Contents

Code for two dimensional natural convection unsteady state simulation

```
clear all
close all
```

Inputs

```
n = 61;
x = linspace(0, 1, n);
y = linspace(0, 1, n);
dt = 0.001;
endTime = 1;
writeInterval = 100; % Timesteps
h = x(2) - x(1);
nu = 0.1;
alpha = 0.1;
g = 9.81;
T0 = 20;
T1 = 30;

beta = 1 / T0;
[xx, yy] = meshgrid(x, y);
vort = zeros(n, n);
psi = zeros(n, n);
temp = T0 * ones(n, n);
temp(:, 1) = T1;
t = 0;
iter = 0;
run = 1;
% Add these lines before the main loop

u_history = [];

temp_history = [];
```

Loop Initiation

```
while t < endTime
```

```
    vortold = vort;
    psiold = psi;
    tempold = temp;
    iter = iter + 1;
    t = t + dt;
```

Update vorticity

```
    for i = 2:n-1
        for j = 2:n-1
            u = (psi(i, j + 1) - psi(i, j - 1)) / (2 * h);
            v = (psi(i + 1, j) - psi(i - 1, j)) / (2 * h);
            vort(i, j) = vort(i, j) + dt * (-(psi(i, j + 1) - psi(i, j - 1)) * (vort(i + 1, j) - vort(i - 1, j)) / (4 * h ^ 2) ...
                          + (psi(i + 1, j) - psi(i - 1, j)) * (vort(i, j + 1) - vort(i, j - 1)) / (4 * h ^ 2) ...
                          + nu * (vort(i + 1, j) + vort(i, j + 1) + vort(i - 1, j) + vort(i, j - 1) - 4 * vort(i, j)) / h ^ 2 ...
```

```
                            - beta * g * (temp(i, j+1) - temp(i, j-1)) / (2 * h));
        end
    end
```

Solve for streamfunction

```
    for i = 2:n-1
        for j = 2:n-1
            psi(i, j) = 0.25 * (h ^ 2 * vort(i, j) + psi(i + 1, j) + psi(i - 1, j) + psi(i, j + 1) + psi(i, j - 1));
        end
    end
```

Update temperature

```
    for i = 2:n-1
        for j = 2:n-1
            u = (psi(i, j + 1) - psi(i, j - 1)) / (2 * h);
            v = (psi(i + 1, j) - psi(i - 1, j)) / (2 * h);
            temp(i, j) = temp(i, j) + dt * (alpha * (temp(i + 1, j) + temp(i - 1, j) + temp(i, j + 1) + temp(i, j - 1) - 4 * temp(i, j)) / h ^ 2 ...
                          - u * (temp(i, j + 1) - temp(i, j - 1)) / (2 * h) ...
                          - v * (temp(i + 1, j) - temp(i - 1, j)) / (2 * h));
end

    end
```

Impose temperature boundary conditions

```
    temp(:, 1) = 10; % Left
    temp(:, end) = 10; % Right
    temp(1, :) = 100; % Bottom
    temp(end, :) = 10; % Top
```

Update velocities

```
    for i = 1:n
        for j = 1:n
            if j ~= 1 && j ~= n && i ~= 1 && i ~= n
                u(i, j) = (psi(i, j + 1) - psi(i, j - 1)) / (2 * h);
                v(i, j) = (psi(i + 1, j) - psi(i - 1, j)) / (2 * h);
            else
                u(i, j) = 0;
                v(i, j) = 0;
            end
        end
    end
    u_history = [u_history, u(:)];
    temp_history = [temp_history, temp(:)];
    if mod(iter, writeInterval) == 0
        fprintf('Calculating ... time = %f \n', t)
    end
```

```
Calculating ... time = 0.100000


Calculating ... time = 0.200000


Calculating ... time = 0.300000


Calculating ... time = 0.400000


Calculating ... time = 0.500000


Calculating ... time = 0.600000


Calculating ... time = 0.700000
Calculating ... time = 0.800000
```

```
Calculating ... time = 0.900000
```

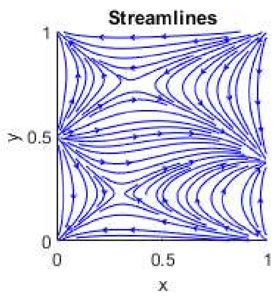
```
Calculating ... time = 1.000000
```

```
end

figure('name', 'Results')
```

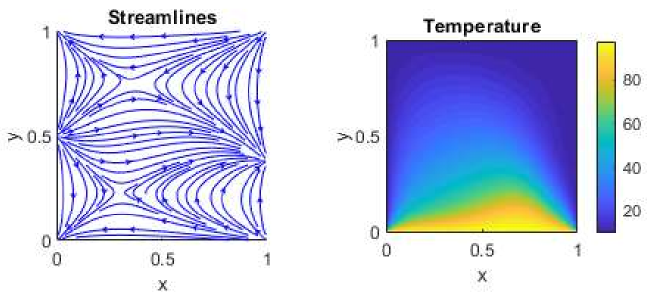
Streamline plot for velocity

```
subplot(2, 2, 1)
streamslice(y, x, u', v')
axis equal
axis([0 1 0 1])
xlabel('x')
ylabel('y')
title('Streamlines')
```



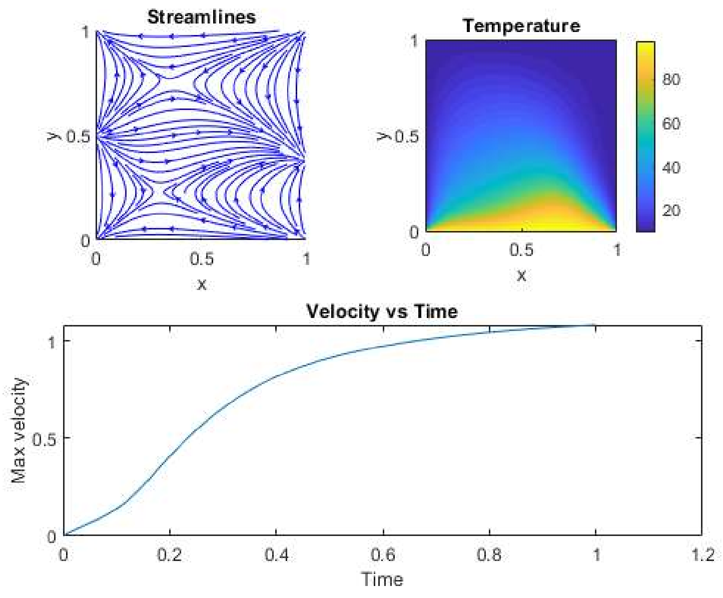
Temperature contour plot

```
subplot(2, 2, 2)
contourf(x, y, temp, 40, 'LineColor', 'none')
axis equal
axis([0 1 0 1])
xlabel('x')
ylabel('y')
title('Temperature')
colorbar
```



XY plot for velocity

```
subplot(2, 2, [3,4])
plot(linspace(0, t, size(u_history, 2)), max(u_history, [], 1))
xlabel('Time')
ylabel('Max velocity')
title('Velocity vs Time')
```


Streamlines
Temperature
Velocity vs Time
Max velocity
Time
y
x