Subject: Re: generalized eigenvectors
Posted by hradilv.nospam on Mon, 15 Apr 2002 15:36:58 GMT
View Forum Message <> Reply to Message

I too had a need not too long ago for the "eig" function. I ended up calling lapack/clapack routines. It worked nicely. Check out: http://epsilon.nought.de/idl_lapack.php I think this is where I got inspiration, but I can't reach it know because of firewall problems 8^{

On Mon, 15 Apr 2002 16:53:44 +0200, Tron Darvann <tdarvann@lab3d.odont.ku.dk> wrote:

> I have a question concerning solving a GENERALIZED EIGENVALUE PROBLEM in
>
> IDL.
>
> Description of the problem:
> I need to find the eigenvalues and eigenvectors of
> Ax = kBx
> where both A and B are nXn matrices and k is a scalar.
>
> The solution to this can be computed in MATLAB by their "eig" function,
> which, according to their documentation uses a math/statistics software
> called lanpack.
>
> Question: Does IDL have a similar routine? Do you have any suggestions
> as to how to solve a generalized eigenvalue problem in IDL?
>
> Thanks in advance,
> Tron Darvann
>
> tdarvann@lab3d.odont.ku.dk
>
>
>
>

Subject: Re: generalized eigenvectors
Posted by Randall Skelton on Mon, 15 Apr 2002 21:29:31 GMT
View Forum Message <> Reply to Message

I am not sure I understand what you mean by a "Generalized Eigenvalue problem".

IDL does have routines for computing the Eigenvalues and Eigenvectors. From the IDL help:

EIGENQL - Compute eigenvectors of a real, symmetric array, given the array.
EIGENVEC - Compute eigenvectors of a real, nonsymmetric array, given
        the array and its eigenvalues.
ELMHES   - Reduce a real, nonsymmetric array to upper-Hessenberg form.
HQR   - Compute the eigenvalues of an upper-Hessenberg array.
TRIQL   - Compute eigenvalues and eigenvectors of a real, symmetric,
        tridiagonal array.
TRIRED   - Use Householder's method to reduce a real, symmetric array to
        tridiagonal form.

Most of these methods are described in the Numerical recipies books.
See http://www.ulib.org/webRoot/Books/Numerical_Recipes/

In IDL the user must decide if the input matrix is symmetric or not then
use the appropriate tools. The Matlab EIG function basically uses the
same tools (to a first order), but automatically determines the "best"
method based on your input matrix... I personally find Matlab's auto-magic
approach to be more trouble than it is worth. Moreover, it promotes the
idea that you don't really need to understand the numerical problem you
are trying to solve...

The general approach is:

- If your matrix is real and symmetric, convert to the tridiagonal form
(TRIRED) and then use the QR procedure (TRIQL) to iteratively find the
eigenvalues/vectors from the tridiagonal array.

- If your matrix is real and not symmetric, reduce to Hessenberg form
using the Householder's transformation method (ELMHES) and then use the QR
procedure (HQR) to get the eigenvalues/vectors of the upper Hessenberg
matrix.

- Unfortunately, there is no built in IDL code for the QZ (Golub and Van
Loan, 1989) algorithm which can be modified to work for complex matricies.

Hope this helps,
Randall

 On Mon, 15 Apr 2002, Tron Darvann wrote:

> I have a question concerning solving a GENERALIZED EIGENVALUE PROBLEM in
>
> IDL.
>
> Description of the problem:
> I need to find the eigenvalues and eigenvectors of
> Ax = kBx

> where both A and B are nXn matrices and k is a scalar.
>
> The solution to this can be computed in MATLAB by their "eig" function,
> which, according to their documentation uses a math/statistics software
> called lanpack.
>
> Question: Does IDL have a similar routine? Do you have any suggestions
> as to how to solve a generalized eigenvalue problem in IDL?
>
> Thanks in advance,
> Tron Darvann
>
> tdarvann@lab3d.odont.ku.dk
>
>
>
>
>

Subject: Re: generalized eigenvectors
Posted by Ralf Flicker on Mon, 15 Apr 2002 21:54:25 GMT
View Forum Message <> Reply to Message

Hey Randall - this is not really related to the original question,
but since you seem to know what you're on about I figured I should
ask you.

In working on large sparse arrays it has become crucial to make the
SVD more efficient than O(n^3), but that seems to be easier said
than done. Do you know of an efficient implementation of the Laczos
bidiagonalization with selective reorthogonalization? I have not
been able to accomplish it - with complete, explicit
reorthogonalization it actually gets even worse than O(n^3).

More to the point, has anyone _ever_ managed to bring down the SVD
significantly below O(n^3) for sparse arrays? Pointers and
suggestions welcome.

aloha
ralf

Randall Skelton wrote:
>
> I am not sure I understand what you mean by a "Generalized Eigenvalue
> problem".

>
> IDL does have routines for computing the Eigenvalues and Eigenvectors.
> From the IDL help:
>
> EIGENQL - Compute eigenvectors of a real, symmetric array, given the array.
> EIGENVEC - Compute eigenvectors of a real, nonsymmetric array, given
>   the array and its eigenvalues.
> ELMHES   - Reduce a real, nonsymmetric array to upper-Hessenberg form.
> HQR   - Compute the eigenvalues of an upper-Hessenberg array.
> TRIQL   - Compute eigenvalues and eigenvectors of a real, symmetric,
>   tridiagonal array.
> TRIRED   - Use Householder's method to reduce a real, symmetric array to
>   tridiagonal form.
>
> Most of these methods are described in the Numerical recipies books.
> See http://www.ulib.org/webRoot/Books/Numerical_Recipes/
>
> In IDL the user must decide if the input matrix is symmetric or not then
> use the appropriate tools. The Matlab EIG function basically uses the
> same tools (to a first order), but automatically determines the "best"
> method based on your input matrix... I personally find Matlab's auto-magic
> approach to be more trouble than it is worth. Moreover, it promotes the
> idea that you don't really need to understand the numerical problem you
> are trying to solve...
>
> The general approach is:
>
> - If your matrix is real and symmetric, convert to the tridiagonal form
> (TRIRED) and then use the QR procedure (TRIQL) to iteratively find the
> eigenvalues/vectors from the tridiagonal array.
>
> - If your matrix is real and not symmetric, reduce to Hessenberg form
> using the Householder's transformation method (ELMHES) and then use the QR
> procedure (HQR) to get the eigenvalues/vectors of the upper Hessenberg
> matrix.
>
> - Unfortunately, there is no built in IDL code for the QZ (Golub and Van
> Loan, 1989) algorithm which can be modified to work for complex matricies.

Subject: Re: generalized eigenvectors
Posted by the_cacc on Tue, 16 Apr 2002 09:57:07 GMT
View Forum Message <> Reply to Message

Ralf Flicker <rflicker@gemini.edu> wrote in message
news:<3CBB4C11.BF62E5D5@gemini.edu>...
>
> (snip)

>
> More to the point, has anyone _ever_ managed to bring down the SVD
> significantly below O(n^3) for sparse arrays? Pointers and
> suggestions welcome.
>

Search SVDPACKC on google. It has a number of sparse SVDs.

I downloaded the package and tried to get it working... but failed.
However, the author assured me the algorithms do work.

Subject: Re: generalized eigenvectors
Posted by Randall Skelton on Tue, 16 Apr 2002 10:48:37 GMT
View Forum Message <> Reply to Message

On Mon, 15 Apr 2002, Ralf Flicker wrote:

> In working on large sparse arrays it has become crucial to make the
> SVD more efficient than O(n^3), but that seems to be easier said
> than done. Do you know of an efficient implementation of the Laczos
> bidiagonalization with selective reorthogonalization? I have not
> been able to accomplish it - with complete, explicit
> reorthogonalization it actually gets even worse than O(n^3).

I have had this same problem in the past when attempting to solve large Jacobian matrices. I haven't tried to implement the algorithm you described, but I have definitely thought about trying it. I probably won't get back to working on my code that relies on SVD for another month or so but I seem to recall that the SVD was near the top of my "need to optimize" list. Sorry I can't be of more help but at the moment CPU cycles are cheaper than my time is... If you find anything in the mean time, please post it!

> More to the point, has anyone _ever_ managed to bring down the SVD
> significantly below O(n^3) for sparse arrays? Pointers and
> suggestions welcome.

You may want to look at http://www.nersc.gov/research/SIMON/trlan.html and the links therein. Unfortunately, the code is f90 so you'll need a compiler as well as the BLAS and LAPACK libraries. If (read: when) I get around to needing this, I will probably write a set of C DLM wrappers around the public f90 functions.

Cheers,
Randall

Subject: Re: generalized eigenvectors
Posted by mvukovic on Tue, 16 Apr 2002 15:11:59 GMT
View Forum Message <> Reply to Message

Randall Skelton <rhskelto@atm.ox.ac.uk> wrote in message
news:<Pine.LNX.4.33.0204152159390.12810-100000@mulligan.atm.ox.ac.uk>...

Randall,

the generalized eigenvalue problem involves two matrices, while the
routines you suggest will solve the ``ordinary'' eigenvalue problem
that deals with one matrix only. Take a look at the original post
(included below), and you will see what I mean. (BTW, that is about
the extent of my expertise on the subject).

Mirko
... stuff deleted
>> Description of the problem:
>> I need to find the eigenvalues and eigenvectors of
>> Ax = kBx
>> where both A and B are nXn matrices and k is a scalar.

Subject: Re: generalized eigenvectors
Posted by Ralf Flicker on Wed, 17 Apr 2002 23:57:06 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:
>
> On Mon, 15 Apr 2002, Ralf Flicker wrote:
>
>> In working on large sparse arrays it has become crucial to make the
>> SVD more efficient than O(n^3), but that seems to be easier said
>> than done. Do you know of an efficient implementation of the Laczos
>> bidiagonalization with selective reorthogonalization? I have not
>> been able to accomplish it - with complete, explicit
>> reorthogonalization it actually gets even worse than O(n^3).
>
> I have had this same problem in the past when attempting to solve large
> Jacobian matrices. I haven't tried to implement the algorithm you
> described, but I have definitely thought about trying it. I probably
> won't get back to working on my code that relies on SVD for another month
> or so but I seem to recall that the SVD was near the top of my "need to
> optimize" list. Sorry I can't be of more help but at the moment CPU
> cycles are cheaper than my time is... If you find anything in the mean
> time, please post it!

Just to follow up on this - I still haven't found anything that

could be of any help to me, but I tested the LANSVD.m Lanczos bidiagonalization implementation from the PROPACK package for matlab (http://soi.stanford.edu/~rmunk/PROPACK/).

For computing only a few singular values, the efficiency for my test case (taken from an adaptive optics wavefront reconstruction application) is improved by an order of magnitude. Computing the full SVD, however, the LANSVD still scales as O(n^3), and with a larger factor, so it's actually a lot slower. And you loose some orthogonality besides. Not good.

I put a plot of it at http://www.astro.lu.se/~ralf/stuff/lansvd.pdf. If it's a bit messy it's because I had to take a crash course in matlab just to test this stuff.

ralf
[back to idl...]
