Subject: Re: rounding errors
Posted by Randall Skelton on Fri, 27 Apr 2001 13:17:28 GMT
View Forum Message <> Reply to Message

On Fri, 27 Apr 2001, Dominic R. Scales wrote:

> Danke Alex :)
> the data is read in from a file with readu and is an array
> of float variables. I want to perform the following maths
> in double but with the cast to double I already introduce
> 'pretty random digits to the right', as you say.
> I'ld really like to avoid calling something like double(string(a))
> for some large array...

Perhaps I am confused, but if you want the data to be represented as doubles, you should read it directly into a double array ('array=dblarr(1000)' or something similar)

While you can legitimately extend the precision of floating point numbers to those of doubles you must always remember the underlying IEEE definition which states floats only have 6 digits precision while doubles have 15 digits of precision. When you recast a float into a double, you expect decimal digits 6-15 will be noise because bits beyond the float precision can truly be anything. Asking IDL to make a floating point number into double precision with 'zero padding' like you suggest is like asking IDL to know what decimal digits 6-15 were before you cast them as floats. Using strings as an intermediate type does avoid the problem you describe but it also shows a genuine misunderstanding of storage types.

For the record, I had no idea that IDL requires you to explicitly state 'a=2.348339d0' instead of a=double(2.348339).

Randall

PS: If you are still having trouble with this consider a simple C program:

```
#include <stdio.h>

main () {
   float a = 2.38492;  /* original float */
   double b = a;       /* recast */

   double c = 2.38492; /* original double */

   printf("a (float)  = %2.18f\n", a);
   printf("b (recast) = %2.18f\n", b);
```

```
    printf("c (double) = %2.18f\n", c);

    return(0);
}
---

[anova ~]% gcc test.c -o test
[anova ~]% ./test
a (float)  = 2.384919881820678711
b (recast) = 2.384919881820678711
c (double) = 2.384920000000000151
```