Subject: Re: Inverting banded-block matrices.
Posted by Liam E. Gumley on Fri, 29 Aug 2003 15:32:04 GMT
View Forum Message <> Reply to Message

"James Kuyper" <kuyper@saicmodis.com> wrote in message
news:3F4E8B4D.B90EDAA8@saicmodis.com...
> I've got a problem where I have to calculate g = C D^-1 f, where g and f
> are vectors, and C and D are matrices. C has m by m blocks, each of
> which is itself an n by n matrix. It is banded, with k non-zero
> co-diagonals above and below the main diagnal, both at the block level
> and within each block. C[i,j] ge 0. Every statement I've made about C
> also applies to D.
> For the sake of definiteness, m=10, n=1354, k = 3.
>
> This seems like it should be a pretty common type of matrix structure
> for problems involving 2-D grids. I could solve this by explicitly
> inverting a m*n by m*n matrix. However, I would assume that there are
> existing routines somewhere which can take good advantage of the
> sparseness of these matrices to speed up the calculations considerably.
> Could anyone point me at such routines?

I believe LAPACK routines were incorporated in IDL 5.6, however I have not
tried them.

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/

Subject: Re: Inverting banded-block matrices.
Posted by the_cacc on Sat, 30 Aug 2003 00:30:08 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@saicmodis.com> wrote in message
news:<3F4E8B4D.B90EDAA8@saicmodis.com>...
> I've got a problem where I have to calculate g = C D^-1 f, where g and f
> are vectors, and C and D are matrices. C has m by m blocks, each of
> which is itself an n by n matrix. It is banded, with k non-zero
> co-diagonals above and below the main diagnal, both at the block level
> and within each block. C[i,j] ge 0. Every statement I've made about C
> also applies to D.
> For the sake of definiteness, m=10, n=1354, k = 3.
>
> This seems like it should be a pretty common type of matrix structure
> for problems involving 2-D grids. I could solve this by explicitly
> inverting a m*n by m*n matrix. However, I would assume that there are
> existing routines somewhere which can take good advantage of the

> sparseness of these matrices to speed up the calculations considerably.
> Could anyone point me at such routines?

There are some sparse routines in IDL - sprsin, sprsax, sprsab, sprstp. They're "general" rather than specifically for a certain structure of sparseness, but that should be OK.

Your matrices are pretty big so you can forget about matrix inversion and use an iterative method instead - conjugate gradient (CG) is good.

Actually, can you even store one of those matrices in memory? (I'm on a 80 MB machine so I can't!). If you can, then try

IDL> sparse_D = sprsin(D)

(NB. May be rather slow) If you can't, then you'll have to write a routine to create your matrix in sparse format - hopefully you won't have to do this - trust me :) Let's leave that alone for the moment.

Rewrite as (1) g = C . h
(2) h = D^-1 . f

The real work is solving (2). You basically have to solve D . h = f, or sparse_D . h = f, which IDL's linbcg may solve for you.

I found linbcg not so good as the simple CG, which I've implemented. You'd be welcome to use it if you want - but first check if you can create the sparse matrix!

Ciao.
