|  |  |  |
| --- | --- | --- |
| gama_logo_color1 | **Cairo University Faculty of Engineering Aerospace Department** | þÿ |

**Computational Fluid Dynamics**

Project 1

Submitted to *Eng. Mina*

|  |  |  |
| --- | --- | --- |
| **Name:** | ***Mohamed Ahmed Hassan Ahmed*** | **BN: 10 Section: 2**  **Date:2022*/* 5 */ 22*** |
|  |  |

Contents

[Table of Figures 2](#_Toc104148531)

[Problem Description 3](#_Toc104148532)

[Givens 3](#_Toc104148533)

[Airfoil 3](#_Toc104148534)

[A. Construct a suitable boundary fitted grid (η1, η2) using (H-grid) or *(****O-grid***) or (C-grid) 4](#_Toc104148535)

[B. Write the governing equation in the proposed body fitted grid (η1, η2) 5](#_Toc104148536)

[C. Choose the numerical method used to solve the governing equation (***PSOR***) or (LSOR) or (ADI) 6](#_Toc104148537)

[D. Determine the values of the stream function ψ at the outer boundaries 7](#_Toc104148538)

[E. Choose a suitable initial value of the stream function ψ for all points in the grid points 7](#_Toc104148539)

[F. Obtain the numerical solution until convergence 8](#_Toc104148540)

[G. Show the results of the convergence history 9](#_Toc104148541)

[H. Show the iso-velocity and iso-pressure lines in the entire domain 10](#_Toc104148542)

[I. Comparing with the potential flow results obtained by the Joukowski transformation between the circle and the airfoil. 11](#_Toc104148543)

[Appendix: Code 12](#_Toc104148544)

[Main.m (script) 12](#_Toc104148545)

[Airfoil.m (Class) 14](#_Toc104148546)

# Table of Figures

[Figure 1. Airfoil 3](#_Toc104148781)

[Figure 2. O-Grid Hole Figure 4](#_Toc104148782)

[Figure 3. Computational Domain 4](#_Toc104148783)

[Figure 4. O-Grid Zoomed-in 5](#_Toc104148784)

[Figure 5. Numerical Solution of non-dimensional velocity using O-Grid 8](#_Toc104148785)

[Figure 6. Numerical Solution of Pressure Coefficient using O-Grid 8](#_Toc104148786)

[Figure 7. Iso-Velocity in Entire Domain 10](#_Toc104148787)

[Figure 8. Iso-Pressure in Entire Domain 10](#_Toc104148788)

[Figure 9. Exact Solution of Velocity 11](#_Toc104148789)

[Figure 10. Exact Solution of Pressure 11](#_Toc104148790)

# Problem Description

The governing equation of an incompressible potential two-dimensional flow past a Joukowski airfoil section can be written as: (Laplace equation) where "ψ" is the stream function.

## Givens

|  |  |  |  |
| --- | --- | --- | --- |
| **BN.** | **Angle of Attach** | **% Camber / chord** | **% Thickness / chord** |
| 10 | Sec(2) alpha = 8 | 5 | 7 |

## Airfoil

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\airfoil.emf  Figure . Airfoil |

# Construct a suitable boundary fitted grid (η1, η2) using (H-grid) or *(****O-grid***) or (C-grid)

Constructing O-Grid was by linear interpolation of the values in the direction of from the surface of the Joukowski Airfoil to the surface of the outer Circle .

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\o-grid-full.emf  Figure . O-Grid Hole Figure |
| Y:\Aerospace\AeroSpace 4\Term 2\CFD (Prof. Mohamed Madbouli)\Project\Results\computitional-grid.emf  Figure . Computational Domain |

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\o-grid-zoomed.emf  Figure . O-Grid Zoomed-in |

# Write the governing equation in the proposed body fitted grid (η1, η2)

# Choose the numerical method used to solve the governing equation (***PSOR***) or (LSOR) or (ADI)

We shall use **Point – SOR** with value of

The general form of the transformed Laplace equation

Where,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Determine the values of the stream function ψ at the outer boundaries

Let at start point "" = 0

Then, Along the outer circle of radius R,

# Choose a suitable initial value of the stream function ψ for all points in the grid points

Thus, we could do linear interpolation from the inner airfoil surface values of to the outer surface values of the boundary conditions, in the direction of .

# Obtain the numerical solution until convergence

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\numVelocity.emf  Figure . Numerical Solution of non-dimensional velocity using O-Grid |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\numCp.emf  Figure . Numerical Solution of Pressure Coefficient using O-Grid |

# Show the results of the convergence history

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\errorHistory.emf |

# Show the iso-velocity and iso-pressure lines in the entire domain

|  |
| --- |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\iso-velocity.emf  Figure . Iso-Velocity in Entire Domain |
| C:\Users\OWNER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\iso-pressure.emf  Figure . Iso-Pressure in Entire Domain |

# Comparing with the potential flow results obtained by the Joukowski transformation between the circle and the airfoil.

|  |
| --- |
| Y:\Aerospace\AeroSpace 4\Term 2\CFD (Prof. Mohamed Madbouli)\Project\Results\exactVelocity.emf  Figure . Exact Solution of Velocity |
| Y:\Aerospace\AeroSpace 4\Term 2\CFD (Prof. Mohamed Madbouli)\Project\Results\exactCp.emf  Figure . Exact Solution of Pressure |

# Appendix: Code

## Main.m (script)

|  |
| --- |
| %% Clear  clear;close all;clc;    %% Initializing Inputs  inputs = struct(...  'chord', 1,...  'Vinf', 100,...  'max\_thickness', 0.07,...  'max\_camber', 0.05,...  'alpha\_deg', 8,...  'i\_max', 61,...  'j\_max', 121,...  'n', 30000,...  'R', 5);    %% Initializing Airfoil Class  airfoil = Airfoil(inputs);    % Calculate Joukowski Airfoil  [~, outerCircle, joukowski] = airfoil.joukowskiAirfoil();    % Plot Joukowski Airfoil  figure;  plot(joukowski.x(1:end/2), joukowski.y(1:end/2),'g-','LineWidth',1); hold on  plot(joukowski.x(end/2: end), joukowski.y(end/2: end), 'b-','LineWidth',1);  xlabel('$X\_{B}$ $for$ $Airfoil$','interpreter','latex','FontSize',14);  ylabel('$Y\_{B}$ $for$ $Airfoil$','interpreter','latex','FontSize',14);  title('$Airfoil$ $shape$ $approximate$','interpreter','latex','FontSize',14);  axis equal  grid on    % Calculate & Plot O-Grid  figure;  hold on;  plot(outerCircle.x, outerCircle.y, '-r', 'LineWidth', 3);  plot(joukowski.x, joukowski.y, '-b', 'LineWidth', 2);  axis equal;  [xGrid, yGrid, ~] = airfoil.generatePhysicalGrid(outerCircle, joukowski);    % Calculate & Plot Computational Grid  figure;  [eta1Grid, eta2Grid] = airfoil.generateComputationalGrid([0, 1], [0, 1]);    % Calculate Metric derivatives x/eta1, y/eta1, x/eta2, y/eta2  % And Calculate Values of C11, C22, C12, J  airfoil.transformationMetrics(xGrid, yGrid, eta1Grid, eta2Grid);    % Calculate & return Boundary Conditions  psi\_ = airfoil.calculateDirichletBoundary();  n = 1;  errorLog10 = ones(1, airfoil.inputs.n);  error = ones(1, airfoil.inputs.n);  while((n <= airfoil.inputs.n) && (min(error) > 1e-8))    psi\_new = airfoil.iterate(psi\_);  psi\_new = psi\_ + 1.1 .\* (psi\_new - psi\_);    error(n) = max(max(abs(psi\_new-psi\_)));  if error(n) > 0; errorLog10(n) = log10(error(n)); end    psi\_ = psi\_new;  psi\_(:,1) = psi\_(1,2);  n = n + 1;  end    %% Plot Error History  figure;  plot(errorLog10,'k')  grid on  xlabel('Iteration number', 'fontsize',14)  ylabel('Log\_1\_0 (Error)', 'fontsize',14)  title('Convergence history using Point-SOR for alpha = 8^o (O-Grid)','fontsize',12)    %% Calculate velocity and Cp    dpsi\_deta1 = airfoil.zerosImaxJmax();  dpsi\_deta2 = airfoil.zerosImaxJmax();  for i=1:airfoil.inputs.i\_max  if i ==1  dpsi\_deta1(i,:) = (psi\_(i+1,:)-psi\_(i,:))./(eta1Grid(i+1,:)-eta1Grid(i,:));  elseif i==airfoil.inputs.i\_max  dpsi\_deta1(i,:) = (psi\_(i,:)-psi\_(i-1,:))./(eta1Grid(i,:)-eta1Grid(i-1,:));  else  dpsi\_deta1(i,:) = (psi\_(i+1,:)-psi\_(i-1,:))./(eta1Grid(i+1,:)-eta1Grid(i,:));  end  end  for j=1:airfoil.inputs.j\_max  if j ==1  dpsi\_deta2(:,j) = (psi\_(:,j+1)-psi\_(:,j))./(eta2Grid(:,j+1)-eta2Grid(:,j));  elseif j==airfoil.inputs.j\_max  dpsi\_deta2(:,j) = (psi\_(:,j)-psi\_(:,j-1))./(eta2Grid(:,j)-eta2Grid(:,j-1));  else  dpsi\_deta2(:,j) = (psi\_(:,j+1)-psi\_(:,j-1))./(eta2Grid(:,j+1)-eta2Grid(:,j-1));  end  end  u = dpsi\_deta1.\*airfoil.deta1\_dy + dpsi\_deta2.\*airfoil.deta2\_dy;  v = -(dpsi\_deta1.\*airfoil.deta1\_dx + dpsi\_deta2.\* airfoil.deta2\_dx);    V = sqrt(u.^2+v.^2);  nonDimV = V/airfoil.inputs.Vinf;  C\_p = 1-(nonDimV).^2;    %% Plotting Results  figure  quiver(xGrid,yGrid,u, v, 'k-', 'LineWidth', 0.5);hold on  plot(joukowski.x, joukowski.y,'r-','LineWidth',2) ; hold on  axis equal  xlim([-(airfoil.inputs.chord+0.5)/2 (airfoil.inputs.chord+0.5)/2])  xlabel('X-axis', 'fontsize',14)  ylabel('Y-axis', 'fontsize',14)  title('Velocity vector for angle of attack =8^o (Numerical Solution - O Grid)','fontsize',12)    figure;  contour(xGrid, yGrid, C\_p, 5000);  title('Contours of Pressure coefficient (angle of attack =8^o) (Numerical Solution - O Grid)','fontsize',12)    figure  plot(xGrid(1:end/2,1),nonDimV(1:end/2,1), 'LineWidth', 1.5);  hold on;  plot(xGrid(end/2:end,1),nonDimV(end/2:end,1), 'LineWidth', 1.5);  xlabel('$X\_{B}$','interpreter','latex','FontSize',14);  ylabel('$\frac{V}{V\_{inf}}$','interpreter','latex','FontSize',14);  title('$Velocity$ $over$ $airfoil$ Using Numerical Solution (O-Grid)','interpreter','latex','FontSize',14);    figure;  plot(xGrid(1:end/2,1),C\_p(1:end/2,1), 'LineWidth', 1.5);  hold on;  plot(xGrid(end/2:end,1),C\_p(end/2:end,1), 'LineWidth', 1.5);  ylim([-4 1.3]);  xlabel('$X\_{B}$','interpreter','latex','FontSize',14);  ylabel('$C\_{p}$','interpreter','latex','FontSize',14);  title('$Pressure$ $over$ $airfoil$ Using Numerical Solution (O-Grid)','interpreter','latex','FontSize',14); |

## Airfoil.m (Class)

|  |
| --- |
| classdef Airfoil < handle    properties  inputs; %Inputs to Constructor function    b, e, beta, a, alpha, x0, y0, cosa, sina; % Calculated form Inputs    delta\_eta1, delta\_eta2;    dx\_deta1, dy\_deta1, dx\_deta2, dy\_deta2; % Metrics of Transformation  deta1\_dx, deta1\_dy, deta2\_dx, deta2\_dy;    Jacobian, C11, C22, C12;    C11\_plusHalf\_i, C11\_negHalf\_i,  C22\_plusHalf\_i, C22\_negHalf\_i,  C12\_plusHalf\_i, C12\_negHalf\_i,    C11\_plusHalf\_j, C11\_negHalf\_j,  C22\_plusHalf\_j, C22\_negHalf\_j,  C12\_plusHalf\_j, C12\_negHalf\_j,    s\_i\_j  s\_in1\_j  s\_ip1\_j  s\_i\_jn1  s\_i\_jp1  s\_in1\_jn1, s\_ip1\_jp1  s\_in1\_jp1, s\_ip1\_jn1  end    methods  function obj = Airfoil(inputs\_struct)  obj.inputs = inputs\_struct;  obj.b = inputs\_struct.chord/4;  obj.e = inputs\_struct.max\_thickness/1.3;  obj.beta = 2\*inputs\_struct.max\_camber;  obj.a = obj.b \*(1+obj.e)/cos(obj.beta);  obj.alpha = inputs\_struct.alpha\_deg \* pi / 180;  obj.x0 = -obj.b \* obj.e;  obj.y0 = obj.a \* sin(obj.beta);  obj.cosa = cos(obj.alpha);  obj.sina = sin(obj.alpha);  end    function [innerCircle, outerCircle, airfoil] = joukowskiAirfoil(this)  theta = linspace(0, 2\*pi, this.inputs.i\_max);    innerCircle = struct(...  'x', this.inputs.chord/2\*cos(theta),...  'y', this.inputs.chord/2\*sin(theta));    outerCircle = struct(...  'x', this.inputs.R\*cos(theta),...  'y', this.inputs.R\*sin(theta));    sign=(sin(theta)./abs(sin(theta)));  sign(1)=1;    joukowski\_y = 2\*this.b\*this.e\*(1-innerCircle.x/2/this.b)...  .\*sign.\*sqrt(1-(innerCircle.x/2/this.b).^2)...  + 2\*this.b\*this.beta\*(1-(innerCircle.x/2/this.b).^2);  airfoil = struct(...  'x', innerCircle.x,...  'y', joukowski\_y);    end    function [xGrid, yGrid, rGrid] = generatePhysicalGrid(this, outerCircle, airfoil)  xGrid = zeros(this.inputs.i\_max, this.inputs.j\_max);  yGrid = zeros(this.inputs.i\_max, this.inputs.j\_max);    for i=1:this.inputs.i\_max  xGrid(i,:)=linspace(airfoil.x(i),outerCircle.x(i),this.inputs.j\_max);  yGrid(i,:)=linspace(airfoil.y(i),outerCircle.y(i),this.inputs.j\_max);  end    rGrid = sqrt(xGrid.^2+yGrid.^2);  colormap([0.95 0.95 0.95]);  g = pcolor(xGrid, yGrid, rGrid);  set(g, 'EdgeColor', [0.7 0.7 0.7]);  end      function [Eta1, Eta2] = generateComputationalGrid(this, eta1\_limit, eta2\_limit)  eta1 = linspace(eta1\_limit(1), eta1\_limit(2), this.inputs.i\_max);  eta2 = linspace(eta2\_limit(1), eta2\_limit(2), this.inputs.j\_max);  [Eta2, Eta1] = meshgrid(eta2, eta1);  plot(Eta1, Eta2, Eta1', Eta2', 'Color', 'b');  axis equal;  this.delta\_eta1 = (eta1\_limit(2)-eta1\_limit(1))/(this.inputs.i\_max-1);  this.delta\_eta2 = (eta2\_limit(2)-eta2\_limit(1))/(this.inputs.j\_max-1);  end    function [] = transformationMetrics(this, xGrid, yGrid, eta1Grid, eta2Grid)  this.dx\_deta1 = this.zerosImaxJmax();  this.dy\_deta1 = this.zerosImaxJmax();  for i = 1:this.inputs.i\_max  if(i == 1)  this.dx\_deta1(i,:) =(xGrid(i+1,:)-xGrid(i,:))./(eta1Grid(i+1,:)-eta1Grid(i,:));  this.dy\_deta1(i,:) =(yGrid(i+1,:)-yGrid(i,:))./(eta1Grid(i+1,:)-eta1Grid(i,:));  elseif (i == this.inputs.i\_max)  this.dx\_deta1(i,:) =(xGrid(i,:)-xGrid(i-1,:))./(eta1Grid(i,:)-eta1Grid(i-1,:));  this.dy\_deta1(i,:) =(yGrid(i,:)-yGrid(i-1,:))./(eta1Grid(i,:)-eta1Grid(i-1,:));  else  this.dx\_deta1(i,:) =(xGrid(i+1,:)-xGrid(i-1,:))./(eta1Grid(i+1,:)-eta1Grid(i-1,:));  this.dy\_deta1(i,:) =(yGrid(i+1,:)-yGrid(i-1,:))./(eta1Grid(i+1,:)-eta1Grid(i-1,:));  end  end    this.dx\_deta2 = this.zerosImaxJmax();  this.dy\_deta2 = this.zerosImaxJmax();  for j = 1:this.inputs.j\_max  if(j == 1)  this.dx\_deta2(:,j) =(xGrid(:,j+1)-xGrid(:,j))./(eta2Grid(:,j+1)-eta2Grid(:,j));  this.dy\_deta2(:,j) =(yGrid(:,j+1)-yGrid(:,j))./(eta2Grid(:,j+1)-eta2Grid(:,j));  elseif (j == this.inputs.j\_max )  this.dx\_deta2(:,j) =(xGrid(:,j)-xGrid(:,j-1))./(eta2Grid(:,j)-eta2Grid(:,j-1));  this.dy\_deta2(:,j) =(yGrid(:,j)-yGrid(:,j-1))./(eta2Grid(:,j)-eta2Grid(:,j-1));  else  this.dx\_deta2(:,j) =(xGrid(:,j+1)-xGrid(:,j-1))./(eta2Grid(:,j+1)-eta2Grid(:,j-1));  this.dy\_deta2(:,j) =(yGrid(:,j+1)-yGrid(:,j-1))./(eta2Grid(:,j+1)-eta2Grid(:,j-1));  end  end    this.Jacobian = this.dx\_deta1.\*this.dy\_deta2-this.dx\_deta2.\*this.dy\_deta1;  this.deta1\_dx = this.dy\_deta2./this.Jacobian;  this.deta1\_dy = -this.dx\_deta2./this.Jacobian;  this.deta2\_dx = -this.dy\_deta1./this.Jacobian;  this.deta2\_dy = this.dx\_deta1./this.Jacobian;      this.C11 = (this.dx\_deta2.^2+this.dy\_deta2.^2)./this.Jacobian;  this.C22 = (this.dx\_deta1.^2+this.dy\_deta1.^2)./this.Jacobian;  this.C12 = -(this.dx\_deta1.\*this.dx\_deta2+this.dy\_deta1.\*this.dy\_deta2)./this.Jacobian;    this.C11\_plusHalf\_i = this.zerosImaxJmax();  this.C11\_negHalf\_i = this.zerosImaxJmax();  this.C22\_plusHalf\_i = this.zerosImaxJmax();  this.C22\_negHalf\_i = this.zerosImaxJmax();  this.C12\_plusHalf\_i = this.zerosImaxJmax();  this.C12\_negHalf\_i = this.zerosImaxJmax();    for i=1:this.inputs.i\_max  if(i ~= this.inputs.i\_max)  this.C11\_plusHalf\_i(i, :) = (this.C11(i, :)+this.C11(i+1, :))/2;  this.C22\_plusHalf\_i(i, :) = (this.C22(i, :)+this.C22(i+1, :))/2;  this.C12\_plusHalf\_i(i, :) = (this.C12(i, :)+this.C12(i+1, :))/2;  end  if(i ~= 1)  this.C11\_negHalf\_i(i, :) = (this.C11(i, :)+this.C11(i-1, :))/2;  this.C22\_negHalf\_i(i, :) = (this.C22(i, :)+this.C22(i-1, :))/2;  this.C12\_negHalf\_i(i, :) = (this.C12(i, :)+this.C12(i-1, :))/2;  end  end  this.C11\_plusHalf\_i(this.inputs.i\_max, :) = this.C11\_plusHalf\_i(1, :);  this.C22\_plusHalf\_i(this.inputs.i\_max, :) = this.C22\_plusHalf\_i(1, :);  this.C12\_plusHalf\_i(this.inputs.i\_max, :) = this.C12\_plusHalf\_i(1, :);    this.C11\_negHalf\_i(1, :) = this.C11\_negHalf\_i(this.inputs.i\_max, :);  this.C22\_negHalf\_i(1, :) = this.C22\_negHalf\_i(this.inputs.i\_max, :);  this.C12\_negHalf\_i(1, :) = this.C12\_negHalf\_i(this.inputs.i\_max, :);    this.C11\_plusHalf\_j = this.zerosImaxJmax();  this.C11\_negHalf\_j = this.zerosImaxJmax();  this.C22\_plusHalf\_j = this.zerosImaxJmax();  this.C22\_negHalf\_j = this.zerosImaxJmax();  this.C12\_plusHalf\_j = this.zerosImaxJmax();  this.C12\_negHalf\_j = this.zerosImaxJmax();    for j=2:this.inputs.j\_max-1  this.C11\_plusHalf\_j(:,j) = (this.C11(:,j)+this.C11(:,j+1))/2;  this.C11\_negHalf\_j(:,j) = (this.C11(:,j)+this.C11(:,j-1))/2;  this.C22\_plusHalf\_j(:,j) = (this.C22(:,j)+this.C22(:,j+1))/2;  this.C22\_negHalf\_j(:,j) = (this.C22(:,j)+this.C22(:,j-1))/2;  this.C12\_plusHalf\_j(:,j) = (this.C12(:,j)+this.C12(:,j+1))/2;  this.C12\_negHalf\_j(:,j) = (this.C12(:,j)+this.C12(:,j-1))/2;  end    deta1\_deta2 = this.delta\_eta1/this.delta\_eta2;    this.s\_i\_j = this.C11\_plusHalf\_i + this.C11\_negHalf\_i + (this.C22\_plusHalf\_j + this.C22\_negHalf\_j)\*(deta1\_deta2)^2;    this.s\_in1\_j = this.C11\_negHalf\_i - (this.C12\_plusHalf\_j - this.C12\_negHalf\_j)\*(deta1\_deta2/2)^2;  this.s\_ip1\_j = this.C11\_plusHalf\_i + (this.C12\_plusHalf\_j - this.C12\_negHalf\_j)\*(deta1\_deta2/2)^2;    this.s\_i\_jn1 = (this.C22\_negHalf\_j)\*(deta1\_deta2)^2 - (this.C12\_plusHalf\_i - this.C12\_negHalf\_i)\*(deta1\_deta2/4);  this.s\_i\_jp1 = (this.C22\_plusHalf\_j)\*(deta1\_deta2)^2 + (this.C12\_plusHalf\_i - this.C12\_negHalf\_i)\*(deta1\_deta2/4);    this.s\_in1\_jn1 = (this.C12\_negHalf\_i + this.C12\_negHalf\_j)\*(deta1\_deta2/4);  this.s\_ip1\_jp1 = (this.C12\_plusHalf\_i + this.C12\_plusHalf\_j)\*(deta1\_deta2/4);    this.s\_in1\_jp1 = -(this.C12\_negHalf\_i + this.C12\_plusHalf\_j)\*(deta1\_deta2/4);  this.s\_ip1\_jn1 = -(this.C12\_plusHalf\_i + this.C12\_negHalf\_j)\*(deta1\_deta2/4);      end    function [psi] = calculateDirichletBoundary(this)  theta = linspace(0, 2\*pi, this.inputs.i\_max);  u = this.inputs.Vinf \* this.cosa;  v = this.inputs.Vinf \* this.sina;    x = this.inputs.R \* cos(theta);  y = this.inputs.R \* sin(theta);    psiBoundary = zeros(this.inputs.i\_max, 1);    for i=2:this.inputs.i\_max  psiBoundary(i,1) = psiBoundary(i-1, 1) - v\*(x(i) - x(i-1)) + u\*(y(i) - y(i-1));  end    psi = this.zerosImaxJmax();  psi(:, end) = psiBoundary;    for i=1:this.inputs.i\_max  psi(i, 1:end) = linspace(psi(i,1),psi(i,end),this.inputs.j\_max);  end  end    function [psi] = iterate(this, psi\_pre)    psi = this.zerosImaxJmax();  psi(:, 1) = psi\_pre(:, 1);  psi(:,end) = psi\_pre(:, end);    for i=1:this.inputs.i\_max  for j=2:this.inputs.j\_max-1  if i==1  psi(i,j)=(this.s\_in1\_j(i,j) \* psi\_pre(this.inputs.i\_max-1,j) + this.s\_ip1\_j(i,j) \* psi\_pre(i+1,j)...  + this.s\_i\_jp1(i,j) \* psi\_pre(i,j+1) +this.s\_i\_jn1(i,j)\*psi(i,j-1)...  + this.s\_in1\_jn1(i,j)\*psi\_pre(this.inputs.i\_max-1,j-1)+this.s\_in1\_jp1(i,j)\*psi\_pre(this.inputs.i\_max-1,j+1)+this.s\_ip1\_jn1(i,j)\*psi\_pre(i+1,j-1)+this.s\_ip1\_jp1(i,j)\*psi\_pre(i+1,j+1))...  /this.s\_i\_j(i,j);  elseif i== this.inputs.i\_max  psi(i,j)=(this.s\_in1\_j(i,j) \* psi(i-1,j) + this.s\_ip1\_j(i,j) \* psi\_pre(1+1,j)...  + this.s\_i\_jp1(i,j) \* psi\_pre(i,j+1) +this.s\_i\_jn1(i,j)\*psi(i,j-1)...  + this.s\_in1\_jn1(i,j)\*psi(i-1,j-1)+this.s\_in1\_jp1(i,j)\*psi(i-1,j+1)+this.s\_ip1\_jn1(i,j)\*psi(1+1,j-1)+this.s\_ip1\_jp1(i,j)\*psi\_pre(1+1,j+1))...  /this.s\_i\_j(i,j);  else  psi(i,j)=(this.s\_in1\_j(i,j) \* psi(i-1,j) + this.s\_ip1\_j(i,j) \* psi\_pre(i+1,j)...  + this.s\_i\_jp1(i,j) \* psi\_pre(i,j+1) +this.s\_i\_jn1(i,j)\*psi(i,j-1)...  + this.s\_in1\_jn1(i,j)\*psi(i-1,j-1)+this.s\_in1\_jp1(i,j)\*psi(i-1,j+1)+this.s\_ip1\_jn1(i,j)\*psi\_pre(i+1,j-1)+this.s\_ip1\_jp1(i,j)\*psi\_pre(i+1,j+1))...  /this.s\_i\_j(i,j);  end  end  end  end    function [m] = zerosImaxJmax(this)  m = zeros(this.inputs.i\_max, this.inputs.j\_max);  end    end  end |