

A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations

Stéphane Popinet

▶ **To cite this version:**

Stéphane Popinet. A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations. Journal of Computational Physics, 2015, 302, pp.336-358. 10.1016/j.jcp.2015.09.009 . hal-01163101v2

HAL Id: hal-01163101
https://hal.science/hal-01163101v2

Submitted on 7 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations

Stéphane Popinet[1,2] *†

7th September 2015

Abstract

The Serre–Green–Naghdi (SGN) equations, also known as the fully-nonlinear Boussinesq wave equations, accurately describe the behaviour of dispersive shoaling water waves. This article presents and validates a novel combination of methods for the numerical approximation of solutions to the SGN equations. The approach preserves the robustness of the original finite-volume Saint-Venant solver, in particular for the treatment of wetting/drying and equilibrium states. The linear system of coupled vector equations governing the dispersive SGN momentum sources is solved simply and efficiently using a generic multigrid solver. This approach generalises automatically to adaptive quadtree meshes. Adaptive mesh refinement is shown to provide orders-of-magnitude gains in speed and memory when applied to the dispersive propagation of waves during the Tohoku tsunami. The source code, test cases and examples are freely available.

1 Introduction

Large-scale water waves, such as waves at the surface of the ocean, are accurately described by the inviscid, incompressible Euler equations with a free-surface boundary condition. Unfortunately these equations are mathematically and numerically difficult to solve. Together with the practical importance of water waves in a wide range of domains, these difficulties have led to the development of many approximations of the full equations, with varying ranges of validity. While most of these approximations have historically been derived using scaling arguments on the relative importance of different terms, they can be formally related to the asymptotic analysis of the full equations in term of a few parameters: the nonlinearity parameter $\epsilon = a/h_0$, with a the characteristic amplitude of the wave and h_0 the characteristic depth, the shallowness parameter $\mu = h_0^2/L_0^2$ with L_0 the characteristic wavelength and $\beta = B/h_0$ the bathymetry nonlinearity parameter. Using combinations of assumptions on these parameters, one can formally derive the full set of historical approximations (see [24] for details).

A very general approximation is obtained when considering $\epsilon \sim 1$, $\mu \ll 1$ as first done by Serre [47] (in one dimension and with a flat bottom i.e. $\beta = 0$) and latter by Green and Naghdi [14] (in two dimensions and for $\beta \sim 1$). This approximation is fully nonlinear (neither the wave amplitude nor the bathymetry variations are assumed small) but weakly dispersive (i.e. the dispersion relation is valid only in the limit $\mu \ll 1$). It is thus closely related to the classical Saint-Venant system. Indeed, the Serre–Green–Naghdi (SGN) equations can be seen as the expansion to order $O(\mu^2)$ of the (non-dispersive) Saint-Venant system (accurate to $O(\mu)$).

The SGN system is thus relevant when waves are both nonlinear and (weakly) dispersive which happens in particular for wind-driven waves close to shore. This raises another difficulty

* *Affiliation:* [1]Sorbonne Universités, UPMC Univ Paris 06, UMR 7190, ; Institut Jean le Rond d'Alembert, Paris, France; [2]CNRS, UMR 7190, Institut Jean le Rond d'Alembert, Paris, France

† *Email:* stephane.popinet@upmc.fr

regarding the theoretical and numerical treatment of the "contact line" separating the wet and dry topographies. The treatment of wetting and drying has been a long-standing problem, for the Saint-Venant equations, but has largely been solved in the past ten years thanks to theoretical and numerical advances [1, 29, 26]. The resulting schemes have the advantage of providing remarkably robust and accurate solutions, including a realistic description of dissipation during wave breaking [6]. The extension of these properties to numerical schemes for the SGN equations is a topic of active research [33, 4, 48, 25].

The numerical approximation of solutions to the SGN equations requires the inversion of a linear second-order differential operator. This can be interpreted as the non-local coupling induced by a non-hydrostatic pressure field and reflects the incompressibility condition of the original system. This coupling also controls the dispersive properties of the equations. This is thus a fundamental difference from the hyperbolic (i.e. local) behaviour of the non-dispersive Saint-Venant system. In practice the overall performance of a solver for the SGN equations (and other "non-hydrostatic" formulations) then rests on the efficient solution of the large linear systems resulting from the discretisation of this differential operator.

In many practical applications wave fields are very inhomogeneous. Wave amplitudes and frequencies are highly variable. This can be due to the broad scale distribution of forcing terms (wind for example), interaction with complex bathymetries and non-linear interactions between waves. In this context variable-mesh discretisation techniques are appealing in order to optimise the accuracy of numerical approximations with respect to computational cost. Adaptive mesh refinement methods have proven very efficient in particular for solving the Saint-Venant equations for tsunami propagation [39, 27, 40] or for spectral ocean wave models [45]. Extensions of these approaches to dispersive wave models are rare however.

The work presented in this article draws on previous studies but makes several novel contributions. Bonneton et al. [4] use Strang splitting to solve the one-dimensional SGN equations. Lannes and Marche [25] recently extended the scheme of [4] to two dimensions. The critical ingredient they identify is the efficient inversion of the 2D differential SGN operator. This is achieved by deriving a new set of equations, asymptotically equivalent to the original SGN system, but only involving a constant-coefficient scalar differential operator which can be efficiently inverted using a tridiagonal solver (on regular Cartesian meshes). Efficiency thus relies on the specific symmetries of regular (non-adaptive) Cartesian meshes. In this article, we show that the original SGN differential operator can be inverted efficiently using a multigrid solver. Provided a suitable programming framework is used, the implementation of this scheme is simple and can be generalised to adaptive quadtree meshes. This draws on previous work by the author [37] but is the first generalisation of the approach to coupled vector unknowns.

The implementation of the solver is described in detail in Section 3. Applications in Section 4 validate the approach in particular with respect to dispersive properties and robustness of the treatment of wetting/drying and wave breaking. A realistic, large-scale application to the dispersive propagation of waves during the Tohoku tsunami is then presented. Absolute and relative performances are discussed with an emphasis on the objective estimation of the computational gain obtained with adaptivity. The source code implementation as well as most of the test cases are freely available under the Free Software GPL licence.

2 Model equations

In this section we define the basic notations and summarise the formulation and principal properties of the Serre–Green–Naghdi equations. Note that the detailed derivation of this system is quite complex and is beyond the scope of this paper. For further details and discussions of the SGN equations, we refer the reader to [24].

The SGN equations can be written in integral form as a generic system of conservation laws

with source terms of the form

$$\partial_t \int_\Omega \boldsymbol{q} d\Omega = \int_{\partial\Omega} \boldsymbol{f}(\boldsymbol{q}) \cdot \boldsymbol{n} d\partial\Omega + \int_\Omega \boldsymbol{S} d\Omega \tag{1}$$

where Ω is a subset of two-dimensional space, $\partial\Omega$ its boundary and $\boldsymbol{n}$ the unit normal vector on this boundary. For the Saint-Venant equations, the state variable $\boldsymbol{q}$ (resp. fluxes $\boldsymbol{f}$) are the mass and momentum (resp. mass and momentum fluxes) which can be written

$$\boldsymbol{q} = \begin{pmatrix} h \\ hu_x \\ hu_y \end{pmatrix}, \boldsymbol{f}(\boldsymbol{q}) = \begin{pmatrix} hu_x & hu_y \\ hu_x^2 + \frac{1}{2}gh^2 & hu_x u_y \\ hu_x u_y & hu_y^2 + \frac{1}{2}gh^2 \end{pmatrix} \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, h the water depth and g the acceleration of gravity. Following [4] the source term $\boldsymbol{S}$ can be written

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ -hg\partial_x z_b + h\left(\frac{g}{\alpha}\partial_x \eta - D_x\right) \\ -hg\partial_y z_b + h\left(\frac{g}{\alpha}\partial_y \eta - D_y\right) \end{pmatrix} \tag{3}$$

where z_b is the height of topography, $\eta = z_b + h$ and α is a parameter used to tune the dispersion relation ($\alpha = 1$ in the original SGN formulation). Note that in the absence of e.g. precipitation or evaporation there is no source term in the mass equation, either for the Saint-Venant or SGN equations (this is not the case for other dispersive approximations such as the linearised Boussinesq equations or the equation set by Nwogu [36]). For the momentum equations, the first source term $-hg\nabla z_b$ is the standard forcing by topography of the Saint-Venant system, while the second term represents the dispersive contributions of the SGN approximation. If this term is removed the system reduces to the classical Saint-Venant formulation.

The vector field $\boldsymbol{D}$ verifies

$$h\left(\boldsymbol{I} + \alpha\mathcal{T}\right)(\boldsymbol{D}) = \frac{g}{\alpha}\nabla\eta + \mathcal{Q}_1(\boldsymbol{u}) \tag{4}$$

with $\boldsymbol{I}$ the identity matrix and $\mathcal{T}$ and $\mathcal{Q}_1$ second-order linear differential operators. As pointed out by [4] this formulation is beneficial from a numerical point of view since $\mathcal{T}$ and $\mathcal{Q}_1$ only depend on second spatial derivatives of $\boldsymbol{D}$ and $\boldsymbol{u}$ (whereas other dispersive formulations often involve third derivatives). On the other hand the detailed expressions for $\mathcal{T}$ and $\mathcal{Q}_1$ are rather complicated. Following the notations in [4] we first define the operators

$$\begin{aligned} \mathcal{R}_1[h, z_b]w &= -\tfrac{1}{3h}\nabla(h^3 w) - \tfrac{h}{2}w\nabla z_b \\ &= -h\left[\tfrac{h}{3}\nabla w + w\left(\nabla h + \tfrac{1}{2}\nabla z_b\right)\right] \\ \mathcal{R}_2[h, z_b]w &= \tfrac{1}{2h}\nabla(h^2 w) + w\nabla z_b \\ &= \tfrac{h}{2}\nabla w + w\nabla\eta \end{aligned}$$

The general form for $\mathcal{T}$ is then

$$\mathcal{T}[h, z_b]w = \mathcal{R}_1[h, z_b]\left(\nabla \cdot w\right) + \mathcal{R}_2[h, z_b]\left(\nabla z_b \cdot w\right)$$

Finally, the general form for $\mathcal{Q}_1$ is

$$\mathcal{Q}_1[h, z_b]\boldsymbol{u} = -2\mathcal{R}_1[h, z_b]\left(\partial_x \boldsymbol{u} \cdot \partial_y \boldsymbol{u}^\perp + (\nabla \cdot \boldsymbol{u})^2\right) + \mathcal{R}_2[h, z_b]\left(\boldsymbol{u} \cdot (\boldsymbol{u} \cdot \nabla)\nabla z_b\right)$$

3 Numerical method

As pointed out in the previous section, the SGN model can be seen as a (complicated) momentum source term added to the Saint-Venant system. It thus seems natural to derive a numerical scheme for the SGN system by adding this source term to an existing Saint-Venant solver. This is the approach we have taken. While [4] proposed to do this using Strang splitting, we note that adding arbitrary source terms to the Saint-Venant system (or other systems of time-dependent partial differential equations) can often be done simply by including this source term in the appropriate stages of the time integration scheme (for example the updating stages of Runge–Kutta schemes) [17, 48, 32, 10, 28, 7, 8, 9]. This approach also has the benefit of preserving the order of accuracy of the overall time-integration scheme and/or being able to easily change this order. Furthermore well-designed Saint-Venant solvers should include facilities for adding additional source terms, so that the approach outlined here should be easily generalisable.

3.1 Saint-Venant solver

Shock-capturing finite-volume methods are well-known for their ability to provide robust solutions for hyperbolic systems of conservation laws such as the Saint-Venant system given by (1) and (2) [1, 12]. We use the numerical scheme described in detail in [39]. The initial version of the scheme was implemented within the Gerris platform [38]. The results presented in this article were obtained using the implementation of the same scheme on the Basilisk platform [41]. Both implementations are open-source and we invite the reader to consult the Basilisk web site for a documented description of the source code of the Saint-Venant solver [41]. As pointed out before, the source-term decoupling between the SGN and Saint-Venant systems means that the approach presented here should work independently of the details of the Saint-Venant solver implementation. Consequently we only summarise the essential properties of the Saint-Venant solver and refer the reader to previous publications and source codes for details. These properties are:

- Second-order, slope-limited, MUSCL unsplit flux integration.
- Second-order predictor-corrector time-integration.
- HLLC approximate Riemann solver takes into account dry states.
- Positivity-preserving, well-balanced scheme of [1] uses the "hydrostatic reconstruction of bathymetry" technique.

The treatment of wetting/drying and well-balancing (i.e. preservation of lake-at-rest equilibrium) have been longstanding issues in numerical models of the Saint-Venant system. Much progress has been made in the past ten years however and the numerical method summarised above has been shown, for different variants and implementations, to lead to very robust and accurate results for a range of applications, including complex scenarios such as tsunami propagation and inundation [40]. The implementations in Gerris and Basilisk have been validated extensively both for academic and more realistic applications [38, 41].

3.2 Serre–Green–Naghdi source term

The main difficulty in the implementation of a numerical scheme for the SGN equations is the solution of the second-order differential equation (4) to obtain vector field $\boldsymbol{D}$. Expanding the definition of $\mathcal{T}$we can rewrite (4) as

$$\alpha h\mathcal{R}_1\left(\nabla\cdot\boldsymbol{D}\right)+\alpha h\mathcal{R}_2\left(\nabla z_b\cdot\boldsymbol{D}\right)+h\boldsymbol{D} \quad = \quad \frac{g}{\alpha}\nabla\eta+\mathcal{Q}_1(\boldsymbol{u})$$

and expanding $\mathcal{R}_1$ and $\mathcal{R}_2$ then gives

$$\begin{aligned} -\alpha h^2 \left[\frac{h}{3}\nabla\left(\nabla\cdot\boldsymbol{D}\right) + \left(\nabla\cdot\boldsymbol{D}\right)\left(\nabla h + \frac{1}{2}\nabla z_b\right)\right] + \\ \alpha h\left[\frac{h}{2}\nabla\left(\nabla z_b\cdot\boldsymbol{D}\right) + \left(\nabla z_b\cdot\boldsymbol{D}\right)\nabla\eta\right] + h\boldsymbol{D} \quad = \quad \frac{g}{\alpha}\nabla\eta + \mathcal{Q}_1(\boldsymbol{u}) \end{aligned} \tag{5}$$

A choice of spatial discretisation for the differential operators – in our case simple second-order finite differences on a regular Cartesian grid – then leads to a large, coupled linear system of equations for the components (D_x, D_y) of the momentum source term $\boldsymbol{D}$. The inversion of this linear system needs to be done at each stage of the time integration scheme (for example the two stages of the predictor-corrector scheme). The right-hand-side of (5) and the remaining dispersive source term ($h\frac{g}{\alpha}\nabla\eta$ in (3)) are computed explicitly from the estimates of the state variables at each stage.

An efficient method for the solution of the linear system is thus essential to the success of the scheme. Lannes and Marche [25] proposed to use a different, but asymptotically equivalent, form of the SGN equations which leads, with an appropriate choice for the spatial discretisation, to decoupled tridiagonal systems for each component of $\boldsymbol{D}$. Optimal solvers for tridiagonal systems are then easily implemented and lead to an efficient overall solution algorithm. Note that this is also the approach used successfully by [52] who use a different (but equivalent [24]) formulation of the SGN equations. A limitation of this approach is that it relies on the specific symmetries of Cartesian grid discretisations and cannot be easily generalised to other grid structures such as the adaptive quadtree grids which we would like to use.

A general solution would be to use a generic solver for linear systems. For example Krylov-subspace-type iterative solvers [46]. In our experience however using such solvers for example to solve quadtree-adaptive elliptic problems is not very efficient. This is due in particular to the fact that, when using adaptive grids, the size and structure of the linear system (i.e. the left-hand-side matrix and its coefficients) change as the grid evolves. This precludes optimisations often performed by linear solver packages, such as pre-computation and re-use of expensive subproblems etc... As noted by [25] this is also the case when the coefficients of the left-hand-side operator are time-dependent, as in (4).

3.3 Multigrid solver

Multigrid solvers are well-known for being efficient for elliptic or parabolic problems such as the Poisson–Helmholtz equation

$$\nabla\cdot\left(\nabla a\right) + \lambda a = b$$

One can in particular show that they are asymptotically optimal, requiring only $O(N)$ operations to solve the system above discretised with N grid points [20]. In [37] and other papers, we also showed that geometric multigrid can be naturally combined with quadtrees (in 2D and octrees in 3D) which directly provide the necessary "multigrid" hierarchy. This leads to efficient, adaptive algorithms for problems where the cost is dominated by the solution of elliptic and parabolic problems (e.g. incompressible Navier–Stokes or Stokes flows, reaction–diffusion equations etc...)

In this context and given the proven advantages of the adaptive quadtree scheme for the hyperbolic part of the problem (the Saint-Venant equations [39]) it was thus natural to wonder whether the second-order differential operator of the SGN source term could also be inverted efficiently with a multigrid solver (note that the leading-order operator in (5) is $\nabla\left(\nabla\cdot\right)$ rather than the Laplacian $\nabla\cdot\left(\nabla\right)$).

We now recall the overall solution procedure for a generic linear problem

$$\mathcal{L}(a) = b.$$

In many cases, for example when solving time-dependent problems, a good initial guess $\tilde{a} = a - \mathrm{da}$ is available (with da an unknown error or correction). It is thus usually more efficient to solve the equivalent problem

$$\mathcal{L}(\mathrm{da}) = b - \mathcal{L}(\tilde{a}) = \mathrm{res} \tag{6}$$

where res is called the *residual*, and then obtain the solution by adding da to $\tilde{a}$. In practice, one does not need to solve (6) exactly; an improved guess is sufficient as the procedure can be repeated until convergence. This can be summarized by the following steps:

1. Given an initial guess $\tilde{a}$ compute $\mathrm{res} = b - \mathcal{L}(\tilde{a})$.
2. If $\| \mathrm{res} \| < \epsilon$, $\tilde{a}$ is good enough, stop.
3. Otherwise solve $\mathcal{L}(\mathrm{da}) \simeq \mathrm{res}$.
4. Add da to $\tilde{a}$ and go to step 1.

The multigrid solver itself is used to perform step 3. The basic principle of geometric multigrid is to decompose the error on successively coarser grids. Each characteristic error wavelength is then represented using only a few grid points i.e. the shorter waves are represented on the finer grids and the longer waves on the coarser grids. The error can then be reduced efficiently by using only a few iterations of simple relaxation techniques such as Jacobi or Gauss-Seidel on each of the grids. Using the notations in Basilisk (see [42] for details) this can be implemented as in Function 1.

Function 1

```
void mg_cycle (field a, field res,
               void (* relax) (field da, field res, int depth))
{
  /* restrict residual */
  for (int l = depth() - 1; l <= 0; l--)
    foreach_level (l)
      restriction (res);
  /* multigrid traversal */
  field da[];
  for (int l = 0; l <= depth(); l++) {
    if (l == 0)
      /* initial guess on coarsest level */
      foreach_level (l)
        da[] = 0.
    else
      /* prolongation from coarser level */
      foreach_level (l)
        prolongation (da);
    boundary (da, l);
    /* relaxation */
    for (int i = 0; i < 4; i++) {
      relax (da, res, l);
      boundary (da, l);
    }
  }
  /* correction */
  foreach()
    a[] += da[];
}
```

Besides $\tilde{a}$ and res, the function takes as argument the problem-dependent relaxation function `relax()`. The `foreach_level(l)` function iterates through the grid at level `l`. The coarsest grid is at level zero and the finest at level `depth()`. The residual is initially defined only on

the finest grid (at level `depth()`). In a first step this residual is *restricted* to all the coarser grids. The second step is the multigrid traversal itself. On the coarsest grid (level zero), the initial guess for `da` is set to zero, otherwise the initial guess is obtained by *prolongation* from the immediately coarser grid (i.e. at level `l-1`). The error is then reduced by applying four iterations of the problem-dependent relaxation function. Finally, the improved guess on the finest grid is obtained by adding the correction to the initial guess (`foreach()` iterates only on the finest grid).

For the moment we have not made any assumptions regarding the details of the spatial discretisation and/or the way successively coarser grids are logically connected. The details of the *restriction* and *prolongation* operations depend on the discretisation however. If we assume a hierarchy of regular Cartesian grids, doubling in resolution between successive levels, the `restriction()` and `prolongation()` functions can be implemented as in Function 2.

Function 2

```
void restriction (field v)
{
  v[] = (fine(v,0,0) + fine(v,1,0) + fine(v,0,1) + fine(v,1,1))/4.;
}

void prolongation (field v)
{
  v[] = (9.*coarse(v,0,0) +
         3.*(coarse(v,child.x,0) + coarse(v,0,child.y)) +
         coarse(v,child.x,child.y))/16.;
}
```

For a given grid point, the `restriction()` function defines the value of the field as the average of the four finer *children* of this grid point. Conversely the `prolongation()` function defines the value of the field as the bilinear interpolation from the four coarser *parents* of the grid point (`child.(x|y)` are the coordinates of the grid point relative to its parents i.e. ± 1).

3.3.1 Application to Poisson equation

To apply the functions to a specific problem, we need to define the `residual()` and `relax()` functions. As a simpler introduction, we will consider the Poisson equation

$$\nabla \cdot (\nabla a) = b$$

with a an unknown, and b a given, scalar fields. The corresponding residual and relaxation functions can be implemented as in Function 3.

Function 3

```
void residual_poisson (scalar a, scalar b, scalar res)
{
  foreach()
    res[] = b[] - (a[1,0] + a[-1,0] + a[0,1] + a[0,-1] - 4.*a[])/sq(Delta);
}

void relax_poisson (scalar a, scalar b, int l)
{
  foreach_level (l)
    a[] = (a[1,0] + a[-1,0] + a[0,1] + a[0,-1] - sq(Delta)*b[])/4.;
}
```

The residual needs only to be computed on the finest grid (hence the `foreach()` iterator). We recognise on the right-hand-side the expression for the standard discrete 5-points Laplacian operator (with `Delta` the mesh size). The relaxation operator needs to be applicable on grids at any level `l`. It is just the Jacobi operator corresponding to the residual. Together with the iteration loop described at the start of section 3.3, Functions 1, 2 and 3 define the entire solution procedure for the Poisson problem. Note also that the notation we have used is actual Basilisk code. For a more detailed description of the generic multigrid solver, including applications to more general Poisson–Helmholtz problems, we invite the reader to consult the documented Basilisk code [44].

3.3.2 Inverting the SGN operator

The second-order differential operator in (5) is obviously more complicated than a Laplacian, however it is still relatively simple to write the corresponding discrete expression for the residual. We first derive the explicit expression for the x-component of the dispersive source term $\boldsymbol{D}$. From (5) we get

$$\begin{aligned} -\frac{\alpha}{3}\partial_x(h^3\partial_x D_x) + h\left[\alpha\left(\partial_x\eta\partial_x z_b + \frac{h}{2}\partial_x^2 z_b\right) + 1\right] D_x + \\ \alpha h\left[\left(\frac{h}{2}\partial_{xy}^2 z_b + \partial_x\eta\partial_y z_b\right) D_y + \frac{h}{2}\partial_y z_b\partial_x D_y \right. \\ \left. -\frac{h^2}{3}\partial_{xy}^2 D_y - h\partial_y D_y\left(\partial_x h + \frac{1}{2}\partial_x z_b\right)\right] \quad = \quad b_x \end{aligned} \tag{7}$$

with

$$\boldsymbol{b} = \frac{g}{\alpha}\nabla\eta + \mathcal{Q}_1(\boldsymbol{u})$$

The explicit expression for the y-component is obtained by rotation of the indices. We now define the discrete schemes for approximating the first- and second-derivatives of a generic field.

Function 4

```
#define dx(s)   ((s[1,0] - s[-1,0])/(2.*Delta))
#define dy(s)   ((s[0,1] - s[0,-1])/(2.*Delta))
#define d2x(s)  ((s[1,0] + s[-1,0] - 2.*s[])/sq(Delta))
#define d2y(s)  ((s[0,1] + s[0,-1] - 2.*s[])/sq(Delta))
#define d2xy(s) ((s[1,1] - s[1,-1] - s[-1,1] + s[-1,-1])/sq(2.*Delta))
```

It is then relatively simple to write the residual function corresponding to (7)

Function 5

```
void residual_sgn (vector D, vector b, vector res)
{
  foreach()
    foreach_dimension() {
      double hc = h[], dxh = dx(h), dxzb = dx(zb), dxeta = dxzb + dxh;
      double hl3 = (hc + h[-1,0])/2.; hl3 = cube(hl3);
      double hr3 = (hc + h[1,0])/2.;  hr3 = cube(hr3);
      res.x[] = b.x[] -
        (-alpha/3.*(hr3*D.x[1,0] + hl3*D.x[-1,0] - (hr3 + hl3)*D.x[])/sq(Delta) +
         hc*(alpha*(dxeta*dxzb + hc/2.*d2x(zb)) + 1.)*D.x[] +
         alpha*hc*((hc/2.*d2xy(zb) + dxeta*dy(zb))*D.y[] +
                   hc/2.*dy(zb)*dx(D.y) - sq(hc)/3.*d2xy(D.y)
                   - hc*dy(D.y)*(dxh + dxzb/2.)));
    }
}
```

where the `foreach_dimension()` operator automatically generates the code for the y-component by rotation of the indices (i.e. swaps `.x` with `.y` and `[-1,0]` with `[0,-1]` etc...). The corresponding Jacobi relaxation function `relax_sgn()` is obtained easily by copying the function above and inverting manually for `D.x[]`. Note that the residual and relaxation functions operate on vector fields (rather than scalar fields for the Poisson problem above). These two functions combined with the generic multigrid Function 1 are all that is required to invert the SGN operator. The complete documented source code is available on the Basilisk web site [43].

3.3.3 Convergence criterion

A convergence criterion is required to stop the multigrid iterations. The residual defined by (5) has dimensions of depth times acceleration. For all the examples discussed in this article we chose to stop the multigrid iterations when

$$\| \operatorname{res} \|_\infty < g/1000,$$

where g is the acceleration of gravity. In most cases only a few multigrid iterations are necessary to reach convergence.

3.4 Wave breaking and wetting/drying

As discussed in the introduction, the validity of the SGN equations is questionable when waves become steep. In reality such waves are unstable and break in a fully three-dimensional turbulent motion [34]. Remarkably, the "shocks" or hydraulic jumps occuring in the Saint-Venant equations, which are in practice regularised by the numerical dissipation of shock-capturing schemes, seem to give a reasonable description of the wave breaking process [6]. The idea is then to rely on the pure Saint-Venant system when waves are close to breaking. This is achieved simply by turning off the SGN dispersive term in the appropriate areas. The exercise is then to find a realistic criterion to detect "breaking waves". We use a simple criterion based on the slope of the free surface $\|\nabla\eta\|$. If it is larger than a threshold (typically one) we turn off dispersive terms. Although more sophisticated schemes could also be used [50], we found that this simple criterion works quite well for the various examples we tried.

The robustness of the scheme for the treatment of wetting and drying relies on the "hydrostatic reconstruction" technique of [1] as implemented in the pure Saint-Venant solver. In practice, we have found that turning off the dispersive term in the vicinity of the "contact line" is necessary to obtain a robust scheme. This is done by only including the dispersive terms when all grid points in a 3×3 neighbourhood are "wet" (i.e. if the water depth h is larger than a threshold set to a value comparable to machine round-off).

3.5 Quadtree adaptivity

Quadtrees are an efficient and simple data structure which allows fine tuning of the local spatial resolution [37]. Furthermore, the (square) elementary control volumes in a quadtree are the same as for pure Cartesian meshes, so that schemes implemented on Cartesian meshes can in principle be easily generalised to quadtrees. To realise this in practice, the underlying programming environment needs to provide the adequate abstractions valid both for pure Cartesian grids and quadtrees. This is the goal of the Basilisk framework we used in this article. The algorithms presented earlier introduce two basic abstractions: *grid traversal* (i.e. the `foreach`, `foreach_level` operators) and *local stencils* (i.e. the `h[-1,0]` etc... indexing). While the generality of the traversal operators is straightforward, the assumption of consistent, regular, local Cartesian stencils seems restricted to pure Cartesian grids. To generalise regular stencils, we need to introduce a third abstraction: *boundary conditions*.

Even in the case of pure Cartesian grids, it is clear that assuming consistent, regular stencils requires a special treatment close to the boundaries of the grid. A very classic way to ensure consistency is simply to extend the grid beyond the boundary and fill the resulting "ghost points" with values corresponding to some discrete approximation of the desired boundary conditions. For example, one could impose symmetry conditions on the boundary simply by filling the ghost points symmetrically in the direction normal to the boundary. In the case of Cartesian grids, this is exactly what the `boundary()` calls do in Function 1 for example, thus ensuring that subsequent calls to the `relax()` function always have access to consistent stencils both in the interior and close to boundaries.

When using quadtrees, the spatial resolution is variable and configurations such as illustated in Figure 1 will occur. The black dots are locations where the discrete values are computed

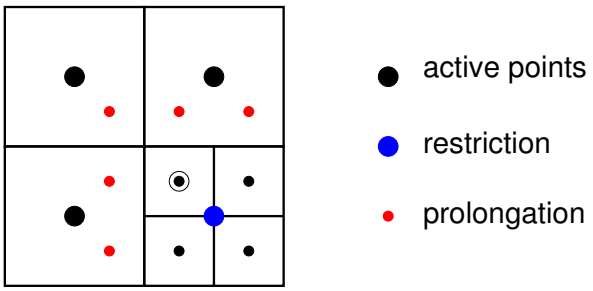


Figure 1: Example of local variable-resolution stencil on a quadtree.

(i.e. locations traversed by the `foreach` operators). We now consider the fine-grid 3×3 stencil centered on the circle. The black dots at the same level (i.e. `[1,0]`, `[1,-1]` and `[0,-1]`) are defined, however the red dots (`[-1,1]`, `[0,1]`, `[1,1]`, `[-1,0]` and `[-1,-1]`) are not. To ensure a consistent regular stencil, we need to fill in the red dots. This is obviously similar to filling ghost points near grid boundaries, however we are now dealing with *resolution boundaries* which occur in the interior of the quadtree grid. A simple way to do this is to use the *prolongation* Function 2 above which uses bilinear interpolation from the coarser level (the larger dots on the figure). We see however that this requires the value of the field at the large blue dot, which is not computed explicitly. The solution is to use the *restriction* function 2 above, before applying the prolongation. If we generalise to multiple grid levels, we get the boundary condition Function 6

Function 6

```
void boundary (field f, int level)
{
  for (int l = level - 1; l <= 0; l--)
    foreach_level (l)
      restriction (f);
  for (int l = 0; l <= level; l++)
    foreach_halo (l)
      prolongation (f);
}
```

In a first pass, the values at the blue dots for all levels are defined by applying the restriction operator from the finest to the coarsest grid. Once this is done, the ghost values of all the red dots can be obtained by application of the prolongation operator from the coarsest to the finest grid. We have introduced a new traversal operator `foreach_halo()`, specific to the quadtree grid, which only traverses the ghost or *halo* cells between level `l` and `l-1`.

Applying this boundary condition is thus sufficient to ensure stencil consistency. All the functions described above will then work without modification on variable-resolution quadtrees.

Furthermore the implementation of Function 6 itself relies almost entirely (with the exception of the `foreach_halo()` traversal operator) on pure Cartesian operations.

Note that for the sake of simplicity, we have restricted the example in Figure 1 to the case of a 3×3 stencil. In practice *halos* between levels of refinement always include the four points sharing a common parent (i.e. *siblings*), so that the layout of Figure 1 looks more like that in Figure 2. The motivation for this choice is that systematically defining the four siblings values

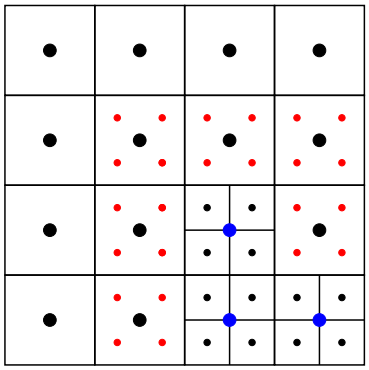


Figure 2: Example of local variable resolution with the "natural" 5×5 stencil.

is simpler (since one does not need to decide which sibling to fill based on the local layout) and only marginally more computationally expensive than only filling (typically) two values. The result of this choice is that Function 6 above will guarantee consistent (at least) 5×5 rather than 3×3 stencils. Although the algorithms described in this article only make use of 3×3 stencils and are consequently only second-order accurate, the availability of 5×5 stencils opens the door to simple, higher-order (typically fourth-order), quadtree-adaptive extensions of these schemes. Implementing such schemes would only require modifications of Functions 2, 4 and 5 (as well as the higher-order extension of the underlying Saint-Venant scheme).

3.5.1 Adaptation and stability

The definition of boundary conditions between levels of refinement also provides a natural adaptation criterion. If we consider the example of Figure 1, the value of the central point in the stencil (circle) could also be obtained by prolongation from the coarser grid. This is indeed what would be done if the local resolution was lowered by one level: in this case the coarse blue dot would become an active point and the finer black dots would become halo points, filled by bilinear interpolation (assuming finer neighbouring stencils required these values).

How can we estimate the error which would be caused by this *coarsening* of the grid? A simple measure is the difference between the actual value at this point (i.e. black dot in Figure 1) and its prolongation from the coarser grid at the same location (computed using Function 2). This can also be seen as an estimate of the local (bilinear) *interpolation error*. Given an error threshold ϵ, *local refinement* happens whenever the local estimate of the interpolation error is larger than ϵ. Conversely, *local coarsening* happens if the error for all siblings is smaller than ϵ/c. The four siblings are then removed and replaced by the parent grid point. Unless otherwise specified, the constant c is 1.5.

When coarsening happens the values of the new active points are simply those obtained by restriction when applying boundary conditions i.e. nothing further needs to be done. In the case of refinement, the values of the new grid points are not necessarily defined (in the case where new grid points were not already halo points). The new values are obtained by interpolation from the coarser grid. It is often desirable to preserve the discrete conservation properties of the numerical scheme. For example, the flux-form Saint-Venant solver we use ensures the discrete

conservation of the local water depth (and thus of the total water volume). The restriction operation in Function 2 trivially ensures conservation of the restricted field. This is not the case however for the bilinear interpolation used for prolongation. To ensure conservation, we use linear interpolation for conserved fields (momentum $h\boldsymbol{u}$ and depth h) i.e. we replace the prolongation in Function 2 with

Function 7

```
void prolongation_linear (field v, vector g)
{
  v[] = coarse(v,0,0) + Delta/2.*(child.x*coarse(g.x,0,0) + child.y*coarse(g.y,0,0));
}
```

where `g` is the slope-limited gradient of `v`.

The stability of the overall scheme is controlled by the time-explicit treatment of the Saint-Venant system of conservation equations. The timestep must respect the CFL condition based on the local value of mesh size Δ and gravity wave speed $\sqrt{gh}$. The inversion of the SGN differential operator guarantees stability of the dispersive terms irrespective of the timestep. In the absence of space-dependent diffusive processes (e.g. horizontal "viscous" terms), the Saint-Venant equations do not define a minimum intrinsic spatial scale. In practice, solutions can develop hydraulic jumps which are true discontinuities (i.e. with a vanishing spatial scale). The addition of dispersion will introduce a characteristic "dispersive" spatial length scale which should lead to a more favourable scaling of spatial resolution. This is verified in the applications below.

3.6 Notes on performance

Performance, ideally measured as the wall-clock runtime for a given accuracy, is a key parameter in the development and evaluation of numerical methods. Indeed, if performance was not a central issue, one could use simple and robust methods with excellent theoretical properties, such as first-order Godunov schemes and direct linear solvers, and this article (together with thousands of others) would not make any sense. Yet few articles actually address this question directly and it is often difficult to find even rough indications of the absolute performance of published numerical methods.

Paradoxically, this is particularly true for articles describing adaptive mesh refinement (AMR) techniques (whose *raison d'être* is performance) where efficiency is often discussed by comparing the runtimes of the AMR code when run non-adaptively to the same code with adaptation. The issue here is that implementing adaptivity often leads to compromises which can affect very badly the non-adaptive performance of the implementation. This is particularly true for methods using Cartesian-like discretisations for which the reference implementations on pure Cartesian grids can be extremely fast on modern processors. The authors are often aware of this issue but may feel their approach is justified by the extra work required when implementing the same algorithm twice: once on an adaptive mesh and once on an optimised Cartesian mesh (with results which would most likely not be favourable to the adaptive method).

The Basilisk framework we have briefly introduced above provides a solution for Cartesian and quadtree-adaptive methods. It allows to implement Cartesian discretisations at a relatively low-level (stencils), thus preserving the full conceptual flexibility of finite-differences or finite-volumes schemes, while separating the details of the grid implementation itself. Together with the simple algorithms presented above, this allows automatic generalisation of Cartesian schemes to adaptive quadtree grids. This decoupling between the low-level implementation (e.g. memory layout, traversal strategies) of the different types of grids (single-level Cartesian, multi-level grids, quadtrees) from the numerical scheme itself also permits the independent development of optimised low-level strategies. For example, the implementation of optimal storage and traversal

strategies is highly non-trivial, even for simple Cartesian grids, and is itself a whole sub-discipline of computer science [16, 49].

Although we have only started to explore these possibilities, the current Basilisk framework already provides efficient Cartesian grid and quadtree implementations. We will give absolute figures in the next section, but as an indication the numerical schemes implementing in Basilisk run roughly eight times faster than the same schemes implemented in Gerris.

4 Applications

In this section we present a short selection of applications of the numerical scheme, starting with a few simple test cases and ending with a realistic simulation of the Tohuku tsunami of 2011. Note that the source codes and results for a wider range of validation cases including: solitary waves, wave runup on beaches, waves overtopping a seawall, sinusoidal wave propagation over a bar, etc. are available on the Basilisk web site [43].

4.1 Simplified tsunami model

Madsen et al [30] used a simplified setup to study the long-distance evolution of tsunami-like perturbations. A rectangular initial perturbation of width $2b$ and height a is added to an ocean of constant depth h_0. The perturbation is meant to imitate the initial seismic uplift which is the main cause of tsunamis. The problem is fully characterised by the nonlinearity parameter $\epsilon = a/h_0$ and the dispersion parameter $\mu = (h_0/b)^2$. We reproduce the case of Figure 1 of [30] with $a/h_0 = 0.1$ and $b/h_0 = 12.2$. Note that while b/h_0 is realistic, a/h_0 is much larger than for a realistic tsunami. The model is discretised using an adaptive "bitree" grid (the one-dimensional equivalent of a quadtree). Adaptation is performed using a threshold of $10^{-6}h_0$ on the estimated interpolation error on the free-surface elevation η. The maximum level of refinement is set to 15 and the domain length is $4000h_0$. These choices lead to a grid-independent solution for the results presented in Figure 3. Both the Serre–Green–Naghdi and the Saint-Venant solutions are represented. Dispersive effects are strong and lead to large differences between the two solutions. Non-linearities are also important and cause a significant evolution of the shape of the waves with time. Also, both sets of waves move at a speed significantly larger than $\sqrt{gh_0}$. For the dispersive model, as expected [13, 30], the initial perturbation breaks up into a train (of two) solitons after a sufficiently long time. As discussed by [30] the timescale on which this occurs is controlled mainly by the nonlinearity parameter ϵ. For the much smaller nonlinearities (in the deep ocean) characteristic of realistic tsunamis, this timescale becomes extremely long so that deep-water tsunamis never have the time to reach this (solitary wave) stage. This does not mean however that dispersive effects are not important also in deepwater as shown next.

The results of this study agree quite well with the results of Madsen et al. given that their model is based on a different Boussinesq-type expansion of the vertical velocity involving up to fifth-order derivatives of the depth-averaged velocity field [31, 24]. This test case can be reproduced using the scripts on the Basilisk web site [43].

We now consider more realistic parameters. A rectangular initial perturbation includes very short-wavelength components and one may wonder whether dispersive effects could be mainly related to these (unrealistic) artefacts. While seismic fault ruptures can lead to sharp discontinuities (i.e. actual cracks in the seafloor), the largest vertical displacements usually occur at depths of several kilometres, so that the resulting seafloor displacement is also distributed over several kilometres. To take this observation into account, we replace the rectangular initial

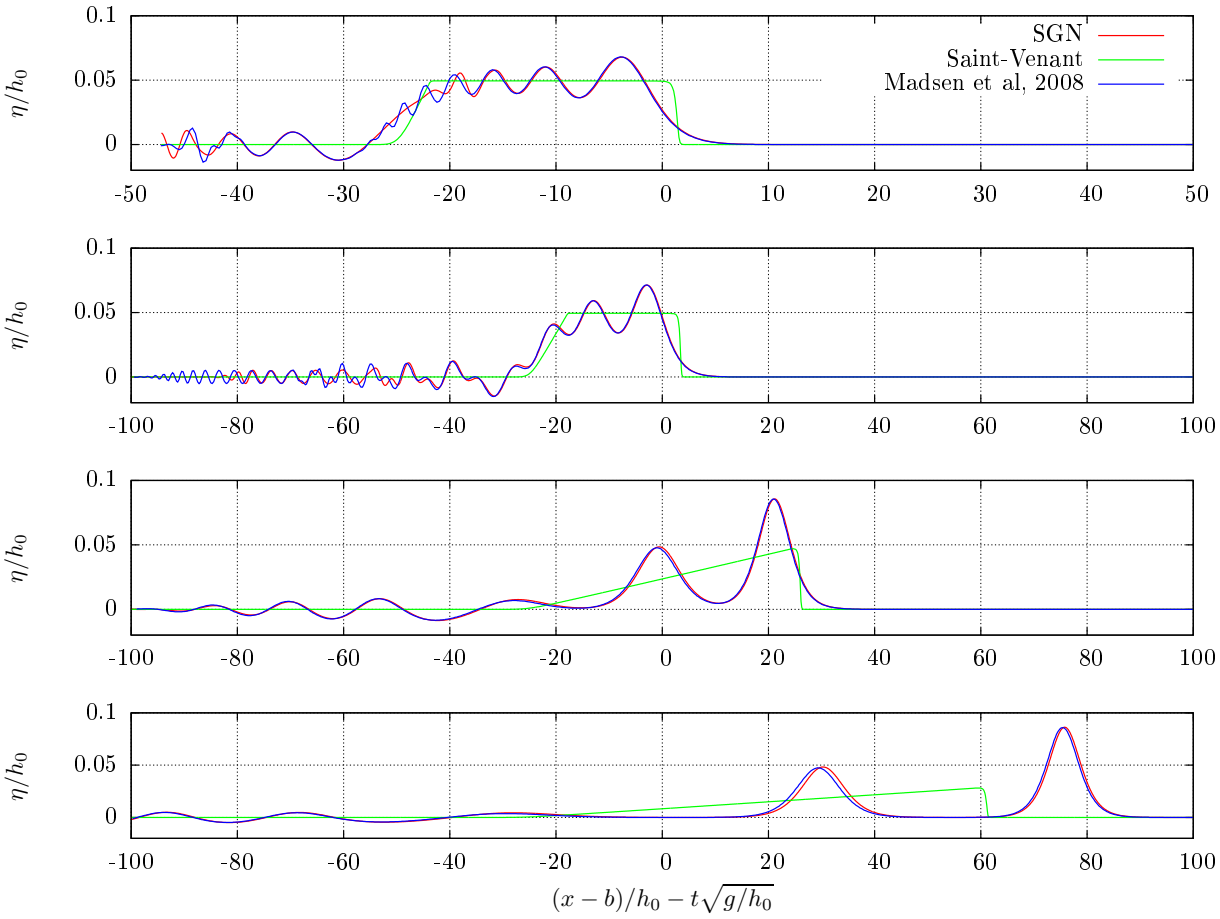


Figure 3: Simplied tsunami model. Evolution of the free-surface elevation η/h_0 with time. The frame of reference moves with the wave at the characteristic wave speed $\sqrt{gh_0}$ i.e. the abscissa is $(x-b)/h_0 - t\sqrt{g/h_0}$. The times are from top to bottom: $t\sqrt{g/h_0} = 35,\ 90,\ 700,\ 2000$. The red curves are reproduced from [30], Figure 1.

displacement with a smoother function

$$\eta_0(x) = \frac{a}{2}\begin{cases} 1-\cos\frac{x}{l_1\pi} & \text{if } 0 \leqslant x < l_1 \\ 2 & \text{if } l_1 \leqslant x < l_1+l_2 \\ 1+\cos\frac{x-l_1-l_2}{l_3\pi} & \text{if } l_1+l_2 \leqslant x < l_1+l_2+l_3 \\ 0 & \text{otherwise} \end{cases}$$

The steepest slopes are now of order a/l_1 and a/l_3. To match the case of the Tohoku tsunami of section 4.4, we choose $h_0 = 6000$ m, $a = 16$ m, $l_1 = 40$ km, $l_2 = 20$ km and $l_3 = 10$ km. The simulation domain is 7200 km long, discretised with a maximum of 13 levels. The adaptation criterion on η is $10^{-6}h_0 = 6$ mm. The computed wave evolution is displayed in Figure 4. Nonlinearities (as measured by a/h_0) are much smaller than in the previous case. This is immediately apparent for the Saint-Venant solution which changes very little as it propagates at a speed close to $\sqrt{gh_0}$. Dispersion however is comparable to that in the previous case (as measured by h_0/b or $h_0/(l_1+l_2+l_3)$) and leads to an important evolution of the wave front with time. Also noticeable is the absence of short waves at initial times when compared to the case in Figure 3 (due to the smoother initial profile).

After three hours of propagation, the dispersive and non-dispersive solutions are very different. For this case, the breakup into a train of solitary waves will occur much later (and would typically require travelling distances much larger than the circumference of the earth).

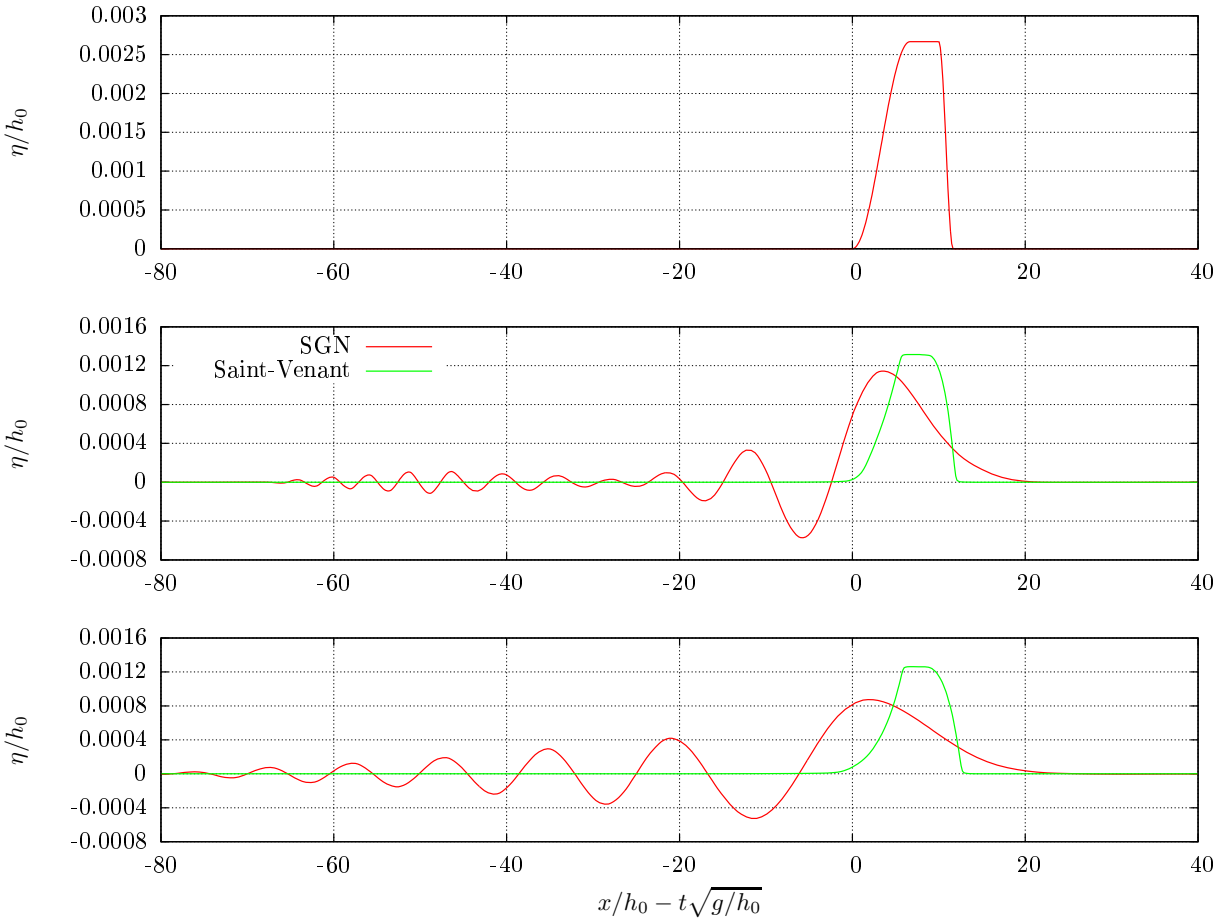


Figure 4: Simplified but realistic tsunami model. Evolution of the free-surface elevation η/h_0 with time. The frame of reference moves with the wave at the characteristic wave speed $\sqrt{gh_0}$ i.e. the abscissa is $x/h_0 - t\sqrt{g/h_0}$. The times are from top to bottom: $t\sqrt{g/h_0} = 0$, 145, 435 corresponding to dimensional times of approximately 0, 1 and 3 hours.

4.2 Periodic wave propagation over an ellipsoidal shoal

We now consider a two-dimensional case. We follow [25] and try to reproduce the experiment of [3]. The numerical wave tank is 25^2 metres and periodic sinusoidal waves are generated on the left-hand side and are damped on the right-hand side. The domain is discretised with 1024^2 grid points. The acceleration of gravity is set to 9.81 m/s^2. The period of the generated waves is one second and the amplitude matches that measured in the experiments. The bathymetry is an inclined and skewed plane combined with an ellipsoidal shoal (see [3, 43] and Figure 5). The absorbing boundary condition on the right-hand-side is important to avoid wave reflections. It is implemented by adding a zone of linearly-increasing quadratic bottom friction for $x > 12$ metres. Starting from a flat free surface, waves are generated for 50 seconds and the maximum wave elevation is recorded over the last 10 seconds. The instantaneous wave field at $t = 50$ is represented in Figure 6 and the maximum wave amplitude over the last 10 seconds in Figure 7. The ellipsoidal shoal causes wave focusing at the back of the shoal. This is primarily a linear effect. Nonlinear effects and dispersion are responsible for the finger-like structures due to the non-linear creation of harmonics and their dispersion in water of variable depth. In Figure 8 the maximum wave amplitude obtained numerically is compared to the experimental measurements obtained along the cross-sections represented in Figure 7. The agreement is quite good and comparable to that obtained by [25] (Figure 19) although we have to use a higher resolution (1024^2 instead of 300^2) because our scheme is only second-order (versus 4th order).

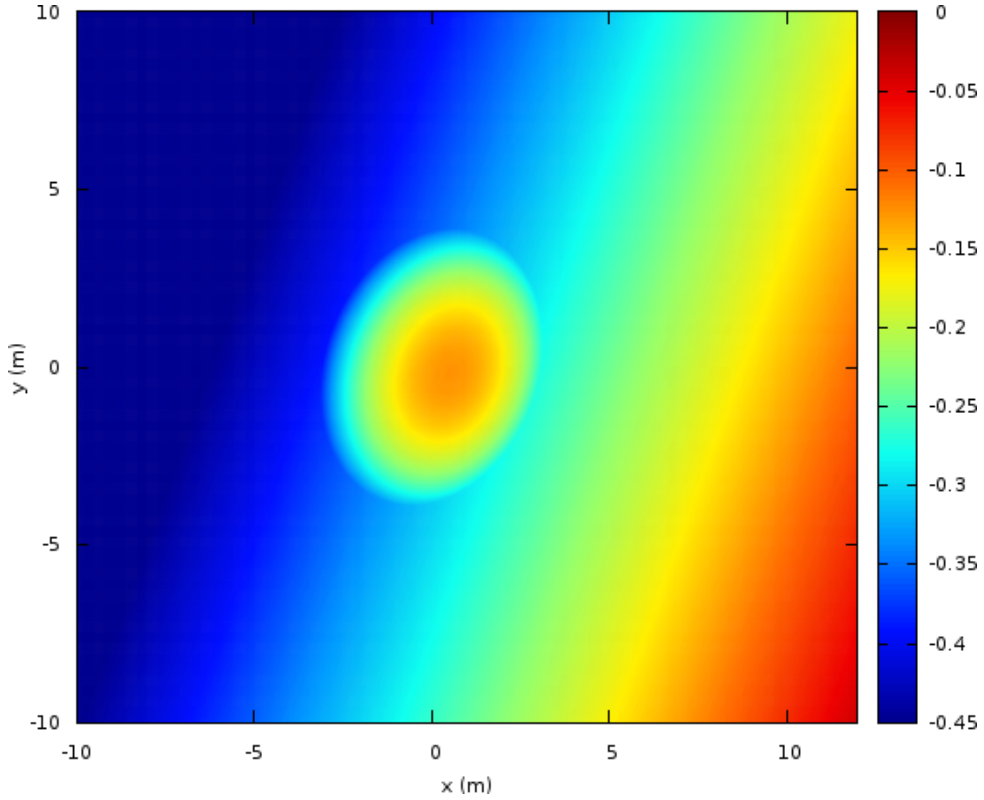


Figure 5: Bathymetry for periodic wave propagation over an ellipsoidal shoal. Depth in metres.

The previous test case was only one-dimensional and ran in less than a minute on a standard laptop computer. This two-dimensional test case is much more expensive. It required 9158 time steps for the 50 seconds of simulated time (the timestep is controlled by the CFL condition for gravity waves i.e. $\Delta t < \Delta/\sqrt{gh}$) and took approximately four hours of wall-clock time on a single Intel i7 core. For the 1024^2 grid we used, this gives an absolute computational speed of approximately $1024^2 \times 9158/4/3600 \simeq 6.7 \times 10^5$ points $\times$ steps/sec/core.

4.3 Runup of a solitary wave on a conical island

We now consider the model of a classical laboratory experiment meant to mimic interaction of a tsunami-like wave with topography (although we have stressed in section 4.1 that solitons are probably not a good model for deep-water tsunamis). This test case has been used by many authors using both Saint-Venant and dispersive models (e.g. [19, 35, 25, 21] etc...). The details of the numerical setup and figure generation are available at [43]. The computational domain is 27.6 metres squared, resolved with a maximum of 256^2 grid points. We reproduce case B, i.e. a relative amplitude of the solitary wave of 0.09. We record the water surface elevation at 5 locations corresponding to the experiment of [5] (WG3, WG6, WG9, WG16 and WG22). We use a quadtree discretisation where the mesh size is adapted according to the estimated error on the free-surface elevation η with ϵ set to 0.3 mm (as defined in section 3.5.1). The evolutions of the free surface and quadtree mesh are displayed in the left- and right-columns of Figure 9. Mesh adaptation clearly captures the reflected waves. Wetting and drying occurs mostly at the front (runup) and back (rundown) of the island. This test case demonstrates the robustness of the solver when combining dispersive terms, wetting/drying and adaptivity. Figure 10 gives a comparison of the experimental and numerical timeseries at the various gauges. Both the dispersive and Saint-Venant results are displayed. The results for the SGN solver are very similar to those of [21] (Figure 14) based on the equations of [36] and to the SGN solution of [25] (Figure 20). The results for the Saint-Venant solver (see also e.g. Figure 18 of [19] and

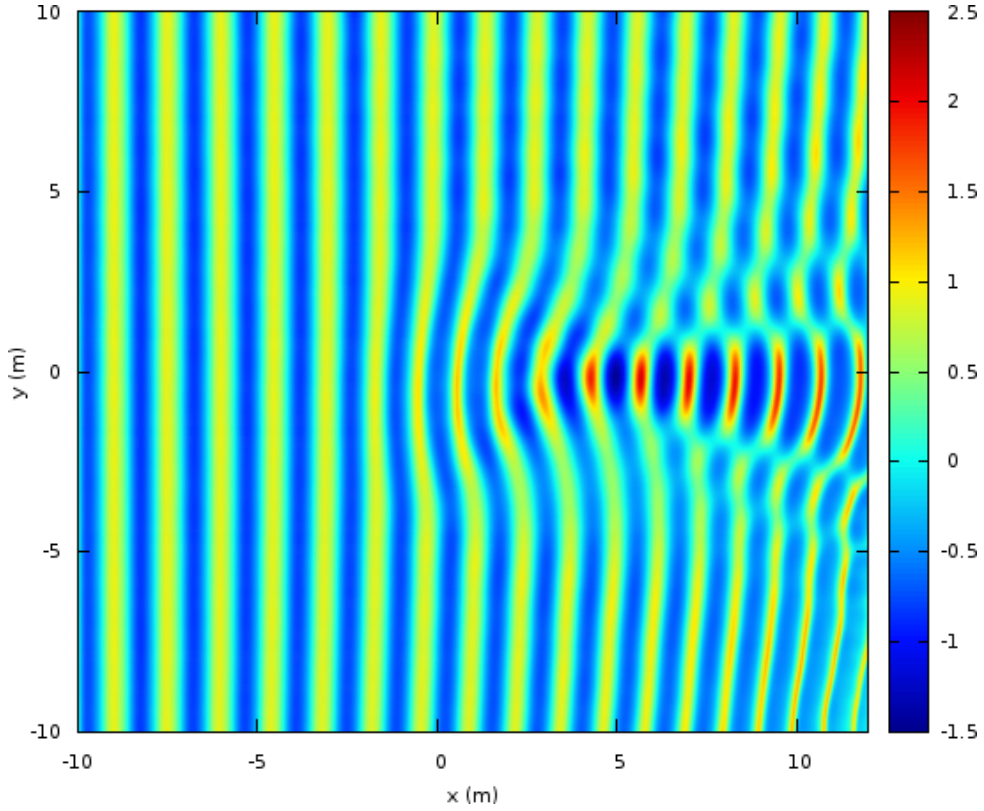


Figure 6: Instantaneous wave field at $t = 50$ seconds. Wave amplitude in centimeters.

Figure 19 of [35]) have wave fronts which are typically too sharp and shallower wave troughs. Although the agreement is not as good for the latter part of the wave, this can probably be explained by experimental errors in the generation of the solitary wave [11].

We can objectively quantify the computational gain obtained with quadtree adaptivity. To do so we ran the same simulation but switched to a non-adaptive Cartesian low-level grid implementation. As discussed previously, this does not require any modification of the high-level implementation discussed above. The overall implementation efficiency is thus identical to that of a pure, optimised Cartesian grid implementation. To further illustrate this point, we also ran a third computation using a non-adaptive quadtree grid implementation. The results of this third computation are of course strictly identical to the results of the second computation (because the grid layouts are identical), the runtimes however are different because the third computation carries the overhead of dealing with the more complex quadtree structure. The results (e.g. the timeseries of Figure 10) are almost identical for all computations. Table 1 gives a summary of the runtimes and speeds for the different computations.

Grid	Quadtree (adaptive)		Cartesian		Quadtree (constant)	
Equations	Saint-Venant	SGN	Saint-Venant	SGN	Saint-Venant	SGN
Wall-clock time	15.3	68.2	34.1	200	74.2	322
Average # points	9238	10556	65536	65536	65536	65536
Speed	0.76	0.21	2.51	0.43	1.16	0.27

Table 1: Computing times and speeds for the conical island test case. The computations were done on one core of an Intel i7 processor. The wall-clock times are in seconds and the speeds in million points×steps/sec/core.

For this example, the average gain in number of grid points when using an adaptive mesh is roughly a factor of six. The corresponding wall-clock time gain when comparing with a pure

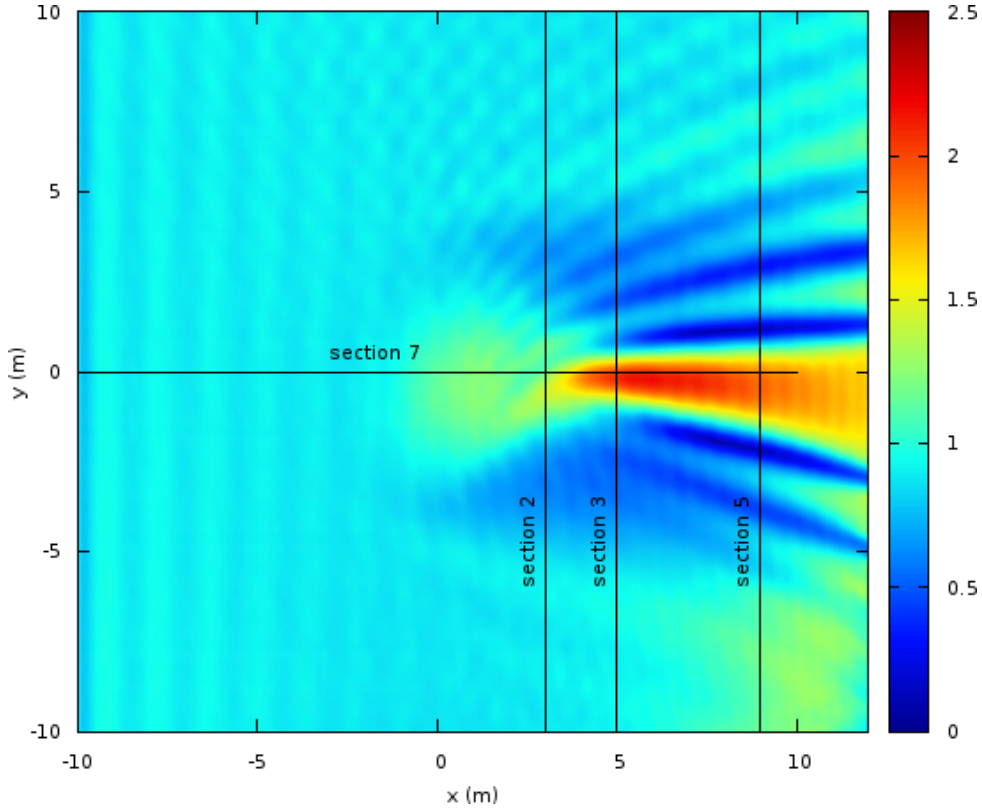


Figure 7: Maximum wave amplitude (centimeters) for $40 < t < 50$ seconds. The experimental cross sections are also represented.

Cartesian grid implementation is a factor of two for the Saint-Venant equations and a factor of three for the SGN equations. This difference is due to the high performance of the Saint-Venant solver on pure Cartesian grids (with one timestep taking one second for a grid with 2.51 millions points). Correspondingly, the computational overhead (per grid point) due to the quadtree grid is roughly a factor of three for Saint-Venant and two for SGN. Note that the corresponding absolute computational speeds are still high.

When using a constant-resolution quadtree grid, the computational speed is about 30% higher than for adaptive quadtree grids. This reflects the relatively small overhead induced by the application of the internal "boundary conditions" described in section 3.5. The overhead compared to the pure Cartesian grid implementation is thus reduced to a factor of two or less. This underlines the good performance of the quadtree implementation in Basilisk.

Table 1 also gives a measure of the cost of adding the SGN dispersive terms to the Saint-Venant system. This is roughly a four- to six-fold increase in computational time. The least favourable case being that of the pure Cartesian grid. This may seem large, however it can be related to the estimated cost of the multigrid solution of the dispersive term. The cost of the algorithm described in section 3.3 will be dominated by the application of the (complicated) relaxation operator on the finest grid. Assuming that this cost is comparable to that of the solution of the Saint-Venant system (also on the finest grid) and given that we do four iterations of the relaxation operator, a four- to six-fold computational overhead is not unexpected.

Finally note that, while a gain of a factor of two or three in computational time may seem modest, this gain will be problem- and resolution-dependent. As discussed in [39] this gain will scale like

$$C\Delta^{2-d}$$

with C a constant, Δ the minimum length scale and d the *effective dimension* of the problem considered. The computational gain will thus increase when increasing spatial resolution. The

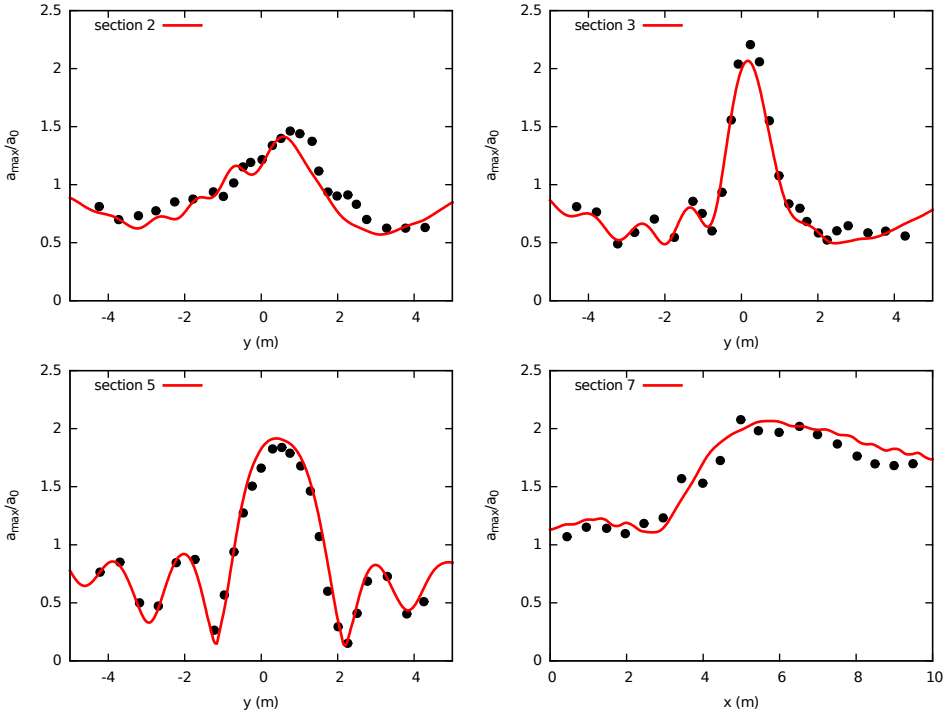


Figure 8: Comparison of the maximum wave height with the experimental data of [3] (circles) along the cross-sections represented in Figure 7.

256^2 resolution we used for this test case is quite low for a simulation of a tsunami-like phenomenon and we can expect much larger gains for more realistic tsunami simulations, as done in the next section.

4.4 Application to the Tohoku tsunami

Adaptive numerical methods for the Saint-Venant and/or SGN equations are particularly interesting for the modelling of tsunami propagation and inundation. Tsunamis indeed involve spatial scales ranging from oceanic basins down to wave interactions with individual buildings. A relatively classical way to deal with this huge range is to rely on "nested models" where successively finer meshes (and often different models) are used to resolve finer scales [15]. This poses a number of problems, some practical, some more fundamental (appropriate choices of resolution, coupling between solutions etc...). It is also possible to use variable-resolution but non-adaptive meshes which implies a number of assumptions about the scale distributions of tsunami processes (similar in fact to the case of nested grids) [51]. Adaptive methods on the other hand can provide efficient solutions which are much less dependent on a priori assumptions. While both block-AMR [2] and quadtree-adaptive methods [39, 40] have already been applied to tsunami modelling with the Saint-Venant equations, their application to dispersive wave propagation is rare. We believe that one of the explanation for this rarity is the difficulty of efficiently solving large linear systems on adaptive meshes.

Although the Tohoku tsunami of 11th March 2011 was a shocking natural disaster, it provided a wealth of field data which will hopefully lead to an improved understanding and thus improved resilience to future tsunamis. Consequently it is a good case to try to reproduce numerically. In a previous publication [40], we showed that using a quadtree-adaptive Saint-Venant solver one could efficiently obtain very satisfactory predictions of both long-distance propagation (as measured by deep-water DART buoys) and local inundation (measured by satellite and local GPS records) within the same model. The maximum spatial resolution was ~ 250 metres for a

domain spanning 73° (∼8100 km).

In this section, we reproduce the numerical simulation of [40] using the new implementation of the scheme in Basilisk and including dispersive effects. The parameters are the same as in [40]: the domain is 73° squared discretised on a longitude-latitude grid (Figure 4 of [40]), the initial surface elevation is given by the seismic inversion of Shao, Li and Ji (Figure 5 of [40]). The maximum deep-water spatial resolution is ~ 0.5 arc-minute. The grid is adapted using an error threshold of one centimetre for the free-surface elevation and five centimetres for the maximum wave elevation. The corresponding wave and grid evolutions for the Saint-Venant solution are displayed in Figure 11. The results are close to those obtained with Gerris (see Figure 6 of [40]).

The importance of dispersive effects in the evolution of tsunamis is still a debated topic. An argument often heard is that, while dispersion is probably necessary to explain the successive wave trains observed close to shore [30], its influence may be small for deep-water propagation. Despite these potential limitations, Saint-Venant models seem to be able to produce convincing predictions compared to field measurements of tsunamis (for which the influence of near-shore effects is not negligible) [40]. Some of these arguments however may be tainted by a certain amount of "wishful thinking" linked to the difficulty of implementing dispersive terms and to their impact on computational cost (as outlined in the previous section). Furthermore, recent studies underline the importance of dispersion in the trans-oceanic, deep-water propagation of tsunamis [23, 18, 22].

Figure 12 reproduces the results of Figure 11 but with the inclusion of the SGN dispersive terms. All other parameters are identical. A clear dispersive wave train is generated behind the leading frontal wave. This wave train is even clearer when looking at the adaptive grid in the right column of the figure. Close-up views of the structure of the wave front for Saint-Venant and SGN are given in Figure 13. It is clear that dispersion plays an important role in the deep-water, long-distance evolution of the leading wave front. Dispersion leads to a much smoother (and lower) primary wave front followed by a train of dispersive waves. The SGN wave front is also slightly slower than for Saint-Venant. These very significant differences in the structure of the leading wave front will clearly have an important influence on the details of long-distance tsunami impacts (interactions with coastlines and inundation).

In light of the simplified one-dimensional tsunami model of section 4.1, these results are not unexpected. In Figure 14 we represent a vertical cross-section through the initial free-surface and bathymetry, corresponding to the source model we have used. The cut plane is perpendicular to the main fault plane and goes through the maximum of free surface elevation (see e.g. Figure 5 of [40]) i.e. it is roughly aligned with the direction of propagation of the primary wave front. Note that, while the details of the source model can be discussed, it has been validated in detail in [40] and the relevant characteristic scales will be similar for other source models. The fault plane clearly matches the deep trough in the bathymetry around 144°, as expected for a subduction zone. The maximum amplitude of vertical deformation is very large ($a \simeq$16 metres) which, based on a characteristic depth $h_0 = 6000$ metres gives a nonlinearity parameter $\epsilon = a/h_0 = 16/6000 \simeq 2.7 \times 10^{-3}$. Also noticeable is the left-right asymmetry in the initial wave shape, with a much "steeper" eastern front. The characteristic scales of the one-dimensional model in section 4.1 were chosen to approximately match this profile. The (largest) dispersive terms are thus of order $\mu = h_0^2/l_3^2 \simeq 0.36$, which is not negligible as clearly apparent in Figures 4, 12 and 13. The cross-section in Figure 15 illustrates the similarity between the structure of the front for the full two-dimensional solution and for the simplified one dimensional model (i.e. last frame of Figure 4).

These simulations also underline the importance of using a high-enough spatial resolution for deep-water wave propagation. If we had assumed characteristic tsunami wavelengths scaling like $\sqrt{h}$, as is often done to generate static variable-resolution grids, dispersive effects would most likely have been wiped-out by the lack of resolution.

The results in this section are comparable to those in Kirby et al [22], who pointed out the

importance of dispersive effects when considering propagation of the Tohoku tsunami over the entire Pacific ocean.

In [40] a detailed study of fine-scale flooding on the Japanese coastline showed that the Saint-Venant model was able to satisfactorily reproduce observed patterns down to the maximum resolution of 250 metres. The inclusion of SGN dispersive terms does not seem to produce significant, systematic changes at these scales. As pointed out by [30] this can be explained by the fact that dispersion will decrease for waves travelling toward the shore (assuming that the water depth decreases faster than the characteristic wavelength). On the other hand, non-linearities will increase and so, we can expect the solution to be well-captured by Saint-Venant only. Dispersion can play a role close to shore, leading to dispersive wave trains riding on top of the primary Saint-Venant tsunami wave [30]. The extent of flooding and damages are likely to be controlled primarily by the non-dispersive wave, however. Note that these arguments apply only regarding inundation for land located close to the source (i.e. Japan for the case of the Tohoku tsunami). Inundation of land located far from the source (e.g. Pacific islands and the coasts of North America in the case of the Tohoku tsunami) will depend on the details of deep-water propagation, including dispersion, as argued previously.

Finally Table 2 summarises the performances of the quadtree-adaptive Saint-Venant and SGN models. The models were run to a physical time $t = 400$ minutes (approximately 18 000 timesteps) in parallel on 8 Intel i7 cores. The performances of the equivalent Cartesian models are also given for comparison. Note that we did not actually run the Cartesian models but estimated the runtimes based on the performances for a few timesteps. The SGN models are approximately four times more expensive than the Saint-Venant models. The gain in number of grid points (i.e. memory) obtained with adaptivity is approximately a factor of 240, while the gain in computing time is roughly a factor of 40.

Grid	Quadtree-adaptive		Cartesian	
Equations	Saint-Venant	SGN	Saint-Venant	SGN
Wall-clock time	44	160	$\simeq 1500$	$\simeq 6300$
Average # points	2.8×10^5	2.8×10^5	$2^{26} \simeq 67 \times 10^6$	$\simeq 67 \times 10^6$
Speed	0.23	0.06	1.59	0.4

Table 2: Computing times and speeds for the Tohoku tsunami. The computations were done on 8 Intel i7 cores. The wall-clock times are in minutes and the speeds in million points×steps/sec/core.

5 Conclusions and perspectives

We have shown how an adaptive Serre–Green–Naghdi (SGN) solver can be implemented by combining an existing, well-balanced, adaptive Saint–Venant solver with a geometric multigrid inversion of the linear system defining the SGN momentum source term. Provided one uses a generic multigrid scheme (such as provided in Basilisk), the implementation of the relaxation and residual SGN operators is not difficult, despite the complexity of their mathematical formulation.

In the same spirit, the Basilisk framework allows a straightforward, high-performance, generalisation of this scheme to adaptive quadtree grids. As was previously demonstrated for the Saint-Venant equations, adaptivity leads to order-of-magnitude gains in computational efficiency for realistic applications to tsunamis. The documented implementation is available freely on the Basilisk web site [43].

Although dispersive terms are often associated with short coastal waves, they are also important for the deep-water propagation of tsunami waves, as illustrated in Figure 13. They should be included in numerical models of tsunamis, despite their high computational overhead compared to Saint-Venant (a four- to six-fold increase in our case).

The scheme we have presented is only second-order accurate in space and time. There are clear indications that higher-order accuracy would be beneficial for the propagation of dispersive waves (for example for the test case of section 4.2 or for the deep-water propagation of tsunamis). The extension to fourth-order accuracy of the multigrid solver would not be difficult within the Basilisk framework, given the availability of 5×5 stencils by default. This would need to be coupled with a higher-order accuracy implementation of the Saint-Venant solver, which should not be an issue either. The resulting scheme would also generalise straightforwardly to high-order, adaptive quadtree grids.

For the moment the Basilisk framework only supports shared-memory OpenMP parallelism. As for quadtrees, this parallelism is transparent and does not appear in the implementation of the algorithms described in this article. The performance of shared-memory parallelism is limited however, and we are currently working on a generalisation of the framework to distributed-memory MPI parallelism. Coupled with adaptive grids, this should allow efficient, very high-resolution, dispersive tsunami modelling on large parallel systems.

Finally, given the importance of dispersion relative to non-linearities for deep-water tsunami propagation, it seems clear that weakly-nonlinear, strongly dispersive models [31] could be better suited to describe this part of the evolution. This raises the interesting question of coupling or generalising the SGN equations to recover the deep-water (linear) dispersion relation.

Acknowledgments

We thank Pierre-Yves Lagrée for his encouragements and helpful suggestions and David Lannes and Fabien Marche for sharing an early version of their article.

References

[1] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM Journal on Scientific Computing*, 25(6):2050–2065, 2004.

[2] M. J. Berger, D. L. George, R. J. LeVeque and K. T. Mandli. The GeoClaw software for depth-averaged flows with adaptive refinement. *Advances in Water Resources*, 34(9):1195–1206, 2011.

[3] J.C.W. Berkhoff, N. Booy and A.C. Radder. Verification of numerical wave propagation models for simple harmonic linear water waves. *Coastal Engineering*, 6(3):255–279, 1982.

[4] P. Bonneton, F. Chazel, D. Lannes, F. Marche and M. Tissier. A splitting approach for the fully nonlinear and weakly dispersive Green-Naghdi model. *Journal of Computational Physics*, 230:1479–1498, 2011.

[5] M.J. Briggs, C.E. Synolakis, G.S. Harkins and D.R. Green. Laboratory experiments of tsunami runup on a circular island. In *Tsunamis: 1992–1994*, pages 569–593. Springer, 1995.

[6] M. Brocchini and N. Dodd. Nonlinear shallow water equation modeling for coastal engineering. *J. Waterway, Port, Coastal, Ocean Eng.*, 134(2):104–120, 2008.

[7] R. Cienfuegos, E. Barthelemy and P. Bonneton. A fourth-order compact finite volume scheme for fully nonlinear and weakly dispersive Boussinesq-type equations. Part I: model development and analysis. *International Journal for Numerical Methods in Fluids*, 51(11):1217–1253, 2006.

[8] R. Cienfuegos, E. Barthelemy and P. Bonneton. A fourth-order compact finite volume scheme for fully nonlinear and weakly dispersive Boussinesq-type equations. Part II: boundary conditions and validation. *International Journal for Numerical Methods in Fluids*, 53(9):1423–1456, 2007.

[9] A. Duran and F. Marche. Discontinuous-Galerkin discretization of a new class of Green-Naghdi equations. *Communications in Computational Physics*, 17(03):721–760, 2015.

[10] D. Dutykh, D. Clamond, P. Milewski and D. Mitsotakis. Finite volume and pseudo-spectral schemes for the fully nonlinear 1D Serre equations. *European Journal of Applied Mathematics*, 24(05):761–787, 2013.

[11] D.R. Fuhrman and P.A. Madsen. Simulation of nonlinear wave run-up with a high-order Boussinesq model. *Coastal Engineering*, 55(2):139–154, 2008.

[12] D.L. George and R.J. LeVeque. High-resolution methods and adaptive refinement for tsunami propagation and inundation. In *Hyperbolic Problems: Theory, Numerics, Applications*, pages 541–549. Springer, 2008.

[13] D. G. Goring. *Tsunamis–the propagation of long waves onto a shelf*. PhD thesis , California Institute of Technology, 1978.

[14] A. E. Green and P. M. Naghdi. A derivation of equations for wave propagation in water of variable depth. *Journal of Fluid Mechanics*, 78(02):237–246, 1976.

[15] S.T. Grilli, S. Dubosq, N. Pophet, Y. Pérignon, J.T. Kirby and F. Shi. Numerical simulation and first-order hazard analysis of large co-seismic tsunamis generated in the Puerto Rico trench: near-field impact on the north shore of Puerto Rico and far-field impact on the US east coast. *Natural Hazards and Earth System Sciences*, 10(10):2109–2125, 2010.

[16] F. Günther, M. Mehl, M. Pögl and C. Zenger. A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves. *SIAM Journal on Scientific Computing*, 28(5):1634–1650, 2006.

[17] D. D. Holm and B. T. Nadiga. Modeling mesoscale turbulence in the barotropic double-gyre circulation. *Journal of physical oceanography*, 33(11):2355–2365, 2003.

[18] J. Horrillo, Z. Kowalik and Y. Shigihara. Wave dispersion study in the Indian Ocean tsunami of December 26, 2004. *Marine Geodesy*, 29(3):149–166, 2006.

[19] J. Hou, Q. Liang, F. Simons and R. Hinkelmann. A stable 2D unstructured shallow flow model for simulations of wetting and drying over rough terrains. *Computers & Fluids*, 82:132–147, 2013.

[20] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer-Verlag Berlin, 1985.

[21] M. Kazolea, A.I. Delis, I.K. Nikolos and C.E. Synolakis. An unstructured finite volume numerical scheme for extended 2D Boussinesq-type equations. *Coastal Engineering*, 69:42–66, 2012.

[22] J. T. Kirby, F. Shi, B. Tehranirad, J. C. Harris and S. T. Grilli. Dispersive tsunami waves in the ocean: model equations and sensitivity to dispersion and Coriolis effects. *Ocean Modelling*, 62(0):39–55, 2013.

[23] E. Kulikov. Dispersion of the Sumatra tsunami waves in the Indian Ocean detected by satellite altimetry. *Russ. J. Earth Sci*, 8, 2006.

[24] D. Lannes and P. Bonneton. Derivation of asymptotic two-dimensional time-dependent equations for surface water wave propagation. *Physics of Fluids*, 21(1):16601, 2009.

[25] D. Lannes and F. Marche. A new class of fully nonlinear and weakly dispersive Green-Naghdi models for efficient 2D simulations. *Journal of Computational Physics*, 282:238–268, 2015.

[26] R. J. LeVeque and D. L. George. High-resolution finite volume methods for the shallow water equations with bathymetry and dry states. In *Proceedings of Long-Wave Workshop, Catalina*. 2004.

[27] R. J. LeVeque, D. L. George and M. J. Berger. Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, 20:211–289, 2011.

[28] M. Li, P. Guyenne, F. Li and L. Xu. High order well-balanced CDG–FE methods for shallow water waves by a Green–Naghdi model. *Journal of Computational Physics*, 257:169–192, 2014.

[29] Q. Liang, A.G.L. Borthwick and G. Stelling. Simulation of dam-and dyke-break hydrodynamics on dynamically adaptive quadtree grids. *International journal for numerical methods in fluids*, 46(2):127–162, 2004.

[30] P. A. Madsen, D. R. Fuhrman and H. A. Schäffer. On the solitary wave paradigm for tsunamis. *Journal of Geophysical Research: Oceans*, 113(C12):0, 2008.

[31] P. A. Madsen, D. R. Fuhrman and B. Wang. A boussinesq-type method for fully nonlinear waves interacting with a rapidly varying bathymetry. *Coastal Engineering*, 53(5–6):487–504, 2006.

[32] D. Mitsotakis, B. Ilan and D. Dutykh. On the Galerkin/Finite-Element method for the Serre equations. *J. Sci. Comput.*, 61:166–195, 2014.

[33] O. Le Métayer, S. Gavrilyuk and S. Hank. A numerical scheme for the Green-Naghdi model. *Journal of Computational Physics*, 229:2034–2045, 2010.

[34] B. T. Nadiga, L. G. Margolin and P. K. Smolarkiewicz. Different approximatons of shallow fluid over an obstacle. *Physics of Fluids*, 8(8):2066–2077, 1996.

[35] I.K. Nikolos and A.I. Delis. An unstructured node-centered finite volume scheme for shallow water flows with wet/dry fronts over complex topography. *Computer Methods in Applied Mechanics and Engineering*, 198(47):3723–3750, 2009.

[36] O. Nwogu. Alternative form of Boussinesq equations for nearshore wave propagation. *Journal of waterway, port, coastal, and ocean engineering*, 119(6):618–638, 1993.

[37] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *J. Comput. Phys.*, 190(2):572–600, 2003.

[38] S. Popinet. The Gerris Flow Solver. `http://gfs.sf.net`, 2003–2014.

[39] S. Popinet. Quadtree-adaptive tsunami modelling. *Ocean Dynamics*, 61(9):1261–1285, 2011.

[40] S. Popinet. Adaptive modelling of long-distance wave propagation and fine-scale flooding during the Tohoku tsunami. *Natural Hazards and Earth System Sciences*, 12(4):1213–1227, 2012.

[41] S. Popinet. A solver for the Saint-Venant equations. `http://basilisk.fr/src/saint-venant.h`, 2013.

[42] S. Popinet. Basilisk C reference manual. `http://basilisk.fr/Basilisk C`, 2013.

[43] S. Popinet. A solver for the Green-Naghdi equations. `http://basilisk.fr/src/green-naghdi.h`, 2014.

[44] S. Popinet. Multigrid Poisson–Helmholtz solvers. `http://basilisk.fr/src/poisson.h`, 2014.

[45] S. Popinet, R.M. Gorman, G.J. Rickard and H.L. Tolman. A quadtree-adaptive spectral wave model. *Ocean Modelling*, 34(1):36–49, 2010.

[46] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of computation*, 37(155):105–126, 1981.

[47] F. Serre. Contribution à l'étude des écoulements permanents et variables dans les canaux. *La Houille Blanche*, (3):374–388, 1953.

[48] F. Shi, J.T. Kirby, J.C. Harris, J.D. Geiman and S.T. Grilli. A high-order adaptive time-stepping TVD solver for Boussinesq modeling of breaking waves and coastal inundation. *Ocean Modelling*, 43:36–51, 2012.

[49] Y. Tang, R.A. Chowdhury, Bradley C. Kuszmaul, C.-K. Luk and C.E. Leiserson. The pochoir stencil compiler. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 117–128. ACM, 2011.

[50] M. Tissier, P. Bonneton, F. Marche, F. Chazel and D. Lannes. A new approach to handle wave breaking in fully non-linear Boussinesq models. *Coastal Engineering*, 67:54–66, 2012.

[51] R.A. Walters. Coastal ocean models: two useful finite element methods. *Continental Shelf Research*, 25(7):775–793, 2005.

[52] G. Wei, J.T. Kirby, S.T. Grilli and R. Subramanya. A fully nonlinear Boussinesq model for surface waves. Part 1. Highly nonlinear unsteady waves. *Journal of Fluid Mechanics*, 294:71–92, 1995.

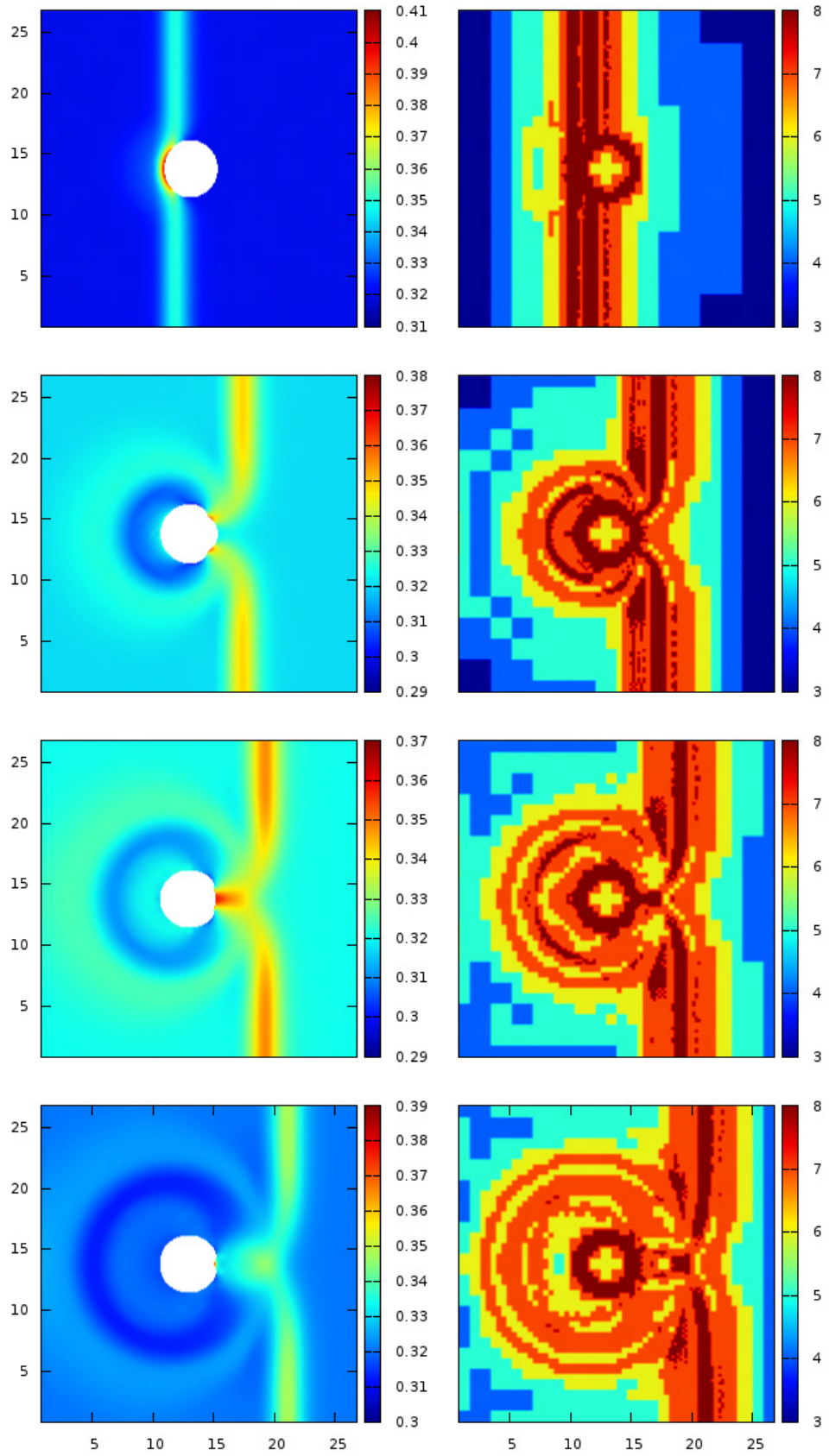


Figure 9: Evolution of the free surface (left column) and quadtree mesh (right column) for the runup of a solitary wave on a conical island. The times are from top to bottom: 9, 12, 13, 14 and 20 seconds. The free surface elevation is in metres on the left column and the "level" l of the quadtree mesh is given on the right column. The corresponding spatial resolution is $\Delta = 27.6/2^l$ metres.

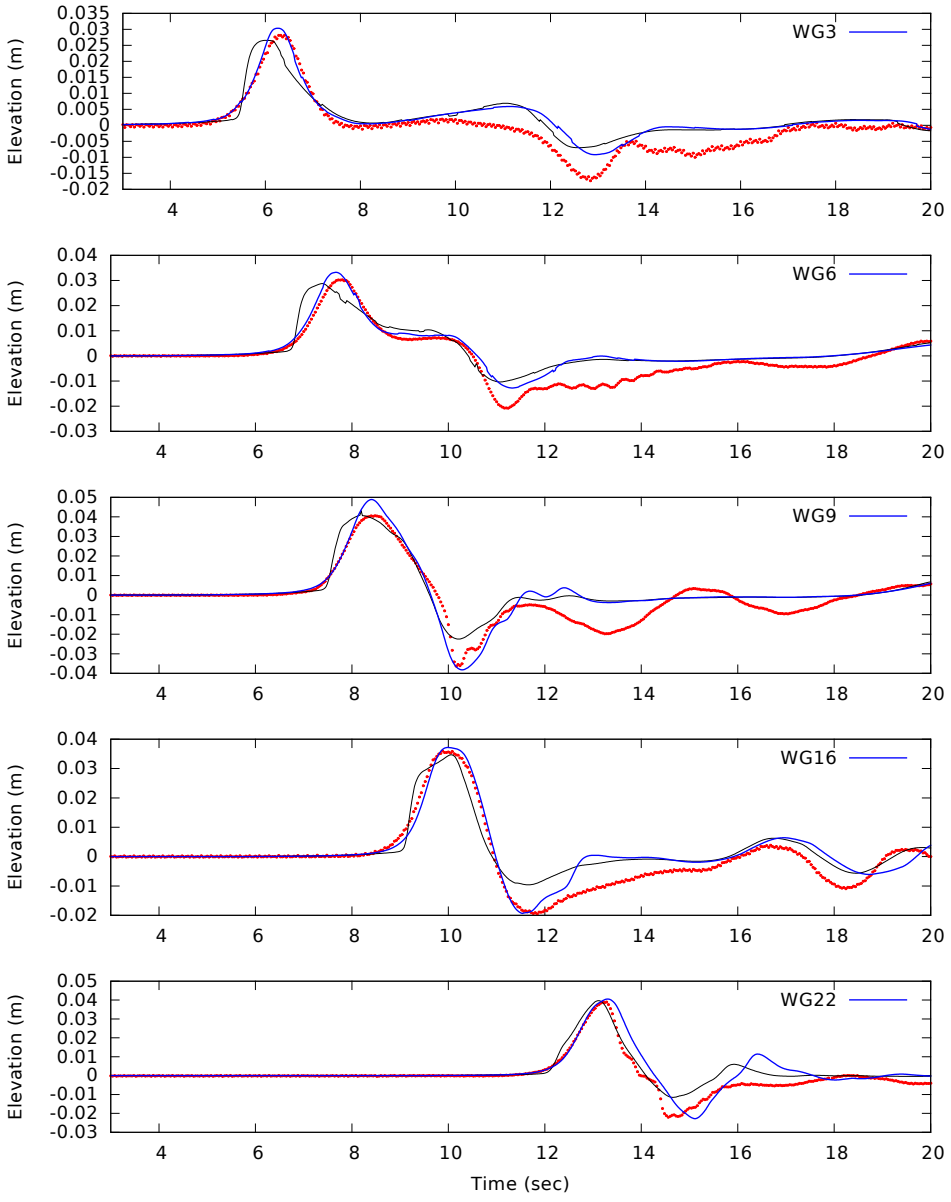


Figure 10: Timeseries of free surface elevation. The symbols are the experimental measurements of [5]. The numerical solution of the SGN equations is given in blue and the Saint-Venant equations in black.

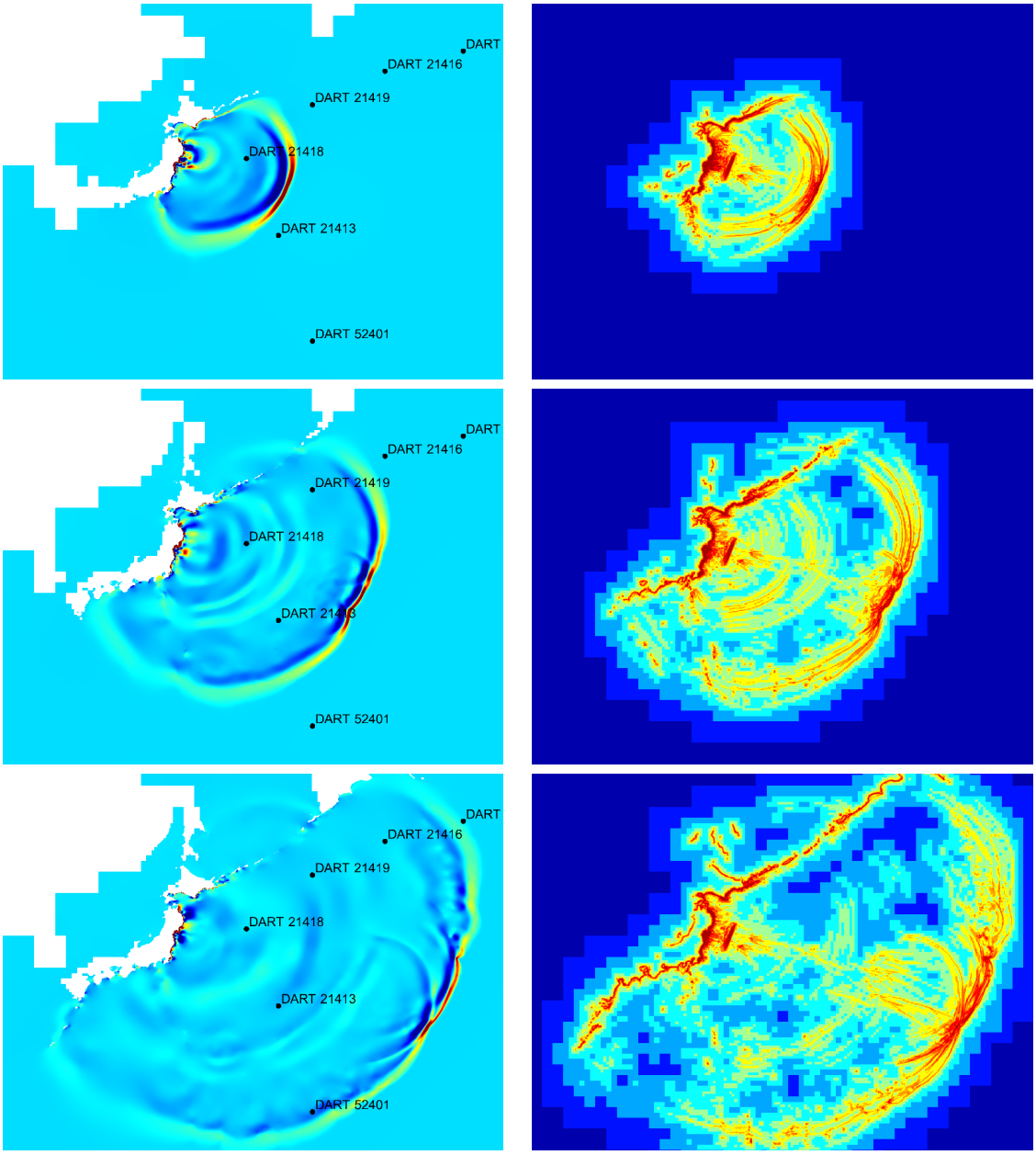


Figure 11: Tohoku tsunami modelled with the Saint-Venant equations. Evolution of the free surface elevation (left column) and adaptive grid (right column). The elevation scale ranges from -1 metre (dark blue) to $+2$ metres (dark red). The level of refinement ranges from 5 (dark blue) to 13 (dark red). The times are from top to bottom: 1, 2 and 3 hours after the fault rupture.

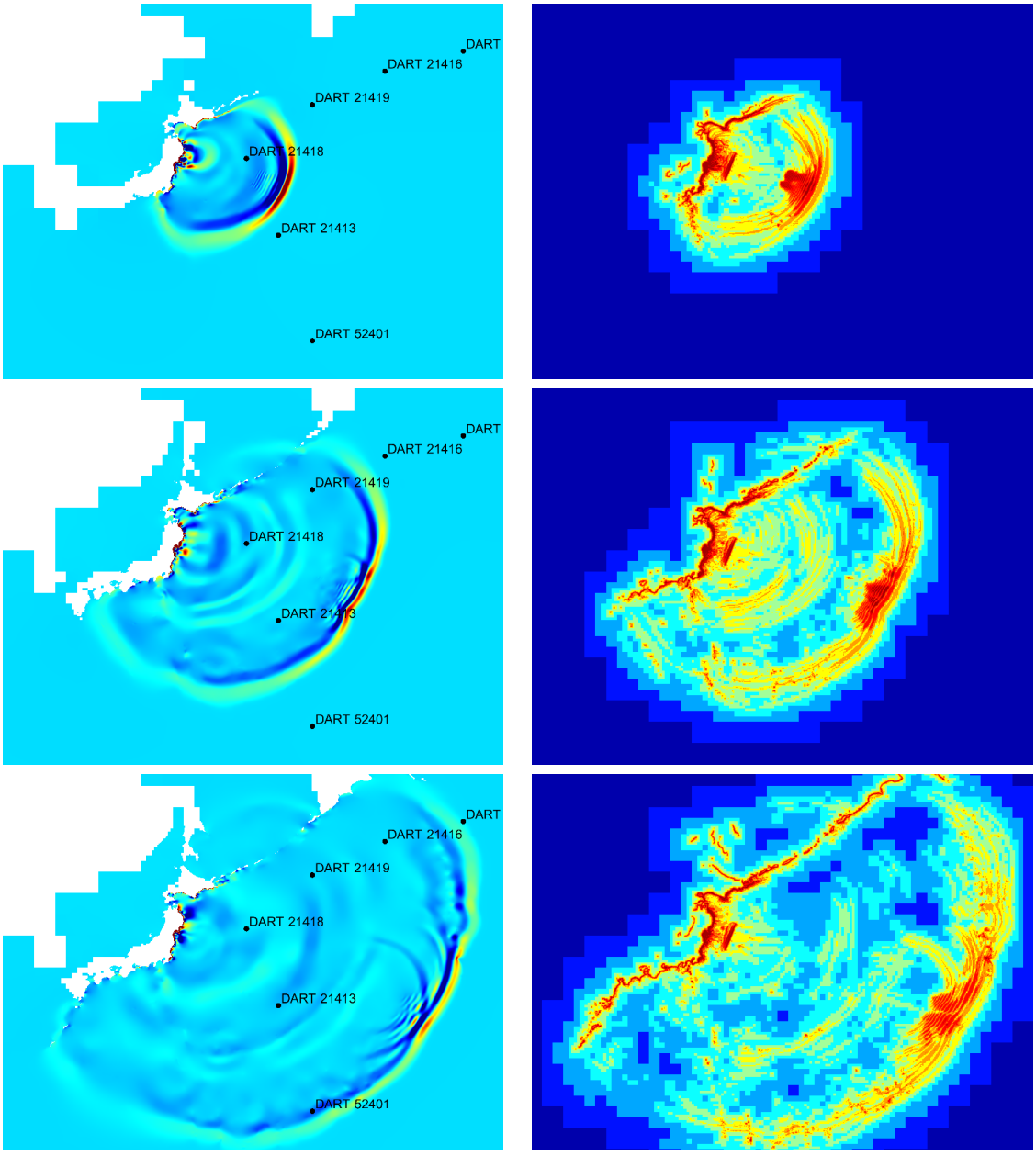


Figure 12: Tohoku tsunami modelled with the Serre–Green–Naghdi equations. The parameters and scales are identical to those of Figure 11.

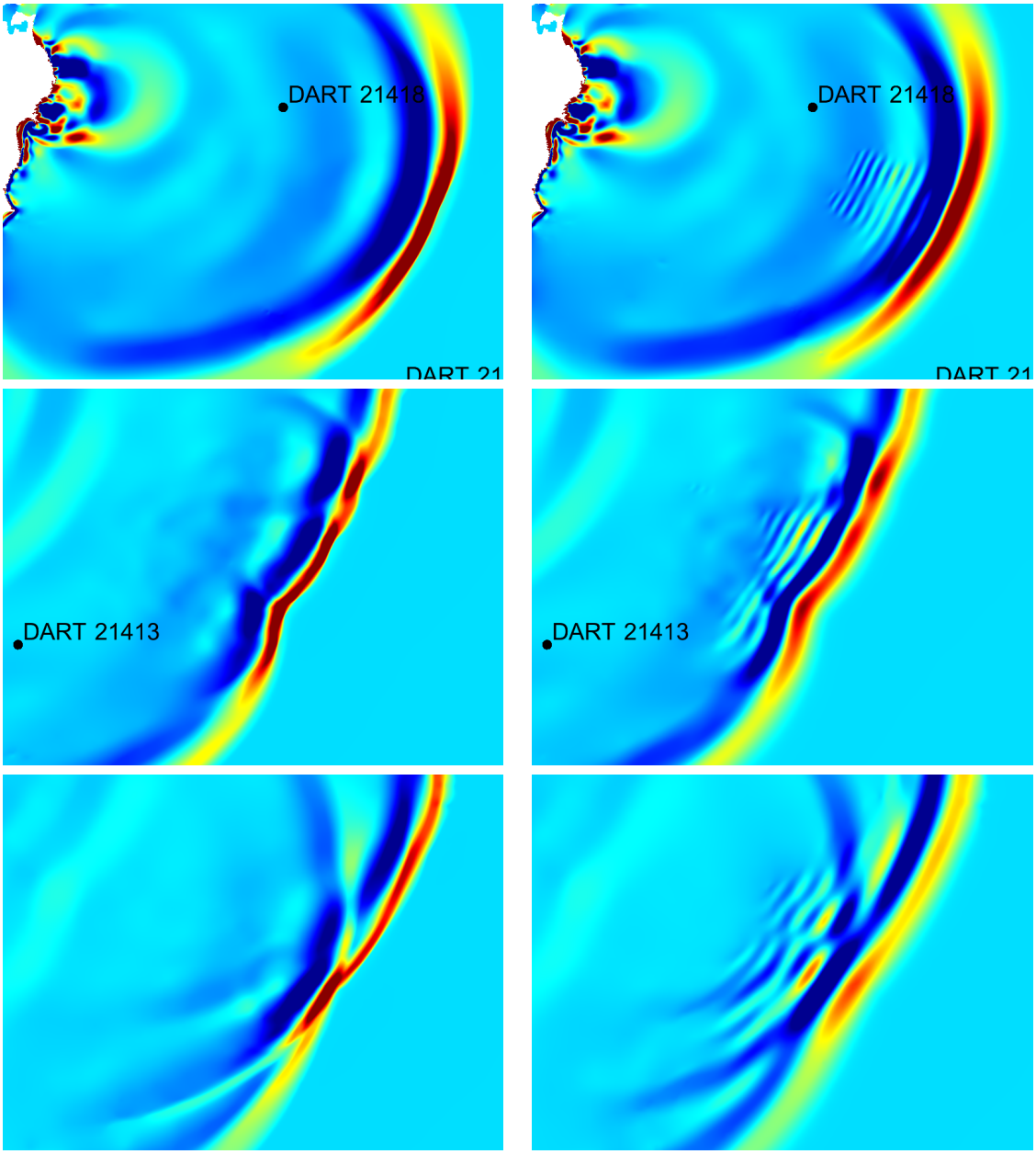


Figure 13: Close-up views of the leading wave front for the times of Figures 11 and 12. Left-column: Saint-Venant; right-column: Serre–Green–Naghdi.

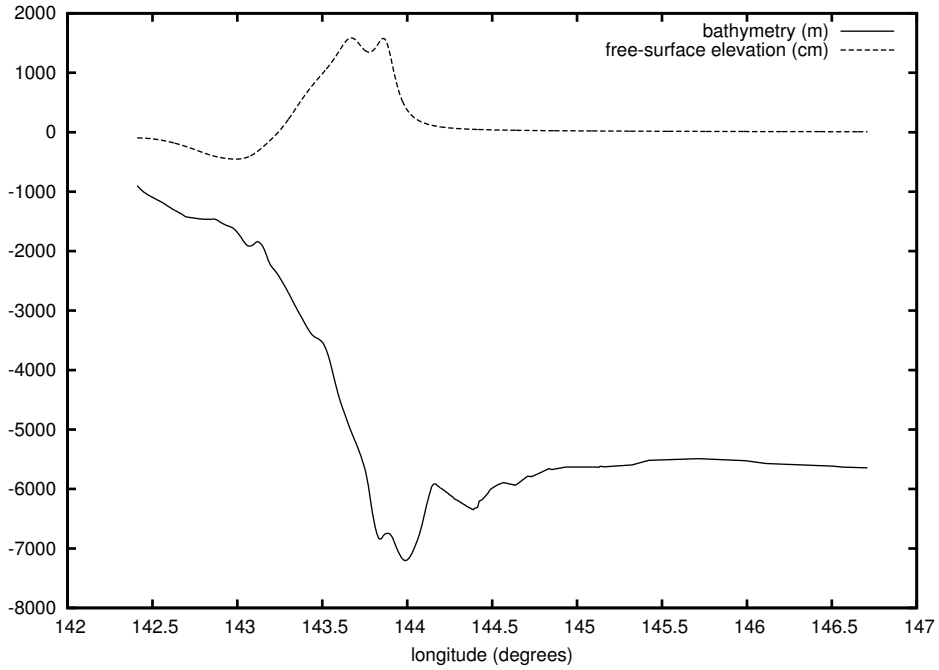


Figure 14: Vertical cross-section through the initial free-surface elevation and bathymetry, close to the maximum amplitude of the seismic source. Note the different units for elevation and bathymetry.

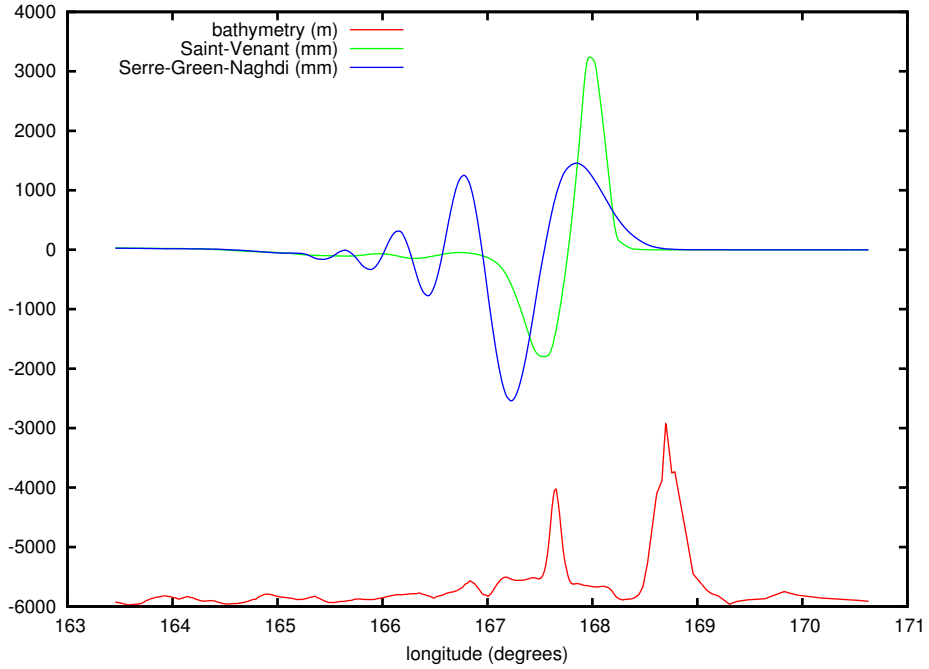


Figure 15: Vertical cross-section through the leading wave front at $t = 3$ hours (see also the last row of Figure 13). Both the Saint-Venant and SGN solutions for the free-surface elevation are shown. Note the different units for elevation and bathymetry.