GRIBEX introduction in FA files

D. Paradis and J. Clochard

1. Context

Moore’s law, namely the doubling of the number of transistors on a chip every 18 months (or the doubling of the computing power at fixed cost every 18 months), has been maintained for 40 years, and still holds true today. However, in the face of this around 60 % annual increase of the CPU power, the memory access speed only grows by 7 % per year. Moreover, data storage and data international exchanges continuously raise, due to the resolution enhancement or the new productions.

This trend motivated us to implement the use of the second order packing in the **FA** (Fichier ARPEGE) software: this feature is already available in GRIB edition 1 (GRIBEX) and allows to store data on less bits than initially planned, without loss of accuracy.

2. Second order packing

Practically speaking, it may well be explained at encoding time. GRIB encoding of N grid point values on B bits leads to transform (in a linear fashion) original field values (generally floating-point) into the interval [0,2B-1], rounding them at the nearest integer values.

Ordinary ("simple") packing ends up processing at this stage, and packs the values obtained into a binary string of N*B bits.

The **basic idea of second-order packing** is to make extra processing, **splitting the full sequence of values into groups**. For each group, an **integer reference** value is taken (such as the minimum value of the group), stored as descriptor (**"first-order value"**), and the **positive deviations** from this reference value, called **"second-order values"**, are then **packed** together, **using just the bit number ("group width") sufficient** for that.

The **goal** is clearly to **reduce the overall size of data directly associated to field values**, with a drawback of **descriptors overhead**. All the associated processing, performed on integer-type data, is data conservative (lossless).

The "first-order" descriptors are stored on a bit number that uses the same place within GRIB as bit number for ordinary packing. There is also the **need to document the group widths, and how field is split into groups**. To do so, there are **three different methods** available in WMO standard: so-called row by row packing, constant width packing and a general method. All these methods (and sub-methods) do not change the data contents themselves at user level.

2.1 Row by row packing.

The idea is to **use coordinate lines as groups**: for instance, latitude lines. The group splitting is "induced", and there is **no need to document group boundaries** (e.g. section 2 is sufficient). Computing of group widths is straightforward. Source coding may even be done without any work array (but then leads to non optimal width of first-order values).

This method is in use at UKMO.

2.2 Constant width packing.

The idea is to save descriptors, but here on group widthes: there is a **single group width**, instead of one per group. Arbitrary group splitting is enabled; **group boundaries are documented by a secondary bit-map** (one bit per effective grid point value, taking into account potential primary bit-map), with one bit set at each group starting point.

2.3 General WMO second-order packing.

In this case, all simplifications described in previous methods are relaxed. **Secondary bit-map describes group boundaries** as described above, and there is **one group width per group**.

Moreover, three extensions for the WMO standard have been coded to increase the compression rate: general extended second-order packing, boustrophedonic ordering, and spatial differencing.

2.4 General extended second-order packing.

The first extension starts from a remark. The general WMO second-order packing exhibits limitations for compression efficiency at 2 levels:

-group widths are stored on 8 bits each (though 3 or 4 bits are generally sufficient);

-secondary bit-map is a fixed but important overhead, and leads to "non-linear" source code for decoding: **group splitting** may be **described** more economically (and decoded more simply) **with group lengths**, provided these lengths are **encoded with an adapted width**.

An extension may be coded, using an additional extended flag and three extra scalar descriptors (**bit number of widths and lengths**, pointer to group lengths), and re-allocating the secondary bit-map area to store group lengths. This extension is a variant of general WMO second-order packing, and may use the same algorithm for group splitting.

2.5 Boustrophedonic ordering.

The second extension aims at giving a more continuous series of values for splitting. There may be large differences of values between 2 consecutive lines of coordinates, especially for non-global fields. To improve numerical continuity, **lines of even rank** (starting from the 2nd line, as specified by scanning mode flags) may **have their values reversed**.

This may be described just by an appropriate extended flag, and coding is easy. It is a **sub-method** that **may be applied to all second-order packing methods**, but is disabled (at encoding level) for row by row packing, where it would have no interest (coordinate lines acting as groups). Such ordering is (in this context) fully handled internally within data section, and not reflected to user at data level.

2.6 Spatial differencing.

The basic idea of second-order packing methods is removal of local minima. But still, there is often remaining point to point redundancy. This **redundancy may be removed by preprocessing field (integer) values** just before second-order packing, using a **spatial differencing scheme**, (with order N=1 to 3) that may be compared to approximation of a Nth-order derivative. That's signal processing, finding some echo with (analogic) derivative circuits in electronics.

This spatial differencing being not efficient for all fields, implementation has to determine *a priori* whether the feature is worthwhile to be used. This is achieved in relationship with splitting algorithm, and so may only be selected sub-method with general extended second-order packing.

3. How to use it

Up to now, there were three possibilities to encode data with FA:

Type 0 encoding: no packing is done, to preserve all the precision of the data (surface geopotential for example);

Type 1 encoding: GRIB version 0 is used only to code data in a packed form (data are presented to GRIB in the integer form);

Type 2 encoding: as type 1 encoding but with the storage (8 bytes) of the min and the max of the data and with the use of all available bits.

From now on (XR28T2), two other types are introduced:

Type –1 encoding: no packing is done (the only difference with type 0 encoding is the ordering of the spectral coefficients);

Type 3 encoding: GRIB version 1 is used, with the possibility to make compression (second order packing).

First thing to do when using the type 3 or –1 encoding, is to keep the model ordering of **spectral fields**: coefficients are ordered along |JM|=cte columns or, differently said, the zonal wave number, JM, is used as an outer loop index. The call to SPREORD routine before writing an horizontal spectral field on a FA file or after reading one, is not yet useful (and must not be done!). This is due to the fact that with the type 3 encoding, the FA software directly uses the GRIBEX capability to pack Arpege spectral coefficients, which are supposed to be ordered as in model. This ordering type has been kept to pack Aladin's spectral coefficients with type 3 and the type –1 encoding has been introduced to give the possibility to have only one spectral ordering type in a FA file.

Secondly, the encoding type has to be specified to the FA software. There are two possibilities to do that:

. before opening the FA file(s), with a call to the FAGIOT routine,

. after you had opened the FA file, just call the FAGOTE routine.

For these two calls, the KNGRIB argument has to be set to 3 (or –1 if no packing is required). The number of bits used to pack the grid point values (KNBPDG) or the spectral coefficients (KNBCSP) and the non-packed sub-truncation (KSTRON) keep their meaning. You can reuse the predefined values for these 3 variables by previously calling the FAVORI or FAVEUR routines. However, the two other arguments (KPUILA and KDMOPL) lose their sense in the type 3 encoding.

Finally, you have to choose the second order packing options. The default in FA software, is to ask the selection of the best method between no-compression, row by row packing and the general extended 2nd-order packing, to make a boustrophedonic ordering and a spatial differencing with a dynamically estimated order. If these options have to be changed, the FAREGI or the FAREGU routines are available to modify the implicit tuning and the file specific tuning respectively. You will then be able to activate/deactivate the boustrophedonic ordering, to modify or activate/deactivate the spatial differencing, to activate/deactivate the general extended 2nd-order packing, to select constant width packing or general WMO second order packing, to ask GRIBEX to retain the most efficient packing method etc…

You can now write the horizontal fields with a FAIENC call!

To summarize, here is the calling sequence you can use to encode 'SURFTEMPERATURE' with the type 3, in an already opened FA file:

```
(...)
integer irep, inumer; ingrib, inbcsp, inbpdg, istron, ipuila, idmopl
real*8 ztsurf(ngptot)
logical llcosp
(...)
call faveur (irep, inumer, ingrib, inbcsp, inbpdg, istron, ipuila, idmopl)
ingrib = 3
call fagote (irep, inumer,  ingrib, inbcsp, inbpdg, istron, ipuila, idmopl)
llcosp = .false.
call faienc (irep, inumer, 'SURF', 1, 'TEMPERATURE', ztsurf, llcosp)
(...)
```

To read an horizontal field, the use of the FACILE routine is identical whatever the encoding type. Just remember that the ordering for the spectral fields is different between the types 0, 1 and 2 and the types –1 and 3 (see above).

4. Some relevant features

4.1 ALADIN spectral fields

The only spectral coefficients that are accepted by the GRIB norm, are the spherical harmonic ones (ARPEGE spectral fields, for example). To bypass this restriction, the FA software give to the

ALADIN spectral fields the appearance of a field on a quasi-regular latitude/longitude grid before calling GRIB code. Moreover, to better pack the spectral coefficients, the packing method for the spherical harmonic coefficients is adapted and applied: first, all coefficients on the axes and in the square whose vertex have the coordinates (0,istron), (istron,0),(0,-istron) and (-istron,0), are extracted from the field and stored separately in the FA article with all the original precision (8 bytes); then an optimal Laplacian exposant ("p") is calculated to smooth the remaining spectrum with the factor (jn**2 + jm**2)**p ("p" is stored in the FA article, as well as the non-packed sub-truncation "istron"); then the smoothed spectral field is given to GRIBEX to be packed as a grid points field, and being so able to take advantage of the second order packing.

4.2 ARPEGE spectral fields

As for the types 1 and 2 encoding, the type 3 encoding for an Arpege spectral field stores the spectral coefficients contained in the triangle defined by (jn<istron, where jn is the total wave number and istron the non-packed sub-truncation) separately in the FA article with all the original accuracy (8 bytes), the only difference being the presence of (istron+1)*(istron+2) non-packed coefficients instead of (istron+1)**2, due to the difference in ordering.

Unlike the types 1 and 2 encoding, the type 3 encoding gives directly the whole original ARPEGE spectral field to GRIBEX and leaves it to transform the spectrum (smoothing by a factor (jn*(jn+1))**p, where jn is the total wave number and p the optimal laplacian exposant) before packing. The dynamical computation of the laplacian exponent (for ARPEGE and for ALADIN) makes all the pre-computed laplacian factors useless if only the type 3 encoding is used. For the type 1 and 2 encodings, these pre-computed factors were actually stored in several arrays in FA software and took up a lot space in memory when the maximal permitted truncation was high: the goal was then to replace the CPU consumption by a memory consumption. Now, the aim has changed and, with the constraint to only use the type 3 encoding, the memory need is greatly weaker and this advantage will become even more important when higher resolution will be used. A unique version of the FA software will also allow very high resolution without needing too much memory if the used resolution is actually crude.

4.3 GRIB auto documentation

The FA software uses the FA documentation articles to provide most of the information necessary to initialize the documentary sections of each GRIB message. However, the routine FAREGU can be used to specify another identification centre (KSEC1(2)=85 by default) or another generating process identification number (KSEC1(3)=177 for ALADIN, 211 for ARPEGE forecast and 201 for ARPEGE analysis, by default). The routine FARPAR allows also to modify or to create a connection between an horizontal field name (prefix and suffix in FA) and its GRIB descriptors: 6 integers which represent the version number of the parameters table (KSEC1(1)), the parameter indicator KSEC1(6), the type level indicator KSEC1(7), the level value KSEC1(8), the bottom of the layer in case of a layer or 0 otherwise (KSEC1(9)) and lastly a time range indicator KSEC1(18) (=0 except if the field is a min/max during a period (=2) or except if the field is an accumulation in a period (=4)). The basic connections are quite numerous (215 cases are taken into account in the set up of FA, see routine FAICOR) but the user can modify the connection of a field (prefix+suffix) or create new ones.

This auto documentation of the GRIB message and the respect for the GRIB norm allow to decode a FA file containing only type 3 encoded fields with other softwares such as the PBIO interface of ECMWF : all GRIB messages (containing an horizontal field) can be completely identified except for GRIB messages containing an ALADIN spectral field (these fields are disguised as grid points fields and the spectral coefficients have been transformed to be better packed).

4.4 Decimal factor

For each grid points field and each ALADIN spectral field, the FA software computes the decimal factor which will optimize the bits use in the packing (for the type 3 encoding only). This feature is automatic in FA and is then handled by GRIB software. It allows to approach the best bits use done in type 2 encoding when the minimum and the maximum of the field were respectively represented by 0 and 2B-1, where B is the bits number used for packing.

4.5 Specific treatment

The latitude/longitude grid points fields built by Full-Pos have their values ordered from SouthWest to NorthEast (for ARPEGE and ALADIN). But for the data base at Toulouse, it is better to order them from NorthWest to SouthEast. So these fields are reordered before being written on the FA file (and the scanning mode flag in the GRIB message is set to NW -> SE to be coherent with the data). However, when these fields will be read on the file, the FA software will reorder them again from SouthWest to NorthEast before giving them to the user.

A special processing is also done for some fields whose values in ARPEGE and ALADIN are comprised between 0 and 1 instead of 0 and 100% (norm in the GRIB): albedo, nebulosity, vegetation proportion and relative humidity. The GRIB message is then slightly modified (just the decimal factor) to multiply all the field by the factor 100. When data are read again by the FA software, the opposite action is done by FA and the user will find the original values (included in [0,1]).

Normally these two specific features have no consequence on the user, GRIB messages being coherent, but in the case of the use of another interface than the FA software, the user must be aware of this GRIB logic.

5. Some results

To test the GRIBEX implementation in the FA software on a VPP5000 Fujitsu processor, an ALADIN FA file has been written during a forecast of the ALADIN-France model (elliptic truncation of 149) without packing the fields (type 0 encoding). This file contains 166 spectral fields and 65 grid points fields and has a size of 133.8 MB.

Similarly, an ARPEGE FA file has been written during a forecast of the global ARPEGE model (TL358C2.4) without packing the fields. This file contains 166 spectral fields and 92 grid points fields and has a size of 290.9 MB.

Then, for these two files, the experience has consisted in reading one file, in writing all the articles on another new file with the type 2 encoding, and then in reading again the packed fields and in checking the decoded data with the original ones. The experience has been run again but with type 3 encoding.

The packing options (identical to those used in operation at Toulouse) are the following:

	Bits number for grid points values packing	**Bits number for spectral coefficients packing**	**Non-packed sub-truncation**
Aladin T149	16	18	20
Arpege TL358	16	16	35

For the type 3 encoding, two methods have been tested: firstly, the default second order packing options (APAC1, GEXTE, BOUST and DIFFE(-1), see paragraph C) and secondly, APAC1 (best method between row by row and no compression).

On the following tables, one can evaluate the second order packing impact:

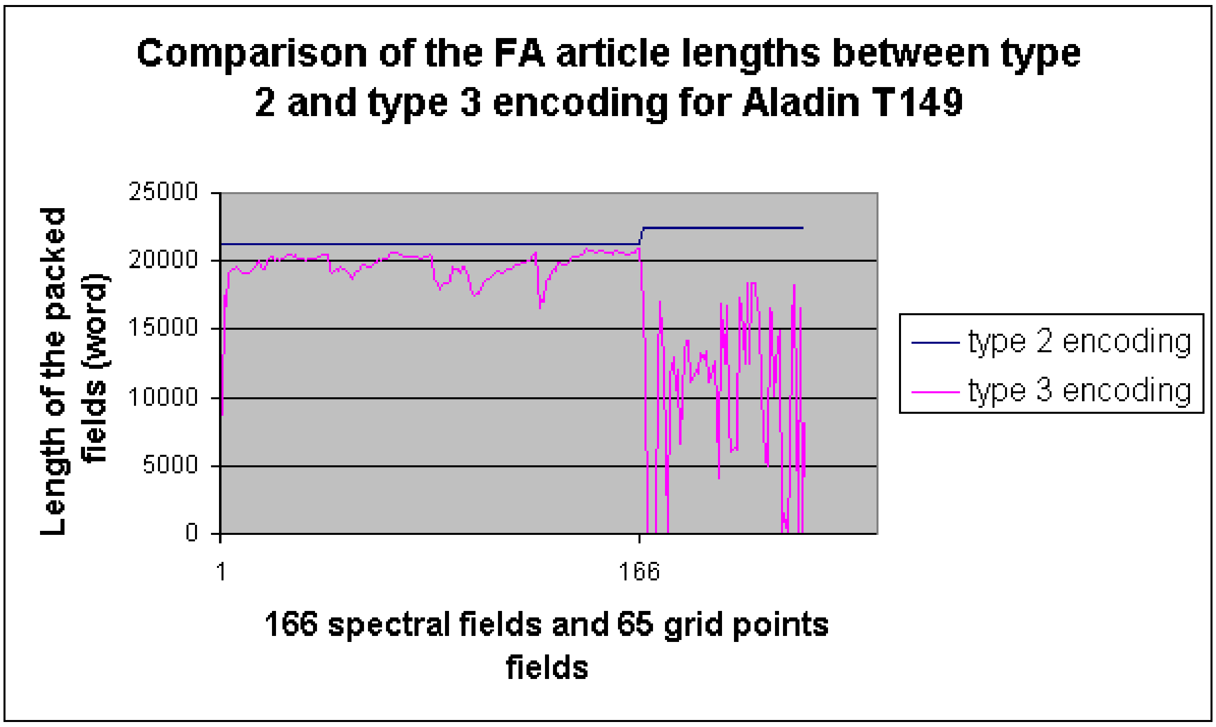
for ALADIN, the default second order packing increases the CPU time by a factor of 4.3 (7 to 30 s), decreases the memory need by 42% and generates a file whose size is 21% smaller. With a more basic method (APAC1), the CPU cost is weak (8s instead of 7s for the type 2 encoding) and the file size decreases even so by 10 %.

ALADIN T149			
type of encoding	**2**	**3** **default options**	**3** **APAC1**
Total user CPU time (s)	7	30	8
User vector CPU time (s)	2	14	1
System CPU time (s)	7	8	7
Memory (MB)	384	224	224
File size (MB)	38.1	30.2	34.5

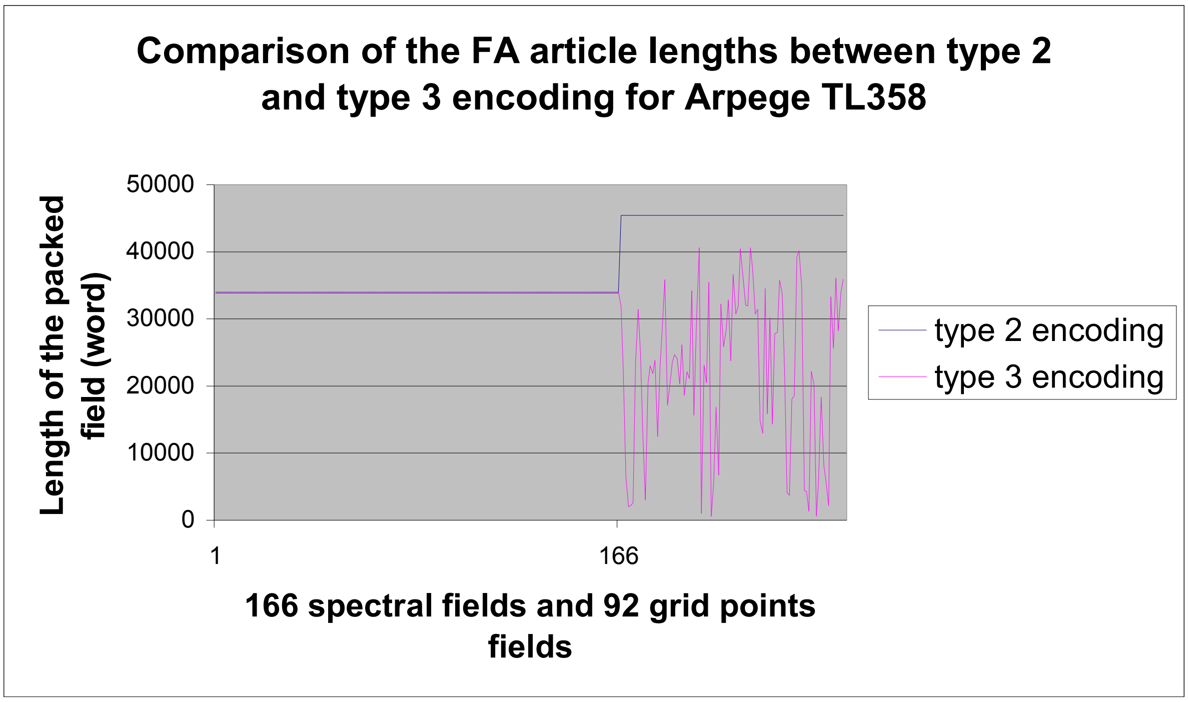
for ARPEGE, the default second order packing increases the CPU time by a factor of 2.8 (12 to 33 s), decreases the memory need by 42% and generates a file whose size is 22% smaller. With the APAC1 method, the CPU cost is, once again, negligible but the file size does not go down greatly: only 5%.

ARPEGE T358			
type of encoding	**2**	**3** **default options**	**3** **APAC1**
Total user CPU time (s)	12	33	13
User vector CPU time (s)	1	12	1
System CPU time (s)	15	14	15
Memory (MB)	384	224	224
File size (MB)	74.9	58.7	71.2

To detail the behaviour of the second order packing (with the defaults options), one can see its work on each field of the two files. The first figure shows the length of each field (spectral ones and then grid points ones) for the ALADIN FA file. In this case, the second order packing is also applied to the spectral coefficients but the gain is weaker than with grid points values (the point to point redundancy is actually less important).



A similar figure is proposed for the ARPEGE fields. No second order packing is then applied to the spectral coefficients and the corresponding articles are a little bit larger than with the type 2 encoding, due to the more documented GRIB sections (which allow the GRIB to be decoded independently from the FA software).



Finally, the precision of the encoding has been compared for the type 2 and 3 encodings. Except for the ALADIN spectral coefficients which lose less precision, all other fields seem to be a little less accurate with the type 3 encoding. It is well explained by the better use of the bits done with the type 2 encoding, despite the decimal factor optimization in the type 3 encoding. Anyway, the overall impact on the quality of the fields have to be evaluated with an e-suite and a comparison based on meteorological criteria instead of statistical ones.

CONTENTS