Basics of surface wave simulation

L. Ridgway Scott

Departments of Computer Science and Mathematics, Computation Institute, and Institute for Biophysical Dynamics, University of Chicago
Fritz Joseph Ursell FRS (28 April 1923 — 11 May 2012)

Sir George Gabriel Stokes, 1st Baronet FRS (13 August 1819 — 1 February 1903), was senior wrangler at Cambridge in 1841. Together with Seidel, he is credited with for identifying the concept of uniform convergence as key to convergence of sums of functions.

Niels Henrik Abel (1802—1829) wrote, in a letter to a colleague in 1824, “in analysis one is largely concerned with functions that can be represented by power-series. As soon as other functions enter—and this happens rarely—then [induction] does not work any more and an infinite number of incorrect theorems arise from false conclusions”
The dispersion relation for (linear) water waves is
\[ \omega(k) = \sqrt{k \tanh(k)} \]. This can be compared to the linearized model equations: \( k = \frac{2\pi}{\lambda} \) (\( k \leq 0.4 \) iff \( \lambda \geq 15.7 \)).
The dispersion relation for water waves is

\[ \omega(k) = \sqrt{k \tanh(k)} \].

The dispersion relation for shallow water is \( \omega(k) = k \).

The dispersion relation for BBM is

\[ \omega(k) = \frac{k}{1 + \frac{1}{6}k^2} \quad (1) \]

The dispersion relation for KdV is

\[ \omega(k) = k \left( 1 - \frac{1}{6}k^2 \right) \quad (2) \]

The dispersion relation for the optimized model [BPS81] is

\[ \omega(k) = \frac{0.9898k}{1 + 0.1325k^2} \quad (3) \]
Discretizing 2nd order derivative

Defining a matrix for solving $\alpha u - u'' = f$:

```matlab
kc=0;
for k=1:nr;
    kc=kc+1; iv(kc)=k; jv(kc)=k;
    sv(kc)=alfa+(2/(dx*dx));
end
for k=2:nr;
    kc=kc+1; iv(kc)=k-1; jv(kc)=k;
    sv(kc)=-(1/(dx*dx));
end
for k=2:nr;
    kc=kc+1; iv(kc)=k; jv(kc)=k-1;
    sv(kc)=-(1/(dx*dx));
end
svm=sparse(iv,jv,sv);
```
Using the sparse matrix \texttt{fod} associated with the first-order finite difference and the sparse matrix \texttt{svm} associated with solving \( \alpha u - u'' = f \), the time-stepping for BBM looks like

\[
\text{cfl} = \frac{\text{dt}}{\text{dx}}
\]

for \( k = 1 : \text{nts} \)

\[
y_u = y_u - \text{cfl} \times \text{svm} \backslash (\text{fod}(y_u + y_u \times y_u));
\]

end

We can test this scheme using a solitary wave.
There is substantial numerical error in a first-order time-stepping scheme (dx=.1, dt=.01)
Define $\hat{y}_{i+1} = y_i + \Delta t f(y_i)$. Then

**explicit Euler:** $y_{i+1} = \hat{y}_{i+1}$

**predictor corrector:** $y_{i+1} = \frac{1}{2}(\hat{y}_{i+1} + y_i + \Delta t f(\hat{y}_{i+1}))$

<table>
<thead>
<tr>
<th></th>
<th>relative max error</th>
<th>dx</th>
<th>dt</th>
<th>cpu seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>explicit Euler</strong></td>
<td>4.3646e-02</td>
<td>1.0e-01</td>
<td>1.0e-03</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>2.2438e-02</td>
<td>5.0e-02</td>
<td>5.0e-04</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>4.4059e-03</td>
<td>2.5e-02</td>
<td>1.0e-04</td>
<td>13.0</td>
</tr>
<tr>
<td><strong>predictor corrector</strong></td>
<td>1.9110e-03</td>
<td>1.0e-01</td>
<td>1.0e-02</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>4.9510e-04</td>
<td>5.0e-02</td>
<td>5.0e-03</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>8.0939e-05</td>
<td>2.0e-02</td>
<td>2.0e-03</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>2.0380e-05</td>
<td>1.0e-02</td>
<td>1.0e-03</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>5.1133e-06</td>
<td>5.0e-03</td>
<td>5.0e-04</td>
<td>25.8</td>
</tr>
</tbody>
</table>

**Table:** Comparison of first-order (explicit Euler) and second-order (predictor-corrector) time-stepping schemes.
More efficient second-order

The predictor-corrector scheme requires twice as many function evaluations (applications of “filter” or sparse matrix operations). Once the first time-step $y_1$ has been computed via predictor-corrector, we can switch to the leap-frog (a.k.a. Verlet) scheme:

$$y_{i+1} = y_{i-1} + 2\Delta t \ f(y_i))$$

(4)

Exercise [BPS85]:

- implement the leap-frog scheme and
- compare it with predictor-corrector with respect to accuracy and efficiency.
Analysis of the difference method

We have combined two centered, second-order differences, one for $d/dx$ and the other for $d^2/dx^2$. So we expect the combination to be second order. However, there is a representation that makes this very clear and also leads the way to a remarkable fourth-order scheme.

We can write the solution to

$$u_t + f(u)_x - u_{xxt} = 0$$

as

$$u_t = -K \ast f(u)$$

where the "\ast" denotes convolution and

$$K(x) = -\text{sign}(x)\frac{1}{2}e^{-|x|}$$

(5)
Convolution representation

The equation \( u_t + f(u)_x - u_{xxt} = 0 \) can be written as a convolution \( u_t = -K \ast f(u) \) because the Green’s function for \( 1 - d^2/dx^2 \) is

\[
G(x) = \frac{1}{2} e^{-|x|}.
\]

Note that \( K(x) = G'(x) \) (6)

is positive for \( x < 0 \) and negative for \( x > 0 \).

We can now approximate \( u_t \) via the trapezoidal rule (\( h = \Delta x \))

\[
 u_t(t, ih) \approx -h \sum_{j \neq i} K((j - i)h) f(u(t, jh))
\] (7)
Convolution approximation

Using the formulation

\[ u_t(t, ih) \approx -h \sum_{j \neq i} K((j - i)h) f(u(t, jh)) \]

would involve \( \mathcal{O}(N^2) \) work, where \( N = L/h \) is the number of spatial grid points (\( L \) is the length of the computational domain). Note that we can write

\[ u_t(t, ih) \approx -h \sum_{j < i} K^-( (j - i)h ) f(u(t, jh)) \]

\[ - h \sum_{j > i} K^+ ( (j - i)h ) f(u(t, jh)) \]  \( (8) \)

where \( K^\pm = \pm \frac{1}{2} e^{-|x|} \) since \( K^-(0) + K^+(0) = 0 \).
Convolution approximation

Write the corresponding algorithm \((v \approx u_t)\) as

\[
v_i = \sum_{j \in \mathbb{Z}} k_{j-i} f_j
\]

where \(f_j = f(u(t, jh))\) and \(k_{\ell} = \frac{1}{2} h \text{sign}(\ell) e^{-|\ell|h}\) for \(\ell \neq 0, k_0 = 0\).

Define the difference operator \(D^2\) via the filter

\[
b = [0 \ 1], \quad a = [0 \ 1 \ 0] + \frac{1}{(e^h - 2 + e^{-h})} [-1 \ 2 \ -1]
\]

Then \(D^2 K^\pm = 0\) and \((D^2 k)_i = 0\) for \(|i| > 1\) and

\[
(D^2 k)_i = \begin{cases} 
\frac{1}{2} h i e^{-h} + \frac{hi(-e^{-2h}+2e^{-h})}{2(e^h-2+e^{-h})} & |i| = 1 \\
0 & i = 0
\end{cases}
\]

Note that \(i = \text{sign}(i)\) when \(|i| = 1\).
Note that

\[(D^2k)_{\pm 1} = \pm \frac{1}{2}he^{-h} + \frac{\pm h(-e^{-2h} + 2e^{-h})}{2(e^h - 2 + e^{-h})} = \pm h \frac{e^{-h}(e^h - 2 + e^{-h}) + (-e^{-2h} + 2e^{-h})}{2(e^h - 2 + e^{-h})} = \pm h \frac{(1 - 2e^{-h} + e^{-2h}) + (-e^{-2h} + 2e^{-h})}{2(e^h - 2 + e^{-h})} = \pm h \frac{\pm h}{2(e^h - 2 + e^{-h})} = \pm h \frac{\pm h}{2(h^2 + h^4/12 + \cdots)} = \pm \frac{1}{2h} (1 - h^2/12 + \cdots).\]
Convolution simplification

Therefore we can write

\[ D^2 v = D^1 f \]

where \( v = k \ast f \) as defined in (9) and

\[
(D^1 f)_i = \frac{h}{2(e^h - 2 + e^{-h})}(f_{i+1} - f_{i-1})
\]

Therefore a 2-nd order scheme is obtained via

\[ u_t \approx v \]

where \( D^2 v = D^1 f \). Note that

\[
e^h - 2 + e^{-h} = h^2 \left( 1 + \frac{h^2}{12} + \frac{h^4}{360} + \cdots \right)
\]
We can compute the integral in

\[ u_t = -K \ast f(u) \]

more accurately via a quadrature related to the Euler-Maclaurin formula:

\[
\int_0^\infty g(x) \, dx \approx \frac{1}{2} h g(0) + h \sum_{i=1}^\infty g(ih) + \frac{h^2}{12} g'(0) \tag{10}
\]

This scheme is 4th order accurate [Sco11] provided \( g \) goes to zero rapidly enough at \( \infty \). By approximating the derivative via a difference, we obtain a scheme known to Gregory.
Application to convolution

Applying this to the convolution integral we get

\[
\int_{-\infty}^{\infty} K(x)f(kh - x) \, dx
\]

\[
\approx h \sum_{i \neq 0} K(ih)f((k - i)h) + \frac{h^2}{12} [(Kf)']
\]

(11)

where \([g] = g(0+) - g(0-)\). We have

\[
[(Kf)'] = [K'f + Kf'] = [K'f] + [Kf'] = [K] f'(0) = -f'(0).
\]

We can retain a 4th order scheme by approximating \(f'(0)\) to second order:

\[
f'(0) \approx \frac{f(h) - f(-h)}{2h}
\]
Convolution simplification

Define

$$(Jf)_i = \frac{h}{24}(f(i + 1) - f(i - 1))$$

Therefore a 4th order scheme is obtained via

$$u_t \approx v^1 + v^2$$

where

$$D^2 v^1 = D^1 f$$

and

$$v^2 = Jf$$

Note that

$$J = \frac{e^h - 2 + e^{-h}}{12} D^1.$$
Using the 4th order GEM scheme together with a 4th order Runge-Kutta scheme for time stepping, we get an overall 4th order scheme. We compare this with the 2nd order approximation for the BBM equation using predictor-corrector (2nd order) time stepping for propagating a soliton of amplitude 3 for 30 time units.

<table>
<thead>
<tr>
<th></th>
<th>rel. max err.</th>
<th>dx</th>
<th>dt</th>
<th>cpu secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>predictor</td>
<td>1.8e-03</td>
<td>5.0e-02</td>
<td>5.0e-03</td>
<td>4.2</td>
</tr>
<tr>
<td>corrector</td>
<td>4.7e-04</td>
<td>2.5e-02</td>
<td>2.5e-03</td>
<td>14.9</td>
</tr>
<tr>
<td>fourth</td>
<td>1.7e-03</td>
<td>1.0e-01</td>
<td>1.0e-01</td>
<td>0.46</td>
</tr>
<tr>
<td>order</td>
<td>7.0e-05</td>
<td>1.0e-01</td>
<td>5.0e-02</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>3.7e-09</td>
<td>1.0e-02</td>
<td>5.0e-03</td>
<td>39.1</td>
</tr>
<tr>
<td></td>
<td>2.7e-10</td>
<td>5.0e-03</td>
<td>2.5e-03</td>
<td>177.0</td>
</tr>
<tr>
<td></td>
<td>9.1e-10</td>
<td>2.0e-03</td>
<td>1.0e-03</td>
<td>1056.9</td>
</tr>
</tbody>
</table>

Table: Comparison of second-order scheme and 4th-order scheme.
Solitary interaction for BBM is not exact.
Solitary interaction details

As reported in [BPS80], the solitary interaction for BBM produces an oscillatory tail of maximum amplitude -0.01674.

[BPS80] noted that the larger solitary wave increases in amplitude due to the interaction. We find that the larger wave has increased in size by $1.822 \times 10^{-05}$, only $3.037 \times 10^{-06}$ times the original amplitude.

In [BPS80], the increase in size was reported to be over two orders of magnitude larger.

The smaller solitary wave decreases in amplitude by $2.422 \times 10^{-04}$, in agreement with the results reported in [BPS80].
Nonexact interaction

The top curve is the **amplitude of the dispersive tail**, middle curve is the **decrease in the smaller wail**, and the bottom curve is the **increase in the larger wave**.
The 4-th order Runge-Kutta method

The standard Runge-Kutta method used is closely related to Simpson’s rule:

\[ w^1 = u^n - \frac{1}{2} \Delta t F(u^n) \]
\[ w^2 = u^n - \frac{1}{2} \Delta t F(w^1) \]
\[ w^3 = u^n - \Delta t F(w^2) \]
\[ u^{n+1} = u^n - \frac{\Delta t}{6} \left( F(u^n) + 2F(w^1) + 2F(w^2) + F(w^3) \right) \]  \hspace{1cm} (12)

This has four function evaluations per time step.
A fifth-order method can be obtained by combining 4-th order Adams-Bashforth (as a predictor)

\[
\hat{u}^{n+1} = u^n - \frac{\Delta t}{24} \left( 55F(u^n) \\
- 59F(u^{n-1}) + 37F(u^{n-2}) - 9F(u^{n-3}) \right) \tag{13}
\]

with 5-th order Adams-Moulton (implicit normally)

\[
u^{n+1} = u^n - \frac{\Delta t}{720} \left( 251F(\hat{u}^{n+1}) + 646F(u^n) \\
- 264F(u^{n-1}) + 106F(u^{n-2}) - 19F(u^{n-3}) \right) \tag{14}
\]

This has two function evaluations per time step.
Solitary wave propagation test for two time-stepping methods: error versus cpu time. For Runge-Kutta, $h = 2\Delta t$. For Adams-Moulton, $h/\Delta t$ takes values between 2 and 8.
Solitary wave propagation test for two time-stepping methods: error versus mesh size for $h = \Delta t$. 
Efficient schemes allow the exploration of new phenomena [HS74]
Close up of big Gaussian
The FEniCS Project automates simulation code [LMW12].

The Boussinesq equations have been implemented using FEniCS tools, in particular using Dolfin [LPT12].

FEniCS tools include GPU support [Rat10].


See “Download from Launchpad”


