Subject: Ohmygod, another round of column/row-major...
Posted by steinhh on Wed, 03 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Sorry about bringing this up once more, but the authors of the IDL 5.1/5.2 help pages managed to twist my mind so thoroughly, I had to come here again for support...

And yes, I know this subject *is* covered in the FAQ, under the heading

I'm confused by the meaning of column-major and row-major,

but a lot of the FAQ text uses the concepts colum-ORDER and row-ORDER. These concepts seem to be opposites, and if they're *not*, then I'm completely lost.... If they are opposites, this should be spelled out in a prominent place in the FAQ entry....(but let's first see if we all agree here!).

So, after reading the FAQ + online help, I felt like asking for another FAQ entry called

But I'm *still* confused about column-major and row-major!

In roaming around to resolve my confusion, I tracked down one of William Clodius' postings (<<3637C46B.167E@lanl.gov>) in the original discussion that led to the FAQ entry, and got even more confused when I read that he AGREED with the first paragraph of the online help text on "Arrays and Matrices"...

He was talking about the online help for version 5.1, where Fortran is referred to as a "column-major language". The current text (version 5.2) says the *exact* opposite, namely that C/Pascal etc are column-major!!

This took me quite a while to figure out. No wonder *we're* confused, when RSI from one IDL version to another decides that all the other languages in the world have suddenly changed the way they index their arrays!! And RSI's current conclusion disagrees with just about everybody else..

How can we trust any help page using the concepts row-major vs column-major IDL, when RSI appears to disagree with the rest of the world on whether IDL as a language is one or the other?

Now, this is my current understanding of this issue:

All 3 contributors to the current FAQ entry have got this

right: You cannot [meaningfully] determine whether a language is row-major or column-major unless you assume a convention for referring to (indexing) a matrix element by it's row and column numbers.

There are two possibilities for the indexing of matrices:

matrix(row,column) or matrix(column,row)

According to reliable sources the overwhelmingly dominant way of specifying matrix elements in *mathematics* is, in LaTeX notation, a_{row,column}. The first index is the row number, and the last index is the column.

I.e., you'd write a 3 by 4 matrix like this:

a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}

This indexing convention (row,column) is so common in mathematics that it forms the basis of the traditional classification of computer *languages* as either column-major or row-major, given their array storage/indexing rules.

The corollary that "everybody else" agrees on:

If the first index runs faster when stepping through the elements of an array, it's a "column-major language".

If the last index runs faster when stepping through the elements of an array, it's a "row-major language".

So, if anybody insists on classifying IDL as a language, it's a column-major language. (And I would really like to know if everybody agrees that column-major == row-order ???)

Then, somebody started to write IDL's online "help"....sigh!

Given a computer language with multi-dimensional arrays, it's up to anyone to write a package of matrix routines using either one of the two possible indexing conventions.

It seems like RSI is trying to persuade everybody to switch to the (column,row) indexing convention - most likely because this is easier to map onto the [row][column] convention used in the numerical recipes library (in C). And the "help" page for Numerical Recipes Functions puts this in writing:

> In IDL versions up to and including IDL version 3.6,
> mathematics functions based on Numerical Recipes algorithms
> required that input be in column-major format. This is no
> longer the case. Routines based on Numerical Recipes
> algorithms have been reworked and renamed, so that all IDL
> functions now expect input arrays to be in row-major format
> (composed of row vectors).
[...]
> We recommend that all new IDL programs take advantage of the
> new names and input convention.

That's fine. I have no quarrel with this text, but my advise is to include a sentence "Row-major format in IDL means that matrices are indexed as matrix(column,row)".

Given how IDL prints out arrays and displays images, the new convention has the beneficial side effect of aligning the "image notation" and "matrix notation" for IDL, in that the first dimension will always be horizontal, and the second dimension will always be vertical - whether you're *thinking* about matrixes, *printing* matrices/arrays or *displaying* images. I.e. you may think of indexing two-dimensional arrays as "matrix(x,y)" or "image(x,y)", "array(x,y)" etc.

Please, RSI, leave it at that. It's OK to opt for the (column,row) indexing notation for matrices, to recommend it to everybody, and to supply matrix routines that rely on that convention.

But don't try to "reclassify" IDL as a "row-major language"!! I guarantee that there will be no end to the confusion this will cause.....

The reason is that whenever somebody sees a phrase like "IDL indexes data in row-major format", most seasoned programmers will nod and say, "OK, so it's like C", and they've got the whole thing wrong...

It's just plain *wrong* to say that "IDL is indexing data in row-major format" like it's done in the online help in version 5.2. It's even worse trying to say that C and Pascal is using column-major format!!

It *is*, however, correct to say (that is, I *think* it is correct to say) all of the following:

Most matrix routines in IDL assume that the matrix is

stored in row-major format.

Row-major format corresponds to (column,row) indexing in IDL.

Row-major format corresponds to [row][column] indexing in C.

(column,row) indexing is not the traditional way of indexing matrix arrays.

(column,row) indexing of matrices in IDL means that they will be printed correctly, without transposing them.

(column,row) indexing of matrices means that you can think of the first dimension as "X" and the second dimension as "Y", like for images.

And a very handy mnemonic rule:

A ZZZ-major matrix means that the ELEMENTS of a ZZZ are stored contiguously in memory.

Note that this rule doesn't say *anything* about the language. It says something about how a matrix is stored in memory.

To come back to whether or not we can trust most online help pages speaking about column-major or row-major stuff, the answer seems to be *yes*.

The reason is that most help pages on matrix functions (e.g. the Numerical Recipes functions) seem to always stick to the concept "ZZZ-major matrices" (i.e. not messing about with a classification of languages as such). That means they cannot go wrong.

It's left up to the user, however, to remember what this means in terms of how to index their matrices - (column,row) or (row,column), and how to interpret a printout of such a matrix.

The help pages often use the phrase "composed of column vectors" to "explain" the meaning of "column-major format".

To me, the phrase "composed of column vectors" has zero information content, at best.

I mean, using RSI's notation, a column vector is a fltarr(1,N), right? And a row vector is fltarr(N,1), though the last dimension is cannibalized by IDL for your own good :-)

Thus any two-dimensional array or matrix is composed of column
vectors, and at the same time it is in fact also composed of
row vectors!!

Take for example the explanation of the LUSOL routine. To me,
it would be a *lot* easier to understand than the current
version if it said e.g:

> The LUSOL function is used in conjunction with the LUDC
> procedure to solve a set of n linear equations in n unknowns
*> Ax=b. The parameter A is input not as the original matrix, but
> as its LU decomposition, created by the routine LUDC. The
> result is an n-element vector whose type is identical to A.
>
*> LUSOL assumes that the matrix A is indexed as A(column,row)
*> i.e. that A is a row-major matrix.
>
[.....]
> Keywords:
>   COLUMN
*>   Set this keyword if the input matrix A is indexed as
*>   a column-major matrix, i.e. A(row,column).

So, my advice is:

* When speaking of matrices, use the word "matrix", not the
word "array". Strictly speaking, an array isn't column-major or
row-major, its simply an array. It can be *interpreted* as a
column-major matrix or a row-major matrix, however.

* When speaking of matrices that are column-major or row-major,
always mention what this means in terms of indexing it.

* Never attempt to classify/label IDL as "a row-major
language", or as "a language that uses row-major indexing of
arrays"!

* Be very, very careful if you have to speak of IDL as a
column-major language. Make sure to mention that this is
really just another name for how IDL stores and indexes it's
data arrays (first index runs faster), and that it's based on
the (row,column) indexing convention [which you don't
recommend].

There! I feel a lot better now, I think. Unless, of course,
somebody replies that I've got the whole thing backwards.

But wouldn't that just prove my point, that things are still

confusing..?

Regards,

Stein Vidar