Subject: IDL as programming language?
Posted by abz on Sun, 28 Aug 1994 00:24:41 GMT
View Forum Message <> Reply to Message

OK. One the lecturers in our department recently went to Hawaii & America for a few weeks (lucky him) and came back with the idea of using IDL as a complete programming language rather than as just a plotter for FORTRAN generated data. Apparently this is being done overseas. Our department (mathematics) has been considering this possibility, but nmaking this step would be fairly costly for us, as we'd have to extend our license, so we want to be sure that we would be doing the right thing. We have a number of questions, concerning comparisons between IDL and other programming languages (particularly FORTRAN). We are currently running an older version of IDL (2.2.2) on a Sun SPARC station.

(i) Accuracy. Our current version of IDL seems to prefer doing calculations in single precision, while we prefer double. Has this been improved in the latest version? (e.g. in our current version, routines like LUDCMP work in s.p., despite being passed d.p. arguments.)

(ii) Speed. Some of us Grads are running some really time consuming programs (large arrays, large loops). How does IDL compare with (say) FORTRAN in general, speedwise? (my impression is that it's pretty slow, but I could be wrong...)

(iii) Memory. How does IDL's memory management compare? Again, some of our programs (FORTRAN) have a tendency to gobblelarge chunks of memory (probably bad programming, but still...)

(iv) What is a large IDL code like to debug?

(v) How 'robust' is IDL as a programming language? We have a variety of different programming styles here -- some prefer 'quick and dirty' programming, others a more structured approach. Forgive my possible ignorance, but I have the impression that IDL as a language is more suited to the 'quick and dirty' approach. Is this true? Does IDL as a programming language have many glitches or inconveniences from a mathematical programmers point of view?

Any info/advice would be much appreciated. The types of stuff we do here are generally large numerical (finite difference) codes on 2D and 3D grids. I'll wait a while before sending this to see if there is/has been any discussion on this topic in the newsgroup.

Please email me, or post to this newsgroup if you think anyone else will be interested.... Thanks

Alec.

Subject: Re: IDL as programming language?
Posted by thompson on Thu, 01 Sep 1994 14:10:51 GMT
View Forum Message <> Reply to Message

stl@sma.ch (Stephen Strebel) writes:

> Granted, it has no GUI oriented interface for a debugger, but I think
> its easier then debugging compiler/linker/memory and other problems that
> you might encounter in C. Its A a higher level language then C and
> fortran, so its pretty tough to compare them.

I've been told that the latest version of IDL (3.6) does have a Motif debugger on Unix systems, but I haven't tried it myself yet. One is supposed to run idltool instead of idl. Has anyone tried it yet?

Bill Thompson

Subject: Re: IDL as programming language?
Posted by wally on Thu, 01 Sep 1994 16:43:53 GMT
View Forum Message <> Reply to Message

The comments that IDL is a quick and dirty language aren't totally fair. IDL is a very powerful language that is easy to use, but that doesn't mean that code written in IDL is necessarily dirty. It has the constructs that allow you to write clear well structured programs if you take the time. If you're not willing to invest the time in good software development practices then you will get confusing unmaintainable software, but that's true in Fortran or C also. I've seen plenty of examples of spaghetti code written in Fortran. However, I will say that there is one feature of IDL that is easily abused and tends to create dirty code. IDL is not a strongly typed language like PASCAL. Not only don't you have to define variables, IDL is perfectly happy to dynamically redefine variables in the middle of a program changing an array to a scalar. For example you could have

```
a = fltarr(10,10)
```

but later if you have

```
a = 1
```

you won't get an error; a is redefined to be a scalar integer. So even if you define all your variables at the front of a program you're not guaranteed those definitions are valid. On the other hand this feature is one of the aspects of IDL that makes it so easy to use.

In addition IDL is funny in how it creates variables on the fly.
If you have

```
a = fltarr(10)
b = a
```

that's ok. On the other had if you have

```
c(0:4) = a(0:4)
```

then you get an error that c is undefined

--

Wally Gross phone : 301-286-6690
Code 971 Bldg. 22 Room 335 e-mail : wally@halfhalt.gsfc.nasa.gov
Goddard Space Flight Center
Greenbelt, MD 20771

Subject: Re: IDL as programming language?
Posted by harmonic on Wed, 07 Sep 1994 23:55:34 GMT
View Forum Message <> Reply to Message

In article <34fhs7$ak8@nkosi.well.com>,
Mike Boucher <scisoft@well.sf.ca.us> wrote:
>
> Another alternative is the IDL and PV-Wave interface to the optimized and
> parallelized math library that our company sells. Dr. Thompson is
> certainly correct that A=B*C is a faster matrix multiple than the
> equivalent nested loops, but CALL GEMM(A,B,C) is about three times
> faster than that even on a single-CPU box. On the multi-processors,
> we're unbeatable.
>
> scisoft@well.sf.ca.us for more information, end of plug.

I would be interested in seeing your timings. To date we haven't
seen any of the "matrix languages", (i.e. IDL, MATLAB, Gauss, etc.) that
do matrix multiplication faster than O-Matrix.

Much appreciated,

Beau Paisley
Harmonic Software Inc.
12223 Dayton Avenue North
Seattle WA 98133
206-367-8742
FAX: 206-367-1067

Internet e-mail: harmonic@world.std.com

Subject: Re: IDL as programming language?
Posted by scisoft on Fri, 09 Sep 1994 21:11:45 GMT
View Forum Message <> Reply to Message

Beau Paisley (harmonic@world.std.com) wrote:
: In article <34fhs7$ak8@nkosi.well.com>,
: Mike Boucher <scisoft@well.sf.ca.us> wrote:
: >
: >Another alternative is the IDL and PV-Wave interface to the optimized and
: >parallelized math library that our company sells. Dr. Thompson is
: >certainly correct that A=B*C is a faster matrix multiple than the
: >equivalent nested loops, but CALL GEMM(A,B,C) is about three times
: >faster than that even on a single-CPU box. On the multi-processors,
: >we're unbeatable.
: >
: >scisoft@well.sf.ca.us for more information, end of plug.

: I would be interested in seeing your timings. To date we haven't
: seen any of the "matrix languages", (i.e. IDL, MATLAB, Gauss, etc.) that
: do matrix multiplication faster than O-Matrix.

What is O-Matrix? Is it similar to IDL and PV-Wave?

- Mike

Subject: Re: IDL as programming language?
Posted by scisoft on Sat, 10 Sep 1994 03:42:28 GMT
View Forum Message <> Reply to Message

Beau Paisley (harmonic@world.std.com) wrote:
: In article <34fhs7$ak8@nkosi.well.com>,
: Mike Boucher <scisoft@well.sf.ca.us> wrote:
: >
: >Another alternative is the IDL and PV-Wave interface to the optimized and
: >parallelized math library that our company sells.
: I would be interested in seeing your timings. To date we haven't
: seen any of the "matrix languages", (i.e. IDL, MATLAB, Gauss, etc.) that
: do matrix multiplication faster than O-Matrix.

Our product is primarily a FORTRAN- or C-callable library of math
subroutines, so I cannot claim to have the same flexibility of the
matrix languages such as IDL or O-Matrix. As for speed, we tend
to run BLAS2 in the 15-25 mflop range and BLAS3 in the 25-35 mflop

range on a single CPU, up to 150+ mflops for some of the more
important subroutines like matrix multiply on a four-CPU system.

Subject: Re: IDL as programming language?
Posted by harmonic on Thu, 15 Sep 1994 21:24:48 GMT
View Forum Message <> Reply to Message

> : I would be interested in seeing your timings. To date we haven't
> : seen any of the "matrix languages", (i.e. IDL, MATLAB, Gauss, etc.) that
> : do matrix multiplication faster than O-Matrix.
>
> What is O-Matrix? Is it similar to IDL and PV-Wave?
>
> - Mike

O-Matrix is a reasonably priced,($95) object-oriented analysis and visualization
tool for the PC that uses a simple interpreted language that operates on
matrices instead of individual numbers.

Features:
- Diverse set of built-in functions including:
 Algebraic and trigonometric functions, interpolation, FFTs,
 inverse FFTs and 2D FFTs, QR factorization, Bessel functions,
 singular value decomposition, sorting, random number generation
- Comprehensive graphics including:
 x-y plots with multiple viewports, contour plots, mesh plots,
 polar plotting, histograms, bar plots, stair plots, error plots,
 mouse manipulation of plotted data, and publication quality output
- Extensive library of functions including:
 wavelets ,spectral analysis, Kalman filter, normal distributions,
 circuit simulator, circuit optimizer, signal processing,
 splines, optimization, two-dimensional integration, convolution,
 auto-regressive correlation, special functions, Cholesky factorization,
 spectral estimation, differential equation solvers,
 F-test, t-test, means, medians, eigenvalue solver,
 matrix exponentiation, discrete prolate spheroidal sequences,
 GPS tracking functions ...
- Built in debugger with extensive error messages
- Extensive on-line help, and free technical support by phone, fax, and e-mail
- 32-bit implementation allows matrices and programs up to 32 MB.
- An optimized environment for execution throughtput faster than other
 "matrix" languages.
- User-definable functions with variable numbers of arguments and recursion
- Six matrix types including character, integer, real, double-precision,
 logical, and complex double-precision

If you have further questions, please contact us at:

Harmonic Software
(206) 367-8742
fax (206) 367-1067
harmonic@world.std.com