Everything you never wanted to know about IBM and IEEE floating point numbers

Kevin W. Hall*
kwhall@ucalgary.ca

ABSTRACT

The CREWES Matlab® toolbox SEG-Y I/O functions have long been able to read IBM floating-point, but have not been able to differentiate between IBM and IEEE trace data. This was left to the user. In addition, it was not possible to write IBM floating point SEG-Y files using the CREWES tools, which may be desirable if a user needs to import data into legacy software. Functions *num2ibm()* and *ibm2num()* have been written, tested, and are now available in the toolbox for writing IBM floats.

Four-byte IBM floating-point can store both smaller and larger numbers than Four-byte IEEE floating-point. Figure 1 shows the results of reading sinusoids of various amplitudes stored on disk as IBM into Matlab doubles (eight-byte IEEE) using *ibm2num().* Figures 2-4 show the results of reading the same sinusoids into three commercial seismic processing packages. If numbers larger or smaller that can be stored in four-byte IEEE must be stored, I recommend using eight-byte IEEE and SEG-Y revision 2.

SEG-Y revision 0 allowed trace data to be stored as four-byte IBM floating point with the data format code in the binary file header set to 1. As time went on people began to write SEG-Y files using four-byte IEEE floating-point, but continued to use format code 1. This lead to ambiguity when it came time to read these files. Figures 5-8 show the (normalized) consequences of choosing the wrong floating-point format when reading seismic data.

Figure 9 shows a 3C Vibroseis source gather that has been stored using IEEE floating-point, but read into Matlab assuming the data is IBM floating-point. Figure 5 shows a single trace from this gather. Visually, it is difficult to tell that something is wrong at this scale. Converting this result to IBM and back to IEEE and subtracting gives the gather shown at the top of Figure 10. The trace-by-trace sum of the differences is shown at the bottom. None of these numbers are exactly zero. If the data on disk was actually IBM floating-point and is correctly read as IBM, this difference plot would be all zeros (not shown). *readsegy()* in the CREWES Matlab toolbox now uses this observation to guess if data on disk are IBM or IEEE floating point.

Why does this happen? While the IEEE fraction is always stored using 24 bit precision, an IBM fraction can be stored using anywhere from 21 to 24 bits. Figure 11 shows the precision with which the IBM fraction is stored for the amplitudes shown in Figure 9. Figure 12 shows that, in this case, roughly one-quarter of the data is stored with each precision. As expected, IBM numbers stored using the fewest bist for the fraction show the greatest variances from the IEEE numbers before conversion.

1) Large and Small Numbers

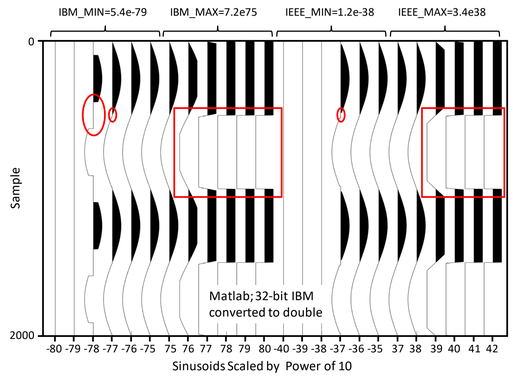

FIG. 1. Red ovals highlight artifacts at values between 0.0 and IBM minimum, and red rectangles highlight artifacts at values greater than IBM maximum. This plot shows that *ibm2num()* correctly converts 4-byte IBM amplitudes to 8-byte IEEE.

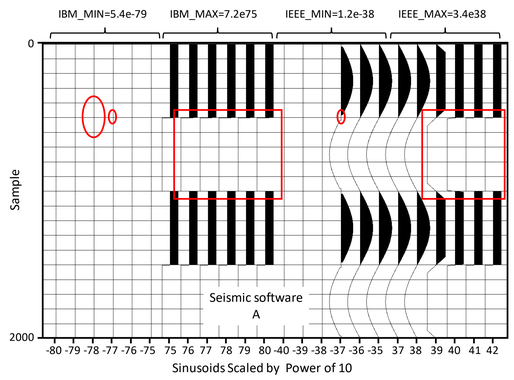

FIG. 2. This plot shows that Seismic software A correctly converts 4-byte IBM to 4-byte IEEE floating point.

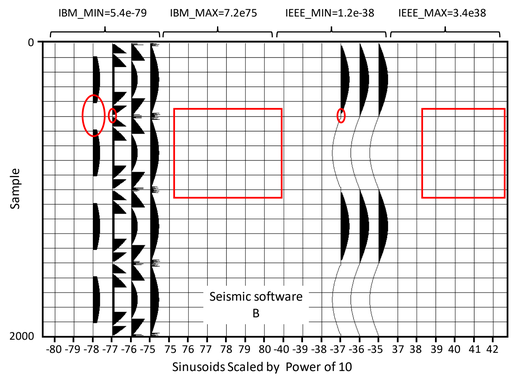

FIG. 3. This plot shows that Seismic software B does not correctly convert large or small 4-byte IBM amplitudes to 4-byte IEEE floating point.

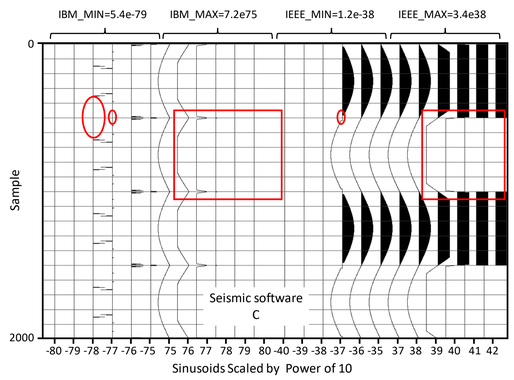

FIG. 4. This plot shows that Seismic software c does not correctly convert large or small 4-byte IBM amplitudes to 4-byte IEEE floating point.

2) Data Read Using Incorrect Floating-Point Format

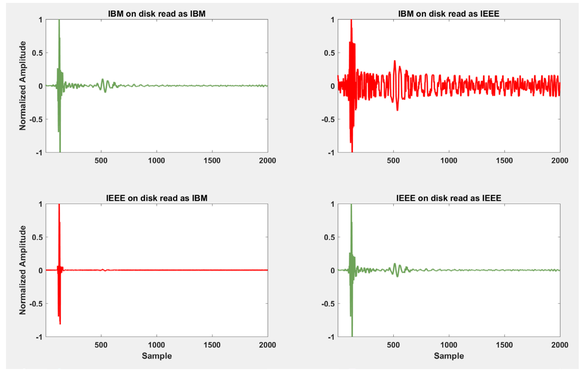

FIG. 5. Correlated Vibroseis data acquired with a 10-250 Hz sweep stored as IBM (top) and IEEE (bottom) floating point, that has been read into memory as IBM (left) and IEEE (right). Trace amplitudes have been normalized for comparison, but no other processing has been applied. Correct answers are shown in green.

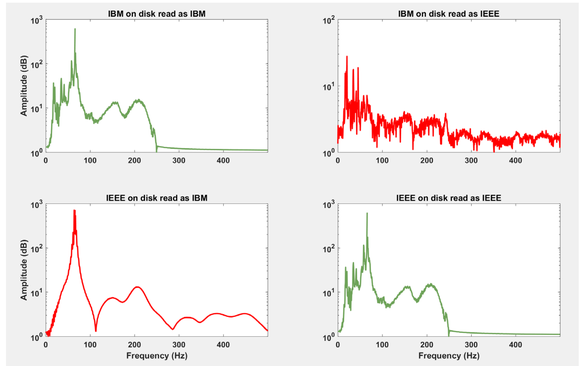

FIG. 6. Amplitude spectra corresponding to the traces shown in FIG. 5.

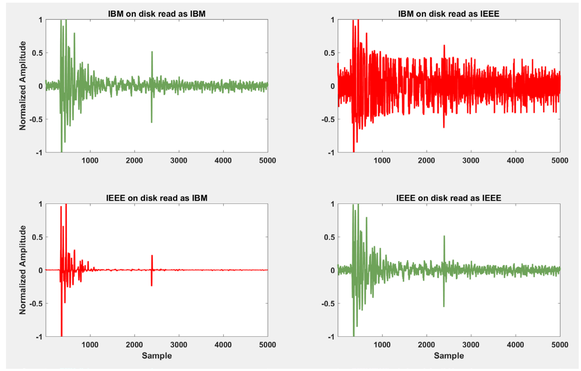

FIG. 7. Dynamite data stored as IBM (top) and IEEE (bottom) floating point, that has been read into memory as IBM (left) and IEEE (right). Trace amplitudes have been normalized for comparison, but no other processing has been applied. Correct answers are shown in green.

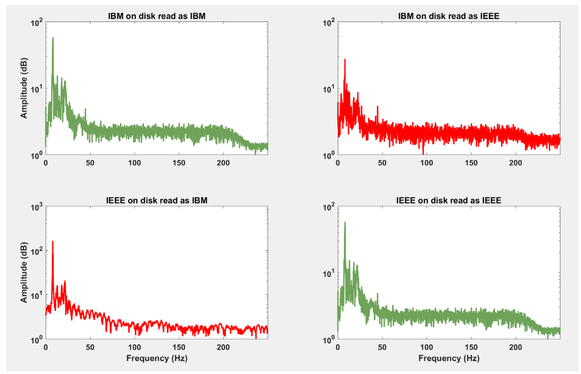

FIG. 8. Amplitude spectra corresponding to the traces shown in Figure 7.

3) Is SEG-Y Data IBM or IEEE Floating-Point?

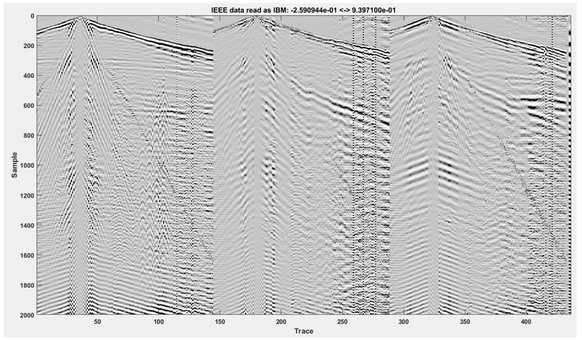

FIG. 9. Real data stored as little-endian IEEE floating point and read incorrectly into Matlab as IBM floating point with *readsegy()* and displayed using *plotimage()*.

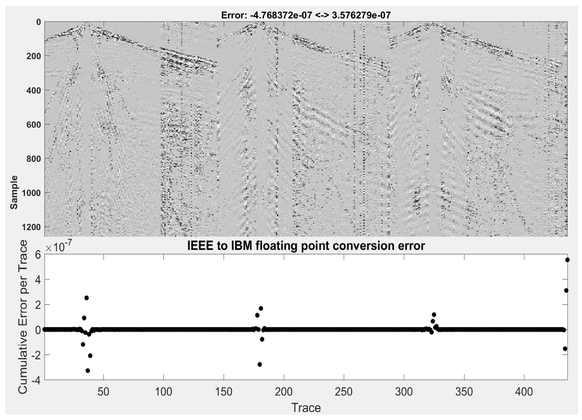

FIG. 10. Data shown in Figure 11 after conversion to IBM and back, subtracting from the data shown in Figure 9 and displayed using *plotimage()*. The cumulative error (sum of amplitude differences) per trace is displayed across the bottom.

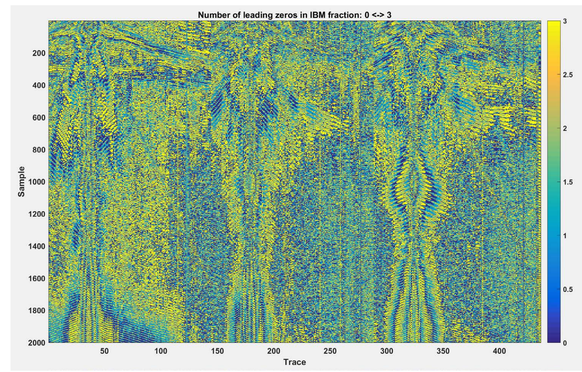

FIG. 11. Number of leading zeros in the most significant 4-bits (nibble) of the IBM floating-point fraction. Zero leading zeros means the fraction is stored with 24-bit precision. Three leading zeros means the fraction is stored with 21-bit precision.

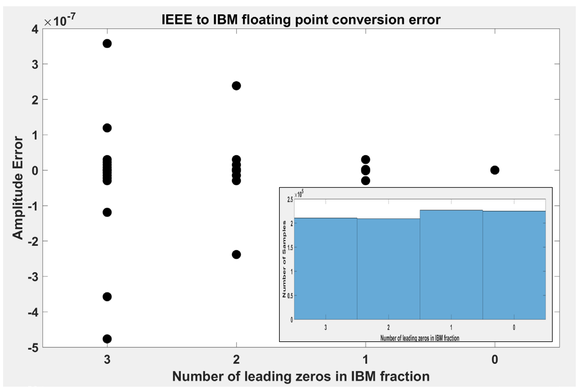

FIG. 12. Sum of amplitude differences plotted against the number of leading zeros in the most significant 4-bits (nibble) of the IBM floating-point fraction.