Good numerical precision case study example

ASSIGN BUSTER

Floating point is a numerical-interpretation system in which strings of digits or bits represent real numbers. A system of arithmetic is defined to permit these representations to be manipulated with resultant being similar to the arithmetic operations done over real numbers. The representation uses an explicit designation of where the radix point should be located relative to the string. In computing, fixed-point number representation is a real data type representing a number with a fixed number of digits before and after the radix. The most common way of representing floating-point numbers in computers is the IEEE 754 standards or IEEE floating point. Several different representation of real numbers have been proposed but the most widely used method is floating point representation. Floating point representation is generally represented as ± d0 . d1 d2 dp-1 × e where, assumed to be even is the base and p is the precision, d is called the significand.

Floating point numbers are stored either as single precision or double precision. Single precision requires 32 bits while double precision requires 64 bits.

They are stored in the format m x b^e. For single precision floating point there are 1 sign bit, 8 exponent bits (biased by 127) and 23 mantisa. For double precision floating points, there are 1 sign bit, 11 exponent bits (biased by 1023) and 52 mantisa.

The floating point representation of a nonzero number is unique as long as the condition 1 < m < 2 must be satisfied.

Floating point representations are necessarily unique due to the requirement of normalization. When representing a number in floating –point format, base and precision is determined. Additionally, the largest and smallest allowable exponent’s emax and emin are computed. Floating point numbers do not

represent numbers exactly; rather it represents exponents multiplied by an arithmetic series. Thus manipulating numbers that fall between the gaps between any two floating numbers presents difficulties.

Floating point numbers are memory efficient. This is because an approximate of the floating-point number is stored in memory thereby providing space to store a range of numbers in a small memory.

Binary-coded decimal sometimes referred to as natural binary-coded decimal is a common modern implementation, packed decimal, encoding for decimal numbers in which each digit is represented by its own binary sequence. Its main characteristic is that it allows easy conversion to decimal digits for printing or display functionalities. A digit is normally represented by four bits in BCD representing the decimal digits 0 to 9. Other bit combinations are sometimes used for a sign or for other indications.

Comparison

Precision

Floating point math is not exact, and the resultant values have a small approximation error. The error is as a result of floating point values covering a huge range, so the internal representation of the value can hold an approximation. Thus, it is essential to test if the values lie within a specific range of tolerance.

The single precision floating point format has approximately 6-1/2 digits of precision. The dynamic range is given by 2±128 or about 10±38 and is the limiting factor for many scientific applications. The limited precision for single precision format may introduce serious errors.

Double precision floating point format overcomes the problem of single

precision with a dynamic range of 10±308and 14-1/2 digits of precision, sufficient for many applications.

BCD representation is more precise than floating point format. Programmers use BCD to counter the effects of floating point format.

Floating point method is memory efficient because it stores a huge range of values by storing an approximation of the number

BDC are memory inefficient than Floating point numbers. In fact, uncompressed BCD is a relatively inefficient encoding that occupies more space than a purely binary representation. For instance, an eight-bit BCD variable can represent values in the range 099 while that eight-bit value when holding a binary value can represent values in the range 0.. 255. The same can be said of 16-bit binary which can represent value in the range 0.. 65535 but can only represent a sixth of those values 0.. 9999.

In terms of performance, BCD can be easily converted into human readable representations or between internal numeric representations and their string representations. Also, in BCD, it is easy to encode multi-digit decimal values in hardware than when using binary. BCDs are commonly used in embedded systems such as alarm clocks and toaster ovens.

Floating point formats talks as much as 2. 5 times to be executed in a PC than it does to execute same integer instruction in BCD. This is attributed to the processes of equalizing the exponents, adding the mantissas and normalizing the result.

In order to attain processing efficiency, BCD is preferred. This is because BCD is easy to execute in computer applications compared to floating-point and exponent formats. Thus, BCD is utilized in applications where processing efficiency is preferred.

Decimal data types are used in banking industries because binary floating point numbers introduce errors. There are regulations that mandate conversion of financial transactions such as Euro and national currencies. Though there have been no previously suggested standard for rounding in banking industry, regulations such as Council Regulation (EC) No 1103/97 of European Commission detailing the rules of rounding have been developed. Rounding error can be handled by using such modes as Round-Half-To-Even or bakers rounding to avoid systematic bias during calculations which may cost banks or clients. According to ISO, the most stable mode for rounding in financial industry to preserve precision loss is to round towards the nearest even value to average out the rounding error and remain with the most consistent value.

Difference between Exponential format and floating format

Format

Exponential format are used to represent numbers much bigger or smaller that are essentially difficult to read or write due to the mistake of counting zeros. The format used is a combination of fixed point number and whole numbers. A fixed point number is written followed by an E, followed by a whole number. The whole number indicates how many places to the right the decimal point of the fixed point number would have to be moved to obtain the dame value as an ordinary number. Their format is represented as N E M e. g. 4. 5E-3.

The format of an exponent number requires the decimal point, sign of the entire number, sign of the exponent, magnitude of the exponent and the letter E.

Floating point numbers are represented in the format d0 . d1 d2 dp-1 × e where, assumed to be even is the base and p is the precision, d is called the significand. The sign of an exponent in floating-point format is represented using sign/magnitude and twos complement. The number of bits in floating point differs according to single, double and extended precision.

Processing speed

Exponent format is used to represent a number which encompasses a wide range of values. If the likely results of an operation lie in a very large range, then the most significant part is given. As compared with floating-point, the processing speed of exponent forward is lower. This is because exponent methods encompass a wide range of values of a number.

Accuracy

The accuracy attained while using exponent method is better than while using floating-point format. In exponent method, the number after the decimal point indicates the number of positions to be written after the decimal point. All exponent formats are written so that the number is between 0. 1 and 0. 9999 implying that the value given is exact and not an approximate. Floating-point format output approximate values with a deviation from the true value.

References

Govindarajalu, B. (2010). Comp Arch And Org, 2E. Tata McGraw-Hill Education.

Ian Chivers, J. S. (2012). Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003, 2008 and 77. Springer.

Maini, A. K. (2007). Digital Electronics: Principles, Devices and Applications. John Wiley & Sons.

Overton, M. L. (2001). Numerical Computing with IEEE Floating Point Arithmetic: Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises. Society for Industrial and Applied Mathematics.