Subject: Re: Adding sparse matrices
Posted by Jeremy Bailin on Fri, 21 Nov 2008 18:52:09 GMT
View Forum Message <> Reply to Message

On Nov 21, 12:15 pm, Luis <lgmen...@gmail.com> wrote:
> Unfortunately I can not use c = sprsin(fulstr(a) + fulstr(b)) (Out of
> memory problems)
>>> On Nov 21, 9:27 am, Luis <lgmen...@gmail.com> wrote:
>
>>>> Hi,
>
>>>> Does anyone know how to add two sparse matrices?
>
>>>> Tks,
>>>> Luis

Try this. It worked for my test cases, but I can't say I tested it
extensively or worked hard to make it particularly efficieint...

```
;+
; NAME:
;    SPRSADD
;
; PURPOSE:
;    Adds two sparse matrices (as generated by SPRSIN).
;
; CATEGORY:
;    Math
;
; CALLING SEQUENCE:
;    Result = SPRSADD(A,B)
;
; INPUTS:
;    A:   Sparse matrix to be added.
;    B:   Sparse matrix to be added.
;
; OUTPUTS:
;    The addition of the matrices, in sparse format. This is
functionally
;    equivalent to:
;      SPRSIN(FULSTR(A) + FULSTR(B))
;    but can be used even when the full matrices take up too much
memory
;    for that operation.
;
; EXAMPLE:
;    IDL> a = sprsin([[1,0,0],[0,1,0],[0,0,1]])
;    IDL> b = sprsin([[0,0,2],[0,2,0],[2,0,0]])
```

```
;    IDL> c = sprsadd(a,b)
;    IDL> print, fulstr(c)
;          1.00000      0.00000      2.00000
;          0.00000      3.00000      0.00000
;          2.00000      0.00000      1.00000
;
; MODIFICATION HISTORY:
;    Written by:   Jeremy Bailin, November 2008
;
;-
function sprsadd, a, b

; what is the size of the matrix?
N = a.ija[0]-2
if b.ija[0]-2 ne N then message, 'SPRSADD: A and B must have the same
size.'

; figure out the full indices of all specified elements
a_i = lonarr(n_elements(a.sa)-1)
b_i = lonarr(n_elements(b.sa)-1)
; diagonal elements
a_i[0] = lindgen(N)*N+lindgen(N)
b_i[0] = lindgen(N)*N+lindgen(N)
an=N & bn=N
for i=0l,N-1 do begin  ; loop through rows
  a_non0 = a.ija[i+1]-a.ija[i]
  b_non0 = b.ija[i+1]-b.ija[i]
  if a_non0 gt 0 then a_i[an:an+a_non0-1] = i*N + a.ija[a.ija[i:i
+a_non0-1]-1]-1
  an += a_non0
  if b_non0 gt 0 then b_i[bn:bn+b_non0-1] = i*N + b.ija[b.ija[i:i
+b_non0-1]-1]-1
  bn += b_non0
endfor

all_index = [a_i,b_i]
if n_elements(a.sa) eq N+1 then a_vals = a.sa[0:N-1] $
  else a_vals = [a.sa[0:N-1],a.sa[N+1:*]]
if n_elements(b.sa) eq N+1 then b_vals = b.sa[0:N-1] $
  else b_vals = [b.sa[0:N-1],b.sa[N+1:*]]
all_vals = [a_vals,b_vals]
indexsort = sort(all_index)
indexuniq = uniq(all_index[indexsort])
dupes = where(indexuniq-[-1,indexuniq] gt 1, comp=single, $
  ncomp=nsingle)
; we know there are at least N dupes because the diagonals are
; always represented, so we don't need to test the where
combined_index = all_index[indexsort[indexuniq[dupes]]]
```

```
combined_vals = all_vals[indexsort[indexuniq[dupes]]]+all_vals
[indexsort[indexuniq[dupes]-1]]
if nsingle gt 0 then begin
  combined_index = [combined_index, all_index[indexsort[indexuniq
[single]]]]
  combined_vals = [combined_vals, all_vals[indexsort[indexuniq
[single]]]]
endif

; create the output array using SPRSIN in col, row, val form
return, sprsin(combined_index mod N, combined_index/N, combined_vals,
N)
end
```