

Roundoff

Surprisingly, as a matter of practice, the simple conjugate-direction method defined in this book is more reliable than the conjugate-gradient method defined in the formal professional literature. I know this sounds unlikely, but I'll tell you why.

In large problems numerical roundoff can be a problem. Calculations need to be done in higher precision. The conjugate gradient method depends on you to supply an operator whose adjoint is correctly computed. Any roundoff in computing the operator should somehow be matched by the roundoff in the adjoint. This is unrealistic. Thus optimization may diverge while theoretically converging. The conjugate direction method doesn't mind the roundoff. It simply takes longer to converge.

Let us see an example of a situation where roundoff becomes a problem. Suppose we add 100 million ones. You expect the sum to be 100 million. I got a sum of 16.7 million. Why is this? After the sum gets to 16.7 million adding a one to it adds nothing. The extra one disappears in single precision roundoff.

```
real function one(sum); one=1.; return; end
integer i; real sum
do i=1, 100000000
    sum = sum + one(sum)
write (0,*) sum; stop; end
1.6777216E+07
```

The code must be a little more complicated than I had hoped because modern compilers are so clever. When told to add all the values in a vector they know it is wise to add the numbers in groups, and then add the groups. Thus I had to hide the fact I was adding ones by getting them from a subroutine that seems to depend upon the sum (but really doesn't).