Tafkaa Users' Guide

Marcos Montes & Bo-Cai Gao

Version 2002 June 14
for Tafkaa version 2002-04-23

Abstract

We present some background on *Tafkaa* , including differences and similarities in both the current tabular and 6S versions. Furthermore, we discuss a few data issues that may arise as *Tafkaa* is used in the *COIS* data stream.

1 Background

In the equations below, I will assume that the observations are being made from above the atmosphere, and, additionally, the observed surface is at sea level. The tabular version of *Tafkaa* is currently configured for aircraft above 2.5 km, and for sea-level surface. The current 6S version of *Tafkaa* can deal with aircraft in the atmosphere, and with elevated features. Oh, both assume that we're dealing with Earth (for atmospheric properties, latitude, longitude grid) at the current epoch (i.e., with the sun at its current luminosity).

Please consult Fraser et al. (1997), and Vermote et al. (1997) for further discussions about these equations. Actual papers about implementation of Tafkaa may be found in Gao et al. (2000) and Montes et al. (2001).

Our basic equation may be written as (please note that all the quantities in this and all other equations are described in the glossary section on page 16)

$$\begin{aligned} L_{\rm t} &= L_0 + L_{\rm sfc} t + L_{\rm g} t' \ , \qquad (1) \\ \text{where} & \\ L_0 &= L_0(\lambda;\theta,\phi;\theta_0,\phi_0;\tau_{\rm a}) \ ; \\ L_{\rm sfc} &= L_{\rm sfc}(\lambda;\theta,\phi;\theta_0,\phi_0;W;\tau_{\rm a}) \ ; \\ L_{\rm g} &= L_{\rm g}(\lambda;\theta,\phi;\theta_0,\phi_0;W;C;\tau_{\rm a}) \ ; \\ t &= t(\lambda;\theta;\tau_{\rm a}) \ ; \\ t' &= t'(\lambda;\theta;\tau_{\rm a}) \ . \end{aligned}$$

L_0 is the so-called "path radiance" term; that is, sunlight scattered by the atmosphere that never interacts with the ground. $L_{\rm sfc}$ accounts for the specular reflections off the target, and is only necessary over water; otherwise it is 0. $L_{\rm g}$ is the radiance reflected (in a Lambertian manner) from the ground (or water); $L_{\rm w} = L_{\rm g}$ for observations over water.

In the discussion below, it will be convenient to combine the first two terms in Equation 1 so that

$$L_{\rm sa} = L_0 + L_{\rm sfc} t \ . \qquad (2)$$

We assume that the absorptive processes are multiplicative and can be factored out directly; in essence we are breaking the transmission into absorptive and scattering parts. This is valid as long as either the absorptive or scattering processes dominate. There are some errors if they are similar in magnitude. This should only be a problem when the absorptive processes are changing rapidly (from the scattering dominated regime to the absorptive dominated regime) such as near edges of strong absorptive features. Thus,

$$L_{\rm t} = T_{\rm g} \left(L'_{\rm sa} + L_{\rm g} t'_{\rm u} \right) \ , \qquad (3)$$

where $T_{\rm g}$ is the transmission due to the absorptive processes in the gas.

We would like to work with reflectances instead of radiances, thus we divide by the incident flux at the top of the atmosphere $\mu_0 E_0$, and multiply by π:

$$\frac{\pi L_{\rm t}}{\mu_0 E_0} = T_{\rm g}\left[\frac{\pi L'_{\rm sa}}{\mu_0 E_0} + \frac{\pi L_{\rm g} t'_{\rm u} t_{\rm d}}{\mu_0 E_0 t_{\rm d}}\right] \ , \tag{4}$$

where we have multiplied the numerator and denominator of the second term on the right by the downward transmittance $t_{\rm d}$ in order to obtain the ground (or water-leaving) reflectance.

We will use the reflectances defined as

$$\rho_{\rm obs} \equiv \frac{\pi L_{\rm t}}{\mu_0 E_0} \ , \tag{5}$$

$$\rho_{\rm atm+sfc} \equiv \frac{\pi L'_{\rm sa}}{\mu_0 E_0} \ , \text{ and} \tag{6}$$

$$\rho_{\rm g} \equiv \frac{\pi L_{\rm g}}{\mu_0 E_0 t_{\rm d}} \ . \tag{7}$$

A final step is to remove the effect of multiple reflections of the ground or water leaving radiance from $t'_{\rm u}$. Neglecting specular reflection of the water leaving radiance off the ocean surface and assuming Lambertian reflectance for the water leaving radiance, we find

$$t'_{\rm u} = \frac{t_{\rm u}}{1 - \bar{s}\rho_{\rm g}} \ , \tag{8}$$

where $\bar{s} = \bar{s}(\lambda, \tau_{\rm a})$ is the average reflectivity of the atmosphere. This implies that

$$\rho_{\rm obs} = T_{\rm g}\left[\rho_{\rm atm+sfc} + \frac{\rho_{\rm g} t_{\rm u} t_{\rm d}}{1 - \bar{s}\rho_{\rm g}}\right] \ . \tag{9}$$

A more exact calculation would also include double reflections off of the ocean surface (water leaving light that is reflected downward of the atmosphere then specularly reflected by the surface); however, this effectively implies that the water leaving radiance is not Lambertian, and this makes it much more difficult to extract $\rho_{\rm w}$.

In the above equation, we regard $\rho_{\rm obs}$ as the measured quantity (since μ_0 and E_0 are well known); $\rho_{\rm g}$ is the quantity we wish to measure, and $\rho_{\rm atm+sfc}$, $t_{\rm u}, t_{\rm d}, \bar{s}$, and $T_{\rm g}$ are to be calculated by *tAKaA*. Thus, we rewrite the above equation with $\rho_{\rm g}$ on the left as

$$\rho_{\rm g} = \frac{\frac{\rho_{\rm obs}}{T_{\rm g}} - \rho_{\rm atm+sfc}}{t_{\rm u} t_{\rm d} + \bar{s}\left(\frac{\rho_{\rm obs}}{T_{\rm g}} - \rho_{\rm atm+sfc}\right)} \ . \tag{10}$$

2 Gaseous Absorptive Effects

The gaseous absorptive effects are treated differently for different sets of gases. The plane-parallel, two-way transmission of certain gases is calculated on a high-resolution (spectrally) grid, and this is carefully convolved to the medium resolution grid. At this point, ozone and NO_2 are included, and finally this is convolved with the instrumental response function to the instrumental grid.

In previous version of *Tafkaa* the instrumental response function (IRF) has been assumed to be Gaussian, thus only the center wavelength and the full-width-half-maximum (FWHM) were needed in order to perform the convolution. Certain instruments such as AVIRIS are very well represented by this model. Other instruments may not be. In order for *Tafkaa* to be as general as possible, I have added the ability for it to use a user-supplied instrumental response function. The default behavior is still to use the Gaussian IRF. If the keyword `tafkaa_instrumental_response_file` is set, then *Tafkaa* will use that value as the filename that contains the instrumental response function. Please see §C for a description of the correct format to use for the IRF file.

The different atmospheric models that can be selected in *Tafkaa* are principally used only to determine the pressure and water vapor volume mixing ratio at different heights in the atmosphere.

Temperature and pressure dependent effects are computed in a line-by-line code developed by W. Ridgway at NASA Goddard Space Flight Center to generate a database of gaseous absorption coefficients at a spectral resolution of 0.05 $\rm cm^{-1}$ in the 0.56 – 3.1 μm range for H_2O, CO_2, N_2O,

CO, CH_4, and O_2. These files are stored in the *Tafkaa* data directory, and each consists of 300,000 spectral locations and 19 atmospheric layers.

Water vapor has a unique treatment. *Tafkaa* , as originally written, calculated the water vapor amount on a pixel-to-pixel basis on the fly. In order to do this, 60 different water vapor values are used to calculate 60 high resolution gas transmission spectra. Indeed, this is the most time consuming portion of *Tafkaa* before the pixel-to-pixel processing begins, as this implies the high-to-medium resolution convolution needs to be done sixty times. Alternately, one can now neglect water vapor, or include a particular amount of water vapor. Either of these options greatly decreases the startup time, but water vapor features are not corrected as well (or at all). Still, if the strong water vapor bands are not available or the signal to noise in the bands is poor, these new features allow one to continue the analysis with *Tafkaa* . In any case, the water vapor volume mixing ratios of the particular atmospheric models are used to place the relative amounts of water vapor in the atmosphere. Thus, the portions of the spectrum most affected by the choice of atmospheric model are those portions containing water vapor features.

Ozone (O_3) absorption coefficients in the 0.3 – 0.8 μm region were derived from `LOWTRAN7` O_3 transmittance spectra calculated with the US76 atmospheric model. The wavelength spacing between any two adjacent points is 0.1 nm, with a resolution of 0.2 nm. *Tafkaa* assumes that O_3 occurs in a nearly infinitesimally thin layer at an altitude of 27 km above mean sea level. As long as the sensor is well below or above the true ozone layer, this is not a problem. If the sensor is in the true ozone layer, *Tafkaa* will probably return incorrect results in this portion of the spectrum. The absorption effects of man-made ozone are usually quite small compared to those of the stratospheric ozone, so the errors in neglecting the low-altitude man-made distribution are quite small.

The NO_2 cross sections in the 0.3 – 0.8 μm region correspond to the curve in the top plot of Fig. 1 of Solomon et al. (1999). Adjacent points are spaced at 0.1 nm intervals, each with a resolution of 0.2 nm. As this gas has just been added to those available for *Tafkaa* , a few important notes are included here. A column amount of 5×10^{15} molecules reduces the transmittance at 410 nm by about 0.5%, and currently it is treated as having the same distribution as described above for ozone. The new keyword `tafkaa_atmo_no2_scale` specifies a multiplicative factor to be used to scale the amount of NO_2 present. NO_2 is also produced by burning fossil fuels, and so is present in the lower layers of the atmosphere. In fact, in certain locations there may be more present in the troposphere due to pollution than is present in the stratosphere. In these cases, results from *Tafkaa* will not be as accurate.

So, until modifications are made to the O_3 and NO_2 distributions of *Tafkaa* , results should be accurate for space sensors (since no matter what it will be above both gases, and so the geometry is calculated correctly). Results for airborne sensors may be less accurate in polluted areas where significant amounts of NO_2 are present below the airplane.

Both the ozone amount and the NO_2 scale factor may be adjusted to obtain the correct absorption if the user is particularly careful in choosing the values of each that are given to *Tafkaa* . Both are scaled internally by $\mu_0^{-1} + \mu^{-1}$ when the sensor is above 27 km, and by μ_0^{-1} when the sensor is below 27 km. Since it is only the total amount that matters, it should be possible to determine a reasonable input if the tropospheric and stratospheric amounts can be measured separately.

3 Application to `tafkaa_6s`

Thorough discussions of the *6S* version of *Tafkaa* (i.e., the one we call `tafkaa_6s`) can be found in Gao et al. (1993) and in the *ATREM Users Guide* (Gao et al. 1997), which is available online.

`tafkaa_6s` uses a modified implementation of the subroutine *6S* (the acronym is for "Second Simulation of the Satellite Signal in the Solar Spectrum") by Vermote et al. to calculate the quantities $\rho_{\rm atm}, t_{\rm u}, t_{\rm d}$, and $\bar{s}$. Currently, there are continental, maritime, and urban aerosols available, as well as zero aerosol option. The aerosol models all assume 70% relative humidity, and each is made up of differing amounts of four optically different components: a dust-like component, an oceanic component, a water-soluble component, and a soot component. The fractions for the continental model are 0.70, 0.00, 0.29, and 0.01, respectively; the fractions for the maritime model are 0.00, 0.95, 0.05, 0.00, respectively; and finally, the fractions for the urban model are 0.17, 0.00, 0.61, and 0.22, respectively. The optical parameters for the four basic components, as well as the ratios indicated for the three specified types of aerosols, have been taken from the World Meteorological Organization report (WMO 1986).

6S is able to read files in order to use other aerosol models. This is disabled in the modified implementation used in `tafkaa_6s`.

The modified implementation of *6S* provides $\rho_{\rm atm}$, not $\rho_{\rm atm+sfc}$. That is, neither the direct beam nor the skylight reflected specularly off the ocean surface is accounted for. In "nice" conditions (favorable observing and solar geometry and little or no wind), the corrections provided by `tafkaa_6s` should be reasonable, even over the water.

An excellent paper discussing the specular reflection from an ocean surface is Mobley (1999).

4 Application to `tafkaa_tabular`

The principal difference between the tabular version of *Tafkaa* (i.e., `tafkaa_tabular`) and `tafkaa_6s` is the source of the quantities $\rho_{\rm atm+sfc}, t_{\rm u}, t_{\rm d}$, and $\bar{s}$. A computer program by Zia Ahmad (a somewhat modified version of that discussed in the article Ahmad & Fraser 1982) performs vector radiative transfer computations in order to calculate the desired quantities. The calculations are performed for a variety of solar and observational geometries, as well as for five general aerosol types[1] (each at five relative humidities and ten optical depths). Furthermore, the quantities were calculated at only a few specially selected wavelengths in atmospheric windows where the molecular (gas) absorption is negligible. The geometrical, wavelength, optical depth, and relative humidity grids may be found in Appendix B.

The tables were constructed precisely because it takes a very long time to calculate the aforementioned quantities, and this led to careful choices of the aerosol models, relative humidities, observational and solar angles, and optical depths where the calculations would be performed. In practice, linear interpolations are performed in the angular quantities in order to reduce the size of the table that needs to be searched in order to determine best-fit aerosol model. Finally, since the ocean surface is not completely flat (due to the wind), and the sky is not dark (due to atmospherically scattered light), it is necessary to calculate the above quantities over a realistic wind-blown ocean surface (W =2 m/s, 6 m/s, and 10 m/s). (Only $\rho_{\rm atm+sfc}$ depends on wind speed.)

We have calculated $\rho_{\rm atm}$ for a zero-reflectance Lambertian and assembled these quantities into the proper format for `tafkaa_tabular`, which allows us to use the same aerosol models over both the ocean and land.

The total number of tables currently needed for sea-level uses of `tafkaa_tabular` from above the atmosphere is 7: one each table of $t_{\rm u}, t_{\rm d}$, and $\bar{s}$; a table of $\rho_{\rm atm+sfc}$ for each of three velocities, appropriate for use over water; and a table of $\rho_{\rm atm+sfc}$ appropriate for use over land.

The wind speed is **not** used to calculate aerosol properties in any way, as it is for some aerosol models, such as those contained in *MODTRAN*.

As of the 2000 September version and later, tabular versions of `tafkaa_tabular` now contain tables for several sensor altitudes. In particular, we should be able to deal with sensor altitdues greater than 2.5 km. Any altitude above 84 km is considered to be "above the atmosphere" and all calculations at or above this altitude will use the same table entries. The tables that needed to be calculated were reflectance over land and at each of the three wind speeds over water, and the upwards transmission; each of them needed to be calculated at the new sensor altitudes. The list of sensor altitudes the tables were computed at is in Appendix B. For any other altitude, the various quantities are calculated (reflectance, upward transmission) via interpolation in pressure. While it functions, this should still be considered a work in progress.

4.1 Determining aerosol parameters

In order to determine the correct aerosol parameters, we assume that there is no water leaving radiance at particular wavelengths. Due to the strong absorption of light by water beyond about 0.75μm, this is true for most waters. (There may, however, be significant $L_{\rm w}$ out to $\sim 1\mu$m in waters with high amounts of suspended sediments.) In this case, and at the wavelengths chosen because the atmospheric absorption is negligible, the observed reflectance ($\rho_{\rm obs}$) is equal to the atmospheric reflectance ($\rho_{\rm atm+sfc}$). Thus, the aerosol type and optical depth are determined by matching the observed atmospheric reflectance to the calculated atmospheric reflectance at two or more wavelengths, and using a least squares determination in order to determine the best match.

[1] The aerosols used are desribed in more detail in Appendix E.

(Linear interpolations are performed on the aerosol optical depth[2], but not on relative humidity or between the general model types.) It is frequently necessary to use only the wavelengths $> 1\mu$m when the water is very turbid. [Or, more exactly, if in the upper few optical depths (say, $z < 2/a_{\rm w}$) the light at the wavelengths in question is subject to much greater scattering so that there is measurable water leaving radiance. Thus the water is not a dark surface, violating one of our assumptions.] Once the parameters of the best-fit aerosol model are determined, we can use the appropriate $\rho_{\rm atm}, t_{\rm u}, t_{\rm d}$, and $\bar{s}$ in order to atmospherically correct the observed reflectance, and thus determine the water leaving radiance.

Based on the above discussion, it is obvious why we cannot determine the aerosol type and optical depth over the land: the land is rarely dark enough at any particular wavelength in order for the assumption "the observed reflectance equals the atmospheric reflectance," to be true. [Technically, this assumption either requires the reflectivity of the surface to be 0, or the atmospheric reflectance be much greater than the reflected signal from a surface of non-zero reflectivity. We only use the first of these assumptions. If the calibration of the blue portion of the spectrum is very good, it may be possible to use the blue ($\lambda < 0.4\mu$m) in order to help determine the aerosol parameters since the path radiance dominates over all other signals at sufficiently small wavelengths.] This is one of the reasons that the current `tafkaa_6s` needs to be told the aerosol type and optical depth (or, equivalently, the visibility).

4.1.1 Aerosol properties and measurement error

Both measurement and discretization error come into play in the determination of aerosol properties. This is largely because the reflectance is fairly small in the portions of the spectrum that we must use; thus discretization errors are fractionally larger than they might otherwise be. Monte Carlo style numerical experiments show that in order to obtain the correct results, it is best to average several pixels, possibly as few as 20 (with about 3% measurement errors, discretization errors, and using the reflectance at four wavelengths to determine the aerosol properties). This is not currently implemented in `tafkaa_tabular`. Also note that if discretization errors are larger than the separation between models or larger than the measurement errors, averaging can't help.

5 Data Issues

5.1 `tafkaa_tabular` processing with only *VNIR*

If only the *VNIR* bands ($\lambda < 1\mu$m) are available for certain Open Ocean Mode observations, aerosol determination will be restricted to the bands available, the most likely candidates being the 0.75μm and 0.865μm bands.

- The first option will be to use just the 0.75μm and 0.865μm bands – the minimum needed in order to determine both the optical depth and aerosol type (assuming no binning of pixels). This should work reasonably well in areas where there is no silt or other material suspended in the topmost part of the ocean, that is, in truly open ocean situations.
- The second option is to use other available information in order to determine aerosol types. This may be possible if there are ground observations available (as there will be for certain scenes, right?)
- A third option is to determine the aerosol type and optical depth from a nearby scene where both *VNIR* and *SWIR* mode are being used.
- Finally, we could use just one band, say the 0.75μm band, in order to determine one of the parameters (optical depth or aerosol type) if the other quantity is independently available. *This option is not yet available.*

A few caveats of *VNIR* only observations:

- We will not be able to detect thin cirrus clouds.
- It will be difficult and/or impossible to distinguish clouds from coccolithophore bloom.

[2]It is important to note that the linear interpolation on aerosol optical depth is performed so that there are *only* 9 intervals spaced equally between any two neighboring τ grid values, so there are technically only 91 different values of aerosol optical depth that can be derived.

5.2 Differing view angles across a scene

Some sensors have wide fields of view, and we'd like to be able to apply more than just a nadir correction to the data. I've added some simple abilities to *Tafkaa* in order to deal with this issue. First, assume that the sensor is nadir viewing in the middle of the array, and that there is no roll, that the sensor follows a straight, level path, and that the cross-track is perpendicular to the sensor path. Then the view zenith angle is simply a function of the location of the pixel away from the center of the image, and the view zenith angle changes from one value to another at the center of the array. In order to calculate both the view zenith and view azimuth angle for all the pixels, *Tafkaa* needs the the pixel-to-pixel angular spacing and the heading of the sensor. *Tafkaa* then divides the scene into a number of regions and uses the average zenith angle of each region to determine the atmospheric correction for that region. The necessary keywords are `tafkaa_geometry`, `tafkaa_p2p_delta_theta` , and `tafkaa_sensor_heading`.

It should be noted that this is still an approximate solution. Most planes have some roll and pitch. Planes that fly in the lower atmosphere may have quite a bit of each. Some paths are neither level nor straight. And sometimes, for whatever reason, some pixels are added to (or subtracted from) the edge, which will destroy *Tafkaa* 's assumption that the center pixel is nadir-viewing. I'm open to suggestions for improvements. The big pitfall to using correct angular information for every pixel in the scene is that the absorption terms need to be recalculated and the scattering tables re-interpolated. Furthermore, any changes in heading tend to change the view azimuth angles, and that also requires a re-interpolation of the scattering tables.

Future versions of *Tafkaa* may use different methods: i.e., a "pixel pointing file" that would be able to tell *Tafkaa* the direction each pixel is viewing the ground. This is probably the easiset way to allow for both added or missing pixels, as well to allow for a mean-roll or mounting offset in the roll direction.

5.2.1 `tafkaa_tabular` COIS Scene Segmentation issues

1. In non-nodding viewing modes, with an orbital velocity of between 7.5 and 8km/s, observations of a 200km swath along the path takes only about 20-30s - neither the earth nor the sun moves appreciably in this time.

2. Discussions with Bo-Cai indicate that we have a tolerance of ≈ 2–3° in the angular quantities. Since *COIS* has $\approx 2.84^\circ$ field of view (or $\approx 1.42^\circ$ from the cross track from the center), problems are minimized in that direction.

3. 30 km $\times$ 30 km chunks of observation will be easy to deal with – the angles don't change much at all in a field of that size. The maximum size field for *Tafkaa* is on the order of 30km by 100km.

5.3 `tafkaa_tabular` Land-Water-Both Approach

In generating our transmittance tables, we co-incidentally also generated tables of atmospheric reflectance assuming a surface of zero reflectance. These tables are appropriate for use in atmospherically correcting observations over land, and they have the advantage of being calculated for the identical parameters (geometric and aerosol) as the atmospheric reflectance tables over the ocean. However, one still needs to determine the aerosol type and optical depth in order to use these new tables. Thus, the choices are as follows:

1. Determine the aerosol type and optical depth from water/ocean areas of the scene (i.e., `tafkaa_aerosol_method > 0`). Use the same aerosol model and optical depth over the water and land, but use the ocean tables over the ocean and the land tables over the land. Ocean and land can be discriminated via the previously generated *NDVI* land mask. *Note: There is limited availability of this function. See the notes about the* `tafkaa_aerosol_method` *keyword.*

2. Alternately the user may input the aerosol type and optical depth for the scene (i.e., `tafkaa_aerosol_method < 0`), thus `tafkaa_tabular` does not need to determine these quantities on-the-fly. Once again, the previously generated *NDVI* land mask is used in order to determine whether the table appropriate for ocean or land is used.

3. When determining the aerosol type over water in pixel-by-pixel mode (i.e., `tafkaa_aerosol_method = 0`), deciding the appropriate treatment over land is problematic. Thus, in pixel-by-pixel mode, the user is allowed to enter the aerosol type, relative humidity, and optical depth (at $\lambda = 0.550\mu$m) to apply over land pixels.

6 Running *Tafkaa*

6.1 Source Code

Each version of *Tafkaa* has been created from a single[3] source code that can be compiled on at least three platforms, and the versions share many identical modules. Currently, it has been compiled on a PC platform with the *MicroSoft Fortran PowerStation 4* compiler, on an SGI with the *MIPS Pro 7.3 Fortran 90* compiler, and on a Sun with *Sun Forte f95 version 6 update 2*. All attempts have been made to make the source code strictly ANSI[4]/ISO[5] Fortran 90[6] compliant. As a special note, it is guaranteed that this source code *will not* run when compiled using *Lahey* compilers. This unfortunate situation principally stems from the fact that the Fortran 90 Standard does not specify the layout of files that are opened as `access = direct, form = unformatted` . The compilers I currently use assume a strict binary file, without any other information in the file. *Lahey's* compiler, unfortunately for us, assumes that files opened with this method have an extra record at the beginning of the file that contains the record length of the file. In order to get the behavior we want using *Lahey's* compiler, we either need to insert such a record at the beginning of *all* binary files we attempt to read, or use the non-standard `transparent` directive that *Lahey* provides for this situation. In order to *not* add non-data records to binary files, and use strict ANSI/ISO standard Fortran 90, I cannot at this time use the *Lahey* compiler.

6.2 *Tafkaa* Inputs

The easiest way to run *Tafkaa* on either of the currently supported platforms is to use the *ENVI* interface[7] provided by J. Rhea & W. Snyder. Their graphical interface gathers the required information from the user, and prepares the image header file and the *Tafkaa* input file.

The *Tafkaa* input file is made up of many keywords. Except for `tafkaa_input_image_name`, the keywords may appear in either the input file or the image's *ENVI* header file. However, it is considered good form to include keywords that concern information required to read the data or concerning the actual image, to appear in the image's header file. Keywords specific to *Tafkaa* should then appear in the input file.

The keywords come in 6 basic types: integer, real, and string; integer array , real array, and string array. Array types are enclosed in curly braces, i.e., { }.

Please note that any lines that appear in the input file must have no more than 132 characters in the line; *lines longer than 132 characters will lead to some type of input errors, such as not finding a closing "}" to end processing of input array values.* Any values that must wrap around (such as long arrays) should have line breaks *after* a comma, after a left curly brace, or before a right curly brace. *While not stated in the ENVI documentation, this line-breaking rule follows the observed behavior of ENVI with the* `band names` *keyword.*

Certain other keywords have been defined, and may occur in the header file. The lists here do *not* include all these keywords. Only the keywords *required* by *Tafkaa* are listed below. *Tafkaa* will ignore other text in the file, and is supposed to ignore everything but the keywords listed below.

If a keyword is repeated in the input file and in the header file, the value in the input file is the value that will be used.

On the below pages, items in the `teletype` font style are meant to represent what you would actually expect to see or enter in a file. Any other text, in particular, text in parentheses, is a useful comment indicating restrictions on value choices, implied units, suggested reasonable values, etc.

[3] This is *not quite* true. A single file called `name_and_version.f90` differs on the two machines, usually only with either the text `PC` or `SGI`, although sometimes the `version_date` may differ.

[4] American National Standards Institute

[5] International Standards Organization

[6] ISO/IEC 1539: 1991

[7] I refer the reader to them or their documentation.

6.2.1 Input Image Header File

The following keywords should appear in the image's header file (the values are examples only). An example input image header file is listed in Appendix D.1.2.

- `samples = 614`
- `lines = 600`
- `bands = 224`
- `header offset = 0` (In bytes; *Tafkaa* requires this to be `0`.)
- `data type = 2` (2 = signed 2 byte integer, which *Tafkaa* requires)
- `interleave = bip` (`bip`, `bil`, or `bsq`.)
- `byte order = 1` (0 which is Least Significant Byte first, for PC/DEC; else 1 which is Most Significant Byte First)
- `wavelength = { 0.4121, ..., 2.5090}` (Center wavelength of band, in microns)
- In order to calculate the instrumental response function, one of the following two keywords needs to be provided. Either
  - `fwhm = { 0.009691, ..., 0.010030}` (microns, assuming Gaussian instrumental response function)

  or
  - `tafkaa_instrumental_response_file = complete_path_and_filename` (Filename with the complete path to the instrumental response file. The file should be as described in §C.)

The following items should also appear in the image's header file, since it is the logical location for them. However, while they are *ENVI* style keywords, they are not *ENVI* keywords, and will disappear from the header file if you edit the header file from within *ENVI* . Because of this, an argument can made for them to be put in the input file. Current practice is to include them in the image's header file, and remind the user to not edit the header file from within *ENVI* .

- `image_scale_factor= { 100., ..., 50.}` (Either 1 element if all `bands` elements have the same scale factor, or `bands` elements if they don't. We divide these numbers into the integers in the file by in order to obtain the physical, measured radiance in units of W m^{-2} sr^{-1} μm^{-1}.
- `image_center_date = {yyyy, mm, dd}` (GMT date, assuming standard civil Gregorian calendar)
- `image_center_time = {hh, mm, ss.ss}` (GMT time, $0 \leq$ `hh` < 24)
- `image_center_long = {ddd, mm, ss.sss}` (non-negative, degrees, minutes, seconds)
- `image_center_long_hem = W` (Either `W` or `E` of the Prime Meridian)
- `image_center_lat = {dd, mm, ss.sss}` (non-negative, degrees, minutes, seconds)
- `image_center_lat_hem = N` (Either `N` or `S` of the Equator)
- `image_center_zenith_ang = {dd, mm, ss.sss}` (non-negative, degrees, minutes, seconds; { 0., 0., 0.} implies nadir-viewing)
- `image_center_azimuth_ang = {dd, mm, ss.sss}` (non-negative, degrees, minutes, seconds, relative to north, clockwise)
- `sensor_altitude = 20.0` (kilometers above mean sea level)

6.2.2 *Tafkaa* Input File

The following items are specific to *Tafkaa* , and thus should appear in the *Tafkaa* input file. An example input file listed in Appendix D.1.1.

The defining directories, files, masks, and output format:

- `tafkaa_input_image_name = /u1/mmontes/coral.bip`
- `tafkaa_data_directory = /u1/mmontes/data/` (The directory that contains the high resolution exoatmospheric solar irradiance spectrum in units of W m^{-2} sr^{-1} μm^{-1}. The data is in a binary file named `sun_binary` that consists of reals and is expected to be in the native byte order. This directory also contains all the files containing the line absorption coefficients and the various tables needed by the tabular version.)
- `tafkaa_output_root_name = /u1/mmontes/testing` (The name, including directory, to which is appended various suffices in order to form the various output image and header file names.)
- `tafkaa_use_which_masks = { land }` (Select either `none` or one or more of `land`, `cirrus`, or `low cloud`, in any order. This tells *Tafkaa* which masks to utilize.)
- `mask_image_name = ./coral_test_mask.img` (The name of the mask image that has been previously produced; not necessary if `tafkaa_use_which_masks = { none }` .)
- `tafkaa_output_type = refl` (One of `refl, rrs, lgn, lg` or `aprefl`, which imply reflectance (ρ_g), remote-sensing reflectance (R_{rs}), normalized ground-leaving radiance ($[L_g]_N$), ground-leaving radiance (L_g), and apparent at-sensor reflectance (ρ_{obs}), respectively. *The* `lg` *option is not currently supported.* The apparent at-sensor reflectance (see Equation 5) does not perform any atmospheric correction at all, and the various atmospheric and aerosol keywords are not necessary for the input file. The apparent at-sensor reflectance is a useful diagnostic tool.)
- `tafkaa_output_scale_factor = 10000.` [In order to get to the relevant units, divide the integers in the output image by this number. The *best* output scale factor to use depends on the output quantity, and aesthetics. Let me define `tafkaa_output_scale_factor` $=\chi$, and the maximum value of the chosen quantity (ρ, R_{rs}, $[L_g]_N$, or L_g) will be represented as M. Since we are representing numbers as signed 2-byte integers, the maximum number we can have is $2^{15} - 1 = 32767$. Thus, usually we want $\chi \leq 32767/M$. Since $\rho \leq 1 = M$, in this case, $\chi_\rho <= 32767$. Aesthetics plays a role since many people prefer powers of 10, so a typical value for this case would be $\chi_\rho = 10000$. In the case of $R_{rs} = \rho/\pi$, $M = 1/\pi$, thus $\chi_{R_{rs}} \leq 32767\pi = 102940.57$. This would allow a nice aesthetic value of $\chi_{R_{rs}} = 100000$. $[L_g]_N = E_0 R_{rs} \leq 2000/\pi$ thus in this case $M \sim 640$. E_0 is wavelength dependent, and it peaks at about $0.48\mu m$; thus the resolution will be better in some bands than in others. This implies that $\chi_{[L_g]_N} \leq 51$. Finally, $L_g = [L_g]_N \mu_0 t_d$. Clearly, $\mu_0 \leq 1$ and $t_d \leq 1$. Away from the absorption bands, $t_d \sim 1$. μ_0 is known, and is usually not too close to unity. Thus, $L_w \leq 640/\mu_0$, so $\chi_{L_g} \leq 51/\mu_0$.]
- `tafkaa_ground_elevation = 0.00` (Average elevation of viewed surface in kilometers above sea-level.)
- `tafkaa_line_range = { start_line, stop_line}` (Optional; integer array. By default, all lines are analyzed if this keyword is absent. `y start` is respected: the values are assumed to be in the system where the first line is identified by the `y start` keyword in the *ENVI* header. Thus, the values in this keyword must be such that `y start` $\leq$ `start_line` $\leq$ `stop_line` $\leq$ `y start+lines-1`. The output files will have `y start = start_line` and `lines = stop_line-start_line+1`. **New**
- `tafkaa_sample_range = { start_sample, stop_sample}` (Optional; integer array. By default, all samples are analyzed if this keyword is absent. `x start` is respected: the values are assumed to be in the system where the first sample is identified with the `x start` keyword in the *ENVI* header. Thus, the values in this keyword must be such that `x start` $\leq$ `start_sample` $\leq$ `stop_sample` $\leq$ `x start+samples-1`. The output files will have `x start = start_sample` and `samples = stop_sample-start_sample+1`. **New**

View angle parameters, used to determine absorptive and scattering properties of the atmosphere:

- `tafakaa_geometry = 1` (Either `1` in order to use multiple view angles across a scene, or `0` in order to use `image_center_zenith_ang`. If this keyword is not present, `tafakaa_geometry = 0` is set.)

- `tafkaa_p2p_delta_theta = 0.8745e-3` (Used if `tafakaa_geometry = 1`; the cross-track pixel-to-pixel spacing, in radians.)

- `tafkaa_sensor_heading = 260.0` (Used if `tafakaa_geometry = 1`; the sensor heading in degrees from north.)

Parameters that describe the absorptive properties of the atmosphere:

- `tafkaa_atmo_model = tropical` (Choices are `tropical`, `mid latitude summer`, `mid latitude winter`, `subarctic summer`, `subarctic winter`, and `US Standard 1962`.)

- `tafkaa_atmo_gasses = { H2O, CO2, O3, N2O, CO, CH4, O2}` (One or more of `H2O`, `CO2`, `O3`, `N2O`, `CO`, `CH4`, `O2`, or `NO2`, in any order.)

- `tafkaa_atmo_ozone = 0.34` (Measured in units of atm-cm, and required if `O3` is selected in `tafkaa_atmo_gasses`. A online resource for ozone values is: `http://toms.gsfc.nasa.gov/-ozone/ozone.html` or `http://toms.gsfc.nasa.gov/teacher/ozone_overhead.html`.)

- `tafkaa_atmo_no2_scale = 1.1` (Multiplicative factor: multiplies $5{\times}10^{15}$ molecules NO_2; required if `NO2` is selected in `tafkaa_atmo_gasses`.)

- `tafkaa_atmo_clmwvap = 1.1` (Vertical column of water vapor, in centimeters. **Including a positive value for this keyword implies the next 5 keywords listed below are unnecessary, and will be ignored if included in the input file.** You *must* include `H2O` in the `tafkaa_atmo_gasses` list in order for this value to be used.) **Do not use this keyword if** `tafkaa_h2o_enter_inputs = 1 .`

- `tafkaa_h2o_enter_inputs = 1` [`1` (`0`) implies the user will (won't) enter the parameters for the water vapor bands, described in the next four parameters.]**Note 1: Only necessary when determining column water vapor pixel-by-pixel. Note 2: Do not use** `tafkaa_atmo_clmwvap` **keyword when** `tafkaa_h2o_enter_inputs = 1 .`

- `tafkaa_h2o_wl_set1 = {0.8650, 1.030, 0.945}` (Wavelengths in μm of the two adjacent windows and the center of the H_2O absorption band.)

- `tafkaa_h2o_nb_set1 = {3, 3, 5}` (Number of bands to use centered at each of the above wavelengths. The number of bands chosen depends on the particular water vapor that is chosen, and on both the spectral resolution and spectral spacing of the instrument.)

- `tafkaa_h2o_wl_set2 = {1.0650, 1.240, 1.140}` (Wavelengths in μm of the two adjacent windows and the center of the H_2O absorption band.)

- `tafkaa_h2o_nb_set2 = {3, 3, 5}` (Number of bands to use centered at each of the above wavelengths. The number of bands chosen depends on the particular water vapor that is chosen, and on both the spectral resolution and spectral spacing of the instrument.)

- `tafkaa_use_prev_atmo_trans = 1` [`1` (`0`) implies you do (don't) want to use the gas absorption results from a previous calculation.]

- `tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin` (If `tafkaa_use_prev_atmo-_trans = 1`, then you must set this value to the filename that contains the results from a previous calculation.)

Parameters that describe the the aerosol model and/or how the aerosol model(s) are chosen *for the tabular version.*

- `tafkaa_aerosol_method = 0` [`0` implies attempt a pixel-by-pixel solution; any negative value implies the user will enter an aerosol model described by the next four parameters; a value > 2 value implies use that number of pixels to determine the aerosol type, then apply it over the whole scene (*this last option is not yet available*).]

If `tafkaa_aerosol_method = 0`, then the following three parameters describe the model that will be applied over the land pixels (as described by the land mask) if they are supplied; if the following parameters are not supplied when `tafkaa_aerosol_method = 0`, then no models will be applied over the land and the land will be blacked out. If `tafkaa_aerosol_method < 0`, then this model will be applied over the whole scene. The choices are different for the 6S version.

- `tafkaa_aerosol_rh = 80%` (Choose one of `50%`, `70%`, `80%`, `90%`, or `98%`.)
- `tafkaa_aerosol_model = coastal-a` (Choose one of `maritime`, `coastal`, `coastal-a`, `tropospheric`, or `urban`.)
- `tafkaa_aerosol_tau550 = 0.1` (Choose a value in the range $0 \leq \tau_{550\ \mathrm{nm}} \leq 2.0$.)

If `tafkaa_aerosol_method = 1` or `tafkaa_aerosol_method = 2` then *Tafkaa* will determine the aerosol type and optical depth using a rectangular area previously selected by the user. `tafkaa_aerosol_method = 1` determines $\rho_{\mathrm{obs}}/T_{\mathrm{g}}$ for each pixel, averages over the rectangular region, and uses that quantity to determine the aerosol type and optical depth. `tafkaa_aerosol_method = 2` determine ρ_{obs} for each pixel, uses that to determine an average water-vapor value, then determines the aerosol model and type from $\overline{\rho_{\mathrm{obs}}}/T_{\mathrm{g}}^{\mathrm{rectangle}}$. The coordinates of the rectangular are are given in the keyword

- `tafkaa_aerosol_rectangular_region = {llx, lly, urx, ury}` (`llx` is the lower-left x-coordinate of the region; `lly` is the lower-left y-coordinate of the region; `urx` is the upper-right x-coordinate of the region; `ury` is the upper-right y-coordinate of the region. The coordinate should be measured assuming the `x start` and `y start` values of the header. If either `x start` or `y start` is not present, the value is assumed to be unity. If `tafkaa_geometry = 1`, then it must be the case that `sample_range(1)` $\leq$ `llx` $\leq$ `urx` $\leq$ `sample_range(2)`.)

When `tafkaa_aerosol_method`≥ 0, *Tafkaa* attempts to determine an aerosol model. We've recently added an urban aerosol model to our tables. Unfortunately, there are some situations where *Tafkaa* chooses the urban aerosol in places where we know it can't be present – apparently there is a slight degeneracy between some of the higher optical depth urban aerosol models and some of the lower optical depth non-urban models. In order to deal with this, we've instituted a few new keywords to exclude choosing certain aerosols or optical depths. The following lists to exclude are "or"ed together.

- `tafkaa_exclude_aerosol_models = { urban }` (A string array accepting the names of aerosol models you wish to exclude. If this keyword is not listed, then no aerosol models are excluded.) **New**
- `tafkaa_exclude_aerosol_rh = { 80%, 50% }` (A string array accepting a list of relative humidities you with to exclude. If this keyword is not listed, then no aerosol relative humidities are excluded.) **New**

On output, the determined aerosol type and optical depth will be reported in the history section of the output header files.

- `tafkaa_wind_speed = 2` (*Not* used for interpolation, effectively chooses which table to use: the 2 m/s, 6m/s, or 10 m/s table. The units are m/s.)
- `tafkaa_aerosol_weights = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}` (Used only when `tafkaa_aerosol_method` $\geq$ `0`; a set of 14 reals, associated with the wavelengths 0.39, 0.41, 0.44, 0.47, 0.51, 0.55, 0.61, 0.67, 0.75, 0.865, 1.04, 1.24, 1.64, and 2.25 μm, respectively. Chooses which parts of the spectra to use when determining the aerosol model, relative humidity, and optical depth.)
- `tafkaa_interp_sensor_altitude = true` (Possible values are `true` and `false`. Honestly, though, anything except `true` implies `false`. Use this when `sensor_altitude` is anything inside the atmosphere. A value of `true` implies that we'll use linear interpolation in pressure to determine the sensor values appropriate for the sensor height. A value of `false` implies that we'll use the nearest altitude. This may be used when you fly at one of the altitudes the tables have been generated at, and you don't want to interpolate. Anytime you **want** to **New keyword**

interpolate, set this keyword to **true**. The current altitude values of the aerosol tables are: 84.0, 22.0, 16.0, 10.5, 8.0, 5.5, 4.0, 2.59, and 2.50 kilometers.)

Parameters that describe the aerosol model *for the 6S version*:

- `tafkaa_aerosol_model = maritime` (Choose one of `none`, `continental`, `maritime`, or `urban`.)
- `tafkaa_aerosol_visibility = 50.0` (The visibility in kilometers; put 0 here if you would rather use `tafkaa_aerosol_tau550`.)
- `tafkaa_aerosol_tau550 = 0.1` (Used only when `tafkaa_aerosol_visibility = 0.0`; a reasonable range $0 \leq \tau_{550\ \mathrm{nm}} \leq 2.0$. Note: Check the range!)

6.3 Output Files

An output image file, along with an output data products file, are always created; their header files will contain a "history" section that indicates that version and the settings that were used to to generate them. Additionally, there are two additional ASCII files and possibly a binary file that may be output. All the output files will be described below.

The principal output image file and its header have the following names:

- `tafkaa_output_root_name`_tafkaa_refl.hdr, and
- `tafkaa_output_root_name`_tafkaa_refl.img

if reflectance was chosen as the output type. When remote-sensing reflectance is the output type, 'refl' will be replaced with 'R_rs'. When normalized ground-leaving radiance is selected, 'refl' will be replaced with 'NLsf'. Finally, when ground-leaving radiance is selected, 'refl' will be replaced with 'Lsfc'. The file will have the same storage order and number of bands as the input observed radiance file had, and it will always be in the machine's native byte-order in two-byte integers. The values of `samples`, `lines`, `x start`, and `y start` in the output file depend on the values of the input parameters `tafkaa_line_range` and `tafkaa_sample_range`. This will, of course, be reflected in the header file.

The products file and its header have the following names:

- `tafkaa_output_root_name`_tafkaa_prod.hdr, and
- `tafkaa_output_root_name`_tafkaa_prod.img .

This file is always made up of two-byte integers in the machine's native byte-order in BSQ sort-order, with the same number of samples and lines as the principle output image (above). The first plane is always the derived water-vapor plane. When the *tabular* version is used and `tafkaa_aerosol_method = 0`, there will be additional planes that contain $\tau_{\mathrm{aerosol\ 550\ nm}}$, relative humidity, and the aerosol model. There may also be a plane with some sort of measure of the quality of the fit, especially in the early development of *Tafkaa* . All this information, and the appropriate scale factors, are, of course, provided in the header file.

The two ASCII files that are output are:

- `tafkaa_output_root_name`_tafkaa_solar_irr.asc, and
- `tafkaa_output_root_name`_tafkaa_vaplib.asc .

The two columns in the first file are the wavelength (in μm) and the top-of-the-atmosphere solar irradiance (in W m^{-2} μm^{-1}). The second file contains a lot of information about the tables used to calculate the derived water vapor.

Finally, if `tafkaa_use_prev_atmo_trans = 0` or if `tafkaa_use_prev_atmo_trans = 1` and `tafkaa_prev_atmo_trans_file` is not appropriate for the atmospheric parameters and geometry that the user indicates, a binary file called

- `tafkaa_output_root_name`_tafkaa.vapbin

will be written for the atmospheric parameters specified by the user. This file is valuable if the user is experimenting with many different aerosol types, but the geometry (date, time, view angles, and solar angles) and atmospheric parameters are constant.

6.4 Invocation

To invoke or execute *Tafkaa* , it is best to feed it the input file via a "standard input redirect," which is allowed under both the command prompt window on a PC, and under most shells available on SGIs. Thus, if I symbolize the prompt as `prompt>` , the following is valid on both PCs and under `csh` & `tcsh` on SGIs:
`prompt> path-to-tafkaa < tafkaa_input_file`.
It is always safest to use the full path to both the *Tafkaa* executable and the *Tafkaa* input file.

7 Other Issues

1. Will our commercial partners need atmospheric corrections over elevated features? Currently this is available only in the `tafkaa_6s`. Even our land tables are currently only at sea-level. Zia Ahmad has updated the code to allow above sea-level ground; how many tables will we need to create in that case?

2. On-the-fly atmospheric correction may not always be possible: fronts may occur in the scene, aerosol types change over the scene (especially near urban areas?). How will the aerosol type and optical depth be determined automatically in scenes over land? In other words, only a 0th order atmospheric correction may be available on-the-fly. How does this affect other products that rely on atmospherically corrected data?

3. Size of scene for non-nadir viewing angles: more ground is observed the further off axis one observes; with $\theta = 30^\circ$, the field of view is somewhat more than 40km (instead of the 30km at when observing at nadir).

4. For a perfect sphere the size of the earth, a 30km scene has a 20m change in elevation compared to a flat plane, when viewed at nadir. (The center of the scene is 20m above a flat plane going through the edges of the scene.)

5. How to handle stratospheric aerosols from volcanic eruptions, etc?

7.1 Miscellaneous

This is mostly comments to remind me to insert the following items coherently elsewhere in the document.

1. The approximations, assumptions, and tables have been calculated to keep the errors in reflectance $\lesssim 0.001$.

2. Wavelength range of applicability of *Tafkaa* : *Tafkaa* should work well over the range $0.370\mu\text{m} < \lambda < 2.55\mu\text{m}$.

3. List the aerosol model parameters for the models used to generate the tables.

8 Known Bugs and Problems

On 32-bit systems (such as PCs) there is no way to address files that are larger than $2^{31} - 1 = 2147483647$ bytes, one byte less than 2GB (where k implies $\times 1024$, M implies $\times 1024^2$, and G implies $\times 1024^3$). This is because the addressing uses signed 32-bit integers. SGIs do not have this problem since they use signed 64-bit integers, which implies a maximum file size of $2^{63} - 1$ bytes, one byte less than $8 \times 1024^6 = 8$EB (E is the standard abbreviation for the metric prefix for "Exa", used for 1000^6; continuing previous usage in the computer world, 1 EB= 1 ExaByte = 1024^6 bytes).

Tafkaa cannot fix bad data. Using fully absorbed transitions to determine the amount of water vapor is dangerous, in the sense that you'll get bizarre results. So is using transitions that are not fully calibrated and/or transitions where *negative* value of radiance occur. In all these cases the retrieved value of water vapor will be wrong for the particular bad pixels, and so all the water transitions in the spectra of those pixels will be corrected incorrectly– i.e, they'll be mis-corrected. Perhaps *Tafkaa* should write out another plane of information to note this problem?

9 *Tafkaa* wish list

- Gracefully handle blank pixels, just like the new version of *Mask*.
- Use a pointing file for the view angles instead of calculating them.
- New generic spectral response function that is non-gaussian but the same shape for all the bands. Need to think about this.
- Gracefully handle bad data yielding bad water vapor values.
- Ability to read 4-byte integer and reals, instead of just two-byte integers.
- Ability to write output as reals, etc.

Acknowledgments

Tafkaa was generated using many pieces of B.-C. Gao's *ATREM*, including most of the geometry, solar-spectral, and water vapor subroutines, and it maintains the overall algorithms of *ATREM* created by Bo-Cai. All of the subroutines have been rewritten by converting them to Fortran 90, and many incorporate modifications that allow them to execute much more quickly. This document was typeset using LaTeX.

Bill Snyder has probably run *Tafkaa* more than anyone else, and he's found a number of bugs– all of which have been fixed. Thanks Bill!

References

For further information about *Tafkaa* , *6S*, and atmospheric correction, please consult the following papers:

- Ahmad, Z., and Fraser, R. S., *An Iterative Radiative Transfer Code for Ocean-Atmosphere Systems*, J. Atmos. Sci., 39, 656, 1982.
- Fraser, R. S., Mattoo, S., Yeh, E.-N., and McClain, C. R., *Algorithm for Atmospheric and glint correction of satellite measurements of ocean pigment*, J. Geo. Res., Vol. 102, No. D14, pp 17,107–17118, 1997.
- Gao, B.-C., and C. O. Davis, *Development of an operational algorithm for removing atmospheric effects from HYDICE and HSI data*, in SPIE'96 Conference Proceedings, Vol. 2819, 45-55, 1996.
- Gao, B.-C., and A. F. H. Goetz, *Column atmospheric water vapor and vegetation liquid water retrievals from airborne imaging spectrometer data*, J. Geophys. Res., 95, 3549-3564, 1990.
- Gao, B.-C., K. Heidebrecht, and A. F. H. Goetz, *Atmospheric Removal Program (ATREM) Users Guide, Version 3.0*
- Gao, B.-C., K. Heidebrecht, and A. F. H. Goetz, *Derivation of scaled surface reflectances from AVIRIS data*, Remote Sens. Env., 44,165-178, 1993.
- Gao, B.-C., M. J. Montes, Z. Ahmad, & C. O. Davis 2000, *An Atmospheric Correction Algorithm for Hyperspectral Remote Sensing of Ocean Color from Space*, Applied Optics, 39, 887.
- Goetz, A. F. H., and M. Herring, *The high resolution imaging spectrometer (HIRIS) for Eos*, IEEE Trans. Geosci. Remote Sens., 27, 136-144, 1989.
- Goetz, A. F. H., G. Vane, J. Solomon, and B. N. Rock, *Imaging spectrometry for Earth remote sensing*, Science, 228, 1147-1153,1985

- Green, R. O., and B.-C. Gao, *A Proposed Update to the Solar Irradiance Spectrum used in LOWTRAN and MODTRAN*, in Summaries of the Fourth Annual JPL Airborne Geoscience Workshop, October 25-29, (Editor, R. O. Green), JPL Publ. 93-26, Vol. 1, pp. 81-84, Jet Propul. Lab, Pasadena, Calif., 1993.

- Kneizys, F. X., E. P. Shettle, L. W. Abreu, J. H. Chetwynd, G. P. Anderson, W. O. Gallery, J. E. A. Selby, and S. A. Clough, Users to LOWTRAN7, AFGL-TR-8-0177, Air Force Geophys. Lab., Bedford, Mass., 1988.

- Iqbal, M., An Introduction To Solar Radiation, Academic, San Diego, Calif., 1983.

- Malkmus, W., *Random Lorentz band model with exponential-tailed S line intensity distribution function*, J. Opt. Soc. Am., 57, 323-329,1967

- Mobley, C. D. 1999, *Estimation of the remote-sensing reflectance from above surface measurements*, Applied Optics, 38, 7442.

- Montes, M. J., B.-C. Gao, and C. O. Davis 2001, *A new algorithm for atmospheric correction of hyperspectral remote sensing data*, Proc. SPIE, 4383, 23-30.

- Solomon et al. *On the role of nitrogen dioxide in the absorption of solar radiation*, Journal of Geophysical Research, V.104, 12047-12058, 1999.

- Shettle, E. P. & Fenn, R. W., *Models for the Aerosols of the Lower Atmosphere and the Effects of Humidity Variations on Their Optical Properties*, AFGL-TR-79-0214

- Tanre, D., C. Deroo, P. Duhaut, M. Herman, J. J. Morcrette, J. Perbos, and P. Y. Deschamps, *Description of a computer code to simulate the satellite signal in the solar spectrum: the 5S code*, Int. J. Remote Sens., 11, 659-668, 1990.

- Tanre, D., C. Deroo, P. Duhaut, M. Herman, J. J. Morcrette, J. Perbos, and P. Y. Deschamps, *Simulation of the satellite signal in the solar spectrum (5S), Users' Guide* (U. S. T. De Lille, 59655 Villeneu d'Ascq, France: Laboratoire d'Optique Atmospherique), 1986.

- Vane, G., R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter, *The Airborne Visible/Infrared Imaging Spectrometer*, Sens. Env., 44, 127-143, 1993.

- Vane, G. (Ed), *Airborne visible/infrared imaging spectrometer (AVIRIS)*, JPL Publ. 87-38, Jet Propul. Lab, Pasadena, Calif., 1987.

- Vermote, E., D. Tanre, J. L. Deuze, M. Herman, and J. J. Morcrette, *Second simulation of the satellite signal in the solar spectrum (6S), 6S User's Guide Version 1*, NASA-GSFC, Greenbelt, Maryland, 134 pages, 1994.

- Vermote, E., D. Tanre, J. L. Deuze, M. Herman, and J. J. Morcrette, *Second Simulation of the Satellite Signal in the Solar Spectrum, 6S: An Overview*, IEEE Trans. Geosci. Remote Sen., 35, 675, 1997.

- WMO (World Meteorological Organization), 1986, *A preliminary cloudless standard atmosphere for radiation computation.* World Climatic Program, WCP 112, WMO/TD No. 24.

A Glossary

θ, ϕ	=	polar and azimuth angles of the line of sight at the sensor
θ_0, ϕ_0	=	polar and azimuth angles of the direct sunlight
λ	=	wavelength
W	=	surface wind speed
C	=	the set of parameters describing all in-water processes
τ_a	=	aerosol optical depth and model
L_t	=	total observed radiance at satellite
L_0	=	radiance above the atmosphere if the radiance just above the sea surface was 0
L_sfc	=	radiance of direct and diffuse light specularly reflected off the sea surface
L_g	=	radiance reflected of from the target (assumed to be Lambertian)
L_w	=	radiance of water leaving light scattered from beneath the surface and penetrating it (assumed to be Lambertian)
t	=	transmission through the atmosphere of L_sfc
t'	=	transmission through the atmosphere of L_g
L_sa	=	$L_0 + L_\mathrm{sfc} t$
T_g	=	the transmission due to the absorptive processes in the gas
L'_sa	=	$L_\mathrm{sa}/T_\mathrm{g}$
t'_u	=	effective upward transmission
t_u	=	t'/T_g is the upward transmission due to scattering where $t_\mathrm{u} = t_\mathrm{u}(\lambda, \tau_\mathrm{a}, \theta)$
t_d	=	$t_\mathrm{d}(\lambda, \tau_\mathrm{a}, \theta_0)$ is the downward transmittance through the atmosphere to the target
μ_0	=	$\cos\theta_0$
E_0	=	the solar spectral irradiance at the top of the atmosphere
ρ_obs	=	observed reflectance
$\rho_\mathrm{atm+sfc}$	=	reflectance due to atmospheric scattering and direct sunlight & diffuse skylight reflection off of the rough ocean surface; this is a tabulated quantity.
ρ_g	=	reflectance of ground target; could be water (assumed Lambertian)
ρ_w	=	the water leaving reflectance (assumed to be Lambertian)
$\bar{s}$	=	$\bar{s}(\lambda, \tau_\mathrm{a})$ is the average reflectivity at the base of the atmosphere; that is, the fraction of upwelling light that is reflected back to the base
$NDVI$	$\equiv$	$\dfrac{\rho_\mathrm{obs}(0.86\mu\mathrm{m}) - \rho_\mathrm{obs}(0.66\mu\mathrm{m})}{\rho_\mathrm{obs}(0.86\mu\mathrm{m}) + \rho_\mathrm{obs}(0.66\mu\mathrm{m})}$ "Normalized Difference Vegetative Index"
$NDVI$	$>$	0.05 implies "Land"
R_rs	=	remote-sensing reflectance
	=	ρ_g/π
$[L_\mathrm{g}]_N$	=	normalized ground-leaving radiance reflected of the target (assumed to be Lambertian)
	=	$E_0 R_\mathrm{rs}$

B Parameters for the Tables

The tables used in the tabular version of *Tafkaa* are stored in the following grid:

- Wavelengths (μm): 0.39, 0.41, 0.44, 0.47,0.51, 0.55, 0.61, 0.67, 0.75, 0.865, 1.04, 1.24, 1.64, 2.25
- View Zenith Angle (degrees): 0.0, 1.5, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0, 54.0, 60.0, 66.0, 72.0, 78.0, 84.0, 88.0

- Relative Azimuth Angle (degrees): 0.0, 12.0, 24.0, 36.0, 48.0, 60.0, 72.0, 84.0, 90.0, 96.0, 108.0, 120.0, 132.0, 144.0, 156.0, 168.0, 180.0
- Solar Zenith Angle (degrees): 1.5, 12.0, 24.0, 36.0, 48.0, 54.0, 60.0, 66.0, 72.0
- Aerosol optical depth at 0.55μm: 0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.3, 1.6, 2.0
- Aerosol Relative Humidity[8] (%): 50, 70, 80, 90, 98
- Aerosol Names[8]: Maritime, Coastal, Coastal-a, Tropospheric, Urban
- Aerosol Table Altitudes (km): 84.0, 22.0, 16.0, 10.5, 8.0, 5.5, 4.0, 2.59, 2.50

C The Instrumental Response Function Input File

The typical convolution takes, for example, the transmission on some high-resolution grid (labeled A) to some lower resolution grid (labeled B), such that

$$T_B(\lambda) = \int_{\lambda'_{\rm min}}^{\lambda'_{\rm max}} T_A(\lambda') F(\lambda, \lambda') d\lambda' \ , \tag{11}$$

with the restriction that

$$\int_{\lambda'_{\rm min}}^{\lambda'_{\rm max}} F(\lambda, \lambda') d\lambda' = 1 \ , \tag{12}$$

and assuming that $F(\lambda, \lambda')$ is nonzero only in the range $\lambda'_{\rm min} \leq \lambda' \leq \lambda'_{\rm max}$. In our case, the higher resolution grid has a range of 0.3 – 3.1μm, has a spacing of 0.1 nm, and a FWHM of 0.2 nm; there are a total of 28,001 points in this grid. The instrumental response function that we require relates the observed grid to the aforementioned high resolution grid. For our situation of discretized grids, we have a situation that may be written as

$$T_B(\lambda_j) = \sum_{k=k_{\rm min}}^{k_{\rm max}} T_A(k) F(\lambda_j, k) \Delta\lambda'. \tag{13}$$

Thus, the information about the instrumental response function that needs to be tabulated is the product $F(\lambda_j, k)\Delta\lambda'$ for each observed wavelength λ_j. I try to do this using minimal storage space in the manner described below.

The processing of the IRF input file is done by the subroutine `read_instrumental_response_file` in the file `std_to_instr.f90`. The file is opened as a direct access unformatted file, thus it is a binary file with no header records.

It is first opened with a standard real or integer record length (this assumes that default reals and integers have the same length, such as 4 bytes on the machines currently used). The first record is then the typical record length of the file. The file is closed and re-opened with this record length. Because of this, the first record must be padded (usually with zeroes). Thus, if the record length is `nrecl`, the first record consists of the number `nrecl` followed by `nrecl-1` zeroes.

There then follow `nobs` records, one for each of the observed wavelengths. The statement that reads the `nobs` records is

```
do j=1,nobs
   read(unit=file_num,rec=j+1)initial_index(j),ncvtot(j),finstr(:,j)
end do
```

where `inital_index` corresponds to $k_{\rm min}$ above on the internal intermediate grid described above; `ncvtot(j)` is then equal to the number of non-zero elements of the instrumental response function, and as written in the discretized form above `ncvtot(j)`$= k_{\rm max} - k_{\rm min} + 1$. Finally, the instrumental response function is then `finstr(:,j)`$= F(\lambda_j, k)\Delta\lambda'$, where the first value is the first non-zero element. The record should be padded with zeroes to fill the the record up to the standard record length. Thus, the record length should be chosen as $2 + maximum(1 + k_{\rm max} - k_{\rm min})$.

[8] A discussion the aerosol properties may be found in Appendix E.

D Example Header Files

I've decided to include full examples of input files, input image header files, and output header files. About the only difference between the files as shown here and the files as they exist on the computer is the necessary existence of page breaks when the files are displayed as I have chosen to display them here. There are no page breaks in the input files on the computer.

D.1 Input Files

D.1.1 *Tafkaa* Input File

```
tafkaa_input_image_name = ./coral.bip
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_root_name = ./coral_test
tafkaa_output_type = refl
tafkaa_output_scale_factor = 10000
tafkaa_ground_elevation = 0.00
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O, CO2, O3, N2O, CO, CH4, O2}
tafkaa_atmo_ozone = 0.34
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = {0.8650, 1.030, 0.945}
tafkaa_h2o_nb_set1 = {3, 3, 5}
tafkaa_h2o_wl_set2 = {1.0650, 1.240, 1.140}
tafkaa_h2o_nb_set2 = {3, 3, 5}
tafkaa_aerosol_method = 0
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_tau550 = 0.1
tafkaa_aerosol_weights = {0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.}
tafkaa_wind_speed = 2
```

D.1.2 *ENVI* Style Input Radiance Image Header File

Based on the input file in Appendix D.1.1, the name of the header file below should be `./coral.hdr`. On SGIs, however, if that name is not found, *Tafkaa* will also search for the name `./coral.bip.hdr`. As can be seen from the history portion of the file below, it was generated on PC using W. Snyder's header population widget from his *ENVI* interface routines. Additionally, I have limited the `wavelength` and `fwhm` entries to only ≤ 72 characters wide so they can fit the page; originally, each line in those entries was 100 characters wide.

```
ENVI
description = {
File Resize Result, x resize factor: 1.0000, y resize factor: 1.0000. [Thu
Sep 11 17:21:43 1997]}
samples = 614
lines = 600
bands = 224
header offset = 0
interleave = bip
data type = 2
byte order = 1
file type = ENVI Standard
sensor type = AVIRIS
y start = 298
image_unscaled_units = W/m2/micron/ster
```

```
image_center_date = {1996, 3, 23}
image_center_time = {19, 44,   29.000}
image_center_long = {81, 47,   54.000}
image_center_long_hem = W
image_center_lat = {24, 36,   44.000}
image_center_lat_hem = N
image_center_zenith_ang = {0, 0,    0.000}
image_center_azimuth_ang = {0, 0,    0.000}
sensor_altitude = 19.768000
image_scale_factor = { 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100}
wavelength = { 0.369850, 0.379690, 0.389530, 0.399370, 0.409210, 0.419060, 0.428910,
0.438760, 0.448610, 0.458460, 0.468310, 0.478170, 0.488020, 0.497880, 0.507740, 0.517600,
0.527470, 0.537330, 0.547200, 0.557070, 0.566940, 0.576810, 0.586680, 0.596560, 0.606430,
0.616310, 0.626190, 0.636070, 0.645960, 0.655840, 0.665730, 0.675620, 0.664140, 0.673720,
0.683300, 0.692880, 0.702460, 0.712040, 0.721630, 0.731210, 0.740800, 0.750380, 0.759970,
0.769560, 0.779140, 0.788730, 0.798320, 0.807910, 0.817500, 0.827090, 0.836680, 0.846280,
0.855870, 0.865470, 0.875060, 0.884660, 0.894250, 0.903850, 0.913450, 0.923050, 0.932640,
0.942240, 0.951850, 0.961450, 0.971050, 0.980650, 0.990250, 0.999860, 1.009460, 1.019070,
1.028680, 1.038280, 1.047890, 1.057500, 1.067110, 1.076720, 1.086330, 1.095940, 1.105550,
1.115160, 1.124780, 1.134390, 1.144010, 1.153620, 1.163240, 1.172860, 1.182470, 1.192090,
1.201710, 1.211330, 1.220950, 1.230570, 1.240190, 1.249820, 1.259440, 1.269060, 1.254320,
1.264280, 1.274240, 1.284200, 1.294150, 1.304110, 1.314070, 1.324030, 1.333990, 1.343950,
1.353910, 1.363870, 1.373820, 1.383780, 1.393740, 1.403700, 1.413660, 1.423610, 1.433570,
1.443530, 1.453490, 1.463440, 1.473400, 1.483360, 1.493310, 1.503270, 1.513230, 1.523180,
1.533140, 1.543100, 1.553050, 1.563010, 1.572970, 1.582920, 1.592880, 1.602830, 1.612790,
1.622740, 1.632700, 1.642650, 1.652610, 1.662560, 1.672520, 1.682470, 1.692430, 1.702380,
1.712340, 1.722290, 1.732250, 1.742200, 1.752150, 1.762110, 1.772060, 1.782020, 1.791970,
1.801920, 1.811880, 1.821830, 1.831780, 1.841730, 1.851690, 1.861640, 1.871590, 1.881540,
1.878300, 1.888340, 1.898390, 1.908430, 1.918460, 1.928500, 1.938530, 1.948560, 1.958590,
1.968620, 1.978640, 1.988670, 1.998690, 2.008700, 2.018720, 2.028730, 2.038740, 2.048750,
2.058760, 2.068760, 2.078770, 2.088770, 2.098770, 2.108760, 2.118760, 2.128750, 2.138740,
2.148720, 2.158710, 2.168690, 2.178670, 2.188650, 2.198630, 2.208600, 2.218570, 2.228540,
2.238510, 2.248480, 2.258440, 2.268400, 2.278360, 2.288320, 2.298270, 2.308220, 2.318170,
2.328120, 2.338070, 2.348010, 2.357950, 2.367890, 2.377830, 2.387760, 2.397690, 2.407620,
2.417550, 2.427480, 2.437400, 2.447320, 2.457240, 2.467160, 2.477080, 2.486990, 2.496900,
2.506810}
fwhm = { 0.009610, 0.009580, 0.009550, 0.009530, 0.009500, 0.009480, 0.009460, 0.009440,
0.009420, 0.009400, 0.009380, 0.009370, 0.009350, 0.009340, 0.009330, 0.009310, 0.009300,
0.009290, 0.009280, 0.009280, 0.009270, 0.009260, 0.009260, 0.009260, 0.009250, 0.009250,
0.009250, 0.009250, 0.009260, 0.009260, 0.009260, 0.009270, 0.008300, 0.008310, 0.008330,
0.008340, 0.008350, 0.008360, 0.008370, 0.008390, 0.008400, 0.008410, 0.008420, 0.008430,
0.008440, 0.008450, 0.008460, 0.008460, 0.008470, 0.008480, 0.008490, 0.008500, 0.008500,
0.008510, 0.008520, 0.008530, 0.008530, 0.008540, 0.008540, 0.008550, 0.008550, 0.008560,
0.008560, 0.008570, 0.008570, 0.008570, 0.008580, 0.008580, 0.008580, 0.008580, 0.008590,
0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590,
0.008590, 0.008590, 0.008590, 0.008590, 0.008580, 0.008580, 0.008580, 0.008580, 0.008570,
0.008570, 0.008570, 0.008560, 0.008560, 0.008550, 0.008550, 0.008540, 0.010860, 0.010870,
0.010880, 0.010890, 0.010890, 0.010900, 0.010900, 0.010910, 0.010910, 0.010910, 0.010920,
```

```
0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010910,
0.010910, 0.010910, 0.010900, 0.010900, 0.010890, 0.010880, 0.010880, 0.010870, 0.010860,
0.010850, 0.010840, 0.010830, 0.010820, 0.010810, 0.010800, 0.010790, 0.010770, 0.010760,
0.010750, 0.010730, 0.010720, 0.010700, 0.010690, 0.010670, 0.010650, 0.010630, 0.010610,
0.010590, 0.010570, 0.010550, 0.010530, 0.010510, 0.010490, 0.010470, 0.010440, 0.010420,
0.010390, 0.010370, 0.010340, 0.010320, 0.010290, 0.010260, 0.010230, 0.010210, 0.011180,
0.011160, 0.011140, 0.011130, 0.011110, 0.011090, 0.011080, 0.011060, 0.011040, 0.011020,
0.011010, 0.010990, 0.010970, 0.010950, 0.010940, 0.010920, 0.010900, 0.010880, 0.010870,
0.010850, 0.010830, 0.010810, 0.010800, 0.010780, 0.010760, 0.010740, 0.010720, 0.010710,
0.010690, 0.010670, 0.010650, 0.010630, 0.010620, 0.010600, 0.010580, 0.010560, 0.010540,
0.010520, 0.010500, 0.010490, 0.010470, 0.010450, 0.010430, 0.010410, 0.010390, 0.010370,
0.010350, 0.010340, 0.010320, 0.010300, 0.010280, 0.010260, 0.010240, 0.010220, 0.010200,
0.010180, 0.010160, 0.010140, 0.010120, 0.010110, 0.010090, 0.010070, 0.010050, 0.010030}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

history = ends
```

D.2 Output Header Files

D.2.1 *Tafkaa* Output Header File

Based on the above files, the name of the *Tafkaa* output image will be `./coral_test_tafkaa_refl.img` and the image header file will be `./coral_test_tafkaa_refl.hdr`.

```
ENVI
description = {
  Image data cube: surface  reflectance (dimensionless)x10000.0
  derived from tafkaa (tabular);
  measurement date: 1996-03-23; measurement time: 19:44:29.000 GMT
  latitude:  24deg 36m 44.000s N ; longitude:  81deg 47m 54.000s W
  view zenith angle:   0deg  0m  0.000s ;
  relative azimuth angle:   0deg  0m  0.000s ;
  atmospheric model: tropical           ;
  absorption includes : H_2O CO_2 O_3 N_2O CO CH_4 O_2;
  ozone amount:  0.340 atm-cm ;
  windspeed:   2.00 m/s;
  weights for determining aerosols:
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00 1.00 1.00 }
samples = 614
lines   = 600
bands   = 224
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bip
```

```
byte order = 1
sensor type = AVIRIS
image_center_date = { 1996, 03, 23}
image_center_time = { 19., 44., 29.000}
image_center_lat = { 24., 36., 44.000}
image_center_lat_hem = N
image_center_long = {  81., 47., 54.000}
image_center_long_hem = W
image_center_zenith_ang = {  0.,  0.,  0.000}
image_center_azimuth_ang = {   0.,  0.,  0.000}
sensor_altitude =    19.768
image_scale_factor = {   10000.0000}
image_unscaled_units = { dimensionless}
wavelength = {
 0.36985, 0.37969, 0.38953, 0.39937, 0.40921, 0.41906, 0.42891, 0.43876, 0.44861,
 0.45846, 0.46831, 0.47817, 0.48802, 0.49788, 0.50774, 0.51760, 0.52747, 0.53733,
 0.54720, 0.55707, 0.56694, 0.57681, 0.58668, 0.59656, 0.60643, 0.61631, 0.62619,
 0.63607, 0.64596, 0.65584, 0.66573, 0.67562, 0.66414, 0.67372, 0.68330, 0.69288,
 0.70246, 0.71204, 0.72163, 0.73121, 0.74080, 0.75038, 0.75997, 0.76956, 0.77914,
 0.78873, 0.79832, 0.80791, 0.81750, 0.82709, 0.83668, 0.84628, 0.85587, 0.86547,
 0.87506, 0.88466, 0.89425, 0.90385, 0.91345, 0.92305, 0.93264, 0.94224, 0.95185,
 0.96145, 0.97105, 0.98065, 0.99025, 0.99986, 1.00946, 1.01907, 1.02868, 1.03828,
 1.04789, 1.05750, 1.06711, 1.07672, 1.08633, 1.09594, 1.10555, 1.11516, 1.12478,
 1.13439, 1.14401, 1.15362, 1.16324, 1.17286, 1.18247, 1.19209, 1.20171, 1.21133,
 1.22095, 1.23057, 1.24019, 1.24982, 1.25944, 1.26906, 1.25432, 1.26428, 1.27424,
 1.28420, 1.29415, 1.30411, 1.31407, 1.32403, 1.33399, 1.34395, 1.35391, 1.36387,
 1.37382, 1.38378, 1.39374, 1.40370, 1.41366, 1.42361, 1.43357, 1.44353, 1.45349,
 1.46344, 1.47340, 1.48336, 1.49331, 1.50327, 1.51323, 1.52318, 1.53314, 1.54310,
 1.55305, 1.56301, 1.57297, 1.58292, 1.59288, 1.60283, 1.61279, 1.62274, 1.63270,
 1.64265, 1.65261, 1.66256, 1.67252, 1.68247, 1.69243, 1.70238, 1.71234, 1.72229,
 1.73225, 1.74220, 1.75215, 1.76211, 1.77206, 1.78202, 1.79197, 1.80192, 1.81188,
 1.82183, 1.83178, 1.84173, 1.85169, 1.86164, 1.87159, 1.88154, 1.87830, 1.88834,
 1.89839, 1.90843, 1.91846, 1.92850, 1.93853, 1.94856, 1.95859, 1.96862, 1.97864,
 1.98867, 1.99869, 2.00870, 2.01872, 2.02873, 2.03874, 2.04875, 2.05876, 2.06876,
 2.07877, 2.08877, 2.09877, 2.10876, 2.11876, 2.12875, 2.13874, 2.14872, 2.15871,
 2.16869, 2.17867, 2.18865, 2.19863, 2.20860, 2.21857, 2.22854, 2.23851, 2.24848,
 2.25844, 2.26840, 2.27836, 2.28832, 2.29827, 2.30822, 2.31817, 2.32812, 2.33807,
 2.34801, 2.35795, 2.36789, 2.37783, 2.38776, 2.39769, 2.40762, 2.41755, 2.42748,
 2.43740, 2.44732, 2.45724, 2.46716, 2.47708, 2.48699, 2.49690, 2.50681}
fwhm = {
 0.00961, 0.00958, 0.00955, 0.00953, 0.00950, 0.00948, 0.00946, 0.00944, 0.00942,
 0.00940, 0.00938, 0.00937, 0.00935, 0.00934, 0.00933, 0.00931, 0.00930, 0.00929,
 0.00928, 0.00928, 0.00927, 0.00926, 0.00926, 0.00926, 0.00925, 0.00925, 0.00925,
 0.00925, 0.00926, 0.00926, 0.00926, 0.00927, 0.00830, 0.00831, 0.00833, 0.00834,
 0.00835, 0.00836, 0.00837, 0.00839, 0.00840, 0.00841, 0.00842, 0.00843, 0.00844,
 0.00845, 0.00846, 0.00846, 0.00847, 0.00848, 0.00849, 0.00850, 0.00850, 0.00851,
 0.00852, 0.00853, 0.00853, 0.00854, 0.00854, 0.00855, 0.00855, 0.00856, 0.00856,
 0.00857, 0.00857, 0.00857, 0.00858, 0.00858, 0.00858, 0.00858, 0.00859, 0.00859,
 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859,
 0.00859, 0.00859, 0.00859, 0.00858, 0.00858, 0.00858, 0.00858, 0.00857, 0.00857,
 0.00857, 0.00856, 0.00856, 0.00855, 0.00855, 0.00854, 0.01086, 0.01087, 0.01088,
 0.01089, 0.01089, 0.01090, 0.01090, 0.01091, 0.01091, 0.01091, 0.01092, 0.01092,
 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01091, 0.01091,
 0.01091, 0.01090, 0.01090, 0.01089, 0.01088, 0.01088, 0.01087, 0.01086, 0.01085,
 0.01084, 0.01083, 0.01082, 0.01081, 0.01080, 0.01079, 0.01077, 0.01076, 0.01075,
 0.01073, 0.01072, 0.01070, 0.01069, 0.01067, 0.01065, 0.01063, 0.01061, 0.01059,
 0.01057, 0.01055, 0.01053, 0.01051, 0.01049, 0.01047, 0.01044, 0.01042, 0.01039,
 0.01037, 0.01034, 0.01032, 0.01029, 0.01026, 0.01023, 0.01021, 0.01118, 0.01116,
 0.01114, 0.01113, 0.01111, 0.01109, 0.01108, 0.01106, 0.01104, 0.01102, 0.01101,
```

```
 0.01099, 0.01097, 0.01095, 0.01094, 0.01092, 0.01090, 0.01088, 0.01087, 0.01085,
 0.01083, 0.01081, 0.01080, 0.01078, 0.01076, 0.01074, 0.01072, 0.01071, 0.01069,
 0.01067, 0.01065, 0.01063, 0.01062, 0.01060, 0.01058, 0.01056, 0.01054, 0.01052,
 0.01050, 0.01049, 0.01047, 0.01045, 0.01043, 0.01041, 0.01039, 0.01037, 0.01035,
 0.01034, 0.01032, 0.01030, 0.01028, 0.01026, 0.01024, 0.01022, 0.01020, 0.01018,
 0.01016, 0.01014, 0.01012, 0.01011, 0.01009, 0.01007, 0.01005, 0.01003}
bbl = {
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1}
band names = {
 refl[0.370 micron] x 10000.0, refl[0.380 micron] x 10000.0,
 refl[0.390 micron] x 10000.0, refl[0.399 micron] x 10000.0,
 refl[0.409 micron] x 10000.0, refl[0.419 micron] x 10000.0,
 refl[0.429 micron] x 10000.0, refl[0.439 micron] x 10000.0,
 refl[0.449 micron] x 10000.0, refl[0.458 micron] x 10000.0,
 refl[0.468 micron] x 10000.0, refl[0.478 micron] x 10000.0,
 refl[0.488 micron] x 10000.0, refl[0.498 micron] x 10000.0,
 refl[0.508 micron] x 10000.0, refl[0.518 micron] x 10000.0,
 refl[0.527 micron] x 10000.0, refl[0.537 micron] x 10000.0,
 refl[0.547 micron] x 10000.0, refl[0.557 micron] x 10000.0,
 refl[0.567 micron] x 10000.0, refl[0.577 micron] x 10000.0,
 refl[0.587 micron] x 10000.0, refl[0.597 micron] x 10000.0,
 refl[0.606 micron] x 10000.0, refl[0.616 micron] x 10000.0,
 refl[0.626 micron] x 10000.0, refl[0.636 micron] x 10000.0,
 refl[0.646 micron] x 10000.0, refl[0.656 micron] x 10000.0,
 refl[0.666 micron] x 10000.0, refl[0.676 micron] x 10000.0,
 refl[0.664 micron] x 10000.0, refl[0.674 micron] x 10000.0,
 refl[0.683 micron] x 10000.0, refl[0.693 micron] x 10000.0,
 refl[0.702 micron] x 10000.0, refl[0.712 micron] x 10000.0,
 refl[0.722 micron] x 10000.0, refl[0.731 micron] x 10000.0,
 refl[0.741 micron] x 10000.0, refl[0.750 micron] x 10000.0,
 refl[0.760 micron] x 10000.0, refl[0.770 micron] x 10000.0,
 refl[0.779 micron] x 10000.0, refl[0.789 micron] x 10000.0,
 refl[0.798 micron] x 10000.0, refl[0.808 micron] x 10000.0,
 refl[0.817 micron] x 10000.0, refl[0.827 micron] x 10000.0,
 refl[0.837 micron] x 10000.0, refl[0.846 micron] x 10000.0,
 refl[0.856 micron] x 10000.0, refl[0.865 micron] x 10000.0,
 refl[0.875 micron] x 10000.0, refl[0.885 micron] x 10000.0,
 refl[0.894 micron] x 10000.0, refl[0.904 micron] x 10000.0,
 refl[0.913 micron] x 10000.0, refl[0.923 micron] x 10000.0,
 refl[0.933 micron] x 10000.0, refl[0.942 micron] x 10000.0,
 refl[0.952 micron] x 10000.0, refl[0.961 micron] x 10000.0,
 refl[0.971 micron] x 10000.0, refl[0.981 micron] x 10000.0,
 refl[0.990 micron] x 10000.0, refl[1.000 micron] x 10000.0,
 refl[1.009 micron] x 10000.0, refl[1.019 micron] x 10000.0,
```

```
refl[1.029 micron] x 10000.0, refl[1.038 micron] x 10000.0,
refl[1.048 micron] x 10000.0, refl[1.058 micron] x 10000.0,
refl[1.067 micron] x 10000.0, refl[1.077 micron] x 10000.0,
refl[1.086 micron] x 10000.0, refl[1.096 micron] x 10000.0,
refl[1.106 micron] x 10000.0, refl[1.115 micron] x 10000.0,
refl[1.125 micron] x 10000.0, refl[1.134 micron] x 10000.0,
refl[1.144 micron] x 10000.0, refl[1.154 micron] x 10000.0,
refl[1.163 micron] x 10000.0, refl[1.173 micron] x 10000.0,
refl[1.182 micron] x 10000.0, refl[1.192 micron] x 10000.0,
refl[1.202 micron] x 10000.0, refl[1.211 micron] x 10000.0,
refl[1.221 micron] x 10000.0, refl[1.231 micron] x 10000.0,
refl[1.240 micron] x 10000.0, refl[1.250 micron] x 10000.0,
refl[1.259 micron] x 10000.0, refl[1.269 micron] x 10000.0,
refl[1.254 micron] x 10000.0, refl[1.264 micron] x 10000.0,
refl[1.274 micron] x 10000.0, refl[1.284 micron] x 10000.0,
refl[1.294 micron] x 10000.0, refl[1.304 micron] x 10000.0,
refl[1.314 micron] x 10000.0, refl[1.324 micron] x 10000.0,
refl[1.334 micron] x 10000.0, refl[1.344 micron] x 10000.0,
refl[1.354 micron] x 10000.0, refl[1.364 micron] x 10000.0,
refl[1.374 micron] x 10000.0, refl[1.384 micron] x 10000.0,
refl[1.394 micron] x 10000.0, refl[1.404 micron] x 10000.0,
refl[1.414 micron] x 10000.0, refl[1.424 micron] x 10000.0,
refl[1.434 micron] x 10000.0, refl[1.444 micron] x 10000.0,
refl[1.453 micron] x 10000.0, refl[1.463 micron] x 10000.0,
refl[1.473 micron] x 10000.0, refl[1.483 micron] x 10000.0,
refl[1.493 micron] x 10000.0, refl[1.503 micron] x 10000.0,
refl[1.513 micron] x 10000.0, refl[1.523 micron] x 10000.0,
refl[1.533 micron] x 10000.0, refl[1.543 micron] x 10000.0,
refl[1.553 micron] x 10000.0, refl[1.563 micron] x 10000.0,
refl[1.573 micron] x 10000.0, refl[1.583 micron] x 10000.0,
refl[1.593 micron] x 10000.0, refl[1.603 micron] x 10000.0,
refl[1.613 micron] x 10000.0, refl[1.623 micron] x 10000.0,
refl[1.633 micron] x 10000.0, refl[1.643 micron] x 10000.0,
refl[1.653 micron] x 10000.0, refl[1.663 micron] x 10000.0,
refl[1.673 micron] x 10000.0, refl[1.682 micron] x 10000.0,
refl[1.692 micron] x 10000.0, refl[1.702 micron] x 10000.0,
refl[1.712 micron] x 10000.0, refl[1.722 micron] x 10000.0,
refl[1.732 micron] x 10000.0, refl[1.742 micron] x 10000.0,
refl[1.752 micron] x 10000.0, refl[1.762 micron] x 10000.0,
refl[1.772 micron] x 10000.0, refl[1.782 micron] x 10000.0,
refl[1.792 micron] x 10000.0, refl[1.802 micron] x 10000.0,
refl[1.812 micron] x 10000.0, refl[1.822 micron] x 10000.0,
refl[1.832 micron] x 10000.0, refl[1.842 micron] x 10000.0,
refl[1.852 micron] x 10000.0, refl[1.862 micron] x 10000.0,
refl[1.872 micron] x 10000.0, refl[1.882 micron] x 10000.0,
refl[1.878 micron] x 10000.0, refl[1.888 micron] x 10000.0,
refl[1.898 micron] x 10000.0, refl[1.908 micron] x 10000.0,
refl[1.918 micron] x 10000.0, refl[1.929 micron] x 10000.0,
refl[1.939 micron] x 10000.0, refl[1.949 micron] x 10000.0,
refl[1.959 micron] x 10000.0, refl[1.969 micron] x 10000.0,
refl[1.979 micron] x 10000.0, refl[1.989 micron] x 10000.0,
refl[1.999 micron] x 10000.0, refl[2.009 micron] x 10000.0,
refl[2.019 micron] x 10000.0, refl[2.029 micron] x 10000.0,
refl[2.039 micron] x 10000.0, refl[2.049 micron] x 10000.0,
refl[2.059 micron] x 10000.0, refl[2.069 micron] x 10000.0,
refl[2.079 micron] x 10000.0, refl[2.089 micron] x 10000.0,
refl[2.099 micron] x 10000.0, refl[2.109 micron] x 10000.0,
refl[2.119 micron] x 10000.0, refl[2.129 micron] x 10000.0,
refl[2.139 micron] x 10000.0, refl[2.149 micron] x 10000.0,
```

```
 refl[2.159 micron] x 10000.0, refl[2.169 micron] x 10000.0,
 refl[2.179 micron] x 10000.0, refl[2.189 micron] x 10000.0,
 refl[2.199 micron] x 10000.0, refl[2.209 micron] x 10000.0,
 refl[2.219 micron] x 10000.0, refl[2.229 micron] x 10000.0,
 refl[2.239 micron] x 10000.0, refl[2.248 micron] x 10000.0,
 refl[2.258 micron] x 10000.0, refl[2.268 micron] x 10000.0,
 refl[2.278 micron] x 10000.0, refl[2.288 micron] x 10000.0,
 refl[2.298 micron] x 10000.0, refl[2.308 micron] x 10000.0,
 refl[2.318 micron] x 10000.0, refl[2.328 micron] x 10000.0,
 refl[2.338 micron] x 10000.0, refl[2.348 micron] x 10000.0,
 refl[2.358 micron] x 10000.0, refl[2.368 micron] x 10000.0,
 refl[2.378 micron] x 10000.0, refl[2.388 micron] x 10000.0,
 refl[2.398 micron] x 10000.0, refl[2.408 micron] x 10000.0,
 refl[2.418 micron] x 10000.0, refl[2.427 micron] x 10000.0,
 refl[2.437 micron] x 10000.0, refl[2.447 micron] x 10000.0,
 refl[2.457 micron] x 10000.0, refl[2.467 micron] x 10000.0,
 refl[2.477 micron] x 10000.0, refl[2.487 micron] x 10000.0,
 refl[2.497 micron] x 10000.0, refl[2.507 micron] x 10000.0}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

mask_name = mask for SGI
mask_version = 0.9i 1999-Oct-21-16:25:00 EDT
mask_execution_date = 1999-Oct-26
mask_execution_time = 10:31:40   -0500
mask_input_image_name = ./coral.bip
mask_which_masks = { land, cirrus, low cloud}
mask_data_directory = ../
mask_output_root_name = ./coral_test
ndvi_scale_factor =  10000.0000
land_mask_threshold =      0.0500
cirrus_mask_threshold =      0.0045
cloud_mask_ocean_threshold =      0.1000

tafkaa_name = tAFKaA (tabular) for SGI
tafkaa_version = 0.9i 1999-Nov-09-16:24:00 EDT
tafkaa_execution_date = 1999-Nov-10
tafkaa_execution_time = 17:00:20   -0500
tafkaa_input_image_name = ./coral.bip
tafkaa_ground_elevation =  0.000
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O,CO2,O3 ,N2O,CO ,CH4,O2 }
tafkaa_atmo_ozone = 0.340
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = { 0.8650, 1.0300, 0.9450}
```

```
tafkaa_h2o_nb_set1 = {  3,  3,  5}
tafkaa_h2o_wl_set2 = { 1.0650, 1.2400, 1.1400}
tafkaa_h2o_nb_set2 = {  3,  3,  5}
tafkaa_wind_speed =  2.00
tafkaa_aerosol_method =       0
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_tau550 = 0.100
tafkaa_aerosol_weights = { 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_type = refl
tafkaa_output_root_name = ./coral_test
tafkaa_output_scale_factor =  10000.0000

history = ends
```

D.2.2 Product Output Header File

Based on the above files, the name of the mask output image will be `./coral_test_tafkaa_prod.img` and the image header file will be `./coral_test_tafkaa_prod.hdr`.

```
ENVI
description = {
  Product file
  derived from tafkaa (tabular);
  measurement date: 1996-03-23; measurement time: 19:44:29.000 GMT
  latitude:  24deg 36m 44.000s N ; longitude:  81deg 47m 54.000s W
  view zenith angle:   0deg  0m  0.000s ;
  relative azimuth angle:   0deg  0m  0.000s ;
  atmospheric model: tropical           ;
  absorption includes : H_2O CO_2 O_3 N_2O CO CH_4 O_2;
  ozone amount:  0.340 atm-cm ;
  windspeed:   2.00 m/s;
  weights for determining aerosols:
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00 1.00 1.00 }
samples = 614
lines   = 600
bands   = 5
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bsq
byte order = 1
sensor type = AVIRIS
image_center_date = { 1996, 03, 23}
image_center_time = { 19., 44., 29.000}
image_center_lat = { 24., 36., 44.000}
image_center_lat_hem = N
image_center_long = {  81., 47., 54.000}
image_center_long_hem = W
image_center_zenith_ang = {  0.,  0.,  0.000}
image_center_azimuth_ang = {   0.,  0.,  0.000}
sensor_altitude =    19.768
image_scale_factor = {
   1000.0000,   1000.0000,      1.0000,      1.0000,    100.0000}
image_unscaled_units = {  cm of H2O, tauaer550, RH %, none, none}
```

```
band names = {
1000.000x cm of H2O,1000.000x tau aerosol (550nm), %RH, model, 100.000xfitqual}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

mask_name = mask for SGI
mask_version = 0.9i 1999-Oct-21-16:25:00 EDT
mask_execution_date = 1999-Oct-26
mask_execution_time = 10:31:40   -0500
mask_input_image_name = ./coral.bip
mask_which_masks = { land, cirrus, low cloud}
mask_data_directory = ../
mask_output_root_name = ./coral_test
ndvi_scale_factor =  10000.0000
land_mask_threshold =      0.0500
cirrus_mask_threshold =      0.0045
cloud_mask_ocean_threshold =      0.1000

tafkaa_name = tAFKaA (tabular) for SGI
tafkaa_version = 0.9i 1999-Nov-09-16:24:00 EDT
tafkaa_execution_date = 1999-Nov-10
tafkaa_execution_time = 17:00:20   -0500
tafkaa_input_image_name = ./coral.bip
tafkaa_ground_elevation =  0.000
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O,CO2,O3 ,N2O,CO ,CH4,O2 }
tafkaa_atmo_ozone = 0.340
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = { 0.8650, 1.0300, 0.9450}
tafkaa_h2o_nb_set1 = {  3,  3,  5}
tafkaa_h2o_wl_set2 = { 1.0650, 1.2400, 1.1400}
tafkaa_h2o_nb_set2 = {  3,  3,  5}
tafkaa_wind_speed =  2.00
tafkaa_aerosol_method =       0
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_tau550 = 0.100
tafkaa_aerosol_weights = { 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_type = refl
tafkaa_output_root_name = ./coral_test
```

```
tafkaa_output_scale_factor =  10000.0000

history = ends
```

E Aerosol Properties

Mode	Comments	Model Name				
		M	C	C-a	T	U
1	Continental origin rural aerosol mixture	0.990	0.995	0.998	1.000	
2	Oceanic origin, sea salt solution in water	0.010	0.005	0.002		
3	Urban aerosol mixture, rural & soot					$1 - 1.25\text{e-}4$
4	Urban aerosol mixture, rural & soot					1.25e-4

Table 1: A rough description of the types of aerosols present, and the fractions present in the different aerosol models used in *Tafkaa* . The abbreviations used for the model names are: M (Maritime); C (Coastal); C-a (Coastal-a); T (Tropospheric); and U (Urban).

All the aerosol distributions used are log-normal; most are bi-modal[9] . Component i is represented as

$$\frac{dN_i(r)}{dr} = n_i(r) = \frac{N f_i}{r\sigma_i\sqrt{2\pi}} \exp\left\{-\frac{(\ln r - \ln r_{m_i})^2}{2\sigma_i^2}\right\} . \tag{14}$$

Here, N is the total number density of particles, f_i is the fractional number density of species i, r is the particle radius, σ_i is the standard deviation of $\ln r$, i.e., $\sigma_i^2 = \langle(\ln r - \ln r_{m_i})^2\rangle$, and r_{m_i} is the mean radius of $\ln r$. Then, the total particle distribution is described as the sum over K components as

$$\frac{dN(r)}{dr} = n(r) = \sum_{i=1}^{K} \frac{dN_i(r)}{dr} . \tag{15}$$

The properties used in determining the aerosols come from Shettle & Fenn (1979). Currently four different aerosol modes are used, and they are used to make 5 general aerosol models. The proportions of each mode in the different aerosol models is given in Table 1, the parameters of their lognormal distributions are given in Table 2, and the frequency dependent aerosol optical properties are listed in Tables 3 – 7.

The aerosol information listed in these tables is read by Zia Ahmad's program `phs`[10] which creates phase function files for the various aerosols. His `rt1` or `rt1.crft` then applies a distribution of the aerosols throughout the atmosphere in order to get reasonable distributions. Finally, `rt2` or `rt2.crft` perform the radiative transfer calculations using output from both of the above, and create the files that are later parsed into the lookup tables used by *Tafkaa* .

Mode	r_m (micron)					
	RH=50%	RH=70%	RH=80%	RH=90%	RH=98%	σ
1	2.748e-02	2.846e-02	3.274e-02	3.884e-02	4.751e-02	8.059e-01
2	1.711e-01	2.041e-01	3.180e-01	3.803e-01	6.024e-01	9.210e-01
3	2.563e-02	2.911e-02	3.514e-02	4.187e-02	5.996e-02	8.059e-01
4	4.113e-01	4.777e-01	5.805e-01	7.061e-01	1.169e-00	9.210e-01

Table 2: $r_{m_i} = \exp(\langle \ln r \rangle)$ (columns 2 – 6) and $\sigma_i = \sqrt{\langle(\ln r - \ln r_{m_i})^2\rangle}$ for different aerosol modes. σ_i does not depend on relative humidity.

[9] The troposheric model is unimodal, as seen in Table 1.

[10] The format is different than is listed here. The format of the tables in this document is more concise.

λ	RH = 50%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.520	5.60e-3	1.472	3.75e-8	1.557	9.01e-2	1.555	8.93e-2
0.410	1.520	5.60e-3	1.471	2.37e-8	1.557	8.95e-2	1.555	8.88e-2
0.440	1.520	5.60e-3	1.470	2.07e-8	1.557	8.89e-2	1.555	8.81e-2
0.470	1.520	5.60e-3	1.470	1.80e-8	1.557	8.83e-2	1.555	8.75e-2
0.510	1.520	5.60e-3	1.470	9.42e-9	1.557	8.79e-2	1.555	8.71e-2
0.550	1.520	6.26e-3	1.470	8.54e-9	1.557	8.66e-2	1.555	8.58e-2
0.610	1.520	6.26e-3	1.464	1.53e-8	1.557	8.52e-2	1.555	8.45e-2
0.670	1.520	6.65e-3	1.461	4.78e-8	1.557	8.50e-2	1.555	8.43e-2
0.750	1.517	7.90e-3	1.458	2.70e-7	1.554	8.61e-2	1.552	8.53e-2
0.865	1.510	1.03e-2	1.452	2.79e-6	1.549	8.79e-2	1.547	8.72e-2
1.040	1.510	1.32e-2	1.445	1.08e-4	1.549	9.18e-2	1.547	9.10e-2
1.240	1.492	1.50e-2	1.443	2.80e-4	1.536	9.48e-2	1.534	9.40e-2
1.640	1.448	1.59e-2	1.430	5.69e-4	1.505	9.94e-2	1.503	9.85e-2
2.250	1.356	9.22e-3	1.413	1.71e-3	1.439	1.00e-1	1.437	9.92e-2

Table 3: Optical properties for the aerosol modes for RH = 50%.

λ	RH = 70%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.502	5.04e-3	1.418	2.31e-8	1.488	6.15e-2	1.477	5.70e-2
0.410	1.502	5.04e-3	1.416	1.47e-8	1.488	6.11e-2	1.477	5.66e-2
0.440	1.502	5.04e-3	1.416	1.27e-8	1.487	6.07e-2	1.476	5.62e-2
0.470	1.502	5.04e-3	1.415	1.11e-8	1.487	6.03e-2	1.476	5.58e-2
0.510	1.501	5.04e-3	1.414	6.04e-9	1.486	6.00e-2	1.475	5.56e-2
0.550	1.501	5.64e-3	1.413	5.84e-9	1.486	5.91e-2	1.474	5.48e-2
0.610	1.501	5.64e-3	1.410	1.28e-8	1.485	5.82e-2	1.474	5.39e-2
0.670	1.501	5.99e-3	1.408	3.81e-8	1.485	5.80e-2	1.474	5.38e-2
0.750	1.498	7.11e-3	1.406	1.89e-7	1.483	5.88e-2	1.472	5.45e-2
0.865	1.492	9.29e-3	1.402	1.79e-6	1.479	6.00e-2	1.468	5.56e-2
1.040	1.492	1.19e-2	1.396	6.53e-5	1.478	6.27e-2	1.467	5.81e-2
1.240	1.475	1.35e-2	1.394	1.75e-4	1.469	6.47e-2	1.458	6.00e-2
1.640	1.435	1.43e-2	1.383	3.79e-4	1.445	6.78e-2	1.435	6.29e-2
2.250	1.350	8.34e-3	1.363	1.17e-3	1.392	6.84e-2	1.385	6.34e-2

Table 4: Optical properties for the aerosol modes for RH = 70%.

λ	RH = 80%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.447	3.31e-3	1.361	7.94e-9	1.424	3.49e-2	1.417	3.17e-2
0.410	1.446	3.31e-3	1.359	5.16e-9	1.423	3.47e-2	1.416	3.16e-2
0.440	1.445	3.31e-3	1.358	4.40e-9	1.422	3.45e-2	1.415	3.13e-2
0.470	1.445	3.31e-3	1.357	3.75e-9	1.422	3.43e-2	1.414	3.11e-2
0.510	1.444	3.31e-3	1.355	2.46e-9	1.421	3.41e-2	1.413	3.10e-2
0.550	1.443	3.70e-3	1.354	3.00e-9	1.420	3.36e-2	1.412	3.05e-2
0.610	1.443	3.70e-3	1.353	9.76e-9	1.419	3.31e-2	1.411	3.00e-2
0.670	1.443	3.93e-3	1.352	2.71e-8	1.419	3.30e-2	1.411	3.00e-2
0.750	1.440	4.67e-3	1.350	1.02e-7	1.417	3.34e-2	1.409	3.04e-2
0.865	1.436	6.10e-3	1.348	7.35e-7	1.414	3.41e-2	1.406	3.10e-2
1.040	1.435	7.80e-3	1.345	2.00e-5	1.413	3.56e-2	1.405	3.24e-2
1.240	1.423	8.90e-3	1.342	6.40e-5	1.406	3.68e-2	1.399	3.35e-2
1.640	1.394	9.42e-3	1.334	1.78e-4	1.389	3.86e-2	1.382	3.51e-2
2.250	1.330	5.61e-3	1.311	5.95e-4	1.349	3.91e-2	1.344	3.55e-2

Table 5: Optical properties for the aerosol modes for RH = 80%.

λ	RH = 90%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.404	1.98e-3	1.352	5.66e-9	1.390	2.07e-2	1.383	1.76e-2
0.410	1.403	1.98e-3	1.351	3.74e-9	1.389	2.05e-2	1.381	1.75e-2
0.440	1.402	1.98e-3	1.349	3.15e-9	1.388	2.04e-2	1.380	1.74e-2
0.470	1.401	1.98e-3	1.348	2.66e-9	1.387	2.02e-2	1.379	1.73e-2
0.510	1.400	1.98e-3	1.347	1.92e-9	1.385	2.02e-2	1.378	1.72e-2
0.550	1.399	2.22e-3	1.345	2.58e-9	1.384	1.99e-2	1.377	1.70e-2
0.610	1.399	2.22e-3	1.344	9.24e-9	1.384	1.95e-2	1.376	1.67e-2
0.670	1.398	2.36e-3	1.343	2.53e-8	1.383	1.95e-2	1.375	1.67e-2
0.750	1.396	2.80e-3	1.342	8.83e-8	1.382	1.97e-2	1.374	1.69e-2
0.865	1.393	3.65e-3	1.340	5.77e-7	1.379	2.02e-2	1.372	1.72e-2
1.040	1.391	4.67e-3	1.337	1.31e-5	1.377	2.11e-2	1.370	1.80e-2
1.240	1.383	5.34e-3	1.335	4.71e-5	1.372	2.18e-2	1.365	1.86e-2
1.640	1.362	5.69e-3	1.326	1.48e-4	1.359	2.29e-2	1.353	1.95e-2
2.250	1.315	3.52e-3	1.303	5.10e-4	1.326	2.32e-2	1.321	1.99e-2

Table 6: Optical properties for the aerosol modes for RH = 90%.

λ	RH = 98%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.375	1.08e-3	1.343	3.22e-9	1.357	7.03e-3	1.349	3.89e-3
0.410	1.374	1.08e-3	1.342	2.24e-9	1.356	6.99e-3	1.348	3.86e-3
0.440	1.373	1.08e-3	1.340	1.84e-9	1.354	6.94e-3	1.347	3.84e-3
0.470	1.371	1.08e-3	1.339	1.51e-9	1.353	6.89e-3	1.345	3.81e-3
0.510	1.370	1.08e-3	1.337	1.35e-9	1.352	6.87e-3	1.344	3.79e-3
0.550	1.369	1.21e-3	1.336	2.13e-9	1.350	6.76e-3	1.343	3.74e-3
0.610	1.369	1.21e-3	1.335	8.64e-9	1.350	6.66e-3	1.342	3.68e-3
0.670	1.368	1.29e-3	1.334	2.34e-8	1.349	6.64e-3	1.341	3.67e-3
0.750	1.366	1.53e-3	1.333	7.32e-8	1.348	6.72e-3	1.340	3.72e-3
0.865	1.364	2.00e-3	1.332	4.09e-7	1.346	6.87e-3	1.338	3.80e-3
1.040	1.362	2.56e-3	1.329	5.80e-6	1.344	7.18e-3	1.336	3.97e-3
1.240	1.356	2.93e-3	1.326	2.85e-5	1.340	7.43e-3	1.333	4.12e-3
1.640	1.341	3.16e-3	1.318	1.16e-4	1.330	7.86e-3	1.324	4.39e-3
2.250	1.304	2.10e-3	1.295	4.20e-4	1.303	8.17e-3	1.298	4.69e-3

Table 7: Optical properties for the aerosol modes for RH = 98%.