Analysis of Error in a CML Diffusion Operation

Mark J. Harris
University of North Carolina at Chapel Hill

1 Introduction

This report supports the paper [Harris, et al. 2002], which describes the implementation of various CML simulations using computer graphics hardware. Because of the limited precision available for fragment computations in graphics hardware, it is important to understand the effect of roundoff error propagation in these computations. Diffusion is an operation that is often used in Coupled Map Lattice (CML) simulations. For this reason, it is useful to study the error in CML simulations by studying diffusion.

2 Fixed Point Error Analysis

We will write a fixed-point number x that is the result of an arithmetic operation in a computer as

$$x + \varepsilon$$

where ε is the error in the number resulting from roundoff or truncation [Hamming 1973]. In the case of roundoff, we will let $\varepsilon = \varepsilon_r$, and

$$|\varepsilon_r| < \frac{1}{2} \times 2^{-p},$$

where p is the number of bits of precision to the right of the binary point. If truncation is used instead of roundoff, then we will let $\varepsilon = \varepsilon_t$, and

$$|\varepsilon_t| < 2^{-p}.$$

2.1 Error in Addition

The error from adding two numbers is just the sum of the errors of each of the numbers plus the error induced by the addition itself (roundoff or truncation):

$$x_3 + \varepsilon_3 = (x_1 + \varepsilon_1) + (x_2 + \varepsilon_2) + \varepsilon$$

2.2 Error in Multiplication

The error from multiplying two numbers is slightly more complicated:

$$x_3 + \varepsilon_3 \quad = (x_1 + \varepsilon_1)(x_2 + \varepsilon_2) + \varepsilon = x_1 x_2 + \varepsilon_1 x_2 + \varepsilon_2 x_1 + \varepsilon_1 \varepsilon_2 + \varepsilon$$

Since $|\varepsilon_1 \varepsilon_2| < \varepsilon_r^2$, we can typically disregard this term, as it will be very small. Thus,

$$x_3 + \varepsilon_3 \quad = x_1 x_2 + \varepsilon_1 x_2 + \varepsilon_2 x_1 + \varepsilon$$

2.3 Diffusion

In one dimension, a discrete diffusion operator can be written

$$x_i' = (1-d)x_i + \frac{d}{2}(x_{i-1} + x_{i+1}), \qquad \textbf{(1)}$$

where d is the diffusion coefficient. In what follows, we will assume that there is no error in d itself. We will also assume that there is no error in $d / 2$, since fixed-point division by powers of 2 does not induce any roundoff error. Thus, equation (1) becomes

$$x_i' + \varepsilon_i' = \left[(1 - d + \varepsilon)(x_i + \varepsilon_i) + \varepsilon\right] + \left[\frac{d}{2}\left[\left(x_{i-1} + \varepsilon_{i-1}\right) + \left(x_{i+1} + \varepsilon_{i+1}\right) + \varepsilon\right] + \varepsilon\right] + \varepsilon$$

where the error from each multiplication and addition, ε, has been added and grouped to correctly propagate it into the total error. This expression simplifies to (dropping the small $\varepsilon\varepsilon_i$ term)

$$x_i' + \varepsilon_i' = (1-d)x_i + \frac{d}{2}\left(x_{i-1} + x_{i+1}\right) + (1-d)\varepsilon_i + \frac{d}{2}\left(\varepsilon_{i-1} + \varepsilon_{i+1}\right) + \varepsilon(3 + \frac{d}{2} + x_i). \qquad \textbf{(2)}$$

We now make the assumption that $\varepsilon_i \approx \varepsilon_{i-1} \approx \varepsilon_{i+1}$. The justification for this is that the error in all cells comes from the same arithmetic computation, since the diffusion operation operates in a SIMD manner. The small dependence of the error on the values themselves will cause some variance, but that variance will diminish as diffusion proceeds, since the values at

nearby lattice nodes will become more homogeneous. Making this assumption, Equation (2) becomes

$$x_i' + \varepsilon_i' = (1-d)x_i + \frac{d}{2}\left(x_{i-1} + x_{i+1}\right) + \varepsilon_i + \varepsilon(3 + \frac{d}{2} + x_i),$$

and therefore, the error after each diffusion step is

$$\varepsilon_i' = \varepsilon_i + \varepsilon(3 + \frac{d}{2} + x_i).$$

So diffusion induces additional error of

$$\varepsilon(3 + \frac{d}{2} + x_i)$$

at each application. Similar analyses show that in 2 dimensions the diffusion error is

$$\varepsilon(3 + \frac{3d}{4} + x_{i,j}),$$

if we ignore the additional error caused by the fact that d / 6 is not exact in binary fixed-point arithmetic. In N dimensions, the error is approximately

$$\varepsilon(3 + \frac{(2N-1)d}{2N} + x_{i,\ldots,n}).$$

If we assume that either diffusion is the only operation at each iteration, or that the errors of other operations are less than or equal to the error in a diffusion operation, then over M iterations, the error at each lattice node i will be

$$\left|\varepsilon_i^M\right| \le \varepsilon_i^0 + \varepsilon\left(M\left[\frac{(2N-1)d}{2N} + 3\right] + \sum_{j=0}^{M} x_{i,\ldots,n}^j\right),$$

where ε^0 is the initial error and x_i^j is the value of lattice node i at iteration j. Since our computations in graphics hardware operate on values the range [0, 1], we may simplify this to (also assuming that the initial error is 0)

$$\left|\varepsilon_{i,j}^M\right| \le 4.75M\varepsilon$$

in the case of two dimensions. Under these assumptions, the error induced at each step is bounded by

$$\left|\varepsilon_{i,j}'\right| \le 4.75\varepsilon. \quad \textbf{(3)}$$

3 Floating Point Error Analysis

Whereas we wrote fixed-point numbers as $x + \varepsilon$, a better representation of floating point error is achieved by writing a floating-point number as $x(1+\varepsilon)$ [Hamming 1973]. This is because roundoff error in a floating-point mantissa is scaled by $2^{exponent}$, which can be approximated by scaling by x itself. For this reason, a similar analysis to the one above reveals that the result of a 2D floating-point diffusion computation is

$$x_{i,j}'(1+\varepsilon_{i,j}') \approx \left(1-d\right)x_{i,j}(1+\varepsilon_{i,j}+2\varepsilon) + \frac{d}{4}\left(x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1}\right)\left(1+\varepsilon_{i,j}+3\varepsilon\right).$$

If we assume as above that x and d are within [0,1], then we can bound the error in this computation as

$$\left|\varepsilon_{i,j}'\right| \le 3\varepsilon. \quad \textbf{(4)}$$

4 Results

Equations (3) and (4) imply that we need at least as many bits in fixed-point numbers as in the mantissa of floating point numbers in order to get the same results for diffusion. This is true, but it may be possible to get useful results with less precision in the floating-point mantissa, which means we could possibly get away with fewer bits in a fixed-point implementation, too.

Since a CML simulation will typically consist of more operations than just diffusion, error propagation becomes more complex in reality. We have implemented a simple CML simulation of reaction-diffusion that can operate using either floating-point or fixed-point numbers. By experimenting with the number of bits used in the fractional part of the fixed-point numbers, we have found that the fixed-point simulation is visually very similar to the floating-point simulation when 14 or more bits are used for computation.

References

[Hamming 1973] Hamming, R.W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill. 1973.

[Harris, et al. 2002] Harris, M.J., Coombe, G., Scheuermann, T. and Lastra, A. Physically-Based Visual Simulation on Graphics Hardware. *Submitted to Graphics Hardware 2002*. 2002.