Lab 3 report

StudentID 83459

**Loading the data**

damages = []

with open('damages.txt','r') as csvfile:

reader = csv.reader (csvfile, delimiter=";")

for row in reader:

damages.append(int(row[1]))

damages\_ln = sc.log(damages)

**K-Test**

test2 = kstest(damages\_ln, sc.stats.norm.cdf,

args = (sc.mean(damages\_ln), sc.std(damages\_ln)))

if test2[1] > 0.05:

print ("p-value: ", round(test2[1], 4),

"- fail to reject the null hypothesis")

else:

print("null hypothesis should be rejected")

p-value: 0.9979 - fail to reject the null hypothesis

**simulation parameters**

damages\_ln\_avg = sc.mean(damages\_ln)

damages\_ln\_std = sc.std(damages\_ln)

number\_of\_policies = 1500

number\_of\_damages\_avg

time\_horizon = 2

balance = 10000

insurance\_premium = 500

**Simulation model**

def model (number\_of\_policies, number\_of\_damages\_avg,

damages\_ln\_avg, damages\_ln\_std, time\_horizon,

balance, insurance\_premium, seed):

# Common Random Numbers:

# sc.random.seed(seed)

# contracts signing calendar:

dates = [sc.random.randint(0, 364) for i in range(number\_of\_policies)]

calendar\_contracts = [0]\*365

for day in dates:

calendar\_contracts[day] += 1

# number of damages per one policy:

number\_of\_damages = sc.random.poisson(number\_of\_damages\_avg,

number\_of\_policies)

# compensations calendar:

calendar\_compensations = [0]\*(365\*time\_horizon)

for k in range(number\_of\_policies):

for s in range(number\_of\_damages[k]):

compensation\_date = dates[k] + sc.random.randint(0, 364)

calendar\_compensations[compensation\_date] += 1

# time\_horizon simulation:

for day in range(365\*time\_horizon):

if day <= 364:

balance += calendar\_contracts[day] \* insurance\_premium

number\_of\_compensations = calendar\_compensations[day]

compensations = 0

if number\_of\_compensations > 0:

compensations = sum(sc.exp(sc.random.normal(damages\_ln\_avg,

damages\_ln\_std,

number\_of\_compensations)))

if balance < compensations:

return balance - compensations

else:

pass

balance -= compensations

return balance

**run function**

def run(balance, insurance\_premium, repeats,

number\_of\_policies, number\_of\_damages\_avg ,

damages\_ln\_avg, damages\_ln\_std, time\_horizon):

results = []

bankruptcy = 0

results\_positve = []

for seed in range(repeats):

results.append(model(number\_of\_policies, number\_of\_damages\_avg ,

damages\_ln\_avg, damages\_ln\_std, time\_horizon,

balance, insurance\_premium, seed))

if results[seed] < 0:

bankruptcy += 1

if results[seed] > 0:

results\_positve.append(results[seed])

results\_average = sc.mean(results\_positve)

bankruptcy\_risk = bankruptcy / repeats

return [bankruptcy, bankruptcy\_risk, results\_average]

**Simulation results**

# model\_parameters

results = [] # średni wynik finansowy firmy

insurance\_cost = []

bankruptcy\_risk = []

bankruptcy\_cnt = []

repeats = 100

number\_of\_policies = 1500

time\_horizon = 2 # 2 years

for balance in range(10000, 50000,10000):

for insurance\_premium in range(500,1000,100):

run\_result = run(balance, insurance\_premium, repeats,

number\_of\_policies, number\_of\_damages\_avg,

damages\_ln\_avg, damages\_ln\_std, time\_horizon)

insurance\_cost.append(insurance\_premium)

bankruptcy\_cnt.append(run\_result[0])

bankruptcy\_risk.append(run\_result[1])

results.append(run\_result[2])

print("BALANCE: ", balance, "INSURANCE PREMIUM: ", insurance\_premium,

"BANKRUPTCY CNT: ", run\_result[0], "AVERAGE RESULT: ", round(run\_result[2]),

"BANKRUPTCY RISK: ", run\_result[1])

plt.plot(insurance\_cost, bankruptcy\_risk)

plt.ylabel('BANKRUPTCY RISK')

plt.show()

A close up of a logo

Description automatically generated

**Conclusion:**

After running simulation with 100 repeats and 1500 policies we can clearly see that the there are few scenarios of the lowest bankruptcy risk:

\* bankruptcy risk = 0.01: balance - 10000, insurance premium - 800/900

\* bankruptcy risk = 0.01/0.02: balance - 20000, insurance premium - 800/900

\* bankruptcy risk = 0.01: balance - 30000, insurance premium - 900

\* bankruptcy risk = 0.01: balance - 40000, insurance premium - 900

Interestingly, that all balances with insurance premium 500/600 give bankruptcy risk > 0.9