

Re: Replicating results

Source: <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2007-08/msg00152.html>

- *From:* Louis Krupp <lkrupp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 05 Aug 2007 23:11:38 -0600
-

kalu wrote:

I have the following piece of fortran code that seems to give me different answers every time I run it. I need to get the same result every time because, although these differences look small, they unravel into substantial differences. This code snippet is part of a gradient calculation used in a large optimization routine.

I have printed the code and results below. Has anybody else run into this problem? I am using the absoft compiler on an intel64 machine. Please let me know if more information is needed.

-Kalu

begin CoDE:

```
-----  
real*8 :: x(n),grad(n),dh(nmax),ee(nmax,nmax),xsav(nmax)
```

```
! ... some more code goes here ...
```

```
do i = 1,n
```

```
do j = 1,n
```

```
x(j) = xsav(j) - ee(i,j)
```

```
end do
```

```
call evalf(x,tempf1)
```

```
do j = 1,n
```

```
x(j) = xsav(j) + ee(i,j)
```

```
end do
```

```
call evalf(x,tempf2)
```

```
! point of interest
```

```
grad(i) = (tempf2-tempf1)/(2.0D+00*dh(i))
```

Re: Replicating results

! output below shows that the arguments to the right of the equal sign are the same every time while the results of this calculation differ

```
print *,tempf2 = ',tempf2,' tempf1 = ',tempf1,' dh('i,') =  
'dh(i),'grad('i,') = ',grad(i)
```

```
end do
```

```
-----  
End CoDE
```

```
Begin Output  
-----
```

```
tempf2 = -11.2891215185643 tempf1 = -11.2891217937382 dh( 4 )  
= 8.680700000000000E-007 grad( 4 ) = 0.158497573828246
```

```
tempf2 = -11.2891215185643 tempf1 = -11.2891217937382 dh( 4 )  
= 8.680700000000000E-007 grad( 4 ) = 0.158497571781917
```

```
-----  
End Output
```

The two values of tempf1 and tempf2 may *look* the same to 14 decimal places, but they may be different beyond that. The same is true of dh(4). The question is why.

If you don't already use "implicit none" in your main program and in all subprograms, use it. It will make sure that if you do something like declare a variable called "x1" and use a variable called "x1" the compiler will tell you that x1 was undeclared. If x1 was undeclared and never initialized and then used in your calculations, it will have a random value and you will get random results.

Make sure that you initialize all of your variables, including all array elements. If you don't, you will get varying results.

(There are systems that initialize everything to zero. They are the exception. Some compilers have switches that may help you trap some uninitialized variables.)

Make sure that evalf declares its arguments the same as they're declared in the main program. You could be changing the single precision half of a double precision variable and leaving the garbage in the other half unchanged.

Louis

.