Data Bank Compression Programs

There are two compression programs: *CPress* and *HPress*. *CPress* is the orginal *Press* program, while *HPress* compresses *G* workspace bank directly into a hashed bank. Hashed banks are now the standard data banks that Inforum creates and maintains. The compression routines for hashed and compressed banks are essentially the same, however, with the hashed banks we have added a far superior method of indexing the data series.

Since the compression of both types of banks is so similar, the syntax and usage for both compression programs are essentially the same as well. Therefore, what follows assumes one is using *HPress*. If you wish to use the original compression program, simply substitute *CPress* where *HPress* appears and read 'compress,' where 'hash' appears below.

The *HPress* data bank compression program can generally reduce the size of a *G* data bank by at least a factor of 2 and sometimes by much more. The banks which it produces can be used directly in *G7* and *G*.

The general invocation of *HPress* is:

```
hpress <original_bank> [hashed_bank] [-m<missing>] [-s<maxslash>]
```

As usual, the <> enclose required items; [ ], optional items. Here are several examples:

```
hpress nipaq
hpress nipaq e:nipaq
hpress nipaq -m-999999
hpress nipaq -s2
hpress nipaq e:nipaq -m-999999 -s4
```

If no hashed bank is named, as in the first example, the original bank name, but with different extensions is used for the output bank. If the data source has used some unusual value, such as -999999, to indicate a missing value, these observations may be turned to zeroes using the -m option, as in the last example. The meaning of the "maxslash" option is explained below.

Banks which have been pressed by *HPress* have the extensions ".hin" and ".hbk" for their index and data files, respectively. To assign a hashed bank in *G7* or *G*, the command is simply

```
hbk <bank_name>
```

for example

```
hbk nipaq
```

assigns the hashed quarterly NIPA bank. All other commands should then work exactly as with any other assigned bank. It is NOT possible to assign a hashed bank as a workspace with the wsb command. It IS possible to assign a hashed bank as the initial bank in the G.CFG file.

HPress does two things to compress each series.

1. Leading and trailing zeroes are removed from the series.
2. Whenever possible, a series is represented mainly by 2-byte integers rather than by 4-byte floating point numbers.

To accomplish the second step, the number of decimal places in each series is found and the decimal point is slid to the right that many places and the result expressed as a 4-byte integer. (If some number is too big to be expressed as a 4-byte integer, the number of decimal places is reduced and the process repeated.) First differences of the (non-zero) values are then calculated and checked for their expressability as two-byte integers. If all of them pass, the series is then stored by recording the starting date, frequencey, number of observations, number of decimal places, starting observation as a four-byte integer and then the first differences as two-byte integers. If this compression fails, then either (a) the numbers are stored as four-byte floating-point numbers or (b) the differences are divided by a power of 2 up to a maximum of "maxslash" as given by the -s option.

What precision is possible in a compressed series? A laser printer typically prints 300 dots per inch. The precision of the first differences in a compressed series is comparable to one such dot in a graph nine feet high! Although all series in the US quarterly national accounts from 1947 to 1988 compress with complete accuracy, about five percent of the series in the Blue Pages of the Survey of Current Business fail, and about ten percent of the series in the International Financial Statistics fail to compress. This failure occurs when series are being carried to six or seven significant figures in the sources. Obviously, this much accuracy is seldom of any value in economic use of the series. If compaction is quite important, you may therefore want to compress these series at a minor cost in terms of accuracy. To do so, use the -s option on the command line to set "maxslash", the maximum power of 2 which will be used to divide the differences to get them down to the size which can be expressed as a two-byte integer. Obviously, the slash value actually used for each series is stored with the series and is used by G in interpreting the compressed series. A file called "forced" is created with each compression. It lists all series either slashed (marked "forced") or dumped as four-byte floating point numbers (marked "gave up"). This file has the form of an add file for G to draw graphs of the original and compressed series. The default value of maxslash is 0; only compression with perfect accuracy allowed. However, values of maxslash as high as 4 have not led to graphically distinguishable series. (Files on Econdata are compressed with maxslash = 0.)

Even in the case of failure in compression, the elimination of leading and trailing zeroes often reduces the size of the bank. Also, the organization of index file in a hashed bank greatly speeds up the series searching process in PDG, among others.

The easist way to update a hashed bank is by using HSPLICE. There is another (and, unfortunately, more tedious) way to update a hashed bank. For example, to update a hashed bank, say nipaq.hbk, from the file newnipq.hbk, the steps are as follows:

1. Run the BUPS program.

```
bups nipaq
```

This will create the ascii file NIPAQ.BUP.

2. Start *G* and use the "zap" command to set the starting date and size of the updated bank. Then do:

```
hbk nipaq
add nipaq.bup
hbk newnipq
add nipaq.bup
q
```

3. The workspace bank of G is now the updated bank. Rename it. If it was named ws and we wanted to rename it upnipaq, then we would do:

```
ren ws.* upnipaq.*
```

4. If you wish to compress the bank, do:

```
hpress upnipaq
```

When you have checked that all is well in the updated bank, you may, of course, wish to rename it to nipaq.

Note for Programers

The precise form of the hashed bank .hin and .hbk files is as follows:

The ".hin" file contains:

item	size in bytes	C type
ns	4	long
nbins	2	unsigned
nsb	2*nbins	unsigned
ncharb	2*nbins	unsigned
posbin	4*nbins	unsigned
binname(0)	nchar[0]	char
binposts(0)	4*nnmsb[0]	long
binname(1)	nchar[1]	char
binposts(1)	4*nnmsb[1]	long
binname(2)	nchar[2]	char
binposts(2)	4*nnmsb[2]	long
.		
.		
binname(nbins-1)	nchar[nbins-1]	char
binposts(nbins-1)	4*nnmsb[nbins-1]	long

Here, *ns* denotes the number of series in the bank. The series are separated into "bins". The number of bins in the bank is denoted by *nbins*. The number of series in each bin is denoted by the array *nsb*. The sum of the number of characters in the names (including each '\0') of the series contained in each bin is denoted by the array *ncharb*. The beginning positions in the ".hin" file of the first bytes of the binname() strings is given by the array *posbin*. The string *binname(i)* denotes the concatenation (including the \0's) of all the series names in the i-th bin. Finally, *binposts(i)* denotes the array of beginning positions in the associated ".hbk" of the series in the the i-th bin. Of course, the ordering of the series in the *binname()* and *binposts()* arrays must be the same.

Consider an example. Suppose that the 3rd bin contains the series "joe", "dave", and "bill". The string binname(3) would be:

```
        "joe\0dave\0bill\0"
```

Suppose that the starting positions in the ".hbk" bank for the three series are 40700008, 490987, 3378294. The array binposts(3) would then be [40700008, 490987, 3378294]. And *nsb[3]* = 3, and *ncharb[3]* = 14. If the beginning position of *binname(3)* in the ".hin" file is 4724, then *posbin[3]* = 4724.

To assign a bin number to a series you must use the following hashing routine. In C, the routine is:

```
unsigned hash(char *s);

hash(char *s)
{
   unsigned bill;

   for (bill=0;*s!='\0';s++) bill = *s + 31*bill;
   bill = bill%nbins;
   return(bill);
   }
```

To continue with the example, to determine the bin which the series "joe" really belongs to you'd evaluate the function hash("joe").

The .hbk file:

```
    0 - 79           char       Name of bank (terminated with a null)
    80 - 81          int        ns, number of series in the bank
    82 - 85          long       psn, position in file of index
    86 -                        first series, as described below
    *(psn+1) -                  second series,
    ...                         ...
    psn              long       position in file of first byte of first
series
    psn+4            long       position in file of first byte of second
series
    ...                         ... on out to ns series
```

For each series, the format is:

```
byte  Content
0           base year
1           frequency*16+period
2           slash*16+maxplaces or 255 if not compressed
3-4         number of observations
5-8         first observation as a long
9 -         differences as integers
if not compressed, floats begin in byte 5
```