IMDS Individual Assignment 1

Chen-Wei Huang

04/11/2025

Question 1

(a) Modelling infection spread

During the summer months the infection spreads exponentially, while during the winter the transmission rate is near zero. To capture both long-term exponential growth and seasonal variation, we model the rate of new cases as

$$f(t) = C_0 e^{rt} S(t), \qquad t \text{ measured in years,}$$

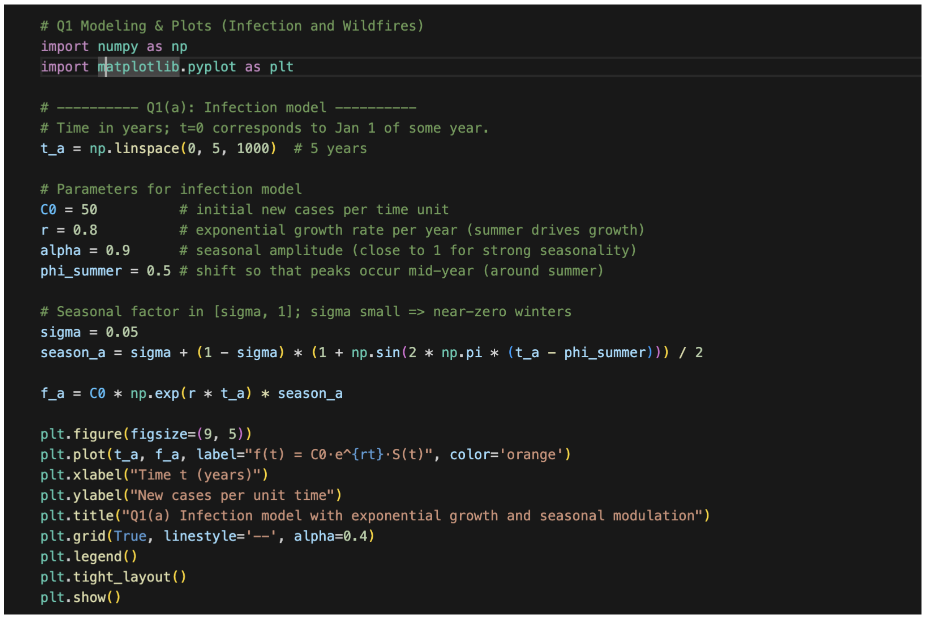
where $C_0 > 0$ is the initial infection rate, $r > 0$ is the exponential growth rate, and $S(t)$ is a seasonal modulation factor. We choose

$$S(t) = \sigma + (1-\sigma)\frac{1+\sin\big(2\pi(t-\phi)\big)}{2},$$

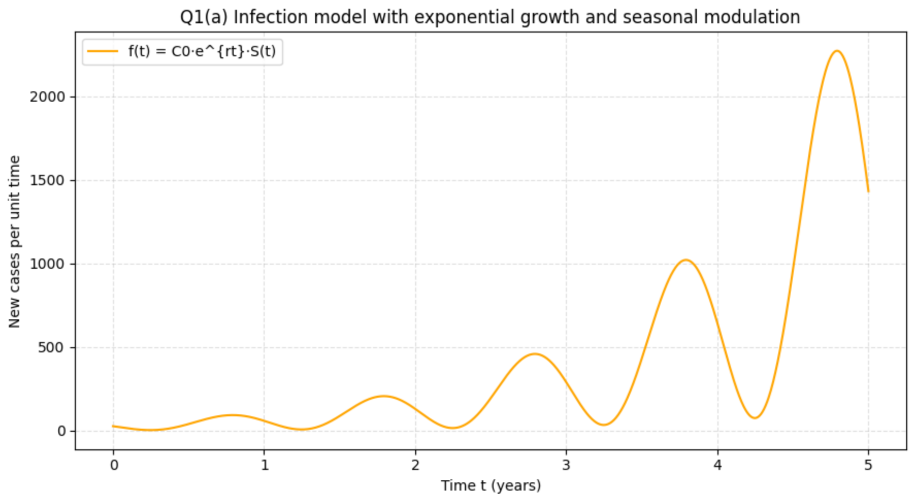
where $0 < \sigma \ll 1$ ensures the value approaches zero in winter, and ϕ shifts the summer peak to mid-year.

In summer, $\sin(\cdot) \approx 1$ and thus $S(t) \approx 1$; in winter $\sin(\cdot) \approx -1$ and $S(t) \approx \sigma$, so transmission is near zero. This matches the biological description of seasonally-driven disease dynamics.

A plot of this model, generated in Python, is shown below.



```
# Q1 Modeling & Plots (Infection and Wildfires)
import numpy as np
import matplotlib.pyplot as plt

# ---------- Q1(a): Infection model ----------
# Time in years; t=0 corresponds to Jan 1 of some year.
t_a = np.linspace(0, 5, 1000)  # 5 years

# Parameters for infection model
C0 = 50          # initial new cases per time unit
r = 0.8          # exponential growth rate per year (summer drives growth)
alpha = 0.9      # seasonal amplitude (close to 1 for strong seasonality)
phi_summer = 0.5 # shift so that peaks occur mid-year (around summer)

# Seasonal factor in [sigma, 1]; sigma small => near-zero winters
sigma = 0.05
season_a = sigma + (1 - sigma) * (1 + np.sin(2 * np.pi * (t_a - phi_summer))) / 2

f_a = C0 * np.exp(r * t_a) * season_a

plt.figure(figsize=(9, 5))
plt.plot(t_a, f_a, label="f(t) = C0·e^{rt}·S(t)", color='orange')
plt.xlabel("Time t (years)")
plt.ylabel("New cases per unit time")
plt.title("Q1(a) Infection model with exponential growth and seasonal modulation")
plt.grid(True, linestyle='--', alpha=0.4)
plt.legend()
plt.tight_layout()
plt.show()
```



(b) Modelling wildfire frequency in Wales

Wildfires were historically rare but increased between 2000 and 2025 and have now stabilised at approximately 20 events per year. The seasonal pattern features summer peaks and near-zero winter activity.

We therefore model the long-term trend using a logistic curve

$$L(t) = \frac{K}{1 + e^{-k(t-t_0)}}, \qquad t \text{ years since 2000,}$$

where $K \approx 20$ is the saturation level, $k > 0$ determines growth steepness, and $t_0 \approx 15$ corresponds to the midpoint around year 2015.

The seasonal component must peak in summer and be near zero in winter, without changing the annual mean. We use

$$\tilde{S}(t) = \tau + (1-\tau)\frac{1 + \sin\big(2\pi(t-\phi)\big)}{2}, \qquad \bar{S} = \frac{1+\tau}{2}, \qquad S(t) = \frac{\tilde{S}(t)}{\bar{S}},$$

so that $S(t)$ has mean 1. Here $0 < \tau \ll 1$ ensures winter values are very small.

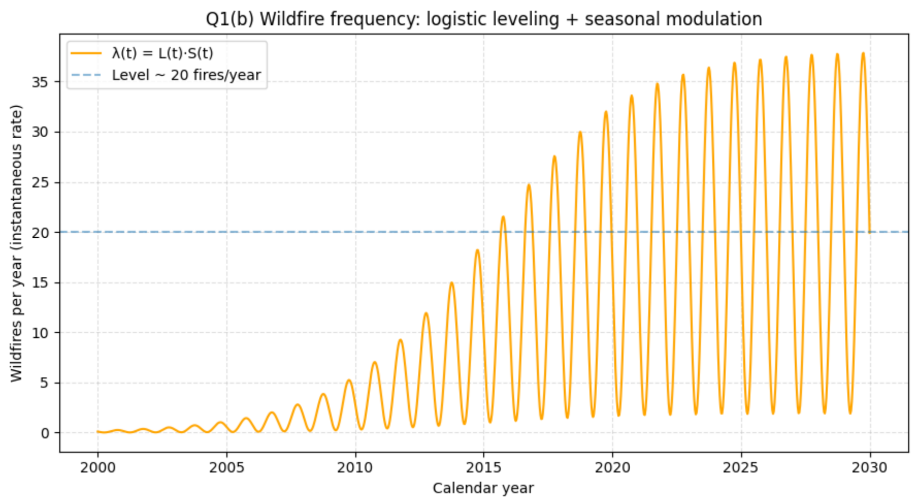
The full model is therefore

$$\lambda(t) = L(t)S(t),$$

which represents the annual wildfire rate modulated by seasonality.

A plot generated using Python is shown below.

```python
# ---------- Q1(b): Wildfire frequency model ----------
# Time in years since 2000; t=0 => year 2000
t_b = np.linspace(0, 30, 1000)  # 2000–2030 for context

# Logistic long-term trend leveling at K ≈ 20 fires/year
K = 20          # carrying capacity (leveling frequency)
k = 0.35        # logistic growth steepness
t0 = 15         # midpoint year (2000 + 15 = 2015)
L = K / (1 + np.exp(-k * (t_b - t0)))  # long-term average frequency

# Seasonal factor with mean 1 but near-zero winters
tau = 0.05  # winter floor (near zero)
season_raw = tau + (1 - tau) * (1 + np.sin(2 * np.pi * (t_b - phi_summer))) / 2
season_mean = (1 + tau) / 2  # mean of season_raw
season_b = season_raw / season_mean  # renormalize to mean 1

lambda_b = L * season_b  # instantaneous frequency per unit time

plt.figure(figsize=(9, 5))
plt.plot(t_b + 2000, lambda_b, label="λ(t) = L(t)·S(t)", color="orange")
plt.axhline(K, linestyle='--', alpha=0.5, label="Level ~ 20 fires/year")
plt.xlabel("Calendar year")
plt.ylabel("Wildfires per year (instantaneous rate)")
plt.title("Q1(b) Wildfire frequency: logistic leveling + seasonal modulation")
plt.grid(True, linestyle='--', alpha=0.4)
plt.legend()
plt.tight_layout()
plt.show()
```



Both models reproduce the qualitative features described: exponential or logistic long-term dynamics combined with near-zero winter levels and strong summer peaks.

Question 2

(a) Estimating $f'(5)$

To estimate the derivative at $x = 5$, we approximate the slope of the curve using two nearby points read from the graph:

$$f(4) \approx 4, \qquad f(6) \approx 2.2.$$

Thus,

$$f'(5) \approx \frac{2.2 - 4}{6 - 4} = -0.9.$$

Rounding to the nearest 0.5 gives

$$\boxed{f'(5) \approx -1.0.}$$

(b) Estimating $\int_6^{10} f(x)\,dx$

To estimate the area under the curve from $x = 6$ to $x = 10$, the region is divided into trapezoids. Approximate values read from the graph are:

$$f(6) \approx 2.2, \quad f(7) \approx 3.8, \quad f(8) \approx 6.8, \quad f(9) \approx 7.8, \quad f(10) \approx 7.2.$$

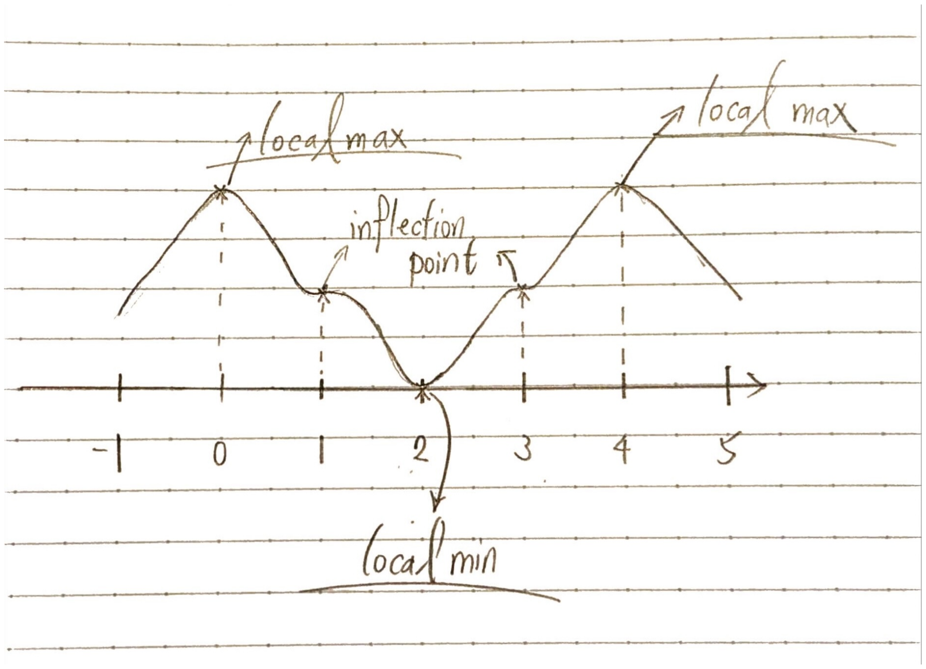
Applying the trapezoidal rule,

$$\frac{2.2 + 3.8}{2} \cdot 1 + \frac{3.8 + 6.8}{2} \cdot 1 + \frac{(6.8 + 7.8) \cdot 0.8}{2} + \frac{(7.8 + 7.2) \cdot 1.2}{2} \approx 23.1.$$

Therefore,

$$\boxed{\int_6^{10} f(x)\,dx \approx 23.1}$$

(c) Sketch of $h(x)$ on $[-1, 5]$ Showing Local Extrema and Inflection Points



Question 3

We are given a table of values of $g(x,y)$ on a uniform grid with spacing 0.1 in both variables. We estimate the gradient at $(x,y) = (0.2, 0.4)$ using central differences.

(a) Estimating the gradient $\nabla g(0.2, 0.4)$

The central difference approximation for the partial derivatives is

$$\frac{\partial g}{\partial x}(0.2,0.4) \approx \frac{g(0.3,0.4) - g(0.1,0.4)}{0.2}, \qquad \frac{\partial g}{\partial y}(0.2,0.4) \approx \frac{g(0.2,0.5) - g(0.2,0.3)}{0.2}.$$

From the table,

$$g(0.3,0.4) = -0.594, \quad g(0.1,0.4) = -1.034,$$
$$g(0.2,0.5) = -0.815, \quad g(0.2,0.3) = -0.757.$$

Thus,

$$\frac{\partial g}{\partial x}(0.2,0.4) \approx \frac{-0.594 - (-1.034)}{0.2} = \frac{0.440}{0.2} = 2.2,$$
$$\frac{\partial g}{\partial y}(0.2,0.4) \approx \frac{-0.815 - (-0.757)}{0.2} = \frac{-0.058}{0.2} = -0.29.$$

Therefore,

$$\boxed{\nabla g(0.2,0.4) \approx (2.2,\ -0.29)}.$$

(b) Directional derivative along $(7,8)$

We compute the directional derivative using

$$D_{\mathbf{u}}g = \nabla g \cdot \hat{\mathbf{u}}, \qquad \hat{\mathbf{u}} = \frac{(7,8)}{\sqrt{7^2+8^2}} = \frac{(7,8)}{\sqrt{113}}.$$

Thus,

$$\nabla g(0.2,0.4) \cdot (7,8) = (2.2)(7) + (-0.29)(8) = 15.4 - 2.32 = 13.08,$$

$$D_{\mathbf{u}}g(0.2,0.4) = \frac{13.08}{\sqrt{113}} \approx \frac{13.08}{10.63} \approx 1.23.$$

Therefore,

$$\boxed{D_{\mathbf{u}}g(0.2,0.4) \approx 1.23.}$$

Question 4

We consider the function

$$F(x,y) = x^2 + y^2 - 6\sin^2(x-y).$$

To perform gradient descent, we first compute the gradient. Using the chain rule,

$$\frac{\partial}{\partial x}\sin^2(x-y) = 2\sin(x-y)\cos(x-y), \qquad \frac{\partial}{\partial y}\sin^2(x-y) = -2\sin(x-y)\cos(x-y).$$

Thus,

$$\nabla F(x,y) = \begin{pmatrix} 2x - 12\sin(x-y)\cos(x-y) \\ 2y + 12\sin(x-y)\cos(x-y) \end{pmatrix}.$$

We apply gradient descent with step size $\alpha = 0.1$, starting from the point

$$(x_0, y_0) = (-3.2,\ -3).$$

The iterative update rule is

$$(x_{k+1}, y_{k+1}) = (x_k, y_k) - \alpha \nabla F(x_k, y_k).$$

Computing three iterations yields:

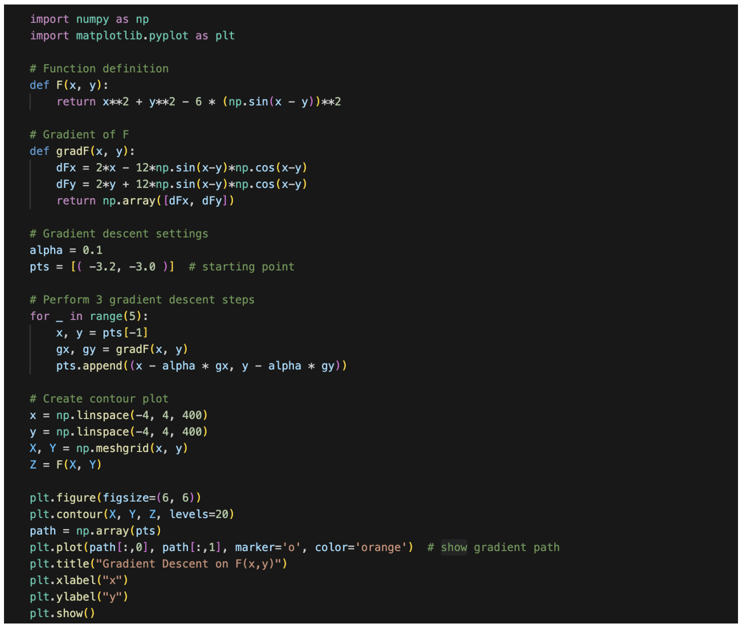
$$\begin{aligned} (x_1, y_1) &\approx (-2.7937,\ -2.1663), \\ (x_2, y_2) &\approx (-2.8052,\ -1.1628), \\ (x_3, y_3) &\approx (-2.1586,\ -1.0158). \end{aligned}$$

The path of three gradient descent steps is therefore:

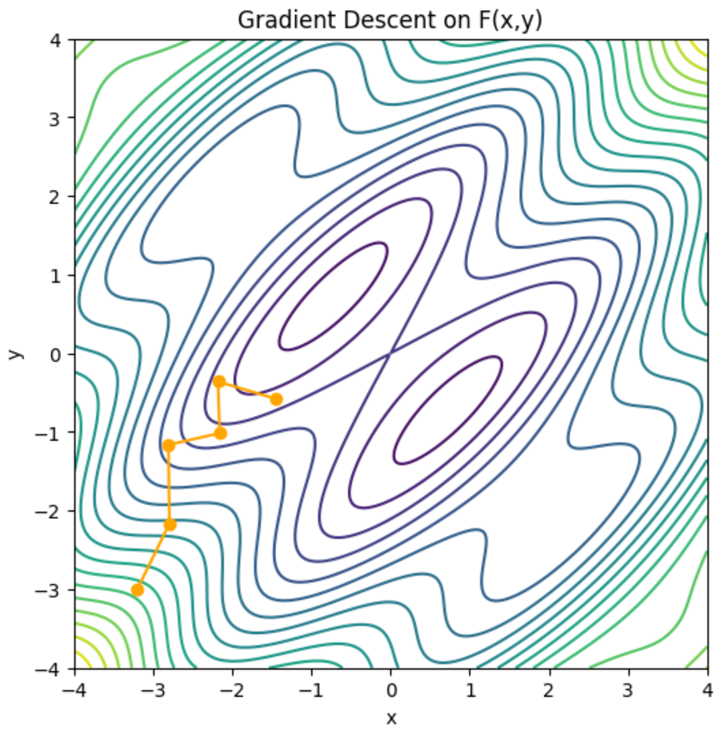
$$(-3.2, -3) \ \to\ (-2.7937, -2.1663) \ \to\ (-2.8052, -1.1628) \ \to\ (-2.1586, -1.0158).$$

Iteration	x	y
0	-3.2	-3.0
1	-2.7937	-2.1663
2	-2.8052	-1.1628
3	-2.1586	-1.0158

A contour plot of F with the gradient descent trajectory is shown below.



```python
import numpy as np
import matplotlib.pyplot as plt

# Function definition
def F(x, y):
    return x**2 + y**2 - 6 * (np.sin(x - y))**2

# Gradient of F
def gradF(x, y):
    dFx = 2*x - 12*np.sin(x-y)*np.cos(x-y)
    dFy = 2*y + 12*np.sin(x-y)*np.cos(x-y)
    return np.array([dFx, dFy])

# Gradient descent settings
alpha = 0.1
pts = [( -3.2, -3.0 )]  # starting point

# Perform 3 gradient descent steps
for _ in range(5):
    x, y = pts[-1]
    gx, gy = gradF(x, y)
    pts.append((x - alpha * gx, y - alpha * gy))

# Create contour plot
x = np.linspace(-4, 4, 400)
y = np.linspace(-4, 4, 400)
X, Y = np.meshgrid(x, y)
Z = F(X, Y)

plt.figure(figsize=(6, 6))
plt.contour(X, Y, Z, levels=20)
path = np.array(pts)
plt.plot(path[:,0], path[:,1], marker='o', color='orange')  # show gradient path
plt.title("Gradient Descent on F(x,y)")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



The trajectory moves toward a region of lower function value, demonstrating the expected behavior of gradient descent.

Question 5

Let

$$\mathbf{u} = (9, 12), \qquad \mathbf{v} = (-1, 7).$$

(a) Lengths and the angle between u and v

The magnitudes are

$$\|\mathbf{u}\| = \sqrt{9^2 + 12^2} = \sqrt{225} = 15, \qquad \|\mathbf{v}\| = \sqrt{(-1)^2 + 7^2} = \sqrt{50} = 5\sqrt{2}.$$

The dot product is

$$\mathbf{u} \cdot \mathbf{v} = 9(-1) + 12(7) = -9 + 84 = 75.$$

Hence

$$\cos\theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|} = \frac{75}{15 \cdot 5\sqrt{2}} = \frac{75}{75\sqrt{2}} = \frac{1}{\sqrt{2}},$$

so

$$\boxed{\theta = \arccos\Big(\tfrac{1}{\sqrt{2}}\Big) = \frac{\pi}{4} = 45^\circ.}$$

Therefore,

$$\boxed{\|\mathbf{u}\| = 15, \quad \|\mathbf{v}\| = 5\sqrt{2}, \quad \theta = \frac{\pi}{4}\ (45^\circ)}.$$

(b) Orthogonal projection $\mathrm{Proj}_{\mathbf{v}}(\mathbf{u})$

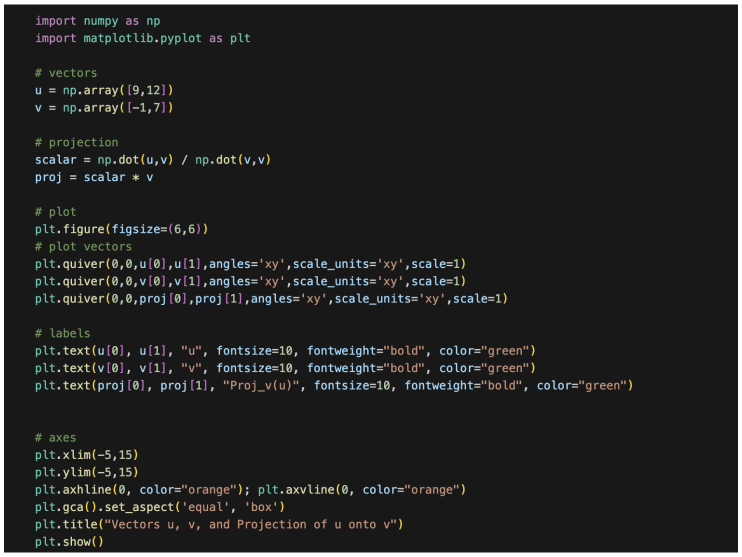
By definition,

$$\mathrm{Proj}_{\mathbf{v}}(\mathbf{u}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}}\,\mathbf{v} = \frac{75}{(-1)^2 + 7^2}\,(-1, 7) = \frac{75}{50}\,(-1, 7) = 1.5\,(-1, 7).$$

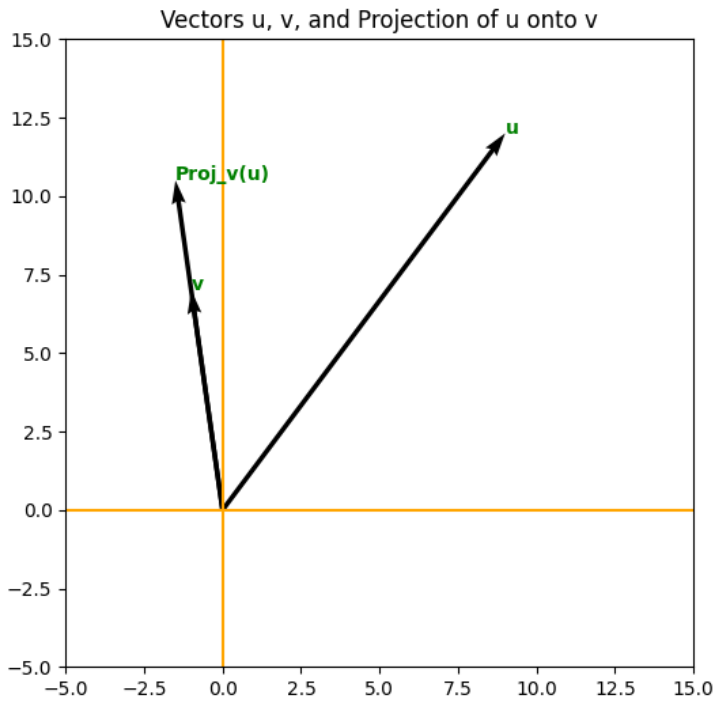
Thus

$$\boxed{\mathrm{Proj}_{\mathbf{v}}(\mathbf{u}) = (-1.5,\ 10.5)}.$$

Illustration



```python
import numpy as np
import matplotlib.pyplot as plt

# vectors
u = np.array([9,12])
v = np.array([-1,7])

# projection
scalar = np.dot(u,v) / np.dot(v,v)
proj = scalar * v

# plot
plt.figure(figsize=(6,6))
# plot vectors
plt.quiver(0,0,u[0],u[1],angles='xy',scale_units='xy',scale=1)
plt.quiver(0,0,v[0],v[1],angles='xy',scale_units='xy',scale=1)
plt.quiver(0,0,proj[0],proj[1],angles='xy',scale_units='xy',scale=1)

# labels
plt.text(u[0], u[1], "u", fontsize=10, fontweight="bold", color="green")
plt.text(v[0], v[1], "v", fontsize=10, fontweight="bold", color="green")
plt.text(proj[0], proj[1], "Proj_v(u)", fontsize=10, fontweight="bold", color="green")


# axes
plt.xlim(-5,15)
plt.ylim(-5,15)
plt.axhline(0, color="orange"); plt.axvline(0, color="orange")
plt.gca().set_aspect('equal', 'box')
plt.title("Vectors u, v, and Projection of u onto v")
plt.show()
```



The figure indicates **u**, **v**, and the projection $\text{Proj}_{\mathbf{v}}(\mathbf{u})$ along the direction of **v**.