Untitled

May 4, 2023

[15]:
```
def draw_dartboard(ax, center, radius):
    ax.set_aspect('equal')
    ax.set_xlim([-radius, radius])
    ax.set_ylim([-radius, radius])

    circle = plt.Circle(center, radius, fill=False, color='black')
    ax.add_artist(circle)

    circle = plt.Circle(center, radius * 0.1, fill=False, color='black')
    ax.add_artist(circle)

    n = 20
    for i in range(n):
        angle = i * 2 * np.pi / n
        x = center[0] + radius * np.cos(angle)
        y = center[1] + radius * np.sin(angle)
        if radius * 0.15 <= np.sqrt(x ** 2 + y ** 2) <= radius * 0.5:
            continue
        line = plt.Line2D([center[0], x], [center[1], y], color='black',␣
↪linestyle='--')
        ax.add_artist(line)

    circle = plt.Circle(center, radius * 0.9, fill=False, color='black')
    ax.add_artist(circle)

    circle = plt.Circle(center, radius * 0.6, fill=False, color='black')
    ax.add_artist(circle)

    circle = plt.Circle(center, radius * 0.5, fill=False, color='black')
    ax.add_artist(circle)

    circle = plt.Circle(center, radius * 0.15, fill=False, color='black')
    ax.add_artist(circle)

    circle = plt.Circle(center, radius * 0.1, fill=False, color='black')
    ax.add_artist(circle)
    ax.set_ylim(-0.2255*1.5, 0.2255*1.5)
```

```
    ax.set_xlim(-0.2255*1.5, 0.2255*1.5)
```

[34]:
```
# F = -kv
import numpy as np
import matplotlib.pyplot as plt


def calculate_trajectory(alpha, beta, V0, difficulty, x_target=2.37, m=0.025,␣
 ↪k=0.05):
    alpha = np.radians(alpha)
    beta = np.radians(beta)
    value_under_sqrt = 1 - np.cos(alpha)**2 - np.cos(beta)**2
    if value_under_sqrt < 0:
        raise ValueError("Invalid combination of alpha and beta angles.")

    ramda = np.arccos(np.sqrt(value_under_sqrt))

    if difficulty == "beginner":
        V_prime = 5
    elif difficulty == "intermediate":
        V_prime = 10
    elif difficulty == "advanced":
        V_prime = 15
    else:
        raise ValueError("Invalid difficulty level. Choose from 'beginner',␣
 ↪'intermediate', or 'advanced'.")

    # Calculate time of flight using t = R / V0cos(alpha)
    t_flight = x_target / (V0 * np.cos(alpha))

    # Calculate x, y, and z positions over time
    t = np.linspace(0, t_flight, num=1000)
    x_vals = V0 * np.cos(alpha) * t
    y_vals = V0 * np.cos(beta) * t - 0.5 * 9.81 * t**2
    z_vals = (V0 * np.cos(ramda) + V_prime) * m / k * (1 - np.exp(-k * t / m))␣
 ↪- V_prime * t

    return x_vals, y_vals, z_vals

def plot_all_graphs(x_vals, y_vals, z_vals):
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(30, 5))

    center = (0, 0)
    radius = 0.2255

    draw_dartboard(ax1, center, radius)
```

```
    ax1.scatter(z_vals[-1], y_vals[-1], s=50, c='r', marker='o', label='Dart␣
 ↪arrival point')
    ax1.set_xlabel('z')
    ax1.set_ylabel('y')
    ax1.set_title('Dart Arrival Point on (y, z) plane')
    ax1.legend()
    ax1.grid()

    # Plot trajectory on (x, y) plane
    ax2.plot(x_vals, y_vals, label='Trajectory')
    ax2.scatter(x_vals[-1], y_vals[-1], label='Dart arrival point', color='red')
    ax2.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax2.set_ylabel('y')
    ax2.set_title('Dart Trajectory on (x, y) plane')
    ax2.axvline(2.37, color='black', linestyle='--', label='Location of the␣
 ↪Dart Board')
    ax2.legend()
    ax2.grid()

    # Plot trajectory on (x, z) plane
    ax3.plot(x_vals, z_vals, label='Trajectory')
    ax3.scatter(x_vals[-1], z_vals[-1], label='Dart arrival point', color='red')
    ax3.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax3.set_ylabel('z')
    ax3.set_title('Dart Trajectory on (x, z) plane')
    ax3.axvline(2.37, color='black', linestyle='--', label='Location of the␣
 ↪Dart Board')
    ax3.legend()
    ax3.grid()

    plt.show()

if __name__ == "__main__":
    try:
        alpha = float(input("Enter alpha angle (in degrees): "))
        beta = float(input("Enter beta angle (in degrees): "))
        V0 = float(input("Enter initial velocity (in m/s): "))
        difficulty = input("Enter difficulty level (beginner, intermediate,␣
 ↪advanced): ").lower()
        x_vals, y_vals, z_vals = calculate_trajectory(alpha, beta, V0,␣
 ↪difficulty)
        plot_all_graphs(x_vals, y_vals, z_vals)
    except ValueError as e:
        print(e)
```

```
print("The dart hits the target at (z, y) = ({:.2f}, {:.2f})".
 ↪format(z_vals[-1], y_vals[-1]))
```

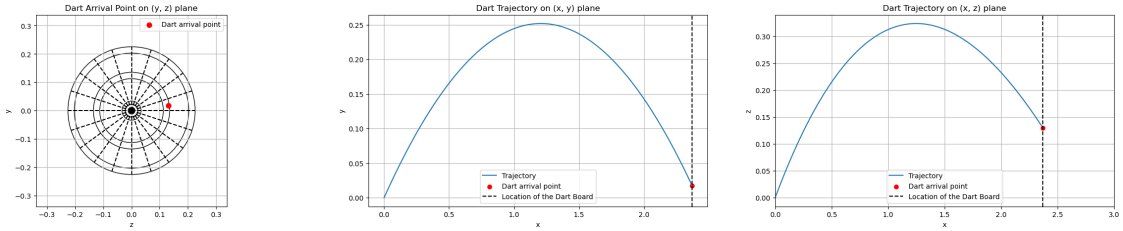
```
Enter alpha angle (in degrees): 20
Enter beta angle (in degrees): 20
Enter initial velocity (in m/s): 10
Enter difficulty level (beginner, intermediate, advanced): beginner
Invalid combination of alpha and beta angles.
The dart hits the target at (z, y) = (-2.09, 0.02)
```

[35]:
```
# F = -kv
import numpy as np
import matplotlib.pyplot as plt


def calculate_trajectory(alpha, beta, V0, difficulty, x_target=2.37, m=0.025,␣
 ↪k=0.05):
    alpha = np.radians(alpha)
    beta = np.radians(beta)
    value_under_sqrt = 1 - np.cos(alpha)**2 - np.cos(beta)**2
    if value_under_sqrt < 0:
        raise ValueError("Invalid combination of alpha and beta angles.")

    ramda = np.arccos(np.sqrt(value_under_sqrt))

    if difficulty == "beginner":
        V_prime = 5
    elif difficulty == "intermediate":
        V_prime = 10
    elif difficulty == "advanced":
        V_prime = 15
    else:
        raise ValueError("Invalid difficulty level. Choose from 'beginner',␣
 ↪'intermediate', or 'advanced'.")

    # Calculate time of flight using t = R / V0cos(alpha)
    t_flight = x_target / (V0 * np.cos(alpha))

    # Calculate x, y, and z positions over time
    t = np.linspace(0, t_flight, num=1000)
    x_vals = V0 * np.cos(alpha) * t
    y_vals = V0 * np.cos(beta) * t - 0.5 * 9.81 * t**2
    z_vals = (V0 * np.cos(ramda) + V_prime) * m / k * (1 - np.exp(-k * t / m))␣
 ↪- V_prime * t

    return x_vals, y_vals, z_vals
```

```
def plot_all_graphs(x_vals, y_vals, z_vals):
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(30, 5))

    center = (0, 0)
    radius = 0.2255

    draw_dartboard(ax1, center, radius)
    ax1.scatter(z_vals[-1], y_vals[-1], s=50, c='r', marker='o', label='Dart␣
 ↪arrival point')
    ax1.set_xlabel('z')
    ax1.set_ylabel('y')
    ax1.set_title('Dart Arrival Point on (y, z) plane')
    ax1.legend()
    ax1.grid()

    # Plot trajectory on (x, y) plane
    ax2.plot(x_vals, y_vals, label='Trajectory')
    ax2.scatter(x_vals[-1], y_vals[-1], label='Dart arrival point', color='red')
    ax2.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax2.set_ylabel('y')
    ax2.set_title('Dart Trajectory on (x, y) plane')
    ax2.axvline(2.37, color='black', linestyle='--', label='Location of the␣
 ↪Dart Board')
    ax2.legend()
    ax2.grid()

    # Plot trajectory on (x, z) plane
    ax3.plot(x_vals, z_vals, label='Trajectory')
    ax3.scatter(x_vals[-1], z_vals[-1], label='Dart arrival point', color='red')
    ax3.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax3.set_ylabel('z')
    ax3.set_title('Dart Trajectory on (x, z) plane')
    ax3.axvline(2.37, color='black', linestyle='--', label='Location of the␣
 ↪Dart Board')
    ax3.legend()
    ax3.grid()

    plt.show()

if __name__ == "__main__":
    try:
        alpha = float(input("Enter alpha angle (in degrees): "))
        beta = float(input("Enter beta angle (in degrees): "))
        V0 = float(input("Enter initial velocity (in m/s): "))
```

```
        difficulty = input("Enter difficulty level (beginner, intermediate,␣
 ↪advanced): ").lower()
        x_vals, y_vals, z_vals = calculate_trajectory(alpha, beta, V0,␣
 ↪difficulty)
        plot_all_graphs(x_vals, y_vals, z_vals)
    except ValueError as e:
        print(e)

print("The dart hits the target at (z, y) = ({:.2f}, {:.2f})".
 ↪format(z_vals[-1], y_vals[-1]))
```

```
Enter alpha angle (in degrees): 35
Enter beta angle (in degrees): 70
Enter initial velocity (in m/s): 6.5
Enter difficulty level (beginner, intermediate, advanced): beginner
```



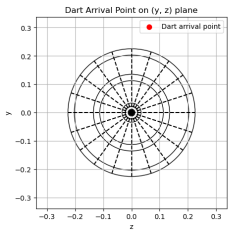

```
The dart hits the target at (z, y) = (0.13, 0.02)
```

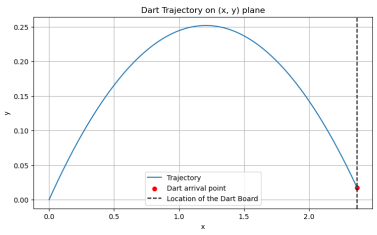
[36]:
```
# F = -kv^2

import numpy as np
import matplotlib.pyplot as plt


def calculate_trajectory(alpha, beta, V0, difficulty, x_target=2.37, m=0.025,␣
 ↪k=0.05):
    alpha = np.radians(alpha)
    beta = np.radians(beta)
    value_under_sqrt = 1 - np.cos(alpha)**2 - np.cos(beta)**2
    if value_under_sqrt < 0:
        raise ValueError("Invalid combination of alpha and beta angles.")

    ramda = np.arccos(np.sqrt(value_under_sqrt))

    if difficulty == "beginner":
        V_prime = 5
    elif difficulty == "intermediate":
        V_prime = 10
```

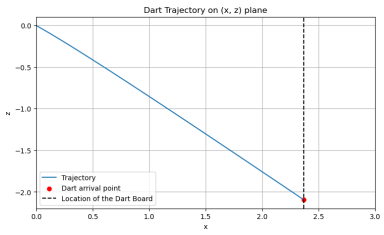
```
    elif difficulty == "advanced":
        V_prime = 15
    else:
        raise ValueError("Invalid difficulty level. Choose from 'beginner',␣
↪'intermediate', or 'advanced'.")

    # Calculate time of flight using t = R / V0cos(alpha)
    t_flight = x_target / (V0 * np.cos(alpha))

    # Calculate x, y, and z positions over time
    t = np.linspace(0, t_flight, num=1000)
    x_vals = V0 * np.cos(alpha) * t
    y_vals = V0 * np.cos(beta) * t - 0.5 * 9.81 * t**2
    z_vals = m / (k * (V0 * np.cos(ramda) + V_prime)) * np.log(k * t * (V0 * np.
↪cos(ramda) + V_prime) / m + 1) - V_prime * t

    return x_vals, y_vals, z_vals

def plot_all_graphs(x_vals, y_vals, z_vals):
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(30, 5))

    center = (0, 0)
    radius = 0.2255

    draw_dartboard(ax1, center, radius)
    ax1.scatter(z_vals[-1], y_vals[-1], s=50, c='r', marker='o', label='Dart␣
↪arrival point')
    ax1.set_xlabel('z')
    ax1.set_ylabel('y')
    ax1.set_title('Dart Arrival Point on (y, z) plane')
    ax1.legend()
    ax1.grid()

    # Plot trajectory on (x, y) plane
    ax2.plot(x_vals, y_vals, label='Trajectory')
    ax2.scatter(x_vals[-1], y_vals[-1], label='Dart arrival point', color='red')
    ax2.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax2.set_ylabel('y')
    ax2.set_title('Dart Trajectory on (x, y) plane')
    ax2.axvline(2.37, color='black', linestyle='--', label='Location of the␣
↪Dart Board')
    ax2.legend()
    ax2.grid()

    # Plot trajectory on (x, z) plane
    ax3.plot(x_vals, z_vals, label='Trajectory')
```

```
    ax3.scatter(x_vals[-1], z_vals[-1], label='Dart arrival point', color='red')
    ax3.set_xlabel('x')
    ax3.set_xlim(0, 3)
    ax3.set_ylabel('z')
    ax3.set_title('Dart Trajectory on (x, z) plane')
    ax3.axvline(2.37, color='black', linestyle='--', label='Location of the␣
 ↪Dart Board')
    ax3.legend()
    ax3.grid()

    plt.show()

if __name__ == "__main__":
    try:
        alpha = float(input("Enter alpha angle (in degrees): "))
        beta = float(input("Enter beta angle (in degrees): "))
        V0 = float(input("Enter initial velocity (in m/s): "))
        difficulty = input("Enter difficulty level (beginner, intermediate,␣
 ↪advanced): ").lower()
        x_vals, y_vals, z_vals = calculate_trajectory(alpha, beta, V0,␣
 ↪difficulty)
        plot_all_graphs(x_vals, y_vals, z_vals)
    except ValueError as e:
        print(e)

print("The dart hits the target at (z, y) = ({:.2f}, {:.2f})".
 ↪format(z_vals[-1], y_vals[-1]))
```

```
Enter alpha angle (in degrees): 35
Enter beta angle (in degrees): 70
Enter initial velocity (in m/s): 6.5
Enter difficulty level (beginner, intermediate, advanced): beginner
```








```
The dart hits the target at (z, y) = (-2.09, 0.02)
```

0.1

[ ]: