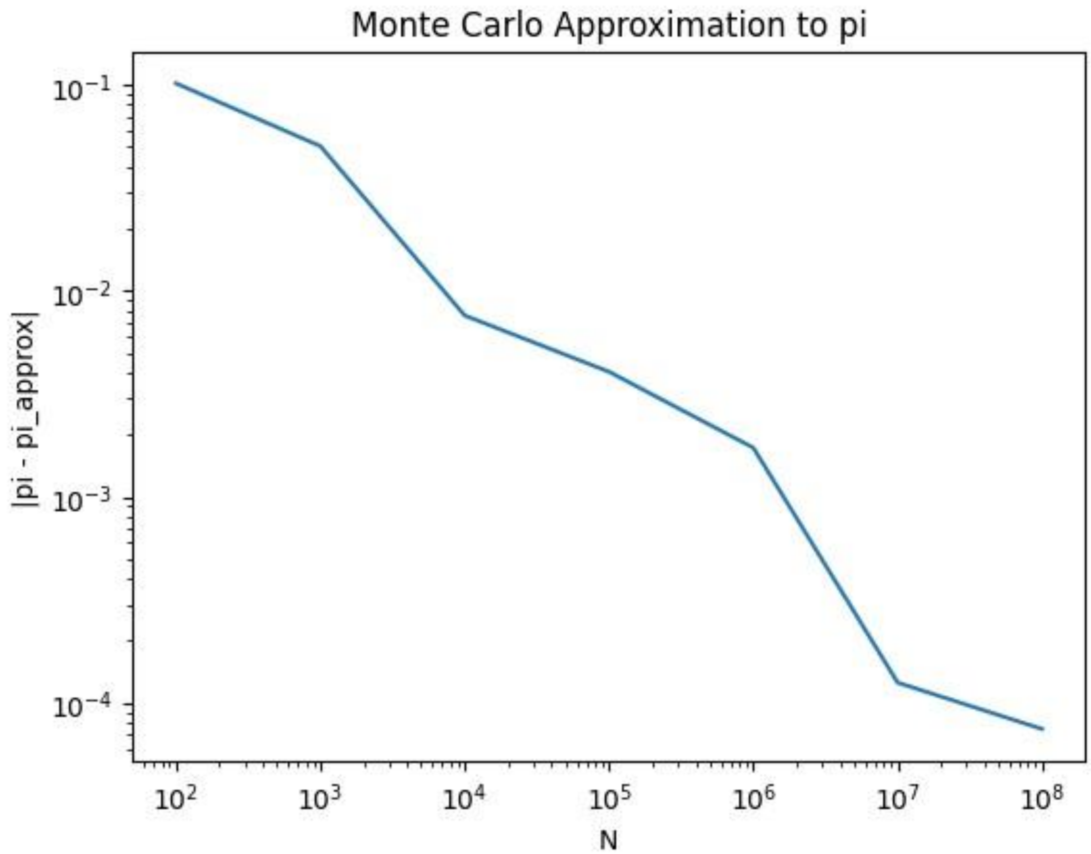
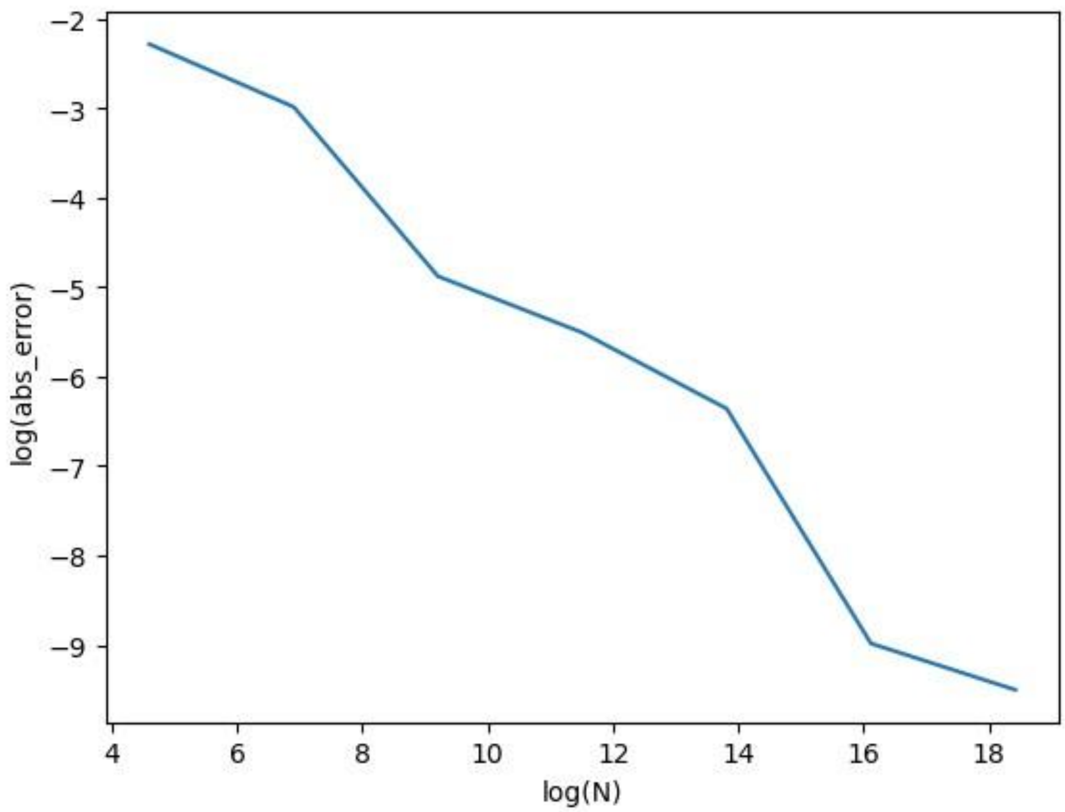
Report

November 8, 2023

[39]:
```
import numpy as np
import matplotlib.pyplot as plt
# Load the data
N = []
pi_approx = []
with open("pi_results.dat") as f:
  for line in f:
   N_, pi_approx_ = line.split(',')
   N.append(int(N_))
   pi_approx.append(float(pi_approx_))
# Calculate the absolute error
abs_error = [abs(np.pi - pi) for pi in pi_approx]
# Plot the data on a log-log plot
plt.loglog(N, abs_error)
plt.xlabel("N")
plt.ylabel("|pi - pi_approx|")
plt.title("Monte Carlo Approximation to pi ")
plt.show()
```



[40]:
```
x = np.log(N)
y = np.log(abs_error)
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_xlabel('log(N)')
ax.set_ylabel('log(abs_error)')
plt.show()
```

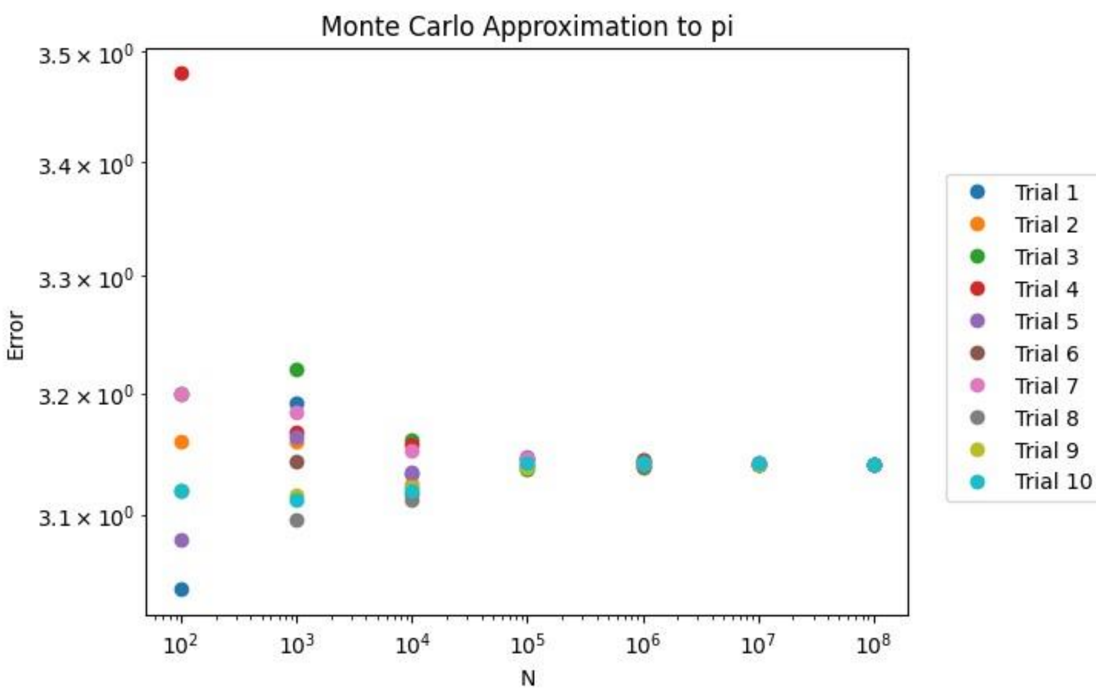


[16]:

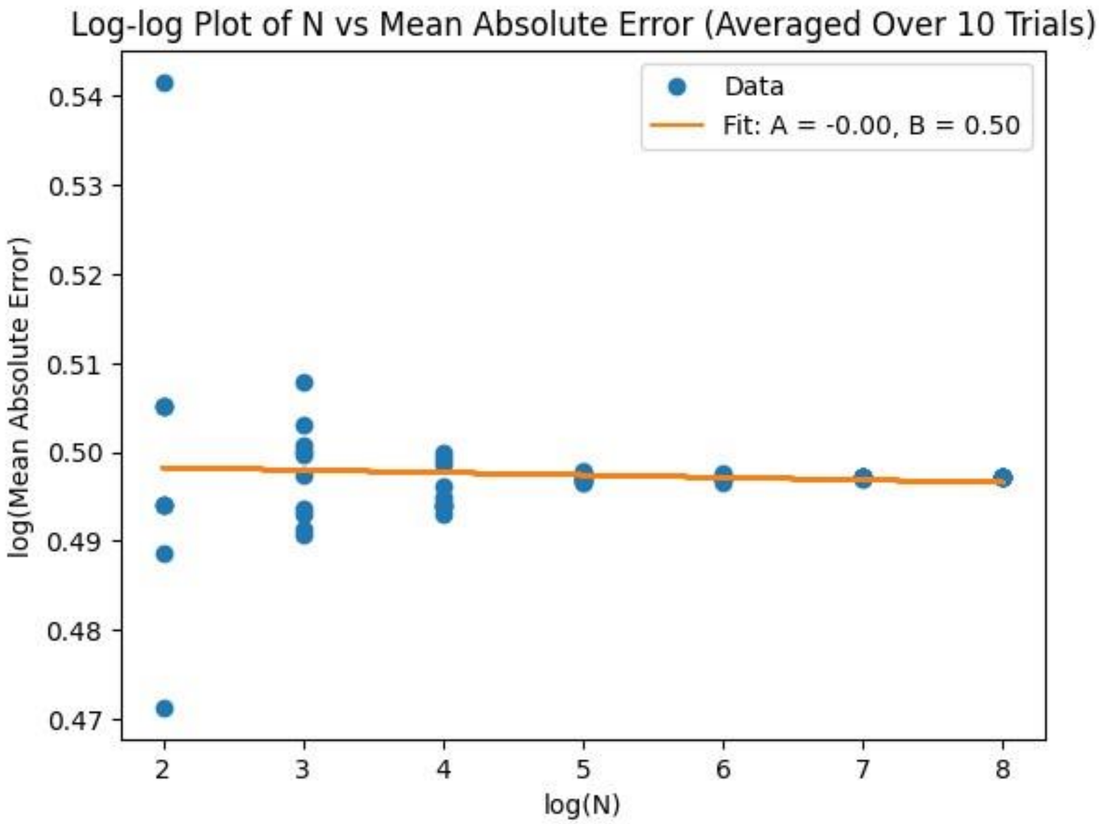
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Load data for the first trial
N, err = np.loadtxt("pi_results.dat", delimiter=',', unpack=True)
plt.loglog(N, err, marker="o", lw=0, label='Trial 1')

num_trials = 10

# Store data for all trials
all_N = []
all_err = []

for j in range(1, num_trials):
    N, err = np.loadtxt(f"output_{j}.dat", delimiter=',', unpack=True)
    plt.loglog(N, err, marker="o", lw=0, label=f'Trial {j + 1}')

    # Store N and err for linear regression
    all_N.extend(N)
    all_err.extend(err)
```

```
plt.legend(loc=(1.05, 0.2))
plt.xlabel("N")
plt.ylabel("Error")
plt.title("Monte Carlo Approximation to pi ")

# Convert data to NumPy arrays
all_N = np.array(all_N).reshape(-1, 1)
all_err = np.array(all_err)

# Perform linear regression
model = LinearRegression()
model.fit(np.log(all_N), np.log(all_err))

# Print the slope (exponent in the power-law relationship)
slope = model.coef_[0]
print("Slope (Exponent in the Power-Law Relationship): ", slope)

# Show the plot
plt.show()
```

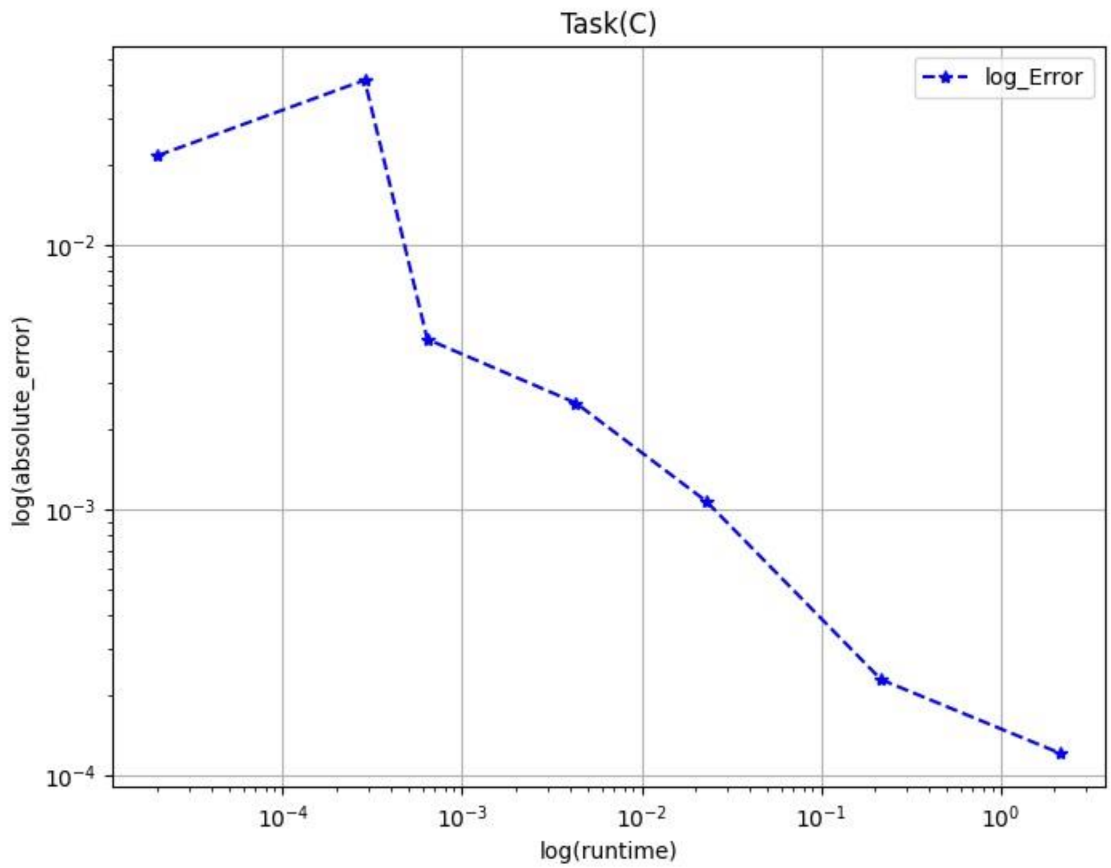
```
Slope (Exponent in the Power-Law Relationship): -
0.0006910254873322644
```



[46]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Load data from the combined_data.dat file
filename = 'combined_data.dat'

# Adjust the delimiter and skiprows based on your data file
data = np.loadtxt(filename, delimiter=',', skiprows=1)

# Extract N and error columns
N_values = data[:, 0]
error_values = data[:, 1:]

# Calculate the mean error for each N across the 10 trials
mean_error_values = np.mean(error_values, axis=1)

# Convert to NumPy arrays
N_values = np.array(N_values)
mean_error_values = np.array(mean_error_values)

# Calculate log(N) and log(mean_error)
log_N_values = np.log10(N_values)
log_mean_error_values = np.log10(mean_error_values)

# Fit a line to the data using linear regression
slope, intercept = np.polyfit(log_N_values, log_mean_error_values, 1)

# Plot log-log data
plt.plot(log_N_values, log_mean_error_values, marker='o', linestyle='',␣
 ↪label='Data')
plt.plot(log_N_values, slope * log_N_values + intercept, label=f'Fit: A =␣
 ↪{slope:.2f}, B = {intercept:.2f}')

plt.xlabel('log(N)')
plt.ylabel('log(Mean Absolute Error)')
plt.title('Log-log Plot of N vs Mean Absolute Error (Averaged Over 10 Trials)  ')
plt.legend()
plt.show()
```



[34]:
```python
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt('time.dat')
N = data[:, 0]
error = data[:, 1]
time = data[:, 2]

plt.figure(figsize=(8, 6))
plt.loglog(time, error, '--b*')
plt.xlabel('log(runtime)')
plt.ylabel('log(absolute_error)')
plt.title('Task(C)')
plt.legend(['log_Error'])
plt.grid(True)
plt.show()
```



[35]:
```
import numpy as np
data = np.loadtxt('time.dat')
N = data[:, 0]
error = data[:, 1]
time = data[:,2]
log_time = np.log10(time)
log_error = np.log10(error)
slope, intercept = np.polyfit(log_time, log_error, 1)
print(f'Slope: {slope:.2f}')
print(f'Intercept: {intercept:.2f}')

e=[-16, -70030]
for i in e:
 time=(slope*i)+intercept #using linear equation i.e. slope intercept form
 print("Approximate runtime for 10^%f is 10^%f" %(i,time))
```

```
Slope: -0.52
Intercept: -3.81
Approximate runtime for 10^-16.000000 is 10^4.459656
```

```
Approximate runtime for 10^-70030.000000 is 10^36199.907899
```

[ ]:

```

```