CS F372: Operating Systems Assignment

Sem 1 AY 2021-22

November 15, 2021

1 Contributors

Group No. 31

Name	ID
Nandan H R	2019A7PS0164H
Jeevan Jyot Singh	2019A7PS0172H
Ankita Behera	2019A7PS0075H
Anuradha Pandey	2019A7PS0265H
Omkar Pitale	2019A7PS0083H
Aditya Chopra	2019A7PS0178H
Durba Satpathi	2019A7PS0972H
Hriday G	2019A7PS1212H

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("darkgrid")
```

[2]:
```
df = pd.read_csv("formatted_stats.csv")
```

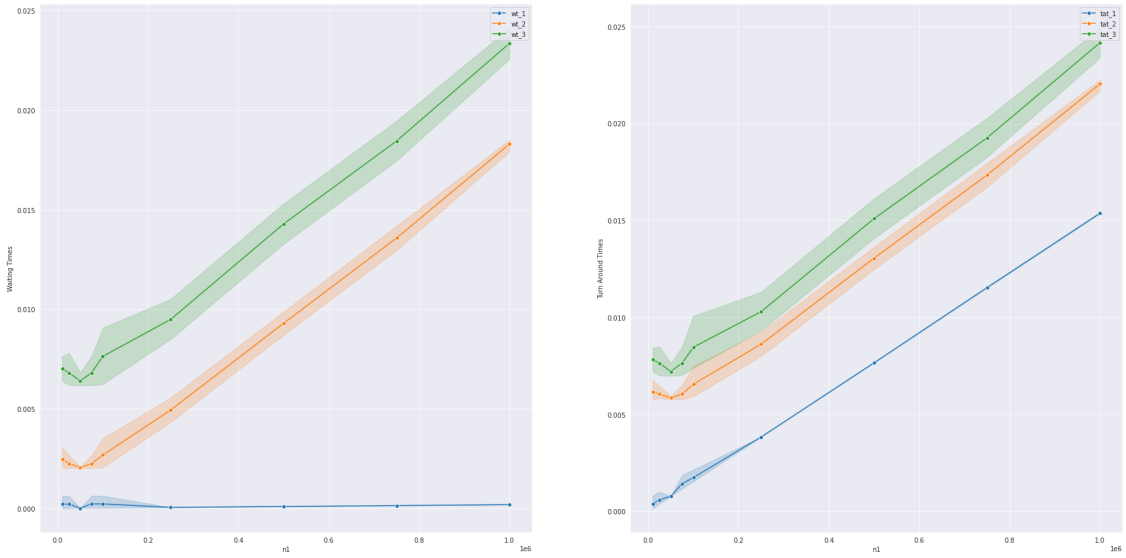
2 FCFS Performance Analysis

2.1 Waiting Times and Turn Around Times with Nx varying, and Ny Nz constant

[3]:
```
Ny = 2.5e4
Nz = 1.0e4

fig, axs = plt.subplots(1, 2, figsize=(30, 15))
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n2"] == Ny) & (df["n3"] == Nz) & (df["scheduling_algorithm"] == "fcfs")
]

axs[0].set_ylabel('Waiting Times')
```

```
sns.lineplot(data=filtered, x="n1", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n1", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n1", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n1", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n1", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n1", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[3]: <AxesSubplot:xlabel='n1', ylabel='Turn Around Times'>



2.2 Waiting Time Analysis

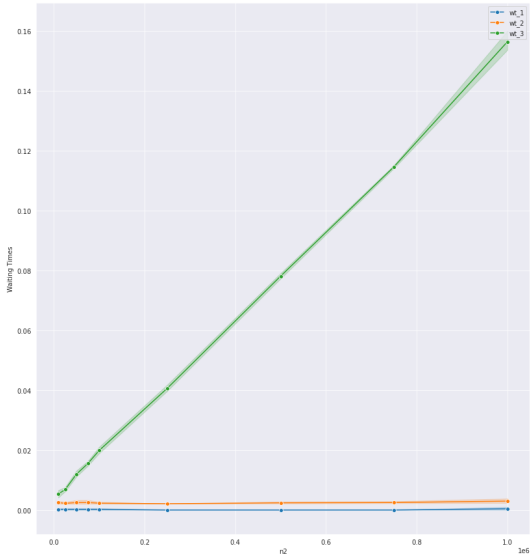
- The WT of C1 remains close to 0 for all values of **n1** when **n2** and **n3** are constant. This can be explained by the fact that in FCFS, C1 always executes first. Also, the WT of C2 and C3 increases almost linearly with **n1**.
- C2 executes after C1 terminates and hence as the wWorkload of C1 increases so does its Execution Time, as a result of which the WT of C2 increases.
- The same can be said for C3.

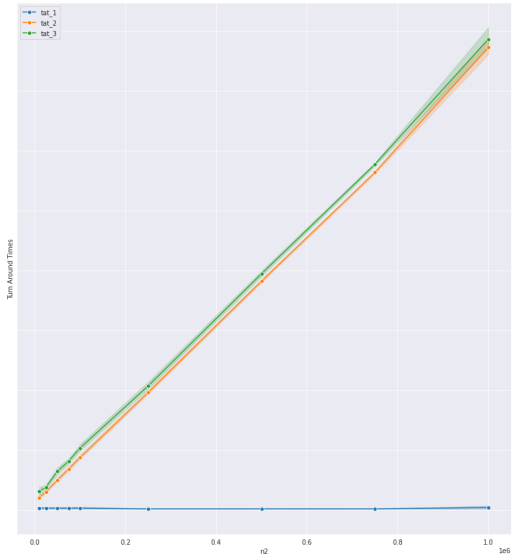
2.3 Turn Around Time Analysis

- As the Workload of C1 increases, its Execution Time increases. As a result, the TAT of C1 follows a linear trend and increases with **n1**.
- As is the case with WT, the TAT of C2 increases linearly as the Execution Time of C1 increases with **n1**.
- Similarly, the TAT of C3 increases linearly with **n1**.

[4]:
```
Ny = 2.5e4
Nz = 1.0e4

fig, axs = plt.subplots(1, 2, figsize=(30, 15),)
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n1"] == Ny) & (df["n3"] == Nz) & (df["scheduling_algorithm"] == "fcfs")
]

axs[0].set_ylabel('Waiting Times')
sns.lineplot(data=filtered, x="n2", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n2", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n2", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n2", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n2", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n2", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[4]:
```
<AxesSubplot:xlabel='n2', ylabel='Turn Around Times'>
```





2.4 Waiting Time Analysis

- The WT of C1 remains close to 0 for all values of **n2** when **n1** and **n3** are constant. This can be explained by the fact that in FCFS, C1 always executes first.
- Similarly, the WT for C2 remains constant with **n2** as in FCFS, C2 executes right after C1.
- The WT of C3 increases almost linearly with **n2**. C3 executes after C2 terminates and hence as the Workload of C2 increases so does its Execution Time, as a result of which the WT of C3 increases.

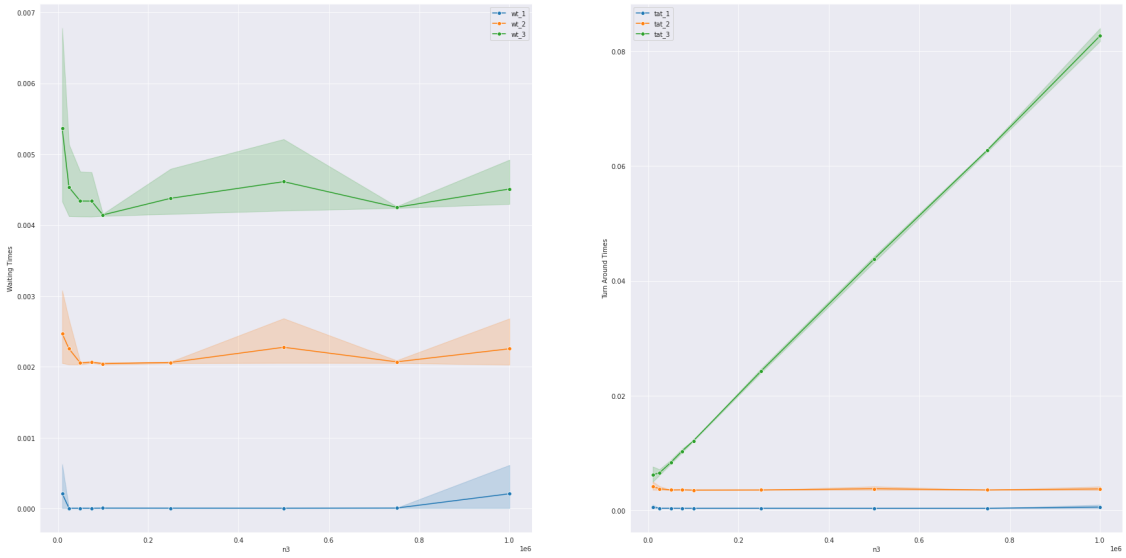
2.5 Turn Around Time Analysis

- Increasing the Workload of C2 has no effect on the TAT of C1 as C1 always executes before C2. Hence the TAT of C1 remains constant with increase in **n2**.
- As the Workload of C2 increases, so does its Execution Time, as a result of which its TAT increases almost linearly with **n2**.
- Similarly, as the Execution Time of C2 increases with its Workload, the WT of C3 increases and so does its TAT.

[5]:
```
Ny = 2.5e4
Nz = 1.0e4

fig, axs = plt.subplots(1, 2, figsize=(30, 15))
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n1"] == Ny) & (df["n2"] == Nz) & (df["scheduling_algorithm"] == "fcfs")
]

axs[0].set_ylabel('Waiting Times')
```

```
sns.lineplot(data=filtered, x="n3", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n3", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n3", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n3", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n3", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n3", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[5]: <AxesSubplot:xlabel='n3', ylabel='Turn Around Times'>



2.6 Waiting Time Analysis

- The WT of C1 remains close to 0 for all values of **n3** when **n1** and **n2** are constant. In FCFS, C1 always executes first and Workload of C3 has no effect on its WT.
- The WT for C2 remains constant with **n3** as in FCFS, C2 executes before C3 and Workload of C3 has no effect on its WT.
- The WT of C3 too remains constant as its Workload has no effect on its WT.

2.7 Turn Around Time Analysis

- Increasing the Workload of C3 has no effect on the TAT of C1 or C2 as they are executed before C3, hence the constant plots.

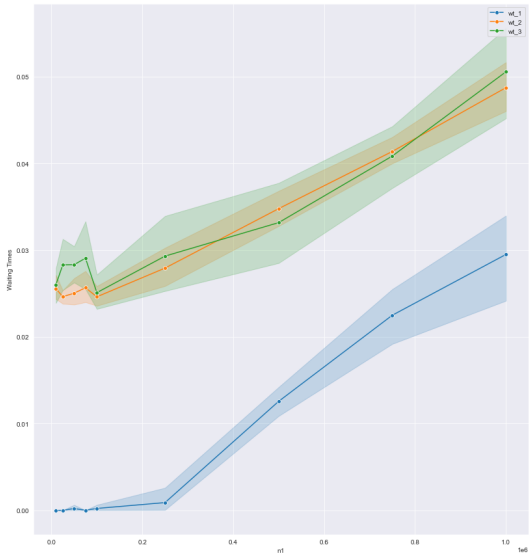
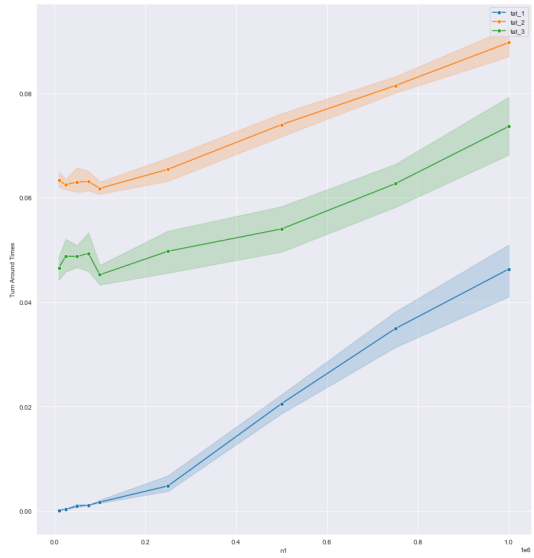
- As the Workload of C3 increases, its Execution Time increases. As a result, its WT increases linearly with increase in **n3**.

3 RR Performance Analysis

3.1 Waiting Times and Turn Around Times with Nx varying, and Ny Nz Time-Quantum constant

[6]:
```
Ny = 2.5e5
Nz = 2.5e5
tq = 500

fig, axs = plt.subplots(1, 2, figsize=(30, 15))
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n2"] == Ny) & (df["n3"] == Nz) & (df["scheduling_algorithm"] == "rr") &␣
 ↪(df['time_quantum'] == tq)
]

axs[0].set_ylabel('Waiting Times')
sns.lineplot(data=filtered, x="n1", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n1", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n1", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n1", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n1", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n1", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[6]:
```
<AxesSubplot:xlabel='n1', ylabel='Turn Around Times'>
```

3.2 Waiting Time Analysis

- For smaller workloads, the WT of C1 remains almost constant as it executes for fewer time quantums. As the Workload of C1 increases, the number of time quantums it executes for increases, adding to its WT in RR Scheduling.
- WT of C2 and C3 for smaller values of **n1** remains close to constant as C1 finishes execution in a fewer time quantums as compared to C2 and C3. The WT increases linearly for larger values of **n1** as their WT in RR Scheduling increases the more number of time quantums C1 executes for.

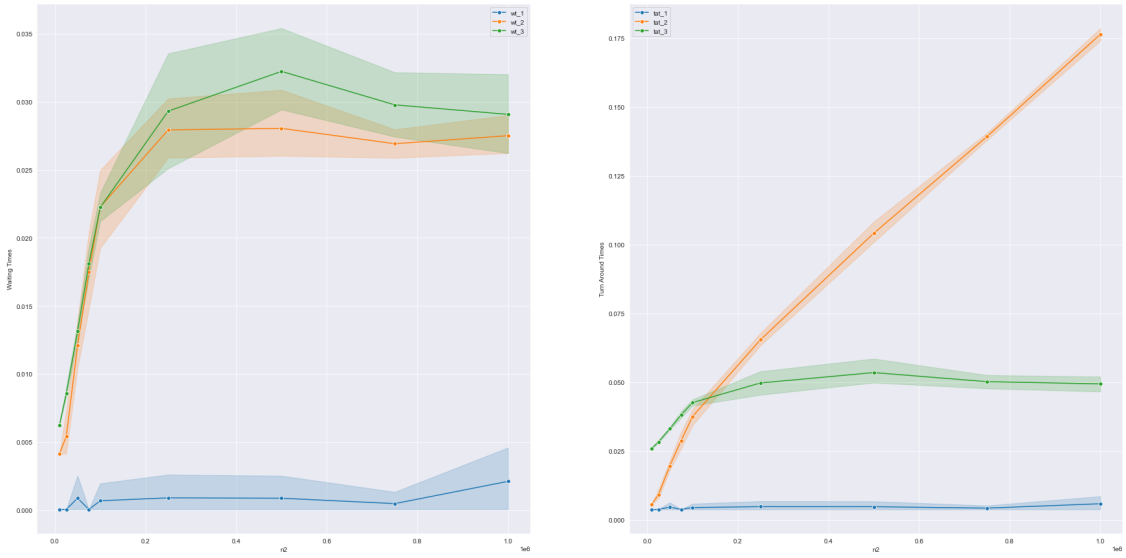
3.3 Turn Around Time Analysis

- For smaller values of **n1**, TAT of C1 slowly increases linearly as it executes for fewer time quantums. As the Workload of C1 increases, its WT also increases and TAT increases steeply.
- For C2 and C3, the TAT is almost constant for smaller values of **n1** and increases linearly for larger values. TAT of C2 is the highest as it is the costliest.

[7]:
```
Ny = 2.5e5
Nz = 2.5e5
tq = 500

fig, axs = plt.subplots(1, 2, figsize=(30, 15))
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n1"] == Ny) & (df["n3"] == Nz) & (df["scheduling_algorithm"] == "rr") &␣
 ↪(df['time_quantum'] == tq)
]
```

```
axs[0].set_ylabel('Waiting Times')
sns.lineplot(data=filtered, x="n2", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n2", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n2", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n2", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n2", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n2", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[7]: <AxesSubplot:xlabel='n2', ylabel='Turn Around Times'>



3.4 Waiting Time Analysis

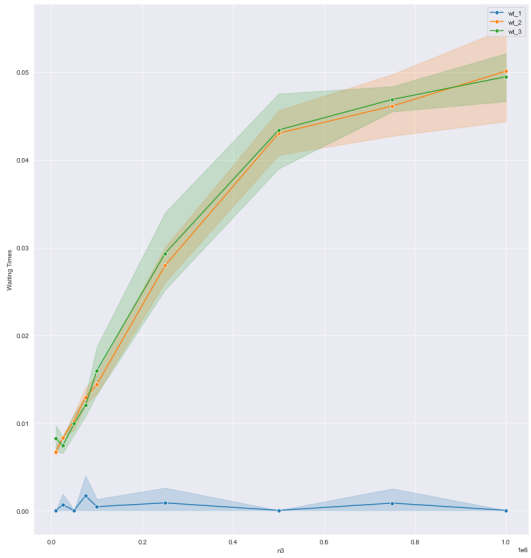
- The WT of C1 remains almost constant as it executes for fewer time quantums.
- C2 being the costliest process, a small increase in its Workload results in a large increase in WT as it requires more time quantums to execute, hence the steep curve.
- The increase in the number of time quantums required by C2 also contributes to the steep increase in WT of C3. Once either of them finishes execution, their is no additional WT for the remaining process. Hence the WT will be nearly the same for C2 and C3.

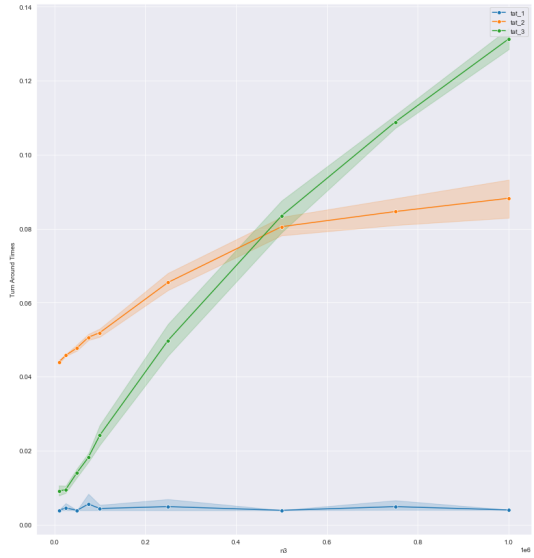
3.5 Turn Around Time Analysis

- The TAT of C1 remains almost constant as it executes for fewer time quantums.
- For smaller values of **n2**, the TAT of C2 increases steeply then increases linearly with Workload as C1 and C3 complete execution before C2.
- C3 increases linearly because it runs with C2 and for larger values of **n2** it remains close to constant as it finishes execution before C2.

[8]:
```
Ny = 2.5e5
Nz = 2.5e5
tq = 500

fig, axs = plt.subplots(1, 2, figsize=(30, 15))
# fig.suptitle('Times vs N1 Plots')

filtered = df.loc[
    (df["n1"] == Ny) & (df["n2"] == Nz) & (df["scheduling_algorithm"] == "rr") &␣
 ↪(df['time_quantum'] == tq)
]

axs[0].set_ylabel('Waiting Times')
sns.lineplot(data=filtered, x="n3", y="wt_1", marker='o', label="wt_1",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n3", y="wt_2", marker='o', label="wt_2",␣
 ↪ax=axs[0])
sns.lineplot(data=filtered, x="n3", y="wt_3", marker='o', label="wt_3",␣
 ↪ax=axs[0])

axs[1].set_ylabel('Turn Around Times')
sns.lineplot(data=filtered, x="n3", y="tat_1", marker='o', label="tat_1",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n3", y="tat_2", marker='o', label="tat_2",␣
 ↪ax=axs[1])
sns.lineplot(data=filtered, x="n3", y="tat_3", marker='o', label="tat_3",␣
 ↪ax=axs[1])
```

[8]:
```
<AxesSubplot:xlabel='n3', ylabel='Turn Around Times'>
```





3.6 Waiting Time Analysis

- The WT of C1 remains almost constant as it executes for fewer time quantums.
- When only two processes are running, C2 and C3, their waiting times are nearly equal. When either of them finishes execution, their is no additional WT for the remaining process. Hence the WT will be same for C2 and C3 for all values of **n3**.

3.7 Turn Around Time Analysis

- The TAT of C1 remains almost constant as it executes for fewer time quantums.
- As **n3** increases, the TAT of C3 increases linearly as C2 and C1 complete execution before C3.
- C2 increases linearly while it executes with C3 and for larger values of **n3** it remains close to constant as it finishes execution before C3 for larger values of **n3**.

3.7.1 Note

The irregularities in the plots are due to interference from background processes and hardware constraints of the ~~Central Potato Unit~~ CPU.