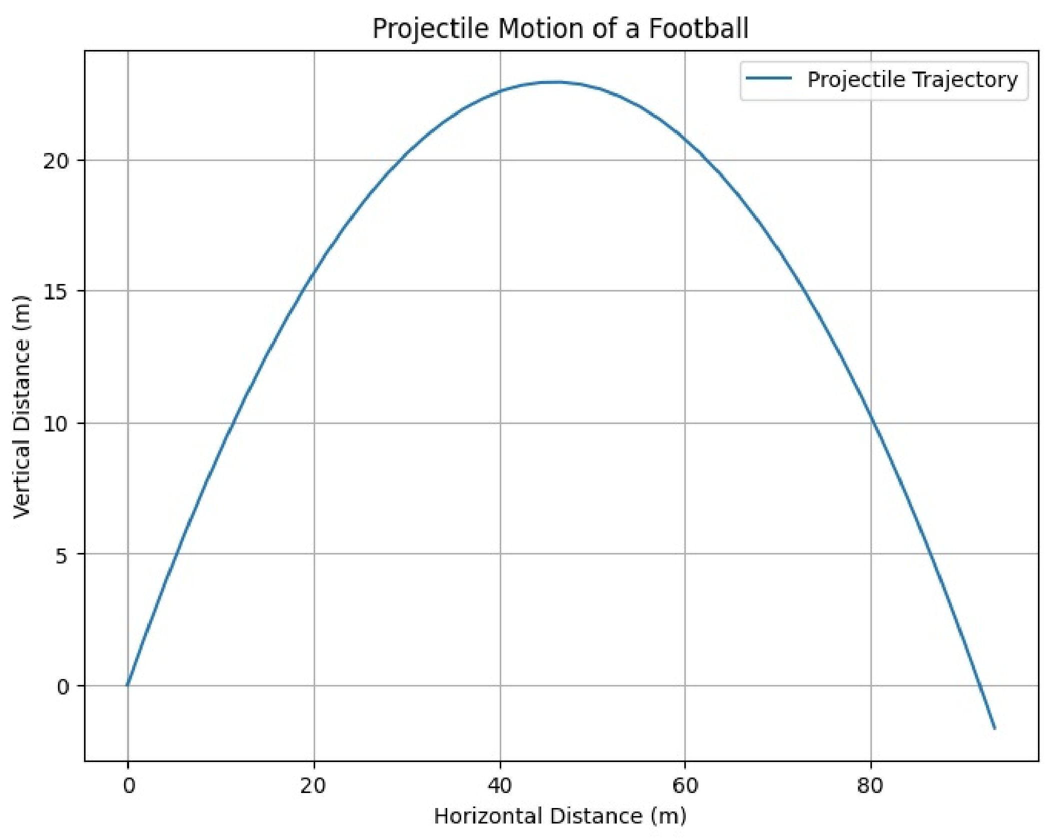
Numerical Elements -- Kicking a Football

A 1 kg football is kicked from the ground with an initial velocity of $v_0 = 30 m/s$ at an angle of $\theta = 45$ above the horizontal.

a. (5pts) First, let's assume that air resistance is negligible and $g = 9.81 m/s^2$.

1. Initialize any constants and define the initial conditions. **Remember:** `np.cos()` and `np.sin()` use radians and not degrees.
2. Find the trajectories of the motion for the projectile analytically (e.g. *x(t)* and *y(t)*).
3. Use these equations to calculate: (i) the maximum height reached by the ball, (ii) the total time of flight, and (iii) the horizontal range of the ball.
4. Using this information from part 2, write a program to simulate the projectile motion and compute the same quantities numerically. You should use a time step of $\Delta t = 0.1$
5. Plot the trajectory of the projectile (e.g. y vs. x) and print the maximum height, the total time of flight, and the horizontal range of the ball.

In [4]:
```python
#Part_A_5pts

import numpy as np
import matplotlib.pyplot as plt


# Constants
g = 9.81  # acceleration due to gravity (m/s^2)
v0 = 30  # initial velocity (m/s)
theta_deg = 45  # launch angle in degrees
theta_rad = np.radians(theta_deg)  # launch angle in radians
m = 1  # mass of the football (kg)

# Initial conditions
v0x = v0 * np.cos(theta_rad)  # initial velocity in x direction (m/s)
v0y = v0 * np.sin(theta_rad)  # initial velocity in y direction (m/s)

# Analytical equations for x(t) and y(t)
# x(t) = v0x * t
# y(t) = v0y * t - 0.5 * g * t^2

# Maximum height (v0y^2 / (2 * g))
t_max_height = v0y / g
y_max = (v0y**2) / (2 * g)

# Total time of flight (2 * v0y / g)
t_flight = 2 * v0y / g

# Horizontal range (v0x * total time of flight)
range_ = v0x * t_flight

# Numerical simulation parameters
delta_t = 0.1  # time step (s)
t_values = np.arange(0, t_flight + delta_t, delta_t)

# Calculate x(t) and y(t) values for the simulation
x_values = v0x * t_values
y_values = v0y * t_values - 0.5 * g * t_values**2

# Plotting the trajectory
plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label="Projectile Trajectory")
plt.title("Projectile Motion of a Football")
plt.xlabel("Horizontal Distance (m)")
plt.ylabel("Vertical Distance (m)")
plt.grid(True)
plt.legend()
plt.show()

# Printing the results
max_height = y_max
total_time = t_flight
horizontal_range = range_

(max_height, total_time, horizontal_range)
```

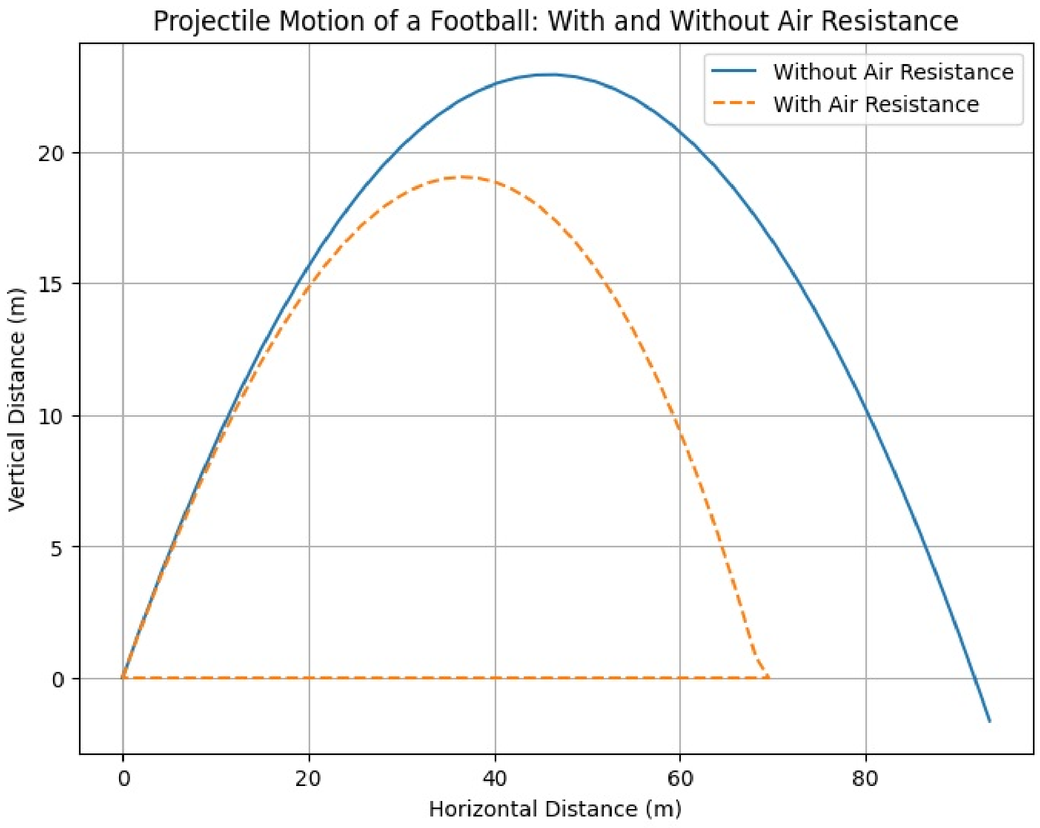


```
Out[4]: (22.935779816513755, 4.324812117348913, 91.74311926605502)
```

b. (5pts) Now consider the ball experiences a linear drag force proportional to its velocity, modeled as $F_d = -b\mathbf{v}$, where $b = 0.1 kg/s$ is the drag coefficient.

1. Initialize any new constants. Again, **remember:** `np.cos()` and `np.sin()` use radians and not degrees.
2. Find the trajectories of the motion for the projectile analytically (e.g. *x(t)* and *y(t)*). You can reference your notes from class lecture and/or homework problems.
3. Using this information from part 2, write a program to simulate the projectile motion and compute the same quantities numerically. You should use a time step of $\Delta t = 0.1$
4. Plot the trajectory of the projectile (e.g. y vs. x) on the same plot as the trajectory of the football assuming no air resistance.

```
In [5]: #Part_B_5pts

# Constants for drag
b = 0.1  # drag coefficient (kg/s)

# Numerical simulation with drag
t_values_drag = np.arange(0, t_flight + delta_t, delta_t)
x_values_drag = np.zeros_like(t_values_drag)
y_values_drag = np.zeros_like(t_values_drag)

# Initial velocities with drag
vx = v0x
vy = v0y

# Euler's method for updating position and velocity with drag
for i in range(1, len(t_values_drag)):
    t = t_values_drag[i]
    # Update velocities
    vx = vx * np.exp(-b * delta_t / m)
    vy = (vy + g * m / b) * np.exp(-b * delta_t / m) - g * m / b
    # Update positions
    x_values_drag[i] = x_values_drag[i-1] + vx * delta_t
    y_values_drag[i] = y_values_drag[i-1] + vy * delta_t

    # Stop if the projectile hits the ground
    if y_values_drag[i] < 0:
        y_values_drag[i] = 0
        break

# Plotting both trajectories: without and with air resistance
plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label="Without Air Resistance")
plt.plot(x_values_drag, y_values_drag, label="With Air Resistance", linestyle='--')
plt.title("Projectile Motion of a Football: With and Without Air Resistance")
plt.xlabel("Horizontal Distance (m)")
plt.ylabel("Vertical Distance (m)")
plt.grid(True)
plt.legend()
plt.show()
```

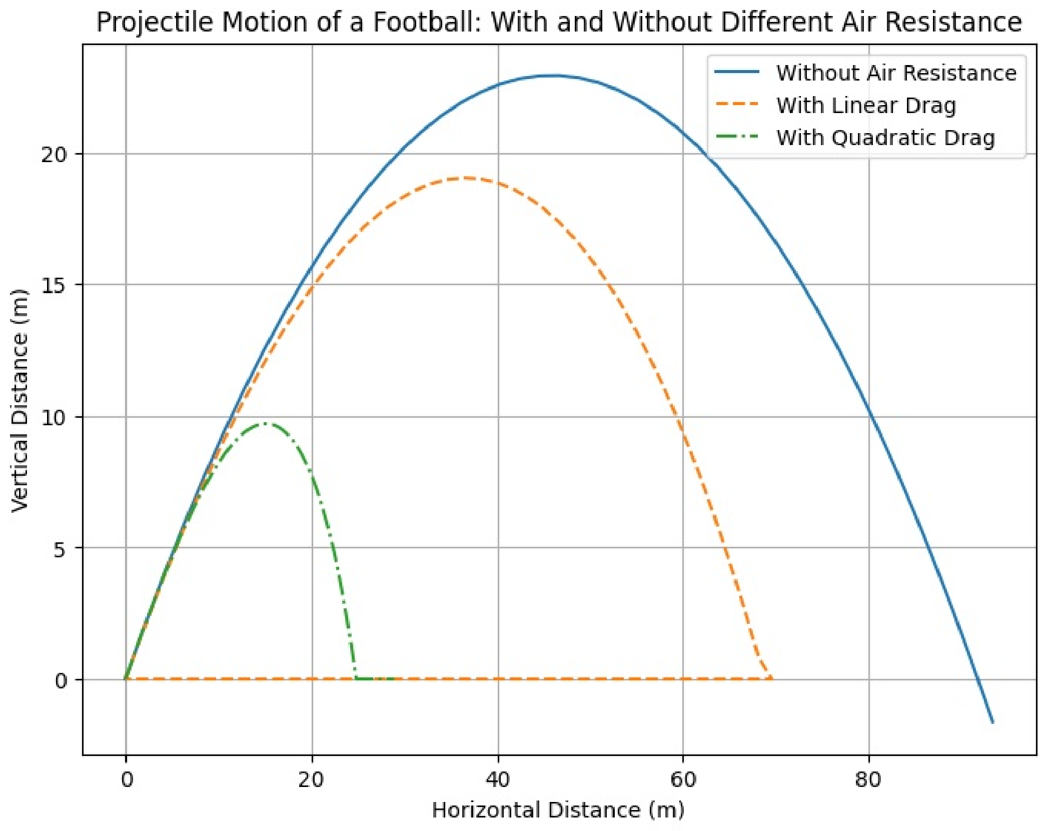


c. (10pts) Now consider that the football experiences a quadratic drag force, modeled as $F_d = -c_d v^2 \hat{v}$, where $c_d = 0.05 kg/m$ is the drag coefficient.

1. Initialize any new constants. Again, **remember:** `np.cos()` and `np.sin()` use radians and not degrees.
2. Find the trajectories of the motion for the projectile analytically (e.g. *x(t)* and *y(t)*). You can reference your notes from class lecture and/or homework problems.
3. Here we will finally use python to help us solve our coupled differential equations. Using the trajectories from part 2 and your knowledge about the relationship between position and velocity, complete the program to simulate the projectile motion. For more information on the `solve_ivp` function from `scipy.integrate` see the link **here**.
4. Plot the trajectory of the projectile (e.g. y vs. x) on the same plot as above.

In [8]:
```
from scipy.integrate import solve_ivp

# Constants for quadratic drag
cd = 0.05  # quadratic drag coefficient (kg/m)

# Define the system of equations with quadratic drag
def equations(t, state):
    # Unpack the state vector: [x, y, vx, vy]
    x, y_pos, vx, vy = state

    # Magnitude of velocity
    v = np.sqrt(vx**2 + vy**2)

    # Equations of motion under quadratic drag
    dvx_dt = -cd * v * vx / m
    dvy_dt = -cd * v * vy / m - g

    return [vx, vy, dvx_dt, dvy_dt]

# Initial conditions: [x0, y0, vx0, vy0]
y0 = [0, 0, v0x, v0y]  # initial positions and velocities in x and y directions

# Time span for the simulation
t_span = (0, t_flight)  # simulate for total time of flight
t_eval = np.linspace(t_span[0], t_span[1], 500)  # time points for evaluation

# Solve the ODE
sol = solve_ivp(equations, t_span, y0, t_eval=t_eval)

# Extract the results
x_num = sol.y[0]  # x positions
y_num = sol.y[1]  # y positions
```

```python
# Stop if the projectile hits the ground
y_num = np.where(y_num < 0, 0, y_num)

# Plotting all trajectories
plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label="Without Air Resistance")
plt.plot(x_values_drag, y_values_drag, label="With Linear Drag", linestyle='--')
plt.plot(x_num, y_num, label="With Quadratic Drag", linestyle='-.')
plt.title("Projectile Motion of a Football: With and Without Different Air Resistance")
plt.xlabel("Horizontal Distance (m)")
plt.ylabel("Vertical Distance (m)")
plt.grid(True)
plt.legend()
plt.show()
```



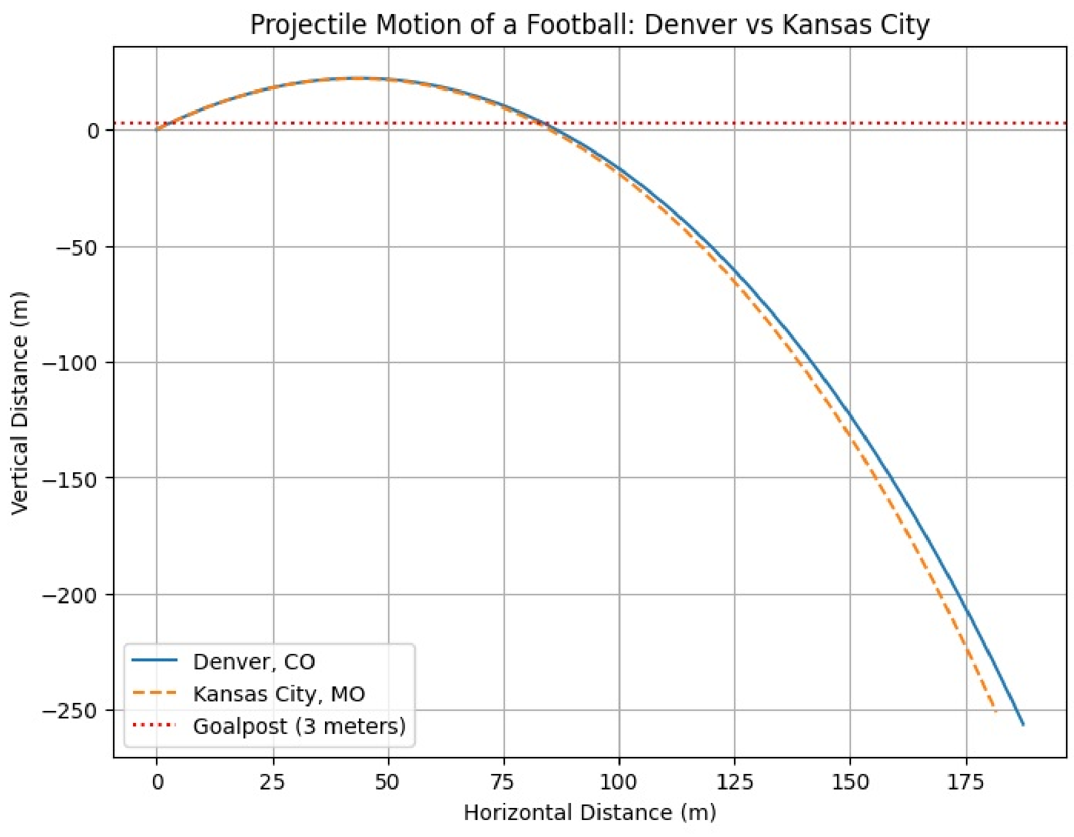
d. (20pts) Now that we have code to solve coupled differential equations, let's explore a real world scenario.

The Denver Broncos are scheduled to play against the Kansas City Chiefs on November 10th. The field goal kicker for the Broncos is investigating his maximum distance he can successfully kick the football to score points for his team. Let's assume that the football experiences a quadratic drag force, modeled as $F_d = -\frac{1}{2}\rho c_d A v^2 \hat{v}$ in both cities and the field goal kicker strikes the football with an initial velocity of $v_0 = 30 m/s$ and an angle of $\theta = 45$ from the horizontal each time he kicks.

Using the `solve_ivp` code from above, plot the trajectories of the football in Denver, CO and Kansas City, MO. You will need to use the internet to find the elevation and air density for each city and assume that $c_d = 0.05 kg/m$ and the cross-sectional area of a football is $A = 0.038 m^2$.

1. If the goal post is 3 meters above the ground, what is the longest field goal the kicker can successfully make in Denver?
2. If the goal post is 3 meters above the ground, what is the longest field goal the kicker can successfully make in Kansas City?
3. Discuss how environmental factors, such as altitude and air density, impact the projectile motion.

In [9]:
```python
#Part_D_20pts
# Constants for the simulation
cd = 0.05  # drag coefficient (kg/m)
A = 0.038  # cross-sectional area of the football (m^2)
rho_denver = 0.909  # air density in Denver (kg/m^3)
rho_kc = 1.168  # air density in Kansas City (kg/m^3)
g = 9.81  # acceleration due to gravity (m/s^2)
v0 = 30  # initial velocity (m/s)
theta_deg = 45  # launch angle in degrees
theta_rad = np.radians(theta_deg)  # launch angle in radians

# Initial velocities in x and y directions
v0x = v0 * np.cos(theta_rad)
v0y = v0 * np.sin(theta_rad)

# Define the system of equations with quadratic drag for different air densities
def equations_with_drag(t, state, rho):
    # Unpack the state vector: [x, y, vx, vy]
    x, y_pos, vx, vy = state

    # Magnitude of velocity
    v = np.sqrt(vx**2 + vy**2)

    # Quadratic drag forces
    dvx_dt = -0.5 * rho * cd * A * v * vx / m
    dvy_dt = -0.5 * rho * cd * A * v * vy / m - g

    return [vx, vy, dvx_dt, dvy_dt]

# Initial conditions: [x0, y0, vx0, vy0]
y0 = [0, 0, v0x, v0y]  # initial positions and velocities in x and y directions

# Time span for the simulation
t_span = (0, 10)  # simulate for 10 seconds
t_eval = np.linspace(t_span[0], t_span[1], 500)  # time points for evaluation

# Solve the ODE for Denver and Kansas City
sol_denver = solve_ivp(equations_with_drag, t_span, y0, t_eval=t_eval, args=(rho_denver,))
sol_kc = solve_ivp(equations_with_drag, t_span, y0, t_eval=t_eval, args=(rho_kc,))

# Extract the results
x_denver = sol_denver.y[0]
y_denver = sol_denver.y[1]

x_kc = sol_kc.y[0]
y_kc = sol_kc.y[1]

# Stop when the ball hits the ground or passes 3 meters (goalpost height)
goalpost_height = 3
x_denver_final = x_denver[y_denver >= goalpost_height][-1]
x_kc_final = x_kc[y_kc >= goalpost_height][-1]

# Plotting both trajectories
plt.figure(figsize=(8, 6))
plt.plot(x_denver, y_denver, label="Denver, CO")
plt.plot(x_kc, y_kc, label="Kansas City, MO", linestyle='--')
plt.axhline(y=goalpost_height, color='r', linestyle=':', label="Goalpost (3 meters)")
plt.title("Projectile Motion of a Football: Denver vs Kansas City")
plt.xlabel("Horizontal Distance (m)")
plt.ylabel("Vertical Distance (m)")
plt.grid(True)
plt.legend()
plt.show()

(x_denver_final, x_kc_final)
```



```
Out[9]:  (83.18787960802355, 81.86099273435197)
```

Processing math: 100%