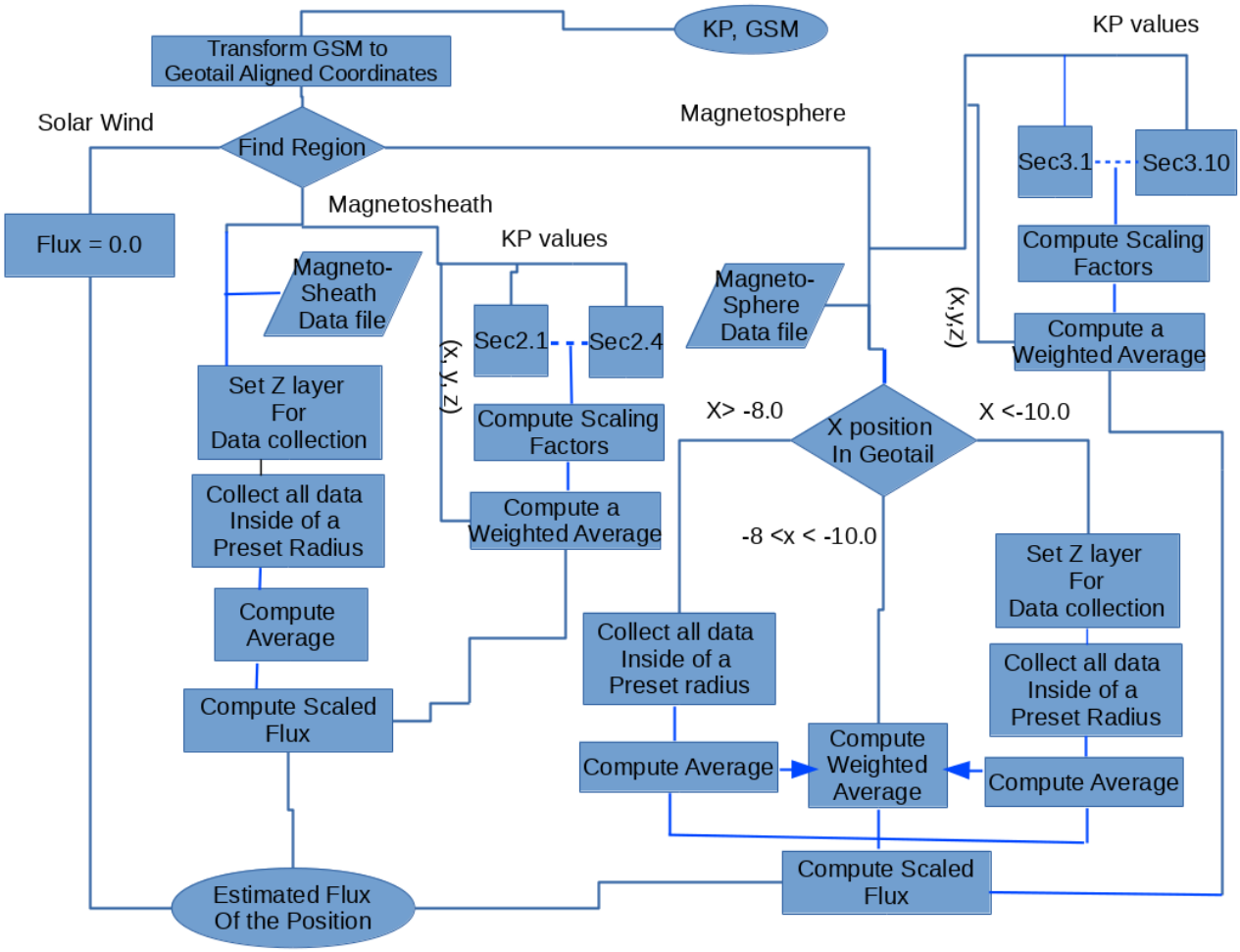
The crmflx code estimates an ion flux value at a satellite position for a given magnetic activity KP index based on the preset ion flax maps. The code was originally written in early 2000 in a FORTRAN. Since no one was maintaining it, the MTA group decided to re-write it in Python(3.6) so that it is easier to maintain in the future.

CRM Flux Computation Steps



1. Read solar wind, magnetosheath, and magnetosphere ion flux models from data files.

Each data file contains:

* cell positions in x, y, z between -30 and 30 with an increment of one Earth radius at the center of the Earth (0, 0, 0).
* average flux value of each cell of different KP values between 1 and 9.
* numbers of non-zero flux values used to compute the average flux in each cell of different KP values between 1 and 9.

Notes:
* It seems that the model data tables were created with Tsyganenko's geopack.f, but we cannot confirm that. Since the geopack.f and its parameter files were updated recently; probably, our data files are out of date.
* We do not use the flux data in the solar wind region (the outside of the bowshock), and the flux values are set to zero in this script.

2. Read a file contains the locations of CXO in the GSM coordinates. (PE.EPH.gsme_in_Re_short from EPHEM directory) and set 28 KP values.

For each satellite position and a KP value:

3. Find where the satellite is located regarding the geotail.
* use GSM coordinates and a KP value to find in which region (solar wind/magnetosheath/magnetosphere) the satellite is located.
* the coordinates are transformed into the system aligned with the geotail.

4. If the satellite is in the magnetosheath,
a. Compute a scaling factor
* divide the region into four sectors
* compute a scaling factor of each sector depending on the KP value given
* find the distance to each sector from the satellite position
* order them from the nearest to the furthest
* compute a distance weighted scaling factor based on the scaling factors from all sectors

b. Determine how far the flux cell data should be included in the z-direction

c. Compute an average flux
* from magnetosheath database, collect the flux cell data around the satellite
* order the cell from the nearest to the furthest
* add them up from the closest until either the distance to the data cell reaches the threshold or the maximum number of the cells collected.
* take an average

d. Compute the scaled flux
* multiply the scaling factor to the averaged flux value to estimate the final flux.

5. If the satellite is in the magnetosphere,
a. Compute a scaling factor
* divide the region into ten sectors
* compute a scaling factor of each sector depending on the KP value given
* find the distance to each sector from the satellite position
* order them from the nearest to the furthest
* compute a distance weighted scaling factor based on the scaling factors from all sectors

b. Read data and set a cell region
* from magnetosphere database, collect the flux cell data around the satellite

* if x coordinate in the geotail is in the head side or less than 8 Earth radius in the tail side, use all available fluxes.

* if it is beyond 10 radii in the tail side, use data cells in the limited layer in the z-direction.

* if the distance is between 8 and 10 radii in the tail side, both regions above are used, and the weighted average is computed.

c. Compute an average flux

* order the cell from the nearest to the furthest

* add them up from the closest until either the distance to the data cell reaches the threshold or the maximum numbers of the cell collected.

* take an average

* if the satellite is between 8 and 10 radii in the tail side, make a distance weighted average of fluxes computed in the two regions

d. Compute the scaled flux

* multiply the scaling factor to the averaged flux value to estimate the final flux.

6. The computation is repeated for 28 different KP values in 2000 different satellite positions.

How to Speed Up the Python Script

The Python script, which was directly translated from the FORTRAN code, took nearly sixty minutes to compute the entire CRM data set. To shorten the computation time, the following changes are made.

* When possible/useful, NumPy arrays are used which assign the memory space and speed up the computation.

* The original FORTRAN code is designed to take many options. I streamlined the code and left only essential for our needs. This reduced many condition statements and loops, which slow down the computation.

* The model data in the data files were mostly null data. I removed them from the data file, and let this script reads only valid data. This reduced the computation time slightly.

The above modifications did not speed up much; however, the following two steps significantly improved the computation time.

* The code is compiled with Cython. The computation time is reduced to fifteen minutes.

    Ref: https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html

* Use of "typed memoryview." The script spends the majority of time in two functions. By assigning typed memoryviews to the variables in these two functions, the computation time is reduced to three minutes.

    Ref: https://cython.readthedocs.io/en/latest/src/userguide/memoryviews.html#memoryviews

* I tried Typed memoryviews in the other functions, but it did not reduce the computation time any farther, or, in other instances, it increased the computation time.

The Flux Discrepancy in the Magnetosphere between the FORTRAN Code and the Python Code

When computing the fluxes in the magnetosphere, there are some discrepancies between values calculated by the FORTRAN code and the Python code. It is because the FORTRAN code often misses collecting the flux values in the outer area.

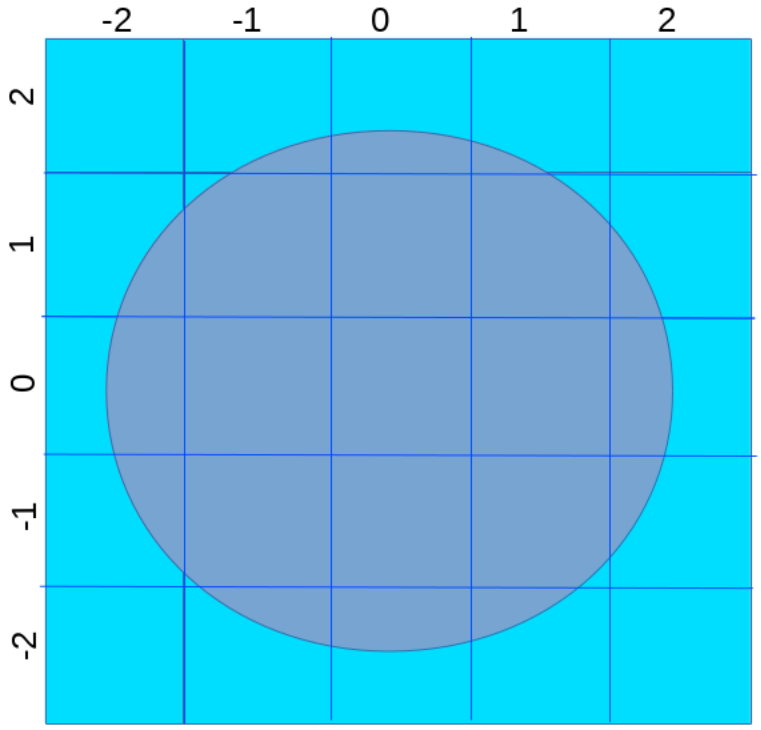
The algorithm of the FORTRAN code to compute the flux is the following.

1. In the flux map of n by n by n 3D binned space, locate the satellite position.
2. Start accumulating the flux from the nearest bin to the satellite position.
3. Increment step 1 in one of the three directions and find the distance to the cell.
4. Check the distance to the cell is inside of the predetermined distance, add the flux.
5. If the distance is the outside of the predetermined distance, stop.

The problem is step 5. Please see the figure below. In this 2D surface, the satellite is located at the bin (0, 0) and supposed to accumulate all flux values in the bins covered by the gray circle (predetermined distance from the center).

The algorithm check (0, 1), (0, -1), (1, 0), (-1, 0), since they are the same distance, then move to check (0, 2), (0,-2) etc. However, once it reaches the -2 region, the problem happens. Because of binning, the code may check (0, -2), (-1, -2), (1, -2), and then (-2, -2), which is outside of the predetermined distance. The FORTRAN code stops accumulating the fluxes at that point, even though some other bins (e.g.,(0, 2), (-1,2), etc.) are still not checked.

The Python version fixed this deficiency and covers all bins under the predetermined distance. Because of this, the flux computed by the Python version is often larger than that by the FORTRAN code, but it is not always the case. The reason is that the final estimated flux is an average of all bins. If the extra bins collected by the Python code contain minimal values, the averaged value could be smaller than the averaged value did not include these small value bins.



The Script Details

runcrmf.py:
==========
This is a control Python code. It reads the model data files, EPHEM data in GSM coordinates, and sets up KP values.
It calls swinit, mshinit, and mspinit (from crmflx.py), and run the main function crmflx (from crmflx.py)

crmflx.pyx
==========

crmflx:
----
the primary function to calculates the ion flux as a function of the magnetic activity kp index.
main inputs: (X, Y, Z) GSM coordinates of the satellite
KP index
Magnetsheath data table
Magnetosphere data table
Data table binning map

output: ion flux at the GSM coordinates

function used: locreg --- identify which region the satellite is in
scalkp2 --- collect data from sections in the magnetosheath
nbrflux --- compute the ion flux in the magnetosheath
scalkp3 --- collect data from sections in the magnetosphere
nbrflux_map_z --- compute the ion flux in the magnetosphere

mspinit:
----
read magnetosphere database
mshinit:
----
read magnetosheath database
swinit:
----
read solar wind database

output: cell positions in x, y, z between -30 and 30 with an increment of one Earth radius at the center of the Earth as (0, 0, 0).
average flux value of each cell of different KP values between 1 and 9.
numbers of non-zero flux values used to compute the average flux in each cell of different KP values between 1 and 9.

function used: read_init_data_file --- read the data

mapsphere:
---
finds the (i,j,k) index offset values used to define the search volume for the near-neighbor flux

input: step and step sizes
output: sets of offset in x, y, z coordinates

logreg:
---
determines which phenomenological region the spacecraft is in

input: KP value
GSM coordinates of the satellite

output: id to indicate in which region the satellite is in
x, y, z coordinates in the geotail system.

function used: solwnd --- set solar wind parameters
rog8ang --- rotate coordinates
locate --- defines the position of a point at the model magnetopause
bowshk2 --- give the bow shock radius, at a given x

solwind:
---
set solar wind parameters

input: KP value
output: a set of solar wind parameters for the given KP value

bowshk2:
---
give the bow shock radius, at a given x

input: a set of solar wind parameters given by "solwind" function
output: cylindrical radius of the bow shock

function used: fast --- local fast magnetosonic speed

fast:
---
local fast magnetosonic speed

input: a set of solar wind parameters
output: local fast magnetosonic speed

locate:
---
defines the position of a point at the model magnetopause

input: gsm coordinates
       solar wind proton density/ram pressure/velocity

output: coordinates of a point at the magnetopause

function used: compute_rng --- compute a distance between two points

nbrflux:
---
provides the region's ion flux as a function of k

input: sector information
          geotail coordinates
          data tables
output: estimated flux

function used: neighbr --- the nearest neighbors to the point
              wtscal --- compute weighted scaling factors
              zbinner --- find which z layer we should collect data
              flxdat1 --- finds the flux corresponding to the satellite's position

flxdat1
---
finds the flux corresponding to the satellite's position (magnetosheath)

input: satellite coordinates
          data table
          range in the z-direction of data collection
output: flux

function used: comput_rng --- compute a distance between two points

nbrflux_map_z:
---
provides the region's ion flux as a function of k
this function is used magnetosphere computation and requires distinguishing where in the geotail the satellite is located. See the algorithm page for more details.

input: sector information
          geotail coordinates
          data tables
output: estimated flux

function used: neighbr   --- the nearest neighbors to the point
              wtscal   --- compute weighted scaling factors
              zbinner   --- find which z layer we should collect data
              flxdat1_map --- finds the flux corresponding to the satellite's position

flxdat1_map:

---
finds the flux corresponding to the satellite's position (magnetosphere)

input: satellite coordinates
data table
range in the z-direction of data collection
output: flux

function used: comput_rng --- compute a distance between two points

compute_rng:
compute a distance between two points

input: two coordinates
output: the distance between two coordinate

neighbr:
---
finds the nearest neighbors from the given location to the sector location

input: satellite x, y coordinates
arrays of sector locations

output: sorted ranking of distance to each sector

rot8ang:
---
rotates the 2-d vector about its hinge point in the XY-plane

input: angle, x, y, and x value of aberration hinge point

output: computed x, y values

scalkp1:
---
compute scaling parameters in the solar wind region
scalkp2:
---
compute scaling parameters in the magnetosheath region
scalkp3:
---
compute scaling parameters in the magnetosphere region

input: KP value
output: scaling factors

function used: get_scalkp --- compute the scaling factor

get_scalkp:
---
compute the scaling factor

input: KP value
a list of section functions (see sectr** below)

output: scaling factors

sectr11 --- sectr13:
---
provide the proton flux against kp scaling for the solar wind region
input: KP value
output: scaling factor

function used: sect_comp --- compute scaling factor for the given sector

sectr21 --- sectr24:
---
provide the proton flux against kp scaling for the magnetosheath region
input: KP value
output: scaling factor

function used: sect_comp --- compute scaling factor for the given sector

sectr31 --- sectr310:
---
provide the proton flux against kp scaling for the magnetosphere region
input: KP value
output: scaling factor

function used: sect_comp --- compute scaling factor for the given sector

sect_comp:
---
compute scaling factor for given parameters

input: KP values
a parameter list
output: scaling factors

wtscal:
---
calculates the distance weighted sum of the Kp scaling factors

input: sector information including flux scale factor
distances to each sector

output: distance weighted scaling factor

function used: create_weighted_sum --- calculate a weighted sum

create_weighted_sum:
---
calculate a weighted sum

y_interpolate:
---
interpolate the coordinates between two points
input: coordinates of two points
x coordinates of the third point
output: estimated y coordinate

zbinner:
---
determine the z-layer of the magnetosphere

input: x and z gsm coordinates
output: bottom and top z coordinates