eWave: Using an Exponential Solver on the iWave Problem

Jerry Tessendorf

March 16, 2014

1 iWave Dynamics

The iWave equations of motion for the surface displacement h and velocity potential ϕ are:

$$\frac{\partial h(\mathbf{x},t)}{\partial t} = \sqrt{-\nabla^2}\phi(\mathbf{x},t) \quad (1)$$

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = -gh(\mathbf{x},t) \quad (2)$$

$$(3)$$

To see how the exponential solver approach helps, it would be good to set up a different way to express these equations in terms of a couplet

$$W(\mathbf{x},t) = \begin{bmatrix} h(\mathbf{x},t) \\ \phi(\mathbf{x},t) \end{bmatrix} \quad (4)$$

The couplet has the equation of motion

$$\frac{\partial W(\mathbf{x},t)}{\partial t} = \mathcal{M}\ W(\mathbf{x},t) \quad (5)$$

where $\mathcal{M}$ is the matrix

$$\mathcal{M} = \begin{bmatrix} 0 & \sqrt{-\nabla^2} \\ -g & 0 \end{bmatrix} \quad (6)$$

2 eWave: Exponential Solution

Equation 5 has the exact exponential solution

$$W(\mathbf{x},t) = \exp\{\mathcal{M}t\}\ W(\mathbf{x},0) \quad (7)$$

This can be built more explicitly, because if you work it out, you see the following identities:

$$\mathcal{M}^2 = -\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} g\sqrt{-\nabla^2} \tag{8}$$

$$\mathcal{M}^3 = -\mathcal{M}\, g\sqrt{-\nabla^2} \tag{9}$$

$$\mathcal{M}^{2n} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left(-g\sqrt{-\nabla^2}\right)^n \tag{10}$$

$$\mathcal{M}^{2n+1} = \mathcal{M}\, \left(-g\sqrt{-\nabla^2}\right)^n \tag{11}$$

So, if we expand the exponential into a Taylor series:

$$\exp\{\mathcal{M}t\} = \sum_{n=0}^{\infty} \frac{(\mathcal{M})^n t^n}{n!} \tag{12}$$

and separate the powers into even and odd sets

$$\exp\{\mathcal{M}t\} = \sum_{n=0}^{\infty} \frac{(\mathcal{M})^{2n} t^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \frac{(\mathcal{M})^{2n+1} t^{2n+1}}{(2n+1)!} \tag{13}$$

Using the identities above,

$$\exp\{\mathcal{M}t\} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \sum_{n=0}^{\infty} \frac{(-g\sqrt{-\nabla^2}t^2)^n}{(2n)!} + \mathcal{M}t \sum_{n=0}^{\infty} \frac{(-g\sqrt{-\nabla^2}t^2)^n}{(2n+1)!} \tag{14}$$

The first infinite series is the series for the cosine, and the second is the series for the sine. So if we define $\hat{\omega} \equiv \sqrt{g\sqrt{-\nabla^2}}$

$$\exp\{\mathcal{M}t\} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cos(\hat{\omega}t) + \frac{\mathcal{M}}{\hat{\omega}} \sin(\hat{\omega}t) \tag{15}$$

$$= \begin{bmatrix} \cos(\hat{\omega}t) & \frac{\sqrt{-\nabla^2}}{\hat{\omega}} \sin(\hat{\omega}t) \\ \frac{-g}{\hat{\omega}} \sin(\hat{\omega}t) & \cos(\hat{\omega}t) \end{bmatrix} \tag{16}$$

The nice thing about this is that, because g and ∇^2 are not time dependent quantities, this solution is exact for any value of time t. This is the exact solution for any time period:

$$h(\mathbf{x}, t+\Delta t) = \cos(\hat{\omega}\Delta t)\ h(\mathbf{x}, t) + \frac{\sqrt{-\nabla^2}}{\hat{\omega}} \sin(\hat{\omega}\Delta t)\ \phi(\mathbf{x}, t) \tag{17}$$

$$\phi(\mathbf{x}, t+\Delta t) = \cos(\hat{\omega}\Delta t)\ \phi(\mathbf{x}, t) - \frac{g}{\hat{\omega}} \sin(\hat{\omega}\Delta t)\ h(\mathbf{x}, t) \tag{18}$$

In real space, the tricky part of this is the complex way that ∇^2 is in the solution. Using convolution on this would be similar to the old iWave approach, but still better because the dynamics has been integrated already.

3 FFT Form

In Fourier space, the appearance of ∇^2 simplifies considerable, because now the operator $\hat{\omega}$ becomes the dispersion relation $\hat{\omega} \rightarrow \omega(k) = \sqrt{gk}$, where k is the absolute magnitude of the Fourier vector. For the initial Fourier amplitudes $\tilde{h}(\mathbf{k})$ and $\tilde{\phi}(\mathbf{k})$

$$\tilde{h}(\mathbf{k}, t + \Delta t) = \cos(\omega(k)\Delta t)\ \tilde{h}(\mathbf{k}, t) + \frac{k}{\omega(k)}\ \sin(\omega(k)\Delta t)\ \tilde{\phi}(\mathbf{k}, t) \quad (19)$$

$$\tilde{\phi}(\mathbf{k}, t + \Delta t) = \cos(\omega(k)\Delta t)\ \tilde{\phi}(\mathbf{k}, t) - \frac{g}{\omega(k)}\ \sin(\omega(k)\Delta t)\ \tilde{h}(\mathbf{k}, t) \quad (20)$$

Equations 19 and 20 are what is in the FFTDynamics code.

4 Convolution Form

If we want to evaluate the dynamics as a convolution, the FFT form can be used to construction the convolutions. In convolution form, there are three convolution kernels that apply as

$$h(\mathbf{x}, t) = C(\mathbf{x}) \otimes h(\mathbf{x}) + S(\mathbf{x}) \otimes \phi(\mathbf{x}) \quad (21)$$

$$\phi(\mathbf{x}, t) = C(\mathbf{x}) \otimes \phi(\mathbf{x}) - T(\mathbf{x}) \otimes h(\mathbf{x}) \quad (22)$$

where $\otimes$ denotes spatial convolution and the convolution kernels are:

$$C(\mathbf{x}) = \int \frac{d^2k}{(2\pi)^2}\ e^{i\mathbf{k}\cdot\mathbf{x}}\ \cos(\omega(k)\Delta t) \quad (23)$$

$$S(\mathbf{x}) = \int \frac{d^2k}{(2\pi)^2}\ e^{i\mathbf{k}\cdot\mathbf{x}}\ \sin(\omega(k)\Delta t)\ \frac{k}{\omega(k)} \quad (24)$$

$$T(\mathbf{x}) = \int \frac{d^2k}{(2\pi)^2}\ e^{i\mathbf{k}\cdot\mathbf{x}}\ \sin(\omega(k)\Delta t)\ \frac{g}{\omega(k)} \quad (25)$$

Explicitly in terms of integration, these convolutions are:

$$h(\mathbf{x}, t) = \int d^2y\ C(\mathbf{x} - \mathbf{y})\ h(\mathbf{y}) + S(\mathbf{x} - \mathbf{y})\ \phi(\mathbf{y}) \quad (26)$$

$$\phi(\mathbf{x}, t) = \int d^2y\ C(\mathbf{x} - \mathbf{y})\ \phi(\mathbf{y}) - T(\mathbf{x} - \mathbf{y})\ h(\mathbf{y}) \quad (27)$$

The practical implementation of convolution is as a moving window filter on the 2D gridded data for h and ϕ. Imagining these quantities have values h_{ij}, ϕ_{ij} on the rectangular grid, the time updates are

$$h_{ij}(t + \Delta t) = \sum_{k,l=-N/2}^{N/2} C_{kl}\ h_{i+k\ j+l}(t) + S_{kl}\ \phi_{i+k\ j+l}(t) \quad (28)$$

$$\phi_{ij}(t+\Delta t) \;=\; \sum_{k,l=-N/2}^{N/2} C_{kl}\ \phi_{i+k\ j+l}(t) \;-\; T_{kl}\ h_{i+k\ j+l}(t) \qquad (29)$$

Where N is the size of the square moving window, and C_{kl}, S_{kl}, and T_{kl} are the values our the kernels at discrete grid intervals. Strategies for obtaining these discrete kernels are discussed below.

Theoretically, if we take the limit that N is the size of the full simulation grid and apply periodic boundary conditions, the moving window filter produces identical results to the FFT approach, although the FFT approach is dramatically faster at it. But the moving window filter opens up many other strategies that can be applied when simple periodic boundary conditions are no longer feasible, and assuming that a small-enough window size N produces results that look good. In iWave, the smallest window size recommended was 13×13, but here a size that looks good will probably be larger, perhaps 30×30 or larger depending on the circumstances and desired visual qualities. In general, the larger the value of N, the better the quality of the propagation motion.

5 Boundary Conditions

The convolution approach breaks away from the constraint that the fields h and ϕ be periodic. That freedom also imposes a requirement that boundary conditions be specified when applying the convolution. Here are a few different strategies. Note that all of these strategies can be applied in a single simulation by choosing different ones on each boundary.

5.1 Periodic

Of course, periodic boundary conditions can still be imposed, simply by wrapping the indexing of $i+k$ and $j+l$ in the convolution sums.

5.2 Fixed Ghost Values

When the values of $i+k$ or $j+l$ extend beyond the bound of the grid, ghost values for h and ϕ can be prescribed.

5.3 Tiled Simulation Grids

If you are running independent eWave simulations on multiple grids that butt up agains each other, the simultions can be coupled via the boundary conditions. When the values of $i+k$ or $j+l$ extend beyond the bound of one grid, values for h and ϕ can be retrieved from the adjoining grid. This will naturally propagate waves and momentum between simulation grids without requiring extending each grid with gridpoints for ghost values (although ghost values is one approach to implementing the approach here).

6 Convolution Kernel Construction

The approach below is to find expressions for the convolution kernels C, S, and T. There are two methods we will look at. The first is a highly theoretical way that produces practically poor performance. The second is a brute-force method that is fast, flexible, and works.

6.1 The Impractical Theoretical Way

For the deep water case, using the symmetries of the integrands, we can define the dimensionless variable ξ as

$$\xi^2 \;=\; \frac{gt^2}{|\mathbf{x}|} \tag{30}$$

Using this variable, these convolution kernels can be written as

$$C(\mathbf{x}) = \frac{1}{|\mathbf{x}|^2}\ \mathcal{C}(\xi) \tag{31}$$

$$S(\mathbf{x}) = \frac{1}{|\mathbf{x}|^{5/2}\sqrt{g}}\ \mathcal{S}(\xi) \tag{32}$$

$$T(\mathbf{x}) = \frac{\sqrt{g}}{|\mathbf{x}|^{3/2}}\ \mathcal{T}(\xi) \tag{33}$$

where

$$\mathcal{C}(\xi) = \int_0^\infty du\ u\ J_0(u)\ \cos(\sqrt{u}\ \xi) \tag{34}$$

$$\mathcal{S}(\xi) = \int_0^\infty du\ u^{3/2}\ J_0(u)\ \sin(\sqrt{u}\ \xi) \tag{35}$$

$$\mathcal{T}(\xi) = \int_0^\infty du\ u^{1/2}\ J_0(u)\ \sin(\sqrt{u}\ \xi) \tag{36}$$

While this approach to generating the convolution kernels may prove interesting in the future, trying to apply these formula has not yet produced a practically useful approach.

6.2 The Practical Brute-Force Way

The simplest way to quickly and accurately construct C, S, and T as $N \times N$ moving window filters is to use FFTs to directly evaluate the integrals in equations 23 – 25. The procedure is as follows:

1. Create three grids, one each for C, S, and T. The dimensions of these grids should *not* be $N \times N$. They should be the dimensions of the simulation grid for h and ϕ. This is to ensure that spatial scales properly contribute to the filter values.

2. Assuming you are using an FFT package like FFTW, initialize the values in the three grids to 0, with the single gridpoint $i = 0, j = 0$ initialized to $\Delta x \Delta y$.

3. FFT the three grids to Fourier space. All of the grid points should now have identical values, and the exact value depends on (a) with FFT package you use, and (b) the number of grid points.

4. For each gridpoint in each grid, multiply the value in the gridpoint by the integrand in the corresponding equation. For the C grid, multiply by $\cos(\omega(k)\Delta t)$, the S grid by $\sin(\omega(k)\Delta t)\ \frac{k}{\omega(k)}$, and the T grid by $\sin(\omega(k)\Delta t)\ \frac{g}{\omega(k)}$.

5. Inverse FFT the three grids back to real space. Be sure to apply the normalization that the FFT package requires.

6. The three grids should now be filled with values of C, S, and T for any moving window size up the size of the full simulation grid.

7. Choose a value for N, create three $N \times N$ filter grids, and fill them with values from the larger C, S, T grids just generated.

8. You can now discard the full-sized C, S, and T grids.