DIGITAL ATMOSPHERE
Technical Documentation

DIGITAL ATMOSPHERE

Objective Analysis

Program Version 2000 V1.0a (Edition 1A)
February 2003
Tim Vasquez

This technical paper is designed to document the numerical analysis methods used by the Weather Graphics Technologies software product Digital Atmosphere. It is intended to provide a basis for academic and operational understanding of the algorithms used for Digital Atmosphere contouring and objective analysis.

Feedback. Weather Graphics Technologies is committed to quality and improvement in all its products, and will gladly accept feedback and suggestions on these algorithms. This should be sent in writing, either to E-mail to support@weathergraphics.com or by fax to (405) 329-5275.

Source Code. For those users who want to examine specific parts of the objective analysis algorithms for academic or quality improvement purposes, some source code may be shared with end-users in special circumstances. We must have a nondisclosure agreement on file (see the end of this document) faxed to (405) 329-5275. For legal reasons, source code is not releasable outside the United States except to certain European countries, including Great Britain and Germany. The source code is written in Object Pascal V100.

Expressions. In this paper, f is used to represent the value of an observation. This observation may be located at grid coordinates (x,y), which yield the expression $f_{x,y}$. In other cases, f may represent an observation in a random list of reports at arbitrary position k, which yields the value f_k. Various coordinate transformation functions in Digital Atmosphere are responsible for assigning position k to an appropriate (x,y) coordinate and will not be discussed here.

Errors. If you spot an error in this documentation or in these algorithms, be sure that you have the latest version of this technical document (see http://www.weathergraphics.com/da). If the error exists inthe latest technical document, this may be an issue that needs to be fixed in Digital Atmosphere. Help us improve the product! Details should be sent in writing, either to E-mail to support@weathergraphics.com or by fax to (405) 329-5275.

GENERAL DESCRIPTION

Domain. Currently, the analysis domain is a floating (fixed to the screen window) gridpoint model fixed at a size of 30 x 30 in the x and y direction (900 points). This scheme was chosen because the vast majority of objective analysis operations are performed with on-screen results in mind. Since the pixel size of a window rarely exceeds 1200 x 1200 pixels, this allows for 40 x 40 pixels within each grid box. This is close to the minimum level of meaningful on-screen data for most users. Therefore it has not been anticipated that it is necessary to boost the analysis domain beyond 30 x 30 gridpoints, and we have received few such requests. Each objective analysis is an operation that starts completely from scratch each time, therefore there are no complicating factors involved in moving the screen window to a different part of the world. The floating domain simply moves to whatever geographic area is displayed and is not used until an analysis is requested.

Entry. When a field is chosen for analysis or contouring, Digital Atmosphere enters the contouring module through the **GenerateContour** function. Here it is first determined whether a scalar or vector field must be calculated.
— If it is a scalar field, program flow goes directly to **ObjectiveAnalysis**.
— If it is a vector field, **ObjectiveAnalysis** is called for each component of the analysis (i.e. for wind calculations, a calculation is done on the U and then the V wind component).

Following this objective analysis, a contouring or grid plotting routine is called to display the data. The workings of this routine will not be discussed in this paper.

PROCESSING FUNCTIONS

■ **ObjectiveAnalysis**. Upon entry of this function, **BuildWeatherDataFile** and **Build2DGrid** are called in succession (and if height calculations are involved and the user prefers Geostrophic Heights in Preferences > Analysis, the **CalculateHeights** function is called). The module aborts if there are 5 or less observations in the domain (i.e. not enough data).

Otherwise, the Preferences > Analysis panel is checked to see which objective analysis scheme is desired:
— For Nearest Neighbor method: **Expand** is called, followed by one **HaltinerSmooth** iteration, resulting in a crude but efficient objective analysis.
— For Weighted method: **Weighted** is called.
— For Barnes method: **Barnes** is called.
— For Cressman method: **Cressman** is called.

Following the objective analysis, regardless of type, **Expand** is called to make sure there are no null gridpoints. This is necessary because some gridpoint locations may reside at a distance from the nearest observation that far exceeds *a*, which itself is often used to establish the maximum radius of weighting.

At this point, regardless of the analysis scheme, the Smoothing Passes value is obtained from the Preferences > Analysis panel, and the corresponding number of **HaltinerSmooth** iterations are performed. The primary purposes this serves is to smooth out remote gridpoints filled by the final Expand function pass.

■ **BuildWeatherDataFile**. This function constructs a Domain Data File (filename OUTPUT.UX0). This is different from the Processed Data File (OUTPUT.*) in that every observation is applicable to the on-screen analysis window, matching the domain and the chosen date/time. Data outside the on-screen analysis window is discarded.

■ **Build2DGrid**. This function uses the Domain Data File constructed in BuildWeatherDataFile to map the data to a two-dimensional grid. For each observation point, the latitude and longitude is evaluated in respect to the screen window (the domain available to the user in a screen window). The observation is assigned a real-number x coordinate (x=0..30) and y coordinate (y=0..30) as follows:

$$x_k = x / x_m * 30 \qquad y_k = y / y_m * 30$$

where

x,y is the coordinate of the observation on the screen map in pixels
x_m,y_m is the size of the map in pixels
x_k,y_k is the coordinate of the observation in grid units (0..30)

Then, the appropriate observation parameter is determined, using the value only if valid data for the parameter is present in the observation. The station pressure p was determined in the **BuildWeatherDataFile** unit.
— Temperature, dewpoint, sea-level pressure, altimeter setting, wind speed, height. The value

is used verbatim. A conversion to metric units may be accomplished.
— Wind components. Background maps are never displayed in Cartesian coordinates, therefore it is necessary that the declination for a given station be obtained. If a wind component is to be calculated (for any wind operation), the wind declination relative to the map is rectified through the **WindDeclinationCorrect** function. This angle, θ, is the declination produced by plotting a point 300 miles north of the station through a forward coordinate transformation (allowing flexibility if another coordinate transformation is in use, and preventing the need to devise reverse coordinate transformations) . From there, it is a simple matter to calculate wind component (u,v), as follows:

u = -1 * sin(θ) ∗ V v = -1 * cos(θ) ∗ V

where V is the wind velocity.
— Relative humidity. Calculated according to f_k = **MixingRatio**(T_d,p) / **MixingRatio**(T,p).
— Wet bulb temperature. Calculated according to **WetBulb**(T,T_d,p).
— Heat index. Calculated according to **HeatIndex**(T,T_d,p).
— Humidex. Calculated according to **Humidex**(T,T_d).
— Equivalent potential temperature. Calculated according to **EqPotTemp**(p,T,T_d).
— Potential temperature. Calculated according to **PotTempK**(p,T).
— Precipitation type. Freezing rain is triggered by *ww* values of 56, 57, 66, or 67. Ice pelleters are triggered by *ww* values of 79, 87, 88, or 89. Snow is triggered by *ww* values of 70-75 and 83-86. Rain is triggered by *ww* values of 50-55, 60-65, 58, 59, 68, 69, or 80-84.
— IFR/MVFR. MVFR conditions are triggered by ceiling less than 3000 ft or visibility less than 8046 meters. IFR conditions are triggered by ceiling less than 1000 ft or visibility less than 4828 meters. LIFR conditions are triggered by ceiling less than 500 ft or visibility less than 1600 meters.

OBJECTIVE ANALYSIS FUNCTIONS

Nearly all of these algorithms use a smoothing coefficient, *s*, of the range (0..1) which is obtained directly from the Preferences > Analysis panel. This smoothing coefficient is used by the smoothing operators to remove unnecessary noise which would be otherwise evident in the data field.

The Barnes and Cressman algorithms also depend on the average station spacing, *a*. This may be computed *automatically* by setting the Preferences > Analysis > Automatic Smoothing checkmark to **checked**. A *manual* value may be specified by setting the above checkmark to **unchecked**, and the algorithms will use the values specified in Surface Radius or Upper Radius.

■ **Nearest Neighbor analysis**. The Nearest Neighbor method is a crude algorithm designed for brute speed. Each observation is simply mapped to the nearest gridpoint, overwriting any previous values that exist for that gridpoint. Empty gridpoints are filled using an Expansion operator that nudges data from filled to void areas (see **Expand**). Then one pass of the HaltinerSmooth operator (q.v.) is performed to remove noise from the field, using whatever smoothing coefficient has been chosen by the user in the Preferences > Analysis panel.

■ **Weighted analysis**. This is a single-pass objective analysis that simply uses distance-dependent values of observations. Then values at each observation point f_k are used to produce a weighted average for $f_{x,y}$. *NOTE: If the user has selected automated station spacing, then a = a * 1.333.* The source of this analysis scheme is not known at this time.

The algorithm operates by iterating through all 900 gridpoints (x,y). At each gridpoint, each observation f_k is examined. Only those observations whose distance is less than a from (x,y) are considered. A distance-dependent weighting coefficient is calculated for each observation f_k. The value d_k is the distance between gridpoint $f_{x,y}$ and observation f_k in units of a; therefore d_k / a = (0..1). It can be seen that the negative w term forces the expected inverse relationship of distance to weight.

$$f_{x,y} = \Sigma(k=1..n)\ (f_k * -w_k)\ /\ \Sigma(k=1..n)\ w_k \qquad \text{where } w_k = (a^2 - d_k^2) / (a^2 + d_k^2)$$

■ **Barnes analysis**. The Barnes analysis scheme (Barnes, 1964) is considered to be a refinement of the Cressman method. It uses a one-pass scheme, rather than successive iterations, to achieve convergence of the analysis field. *NOTE: If automatic smoothing is enabled, the average station spacing value is set to a = a * 1.333.*

Only observations which fall within a rather large area of influence, a^3, are considered. The value d_k is the distance between gridpoint $f_{x,y}$ and observation f_k in units of a; therefore d_k / a = (0..1). The distance-dependent weighting coefficient and the summation to produce a first guess of gridpoint data value $f_{x,y}$ are as follows:

$$f_{x,y} = \Sigma(k=1..n)\ (f_k * \exp(-w_k))\ /\ \Sigma(k=1..n)\ w_k \qquad \text{where } w_k = d_k^2 / a^2$$

At the completion of the above first-guess calculations, a difference pass is executed to compare the "error" of each actual data value f_k against its *interpolated* predicted value p_k within the objectively analyzed field. The value i_k and j_k will refer to the fractional coordinate (0..1) of the observation within its surrounding objective analysis grid box of ($x..x+1$) and ($y..y+1$). The calculations are done as follows:

$$p_k = [\ (((f_{x+1,y} - f_{x,y}) * i_k) + f_{x,y}) + (((f_{x+1,y+1} - f_{x,y+1}) * i_k) + f_{x,y+1}) + (((f_{x,y+1} - f_{x,y}) * j_k) + f_{x,y}) + (((f_{x+1,y+1} - f_{x+1,y}) * j_k) + f_{x+1,y})\ ]\ / 4$$

The "error" of each data value f_k against its predicted value p_k is determined through simple subtraction.

$$e_k = f_k - p_k$$

Using this "error" information, a final pass is made which uses the above difference to correct the objective analysis field. Again, d_k is the distance between gridpoint $f_{x,y}$ and observation point d_k in units of a. Gamma, γ, ((0..1), usually 0.3) is obtained from the Analysis settings panel. Using the weighting coefficient w, an "error" value for each gridpoint, $z_{x,y}$, can be calculated as follows.

$$z_{x,y} = \Sigma(k=1..n)\ (e_k * \exp(-w_k))\ /\ \Sigma(k=1..n)\ w_k \qquad \text{where } w_k = d_k^2 / a^2 / \gamma$$

Finally, the "error" field is added to the first-guess field to generate a corrected field, yielding a completed objective analysis.

$$f_{x,y} = f_{x,y} + z_{x,y}$$

■ **Cressman analysis**. The Cressman objective analysis scheme (Cressman, 1959) is known as the successive correction method. It achieves its result by forcing convergence of the data to observed, interpolated values using multiple iterations. *NOTE: If automatic smoothing is enabled, the average station spacing value is set to a = a * 1.333.*

First, the number of convergence iterations, p, is established. The value q originates from the Reduction value specified in the Preferences > Analysis panel. The value a is the average station spacing as determined above.

$$p = a / q$$

The predicted value $f_{x,y}$ is calculated for a gridpoint using all observations k=1..n. A distance-dependent weighting factor is determined for each observation before summing it into the equation. The value d_k is the distance between gridpoint $f_{x,y}$ and observation f_k in units of a; therefore d_k / a = (0..1).

$$f_{x,y} = \Sigma(k=1..n)\ (f_k * \exp(-w_k))\ /\ \Sigma(k=1..n)\ w_k \qquad \text{where } w_k = d_k^2 / a^2$$

At the completion of the above calculations, a difference pass is run to compare the "error" of the actual data value f_k against its interpolated, predicted value p_k within the objectively analyzed field. The value i_k and j_k refer to the fractional coordinate of the observation (0..1) within its surrounding objective analysis grid box of (x..x+1) and (y..y+1). The calculations are done as follows:

$$p_k = [\ (((f_{x+1,y}-f_{x,y})*i_k)+f_{x,y}) + (((f_{x+1,y+1}-f_{x,y+1})*i_k)+f_{x,y+1}) + (((f_{x,y+1}-f_{x,y})*j_k)+f_{x,y}) + (((f_{x+1,y+1}-f_{x+1,y})*j_k)+f_{x+1,y})\ ]\ /4$$

The difference between the observation value f_k and its predicted value p_k establishes an difference field (an error field) e_k in much the same way as the Barnes method.

$$e_k = f_k - p_k$$

Then a final pass is made using multiple observations for each gridpoint. A distance-dependent weighting coefficient, w, is determined for each observation point throughout the iterations.

$$z_{x,y} = \Sigma(k=1..n)\ (e_k * -w_k)\ /\ \Sigma(k=1..n)\ w_k \qquad \text{where } w_k = (a^2-d_k^2) / (a^2+d_k^2)$$

The difference field or "error" field is then added to the first-guess field to produce a refined analysis field.

$$f_{x,y} = f_{x,y} + z_{x,y}$$

Then the convergence counter value a is decremented by p as shown below. These values are not processed into the equations; they are only used to loop the Cressman iterations as necessary.

$$a = a - p$$

If a <= 0, the objective analysis is complete and the resulting field $f_{x,y}$ is valid. Otherwise further convergence is necessary, so that the difference values of d_k are reduced to lower and lower values, resulting in more accurate values of $f_{x,y}$. Program flow cycles back to the difference field calculations to achieve this.

NUMERICAL OPERATORS

■ **HaltinerSmooth**. This is a one-dimensional smoothing operator, which is a modified version of the Haltiner smoothing operator defined in Haltiner (1971). The **smoothing coefficient, s,** is obtained from the user-defined value of (0..1) found in the Preferences > Analysis > Smoothing

Coefficient menu. The value of each gridpoint $f_{x,y}$ is smoothed using iterations in all four x/y directions as described below in Steps 1-4. This is done to prevent "smearing" of the fields in a particular direction (something that has been observed in previous implementations). When this is completed, the result is a smoothed data field.

1. Positive Y-direction iteration. Iterating from $f_{x,1}$ to $f_{x,29}$, stepping through each value of x (0..30), $f_{x,y}$ is altered as follows:

$f_{x,y} = (1\text{-}s) * f_{x,y} + ((s * 0.5) * (f_{x,y-1}+f_{x,y+1})$

2. Positive X-direction iteration. Iterating from $f_{1,y}$ to $f_{29,y}$, stepping through each value of y (0..30), $f_{x,y}$ is altered as follows:

$f_{x,y} = (1\text{-}s) * f_{x,y} + ((s * 0.5) * (f_{x-1,y}+f_{x+1,y})$

3. Negative Y-direction iteration. Iterating from $f_{x,29}$ to $f_{x,1}$, stepping through each value of x (0..30), $f_{x,y}$ is altered as follows:

$f_{x,y} = (1\text{-}s) * f_{x,y} + ((s * 0.5) * (f_{x,y-1}+f_{x,y+1})$

4. Negative X-direction iteration. Iterating from $f_{29,y}$ to $f_{1,y}$, stepping through each value of y (0..30), $f_{x,y}$ is altered as follows:

$f_{x,y} = (1\text{-}s) * f_{x,y} + ((s * 0.5) * (f_{x-1,y}+f_{x+1,y})$

■ **QuickFilter**. This fast, two-dimensional smoothing operator is called by certain objective analysis routines. It is essentially a two-dimensional averaging operator that uses 50% self-weight and 50% adjacent weight from neighboring observation values. The filter is iterated through each value of $f_{x,y}$ in the +y direction for each increment of x as follows:

$f_{x,y} = (f_{x,y} * 8) + f_{x-1,y-1} + f_{x,y-1} + f_{x+1,y-1} + f_{x-1,y} + f_{x+1,y} + f_{x-1,y+1} + f_{x,y+1} + f_{x+1,y+1} / 16$

If null data or a border area contains an unusable value of *f*, the value is not included in the summation total, and the divisor is reduced by one.

■ **Expansion**. The Expansion operator is used by certain objective analysis functions to systematically fill null gridpoint values by causing any null gridpoint to assume the value of a neighboring gridpoint which has valid data. This assignment is done only if $f_{x,y}$ is null and the source gridpoint is not null. The calculation, which is done in four directions to avoid "smearing" of the field in a particular direction, is done as follows:

For x = 0..30 do [for y=0..29 do [$f_{x,y} = f_{x,y+1}$]]
For y = 0..30 do [for x=0..29 do [$f_{x,y} = f_{x+1,y}$]]
For x = 0..30 do [for y=30..1 do [$f_{x,y} = f_{x,y-1}$]]
For y = 0..30 do [for x=30..1 do [$f_{x,y} = f_{x-1,y}$]]
Repeat until there are no instances of $f_{x,y}$ = null

■ **AverageStationSpacing**. This is used to determine appropriate filtering values. In this function, Digital Atmosphere measures the great circle distance between all points in the domain and all other points in the domain. The result is the average of these values. Therefore if there are *n* number of observations, there will be $(n\text{-}1)^2$ number of distance measurements which will be averaged (*n*-1 since the distance from an originating station to itself cannot be included).

REFERENCES

Barnes, S. L., 1964: A Technique for Maximizing Details in Numerical Weather Map Analysis. *Journal of Applied Meteorology*, **3**:396-409.

Cressman, G. P., 1959: An Operational Objective Analysis System. *Monthly Weather Review*, **87**:367-374.

Haltiner, G. J., 1971. *Numerical Weather Prediction*, John Wiley & Sons.

_____, and Williams, R. T., 1979: *Numerical Prediction and Dynamic Meteorology*. John Wiley & Sons.

Koch, S. E.; DesJardins, M.; and Kocin, P. J., 1983: An Interactive Barnes Objective Map Analysis Scheme for Use with Satellite and Conventional Data. *Journal of Climate and Applied Meteorology*, **22**:1487-1503.

Little, C. D.; and Phillips, G. L., 1982: A Method of Analyzing Height and Vorticity Fields using AFOS. *NOAA Southern Region Computer Programs and Problems NWS SRCP No. 6*. 15 pp.

NonDisclosure Agreement

If you feel you have a specific need for access to certain parts of Digital Atmosphere objective analysis source code, you may sign and fax this agreement to (405) 329-5275 for consideration.

This is an agreement, effective upon execution, between Weather Graphics Technologies at www.weathergraphics.com (the "Company") and you (the "User"), in which User agrees to respect and protect trade secrets relating to the software program Digital Atmosphere (the "Software") and its algorithms and programming source code (the "Source Code").

Company's Obligations. The Company may, at its discretion, provide User with excerpts of Source Code. The Company is under no obligation to provide User with any documentation, support, or training.

Source Code a Trade Secret. User acknowledges that Source Code is proprietary to, and a valuable trade secret of, the Company and is entrusted to User only for the purpose set forth in this Agreement. User shall treat Source Code in the strictest confidence. User agrees that it will not, without the Company's prior written consent, disclose any Source Code or information about Source Code to any other person, including employers. The User may not copy, share, redistribute, or sell any portion of Source Code. The User understands that provided Source Code may be watermarked with unique equations, configurations, or programming code to allow tracing of unauthorized distribution activities.

Security Precautions. User shall take reasonable security precautions to prevent Source Code from being seen by unauthorized individuals. This includes locking all copies of Source Code and associated documentation in a desk or file cabinet when not in use.

Disclaimer of Warranty. User understands and acknowledges that Source Code accuracy and reliability is not guaranteed. User waives any and all claims it may have against the Company arising out of the performance or nonperformance of Software and Source Code.

Limitation of Liability. The Company shall not be responsible for any loss or damage to User or any third parties caused by Source Code or Software, or by the Company's performance of this Agreement.

No Rights Granted. User understands and acknowledges that the Software and/or Source Code are provided for validation and improvement purposes only. This Agreement does not constitute a grant or an intention or commitment to grant any right, title or interest in the Software, Source Code, or the Company's trade secrets to User. User may not sell or transfer any portion of the Software or Source Code to any third party; reverse-engineer or decompile the Software; or use the Source Code in any manner to produce, market or support its own products and services.

No Assignments. This Agreement is personal to User. User shall not assign or otherwise transfer any rights or obligations under this Agreement.

Entire Agreement. This Agreement contains the entire understanding and agreement of the parties relating to the subject matter hereof. Any representation, promise or condition not explicitly set forth in this Agreement shall not be binding on either party. All additions or modifications to this Agreement must be made in writing and acknowledged by both parties to be effective.

Applicable Law. This Agreement is made under, and shall be construed according to, the laws of the State of Oklahoma.

Signature of User **x**_____ Date _____

Full name/address of User_____

Types of Algorithms required _____

Reason for request_____