Recalling The Single-FFT Direct Poisson Solve

James McCann*
Carnegie Mellon University

1 Introduction

Large Poisson's equation problems arise in gradient-domain image compositing. Agarwala [2007] and Kazhdan and Hoppe [2008] recently presented iterative solution methods using a smooth correction term and multigrid with higher-order elements, respectively. In both papers, however, direct solution methods are somewhat glossed over. This work aims to remedy these omissions by presenting timing numbers for a classic single-FFT – so-called because it performs a fast Fourier transform only on one dimension of the image – solution method.

2 Method

The single-FFT method summarized here has been known for many years; I have found mentions as early as 1965 [Hockney 1965]. The idea is to decouple the column solutions by performing an FFT on each row, then solve the columns using simple row-reduction.

Poisson's equation is

$$\nabla^2 u = f \tag{1}$$

That is, we wish to solve for image u with given Laplacian f. Often, ∇^2 is approximated by a five-point stencil (-4 in the center, with 1's adjacent). Note that if one takes pixels outside the image to have the same color as the nearest pixels inside the image, this leaves u undefined up to an additive constant.

The first step of the single-FFT algorithm involves changing the basis of the rows by running a DCT-II ('discrete cosine transform' – a variant of the fast Fourier transform) on f. This separates our equation into a set of 1-d problems:

$$\begin{bmatrix} 1 \\ -4 + \cos(\pi \frac{i}{w}) \\ 1 \end{bmatrix} * c_i' = f_i' \tag{2}$$

where we now solve for column vectors c_i', $0 \leq i < w$ (the frequency bands of the image) based on f_i' (frequency bands of the DCT'd f).

These new problems have a tridiagonal matrix structure amenable to solution with row-reduction. Performing this row-reduction in lockstep for all the columns at once gives us the code in Figure 1. Notice that this code is trivial to perform out-of-core, requiring only that one store the new right-hand-side and value on the diagonal between the first (forward) and second (backward) row-reduction passes. In practice, the regular structure of the matrix allows simple run-length encoding to compress away most of the value-on-diagonal overhead – requiring only a few megabytes more storage even on the 87 megapixel redrock image.

3 Comparison

I ran the single-FFT algorithm on the same datasets as used in the more recent papers, and report the results in Table 1. Timing information for the previous approaches comes from Kazhdan and Hoppe's table [2008]. The single-FFT algorithm is slower, but decidedly not glacially so, and the error accumulation due to numerical imprecision is far lower than the error in the iterative methods.

```
Single-FFT-Solve:
 for each row:
    perform DCT-II to decouple columns
    subtract previous row to eliminate below-diagonal
 for each row (reverse order):
    subtract next row to eliminate above-diagonal
    perform DCT-III to get final answer
```

Figure 1: *Pseudocode for the single-FFT solution. Notice that only two rows need to be stored in memory at once, and only two passes made over the image. The DCT-III is also known as the 'inverse discrete cosine transform'.*

		Time (s)			Max Error		
Image	MP	SM	QT	SFFT	SM	QT	SFFT
beynac	12	17	8	34	3e-4	5e-3	1e-13
rainier	23	33	14	76	3e-4	4e-3	1e-12
edinburgh	50	79	122	190	3e-4	·	5e-12
redrock	87	118	188	333	4e-4	·	3e-12

Table 1: *Comparison of the single-FFT method to more recent approaches. MP is the panorama size in megapixels. SM is the method of Kazhdan and Hoppe, QT is the method of Agarwala. Panorama data courtesy Aseem Agarwala. The single-FFT method is mathematically exact; reported errors are calculated numerical error on stress-test noise images of the same dimensions as the provided data.*

Additionally, the memory usage of the single-FFT method is much lower than either of the previous methods (e.g. 80MB for redrock, as compared to SM's 133MB and QT's 112MB).

4 Conclusion

In this paper I presented timing information for a classic direct solution method for Poisson's equation, demonstrating it to produce orders-of-magnitude more accurate solutions with runtimes roughly twice that of modern iterative approaches. One downside of the approach is that these low numerical error rates were only achievable with double-precision intermediate values, quadrupling the storage requirements over streaming multigrid's half-float intermediate values. One could mitigate this problem by storing only every nth line of the intermediate data and recomputing parts of the forward sweep on the fly. Another interesting direction would be to use integer FFT's and rational number arithmetic to eliminate numerical error entirely. I hope that this work has provided some incentive for the community to continue to investigate and remember older, direct, methods for solution of Poisson's equation.

References

AGARWALA, A. 2007. Efficient gradient-domain compositing using quadtrees. *ACM Transactions on Graphics 26*, 3.

HOCKNEY, R. W. 1965. A fast direct solution of poisson's equation using fourier analysis. *J. ACM 12*, 1, 95–113.

KAZHDAN, M., AND HOPPE, H. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on Graphics 27*, 3.

*e-mail: jmccann@cs.cmu.edu