
Subject: Idea for an IDL numerical toolbox

Posted by [Liam Gumley](#) on Mon, 12 Apr 1999 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

In light of some recent discussions about calling FORTRAN routines from within IDL, I started thinking about how one might go about putting together an IDL toolbox based on a well-known freely available FORTRAN library such as LAPACK (<http://www.netlib.org/lapack>).

One reason to do this would be to extend the numeric capabilities of IDL by making use of a very well tested FORTRAN library that is available in optimized form for many architectures, e.g.

SGI: <http://www.sgi.com/software/scsl.html>

Sun:

<http://www.sun.com/workshop/performance/wp-perflib/capabilities.html>
and is also freely available in source form (see LAPACK link above).

Another reason would be to help combat a view I've heard from some Matlab users, which is that IDL is not well suited to intensive numerical computations.

Some items of information that I think could be relevant:

(1) It seems like there would be a fair amount of work involved in writing the C wrappers and IDL interface routines for a large FORTRAN library. Could this be automated somehow? NAG markets something called the NAGWare Gateway Generator, which is described at <http://www.nag.co.uk/nagware/NN.html>. Here's a quote:

"The NAGWare Gateway Generator (NGG) produces an interface that seamlessly integrates Fortran subprograms with MATLAB. It analyses the source code of a Fortran routine and automatically generates a MATLAB gateway. This allows the user to access the Fortran subprogram as if it were any other MATLAB command. Both forward and reverse gateways can be generated. Reverse gateways allow the Fortran program to access MATLAB commands and scripts."

There is a very nice freely available FORTRAN analyzer named ftnchek, available at <http://www.dsm.fordham.edu/~ftnchek> which will automatically extract information about the arguments of a FORTRAN subroutine. Using this tool it shouldn't be too difficult to build a script that automatically creates a C wrapper and IDL interface routine for a FORTRAN subroutine.

(2) There are a number of well documented and tested freely available FORTRAN numerical libraries built around LAPACK, one of which is "The Control and Systems Library SLICOT", available at <http://www.win.tue.nl/niconet/NIC2/slicot.html>

(3) Some work would be needed to make the compiling options work seamlessly on all IDL platforms (Unix at least). Right now it seems like the first time someone tries to link IDL with FORTRAN, there's a steep learning curve. I'm thinking of a package that can be downloaded, installed once, and then used directly from IDL without the user having to worry about how it was compiled or linked. The package would be smart enough to check (at installation time?) for the presence of optimized LAPACK on the host architecture, and if it was not present, LAPACK would be compiled from source.

Sounds like a nice cooperative project for the readers of comp.lang.idl-pvwave.

Any comments? Volunteers?

Cheers,
Liam.

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
<http://cimss.ssec.wisc.edu/~gumley>

Subject: Re: Idea for an IDL numerical toolbox
Posted by [Liam Gumley](#) on Tue, 13 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hey Stein,

I'm glad to see I got you fired up.

Stein Vidar Hagfors Haugan wrote:

- > One of the most difficult tasks is probably to figure out
- > which parameters are input vs output, and thus what kind of
- > initialization should be applied. That's why I'm including all
- > comments from the fortran subroutine header in the C wrapper
- > file.

While ftnchek can help in figuring out the type and size of arguments, e.g.

```
ftnchek -makedcls=4 test.f
```

you correctly point out that deciding which arguments are input vs. output is tricky. It seems to me there would have to be some simple rules, e.g.

(1) There is always an IDL wrapper for the routine being called (the IDL wrapper calls a C wrapper, which calls the FORTRAN routine).

(2) The IDL wrapper routine checks each of the arguments passed to it for existence, type, and size:

- If an argument is undefined, it must be output. So the argument is created, according to the requirements of the particular FORTRAN routine.

- If an argument is defined, it must be input. So the argument is checked for the correct type and size. If it does not have the correct type and size, the wrapper program generates an error message.

The description of the NAG Gateway Generator talks about a GUI front end which allows the user to customize the input and output argument definitions - we could do one in IDL!

FORTRAN allows subroutines and functions, much like IDL. Functions are similar to IDL in that they return a variable of a particular type.

Here's a very simple example:

```
FUNCTION SQUARE( X )  
REAL SQUARE, X  
SQUARE = X * X  
END
```

Just like in IDL, the name of the function is treated as a variable.

FORTRAN subroutines and functions can accept variables and function names as arguments.

- > Yep. Using RSI's makefiles as a template by replacing "their"
- > file names with yours is a good start, though. To circumvent
- > RSI's needlessly strict copyrights on these files (it only
- > hurts themselves) one could distribute patches only..?

Well we could always create our own makefiles, and just distribute compiled libraries. That way users don't have to wait while LAPACK compiles on their box (they might not have a FORTRAN compiler anyway).

- > Volunteers? Who, me? No! ...aaaaaaaargh....I'm losing control
- > over my fingers..... Ok - watch this space for an updated
- > version of "dlmform" - Version 1 follows below.

I'll give your perl script a go.

Cheers,
Liam.

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
<http://cimss.ssec.wisc.edu/~gumley>

Subject: Re: Idea for an IDL numerical toolbox
Posted by [steinhh](#) on Tue, 13 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <37124B1B.73391DCB@ssec.wisc.edu> Liam Gumley
<Liam.Gumley@ssec.wisc.edu> writes:

> In light of some recent discussions about calling FORTRAN routines from
> within IDL, I started thinking about how one might go about putting
> together an IDL toolbox based on a well-known freely available FORTRAN
> library such as LAPACK (<http://www.netlib.org/lapack>).

I couldn't help it, either.... It's a really intriguing
thought.

> Another reason would be to help combat a view I've heard
> from some Matlab users, which is that IDL is not well suited
> to intensive numerical computations.

And this is partly true, with the current lack of such
libraries.

> Some items of information that I think could be relevant:
>
> (1) It seems like there would be a fair amount of work
> involved in writing the C wrappers and IDL interface
> routines for a large FORTRAN library. Could this be
> automated somehow?

Uh - I spent yesterday investigating this - - the best thing I
could come up with was a simple yet ugly perl script (I'm by
no means a perl expert) geared towards the functions written
by Shanjie Zhang and Jianming Jin which were mentioned here
yesterday (<http://iris-lee3.ece.uiuc.edu/~jjin/specfunc.html>).

The script (attached at the end of this post) processes one of
their *.for files to produce:

a *.f file (with the main program commented out),
a *.c file with stubs for the C wrappers, including:
All comments from the fortran subroutine header
Decl./assignment of all parameters' variable pointers
Best guesses as to the type/dimensions of parameters

For each variable, a block of IDL_EXCLUDE/IDL_ENSURE macro calls to be edited by a programmer.

An appropriate IDL_Load() routine to install the encountered procedures.

a *.dlm file that'll make the functions visible as system routines within IDL.

The *.c file will not be overwritten if it already exists.

Only the *.c file will need to be edited by the user.

My knowledge of fortran is *minimal*, so it's probably not difficult to make routines that break this script. Does fortran have *functions*, btw? My script assumes procedures only...

The script does (sort of) handle implicit typing, whenever it's spelled out in the program (not the default stuff, because I know too little about it...:-)

> NAG markets something called the NAGWare Gateway Generator,
> which is described at <http://www.nag.co.uk/nagware/NN.html>.

Though this is a commercial package.... which of course also will promote the sales of the NAG software itself... just a hint on how much work needs to be spent to develop a *fully* automatic scheme.. But it could (should?) be in RSI's interests to do something like this!

One of the most difficult tasks is probably to figure out which parameters are input vs output, and thus what kind of initialization should be applied. That's why I'm including all comments from the fortran subroutine header in the C wrapper file.

> There is a very nice freely available FORTRAN analyzer named
> ftnchek, available at <http://www.dsm.fordham.edu/~ftnchek>
> which will automatically extract information about the
> arguments of a FORTRAN subroutine. Using this tool it
> shouldn't be too difficult to build a script that
> automatically creates a C wrapper and IDL interface routine
> for a FORTRAN subroutine.

THANKS! Wow - why didn't I know about this yesterday! No problems installing it on Digital Unix, btw.

> (2) There are a number of well documented and tested freely
> available FORTRAN numerical libraries built around LAPACK...

- > (3) Some work would be needed to make the compiling options work
- > seamlessly on all IDL platforms (Unix at least). Right now it seems like
- > the first time someone tries to link IDL with FORTRAN, there's a steep
- > learning curve.

Yep. Using RSI's makefiles as a template by replacing "their" file names with yours is a good start, though. To circumvent RSI's needlessly strict copyrights on these files (it only hurts themselves) one could distribute patches only..?

- > I'm thinking of a package that can be downloaded,
- > installed once, and then used directly from IDL without the user having
- > to worry about how it was compiled or linked. The package would be smart
- > enough to check (at installation time?) for the presence of optimized
- > LAPACK on the host architecture, and if it was not present, LAPACK would
- > be compiled from source.

Sounds nice!

- > Sounds like a nice cooperative project for the readers of
- > comp.lang.idl-pvwave.

Sounds very nice, indeed.

- > Any comments? Volunteers?

Volunteers? Who, me? No! ...aaaaaaaargh....I'm losing control over my fingers..... Ok - watch this space for an updated version of "dlmform" - Version 1 follows below.

Regards,

Stein Vidar

dlmform:-----

#!/local/bin/perl

```
#####
# write_header(CFILE);
#####
sub write_header {
    local (*FILE) = $_[0];
    print FILE <<HEADER_STOP;
#include <unistd.h>
#include <stdio.h>
```

```

#include "export.h"

#define NULL_VPTR ((IDL_VPTR) NULL)

#define GETVARDATA(v,n) ((v->flags&IDL_V_ARR) \
    ? (*(n) = v->value.arr->n_elts, v->value.arr->data) \
    : (*(n) = 1, & v->value.c ) )

HEADER_STOP
}

#####
# write_funcheader(CFILE,funcname)
#####

sub write_funcheader {
    local (*FILE,$funcname) = @_ ;
    print FILE <<FUNCHEADER_STOP;
    IDL_VPTR \U$funcname\E(int argc, IDL_VPTR argv[])
    {
        FUNCHEADER_STOP
    }
}

#####
# write_sysfun_defs(CFILE,@funs,@nargs)
#####

sub write_sysfun_defs {
    local (*FILE,*funs,*nargs) = @_ ;

    local (@nparms) = @nargs;

    print FILE <<LOADSTOP;
    int IDL_Load(void)
    {
        static IDL_SYSFUN_DEF proc_def[] = {
        LOADSTOP

        foreach $fun (@funs) {
            $narg = shift(@nparms);
            $comma = $#nparms+1 ? ", " : "";

```

```

print FILE << ;
{(IDL_FUN_RET) \U$fun\E, "\U$fun\E", $narg, $narg}$comma

}
print FILE <<LOADSTOP2
};

return IDL_AddSystemRoutine(proc_def,FALSE,$#funs+1);
}
LOADSTOP2

}

```

```

#####
# write_dlm_file(DLMFILE,$module,@funs,@nargs)
#####
sub write_dlm_file {
    local (*FILE,$module,*funs,*nargs) = @_ ;
    local (@nparms) = @nargs;
    local ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
gmtime(time);

    print FILE <<DLM_STOP;
# Automatically generated from Fortran source file
MODULE \U$module\E
DESCRIPTION Fortran function(s): @funs
BUILD_DATE $mday $mon $year
SOURCE Perl script
DLM_STOP

    foreach $fun (@funs) {
    $narg = shift(@nparms);
    print FILE "PROCEDURE \U$fun\E $narg $narg\n";
    }
}

```

```

##### #####
# Parses possible declaration a la "CHARACTER P(155)*4,P2(10)"

# 1st arg is input line to parse
# 2nd arg is e.g. "CHARACTER"
# 3rd arg is the type ("" if 2nd arg is DIMENSION)
# 4th arg is *var
# 5th arg is *type
# 6th arg is *dim

```

```

sub parselist

```



```

{
  local($_,$srch,$thistype,*var,*type,*dim) = @_;

  if ( /(b$srch\s*)/ ) {

$matchsofar = $1;
while (/(($matchsofar([^\n, ]+),*)/ ) {
  $matchsofar = $1;
  $decl = $2;
  $matchsofar =~ s/(\\)/g; # Escape (
  $matchsofar =~ s/(^)/g; # Escape ^
  $matchsofar =~ s/(\\)/g; # Escape \
  $decl =~ /(w+)\b([^\s]*)/;
  $thisvar = $1;
  $thisdim = $2;
  VARLOOP: for ($j=0; $j<=$#var; $j++) {
if ( $var[$j] =~ /^$thisvar$/ ) {
  $type[$j] = $thistype if $type[$j] eq "";
  $dim[$j] = $thisdim if $dim[$j] eq "";
}
}
}
}
}
}

```

LINE:

```

while (<>) {

  # NEW FILE:
  if ($ARGV ne $oldargv) {
$oldargv = $ARGV;
$module = $ARGV;
$module =~ s/\.for//;

open (FORTRAN,">$module.f");
select(FORTRAN);

if ( !-f "$module.c" ) {
  open (CFILE,">$module.c");    # Open C file
  do write_header(CFILE);
}

open (DLMFILE,">$module.dlm"); # Open DLM file

# Zero lists.
@funs = ();
@nargs = ();

```

```

do {
  # Main program should be commented out!
  s/^(^C)/C$1/;
  print;
  $_ = <>;
} until (/^END\b/);
s/^(^C)/C$1/;
print;
next LINE;
}

print;

if (/^C/) {
s/^C//;
s/^    //;
s/\n//;
$commmlen = length if $commmlen < length;
$_ .= " " while length lt $commmlen;
@comments = (@comments,"/* $_ */\n");
next LINE;
}

# Find SUBROUTINE declarations & parameters.

($sn,$var,$r) = /\bSUBROUTINE\b\s*(\w+)\s*\((\w+),?(.*)/;
if ($sn) {
$subname = $sn;
$_=$r;
@comments = (); # Initialize lists of information for this routines
$commmlen = 10;
@var = ();
@type = ();
@dim = ();
@implicit = ();
@implicitl = ();

while ($var) {
  @var = (@var,$var);
  @type = (@type,"");
  @dim = (@dim,"");
  ($var,$_)= /\s*(\w+),?(.*)/;
}
}

# Find IMPLICIT declarations for this routine.

```

```

($type,$letters) = /\bIMPLICIT\s*(.*)\s*\((.*)/;
if ($type) {
$letters =~ s/\)/;/ # Take away end parenthesis
$letters =~ s/,/;/ # Take away commas to use as eg [A-FH-J]
@implicit = (@implicit,$type);
@implicitl = (@implicitl,$letters);
}

do parselist($_, "CHARACTER", "CHARACTER", *var, *type, *dim);
do parselist($_, "DIMENSION", "", *var, *type, *dim);

# Print out routine stub if it's the end of the routine

if (\bEND\b/ & !eof(CFILE)) {

print CFILE @comments;

print CFILE "\n";

do write_funcheader(CFILE,$subname);

@funs = (@funs,$subname);
@nargs = (@nargs,$#var+1);

for ($i=0; $i<=$#var; $i++) {
    $lvar = $var[$i];
    $type = $type[$i];
    $dim = $dim[$i];
    @letters = @implicitl;

    foreach $imp (@implicit) {
    $letters = shift( @letters );
    if ($type eq "" && $lvar =~ /\^[${letters}]/) {
        $type = $imp;
    }
    }
    print CFILE " IDL_VPTR $var[$i]=argv[$i]; ";
    print CFILE "/* $lvar : $type : $dim    *\n";
}

print CFILE "\n\n";

for ($i=0; $i<=$#var; $i++) {
    $lvar = $var[$i];
    $type = $type[$i];
    $dim = $dim[$i];

```

```

    print CFIL " IDL_EXCLUDE_UNDEF($lvar);\n";
    print CFIL " IDL_EXCLUDE_EXPR($lvar);\n";
    print CFIL " IDL_ENSURE_SIMPLE($lvar);\n";
    print CFIL " IDL_ENSURE_SCALAR($lvar);\n" if $dim eq "";
    print CFIL " IDL_ENSURE_ARRAY($lvar);\n" if $dim ne "";
    print CFIL " IDL_EXCLUDE_COMPLEX($lvar);\n"
if $type !~ /COMPLEX/;
    print CFIL " IDL_EXCLUDE_STRING($lvar);\n";

    print CFIL "\n\n";
}
print CFIL "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" if !eof(CFIL);

}

^bEND\b/ && ($main = 0);

if (eof) {
do write_sysfun_defs(CFIL,*funs,*nargs);
do write_dlm_file(DLMFILE,$module,*funs,*nargs);
close CFIL;
close DLMFILE;
}

}

```

Subject: Re: Idea for an IDL numerical toolbox
Posted by [rmlongfield](#) on Tue, 13 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <37124B1B.73391DCB@ssec.wisc.edu>,
 Liam Gumley <Liam.Gumley@ssec.wisc.edu> wrote about simplifying the
 Fortran-IDL interface.

Hi Liam, That was a really nice summary of what needs to be done in
 order to make such a tool. Although I would find it interesting to work on
 such a tool (mostly to learn something new), I do not have the time to
 devote to it. It looks like a full time job to me.

Just a short note.
 Rose

-----== Posted via Deja News, The Discussion Network ==-----
<http://www.dejanews.com/> Search, Read, Discuss, or Start Your Own

Subject: Re: Idea for an IDL numerical toolbox
Posted by [Dr. G. Scott Lett](#) on Wed, 14 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

I believe Michel Olagnon has a program that generates Fortran 90 interface blocks, with INTENT() from fortran code. If I understand your problem, this should help with a solution.

Try:

<http://www.ifremer.fr/ditigo/molagnon/fortran90/contenu.html>

Liam Gumley <Liam.Gumley@ssec.wisc.edu> wrote in message
news:371357CE.BA2ACD10@ssec.wisc.edu...

> Hey Stein,
> <much good stuff snipped>
>
> you correctly point out that deciding which arguments are input vs.
> output is tricky. It seems to me there would have to be some simple
> rules, e.g.
>

Subject: Re: Idea for an IDL numerical toolbox
Posted by [steinhh](#) on Wed, 14 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Ok, now I think I have it....

With the assistance of the ftnchek utility, I have now successfully generated several DLMS from f77 source files.

The script assumes that a *.for file and its *.dcl file (generated with ftnchek) is to be converted into a *.c file, a *.f file (with no main program), and a *.dlm file.

Compiling the .c file and the .f file together into a shareable object file makes the thing available for IDL, if the IDL_DLM_PATH includes the directory.

It's a pretty "dumb" perl script, but it does the job for many applications -- more info later (if I have the time, which I really don't!), but for now you can take a look at

<http://www.astro.uio.no/~steinhh/idl/dlmform>

Regards,

Stein Vidar

Subject: Re: Idea for an IDL numerical toolbox
Posted by [davidf](#) on Fri, 16 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Mr.Stein Vidar Hagfors Haugan
University of Oslo
Oslo, Norway

Dear Mr. Haugan:

This is to inform you that you have been nominated and are a finalist for the 1999 Coyote's Howl Award. As you may or may not know, this award is given annually (or whenever the committee gets around to it) to the person who has made the most significant contribution to the IDL newsgroup.

The person who nominated you had this to say about your contribution:

"God bless, Stein Vidar. I wouldn't have a clue how to link my FORTRAN code without him!"

This anonymous person also pointed out that your solution of the LOG2 problem saved him weeks of work.

On behalf of the committee, let me extend to you our sincerest gratitude and thanks for a job well done.

And on a personal note, let me assure you that even though you have won the award five of the last seven times it has been given, the committee will not hold that against you. Although it is still early in the year, I feel certain that your chances for another trophy are excellent.

I will be in touch again just as soon as the votes have been counted.

Very sincerely yours,

Dr. Wily J. Coyote
Coyote's Coordinating Committee Chairman (C^4)

Subject: Re: Idea for an IDL numerical toolbox
Posted by [steinhh](#) on Fri, 16 Apr 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

..and now for the final version.

I'll even produce array- crunching system routines out of all-
scalar fortran functions.

See <http://www.astro.uio.no/~steinhh/idl/dlmform.html>

Regards,

Stein Vidar

Subject: Re: Idea
Posted by on Sun, 15 Jun 2008 08:21:26 GMT
[View Forum Message](#) <> [Reply to Message](#)

1/ Muistamme miten aikoinaan asbesti oli tie loistavaan tulevaisuuteen.

Miljoonat jarrupalat olivan mailman turvan tae, jne.. Asbesti oli
kunnes... ..!

liuotinmaailmat, kuin sen TARKOIN tiedettyyn turvallisuuteen ja

kunnes!

kaupata ja suorittaa myynit. Niin oli DDT pelastava koko kurjan maailman.

..!

4/Maailman energiamonopoliin kuningasajatus oli vaikka nyt
nakutuksenestoaine lyijyinen hermomyrkky. Toki jo Rokkefellerin

koska JUURI vaarallisuus takasi sen, ettei kukaan nyksityinen voisi

kaupunkilaisnuorissa 70-luvun energiakriisin alta tavattiin huikeaa 30%

kasautumisista biotooppeihin kohtalokkaasti ja joku Nobelisti pari kertoili

valtiotoimin vaiettu kuoliaaksi! IAEA ei totta tosiaan STUK/SUPO:n kanssa
halua rahantuotolleen realisoijia ja haittaajia! Niin .. kunnes. .. kerran