Multiscale models for assimilation data and forecast : GE Echeladata challenge

Valentin Lefranc, Étienne Gay, Emmanuel Frenod, Marianne Hemous, Remy Fouchereau

Multiscale models for assimilation data and forecast : *GE Echeladata challenge*[1]

Valentin Lefranc[a,b], Étienne Gay[a], Émmanuel Frenod[a,b], Marianne Hemous[a], Rémy Fouchereau[a,b]

[a] *See-d, 6 bis, rue Henri Becquerel - CP 101 56038 Vannes Cedex, https://www.see-d.fr*
[b] *Université de Bretagne Sud, Laboratoire de Mathématiques de Bretagne Atlantique, UMR CNRS 6205, Campus de Tohannic, Vannes, France*

Keywords: Science, Machine Learning, Smart building

1. Introduction

1.1. Objectives of the project

The purpose of this work is to show the possibility of creating a multiscale tool enabled to analyse data from smart buildings. Here we explore the coupling model/data using Machine Learning algorithms. The model used is the heat diffusion equation and the data come from simulations. To recreate real life sensor data, we added noise to the data after running the simulations.

1.2. Challenges

The main challenge is to build a link between the variables and the different scales considered here. The simulations will help us to fix this point thanks to the huge amount of data we will be able to generate. The final challenge is to have a model that predicts accurate results.

1.3. Technological choices

The technologies we use in this project are the followings:

[1] Sponsored by General Electric, Teratec and Bpifrance

- Python[2] language is used because it can be object oriented and executable. Python is also very useful for scientific simulation and provides a lot of important packages such as Numpy[3], Pandas[4] or Scipy[5]. Finally, packages for Machine Learning algorithms are already developped for python in *scikit learn* libraries [6].

- MongoDB[7]: to store all the data generated by our simulation, we choose to use a noSQL database technology.

- Web services to build the API.

2. Theory

2.1. Heat diffusion and convection

Our model is based on the use of the diffusion equation in a homogeneous and isotropic medium with constant thermodynamics coefficients :

$$\frac{\partial T}{\partial t} = \alpha(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}) \tag{1}$$

where :

- α is a real coefficient called the thermal diffusivity.

- $T = T(t, x, y, z)$ is temperature as a function of space and time.

Diffusion is taken into account following the classical finite differences approximation (3D). The temperature at time $t + \Delta_t$ (note $T(t + \Delta_t; i; j; k)$) depends on 2 main things:

1. An evaluation of $\frac{\partial T(t;x;y;z)}{\partial t} \simeq \frac{T(t+\Delta_t;x;y;z)-T(t;x;y;z)}{\Delta_t}$

[2] www.python.org/
[3] www.numpy.org/
[4] pandas.pydata.org/
[5] www.scipy.org/
[6] scikit-learn.org/stable/
[7] www.mongodb.com

2. An evaluation of the Laplacian operator with the use of Taylor's approximation. For example, the derivative order following x can be approximated with the following equation:

$$\frac{\partial^2 T(t;x;y;z)}{\partial x^2} = \frac{T(t;x+h;y;z) - 2T(t;x;y;z) + T(t;x-h;y;z)}{h^2} + h\epsilon(x,h) \tag{2}$$

which leads to (3) when the Laplacian is calculated on 3 dimensions.

The temperature at a given point $(i;j;k)$ corresponding to the $(x;y;z)$ axes is given by the following numerical approximation :

$$\begin{aligned} T(t+\Delta_t;i;j;k) \simeq & T(t;i;j;k) + \cdots \\ & \alpha\Delta_t\left(\frac{T(t;i+1;j;k) + T(t;i-1;j;k) - 2T(t;i;j;k)}{\Delta_x^2} + \cdots\right. \\ & \frac{T(t;i;j+1;k) + T(t;i;j-1;k) - 2T(t;i;j;k)}{\Delta_y^2} + \cdots \\ & \left.\frac{T(t;i;j;k+1) + T(t;i;j;k-1) - 2T(t;i;j;k)}{\Delta_z^2}\right) \end{aligned} \tag{3}$$

The aim is to have a quick approximate solution of the heat fluctuations over time. Each room is only seen as a regular mesh with only a few points ($\simeq$ 1000 for a room). Every heat source (radiant, window, door) takes 5-6 points from the mesh. To take into account convection, we just have to implement the following approximation of the first derivate following the z axe:

$$T(t+\Delta_t;i;j;k) \simeq T(t;i;j;k) + \beta\Delta_t\left(\frac{T(t;i;j;k+1) - T(t;i;j;k-1)}{2 * \Delta_z}\right) \tag{4}$$

2.2. Machine learning

Pure physical models are generally difficult to set up. They have a lot of parameters to evaluate, a slight variation in the initial conditions can lead to big differences in the final result. The use of Machine Learning algorithms will help us to overcome these problems:

- We can use a simpler physical modeling. The gaps between the real world and the physical model will be learned through a self-learning algorithm. If large differences appear, they will be filled following the assimilation of data from ground return. The improvements will not be the result of ever more complexity but the integration of multiple data.

- As the physical model is simpler it will present fewer parameters, which means a smaller dimensionality. The evaluation of the parameters will therefore be simplified. In the end, the process will be transferable from one building to another without major modification of the physical part of the model. Only the layout of the rooms will have to be changed. The data will make the adjustments.

- Classical Machine Learning methods make it easy to take into account non-linearities (unlike classical statistical models for example). That is to say we don't need to make assumptions about the shape of the model errors.

All these advantages have a flaw, they require a lot of data in order to learn the different underlying schemes. This defect will be partially circumvented with the use of at least one sensor per room.

The aim will be to determine if a window (for example) is open by comparing the states of the physical simulations to the states of the sensors.

In the next sub-section, we will discuss the main Machine Learning method used in this article.

2.2.1. Random forests

Random forests were introduced by Breiman in 2001 and are based on the classification and regression trees (Breiman and al. 1984). It's an estimator that fits a huge number of regression trees on various sub-samples of the dataset. Each tree is constructed taking a small number of variables (columns) and a small number of lines ($\approx\sqrt{n}$).

A tree is constructed by recursively partitioning the data space (with an inter-class variance criterion) and fitting a simple prediction model within each partition.

In the end, the prediction is given by averaging the results on all trees. Random forests have the advantage over a single tree to control over-fitting, it also gives good predictive accuracy in practice. In fact the average of the

errors of these forest predictions is not larger than to the average tree grown from the entire dataset.

In our case, the inputs of the model will therefore be the data of the sensors at a given time as well as the corresponding derivatives. The predicted values will be an estimate of the given state of the building (which doors are open, which radiators are lit).

3. Decomposition of the house into three scales

The model will consist in three scales: The macroscopic scale is the one of an entire building seen as an object, the meso-scopic scale describes the structure of the building (architecture) and finally the microscopic scale represents the rooms and its elements (windows, door, radiator, air conditioner)

3.1. Micro-scale

The microscopic scale details the behaviour of each room. It is the place where partial derivative equation are resolved in order to determine the heat evolution. A set of state variables are associated with this scale.

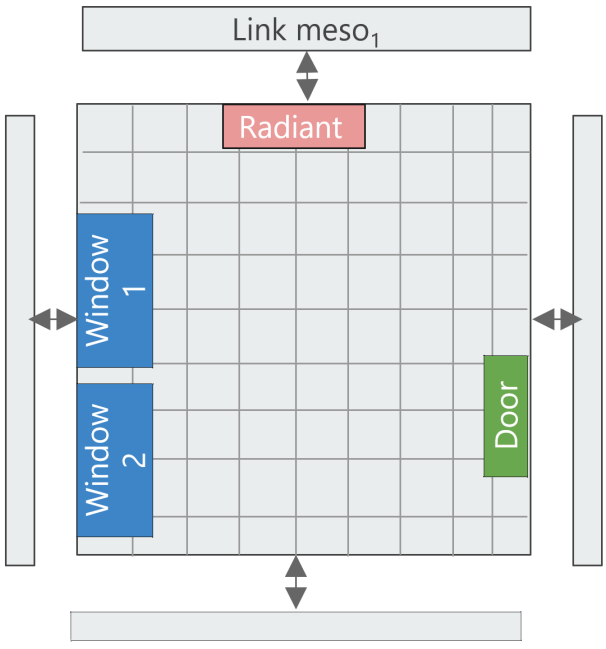

Figure 1: 2D mapping of a room example

3.2. Meso-scale

The mesoscopic scale will be a graph from which each node represents a room. The connection between the rooms are defined by wrappers (see image 2). Each wrapper contains a room. They play a role of a connector between the microscopic and the macroscopic scales. Wrappers carry the information about the heat penetration coefficient of each wall.

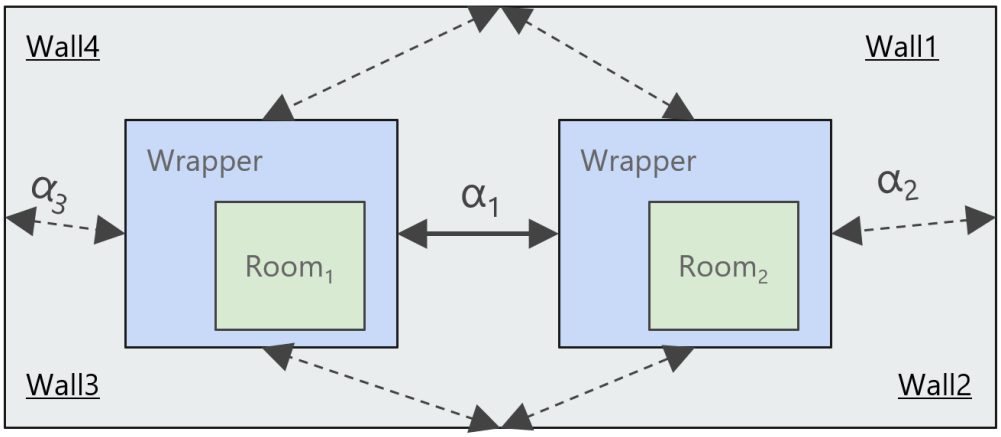

Figure 2: Wrapper scheme

3.3. Macro-scale

The macroscopic model describs the buildings with several parameters: surface, isolation level, heating type, orientation energy consumption, thermal dissipation.

4. Simulation

4.1. Inside the rooms

The rooms scale is the micro-scale and for each cells the equation [5] is resolved for each time step. Thus we can vizualize the heat diffusion created in each room such as describes in figures 3, 4 and 5. Figure 3 represents the temperature distribution after the first time step in a test room. Each image gives the 2D (x,y) temperature distribution of strata in z_i from z_0 to z_n, with n the roof and 0 the floor. Figures 4 and 5 represent the same room after $t + 500$ and $t + 5000$ time steps respectively.

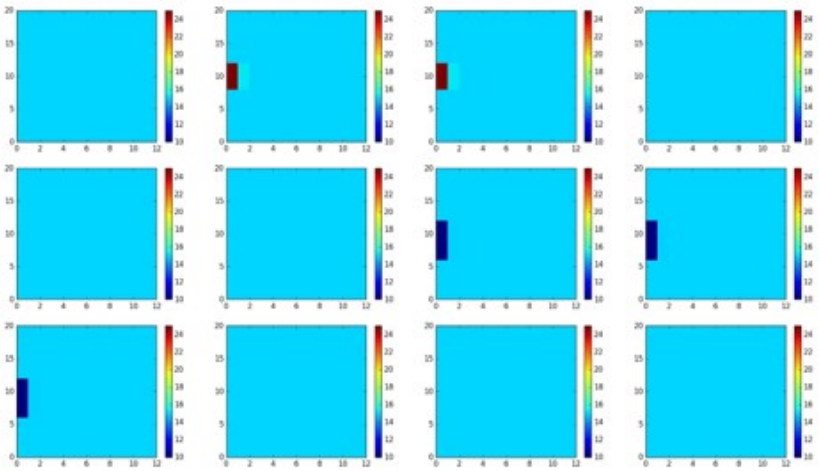

Figure 3: Simulation of the temperature evolution inside a room after one time step. Each slice represents one z_i level. z_0 (top left) and z_{11} (bottom right) are respectively the floor and the ceiling of the room.

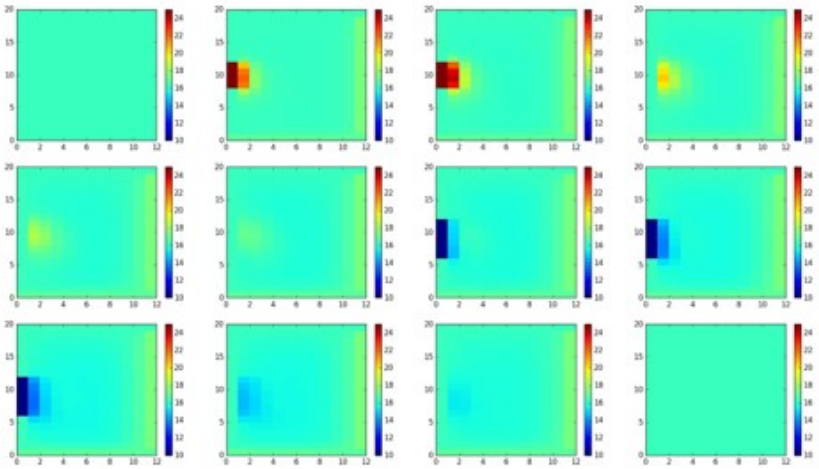

Figure 4: Simulation of the temperature evolution inside a room after 500 time step. Each slice represents one z_i level. z_0 (top left) and z_{11} (bottom right) are respectively the floor and the ceiling of the room.

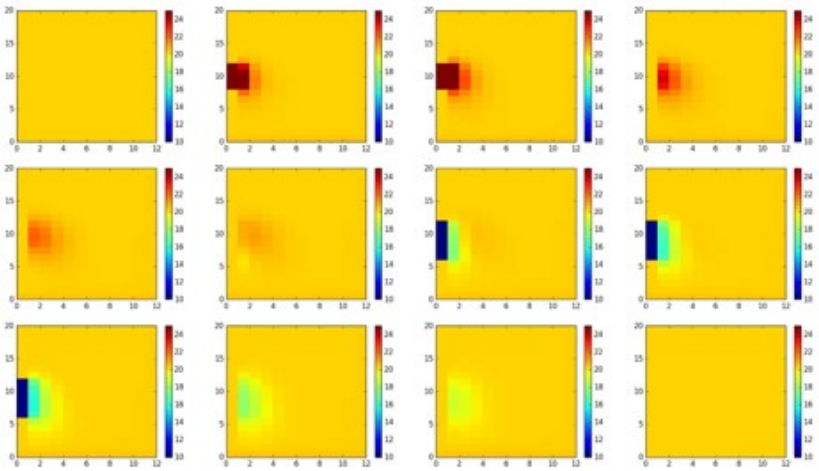

Figure 5: Simulation of the temperature evolution inside a room after 5000 time step. Each slice represents one z_i level. z_0 (top left) and z_{11} (bottom right) are respectively the floor and the ceiling of the room.

4.2. Between the rooms

Between the rooms the temperature can be transfered in two different ways: passing through the wall /closed door or as a direct exchange if the door is open. The temperature in each cell of a room is written $T_r(t, i, j, k)$, with r the room number, t the time value, i the cell number in x, j in y and k in z. The heat exchange in the case of an open door between room 1 and room 2 is computed by:

$$T_1(t; i; j; k) = T_2(t; i'; j'; k') \tag{5}$$

For i, j and k at the door position. In the case of two rooms separated by a wall the temperature exchange will depend on the penetration coefficient α as described in figure 2. We can write the temperature interaction for a cell $(i; j; k)$ in Room 1 in contact with the wall of Room 2 (cell $(i'; j'; k')$):

$$\begin{aligned} T_1(t + \Delta_t; i; j; k) &= T_1(t + \Delta_t; i; j; k) + \alpha(\frac{T_2(t; i'; j'; k') - T_1(t; i; j; k)}{2}), \\ T_2(t + \Delta_t; i'; j'; k') &= T_2(t + \Delta_t; i'; j'; k') + \alpha(\frac{T_1(t; i; j; k) - T_2(t; i'; j'; k')}{2}) \end{aligned} \tag{6}$$

The penetration coefficient α is meant to be learnt on real data for each building. Each wall being made of a specific material, the penetration coefficient will reflect its physical properties.

4.3. Outside the building

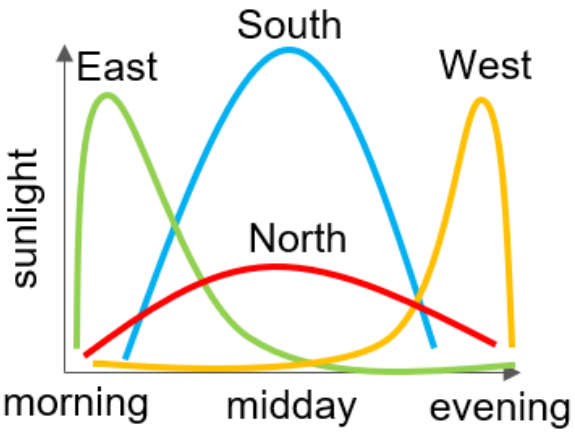

Figure 6: Light intensity on walls

In order to learn the behavior of the building with respect to the exterior temperature and sunlight, we simulate the radiation on each wall of the macro-scale. The simulation can be summarized by the figure 6. For instance the radiation power on the east wall is typically stronger in the morning. To simulate properly this effect we use a Weibull function given by:

$$f(x;k,\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{(-x/\lambda)^k} \tag{7}$$

The peak intensity abscissa is given by λ and is randomly chosen between 8 am and 10 am for the east wall, 12 pm and 2 pm for the south and north wall (north wall intensity being reduced by 80%) and between 4 and 6 pm for the west wall.

5. Code architecture

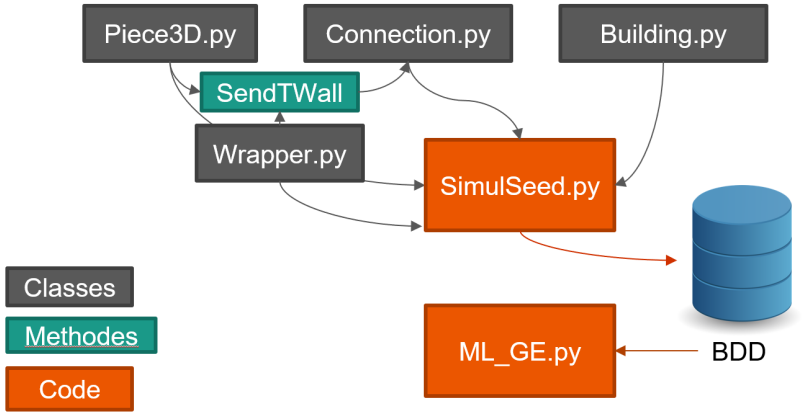

Figure 7: Code architecture

The code architecture is described in figure 7. The micro-scale class is defined in 'Piece3D.py', the meso-scale and the wrappers are defined in 'Connection.py' and 'Wrapper.py' and the macro-scale in 'Building.py'. The method 'SendTWall' is explicitly showed on the figure because it defines the entire communication between the rooms. All the classes are compiled in the main code 'SimulSeed.py'. Through API calls, this code allows the user to construct its own building architecture and simulate the heat diffusion for all the initial conditions available. The data created are stored in a mongo database along with the real data collected from the existing building (if data are collected). The database is then loaded in 'ML_GE.py' which uses Machine Learning algorithm such as Random Forest described in sections 2.2.1 to train the algorithm. After this training phase, the code can provide the most likely state of the building (Temperature, windows and heater closed or open).

6. Case study

6.1. Test with 3 simply connected rooms

For our first test, we have considered a simple case pictured in figure 8. The building consists in three rooms. Room 1 & 3 are connected to room 2 with doors and both are equipped with a radiator. We have imagined that

3 temperature sensors are placed in the middle of each room. After running a simulation of this configuration, we represent on figure 9 the temperature given by the sensors. The initial conditions are the following,
$\forall i; j; k \in \Omega_3$:

- Room 1 : $T_1(0; i; j; k) = 10°\text{C}$
- Room 2 : $T_2(0; i; j; k) = 25°\text{C}$
- Room 3 : $T_3(0; i; j; k) = 15°\text{C}$

With this configuration we have 16 (2^4) different cases (windows and heater closed or open) we need to simulate.

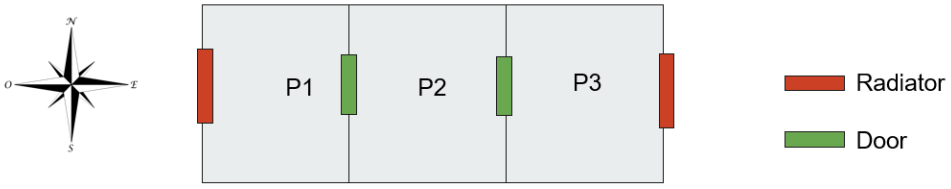

Figure 8: Mapping of test rooms

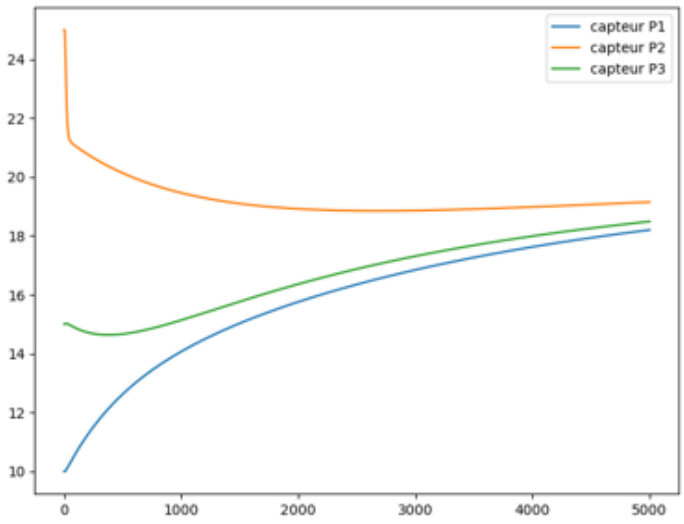

Figure 9: Temperature evolution for the initial conditions $T_1(0;i;j;k) = 10°C$, $T_2(0;i;j;k) = 25°C$ and $T_3(0;i;j;k) = 15°C$

6.1.1. Configuration classifier

We will now use a random forest classifier to learn the 16 different configurations.

- Build database :
  1. Simulate 5000 time steps for each case (note $S(t)$ the simulated temperature at time t)
  2. Add a gaussian random factor with a squared deviation of 3% of the sensor values. (It represents the sensors measurement errors): $\tilde{S}(t) = S(t) + \epsilon(t)$ where $\epsilon(t) \sim \mathcal{N}(0, [0.03 * S(t)]^2)$
  3. Let $X(t) = \left(\tilde{S}_{s1}(t), \tilde{S}_{s2}(t), \tilde{S}_{s3}(t), \Delta S_{s1}(t), \Delta S_{s1}(t), \Delta S_{s1}(t)\right)$ a vector of prediction variables. With $\tilde{S}_{s1}(t)$ (resp. $\tilde{S}_{s2}$, $\tilde{S}_{s3}(t)$) the observed temperatures of sensor 1 (resp. 2,3) at time t, and $\Delta S_{s1}(t) = S_{s1}(t) - S_{s1}(t - \Delta t)$
  4. Let $Y(t) \in \mathcal{Y} = \{\texttt{conf}_1; \texttt{conf}_2; \cdots; \texttt{conf}_{16}\}$ the label to predict.

5. Our global sample is now build :

$$\mathcal{D}_{\{16;5000\}} = \{(X_1(1), Y_1(1)), \cdots, (X_1(5000), Y_1(5000)), \cdots \\ (X_{16}(1), Y_{16}(1)), \cdots, (X_{16}(5000), Y_{16}(5000))\}$$

6. Randomly split the dataset in a train sample set (80%) and a test sample set (20%).

- Learning and testing :

1. Construct a collection of N randomized classification trees (c_k) with the train sample set :

$$\{c_k(\mathtt{x}, \Theta_k), 1 \leq k \leq N\}$$

$\mathtt{x} \in \mathbb{R}^6$ is an input vector.
$\{\Theta_k\}$ are independent and identically distributed random vectors independent of, but distributed as $\mathcal{D}_{\{16;5000\}}$
The classifier c_k is a mapping function : $\mathbb{R}^6 \to \mathcal{Y}$

2. The random forest classifier RFc is obtained via a majority vote among the classification trees, that is :

$$RFc(\mathtt{x}, \Theta_1, \cdots, \Theta_N, \mathcal{D}_{\{16;5000\}}) = \\ \arg\max_{j \in \mathcal{Y}} (\mathcal{C}\mathtt{ard}\{c_k(\mathtt{x}, \Theta_k) = j, 1 \leq k \leq N\})$$

3. We can now apply our random forest predictor on our test sample, and compare the predicted $\hat{Y}$ labels against the real one Y (results Figure 10)

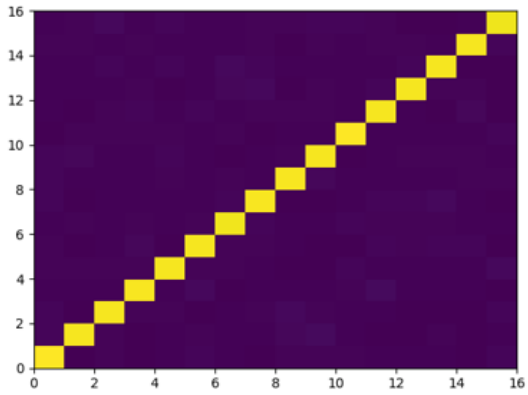

Figure 10: Confusion matrix for the 16 cases. Horizontal axis represents Y and the vertical axis $\hat{Y}$. The diagonal in yellow shows a very low misclassification rate.

Most of the failure cases are due to low time step while the building heat diffusion process is still unclear. After these results we went for a real test using See-d's office architecture.

6.2. Test on See-d's office

The case of See-d's office is more complicated due to the numerous doors, windows and radiators. On the figure 11 See-d's office structure of six rooms is represented. Doors are colored in green, windows in purple and radiators in red. The number of possible combinations is huge in this case (2^{16}) and no Machine Learning algorithm would be able to learn each of them. The number of sensors has to be increased. So, each room is equipped in its center with temperature sensors and in addition each door is equipped with status (closed/open) sensors.

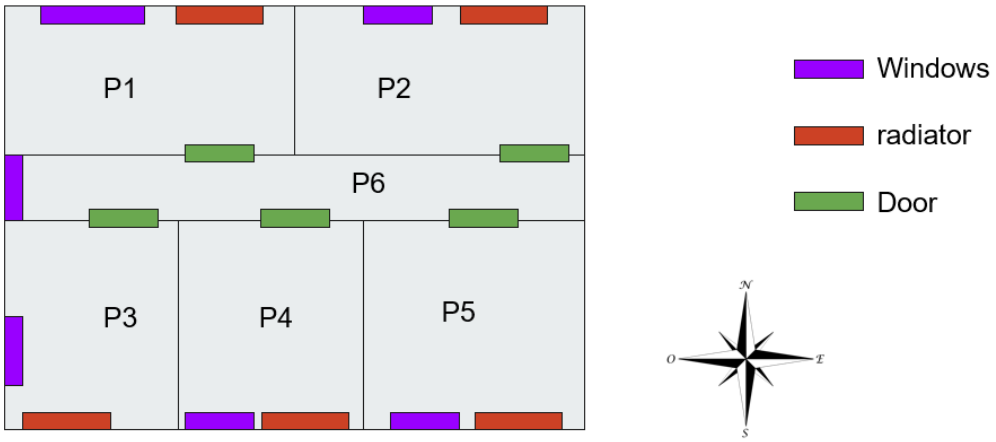

Figure 11: Mapping of See-d's desk

Using the exact same methodology as in section 6.1 we reach a success rate of 85%.

7. Summary

We prooved in this POC that the multi-scale model we developed enables to represent a building in a very robust way.The Machine Learning algorithm is improved by the imformation on the derivative of the sensors temperature. The full model is now callable by API and each user can create a specific building. The results are very encouraging with only a few sensors and this can be improved. In the future this tool could be embedded in a full scale smart building computer system.

The next steps in this project are:

- Gathering real data in a first test environment
- Find a building partner equipped with a lot of sensors
- Develop a user interface

The commercial prospections for the future of such a tool are:

- Computer system of smart building
- Architecture software for energy optimization

- Construction and public work software

Reference

[1] Geankoplis, Christie John (2003). Transport Processes and Separation Principles (4th ed.). Prentice Hall. ISBN 0-13-101367-X.

[2] http://scikit-learn.org

[3] Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A. (1984) Classification and Regression Trees. ISBN 9780412048418

[4] Breiman, Leo (2001). Random forests. Machine learning (Vol 45, p 5-32). Springer Netherlands

[5] Smith, G. D. (1985). Numerical Solution of Partial Differential Equations: Finite Difference Methods (3rd ed.). Oxford University Press.

[6] Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University.

[7] David E. Rumelhart, James L. McClelland and PDP Research Group (1987). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. ISBN 9780262680530.