4.0 PRODUCT GENERATION ALGORITHMS

Detailed information on certain algorithms required by various processing functions defined in 3 of this specification are provided in this Appendix as follows:

Function	Algorithm	Subsection
Precipitation Analysis	Weak Echo Region	4.1
Product Synthesis	Contour	4.2
Product Synthesis	Polar to Grid Conversion	4.3
Product Compression	Run Length Encoding	4.4
Product Synthesis	Cross-Section	4.5
Product Synthesis	Polar/LFM Conversion	4.6

4.0 Weak Echo Region Product A1gorithm Procedures

4.1 Algorithm

Begin Algorithm (Weak Echo)

1.0 Do For All (Weak Echo Products)
1.1 Identify Storm

1.2 Generate 50 by 50-km grid centered at x,y centroid position of storm

1.3 Do For All (Storm Components)

1.3.1 Compute storm motion correction **(See Note 1)**

1.3.2 Identify base reflectivity data within 50 by 50 window

1.3.3 Convert base data levels into three data levels (e.g., 32, 40 and 50 dBz)

1.3.4 Project data onto 50 by 50 grid (on flat earth surface) correcting for storm motion **(See Note 1) (See Note 2)**

End Do

1.4 Do For All (Planes)

1.4.1 Perform Isometric Projection

End Do

Note 1: Refer to 1.1 Computation (Next Page)

Note 2: Gridded value contains the “greatest of” the polar cells that map into the grid box.

End Do

End Algorithm (Weak Echo)

4.1 Computation

1.3.1 Compute (Storm Motion Correction)

$$XCORR_{i+n} = -[-SIN(DS)](SS)(T_{i+n} - T_i)$$

$$YCORR_{i+n} = -[-\cos(DS)](SS)(T_{i+n} - T_i)$$

where,

n = number of elevations above lowest storm component elevation i.
Time (T) = time at which data was collected in hours.
Direction (DS) = direction from which a storm is moving in radians.
Speed (SS) = speed of a storm in km/hr.

1.3.4 Compute (Polar to Grid Conversion) **(See Note 3)**

- Get start/stop location of azimuth and range for base data
- Calculate sine and cosine for the current azimuth

  ST = sin (θ)
  CT = cos (θ)

- For each azimuth, identify in what zones do the start/stop ranges apply.

  Zone 1 = Range ≤ 57 km
  Zone 2 = Range ≤ 114 km
  Zone 3 = Range ≤ 171 km
  Zone 4 = Range ≤ 230 km

- For those zones identified, perform the following angle computations (Figure 4-1):

  Zone 1: No additional computation

  Zone 2:
  ST3A = sin(θ – 1/4°) = ST – CT(0.0043633)
  ST3B = sin(θ + 1/4°) = ST + CT(0.0043633)
  CT3A = cos(θ + 1/4°) = CT + ST(0.0043633)
  CT3B = cos(θ – 1/4°) = CT – ST(0.0043633)

Note 3 : For more detail on polar to Cartesian conversion, refer to

Zone 3:
ST4A = sin(θ - 1/3°) = ST - CT (0.058178)
ST4B = sin(θ) = ST
ST4C = sin(θ + 1/3°) = ST + CT(0.058178)
CT4A = cos(θ - 1/3°) = CT + ST(0.058178)
CT4B = cos(θ) = CT
CT4C = cos(θ + 1/3°) = CT - ST(0.058178)

Zone 4:
ST5A = sin(θ - 3/8°) = ST - CT(0.0065450)
ST5B = sin(θ - 1/8°) = ST - ST(0.0021817)
ST5C = sin(θ + 1/8°) = ST + CT(0.0021817)
ST5D = sin(θ + 3/8°) = ST + CT(0.0065450)
CT5A = cos(θ - 3/8°) = CT + ST(0.0065450)
CT5B = cos(θ - 1/8°) = CT + ST(0.0021817)
CT5C = cos(θ + 1/8°) = CT - ST(0.0021817)
CT5D = cos(θ + 3/8°) = CT - ST(0.0065450)

! Compute Constants

x=1
y=1
MULT = SF/GBS
Constant = DR * cos(θ)
.....
x-constant = [constant + XCORR + Offset] * MULT
.....
y-constant = [constant + YCORR + Offset] * MULT
.....

! Do For All (Azimuths)

IF (Zone 1) Then

dx = integer [x-constant * ST]
dy = integer [y-constant * CT]

Do For All (Zone 1 cells)

y = y + dy
x = x + dx
Box (XH(1), YH(1) = ...
End Do
End IF

IF (Zone 2) Then

dx1 = integer [x-constant * ST3A]
dx2 = integer [x-constant * ST3B]
dy1 = integer [y-constant * CT3A]
dy2 = integer [y-constant * CT3B]

Do For All (Zone 2 Cells)

yl = yl + dy1

```
            y2 = y2 + dy2
            x1 = x1 + dx1
            x2 = x2 + dx2
            Box (XH1(1), YH1(1)) = .....
            Box (XH2(1), YH2(1)) = .....

        End Do

    End IF

    IF (Zone 3) Then

        .....Logic identical to Zone 2 .....


    End  IF

    IF (Zone 4) Then

        .....Logic identical to Zone 2 .....

    End IF

End  Do
```

4.2 Contour

4.2.1 Algorithm Description

The Contouring Algorithm, applied to all NEXRAD contour products, utilizes multiple passes of input grid smoothing. interpolation between adjacent grid-point values, and output data filtering in order to generate contour paths at the levels specified by the contour base and interval, in linked-vector format.

First, the amount of scratch buffer space needed for smoothing, and for auxiliary arrays used during the contouring process. is determined as a function of the number of gridpoints in each dimension of the field being contoured. Once the scratch buffer space is acquired. several passes of smoothing are performed on be input grid field.

The preliminary, or zeroeth, pass is only performed when the combination of the product resolution and coverage area yields a grid size of greater than 116 points in each direction. During this pass, each set of four points, representing the corners of a square, is averaged in isolation from all neighboring sets of points, with the results being stored at the midpoints of the squares. The averaging is performed in the biased, integer units in which the particular field being contoured is represented internally, with only those gridpoints containing valid data values (i.e., not flagged 'Below Thresholds ', 'No Data', etc.) included in the averaging. If more than half (i.e., three or more) of the points contributing towards an average contain flag values, a flag value is, in turn, stored as the result. At the completion of this pass of the smoothing, the original grid dimensions will have been reduced by 2:1 in each direction.

The first and second smoothing passes are performed on all products, in a manner similar to that described above (i.e., four-point averages using only non-flagged points. with the results, or a flag, stored at the midpoints. etc.). although now with continuity achieved between adjacent gridpoints. This is accomplished by having each point contribute towards the averages of each of the squares which it corners (i.e., each interior point contributes towards four averages; each border point towards two; and each point at the corners of the grid domain towards one). There is no gridpoint reduction during these two passes. In fact, when considered together, these two passes return the results to the same set of gridpoints at which they were represented at the start. The net effect of these two passes is, in fact, a nine-point, centered average (for each interior gridpoint) using the weights: 1,2,1 (row above gridpoint): 2,4,2 (row containing gridpoint); and 1,2,1 (row below gridpoint). Along the borders, all points are set to a flag value which will always be below the lowest possible contour value in biased, integer units.

At the end of the smoothing process, the maximum field value on the final, smoothed grid is determined. It is retained in both biased, integer units and in the meteorological units commonly associated with the field being contoured.

The Contour Table, containing the levels to be contoured, is now built from the contour base and interval (which may be either site adaptable, user selectable, or internally set, depending on the particular contour product). This is done by starting wizh the base and incrementing, successively, by the interval until either the maximum field value is exceeded or the maximum number of contour levels for the product is reached.

The amount of output buffer space needed, considering the maximum number of contour levels and the grid size of the current product, as well as any space needed for product headers and alphanumeric blocks, is now acquired. An estimate is then made of the size of the largest conceivable contour for the product (in output buffer words), and this amount is subtracted from the portion of the output buffer used for contouring in order to establish a Critically Full Level, which, if exceeded, will cause processing to cease prematurely.

The contour levels are processed from lowest to highest, with all potential contours analyzed at each level before proceeding to the next. The basic method involves the notion that the gridded data field is superimposed on a pixel field of much higher resolution, on which the contours will be drawn, separating regions above the contour value from regions below it. Each contour is initialized at a pixel point which is between a pair of adjacent gridpoints in the same row, where one of the gridpoints has a value greater than or equal to the contour value and the other has a value below it. The exact location of the pixel is determined by linear interpolation of the contour value and the two gridpoint values, and is identified in i,j coordinates relative to the grid-domain center, where i corresponds to the X axis (i.e., moving along a row) and j corresponds to the negative Y axis (i.e,, moving down a column).

The processing proceeds with the contour being assigned a Direction of Travel, determined on the basis that as the contour is drawn, higher field values will always be kept to its right and lower field values to its left. The Direction of Travel may thus be either North or South (if along a row axis) or East or West (if along a column axis). The contour is also associated with a Reference Gridpoint (I,J) on the gridded data field, which will always be the point to its immediate right, relative to its Direction of Travel.

An iterative process is then undertaken whereby the contour is "drawn" by successively moving to new locations while vectors, representing these moves, are stored in the product output buffer. This is done by examination of several gridpoints ahead of and to the sides of the contour's current location, in order to find another pair of points satisfying the condition that the one to the right is greater than or equal to the current contour value while the one to the left is less than it (a hierarchy of logic is used to determine the order in which the gridpoint pairs are searched, which generally involves starting ahead and to the left of the current Reference Gridpoint and sweeping clockwise). Once such a pair is found, linear interpolation is again employed to find the exact pixel location of the contour value between the two gridpoint values, and the contour is moved to this point (i.e., a linked vector is stored in the product output buffer with the point's i,j coordinates). The contour's Direction of Travel and Reference Gridpoint are then updated in preparation for the next move. This process continues until the contour returns to its initial location.

In order to avoid duplication of contour paths and, at the same time, guarantee that all possible contours of a given level are analyzed, a Marker array is utilized which denotes the locations in all rows where the condition exists that one of a pair of adjacent gridpoints is greater than or equal to the current contour value while the other is less than it. This Marker array, which is in the form of a bit map, is fully established at each level prior to the analysis of any contours at that level. It is then used to identify starting locations of contours. Furthermore, as each contour is traced, markers are cleared as they are passed, preventing the same path from being taken again. When each contour is completed, the Marker array is re-examined and, if any markers remain, they are used to identify new starting locations and to assist in the tracing of subsequent contours. This process continues until all markers at the current contour level have been turned off, at which time a new set of markers is established in preparation for processing the next level in the Contour Table (if any).

Coincident with the above-described operations, a technique is employed to reduce processing time by limiting the remaining portion of the grid field which needs to be examined as the analysis proceeds to higher and higher contour levels. This is accomplished by the setting of Boundary arrays so as to enclose the regions of the gridded field where contour analysis may still be necessary, excluding regions where it has been determined that no more contours can be drawn.

Two more features are incorporated in an attempt to limit the size of the contour product buffers. A Filter Level, set via adaptation data. is used to define the minimum size of contours (in number of vector-moves) that will be retained in the output buffer. Any contour that is at or below this size limit is discarded, which is accomplished by the resetting of pointers so as to erase the current contour from the output buffer. (It should be noted that for contours that are retained. an accompanying header field, and the starting position of the contour, are packed only at this time - after the determination has been made to keep the contour.)

Finally. the previously referenced Critically Full Level is compared to the word count in the output buffer after each valid contour is packed. If the Critically Full Level is exceeded. processing immediately terminates, yielding a product which may exclude some of the higher-level contours.

4.2.2 Algorithm Inputs

Identification

INGRID	Two-dimensional, input grid field to be contoured, with data values represented in biased, integer units.
GRIDSIZE	Size of input grid (i.e., number of gridpoints) in each of the X and Y directions.
GRIDRES	Resolution of input grid, in km.
THR_REDUC	Grid size greater than which input grid field must be reduced during smoothing (=116).
VALIDAT	Valid Data Threshold. separating flag values from valid data values on the particular grid field being contoured (=2: Comp. Refl; =1: Echo Tops; **(See Note 1)**: Comb. Shear).
BASE	Contour Base. used to define lowest contour level. (Note: this will always be above any 'Below Threshold' or 'No Data' levels.)
INTERVAL	Contour Interval. i.e., interval by which the Base is successively incremented to determine other contour levels.
SCALE	Scaling factor used for internal representation of product data (=2: Comp. Refl; =1: Echo Tops; =2: Comb. Shear).
BIAS_1	Offset factor used for internal representation of product data (=2: Comp. Refl; =1: Echo Tops; =2: Comb. Shear).

Note 1: Computed as a function of adaptation data: Default = 6

BIAS_2	Biasing factor used for internal representation of product data (=32: Comp. Refl; =0: Echo Tops; =0; Comb. Shear).
FILTLEV	Contour Filter Level - Size up to which contours will be filtered from the product output buffer. in number of vector-moves.
MAX_INPUT _VAL	Maximum value on input grid, in biased integer units. (particular to the field being contoured).
MAXLEVS	Maximum number of levels to be contoured for the given product (=11: Comp.

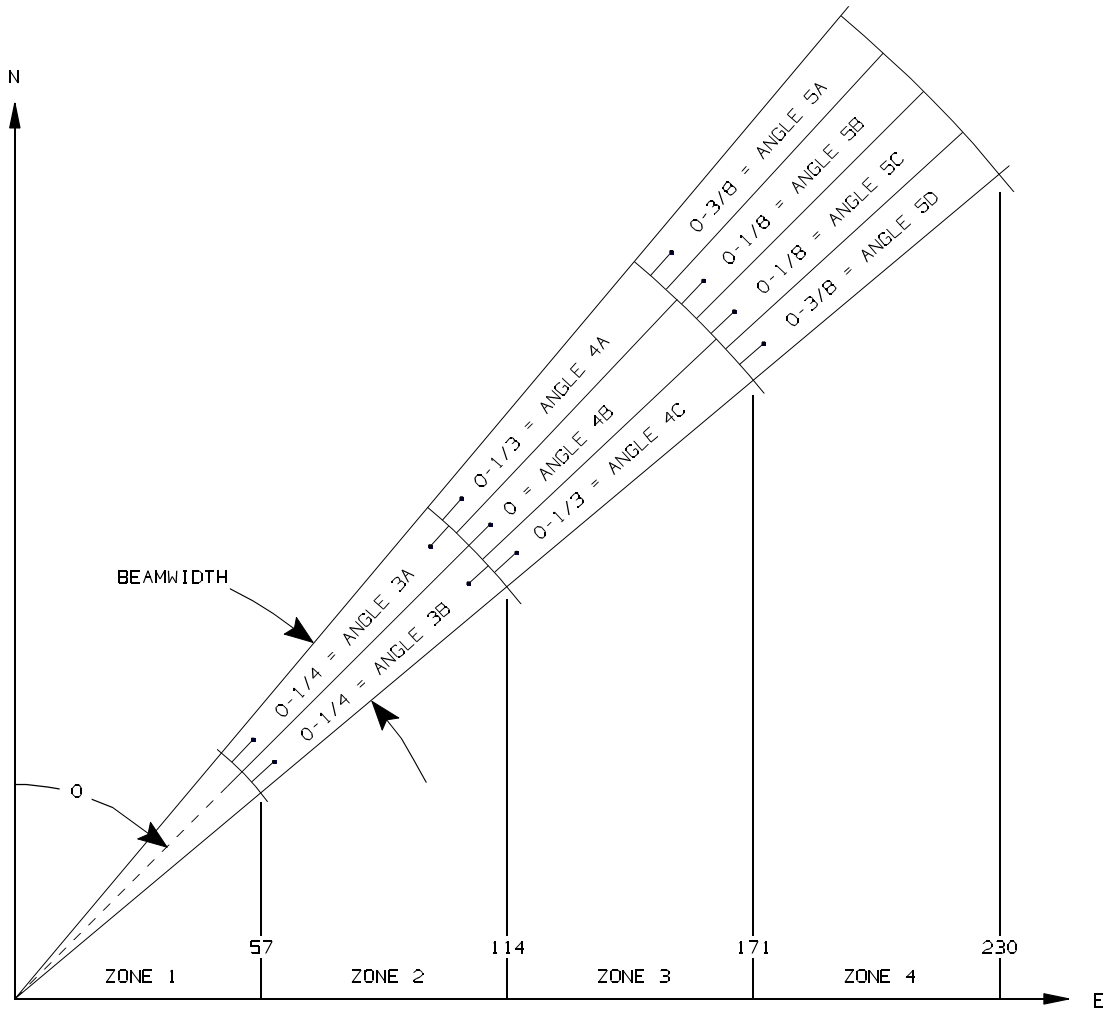

Figure 4-1. Beam Splitting Procedure

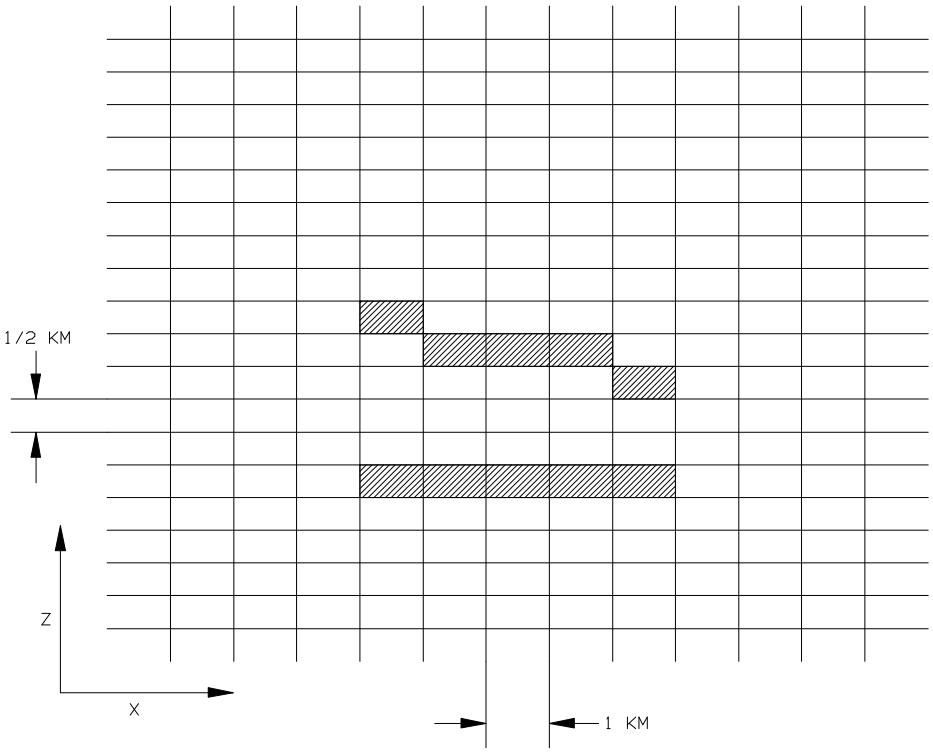

Figure 4-2. Cross-Section Intercepted Range (Top View)

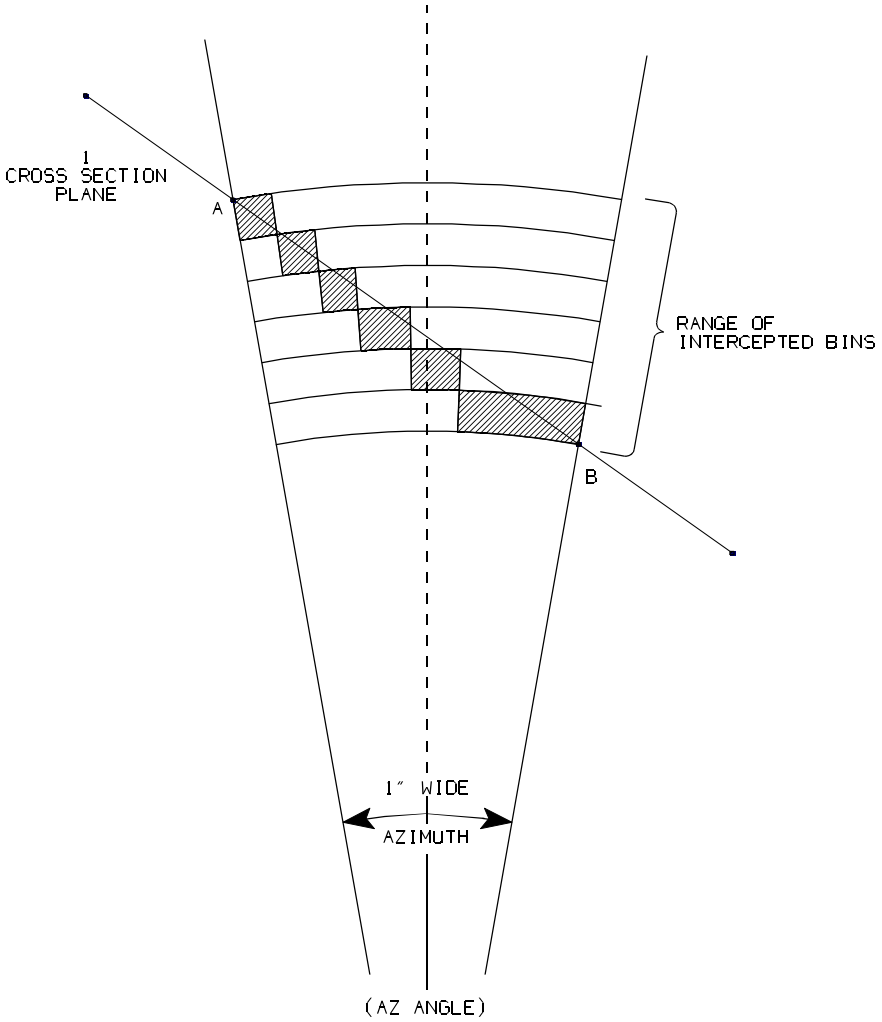

Figure 4-3. Cross-Section Intercepted Grid (Side View)

Refl.; =11: Echo Tops; =5: Comb. Shear).

MAX_BUFSIZ Maximum allowable size for contour portion of output buffer, for any contour product. in (32-bit) integer fullwords (=10,000; equivalent to 40,000 bytes).

Acquisition

BASE and INTERVAL are acquired from Adaptation Data, via User Request, or are internally set, depending on the product:

- For Echo Tops Contour, the Base is set via adaptation data and the Interval is user specified;
- For Composite Reflectivity Contour, the Base is set internally (at 20 dBz) and the Interval is specified via adaptation data;
- For Combined Shear Contour, the Base is set equal to the Combined Shear Threshold adaptable parameter used in the corresponding algorithm, and the Interval is specified via algorithm adaptation data as well.

FILTLEV is set via Adaptation Data.

INGRID and MAX_INPUT_VAL are derived fields generated by a preliminary algorithm in each instance, and are passed to the Contouring algorithm via the calling, product-processing task.

GRIDSIZE and GRIDRES are supplied via the product-processing task as calling arguments.

VALIDAT, SCALE, BIAS_1, BIAS_2, MAXLEVS, MAX_BUFSIZ and THR_REDUC are supplied via the product-processing task as parameterized constants.

4.2.3 Algorithm Enunciation Language (AEL)

4.2.3.1 Procedure

```
BEGIN ALGORITHM (Contour)

1.0 ACQUIRE (Scratch Workspace)
1.1 IF (Successfully acquired) THEN

    1.1.1  IF product gridsize exceeds threshold requiring reduction (THR_REDUC) THEN
      2.0    DO FOR ALL. input grid rows (JG), by two
        2.1    DO FOR ALL input grid columns (IG), by two
          2.2    IF half, or more, of the (four) points given by the indices: {(IG,JG),
                 (IG+1,JG), (IG,JG+1), (IG+l,JG+1)} contain Valid Data values THEN
                     2.2.1  COMPUTE (Smooth Grid-0im,jm)
                 ELSE
                     2.2.2  SET (Smooth Grid-0im,jm) to flag value TEMPFLAG
                 END IF
```

2.3 STORE result at midpoint of the square represented by the four gridpoints (note that total grid field size is reduced by 2:1 in each direction in this smoothing pass)

END DO
END DO
END IF

3.0 DO FOR ALL rows (JS) of current grid field, from top to bottom-minus-one (i.e., in increasing direction), by one

3.1 DO FOR ALL columns (IS) of current grid field, from left to right-minus-one (i.e., in increasing direction) , by one

3.2 IF half. or more, of the (four) points given by the indices: {(IS,JS),(IS+1,JS),(IS,JS+1),(IS+1,JS+1)} contain Valid Data values THEN

3.2.1 COMPUTE (Smooth Grid-$1_{is,js}$)
ELSE
3.2.2 SET (Smooth Grid-$1_{is,js}$) to flag value TEMPFLAG
END IF

END DO
END DO

4.0 DO FOR ALL rows (JS) of current grid field, from bottom-minus-one to top-plus-one (i.e., in decreasing direction) , by one

4.1 DO FOR ALL columns (IS) of current grid field, from right-minus-one to left-plus-one (i.e., in decreasing direction) , by one

4.2 IF half, or more, of the (four) points given by the indices: {(IS,JS),(IS-1,JS), (IS,JS-1),(IS-1,JS-l)} contain Valid Data values THEN

4.2.1 COMPUTE (Smooth Grid-$O_{is,js}$)
ELSE
4.2.2 SET (Smooth Grid-$2_{is,js}$) to flag value FINALFLAG
END IF

END DO
END DO

4.3 DO FOR ALL points around borders of current grid field

4.3.1 SET (Smooth Grid-$2_{is,js}$) to flag value FINALFLAG

END DO

5.0 COMPUTE (Maximum (Biased) Smoothed Grid Value)
5.1 COMPUTE (Maximum (Unbiased) Smoothed Grid Value)

6.0 DO UNTIL (Maximum Number of Levels to be Contoured reached OR Maximum (Unbiased) Smoothed Grid Value exceeded)
6.1 COMPUTE (Contour Table (Unbiased)$_n$)

6.2 COMPUTE (Contour Table (Biased)$_n$)
END DO
7.0 COMPUTE (Number of Contour Levels)

8.0 COMPUTE (Output Size)
8.1 ACQUIRE (Output Buffer)
8.2 IF (Successfully acquired) THEN

9.0 COMPUTE (Spare Size)
9.1 COMPUTE (Critical Level)

10.0 COMPUTE (Center Offset Position)
10.1 COMPUTE (Quarter-Kilometer Expansion Factor)

11.0 DO FOR ALL levels in Contour Table (Biased)

12.0 DO FOR ALL rows on smoothed grid
12.1 Set the Left Boundary at the first gridpoint encountered when searching in from the left border, whose value is greater than or equal to the current contour value
12.2 Set the Right Boundary at the first gridpoint encountered, when searching in from the right border, whose value is greater than or equal to the current contour value
12.3 IF no such points are encountered, flag the row as having 'None'
END DO

12.4 Set the Top Boundary at the first row encountered, when searching in from the top border, which is not flagged as having no points with values greater than or equal to the current contour value
12.5 Set the Bottom Boundary at the first row encountered, when searching in from the bottom border, which is not-flagged as having no points with values greater than or equal to the current contour value
12.6 IF no rows are encountered having at least one point with a value greater than or equal to the current contour value, flag the entire grid field as having 'None'

13.0 DO FOR ALL rows from Top Boundary to Bottom Boundary
13.1 DO FOR ALL columns from Left Boundary to Right Boundary plus one
13.2 IF a transformation occurs between a pair of successive gridpoints such that the left point is less than the current contour value while the right point is greater than or equal to the current contour value THEN
13.2.1 Set a Left Marker on a bit map corresponding to the interface between the two points
13.3 ELSE IF a transformation occurs between a pair of successive gridpoints such that the left point is greater than or equal to the current contour value while the right point is less than the current contour value THEN
13.3.1 Set a Right Marker on a bit map corresponding to the interface between the two points
END IF

END DO

END DO

14.0 DO UNTIL no more Markers remain turned on for this contour level. OR the output buffer becomes full above the Critical Level

14.1 SEARCH the entire gridded field for Markers, representing the Starting Points of contours (searching first for Left Markers, then for Right Markers)

14.2 IF marker found is a Right Marker THEN

14.2.1 Set initial Direction of Travel as South

14.2.2 Set $Factor_i$ to +1

14.2.3 Set $Factor_j$ to 0

14.3 ELSE IF marker found is a Left Marker THEN

14.3.1 Set initial Direction of Travel as North

14.3.2 Set $Factor_i$ to -1

14.3.3 Set $Factor_j$ to 0

END IF

15.0 Set Reference Gridpoint as point on smoothed grid to immediate right of Marker, relative to contour's Direction of Travel

15.1 COMPUTE (Interpolation)

15.2 COMPUTE (Contour $Coordinates_{ip,jp}$)

15.3 STORE Initial (Contour $Coordinates_{ip,\,jp}$) in the product buffer, as the Starting Points of a linked vector

16.0 DO UNTIL contour reaches point where Marker has been turned off OR returns to initial point

16.1 IF Marker exists along interface where contour currently located, turn Marker off

16.2 Examine the points on the smoothed grid to the Left, Forward-Left. Forward-Right and Right (in that order) of the contour's current location until the first interface between an adjacent pair of points is found such that the left point of the pair is less than the contour value while the right point is greater than or equal to the contour value. WHEN FOUND

16.3 IF the current and what would be the next Reference Gridpoint are not both exactly equal to the contour value while all points to either side of them are less than the contour value THEN

16.3.1 Move (i.e. draw) the contour to the interface between the appropriate pair of points

16.3.2 Reset Reference Gridpoint as the right point, among pair along new interface

16.3.3 Reset Direction of Travel appropriately to having Reference Gridpoint to right, when intersecting interface

16.3.4 IF Direction of Travel is now South THEN

16.3.4.1 Set $Factor_i$ to +1

16.3.4.2 Set $Factor_j$ to 0

16.3.5 ELSE IF Direction of Travel is now West THEN

16.3.5.1 Set $Factor_i$ to 0

16.3.5.2 Set $Factor_j$ to +1

16.3.6 ELSE IF Direction of Travel is now North THEN

16.3.6.1 Set $Factor_i$ to -1

16.3.6.2 Set $Factor_j$ to 0

16.3.7 ELSE IF Direction of Travel is now East THEN
16.3.7.1 Set $Factor_i$ to 0
16.3.7.2 Set $Factor_j$ to -1
END IF

16.3.8 COMPUTE (Interpolation)
16.3.9 COMPUTE (Contour $Coordinates_{ip,jp}$)
16.3.10 STORE new (Contour $Coordinates_{ip,jp}$) in the product buffer, as the Ending Points of a linked vector

16.4 ELSE
16.4.1 Do Not move contour to new interface, which would result in the drawing of a 'Line Contour'
16.4.2 Retain Reference Gridpoint as is
16.4.3 Rotate Direction of Travel by +90° (clockwise) (i.e.. North-->East-->South-->Wes-->North)
16.4.4 Do Not store (Contour $Coordinates_{ip,jp}$) in product buffer (since same as previous)
END IF

END DO (...until contour reaches point where marker turned off or returns to initial point)

17.0 If number of vector moves for completed contour is greater than the Filter Level, FILTLEV, THEN
17.1 Store header fields for contour in product buffer
17.2 Store Contour Length in product buffer
17.3 Reset End-of-Buffer Pointer to last position occupied by contour just packed

ELSE

17.4 Remove contour from product buffer by restoring End-of-Buffer Pointer to position occupied just prior to processing of present contour

END IF
END DO (... until no more Markers on, or full above Critical Level)

18.0 IF End-of-Buffer Pointer now exceeds Critical Level, discontinue processing and generate message indicating premature termination; all contours not drawn

END DO (... for all levels in Contour Table)

END IF (... output buffer successfully acquired)
END IF (... scratch workspace successfully acquired)

END ALGORITHM (Contour)

4.2.3.2 Data Definitions

SM_GRID_O Input grid field after zeroeth smoothing pass, in biased, integer units.

IG	Input gridpoint index in X direction (i.e.. column index).
JG	Input gridpoint index in Y direction (i.e., row index).
NVALIDAT	No. of points (among the four) with valid data values.
SM_GRID_1	Input grid field after first smoothing pass, in biased, integer units.
IS	Gridpoint index in X direction on smoothed grid (i.e., column index).
JS	Gridpoint index in Y direction on smoothed grid (i.e., row index).
SM_GRID_2	Input grid field after second smoothing pass, in biased, integer units.
TEMPFLAG	Temporary value to which flagged gridpoints will be set daring the input grid smoothing process. until final pass (=0, all products).
FINALFLAG	Final value to which flagged gridpoints will be set after completion of input grid smoothing process, just prior to contouring (=66: Comp. Refl; =0: Echo Tops; =0: Comb. Shear).
IS_{min}	Minimum gridpoint index in X direction on smoothed grid.
IS_{max}	Maximum gridpoint index in X direction on smoothed grid.
JS_{min}	Minimum gridpoint index in Y direction on smoothed grid.
Js_{max}	Maximum gridpoint index in Y direction on smoothed grid.
MAX_SM_VAL_BIAS	Maximum value on final, smoothed grid. in biased. integer units (particular to the field being contoured).
MAX_SM_VAL_UNBI	Maximum value on final. smoothed grid, in meteorological units commonly associated with the field being contoured.
CONTABLE	Table of levels to be contoured. in meteorological units.
BICONTABLE	Table of levels to be contoured. in biased, integer units.
NUM_LEVS	Number of levels to be contoured.
N_{max}	Highest level-index determined during analysis of Contour Tables.
OUT_SIZ	Total output buffer space needed per product. in (32-bit) integer fullwords (will not exceed 11,000; or 44,000 bytes).
HDR_SIZ	Size of header for non-contour portion of product. in (32-bit) integer fullwords.
CON_SIZ	Output buffer space needed for contour portion of product. in (32-bit) integer fullwords (product/version dependent; not to exceed MAX_BUFSIZ).
ALPHA_SIZ	Output buffer space needed for alphanumeric portion of product. (32-bit) integer fullwords (product dependent may be zero)

SPARE_SIZ Spare space needed within contour portion of output buffer to prevent overflow, in (32-bit) integer fullwords.

VEC_SIZ Size of one contour (i.e., linked) vector. in integer half-words (=2)

NBORDS No. of borders on grid field (=4).

FWD_HWD Full word-half word ratio (=2).

CRITLEV Critically-full output buffer level. in (32-bit) integer fullwords.

OFFCENT Offset to grid center, in smoothed-grid coordinates.

CORREC Correction.Factor (=1.0).

NSIDES No. sides in each direction of grid domain (=2).

QKM_EXPANS Expansion factor for ratio of smoothed grid to pixel grid (of 1/4-km resolution) on which contours analyzed.

SMOOTH_RES Resolution of final, smoothed grid. in km.

QRT_KLM Constant inversely proportional to pixel grid resolution (=4).

INTERP Linearly-interpolated value indicating fractional distance to present position of contour line along interface between Reference (i.e. Right) and Left Gridpoints, on smoothed grid.

$GRIDREF_{i,j}$ Value. in biased. integer units, of Reference Gridpoint; i.e. point on smoothed grid to immediate right of contour. relative to its Direction of Travel.

GRIDLT Value. in biased, integer units, of point on smoothed grid to immediate left of contour. relative to its Direction of Travel (and adjacent to the Reference Gridpoint)

CONVAL Current contour value, in biased, integer units (=n^{th} entry in Contour Table (Biased))

C0N_ip Current contour i-location (i.e.. column index). in pixel coordinates of 1/4-km resolution. where increasing i corresponds to the positive X axis.

CON_jp Current contour j-location (i.e.. row index), in pixel coordinates of 1/4-km resolution, where increasing j corresponds to the negative Y axis.

$FACTOR_i$ Multiplication factor for I-coordinate. indicating how interpolation should be applied in this direction.

$FACTOR_j$ Multiplication factor for j-coordinate. indicating how interpolation should be applied in this direction.

4.2.3.3 Symbolic Formulae

COMPUTE (Smooth Grid-$0_{im,jm}$):

$$SM\_GRID\_0_{im,jm} \quad [\sum_{j=JG}^{JG+1} \sum_{i=IG}^{IG+1} INGRID(i,j)] \; / \; NVALIDAT$$

where:

∑∑ (Summation) Includes only those gridpoints with valid (i.e. non-flagged) data values;
im,jm = Index or the midpoint of the square represented by the four input gridpoints {(IG,JG), (IG+1,JG), (IG,JG+1), (IG+l,JG+l)};
i = Incremental index in X direction;
j = Incremental index in Y direction.

COMPUTE (Smooth Grid-$l_{IS,JS}$):

$$SM\_GRID\_1_{IS,JS} = [\sum_{j=JG}^{JG+1} \sum_{i=IG}^{IG+1} INGRID(i,j)] \; / \; NVALIDAT$$

(If zeroeth smoothing pass not performed)

or:

$$SM\_GRID\_1_{IS,JS} = [\sum_{j=JS}^{JS+1} \sum_{i=IS}^{IS+1} SM\_GRID\_0(i,j)] \; / \; NVALIDAT$$

(If zeroeth smoothing pass is performed)

where:

∑∑ (Summation) Includes only those gridpoints with valid (i.e. non-flagged) data values;
i = Incremental index in X direction;
j = Incremental index in Y direction.

COMPUTE (Smooth Grid-$2_{IS,JS}$):

$$SM\_GRID\_2_{IS,JS} = [\sum_{j=JS}^{JS-1} \sum_{i=IS}^{IS-1} SM\_GRID\_1(i,j)] \ / \ NVALIDAT$$

where:

$\sum\sum$ (Summation)		Includes only those gridpoints with valid (i.e. non-flagged) data values;
i	=	Decremental index in X direction;
j	=	Decremental index in Y direction.

COMPUTE (Maximum (Biased) Smoothed Grid Value):

$$MAX\_SM\_VAL\_BIAS = Max[\ [(SM\_GRID\_2_{i,j}), i = IS_{min}, IS_{max}],\ j = JS_{min}, JS_{max}]$$

where:

i	=	Incremental index in X direction;
j	=	Incremental index in Y direction;
Max	=	A function which extracts the maximum value amongst a field of data.

COMPUTE (Maximum (Unbiased) Smoothed Grid Value):

$$MAX\_SM\_VAL\_UNBI = (MAX\_INPUT\_VAL - BIAS\_1) \ / \ SCALE - BIAS\_2$$

COMPUTE(Contour Table (Unbiased)$_n$):

$$CONTABLE_n = BASE + (n-1) * INTERVAL$$

where:

n	=	Table-level index.

COMPUTE (Contour Table (Biased)$_n$):

$$BICONTABLE_n = SCALE * (CONTABLE_n + BIAS\_2) + BIAS\_1$$

where:

n	=	Table-level index.

COMPUTE (Number of Contour Levels):

$$NUM\_LEVS = Min[N_{max}, MAXLEVS]$$

where:

Min = A function which extracts the minimum value amongst a field of data.

COMPUTE (Output Size):

$$OUT\_SIZ = HDR\_SIZ + CON\_SIZ + ALPHA\_SIZ$$

COMPUTE (Spare Size):

$$SPARE\_SIZ = NBORDS * VEC\_SIZ * IS_{MAX} / FWD\_HWD$$

COMPUTE (Critical Level):

$$CRITLEV = CON\_SIZ - SPARE\_SIZ$$

COMPUTE (Center Offset Position)

$$OFFCENT = (IS_{\max} + CORREC) / NSIDES$$

COMPUTE (Quarter-Kilometer Expansion Factor):

$$QKM\_EXPANS = SMOOTH\_RES * QRT\_KLM$$

COMPUTE (Interpolation):

$$INTERP = (GRIDREF_{i,j} - CONVAL) / (GRIDREF_{i,j} - GRID_{LEFT})$$

COMPUTE (Contour Coordinates $_{ip,jp}$):

$$CON\_ip = Nint\ [\ [GRIDREF_{i,j} - OFFCENT + INTERP * FACTOR_i] * QKM\_EXPANS]$$

$$CON\_jp = Nint[[GRIDREF_{i,j} - OFFCENT + INTERP * FACTOR_j] * QKM\_EXPANS] * Neg$$

where:

Nint = An operator which rounds a real result to the nearest integer;

Neg = A multiplication factor which negates a result.

4.2.4 Algorithm Outputs

Identification

CONTBUF — The portion of the product output buffer filled with encoded, contour data in linked-vector format.

WORDCNT — The final count of contour half-words (i.e.. 16-bit) in contour portion of product buffer.

Distribution

Both CONTBUF and WORDCNT are returned to the calling product task, for incorporation into its complete product buffer.

4.2.5 Inferences

Limitations

If the Critically-Full level (CRITLEV) is exceeded during processing, the algorithm will not draw any more higher level contours.

Choosing a high Filter Level (FILTLEV) will help reduce the size of the product but will result in some small contours (i.e., those consisting of a small number of vector-moves) being omitted.

4.3 Polar To Grid Conversion A1gorithm

1.0 Procedure

1.1 Algorithm

Begin Algorithm (Polar to Grid Conversion)

> Variables <

GBS	=	Grid box size
SF	=	Scaling Factor
DR	=	Size of range bin
ϕ	=	Elevation angle
θ	=	Azimuth
dx, dy	=	Scaled integer numbers
x, y	=	Scaled integer numbers
XH(1), YH(1)	=	Bin location in matrix
SF	=	2^{16}
x	=	1
y	=	1

Integer*4	X,Y, DX, DY
Integer*2	YH(2), XH(2)
Real*4 Constant,	dR, ϕ, θ, SF, GBS

1.0 Do For All (Elevations)

$$\text{Constant} = \frac{dR * \cos \phi}{GBS} * SF$$

1.1 Do For All (azimuths)

dx = integer [constant * sin θ]
dy = integer [constant * cosθ]

1.1.1 Do For All (Bins)

y = y + dy
x = x + dx

Box (XH(1), YH(1)) = ...

End do (bins)
End do (azimuths)
End do (elevation)

End Algorithm (Polar to Grid Conversion)

For layer composite products (23 and 24), echo tops (8), VIL (17), as well as Grid Group alert category (a) "products", assume that no polar cell maps into more than one Cartesian grid space; hence, no reverse mapping " or fill" processing is required.

4.4 Run Length Encoding

1.0 Procedure

An algorithm for run-length encoding a series S of N number, and generating a series V of K run-length values and a series L of K lengths

1.1 Algorithm

Begin Algorithm (Run Length Encoding)

```
1.0     K = 1
        V(K) = S(1)
        L(K) = 1

1.1     DO for I = 2 to NO
    1.1.1   IF (S(I) = S(I - 1)), THEN
                L(K) = L(K) + 1
                IF (L(K) = 16), THEN
                    L(K) = L(K) - 1
                    K=K+1
                    V(K) = S(I)
                    L(K) = 1
                END IF
            ELSE
                K=K+1
```

V(K) = S(I)
L(K) = 1
END IF
END DO

1.2 For J = 1 to K DO:
Bits (0:3) of Table (J) = L(K) **(See Note 2) (See Note 3)**
Bits (4:7) of Table (J) = V(K)

END DO

END Algorithm (Run Length Encoding)

4.5 Cross-Section

1.0 Procedure

1.1 Algorithm

Begin Algorithm (Cross-Section)

1.0 DO FOR ALL ELEVATIONS

1.1 DO FOR ALL RADIALS

1.1.1 DO FOR ALL REQUESTED GRIDS

1. 1. 1. 1 DETERMINE WHETHER RADIAL INTERSECTS GRID

1. 1. 1. 2 IF YES THEN
DETERMINE BREADTH OF PLANE INTERCEPTED BY BEAM (ASSUMED 1° WIDE)
DETERMINE SECTION OF RADIAL CORRESPONDING TO END POINTS OF INTERCEPTED SECTION OF PLANE
MAP APPROPRIATE BASE DATA FROM ALL BINS WITHIN INTERCEPTED SECTION OF RADIAL TO CORRESPONDING BOXES X-Z GRID
END IF
END DO
END DO
END DO

2.0 DO FOR ALL REQUESTED GRIDS

2.1 VERTICALLY INTERPOLATE BETWEEN LOWEST AND HIGHEST POINTS IN EACH COLUMN THAT HAD DATA TO THEM (TO FILL ANY HOLES)

2.2 RETAIN ANY POINTS OUTSIDE THIS VERTICAL RANGE FLAGGED TO

Note 2 : Table is an Integer*1 array.

Note 3 : Bit 0 is MSB.

INDICATE "NO DATA"

END DO

End Algorithm (Cross-Section)

4.6 Polar/LFM Conversion

4.6.1 Description

An algorithm is provided to identify the subset LFM grid box (I.J coordinates) corresponding to each display plane polar radar coordinate. Grid boxes for the 1/4, 1/16. and 1/40 LFM grids centered on the radar site are identified for each polar radar coordinate. The methodology used is to first translate each polar radar coordinate to a corresponding latitude, longitude in a manner consistent with the projection of background map data to the NEXRAD display plane. A polar stereographic projection of latitude, longitude to the LFM grid plane (tangent to the earth at the North Pole) is then performed to identify the grid box. The equations given apply for a northern hemispherical grid. However, these equations are useable to about 30 degrees latitude into the southern hemisphere. The grid is oriented such that the J-axis is parallel with the specified prime meridian (105 degrees W) and has a 1/4 LFM box size of 47.625 km true at 60 degrees latitude. For computational convenience. the I-axis for the grid is defined as positive to the right and the J-axis is positive down with the I, J coordinates of the North Pole set to 433.433. An earth's radius of 6371.221 km is used for the polar stereographic projection of latitude, longitude to the LFM grid plane.

It should be noted that the algorithm provided assumes a polar radar coordinate resolution of 1-km range by 1-degree azimuth with the azimuths centered on 0.5 degree. 1.5 degrees. etc, and range extending to 460 km. A range resolution of 2 km may be used with coarse resolution products where desirable to save throughput or simplify program design. Centering of the azimuths is arbitrary since the radar has no fixed azimuth position for any given radar beam. Grid boxes need not be defined beyond a range of 230 km for the 1/4 and 1/40 grids since the associated products are limited to this range.

4.6.2 Definition of Terms

L_S, λ_S	=	site latitude, longitude (deg)
L, λ	=	latitude, longitude of a given polar radar coordinate
R, B	=	range, bearing of a given polar radar coordinate
GI, GJ	=	global grid coordinates of a given polar radar coordinate in units of 1/4 LFM boxes (I positive to right, J positive down)
I(M), J(M)	=	local grid box number of a given polar radar coordinate for 1/4, 1/16, 1/40 LFM grids (M = 1,2,3, respectively). Upper left corner box of grid is numbered I,J = 1,1
I_P, J_P	=	global grid coordinates of North Pole in units of 1/4 LFM grid boxes (I_P, J_P = 433, 433)
GI_S, GJ_S	=	global grid coordinates of radar site in units of 1/4 LFM boxes

$I_S(M)$, $J_S(M)$ = global grid box number for box 0.0 of 1/4, 1/16, 1/40 local LFM grids (M = 1,2,3, respectively)

S = angular great circle distance between polar radar coordinate and radar site

K_C = grid scale factor = $\frac{2 * R (1 + \sin 60°)}{47.625}$ = 249.6348607

R_S = earth's radius (= 6,371.221 km)

4.6.3 Procedure

1.0 Compute sin L_S, cos L_S, cos (λ_S + 105°), sin (λ_S + 105°)

2.0 Compute Reference Grid Box Coordinates

$$GI_S = K_C \frac{\cos L_S}{1+\sin L_S} \sin(\lambda_S+105°)+I_P$$

$$GJ_S = K_C \frac{\cos L_S}{1+\sin L_S} \cos(\lambda_S+105°)+J_P$$

2.1 Compute Grid Box Number for box, 0.0 of local grids

$I_S(1)$ = INT(GI_S) - 7

$J_S(1)$ = INT(Gj_S) - 7

$I_S(2)$ = INT(GI_S) - 49

$J_S(2)$ = INT(Gj_S) - 49

$I_S(3)$ = INT(GI_S) - 66

$J_S(3)$ = INT(Gj_S) - 66

where I_S, J_S(1,2,3) are box numbers for 1/4, 1/16 and 1/40 LFM grids

3.0 Initialize Bearing

B = -0.5 degree

4.0 Do or each bearing (36O times)

B = B + 1 degree

R = -1/2 km

compute sin B, cos B

4.1 Do for each range (460 times)

R = R + 1km

$$\sin S = \frac{R}{6.380}\left(1 - \frac{135R}{6.380^2}\right)$$

$$\cos S = (1 - \sin^2 L)^{1/2}$$

$$\sin L = \sin L_S \cos S + \cos L_S \sin S \cos B$$

$$\cos L = (1 - \sin^2 L)^{1/2}$$

$$\sin \Delta\lambda = \sin S \sin B / \cos L$$

$$\cos \Delta\lambda = (1 - \sin^2 \Delta\lambda)^{1/2}$$

where: $\Delta\lambda = \lambda - \lambda_S$

$$R_L = K_C * \frac{\cos L}{1 + \sin L}$$

GI = $R_L \sin\Delta\lambda \cos(\lambda_S + 105°) + R_L \cos\Delta\lambda \sin(\lambda_S + 105°) + I_P$

GJ = $R_L \cos\Delta\lambda \cos(\lambda_S + 105°) - R_L \sin\Delta\lambda \sin(\lambda_S + 105°) + J_P$

4.1.1 Do for M = 1,2,3

$I(M) = INT[GI * K_A(M)] - I_S(M)$

$J(M) = INT[GJ * K_A(M)] - J_S(M)$

where K_A - 1,4,10 or M = 1,2,3 respectively

4.2 End of range computation

5.0 End of bearing computation

Notes:

1. For 1/4 LFM, box MM has I,J = 7,7
   For 1/16 LFM box MMA has I,J = 49,49

2. I,J values outside of the following ranges should be set to Not Applicable:
   For 1/4 LFM, I,J = 1,1 to 13,13
   For 1/16LFM, I,J = 1,1 to 100,100
   For 1/40 LFM, I,J = 1,1 to 131,131

3. To minimize storage requirements, local grid box numbers may be encoded into single numbers. To encode by row first, use $I + N_G(J-l)$. To encode by column first, use $J + N_G(I - 1)$
   N_G = 25, 100, 131 for 1/4, 1/16, 1/40 LFM grids respectively.

4.6.4 Additional Considerations

At longer ranges and at lower latitudes, the grid box size becomes smaller relative to the polar radar coordinate resolution. In such circumstances, it is possible that certain grid boxes will not be mapped to any polar radar coordinate at the completion of the above algorithm. For these cases, the closest coordinate will be used. Thus, for example, if box i,j on the local grid has no cell mapped, the range, azimuth of the closest coordinate is defined as:

$R = INT(RC_{ij}) + 0.5$

$B = INT(RC_{ij}) + 0.5$

where:

RC_{ij} = Range to center of box i,j (km)
RC_{ij} = Azimuth to center of box i,j in degrees (0° to 359.999°)

A method for computing RC_{ij}, θC_{ij} is given in Section 4.6.4.1.

4.6.4.1 Computation for Coordinates of Grid Box Center

Coordinates of box centers (relative to pole) in units of 1/4 LFM boxes can be computed as:

$CI_I = AI(M) + B(M) * i$

$CJ_J = AJ(M) + B(M) * j$

where M = 1,2,3 for 1/4, 1/16, 1/40 grids

i,j = 1,1 to 13,13 for 1/4 grid
i,j = 1,1 to 100,100 for 1/16 grid
i,j = 1,1 to 131,131 for 1/40 grid

AI, AJ, B are constants defined as:

AI(1) = $I_S(1) - I_P + 0.5$

AJ(1) = $J_S(1) - J_P + 0.5$

AI(2) = $(I_S(2) - 4I_P + 0.5) / 4$

AJ(2) = $(J_S(2) - 4J_P + 0.5) / 4$

AI(3) = $(I_S(3) - 10I_P + 0.5) / 10$

AJ(3) = $(J_S(3) - 10J_P + 0.5) / 10$

B(1), (2), (3) = 1, 1/4, 1/10, respectively

Latitude, longitude of box i,j center is then:

$$L_{IJ} = 90° - 2\ tan^{-1}[\frac{(CI_i^2 + CJ_j^2)^{1/2}}{K_C}]$$

$$\lambda_{IJ} = -105° + tan_{-1}[\frac{CI_i}{CJ_j}]$$ (use two-variable arctangent function)

Range, Az of box i,j center is then computed as follows:

sin S = $(A^2 + B^2)^{½}$

cos S = $(1 - \sin^2 S)^{½}$

where:.

A = $\cos L_{ij} \sin(\lambda_{ij} - \lambda_S)$

B = $\cos L_S \sin L_{ij} - \sin L_S \cos L_{ij} \cos (\lambda_{ij} - \lambda_S)$

Rc_{ij} = $(135 \sin S + 6.380) \sin S$

if $\sin S \geq 9.81 \times 10^{-6}$,

$$\theta C_{ij} = \tan^{-1}[\frac{cosL_{ij}\ \cos L_S\ \sin(\lambda_{ij} - \lambda_S)}{\sin L_{ij} - \sin L_S\ \cos\ S}]$$ (use two-variable arctangent function)

If $\sin S \leq 9.81 \times 10^{-6}$,

$$\theta C_{ij} = 0.0$$