Computer Numbers and their Precision, I Number Storage

Landau's Rules of Education: Much of the educational philosophy applied in these modules is summarized by these three rules:

1. Most of education is learning what the words mean; the concepts can be quite simple once the understanding matches what is being said.
2. Confusion is the first step to understanding.
3. Traumatic experiences tend to be the most educational ones.

Exercise: Aunt Sally

1) Use your handy calculator, and then computer, to evaluate

i) $6 + 3 \times 2$
ii) $6 + (3 \times 2)$
iii) $(6 + 3) \times 2$
iv) $6 \div 3 + 2$
v) $6 \div (3 + 2)$
vi) $6 + 2 \div 3$

Integer Arithmetic

Exercise: Integer Arithmetic

Now try some of these exercises on your computer to see the effect of Integer arithmetic. Use excel or python.

1. $6 \times 2 = ?$
2. $6 \div 2 = ?$
3. $3 \div 2 = ?$
4. $8 \div 3 = ?$
5. $2 \times (6 \div 2) = ?$
6. $2 \times 6 \div 2 = ?$
7. $2 \times (3 \div 2) = ?$
8. $12 \div 2 \div 3 = ?$
9. $12 \div (2 \div 3) = ?$
10. $(12 \div 2) \div 3 = ?$
11. $12 \div (3 \div 2) = ?$

Exercise: Determining Range of Integers

Write a program in python or use excel to determine experimentally the range of positive and negative integers on your computer system and for your computer language.

A sample pseudocode (the elements of a program not written in a specific computer language) is

```
max = 1
min = -1.
do 1 < i < N
        max = max *2.
        min = min * 2.
        write i, max, min
```

end do

Hint: we have indicated above that the integers in Python lies between $\mp 10^{19}$, which is equivalent to approximately $\mp 2^{64}$.

Floating-Point Numbers

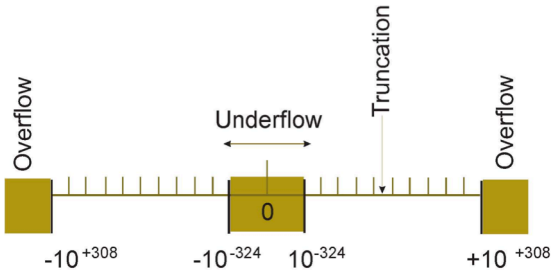

Figure 1 The approximate limits (-10^308, -10^324, 10^324, 10^308; black lines) of double-precision (64 bit) floating-point numbers and the consequences (overflow, underflow) of exceeding these limits. The hash marks abstractly represent the number values that can be stored ; storing a number in between these values leads to round-off error. The shaded areas correspond to over- and underflow.

Exercise: Over and Underflows

1. Write a program and test for the **underflow** and **overflow** limits (within a factor of 2) of your computer system.
   A sample pseudocode is

   under = 1.
   over = 1.
   do 1 < i <N
   under = under/2.
   *over = over * 2.*
   write out: i, under, over
   end do

You may need to increase N if your initial choice does not lead to underflow and overflow.

2. If your computer system lets you specify word length , check where under- and overflow occur for single-precision floating-point numbers (floats). Give your answer in decimal.
3. Check where under- and overflow occur for double-precision floating-point numbers (doubles in Java, floats in Python).

Machine Precision (Model)

Exercise: Determine Your Computer's Machine Precision

We want you to either write a program or just enter numbers on your computer terminal that will permit you to determine the machine precision ϵ_m of your computer system. Getting the answer within a factor of 2 will be just fine.

Start with an arbitrary value of ϵ_m, say $\epsilon_m = 1$, and add that to 1.0. If the answer is not 1.0, then make ϵ_m smaller and again add it to 1.0.

Keep repeating the process until you get 1.0 as an answer. The largest value of ϵ_m for which this happens is your machine precision.

Implementation

Note, if you are doing this by entering the values of ϵ_m by hand, you may want to divide by 10 each time until you get a gross answer, and then do some fine tuning. If you are writing a program, you can divide by 2 and just let the computer churn away for a long time. Here is a sample pseudo code (the basic elements of a computer program not specific to any language):

```
eps = 1.
do N times
        eps = eps/2.                    # Make smaller
        one = 1. + eps
        output one, eps        # Write loop number, one, eps
end do
```

Python Implementation

A model Python program that needs to be extended to actually determine the precision is *Limits.py:*

```
# Limits.py: Increase N to determine approximate machine precision

N = 3
eps = 1.0
for  I  in range(N):
        eps = eps/2
        one = 1.0  +  eps
        print('eps = ', eps,  'one = ', one)
print("Enter and return a character to finish")
s=raw_input()
```

Vensim:

Excel:

Summary and Conclusions

You must know something about how computers store numbers in order to judge how reliable the results are. Unless you are carrying out symbolic manipulations, computed numbers are never exact, but can still be used to carry out calculations within a stated level of precision. Specifically, floating point numbers can give up to 16 places of precision, while integers can be exact, but have a much more limited range of values. Integers are thus appropriate for counting, such done when looping.

Where's Computational Scientific Thinking

- Understanding that computers are finite and therefore have limits.
- Being cognizant that uncertainties in numerical calculations are unavoidable.
- Understanding how it is possible to work within the limits of a computer to obtain meaningful results.
- Understanding the range of numbers that may be necessary to describe a natural phenomenon.

Intuition Development

References

[CP] Landau, R.H., M.J. Paez and C.C. Bordeianu, (2008), *A Survey of Computational Physics,* Chapter 5, Princeton Univ. Press, Princeton.

[UNChem] *UNC-Chapel Hill Chemistry Fundamentals Program, Mathematics Review,* http://www.shodor.org/unchem/math/; *Using Your Calculator,* http://www.shodor.org/unchem/math/calc/.