Benjamin Bemis Numerical Method's HW 8 AME 60614

10. Consider the two-dimensional Burgers equation, which is a non-linear model of the convection–diffusion process

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$
$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right).$$

We are interested in the steady state solution in the unit square, $0 \le x \le 1, 0 \le y \le 1$ with the following boundary conditions

$$u(0,y)=u(1,y)=v(x,1)=0, \quad v(x,0)=1$$
$$u(x,0)=u(x,1)=\sin 2\pi x, \quad v(0,y)=v(1,y)=1-y.$$

The solutions of the Burgers equation usually develop steep gradients like those encountered in shock waves. Let $\nu = 0.015$.

(a) Solve this problem using an explicit method. Integrate the equations until steady state is achieved (to plotting accuracy). Plot the steady state velocities u, v. (If you have access to a surface plotter such as in MATLAB, use it. If not, plot the velocities along the two lines: $x = 0.5$ and $y = 0.5$.) Make sure that you can stand behind the accuracy of your solution. Note that since we seek only the steady state solution, the choice of the initial condition should be irrelevant.

(b) Formulate the problem using a second-order ADI scheme for the diffusion terms and an explicit scheme for the convection terms. Give the details including the matrices involved.

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$
$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

BC.

$u(0,y)=u(1,y)=v(x,1)=0 \qquad v(x,0)=1$

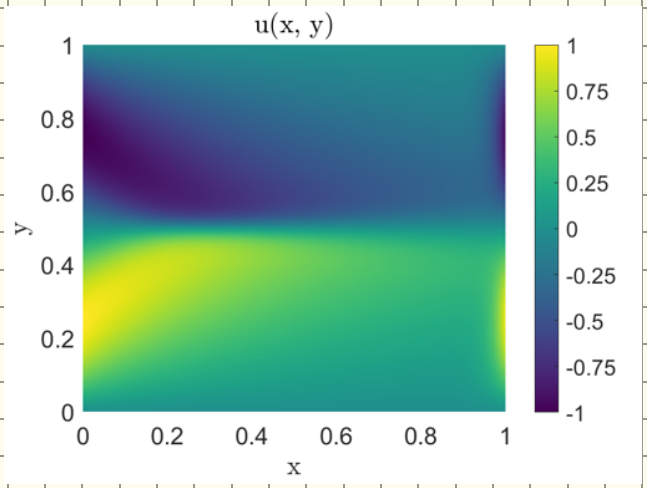
$u(x,0)=u(x,1)=\sin \pi x \qquad v(0,y)=v(1,y)=1-y$

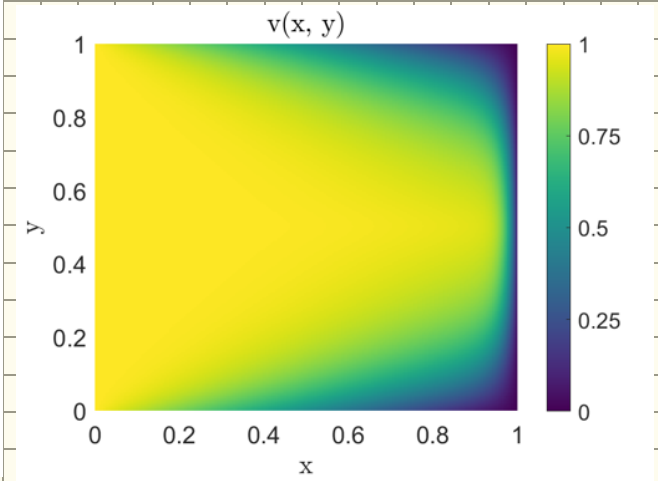
a) $\Delta x=\Delta y \quad \Delta t$ Explicit time step of 2nd order central differencing.

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}+u_{i,j}^{n}\left(\frac{u_{i+1,j}^{n}-u_{i-1,j}^{n}}{2\Delta x}\right)+v_{i,j}^{n}\left(\frac{u_{i,j+1}^{n}-u_{i-1,j}^{n}}{2\Delta y}\right)=\nu\left(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^2}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta y^2}\right)$$

$$u_{i,j}^{n+1}=u_{i,j}^{n}-\Delta t\left\{u_{i,j}^{n}\left(\frac{u_{i+1,j}^{n}-u_{i-1,j}^{n}}{2\Delta x}\right)+v_{i,j}^{n}\left(\frac{u_{i,j+1}^{n}-u_{i-1,j}^{n}}{2\Delta y}\right)\right\}+\Delta t\nu\left(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^2}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta y^2}\right)$$

Similarly for V instead of U.





B). $U \rightarrow \frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$ $V \rightarrow \frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$

Using Explicit Euler for convection and Crank-Nicholson for Diffusion then 2nd order central for x,y

$$\frac{u^{n+1}-u^{n}}{\Delta t}=-u^{n}\frac{\partial u^{n}}{\partial x}+v^{n}\frac{\partial u^{n}}{\partial y}+\frac{\nu}{2}\left(\frac{\partial^2 u^{n+1}}{\partial x^2}+\frac{\partial^2 u^{n}}{\partial x^2}+\frac{\partial^2 u^{n+1}}{\partial y^2}+\frac{\partial^2 u^{n}}{\partial y^2}\right)+O(\Delta t^2,\Delta x^2,\Delta y^2)$$

$$u^{n+1}-\frac{\nu\Delta t}{2}\left(\frac{\partial^2 u^{n+1}}{\partial x^2}+\frac{\partial^2 u^{n+1}}{\partial y^2}\right)=u_n-\Delta t\left[u^{n}\frac{\partial u^{n}}{\partial x}+v^{n}\frac{\partial u^{n}}{\partial y}\right]+\frac{\nu\Delta t}{2}\left(\frac{\partial^2 u^{n}}{\partial x^2}+\frac{\partial^2 u^{n}}{\partial y^2}\right)+\Delta t\,O(\Delta t^2,\Delta x^2,\Delta y^2)$$

$$\left(I-\frac{\nu\Delta t}{2}A_x+\frac{\nu\Delta t}{2}A_y\right)u^{n+1}=\left(I+\frac{\nu\Delta t}{2}A_x+\frac{\nu\Delta t}{2}A_y\right)u^{n}-\Delta t\left(u^{n}B_x+v^{n}B_y\right)u^{n}+TE$$

$$\left(I-\frac{\nu\Delta t}{2}A_x\right)\left(I-\frac{\nu\Delta t}{2}A_y\right)u^{n+1}=\left(I+\frac{\nu\Delta t}{2}A_x+\frac{\nu\Delta t}{2}A_y\right)u^{n}-\Delta t\left(u^{n}B_x+v^{n}B_y\right)u^{n}-\frac{\nu^2\Delta t^2}{4}A_xA_y\left(u^{n+1}-u^{n}\right)+TE\,O(\Delta t^3)$$

combined into $TE\,O(\Delta t^3)$

looking @ LHS

$\psi^{n+1} = \left(I - \frac{\nu\Delta t}{2} A_y\right) u^{n+1}$ LHS becomes $\left(I - \frac{\nu\Delta t}{2} A_x\right) \psi^{n+1} = RHS$

$$\psi_{i,j}^{n+1} - \frac{\nu\Delta t}{2\Delta x^2}\left(\psi_{i+1,j}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i-1,j}^{n+1}\right) = RHS_{i,j}$$

$$\psi_{i,j}^{n+1} = u_{i,j}^{n+1} - \frac{\nu\Delta t}{2\Delta y^2}\left(u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}\right)$$

Solve for $i = 1, 2, 3, \cdots N-1$
$j = 1, 2, 3, M-1$

Process follows for $v_{i,j}^{n+1}$ in the same manner.

Matrix setup: $\alpha = \frac{\nu\Delta t}{\Delta x^2}$

$$A = \begin{bmatrix} 1+2\alpha & -\alpha & & & \\ -\alpha & 1+2\alpha & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -\alpha \\ & & & -\alpha & 1+2\alpha \end{bmatrix}^{n+1} \qquad RHS$$

$$(1+2\alpha)\psi_{i,j}^{n+1} - \alpha\psi_{i+1,j}^{n+1} - \alpha\psi_{i-1,j}^{n+1} = RHS \qquad \begin{matrix} i = 1,2,3,\cdots N-1 \\ j = 1,2,3,\cdots M-1 \end{matrix}$$

Applying the Periodic BC

$$\begin{bmatrix} 1 & 0 & \cdots & & 0 \\ -\alpha & 1+2\alpha & -\alpha & & \vdots \\ & \ddots & \ddots & \ddots & \\ & & -\alpha & 1+2\alpha & 0 \\ 0 & \cdots & & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} RHS_1 - (1+2\alpha)\psi_{0,j} \\ RHS_2 \\ \vdots \\ RHS_{M-1} \\ RHS_M - (1+2\alpha)\psi_{M,j} \end{bmatrix}$$

$u_{i,j}^{n+1}$ where $\gamma = \frac{\nu\Delta t}{2\Delta y^2}$

$$\begin{bmatrix} 1+2\gamma & -\gamma & & & \\ -\gamma & 1+2\gamma & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -\gamma \\ & & & -\gamma & 1+2\gamma \end{bmatrix} \qquad RHS = \psi_{i,j}$$

$$(1+2\gamma)u_{i,j}^{n+1} - \gamma u_{i,j+1}^{n+1} - \gamma u_{i,j-1}^{n+1} = RHS \qquad BC \quad \sin(2\pi x) = u_{i,0} = u_{i,N}$$

$$\begin{bmatrix} 1 & 0 & \cdots & & 0 \\ -\gamma & 1+2\gamma & -\gamma & & \vdots \\ & \ddots & \ddots & \ddots & \\ & & -\gamma & 1+2\gamma & 0 \\ 0 & \cdots & & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \psi_{1,1} - (1+2\gamma)u_{i,0} \\ 2 \\ \vdots \\ \psi_N - (1+2\gamma)u_{i,N} \end{bmatrix}^{m-1}$$

The same will apply to V.

Contents


```
%{

@author: Benjamin Bemis Ph.D Student,
Advisor: Dr Juliano


Description:
AME 60614: Numerical Methods
Homework: 8
Due: 12/12/2024


%}
```

Preperation of the workspace

```
clear all
clc
close all
fontsize = 16;


% set(0,'DefaultFigureWindowStyle','default')
set(0,'DefaultTextInterpreter','latex')
set(0,'DefaultAxesFontSize',fontsize)
set(0,'DefaultLegendFontSize',fontsize)
colors  = ["#000000","#1b9e77","#d95f02","#7570b3","#0099FF"]';
```

Setting data paths

Make sure to update this for the machine that you are working on. (Maybe, This should now run on any machine without change. 7/24/24) Change the current folder to the folder of this m-file.

```
if(~isdeployed)
  cd(fileparts(matlab.desktop.editor.getActiveFilename));
end

addpath(cd)
% cd ..; % Moving up a directory (from processing_code)
basepath = cd; % Pulling the current directory


imagepath = [basepath filesep 'images' filesep];
mkdir(imagepath);
```

```
Warning: Directory already exists.
```

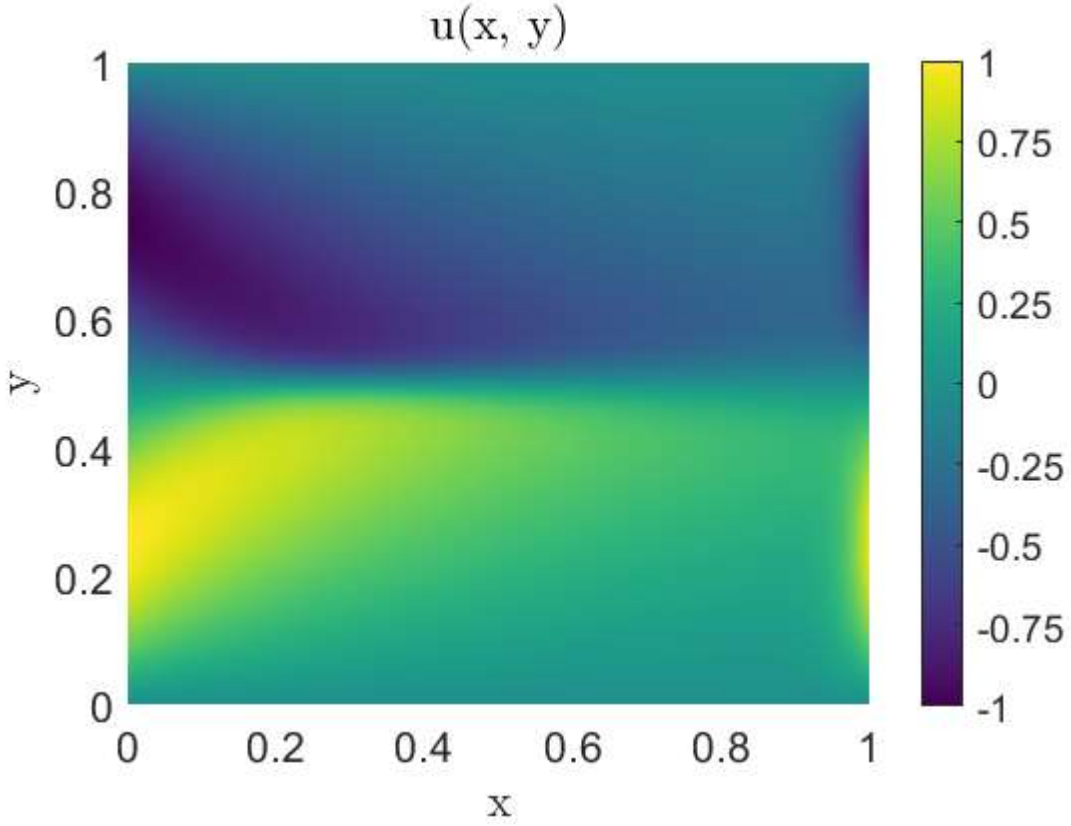
Chapter 10 Problem 5

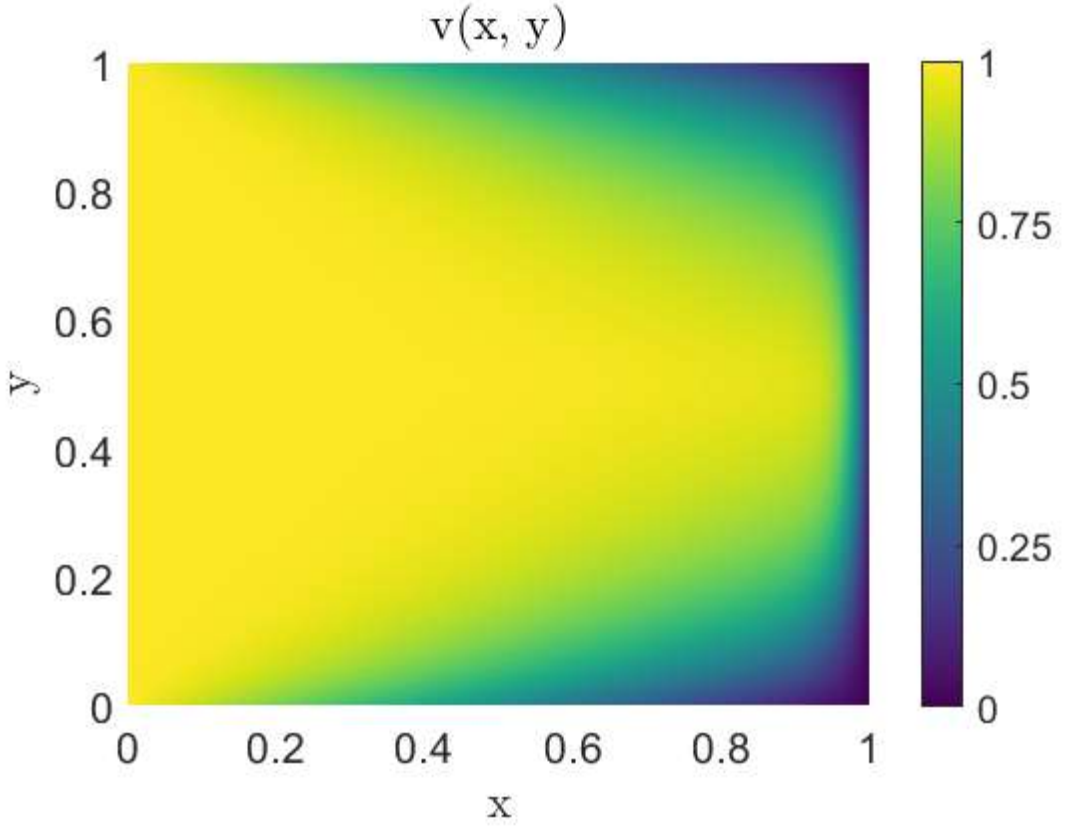
```
% Parameters
nu = 0.015;
N = 200; % Number of grid points
L = 1; % Domain length
h = L / (N - 1); % Grid spacing
dt = 0.0001; % Time step
tolerance = 1e-6; % Steady state tolerance
maxIter = 1e5; % Maximum iterations

x = linspace(0, L, N);
y = linspace(0, L, N);
[X, Y] = meshgrid(x, y);

u = zeros(N, N);
v = zeros(N, N);

% Boundary conditions
u(:, 1) = sin(2 * pi * x); % u(x, 0)
u(:, end) = sin(2 * pi * x); % u(x, 1)
v(1, :) = 1 - y; % v(0, y)
v(end, :) = 1 - y; % v(1, y)

% Time-stepping
for iter = 1:maxIter
    u_old = u;
    v_old = v;

    % u
    for i = 2:N-1
        for j = 2:N-1
            u(i, j) = u_old(i, j) - dt * ( ...
                u_old(i, j) * (u_old(i+1, j) - u_old(i-1, j)) / (2 * h) + ...
                v_old(i, j) * (u_old(i, j+1) - u_old(i, j-1)) / (2 * h) ) ...
                + nu * dt * ( ...
                (u_old(i+1, j) - 2*u_old(i, j) + u_old(i-1, j)) / h^2 + ...
                (u_old(i, j+1) - 2*u_old(i, j) + u_old(i, j-1)) / h^2);
        end
    end

    % v
    for i = 2:N-1
        for j = 2:N-1
            v(i, j) = v_old(i, j) - dt * ( ...
                u_old(i, j) * (v_old(i+1, j) - v_old(i-1, j)) / (2 * h) + ...
                v_old(i, j) * (v_old(i, j+1) - v_old(i, j-1)) / (2 * h) ) ...
                + nu * dt * ( ...
                (v_old(i+1, j) - 2*v_old(i, j) + v_old(i-1, j)) / h^2 + ...
                (v_old(i, j+1) - 2*v_old(i, j) + v_old(i, j-1)) / h^2);
        end
    end

    % Boundary conditions
    u(:, 1) = sin(2 * pi * x); % u(x, 0)
    u(:, end) = sin(2 * pi * x); % u(x, 1)
    v(1, :) = 1 - y; % v(0, y)
    v(end, :) = 1 - y; % v(1, y)
    v(:, 1) = 1; % v(x, 0)
    v(:, end) = 0; % v(x, 1)
```

```
    % Convergence
    if max(max(abs(u - u_old))) < tolerance && max(max(abs(v - v_old))) < tolerance
        disp(['Converged in ', num2str(iter), ' iterations.']);
        break;
    end
end


figure;
f = pcolor(X, Y, u)
set(f, 'edgecolor','none')
title('u(x, y)')
xlabel('x'); ylabel('y'); zlabel('u');
colormap viridis
a = colorbar
set(a,'YTick',-1:.25:1)
set(gca, 'CLim', [-1 1])
print(gcf,[imagepath,'u.png'],'-dpng');


figure
f = pcolor(X, Y, v)
set(f, 'edgecolor','none')
title('v(x, y)')
xlabel('x'); ylabel('y'); zlabel('v');
colormap viridis
a = colorbar
set(a,'YTick',0:.25:1)
set(gca, 'CLim', [0 1])
print(gcf,[imagepath,'v.png'],'-dpng');
```

```
Converged in 26021 iterations.

f =

  Surface with properties:

      EdgeColor: [0 0 0]
      LineStyle: '-'
      FaceColor: 'flat'
    FaceLighting: 'flat'
      FaceAlpha: 1
          XData: [200×200 double]
          YData: [200×200 double]
          ZData: [200×200 double]
          CData: [200×200 double]

  Use GET to show all properties


a =

  ColorBar with properties:

    Location: 'eastoutside'
      Limits: [-1.0000 1.0000]
    FontSize: 14.4000
    Position: [0.8315 0.1500 0.0381 0.7690]
```

```
      Units: 'normalized'

  Use GET to show all properties


f =

  Surface with properties:

      EdgeColor: [0 0 0]
      LineStyle: '-'
      FaceColor: 'flat'
    FaceLighting: 'flat'
      FaceAlpha: 1
          XData: [200×200 double]
          YData: [200×200 double]
          ZData: [200×200 double]
          CData: [200×200 double]

  Use GET to show all properties


a =

  ColorBar with properties:

    Location: 'eastoutside'
      Limits: [0 1]
    FontSize: 14.4000
    Position: [0.8315 0.1500 0.0381 0.7690]
      Units: 'normalized'

  Use GET to show all properties
```





Published with MATLAB® R2021a