PRINCIPLES OF MODELING FOR CYBER PHYSICAL SYSTEMS

Assignment #3
Parameter Estimation

Due Date: 10-20-2020

Instructor: Madhur Behl
madhur.behl@virginia.edu

Part 1 - Energy Plus and Model Parameters - 140 points
Part 2 - Parameter Estimation - 130 points

1 PART 1 - ENERGY PLUS MODEL PARAMETERS

In this worksheet, you will encode the zone model structure (from Assignment 2) in `MATLAB`.

Problem 3-1: Assigning nominal values to model parameters using building construction details from the IDF file, [50 pts]

Problem 3-2: Encoding the model structure as a `MATLAB` function, [50 pts] and

Problem 3-3: Generating building operation data. [40 pts]

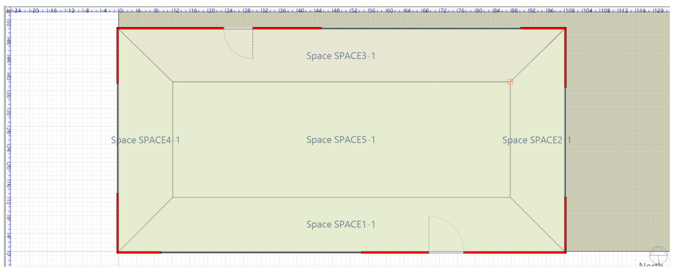


Figure 1.1: `SPACE3-1` is the north facing exterior zone of the building.

The following files are available on the course website and on UVA Collab for this assignment. `A03.zip`

1. `5ZoneAirCooled.idf`
2. `model_structure.m`
3. `construction.m`

2 **Problem 3-1:** Nominal parameter values

As discussed in the lectures, the parameter estimation problem for our state space model is non-linear in the parameters. Therefore, it is essential to compute nominal values of the parameter set in order to start the optimal parameter search during the estimation process. Much like the model structure design, the computation of the nominal values depends on the construction of the zone. The model structure contains two different types of lumped parameters describing the zone geometry.

- **Thermal resistance R** (in K/W): Thermal resistance is defined as the ratio of the temperature difference to the heat flow.
- **Conduction:** Under the one-dimensional steady-state heat flux conditions, the thermal resistance is given by:

  $$R_{conduction} = \frac{L}{kA}$$

  where L is the thickness of the conduction plane (in m), A is the surface area (in m^2), and k is the thermal conductivity (in W/mK). The reciprocal of thermal resistance $1/R$ is referred to as thermal conductance (often denoted by U).
- **Convection:** Convective heat transfer is the transfer of energy between a surface and a fluid (such as air) in contact with the surface. Convection can be represented using thermal resistance similar to conduction. The thermal resistance for convection is given by:

  $$R_{convection} = \frac{1}{h_c A}$$

  where, h_c (in W/m^{2}K) is the convection heat transfer coefficient of the fluid medium.
- **Thermal capacitance C** (in J/K): Thermal capacitance is the measure of temperature change in a material based on its volume. It is given by:

$$C = \rho L A c_p$$

where ρ is the density of the material (in k/m^3), c_p is the specific heat capacity (in J/kK, and LxA is the volume (in m^3).

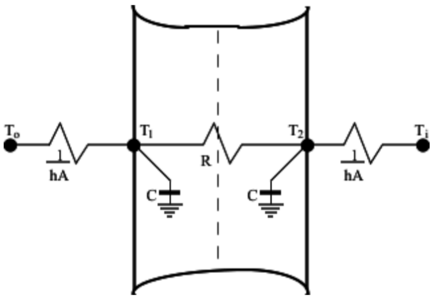
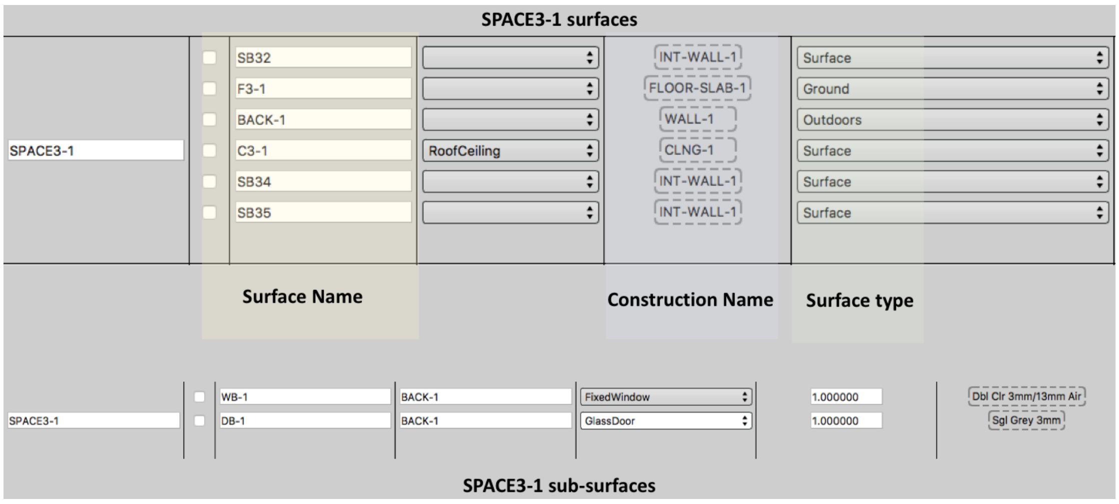


Figure 2.1: 3R2C lumped parameter configuration for a wall.

- Remember that in the popular **3R2C** parameter configuration as shown in Figue 2.1, each of the thermal capacitance coefficients C are considered equivalent. In which case:

$$C = \frac{\rho L A c_p}{2}$$

Each of these parameters, further depends on the characteristics of the material of construction, or the medium of heat transfer. For each surface, based on the material of construction, we will compute the nominal R and C values for that associated lumped parameter.
In EnergyPlus the building is described by its Input Data File or IDF file. This contains all the geometry, construction, operation, and equipment details about the building. Each zone is described by different surfaces, which are further described by different constructions, and each construction comprises of layers of one or often more materials.
So the zone layout is as follows: SURFACES » CONSTRUCTION » LAYERS » MATERIALS Figure 2.2 shows all the IDF variable names of the different surfaces and constructions for each surface. We can see that the zone `SPACE3-1` has a total of 6 different surfaces and 2 sub-surfaces. There are three internal walls, a floor, a ceiling, and an outdoor/external wall. The external wall surface (name `BACK-1`) has two sub-surfaces: one for the window (`WB-1`) and one for the door (`DB-1`).
Figure 2.3 shows two examples of how the surfaces are described in the IDF file. If you search the keyword 'BACK-1' in the IDF you will find that it the name of a 'BuildingSurface' , the surface type is a 'WALL', and the surface comprises of the construction WALL-1. The exact dimensions of the surface and its area can be computed using the 4 *(x,y,z)* vertices of the surface.



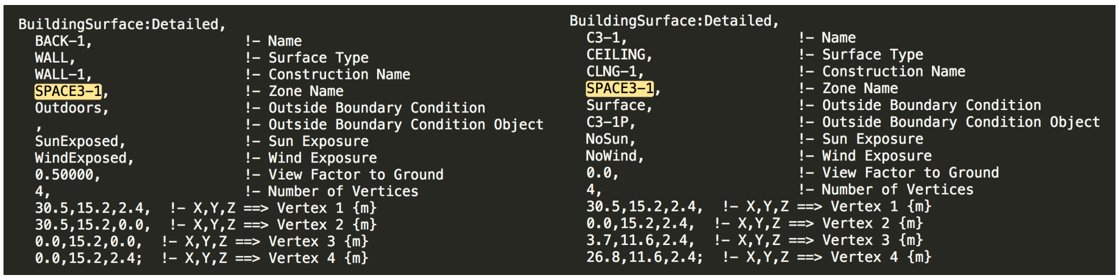
SPACE3-1 surfaces

Zone	Surface Name		Construction Name	Surface type
SPACE3-1	SB32		INT-WALL-1	Surface
	F3-1		FLOOR-SLAB-1	Ground
	BACK-1		WALL-1	Outdoors
	C3-1	RoofCeiling	CLNG-1	Surface
	SB34		INT-WALL-1	Surface
	SB35		INT-WALL-1	Surface

SPACE3-1 sub-surfaces

SPACE3-1	WB-1	BACK-1	FixedWindow	1.000000	Dbl Clr 3mm/13mm Air
	DB-1	BACK-1	GlassDoor	1.000000	Sgl Grey 3mm

Figure 2.2: All surfaces and constructions of zone SPACE3-1



```
BuildingSurface:Detailed,
  BACK-1,                    !- Name
  WALL,                      !- Surface Type
  WALL-1,                    !- Construction Name
  SPACE3-1,                  !- Zone Name
  Outdoors,                  !- Outside Boundary Condition
  ,                          !- Outside Boundary Condition Object
  SunExposed,                !- Sun Exposure
  WindExposed,               !- Wind Exposure
  0.50000,                   !- View Factor to Ground
  4,                         !- Number of Vertices
  30.5,15.2,2.4,  !- X,Y,Z ==> Vertex 1 {m}
  30.5,15.2,0.0,  !- X,Y,Z ==> Vertex 2 {m}
  0.0,15.2,0.0,  !- X,Y,Z ==> Vertex 3 {m}
  0.0,15.2,2.4;  !- X,Y,Z ==> Vertex 4 {m}
```

(a) BACK-1 surface (outdoor wall)

```
BuildingSurface:Detailed,
  C3-1,                      !- Name
  CEILING,                   !- Surface Type
  CLNG-1,                    !- Construction Name
  SPACE3-1,                  !- Zone Name
  Surface,                   !- Outside Boundary Condition
  C3-1P,                     !- Outside Boundary Condition Object
  NoSun,                     !- Sun Exposure
  NoWind,                    !- Wind Exposure
  0.0,                       !- View Factor to Ground
  4,                         !- Number of Vertices
  30.5,15.2,2.4,  !- X,Y,Z ==> Vertex 1 {m}
  0.0,15.2,2.4,  !- X,Y,Z ==> Vertex 2 {m}
  3.7,11.6,2.4,  !- X,Y,Z ==> Vertex 3 {m}
  26.8,11.6,2.4;  !- X,Y,Z ==> Vertex 4 {m}
```

(b) C3-1 surface (ceiling)

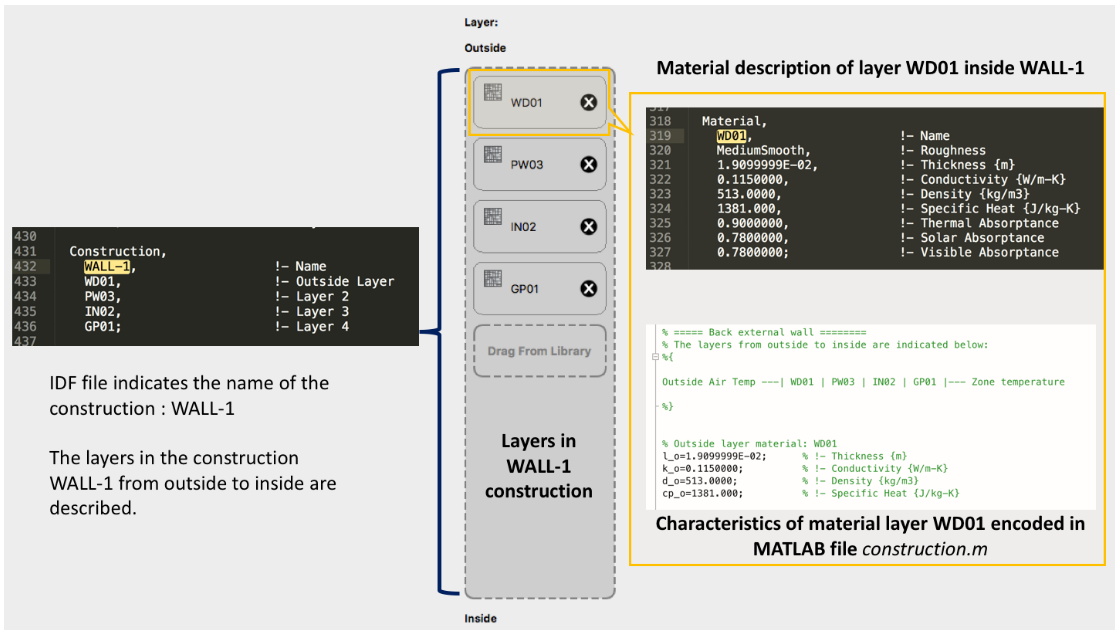
Figure 2.3: IDF surface descriptions examples

Having understood how surfaces are described, let us now look at constructions. Figure 2.4, shows the breakdown of the construction WALL-1. Remember that WALL-1 corresponds to the surface BACK-1 (which is the outdoor wall). We can see from the figure that the construction WALL-1 comprises of 4 layers of materials stacked together. The outermost material layer is called WD01 (exposed to the outside air), this is followed by 2 internal layers (PW03 and IN02) and lastly, the innermost layer (facing the inside of the zone) GP01. The IDF file describes the thermal properties of each layer in detail. We are interested in these thermal properties as we will use them to calculate the lumped thermal resistance and capacity of each surface.

Figure 2.4 also shows how the thermal characteristics of a material layer can be specified in MATLAB.

Figure 2.5 shows the layers for most of the constructions in the zone. However, it is highly advised that you navigate the IDF file by searching for the surface name, construction name, and then the layer name.

Remember the building elements are related in the following hierarchy SURFACES » CON-


Layer:
Outside
WD01
PW03
IN02
GP01
Drag From Library
Layers in WALL-1 construction
Inside
Material description of layer WD01 inside WALL-1
Characteristics of material layer WD01 encoded in MATLAB file *construction.m*

```
Construction,
  WALL-1,                    !- Name
  WD01,                      !- Outside Layer
  PW03,                      !- Layer 2
  IN02,                      !- Layer 3
  GP01;                      !- Layer 4
```

IDF file indicates the name of the construction : WALL-1

The layers in the construction WALL-1 from outside to inside are described.

```
Material,
  WD01,                      !- Name
  MediumSmooth,              !- Roughness
  1.9099999E-02,             !- Thickness {m}
  0.1150000,                 !- Conductivity {W/m-K}
  513.0000,                  !- Density {kg/m3}
  1381.000,                  !- Specific Heat {J/kg-K}
  0.9000000,                 !- Thermal Absorptance
  0.7800000,                 !- Solar Absorptance
  0.7800000;                 !- Visible Absorptance
```

```
% ===== Back external wall ========
% The layers from outside to inside are indicated below:
%{

Outside Air Temp ---| WD01 | PW03 | IN02 | GP01 |--- Zone temperature

%}


% Outside layer material: WD01
l_o=1.9099999E-02;      % !- Thickness {m}
k_o=0.1150000;          % !- Conductivity {W/m-K}
d_o=513.0000;           % !- Density {kg/m3}
cp_o=1381.000;          % !- Specific Heat {J/kg-K}
```

Figure 2.4: All surfaces and constructions of zone SPACE3-1

STRUCTION » LAYERS » MATERIALS
You should now be able to navigate to the thermal properties of any material, of any construction for the zone SPACE3-1 in the IDF.
You can use the MATLAB function `construction.m` template provided to compute the nominal values of the parameters based on the material of construction details, as described in the input data file (IDF).

```
function [paranom,surfaces] = construction()

% Goal:
% Get familiar with zone construction
% Get familair with IDF files
% Parameter details and calclulation of the nominal values.

%{
Function Name: construction.m
Input arguments:
    - None
Outputs:
    paranom: A struct of nominal values with two fields
        - rvalues: thermal conductance nominal values.
        - cvalues: thermal capacity nominal values.
    surfaces: A struct with two fields:
        - areas: calculated areas of different zone surfaces
        - sol_abs: coefficient of irradiance absorpiton or ...
            transmission, where applicable.
```

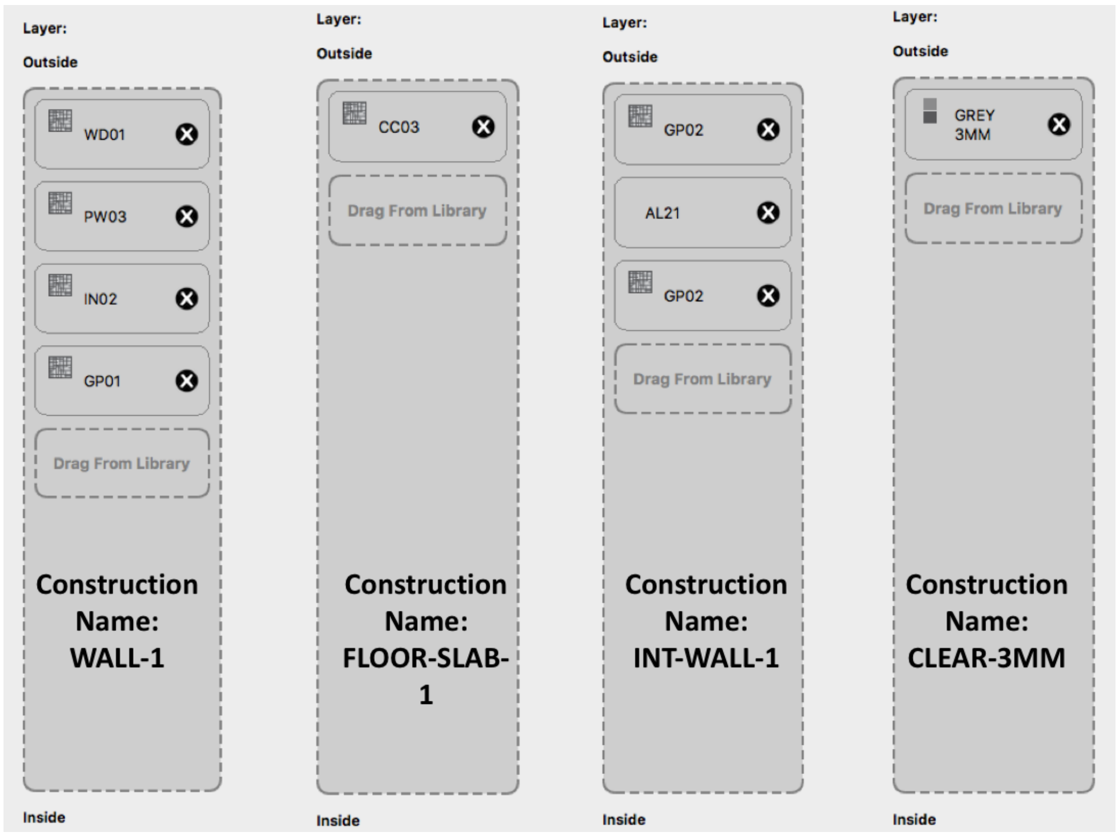


Figure 2.5: Layers element within different zone constructions

```
%}
```

Here is an example of how the thermal properties can be specified in the construction.m function:

```
% ===== Back external wall ========
% The layers from outside to inside are indicated below:
%{

Outside Air Temp ---| WD01 | PW03 | IN02 | GP01 |--- Zone temperature

%}


% Outside layer material: WD01
l_o=1.9099999E-02;       % !- Thickness {m}
k_o=0.1150000;           % !- Conductivity {W/m-K}
d_o=513.0000;            % !- Density {kg/m3}
cp_o=1381.000;           % !- Specific Heat {J/kg-K}
```

Majority of the parameters have been initialized to zero, but some parameter have been pre-computed for assistance. Follow the instructions in the `construction.m` template and fill in all the missing values.

Submit the following: (50 points)

- **[P3-1.a] (40)** Submit your `construction.m` solution file. Do not change the format or the fields of the output structures `paranom`, and `surfaces`, although you may edit/add the number of elements in the structures based on your model. Be sure to comment your code and edits and describe the parameters, should you make any changes.
- **[P3-1.b] (10)** Submit a CSV file `nominal.csv` where the first column contains the parameter variable name, and the second column contains the computed value of the parameter, and the thrid column contains a breif description of the parameter. For e.g. an entry in this file could be: `Ue, value, Thermal conductance external wall`

3 Problem 3-2: Model structure in MATLAB

Now that the nominal values of the parameters have been computed, the next step is to import the structure of the state-space model from the previous worksheet into `MATLAB`. This implies, creating the A,B,C, and D in `MATLAB` with the correct elements which depend on the values of the different R and C parameters of the model. This step should be straight forward since you submitted the A, B, C, and D matrices as the solutions to Worksheet #2. All we need to do is create the same matrices as the output of a function.
Feel free to use the `model_structure.m` function template provided.

```
function [A,B,C,D] = model_structure(p,nstates,ninputs)
%{
Input the A, B, C and D matrices for the state space model in terms of ...
    the parameters specified in paranom.

Function Name: model_structure.m
Function Description: Given a vector of parameters, the function creates
the state transition (A), input (B), output (C), and feedforward (D)
matrices by specifying the non-zero elements in these matrices.
Input arguments: 3
    - p: This is a vector of all the unknown parameters. For parameter ...
        identification we initialize the vector p using the nominal
    values computed in the paranom structure.
    - nstates: Number of states in the model (from Worksheet 2)
    - ninputs: Number of inputs in the model (from Worksheet 2)

Outputs:
    Matrices A,B,C, and D with the correct dimentsions:
    dim(A) = nstates x nstates
    dim(B) = nstates x ninputs
    dim(C) = nstates x nstates
    dim(D) = nstates x ninputs
```

As can be seen, the input to the function is a vector,

$$\vec{p}$$

of parameters to be estimated. Let us briefly review how the parameter estimation works:

Algorithm 1 How parameter estimation will be setup later.

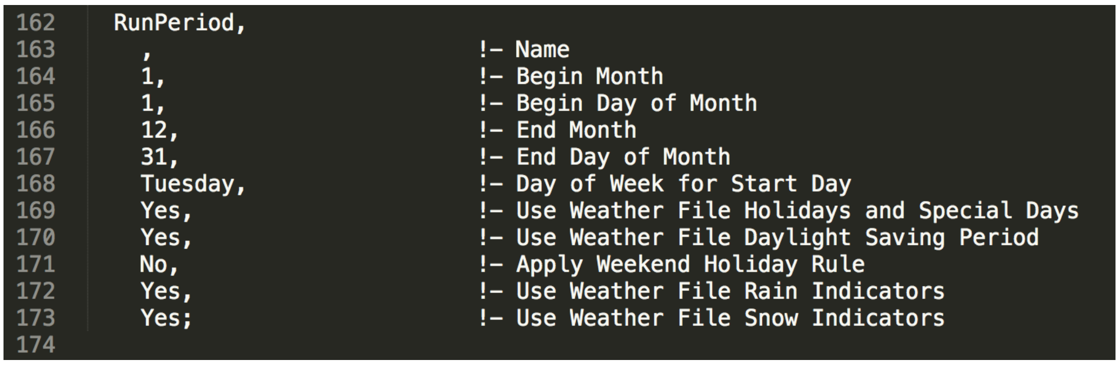
```
1: procedure PARAMETER ESTIMATION PSEUDO-CODE
2:     Compute nominal values
3:     Initialize parameter vector $\vec{p}$ with nominal values
4:     Define model structure (A,B,C,D) in terms of $\vec{p}$
5:     while Repeat until convergence do
6:         Estimate new value of $\vec{p}$ using data and non-linear least squares.
7:         Update elements of A,B,C,D with new $\vec{p}$
8:     return best estimate of $\vec{p}$
```

Therefore, we need to specify the elements of the A,B,C, and D matrices in terms of the parameter vector $\vec{p}$ Refer to the example in the `model_structure.m` template file for details on how to define the vector $\vec{p}$ of unknown parameters.

Submit the following: (50 POINTS)

- Submit your *model_structure.m* solution file.

4 PROBLEM 3-3: GENERATING BUILDING OPERATION DATA



```
RunPeriod,
  ,                      !- Name
  1,                     !- Begin Month
  1,                     !- Begin Day of Month
  12,                    !- End Month
  31,                    !- End Day of Month
  Tuesday,               !- Day of Week for Start Day
  Yes,                   !- Use Weather File Holidays and Special Days
  Yes,                   !- Use Weather File Daylight Saving Period
  No,                    !- Apply Weekend Holiday Rule
  Yes,                   !- Use Weather File Rain Indicators
  Yes;                   !- Use Weather File Snow Indicators
```

Figure 4.1: Run Period is specified in the IDF.

- Run the `5ZoneAirCooled.idf` file for the entire month of July with a **time-step of 5 mins.** Assume the first day of the month is Monday. You will need to change the RunPeriod specified in the IDF as well as the timestep.

- You will notice that the simulation output creates a directory will multiple files. Import the following columns from the *correct* csv file into `MATLAB`. Write a matlab script to read the data from the CSV file and arrange it in a matrix with **columns in the order specified below**. Save this matrix as `<Your_Name>_Building_Data.mat`

  1. Ground Temperature, T_g (°C)
  2. Outside ambient temperature, T_a (°C)
  3. Return air plenum temperature, T_p (°C)
  4. Total external solar heat gain, $Q_{sol,e}$ (W)
  5. Total internal heat gain, Q_{gain} (W)
  6. Total sensible cooling load, Q_{cool} (W)
  7. Neighboring zone temperatures for `SPACE2-1` T_2, `SPACE4-1` T_4, `SPACE5-1` T_5 (°C) [in this order]
  8. SPACE3-1 zone temperature, T_z (°C)

Please adhere to the order of the columns specified above as we will grade the problems below based on that.

Submit the following: (20 POINTS)

- For each of the above inputs/states (1-8) report the minimum value, maximum value, and average value of the input for the period of the simulation.
- Submit the entries of row 141 from your mat file.

Submit the following: (20 POINTS) You are now the building facilities manager for this building. On July 17, you know there will be a demand response event from 2:00-4:00pm in the afternoon. You know you have notified the occupants of SPACE3-1 about this event and expect that during the event the occupancy of the SPACE3-1 will remain constant at 0.4 fraction of the maximum occupancy.
You now want to use EnergyPlus to simulate the effect of the changed occupancy schedule. Modify your IDF file such that:

- On July 17, between 2-3pm the occupancy of SPACE3-1 will be at 0.4 fraction of the maximum occupancy.
- Occupancy outside of the demand response event can remain as usual.
- Note that we only want the occupancy of SPACE3-1 to change and not affect any other zones.

Submit all the edits you made to the IDF file for this. You may need to define custom schedules and make other changes to link that schedule to SPACE3-1.

PART 2- ESTIMATING THE MODEL PARAMETERS

By the end of this worksheet, you will have a state-space model of a single zone in EnergyPlus. From the previous section, you should have the following:

- `construction.m` - Computes the nominal values of the model parameters from the material characteristics in IDF.
- `model_structure.m` - The function which describes how the elements of the A,B,C,and D matrices depend non-linearly on the parameter vector 'p' in MATLAB.
- A csv file generated from EnergyPlus with the model input and output data.

So far, you have successfully managed to create, and encode the state-space model structure in `MATLAB`. However, this is just half of the effort. The other half is to tune the model parameters (R,C values) to match the operation data from the zone. Remember that our goal with model based design in this case is to produce a dynamical state-system model, which can predict the zone's temperature evolution, given a forecast of disturbances and inputs.
In this worksheet, we will:

Problem 4-1: Use non-linear parameter estimation algorithm to estimate the R and C parameter values using building's data.[60pt],

Problem 4-2: Write a function which calculates the predicted temperature of the zone from input data, for a given value of the parameter vector[40pt], and

Problem 4-3: Evaluate the prediction accuracy of the model.[30pt]

The following files are available on the course website on UVA Collab.

1. `5ZoneAirCooled_altmod.csv` - EnergyPlus csv data dump.
2. `data.mat` - Data file for model training and testing.
3. `parameter_estimation.m` - Main program for parameter estimation and model evaluation.
4. `getlabel.m` - Generate response predictions for a given set of parameter values.

5 **PROBLEM 4-1:** PARAMETER ESTIMATION

As discussed in the lectures, the parameter estimation problem for our state space model is non-linear with respect to the parameters of the model (i.e. the elements of the A,B,C, and D matrices depend non linearly on the RC parameters). Therefore, we need to rely on non-linear least squares estimation algorithms to search for the optimal values of the parameters. Least squares for parameter estimation, is the problem of finding a parameter vector θ that is a local minimizer to a function that is a sum of squared errors, possibly subject to some constraints:

$$
\begin{aligned}
&\text{minimize}_\theta \sum_{i=1}^{N} (y_{true} - \hat{y}_{predict})^2 \\
&\text{subject to:} \\
&\qquad x(i+1) = A(\theta)x(i) + B(\theta)u(i) \\
&\qquad y(i) = C(\theta)c(i) + D(\theta)u(i) \\
&\qquad x_0 = x_{init} \\
&\qquad lb <= \theta <= ub
\end{aligned}
$$

The notation in the optimization problem above explicitly denotes the dependence of $A(\theta)$, $B(\theta)$, $C(\theta)$, and $D(\theta)$ matrices on the parameter vector θ. In fact, you encoded this dependence in `MATLAB` in the previous exercise in the function `model_structure.m`

5.1 Levenberg-Marquardt and Trust-Region-Reflective algorithms

Nonlinear least squares methods iteratively reduce the sum of the squares of the errors between the predicted function output and the measured data points through a sequence of updates to parameter values. The Levenberg-Marquardt curve-fitting method is a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic (and utilizing the Jacobian approximation to the Hessian) The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value. It belongs to a class of trust region algorithms for non-linear optimization.
In `MATLAB`, several optimization solvers are available for solving the non-linear least squares problem:

- `lsqnonlin`
- `nlinfit`
- `lsqcurvefit`

They either use the Levenberg-Marquardt or a Trust-Region-Reflective estimation algorithm.

5.2 MATLAB implementation

A `parameter_estimation.m` `MATLAB` program template has been provided. This `MATLAB` program, performs the following tasks:

- Obtains the nominal parameters from `construction.m` - From Worksheet 3.

```
%% Construction Details and calclulation of the nominal values.

%Obtain nominal parameters structure and some zone construction ...
    information
[paranom,surfaces]=construction(); % NO input arguments should be ...
    provided
```

- Loads the input-output data `data.mat` from the working directory. - This mat file is provided on the course website.

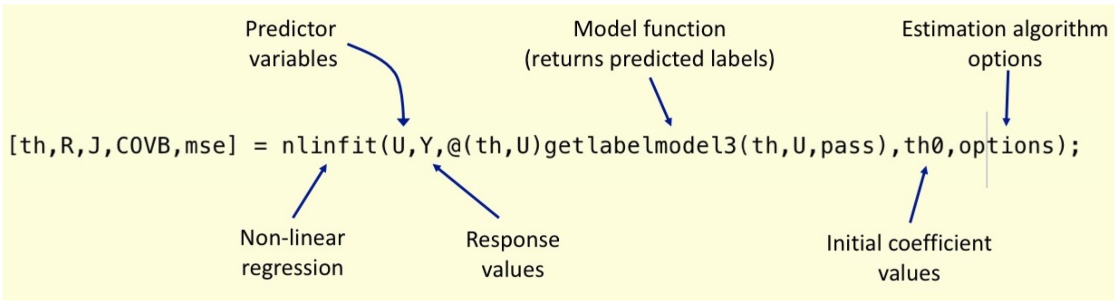
```
%% Load the training dataset.
%{
(1) Read and store training data
(2) partition data into two modes: cooling ON and cooling OFF: ...
    The indices
derived above will be used here.

More about the training data:

Column# | Input data stream of interest.
3    |   Ground temperature (C)
2    |   Outside ambient temp (c)
4    |   Total internal radiative gain (W)
6    |   Total internal convective gain (W)
14   |   solar radiation on external wall (W)
10   |   solar radiation transmitted through window +door (W)
9    |   other equipment (W)
23   |   SPACE2-1 temp (C)
25   |   SPACE3-1 temp (C)
26   |   SPACE4-1 temp (C)
27   |   SPACE5-1 temp (C)
22   |   plenum temp (C)
%}

%alldata = readtrainingdata('5ZoneAirCooled_altmod.csv');
load('alldata.mat');
alldata(1,:)=[];
alldata(:,12) = (alldata(:,12)+alldata(:,20)+alldata(:,21))/3; % ...
    average indoor temperatures of all neighboring zones


% Load date/time schedule info:
% This is usedful for plotting as well as separating weekend and ...
    weekday
% schedules.
load('training_dates.mat');
dateinfo=datenum(training_dates(1:end));
```

- Creates the training and testing data-sets from the input-output data.

```
% split into training and testing datasets.
% specify the fraction of data to be used for model training ...
    (parameter
% estimation)
ntrain = floor(0.7*ndata);
ntest = ndata - ntrain;

% form the training data structure
U=zeros(ntrain,7);

U(:,1)=alldata(1:ntrain,2);      % ta: outside ambient temperature.
U(:,2)=alldata(1:ntrain,3);      % tg: ground slab temp.
U(:,3)=alldata(1:ntrain,22);     % tp: plenum temperature.
U(:,4)=alldata(1:ntrain,12);     % <tiw>: Average indoor ...
    interaction temp.
U(:,5)=alldata(1:ntrain,15);     % qsole: External solar irradiation.
U(:,6)=alldata(1:ntrain,8);      % qgain: Internal heat gains.
U(:,7)=alldata(1:ntrain,9);      % qcool: Cooling rate.

Y=alldata(1:ntrain,25);          % SPACE3-1 temp (output)
```

- Specifies the option-set for a multi-start non-linear least squares regression problem - *You will complete this step.*
- Returns the estimated set of parameters, and a vector of predicted values of the output (zone temperature). - *You will complete this step.*
- Evaluates the model: *You will complete this step.*

Figure 5.1 shows an example of how the estimation function `nlfit` in `MATLAB` can be used for the parameter estimation. You need not use the exact same function.



```
[th,R,J,COVB,mse] = nlinfit(U,Y,@(th,U)getlabelmodel3(th,U,pass),th0,options);
```

Predictor variables
Model function (returns predicted labels)
Estimation algorithm options
Non-linear regression
Response values
Initial coefficient values

Figure 5.1: Example of a non-linear estimation function in matlab.

For non-linear parameter estimation algorithms, you need to provide a model function to generate a vector of predicted responses in order to evaluate the least squares objective function. You will learn about writing this function next.

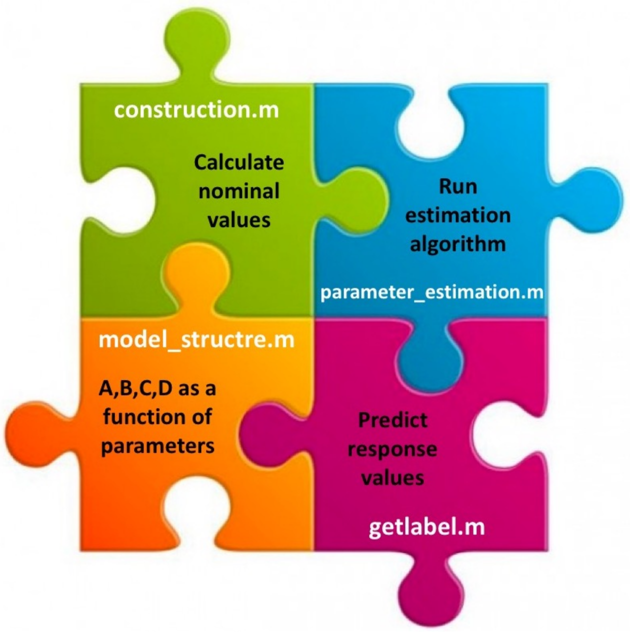


Figure 6.1: All the elements of a parameter estimation problem.

6 **Problem 4-2:** Obtaining predicting response values

As described earlier, the non-linear parameter estimation is essentially a search over the parameter space. For the estimation algorithm to converge to a minima we need to compute the least squares cost function:

$$\sum_{i=1}^{N} (y_{true} - \hat{y}_{predict})^2$$

which requires constructing the the prediction response vector $\hat{y}_{predict}$ (of length equal to N) for a given point in the parameter space.
You can use the `getlabel.m` template function for this part of the worksheet.
Given a parameter vector, the `getlabel` function will return a vector of predicted responses (y) of length equal to the number of samples in the training data.
Fill in the missing code to compute the vector of predicted labels from the model, with A, B, C, and D matrices computed using a given parameter vector.

7 **Problem 4-3:** Model evaluation

We now have all the missing pieces of the parameter estimation puzzle (Figure 6.1):

- The `construciton.m` function computes the nominal values of the parameters.

- The `model_structure.m` function specifies the elements of the state-space models A, B, C, and D in terms of the parameter vector.
- The `getlabel.m` function returns the vector of predicted responses, for a given estimate of the parameter values.
- The script `parameter_estimation.m` loads the input data, constructs the training and the test data, estimates the parameters, and evaluates the model accuracy.

Based on which estimation algorithm you chose to implement and your model structure, you can obtain an estimate of the parameter vector (with R and C values) which has the least squared error on the training data. It is recommended that you play around with different options and thresholds for the estimation algorithm and see their effect on the converged parameter values.
However, the real test for any model is how well the model generalizes on a data-set that it has not 'seen' before i.e. how well can it predict the model output on an unseen input data-set.
There are several ways to validate the output of the model. A very common way is to obtain the predictions from the model, on an input data-set, and then compare these predictions with the ground truth, of what the system actually did. The question, then becomes, how do you compare the predicted response with the ground truth values of the response variable. While the first thing you should do is to plot both the data-sets against each other, you can use several functions which compute some measure of the *goodness of fit*. A common and effective way of evaluating how well the predictions match with reality is by calculating the root mean square error (RMSE) of the data-sets. The RMSE measures the standard deviation of the error between the predicted values and the ground truth (or simply put, its the standard deviation of the residuals of the model predictions).

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N}(y_{true} - \hat{y}_{predict})^2}{N}}$$

It is beneficial to also calculate the normalized root mean squared error (NRMSE) which is a dimensionless value that helps compare the RMSE with the average of the ground truth or the range of the ground truth data. It is especially useful when the scale of the response variable is too large or too small.

$$\text{NRMSE} = \frac{RMSE}{y_{true_{max}} - y_{true_{min}}} = \frac{RMSE}{\bar{y_{true}}}$$

Submit the following:(130 points)

- **[P4.1](60)** Submit you entire code base with all the functions `construction.m`, `model_structure.m`, `getlabel.m`, and `parameter_estimation.m`. Create a `<your computing ID.zip>` file with all the functions above and any other functions or mat files that your code may use. Ensure that:

  - Your submitted code directory runs without any errors (rank or inversion warnings due to parameter estimations are fine, but no hard errors) and displays the NRMSE values for the training and testing data-sets as defined in the `parameter_estimation_exercise.m` template,
  - your code creates at least two plots, one for the training data model accuracy and one for the test data model accuracy. You may plot other relevant variables, parameters, or states, as needed.

- **[P4.2] (40)** Try the parameter estimation with and without multi-start, and report your observations (NRMSE values and test data plots) on the performance difference between the two cases.

- **[P4.3] (30)** Instead of the input-output data provided in the `data.mat` file; use the input-output data you created in the previous section. Evaluate how the model accuracy changes between the two data-sets, given that everything else (model_structure, estimation algorithm) still remains the same.