Homework2

October 31, 2022

[32]:
```
# Load required packages
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
import pandas as pd
import random
import sklearn as sk
from sklearn.linear_model import LinearRegression
```

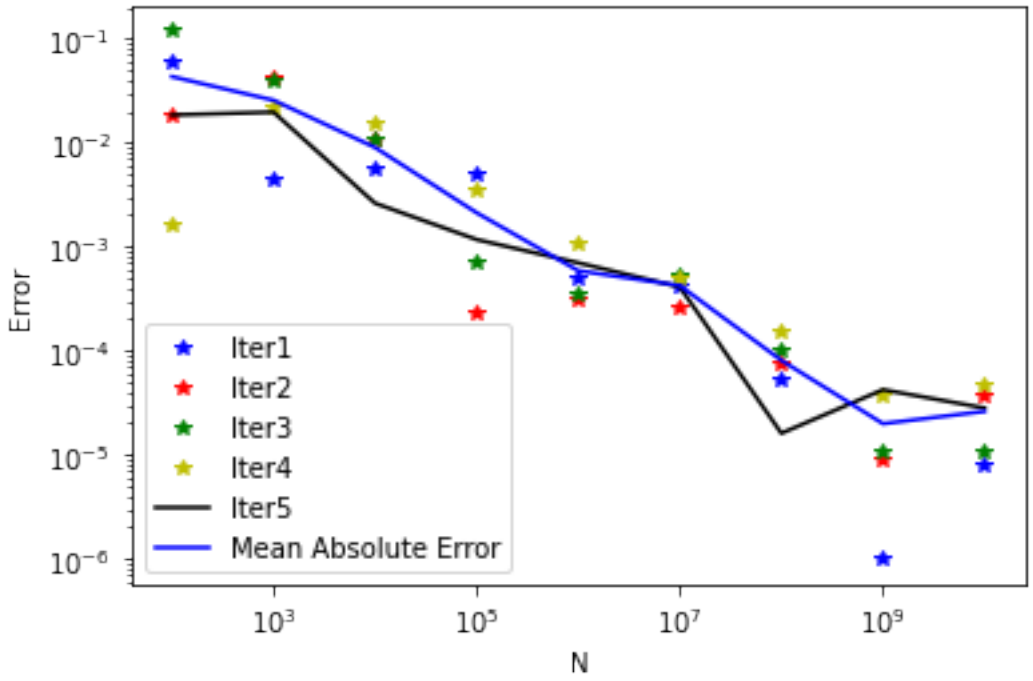
[33]:
```
#Read in the .dat files
montecarlo1 = np.loadtxt("mc2_1.dat")
montecarlo2 = np.loadtxt("mc2_2.dat")
montecarlo3 = np.loadtxt("mc2_3.dat")
montecarlo4 = np.loadtxt("mc2_4.dat")
montecarlo5 = np.loadtxt("mc2_5.dat")
```

[34]:
```
# Create two atomic vectors/variables to store the dat file contents
x = montecarlo1[:,0]
y1 = montecarlo1[:,2]
y2 = montecarlo2[:,2]
y3 = montecarlo3[:,2]
y4 = montecarlo4[:,2]
y5 = montecarlo5[:,2]
```

[35]:
```
# Smoothing out the Monte Carlo error
y_add = y1 + y2 + y3 + y4 + y5
y_mean = y_add/5 #Divide it by the number of iterations and not the number of␣
 ↪trials
```

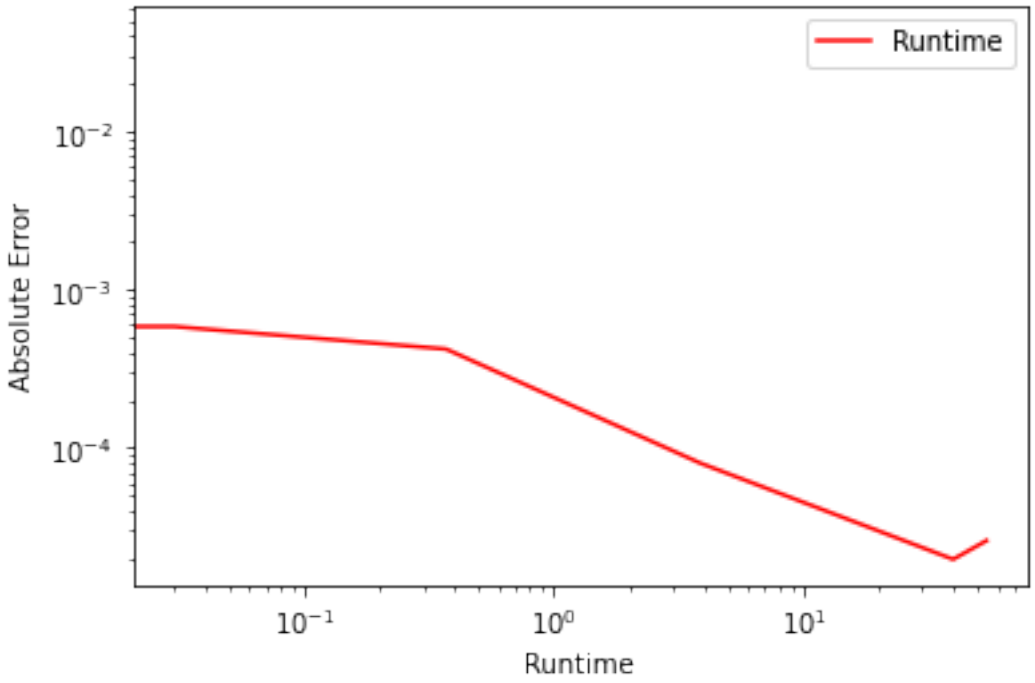
[36]:
```
# Create a log-log plot
plt.loglog(x, y1,'b*',label='Iter1')
plt.loglog(x, y2,'r*',label='Iter2')
plt.loglog(x, y3,'g*',label='Iter3')
plt.loglog(x, y4,'y*',label='Iter4')
plt.loglog(x, y5,'black',label='Iter5')
plt.loglog(x, y_mean,'b',label='Mean Absolute Error')
```

```
plt.legend(loc=3)
plt.xlabel("N")
plt.ylabel("Error")
plt.savefig("montecarlo_plot.png")
```



[37]:
```
runtime = [0.00, 0.00, 0.00, 0.00, 0.03, 0.37, 3.85, 39.87, 54.25]
runtime_array = np.asarray(runtime)
#type(runtime)
#type(runtime_array)
#print (runtime_array.dtype)
error_mean = y_mean
```

[38]:
```
plt.loglog(runtime_array,error_mean,'r',label='Runtime')
plt.legend(loc=0)
plt.xlabel("Runtime")
plt.ylabel("Absolute Error")
plt.savefig("Runtime vs. Absolute Error.png")
```



[39]:
```
# plt.loglog(error_mean,runtime_array,'r',label='Runtime')
# plt.legend(loc=0)
# plt.xlabel("Runtime")
# # plt.ylabel("Absolute Error")
# plt.savefig("Runtime vs. Absolute Error.png")
```

From the above graph we see that we reach an accuracy of 10 raised to the power minus three staggeringly fast. However, as the Absoulute Error keeps getting smaller, the number of trials keeps increasing which leads to a corresponding increase in the runtime as well.

We'll use a Linear Regression to estimate the runtime for differing levels of desired accuracy. The model to estimate the runtime can be give by the following equation -

R = I + B*E + residual error - (1)

where, R = runtime I = intercept B = weight/leverage/coefficient value E = Absolute Error

[40]:
```
random.seed(123)
model = LinearRegression().fit(error_mean.
  ↪reshape(len(error_mean),1),runtime_array.reshape(len(runtime_array),1))
```

[41]:
```
model.coef_
```

[41]:
```
array([[-472.08758366]])
```

[42]:
```
model.intercept_
```

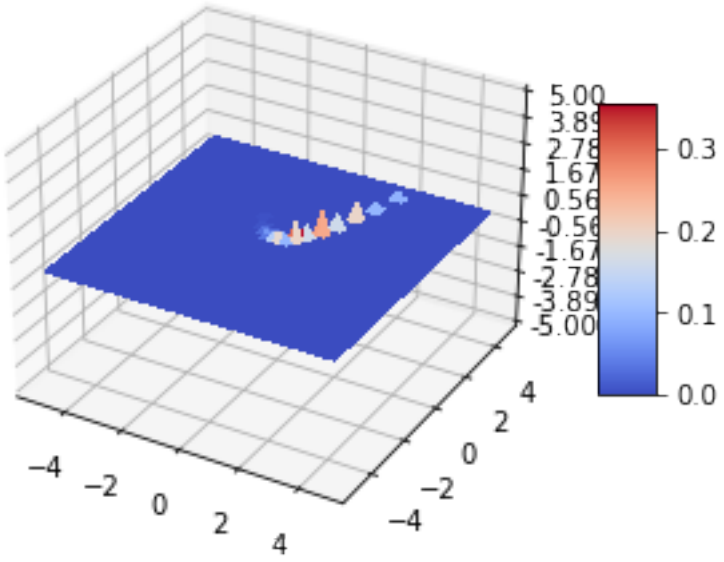
[42]:
```
array([15.16696508])
```

Plugging the coefficient and the intercept values in (1) we get the following relation -

R = 15.16696508 + (-472.08758366)*E + residual error -(2)

Substituting the values pow(10,-16) and pow(10,-70030) for E in (2), I estimate that the program would need to run for 2hours and 13 hours respectively.

[43]:
```
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Make data.
X1 = np.arange(-5, 5, 0.25)
X2 = np.arange(-5, 5, 0.25)
X1, X2 = np.meshgrid(X1, X2)
Z = np.exp(-pow((1-X1),2) - 100 * pow((X2 - pow(X1,2)),2))

# Plot the surface.
surf = ax.plot_surface(X1, X2,Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
#ax.set_zlim(X1.all(), X2.all())
ax.set_zlim(-5,5)
ax.zaxis.set_major_locator(LinearLocator(10))
# A StrMethodFormatter is used automatically
ax.zaxis.set_major_formatter('{x:.2f}')

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```



For Q2(e), taking cognizance of my laptop specs - 4 cores with 1.38GHz base speed the calculation of 40 different Bayes Factors can take 'n' seconds. However, assuming that Stampede2 allocates me 'n' cores then the same calculation on Stampede2 will take (1/n) time given that all sub-problems are pre-defined and the computation of the sub-problems is independent.

For Q2(f), we see that the value of Z1 $<$ Z2 over many trials. Hence, model 2 fits the data better and should be chosen.