User manual

Ventspils RTMC (Radiative Transfer Monte Carlo)

Program authors:

Juris Freimanis

Romāns Peženkovs

Engineering Research Institute

“Ventspils International Radio Astronomy Centre”

of Ventspils University of Applied Sciences

E-mails: jurisf@venta.lv , romans.pezenkovs@venta.lv

The code described here has been developed by Juris Freimanis and Romāns Peženkovs at the Engineering Research Institute "Ventspils International Radio Astronomy Centre" (ERI VIRAC) of Ventspils University of Applied Sciences (VUAS) as part of the European Regional Development Fund (ERDF) project No. 1.1.1.1/16/A/213 "Physical and chemical processes in the interstellar medium" ("ASTRA").

Program using guide

In this manual we describe how to use Ventspils RTMC and most important functions of it. The result of our program is CCD matrix images, each pixel of this matrix consists four Stokes vector parameters (I,Q,U,V). The CCD matrix images could be compared with results of other program (for example Integral_equ [Integral equations for source functions in homogeneous sphere. Rayleigh scattering] also created by J.Freimanis and R.Peženkovs).

1. Ventspils RTMC parallelization and compilation

Venspils RTMC Program was written for Linux computers and supercomputers (we have tested it on PC [6-th gen Intel Core i7-6785R, 16GB RAM] and on one node of VIRAC HPC cluster [Intel Xeon Gold 6252, 24 cores; 384 GB RAM per node]). The **C++11** standard libraries **<thread>** and **<mutex>** were used for program parallelization. The thread that simulates the light scattering of one photon package is created for each processor core, when package was scattered another package is created. For instance, if processor has 24 cores, then 24 threads are created and 24 photon packages are scattered simultaneously. The <mutex> library is used to avoid collision, when scattered photon package's data is added to CCD matrix. We use following command to compile our program:

g++ -std=c++11 –o prog.out –pthread *.cpp

It is essential to connect to cluster, then copy all program files, then compile program and run it using **nohup** . If you would not use **nohup** , then you need to be connected to cluster all time when program runs and can't close the connection, but by using **nohup** you can close connection. Use command:

nohup ./prog.out &

Now you can close connection. If you would like to check program's progress use command:

tail –f nohup.out

2. Creation of stars

Use array of pointers **star_coordinates** to create stars. Each array element is a pointer to structure, that consist **float radius** – radius of the star (1 equals to 0,5128 AU), **float cloud_radius** - radius where temperature is too high and dust particles do not condensate, **float x_pos** , **float y_pos** , **float z_pos** – x,y,z coordinates of star. The constant **star_quantity** is defined in file **dust_cloud.h** and it is equal to 10, but user can change it. The pointers that are not defined should be equal to **NULL** .

3. Creation of dust cloud

Use class **Dust_cloud** to create shape of dust cloud and set dust concentration. It is essential to choose size of three-dimensional array used to split dust cloud in sub-volumes. To set the size of three-dimensional array the constants **ARRAY_X**, **ARRAY_Z**, **ARRAY_Z** are defined, default value of all these constants is "80", but user can change the values of these constants. To use **Dust_cloud** class the object should be created, and constructor's parameter **cloud_Type** should be set (use "1" for spherical cloud, "2" for asymmetrical cloud). It is necessary to create the object dynamically, so after usage it could be deleted to save computer's memory.

It is essential to set minimum and maximum azimuth and polar angles (in radians) using function:

void set_min_max_phi_thet(double min_Phi, double min_Thet,

double max_Phi, double max_Thet) .

These angles are used in parametric equations that are used to create surface of dust cloud. For sphere **min_Phi** = 0, **min_Thet** = 0, **max_Phi** = 2*PI, **max_Thet** = PI.

Function **void set_outer_rad(double out_rad)** is used to set **out_rad** - outer radius of spherical cloud. The parameter **out_rad** should be less than half of array's constant, for example, if **ARRAY_X** = 80, **ARRAY_Y** = 80; **ARRAY_Z** = 80 the **out_rad** should be less than "40".

Function **void make_cloud_surface()** is used to create the outer surface of dust cloud. Function **void cut_off_sphere()** is used to cut off spheres from dust cloud and create the inner surface of dust cloud. The stars are inside these spheres and temperature is too high so dust particles do not condensate. Function **void fill_cloud()** is used to fill the dust cloud volume between outer and inner surfaces.

After creating and initializing three-dimensional array of short variables using class **Dust_cloud**, the array of pointers to **Sub_Volume** objects with the same size should be created. The elements of three-dimensional array that are outer, inner surfaces or volume between surfaces should point to **Sub_Volume** objects, others pointers are NULL. The array of pointers imitates the dust cloud, every **Sub_Volume** object consists of optical features like average extinction cross section, average scattering cross section. The extinction should be set using function **void set_extinction(float extinction)**, the scattering should be set using function **void set_scattering(float scattering)**, the concentration should be set using function **void set_concentration(short number)**.

One important variable of **Sub_Volume** object is **cloud_surface** that defines the object as a part of the dust cloud. If **cloud_surface** = 1, then **Sub_Volume** object is the outer surface of the cloud; if **cloud_surface** = 2, then the object is volume between surfaces; if **cloud_surface** is bigger than 3 or equal to 3, then the object is the inner surface. In this case, the value of **cloud_surface** is equal to number of star plus 2; for example if we have only one star, then **cloud_surface** = 1 + 2 = 3 . The variable **cloud_surface** is used to calculate photon package position in dust cloud.

4. Setting number of photon packages and number of threads

To set number of photon packages which will be scattered by program use variable **int photon_pack_q** . Maximum value of threads is 32, but user can change this value (**thread thread_array[32]**). To check how many processor cores are available use function **thread::hardware_concurrency()**. If core number is bigger than 32 (**thread thread_array[32]**) only one thread will be created. If core number is less or equal to 32 then program creates one thread for each core. For instance, if processor has 24 cores, then 24 threads will be created. Function

void photon_packages_traveling(int thread_number, Sub_Volume
***arr_Vol[ARRAY_X][ARRAY_Y][ARRAY_Z], CCD_matr *matrix_obj,**
star_coordinates *star_ar[star_quantity], short outer_Rad,
int photon_packet_q, unsigned int number_thre)

is called for each thread so photon emission, traveling, scattering and adding data to CCD matrix are modeled in this function.

5. CCD matrix images

Essential to accumulate all data of each photon package scattering, it is done using class **CCD_matr**. Each photon package scattering is accumulated in two-dimensional array using "peeling-off" method. The dimensions of image are set in private array of structures **one_cell matrix[(int) (ARRAY_X * CCD_PIXELS)][(int) (ARRAY_Y * CCD_PIXELS)]** of class **CCD_matr** . Each element of this array consists five **long double** variables: **photon_weight** – photon "weight" accumulated by this pixel, **I_stokes** – (I) first Stokes parameter accumulated by pixel, **q_stokes** – Q stokes parameter divided by I, **u_stokes** – U Stokes parameter divided by I, **v_stokes** – V Stokes parameter divided by I. The dimensions of CCD matrix image is set using constants **ARRAY_X** , **ARRAY_Y** and **CCD_PIXELS** . User can change these dimensions. The size of CCD matrix image is 200x200 pixels **ARRAY_X * CCD_PIXELS** = 80*2,5 = 200 and **ARRAY_Y * CCD_PIXELS** = 80*2,5 = 200.

CCD_matr object ***matr_pointer** should be created dynamically and it is input parameter of function **photon_packages_traveling** . After scattering all photon packages, the results are

saved in four files **data_gnu_I.txt** , **data_gnu_Q.txt** , **data_gnu_U.txt** , **data_gnu_V.txt** Each file consists two-dimensional array of Stokes parameter. For instance, file **data_gnu_I.txt** consists 200x200 array of first Stokes parameter (I) for each pixel. File **data_gnu_Q.txt** consists 200x200 array of second Stokes parameter (Q). File **data_gnu_U.txt** consists array of third Stokes parameter (U) and file **data_gnu_V.txt** consists array of fourth Stokes parameter (V).

To create image, the **python** program **p03_Ventspils.py** is used.

6. Light source parameters

To create light source use class **Photon_source** . You should set Stokes vector parameters of emitted light using functions: **void set_stokes_I_param(double I_par)** for first Stokes vector parameter; **void set_stokes_Q_param(double Q_par)** for second Stokes vector parameter; **void set_stokes_U_param(double U_par)** for third Stokes vector parameter; and **void set_stokes_V_param(double V_par)** for fourth Stokes vector parameter. It is essential to set photon quantity in emitted photon package (photon weight), use function
void set_phot_weight(double phot_weight) .

Function **void initialize_photon_source(star_coordinates *star_coord)** is used to set star coordinates and radius for light source. Stars are treated as spherical light sources. It is necessary to calculate angles of distribution of photon package, use function
void gener_emit_angles_point() for this purpose.

To emit photon package use function **emit_phot_param emit_phot()** . The output of this function is structure **emit_phot_param** , which consists **double s_PHI_1** - azimuth angle of emitted photon package; **double s_THET_1** polar angle of emitted photon package;
double stokes_I – Stokes vector I parameter; **double stokes_Q** – Stokes vector Q parameter; **double stokes_U** – Stokes vector U parameter; **double stokes_V** – Stokes vector V parameter; **double phot_weight** – weight of photon package; **double phot_pack_start_X** – the starting X coordinate of photon package; **double phot_pack_start_Y** – the starting Y coordinate ;
double phot_pack_start_Z – the starting Z coordinate.

7. Telescope parameters and physical dimensions of cloud

Class Telescope is used to set telescope parameters and physical dimensions of model. Use function **void set_cloud_rad_AU(double astron_units)** to set physical size of dust cloud (in Astronomical Units). Use function **void set_dist_cloud_tel_PC(double parsecs)** to set distance between dust cloud and telescope (in parsecs). Use function **void set_telescope_rad_meters(double meters)** to set radius of telescope (in meters). Use function **void calc_telescope_area()** to calculate area of telescope. It is essential to use function **void calc_solid_angle()** to calculate solid angle of telescope.

You need to set position of telescope (in spherical coordinates); use function **void set_tel_polar_angle(double thet_angle)** to set polar angle of telescope; use function **void set_tel_azimuth_angle(double phi_angle)** to set azimuth angle of telescope. It is essential to set distance between telescope coordinate system and cloud coordinate system in Descartes coordinates (1 = 0,5128 AU), for example if distance is equal to 600 pc, then it is $2{,}413*10^8$ in Descartes coordinates, use function **void set_tel_dist_cloud_sys_tel_sys(double distance)** for this purpose.

Use function **void calc_parameters_for_tel_param()** to calculate parameters of telescope's plane and for cloud projection to telescope's plane. Use function **tel_param get_tel_param()** to initialize photon package with telescope's parameters.

8. Optical parameters of dust

Class Angles_montecarlo should be initialized with file **F11F22F33F44F12F34** which contains dust particle optical parameters (see **Michael I Mishchenko, Larry D Travis, and Andrew A Lacis. Scattering, absorption, and emission of light by small particles. Cambridge university press, 2002**) . Scattering, absorption and emission of light are strongly dependent of dust particle parameters such as size, shape, refractive index and orientation of dust grain. Computerized theoretical methods of dust optical properties calculation are based on T-matrix method.

Codes for computing electromagnetic scattering by nonspherical particles were created by M.I.Mishchenko and colleagues using FORTRAN language. The programs are available at

https://www.giss.nasa.gov/staff/mmishchenko/t_matrix.html . Those programs could be used to create the base of optical parameters of dust particles.

Step of optical parameters of dust in file **F11F22F33F44F12F34** is 0,25°.

9. Travel of photon package and scattering

The class **Photon_packet** is used to simulate photon package scattering and traveling inside dust cloud. The **Photon_packet** objects are created dynamically; when **Photon_packet** object is out of cloud or the photon weight is less than one photon, then the object is deleted, and the next **Photon_packet** object will be created. The dynamic way of object creation is used for rational use of computer memory. You should initialize photon package with star parameters using functions **void initialize_struct_star_p(star_coordinates *star_p[star_quantity])** and **void set_star_numb(short star_n)**. It is essential to initialize the photon package with the parameters generated by **Photon_source** object; the initialization is done using function **void initialize_packet(emit_phot_param pho_struct)** . You should set the telescope parameters using function **void set_tel_parameters(tel_param tel_p)** .

After **Photon_packet** object initialization the parameters of dust cloud should be set. The function **void set_cloud_type(short cloud_typ)** is used to set variable **cloud_type** that describes the shape of dust; if **cloud_type** = 1 - the cloud is sphere; If **cloud_type** = 2 - the cloud is asymmetrical. The function **void set_concentration_type(short concent_type)** is used to set the dust cloud particle concentration type if **concentration_type** = 1 – the concentration is constant and cloud is homogeneous; else if **concentration_type** = 2 – the concentration is inversely proportional to the distance from dust cloud center to photon package, cloud is inhomogeneous; else if **concentration_type** = 3 – the concentration is inversely proportional to the square of distance from dust cloud center to photon package, cloud is inhomogeneous.

It is essential to set CCD matrix pointer using function **void set_CCD_matr_point(CCD_matr *matr_p)** . You should set geometrical parameters of dust cloud using function **void set_outer_inner_radius(short outer_Rad, short inner_Rad)** . Calculate maximum optical way using function

float max_optical_way_calc(Sub_Volume *arr[ARRAY_X][ARRAY_Y][ARRAY_Z]) . Use function to generate optical way for first photon package scattering using function **float gener_first_optical_way()** . Calculate geometrical way of photon package using function **double calc_geometrical_way(Sub_Volume *arr[ARRAY_X][ARRAY_Y][ARRAY_Z])** . It is essential to check state of photon package using function **short get_photon_location()** , this function gets variable that describes state of photon package 0 – after emission, 1 – in cloud, 3 – out of cloud, 4 – weight of photon package is less than one photon.

It is essential to calculate spherical coordinates (for cloud projection on CCD matrix) using function **void calc_spher_coor_cur_tel()** . Use function **void calc_Thet_peel()** to calculate scattering angle in telescope's direction. It is necessary to get angles for "peeling-off" method, use function **angles_for_peel get_angles_for_peel()** for this purpose.

It is essential to get normalized Stokes vector parameters (Q/I; U/I; V/I) of photon package, use functions **double get_stokes_q_param()** ; **double get_stokes_u_param()** ; **double get_stokes_v_param()** . Use functions **void set_stokes_q_tel(double q_tel)** ; **void set_stokes_u_tel(double u_tel)** ; **void set_stokes_v_tel(double v_tel)** to set normalized Stokes vector parameters.

It is necessary to set F11 and F12 parameters for "peeling-off" method, use functions **void set_F11_value(float F11_val)** and **void set_F12_value(float F12_val)** for this purpose. Use functions **void set_cos_2psi_tel(float cos_2psi_t)** and **void set_sin_2psi_tel(float sin_2psi_t)** to set cosine and sine of angle 2*psi for "peeling-off" method. Use function **float optical_way_peel_calc(Sub_Volume *arr[ARRAY_X][ARRAY_Y][ARRAY_Z])** to calculate the optical way from scattering point till end of cloud in telescope direction (for "peeling-off" method).

Use function **void first_scatter_phot_pack(double albedo)** for modeling the first scattering of photon package. You should set state of photon package using function **void set_photon_location(short location)** ; 0 – after emission, 1 – in cloud, 3 – out of cloud, 4 – weight of photon package is less than one photon.

It is essential to get and set propagation angles of direction of photon package (polar and azimuth angles), use functions **double get_THET_1()** ; **double get_PHI_1()** ;

void set_THET_1(double s_THET_1) and **void set_PHI_1(double s_PHI_1)** for this purpose.

You should use function **float gener_optical_way()** to generate optical way for photon package.

If state of photon package is “3” – out of cloud or “4” – weight of photon package is less than one photon, then photon package must be deleted and another package should be created.

10. Angles calculation for photon package

Class **Angles_montecarlo** is used to calculate photon package travelling angles after each scattering and also to calculate Stokes vector parameters.

It is essential to use function **void set_angles_for_peel(angles_for_peel angles_for_p)** for “peeling-off” method, to set scattering angle in telescope direction, polar and azimuth angles in telescope direction, polar and azimuth angles of incident photon package. Use function void **set_iinc_qinc_uinc_vinc(double iinc, double qinc, double uinc, double vinc)** to set normalized Stokes vector parameters (I/I; Q/I; U/I; V/I). It is necessary to use function **void calc_stokes_tel_par()** to calculate normalized Stokes vector parameters for telescope.

Use functions **double get_q_tel()** ; **double get_u_tel()** ; **double get_v_tel()** to get calculated normalized Stokes vector parameters for telescope. Use functions **float get_F11_val()** and **float get_F12_val()** to get **F11** and **F12** values for “peeling-off” method. It is essential to use functions **double calc_cos_2psi_tel()** and **double calc_sin_2psi_tel()** to calculate cosine and sine of 2*psi angle (for “peeling-off” method).

It is necessary to use function **double calc_bigThet()** to calculate scattering angle. Use function **double bisection(double Epsilon)** to solve transcendental equation to calculate angle psi (for photon package travelling). It is essential to set polar and azimuth angles of incident photon package, use function **void set_THET_1_PHI_1(double thet_1, double phi_1)** for this purpose. Use functions **double calc_THET_2()** and **double calc_PHI_2()** to calculate photon package travelling angles (polar and azimuth angles).

It is necessary to calculate normalized Stokes parameters for scattered photon package, use functions **void calc_stokes_scat_par()** for this purpose. It is essential to use functions

double get_q_sca() ; **double get_u_sca()** and **double get_v_sca()** to set normalized Stokes vector parameters for scattered photon package.