Example of Attacks on two days

I experiment two kinds of attacks, the shifting attacks and the scaling attacks. We can divided this last type in three categories, namely, scaling up, scaling down and mirroring. The scaling down attack is flattening the curve and the mirroring attack make the high demand is low and a low demand is high. You can see the effect of these attacks on the following figures.

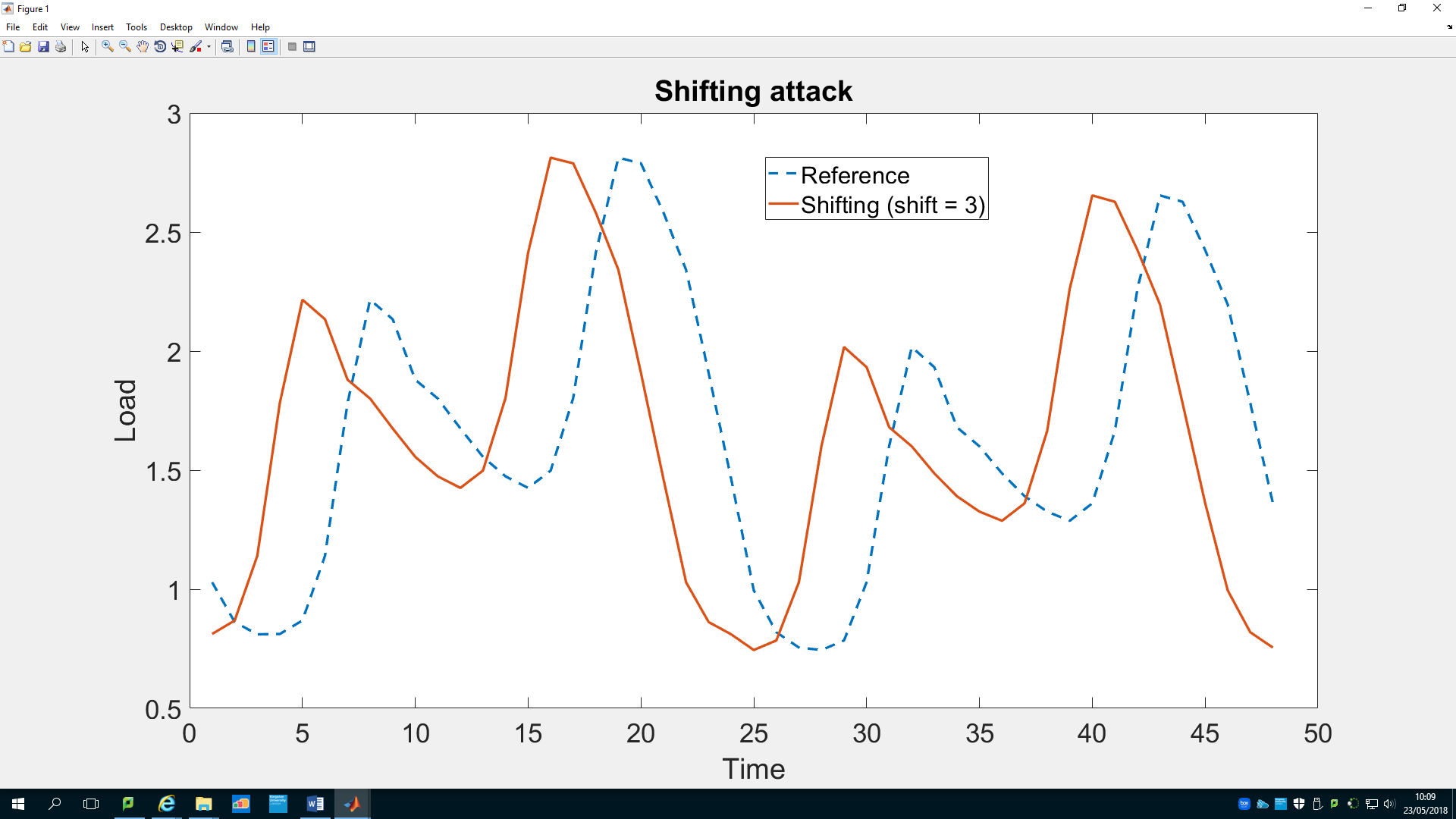


Figure 1

The value of the shift is an integer and is the number of hour to shift. For example on Figure 1, the curve is shifted 3 hours earlier and the three first values of each day become the three last values of the day.

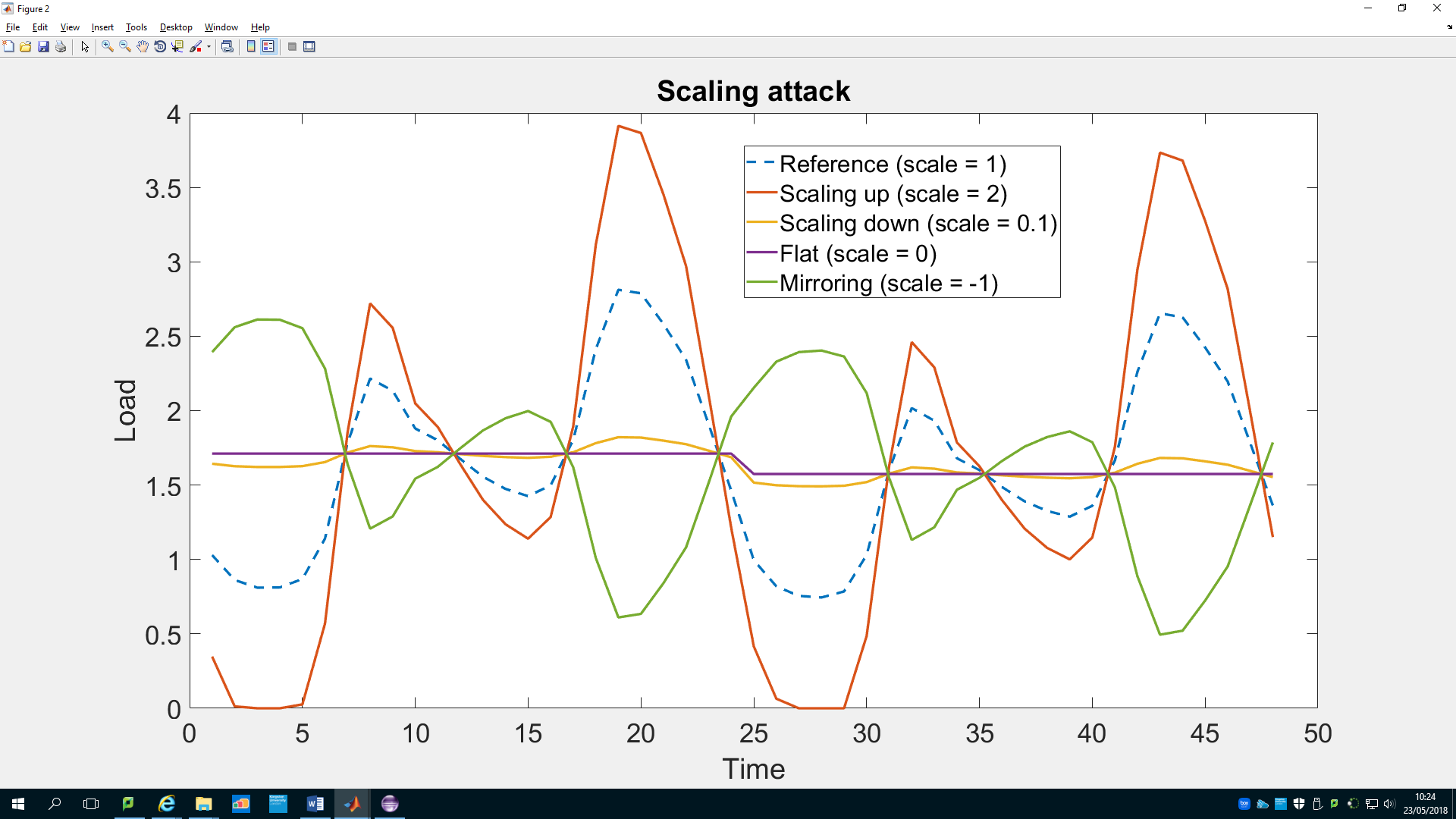


Figure 2

The scaling attack try to keep the same mean for the day, there is a difference if a new value become negative, this value is put to 0 and makes the mean changes. Each value is calculated as follow, it is the mean of the day more scale value times the difference between the old value and the mean. new = mean + scale \* (old -mean)  
You can see on the figure 2 to have some examples.

Weak attacker, he can only change his own forecast

(result/oneDayOnlyAttacker[…])

1. Update only his schedule after the game

Fariborz Baghaei tried some attacks of this kind but I did it again to have a better view of the algorithm and to see how the association of MatLab and Java works.

I tried different attacks among the scaling up, flattening and shifting attacks. As we could expect, we observe that the attacker doesn’t get benefit from these attacks. What we can conclude is the forecast of the attacker has only a small impact on the aggregated forecast and so on the actions the other players take. He is surely paying less on some interval but he is also paying more on other because he doesn’t change his demand. Moreover, the polynomial form of the cost function doesn’t help the attacker.

I tried to use at the same time a shifting attack and a scaling attack, I saw than the impact is bigger than the previous attack but still to small to make the attacker get a benefit from it.

Efficiency of the attacks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attack | Attacker (% of the reference utility) | | | All (% of the reference utility) | | |
| Low | Base | High | Low | Base | High |
| Scale 2 | 0.0008 | 0.0036 | 0.0041 | 0.0058 | 0.0132 | 0.0149 |
| Scale 3 | 0.0024 | 0.0094 | 0.0233 | 0.0147 | 0.0324 | 0.1178 |
| Scale 4 | 0.0041 | 0.0195 | 0.0397 | 0.0240 | 0.0756 | 0.2595 |
| Flat | -0.0008 | -0.0036 | -0.0044 | -0.0060 | -0.0130 | -0.0149 |
| Shift 2 | 0.0041 | 0.0052 | 0.0061 | 0.0069 | 0.0142 | 0.0197 |
| Shift 4 | 0.0066 | -0.0025 | 0.0071 | -0.0042 | -0.0062 | 0.0052 |
| Shift 6 | 0.0003 | -0.0122 | -0.0141 | -0.0309 | -0.0535 | -0.0589 |
| Shift 3 Scale 3 | 0.0154 | 0.0110 | 0.0231 | 0.0321 | 0.0791 | 0.1193 |

1. Update also his demand after the game (TO DO)

Strong attacker, he can change all the forecasts

(result/oneDayAttacker[…])

1. Update only his schedule after the game

In this part, the attacker can change the forecast of every player in addition to his own. So he can sharply change the aggregated forecast. At this point, we are considering that the players are naïve because they calculate their own forecast and then they play the game with a wrong forecast because of the attacker. We will see this problem more precisely in the next part. That means that the attacker is strong.

I tried the same attacks than previously but I reduced a bit the scale value (1.5, 2.5 and 3.5) and I find some interesting results. With all attacks, the attacker gets more or less benefit. The attacker is strong enough to sharply change the action of the other players. If we look at a shift attacks of four hours, the peaks are shifted four hour earlier, each player sees this wrong peak and so doesn’t charge his battery at this moment. Then the attacker can get back to the reality and decides to charge his battery four hours before the real peak, where all the other players don’t use much electricity and don’t charge their batteries.

The scale attack has a similar effect but at the peak time, it’s less efficient because the attacker try to make the other player to not consume at the peak time.

The flat attack works well because each player sees a flat aggregated forecast and has a flat own forecast, so he decides to not use his battery. The result is that the aggregated load is the same than without game and then the attacker can charge his battery alone before the peak when it is cheaper.

Efficiency of the attacks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attack | Attacker (% of the reference utility) | | | All (% of the reference utility) | | |
| Low | Base | High | Low | Base | High |
| Scale 1.5 | -5.2 | -1.7 | -0.7 | 7.6 | 7.6 | 7.5 |
| Scale 2.5 | -8.9 | -9.6 | -6.3 | 44.8 | 43.1 | 42.6 |
| Scale 3.5 | -3.6 | -2.9 | -2.1 | 79.6 | 78.0 | 78.2 |
| Flat | -19.5 | -10.1 | -6.5 | 16.9 | 16.7 | 16.4 |
| Shift 2 | -13.1 | -4.1 | -1.0 | 23.3 | 23.1 | 22.8 |
| Shift 4 | -22.6 | -7.1 | -1.7 | 43.5 | 40.4 | 39.9 |
| Shift 6 | -19.5 | -7.7 | -2.0 | 42.4 | 42.0 | 41.5 |
| Shift 3 Scale 1.5 | -24.4 | -5.7 | 0.6 | 53.9 | 53.8 | 53.1 |

I tried many values for the scale and the shift and I compared the benefit of the attacker for each value on the two following figures.

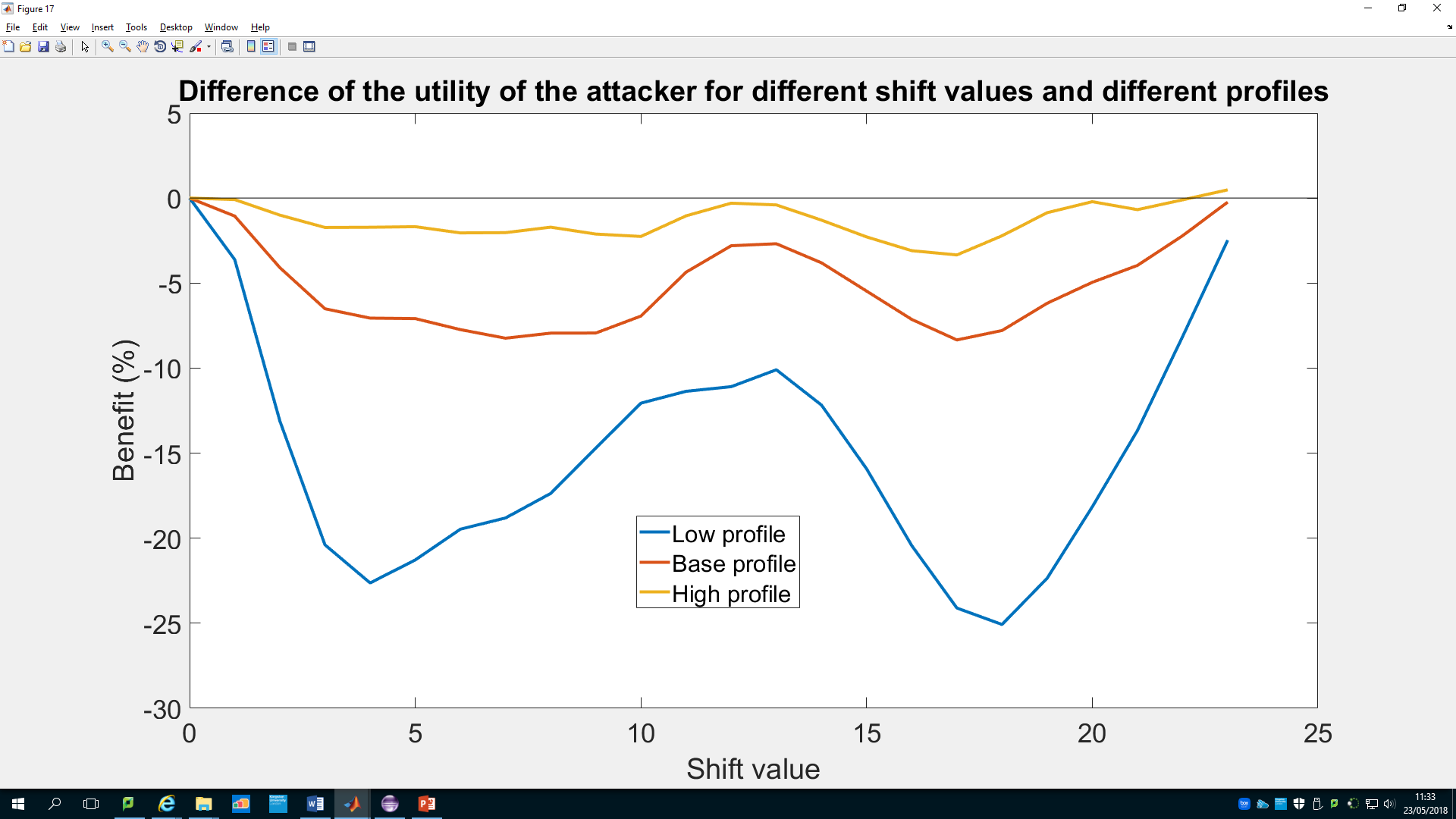


Figure 3

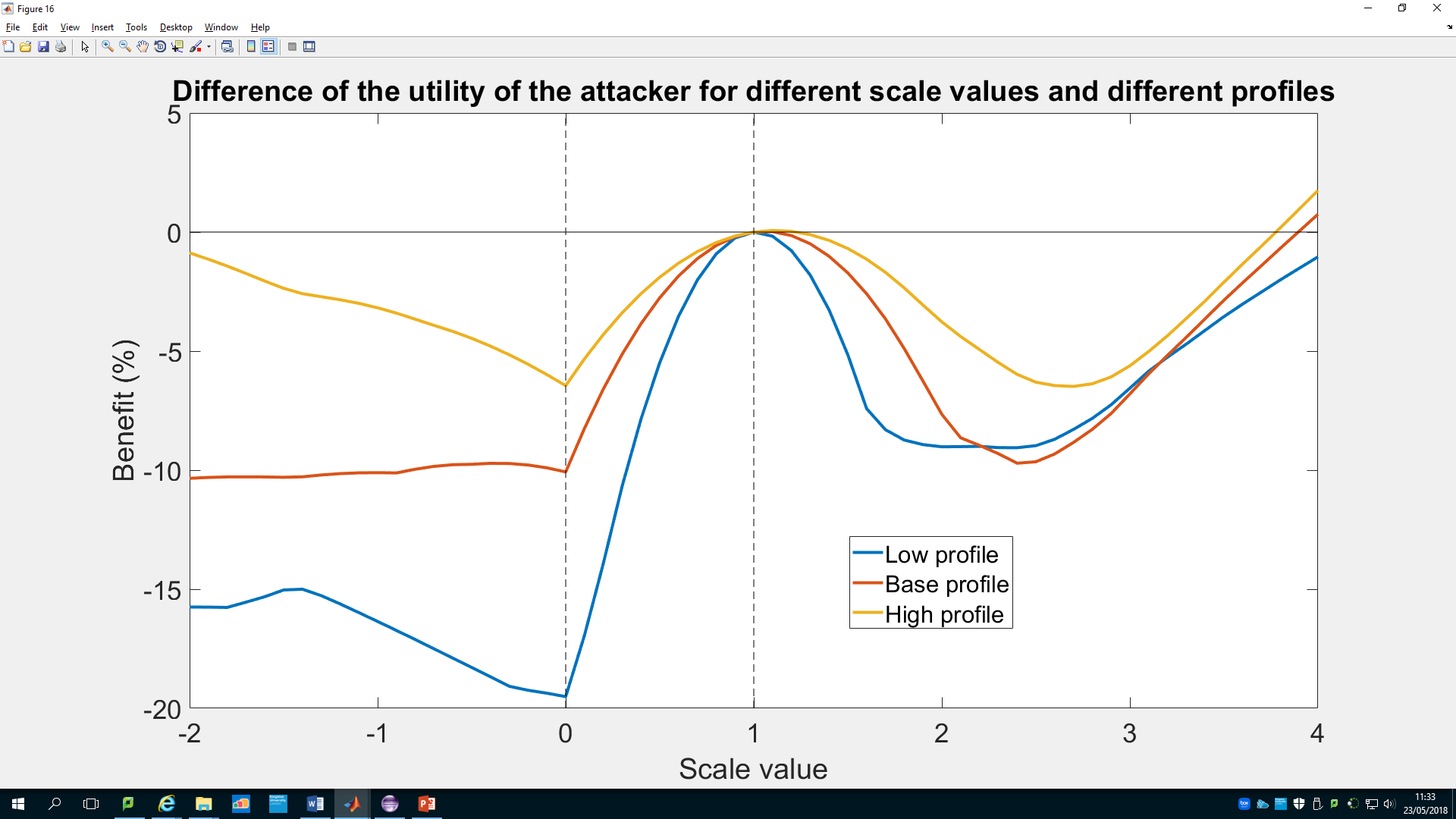


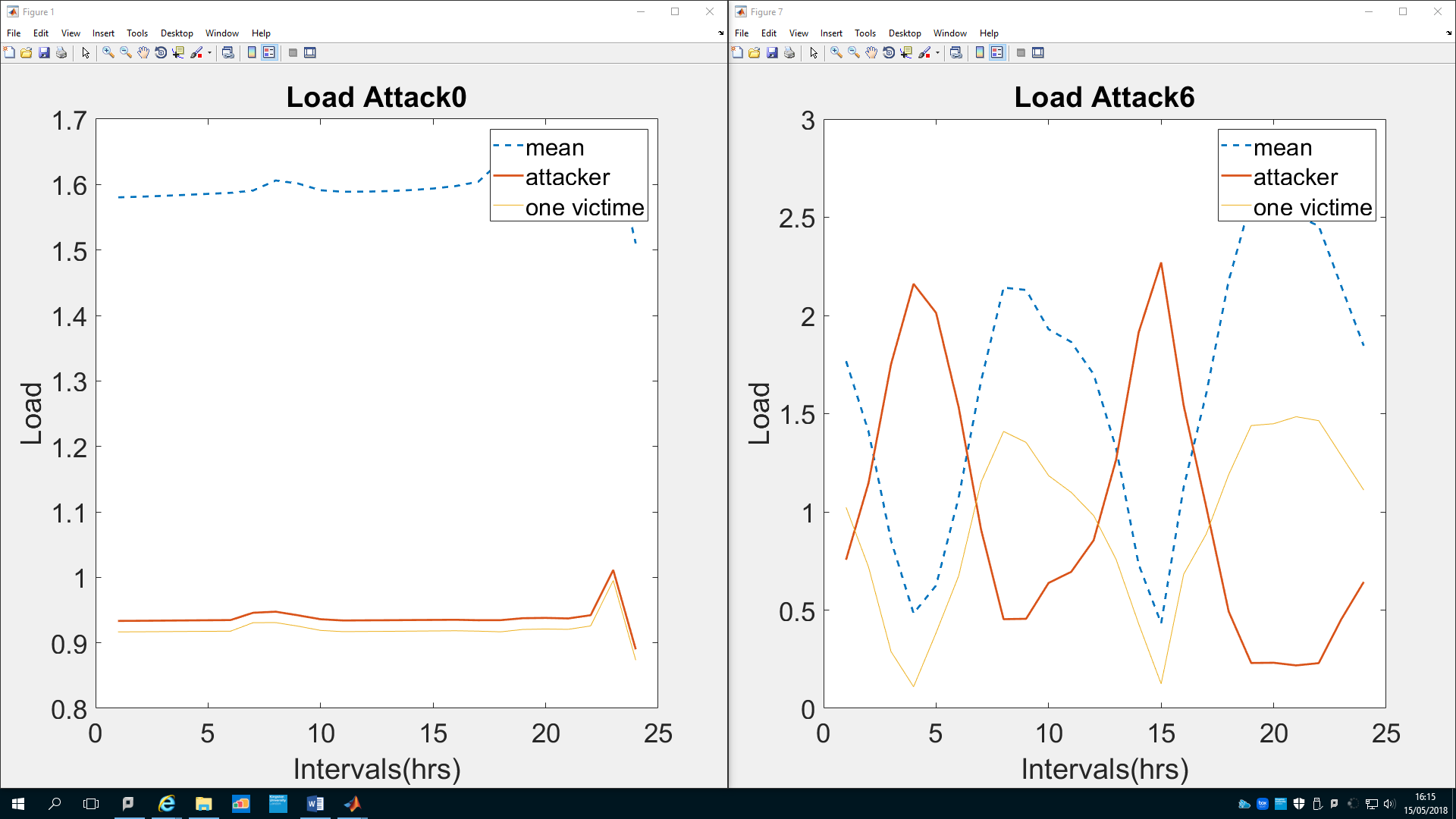
Figure 4

The benefit the attacker get is linked to his profile. We can see it on the two previous curves and it seems to be normal because when the attacker has a high profile, he has more difficulties to charge his battery to get a benefit from the aggregated load with peak. On the figure 4, the vertical lines make the difference between the three categories of attacks we saw at the beginning, namely, the scaling up attack (scale > 1), the scaling down attack (0 < scale < 1) and the mirroring attack (scale < 0).

1. Update also his demand after the game (TO DO)

Problem of the forecasts used during the game by the victims

The objective of the attacker is to make the other players buy their electricity at different moment than he does. That is represented on the following figure, the blue line is the mean of the load of all the player, it is a bit higher than the two other lines because our attacker has a low profile, it is just to have an idea of the form of the curve. The line in yellow is the load of a victim with the same profile than our attacker. On the left, we are just after a game without attack, that’s why the aggregated load is flat and the attacker has the same load than the victim. However, on the right, the attacker shifted all the forecasts, we can see that the aggregated load isn’t flat at all and that the attacker buy his electricity when the load is low.



Changing the forecast of each player directly is to powerful and can be easily detected by the player. He calculated his own forecast so he could see if the one he used is different.

Changing the forecast while it is sent from a player to the utility company will be easier but then, we have two options. The first will be to consider that the utility company sends back to each player the aggregated forecast and his own forecast and that the player doesn’t compare this forecast with the one he calculated earlier, which is not very satisfying. The second option will be to think that the utility company sends back only the aggregated forecast, but it means that each player is playing a different game. Each player is going to find a Nash equilibrium of his own game and the result of the real game could be a catastrophe… or not.

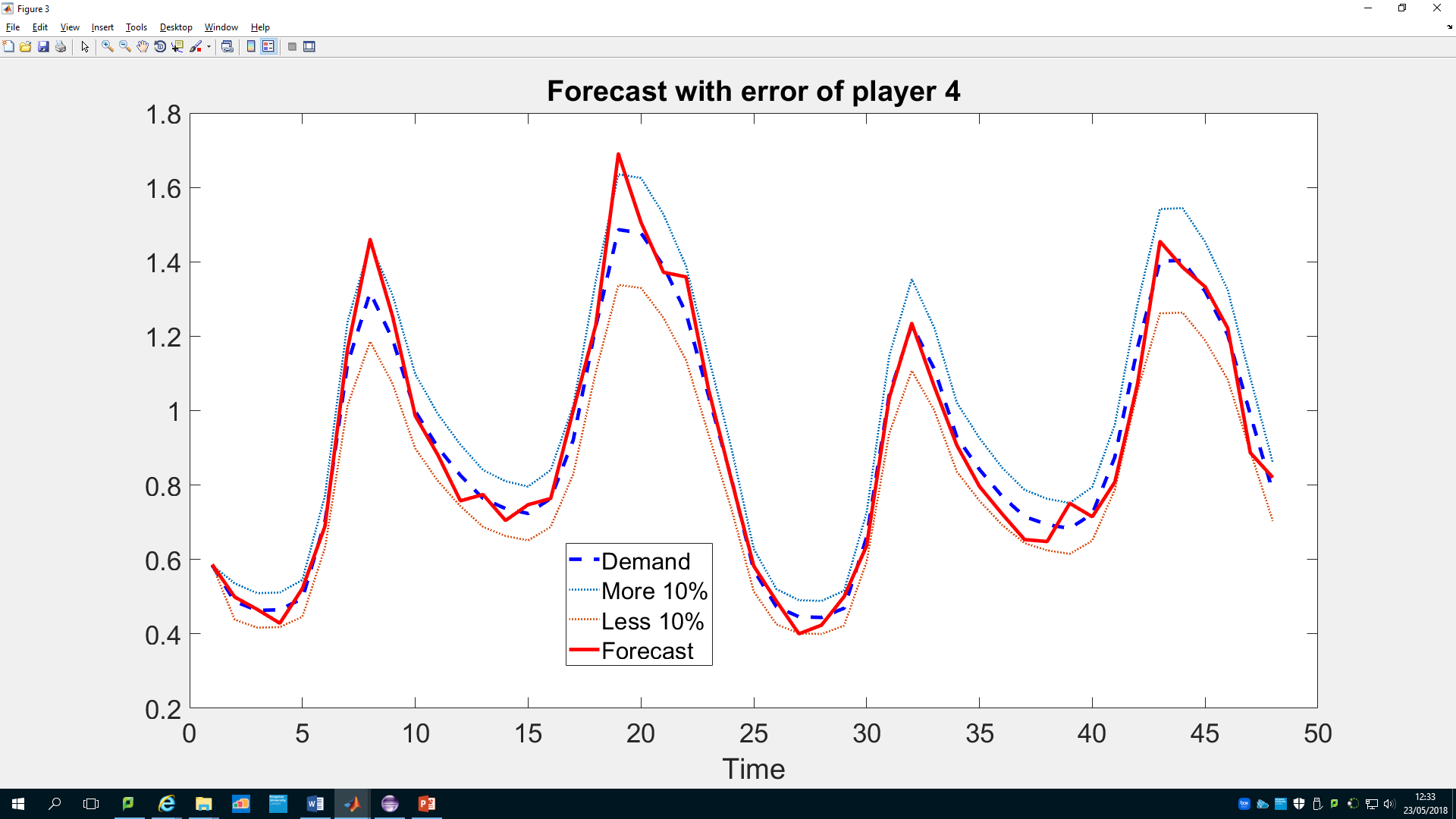
The second scenario will be what happen if the attacker changes directly the aggregated forecast.

NEXT STEP: try to run the algorithm to see if the changes are huge or not

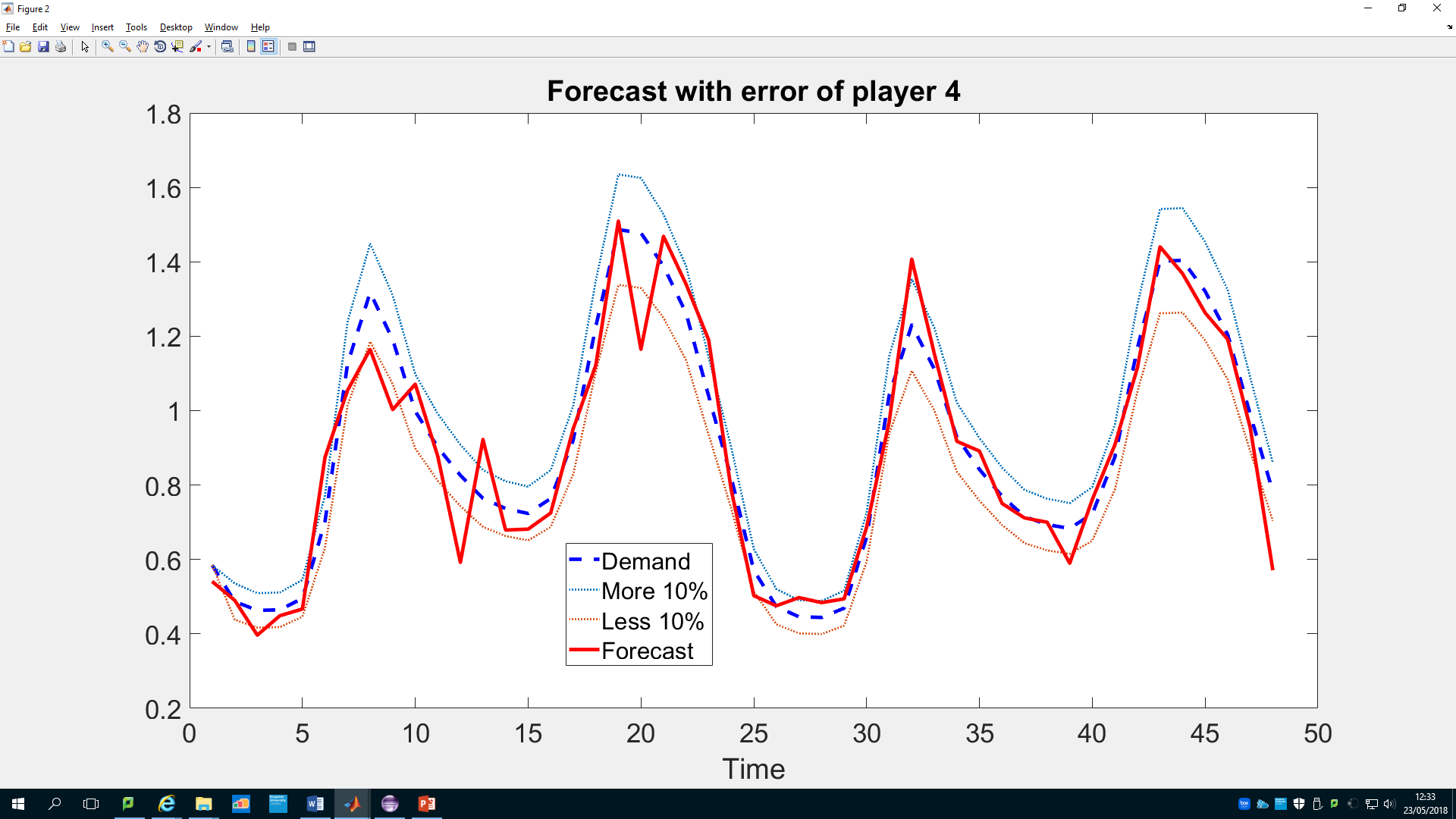
Add error to simulate a forecast

At the moment, each player give a forecast without error, they give directly the demand. But in reality they don’t have their demand and they give only a forecast. Because of that, the attacker has all the correct data and can adjust his schedule to get more benefit but if the players use forecast, then the attacker is going to make less benefit from the attack. That is why in this part, I try to add errors in the demand to simulate a forecast.

My first idea was to use a normal law to make new values a bit different than the real ones. For each value, the law defines the percentage of error I add. So the mean of my law is 0, I needed to find the good standard deviation. I decided to have 95 % of the new values between -10 % and 10 % and that is the case with a standard deviation equal to 5. You can see the result on two day on the following figure.



But with this law, we have a total error between 3.5 and 4.5 % which is not much. For make it more realistic, I would like to have 8 % of total error, and I find this with a standard deviation equal to 10 but the result on the following figure is not really good, we have a rough curve.



I’ll try an other idea which consist in applying the law on few value on the day and then choose the value regarding these particular error and the real value. (TO DO)

What next?

Swap attack

The attacker can change his demand and not only his schedule after the game.

See what benefit the attacker get when he changes only few forecasts.

Play with errors in the forecasts

Study attacks which could pass for errors (forecast errors allowed: 8-10 %)

Try to run the algorithm with the conceptual problem to see if the changes are huge or not

Attack defence system.

Gang attack.

Change architecture to make communication easier.

Mixed strategy.