Author :

- Lenny
- Lokmane

0.1 Partie discrète (évènement de garde) + partie continue

```
def update_x_z(self):
        while True:
            if self.state == '0':
                self.x += 0.1
                self.z = self.z
                if self.x >= 10:
                    self.e1.succeed()

            if self.state == '1':
                self.x += 0.1
                self.z -= 0.1
                if self.z <= 10:
                  self.e2.succeed()

            if self.state == '2':
                self.z -=0.1
                self.x += 0.1
                if self.z <=0:
                     self.e3.succeed()

            if self.state == '3':
                self.z = 0
                self.x += 0.1



            self.x_data.append((self.env.now, self.x))
            self.z_data.append((self.env.now, self.z))
            yield self.env.timeout(0.1)
```

0.2 Partie discrète

```
    def start(self):
        while True:
            if self.state == '0':
                print(f'Time: [{self.env.now}] --> state[0] [x: {self.x:.2f} z: {self.z:.2f}]')
                yield self.e1
                self.state='1'

            elif self.state == '1':
                print(f'Time: [{self.env.now}] --> state[1] [x: {self.x:.2f} z: {self.z:.2f}]')
                yield self.e2
                self.state='2'



            elif self.state == '2':
                print(f'Time: [{self.env.now}] --> state[2] [x: {
                   self.x:.2f} z: {self.z:.2f}]')
                yield self.e3
                self.state='3'


            elif self.state == '3':
                print(f'Time: [{self.env.now}] --> state[3] [x: {
                   self.x:.2f} z: {self.z:.2f}]')
                yield self.env.timeout(0.1)
```

0.3 Résultat obtenue

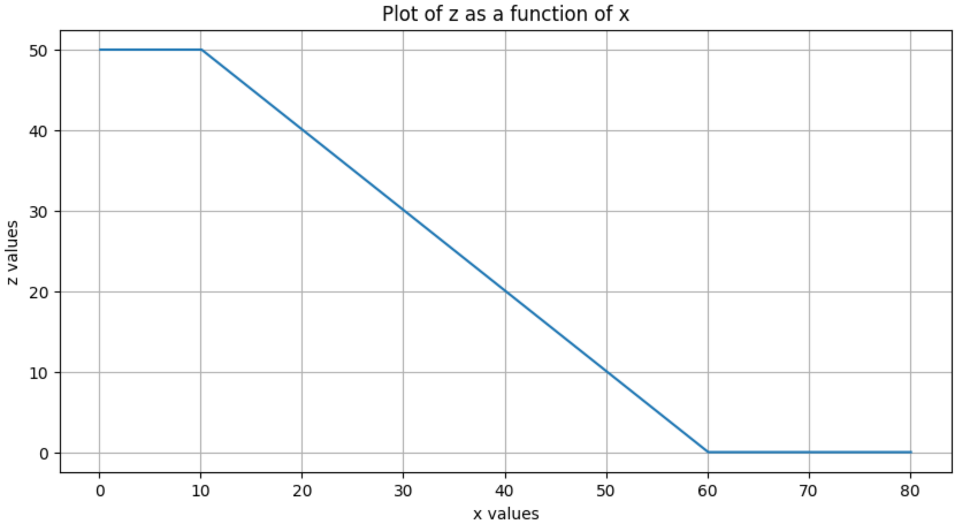

Figure 1 – Trajectoire d'aterrissage d'avion (avant contact)

0.4 Code source complet

```
    from simpy import *
import random
import matplotlib.pyplot as plt
import numpy as np


class MachineState:
    def __init__(self, env):

        #ENVIRONEMENT
        self.env = env
        #Condition initial
        self.state = '0'
        self.x = 0
        self.z = 50
        #LIST_DATA
        self.x_data = []
        self.z_data = []
        #EVENEMENT
        self.e1 = Event(env)
        self.e2 = Event(env)
        self.e3 = Event(env)

    def update_x_z(self):
        while True:
            if self.state == '0':
                self.x += 0.1
                self.z = self.z
                if self.x >= 10:
                    self.e1.succeed()

            if self.state == '1':
                self.x += 0.1
                self.z -= 0.1
                if self.z <= 10:
                  self.e2.succeed()

            if self.state == '2':
                self.z -=0.1
                self.x += 0.1
                if self.z <=0:
                     self.e3.succeed()

            if self.state == '3':
                self.z = 0
                self.x += 0.1



            self.x_data.append((self.env.now, self.x))
            self.z_data.append((self.env.now, self.z))
            yield self.env.timeout(0.1)

    def start(self):
        while True:
            if self.state == '0':
                print(f'Time: [{self.env.now}] --> state[0] [x: {
                   self.x:.2f} z: {self.z:.2f}]')
                yield self.e1
                self.state='1'
            elif self.state == ’1’:
                print(f’Time: [{self.env.now}] --> state[1] [x: {self.x:.2f} z: {self.z:.2f}]’)
                yield self.e2
                self.state=’2’



            elif self.state == ’2’:
                print(f’Time: [{self.env.now}] --> state[2] [x: {self.x:.2f} z: {self.z:.2f}]’)
                yield self.e3
                self.state=’3’


            elif self.state == ’3’:
                print(f’Time: [{self.env.now}] --> state[3] [x: {self.x:.2f} z: {self.z:.2f}]’)
                yield self.env.timeout(0.1)



def main():

    env = Environment()
    machine = MachineState(env)

    env.process(machine.start())
    env.process(machine.update_x_z())

    env.run(until=80)


    time_points = [point[0] for point in machine.x_data]
    x_values = [point[1] for point in machine.x_data]
    z_values = [point[1] for point in machine.z_data]


    plt.figure(figsize=(10, 5))
    plt.plot(x_values, z_values, marker=’’)
    plt.title(’Plot of z as a function of x’)
    plt.xlabel(’x values’)
    plt.ylabel(’z values’)
    plt.grid(True)
    plt.show()


if __name__ == "__main__":
    main()
```