Subject: IDL versus MATLAB : could you help me ?????
Posted by Christophe Dornier on Mon, 26 Nov 2001 11:27:23 GMT
View Forum Message <> Reply to Message

I work in a Digital Imaging Unit,
and a future orientation of this group will be to developp algorithms of
image processing.

We have the choice between two softwares which are IDL (RSI) or MATLAB
(MathWorks).
What are the advantages or inconvenients of these softwares.
What soft is the most cheaper ?
Is it much easier to develop in Matlab or not ?
On which soft is there available the most external libraries ?
....
Thanks in advance
best regards

--
Christophe DORNIER, PhD

Division d'Imagerie Medicale
Hopital Universitaire de Geneve
CH-1211 GENEVA 14
SWITZERLAND

phone: 41 22 372 62 71
fax: 41 22 372 61 98
e-mail: christophe.dornier@dim.hcuge.ch

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Richard Tyc on Tue, 27 Nov 2001 15:19:58 GMT
View Forum Message <> Reply to Message

Well, I have worked a bit with MatLab. Not being a an expert in either,
here are my two bits:

1. I don't believe Matlab has anything like object graphics in IDL. I would
consider this to be a major weakness of Matlab.

2. It does not take advantage of multiple CPU's. Before IDL 5.5, the Volume
object had the ability to use multple CPU's to perform volume rendering. Now
in 5.5, IDL is multi-threaded taking further advantage of it.

3. Matlab has excellent C/C++ functionality. In fact, you can convert your
Matlab code into C and distribute it as a C program (with Matlab libraries

bundled in) if you wish (ie. no runtime license needed for others to try
your code) - something which IDL cannot do.

4. If you're into medical imaging, the new Watsyn product from RSI maybe
something of great interest leaning you toward IDL. I will be looking
closely at this myself.

Rich

Christophe Dornier <christophe.dornier@dim.hcuge.ch> wrote in message
news:3c0226d8@news.unige.ch...
> I work in a Digital Imaging Unit,
> and a future orientation of this group will be to developp algorithms of
> image processing.
>
> We have the choice between two softwares which are IDL (RSI) or MATLAB
> (MathWorks).
> What are the advantages or inconvenients of these softwares.
> What soft is the most cheaper ?
> Is it much easier to develop in Matlab or not ?
> On which soft is there available the most external libraries ?
> ....
> Thanks in advance
> best regards
>
>
> --
> Christophe DORNIER, PhD

> Division d'Imagerie Medicale
> Hopital Universitaire de Geneve
> CH-1211 GENEVA 14
> SWITZERLAND
>
> phone: 41 22 372 62 71
> fax: 41 22 372 61 98
> e-mail: christophe.dornier@dim.hcuge.ch
>
>
>

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by David Fanning on Tue, 27 Nov 2001 15:33:25 GMT

View Forum Message <> Reply to Message

Richard Tyc (richt@sbrc.umanitoba.ca) writes:

> 3. Matlab has excellent C/C++ functionality. In fact, you can convert your
> Matlab code into C and distribute it as a C program (with Matlab libraries
> bundled in) if you wish (ie. no runtime license needed for others to try
> your code) - something which IDL cannot do.

Is it still true that this can only be done
with non-GUI code?

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Paul van Delst on Tue, 27 Nov 2001 16:06:24 GMT
View Forum Message <> Reply to Message

Richard Tyc wrote:
>
> Well, I have worked a bit with MatLab. Not being a an expert in either,
> here are my two bits:
>
> 1. I don't believe Matlab has anything like object graphics in IDL. I would
> consider this to be a major weakness of Matlab.

Huh - I thought *all* of the graphics capabilities in Matlab were implemented as objects. It's
just done in such a fashion as to be totally transparent to the user. I've seen people do stuff
on the command line in Matlab that would take pages of code in IDL object graphics (at least as
I understand them....which is not a lot).

But I used Matlab last a long time ago so what the hell would I know. :o)

paulv

--
Paul van Delst   Religious and cultural
CIMSS @ NOAA/NCEP   purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545   V.S.Naipaul

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Steve Eddins on Tue, 27 Nov 2001 16:10:11 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:

> Richard Tyc (richt@sbrc.umanitoba.ca) writes:
>
>> 3. Matlab has excellent C/C++ functionality. In fact, you can convert your
>> Matlab code into C and distribute it as a C program (with Matlab libraries
>> bundled in) if you wish (ie. no runtime license needed for others to try
>> your code) - something which IDL cannot do.
>
> Is it still true that this can only be done
> with non-GUI code?

No, that limitation has been removed --- about 2 years ago, I think.

[I removed comp.lang.idl from the list of newsgroups since that group discusses
a different "IDL."]

Steve

--
Steve Eddins
The MathWorks, Inc.
eddins@mathworks.com
http://www.mathworks.com

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by mfein on Tue, 27 Nov 2001 16:51:26 GMT
View Forum Message <> Reply to Message

On Tue, 27 Nov 2001 11:06:24 -0500, Paul van Delst
<paul.vandelst@noaa.gov> wrote:

> Richard Tyc wrote:
>>
>> Well, I have worked a bit with MatLab. Not being a an expert in either,
>> here are my two bits:
>>
>> 1. I don't believe Matlab has anything like object graphics in IDL. I would
>> consider this to be a major weakness of Matlab.
>
> Huh - I thought *all* of the graphics capabilities in Matlab were implemented as objects. It's
> just done in such a fashion as to be totally transparent to the user. I've seen people do stuff
> on the command line in Matlab that would take pages of code in IDL object graphics (at least as

> I understand them....which is not a lot).
>

Not sure what you mean by this. -Properties- in Matlab graphics have
an object-oriented flavor-- graphic 'objects' are located in a
hierarchy where properties are encapsulated at various levels.
However, that's really all there is to it (IMO).

More generally, Matlab doesn't have pointer variables or event
management so 'object-orientation' in Matlab has definite limits.
Effectively, object-orientation in Matlab is a way of extending the
type system in a transparent fashion. Very useful, but not the whole
nine yards.

Matt Feinstein   does not include his email address
                 in the text of usenet postings.

Harvard Law of Automotive Repair: Anything that goes away
by itself will come back by itself.

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Roy E Hansen on Sun, 02 Dec 2001 11:48:19 GMT
View Forum Message <> Reply to Message

Christophe Dornier wrote:

> I work in a Digital Imaging Unit,
> and a future orientation of this group will be to developp algorithms of
> image processing.
>
> We have the choice between two softwares which are IDL (RSI) or MATLAB
> (MathWorks).
> What are the advantages or inconvenients of these softwares.
> What soft is the most cheaper ?
> Is it much easier to develop in Matlab or not ?
> On which soft is there available the most external libraries ?

Here are some of my personal opinion on Matlab/IDL strong/weak points
(note that I have only been using MatLab/IDL for Radar/sonar signal
processing
and numerical modelling, NOT image processing).

MatLab weak points / IDL strong points:

1) MatLab does not have data types: Everything is Double presision. This is
very
   memory and computational intensive. Matlab can read different data types,

but as
  soon as you add two variables, they are converted to double. IDL supports
all
  data types. Especially DSP (ie FFT) routines in single precision do I
miss in matlab.

2) MatLab IO routines can only read data files up to 2GB, while IDL supports
data files
  up to 64 bit address space.

3) IDL have very efficient IO routines and does understand arbitrary
file/data formats.
  Matlab IO routines are OK for simple formats. However, for mixed data
types (i.e. a
  typical radar data file could contain a start record with strings,
integers and floats
  specifying the system parameters, then data records of arbitrary length,
which is specified
  in the header) you'll soon find out that writing your own IO routine in
C/C++ reduces the
  read time by orders of magnitude.

4) IDL have keyword parameters, MatLab have not. This is really a clean way
of using variable
  number of input parameters to functions, which maintain a high level of
readability.

5) IDL have better/more object orientation than MatLab (I've never really
used this for any
  thing seriously, so I wouldnt know how much this means).

6) IDL 5.5 have multithreading. MatLab does not.

MatLab strong points / IDL weak points:

1) MatLab have toolboxes for everything (this may not be only positive,
since these toolboxes
  cost very much). For me, the built-in functionality for signal processing
in MatLab, far
  exceeds what IDL have (even with the free librarys around).

2) MatLab graphics is far more user friendly than IDL direct graphics. Up to
IDL 5.4 that
  also includes IDL object graphics.

3) MatLab has excellent C/C++ interfacing both ways.

4) MatLab have functionality for variable number of output parameters.

5) More people use MatLab.

I only have experiense with IDL up to 5.4 and Matlab 6
(I have changed from IDL to MatLab solely due to change of company).

-Roy

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Mark Hadfield on Sun, 02 Dec 2001 21:28:34 GMT
View Forum Message <> Reply to Message

From: "Roy E Hansen" <ro-edgah@online.no>

Thanks for the thorough comparison Roy.

> MatLab strong points / IDL weak points:
>
> 4) MatLab have functionality for variable number of output parameters.

I'll just elaborate on that one for those who don't know both packages.

Matlab has the ability to pass more than one variable from a function via
the return value. Thus the standard function griddata can be called with
either of the following syntaxes (from the griddata documentation):

```
   ZI = griddata(x,y,z,XI,YI)
   [XI,YI,ZI] = griddata(x,y,z,xi,yi)
```

In the former a single array (sorry, matrix) is returned; in the latter
three ,matrices are returned, ZI containing z-grid data in both cases, and
XI and YI containing ancillary x- and y-grid data.

This is cool.

However there is a deeper reason why Matlab needs this facility and IDL can
get by without it: Matlab passes function arguments by value only, whereas
IDL passes them by reference (with important exceptions I won't go into
here). In other words, IDL can pass information out through function
arguments and Matlab can't. So if there were a griddata in IDL (there isn't
but it would be easy to write) you would invoke it something like this
(remembering that IDL is not case-sensitive):

```
   zgrid = griddata(x,y,z,xi,yi,XGRID=xgrid,YGRID=ygrid)
```

Here zgrid receives the return value of the function and xgrid and ygrid
optionally receive the ancillary data.

There are pros and cons to the "reference-vs-value" issue. Passing by
reference should be significantly more efficient in some cases because it
reduces unnecessary copying of data. It is my impression that (for this
reason & others) IDL is a much better tool when you want to deal with really
large arrays. On the other hand IDL users are occasionally bitten by
routines unexpectedly changing arguments (or unexpectedly failing to change
arguments). When writing IDL routines you should always be aware that your
arguments belong to the caller and not to you.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Nabeel on Mon, 03 Dec 2001 00:31:38 GMT
View Forum Message <> Reply to Message

Hi,

Just to clarify one point here:

> 4) IDL have keyword parameters, MatLab have not. This is really a clean way
> of using variable
>   number of input parameters to functions, which maintain a high level of
> readability.

I'm not sure what you're referring to here, but MATLAB does let you pass
in a variable number of inputs to a file, as well as produce a variable
number of outputs. The VARARGIN and VARARGOUT keywords are used for
this.

-- Nabeel

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Roy Edgar Hansen on Mon, 03 Dec 2001 08:13:26 GMT
View Forum Message <> Reply to Message

>
>> 4) IDL have keyword parameters, MatLab have not. This is really a clean way
>> of using variable
>>   number of input parameters to functions, which maintain a high level of
>> readability.
>
> I'm not sure what you're referring to here, but MATLAB does let you pass
> in a variable number of inputs to a file, as well as produce a variable
> number of outputs. The VARARGIN and VARARGOUT keywords are used for
> this.

Well, in MatLab you must either keep track of the order of input parameters, or
you have
to parse the input manually (of type which is done in the plotting routines:
'XLim', [10,20]).
Say you have a number of optional parameters, and only want to specify parameter
N.
Either you have to specify all parameters before N, or parse the input yourself.
In IDL, the keyword parameter is named and any keyword parameter can be specified

regardless of all other input parameters, i.e. y = func(x,y, KEYW=14).

In my opinion, the keyword parameter functionality is a more readable way of
using variable
number of input parameters. The only alternative in MatLab is to parse the input
manually
(which of course can be done in IDL too).

-Roy

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Nabeel on Mon, 03 Dec 2001 14:47:53 GMT
View Forum Message <> Reply to Message

Hi,

> Well, in MatLab you must either keep track of the order of input parameters, or
> you have to parse the input manually (of type which is done in the plotting routines:

Not anymore - there's a relatively common programming pattern in MATLAB
that makes use of structures to address this functionality. By
providing inputs and outputs of your function as structures, the
identification of parameters is unambiguous, and the assignment of
default values for parameters that aren't supplied becomes relatively
straightforward. Additionally, by using structures as outputs, the
function's signature is unaffected by the addition of new outputs.

A MATLAB structure could look like:

```
>>  S.param1 = 42;S.param2 = 'Nabeel';
>>  S

S =

    param1: 42
    param2: 'Nabeel'
```

I'm not sure about what other programming languages everyone here is
familiar with, but if you aren't familiar with structures you can think
of them as hashtables, with the key being specified after the "."

Here's a quick example of the inptu checking I was talking about:

```
%%%%%%%%%%%%%%%%
function outStruct = foo(inStruct)

%  input checking
defaultFields = {'param1','param2','param3'}
defaultValues = {val1,val2,val3}
givenFields = lower(fieldnames(inStruct));
%  Now assign default values to unspecified parameters
[missingParameters idx] = setdiff(defaultFields,givenFields)
for i = 1:length(missingParameters)
  inStruct =
 setfield(inStruct,missingParameters{i},defaultValues{idx(i)} );
end

%  rest of function goes here
...
%%%%%%%%%%%%%%%
```

In fact, to make things smoother, you could write an "inputcheck.m" file
which does all of this given a given structure, default fields, and
default values.

Assigning outputs to outStruct is straightforward, and if you want to
assign more outputs you can just give outStruct extra fields. The
beauty of this is that you won't need to change any preexisting function
calls. This is great for when a function's outputs are changing as you
develop a program, or for functions that produce many outputs whose
order is difficult to track.

Both of these approcaches should address the concern you brought up.

-- Nabeel

Roy Edgar Hansen wrote:
>
>>
>>> 4) IDL have keyword parameters, MatLab have not. This is really a clean way
>>> of using variable
>>> number of input parameters to functions, which maintain a high level of
>>> readability.
>>
>> I'm not sure what you're referring to here, but MATLAB does let you pass
>> in a variable number of inputs to a file, as well as produce a variable
>> number of outputs. The VARARGIN and VARARGOUT keywords are used for
>> this.
>
> Well, in MatLab you must either keep track of the order of input parameters, or
> you have
> to parse the input manually (of type which is done in the plotting routines:
> 'XLim', [10,20]).
> Say you have a number of optional parameters, and only want to specify parameter
> # N.
> Either you have to specify all parameters before N, or parse the input yourself.
> In IDL, the keyword parameter is named and any keyword parameter can be specified
>
> regardless of all other input parameters, i.e. y = func(x,y, KEYW=14).
>
> In my opinion, the keyword parameter functionality is a more readable way of
> using variable
> number of input parameters. The only alternative in MatLab is to parse the input
> manually
> (which of course can be done in IDL too).
>
> -Roy

Subject: Re: IDL versus MATLAB : could you help me ?????
Posted by Mark Hadfield on Mon, 03 Dec 2001 21:56:24 GMT

View Forum Message <> Reply to Message

From: "Nabeel" <nabeel@mathworks.com>
>> Well, in MatLab you must either keep track of the
>> order of input parameters, or you have to parse the
>> input manually (of type which is done in the plotting routines:
>

> Not anymore - there's a relatively common programming pattern in MATLAB
> that makes use of structures to address this functionality. ...

Thanks for that info, Nabeel. I am a long-time IDL user in a discipline
where Matlab is the de facto standard so every so often I try Matlab out.
Invariably I find myself stuck trying to do things in an IDL-ish way. The
issue of optional arguments is a particularly sticky area. When I explain to
colleagues what I am trying to do they invariably say, "Huh? Why would you
want to do that?" The next time I dabble with Matlab I will try out your
suggestion.

But what I would like to know is, does Matlab support anything like IDL's
keyword inheritance? It is extremely useful for wrapper functions. I don't
know if you know that means so I will try to illustrate with an example. The
IDL plot command (like Matlab's plot, i think) accepts a huge list of
optional arguments. In IDL they are implemented as keywords, eg you might
write

```
   plot, x, y, xrange=[10,20], yrange=[0,3], color=10B
```

Let's say I want a routine that, by default, plots data in red. (Let's also
assume that through a deep and mystical knowledge of IDL
colour-specification I have associated red with value 10B. Hmmm maybe this
isn't such a good example but I will persist.) Then I could write plot_red
like this

```
 pro plot_red, p1, p2, _EXTRA=extra
   plot, p1, p2, COLOR=10B, _EXTRA=extra
end
```

When plot_red is called all its keywords are bundled up (into a structure in
this case) and passed to plot. If there is no COLOR keyword amongst them
then the "COLOR=10B" keyword is passed to plot, but if there is a COLOR
keyword then it takes precedence.

This is extremely useful once you get the hang of it. They key thing is that
in defining plot_red I don't have to know or care what keywords are accepted
by plot, other than COLOR.

> I'm not sure about what other programming languages
> everyone here is familiar with, but if you aren't familiar
> with structures you can think of them as hash tables, with
> the key being specified after the "."

IDL users will find that Matlab structures are a lot like IDL ones, but
there is a significant difference. IDL structures are immutable, i.e. once
you've made one the only way to add a tag is to destroy the old structure
and create a new one. This is what the following code does:

```
    s = {param1: 42}
    s = create_struct(s, 'param2', ''Nabeel'}
```

In Matlab, you can extend an existing structure any time you like, and the syntax is simpler, eg:

```
    S.param1 = 42;
    S.param2 = 'Nabeel';
```

So Matlab wins on flexibility here. Of course, there may be efficiency advantages to IDL's approach

---
Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG