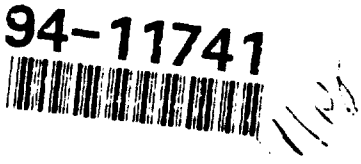
Naval Command,
Control and Ocean
Surveillance Center RDT&E Division San Diego, CA
92152-5001

(4)

Technical Report 1634
December 1993

The Navy Oceanic Vertical Aerosol Model

S. G. Gathman
K. L. Davidson, NPGS

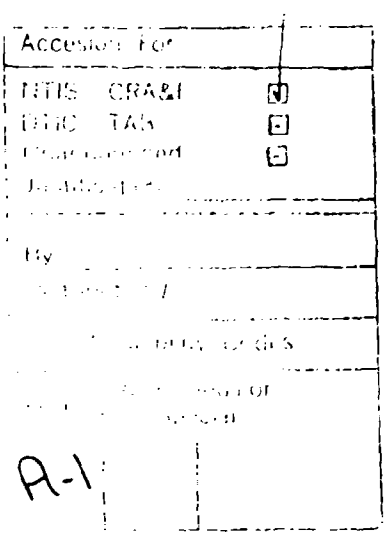


NRaD

Technical Report 1634
December 1993

The Navy Oceanic Vertical Aerosol Model

S. G. Gathman
K. L. Davidson, NPGS



NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
RDT&E DIVISION
San Diego, California 92152–5001

K. E. EVANS, CAPT, USN
Commanding Officer

R. T. SHEARER
Executive Director

ADMINISTRATIVE INFORMATION

This work was prepared by the Ocean and Atmospheric Sciences Division, Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA 92152–5001.

Released by
R. A. Paulus, Head
Tropospheric Branch

Under authority of
J. H. Richter, Head
Ocean and Atmospheric
Sciences Division

ACKNOWLEDGMENTS

The authors wish to express their special thanks to Dr. Juergen Richter of NRaD who has provided both moral and financial support for the development of this program over the time it took to develop and test. We further wish to thank our colleague Ray Noonkester, formerly of NOSC, who was very supportive in the early development of NOVAM. In addition, we wish to acknowledge the contributions of Charles McGrath and Linda Hitney of NRaD who provided support in the development of the code for NOVAM.

EXECUTIVE SUMMARY

OBJECTIVE

The object was the development of the FORTRAN version of the Navy oceanic vertical model (NOVAM). The model predicts the vertical distribution of aerosol in the first 6000 meters above the ocean.

RESULTS

The NOVAM was developed from extensive marine aerosol studies from different laboratories. The climax was a multimenu-driven interactive program that allows mouse selection of menu items needed for the calculation. It includes graphics and editing capabilities useful to the researcher. When used with an appropriate method to determine the profile parameters, it could be used in a fully automatic mode in which the program could access sets of data and produce an analysis of the data sets in an unattended background mode. The final NOVAM code is intended, however, to be used in conjunction with a LOWTRAN/MODTRAN program and to supply the electro-optical (EO) propagation characteristics to the calling program that are produced by the unique aerosol found in the marine atmosphere. It is written in FORTRAN so it can be integrated into LOWTRAN/MODTRAN codes to improve model performance in marine environment.

RECOMMENDATIONS

The model has several shortcomings that will be addressed in future modifications. The region of applicability leaves two areas not covered well by the model. First, higher altitudes, various models developed by the U. S. Air Force and included in the LOWTRAN/MODTRAN code will be more accurate. Second, propagation paths that graze the sea surface or pass through the region within a meter or so of the sea surface are not adequately covered by NOVAM. This problem is being remedied by a large-scale experiment off the Dutch coast sponsored by NATO. Another shortcoming is the limited types of weather situations in which it is applicable. Advances in these areas await development of models from the basic research community in the future. Another area of concern is the use of the model in close-in coastal areas. Compensation was introduced, but experience has shown this is one of the weakest parts of the model. It is the author's opinion that a special coastal aerosol model needs to be developed that will adequately take into account local sources of aerosol.

CONTENTS

FIGURES

TABLES

INTRODUCTION

The need for a model to accurately describe the atmosphere's optical and infrared (IR) properties of its meteorological state has been an important driving force for the Navy's electro-optics research program. Ideally, this model should describe the optical and/or IR properties of the atmosphere by using easily obtained meteorological data as input. Several atmospheric "windows" exist in the molecular absorption of the electromagnetic energy through which transmissions in IR communication can take place. In these windows, the natural atmospheric aerosol plays an important part in degrading the transmission of radiation from point A to point B. This is done when the aerosol both scatters and absorbs electromagnetic energy. Of particular interest to the Navy is the role natural marine aerosols play within the marine boundary layer (MBL) in causing extinction at both visible and IR wavelengths.

The Navy aerosol model (NAM) was developed in the 1980s [1–3] to describe the marine aerosol and its optical and IR propagation properties for the atmosphere's MBL from the local meteorological parameters at shipboard level. IR extinction over the oceans is a function of various meteorological parameters such as visibility, absolute humidity, relative humidity (RH), and wind. The optical and IR properties of the MBL are related by this model to the meteorological quantities that are measurable from a shipboard environment. The NAM model has certain limitations, however, because it was developed from a limited database containing only measurements made at or near the shipboard level. It is essentially a nondimensional model that contains no real vertical structure. When the model is used in the MBL at levels away from the surface, it must assume that the MBL is extremely well mixed and that the size distribution of the aerosol within this layer is constant from the sea surface to the top of the MBL. This may or may not be a good assumption. With the addition of other types of observations and with our growing knowledge of the mesoscale properties of the MBL, it is now possible to better take into account the vertical structure of the MBL in making optical and IR predictions.

The NOVAM is developed from extensive marine aerosol studies from several different laboratories including: the Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division (NRaD, formerly NOSC), San Diego, CA; the Naval Research Laboratory,Washington DC; the Naval Post Graduate School, Monterey, CA; and the TNO Physics and Electronics Laboratory (TNO-PEL) The Hague, The Netherlands.

Several preliminary versions of NOVAM were developed over several years and tested for both the programming integrity and the physical correctness of the modeling assumptions. Thus, the basic concepts of NOVAM have been programmed in several computer languages as the model developed with time. The first limited versions were programmed in BASIC but when the program became too complex for a BASIC type language, and thus too slow for applications, the code was converted to TURBO Pascal for speedier operation on a typical personal computer (PC). The Pascal programs developed into a user friendly PC program designed to exercise the NOVAM code with real sets of data. The climax of this development was a multimenu-driven interactive program that allows mouse selection of menu items needed for the calculation. It includes graphics and editing capabilities that are useful for the researcher. This program uses a semiautomatic method of characterizing the atmospheric profile parameters that allows greater operator interaction with the calculation. When used with an appropriate method to determine the profile parameters, it could be used in a fully automatic mode that the program could access

sets of data and produce an analysis of the data sets in an unattended background mode. The final NOVAM code is intended, however, to be used in conjunction with a LOWTRAN/MODTRAN program and to supply the EO propagation characteristics to the calling program that are produced by the unique aerosol found in the marine atmosphere.

As a result of this evolutionary development, it was found over time that certain aspects of the program should be improved and others perhaps left out or simplified. The current FORTRAN subroutine is based on the results of this previous work and testing. It is written in FORTRAN so it can be easily integrated into the LOWTRAN/MODTRAN codes to improve the performance of these models in the marine environment. Being a subroutine, it is no longer an interactive program, but requires a certain set of information supplied to it and produces a limited utilitarian output needed for the larger scale programs. These features and improvements are incorporated into the Navy oceanic vertical aerosol model subroutine (NOVAMSR) code presented here.

To meet this need, NOVAM was changed into the form of a subroutine written in the FORTRAN language. It provides information on the optical properties of the aerosol at any altitude and wavelength in the marine atmosphere. The subroutine gives additional information not available in the development model. In addition to extinction and absorption coefficients at altitudes, the subroutine returns the relative humidity and the set of lognormal coefficients needed to reproduce the size distribution of the aerosol from which other characteristics of the aerosol could be obtained, such as the phase function. The NOVAMSR is also an intelligent routine that uses as input a radiosonde profile data and automatically provides the significant profile parameters needed for the NOVAM. Thus, the calculation can be done completely under machine control without the interaction of a knowledgeable operator.

TESTING THE NOVAM CONCEPTS

The NOVAM concepts have been tested with data from several sets of field experiments from different geographical regions of the world's oceans. These include the areas off the California coast, off the Florida Keys, and off the Virginia coast. In each case care was taken to assure that the air mass being measured was marine in origin and not freshly blown in from the nearby coastal regions. In all of these experiments, there were sufficient meteorological measurements taken so all of the input meteorological data needed by NOVAM in both its surface observation file and the radiosonde observation file were available. This included a certain amount of redundancy in measurements to assure the accuracy and determine the size of the error bars associated with them. In addition, profiles of the aerosol size distribution were measured and these data, when used with the Mie Theory, allowed profiles of "measured" aerosol extinction to be determined. The experiment then consisted of comparing the "measured" and the NOVAM predicted extinction profiles with each other to determine if NOVAM was indeed working. It was determined that it is impossible to precisely measure the extinction profile with present day techniques. There is usually a sufficient spread in "measured" extinction data that the best that could be determined is an envelope of the "measured" data. In these tests, then, when the NOVAM model predicted extinction values at any particular altitude that was within the "measured" envelope, it was considered a successful prediction.

The first experiment in the evaluation process was in conjunction with the FIRE experiment. FIRE stands for First [International Satellite Cloud Climatology Project], ISCCP Regional

Experiment. This was a series of profile measurements made with a tethered balloon on San Nicolas Island, CA. [4,5,6].

The second experiment was done in the semitropical waters to the east of the Florida Keys in an experiment called **KEY-90**. [7,8,9,10]. In this experiment, the surface observations were made with an instrumented fishing boat. Meteorological and electro-optical measurements were made with aircraft, balloons, and lidars.

The third sets of measurements were made with aircraft profiles in conjunction with the **I**nfra**R**ed **A**nalysis **M**easurement and **M**odeling **P**rogram (IRAMMP) project. Two sets of measurements (1991 and 1992) were made off the Virginia coast and some of the data were used to test the ability of NOVAM to operate in the western Atlantic scenario. The profile data in these experiments came from the NRaD instrumented Navajo aircraft, while the surface observation file was made by a combination of available data obtained from shore stations and other sources.

THEORY

This version of NOVAM recognizes three types of meteorological profiles. They are differentiated by the characteristics of the temperature profile in terms of the existence or nonexistence of temperature inversions. Three cases recognized by NOVAM are

1.	No inversion	The free convection mode
2.	Two inversions	The weak convection mode
3.	One inversion	The developed boundary layer.

The NOVAM is, however, an aerosol model not a meteorological model. It looks at the meteorological data and then estimates what is happening to the scalar contaminants, such as aerosol, and how they are transported throughout the atmosphere. The optical and IR properties obtained from this model describe the effects due to the aerosol loading of the atmosphere. Molecular effects are well described with other types of models such as LOWTRAN or MODTRAN when they are operating in the aerosol "free" mode. These effects should be kept in mind when comparing the total extinction measured from instruments and the aerosol extinction due to atmospheric aerosol and molecular extinction, due to the molecules in the atmosphere. Fortunately the molecular composition of air remains relatively fixed with respect to time and place (with the exception of water vapor which can be easily measured) so that these results can be well known.

The model does not attempt to describe the situations within clouds if they exist in the propagation path. The sections of the propagation path in which clouds exist (when the RH approaches saturation), a separate set of physical laws must be used to describe the properties of the cloud. In these cases a separate model should be used for optical and IR properties within clouds. Likewise, if precipitation is present, the model cannot describe optical and IR properties from the precipitation. In this case, the additional precipitation effects should be computed using an appropriate precipitation model.

The aerosol size distribution has been represented by several different functional forms based on the intuition of several different investigators over the years. The lognormal type functions

have been used because there is a close relationship between statistical quantities and these functions. The NAM and NOVAM use a form of lognormal that differs somewhat from other published forms of the function. The reason for this is pragmatic in that it simplifies working with the plotting of measured data on log-log plots and in transforming these data into other moments of the lognormal for emphasizing important aspects of the data. Davies [11] in his description of an aerosol size distribution used $dN/d\log r$ as a lognormal function that has the form

$$\frac{dN}{d\log r} = \frac{2.303N}{\sqrt{2\pi}\cdot\log(\sigma_g)}\cdot\exp\left[-\frac{1}{2}\left[\frac{\log(\frac{r}{r_m})}{\log(\sigma_g)}\right]^2\right] \quad , \tag{1}$$

where N is the total number of particles per cm^3 in the population, r is the radius of the particle, r_m is the median radius, and $\log(\sigma_g)$ is the standard deviation of $\log(r)$. Whereas in NAM, Gathman [1,2&3], and here in the NOVAM formulation, the description of a population of aerosols is expressed as dN/dr and a slightly different form of the lognormal is used. Here it is expressed as

$$\frac{dN}{dr} = \frac{A}{f}\cdot\exp\left[-C\cdot\log^2\left[\frac{r}{f\cdot r_0}\right]\right] \quad . \tag{2}$$

These two expressions are related to each other by

$$\frac{dN}{dr} = \frac{1}{r}\cdot\frac{dN}{d\log r}\cdot\frac{1}{\log(10)} \quad . \tag{3}$$

This transposition influences the position of the mode radii however. In the first equation, the parameters of the equation are statistical parameters of the population of aerosols. In the latter equation, the parameters have graphical significance when plotted on a log-log scale. The parameter A/f is the amplitude or maximum dN/dr of the function, f is a swelling factor that represents the effect of RH on hygroscopic aerosol, r_0 represents the mode radius, i.e., that radius which coincides with the maximum dN/dr when $f = 1$ and C represents the "breadth" of the lognormal. The factor f describes the change in size of the particle when it grows or shrinks in different RH environments. The value of f is set to be 1 when the surrounding RH is 80% (the mean RH found over the ocean). Thus, it is only important when the surrounding RH differs from the mean value of 80%.

The relationship between the two forms are shown to be equivalent in appendix A of Smith and Bates [12]. Their results are summarized in equations (4) and (5) and express the NAM amplitude, A_i and width, C of the i^{th} component in terms of the statistical properties N, r_{io}, and σ_g of the Davies distribution.

$$A_i = \frac{N}{\sqrt{2\pi}\cdot r_{i0}\cdot\log(\sigma_g)}\exp\left[-\frac{1}{2}\log^2(\sigma_g)\right] \tag{4}$$

$$C = \frac{1}{2\log^2(\sigma_g)} \quad . \tag{5}$$

Here the peak value in NAM is A_i/f_i and is at the place where

$$r = r_{mode} = f_i r_{i0} \, . \tag{6}$$

It is important to be able to convert between various moments of the size distribution. This is because the size distribution of aerosol can be expressed as $dN/d\log r$, dN/dr, dN/dA, or dN/dV where A and V are the area and volume elements of the derivative. It is a property of these functions that can be easily transformed from one moment to another and still be represented by a basic "lognormal" type of function described by different coefficients.

The various moments of the lognormal can be transformed into another lognormal form by the following transformation. This handy transformation allows conversion of data into other moments of the lognormal.

$$C_2 \cdot r^n \cdot \exp\left(-C_3 \cdot \left(\log^2\left(\frac{r}{r_0}\right)\right)\right) = C_4 \cdot \exp\left(-C_3 \cdot \left(\log^2\left(\frac{r}{r_1}\right)\right)\right)$$

$$\text{where} \quad C_4 = C_2 \cdot r_0^n \cdot \exp\left(\frac{n^2}{4C_3}\right) \quad \text{and} \quad r_1 = r_0 \cdot \exp\left(\frac{n}{2C_3}\right). \qquad (7)$$

Here n is an integer representing the moment of the lognormal, the C's , r_0 and r_1 are constants and r is the variable.

In modeling these aerosols, several different species of aerosol are recognized and are each distinguished here by their own "lognormal" distribution. The net effect of all of these species on the aerosol size distribution is represented by the sum of all of these components. This modeling process leaves open the prospect that other species of aerosol might be present. It is expected that in the coastal regions and other special places, unique populations of aerosol exist and must be represented by their own unique size distribution. These additional elements of the aerosol population can be introduced into this model by adding additional lognormal or other types of functions. When our knowledge is expanded as to the characteristics of the aerosol in coastal regions and areas very close to the sea surface, additional components will be added to the current description.

The single lognormal function (as seen in equation (2)) is described by four parameters. The amplitude A is related to the concentration of the aerosol; the mode radius, r_0, refers to the "size" of the most populous part of the distribution or the peak of the distribution at an RH of 80%; C refers to the size diversity or breadth of that population; and the factor f refers to the growth or shrinking of the particle size at relative humidities different than 80%. In the NAM and NOVAM series of models, C has a value of 1 for all of the species of aerosol used and mode radii, r_0 of the various distributions has been empirically fixed at values of 0.03 μm, 0.24 μm, and 2.0 μm. The factor f will be different for each species of aerosol depending on its assumed chemical composition and will be a factor of the surrounding RH. The model assumes that the particles are all in equilibrium with the RH of its surrounding environment. This leaves only the amplitude factors A_i to be related to the generation and distribution processes at work at the sea surface and in the marine atmosphere. The A_i parameters are the key factors that relate aerosol size distribution as it changes in the vertical regions of the atmosphere to the meteorology of the MBL. The model assumes that all the mixing and other processes at work on the aerosol in the marine atmosphere work only on four different sized aerosols representing the four size distributions. Each size distribution is represented by a single size, r_{0i}, and concentration, A_i, with an f factor of 1 and then at the levels of interest, the full-size distribution is expressed by these factors and the in situ RH is allowed to change the f factor to represent the size distribution at that altitude.

NOVAM further assumes that the aerosol size distribution is correctly described by NAM at the ship level above the sea surface. Then the spatial distributions of the aerosol are represented by changes from this surface value.

In general now, the size distribution of NOVAM at the altitude z is represented as

$$\frac{dN(z)}{dr} = \sum_{i=0}^{3} k_i \cdot \frac{A_i(z)}{f_i} \cdot \exp\left\{-1 \cdot \left[\log\left[\frac{r}{(r_{0i} \cdot f_i)}\right]\right]^2\right\} \tag{8}$$

where f_i are functions of the RH. The values of r_{0i} are given above, and the k_i's are correction functions, which correct the surface values to direct measurements of visibility and/or IR extinction when available. The functioning of the NOVAM code supplies the values of the $A_i(z)$. Now that the aerosol size distribution is known at the altitude z, various optical parameters can be obtained from this size distribution.

NOVAM uses the index of refraction of water from Hale and Query [13] and makes an assumption on the chemical composition of each of the four aerosol modes in the model. In the smallest size classes, two chemical species of both soluble and nonsoluble aerosol are allowed. It is known that the chemical composition of the soluble droplets change as a function of RH. A perfectly dry particle has a complex index of refraction of the nucleus material. On the other hand, in a very high RH environment, a hygroscopic particle will have changed size, taking on a considerable amount of water with the nucleus being dissolved into the water. The net complex index of refraction used in the Mie calculations for NOVAM is then assumed to be the volume average of the index of refraction of water and the index of refraction of the nucleus material using the method of Hänel [14]. Each of the classes of aerosol in the model are allowed to have their own complex index of refraction. The amount of growth (represented by the f factor) for each class also depends on the chemical composition of the nucleus of that class.

The first class is a nonsoluble aerosol present in NOVAM only when high amp are indicated and is assumed to be dust with mode radius of 0.03 μm. The refractive index for this "dust" species of aerosol is taken from Shettle and Fenn [15] which is based on references [16 & 17]. Since this component is assumed to be nonsoluble, this class of aerosol does not change size with changes in RH; thus, the optical refractive index parameter remains constant with changes in RH.

The next aerosol species is assumed to be water soluble and is represented at an RH of 80% by a lognormal also with a mode radius of 0.03 μm. The chemical species of this aerosol is assumed to be the Volz's general water-soluble aerosol described in reference [16].

These first two classes of aerosols are the background aerosol related to the air mass characteristics and do not seem to respond to local wind parameters. Being air-mass dependent, however, it can be modified when transported over the ocean to the point of reference from the land areas.

The third class of aerosol in the model is represented by a lognormal distribution that has a mode radius of 0.24 μm at the standard RH of 80%. This class is populated by marine aerosols that have been produced by earlier high wind conditions that exhibit a relatively long residence time and remain mixed throughout the MBL once they are introduced into the atmosphere. It

does not make much sense to say that the concentrations of these aerosol are related to the current wind speed. They are, in effect, related to the history of the wind. In the NAM/NOVAM simplification of the problem, the previous 24-hour average of the wind speed is an attempt to tie this class of aerosol to a historical relationship with the wind.

The fourth and last class of aerosols is that of the largest nuclei originating from the sea surface; this can be important in the propagation of IR radiation near the sea surface. These aerosol are represented in the NAM by a lognormal distribution that has a mode radius of 2.0 μm. The amplitude of this mode is definitely related to the current wind speed and the current white water phenomenon.

Finally, the volume extinction coefficient for any size distribution $dN(z)/dr$ and any wavelength within the range of application could be expressed by using the Mie Theory. Here of course, we have to know Q_{ext} for the population of aerosol that is a function of the wavelength, the radius of the sphere, and the index of refraction of the sphere doing the scattering or absorption of the EO propagation. This is expressed in equation (9) as an integration over the whole population of aerosol sizes. A volume absorption coefficient can be calculated in a similar way except that the Mie coefficient Q_{abs} must be used instead of Q_{ext}. The volume extinction coefficient as a function of altitude is

$$\beta_{ext}(z) = \frac{\pi}{1000} \int Q_{ext} \cdot \frac{dN(z)}{dr} \cdot r^2 \cdot dr \quad . \tag{9}$$

If we now substitute into equation (9) our formulation of the size distribution in terms of the sum of four lognormal type functions we get

$$\beta_{ext}(z) = \frac{\pi}{1000} \int Q_{ext} \cdot \sum_{i=0}^{3} \frac{A_i(z)}{f_i} \cdot \exp\left[-1 \cdot \left[\log\left(\frac{r}{(r_{0i} \cdot f_i)}\right)\right]^2\right] \cdot r^2 \cdot dr \; . \tag{10}$$

The integration and the summation can be switched in equation (10) so we really have a sum of several integrals. The volume extinction coefficient is shown in equation (11), and using a similar process, the volume absorption coefficient can also be calculated and this is shown in equation (12).

$$\beta_{ext}(z) = \frac{\pi}{1000} \sum_{i=0}^{3} \frac{A_i(z)}{f_i} \int Q_{ext} \cdot \exp\left[-1 \cdot \left[\log\left(\frac{r}{(C f_i)}\right)\right]^2\right] \cdot r^2 \cdot dr \tag{11}$$

$$\beta_{abs}(z) = \frac{\pi}{1000} \sum_{i=0}^{3} \frac{A_i(z)}{f_i} \int Q_{abs} \cdot \exp\left[-1 \cdot \left[\log\left(\frac{r}{(C f_i)}\right)\right]^2\right] \cdot r^2 \cdot dr \; . \tag{12}$$

This technique speeds up the calculations since the integrals can be calculated earlier and their values stored as numbers in a lookup table. The NOVAM then only needs to address this table and interpolate between values to obtain the desired values. This is done by storing the integrals for four different relative humidities and for 40 different wavelengths for each of the four aerosol modes in a table. NOVAMSR then needs only to do a table lookup and interpolation for values

that do not happen to be one of the stored values. The integration was done using Dave's Mie code [18] for the values of Q_{ext} and Q_{abs}, and the integration was done in decades for the variable r. Each decade was then numerically integrated using Simpson's rule with 90 strips. The results of these integrations are stored in the program as logarithms for simplicity of storage.

FEATURES OF THE NOVAM SUBROUTINE

The NOVAMSR is called with six input parameters and seven output parameters. These are shown in table 1 in the order needed by the subroutine along with a brief explanation and the units needed by the program. A fuller explanation is given below to the types and kinds of data used as input and output.

Table 1. The parameters of the NOVAMSR subroutine.

Input Parameters	Explanation	Units
Altitude	Altitude of calculation	meters
Wavelength	Wavelength of calculation	micrometers
The surface obs. data, **SurfObsFileName**	Name and directory of the surface observation file.	ASCII text
The profile data, **RSondeFileName**	Name and directory of the radiosonde observation file.	ASCII text
DataType	Description of above data file.	"N" or "R"
Repeatflag	Use true to bypass analysis.	LOGICAL "T" or "F"

Output Parameters		
Extinction	Volume Extinction Coefficient	1/km
Absorption	Volume Absorption Coefficient	1/km
A0	Mode 0, amplitude	
A1	Mode 1, amplitude	
A2	Mode 2, amplitude	
A3	Mode 3, amplitude	
RH	Relative Humidity	Percent

INPUT PARAMETERS

Altitude

This parameter tells NOVAM at what altitude (in meters) the calculation is to be made. There are restrictions in the values which this altitude can have based on the validity of the model. Altitude must be between 9 meters and 6000 meters or the model will return an error flag of all –999.9 data values on its call. At high altitudes, the upper atmospheric aerosol models developed by the U. S. Air Force [19] should be used even over the oceanic regions.

Wavelength

This parameter tells NOVAM at what wavelength the scattering and absorption analysis is to be calculated. It is expressed in micrometers and can range from 0.2 μm to 40 μm. The subroutine will return an error flag of all –999.9 data if the wavelength is outside of this range.

Surface Observation File

One of the required input files contains a set of shipboard-level surface observation data. The kind, units, and order of these data are shown in table 2 and are self-explanatory except for certain items that are explained below. These values are to be provided in the units given and if these observations are not available, a value of –999.0 indicates to the program that these data are indeed missing and it will provide a default value. This surface data file has fewer elements than the earlier files used with NOVAM codes. Therefore, caution must be taken to convert earlier files to work with NOVAMSR. The data in the file must be real and in ASCII format. The individual items in the list must be separated from each other by spaces. In addition, for values <1, the decimal point must be preceded by a zero.

Table 2. Surface observation data file.

Position	Meteorological Parameter
1	Sea Surface Temperature (C)
2	Air Temperature (C)
3	Relative Humidity (%)
4	Optical visibility (km)
5	Current real wind speed (m/s)
6	Averaged wind speed [24 hours] (m/s)
7	Air Mass Parameter (1..30)
8	Surface IR ext. (1/km) @ 10.6 μ
9	Zonal/seasonal category (1..6)

Item 7 is the amp which is a number used to describe the general condition of the air mass. It is a real number between 1 and 30, where 1 indicates a pure air mass without contaminants and 30 describes a worse case situation such as would be expected downwind of a large industrial complex. It can be based on empirical judgment, or it can be related to certain measurements in the MBL such as the atmospheric radon content, Rn, (expressed in (pCi/m^3)) as

$$amp = Rn/4 + 1 \tag{13}$$

or it can be related to t, the elapsed time for the current air mass to reach the point of observation in the sea from a distant land mass. With this information, the expression is

$$amp = \{9 \cdot \exp(-t/4)\} + 1 \tag{14}$$

where t is expressed in days.

If the amp is not known, entering the value of –999.0 will permit NOVAM to estimate an amp for the run by using other data. In the case where sea surface visibility is available, we can calculate an "effective" amp using a simplified three-component NAM and the measured visibility.

This method assumes that the extinction in the marine atmosphere is because aerosols are composed of three kinds of aerosols distinguished by their origin. All three are found over the ocean in varying degrees. The model assumes that the size distribution of the three aerosol classes can each be described by an independent lognormal function. The net size distribution responsible for scattering and absorbing the optical/IR energy as it transverses a cloud of these

aerosols is the superposition of the three classes. Given this net aerosol size distribution, the extinction and absorption of optical/IR radiation can be calculated by using the Mie Theory as discussed. The assumptions here are that the aerosols are spherical in shape and that we know their index of refraction. Furthermore, if the aerosol nuclei are soluble as is the case with most marine aerosol, then the size of the aerosol will change as the RH of the surroundings change. Therefore, it is assumed that the final index of refraction of the mixed aerosol is a volume related combination of water and the nuclei's index of refraction.

The following analysis shows how an approximate amp can be determined given the current wind speed, average wind speed, RH, and visibility. First, a differentiation must be made between the total extinction at visible wavelengths and the extinction at visible wavelengths due to only aerosols. These differ by the Rayleigh extinction because of atmospheric molecules. While the Rayleigh extinction can vary somewhat depending on temperature, pressure, etc., we will use the value for Rayleigh scattering for air at 288.15 °K and 1013 mb which is β_m = 0.01162 (per kilometer), from E. J. McCartney's book [20].

Thus, if we have a measured extinction based on a scattering or transmission measurement, or on a direct visibility measurement, it will include the Rayleigh extinction. This measured visibility, Vis_λ, is used to determine the amp. Then for λ = 0.55 μm, using the Koschmieder [21] formula for the relation between visual range and total extinction, we get

$$\beta_\lambda = \frac{3.912}{Vis_\lambda} \quad . \tag{15}$$

The aerosol part of the above extinction, $\beta_{\lambda 0}$, is the value defined as the total extinction less the Rayleigh extinction, i.e., $\beta_{\lambda 0} = \beta_\lambda - \beta_m$. When λ = 0.55 μm (visible wavelengths), we get

$$\beta_{\lambda 0} = \frac{3.912}{Vis_\lambda} - 0.01162 \quad . \tag{16}$$

The NAM assumes that each of the three components of the aerosol size distribution can be described by a lognormal function as

$$\frac{dN_i}{dr} = A_i \frac{\exp\left\{-1 * \left[\log\left(\frac{r}{r_i * f}\right)\right]^2\right\}}{f} \quad , \tag{17}$$

were the function A_i is defined for each of the i's as

$$A_1 = 2000 \cdot amp^2 \quad , \tag{18}$$

$$A_2 = \text{MAX}[5.866 \cdot (\overline{w} - 2.2), 0.5], \tag{19}$$

$$A_3 = 10^{(0.06 \cdot w' - 2.8)} \quad , \tag{20}$$

where $\overline{w}$, and w' are the average and current wind speeds in m/s, and *amp* is the unknown amp. The constants r_1, r_2, and r_3 are 0.03, 0.24, and 2.0, respectively.

The size parameter, f, is here for simplicity in this approximation to be the same for all classes of aerosol and can be expressed as a function of RH for all relative humidities less than or equal to 98% (Fitzgerald [22]).

$$f = \left[\frac{2 - \frac{RH}{100}}{6 \cdot \left(1 - \frac{RH}{100}\right)} \right]^{1/3} \quad . \tag{21}$$

It has the value of "1" at an RH of 80%, which is the typical RH found over the world's ocean.

The functional form of the size distribution can then be written in a simpler form as

$$\frac{dN_i}{dr} = \frac{A_i}{f} \cdot S_i(f, r) \quad . \tag{22}$$

where

$$S_i(f, r) = \exp\left\{ -1 * \left[\log\left(\frac{r}{r_i * f} \right) \right]^2 \right\} \quad .$$

The original assumption that $\overline{w}$, w' and RH are known quantities means that all of the parameters necessary to describe the size distribution of modes 2 and 3 are known. The size distribution of mode 1 is still undetermined since the value of *amp* is not known at this point.

If we use the Mie Theory, we can calculate the volume extinction coefficient for each of these classes of aerosol for a wavelength of 0.55 μm as

$$\beta_{i\lambda} = \frac{\pi}{1000} \int Q_{e\lambda} \cdot \frac{dN_i}{dr} \cdot r^2 \cdot dr \quad . \tag{23}$$

Substituting for dN/dr we get

$$\beta_{i\lambda} = \frac{\pi \cdot A_i}{1000 \cdot f} \int Q_{e\lambda} \cdot S_i(f, r) \cdot r^2 \cdot dr \quad , \tag{24}$$

and the integrals $\int Q_{e\lambda} \cdot S_i(f, r) \cdot r^2 \cdot dr$ are calculated as described above. There is a separate value for each mode and RH. Using these values of RH for each mode, the volume extinction coefficient can then be expressed as

$$\beta_{i\lambda} = \frac{\pi \cdot A_i}{1000 \cdot f(\mathrm{RH})} \cdot T_i(\mathrm{RH}) \quad . \tag{25}$$

Since for all of these cases, $\lambda = 0.55$ μm, we can find three functions, T_i(RH) of RH which express these integrals for each aerosol mode. These functions were obtained by finding the numerical value of the Mie integrals for four relative humidities of 50, 85, 95, and 99%. The curves were fitted with the relatively simple functions with the assistance of a special software package [23]. This package fits the input data points to over 7000 different equations and ranks the results as to the least error in the fitting. The fittings here had the resultant equations for each of the modes within the first four ranks. An example of the fitting of the data for mode 1 is shown in figure 1.

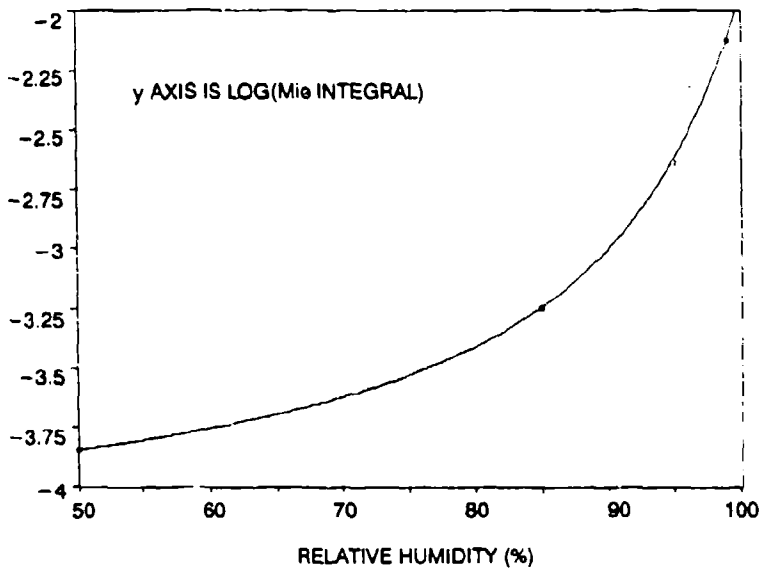


Figure 1. Plot of the fit of the T_1(RH) function to the Mie integral calculations for the mode 1 aerosol at λ = 0.55 μm. Note that for convenience, the y axis is the log10 of the integral values.

The equations for each mode are shown in equation (26).

$$T_i(RH)\frac{(a + c \cdot RH)}{(1 + b \cdot RH)},\ T_2(RH) = \frac{(d + g \cdot RH)}{(1 + e \cdot RH)},\ T_3(RH) = \frac{(l + n \cdot RH)}{(1 + m \cdot RH)} \qquad (26)$$

where a = –4.05801664, b = –0.00890166, c = 0.038432675
d = –0.64465936, e = –0.00899986, g = 0.007232437
l = 2.019394568, m= –0.00900429, and n = –0.01670367.

The measured total volume extinction due to aerosols can be equated to that calculated by the model as

$$\beta_{\lambda 0} = 2\pi \cdot \frac{amp^2}{f(RH)} \cdot T_1(RH) + \beta_{2,\lambda=0.55}(RH, \overline{w}) + \beta_{3,\lambda=0.55}(RH, w'), \qquad (27)$$

where we have only the parameter *amp* as an unknown. The unknown can be determined algebraically as

$$amp = \sqrt{\frac{\beta_{\lambda 0} - \left(\beta_{2\lambda=0.55}(RH, \overline{w}) + \beta_{3\lambda=0.55}(RH, w')\right)}{2\pi \cdot T_1(RH)/f(RH)}} \qquad (28)$$

In order to keep *amp* from becoming imaginary, $\beta\lambda_0$ must be equal to or greater than the sum of the extinctions of the second and third mode,

i.e., $$\beta_{\lambda 0} \geq \beta_{2,\lambda=0.55} + \beta_{3,\lambda=0.55}\ .$$

In addition, we know from measurements, that *amp* should never be zero; therefore, the following logical "IF-THEN-ELSE" evaluation is added to the relationship.

$$\text{IF} \qquad \beta_{\lambda 0}(VIS) \le 0.02\pi \cdot \frac{T_1(RH)}{f(RH)} + \beta_{2,\lambda=0.55}(RH,\overline{w}) + \beta_{3,\lambda=0.55}(RH,w') \tag{29}$$

$$\text{THEN} \qquad amp = 0.1$$

$$\text{ELSE} \qquad amp = \sqrt{\frac{\beta_{\lambda 0}(VIS) - \left(\beta_{2,\lambda=0.55}(RH,\overline{w}) + \beta_{3,\lambda=0.55}(RH,w')\right)}{2\pi \cdot T_1(RH)/f(RH)}} \; .$$

The *amp* can now be plotted as a function of visibility for various sets of the meteorological parameters such as RH, average wind speed, and current wind speed. Figure 2 shows the relationship between *amp* and visibility when there is no wind speed either average or current. This is the "glassy sea" stage. Here, there is a small residual component of the mode 2 and mode 3 aerosols, but they are not being produced by wind-wave interaction. In this case, *amp* is a straightforward function of the visibility.

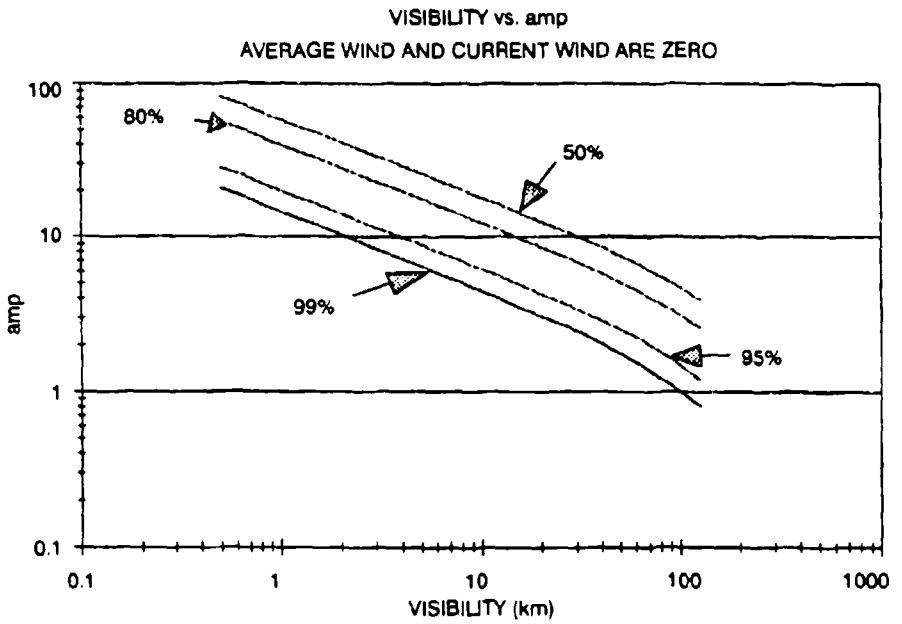


Figure 2. Plot of the visibility and amp relationship from this model for various relative humidities and with no current or average wind speeds. Note that there are amps which can be greater than 10 for certain conditions.

The other extreme case is shown in figure 3 where we have strong concentrations of aerosol being produced at the sea surface due to interaction of the wind and waves. In this case, there is a significant extinction at the visible wavelength of 0.55 μm produced by aerosol from the sea surface, which must be taken into account. If our measured visibility has an aerosol extinction value less than the extinction from modes 2 and 3, there is a logical problem since the total cannot be less than the sum of its components. In this case, either the measurement is in error or the parameterization of modes 2 and 3 components of the model is wrong. In either case, when this impossible situation occurs for whatever reason, the *amp* is arbitrarily set to 0.1.

Item 8 in table 2 is a parameter that gives the model more information on which to base its predictions. It is used to describe the IR extinction at the surface at the wavelength of 10.6 μm. It is anticipated that data from such a device (several are now available on the commercial market) will greatly improve the overall operation of the model. It will do this in conjunction with the optical visibility by pinning down the surface extinction at two widely separate wavelengths.

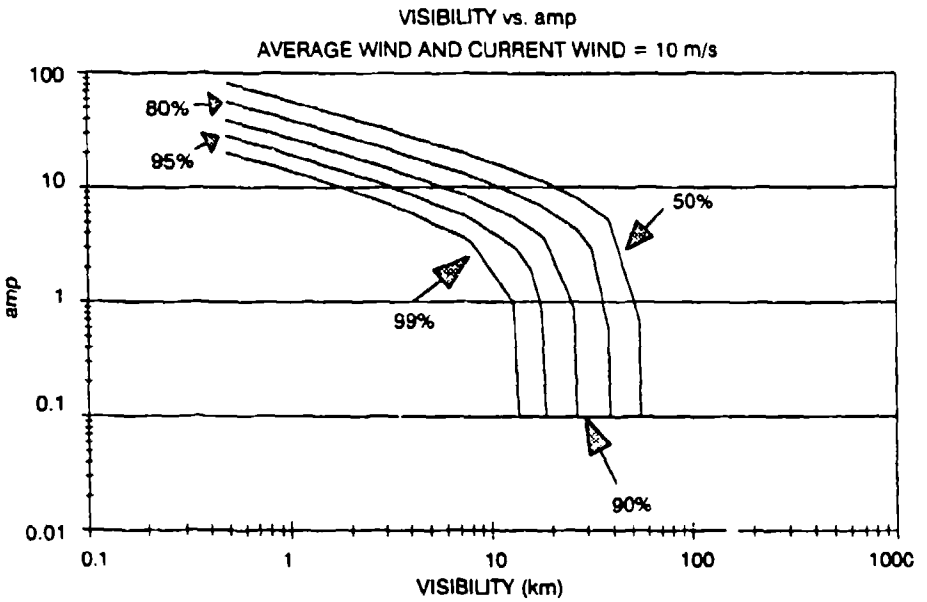


Figure 3. A plot of the visibility and amp relationship from this model for various relative humidities and with the average wind speed and current wind speed set at 10 m/s. The plot reflects the condition that the amp can never be less than 0.1.

Item 9 in table 2, the zonal/seasonal category, is a number between 1 and 6 that signals to the computer an approximate geographical zone where the calculation is to be used. This is used in the default definitions where data from a climatic atlas [24] are used. If a value outside the limits of 1 to 6 is used, the model assumes that a mid-latitude summer is the situation where the calculation is being made. The number is selected from table 3.

Table 3. Zonal/global categories.

1	Tropical winter
2	Tropical summer
3	Mid-latitude summer
4	Mid-latitude winter
5	Sub-Arctic summer
6	Sub-Arctic winter

Radiosonde Observation File

This is an ASCII file that contains the available information on the temperature and humidity throughout the marine atmosphere. It can be obtained in several different ways. Profiles of the vertical meteorological structure of the atmosphere are traditionally obtained with the

radiosonde. This is a device that senses pressure, temperature, and RH and is lifted aloft by either kites, free balloons, or tethered balloons. The measurements are sent via radio to a ground receptor that translates the signals into profile data. Other methods are also available for obtaining this information such as from an instrumented aircraft. This method was used in the NOVAM testing processes where the aircraft flew in a tight spiral either ascending or descending. NOVAMSR is set up so it will accept two kinds of profile data files. The files structures for each of these types are described under the data types section.

Data Type

There are two types of data format for the radiosonde observation file that are recognized by NOVAMSR. They are

1. The NOVAM form (N) consists of a "n $\times$ 3" matrix where "n" is the number of observations made. Each row of this matrix consists of the altitude at which the measurement is made and expressed in meters, the potential temperature, Θ, at that altitude expressed in °C, and the mixing ratio at that altitude expressed in g/kg. An example is given in table 4.

Table 4. A NOVAM type radiosonde profile in the "n" mode.

Altitude (m)	Θ (C)	Mixing ratio (g/kg)
16.510	14.852	8.640
28.630	14.642	8.850
39.940	14.504	8.780
48.710	14.383	8.820
61.340	14.357	8.770

2. Another form of profile data (R) can be used, which is the output of a PP11 from a RS-80 RAOB SONDE system. It has the form of a "n $\times$ 5" matrix that contains the following information in each of its rows: observation number, the log (base10) of the pressure multiplied by 10^4, the air temperature in °C multiplied by 10, the RH in percent, and the pressure in millibars multiplied by 10. An example is given in table 5. The calling parameter should be either an "N" or "R" to indicate to NOVAMSR which type of data to expect when it opens and tries to read the radiosonde observation file.

Table 5. An example of the radiosonde profile in the "R" mode.

1	30043	154	81	10099
2	30037	152	81	10086
3	30033	150	83	10076
4	30027	149	83	10063
5	30022	148	84	10050

Repeat Flag

The repeat flag is a logical parameter that is used to eliminate repeated analysis of the radiosonde profile every time the NOVAMSR is called. It should be "false" the first time a particular radiosonde data set is run. When this happens, the radiosonde preamble is calculated from the input data and appended in front of the significant data file. This information is then used in this and subsequent calculations with this particular radiosonde data set. The next time a call is

made to this subroutine with this radiosonde data set, (assuming no other radiosonde data sets have been used in the meantime) the repeat flag should be set to "true," and the duplicate analysis needed in the first call to the subroutine will be eliminated.

OUTPUT PARAMETERS

Relative Humidity

This is the RH obtained from the analyzed radiosonde profile data at the referenced altitude. This humidity may differ somewhat from the raw measure of RH at that altitude because it is the humidity that the model obtained from the smoothed radiosonde data. It is based on the linear interpolations of air temperature and mixing ratio obtained from the nearest significant level above the reference altitude and the nearest significant level below the reference altitude. Should the reference level be identical to one of the significant levels, then only the values at that significant level are used. This humidity is provided so it and the associated size distribution parameters will describe the same aerosol size distribution used by the model at the reference altitude.

Size Distribution Parameters

The size distribution parameters produced by the subroutine refer the model-generated A0, A1, A2, and A3 terms of equation (8). Each of these reflect the "amplitude" of the lognormal value of a particular component of the *dN/dr* size distribution, but it has not been adjusted for RH at that particular height. An *f* factor must be calculated from the RH for each component if the actual *dN/dr* of the aerosol size distribution is to be used. Given the size distribution, *dN/dr*, various other calculations of optical properties of the aerosol can be made from a knowledge of the Mie Theory and an assumption of the index of refraction of the droplets.

Extinction and Absorption Coefficients

The extinction coefficient given by NOVAMSR is related to the size distribution at the reference altitude and is determined by the superposition of the lognormal components, which are represented by amplitudes A_0, A_1, A_2, and A_3. The growth factor f_i is different for each component and is given by Gerber's [25] formulation. The actual method of calculation of this coefficient is shown in equation 11.

Here, it is assumed that the A_0 term is made up of a nonhygroscopic material and does not grow with increased RH. Hence, the *f* factor for the A_0 term is always 1.0. On the other hand, we assume that the remainder of the lognormal populations of aerosols are hygroscopic.

Gerber formulates *f* as

$$f = \left[\frac{C7-S}{C8(1-S)}\right]^{1/3} \tag{30}$$

where the values of $C7$ and $C8$ are given in table 6. For amp greater than 5, we assume that composition of the smallest size aerosol is composed of two groups of particles. One which has an *f* of 1 and is not hygroscopic and composes about 30% of the aerosol, while the remainder of the aerosol is assumed to be made up of an urban aerosol. The two sea salt components differ only slightly from each other.

In a similar way the absorption coefficient is calculated in equation (12) with the *f* factor described above.

Table 6. Constants for equation (17) and the range of relative humidity (RH) validity. Gerber [25].

Aerosol Model	NAM Component	C7	C8	Range of Validity
Sea Salt	2	1.83	5.13	RH<99.9%
Sea Salt	3	1.97	5.83	RH<99.99%
Urban	1	1.28	2.41	RH<99%
Rural	1	1.17	1.87	RH<99%

A TEST DRIVER FOR NOVAMSR

A driver is included (see appendix A) that illustrates the use of NOVAMSR. This is a simple program that makes a profile of the NOVAM outputs at altitudes of 10 meters, 100 meters, and 200 meters, etc., up to the end of the available meteorological data. The use of the logical parameter REPEATFLAG is illustrated where it is "false" for the first call to the subroutine and "true" for the following calls to it. The information needed for the operation of NOVAMSR is included in the file NOVAM.INI, which contains the file names of the surface observation file, the meteorological profile file, the type of data to expect the profile data to be in, and the wavelength at which the calculations are to be made. The operation of this driver is illustrated in figure 4. Once started, the program uses the file NOVAM.INI, operates the NOVAMSR and prints the results in the OUT1 file, and then ends. A detailed view of this operation is shown in figure 5 in which the NOVAMSR is seen using the information in the surface observation file and in the meteorological profile file. The names of these files are given to the driver program via the NOVAM.INI file and then the driver supplies the names to the NOVAMSR by means of the parameters exchange. The driver must call NOVAMSR a number of times to make a complete profile. Each call to NOVAMSR returns one set of parameters for one altitude and one wavelength.

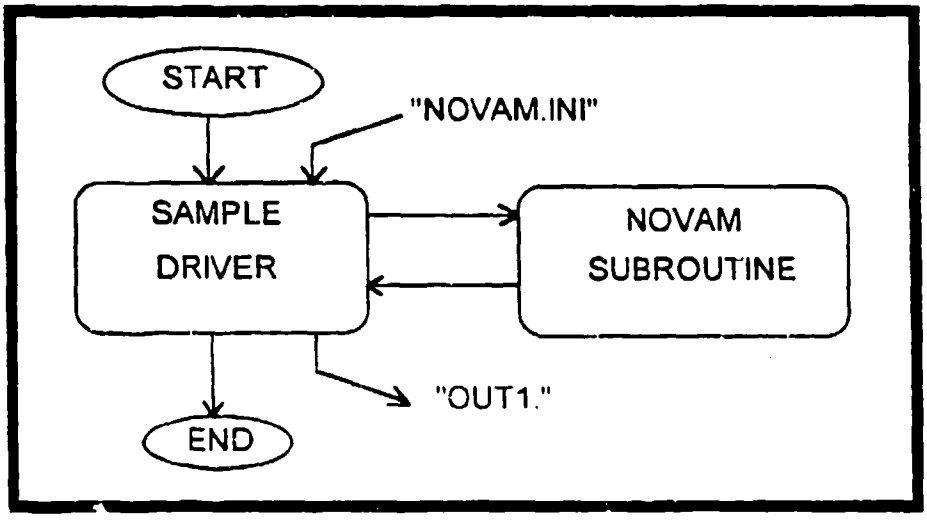


Figure 4. A flowchart showing the operation of the subroutine NOVAM under the control of a driver.

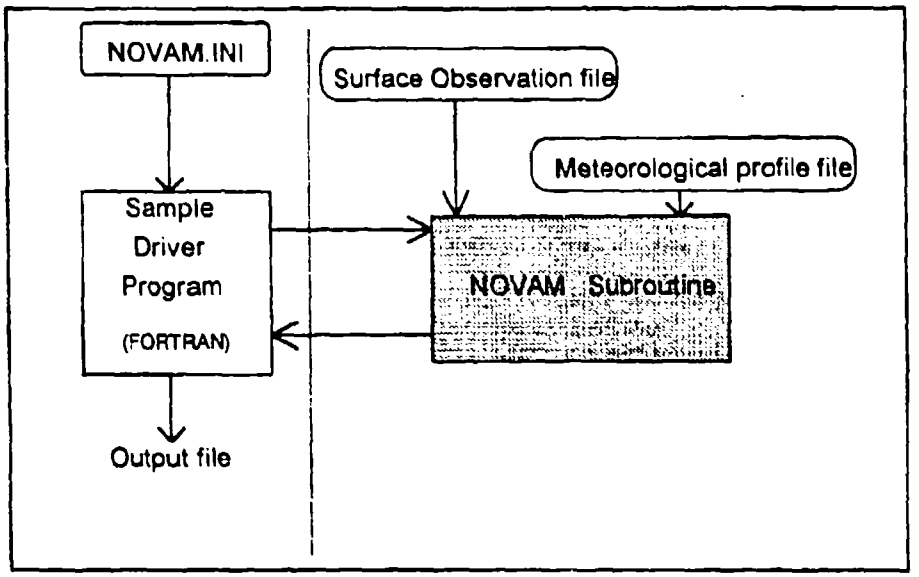


Figure 5. Flowchart showing the flow of information in a simple program which utilizes a driver and the NOVAM.

THE STRUCTURE OF NOVAMSR

The structure of the subroutine is shown in simplified form in figure 6. The structure shows the general flow of the program including how the "repeat" flag directs the program around the parts of the code that performs the conditioning and the analysis of the radiosonde data when a repeated calculation of this would be redundant. This allows for the bypassing of unneeded programming and shortens the net processing time required to obtain a series of runs with the same set of input data. The chart also shows how one of three paths is selected for the actual modeling of the vertical structure depending on the values contained in the preamble analysis.

One problem of the flowchart type of presentation shown in figure 6 is the production, manipulation, and destruction of data files is not shown. Therefore, this aspect of the program must be described with words. The input data list for the program is given in table 1, and in this list are two file names that contain the information on which NOVAM calculation is based. The structure of the surface observation file discussed in detail above and sample structures of the radiosonde data file are given in tables 4 and 5. The profile data must be such that the altitude values be monotonically increasing for the type "N" file or the pressure be monotonically decreasing for the type "R" file. The type of file "N" or "R" must also be given as the datatype parameter of the subroutine. If the data are of the "N" type, they contain altitude, potential temperature, and mixing ratio for each observation. These data must be converted into a type "R" file so that the data conditioning can take place. Thus, if the radiosonde data are of the "N" type, an "R" type is generated called PATRH.DAT and is in the local directory. On the other hand, if the data are already of the "R" type, then they are copied into the file called PATRH.DAT for conditioning and analysis.

At the conclusion of the filtering, conditioning, and analysis section of NOVAMSR, a file called SIGFILE is generated that contains values of pressure, air temperature, and RH for levels at which significant changes take place. The first line of SIGFILE contains a single number that is the number of significant levels contained in the file.

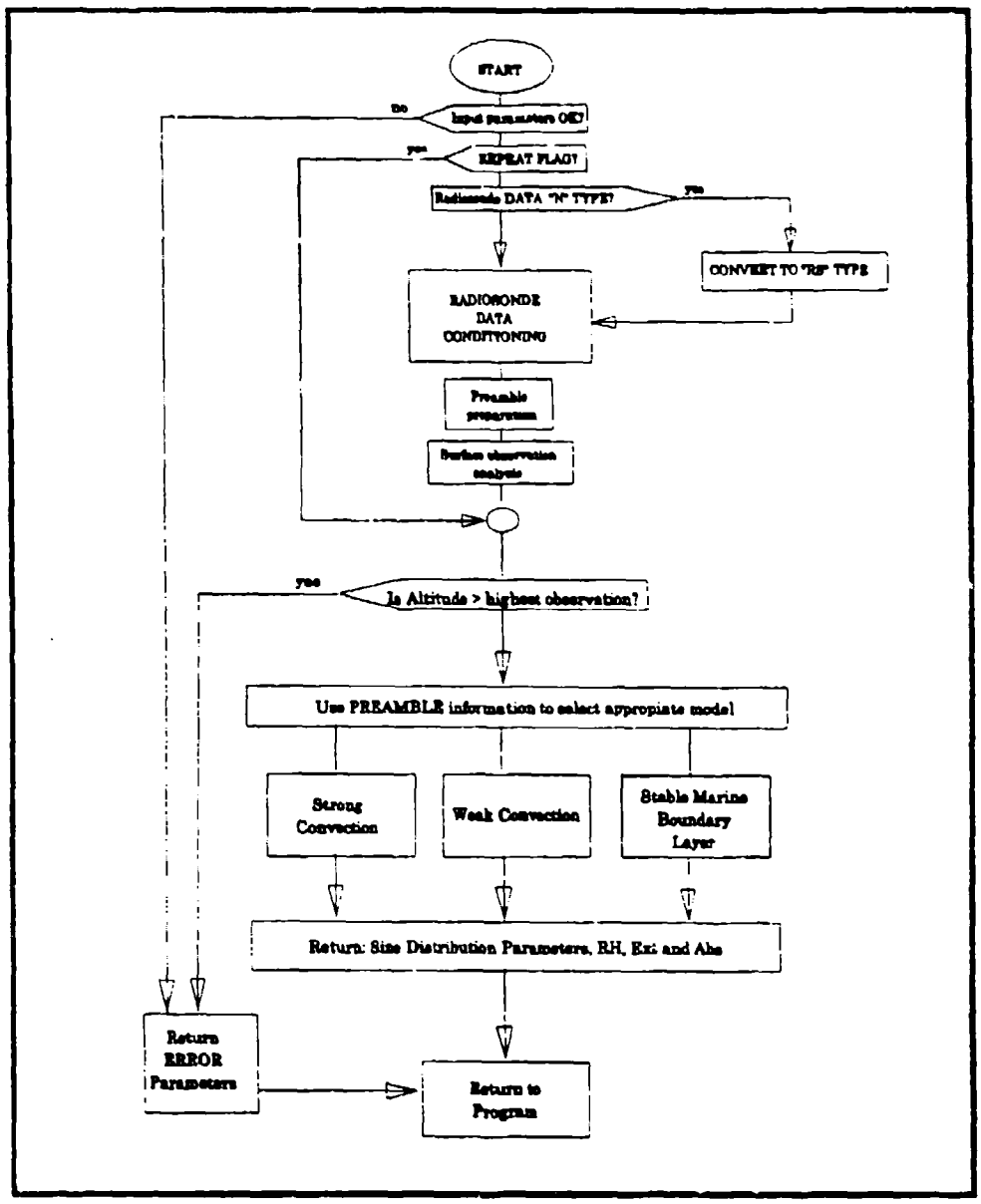


Figure 6. A simplified flowchart of NOVASMSR.

The next step in the conditioning process is to use the data in SIGFILE to obtain the profile parameters needed by NOVAM. This implies the detection of the existence and number of significant air temperature inversions, and the determination of information about altitude, potential temperature, and mixing ratio of points where the inversion starts and ends. This preamble information is added to the beginning of an internal profile data in an array called RDATAARY, which is similar in structure to the previous NOVAM input radiosonde files. This concludes the preliminary conditioning, filtering, and analysis part of the program. This part may be bypassed if the "repeat flag" parameter of the NOVAMSR input is "true." The internal array called RDATAARY contains all of the radiosonde profile information needed by NOVAM to make a calculation.

The first step of the actual NOVAM calculation is to make the selection of which of the three models to use. This decision is based on the information in the preamble part of RDATAARY. If there is no inversion detected, then the "strong convection" case is selected. If there are two air temperature inversions detected, then the "weak convection" code is selected, and if there is only one inversion detected, then the "stable marine boundary layer model is used." In each of these models, the size distribution parameters, the RH, and the extinction and absorption at the wavelength and altitude requested is calculated and returned to the calling program.

Filtering

The first thing that is done in the NOVAMSR is to condition the raw radiosonde data to eliminate spurious observations and noise that often accompany raw data. This is accomplished by employing a series of numerical filters to smooth the pressure, air temperature, and RH data points found in the file "PATRH.DAT." Much of the programming code is patterned after the data-smoothing programs found in Press et al. [26]. After the data have been sufficiently smoothed, the meteorologically significant levels and the data associated with them are sent to a file called "SIGFILE" as shown below.

"PATRH.DAT" -----> [FILTER] -----> "SIGFILE"

Determination of Profile Parameters

The profile parameters are determined by the subroutine called PREAMB. This subroutine reads the conditioned parameters, pressure, air temperature, and RH from the "SIGFILE" profile data file. The logic behind this program is that it looks for significant temperature inversions in the SIGFILE data profile and uses meteorological values associated with these inversions to fill a (5,3) matrix preamble (see table 7) is inserted ahead of the altitude, potential temperature, and mixing ratio data used by the NOVAM models. The elements of the preamble preceding the actual radiosonde observations are parameters that describe the structural characteristics of the profile. This discussion involves the potential temperature and the mixing ratio—two meteorological parameters that change in altitude for a well-mixed situation. The structure of the profile is represented by a multisegmented line fitted to the SIGFILE profile data points.

Table 7. The NOVAM preamble.

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)
(4,1)	(4,2)	(4,3)
(5,1)	(5,2)	(5,3)

Starting at the lowest levels and going up, temperature inversions are looked for. For an inversion to be "significant" for this program, a certain amount of logical analysis must be done on the data before they are accepted as a real inversion. The actual code required to do the analysis is recorded in the APPENDIX under the subroutine PREAMB. This analysis essentially tries to fill the preamble matrix with the appropriate data starting with element (1,1) that contains the number that is five more than the number of data points in SIGFILE. Elements (1,2) and (1,3) contain the surface observations of the air temperature (potential temperature at the sea surface), and the mixing ratio, respectively. The process then continues to search for one or two

significant temperature inversions and to record the appropriate sounding data into the other parts of the preamble matrix. This is done by making a gradient profile calculated by $\Delta T/\Delta p$ for each two adjacent data points and recording the value as long as the absolute value of ΔT is greater than 0.19°C and the absolute value of the pressure step, Δp, is greater than 0.9 mb. Values less than this are defined as 0.0. This assures us that the measured data differences are greater than those due to instrument error. Finally, a series of negative $\Delta T/\Delta p$ values are only accepted if the net increase in temperature from the bottom to the top of the inversion is greater than 1.6°C. When the first legitimate temperature inversion is found, then the elements (2,1), (2,2), (2,3), (3,2), and (3,3) are recorded in the preamble matrix. See figure 7 for an illustration of the meaning of the matrix elements on a stylized segmented type "N" profile plot. Element (2,1) contains the altitude the inversion starts, element (2,2) contains the potential temperature just below the inversion, and element (3,2) contains the potential temperature just above the inversion. In a similar way, element (2,3) contains the mixing ratio just below the inversion and element (3,3) contains the mixing ratio just above the inversion. A similar searching process is continued as altitude increases until either the end of the data is reached or a second inversion is found. If the second inversion passes the criterion discussed above, then the matrix elements (4,1), (4,2), (4,3), (5,2), and (5,3) are also recorded. Element (4,1) contains the altitude the second inversion starts. Element (4,2) contains the potential temperature just below the inversion, and element (5,2) contains the potential temperature just above the inversion. In a similar way, element (4,3) contains the mixing ratio just below the inversion and element (5,3) contains the mixing ratio just above the inversion.

When there is only one inversion found, the data elements (4,1), (4,2), (4,3), (5,2), and (5,3) all contain the value –999.0. This is the simple MBL case. Finally, if no temperature inversions are found, then elements (2,1), (2,2), (2,3), (3,2), (3,3), (4,1), (4,2), (4,3), (5,2), and (5,3) all contain values of –999.0. This is the case where convective activity is usually in operation. This information in the preamble is used to decide which of the three model to use in a particular calculation.

Almost Well-Mixed Boundary Layer Model (One Inversion)

The important feature of this model is it allows gradients to exist in the MBL for passive scalar contaminants such as marine aerosol. Furthermore, these gradients are calculated from the available meteorological parameters using the various scaling laws developed by Fairall and Davidson [27]. Using these ideas, we calculate a gradient in the concentration, dA/dz, of marine aerosol in the boundary layer to be

$$\frac{dA}{dz} = \frac{-1.5}{h \cdot w_*}(S - V_d + 2.5 \cdot W_e(A_{dl} - A_b)) \tag{31}$$

where

(1) "h" is the height of the MBL, which is the height of the inversion and is stored in element (2,1) of the DATAARY.

(2) w_* is the convective scaling velocity calculated from the bulk micro meteorological scaling equations based on information from the surface observation file. This is calculated as "**wstr**" in the subroutine WE2.

(3) S is the source terms of the aerosol at the sea surface. This model uses the table lookup routine in WHITEFLUX based on observations by Fairall, Davidson and Schacher [28].

(4) V_d is the dry deposition velocity or fallout term of the aerosol at the sea surface. This calculation is based on the theory of Slinn and Slinn [29] and modified by Fairall and Davidson [27].

(5) W_e is the entrainment velocity at the top of the MBL, and is based on measured information about the boundary layer contained in the surface observation file and the preamble to the profile data. This is calculated in the WE2 subroutine.

(6) A_{dl} is the concentration of aerosol size at deck level (in this case it is determined by the four-component NAM).

(7) A_b is the concentration above the MBL of that class of aerosol. This will be set to zero for sea salt-generated aerosol and to the surface value for the background aerosol.

These values are calculated by the various methods and put together into equation (31) to determine the gradients of the various modes.

Given the calculated gradients of each of the four aerosol types, as well as their concentration at the surface, this model gives the concentration of any atmospheric constituent as a function of altitude within the MBL.

$$A_i(z) = A_i(0) + z \cdot \frac{dA_i}{dz} \quad . \tag{32}$$

The realistic assumption is that the concentration of the aerosol can never be negative. It is further assumed that the concentration of the sea salt aerosol above the boundary layer is zero and that the background aerosol has the same value at altitudes above the MBL as it had at the MBL. It is also assumed that the background aerosol has no source at the sea surface.

This technique can estimate the concentrations of different aerosol classes for which surface concentrations are available at any altitude, z. The total aerosol size distribution at z consists of three or four lognormal curves representing the size distribution of the classes of aerosol produced at the sea surface, modified by the concentration determined by the different gradients and the altitude. Given this size distribution at z and the measured or interpolated RH at z obtained from the atmospheric sounding, the aerosol size distribution adjusted for an RH is determined. At this point, the optical properties of the aerosol can be obtained by invoking the set of simplified conversion algorithms that gives the optical properties of the parcel of air at that particular level. This process (described in equation (20)) is a rapidly executable table lookup routine that requires the databases of precalculated Mie computations for lognormal distributions of the various types of aerosols for its operation.

Weak Convection Model (Two Inversions)

This model describes the situation in which scattered cumulus clouds may be present and the well-mixed layer is confined to the region below the cloud base. In the region above the cloud base and below the cloud tops, strong vertical gradients of aerosol concentrations may be observed, Davidson and Fairall [30]. They show the profile of concentrations of various classes of aerosol are dependent on various meteorological parameters. These parameters are determined from the structural characteristics of the potential temperature and the mixing ratio profiles obtained from radiosonde profiles. Figure 8 shows a simplified profile of the aerosol concentrations.

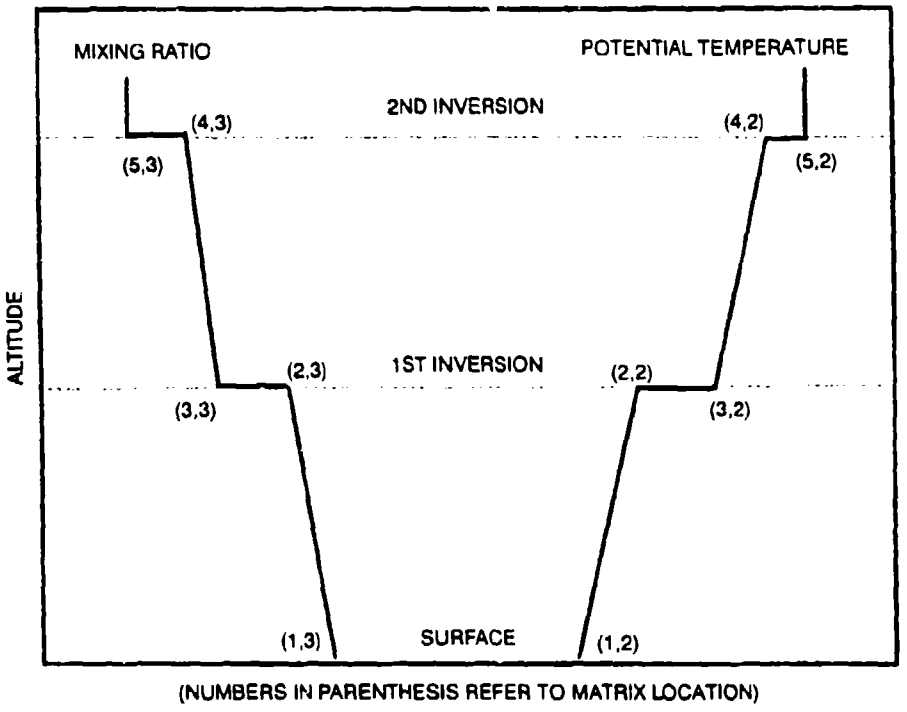


Figure 7. Stylized profile definitions for use in the preamble. The figure represents the segmented lines that compare data plots of mixing ratio and potential temperature against altitude. Three altitudes are shown on the figure, with the base of the plot being zero. Two values are to be placed in the preamble matrix from data at this level. The surface values of the potential temperature (as measured by the sounding device) is inserted in position (1,2) of the preamble; the surface value of the mixing ratio is put into position (1,3).

Our data, together with the techniques of Davidson and Fairal [30], are used by the NOVAMSR to determine the gradient of the concentration between the two inversions. The simplifying assumption is made that for each particular class of aerosol, the aerosol concentration at the sea surface is the same as at the top and bottom of the cloud base level. With this assumption, the concentration gradient is given by

$$\Gamma = \frac{(A_{i+} - A_r) \cdot n}{(1 + n) \cdot \delta z} \tag{33}$$

where

$$n = \frac{(\delta q_b \cdot \Gamma_\theta - \delta\theta_b \cdot \Gamma_q) \cdot \delta z}{(\delta\theta_i \cdot \delta q_b - \delta q_i \cdot \delta\theta_b)} \quad . \tag{34}$$

δq_b and $\delta\theta_b$ refer to the "jumps" in the mixing ratio and the potential temperature at the lower inversion; δq_i and $\delta\theta_i$ refer to the "jumps" in the same quantities at the upper inversion; and δz refers to the depth of the layer. A_r is the concentration of the particular aerosol mode at the surface, and A_{i+} is the concentration above the top inversion.

These quantities are calculated by the computer from the data in the radiosonde preamble. For instance, δq_b refers to the jump in the mixing ratio values and is calculated from the matrix

element points in figure 7 as R[4,3] – R[5,3]. Likewise, Γ_q refers to the slope in the mixing ratio in the region between the inversions and can thus be calculated from the various matrix element terms.

The concentration profiles needed by NOVAMSR are then calculated with the gradient Γ of equation (33) and with the surface values of the four modes of the aerosol concentration obtained with the four-mode NAM at the surface. Assumptions about the aerosol concentration above all of the inversions are made in the following manner: sea salt aerosols do not make it above the second inversion; therefore, the A2 and A3 terms should be zero in this region. On the other hand, the background aerosols A1 and A0 (if it exists) are known at the surface but can be estimated at altitudes above $z2$ in figure 8.

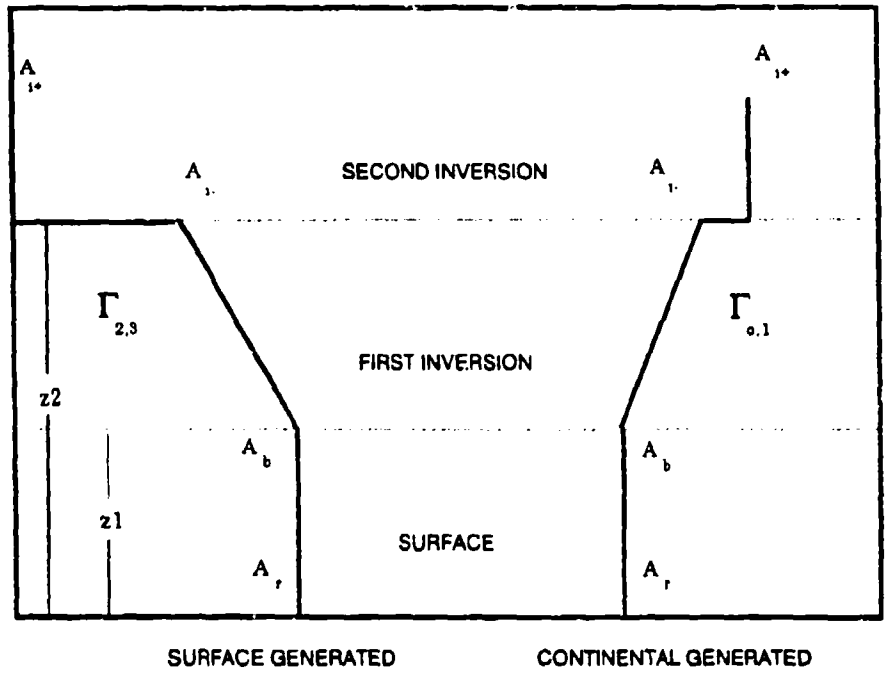


Figure 8. Simplified aerosol concentration profiles. When the index "i" is 0 or 1, then the continental-generated aerosol profile applies. When the index "i" is 2 or 3, then the surface-generated aerosol profile applies.

It is assumed that A_{0-} and A_{0+} are the same and that A_{1-} and A_{1+} are the same. It is also assumed that the region below $z1$ is well mixed forcing A_{ir} and A_{ib} to be the same for i = 0,1,2,3. This leads to the description of $Ai(z)$ as

$$\begin{array}{lll} \text{if } 0 < z < z1 & A_i(z) = A_i(0) & \text{for } i = 0,1,2,3 \\ \text{if } z1 <= z < z2 & A_i(z) = A_i(0) + (z - z1)^*\Gamma_i & \text{for } i = 0,1,2,3 \\ \text{if } z2 <= z & A_i(z) = A_i(0) + (z2 - z1)^*\Gamma_i & \text{for } i = 0,1 \\ & A_i(z) = 0 & \text{for } i = 2,3 \end{array} \qquad (35)$$

Given these relationships for the height variation of the concentration of the four aerosol modes, they can be directly substituted into equations (11) and (12) to provide the optical properties of the aerosol at the heights in question.

No Inversion Model

The third of the three models is the case where there is no real temperature inversion apparent in the measured and filtered profile. Such a profile was the usual form obtained in the tropical waters during the KEY-90 experiment. An example is shown in figure 9 where three soundings were taken in the same general area to the east of the Florida Keys near the town of Marathon, Florida. Two of the soundings were done with the conventional balloon-borne probe and one with an instrument aircraft making a spiral vertical ascent from near the sea surface to 3 kilometers. They all show a constant decrease in air temperature with increasing altitude except for "noise" superimposed on the record. The processing of the NOVAM subroutine would completely eliminate these "noise" blips and show a clean, no inversion profile with few significant points.

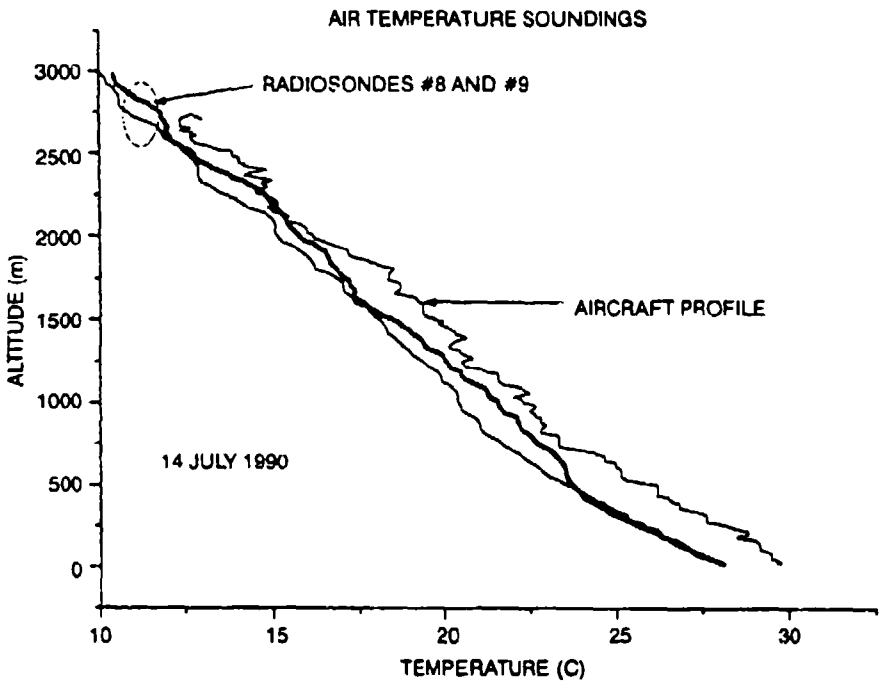


Figure 9. Temperature profiles plotted from three soundings taken from the KEY-90 data set on 14 July 1990. There are fluctuations and kinks in these curves but the main temperature decreased monotonically as altitude increased.

The no-inversion model assumes an exponential decrease in the marine aerosol produced at the sea surface. The small-sized aerosol, which we assume are from the continental areas, are well mixed in such an atmosphere; therefore, we assume that the concentration values determined at the surface are constant throughout the vertical dimension of the MBL. This refers to both the A0 and the A1 components. On the other hand, the aerosol generated at the sea surface are assumed to have an exponential drop off with altitude. The scale heights of these exponential curves will be different. The scale height of the A2 parameter has been determined empirically by Wells, Gel, and Munn [31] to be about 800 meters; whereas the scale height of the largest sized aerosol has been estimated, from various data sets the authors have looked at, to be about 50 meters.

This is expres_ed in mathematical form for the smallest size aerosol as

$$A0(z) = A0(0) \quad \textit{and} \quad A1(z) = A1(0) \ , \tag{36}$$

for the midsized aerosol as

$$A2(z) = A2(0) \cdot \exp\left(-\frac{z}{800}\right) \ , \tag{37}$$

and for the largest class of aerosols as

$$A3(z) = A3(0) \cdot \exp\left(-\frac{z}{50}\right) \ . \tag{38}$$

Of course the size distribution at any altitude will depend on these concentration vaiues as well as the growth of these particles in the lognormal function due to hygroscopic action. In addition, the concentrations at the surface, $A0(0)$, $A1(0)$, $A2(0)$, and $A3(0)$ will be calculated from the amp, average wind speed, and current wind speed values corrected by visibility measurement and IR measurements, if they are available.

This model then calculates the volume extinction coefficient and the volume absorption coefficient from the multicomponent lognormal size distribution using the method of interpolation of data between precalculated Mie calculations for the RH at that height. The subroutine then returns all the EO data and size-distribution data for the given altitude and wavelength.

An example of how well the exponential model works in a no-inversion environment is shown in figure 10. In this figure are plotted all of the 3.5 μ extinction data as a function of height obtained during the KEY-90 experiment mentioned above. Each of these individual measurements is shown as a point in the figure regardless of the circumstances of the measurement, such as what the RH was, or if clouds were nearby. Superimposed on these points are two lines. One line is the least square fit of all of the points on the curve and it is labeled the least squares fit of all aircraft data. The slope of this line shows that in the environment typical in the Florida Keys area, the exponential model works well. The other line drawn in figure 9 is the least squares line fitted to all of the NOVAM-predicted profiles obtained from all of the meteorological soundings done with both the aircraft and with radiosondes.

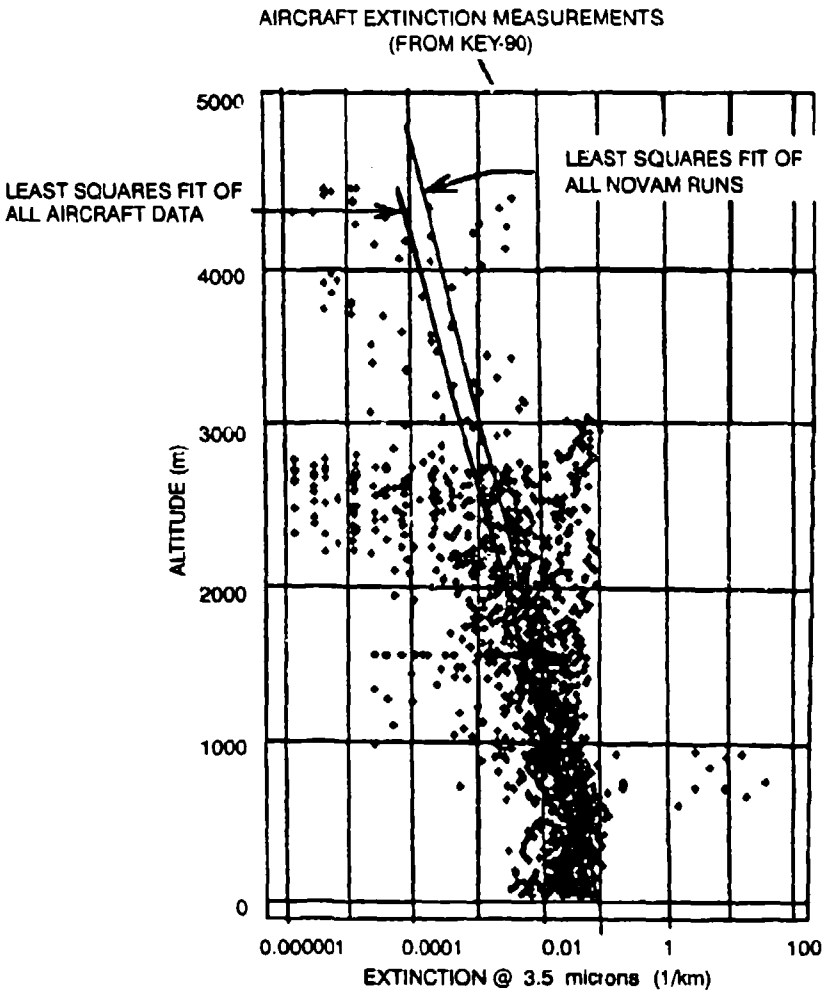


Figure 10. A combined plot of all KEY-90 3.5 μ extinction data obtained from aircraft measurements of aerosol size distribution. These points are plotted together on the same chart and exhibit a mean characteristic. Least square fits to the data and also from all of the NOVAM runs are plotted as lines in the figure. These data show that in the mean, the exponential function with altitude works quite well for the "no-inversion" types of data.

CONCLUSIONS

We have presented a model for describing the EO properties of the unique marine aerosol found in the regions from shipboard height to above several kilometers in altitude. The model has been written as a self-contained FORTRAN subroutine so it could be incorporated into larger scale models such as the LOWTRAN and MODTRAN codes. The model needs information on the meteorological sounding at the site where the calculation is made as well as information on certain meteorological parameters near the surface of the sea. The model has certain shortcomings that need to be addressed in future modifications. First of all, the region of the applicability is from shipboard level (about 5 to 10 meters) to regions above the lower troposphere where other aerosol models will be more appropriate. This leaves two areas that are not covered well by the model. At the higher altitudes, various models developed by the U. S. Air Force and included in the LOWTRAN/MODTRAN codes will be more accurate. On the other extreme, an important propagation path that grazes the sea surface or passes through the region within a meter or so of the sea surface is not adequately covered by NOVAM. This is because NOVAM is in part an empirical model and based on measurements in the real world. Current interest from shipboard level on down currently lacks observation data because of the difficulty in obtaining them. This region will be especially important to IR propagation during rough weather and high seas where many marine-generated drops and droplets are suspended in these lower levels of the atmosphere. This problem is currently being remedied by a large-scale experiment called the Marine Aerosol Properties and Imager Performance (MAPTIP) trial off the Dutch coast sponsored by NATO. The results of this experiment will contribute to the development of an advanced Navy aerosol model (ANAM) currently under development. These results can be added into the modular format of NOVAM to increase its regions of application.

Another shortcoming in NOVAM is its somewhat limited types of weather situations in which it is applicable. An earlier version of NOVAM included the region just below stratus clouds, but because this model had a limited band of wavelength validity and required inputs that are really incompatible with a self-contained model, this submodel was dropped from the current model. Advances in these areas await the development of models from the basic research community sometime in the future.

An area of concern in the application of models such as NAM and NOVAM is the use of the models in the close-in coastal areas. As these models were developed for the open ocean region far away from the land influences, error would be expected when unusual sources of aerosol are sent into the atmosphere by man-made sources. The amp concept was introduced into NAM and NOVAM to compensate for these problems, but experience has shown that this is one of the weakest parts of the models. It is the author's opinion that a special coastal aerosol model needs to be developed that will adequately take into account local sources of aerosol.

REFERENCES

[1] Gathman, S. G. Opt. Engi., 22 (1) p. 57. 1983.

[2] Gathman, S. G. "Optical Properties of the Marine Aerosol as Predicted by a BASIC Version of the Navy Aerosol Model," NRL Memorandum Report 5157. 1983.

[3] Gathman, S. G. "Navy Hygroscopic Aerosol Model," *Hygroscopic Aerosols* L. H. Ruhnke and A. Deepak, eds. (A. Deepak Publishers), Hampton, VA, pp. 93–115. 1984.

[4] Davidson, K. L., G. deLeeuw, S. G. Gathman, D. R. Jensen. 1990. "Verification of the Naval Oceanic Vertical Aerosol Model During FIRE," *FIRE Science Results 1989*. D. S. McDougal, ed., NASA Conference Report 3079, pp. 191–196.

[5] Gerber, H., S. Gathman, J. James, M. Smith, I. Consterdine, and S. Brandeki. 1990. "NRL Tethered Balloon Measurements at San Nicolas Island during FIRE IFO 1987," *Fire Science Results 1988*, edited by D. S. McDougal and H. Scott Wagner, NASA Conference pub. 3083.

[6] Gathman, S. G., G. deLeeuw, K. L. Davidson, and D. R. Jensen. "The Navy Oceanic Vertical Aerosol Model: Progress Report," *AGARD, 45th symposium Electromagnetic Wave Propagation Panel, on Atmospheric Propagation*, Copenhagen, Denmark, 9–11 Oct 1989, c.p. 454, #17. 1990.

[7] Gathman, S. G. "Ocean Aerosol Measurements and Models in the Straits of Florida (The KEY-90 Experiment)," *SPIE vol 1688, Atmospheric Propagation and Remote Sensing*, pp. 2–13. 1992.

[8] Gathman, S. G., D. R. Jensen, W. P. Hooper, J. E. James, H. E. Gerber, K. Davidson, M. H. Smith, I. E. Consterdine, G. de Leeuw, G. J. Kunz, and M. M. Moerman. "NOVAM Evaluation Utilizing Electro-Optics and Meteorological Data from KEY-90," NRaD Technical Report: TR-1608. San Diego, CA. 1993.

[9] de Leeuw G., G. J. Kunz, and M. M. Moerman. "Lidar and aerosol measurements by the TNO Physics and Electronics Laboratory during KEY90 (Marathon, FL, USA; 2–19 July 1990)," TNO Report, FEL-90-B375. 1990.

[10] de Leeuw, G. and G. J. Kunz. "NOVAM evaluation from aerosol and lidar measurements in a tropical marine environment," SPIE Vol 1688, *Atmospheric Propagation and Remote Sensing*, pp. 14–28. 1992.

[11] Davies, C. N. *Aerosol Science*, 5, 293–300. 1974.

[12] Smith, M. H. and D. R. Bates. "Radon Concentrations over the North East Atlantic," UMIST Interim report for April 1991 to March 1992, Department of Pure and Applied Physics, UMIST, Manchester, M60 1QD. 1992.

[13] Hale, G. M., and M. R. Query. *Appl. Opt.*, 12, pp. 555–563. 1973.

[14] Hänel, G. *Contrib. Atmos. Phys.*, 44, p. 137. 1971.

[15] Shettle, E. P., and R. W. Fenn. "Models for the Aerosols of the Lower Atmosphere and the Effects of Humidity Variations on their Optical Properties," AFGL-TR-79-0214 Environmental Research Papers, #676, AFGL, Hanscom AFB, MA 01731, pp. 94. 1979.

[16] Volz, F. E. *Appl. Opt.*, 11, pp. 755–759. 1973.

[17] Volz, F. E. *Appl. Opt.*, 12, pp. 564–568. 1973.

[18] Dave, J. V. "Subroutines for Computing the Parameters of Electromagnetic Radiation Scattered by a Sphere," IBM Palo Alto Scientific Center Report #320–3237. 1968.

[19] Kneizys, F. X., E. P. Shettle, W. O. Gallery, J. H. Chetwynd, Jr., L.W. Abrew, J. E. A. Selby, S. A. Clough, and R. W. Fenn. "Atmospheric Transmittance/Radiance: Computer Code LOWTRAN 6," AFGL-TR-83-0187 Environmental Research Papers, #846. AFGL, Hanscom Air Force Base, MA. 1983.

[20] McCartney, E. J. *Optics of the Atmopshere, Scattering by Molecules and Particles*, John Wiley & Sons, New York. 1976.

[21] Kochsmeider, H. Beitr. Phys. Freien Atmos., p. 439. 1924.

[22] Fitzgerald, J. W. "On the Growth of Aerosol Particles with Relative Humidity," NRL Memorandum Report 3847. 1978.

[23] *Table Curve Windows, v 1.0* made by *Jandel Scientific*.

[24] Meserve, J. M. *U.S. Navy Marine Climatic Atlas of the World, Volume 1, North Atlantic Ocean*, NAVAIR 50-1C-528, U.S. Government Printing Office, Washington DC, (Dec). 1974.

[25] Gerber, H. E. "Relative Humidity Parameterization of the Navy Aerosol Model (NAM)," NRL Report 8956. 1985.

[26] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vettering. *Numerical Recipes, the Art of Scientific Computing*, Cambridge University Press, Cambridge, p. 495. 1986.

[27] Fairall, C. W., and K. L. Davidson. "Aerosols in the Marine Atmospheric Boundary Layer" in *Oceanic Whitecaps and their Role in Air-Sea Exchange Processes*, E. C. Monahan and C. MacNiocaill (eds.) D. Reidel Publishing Co. Dortrecht, p. 195. 1986.

[28] Fairall, C. W., K. L. Davidson, and G. E. Schacher. "Application of a Mixed Layer Model to Aerosols in the Coastal Marine Boundary Layer, first International Conference on Meteorology and Air/Sea Interaction of the Coastal Zone," The Hague, Netherlands, 10–14 May 1982.

[29] Slinn, S. A., and W. G. N. Slinn. *Atmos. Environ.*, 14, 1013–1016. 1980.

[30] Davidson, K. L., and C. W. Fairall. "Optical properties of the marine atmospheric boundary layer: aerosol profiles," SPIE vol. 637—*Ocean Optics* VIII, pp. 18–24. 1986.

[31] Wells, W. C., G. Gei, and M. W. Munn. *Appl. Opt.* 18, 654–659. 1977.

APPENDIX INDEX Location of Functions and Subroutines

TYPE	NAME	PG #
SUBROUTINE	SMOOFT	71
FUNCTION	Sgn	53
SUBROUTINE	SimpleBLCase	47
SUBROUTINE	Sky2	84
FUNCTION	Smixr	56
SUBROUTINE	Sufin	98
SUBROUTINE	Swell	98
FUNCTION	Td	62
FUNCTION	Th	56
FUNCTION	Thstar	58
FUNCTION	Thz	58
FUNCTION	Vappr	53
FUNCTION	Vp	58
SUBROUTINE	We2	88
SUBROUTINE	WeakConvectionCase	43
SUBROUTINE	Wecalc	80
SUBROUTINE	Whiteflux	82
FUNCTION	Xtoy	60
FUNCTION	Zcon	59
SUBROUTINE	Zerobin	72
FUNCTION	Zzero	55

APPENDIX A A NOVAM TEST DRIVER

```
      PROGRAM DRIVER2
c (=====================================================================)
c (  File Name:  Driver2.FOR                                            )
c (                                                                     )
c (  Description: The main driver unit for all the Novam Calculation models. )
c (                                                                     )
c (  Purpose: Main driver for NOVAM subroutines calculations. NOVAM produces )
c (    an output file, OUT1 containing extinction & absorption, RH and the  )
c (    a0 -> a3 size distribution parameters at an altitude of 200 meters   )
c (                                                                     )
c (  Preconditions: SurfObsFile, RSondefile, and NOVAM.INI exists;      )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (-------------------------------------------------------------------)
c (   Date       |    Programmer     | Remarks                        )
c (    Jun 1993 | Stuart Gathman     | made into a NOVAMSR driver      )
c (-------------------------------------------------------------------)

      logical readok
      logical repeatflag
      character*10 WaveLenStr
      character*12 RSondeFileName
      character*12 SurfObsFileName
      character*1  DataType
      integer nettime, time1,H,M,S,I100TH

c     (start the timer  )
      call gettim(h,m,s,i100th)
      time1 = 3600*h+60*m+s

c     ( Run the NOVAM model )
      open(unit=31,file='out1',status='unknown') ! The output file.
      call ReadSetupFile(WaveLenStr, SurfObsFileName,
     $                   RSondeFileName, DataType, ReadOk)
      if (.NOT. ReadOK) then
        stop 'setup file novam.ini does not exist'
      else
        read(WaveLenStr,'(f10.4)') WaveLen
      endif
        alt=10.0  !first altitude for calculation
        repeatflag=(1.eq.0)  !False define  use inputdata for first point
      call NOVAMSR(alt,wavelen,SurfObsFileName,RSondeFileName,
     $             DataType,repeatflag,
     $             Ext,Abs,a0,a1,a2,a3,rh)
      write(31,30) alt, rh, ext, abs,a0,a1,a2,a3
   30 format(1h ,f7.0,1x,f5.1,1x,e10.3,1x,e10.3,4(1x,e10.3))
      alt=0.0
      do i=1,50
      alt=alt+100
      repeatflag=(1.eq.1)
      call NOVAMSR(alt,wavelen,SurfObsFileName,RSondeFileName,
     $             DataType,repeatflag,
     $             Ext,Abs,a0,a1,a2,a3,rh)
      if(ext .lt. 0.0) goto 55  !jump out of loop when data not available
      write(31,30)    alt, rh, ext, abs,a0,a1,a2,a3
      enddo

c     (Write the elapsed time for the run on the screen)
   55 call gettim(h,m,s,i100th)
      nettime = 3600*h+60*m+s - time1
      write(*,*) 'The net time of computation is:',nettime
      end  ! ( driver2 main )
```

APPENDIX B MAIN NOVAM SUBROUTINE

```
c (=====================================================================)
c (  File Name:  NOVAMsr.FOR                                            )
c (                                                                     )
c ( Purpose: Main driver subroutines for NOVAM calculations.  NOVAM produces )
c (     output file ExtincFileName which contains extinction & absorption   )
c (     vs. altitude data for later plotting.                           )
c (                                                                     )
c (  Description: The Novam Calculation models.                         )
c (                                                                     )
c (  Revision History:                                                  )
c (  10 Jan 1987 | Stu Gathman, NRL | Ver 0.02 Translated from TBASIC v1.02  )
c (  24 Mar 1989 | Stu Gathman      | Ver 0.07                          )
c (  28 Mar 1989 | Stu Gathman      | Ver 0.08 Logic tree simplified    )
c (  01 Jan 1991 | Stu Gathman      | Ver 0.10                          )
c (  22 Jan 1991 | Stu Gathman      | Ver 0.11 many improvements        )
c (  27 Sep 1991 | Charles McGrath  | Comments added                    )
c (  29 Oct 1991 | Charles McGrath  | Restructure/added procedures      )
c (     Nov 1992 | Linda Hitney     | converted from PASCAL to FORTRAN  )
c (     Jun 1993 | Stu Gathman,NRaD | convert into Fortran Subroutine   )
c (-------------------------------------------------------------------)
c (  Version | Date        |   Programmer    | Remarks                 )
c (-------------------------------------------------------------------)
c (  1.00    | 01 Jan 1991 | Stu G. Gathman  |                         )
c (  1.01    | 27 Sep 1991 | Charles McGrath |  Added comments         )
c (  1.02    | 21 Oct 1991 | Charles McGrath |  many Globals made local )
c (  1.03    | 28 Oct 1991 | Charles McGrath |  NOVAM broken into procedures)
c (  1.04    | 29 Oct 1991 | Charles McGrath |  Divided into 3 smaller units)
c (  2.15    | 03 Feb 1992 | Charles McGrath |  made an independent progarm )
c (  2.15    | 10 Feb 1992 | Charles McGrath |  mod read inputs at NOVAM.INI)
c (  3.00    |    Nov 1992 | Linda Hitney    |  converted from PASCAL to  )
c (          |             |                 |  FORTRAN                   )
c (  3.10    | 01 Jul 1993 | Stuart Gathman  |  NOVAM Subroutine in FORTRAN )
c (=====================================================================)

      SUBROUTINE NOVAMSR(alt,wavelen,SurfObsFileName,
     $            RSondeFileName,DataType,repeatflag,
     $            Ext,Abs,a0,a1,a2,a3,rh)

c  This is a subroutine which exercises the NOVAM subroutines.
c  Inputs to this subroutine are:
c                              alt, the altitude in meter
c                          wavelen, in microns
c                   SurfObsFileName, name of surface observation file
c                    RSondeFileName, name of the radiosonde data file
c                         DataType, n for NOVAM type or r for rs type
c                       repeatflag, "true" if input data used before
c                                   "false" if new set of data.
c  Outputs from this subroutine are:
c                              Ext, Extinction in 1/km at Wavelen
c                              Abs, Absorption in 1/km at Wavelen
c                                a0,a1,a2,a3,rh (at alt)
c
c     ( Initialize Variables )

      save rdataary,rsdata,rscalc,sodata
      include 'rscalc.inc'
      include 'rsdata.inc'
      include 'sodata.inc'
      real RdataAry (200,3)
      real preamble(5,3)
      logical repeatflag
      character*1    DataType
      character*12   RSondeFileName
      character*12   SurfObsFileName
c     ( Check Initial Variables )
        if(ALT .LT. 9.0) GOTO 98  !OUT OF RANGE
        IF(ALT .GT. 6000.0) GOTO 98  !OUT OF RANGE
        IF(WAVELEN .LT. 0.2) GOTO 98 !OUT OF RANGE
```

```
        IF(WAVELEN .GT. 40.0) GOTO 98 !OUT OF RANGE

      if(repeatflag) goto 11
        do 2 j=1,3
        do 1 i=1,200
        rdataary(i,j)=0.0
    1 continue
    2 continue

c Check for NOVAM type data or Radio sonde type data from NOVAM.INI

      if ( DataType .eq. 'N' .or. DataType .eq. 'n' ) then
        if(RSondeFileName .NE. "aptmr.dat") then
          open(unit=24, file=RSondeFileName, status='old')
          open(unit=23, file='aptmr.dat', status= 'unknown')
    6     read(unit=24,fmt=8,end=10) a,b,c
          write(unit=23,fmt=9)a,b,c
          goto 6
        endif
    8   format(3f10.2)
    9   format(3(f10.3,2x))
   10   continue
        close (24)
        close (23)
        call convert   ! aptmr.dat ---> convert ---> patrh.dat

      else
        !rsonde type data
        if ( DataType .ne. 'R' .and. DataType .ne. 'r' ) then
          write(*,*) '4th line of novam.ini file must indicate'
          write(*,*) 'data type in column 13'
          write(*,*) 'n or N for NOVAM  type data'
          write(*,*) 'r or R for RSONDE type data'
          GOTO 98 ' DOES NOT RECOGNIZE THE DATA TYPE
        endif
      endif

      call filter   ! patrh.dat ---> filter----> sigfile

      call preamb(SurfObsFileName,preamble) ! sigfile---> preamb -->preamble()

c     ( Read RADIOSONDE data file )

      call MAKE_RDATAARY(preamble,rdataary)
      call CheckRSondeData(RdataAry)

c     ( Read in SURFACE OBSERVATIONS file )

      call SufIn(SOData, SurfObsFileName)

c     ( Assign Surface temp, rel humidity, & mixing ratio if not measured )

      call AssignT10_RHH_QP(RDataAry, SOData)

c     ( Assign Rsonde data to GLOBALS globals )

      call AssignRSondeVars(RDataAry, RSData, RSCalc, SOData)

   11 continue
      rnum=rdataary(1,1)
      lastline=int(rnum)
      if (rdataary(lastline,1) .lt. alt) goto 98

c     ( Run the NOVAM model )

          if(rdataary(2,1) .lt. 0.0) goto 77
          if(rdataary(4,1) .gt. 0.0) goto 888
          if(rdataary(2,1) .gt. 0.0) goto 999
          write(*,*) 'miss'
          goto 98
```

```
 77   if( .NOT. repeatflag) write(*,*)'Strong Convection Case'
      call NoInversionsCase(WaveLen, Alt, Ext, Abs, RH,a0,a1,a2,
     $                      a3,rdataary,sodata)
      goto 99

888   if( .NOT. repeatflag) write(*,*)'Weak Convection Layer Case'
      call WeakConvectionCase(rsdata,WaveLen,Alt,Ext,Abs,
     $                        RH,a0,a1,a2,a3,rdataary,sodata)
      goto 99


999   if( .NOT. repeatflag) write(*,*)'Simple Boundary Layer Case'
      call SimpleBLCase(rscalc,WaveLen, Alt, Ext, Abs,
     $                        RH,a0,a1,a2,a3,rdataary,sodata)
      goto 99

 98   ext=-999.9  !no data  output
      abs=-999.9
      a0=-999.9
      a1=-999.9
      a2=-999.9
      a3=-999.9
      rh=-999.9

 99 end  ! ( NovamSR subroutine  )
```

APPENDIX C NO INVERSION, CONVECTION MODEL

```
      SUBROUTINE NoInversionsCase(
     $ WaveLen, alt, ext, abs, rh,a0,a1,a2,a3, RDataAry, SOData )
c (=====================================================================)
c (  Purpose:  To provide extinction and absorption at altitude, alt and  )
c (            wavelength, wavelen,  for the case of no inversion.       )
c (                                                                      )
c (  Called by:  Novamsr                                                 )
c (  Calls out:  Optics, fna, fnb, fnc, palt, xtoy, smixr                )
c (                                                                      )
c (  Preconditions:                                                      )
c (  Global Variables:                                                   )
c (  Revision History:                                                   )
c (----------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                         )
c (----------------------------------------------------------------------)
c (  29 Oct 1991  | Charles McGrath   | Created Procedure from UNOVAM.NOVAM )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN    )
c (  June 1993    | Stuart Gathman    | Converted into a one input,         )
c (                                   | one output subroutine               )
c (=====================================================================)

      real ko, ks
      real smr, mr
      real RDataAry(200,3)

      include 'sodata.inc'
      integer exactflag, altflag

      exactflag=0
      H2  = 800.0   !(Scale height given by Wells,  Gel & Munn )
      H3  = 50.0    !(Scale height estimated by Gathman )
      Z10 = 10.0
      if (SOData.SVis .GT. 0.0) then
          ! (Use input visibility to correct data)
            if (SOData.AMP .GT. 5.0) then

               call  Optics(0.3*fna(SOData.AMP),
     $                      0.7*fna(SOData.AMP),
     $                      fnb(SOData.UAve),
     $                      fnc(SOData.U10),
     $                      SOData.RHH, 0.55,
     $                      ext, absorb,
     $                      SOData.AMP)
            else

               call  Optics(0.0, fna(SOData.AMP),
     $                      fnb(SOData.UAve), fnc(SOData.U10),
     $                      SOData.RHH, 0.55, ext, absorb,
     $                      SOData.AMP)
            endif

            extvis = 3.9/SOData.SVis
            ko = extvis/ext
      else

        ko = 1.0
      endif

c  At this point ko is the correction term based on svis
c  Calculate ks for the correction term for the sea salt terms from the
c  IR measurements at 10.6 microns
c  Use input ir extinction @ 10.6 MICRONS to correct data.

      if (SOData.IRExt .GT. 0.0) then

        if (SOData.AMP .GT. 5.0) then
          call  Optics( 0.3*fna(SOData.AMP),
```

```
$                  0.7*fna(SOData.AMP),
$     fnb(SOData.UAve), fnc(SOData.U10),
$               SOData.RHH, 10.6, ext, absorb,
$               SOData.AMP)
      else
        call  Optics(0.0, fna(SOData.AMP),
$                  fnb(SOData.UAve), fnc(SOData.U10),
$                  SOData.RHH, 10.6, ext, absorb,
$                  SOData.AMP)
      endif
      ks = SOData.IRExt/ext  ! ks is the correction factor for 10.6 micron wavelength
     else

      ks = ko
     endif

c (At this point ks is the correction term based on irext)


     if (SOData.AMP .GT. 5.0) then
        call  Optics( 0.3*ko*fna(SOData.AMP),
$                  0.7*ko*fna(SOData.AMP),
$                  ((ko+ks)/2.0)*fnb(SOData.UAve),
$                  ks*fnc(SOData.U10), SOData.RHH, WaveLen,
$                  ext, absorb, SOData.AMP)
     else
        call  Optics(0.0, ko*fna(SOData.AMP),
$                 ((ko+ks)/2.0)*fnb(SOData.UAve),
$                 ks*fnc(SOData.U10), SOData.RHH, WaveLen,
$                 ext, absorb, SOData.AMP)
     endif

     temprhh=SOData.RHH


       a=RdataAry(1,1)
       npts = int(a)
       ZHgt = RdataAry(npts,1)

c set up for 4 lognormal system for altitude ALT.

       if ((SOData.AMP .GT. 5.0) .AND. (alt .LT. 1000.0)) then
         v0 = 0.3*fna(SOData.AMP)
         v1 = 0.7*fna(SOData.AMP)
       else
         v0 = 0
         v1 = fna(SOData.AMP)
       endif

       ex = alt/h2

       if (ex .GT.80.0) then  !(This is a numerical trick to keep functions)
         V2 = 0.0             !(working at small numbers ie less than 1/10^36.)
       else
         V2 = fnb(SOData.UAve)*EXP(-alt/H2)
       endif

       ex = alt/h3

       if (ex .GT.80.0) then
         V3 = 0.0
       else
         V3 = fnc(SOData.U10)*EXP(-alt/H3)
       endif

c  At this point, We need to find where the input altitude is with respect to the
c  various altitudes on the radiosonde data array.  We then will find the closest
c  altitude that the radiosonde uses just above the input altitude and just below
c  the input altitude.
```

```
c Technique for bracketing the input altitude with the two nearest
c sigfile altitudes.

          altflag=-1
          exactflag=-1
        do 10 i=6,npts

         if (alt .lt. RdataAry(i,1)) goto 11
         if (alt .eq. RdataAry(i,1)) goto 12
   10    continue
   11    altflag=i
         goto 13
   12    exactflag=i
   13    continue

         if (exactflag .gt. 0)  goto 20   !no interpolation necessary in this case
           if(altflag .eq. 6) goto 15

c  Here altflag is the index just below the altitude and altflag + 1 is just above
c  the desired altitude.  We will do linear interpolation to find the require
c  relative humidity from the knowledge of the RdataAry and Altflag.

        Theta=(RdataAry(altflag,2)-RdataAry(altflag-1,2))
        theta=theta*(alt-RdataAry(altflag-1,1))
        theta=theta/(RdataAry(altflag,1) - RdataAry(altflag-1,1))
        theta=theta + RdataAry(altflag-1,2)

        mr=(RdataAry(altflag,3)-RdataAry(altflag-1,3))
        mr=mr*(alt-RdataAry(altflag-1,1))
        mr=mr/(RdataAry(altflag,1) - RdataAry(altflag-1,1))
        mr=mr+(RdataAry(altflag-1,3))
        goto 30

   15   rh_at_z=SODATA.RHH
          goto 40

   20    Theta=rdataary(exactflag,2)
         mr=RdataAry(exactflag,3)

   30    continue
        ! (-----------------------------------------------------------
           ! at this point we need to calculate the relative humidity from
           ! the potential temperature, RdataAry(i,2) and the mixing ratio
           ! RdataAry(i,3).  This is accomplished by finding the saturation
           ! mixing ratio at the potential temperature and then the ratio
           ! of 100 * r/rs is the relative humidity.  Note that I have a
           ! routine which gives rs = 0.622 * vappr(T) /(p(z) -vappr(T))
           ! if I know T in degrees C.   from List p308, we see that
           ! th = t *(1000/p)^ (2/7)
        ! -----------------------------------------------------------------)

          testp = palt(alt)

c         Here tatz is the temperature at altitude alt

          tatz  = (Theta+273.15) * xtoy(1000.0/testp, -0.286)-273.15
          smr   = 1000.0* smixr(alt, tatz)
          rh_at_z = 100.0*mr/smr
          rh_at_z = MIN(rh_at_z, 99.9)
   40     call  OPTICS(ko*v0, ko*v1, ((ko+ks)/2.0)*v2, ks*v3, rh_at_z,
         $                   WaveLen, ext, abs, SOData.AMP)

          rh=rh_at_z
          a0=v0
          a1=v1
          a2=v2
          a3=v3
         END  ! (  NoInversionsCase )
```

APPENDIX D THE WEAK CONVECTION MODEL

```
      subroutine WeakConvectionCase(rsdata,
     $ WaveLen, alt, ext, abs, rh,a0,a1,a2,a3, RDataAry, SOData )
c (=====================================================================)
c (  Purpose:  To provide extinction and absorption at altitude, alt and  )
c (           wavelength, wavelen,  for the case of a weak convection     )
c (  Called by:  Novamsr                                                  )
c (  Calls out:  Novamw                                                   )
c (  Preconditions:                                                       )
c (  Global Variables:                                                    )
c (---------------------------------------------------------------------)
c (  29 Oct 1991 | Charles McGrath  |  Created Procedure from UNOVAM.NOVAM  )
c (     Nov 1992 | Linda Hitney     |  Converted from PASCAL to FORTRAN     )
c (     Jun 1993 | Stuart Gathman   |  Prepared for NOVAMSR operation       )
c (=====================================================================)


      real RDataAry(200,3)
      include 'rsdata.inc'
      include 'sodata.inc'
      integer ALTFLAG,EXACTFLAG
      real abs, ko, ks



c (At this point we calculate what the effect of the measured svis would be
c  on the the profile.  this part of the code added 1/22/91 by sgg )
c (Find what the estimated extinction would be a 0.55 microns)

      if (SOData.SVis .GT. 0.0) then
        ! (Use input visibility to correct data)
        if (SOData.AMP .GT. 5.0) then
          call  Optics(0.3*fna(SOData.AMP), 0.7*fna(SOData.AMP),
     $                 fnb(SOData.UAve), fnc(SOData.U10),
     $                 SOData.RHH, 0.55, ext, absor, SOData.AMP)
        else
          call  Optics(0.0, fna(SOData.AMP), fnb(SOData.UAve),
     $                 fnc(SOData.U10),
     $                 SOData.RHH, 0.55, ext, absor, SOData.AMP)
        endif

        extvis = 3.9/SOData.SVis
        ko = extvis/ext     ! (ko is the correction factor for vis wavelength)
      else
        ko = 1.0
      endif
c (At this point ko is the correction term based on svis)
c (Calculate ks for the correction term for the sea salt terms from the
c   IR measurements at 10.6 microns)
        if (SOData.IRExt .GT. 0.0) then
          ! (Use input ir extinction @ 10.6 MICRONS to correct data)
          if (SOData.AMP .GT. 5.0) then
            call  Optics(0.3*fna(SOData.AMP), 0.7*fna(SOData.AMP),
     $                 fnb(SOData.UAve), fnc(SOData.U10), SOData.RHH,
     $                 10.6, ext, absor, SOData.AMP)
          else
            call  Optics(0.0, fna(SOData.AMP), fnb(SOData.UAve),
     $                 fnc(SOData.U10),
     $                 SOData.RHH, 10.6, ext, absor, SOData.AMP)
          endif

          ! (ks is the correction factor for 10.6 micron wavelength)
          ks = SOData.IRExt/ext
        else
          ks = ko     ! (the default case sets ks=ko)
        endif

c (At this point ks is the correction term based on irext)
```

```
c Technique for bracketing the input altitude with the two nearest
c sigfile altitudes.

          altflag=-1
          exactflag=-1
        do 10 i=6,RDATAARY(1,1)

         if (alt .lt. RdataAry(i,1)) goto 11
         if (alt .eq. RdataAry(i,1)) goto 12
   10    continue
   11    altflag=i
         goto 13
   12    exactflag=i
   13    continue
        IF(exactflag .gt. 6)  goto 100 !no ir   polation necessary in this case
        IF(altflag .gt. 6)  goto 50 !interpolate mmr and theta.
c     SURFACE CALCULATION

          RH_AT_Z=SODATA.RHH


          if (SOData.AMP .GT. 5.0) then
            A0 = 0.3*ko*fna(SOData.AMP)
               A1 = 0.7*ko*fna(SOData.AMP)
               A2 = ((ko+ks)/2)*fnb(SOData.UAve)
                 A3 = ks*fnc(SOData.U10)
            call  Optics(A0,A1,A2,A3,
     $                   RH_AT_Z, WAVELEN,
     $                   ext, abs, SOData.AMP)
          else
                    A0 = 0.0
               A1 = ko*fna(SOData.AMP)
               A2 = ((ko+ks)/2)*fnb(SOData.UAve)
                 A3 = ks*fnc(SOData.U10)
            call  Optics(A0,A1,A2,A3,
     $                   RH_AT_Z, WAVELEN,
     $                   ext, abs, SOData.AMP)

          endif

          goto 500


c  Here altflag is the index just below the altitude and altflag + 1 is just above
c  the desired altitude.  We will do linear interpolation to find the require
c  relative humidity from the knowledge of the RdataAry and Altflag.


    50    Theta=(RdataAry(altflag,2)-RdataAry(altflag-1,2))
          theta=theta*(alt-RdataAry(altflag-1,1))
          theta=theta/(RdataAry(altflag,1) - RdataAry(altflag
     $                                              -1,1))
          theta=theta + RdataAry(altflag-1,2)
          mr=(RdataAry(altflag,3)-RdataAry(altflag-1,3))
          mr=mr*(alt-RdataAry(altflag-1,1))
          mr=mr/(RdataAry(altflag,1) - RdataAry(altflag-1,1))
          mr=mr+(RdataAry(altflag-1,3))
          goto 200

        ! (-----------------------------------------------------------------
          ! at this point we need to calculate the relative humidity from
          ! the potential temperature, RdataAry(i,2) and the mixing ratio
          ! RdataAry(i,3).  This is accomplished by finding the saturation
          ! mixing ratio at the potential temperature and then the ratio
          ! of 100 * r/rs is the relative humidity.  Note that I have a
          ! routine which gives rs = 0.622 * vappr(T) /(p(z) -vappr(T))
          ! if I know T in degrees C.   from List p308, we see that
          ! th = t *(1000/p)^ (2/7)
        ! -----------------------------------------------------------------)
   100    Theta=rdataary(exactflag,2)
```

```
        mr=RdataAry(exactflag,3)
  200   testp = palt(alt)

c       Here tatz is the temperature at altitude alt

        tatz  = (Theta+273.15) * xtoy(1000.0/testp, -0.286)-273.15
        smr   = 1000.0* smixr(alt, tatz)
        rh_at_z = 100.0*mr/smr
        rh_at_z = MIN(rh_at_z, 99.9)

        if (RH_AT_Z .GT. 99.0) THEN
          Ext   =  999.0
          Absor =  999.0  ! (! extinction and absorbtion cannot be calculated)
        else
         ! (Find A's)
         call novamw(RSData,Alt,ko,          0, A0, SOData)

         call novamw(RSData,Alt,ko,          1, A1, SOData)

         call novamw(RSData,Alt,((ko+ks)/2), 2, A2, SOData)

         call novamw(RSData,Alt,ks,          3, A3, SOData)

        endif
       ! (calculate optics)
        call  Optics(A0, A1, A2, A3, SOData.RHH, wavelen, Ext,
     $              Abs, SOData.AMP)

  500   RH=RH_AT_Z


       end  !  subroutine WeakConvectionCase
```

APPENDIX E THE SIMPLE BOUNDARY LAYER CASE (SINGLE INVERSION)

```
      SUBROUTINE SimpleBLCase(rscalc,WaveLen, Alt, Ext, Abs,
     $                        RH,a0,a1,a2,a3,rdataary,sodata)

c (=====================================================================)
c (  Purpose:  Mixed Boundary Layer Case (formerly called NOVAMC)       )
c (           Uses  Davidson-Fairall  model                            )
c (  Called by:  NOVAMSR                                               )
c (  Calls out:  Bulk, InitPDataVector,                                )
c (              Wecalc, Deposel, Whiteflux,                           )
c (              Optics, fna,fnb, fnc, Rh,                             )
c (              RealToStr                                             )
c (  Preconditions:                                                    )
c (  Global Variables: (many from GLOBALS)                             )
c (  Revision History:                                                 )
c (---------------------------------------------------------------------)
c (   Date       |    Programmer    |  Remarks                         )
c (---------------------------------------------------------------------)
c (  29 Oct 1991 | Charles McGrath   |  Created Procedure from UNOVAM.NOVAM  )
c (     Nov 1992 | Linda Hitney      |  Converted from PASCAL to FORTRAN     )
c (     Jun 1993 | Stuart Gathman    |  Prepared for NOVAMSR operation       )
c (=====================================================================)

      include 'rscalc.inc'
      include 'sodata.inc'

      real PData(12)
      integer exactflag,altflag
      real RDataAry(200,3)
      real R0(3) /0.03, 0.24, 2.0/
      real GAMA(3)/0.0, 0.0, 0.0/
      real extvis, ko, ks
      real Lcp


      common/globals/dte,dqw,wthe,wq,ustar,tstar,qstar,tstr,tsky

c begin subroutine SimpleBLCase

      Z10             = 10.0   ! from GLOBALS ( altitude of 10 m )
      Lcp             = 2460.0 ! from GLOBALS


c (---------------------------------------------------------------------)
c (  We need to define dte, dqw, gamap,  gamar & zinv at this time!     )
c (  In order to do this calculation we must assume that all of        )
c (  this data is available before the subroutine is called.           )
c (---------------------------------------------------------------------)

c (------- Assign Radiosonde Matrix values to more descript structure ----)
c ( Rst10 is surface potential temp from radiosonde                     )
c ( tRsqp  is surface mixing ratio from  radiosonde                     )
c ( Zbase is height of base of cloud layer                              )
c ( Tbm    is potential temperature at cloud base      -                )
c ( Qmb    is mixing ratio at cloud base               -                )
c ( Tunits should be 1 if potential temperature C.                      )
c ( Tbp    is potential temperature at cloud base      +                )
c ( Qbp    is mixing ratio at cloud base               +                )
c ( Zi     is height of the cloud top                                   )
c ( Thim   is potential temperature at cloud top       -                )
c ( Qim    is mixing ratio at cloud top                -                )
c ( Qunits should be 1 if mixing ratio in g/kg?????  ???                )
c ( Tip    is potential temperature at cloud top       +                )
c ( Qip    is mixing ratio at cloud top                +                )
c (---------------------------------------------------------------------)

      rRst10  = RdataAry(1,2)
      rRsqp   = RdataAry(1,3)
```

```
      rZbase  = RdataAry(2,1)
      rTbm    = RdataAry(2,2)
      rQbm    = RdataAry(2,3)
      rTunits = RdataAry(3,1)
      rTbp    = RdataAry(3,2)
      rQbp    = RdataAry(3,3)
      rZi     = RdataAry(4,1)
      rThim   = RdataAry(4,2)
      rQim    = RdataAry(4,3)
      rQunits = RdataAry(5,1)
      rTip    = RdataAry(5,2)
      rQip    = RdataAry(5,3)

c (--- Initialize variables ---)
      Dte   = rTbp-rTbm
      Dqw   = rQbp-rQbm
      Gamap = (rTbm - rRst10)/rZbase
      Gamar = (rQbm - rRsqp)/rzbase

c (--- Find Longwave Radiation Fluxes to calculate sky temperature, Tsky ---)

      call sky2(alt, RdataAry, SOData.T10, SOData.Qp, tsky)


c (--- Calculate Ustar, Tstar, VarQstar, and Tstr ---)

      call Bulk(SOData.U10, RSCalc.thab, RSCalc.Tdelta, RSCalc.Qdelta,
     $          Ustar, Tstar, Qstar, Tstr)

c (--- Calculate Wq, Wt, Wtv, Wte, Wthe ---)
      Wt   =  -Tstar * Ustar
      Wtv  =  -Tstr  * Ustar
      Wq   =  -Qstar * Ustar / 1000.0
      Wte  =   Wt + RSCALC.Fthet * Lcp * Wq
      Wthe =  -Ustar * Tstar + 2460.0 * (-Ustar * Qstar) / 1000.0

c (--- Calculate We and Wstr ---)

      call Wecalc(PData, RSCalc, SOData)



      tempwe=RSCalc.We

c (--- Calculate ??? ---)
      Rrhum = 98.0   ! (! rh used in depovel routine)

      do I = 1 , 3
        dsize = R0(I)/5.0E+05
        call Depovel(i, dsize, Rrhum, RSCalc.thab-273.15,
     $                    Ustar, Vd, Vgdry, SOData)

        ! (! Note that R0 is radius in microns, while the 3rd depovel )
        ! (  size parameter, A0 is diameter in meters.                 )

        call Whiteflux(Sr, SOData.U10, R0(I))

        if (I .EQ. 1) then
          Xsr = FNA(SOData.AMP)
          Xp  = Xsr
        endif

        if (I .EQ. 2) then
          Xsr = FNB(SOData.UAve)
          Xp  = 0.0
        endif

        if (I .EQ. 3) then
          Xsr = FNC(SOData.U10)
          Xp  = 0.0
        endif
```

```
        tempwe=RSCalc.We

         ! (! make initial guess at tttt)
        Tttt = 1.5*(Sr-Vd*Xsr+2.5*RSCAlc.We*(Xsr-Xp))
        temp=rscalc.zinv
        temp=rscalc.wstr
        GAMA(I) = Tttt/(RSCalc.Zinv*RSCalc.Wstr)
        Tttt = 1.5*(Sr-Vd*Xsr+2.5*RSCalc.We*
     $           (RSCalc.Zinv*GAMA(I)+Xsr-Xp))

         ! (!  cac. gama after first interation in tttt)
        GAMA(I) = Tttt/(RSCalc.ZInv*RSCalc.WStr)
        GAMA(i) = -1.0*GAMA(i)

        enddo  ! (for loop)


c ( At this point we calculate what the effect of the measured svis would )
c ( be on the the profile.  this part of the code added 1/22/91 by sgg    )
c ( Find what the estimated extinction would be a 0.55 microns            )

      if (SOData.SVis .GT. 0.0) then
         ! (Use input visibility to correct data)
         if (SOData.AMP .GT. 5.0) then
           call  Optics(0.3*fna(SOData.AMP),
     $                  0.7*fna(SOData.AMP),
     $                  fnb(SOData.UAve), fnc(SOData.U10),
     $                  SOData.RHH, 0.55, ext, absorb, SOData.AMP)
         else
           call  Optics(0.0, fna(SOData.AMP), fnb(SOData.UAve),
     $                  fnc(SOData.U10), SOData.RHH, 0.55,
     $                  ext, absorb, SOData.AMP)
         endif

         extvis = 3.9/SOData.SVis
         ko = extvis/ext
      else
        ko = 1.0
      endif

c ( At this point ko is the correction term based on svis )


c (--- Calculate ks for the correction term for the sea salt terms ---)
c (--- from the IR measurements at 10.6 microns                    ---)
      if (SOData.IRExt .GT. 0.0) then
         ! (Use input ir extinction @ 10.6 MICRONS to correct data)
         if (SOData.AMP .GT. 5.0) then
           call  Optics(0.3*fna(SOData.AMP),
     $                  0.7*fna(SOData.AMP),
     $                fnb(SOData.UAve), fnc(SOData.U10),
     $                SOData.RHH, 10.6, ext, absorb, SOData.AMP)
         else
           call  Optics(0.0, fna(SOData.AMP), fnb(SOData.UAve),
     $                  fnc(SOData.U10), SOData.RHH, 10.6,
     $                  ext, absorb, SOData.AMP)
         endif

         ks = SOData.IRExt/ext  ! (ks is the correction factor for
                                !           10.6 micron wavelength)
      else
        ks = ko
      endif

c ( At this point ks is the correction term based on irext )

      if (SOData.AMP .GT. 5.0) then
         call  Optics(0.3*ko*fna(SOData.AMP),
     $               0.7*ko*fna(SOData.AMP),
     $               ((ko+ks)/2)*fnb(SOData.UAve),
```

```
     $               ks*fnc(SOData.U10), SOData.RHH,
     $               WaveLen, ext, absorb, SOData.AMP)
      else
         call  Optics(0.0, ko*fna(SOData.AMP),
     $               ((ko+ks)/2)*fnb(SOData.UAve),
     $               ks*fnc(SOData.U10), SOData.RHH, WaveLen,
     $               ext, absorb, SOData.AMP)
      endif

      temprhh=SOData.RHH

c Technique for bracketing the input altitude with the two nearest
c sigfile altitudes.

         altflag=-1
         exactflag=-1
       do 10 i=6,RDATAARY(1,1)

        if (alt .lt. RdataAry(i,1)) goto 11
        if (alt .eq. RdataAry(i,1)) goto 12
   10   continue
   11   altflag=i
        goto 13
   12   exactflag=i
   13   continue
       IF(exactflag .gt. 6)  goto 100 !no interpolation necessary in this case
       IF(altflag .gt. 6)  goto 50 !interpolate mmr and theta.
c    SURFACE CALCULATION

         RH_AT_Z=SODATA.RHH

         if (SOData.AMP .GT. 5.0) then
           A0 = 0.3*ko*fna(SOData.AMP)
              A1 = 0.7*ko*fna(SOData.AMP)
              A2 = ((ko+ks)/2)*fnb(SOData.UAve)
                A3 = ks*fnc(SOData.U10)
           call  Optics(A0,A1,A2,A3,
     $                 RH_AT_Z, WAVELEN,
     $                 ext, abs, SOData.AMP)
         else
                   A0 = 0.0
              A1 = ko*fna(SOData.AMP)
              A2 = ((ko+ks)/2)*fnb(SOData.UAve)
                A3 = ks*fnc(SOData.U10)
           call  Optics(A0,A1,A2,A3,
     $                 RH_AT_Z, WAVELEN,
     $                 ext, abs, SOData.AMP)

         endif


         goto 5000


c  Here altflag is the index just below the altitude and altflag + 1 is just above
c  the desired altitude.  We will do linear interpolation to find the require
c  relative humidity from the knowledge of the RdataAry and Altflag.


   50    Theta=(RdataAry(altflag,2)-RdataAry(altflag-1,2))
         theta=theta*(alt-RdataAry(altflag-1,1))
         theta=theta/(RdataAry(altflag,1) - RdataAry(altflag
     $                                          -1,1))
         theta=theta + RdataAry(altflag-1,2)
         mr=(RdataAry(altflag,3)-RdataAry(altflag-1,3))
         mr=mr*(alt-RdataAry(altflag-1,1))
         mr=mr/(RdataAry(altflag,1) - RdataAry(altflag-1,1))
         mr=mr+(RdataAry(altflag-1,3))
         goto 200
```

```
! (-------------------------------------------------------------
        ! at this point we need to calculate the relative humidity from
        ! the potential temperature, RdataAry(i,2) and the mixing ratio
        ! RdataAry(i,3). This is accomplished by finding the saturation
        ! mixing ratio at the potential temperature and then the ratio
        ! of 100 * r/rs is the relative humidity. Note that I have a
        ! routine which gives rs = 0.622 * vappr(T) /(p(z) -vappr(T))
        ! if I know T in degrees C.   from List p308, we see that
        ! th = t *(1000/p)^ (2/7)
     ! ---------------------- ---------------------------------------)
 100    Theta=rdataary(exactflag,2)
        mr=RdataAry(exactflag,3)
 200    testp = palt(alt)

c       Here tatz is the temperature at altitude alt

        tatz  = (Theta+273.15) * xtoy(1000.0/testp, -0.286)-273.15
        smr   = 1000.0* smixr(alt, tatz)
        rh_at_z = 100.0*mr/smr

        if (RH_AT_Z .le. 99.0) goto 2000
          Ext   =  999.0
          Absor =  999.0  ! (! extinction and absorbtion cannot be calculated)
          goto 5000

2000    Alt_of_obs =  alt

      ! (????! CALL Parcel2(Pdata, Rh, D1, D2, D3, D4, D5, D6, D7, D8))
      ! (find A0  & A1)
        if (Alt_of_obs .LE. RSCalc.ZInv) then
          if (SOData.AMP .GT. 5.0) then
            A1 = Alt_of_obs*GAMA(1)+fna(SOData.AMP)
            A0 = 0.3 * A1
            A1 = 0.7 * A1
          else
            A0 = 0.0
            A1 = Alt_of_obs*GAMA(1)+fna(SOData.AMP)
          endif
        else
          A0 = 0.0
          A1 = RSCalc.ZInv*GAMA(1)+fna(SOData.AMP)
        endif

        ! (find A2)
        if ((Alt_of_obs.LE.RSCalc.ZInv) .AND.
     $      (Alt_of_obs*GAMA(2)+fnb(SOData.UAve).GT.0.0)) then

          A2 = Alt_of_obs*GAMA(2)+fnb(SOData.UAve)
        else
          A2 = 0.0
        endif

        ! (find A3)
        if ((Alt_of_obs .LE. RSCalc.ZInv) .AND.
     $      (Alt_of_obs*GAMA(3)+fnc(SOData.U10) .GT. 0.0)) then

          A3 = Alt_of_obs*GAMA(3)+fnc(SOData.U10)
        else
          A3 = 0.0
        endif

        call  Optics(ko*A0, ko*A1, ((ko+ks)/2)*A2, ks*A3, rh_at_z,
     $              WaveLen, ext, Abs, SOData.AMP)


 5000    rh=rh_at_z
         a0=ko*a0
         a1=ko*a1
         a2=((ko+ks)/2)*a2
         a3=ks*a2
      end  ! ( procedure SimpleBLCase )
```

APPENDIX F FUNCTIONS USED IN NOVAMSR

```
      FUNCTION sgn(x)
      if (x .eq. 0.0) then
        sgn=0.0
      else
        sgn=sign(1.0,x)
      endif
      return
      end


      FUNCTION Potential_temperature(p,at)
      Potential_temperature=(at+273.15)*xtoy(1000.0/p,0.286) -273.15
      end  ! (end Potential_temperature)


      FUNCTION rMixing_ratio(h,at,p)
      rMixing_ratio=h*620.0*vappr(at)/(100.0*(p-vappr(at)))
      end  ! (end rMixing_ratio)

      FUNCTION Power10(X)! real) : real;
c (=====================================================================)
c (  Purpose: Calculate the value of 10 raised to the X power.         )
c (                                                                    )
c (                                                                    )
c (  Preconditions:                                                    )
c (  Global Variables:                                                 )
c (  Revision History:                                                 )
c (--------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                       )
c (--------------------------------------------------------------------)
c (  02 Oct 1991 | Charles McGrath    | Comments added                )
c (     Nov 1992 | Linda Hitney       |  Converted from PASCAL to FORTRAN )
c (=====================================================================)
c begin
      Power10 = exp(X*2.3025850930)
      end  ! (funciton Power10 )

      FUNCTION Vappr(T)
c (=====================================================================)
c (  Purpose: To calculate the saturation vappor pressure over water at the )
c (           temperatue T.                                            )
c (                                                                    )
c (  Preconditions:                                                    )
c (  Global Variables:                                                 )
c (  Revision History:                                                 )
c (--------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                       )
c (--------------------------------------------------------------------)
c (  9/29/87      | Stuart Gathman    | Function designed
c (  02 Oct 1991 | Charles McGrath    | Comments added                )
c (     Nov 1992 | Linda Hitney       |  Converted from PASCAL to FORTRAN )
c (=====================================================================)
c (*********************************************************************
 ! This is an approximation to the Goff-Gratch Temperature,  Saturated
 ! Vapor pressure over liquid water formula.  It was described by
 ! Richards (1971) and Wigley(1974).  In this subroutine,  T is in
 ! degree C,  and vappr(zzz) is expressed in mbs.
c *********************************************************************)

      R1 =  13.3185
      R2 =  1.976
      R3 =  0.6445
      R4 =  0.1299
      R0 =  1013.25

      If ((T .GT. 50.0) .OR. (T .LT. -50.0)) write(*,9000) T
```

```
9000  format('temperature input to Vappr function is out of range: ',
     $          f12.6)
      T00  =  1.0-373.0/(T+273.0)
      Vappr=R0*EXP(R1*T00-R2*t00*t00-R3*t00*t00*t00-R4*t00*t00*t00*t00)
      end  ! (function Vappr)


      FUNCTION PaLt(Z)
c (=====================================================================)
c (  Purpose: To calculate the pressure at an altitude Z assuming a     )
c (           standard atmosphere.                                      )
c (                                                                     )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (---------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                        )
c (---------------------------------------------------------------------)
c (  9/29/87      | Stuart Gathman    | Function coded                  )
c (  02 Oct 1991  | Charles McGrath   | Comments added                  )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=====================================================================)
c  Converts z in meters to pressure (mb).  This
c  is a rough fit to the NACA standard atmopshere
c  data found in table 63,  p267 of List.)


      if (Z.GT.500.0) then
        paLt = 1021.38 * exp(-1.2739E-4 * Z)
      else
        paLt = 1013.0 - 55.0 *z / 500.0
      endif
      end  ! PaLt


      FUNCTION Altitude(P)          ! (OK 9/29/97)
c (=====================================================================)
c (  Purpose: To calculate the altitude in a standard atmosphere which  )
c (           would have a pressure of P.                               )
c (                                                                     )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (---------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                        )
c (---------------------------------------------------------------------)
c (  9/29/87      | Stuart Gathman    | Function coded                  )
c (  02 Oct 1991  | Charles McGrath   | Comments added                  )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=====================================================================)
  (! T   . function computes altutude in meters from pressure in mbs.
 ! This formula is a fit to the NACA standard lower atmosphere data
 ! (Smithsonian Met. Tables #63,  List (1968)))

      ! begin  (altitude)
      altitude  =  0.0
      if (P.GT.1013.0) return
      if (P.GT.958.0) then
         altitude =  9.09*(1013.0-P)
      else
         altitude  =  7850.0*log(1021.38/P)
      endif
      end  !  altitude
```

```
      FUNCTION C10( lws, liws, lq)
c (=====================================================================)
c (  Purpose: To calculate the drag coefficient give a current wind speed.  )
c (                                                                         )
c (  Used in: functions Zzero, Fqstar, Fmint, and Thstar                    )
c (  Preconditions:                                                         )
c (  Global Variables:                                                      )
c (  Revision History:                                                      )
c (---------------------------------------------------------------------)
c (  Date        |    Programmer     |  Remarks                           )
c (---------------------------------------------------------------------)
c (  Sept 1987   | Stuart Gathman    | Function coded                     )
c (  02 Oct 1991 | Charles McGrath   | Comments added                     )
c (     Nov 1992 | Linda Hitney      | Converted from PASCAL to FORTRAN   )
c (=====================================================================)

c (*********************************************************************
c (! This subroutine calculates the drag coefficient.  if wind speed at
 ! 10 meters is measure,
 ! Deacons suggested form is used :  Roll (1965), p161.
 ! if no wind is measured,  a constant value is used.
c  *********************************************************************)


      if (liws.NE.-1) then
         C10 =  0.0011+4.0E-05*lws
      else
        lq  =  lq-1
        C10  =  0.002
      endif
      end  ! (function C10)


      FUNCTION Zzero(liws, lws, lq)
c (=====================================================================)
c (  Purpose: To calculate the dynamic roughness of the sea surface.        )
c (                                                                         )
c (                                                                         )
c (  Preconditions:                                                         )
c (  Global Variables:                                                      )
c (  Revision History:                                                      )
c (---------------------------------------------------------------------)
c (  Date        |    Programmer     |  Remarks                           )
c (---------------------------------------------------------------------)
c (    Sept 1987 | Stuart Gathman    | Function coded                     )
c (  02 Oct 1991 | Charles McGrath   | Comments added                     )
c (     Nov 1992 | Linda Hitney      | Converted from PASCAL to FORTRAN   )
c (=====================================================================)
c (! This subroutine calculates the dynamic roughness,  zzero in meters as
   ! a function of wind speed following Charnock(1955) and expresses
   ! friction velocity in terms of the drag coefficient and measured
   ! wind spd.)


      if (liws.NE.-1) then
        Zzero  =  C10(lws, liws, lq)*0.000333*lws*lws
      else
        lq  =  lq-2
        Zzero  =  1.0E-05
      endif
      end    !  Zzero
```

```
      FUNCTION Smixr(X, T)
c {=============================================================================}
c {  Purpose: Calculates saturation mixing ratio at altitude x and temp T     }
c {                                                                           }
c {  Preconditions:                                                           }
c {  Global Variables:                                                        }
c {  Revision History:                                                        }
c {---------------------------------------------------------------------------}
c {   Date        |    Programmer     |  Remarks                              }
c {---------------------------------------------------------------------------}
c {    Sept 1987  | Stuart Gathman    | Function coded                        }
c {  02 Oct 1991  | Charles McGrath   | Comments added                        }
c {     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN      }
c {=============================================================================}
c {| Calculates the saturation mixing ratio in kg/kg or g/g at height x
   | and temperature T(c).}

      Vpsm  = vappr(t)
      Smixr =  0.622*vpsm/(PaLt(x)-Vpsm)
      end  ! Smixr


      FUNCTION Fqstar( Idpt, Iiws, Iws, Isst, Iq)
c {=============================================================================}
c {  Purpose:                                                                 }
c {  This is the "Friction Mixing Ratio" used in QZ and calculated using      }
c {  the approximations of the bulk aerodynamic method: Roll(1965) p252,272   }
c {                                                                           }
c {  Preconditions:                                                           }
c {  Global Variables:                                                        }
c {  Revision History:                                                        }
c {---------------------------------------------------------------------------}
c {   Date        |    Programmer     |  Remarks                              }
c {---------------------------------------------------------------------------}
c {   Sept  1987  | Stuart Gathman    | Function coded                        }
c {  02 Oct 1991  | Charles McGrath   | Comments added                        }
c {     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN      }
c {=============================================================================}

      Fk =  0.38
      zz  =  10.0
      yy  =  Idpt/10.0
      Qa  =  Smixr(zz, yy)
      zz  =  0.0
      yy  =  Isst/10.0
      Q0  =  0.98*Smixr(zz, yy)
      Ce  =  C10(Iws, Iiws, Iq)
      Fqstar =  SQRt(Ce)*(Qa-Q0)/Fk
      end  ! Fqstar


      FUNCTION Th( T, Z)
c {=============================================================================}
c {  Purpose:  converts temperature (K) to potential temperature(k) at alt z  }
c {                                                                           }
c {                                                                           }
c {  Preconditions:                                                           }
c {  Global Variables:                                                        }
c {  Revision History:                                                        }
c {---------------------------------------------------------------------------}
c {   Date        |    Programmer     |  Remarks                              }
c {---------------------------------------------------------------------------}
c {   Sept  1987  | Stuart Gathman    | Function coded                        }
c {  02 Oct 1991  | Charles McGrath   | Comments added                        }
c {     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN      }
c {=============================================================================}

      Th  =  T*EXP(0.286*log(1000.0/PaLt(Z)))
      end  !  Th
```

```
      FUNCTION FmLnt(lws, liws, lsst, lat, lq)
c (=====================================================================)
c (  Purpose:  This is an approximation for the mixing length using     )
c )
c (           ROLL (1965), p144, 252.                                   )
c (                                                                     )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (---------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                        )
c (---------------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                  )
c (  02 Oct 1991  | Charles McGrath   | Comments added                  )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=====================================================================)

      if (liws .EQ.  -1) then
        FmLnt  =  -100.0
        lq  =  lq-2
      else
        T0  =  273.15+lsst/10.0
        Ta  =  273.15+lat/10.0
        Ca  =  C10(lws, liws, lq)
        X  =  T0*SQRt(Ca)*0.2646*lws*lws
        zz1  =  0.0
        zz2  =  10.0
        Y  =  9.8*0.38*(Th(T0, zz1)-Th(Ta, zz2))
        FmLnt  =  -X/Y
      endif
      end  ! FmLnt


      FUNCTION Qz(liws, lws, lq, lsst, ldpt, Z, lat)
c (=====================================================================)
c (  Purpose: This subroutine calculates the mixing ratio (Qz) in g/g   )
c (           at altitude,  z in an atmosphere which obeys a Log-Linear )
c (           relationship :  Roll(1965), p273.                         )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (---------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                        )
c (---------------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                  )
c (  02 Oct 1991  | Charles McGrath   | Comments added                  )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=====================================================================)


      T0  =  lsst/10.0
      zz1  =  0.0
      Q0  =  0.98*Smixr(zz1, T0)
      Qsr =  Fqstar(ldpt, liws, lws, lsst, lq)
      Z0  =  Zzero(liws, lws, lq)
      X   =  (Z+Z0)/Z0
      FL  =  FmLnt(lws, liws, lsst, lat, lq)
      Qz  =  Q0+Qsr*(log(X)+4.8*Z/FL)
      end  !  Qz
```

```
      FUNCTION Thstar(lat, liws, lws, lsst, lq)
c (=================================================================)
c (  Purpose: This is the "Friction Potential Temperature" used in the  )
c (           calculation of thz using the approximation of the bulk    )
c (           aerodynamic method;Roll(1965) pages 252 & 272.            )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (-----------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                      )
c (-----------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                )
c (  02 Oct 1991  | Charles McGrath   | Comments added                )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=================================================================)

      Fk = 0.38
      Ta  = 273.15+lat/10.0
      T0  = 273.15+lsst/10.0
      Ca  = C10(lws, liws, lq)
      zz1  = 10.0
      zz2  = 0.0
      Thstar  =  SQRt(Ca)*(Th(Ta, zz1)-Th(T0, zz2))/Fk
      end  ! Thstar

      FUNCTION Thz(liws, lws, lq, lsst, lat, Z)
c (=================================================================)
c (  Purpose:  This function calculates the potential temperature    )
c (           at altitude z in an atmosphere which obeys a Log-Linear   )
c (           relationship :  Roll(1965),  p 273.                       )
c (                                                                     )
c (       th(z) - th(0) =  thstar*(log((z+z0)/z0) + aLpha * z / L       )
c (                                where aLpha =  4.8                   )
c (  Preconditions:                                                     )
c (  Global Variables:                                                  )
c (  Revision History:                                                  )
c (-----------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                      )
c (-----------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                )
c (  02 Oct 1991  | Charles McGrath   | Comments added                )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN )
c (=================================================================)

      T0  = 273.15+lsst/10.0  ! (T0 is the sea surface temp in K.)
      X  = 0.0
      Th0  =  Th(T0, X)   ! (Th0 is the "potential temperature" at surface.)
      Z0  =  Zzero(liws, lws, lq)
      X  =  (Z+Z0)/Z0
      FL  =  FmLnt(lws, liws, lsst, lat, lq)  ! (Monin Obukhov mixing Length)
      Tstar  =  Thstar(lat, liws, lws, lsst, lq)
      Thz  =  Th0+Tstar*(log(X)+4.8*Z/FL)
      end  ! Thz


      FUNCTION Vp( R, Z)
c (=================================================================)
c (  Purpose:  To convert mr(g/kg) at z to vp(mb)                       )
c (  Revision History:                                                  )
c (-----------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                      )
c (-----------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                )
c (  02 Oct 1991  | Charles McGrath   | Comments added                )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=================================================================)
      Vp  =  PaLt(Z)*R/(0.622+R)
      end  ! Vp
```

```
      real FUNCTION LOG10(X)! real) : real;
c {=============================================================================}
c ( Purpose:  Calculate log to base 10 of the argument.                         )
c {-----------------------------------------------------------------------------}
c (   Date        |    Programmer     | Revision History                        )
c {-----------------------------------------------------------------------------}
c (   Sept  1987  | Stuart Gathman    | Function coded                          )
c (  02 Oct 1991  | Charles McGrath   | Comments added                          )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN        )
c {=============================================================================}

      if (x.LT.0.0) write(*,*) 'error in Log10 routine'
      LOG10  =  Log(x)/Log(10.0)
      end


      FUNCTION Zcon( Tasa, Rsa, Zsa)
c {=============================================================================}
c ( Purpose: This subroutine calculates the lifting condensation level where    )
c (      tasa is the potential temperature in Kelvin at Alt,  zsa and Rsa is    )
c (      the mixing ratio at this level in g/kg.  This formulation is an       )
c (      empirical fit to data in the Smithsonian Met. Tables,  p.328.         )
c (  Revision History:                                                          )
c {-----------------------------------------------------------------------------}
c (   Date        |    Programmer     | Remarks                                 )
c {-----------------------------------------------------------------------------}
c (   Sept  1987  | Stuart Gathman    | Function coded                          )
c (  02 Oct 1991  | Charles McGrath   | Comments added                          )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN        )
c {=============================================================================}

      Tsa  =  Tasa/EXP(0.286*log(1000.0/PaLt(Zsa)))
      Esa  =   Vp(Rsa, Zsa)
      Fk   =  1.0/0.286
      X    =  EXP(Fk*log(Tsa))
      Tc   =  732.02-150.41*(Log10(X)-LOG10(Esa))
      Tc   =  Tc+7.21*(LOG10(X)-LOG10(Esa))*(LOG10(X)-LOG10(Esa))+273.15
      Pz   =  1000.0*EXP(-1.0*Fk*log(Tasa/Tc))
      Zcon =  ALtitude(Pz)
      end  !  Zcon


      FUNCTION FNTabs(I)
c {=============================================================================}
c (  Purpose:  converts input values of I (10*T in integer form) into T(K)      )
c (  Revision History:                                                          )
c {-----------------------------------------------------------------------------}
c (   Date        |    Programmer     | Remarks                                 )
c {-----------------------------------------------------------------------------}
c (   Sept  1987  | Stuart Gathman    | Function coded                          )
c (  02 Oct 1991  | Charles McGrath   | Comments added                          )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN        )
c {=============================================================================}
c begin
      FNTabs  =  273.15+I/10.0     ! ( I )
      end  !  FNTabs


      FUNCTION Fdpt(T10, Qp)
c {=============================================================================}
c (  Purpose:  Calculate the dew point given the temperature t10 and Qp         )
c (  Revision History:                                                          )
c {-----------------------------------------------------------------------------}
c (   Date        |    Programmer     | Remarks                                 )
c {-----------------------------------------------------------------------------}
c (   Sept  1987  | Stuart Gathman    | Function coded                          )
c (  02 Oct 1991  | Charles McGrath   | Comments added                          )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN        )
c {=============================================================================}
```

```
      Eprime  =  Qp*1.013/(0.62197+Qp*0.001)
      Ttest = T10     !Start with dew point at air temp and go down.
      do while (Vappr(Ttest).GT.Eprime)
         Ttest  =  Ttest-0.1
      enddo
      Fdpt  =  Ttest
      end  ! Fdpt

      FUNCTION xtoy(x, y)
c (=================================================================)
c (  Purpose: Calculate the value of X raised to the yth power     )
c (  Revision History:                                             )
c (-----------------------------------------------------------------)
c (   Date       |    Programmer    | Remarks                     )
c (-----------------------------------------------------------------)
c (   Sept  1987 | Stuart Gathman   | Function coded              )
c (  02 Oct 1991 | Charles McGrath  | Comments added              )
c (     Nov 1992 | Linda Hitney     | Converted from PASCAL to FORTRAN )
c (=================================================================)

      if ((y .EQ. 0.0) .and. (x.NE.0.0)) goto 1000
      if ((x .EQ. 0.0) .and. (y.NE.0.0)) goto 2000
      zzz = y*log(x)
      if (zzz .GT. 25.0) then
        xtoy  =  99999999999999999999.9
          write(*,*) 'numerical errors in xtoy routine'
        goto 9000
      endif
      if ((x.GT.0.0) .and. (y.NE.0.0)) xtoy  =  exp(y*log(x))
      if (x .GE.  0.0) goto 9000
c  At this point we worry about negative x's
      if ((frac(abs(y)) .GT. 0.000001) .and. (x.LT.0.0))  then
        write(*,*) 'error non integer y and negative x'
        xtoy = 99999999999999999.90
        goto 9000
      else
        z =  abs(y/2.0)
        x = -x
        value = exp(y*log(x))
        zz = frac(z)
        if (zz .LT. 0.000001) then
           xtoy = value
        else
           xtoy = -1.0*value
        endif
      endif
      goto 9000
1000  continue
      xtoy  = 1.0
      goto 9000
2000  continue
      xtoy =  0.0
9000  continue
      end  !  xtoy

      FUNCTION frac(x)
      frac=x-int(x)
      end

      real FUNCTION Mu_calc(T)
c (=================================================================)
c (  Purpose: Calculate the dynamic viscosity at T(K).  Based on   )
c (           Sutherland's equation in List p 394.                 )
c (  Revision History:                                             )
c (-----------------------------------------------------------------)
c (   Date       |    Programmer    | Remarks                     )
c (-----------------------------------------------------------------)
c (   Sept  1987 | Stuart Gathman   | Function coded              )
c (  10 Oct 1991 | Charles McGrath  |                             )
c (     Nov 1992 | Linda Hitney     | Converted from PASCAL to FORTRAN )
c (=================================================================)
```

```
      real Mu0
      C = 393.16                        ! DEG K
      Mu0 = 1.83257E-05                 ! kg/m/SEC
      T0 = 296.16                       ! K
      Mu_CALC  =  Mu0*(T0+C)*EXP(1.5*Log(T/T0))/(T+C)
      end ! Mu_calc


      FUNCTION FNR(F)
c (=============================================================================)
c (  Purpose: Specialize function used in Deposition velocity equation.       )
c (  Revision History:                                                        )
c (-----------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                           )
c (-----------------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                    )
c (  10 Oct 1991  | Charles McGrath   |                                   )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=============================================================================)

      FNR  =  (1.0+1.1/(F*F*F))*1000.0
      end !  FNR

      FUNCTION Diff_coef(A0)
c (=============================================================================)
c (  Purpose: Specialize function used to calculate the diffusion coefficient)
c (           based on a curve fit to a graph obtained from Toomey(1977) p66.)
c (  Revision History:                                                        )
c (-----------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                           )
c (-----------------------------------------------------------------------)
c (   Sept  1987  | Stuart Gathman    | Function coded                    )
c (  10 Oct 1991  | Charles McGrath   |                                   )
c (     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN )
c (=============================================================================)

      Aa=100.0*A0                       ! converts from meters to cm.)
      Bb=1.5849E-16/(Aa*Aa)+1.3E-11/Aa ! Toomey (1977) fig 3.6 p66)
      Diff_coef=Bb * 0.0001             ! converts from cm^2/sec to m^2/sec.)
      end ! (function Diff_coef)


      FUNCTION FNH(A)
      FNH  =  MAX(0.0, A)
      end


      FUNCTION FNA(Amp)
      FNA  =  2000.0*Amp*Amp
      end


      FUNCTION FNB(Uave)
      FNB  =  MAX(0.5, 5.866*(Uave-2.2))
      end


      FUNCTION FNC(U)
      FNC  =  exp((0.06*U-2.8)*2.3025850930) ! power10 function
      END


      character*2 FUNCTION pad(Value)
      integer*2  Value
      character*2 S
      write(S,'(i2.2)') Value
      Pad = S
      end  !  pad
```

```
      FUNCTION Td( Q, P)
c (=============================================================================)
c (  Purpose: Calculate dew point temperature from mixing ration and pres.     )
c (           see Bolton MWR (1980) v108.                                      )
c (  Called by:  Parcel2                                                       )
c (  Calls Out:  (none)                                                        )
c (  Revision History:                                                         )
c (-----------------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                               )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath    |                                       )
c (   5 Oct 1992 | Stuart Gathman     |  Clean up the set of routines         )
c (=============================================================================)

      Qw = Q/1000.0
      E  = P*Qw/(0.622+Qw)
      Td = (243.5*Log(E)-440.8)/(19.48-Log(E))

      end ! Td


      FUNCTION rMix_ratio( Tcent, Rh, P)
c (=============================================================================)
c (  Purpose: Calculate mixing ratio from T, Rh and press.                      )
c (           This is an approximation to the goff-gatch formula which is      )
c (           good to 1/2 % error for - to over 25 deg.                        )
c (           ( p is in millibars,  and rMix_ratio will be in g/kg)            )
c (  Called by:  Parcel2                                                      )
c (  Calls out:  (none)                                                        )
c (  Revision History:                                                         )
c (-----------------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                               )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath    |                                       )
c (   5 Oct 1992 | Stuart Gathman     |Cleaned up routines                    )
c (=============================================================================)

      Ex   = 6.112*EXP(17.67*Tcent/(Tcent+243.5))
      Qsat = 622.0*Ex/(P-Ex)  ! List,  p302
      rMix_Ratio  = Qsat*Rh/100.0
      end
```

APPENDIX G SUBROUTINES USED BY NOVAMSR

```
      SUBROUTINE  ReadSetupFile(WaveLengthString, SurfaceFilePath,
     $                          ProfileFilePath, DataType, ReadOk)

c {=============================================================================}
c (  Purpose:  Reads variables from setup file (NOVAM.INI)                      )
c (  Called by:  NOVAM driver                                                   )
c (  Calls out:  none                                                           )
c (  Preconditions:  setup file should exist                                    )
c (-----------------------------------------------------------------------------)
c (   Date        |    Programmer     |  Revision History                       )
c (-----------------------------------------------------------------------------)
c (  10 Feb 1992  | Charles McGrath   | created procedure                       )
c (     Nov 1992  | Linda Hitney      | converted from PASCAL to FORTRAN        )
c (  21 May 1993  | Stu Gathman       | coordinate with NOVAMSW                 )
c {=============================================================================}

      logical ReadOk
      logical FileExists
      integer TextFile
      character*10 WaveLengthString
      character*1  DataType
      character*12 SurfaceFilePath, ProfileFilePath
      character*12 SetupFile

      SetupFile   = 'NOVAM.INI'
      TextFile=30

      inquire(file=SetupFile,exist=FileExists)
      if (FileExists) then
        open(unit=TextFile,file=SetupFile,status='old')
        Read(TextFile,'(a)') WaveLengthString
        Read(TextFile,'(a)') SurfaceFilePath
        Read(TextFile,'(a)')  ProfileFilePath
        Read(TextFile,'(12x,a1)') DataType
        close(TextFile)
        ReadOK = .true.
      else
        ReadOK = .false.
      endif
      end   !  ReadSetupFile


      SUBROUTINE convert
c {=============================================================================}
c (  Purpose:  Converts type aptmr.dat files inot patrh.dat type of file        )
c (  This is a program to convert the altitude, Potential Temperature and       )
c (  mixing ratio data into the pressure, air temperature and rel humidity      )
c (  profile data.  This is the batch file version which doesn't need to ask    )
c (  questions of the operator.                                                 )
c (-----------------------------------------------------------------------------)
c (  Called by:  NOVAMSR                                                        )
c (  Calls out:  none                                                           )
c (  Preconditions:  setup file should exist                                    )
c (-----------------------------------------------------------------------------)
c (   Date        |    Programmer     |  Revision History                       )
c (-----------------------------------------------------------------------------)
c (      Feb 1992 | Stu Gathman       | created subroutine                      )
c {=============================================================================}


      integer input,output
      real logpres,a,b,c,a0,b0,c0,satmr,satvp
      real pres(1000),airt(1000),rh(1000)

c     read in the type 1 data file
      input=10
      open(unit=input,file='aptmr.dat',status='old')
```

```
      output=11
      open(unit=output,file='patrh.dat',status='unknown')

      read(input,*) a,b,c
      a0=a
      b0=b
      c0=c
      j=0
      do while (.not. eof(input))
        read(input,*,end=999) a,b,c
        if (a .GT. a0) then
          j=j+1
          if (a .LT. 500.0) then
            pres(j)=1013.0-55.0*a/500.0
          else
            pres(j)=1021.38*exp(-0.00012739*a)
          endif
          logpres=4342.9448*log(pres(j))
          airt(j)=(b+273.15)/exp(0.286*log(1000.0/pres(j)))-273.15
          satvp=vappr(airt(j))
          satmr=620.0*satvp/(pres(j)- satvp)
          rh(j)=100.0*c/satmr
          write(output,'(i8,4f8.0)') j,logpres,10.0*airt(j),rh(j),
     $                               10.0*pres(j)
          a0=a
          b0=b
          c0=c
        else
          read(input,*,end=999) a0,b0,c0
        endif
      enddo

999   continue
      close(input)
      close(output)
      end   !convert


      SUBROUTINE filter
c {=============================================================================}
c ( Purpose: To filter ROAB data to determine significant levels              )
c ( Called by:  NOVAMSR                                                       )
c ( Calls out:  Histogram, Smooft                                             )
c ( Preconditions:                                                            )
c {-----------------------------------------------------------------------------}
c (  Date       |    Programmer    | Revision History                        )
c {-----------------------------------------------------------------------------}
c (  10 Feb 1992 | Rich Paulus      | created BASIC program                   )
c (     Nov 1992 | Linda Hitney     | converted from BASIC to FORTRAN         )
c (  21 May 1993 | Stu Gathman      | coordinate with NOVAMSW                 )
c {=============================================================================}
      CHARACTER*12 Filename,file
      INTEGER*2 NMax
      INTEGER*2 Time
      INTEGER*2 s
      REAL*4 Binsize
      REAL*4 DT
      LOGICAL Flips

      DIMENSION Temp(21), X(2000), Y(2000), d(2000)
      DIMENSION Time(2000)
      DIMENSION parray(2000), tarray(2000), rharry(2000)
      DIMENSION Psmth(2000), Tsmth(2000), RHsmth(2000)
      DIMENSION dtdp(2000), drhdp(2000), d2tdp2(2000)
      DIMENSION Ptmp(500), Ttmp(500), RHtmp(500)
      DIMENSION Psig(100), Tsig(100), RHsig(100)
```

```
c     This section reads and RS-80 roab data as output from
c     the PP-11.
      N = 1 !default median filter order of 1

      FILE = "patrh.dat"
        Nlevels = 200
      OPEN (unit=12,file=FILE,status='old')
      HTZERO = 0.0
      I = 0
      DO WHILE (.NOT. EOF(12))
        READ(12,*) s, b, c, dinput, E
        p = E / 10.0
        t = c / 10.0
        rh = dinput
        IF (I .EQ. Nlevels) GOTO 7000
        I = I + 1
        Time(I) = s
        parray(I) = p
        tarray(I) = t
        rharry(I) = rh
      ENDDO

7000  CONTINUE
      Nmax = I

      iopt = 1

      IF (iopt .EQ. 1) THEN
         DO I = 1 ,  Nmax
            X(I) = parray(I)
         ENDDO
      ELSEIF (iopt .EQ. 2) THEN
         DO I = 1 ,  Nmax
            X(I) = tarray(I)
         ENDDO
      ELSE
         DO I = 1 ,  Nmax
            X(I) = rharry(I)
         ENDDO
      END IF

c     for Nth order filter, copy N end points from input to output

      DO I = 1 ,  N
         Y(I) = X(I)
         Y(Nmax + 1 - I) = X(Nmax + 1 - I)
         ENDDO
      L = 2 * N + 1       ! window length

c     median filter of order N without replacement

      DO I = N + 1 ,  Nmax - N
         DO J = 1 ,  L
            Temp(J) = X(I + J - N - 1)
         ENDDO

c     sort Temp into ascending order
         Flips = .true.
         DO WHILE (Flips)
            Flips = .false.
            DO J = 2 ,  L
              IF (Temp(J - 1) .GT. Temp(J)) THEN
                Flips = .true.
                Save=Temp(J-1)
                Temp(J-1)=Temp(J)
                Temp(J)=Save
              END IF
            ENDDO
         ENDDO
         Y(I) = Temp(N + 1)        ! median value
      ENDDO
```

```
c     compute difference histogram and determine threshold.  initial bin
c     size is .2 for P & T and 2 for RH

      Binsize = .2
      IF (iopt .EQ. 3) Binsize = 2
      DO I = 1 ,  Nmax
         d(I) = X(I) - Y(I)
      ENDDO
      CALL histogram(d, Nmax, Binsize, DT)

c     generate final output as x(i)

      DO I = 1 ,  Nmax
        IF (ABS(d(I)) .GT. DT) X(I) = Y(I)
      ENDDO

      IF (iopt .EQ. 1) THEN
        pts = 3
      ELSE
        pts = 6
      ENDIF
           CALL SMOOFT(X, Nmax, pts)

      IF (iopt .EQ. 1) THEN
         DO I = 1 ,  Nmax
            Psmth(I) = X(I)
         ENDDO
      ELSEIF (iopt .EQ. 2) THEN
         DO I = 1 ,  Nmax
            Tsmth(I) = X(I)
         ENDDO
      ELSE
         DO I = 1 ,  Nmax
            RHsmth(I) = X(I)
         ENDDO
      END IF

         iopt = 2

      IF (iopt .EQ. 1) THEN
         DO I = 1 ,  Nmax
            X(I) = parray(I)
         ENDDO
      ELSEIF (iopt .EQ. 2) THEN
         DO I = 1 ,  Nmax
            X(I) = tarray(I)
         ENDDO
      ELSE
         DO I = 1 ,  Nmax
            X(I) = rharry(I)
         ENDDO
      END IF

c     for Nth order filter, copy N end points from input to output

      DO I = 1 ,  N
         Y(I) = X(I)
         Y(Nmax + 1 - I) = X(Nmax + 1 - I)
         ENDDO
      L = 2 * N + 1        ! window length

c     median filter of order N without replacement

      DO I = N + 1 ,  Nmax - N
         DO J = 1 ,  L
            Temp(J) = X(I + J - N - 1)
         ENDDO
```

```
c     sort Temp into ascending order

          Flips = .true.
          DO WHILE (Flips)
             Flips = .false.
             DO J = 2 ,  L
           IF (Temp(J - 1) .GT. Temp(J)) THEN
             Flips = .true.
               Save=Temp(J-1)
               Temp(J-1)=Temp(J)
               Temp(J)=Save
            END IF
             ENDDO
          ENDDO
          Y(I) = Temp(N + 1)          ! median value
        ENDDO

c  compute difference histogram and determine threshold.  initial bin
c  size is .2 for P & T and 2 for RH

        Binsize = .2
        IF (iopt .EQ. 3) Binsize = 2
        DO I = 1 ,  Nmax
          d(I) = X(I) - Y(I)
        ENDDO
        CALL histogram(d, Nmax, Binsize, DT)

c     generate final output as x(i)

        DO I = 1 ,  Nmax
          IF (ABS(d(I)) .GT. DT) X(I) = Y(I)
        ENDDO

        IF (iopt .EQ. 1) THEN
          pts = 3
        ELSE
          pts = 6
        ENDIF
        CALL SMOOFT(X, Nmax, pts)

        IF (iopt .EQ. 1) THEN
           DO I = 1 ,  Nmax
              Psmth(I) = X(I)
           ENDDO
        ELSEIF (iopt .EQ. 2) THEN
           DO I = 1 ,  Nmax
              Tsmth(I) = X(I)
           ENDDO
        ELSE
           DO I = 1 ,  Nmax
              RHsmth(I) = X(I)
           ENDDO
        END IF

           iopt = 3

        IF (iopt .EQ. 1) THEN
           DO I = 1 ,  Nmax
              X(I) = parray(I)
           ENDDO
        ELSEIF (iopt .EQ. 2) THEN
           DO I = 1 ,  Nmax
              X(I) = tarray(I)
           ENDDO
        ELSE
           DO I = 1 ,  Nmax
              X(I) = rharry(I)
           ENDDO
        END IF
```

```
c   for Nth order filter, copy N end points from input to output

      DO I = 1 ,  N
         Y(I) = X(I)
         Y(Nmax + 1 - I) = X(Nmax + 1 - I)
         ENDDO
      L = 2 * N + 1        ! window length

c   median filter of order N without replacement

      DO I = N + 1 ,  Nmax - N
         DO J = 1 ,  L
            Temp(J) = X(I + J - N - 1)
         ENDDO

c   sort Temp into ascending order

          Flips = .true.
          DO WHILE (Flips)
            Flips = .false.
            DO J = 2 ,  L
              IF (Temp(J - 1) .GT. Temp(J)) THEN
                Flips = .true.
               Save=Temp(J-1)
               Temp(J-1)=Temp(J)
               Temp(J)=Save
              END IF
            ENDDO
          ENDDO
          Y(I) = Temp(N + 1)        ! median value
        ENDDO

c   compute difference histogram and determine threshold.  initial bin
c   size is .2 for P & T and 2 for RH

        Binsize = .2
        IF (iopt .EQ. 3) Binsize = 2
        DO I = 1 ,  Nmax
          d(I) = X(I) - Y(I)
        ENDDO
        CALL histogram(d, Nmax, Binsize, DT)

c   generate final output as x(i)

        DO I = 1 ,  Nmax
           IF (ABS(d(I)) .GT. DT) X(I) = Y(I)
        ENDDO

        IF (iopt .EQ. 1) THEN
          pts = 3
        ELSE
          pts = 6
        ENDIF
        CALL SMOOFT(X, Nmax, pts)

        IF (iopt .EQ. 1) THEN
           DO I = 1 ,  Nmax
              Psmth(I) = X(I)
           ENDDO
        ELSEIF (iopt .EQ. 2) THEN
           DO I = 1 ,  Nmax
              Tsmth(I) = X(I)
           ENDDO
        ELSE
           DO I = 1 ,  Nmax
              RHsmth(I) = X(I)
           ENDDO
        END IF
```

```
Analysis:

c  This section analyzes the smoothed P,T,U profiles to determine the
c  significant levels.  Central difference finite derivatives are
c  calculated to locate relative max/min and saddle points.

      dtdp(1) = 0
      dtdp(Nmax) = 0
      drhdp(1) = 0
      drhdp(Nmax) = 0
      d2tdp2(1) = 0
      d2tdp2(Nmax) = 0
      DO I = 2 ,  Nmax - 1
         dp = LOG(Psmth(I - 1) / Psmth(I + 1))
         dtdp(I) = (Tsmth(I + 1) - Tsmth(I - 1)) / dp
         drhdp(I) = (RHsmth(I + 1) - RHsmth(I - 1)) / dp
         d2tdp2(I) = (Tsmth(I - 1) - 2 * Tsmth(I) +
     $    Tsmth(I + 1)) / ((.5 * dp) ** 2
      ENDDO

      Ptmp(1) = Psmth(1)
      Ttmp(1) = Tsmth(1)
      RHtmp(1) = RHsmth(1)

c     look for change in sign of derivatives to pick level

      J = 1
      DO I = 3 ,  Nmax - 1
        IF (SGN(dtdp(I)) .NE. SGN(dtdp(I - 1)) .OR.
     $      SGN(drhdp(I)) .NE. SGN(drhdp(I - 1))) THEN
          J = J + 1
          Ptmp(J) = Psmth(I)
          Ttmp(J) = Tsmth(I)
          RHtmp(J) = RHsmth(I)
        ELSEIF (SGN(d2tdp2(I)) .NE. SGN(d2tdp2(I - 1))) THEN
          J = J + 1
          Ptmp(J) = Psmth(I)
          Ttmp(J) = Tsmth(I)
          RHtmp(J) = RHsmth(I)
        END IF
      ENDDO
      Jmax = J + 1
      Ptmp(Jmax) = Psmth(Nmax)
      Ttmp(Jmax) = Tsmth(Nmax)
      RHtmp(Jmax) = RHsmth(Nmax)


c    select significant levels iaw FMH #3

6000  CONTINUE

      Ttol = .2
      RHtol = 2

      Psig(1) = Ptmp(1)
      Tsig(1) = Ttmp(1)
      RHsig(1) = RHtmp(1)
      Kmax = 1

      Lastsig = 1
      DO J = 3 ,  Jmax
        DO L = Lastsig + 1 ,  J
          Dlog = LOG(Psig(Kmax) / Ptmp(J)) / LOG(Psig(Kmax) / Ptmp(L))
          t = Tsig(Kmax) - (Tsig(Kmax) - Ttmp(J)) / Dlog
          rh = RHsig(Kmax) - (RHsig(Kmax) - RHtmp(J)) / Dlog
          IF (ABS(Ttmp(L) - t) .GT. Ttol .OR.
     $        ABS(RHtmp(L) - rh) .GT. RHtol) THEN
            Kmax = Kmax + 1
            Psig(Kmax) = Ptmp(L)
            Tsig(Kmax) = Ttmp(L)
            RHsig(Kmax) = RHtmp(L)
```

```
            Lastsig = L
            GOTO 5000
          END IF
        ENDDO

5000  CONTINUE
      ENDDO
      Kmax = Kmax + 1
      Psig(Kmax) = Ptmp(Jmax)
      Tsig(Kmax) = Ttmp(Jmax)
      RHsig(Kmax) = RHtmp(Jmax)

      Filename = "sigfile"
      OPEN (unit=13,file=Filename,status='unknown')
      WRITE(13,*) Kmax
      DO I = 1 ,  Kmax
        WRITE(13,2000) Psig(I), Tsig(I), RHsig(I)

2000    FORMAT(f7.1,f8.1,f8.1)
      ENDDO
      CLOSE (12) !"patrh.dat"
      CLOSE (13) !"sigfile"
      RETURN
      END   ! filter


      SUBROUTINE histogram (d, Nmax, Binsize, DT)
c (=============================================================================)
c (  Purpose:                                                                   )
c (  Called by:  filter                                                         )
c (  Calls out: zerobin                                                         )
c (  Comments:   d is the datasarray, Nmax  is the number of points,            )
c (        Binsize is the initial size, and DT is the difference threshold.     )
c (        construct a 25 bin histogram with bin size of !Binsize! from the     )
c (        difference data in array !D!.  The histogram is checked for          )
c (        empty bins; IF (none exist, the bin size is doubled until at         )
c (        least one bin is empty.   The difference threshold, DT, is           )
c (        THEN calculated and returned.                                        )
c (-----------------------------------------------------------------------------)
c (  Date        |    Programmer     | Revision History                       )
c (-----------------------------------------------------------------------------)
c (  10 Feb 1992 | Rich Paulus       | created BASIC program                  )
c (     Nov 1992 | Linda Hitney      | converted from BASIC to FORTRAN        )
c (  21 May 1993 | Stu Gathman       | coordinate with NOVAMSW                )
c (=============================================================================)

      REAL*4 D(2000)
      INTEGER*2 Nmax
      REAL*4 Binsize
      REAL*4 DT
      INTEGER*2 Counts
      DIMENSION Counts(25)
      INTEGER*2 Index
      Index = 0
      DO WHILE (Index .EQ. 0)
        DO I = 1 ,  25
          Counts(I) = 0
        ENDDO
        DO I = 1 ,  Nmax
          IF (ABS(d(I)) .GT. Binsize * 11 + Binsize / 2) THEN
            IF (d(I) .LT. 0) THEN
              J = 1
            ELSE
              J = 25
            END IF
          ELSE
            IF (d(I) .LT. 0) THEN
               J = INT(d(I) / Binsize - .5) + 13
            ELSE
               J = INT(d(I) / Binsize + .5) + 13
```

```
          END IF
        END IF
        Counts(J) = Counts(J) + 1
      ENDDO
      CALL zerobin(Counts, Index)
      IF (Index .NE. 0) THEN
        DT = Index * Binsize + Binsize / 2
      ELSE
        Binsize = Binsize * 2
      END IF
    ENDDO
    END  !histogram


    SUBROUTINE SMOOFT (Y, N, pts)
c (=====================================================================)
c (  Purpose: To smooth an array of data                                 )
c (  Called by:  filter                                                  )
c (  Calls out:  REALFT                                                  )
c (  Comments: Smooths an array Y of length N, with a window whose full width)
c (           is of order PTS neighboring points, a user supplied value. Y is)
c (           modified.  From Numerical Recipes by Press et al, Ch 13.9  )
c (---------------------------------------------------------------------)
c (  Date        |    Programmer     |  Revision History                 )
c (---------------------------------------------------------------------)
c (  10 Feb 1992 | Rich Paulus       | created BASIC program             )
c (     Nov 1992 | Linda Hitney      | converted from BASIC to FORTRAN   )
c (  21 May 1993 | Stu Gathman       | coordinate with NOVAMSW           )
c (=====================================================================)

    REAL*4 Y(2000)
    INTEGER*2 N
    INTEGER*2 No2
    INTEGER*2 Isign
    Nmax = 1024                 !max size of padded array
    M = 2
    Nmin = N + 2 * pts          ! min size including buffer against wrap around
    DO WHILE (M .LT. Nmin)      ! find next larger power of 2
      M = 2 * M
    ENDDO
    IF ((M .GT. Nmax)) THEN
      STOP "Nmax too small"
    END IF
    Constant = (pts / M) ** 2 ! useful constants below
    Y1 = Y(1)
    YN = Y(N)
    RN1 = 1.0 / float(N - 1)
    DO J = 1 , N                ! remove linear trend & transfer data
      Y(J) = Y(J) - RN1 * (Y1 * (N - J) + YN * (J - 1))
    ENDDO
    IF (((N + 1) .LE. M)) THEN   ! zero pad
      DO J = N + 1 , M
        Y(J) = 0
      ENDDO
    END IF
    No2 = M / 2
    Isign=1
    CALL REALFT(Y, No2, Isign)       ! Fourier transform
    Y(1) = Y(1) / No2
    FAC = 1                     ! window function
    DO J = 1 , No2 - 1         ! multiply the data by the window function
      K = 2 * J + 1
      IF ((FAC .NE. 0)) THEN
        FAC = (1 - Constant * J ** 2) / No2
        IF (FAC .LT. 0.0) FAC = 0.0
        Y(K) = FAC * Y(K)
        Y(K + 1) = FAC * Y(K + 1)
      ELSE                          ! don!t do unnecessary multiplies after
        Y(K) = 0                    ! window function is zero
        Y(K + 1) = 0
```

```
      END IF
    ENDDO
    FAC = (1 - .25 * pts ** 2) / Mo2       ! last point
    IF (FAC .LT. 0.0) FAC = 0.0
    Y(2) = FAC * Y(2)
    Isign=-1
    CALL REALFT(Y, Mo2, Isign)          ! inverse Fourier transform
    DO J = 1 , N                        ! restore linear trend
      Y(J) = RN1 * (Y1 * (N - J) + YN * (J - 1)) + Y(J)
    ENDDO
    END  ! Smooft


    SUBROUTINE zerobin (Counts, Index)
c {=============================================================================}
c {  Purpose: To find empty bins in a histogram                                 }
c {  Called by:  histogram                                                      }
c {  Calls out:                                                                 }
c {  Comments: find first empty bin either side of bin 13;                      }
c {            IF (none found,) THEN   indicate by setting Index=0              }
c {-----------------------------------------------------------------------------}
c {  Date        |    Programmer     | Revision History                       }
c {-----------------------------------------------------------------------------}
c {  10 Feb 1992 | Rich Paulus       | created BASIC program                  }
c {     Nov 1992 | Linda Mitney      | converted from BASIC to FORTRAN        }
c {  21 May 1993 | Stu Gathman       | coordinate with NOVAMSW                }
c {=============================================================================}


    INTEGER*2 Counts
    DIMENSION Counts(25)
    INTEGER*2 Index
    Index = 0
    Index1 = 0
    Index2 = 0
    DO I = 1 ,  12
          IF (Counts(I) .EQ. 0) Index1 = ABS(I - 13)
          IF (Counts(26 - I) .EQ. 0) Index2 = ABS(I - 13)
    ENDDO

    IF (Index1 .NE. 0 .AND. Index2 .NE. 0) THEN
       IF (Index1 .LT. Index2) THEN
          Index = Index1
         ELSE
            Index = Index2
         END IF
    ELSEIF (Index1 .NE. 0 .OR. Index2 .NE. 0) THEN
          IF (Index1 .GT. Index2) THEN
            Index = Index1
          ELSE
            Index = Index2
          END IF
    ELSE
       Index = 0
    END IF

    END  ! zerobin


    SUBROUTINE REALFT (Y, N, Isign)
c {=============================================================================}
c {  Purpose: To calculate a Fourier Transform                                  }
c {  Called by:  Smooft                                                         }
c {  Calls out: FOUR1                                                           }
c {  Comments: Calculates the Fourier Transform of a set of 2N real-valued      }
c {  data points.  Replaces this data by the positive frequency half of its     }
c {  complex Fourier Transform.  The real-valued first and last                 }
c {  components of the complex transform are returned as elements Y(1)          }
c {  and Y(2) respectively.  N must be a power of 2.  This routine also         }
c {  calculates the inverse transform of a complex data array IF (it is         }
c {  the transform of real data.  (Result in this case must be multiplied       }
```

```
c (  by 1/N.)                                                          )
c (-------------------------------------------------------------------)
c (   Date       |    Programmer     |  Revision History               )
c (-------------------------------------------------------------------)
c (  10 Feb 1992 | Rich Paulus       | created BASIC program           )
c (     Nov 1992 | Linda Hitney      | converted from BASIC to FORTRAN )
c (  21 May 1993 | Stu Gathman       | coordinate with NOVAMSW         )
c (===================================================================)

      REAL*8 WPR,WPI,WR,WI,Theta,WTEMP
      REAL*4 Y(2000)
      INTEGER*2 N
      INTEGER*2 Isign
      INTEGER*2 Isgn

      Theta = 3.14159265358979300 / DBLE(N)
      C1 = .5
      IF ((Isign .EQ. 1)) THEN
         C2 = -.5
         Isgn = 1
         CALL FOUR1(Y, N, Isgn) ! the forward transform is here
      ELSE                      ! otherwise set up for an inverse transform
         C2 = .5
         Theta = -Theta
      END IF

      WPR = -2.000 * SIN(.500 * Theta) ** 2
      WPI = SIN(Theta)
      WR = 1.000 + WPR
      WI = WPI
      N2P3 = 2 * N + 3
      DO I = 2 ,  N / 2 + 1          ! case I=1 done separately below
         I1 = 2 * I - 1
         I2 = I1 + 1
         I3 = N2P3 - I2
         I4 = I3 + 1
         WRS = REAL(WR)
         WIS = REAL(WI)
         H1R = C1 * (Y(I1) + Y(I3))  ! The 2 separate transforms are separated
         H1I = C1 * (Y(I2) - Y(I4))  !  out OF z
         H2R = -C2 * (Y(I2) + Y(I4))
         H2I = C2 * (Y(I1) - Y(I3))
         Y(I1) = H1R + WRS * H2R - WIS * H2I  ! Here they are recombined to form
         Y(I2) = H1I + WRS * H2I + WIS * H2R  ! the true transforms of the
         Y(I3) = H1R - WRS * H2R + WIS * H2I  ! original real data
         Y(I4) = -H1I + WRS * H2I + WIS * H2R
         WTEMP = WR                           ! The recurrence
         WR = WR * WPR - WI * WPI + WR
         WI = WI * WPR + WTEMP * WPI + WI
      ENDDO
      IF ((Isign .EQ. 1)) THEN
         H1R = Y(1)
         Y(1) = H1R + Y(2)
         Y(2) = H1R - Y(2)             !Squeeze the first and last data together
      ELSE                             !to get them all within the original
         H1R = Y(1)                    !array
         Y(1) = C1 * (H1R + Y(2))
         Y(2) = C1 * (H1R - Y(2))
         Isgn = -1
         CALL FOUR1(Y, N, Isgn)        !This is the inverse transform for the
      END IF                           !case Isign=1

      END  !realft
```

```
      SUBROUTINE FOUR1 (Y, NN, Isign)
c {=============================================================================}
c (  Purpose:                                                                   )
c (  Called by: REALFT                                                          )
c (  Calls out: FOUR1                                                           )
c (  Comments:  Replaces Y by its discrete Fourier Transform, IF (Isign=1; or)
c (          replaces Y by NN times its inverse discrete Fourier transform if
c (          Isign=-1.  Y is a complex array of length NN or, equivalently, a
c (          real array of length 2*NN.  NN MUST be an integer power of 2.
c (----------------------------------------------------------------------------)
c (   Date       |    Programmer     |  Revision History                       )
c (----------------------------------------------------------------------------)
c (  10 Feb 1992 | Rich Paulus       | created BASIC program                   )
c (     Nov 1992 | Linda Hitney      | converted from BASIC to FORTRAN         )
c (  21 May 1993 | Stu Gathman       | coordinate with NOVAMSW                 )
c {=============================================================================}

      REAL*8 WPR,WPI,WR,WI,Theta,WTEMP
      REAL*4 Y(2000)
      INTEGER*2 NN
      INTEGER*2 Isign
      N = 2 * NN
      J = 1
      DO I = 1 ,  N , 2              ! This is the bit reversal section
        IF ((J .GT. I)) THEN
          TempR = Y(J)
          TempI = Y(J + 1)
          Y(J) = Y(I)
          Y(J + 1) = Y(I + 1)
          Y(I) = TempR
          Y(I + 1) = TempI
        END IF
        M = N / 2
        DO WHILE ((M .GE. 2) .AND. (J .GT. M))
          J = J - M
          M = M / 2
        ENDDO
        J = J + M
      ENDDO
      Mmax = 2                  ! Here begins the Danielson-Lanczos section
      DO WHILE (N .GT. Mmax)  ! outer loop executed LOG(NN) (base 2) times
        Istep = 2 * Mmax

c  Init for trig recurrence

        Theta = 6.283185307179590D0 /DBLE(Isign * Mmax)
        WPR = -2.000 * SIN(.500 * Theta) ** 2
        WPI = SIN(Theta)
        WR = 1.000
        WI = 0.000
        DO M = 1 ,  Mmax , 2          !Here are 2 nested inner loops
          DO I = M ,  N , Istep
            J = I + Mmax           ! This is the Danielson-Lanczos formula
            TempR = REAL(WR) * Y(J) - REAL(WI) * Y(J + 1)
            TempI = REAL(WR) * Y(J + 1) + REAL(WI) * Y(J)
            Y(J) = Y(I) - TempR
            Y(J + 1) = Y(I + 1) - TempI
            Y(I) = Y(I) + TempR
            Y(I + 1) = Y(I + 1) + TempI
          ENDDO
          WTEMP = WR   ! trignometric recurrence
          WR = WR * WPR - WI * WPI + WR
          WI = WI * WPR + WTEMP * WPI + WI
        ENDDO
        Mmax = Istep
      ENDDO
      END  ! four1
```

```
      SUBROUTINE NOVAMW( RSData, zalt, ko, mmode, Calt, SOData)
c {=============================================================================}
c {  Purpose:                                                                   }
c {                                                                             }
c {  Called by: MSR_WEAK.FOR    (The Weak Convection NOVAM Case)                }
c {  Calls:                                                                     }
c {-----------------------------------------------------------------------------}
c {  Comments:                                                                  }
c { Input variables are :                                                       }
c {     zalt :  the altitude in meters at which conc is desired.                }
c {       ko :  the correction factor at the surface                            }
c {    mmode :  the integer mode number (0..3).                                 }
c {      amp :  the air mass parameter.                                         }
c {                                                                             }
c { Output variable is  :  Calt :   the concentration of the nth mode at alt.   }
c { Note that the input calculations for Csurf  ie. the concentration of the    }
c { nth mode at the surface is done independently in this procedure.  It is     }
c { however modified by the correction factor,  ko,  obtained from the input    }
c { visibility measurements - sgg 1/22/91.                                      }
c {                                                                             }
c { There are certain assumptions made in this routine which are :              }
c { for the sea salt components,  mode 2 is mixed in the cloudy layer model.    }
c { where as the mode 3 component is an exponential scale height decay.         }
c {                                                                             }
c { More on mode 0,  if amp > 5 then this is mixed through out the layer much   }
c { like the sea salt mode 2.  The value at the surface is 30% of what          }
c { would be predicted for the old Mode 1 system.  When amp >5, then            }
c { above the marine layer, the concentration is 100% what it would have        }
c { been under the old  skeme while on the other hand, the value at the         }
c { surface is 70% of the old skeme values.  On the other hand, if amp <=5,     }
c { then the mode 0 component is zero completely and the mode 1 component       }
c { has no vertical structure.                                                  }
c {-----------------------------------------------------------------------------}
c {  Revision History:                                                          }
c {-----------------------------------------------------------------------------}
c {   Date        |    Programmer     |  Remarks                                }
c {-----------------------------------------------------------------------------}
c {  21 Aug 1991  | Stuart Gathman    | Improved version coded for PASCAL       }
c {  02 Oct 1991  | Charles McGrath   | Comments added                          }
c {     Nov 1992  | Linda Hitney      |  Converted from PASCAL to FORTRAN       }
c {     May 1993  | Stuart Gathman    |  Integrated with NOVAMSR                }
c {=============================================================================}



      real ko
      real n, n1, n2, Cip, Csurf

      include 'rsdata.inc'
      include 'sodata.inc'


c (define the local variables from the RSAry matrix values)
      DeltaQb  =  ( RSData.Qbp - RSData.Qbm )
      DeltaTb  =  ( RSData.Tbp - RSData.Tbm )
      DeltaQi  =  ( RSData.Qip - RSData.Qim )
      DeltaTi  =  ( RSData.Tip - RSData.Thim )
      DeltaZ   =  ( RSData.Zi  - RSData.ZBase )

c (Here I want to make checks on the signs of the Deltas)
      if (DeltaQb.GT.0.0) DeltaQb = 0.0
      if (DeltaQi.GT.0.0) DeltaQi = 0.0
      if (DeltaTb.LT.0.0) DeltaTb = 0.0
      if (DeltaTi.LT.0.0) DeltaTi = 0.0
      if (DeltaZ.LE.0.0)  write(*,*) 'error in radiosondes Z'

      GamaT    =  ( RSData.Thim - RSData.Tbp )/DeltaZ
      GamaQ    =  ( RSData.Qim  - RSData.Qbp )/DeltaZ
      n1       =  (DeltaQb*GamaT - DeltaTb*GamaQ)*Deltaz
      n2       =  (DeltaTi*DeltaQb - DeltaQi*DeltaTb)
      n        =  n1/n2      ! (This is the "n" in Davidson * Fairall's paper)
```

```
c (What is the concentration above the cloud layer? )
      if (mmode .EQ. 1) then
        cip =  fna(SOData.AMP)
      else
        cip =  0.0
      endif

c (What is the concentration at the sea surface?)
      if (mmode .EQ. 0) then
        if (SOData.AMP .GT. 5.0) then
          Csurf  =  0.3*fna(SOData.AMP)
        else
          Csurf  =  0.0
        endif
      endif

      if (mmode .EQ.  1) then
        if (SOData.AMP .GT. 5.0) then
          Csurf  =  0.7 * fna(SOData.AMP)
        else
          Csurf  =  fna(SOData.AMP)
        endif
      endif

      if (mmode .EQ.  2) Csurf  =  fnb(SOData.UAve)
      if (mmode .EQ.  3) Csurf  =  fnc(SOData.U10)

c (Calculate the gama of concentration here)
      n1       =  ( Cip - Csurf ) * n
      n2       =  ( 1.0 + n ) * DeltaZ
      GamaC    =  n1/n2

c (calculation of the concentration of the mmode mode at altitude zalt.)

c (-------------------------------------------------------------------
c  Assume GamaC has been calculated.  Also assume that the amp, u10,  and
c  uave are known. An input variable,  the height,  Zalt,  is used and the
c  output is the concentration amplitude Calt at that altitude.
c -------------------------------------------------------------------)
      select CASE (mmode)

        case(0)        ! ( The case for the dust component )
          if (zalt .LE. RSData.ZBase) then
            Calt  =  Csurf
          else
            Calt  =  0.3*fna(SOData.AMP)+GamaC*(zalt-RSData.ZBase)
          endif

          if ((zalt .GT. RSData.zi) .or. (SOData.AMP .LE. 5.0)) then
            Calt  =  0.0
          endif

        case(1)        ! (  The case for the background component )
          Csurf  =  fna(SOData.AMP)
          if (zalt .LE. RSData.ZBase) then        ! (below cloud layer)
            Calt  = Csurf
          else
            Calt  = CSurf+GamaC*(zalt-RSData.ZBase)
          endif

          if (SOData.AMP .LE. 5.0)   Calt = Csurf
          if (zalt .GT. RSData.zi) Calt = Csurf

        case(2)     ! (  The mode 2 seasalt component )
          if (zalt .LE. RSData.ZBase) then
            Calt = Csurf
          else
            Calt = Csurf+GamaC*(zalt-RSData.ZBase)
          endif
```

```
            if (zalt .GT. RSData.zi) Calt = 0.0

          case(3)     ! (   The fresh , mode3,  seasalt component          )
             H3=50.0
             ! (                )
             ! (   Uses a scale height for these large droplets    )
            if (zalt .LE. RSData.ZBase) then
               Calt = Csurf*exp(-zalt/H3)
            else
               Calt = Csurf*exp(-zalt/H3)+GamaC*(zalt-RSData.ZBase)
            endif

            if (zalt .GT. RSData.zi) Calt = 0.0

         end select ! c (CASE)
         if(calt .Lt. 0.0) calt=0.0
         Calt =  ko * Calt   ! (correction for visibility)
         end  ! (PROCEDURE NOVAMW)

         SUBROUTINE Optics(    A0, A1, A2, A3,
        $                      Rh, wavln     ,
        $                      Ext, Abs      ,
        $                      AMPVal        )
c (=================================================================)
c (  Purpose: To calculate the extinction and absorption give input   )
c (           parameters A0 -> A3 , wavelength, and relative humidity )
)
c (  Called by:  MSR_WINV (Convection Case), MSR_WEAK (Weakconvection))                )
c (              MSR_SBL (Single Inversion Marine Boundary Layer)     )
c (  Calls Out:  Swell, Power10                                       )
c (  Preconditions: The tables in opmat exist                         )
c (  Global Variables:   common/opmet                                )
c (  Revision History:                                                )
c (-----------------------------------------------------------------)
c (   Date       |    Programmer     |  Remarks                      )
c (-----------------------------------------------------------------)
c (         1988 | Stuart Gathman    | Designed Code                 )
c (  02 Oct 1991 | Charles McGrath   | Comments added                )
c (     Nov 1992 | Linda Hitney      | Converted from PASCAL to FORTRAN)
c (=================================================================)

         real y7(40),y8(40)
         real f(3)
         real pi/3.14159265/
         real T0,T1,T2,T3
         real E0,E1,E2,E3
         real L
         real R
         common/opmat/T0(40),T1(4,40),T2(4,40),T3(4,40),
        $             E0(40),E1(4,40),E2(4,40),E3(4,40),L(40),R(4)

c begin  Optics ROUTINE
         do mode =  1 , 3
           call Swell(mode, f0, Rh, AMPVal)
           f(mode) = f0
         enddo


c ****************************************
! Extcalc:
!   : ------------------------------ :
!   :    Extinction/Absorption       :
!   : ------------------------------ :
c ****************************************

         j = 0
         if (Rh .EQ. r(1)) j = 1
         if (Rh .EQ. r(2)) j = 2
         if (Rh .EQ. r(3)) j = 3
         if (Rh .EQ. r(4)) j = 4
```

```
      if (j .EQ. 0) then
        jj = 0
      else
        jj = 1
      endif

c (jj=0 shows that there is not an exact humidity fit)
      if (jj .EQ. 0) then
        if  (Rh .LT. r(2)) j = 2
        if ((Rh .LT. r(3)) .and. (Rh.GT.r(2))) j = 3
        if ((Rh .LT. r(4)) .and. (Rh.GT.r(3))) j = 4
      endif

      do ii = 1 , 40   ! (calculate y7 and y8 for all wavelengths)
        if (jj .NE. 0) then  ! (ie an exact rh match)
              ! (Exact Relative humidity match)
                Y7(ii)  =  A0*power10(T0(ii))
     $                     + A1*power10(T1(j,ii))/f(1)
     $                     + A2*power10(T2(j,ii))/f(2)
     $                     + A3*power10(T3(j,ii))/f(3)

                Y8(ii)  =  A0*power10(E0(ii))
     $                     + A1*power10(E1(j,ii))/f(1)
     $                     + A2*power10(E2(j,ii))/f(2)
     $                     + A3*power10(E3(j,ii))/f(3)
        else
              ! (! INTERPOLATION FOR RELATIVE HUMIDITY)
                D6  =  R(J)-R(J-1)
                D5  =  Rh-R(J-1)
                R5  =  D5/D6
                X1  =  T1(J-1,ii)+
     $                  (T1(J,ii)-T1(J-1,ii))*R5
                X2  =  T2(J-1,ii)+
     $                  (T2(J,ii)-T2(J-1,ii))*R5
                X3  =  T3(J-1,ii)+
     $                  (T3(J,ii)-T3(J-1,ii))*R5
                B1  =  E1(J-1,ii)+
     $                  (E1(J,ii)-E1(J-1,ii))*R5
                B2  =  E2(J-1,ii)+
     $                  (E2(J,ii)-E2(J-1,ii))*R5
                B3  =  E3(J-1,ii)+
     $                  (E3(J,ii)-E3(J-1,ii))*R5
                Y7(ii)  =   A0*power10(T0(ii))
     $                      + A1*power10(X1)/f(1)
     $                      + A2*power10(X2)/f(2)
     $                      + A3*power10(X3)/f(3)

                Y8(ii)  =   A0*power10(E0(ii))
     $                      + A1*power10(B1)/f(1)
     $                      + A2*power10(B2)/f(2)
     $                      + A3*power10(B3)/f(3)
        endif

        Y7(ii)  =  0.001*Pi*Y7(ii)
        Y8(ii)  =  0.001*Pi*Y8(ii)
      enddo  ! end calculations of wavlength values of y7 and y8

c  (*************************************************************)
  ! Extabs:
  !     : ------------------------------------------------- :
  !     :            Extinction and absorption              :
  !     :            of individual wavelengths              :
  !     :                  calculation                      :
  !     : ------------------------------------------------- :
c  *************************************************************)
      i0  =  0
      if (wavln .EQ. L(40)) then
        i0 = 40
      else
        do I  =  1 ,39      ! (bracket wavelength)
          IF (wavln .EQ. L(I)) THEN
```

```
          I0  =  i  ! (wavelength is exact i0 is positive)
        ENDIF
        IF ((wavln.GT.L(I)) .and. (wavln.LE.L(I+1)))THEN
          I0 = -i
        ENDIF
      enddo
    endif

    if (I0 .GT. 0) then
      Ext  =  Y7(I0)
      Abs  =  Y8(I0)
    else
       I0  =  -I0
       A   =  Y7(I0+1)-Y7(I0)    ! (! WAVELENGTH INTERPOLATION)
       B   =  Y8(I0+1)-Y8(I0)
       C   =  L(I0+1)-L(I0)
       Ext =  Y7(I0)+A*(wavln-L(I0))/C
       Abs =  Y8(I0)+B*(wavln-L(I0))/C
    endif
    END ! OPTICS



      SUBROUTINE Depovel(Mmode, A0, Rrh, T, Ustar, Vd, Vgdry, SODats)
c (=========================================================================)
c (  Purpose: This routine calculates the deposition velocity (m/s)         )
c (  Called by:  MSR_SBL  (SimpleBLCase)                                    )
c (  Calls Out:  SWELL,Diff_coef, Mu_calc, fnr, sqrt, and Max               )
c (  Comments: This routine calculates the deposition velocity (m/s)        )
c (            based on the theory of Slinn & Slinn - A.E. [1980] p1013 and )
c (            modified by Fairall & Davidson [1986]                        )
c (       the inputs are :                                                  )
c (              A0       the diameter of the dry aerosol (m)             )
c (              Rh       the relative humidity in percent.               )
c (              Mmode    an integer (1, - 3) which describes the modes   )
c (                       of the particular component of the NAM as to    )
c (                       it's chemical nature.                           )
c (               T       air temp (C)                                    )
c (               Ustar   friction velocity (m/s)                         )
c (       The outputs are :                                                 )
c (              Vd       the deposition velocity (m/s)                   )
c (              Vgdry    the stokes fall velocity of a  dry particle.(m/s) )
c (                                                                        )
c (-------------------------------------------------------------------------)
c ( Revision History:                                                      )
c (-------------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                           )
c (-------------------------------------------------------------------------)
c (         1988  | Stuart Gathman    | Code designed                      )
c (  02 Oct 1991  | Charles McGrath   | Comments added                     )
c (      Nov 1992 | Linda Hitney      | Converted from PASCAL to FORTRAN   )
c (=========================================================================)

      real R0, F, F99, D, St, h99, Sc
      real Nu, Vgc, Vgd0, Kcp, Kdp, Kc, Kd
      real rh
      real K
      real Mu_calc
      include 'sodata.inc'

      K  = 0.4
      T0 = 296.16
      C  = 393.16
      G  = 9.8
      Dw = 1.0
      Cd = 0.0013
      Denairstp = 1.19

      rh = rrh
```

```
c    calculates the swelling factor F for the mode in question
c    use mmode to determine the specific gravity of dry particles
     call  Swell(Mmode, F, Rh, SOData.AMP)

     if (Mmode .EQ. 1) then
        R0  =  1.8
     else
        R0  =  2.165
     endif

     R0  =  R0*1000.0          !  convert specific gravity to density (MKS)
     Nu  =  Mu_calc(T+273.16)/Densirstp  !   calculates kinematic viscosity
                                         !   calculates the dynamic
                                         !   viscosity from T


     Vgdry  =  2.0*R0*G*A0*A0/(9.0*Nu)       !  stokes fall velocity
     Vgc    =  FNR(F)/R0*F*F*Vgdry
     h99    =  99.0

     call  Swell(Mmode, F99, h99, SOData.AMP) !  calcs swelling factor F99

     Vgd0   =  FNR(F99)/R0*F99*F99*Vgdry
     St     =  Vgd0*Ustar*Ustar/(Nu*G)   !  stokes number
     Kcp    =  Ustar*SQRT(Cd)/(1.0-K)
     d      =  Diff_coef(A0/2.0)   !  Brownian Diffusion Coefficient
                                   !  Toomey Fig 3.1 (the divide by two
                                   !  and, converts from dia to radius)

     Sc     =  Nu/D  ! Schmidt number

c  The following formula is from Fairall and Davidson [1986]
     Kdp    =  Ustar*SQRT(Cd)/(SQRT(Sc)*K)
     Kc     =  Kcp+Vgc
     Kd     =  Kdp+Vgd0
     Vd     =  Kc*Kd/(Kd+Kcp)
     end ! Depovel



      SUBROUTINE Wecalc( RSCalc, SOData)
c (=============================================================================)
c (  Purpose:                                                                   )
c (  Called by:  MSR_SBL (Simple Marine BL Case)                                )
c (  Calls Out:  Parcel2, We2                                                   )
c (  Preconditions:                                                             )
c (  Global variables:                                                          )
c (  Revision History:                                                          )
c (-----------------------------------------------------------------------------)
c (   Date       |    Programmer     | Remarks                                  )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath   |                                          )
c (=============================================================================)


      include 'rscalc.inc'
      include 'sodata.inc'
      Weold =  0.0
      Iloop =  0

320   continue
      call We2(RSCalc, SOData)
      if (Iloop .EQ. 0) then
        Iloop  =  1
        Weold  =  RSCalc.We
        konstant = Weold
        GOTO 320
      endif

      if (Iloop .GT. 5 ) goto 330
      if ((RSCalc.We .LT. 0.9*Weold) .OR.
     $    (RSCalc.We .GT. 1.1*Weold)) then
```

```
        Weold  =  RSCalc.We
        Iloop  =  Iloop+1
        GOTO 320
      endif
330   continue
      end  ! WEcalc


      SUBROUTINE  Bulk(U10, Th, Tdelta, Qdelta ,
     $                 Ustar, Tstar, Qstar, Tstr)
c (=============================================================================)
c (  Purpose:  Calculate Bulk temperatures and mixing ratio                     )
c (  Called by:  MOD_NPS1.SimpleBLCase                                          )
c (  Calls Out:  SUBS4.min, SUBS4.max,                                          )
c (  Preconditions:                                                             )
c (  Global variables:                                                          )
c (  Revision History:                                                          )
c (-----------------------------------------------------------------------------)
c (  Date        |    Programmer     |  Remarks                                 )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath   |                                          )
c (     Nov 1992 | Linda Hitney      |  Converted from PASCAL to FORTRAN        )
c (=============================================================================)
      real KKK
      real pi/3.14159265/
      real Mult

c const
      KKK     = 0.35
      ALPHA_T = 1.35
      ZOT     = 2.0E-05

      Z10             = 10.0   ! from GLOBALS ( altitude of 10 m )

      !  (! from NPGS lines 20435 - 20645)

      ! begin  ( procedure Bulk )
      if (U10 .EQ. 0.0) U10 =  0.1

      if (U10 .GE. 2.2) then
         Cdn =  0.789*Xtoy(U10, 0.259)/1000.0
      else
         Cdn =  Xtoy(Max(0.2, U10), -0.15)*1.08/1000.0
      endif

      Zo  =  Z10*EXP(-KKK/SQRT(Cdn))
      Ctn_sqrt  =  ALPHA_T*KKK/LOG(Z10/ZOT)
      Ss1 =  KKK*9.8*Z10/Th*(Ctn_sqrt/Cdn)
     $            *(Tdelta+0.00061*Th*Qdelta)/(U10*U10)
      Ss  =  Ss1
      So  =  Ss1

      if (So .LT. 0.0) GOTO 20360  ! (! unstable)

      if (So .LT. 1.8) GOTO 20280  ! (! stable)

      !c (!  !very stable)
      Ss   =  10.0
      Psi1 =  -47.0
      Psi2 =  -64.0
      GOTO 20470  ! (! stars)

20280 continue    ! ( !  !stable)
      Psi1 =  -Ss*4.7
      Psi2 =  -Ss*6.4
      Ss   =  So*ALPHA_T/KKK
      Ss   =  Ss*((KKK-SQRT(Cdn)*Psi1)*(KKK-SQRT(Cdn)*Psi1)
     $        /(ALPHA_T*KKK*Ctn_sqrt*Psi2))

      if (ABS(Ss-Ss1) .LT. 0.051) GOTO 20470  ! (! exit to stars)
```

```
      Ss1  =  Ss
      GOTO 20280  ! (! stble)

20360 continue  ! (  !  !unstable)
      Xx1  =  Xtoy((1.0-15.0*Ss), 0.25)
      Xx2  =  Xtoy((1.0-9.0*Ss), 0.5)

c ( ANGLES in  RADIANS)
      Psi2 =  2.0*LOG((1.0+Xx2)/2.0)
      Psi1 =  2.0*LOG((1.0+Xx1)/2.0)+LOG((1.0+Xx1*Xx1)/2.0)
     $       -2.0*ATAN(Xx1)+Pi/2.0
      Ss   =  So*ALPHA_T/KKK
      Ss   =  Ss*((KKK-SQRT(Cdn)*Psi1)*(KKK-SQRT(Cdn)*Psi1)/
     $             (ALPHA_T*KKK-Ctn_sqrt*Psi2))

      if (ABS(Ss-Ss1) .LT. 0.051) GOTO 20470  ! ( ! Exit to stars)

      Ss1  =  Ss
      GOTO 20360  ! (! unstable)

20470 continue ! (   !  !stars)
      Mult  = ALPHA_T*KKK/(LOG(Z10/ZOT)-Psi2)
      Ustar =  KKK*U10/(LOG(Z10/Zo)-Psi1)
      Tstar =  Tdelta*Mult
      Qstar =  Qdelta*Mult
      Tstr  =  Tstar+0.61*Qstar*Th/1000.0
      end  ! ( procedure Bulk )


      SUBROUTINE Whiteflux( Sr, U, Rin)
c (*=============================================================================*)
c (  Purpose:                                                                    )
c (                                                                              )
c (                                                                              )
c (  Preconditions:                                                              )
c (  Global Variables:                                                           )
c (  Revision History:                                                           )
c (------------------------------------------------------------------------------)
c (   Date        |    Programmer     | Remarks                                  )
c (------------------------------------------------------------------------------)
c (  01 Jan 1991 | Stu Gathman        |                                          )
c (  27 Sep 1991 | Charles McGrath    | Comments added                           )
c (     Nov 1992 | Linda Hitney       | Converted from PASCAL to FORTRAN         )
c (*=============================================================================*)
c (* This is a procedure based on the table in Fairall, Davidson & Schacher
c    1982, Table 3

c    Sr is the flux predicted by this procedure which would be produced
c    by whitecapping at a wind speed of U m/s and of radius Rin in microns
c    of the droplets.  This assumes a relative humidity of 80%.  This routine
c    performs a simple linear interpolation inbetween table values

c    Programmed into Turbo Pascal by S. Gathman 12/28/87
c        This procedure needs procedure linfit                                *)


      integer row, rows, I, cool, cols
      real    R0 (7)  /0.0, 0.8, 2.0, 5.0, 10.0, 15.0, 50.0/
      real    U0 (7)  /2.2, 6.0, 9.0, 11.0, 13.0, 15.0, 18.0/
      real    M  (7, 7)
      data  ((M  (i, j), j=1,7), i=1,7)
     $    /0 0, 0.0, 0.0, 0.0, 0.0,  0.0,   0.0 ,
     $     0.0, 1.3, 1.1, 2.5, 1.0,  3.0,   0.0 ,
     $     0.0, 4.5, 3.1, 4.2, 3.3   2.3,   0.0 ,
     $     0.0, 8.2, 7.7,11.0,21.0, 27.0,   0.0 ,
     $     0.0, 9.1, 9.2,17.0,49.0, 48.0,   0.0 ,
     $     0.0,11.0,10.0,19.0,72.0,140.0,   0.0 ,
     $     0.0,17.0,11.0,24.0,92.0,180.0,   0.0 /

      ! begin   ( Whiteflux )
      Sr = 0.0
```

```
      Rows = 0
      Row = 0
      I = 1

      if (u .LT.2.2) return      ! (Sr remains zero if no whitecapping exists)

100   continue
clrh  repeat
        if (u .EQ. U0(i)) then
          row = i
          rows = 1
        endif

        if (u .LT. U0(i)) row  =  i
        i = i+1
clrh  until (i .GT.6) or (row .NE. 0)
      if ((i .LE. 6) .and. (row .EQ. 0)) go to 100

      ! (Wind is sorted out at this point)
      If ((Rows .EQ. 0) .and. (row .EQ. 0)) then
        sr = 9999.0
        return
      endif

      Cool = 0
      cols = 0
      i = 1

200   continue
clrh  repeat
        if (Rin.EQ.R0(i)) then
          cool =  i
          cols = 1
        endif

        if (Rin.LT.R0(i)) cool = i
        i = i+1
clrh  until (i.GT.7) or (Cool .NE.0)
      if ((i .LE. 7) .and. (Cool .EQ. 0)) go to 200

      ! (exact fit!)
      if ((rows .EQ. 1) .and. (cols .EQ. 1)) then
        sr = M(row, cool)
        return
      endif

      ! (exact wind, fit size)
      if ((rows.EQ.1) .and. (cols.NE.1)) then
        call Linfit(Sr, Rin, R0(Cool-1), M(Row, Cool-1),
     $              R0(Cool), M(Row, Cool))
        return
      endif

      ! (exact size, fit wind)
      if ((rows.NE.1) .and. (cols.EQ.1)) then
        call Linfit(Sr, U, U0(Row-1), M(Row-1, Cool),
     $              U0(Row), M(Row, Cool))
        return
      endif

      ! (fit both size and wind)
      call Linfit(Sr1, U, U0(Row-1), M(Row-1, Cool-1),
     $            U0(Row), M(Row, Cool-1))
      call Linfit(Sr2, U, U0(Row-1), M(Row-1, Cool),
     $            U0(Row), M(Row, Cool))
      call Linfit(Sr, Rin, R0(Cool-1), Sr1, R0(Cool), Sr2)
      end  ! ( procedure White Flux )
```

```
      SUBROUTINE Linfit( Y, X, X0, Y0, X1, Y1)! real)
c {=============================================================================)
c (  Revision History:                                                         )
c (-----------------------------------------------------------------------------)
c (  Date        |    Programmer     |  Remarks                                 )
c (-----------------------------------------------------------------------------)
c (  01 Jan 1991 | Stu Gathman       |                                          )
c (  27 Sep 1991 | Charles McGrath   | Comments added                           )
c (     Nov 1992 | Linda Hitney      |  Converted from PASCAL to FORTRAN        )
c {=============================================================================)

      ! begin  ( LinFit )
      Y  =  Y0+(X-X0)*(Y1-Y0)/(X1-X0)
      end  ! ( LinFit )


      SUBROUTINE  Sky2(Zd, RDataAry, T10, Qp, Tsky)
c {=============================================================================)
c (  Purpose:  Calculate sky temperature based on calculated fluxes            )
c (  Called by:  MOD_NPS1.SimpleBLCase                                         )
c (  Calls Out:  SUBS2.xtoy                                                    )
c (  Comments:                                                                 )
c (-----------------------------------------------------------------------------)
c (  DEFINITIONS:                                                              )
c (                    NSND = NUMBER OF POINTS IN THE SOUNDING                 )
c (                   DTDZU = POTENTIAL TEMPERATURE LAPSE RATE C/M             )
c (                   DQDZU = MIXING RATIO LAPSE RATE G/KG/M                   )
c (                    ZINV = INVERSION HEIGHT,  TOP OF MIXED LAYER M          )
c (                     DTH = POTENTIAL TEMPERATURE JUMP AT INVERSION          )
c (                     DQP = MIXING RATIO JUMP AT INVERSION G/KG              )
c (                      SO = DOWNWARD FLUX AT Z C-M/S                         )
c (                     EFO = EMMISSIVITY OF MIXED LAYER                       )
c (                      SH = UPWARD FLUX AT ZB  C-M/S                         )
c (!  Rst10   =  Rdata(1,2);     !  surface potential temp from radiosonde     )
c (!  Rsqp    =  Rdata(1,3);     !   surface mixing ratio from  radiosonde     )
c (!  Zbase   =  Rdata(2,1);     !  height of base of cloud layer               )
c (!  Zb      =  Zbase;                                                        )
c (!  Tbm     =  Rdata(2,2);     !    potential temperature at cloud base -     )
c (!  Qbm     =  Rdata(2,3);     !    mixing ratio at cloud base -              )
c (!  Tunits  =  Rdata(3,1);     !                                              )
c (!  Tbp     =  Rdata(3,2);     !    potential temperature at cloud base +     )
c (!  Qbp     =  Rdata(3,3);     !    mixing ratio at cloud base +              )
c (!  Zi      =  Rdata(4,1);     !     height of the cloud top                  )
c (!  Thim    =  Rdata(4,2);     !   potential temperature at cloud top -       )
c (!  Qim     =  Rdata(4,3);     !    mixing ratio at cloud top -               )
c (!  Qunits  =  Rdata(5,1);     !                                              )
c (!  Tip     =  Rdata(5,2);     !    potential temperature at cloud top +      )
c (!  Qip     =  Rdata(5,3);     !    mixing ratio at cloud top +               )
c (                                                                            )
c (  INPUTS:                                                                   )
c (                      ZD = HEIGHT AT WHICH CALCULATION TO BE DONE           )
c (                     T10 = THE MIXED LAYER POTENTIAL TEMPERATURE            )
c (                      QP = THE MIXING RATIO IN THE MIXED LAYER (G/KG)       )
c (                                                                            )
c (  OUTPUTS :         TSKY = EFFECTIVE SKY TEMP AT HEIGHT Z                   )
c (-----------------------------------------------------------------------------)
)                                                  )
c (  Revision History:                                                         )
c (-----------------------------------------------------------------------------)
c (  Date        |    Programmer     |  Remarks                                 )
c (-----------------------------------------------------------------------------)
c (              | S. Stage, NPS     | Original Code
c (              | C. Fairall, NPS   | Modifications
c (         1988 | S. Gathman, NRL   | Adaptation to NOVAM
c (  02 Oct 1991 | Charles McGrath   | Comments added                           )
c (     Nov 1992 | Linda Hitney      | Converted from PASCAL to FORTRAN        )
c {=============================================================================-)

      real RDataAry(200,3)
      real Pot_temp(200), Spec_hum(200), Zsnd(200)
```

```
      real Emit(2)
      real Thrsnd(200), Qtsnd(200), Zz(200)
      real Nsnd
      real Nemis,index
      real Ie
      integer Keep, I, ii

      Gad  = 0.0098
      Rho0  = 1.225
      Zscale  = 10094.0
      Ssig  = 4.61E-11
      U  = 0.0
      S  = 0.0
      ii = 1

      nsnd   =  int(RDataAry(1,1))-5
      Zbase  =  RDataAry(2,1)
      Dtdzu  =  (RDataAry(2,2) - RDataAry(1,2))/Zbase
      Dqdzu  =  (RDataAry(2,3) - RDataAry(1,3) )/Zbase
      Zinv   =  Zbase
      Zb     =  Zbase
      Dth    =  (RDataAry(3,2) - RDataAry(2,2))
      Dqp    =  (RDataAry(3,3) - RDataAry(2,3))
      T10    =  (RDataAry(1,2) + RDataAry(2,2))/2.0
      Qp     =  (RDataAry(1,3) + RDataAry(2,3))/2.0
      index  =  1
      i      =  1

300   continue
clrh  repeat
         Zsnd(I)      =  RDataAry(I+5,1)
         Pot_temp(I)  =  RDataAry(I+5,2)
         Spec_hum(I)  =  RDataAry(I+5,3)
         index        =  index+1
         I            =  i+1
clrh  until index .GE. nsnd;
      if(index .lt. nsnd) go to 300

      Keep  =  0
      index =  1
      i     =  1

305   continue
clrh  repeat
         if (Zsnd(I) .LT. Zb) then
           Zz(I+Keep)     =  Zsnd(I)
           Qtsnd(I+Keep)  =  Qp
           Thrsnd(I+Keep) =  T10
         endif
         if (Zsnd(I) .LT. Zb) GOTO 19090
         if (Keep .EQ. 1) GOTO 19090
         Zz(I)     =  Zb
         Qtsnd(I)  =  Qp
         Thrsnd(I) =  T10
         Keep      =  1

19090 continue
         if ((Zsnd(I) .GE. Zb) .AND. (Zsnd(I) .LT. 3000.0)) then
           Qtsnd(I+Keep)   =  Qp+Dqp+Dqdzu*(Zsnd(I)-Zb)
           Thrsnd(I+Keep)  =  T10+Dth+Dtdzu*(Zsnd(I)-Zb)
         endif

         if (Zsnd(I) .GE. 3000.0) then
           Qtsnd(I+Keep)   =  Spec_hum(I)
           Thrsnd(I+Keep)  =  Pot_temp(I)
         endif

         if ((I .GT. 1) .AND. (Zsnd(I) .EQ. 0.0)) then
           Qtsnd(I+Keep)   =  0.0
           Thrsnd(I+Keep)  =  0.0
         endif
```

```
        if (Keep .EQ. 1) Zz(I+Keep)  =  Zsnd(I)
        index        =  index+1
        i            =  i+1

clrh  until (index .GE. Nsnd-1);
      if(index .lt. Nsnd-1) go to 305

      Nemis  =  22.0
      Check  =  0.0
      Chick  =  0.0
      S      =  0.0
      U      =  0.0  ! (added 6/17/88 so that sum starts out properly SGG)
      Icloud =  int(Nsnd)  ! (! ASSUME NO UPPER LEVEL CLOUDS)
      Nsnd   =  Nsnd-1

      if (Nsnd .LE. 0) return
      itop = 1

      do while (zz(itop) .LE. zd)
         itop = itop+1
      enddo

      Itop = Itop-1
      if (Itop .LE. 0) Itop = 1
19400 continue
      Tts      =  Thrsnd(Itop)-Gad*Zz(Itop)  ! (! line 16550 in npgs program)
      Emit(Ii) =  0.0
      index    =  itop+1
      i        =  itop+1

      do while (index .LT. (nsnd-1))
           Ii  =  3-Ii  ! ( ii = 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, ..... changes )
                        ! ( every pass )
           if (I .GT. Icloud) GOTO 19710
           if ((Zz(I) .GT. Zb) .AND. (Check .EQ. 0.0)) GOTO 19800
           if ((Zz(I) .GT. Zb) .AND. (Chick .EQ. 1.0)) GOTO 19840

           U = U+Qtsnd(I)*(5.0E-05*(Zz(I+1)-Zz(I-1))*Rho0*
     $          EXP(-Zz(I)/Zscale))
           V  =  U
           if (-Chick .EQ. 1.0) V = U0-U
           call lint779(Nemis, V, Emit(Ii), Es, Ie)
           Tt     =  Thrsnd(I)-Gad*Zz(I)
           S      =  S-Emit(Ii)*Ssig*( Xtoy((Tt+273.16), 4.0)-
     $                     Xtoy((Tts+273.16), 4.0))/100.0
           Tts    =  Tt
           index  =  index+1
           i      =  i+1
       enddo ! ( While )

 19580 continue
       tt  =  zz(int(nsnd))
       Tt  =  Thrsnd(int(Nsnd-1))-Gad*Zz(int(Nsnd-1))
       Xyz =  Rho0*EXP(-Zz(int(Nsnd-1))/Zscale)
       U   =  U+Qtsnd(int(Nsnd-1))*0.0001*(Zz(int(Nsnd-1))-
      $          Zz(int(Nsnd-1)-1))*Xyz
       V   =  U
       if (Chick .EQ. 1.0) V  =  U0+U
       call lint779(Nemis, U, Emit(Ii), Es, Ie)
       S      =  S+Ssig*Xtoy((Tt+273.16), 4.0)*Emit(Ii)/100.0
       Icloud =  int(Nsnd)
       Ftop   =  0.0
       GOTO 19730

 19710 continue
       Tt   =  Thrsnd(Icloud)-Gad*Zz(Icloud)
       Ftop =  Ssig*Xtoy((Tt+273.16), 4.0)

 19730 continue
       tsky = (ftop*(1.0-emit(ii))+S)/ssig
```

```
      if (tsky .LT. 0.0) then
         tsky = 273.14  ! (there is a failure in routine)
      else
         Tsky =  Xtoy(((Ftop*(1.0-Emit(ii))+S)/Ssig), 0.25)
      endif
      S0  =  S
      if (Zd .NE. 0.0) return      !  EXIT SKY2
      Chick  =  1.0
      U      =  0.0
      S      =  0.0
      GOTO 19400

19800 continue
      Ef0    =  Emit(ii)
      U0     =  U
      Check  =  1.0
      GOTO 19580

19840 continue
      Sh  =  S
      return
      end  ! (  Sky2 )


      SUBROUTINE  Iint779(Nxy, X, Y, Slope, Ix )

c (=================================================================)
c (  Purpose:  Interpolation routine used by procedure Sky2         )
c (  Called by:  Sky2                                               )
c (  Calls Out:  SUBS4.min, SUBS4.max                               )
c (  Preconditions:                                                 )
c (  Global variables:                                              )
c (  Revision History:                                              )
c (-----------------------------------------------------------------)
c (  Date       |    Programmer     |  Remarks                      )
c (-----------------------------------------------------------------)
c (  02 Oct 1991 | Charles McGrath   | Comments added                )
c (     Nov 1992 | Linda Hitney      |  Converted from PASCAL to FORTRAN     )
c (=================================================================)

c     begin c ( Procedure Iint779)
c (! this is an interpolating routine called by SKY2 twice  )
c (! INPUTS :  nxy, x                                       )
c (! Stored constants xarray, yarray are Tu and Temis       )
c (! outputs :  y, slope, ix                                )


      real Tu (22)
     $       / 0.0, 1.0E-05, 2.15E-05, 4.64E-05, 0.0001, 0.000215,
     $       0.000464, 0.001, 0.00215, 0.00464, 0.01, 0.0215,
     $       0.0464, 0.1,0.215,0.464,1.0, 2.15, 4.64,10.0, 21.5, 46.4/

      real Temis (23)
     $        /  0.0, 1.86, 2.58, 4.11, 5.72, 7.81, 11.4, 14.6, 18.3,
     $           23.6, 27.7, 31.9, 37.4, 41.7, 46.2, 52.9, 59, 66.6,
     $           78.8, 88.1, 95.1, 98.8, -999.0/

      real Ixmax
      real Nxy, X, Y, Slope, Ix

      Ix  =  0.0
      Ixmax  =  Max(1.0, Ix)
      Ix  =  Min(Ixmax, Nxy-1.0)

19940 continue
      if (Tu(int(Ix)).LT.X) GOTO 19980
      if (Ix .EQ. 1.0) GOTO 20020
      Ix  =  Ix-1.0
      GOTO 19940

19980 continue
```

```
      if (X.LE.Tu(int(Ix)+1)) GOTO 20020
      if (Ix+1.0 .EQ. Nxy) GOTO 20020
      Ix  =  Ix+1.0
      GOTO 19940

20020 continue
      Slope  =  (Temis(int(Ix)+1)-Temis(int(Ix)))/(Tu(int(Ix)+1)
     $          -Tu(int(Ix)))
      Y  =  Slope*(X-Tu(int(Ix)))+Temis(int(Ix))
      end  ! ( Procedure Iint779)


      SUBROUTINE We2(RSCalc, SOData)
c (=============================================================================)
c (  Purpose:                                                                   )
c (  Called by:  WeCalc                                                         )
c (  Calls Out:  SUBS4.max, SUBS4.min, SUBS4.fnK, SUBS2.xtoy                    )
c (  Preconditions:                                                             )
c (  Global variables:                                                          )
c (  Revision History:                                                          )
c (-----------------------------------------------------------------------------)
c (  Date       |    Programmer     |  Remarks                                  )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath  |                                           )
c (=============================================================================)


c (-----------------------------------------------------------------
 ! This sub program calculates entrainment velocity,  longwave
 ! cooling and average cloud temperature if cloudy,  and emissivity.
 ! routine PARCEL2 must be called previously.
 !  Inputs :
 !        ---------------------------------------------
 !        1 Zinv    Height at top of mixed layer (m)
 !        2 Zlcl  Lifted condensaction level (m)
 !        3 Dte   Equivalent temp jump at zinvnv (C or K)
 !        4 Dqw   Mixing ratio jump just below zinvnv (g/g)
 !        5 Qs    Mixing ratio just below Zinv (g/g)
 !        6 Thcent_at_z    Potential temperature just below Zinv
 !        7 Qliq    Liquit Water Mixing Ratio just below Zinv
 !        8 T10     Mixed Layer Potential Temperature (c)
 !        9 Tsfc    Sea surface Temperature (K)
 !       10 Ustar
 !       11 Tstar
 !       12 Qstar
 !       13 Tstv
 !       14 Tsky    Effective Sky Temperature (k)
 !          ---------------------------------------------
 ! OUTPUTS :   We     entrainment Velocity (m/s) updates RSCALC.WE
 !          R2DEL          cooling if cloudy (m C/s)
 !          tbar   Average Cloud Temperature (K)
 !          Emiss  Emissivity
c -----------------------------------------------------------------)

      common/globals/dte,dqw,wthe,wq,ustar,tstar,qstar,tstr,tsky
      real Wth, Wq, Wthv, M1, N1, M2, N2, M3, N3, We2tmp, Wx, Bb
      real I_S
      include 'rscalc.inc'
      include 'sodata.inc'

      ! (added InitWeData 3/8/89 since qliq wasn't known. sgg)
      call Parcel2(RSCalc, SOData, RH_in_Situ, TH_at_ZHgt,
     $            Q_V, Q_L, Z_L_CL, TH_V, T_Cent, I_S)


      Zlcl  =  Z_L_CL
      Dqws   =  dqw/1000.0
c      dqws = dqw
      Qs    =  Q_V/1000.0
      Thcent_at_z = TH_at_ZHgt
      Qliq  =  Q_L/1000.0
```

```
      T10   =  SOData.T10
      Tsfc  =  SOData.Tsea+273.16
c     Ustar =  WeData(10)
c     Tstar =  WeData(11)
c     Qstar =  WeData(12)
      Tstv  =  tstr
c     Tsky  =
      Dth   =  Dte-2460*Dqws
      Thr   =  Thcent_at_z+273.16
      Th    =  T10+273.16
      Epsilon =  0.622
      Ssigma  =  4.61E-11
      Wl    =  Qliq*1250.0
      Emiss =  -(RSCalc.Zinv-Zlcl)*0.138*Wl/2.0

      if (emiss .LT. 10.0) then
         emiss = 1.0
      else
         Emiss =  1.0-EXP(emiss)
      endif

      Emiss =  Max(0.0, Emiss)
      Tc    =  Th-0.0098*Zlcl
      Tb    =  Thr-0.0098*RSCalc.Zinv
      Tbar  =  (Tb+Tc)/2.0
      Dqsdt =  0.622*Qs*(2460.0/(0.287*Tbar*Tbar))
      Beta  =  (1.0+Th*(1.0+Epsilon)*Dqsdt)/(1.0+2460.0*Dqsdt)
      D2    =  Beta*Dte-Epsilon*Tbar*Dqws
      Dthl  =  Dth-(Thr-Th)
      D1    =  Dthl*(1.0+Epsilon*Qs)+Epsilon*Tbar*Dqws

!(!   !D1=Dthl+Epsilon*Dqw  !stage & businger ! eqn 12)
      See   =  Max(0.0, Zlcl/RSCalc.Zinv)
      See   =  Min(See, 1.0)
      Rb    =  Emiss*Ssigma*(Xtoy(Tb, 4.0)-Xtoy(Tsky, 4.0))
      Rc    =  Emiss*Ssigma*(Xtoy(Tsfc, 4.0)-Xtoy(Tc, 4.0))

      if (See .EQ. 1.0) then
        Rb  =  0.0
        Rc  =  0.0
      endif

      Wth  = -Ustar*Tstar
      Wq   = -Ustar*Qstar/1000.0
      Wthv = -Ustar*Tstv
      Wthe =  Wth+2460.0*Wq
      M1   =  (2.0-See)*See*fnH(Wthv)+
     $        (1.0-See)*(1.0-See)*fnH(Beta*Wthe-Th*Wq)
      N1   =  (2.0-See)*See*fnH(-Wthv)+
     $        (1.0-See)*(1.0-See)*fnH(-Beta*Wthe+Th*Wq)
      M2   =  See*See*(fnH(Rb)-Rc)+
     $        (1.0-See)*(1.0-See)*Beta*Rc+(1.0-See*See)*Beta*fnH(Rb)
      N2   =  See*See*fnH(-Rb)+(1.0-See*See)*Beta*fnH(-Rb)
      M3   =  (1.0-See*See)*fnH(-D2)

      N3   =  See*See*D1+(1.0-See*See)*fnH(D2)
      We2tmp  =  (0.2*(M1+M2)-(N1+N2))/(N3-0.2*M3)
      Wx   =  We2tmp

      if (We2tmp .LT. 0.0) Wx = 0.0001

      Bb =  0.5*(M1+M2+N1+N2+Wx*(M3+N3))*RSCalc.Zinv

      if (bb .GE. 0.0) then
         Wstr =  Xtoy((2.5*9.8/300.0*Bb), 0.333)
      else
         Wstr  =  0.0
      endif


      if (Wstr .LT. 0.0) Wstr  =  0.0
```

```
      !f (We2tmp .GE. 0.0) GOTO  23520

      if ((D2 .LT. 0.0) .AND. (See .LT. 1.0)) then
        We2tmp = 0.005*(1.0-See*See)
c       call UpdateLog('CLOUD INSTABILITY')
      endif

      if ((D1 .LT. 0.0).AND.((See .EQ. 1.0).AND.(Wthv .GT. 0.0))) then
        We2tmp  =  0.001
c       call UpdateLog('ENCROACHMENT')
      endif

      if (We2tmp .LT. 0.0) then
        We2tmp  =  0.0001
c       call UpdateLog('NEGATIVE CALCULATED ENTRAINMENT')
      endif

23520 continue
      R2del =  Rb-Rc
      We    =  We2tmp
      RSCalc.WE = we
        RSCalc.wstr = wstr
      end  ! ( Procedure WE2 )


      SUBROUTINE Parcel2(rscalc,sodata,Rh, Th_at_z, Qv, Ql,
     $                   Zlcl,Thv,Tcent,Is)
c ({=============================================================================})
c ( Purpose:                                                                      )
c ( Called by: We2                                                               )
c ( Calls Out:  SUBS4.max, Qvalue, TDFind                                         )
c ( Preconditions:                                                                )
c ( Global variables:                                                             )
c ( Revision History:                                                             )
c ({-----------------------------------------------------------------------------})
c ( Date       |    Programmer     | Remarks                                      )
c ({-----------------------------------------------------------------------------})
c ( 10 Oct 1991 | Charles McGrath  |                                              )
c ( 27 Aug 1993 | Stuart Gathman   | Simplify
c ({=============================================================================})

      common/globals/dte,dqw,wthe,wq,ustar,tstar,qstar,tstr,tsky

      real Is
      real Lcpgkg   / 2.46/ !from globals.pas
      include 'rscalc.inc'
      include 'sodata.inc'

      T10e =  RSCalc.t10e  ! equiv temperature of mixed layer at 10 varINIT)
      Qp   =  SOData.Qp    ! mixing ratio of mixed layer varINIT)
c     Dte  =               ! jmpt in equiv temp at inversion PROfile)
c     Dqw  =               ! jump in mixing ratio at inversion PROFILE)
      We   =  RSCalc.We    ! entrainment velocity initially .001 then from)
c     Wthe =               ! input from surface flux routine)
c     Wq   =               ! input from surface flux routine)
      Wstr =  RSCalc.Wst
      Ustr =  ustar        ! ustar from bulk routine)
      Zb   =  RSCalc.Zinv  ! height of boundary layer from varINIT)
      Z    =  500          ! height of desired calculations)
      Psfc =  1013.0       ! surface pressure from varINIT)
      Gam  =  0.0098
      Gamdw =  0.00804
      Qq  =  Qp
      Qt  =  Qp
      Zz  =  0.0
      Td10 = Td(Qq, Psfc)
      Zz  =  Z
      Thr_at_z =  T10e-Lcpgkg*Qt
      Th_at_z  =  Thr_at_z
```

```
      Tdot     =  Td10

      if (Wstr .GT. 0.0) then
        Gamque =  0.0
        Gamthe =  0.0
      else
        Wm     =  SQRT(Wstr*Wstr+6.0*Ustr*Ustr)
        Gamque =  (2.5*We*Dqw-Wq)/(0.3*Zb*Wm)
        Gamthe =  (2.5*We*Dte-Wthe)/(0.15*Zb*Wm)
      endif

      Gamdew =  0.008+(Td10+273.06)*(Td10+273.06)/5400.0/Qp*Gamque
      Gamth  =  Gamthe-2.43*Gamque
      Zs     =  Max((Thr_at_z-Tdot)/Gamdew, 0.0)
      Zt     =  Zs
      Aa     =  1.0
      Dz     =  5.0

      do While (Dz .LE. 250.0)
        Zt   =  Zt+Aa*5.0
        Qtz  =  Qt+Gamque*Zt
        Zthe =  T10e+Gamthe*Zt
        Th_at_z =  Zthe-Lcpgkg*Qtz

120     continue
           Tdo   =  Th_at_z-Gam*Zt
           Tcent =  Tdo
           Tx    =  Tdo
           Zz    =  Zt
           Pr    =  EXP(-0.00012145*Zz)*Psfc
           Qv    =  rMix_ratio(Tx, 100.0, Pr)
           Tx    =  Tdo+0.001
           Qva   =  rMix_ratio(Tx, 100.0, Pr)
           Dqdt  =  (Qva-Qv)/0.001
           Thk   =  Th_at_z+273.16
           Tk    =  Tcent+273.16
           Dthe  =  Zthe-Thk*(1.0+Lcpgkg*Qv/Tk)+273.16
           Dthe  =  Dthe/(1.0+Lcpgkg*Qv/Tk+Thk/
     $                       Tk*Lcpgkg*Dqdt)
           if (ABS(Dthe) .GE. 0.0001) then
             Th_at_z =  Th_at_z+Dthe
             Qsat    =  Qv
           endif

        if (ABS(Dthe) .GE. 0.0001) go to 120

 1 (return)
        if ((Dz .EQ. 5.0) .AND. (Qsat.LT.Qtz)) Aa  =  -1
        if ((Aa .EQ.  1.0) .AND. (Qsat.LT.Qtz)) GOTO 21590
        if ((Aa .EQ. -1.0) .AND. (Qsat.GT.Qtz)) GOTO 21590
        dz  =  dz+5.0
      enddo

21590 continue
      Zlcl =  Zt-2.5
      Qtz  =  Qt+Gamque*Z
      Zthe =  T10e+Gamthe*Z

      if (Wstr.GT.0.0) then
        Gamque  =  0.0
        Gamthe  =  0.0
      else
        Wm      =  SQRT(Wstr*Wstr+6.0*Ustr*Ustr)
        Gamque  =  (2.5*We*Dqw-Wq)/(0.3*Zb*Wm)
        Gamthe  =  (2.5*We*Dte-Wthe)/(0.15*Zb*Wm)
      endif

      Zt  =  Z
      if (Z .GE. Zlcl) GOTO 21790
      Ia  =  0.0
      Th_at_z =  Zthe-Lcpgkg*Qtz
```

```
      Tx   =  Th_at_z-Gam*Z
      Zz   =  Z
      Qvb  =  rMix_ratio(Tx, 100.0, Psfc)
      Rh   =  Qtz/Qvb*100.0
      Qq   =  Qtz
      Qv   =  Qtz
      Tdot=   Td(Qq, Psfc)
      Thd    =  Tdot+Gamdew*Z
      Tcent =  Th_at_z-Gam*Z-273.15

21760 continue
      Ql   =  Qtz-Qv
      Thv =  (Th_at_z+273.16)*(1.0+0.00061*Qv-Ql*0.001)-273.16
      GOTO 22170

21790 continue
      Is   =  1.0
      Th_at_z =  Zthe-Lcpgkg*Qtz

220   continue
         Tdo   =  Th_at_z-Gam*Zt
         Tcent =  Tdo
         Tx    =  Tdo
         Zz    =  Zt
         Pr    =  EXP(-0.00012145*Zz)*Psfc
         Qv    =  rMix_ratio(Tx, 100.0, Pr)
         Tx    =  Tdo+0.001
         Qvs   =  rMix_ratio(Tx, 100.0, Pr)
         Dqdt  =  (Qvs-Qv)/0.001
         Thk   =  Th_at_z+273.16
         Tk    =  Tcent+273.16
         Dthe  =  Zthe-Thk*(1.0+Lcpgkg*Qv/Tk)+273.16
         Dthe  =  Dthe/(1.0+Lcpgkg*Qv/Tk+Thk/Tk*Lcpgkg*Dqdt)
         if (ABS(Dthe) .GE. 0.0001) then
           Th_at_z =  Th_at_z+Dthe
           Qsat    =  Qv
         endif

      if (ABS(Dthe) .GE. 0.0001) go to 220

1 (return)
      Rh   =  100.0
      Tcent =  Th_at_z-Gam*Z
      GOTO 21760

22170 continue
      end  !  Parcel2



      block data
c (=============================================================================)
c (  File Name:  TABLES                                                        )
c (                                                                            )
c (  Purpose:  Define global constants                                         )
c (            for NOVAMSR                                                     )
c (                                                                            )
c (  Revision History:                                                         )
c (-----------------------------------------------------------------------------)
c (  Version |  Date       |    Programmer    |  Remarks                       )
c (-----------------------------------------------------------------------------)
c (  1.00    | 05 Nov 1991 | Charles McGrath  |  Made many globals local       )
c (=============================================================================)

c (-----------------------------------------------------------------------------
c THIS UNIT DEFINES THE REFRACTIVE INDE' MATRICIES USED BY THE
c NOSC PROGRAMS
c    AERO(149)(3) & WATER(169)(3)
c AND THE PRE CALCULATED OPTICAL PARAMETERS MATRICIES
c    TO, T1, T2, T3, E0, E1, E2, E3,  L AND R
c-----------------------------------------------------------------------------)
```

```
      real T0,T1,T2,T3
      real E0,E1,E2,E3
      real L
      real R
      common/opmat/T0(40),T1(4,40),T2(4,40),T3(4,40),
     $             E0(40),E1(4,40),E2(4,40),E3(4,40),L(40),R(4)


c (-------------------------------------------------------------------
c! The following data was added 7Aug86
c! Datain:
c!   : ---------------------------------------------------:
c!   :                                                     :
c!   :         Data Array Input Code                       :
c!   :  This is the latest set of data from the runs on    :
c!   :   the NTL - CRAY using 90 strips for the integration.  :
c!   :    This recalculation done in Jan 1986 by SGG.      :
c!   :    This is now coded to work with FORTRAN           :
c!   :    It now includes dust, b1 aerosol, and Seasalt     :
c!   : ---------------------------------------------------:
c-------------------------------------------------------------------)

c ( Data T0Qext  this is log10 of extinctions for dust,  no growth)

      data (T0(i),i=1,40)
     $/-2.923469, -2.959912, -2.987542, -3.173103, -3.302204, -3.616221,
     $ -4.137672, -4.703840, -4.895752, -5.076487, -5.037593, -5.163872,
     $ -5.103556, -5.217363, -5.296889, -5.327413, -5.289900, -5.102895,
     $ -5.095074, -5.084157, -5.048526, -5.195929, -4.944125, -4.894694,
     $ -4.842967, -4.831179, -4.850442, -4.968551, -5.043366, -5.077295,
     $ -5.139279, -5.190232, -5.196406, -5.246624, -5.277893, -5.235742,
     $ -4.978893, -5.186926, -4.969319, -5.060012/

c (! DATA T1QEXT(I1) RH=50%, Log10 of extinction, Gerber growth, b1 aerosol)

      data ((T1(i,j),j=1,40),i=1,4)
     $/-3.109512, -3.257849, -3.312177, -3.607813, -3.783940, -4.165230,
     $ -4.566198, -4.872668, -5.026904, -5.145560, -5.032574, -4.290340,
     $ -5.104390, -5.450016, -5.497191, -5.579665, -5.614251, -4.962693,
     $ -5.072424, -5.373218, -5.411415, -5.457934, -5.390972, -5.322320,
     $ -5.318533, -5.299478, -5.443818, -5.392556, -5.311473, -5.180292,
     $ -4.963052, -4.880975, -4.880810, -4.919121, -4.935093, -4.987036,
     $ -5.100979, -5.190460, -5.278560, -5.269282,


c (! DATA T1QEXT(I2) RH=85%, Log10 of extinction, Gerber growth, b1 aerosol )

     $ -2.881372, -3.006745, -3.054640, -3.323874, -3.487249, -3.847192,
     $ -4.234503, -4.538456, -4.709453, -4.868959, -4.779787, -3.914959,
     $ -4.732570, -5.105147, -5.157909, -5.262640, -5.348955, -4.621638,
     $ -4.728809, -5.058379, -5.157147, -5.191931, -5.166802, -5.130182,
     $ -5.122220, -5.108379, -5.146247, -5.046100, -4.941043, -4.795771,
     $ -4.572189, -4.491915, -4.493928, -4.531003, -4.557207, -4.613376,
     $ -4.733110, -4.836958, -4.947191, -4.973058,

c (! DATA T1QEXT(I3)RH=95%,  Log10 of extinction,  Gerber growth,  b1 aerosol)

     $ -2.544668, -2.624227, -2.659536, -2.876508, -3.015995, -3.334879,
     $ -3.690242, -3.978232, -4.147185, -4.327570, -4.319030, -3.445402,
     $ -4.174977, -4.519074, -4.627327, -4.745790, -4.865855, -4.164842,
     $ -4.261830, -4.592218, -4.720538, -4.756763, -4.753329, -4.739547,
     $ -4.733251, -4.724435, -4.711348, -4.594978, -4.482949, -4.334081,
     $ -4.108630, -4.027663, -4.030286, -4.066204, -4.095696, -4.153403,
     $ -4.275462, -4.384523, -4.502891, -4.542451,

c (! DATA T1QEXT(I4)RH=99%, Log10 of extinction, Gerber growth, b1 aerosol)

     $ -1.936329, -1.941536, -1.953817, -2.066715, -2.156580, -2.387460,
     $ -2.670094, -2.914032, -3.061941, -3.242225, -3.365956, -2.588397,
```

```
     $ -3.105357, -3.379822, -3.564331, -3.689817, -3.838812, -3.299625,
     $ -3.361900, -3.659516, -3.809977, -3.866429, -3.880711, -3.891503,
     $ -3.893537, -3.891807, -3.867997, -3.754512, -3.643898, -3.496727,
     $ -3.272239, -3.187722, -3.190333, -3.224186, -3.254387, -3.312034,
     $ -3.435535, -3.547922, -3.671294, -3.719149/


c (! DATA T2QEXT(I1)RH=50%, Log10 of extinction,
c                                Gerber growth, seasalt aerosol)

      data ((T2(i,j),j=1,40),i=1,4)
     $/-0.562154, -0.541000, -0.535838, -0.501248, -0.491578, -0.505651,
     $ -0.577361, -0.670094, -0.734522, -0.820333, -0.961658, -0.705446,
     $ -0.812254, -0.923323, -1.065244, -1.160931, -1.299625, -1.253249,
     $ -1.139141, -1.317539, -1.461942, -1.560225, -1.575820, -1.530075,
     $ -1.541997, -1.583959, -1.709209, -1.740167, -1.716066, -1.641780,
     $ -1.484418, -1.389148, -1.387269, -1.383766, -1.393447, -1.432997,
     $ -1.532836, -1.628341, -1.716654, -1.611100,


c (! DATA T2QEXT(I2)RH=85%, Log10 of extinction,
c          Gerber growth, seasalt aerosol)

     $ -0.180746, -0.168194, -0.161548, -0.128083, -0.113876, -0.104589,
     $ -0.144638, -0.210292, -0.262092, -0.342170, -0.500313, -0.252596,
     $ -0.294436, -0.391774, -0.529266, -0.609083, -0.728925, -0.692590,
     $ -0.601609, -0.734522, -0.864073, -0.959081, -0.989022, -1.002815,
     $ -1.027173, -1.058598, -1.161529, -1.179575, -1.144021, -1.058086,
     $ -0.883027, -0.777310, -0.776011, -0.780520, -0.793147, -0.829592,
     $ -0.930369, -1.032326, -1.147892, -1.156823,


c (! DATA T2QEXT(I3)RH=95%, Log10 of extinction,
c                        Gerber growth, seasalt aerosol)

     $  0.233022,  0.248537,  0.255490,  0.280738,  0.296358,  0.322323,
     $  0.317186,  0.283776,  0.250883,  0.190528,  0.043991,  0.227064,
     $  0.240150,  0.171375,  0.056714, -0.005753, -0.104445, -0.107049,
     $ -0.019760, -0.111894, -0.222269, -0.309795, -0.343471, -0.385008,
     $ -0.416325, -0.443107, -0.543619, -0.572969, -0.545765, -0.468100,
     $ -0.299288, -0.187568, -0.185732, -0.186699, -0.196543, -0.225060,
     $ -0.314688, -0.409771, -0.528987, -0.599376,


c (! DATA T2QEXT(I4)RH=99%, Log10 of extinction,
c                     Gerber growth, seasalt aerosol)

     $  0.904948,  0.913119,  0.915880,  0.932692,  0.942722,  0.970068,
     $  0.992841,  1.000130,  0.994599,  0.971842,  0.877343,  0.944527,
     $  0.999687,  0.976859,  0.913114,  0.878096,  0.815438,  0.763023,
     $  0.844197,  0.802733,  0.727281,  0.659441,  0.629980,  0.582097,
     $  0.550094,  0.527088,  0.425844,  0.369142,  0.370661,  0.420764,
     $  0.561781,  0.673620,  0.676547,  0.685052,  0.682037,  0.668041,
     $  0.605316,  0.533340,  0.431814,  0.329011/

c (! DATA T3QEXT(I1)RH=50%, Log10 of extinction,
c                        Gerber growth, seasalt aerosol)

      data ((T3(i,j),j=1,40),i=1,4)
     $ /2.145352,  2.149712,  2.150081,  2.157819,  2.160943,  2.172369,
     $  2.182300,  2.198024,  2.206151,  2.215479,  2.220396,  2.200577,
     $  2.224429,  2.235932,  2.247089,  2.252125,  2.251881,  2.216931,
     $  2.245858,  2.249565,  2.240799,  2.223184,  2.219349,  2.233529,
     $  2.233301,  2.223522,  2.183327,  2.135769,  2.100405,  2.072581,
     $  2.057780,  2.112337,  2.118033,  2.145880,  2.169792,  2.173914,
     $  2.157910,  2.133092,  2.094[illegible]6,  2.071551,
```

```
c (! DATA T3QEXT(12)RH=85%, Log10 of extinction,
c              Gerber growth, seasalt aerosol)

     $  2.546802,  2.550559,  2.551450,  2.557002,  2.556303,  2.568483,
     $  2.579944,  2.588787,  2.594227,  2.602700,  2.609861,  2.589771,
     $  2.609658,  2.619740,  2.633256,  2.640750,  2.646168,  2.612625,
     $  2.634054,  2.645353,  2.645560,  2.638469,  2.635293,  2.634195,
     $  2.630916,  2.624179,  2.587251,  2.536697,  2.499550,  2.473487,
     $  2.492271,  2.555602,  2.559392,  2.579292,  2.591443,  2.598692,
     $  2.594271,  2.583119,  2.557543,  2.511563,


c (! DATA T3QEXT(13)RH=95%, Log10 of extinction,
c              Gerber growth, seasalt aerosol)

     $  2.979771,  2.983924,  2.984100,  2.989356,  2.994920,  2.998338,
     $  3.007748,  3.013217,  3.017576,  3.023294,  3.029384,  3.015066,
     $  3.028815,  3.036349,  3.047586,  3.055225,  3.063446,  3.040523,
     $  3.053309,  3.065131,  3.071330,  3.072728,  3.072287,  3.070297,
     $  3.067777,  3.064308,  3.039097,  2.995561,  2.960823,  2.935981,
     $  2.958272,  3.012584,  3.015444,  3.031166,  3.039136,  3.047314,
     $  3.050457,  3.049412,  3.037825,  2.997041,


c (! DATA T3QEXT(14)RH=99%, Log10 of extinction,
c              Gerber growth, seasalt aerosol)

     $  3.653473,  3.655388,  3.656424,  3.659012,  3.659869,  3.665281,
     $  3.669586,  3.675283,  3.677698,  3.681341,  3.683875,  3.678327,
     $  3.686136,  3.690311,  3.696671,  3.701421,  3.707579,  3.699161,
     $  3.704202,  3.711293,  3.718369,  3.724243,  3.726311,  3.728670,
     $  3.729554,  3.729634,  3.723218,  3.698631,  3.672587,  3.649345,
     $  3.660752,  3.697160,  3.699092,  3.710320,  3.715769,  3.723061,
     $  3.731008,  3.738011,  3.741238,  3.723513/


c ( Data E0Abab this is log10 of absorption for dust, no growth)

      data (EU(i),i=1,40)
     $/-3.405276, -4.212023, -4.264226, -4.504553, -4.631230, -4.87988,
     $ -5.120961, -5.270301, -5.277629, -5.327127, -5.204384, -5.286820,
     $ -5.319538, -5.441112, -5.421143, -5.410464, -5.330776, -5.112107,
     $ -5.102203, -5.089360, -5.077518, -5.200322, -4.946115, -4.905005,
     $ -4.864899, -4.851829, -4.867324, -4.981424, -5.056550, -5.088320,
     $ -5.146417, -5.195315, -5.201301, -5.250844, -5.281980, -5.238621,
     $ -4.980219, -5.188432, -4.969724, -5.060256/

c (! DATA T1QABS(11) RH=50%, Log10 of absorption, Gerber growth, b1 aerosol)

      data ((E1(i,j),j=1,40),i=1,4)
     $/-7.542300, -7.634400, -7.645008, -7.626563, -7.566887, -7.281856,
     $ -6.784944, -6.176741, -6.103452, -5.786589, -5.203967, -4.332482,
     $ -5.421498, -6.048211, -5.737596, -5.773142, -5.740526, -4.979722,
     $ -5.099890, -5.417403, -5.439400, -5.481710, -5.405530, -5.335857,
     $ -5.333229, -5.312980, -5.454742, -5.399049, -5.315550, -5.182573,
     $ -4.964210, -4.881868, -4.881669, -4.920023, -4.935954, -4.987796,
     $ -5.101571, -5.190865, -5.278808, -5.269371,

c (! DATA T1QABS(12) RH=85%, Log10 of absorption, Gerber growth, b1 aerosol)

     $ -7.651715, -7.765837, -7.777154, -7.757384, -7.692205, -7.348693,
     $ -6.733463, -5.980842, -6.103909, -5.697431, -4.948577, -3.960784,
     $ -5.092422, -5.829035, -5.406636, -5.472576, -5.500547, -4.637555,
     $ -4.757856, -5.106860, -5.192891, -5.218668, -5.186319, -5.146034,
     $ -5.137124, -5.121685, -5.154455, -5.050210, -4.943438, -4.797158,
     $ -4.573147, -4.492914, -4.494904, -4.531993, -4.558164, -4.614269,
     $ -4.733815, -4.837465, -4.947537, -4.973181,
```

```
c (! DATA T1QABS(I3) RH=95%, Log10 of absorption, Gerber growth, b1 aerosol)

     $ -7.607250, -7.769091, -7.781123, -7.757856, -7.686234, -7.212044,
     $ -6.437814, -5.580540, -5.868285, -5.372983, -4.529090, -3.507044,
     $ -4.633166, -5.418893, -4.953739, -5.032096, -5.087661, -4.187321,
     $ -4.305105, -4.665164, -4.778899, -4.798521, -4.786695, -4.765103,
     $ -4.755377, -4.743884, -4.721887, -4.599911, -4.485798, -4.335828,
     $ -4.110133, -4.029384, -4.032008, -4.067917, -4.097372, -4.155001,
     $ -4.276717, -4.385440, -4.503472, -4.542663,

c (! DAYA T1QABS(I4)RH=99%, Log10 of absorption, Gerber growth, b1 aerosol)

     $ -7.263078, -7.589290, -7.619734, -7.595252, -7.499311, -6.627106,
     $ -5.663140, -4.738333, -5.147172, -4.582545, -3.698927, -2.691500,
     $ -3.757161, -4.569554, -4.094820, -4.181167, -4.253755, -3.349294,
     $ -3.457411, -3.820333, -3.949698, -3.969117, -3.968147, -3.958370,
     $ -3.949698, -3.941119, -3.893877, -3.766420, -3.650859, -3.501207,
     $ -3.276528, -3.192946, -3.195581, -3.229487, -3.259653, -3.317187,
     $ -3.439770, -3.551139, -3.673439, -3.719968/

c (! DATA T2QABS(I1)RH=50%, Log10 of absorption,
c                  Gerber growth, Seasalt aerosol)

      data ((E2(i,j),j=1,40),i=1,4)
     $/-3.196659, -4.978563, -5.772113, -7.459721, -6.504608, -3.589729,
     $ -3.199916, -2.724781, -2.829680, -2.455374, -1.814656, -0.994305,
     $ -1.860373, -2.607778, -2.200653, -2.274244, -2.344228, -1.541438,
     $ -1.576574, -1.950472, -2.075421, -2.074688, -2.048085, -2.005947,
     $ -2.014354, -2.023563, -2.043279, -1.950007, -1.855831, -1.729717,
     $ -1.532377, -1.437279, -1.436590, -1.438600, -1.457922, -1.499242,
     $ -1.593187, -1.678775, -1.753723, -1.626059,

c (! DATA T2QABS(I2)RH=85%,  Log10 of absorption,
c             Gerber growth, Seasalt aerosol)

     $ -3.090813, -4.921326, -5.659953, -7.151398, -6.136635, -3.431118,
     $ -2.910519, -2.199978, -2.478993, -2.006392, -1.258462, -0.508386,
     $ -1.244034, -2.011383, -1.576197, -1.657795, -1.738690, -0.953778,
     $ -0.996712, -1.328892, -1.460297, -1.479530, -1.474398, -1.458321,
     $ -1.459170, -1.460573, -1.445584, -1.345708, -1.250503, -1.125628,
     $ -0.933898, -0.837555, -0.837256, -0.847161, -0.865345, -0.905145,
     $ -1.003589, -1.098591, -1.202255, -1.182441,

c (! DATA T2QABS(I3)RH=95%, Log10 of absorption,
c                Gerber growth, Seasalt aerosol)

     $ -2.989658, -4.785633, -5.401100, -6.710545, -5.642313, -3.204621,
     $ -2.467437, -1.622366, -1.994262, -1.461992, -0.686723, -0.032705,
     $ -0.647991, -1.401472, -0.963810, -1.044389, -1.126697, -0.385177,
     $ -0.421361, -0.720881, -0.849182, -0.874194, -0.876606, -0.872215,
     $ -0.871278, -0.870117, -0.846582, -0.751144, -0.661623, -0.545399,
     $ -0.367280, -0.270179, -0.269347, -0.276257, -0.290467, -0.323636,
     $ -0.413154, -0.503140, -0.611171, -0.644089,

c (! DATA T2QABS(I4)RH=99%, Log10 of absorption,
c                Gerber growth, Seasalt aerosol)

     $ -2.638763, -4.393844, -4.905599, -5.768658, -4.775933, -2.611970,
     $ -1.658763, -0.721544, -1.135655, -0.573879,  0.177854,  0.668563,
     $  0.223028, -0.480172, -0.054477, -0.128106, -0.205909,  0.446941,
     $  0.425714,  0.176439,  0.060622,  0.035630,  0.030478,  0.028775,
     $  0.029587,  0.031368,  0.053463,  0.135260,  0.211094,  0.307988,
     $  0.456032,  0.547947,  0.549714,  0.551206,  0.543956,  0.523720,
     $  0.457942,  0.387817,  0.294576,  0.240225/
```

```
c (I DATA T3QABS(I1)RH=50%, Log10 of absorption,
c         Gerber growth, Seasalt aerosol)

      data ((E3(i,j),j=1,40),i=1,4)
     $/ 0.362765, -1.481091, -2.168143, -3.866525, -2.868061,  0.013596,
     $  0.426039,  0.894322,  0.823031,  1.164353,  1.646051,  1.894338,
     $  1.616497,  1.063108,  1.421801,  1.377397,  1.331042,  1.821546,
     $  1.823474,  1.632569,  1.557640,  1.558252,  1.579486,  1.622784,
     $  1.621374,  1.612498,  1.591376,  1.638419,  1.684585,  1.743620,
     $  1.822273,  1.881755,  1.885333,  1.904071,  1.914703,  1.913077,
     $  1.892696,  1.869654,  1.842990,  1.894576,

c (I DATA T3QABS(I2)RH=85%,  Log10 of absorption,
c               Gerber growth, Seasalt aerosol)

     $  0.497096, -1.340274, -2.058066, -3.553587, -2.501593,  0.187887,
     $  0.717704,  1.407951,  1.163519,  1.601484,  2.127623,  2.281647,
     $  2.121494,  1.612540,  1.967225,  1.922845,  1.874563,  2.270143,
     $  2.268859,  2.142984,  2.079362,  2.070924,  2.076240,  2.090223,
     $  2.091280,  2.089658,  2.091421,  2.132932,  2.169145,  2.211921,
     $  2.270306,  2.320541,  2.323108,  2.336380,  2.342561,  2.345060,
     $  2.336099,  2.323046,  2.299551,  2.306961,

c (I DATA T3QABS(I3)RH=95%,  Log10 of absorption,
c                    Gerber growth, Seasalt aerosol)

     $  0.646286, -1.165198, -1.774200, -3.101939, -2.022990,  0.410760,
     $  1.159116,  1.955976,  1.630275,  2.117603,  2.612318,  2.699664,
     $  2.606790,  2.162116,  2.488635,  2.451280,  2.409324,  2.719107,
     $  2.717912,  2.634991,  2.586700,  2.579429,  2.580457,  2.585145,
     $  2.586644,  2.587172,  2.594426,  2.626453,  2.652140,  2.68[illegible],
     $  2.719248,  2.758473,  2.760566,  2.772256,  2.777652,  2.78[illegible],
     $  2.783925,  2.781202,  2.768912,  2.765445,

c (I DATA T3QABS(I4)RH=99%,  Log10 of absorption,
c          Gerber growth, Seasalt aerosol)

     $  0.870942, -0.775389, -1.289282, -2.281415, -1.181517,  0.960680,
     $  1.930231,  2.783968,  2.430543,  2.920468,  3.328420,  3.352491,
     $  3.317374,  2.975119,  3.246055,  3.219191,  3.189378,  3.390246,
     $  3.388190,  3.349355,  3.322798,  3.320416,  3.321080,  3.323809,
     $  3.325782,  3.327257,  3.335217,  3.354205,  3.366684,  3.377834,
     $  3.392662,  3.415608,  3.417106,  3.426446,  3.431316,  3.438210,
     $  3.447855,  3.455637,  3.458985,  3.462727/

c (-------------------------------------------------------------------


c (**** Wavelengths of these calculations in micro meters ********)
      data L
     $         / 0.2,    0.3,    0.3371,  0.55,  0.6943,
     $           1.06,   1.536,  2.0,     2.25,  2.5,
     $           2.7,    3.0,    3.3923,  3.75,  4.5,
     $           5.0,    5.5,    6.0,     6.2,   6.5,
     $           7.2,    7.9,    8.2,     8.7,   9.0,
     $           9.2,   10.0,   10.591,  11.0,  11.5,
     $          12.5,   14.8,   15.0,    16.4,  17.2,
     $          18.5,   21.3,   25.0,    30.0,  40.0 /

c (**** Relative humidity of the calculations *********)
      data R /50, 85, 95, 99/
      end
```

```
      SUBROUTINE Swell( mmode , f, rh, AMPVal)
c (=============================================================)
c (  Purpose: To calculate the growth of hygroscopic aerosol wrt 80%   )
c (  Called by:  Depovel                                               )
c (  Calls Out:  xtoy                                                  )
c (  Revision History:                                                 )
c (------------------------------------------------------------------)
c (   Date        |    Programmer     |  Remarks                       )
c (------------------------------------------------------------------)
c (         1978. | J. Fitzgerald     | Original concepts reported in  )
c (               |                   | the literature, Fitzgerald 1978) )
c (         1983  | S. Gathman        | Fitzgerald's formulation in NAM  )
c (         1986  | M. Gerber         | Expanded Humidity Parameters for )
c (               |                   | NAM,  Gerber(1985)             )
c (     Jun 1989  | S. Gathman        | Used Gerber's formula in NOVAM   )
c (  02 Oct 1991  | Charles McGrath   | Comments added                 )
c (     Nov 1992  | Linda Hitney      | Converted from PASCAL to FORTRAN )
c (     Sep 1993  | S. Gathman        | Integrated into NOVAMSR        )
c (=============================================================)
c (*************************************************************
c (Gerber's formula)
      select case (mmode)
        case (2)
          c7 = 1.83
          c8 = 5.13
        case (3)
          c7 = 1.97
          c8 = 5.83
        case (1)
          if (AMPVal .GT. 5.0) then
            c7 = 1.28
            c8 = 2.41
          else
            c7 = 1.17
            c8 = 1.87
          endif
      end select ! (case)
      a =  c7 - rh/100.0
      b =  c8*(1.0-rh/100.0)
      f =  a/b
      b = 1.0/3.0
      f = xtoy(f, b)
      end   ! (Swell)


      SUBROUTINE sufin(SOData, SurfObsFileName)
c (=============================================================)
c (  Purpose: Read surface observation file & assign to SOData or is   )
c (           dummy value read from surface obs file, then assign default )
c (  Preconditions:  SurfaceObsFileName exists & is in proper format   )
c (  Global Variables:  none                                           )
c (  Called by:  NOVAMSR                                               )
c (------------------------------------------------------------------)
c (  01 Jan 1991  | Stu Gathman       | modified                       )
c (  27 Sep 1991  | Charles McGrath   | Comments added                 )
c (     Nov 1992  | Linda Hitney      | converted from PASCAL to FORTRAN )
c (     Sep 1993  | S. Gathman        | Integrated into NOVAMSR        )
c (=============================================================)
c (-------------------------------------------------------------)
 !  This program is used to input all of the surface observations and to  )
 !  substitute the default values when the value of -99 for temperature   )
 !  or -1 for other surface observations is found in                      )
 !  the surface observation file at a particular location.                )
 !                                                                        )
 ! A. Surface Observation file (9 parameters, use "-1" if the parameter   )
 !    is not measured, or if temperature, use -99 if not measured)       )
 !       1. Tsea   =  Sea surface temperature (C)                         )
 !       2. T10    =  Air temperature (C) @ 10 meters                     )
 !       3. Rh     =  Relative Humidity (%) @10 meters; used to           )
 !                    calulate:  Qp =  Mixing ratio (g/kg) @10 meters     )
```

```
!       4. Svis   =  Optical Visibility (kilometers)                    )
!       5. U10    =  Current real wind speed (m/s) @10 meters           )
!       6. Uave   =  Averaged wind speed (24hr) (m/s)                   )
!       7. Amp    =  Air Mass Parameter (1..10)                         )
!                    (note:  amp & svis are inputs to the NAAWS model   )
!                            to get new amp.)                           )
!       8. Irext  =  Surface IR Ext. (1/km) (@10.6 microns)             )
!       9. I      =  Zonal category (1..6)                              )
c  ------------------------------------------------------------------------)
c (  Default surface Met. data is:                                        )
c (      Tropical is square #49,                                          )
c (      midlatitude is ship station #e,                                  )
c (      sub artic is ship station #i .                                   )
c (  Winter is January and Summer is July.                                )
c (  Data from U.S. Navy Marine Climatic Atlas of the World               )
c (  (revised) Vol 1, North Atlantic Ocean,  by J.M. Neserve (1974).      )
c  ------------------------------------------------------------------------)

      real dtsea(6)  /26.5, 26.5, 24.8, 19, 12.2, 9.2/
      real dwind(6)  /5.47, 4.16, 5.26, 10.58, 8.05, 12.47/
      character*12 SurfObsFileName
      include 'sodata.inc'
      real surfops(9)

      real a/ -4.05801664  /
      real b/ -0.00890166  /
      real c/  0.038432675 /
      real d/ -0.64465936  /
      real e/ -0.00899986  /
      real g/  0.007232437 /
      real l/  2.019394568 /
      real m/ -0.00900429  /
      real n/ -0.016770367 /
      real pi/ 3.14159265  /

      ! (---- initialize mixing ratio to flagged value ----)
      SOData.QP = -99.0

      ! (----  Read in surface observation file to list surfops() ----)
      open(unit=35,file=SurfObsFileName,status='old')
      read (35,*) surfops
      close(35)

c (----  SOData.ZoneIndex becomes now the index of zonal catagory ----)
c (----  checking the range of SOData.ZoneIndex for validity ----)
      SOData.ZoneIndex =  int(surfops(9))

      if ((SOData.ZoneIndex .LT. 1).OR.(SOData.ZoneIndex .GT. 6)) then
        SOData.ZoneIndex = 3
      ENDIF

c (-----  filling in the default values if necessary -----)
c (SEA SURFACE TEMPERATURE acceptable range of -5 to +35 deg C )

      if ((surfops(1) .LT. -5.0) .OR. (surfops(1) .GT. 35.0))  then
         SOData.TSea = dtsea(SOData.ZoneIndex)
      else
        SOData.TSea =  surfops(1)
      endif

c (AIR TEMPERATURE (C) acceptable range of -50 to +40 deg C )
      if ((surfops(2) .LT. -50.0) .OR. (surfops(2) .GT. 40.0)) then
        ic ( set flag to check radiosonde data for a usable temperature )
        SOData.T10 = -99.0
        SOData.T10Meas = .false.
      else
        SOData.T10 = surfops(2)
        SOData.T10Meas = .true.
      endif

c (RELATIVE HUMIDITY acceptable range of 0 to 100 percent )
```

```
      if ((surfops(3) .LT. 0.0) .OR. (surfops(3) .GT.100.0)) then
        SOData.RHH = -99.0      ! (flagged for radiosonde calculation in  )
        SOData.RHHMeas = .false. ! (AssignRSondeVars procedure in this unit)
      else  ! ( sgg fix 11/26/90 )
        SOData.RHH = surfops(3)   ! (Use this RH value directly)
        SOData.RHHMeas = .true.
        if (SOData.T10Meas) then
          vp0 = vappr(SOData.T10)
          vp0 = vp0 * surfops(3) / 100.0
          SOData.Qp = 623.0 * vp0 / (1013.0 - vp0)
        endif
      endif

c (CURRENT WIND SPEED (M/S) acceptable range of 0 to 40 m/s)
      if ((surfops(5) .LT. 0.0) .OR. (surfops(5) .GT. 40.0)) then
        ! ( default values:  5.47, 4.16, 5.26, 10.58, 8.05, 12.47 m/s )
        SOData.U10 = dwind(SOData.ZoneIndex)
        SOData.U10Meas = .false.
      else
        SOData.U10 = surfops(5)
        SOData.U10Meas = .true.
      endif

c (AVERAGE WIND SPEED (M/S) acceptable range of 0 to 40 m/s)
      if ((surfops(6) .LT. 0.0) .OR. (surfops(6) .GT. 40.0), then
        SOData.UAve = SOData.U10
      else
        SOData.UAve = surfops(6)
      endif

c (OPTICAL VISIBILITY (KM))
      if (surfops(4) .LE. 0.5) then
       Qfac = Qfac - 2.0
        SOData.SVis = 10.0
        SOData.SVisMeas = .false.
      else
        SOData.SVis = surfops(4)
        SOData.SVisMeas = .true.
      endif

      if (surfops(4) .GT. 500.0) then
       Qfac = Qfac - 2.0
        SOData.SVis = 500.0
        SOData.SVisMeas = .false.
      else
        SOData.SVis = surfops(4)
        SOData.SVisMeas = .true.
      endif

c (AIR MASS PARAMETER acceptable range of 1 to 50 )

c  This is a technique to do the air mass parameter logic
c  and the method of using the visibility if it is available
c  to determine the air mass parameter.

      IF(SURFOPS(7) .GT. 0.0) GOTO 100
      IF( .NOT. SOData.U10Meas ) GOTO 200
      IF( .NOT. SOData.SVISMeas) GOTO 200

c     determine the w.m.p. from the visibility and wind data

      f  = ((2-SOData.RHH/100)/(6*(1-SoData.RHH/100)))**(1/3)

      betavis0=(3.912/SOData.SVis)-0.01162

      t1 = (a+c*SOData.RHH)/(1+b*SOData.RHH)
      t2 = (d+g*SOData.RHH)/(1+e*SOData.RHH)
      t3 = (l+n*SOData.RHH)/(1+m*SOData.RHH)

      beta2=pi*FNB(SOData.Uave)*t2 / (1000 * f)
      beta3=pi*FNC(SOData.U10)*t3  / (1000 * f)
```

```
      if(betavis0 .le. (20.0 *T1 +beta2+beta3)) then
       amp = 0.1
      else
       amp = SQRT((betavis0-(beta2+beta3))/(2*pi*t1/f))
      endif

      SOData.Amp = amp
      goto  500

  100 SOData.Amp = surfops(7)
      GOTO 500
  200 SOData.Amp = 2.0
  500 continue

c (SURFACE IR EXTINCTION)
      if ((surfops(8) .LT. 0.001) .OR. (surfops(8) .GT. 100.0)) then
        Qfac = Qfac - 2.0
        SOData.IRExt = -1.0
        SOData.IRExtMeas = .false.
      else
        SOData.IRExt = surfops(8)
        SOData.IRExtMeas = .true.
      endif

      end ! ( sufin )


      SUBROUTINE CheckRSondeData(RdataAry)
c (=============================================================================)
c (  Purpose:  Check range of potential temp & mixing ratio                    )
c (-----------------------------------------------------------------------------)
c (  01 Oct 1992 | Stu Gathman      |                                          )
c (     Nov 1992 | Linda Hitney     | converted from PASCAL to FORTRAN         )
c (     Sep 1993 | S. Gathman       | Integrated into NOVAMSR                  )
c (=============================================================================)

      real alt, pt, mr, rh, airt, pres
      real RdataAry (200,3)

      ! begin  ( CheckRSondeData )
      do i=6 , int(RdataAry(1,1))
        if((i.GT.6) .and.
     $     ((RdataAry(i,2).GT.50.0).or.(RdataAry(i,2).LT.-30.0))) then
          RdataAry(i,1)=RdataAry(i-1,1)
          RdataAry(i,2)=RdataAry(i-1,2)
          RdataAry(i,3)=RdataAry(i-1,3)
        endif
        alt=RdataAry(i,1)
        pt =RdataAry(i,2)
        mr =RdataAry(i,3)
        call calc_rh_atemp_press ( alt, pt, mr, rh, airt, pres )
        if (rh .GT. 99.0) then
          call Convert_Rdata ( pres, airt, 99.0, alt, pt,mr)
          RdataAry(i,3)=mr
        endif
      enddo
      end ! ( CheckRSondeData )

      SUBROUTINE calc_rh_atemp_press( alt, pt, mr , rh, airt, pres )
c (=============================================================================)
c (  Purpose:   This is a program to convert the altitude,                     )
c (             Potential Temperature and mixing ratio data into the           )
c (             pressure, air temperature and relative humidity                )
c (             profile data.                                                  )
c (-----------------------------------------------------------------------------)
c (  01 Oct 1992 | Gathman & McGrath | designed program for NOVAM215           )
)
c (     Nov 1992 | Linda Hitney      | converted from PASCAL to FORTRAN       ) )
c (     Sep 199.3 | S. Gathman       | Integrated into NOVAMSR                ) )
c (=============================================================================)
```

```
      real alt,pt,mr
      real rh,airt,pres

      if (alt.LT.500.0) then
        pres=1013.0-55.0*alt/500.0
      else
        pres=1021.38*exp(-0.00012739*alt)
      endif

      airt=(pt+273.15)/exp(0.286*log(1000.0/pres))-273.15

      satvp= vappr(airt)
      satmr=620.0*satvp/(pres- satvp)
      rh=100.0*mr/satmr
      end  !calc_rh_atemp_press


      SUBROUTINE Convert_Rdata (pres,at,rh, alt, pt,mr)
c (=============================================================================)
c (  Purpose:  This is a program which takes as input the pressure,            )
c (            airtemp and rh data and gives back alt, potential temperature)
c (            and mixing ratio at that level.                                )
c (                                                                           )
c (-----------------------------------------------------------------------------)
c (-----------------------------------------------------------------------------)
c (     Sep 1992 | S.Gathman          | designed program for NOVAM215          )
c )                                                                           ) )
c (     Nov 1992 | Linda Hitney       | converted from PASCAL to FORTRAN       ) )
c (     Sep 1993 | S.Gathman          | Integrated into NOVAMSR                ) )
c (=============================================================================)

      real alt, pt, mr, rh, at, pres

      if (pres.GT.958.0) then !c (see formula e5,f5 and g5 in Qpro prog2.wq1)
        alt=(1013.0-pres)*500.0/55.0
      else
        alt=-log(pres/1021.38)/0.00012739
      endif

      pt=(at+273.15)*xtoy(1000.0/pres,0.286) -273.15

      mr=rh*620.0*vappr(at)/(100.0*(pres-vappr(at)))

      END    !Convert_Rdata


      SUBROUTINE MAKE_RDATAARY(preamble,rdataary)
c (=============================================================================)
c (  Purpose:  Purpose to put preamble in front of radiosonde data in the     )
c (            data array called rdataary which will be used by NOVAM.        )
c (-----------------------------------------------------------------------------)
c (-----------------------------------------------------------------------------)
c (     Sep 1992 | S.Gathman          | designed program for NOVAMSR           )
c )                                                                           ) )
c (     Sep 1993 | S.Gathman          | Integrated into NOVAMSR                ) )
c (=============================================================================)

      real preamble(5,3),rdataary(200,3)

      do i=1,5
         rdataary(i,1)=preamble(i,1)
         rdataary(i,2)=preamble(i,2)
         rdataary(i,3)=preamble(i,3)
      enddo

        if (int(RdataAry(1,1)) .GT. 200) stop
     $     'more than 200 records in rsonde'

        open(unit=13,file='sigfile',status='old')
        read(13,*) j
      do I = 1 , j
        Read(13,*) a,b,c
```

```
c       a is pressure, b is air temperature, and c is relative humidity.
c     we must translate these back to the NOVAM values of altitude, pt and mr.

        RdataAry(I+5,1)=altitude(a)
        RdataAry(I+5,2)=Potential_temperature(a,b)
        RdataAry(I+5,3)=rMixing_ratio(c,b,a)
       enddo
        close(13)

c (  RdataAry(1,1) is number of rows in the radiosonde matrix )
c (  note that RdataAry(1, 1) is five more than number actual )
c (  radiosonde observations, unless = 1, then no rsond data. )

      end  ! (MAKE_RDATAARY)


      SUBROUTINE AssignT10_RHH_QP( RSondData, SOData)
c (=============================================================================)
c (  Purpose: checks  surface temp, rel humidity, and mixing ratio values   )
c (           in surf obs file for being measured, uses Rsonde if possible  )
c (  Called by:  NOVAM215.FOR                                                )
c (  Calls out:  (none)                                                      )
c (  Preconditions:                                                          )
c (  Global Variables:                                                       )
c (-----------------------------------------------------------------------)
c (  26 Feb 1992 | Charles McGrath  | created                                )
c (     Nov 1992 | Linda Hitney     | converted from PASCAL to FORTRAN       )
c (     Sep 1993 | S. Gathman       | integrated into NOVAMSR                )
c (=============================================================================)

      real RSondData (200,3)
      real dt10 (6) /26.3, 26.3, 24.2, 17.5, 9.5, 7.0/
      include 'sodata.inc'
      Psfc            = 1013.01  ( surface pressure in millibars )
c ( Surface Air Temperature -- if flagged as not already measured, then if )
c ( radiosonde data exists at below 100 meter altitude, then use the       )
c ( radiosonde temperature, otherwise use table value for geographic zone. )
      if (.NOT. SOData.T10Meas) then
        if ((RSondData(1,1) .GT. 5.0).AND.
     $      (RSondData(6,1) .LE. 100.0)) then
          SOData.T10 = RSondData(6,2) !  c ( assign temp at lowest height )
        else
          SOData.T10 = dt10(SOData.ZoneIndex)
        endif
      endif
c ( Surface Relative Humidity -- if flagged as not measured and    )
c ( there is radiosonde data from which to derive humidity data    )
c ( if radiosonde data exists at an altitude below 100 meters      )
      if (.NOT. SOData.RHHMeas) then
        if ((RSondData(1,1) .GT. 5.0).AND.
     $      (RSondData(6,1) .LE. 100.0)) then
          !c ( RelHum = ratio of measured mixing ratio divided by saturation )
          !c (    mixing ratio.  Measured mixing ratio, QP in g/kg is from   )
          !c (    rsonde(1,3) and saturation mixing ratio is calculated from)
          !c (    vapor pressure at surface temperature RSonde(1,2) value    )
          !c (    The saturation mixing ratio formula gives results in g/g,  )
          !c (    so needs to be multiplied by 1000 for g/kg units conversion)
          !c (    SatQP = .622*SurfVapPr/SurfAirPr-SurfVapPr                 )
          !c ( RelHum = QP*((SurfPr-VaporPr)/(0.622*VaporPr))/1000*100%       )
          VP0 = VapPr(RSondData(1,2))
          SOData.RHH = RSondData(1,3)*(Psfc-VP0)/(0.622*VP0)/1000.0*100.0
      ! ( QP set to mixing ratio in radiosonde mess at lowest altitude )
          SOData.Qp  = RSondData(1,3)
        else
          SOData.RHH = 80.0
          vp0 = 0.8 * vappr(SOData.T10) !c (Vapor Press assumes RH of 80%)
          SOData.Qp = 623.0 * vp0 / (1000.0 - vp0) ! c (sgg fix 1/18/91)
        endif
      endif
      end   ! (  AssignT10_RHH_QP )
```

```
      SUBROUTINE AssignRSondeVars(RSondData, RSData, RSCalc, SOData)
c {=============================================================================}
c (  Purpose: Assigns radiosonde parameters from array to RSData & RSCalc      )
c (  Called by:  NOVAMSRD.FOR                                                  )
c (  Global Variables: RSData & RSCalc                                         )
c (-----------------------------------------------------------------------------)
c (  10 Oct 1991 | Charles McGrath  | created from sufin procedure & unovam)
c (     Nov 1992 | Linda Hitney     | converted from PASCAL to FORTRAN     )
c (     Sep 1993 | S. Gathman       | integrated into NOVAMSR              )
c {=============================================================================}
c (                      Rst10 is surface potential temp from radiosonde   )
c (                              Rsqp  is surface mixing ratio from  radiosonde    )
c (                              Zbase & Zb are height of base of cloud layer      )
c (                              Tmb   is potential temperature at cloud base    - )
c (                              Qmb   is mixing ratio at cloud base             - )
c (                              Tunits should be 1 if potential temperature C.    )
c (                              Tbp   is potential temperature at cloud base    + )
c (                              Qbp   is mixing ratio at cloud base             + )
c (                              Zi    is height of the cloud top                  )
c (                              Thim  is potential temperature at cloud top     - )
c (                              Qim   is mixing ratio at cloud top              - )
c (                              Qunits should be 1                                )
c (                              Tip   is potential temperature at cloud top     + )
c (                              Qip   is mixing ratio at cloud top              + )
c (-----------------------------------------------------------------------------)

      real RSondData (200,3)
      real Lcpgkg,Psfc

      include 'rscalc.inc'
      include 'rsdata.inc'
      include 'sodata.inc'

      RSData.Rst10  = RSondData(1,2)
      RSData.Rsqp   = RSondData(1,3)
      RSData.Zbase  = RSondData(2,1)
      RSData.Tbm    = RSondData(2,2)
      RSData.Qbm    = RSondData(2,3)
      RSData.Tunits = RSondData(3,1)
      RSData.Tbp    = RSondData(3,2)
      RSData.Qbp    = RSondData(3,3)
      RSData.Zi     = RSondData(4,1)
      RSData.Thim   = RSondData(4,2)
      RSData.Qim    = RSondData(4,3)
      RSData.Qunits = RSondData(5,1)
      RSData.Tip    = RSondData(5,2)
      RSData.Qip    = RSondData(5,3)

c (--- define met parameters needed by models in terms of inputs. ---)
      RSCalc.Zinv     = RSData.zbase
      RSCalc.Thab     = SOData.T10+274.15
      Tave            = RSCalc.Thab-0.0049*RSCalc.Zinv
      RSCalc.Fthet    = RSCalc.Thab/Tave
      RSCalc.Tdelta   = SOData.T10-SOData.TSea

c (-------------------------------------------------------)
c ( ! This is an approximation to the                     )
c ( ! goff-gatch formula which is good                    )
c ( ! to 1/2 % error for - to over 25 deg.                )
c ( ! List (1968).                                        )
c (-------------------------------------------------------)
      Psfc            = 1013.0  !( surface pressure in millibars )
      Lcpgkg          = 2.46
      Qsfc  =  6.112*EXP(17.67*SOData.TSea/(SOData.TSea+243.5))
      Qsfc  =  622.0*Qsfc/(Psfc-Qsfc)
      RSCalc.Qdelta = SOData.Qp-Qsfc
      RSCalc.T10e   = SOData.T10+Lcpgkg*SOData.Qp
      RSCalc.We     = 1.0
      RSCalc.Wstr   = 0.001
      end  ! (procedure AssignRSondeVars)
```

```
      SUBROUTINE preamb(infsof,preamble)
c (=============================================================================)
c (  Purpose:  This is a program which takes as input the "filtered"            )
c (          signficant levels from the processed data and produces an         )
c (          acceptable preamble for the NOVAM input data.                     )
c (                                                                            )
c (  Called by:  NOVAMSRD.FOR                                                  )
c (----------------------------------------------------------------------------)
c (      Sep 1992 | S. Gathman        | Logic designed                         )
c (      Sep 1993 | S. Gathman        | integrated into NOVAMSR                )
c (=============================================================================)

      integer flag
      integer input,sof,novam
      real gradient(200),alt(200),pt(200),mr(200),pres(200),airt(200)
      real preamble(5,3)
      real rh(200)
      real rMixing_ratio,temperr
      character*12 infile,infsof

      temperr=0.19
      input=14
      sof=16
      novam=17
      infile='sigfile'
      open(unit=input,file=infile,status='old')
      open(unit=sof,file=infsof,status='old')

      read(input,*) num

      j=1
      read(input,*) p,at,h
      pres(j)=p
      airt(j)=at
      rh(j)  =h
      alt(j)=Altitude(p)
      pt(j)=Potential_temperature(p,at)
      mr(j)=rMixing_ratio(h,at,p)
      j=j+1
      do i=2 ,  num
        read(input,*) p,at,h
        if (p.LT.pres(j-1)) then
          pres(j)=p
          airt(j)=at
          rh(j)  =h
          alt(j)=Altitude(p)
          pt(j)=Potential_temperature(p,at)
          mr(j)=rMixing_ratio(h,at,p)
          j=j+1
        endif
      enddo
      num=j-1
      do i=1 ,  num-1  ! (generate the gradient for each level)
        deltap=pres(i+1)-pres(i)
        deltat=airt(i+1)-airt(i)
        if ((abs(deltap).GT.0.9).AND.(abs(deltat).GT.temperr)) then
          gradient(i)=deltat/deltap
        else
          gradient(i)=0.0
        endif
      enddo

      ! (set up default preamble)
      preamble(1,1)=num+5
      read(sof,*) sst,at,h
      preamble(1,2)=(at+273.15)*xtoy(1000.0/1013,0.286) -273.15
      preamble(1,3)=h*620.0*vappr(at)/(100.0*(1013.0-vappr(at)))
      preamble(2,1)=-999.0  ! (set default values)
      preamble(4,1)=-999.0
      preamble(3,1)=-1.0
      preamble(5,1)=-1.0
```

```
      do i=2 ,  5   ! (Sets up a default no data preamble.)
        do j=2 ,  3
          preamble(i,j)=-999.0
        enddo
      enddo
      flag=0              !    (The variable "flag" determines how
                          !    many inversions there are.)
      do i=1 ,  NUM-2
        ! begin  (looking for gradients)
        if ((gradient(i).LT.0.0) .and. (flag .EQ. 0)) then
             preamble(2,1)=alt(i)  ! (looking for first inversion)
             preamble(2,2)=pt(i)
             preamble(2,3)=mr(i)
             flag=1  ! (flag remembers where we are in getting values)
             temp1=pt(i)
        endif

        if ((gradient(i).GT.0.0) .and. (flag .EQ. 1)) then
             preamble(3,2)=pt(i)
             preamble(3,3)=mr(i)
             flag=2
             temp2=pt(i)
        endif

c  Checking to see if the inversion is deep enough for a real one.

        if((flag .eq. 2) .and. ((temp2-temp1).lt. 1.5)) then
             flag=0
             preamble(3,2)=-999
             preamble(3,3)=-999
             preamble(2,2)=-999
             preamble(2,3)=-999
             preamble(2,1)=-999
        endif

        if (((gradient(i).LT.0.0). and. (flag .EQ. 2))
     $    .and.(preamble(2,1).LT.(alt(i)-100))) then
             preamble(4,1)=alt(i)  ! (looking for second inversion)
             preamble(4,2)=pt(i)
             preamble(4,3)=mr(i)
             flag=3
             temp3=pt(i)
        endif

           if ((gradient(i).GT.0.0) .and. (flag .EQ. 3)) then
             preamble(5,2)=pt(i)
             preamble(5,3)=mr(i)
             flag=4
             temp4=pt(i)
           endif
        if ((flag .eq. 4) .and. ((temp4-temp3) .lt. 1.5)) then
             flag=2
             preamble(4,1)=-999
             preamble(4,2)=-999
             preamble(4,3)=-999
             preamble(5,2)=-999
             preamble(5,3)=-999
        endif

      enddo
        if ((flag.GE.1) .and. (preamble(3,2) .EQ. -999.0)) then
            !begin (we can't leave an undefined inversion cap.)
             preamble(3,2)=pt(num-2)
             preamble(3,3)=mr(num-2)
        endif

        if ((flag.GE.3) .and. (preamble(5,2) .EQ. -999.0)) then
            !begin (we can't leave an undefined inversion cap.)
             preamble(5,2)=pt (num-2)
             preamble(5,3)=mr(num-2)
        endif
```

```
   if ((flag.GE.3) .and. (preamble(5,2) .EQ. preamble(4,2))) then
     ! begin (we can't leave an undefined inversion cap.)
      preamble(5,2)=preamble(4,2)+1
   endif

if ((flag.GE.3) .and. (preamble(5,3) .EQ. preamble(4,3))) then
  !begin (we can't leave an undefined inversion cap.)
  preamble(5,3)=preamble(4,3)+1
endif
close(input)
close(sof)

END    ! preamb
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Final: September 1993

4. TITLE AND SUBTITLE

THE NAVY OCEANIC VERTICAL AEROSOL MODEL

5. FUNDING NUMBERS

Proj: R035E82
Prog Element: 62435N

6. AUTHOR(S)

S. G. Gathman and K. L. Davidson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 1634

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the FORTRAN version of the Navy oceanic vertical aerosol model (NOVAM). This consists of a written description of the model and the printed code for the subroutine, associated subroutines and functions, and a sample driver which can be used for batch processing of data. To operate, the model requires a surface data file and a meteorological sounding data file. The model predicts the vertical distribution of aerosol in the first 6000 meters above the ocean. Optical properties in the wavelength band of 0.2 to 40 mm, such as the volume extinction coefficient and the volume absorption coefficient as well as the parameters describing dN/dr of the aerosol and the relative humidity at any level, are produced as output from the model.

14. SUBJECT TERMS

Navy oceanic vertical aerosol model (NOVAM)
Navy aerosol model (NAM)

15. NUMBER OF PAGES

112

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE (Include Area Code)	21c OFFICE SYMBOL
S. G. Gathman	(619) 553-1418	Code 543

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 0274B	Library	(2)
Code 0275	Archive/Stock	(6)
Code 50	H. O. Porter	(1)
Code 54	J. H. Richter	(1)
Code 543	R. A. Paulus	(1)
Code 543	D. R. Jensen	(1)
Code 543	K. M. Littfin	(1)
Code 543	C. R. Zeisse	(1)
Code 543	C. P. McGrath	(1)
Code 543	S. G. Gathman	(100)

Defense Technical Information Center
Alexandria, VA 22304–6145 (4)

NCCOSC Washington Liaison Office
Washington, DC 20363–5100

Navy Acquisition, Research and Development
Information Center (NARDIC)
Arlington, VA 22244–5114

GIDEP Operations Center
Corona, CA 91718–8000

NCCOSC Division Detachment
Warminster, PA 18974–5000

Naval Research Laboratory
Washington, DC 20375–5320 (15)

Naval Postgraduate School
Monterey, CA 93943–5100 (31)

Naval Air Warfare Center
Weapons Division
Point Mugu, CA 93042–5000

Naval Surface Warfare Center
Dahlgren Division, Coastal Systems Station
Panama City, FL 32407–5000

Phillips Laboratory (AFSC)
Hanscom Air Force Base, MA 01731–5000 (2)

USAFETAC/DNE
Scott AFB, IL 62226

Pennsylvania State University
University Park, PA 16802

Arete' Associates
Sherman Oaks, CA 91403

Gerber Scientific, Inc.
Reston, VA 22090

TASC
Reading, MA 01867