Subject: Any interest in READ/WRITE functions for sparse (CSR) matrices?
Posted by tom.grydeland on Mon, 26 Nov 2012 09:14:23 GMT
View Forum Message <> Reply to Message

Hi all,

I tried searching for a bit, but couldn't find an established (binary) format for sparse matrices. Hence, I've done what one should never do -- dreamt up my own file format for saving such data. I'm using a slight variation on the Yale or CSR representation (http://en.wikipedia.org/wiki/Sparse_matrix#Compressed_sparse _row_.28CSR_or_CRS.29), although I'm not using 'row' and 'column' in the code or documentation, but rather 'x' and 'y'. I differ from CSR in that I'm storing an x count instead of an x start index (x count is limited to LE x size, so the width of the x count and x index can be known in advance.)

The format allows for an arbitrary number of 'channels' in the file, all of which share the index and have the same data type. The usual case of a sparse matrix means one channel. Zero channels is also allowed, and can be used as an index (e.g. a mask)

I'm including the RST documentation for the READ_CSR function here, and if there is interest, I can probably get permission to release the code into the wild.

```
;+
; Reader for CSR (Compressed sparse row) data.
;
; Primarily to serve as file format demonstration.
;
; File format:
;   24-byte header:
;     0-3:   the string 'GCSR'
;     4:     number of bytes per index.  Should be 2 or 4 (maybe 3?)
;     5:     The IDL typecode of each value
;     6-7:   NCH  - number or values per entry, lsb first.
;     8-11:  XSIZE - length of first dimension, lsb first
;     12-15: YSIZE - length of second dimension, lsb first
;     16-23: NNZ  - number of non-zero entries, lsb first
;   Next follows YSIZE xcount values (number of valid x'es per y)
;   Next follows NNZ x indices
;   Next follows NCH * NNZ data values, in whatever format
;
; :Params:
;   FILENAME: required, in
;
; :Returns:
;   DATA from file.  either [[nch, ] xsize, ysize] with values not found in file
;   set to MISSING, or [[nch, ], nnz] if the SPARSE keyword was set (in which
;   case you should also specify the INDEX keyword to know where in the data
;   world the actual data lives.)  If the file is a sparse index file, (no
```

```
;   data), then !NULL is returned.
;
; :Keywords:
;   SPARSE: optional, in, type=boolean
;     If set, only the data points given in the file are returned.  The default
;     is to make a dense array and fill it with data at appropriate points.
;   MISSING: optional, in
;     When not using SPARSE, this is the value that all missing points will have
;     upon return.  The default is 0 for integral types and NaN for
;     floating-point and complex floating-point types
;   GET_INDEX: optional, in, type=boolean
;     If set, any data in the file is ignored and !NULL is returned.  Can be
;     used with keyword INDEX to retrieve indices only.
;   INDEX: optional, out
;     If given, will be an array of [2, NNZ] 2-D indices.
;   XSIZE: optional, out
;     When given, will be filled with the XSIZE field from the file header. May
;     be useful when the keyword SPARSE is set.
;   YSIZE: optional, out
;     When given, will be filled with the YSIZE field from the file header. May
;     be useful when the keyword SPARSE is set.
;   SUB_RECT: optional, in
;     defines a subset [x, y, nx, ny] to read from the file
;
;
; :Examples:
;   Read a sparse file, produce a dense array::
;     data = READ_CSR('sparse.gcsr')
;
;   Read a sparse field of values into `val`, with indices in `idx`::
;     val = READ_CSR('sparse.gcsr', index=idx)
;
;   Read a sparse file with integers into a dense array, substituting the value
;   -65500 for all missing data::
;     data = READ_CSR('sparse.gcsr', missing=-65500)
;-
```

Regards,

--T (Tom Grydeland @ norut no)
