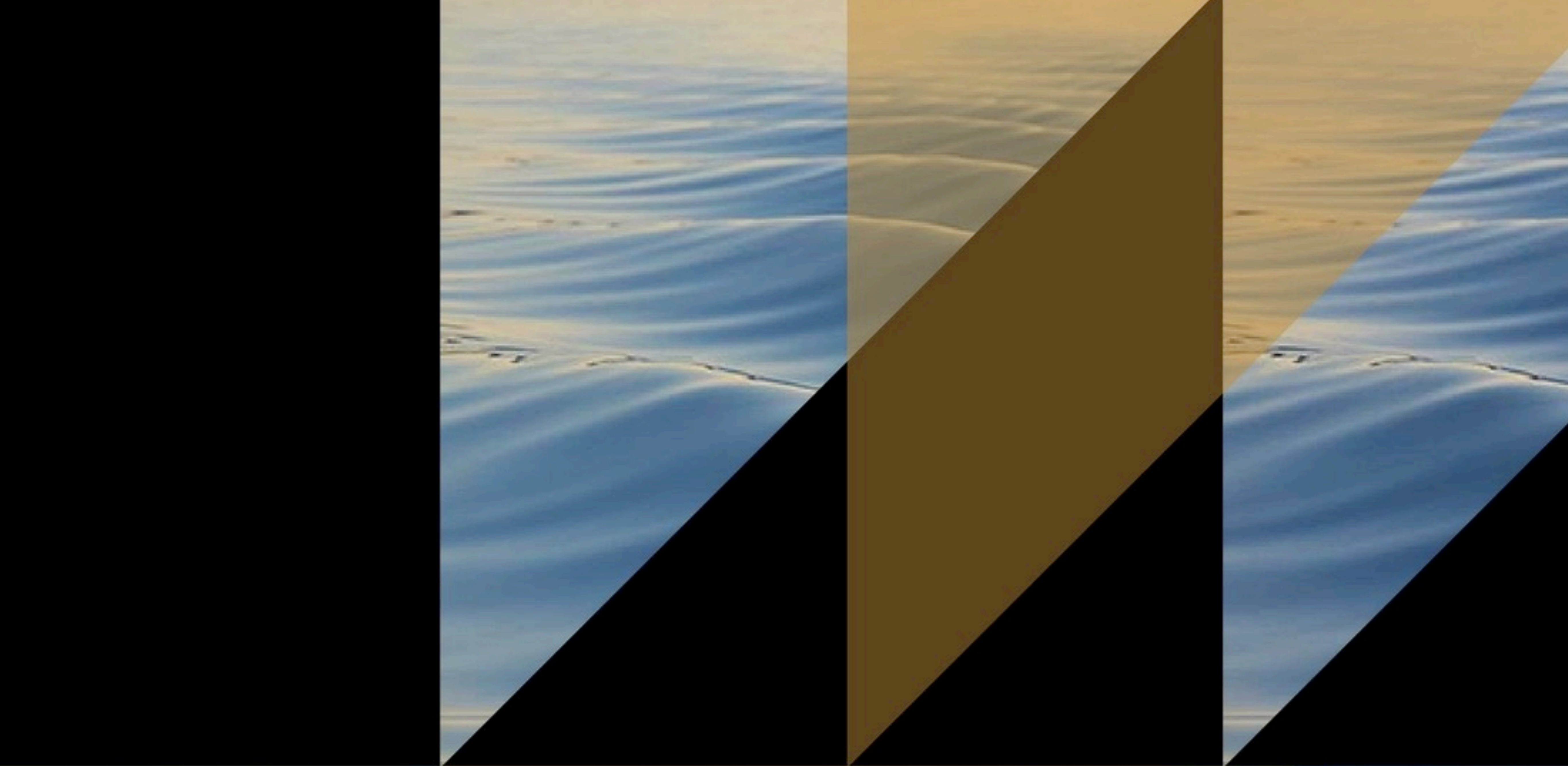


La Modelación de las Ecuaciones de Euler en un fluido con superficie libre en un sistema de coordenadas curvilíneo generalizado en 2D descrito en 10 pasos

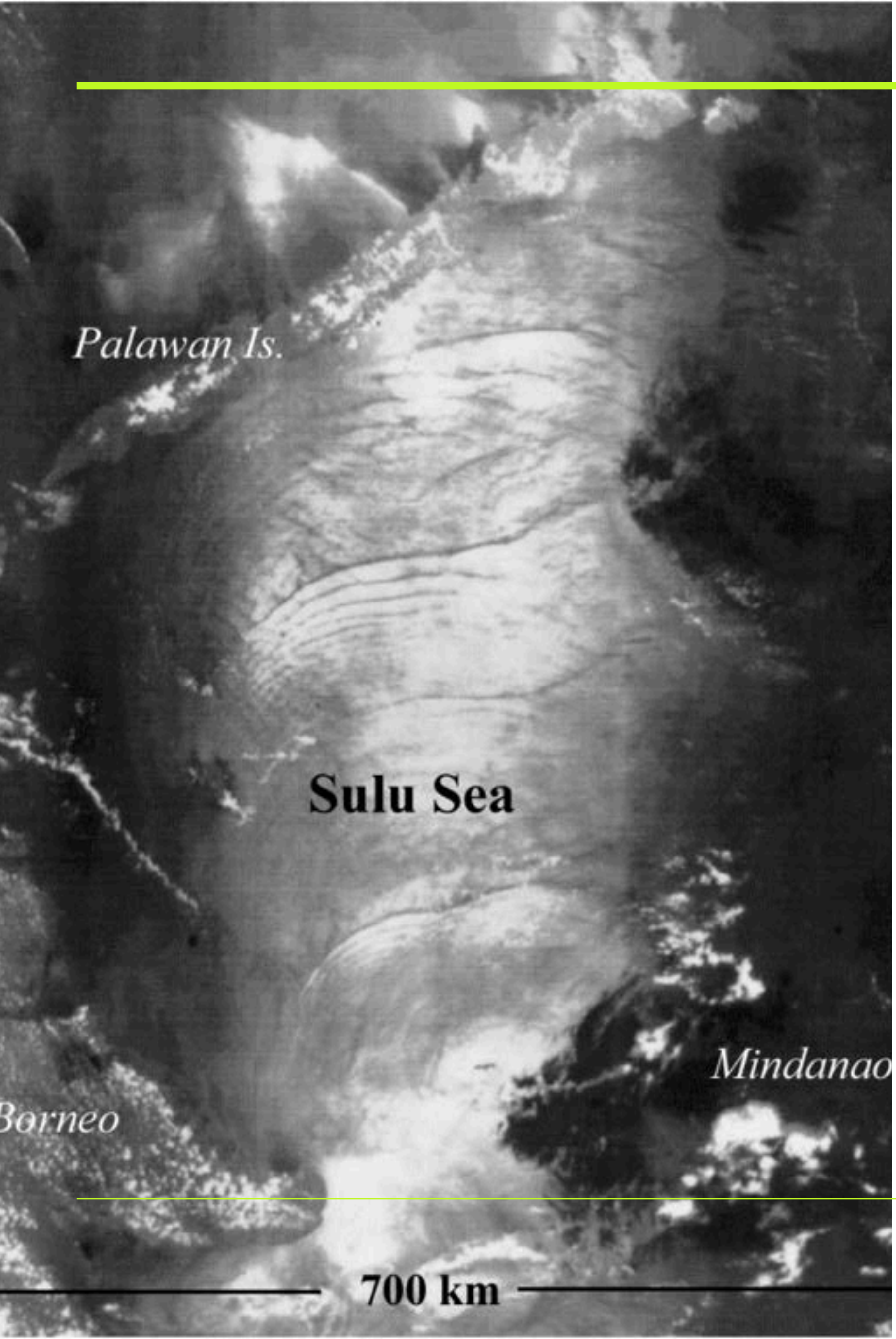
ROSA MARIA VARGAS MAGAÑA

AÑO 2021

https://rosavargas.github.io/

Curso dirigido a estudiantes e Investigadores Interesados en la Modelación de la Mecánica de Fluidos.

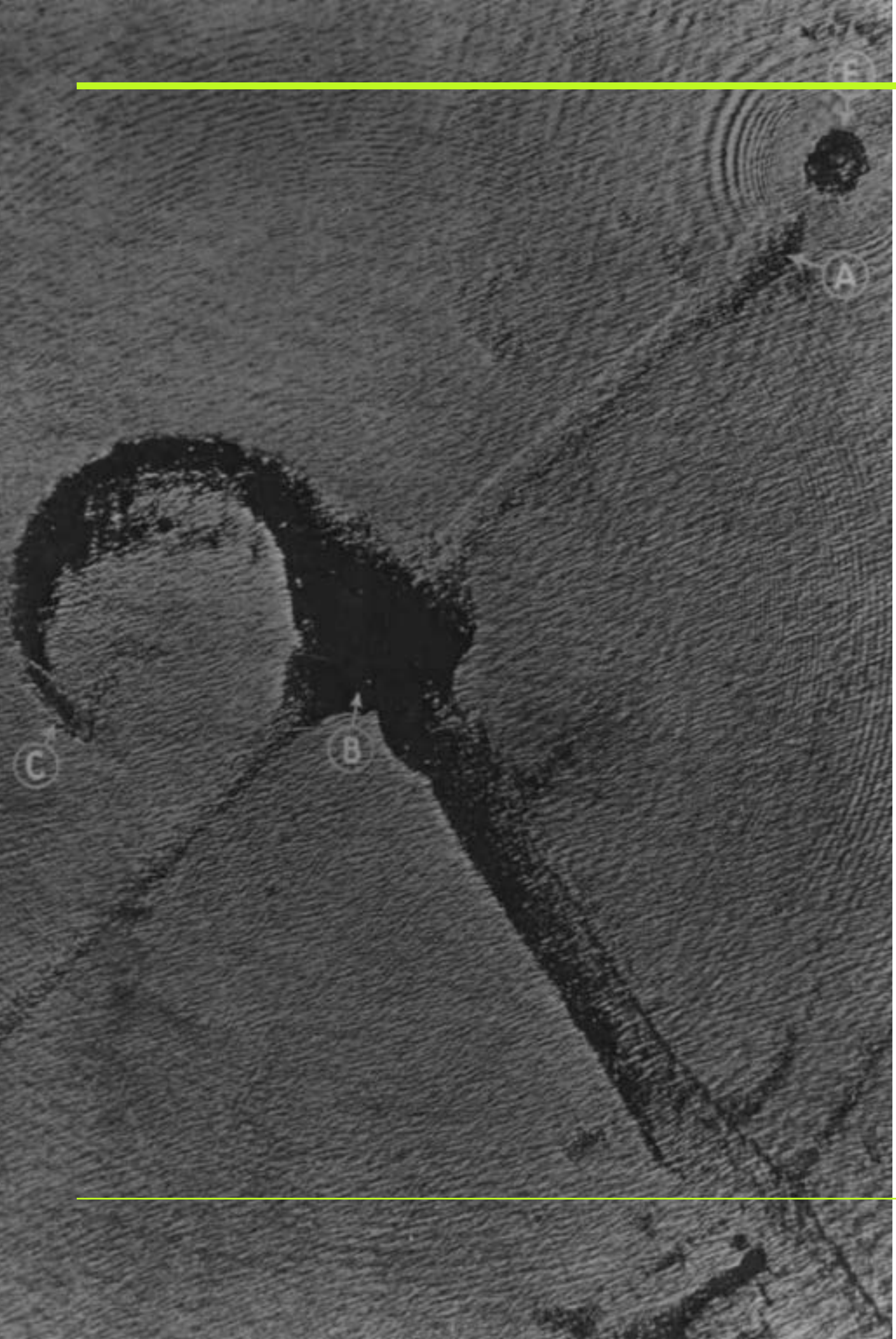
El curso es auto contenido y no requiere experiencia previa en programación.



Importancia

de la modelación de las ecuaciones de Euler de superficie libre

Los métodos numéricos son cruciales para la comprensión y la solución de problemas del mundo real. La dinámica de fluidos computacional debe incluir esquemas numéricos de alto orden, modelado preciso de superficies libres y buena escalabilidad de los esquemas numéricos y que sean de alto rendimiento. La versatilidad en la generación de mallas computacionales también es importante para geometrías complejas como las que se observan en la hidrodinámica de los barcos. Los códigos de Euler requieren almacenamiento y trabajo computacional para su solución, incluso con la clase actual de supercomputadoras, todavía no hemos llegado a una etapa en la que se puedan ignorar las restricciones de velocidad y almacenamiento computacionales. Es por todos los motivos anteriormente enunciados que la construcción de códigos de Euler es un área activa de investigación y de creación.



Objetivo:

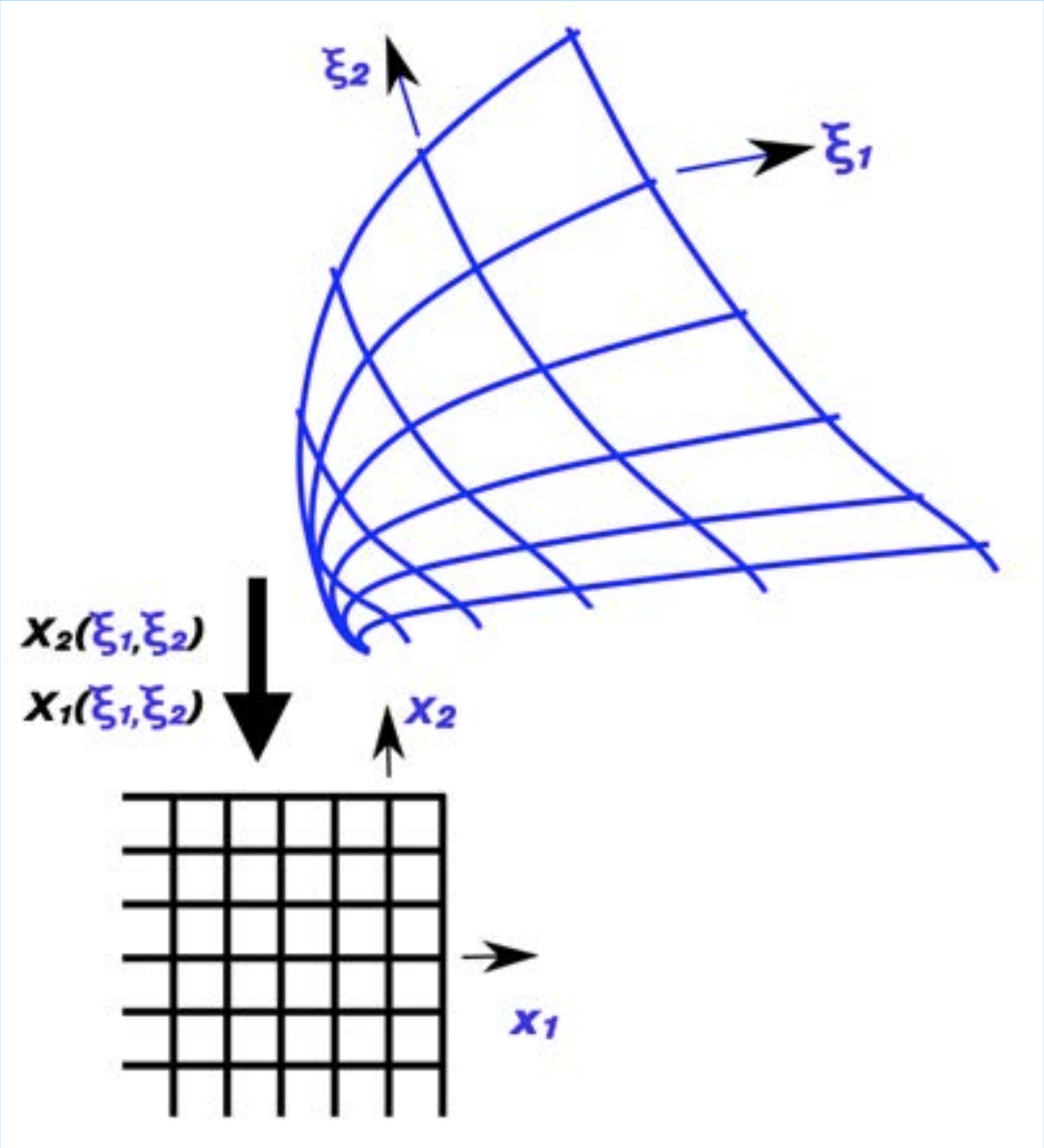
La implementación de un nuevo enfoque basado en el método de diferencias finita, predictor-corrector descrito en un sistema de coordenadas curvilíneas para estudiar la propagación de ondas de superficie y que ha sido probado su eficacia.

Puesto que la superficie del fluido está libre, estas ecuaciones se establecen en un sistema de coordenadas curvilíneas (no ortogonales) para que la superficie se encuentre en una dirección de coordenada.

Numerical study of nonlinear shallow water waves produced by a submerged moving disturbance in viscous flow
Cite as: Physics of Fluids 8, 147 (1996); https://doi.org/10.1063/1.868822
Daohua Zhang, and Allen T.

◆

On solitary waves forced by underwater moving objects
By DAO HA ZHAN G AND A L LENT. CHWAN G
Department of Mechanical Engineering, The University of Hong Kong,



Representación de Ecuaciones en coordenadas curvilíneas generalizadas en 2D

Ecuaciones en coordenadas curvilíneas generalizadas

Pasos a seguir:

- **Paso 1 : Escribir las Ecuaciones de Euler en su forma adimensional.**
- **Paso 2 : Escribir las ecuaciones obtenidas en paso 1 en su forma fuerte de Ley de conservación**

ECUACIONES FORMA I

$$
\begin{cases}
i. \ \dfrac{\partial \tilde{u}_1}{\partial \tilde{x}_1} + \dfrac{\partial \tilde{u}_2}{\partial \tilde{x}_2} = 0 & -\tilde{h}_0 < \tilde{x}_2 < \tilde{\eta}, \\
ii. \ \dfrac{\partial \tilde{u}_1}{\partial \tilde{t}} + \dfrac{\partial (\tilde{u}_1 \tilde{u}_1)}{\partial \tilde{x}_1} + \dfrac{\partial (\tilde{u}_2 \tilde{u}_1)}{\partial \tilde{x}_2} = -\dfrac{1}{\rho_0} \dfrac{\partial p}{\partial \tilde{x}_1} & -\tilde{h}_0 < \tilde{x}_2 < \tilde{\eta}, \\
iii. \ \dfrac{\partial \tilde{u}_2}{\partial \tilde{t}} + \dfrac{\partial (\tilde{u}_1 \tilde{u}_2)}{\partial \tilde{x}_1} + \dfrac{\partial (\tilde{u}_2 \tilde{u}_2)}{\partial \tilde{x}_2} = -\dfrac{1}{\rho_0} \dfrac{\partial p}{\partial \tilde{x}_2} - \dfrac{\partial g x2}{\partial \tilde{x}_2} & -\tilde{h}_0 < \tilde{x}_2 < \tilde{\eta}, \\
iv. \ \tilde{u}_2 = 0 & \text{in } \tilde{x}_2 = -\tilde{h}_0 \\
iv. \ \tilde{u}_1 = 0 & \text{in } \tilde{x}_1 = 0 \\
iv. \ \tilde{u}_1 = 0 & \text{in } \tilde{x}_1 = \tilde{x}_{max} \\
iv. \dfrac{\partial \tilde{\eta}}{\partial \tilde{t}} + \tilde{u}_1 \dfrac{\partial \tilde{\eta}}{\partial \tilde{x}_1} = \tilde{u}_2 & \tilde{x}_2 = \eta(\tilde{x}_1, t)
\end{cases}
$$

donde i, j = 1, 2; u_i representa los componentes de la velocidad cartesiana; *p* es la presión cinemática; y g_i representa los componentes de la gravedad.
A continuación, el espacio físico se denota por *(x, y)* y el espacio computacional por *(ξ, η)*. Hemos asumido que la densidad del agua se considera idénticamente igual a uno y que el flujo es irrotacional e invisible.

Ecuaciones de movimiento

Ecuaciones de Euler de superficie libre

El problema de las ondas de agua, para un fluido invisible, incompresible y homogéneo bajo el efecto de una presión superficial constante y una fuerza de gravedad volumétrica son las bien conocidas ecuaciones de Euler en un dominio de fluidos de superficie libre. Aquí se presentan las ecuaciones de Euler y de continuidad en la forma de densidad constante:

Ecuaciones en coordenadas curvilíneas generalizadas

Pasos a seguir:

- **Paso 1 : Escribir las Ecuaciones de Euler en su forma adimensional.**
- **Paso 2 : Escribir las ecuaciones obtenidas en paso 1 en su forma fuerte de Ley de conservación**

Ecuaciones en forma adimensional

ECUACIONES FORMA II

Las ecuaciones de Euler en su forma adimensional son:

$$\text{i.}\quad \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}+\frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}=\frac{\partial(\sqrt{gh_0}u_1)}{\partial(h_0x_1)}+\frac{\partial(\sqrt{gh_0}u_2)}{\partial(h_0x_2)}=\frac{\sqrt{gh_0}}{h_0}\frac{\partial u_1}{\partial x_1}+\frac{\sqrt{gh_0}}{h_0}\frac{\partial u_2}{\partial x_2}$$
$$=\sqrt{\frac{g}{h_0}}\frac{\partial u_1}{\partial x_1}+\sqrt{\frac{g}{h_0}}\frac{\partial u_2}{\partial x_2}=0.$$

$$\text{ii.}\quad \frac{\partial \tilde{u}_1}{\partial \tilde{t}}+\frac{\partial(\tilde{u}_1\tilde{u}_1)}{\partial \tilde{x}_1}+\frac{\partial(\tilde{u}_2\tilde{u}_1)}{\partial \tilde{x}_2}+\frac{1}{\rho_0}\frac{\partial \tilde{p}}{\partial \tilde{x}_1}=\frac{\partial(\sqrt{gh_0}u_1)}{\partial(\sqrt{\frac{h_0}{g}}t)}+\frac{\partial(gh_0u_1^2)}{\partial(h_0x_1)}+\frac{\partial(gh_0u_2u_1)}{\partial(h_0x_2)}+gh_0\frac{\partial p}{\partial(h_0x_1)}$$

$$=g\frac{\partial u_1}{\partial t}+g\frac{\partial u_1^2}{\partial x_1}+g\frac{\partial u_2u_1}{\partial x_2}+g\frac{\partial p}{\partial x_1}=\frac{\partial u_1}{\partial t}+\frac{\partial u_1^2}{\partial x_1}+\frac{\partial u_2u_1}{\partial x_2}+\frac{\partial p}{\partial x_1}=0$$

Para convertir a la forma adimensional las ecuaciones anteriores, sea h_0 una profundidad de canal típica, l una longitud de onda típica de las ondas de agua. Las variables adimensionales están dadas por :

$$x_1=\frac{\tilde{x}_1}{h_0},\ x_2=\frac{\tilde{x}_2}{h_0},\ t=\sqrt{\frac{g}{h}}\tilde{t},\ \eta=\frac{\tilde{\eta}}{h_0}$$

donde $c_0=\sqrt{\frac{g\tanh(kh_0)}{k}}$ es la velocidad de fase.

$$\omega^2(k)=gk\tanh(kh_0)$$

Para las velocidades tenemos:

$$\tilde{u}_1=\sqrt{gh_0}u_1 \text{ and } \tilde{u}_2=\sqrt{gh_0}u_2$$

Ecuaciones en forma adimensional

iii.

$$\frac{\partial \tilde{u}_2}{\partial \tilde{t}} + \frac{\partial(\tilde{u}_1\tilde{u}_2)}{\partial \tilde{x}_1} + \frac{\partial(\tilde{u}_2\tilde{u}_2)}{\partial \tilde{x}_2} + \frac{1}{\rho_0}\frac{\partial p}{\partial \tilde{x}_2} + g$$

$$= \frac{\partial(\sqrt{gh_0}u_2)}{\partial(\sqrt{\frac{h_0}{g}}t)} + \frac{\partial(gh_0u_1u_2)}{\partial(h_0x_1)} + \frac{\partial(gh_0u_2^2)}{\partial(h_0x_2)} + gh_0\frac{\partial p}{\partial(h_0x_1)} + g$$

$$= g\frac{\partial u_2}{\partial t} + g\frac{\partial(u_1u_2)}{\partial x_1} + g\frac{\partial u_2^2}{\partial x_2} + g\frac{\partial p}{\partial x_1} + g$$

$$= \frac{\partial u_2}{\partial t} + \frac{\partial(u_1u_2)}{\partial x_1} + \frac{\partial u_2^2}{\partial x_2} + \frac{\partial p}{\partial x_1} + 1 = 0$$

iv.

$$\frac{\partial \tilde{\eta}}{\partial \tilde{t}} + \tilde{u}_1\frac{\partial \tilde{\eta}}{\partial \tilde{x}_1} = \tilde{u}_2 = \frac{\partial(h_0\eta)}{\partial(\sqrt{\frac{h_0}{g}}t)} + \sqrt{gh_0}u_1\frac{\partial(h_0\eta)}{\partial(h_0x_1)} = \sqrt{gh_0}u_2$$

$$= \sqrt{gh_0}\frac{\partial \eta}{\partial t} + \sqrt{gh_0}u_1\frac{\partial \eta}{\partial x_1} = \sqrt{gh_0}u_2$$

$$= \frac{\partial \eta}{\partial t} + u_1\frac{\partial \eta}{\partial x_1} = u_2$$

Ecuaciones en su forma fuerte de ley de conservación

ECUACIONES FORMA III

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

El siguiente paso es transformar las ecuaciones en su forma II en coordenadas curvilíneas (no ortogonal) . De esta forma la superficie libre se encuentra en una dirección coordenada. El sistema de coordenadas se determina como parte de la solución numérica para hacer eso. necesitamos escribir las ecuaciones en su forma II en forma de ley de conservación fuerte.

- donde la condición de frontera para *f* en la superficie es como $\varphi_0 = p_0 + x_2$, donde $p_0 = 0$ en la superficie.

- La condición inicial considerada aquí es un paso inicial discontinuo. Condición dada por :

$$\eta_0(x,0) = \{A;\ x < 0;\ \ 0;\ x > 0$$

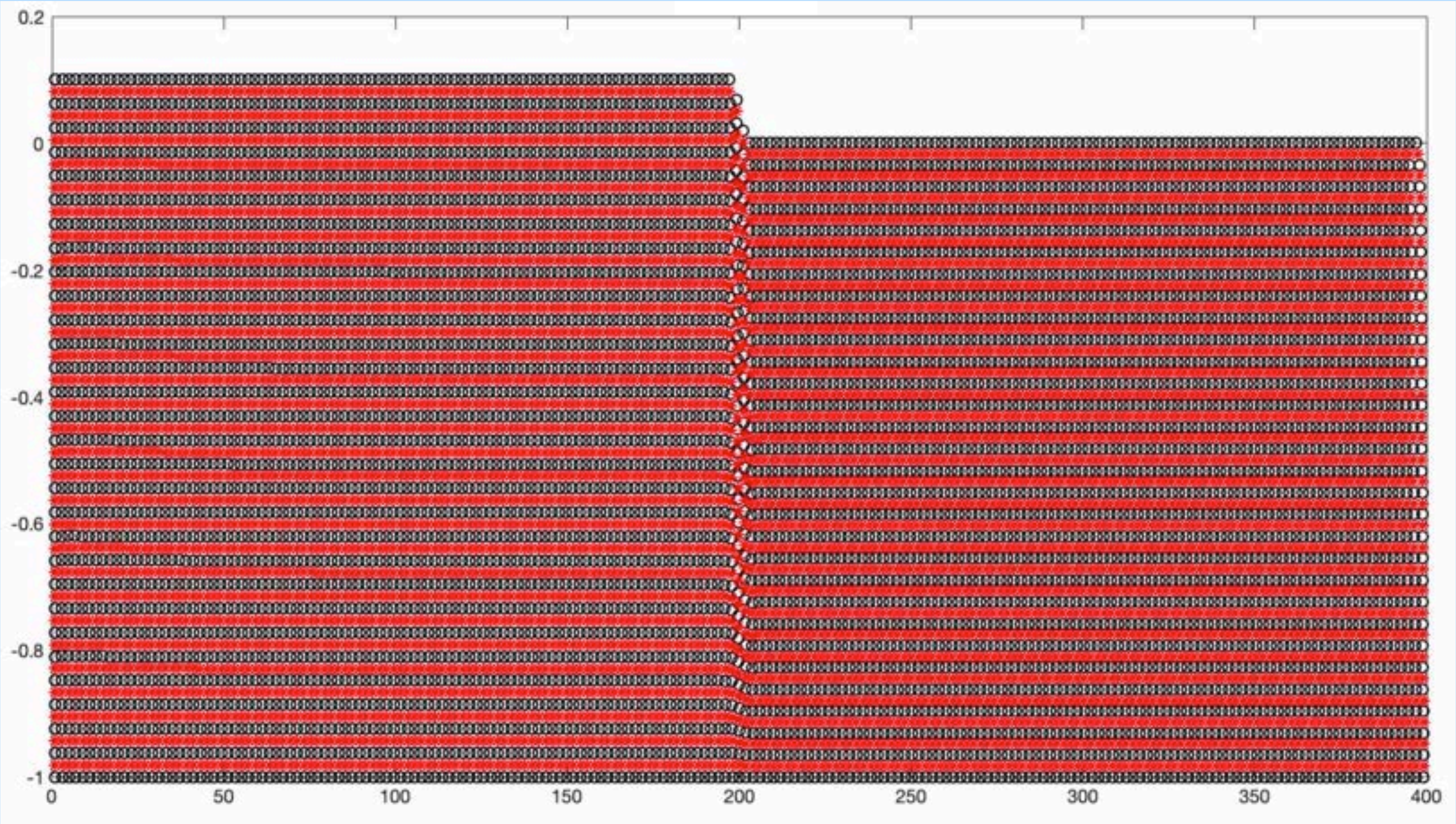
Entonces, las condiciones iniciales para p_0 y f en todo el dominio de la red son:

- p_0 hidrostático y
- $\varphi_0 = p_0 + x_2$ donde J^{-1} es la inversa del Jacobiano o el volumen de la celda;

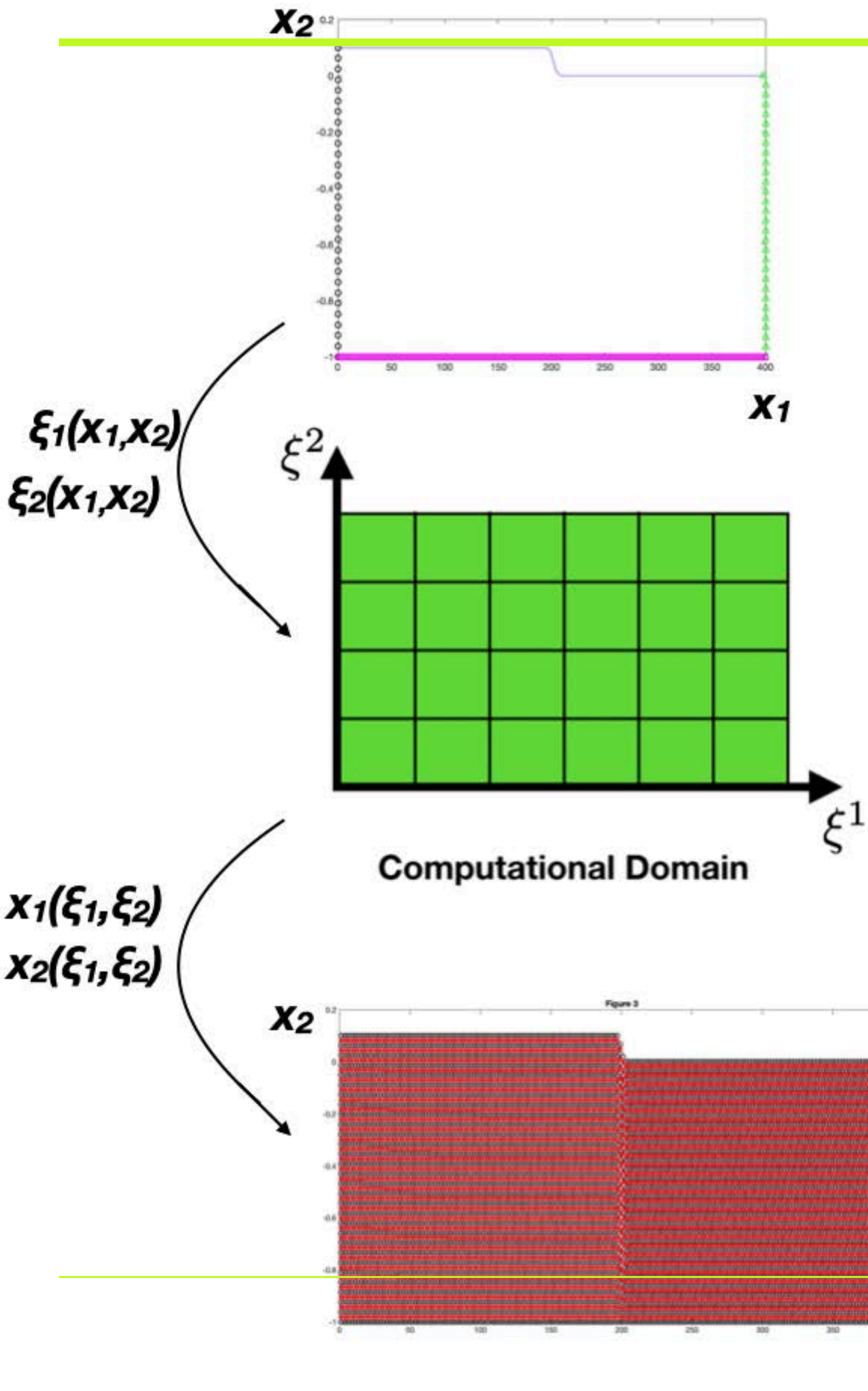
- U_m = es el flujo de volumen (velocidad contravariante multiplicada por J^{-1}) normal a la superficie de constante ξ;

- G^{mn} es el "tensor de sesgo de la malla”.

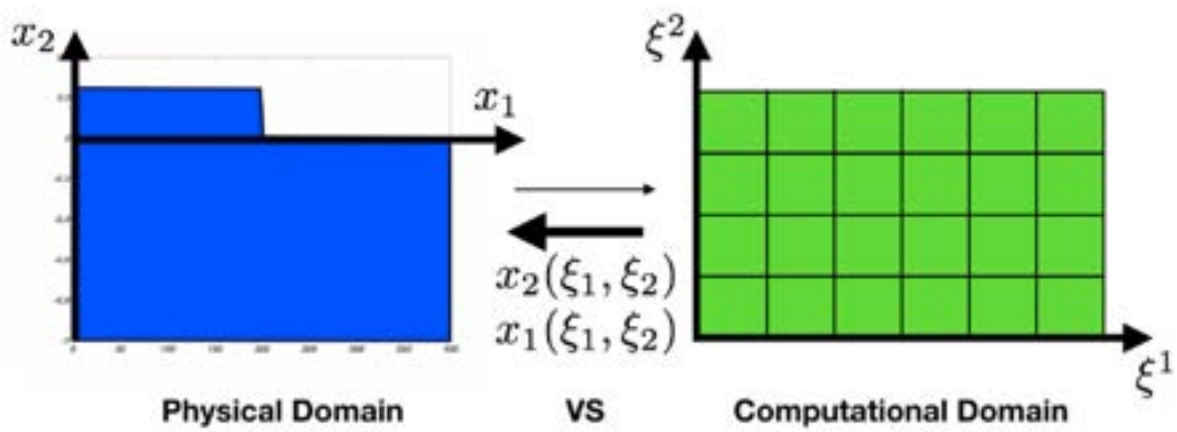
Resolvemos las ecuaciones gobernantes que describen el movimiento del fluido en el transformado plano por el método de diferencias finitas.



Generación del dominio computacional



La idea básica de las coordenadas ajustadas a la frontera del dominio es transformar los límites físicos de un problema en “curvas” coordinadas en el espacio mapeado. En donde los cálculos en diferencias finitas se pueden realizar en un sistema de malla regular sin la necesidad de interpolar a lo largo de la frontera que es en donde se presenta los comportamientos mas dramáticos.
Si (ξ,η) representan las coordenadas en el dominio computacional, bidimensional, la malla completa corresponde como la solución de la ecuación de Poisson que consiste en un sistema de ecuaciones elípticas.



En otras palabras, para modelar la ecuación en un sistema de coordenadas curvilíneas generalizadas en 2D, usamos una transformación del dominio físico en el dominio computacional (una cuadrícula uniforme) .
Determinamos esta transformación implícitamente resolviendo una ecuación de Poisson en la cuadrícula elíptica del dominio físico en términos de las coordenadas curvilíneas que consiste en un sistema de ecuaciones elípticas

Discretización de las Ecuaciones en coordenadas curvilíneas generalizadas

1. La **presión** y los **componentes de la velocidad cartesiana** se definen en el centro de la celda

2. Los **flujos de volumen** se definen en el punto medio de sus caras correspondientes del control volumen en el **espacio computacional.**

3. Las **derivadas espaciales** se discretizan usando diferencias centrales de segundo orden.

4. Con la excepción de los **términos convectivos,** que se discretizan utilizando una variación del esquema **QUICK el cual calcula el valor en la cara de la celda desde el valor nodal con un esquema "quadratic upwind interpolation"**

5. Los esquemas se llevan a cabo calculando flujos de volumen negativos y positivos

6. El término de **flujo convectivo** ***Uu*** en la **cara de la celda** tiene una expresión detallada.

Procedimiento de solución numérica de las EE

Descripción de la sucesión de pasos a seguir para integrar el sistema de movimiento fluido y superficie libre desde el paso de tiempo t_n hasta t_{n+1} es el siguiente.

- ***Definición del dominion físico y del domino computacional***

Paso 1

Paso 2

- ***Cómputo de las variables requeridas (i.e., métricas Jacobianas, velocidades, flujos).***

Paso 3

Paso 4

- ***Cálculo de las presiones***

Paso 5

- ***Cálculo de las velocidades mediante un esquema "Predictor-Corrector Scheme"***

Paso 6

Paso 7

Paso 8

- ***Obtención de la nueva superficie libre***

Paso 9

Paso 10

Paso 1

Elija alguna forma inicial de la superficie libre. Genere la malla haciendo un parametrización de los límites del dominio computacional al dominio físico y resuelva una ecuación de Laplace en el dominio completo para obtener la malla computacional.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% .               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
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

Paso 2

Implementar un método SOR para obtener las coordenadas curvilíneas a lo largo el dominio, esta es la malla elíptica en el dominio físico. En este paso calcule numéricamente las variables requeridas (es decir, velocidades, flujos).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  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
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

Paso 3

Calcular las métricas y las métricas Jacobianas asociadas con cada celda de la malla y almacenarlas en este paso.

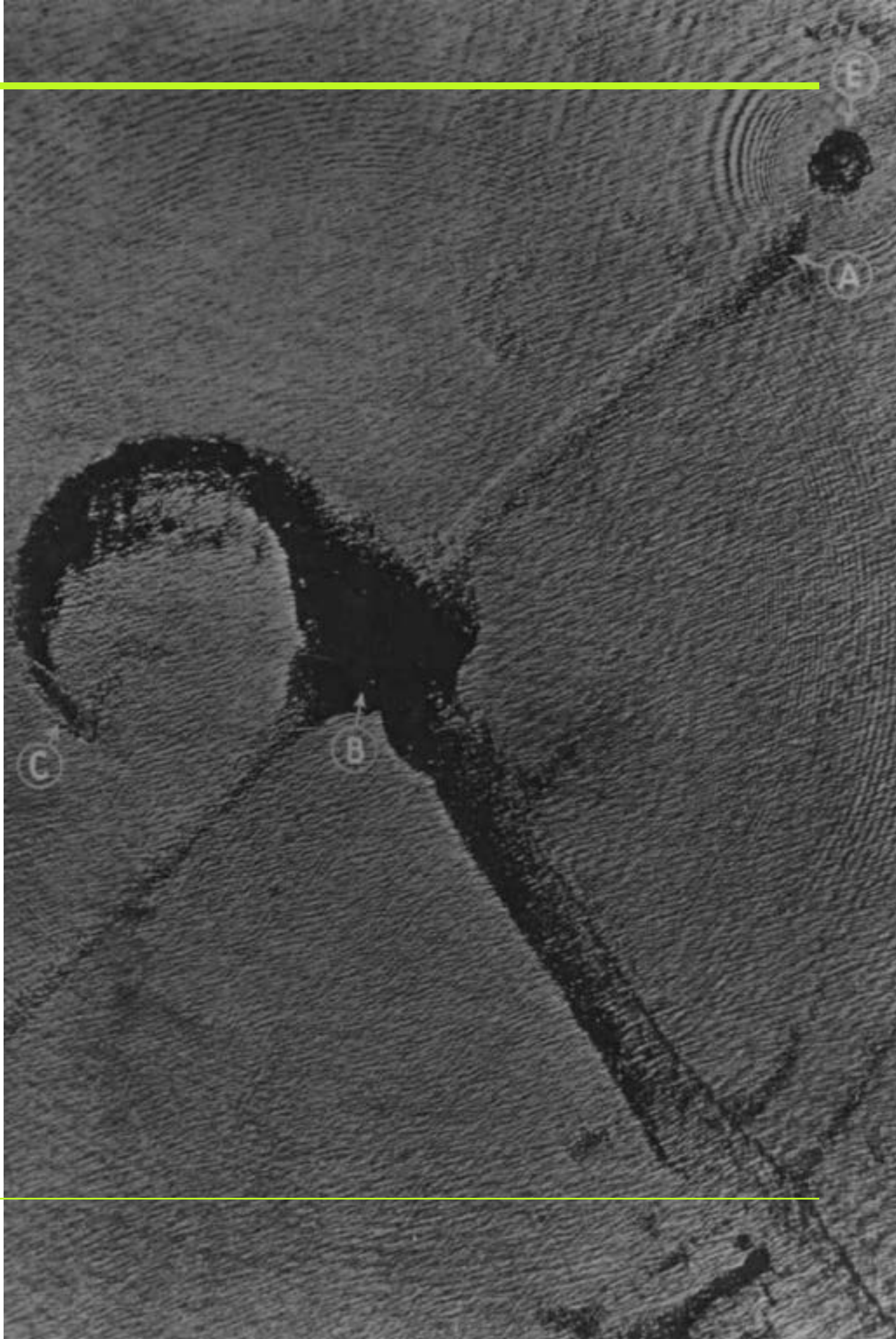
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                The computation of  J^{-1} for each cell                  %
%        This is just the area of the cell We could compute J^{-1}         %
%   using equation od the discriminant and the finite differences given    %
%  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;
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

Paso 4

En este paso se calculan los flujos covariantes en cada celda de la malla y en las caras del malla.



2

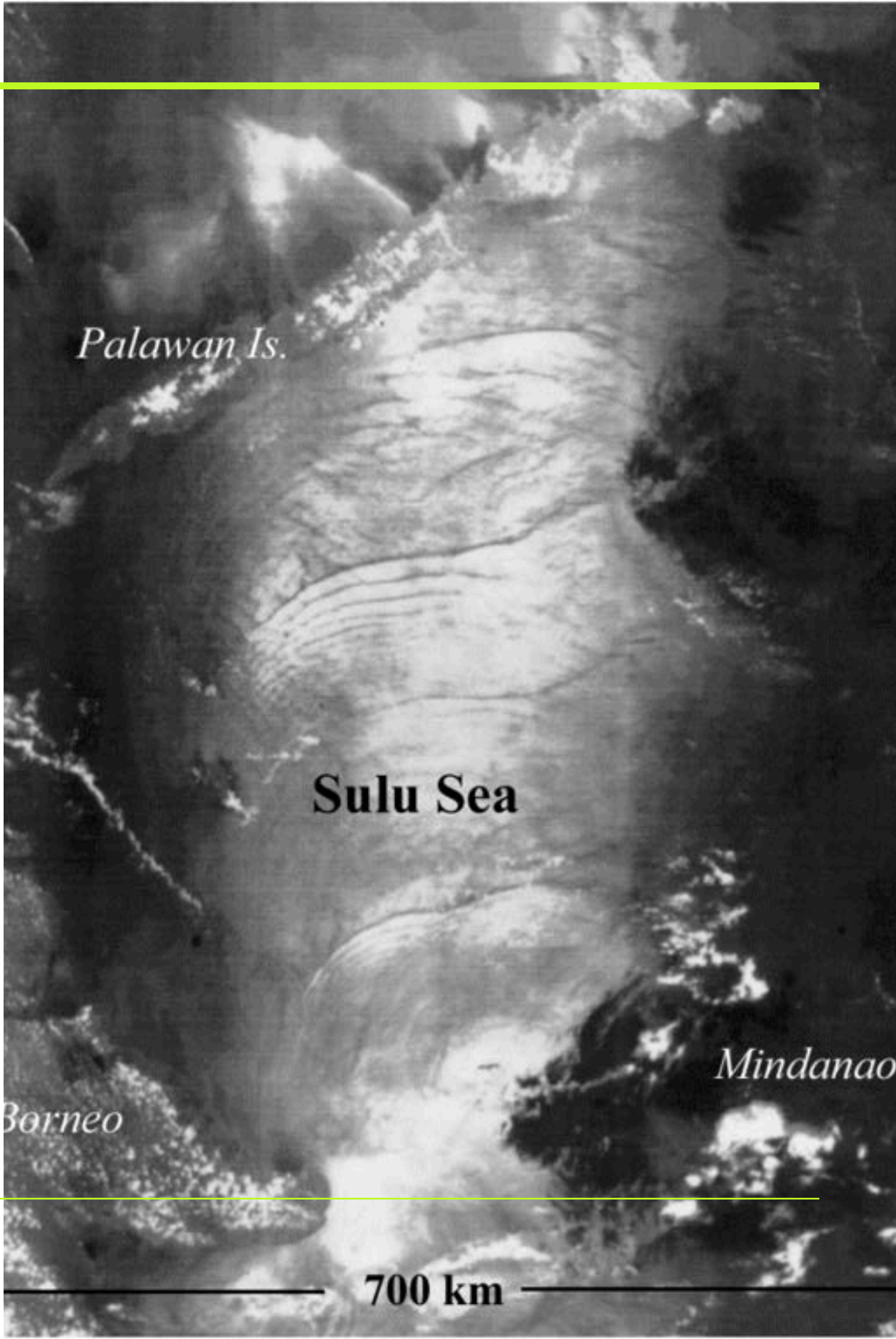
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     FLUX TERMS U1 U2                                      %
%           these FLUXES are defined in the center                          %
% The present method requires, in general, the calculation of fluxes        %
%(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);
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

Paso 5

Resolución de una ecuación de Poisson general para las diferencias de presión mediante un método SOR para sistemas no homogéneos. Y con ello se calculan las presiones en el nuevo paso de tiempo de acuerdo a ecuación iii) en coordenadas curvilíneas generalizadas.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%-------->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 nearest to the parametrization
%I want to interpolate them to the centers %
%%%I will create a new grid in hich the centers I woul define the preassures
[xNv0, index] = unique(xNv0);
eta0Center=interp1(xNv0,eta0Nt(index),xcvC0);
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

```
% SOR method for getting the preassures(iteratively solve)
      maxit2 = 500; err = 1; tol = 1.0e-7;icount = 0;
      while (icount < maxit2 && err > tol)
          icount = icount + 1;
          pold = deltap;
          for kp = 2:1:Nxi2-2
              for jp = 2:1:Nxi1-2
                  G11(kp,jp)=(dxi1dxfr(kp,jp,i).*Jm1dxi1dx29x249r(kp,jp,i)) +...
                      (dxi1dyfr(kp,jp,i).*Jm1dxi1dy29x249r(kp,jp,i));
                  G21(kp,jp)=((dxi1dxfr(kp,jp,i).*Jm1dxi2dx29x249t(kp,jp,i)) +...
                      (dxi2dxft(kp,jp,i).*Jm1dxi1dx29x249r(kp,jp,i))+...
                      (dxi1dyfr(kp,jp,i).*Jm1dxi2dy29x249t(kp,jp,i))+(dxi2dyft(kp,jp,i).*Jm1dxi1dy29x249r(kp,jp,i)));
                  G22(kp,jp)= (dxi2dxft(kp,jp,i).*Jm1dxi2dx29x249t(kp,jp,i)) +...
                      (dxi2dyft(kp,jp,i).*Jm1dxi2dy29x249t(kp,jp,i));
          A(kp,jp) = G11(kp,jp);
          B(kp,jp) = G21(kp,jp);
          C(kp,jp)= G22(kp,jp);
                  deltap(kp,jp)= h*((A(kp,jp)*(0.5*(pold(kp,jp+1)-2*pold(kp,jp)+pold(kp,jp-1)))+...
                      B(kp,jp)*(0.25*(pold(kp+1,jp+1)+pold(kp-1,jp-1)-...
                      pold(kp+1,jp-1)-pold(kp-1,jp+1))) + C(kp,jp)*(0.5*(pold(kp+1,jp)-...
                      2*pold(kp,jp)+pold(kp-1,jp)))+deltaxi1carreJxi1yB1(kp-1,jp-1)+...
                      deltaxi2carreJxi2yB1(kp-1,jp-1)+ deltaxi1xi2Jxi2yB(kp-1,jp-1)+...
                      deltaxi2xi1Jxi1yB(kp-1,jp-1)))-(deltaxi1Jm1U1tilde(kp,jp)+deltaxi2Jm1U2tilde(kp,jp));
              end
          end
          err = 0.0;
          for kp = 2:1:Nxi2-2
          for jp = 2:1:Nxi1-2
          err = err + (deltap(kp,jp) - pold(kp,jp))^2;
          end
          end
          err = sqrt(err/((Nxi2-2)*(Nxi1-2)));
          pold=deltap;
      end
      deltap(:,:,i)=pold;
```

Para dudas sobre el script contáctame en sitio web

https://rosavargas.github.io/

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
deltaphi(:,:,i)=deltap(:,:,i)+gm1*deltayc;
p(:,:,i)=p(:,:,i-1)+deltap(:,:,i);
phip1(:,:,i)=p(:,:,i)+gm1*ycp(:,:,i);
```

```
deltapPORJm1dxi1dx=deltap(:,:,i).* Jm1dxi1dx29x249r(:,:,i); deltaphiPORJm1dxi1dy=deltaphi(:,:,i).* Jm1dxi1dy29x249r(:,:,i);
 deltapdxi1dxv=deltapPORJm1dxi1dx(:,Nxi1-1); deltapdxi1dxB1=horzcat(deltapPORJm1dxi1dx,deltapdxi1dxv);
 deltaphidxi1dyv=deltaphiPORJm1dxi1dy(:,Nxi1-1); deltaphidxi1dyB1=horzcat(deltaphiPORJm1dxi1dy,deltaphidxi1dyv);
deltaxi1deltapdxi1dxP1=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
     for j=1:Nxi2-1
    deltaxi1deltapdxi1dxP1(j,k)=(deltapdxi1dxB1(j,k+1)-deltapdxi1dxB1(j,k))*0.5;
    end
 end
 deltaxi1deltapdxi1dx(:,:,i)=deltaxi1deltapdxi1dxP1;
 deltaxi1deltaphidxi1dyP1=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
     for j=1:Nxi2-1
    deltaxi1deltaphidxi1dyP1(j,k)=(deltaphidxi1dyB1(j,k+1)-deltaphidxi1dyB1(j,k))*0.5;
    end
 end
 deltaxi1deltaphidxi1dy(:,:,i)=deltaxi1deltaphidxi1dyP1; deltapPORJm1dxi2dx=deltap(:,:,i).*Jm1dxi2dx29x249t(:,:,i);
deltaphiPORJm1dxi2dy=deltaphi(:,:,i).*Jm1dxi2dy29x249t(:,:,i); deltapdxi2dxv=deltapPORJm1dxi2dx(1,:);
deltapdxi2dxB1=vertcat(deltapdxi2dxv,deltapPORJm1dxi2dx);
%%%%%
deltaphidxi2dyv=deltaphiPORJm1dxi2dy(1,:); deltaphidxi2dyB1=vertcat(deltaphidxi2dyv,deltaphiPORJm1dxi2dy);
 deltaxi2deltapdxi2dxP1=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
     for j=1:Nxi2-1
    deltaxi2deltapdxi2dxP1(j,k)=(deltapdxi2dxB1(j+1,k)-deltapdxi2dxB1(j,k))*0.5;
    end
 end
  deltaxi2deltapdxi2dx(:,:,i)= deltaxi2deltapdxi2dxP1;
deltaxi2deltaphidxi2dyP1=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
     for j=1:Nxi2-1
    deltaxi2deltaphidxi2dyP1(j,k)=(deltaphidxi2dyB1(j+1,k)-deltaphidxi2dyB1(j,k))*0.5;
    end
 end
 deltaxi2deltaphidxi2dy(:,:,i)=deltaxi2deltaphidxi2dyP1;
```

Paso 6

Resuelva para la velocidad intermedia uU en los centros de la celda. Interpolar uU en las caras de la celda de cada celda y calcular las velocidades intermedias resolviendo el lado derecho de la ecuación ii: y iii: en las ecuaciones III. -

4

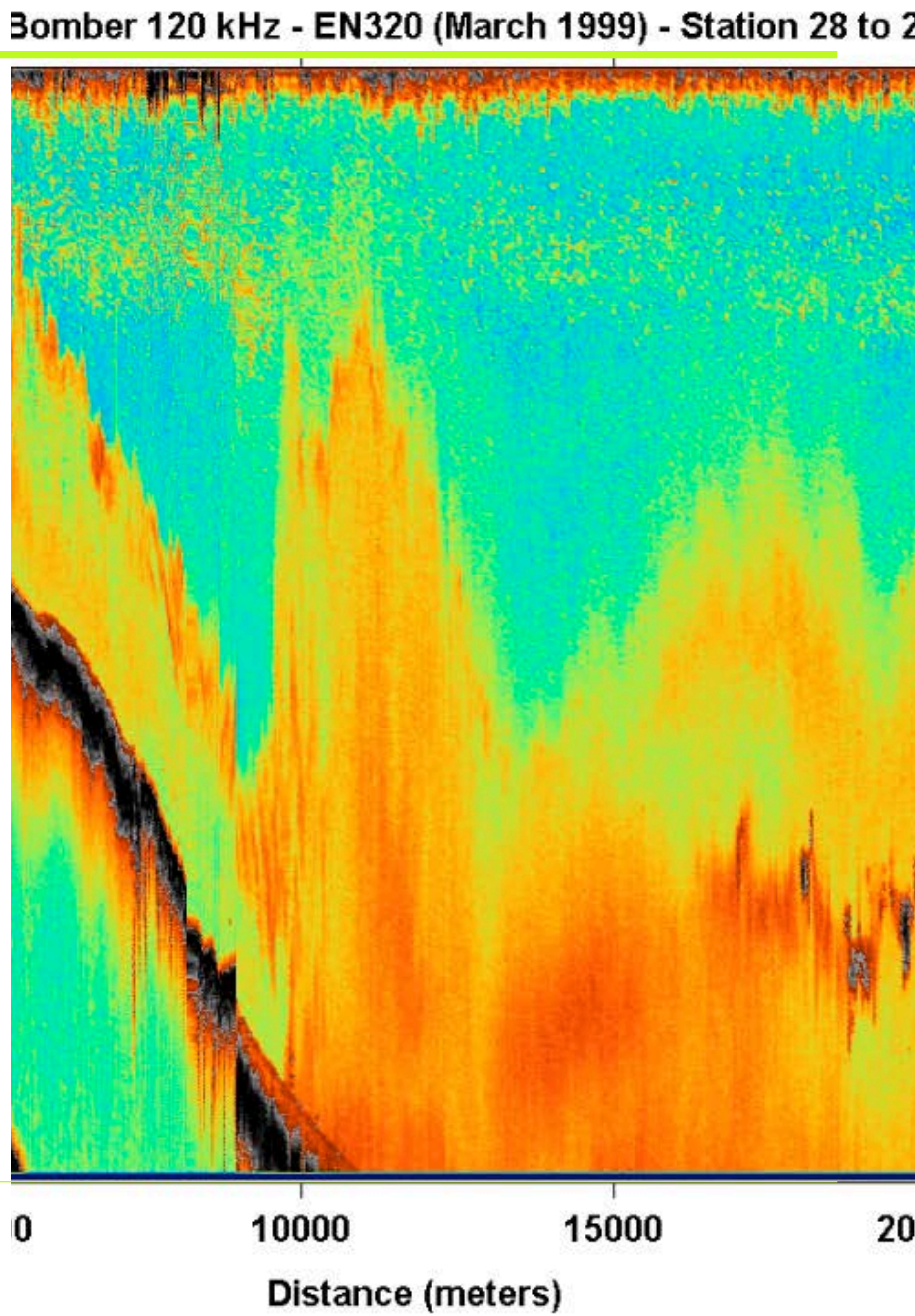
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               QUICK method after getting the left and rigth values of u  %
% in the cell face. If U is constant and the grid spacing dx is constant   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


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);
```

6

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           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;
%%%%%
U1u1=U1.*u1f;
U1u2=U1.*u2f;
%%
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;
%%
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);
```

Paso 7

Se aplica el Corrector para calcular las velocidades en el nuevo paso de tiempo, que es una función de las diferencias de presión calculadas en el **Paso 5** y en las métricas y Jacobiano del **Paso 3.**

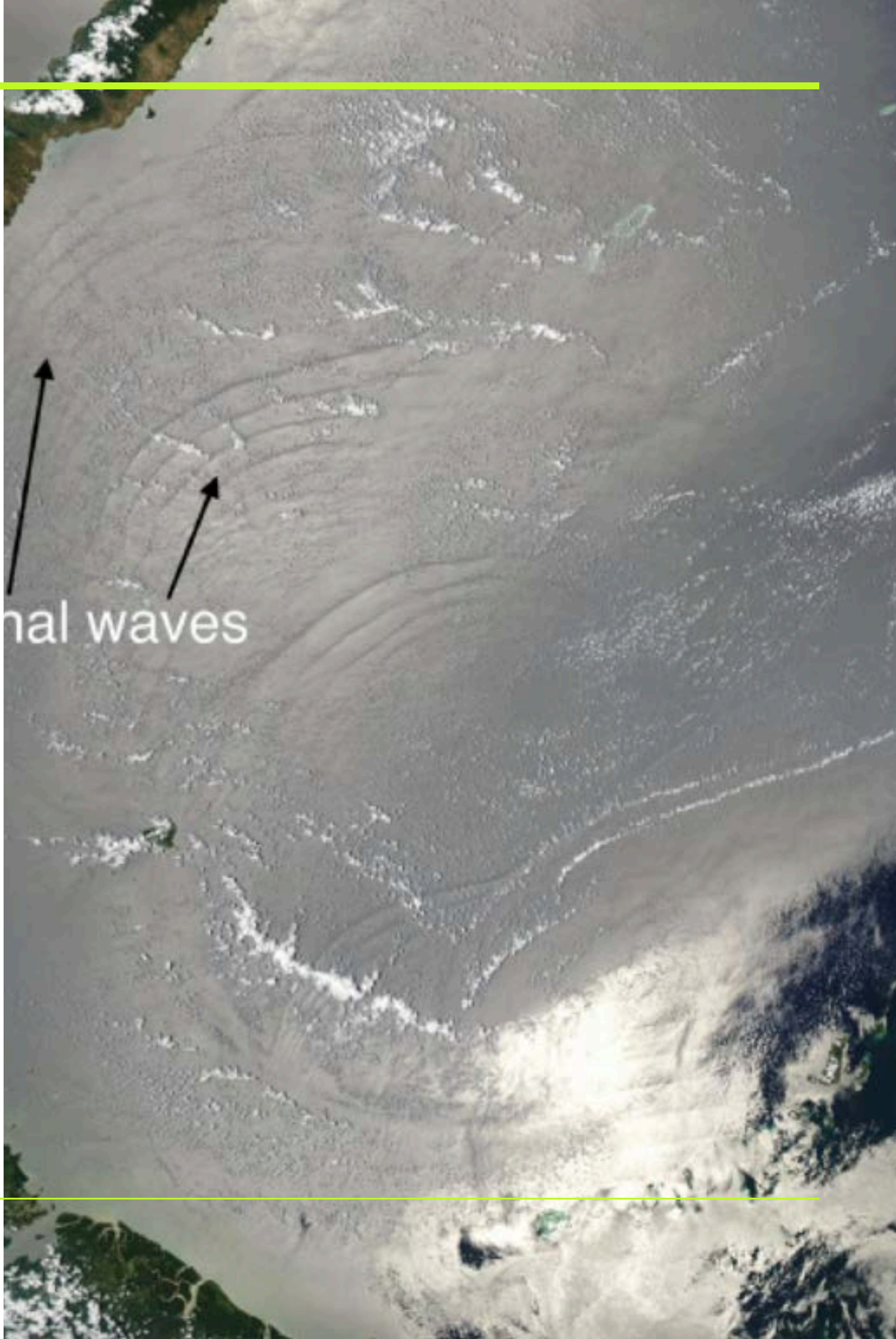


6

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%AND HERE WE ARE AT the Corrector step
 u1PORJm1p(:,:,i)= -h*(deltaxi1deltapdxi1dx(:,:,i)+deltaxi2deltapdxi2dx(:,:,i))+(u1tildePORJm1c(:,:,i));
 u2PORJm1p(:,:,i)= -h*(deltaxi1deltaphidxi1dy(:,:,i)+deltaxi2deltaphidxi2dy(:,:,i))+(u2tildePORJm1c(:,:,i));
 u1(:,:,i)=(Jm1(:,:,i).^(-1)).*u1PORJm1p(:,:,i);
 u2(:,:,i)=(Jm1(:,:,i).^(-1)).*u2PORJm1p(:,:,i);
 %%%The boundary conditions in the bottom
 u2(1,:,i)=ones(Nxi1-1,1)'*0;
  %%%The boundary conditions in left boundary
 u1(:,1,i)=ones(Nxi2-1,1)*0;
 u2(:,1,i)=ones(Nxi2-1,1)*0;
   %%%The boundary conditions in right boundary
 u1(:,Nxi1-1,i)=ones(Nxi2-1,1)*0;
 u2(:,Nxi1-1,i)=ones(Nxi2-1,1)*0;
 %(deltau1/dxi2 in the bottom is zero)
 for j=1:Nxi1-1
     deltau2deltaxi2A(j)=-u2(Nxi2-2,j,i)+u2(Nxi2-1,j,i);
 end
deltau2deltaxi2(:,i)=deltau2deltaxi2A;
  for j=1:Nxi1-1
     deltau1deltaxi2A(j)=-u1(Nxi2-2,j,i)+u1(Nxi2-1,j,i);
  end
 deltau1deltaxi2(:,i)=deltau1deltaxi2A;
  for k=1:Nxi1-1
   for j= 1:Nxi2-1
   %vg(j,k,i)=xc(j,k,i)*0-yc(j,k,i)*(g);
   vg(j,k,i)=1;
   end
  end
```

Paso 8

Se obtienen así las presiones y velocidades para reiniciar el **MAIN LOOP** en el **Paso 1** hasta el tiempo deseado en un cierto paso de tiempo.



Paso 9

Se resuelve la condición de frontera cinemática usando la ecuación v, produciendo el nuevo perfil de superficie libre.

6

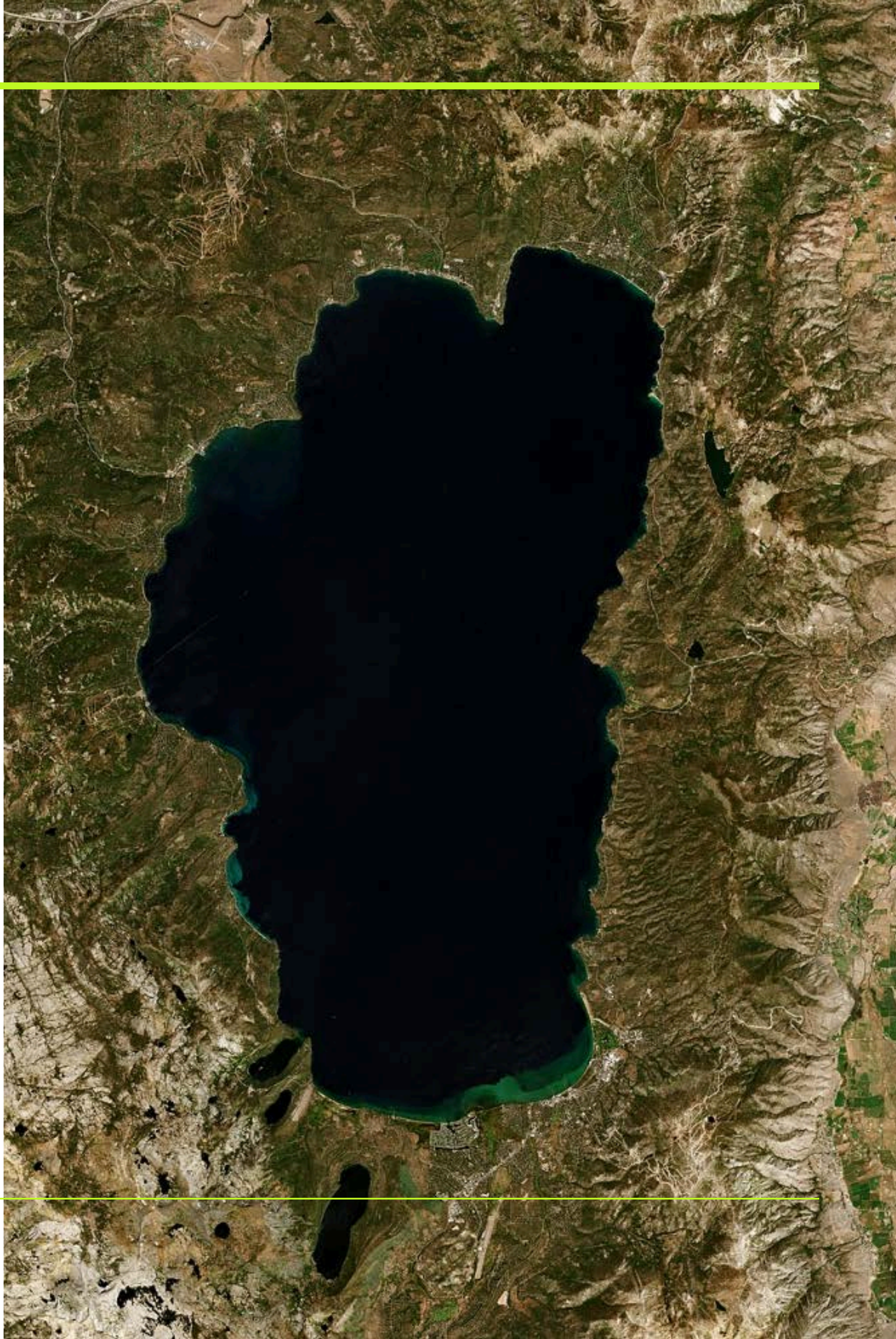
```
%%Towars the definition of the new surface profile ;
 %etan=zeros(Nxi2-1,Nxi1-1);
 for k=1:Nxi1-1
 for j= 1:Nxi2-1
 etanplus1(j,k,i)=g*ycp(j,k,i);
 end
 end
 %etanplus1(:,:,i)=etan;
 etanlastv=etanplus1(:,Nxi2-1,i);
 etanB1=horzcat(etanplus1(:,:,i),etanlastv);

 etanfirstr=etanplus1(1,:,i);
 etanB2=vertcat(etanfirstr,etanplus1(:,:,i));
%deltadxi1etan=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
    for j=1:Nxi2-1
    deltadxi1etan(j,k,i)=etanB1(j,k+1)-etanB1(j,k);
    end
end
%deltadxi1etan(:,:,i)=deltadxi1etan0;
%deltadxi2etan=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
    for j=1:Nxi2-1
    deltadxi2etan(j,k,i)=etanB2(j+1,k)-etanB2(j,k);
    end
end
%         Jm1dxi1dx29x249r(:,:,i)= dydxi2rc(:,:,i);
%         Jm1dxi1dy29x249r(:,:,i)=-dxdxi2rc(:,:,i);
%         Jm1dxi2dx29x249t(:,:,i)=-dydxi1topr(:,:,i);
%         Jm1dxi2dy29x249t(:,:,i)=dxdxi1topr(:,:,i);
etanJm1dxi1dy(:,:,i)=etanplus1(:,:,i).*Jm1dxi1dy29x249r(:,:,i);
etanJm1dxi2dy(:,:,i)=etanplus1(:,:,i).*Jm1dxi2dy29x249t(:,:,i);
```

6

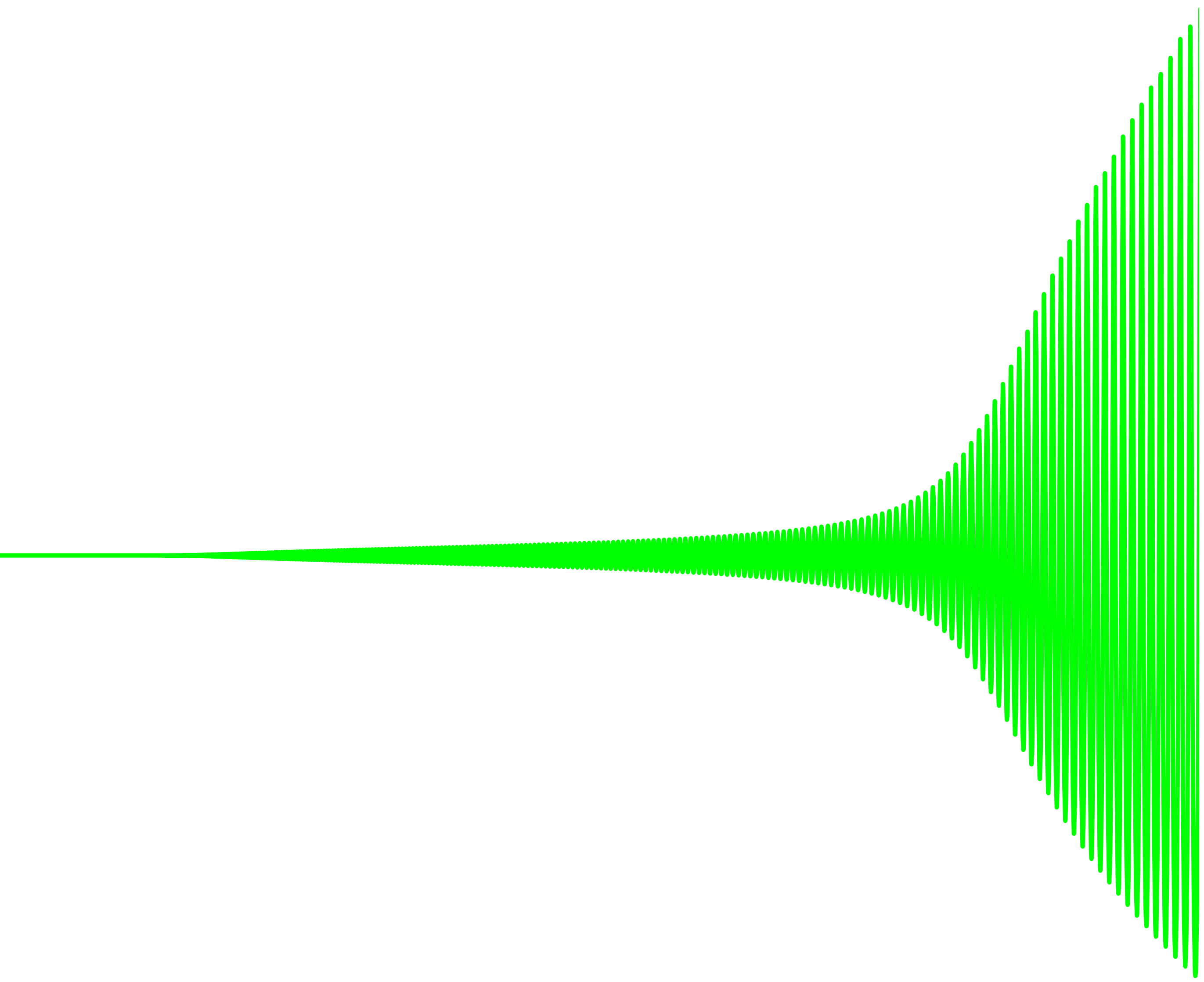
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%more ghost points:

etanJxi1ylast=etanJm1dxi1dy(:,Nxi1-1,i);  %%last vertical
etanJm1dxi1dyB=horzcat(etanJm1dxi1dy(:,:,i),etanJxi1ylast);
% %

etanJxi2yfirst=etanJm1dxi2dy(1,:,i);  %%first horizontal
etanJm1dxi2dyB=vertcat(etanJxi2yfirst,etanJm1dxi2dy(:,:,i));

%%%%Aqui falta un deltaetanplus1dxi1 y deltaetanplus1dxi2!!!!!
% etanJxi1yfirst=etanJm1dxi1dy(1,:);
% vetanJxi1yfirst=Num_r.*etanJxi1yfirst;
% etanJm1dxi1dyB=vertcat(etanJm1dxi1dy,vetanJxi1yfirst);

%deltaxi1etanJm1dxi1dx=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
    for j=1:Nxi2-1
    deltaxi1etanJm1dxi1dy(j,k,i)=etanJm1dxi1dyB(j,k+1) -etanJm1dxi1dyB(j,k);
    end
end

%deltaxi1etanJm1dxi1dx(:,:,i)=deltaxi1etanJm1dxi1dx0;

%deltaxi1etanJm1dxi1dy=zeros(Nxi2-1,Nxi1-1);
for k=1:Nxi1-1
    for j=1:Nxi2-1
    deltaxi2etanJm1dxi2dy(j,k,i)=etanJm1dxi2dyB(j+1,k) -etanJm1dxi2dyB(j,k);
    end
end
%deltaxi1etanJm1dxi1dy(:,:,i)=deltaxi1etanJm1dxi1dy0;
% % % %
```

Paso 10

Actualice la malla en función de la elevación de la superficie calculada en el **paso 9**.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% .             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
```



Avanzar la solución en el tiempo...

Repitiendo el procedimiento (Paso 1-10) en cada paso del tiempo.

Referencias

Solution Methods In Computational Fluid Dynamics
Thomas H Pulliam
Research Scientist CFD Branch
NASA Ames Research Center

◆

Numerical modelling of surface water wave interaction
with a moving wall
Gayaz Khakimzyanov, Denys Dutykh

◆

Generation of solitary waves by transcritical flow over a step
R. H. J. G R IM S H AW, D.-H. Z H A N G AND K. W. C H OW

Numerical study of nonlinear shallow water waves produced by a submerged moving disturbance in viscous flow
Cite as: Physics of Fluids 8, 147 (1996); https://doi.org/10.1063/1.868822
Submitted: 13 March 1995 . Accepted: 25 September 1995 . Published Online: 02 September 1998
Daohua Zhang, and Allen T.

◆

On solitary waves forced by underwater moving objects
By D A O H U A Z H A N G AND A L L E N T. C HW A N G
Department of Mechanical Engineering, The University of Hong Kong,

◆

A Non-staggered Grid, Fractional Step Method for Time-Dependent Incompressible Navier–Stokes Equations in Curvilinear Coordinates
YAN ZANG, ROBERT L. STREET, AND JEFFREY R. KOSEFF
Environmental Fluid Mechanics Laboratory, Stanford University, California 94305-4020