Wavelets Applied to Lossless Compression and Progressive Transmission of Floating Point Data in 3-D Curvilinear Grids

Aaron Trott, Robert Moorhead, John McGinley
NSF Engineering Research Center for CFS
Mississippi State University *

Abstract

A method of lossless compression using wavelets is presented that enables progressive transmission of Computational Fluid Dynamics (CFD) data in PLOT3D format. The floating point data is first converted to double-precision floating point format to maintain adequate precision throughout the transform process. It is then transformed using Haar wavelets — four times in two spatial dimensions, twice in the third spatial dimension, and twice in time for a total compression factor of 64 times. The double precision format will maintain enough precision during the transform to keep the process lossless. Next, the transformed data is compressed using Huffman coding and transmitted progressively using spectral selection. This allows most of the information to be transmitted in the first pass. Details are transmitted in later passes which ultimately provide for lossless reconstruction of the original data.

1 Problem and Underlying Principles

1.1 Introduction

Many times in Computational Fluid Dynamics (CFD) work very large datasets are produced on remote machines. This vast amount of data must often be moved to a local machine for post processing and visualization. However, this can take large amounts of time because of the large quantity of data that must be transmitted. Compressing the data can speed up the transmission and save several hours of the researchers' time. Progressive transmission can further increase efficiency of the visualization process by giving researchers an approximation of the data very quickly. They can then make a decision based on this approximation about whether to continue the transmission or, if the data is determined to be undesirable, to abort it.

1.2 Haar Wavelets

This application currently uses Haar Wavelets. These are very simple functions with the scaling and detail filters defined in Eq. (1) and Eq. (2) respectively.

$$\Phi_{Haar}(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\Psi_{Haar}(t) = \begin{cases} 1 & \text{if } 0 \le t < 1/2 \\ -1 & \text{if } 1/2 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

*Engineering Research Center, P.O. Box 9627, Mississippi State University, Mississippi State, MS, 39762. Email: {atrott,rjm,jam}@erc.msstate.edu

0-7803-3707-7/96 ..$4.00 © 1996 IEEE

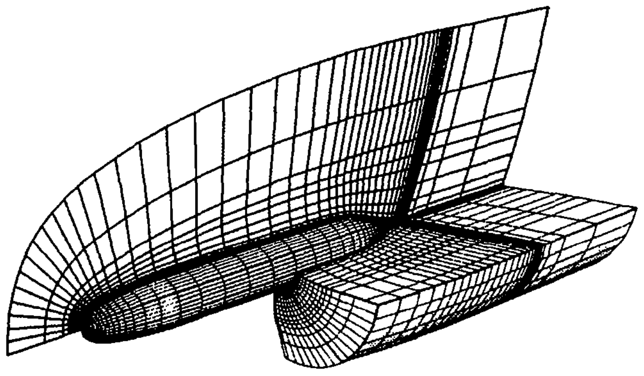

Figure 1: Example of a Multi-Zone Grid

```
Number of zones in file (if multi-zone format)
(zone 1)  i_dim  j_dim  k_dim
(zone 2)  i_dim  j_dim  k_dim
    :
(zone n)  i_dim  j_dim  k_dim

(zone 1)  all x values
          all y values
          all z values
(zone 2)  all x values
          all y values
          all z values
    :
(zone n)  all x values
          all y values
          all z values
```

Figure 2: Format for Plot-3D Grid Files

1.3 The PLOT3D File Format

The PLOT3D file format has become a standard for dealing with curvilinear grids. Fig. 1 shows an example of a multi-zone curvilinear grid for the hull of a submarine. The hemisphere displayed in the figure is divided into four zones running the length of the hull and radiating outward as indicated by the extended sections of the grid. The three PLOT3D file types are grid, solution, and function files. All of these file types support multi-zone grids [2]. Grid files contain the location of each point in a grid as shown in Fig. 2. Solution files contain information global to each zone as well as the value of five parameters at each grid point. The global values are the free stream mach number, the angle of attack, the Reynolds number, and the time (Fig. 3). The five parameters are density, three components of momentum, and energy. Function files can be user defined and are not dealt with in this discussion.

```
Number of zones in file (if multi-zone format)
(zone 1)  i_dim  j_dim  k_dim
(zone 2)  i_dim  j_dim  k_dim
    ⋮
(zone n)  i_dim  j_dim  k_dim

(zone 1)  fsmach
          alpha
          re
          time
          all density values
          all momentum (u) values
          all momentum (v) values
          all momemtum (w) values
    ⋮     all energy values
(zone n)  fsmach
          alpha
          re
          time
          all density values
          all momentum (u) values
          all momentum (v) values
          all momemtum (w) values
          all energy values
```

Figure 3: Format for Plot-3D Solution Files

Solution files are more complex than grid files, but they can contain redundancy that aids in compression. In the data sets discussed here the global attributes for all of the zones are the same. This allows the elimination of all but the first set of global attributes. Also, the density of the fluid (water) remains a constant of 1.0 for all of the current work. Since the density values make up 20 percent of each solution file, it gives a significant boost to compression. However, there are four other floating point values at each point, so there is much more data to compress.

The data explored in this paper are contained in multiple zones. Each zone can contain either part of the structure being simulated or a time step in the simulation. Since we use the zones for separate time steps our compression tool only needs to deal with one file at a time. This greatly simplifies the I/O procedure. However, if the zones are used for different parts of the structure that have equal dimensions, the zones can be compressed pairwise spatially instead of in time.

1.4 Floating Point Numbers and Lossless Compression

Floating point numbers are more difficult to compress than integers because their structure is more complicated. For example, single precision floating point numbers are composed of a sign bit, an eight bit biased exponent, and a 23 bit mantissa (Fig. 4). Double precision floating point numbers have a sign bit, an 11 bit biased exponent, and a 52 bit mantissa. The different parts of these numbers are not on byte boundaries in memory, so breaking them into pieces requires bit manipulations.

Another problem in lossless compression is that the number of significant digits increases as a result of the transform process [1]. When numbers are added, extra bits are needed to keep track of the significant digits. Let P_{max} and P_{min} be the maximum and minimum exponents in a series of floating point additions. Then the total number of extra bits needed is calculated in Eq. (3). Note that this is the worst case.

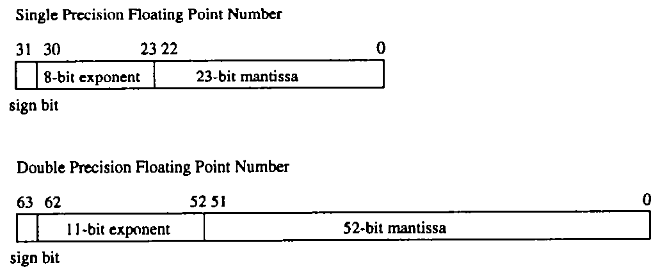

Figure 4: Formats for Floating Point Numbers

$$digits = P_{max} - P_{min} + n \tag{3}$$

where:
$\min(2^n) \geq$ number of coefficients added

For example, the addition of five numbers requires $P_{max} - P_{min} + 3$ extra bits of precision because 2^3 is the smallest power of two that is greater than or equal to five.

A mechanism must be developed to retain these extra bits. One way to do this is to convert the single precision floating point data to double precision. As long as the additions in the wavelet transform do not increase the precision by more than $52-23 = 29$ bits, there will not be any lost information. This should not be a problem in this implementation because there will only be 64 numbers added per data point. These additions will require only $n = 6$ extra bits of precision as calculated in Eq. (3). The remaining 23 bits gained from the conversion to double precision can be used for shifting the mantissa to equalize exponents. An appropriate data set will not require more than 23 extra mantissa bits in each 4x4x2 spatial footprint over two time steps so this is a reasonable constraint.

After the data is transformed, it can be encoded in a variety of ways to ensure that the correct number of significant digits is retained as well as provide for maximum compression.

2 Solution

By taking advantage of the principles discussed above as well as the characteristics of the data we are using, we can implement a reasonable solution that meets both of our goals—lossless compression and progressive transmission.

After reading the data from a PLOT3D file into memory, it can be transformed block by block. The data is first transformed in two spatial dimensions dividing the data into slices. The block size for this set of transformations is four by four. The results of these transforms are placed into a double precision floating point array to maintain the correct number of significant digits as discussed previously. The data is next transformed in the third spatial dimension using pairs of slices. Finally, it is transformed in time (Fig. 5). This gives a combined compression factor of 64 to 1 if only the scaling terms are retained.

After transforming the data, it is Huffman encoded based on a statistics file that is generated beforehand. The statistics file can be specific to a particular data file or general for a class of files. There is a variety of ways in which to encode the transformed data. Utilizing spectral selection [3], bands of coefficients in each block can be coded separately.

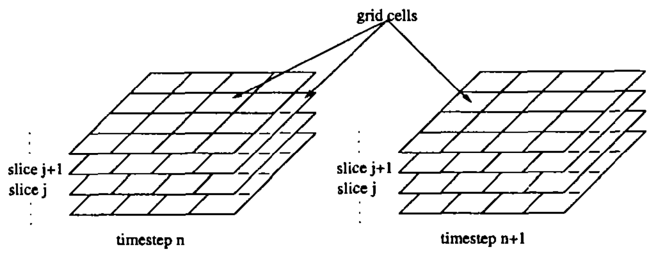

Figure 5: Structure of 4-Dimensional Data

File Type	Max Comp	Min Comp
grid	14.94%	13.22%
solution	6.75%	4.62%

Table 1: Compression Ratios

For example, all of the scaling coefficients can be encoded first then the coefficients corresponding to finer resolutions (detail) follow.

On the receiving end the encoded data is decoded, inverse transformed, and reassembled into its original form. Basically, the process described above is reversed and repeated once for each spectrum of data that is received.

Currently, a very simple Huffman coding scheme is used to perform the actual compression. The value of each byte of the transformed data is used without regard to its position in a double precision floating point number. These statistics are used to compress the data byte wise instead of per double precision floating point value. Although this is not the most efficient means of compressing the data, it reduces the number of entries in the statistics table from over 2^{64} to 2^8.

3 Results

The initial results from this scheme are very encouraging. Table 1 shows the compression ratios achieved in test cases for both grid files and solution files. Although the compression ratios for solution files are not very high, actual compression is only part of the goal. The progressive nature of the scheme also aids in transmitting the most important data quickly. There is great potential to improve these compression ratios in the future as will be discussed in the next section.

Table 2 shows the percentage of the coefficients used to reconstruct the data set for each pass through the reconstruction algorithm. It also shows the relative size of the accumulated data compared to the size of the original file for one of the test cases.

The reconstruction of the first pass is too coarse in many cases to show the researchers much (Fig. 6). The first pass can be combined with the second pass to save CPU time.

Rec Pass	% Coeffs	% Data
Pass 1	1.56%	1.46%
Pass 2	6.25%	5.83%
Pass 3	50%	46.63%
Pass 4	100%	93.25%

Table 2: Reconstruction Coefficients

The second pass (Fig. 7) provides enough resolution to enable researchers to decide if a solution is acceptable or not. If it is unacceptable, transmission of the data can be terminated. The third pass (Fig. 8) gives a very good reconstruction and is accurate enough to begin analysis of many aspects of a solution. Finally, pass 4 (Fig. 9) perfectly reconstructs the original data to give researchers full confidence in their analysis. Note that the multiresolutional representation of the original data is smaller than the original (PLOT3D) representation.

4 Conclusions and Future Work

The compression technique discussed here holds much promise for compressing three dimensional floating point data. There are currently few or no alternatives, so any work in this area provides valuable insight into nature of this problem. Also, since this technique transmits the most important data first, it provides even more compression to its users than is indicated by file size alone. PLOT3D files are a good format with which to work because they are very flexible and are widely used throughout the CFD community.

There are still many aspects of this technique that need improvement. The current code can be streamlined to speed up the compression process. It could also work much faster if it was implemented on parallel machines. Another improvement could be made by examining the relationship between compression time and transmission time to help determine the optimal compression ratio. Much work can also be done to generalize the method for use with many different file structures.

Two other areas that need to be addressed are the wavelets used and the Huffman coding scheme. Haar wavelets are most likely not the best choice for this compression scheme, so comparisons can be made with other wavelets to find one better suited to both this method and the type of data being compressed. Another promising improvement is to increase the number of Huffman statistics tables so that different parts of the double floating point data are treated separately. For example, the bytes near the end of the mantissa will in most cases always be zero. If run length coding were employed here, it could have a great impact on the compression ratio.

This compression and transmission scheme attempts to move data compression toward time-varying three dimensional scientific data instead of text and image compression, which have already been addressed extensively. It will hopefully impact the scientific community by speeding up analysis of the vast amounts of data produced by researchers.

References

[1] H. Tao and R. Moorhead. Progressive transmission of scientific data using biorthogonal wavelet transform. In *Proc. Visualization '94*, pages 93–99. IEEE, 1994.

[2] P. Walatka and P Buning. *Plot3D User's Manual*, volume version 3.5, chapter 8. NASA, 1988.

[3] G. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):39–41, 1991.

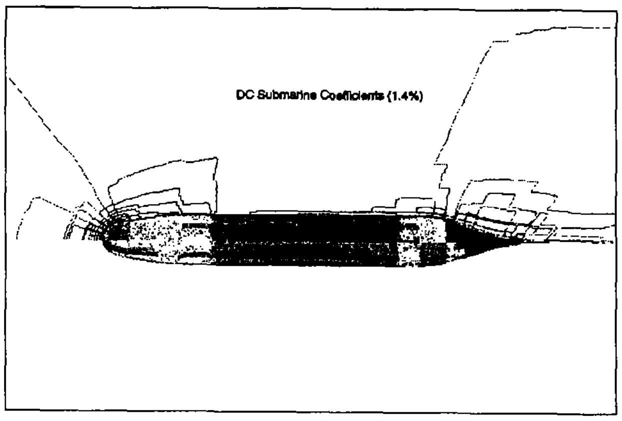

Figure 6: Reconstruction Using 1.56% of Coefficients.

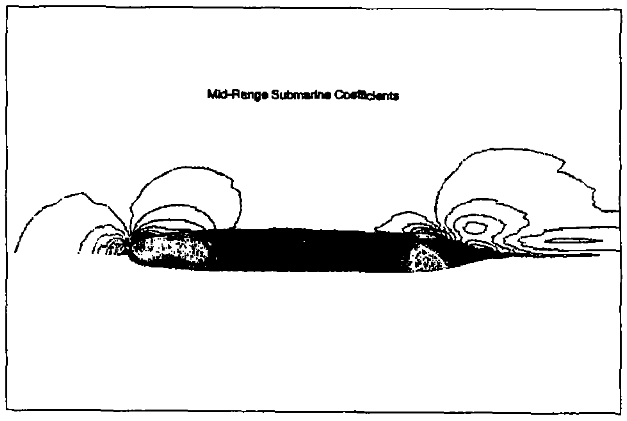

Figure 8: Reconstruction Using 50% of Coefficients.

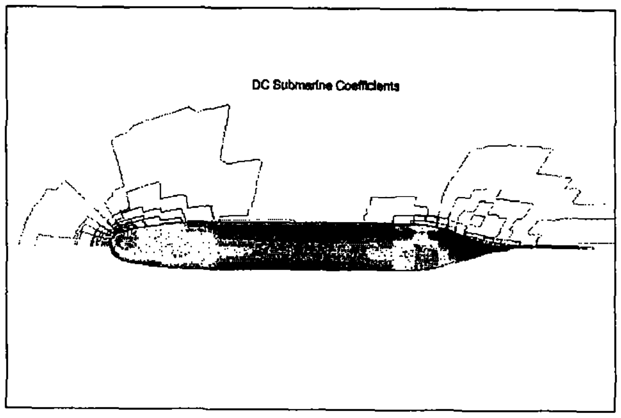

Figure 7: Reconstruction Using 6.25% of Coefficients.

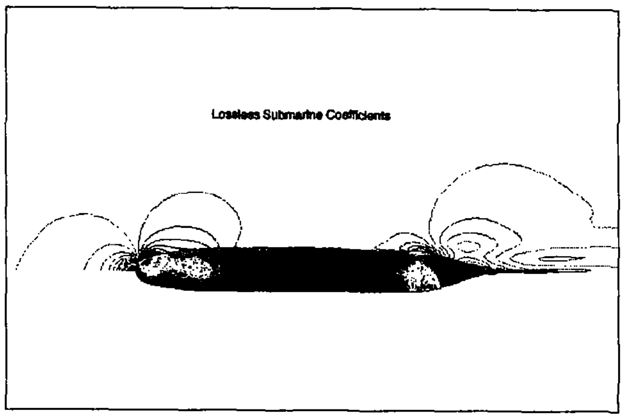

Figure 9: Pass 4 – Lossless Reconstruction.

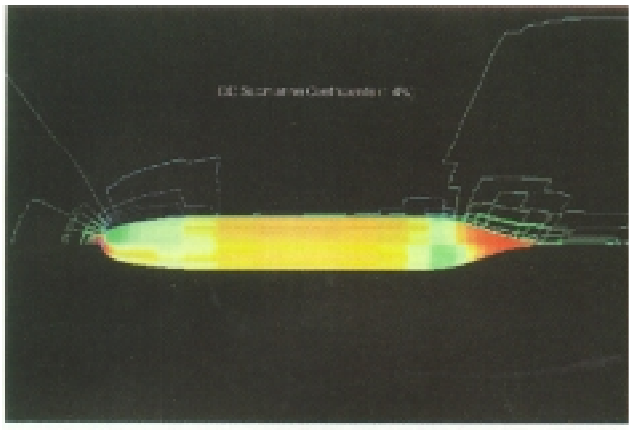

Figure 6: Reconstruction Using 1.56% of Coefficients.

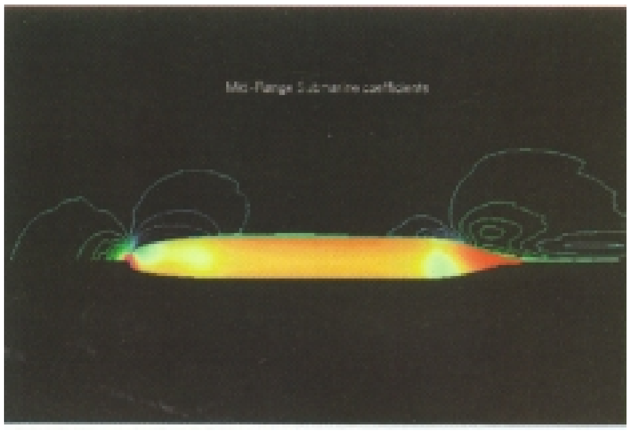

Figure 8: Reconstruction Using 50% of Coefficients.

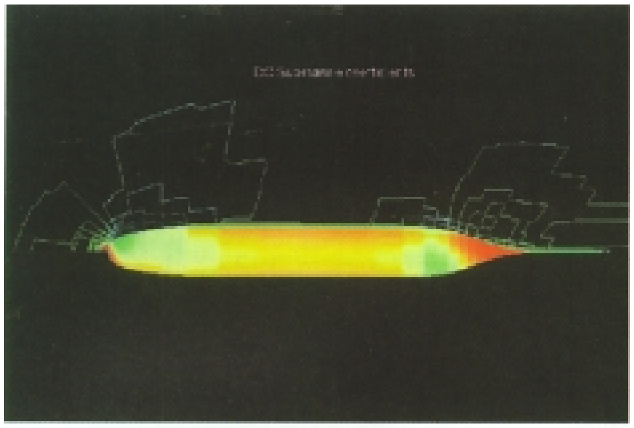

Figure 7: Reconstruction Using 6.25% of Coefficients.

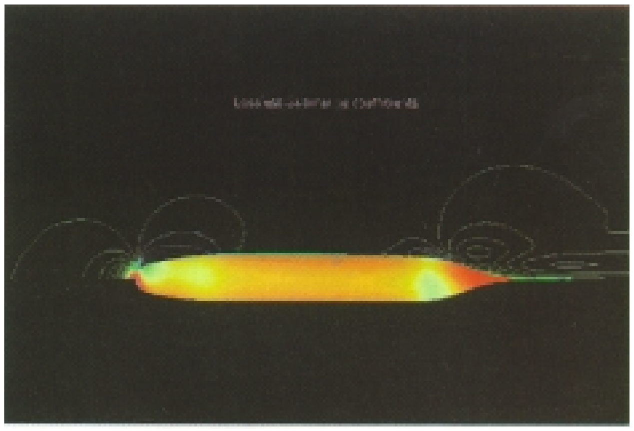

Figure 9: Pass 4 – Lossless Reconstruction.