Study on phase portraits and streamlines

Chahak Mehta

08 May, 2021

Contents

1 Introduction to Streamlines

Let a vector field in an environment be defined as

$$\boldsymbol{V}(x, y, z, t) = u\hat{i} + v\hat{j} + w\hat{k}$$

A *streamline* is defined as a line which is parallel everywhere to the local velocity vector at any point. Let

$$d\boldsymbol{s} = dx\hat{i} + dy\hat{j} + dz\hat{k}$$

be an infinitesimal arc-length vector along the streamline. Since this is parallel to $\boldsymbol{V}$ at that point,

$$d\boldsymbol{s} \times \boldsymbol{V} = 0$$
$$(wdy - vdz)\hat{i} + (udz - wdx)\hat{j} + (vdx - udy)\hat{k} = 0$$

In 2-D, we have $dz = 0$ and $w = 0$, and only the $\hat{k}$ component of the above equation is non-trivial. In that case, we can write the streamline shape as an ODE

$$\frac{dy}{dx} = \frac{v}{u} \tag{1}$$

where the initial conditions at $t = 0$ is represented by $x_0 = 0, y_0 = 0$. We can get the streamlines of the system by integrating the above equation and plotting the curves defined for (x, y).

2 Phase portraits

Now that we have some background of streamlines, we will see how it relates to phase portraits of nonlinear systems. I will be closely following the book Nonlinear Dynamics and Chaos, by Steven Strogatz. Let the general form of a vector field on the phase plane be given as

$$\dot{x_1} = f_1(x_1, x_2)$$
$$\dot{x_2} = f_2(x_1, x_2)$$

where f_1 and f_2 are given functions defining the system. This system can be represented in a more compact vector notation as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{2}$$

where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$. Here, $\boldsymbol{x}$ represents a point in the phase plane, and $\dot{\boldsymbol{x}}$ is the velocity vector at that point.

3 How to plot and animate streamlines

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from itertools import product
from matplotlib.animation import FuncAnimation
from matplotlib.collections import LineCollection

def ode(t, z):
    x, y = z
    return [-x + x**3, -2*y]
```

```
t = np.linspace(1, 8, 1000)
x_space, y_space = np.linspace(-np.pi/2 - 1, np.pi/2+1, 20), np.linspace(-np.pi/2-1, np
solutions = []

for x0, y0 in product(x_space, y_space):
    sol = solve_ivp(ode, (1, 20), (x0, y0), t_eval=t)
    solutions.append(sol['y'])

fig, ax = plt.subplots()
for sol in solutions:
    ax.plot(sol[0], sol[1], 'k', linewidth=0.7)
ax.set_xlim(-1.5, 1.5)
ax.set_ylim(-1.5, 1.5)
ax.set_xticks([])
ax.set_yticks([])
fig.tight_layout()
fig.savefig('phase.png', dpi=300)

fig_anim, ax_anim = plt.subplots()
ax_anim.set_xticks([])
ax_anim.set_yticks([])
ax_anim.set_xlim(-1.75, 1.75)
ax_anim.set_ylim(-2, 2)

def update(frame):
    ax_anim.collections = []
    if frame < 140:
length = [int(0.1*sol.shape[1]) for sol in solutions]
xframe = [sol[0, :frame+l] for l, sol in zip(length, solutions)]
yframe = [sol[1, :frame+l] for l, sol in zip(length, solutions)]
stack = [np.column_stack([x, y]) for x, y in zip(xframe, yframe)]
segments = LineCollection(stack, color='k', linewidth=0.7)
ax_anim.add_collection(segments)
    return

anim = FuncAnimation(fig_anim, update, frames=200)
# anim.save('phase.mp4', writer='ffmpeg', fps=30)
anim.save('phase-small.gif', writer='imagemagick', fps=30)
```