U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
OFFICE OF SYSTEMS DEVELOPMENT
TECHNIQUES DEVELOPMENT LABORATORY

TDL OFFICE NOTE 94-1

AN IMPROVED ALGORITHM FOR DETERMINING THE GROUPS IN GRIB SECOND-ORDER PACKING

Harry R. Glahn

February 1994

AN IMPROVED ALGORITHM FOR DETERMINING THE GROUPS IN GRIB SECOND-ORDER PACKING

Harry R. Glahn

1. INTRODUCTION

The GRIdded Binary (GRIB) code contained in the Manual on Codes (WMO 1988) is becoming the international standard for exchanging meteorological gridpoint data and has been specified as the form of exchange of such data between the U.S. Government and the Automated Weather Interactive Processing System (AWIPS) contractor. As part of National Weather Service (NWS) risk reduction activities, many gridpoint products are being transmitted from the National Meteorological Center (NMC) for use at designated sites such as Norman, Oklahoma. Also, in order to get the benefit of such data before AWIPS is deployed and to ready the field for the use of such data, a limited amount of GRIBbed data is being transmitted on the Automated Field Operations and Services (AFOS) network; these data are offloaded at local stations to personal computers where they are decoded and used. Only a relatively small amount of such data can be handled by AFOS communication circuits, competition for the limited bandwidth being keen.

The Techniques Development Laboratory (TDL) is rebuilding its MOS developmental and implementation systems. The new software will include improved data file structures and more efficient packing procedures. In order to add to our experience in the use of GRIB and its implications, to verify our estimates for point to multipoint data transmission for AWIPS contained in the AWIPS Systems Requirements Specification, Appendix K, (U.S. Government 1989), and to determine whether or not the current GRIB products on AFOS could be slimmed down appreciably, 195 messages containing eta model (Mesinger et al. 1990) data were transferred from AFOS to the AWIPS Government Development Platform workstations for study. Each message contained several grids. In all, 917 grids from late October and early November 1993, were available for this test.

There are several features of the GRIB code that could be improved. Those that can be considered important relative to the efficiency (in terms of number of bits required) of the packing specifications are: (1) a bit map is used to specify the location and size of the 2nd order (complex) packing groups (WMO 1988, p. I-Bi-20), (2) full octets are used to define the group "widths" (WMO 1988, p. I-Bi-20), (3) the scanning modes do not include alternate row or column reversal (WMO 1988, p. I-Bi-12), and (4) spatial redundancy of meteorological data is not adequately addressed. Each of these has been discussed previously by Glahn (1993).

NMC has implemented the GRIB code for AFOS as contained in the Manual on Codes (WMO 1988), the only difference being that wherever desirable the field packed is not the original field, but rather 2nd order (spatial) differences of that field. In this way, the "field" is packed according to the GRIB specifications, but the field has first been reduced to 2nd order differences. This does necessitate sending the first original value and the first first order difference, which are inserted into an expanded Section 1--50 octets rather than the mandatory 28. The grids are with respect to the AWIPS Lambert map projection (Stackpole 1993, p. 24) but are currently 29 X 25 points at 160-km resolution--the so called thinned and trimmed grids.

This paper discusses the benefits of using an algorithm for determining whether or not to pack 2nd order differences, the performance of a specific algorithm for determining groups when complex packing is used, and the desirability of alternate row reversal in scanning.[1] This leads to a few recommendations, which are contained in Section 7.

2. NMC PACKING OF ETA MODEL FIELDS

Each of the 917 grids processed had been packed by NMC in one of three ways: (1) simple packing of the original values (no second order minimum removal), (2) complex packing of the original values, or (3) complex packing of 2nd order differences. The fields were generally heights, temperatures, U- and V-winds, relative humidities, and vertical velocities at several atmospheric levels, and also precipitation amounts, precipitable water, and lifted indices, all at several projections. In each case, the original data had been put into integer form by use of a suitable decimal scale factor. Overall statistics for each of these types of packing are shown in Table 1 in the column headed "NMC." The determination at NMC of the type of packing was made based on experience with the specific fields and packing algorithms being used. Of the 917 fields, 177 were simple-packed, 215 had original values complex packed, and 525 had 2nd order differences complex packed.

3. ALGORITHM FOR DETERMINING WHETHER TO USE 2ND ORDER DIFFERENCES

The 2nd order (spatial) differencing is described by Davis (1978) and Glahn (1992, p. 2). Many times the 2nd order differences will be smaller than the original values (even with the minima removed) and will require less bits for packing. But not always. Highly variable fields will not benefit from this process. One could, of course, pack both such fields and use the one requiring less bytes, but this seems unnecessarily machine-intensive. A rudimentary procedure will suffice--one that will clearly and cheaply give the correct answer in clear-cut situations. When the choice is difficult, it matters very little whether or not the absolutely correct choice is made.

Basically, the problem is to determine the average variation of each procedure (use of original values and use of 2nd order differences) on groups of size MINPK, where MINPK is the minimum group size to be used by the grouping algorithm. The algorithm in Appendix I does this, and is the method used in this study in making the choices as required. It also returns to the calling program the 2nd order differences (which are needed in the computation) in case they are to be used for packing.

[1]GRIB allows for scanning by row (left to right or vice versa) or column (top to bottom or vice versa). However, once started, the scanning is always in the same direction. That is, if the bottom row is scanned left to right, the second row is also scanned left to right. This makes for a large change between "adjacent" gridpoints--compared to real neighbors--at the end of each row. If one were to alternate the scanning direction on alternate rows, all "adjacent" gridpoints would be real neighbors. Stackpole (private communication) calls this process "boustrophedonic."

4. ALGORITHM FOR DETERMINING GROUPS FOR COMPLEX PACKING

Usually, the packing of a meteorological field will benefit by using the 2nd order minimum removal method called in the GRIB documentation "complex" packing. Even noisy fields with some "spikes" should benefit. Therefore, a good algorithm for determining groups is a necessity--one that will determine near-optimal groups without excessive computer time. The FORTRAN code shown in Appendix II basically works according to the diagram in Fig. 1.

The inputs to the algorithm, other than grid size, etc., are MINPK, the minimum size group to pack, and INC, the number of points to try adding to a group at a time. Ideally, INC would be 1 and should probably be used as such. MINPK should be in the range 10 to 20--and be less than half the "row" length (assuming scanning is by row).

5. EXAMPLE OF PERFORMANCE ON AN INDIVIDUAL FIELD

Fig. 2 shows the second order differences of an eta model field--not chosen for any particular reason, just the first one looked at (first field in message NMCGREE4X--24-h 200-mb temperature). Because the alternate row reversal scanning was not used, a pair of rather large 2nd order differences is associated with the end of each grid row.

Fig. 3 shows characteristics of NMC's packing of this field. It is noted that the minimum size of the 45 groups is 20, and 20 is consistently the group size. The total message length (hereafter, "message" will refer to a single packed grid) is 1022 bytes, and the number of bits required per gridpoint is 8.39 including GRIB Sections 0, 1, 2, 4, and 5, not just the gridpoints themselves.

The algorithm shown in Fig. 1 and Appendix II used with MINPK = 20 and INC = 1, gives the statistics shown in Fig. 4a. The number of groups is relatively small--33. The algorithm tends to put few points into groups requiring a large number of bits (7). Some of the individual spikes and pairs of large plus and minus values are isolated. Figs. 4b-4f show statistics for MINPK = 19, 18, 16, 14, and 12, respectively. Note the big improvement of message length of 989 to 927 bytes when going from MINPK = 20 to 19. This is due to the double spike (one up, one down) associated with each 39-gridpoint row--the minimum group size needs to be less than one-half the row size.

It is apparent that the groups tend to become oriented to the rows with the sum of two adjoining groups many times summing to the row length--39. As MINPK is reduced, the spikes tend to become isolated. In a few cases, a group is composed of only one value, the number of bits being listed as zero because all values in the group are the same (the single value) and the information is carried in the minimum itself. A message length of 906 bytes is attained with MINPK = 12; further reduction of MINPK was counterproductive.

When alternate rows are reversed, the 2nd order differences to pack are shown in Fig. 5. Here, the "spikes" associated with the ends of rows are eliminated. Figs. 6a-6d show the performance of the algorithm on this field for MINPK = 20, 18, 16, and 14, respectively. It can be noted that the groups still have some slight inclination to be "row oriented," but the association is with the meteorological features that persist from row to row rather than with the ends of rows. The message size at MINPK = 14 was 869.

6. PERFORMANCE ON 917 ETA MODEL FIELDS

Table 1 contains the results of various experiments. As stated earlier, it was found that of the 917 fields, 177 were simple-packed, 215 had the original values complex-packed, and 525 had 2nd order differences complex-packed. Results are presented in Table 1 according to this breakdown, as well as overall. Results are presented for MINPK = 20, 19, and 14 for both the original order of the points and for the alternate row reversal.

In all cases, it was assumed that complex packing would be better than simple packing. This may not always be the case (primarily because a bit map is used to define the groups), but no attempt was made to determine whether there were fields that would be better simple-packed.

The advantage of MINPK = 19 over 20 is seen for all three original groups, but marked improvement is evident for the 2nd order fields (1088 bytes per message versus 1171). Also, further reduction, albeit small, is possible when MINPK is reduced to 14. Overall, the improvement over the grids as they were transmitted on AFOS is (1107-1036)/1107 = 6.4%.

It is noted that the algorithm in Appendix I makes more decisions in favor of 2nd order differences when row reversal is used; the end-of-row problem is exacerbated when 2nd order differences are used. This is evident for all values of MINPK and for all types of original packing.

The use of alternate row reversal further reduces the average message length by (1036-986)/1086 = 4.8%. The total advantage of using the algorithms presented here and alternate row reversal is (1107-986)/1107 = 10.9%. This 10.9% advantage achieved on these 917 fields is about equally due to the algorithms in Appendices I and II and the alternate row reversal. The number of bits required per point was about 8.1.

The results above pertain to the AFOS implementation of the GRIB code. Because the WMO standard will likely be strictly followed for use in AWIPS, the 917 fields were again packed without 2nd order differences or row reversal. With MINPK = 14, the average bits required per point was 9.28 for the 160-km grids. In the AWIPS-era, much of the gridpoint data will be at 80-km or higher resolution. The high frequency detail that may be contained in these higher resolution fields will probably not greatly affect the number of bits required to represent the field in the GRIB code. In order to see how a reduced grid spacing might affect the results, the 917 fields were bilinearly interpolated to an 80-km (40-km) grid. The resulting average bits per point was 7.8 (6.8). These values are probably (only) slight underestimates (especially the 7.8) because of the interpolation.

7. CONCLUSIONS AND RECOMMENDATIONS

It is likely the basic GRIB as contained in WMO (1988) will continue to be used for packing gridpoint fields until WMO specifies a new version. However, the alternate row reversal can be implemented on AFOS in conjunction with the 2nd order differences--the field packed can just be defined appropriately and be run through the GRIBber as before. Only a few (like 10) lines of code are necessary to implement row reversal in each of the packing and unpacking routines. <u>It is recommended that row reversal be implemented.</u>

The algorithm for determining groups is important, and the one contained in Appendix II and tested here has been shown to give good results. It can be used as a module without rewrite. It is recommended that this algorithm be implemented.

The algorithm in Appendix I for determining whether or not to pack 2nd order differences operates efficiently in conjunction with the grouping algorithm. The decisions it makes are reasonably inexpensive and are adequate. Use of such an algorithm makes it unnecessary to predefine the type of packing by field. Because the information content per gridpoint will in the future change with model, projection, and gridpoint spacing, a decision made on-the-fly seems appropriate. It is recommended that an algorithm such as that in Appendix I be used.

The estimate given here is above 9 bits per gridpoint for the strict WMO standard GRIB packing. Of course, this is based on the degree of accuracy retained in the data; it is assumed here that the accuracy specified and used in the field test for the eta model data was adequate. When interpolated 80-km (40-km) grids rather than 160-km grids were packed, the average bits required per point was 7.8 (6.8). Because the preponderance of gridpoint data to be packed in the AWIPS-era will be at 80-, 40-, or even 20-km resolution, it is recommended that the 8-bit per point estimate in Appendix K be retained.

REFERENCES

Davis, R. A., 1978: Universal transmission format, Version 2.1. Unpublished manuscript, National Weather Service, NOAA, U.S. Department of Commerce, 22 pp.

Glahn, H. R., 1992: On the packing of gridpoint data for efficient transmission. TDL Office Note 92-11, National Weather Service, NOAA, U.S. Department of Commerce, 32 pp.

_____, 1993: An analysis of some features of the GRIB code. TDL Office Note 92-11, National Weather Service, NOAA, U.S. Department of Commerce, 42 pp.

Mesinger, R., T. L. Black, D. W. Plummer, and J. H. Ward, 1990: Eta model precipitation forecasts for a period including tropical storm Allison. Wea Forecasting, 5, 483-493.

Stackpole, J. D., 1993: GRIB, Edition 1, the WMO format for the storage of weather product information and the exchange of weather product messages in gridded binary form. National Meteorological Center Office Note 388, National Weather Service, NOAA, U.S. Department of Commerce, 75 pp.

U.S. Government, 1989: AWIPS-90 System Requirements Specification, Vol. I, Appendix K.

WMO, 1988: Manual on Codes, Vol. 1, Part B--Binary Codes. WMO No. 306, World Meteorological Organization, Geneva.

Table 1. Statistics for various packing experiments, shown for the three original types of packing of the 917 AFOS fields, and overall. "Message Length" is the average message length in bytes. "Bits/Point" are calculated over the complete message, including GRIB Sections 0, 1, 2, 4, and 5. "Points/Group" is the average number of grid-points per group. The algorithm in Appendix I gave the breakdowns shown, where "No. Original" means that the original values were packed, and "No. 2nd Order" means that the 2nd order differences were packed.

NMC Packing Method	Statistics	NMC	Original Order			Alternate Row Reversal		
			MINPK=20	MINPK=19	MINPK=14	MINPK=20	MINPK=19	MINPK=14
Simple	Message Length	1050	1074	1055	1050	1010	1010	1009
Original Values	Bits/Point	8.62	8.81	8.66	8.62	8.29	8.29	8.28
(177 Fields)	No. Groups	N/A	31.6	41.6	49.6	31.6	33.9	47.9
	Points/Group	N/A	30.8	23.4	19.6	30.9	28.8	20.4
	No. Original	N/A	18	21	16	2	2	2
	No. 2nd Order	N/A	159	156	161	175	175	175
Complex	Message Length	965	932	913	910	890	889	885
Original Values	Bits/Point	7.91	7.65	7.49	7.47	7.30	7.30	7.26
(215 Fields)	No. Groups	37.9	31.1	38.6	47.9	30.4	32.5	45.6
	Points/Group	25.7	31.2	25.3	20.3	32.0	30.0	21.4
	No. Original	N/A	120	116	121	86	86	90
	No. 2nd Order	N/A	95	99	94	129	129	125
Complex	Message Length	1185	1171	1088	1082	1028	1021	1019
2nd Order	Bits/Point	9.72	9.61	8.93	8.88	8.43	8.38	8.36
(525 Fields)	No. Groups	41.9	33.6	46.2	50.9	32.0	36.0	48.9
	Points/Group	23.3	29.0	21.1	19.1	30.5	27.1	19.9
	No. Original	N/A	19	18	9	0	0	0
	No. 2nd Order	N/A	506	507	516	525	525	525
Total	Message Length	1107	1096	1041	1036	992	988	986
(917 Fields)	Bits/Point	9.09	9.00	8.54	8.50	8.14	8.11	8.09
	No. Groups	40.7	32.6	43.6	50.0	31.5	34.8	47.9
	Points/Group	23.9	29.9	22.4	19.5	30.9	28.0	20.3
	No. Original	N/A	157	155	146	88	88	92
	No. 2nd Order	N/A	760	762	771	829	829	825

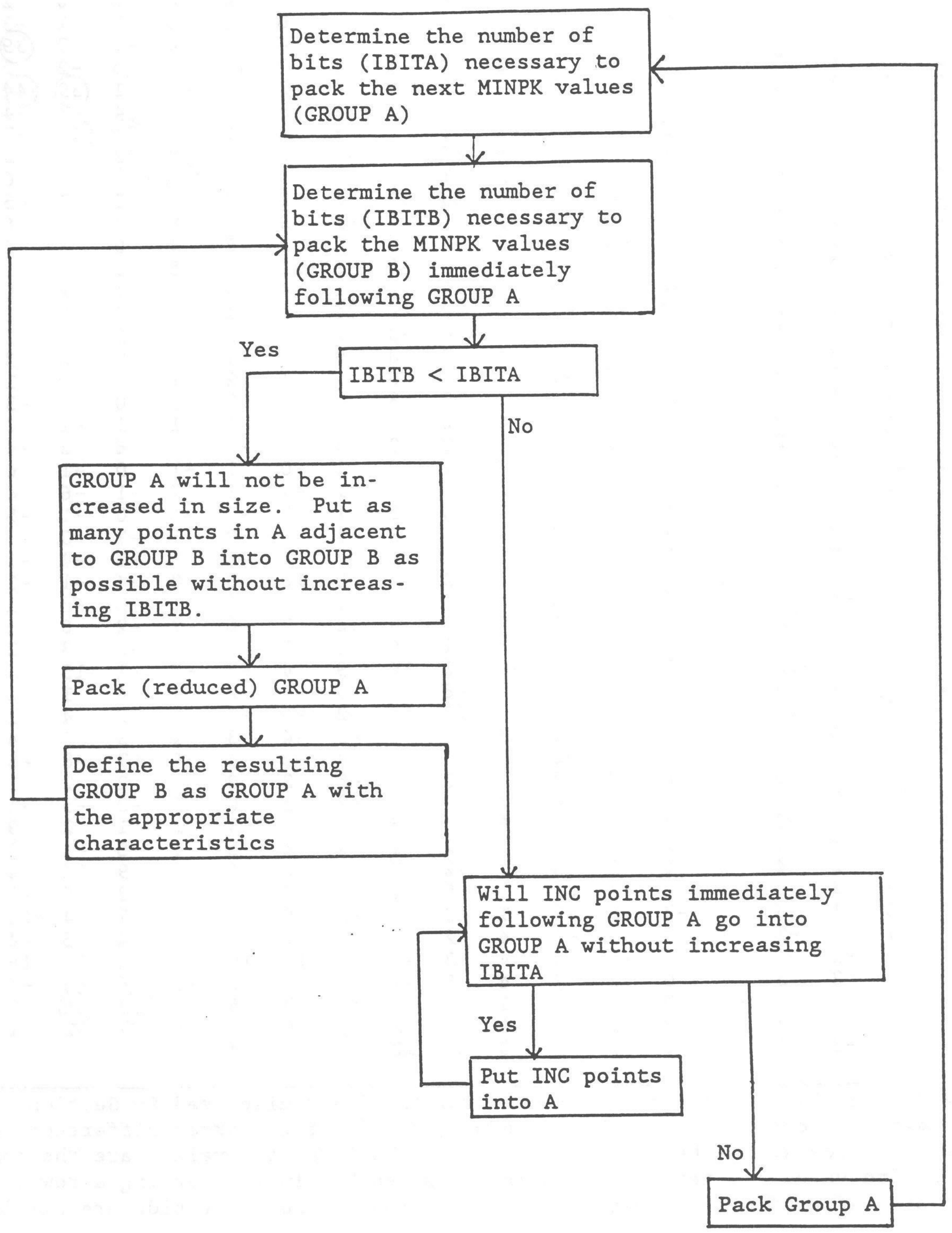
Determine the number of bits (IBITA) necessary to pack the next MINPK values (GROUP A)
Determine the number of bits (IBITB) necessary to pack the MINPK values (GROUP B) immediately following GROUP A
IBITB < IBITA
Yes
No
GROUP A will not be increased in size. Put as many points in A adjacent to GROUP B into GROUP B as possible without increasing IBITB.
Pack (reduced) GROUP A
Define the resulting GROUP B as GROUP A with the appropriate characteristics
Will INC points immediately following GROUP A go into GROUP A without increasing IBITA
Yes
Put INC points into A
No
Pack Group A

Figure 1. Algorithm for determining the groups for complex packing.

-3	-3	-3	-2	3	1	0	-3	0	1	1	2	0	1	-2	3	3	-7	-1	-2
1	-1	2	0	2	0	0	0	0	-1	0	0	-1	1	4	-4	-2	0	2	38
-39	-3	0	1	-3	-1	5	1	3	0	-1	3	4	0	1	-8	-2	1	-3	1
0	1	0	1	0	1	0	0	-1	0	1	-2	1	4	-1	-4	-2	3	39	-44
0	1	-5	-6	5	8	5	-1	1	-2	6	6	1	-7	-12	3	5	-2	-2	0
1	-1	1	-1	1	1	0	-1	0	-1	0	0	4	2	-2	-5	1	39	-44	-2
-4	-2	1	8	4	4	0	-1	2	7	8	-2	-16	-9	3	5	4	-1	-4	-1
1	1	0	0	0	3	-2	-1	0	-1	0	2	3	2	-6	-2	34	-42	-7	1
5	6	7	-1	1	0	-1	7	9	6	-7	-17	-11	2	5	8	2	-4	-2	-1
0	3	0	1	0	-1	-1	1	0	-2	1	4	1	-3	-4	23	-38	0	6	6
7	3	-2	-1	-4	4	10	8	1	-10	-13	-10	-1	5	7	6	0	-3	-5	0
5	0	-2	-2	0	2	3	0	0	-2	2	2	-1	-2	-1	-18	7	5	6	3
0	-1	-4	-2	8	11	7	-6	-10	-10	-6	-2	2	3	5	4	0	-3	-2	3
1	-4	-2	1	2	1	4	3	0	-1	0	0	-3	-22	10	5	3	4	1	-1
0	-3	-2	7	10	5	-4	-8	-8	-6	-2	-1	-1	4	4	2	1	-1	0	2
-3	-1	-2	1	2	7	4	-3	-1	-3	0	-1	-24	21	1	0	2	2	-1	2
-2	-4	5	5	6	2	-5	-8	-7	-3	0	-3	1	1	3	2	1	0	2	-2
-1	-1	2	3	3	0	-2	-1	-3	1	3	-10	9	-1	-1	4	1	-2	3	-1
-5	2	2	8	8	-6	-8	-6	-1	-3	-5	0	1	1	2	1	2	0	1	-1
0	1	1	-1	1	-1	1	-2	-1	4	15	-13	0	-1	3	-2	0	5	-3	-2
1	-2	6	13	-4	-11	-4	-2	-7	-4	2	3	0	2	0	1	1	-1	0	0
-1	-1	-2	3	2	1	1	-4	1	37	-32	0	3	-1	-2	2	6	-2	-4	4
-5	2	16	-1	-13	-5	-6	-5	0	3	1	0	1	0	1	-1	0	-1	0	-4
0	0	2	3	3	1	-3	-3	51	-45	1	4	1	-4	3	7	-1	-6	4	-4
-3	20	0	-13	-5	-7	-5	0	3	2	0	-1	-1	1	-2	-1	0	-1	-5	1
3	2	1	2	2	-3	1	52	-49	2	3	3	-3	1	8	1	-7	0	-3	-1
20	0	-12	-5	-7	-4	-1	1	2	2	-1	-1	-2	-2	-1	-1	-1	-4	-1	4
3	3	1	-2	0	8	35	-39	4	2	3	-3	0	10	3	-8	-4	-1	-1	19
3	-13	-5	-6	-2	0	-1	0	0	1	-1	-2	-3	-3	0	1	-3	-3	1	4
5	2	-4	5	10	3	-13	4	1	2	-3	0	12	7	-8	-7	-2	0	16	5
-15	-3	-2	-3	0	2	-3	-1	-1	1	-1	-6	-5	-2	2	0	-2	3	2	4
1	0	7	5	-23	14	2	1	1	-4	0	14	9	-4	-8	-5	-1	15	4	-15
-1	1	-5	-1	3	0	-5	0	3	2	-8	-8	-3	-1	-2	4	4	1	2	3
3	5	-1	-38	34	1	0	0	-4	0	12	13	-2	-6	-7	-2	12	4	-11	-4
3	-6	-3	4	2	-7	-1	6	5	-5	-9	-4	-5	-4	3	4	3	2	4	2
1	-1	-39	38	0	2	-1	-2	-2	10	13	1	-4	-5	-4	6	7	-8	-5	2
-5	-4	2	1	-6	-1	6	8	-2	-7	-5	-7	-5	2	2	4	2	2	1	2
4	-44	37	3	-1	3	-2	-2	6	12	2	0	-2	-5	0	7	-2	-5	-1	-3
-5	-1	0	-4	-2	6	7	2	-6	-6	-6	-5	1	3	3	1	1	2	9	7
-59	35	7	3	-1	1	-2	6	8	3	2	1	-4	-4	5	4	-5	-4	-3	-3
-3	-1	-3	-2	4	5	1	-3	-5	-7	-2	2	5	2	-1	2	9	11	4	-67
22	13	7	2	1	-2	6	7	1	1	5	-2	-7	5	5	-3	-3	-5	-5	-1
-2	-1	-3	1	3	0	-3	-5	-2	1	3	5	-2	0	9	13	7	-4	-60	12
3	13	7	3	-1	4	7	1	-1	4	2	-6	2	6	0	-4	-4	-5	-4	-1
0	-1	-1	-1	-1	-2	-3	-1	2	4	3	-3	5	15	10	-1	-5	-51	16	-13
5	13	8	2	3	7	0	-2	3	4	-5	-1	6	3	-1	-4	-6	-6	-1	1
1	-2	-3	-2	-2	-2	0	0	4	1	1	9	16	3	-5	0	-53	36	-22	-11
8	13	9	5	6	-2	-3	2	4	-2	-2	3	6	0	-1	-6	-7	-2	1	1
1	-4	-4	-3	-1	-2	0	3	1	3	12	12	-1	-2	4					

Figure 2. The 945 2nd order differences for the field discussed in Section 5 of the text. According to the NMC procedure, the first 2nd order difference is repeated twice at the beginning; therefore, the first two values are the same as the 3rd value. A few of the "spikes" caused by always starting a row at the left side after finishing with the previous row at the right side are circled.

```
JMIN( ) =
    60   63   28   23   55   23   51   25   50   29   54   49   57   45   43
    59   57   59   54   63   35   22   54   18   55   28   54   52   61   44
    29   56   28   59   23   62    8   60    0   62    7   16   54   14   60

LBIT( ) =
     4    6    6    7    5    7    5    7    5    6    5    5    4    6    5
     5    5    4    5    6    6    7    6    7    6    7    6    5    4    6
     6    6    7    5    7    5    7    4    7    5    7    7    5    7    5

NOV( ) =
    20   20   20   20   20   20   20   20   20   20   20   27   26   20   20
    20   32   25   20   20   20   20   20   20   20   20   20   31   23   20
    20   20   20   20   20   20   20   32   20   22   20   20   38   20   19

COMPLEX PACKING OF 2ND ORDER DIFFERENCES

SUMMARY
    VARIABLE =       11
    TYPE =          100
    LEVEL =         200
    MSG LENGTH =   1022  BYTES
    NO. POINTS =    975    VALUES ACTUALLY PACKED =   975
    BITS/PT =         8.39
    SCALING =         1
    NO. GROUPS =     45
    AVG PTS/GP =     21.67
```

Figure 3. Statistics for the GRIBbed 24-h, 200-mb temperature in the message NMCGREE4X. The number of points is 975, the message length is 1022 bytes, the number of groups is 45, the average number of points per group is 21.67, and the bits required per point is 8.39. The minimum value of each group, the number of bits required for that group, and the number of values in the group are shown in JMIN( ), LBIT( ), and NOV( ), respectively. The overall minimum of these 2nd order differences is -67, which must be subtracted from the values in Fig. 2 to get values related to the minimum values shown here. That is, -67 subtracted from the minimum in the first 20 values gives 60.

```
JMIN( ) =
     60   28   59   23   23   51   25   29   62   49   45   43   57   59   54
     35   61   18   55   28   52   61   29   56   28   23   61    0   60    7
     54   14   56

LBIT( ) =
      4    7    6    6    7    6    6    6    4    5    6    5    5    4    5
      7    6    7    6    6    5    4    6    6    7    6    6    7    5    7
      5    7    5

NOV( ) =
     39    2   38   38    2   38   38   17   23   38   12   28   52   25   39
     16   24   40   38   13   41   23   40   38    2   38   38   40   38   40
     38    3   36

SUMMARY
    VARIABLE =        11
    TYPE =           100
    LEVEL =          200
    MAX VALUE =      119
    MSG LENGTH =     989  BYTES
    NO. POINTS =     975                                  (a)  MINPK = 20
    BITS/PT =          8.11
    SCALING =          1
    NO. GROUPS =      33
    AVG PTS/GP =      29.55
```

```
JMIN( ) =
     60   28   59   23   62   23   58   25   63   29   62   49   57   45   59
     43   64   57   62   54   56   35   63   22   60   18   60   28   61   52
     61   44   59   29   58   28   60   23   61    8   60    0   60    7   61
     16   54   14   56   71
LBIT( ) =
      4    7    4    7    4    7    4    7    4    6    4    5    4    6    4
      6    3    5    3    5    4    7    3    7    4    7    4    7    4    5
      4    6    4    7    4    7    4    7    4    7    5    0    5    0    5
      0    5    7    5    0

NOV( ) =
     39    2   37   17   22   16   23   17   22   17   23   12   26   12   27
     17   22   17   21   15   25   19   20   16   23   16   23   16   22   17
     23   16   23   16   23   16   23    9   30    2   37    1   38    1   38
      1   38    3   35    1

SUMMARY
    VARIABLE =        11
    TYPE =           100
    LEVEL =          200
    MAX VALUE =      119
    MSG LENGTH =     927  BYTES
    NO. POINTS =     971                                  (b)  MINPK = 19
    BITS/PT =          7.61
    SCALING =          1
    NO. GROUPS =      50
    AVG PTS/GP =      19.50
```

Figure 4. Statistics for same field for which data are shown in Figs. 2 and 3. Figs. 4a through 4f show the results of the packing algorithm in Appendix II for MINPK = 20, 19, 18, 16, 14, and 12, respectively. See Fig. 3 caption for more details.

```
JMIN( ) =
    60   28   64   23   55   23   58   25   63   29   62   49   57   45   59
    43   64   57   62   54   56   35   61   22   60   18   60   28   61   52
    61   44   59   29   58   28   60   23   61    8   60    0   60    7   61
    16   61   14   60   65

LBIT( ) =
     4    7    3    7    5    7    4    7    4    6    4    5    4    6    4
     6    3    5    3    5    4    7    4    7    4    7    4    7    4    5
     4    6    4    7    4    7    4    7    4    7    4    7    4    7    4
     7    4    7    4    4

NOV( ) =
    39   17   19    5   37   16   23   17   22   17   23   12   26   12   27
    17   22   17   21   15   25   16   23   16   23   16   23   16   22   17
    23   16   23   16   23   16   23    9   30    2   34    6   32    8   31
     9   30   11   25    7

SUMMARY
    VARIABLE =       11
    TYPE =          100
    LEVEL =         200
    MAX VALUE =     119
    MSG LENGTH =    922 BYTES                         (c) MINPK = 18
    NO. POINTS =    975
    BITS/PT =         7.57
    SCALING =         1
    NO. GROUPS =     50
    AVG PTS/GP =     19.50

JMIN( ) =
    64   60   28   59   23   55   23   58   25   50   29   49   57   45   59
    43   59   57   59   54   56   35   61   22   60   18   55   65   28   52
    61   44   59   29   58   28   60   23   61    8   60    0   60    7   61
    16   61   14   60   65

LBIT( ) =
     3    4    7    4    7    5    7    4    7    5    6    5    4    6    4
     6    4    5    4    5    4    7    4    7    4    7    4    6    6    5
     4    6    4    7    4    7    4    7    4    7    4    7    4    7    4
     7    4    7    4    4

NOV( ) =
    17   22    2   37    2   37   16   23    2   37    2   50   26   12   27
     2   37   14   25   14   25   16   23   16   23   14   18    8   13   41
    23   16   23   16   23   16   23    9   30    2   34    6   32    8   31
     9   30   11   23    9

SUMMARY
    VARIABLE =       11
    TYPE =          100
    LEVEL =         200
    MAX VALUE =     119
    MSG LENGTH =    919 BYTES                         (d) MINPK = 16
    NO. POINTS =    975
    BITS/PT =         7.54
    SCALING =         1
    NO. GROUPS =     50
    AVG PTS/GP =     19.50
```

Figure 4. (Continued)

```
JMIN( ) =
      64    60    28    59    23    55    23    51    25    50    29    49    57    45    59
      43    59    57    59    54    56    35    54    22    54    18    55    65    28    61
      52    61    44    52    29    56    28    59    23    61     8    60     0    60     7
      61    16    61    14    60    65

LBIT( ) =
       3     4     7     4     7     5     7     5     7     5     6     5     4     6     4
       6     4     5     4     5     4     7     5     7     5     7     4     6     6     4
       5     4     0     5     7     5     7     5     7     4     7     4     7     4     7
       4     7     4     7     4     4

NOV( ) =
      17    22     2    37     2    37     2    37     2    37     2    50    26    12    27
       2    37    14    25    14    25     2    37    14    25    14    18     8    15    22
      17    23     1    38     2    37     2    37     9    30     2    34     6    32     8
      31     9    30    11    24     8

SUMMARY
    VARIABLE =        11
    TYPE =           100
    LEVEL =          200
    MAX VALUE =      119
    MSG LENGTH =     918 BYTES                                    (e) MINPK = 14
    NO. POINTS =     974
    BITS/PT =          7.53
    SCALING =          1
    NO. GROUPS =      51
    AVG PTS/GP =      19.12

JMIN( ) =
      64    60    28    59    64    23    55    65    23    51    58    25    50    64    29
      54    62    49    57    45    59    43    59    64    57    59    65    54    56    63
      35    54    63    22    54    60    18    55    60   102    28    61    52    61    44
      52    59    29    56    58    28    59    60    23    61     8    60     0    60     7
      61    64    16    62    61    14    56    60    65

LBIT( ) =
       3     4     7     4     3     7     5     2     7     5     4     7     5     3     6
       5     4     5     4     6     4     6     4     3     5     4     2     5     4     3
       7     5     3     7     6     4     7     6     4     0     6     4     5     4     0
       5     4     7     5     4     7     4     4     7     4     7     4     7     4     7
       4     3     7     4     3     7     5     4     4

NOV( ) =
      17    22     2    15    19     5    17    15     7    14    23     2    20    16     3
      15    23    12    26    12    27     2    15    22    14    12    12    15     5    20
       2    17    20     2    14    23     2    14    23     1    15    22    17    23     1
      15    23     2    14    23     9    16    14     9    30     2    34     6    32     8
      17    13    10    12    14    11     4    25     7

SUMMARY
    VARIABLE =        11
    TYPE =           100
    LEVEL =          200
    MAX VALUE =      119
    MSG LENGTH =     906 BYTES                                    (f) MINPK = 12
    NO. POINTS =     973
    BITS/PT =          7.43
    SCALING =          1
    NO. GROUPS =      69
    AVG PTS/GP =      14.13
```

Figure 4. (Continued)

-3	-3	-3	-2	3	1	0	-3	0	1	1	2	0	1	-2	3	3	-7	-1	-2
1	-1	2	0	2	0	0	0	0	-1	0	0	-1	1	4	-4	-2	0	2	3
-1	3	-2	-4	-1	4	1	-2	1	0	-1	0	0	1	0	1	0	1	0	1
-3	1	-2	-8	1	0	4	3	-1	0	3	1	5	-1	-3	1	0	-3	1	-9
0	1	-5	-6	5	8	5	-1	1	-2	6	6	1	-7	-12	3	5	-2	-2	0
1	-1	1	-1	1	1	0	-1	0	-1	0	0	4	2	-2	-5	1	3	2	-2
-6	2	3	2	0	-1	0	-1	-2	3	0	0	0	1	1	-1	-4	-1	4	5
3	-9	-16	-2	8	7	2	-1	0	4	4	8	1	-2	-4	-2	-12	-6	-7	1
5	6	7	-1	1	0	-1	7	9	6	-7	-17	-11	2	5	8	2	-4	-2	-1
0	3	0	1	0	-1	-1	1	0	-2	1	4	1	-3	-4	5	-3	-2	-1	2
2	-2	0	0	3	2	0	-2	-2	0	5	0	-5	-3	0	6	7	5	-1	-10
-13	-10	1	8	10	4	-4	-1	-2	3	7	6	6	0	-36	0	7	5	6	3
0	-1	-4	-2	8	11	7	-6	-10	-10	-6	-2	2	3	5	4	0	-3	-2	3
1	-4	-2	1	2	1	4	3	0	-1	0	0	-3	-11	8	-1	0	-3	-1	-3
4	7	2	1	-2	-1	-3	2	0	-1	1	2	4	4	-1	-1	-2	-6	-8	-8
-4	5	10	7	-2	-3	0	-1	1	4	3	5	-22	10	1	0	2	2	-1	2
-2	-4	5	5	6	2	-5	-8	-7	-3	0	-3	1	1	3	2	1	0	2	-2
-1	-1	2	3	3	0	-2	-1	-3	1	3	-27	25	4	-1	-2	1	-1	1	-1
1	1	0	-1	1	0	2	1	2	1	1	0	-5	-3	-1	-6	-8	-6	8	8
2	2	-5	-1	3	-2	1	4	-1	-1	-1	4	0	-1	3	-2	0	5	-3	-2
1	-2	6	13	-4	-11	-4	-2	-7	-4	2	3	0	2	0	1	1	-1	0	0
-1	-1	-2	3	2	1	1	-4	1	-9	20	-3	-3	1	3	3	2	0	0	-4
0	-1	0	-1	1	0	1	0	1	3	0	-5	-6	-5	-13	-1	16	2	-5	4
-4	-2	6	2	-2	-1	3	0	-1	1	1	4	1	-4	3	7	-1	-6	4	-4
-3	20	0	-13	-5	-7	-5	0	3	2	0	-1	-1	1	-2	-1	0	-1	-5	1
3	2	1	2	2	-3	1	13	-9	8	0	-2	1	3	3	4	-1	-4	-1	-1
-1	-2	-2	-1	-1	2	2	1	-1	-4	-7	-5	-12	0	20	-1	-3	0	-7	1
8	1	-3	3	3	2	-5	0	4	2	3	-3	0	10	3	-8	-4	-1	-1	19
3	-13	-5	-6	-2	0	-1	0	0	1	-1	-2	-3	-3	0	1	-3	-3	1	4
5	2	-4	5	10	20	-34	5	7	0	1	4	2	3	-2	0	2	-2	-5	-6
-1	1	-1	-1	-3	2	0	-3	-2	-3	-15	5	16	0	-2	-7	-8	7	12	0
-3	2	1	4	-2	-3	2	1	1	-4	0	14	9	-4	-8	-5	-1	15	4	-15
-1	1	-5	-1	3	0	-5	0	3	2	-8	-8	-3	-1	-2	4	4	1	2	3
3	5	-1	1	-3	-1	1	2	4	2	3	4	3	-4	-5	-4	-9	-5	5	6
-1	-7	2	4	-3	-6	3	-4	-11	4	12	-2	-7	-6	-2	13	12	0	-4	0
0	1	-2	-1	0	2	-1	-2	-2	10	13	1	-4	-5	-4	6	7	-8	-5	2
-5	-4	2	1	-6	-1	6	8	-2	-7	-5	-7	-5	2	2	4	2	2	1	2
4	11	-25	7	9	2	1	1	3	3	1	-5	-6	-6	-6	2	7	6	-2	-4
0	-1	-5	-3	-1	-5	-2	7	0	-5	-2	0	2	12	6	-2	-2	3	-1	3
3	-20	7	3	-1	1	-2	6	8	3	2	1	-4	-4	5	4	-5	-4	-3	-3
-3	-1	-3	-2	4	5	1	-3	-5	-7	-2	2	5	2	-1	2	9	11	4	-2
-38	-4	7	13	9	0	-2	5	3	1	-2	-5	-3	0	3	1	-3	-1	-2	-1
-5	-5	-3	-3	5	5	-7	-2	5	1	1	7	6	-2	1	2	7	13	0	-53
3	13	7	3	-1	4	7	1	-1	4	2	-6	2	6	0	-4	-4	-5	-4	-1
0	-1	-1	-1	-1	-2	-3	-1	2	4	3	-3	5	15	10	-1	-5	-16	-20	0
-5	3	16	9	1	1	4	0	0	-2	-2	-2	-3	-2	1	1	-1	-6	-6	-4
-1	3	6	-1	-5	4	3	-2	0	7	3	2	8	13	5	-13	-17	1	-22	-11
8	13	9	5	6	-2	-3	2	4	-2	-2	3	6	0	-1	-6	-7	-2	1	1
1	-4	-4	-3	-1	-2	0	3	1	3	12	12	-1	-2	4					

Figure 5. The 945 2nd order differences for the field discussed in Section 5 of the text with alternate rows reversed.

```
JMIN( ) =
       44   41   44   36   48   17   42   47   31   45   26   45   42   47   40
       73   40   45   40   19   38   44   42   46   28   47   33   15    0   33
       40   31   42

LBIT( ) =
        4    5    4    5    4    6    5    4    6    4    6    5    4    5    5
        0    5    5    5    6    5    4    5    4    6    5    5    6    0    6
        5    5    5

NOV( ) =
       85   10   47   31   46   16   39   24   16   37    2   51   26   24   27
        1   52   27   24    2   53   48   30   23    2   38   39   39    1   39
       37    3   36

0SUMMARY
    VARIABLE =       11
    TYPE =          100
    LEVEL =         200
    MAX VALUE =      78
    MSG LENGTH =    890 BYTES
    NO. POINTS =    973                                   (a) MINPK = 20
    BITS/PT =         7.30
    SCALING =         1
    NO. GROUPS =     33
    AVG PTS/GP =     29.55
```

```
JMIN( ) =
       46   50   44   41   44   36   48   17   49   43   42   31   50   26   45
       42   47   40   73   40   45   40   19   38   44   42   46   28   47   33
       15   46    0   47   33   47   31   46   51

LBIT( ) =
        4    3    4    5    4    5    4    6    4    4    5    6    3    6    5
        4    5    5    0    5    5    4    6    5    4    5    4    6    5    5
        6    4    7    4    6    4    6    4    4

NOV( ) =
       44   19   22   10   47   31   46   16   12   26   39   17   22    2   51
       26   24   27    1   52   27   18    8   53   48   30   23    2   38   39
        5   32    5   31   10   30   10   26    6

SUMMARY
    VARIABLE =       11
    TYPE =          100
    LEVEL =         200
    MAX VALUE =      78
    MSG LENGTH =    875 BYTES
    NO. POINTS =    974                                   (b) MINPK = 18
    BITS/PT =         7.18
    SCALING =         1
    NO. GROUPS =     39
    AVG PTS/GP =     25.00
```

Figure 6. Statistics for same field for which data are shown in Fig. 5 for MINPK = 20, 18, 16, and 14.

```
JMIN( ) =
     50   46   50   44   41   47   49   36   48   17   49   43   42   31   45
     26   45   47   42   49   40   73   40   46   41   40   19   38   44   42
     45   28   47   33   46   15   46    0   47   33   47   31   46   51

LBIT( ) =
      3    4    3    4    5    4    3    5    4    6    4    4    5    6    4
      6    4    4    5    3    5    0    5    4    6    5    6    5    4    5
      4    6    4    6    4    6    4    7    4    6    4    6    4    4

NOV( ) =
     17   27   19   22   10   26   17   35   46   16   12   26   39    2   37
      2   24   26   28   20   30    1   27   23   28   25    2   53   48   23
     16   16   30    9   34    9   32    5   31   10   30   10   23    9

SUMMARY
     VARIABLE =       11
     TYPE =          100
     LEVEL =         200
     MAX VALUE =      78
     MSG LENGTH =    870 BYTES                            (c) MINPK = 16
     NO. POINTS =    974
     BITS/PT =         7.14
     SCALING =         1
     NO. GROUPS =     44
     AVG PTS/GP =     22.16

JMIN( ) =
     50   46   50   44   41   51   44   36   49   50   40   17   43   42   45
     31   45   50   26   45   47   42   49   40   47   73   40   46   41   45
     40   19   47   38   44   42   45   46   28   47   33   46   15   48    0
     47   48   33   47   31   46   51

LBIT( ) =
      3    4    3    4    5    3    4    5    4    3    5    6    4    5    4
      6    4    3    6    4    4    5    3    5    4    0    5    4    0    5
      4    6    4    5    4    5    4    4    6    4    6    4    6    4    7
      5    3    6    4    6    4    4

NOV( ) =
     17   27   19   22   10   20   27   31   23   14   24   13   26    2   27
     12   15   22    2   24   26   28   20    6   24    1   27   23    1   28
     18    8   23   30   48   23   16   14    2   30    9   34    8   22   14
     14   15   14   30   10   24    8

0SUMMARY
     VARIABLE =       11
     TYPE =          100
     LEVEL =         200
     MAX VALUE =      78
     MSG LENGTH =    869 BYTES                            (d) MINPK = 14
     NO. POINTS =    973
     BITS/PT =         7.13
     SCALING =         1
     NO. GROUPS =     52
     AVG PTS/GP =     18.75
```

Figure 6. (Continued)

APPENDIX I

Algorithm to Determine Whether to Pack 2nd Order Differences

This appendix contains a listing of the FORTRAN code for an algorithm to determine whether or not to pack the original values or 2nd order differences. It is very nearly like the subroutine with the same name in Glahn (1993, pp. 21-23), but agrees with NMC's practice of using the first two values in the 2nd order difference field as duplicates of the third point.

```
      SUBROUTINE GRIBPR(KFIL10,KFIL12,IC,IA,IB,IDIM,NXY,MINPK,
     1                  IMIN,IFIRST,IFOD,SECOND)
C
C        JANUARY 1994   GLAHN   TDL   HP
C
C        PURPOSE
C            USED TO DETERMINE WHETHER TO USE SECOND ORDER
C            DIFFERENCES OR ORIGINAL VALUES TO PACK.  ORIGINAL VALUES
C            ARE INDICATED WHEN THE AVERAGE RANGE OF CONSECUTIVE GROUPS
C            OF SIZE MINPK OF THE SECOND ORDER DIFFERENCES IS
C            LARGER THAN THE AVERAGE RANGE OF CONSECUTIVE GROUPS OF
C            SIZE MINPK OF THE ORIGINAL VALUES.  VALUES AT THE END OF
C            THE ARRAY ARE NOT USED IN THE COMPUTATIONS UNLESS THEY ARE
C            IN A FULL-SIZE GROUP.
C
C        DATA SET USE
C           KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C           KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C        VARIABLES
C              KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C              KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C               IC(K) = HOLDS THE NXY ORIGINAL VALUES ON INPUT (K=1,IDIM).
C                       HOLDS THE NXY SECOND ORDER DIFFERENCES ON
C                       OUTPUT WHEN SECOND ORDER DIFFERENCES ARE TO BE
C                       USED.  IN THAT CASE, SECOND IS .TRUE. AND THE
C                       FIRST 2 VALUES ARE DUMMY.  (INPUT-OUTPUT)
C               IA(K) = WORK ARRAY (K=1,IDIM).  (INTERNAL)
C               IB(K) = WORK ARRAY (K=1,IDIM).  (INTERNAL)
C                IDIM = DIMENSION OF IC( ), IB( ), AND IA( ).  (INPUT)
C                 NXY = NUMBER OF VALUES IN IC( ) ON INPUT.  ON RETURN,
C                       NXY WILL ALSO BE THE NUMBER OF VALUES IN IC( ).
C                       (INPUT)
C               MINPK = INCREMENT IN WHICH RANGES WILL BE COMPUTED.  (INPUT)
C                IMIN = WHEN SECOND ORDER DIFFERENCES ARE USED, IMIN
C                       IS THE MINIMUM OF THEM.  (OUTPUT)
C              IFIRST = WHEN SECOND ORDER DIFFERENCES ARE USED, IFIRST
C                       IS THE FIRST ORIGINAL VALUE.  (OUTPUT)
C                IFOD = WHEN SECOND ORDER DIFFERENCES ARE USED, IFOD
C                       IS THE FIRST FIRST ORDER DIFFERENCE.  (OUPUT)
C              SECOND = TRUE (FALSE) WHEN SECOND ORDER DIFFERENCES
C                       ARE (ARE NOT) USED.  (OUTPUT)
C
C        NON SYSTEM SUBROUTINES CALLED
C           NONE
```

```
C
      LOGICAL SECOND
C
      DIMENSION IC(IDIM),IA(IDIM),IB(IDIM)
C
C        COMUTE FIRST ORDER DIFFERENCES.
C
      IFIRST=IC(1)
      IFOD=IC(2)-IC(1)
C
      DO 120 K=1,NXY-1
      IA(K)=IC(K+1)-IC(K)
 120  CONTINUE
C
C        COMPUTE SECOND ORDER DIFFERENCES
C
      IMIN=999999
C
      DO 130 K=1,NXY-2
      IB(K)=IA(K+1)-IA(K)
      IF(IB(K).LT.IMIN)IMIN=IB(K)
 130  CONTINUE
C
C        COMPUTE AVERAGE RANGE OF NXY ORIGINAL VALUES IN INCREMENTS OF
C        MINPK.
C
      SUMR=0
      KOUNT=0
C
      DO 140 K=1,NXY,MINPK
      JMIN=999999
      JMAX=-999999
      IF(K+MINPK-1.GT.NXY)GO TO 140
C        THE LAST GROUP MAY BE VERY SMALL AND NOT BE REPRESENTATIVE OF
C        THE RANGE.
C
      DO 135 J=K,K+MINPK-1
      IF(IC(J).GT.JMAX)JMAX=IC(J)
      IF(IC(J).LT.JMIN)JMIN=IC(J)
 135  CONTINUE
C
      KOUNT=KOUNT+1
      IRANGE=JMAX-JMIN
      SUMR=SUMR+IRANGE
 140  CONTINUE
C
      AVGR=99999.
      IF(KOUNT.NE.0)AVGR=SUMR/KOUNT
```

```
C
C        COMPUTE AVERAGE RANGE OF NXY-2 2ND ORDER VALUES IN INCREMENTS OF
C        MINPK.
C
      SUMR=0
      KOUNT=0
C
      DO 150 K=1,NXY-2,MINPK
      JMIN=999999
      JMAX=-999999
      IF(K+MINPK-1.GT.NXY-2)GO TO 150
C        THE LAST GROUP MAY BE VERY SMALL AND NOT BE REPRESENTATIVE OF
C        THE RANGE.
C
      DO 145 J=K,K+MINPK-1
      IF(IB(J).GT.JMAX)JMAX=IB(J)
      IF(IB(J).LT.JMIN)JMIN=IB(J)
 145  CONTINUE
C
      KOUNT=KOUNT+1
      IRANGE=JMAX-JMIN
      SUMR=SUMR+IRANGE
 150  CONTINUE
C
      AVGR2=99999.
      IF(KOUNT.NE.0)AVGR2=SUMR/KOUNT
C
      SECOND=.FALSE.
      WRITE(KFIL12,155)AVGR,AVGR2,IMIN
 155  FORMAT('0AVERAGE RANGE OF ORIGINAL SCALED VALUES =   'F10.2/
     1       ' AVERAGE RANGE OF SECOND ORDER DIFFERENCES ='F10.2/
     2       ' MINIMUM OF SECOND ORDER DIFFERENCES =     'I10)
      IF(AVGR2.GE.AVGR)GO TO 300
C        SECOND ORDER DIFFERENCES WILL BE PACKED.  IN AGREEMENT WITH NMC
C        PROCEDURES, NXY (RATHER THAN NXY-2) VALUES WILL BE PACKED;
C        THE FIRST TWO VALUES WILL BE DUMMY.
C
D     WRITE(KFIL12,160)
D160  FORMAT('0SECOND ORDER DIFFERENCES WILL BE PACKED')
C
      IC(1)=IB(1)
      IC(2)=IB(1)
C
      DO 200 K=1,NXY-2
      IC(K+2)=IB(K)
 200  CONTINUE
C
      SECOND=.TRUE.
 300  CONTINUE
      RETURN
      END
```

APPENDIX II

Algorithm for Determining Groups in GRIB Complex Packing

This appendix contains a listing of the FORTRAN code for the algorithm to determine the groups for complex packing. This subroutine, BGVRBL, is much like the routine with the same name contained in Glahn (1993, pp. 24-27). However it has been improved to contain a "backward look" feature. It is noted this algorithm will isolate long strings of a constant value, but strings of ≤ MINPK will not be isolated. As a worst case, 2*MINPK-2 values of a constant value could exist without being isolated. This is generally of concern only for precipitation fields in which long strings of zeros can exist. The extra computation required to search for shorter strings of a constant value is probably not cost effective.

```
      SUBROUTINE BGVRBL(KFIL10,KFIL12,IC,NDP,MINPK,INC,
     1                  JMIN,JMAX,IBIT,LBIT,NOV,NDQ,LX)
C
C        JANUARY 1994   GLAHN    TDL    HP
C
C        PURPOSE
C            TO DETERMINE GROUPS OF VARIABLE SIZE, BUT AT LEAST OF
C            SIZE MINPK, AND THE ASSOCIATED MAX AND MIN OF EACH GROUP,
C            THE NUMBER OF BITS NECESSARY TO PACK EACH GROUP, AND THE
C            NUMBER OF VALUES IN EACH GROUP.  THE ROUTINE IS DESIGNED
C            TO DETERMINE THE GROUPS SUCH THAT A SMALL NUMBER OF BITS
C            IS NECESSARY TO PACK THE DATA WITHOUT EXCESSIVE
C            COMPUTATIONS.  IF ALL VALUES IN THE GROUP ARE ZERO, THE
C            NUMBER OF BITS TO USE IN PACKING IS DEFINED AS ZERO.
C            ALL VARIABLES ARE INTEGER.  EVEN THOUGH THE GROUPS ARE
C            INITIALLY OF SIZE MINPK OR LARGER, AN ADJUSTMENT BETWEEN
C            TWO GROUPS (THE LOOKBACK PROCEDURE) MAY MAKE A GROUP
C            SMALLER THAN MINPK.  THE CONTROL ON GROUP SIZE IS THAT
C            THE SUM OF THE SIZES OF THE TWO CONSECUTIVE GROUPS, EACH OF
C            SIZE MINPK OR LARGER, REMAINS THE SAME.
C
C        DATA SET USE
C           KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C           KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C        VARIABLES IN CALL SEQUENCE
C              KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C              KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C                IC( ) = ARRAY TO HOLD DATA FOR PACKING.  THE VALUES
C                        DO NOT HAVE TO BE POSITIVE AT THIS POINT, BUT
C                        MUST BE IN THE RANGE -9999999 TO +9999999.
C                        THESE INTEGER VALUES WILL BE RETAINED EXACTLY
C                        THROUGH PACKING AND UNPACKING.  (INPUT)
C                  NDP = NUMBER OF VALUES IN IC( ).  ALSO TREATED
C                        AS ITS DIMENSION.  (INPUT)
C                MINPK = THE MINIMUM SIZE OF EACH GROUP, EXCEPT POSSIBLY
C                        THE LAST ONE.  (INPUT)
C                  INC = THE NUMBER OF VALUES TO ADD TO AN ALREADY
C                        EXISTING GROUP IN DETERMINING WHETHER OR NOT
C                        TO START A NEW GROUP.  IDEALLY, THIS WOULD BE
```

```
C                    1, BUT EACH TIME INC VALUES ARE ATTEMPTED, THE
C                    MAX AND MIN OF THE NEXT MINPK VALUES MUST BE
C                    FOUND.  THIS IS "A LOOP WITHIN A LOOP," AND
C                    A SLIGHTLY LARGER VALUE MAY GIVE ABOUT AS GOOD
C                    RESULTS WITH SLIGHTLY LESS COMPUTATIONAL TIME.
C                    IF INC IS LE 0, 1 IS USED, AND A DIAGNOSTIC IS
C                    OUTPUT.  (INPUT)
C          JMIN(J) = THE MINIMUM OF EACH GROUP (J=1,LX).  (OUTPUT)
C          JMAX(J) = THE MAXIMUM OF EACH GROUP (J=1,LX).  (OUTPUT)
C             IBIT = THE NUMBER OF BITS NECESSARY TO PACK JMIN(J)
C                    VALUES, J=1,LX.  (OUTPUT)
C          LBIT(J) = THE NUMBER OF BITS NECESSARY TO PACK EACH GROUP
C                    (J=1,LX).  IT IS ASSUMED THE MINIMUM OF EACH
C                    GROUP WILL BE REMOVED BEFORE PACKING, AND THE
C                    VALUES TO PACK WILL, THEREFORE, ALL BE POSITIVE.
C                    HOWEVER, IC( ) DOES NOT NECESSARILY CONTAIN
C                    ALL POSITIVE VALUES.  IF THE OVERALL MINIMUM
C                    HAS BEEN REMOVED, THEN IC( ) WILL CONTAIN
C                    ONLY POSITIVE VALUES.  (OUTPUT)
C           NOV(J) = THE NUMBER OF VALUES IN EACH GROUP (J=1,LX).  (OUTPUT)
C              NDQ = THE DIMENSION OF JMIN( ), JMAX( ), LBIT( ), AND
C                    NOV( ).  (INPUT)
C               LX = THE NUMBER OF GROUPS DETERMINED.  (OUTPUT)
C
C        INTERNAL VARIABLES
C             KINC = WORKING COPY OF INC.  MAY BE MODIFIED.
C           ISKIPA = 0 UNLESS GROUP B BECOMES GROUP A.  THEN, ISKIPA
C                    IS SET TO 1 TO SKIP A PORTION OF CODE.
C             MINA = MINIMUM VALUE IN GROUP A.
C             MAXA = MAXIMUM VALUE IN GROUP A.
C            NENDA = THE PLACE IN IC( ) WHERE GROUP A ENDS.
C           KSTART = THE PLACE IN IC( ) WHERE GROUP A STARTS.
C            IBITA = NUMBER OF BITS NEEDED TO HOLD VALUES IN GROUP A.
C             MINB = MINUMUM VALUE IN GROUP B.
C             MAXB = MAXIMUM VALUE IN GROUP B.
C            NENDB = THE PLACE IN IC( ) WHERE GROUP B ENDS.
C            IBITB = NUMBER OF BITS NEEDED TO HOLD VALUES IN GROUP B.
C             MINC = MINUMUM VALUE IN GROUP C.
C             MAXC = MAXIMUM VALUE IN GROUP C.
C           KTOTAL = COUNT OF NUMBER OF VALUES IN IC( ) PROCESSED.
C            NOUNT = NUMBER OF VALUES ADDED TO GROUP A.
C
C        NON SYSTEM SUBROUTINES CALLED
C           NONE
C
      DIMENSION IC(NDP)
      DIMENSION JMIN(NDQ),JMAX(NDQ),LBIT(NDQ),NOV(NDQ)
C
      IF(INC.LE.0)WRITE(KFIL12,100)INC
 100  FORMAT('0INC ='I8,' NOT CORRECT.  1 IS USED.')
      KINC=MAX(INC,1)
      KSTART=1
      KTOTAL=0
      LX=0
      ISKIPA=0
```

```
C        ************************************
C
C        THIS SECTION COMPUTES STATISTICS FOR GROUP A.  GROUP A IS
C        A GROUP OF SIZE MINPK IMMEDIATELY FOLLOWING THE GROUP JUST
C        PACKED.
C
C        ************************************
C
 110  KOUNTA=0
      IBITA=0
      MINA=9999999
      MAXA=-9999999
C
C        FIND THE MIN AND MAX OF GROUP A.  THIS WILL INITIALLY BE OF
C        SIZE MINPK (IF THERE ARE STILL MINPK VALUES IN IC( )), BUT
C        WILL INCREASE IN SIZE IN INCREMENTS OF INC UNTIL A NEW
C        GROUP IS STARTED.
C
      NENDA=MIN(KSTART+MINPK-1,NDP)
      IF(NDP-NENDA.LE.MINPK/2)NENDA=NDP
C        ABOVE STATEMENT GUARANTEES THE LAST GROUP IS GT MINPK/2 BY
C        MAKING THE ACTUAL GROUP LARGER.  IF A PROVISION LIKE THIS IS NOT
C        INCLUDED, THERE WILL MANY TIMES BE A VERY SMALL GROUP AT THE END.
C
      DO 120 K=KSTART,NENDA
      MINA=MIN(MINA,IC(K))
      MAXA=MAX(MAXA,IC(K))
      KOUNTA=KOUNTA+1
 120  CONTINUE
C
C        INCREMENT KTOTAL AND FIND THE BITS NEEDED TO PACK THE A GROUP.
C
      KTOTAL=KTOTAL+KOUNTA
 125  IF(MAXA-MINA.LT.2**IBITA)GO TO 130
      IBITA=IBITA+1
      GO TO 125
C
 130  CONTINUE
C
 133  IF(KTOTAL.GE.NDP)GO TO 200
C
```

```
C         ************************************
C
C         THIS SECTION COMPUTES STATISTICS FOR GROUP B.  GROUP B IS A
C         GROUP OF SIZE MINPK IMMEDIATELY FOLLOWING GROUP A.
C
C         ************************************
C
 140  MINB=9999999
      MAXB=-9999999
      IBITB=0
      JOUNT=0
      NENDB=MIN(KTOTAL+MINPK,NDP)
C
      DO 160 K=KTOTAL+1,NENDB
      MINB=MIN(MINB,IC(K))
      MAXB=MAX(MAXB,IC(K))
      JOUNT=JOUNT+1
 160  CONTINUE
C
      KOUNTB=NENDB-KTOTAL
C
 165  IF(MAXB-MINB.LT.2**IBITB)GO TO 170
      IBITB=IBITB+1
      GO TO 165
C
C        DETERMINE WHETHER THE NEXT MINPK VALUES CAN BE PACKED IN
C        LESS BITS THAN GROUP A.  IF SO, PACK GROUP A AND START
C        ANOTHER GROUP.
C
 170  CONTINUE
C
      IF(IBITB.GE.IBITA)GO TO 180
C
```

```
C         **************************************
C
C         GROUP B REQUIRES LESS BITS THAN GROUP A.  PUT AS MANY OF A'S
C         POINTS INTO B AS POSSIBLE WITHOUT EXCEEDING THE NUMBER OF
C         BITS NECESSARY TO PACK GROUP B.
C
C         **************************************
C
      KOUNTS=KOUNTA
C        KOUNTA REFERS TO THE PRESENT GROUP A.
C
      DO 173 K=KTOTAL,KSTART,-1
C        START WITH THE END OF THE GROUP AND WORK BACKWARDS.
      MINTST=MIN(MINB,IC(K))
      MAXTST=MAX(MAXB,IC(K))
      IF(MAXTST-MINTST.GE.2**IBITB)GO TO 174
      MINB=MINTST
      MAXB=MAXTST
      KOUNTA=KOUNTA-1
C        THERE IS ONE LESS POINT NOW IN A.
 173  CONTINUE
C
C        AT THIS POINT, KOUNTA CONTAINS THE NUMBER OF POINTS TO CLOSE
C        OUT GROUP A WITH.  GROUP B NOW STARTS WITH KSTART+KOUNTA AND
C        ENDS WITH NENDB.  MINB AND MAXB HAVE BEEN ADJUSTED AS
C        NECESSARY TO REFLECT GROUP B (EVEN THOUGH THE NUMBER OF BITS
C        NEEDED TO PACK GROUP B HAVE NOT INCREASED, THE END POINTS
C        OF THE RANGE MAY HAVE).  (Q:  CAN GROUP A BE EMPTY???)
C
 174  IF(KOUNTA.EQ.KOUNTS)GO TO 200
C        ON TRANSFER, GROUP A WAS NOT CHANGED.  CLOSE IT OUT.
C
C        ONE OR MORE POINTS WERE TAKEN OUT OF A.  RANGE  AND IBITA
C        MUST BE RECOMPUTED; IBITA COULD BE LESS THAN ORIGINALLY COMPUTED.
C        IN FACT, GROUP A CAN NOW CONTAIN ONLY ONE POINT AND BE
C        PACKED WITH ZERO BITS.
C
      KTOTAL=KTOTAL-(KOUNTS-KOUNTA)
      KOUNTB=KOUNTB+(KOUNTS-KOUNTA)
      IBITA=0
      MINA=9999999
      MAXA=-9999999
C
      DO 175 K=KSTART,NENDA-(KOUNTS-KOUNTA)
      MINA=MIN(MINA,IC(K))
      MAXA=MAX(MAXA,IC(K))
 175  CONTINUE
C
 176  IF(MAXA-MINA.LT.2**IBITA)GO TO 177
      IBITA=IBITA+1
      GO TO 176
C
 177  ISKIPA=1
      GO TO 200
C
```

```
C        ************************************
C
C        AT THIS POINT, GROUP B REQUIRES AS MANY BITS TO PACK AS GROUPA.
C        THEREFORE, TRY TO ADD INC POINTS TO GROUP A WITHOUT INCREASING
C        IBITA.  THIS AUGMENTED GROUP IS CALLED GROUP C.
C
C        ************************************
C
 180  MINC=MINA
      MAXC=MAXA
      NOUNT=0
      IF(NDP-(KTOTAL+INC).LE.MINPK/2)KINC=NDP-KTOTAL
C        ABOVE STATEMENT CONSTRAINS THE LAST GROUP TO BE NOT LESS THAN
C        MINPK/2 IN SIZE.  IF A PROVISION LIKE THIS IS NOT INCLUDED,
C        THERE WILL MANY TIMES BE A VERY SMALL GROUP AT THE END.
C
      DO 190 K=KTOTAL+1,MIN(KTOTAL+KINC,NDP)
      MINC=MIN(MINC,IC(K))
      MAXC=MAX(MAXC,IC(K))
      NOUNT=NOUNT+1
 190  CONTINUE
C
C        IF THE NUMBER OF BITS NEEDED FOR GROUP C IS GT IBITA,
C        THEN THIS GROUP A IS A GROUP TO PACK.
      IF(MAXC-MINC.GE.2**IBITA) GO TO 200
C
C        THE BITS NECESSARY FOR GROUP C HAS NOT INCREASED FROM THE
C        BITS NECESSARY FOR GROUP A.  ADD THIS POINT TO GROUP A.
C        COMPUTE THE NEXT GROUP B, ETC., UNLESS ALL POINTS HAVE BEEN
C        USED.
C
      KTOTAL=KTOTAL+NOUNT
      KOUNTA=KOUNTA+NOUNT
      MINA=MINC
      MAXA=MAXC
C        KOUNTA IS THE NUMBER OF VALUES IN GROUP A.  THIS GROUP WILL
C        NEVER BE SPLIT.
      IF(KTOTAL.LT.NDP)GO TO 140
C
```

```
C         ************************************
C
C         GROUP A IS TO BE PACKED.  STORE VALUES IN JMIN( ), JMAX( ),
C         LBIT( ), AND NOV( ).
C
C         ************************************
C
 200  LX=LX+1
      IF(LX.LE.NDQ)GO TO 205
      WRITE(KFIL12,201)
 201  FORMAT('0LX NOT LARGE ENOUGH.  STOP IN VRBLPK AT 201')
      STOP 201
C
 205  JMIN(LX)=MINA
      JMAX(LX)=MAXA
      LBIT(LX)=IBITA
      NOV(LX)=KOUNTA
      KSTART=KTOTAL+1
      IF(KTOTAL.GE.NDP)GO TO 209
      IF(ISKIPA.EQ.0)GO TO 110
C         WITH THE ABOVE TRANSFER, A NEW GROUP A OF SIZE MINPK WILL
C         BE DEFINED.
C
C         THE NEW GROUP A WILL BE THE EXPANDED GROUP B.  SET LIMITS, ETC.
C
      IBITA=IBITB
      MINA=MINB
      MAXA=MAXB
      NENDA=NENDB
      KOUNTA=KOUNTB
      KTOTAL=KTOTAL+KOUNTA
      ISKIPA=0
      GO TO 133
C
C         ************************************
C
C         CALCULATE IBIT, THE NUMBER OF BITS NEEDED TO HOLD THE GROUP
C         MINIMUM VALUES.
C
C         ************************************
C
 209  IBIT=1
C
 210  DO 220 L=1,LX
      IF(JMIN(L).LT.2**IBIT)GO TO 220
      IBIT=IBIT+1
      GO TO 210
C
 220  CONTINUE
C
      RETURN
      END
```