

# Using Python as Platform for a Unified Geophysical Data Processing Tool

Stanley Engle, Matt Funk, Theodore W. Sammis

## Introduction

Python is an interpreted object-oriented programming language developed in the 1990's and is a free language comparable to Java and Perl (Lutz, 1996). Information about Python and the installation packages of the Python interpreter can be found at <http://www.python.org>, which is maintained by the Python programming community and contains links to projects and packages using or extending Python. The Python interpreter runs on Windows, Linux/Unix, and Mac OS X. Further, it has been ported to the Java and .NET virtual machines. Python has a small number of built in data types (numerical types, strings, lists, dictionaries, and class-objects), but it can be extended by adding new modules implemented in either "Pure" Python or a compiled package written in C or C++ which can define new functions and variables as well as new object types.

Originally, Python was used as a scripting front end to support computation code written in a lower-level language such as C or Fortran, but in the last eight years, the language has begun to be used as the primary language for large scientific applications, including database applications, user interfaces, and numerical computation (Varoquaux et al., 2009). There are several reasons for this shift. Python provides greater platform transparency, where almost all "pure" Python programs will run on Windows, Linux/Unix and Mac OS X in exactly the same fashion. Second, Python allows the use of human-efficient coding techniques, saving development time. Further, Python provides strong support for building, testing and deploying large-scale applications using a mixture of Python, Fortran, C, and C++ code (Wilbers et al., 2009). Finally, most performance issues in running calculations of data sets have been addressed. The Numpy package, <http://numpy.scipy.org>, is available for all major operating system platforms, providing an extension to Python that greatly speeds up calculations to be on par with Matlab and to be 51 times faster than "pure" Python code (Linke, 2011).

In the remote sensing area, the simple syntax of Python allows construction of environmental models. Karssenberget al. (2007) added a PCRaster Python library to the Python code in order to construct a spatio-temporal soil moisture model, integrating a soil-water balance model and a remote sensing based energy flux. The Surface Energy Balance System (SEBS) estimates turbulent heat fluxes and evaporative fraction using data derived from remote sensing meteorological parameters (albedo, emissivity, temperature, fractional vegetation cover, Leaf Area Index, height of vegetation/roughness) at reference height (air pressure, temperature, relative humidity, wind speed) and radiation data (downward solar radiation, downward long-wave radiation). The first Python program implementing the algorithm was done using the PCRaster library and used remote sensing data from Landsat for the Rabat region in Morocco (van der Kwast et al., 2004). This system was not executed automatically using Windows task schedulers or Unix cronjobs and did not have a web interface.

Python, again, provides the developer with several non-computational modules that can be used to implement a web interface to scientific applications. Thus, programming a scientific application in Python makes sense because an acceptable computation speed is enhanced by a vast number of modules available to the Python developer. These modules encourage rapid development of large applications and the deployment of such applications in a web environment.

## Objective

The overall objective of the research was to construct a Python-based tool that allows for easy implementation of different algorithms for automated calculations of dust concentrations and evapotranspiration on a 1km grid using satellite and weather data. More specifically, the tool would be developed using object-oriented design paradigms so that the downloading and processing of raw data and calculating results would be functionally separate. Further, the use of Python as the programming language and the object-oriented design would allow for rapid development of new functionality.

## Materials

Python is usually available in most popular Linux distributions (Ubuntu, Fedora, openSUSE, etc.) by default. The Windows operating system does not include Python; therefore, the Python install program for Windows must be downloaded from the Python website. The RSETPy application uses the 64-bit Python 2.7 interpreter, located at <http://python.org/ftp/python/2.7.2/python-2.7.2.amd64.msi>. The standard installation of Python includes an FTP client; therefore, no third-party package that supplies FTP functionality is needed. RSETPy requires several third-party Python packages to fully implement the objectives of the research project. Table 1 describes the required packages, and links to the packages for Windows Operating Systems can be found at the web site <http://www.lfd.uci.edu/~gohlke/pythonlibs/>.

*Table 1. Description of Packages used in RSETPy.*

Package	Description
Base-11.8.27	Base distribution of many smaller basic or pure Python packages and tools.
configobj-4.7.2	RSETPy makes extensive use of text configuration files. This library provides functionality to read the configuration file while being very simple to use from a programmatic standpoint.
numpy-MKL-1.6.1	RSETPy uses the numpy package for fast computation of algorithmic calculations.
GDAL-1.8.1	This package allows RSETPy to read GRIB format files.
h5py-2.0.0	MODIS data is stored in HDF format. This package allows RSETPy to read MODIS data files.
MySQL-python-1.2.3	This package provides the Python interface to a MySQL database.
Shapely-1.2.13	This package is used for manipulation and analysis of planar geometric objects.
matplotlib-1.0.1	This package is used to plot time-series graphs
scipy-0.9.0	RSETPy uses the scipy package for fast computation of algorithmic calculations.
PIL-1.1.7	PIL is used to add image processing capabilities to RSETPy
basemap-1.0.1	RSETPy uses this package to support plotting data to a map.
natgrid-0.2.1	This package is used to interpolate 2 dimensional data.

RSETPy stores information in a MySQL database, which is accessed using the MySQL-python-1.2.3 package. Therefore, it is necessary to install MySQL on the computer: <http://www.mysql.com/downloads/mysql/>. For the development of RSETPy Bazaar (<http://bazaar.canonical.com/en/>), a version control system was used, and consequently installed on the development computer.

The development computer for RSETPy consists of a quad-core 3.4 GHZ AMD Phenom II processor with 12 GB of memory and a one Terabyte hard drive. In order to utilize the 12 GB of memory, the 64-bit version of Windows 7 was installed.

RSETPy is deployed on a production server, which is a 64-bit Windows 2008 R2 Enterprise server with 4 GB of memory and a dual-core Intel Core2Duo processor running at 2.67 GHZ. The deployment server contains seven 2 TB hard drives, where each hard drives stores data for one year, from 2005 to 2011. A back-up scheme, such as a RAID implementation, is not implemented. If a data drive fails, the drive is replaced and data is re-downloaded from source.

## Design

The structure of RSETPy can be thought of as a collection of execution layers, where each layer is capable of being run independent of the other layers. Each layer has a specific Python module that is responsible for governing the execution of that layer and all the layers are governed by a main module called RSETProcess. Table 2 describes what each layer is responsible for, and the module that governs that layer.

*Table 2. Description of the RSETPy Application Layers.*

Layer	Module Name	Description
Downloading	RSETDownloads	Downloads raw data from various sources.
Database Indexing	RSETBuildSrcDB	Indexes the raw data files in the MySQL database.
Algorithm Execution	RSETAlgorithms	Reads raw data and produces results using specific algorithms.
Post Processing	RSETPostProcess	Produces graphical representations of the results data. Is exclusive of the execution path governed by the RSETProcess module.

RSETPy uses text based configuration files to dictate the execution flow of the application. Each layer module, including the main module, RSETProcess, reads these configuration files. This provides the ability to enable and disable certain functions within RSETPy without having to modify the Python code.

RSETProcess begins the program execution by reading the [Main.in](#) configuration file. This file dictates what layers will be run, and specifies the geographic datum and whether the program will output error logs. Table 3 describes the configuration values under the Actions section of the Main.in file. The Actions section dictates layer activation.

Table 3. Description of configuration values for “Actions” section of Main.in.

Actions	
Name	Description
Downloads	1 activates Downloads layer, deactivation is specified by 0
DBIndexing	1 activates DBIndexing layer, deactivation is specified by 0
Algorithm	1 activates Algorithm layer, deactivation is specified by 0

Table 4 describes the Modules section of the Main.in configuration files. This section specifies the configuration file names used by the layers of RSETPy.

Table 4. Description of configuration values for “Modules” section of Main.in.

Modules	
Name	Description
Downloads	Defines the name of the Downloads layer configuration file (e.g. downloads.in)
Algorithm	Defines the name of the Algorithms layer configuration file (e.g. Algorithm.in)
Database	Defines the name of the MySQL database connection information configuration file (e.g. Database.in). This file is used by the Database layer to connect to the MySQL database.
Sources	Defines the name of the Database layer configuration file (e.g. Sources.in)
PostProcess	Defines the name of the PostProcessing layer configuration file (e.g. PostProcess.in)

The configuration file named within the Modules section are given with respect to the Main.in file. That is, if the Downloads configuration file name is given as input/downloads.in, for example, then the downloads.in file will be in a sub-directory called input. This sub-directory will be in the same directory as Main.in.

Simple if statements are used within the RSETProcess module to activate or deactivate the various layers of RSETPy based on the values within the Main.in configuration file. Figure 1 displays this process.

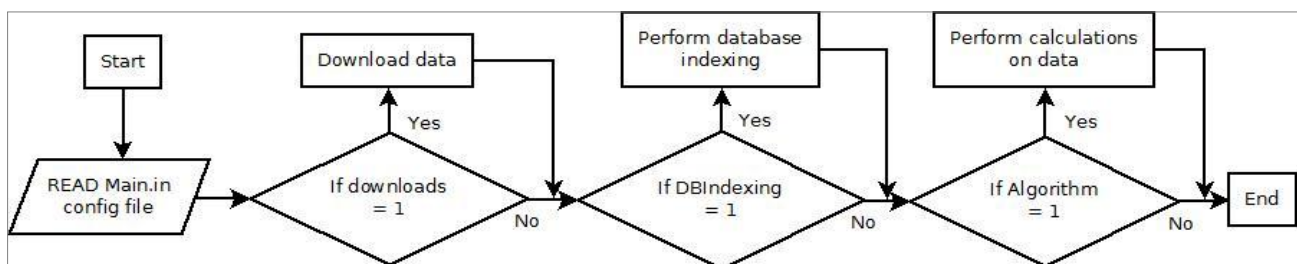


Figure 1

While each layer can be executed separately from the other RSETPy layers, it makes sense that raw data would already have to be downloaded for database indexing to occur. Further, for the algorithms layer to produce usable results, both downloading and database indexing must have already been completed. Thus, while RSETPy layers can be activated or deactivated at will, results will only be produced if preceding layers have been executed at least once.

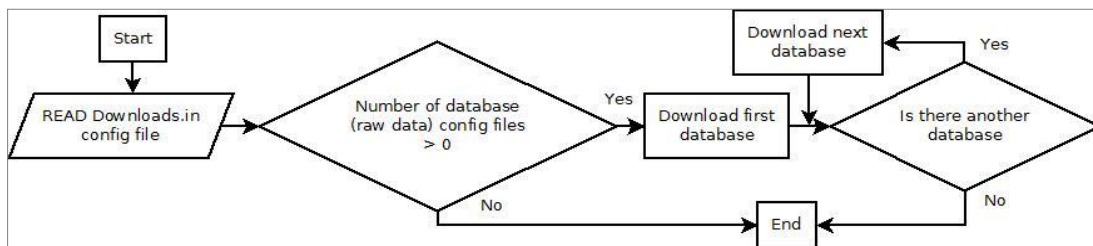
## Download Layer

The download layer of RSETPy handles the retrieval of raw data from various remote sources which are defined in the Downloads.in configuration file. The Downloads.in file specifies these remote sources in a section called databases. Within the databases section of the configuration file is a list of database configuration files that define the parameters necessary to download the raw data. In the computer's file-system, the database configuration file locations are relative to the location of the Downloads.in file. Table 5 describes the databases section of the Downloads.in file.

*Table 5. File names of the sub-configuration files specifying raw data locations.*

databases	
Name	Description
file_0	Configuration file name for first raw data database (e.g. Downloads/NWSDaily.in)
file_1	Configuration file name for second raw data database (e.g. Downloads/MODIS_2010.in)
file_2	Configuration file name for third raw data database (e.g. Downloads/NOMADS.in)
file_n	Configuration file name for n <sup>th</sup> raw data database

The RSETDownloads module performs a loop over the configuration file names listed in the databases section and is shown in Figure 2.



*Figure 2*

Each database configuration file contains a section called database. Within that section is a value called Name, which specifies to RSETDownloads the name of a source module to load. This source module will handle the downloading of the raw data from the database that is in the current iteration of the download loop. This functionality is shown in Figure 3.

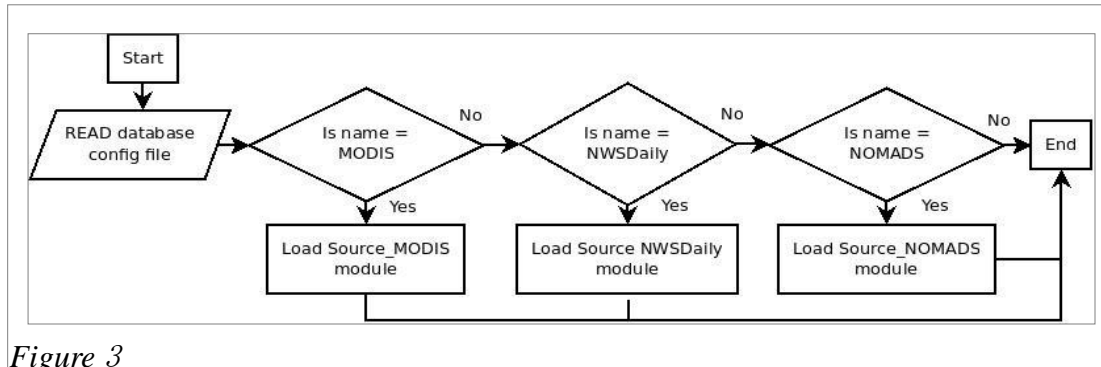


Figure 3

Once RSETDownloads loads the appropriate source module, the source module will read the [database configuration file](#) to determine the groups of raw data that will be downloaded for the current database. Within the current database configuration file, the database section contains another value called groupstoprocess. That value defines a list of group names which are defined in the database configuration file as subsections of a section called Groups. Table 6 describes the values under a typical group subsection.

Table 6. Description of the “Groups” sub-section values in a database configuration file.

Groups Subsection	
Name	Description
name	
type	Type of download to perform. A value of “download_ftp” will instruct the sub-module to use Python's built-in FTP client to retrieve data from a remote source. A value of “migrate” will result in raw data that is stored locally to be moved to migration location, specified by “deststorepath.”
deststorepath	The top-level directory location where the data will be stored.
startdate	Specifies the start date of the data that will be downloaded.
enddate	Specifies the end date of the data that will be downloaded.
srcdatalocation	Used only when the type value is set to “migrate.” Specifies where the raw data that will be moved to the deststorepath is located.
srcdatalocsubpathpattern	Used only when the type value is set to “migrate.” Specifies the name of a sub-directory under srcdatalocation that contains the data that will be migrated to the new storage paradigm.
alloweddata_ext	Used only when the type value is set to “migrate.” Specifies the allowed extensions of the data that will be migrated.
networkprotocol	Specifies the protocol to use to download source data. For example, “ftp” specifies the file transfer protocol will be used.
domain	The remote domain name of the raw data database. Used to initiate a remote connection.
remotepath	The location of the raw data on the remote server.

The source module loads all the group information within the current database configuration file into lists based on the download type value. This program flow is shown in Figure 4.

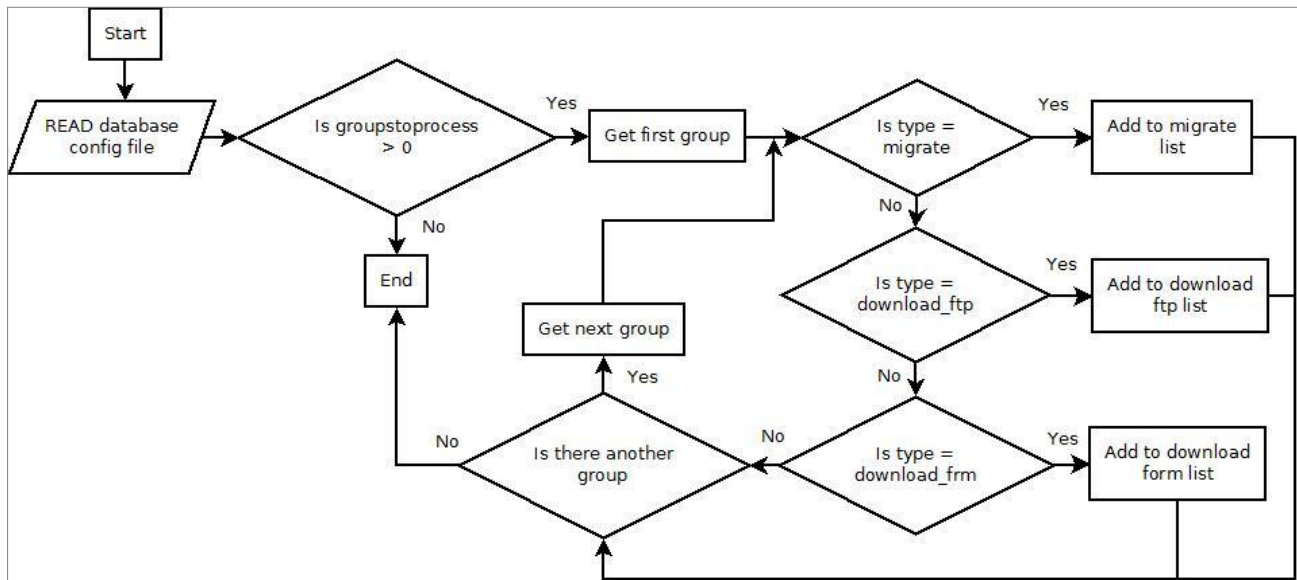


Figure 4

Once the lists are populated, the source module iterates over each list, thus downloading the raw data to the local file system destination. Figure 5 describes this procedure.

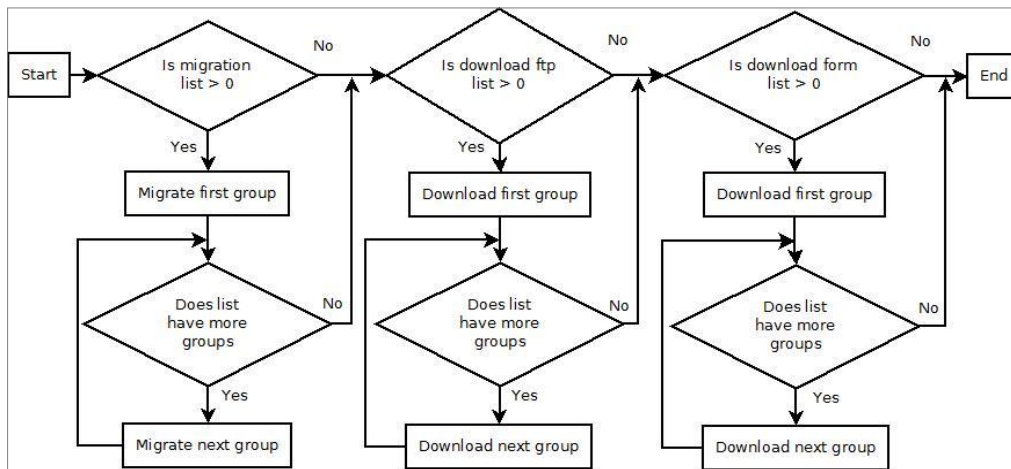


Figure 5

As shown in Table 6, the value of deststorepath is top level location where the download layer saves raw data. Raw data files are further organized by year, month, day, and download type. Figure 6 shows a data directory structure for MOD03 data where the top level destination directory for raw data is "D:/."

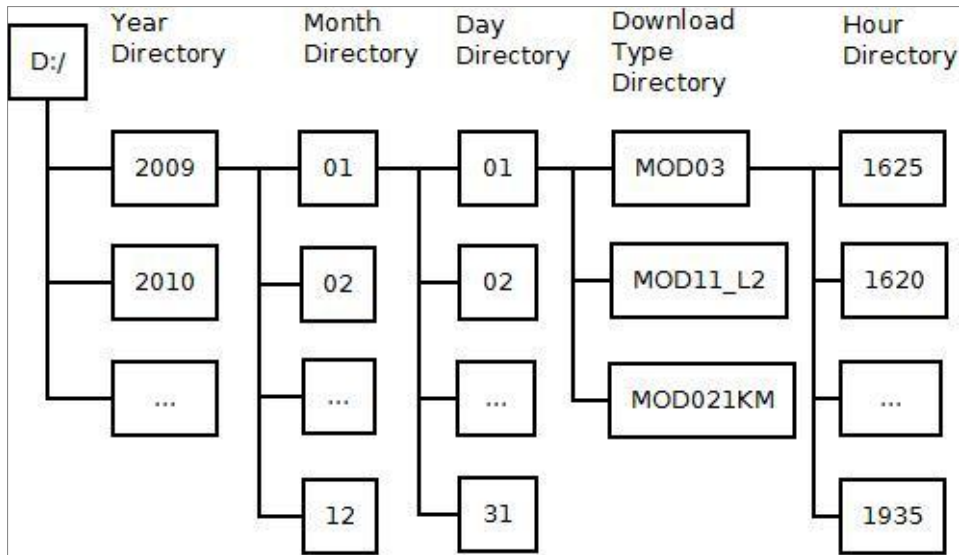


Figure 6

After program execution has flowed through the source module, RSETDownloads iterates over the remaining raw data database configuration entries, thus finishing program execution in the download layer of RSETPy.

### Database Indexing Layer

Since RSETPy is designed to download data from multiple sources, there is no guarantee that the raw data will be in the same format and contain the same variables. Further, source data files for a given date and time may be quite large. MODIS data is an example, with possible files sizes around 200MB. This makes it difficult to store data directly in a database. Instead, the RSETBuildSrcDB layer parses and indexes the raw data files, storing the file paths to the files for a given set of parameters (date, time, variable type, and location).

To store the geographical locations of the downloaded raw data, the RSETBuildSrcDB layer creates a scenes database table. This table stores latitude and longitude pairs that define a bounding box where the data was observed. Table 7 describes the scenes database table column values.

Table 7. Description of the Scenes table in an RSETPy MySQL database.

Name	Type	Description
id	Integer	Primary Key for scenes table.
sourcetype	String	Type of source data (MODIS, NWSDaily, ...)
sw_lat	Floating Point	Latitude value of South-West corner of the scene.
sw_lon	Floating Point	Longitude value of South-West corner of the scene.
se_lat	Floating Point	Latitude value of South-East corner of the scene.
se_lon	Floating Point	Longitude value of South-East corner of the scene.

<b>Name</b>	<b>Type</b>	<b>Description</b>
ne_lat	Floating Point	Latitude value of North-East corner of the scene.
ne_lon	Floating Point	Longitude value of North-East corner of the scene.
nw_lat	Floating Point	Latitude value of North-West corner of the scene.
nw_lon	Floating Point	Longitude value of North-West corner of the scene.

For every year of raw data downloaded, a scenes table will be created. Thus, if raw data for 2010 and 2011 were downloaded, the RSETPy MySQL database will have scenes\_2010\_wgs84 and a scenes\_2011\_wgs84 tables.

During the process of indexing the raw data, RSETBuildSrcDB creates a sources table that contains the information about a single day's worth of downloaded raw data. Therefore, if data was downloaded for January 1<sup>st</sup>, 2010 and January 2<sup>nd</sup>, 2010, the database will contain a date\_20100101\_src table and a date\_20100102\_src table. Table 8 describes the sources table column values.

*Table 8. Description of the Sources table in an RSETPy MySQL database.*

<b>Name</b>	<b>Type</b>	<b>Description</b>
id	Integer	Primary key for table.
basepath	String	Path to source data for a specific date and source type.
subpath	String	Path to a specific file under the basepath.
sourcetype	String	Type of source data (MODIS, NWSDaily, ...)
drivertype	String	The file type of the raw data file.
year	Integer	Year the raw data was observed.
month	Integer	Month the raw data was observed.
day	Integer	Day the raw data was observed.
hour	Integer	Hour the raw data was observed.
minute	Integer	Minute the raw data was observed.
varname	String	Name of the variable this database entry references.
vardesc	String	A description of the variable this database entry references.
infilepath	String	The path within the source file, for files like HDF 5.
subsetindex	Integer	An index to a subset of data stored in an HDF 5 file.
sceneid	Integer	Foreign key to the scenes table for the year of the raw data.

Assuming that the RSETDownloads module has already completed the task of downloading raw data, then the database indexing layer will begin the task of creating the necessary database tables and indexing the raw data. RSETBuildSrcDB begins this process by reading the [Database.in](#) configuration file. This file contains one section, Database, which holds the values required to connect to the RSETPy MySQL database. Table 9 describes this section of the Database.in configuration file.

Table 9. Database values needed to connect to the MySQL database.

Database	
Name	Description
db_host	The host computer name where the RSETpy MySQL database is served.
db_name	The name of the RSETpy MySQL database.
mysql_username	The user name that has read and write permissions on the RSETpy MySQL database.
mysql_passwd	The password value for the user that has access to the RSETpy MySQL database.
wipe_db_tables_all	Set to "1," this value instructs the database indexing layer to wipe all pre-existing scenes and sources tables from the database.

Once a database connection is achieved using the values in the Database.in configuration file, the RSETBuildSrcDB module reads the [Sources.in](#) configuration file. This file contains a section, Groups, which defines subsections where the subsection values specify the raw data that will be indexed by the database indexing layer. Table 10 describes the values under a typical group subsection.

Table 10. Description of the Groups sub-section values in the Sources.in configuration file.

Groups Subsection	
Name	Description
type	The source type. Options are: NWS, MODIS, NOMADS.
basepath	The base path under which all files with permissible extensions are evaluated.
extension	The program will search only files to be indexed with the given extensions. Examples include: h5, hdf, or bin.
converter	

Figure 7 describes the initial program flow of the database indexing layer.

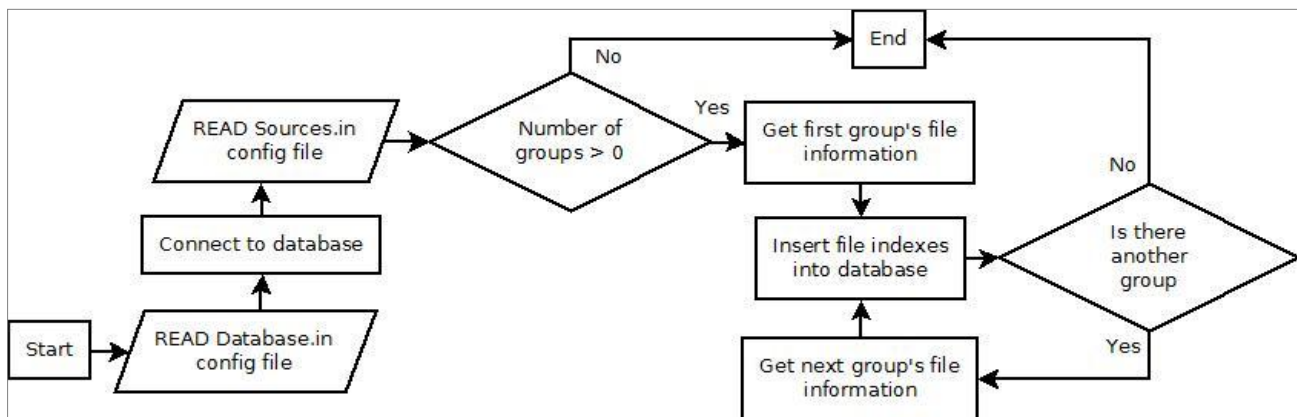


Figure 7

Within the file insertion step in Figure 7, more complex operations are performed. A list of files is determined for a group in a given iteration of the RSETBuildSrcDB loop, and the files' metadata is inserted into the source database tables. The geographic scenes, sets of latitude and longitude points that form a bounding box, are also stored in the database, which puts the metadata for the files in a geographic context. Figure 8 shows this operation in more detail.

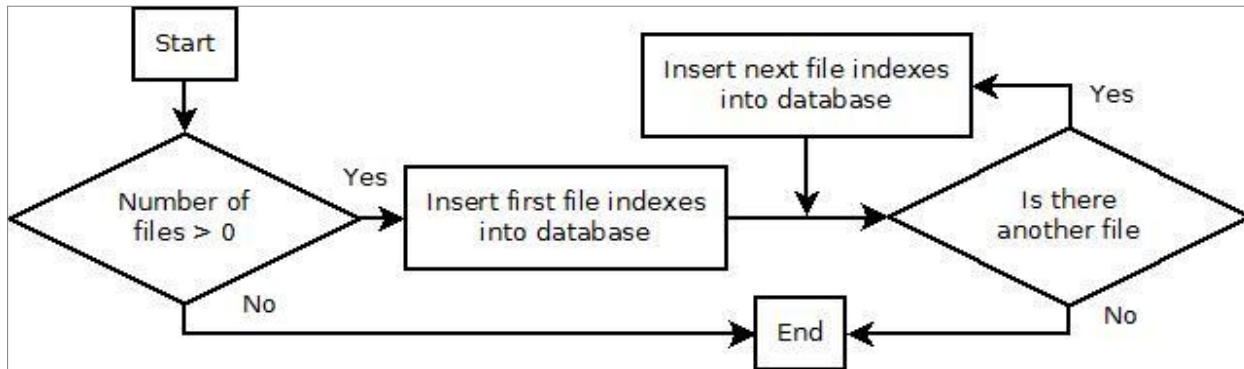


Figure 8

Further, inserting file indexes into the database requires the database indexing layer to load a specific file parser sub-module for each file in the file list. The type of file is determined by the type value in the Groups subsection of the Sources.in configuration file. Once the proper sub-module is loaded, the file is parsed for the geographic location information and for the data contained within the file. The geographic scene information and the indexes to the file data are written to the database. These operations are shown in Figure 9.

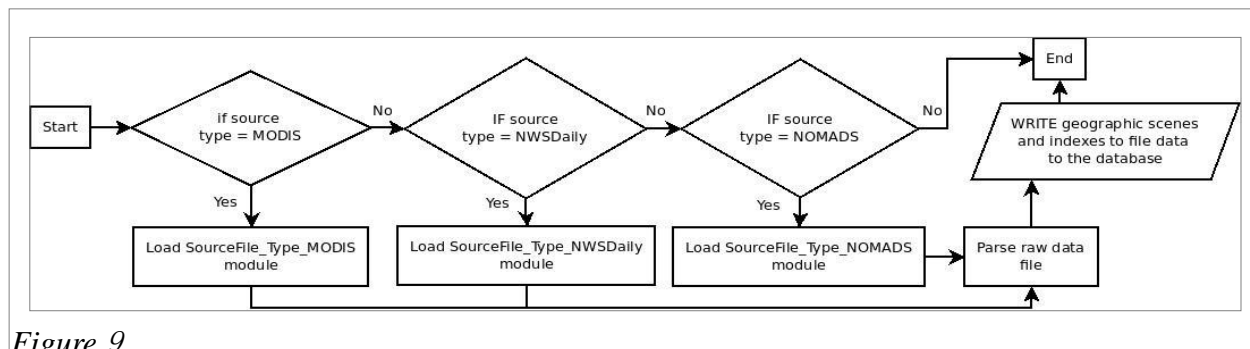


Figure 9

With the database indexing completed, the algorithms layer of RSETPy will be able to find the raw data files by the type of data, the date of observation, the variable types, and the location where the data was observed.

## Algorithms Layer

The algorithms layer is the last layer to be executed by the RSETProcess main module. This layer requires that raw data has been downloaded by the RSETDownloads module and indexed by the RSETBuildSrcDB module. Execution of this layer begins with the RSETAlgorithms module reading the [Algorithm.in](#) configuration file. This configuration file contains values that determine which algorithms will be run and where the specific algorithm configuration files are located. Table 11 describes the Algorithm section of the Algorithm.in configuration file.

Table 11. Description of the Algorithm section of the Algorithm.in configuration file.

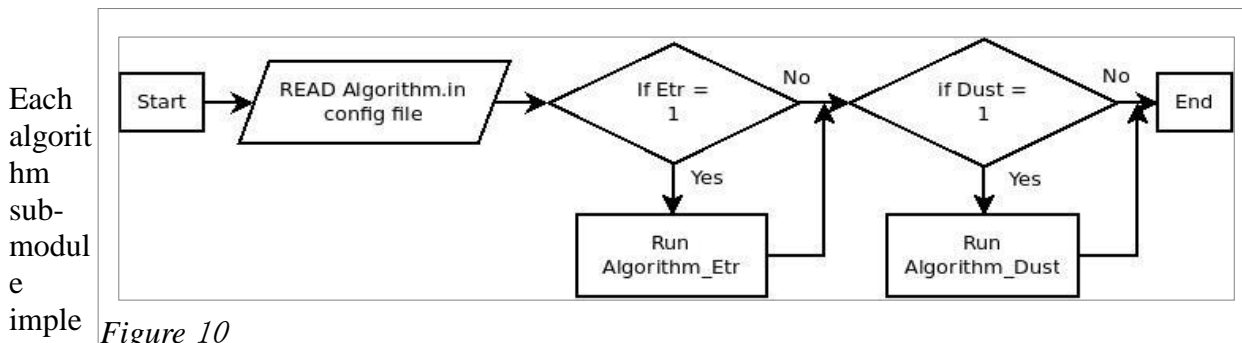
Algorithm	
Name	Description
Etr	Set to “1,” the RSETAlgorithms module will run the Etr algorithm
Dust	Set to “1,” the RSETAlgorithms module will run the Dust algorithm

The Algorithm.in configuration file also contains a section called Modules, which contain values that describe where to find the specific algorithm configuration files. As with the Main.in configuration file, the specific algorithm configuration files are located relative to the Algorithm.in file. The Modules section is described in Table 12.

Table 12. Description of the Modules section of the Algorithm.in configuration file.

Modules	
Name	Description
Etr	Configuration file name for the Etr algorithm (e.g. “Algorithm/Etr.in”)
Dust	Configuration file name for the Dust algorithm (e.g. “Algorithm/Dust.in”)

The RSETAlgorithms uses the Modules section of the Algorithm.in configuration file to determine the sub-module to load. This sub-module is responsible for the calculation of the results data. Figure 10 describes this process.



the same base functionality, so the run operations depicted in Figure 10 proceed in a similar manner. The algorithm sub-module creates a connection to the database using the Database.in configuration file. Next, the configuration file specifically for the sub-module is read. Within this algorithm configuration file, there is a value called groupstoprocess. That value defines a list of group names which are defined in the algorithm configuration file as subsections of a section called Groups. Table 13 describes the values under a typical group subsection.

Table 13. Description of Groups sub-section of an algorithm configuration file.

Groups Subsection	
Name	Description
deststorepath	The file location where calculated data will be stored
mode	The mode of calculation: automatic or manual
automatic	Specifies start and end dates for the algorithm
manual	Specifies a driver file location for manual control over the algorithm execution

After the algorithm configuration file is referenced, the algorithm sub-module iterates over the groupstoprocess list. Figure 11 shows the program flow of the algorithm sub-module.

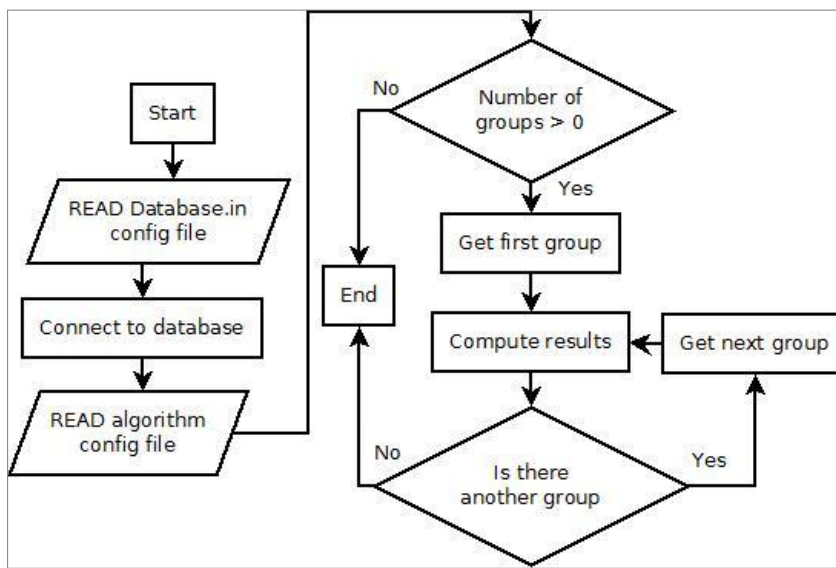


Figure 11

Within the compute results operation of Figure 11, the algorithm sub-module iterates over the dates between the given start and end dates from either the algorithm configuration file or the driver file, specified in the algorithm configuration file. For each date in the loop, results data is computed. Figure 12 describes this process.

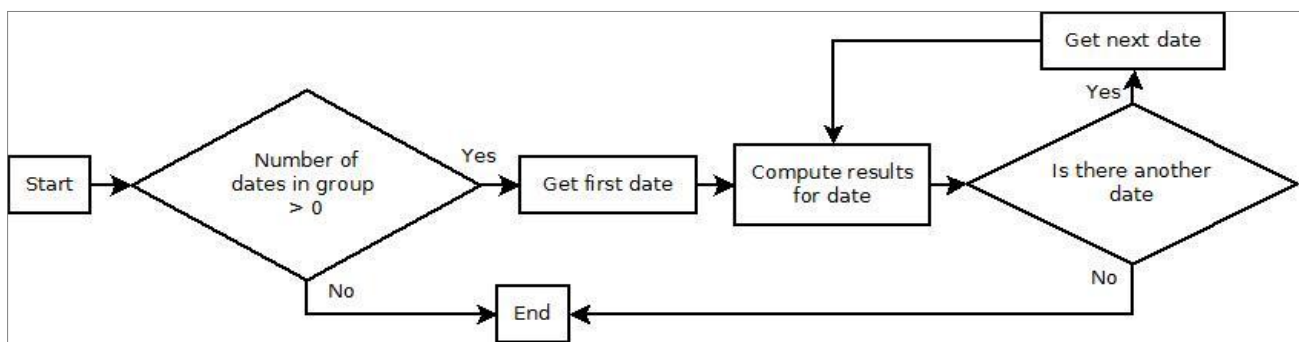


Figure 12

During each calculation of results per date, the algorithms layer accesses the database to determine if raw data exists for the location and time of results data. If so, the results data is written to the file system, and indexes to the results data are written to the RSETPy MySQL database for use by the post processing layer. Figure 13 describes this process.

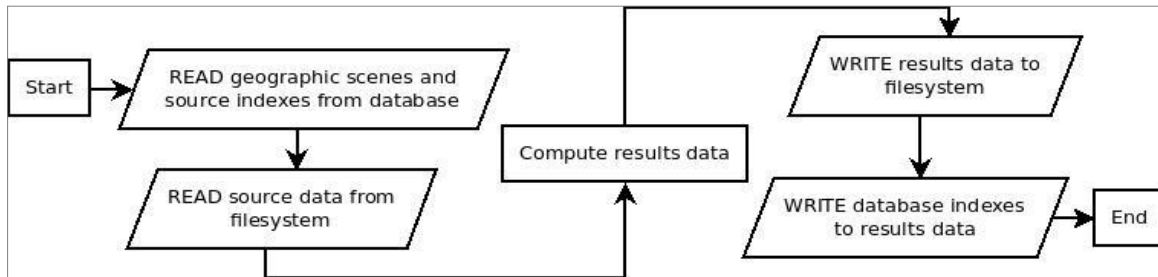


Figure 13

As with the sources tables, the results tables store indexes to results data files located in the local file system. Further, results tables exist for each date of calculated results and for each algorithm used to calculate results. The results table is described in Table 14.

Table 14. Description of the columns in a results table.

Name	Type	Description
id	Integer	Primary key for table.
basepath	String	Path to results data for a specific date and output file type.
subpath	String	Path to a specific file under the basepath.
drivertype	String	The file type of the raw data file.
year	Integer	Year value of the calculated data.
month	Integer	Month value of the calculated data.
day	Integer	Day value of the calculated data.
hour	Integer	Hour value of the calculated data.
minute	Integer	Minute value of the calculated data.
varname	String	Name of the variable this database entry references.
infilepath	String	The path within the results file, for files like HDF 5.
subsetindex	Integer	An index to a subset of data stored in an HDF 5 file.
sceneid	Integer	Foreign key to the scenes table for the year of the calculated data.

## Post Processing Layer

The post processing layer of RSETPy provides the capability to create time-series data and Google Earth overlays from data calculated by the algorithms layer. Post processing is not handled by the

main RSETProcess module and is instead executed by running the RSETPostProcess module separately. When post processing begins, the RSETPostProcess module reads the PostProcess.in text configuration file which contains a value called groupstoprocess. This value specifies a list of group names within the configuration file that dictates the attributes necessary to produce output. Table 15 describes the values of a typical groups subsection.

Table 15. Description of a Groups sub-section of the PostProcess.in configuration file.

Groups Subsection	
Name	Description
name	Specifies the name of the post processing output.
type	Denotes the type of results data to use to create output.
operator	Specifies the type of statistical output to create.
startdate	The start date of the output.
enddate	The end date of the output.
storepath	Specifies where the post processing output will be stored.

As with the other layers, post processing is accomplished by looping over post process groups. Figure 14 show this process.

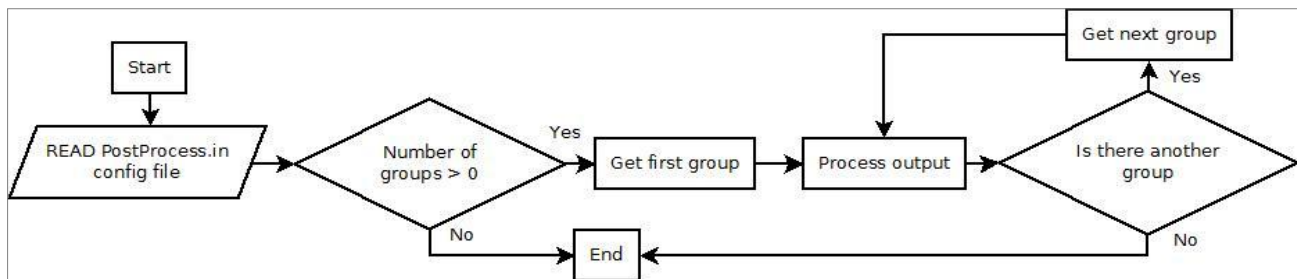


Figure 14

Currently, the RSETPy Post Processing layer is only implemented for post processing of dust algorithm output. Further, the post processing of the dust output requires modifications to the RSETPy source code to run properly.

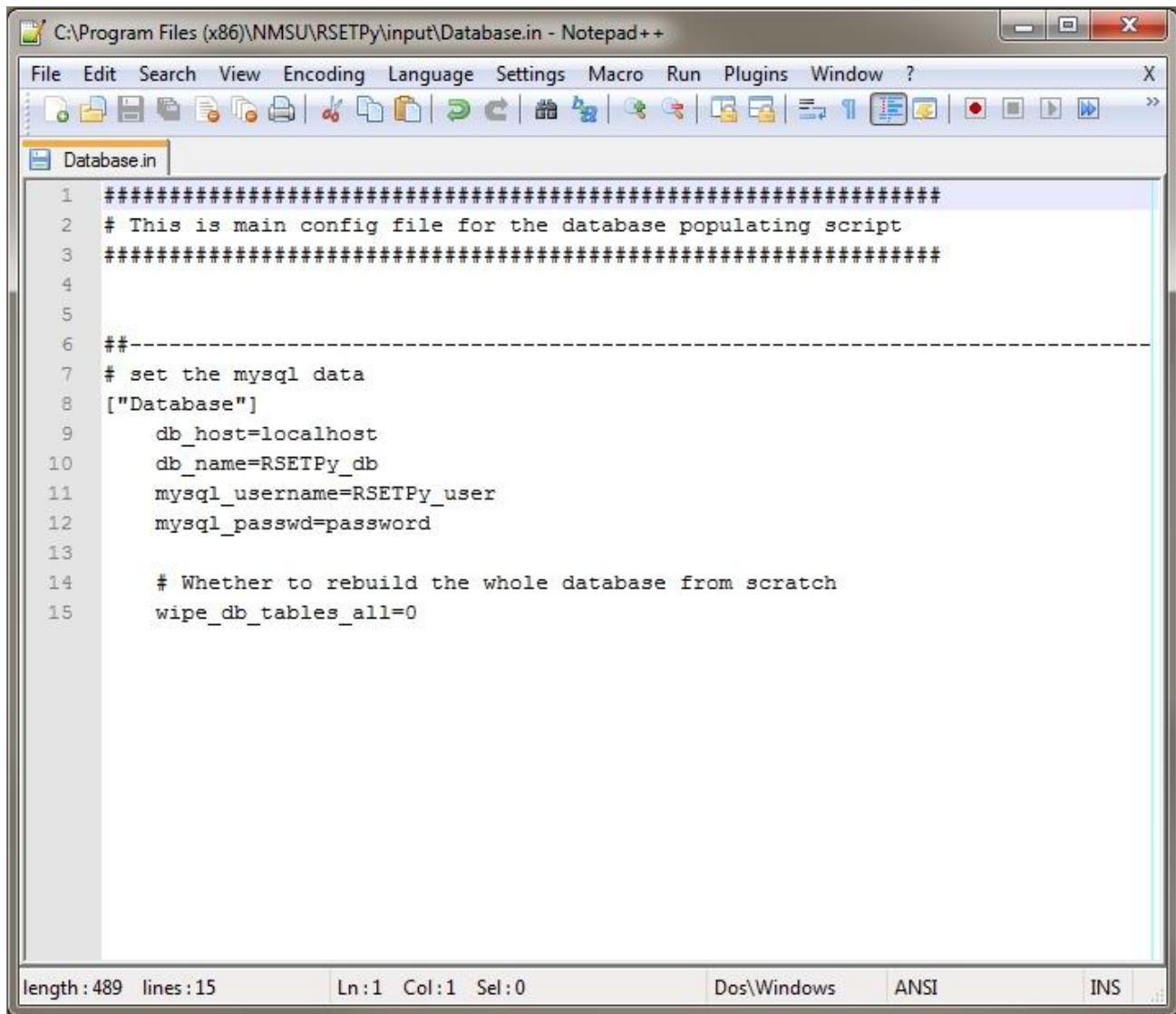
## Installation

The RSETPy Windows installation program only installs the core RSETPy code on the target system. In order to use RSETPy, the packages listed in Table 1 must also be installed. Instructions for the required packages are provided by the package authors. To install RSETPy, a Windows executable, RSETPy.exe, is provided. This will install the source code in a given directory location, or it will be installed in the default location: C:\Program Files (x86)\NMSU\RSETPy.

## Configuring RSETPy

Once the third-party packages and RSETPy are installed, RSETPy must be configured to work properly. In order to be able to execute all layers of RSETPy, a MySQL database must be created and RSETPy must be configured to read and write to the database. Assuming a database called

RSETPy\_db is created with full access granted to a user called RSET\_user, and the RSET\_user is given a password value of “password,” then the Database.in configuration file must be changed to reflect those values. This is accomplished by opening the text configuration files in a text editor. Figure 15 shows the Database.in configuration file open in Notepad++ (<http://notepad-plus-plus.org/>) with the changes necessary for RSETPy to access the database.

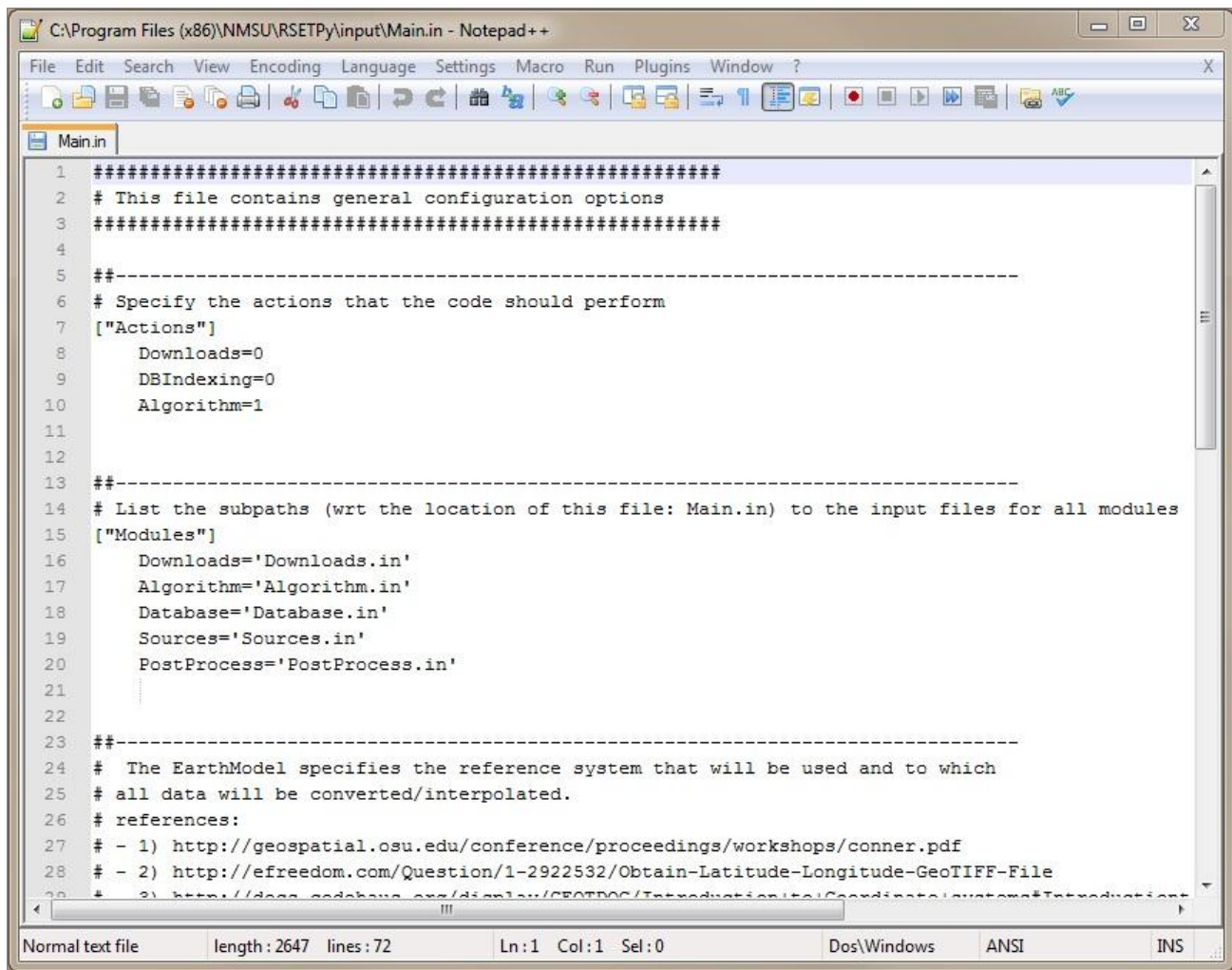


```
1 #####
2 # This is main config file for the database populating script
3 #####
4
5
6 ##-----
7 # set the mysql data
8 ["Database"]
9     db_host=localhost
10     db_name=RSETPy_db
11     mysql_username=RSETPy_user
12     mysql_passwd=password
13
14     # Whether to rebuild the whole database from scratch
15     wipe_db_tables_all=0
```

length: 489 lines: 15 Ln: 1 Col: 1 Sel: 0 Dos\Windows ANSI INS

Figure 15

To begin the process of producing results, the Main.in configuration file must be edited to activate the downloading, database indexing, and algorithms layers. Figure 16 shows the Main.in file with only the Algorithms layer enabled.

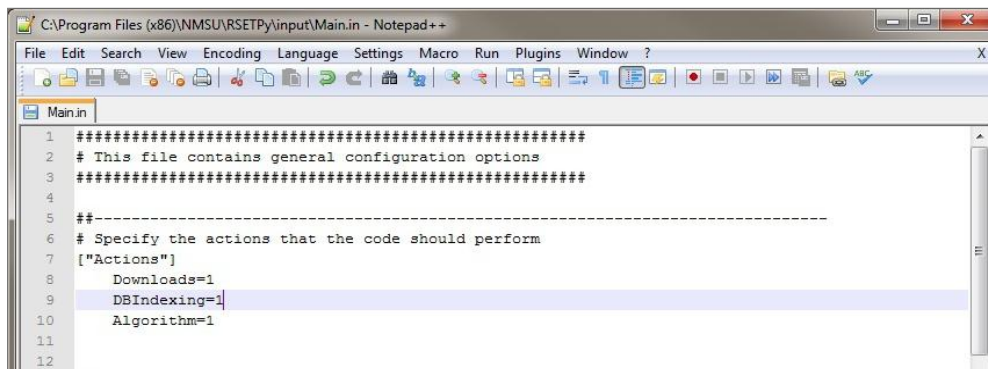


```
1 #####
2 # This file contains general configuration options
3 #####
4
5 ##-----
6 # Specify the actions that the code should perform
7 ["Actions"]
8     Downloads=0
9     DBIndexing=0
10    Algorithm=1
11
12
13 ##-----
14 # List the subpaths (wrt the location of this file: Main.in) to the input files for all modules
15 ["Modules"]
16     Downloads='Downloads.in'
17     Algorithm='Algorithm.in'
18     Database='Database.in'
19     Sources='Sources.in'
20     PostProcess='PostProcess.in'
21
22
23 ##-----
24 # The EarthModel specifies the reference system that will be used and to which
25 # all data will be converted/interpolated.
26 # references:
27 # - 1) http://geospatial.osu.edu/conference/proceedings/workshops/conner.pdf
28 # - 2) http://efreedom.com/Question/1-2922532/Obtain-Latitude-Longitude-GeoTIFF-File
29 # - 3) http://docs.cadaba.org/display/CEOTDOC/Introduction+to+Coordinate+systems+Introduction
30 #
```

Normal text file    length: 2647    lines: 72    Ln: 1    Col: 1    Sel: 0    Dos\Windows    ANSI    INS

Figure 16

As described in Table 3, the Actions section of the configuration file control which RSETPy layers are activated. To activate the Downloads and Database Indexing layer, change the values of Downloads and DBIndexing to 1. Figure 17 shows this change.



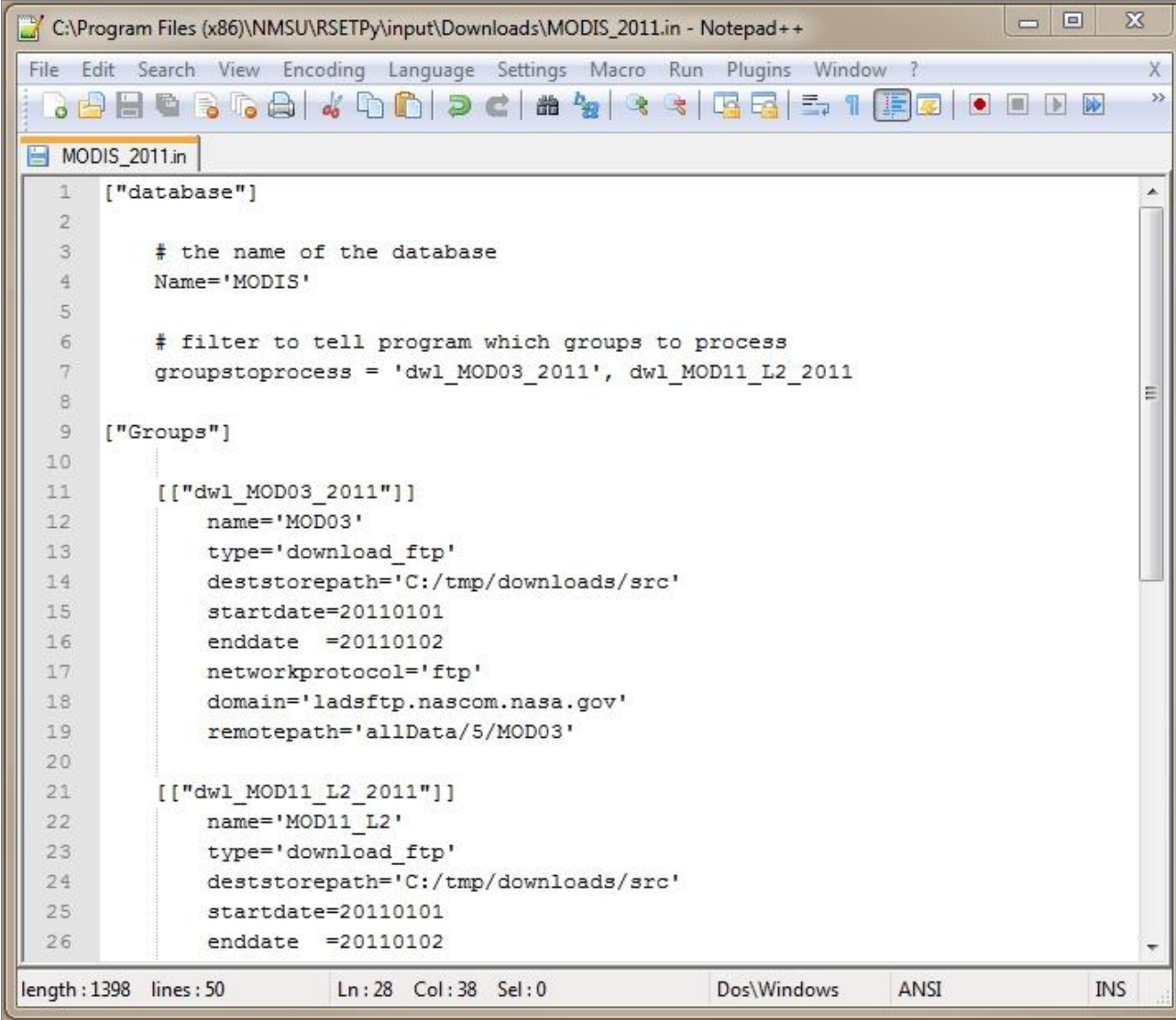
```
1 #####
2 # This file contains general configuration options
3 #####
4
5 ##-----
6 # Specify the actions that the code should perform
7 ["Actions"]
8     Downloads=1
9     DBIndexing=1
10    Algorithm=1
11
12
```

Normal text file    length: 2647    lines: 72    Ln: 1    Col: 1    Sel: 0    Dos\Windows    ANSI    INS

Figure 17

The download layer reads the Downloads.in configuration file. As shown in Table 5, the configuration

file lists locations for specific download configuration files under the databases section. To download data from MODIS, a text configuration file called Downloads/MODIS\_2011.in will be added to the databases section of the Downloads.in configuration file. Figure 18 shows this new configuration file open in Notepad++.



```

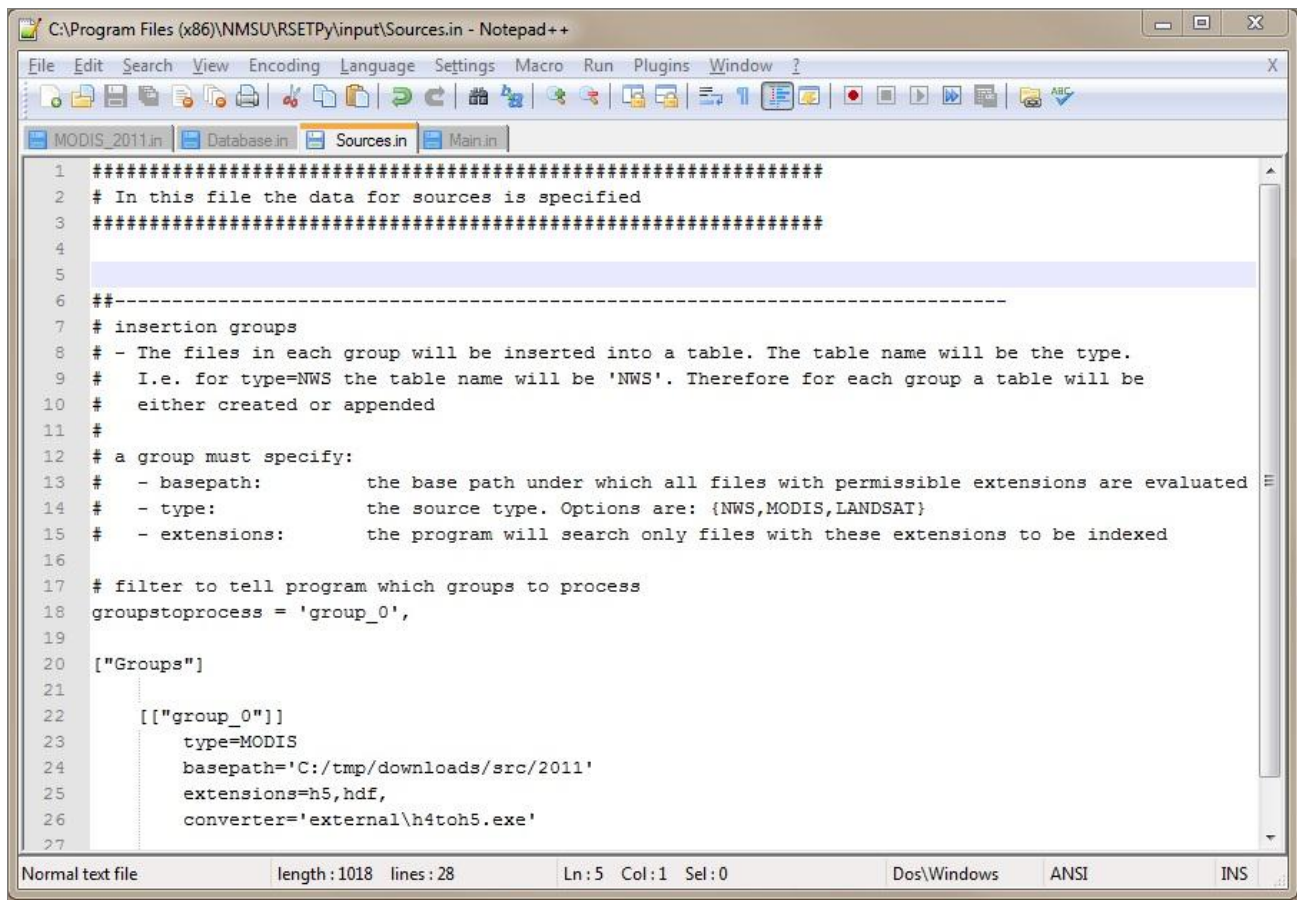
1  ["database"]
2
3  # the name of the database
4  Name='MODIS'
5
6  # filter to tell program which groups to process
7  groupstoprocess = 'dwl_MOD03_2011', dwl_MOD11_L2_2011
8
9  ["Groups"]
10
11  [{"dwl_MOD03_2011"}]
12      name='MOD03'
13      type='download_ftp'
14      deststorepath='C:/tmp/downloads/src'
15      startdate=20110101
16      enddate =20110102
17      networkprotocol='ftp'
18      domain='ladsftp.nascom.nasa.gov'
19      remotepath='allData/5/MOD03'
20
21  [{"dwl_MOD11_L2_2011"}]
22      name='MOD11_L2'
23      type='download_ftp'
24      deststorepath='C:/tmp/downloads/src'
25      startdate=20110101
26      enddate =20110102

```

length:1398 lines:50 Ln:28 Col:38 Sel:0 Dos\Windows ANSI INS

Figure 18

This configuration file directs RSETPy to download MOD03 and MOD11\_L2 data for the first and second days of 2011. In order for the downloaded data to be indexed by the database, the Sources.in configuration file must contain an entry in the Groups section of the file that points to the location of the data that was downloaded. Table 10 describes the value attributes of the group sub-section and Figure 19 shows the Sources.in configuration file configured to index the 2011 MODIS data.

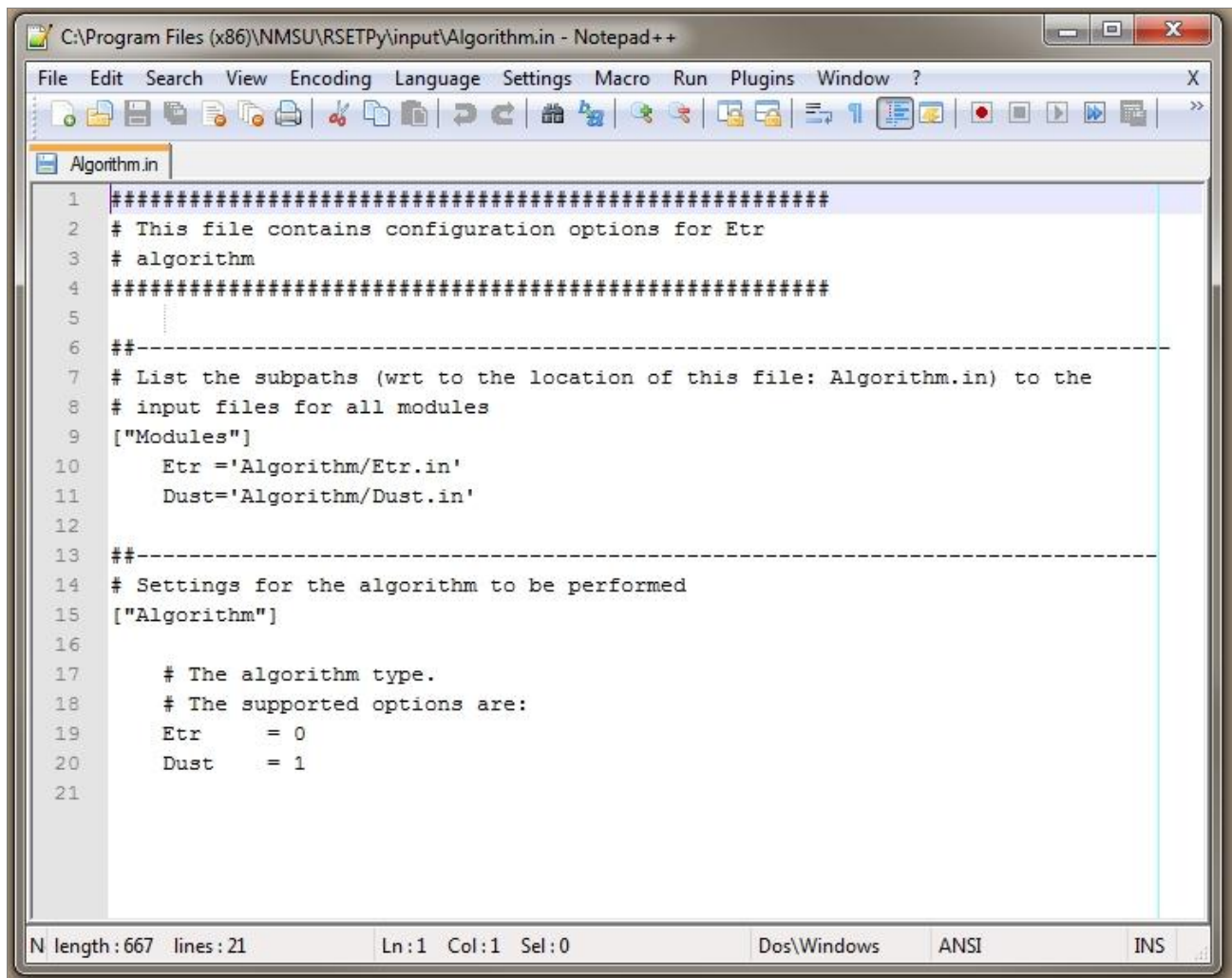


```
1 #####
2 # In this file the data for sources is specified
3 #####
4
5
6 ##-----
7 # insertion groups
8 # - The files in each group will be inserted into a table. The table name will be the type.
9 #   I.e. for type=NWS the table name will be 'NWS'. Therefore for each group a table will be
10 #   either created or appended
11 #
12 # a group must specify:
13 #   - basepath:      the base path under which all files with permissible extensions are evaluated
14 #   - type:          the source type. Options are: {NWS,MODIS,LANDSAT}
15 #   - extensions:    the program will search only files with these extensions to be indexed
16
17 # filter to tell program which groups to process
18 groupstoprocess = 'group_0',
19
20 ["Groups"]
21
22   [{"group_0"}]
23     type=MODIS
24     basepath='C:/tmp/downloads/src/2011'
25     extensions=h5,hdf,
26     converter='external\h4toh5.exe'
27
```

Normal text file      length: 1018   lines: 28      Ln: 5   Col: 1   Sel: 0      Dos\Windows      ANSI      INS

Figure 19

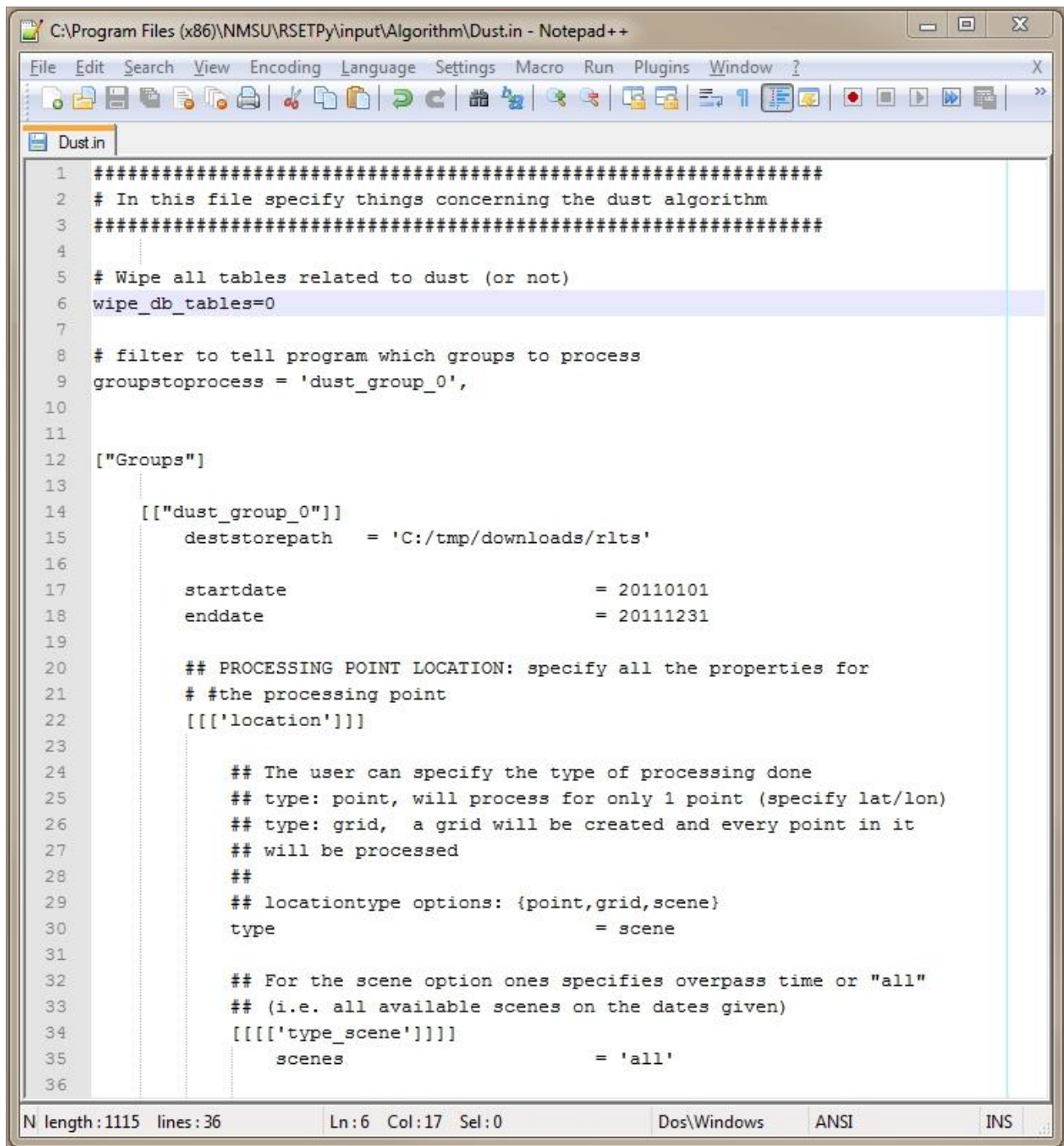
The algorithms layer references the Algorithm.in configuration file to determine which algorithm module to load and execute as shown in Table 11. Further, as shown in Table 12, the Modules section specifies where the specific algorithm module configuration files are located. As a result of the Dust Algorithm being activated, the Algorithm/Dust.in configuration file will be read by the algorithms layer. Figure 20 shows the Algorithm.in file configured to run the Dust algorithm.



```
#####
# This file contains configuration options for Etr
# algorithm
#####
.....
##-----
# List the subpaths (wrt to the location of this file: Algorithm.in) to the
# input files for all modules
["Modules"]
    Etr='Algorithm/Etr.in'
    Dust='Algorithm/Dust.in'
.....
##-----
# Settings for the algorithm to be performed
["Algorithm"]
.....
# The algorithm type.
# The supported options are:
Etr    = 0
Dust   = 1
.....
```

Figure 20

The Dust algorithm module configuration file specifies the points to compute results for a given date range. Figure 21 shows the Algorithm/Dust.in configuration file modified to compute results from January 1<sup>st</sup> 2011 to December 31<sup>st</sup> 2011 for all scene entries in the database.



```
1 #####
2 # In this file specify things concerning the dust algorithm
3 #####
4
5 # Wipe all tables related to dust (or not)
6 wipe_db_tables=0
7
8 # filter to tell program which groups to process
9 groupstoprocess = 'dust_group_0',
10
11
12 ["Groups"]
13
14     ["dust_group_0"]
15         deststorepath    = 'C:/tmp/downloads/rlts'
16
17         startdate        = 20110101
18         enddate          = 20111231
19
20         ## PROCESSING POINT LOCATION: specify all the properties for
21         # #the processing point
22         [[['location']]
23
24             ## The user can specify the type of processing done
25             ## type: point, will process for only 1 point (specify lat/lon)
26             ## type: grid, a grid will be created and every point in it
27             ## will be processed
28             ##
29             ## locationtype options: {point,grid,scene}
30             type          = scene
31
32             ## For the scene option ones specifies overpass time or "all"
33             ## (i.e. all available scenes on the dates given)
34             [[['type_scene']]
35                 scenes          = 'all'
36
```

Figure 21

## Running RSETPy

Once RSETPy is installed in the default location, it is possible to run the program manually using the Windows Command Line Interface or using the Windows Task Scheduler. Figure 22 shows the commands to run RSETPy manually, via the Windows Command Line Interface.

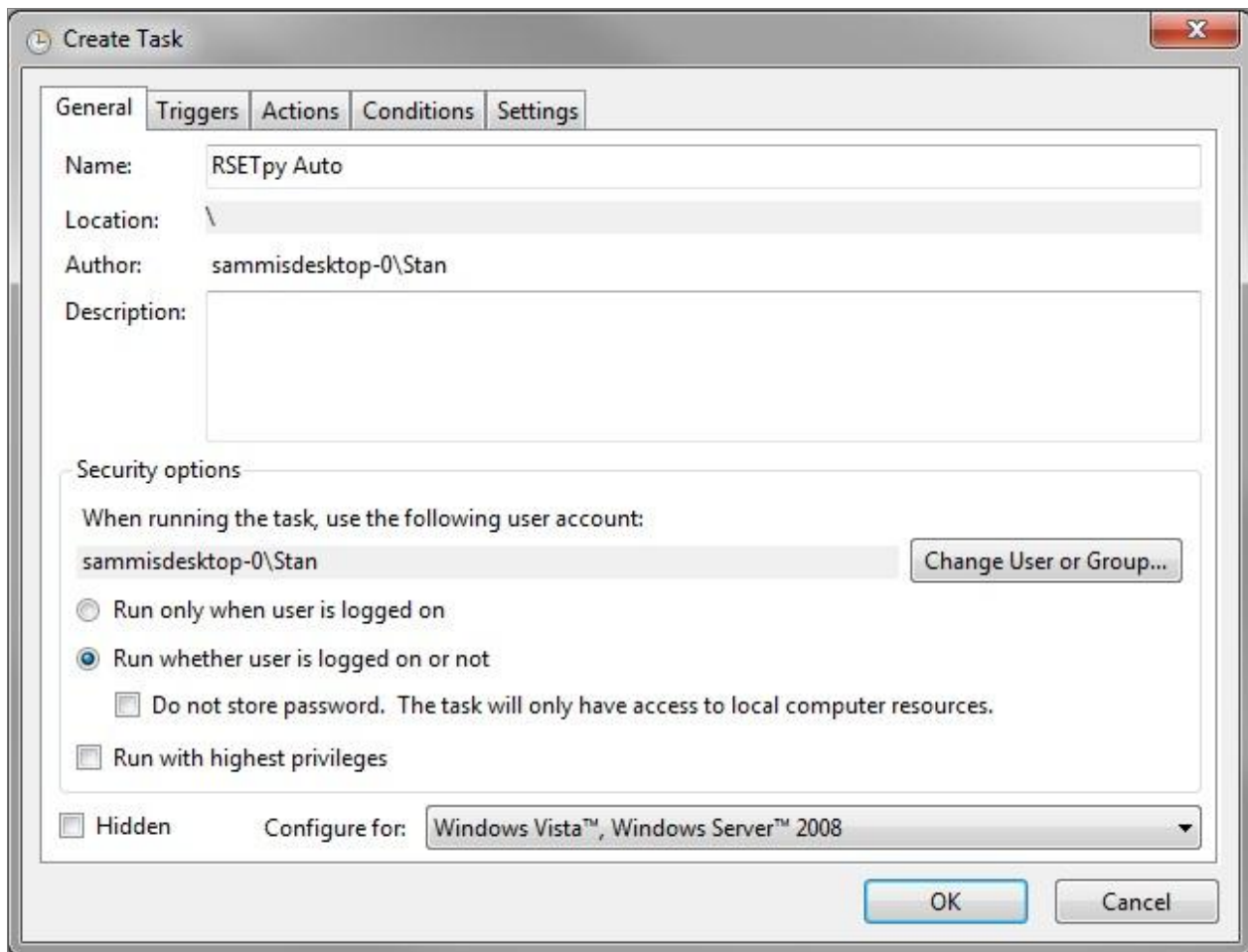


```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Stan>cd c:\
c:\>cd "Program Files (x86)"
c:\Program Files (x86)>cd NMSU
c:\Program Files (x86)\NMSU>cd RSETPy
c:\Program Files (x86)\NMSU\RSETPy>cd RSETProcess
c:\Program Files (x86)\NMSU\RSETPy\RSETProcess>cd src
c:\Program Files (x86)\NMSU\RSETPy\RSETProcess\src>python RSETProcess.py_
```

*Figure 22*

Running RSETPy automatically on a schedule requires configuring a task using the Windows Task Scheduler. After opening the Task Scheduler, create a new task that will run without needing a user logged into the system. Figure 23 shows the first step in creating an automatic task.



*Figure 23*

The next step in creating an automatic task is to set the schedule when the task will run. In this example, RSETPy is set to run daily at 8PM. Figure 24 shows this step.

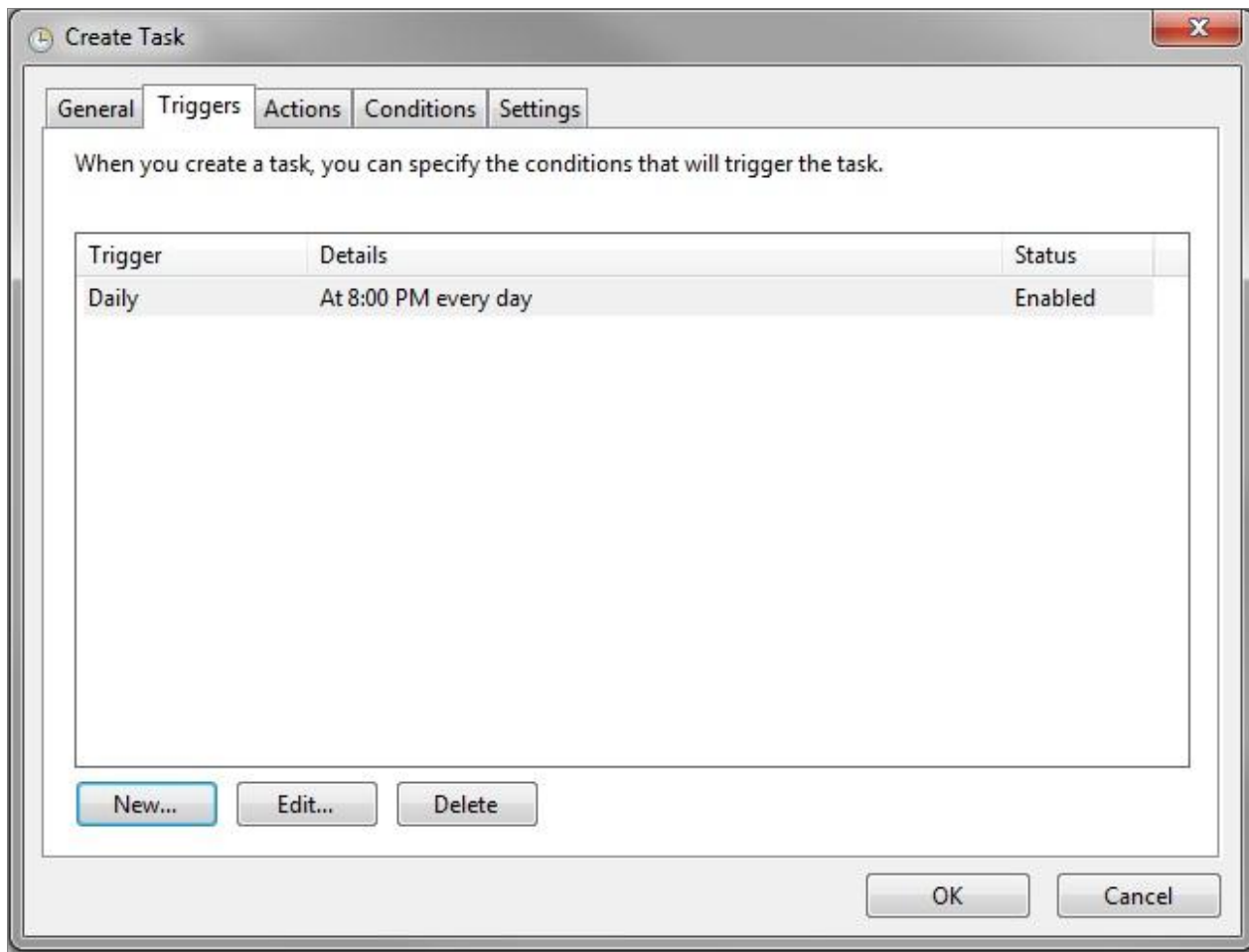
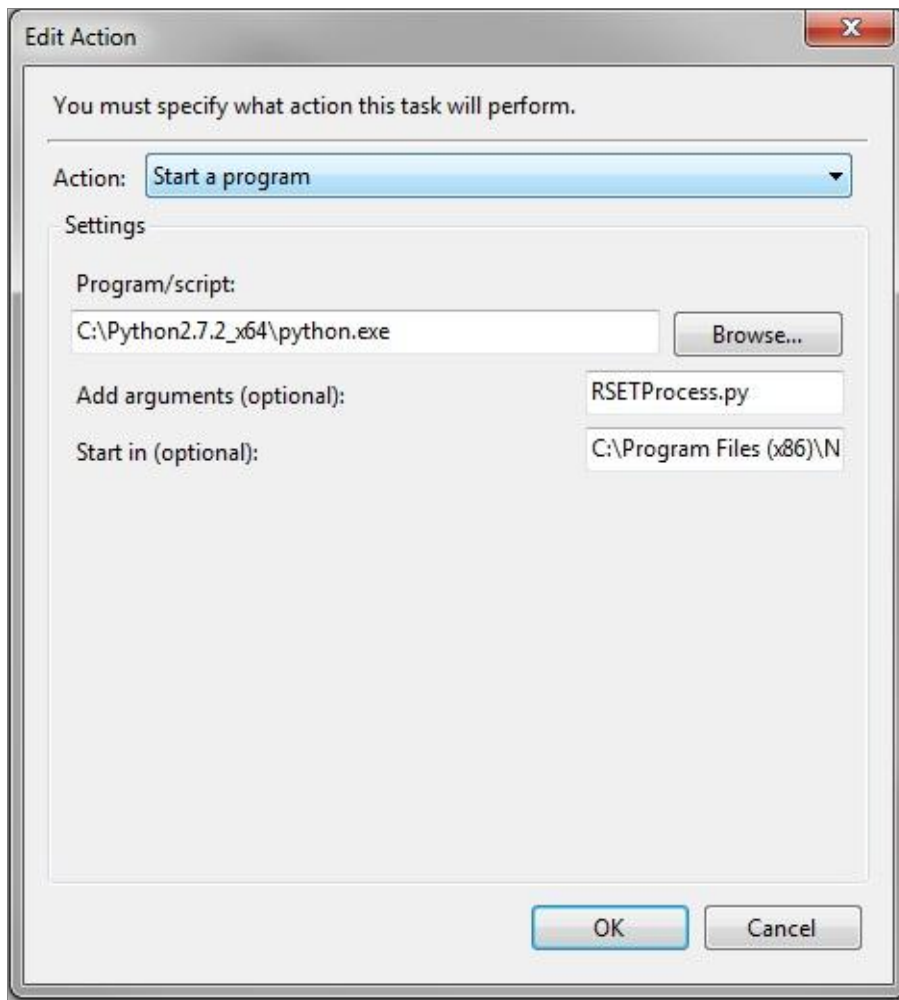


Figure 24

The third step requires setting the Python interpreter to run the RSETPy main module: RSETProcess.py. This is accomplished by creating a new action and pointing the program location to the Python interpreter. In this case, the Python interpreter is located at C:/Python2.7.2\_x64/Python.exe. The value for the Add Argument attribute is the name of the RSETPy main module. Finally, the value of the Start In attribute must be set to the installed location of the RSETProcess.py module. In this example, it is located in the following directory: C:/Program Files (x86)/NMSU/RSETPy/RSETProcess/src/. Figure 25 displays this step.

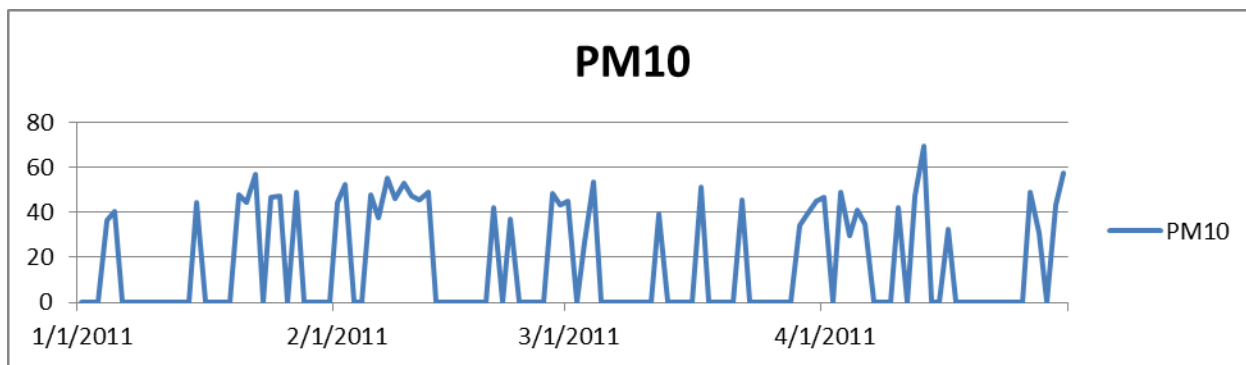


*Figure 25*

It is important to note that, while it is a simple command to run the RSETPy program, program execution still depends on the configuration of the text configuration files. If different results are required, the configuration files must be modified so that the next RSETPy run will follow the new modifications.

## **Results**

MODIS data for 2011 is downloaded via the RSETDownloads module and file metadata is stored in the database. A Dust algorithm calculates Aerosol Optical Thickness and PM10 concentrations from the 2011 data. Using the post processing layer, a time series of PM10 data is produced for a latitude/longitude location near Sacramento, CA (38.410558,-120.9507). Figure 26 shows this time series graph.



*Figure 26*

The zeroes shown in this time series is missing data.

## Future Work

The core RSETPy code is complete and able to produce results from source data. However, the post processing layer is incomplete and requires manual modifications to the source code to produce output from calculated data. Further, post processing for the ETr algorithm has not been implemented.

The text based configuration files that allow RSETPy to be configured without changing the source code can only be modified on a local machine. In order for a web interface to be implemented, a system of configuration file bookkeeping would have to be implemented for each web user. This would allow a web user to save custom configuration files and produce results independent from other web users.

## References

Lutz, M., 1996. Programming Python. O'Reilly & Associates, Inc.

Varoquaux, G., van der Walt, S., Jarrod Millman, J. 2009. Editorial Proceedings of the 8th Python in Science Conference. SciPy 2009.

<[http://conference.scipy.org/proceedings/SciPy2009/SciPy2009\\_proceedings.pdf](http://conference.scipy.org/proceedings/SciPy2009/SciPy2009_proceedings.pdf)>

Wilbers, I., Langtangen, H. P., Ødegård, A., 2009. Using Cython to Speed up Numerical Python Programs. In: Skallerud, B., Andersson, H. I. (Eds.), Proceedings of MekIT'09, NTNU, Tapir pp. 495-512

Linke, D., 2011. Performance Python. Retrieved December 19<sup>th</sup>, 2011 from

<<http://www.scipy.org/PerformancePython>>

Karssenber, D., de Jong, K., van der Kwast, J., 2007. Modeling landscape dynamics with Python, International Journal of Geographical Information Science, 21:5, 483-495

van der Kwast, H., de Jong, S., 2004. Modeling evapotranspiration using the surface balance system (SEBS) and landstat TM data (Rabat Region, Morocco). EARSeL Workshop on Remote Sensing for Developing Countries, Cairo, pp. 1-11

Hao, X., Qu, J.J., 2007. Saharan dust storm detection using moderate resolution imaging spectroradiometer thermal infrared bands. *Journal of Applied Remote Sensing* 1, 1-9

Chew, B.N., Chang, C.W., Salinas, S.V., Liew, S.C., 2008. Remote Sensing Measurements of Aerosol Optical Thickness and Correlation with In-Situ Air Quality Parameters During a Biomass Burning Episode in Southeast Asia. 28th Asian Conference on Remote Sensing (ACRS) Kuala Lumpur, Malaysia

## Appendix A. Theory of PM10 Dust Concentration.

### Dust

In the Dust algorithm aerosol optical thickness (AOT) is calculated first based on brightness temperatures using four bands (Table 16) from the MODIS satellite data and a linear regression Equation 1:

$$AOT = c0 + c1*BT20+c2*BT30+c3*BT31+c4*BT32 . \quad (1)$$

The coefficients in equation 1 were given by Xianjun Hao, John J. Qu 2007 and are expressed in Table 17. The Modis data is acquired from the MOD11\_L1 HDF files along with MODIS geo-location data presented in MOD03 files on the Internet and are read in by the RSETDownloads module of RSETPy.

*Table 16. Band and band width of MODIS data used in Dust algorithm.*

Band	Band Width
20	3.660 -- 3.840 $\mu$ m
30	9.580 -- 9.880 $\mu$ m
31	10.780 -- 11.280 $\mu$ m
32	11.770 -- 12.270 $\mu$ m

*Table 17. Coefficients for Equation 1.*

Coefficient	c0	c1	c2	c3	c4
Value	-7.9370	0.1227	0.0260	-0.7068	0.5883

After the AOT is calculated, it is converted to PM 10 concentration at the ground using a liner regression equation presented by Boon Ning CHEW et al., 2008. The equation is shown in Equation 2.

$$PM10 = 34.294 * AOT + 23.401 \quad (2)$$

## Appendix B. RSETPy Configuration Files Text

The contents of the configuration files for RSETPy are listed in this Appendix.

## Main.in Configuration File

```
#####
# This file contains general configuration options
#####

##-----
# Specify the actions that the code should perform
["Actions"]
    Downloads=0
    DBIndexing=0
    Algorithm=1

##-----
# List the subpaths (wrt the location of this file: Main.in) to the input files for all modules
["Modules"]
    Downloads='Downloads.in'
    Algorithm='Algorithm.in'
    Database='Database.in'
    Sources='Sources.in'
    PostProcess='PostProcess.in'

##-----
# The EarthModel specifies the reference system that will be used and to which
# all data will be converted/interpolated.
# references:
# - 1) http://geospatial.osu.edu/conference/proceedings/workshops/conner.pdf
# - 2) http://efreedom.com/Question/1-2922532/Obtain-Latitude-Longitude-GeoTIFF-File
# - 3) http://docs.codehaus.org/display/GEOTDOC/
# Introduction+to+Coordinate+systems#IntroductiontoCoordinatesystems-SomeGeodeticConcepts
# - 4) http://www.fgdl.org/tutorials/howto\_reproject/MapProjectionBasics.html
# - 5) http://www.stanford.edu/dept/itss/docs/oracle/10g/appdev.101/b10826/sdo\_cs\_concepts.htm
["EarthModel"]

    # see 1): A datum is a set of reference locations describing the surface of the earth
    # The supported options are:
    #     - WGS84
    GeodeticDatum='WGS84'

    # The coordinate system used to convert the data to and express the output in
    [[ CoordinateSystem ]]

        # The types of Coordinate System(s) are:
        #     - geographic (i.e. a position is projected to a ellipsoidal surface
        #     (units are expressed as lat/lon)
```

```

#      - projected (i.e. a position is projected to a planar surface)
# The supported options are:
#      - geographic
Type=geographic

# The projection type (this option is ignored unless Type=projected)
# (examples are (see 3): Mercator, Transverse Mercator,Lambert Conformal Conic and
# Stereographic)
# The supported options are:
#      -
Projection=-1

```

```

##-----
# Relevant information during the processing will be written to a logfile
["Logs"]

```

```

# the subpath (under the install path) where the logfile will be stored
subpath = 'logs'

```

```

# verbosity level (i.e. how much to print)
# The supported options are:
#      - 0      (nothing printed)
#      - 1      (print on error)
verbosity=1

```

## Downloads.in Configuration File

```

#####
# This file contains configuration options for download
#####

["databases"]
    File_0 ='Downloads/MODIS_2010.in'
    File_1 ='Downloads/MODIS_2011.in'

```

## Download Source Configuration File

The following is an example of a configuration file to download MODIS data.

```

["database"]

# the name of the database
Name='MODIS'

```

```

# filter to tell program which groups to process
groupstoprocess = 'dwl_MOD03_2011',

["Groups"]

[["dwl_MOD03_2011"]]
    name='MOD03'
    type='download_ftp'
    deststorepath='C:/tmp/downloads/src'
    startdate=20110101
    enddate      =20110102
    networkprotocol='ftp'
    domain='ladsftp.nascom.nasa.gov'
    remotepath='allData/5/MOD03'

## define special properties only relevant to all MODIS libraries
["SpecialProperties"]
    [["OverPassTimes"]]
        Day_0 =1745,1740,1920,1915,1600,1605
        Day_1 =1820,1825,1650,1645
        Day_2 =1730,1725,1905,1910,1915,1735,1550
        Day_3 =1815,1810,1950,1635,1630,1455
        Day_4 =1720,1855,1715,1540,1545
        Day_5 =1935,1800,1620,1625,1805
        Day_6 =1840,1845,1705,1710,1525,1530
        Day_7 =1745,1750,1925,1610,1615
        Day_8 =1835,1830,1650,1655,1515
        Day_9 =1735,1910,1915,1740,1600,1555
        Day_10      =1820,1815,1640,1955,1645,1500
        Day_11      =1900,1725,1720,1905,1550,1545
        Day_12      =1810,1810,1630,1945,1625
        Day_13      =1710,1845,1850,1715,1530,1535
        Day_14      =1750,1755,1930,1615,1620
        Day_15      =1835,1840,1700,1655,1520

```

## Database.in Configuration File

```

#####
# This is main config file for the database populating script
#####

##-----
# set the mysql data
["Database"]
    db_host=localhost
    db_name=RSETPy_db
    mysql_username=rset_user

```

```
mysql_passwd=rset_password

# Whether to rebuild the whole database from scratch
wipe_db_tables_all=0
```

## Sources.in Configuration File

```
#####
# In this file the data for sources is specified
#####

##-----
# insertion groups
# - The files in each group will be inserted into a table. The table name will
#   be the type. I.e. for type=NWS the table name will be 'NWS'. Therefore
#   for each group a table will be either created or appended
#
# a group must specify:
#   - basepath:          the base path under which all files with permissible
#                         extensions are evaluated
#   - type:              the source type. Options are: {NWS,MODIS,LANDSAT}
#   - extensions:        the program will search only files with these
#                         extensions to be indexed
#

# filter to tell program which groups to process
groupstoprocess = 'group_0',

["Groups"]

    ["group_0"]
        type=MODIS
        basepath='C:/tmp/downloads/src/2011'
        extensions=h5,hdf,
        converter='external\h4toh5.exe'
```

## Algorithm.in Configuration File

```
#####
# This file contains configuration options for Etr
# algorithm
#####

##-----
# List the subpaths (wrt to the location of this file: Algorithm.in) to the
# input files for all modules
["Modules"]
    Etr ='Algorithm/Etr.in'
```

```
Dust='Algorithm/Dust.in'
```

```
##-----
```

```
# Settings for the algorithm to be performed
```

```
["Algorithm"]
```

```
    # The algorithm type.
```

```
    # The supported options are:
```

```
    Etr    = 0
```

```
    Dust   = 1
```

## PostProcess.in Configuration File

```
#####
```

```
# This file contains configuration options for postprocessing
```

```
#####
```

```
# filter to tell program which groups to process
```

```
    groupstoprocess = 'pp_average',
```

```
##-----
```

```
# Settings for the postprocessing to be performed
```

```
["Groups"]
```

```
    ["pp_average"]
```

```
        name='PPAverage'
```

```
        type='DUST'
```

```
        operator='average'
```

```
        startdate=20100101
```

```
        enddate      =20100110
```

```
        storepath='C:/tmp/downloads/rlts'
```