swe_ws2021_2_flow_visualization

November 10, 2020

1 Tutorial 2. Flow Visualization

1.0.1 Description: Flow visualization is important when assessing a flow field and is helpful in better understanding the flow characteristics. Streamlines, pathlines and streaklines are introduced in this exercise. The first part deals with defining, setting up and visualizing a vector field. Here it represents a steady flow, more precisely a pure rotational field. In-built streamline plot has also underlying numerical approximation due to which the streamline is instead of a circle (analytical solution) a spiral. Isocontours are used to show the velocity magnitude. Numerical integration methods are implemented to show the differences for pathline calculation. This follows the Lagrangian view, i.e. particle tracking. While the flow domain is defined by a vector field independent of time, i.e. steady, the particles in the flow have a changing velocity as this depends by definition on the position. Some exercises are proposed.

Students are advised to complete the exercises. Project: Structural Wind Engineering WS20-21
Chair of Structural Analysis @ TUM - R. Wüchner, M. Péntek, A. Kodakkal

Author: anoop.kodakkal@tum.de mate.pentek@tum.de

Created on: 08.11.2015
Last update: 09.11. 2020

Contents:

1. Calculation of divergence and curl
2. Vector field visualization
3. Streamlines
4. Pathlines with different numerical integration schemes

[1]:
```
# import
import time
import matplotlib.pyplot as plt
import numpy as np
import sympy
from matplotlib import pyplot as plt
```

Symbolic vector calculus using the symbolic toolbox (sympy) to define and display the velocity field u.

[2]:
```
# declare symbolic variables
x = sympy.Symbol('x')
y = sympy.Symbol('y')
z = sympy.Symbol('z')
t = sympy.Symbol('t')
```

Let us define each component of the velocity field.

[3]:
```
##
## modify here for a different velocity field
## use here the format: symbolic_vel_component = f(t,x,y) = a + b*x + c*y + d*t
##
symbolic_velocity_x = 0 + 0 * x + (-2) * y + 0 * t
symbolic_velocity_y = 0 + 2 * x + 0 * y + 0 * t
symbolic_velocity_z = 0 + 0 * x + 0 * y + 0 * t
print('u = [', symbolic_velocity_x,' ', symbolic_velocity_y,' ',␣
 ↪symbolic_velocity_z,']')

# creating function from the symbolic expression
# which are later used for numerical evaluations
velocity_x = sympy.lambdify((t,x,y), symbolic_velocity_x)
velocity_y = sympy.lambdify((t,x,y), symbolic_velocity_y)
velocity_z = sympy.lambdify((t,x,y), symbolic_velocity_z)
```

```
u = [ -2*y   2*x   0 ]
```

For exercise 1

Try a different velocity field. Modify in block 3 and not block 4!

[4]:
```
# symbolic_velocity_x = 2
# symbolic_velocity_y = 2
# symbolic_velocity_z = 0
# print('u = [', symbolic_velocity_x,' ', symbolic_velocity_y,' ',␣
 ↪symbolic_velocity_z,']')
```

1.0.2 Calculation of divergence and curl

Two key operators in vector calculus are divergence and curl. Recall from lecture the definitions of these two quantities. What do these two quantities signify? Divergence signifies how a vector field changes its magnitude in the neighborhood of a point and curl is an indicator of how its direction changes.

Divergence of a velocity field $u(x,y,z)$ is

$$div(u) = \nabla \cdot u = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$$

[5]:
```
symbolic_div = sympy.diff(symbolic_velocity_x, x) + sympy.
 ↪diff(symbolic_velocity_y, y) + sympy.diff(symbolic_velocity_z, z);
print('div(u)= ', symbolic_div)
```

```
div(u)=  0
```

Curl of a velocity field $u(x, y, z)$ is

$$curl(u) = \nabla \times u = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u_x & u_y & u_z \end{vmatrix}$$

[6]:
```
symbolic_curl_x = sympy.diff(symbolic_velocity_z, y) - sympy.
 ↪diff(symbolic_velocity_y, z)
symbolic_curl_y = sympy.diff(symbolic_velocity_x, z) - sympy.
 ↪diff(symbolic_velocity_z, x)
symbolic_curl_z = sympy.diff(symbolic_velocity_y, x) - sympy.
 ↪diff(symbolic_velocity_x, y)
print('curl(u) = [', symbolic_curl_x,' ', symbolic_curl_y,' ',␣
 ↪symbolic_curl_z,']')
```

```
curl(u) = [ 0   0   4 ]
```

1.0.3 Vector (velocity) field visualization

We need to generate a grid to visualize.

[7]:
```
# give grid size parameters
grid_spacing_start = -2
grid_spacing_end = 2
grid_spacing_size = 0.4

# generate grid size vectors 1D
grid_spacing_x = np.arange(grid_spacing_start, grid_spacing_end +␣
 ↪grid_spacing_size, grid_spacing_size)
grid_spacing_y = np.arange(grid_spacing_start, grid_spacing_end +␣
 ↪grid_spacing_size, grid_spacing_size)

meshgrid_x, meshgrid_y = np.meshgrid(grid_spacing_x, grid_spacing_y)
```

Refer to meshgrid for more details on mesh grid function on numpy

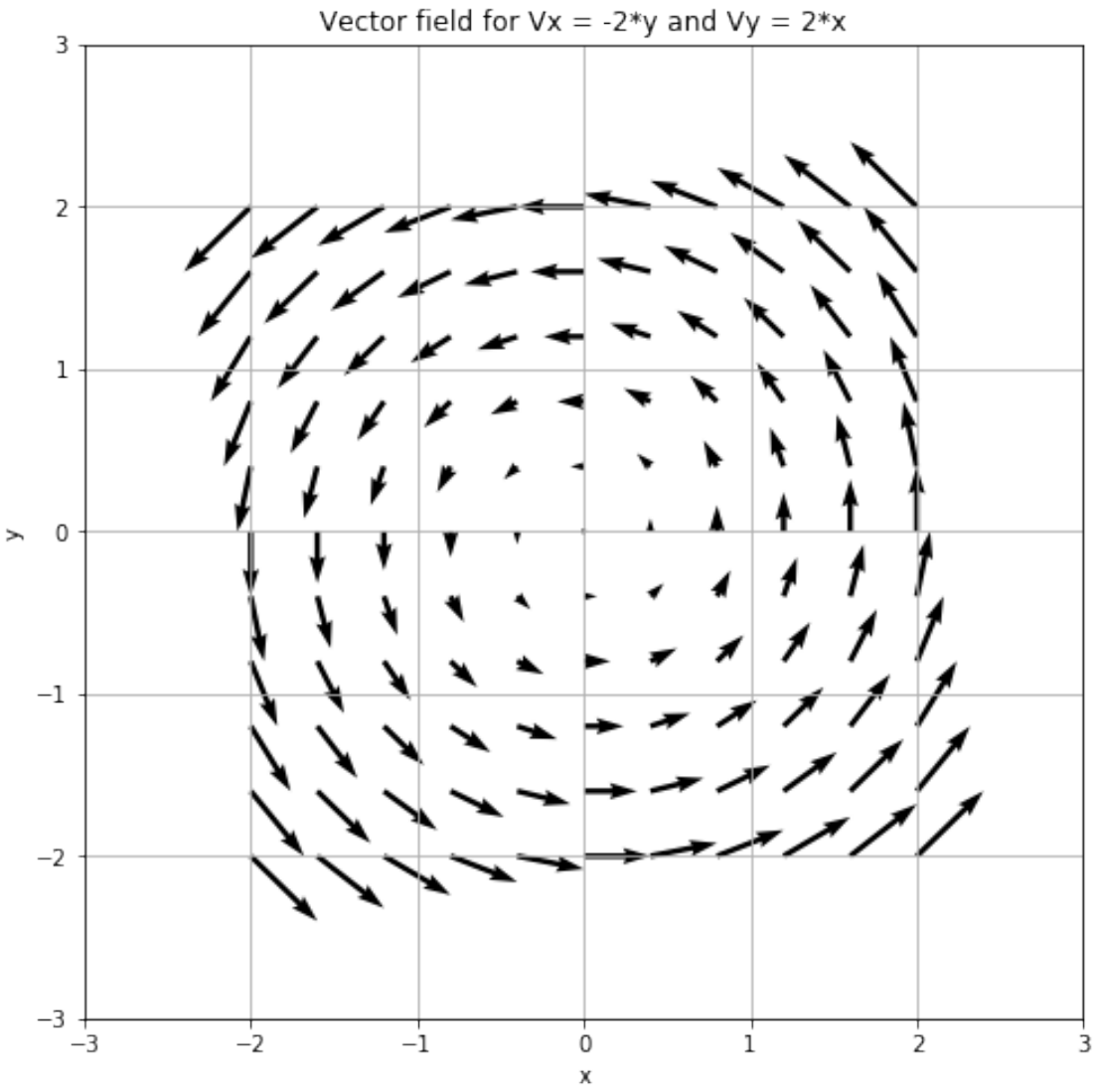
Let us plot the vector field to visualize

[8]:
```
plt.figure(num=1, figsize=(8, 8))
plt.title('Vector field for Vx = ' + str(symbolic_velocity_x) + ' and Vy = ' +␣
 ↪str(symbolic_velocity_y))
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(grid_spacing_start-1, grid_spacing_end+1)
plt.ylim(grid_spacing_start-1, grid_spacing_end+1)

# 0.0 stands for t=0 -> considering one certain time step
```

```
t = 0.0
plt.quiver(meshgrid_x, meshgrid_y,
           velocity_x(t, meshgrid_x, meshgrid_y), velocity_y(t, meshgrid_x,␣
 ↪meshgrid_y),
           angles='xy',scale_units='xy',scale=10)

#plt.axis('equal')
plt.grid()
```



1.0.4 Exercise 1: Visualizing different fields

Try to visualize following vector fields and compute their divergence and curl. What is the relationship between these quantities and the vector fields? Modify in block 3 and not block 4!

1. constant u = [2, 2, 0]
2. constant u = [2, -2, 0]
3. u = [x, y, 0]
4. u = [2y, -2x, 0]
5. u = [x-y, x+y, 0]

1.0.5 Streamline visualization

using the inbuilt function of streamplot to plot the streamlines

[9]:
```
plt.figure(num=2, figsize=(8, 8))

# 0.0 stands for t=0 -> considering one certain time step
t = 0.0
plt.streamplot(meshgrid_x, meshgrid_y,
               velocity_x(t, meshgrid_x, meshgrid_y), velocity_y(t, meshgrid_x,␣
 ↪meshgrid_y),
               density= 0.5)

plt.title('Streamlines for vector field')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([grid_spacing_start-1, grid_spacing_end+1])
plt.ylim([grid_spacing_start-1, grid_spacing_end+1])
plt.grid(True)
```