CUDA-Accelerated HD-ODETLAP: Lossy High Dimensional Gridded Data Compression*

W. Randolph Franklin, You Li, Tsz-Yam Lau, and Peter Fox

Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY, USA 12180
mail@wrfranklin.org, liy13@cs.rpi.edu, laut@cs.rpi.edu, pfox@cs.rpi.edu

Abstract. We present *High-dimensional Overdetermined Laplacian Partial Differential Equations* (HD-ODETLAP), a high dimensional lossy compression algorithm and CUDA implementation that exploits data correlations across multiple dimensions of gridded GIS data. Exploiting the GPU gives a considerable speedup. In addition, HD-ODETLAP compresses much better than JPEG2000 and 3D-SPIHT, when fixing either the average or the maximum error.

1 Introduction

This paper reports the use of recent parallel computing advances to address the problem of storing increasing volumes of geospatial data. The good compression algorithms needed to maximize the feasible dataset size are quite computation-intensive. Recent datasets also have a multi-dimensional structure that is not exploited by current compression algorithms. In this paper, we introduce **HD-ODETLAP**, a High-Dimensional Over-determined Laplacian Partial Differential Equation compression algorithm and implementation. HD-ODETLAP has various versions, such as 4D-ODETLAP and 5D-ODETLAP.

Our test datasets are the *World Ocean Atlas 2005* and *2009* [21] from the National Oceanographic Data Center (NODC) and the National Geophysical Data Center (NGDC), subsidiaries of the NOAA Atmospheric Administration (NOAA). They contain *temperature, salinity, nitrate*, and *silicate* over 12 months at 24 standard depths in the ocean. They can be considered to be five-dimensional: *(x, y, z, time, property)*, of size $180 \times 360 \times 24 \times 12 \times 4$. The data is autocorrelated along each dimension: small changes in any coordinate cause small changes in the measured quantity. This applies even to the fifth dimension; the observed variables do, to some extent, vary in concert.

However, current compression methods treating the data as a single-dimensional stream of bytes ignore that. To the extent that HD-ODETLAP can exploit this, HD-ODETLAP will compress better. In fact, we observe that HD-ODETLAP features a much higher compression ratio, either with user-specified maximum or average error limits, than do conventional approaches. The problem is that HD-ODETLAP is very compute-intensive, and so, parallelization is desirable.

* This research was partially supported by NSF grants CMMI-0835762 and IIS-1117277.

ODETLAP, initially in a 2D version, was developed as part of a project to lossily compress terrain to facilitate operations like multi-observer siting and then path planning to avoid the observers (the *Smugglers and Border Guards Problem*). A related goal was to interpolate terrain from isolated data points and contour lines. Such data often has limited precision. However, the slope should be continuous across lines of data points. ODETLAP addresses these problems well [1, 5, 9, 10, 15, 16, 17, 19, 20].

2 Compression Basics

Various compression methods have been created for lower dimensional data, such as 3D image sequences [13] and 4D Functional magnetic resonance imaging (FMRI) [14]. Among those methods, the wavelet-based lossy and lossless ones are the most popular. For example, JPEG 2000 [18] uses irreversible and reversible wavelet transform for its lossy and lossless compression for 2D images. For 3D video data, Three Dimensional Set Partitioning in Hierarchical Trees (3D-SPIHT) [4] also bases its compression scheme on 3D wavelet transforms. There are some 4D compression algorithms, such as 4D wavelets [6], run length encoding (RLE) [22] and discrete cosine transform (DCT) [3].

Data compression techniques may be lossless or lossy. Lossless schemes allow exact reconstruction of the original data but suffer from a low compression ratio. Lossy schemes produce much more compact datasets, often at a modest cost in reconstruction accuracy, which might still be better than the original dataset's accuracy. Therefore, essentially, nothing is lost. HD-ODETLAP is lossy.

Compression algorithms operate by exploiting redundancies and correlations in the data. Current techniques that compress high dimensional data by compressing 2D or 3D slices separately ignore the correlation between the slices. We don't.

In GIS, there is an increasing awareness of the significance of compression and progressive transmission of high dimensional spatial data. [11] introduces adaptive run-time data compression of spatial data. [12] comprehensively studied different compression schemes for efficient maintenance and dissemination of spatial databases. Research on compression and spatial decorrelation has been done in digital terrain models, [8]. We have published a 3D oceanographic data compression method, [2].

3 5D-ODETLAP

Definition *Five Dimensional Over-Determined Laplacian Partial Differential Equations* (5D-ODETLAP) is an extension of the Laplacian PDE $\frac{\delta^2 z}{\delta x^2} + \frac{\delta^2 z}{\delta y^2} = 0$ to an overdetermined linear system. It comprises two types of equations. First, every point, known or unknown, induces an averaging equation that sets its value to the average of its neighbors. Second, we add another equation for each known point, $v_i = s_i$ where s_i stands for the known value of the point, and v_i its "computed" value. Since ODETLAP is lossy, these values will differ slightly. This

system is over-determined, since there are equations than unknowns. ODETLAP, being over-determined, has essentially different mathematical properties from a Laplacian, in spite of the superficial similarities. For example, with ODETLAP, unlike with a Laplacian, interior local extrema can be inferred. Also, unlike with a Laplacian PDE, the inferred surface is more continuous across lines of known points, and so, the data points are not as visible (or not at all visible) in the generated surface.

The least squares solution to this system can be biased by weighting different equations differently using a multiplicative parameter $\mathbf{R}$ on both sides of each equation. $\mathbf{R}$ trades off smoothness (large $\mathbf{R}$) versus accuracy (small $\mathbf{R}$) at that point. Therefore, inconsistent datasets of varying accuracy can be conflated.

Framework From the $90 \times 180 \times 24 \times 12 \times 4$ 5D dataset, we selected a random subset S of the points, and then used ODETLAP to compute an initial approximation of the dataset. Indeed, we applied 4D reconstructions to each separate 4D slice of the 5D dataset and then merged them to form the 5D approximation. Then we iterated until reaching some stopping condition, such as a specified average mean percentage error of all 4D datasets within the 5D dataset (to compensate for each 4D dataset having a different data range).

The repeated iteration goes as follows. First, select the points whose computed values have the largest relative errors. Then insert them into S and re-solve with ODETLAP.

The representation of the compressed dataset is the set of positions and values of the elements of S. We code the positions with a standard binary compression algorithm, and code the values with a floating point compression algorithm. These choices produce a good result but could be improved.

The biggest challenge is that solving overdetermined sparse linear systems is compute-intensive. The normal equations transformation, taking $Ax = b$ to $A^T Ax = A^T b$, reduces the time considerably, but the computation is still slow. For example, solving a 104976×104976 linear system takes up to 58 GB main memory and about 1.8 hours on a workstation with four 2.4GHz processors and 60 GB of main memory running Ubuntu 10.04.2 and 64-bit Matlab R2009a.

3.1 CUDA-Based Solver

Solver Introduction CUDA by Nvidia provides a widely used developer-friendly GPGPU interface. CUSP [7] is an open source library for sparse linear algebra computations using CUDA. It provides a flexible, high-level interface for manipulating sparse matrices and solving sparse linear systems. The CUSP library contains two iterative linear solvers, the Conjugate Gradient solver (CG) and Biconjugate Gradient Stabilized solver (BICGS) with an optional Jacobi preconditioner. With a proper construction of the linear system, a simple function call in Matlab can solve the linear system using the GPU computing power without any prior knowledge about CUDA GPU programming.

Solver Selection and Matlab Integration Matlab's current CF solver works directly, which is both compute-intensive and unnecessary because a precise solution is not required. Since 5D-ODETLAP is lossy, the conjugate gradient iterative solver in the CUSP library is ideal.

First, we tested the CUSP solver on a linear system of size 234256×234256 constructed by 5D-ODETLAP. The CUSP CG solver (179 seconds) was much more efficient than the CF direct solver (49237 seconds) and CG solver (5495 seconds). However, of that 179 seconds, only 9 seconds was spent on actually solving the linear system. The remaining 170 seconds was spent on the data transfer between 5D-ODETLAP, implemented in Matlab, and the CUSP solver as a C++ executable. To improve this, we utilized the Matlab Executable (MEX), which allows users to interface C, C++ or Fortran subroutines to MATLAB. MEX-files are a way to call the custom C, C++ or FORTRAN routines directly from MATLAB as if they were MATLAB built-in functions. Therefore we can largely reduce the overhead for transferring data between the Matlab program and the CUSP library. But since the CUSP library is implemented both in C++ and CUDA, this adds a certain amount of complexity to incorporate it into the MEX file. Fortunately, CUDA compiler allows users to compile CUDA code into C++ code as an intermediate step using *nvcc –cuda*. So first we write the CUSP code in a MEX style. Then this mixed source code will be compiled into C++ code, which can then be compiled into a MEX file and called directly from the Matlab program.

Table 1. Effective solver time efficiency comparisons between the Matlab Cholesky Factorization (CF) direct solver, Matlab CG solver, CUSP CG solver, CUSP CG solver with Jacobi preconditioner, CUSP BICGS solver with preconditioner and CUSP CG solver. The time measurement is in seconds.

System Size	**Cholesky Factor. Solver**	**Matlab CG**	**CUSP CG**	**CUSP CG Jacobi**	**CUSP BICGS Jacobi**
4096^2	2.46	0.63	0.23	0.20	0.21
6561^2	6.49	1.11	0.56	0.30	0.36
10000^2	17.43	2.01	0.52	0.48	0.55
14641^2	43.02	3.02	0.75	0.70	0.77
20736^2	93.20	5.55	0.94	0.92	1.08
28561^2	203.25	5.40	1.29	1.28	1.32
38416^2	454.41	13.35	1.91	1.74	1.86
50625^2	816.90	15.58	2.28	2.20	2.34
65536^2	1791.78	29.30	2.89	2.54	3.03
83521^2	3332.05	30.67	3.57	3.57	3.41
104976^2	6488.24	39.67	4.86	4.84	5.14

In Table 1, a comparison of solvers in Matlab and CUSP demonstrates that for the linear system from 5D-ODETLAP, the CUSP CG solver with Jacobi preconditioner runs the fastest. The linear sysytems in this table are constructed

from original 4D datasets ranging from 8^4 to 18^4, so the resulting linear systems' size ranges from 4096×4096 to 104976×104976. This solver runs more than seven times faster (39.67:4.84) than its CPU counterpart with the same residual size of their solutions in the test linear system of size 104976×104976. And, it's more than 1340 times faster than the CF direct solve in Matlab. Not only is the CG solver with Jacobi preconditioner from CUSP better in terms of running time, it uses only 13.2 GB main memory and less than 512 MB device memory on GPU on the workstation described above.

4 Comparison with JPEG 2000 and 3D-SPIHT

Our test of 5D-ODETLAP is based on two real world geospatial 5D datasets—World Ocean Atlas 2005 (WOA05) and WOA 2009 (WOA09), from the National Oceanographic Data Center. WOA05 and WOA09 contain sets of objectively analyzed (1 degree grid) climatological fields of in situ *temperature, salinity, dissolved oxygen, Apparent Oxygen Utilization (AOU), percent oxygen saturation, phosphate, silicate,* and *nitrate* at standard depth levels for annual, seasonal, and monthly compositing periods for the global ocean. They serve as examples of high dimensional geospatial datasets. The monthly data we derive from WOA05 and WOA09 is $180 \times 360 \times 24 \times 12 \times 4$, which stores the *temperature, AOU, percentage oxygen saturation* and *dissolved oxygen* for 12 months. We geographically divide these two datasets into 8 different datasets each of size $90 \times 180 \times 24 \times 12 \times 4$, named WOA05_1, WOA05_2, WOA05_3, WOA05_4, WOA09_1, WOA09_2, WOA09_3 and WOA09_4. This allows us to test the robustness of 5D-ODETLAP on 8 distinct datasets, to see how sensitive it is to the particular data. Table 2 shows the properties of WOA09_1.

Table 2. Properties of WOA09_1

Variable	**Unit**	**Data Range**	**Size(MB)**
Apparent oxygen utilization	ml^{-1}	[-2.29, 6.57]	17.80
Percent oxygen saturation	%	[0, 131.42]	17.80
Dissolved oxygen	ml^{-1}	[0.05, 11.10]	17.80
Temperature	$^{\circ}C$	[-2.10, 31.18]	17.80

Since 3D-SPIHT and JPEG 2000 are among the most popular lossy compression methods, we compared 5D-ODETLAP to them. We assume each value in WOA05 and WOA09 datasets is stored as a single precision floating point, but the actual storage of the data is not so precise (seven digit real number with 4 places to the right of the decimal). So the compression ratio comes in part from the truncation of digits for storage, but this does not affect the comparison. To have a fair comparison, we used binary search in 3D-SPIHT to find a suitable bit rate to produce results with the same mean percentage error in the one test and the same maximum percentage error in the other. The $90 \times 180 \times 24 \times 12 \times 4$ 5D

Table 3. Compression comparison between 5D-ODETLAP, JPEG 2000 and 3D-SPIHT with the same **mean** percentage error on eight different datasets. The last two columns show the ratio of 5D-ODETLAP's compression ratio over the compression ratio of JPEG 2000 and 3D-SPIHT respectively.

Dataset	**% Fixed Mean Err**	**% Max Err, JPEG 2000**	**% Max Err, 3D-SPIHT**	**% Max Err, 5D-ODETLAP**	**Ratio** $(\frac{5D-ODETLAP}{JPEG2000})$	**Ratio** $(\frac{5D-ODETLAP}{3D-SPIHT})$
woa05_1	1.42	44.83	66.46	10.41	8.16	2.40
woa05_2	1.48	49.33	59.18	9.35	9.56	3.61
woa05_3	1.47	65.56	80.23	8.94	4.24	1.13
woa05_4	1.56	67.56	74.14	10.81	8.57	2.41
woa09_1	1.46	48.13	68.18	9.02	8.18	2.41
woa09_2	1.49	51.21	62.15	8.77	9.80	3.75
woa09_3	1.54	75.00	79.35	11.13	4.24	1.14
woa09_4	1.58	65.55	71.50	11.58	8.58	2.42

dataset contains 12×4 3D datasets of size $90 \times 180 \times 24$. We applied 3D-SPIHT on each of these 3D datasets and measured the error together as a 5D reconstruction in decompression.

Similarly, we used binary search to find a compression ratio in JPEG 2000 to ensure results with the same error in both cases. We also applied JPEG 2000 to each 90×180 2D dataset of the overall $24 \times 12 \times 4$ number of 2D datasets. In addition, since JPEG 2000 takes only unsigned 1, 8 or 16 bit integer input, we first used uniform quantization to reduce each input 2D dataset to 16 bit unsigned integer. Thus, all the three methods had the same mean or maximum percentage error on all the 8 test datasets, to produce a fair comparison.

Table 3, with data from [1], compares the three methods for the same *mean* percentage error. The first observation is that the maximum percentage error for 5D-ODETLAP is much smaller than that of both JPEG 2000 and 3D-SPIHT. This advantage of 5D-ODETLAP is credited to its iterative greedy sampling process, since it eliminates the points with the largest error at each iteration by adding them into the sample set. The JPEG 2000 and 3D-SPIHT methods do not have this adaptability, and thus produce a much larger maximum percentage error. Second, the compression ratio of 5D-ODETLAP is generally 4.24–9.8 times as large as that of the JPEG 2000 method and 1.13–3.75 times as large as that of the 3D-SPIHT method.

Table 4, with data from [1], shows the result of forcing the *maximum* percentage error to be the same for all three methods. 5D-ODETLAP's mean error is larger than the others because it spreads out the error more evenly. However 5D-ODETLAP's compression ratio is even larger than in the previous fixed mean percentage error case. The utility of this comparison is that maximum error is essential in some applications. Perhaps the user needs to have a compressed file with guaranteed no more than 10% error. Here, the compression ratio of JPEG 2000 and 3D-SPIHT will be considerably worse than 5D-ODETLAP, since

Table 4. Compression comparison between 5D-ODETLAP, JPEG 2000 and 3D-SPIHT with approximately the same **maximum** percentage error on eight different datasets. The last two columns show the ratio of 5D-ODETLAP's compression ratio over the compression ratio of JPEG 2000 and 3D-SPIHT respectively.

Dataset	**% Fixed Max Err**	**% Mean Err, JPEG 2000**	**% Mean Err, 3D-SPIHT**	**% Mean Err, 5D-ODETLAP**	**Ratio** $(\frac{5D-ODETLAP}{JPEG2000})$	**Ratio** $(\frac{5D-ODETLAP}{3D-SPIHT})$
woa05_1	10.41	0.52	0.37	1.42	16.33	11.51
woa05_2	9.35	0.32	0.35	1.48	24.02	15.74
woa05_3	8.94	0.26	0.28	1.47	13.84	9.47
woa05_4	10.81	0.37	0.34	1.56	22.23	14.91
woa09_1	9.02	0.39	0.29	1.46	18.97	13.87
woa09_2	8.77	0.31	0.36	1.49	24.43	15.51
woa09_3	11.13	0.36	0.31	1.54	12.05	8.91
woa09_4	11.58	0.39	0.34	1.58	21.92	15.22

the mean error of the compressed file is unnecessarily small. Comparatively, 5D-ODETLAP's iterative sampling process ensures that the points with largest error are always selected, which reduces the maximum error at each step.

5 Conclusion and Future Work

5D-ODETLAP demonstrates efficiently applying a massively parallel GPU to an important GIS problem — compressing multidimensional GIS data. 5D-ODETLAP also exploits spatial and temporal redundancies in the data better than previous methods. 5D-ODETLAP's potential impact extends to multidimensional datasets in other domains, such as computational fluid dynamics (CFD).

Currently 5D-ODETLAP only partially exploits correlations between 4D datasets within each 5D dataset. Also, its coding of the bitmap denoting the points in S, and its compression of the floating values at those points might be improvable. Finally, overdetermined extensions to other PDEs remain to be investigated. Therefore, we expect even better compression of high-dimensional data in the future.

References

[1] Y Li. "CUDA-accelerated HD-ODETLAP: a high dimensional geospatial data compression framework". PhD thesis. Rensselaer Polytechnic Institute, 2011.

[2] Y Li, TY Lau, C Stuetzle, P Fox, and WR Franklin. "3D oceanographic data compression using 3D-ODETLAP". In: *18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2010)*. (PhD Dissertation showcase). San Jose, CA, USA, 2010.

[3] EB Lum, KL Ma, and J Clyne. "Texture hardware assisted rendering of time-varying volume data". In: *VIS '01: Proceedings of the conference on Visualization '01*. San Diego, California: IEEE Computer Society, 2001, pp. 263–270.

[4] BJ Kim and W Pearlman. "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)". In: *Data Compression Conference, 1997. DCC '97. Proceedings*. 1997, pp. 251–260.

[5] Z Xie. "Representation, Compression and Progressive Transmission of Digital Terrain Data Using Over-Determined Laplacian Partial Differential Equations". MA thesis. Rensselaer Polytechnic Institute, 2008.

[6] W Yang et al. "4-D wavelet-based multiview video coding". In: *IEEE Transactions on Circuits and Systems for Video Technology* 16.11 (2006), pp. 1385–1396.

[7] N Bell and M Garland. *CUSP: Generic Parallel Algorithms for Sparse Matrix and Graph Computations*. `http://cusp-library.googlecode.com`. Version 0.1.0. 2010.

[8] JT Bjøke and S Nilsen. "Efficient representation of digital terrain models: compression and spatial decorrelation techniques". In: *Computers & Geosciences* 28.4 (2002), pp. 433–445.

[9] M Inanc. "Compressing Terrain Elevation Datasets". PhD thesis. Rensselaer Polytechnic Institute, 2008.

[10] DM Tracy. "Path Planning and Slope Representation on Compressed Terrain". PhD thesis. Rensselaer Polytechnic Institute, 2009.

[11] A Plaza, J Plaza, and A Paz. "Improving the scalability of hyperspectral imaging applications on heterogeneous platforms using adaptive run-time data compression". In: *Computers & Geosciences* 36.10 (2010), pp. 1283–1291.

[12] DB Kidner and DH Smith. "Advances in the data compression of digital elevation models". In: *Computers & Geosciences* 29.8 (2003), pp. 985–1002.

[13] G Menegaz and JP Thiran. "Lossy to lossless object-based coding of 3-D MRI data". In: *IEEE Transactions on Image Processing* 11.9 (Sept. 2002), pp. 1053–1061.

[14] H Lalgudi, A Bilgin, M Marcellin, and M Nadar. "Compression of fMRI and ultrasound images using 4D SPIHT". In: *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. Vol. 2. 2005, pp. II –746–9.

[15] J Muckell. "Evaluating and Compressing Hydrology on Simplified Terrain". MA thesis. Rensselaer Polytechnic Institute, 2008.

[16] J Stookey. "Parallel Terrain Compression and Reconstruction". MA thesis. Rensselaer Polytechnic Institute, 2008.

[17] J Stookey et al. "Parallel ODETLAP for terrain compression and reconstruction". In: *16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS 2008)*. Ed. by WG Aref et al. Irvine CA, 2008.

[18] DS Taubman, MW Marcellin, and M Rabbani. "JPEG2000: Image compression fundamentals, standards and practice". In: *Journal of Electronic Imaging* 11 (2002), p. 286.

[19] WR Franklin. "The RPI GeoStar project". In: *25th International Cartographic Conference*. Paris, 2011.

[20] WR Franklin, M Inanc, and Z Xie. "Two Novel Surface Representation Techniques". In: *Autocarto 2006*. Cartography and Geographic Information Society. Vancouver Washington, 2006.

[21] RA Locarnini et al. "World Ocean Atlas 2009, Volume 1: Temperature". In: NOAA Atlas NESDIS 68 (2010). Ed. by S Levitus, p. 184.

[22] K Anagnostou, TJ Atherton, and AE Waterfall. "4D volume rendering with the Shear Warp factorisation". In: *Proceedings of the 2000 IEEE symposium on Volume Visualization*. VVS '00. Salt Lake City, Utah, United States: ACM, 2000, pp. 129–137.