Competing in a Prediction Tournament

Bachelor Project

Author

T. den Rooijen
(6089178)

Graduation Committee

J. Söhl (Supervisor)
R. Versendaal

Date of defence: June 29, 2026

Delft
University of
Technology

Layman's Summary

Prediction tournaments have been organised more often in the last few years in order to find precise forecasters. In prediction tournaments, contestants need to predict how large the chance is that an event will happen. Simulations indicate that predictors who predict the real chance precisely almost never win the prediction tournament. This is caused by the luck of the less precise forecasters. Because of this effect, accurate predictors can adapt their strategy to increase their chance of winning. This report investigates which strategies contestants can use to increase their chance of winning. Therefore, simulations are done. From these simulations it seems that contestants can use strategies called the exponential strategy and the random exponential strategy to increase their chance of winning if none or half of the opponents use a strategy. If all opponents use a strategy, then it is recommended to use no strategy. So, it is important to do something different from what the opponents do.

Abstract

In the last few years, prediction tournaments are organised more often. Organisers of these tournaments want to find statistical models that are the best for predicting future events. Most of the time, the winner of a prediction tournament receives a reward.
In a prediction tournament, every contestant gets a number of questions about how large the probability is that an event will happen before a specific date. Simulations indicate that contestants who perfectly predict these probabilities almost never win the tournament. This effect ensures that an accurate forecaster could increase her chance of winning by introducing some noise in her predictions. The aim of this report is to find strategies that contestants can use to increase their chance of winning.
In this report, five strategies are introduced. These strategies are called hard-thresholding, soft-thresholding, polynomial strategy, exponential strategy and random exponential strategy. Each strategy depends on one parameter. For each strategy, simulations are performed for different settings to determine which strategy results in the most victories. In order to determine the best parameter for a strategy, polynomial regression is used on the data from the simulations.
It seems from the simulations that the exponential strategy has the largest positive impact on the number of wins for accurate contestants if all opponents use no additional strategies. If half of the opponents use an exponential or random exponential strategy, then the most accurate contestants are recommended to use a random exponential strategy. Using a strategy seems to have only negative impact on the chance of winning of a contestant if all opponents use a exponential or random exponential strategy.
The best parameter for an exponential strategy seems to be smaller for more inaccurate forecasters. The most inaccurate contestants who compete in a prediction tournament are recommended to use no strategy in all previous situations.

Contents

Chapter 1

Introduction

In the last few years, prediction tournaments have been organised more frequently. These tournaments are held to find the best statistical models for predicting the future. Usually, the contestants with the best results are rewarded with money. An example of such a prediction tournament from 2018 was IARPA's Geopolitical Forecasting Challenge [1]. Contestants get questions like who will win elections in a country or what is the price of an specific resource on a specific date? After this tournament $200,000 was divided amongst the best contestants in the tournament [1]. Another example of a prediction tournament is the Metaculus Quarterly Cup [2]. Most questions were about the political and economic situation in the world. However, there were also a few questions about sports and music like will a specific concert take place? A third example of a prediction tournament is the 2026 World Elections Challenge of Good Judgement Open [3]. In this tournament, contestants get questions about elections in different countries.

David J. Aldous has studied prediction tournaments and found a paradox in these tournaments [4]. Contestants who do prefect predictions almost never win the tournament [4]. This can be viewed from two different perspectives. The fact that accurate forecasters almost never win the prediction tournament is a problem for the organiser, since this organiser wants to find the best forecasters and wants them to be the winners. However, an individual contestant could try to use the paradox to increase her chance of winning. The bachelor thesis of V. M. van der Eng [5] discusses prediction tournaments more in general by analysing various aspects of prediction tournaments such as scoring rules and the distribution of the true probabilities. My bachelor thesis will be focussed more on

the perspective of an individual contestant. A part of the bachelor thesis of V. M. van der Eng [5] discusses a strategy for contestants in which contestants predict only 0 or 1. This strategy was not recommended to contestants if half of the other contestants did not use a strategy. V. M. van der Eng [5] suggested that adding some noise to the predictions of a contestant could increase the chance of winning. In my bachelor thesis, several other strategies that introduce some noise in a contestant's predictions will be analysed to increase the chance of winning of an individual contestant. Therefore, the question that this report will discuss is: which strategy could a contestant use to increase her probability to win?

To answer this question various strategies will be introduced. Simulations will be performed to investigate which strategy has the most positive impact on the number of wins if all opponents use no strategy. After this, other simulations are done for situations in which opponents use strategies.

Before studying how the probability that a contestant wins the prediction tournament can be increased, it is important to know how the prediction tournament works. This will be discussed in chapter 2. An important rule of the prediction tournament that will be explained in this chapter is how the scores of the contestants are calculated and how the winner is determined on the basis of these scores. At the end of chapter 2 the paradox in these prediction tournaments will be discussed.

Chapter 3 consists of two parts. In the first part of this chapter five strategies are introduced that could potentially increase the number of wins of the contestant who uses this strategy. In the second part of chapter 3 tournaments are simulated in which one of the contestants uses one of the strategies. This simulations are used to determine which strategies are preferable for different contestants with various levels of accuracy, in the case that all opponents use no strategy.

If more contestants gain knowledge about which strategies are useful to increase their chance to win, then more contestants will use these strategies. Most times when opponents use strategies, the strategies introduced in chapter 3 do not result in a larger chance of winning than using no strategy. Therefore, a random strategy is introduced. This strategy uses per question a strategy from chapter 3 with probability 0.5 or no additional strategy with probability 0.5. Chapter 4 studies which strategies are useful when some of the opponents use a specific strategy. Three strategies of opponents will be discussed. One of these is better for the

most accurate forecasters, an other one is better for forecasters that are in the middle of the ranking of accuracy and the last one is a random strategy. For all three strategies of the opponents it will be determined which strategy our contestant could use to increase the chance of winning, if all opponents use a strategy or half of them use a strategy.

Chapter 2

Prediction Tournament

Before analysing different strategies for contestants, it is important to have a clear understanding of how the prediction tournament works. This is covered in section 2.1. After that the prediction tournament paradox is explained further in section 2.2. The aim of this is to gain knowledge which contestants can use to design strategies.

2.1 Design of the prediction tournament

In a prediction tournament contestants get questions about events that could happen in the future. For every question, the contestants need to predict the probability that a certain event happens before a specific date [4]. In the prediction tournament that will be studied in this report the contestants state their predicted probabilities once and cannot change their predictions afterwards.

First consider one question and one contestant. The contestant states a probability which she thinks is the probability that the event will happen. This probability stated by the contestant is called q. The true probability that the event will happen before the specific date is denoted by p. After the specified date there are two options, either the event took place before the specific date or the event did not. The score of the contestant for this question will be determined as in the following equation:

$$Score = \begin{cases} (1-q)^2 & \text{if the event happened} \\ q^2 & \text{if the event did not happen} \end{cases} \tag{2.1}$$

This is for 1 question, but in a normal prediction tournament contestants get more questions. The prediction tournament that will be studied in this report consists of 100 questions. A contestant's total score is calculated as the sum of the scores per question. The winner of the tournament is the contestant with the lowest total score.

2.2 Prediction tournament paradox

Let X be the random variable for the score of a contestant for 1 question. From the scoring rule it is known that $\mathbb{P}[X = (1-q)^2] = p$ and $\mathbb{P}[X = q^2] = 1 - p$. Using this, the expected score of a contestant is calculated by:

$$\begin{aligned} \mathbb{E}[X] &= p(1-q)^2 + (1-p)q^2 \\ &= p - 2pq + pq^2 + q^2 - pq^2 \\ &= p - p^2 + p^2 - 2pq + q^2 \\ &= p(1-p) + (q-p)^2 \end{aligned}$$

Let S be the total score of the tournament of a contestant. S is the sum of the X's for every question. So, the expectation of S is the sum of the expectations of the X's. Therefore, the expectation of S can be written as

$$\mathbb{E}[S] = \sum_{i=1}^{100} p_i(1-p_i) + \sum_{i=1}^{100} (q_i - p_i)^2.$$

The first term in the equation above only depends on the true probabilities that events happen. So, this is the same for all contestants. Conversely, the second term depends on the differences between the probabilities stated by the contestant and the true probabilities. The larger this differences are, the higher the contestant's expected total score.

To gain more information on the contestant's scores and which contestants will win the tournament David J. Aldous did some simulations [4].

To do so a variable σ called a contestant's prediction error is introduced. Every contestant in the tournament has a own σ. This σ indicates how accurately the contestant predicts probabilities compared to the true probabilities. The tournament that will be studied has 300 contestants. Different contestants have different accuracies. David J. Aldous modelled this using σ. For each question, a contestant's prediction is modelled by taking arbitrary $p + \sigma$ or $p - \sigma$ with probability 0.5. The true probabilities p are not known to the contestants, but these true probabilities are used to model the predictions of the contestants. The probabilities stated by the contestants are bounded such that they stay in the interval [0,1]. This is done by taking 0 instead of $p - \sigma$ if $p - \sigma < 0$ and taking 1 instead of $p + \sigma$ if $p + \sigma > 1$. As mentioned in section 2.1, the tournament consists of 100 question. The true probabilities 0.05, 0.15, 0.25 ... 0.95 all occur 10 times.

In the prediction tournament, 300 contestant participate. These contestants all have a different σ. In the first situation, the values of σ lie between 0 and 0.299 and differ with steps of 0.001. All contestants have a rank, the lower the rank of the contestant, the closer the contestant's σ is to 0. David J Aldous has done simulations of this tournament, that resulted in the histogram in figure 2.1. This histogram shows how the number of wins are divided over the different ranks after competing different times in a prediction tournament. This histogram shows something unexpected. The contestants with the lowest rank that predict almost precisely the true probabilities almost never win the tournament. However, the contestants with ranks around 100 who differ with about 0.1 from the true probabilities have the largest chance to win the tournament. This phenomenon is called the prediction tournament paradox.

Most contestants with a larger σ predict for most questions probabilities further from the true outcome than the true probabilities. They will clearly lose from the contestants that predict the true probabilities almost perfectly. However, there are always a few contestants with a larger prediction error that predict most of the times closer to 0 than the true probabilities if the event does not happen and closer to 1 if the event happens. These contestants will beat the perfect predictors due to luck. This is how the prediction tournament paradox is caused.

In figure 2.2 it can be seen that if the σ's become higher, then the most precise predictors have actually the largest chance to win the tournament. But this chance is smaller than intuition would tell.

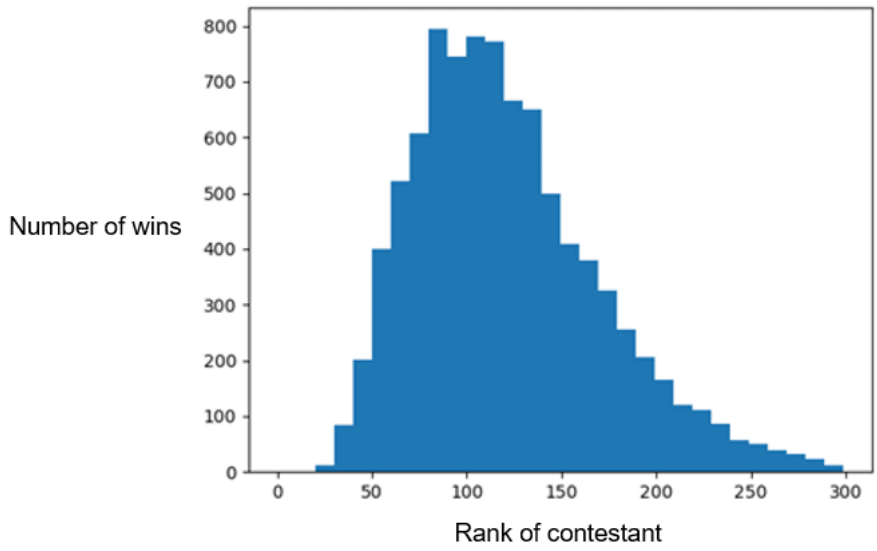


Figure 2.1: Histogram of number of wins per rank of contestant. The contestants in the prediction tournaments have σ's within the interval [0,0.3]. Resource: Aldous, D. J. (2019). A Prediction Tournament Paradox. The American Statistician, 75(3), 243-248. `https://doi.org/10.1080/00031305.2019.1604430`

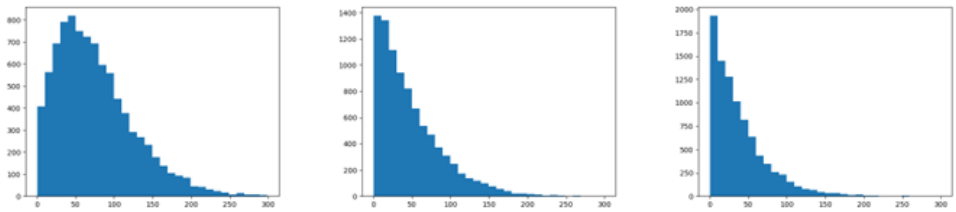


Figure 2.2: Histograms of number of wins per rank of contestant. The contestants in the prediction tournaments have σ's within the intervals [0.05,0.35], [0.1,0.4] and [0.15,0.45]. Resource: Aldous, D. J. (2019). A Prediction Tournament Paradox. The American Statistician, 75(3), 243-248. `https://doi.org/10.1080/00031305.2019.1604430`

Chapter 3

Analysis of different strategies

In the rest of this report, the case in which the contestants have σ's between 0 and 0.3 will be studied. In chapter 2 it became clear that in this case the most accurate predictors did not have the largest chance to win the tournament. As mentioned in the bachelor thesis of V.M. van der Eng, accurate predictors could increase their chance of winning by adding some noise in their predictions [5]. In section 3.1 four different strategies are introduced that add some noise to the predictions by predicting closer to 0 for small probabilities and closer to 1 for high probabilities. After that, tournaments will be simulated in section 3.2 in which only one contestant uses such a strategy.

3.1 Different strategies

To increase the winning chance four different strategies are introduced. Each strategy adds a bit of noise to the predictions. The aim of this is to predict often closer to the true outcome than without using a strategy. A strategy is a function that takes the probability q predicted by the contestant as input and gives a value that the contestant can state to increase the chance of winning. The probability stated by a contestant after using a strategy is called $\tilde{q}$. As for all probabilities, $\tilde{q}$ needs to be between 0 and 1.

3.1.1 Hard-thresholding

A strategy that could be used by contestants is hard-thresholding. The contestant predicts 0 if the true probability is small and 1 if the true probability is large. For true probabilities around 0.5, the contestant state this true probabilities. This strategy depends on a parameter $c \in (0; 0.5]$. This parameter indicates for which true probabilities the contestant states 0 or 1. If $q < c$, then the contestant predicts 0. If $q > 1 - c$, then the contestant predicts 1. For the q's between c and $1 - c$, the contest predicts q. So, $\tilde{q}$ after using hard-thresholding can be determined by

$$\tilde{q} = \begin{cases} 0 & \text{if } q < c \\ p & \text{if } c \leq q \leq 1 - c \\ 1 & \text{if } q > 1 - c \end{cases} .$$

Hard-thresholding is shown for $c = 0.2$ in figure 3.1.

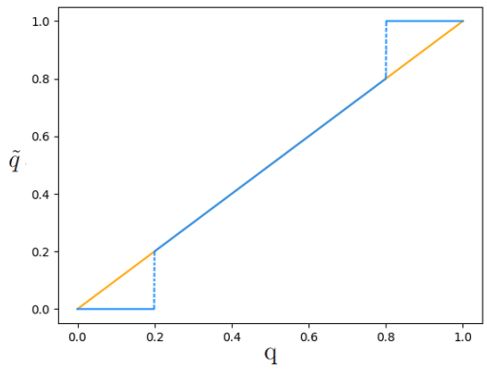


Figure 3.1: Probabilities that contestants using hard-thresholding state. The orange line shows contestants without strategy who state the true probabilities. In this case the parameter is equal to 0.2.

3.1.2 Soft-thresholding

Another strategy that predicts 0 for small true probabilities and 1 for large true probabilities is soft-thresholding. This strategy also depends on a parameter $c \in (0; 0.5)$. Just like hard-thresholding, contestants using soft-thresholding predict 0 if $q < c$ and 1 if $q > 1 - c$. However, for

true probabilities between c and $c-1$, contestants using soft-thresholding state probabilities on the line between the points $(c, 0)$ and $(1-c, 1)$. Therefore, $\tilde{q}$ can be calculated by

$$\tilde{q} = \begin{cases} 0 & \text{if } q < c \\ \frac{q-c}{1-2c} & \text{if } c \leq q \leq 1-c \\ 1 & \text{if } q > 1-c \end{cases} .$$

Soft-thresholding is shown for $c = 0.2$ in figure 3.2.

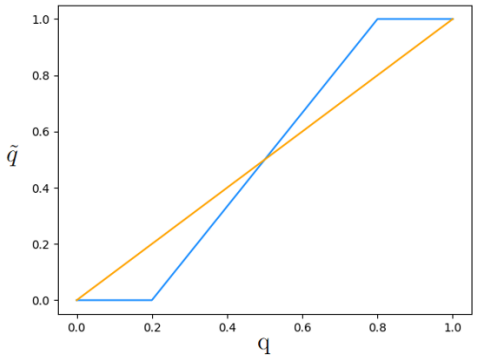


Figure 3.2: Probabilities that contestants using hard-thresholding state. The orange line shows contestants without strategy who state the true probabilities. In this case the parameter is equal to 0.2.

3.1.3 Polynomial strategy

The third strategy is called the polynomial strategy. This strategy is based on a polynomial function of degree 3. This polynomial is

$$\tilde{q} = aq^3 + bq^2 + cq + d$$

with a, b, c and d parameters that are free to choose. Two requirements for an efficient strategy are that $\tilde{q} = 0$ if $q = 0$ and $\tilde{q} = 1$ if $q = 1$. This first requirement results into $d = 0$. The second requirement implies that $a + b + c = 1$, so $c = 1 - a - b$. If these requirements are plugged into the original polynomial, then the polynomial for $\tilde{q}$ becomes

$$\tilde{q} = aq^3 + bq^2 + (1 - a - b)q.$$

For simulations it is preferred to only have one unknown parameter. Therefore, a third requirement is added. It must hold that if $q = 0.5$,

then $\tilde{q} = 0.5$. This requirement ensures that the contestant does not always overestimate or always underestimate the actual probabilities. The third requirement gives that:

$$0.125a + 0.25b + 0.5(1 - a - b) = 0.5$$

Collecting all terms with a and b together on the left side of the equation and taking 0.5 to the right side yields:

$$-0.375a - 0.25b = 0$$

Rearranging the terms gives that $b = -1.5a$. If this is substituted into the polynomial for $\tilde{q}$, the polynomial used in the polynomial strategy can be expressed as

$$\tilde{q} = aq^3 - 1.5aq^2 + (1 + 0.5a)q.$$

The predictions must be in the interval [0, 1], so the polynomial only may give values that lie between 0 and 1. To ensure this the derivative of the function must be larger or equal to 0 at 0. Because otherwise the function will give negative values for p's close to 0. So, $0.5a + 1 \geq 0$ must hold. This results into a constraint on the parameter a that this parameter must be greater or equal to -2. If $a = 0$, then every $\tilde{q} = q$ for $q \in [0, 1]$. If $a < 0$, then $\tilde{q} < q$ for $q < 0.5$ and $\tilde{q} > q$ for $q > 0.5$. For $a > 0$ it works the other way around. It is preferred that for low values of q $\tilde{q}$ is lower or equal to q and that for large values of q $\tilde{q}$ is greater or equal to q. Therefore, it is preferred to have $a < 0$. So, parameter a can be chosen freely in the interval [-2, 0). A graph of the polynomial strategy using parameter $a = -2$ is shown in figure 3.3.

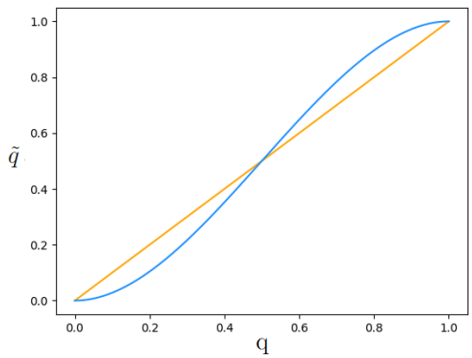


Figure 3.3: Probabilities that contestants using the polynomial strategy state. The orange line shows contestants without strategy who state the true probabilities. In this case the parameter is equal to -2.

3.1.4 Exponential strategy

The last strategy that is analysed in this chapter is the exponential strategy. This strategy is constructed out of two exponential functions. In the exponential strategy, $\tilde{q}$ is calculated by

$$\tilde{q} = \begin{cases} a_1 e^{bq} + c_1 & \text{if } q \leq 0.5 \\ a_2 e^{-bq} + c_2 & \text{if } q > 0.5 \end{cases}$$

where variables a_1, a_2, b, c_1 and c_2 are yet to be determined. The aim is to express four of these parameters in terms of one other parameter.

Four of the parameters will be fixed such that the first part goes from (0, 0) to (0.5, 0.5) and the second part goes from (0.5, 0.5) to (1, 1). The first constraint that $\tilde{q} = 0$ if $q = 0$ results into $c_1 = -a_1$. The second constraint yields:

$$a_1 e^{0.5b} - a_1 = 0.5$$

Dividing by $e^{0.5b} - 1$ on both sides gives:

$$a_1 = \frac{1}{2(e^{0.5b} - 1)}$$

Plugging the results from these constraints into the first part of the formula for the exponential strategy yields the following formula for $\tilde{q}$ if $q \leq 0.5$:

$$\tilde{q} = \frac{e^{bq} - 1}{2(e^{0.5b} - 1)}$$

The second part of the function must go through the points (0.5, 0.5) and (1, 1). This first requirement results into:

$$a_2 e^{-0.5b} + c_2 = 0.5$$

Rearranging the terms gives:

$$c_2 = 0.5 - a_2 e^{-0.5b}$$

Plugging this and the point (1, 1) into part two of the formula for the exponential strategy yiels:

$$a_2 e^{-b} + 0.5 - a_2 e^{-0.5b} = 1$$

Taking 0.5 to the right side of the equation and dividing both sides by $e^{-b} - e^{-0.5b}$ results into the following expression for a_2:

$$a_2 = \frac{1}{2(e^{-b} - e^{-0.5b})}$$

Plugging these expressions of the parameters into part two of the formula for the exponential strategy gives for $q > 0.5$:

$$\tilde{q} = \frac{e^{-bq}}{2(e^{-b} - e^{-0.5b})} + 0.5 - \frac{e^{-0.5b}}{2(e^{-b} - e^{-0.5b})}$$

All three terms can be written as one fraction to receive the final expression for the second part of the function for the exponential method. At the end, $\tilde{q}$ for the exponential strategy can be determined by

$$\tilde{q} = \begin{cases} \frac{e^{bq}-1}{2(e^{0.5b}-1)} & \text{if } q \leq 0.5 \\ \frac{e^{-bq}+e^{-b}-2e^{-0.5b}}{2(e^{-b}-e^{-0.5b})} & \text{if } q > 0.5 \end{cases}.$$

To guarantee that $\tilde{q} < q$ for $q < 0.5$ and $\tilde{q} > q$ for $q > 0.5$, parameter b must be larger than 0. A graph of the exponential strategy with parameter $b = 10$ is shown in figure 3.4.

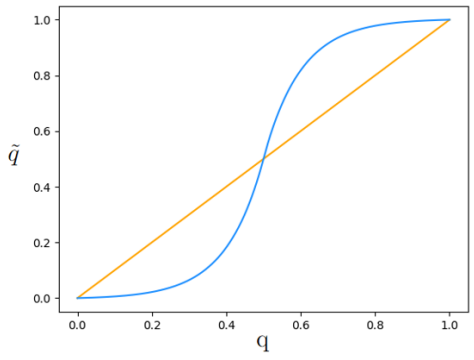


Figure 3.4: Probabilities that contestants using the exponential strategy state. The orange line shows contestants without strategy who state the true probabilities. In this case the parameter is equal to 10.

3.2 Simulations

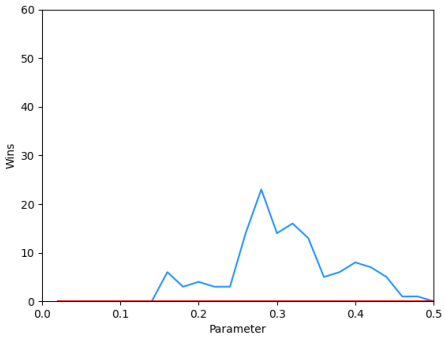
To test whether using one of these strategies actually increase the probability to win, simulations are done in this section. The setting of the simulations is the same as for the simulations done by Aldous as described in section 2.2. In this case the σ's of the contestants goes from 0 to 0.299 with steps of 0.001. In the simulations in this section one of the contestants is replaced by a contestant with the same σ who uses a strategy. The other contestants called the opponents, still use no strategy. An assumption of this model is that contestants know their own σ. The contestant using a strategy uses this strategy on the probabilities q that she would predict without a strategy, that could be $p + \sigma$ or $p - \sigma$. The aim of this research is to increase the number of wins of the contestant with a strategy as much as possible. To do this simulations the python code from the bachelor thesis of V. M. van der Eng [5] is used and extended to our cases. The resulting python code for prediction tournaments with contestants that use strategies can be found in appendix I.

3.2.1 Global simulations

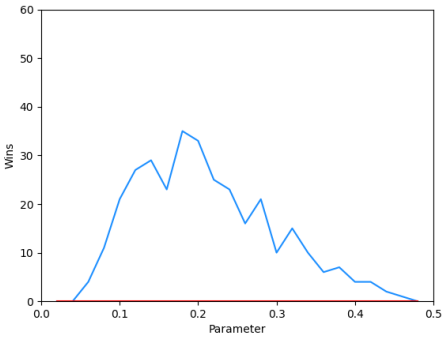
As a first step, the number of wins of different contestants are determined after 3,000 prediction tournaments. In order to get more precise results, this is done trice and the final result is the average number of wins after these 3 simulations. This is done for all four strategies with varying parameters. Figure 3.5 shows the number of wins of the contestant that predicts always the true probabilities when she uses a certain strategy. This figure shows that for this contestant every strategy can cause some improvements. However, the exponential strategy with parameter between 3 and 7.5 gives the best results. More accurate simulations will follow for the exponential strategy later this section, in order to determine the best parameter more precisely.

Figure 3.6 shows that every strategy could increase the number of wins for a contestant with $\sigma = 0.05$. Hard thresholding has less positive impact than the other three strategies. The strategy that increases the number of wins the most is in this case again the exponential strategy. But for the contestant with prediction error equal to 0.05 the best values of the parameter lie between 2 and 5.5.

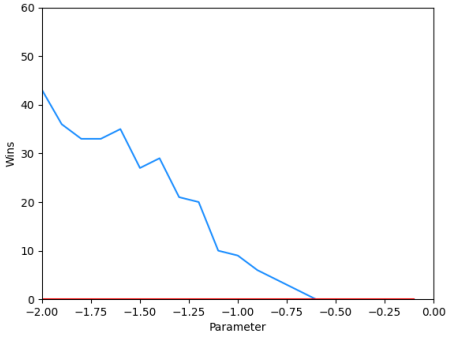
Figure 3.7 shows that using hard thresholding has negative impact on the number of wins for contestants with $\sigma = 0.1$. The best strategies



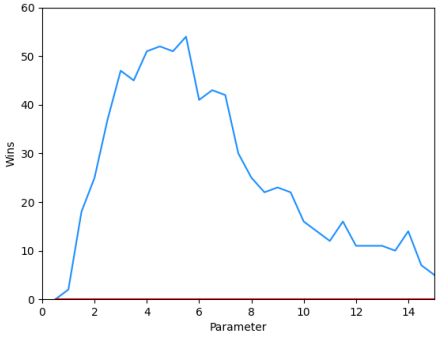
(a) Hard thresholding



(b) Soft thresholding

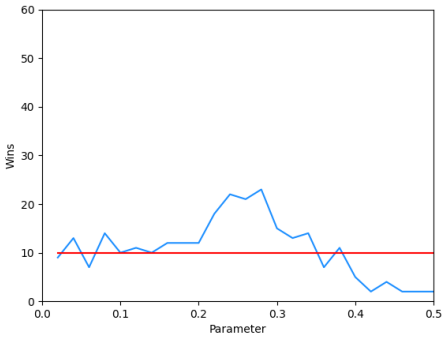


(c) Polynomial strategy

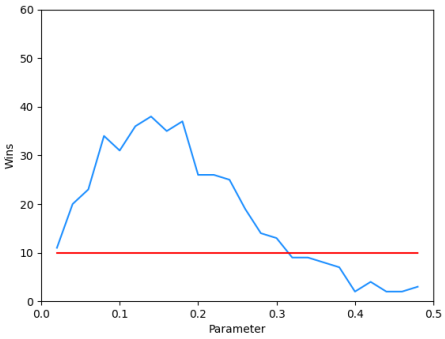


(d) Exponential strategy

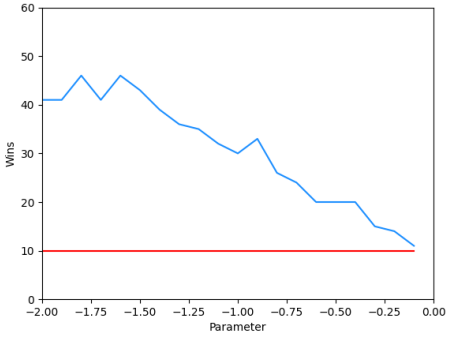
Figure 3.5: Comparison of four strategies for the contestant with $\sigma = 0$. The red horizontal line on the bottom of the plot corresponds to the number of wins of this contestant using no strategy.



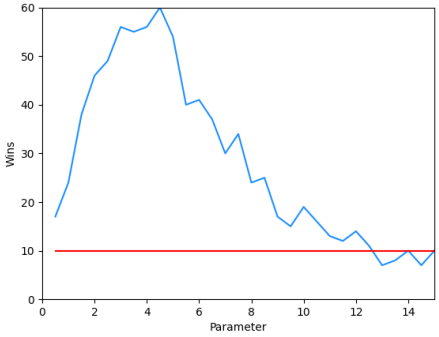
(a) Hard thresholding



(b) Soft thresholding

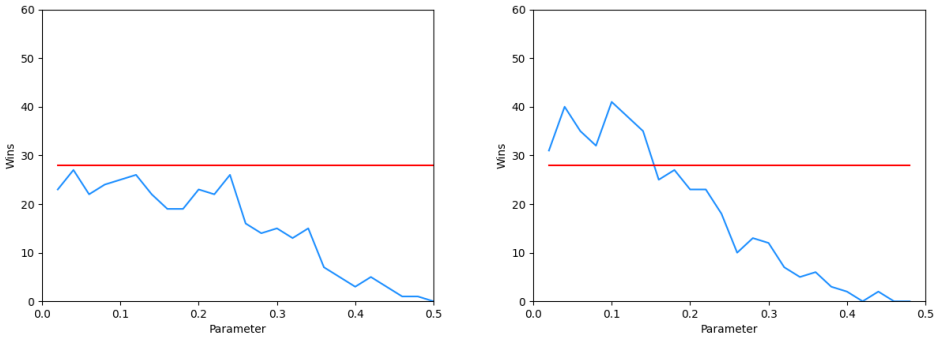


(c) Polynomial strategy



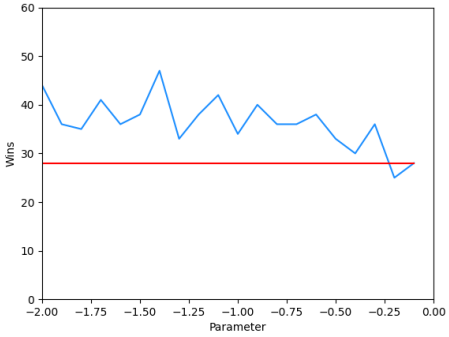
(d) Exponential strategy

Figure 3.6: Comparison of four strategies for the contestant with $\sigma = 0.05$. The red line corresponds to the number of wins of this contestant using no strategy.

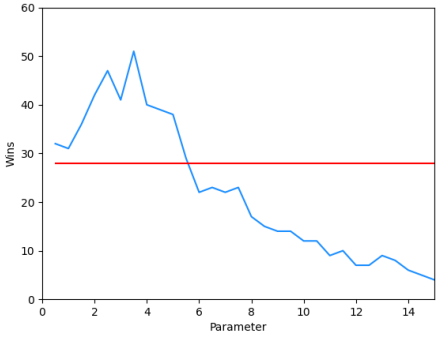


(a) Hard thresholding

(b) Soft thresholding

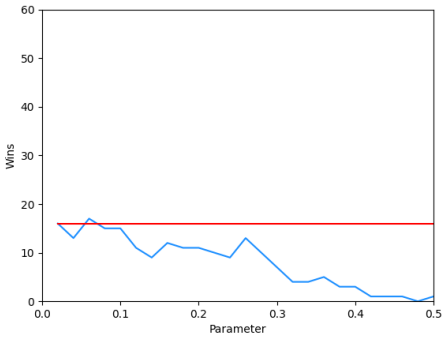


(c) Polynomial strategy

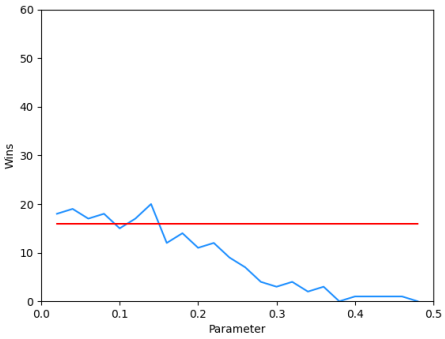


(d) Exponential strategy

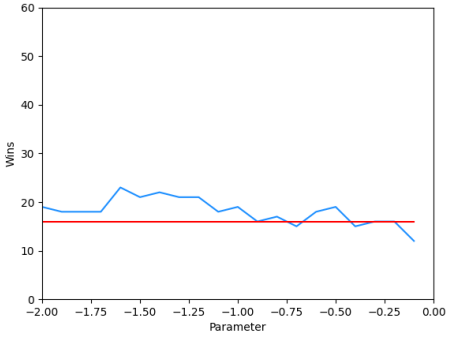
Figure 3.7: Comparison of four strategies for the contestant with $\sigma = 0.1$. The red line corresponds to the number of wins of this contestant using no strategy.



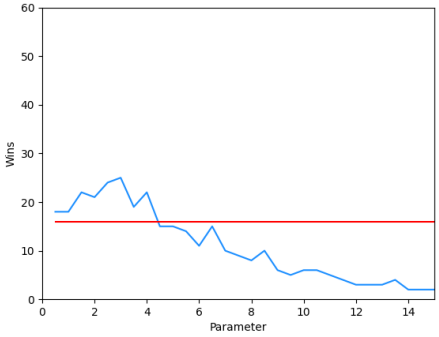
(a) Hard thresholding



(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

Figure 3.8: Comparison of four strategies for the contestant with $\sigma = 0.15$. The red line corresponds to the number of wins of this contestant using no strategy.

are polynomial and exponential. But, the highest number of victories is achieved with the exponential strategy. The best parameters of the exponential strategy for a contestant with prediction error equal to 0.1 are in the interval [2,5].

Figure 3.8 shows that for contestants with $\sigma = 0.15$ the polynomial and exponential strategies could have positive impact on the number of wins for certain values of the parameters. The polynomial strategy could increase the number of wins for parameters between -1.7 and -1. The exponential strategy has positive impact on the number of wins for parameters between 1 and 4. To determine which of these strategies have the most positive impact, more precise simulations are done later in this chapter.

Simulations are also done for contestants with larger prediction errors. Graphs for contestants with prediction errors of 0.2, 0.25 and 0.299 can be found in appendix A. From these simulations it seems that contestants with these prediction errors are probably too inaccurate forecasters to benefit from applying a strategy. There are a few peaks above the red lines for some strategies. However, these peaks probably do not indicate an efficient strategy, since they seem to be caused by noise. So, the results from the simulations indicate that contestants with σ's above 0.2 could probably not cause a significant increase of number of wins by using a strategy.

3.2.2 Polynomial regression

To determine the best parameters for the strategies, polynomial regression is used and more precise simulation are executed. For these more precise simulations 30000 prediction tournaments are done in each step. From the graphs in subsection 3.2.1 it is clear that for different σ the exponential strategy has the most positive impact on the number of wins in the interval [2, 6]. So, more precise simulations are done for the exponential strategy in the interval [2,6] with step size 0.05. For $\sigma = 0.15$ the polynomial strategy could generate more wins. To investigate this, simulations are done for the polynomial strategy on the interval [-2,0] with step size 0.04. The graphs corresponding to these simulations are shown in figure 3.9.

Polynomial regression is used to find a relation between the number of wins N and the parameter a for the polynomial strategy or b for the exponential strategy. Take for example the polynomial strategy. Then,

the polynomial regression model has the form $N = \beta_0 + \beta_1 a + \beta_2 a^2 + ... + \beta_d a^d + e$ with d the degree of the polynomial as explained by E. Ostertagová [6]. e is an error with mean 0.

The Akaike information criterion (AIC) will be used to determine for which degree the polynomial regression model will fit the data best. As described by B. Shipley [7], the AIC of a model is calculated as

$$AIC = 2k - 2\ln L$$

where L is the value received when maximizing the likelihood function of the regression model and k is the number of parameters in the model which will be estimated. In the case of polynomial regression $k = d + 1$. The polynomial regression model with the smallest AIC value fits the data the best [7]. It is not the case that increasing the degree always improves the AIC, since the $2k$-term compensates for this. In the simulations of this research the AIC is computed for polynomial regression model from degree 2 to 10. In each situation, the polynomial regression model with the lowest AIC will be used to estimate the parameter of the strategy. Computing the AIC values is done in python using the packages statsmodel and sklearn [8]. The python code for polynomial regression can be found in appendix J.

This approach of polynomial regression with AIC values is applied to the data from the simulations at the beginning of this subsection. For $\sigma = 0$ it turns out that the relation between N and the parameter of the exponential strategy b can be estimated by the polynomial $N = 12.0810b^3 - 193.9914b^2 + 968.9701b - 1010.2220$. This polynomial attains a maximum at $b = 3.969$. So, 3.969 is approximately the best parameter for the exponential strategy for a contestant with $\sigma = 0$.

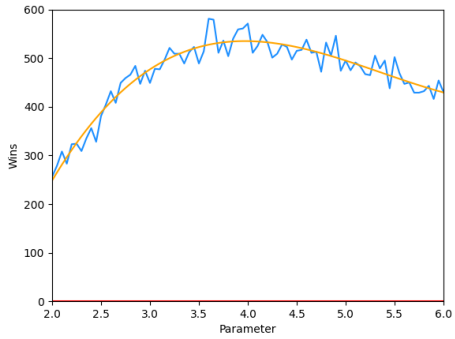
This is repeated for the exponential strategy for contestants with other values of σ. For $\sigma = 0.05$ the polynomial becomes $N = 6.5650b^3 - 111.4242b^2 + 540.1755b - 268.1084$. The best parameter for the exponential strategy becomes in this case 3.517, since it maximizes the polynomial. If $\sigma = 0.1$, then the polynomial from the regression is $N = -2.6482b^4 + 44.0896b^3 - 277.6551b^2 + 739.1666b - 268.1084$. From this it follows that the best $b = 2.641$ is the best parameter when $\sigma = 0.1$. The last case is when a contestant has a prediction error of 0.15. The relation between N and b can approximately be described by $N = -2.8538b^4 + 46.9470b^3 - 284.5427b^2 + 729.9371b - 453.4155$. Therefore, the best parameter for the exponential strategy when $\sigma = 0.15$ is 2.626.

All simulations with fit are shown in figure 3.9. Subsection 3.2.1 showed

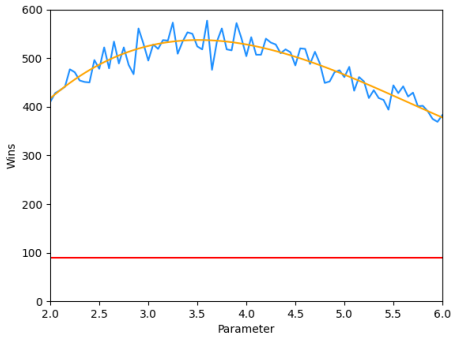
that for $\sigma = 0.15$ possibly both the exponential and polynomial strategy could work. Therefore, polynomial regression is also used on the relation between N and the parameter a of the polynomial strategy for the data from the simulations described at the beginning of this subsection. The parameter seems the have less influence as in the previous cases, but there is still a small curve in the graph, so polynomial regression is used to estimate the parameter for the polynomial strategy. This results to the polynomial $N = -56.4330a^2 - 153.2907a + 103.3757$. This function attains a maximum at $a = -1.358$. So, that is the best parameter for a contestant with prediction error equal to 0.15 who uses the polynomial strategy.

3.2.3 Results

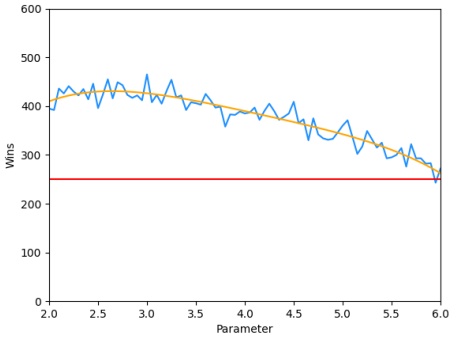
Subsection 3.2.2 made clear which strategies with which parameters could result in the most wins for different values of σ. The number of wins with these strategies after 100,000 tournaments are shown in table 3.1. Dividing these number of wins by 1,000 gives an approximation of the chance of winning as a percentage. For the most precise predictors with prediction errors around 0 the exponential strategy with parameter 3.969 is the most efficient strategy as expected. This strategy has more positive impact for lower values of σ and increases the chance of winning with about 1.9% compared to using no strategy for predictors that predict the true probabilities precisely. For $\sigma = 0.05$ and $\sigma = 0.1$ the best strategy is the exponential strategy with respectively parameters 3.517 and 2.641. These strategies cause increases in number of wins of about 1.5 % for $\sigma = 0.05$ and 0.6% for $\sigma = 0.1$. For $\sigma = 0.15$ the exponential and polynomial strategy were investigated, since the differences in the previous steps were not clear enough to draw conclusions. However, table 3.1 shows that the polynomial strategy is not better than the exponential strategy. This holds for all σ for which simulations were done. The idea from subsection 3.2.2 was that the exponential strategy with parameter 2.626 would be a bit better than the same strategy with parameter 2.641 for contestants with prediction errors around 0.15. This turned out to be not the case. The best strategy for contestants with $\sigma = 0.15$ is the exponential strategy with parameter 2.641. This is probably caused by some inaccuracies in the simulations, since the results are estimations and these parameters lay close to each other. From table 3.1 it became clear that there is some positive impact on the number of wins for contestants with $\sigma = 0.2$ or $\sigma = 0.25$, if they use the exponential strategy with parameters 2.626 or 2.641, respectively.



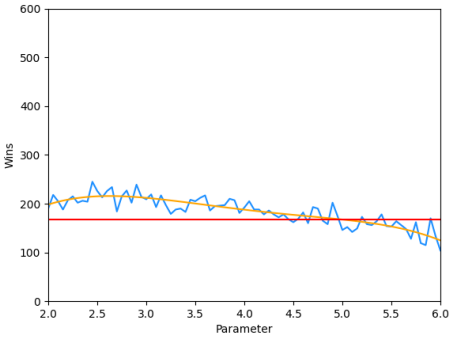
(a) Exponential $\sigma = 0$



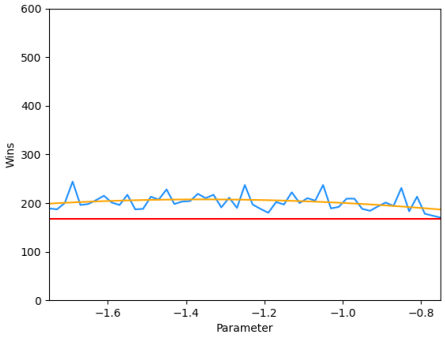
(b) Exponential $\sigma = 0.05$



(c) Exponential $\sigma = 0.1$



(d) Exponential $\sigma = 0.15$



(e) Polynomial $\sigma = 0.15$

Figure 3.9: The number of wins after 30,000 tournaments of the only contestant using a strategy for different parameters and different values of σ. The red line corresponds to the number of wins of this contestant when he uses no strategy. The orange curve is the polynomial fitted by the regression. In every case, the degree of the polynomial is chosen between 2 and 10 based on AIC values.

σ	No Strategy	Exponential 3.969	Exponential 3.517	Exponential 2.641	Exponential 2.626	Polynomial -1.358
0	0	**1874**	1754	1421	1277	691
0.05	318	1722	**1783**	1683	1674	1317
0.1	896	1333	1346	**1493**	1422	1346
0.15	524	650	636	**726**	713	634
0.2	205	242	237	245	**260**	245
0.25	65	44	56	**68**	62	59
0.299	**24**	8	11	13	18	13

Table 3.1: Number of wins for different σ's and different strategies after competing 100,000 times in a prediction tournament.

Chapter 4

Situations in which opponents use strategies

In chapter 3, the exponential strategy with different parameters was recommended for different contestants with different prediction errors in the case that all opponents use no strategy. Which parameter was best depends on the contestant's prediction error. But, if it becomes public that these are the best strategies, then other contestants will also use the exponential strategy with a certain parameter. Therefore, this chapter investigates which strategies are the best if some opponents uses the exponential strategy. Firstly, two strategies for opponents will be analysed. This are the exponential strategies with parameters 3.969 and 2.641. For both strategies two situations are distinguished, all opponents use the exponential strategy with the specified parameter and half of the opponents use the exponential strategy with the specified parameter and half of the opponents uses no strategy. For these analyses, a new strategy will be introduced. In this strategy it is random for every question whether the contestant uses a strategy or whether she do not. At the end of this chapter, the settings will be analysed in which all opponents or half of the opponents use such a random strategy.

4.1 Half of opponents uses exponential strategy with parameter 3.969

In this simulations half of the contestants use the exponential strategy with parameter 3.969 and half of the contestants use no additional strat-

egy. The exponential strategy with parameter 3.969 has in the situation with opponents who use no strategy most impact on contestants with small σ's. For different values of σ, the contestant with that σ changes his strategy for one of the strategies from section 3.1. The first simulations in this section contain 10,000 tournaments in each step.

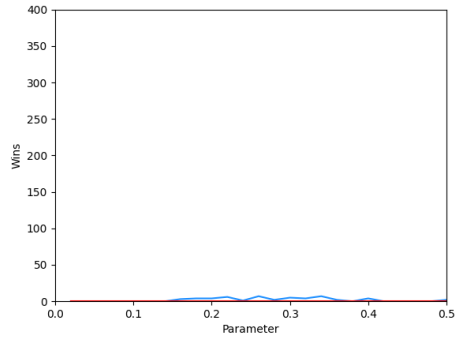
Figure 4.1 (a-d) show no increase in number of wins when the contestant with $\sigma = 0$ uses one of the four strategies from 3.1. Therefore, a new strategy will be introduced to create an increase in the number of wins.

This new strategy is a variation on the exponential strategy and is called the random exponential strategy. Just like the exponential strategy the random exponential strategy depends on a parameter b. The random exponential strategy works as follows. For every question it will be arbitrary determined with probabilities 0.5 for each option whether the contestant uses the exponential strategy with parameter b or whether the contestant uses no additional strategy. Graphs in appendix B show that this strategy has no larger effect on the number wins than the exponential strategy when all opponents use no strategy. In the setting where half of the opponents uses the exponential strategy with parameter 3.969, the random exponential strategy creates an increase in the number of wins when $\sigma = 0$ as can be seen in figure 4.1 (e). The best parameter lies somewhere between 6 and 15.

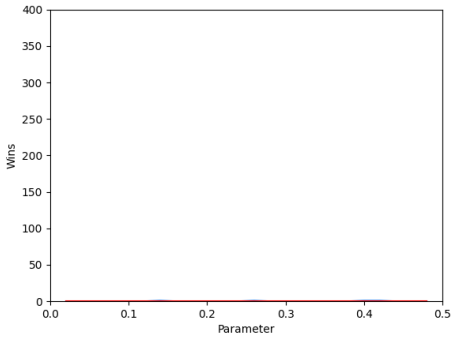
Figure 4.2 shows that also for contestants with prediction error 0.05 the random exponential strategy is the only strategy that could increase the number of wins of the contestant. The most efficient parameters lie between 3 and 7. Contestants with $\sigma = 0.1$ can increase their chance of winning by using a polynomial or exponential strategy as can be seen in figure 4.3. The best parameter for the polynomial strategy lies somewhere between -1.3 and -0.2. The best parameter for the exponential strategy can be found in the interval [0, 3].

To determine the best values of the parameter, polynomial regression is used in the same way as in subsection 3.2.2. Figure 4.4 shows the number of wins obtained by different strategies with fitted regression polynomials. The formula of the fitted polynomials and the recommended parameters after using polynomial regression can be found in table 4.1.

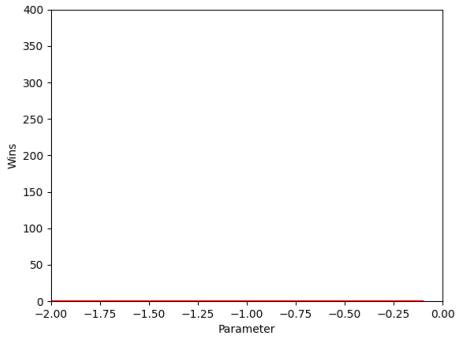
The number of wins after 100,000 tournaments for different strategies and σ's is presented in table 4.2. The chance of winning is for all σ at most 0.4% higher for contestants with strategy than for contestants who do not use a strategy. The best strategy for $\sigma = 0$ is the random exponential



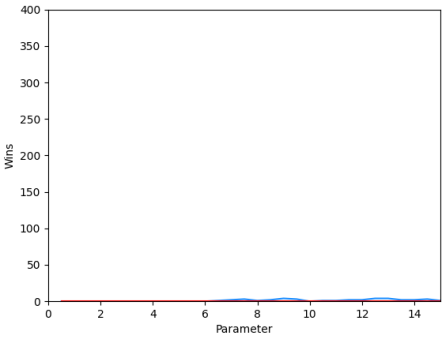
(a) Hard thresholding



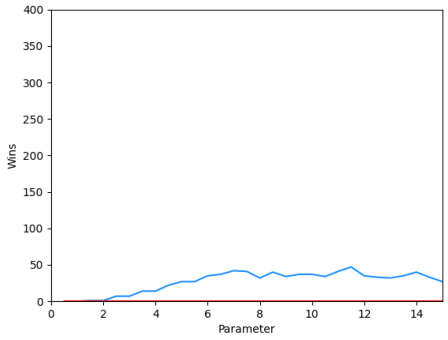
(b) Soft thresholding



(c) Polynomial strategy

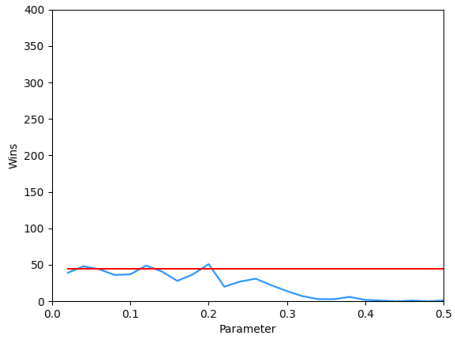


(d) Exponential strategy

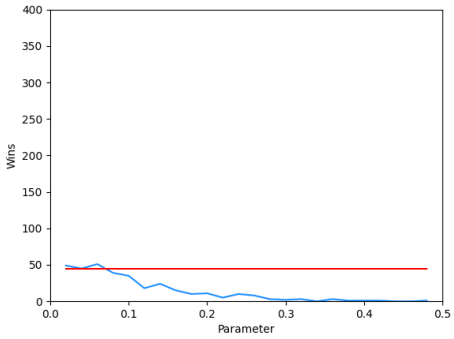


(e) Random exponential strategy

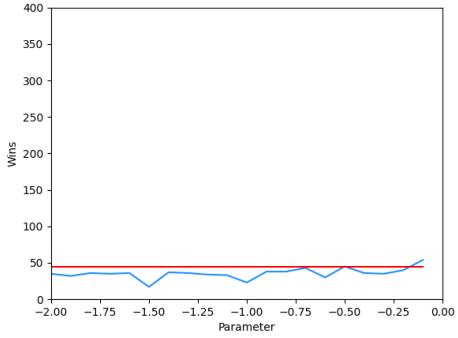
Figure 4.1: Comparison of five strategies for the contestant with $\sigma = 0$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs are after 10,000 tournaments.



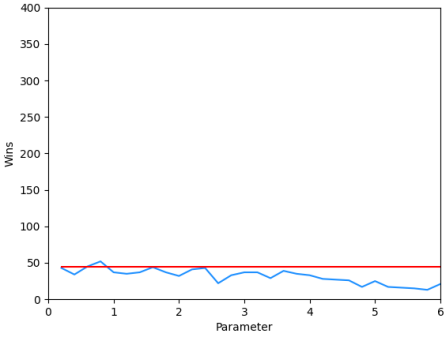
(a) Hard thresholding



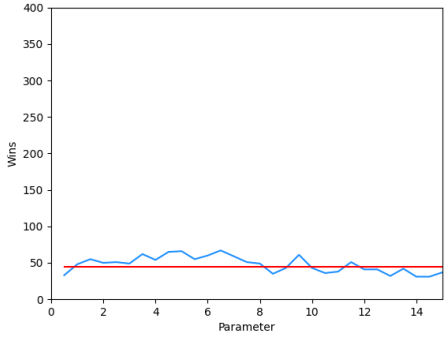
(b) Soft thresholding



(c) Polynomial strategy

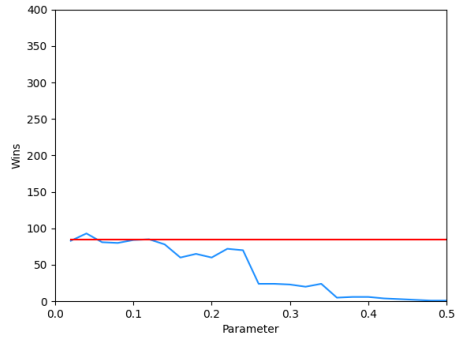


(d) Exponential strategy

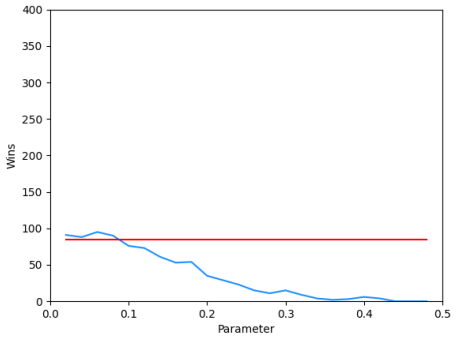


(e) Random exponential strategy

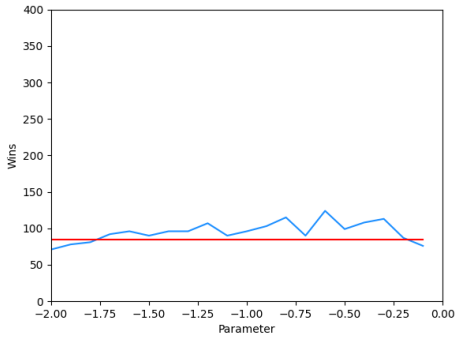
Figure 4.2: Comparison of five strategies for the contestant with $\sigma = 0.05$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



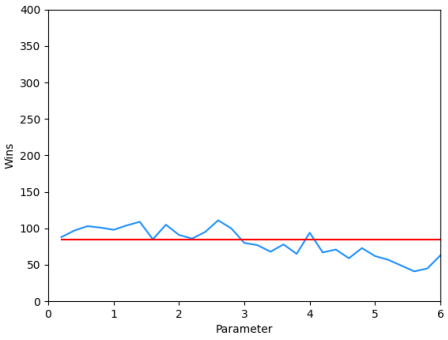
(a) Hard thresholding



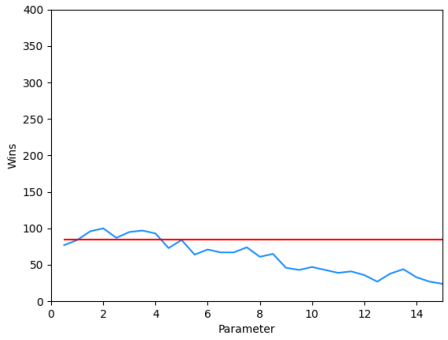
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy



(e) Random exponential strategy

Figure 4.3: Comparison of five strategies for the contestant with $\sigma = 0.1$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

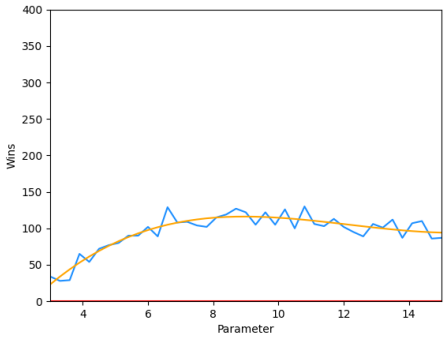
σ	Strategy	Fitted polynomial	Best parameter
0	Random exponential	$N = 0.1685b^3 - 6.147b^2 + 69.5595b - 134.804$	8.956
0.05	Random exponential	$N = -3.0775b^2 + 31.6535b + 96.1123$	5.143
0.1	Polynomial	$N = -54.6198a^2 - 88.5971a + 265.5559$	-0.811
0.1	Exponential	$N = 3.234b^3 - 31.3638b^2 + 62.309b + 265.4513$	1.226

Table 4.1: Fitted polynomial and best parameter for different strategies and different values of σ after using polynomial regression. In the polynomials, N is the number of wins, a is the parameter for the polynomial strategy and b is the parameter for the exponential strategy.

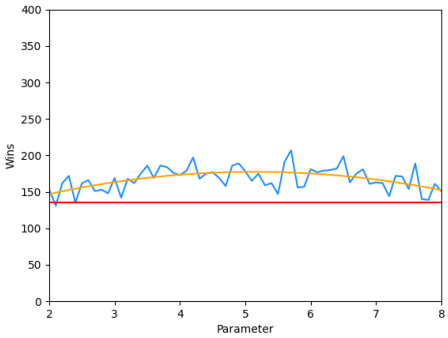
strategy with parameter equal to 8.956. For $\sigma = 0.05$ the best strategy is the random exponential strategy with parameter equal to 5.143. The polynomial strategy with parameter -0.811 is recommended for $\sigma = 0.1$. There seems to be no significant difference in the number of wins for a contestant with $\sigma = 0.15$ between using the polynomial strategy with parameter -0.811 or the exponential strategy with parameter 1.226. The simulations indicate that both strategies could lead to a higher chance of winning than using no strategy. The strategy that can result in the most victories for contestants with prediction errors around 0.2 or 0.25 seems to be the exponential strategy with parameter 1.226. For the most inaccurate predictors it is better to use no extra strategy.

σ	No Strategy	Exponential 3.969	Random exponential 8.956	Random exponential 5.143	Polynomial -0.811	Exponential 1.226
0	0	0	**379**	296	0	0
0.05	426	294	485	**625**	400	379
0.1	897	695	554	786	**1022**	960
0.15	558	502	321	508	**607**	**608**
0.2	220	206	144	198	237	**268**
0.25	69	61	38	65	66	**72**
0.299	**24**	20	9	14	13	12

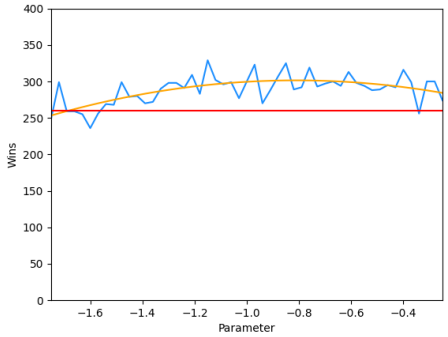
Table 4.2: Number of wins for different σ's and different strategies after competing 100 000 times in a prediction tournament where half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969.



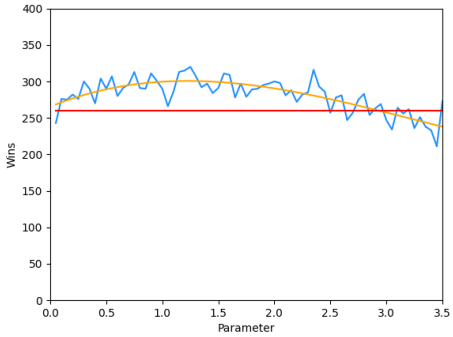
(a) Random exponential strategy, $\sigma = 0$



(b) Random exponential strategy, $\sigma = 0.05$



(c) Polynomial strategy, $\sigma = 0.1$



(d) Exponential strategy, $\sigma = 0.1$

Figure 4.4: The best strategies for different values of σ if half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10000 tournaments.

4.2 All opponents use exponential strategy with parameter 3.969

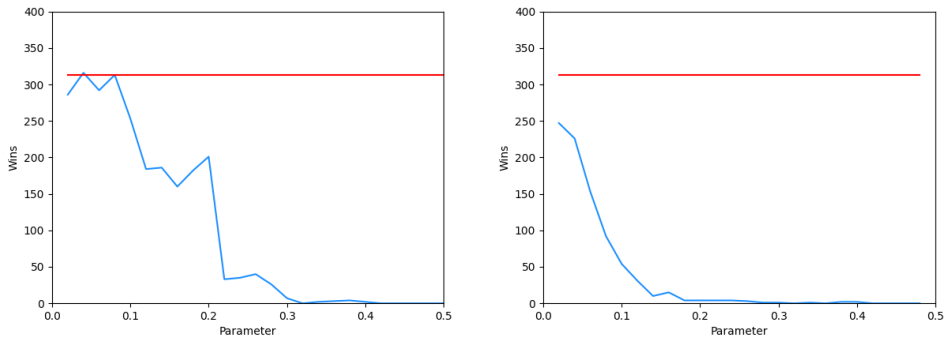
Now prediction tournaments will be studied in which a contestant has 299 opponents that use the exponential strategy with parameter 3.969. If a contestant is the only one who uses this strategy, then this strategy has more positive impact on the number of wins for low values of σ.

All five different strategies have negative impact on the number of wins in this situation compared to predicting the probabilities that the contestant expects. This is showed for $\sigma = 0.05$ in figure 4.5. Graphs for other values of σ can be found in Appendix C.

Table 4.3 shows the number of wins for different strategies and σ's after 100,000 prediction tournaments. For values of σ around 0.05 or closer predicting the probabilities that the contestant thinks that are the real probabilities increases the winning chance with around 2.9% compared to using the exponential strategy with parameter 3.969. So, if all the opponents use the exponential strategy with parameter 3.969, then it is preferred to use no additional strategy. This holds for all σ's between 0 and 0.3.

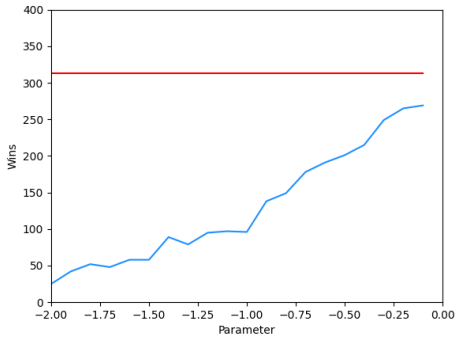
σ	Exponential 3.969	No Strategy
0	0	**2891**
0.05	202	**3084**
0.1	762	**2285**
0.15	607	**1043**
0.2	317	**422**
0.25	76	**113**
0.299	19	**40**

Table 4.3: Number of wins for different σ's and different strategies after competing 100 000 times in a prediction tournament where all opponents are using the exponential strategy with parameter 3.969.

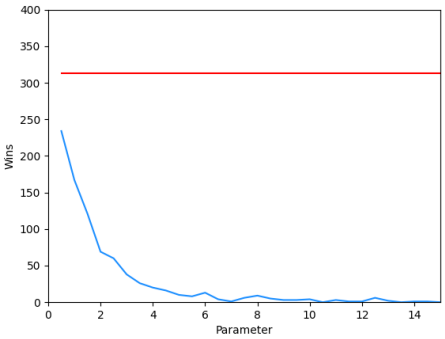


(a) Hard thresholding

(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

Figure 4.5: Comparison of four strategies for the contestant with $\sigma = 0.05$. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

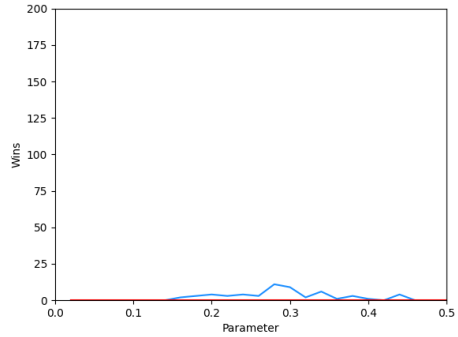
4.3 Half of opponents uses exponential strategy with parameter 2.641

In the previous sections situations were analysed in which opponents use a strategy that has the most benefits for the most accurate predictors. Chapter 3 also introduced strategies which have the most positive impact on less accurate predictors. An example of such a strategy is the exponential strategy with parameter 2.641. In the case that all opponents use no strategy, this strategy has the most positive impact on predictors in the middle of the ranking. In this section and the next section, situations will be discussed in which opponents use the exponential strategy with parameter 2.641. This section discusses a situation in which half of the opponents use the exponential strategy with parameter equal to 2.641 and half of the opponents do not use a strategy.

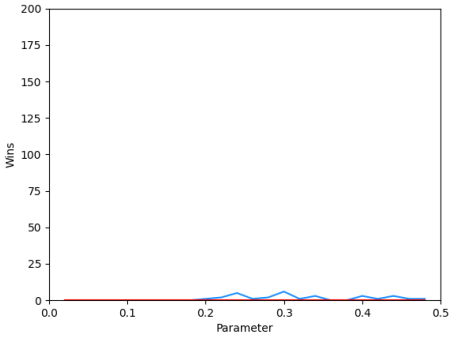
Figure 4.6 shows that the random exponential strategy is the best strategy for contestants with $\sigma = 0$. This phenomenon was also evident in the case that half of the opponents uses the random exponential strategy with parameter 3.969. The best value of the parameter lies somewhere between 5 and 15. Also for contestants with prediction error 0.05 the random exponential strategy is the strategy with the most positive impact on the number of wins. In this case the best parameter can be found in the interval [4,9], the corresponding graph is shown in figure 4.7.

Figure 4.8 shows the number of wins realised with the five strategies for a contestant with $\sigma = 0.1$. The peaks seen for hard thresholding and soft thresholding seemed to be caused by noise. The other three strategies could potentially increase the number of wins. The best parameter lies between 0 and 4 for the exponential and random exponential strategy. For the polynomial strategy, all parameters in the interval [-2, 0] could work.

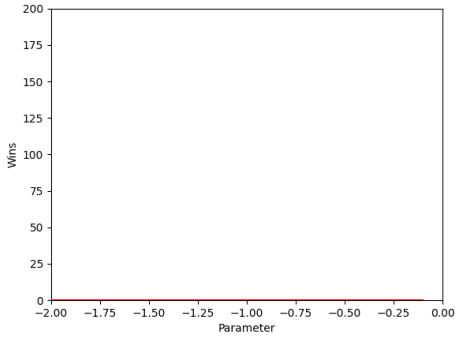
Contestants with $\sigma \geq 0.15$ do not seem to receive many benefits from using strategies. Graphs showing this can be found in appendix E.



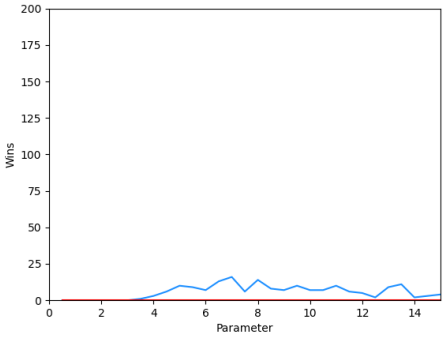
(a) Hard thresholding



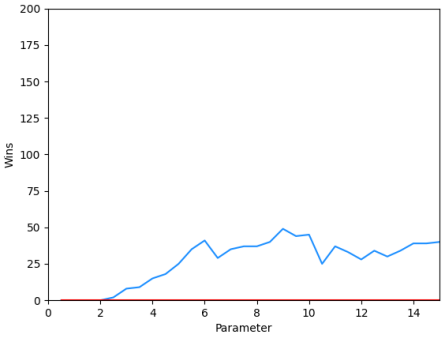
(b) Soft thresholding



(c) Polynomial strategy

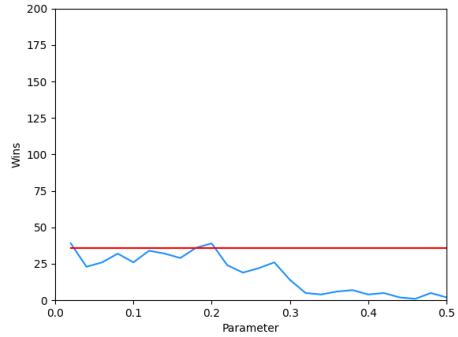


(d) Exponential strategy

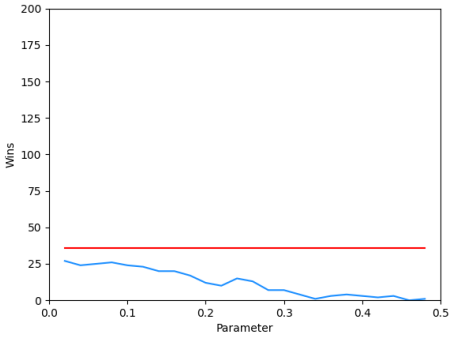


(e) Random exponential strategy

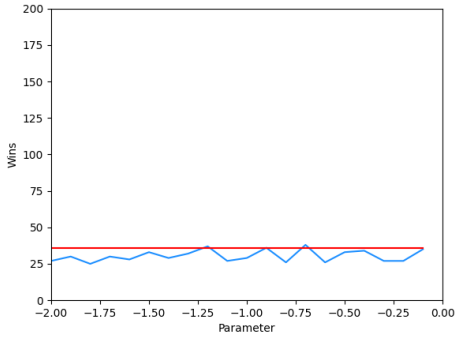
Figure 4.6: Comparison of five strategies for the contestant with $\sigma = 0$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



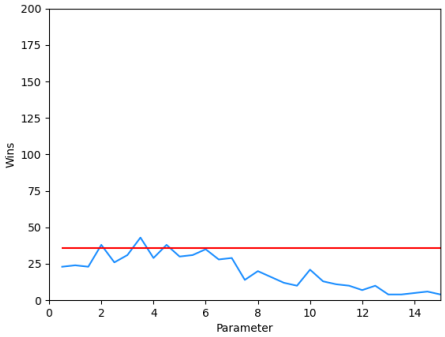
(a) Hard thresholding



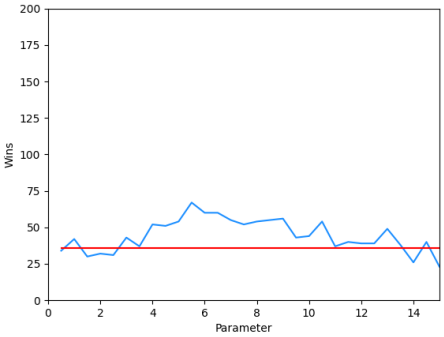
(b) Soft thresholding



(c) Polynomial strategy

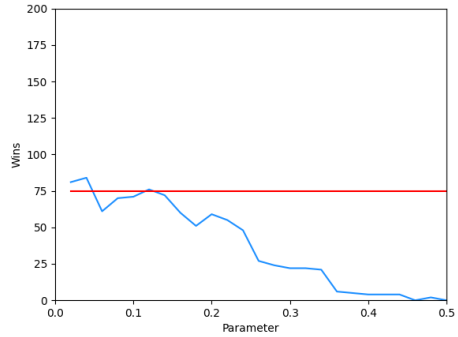


(d) Exponential strategy

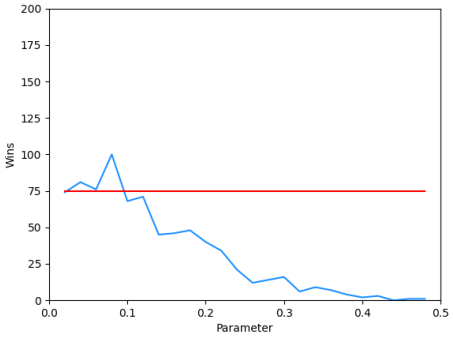


(e) Random exponential strategy

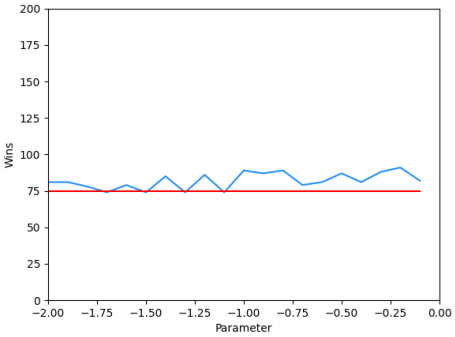
Figure 4.7: Comparison of five strategies for the contestant with $\sigma = 0.05$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



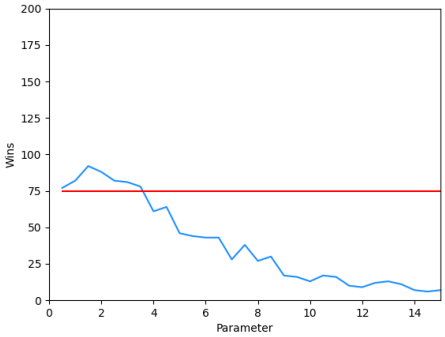
(a) Hard thresholding



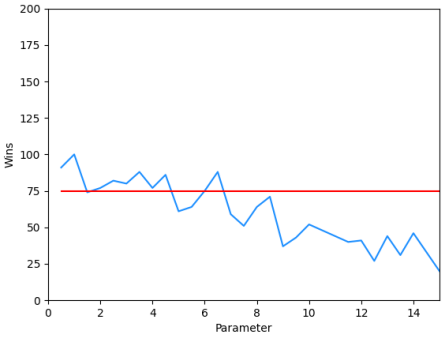
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

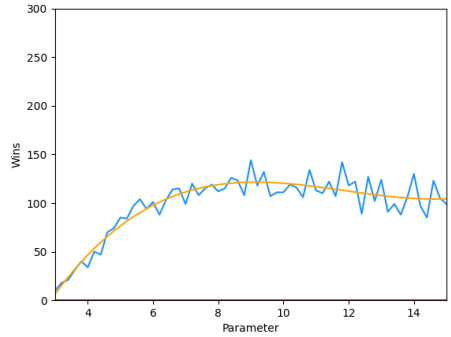


(e) Random exponential strategy

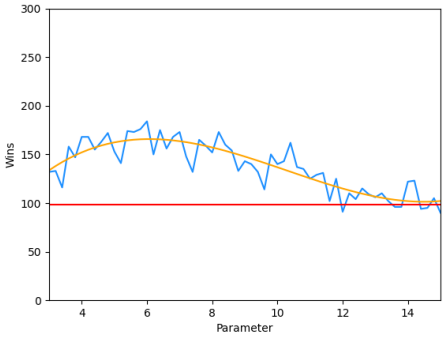
Figure 4.8: Comparison of five strategies for the contestant with $\sigma = 0.1$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

The first simulations gave an impression of which strategy could lead to more wins for contestants with $\sigma \leq 0.1$. To determine which parameters are the best for different strategies and values of σ, polynomial regression is used in the same way as described in subsection 3.2.2. Therefore, more precise simulations are done with 30,000 tournaments in each step. For $\sigma = 0$ and $\sigma = 0.05$, the number of wins for the random exponential strategy with parameters in the interval [3, 15] is determined with step size 0.2. For $\sigma = 0.1$ three different strategies are investigated, since there were no significant differences after the first simulations. The random exponential and exponential strategies will be simulated in the interval [0, 4] with step size 0.1. The polynomial strategy will be simulated in the interval [-2, 0] with step size 0.05. The graphs corresponding to this simulation with fitted polynomials are shown in figure 4.9. Table 4.4 shows for each strategy which polynomial is fitted and what the best parameter is.

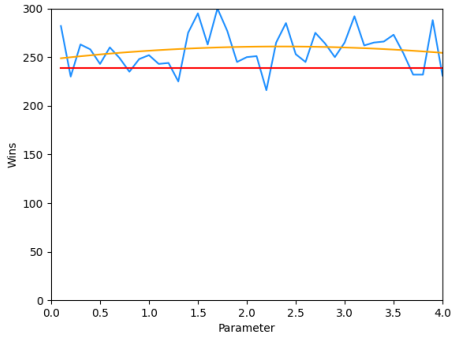
For the strategies with the best parameters from table 4.4 and seven different values of σ, the number of wins is determined after 100,000 tournaments. These results are summarized in table 4.5. For $\sigma = 0$ and $\sigma = 0.05$ the recommended strategies are the polynomial strategies with parameters 9.141 and 6.074 respectively. For precise predictors the increase in number of wins compared to using no strategy is about 0.4%. This positive impact of a strategy decreases as σ increases. There were three strategies that could work for contestants with prediction errors around 0.1. The random exponential strategy with parameter 2.347 and the polynomial strategy with parameter -0.854 have approximately the same effect on the number of wins of contestants with $\sigma = 0.1$. However, in reality the random exponential strategy with parameter 2.347 is preferred. Since that strategy gives more wins for $\sigma = 0.05$ and $\sigma = 0.15$ and is therefore more robust with respect to the value of σ. The random exponential strategy with parameter 2.347 is also recommended for contestants with $\sigma \in [0.15, 0.25]$. Since, this strategy also has the largest positive impact on contestant with high σ.



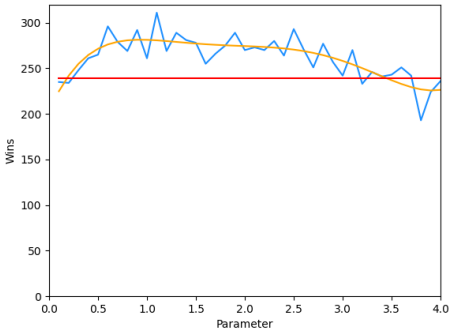
(a) Random exponential ($\sigma = 0$)



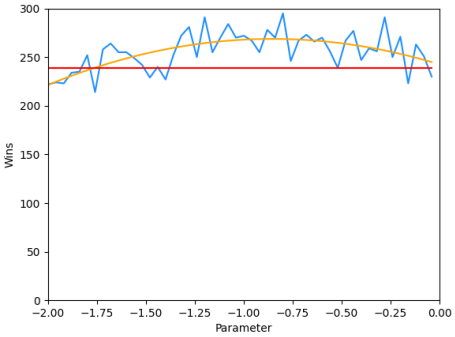
(b) Random exponential ($\sigma = 0.05$)



(c) Random exponential ($\sigma = 0.1$)



(d) Exponential ($\sigma = 0.1$)



(e) Polynomial ($\sigma = 0.1$)

Figure 4.9: Regression used for different strategies for different values of σ after competing 30,000 in a prediction tournament when half of the opponents use the exponential strategy with parameter 2.641 and half of the opponents do not use a strategy. The red line corresponds to the number of wins of this contestant using no strategy.

σ	Strategy	Fitted polynomial	Best parameter
0	Random exponential	$N = 0.2122b^3 - 7.5628b^2 + 85.0705b - 186.3091$	9.141
0.05	Random exponential	$N = 0.2162b^3 - 6.6688b^2 + 57.0854b + 16.5985$	6.074
0.1	Random exponential	$N = -2.3983b^2 + 11.2561b + 246.7246$	2.347
0.1	Exponential	$N = 3.2982b^5 - 36.389b^4 + 148.4937b^3 - 282.7789b^2$ $+245.6282b + 202.986$	0.933
0.1	Polynomial	$N = -35.9791a^2 - 61.4369a + 242.4952$	-0.854

Table 4.4: Fitted polynomial and best parameter for different strategies and different values of σ after using polynomial regression. In the polynomials, N is the number of wins, a is the parameter for the polynomial strategy and b is the parameter for the exponential strategy.

σ	No Strategy	Exponential 2.641	Random exponential 9.141	Random exponential 6.074	Random exponential 2.347	Exponential 0.933	Polynomial -0.854
0	0	0	**444**	350	14	0	0
0.05	276	391	467	**598**	445	292	330
0.1	804	864	522	663	**887**	836	**888**
0.15	474	548	313	480	**608**	536	581
0.2	207	221	155	170	**241**	216	236
0.25	64	64	38	52	**71**	58	63
0.299	**18**	15	12	17	17	11	17

Table 4.5: Number of wins for different σ's and different strategies after competing 100 000 times in a prediction tournament where half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641.

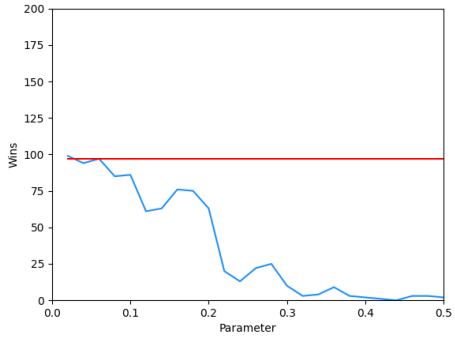
4.4 All opponents use exponential strategy with parameter 2.641

In this section, the setting in which all opponents use the exponential strategy with parameter 2.641 will be analysed. This strategy has the most positive impact on the number of wins if the contestant has a prediction error in the middle of the interval around 0.1 and 0.15.

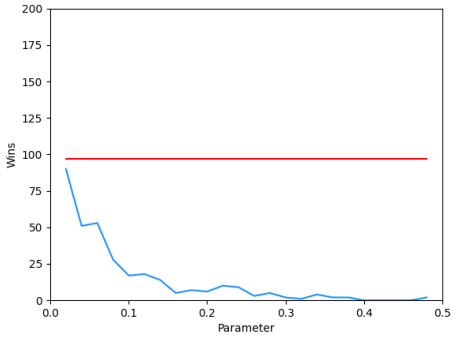
For a contestant with $\sigma = 0.05$, the best choice is to use no strategy as can be seen in figure 4.10. This is the case for all values of σ. Graphs of simulations for contestants with other values of σ can be found in appendix F. The number of wins that contestants with different σ's achieve after 100,000 tournaments is shown in table 4.6. When comparing the settings in which all opponents use the exponential strategy with parameter 3.969 or 2.641, it stands out that the differences in number of wins between using the same strategy as the opponents and using no strategy are larger in the case that the opponents use the exponential strategy with parameter 3.969.

σ	Exponential 2.641	No Strategy
0	0	**328**
0.05	249	**974**
0.1	856	**1202**
0.15	607	**658**
0.2	261	**273**
0.25	82	**90**
0.299	22	**36**

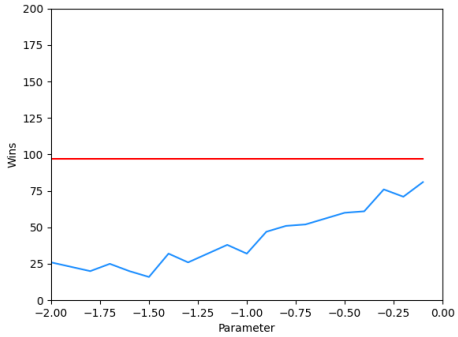
Table 4.6: Number of wins for different σ's and different strategies after competing 100,000 times in a prediction tournament where all opponents are using the exponential strategy with parameter 2.641.



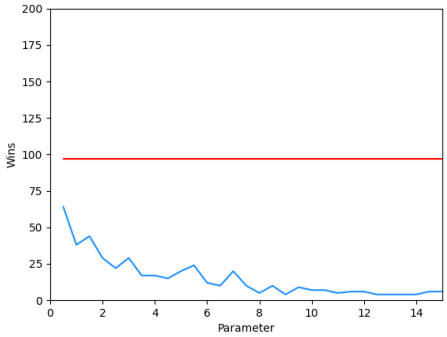
(a) Hard thresholding



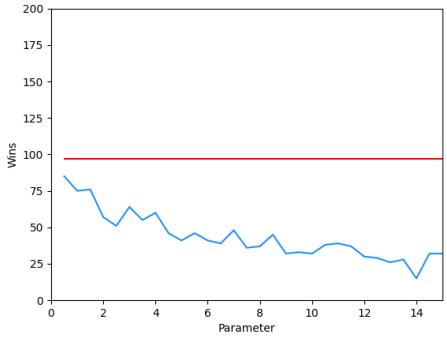
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy



(e) Random exponential strategy

Figure 4.10: Comparison of five strategies for the contestant with $\sigma = 0.05$ in the case that all opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

4.5 Half of opponents uses random exponential strategy with parameter 8.956

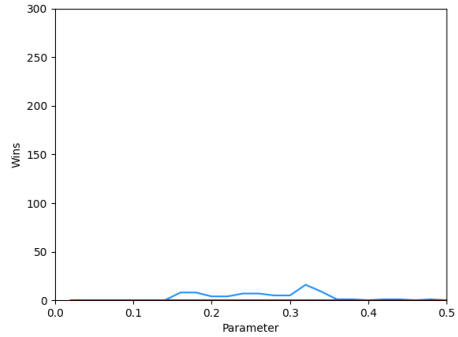
In the previous sections, situations were analysed in which all opponents or half of the opponents use the exponential strategy with a certain parameter. This analyses have resulted in recommendations for contestants. The strategies used by opponents in the previous sections were the same for all questions. But, what happens if some of the opponents use a random strategy? In this section simulations are done in which half of the opponents use the random exponential strategy with parameter 8.956. This strategy is probably the best strategy for a contestant with $\sigma = 0$ in the case that half of the opponents use the exponential strategy with parameter 3.969 and half of the opponents use no strategy.

Firstly, simulations are done in which the number of wins of a specific contestant using a specific strategy after 10,000 tournaments is determined for different values of the parameters. Figure 4.11 shows that the random exponential strategy could result in the highest number of wins of a contestant with prediction error equal to 0. It is important to choose a correct parameter to achieve the largest number of wins. The best parameter for the random exponential strategy in this case seems to be between 6 and 15 according to figure 4.11.

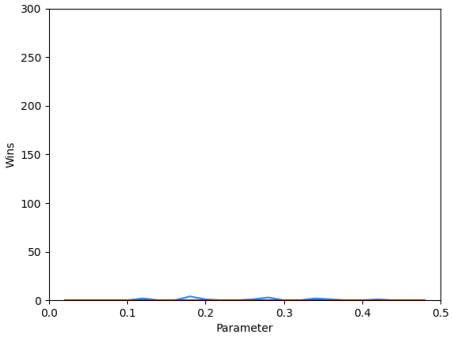
The random exponential strategy seems to be the best for a contestant with $\sigma = 0.05$ based on figure 4.12. The parameter that has the most positive impact on the number of wins lies somewhere between 2 and 10. Figure 4.13 indicates that using one of the strategies has no significant positive impact on the number of wins for contestants with prediction errors around 0.1.

Figure 4.14 shows that different strategies could increase the number of wins for contestants with $\sigma = 0.15$. The three strategies that are most likely to do so are the polynomial, exponential and random exponential strategies. The graph for the polynomial strategy in figure 4.14 (c) do not give enough information to indicate which parameter is the best. To find the best parameter for the polynomial strategy more precise simulations are needed. For the exponential and random exponential strategy the best parameter can be found between 0 and 5.

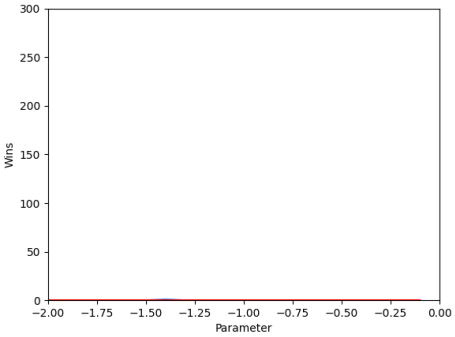
Using a strategy does not seem to cause a significant increase in the number of wins for contestants with $\sigma \geq 0.2$. Figures illustrating this can be found in appendix G.



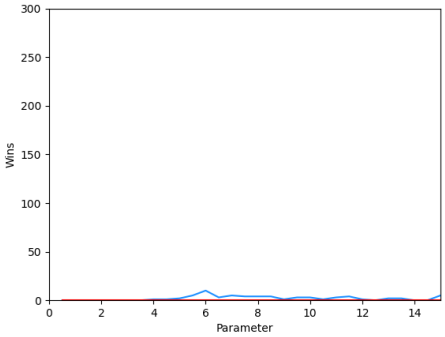
(a) Hard thresholding



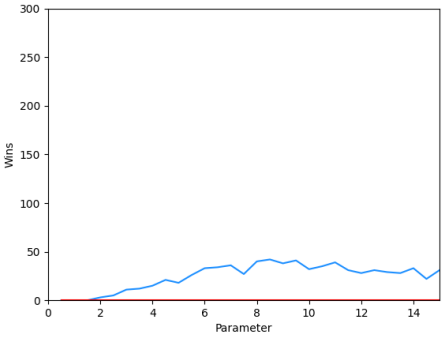
(b) Soft thresholding



(c) Polynomial strategy

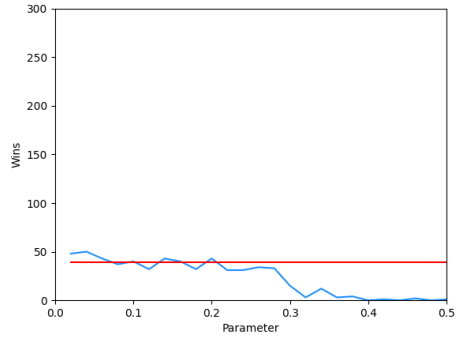


(d) Exponential strategy

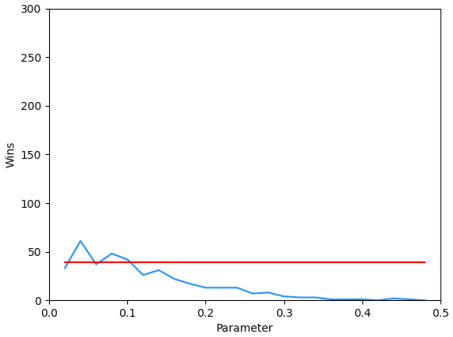


(e) Random exponential strategy

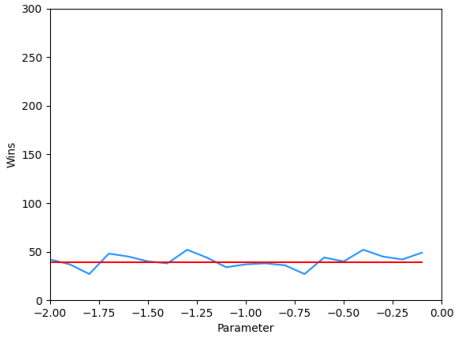
Figure 4.11: Comparison of five strategies for the contestant with $\sigma = 0$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



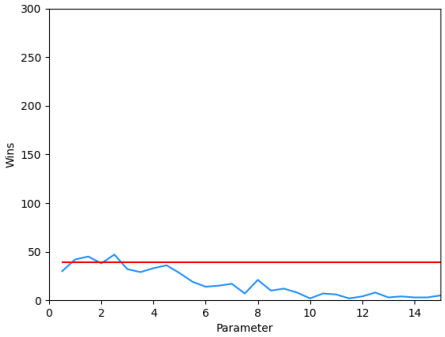
(a) Hard thresholding



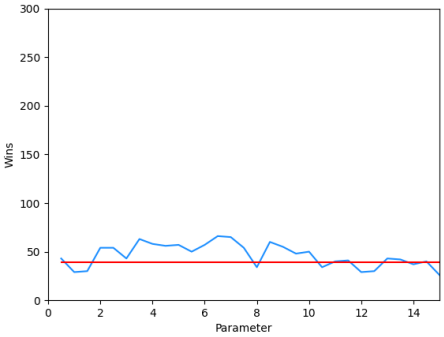
(b) Soft thresholding



(c) Polynomial strategy

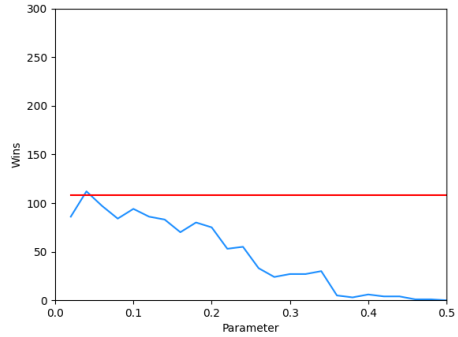


(d) Exponential strategy

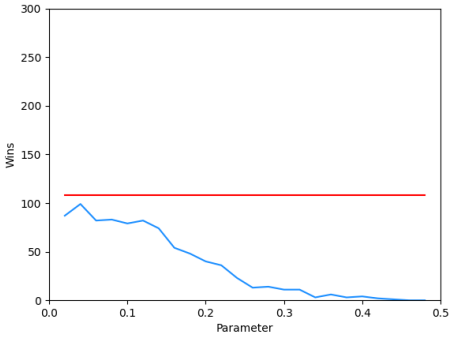


(e) Random exponential strategy

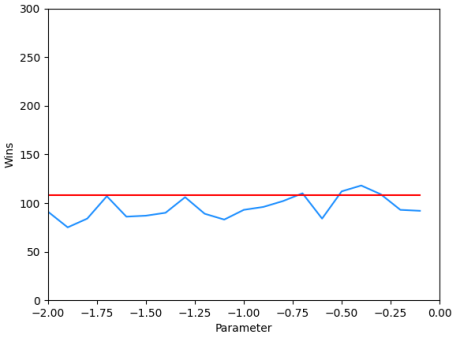
Figure 4.12: Comparison of five strategies for the contestant with $\sigma = 0.05$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



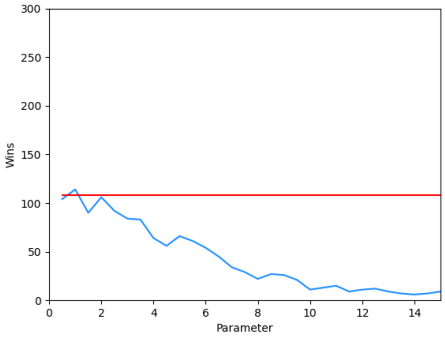
(a) Hard thresholding



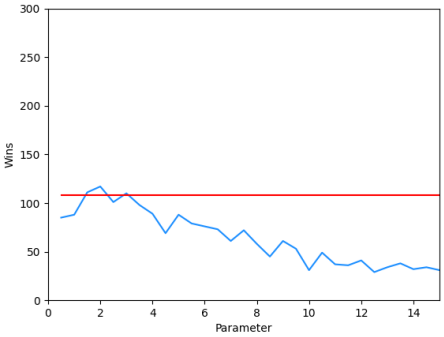
(b) Soft thresholding



(c) Polynomial strategy

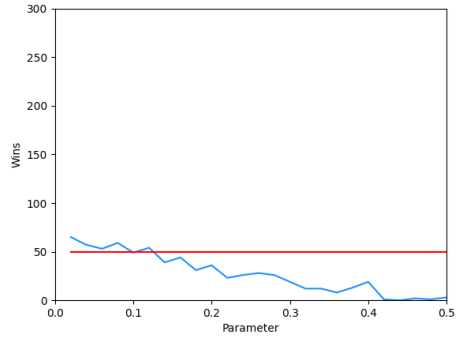


(d) Exponential strategy

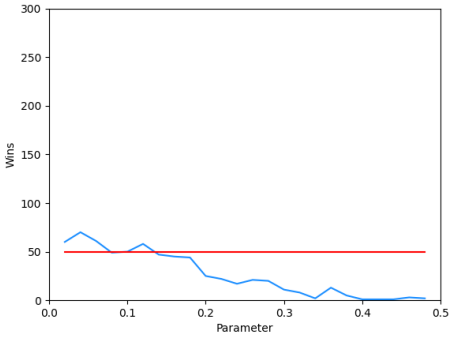


(e) Random exponential strategy

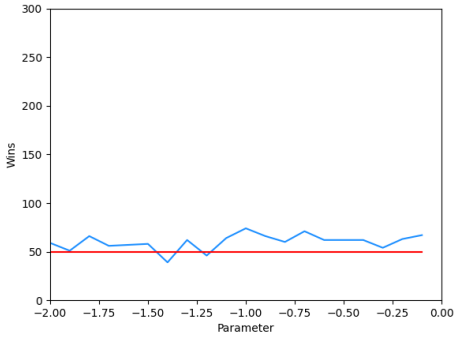
Figure 4.13: Comparison of five strategies for the contestant with $\sigma = 0.1$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



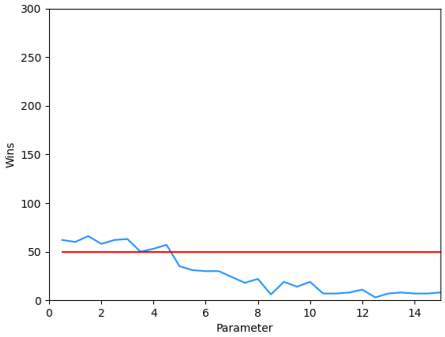
(a) Hard thresholding



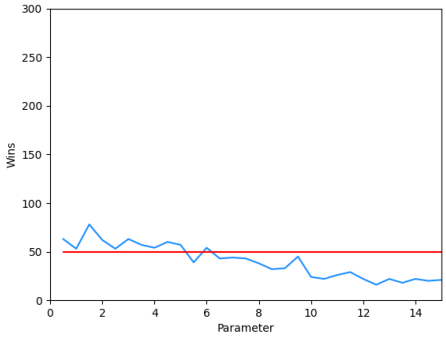
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

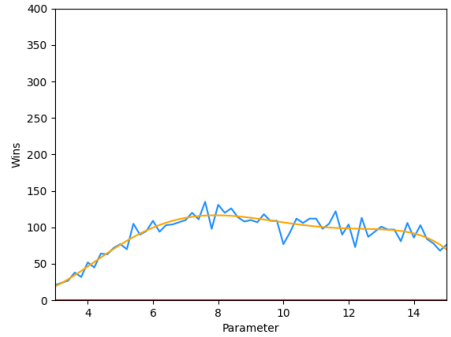


(e) Random exponential strategy

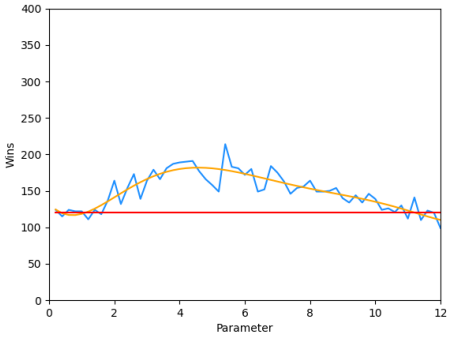
Figure 4.14: Comparison of five strategies for the contestant with $\sigma = 0.15$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

More accurate simulations are performed to find the best parameters for the preferred strategies for different contestants with different prediction errors. In these simulations, the number of wins after 30,000 prediction tournaments is determined for different parameters of the chosen strategies. To find the best parameter, polynomial regression is used in the same way as in sections 3.2.2, 4.1 and 4.3. The graphs of the number of wins per parameter in different situations with fitted polynomials can be found in figure 4.15. The formula's of these fitted polynomials and the best parameters according to the simulations are shown for various situations in table 4.7.

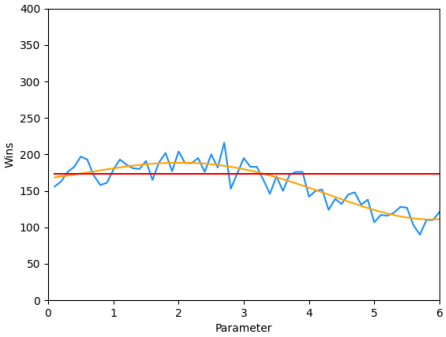
Finally, simulations are done in which the number of wins is determined after 100,000 prediction tournaments. The results for various contestants with different prediction errors and strategies can be found in table 4.8. As expected, the random exponential strategy with parameter 7.914 is the best for contestants with $\sigma = 0$ based on the number of wins. This strategy results in a chance of winning of approximately 0.36%. This is better than a chance of winning of 0% when this contestant does not use a strategy. The random exponential strategy with parameter 4.607 seems to result in the largest increase of number of wins for a contestant with a prediction error of 0.05. The simulations at the beginning of this section indicate that neither of the five strategies could increase the number of wins for a contestant with $\sigma = 0.1$. However, table 4.8 shows that an increase in the number of wins of about 80 wins is possible for a contestant with $\sigma = 0.1$ when using the polynomial strategy with parameter -0.907. Simulations earlier in this section indicate that the polynomial, exponential and random exponential strategies could increase the chance of winning for contestants with prediction errors around 0.15. From the simulations with 100,000 prediction tournaments in each step, it seems that the exponential strategy with parameter 2.019 is the best strategy with respect to the number of wins for these contestant. This is probably also the best strategy for contestants with $\sigma = 0.2$. Using no strategy seems to be the best option for contestants with prediction errors around 0.25. From the simulations, there seems to be no significant difference in number of wins between using no strategy or using a strategy from table 4.8 for the most inaccurate forecasters with prediction errors around 0.299.



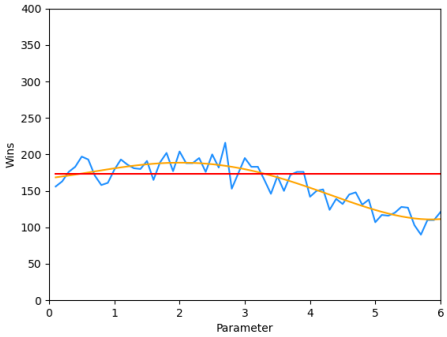
(a) Random exponential ($\sigma = 0$)



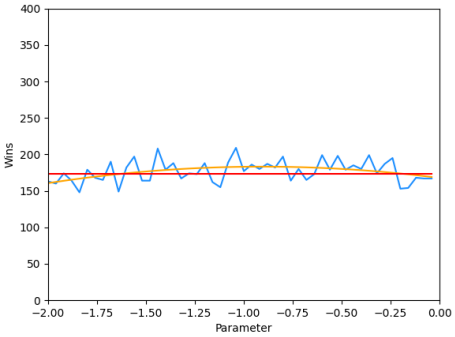
(b) Random exponential ($\sigma = 0.05$)



(c) Random exponential ($\sigma = 0.15$)



(d) Exponential ($\sigma = 0.15$)



(e) Polynomial ($\sigma = 0.15$)

Figure 4.15: Regression used for different strategies for different values of σ after competing 30000 in a prediction tournament. The red line corresponds to the number of wins of this contestant using no strategy.

σ	Strategy	Fitted polynomial	Best parameter
0	Random exponential	$N = -0.0120b^5 + 0.5270b^4 - 8.5499b^3$ $+60.7460b^2 - 164.5078b + 157.0129$	7.914
0.05	Random exponential	$N = 0.0025b^6 - 0.1060b^5 + 1.7604b^4 - 13.8815b^3$ $+49.3167b^2 - 51.4344b + 132.8774$	4.607
0.15	Random exponential	$N = -3.4028b^2 + 16.2785b + 159.7943$	2.392
0.15	Exponential	$N = 0.4731b^4 - 4.6971b^3 + 7.9915b^2$ $+9.5911b + 167.5301$	2.019
0.15	Polynomial	$N = -18.8859a^2 - 34.2464a + 167.6904$	-0.854

Table 4.7: Fitted polynomial and best parameter for different strategies and different values of σ after using polynomial regression in the setting that half of the opponents use the random exponential strategy with parameter 8.956 and half of the opponents use no strategy. In the polynomials, N is the number of wins, a is the parameter for the polynomial strategy and b is the parameter for the exponential strategy.

σ	No Strategy	Random Exponential 8.956	Random exponential 7.914	Random exponential 4.607	Random exponential 2.392	Exponential 2.019	Polynomial -0.907
0	0	351	**359**	231	32	1	0
0.05	414	464	519	**591**	502	395	418
0.1	956	496	647	884	989	1026	**1036**
0.15	550	321	386	542	600	**635**	580
0.2	235	146	153	214	242	**265**	258
0.25	**80**	52	48	58	59	77	52
0.299	**17**	14	**17**	15	13	**18**	13

Table 4.8: Number of wins for different σ's and different strategies after competing 100 000 times in a prediction tournament where half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956.

4.6 All opponents use random exponential strategy

Sections 4.2 and 4.3 have shown that none of the five strategies from this report result in a higher chance of winning than using no strategy if all opponents use an exponential strategy. In this section, it will be

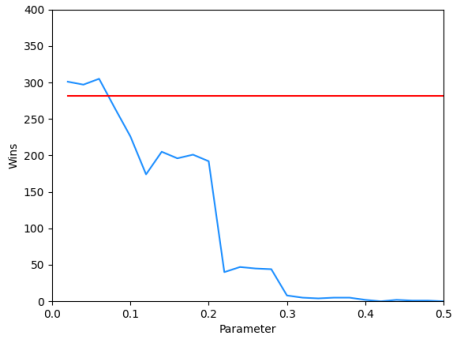
investigated whether this is also the case when all opponents use the random exponential strategy with parameter 8.956.

Figure 4.16 shows the number of wins of a contestant with $\sigma = 0.05$ after 10,000 prediction tournaments when she uses different strategies. None of the five strategies seems to result in a significant increase in the number of wins. The graph for hard-thresholding is above the red line for small values of the parameter. However, this seems to be caused by noise. Also contestants with other prediction errors can better use no strategy to achieve the most wins. Graphs to show this can be found in appendix H.

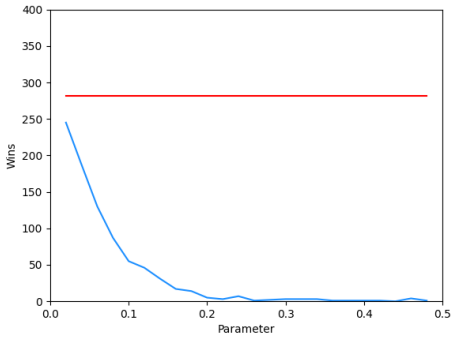
Table 4.9 shows the number of wins after 100,000 prediction tournaments for contestants who uses no strategy or the random exponential strategy with parameter 8.956. Again, it seems from this simulation that it is better to use no strategy than using the same strategy as the opponents.

σ	Random exponential 8.956	No Strategy
0	322	**2671**
0.05	474	**2926**
0.1	640	**2270**
0.15	430	**1096**
0.2	233	**429**
0.25	78	**165**
0.299	17	**39**

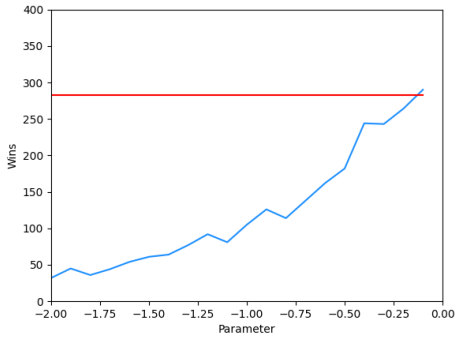
Table 4.9: Number of wins for different σ's and different strategies after competing 100,000 times in a prediction tournament where all opponents are using the random exponential strategy with parameter 8.956.



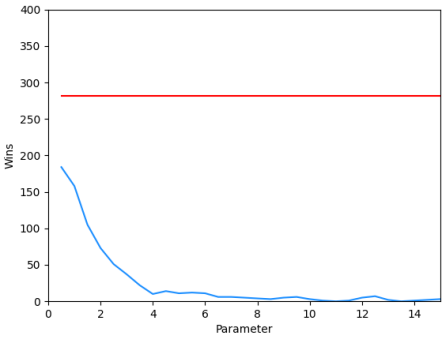
(a) Hard thresholding



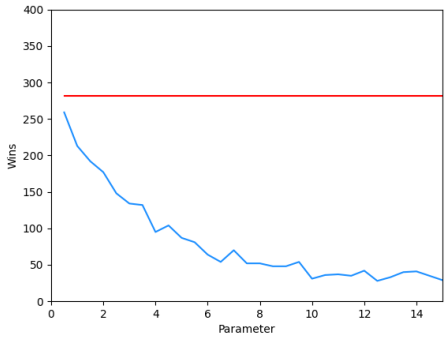
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy



(e) Random exponential strategy

Figure 4.16: Comparison of five strategies for the contestant with $\sigma = 0.05$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

Chapter 5

Conclusion

The central question of this report is, which strategy could a contestant use to increase her chance of winning? In order to answer this question, a few situations were simulated and analysed. In each situation, the opponents of the contestant use other strategies and it was analysed which strategy was the best to use for a specific contestant with a fixed prediction error. Firstly, a prediction tournament was considered in section 3.2 in which the opponents just predict the probabilities they believe to be the true probabilities. The exponential strategy turned out to be the best strategy in terms of chance of winning. The parameter corresponding to this strategy differs for various values of prediction error σ. Precise predictors who always predict the true probabilities are recommended to choose parameter 3.969. For contestants with $\sigma = 0.05$ the best parameter is 3.517. The parameter 2.641 is preferred for contestants with $\sigma = 0.1$, $\sigma = 0.15$ or $\sigma = 0.25$. Contestants with prediction error 0.2 can choose parameter 2.626 for a higher chance of winning. The most inaccurate contestants in the prediction tournament are advised to use no strategy, since using strategies only diminishes the chance of winning.

After that, situations were discussed in chapter 4 in which some of the opponents use a strategy. A situation that is analysed is the situation in which half of the opponents use the exponential strategy with parameter 3.969 and half of the opponents do not use any strategy. For accurate predictors, the random exponential strategy is the best strategy. Parameter 8.956 is recommended for contestants with $\sigma = 0$ and parameter 5.143 is recommended in the case that $\sigma = 0.05$. The polynomial strategy with parameter 0.811 results in the highest chance of winning for contestants with prediction errors around 0.1. For contestants with $\sigma = 0.15$, both

the polynomial strategy with parameter -0.811 and the exponential strategy with parameter 1.226 can increase the chance of winning by approximately the same factor. The exponential strategy with parameter 1.226 is recommended for contestants with $\sigma = 0.2$ or $\sigma = 0.25$. For inaccurate contestants with $\sigma = 0.299$, the best choice is to use no strategy. Similarly to the previous case, the case in which half of the contestants use the exponential strategy with parameter 2.641 is analysed. In this case, the polynomial strategy is the best strategy for contestants with prediction errors around 0 or 0.05. The recommended parameters are 9.141 and 6.074 respectively. The random exponential strategy with parameter 2.347 is advised to contestants with $\sigma \in [0.1, 0.25]$. The polynomial strategy with parameter -0.854 has an equally large positive impact on the number of wins as the random exponential strategy with parameter 2.347 for contestants with $\sigma = 0.1$. As in previous cases, the most inaccurate contestants are recommended to use no additional strategy. In the following case, half of the opponents use the random exponential strategy with parameter 8.956 and half of the opponents use no additional strategy. In this case, the random exponential strategy with parameters 7.194 and 4.607 results in the largest chance of winning for contestants with $\sigma = 0$ and $\sigma = 0.05$, respectively. The polynomial strategy with parameter -0.907 is recommended for contestants with prediction errors around 0.1. For contestants with $\sigma = 0.15$ or $\sigma = 0.2$, the strategy that results in the most wins is probably the exponential strategy with parameter 2.019. The most inaccurate contestants with $\sigma \geq 0.25$ could not achieve a significant increase in the number of wins if they use a strategy. If all opponents use the exponential strategy with parameter 3.969 or 2.641 or the random exponential strategy with parameter 8.956, then the best choice is to use no additional strategy.

In all situations a phenomenon stands out. Every time the best option for a contestant is to do something totally different from what the opponents do. Therefore, the strategies used by the contestant's opponents are an important factor that effects a contestant's chance of winning. For the exponential and random exponential strategy, it stands out that in different situations the best parameter will be smaller for higher values of σ. So, the curve of the graph of the exponential strategy will be smaller if the prediction error of a contestant becomes larger. This effect reduces when σ becomes larger.

In this research the number of contestants, the grid of true probabilities and the interval containing all prediction errors were fixed. However, these factors will probably effect the number of wins. A topic of further research could be how the contestant needs to change her strategy if one

of these factors vary.

Using no strategy is preferred for all contestants if all opponents use the exponential strategy with parameter 3.969 or 2.641. Because using the random exponential strategy does not result in a larger chance of winning in these cases, a variation on the random exponential strategy was tried. In the usual random exponential strategy the probability that the exponential strategy is used is 0.5. In the new strategy the probability of choosing the exponential strategy is 0.1 or 0.25. But, it turns out that this change does not result in more wins than achieved without using any strategy. So, an idea for further research could be to design a strategy totally different from the ones in this report that results in more wins than using no additional strategy when all opponents use the exponential strategy with parameter 3.969 or 2.641. The same could be tried in the case that all opponents use the random exponential strategy with parameter 8.956.

Various factors of prediction tournaments are unknown to the contestants. An topic for further research could be to analyse how more information about the prediction tournament can be gathered. For example, it might be interesting to investigate how a contestant can estimate her prediction error, the prediction errors of opponents or the strategies used by opponents. This is not studied in this report and it is unclear for which data and under which assumptions this can be estimated. Also a game-theoretic analysis is possible for further research to determine which strategy is the best to use with regard of the chance of winning.

Bibliography

[1] Herox (2018). Geopolitical Forecasting [GF] Challenge. https://www.herox.com/IARPAGFChallenge

[2] Metaculus (2023). Q3 2023 Quarterly Cup. https://www.metaculus.com/tournament/quarterly-cup-2023q3/?page=3

[3] Good Judgement Open (2026). 2026 World Elections Challenge. https://www.gjopen.com/challenges/128-2026-world-elections-challenge

[4] Aldous, D. J. (2019). A Prediction Tournament Paradox. The American Statistician, 75(3), 243-248. https://doi.org/10.1080/00031305.2019.1604430

[5] Van der Eng, V. M. (2025). Analysis of the Prediction Tournament Paradox. Bachelor thesis, TU Delft. https://resolver.tudelft.nl/uuid:512ec4c2-dd44-4dfb-9a89-cb413a5bbee3

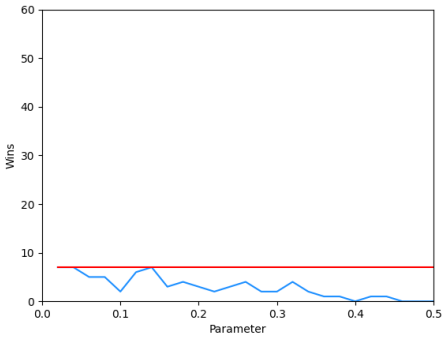
[6] Ostertagová, E. (2012). Modelling using polynomial regression. Procedia Engineering, 48, 500-506. https://www.sciencedirect.com/science/article/pii/S1877705812046085

[7] Shipley, B. (2013) The AIC model selection method applied to path analytic models compared using a d-separation test. Ecology, 94, 560-564. https://esajournals.onlinelibrary.wiley.com/doi/full/10.1890/12-0976.1

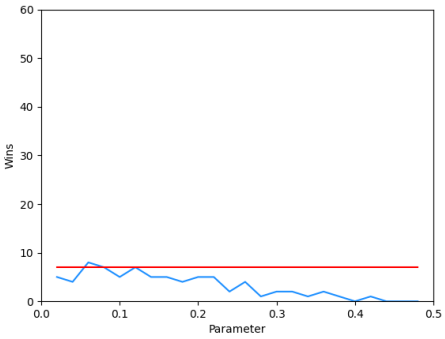
[8] Ostwal, P. (2019) Polynomial regression using statsmodel. Blog post on personal blog. https://ostwalprasad.github.io/machine-learning/Polynomial-Regression-using-statsmodel.html

Appendix A

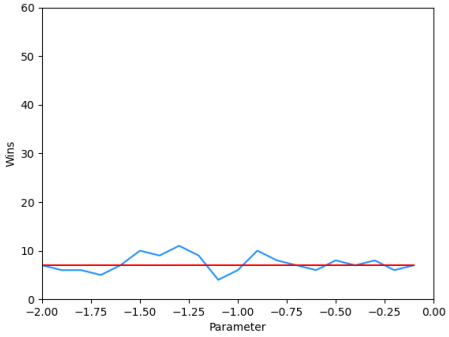
Graphs of contestants with high σ's when all opponents use no strategy



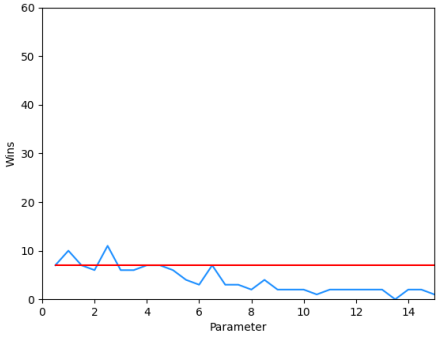
(a) Hard thresholding



(b) Soft thresholding

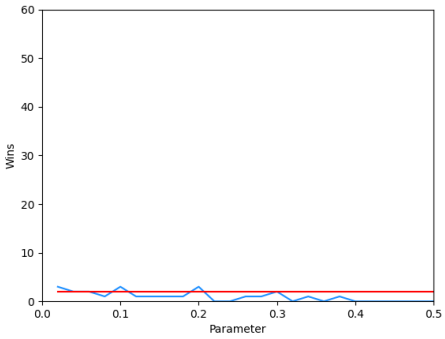


(c) Polynomial strategy

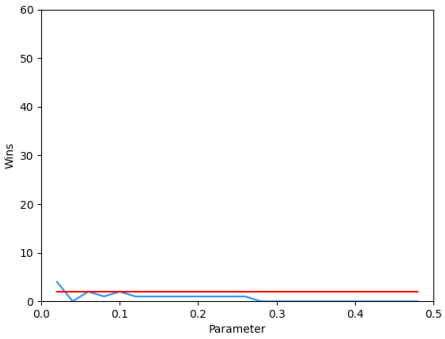


(d) Exponential strategy

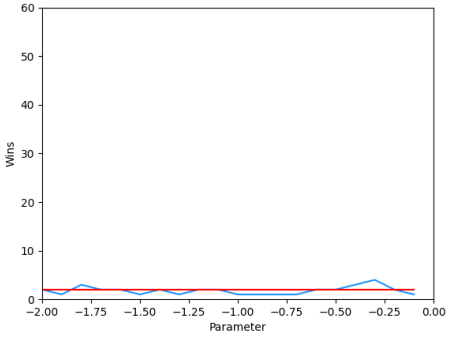
Figure A.1: Comparison of four strategies for the contestant with $\sigma = 0.2$. The red line corresponds to the number of wins of this contestant using no strategy.



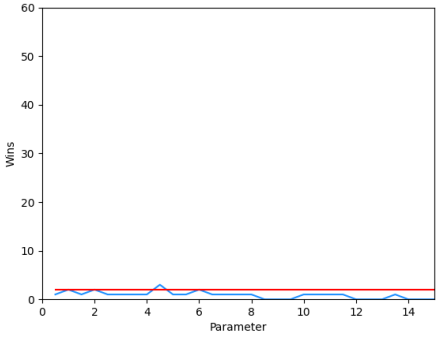
(a) Hard thresholding



(b) Soft thresholding

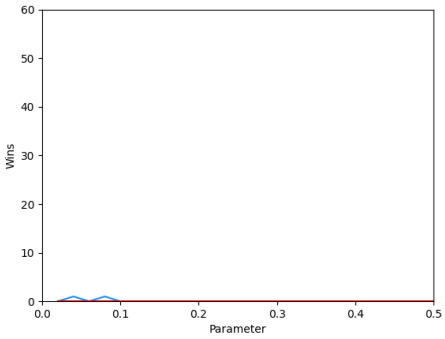


(c) Polynomial strategy

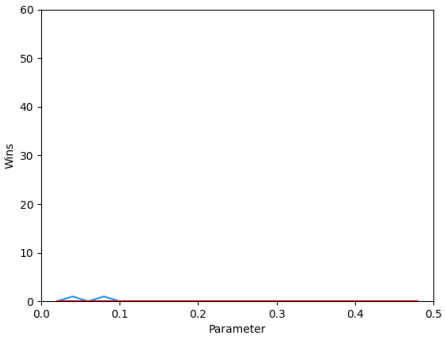


(d) Exponential strategy

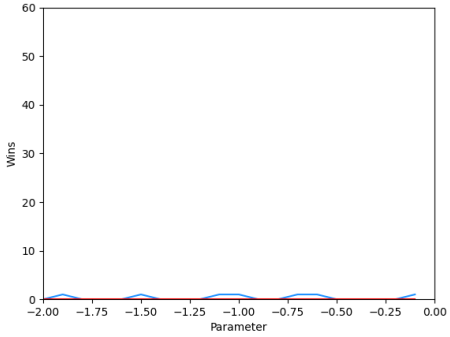
Figure A.2: Comparison of four strategies for the contestant with $\sigma = 0.25$. The red line corresponds to the number of wins of this contestant using no strategy.



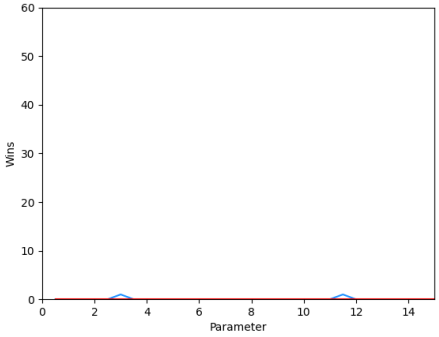
(a) Hard thresholding



(b) Soft thresholding



(c) Polynomial strategy

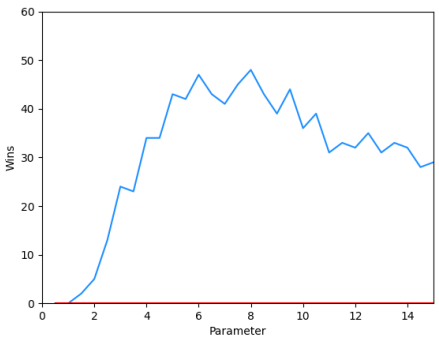


(d) Exponential strategy

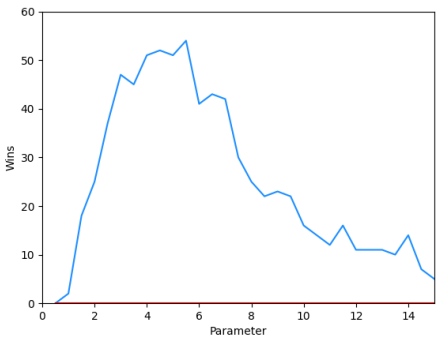
Figure A.3: Comparison of four strategies for the contestant with $\sigma = 0.299$. The red line corresponds to the number of wins of this contestant using no strategy.

Appendix B

Graphs of contestants who are using the random exponential or exponential strategy when all opponents use no strategy.

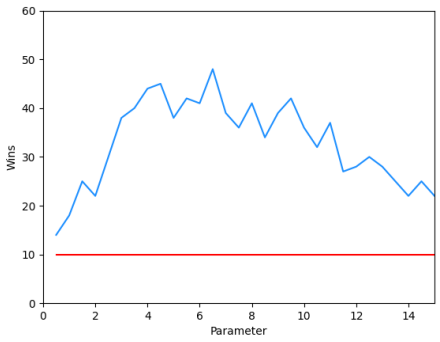


(a) Random exponential strategy

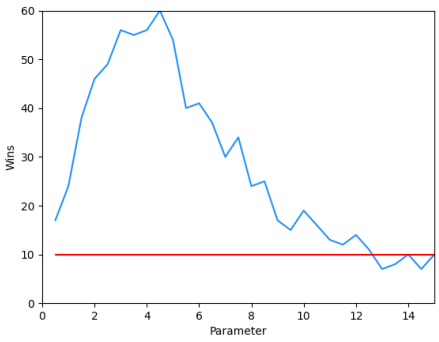


(b) Exponential strategy

Figure B.1: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0$ when all opponents use no strategy. The red line on the bottom of the plot corresponds to the number of wins of this contestant using no strategy.

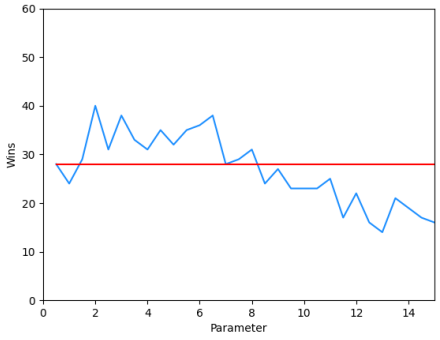


(a) Random exponential strategy

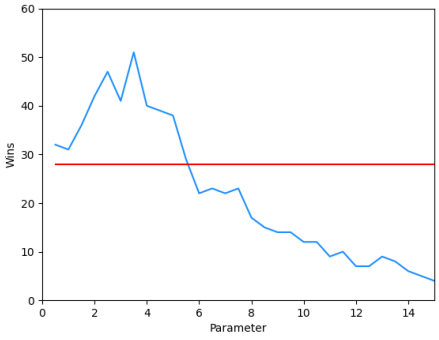


(b) Exponential strategy

Figure B.2: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.05$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.

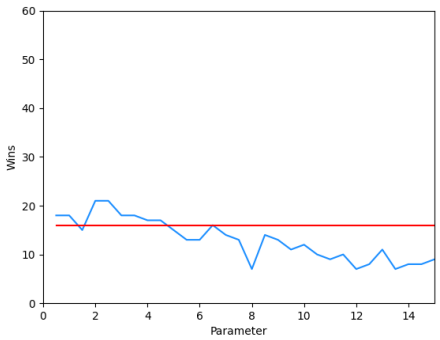


(a) Random exponential strategy

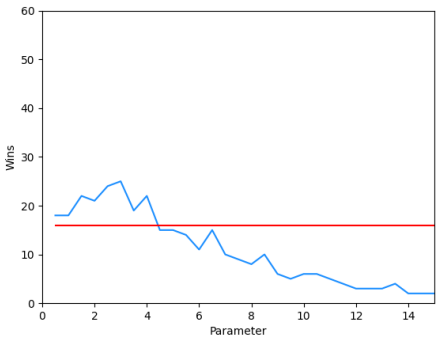


(b) Exponential strategy

Figure B.3: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.1$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.

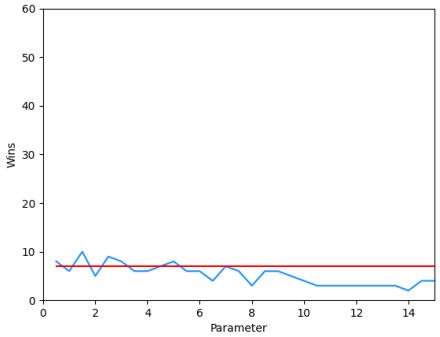


(a) Random exponential strategy

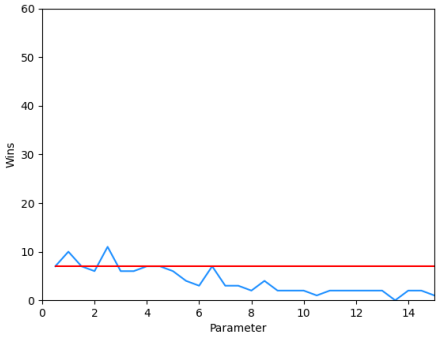


(b) Exponential strategy

Figure B.4: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.15$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.

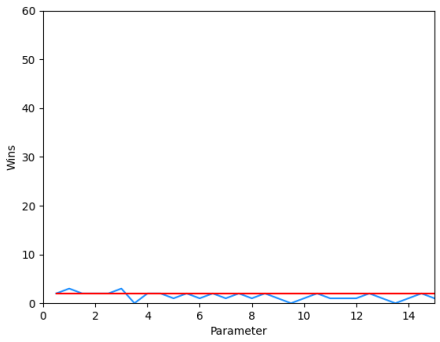


(a) Random exponential strategy

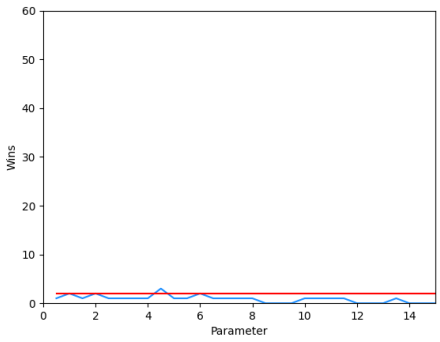


(b) Exponential strategy

Figure B.5: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.2$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.

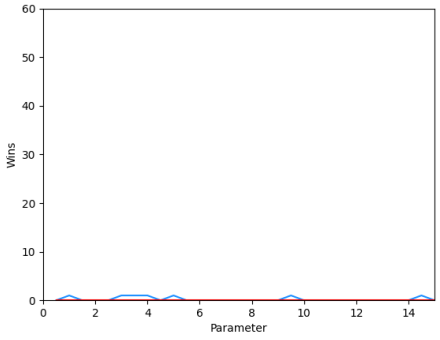


(a) Random exponential strategy

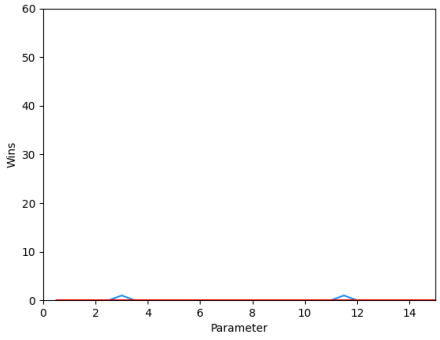


(b) Exponential strategy

Figure B.6: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.25$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.



(a) Random exponential strategy

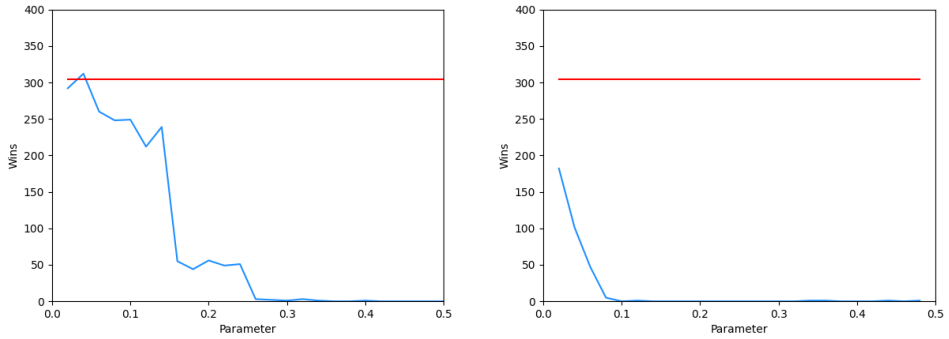


(b) Exponential strategy

Figure B.7: Comparison of the exponential and random exponential strategy for the contestant with $\sigma = 0.299$ when all opponents use no strategy. The red line corresponds to the number of wins of this contestant using no strategy.

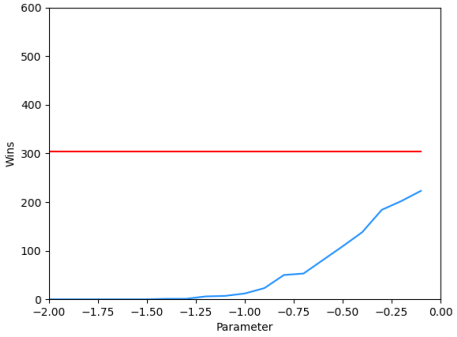
Appendix C

Graphs all opponents exponential 3.969

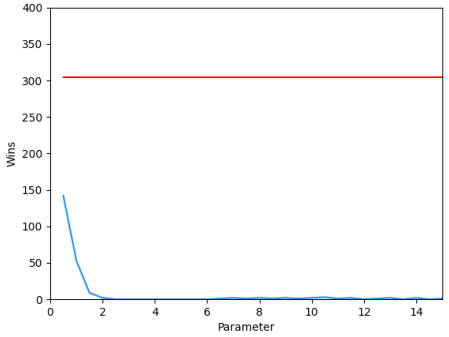


(a) Hard thresholding

(b) Soft thresholding

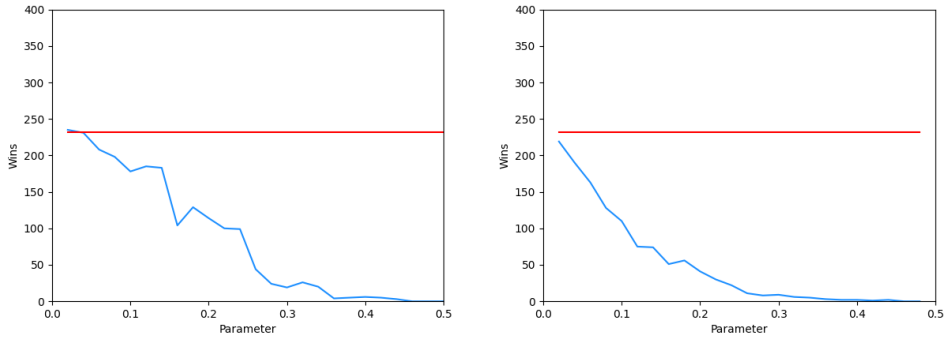


(c) Polynomial strategy



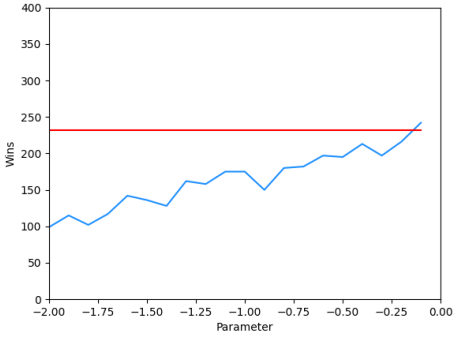
(d) Exponential strategy

Figure C.1: Comparison of four strategies for the contestant with $\sigma = 0$. The red line corresponds to the number of wins of this contestant using no strategy.

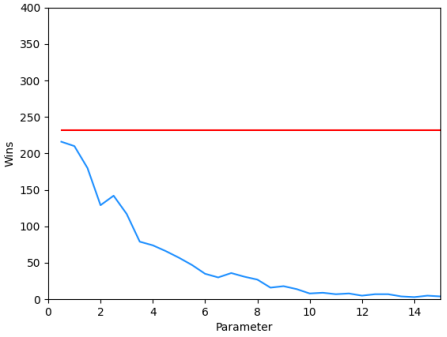


(a) Hard thresholding

(b) Soft thresholding

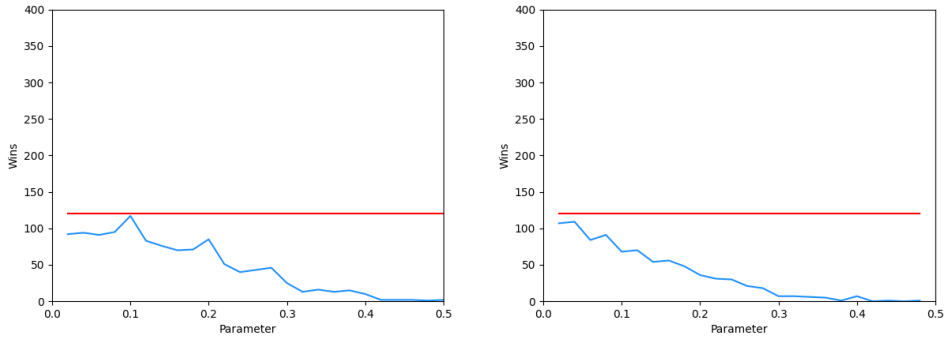


(c) Polynomial strategy



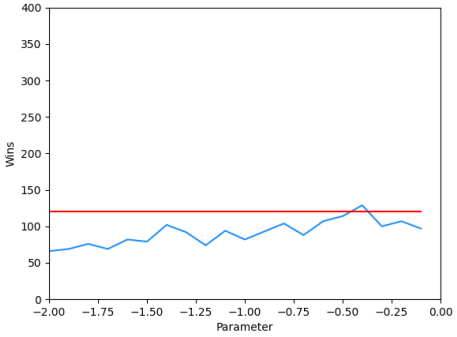
(d) Exponential strategy

Figure C.2: Comparison of four strategies for the contestant with $\sigma = 0.1$. The red line corresponds to the number of wins of this contestant using no strategy.

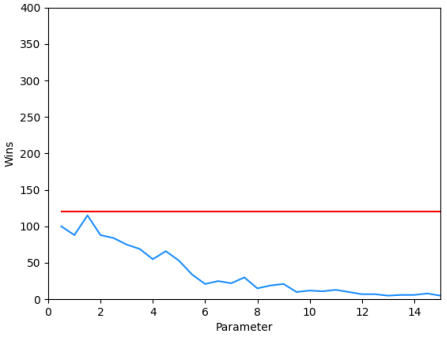


(a) Hard thresholding

(b) Soft thresholding

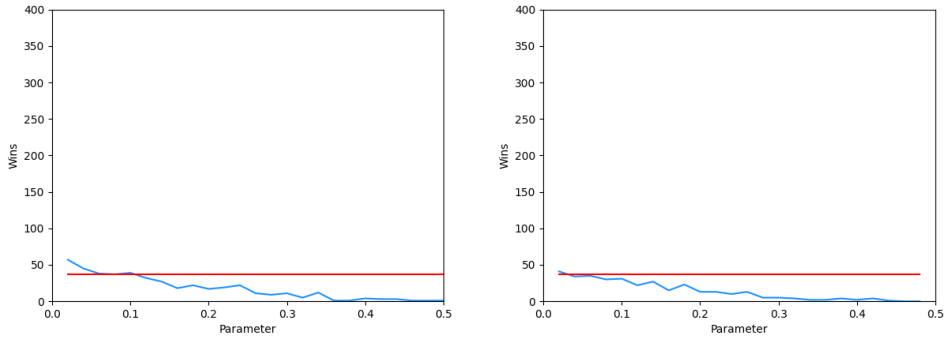


(c) Polynomial strategy



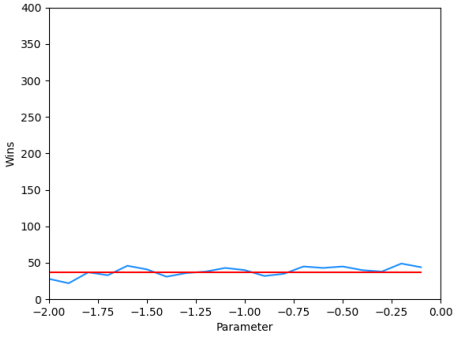
(d) Exponential strategy

Figure C.3: Comparison of four strategies for the contestant with $\sigma = 0.15$. The red line corresponds to the number of wins of this contestant using no strategy.

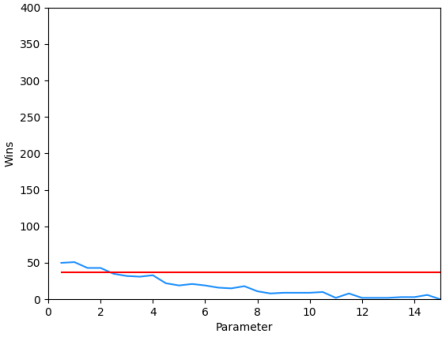


(a) Hard thresholding

(b) Soft thresholding

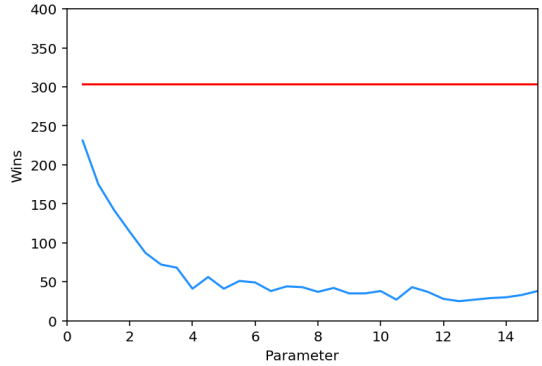


(c) Polynomial strategy

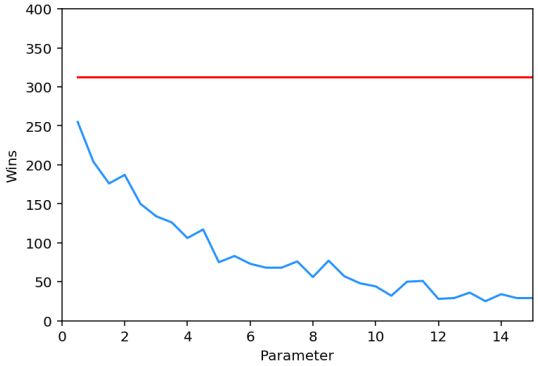


(d) Exponential strategy

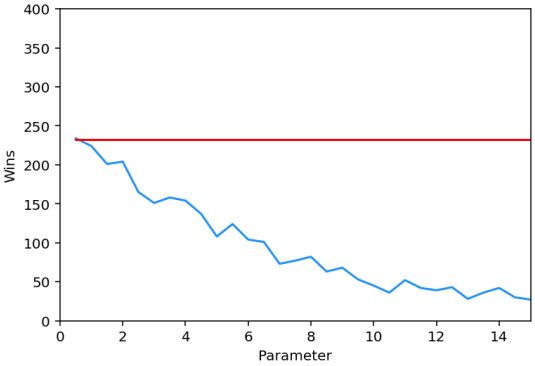
Figure C.4: Comparison of four strategies for the contestant with $\sigma = 0.2$. The red line corresponds to the number of wins of this contestant using no strategy.



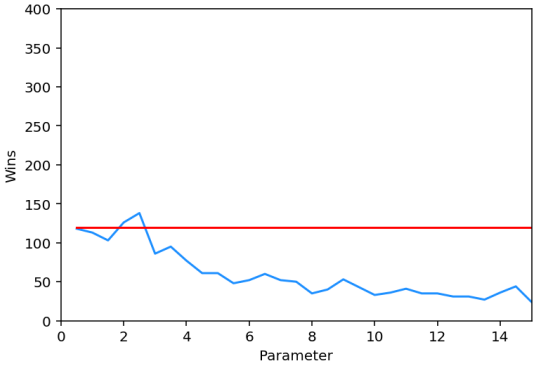
(a) $\sigma = 0$



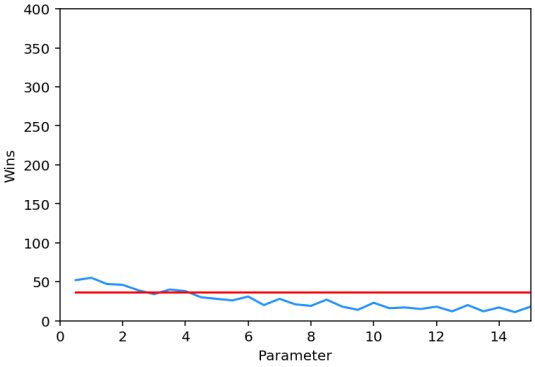
(b) $\sigma = 0.05$



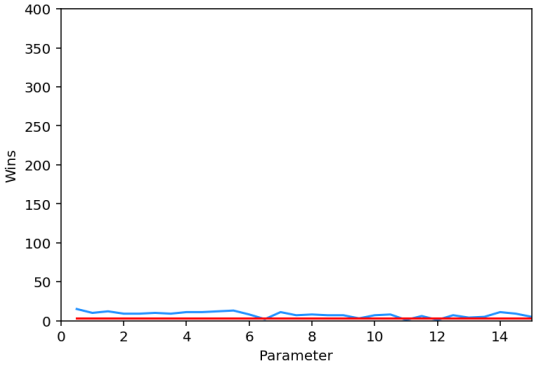
(c) $\sigma = 0.1$



(d) $\sigma = 0.15$



(e) $\sigma = 0.2$

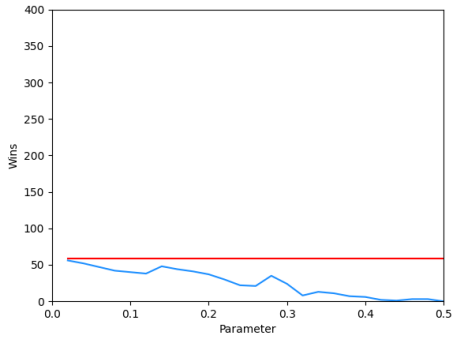


(f) $\sigma = 0.25$

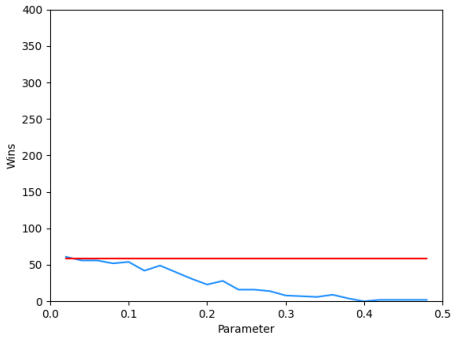
Figure C.5: Graphs of the number of wins of the only contestant using the random exponential strategy when the opponents use the exponential strategy with parameter 3.969 for different values of σ. The red line corresponds to the number of wins of this contestant using no strategy.

Appendix D

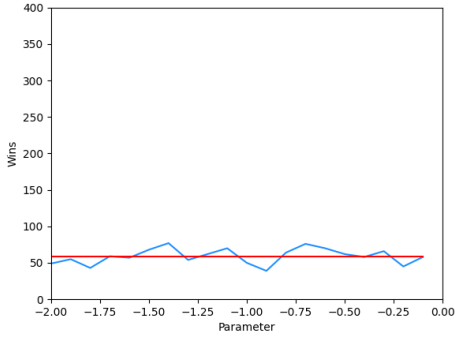
Graphs of contestants with high σ's when half of the opponents use exponential strategy with parameter 3.969



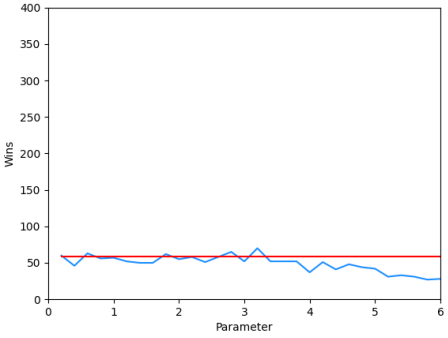
(a) Hard thresholding



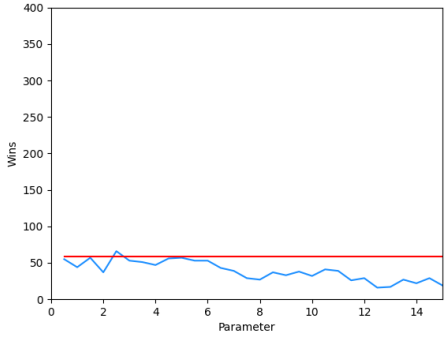
(b) Soft thresholding



(c) Polynomial strategy

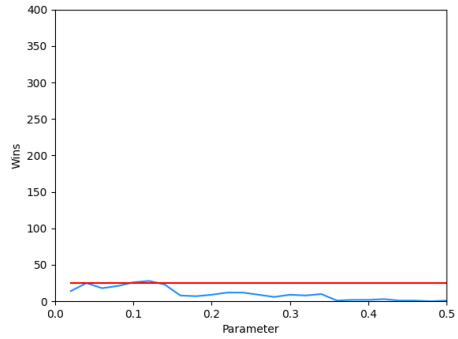


(d) Exponential strategy

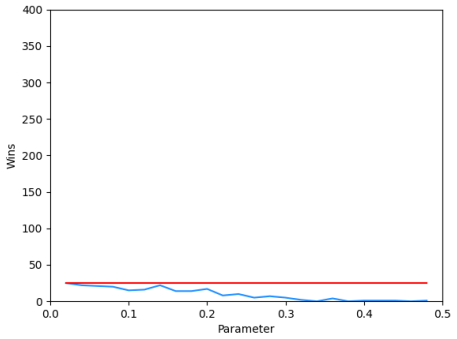


(e) Random exponential strategy

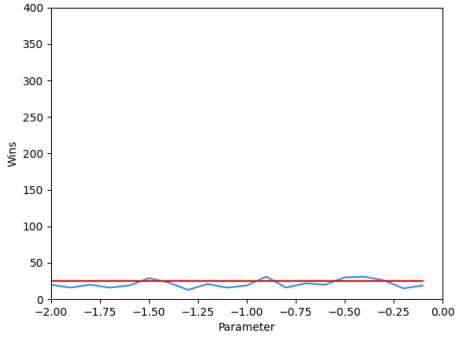
Figure D.1: Comparison of five strategies for the contestant with $\sigma = 0.15$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy.



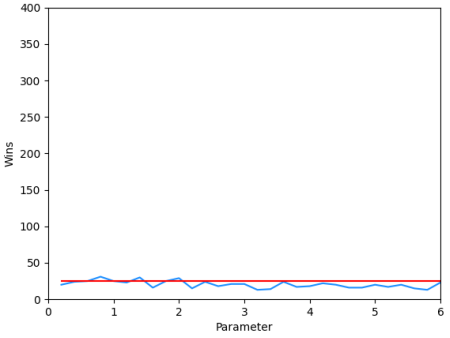
(a) Hard thresholding



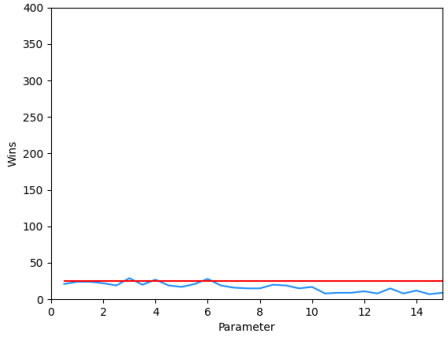
(b) Soft thresholding



(c) Polynomial strategy

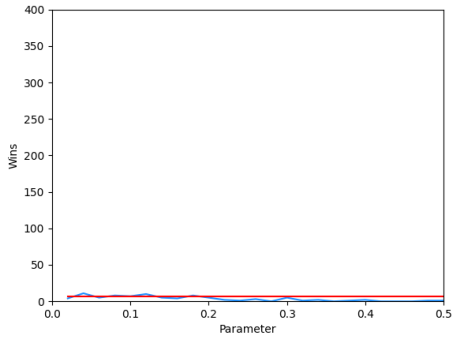


(d) Exponential strategy

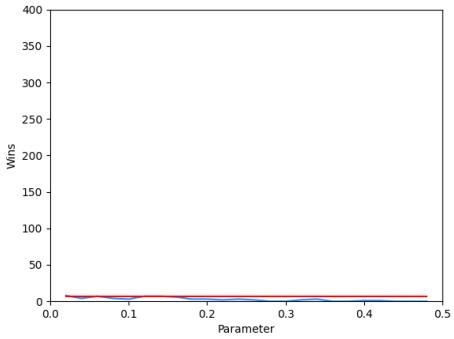


(e) Random exponential strategy

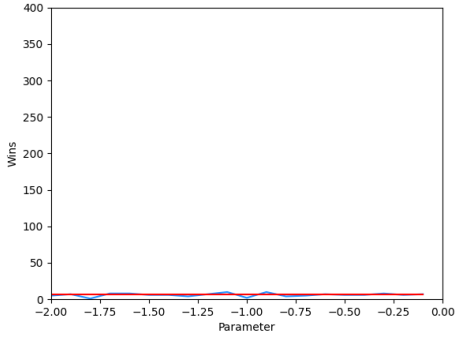
Figure D.2: Comparison of five strategies for the contestant with $\sigma = 0.2$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy.



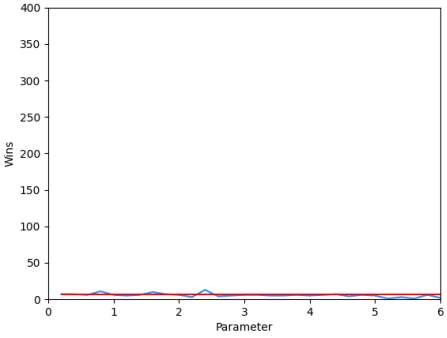
(a) Hard thresholding



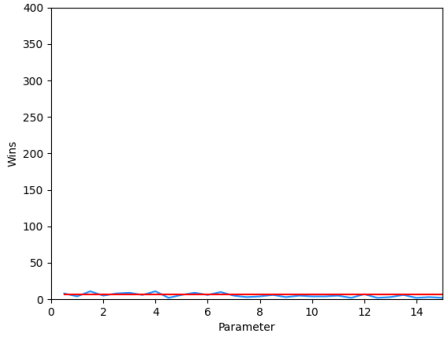
(b) Soft thresholding



(c) Polynomial strategy

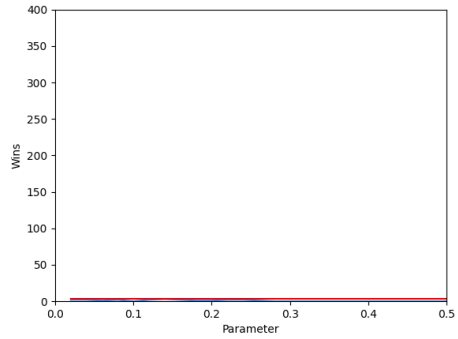


(d) Exponential strategy

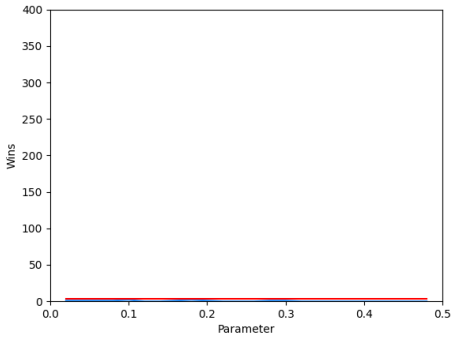


(e) Random exponential strategy

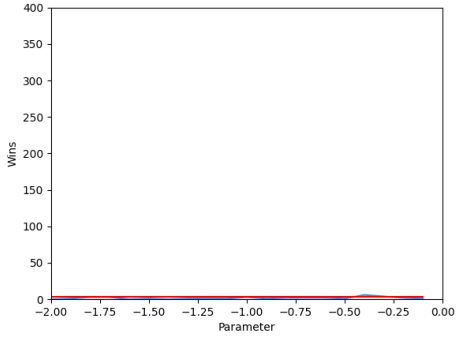
Figure D.3: Comparison of five strategies for the contestant with $\sigma = 0.25$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy.



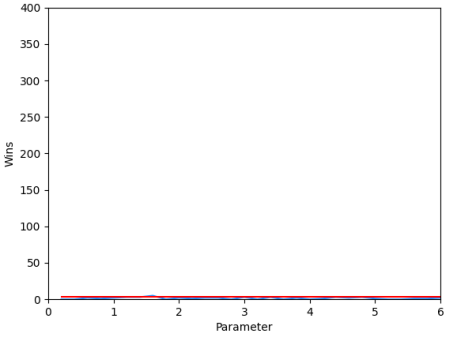
(a) Hard thresholding



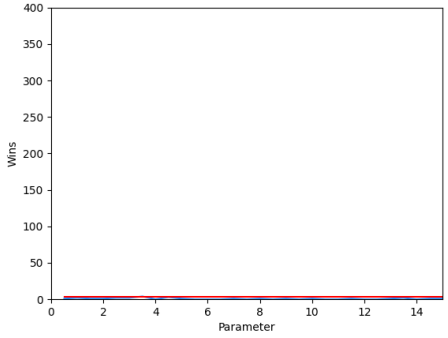
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

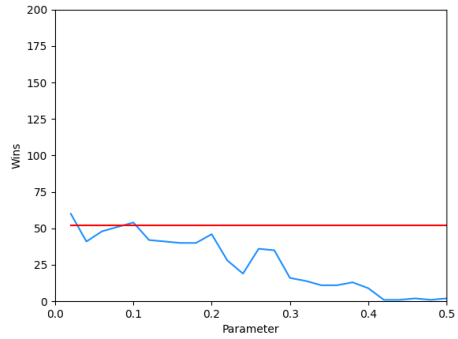


(e) Random exponential strategy

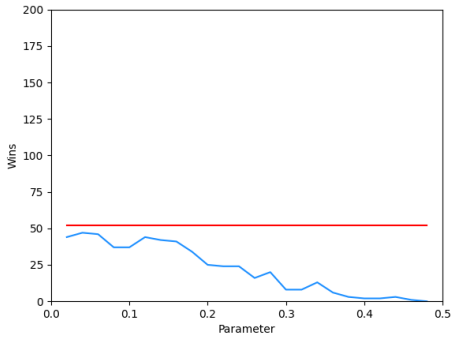
Figure D.4: Comparison of five strategies for the contestant with $\sigma = 0.299$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 3.969. The red line corresponds to the number of wins of this contestant using no strategy.

Appendix E

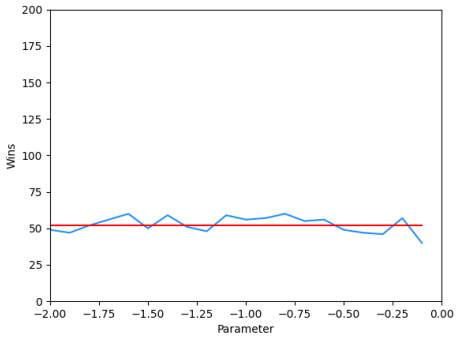
Graphs of contestants with high σ's when half of the opponents use exponential strategy with parameter 2.641



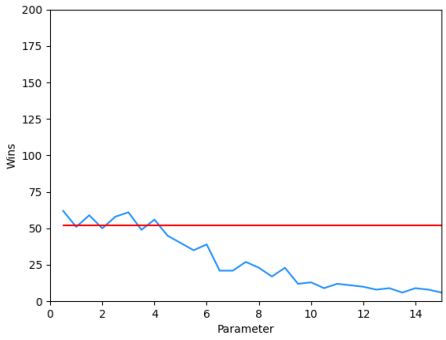
(a) Hard thresholding



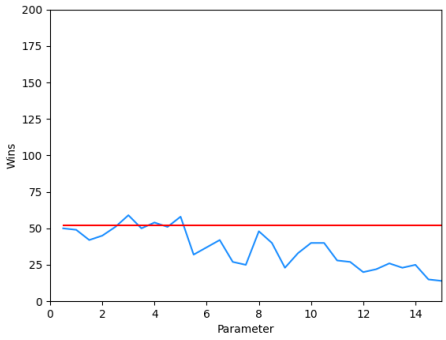
(b) Soft thresholding



(c) Polynomial strategy

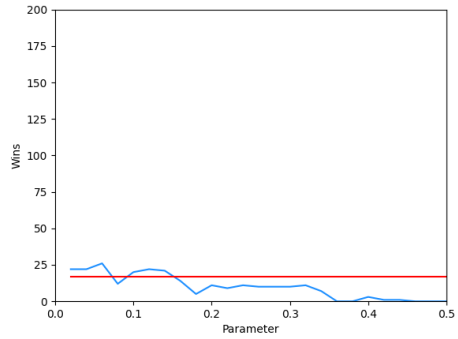


(d) Exponential strategy

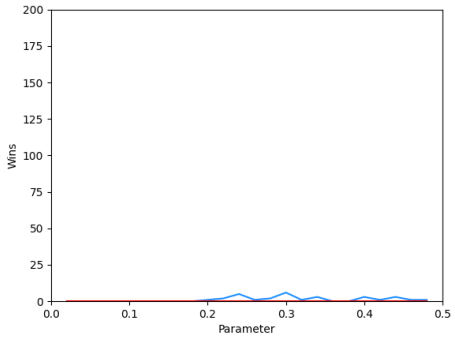


(e) Random exponential strategy

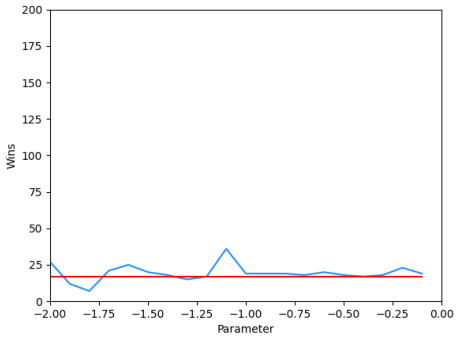
Figure E.1: Comparison of five strategies for the contestant with $\sigma = 0.15$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



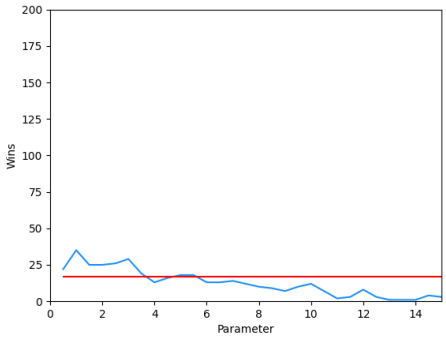
(a) Hard thresholding



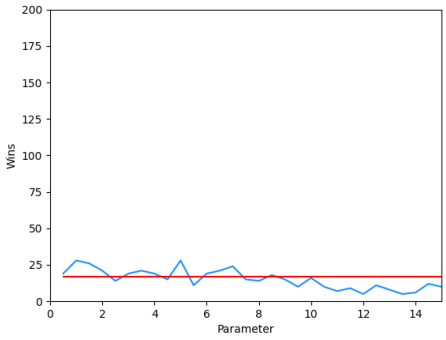
(b) Soft thresholding



(c) Polynomial strategy

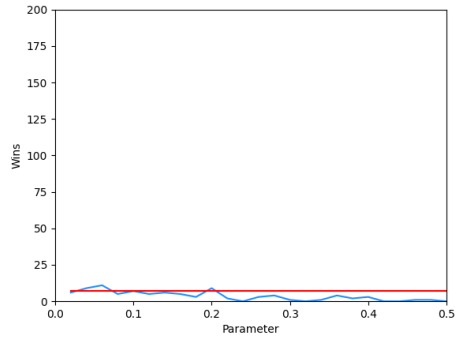


(d) Exponential strategy

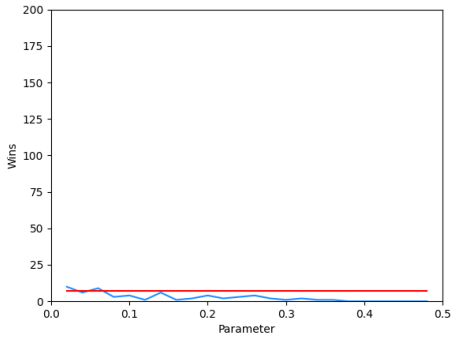


(e) Random exponential strategy

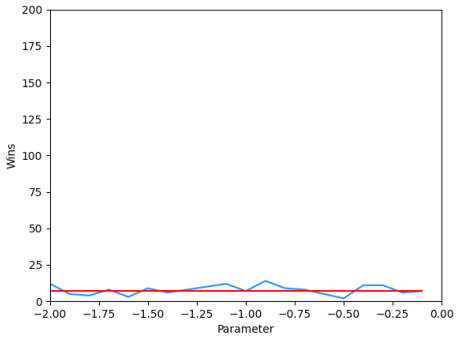
Figure E.2: Comparison of five strategies for the contestant with $\sigma = 0.2$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



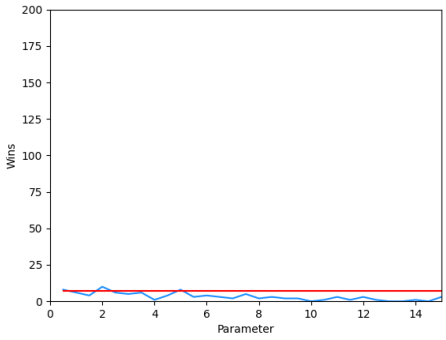
(a) Hard thresholding



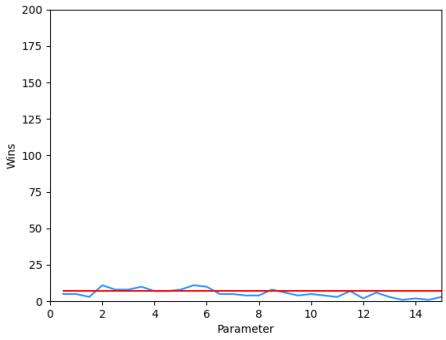
(b) Soft thresholding



(c) Polynomial strategy

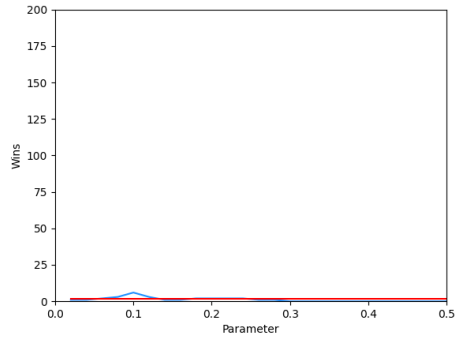


(d) Exponential strategy

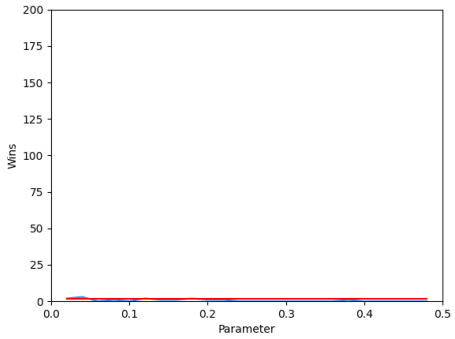


(e) Random exponential strategy

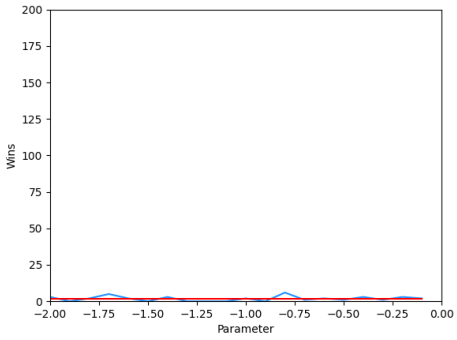
Figure E.3: Comparison of five strategies for the contestant with $\sigma = 0.25$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



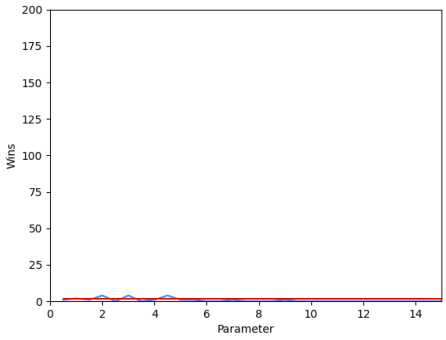
(a) Hard thresholding



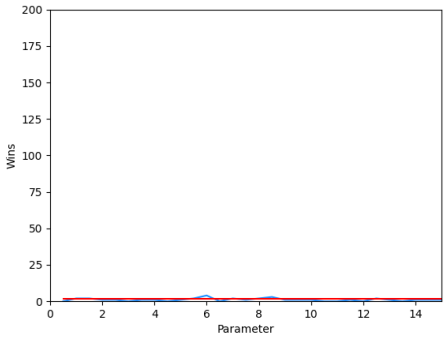
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

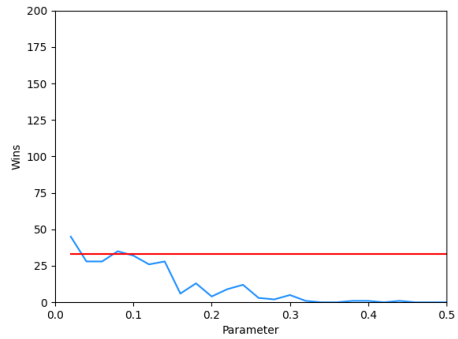


(e) Random exponential strategy

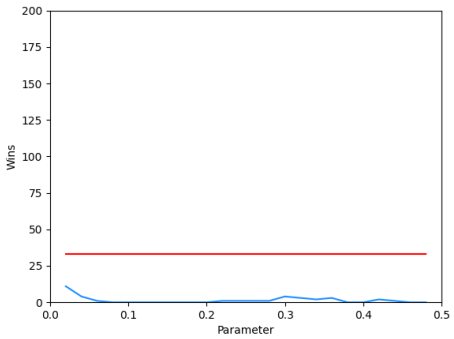
Figure E.4: Comparison of 5 strategies for the contestant with $\sigma = 0.299$ in the case that half of the opponents use no strategy and half of the opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10000 tournaments.

Appendix F

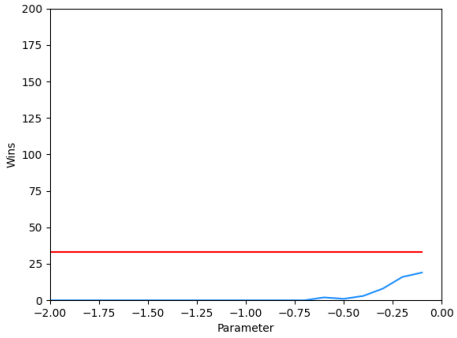
Graphs all opponents exponential 2.641



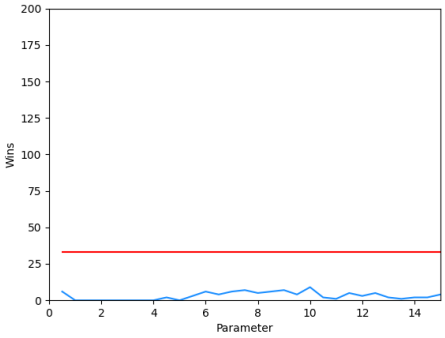
(a) Hard thresholding



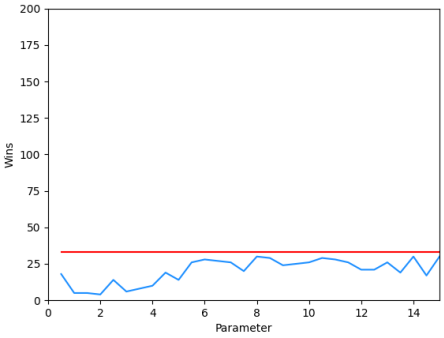
(b) Soft thresholding



(c) Polynomial strategy

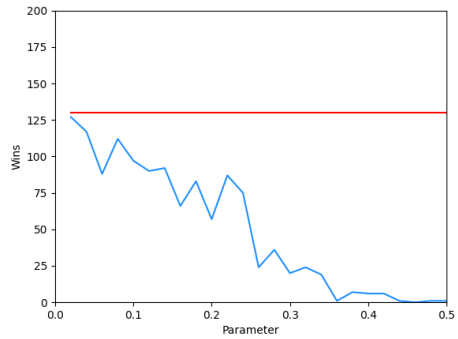


(d) Exponential strategy

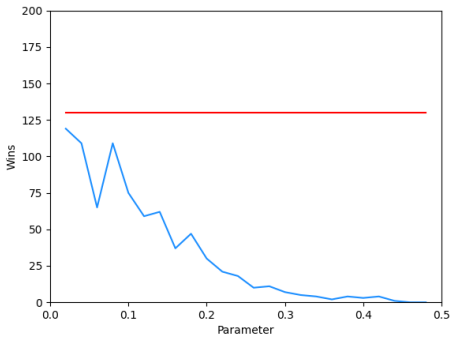


(e) Random exponential strategy

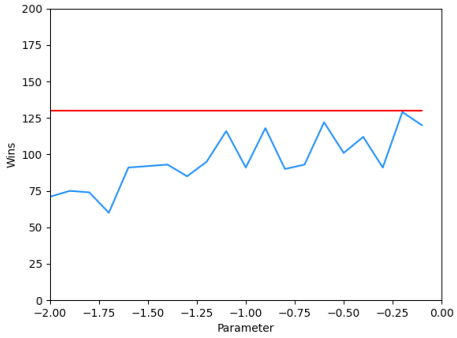
Figure F.1: Comparison of five strategies for the contestant with $\sigma = 0$ in the case that all opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy.



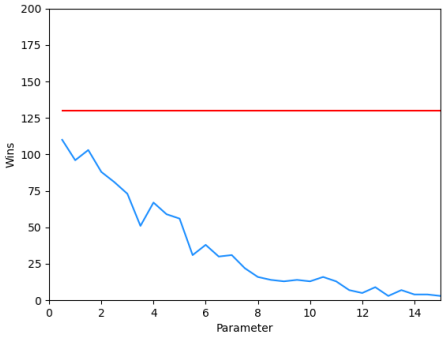
(a) Hard thresholding



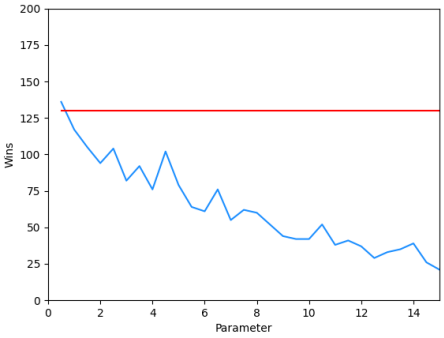
(b) Soft thresholding



(c) Polynomial strategy

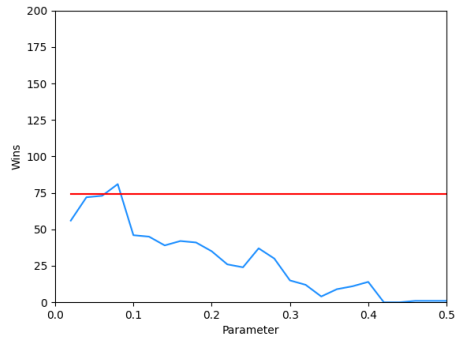


(d) Exponential strategy

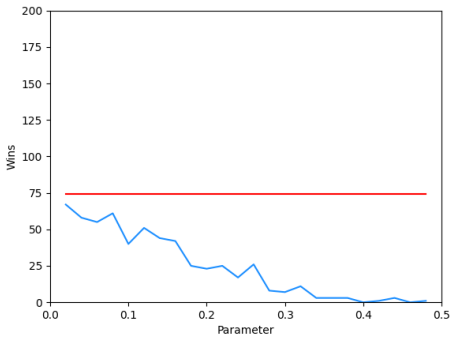


(e) Random exponential strategy

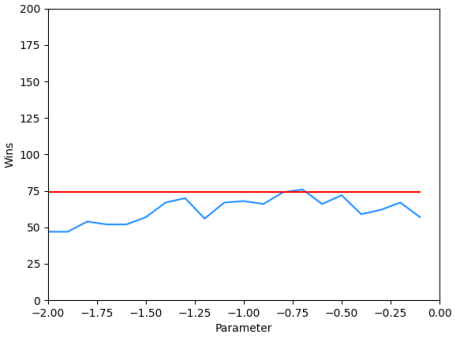
Figure F.2: Comparison of five strategies for the contestant with $\sigma = 0.1$ in the case that all opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy.



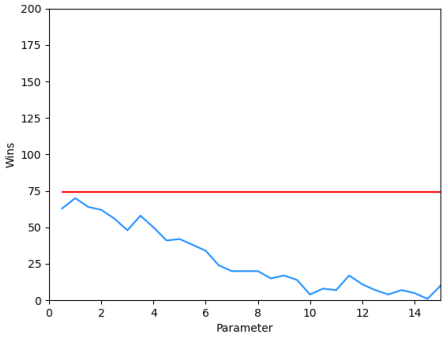
(a) Hard thresholding



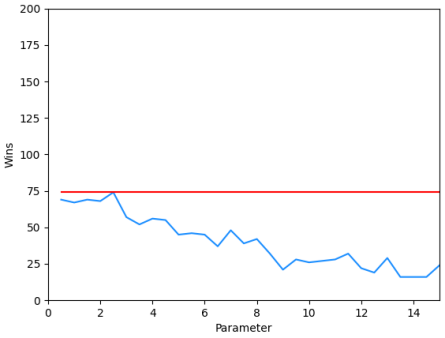
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

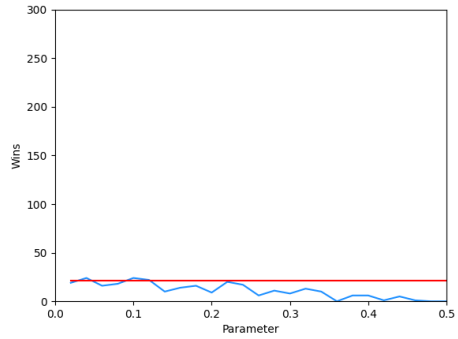


(e) Random exponential strategy

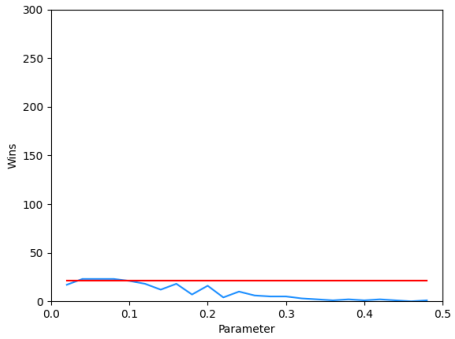
Figure F.3: Comparison of five strategies for the contestant with $\sigma = 0.15$ in the case that all opponents use the exponential strategy with parameter 2.641. The red line corresponds to the number of wins of this contestant using no strategy.

Appendix G

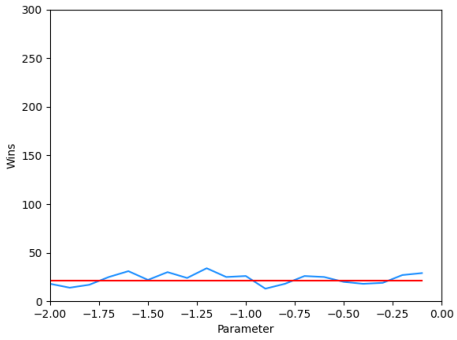
Graphs of contestants with high σ's when half of the opponents use random exponential strategy with parameter 8.956



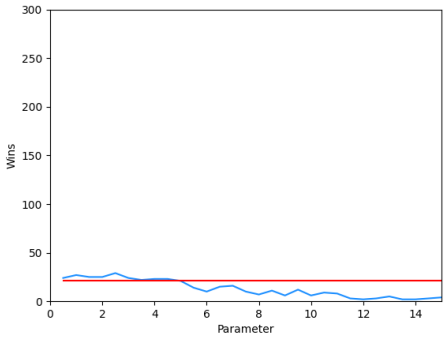
(a) Hard thresholding



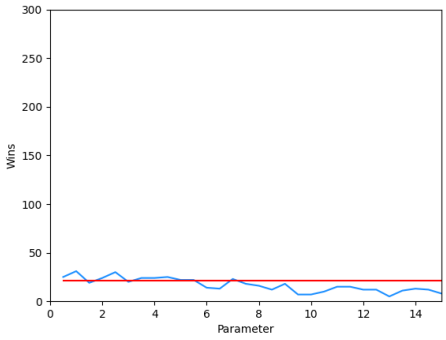
(b) Soft thresholding



(c) Polynomial strategy

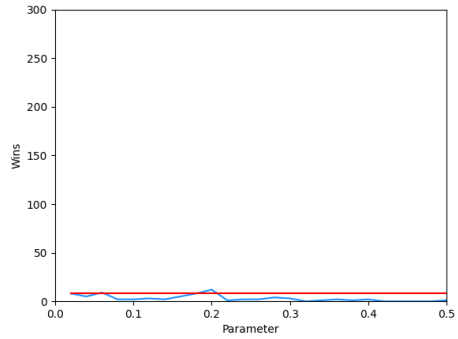


(d) Exponential strategy

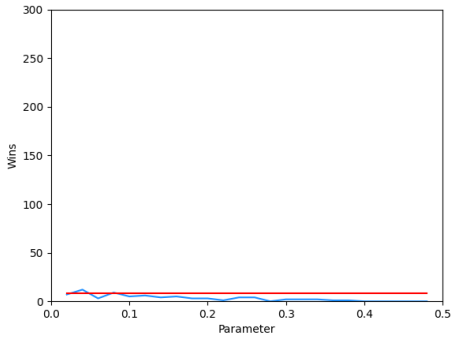


(e) Random exponential strategy

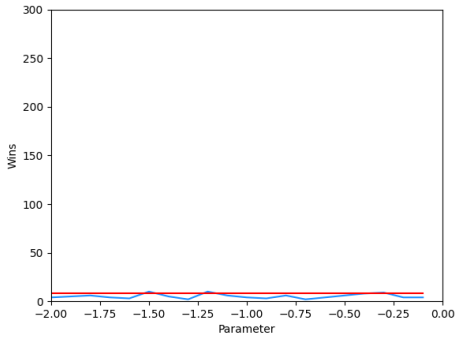
Figure G.1: Comparison of five strategies for the contestant with $\sigma = 0.2$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



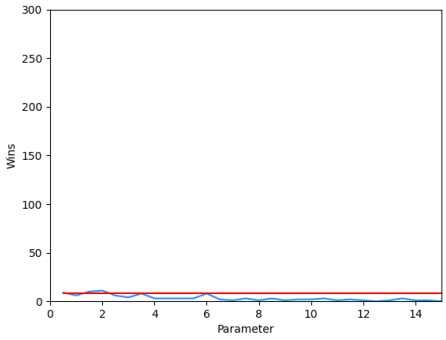
(a) Hard thresholding



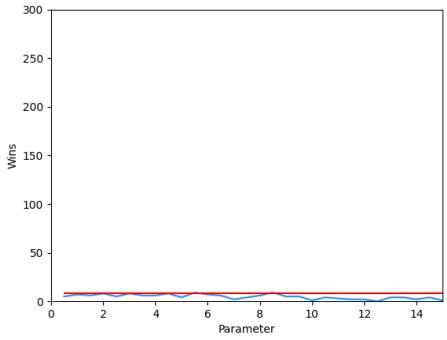
(b) Soft thresholding



(c) Polynomial strategy

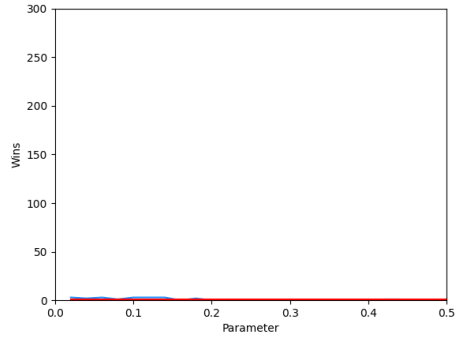


(d) Exponential strategy

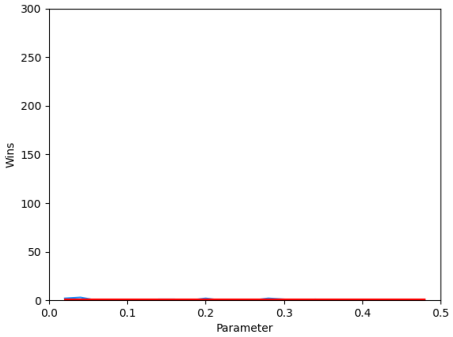


(e) Random exponential strategy

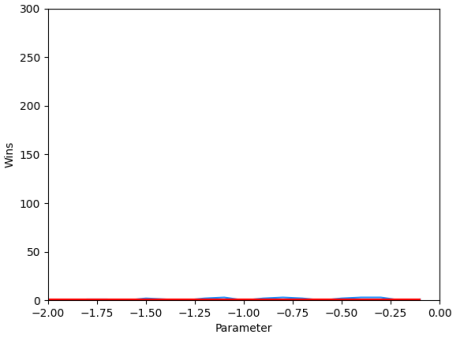
Figure G.2: Comparison of five strategies for the contestant with $\sigma = 0.25$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



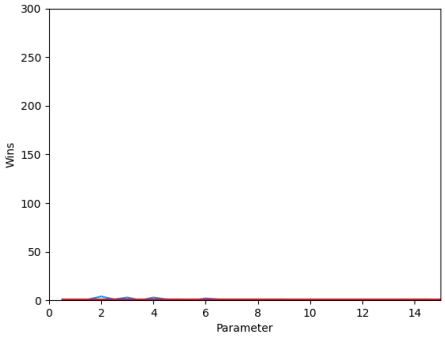
(a) Hard thresholding



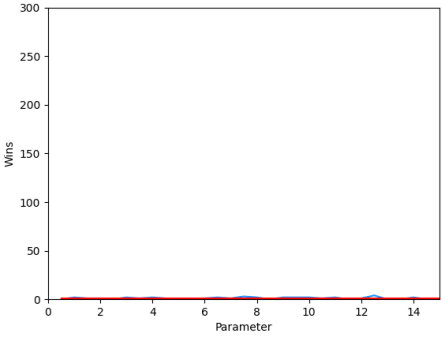
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy

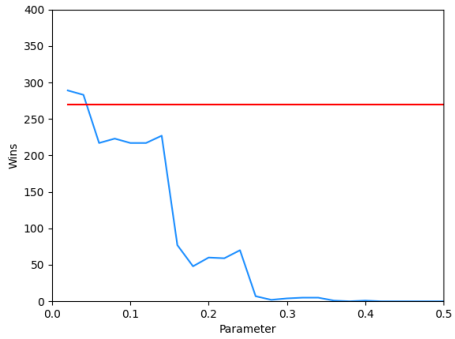


(e) Random exponential strategy

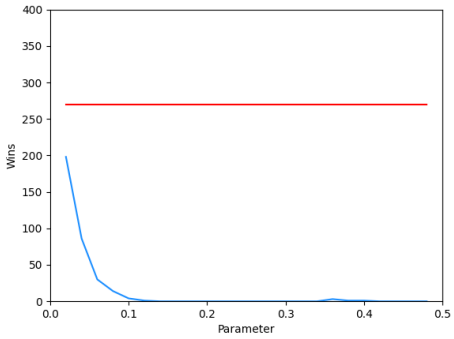
Figure G.3: Comparison of five strategies for the contestant with $\sigma = 0.299$ in the case that half of the opponents use no strategy and half of the opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

Appendix H

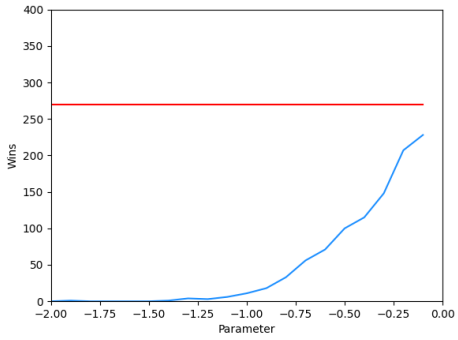
Graphs all opponents use random exponential strategy with parameter 8.956



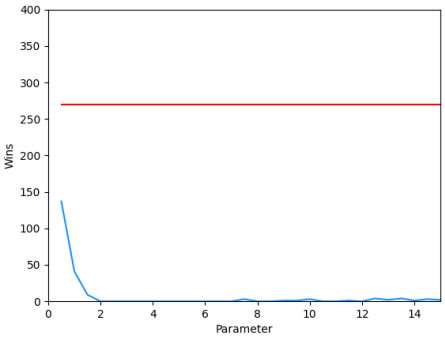
(a) Hard thresholding



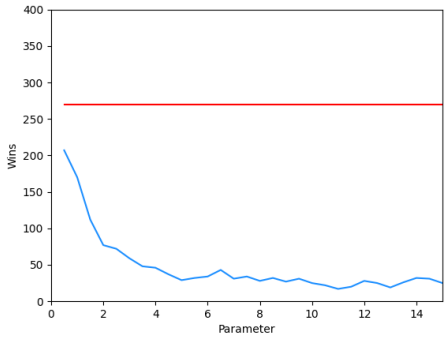
(b) Soft thresholding



(c) Polynomial strategy

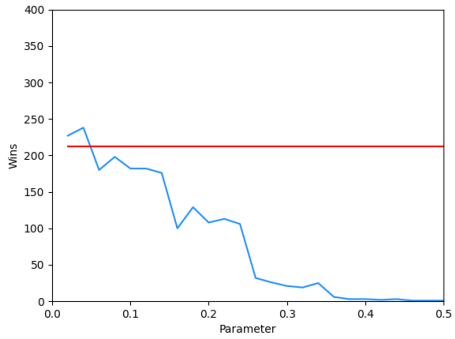


(d) Exponential strategy

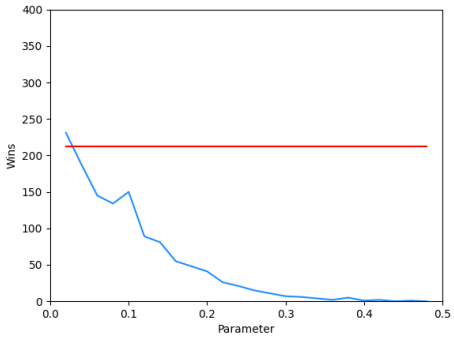


(e) Random exponential strategy

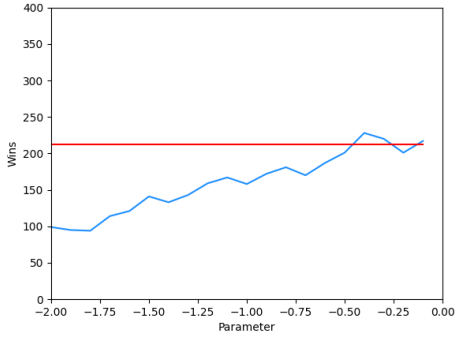
Figure H.1: Comparison of five strategies for the contestant with $\sigma = 0$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



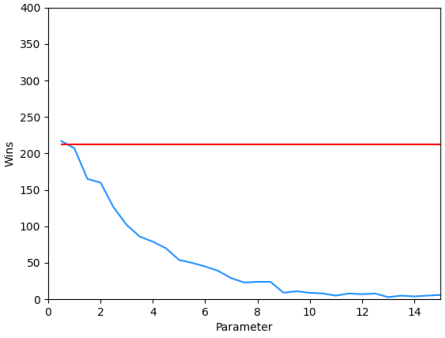
(a) Hard thresholding



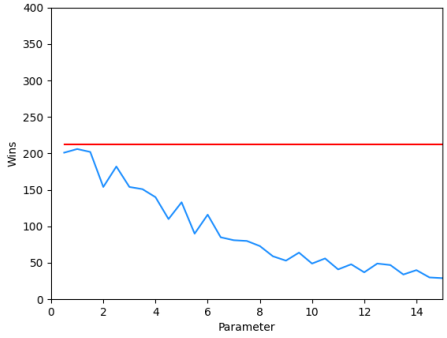
(b) Soft thresholding



(c) Polynomial strategy

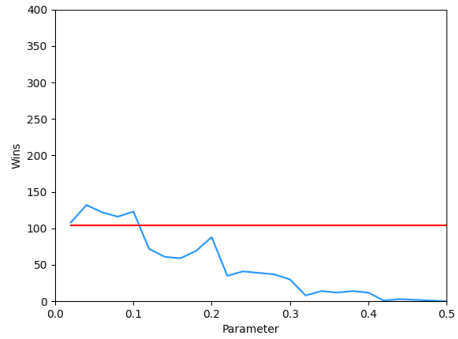


(d) Exponential strategy

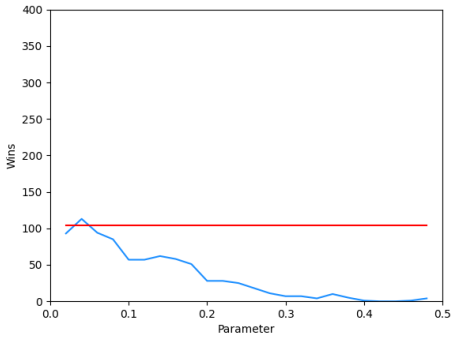


(e) Random exponential strategy

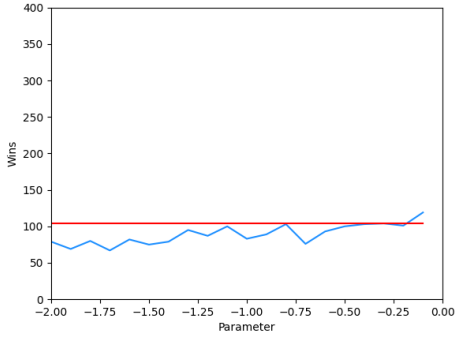
Figure H.2: Comparison of five strategies for the contestant with $\sigma = 0.1$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



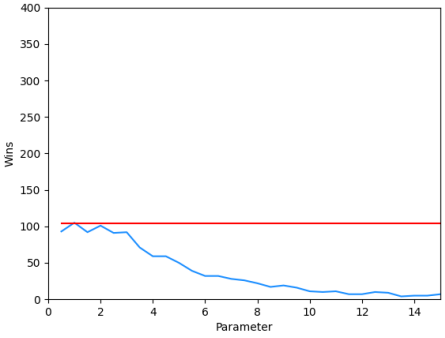
(a) Hard thresholding



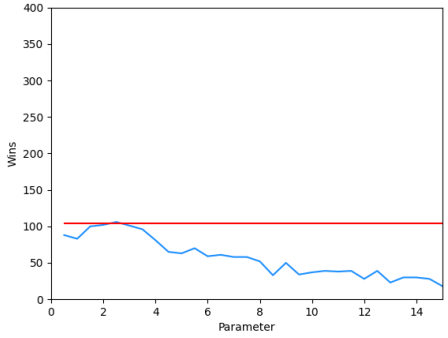
(b) Soft thresholding



(c) Polynomial strategy

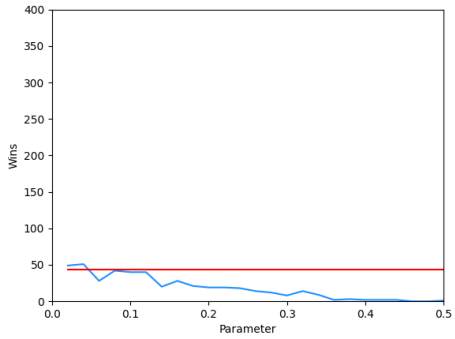


(d) Exponential strategy

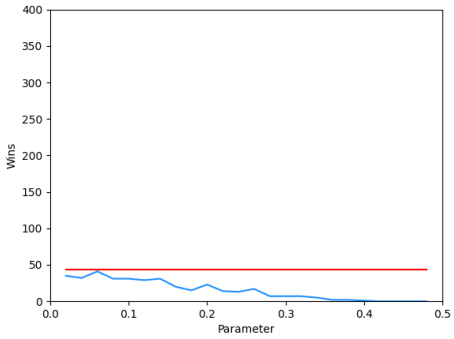


(e) Random exponential strategy

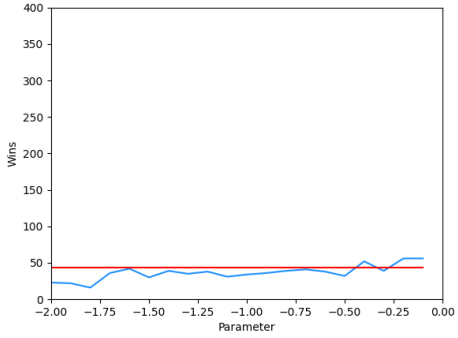
Figure H.3: Comparison of five strategies for the contestant with $\sigma = 0.15$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



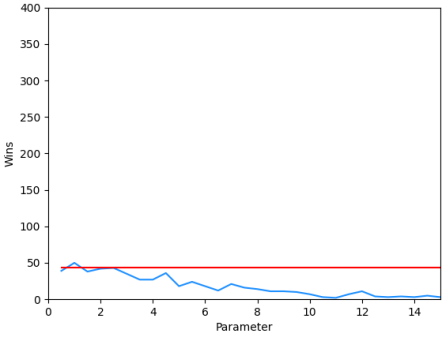
(a) Hard thresholding



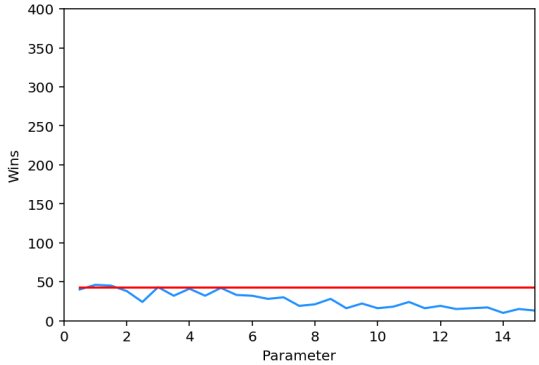
(b) Soft thresholding



(c) Polynomial strategy

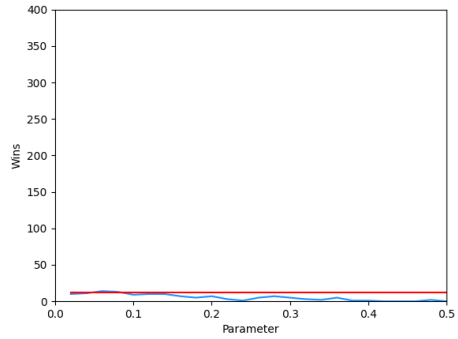


(d) Exponential strategy

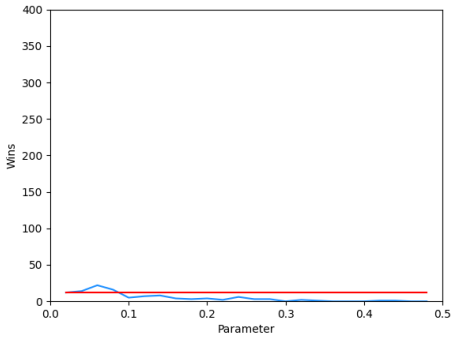


(e) Random exponential strategy

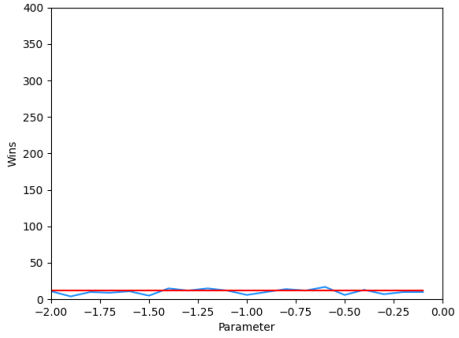
Figure H.4: Comparison of five strategies for the contestant with $\sigma = 0.2$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



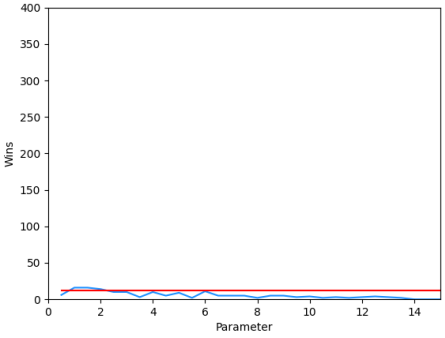
(a) Hard thresholding



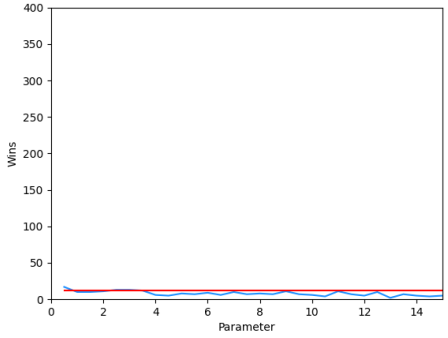
(b) Soft thresholding



(c) Polynomial strategy

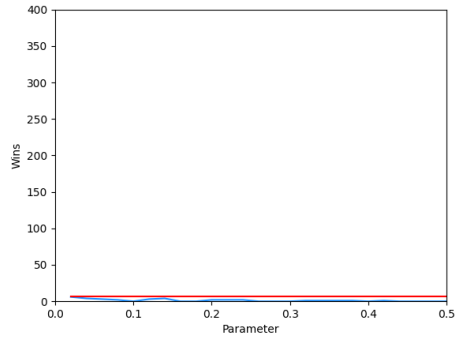


(d) Exponential strategy

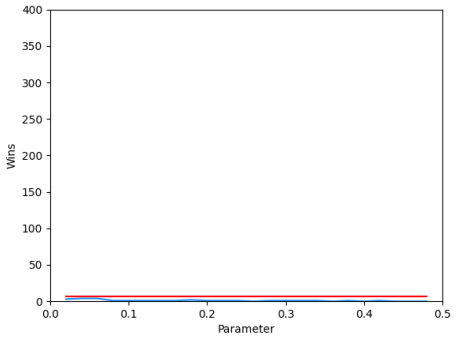


(e) Random exponential strategy

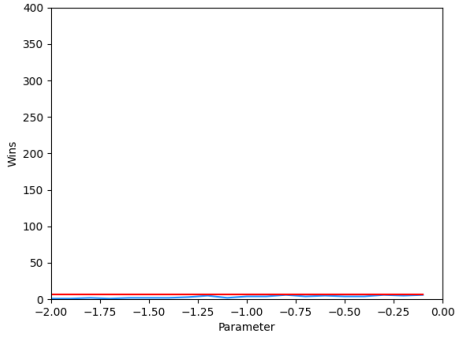
Figure H.5: Comparison of five strategies for the contestant with $\sigma = 0.25$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.



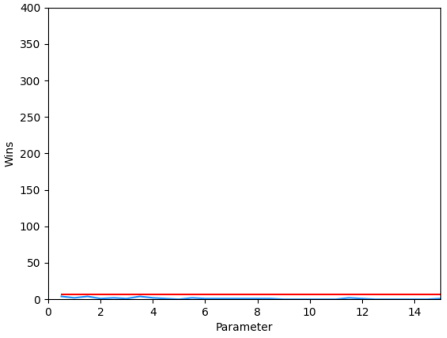
(a) Hard thresholding



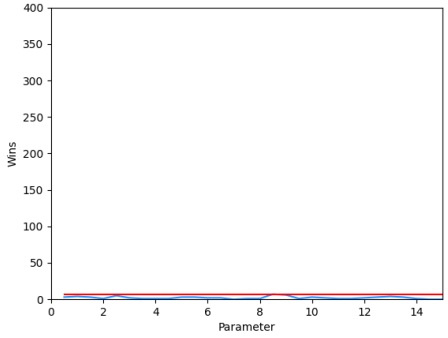
(b) Soft thresholding



(c) Polynomial strategy



(d) Exponential strategy



(e) Random exponential strategy

Figure H.6: Comparison of five strategies for the contestant with $\sigma = 0.299$ in the case that all opponents use the random exponential strategy with parameter 8.956. The red line corresponds to the number of wins of this contestant using no strategy. The number of wins in these graphs is after 10,000 tournaments.

Appendix I

Python Code Prediction Tournament

```
import numpy.random as npr

class Event:
    def __init__(self, p):
        self.p = p #the probability that the event happens
        self.outcome = 0 #wether the event happens or not

    def setOutcome(self):
        "samples an outcome for the event"
        self.outcome = npr.binomial(1, self.p) #outcome of the event

    def Outcome(self):
        "returns the outcome of the event"
        return self.outcome

    def Prob(self):
        "returns the probability of the event happening"
        return self.p

import random
import numpy.random as npr
from math import sqrt, log
import math
#import scipy.stats as stats
```

```
class Contestant:
    def __init__(self, sigma = 0):
    #contestant has a deviation sigma and a score
        self.sigma = sigma
        self.score = 0

    def calcScore(self, event, sig):
        "calculates the Brier score based on the outcome of an event"
        q = max(0, min(event.Prob() + sig, 1)) #q = p +- sig bounded to the interval [0,1]
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcScorePower(self, event, sig, b):
        "calculates the Power score based on the outcome of an event"
        q = max(0, min(event.Prob() + sig, 1)) #q = p +- sig bounded to the interval [0,1]
        if event.Outcome() == 1:
            return -b*(q**(b-1)) + (b-1)*(q**b + (1-q)**b)
        else:
            return -b*((1-q)**(b-1)) + (b-1)*(q**b + (1-q)**b)

    def calcTotalScoreOriginal(self, events):
        "calculates total score based on a sequence of events in the +- model using the Brier
score"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            s += self.calcScore(event, signs[i]*self.sigma) #and sum the scores of each event
        self.score = s
        return s

    def calcTotalScoreOriginalWithP(self, events):
        "calculates total score based on a sequence of events in the +- model using the Brier
score"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        p_predictions = []
        for i, event in enumerate(events): #we run through all events
            s += self.calcScore(event, signs[i]*self.sigma) #and sum the scores of each event
            p_predictions.append(max(0, min(event.Prob() + signs[i]*self.sigma, 1)))
        self.score = s
        return s, p_predictions
```

```
    def calcTotalScorePower(self, events, b = 50):
        "calculates total score based on a sequence of events in the +- model using the Power
score"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScorePower(event, signs[i]*self.sigma, b)
        self.score = s
        return s


#Different sigma
    def calcTotalScoreExtreme(self, events):
        "calculates total score based on a sequence of events in the +- model where
contestants make ,→ extreme predictions"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScoreExtreme(event, signs[i]*self.sigma)
        self.score = s
        return s

    def calcScoreExtreme(self, event, sig):
        "calculates the Brier score based on the outcome of an event when contestants make
extreme predictions"
        if event.Prob() + sig < 0.5:
            q = 0
        else:
            q = 1
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcTotalScoreJump(self, events, C1):
        "calculates the total score based when the contestant uses hard thresholding"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScoreJump(event, signs[i]*self.sigma, C1)
        self.score = s
```

```
        return s

    def calcScoreJump(self, event, sig, C1):
        "calculates the Brier score based on the outcome of an event when contestants use
hard thresholding"
        if event.Prob() + sig < C1:
            q = 0
        elif event.Prob() + sig > 1 - C1:
            q = 1
        else:
            q = event.Prob() + sig
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcTotalScorePolynomial(self, events, C2):
        "calculates the total score based on the outcome of an event when contestants make
predictions based on polynomials"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScorePolynomial(event, signs[i]*self.sigma, C2)
        self.score = s
        return s

    def calcScorePolynomial(self, event, sig, C2): # -2 <= C2 < 0
        "calculates the Brier score based on the outcome of an event when
contestants make predictions based on polynomials"
        p_sigma = max(0, min(event.Prob() + sig, 1))
        q = C2*(p_sigma)**3 - 1.5*C2*p_sigma*p_sigma + (1 + 0.5*C2)*p_sigma
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcTotalScoreExp(self, events, C3):
        "calculates the total score based on the outcome of an event when
contestants make predictions based on powers of e"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScoreExp(event, signs[i]*self.sigma, C3)
        self.score = s
```

```
        return s

    def calcScoreExp(self, event, sig, C3):
        "calculates the Brier score based on the outcome of an event when
contestants use powers of e"
        p_sigma = max(0, min(event.Prob() + sig, 1))
        if p_sigma > 0.5:
            q = (math.exp(-C3*p_sigma) + math.exp(-C3) - 2*math.exp(-0.5*C3))/
(2*(math.exp(-C3) - math.exp(-0.5*C3)))
        else:
            q = (math.exp(C3*p_sigma)-1)/(2*(math.exp(0.5*C3)-1))
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcTotalScoreLines(self, events, C4, C5):
        "calculates the total score based on the outcome of an event when contestants
make predictions based on 3 lines"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            #and sum the scores of each event
            s += self.calcScoreLines(event, signs[i]*self.sigma, C4, C5)
        self.score = s
        return s

    def calcScoreLines(self, event, sig, C4, C5):
        "calculates the Brier score based on the outcome of an event when contestants
make predictions based on lines"
        p_sigma = max(0, min(event.Prob() + sig, 1))
        if p_sigma < C4:
            q = (C5/C4)*p_sigma
        elif p_sigma > 1 - C4:
            q = (C5/C4)*p_sigma + 1 - (C5/C4)
        else:
            q = ((1-2*C5)/(1-2*C4))*p_sigma + (C5 - C4)/(1-2*C4)
        a = q - event.Outcome()
        return a*a #score is (q - outcome event)^2

    def calcTotalScoreRandomExp(self, events, C3):
        "calculates the total score based on the outcome of an event when contestants
make predictions based on the random exponential strategy"
        signs = random.choices([-1,1], k = len(events)) #choose a random sign for sigma
```

```
        rand = random.choices([0,1], k = len(events))
        s = 0 #score s start at 0
        for i, event in enumerate(events): #we run through all events
            if rand[i] == 1:
                s += self.calcScoreExp(event, signs[i]*self.sigma, C3)
            else:
                s += self.calcScore(event, signs[i]*self.sigma)
        self.score = s
        return s

    def Score(self):
        "returns the score of a contestant"
        return self.score

    def Sigma(self):
        "returns the deviation (sigma) of a contestant"
        return self.sigma

import numpy as np
from math import floor
#from Event import Event
#from Contestant import Contestant
#from scipy.stats import rankdata
import random

class Tournament:
    def __init__(self, probabilities, contestants):
        self.n = len(probabilities) #number of events
        self.m = len(contestants) #number of contestants
        self.probabilities = probabilities #list of probabilities for each event
        self.events = [Event(p) for p in self.probabilities] #create list of events
        self.contestants = contestants #list of contestants

    def setStandard(self):
        "set standard tournament with 100 events (p = 0.05, 0.15, ..., 0.95)
(each prob occurs 10x) \ 18 300 contestants with sigma evenly distributed on [0.05, 0.3]"

        self.n = 100
        self.m = 300
        self.probabilities = [round(0.05 + 0.1*floor(i/10), 2) for i in range(0,self.n)]
        self.events = [Event(p) for p in self.probabilities]
        self.contestants = [Contestant(round(0.05 + 0.3*i/self.m, 3)) for i in range(self.m)]
```

```
    def setStandardSigma(self, low, high, m = 300):
        "specifiy the tournament to have m contestants with sigma evenly distributed
on [low, high]"
        self.m = m
        self.contestants = [Contestant(round(low + (high-low)*i/self.m, 3)) for i in
range(self.m)]

    def setStandardProb(self):
        "specifiy the tournament to have 100 events (p = 0.05, 0.15, ..., 0.95)
(each prob occurs 10x)"
        self.n = 100
        self.probabilities = [round(0.05 + 0.1*floor(i/10), 2) for i in range(0,self.n)]
        self.events = [Event(p) for p in self.probabilities]

    def run(self, model = 'original', q = 0.5, b = 5, alpha = 1, beta = 1, C1 = 0.1, r = 0,
C2 = -1, C3 = 5, C4 = 0.2, C5 = 0):
        "runs the tournament"
        #set the outcomes for all events
        for e in self.events:
            e.setOutcome()

        #uses the appropriate model for contestant predictions and scores
        if model == 'original':
            for c in self.contestants:
                c.calcTotalScoreOriginal(self.events)

        if model == 'extreme':
            for c in self.contestants:
                c.calcTotalScoreExtreme(self.events)

        if model == 'extremeComparison':
            for index, c in enumerate(self.contestants):
                if index % 2 == 1:
                    c.calcTotalScoreExtreme(self.events)
                else:
                    c.calcTotalScoreOriginal(self.events)

        if model == 'power':
            for c in self.contestants:
                c.calcTotalScorePower(self.events, b)
```

```
if model == 'jump':
    for c in self.contestants:
        c.calcTotalScoreJump(self.events, C1)

if model == 'jumpComparison':
    for index, c in enumerate(self.contestants):
        if index % 2 == 1:
            c.calcTotalScoreJump(self.events, C1)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'jumpNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'jumpExp3.969Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        else:
            c.calcTotalScoreExp(self.events, 3.969)

if model == 'jumpRandomExp8.956Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        else:
            c.calcTotalScoreRandomExp(self.events, 8.956)

if model == 'jumpExp2.641Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        else:
            c.calcTotalScoreExp(self.events, 2.641)

if model == 'jumpHalfExp3.969HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
```

```
            c.calcTotalScoreJump(self.events, C1)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 3.969)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'jumpHalfExp2.641HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 2.641)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'jumpHalfRandomExp8.956HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreJump(self.events, C1)
        elif index % 2 == 1:
            c.calcTotalScoreRandomExp(self.events, 8.956)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'polynomial':
    for c in self.contestants:
        c.calcTotalScorePolynomial(self.events, C2)

if model == 'polynomialNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'polynomialExp3.969Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        else:
            c.calcTotalScoreExp(self.events, 3.969)
```

```
if model == 'polynomialRandomExp8.956Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        else:
            c.calcTotalScoreRandomExp(self.events, 8.956)

if model == 'polynomialExp2.641Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        else:
            c.calcTotalScoreExp(self.events, 2.641)

if model == 'polynomialComparison':
    for index, c in enumerate(self.contestants):
        if index % 2 == 1:
            c.calcTotalScorePolynomial(self.events, C2)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'polynomialHalfExp3.969HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 3.969)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'polynomialHalfExp2.641HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScorePolynomial(self.events, C2)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 2.641)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'polynomialHalfRandomExp8.956HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
```

```
            c.calcTotalScorePolynomial(self.events, C2)
        elif index % 2 == 1:
            c.calcTotalScoreRandomExp(self.events, 8.956)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'exp':
    for c in self.contestants:
        c.calcTotalScoreExp(self.events, C3)

if model == 'expNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'expExp3.969Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        else:
            c.calcTotalScoreExp(self.events, 3.969)

if model == 'expExp2.641Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        else:
            c.calcTotalScoreExp(self.events, 2.641)

if model == 'expRandomExp8.956Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        else:
            c.calcTotalScoreRandomExp(self.events, 8.956)

if model == 'expComparison':
    for index, c in enumerate(self.contestants):
        if index % 2 == 1:
            c.calcTotalScoreExp(self.events, C3)
```

```
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'expHalfExp3.969HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 3.969)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'expHalfExp2.641HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 2.641)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'expHalfRandomExp8.956HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreRandomExp(self.events, 8.956)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'randomExpNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'randomExpExp3.969Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        else:
```

```
            c.calcTotalScoreExp(self.events, 3.969)

if model == 'randomExpExp2.641Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        else:
            c.calcTotalScoreExp(self.events, 2.641)

if model == 'randomExpRandomExp8.956Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        else:
            c.calcTotalScoreRandomExp(self.events, 8.956)

if model == 'randomExpHalfExp2.641HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 2.641)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'randomExpHalfExp3.969HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreExp(self.events, 3.969)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'randomExpHalfRandomExp8.956HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreRandomExp(self.events, C3)
        elif index % 2 == 1:
            c.calcTotalScoreRandomExp(self.events, 8.956)
        else:
            c.calcTotalScoreOriginal(self.events)
```

```
if model == 'lines':
    for c in self.contestants:
        c.calcTotalScoreLines(self.events, C4, C5)

if model == 'linesNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreLines(self.events, C4, C5)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'linesExp3.969Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreLines(self.events, C4, C5)
        else:
            c.calcTotalScoreExp(self.events, 3.969)

if model == 'linesExp2.641Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreLines(self.events, C4, C5)
        else:
            c.calcTotalScoreExp(self.events, 2.641)

if model == 'linesRandomExp8.956Comparison_1-299':
    for index, c in enumerate(self.contestants):
        if index == r:
            c.calcTotalScoreLines(self.events, C4, C5)
        else:
            c.calcTotalScoreRandomExp(self.events, 8.956)

if model == 'linesComparison':
    for index, c in enumerate(self.contestants):
        if index % 2 == 1:
            c.calcTotalScoreLines(self.events, C4, C5)
        else:
            c.calcTotalScoreOriginal(self.events)

if model == 'linesHalfExp3.969HalfNormalComparison_1-299':
    for index, c in enumerate(self.contestants):
```

```
            if index == r:
                c.calcTotalScoreLines(self.events, C4, C5)
            elif index % 2 == 1:
                c.calcTotalScoreExp(self.events, 3.969)
            else:
                c.calcTotalScoreOriginal(self.events)

    if model == 'linesHalfExp2.641HalfNormalComparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreLines(self.events, C4, C5)
            elif index % 2 == 1:
                c.calcTotalScoreExp(self.events, 2.641)
            else:
                c.calcTotalScoreOriginal(self.events)

    if model == 'linesHalfRandomExp8.956HalfNormalComparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreLines(self.events, C4, C5)
            elif index % 2 == 1:
                c.calcTotalScoreRandomExp(self.events, 8.956)
            else:
                c.calcTotalScoreOriginal(self.events)

    if model == 'normalExp3.969Comparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreOriginal(self.events)
            else:
                c.calcTotalScoreExp(self.events, 3.969)

    if model == 'normalRandexp8.956Comparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreOriginal(self.events)
            else:
                c.calcTotalScoreRandomExp(self.events, 8.956)

    if model == 'normalExp2.641Comparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
```

```
                c.calcTotalScoreOriginal(self.events)
            else:
                c.calcTotalScoreExp(self.events, 2.641)

    if model == 'normalHalfExp3.969HalfNormalComparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreOriginal(self.events)
            elif index % 2 == 1:
                c.calcTotalScoreExp(self.events, 3.969)
            else:
                c.calcTotalScoreOriginal(self.events)

    if model == 'normalHalfExp2.641HalfNormalComparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreOriginal(self.events)
            elif index % 2 == 1:
                c.calcTotalScoreExp(self.events, 2.641)
            else:
                c.calcTotalScoreOriginal(self.events)

    if model == 'normalHalfRandomExp8.956HalfNormalComparison_1-299':
        for index, c in enumerate(self.contestants):
            if index == r:
                c.calcTotalScoreOriginal(self.events)
            elif index % 2 == 1:
                c.calcTotalScoreRandomExp(self.events, 8.956)
            else:
                c.calcTotalScoreOriginal(self.events)



def probabilities(self):
    "returns the success probabilities of the tournament"
    return [event.Prob() for event in self.events]

def outcomes(self):
    "returns the outcomes of the events in the tournament"
    return [event.Outcome() for event in self.events]

def scoreContestants(self):
```

```
        "returns the scores of the contestants in the tournament"
        return [c.Score() for c in self.contestants]

    def sigmaContestants(self):
        "returns the deviations(sigma) of the contestants in the tournament"
        return [c.Sigma() for c in self.contestants]

    def rankWinner(self):
        "returns the rank of the contestant with the lowest score
(the winner of the tournament)"
        #return np.flatnonzero(self.scoreContestants() == np.min(self.scoreContestants()))
        return np.argmin(self.scoreContestants())

    def allWinners(self):
        "returns all contestants with the same lowest score)"
        return np.flatnonzero(self.scoreContestants() == np.min(self.scoreContestants()))

    def randomWinner(self):
        "if multiple people have the same lowest score, we randomly select a winner
from those"
        return random.choice(self.allWinners())

    def rankFinishPos(self, n):
        "returns the rank of the contestant with the lowest score (the winner of
the tournament)"
        return np.argsort(self.scoreContestants())[n]

#113 def rankTopN(self, n=10):
#114 "returns the rank of top n finishers"
#115 return np.argsort(self.scoreContestants())[0:n]

    def rankAll(self):
        "returns the rank for all finish positions"
        return np.argsort(self.scoreContestants())

#     def finishPosAll(self):
#         "returns the finish position for all contestants"
#         return (rankdata(self.scoreContestants()) - 1).astype(int)

import matplotlib.pyplot as plt
import time
#from Tournament import Tournament
```

```
import numpy.random as npr
import numpy as np
#import scipy.stats as stats
from datetime import datetime

def Simulate(n = 4000, low = 0, high = 0.3, m = 300, topN = False, N = 10, model = 'original',
prob = 'standard', q = 0.5, b = 5, filename = None, alpha = 1, beta = 1, C1 = 0.1, r =0,
C2 = -1, C3 = 5, C4 = 0.2, C5 = 0):
    ranks = []

    p = []

    if prob == 'precise':
        p = [i/100 for i in range(100)]

    #save results in a text file
#    if filename != None:
#    f = open(f'allRanks{filename}.txt', 'w')
#    f1 = open(f'Scores{filename}.txt', 'w')
#    f2 = open(f'finishPos{filename}.txt', 'w')

    #options for the true success probilities
    t = Tournament(p, [])
    t.setStandardSigma(low, high, m)

    if prob == 'standard':
        t.setStandardProb()

    if model == 'extremeComparison':
        even = []
        odd = []
    if model == 'jumpComparison':
        even = []
        odd = []
    if model == 'polynomialComparison':
        even = []
        odd = []
    if model == 'expComparison':
        even = []
        odd = []
    if model == 'linesComparison':
        even = []
```

```
        odd = []

    if topN == False:
        for i in range(n):
            t.run(model, q, b, alpha, beta, C1, r, C2, C3, C4, C5)
            winner = t.randomWinner()
            ranks.append(winner)
            #ranks.append(t.rankFinishPos(14))
            if model == 'extremeComparison':

                if winner % 2 == 1:
                    odd.append(winner)
                else:
                    even.append(winner)
            if model == 'jumpComparison':
                if winner % 2 == 1:
                    odd.append(winner)
                else:
                    even.append(winner)
            if model == 'polynomialComparison':
                if winner % 2 == 1:
                    odd.append(winner)
                else:
                    even.append(winner)
            if model == 'expComparison':
                if winner % 2 == 1:
                    odd.append(winner)
                else:
                    even.append(winner)
            if model == 'linesComparison':
                if winner % 2 == 1:
                    odd.append(winner)
                else:
                    even.append(winner)
#             if filename != None:
#76 f.write(str(list(t.finishPosAll())) + '\n')
#77 f1.write(str(t.scoreContestants()) + '\n')
#78 a = str(list(t.rankAll()))
#79 l1 = a.replace('[', '')
#80 l2 = l1.replace(']', '')
#81 items = l2.split(',')
#82 a = [float(item) for item in items]
```

```
#83 if len(a) != 300:
#84 print(len(a))
#85 f2.write(str(list(t.rankAll())) + '\n')
#86 if topN == True:
#87 for i in range(n):
#88 t.run(model, q, b)
#89 ranks.extend(t.rankTopN(N))
    plt.figure(2, dpi = 300)
    plt.hist(ranks, bins = range(0, 300 + 10, 10), color = 'dodgerblue', edgecolor='k')
    plt.xlim(xmin=0, xmax = 300)

    if model == 'extremeComparison':
        plt.figure(3, dpi = 300)
        plt.hist([even, odd], bins = range(0, 300 + 10, 10), color = ['dodgerblue', 'orange'],
label = ['Normal predictions', 'Extreme predictions'])
        plt.xlim(xmin=0, xmax = 300)
        plt.legend(loc = 'upper right')
    if model == 'jumpComparison':
        plt.figure(3, dpi = 300)
        plt.hist([even, odd], bins = range(0, 300 + 10, 10), color = ['dodgerblue', 'orange'],
label = ['Normal predictions', 'Jump predictions'])
        plt.xlim(xmin=0, xmax = 300)
        plt.legend(loc = 'upper right')
    if model == 'polynomialComparison':
        plt.figure(3, dpi = 300)
        plt.hist([even, odd], bins = range(0, 300 + 10, 10), color = ['dodgerblue', 'orange'],
label = ['Normal predictions', 'Polynomial predictions'])
        plt.xlim(xmin=0, xmax = 300)
        plt.legend(loc = 'upper right')
    if model == 'expComparison':
        plt.figure(3, dpi = 300)
        plt.hist([even, odd], bins = range(0, 300 + 10, 10), color = ['dodgerblue', 'orange'],
label = ['Normal predictions', 'Exponential predictions'])
        plt.xlim(xmin=0, xmax = 300)
        plt.legend(loc = 'upper right')
    if model == 'linesComparison':
        plt.figure(3, dpi = 300)
        plt.hist([even, odd], bins = range(0, 300 + 10, 10), color = ['dodgerblue', 'orange'],
label = ['Normal predictions', 'Lines predictions'])
        plt.xlim(xmin=0, xmax = 300)
        plt.legend(loc = 'upper right')
```

```
    plt.show()

#104 if filename != None:
#105 f.close()
#106 f1.close()
#107 f2.close()

def Sim_number_of_wins(n = 4000, low = 0, high = 0.3, m = 300, model = 'original',
prob = 'standard', q = 0.5, b = 5, filename = None, alpha = 1, beta = 1, C1 = 0.1, r = 0,
C2 = -1, C3 = 5, C4 = 0.2, C5 = 0):
    count = 0
    p = []

    if prob == 'precise':
        p = [i/100 for i in range(100)]

    #options for the true success probilities
    t = Tournament(p, [])
    t.setStandardSigma(low, high, m)

    if prob == 'standard':
        t.setStandardProb()

    for i in range(n):
        t.run(model, q, b, alpha, beta, C1, r, C2, C3, C4, C5)
        winner = t.randomWinner()
        if winner == r:
            count += 1
    return count

def Plot_wins_jump(r = 0):
    lst = []
    x_vals = []
    for i in range(10):
        lst.append(Sim_number_of_wins(500, model = 'jumpNormalComparison_1-299', r = 0,
C1 = 0.05+i*0.05))
        x_vals.append(0.05+i*0.05)
    plt.plot(x_vals,lst)

def Plot_wins_polynomial_with_av_normal(R = 0):
    lst = []
    x_vals = [-2+i*0.1 for i in range(20)]
```

```
    norm_av = [0 for i in range(20)]
    for i in range(20):
        abc = 0
        for j in range(3):
            abc += Sim_number_of_wins(3000, model = 'polynomialNormalComparison_1-299', r = R,
C2 = -2+i*0.1)
        lst.append(round(abc/3))
        print(round(abc/3))
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.show()

def Plot_wins_jump_with_av_normal(R = 0):
    lst = []
    x_vals = [0.02+i*0.02 for i in range(25)]
    norm_av = [0 for i in range(25)]
    for i in range(25):
        abc = 0
        for j in range(3):
            abc += Sim_number_of_wins(3000, model = 'jumpNormalComparison_1-299', r = R,
C1 = 0.02+i*0.02)
        lst.append(round(abc/3))
        print(round(abc/3))
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 0.5)
    plt.ylim(ymin = 0,ymax = 60)
    plt.show()

def Plot_wins_exp_with_av_normal(R = 0):
    lst = []
    x_vals = [0.5+i*0.5 for i in range(30)]
    norm_av = [0 for i in range(30)]
    for i in range(30):
```

```
        abc = 0
        for j in range(3):
            abc += Sim_number_of_wins(3000, model = 'expNormalComparison_1-299', r = R,
C3 = 0.5+i*0.5)
        lst.append(round(abc/3))
        print(round(abc/3))
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 15)
    plt.ylim(ymin = 0,ymax = 60)
    plt.show()

def Plot_wins_randexp_with_av_normal(R = 0):
    lst = []
    x_vals = [0.5+i*0.5 for i in range(30)]
    norm_av = [10 for i in range(30)]
    for i in range(30):
        abc = 0
        for j in range(3):
            abc += Sim_number_of_wins(3000, model = 'randomExpNormalComparison_1-299', r = R,
C3 = 0.5+i*0.5)
        lst.append(round(abc/3))
        print(round(abc/3))
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 15)
    plt.ylim(ymin = 0,ymax = 60)
    plt.show()

def Plot_wins_lines_with_av_normal(R = 0):
    lst = []
    x_vals = [0.02+i*0.02 for i in range(24)]
    norm_av = [0 for i in range(24)]
    for i in range(24):
```

```
        abc = 0
        for j in range(3):
            abc += Sim_number_of_wins(3000, model = 'linesNormalComparison_1-299', r = R,
C4 = 0.02+i*0.02, C5=0)
        lst.append(round(abc/3))
        print(round(abc/3))
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 0.5)
    plt.ylim(ymin = 0,ymax = 60)
    plt.show()

def Plot_wins_exp_with_av_normal_nauwkeuriger(R = 0):
    lst = []
    x_vals = [3 + i*0.02 for i in range(51)]
    norm_av = [204 for i in range(51)]
    for i in range(51):
        abc = Sim_number_of_wins(40000, model = 'expNormalComparison_1-299', r = R,
C3 = 3 + i*0.02)
        lst.append(abc)
        print(abc)
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 3, xmax = 4)
    plt.ylim(ymin = 0, ymax = 800)
    plt.show()


def Plot_wins_polynomial_with_av_normal_exp3969comparison(R = 0):
    lst = []
    x_vals = [-2+i*0.1 for i in range(20)]
    norm_av = [304 for i in range(20)]
    for i in range(20):
        abc = Sim_number_of_wins(10000, model = 'polynomialExp3.969Comparison_1-299', r = R,
```

```
C2 = -2+i*0.1)
        lst.append(abc)
        print(abc)
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = -2, xmax = 0)
    plt.ylim(ymin = 0, ymax = 400)
    plt.show()

def Plot_wins_jump_with_av_normal_exp3969comparison(R = 0):
    lst = []
    x_vals = [0.02+i*0.02 for i in range(25)]
    norm_av = [304 for i in range(25)]
    for i in range(25):
        abc = Sim_number_of_wins(10000, model = 'jumpExp3.969Comparison_1-299', r = R,
C1 = 0.02+i*0.02)
        lst.append(abc)
        print(abc)
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 0.5)
    plt.ylim(ymin = 0, ymax = 400)
    plt.show()

def Plot_wins_exp_with_av_normal_exp3969comparison(R = 0):
    lst = []
    x_vals = [0.5+i*0.5 for i in range(30)]
    norm_av = [304 for i in range(30)]
    for i in range(30):
        abc = Sim_number_of_wins(10000, model = 'expExp3.969Comparison_1-299', r = R,
C3 = 0.5+i*0.5)
        lst.append(abc)
        print(abc)
        print(datetime.now())
```

```
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 15)
    plt.ylim(ymin = 0, ymax = 400)
    plt.show()

def Plot_wins_randexp_with_av_normal_exp3969comparison(R = 0):
    lst = []
    x_vals = [0.5+i*0.5 for i in range(30)]
    norm_av = [304 for i in range(30)]
    for i in range(30):
        abc = Sim_number_of_wins(10000, model = 'randomExpExp3.969Comparison_1-299', r = R,
C3 = 0.5+i*0.5)
        lst.append(abc)
        print(abc)
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 15)
    plt.ylim(ymin = 0, ymax = 400)
    plt.show()

def Plot_wins_lines_with_av_normal_exp3969comparison(R = 0):
    lst = []
    x_vals = [0.02+i*0.02 for i in range(24)]
    norm_av = [304 for i in range(24)]
    for i in range(24):
        abc = Sim_number_of_wins(10000, model = 'linesExp3.969Comparison_1-299', r = R,
C4 = 0.02+i*0.02, C5=0)
        lst.append(abc)
        print(abc)
        print(datetime.now())
    print(lst)
    plt.plot(x_vals, lst, color = 'dodgerblue')
    plt.plot(x_vals, norm_av, color = 'red')
    plt.xlabel(xlabel='Parameter')
```

```
    plt.ylabel(ylabel='Wins')
    plt.xlim(xmin = 0, xmax = 0.5)
    plt.ylim(ymin = 0, ymax = 400)
    plt.show()



print(datetime.now())
Plot_wins_randexp_with_av_normal_exp3969comparison(R = 0)
print(datetime.now())
```

Appendix J

Python Code Polynomial Regression

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import sklearn
from sklearn.preprocessing import PolynomialFeatures

x = np.array([-1.75 + i*0.02 for i in range(51)])
y = np.array([189, 187, 200, 244, 196, 198, 206, 215, 201, 196, 217, 187,
188, 213, 207, 228, 198, 203, 204, 219, 210, 217, 191, 211, 190, 237, 197,
188, 180, 202, 197, 222, 200, 210, 205, 237, 189, 192, 209, 209, 188, 184,
193, 201, 194, 231, 183, 213, 178, 174, 170])
norm_av = np.array([167 for i in range(51)])
x= x[:,np.newaxis]

for i in range(2,11):
    polynomial_features= PolynomialFeatures(degree=i)
    xp = polynomial_features.fit_transform(x)
    model = sm.OLS(y, xp).fit()
    print("If the degree is", i, "then the aic is", model.aic)

polynomial_features= PolynomialFeatures(degree=2)
xp = polynomial_features.fit_transform(x)
model = sm.OLS(y, xp).fit()
ypred = model.predict(xp)
plt.plot(x,y,color='dodgerblue')
```

```
plt.plot(x,ypred,color='orange')
plt.plot(x, norm_av, color='red')
plt.xlabel(xlabel='Parameter')
plt.ylabel(ylabel='Wins')
plt.xlim(xmin=-1.75, xmax=-0.75)
plt.ylim(ymin=0, ymax=600)
plt.show()
print(model.summary())
```