = 235.2$

$R_b = (0.0034 - 0.0023) / (3.14 \times 0.47 \times 0.0034) = 0.22$ m K / W

$K_p = 1 / (6.48 / 194.4 \times (1 / (3.14 \times 0.0034 \times 235.2) + 0.22)) = 48.6$ W/(m² K)

$\varepsilon_{px} = 1 - \exp(-48.6 \times 6.48 / 4186 / (0.01 \times 54 / 60)) = 0.999$

$\varepsilon_{PNL} = 0.999 / (1 + 0.999 \times 4186 \times (0.01 \times 54 / 60) / 6.48 / 59) \times 1 / 0.34 - 1) = 0.84$

プログラム例の 4~16 行では、冷温水配管を埋設する壁体を作成しています。18~22 行では上記の方法で求めたフィン効率と熱通過有効度を用いて配管オブジェクトを作成し、壁体に設置しています。壁体の計算間隔を 300 秒とし、100 回の繰り返し計算を行い、壁体内部の温度分布の変化を書き出しています。最初の 50 回は通水量を 0 とし、50~100 回の計算では、0.54L/min で 30°C の温水を通水しています。計算結果を図 6.17 に示します。通水を行っていない最初の 4 時間に関しては、壁体内部の温度が外部の温度に近づいていっていることがわかります。通水開始後は温水配管に隣接している壁温度 2 および壁温度 3 の温度が大きく上昇し、これに伴って、他の壁体の温度も少しずつ上昇することが確認できます。配管から壁体への熱移動量（GetHeatTransferToTube メソッドは配管への熱移動を正にとるため、計算結果は負の値）は温度差の大きい通水開始時が最も大きく、2kW 程度となっています。

```
1  /// <summary>壁熱貫流テスト（冷温水配管埋設）</summary>
2  private static void wallHeatTransferTest2()
3  {
4      WallLayers wl = new WallLayers();
5      wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.FrexibleBoard), 0.0165));
6      wl.AddLayer(new WallLayers.Layer(new WallMaterial("水", 0.59, 4186), 0.02));
7      wl.AddLayer(new WallLayers.Layer(new WallMaterial("水", 0.59, 4186), 0.02));
8      wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.ExtrudedPolystyreneFoam_3), 0.02));
9      wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.Plywood), 0.009));
10     wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.AirGap), 0.015));
11     wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.Plywood), 0.009));
12     Wall wall = new Wall(wl);
13     wall.TimeStep = 300;
14     wall.AirTemperature1 = 20;
15     wall.AirTemperature2 = 10;
16     wall.SurfaceArea = 6.48;
17
18     Tube tube = new Tube(0.84, 0.346, 4186);
19     //配管を埋設
20     wall.AddTube(tube, 1);
21     tube.SetFlowRate(0);  //最初は流量0
22     tube.FluidTemperature = 30;
23
24     wall.InitializeTemperature(20); //壁体温度を初期化
25
26     for (int i = 0; i < wall.Layers.LayerNumber; i++) Console.Write("温度" + i + ", ");
27     Console.WriteLine("配管への熱移動量[W], 配管出口温度[C]");
28     for (int i = 0; i < 100; i++)
29     {
30         if (i == 50) tube.SetFlowRate(0.54);  //通水開始
31         wall.Update();
32         double[] tmp = wall.GetTemperatures();
33         for (int j = 0; j < tmp.Length - 1; j++) Console.Write(((tmp[j] + tmp[j + 1]) / 2d).ToString("F1") + ", ");
34         Console.Write(wall.GetHeatTransferToTube(1).ToString("F0") + ", " + tube.GetOutletFluidTemperature().ToString("F1"));
35         Console.WriteLine();
36     }
37     Console.Read();
38 }
```

図 6.16　冷温水配管埋設壁の熱貫流計算プログラムの例

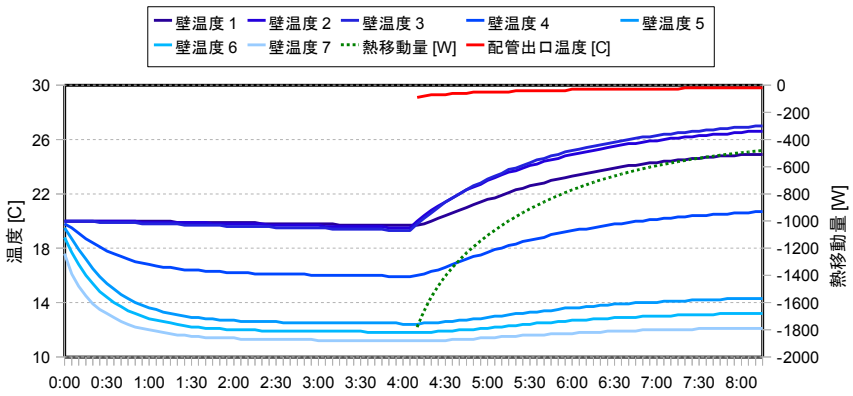

図 6.17　壁体内部の温度分布の推移（4:10 に通水開始）

3) 潜熱蓄熱材料がある場合の熱伝導

凝固熱や融解熱など、相変化に伴う熱移動を利用して大きな蓄熱を行うことができる材料を相変化材料（PCM : Phase Change Materials）と呼びます。相変化の前後で熱伝導率や容積比熱が変化するため、PCM が含まれた壁体の計算を行う場合には、相変化材料に関する特別な設定が必要となります。

PCM を表現するためには LatentHeatStorageMaterial クラスを使用します。PCM は、温度に応じて相が変化し、それぞれの相で熱伝導率と容積比熱が異なります。そこで、異なる物性を持った WallMaterial オブジェクトを作成した後、温度区分とともに LatentHeatStorageMaterial オブジェクトに設定します。概念を図 6.18 に示します。

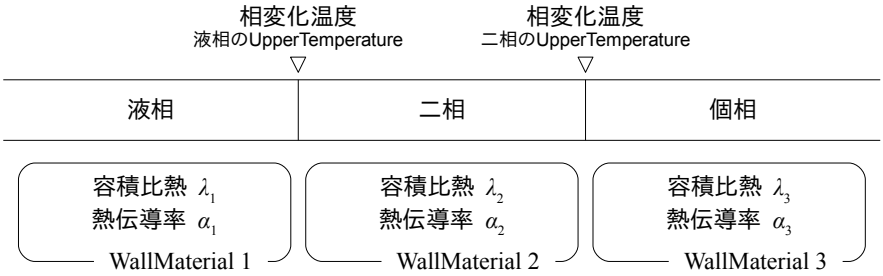

図 6.18　PCM の温度区分に応じた WallMaterial オブジェクトの設定

LatentHeatStorageMaterial クラスのコンストラクタは以下の通りです。第二引数は、壁体の材料を表す ImmutableWallMaterial 型のオブジェクトです。このオブジェクトの物性値は、第一引数で与えられた上限温度まで適用されます。

```
LatentHeatStorageMaterial lmat = new LatentHeatStorageMaterial(upperTemperature, material);
```

主要なメソッドを表 6.22 に示します。

表 6.22　LatentHeatStorageMaterial クラスの主要メソッド

名称	内容			
AddMaterial	概要	壁材料を追加する		
	戻り値	-		
	引数 1	温度上限 [C]	引数 2	壁材料
GetMaterial	概要	指定された温度の時の材料物性を取得する		
	戻り値	指定された温度の時の材料物性 (ImmutableWallMaterial 型)		
	引数 1	温度	引数 2	-
Initialize	概要	指定された温度に初期化する		
	戻り値	-		
	引数 1	温度	引数 2	-
GetHeatStorage	概要	温度 1 から温度 2 に変化した場合の蓄熱量[kJ/m^3]を計算する		
	戻り値	温度 1 から温度 2 に変化した場合の蓄熱量[kJ/m^3]		
	引数 1	温度 1	引数 2	温度 2

作成した LatentHeatStorageMaterial オブジェクトを壁体に設定するためには、表 6.23 に示す Wall クラスのメソッドを使用します。 SetLatentHeatStorageMaterial メソッドで潜熱蓄熱材料を壁体に設定し、RemoveLatentHeatStorageMaterial メソッドで設定を解除します。

表 6.23　Wall クラスの主要メソッド（潜熱蓄熱材設定関連）

名称	内容			
SetLatentHeat StorageMaterial	概要	潜熱蓄熱材料を設定する		
	戻り値	-		
	引数 1	壁層番号	引数 2	潜熱蓄熱材 (LatentHeatStorageMaterial 型)
RemoveLatentHeat StorageMaterial	概要	潜熱蓄熱材料の設定を解除する		
	戻り値	-		
	引数 1	壁層番号	引数 2	-

図 6.20 に、潜熱蓄熱材を含む壁体の非定常熱伝導を計算するプログラムの例を示します。2)で作成した壁体とほぼ同様の構成とし、第二層と第三層の水パックを潜熱蓄熱材で置き換えました。第二層と第三層に設定した潜熱蓄熱材の物性を表 6.24 に示します。

表 6.24　潜熱蓄熱材の物性

層番号	温度区分 [°C]	状態	容積比熱 [kJ/(m^3-K)]	熱伝導率 [W/(m-K)]
第二層	~19	固相	5040	0.19
	19~23	転移相	21140	0.205
	23~	液相	5040	0.22
第三層	~30	固相	5004	0.19
	30~32	転移相	88550	0.205
	32~	液相	4935	0.22

プログラム例では、24~38 行目で表 6.24 の潜熱蓄熱材料を作成し、壁体に設定しています。最初は配管に通水を行わず、初期温度の 35°C から壁体を冷却していきます。100 タイムステップ経過後に通水を開始するとともに壁体周囲の温度を 30°C に上げ、壁体を加熱しています。各タイムステップでは、壁体内部の各層の温度と、初期温度と比較した蓄熱量を書き出しています。

図 6.19 に計算結果を示します。波線が第三層に設置された潜熱蓄熱材の温度で、冷却時および加熱時ともに、転移相となる 30~32°C で非線形に温度変化が生じていることがわかります。また、この影響を受けて他の壁層でも温度変化が変曲していることが確認できます。

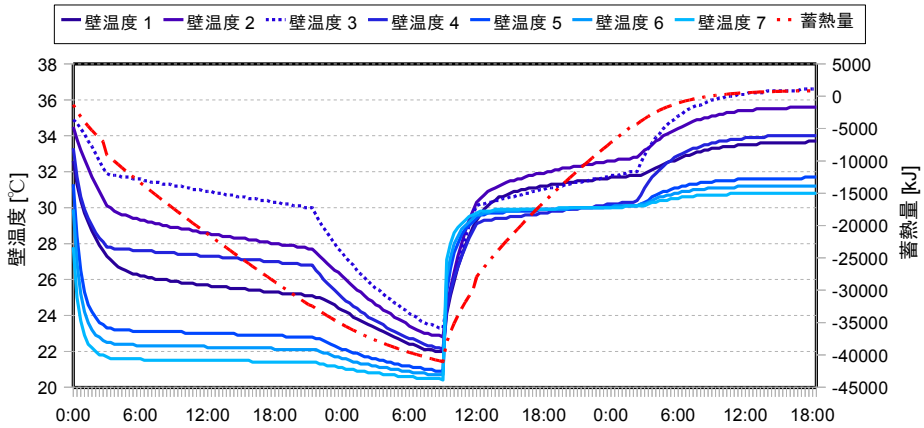

図 6.19　計算結果

```
/// <summary>壁熱貫流テスト（潜熱蓄熱材）</summary>
private static void wallHeatTransferTest3()
{
    //初期温度
    const double INIT_TEMP = 35;

    //壁層を作成
    WallLayers wl = new WallLayers();
    wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.FrexibleBoard), 0.0165));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial("ダミー材料", 1, 1), 0.02));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial("ダミー材料", 1, 1), 0.02));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.ExtrudedPolystyreneFoam_3), 0.02));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.Plywood), 0.009));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.AirGap), 0.015));
    wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.Plywood), 0.009));

    //壁体を作成
    Wall wall = new Wall(wl);
    wall.TimeStep = 1200;
    wall.AirTemperature1 = 20;
    wall.AirTemperature2 = 20;
    wall.SurfaceArea = 6.48;

    //潜熱蓄熱材1を作成して設定
    LatentHeatStorageMaterial material1;
    material1 = new LatentHeatStorageMaterial(19, new WallMaterial("パッシブ・スミターマル（凝固）", 0.19, 3.6 * 1400));
    material1.AddMaterial(23, new WallMaterial("パッシブ・スミターマル（遷移）", (0.19 + 0.22) / 2d, 15.1 * 1400));
    material1.AddMaterial(100, new WallMaterial("パッシブ・スミターマル（融解）", 0.22, 3.6 * 1400));
    material1.Initialize(INIT_TEMP);
    wall.SetLatentHeatStorageMaterial(1, material1);

    //潜熱蓄熱材2を作成して設定
    LatentHeatStorageMaterial material2;
    material2 = new LatentHeatStorageMaterial(30, new WallMaterial("スミターマル（凝固）", 0.19, 3.6 * 1390));
    material2.AddMaterial(32, new WallMaterial("スミターマル（遷移）", (0.19 + 0.22) / 2d, 63.25 * 1400));
    material2.AddMaterial(100, new WallMaterial("スミターマル（融解）", 0.22, 3.5 * 1410));
    material2.Initialize(INIT_TEMP);
    wall.SetLatentHeatStorageMaterial(2, material2);

    //潜熱蓄熱材の間に配管を埋設
    Tube tube = new Tube(0.84, 0.346, 4186);
    wall.AddTube(tube, 1);
    tube.SetFlowRate(0);
    tube.FluidTemperature = 40;

    //壁体温度を初期化
    wall.InitializeTemperature(INIT_TEMP);

    for (int i = 0; i < wall.Layers.LayerNumber; i++) Console.Write("温度" + i + ", ");
    Console.WriteLine("蓄熱量[kJ]");
    for (int i = 0; i < 200; i++)
    {
        if (i == 100)
        {
            tube.SetFlowRate(0.54); //通水開始
            wall.AirTemperature1 = 30;
            wall.AirTemperature2 = 30;
        }
        wall.Update();
        double[] tmp = wall.GetTemperatures();
        for (int j = 0; j < tmp.Length - 1; j++) Console.Write(((tmp[j] + tmp[j + 1]) / 2d).ToString("F1") + ", ");
        Console.Write(wall.GetHeatStorage(INIT_TEMP).ToString("F0"));
        Console.WriteLine();
    }
    Console.Read();
}
```

図 6.20　潜熱蓄熱材を含む壁体の非定常熱伝導を計算するプログラム例

6.2.6 室の温湿度変動計算に関するクラス

本クラスライブラリでは、室の温湿度変動を計算するために、Zone クラス、Room クラス、MultiRoom クラス、Outdoor クラス、IHeatGarin インターフェース、ISurface インターフェースを用意しています。図 6.21 に各クラスおよびインターフェースの概念を示します。

Room クラスは壁面や窓面に囲まれた領域を表現しています。この領域が均一でなく、一部で性状が異なる場合（例えばインテリアゾーン、ペリメータゾーンなど）、Zone クラスを利用して領域を分割することができます。Zone クラスで表現された領域は完全混合であるとみなされ、1 つの空気状態（MoistAir 型オブジェクト）で代表されます。また、Zone に属する熱負荷発生源は IHeatGain インターフェースで表現します。Room クラスに面している壁の表面や窓の表面を表現するためには ISurface クラスを利用します。壁表面の反対側は屋外の場合もありますが、別の Room が続いている可能性もあります。このように、壁体を挟んで連続する 2 以上の Room をまとめて取り扱うためのクラスが MultiRoom クラスです。一方、屋外を特徴付ける要素として太陽位置や外気状態がありますが、これらをまとめて取り扱うクラスとして Outdoor クラスが定義されています。

これらの内、Zone クラスまたは MultiRoom クラスに定義している Update メソッドを利用することで室温変動を解くことができます。両者の違いは、どの範囲までを連成させるかにあります。各ゾーンの温度はそのゾーンを囲む壁体の温度に依存します。しかし、これらの壁体の温度は、壁の反対側に存在する別のゾーンの温度や、そのゾーンに含まれる別の壁表面の温度に依存しています。また、室間換気量がある場合には、各ゾーンの空気状態は相互に影響を与えます。従って、厳密に連成を行い、熱収支を合わせるためには全ての壁表面の温度、ゾーンの温湿度を同時に解く必要があります。この場合には MultiRoom クラスの Update メソッドを利用します。一方で簡易化のために、壁の反対側のゾーンの相当温度を境界条件（例えば 1 タイムステップ前の状態値を与えておくなど）として計算を行うという方法も考えられます。この場合には、Zone クラスの Update メソッドを利用します。

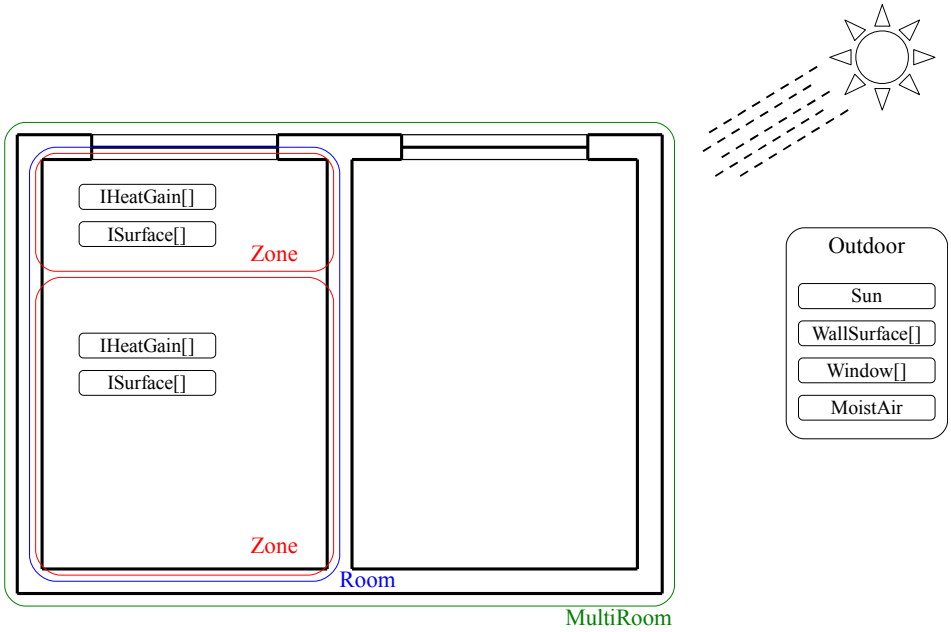

図 6.21 潜熱蓄熱材を含む壁体の非定常熱伝導を計算するプログラム例

以下に計算サンプルの建物の仕様と、Zone クラスおよび MultiRoom クラスを利用して計算を行う方法を示します。

1) 計算対象の建物

図 6.22 に計算対象の建物の平面図と断面図を示します。南側に窓を持つ直方体の室が東西に 2 つ連続したした、単純な形状の建物です。東側の室の窓には庇が設けられており、室内には発熱体が設置されています。庇の横幅は 5m、張り出しは 1m とし、窓の 0.5m 上方に設置されています。窓からの奥行きは 7m ですが、窓側 3m をペリメータゾーンとし、それ以外の領域をインテリアゾーンとして区別します。西側の室は 10CMH で外気が取り入れられていますが、ペリメータ側から給気し、インテリア側から排気することとします。東側の室では外気を取り入れず、ペリメータとインテリアで 10CMH でゾーン間換気を行うこととします。南面窓は Low-E の二重ガラスとし、大きさは 3m×2m とします。壁体は、内壁、外壁ともに 400mm のコンクリートとします。単純化のため、基礎は考慮せず、建物は地面に直接に置かれていることとし、地中温度は 25°C で一定に保たれることとします。

表 6.25、図 6.23、図 6.24 に、外界条件として利用する気象データを示します。2007 年 8 月 3 日に東京で観測された値です。

空調時間帯は 8:00~19:00 とし、無限大の容量を持つ空調機によって、乾球温度は 26°C、絶対湿度は 0.01kg/kg(DA) に保たれることとします。

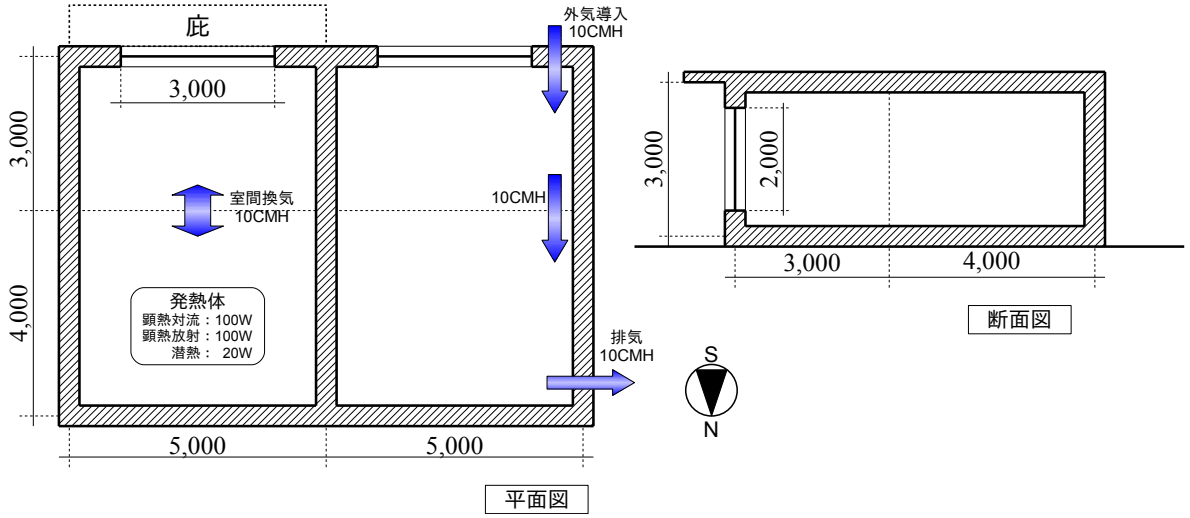

図 6.22　計算対象建物の平面図と断面図

表 6.25　計算用気象データ（東京：2007 年 8 月 3 日）

時刻	乾球温度[℃]	絶対湿度[kg/kg(DA)]	夜間放射[W/m²]	法線面直達日射[W/m²]	水平面天空日射[W/m²]
0:00	24.2	0.0134	32	0	0
1:00	24.1	0.0136	30	0	0
2:00	24.1	0.0134	30	0	0
3:00	24.2	0.0133	29	0	0
4:00	24.3	0.0131	26	0	0
5:00	24.2	0.0134	24	0	0
6:00	24.4	0.0138	24	106	36
7:00	25.1	0.0142	25	185	115
8:00	26.1	0.0142	25	202	198
9:00	27.1	0.0140	25	369	259
10:00	28.8	0.0147	24	427	314
11:00	29.9	0.0149	24	499	340
12:00	30.7	0.0142	24	557	340
13:00	31.2	0.0146	23	522	349
14:00	31.6	0.0140	24	517	319
15:00	31.4	0.0145	24	480	277
16:00	31.3	0.0144	24	398	228
17:00	30.8	0.0146	24	255	167
18:00	29.4	0.0142	23	142	87
19:00	28.1	0.0136	23	2	16
20:00	27.5	0.0136	24	0	0
21:00	27.1	0.0135	26	0	0
22:00	26.6	0.0136	25	0	0
23:00	26.3	0.0140	23	0	0

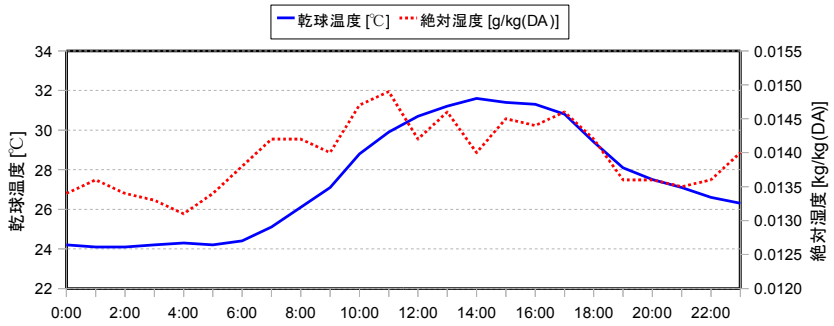

図 6.23　気象データ（乾球温度・絶対湿度）

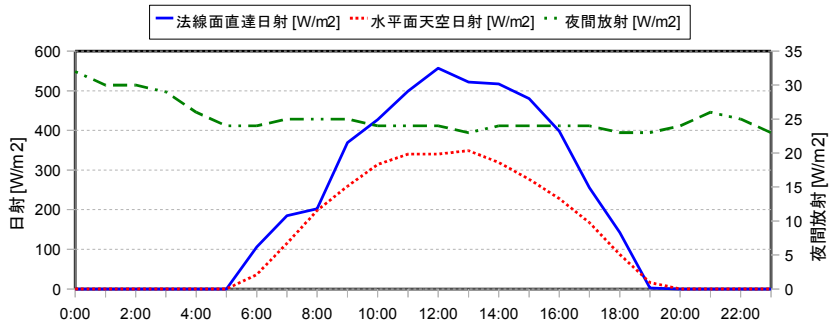

図 6.24　気象データ（日射・夜間放射）

2) Zone クラスを利用して解く方法

Zone クラスを利用すると、式 6.10 および式 6.11 で表現される室内空気の顕熱平衡式および潜熱平衡式†1)を解くことができます[2) 13)]。式 6.10 で表現される数式モデルの大きな特徴は右辺第一項にあります。厳密にはゾーンに面する窓や壁体の表面温度はそれぞれに異なりますが、本式では、平均の表面温度である T_{MRT} で一定として取り扱っています。また、表面とゾーンとの間の総合熱伝達率の対流成分も異なりますが、本式では $_{\alpha(i)}k_c$ で一定としています。

$$ZN_S\frac{dT_R}{dt}=\sum_{n=1}^{NW}A_n\alpha_{(i)}k_c(T_{MRT}-T_R)+c_aG_o(T_a-T_R)+HG_{(c)}-HE_s \tag{6.10}$$

$$ZN_S=c_a\rho\cdot VOL+CPF \tag{6.11}$$

$$ZN_L\frac{dx_R}{dt}=G_o(x_a-x_R)+LG-LE \tag{6.12}$$

$$ZN_L=\rho\cdot VOL+LCPF \tag{6.13}$$

ZN_S : ゾーンの顕熱容量 [J/K], T_R : ゾーンの乾球温度 [°C], t : 時間 [sec], NW : ゾーンに含まれる表面の数, A_n : 表面積 [m²]
$\alpha_{(i)}$: 総合熱伝達率 [W/(m² K)], k_c : 総合熱伝達率の対流比率 [-], T_{MRT} : 周壁の平均温度 [°C], c_a : 空気の比熱 [J/(kg K)]
G_o : 換気量 [kg/s], T_a : 外気の乾球温度 [°C], $HG_{(c)}$: 室内発生顕熱の対流成分 [W], HE_s : 室への供給熱量 [W]
ρ : 空気の密度 [kg/m³], VOL : 室容積 [m³], CPF : 家具などの熱容量 [J/K]
ZN_L : ゾーンの水蒸気容量 [kg], x_R : ゾーンの絶対湿度 [kg/kg(DA)], x_a : 外気の絶対湿度 [kg/kg(DA)],
LG : ゾーン内発生水蒸気量 [kg/s], LE : 水蒸気除去量 [kg/s], $LCPF$: 家具などの水蒸気容量 [kg]

Zone クラスの主要なプロパティとメソッドを表 6.26 と表 6.27 に示します。

SensibleHeatCapacity は空気以外の顕熱容量であり、家具などの熱容量です (上式の CPF)。オフィスの場合は単位容積あたりで 12~15 kJ/(m³ K) [14) 15)]、住宅の場合はその半分程度の値となります。

LatentHeatCapacity は空気以外の水蒸気容量であり、家具などの水蒸気容量です (上式の $LCPF$)。家具や壁体の吸放湿を無視するのであれば 0 としておきます†2)。

HeatTransferCoefficient はゾーンに属する壁や窓の表面の総合熱伝達率で、Zone クラスを利用して計算を行う場合にはすべての表面が同一の値をとると仮定します (上式の $\alpha_{(i)}$)。また、k_c を設定するためには SetConvectiveRate メソッドを使用します。

式 6.10 または式 6.11 を T_r または x_R について解けば室の乾球温度または絶対湿度が求まり、HE_s または LE について解けば顕熱負荷または潜熱負荷が求まります。これらの切り替えを行うためには、ControlDrybulbTemperature または ControlAbusoluteHumidity の設定を変更します。true に設定すると、乾球温度または絶対湿度が、DrybulbTemperatureSetPoint または AbsoluteHumiditySetPoint に指定された値で一定に保たれ、負荷が計算されます。false に設定した場合には、SensibleHeatSupply と LatentHeatSupply で設定された供給顕熱量と供給潜熱量を所与として、乾球温度または絶対湿度が変動することになります。

AtmosphericPressure は大気圧で、空気の密度に影響を与えます。Surfaces と HeatGains はそれぞれ、室に面している壁や窓の表面リストと室内発熱要素のリストです。これらは AddSurface や AddHeatGain メソッド等を利用して設定します。

†1) 参考文献 2)の宇田川の式に水蒸気容量の項を加えた
†2) 松尾陽 : 室の潜熱容量について, 日本建築学会大会学術講演梗概集, 1987, 等を参照して下さい。

表 6.26　Zone クラスの主要なプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
TimeStep	計算時間間隔	○	sec	double
Name	室名称	○	-	string
Volume	室容積	○	m^3	double
SensibleHeatCapacity	空気以外の顕熱容量[J/K]	○	J/K	double
SensibleHeatSupply	顕熱供給[W]（暖房を正とする）	○	W	double
LatentHeatCapacity	空気以外の水蒸気容量[kg]	○	kg	double
LatentHeatSupply	潜熱供給[W]（加湿を正とする）	○	W	double
HeatTransferCoefficient	室に属する壁窓面の総合熱伝達率	○	W/(m^2 K)	double
VentilationVolume	換気量	○	m^3/h	double
VentilationAirState	換気空気状態	○	-	MoistAir
ControlDrybulbTemperature	乾球温度を制御するか否か	○	-	bool
ControlAbsoluteHumidity	絶対湿度を制御するか否か	○	-	bool
DrybulbTemperatureSetPoint	乾球温度設定値	○	°C	double
AbsoluteHumiditySetPoint	絶対湿度設定値	○	kg/kg(DA)	dobule
Surfaces	室に面している壁窓面リスト	-	-	ImmutableSurface[ ]
HeatGains	室内発熱要素リスト	-	-	IheatGain[ ]
AtmosphericPressure	大気圧	○	kPa	double
CurrentDrybulbTemperature	現在の乾球温度	-	°C	double
CurrentAbsoluteHumidity	現在の絶対湿度	-	kg/kg(DA)	double
CurrentSensibleHeatLoad	現在の顕熱負荷	-	W	double
CurrentLatentHeatLoad	現在の潜熱負荷	-	W	double
CurrentMeanRadiantTemperature	現在の平均放射温度	-	°C	double
CurrentDateTime	現在の日時	○	-	DateTime

表 6.27　Zone クラスの主要なメソッド一覧

名称	内容			
AddSurface	概要	表面を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する表面 (ISurface 型)	引数 2	-
RemoveSurface	概要	表面を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する表面 (ISurface 型)	引数 2	-
AddWindow	概要	窓を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する窓 (Window 型)	引数 2	-
RemoveWindow	概要	窓を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する窓 (Window 型)	引数 2	-
AddHeatGain	概要	発熱要素を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する発熱要素 (IHeatGain 型)	引数 2	-
RemoveHeatGain	概要	発熱要素を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する発熱要素 (IHeatGain 型)	引数 2	-
SetConvectiveRate	概要	総合熱伝達率[W/(m^2 K)]のうち、対流熱伝達の割合[-]を設定する		
	戻り値	-		
	引数 1	対流熱伝達の割合[-]	引数 2	-
InitializeAirState	概要	温湿度を初期化する		
	戻り値	-		
	引数 1	乾球温度 [°C]	引数 2	絶対湿度 [kg/kg(DA)]

Zone クラスのモデル更新関連のメソッド一覧を表 6.28 に示します。Update メソッドを使用すると、TimeStep で指定した秒数が経過した後の室の状態を計算することができます。また、GetNext~~という名称のメソッドを利用すると、状態を更新させずに TimeStep 経過後の室の各種状態値を取得することができます。

表 6.28 Zone クラスのモデル更新関連のメソッド一覧

名称	内容			
Update	概要	TimeStep で設定された秒数を経過させ、状態を更新する		
	戻り値	-		
	引数 1	-	引数 2	-
GetNextDrybulbTemperature	概要	指定された顕熱供給[W]があった場合の室の乾球温度[°C]を計算する		
	戻り値	室の乾球温度[°C]		
	引数 1	顕熱供給[W]（暖房を正とする）	引数 2	-
GetNextAbsoluteHumidity	概要	指定された潜熱供給[W]があった場合の室の絶対湿度[kg/kg(DA)]を計算する		
	戻り値	室の絶対湿度[kg/kg(DA)]		
	引数 1	潜熱供給[W]（加湿を正とする）	引数 2	-
GetNextMeanRadiantTemperature	概要	現在の設定が継続した場合の、次タイムステップの周壁平均温度[°C]を計算する		
	戻り値	次タイムステップの周壁平均温度[°C]		
	引数 1	-	引数 2	-
GetNextSensibleHeatLoad	概要	指定された室乾球温度[°C]を達成するための顕熱供給[W]を計算する		
	戻り値	顕熱供給[W]		
	引数 1	室乾球温度[°C]	引数 2	-
GetNextLatentHeatLoad	概要	指定された室絶対湿度[kg/kg(DA)]を達成するための潜熱供給[W]を計算する		
	戻り値	潜熱供給[W]		
	引数 1	絶対湿度[kg/kg(DA)]	引数 2	-

ゾーンに面する壁や窓の表面近傍の相当温度は式 6.14 上段で表現されます。一方、屋外に面する壁や窓の表面近傍の相当温度は式 6.14 中段または下段で表現されます。中段は空気に面している場合、下段は地面に接する場合です。中段および下段の式を一括して計算するために、Outdoor クラスが用意されています。

Outdoor クラスのプロパティ一覧とメソッド一覧をを表 6.29 と表 6.30 に示します。Add~~という名称のメソッドと Remove~~という名称のメソッドを使用することで外表面を追加および削除することができます。その後、AirState、Sun、GroundTemperature、NocturnalRadiation プロパティに屋外の状態を設定し、SetWallSurfaceBoundaryState メソッドを呼び出すと、登録された外表面に式 6.14 に相当する情報が設定されます。

$$T_{SOL}=\begin{cases}\dfrac{a_{(c)}T_R+a_{(r)}T_{MRT}+RS}{a_{(i)}} \\ \dfrac{a_s I_W-\varepsilon F_S RN}{\alpha_o}+T_a \\ T_{GRZ}\end{cases} \tag{6.14}$$

T_{SOL} : 壁や窓表面近傍の相当温度 [°C], T_R : ゾーンの乾球温度 [°C], T_{MRT} : 周壁の平均温度 [°C], RS : 表面への放射 [W/m^2]
$\alpha_{(i)}$: 総合熱伝達率 [W/(m^2 K)], $\alpha_{(c)}$: 対流熱伝達率 [W/(m^2 K)], $\alpha_{(r)}$: 放射熱伝達率 [W/(m^2 K)]
a_s : 日射吸収率 [-], I_W : 日射 [W/m^2], ε : 放射率 [-], F_S : 外表面から天空への形態係数 [-], RN : 夜間放射 [W/m^2]
T_a : 外気乾球温度 [°C], T_{GRZ} : 地中温度 [°C]

表 6.29　Outdoor クラスの主要なプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
AirState	屋外の空気状態	○	-	MoistAir
Sun	太陽の情報	○	-	Sun
WallSurfaces	空気に面している壁表面リスト	○	-	ImmutableWallSurface []
GroundWallSurfaces	地面に接する壁表面リスト	-	-	ImmutableWallSurface []
Windows	窓リスト	-	-	ImmutableWindow []
GroundTemperature	地中温度	○	°C	double
NocturnalRadiation	夜間放射	○	W/m^2	double

表 6.30　Outdoor クラスのメソッド一覧

名称	内容			
SetWallSurfaceBoundaryState	概要	登録されている壁表面の境界条件を設定する		
	戻り値	-		
	引数 1	-	引数 2	-
AddWallSurface	概要	空気に面した壁表面を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する壁表面(WallSurface 型)	引数 2	-
RemoveWallSurface	概要	空気に面した壁表面を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する壁表面(WallSurface 型)	引数 2	-
AddGroundWallSurface	概要	地面に面した壁表面を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する壁表面(WallSurface 型)	引数 2	-
RemoveGroundWallSurface	概要	地面に面した壁表面を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する壁表面(WallSurface 型)	引数 2	-
AddWindow	概要	窓を追加する		
	戻り値	追加成功の真偽		
	引数 1	追加する窓 (Window 型)	引数 2	-
RemoveWindow	概要	窓を削除する		
	戻り値	削除成功の真偽		
	引数 1	削除する窓 (Window 型)	引数 2	-
GetRadiationToIncline	概要	斜面に入射する放射量[W/m^2]を計算する		
	戻り値	斜面に入射する放射量[W/m^2]		
	引数 1	斜面 (Incline 型)	引数 2	アルベド [-]
	引数 3	日影率 [-]	引数 4	-
SetConvectiveRate	概要	登録された壁表面の対流熱伝達の比率を設定する		
	戻り値	-		
	引数 1	対流熱伝達の比率	引数 2	-
SetOverallHeatTransferCoefficient	概要	表面の総合熱伝達率[W/(m^2 K)]を設定する		
	戻り値	-		
	引数 1	総合熱伝達率[W/(m^2 K)]	引数 2	対流熱伝達の比率

Zone クラスを利用して 1)で説明した多数室の計算を行うプログラムの例を図 6.27 に示します。
全体の大きな流れは以下の通りです。

1) 壁体(Wall)と窓(Window)を作成する
2) 壁体表面(WallSurface)と窓(Window)をゾーン(Zone)と屋外(Outdoor)に関連づける
3) 屋外の状態を更新する
4) 屋外の状態を登録された表面に設定する
5) 壁体を更新する
6) ゾーンを更新する
7) ゾーンの状態を壁表面に設定する
8) 3)に戻る

4~11 行では気象データを実数値配列に代入しています。14~17 行では屋外を表現する Outdoor オブジェクトを作成しており、太陽は東京で初期化しています。20~24 行では、壁表面や窓表面の向きを指定するために必要な傾斜面オブジェクトを作成しています。26~42 行は、ゾーンオブジェクトの作成処理です。48, 49 行目で作成した壁体の構成を利用し、54~166 行で壁体を作成しています。また、GetSurface メソッドを使用して壁体表面オブジェクトを取得し、ゾーンまたは屋外オブジェクトに設定を行っています。同様に、52 行で作成した窓構成を利用し、169~178 行で窓の作成および室への設定を行っています。180~240 行は計算の実行と書き出し処理です。定常状態になるように同一気象データで 100 回の繰り返し計算を行い、100 回目に書き出しを行っています。

本プログラムを実行した結果を図 6.25 と図 6.26 に示します。空調時間帯の 8:00~19:00 は乾球温度が一定となっており、顕熱負荷が計上されていることがわかります。空調開始時の 8:00 は相対的に負荷が大きくなっていますが、外気を導入している西側ペリメータに関しては外気温度の変化に合わせて負荷が上昇していることが確認できます。

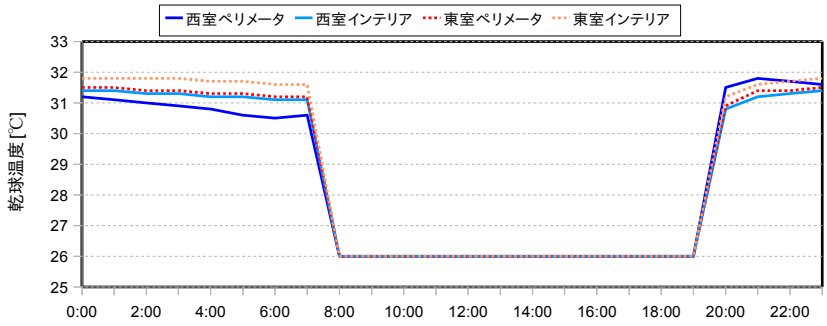

図 6.25　乾球温度の推移

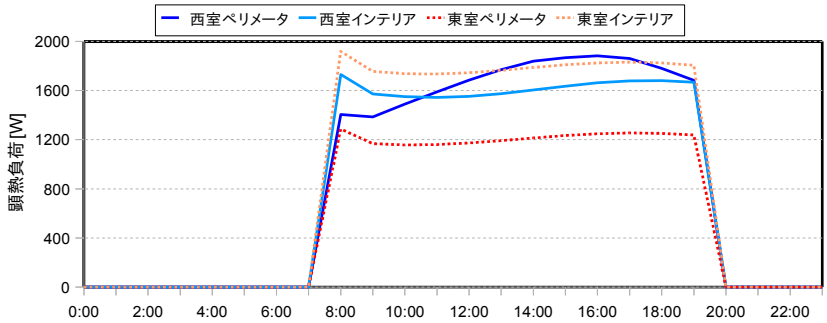

図 6.26　顕熱負荷の推移

```
/// <summary>室の温湿度変動テスト(Zone クラス)</summary>
private static void RoomModelTest1()
{
   //気象データ:乾球温度,絶対湿度,夜間放射,直達日射,天空日射
   double[] dbt = new double[] { 24.2, 24.1, 24.1, 24.2, 24.3, 24.2, 24.4, 25.1, 26.1, 27.1, 28.8, 29.9,
   30.7, 31.2, 31.6, 31.4, 31.3, 30.8, 29.4, 28.1, 27.5, 27.1, 26.6, 26.3 };
   double[] ahd = new double[] { 0.0134, 0.0136, 0.0134, 0.0133, 0.0131, 0.0134, 0.0138, 0.0142, 0.0142, 0.0140, 0.0147, 0.0149,
   0.0142, 0.0146, 0.0140, 0.0145, 0.0144, 0.0146, 0.0142, 0.0136, 0.0136, 0.0135, 0.0136, 0.0140 };
   double[] nrd = new double[] { 32, 30, 30, 29, 26, 24, 24, 25, 25, 25, 24, 24, 24, 23, 24, 24, 24, 24, 23, 23, 24, 26, 25, 23 };
   double[] dnr = new double[] { 0, 0, 0, 0, 0, 0, 106, 185, 202, 369, 427, 499, 557, 522, 517, 480, 398, 255, 142, 2, 0, 0, 0, 0 };
   double[] drd = new double[] { 0, 0, 0, 0, 0, 0, 36, 115, 198, 259, 314, 340, 340, 349, 319, 277, 228, 167, 87, 16, 0, 0, 0, 0 };

   //屋外を作成
   Outdoor outdoor = new Outdoor();
   Sun sun = new Sun(Sun.City.Tokyo);
   outdoor.Sun = sun;
   outdoor.GroundTemperature = 25;

   //傾斜を作成
   Incline nIn = new Incline(Incline.Orientation.N, 0.5 * Math.PI); //北
   Incline eIn = new Incline(Incline.Orientation.E, 0.5 * Math.PI); //東
   Incline wIn = new Incline(Incline.Orientation.W, 0.5 * Math.PI); //西
   Incline sIn = new Incline(Incline.Orientation.S, 0.5 * Math.PI); //南
   Incline hIn = new Incline(Incline.Orientation.S, 0);  //水平

   //ゾーンを作成
   Zone[] zones = new Zone[4];
   Zone wpZone = zones[0] = new Zone("西室ペリメータ");
   wpZone.Volume = 3 * 5 * 3;
   Zone wiZone = zones[1] = new Zone("西室インテリア");
   wiZone.Volume = 4 * 5 * 3;
   Zone epZone = zones[2] = new Zone("東室ペリメータ");
   epZone.Volume = 3 * 5 * 3;
   Zone eiZone = zones[3] = new Zone("東室インテリア");
   eiZone.Volume = 4 * 5 * 3;
   foreach (Zone zn in zones)
   {
      zn.VentilationVolume = 10; //換気量[CMH](ゾーン間換気もこのプロパティを援用する)
      zn.TimeStep = 3600;
      zn.DrybulbTemperatureSetPoint = 26;
      zn.AbsoluteHumiditySetPoint = 0.01;
   }

   //東側インテリアに発熱体を設定
   eiZone.AddHeatGain(new ConstantHeatGain(100, 100, 20));

   //壁構成を作成:400mmコンクリート
   WallLayers wl = new WallLayers();
   wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.ReinforcedConcrete), 0.4));

   //窓構成を作成
   GlassPanes gPanes = new GlassPanes(new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.HeatReflectingGlass06mm));

   //壁体をゾーンに追加
   Wall[] walls = new Wall[18];
   List<WallSurface> outdoorSurfaces = new List<WallSurface>();
   Wall wpwWall = walls[0] = new Wall(wl, "西室ペリメータ西壁");
   wpwWall.SurfaceArea = 3 * 3;
   outdoorSurfaces.Add(wpwWall.GetSurface(true));
   wpZone.AddSurface(wpwWall.GetSurface(false));
   wpwWall.SetIncline(wIn, true);

   Wall wpcWall = walls[1] = new Wall(wl, "西室ペリメータ天井");
   wpcWall.SurfaceArea = 3 * 5;
   outdoorSurfaces.Add(wpcWall.GetSurface(true));
   wpZone.AddSurface(wpcWall.GetSurface(false));
   wpcWall.SetIncline(hIn, true);

   Wall wpfWall = walls[2] = new Wall(wl, "西室ペリメータ床");
   wpfWall.SurfaceArea = 3 * 5;
   outdoor.AddGroundWallSurface(wpfWall.GetSurface(true));
   wpZone.AddSurface(wpfWall.GetSurface(false));

   Wall winWall = walls[3] = new Wall(wl, "西室インテリア北壁");
   winWall.SurfaceArea = 3 * 5;
   outdoorSurfaces.Add(winWall.GetSurface(true));
   wiZone.AddSurface(winWall.GetSurface(false));
   winWall.SetIncline(nIn, true);

   Wall wiwWall = walls[4] = new Wall(wl, "西室インテリア西壁");
```

```
    wiwWall.SurfaceArea = 3 * 4;
    outdoorSurfaces.Add(wiwWall.GetSurface(true));
    wiZone.AddSurface(wiwWall.GetSurface(false));
    wiwWall.SetIncline(wIn, true);

    Wall wicWall = walls[5] = new Wall(wl, "西室インテリア天井");
    wicWall.SurfaceArea = 4 * 5;
    outdoorSurfaces.Add(wicWall.GetSurface(true));
    wiZone.AddSurface(wicWall.GetSurface(false));
    wicWall.SetIncline(hIn, true);

    Wall wifWall = walls[6] = new Wall(wl, "西室インテリア床");
    wifWall.SurfaceArea = 4 * 5;
    outdoor.AddGroundWallSurface(wifWall.GetSurface(true));
    wiZone.AddSurface(wifWall.GetSurface(false));

    Wall epwWall = walls[7] = new Wall(wl, "東室ペリメータ東壁");
    epwWall.SurfaceArea = 3 * 3;
    outdoorSurfaces.Add(epwWall.GetSurface(true));
    epZone.AddSurface(epwWall.GetSurface(false));
    epwWall.SetIncline(eIn, true);

    Wall epcWall = walls[8] = new Wall(wl, "東室ペリメータ天井");
    epcWall.SurfaceArea = 3 * 5;
    outdoorSurfaces.Add(epcWall.GetSurface(true));
    epZone.AddSurface(epcWall.GetSurface(false));
    epcWall.SetIncline(hIn, true);

    Wall epfWall = walls[9] = new Wall(wl, "東室ペリメータ床");
    epfWall.SurfaceArea = 3 * 5;
    outdoor.AddGroundWallSurface(epfWall.GetSurface(true));
    epZone.AddSurface(epfWall.GetSurface(false));

    Wall einWall = walls[10] = new Wall(wl, "東室インテリア北壁");
    einWall.SurfaceArea = 5 * 3;
    outdoorSurfaces.Add(einWall.GetSurface(true));
    eiZone.AddSurface(einWall.GetSurface(false));
    einWall.SetIncline(nIn, true);

    Wall eiwWall = walls[11] = new Wall(wl, "東室インテリア東壁");
    eiwWall.SurfaceArea = 4 * 3;
    outdoorSurfaces.Add(eiwWall.GetSurface(true));
    eiZone.AddSurface(eiwWall.GetSurface(false));
    eiwWall.SetIncline(eIn, true);

    Wall eicWall = walls[12] = new Wall(wl, "東室インテリア天井");
    eicWall.SurfaceArea = 4 * 5;
    outdoorSurfaces.Add(eicWall.GetSurface(true));
    eiZone.AddSurface(eicWall.GetSurface(false));
    eicWall.SetIncline(hIn, true);

    Wall eifWall = walls[13] = new Wall(wl, "東室インテリア床");
    eifWall.SurfaceArea = 4 * 5;
    outdoor.AddGroundWallSurface(eifWall.GetSurface(true));
    eiZone.AddSurface(eifWall.GetSurface(false));

    Wall cpWall = walls[14] = new Wall(wl, "ペリメータ部の内壁");
    cpWall.SurfaceArea = 3 * 3;
    wpZone.AddSurface(cpWall.GetSurface(true));
    epZone.AddSurface(cpWall.GetSurface(false));

    Wall ciWall = walls[15] = new Wall(wl, "インテリア部の内壁");
    ciWall.SurfaceArea = 4 * 3;
    wiZone.AddSurface(ciWall.GetSurface(true));
    eiZone.AddSurface(ciWall.GetSurface(false));

    Wall wpsWall = walls[16] = new Wall(wl, "西側ペリメータ南壁");
    wpsWall.SurfaceArea = 5 * 3 - 3 * 2;
    outdoorSurfaces.Add(wpsWall.GetSurface(true));
    wpZone.AddSurface(wpsWall.GetSurface(false));
    wpsWall.SetIncline(sIn, true);

    Wall epsWall = walls[17] = new Wall(wl, "東側ペリメータ南壁");
    epsWall.SurfaceArea = 5 * 3 - 3 * 2;
    outdoorSurfaces.Add(epsWall.GetSurface(true));
    epZone.AddSurface(epsWall.GetSurface(false));
    epsWall.SetIncline(sIn, true);

    //外表面を初期化
    foreach (WallSurface ws in outdoorSurfaces)
    {
```

```
        //屋外に追加
        outdoor.AddWallSurface(ws);
        //放射率を初期化
        ws.InitializeEmissivity(WallSurface.SurfaceMaterial.Concrete);
    }

    //窓をゾーンに追加
    Window wWind = new Window(gPanes, "西室ペリメータ南窓");
    wWind.SurfaceArea = 3 * 2;
    wpZone.AddWindow(wWind);
    outdoor.AddWindow(wWind);

    Window eWind = new Window(gPanes, "東室ペリメータ南窓");
    eWind.SurfaceArea = 3 * 2;
    eWind.Shade = SunShade.MakeHorizontalSunShade(3, 2, 1, 1, 1, 0.5, sIn);
    wpZone.AddWindow(eWind);
    outdoor.AddWindow(eWind);

    //タイトル行書き出し
    StreamWriter sWriter = new StreamWriter("室の温湿度変動テスト1.csv", false, Encoding.GetEncoding("Shift_JIS"));
    foreach (Zone zn in zones) sWriter.Write(zn.Name + "乾球温度[C], " + zn.Name +
                                             "絶対湿度[kg/kgDA], " + zn.Name + "顕熱負荷[W], " + zn.Name + "潜熱負荷[W], ");
    sWriter.WriteLine();

    //計算実行
    for (int i = 0; i < 100; i++)
    {
        DateTime dTime = new DateTime(2007, 8, 3, 0, 0, 0);
        for (int j = 0; j < 24; j++)
        {
            //時刻を設定
            sun.Update(dTime);
            foreach (Zone zn in zones) zn.CurrentDateTime = dTime;

            //空調設定
            bool operating = (8 <= dTime.Hour && dTime.Hour <= 19);
            foreach (Zone zn in zones)
            {
                zn.ControlAbsoluteHumidity = operating;
                zn.ControlDrybulbTemperature = operating;
            }

            //気象条件を設定
            outdoor.AirState = new MoistAir(dbt[j], ahd[j]);
            outdoor.NocturnalRadiation = nrd[j];
            sun.SetGlobalHorizontalRadiation(drd[j], dnr[j]);

            //換気の設定
            eiZone.VentilationAirState = new MoistAir(epZone.CurrentDrybulbTemperature, eiZone.CurrentAbsoluteHumidity);
            epZone.VentilationAirState = new MoistAir(eiZone.CurrentDrybulbTemperature, eiZone.CurrentAbsoluteHumidity);
            wpZone.VentilationAirState = outdoor.AirState;
            wiZone.VentilationAirState = new MoistAir(wpZone.CurrentDrybulbTemperature, wpZone.CurrentAbsoluteHumidity);

            //外壁表面の状態を設定
            outdoor.SetWallSurfaceBoundaryState();

            //壁体を更新
            foreach (Wall wal in walls) wal.Update();

            //ゾーンを更新
            foreach (Zone zn in zones) zn.Update();

            //時刻を更新
            dTime = dTime.AddHours(1);

            //書き出し設定
            if (i == 99)
            {
                foreach (Zone zn in zones)
                {
                    sWriter.Write(zn.CurrentDrybulbTemperature.ToString("F1") + ", " + zn.CurrentAbsoluteHumidity.ToString("F3") + ", " +
                    zn.CurrentSensibleHeatLoad.ToString("F0") + ", " + zn.CurrentLatentHeatLoad.ToString("F0") + ", ");
                }
                sWriter.WriteLine();
            }
        }
    }

    sWriter.Close();
}
```

図 6.27　室の温湿度変動計算プログラムの例(Zone クラスを使用する場合)

3)　MultiRoom クラスを利用して解く方法

MultiRoom クラスを利用すると複数のゾーンの温湿度を連成させて解くことができます。

図 6.21 で示したとおり、本クラスライブラリでは空間を Zone, Room, MultiRoom という三層の階層構造で捉えています。Zone は対流による熱移動の範囲を表し、Room は放射による熱移動の範囲を表します。この概念を図 6.28 に示します。図 6.28 左に示すように、ある Zone を代表する質点と、対流によって熱移動が生じる範囲はそのゾーンに属している壁や窓表面のみです。この場合ですと、質点 Z1 と壁窓表面 W1, W2, W3 との間で対流による熱移動が生じています。質点 Z1 と壁窓表面 W4, W5, W6 との間では直接的に熱の授受はありません。一方、 Room には 1 以上の Zone が含まれますが、図 6.28 右に示すとおり、同一の Room に属する壁窓表面相互では放射による熱移動が生じるとしています。1 つの Room に壁体の両面を属させると放射による熱移動が計算できず、エラーが生じることに注意してください。

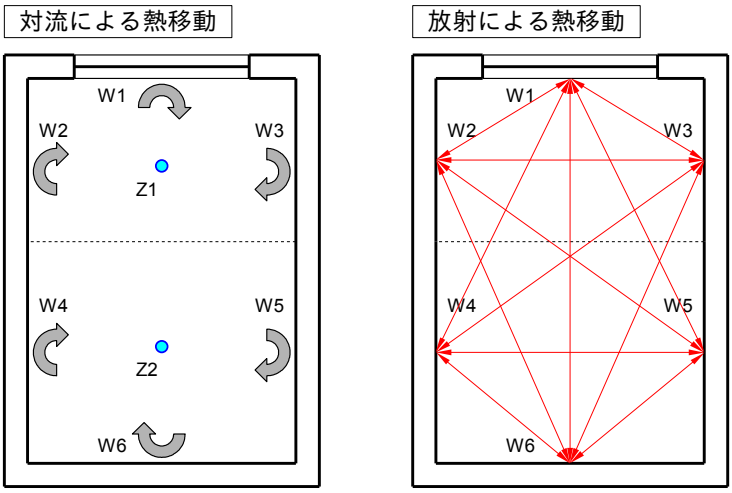

図 6.28　Zone および Room クラスの意味と熱移動の範囲

MultiRoom クラスを使用した場合の、ゾーンの顕熱平衡式を式 6.15 に示します。式 6.10 との違いは右辺第一項にあり、ゾーンに属する表面ごとに異なる表面温度と対流熱伝達率を使用しています[1]。

$$ZN_S \frac{d\,T_R}{dt} = \sum_{n=1}^{NW} A_n \alpha_{(i)n} k_{(c)n} (T_{Sn} - T_R) + c_a G_o (T_a - T_R) + HG_{(c)} - HE_s \tag{6.15}$$

ZN_S : ゾーンの顕熱容量 [J/K],　T_R : ゾーンの乾球温度 [°C],　t : 時間 [sec],　NW : ゾーンに含まれる表面の数,　A_n : 表面積 [m²]
$\alpha_{(i)n}$: 表面 n の総合熱伝達率 [W/(m² K)],　$k_{(c)n}$: 表面 n の総合熱伝達率の対流比率 [-]
T_{MRT} : 周壁の平均温度 [°C],　c_a : 空気の比熱 [J/(kg K)],　G_o : 換気量 [kg/s],　T_a : 外気の乾球温度 [°C]
$HG_{(c)}$: 室内発生顕熱の対流成分 [W],　HE_s : 室への供給熱量 [W]

Room クラスのコンストラクタを表 6.31 に示します。Room に属する Zone オブジェクトの配列が引数です。

表 6.31　Room コンストラクタの引数一覧

No.	引数 1	引数 2
1	Room に属する Zone (Zone 配列型)	Room 名称　(string 型)
2	Room に属する Zone (Zone 配列型)	-

†1) 詳細な解法については文献 2)を参照

Room クラスのプロパティおよびメソッドを表 6.32 と表 6.33 に示します。前述したとおり、Room クラスの主要な機能は放射による熱移動の設定にあり、プロパティおよびメソッドも放射関連の処理が中心となっています。

放射には、窓を透過して入射する日射による短波長放射と、室内の発熱要素が発生させる長波長放射が存在します。これらの放射は Room に属する窓や壁の表面に分配されますが、この分配比率を設定するため、SetShort(Long)WaveRadiationRate メソッドを使用します。設定を行わない場合には各表面の面積比で初期化されます。日射による短波長放射については、まずはペリメータ付近の床面に入射すると考えられますので、ペリメータ床面に大きめの比率を設定することが望ましいと考えられます。また、短波長放射の内、室内で反射を繰り返して窓表面に入射する割合は、そのまま窓を透過して室外へ出て行きます。この量は TransmissionHeatLossFromWindow で取得することができます。

Room に属する壁や窓表面相互の放射熱移動に関して、ある表面から別の表面への放射の比率は SetRadiativeHeatTransferRate で設定することができます。厳密には形態係数を利用した計算が必要になりますが、デフォルトでは各表面の面積を利用した式 6.16 で初期化されます。

$$\Phi_{nj} = \frac{A_j}{\sum_{l}^{NW} A_l} \tag{6.16}$$

Φ_{nj} : 表面 n から表面 j への放射熱交換係数 [-], A_j : 表面 j の面積 [m^2], NW : Room に属する表面の数 [sec]

表 6.32 Room クラスの主要なプロパティ一覧

プロパティ名称	内容	set アクセッサ	単位	型
Name	室の名称	○	-	string
SurfaceNumber	室に属する表面の数	-	-	uint
ZoneNumber	室に属するゾーンの数	-	-	uint
TransmissionHeatLossFromWindow	窓から外部への透過日射損失	-	W	double
CurrentDateTime	現在の日時	-	-	DateTime

表 6.33　Room クラスのメソッド一覧

名称	内容			
SetShortWaveRadiationRate	概要	窓面の短波長放射成分入射比率[-]を設定する		
	戻り値	-		
	引数 1	窓 (Window 型)	引数 2	窓面の短波長放射成分入射比率[-]
SetShortWaveRadiationRate	概要	短波長放射成分入射比率[-]を設定する		
	戻り値	-		
	引数 1	表面 (ISurface 型)	引数 2	表面の短波長放射成分入射比率[-]
GetShortWaveRadiationRate	概要	窓面の短波長放射成分入射比率[-]を取得する		
	戻り値	窓面の短波長放射成分入射比率[-]		
	引数 1	窓面 (Window 型)	引数 2	-
GetShortWaveRadiationRate	概要	短波長放射成分入射比率[-]を取得する		
	戻り値	表面の短波長放射成分入射比率[-]		
	引数 1	表面 (ISurface 型)	引数 2	-
SetLongWaveRadiationRate	概要	窓面の長波長放射成分入射比率[-]を設定する		
	戻り値	-		
	引数 1	窓 (Window 型)	引数 2	窓面の長波長放射成分入射比率[-]
SetLongWaveRadiationRate	概要	長波長放射成分入射比率[-]を設定する		
	戻り値	-		
	引数 1	表面 (ISurface 型)	引数 2	表面の長波長放射成分入射比率[-]
GetLongWaveRadiationRate	概要	窓面の長波長放射成分入射比率[-]を取得する		
	戻り値	窓面の長波長放射成分入射比率[-]		
	引数 1	窓面 (Window 型)	引数 2	-
GetLongWaveRadiationRate	概要	長波長放射成分入射比率[-]を取得する		
	戻り値	表面の長波長放射成分入射比率[-]		
	引数 1	表面 (ISurface 型)	引数 2	-
SetRadiativeHeatTransferRate	概要	表面 1 から表面 2 への放射熱交換係数[-]を設定する		
	戻り値	表面 1 から表面 2 への放射熱交換係数[-]		
	引数 1	表面 1 (ISurface 型)	引数 2	表面 2 (ISurface 型)
GetRadiativeHeatTransferRate	概要	表面 1 から表面 2 への放射熱交換係数[-]を取得する		
	戻り値	表面 1 から表面 2 への放射熱交換係数[-]		
	引数 1	表面 1 (ISurface 型)	引数 2	表面 2 (ISurface 型)

MultiRoom クラスのコンストラクタは、下記に示すように、引数として Room 型配列をとります。

MultiRoom mRoom = new MultiRoom(rooms);

MultiRoom クラスの主要なメソッドを表 6.34 に示します。SetAirFlow を使用することで Room をまたいで室間換気量を設定することができます。各室の状態を更新するためには UpdateRoomTemperatures および UpdateRoomHumidities を使用します。

表 6.34　MultiRoom クラスのメソッド一覧

名称	内容			
UpdateRoomTemperatures	概要	室の乾球温度を更新する		
	戻り値	-		
	引数 1	-	引数 2	-
UpdateRoomHumidities	概要	室の絶対湿度を更新する		
	戻り値	-		
	引数 1	-	引数 2	-
SetTimeStep	概要	計算時間間隔[s]を設定する		
	戻り値	-		
	引数 1	計算時間間隔[s]	引数 2	-
SetAirFlow	概要	室間換気量[m^3/h]を設定する		
	戻り値	-		
	引数 1	上流（空気が吹き出す側）の室	引数 2	下流（空気が吹き込む側）の室
	引数 3	室間換気量[m^3/h]	引数 4	-
GetAirFlow	概要	室間換気量[m^3/h]を取得する		
	戻り値	室間換気量[m^3/h]		
	引数 1	上流（空気が吹き出す側）の室	引数 2	下流（空気が吹き込む側）の室
SetCurrentDateTime	概要	現在の日時を設定する		
	戻り値	-		
	引数 1	現在の日時 (DateTime 型)	引数 2	-

MultiRoom クラスを使用して 1)で説明した多数室の計算を行うプログラムの例を図 6.29 に示します。多くの行は、図 6.27 に示した Zone を使用するプログラムと同じですが、一部、異なるところがあります。

179~183 行で Room オブジェクトおよび MultiRoom オブジェクトを作成しています。西室のペリメータ部およびインテリア部を合わせて西 Room、東室のペリメータ部およびインテリア部を合わせて東 Room としています。

185~189 行では室間換気量を設定しています。西室ペリメータ部についてのみ外気導入を設定していることに注意して下さい。

191~201 行では短波長放射の入射比率を設定しており、ペリメータ床の比率を 0.6、その他の表面については面積比率としています。

定常状態になるように同一気象データで 100 回の繰り返し計算を行い、100 回目に書き出しを行っています。

```
/// <summary>室の温湿度変動テスト(MultiRoom クラス)</summary>
private static void RoomModelTest2()
{
   //気象データ:乾球温度,絶対湿度,夜間放射,直達日射,天空日射
   double[] dbt = new double[] { 24.2, 24.1, 24.1, 24.2, 24.3, 24.2, 24.4, 25.1, 26.1, 27.1, 28.8, 29.9,
   30.7, 31.2, 31.6, 31.4, 31.3, 30.8, 29.4, 28.1, 27.5, 27.1, 26.6, 26.3 };
   double[] ahd = new double[] { 0.0134, 0.0136, 0.0134, 0.0133, 0.0131, 0.0134, 0.0138, 0.0142, 0.0142, 0.0140, 0.0147, 0.0149,
   0.0142, 0.0146, 0.0140, 0.0145, 0.0144, 0.0146, 0.0142, 0.0136, 0.0136, 0.0135, 0.0136, 0.0140 };
   double[] nrd = new double[] { 32, 30, 30, 29, 26, 24, 24, 25, 25, 25, 24, 24, 24, 23, 24, 24, 24, 24, 23, 23, 24, 26, 25, 23 };
   double[] dnr = new double[] { 0, 0, 0, 0, 0, 0, 106, 185, 202, 369, 427, 499, 557, 522, 517, 480, 398, 255, 142, 2, 0, 0, 0, 0 };
   double[] drd = new double[] { 0, 0, 0, 0, 0, 0, 36, 115, 198, 259, 314, 340, 340, 349, 319, 277, 228, 167, 87, 16, 0, 0, 0, 0 };

   //屋外を作成
   Outdoor outdoor = new Outdoor();
   Sun sun = new Sun(Sun.City.Tokyo);
   outdoor.Sun = sun;
   outdoor.GroundTemperature = 25;

   //傾斜を作成
   Incline nIn = new Incline(Incline.Orientation.N, 0.5 * Math.PI); //北
   Incline eIn = new Incline(Incline.Orientation.E, 0.5 * Math.PI); //東
   Incline wIn = new Incline(Incline.Orientation.W, 0.5 * Math.PI); //西
   Incline sIn = new Incline(Incline.Orientation.S, 0.5 * Math.PI); //南
   Incline hIn = new Incline(Incline.Orientation.S, 0);  //水平

   //ゾーンを作成
   Zone[] zones = new Zone[4];
   Zone wpZone = zones[0] = new Zone("西室ペリメータ");
   wpZone.Volume = 3 * 5 * 3;
   Zone wiZone = zones[1] = new Zone("西室インテリア");
   wiZone.Volume = 4 * 5 * 3;
   Zone epZone = zones[2] = new Zone("東室ペリメータ");
   epZone.Volume = 3 * 5 * 3;
   Zone eiZone = zones[3] = new Zone("東室インテリア");
   eiZone.Volume = 4 * 5 * 3;
   foreach (Zone zn in zones)
   {
      zn.TimeStep = 3600;
      zn.DrybulbTemperatureSetPoint = 26;
      zn.AbsoluteHumiditySetPoint = 0.01;
   }

   //東側インテリアに発熱体を設定
   eiZone.AddHeatGain(new ConstantHeatGain(100, 100, 20));

   //壁構成を作成:400mmコンクリート
   WallLayers wl = new WallLayers();
   wl.AddLayer(new WallLayers.Layer(new WallMaterial(WallMaterial.PredefinedMaterials.ReinforcedConcrete), 0.4));

   //窓構成を作成
   GlassPanes gPanes = new GlassPanes(new GlassPanes.Pane(GlassPanes.Pane.PredifinedGlassPane.HeatReflectingGlass06mm));

   //壁体をゾーンに追加
   Wall[] walls = new Wall[18];
   List<WallSurface> outdoorSurfaces = new List<WallSurface>();
   Wall wpwWall = walls[0] = new Wall(wl, "西室ペリメータ西壁");
   wpwWall.SurfaceArea = 3 * 3;
   outdoorSurfaces.Add(wpwWall.GetSurface(true));
   wpZone.AddSurface(wpwWall.GetSurface(false));
   wpwWall.SetIncline(wIn, true);

   Wall wpcWall = walls[1] = new Wall(wl, "西室ペリメータ天井");
   wpcWall.SurfaceArea = 3 * 5;
   outdoorSurfaces.Add(wpcWall.GetSurface(true));
   wpZone.AddSurface(wpcWall.GetSurface(false));
   wpcWall.SetIncline(hIn, true);

   Wall wpfWall = walls[2] = new Wall(wl, "西室ペリメータ床");
   wpfWall.SurfaceArea = 3 * 5;
   outdoor.AddGroundWallSurface(wpfWall.GetSurface(true));
   wpZone.AddSurface(wpfWall.GetSurface(false));

   Wall winWall = walls[3] = new Wall(wl, "西室インテリア北壁");
   winWall.SurfaceArea = 3 * 5;
   outdoorSurfaces.Add(winWall.GetSurface(true));
   wiZone.AddSurface(winWall.GetSurface(false));
   winWall.SetIncline(nIn, true);

   Wall wiwWall = walls[4] = new Wall(wl, "西室インテリア西壁");
   wiwWall.SurfaceArea = 3 * 4;
```

```
81      outdoorSurfaces.Add(wiwWall.GetSurface(true));
82      wiZone.AddSurface(wiwWall.GetSurface(false));
83      wiwWall.SetIncline(wIn, true);
84
85      Wall wicWall = walls[5] = new Wall(wl, "西室インテリア天井");
86      wicWall.SurfaceArea = 4 * 5;
87      outdoorSurfaces.Add(wicWall.GetSurface(true));
88      wiZone.AddSurface(wicWall.GetSurface(false));
89      wicWall.SetIncline(hIn, true);
90
91      Wall wifWall = walls[6] = new Wall(wl, "西室インテリア床");
92      wifWall.SurfaceArea = 4 * 5;
93      outdoor.AddGroundWallSurface(wifWall.GetSurface(true));
94      wiZone.AddSurface(wifWall.GetSurface(false));
95
96      Wall epwWall = walls[7] = new Wall(wl, "東室ペリメータ東壁");
97      epwWall.SurfaceArea = 3 * 3;
98      outdoorSurfaces.Add(epwWall.GetSurface(true));
99      epZone.AddSurface(epwWall.GetSurface(false));
100     epwWall.SetIncline(eIn, true);
101
102     Wall epcWall = walls[8] = new Wall(wl, "東室ペリメータ天井");
103     epcWall.SurfaceArea = 3 * 5;
104     outdoorSurfaces.Add(epcWall.GetSurface(true));
105     epZone.AddSurface(epcWall.GetSurface(false));
106     epcWall.SetIncline(hIn, true);
107
108     Wall epfWall = walls[9] = new Wall(wl, "東室ペリメータ床");
109     epfWall.SurfaceArea = 3 * 5;
110     outdoor.AddGroundWallSurface(epfWall.GetSurface(true));
111     epZone.AddSurface(epfWall.GetSurface(false));
112
113     Wall einWall = walls[10] = new Wall(wl, "東室インテリア北壁");
114     einWall.SurfaceArea = 5 * 3;
115     outdoorSurfaces.Add(einWall.GetSurface(true));
116     eiZone.AddSurface(einWall.GetSurface(false));
117     einWall.SetIncline(nIn, true);
118
119     Wall eiwWall = walls[11] = new Wall(wl, "東室インテリア東壁");
120     eiwWall.SurfaceArea = 4 * 3;
121     outdoorSurfaces.Add(eiwWall.GetSurface(true));
122     eiZone.AddSurface(eiwWall.GetSurface(false));
123     eiwWall.SetIncline(eIn, true);
124
125     Wall eicWall = walls[12] = new Wall(wl, "東室インテリア天井");
126     eicWall.SurfaceArea = 4 * 5;
127     outdoorSurfaces.Add(eicWall.GetSurface(true));
128     eiZone.AddSurface(eicWall.GetSurface(false));
129     eicWall.SetIncline(hIn, true);
130
131     Wall eifWall = walls[13] = new Wall(wl, "東室インテリア床");
132     eifWall.SurfaceArea = 4 * 5;
133     outdoor.AddGroundWallSurface(eifWall.GetSurface(true));
134     eiZone.AddSurface(eifWall.GetSurface(false));
135
136     Wall cpWall = walls[14] = new Wall(wl, "ペリメータ部の内壁");
137     cpWall.SurfaceArea = 3 * 3;
138     wpZone.AddSurface(cpWall.GetSurface(true));
139     epZone.AddSurface(cpWall.GetSurface(false));
140
141     Wall ciWall = walls[15] = new Wall(wl, "インテリア部の内壁");
142     ciWall.SurfaceArea = 4 * 3;
143     wiZone.AddSurface(ciWall.GetSurface(true));
144     eiZone.AddSurface(ciWall.GetSurface(false));
145
146     Wall wpsWall = walls[16] = new Wall(wl, "西側ペリメータ南壁");
147     wpsWall.SurfaceArea = 5 * 3 - 3 * 2;
148     outdoorSurfaces.Add(wpsWall.GetSurface(true));
149     wpZone.AddSurface(wpsWall.GetSurface(false));
150     wpsWall.SetIncline(sIn, true);
151
152     Wall epsWall = walls[17] = new Wall(wl, "東側ペリメータ南壁");
153     epsWall.SurfaceArea = 5 * 3 - 3 * 2;
154     outdoorSurfaces.Add(epsWall.GetSurface(true));
155     epZone.AddSurface(epsWall.GetSurface(false));
156     epsWall.SetIncline(sIn, true);
157
158     //外表面を初期化
159     foreach (WallSurface ws in outdoorSurfaces)
160     {
161       //屋外に追加
```

```
162         outdoor.AddWallSurface(ws);
163         //放射率を初期化
164         ws.InitializeEmissivity(WallSurface.SurfaceMaterial.Concrete);
165     }
166
167     //窓をゾーンに追加
168     Window wWind = new Window(gPanes, "西室ペリメータ南窓");
169     wWind.SurfaceArea = 3 * 2;
170     wpZone.AddWindow(wWind);
171     outdoor.AddWindow(wWind);
172
173     Window eWind = new Window(gPanes, "東室ペリメータ南窓");
174     eWind.SurfaceArea = 3 * 2;
175     eWind.Shade = SunShade.MakeHorizontalSunShade(3, 2, 1, 1, 1, 0.5, sIn);
176     wpZone.AddWindow(eWind);
177     outdoor.AddWindow(eWind);
178
179     //多数室オブジェクトを作成
180     Room eRm = new Room(new Zone[] { epZone, eiZone }); //東側の室
181     Room wRm = new Room(new Zone[] { wpZone, wiZone }); //西側の室
182     MultiRoom mRoom = new MultiRoom(new Room[] { eRm, wRm }); //多数室
183     mRoom.SetTimeStep(3600);
184
185     //換気の設定
186     wpZone.VentilationVolume = 10; //西室ペリメータのみ外気導入
187     mRoom.SetAirFlow(wpZone, wiZone, 10);
188     mRoom.SetAirFlow(epZone, eiZone, 10);
189     mRoom.SetAirFlow(eiZone, epZone, 10);
190
191     //短波長放射の入射比率を調整:ペリメータ床面6割、その他は面積比率
192     double sfSum = 0;
193     foreach (ISurface isf in eRm.GetSurface()) sfSum += isf.Area;
194     sfSum -= epfWall.SurfaceArea;
195     foreach (ISurface isf in eRm.GetSurface()) eRm.SetShortWaveRadiationRate(isf, isf.Area / sfSum * 0.4);
196     eRm.SetShortWaveRadiationRate(epfWall.GetSurface(false), 0.6);
197     sfSum = 0;
198     foreach (ISurface isf in wRm.GetSurface()) sfSum += isf.Area;
199     sfSum -= wpfWall.SurfaceArea;
200     foreach (ISurface isf in wRm.GetSurface()) wRm.SetShortWaveRadiationRate(isf, isf.Area / sfSum * 0.4);
201     wRm.SetShortWaveRadiationRate(wpfWall.GetSurface(false), 0.6);
202
203     //タイトル行書き出し
204     StreamWriter sWriter = new StreamWriter("室の温湿度変動テスト2.csv", false, Encoding.GetEncoding("Shift_JIS"));
205     foreach (Zone zn in zones) sWriter.Write(zn.Name + "乾球温度[C], " + zn.Name + "絶対湿度[kg/kgDA], "
206                     + zn.Name + "顕熱負荷[W], " + zn.Name + "潜熱負荷[W], ");
207     sWriter.WriteLine();
208
209     //計算実行
210     for (int i = 0; i < 100; i++)
211     {
212         DateTime dTime = new DateTime(2007, 8, 3, 0, 0, 0);
213         for (int j = 0; j < 24; j++)
214         {
215             //時刻を設定
216             sun.Update(dTime);
217             mRoom.SetCurrentDateTime(dTime);
218
219             //空調設定
220             bool operating = (8 <= dTime.Hour && dTime.Hour <= 19);
221             foreach (Zone zn in zones)
222             {
223                 zn.ControlAbsoluteHumidity = operating;
224                 zn.ControlDrybulbTemperature = operating;
225             }
226
227             //気象条件を設定
228             outdoor.AirState = new MoistAir(dbt[j], ahd[j]);
229             outdoor.NocturnalRadiation = nrd[j];
230             sun.SetGlobalHorizontalRadiation(drd[j], dnr[j]);
231
232             //換気の設定
233             wpZone.VentilationAirState = outdoor.AirState;
234
235             //外壁表面の状態を設定
236             outdoor.SetWallSurfaceBoundaryState();
237
238             //壁体を更新
239             foreach (Wall wal in walls) wal.Update();
240
241             //多数室を更新
242             mRoom.UpdateRoomTemperatures();
```

```
243             mRoom.UpdateRoomHumidities();
244
245             //時刻を更新
246             dTime = dTime.AddHours(1);
247
248             //書き出し設定
249             if (i == 99)
250             {
251                 foreach (Zone zn in zones)
252                 {
253                     sWriter.Write(zn.CurrentDrybulbTemperature.ToString("F1") + ", " + zn.CurrentAbsoluteHumidity.ToString("F3") + ", " +
254                     zn.CurrentSensibleHeatLoad.ToString("F0") + ", " + zn.CurrentLatentHeatLoad.ToString("F0") + ", ");
255                 }
256                 sWriter.WriteLine();
257             }
258         }
259     }
260
261     sWriter.Close();
262 }
263
```

図 6.29　室の温湿度変動計算プログラムの例(MultiRoom クラスを使用する場合)

図 6.30 と図 6.31 に計算結果を示します。MultiRoom では室間換気を考慮した連成計算を行っているため、西側ペリメータで導入した外気の影響がインテリア部にも表れており、Zone クラスのみを使用した場合の図 6.26 と比較すると、インテリア部の熱負荷が上昇していることがわかります。

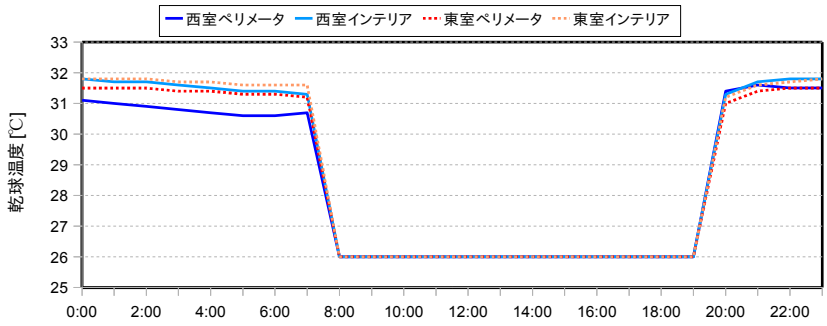

図 6.30　乾球温度の推移

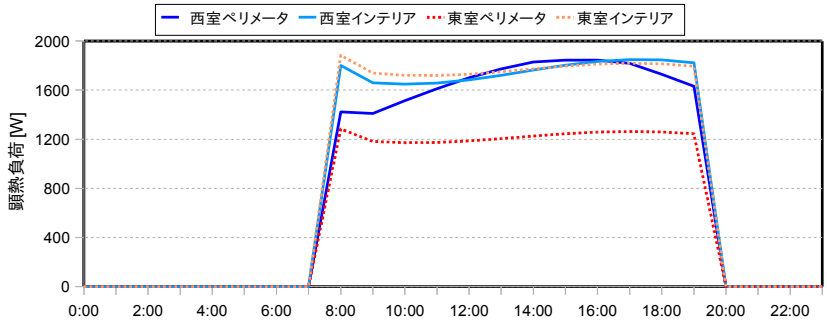

図 6.31　顕熱負荷の推移

1) Daniel R. Clark : HVACSIM building systems and equipment simulation program reference manual
2) 宇田川光弘 : パソコンによる空気調和計算法, オーム社, 1986
3) Hiroyasu Okuyama : Bouilding Heat and Air Transfer Models Based on System Theory, Bulletin of the Japan Society for Industrial and Applied Mathematics, 13(1), pp.61-71, 2003
4) A.P.Gagge et al. : A standard predictive index of human response to the thermal environment, AHRAE Transaction, Vol.93, 1987
5) Berlage,Von H.P.:Zur Theorie der Beleuchtung einer horizontalen Flache durch Tageslicht,Meteorologische Zeitschrift, May 1928,pp.174-180
6) 松尾陽:日本建築学会論文報告集,快晴時の日射について 日射量に関する研究 2,pp.21-24,1960
7) 永田忠彦:晴天空による水平面散乱の日射の式の試案,日本建築学会学術講演梗概集,1978
8) Liu,B.Y.H & Jordan,R.C:The interrelationship and characteristic distribution of direct, diffuse and total solar radiation, solar energy, Vol.4, No.3, 1960
9) 宇田川光弘,木村建一:水平面全天日射量観測値よりの直達日射量の推定,日本建築学会論文報告集,No.267,pp.83-90,1978, 0530
10) 渡辺俊行:水平面全天日射量の直散分離と傾斜面日射量の推定,日本建築学会論文報告集,No.330,pp.96-108,19830830
11) H.Akasaka:Model of circumsolar radiation and diffuse sky radiation including cloudy sky, ISES, Solar World Congress, 1991
12) 三木信博:標準気象データの日射直散分離に関する研究 その 6 日射直散分離法の提案,日本建築学会学術講演梗概集,pp.857-858,1991
13) 松尾陽 : 空調負荷計算におけるふく射伝熱の扱い, 空気調和・衛生工学, 59-4, 1985, pp.323~329
14) 木村建一, 伊藤直明 : 事務所建築の家具の熱容量, 日本建築学会関東支部第 29 会研究発表会, 1961-1, 同続報, 同第 34 回, 1963-5
15) 石野久彌, 郡公子 : 事務所建築における家具類の熱的影響に関する実測・実験研究, 日本建築学会計画系論文報告集 (372), 59-66, 1987, 0228