Joao Quinta

TP 9

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.special import comb
```

Option pricing: Black-Scholes versus Binomial Tree

.1

In [2]:
```
def black_scholes(S, K, T, r, sigma):
    # Calculate d1
    d1 = (np.log(S / K) + (r + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
    # Calculate d2
    d2 = (np.log(S / K) + (r - sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
    # Compute the price of the European call option using Black-Scholes formula
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price
```

In [3]:
```
s_0 = 100
T = 1
risk = 0.05
sigma = 0.2
K = 120

# Calculate the value of a European call option using the Black-Scholes function
val_bs = black_scholes(s_0, K, T, risk, sigma)
# Print the calculated option value
print("Black-Scholes: ", val_bs)
```

```
Black-Scholes:  3.247477416560816
```

.2

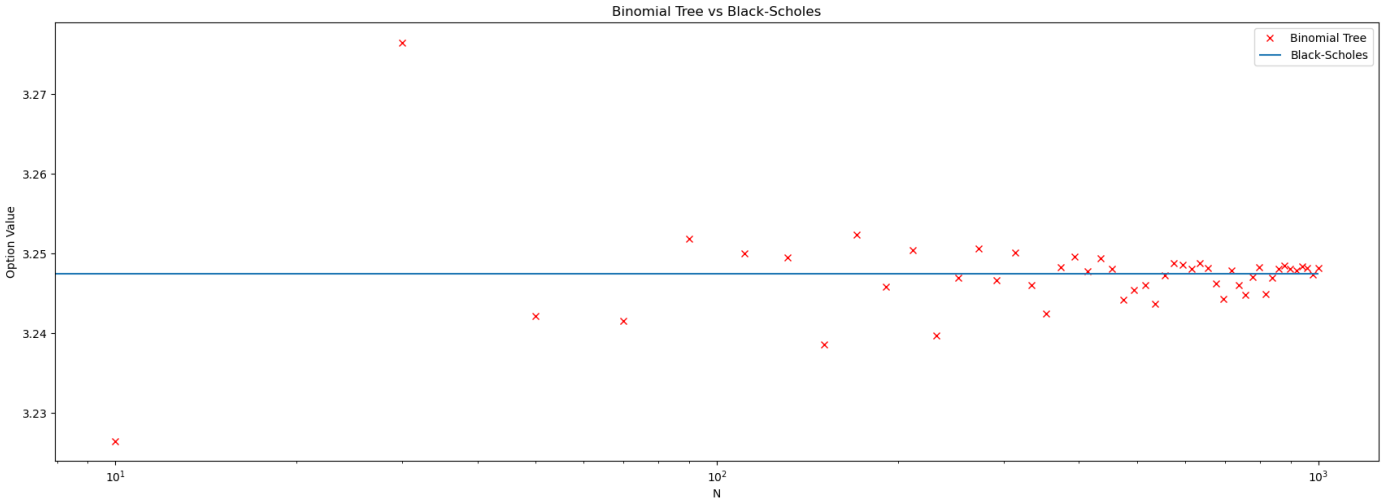
In [4]:
```
def bin_tree_inOnce(ini_price, sigma, risk, T, N, K):
    # Calculate the up (u) and down (d) factors
    u = np.exp(sigma * np.sqrt(T / N))
    d = 1 / u
    # Calculate probabilities of up (p_u) and down (p_d) movements
    p_u = (np.exp(risk * T / N) - d) / (u - d)
    p_d = 1 - p_u
    # Initialize the option value
    option = 0
    # Loop through all possible paths and calculate the option value using the binomial
    for nu in range(N):
        # Calculate the probability of each path
        tmp = comb(N, nu) * p_u**nu * p_d**(N - nu)
        # Calculate the payoff of each path and multiply by its probability
        tmp *= np.max([0, ini_price * u**nu * d**(N - nu) - K])
        # Accumulate the option value over all possible paths
        option +=  tmp
    # Discount the option value back to the present value
    return option * np.exp(-risk * T)
```

.3

In [5]:
```
s_0 = 100
T = 1
risk = 0.05
sigma = 0.2
K = 120

# Generate an array of numbers of steps for the binomial tree model
all_n = np.linspace(10, 1000, 50).astype(int)
# Initialize an empty list to store the option values
res_bin = []
# Loop through all numbers of steps in the binomial tree model
for n in all_n:
    # Calculate the option value using the bin_tree_inOnce function and append it to the
    res_bin.append(bin_tree_inOnce(s_0, sigma, risk, T, n, K))
```

In [6]:
```
plt.figure(figsize=(21,7))
plt.semilogx(all_n, res_bin, "rx", label="Binomial Tree")
plt.hlines(val_bs, 0, max(all_n), linestyles="-", label="Black-Scholes")
plt.legend()
plt.xlabel("N")
plt.ylabel("Option Value")
plt.title("Binomial Tree vs Black-Scholes")
plt.show()
```



In [ ]: