Development of a Coastal Inundation Model using a Triangular Discontinuous Galerkin Method

Shiva Gopalakrishnan

Department of Applied Mathematics
Naval Postgraduate School, Monterey, California

7th August 2012

Motivation

Numerical modeling of tsunamis

Image credit: ©Anders Garwin

Motivation

Numerical modeling of stormsurges

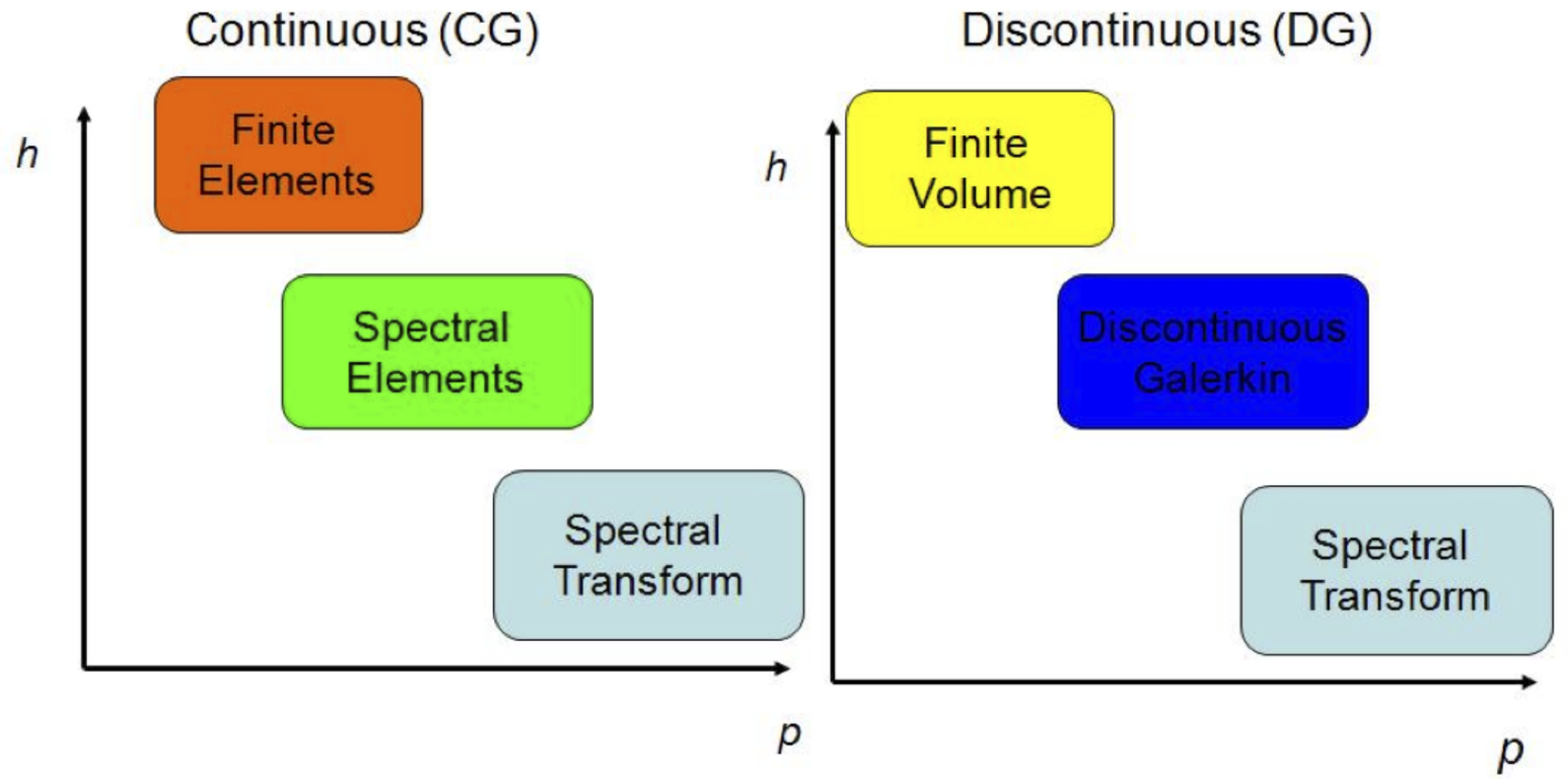
Image credit: ©National Weather Service, Houston

Outline

- Discontinuous Galerkin Method applied to SWE
- Coastal Ocean Modeling
- Numerical Tests
- Tsunami Simulations
- Storm–surge Simulations
- Parallelization

Element Based Galerkin methods

- All EBG methods partition the domain into computational elements and then approximate a function via basis functions.
- Examples: Finite Element, Spectral Elements, Finite Volume, Discontinuous Galerkin.



Shallow Water Equations

$$\frac{\partial q}{\partial t} + \nabla \cdot F(q) = S(q), \text{where } \; q = (\phi, U)^T$$

$$F(q) = \begin{pmatrix} U \\ \frac{U \otimes U}{\phi} + \frac{1}{2}\left(\phi^2 - \phi_b^2\right) I_2 \end{pmatrix}$$

$$S(q) = -\begin{pmatrix} 0 \\ f(k \times U) - \phi_s \nabla \phi_b - \frac{\tau}{\rho H} + \gamma U \end{pmatrix}$$

where,

$$\phi = g(h_s + h_b)$$

$$U = \phi \bar{u}$$

h_s- free surface height, h_b- bathymetry
$g-$ gravitational acceleration
$f = f_0 + \beta(y - y_m) -$ Coriolis parameter
$\tau-$ wind stress, $\gamma-$ bottom friction

Discontinuous Galerkin Method

The domain Ω is decomposed into N_e conforming elements.

$$\Omega = \bigcup_{e=1}^{N_e} \Omega_e$$

For the operators, a non−singular mapping

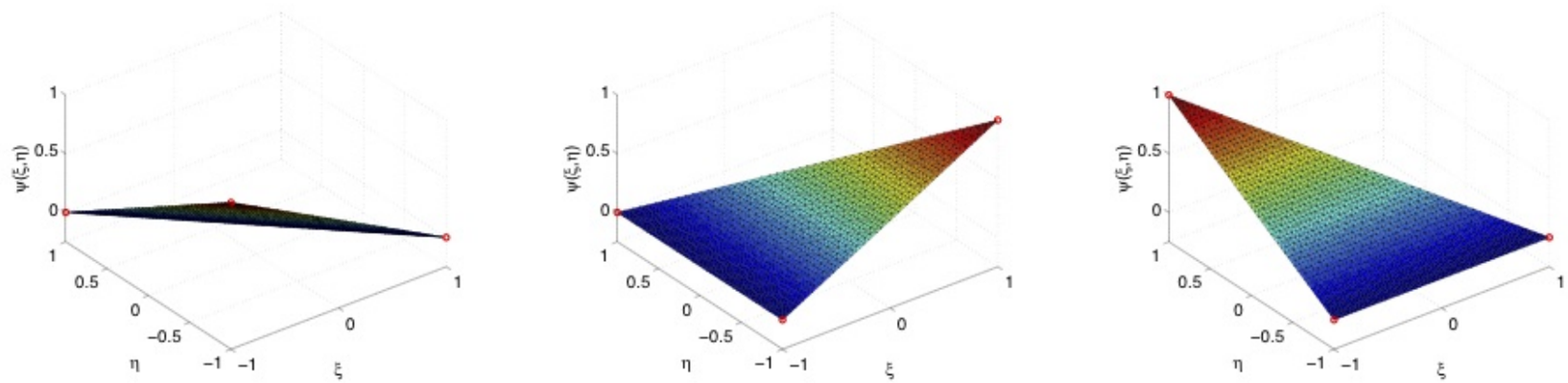
$$x = \Psi(\xi)$$

transforms the physical coordinate system $X = (x, y)^T$ to local reference coordinate system $\xi = (\xi, \eta)^T$. The local elementwise solution is approximated by Nth order polynomial in ξ by

$$q_N(\xi) = \sum_{i=1}^{M_n} \psi_i(\xi)\, q_N(\xi_i)$$

where $M_n = \frac{1}{2}(N+1)(N+2)$ is the number of interpolation

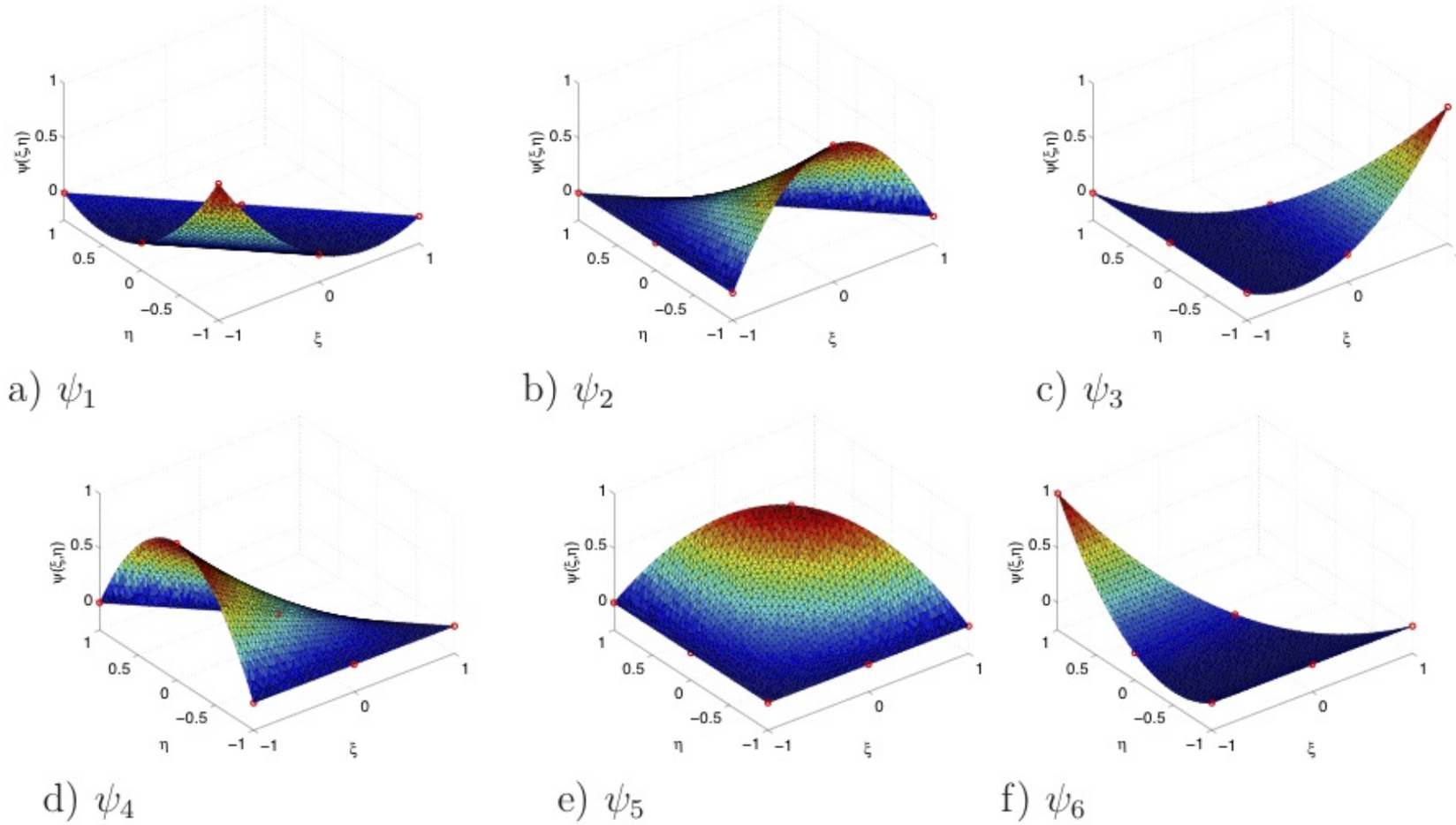
Triangular basis functions



a) ψ_1 b) ψ_2 c) ψ_3

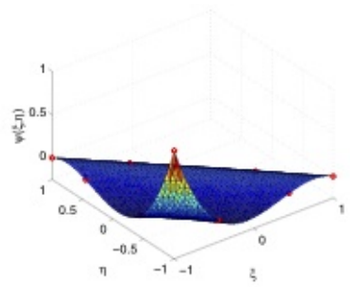
Triangular basis functions of order N=1 at 3 interpolation points.

Triangular basis functions

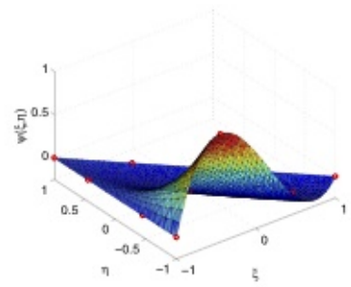


Triangular basis functions of order N=2 at 6 interpolation points.

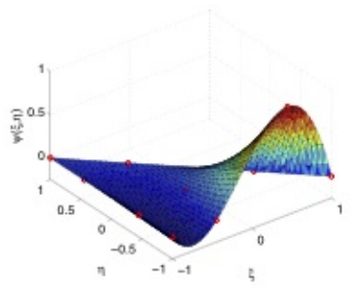
Triangular basis functions



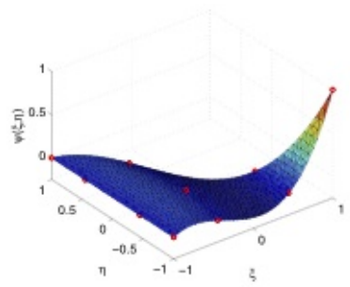
a) ψ_1



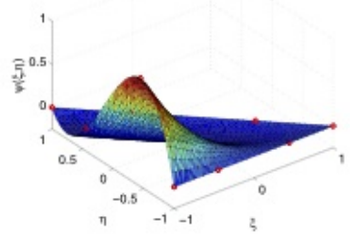
b) ψ_2



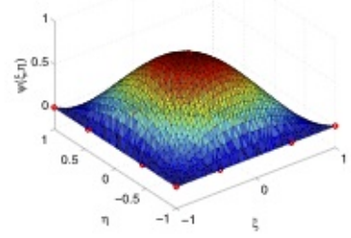
c) ψ_3



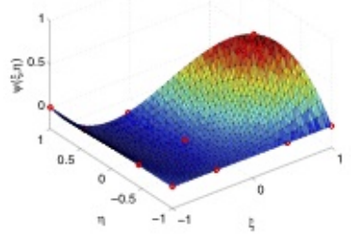
d) ψ_4



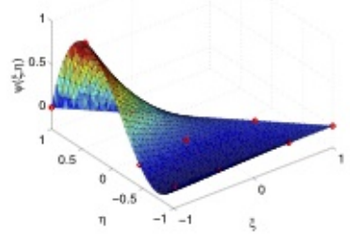
e) ψ_5



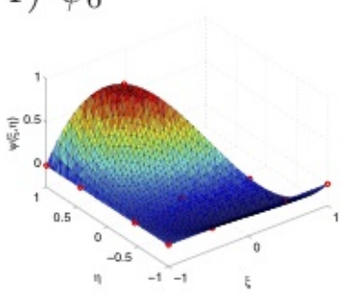
f) ψ_6



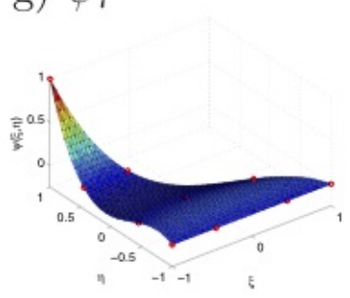
g) ψ_7



h) ψ_8



i) ψ_9



j) ψ_{10}

Triangular basis functions of order N=3 at 10 interpolation points.

Discontinuous Galerkin Method

Applying DG to the Shallow water equations to obtain the weak form

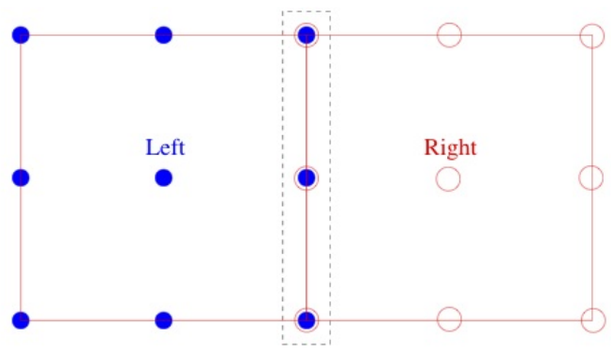
$$\int_{\Omega_e} \left(\frac{\partial q_N^{(e)}}{\partial t} - F_N^{(e)} \cdot \nabla - S_N^{(e)} \right) \psi_i(x)\, dx$$

$$= -\sum_{l=1}^{3} \int_{\Gamma_e} \psi_i(x)\, n^{(e,l)} \cdot F_N^{(*,l)} dx$$

Rusanov Numerical Flux

$$F_N^{(*,l)} = \frac{1}{2} \left[F_N\left(q_N^{(e)}\right) + F_N\left(q_N^{(l)}\right) - |\lambda^{(l)}| \left(q_N^{(l)} - q_N^{(e)}\right) n^{(e,l)} \right]$$

Discontinuous Galerkin Method



Rusanov Numerical Flux

$$F_N^{(*,I)} = \frac{1}{2}\left[F_N\left(q_N^{(e)}\right) + F_N\left(q_N^{(I)}\right) - |\lambda^{(I)}|\left(q_N^{(I)} - q_N^{(e)}\right) n^{(e,I)}\right]$$

Where,

$$\lambda^{(I)} = max\left(|U^{(e)}| + \sqrt{\phi^{(e)}}, |U^{(I)}| + \sqrt{\phi^{(I)}}\right)$$

with,

$$U^{(e,I)} = u^{(e,I)} \cdot n^{(I)}$$

Matrix form of semi−discrete equations

Using the polynomial approximation $q_N = \sum_{i=1}^{M_N} \psi_i q_i$

$$\int_{\Omega_e} \psi_i \psi_j dx \frac{\partial q^{(e)}}{\partial t} - F_j^{(e)} \cdot \int_{\Omega_e} \nabla \psi_i \psi_j dx - \int_{\Omega_e} \psi_i \psi_j dx S_j^{(e)}$$

$$= -\sum_{l=1}^{3} \int_{\Gamma_e} \psi_i \psi_j n^{(e,l)} dx \cdot (F^{(*,l)})_j$$

Defining element matrices as

$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j dx, \quad M_{ij}^{(e,l)} = \int_{\Gamma_e} \psi_i \psi_j n^{(e,l)} dx, \quad D_{ij}^{(e)} = \int_{\Omega_e} \nabla \psi_i \psi_j dx$$

Matrix form of semi−discrete equations

$$M_{ij}^{(e)} \frac{\partial q^{(e)}}{\partial t} - (D_{ij}^{(e)})^{\mathcal{T}} F_j^{(e)} - M_{ij}^{(e)} S_j^{(e)} = -\sum_{l=1}^{3} (M_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j$$

Eliminating mass matrix on LHS

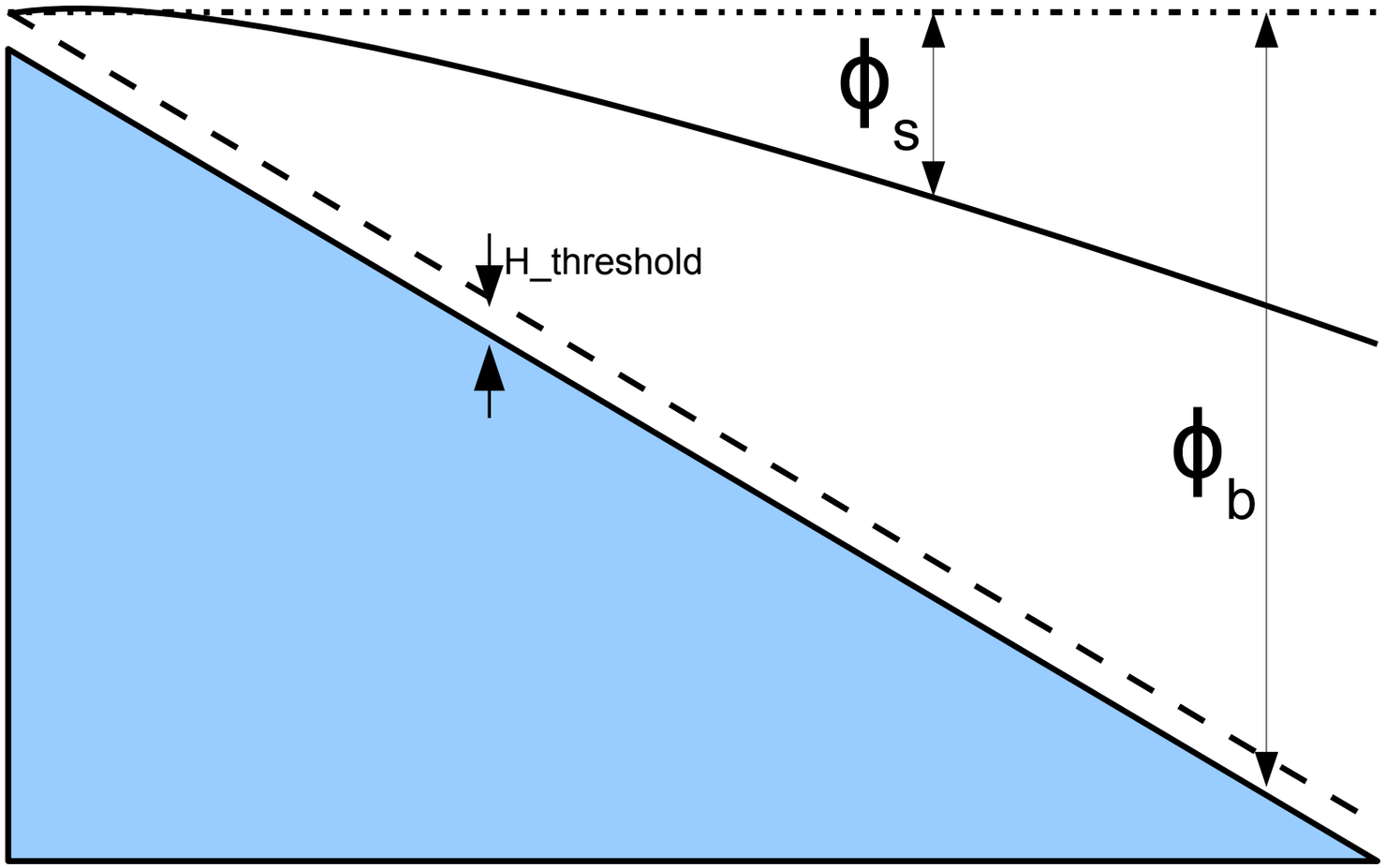
$$\widehat{D}^{(e)} = (M^{(e)})^{-1} D^{(e)}, \quad \widehat{M}^{(e,l)} = (M^{(e,l)})^{-1} M^{(e,l)}$$

$$\frac{\partial q^{(e)}}{\partial t} - (\widehat{D_{ij}}^{(e)})^{\mathcal{T}} F_j^{(e)} - S_j^{(e)} = -\sum_{l=1}^{3} (\widehat{M}_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j$$

Coastal ocean modeling

- The shoreline is represented as a moving boundary condition where $\phi = \phi_s + \phi_b = 0$
- Moving front is described as $x = x_b + \int v_b dt$.
- Where x_b is initial position and v_b the velocity of the front.
- Approaches used to model the wetting and drying of land.
  - Fixed grid methods. – easier to implement. Additional algorithms required to maintain depth positivity.
  - Moving grid methods. – traditionally perceived as cumbersome. (Lynch and Gray 1978)

Wetting and Drying Algorithm



Wetting and Drying Algorithm – based on Gourgue et al 2009

Conservation of Mass

$$\frac{\partial \phi_s}{\partial t} = -F(U)$$

where $\phi = \phi_s + \phi_b$, and operator $F(U) \equiv F(\phi_s, \bar{u})$

- Step 1 – limit ϕ to a threshold value.

$$\phi_s^* = max(\phi_s^n, H_{threshold} - \phi_b)$$

- Step 2

$$\frac{\phi_s^{**}}{\Delta t} = -F(\phi_s^*, \bar{u})$$

- Step 3 – ensure free surface does not move to dry areas.

$$\frac{\phi_s^{n+1}}{\Delta t} = -F^*(\phi_s^*, \bar{u})$$

Wetting and Drying Algorithm

Conservation of Mass in matrix form

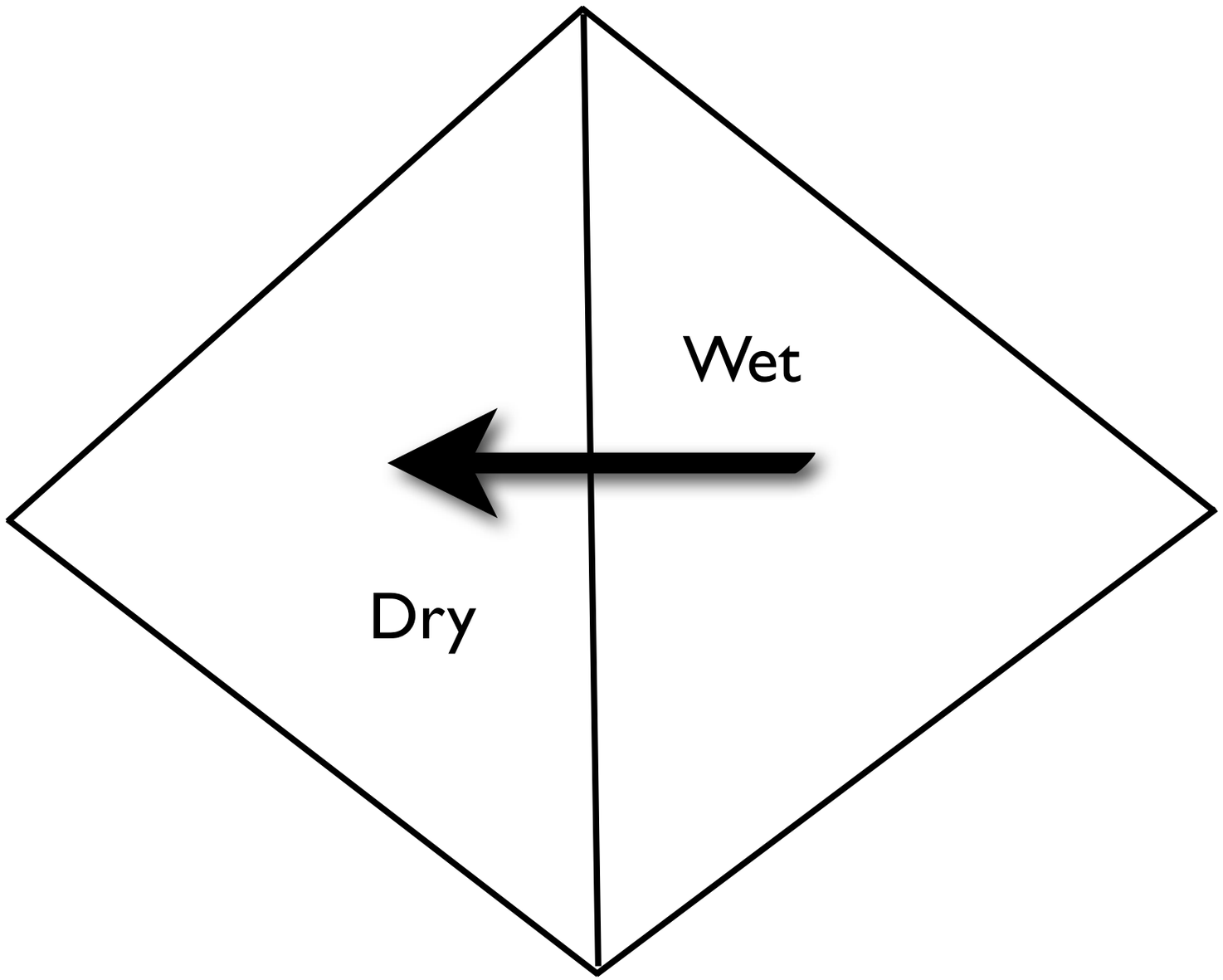
$$\frac{\partial q^{(e)}}{\partial t} = (\widehat{D_{ij}}^{(e)})^{\mathcal{T}} F_j^{(e)} + S_j^{(e)} - \sum_{l=1}^{3} (\widehat{M}_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j$$

Let,

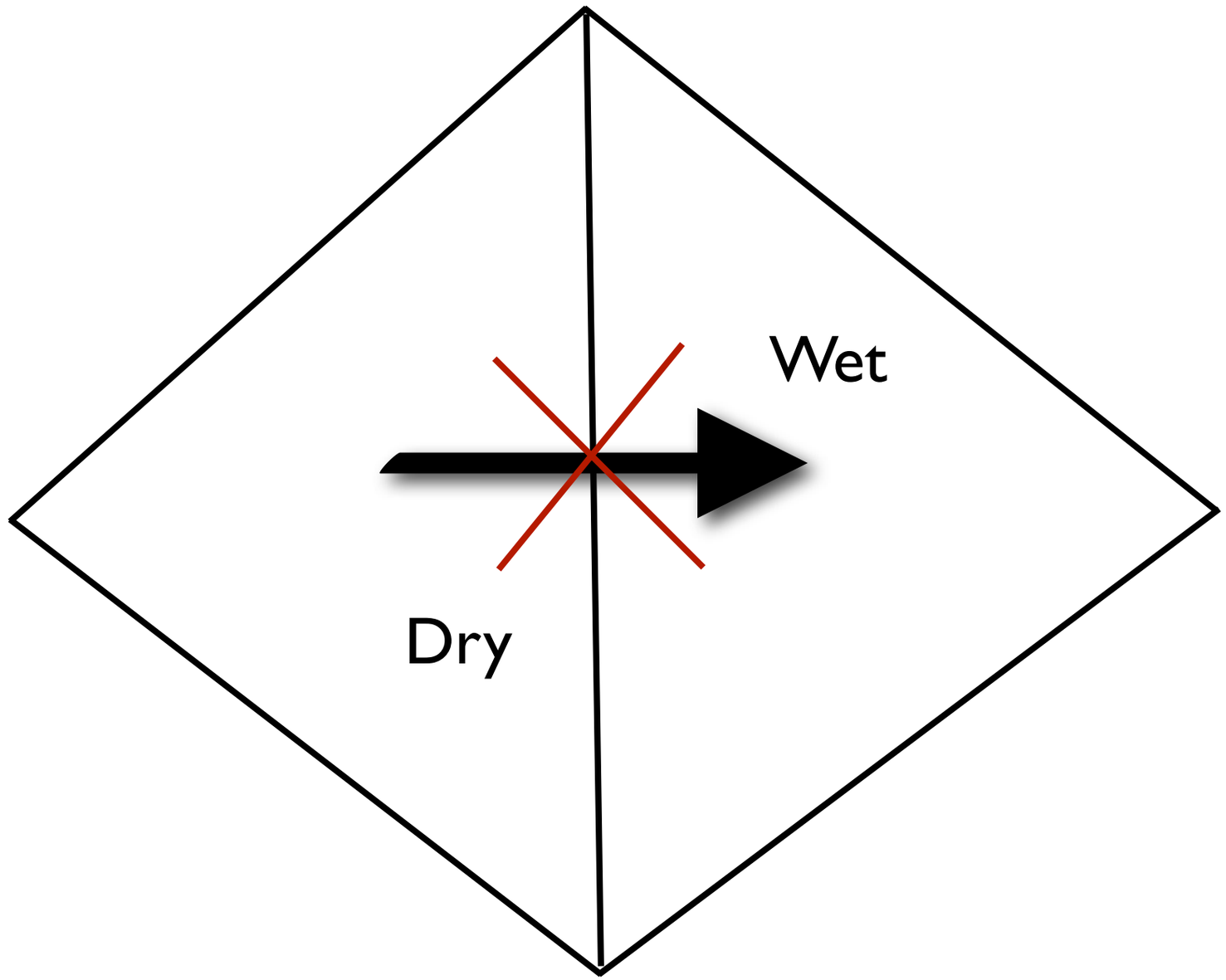
$$F_j^s(\phi_s^*, \bar{u}) = -\sum_{l=1}^{3} (\widehat{M}_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j$$

$$F_j^c(\phi_s^*, \bar{u}) = (\widehat{D_{ij}}^{(e)})^{\mathcal{T}} F_j^{(e)} + S_j^{(e)}$$

Wetting and Drying Algorithm



Wetting and Drying Algorithm



Wetting and Drying Algorithm

$$\frac{\phi_s^{n+1}}{\Delta t} = F_j^{c*}\left(\phi_s^*, \bar{u}\right) + F_j^{s*}\left(\phi_s^*, \bar{u}\right)$$

Where,

$$F_j^{c*} = \begin{cases} 0 \text{ if } F_c^j < 0 \;\&\; \phi^n < H_{threshold} \\ F_c^j \text{ otherwise} \end{cases}$$

$$F_j^{s*} = \begin{cases} 0 \text{ if there is a node } i \in \Omega_e \text{ with } F_s^j < 0 \;\&\; \phi < H_{threshold} \\ F_s^j \text{ otherwise} \end{cases}$$

*limited to using linear elements.

Steady state test

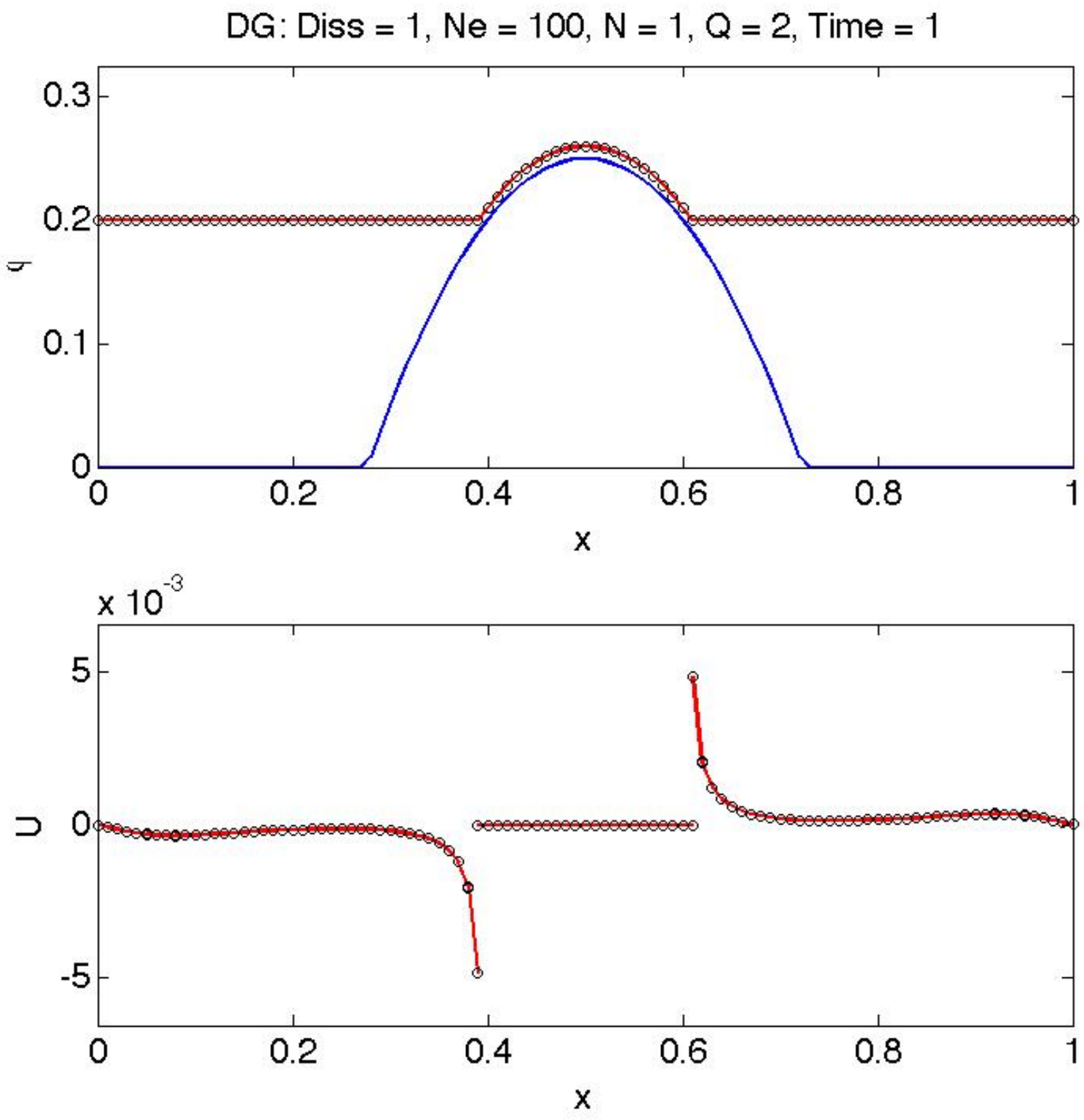
Is the model well balanced ?
Bottom topography is defined as,

$$h_b(x) = max\left(0, 0.25 - 5\left(x - 0.5^2\right)\right), 0 \leq \mathrm{x} \quad eq\ 1$$

Initial condition

$$h_s + h_b = max\left(0.2, b\right)$$
$$\phi U = 0 \quad \text{over entire domain}$$

Steady state test case



Well Balanced Test

*wellBalancedTest.mp4

Balzano [1998] – Planar Beach

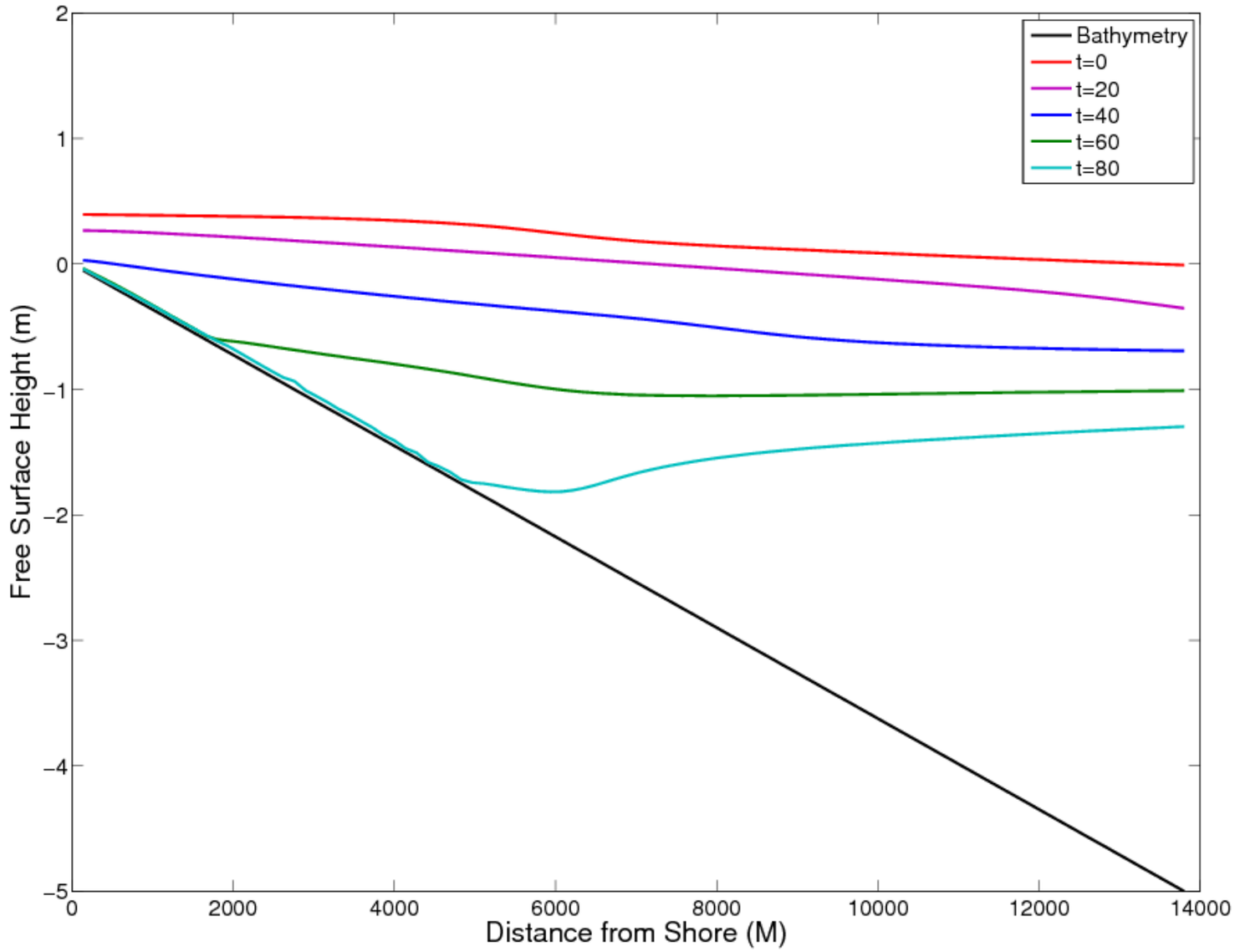
Bottom topography is defined as,

$$h_b(x) = \frac{x}{2760}$$

Domain size is 13,800 meters. Sinusoidal forcing at the open end is given by,

$$\phi_s = g * (2sin\left(\frac{2\pi t}{43200}\right))$$

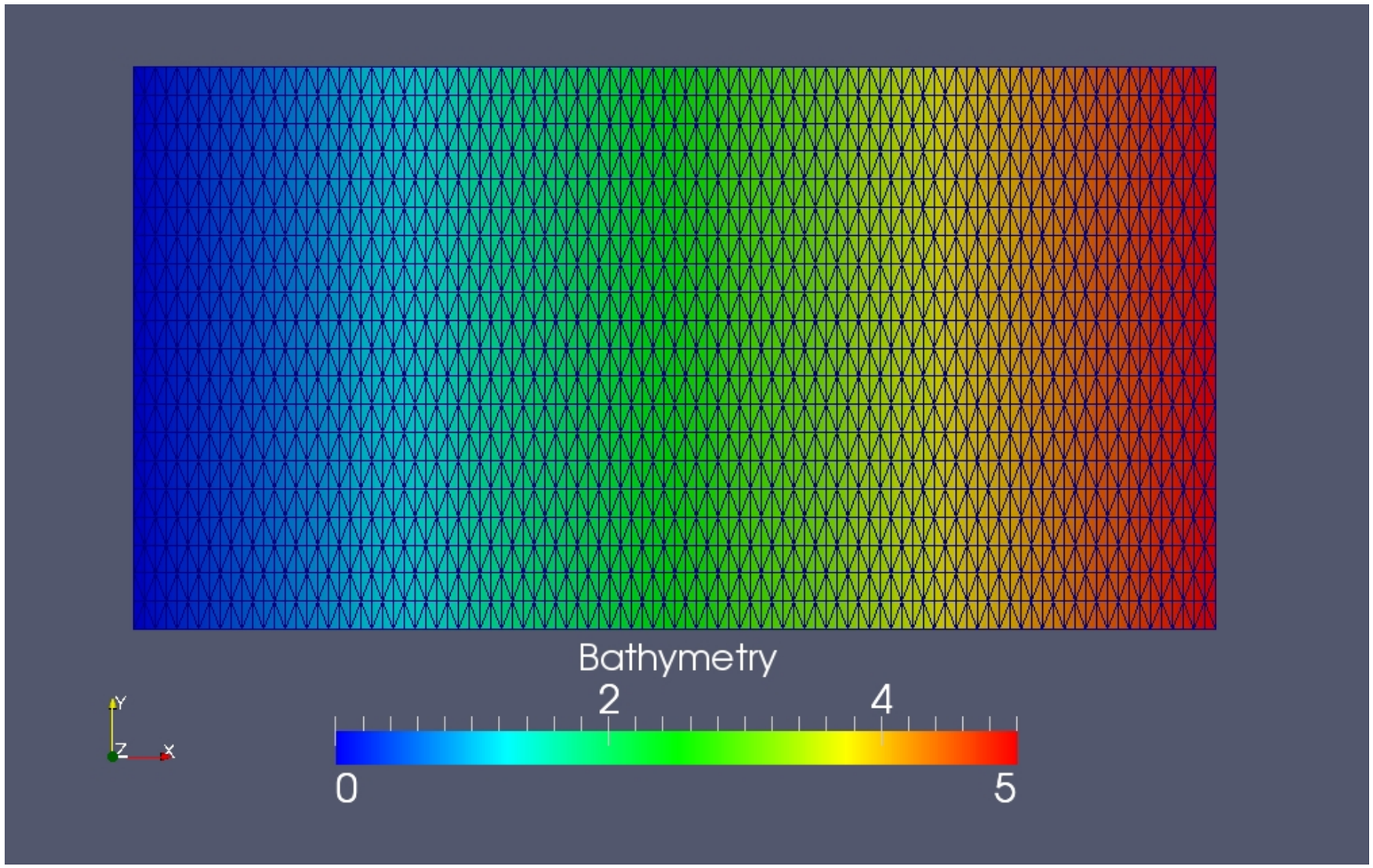
Balzano [1998] – Planar Beach

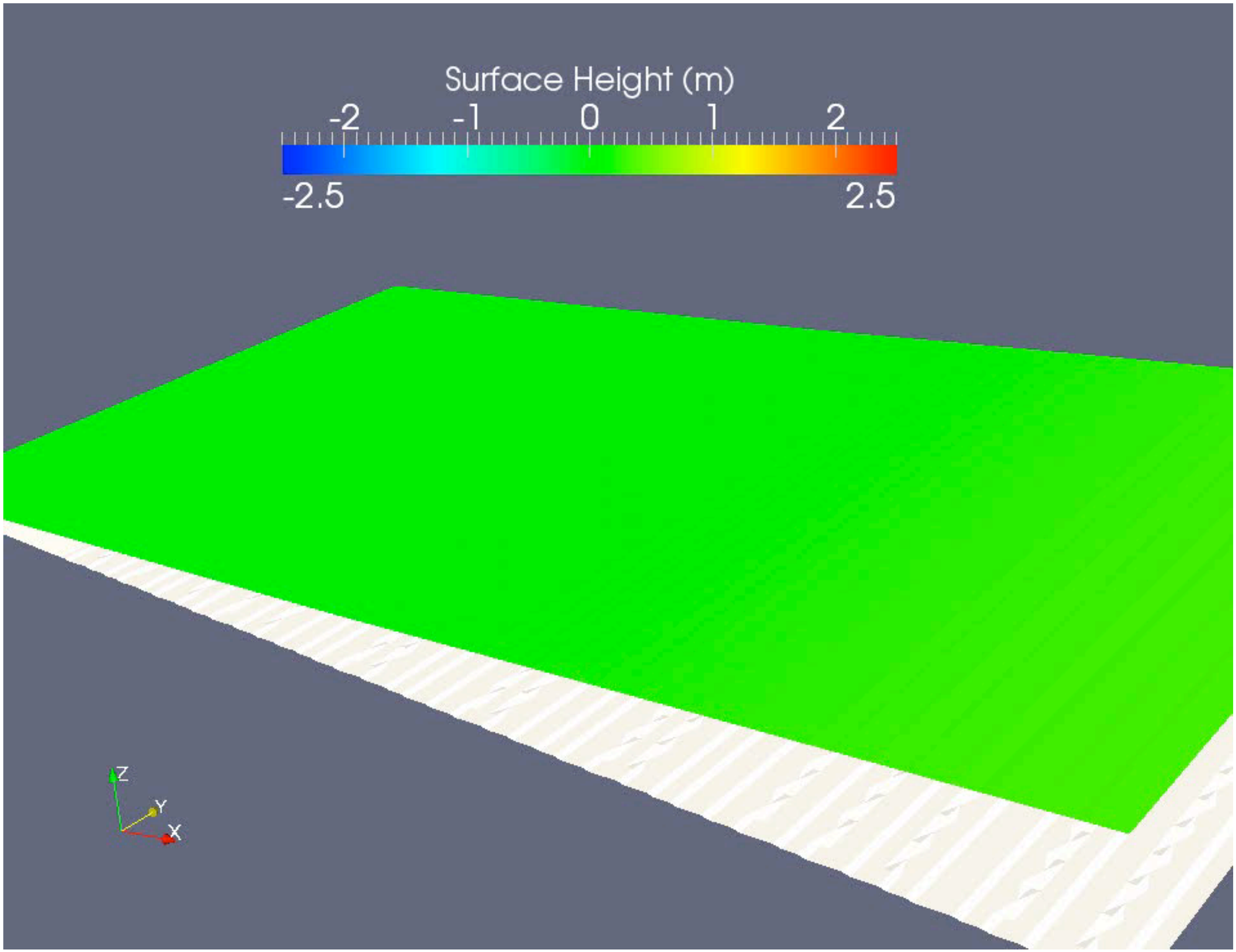


Planar Beach

*planarbeach1d.mp4

Balzano [1998] 2D Planar Beach




Surface Height (m)
-2
-1
0
1
2
-2.5
2.5
Z
Y
X

Balzano [1998] – Tide Pool

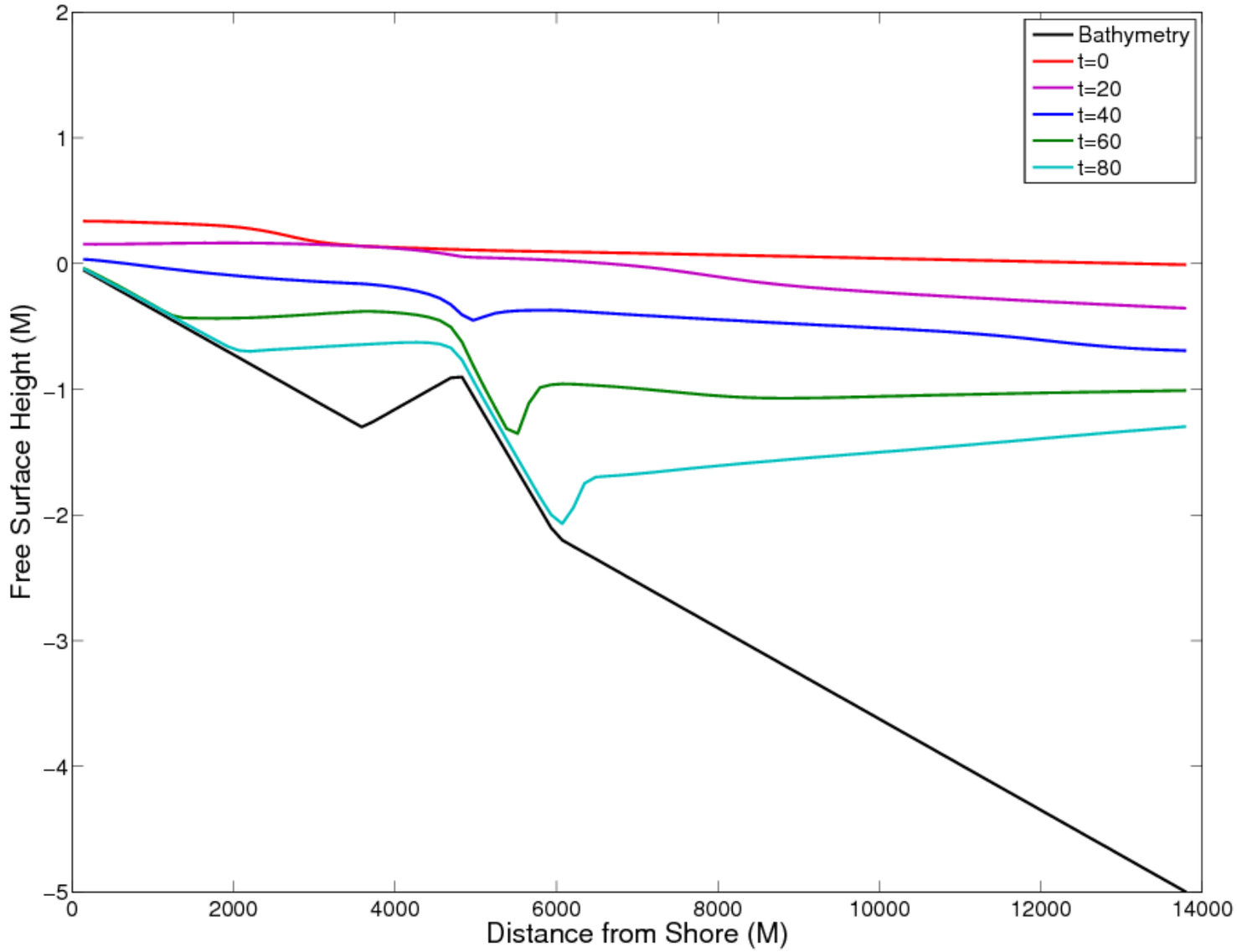
Bottom topography is defined as,

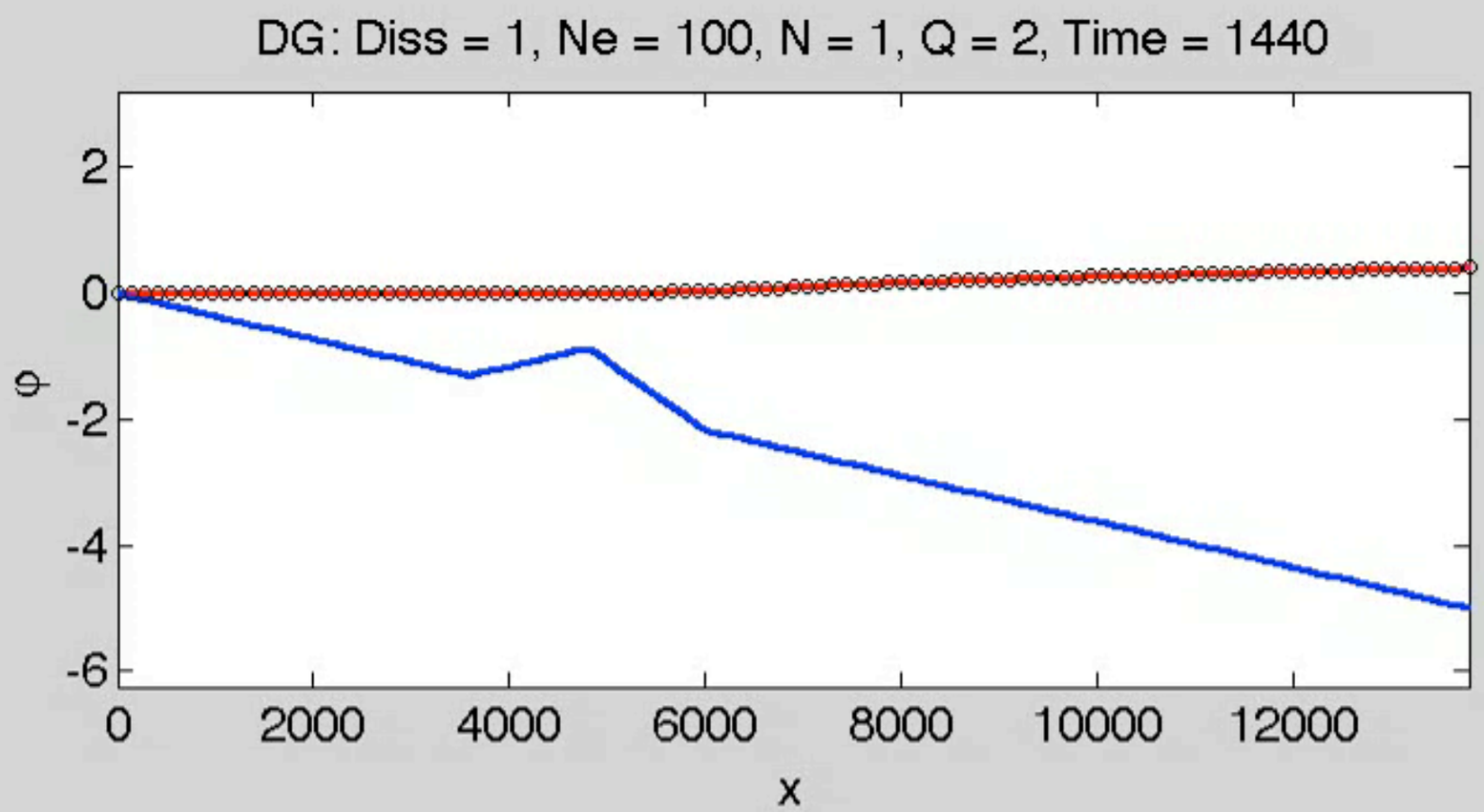
$$h_b(x) = \begin{cases} \frac{x}{2760} & \text{if x} \leq \text{3600m, or if x} \geq \text{6000m} \\ \frac{-x}{2760} + \frac{60}{23} & \text{if 3600m} \leq \text{x} \leq \text{4800m} \\ \frac{x}{920} - \frac{100}{23} & \text{if 4800m} \leq \text{x} \leq \text{6000m} \end{cases}$$

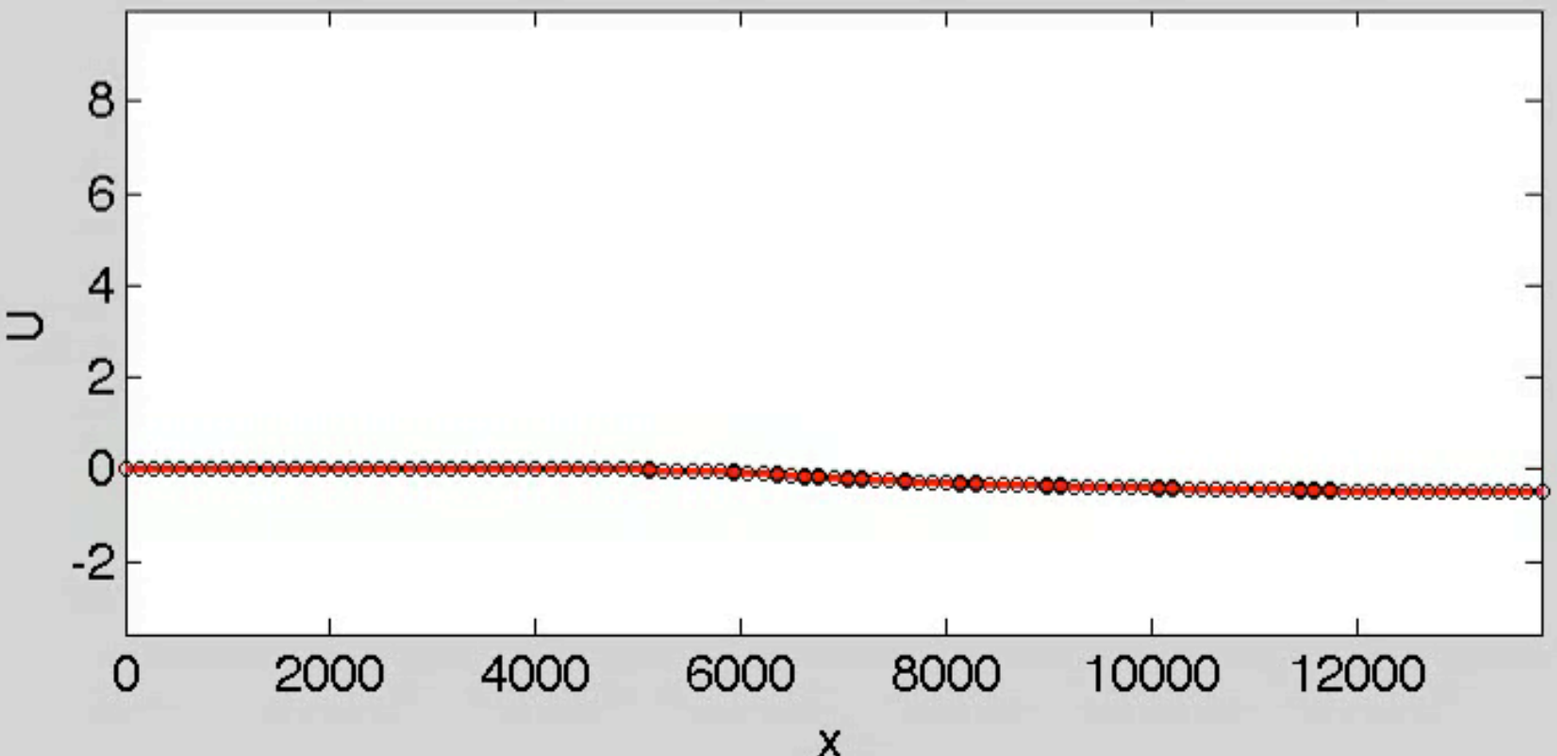
Domain size is 13,800 meters. Sinusoidal forcing at the open end is given by,

$$\phi_s = g * (2sin\left(\frac{2\pi t}{43200}\right))$$

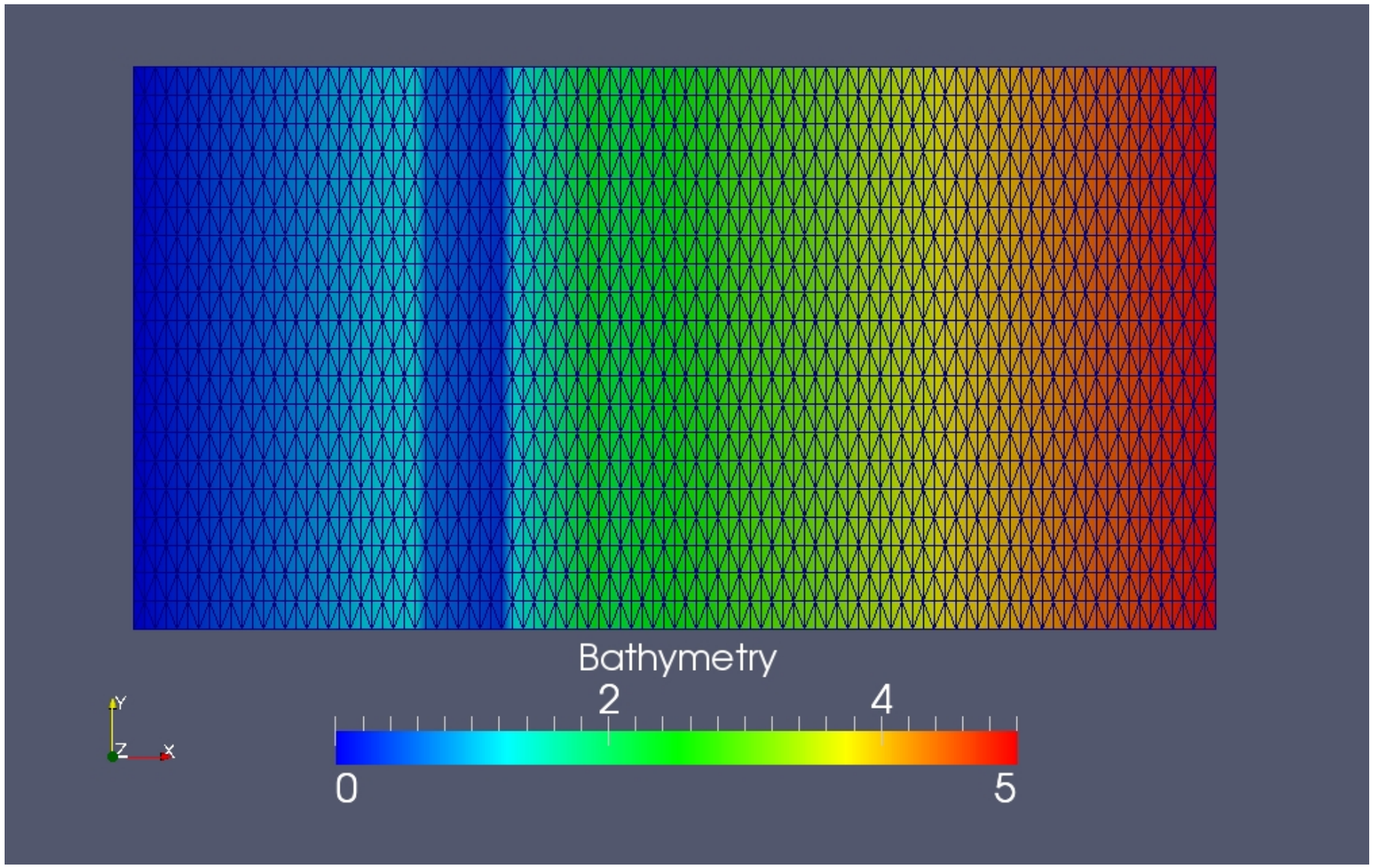
Balzano [1998] – Tide pool

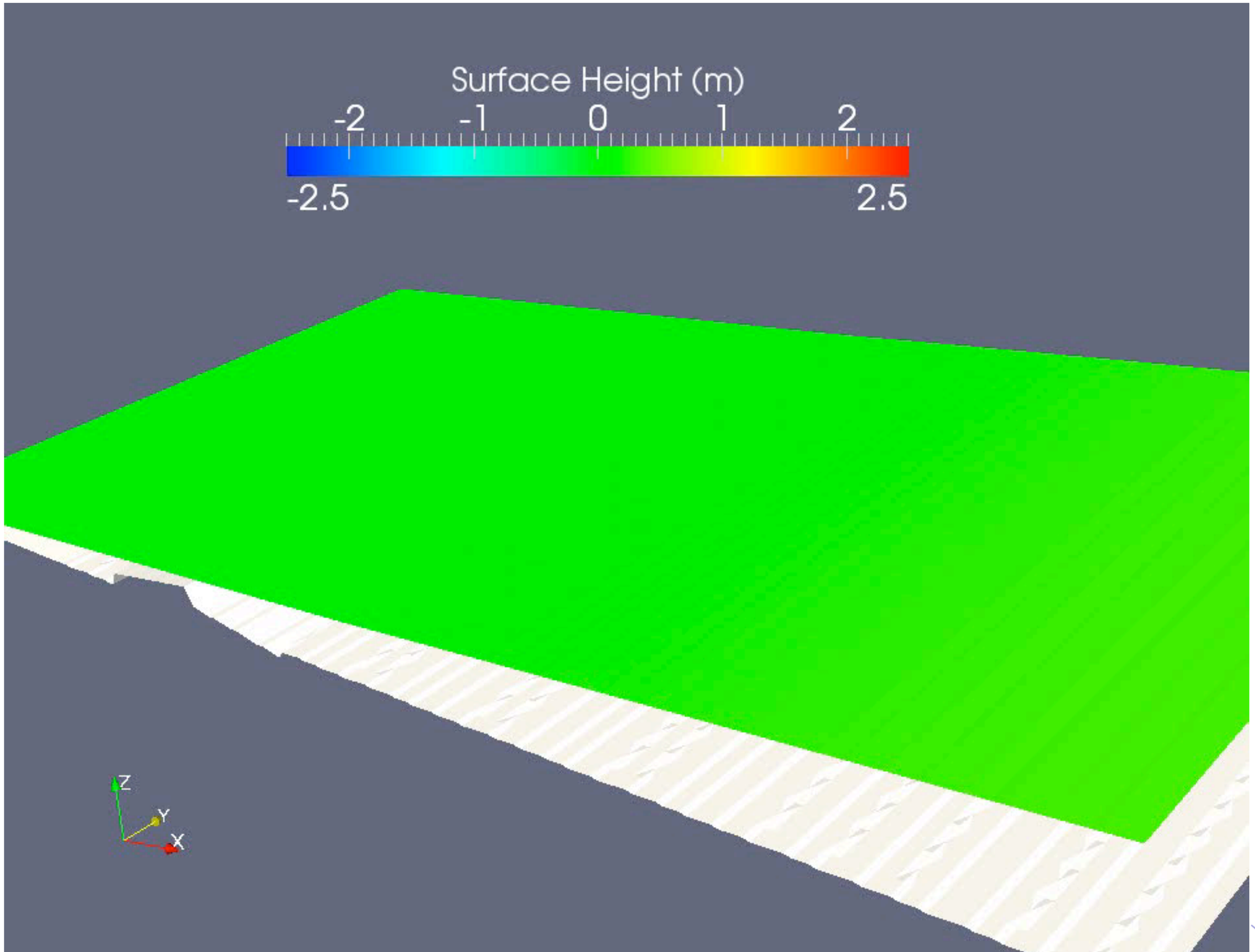



DG: Diss = 1, Ne = 100, N = 1, Q = 2, Time = 1440
2
0
-2
-4
-6
φ
0
2000
4000
6000
8000
10000
12000
x

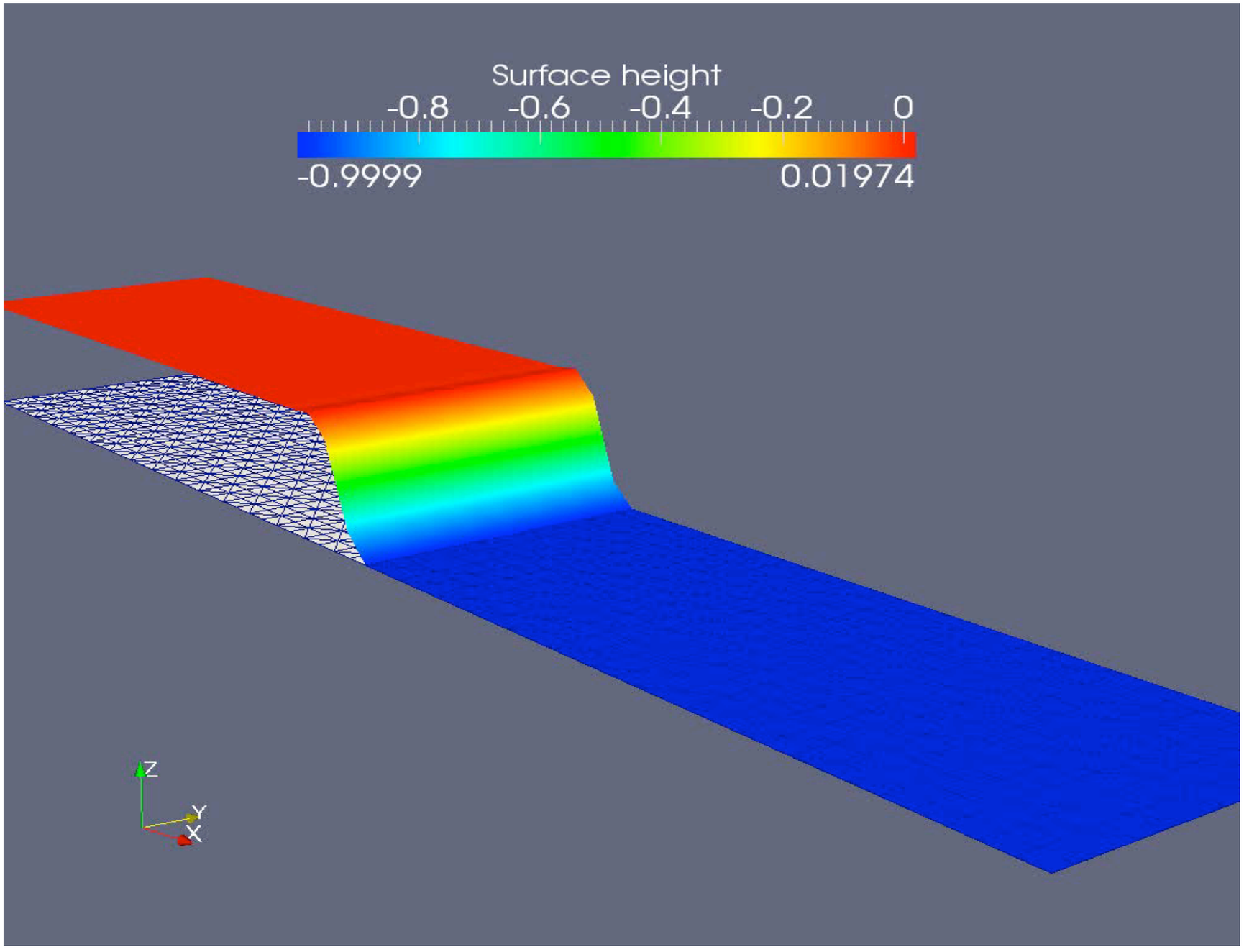

8
6
4
2
0
-2
U
0
2000
4000
6000
8000
10000
12000
x

Balzano [1998] 2D Tide Pool

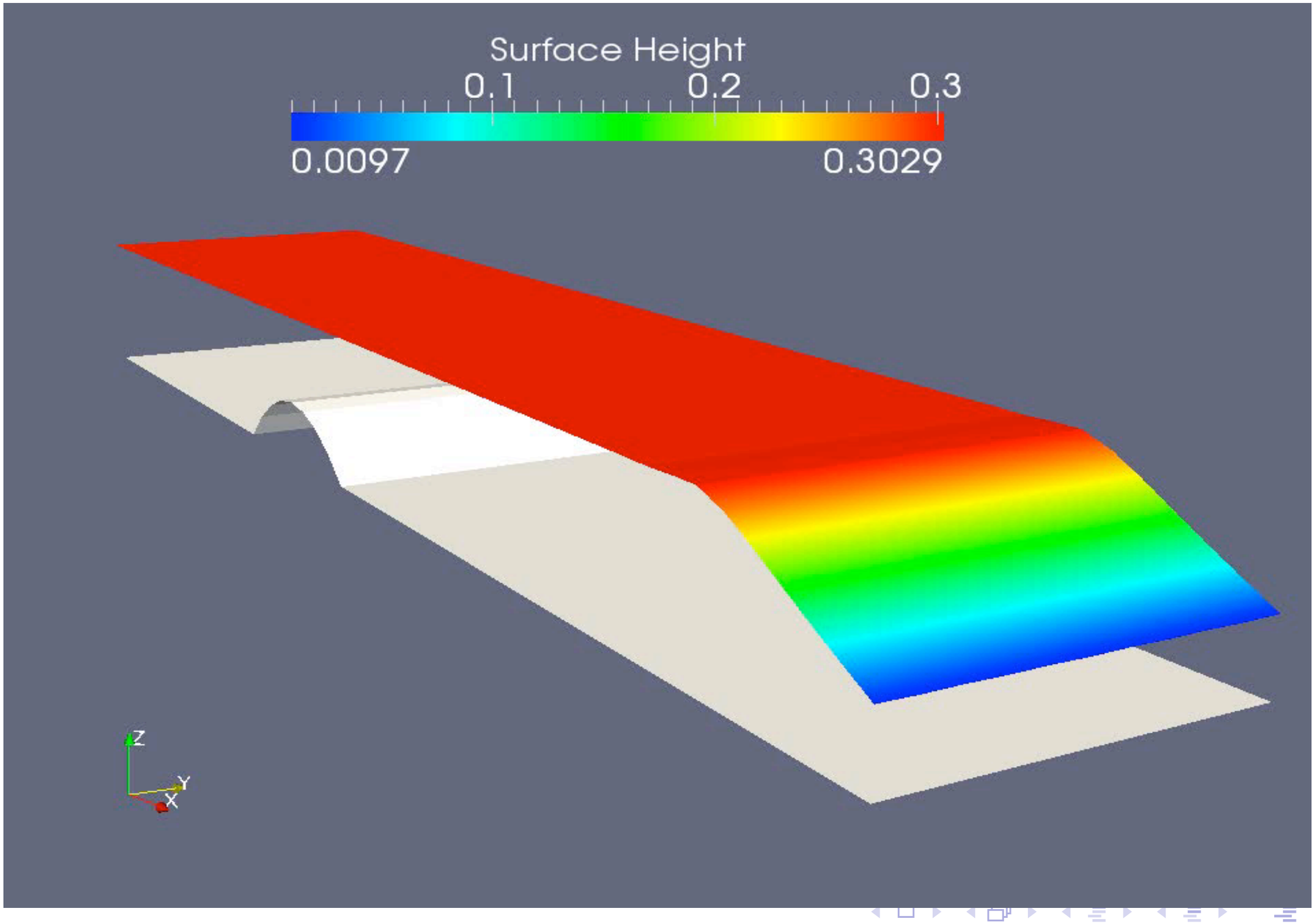



Surface Height (m)
-2
-1
0
1
2
-2.5
2.5
Z
Y
X

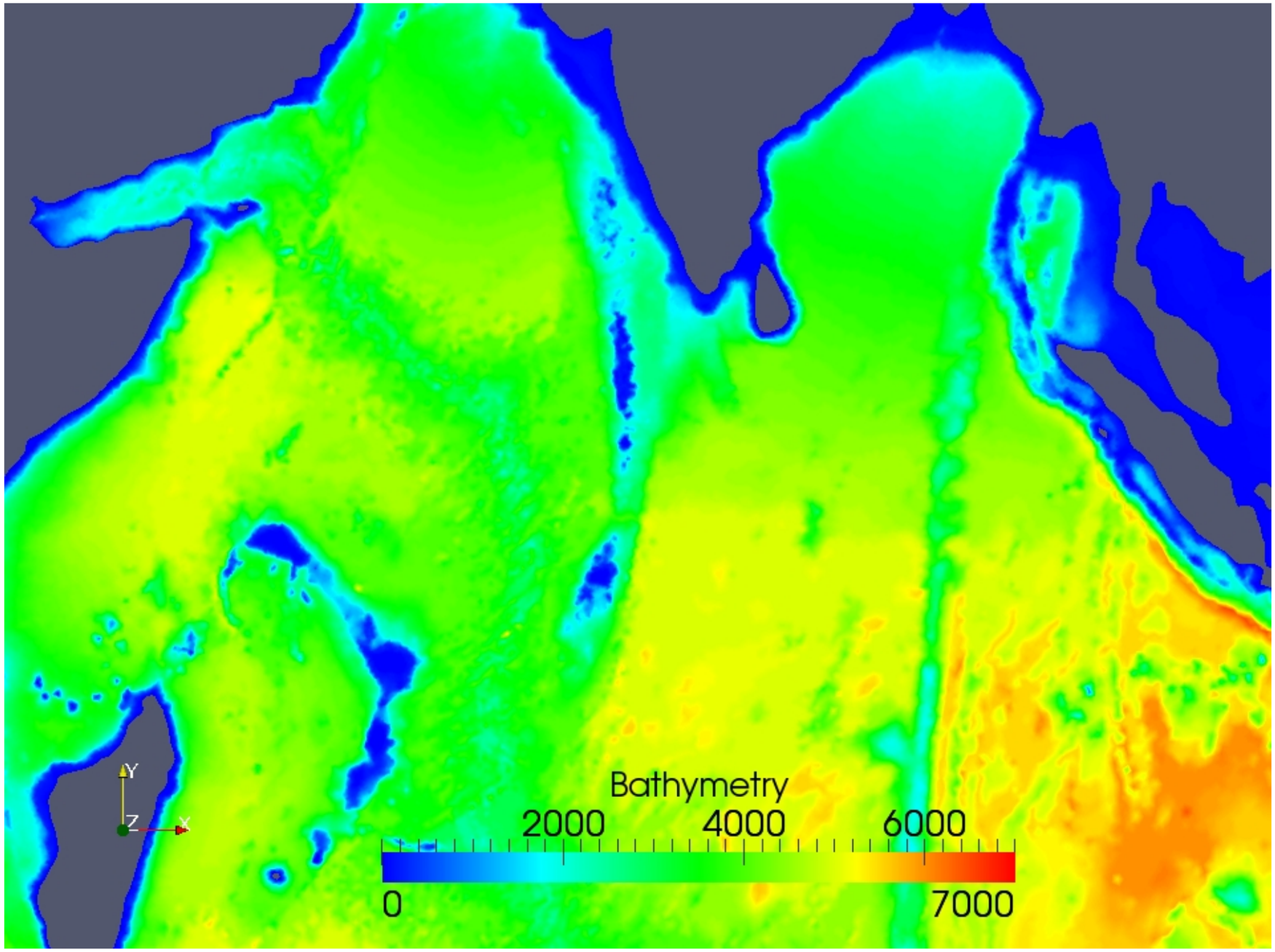
Dam Break



Drain with a non–flat bottom

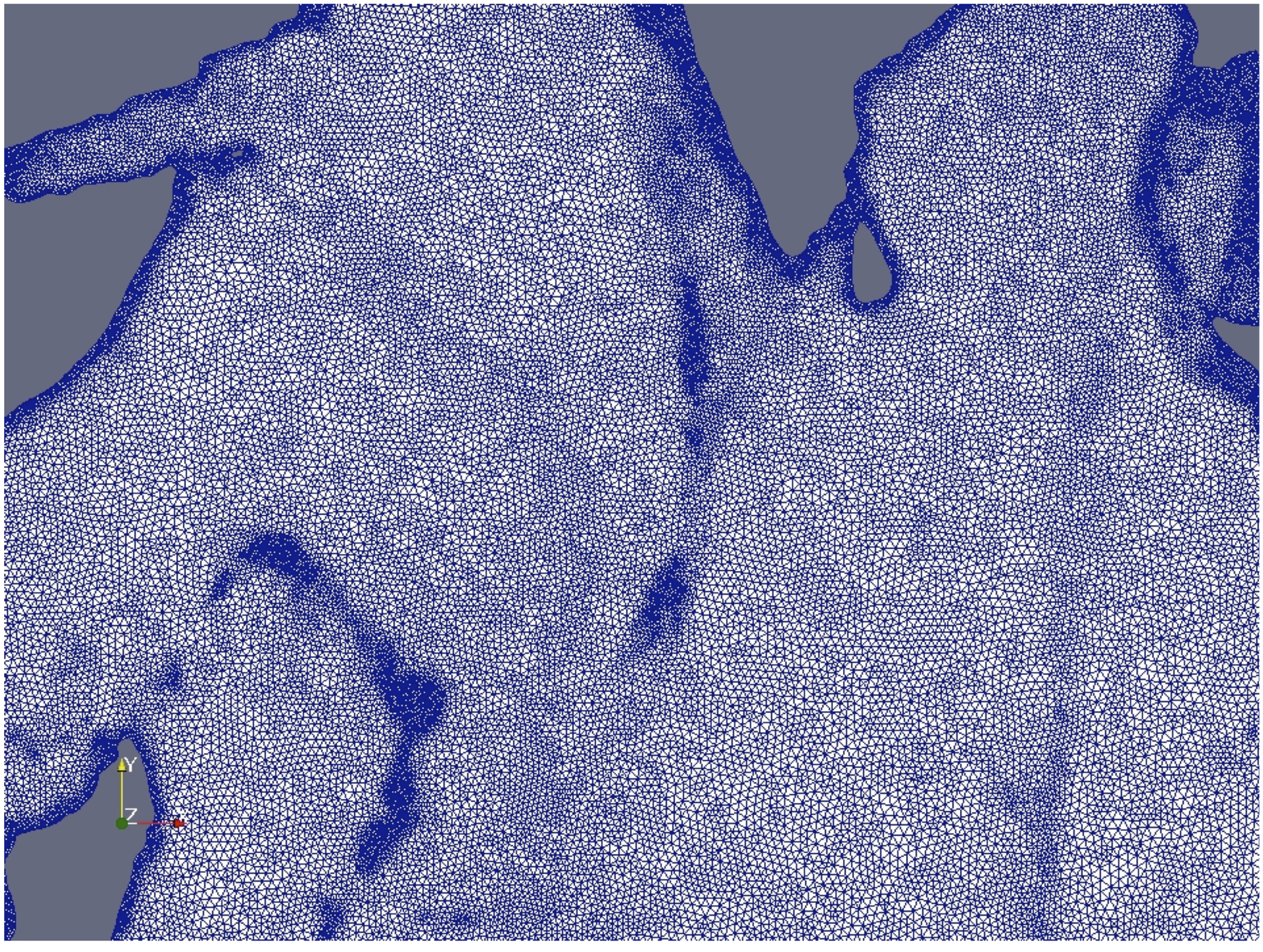


2004 Indian Ocean Tsunami



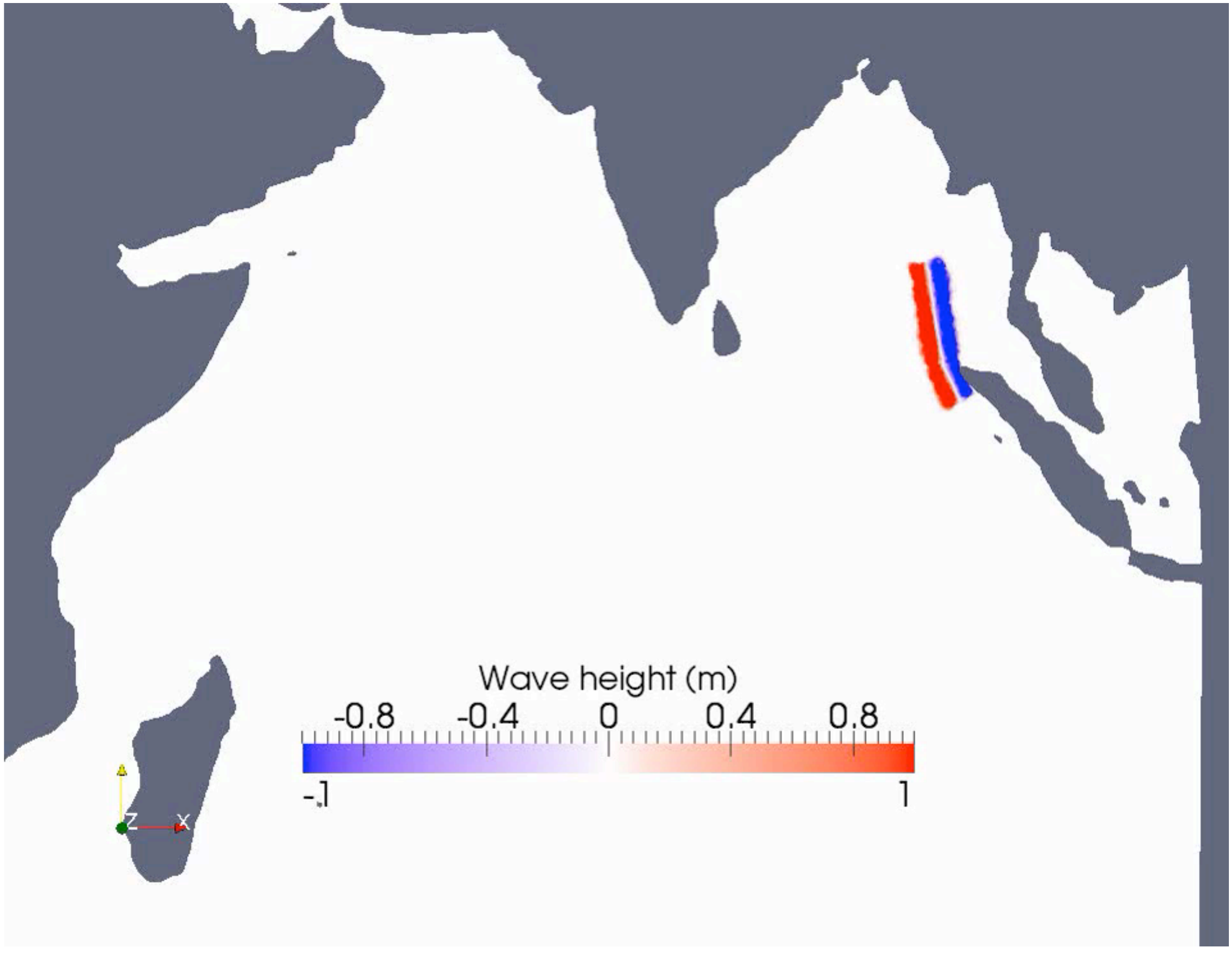
Ref: Data and Grid from Joern Behrens

2004 Indian Ocean Tsunami – Grid



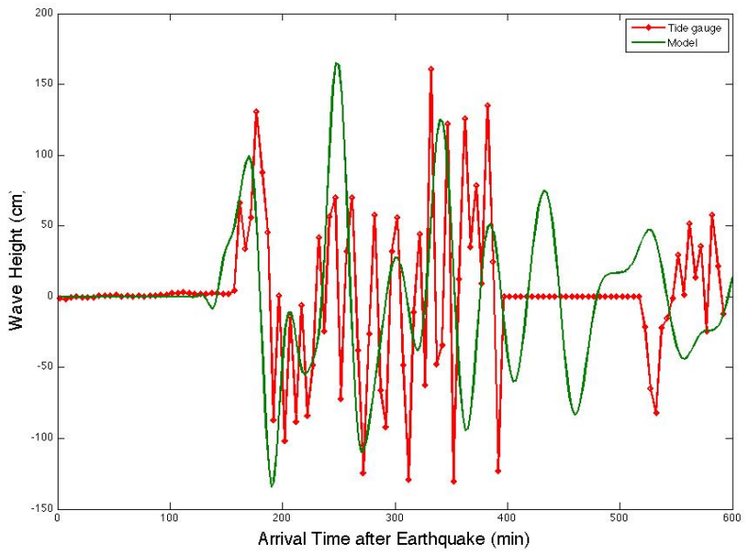
Ref: Data and Grid from Joern Behrens

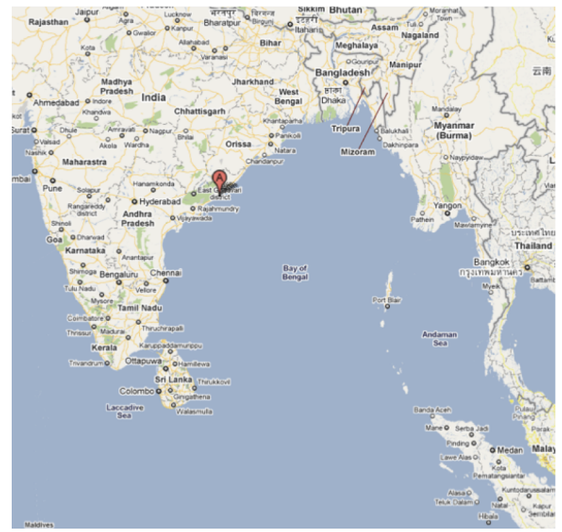
2004 Indian Ocean Tsunami Simulation



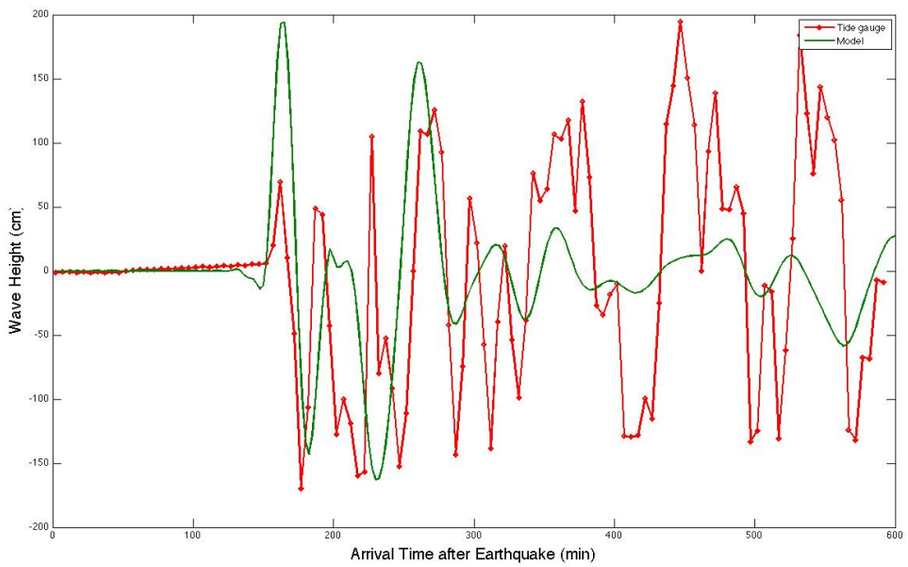
Indian Ocean Tsunami Collaboration with D. Alevras

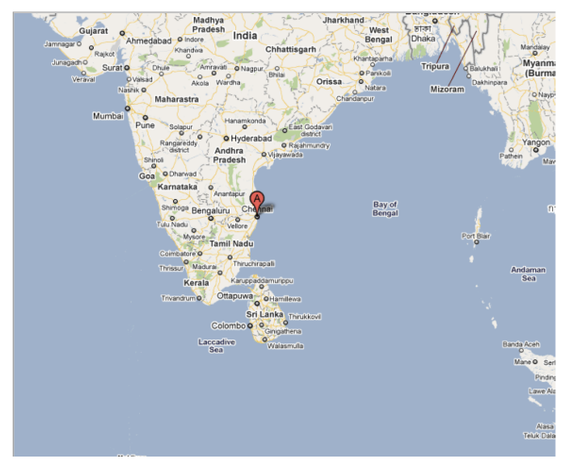
Vizag





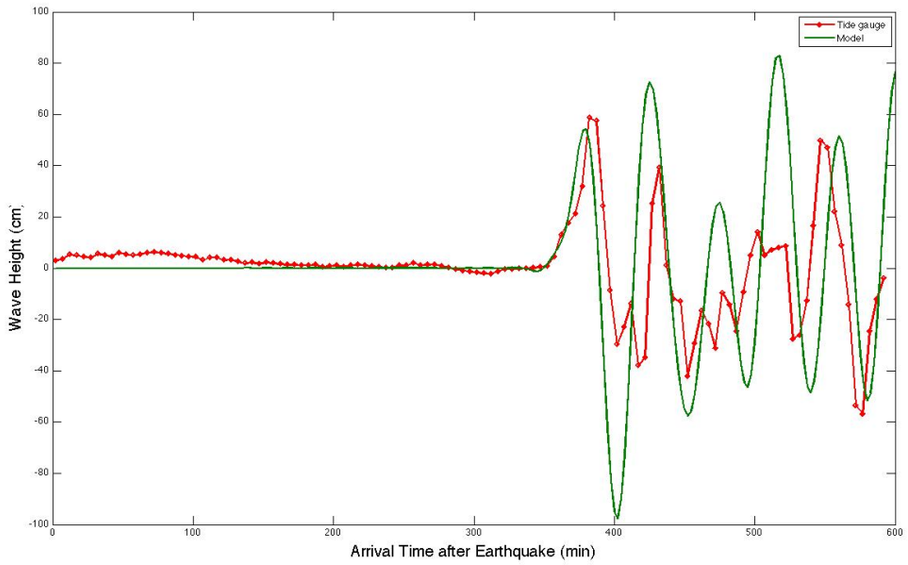
Chennai

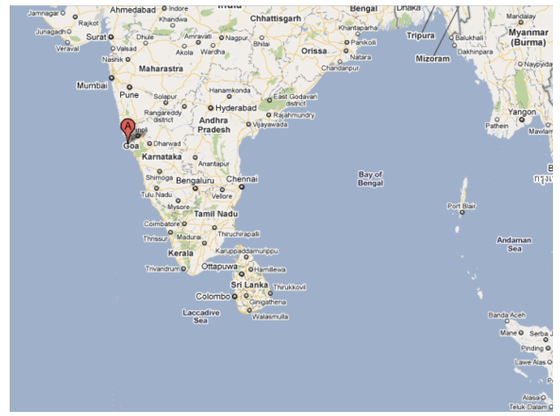




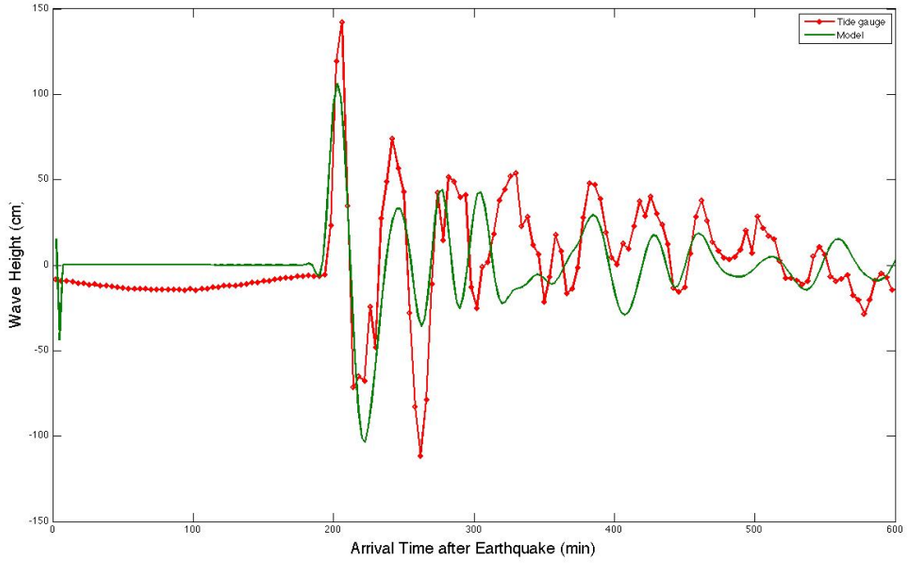
Indian Ocean Tsunami Collaboration with D. Alevras

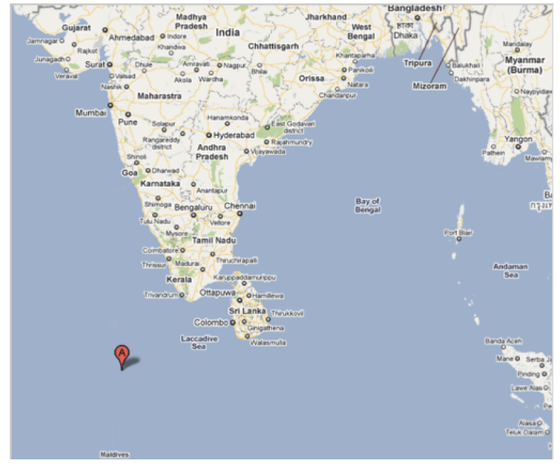
Mormugoa



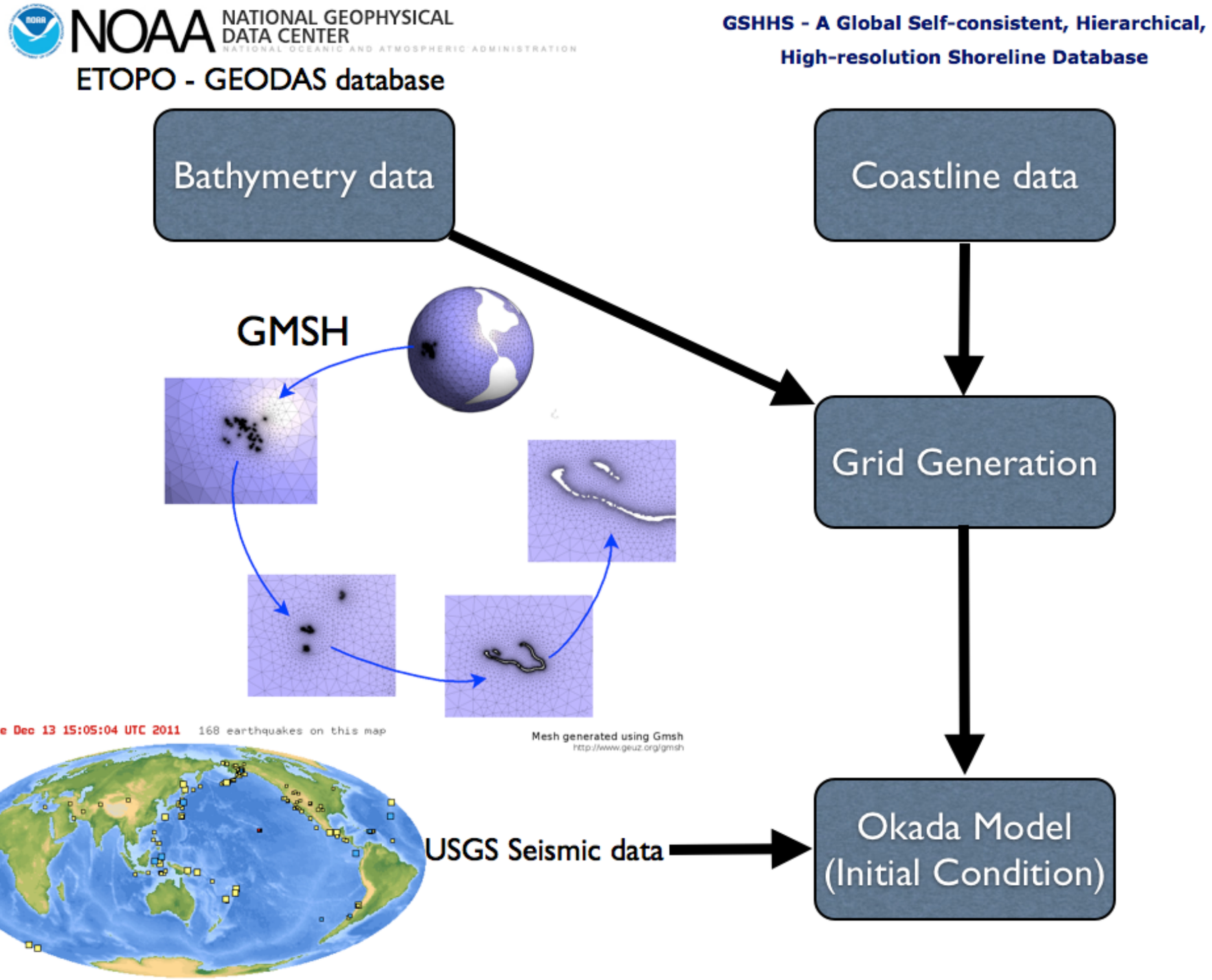


Male





Preprocessing Tsunami Simulations

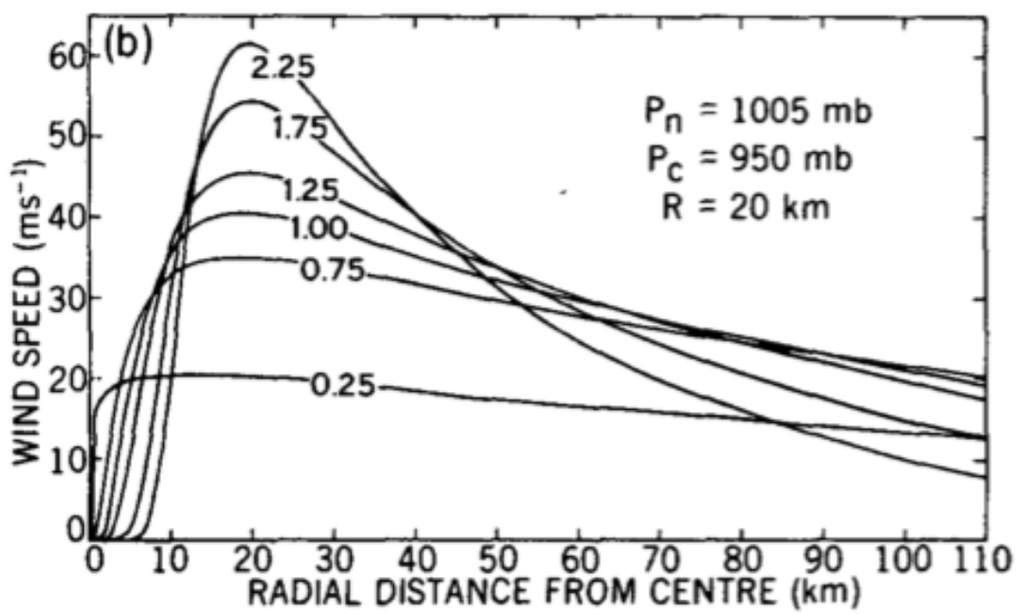


Analytical Storm Model

The wind speed at a radial location r is given as

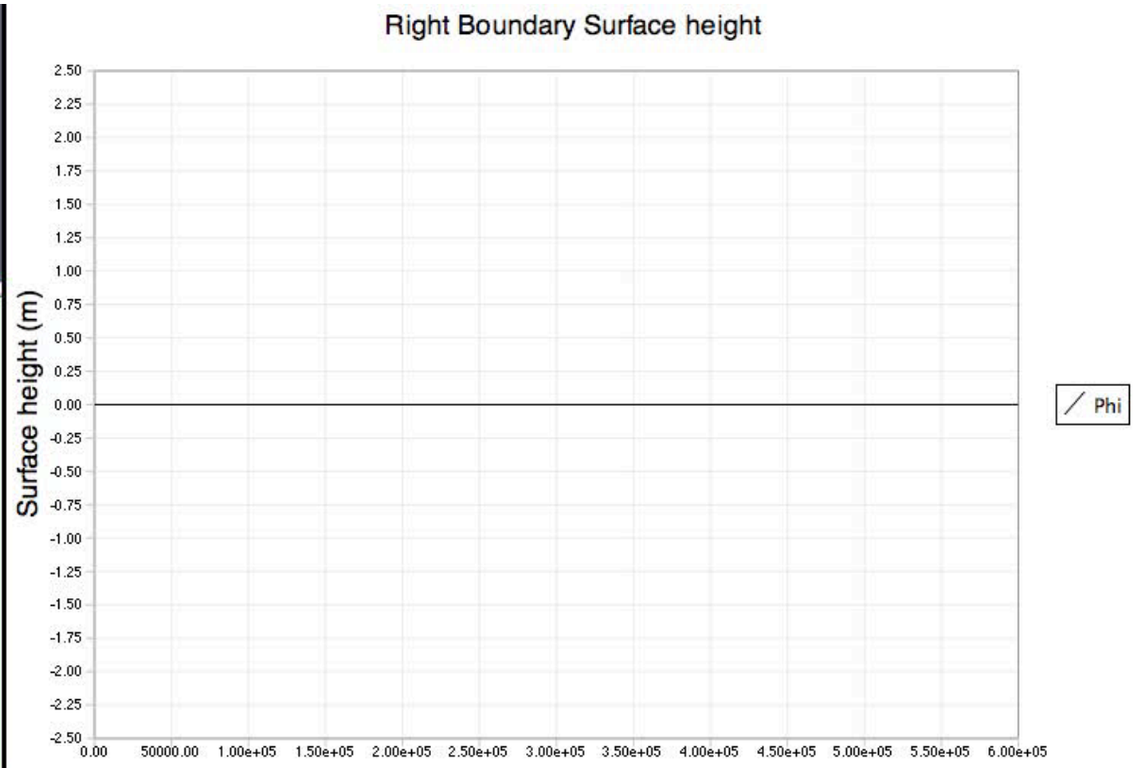
$$V_r = \left(\frac{AB(P_n - P_c)e^{-\frac{A}{r^B}}}{\rho r^B} + \frac{r^2 f^2}{4} \right)^{\frac{1}{2}} - \frac{rf}{2}$$

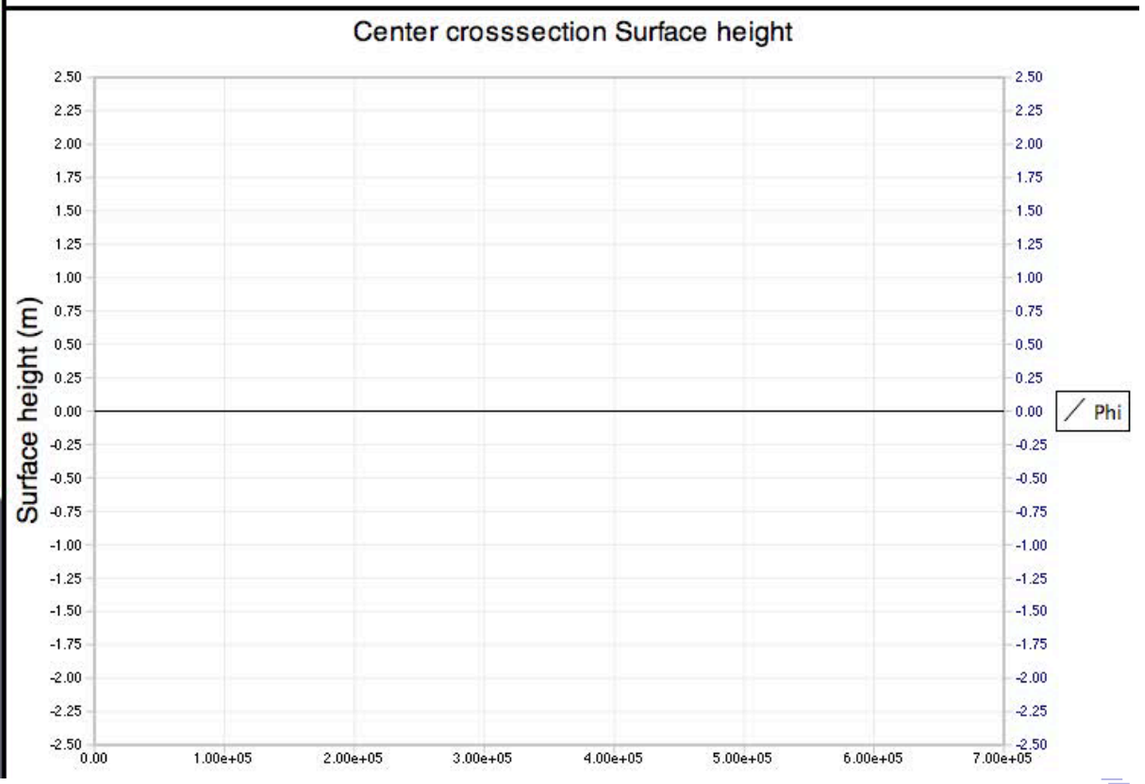
where, P_c – Central pressure, $P - n$ – Ambient pressure,
f – coriolis parameter, A and B – Constants

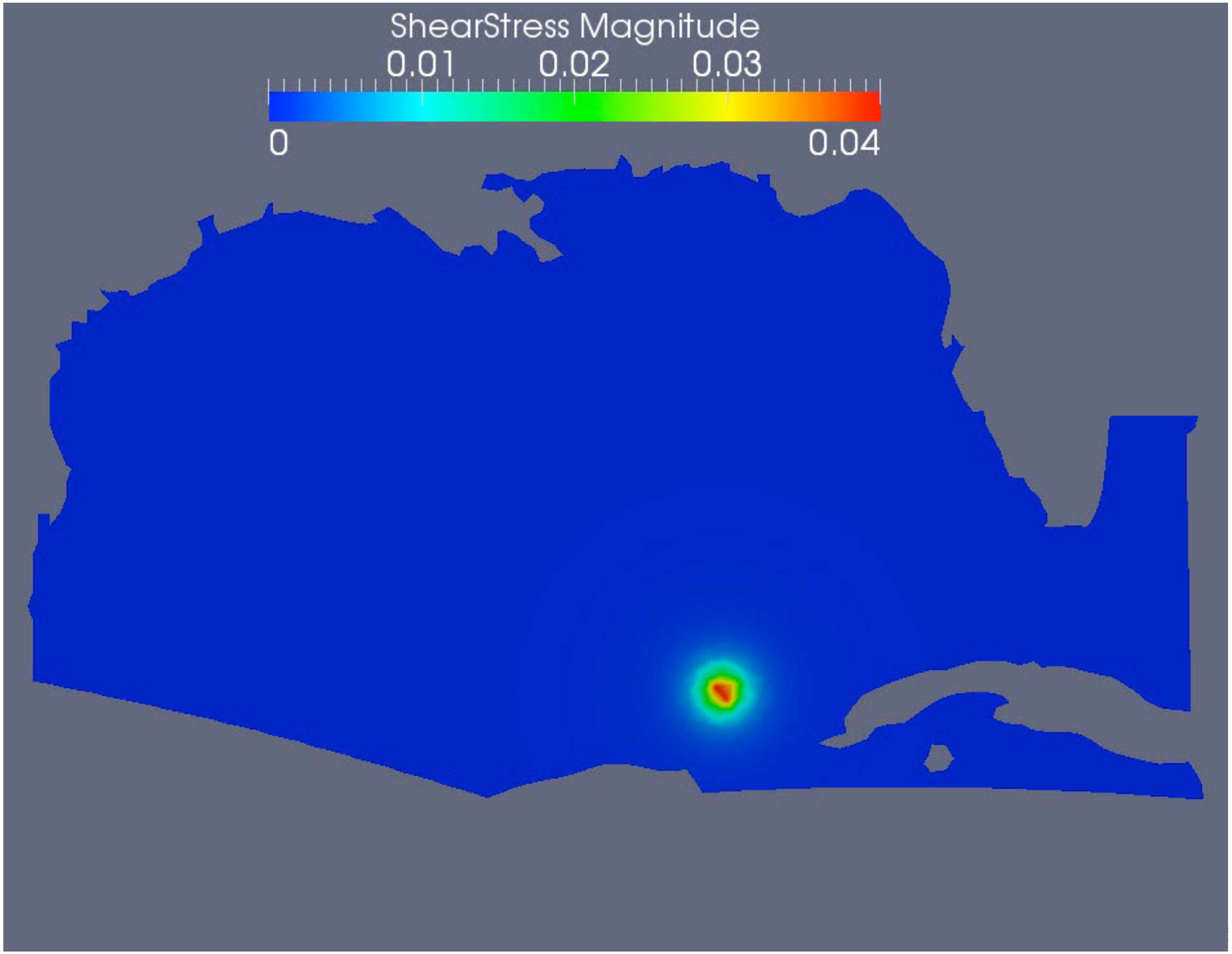


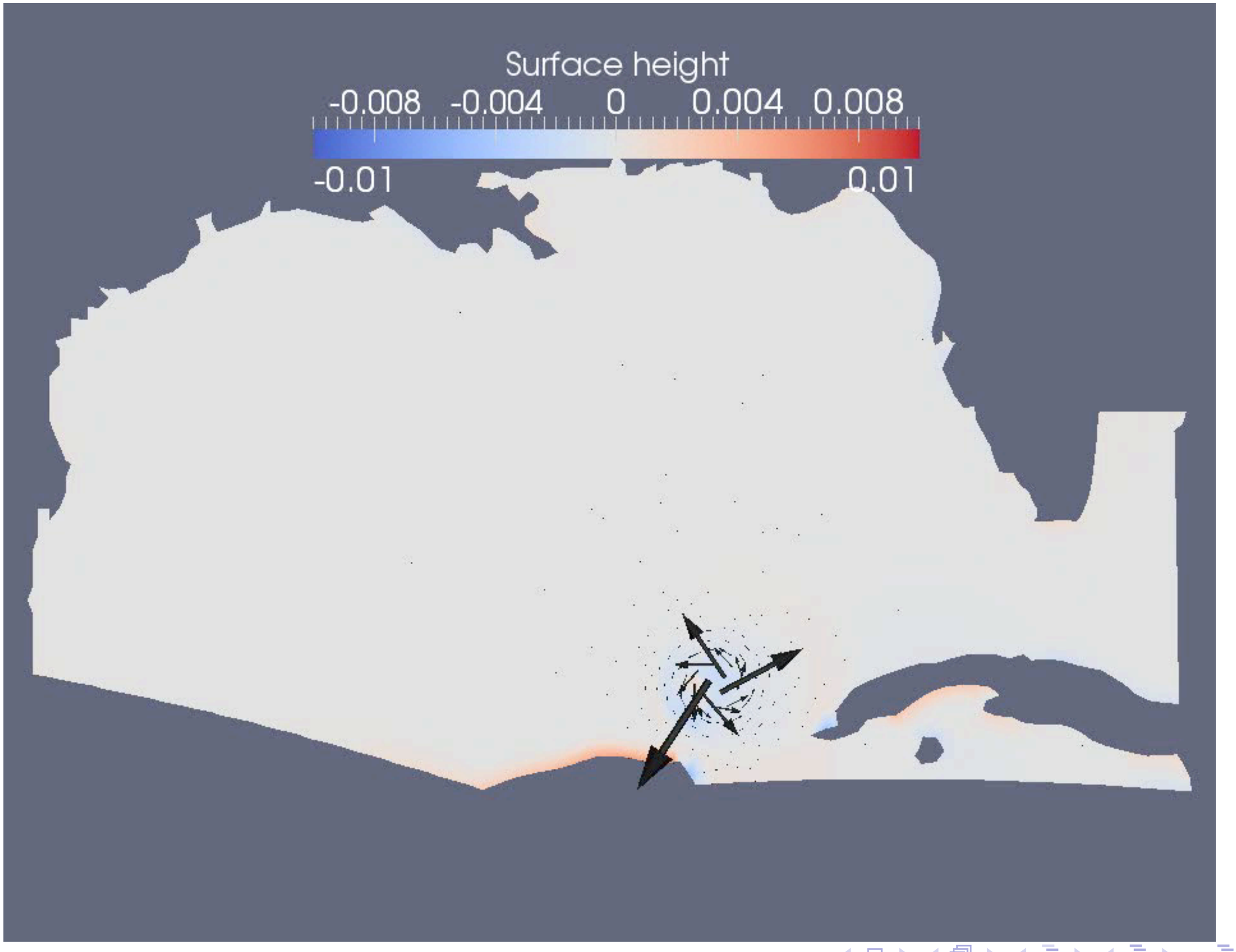
Ref: Holland, G.J. *Monthly Weather Rev.*, 1980, Vol. 108, pg 1212

Idealized storm modeling



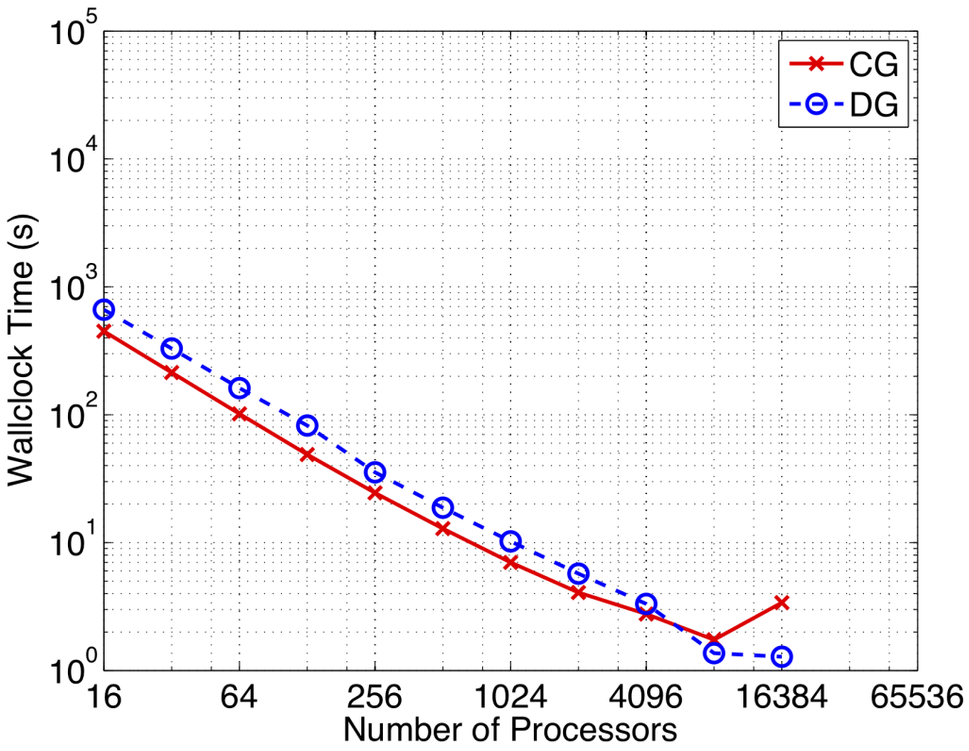



ShearStress Magnitude
0.01
0.02
0.03
0
0.04

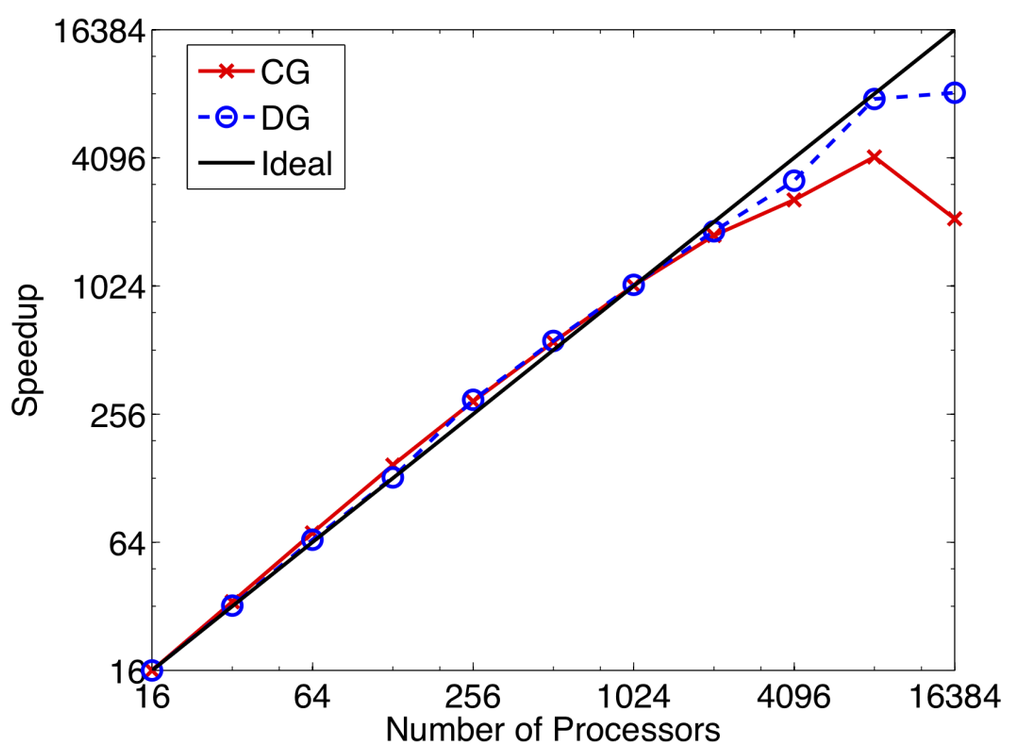

Surface height
-0.008
-0.004
0
0.004
0.008
-0.01
0.01

Speedup of NUMA

4th Order Polynomials



(a) wallclock time

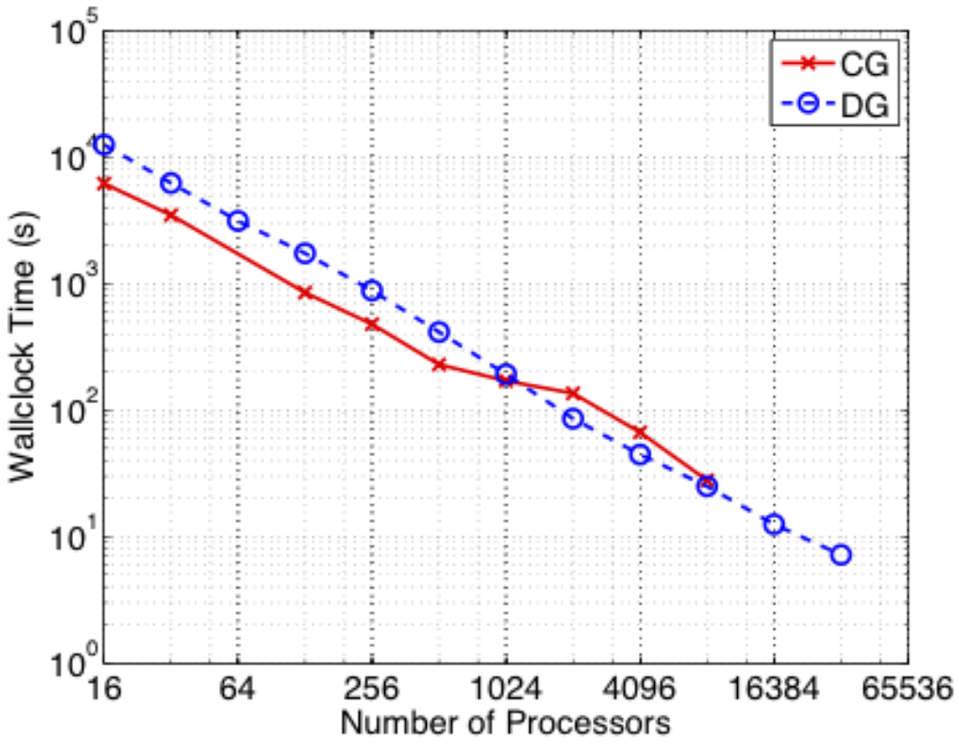


(b) speedup

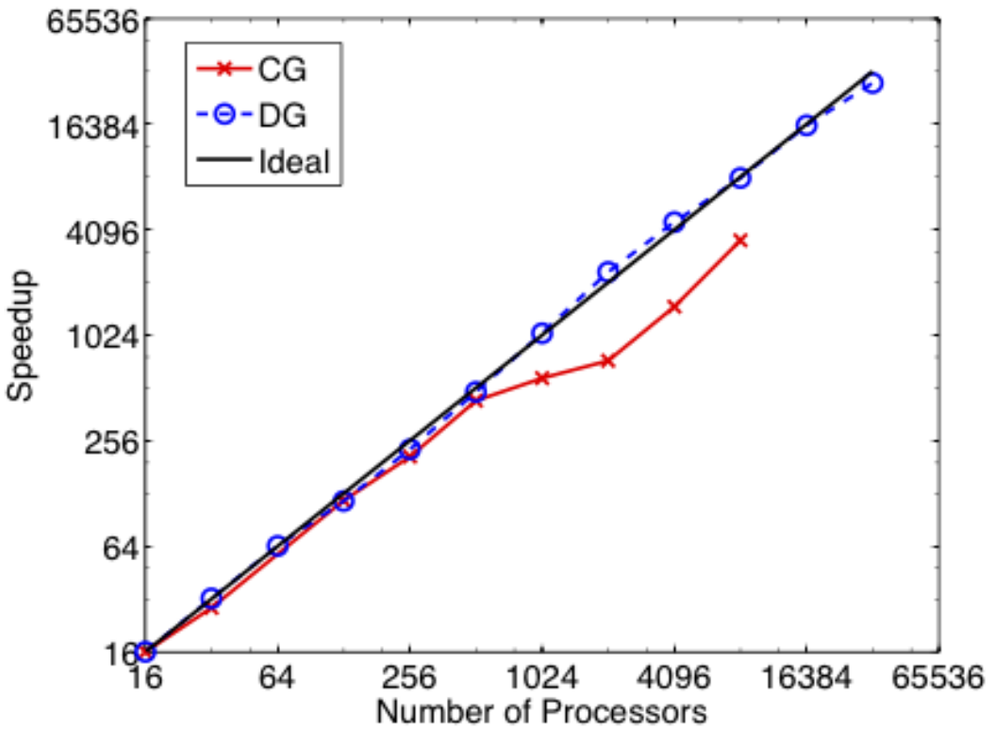
Ref: J.F. Kelly and F.X. Giraldo, Continuous and Discontinuous Galerkin Methods for a Scalable 3D Nonhydrostatic Atmospheric Model: limited-area mode, J. Comp. Phys. (revised 2012)

Speedup of NUMA

8th Order Polynomials



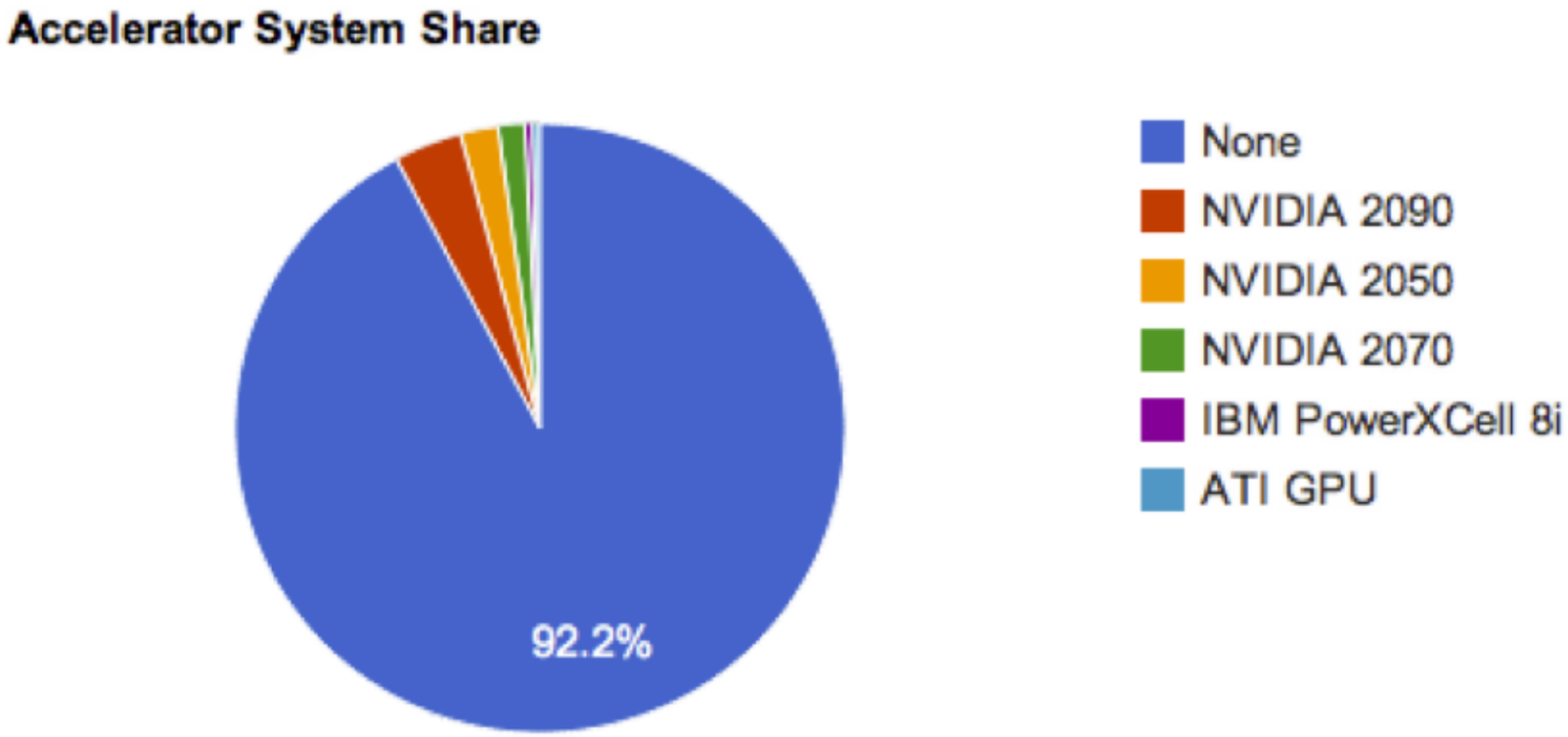
(a) wallclock time



(b) speedup

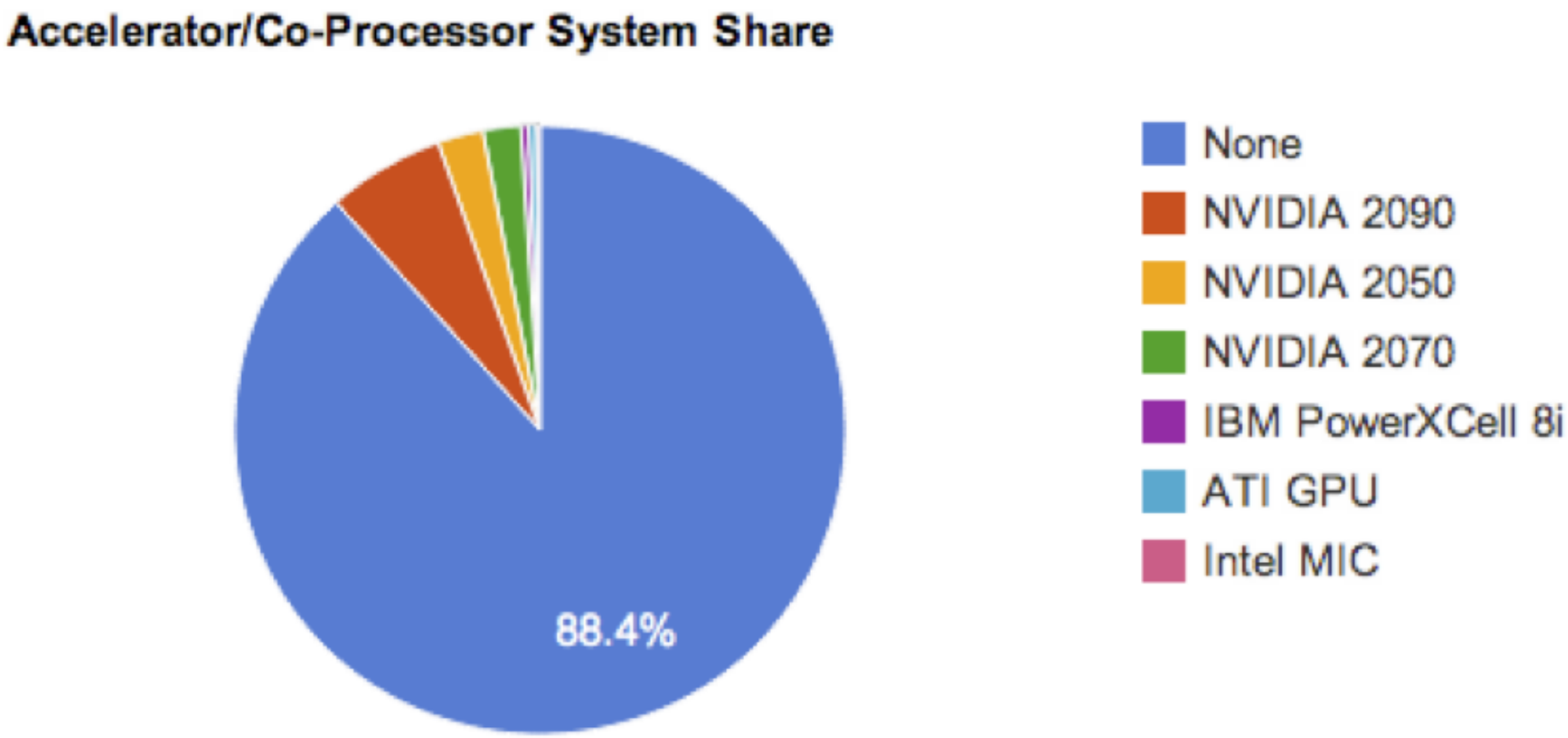
Ref: J.F. Kelly and F.X. Giraldo, Continuous and Discontinuous Galerkin Methods for a Scalable 3D Nonhydrostatic Atmospheric Model: limited-area mode, J. Comp. Phys. (revised 2012)

Growth of Accelarator systems



2011 Top500.org

Growth of Accelarator systems



2012 Top500.org

GPU vs CPU computing

GPU

CPU

Matrix form of semi−discrete equations

Using the polynomial approximation $q_N = \sum_{i=1}^{M_N} \psi_i q_i$

$$\int_{\Omega_e} \psi_i \psi_j dx \frac{\partial q^{(e)}}{\partial t} - F_j^{(e)} \cdot \int_{\Omega_e} \nabla \psi_i \psi_j dx - \int_{\Omega_e} \psi_i \psi_j dx S_j^{(e)}$$

$$= -\sum_{l=1}^{3} \int_{\Gamma_e} \psi_i \psi_j n^{(e,l)} dx \cdot (F^{(*,l)})_j$$

Defining element matrices as

$$M_{ij}^{(e)} = \int_{\Omega_e} \psi_i \psi_j dx, \quad M_{ij}^{(e,l)} = \int_{\Gamma_e} \psi_i \psi_j n^{(e,l)} dx, \quad D_{ij}^{(e)} = \int_{\Omega_e} \nabla \psi_i \psi_j dx$$

Matrix form of semi−discrete equations

$$M_{ij}^{(e)} \frac{\partial q^{(e)}}{\partial t} - \underbrace{(D_{ij}^{(e)})^{\mathcal{T}} F_j^{(e)} - M_{ij}^{(e)} S_j^{(e)}}_{\text{Volume Integration}} = \underbrace{-\sum_{l=1}^{3} (M_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j}_{\text{Flux Integration}}$$

Eliminating mass matrix on LHS

$$\widehat{D}^{(e)} = (M^{(e)})^{-1} D^{(e)}, \quad \widehat{M}^{(e,l)} = (M^{(e,l)})^{-1} M^{(e,l)}$$

$$\frac{\partial q^{(e)}}{\partial t} - (\widehat{D_{ij}}^{(e)})^{\mathcal{T}} F_j^{(e)} - S_j^{(e)} = -\sum_{l=1}^{3} (\widehat{M}_{ij}^{(e,l)})^{\mathcal{T}} (F^{(*,l)})_j$$

GPU speedup - higher order