**Assignment 1 report:**

**Part 1:**

At first, 5 active period vectors, uniformly distributed, are generated using MATLAB random generator function “randi()”. Fig. (1.1 to 1.5)

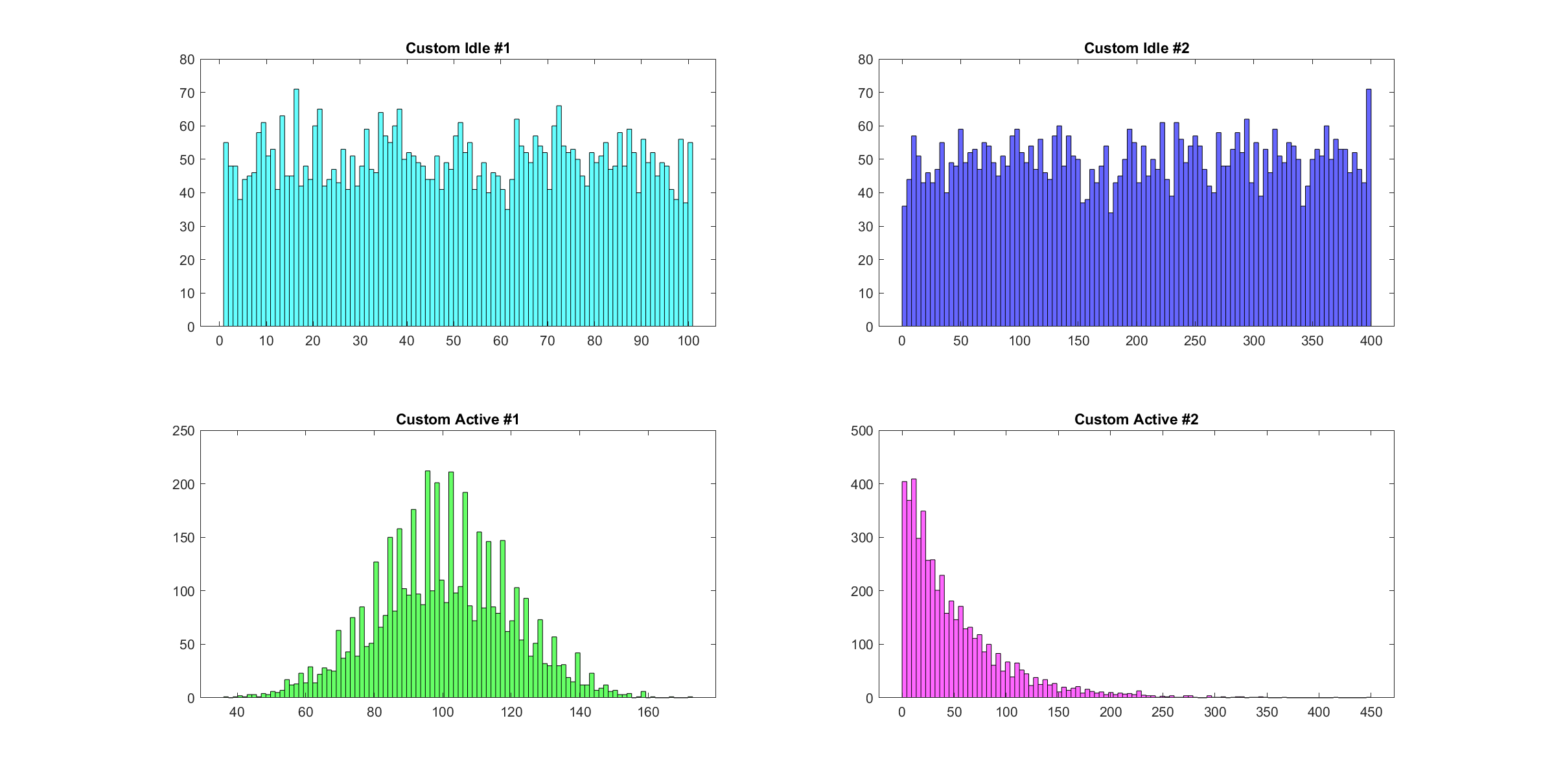
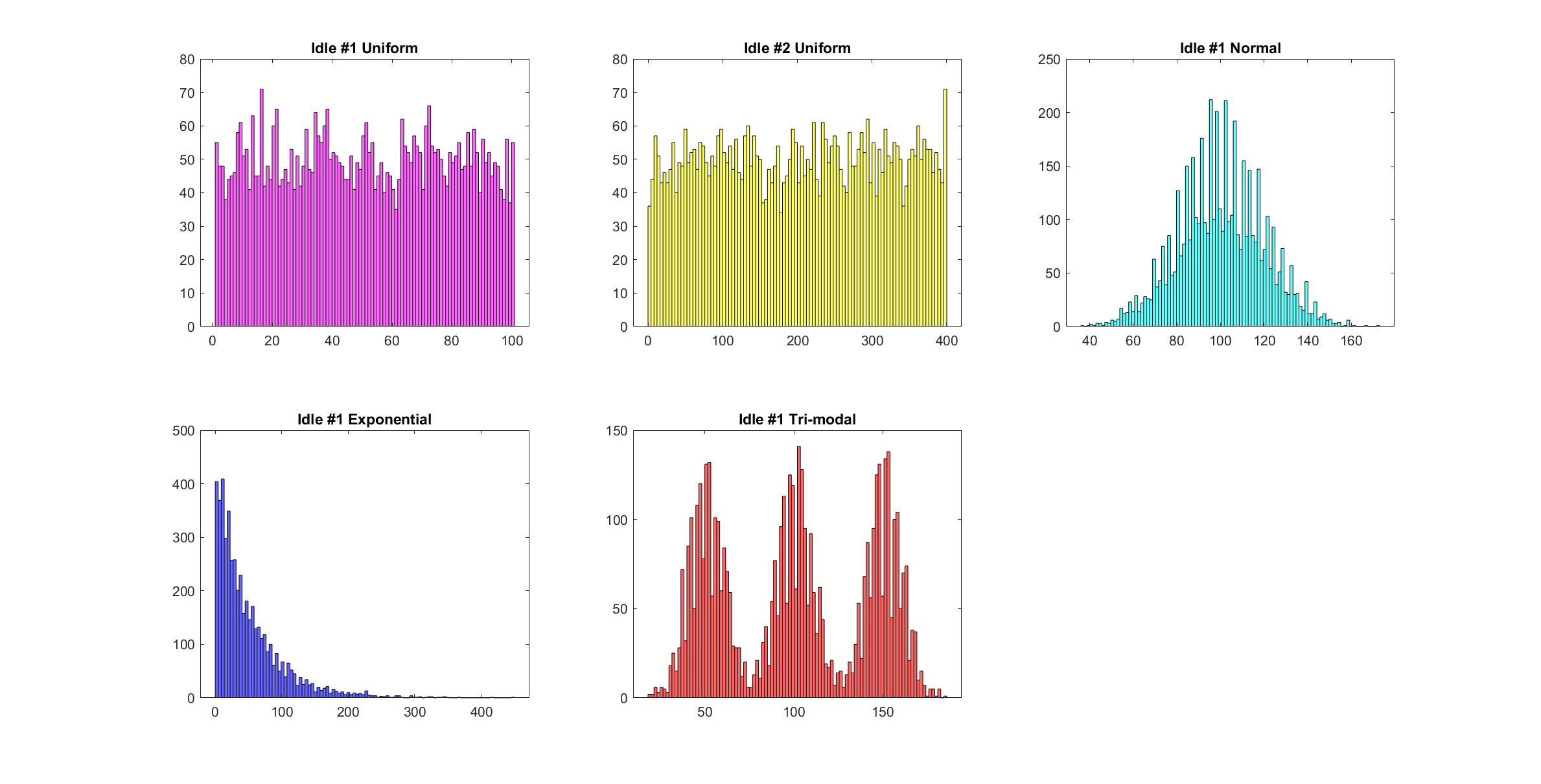
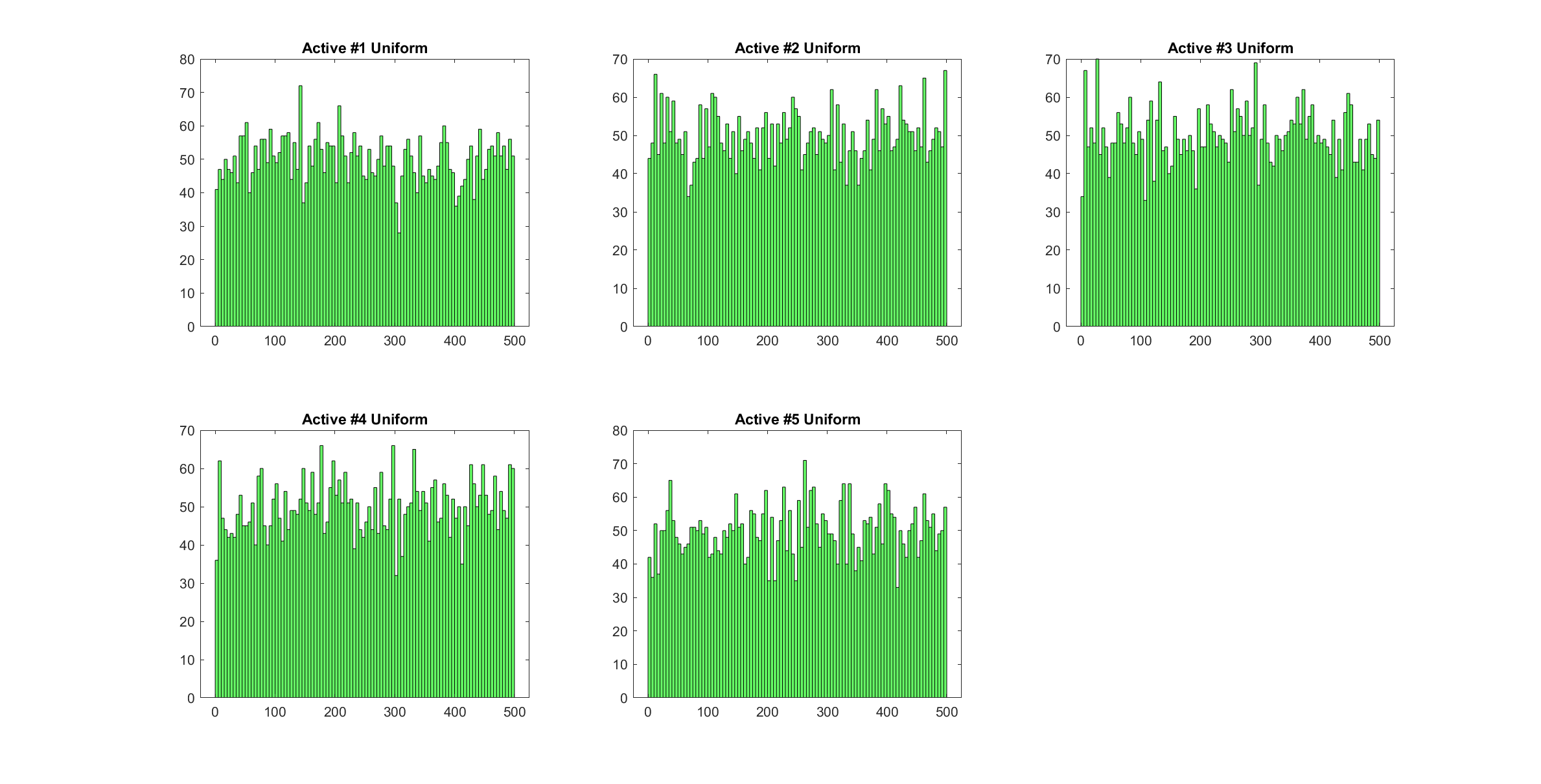
Then 5 different idle vectors are generated:

1. Uniform distribution, min = 1us, max=100us (high utilization) : for this vector 5000 random integer values between 1 to 100 using MATLAB random generator function “randi()”. Fig. (2.1)
2. Uniform distribution, min=1us, max=400us (low utilization): for generating this vector like previous one, 5000 random number between 1 to 400 has generated using MATLAB random generator function “randi()”. Fig. (2.2)
3. Normal distribution, mean=100us, standard deviation=20: for generating this vector, 5000 Normally distributed float numbers has generated, using MATLAB random generator function “random('Normal', ...)”, then using “Ceil()” function they rounded to up(to avoid ‘0’). Fig. (2.3)
4. Exponential distribution, mean=50us: 5000 random number generated using MATLAB random generator function “random(' Exponential', ...)”. Fig. (2.4)
5. Tri-modal distribution, mean = 50, 100, 150us, standard deviation=10: this vector generated by combining 5000 numbers which has made in this way: an integer number generated between 1-3, then based on this random number one of 3 normal distribution with different means used to generate this new number. Fig. (2.5)

All these vectors histograms are shown in Fig. 1 & Fig. 2.

Also, custom vectors histograms are in Fig. 3.

The code to generate, make histogram, save result in workloadProfile file are in “Workload\_profile\_generation.m” which is attached.



**Part 2:**

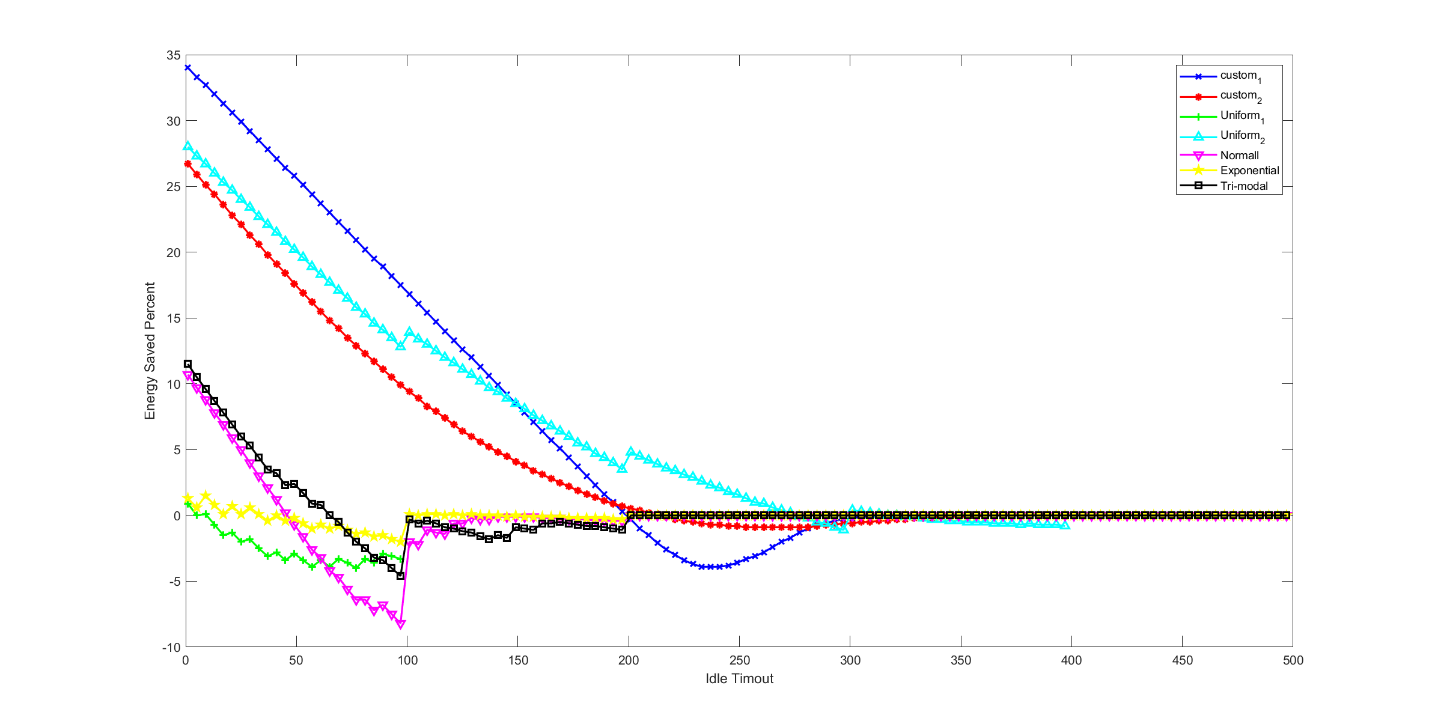
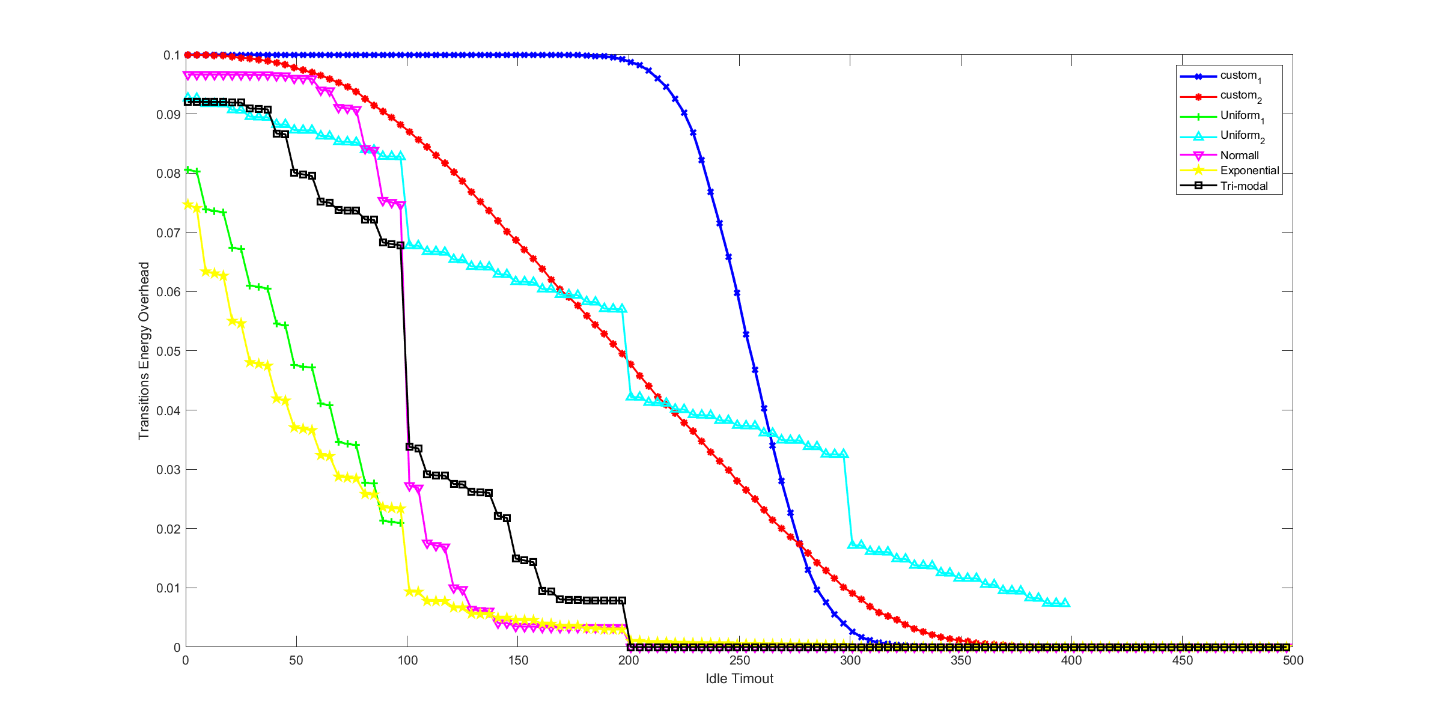
**First case (only idle state):**

In this case different timeouts have tested in 7 different workloads and result of energy overhead **Fig. 4** and saved energy **Fig. 5** using transitions are calculated.

From the figures this result could be observed:

1. for custom\_workload\_1 as timeout increases saved energy decreases until overhead is dominated the saving and it even gets negative. This workload has the most saved energy between others.
2. for custom\_workload\_2 is somehow the same as previous but with the less saved energy in lower timeouts and more overhead in higher.
3. for high utilization uniform distribution with any timeout value overhead is dominated so transitions are worsen the saved energy. However, for low utilization, this policy seems useful.
4. For normal distribution, low timeous works much better, cause it include most of the short idles. as it increases, power saving decreases until overhead dominated the saving. In This case worse power lose has happened.
5. For exponential, as a result most of idles are small, so saving is not that much to be considered.
6. For Tri-modal distribution, low timeouts include most of idles until timeout passes the first hill. Then it saving will decreases until it hit negative peak near the second hill. In this model, same as normal distribution, getting close to hills lowers the overhead as a result of decrees in translons (lower overhead), but missing lots of idles before the hill decrees the overal the efficiency.

Consequently, this model works better with workloads in which there is more long idles than shorts. Except, Exponential model and high utilization, all other models has better saving with low timeouts due to their sudo-convex shape.



- Result of timeout policy with the generated and provided profiles

- Timeout vs. energy saving (sweep timeout from 1us to the maximum length of idle time)

- Comparison between example timeout and extended timeout (with RUN-> SLEEP transition)

- Analysis on timing and energy overhead

**Assignment 3 report:**

- Description of implemented predictive policy

- Result of implemented predictive policy with the profiles

- Analysis on window size vs. energy saving

- Analysis on coefficient values vs. energy saving (model order)

- Comparison between predictive and timeout policies

- Analysis on timing and energy overhead

**A single**