Multilayer Urban Canopy Model

Technical manual

Doan Quang Van

2018/10/10

1. Model description

Conceptual illustration

Multiple-layer urban canopy model (MUCM) consists of infinite arrays of solid three-dimensional buildings with horizontal cross-section. Building size and road width are uniform. The walls have four directions (east, west, south, and north). Buildings are resolved into multiple layers vertically. The model includes anthropogenic heat flux at the ground surface.

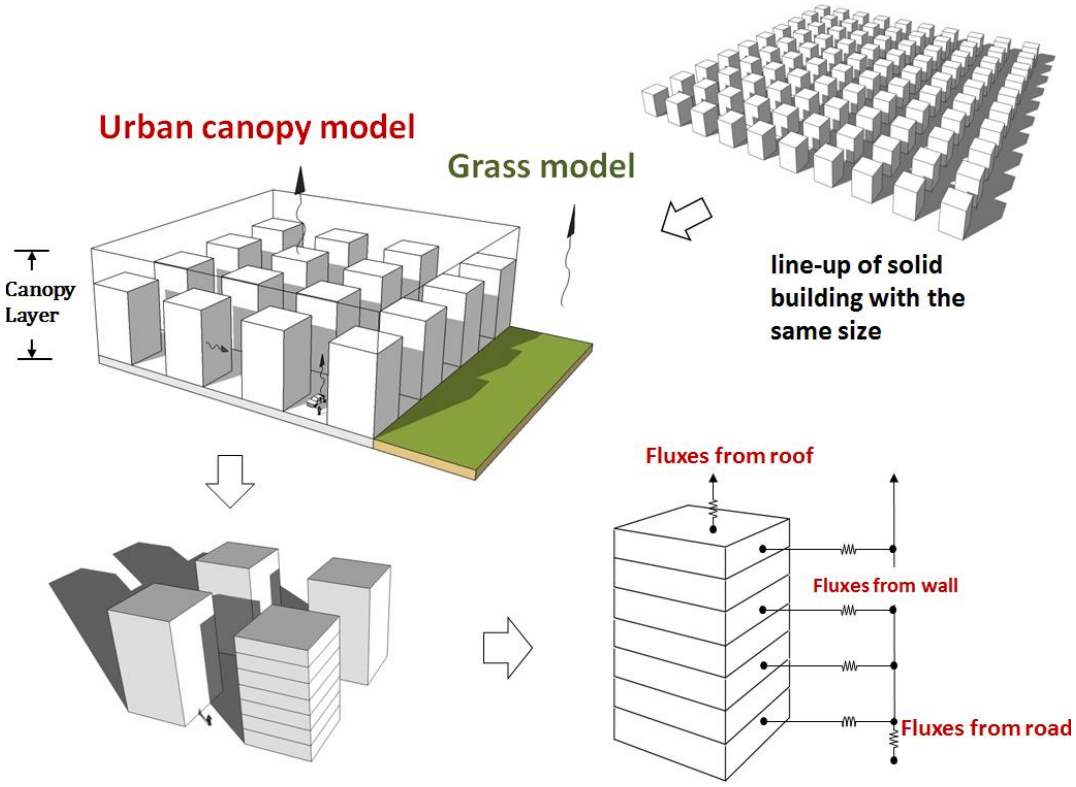


Figure 1 Conceptual image of multiple urban canopy model

For the green portion, a simple grass model is applied. The grass model considers the bulk effect of grass or vegetation land with parameters, i.e., thermal conductivity, heat capacity, roughness length and albedo provided. The total fluxes from the surfaces to above atmosphere are calculated by weighted average of fluxes from urban areas solved by the

MUCM and from vegetation areas by the grass model.

Governing equations

The governing equations of the MUCM are 1-D diffusion equations for momentum, potential temperature, and specific humidity (similar with Kondo et al., 2005):

$$\frac{\partial u}{\partial t} = \frac{1}{m}\frac{\partial}{\partial x}\left(K_m m \frac{\partial u}{\partial z}\right) + f(v - v_g) - c_d A u (u^2 + v^2)^{\frac{1}{2}} \quad (1)$$

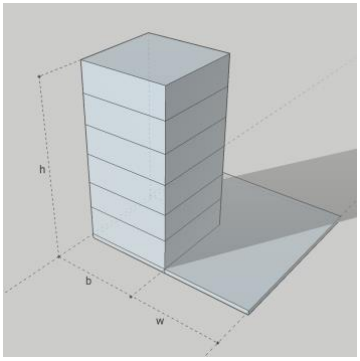
$$\frac{\partial v}{\partial t} = \frac{1}{m}\frac{\partial}{\partial x}\left(K_m m \frac{\partial v}{\partial z}\right) - f(u - u_g) - c_d A v (u^2 + v^2)^{\frac{1}{2}} \quad (2)$$

$$\frac{\partial \theta}{\partial t} = \frac{1}{m}\frac{\partial}{\partial x}\left(K_h m \frac{\partial \theta}{\partial z}\right) + \frac{Q_{AS}}{\rho c_p} \quad (3)$$

$$\frac{\partial q_v}{\partial t} = \frac{1}{m}\frac{\partial}{\partial x}\left(K_q m \frac{\partial q_v}{\partial z}\right) + \frac{Q_{AL}}{\rho l} \quad (4)$$

where u and v are the wind velocity components, θ is the potential temperature, q is the specific humidity, f is the Coriolis parameter, u_g, v_g are the geostrophic wind components, m is the volume porosity, c_d is the drag coefficient, ρ is the air density, l is the evaporative latent heat, and b is the average building width and w is the average road width. Here Q_{AS} and Q_{AL} are the sensible heat and the latent heat exchanges, respectively, between the building's walls and the atmosphere. $K_m, K_h,$ and K_q are the momentum, heat, and water vapor diffusivities, respectively, which are calculated from the Mellor-Yamada model at level 2 (Mellor and Yamada, 1974). The heat budget equation was solved for each urban surface to compute heat flux exchange between surfaces and the atmosphere (equation)

$cau(u^2+v^2)^{1/2}$、$cav(u^2+v^2)^{1/2}$ Are drag effects of the building and has been added in the equations.



$$a = \frac{b.P_w(z)}{(b+w)^2 - b^2.P_w(z)}$$

$$P_w = 0$$

$$P_w = 1$$

Scheme configuration

Table 1. List of schemes used

Schemes	Description
Temporal finite difference (diffusion term)	Euler implicit method
Spatial finite difference (diffusion term)	2nd order centered difference
Planetary boundary layer model	Mellor & Yamada Level 2
Short wave radiation	Kondo (1994)
Long wave radiation	Kondo (1994)
Surface process	1) Slab Model 2) Multilayer urban canopy model
Momentum, sensible and latent heat fluxes exchange (from roof and ground)	Monin-Obukhov method
Momentum, sensible and latent heat fluxes exchange (from wall surfaces)	Jurges method
Boundary condition	Upper: fixed Lower: flux

2. Mathematical descriptions

2.1 Surface heat budget

The heat budget equation was solved for each urban surface to compute heat flux exchange between surfaces and the atmosphere (Eq. 5):

$$R_i = H_i + lE_i + G_i \quad (5)$$

The sensible and latent heat fluxes from ground and building roof surfaces were computed by the Monin-Obukhov similarity method; whereas, those from wall surfaces were calculated by the Jurges formula. On the other hand, the one dimensional thermal conduction equation was used to compute the heat exchange inside the urban structure, i.e., road, building walls and roof.

$$G = -\lambda \partial T_x / \partial x \quad (6)$$

$$\frac{\partial T_x}{\partial t} = \frac{1}{\rho c}\frac{\partial G}{\partial x} \quad (7)$$

where G is sensible heat flux; λ is heat conductivity; T_x is interior temperature at depth x of road, walls, or roof.

Monin-Obukhov method

$$\tau = \rho C_M U^2$$

$$H = rC_H U\left(T_s - T\right)$$

Calculate the bulk Richardson number

$$Rib = \frac{gz}{\Theta}\cdot\frac{\theta - \theta_0}{U^2}$$

Jurges formula

2.2 Ray tracing scheme

Ray tracing is a rendering method that can produce realistic lighting effect. Essentially, the algorithm can trace the path of light, and then simulate the way that the light interacts with the visual objects. From the middle of facet i one can draw a ray $\boldsymbol{r}$. Ray $\boldsymbol{r}$ is defined by its origin and extending direction. Scan all possible direction by gradually moving the shot angle via it horizontal θ and vertical ϕ (Fig. 2a).

View factors from facet i include the sky view factor ($F_{i\to s}$) and geometric view factor ($F_{i\to j}$) from the facet to other facets of urban canopy.

$$F_{i\to j} = \frac{1}{n_\gamma n_\phi}\sum_{\gamma,\phi} I_{i\to j}(\gamma,\phi) \tag{8}$$

$$F_{i\to s} = 1 - \sum_{j}^{n} F_{i\to j} \tag{9}$$

where $I_i(\gamma,\phi) = 1$ if ray $\boldsymbol{r}_i(\gamma,\phi)$ that extents from the middle point of facet i infinitely in a direction defined by θ and ϕ intersect with facet S_j. Otherwise, $I_i(\gamma,\phi) = 0$.

For calculating the sunlit area on object, the similar method is used. For each urban facet, we divided into pixels. From each pixel draw the ray from the middle point of this pixel to solar knowing the solar elevation and azimuth angle.

$$S_i = 1 - \frac{1}{n_\psi}\sum_{\psi}^{n_\psi} I_{i\to o}(\psi) \tag{10}$$

where $I_{i\to o}(\psi) = 1$ if ray $\boldsymbol{r}_i(\psi)$ extending from pixel ψ on facet i toward the sun hits the other objects, otherwise, $I_{i\to o}(\psi) = 0$; n_ψ is the total number of pixels on facet i.

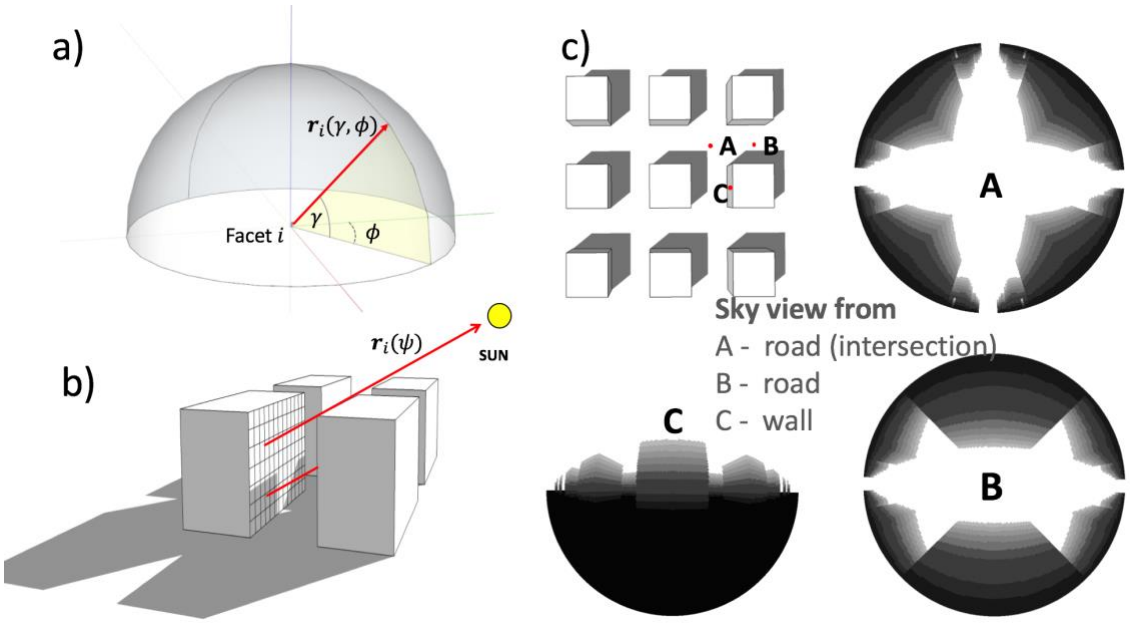


Fig. 2. Generating a ray from a reference point (a); schematic illustration of calculation of sunlit proportion on facets (b); The sky view from three sample points: on road (A and B), road (B) and on building wall (C). The fading-away gray indicates different urban layers.

2.3 Radiation process

The direct short-wave radiation fluxes hitting wall surface with 4 directions (E, W, S, N) are calculated as

$$I_{de} = -I_d \cos\theta \sin\varphi$$

$$I_{dw} = I_d \cos\theta \sin\varphi$$

$$I_{ds} = I_d \cos\theta \cos\varphi$$

$$I_{dn} = -I \cos\theta \cos\varphi$$

2.4 PBL scheme

3. Discrete solution

3.1 Diffusion equation

$$\frac{q_i^n - q_i^{n-1}}{\Delta t} = \frac{1}{m_i}\frac{K_{i+1}m_{i+1}\dfrac{q_{i+1}^n - q_i^n}{z_{i+1} - z_i} - K_i m_i \dfrac{q_i^n - q_{i-1}^n}{z_i - z_{i-1}}}{(z_{i+1} - z_{i-1}) \times 0.5} + F_i$$

$$\frac{q_i^n - q_i^{n-1}}{\Delta t} = \frac{1}{m_i}\frac{K_{i+1}m_{i+1}\dfrac{q_{i+1}^n - q_i^n}{\Delta z_i} - K_i m_i \dfrac{q_i^n - q_{i-1}^n}{\Delta z_{i-1}}}{(\Delta z_i + \Delta z_{i-1}) \times 0.5} + F_i$$

$$\frac{q_i^n - q_i^{n-1}}{\Delta t} = \frac{K_{i+1}m_{i+1}}{m_i \Delta z_i (\Delta z_i + \Delta z_{i-1}) \times 0.5}(q_{i+1}^n - q_i^n) - \frac{K_i m_i}{m_i \Delta z_{i-1}(\Delta z_i + \Delta z_{i-1}) \times 0.5}(q_i^n - q_{i-1}^n) + F_i$$

$$a_i = -\frac{K_{i+1}m_{i+1}\Delta t}{m_i \Delta z_i (\Delta z_i + \Delta z_{i-1}) \times 0.5}$$

$$b_i = 1 + \frac{K_{i+1}m_{i+1}\Delta t}{m_i \Delta z_i (\Delta z_i + \Delta z_{i-1}) \times 0.5} + \frac{K_i m_i \Delta t}{m_i \Delta z_{i-1}(\Delta z_i + \Delta z_{i-1}) \times 0.5}$$

$$c_i = -\frac{K_i m_i \Delta t}{m_i \Delta z_{i-1}(\Delta z_i + \Delta z_{i-1}) \times 0.5}$$

$$d_i = F_i + q_i^{n-1}$$

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

Thomas method for solving 1-D diffusion equation.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2,3,\ldots,n-1 \end{cases}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2,3,\ldots,n. \end{cases}$$

The solution is then obtained by back substitution:

For heat equation

The heat flux added to air from height of z within dz is represented as:

$$F_i = \frac{Q_i}{\rho c_p}$$

$$Q_i \Delta z_i = \left\{ Pw_i - Pw_{i+1} \right\} Q_{roof} \frac{b^2}{(w+b)^2} + Pw_i Q_{wall_i} \frac{4b\Delta z_i}{(w+b)^2} + Q_A \Delta z_i$$

For moisture equation

For momentum equation

4. User manual

4.1 Compilation

- Go to src/compile
- Open Makefile.
- Change line 11-13 to make sure that fortran compiler and link to netcdf library work.

  FC = gfortran

  #FCFLAGS = -I${NETCDF}/include

  FCFLAGS = -I/usr/local/include
- Go back to parent directory.
- Clean up (delete .o, .mod): $ ./clean.sh
- Compile the sourse codes: $ ./compile.sh
- Run model: $ ./main

4.2 Namelist and parameter setting

namelist.ucm

Parameter	Example	Description
&time_control		
run_day	4	Length of run period
run_hour	0	
start_year	2014	Start date
start_month	8	
start_day	1	
start_hour	0	
start_minute	0	
start_second	0	
output_interval	600	Time interval for output (in sec)
ofiname	"./head_"	Head of output file name following by “_mucm_YYYY_MM_DD.nc”
/		

&domains		
dt	2.0d0	Length of time step (in sec)
kms	1	Start index of vertical grid
kme	120	End index of vertical grid
z_top	3000.0d0	
grid_type	1	Grid type: for ideal run only; if 0 then uniform grid spacing (dz = z_top / kme), if 1 then changeable grid spacing with the finer at near surface and courser upper level.
ics_path	"path/to/ics/"	Path to initial condition (for nesting run and for grid height only now)
lon	139.841d0	Longitude of reference point
lat	35.8349d0	Latitude of reference point
/		
&params		
tb	301.5d0	Base temperature (in K)
ganma	0.004d0	Potential temperature lapse rate (K/m)
lapse_rate	-0.01d0	Temperature lapse rate (K/m)
pb	1013.0d0	Base air pressure (hPa)
Ug	10.0d0	Geostrophic wind x component (m/s)
Vg	0.0d0	Geostrophic wind y component (m/s)
/		
&physics		
sf_surface_physics	2	If 1 then only slab model is used; if 2 MUCM and slab models are used for urban fraction and non-urban fraction, respectively.
sltype	3	Type of non-urban slab (parameter for this are set in LANDUSE.TBL)
nw	13	Number of soil layer
soil_temp	16.d0	Soil temperature at lowest layer (deg C)
utype	3	Urban type if sf_surface_physics=2 is chose. Parameters for urban type is set

		in URBANPARM.TBL
/		
&dynamics		
bl_pbl_physics	1	Boundary layer diffusion scheme, only 1 for Mellor-Yamada level 2. If 0 then constant km and kh.
/		
&nesting		
ow_nesting	0	If 0 no nesting (ideal run); If 1 nesting run, then data for vertical grid and input data for temperature, moisture, wind (2 components) at top and radiation data (at bottom) need to be prepared.
dt_obs	3600.d0	Time interval for input data (in sec)
input_fname	"preprocess/20180718/data/ucm_data/kugahara_topvars_20050901.csv"	Path to input data
/		
&end		

URBPARM.TBL

Parameters for urban canopy model.

LANDUSE.TBL

Parameters for types of land use used for slab model are set here (no need to change).

.Registry.ucm:

for control output variables (no need to change).

4.3 Source codes

./src/f90/ directory

module_params.f90 => *namelist.ucm*

module_vars.f90 => *declare variables*

module_io.f90 => *for in / output*

↓

module_setup.f90 => read *namelist.ucm*

↓

module_initialize.f90 => *initialize the variables*

↓

module_ra_driver.f90 => *calculate radiation*

module_ra_kondo94.f90

↓

module_pbl_driver.f90 => *calculate diffusion coefficient by using Mellor-Yamada scheme*

↓

module_sf_driver.f90 => *surface model*

module_sf_slab.f90

module_sf_sucm.f90

↓

module_dyn_driver.f90

module_dyn_urban.f90

↓

main.f90

4.4 Output file

Output file is in netcdf format

```
netcdf case1_mucmout_2014_08_01 {
dimensions:
        time = UNLIMITED ; // (576 currently)
        z_dim = 120 ;
        z_dim_stag = 121 ;
        surface = 6 ;
variables:
        float z_dim(z_dim) ;
        float time(time) ;
```

```
        time:units = "second since 2014-08-01 00:00:00" ;
        time:calendar = "gregorian" ;
        time:timestep = 2. ;
        time:output_interval = 600 ;
float TEMP(time, z_dim) ;
        TEMP:description = "potential temperature" ;
        TEMP:units = "C" ;
        TEMP:dimension = "kn" ;
float QV(time, z_dim) ;
        QV:description = "specific humidity" ;
        QV:units = "g/kg" ;
        QV:dimension = "kn" ;
float U(time, z_dim) ;
        U:description = "u wind" ;
        U:units = "m/s" ;
        U:dimension = "kn" ;
float V(time, z_dim) ;
        V:description = "v wind" ;
        V:units = "m/s" ;
        V:dimension = "kn" ;
float KM(time, z_dim) ;
        KM:description = "turbulence diff momentum" ;
        KM:units = "m2/s" ;
        KM:dimension = "kn" ;
float KH(time, z_dim) ;
        KH:description = "heat diff momentum" ;
        KH:units = "m2/s" ;
        KH:dimension = "kn" ;
float T2(time) ;
        T2:description = "Temperature at 2m" ;
        T2:units = "C" ;
        T2:dimension = "n" ;
float Q2(time) ;
        Q2:description = "Specific humidity at 2m" ;
        Q2:units = "g/kg" ;
        Q2:dimension = "n" ;
float U10(time) ;
        U10:description = "Wind x component at 10m" ;
        U10:units = "m/s" ;
        U10:dimension = "n" ;
float V10(time) ;
        V10:description = "Wind y component at 10m" ;
        V10:units = "m/s" ;
        V10:dimension = "n" ;
float ELEVATION(time) ;
```

```
        ELEVATION:description = "Solar elevation angle" ;
        ELEVATION:units = "rad" ;
        ELEVATION:dimension = "n" ;
float ADI(time) ;
        ADI:description = "Solar azimud angle" ;
        ADI:units = "rad" ;
        ADI:dimension = "n" ;
float COSADI(time) ;
        COSADI:description = "Cos" ;
        COSADI:units = "" ;
        COSADI:dimension = "n" ;
float SINADI(time) ;
        SINADI:description = "Sin" ;
        SINADI:units = "" ;
        SINADI:dimension = "n" ;
float SW(time) ;
        SW:description = "Short wave radiation flux" ;
        SW:units = "rad" ;
        SW:dimension = "n" ;
float SD(time) ;
        SD:description = "Short wave radiation flux" ;
        SD:units = "rad" ;
        SD:dimension = "n" ;
float SS(time) ;
        SS:description = "Short wave radiation flux" ;
        SS:units = "rad" ;
        SS:dimension = "n" ;
float LW(time) ;
        LW:description = "Short wave radiation flux" ;
        LW:units = "rad" ;
        LW:dimension = "n" ;
float BRiNu(time) ;
        BRiNu:description = "Bulk Richardson number" ;
        BRiNu:units = "-" ;
        BRiNu:dimension = "n" ;
float CM_SLB(time) ;
        CM_SLB:description = "extrance coefficient" ;
        CM_SLB:units = "-" ;
        CM_SLB:dimension = "n" ;
float CH_SLB(time) ;
        CH_SLB:description = "extrance coefficient heat" ;
        CH_SLB:units = "-" ;
        CH_SLB:dimension = "n" ;
float TAU_U_SLB(time) ;
        TAU_U_SLB:description = "tau u wind" ;
```

```
        TAU_U_SLB:units = "-" ;
        TAU_U_SLB:dimension = "n" ;
float TAU_V_SLB(time) ;
        TAU_V_SLB:description = "tau v wind" ;
        TAU_V_SLB:units = "-" ;
        TAU_V_SLB:dimension = "n" ;
float TAU_T_SLB(time) ;
        TAU_T_SLB:description = "tau heat" ;
        TAU_T_SLB:units = "-" ;
        TAU_T_SLB:dimension = "n" ;
float RNET_SLB(time) ;
        RNET_SLB:description = "net radiation" ;
        RNET_SLB:units = "W/m2" ;
        RNET_SLB:dimension = "n" ;
float H_SLB(time) ;
        H_SLB:description = "sensible heat flux" ;
        H_SLB:units = "W/m2" ;
        H_SLB:dimension = "n" ;
float LE_SLB(time) ;
        LE_SLB:description = "latent heat flux" ;
        LE_SLB:units = "W/m2" ;
        LE_SLB:dimension = "n" ;
float GR_SLB(time) ;
        GR_SLB:description = "ground flux" ;
        GR_SLB:units = "W/m2" ;
        GR_SLB:dimension = "n" ;
float TS_SLB(time) ;
        TS_SLB:description = "surface temperature" ;
        TS_SLB:units = "C" ;
        TS_SLB:dimension = "n" ;
float RNET_UCM(time, surface) ;
        RNET_UCM:description = "Surface net radiation" ;
        RNET_UCM:units = "W/m2" ;
        RNET_UCM:dimension = "sn" ;
float H_UCM(time, surface) ;
        H_UCM:description = "Surface sensible heat flux" ;
        H_UCM:units = "W/m2" ;
        H_UCM:dimension = "sn" ;
float LE_UCM(time, surface) ;
        LE_UCM:description = "Surface latent heat flux" ;
        LE_UCM:units = "W/m2" ;
        LE_UCM:dimension = "sn" ;
float GR_UCM(time, surface) ;
        GR_UCM:description = "Surface ground flux" ;
        GR_UCM:units = "W/m2" ;
```

```
        GR_UCM:dimension = "sn" ;
float TS_UCM(time, surface) ;
        TS_UCM:description = "Surface surface temperature" ;
        TS_UCM:units = "C" ;
        TS_UCM:dimension = "sn" ;
float CM_UCM(time, surface) ;
        CM_UCM:description = "Surface extrance coefficient momentum" ;
        CM_UCM:units = "-" ;
        CM_UCM:dimension = "sn" ;
float CH_UCM(time, surface) ;
        CH_UCM:description = "Surface extrance coefficient heat" ;
        CH_UCM:units = "-" ;
        CH_UCM:dimension = "sn" ;
float SD_UCM(time, surface) ;
        SD_UCM:description = "Wall direct radiation" ;
        SD_UCM:units = "W/m2" ;
        SD_UCM:dimension = "sn" ;
float Ri_UCM(time, surface) ;
        Ri_UCM:description = "Bulk Richardson number" ;
        Ri_UCM:units = "-" ;
        Ri_UCM:dimension = "sn" ;
float SHADE_UCM(time, surface) ;
        SHADE_UCM:description = "Surface shade rate" ;
        SHADE_UCM:units = "-" ;
        SHADE_UCM:dimension = "sn" ;
float SWNET(time, surface) ;
        SWNET:description = "Net short wave on surface" ;
        SWNET:units = "W/m2" ;
        SWNET:dimension = "sn" ;
float LWNET(time, surface) ;
        LWNET:description = "Net long wave on surface" ;
        LWNET:units = "W/m2" ;
        LWNET:dimension = "sn" ;
float LWDOWN(time, surface) ;
        LWDOWN:description = "Downward long wave on surface" ;
        LWDOWN:units = "W/m2" ;
        LWDOWN:dimension = "sn" ;
float LWUP(time, surface) ;
        LWUP:description = "Upward long wave on surface" ;
        LWUP:units = "W/m2" ;
        LWUP:dimension = "sn" ;
float TAU_U_UCM(time) ;
        TAU_U_UCM:description = "tau u wind" ;
        TAU_U_UCM:units = "-" ;
        TAU_U_UCM:dimension = "n" ;
```

```
float TAU_V_UCM(time) ;
        TAU_V_UCM:description = "tau v wind" ;
        TAU_V_UCM:units = "-" ;
        TAU_V_UCM:dimension = "n" ;
float TAU_T_UCM(time) ;
        TAU_T_UCM:description = "tau heat" ;
        TAU_T_UCM:units = "-" ;
        TAU_T_UCM:dimension = "n" ;
float THETA_T(time) ;
        THETA_T:description = "potential temperature at top" ;
        THETA_T:units = "K" ;
        THETA_T:dimension = "n" ;
float QV_T(time) ;
        QV_T:description = "specific humidity at top" ;
        QV_T:units = "kg/kg" ;
        QV_T:dimension = "n" ;
float U_T(time) ;
        U_T:description = "wind speed (x) at top" ;
        U_T:units = "m/s" ;
        U_T:dimension = "n" ;
float V_T(time) ;
        V_T:description = "wind speed (y) at top" ;
        V_T:units = "m/s" ;
        V_T:dimension = "n" ;
float SW_B(time) ;
        SW_B:description = "downward short-wave radiation" ;
        SW_B:units = "W/m2" ;
        SW_B:dimension = "n" ;
float LW_B(time) ;
        LW_B:description = "downward long-wave radiation" ;
        LW_B:units = "w/mw" ;
        LW_B:dimension = "n" ;
float COSZ(time) ;
        COSZ:description = "COS of SOLAR ZENITH ANGLE" ;
        COSZ:units = "-" ;
        COSZ:dimension = "n" ;
float OMG(time) ;
        OMG:description = "SOLAR HOUR ANGLE" ;
        OMG:units = "-" ;
        OMG:dimension = "n" ;
float DECLIN(time) ;
        DECLIN:description = "SOLAR DECLINATION" ;
        DECLIN:units = "-" ;
        DECLIN:dimension = "n" ;
float AZTH_C(time) ;
```

```
                AZTH_C:description = "Cosin of azimuth angle" ;
                AZTH_C:units = "-" ;
                AZTH_C:dimension = "n" ;
        float AZTH_S(time) ;
                AZTH_S:description = "Sin of azimud angle" ;
                AZTH_S:units = "-" ;
                AZTH_S:dimension = "n" ;
        float SOL_ELEV(time) ;
                SOL_ELEV:description = "Solar elevation angle" ;
                SOL_ELEV:units = "rad" ;
                SOL_ELEV:dimension = "n" ;

// global attributes:
                :MUCM = "Developed by Doan Quang Van" ;
                :ra_sw_physics = "Kondo94" ;
                :ra_lw_physics = "Kondo94" ;
                :bl_pbl_physics = "MY_lev2" ;
                :sf_surface_physics = "multilayer UCM" ;
}
```

4.6