

1. Binary Calculations: Pitfalls with Floating-Point Numbers



3

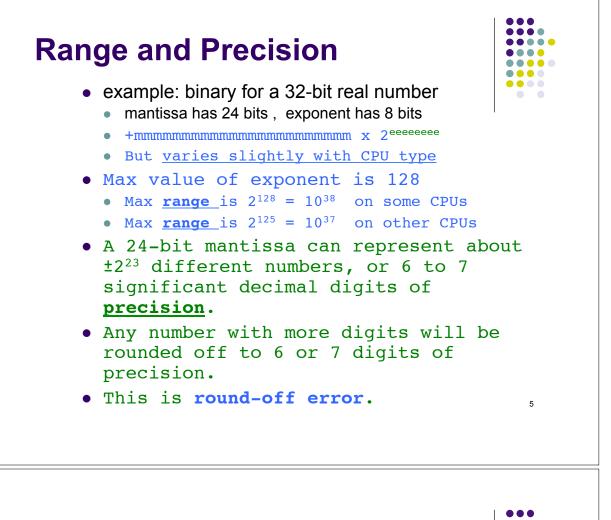
4

(work will be done on the blackboard, based on the readings.)

- representation of floating point numbers in binary – accuracy, precision, & range
- errors: round-off; truncation
- pitfalls: conditional testing with floats; small difference between large floats; be aware of the discrete number line
- tips: add smallest numbers first; reduce number of math operations; use existing math libraries (Num. Recip.)

Binary Representation of Floating-point (real) numbers

- scientific notation: mantissa and exponent
- example: decimal: 8.453 x 10⁴
 - mantissa = 8.453, exponent = 4
 - mantissa = 0.8453, exponent = 5
 - the mantissa has 4 digit precision
- example: binary for a 32-bit real number
 - mantissa has roughly 24 bits (23 bits plus a sign bit)
 - exponent has roughly 8 bits (but varies with CPU type)
 - mmmmmmmmmmmmmmmmmmeeeeeee
 - 11000011100011010010101110111100
 - +10000111000110100101011 x 2¹⁰¹¹¹¹⁰⁰



Pitfalls: Conditional tests using real numbers (floats)

Write a Fortran program to be as follows. Then compile and run.

```
program floater    !test floating point numbers
implicit none    !enforce strong typing
real :: x = 0.0    !initialize a floating-point variable
write(*,*) "Welcome to Floater"
do
    x = x + 0.1    !increment x
    if (x .eq. 3.0) exit    !stop looping when x = 3
    if (x .gt. 10.0) exit    !stop looping when x > 10
    write(*,"(F4.1)") x    !display the value of x
enddo
endprogram floater
```

Pitfalls: Conditional tests using real numbers (floats)



Did the program do what you expected? Why?

Let's run some diagnostic tests to understand why. <u>A useful</u> <u>debugging trick is to add more write statements to get more</u> <u>info about the values of variables as the program runs.</u>

So AFTER the existing write statement for x, add the following write statement that is not formatted. Then save, recompile and run.

write(*,*) x !display the value of x

- 1) Based on these diagnostics, what was the problem?
- 2) Can you suggest ways to fix it?

Increasing the Precision of Real numbers.



8

7

Perhaps if the binary number were more precise, it might successfully execute the program.

In Fortran, a real number with extra precision is called a **"Double Precision"** (dp) number. If often uses 64 bits (8 bytes) to hold the floating point number, which has a **precision** of about 14 or 15 decimal digits, and a **range** of about $\pm 10^{307}$ or $\pm 10^{308}$.

But it varies from CPU to CPU.

Increasing the Precision of Real numbers.

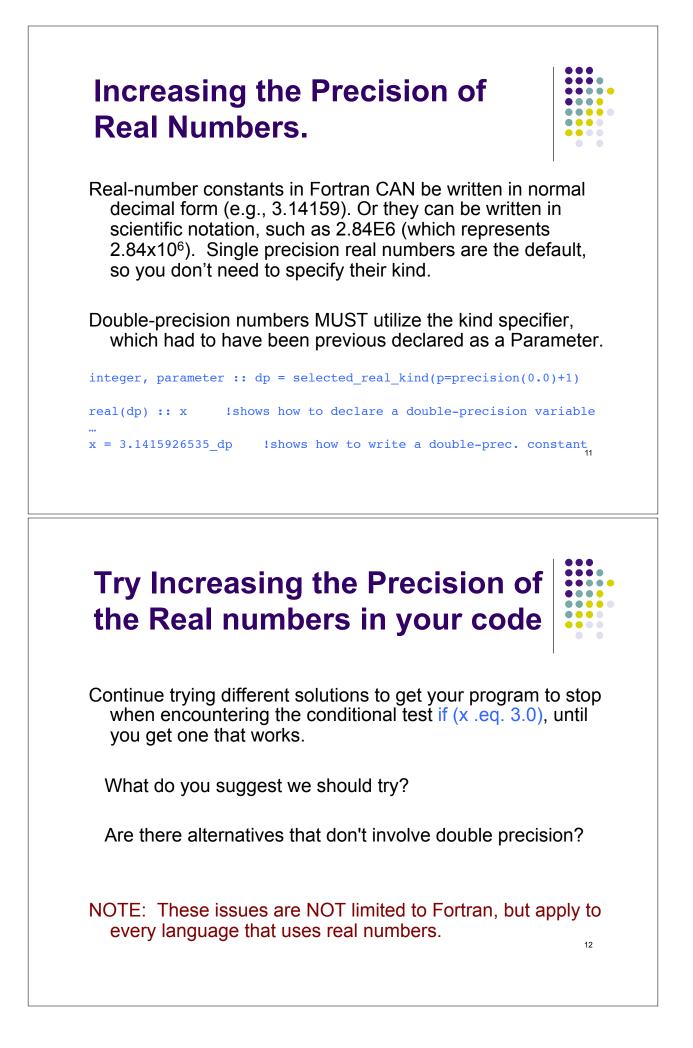


q

The "kind number" is an integer code that specifies the precision of real numbers. Often ,but not always, it is a count of the number of 8-bit bytes used to represent the number. On some machines, kind number 4 indicates a normal, single-precision number. Kind number 8 indicates double precision. But depends on the CPU.

Try it. Modify your code:

...then compile and run.



The Number Line



See the paper by Hayes on "The higher arithmetic."

How are decimal numbers distributed along the number line when represented by binary: (a) integers, (b) floats (discuss at blackboard)

Numbers too small to be represented cause **underflow**, and are usually truncated to zero.

Numbers too large cause **overflow**, and cause the program to STOP with an error message (if you are lucky).

Math operations that cause Bit shifting reduces precision



To add or subtract real numbers, the exponents of the two numbers must match.

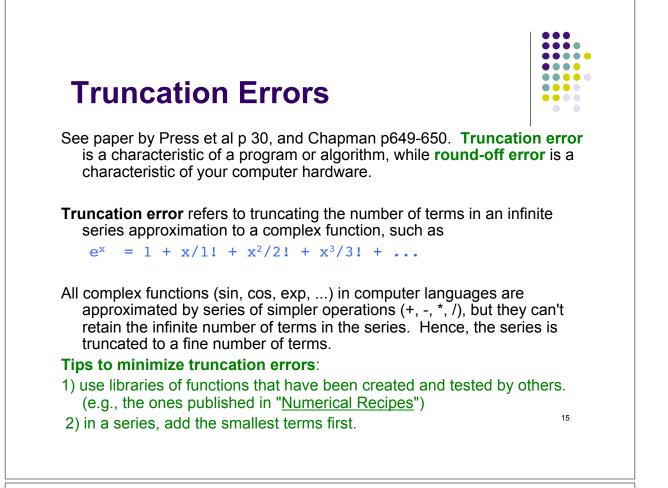
For example in decimal, to add 2.46x10⁵ + 3.15x10³. For the smaller number, we must first shift the digits in the mantissa 2 places to the right and increase the exponent by 2 in order to make the exponents match before doing the addition. Assume the mantissa has precision of 3 digits.

2.46x10 ⁵						2.46x10 ⁵
$+ 3.15 \times 10^3$	is t	che	same	as	+	0.03x10 ⁵
					=	2.49x10 ⁵

But we lost precision in the small number before adding. Similar bit shifting and exponent incrementing is done for binary additions.

Tips to minimize round-off errors:

- 1) add small numbers together first before adding larger ones.
- 2) reduce the number of arithmetic operations if possible.
- 3) avoid small differences between 2 large numbers.



2. Synthesis of Concepts

- For each programming language we studied, for which situations is it the best language to use? (i/e; advantages of one vs. another)
- For which situations would unix/linux be a better operating system to use than Windows or MacOSX?
- What are the programming tools my grad students use?
- What additional programming topics would you like to see taught in this course in future years?

Utility of what you learned: $\bullet \bullet \bullet \bullet$ Computer Skills used by ATSC co-op Students: As undergrad D.B. M.W. co-op student unix/linux unix/linux SQL (MS-SQL) Fortran Fortran windows windows EC visual basic visual basic excel excel bash scripting TecPlot (graphics) unix/linux unix/linux html html SQL Fortran С UBC Perl bash scripting php php python Vis5D (graphics) 17

3. Final Exam Preparation Date, Time, Place. Open book. Open computer. About 10 questions.

Readings (revisited):

- Course Pack section on Computer-language Evolution:
 - Hayes, B., 2006: The semicolon wars. American Scientist, 94, 299-303. •
 - Levenz, E. 2009: History of programming languages. (a timeline chart)
- Course Pack section on Good Programming Practices:
 - Section from: B.W. Kernighan and R. Pike, 1999: The Practice of Programming. Addison Wesley)
 - Epilogue
 - Appendix Collected Rules
 - Chapter 5 Debugging (focus on main topics and methods, not on any one programming language)
 - Section from: S.J. Chapman, 2007: Fortran 95/2003 for Scientists & Engineers, 3rd Ed. McGraw Hill. • Only pages 82-89, which cover top-down design and flowcharting.
- Course Pack on Binary Calculations and Pitfalls:
 - Press, Teukolsky, Vetterline, Flannery, 2007: Numerical Recipes in C, 2nd Ed. only pages 28 31 in Section 1.3 Error, Accuracy, and Stability, which covers binary numbers, numerical accuracy, and numerical errors.
 - S.J. Chapman, 1998: Fortran 90/95 for Scientists & Engineers, 1st Ed. McGraw Hill. pages 4-11 and p646-660, which covers round-off errors, truncation errors, and other in numerical-calculation errors.
 - B. Hayes, 2009: The higher arithmetic. American Scientist, 97, 364-368.

