Hamiltonian PDEs: KAM, Reducibility, Normal Forms and Applications- CMO

Dispersive Shock waves in Hamiltonian Bidirectional Whitham systems.

Rosa María Vargas Magaña
Joint work with Noel Smyth and Tim Marchant

June 13, 2019

Outline

- Dispersive Shock waves (DSW)
- Bidirectional Whitham systems.
- DSW fitting method
- Preliminary results

Main goal: The determination of the macroscopic DSW properties in the general water waves equations.

b)

Fig 1. Three standing paddle board riders on a river undular bore in Turnagain Arm, Alaska,
Gennady El. M. Hoefer "DSW and Modulation theory", 2016

Fig 2. Tidal river bore on a river Ribble Lancashire,
Gennady El. M. Hoefer "DSW and Modulation theory", 2016

A tidal phenomenon in which **the incoming tide forms a wave** (**or waves**) **of water against the direction of the current,** is known as Tidal Bore. It occurs in locations with a large tidal range, typically more than 6 meters, between high and low water.

DSW Everywhere!

Fig 3. Atmospheric DSW. a) Morning glory roll cloud. b) Mountain waves

ATMOSPHERE

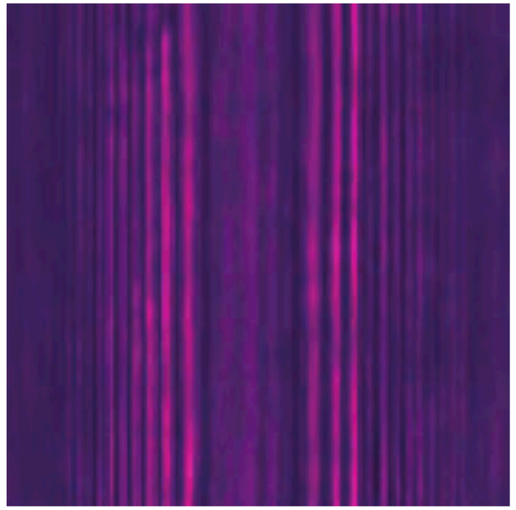


Fig 4. Image of the output set of a defocusing, photo refractive crystal at the input face.

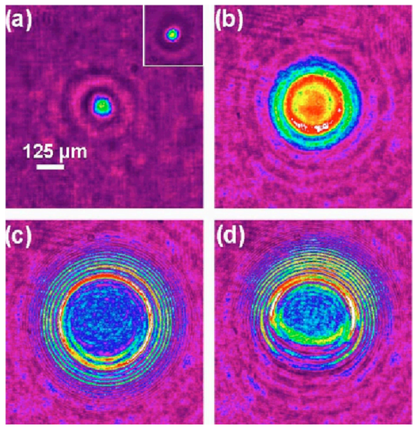


Fig 5. Experimental evolution of DSW after 1cm of propagation in an ethanol + iodine cell

OPTICS

Fig 5. Atmospheric gravity waves moving southward off the Texas coast and out over the western Golf of Mexico

Internal waves

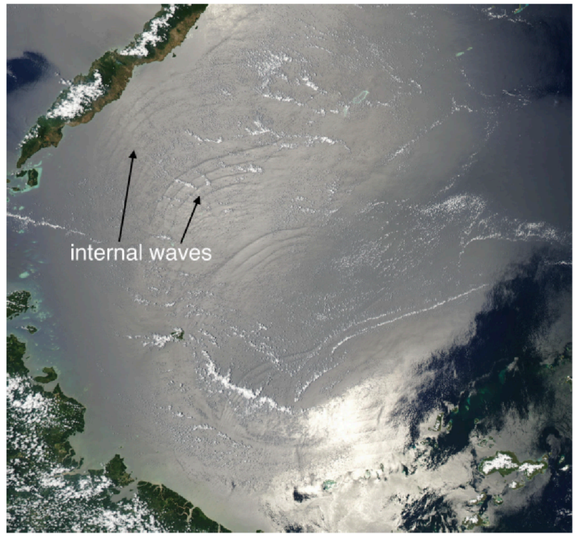


Figure 1.2: Internal waves in the Sulu Sea between the Philippines and Malaysia [97].

Gennady El. M. Hoefer “DSW and Modulation theory”, 2016

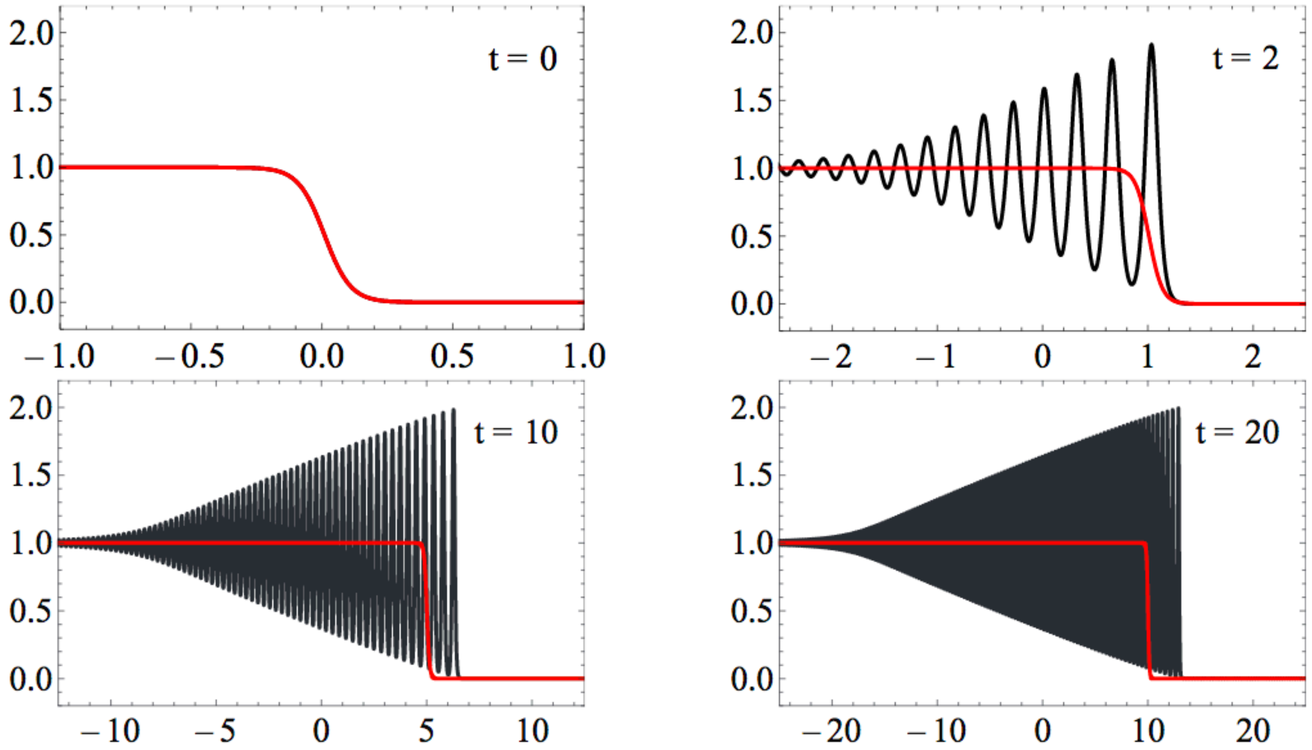
Xin An, PhD Thesis 2018

A DSW is the dispersive resolution of a step, or near-step initial condition

Typical numerically computed solution of the KdV equation is in black and Burgers' equation is in red

$$u_t + uu_x = 0$$

$$u_t + uu_x + u_{xxx} = 0$$

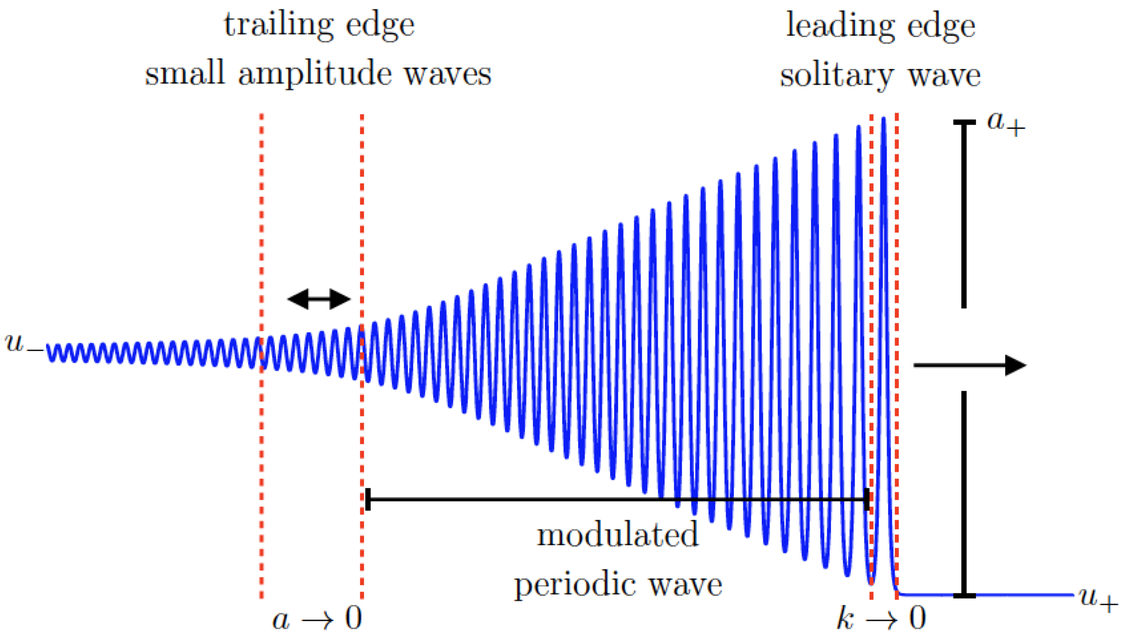


Grimshaw Presentation 2006

The seminal paper of **Gurevich and Pitaevskii*** adapted the Riemann problem to consider the longtime behavior of step initial data for the KdV equation, a dispersively regularized hyperbolic equation. The solution to this problem is fundamental to most all of DSW theory, we refer to the KdV Riemann problem as the GP problem.

[*] A. V. Gurevich, L. P. Pitaevskii, Nonstationary structure of acollisionless shock wave, Sov. Phys. JETP (1974), translation from Russian

Anatomía de un DSW clásico del tipo KdV



Gennady El and M. Hoefer “DSW and Modulation theory”, 2016

The FLUID is:

- Inviscid
- Homogeneous
- Incompressible
- Irrotational

The only volumetric force acting on the fluid is gravity.

The FLUID DOMAIN $\mathcal{D}(\beta(x), \eta(x,t))$ is

- Simple connected.
- In 2D or 3D.

In Eulerian representation $u(x,t)$ represent the velocity of the point x at time t.
The **field of velocities** can be described as the **gradient of a function**

$$u = \nabla \varphi$$

EULER'S EQUATIONS IN 2D

1 $$\varphi_{xx} + \varphi_{yy} = 0 \qquad \mathcal{D}(\beta(x), \eta(x,t))$$

2 $$\eta_t + \eta_x \varphi_x - \varphi_y = 0 \qquad y = \eta(x,t)$$

3 $$\varphi_t + \frac{1}{2}(\varphi_x^2 + \varphi_y^2) + g\eta = 0 \qquad y = \eta(x,t)$$

4 $$\frac{\partial \varphi}{\partial n} = 0 \qquad y = -h_0 + \beta(x)$$

- It is a classical free-boundary problem
- The boundary conditions in the free surface are strongly nonlinear and coupled equations

Hamiltonian formulation

Zakharov (1968) poses the evolution equations in the form of a Hamiltonian system in the canonical variables $(\eta(x), \xi(x))$ **with** $\xi(x,t) = \varphi(x, \eta(x,t), t)$

$$\partial_t \begin{pmatrix} \eta \\ \xi \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} \delta_\eta H \\ \delta_\xi H \end{pmatrix} = J\delta H,$$

V.E. Zakharov (1968), J.W. Miles, (1977)

$$H = \tfrac{1}{2} \int_{\mathbb{R}} (\xi G(\beta, \eta)\xi + g\eta^2) dx$$

W. Craig and C. Sulem (1993)

Hamiltonian in terms of Dirichlet-Neumann operator

Green Identity

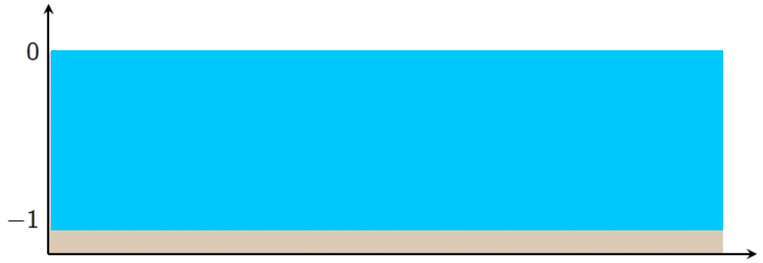
$$K = \int_D \rho \frac{|\nabla\varphi|^2}{2} dV = \frac{\rho}{2} \int_D \nabla \cdot (\varphi \nabla\varphi) dV = \frac{\rho}{2} \int_{\partial D} \xi (\nabla\varphi \cdot \hat{n}) \mid_{y=\eta} dS$$

$$H = \frac{1}{2} \int_{\mathbb{R}} (\xi G(\beta, \eta)\xi + g\eta^2) dx$$

W. Craig and C. Sulem (1993)

Hamiltonian: $H = \frac{1}{2}\int_{\mathbb{R}}(\xi G(\beta,\eta)\xi + g\eta^2)dx$

- In general there is not an explicit expression for it!
- This case gives rise to an explicit expression to de DN operator:



$[G(0,0)]: \quad \xi \longmapsto D\tanh(D)\xi$ What is this object?

Where D is as usual the operator $D = -i\partial_x$

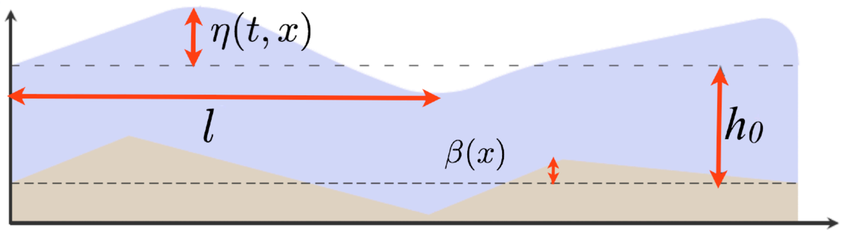
$$\xi(x) \longmapsto \hat{\xi}(\kappa) \longmapsto \kappa\tanh(h_0\kappa)\hat{\xi}(\kappa) \longmapsto \int_{\mathbb{R}} \kappa\tanh(h_0\kappa)\hat{\xi}(\kappa)e^{i\kappa x}d\kappa := [D\tanh(h_0 D)]\xi$$

Hamiltonian: $H = \frac{1}{2}\int_{\mathbb{R}}(\xi G(\beta,\eta)\xi + g\eta^2)dx$

- In general there is not an explicit expression for it!
- This case gives rise to an explicit expression to de DN operator:

$[G(0,0)]: \quad \xi \longmapsto D\tanh(D)\xi$ What is this object?

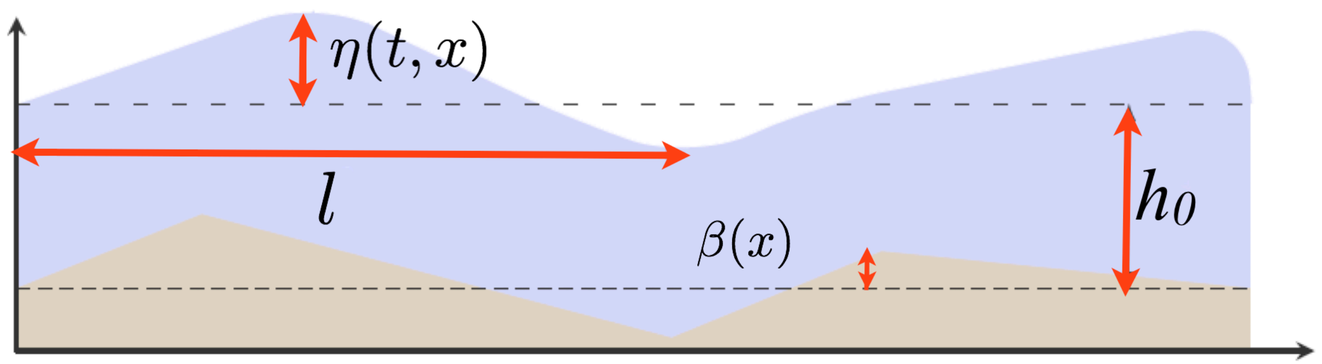
Where D is as usual the operator $D = -i\partial_x$



$$H_{\mathcal{A}} = \frac{1}{2}\int_{\mathbb{R}}(\xi G_{\mathcal{A}}(\beta,\epsilon\eta)\xi + g\eta^2)dx$$

$$\xi(x) \longmapsto \hat{\xi}(\kappa) \longmapsto \kappa\tanh(h_0\kappa)\hat{\xi}(\kappa) \longmapsto \int_{\mathbb{R}}\kappa\tanh(h_0\kappa)\hat{\xi}(\kappa)e^{i\kappa x}d\kappa := [D\tanh(h_0 D)]\xi$$

Shallow water regime



$$\delta = \frac{h_0^2}{l^2} \ll 1 \quad \sim \quad \epsilon = \frac{\eta}{h_0} \qquad \textbf{SMALL}$$

$$[G(\beta, \epsilon\eta)](\xi) = \sum_{n=0}^{\infty} [G_n(\beta, \eta)](\xi)\epsilon^n$$

$$\begin{aligned} G_0(\beta,\eta) &= D\tanh(h_0 D) + DL(\beta), \\ G_1(\beta,\eta) &= D\eta D - G_0\eta G_0, \\ G_2(\beta,\eta) &= \frac{1}{2}(G_0 D\eta^2 D - D^2\eta^2 G_0 - 2G_0\eta G_1), \end{aligned}$$

with $D = -i\partial_x$

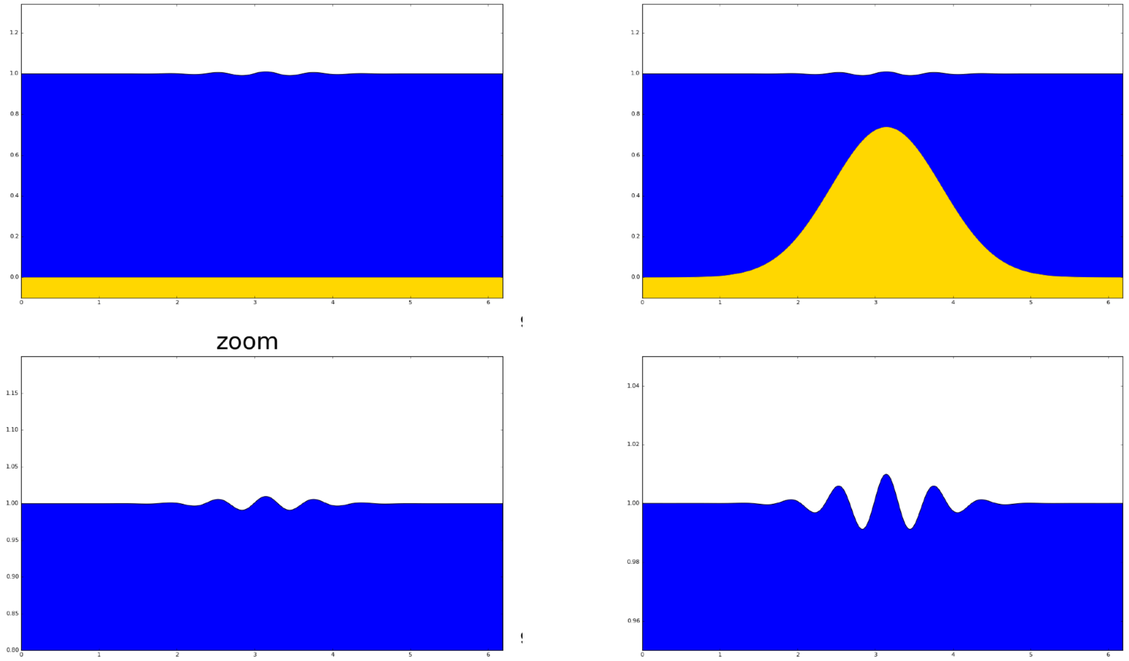
and $L(\beta)$ Involve **pseudo-differential** operators

Coifman and Meyer (1985)
Craig, Schanz and Sulem (1997)
Craig and Sulem (2005)
Craig, Guyenne,
Nicholls, Sulem (2005)

Whitham-Boussinesq non-local shallow water wave model for variable depth

$$H_{\mathcal{A}} = \frac{1}{2}\int_{\mathbb{R}}(\xi G_{\mathcal{A}}(\beta, \epsilon\eta)\xi + g\eta^2)dx$$

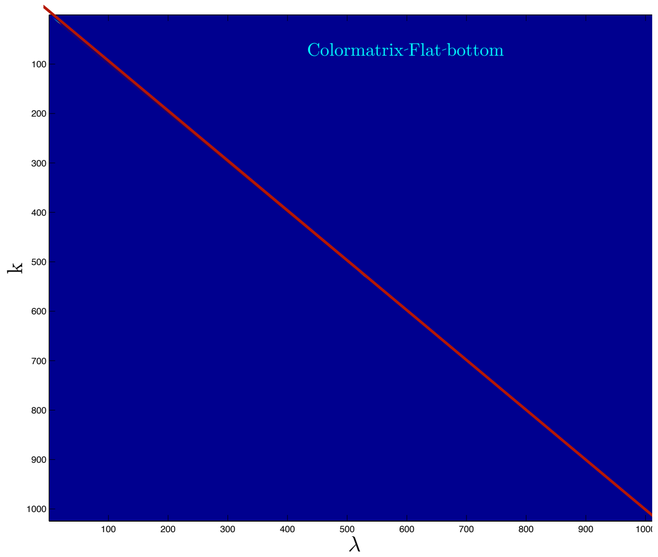
$$G_{\mathcal{A}} = Sym(\frac{D}{\sqrt{\epsilon}}\tanh(\sqrt{\epsilon}h(x)D)) \quad \text{with} \quad D = -i\partial_x$$

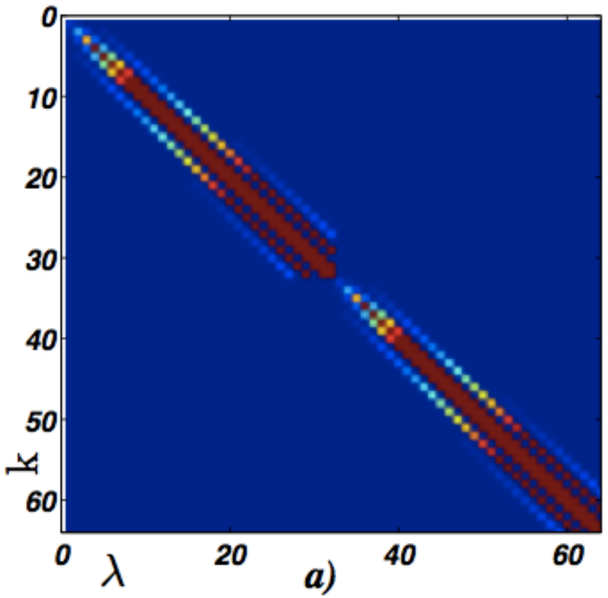


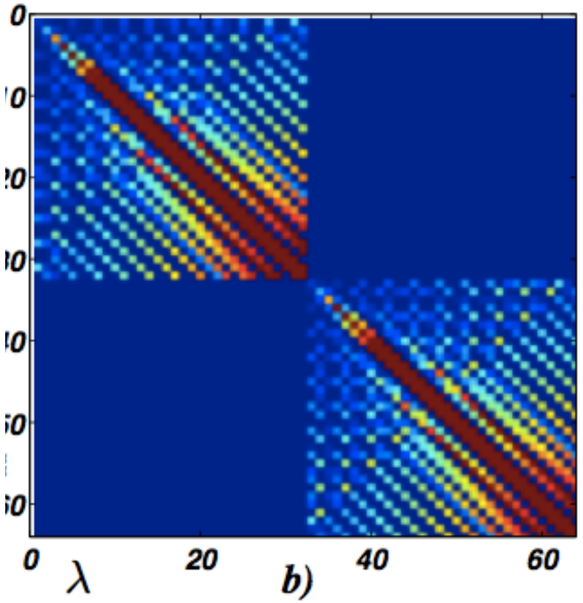
$$H_2^M = \frac{1}{4\pi}(\hat{\xi}^T, (\hat{\xi}^*)^T) Sym \begin{pmatrix} A^* & B \\ B^* & A \end{pmatrix} \begin{pmatrix} \hat{\xi} \\ \hat{\xi}^* \end{pmatrix} + -\frac{\epsilon}{8\pi^2} \Big(\sum_{[k_2,k_3]\in J_M^2} \Big[k_2 k_3 \hat{\xi}_{k_2} \hat{\xi}_{k_3} \hat{\eta}^*_{k_2+k_3} + k_2 k_3 \hat{\xi}^*_{k_2} \hat{\xi}^*_{k_3} \hat{\eta}_{k_2+k_3} \Big]$$

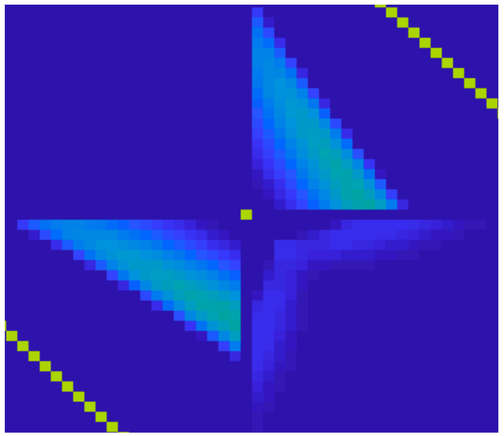
$$+ \sum_{[k_2,k_3]\in J_M^2} \Big[-k_2 k_3 \hat{\xi}_{k_2} \hat{\xi}^*_{k_3} \hat{\eta}_{-k_2+k_3} - k_2 k_3 \hat{\xi}^*_{k_2} \hat{\xi}_{k_3} \hat{\eta}^*_{-k_2+k_3} \Big] \Big)$$

In a periodic framework we obtain an **infinite matrix representation** in terms of the Fourier coefficients of the symbol of the operator.









- **Infinite dimensional Hamiltonian system**

- **The canonical variables are the surface quantities.** $(\eta(x), \xi(x))$

- **The evolution equation are expressed in Darboux coordinates using Variational derivatives of the Hamiltionan.**

$$\partial_t \begin{pmatrix} \eta \\ \xi \end{pmatrix} = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \begin{pmatrix} \delta_\eta H \\ \delta_\xi H \end{pmatrix} = J\delta H,$$

$$\mathcal{F}[u] = \int_a^b f(u(x))dx \qquad \delta\mathcal{F} = \int_a^b dx\{\frac{\partial f}{\partial u}(x)\delta u(x) + \frac{1}{2}\frac{\partial^2 f}{\partial u^2}(x)(\delta u(x))^2 + ..., \}$$

- **Pseudo-differential Operators**

$$[a(f(x)D^m)\xi](x) = \frac{1}{2\pi}\int_{\mathbb{R}} a(f(x)k^m)\hat{\xi}(k)e^{ikx}dk,$$

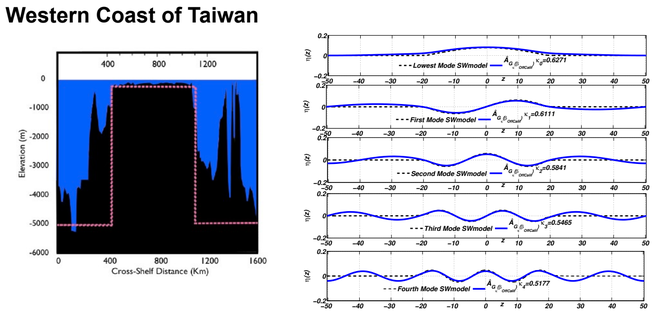
With a, f **real functions and** $\hat{\xi}(k) = \int_{\mathbb{R}} \xi(x)e^{-ikx}dk$

- R. M. Vargas-Magaña, P. Panayotaros
A Whitham-Boussinesq long-wave model for variable topography, Wave Motion (2016)

- R. M. Vargas-Magaña, P. Panayotaros, A. A. Minzoni
Linear modes for channels of constant cross-section and approximate Dirichlet-Neumann operators, *Water waves Journal* ***IN PRESS 2019***

Projects

- Trapped longitudinal modes
in domains with unbounded constant cross-section.



- Bloch Theory, Spectral Gaps and Wannier Functions for linearized waves using the DN operator

Bidirectional Whitham Equations as Models of Waves on Shallow Water

John D. Carter (2018)

The accuracy of the Whitham type equations to model data from real-world experiments of waves on shallow water.

The conclusion is that the most accurate predictions come from models that contain accurate reproductions of the

Euler phase velocity, $C_E = \sqrt{\frac{g \tanh(kh_0)}{k}}$

sufficient nonlinearity,

and surface tension effects.

Current Methods to determine analytically the macroscopic DSW properties.

- Nonlinear wave modulation theory
- DSW Fitting method introduced by Gennady El and collaborator in 2006

For non-integrable dispersive equations, the explicit determination of the Whitham modulation system is not possible in general.

A DSW fitting procedure determine analytically the macroscopic DSW properties.

DSW fitting method

These macroscopic DSW properties include:

- the DSW edge speeds and
- the DSW wave parameters:
  - the harmonic edge wavenumber
  - the soliton edge amplitude.

The determination of these observables represents the fitting of a DSW to the long-time dynamics of piecewise constant, initial Riemann data.

The DSW fitting method proposed by Gennady El and collaborators is based on a fundamental, generic property the Whitham modulation equations admit exact reductions to a set of common, much simpler, analytically tractable equations in the limits of vanishing amplitude and vanishing wavenumber, which correspond to the harmonic and soliton DSW extremes.

DSW fitting method for Whitham-type equations

$$u_t + 2uu_x + D_2[u]_x = 0$$

Modulations equations are strictly hyperbolic, nonlinear and have solitary and linear limits

Non-dispersive form: $\frac{\partial \overline{u}}{\partial t} + 2\overline{u}\frac{\partial \overline{u}}{\partial x} = 0.$

Dispersion relation: $\omega(k)$ **Convexa**

Matching the non dispersive region behind the DSW and the trailing edge is determined by

$$\begin{cases} \frac{dk}{d\overline{u}} = \frac{\frac{\omega}{\partial \overline{u}}}{V(\overline{u}) - \frac{\partial \omega}{\partial k}}, \\ k(u_+) = 0 \end{cases}$$

with $V(\overline{u}) = 2\overline{u}$ **and** $\overline{u}$ **the mean of** u

The leading solitary wave edge of a DSW is determined in a similar fashion but in terms of conjugate variables.

Whitham-Boussinesq

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial \eta}{\partial x} + u\frac{\partial u}{\partial x} &= 0, \\ \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\eta u + \frac{\partial}{\partial x}[\frac{\tanh D}{D}](u) &= 0 \end{aligned}$$

Whitham-Boussinesq with surface Tension

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial \eta}{\partial x} + u\frac{\partial u}{\partial x} &= 0, \\ \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\eta u + \frac{\partial}{\partial x}\frac{\tanh D}{D}\left[1+\alpha D^2\right]u &= 0, \quad \alpha = \frac{T}{\rho}. \end{aligned}$$

Whitham-Boussinesq fondo plano

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial \eta}{\partial x} + u\frac{\partial u}{\partial x} &= 0, \\ \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\eta u + \frac{\partial}{\partial x}[\frac{\tanh D}{D}](u) &= 0 \end{aligned}$$

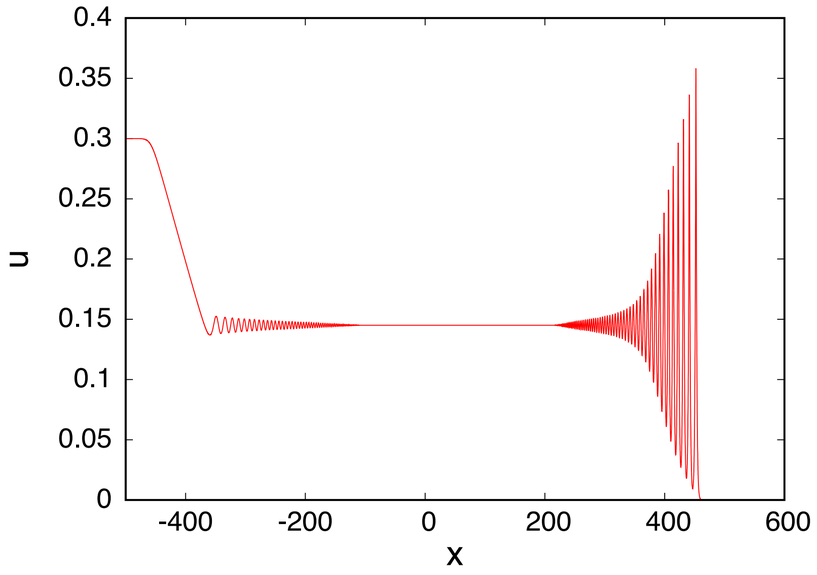


Fig. Undular bore del modelo **Whitham-Boussinesq fondo plano** η=0 and u_0=0. **Smyth 2019**

Whitham-Boussinesq con tensión superficial

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial \eta}{\partial x} + u\frac{\partial u}{\partial x} &= 0, \\ \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\eta u + \frac{\partial}{\partial x}\frac{\tanh D}{D}\left[1+\alpha D^2\right]u &= 0, \quad \alpha = \frac{T}{\rho}. \end{aligned}$$

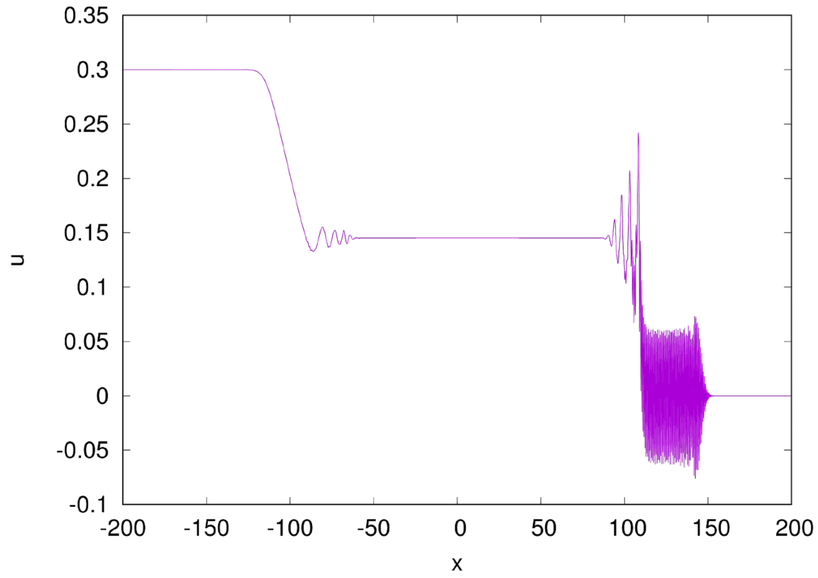
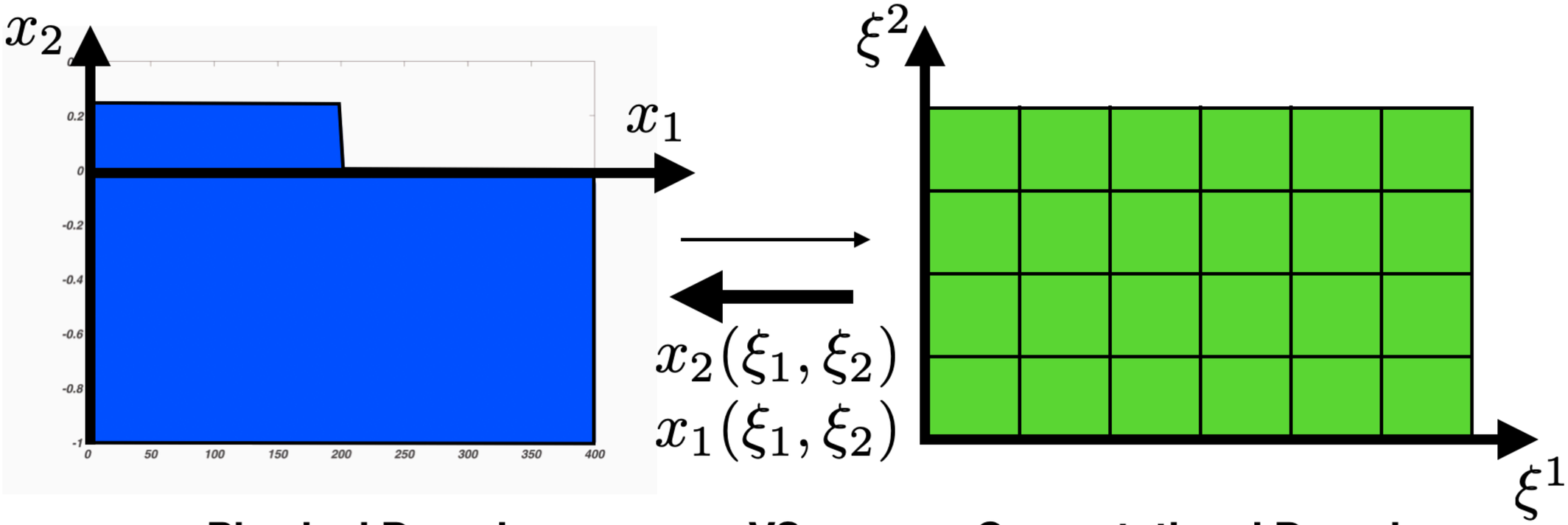


Fig. Undular bore del modelo **Whitham-Boussinesq** con **stensión superficial** y fondo plano T=0 and u_0=0. **Smyth 2019**

Euler Equations in Curvilinear Coordinate System in 2D



Physical Domain **VS** **Computational Domain**

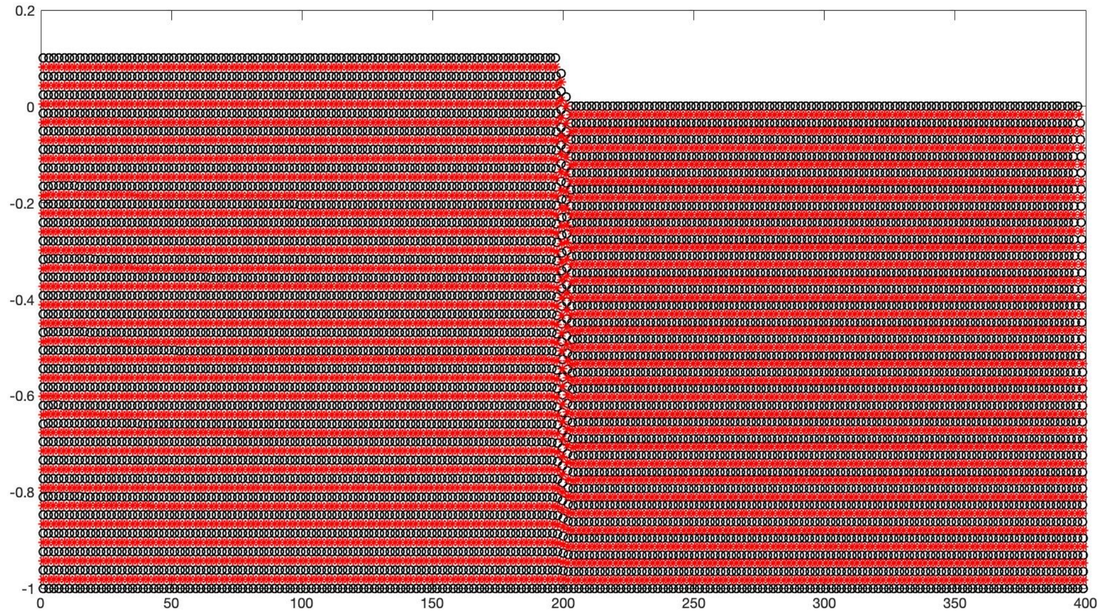
Euler Equations in Curvilinear Coordinate System in 2D

$$
\begin{cases}
i. \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = 0. & -1 < x_2 < \eta(t, x_1), \\
ii. \frac{\partial u_1}{\partial t} + \frac{\partial u_1^2}{\partial x_1} + \frac{\partial u_2 u_1}{\partial x_2} = -\frac{\partial p}{\partial x_1} & -1 < x_2 < \eta(t, x_1) \\
iii. \frac{\partial u_2}{\partial t} + \frac{\partial (u_1 u_2)}{\partial x_1} + \frac{\partial u_2^2}{\partial x_2} = -\frac{\partial (p + x2)}{\partial x_2} & -1 < x_2 < \eta(t, x_1) \\
iv.\ u_2 = 0 & \text{in } x_2 = -1 \\
iv.\ u_1 = 0 & \text{in } x_1 = 0 \\
iv.\ u_1 = 0 & \text{in } x_1 = x_{max} \\
v. \frac{\partial \eta}{\partial t} + u_1 \frac{\partial \eta}{\partial x_1} = u_2 & x_2 = \eta(x_1, t)
\end{cases}
$$

$$
\begin{cases}
i. \frac{\partial}{\partial \xi_1}(J^{-1}\frac{\partial \xi^1}{\partial x_1}u_1 + J^{-1}\frac{\partial \xi^1}{\partial x_2}u_2) + \frac{\partial}{\partial \xi_2}(J^{-1}\frac{\partial \xi^2}{\partial x_1}u_1 + J^{-1}\frac{\partial \xi^2}{\partial x_2}u_2) = 0 & -1 < x_2 < \eta(t, x_1), \\
ii. \frac{\partial J^{-1}u_1}{\partial_t} + \frac{\partial}{\partial \xi^1}(J^{-1}u_1\frac{\partial \xi^1}{\partial t} + J^{-1}u_1\frac{\partial \xi^1}{x_1}u_1 + J^{-1}u_1\frac{\partial \xi^1}{\partial x_2}u_2) + \\
\frac{\partial}{\partial \xi^2}(J^{-1}u_1\frac{\partial \xi^2}{\partial t} + J^{-1}u_1\frac{\partial \xi^2}{x_1}u_1 + J^{-1}u_1\frac{\partial \xi^2}{\partial x_2}u_2) = \\
-\left[\frac{\partial}{\partial \xi^1}(J^{-1}p\frac{\partial \xi^1}{\partial x_1}) + \frac{\partial}{\partial \xi^2}(J^{-1}p\frac{\partial \xi^2}{\partial x_1})\right] \\
iii. \frac{\partial J^{-1}u_2}{\partial_t} + \frac{\partial}{\partial \xi^1}(J^{-1}u_2\frac{\partial \xi^1}{\partial t} + J^{-1}u_2\frac{\partial \xi^1}{x_1}u_1 + J^{-1}u_2\frac{\partial \xi^1}{\partial x_2}u_2) + \\
\frac{\partial}{\partial \xi^2}(J^{-1}u_2\frac{\partial \xi^2}{\partial t} + J^{-1}u_2\frac{\partial \xi^2}{x_1}u_1 + J^{-1}u_2\frac{\partial \xi^2}{\partial x_2}u_2) = \\
-\left[\frac{\partial}{\partial \xi^1}(J^{-1}\phi\frac{\partial \xi^1}{\partial x_2}) + \frac{\partial}{\partial \xi^2}(J^{-1}\phi\frac{\partial \xi^2}{\partial x_2})\right] \\
iv.\ u_2 = 0 & \text{in } x_2 = -1 \\
iv.\ u_1 = 0 & \text{in } x_1 = 0 \\
iv.\ u_1 = 0 & \text{in } x_1 = x_{max} \\
v. \frac{\partial \eta}{\partial_t} + (\frac{\partial \xi^1}{\partial t} + u_1\frac{\partial \xi^1}{\partial x_1})\frac{\partial \eta}{\partial \xi^1} + (\frac{\partial \xi^2}{\partial t} + u_1\frac{\partial \xi^2}{\partial x_1})\frac{\partial \eta}{\partial \xi^2} = u_2
\end{cases}
$$

Elliptic Grid.

Initial Condition: Step profile with zero velocity (u_1, u_2) and surface pressure Zero and hydrostatic pressure in the fluid domain



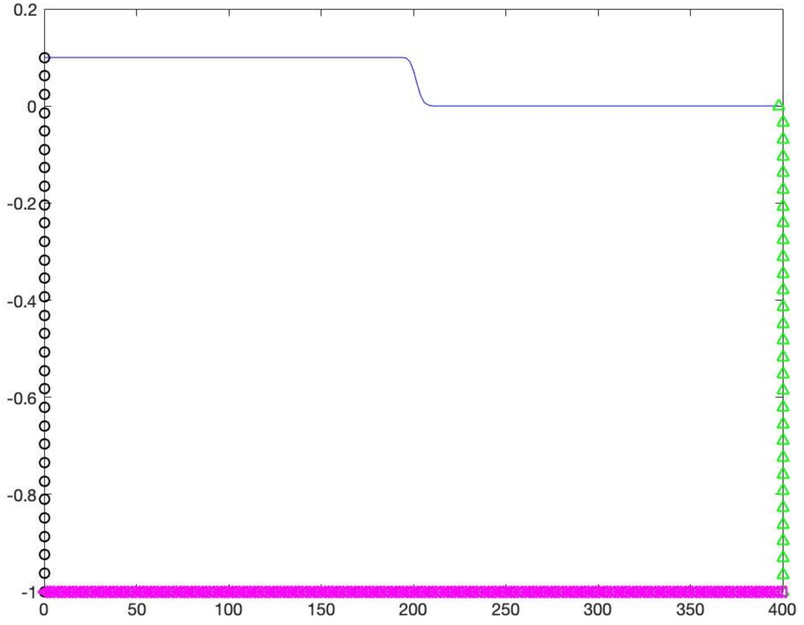
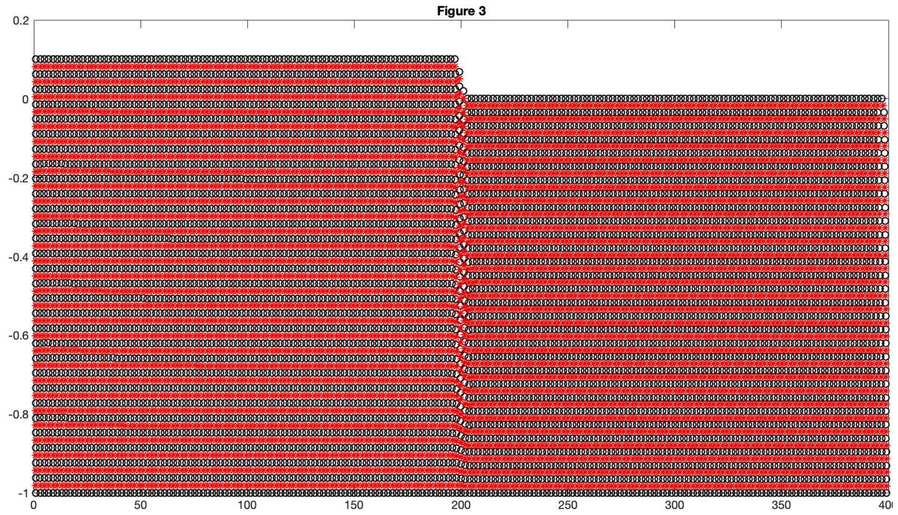
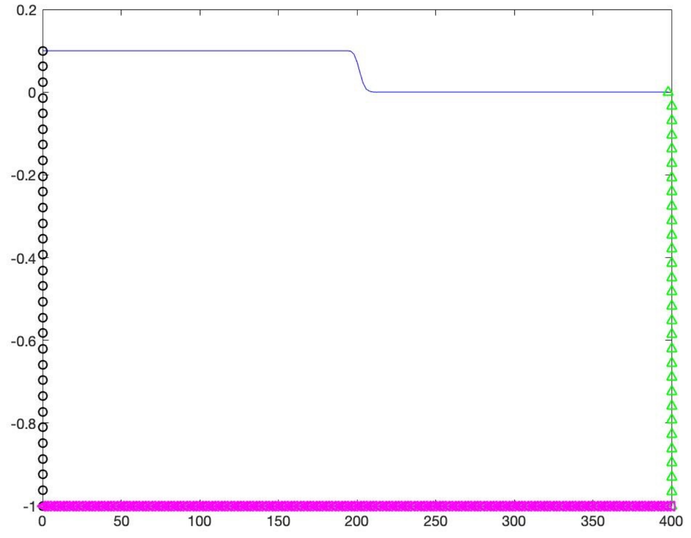
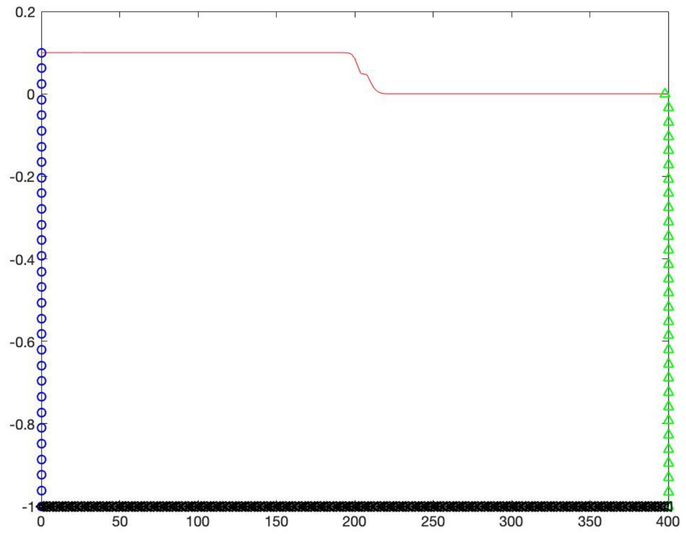


Fig. Physical domain with generalized curvilinear coordinate system in 2D.
Initial STEP profile for the resolution of a DSW in the General Equations of water waves in constant depth.







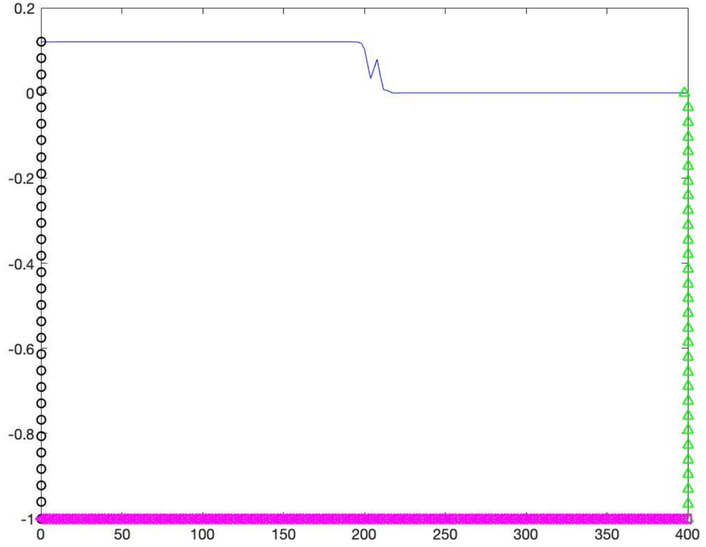
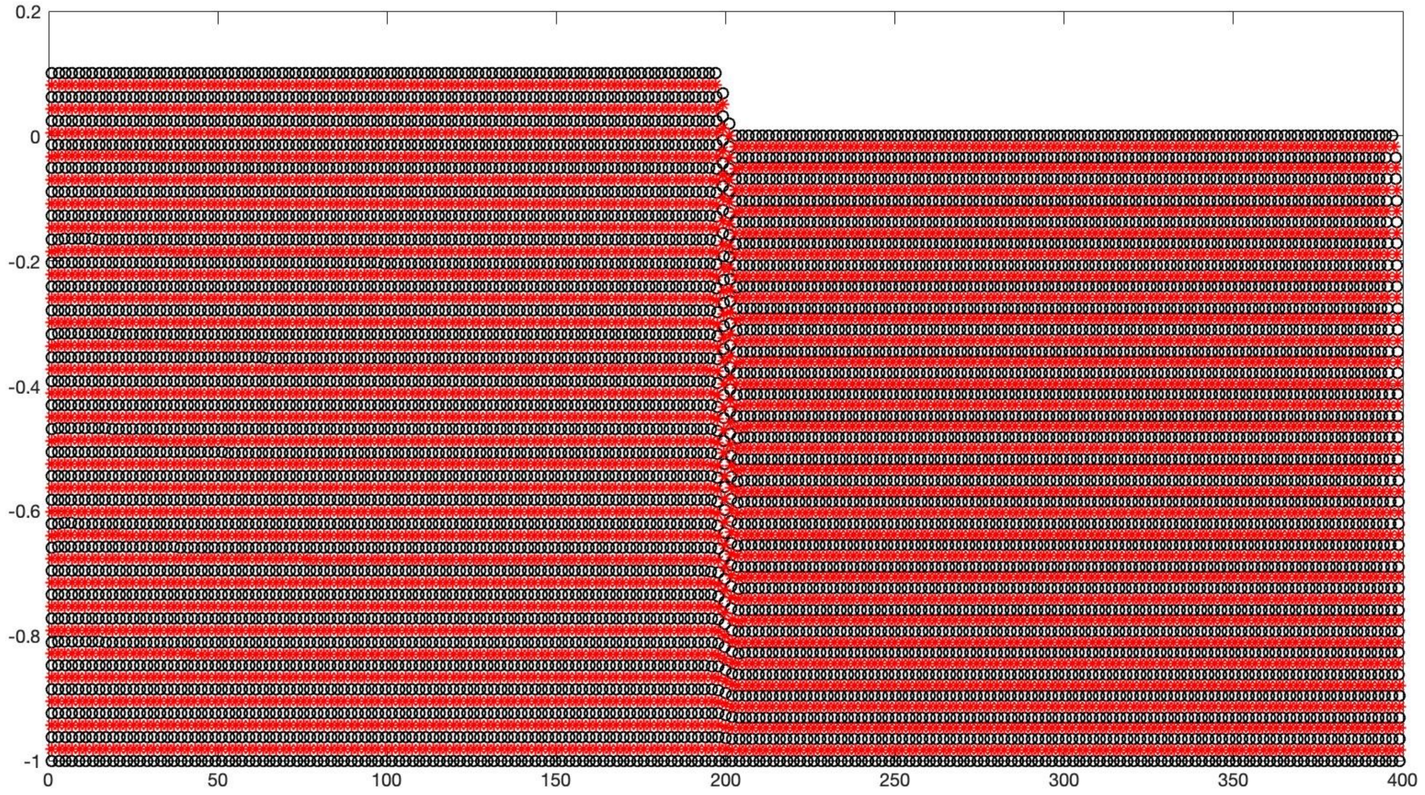
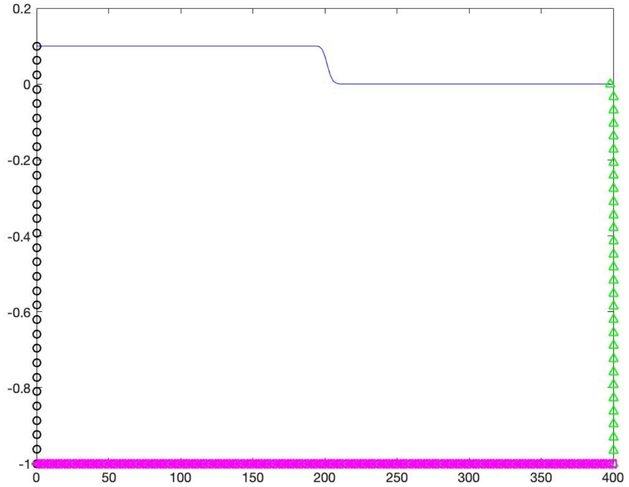
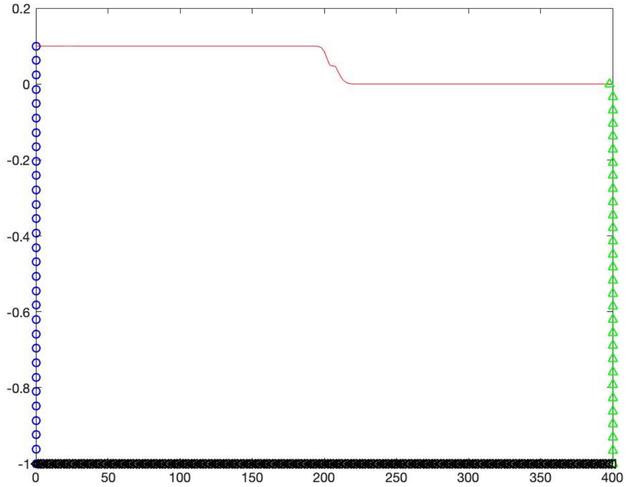
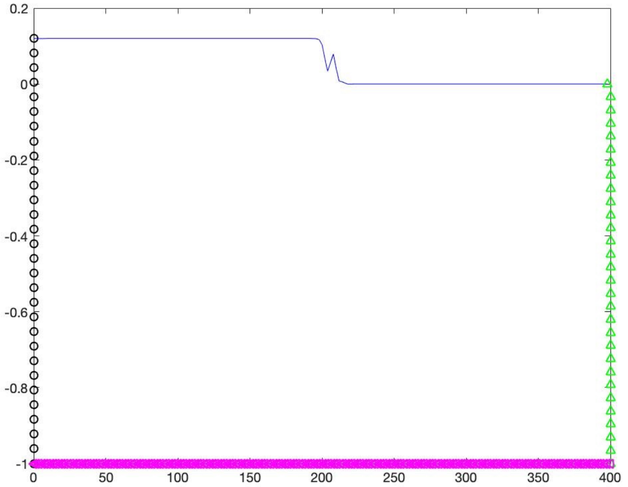


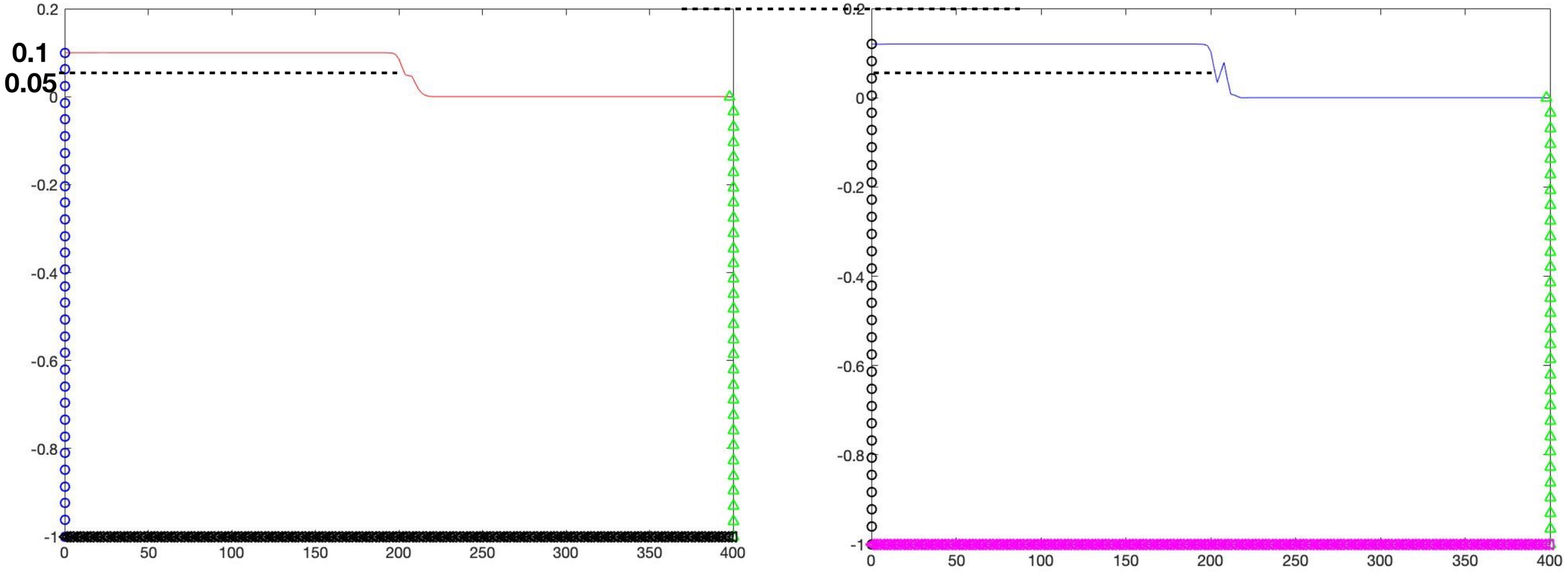
Fig. Physical domain with generalized curvilinear coordinate system in 2D.
Initial STEP profile for the resolution of a DSW in the General Equations of water waves in constant depth.


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400


0.1
0.05
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250
300
350
400

Thank you for your attention

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% .             To define the curvilinear coordinate system:              %
%                    x1(xi1,xi2) and x2=(xi1,xi2)                        %
%                  Initial Profile in [0,x1max]                          %
%             with variable x1 and grid size equal to size(xi1)          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h0=-1;
CL=14;
mlc0=length(x1a);
mlc=CL*length(x1a);
xlc=(0:(x1max/(mlc-1)):x1max)';
xlc0=0:(x1max/(mlc0-1)):x1max;
eta0xlc0=0.08*(1-tanh((xlc0-100)/a1));
eta0 = 0.08*(1-tanh((xlc-100)/a1));
%eta0=(0.3*(sech(0.27*((xlc-(100))+1))).^2)';
%eta0xlc0=(0.3*(sech(0.27*((xlc0-(100))+1))).^2)';
%eta0=(0.1*sin(3*xlc)*sin((pi/2)))';
%%Arc length
dx1=zeros(mlc-1,1);
for k=1:1:mlc-1
    dx1(k)=abs(xlc(k+1)-xlc(k)).^2;
end
lx2=zeros(mlc-1,1);
for k=1:1:mlc-1
    lx2(k)=abs(eta0(k+1)-eta0(k)).^2:
end
lengtheach=zeros(mlc-1,1);
for k=1:1:mlc-1
     lengtheach(k)= sqrt(dx1(k)+lx2(k));
end
  lengthcurve=sum(lengtheach);

  vl0toxa=zeros(mlc-1,1);
  for k=1:1:mlc-1
     vl0toxa(k)=sum(lengtheach(1:k)) ;
  end
  vl0tox=((vertcat(0,vl0toxa)*(Nxi1-1))/lengthcurve);
x20=zeros(Nxi2,Nxi1);
x10=zeros(Nxi2,Nxi1);
  h0v=-1*ones(Nxi2,1);
  etaUP10=eta0(1)*ones(Nxi2,1);
  etaUPend0=eta0(CL*Nxi1)*ones(Nxi2,1);
%%left
for k=1:Nxi2
x20(k,1)=(etaUP10(k)-h0v(k))*(xi2(k)/(Nxi2-1))+h0v(k);
end
for k=1:Nxi2
x10(k,1)=x1a(1);
end
%rigth
for k=1:Nxi2
x20(k,Nxi1)=(etaUPend0(k)-h0v(k))*(xi2(k)/(Nxi2-1))+h0v(k);
end
for k=1:Nxi2
%x10(k,Nxi1)=x1a(x1max);
x10(k,Nxi1)=x1a(x1max);
end
%%bottom
for k=1:Nxi1
x20(1,k)=-1;
end
for k=1:Nxi1
x10(1,k)=x1a(x1max)*(1/(Nxi1-1))*xi1(k);
end
  for k=1:Nxi1
    [am(k),im(k)]=min(abs(xi1(k)-vl0tox(:)));
  end
  for k=1:Nxi1
  xNv0(k)=xlc(im(k));
  end
etaN0=interp1(xlc,eta0',xNv0,'pchip');
 for k=1:Nxi1
    x20(Nxi2,k)=etaN0(k);
 end
for k=1:Nxi1
x10(Nxi2,k)=xNv0(k);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  SOR method to obtain the curvilinear coordinates in the whole domain   %
%      This is an elliptic grid in the physical domain                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


errX0=1;
errY0=1;
err = 1.0e-10;
x0=x10;
y0=x20;
xold=x10;
yold=x20;
iter=0;
while errX0 > err || errY0 > err
    for v=2:Nxi1-1
        for r0=2:Nxi2-1
            alpha10= (1/(2))*(xold(r0+1,v)-x0(r0-1,v));
            alpha20= (1/(2))*(yold(r0+1,v)-y0(r0-1,v));
            gamma10 = (1/(2))*(xold(r0,v+1)-x0(r0,v-1));
            gamma20 = (1/(2))*(yold(r0,v+1)-y0(r0,v-1));
            Alpha0 = alpha10^2 + alpha20^2;
            beta0 = alpha10*gamma10 + alpha20*gamma20;
            gamma0 = gamma10^2 + gamma20^2;
             factor = 1/(4*(Alpha0+gamma0));
            x0(r0,v) = factor*(2*Alpha0*(xold(r0,v+1)+x0(r0,v-1))-beta0*(xold(r0+1,v+1)-↙
x0(r0-1,v+1)-xold(r0+1,v-1)+x0(r0-1,v-1)) + 2*gamma0*(xold(r0+1,v)+x0(r0-1,v)));
            y0(r0,v) = factor*(2*Alpha0*(yold(r0,v+1)+y0(r0,v-1))-beta0*(yold(r0+1,v+1)-↙
y0(r0-1,v+1)-yold(r0+1,v-1)+y0(r0-1,v-1)) + 2*gamma0*(yold(r0+1,v)+y0(r0-1,v)));
        end
    end
    iter=iter+1;
    if mod(iter,60)==0
        iter;
        errX0 = norm(xold-x0,1)/norm(xold,1);
        errY0 = norm(yold-y0,1)/norm(yold,1);
    end
    xold=x0;
    yold=y0;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                The computation of  J^{-1} for each cell                 %
%        This is just the area of the cell We could compute J^{-1}        %
%   using equation od the discriminant and the finite differences given   %
%  above but a more acurate representation is obtained by calculating the %
%  volume of the quadrilateral cell directly. Wich in the 2D planar case  %
%  considered for most of the test calculation in this papers leads to    %
%               \Omega(j,k)= J^{-1}(j,k)= 0.5*/A1-A2/                     %
% A1 is the product of the differences of cross vertex first x and then y %
%                             x_dy_d----x_cy_c                            %
%                                i        i                               %
%                                i        i                               %
%                            x_1y_a----x_by_b                             %
%   A1= (xd-xb)x(ya-yc)                                                   %
%   A2= (yd-yb)x(xa-xc)                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  A1=zeros(Nxi2-1,Nxi1-1);
  for k=1:Nxi1-1
     for j=1:Nxi2-1
         A1(j,k)=((x0(j,k+1)-x0(j+1,k))*(y0(j,k)-y0(j+1,k+1)));
     end
  end
  A2=zeros(Nxi2-1,Nxi1-1);
   for k=1:Nxi1-1
     for j=1:Nxi2-1
         A2(j,k)=((y0(j,k+1)-y0(j+1,k))*(x0(j,k)-x0(j+1,k+1)));
     end
   end
   for k=1:Nxi1-1
     for j=1:Nxi2-1
         Jm10(j,k)=0.5*abs(A1(j,k)-A2(j,k));
     end
   end
  Jm1plusc0=Jm10(:,Nxi1-1);
  Jm129x2500=horzcat(Jm10,Jm1plusc0);
  Jm1plusbr0=Jm129x2500(Nxi2-1,:);
  Jm1plusr0=Jm10(Nxi2-1,:);
  Jm130x2490=vertcat(Jm10,Jm1plusr0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%The Metrics, metric tensor are recalculated according to the new grid %
%      (j,k) centers vertical edges                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

       Jm1dxi1dx29x2500= dydxi20;
       Jm1dxi1dy29x2500=-dxdxi20;
       Jm1dxi2dx30x2490=-dydxi10;
       Jm1dxi2dy30x2490= dxdxi10;

       Jm1dxi1dx29x249l0= dydxi2lc0;
       Jm1dxi1dy29x249l0=-dxdxi2lc0;
       Jm1dxi2dx29x249b0=-dydxi1botr0;
       Jm1dxi2dy29x249b0=dxdxi1botr0;

       Jm1dxi1dx29x249r0= dydxi2rc0;
       Jm1dxi1dy29x249r0=-dxdxi2rc0;
       Jm1dxi2dx29x249t0=-dydxi1topr0;
       Jm1dxi2dy29x249t0=dxdxi1topr0;

       dxi1dxfr0= Jm10.^(-1).*dydxi2rc0;
       dxi1dyfr0=-Jm10.^(-1).*dxdxi2rc0;
       dxi2dxft0=-Jm10.^(-1).*dydxi1topr0;
       dxi2dyft0= Jm10.^(-1).*dxdxi1topr0;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 FLUX TERMS U1 U2 having u1 and u2                        %
% Initial conditions VELOCITIES for u1 and u2 in the  CENTERS of the CELL %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%u10=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
for j= 1:Nxi2-1
u10(j,k)=yc0(j,k)*0;
end
end

%u20=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
for j= 1:Nxi2-1
u20(j,k)=yc0(j,k)*0;
end
end

u1PORJm1p0= Jm10.*u10;
u2PORJm1p0= Jm10.*u20;

%ghost points
u1gl=u10(:,1);
u1gr=u10(:,Nxi1-1);
u1G0=horzcat(u1gl,u10,u1gr);
u2gb=u20(1,:);
u2gt=u20(Nxi2-1,:);
u2G0=vertcat(u2gb,u20,u2gt);

%%%%%Interpolation of u10 and u20 to the right faces
u10f=u1G0(:,1:Nxi1-1)+u1G0(:,2:Nxi1)*0.5;
u20f=u2G0(1:Nxi2-1,:)+u2G0(2:Nxi2,:)*0.5;
 for k=1:Nxi1-1
   for j= 1:Nxi2-1
   vg0(j,k)=1;
   end
 end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%-------->PRESSURES JUST in the BORDER!!!!                   %
%            this eta should be the eta in Coordinate grid centers          %
%%eta0Nt is defined in the physical domain in the cartesian coordinates . %
%        with the curvilinear coordinates x1(xi1,xi2) x2(xi1,xi2)


eta0Nt=etaN0';
xcvC0=horzcat(xc0(1,:),x1a(end));
%xN=xNv(1,:);
%xNv0 are points in x1 that are the closest to the parametrization .
%n I interpolate to the centers
%%%Con ello produzco una nueva malla en cuyos centros definiré las
%%%presiones
[xNv0, index] = unique(xNv0);
eta0Center=interp1(xNv0,eta0Nt(index),xcvC0, 'pchip');
eta0Center(isnan(eta0Center))=0;
gm1=-1;
 pM=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
 for j= 1:Nxi2-1
     pM(j,k)=(eta0Center(k)-yc0(j,k));
 end
 end

 prep=pM(Nxi2-1,:);
pM2 = repmat(prep,Nxi2-1,1);
p0a=pM-pM2;
p0=p0a+(gm1*yc0(1:Nxi2-1,:));
phi0= p0+gm1*ycp0;
 for k=1:Nxi1-1
 for j= 1:Nxi2-1
 etanWhole0(j,k)=ycp0(j,k);
 end
 end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        FLUX TERMS U1 U2                                    %
%          these FLUXES are defined in the center                            %
% The present method requires, in general, the calculation of fluxes        %
%(more precisely,flux Jacobians)with both second and first order spatial     %
%accuracy . First-order accuracy is easily achieved by choosing the cell     %
%face values QL and QR as
%QL_i+1/2= Qi,
%QR_i+1/2= Qi+1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

U10f=dxi1dxfr0.*u10f(:,:)+ dxi1dyfr0.*u20f(:,:);
U20f=dxi2dxft0.*u10f(:,:)+ dxi2dyft0.*u20f(:,:);
%%%%%%%%%%%%%%%%%%%%%%%%Central differences
U1u10=U10f.*u10f;
U1u20=U10f.*u20f;
%% We add initial and final column
U1u1cf0=U1u10(:,1);
U1u1cl0=U1u10(:,Nxi1-1);
U1u10B=horzcat(U1u1cf0,U1u10,U1u1cl0);

U1u2cf0=U1u20(:,1);
U1u2cl0=U1u20(:,Nxi1-1);
U1u20B=horzcat(U1u2cf0,U1u20,U1u2cl0);
%%%%%%
U2u10=U20f.*u10f;
U2u20=U20f.*u20f;

%We add initial and final row
U2u1rf0=U2u10(1,:);
U2u1rl0=U2u10(Nxi2-1,:);
U2u10B=vertcat(U2u1rf0,U2u10,U2u1rl0);
U2u2rf0=U2u20(1,:);
U2u2rl0=U2u20(Nxi2-1,:);
U2u20B=vertcat(U2u2rf0,U2u20,U2u2rl0);

U10Bfirst=U10f(:,1);
U10Blast=U10f(:,Nxi1-2);
U10B=horzcat(U10Bfirst,U10f,U10Blast,U10Blast);
U10BB=horzcat(U10Bfirst,U10f);


U20Bfirst=U20f(1,:);
U20Blast=U20f(Nxi2-2,:);
U20B=vertcat(U20Bfirst,U20f,U20Blast,U20Blast);
U20BB=vertcat(U20Bfirst,U20f);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             QUICK method after getting the left and rigth values of u  %
% in the cell face. If U is constant and the grid spacing dx is constant  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for j=3:Nxi1+2
    for k=1:Nxi2-1
        if  U10B(k,j-2)>0 && U10B(k,j-1)>0
            u10lefta(k,j)=(6/8)*u1B0(k,j-1)+ (3/8)*u1B0(k,j)-(1/8)*u1B0(k,j-2);
        elseif  U10B(k,j-2)<0 && U10B(k,j-1)<0
            u10lefta(k,j)=(6/8)*u1B0(k,j)+ (3/8)*u1B0(k,j-1)-(1/8)*u1B0(k,j+1);
        else
            u10lefta(k,j)=0;
        end
    end
end
u10left=u10lefta(:,3:Nxi1+1);


u20lefta=zeros(Nxi2-1,Nxi1-1);
for j=1:Nxi1-1
    for k=3:Nxi2+2
        if  U20B(k-2,j)>0 && U20B(k-1,j)>0
            u20lefta(k,j)=(6/8)*u2B0(k-1,j)+ (3/8)*u2B0(k,j)-(1/8)*u2B0(k-2,j);
        elseif  U20B(k-2,j)<0 && U20B(k-1,j)<0
            u20lefta(k,j)=(6/8)*u2B0(k,j)+ (3/8)*u2B0(k-1,j)-(1/8)*u2B0(k+1,j);
        else
            u20lefta(k,j)=0;
        end
    end

 end

 u20left=u20lefta(3:Nxi2+1,:);

 %u10righta=zeros(Nxi2+2,Nxi1-1);
 for j=2:Nxi1+1
     for k=1:Nxi2-1
         if  U10B(k,j-1)>0 && U10B(k,j)>0
             u10righta(k,j)=(6/8)*u1B0(k,j)+ (3/8)*u1B0(k,j+1)-(1/8)*u1B0(k,j-1);
         elseif  U10B(k,j-1)<0 && U10B(k,j)<0
             u10righta(k,j)=(6/8)*u1B0(k,j+1)+ (3/8)*u1B0(k,j)-(1/8)*u1B0(k,j+2);
         else
             u10righta(k,j)=0;
         end
     end
 end

 u10right=u10righta(:,2:Nxi1);

 %u20righta=zeros(Nxi2-1,Nxi1-1);
 for j=1:Nxi1-1
     for k=2:Nxi2+1
         if  U20B(k-1,j)>0 && U20B(k,j)>0
             u20righta(k,j)=(6/8)*u2B0(k,j)+ (3/8)*u2B0(k+1,j)-(1/8)*u2B0(k-1,j);
         elseif  U20B(k-1,j)<0 && U20B(k,j)<0
             u20righta(k,j)=(6/8)*u2B0(k+1,j)+ (3/8)*u2B0(k,j)-(1/8)*u2B0(k+2,j);
         else
             u20righta(k,j)=0;
         end
     end
 end

 u20right=u20righta(2:Nxi2,:);

 deltaxi1U1u10=Jm10.*(0.5*((U10f+abs(U10f))+(U10f-abs(U10f))).*u10left- (0.5*(U10f+abs↙
 (U10f))+(U10f-abs(U10f))).*u10right);
 deltaxi2U2u20=Jm10.*(0.5*((U20f+abs(U20f))+(U20f-abs(U20f))).*u20left- (0.5*(U20f+abs↙
 (U20f))+(U20f-abs(U20f))).*u20right);
 deltaxi2U2u10=Jm10.*(0.5*((U20f+abs(U20f))+(U20f-abs(U20f))).*u10left- (0.5*(U20f+abs↙
 (U20f))+(U20f-abs(U20f))).*u10right);
 deltaxi1U1u20=Jm10.*(0.5*((U10f+abs(U10f))+(U10f-abs(U10f))).*u20left- (0.5*(U10f+abs↙
 (U10f))+(U10f-abs(U10f))).*u20right);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Convective terms                                 %
% Compute coefficients for dcu1x,dcu2y, hdfu1x,hdfu1xt2,hdfu2y,hdfu2yt2   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%FLUX TERMS U1 U2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%FLUX TERMS U1 U2 these FLUXES are defined in the center.               %
%The present method requires, in general, the calculation of fluxes       %
%(more precisely, flux Jacobians)
 %ghost points
 u1gl=u1(:,1,i);
 u1gr=u1(:,Nxi1-1,i);
 u1G=horzcat(u1gl,u1(:,:,i),u1gr);
 %
 u2gb=u2(1,:,i);
 u2gt=u2(Nxi2-1,:,i);
 u2G=vertcat(u2gb,u2(:,:,i),u2gt);
 %
 %%%%lets interpolate u10 and u20 to the right faces
 %%%First we need to interpolate u1f and u2f
u1f=u1G(:,1:Nxi1-1)+u1G(:,2:Nxi1)*0.5;
u2f=u2G(1:Nxi2-1,:)+u2G(2:Nxi2,:)*0.5;
% % % % U1(:,:,i)=U1c(:,1:Nxi1-2)+U1c(:,2:Nxi1-1)*0.5;
% % % % U2(:,:,i)=U2c(1:Nxi2-2,:)+U2c(2:Nxi2-1,:)*0.5;
U1(:,:,i)=dxi1dxfr(:,:,i-1).*u1f+ dxi1dyfr(:,:,i).*u2f;
U2(:,:,i)=dxi2dxft(:,:,i-1).*u1f+ dxi2dyft(:,:,i).*u2f;
%%%%%Para NOEL
U1u1=U1.*u1f;
U1u2=U1.*u2f;
%%agregar columna inicial y final
U1u1cf=U1u1(:,1,i);
U1u1cl=U1u1(:,Nxi1-1,i);
%
U1u1B=horzcat(U1u1cf,U1u1(:,:,i), U1u1cl);
%
U1u2cf=U1u2(:,1,i);
U1u2cl=U1u2(:,Nxi1-1,i);
%
U1u2B=horzcat(U1u2cf,U1u2(:,:,i), U1u2cl);
%%%%%%
U2u1=U2.*u1f;
U2u2=U2.*u2f;
%agregar fila inicial y final
U2u1rf=U2u1(1,:,i);
U2u1rl=U2u1(Nxi2-1,:,i);
%
U2u1B=vertcat(U2u1rf,U2u1(:,:,i), U2u1rl);
%
U2u2rf=U2u2(1,:,i);
U2u2rl=U2u2(Nxi2-1,:,i);
%
U2u2B=vertcat(U2u2rf,U2u2(:,:,i), U2u2rl);
```

```
%%% Now we will interpolate the fluxes to the face of the cell grid.
U1Bf=U1(:,1,i);
U1Bl=U1(:,Nxi1-2,i);
U1B=horzcat(U1Bf,U1(:,:,i),U1Bl,U1Bl);
%
U1BB=horzcat(U1Bf,U1(:,:,i));
%
U2Bf=U2(1,:,i);
U2Bl=U2(Nxi2-2,:,i);
%U20B=vertcat(U20Bf,U20Bf,U20Bf,U20Bf,U20,U20Bl,U20Bl);
U2B=vertcat(U2Bf,U2(:,:,i),U2Bl,U2Bl);
%
U2BB=vertcat(U2Bf,U2(:,:,i));
%%%Now we will use the QUICK method but first we should obtain the left and
%%%rigth values of u in the cell face
%If U is constant and the grid spacing dx is constant,then
%u10lefta=zeros(Nxi2-1,Nxi1-2);
for j=3:Nxi1+2
    for k=1:Nxi2-1
        if  U1B(k,j-2)>0 && U1B(k,j-1)>0
            u1lefta(k,j)=(6/8)*u1B(k,j-1)+ (3/8)*u1B(k,j)-(1/8)*u1B(k,j-2);
        elseif  U1B(k,j-2)<0 && U1B(k,j-1)<0
            u1lefta(k,j)=(6/8)*u1B(k,j)+ (3/8)*u1B(k,j-1)-(1/8)*u1B(k,j+1);
        else
            u1lefta(k,j)=0;
        end
    end
end
u1left=u1lefta(:,3:Nxi1+1);
%u2lefta=zeros(Nxi2-1,Nxi1-1);
for j=1:Nxi1-1
    for k=3:Nxi2+2
        if  U2B(k-2,j)>0 && U2B(k-1,j)>0
            u2lefta(k,j)=(6/8)*u2B(k-1,j)+ (3/8)*u2B(k,j)-(1/8)*u2B(k-2,j);
        elseif  U2B(k-2,j)<0 && U2B(k-1,j)<0
            u2lefta(k,j)=(6/8)*u2B(k,j)+ (3/8)*u2B(k-1,j)-(1/8)*u2B(k+1,j);
        else
            u2lefta(k,j)=0;
        end
    end
end
%
u2left=u2lefta(3:Nxi2+1,:);
%u10righta=zeros(Nxi2+2,Nxi1-1);
for j=2:Nxi1+1
    for k=1:Nxi2-1
        if  U1B(k,j-1)>0 && U1B(k,j)>0
            u1righta(k,j)=(6/8)*u1B(k,j)+ (3/8)*u1B(k,j+1)-(1/8)*u1B(k,j-1);
        elseif  U1B(k,j-1)<0 && U1B(k,j)<0
            u1righta(k,j)=(6/8)*u1B(k,j+1)+ (3/8)*u1B(k,j)-(1/8)*u1B(k,j+2);
        else
            u1righta(k,j)=0;
        end
    end
end
u1right=u1righta(:,2:Nxi1);
%u20righta=zeros(Nxi2-1,Nxi1-1);
for j=1:Nxi1-1
    for k=2:Nxi2+1
        if  U2B(k-1,j)>0 && U2B(k,j)>0
            u2righta(k,j)=(6/8)*u2B(k,j)+ (3/8)*u2B(k+1,j)-(1/8)*u2B(k-1,j);
        elseif  U2B(k-1,j)<0 && U2B(k,j)<0
            u2righta(k,j)=(6/8)*u2B(k+1,j)+ (3/8)*u2B(k,j)-(1/8)*u2B(k+2,j);
        else
            u2righta(k,j)=0;
        end
    end
end
u2right=u2righta(2:Nxi2,:);
%
deltaxi1U1u1(:,:,i)=Jm1(:,:,i).*(0.5*((U1(:,:,i)+abs(U1(:,:,i)))+(U1(:,:,i)-abs(U1(:,:,↙
i)))).*u10right- (0.5*((U1(:,:,i)+abs(U1(:,:,i)))+(U1(:,:,i)-abs(U1(:,:,i)))).*u1left));
deltaxi2U2u2(:,:,i)=Jm1(:,:,i).*(0.5*((U2(:,:,i)+abs(U2(:,:,i)))+(U2(:,:,i)-abs(U2(:,:,↙
i)))).*u20right- (0.5*((U2(:,:,i)+abs(U2(:,:,i)))+(U2(:,:,i)-abs(U2(:,:,i)))).*u2left));
%
deltaxi2U2u1(:,:,i)=Jm1(:,:,i).*(0.5*((U2(:,:,i)+abs(U2(:,:,i)))+(U2(:,:,i)-abs(U2(:,:,↙
i)))).*u10right- (0.5*((U2(:,:,i)+abs(U2(:,:,i)))+(U2(:,:,i)-abs(U2(:,:,i)))).*u1left));
%
deltaxi1U1u2(:,:,i)=Jm1(:,:,i).*(0.5*((U1(:,:,i)+abs(U1(:,:,i)))+(U1(:,:,i)-abs(U1(:,:,↙
i)))).*u20right- (0.5*((U1(:,:,i)+abs(U1(:,:,i)))+(U1(:,:,i)-abs(U1(:,:,i)))).*u2left));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```