Cubesat Thermal Modelling Methodology

Khushaldas Badhan, ISM 2021-22, University of Luxembourg
Cubesat Lab 2

Methodology:

For the thermal analysis the model for the satellite is divided into nodes. Each node can interact with the other. By interaction it means transmission of heat by conduction and radiation.

There are two types of nodes: 1) Heat Storage nodes and 2) Interface nodes

1) Heat Storage Node (HSN): These nodes represent actual discreet bulk masses in the satellite and store heat.
2) Interface Node (IFN): These nodes represent interfaces for the HSNs. HSNs have interaction with other HSNs through IFNs. Heat is exchanged by IFNs and the summation of the total heat exchanged by all the IFNs associated to an HSN gives total heat gain/loss in an HSN.

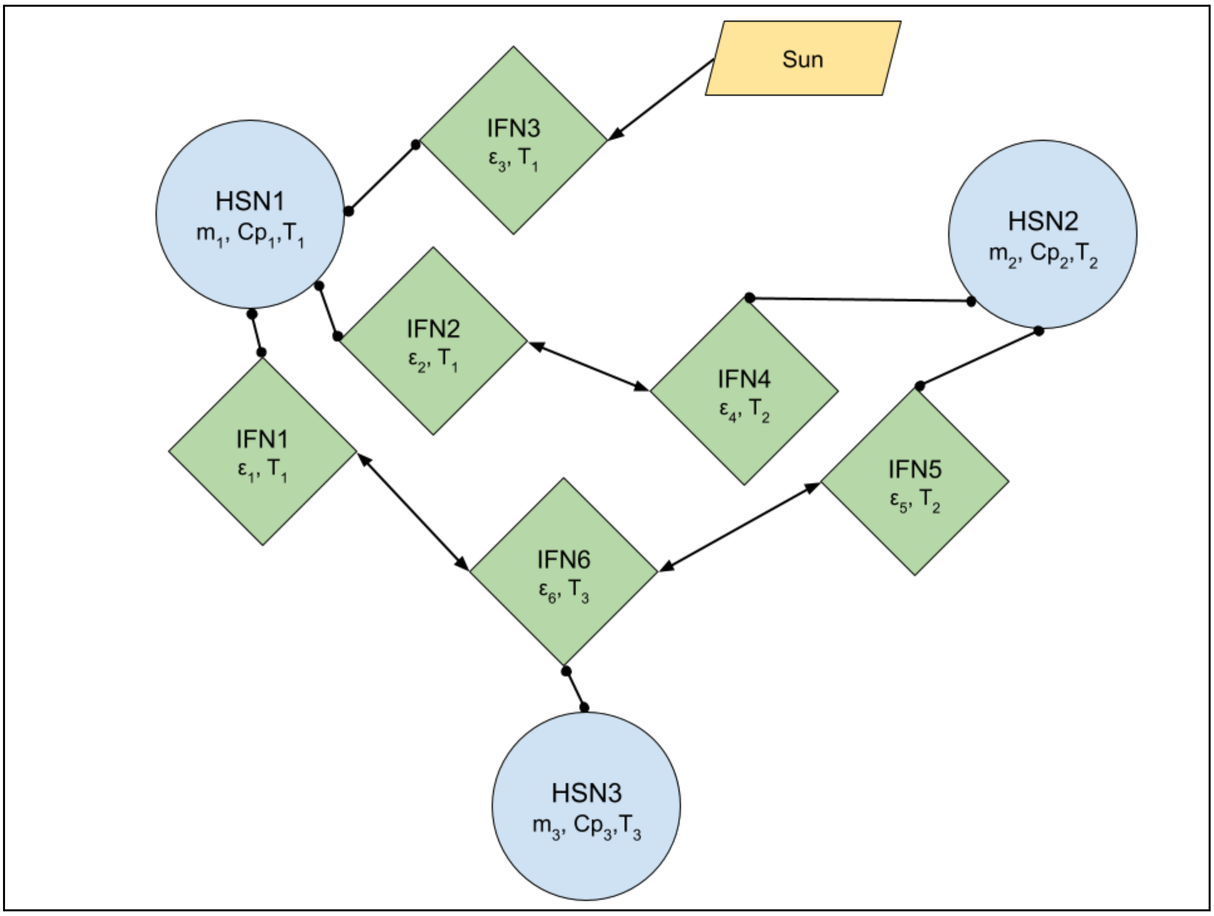

Fig HSN and IFN interaction

For example, an HSN can be a wall panel of the satellite while IFNs for it are the inside face of the wall which exchanges heat with components inside the satellite and outside face of the wall which exchanges heat with elements outside the satellite for example, sun, Earth, deep space, etc. For the wall, total heat will be the summation of heat exchanged by both IFNs. Below is a list of the HSNs and IFNs in the LICEOR satellite thermal model.

IFN Index	**Name of the node**	**HSN Index**
1	ISIS Antenna rod Zenith	1
2	ISIS Antenna rod Nadir	2
3	ISIS Antenna rod Nadir right	3
4	ISIS Antenna rod Nadir left	4
5	ISIS Antenna P	5
6	ISIS Antenna N	6
7	counter weight	7
8	IOBC X+	8
9	IOBC X-	8
10	SGR	9
11	ADCS X1	10
12	ADCS X2 (magtorquer X axis)	11
13	ADCS X3	12
14	ADCS X4	13
15	ADCS X5 (magtorquer Y and Z axis)	14
16	EPS X1 (face near X+)	15
17	ISIS	16
18	Prop X1	17
19	Prop X2 (side single surface as single node)	17
20	Prop X3 (nozzle)	17
21	Frame Vertical Nadir Left X1	18
22	Frame Vertical Nadir Right X1	19
23	Frame Vertical Nadir Left X2	20
24	Frame Vertical Nadir Right X2	21
25	Frame Vertical Zenith Right X1	22
26	Frame Vertical Zenith Left X1	23
27	Frame Vertical Zenith Right X2	24
28	Frame Vertical Zenith Left X2	25
29	Wall Nadir N	26
30	Wall Nadir Left N	27
31	Wall Nadir Right N	28

32	Wall Zenith N	29
33	Wall Nadir P	26
34	Wall Nadir Left P	27
35	Wall Nadir Right P	28
36	Wall Zenith P	29
37	SP Nadir X1	30
38	SP Nadir X2	31
39	SP Zenith X1	32
40	SP Zenith X2	33
41	SP Nadir Right X1	34
42	SP Nadir Right X2	35
43	SP Nadir Left X1	36
44	SP Nadir Left X2	37
45	GPS Rx Nadir	38
46	GPS Rx Zenith	39
47	EPS X2 (face away from X+)	15
48	EPS Nadir face	15
49	EPS Nadir right face	15
50	EPS Nadir left face	15
51	EPS Zenith face	15
52	EPS X3 (face on the X- side)	15
53	Prop X4 (flat face on the side of nozzle (not nozzle exit))	17

Below figures show HSNs in the cubesat:

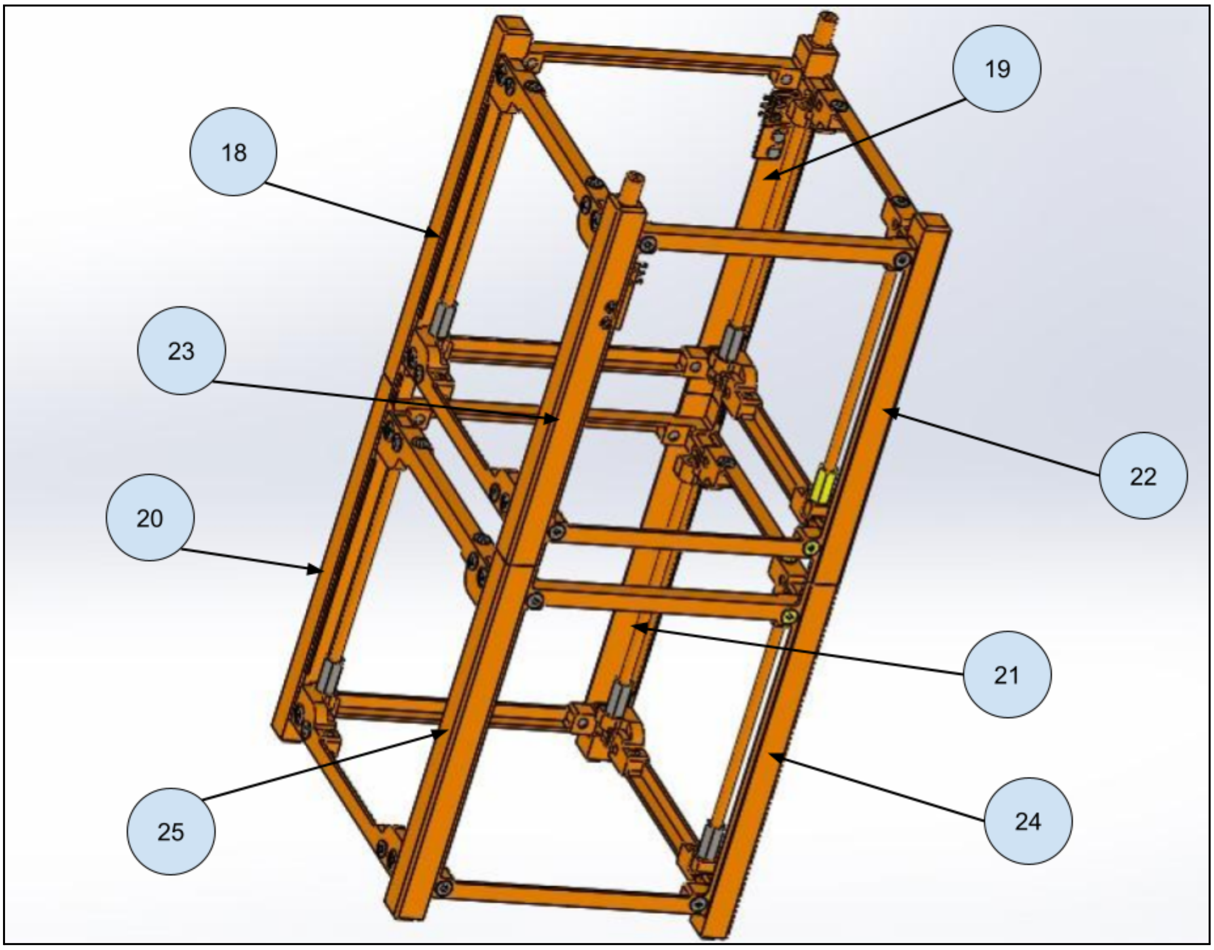

Fig Cubesat Model HSN 18-25

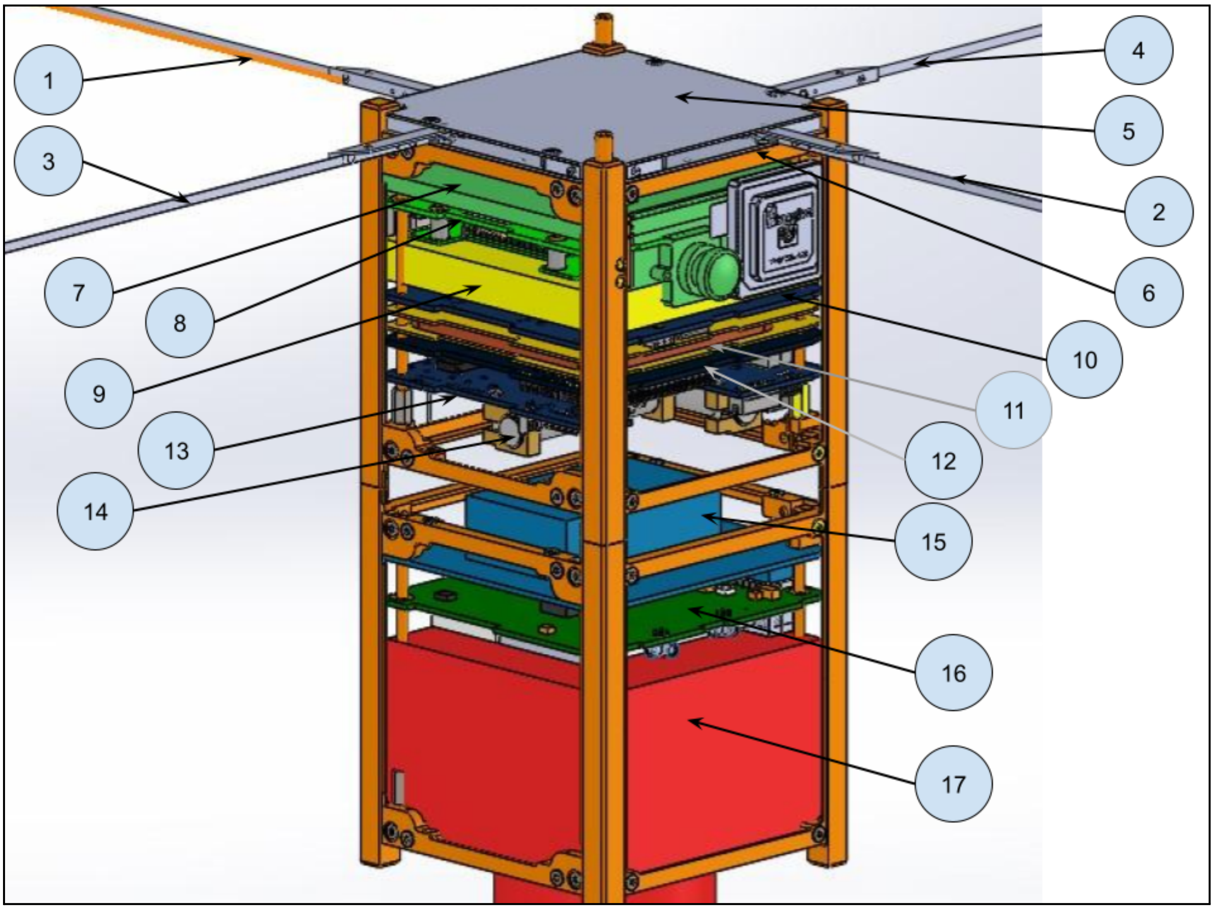

Fig Cubesat Model HSN 1-17

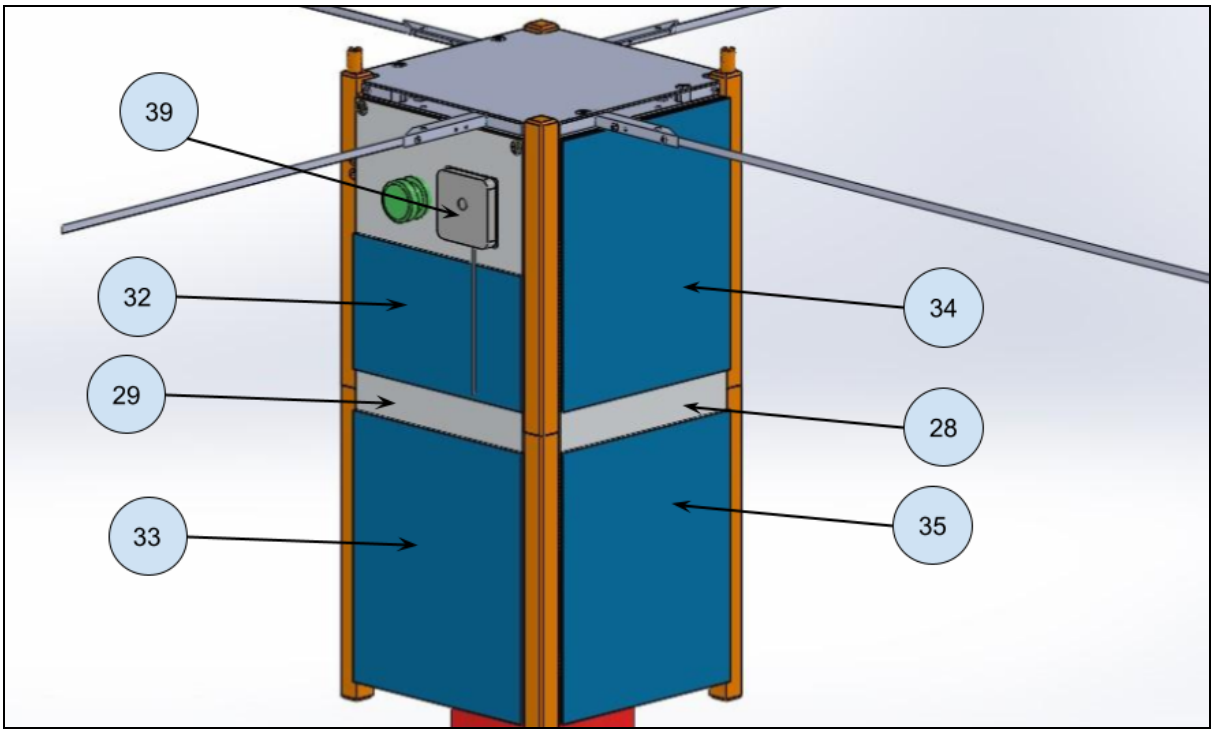

Fig Cubesat Model HSN 28, 29, 33-35, 39

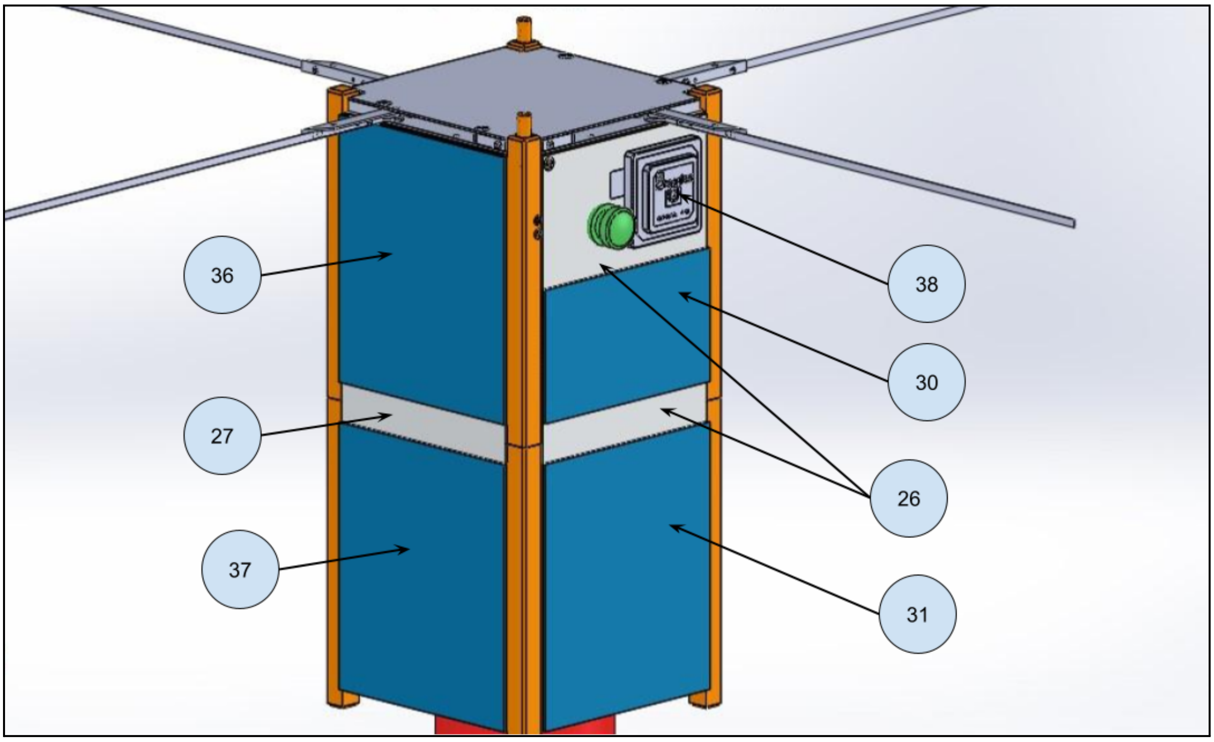

Fig Cubesat Model HSN 26, 27, 30, 31, 36-39

Radiation heat exchange between two IFN indexed *i* and *j* is given by following formula:

$$Q_{ij} = Rr_{ij} \times ({T_i}^4 - {T_j}^4)$$

$$Rr_{ij} = \frac{\sigma}{\frac{1-\epsilon_i}{\epsilon_i} + \frac{1}{A_i F_{ij}} + \frac{1-\epsilon_j}{\epsilon_j A_j}}$$

Where,

$\sigma =$Stefan–Boltzmann constant

$F_{ij} =$view factor from IFN i to IFN j

$\epsilon_i,\ \epsilon_j =$emissivity of respective IFN

$T_i,\ T_j =$temperatures of respective IFN

$A_i,\ A_j =$interface areas of respective IFN}

Conduction heat exchange between two IFN indexed *i* and *j* is given by following formula:

$$Q_{conductionij} = \frac{k_{ij} \times A_{ij}}{L_{ij}} \times (T_i - T_j)$$

The heat received from sun by an IFN is give by:

$$Q_{i\_sun} = F_{sun} \times \alpha_i \times A_i \times sin(\theta) q_{irradiance}$$

Where,

$\theta =$sun elevation angle

The heat received from Earth emitted IR by an IFN is give by:

$$Q_{i\_Earth} = F_{Earth} \times \epsilon_i \times A_i q_{orbit}$$

$$q_{orbit} = q_{surface} \times \frac{A_{\text{orbit surface}}}{A_{\text{Earth Surface}}}$$

Where,

F_{Earth} =access to Earth
α_i =absorptivity of IFN
ϵ_i =emissivity of IFN i
q_{orbit} =per unit area Earth IR in orbit surface, J/m^2
$q_{surface}$ =IR emitted by Earth at surface
$A_{\text{orbit surface}}$ =surface area of orbit
$A_{\text{Earth Surface}}$ =surface area of Earth
Heat rejected to deep space by an IFN is given by:

$$Q_{space\_i} = F_{space}\sigma\epsilon_i A_i(T_i^4 - T_{space}{}^4)$$

Where,
F_{space} =access to space
T_{space} =temperature of deep space = 3 K
Total heat exchange between two IFNs is the sum of heat exchanged by all methods. It can be written as:

$$Q_{ij} = Q_{ij\_conduction} + Q_{ij\_radiation}$$

The total heat exchange in an IFN is the sum of heat exchanged between the IFN and other IFNs and heat exchanged between external elements. Same is explained below:

	1	2	3	4	5	6	7	8	...	i-5	i-4	i-3	i-2	i-1	i
1	Q_1_1	Q_2_1	Q_3_1	Q_4_1	Q_5_1	Q_6_1	Q_7_1	Q_8_1	...	Q_(i-5)_1	Q_(i-4)_1	Q_(i-3)_1	Q_(i-2)_1	Q_(i-1)_1	Q_i_1
2	Q_1_2	Q_2_2	Q_3_2	Q_4_2	Q_5_2	Q_6_2	Q_7_2	Q_8_2	...	Q_(i-5)_2	Q_(i-4)_2	Q_(i-3)_2	Q_(i-2)_2	Q_(i-1)_2	Q_i_2
3	Q_1_3	Q_2_3	Q_3_3	Q_4_3	Q_5_3	Q_6_3	Q_7_3	Q_8_3	...	Q_(i-5)_3	Q_(i-4)_3	Q_(i-3)_3	Q_(i-2)_3	Q_(i-1)_3	Q_i_3
4	Q_1_4	Q_2_4	Q_3_4	Q_4_4	Q_5_4	Q_6_4	Q_7_4	Q_8_4	...	Q_(i-5)_4	Q_(i-4)_4	Q_(i-3)_4	Q_(i-2)_4	Q_(i-1)_4	Q_i_4
5	Q_1_5	Q_2_5	Q_3_5	Q_4_5	Q_5_5	Q_6_5	Q_7_5	Q_8_5	...	Q_(i-5)_5	Q_(i-4)_5	Q_(i-3)_5	Q_(i-2)_5	Q_(i-1)_5	Q_i_5
6	Q_1_6	Q_2_6	Q_3_6	Q_4_6	Q_5_6	Q_6_6	Q_7_6	Q_8_6	...	Q_(i-5)_6	Q_(i-4)_6	Q_(i-3)_6	Q_(i-2)_6	Q_(i-1)_6	Q_i_6
7	Q_1_7	Q_2_7	Q_3_7	Q_4_7	Q_5_7	Q_6_7	Q_7_7	Q_8_7	...	Q_(i-5)_7	Q_(i-4)_7	Q_(i-3)_7	Q_(i-2)_7	Q_(i-1)_7	Q_i_7
8	Q_1_8	Q_2_8	Q_3_8	Q_4_8	Q_5_8	Q_6_8	Q_7_8	Q_8_8	...	Q_(i-5)_8	Q_(i-4)_8	Q_(i-3)_8	Q_(i-2)_8	Q_(i-1)_8	Q_i_8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
i-5	Q_1_(i-5)	Q_2_(i-5)	Q_3_(i-5)	Q_4_(i-5)	Q_5_(i-5)	Q_6_(i-5)	Q_7_(i-5)	Q_8_(i-5)	...	Q_(i-5)_(i-5)	Q_(i-4)_(i-5)	Q_(i-3)_(i-5)	Q_(i-2)_(i-5)	Q_(i-1)_(i-5)	Q_i_(i-5)
i-4	Q_1_(i-4)	Q_2_(i-4)	Q_3_(i-4)	Q_4_(i-4)	Q_5_(i-4)	Q_6_(i-4)	Q_7_(i-4)	Q_8_(i-4)	...	Q_(i-5)_(i-4)	Q_(i-4)_(i-4)	Q_(i-3)_(i-4)	Q_(i-2)_(i-4)	Q_(i-1)_(i-4)	Q_i_(i-4)
i-3	Q_1_(i-3)	Q_2_(i-3)	Q_3_(i-3)	Q_4_(i-3)	Q_5_(i-3)	Q_6_(i-3)	Q_7_(i-3)	Q_8_(i-3)	...	Q_(i-5)_(i-3)	Q_(i-4)_(i-3)	Q_(i-3)_(i-3)	Q_(i-2)_(i-3)	Q_(i-1)_(i-3)	Q_i_(i-3)
i-2	Q_1_(i-2)	Q_2_(i-2)	Q_3_(i-2)	Q_4_(i-2)	Q_5_(i-2)	Q_6_(i-2)	Q_7_(i-2)	Q_8_(i-2)	...	Q_(i-5)_(i-2)	Q_(i-4)_(i-2)	Q_(i-3)_(i-2)	Q_(i-2)_(i-2)	Q_(i-1)_(i-2)	Q_i_(i-2)
i-1	Q_1_(i-1)	Q_2_(i-1)	Q_3_(i-1)	Q_4_(i-1)	Q_5_(i-1)	Q_6_(i-1)	Q_7_(i-1)	Q_8_(i-1)	...	Q_(i-5)_(i-1)	Q_(i-4)_(i-1)	Q_(i-3)_(i-1)	Q_(i-2)_(i-1)	Q_(i-1)_(i-1)	Q_i_(i-1)
i	Q_1_i	Q_2_i	Q_3_i	Q_4_i	Q_5_i	Q_6_i	Q_7_i	Q_8_i	...	Q_(i-5)_i	Q_(i-4)_i	Q_(i-3)_i	Q_(i-2)_i	Q_(i-1)_i	Q_i_i

$$Q_{1\_column} = Q_{1\_1} + Q_{2\_1} + Q_{3\_1} + \dots + Q_{(i-2)\_1} + Q_{(i-1)\_1} + Q_{i\ 1}$$

$$Q_{1\_row} = -Q_{1\_1} - Q_{1\_2} - Q_{1\_3} - \dots - Q_{1\_1(i-3)} - Q_{1\_1(i-2)} - Q_{1\_i-1} - Q_{1\_i}$$

$$Q_1 = Q_{1\_column} + Q_{1\_row} + Q_{generated} + Q_{1\_external}$$

Here the Q_i is total heat flow for IFNs, to calculate for HSNs we sum up the total heat flow of IFNs associated with that HSN. For example, for total heat flow from *Wall Nadir* i.e. HSN 26, we sum up heat flow in IFNs 29 and 33. The generalise form for total heat exchange from an HSN can be written as:

$$HSN_p = IFN_{p\_1} + IFN_{p\_2} + ... + IFN_{p\_(q-2)} + IFN_{p\_(q-1)} + IFN_{p\_q}$$

$$HSN_{WallNadir} = Q_{29} + Q_{33}$$

To calculate change in temperature, thermal inertia of the HSN is considered. The temperature of the HSN at any time step *t* can be given as:

$$T_{p@t} = T_{p@(t-1)} + \frac{HSN_p}{m \times C_p}$$

The input data for the calculation is in a table or array format. Values of view factor, access, areas, emissivity, absorptivity, conductivity, cross-sectional areas and length of conducting elements, IFNs and HSNs association, masses, specific heat, heat generated, initial temperatures, sun elevation are input provided by user defined tables or arrays.

Process Flow in the code:

0) Time step size is set to 1 second
1) Values for Stefan–Boltzmann constant, solar irradiance, radius of Earth, altitude of orbit, total simulation time, temperature of space and temperature of Earth are defined
2) Values for Earth and orbit surface are calculated
3) Input datafiles in csv format are imported
4) Number of HSNs and IFNs is counted
5) Value of R_{ij} is calculated using following loop
    a) First loop for i
        i) Set values associated with i from arrays
        ii) Second loop for j
            (1) Set values associated with j from arrays
            (2) Calculate value of R_{ij}
            (3) Store value of R_{ij} in R_{ij} array at row index of j and column index i
        iii) Second loop complete
    b) First loop complete
6) Save R_{ij} to csv file

7) Create array to store: net heat in IFNs, net heat in HSNs, temperature of IFNs, temperature of HSNs, net heat flow between node i and j

8) Loop for timestep

    a) Loop for IFNs at index *i*

        i) Create variables to store heat flow by radiation and heat flow by conduction from IFN at index *i* through rows

        ii) Loop for going through all the rows *j* of same column index *i*

            (1) Get value of R_{ji} from corresponding array

            (2) Get values of conductivity, length, cross-sectional area of conduction element between *i* and *j*

            (3) Get values of temperature for *i* and *j* at corresponding timestep

            (4) Calculate heat exchange between *j* and *i* by conduction

            (5) Subtract the conduction heat flow value from corresponding variable of heat flow by conduction through rows

            (6) Calculate heat exchange between *j* and *i* by radiation

            (7) Subtract the radiation heat flow value from corresponding variable of heat flow by radiation through rows

            (8) If this is last timestep then calculate total heat flow from *i* to *j* and store it Q_{ji} array at row index *j* and column index *i*

        iii) Create variables to store heat flow by radiation and heat flow by conduction from IFN at index *i* through columns

        iv) Loop for going through all the columns *k* of same row index *i*

            (1) Get value of R_{ik} from corresponding array

            (2) Get values of conductivity, length, cross-sectional area of conduction element between *i* and *k*

            (3) Get values of temperature for *i* and *k* at corresponding timestep

            (4) Calculate heat exchange between *i* and *k* by conduction

            (5) Add the conduction heat flow value to corresponding variable of heat flow by conduction through columns

            (6) Calculate heat exchange between *i* and *k* by radiation

(7) Add the radiation heat flow value to corresponding variable of heat flow by radiation through columns

(8) If this is last timestep then calculate total heat flow from *k* to *i* and store it Q_{ik} array at row index *i* and column index *k*

v) Get values for absovity, sun elevation, sun access, emissivity, area of sun interaction, Earth access, space access and HSN index corresponding to IFN index *i*

vi) Get value of temperature of IFN at index *i* from previous timestep or from initial temperature

vii) Calculate heat received from sun by IFN at index *i*

viii) Calculate heat received from Earth IR by IFN at index *i*

ix) Calculate heat flow between from Earth and IFN at index *i*

x) Calculate heat rejected to space by IFN at index *i*

xi) Calculate total heat flow from IFN at index *i*

xii) Add total heat flow from IFN at index *i* to corresponding HSN in the array at the timestep

b) Loop through all the HSN for temperature calculation

i) Add generated heat to the total heat flow from the HSN at the timestep in the array

ii) Get values of mass and specific heat of the HSN

iii) Calculate and add change in temperature of the HSN at the timestep

iv) Store the temperature values of the HSN in array

c) Loop through all the IFNs for temperature assignment

i) Get associated HSN number for the IFN

ii) Get temperature of the associated HSN and assign it to the IFN

iii) Store the IFN temperature in the respective array

9) Save the created arrays of net heat flow of IFN, net heat flow of HSN, Net heat flow between IFN *i* and *j*, temperature of IFN and temperature of HSN in csv files.

Python Code:

Record start time as per the time in PC:

```
import time
start = time.time()
```

Importing required libraries

```
import numpy as np
import math as mt
import pandas as pd
```

Defining basic parameters for the analysis:

```
t_s = 1 #time step size in seconds
sigma = 5.6703 * (10**(-8)) #J/m2sK4 #Stefan-Boltzmann constant
S_irr = 1344 #solar irradiance in W/m^2
R_e = 6371*1000 # Earth radius in m
A_Surface = 4 * mt.pi * (R_e**2) #Earth surface area in m^2
q_surface = 237 # infrared reflected by Earth on surface W/m^2
Alt_Orbit = 500*1000 #orbit altitude in m
A_orbit = 4 * mt.pi * ((R_e+Alt_Orbit)**2) #surface of sphere of
with orbit altitude as radius in m^2
q_orbit = q_surface * (A_Surface /A_orbit) #infrared reflected by
Earth on at orbit surface in W/m^2
###total time###
total_time = 1800 #total simulation time in seconds
######
T_space = 3 # Space temperature in K
T_Earth = 303 #Earth temperature in K
```

Importing input CSV files:

```
#importing input data
operating_folder = 'E:\Thm_mod' #location of the working directory
F_i_j = pd.read_csv(operating_folder + '\F_i_j.csv') #view factors
for radiation
A_rad = pd.read_csv(operating_folder + '\A_rad.csv') #area of
interacting surfaces
emmv = pd.read_csv(operating_folder + '\e.csv') #emissivity
F_Sun = pd.read_csv(operating_folder + '\F_sun.csv') #access to
sun for the interface nodes
absv = pd.read_csv(operating_folder + '\Absv.csv') #absorptivity
Ele = pd.read_csv(operating_folder + '\ele.csv') #elevation at
every second
F_Earth = pd.read_csv(operating_folder + '\F_Earth.csv') #access
to Earth for the interface nodes
Conductivity = pd.read_csv(operating_folder + '\k_i_j.csv')
```

```
#Conductivity of conduction interfaces
Area_Cond = pd.read_csv(operating_folder + '\A_cond.csv')
#cross-sectional area of Conduction area
Length_Cond = pd.read_csv(operating_folder + '\L_cond.csv')
#length of conduction interfaces
cont_res = pd.read_csv(operating_folder + '\Cont_Res.csv')
#contact resistance
F_space = pd.read_csv(operating_folder + '\F_space.csv') #access
to deep space for the interface nodes
node_comb = pd.read_csv(operating_folder + '\Combining_nodes.csv')
#node combinations for stored energy
mass = pd.read_csv(operating_folder + '\mass.csv') #mass in kg
Sp_Heat = pd.read_csv(operating_folder + '\Cp.csv') #specific heat
in J/kgK
Ti = pd.read_csv(operating_folder + '\Ti.csv') #initial
temperatures
Q_GEN = pd.read_csv(operating_folder + '\heat_generated.csv')
#heat generated at each timestep in heat storing nodes
print('csv files read') #reading input CSV files
```

Heat transfer and heat storage nodes:

```
act_node = node_comb.Heat_Storage_Node.nunique() #calculating
number of heat storing nodes
print (('heat storage nodes = ') + (str(act_node))) #number of
heat storing nodes
```

Initialising to find out matrix of R_{ij}

```
##Initialization###
J = (len(F_i_j)) #nos of rows
I = (len(F_i_j.columns)) #nos of columns
Rr_i_j = pd.DataFrame(index=range(J),columns=range(I)) #creating
dataframe for storing multiplying factor for radiation

i = 0
while i < (len(F_i_j.columns)):
      e_1 = float(emmv.iat[0,i]) #emissivity of 1
      A_1 = float(A_rad.iat[0,i]) #area of 1
      j = 0
      while j < (len(F_i_j)):
      e_2 = float(emmv.iat[0,j]) #emissivity of 2
      A_2 = float(A_rad.iat[0,j]) #area of 2
```

```
        F_1_2 = F_i_j.iat[j,i] #view factor for 1 to 2

        if F_1_2 == 0:
                Rr_1_2 = 0
        else:
                Rr_1_2 = 
sigma/(((1-e_1)/(e_1*A_1))+(1/(A_1*F_1_2))+((1-e_2)/(e_2*A_2)))
#W/(K^4) #calculating multiplying factor for radiation for 1 to 2
W/K^4

        Rr_i_j.iloc[j,i] = Rr_1_2 #storing the calculated multiplying
factor for radiation for i to j W/K^4

        j = j+1
        i = i+1

Rr_i_j.to_csv('Rr_i_j.csv') #writing csv file for multiplying
factors for radiation for i to j W/K^4
print('initialization complete')
```

Creating Dataframes

```
Heat exchange calculations
Q_net = pd.DataFrame(index=range(total_time),columns=range(I))
#creating dataframe to store total power in each interface node
Q_store = pd.DataFrame(np.zeros((total_time, act_node))) #creating
dataframe to store energy stored in each heat storage node
Temp = pd.DataFrame(index=range(total_time),columns=range(I))
#creating dataframe to store temperature of each interface node
T_i_store = pd.DataFrame(np.zeros((1, act_node))) #creating
dataframe to store initial temperature for each heat storage node
T_store = pd.DataFrame(np.zeros((total_time, act_node))) #creating
dataframe to store temperature for each heat storage node
T = pd.DataFrame(index=range(total_time),columns=range(I))
#creating dataframe to store temperature of each interface node
Q_i_j = pd.DataFrame(index=range(J),columns=range(I)) #creating
dataframe to store heat transfer between of each interface node i
to J and i to k

print('dataframes created')
```

Initialising timestep process

```
####Heat exchange calculations###
timestep = 0
while timestep < total_time:
```

Calculating heat transfer from node *i* to node *j* and the summing them up as heat rejected from interface node *i*

```
      i = 0 #interface node number
      while i < (len(Rr_i_j.columns)):
      #radiation heat transfer due to interaction
      ###Going through rows keeping column same###
      j = 0 #row number
      q_net_row = 0 #total heat transfer along the rows with
radiation
      q_cond_net_row = 0 #total heat transfer along the rows with
conduction
      q_cont_net_row = 0 #total heat transfer along the rows with
contact
      while j < ((len(Rr_i_j))-1):
            r = Rr_i_j.iat[j,i] #taking value of multiplying factor
for radiation along row while keeping the column index = node
number (column = i, row = j)
            cond = Conductivity.iat[j,i] #taking value of
conductivity along row while keeping the column index = node
number (column = i, row = j)
            L = Length_Cond.iat[j,i] #taking value of conduction
interface length along row while keeping the column index = node
number (column = i, row = j)
            A_cond = Area_Cond.iat[j,i] #taking value of
cross-section of conduction interface along row while keeping the
column index = node number (column = i, row = j)
            res_cont = cont_res.iat[j,i] #taking value of contact
resistance along row while keeping the column index = node number
(column = i, row = j)
            if timestep == 0: #condition chose temperature of last
timestep or initial temperature
                  T_1 = float(Ti.iat[0,i]) #initial temperature of i
as 1
                  T_2 = float(Ti.iat[0,j]) #initial temperature of j
as 2
            else:
```

```
                T_1 = float(T.iat[(timestep-1),i]) #temperature of
i in previous time step for 1
                T_2 = float(T.iat[(timestep-1),j]) #temperature of
j in previous time step for 2

           ###conduction###
           if cond == 0: #condition to avoid 'divided by zero
error'
                q_cond = 0
           else:
                q_cond = ((cond * A_cond) / L) * (T_1 - T_2)
#calculating heat transfer between i and j by conduction
           q_cond_net_row = q_cond_net_row - q_cond #heat transfer
from node i to other nodes along the row by conduction

           ###contact###
           if res_cont == 0: #condition to avoid 'divided by zero
error'
                q_cont = 0
           else:
                q_cont = (T_1-T_2)/res_cont #calculating heat
transfer between i and j by contact
           q_cont_net_row = q_cont_net_row - q_cont #heat transfer
from node i to other nodes along the row by contact

           ###Radiation###
           q = r * (((T_1)**4)-((T_2)**4)) #calculating heat
transfer between i and j by radiation
           q_net_row = q_net_row - q #heat transfer from node i to
other nodes along the row by radiation

           if timestep == (total_time-1): #checking if last time
step
                q_net_i_j = (q + q_cont + q_cond) * (-1)
#calculating total heat transfer from interface node k to i
                Q_i_j.iloc[j,i] = q_net_i_j #storing total heat
transfer from interface node k to i in the dataframe
           #print(('i = ')+(str(i)) + (' j = ') + (str(j)))
           j = j+1
```

Calculating heat transfer from node *k* to node *i* and the summing them up as heat added to interface node *i*

```
###Going through columns, keeping row same###
      k = 0 #column number
      q_net_col = 0 #total heat transfer along the column with radiation
      q_cond_net_col = 0 #total heat transfer along the columns with conduction
      q_cont_net_col = 0 #total heat transfer along the columns with contact
      while k < (len(Rr_i_j)-1):
            r = Rr_i_j.iat[i,k] #taking value of multiplying factor for radiation along column while keeping the row index = node number (row = i, column = k)
            cond = Conductivity.iat[i,k] #taking value of conductivity along column while keeping the row index = node number (row = i, column = k)
            L = Length_Cond.iat[i,k] #taking value of conduction interface length along column while keeping the row index = node number (row = i, column = k)
            A_cond = Area_Cond.iat[i,k]  #taking value of cross-section of conduction interface along column while keeping the row index = node number (row = i, column= k)
            res_cont = cont_res.iat[i,k] #taking value of contact resistance along column while keeping the row index = node number (row = i, column = k)
            if timestep == 0: #condition chose temperature of last timestep or initial temperature
                  T_1 = float(Ti.iat[0,k]) #initial temperature of i as 1
                  T_2 = float(Ti.iat[0,i]) #initial temperature of k as 2
            else:
                  T_1 = float(T.iat[(timestep-1),k]) #temperature of i in previous time step for 1
                  T_2 = float(T.iat[(timestep-1),i]) #temperature of k in previous time step for 2

            ####Conduction###
            if cond == 0: #condition to avoide 'divided by zero error'
                  q_cond = 0
            else:
                  q_cond = ((cond*A_cond)/L) * (T_1 - T_2)
```

```
#calculating heat transfer between i and k by conduction
            q_cond_net_col = q_cond_net_col + q_cond #heat transfer
from node i to other nodes along the column by conduction

            ###contact###
            if res_cont == 0: #condition to avoide 'divided by zero
error'
                  q_cont = 0
            else:
                  q_cont = (T_1-T_2)/res_cont #calculating heat
transfer between i and k by contact
            q_cont_net_col = q_cont_net_col + q_cont #heat transfer
from node i to other nodes along the column by contact

            ###Radiation###
            q = r * (((T_1)**4)-((T_2)**4)) #calculating heat
transfer between i and k by radtiation
            q_net_col = q_net_col + q #heat transfer from node i to
other nodes along the column by radiation

            if timestep == total_time: #checking if last time step
                  q_net_k_i = q + q_cont + q_cond #calculating total
heat transfer from interface node k to i
                  Q_i_j.iloc[i,k] = q_net_k_i #storing total heat
transfer from interface node k to i in the dataframe

            #print(('i = ')+(str(i)) + (' k = ') + (str(k)))
            k = k+1
```

Calculating external heat transfer such as from sun, Earth and deep space:

```
      #heat received from sun
      a = float(absv.iat[0,i]) #absorvity of the interface node
      Tht = float(Ele.iat[timestep,0])  #theta = sun elevation
value at that timestep
      f_sun = float(F_Sun.iat[0,i])  #access to sun by the
interface node i
      e = float(emmv.iat[0,i]) #emmisivity of the interface node i
      Area = float(A_rad.iat[0,i]) #area of the interface node i
when interacting with sun
      f_earth = float(F_Earth.iat[0,i]) #access to Earth by the
interface node i
```

```
      f_space = float(F_space.iat[0,i]) #access to space by
interface node i
      node_store = node_comb.iat[i, 1] #corresponding storage node
index for i th interface node

      if timestep == 0: #condition to chose temperature of last
timestep or initial temperature
            T_node = float(Ti.iat[0,i]) #initial temperature of the
node
            T_i_store.iloc[0,node_store] = T_node #storing initial
temperature of heat storage node in K
      else:
            T_node = float(T.iat[(timestep-1),i]) #temperature of
the node at the previous timestep

      q_sun = f_sun * a * S_irr * Area *
(mt.sin(mt.radians(abs(Tht)))) # solar irradiance on interface
node i in W = J/s
      q_space = -1 * f_space * sigma * e * Area *
((T_node**4)-(T_space**4)) #heat loss to deep space by interface
node i in W = J/s
      q_earth = -1 * f_earth * e * sigma * Area *
((T_node**4)-(T_Earth**4)) #energy radiated to Earth by interface
node i in W = J/s
      q_ir = f_earth * e * Area * q_orbit #earth emmited IR
received by interface node i in W = J/s
      q_net = q_cont_net_row + q_cont_net_col + q_cond_net_row +
q_cond_net_col + q_net_col + q_net_row + q_sun + q_space + q_earth
+ q_ir #net heat transfer in the interface node in W = J/s

      #storing value of total heat stored in corresponding heat
storage node
      Q_net.iloc[timestep,i] = q_net #storing value of total heat
exchange for i th interface node
      Q_store.iloc[timestep,node_store] =
Q_store.iloc[timestep,node_store] + (q_net * t_s) #total heat
energy exchanged in the from the storage node in the timestep in J
      #print(('i = ')+(str(i)))
      i = i+1
```

Calculating temperatures of each node based on total heat transfer to the node

```
    ###calculating and storing heat storage node temperature values###
    node = 0 #heat storage node index
    while node < (act_node):
    Q_store.iloc[timestep, node] = Q_store.iloc[timestep, node] + ((Q_GEN.iat[timestep,node])*t_s) #adding generated heat in the heat storage node
    q_store_temp = Q_store.iat[timestep, node] #total heat stored by the heat storage node at that timestep
    m = mass.iat[0,node] #mass of the the heat storage node
    Cp = Sp_Heat.iat[0,node] #specific heat capacity of the heat storage node
    if timestep == 0:
        Temp = T_i_store.iat[0,node] + (q_store_temp/(m*Cp)) #temperature of the heat storage node at first timestep
    else:
        Temp = T_store.iat[(timestep-1),node] + (q_store_temp/(m*Cp)) #temperature of the heat storage node at that timestep
    T_store.iloc[timestep,node] = Temp #storing value of the heat storage node in the dataframe
    node = node+1
```

Calculating temperature of the interface nodes

```
    ###assigning calculated values of temperature to virtual nodes###
    o = 0 #interface node index
    while o < I:
    temp_node = node_comb.iat[o, 1] #heat storage node corresponding to the interface node at index o
    T.iloc[timestep,o] = T_store.iat[timestep,temp_node] #storing temperature interface node at that timestep
    o = o+1

    #print(('timestep = ')+(str(timestep)))
    timestep = timestep + 1
```

Saving the results

```
#Saving results to csv files
Q_i_j.to_csv('Q_i_j.csv')
Q_net.to_csv('Q_net.csv')
```

```
Q_store.to_csv('Q_store.csv')
T_store.to_csv('T_store.csv')
T.to_csv('T_interface.csv')
print('run complete')
```

Recording time for calculations

```
end = time.time() - start
print("time required: ", end)
```