Subject: Re: Sparse matrix algorithms
Posted by the_cacc on Fri, 30 Nov 2001 11:54:43 GMT
View Forum Message <> Reply to Message

Hi,

I had this experience a few weeks ago. The NR code is for NxN only so I doubt IDL will generalize to NxM on their own. I wrote to NR suggesting they publish NxM sparse algorithms and they gave a very positive response so expect them to be available next edition of the book.

In the meantime, the penalty for expanding your NxM to MxM (assuming M>N) in sparse format is M-N additional zeros (ie. those on the diagonal). I found this quite acceptable for my matrices: 64000 x 128000 with around 10^6 non-zero entries. So I have to store 64000 zeros unnecessarily.

You will have to write your own version of sprsin to deal with this, although since you probably won't be able to store your full matrix in memory you'll need to do this anyway.

Ciao.

Subject: Re: Sparse matrix algorithms
Posted by Ralf Flicker on Fri, 30 Nov 2001 19:28:44 GMT
View Forum Message <> Reply to Message

trouble wrote:
>
> Hi,
>
> I had this experience a few weeks ago. The NR code is for NxN only so
> I doubt IDL will generalize to NxM on their own. I wrote to NR
> suggesting they publish NxM sparse algorithms and they gave a very
> positive response so expect them to be available next edition of the
> book.

That's interesting; did they say when a new edition can be
expected?

> In the meantime, the penalty for expanding your NxM to MxM (assuming
> M>N) in sparse format is M-N additional zeros (ie. those on the
> diagonal). I found this quite acceptable for my matrices: 64000 x
> 128000 with around 10^6 non-zero entries. So I have to store 64000
> zeros unnecessarily.

I hadn't thought of this...I could probably use this trick. I'm wondering though if there are some other penalties when doing, for instance matrix multiplications? Some of my matrices are extremely skinny, with an extremely large "long" dimension (on the order of millions), and I have to compute the matrix multiply transpose(A)##A which in the sparse algorithm loops over the smaller dimension. Filling this out might be costly (timewise) for me, though I'm just speculating.

Anyway, I started coding and discovered it wasn't so hard, so I forged ahead and implemented the algorithms from Pissanetsky's book. I'm almost done with what I need, and I wrote a converter to the NR row-indexed storage scheme to be able to use the linbcg routine for square matrices. In stark contrast to the lack of simple general purpose routines, they do have the sparse bi-conjugate gradient solver implemented (since they could rip it directly out of NR).

Thanks for the tip.

cheers
ralf

--
Ralf Flicker
Gemini Observatory   http://www.gemini.edu/
670 N. A'Ohoku Pl.   Tel : (808) 974-2569
Hilo 96720, HI, USA   Fax : (808) 935-9235

Subject: Re: Sparse matrix algorithms
Posted by the_cacc on Mon, 03 Dec 2001 14:56:19 GMT

View Forum Message <> Reply to Message

> did they say when a new edition can be expected?

No, but I didn't ask. Email nr@nr.com and they'll write back promptly (in my experience).

> ... Some of my matrices are
> extremely skinny, with an extremely large "long" dimension (on
> the order of millions), and I have to compute the matrix
> multiply transpose(A)##A which in the sparse algorithm loops
> over the smaller dimension. Filling this out might be costly
> (timewise) for me, though I'm just speculating.

I think you may be right - storing millions of zeros does defeat the purpose. In my case, 10^4 zeros is not such a burden. From what you say it seems you may be attempting to solve Ax = b by using A^T A x = A^T b. If so, don't forget that matrix algebra is associative so you can do A^T (Ax) rather than (A^T A)x and save some CPU cycles.

> ...the sparse
> bi-conjugate gradient solver implemented (since they could rip
> it directly out of NR).
>

I found this to be slower than the conjugate gradient solver, which you'll have to implement yourself (v. easy by the way) and which is guaranteed to converge for the problem A^T A x = A^T b.

Ciao.