# Homework No.7

Osamu Katagiri-Tanaka : A01212611

October 7, 2020

**Write a code to generate pairs (x,y) generating a geometry (triangle), to use as figure to see the evolution of a tracer within a 2-D velocity field.**

```
% Program to visualize the velocity field and to track a tracer figure
% By JLLopez CFD 09/29/2020
% Solution adapted from (jose lopez salinas)'s solution
clear;
close all;

% The tracer figure is a circle
rho      = 0.3;           % Radius of the circle
x0       = 0.5;
y0       = 0.55;          % Center of the circle
p1       = [x0 y0 rho];   % parameter to draw the geometry
[x,y]    = ftriangle(p1); % function to generate the shape
[n1,m1]  = size(x);
m        = max(n1,m1);
to       = 0;
tf       = 1.00;

% vector position
for i = 1 : m
    z(2 * i - 1) = x(i);
    z(2 * i)     = y(i);
end

% parameters you may need in the vector field function
p     = 1;
zo    = z;                      % initial condition
tspan = linspace(0, 0.5, 20); % time span to track the fluid parcels
tf    = [0 1];

% Solution of the ODEs dr/dt , here you solve the velocity field eqn (vector
    field)
[time, YS] = ode45(@Vfield, tspan, zo, [], p);

% get the time position at 1/3 of the time
index_1third = round(size(YS,1)/3);
YL1 = YS(index_1third,:);

% get the time position at 2/3 of the time
index_2thirds = 2*index_1third;
YL2 = YS(index_2thirds,:);
```

```
40
41 % get the last time position
42 YLF = YS(end,:);
43
44 % generate plot-able vectors
45 for i = 1 : m
46     x1(i) = YL1(2 * i - 1);
47     y1(i) = YL1(2 * i);
48     x2(i) = YL2(2 * i - 1);
49     y2(i) = YL2(2 * i);
50     xf(i) = YLF(2 * i - 1);
51     yf(i) = YLF(2 * i);
52 end
53
54 % plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y
55 figure;
56 hold all;
57 [xx, yy, Ux, Uy] = MPlotxx1();
58
59 % plots initial position and final to compare
60 plot(x,   y, 'ro-', 'DisplayName', sprintf('t = 0'));
61 plot(x1, y1, 'go-', 'DisplayName', sprintf('t = %i', index_1third));
62 plot(x2, y2, 'bo-', 'DisplayName', sprintf('t = %i', index_2thirds));
63 plot(xf, yf, 'mo-', 'DisplayName', sprintf('t = %i', size(YS, 1)));
64 xlabel('x');
65 ylabel('y');
66 legend;
67 % xlim([0.2 1.4]);
68 % ylim([0.2 1.4]);
69
70 % Export Graphics
71 fig = gcf;
72 fig.PaperUnits = 'inches';
73 fig.PaperPosition = [0 0 9 6];
74 print('velocityField', '-dpng', '-r0')
```

Listing 1: Velocity Field Visualization

Listing 1, is as the provided *vfieldplotHD.m* file. However lines 33 through 52 were amended to visualize the evolution of the triangle points within the velocity field at three different times. Figure 1 is the output of Listing 1, where the red points are the positions os the original triangle at $t = 0$, followed by the green, blue and magenta points which represent the evolution of the red points in space in times $t = 7$, $t = 14$ and $t = 20$ respectively. Function *ftriangle()* was written to replace *fcircle()* and *fsquarex()* in order to generate points that resemble a triangle, as depicted in listing 2. Unlike *fcircle()* and *fsquarex()* that compute the geometry points in function of the angle $\theta$, *ftriangle()* first computes the vertexes of the triangle and then joins those vertexes with lines.

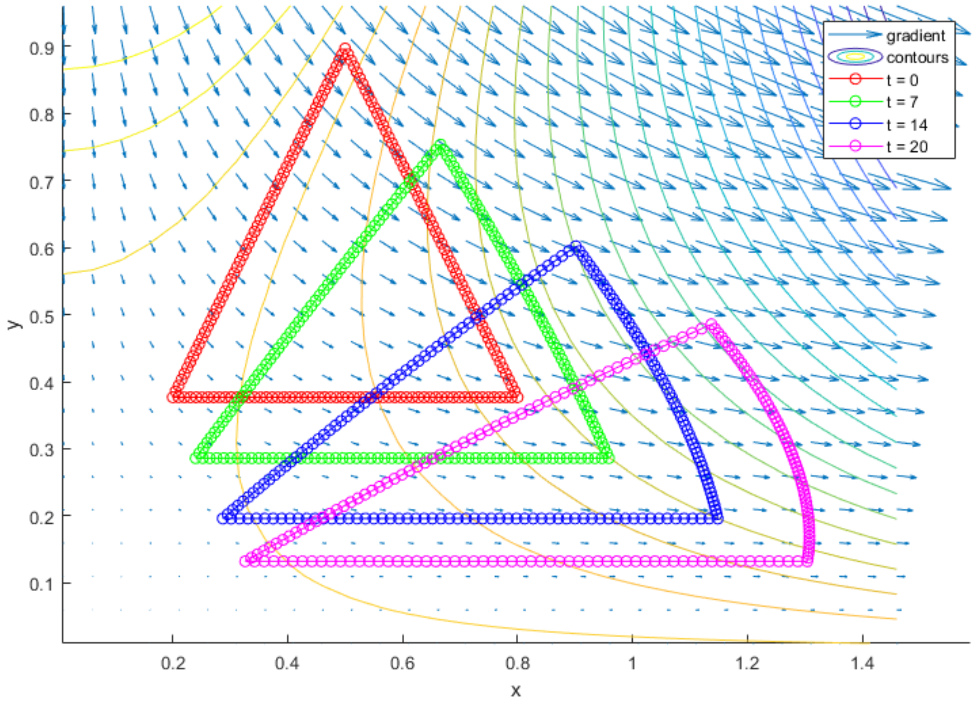


Figure 1: Visualization of a Velocity Field

```
% https://math.stackexchange.com/questions/1344690/is-it-possible-to-find-the-vertices-of-an-equilateral-triangle-given-its-center
function[x, y] = ftriangle(p3)
% p3 are the parameters of the triangle
% x0, y0 is the location of the center and a the apotema
% send parameters xo, yo and a
    x0 = p3(1);
    y0 = p3(2);
    a  = p3(3)*2;
    n  = round(a * 100); % points per side

    top_vertex_x = x0;
    top_vertex_y = y0 + sqrt(3) / 3 * a;

    right_vertex_x = x0 + a / 2;
    right_vertex_y = y0 - sqrt(3) / 6 * a;

    left_vertex_x = x0 - a / 2;
    left_vertex_y = y0 - sqrt(3) / 6 * a;

    % line from top_vertex to right_vertex
    [xx, yy] = lineFunction( ...
        top_vertex_x, top_vertex_y, ...
        right_vertex_x, right_vertex_y, n);
    for i = 1 : n
        % fprintf('%i \n', i)
        x(i) = xx(i);
        y(i) = yy(i);
    end
```

```

    % line from right_vertex to left_vertex
    count = 1;
    [xx, yy] = lineFunction( ...
        right_vertex_x, right_vertex_y, ...
        left_vertex_x, left_vertex_y, n);
    for i = n + 1 : n * 2
        % fprintf('%i \n', i)
        x(i) = xx(count);
        y(i) = yy(count);
        count = count + 1;
    end

    % line from left_vertex to top_vertex
    count = 1;
    [xx, yy] = lineFunction( ...
        left_vertex_x, left_vertex_y, ...
        top_vertex_x, top_vertex_y, n);
    for i = n * 2 + 1 : n * 3
        % fprintf('%i \n', i)
        x(i) = xx(count);
        y(i) = yy(count);
        count = count + 1;
    end

    % plot(x, y, 'ro-')
end
```

Listing 2: Function to draw a triangle