Subject: Re: machine precision
Posted by Wout De Nolf on Wed, 20 May 2009 13:08:47 GMT
View Forum Message <> Reply to Message

Ok, so I was reading the Sky Is Falling paper and the Goldberg paper again. I learned some things I thought I'd share (since this is a recurring issue, despite the Sky Is Falling paper).

A floating point number is stored like this:
f(binary) = sign | exponent | mantissa without leading 1
    sign: 1bit
    exponent: 8bits (11bits when double)
    mantissa: 23bits (52bits when double)

The real number it represents can be found like this
f = sign.mantissa.base^(exponent-bias-n_mantissa)
    sign: -1 when sign-bit=1, +1 when sign-bit=0
    base: 2 (ibeta from MACHAR)
    exponent: 8bit number
    bias: 127 (1023 when double)
    n_mantissa: number of mantissa bits (23, 52 when double)

We will rewrite this as
f = sign.mantissa.eps.base^exp
    eps: base^(-n_mantissa)   (eps from MACHAR)
    exp: exponent-bias

For example: f = 470.
    f(binary) = 0 | 10000111 | 11010110000000000000000
    sign = +1
    exp = 135 - 127 = 8
    mantissa = 15400960
    eps = 2.^(-23)
f(stored) = 15400960*2.^(-15)

The difference between a stored floating point number f1 and its closest neighbour f2:
abs(f1-f2) = eps.(mantissa1.base^exp1-mantissa2.base^exp2)
  smallest possible difference when:
  exp1 = exp2 = exp
  mantissa1 = mantissa2 +1
            = eps.base^exp = 1 ulp (unit in last place)

The absolute error made when storing a real number is therefore
abserr = abs(freal-f) <= c ulp
where c=1 for truncation and c=0.5 for rounding

The relative error made is
relerror = abs(freal-f)/abs(freal)
<= c.eps.base^exp/abs(freal)
<= c.eps (not sure about this last step....)

Finally, two numbers are considered equal if
relerr = abs(f1-f2)/(abs(f1)>abs(f2)) <= eps
I'm not really sure about this one either (e.g. what should be in the denominator, what about c,...)

All this doesn't deal with accumulated errors in floating point arithmetic, only with errors introduced by storing a real number.