The Game Theory of Penalty Kicks

A framework for approximating Nash equilibria

By Vebjørn Monstad

Abstract

This paper presents a game-theoretic analysis of penalty kicks in football. There must exist a Nash equilibrium in penalty kicks, but the nature of it depends on the unique abilities of the penalty taker and the goalkeeper. Thus, this study introduces a flexible framework that approximates a Nash equilibrium, based on a user-inputted set of player-dependent assumptions. The study also seeks to improve our general understanding of the characteristics of Nash equilibria in penalty kicks. This is achieved by investigating a diverse set of player-dependent assumptions, as well as gradually adding new elements of complexity to the models, and observing the shifts in the equilibrium.

In the most basic model, the players make a simultaneous choice, where the penalty taker decides where to aim, and the goalkeeper decides which area to cover. The penalty taker isn't able to shoot with perfect accuracy, so the hit-coordinate will likely differ from the aim-coordinate. In later models, the penalty taker is also allowed to choose the velocity of the ball, which in turn impacts the area coverage of the goalkeeper. In the final model, an element of sequential choice is added, such that the penalty taker may pretend to shoot, and observe if the goalkeeper starts to move.

The Counterfactual Regret Minimization algorithm is employed to locate the Nash equilibrium, while an enhanced Coordinate Search algorithm is developed for determining the optimal aim-coordinates for the penalty taker. The most complex and realistic model indicates that the penalty taker should abstain from aiming at the middle region of the goal, and rather either pretend to shoot, or shoot at one of the sides. This is because the goalkeeper needs to stay in the middle fairly often, to avoid revealing their intention in the case where the penalty taker pretends to shoot. It's also viable for the goalkeeper to commit to diving to either side without waiting to observe the trajectory of the ball.

Table of Contents

Introduction 1

Background and context 5

 Game theory 6

  Definition 7

  A simple toy game 9

  Nash equilibrium 13

 Counterfactual regret minimization 17

Framework development 21

 Simultaneous choice 22

 Domains 23

 Area coverage 25

 Hit-coordinates 28

 Aim-coordinates 30

  Payoffs 31

  Symmetry 32

  An initial result 33

 Coordinate search 34

 Leaning options 43

 Non-uniform velocity 51

 Sequential choices 53

 Asymmetry 56

Conclusion 59

References 62

Appendix I 64

Appendix II 66

Appendix III 69

Appendix IV 73

Introduction

Decision modelling involves formulating, analysing, and solving models, with the aim of informing our decisions, such that we can better satisfy our objective. Game theory is a branch of decision modelling that focuses on strategic interaction between rational agents. This paper presents a game-theoretic analysis of penalty kicks, focusing on the strategic interaction between the penalty taker and the goalkeeper. The study aims to create a framework for approximating Nash equilibrium strategies for both players involved.

Football is widely considered to be the most popular sport in the world, with billions of fans across the globe. Penalty kicks is a highly important subgame of football that can often directly determine the outcome of the game. In high stakes matches such as knockout rounds of major international tournaments or cup finals, penalty shootouts are the final deciding factor in crowning a champion, if the game is drawn. This happened as lately as in the 2022 FIFA world cup, where Argentina beat France in the final by winning in a penalty shootout. With the size of the industry, even small improvements of strategy may be worth millions of dollars in certain situations, making it a significant area of study.

It's clear that penalty kicks can be vitally important in determining the outcome of a football match. Several studies have explored various aspects of penalty kicks, such as the effectiveness of different techniques or the psychology of penalty takers and goalkeepers. However, there's still much to be explored in terms of the optimal strategic interaction between the two players. There have also been some studies that have analysed penalty kicks from a game-theoretic perspective, but these models have been very simple, and not comprehensive attempts at estimating the characteristics of the optimal play. Therefore, there exist a gap in the literature that this study intends to fill.

Penalty kicks is an extremely complex game where, apart from a few rule-based constraints, the strategy space for each player is only really limited by the laws of physics, as well as the inherent capabilities of the players. Yet, some strategies are certainly better than others, and there should exist a mixed strategy Nash equilibrium for the game. The sheer complexity of the game might make the true nature of the equilibrium unknowable to us, but that doesn't mean that it doesn't exist. Nor does it mean that we can't gain insights about it or make approximations of what it might look like, which is one of the main objectives in the study.

To make such approximations, it's necessary to reduce the strategy space down to something manageable. In other words, we need create a simplified model the game. One way of understanding this is that the model will be a new game, which is different from penalty kicks, but it will be a simpler game that imitates the core features of the real game. This will make the model solvable, meaning that we will be able to determine which strategies are optimal. The results generated by the model will then serve as an approximation of what might be optimal in the real game. Notice that the quality of the approximations is strictly contingent on being able to create a model that successfully imitates the core features of penalty kicks. This is the primary limitation of the study.

In this research I'll construct several models of increasing complexity, each of them building on the foundation of the former. The first model is the simplest and establishes the fundamentals of the framework. This model discards all but the most essential factors of a penalty kick. For the penalty taker, the strategic options are limited to selecting a coordinate point to aim for inside the goal. The penalty taker's aim-coordinate, which represents the target inside the goal, may differ from the hit-coordinate due to the penalty taker's inherent inaccuracy in shooting. To incorporate this inaccuracy, the coordinates where the ball hits are drawn from a normal distribution along both the horizontal and vertical axes, using the aim-coordinates as the mean of the distribution. For the goalkeeper, the strategic options are limited to which area of the goal to cover – for example one area on the left side, one area in the middle, or one area on the right side. The model assumes that the penalty taker and the goalkeeper make their choice simultaneously. If the hit-coordinates are within the goal, and not within the area that the goalkeeper has chosen to cover, the result is a goal, and otherwise the result is a miss.

Penalty kicks is a zero-sum game, where the penalty taker is awarded a payoff of 1 if the penalty results in a goal, and 0 if it doesn't. The goalkeeper receives a payoff of -1 if conceding a goal, and 0 otherwise. Given all the options available to the players, we can simulate the expected payoffs associated with each pair of options. This allows us to measure how well two strategies perform against one another. This is a good starting point, but not sufficient if we want to effectively learn how to improve the strategies. To pinpoint which strategies are optimal, we need a self-improving mechanism that continually updates the strategies such that we have a convergence towards the Nash equilibrium. To achieve this, I'm implementing a machine learning algorithm used for solving imperfect- information games, called counterfactual regret minimization. The Counterfactual Regret Minimization algorithm is a reinforcement learning technique that minimizes the regret of actions in previous iterations, gradually converging towards the Nash equilibrium. This is the methodology used for locating the equilibrium in all the models. I also develop an algorithm, called coordinate search, which locates the optimal aim-coordinates within the goal, even if those coordinates weren't part of the original options.

In a later model, the strategic options available to the penalty taker are extended to incorporate a choice of velocity. The velocity scales and moves the areas that the goalkeeper chooses between covering. Generally, if the ball travels at a high velocity, the area is reduced in size, and if the ball travels slowly, the area gets expanded. Additionally, there is a trade-off between velocity and accuracy. A higher velocity results in a lower positional accuracy, and vice versa. The area of the goal that the goalkeeper is able to cover is also dependent on the velocity of the ball. Additionally, if the ball travels along the ground, the velocity is reduced to account for friction. This effectively serves as a disincentive to aiming at a low vertical coordinate point.

The final model attempts to better account for sequential choices. It extends the strategic options of the penalty taker to include an option of first deciding whether to shoot normally, or to pretend to shoot. If the penalty taker decides to shoot normally, we are essentially entering the scenario of the second model. If the penalty taker pretends to shoo, we are also in a similar situation as that of the second model, except the penalty taker becomes limited to shooting with a low velocity, resulting in a significant handicap. This is because it's illegal for the penalty taker to back up and start the run-up again, meaning that the shot now has to be made while standing still.

While pretending to shoot limits the penalty taker to shooting with a low velocity, it also has a big potential upside. The goalkeeper may start to move either to the left or the right, giving the penalty taker solid information about which area of the goal the goalkeeper won't be able to cover. For example, if the penalty taker fakes the shot, and the goalkeeper starts moving to the left, the penalty taker can easily score by aiming to the right. In this case, the handicap of having to shoot with a low velocity will likely be irrelevant. However, if the goalkeeper stays in the middle, the penalty taker is essentially forced to play a version of the second model which is disadvantageous. Thus, if the penalty taker incorporates faking the shot into their strategy, the goalkeeper is incentivized to stay in the middle more often, making it more advantageous for the penalty taker to shoot normally and aim to a side.

It's important to understand that the objective of this study is to create a framework for approximating Nash equilibria in penalty kicks, and not to find a specific solution that is deemed to be optimal in penalty kicks in general. This is because the Nash equilibrium of the game is strictly dependent on the inherent capabilities of the players. Some penalty takers will be able to shoot more accurately than others, and some goalkeepers will be able to cover a larger area of the goal than others. Additionally, even the same players may not be able to perform at the same level in all situations. For example, I would expect players to be able to shoot much more accurately in normal league game, compared to in a penalty shootout in the world cup final, when stakes are high, and players are likely very nervous. This means that there are no assumptions about accuracy or goal area that should be viewed as *correct*. Rather, some assumptions fit some players in some situations, and other assumptions fit other players in other situations. Consequently, another main objective is to design framework such that the user can input the player-dependent assumptions that they think fit their particular set of players.

Background and context

Football, or soccer as it's called in some regions, is a globally renowned sport. A match involves two teams, each composed of eleven players, competing on a rectangular field with a goal at each end. The field is divided in two, where each team defends one half of the field, and attacks the other. The objective for each team is to manoeuvre a spherical ball into the goal situated at the end of the opposing team's half. If a player achieves this, that player has scored a goal. The duration of a match is contained by a time frame, which generally is 90 minutes. The team that has scored the most goals at the end of the time frame wins the match. This means that teams are not only interested in scoring goals, but also to prevent the opposing team from doing so.

A penalty kick is a special subgame of football which occurs when a player commits a foul within their own penalty area, which is a marked rectangle in front of their goal. The subgame is a duel between two players – one from each team: the penalty taker and the goalkeeper. In the penalty kick, the goalkeeper from the team that committed the foul must defend the goal. The ball is placed on a dot, only 12 yards (approximately 11 meters) away from the centre of the goal. The other team gets to choose a penalty taker who will kick the ball and attempt to score a goal.

Various websites (e.g., transfermarkt.co.uk, myfootballfacts.com) track all penalty kicks that have been awarded in the English Premier League since the league's inception in 1992. In the current format of the league, 380 matches are played every season. In the ten seasons between the 11/12 and 21/22 season, about 105 penalties were awarded per season on average. This means that a penalty kick has occurred once about every 326th minutes of playtime on average, which isn't that rare, considering that each match lasts 90 minutes. Throughout these ten seasons, about 78.7% of the penalty kicks resulted in a goal. According to myfootballfacts.com, in the remaining cases, the penalty taker failed to score either as a result of the goalkeeper stopping the shot, which happened 16.1% of the time, or the ball hitting outside of the goal, which happened 5.2% of the time.

In an analysis of penalty kicks in football after the year 1997, Dalton et al. (2015) finds that in matches where a penalty kick was awarded, the team that was awarded the penalty kick ended up winning in 52.3% of the cases, and only losing 17.5% of the time. Furthermore, if the penalty kick resulted in a goal, the team won in 60.9% of the cases, and only lost 10.9% of the time. If the team failed to convert the penalty, the win-rate dropped to a mere 29.4%, and the chance of a loss increased to a staggering 35.3%.

These dramatic shifts in win-rates demonstrate how vitally important a penalty kick can be to the outcome of a football match. Consequently, penalty kicks have been subject to extensive study, encompassing diverse methodologies and areas of focus. Many studies have focused on statistical analysis of empirical data. This includes the study mentioned earlier, as well as works by Jordet et al. (2007), who analysed the impact of stress, skill, and fatigue on a penalty kick, or Bar-Eli et al. (2007) who investigated the determinants of success in penalty kicks, to name a few. Other studies, such as those conducted by Visscher et al. (2006), and Jordet et al. (2009), have focused on the psychological aspects of penalty kicks.

Game theory has also been applied to the study of penalty kicks. One such notable study is by Chiappori et al. (2002). In their research, the authors construct a model of penalty kicks centred around the idea of a mixed strategy equilibrium. While the model is a good starting point for understanding penalty kicks through the lens of game theory, it's too simplistic to adequately capture the complexity of a real-life penalty kick. To achieve a more nuanced understanding of Nash equilibria in penalty kicks, more extensive modelling is required. So far, no papers have been published that attempt this, meaning that there exists a gap in the research, which this paper will attempt to fill.

In the paper by Chiappori et al., there's also a focus on testing whether or not player choices have corresponded well with their approximation of the mixed strategy equilibrium. This is because the authors view the results of their model as a prediction about human decision-making. I do not share this perspective. In my view, the solution of the model should be regarded as guidance for players on how to attain the desired outcome, rather than as a prediction about the strategies that players have employed historically. There are several reasons that players may deviate from the equilibrium strategies. For starters, players are almost certainly ignorant about what the exact equilibrium strategies look like. So, beyond broad strokes, it's unreasonable to expect that historical choices and frequencies will line up with equilibrium strategies.

Game theory

In a game-theoretical analysis, the sole focus is on strategic decision-making. In football, other factors matter as well, such as technical ability with the ball, or even the ability to control nerves and maintain composure in high-pressure situations. In my framework, those other factors are treated as assumptions or parameters in the model. The focus then becomes: Which strategies are optimal, *given* the assumed level of technical ability? To begin to answer that question, a solid understanding of game theory is required.

Definition

In common parlance, the word "game" is used to refer to an activity involving decisions, strategy, competition, and so on. This is in alignment with how the term is used in game theory, albeit not as specific. However, we generally also think of a game as something that's done for entertainment purposes, and that's not relevant to whether or not an activity is considered a game in the technical sense. In the textbook "Game Theory: An Introduction" (Rodrigues-Neto, 2014), a game is defined as follows:

"A game consists of players, actions available to each player, and the payoffs that each player receives for each combination of actions. More formally, a game consists of a set of players, a set of actions for each player, and a payoff function for each player that maps each possible combination of actions to a real number representing the payoff that the player receives."

So, for an activity to be considered a game, it needs to three elements: players, actions, and payoffs. Implicitly, a game also needs rules, as the rules are the overarching principle that define the three elements. The rules define the set of possible actions available to each player at all possible decision points, and the payoffs associated with all possible outcomes. They also define the flow of the game - who acts when, and what information is available to them at that decision point. Decisions can be sequential, where players act in turns, with full knowledge of the previous actions, or they can be simultaneous, where players make their choice without knowing the decision of the other player(s).

The textbook also provides a definition of the field of game theory:

"Game theory is the study of mathematical models of strategic interaction between rational decision-makers."

So, strategic interaction between players is the central theme of game theory. Consequently, any game worth studying must allow for it. To have interaction between players, you need multiple players, and to have strategic choices, you need multiple possible actions.

The definition also includes the point that the decision-makers are rational. This effectively means that the decision-makers have a preference of maximizing their payoffs.

Some people critique the rationality-assumption, arguing that in experiments, players sometimes willingly employ strategies that they know won't maximize their payoff in the game. Such observations are then used to argue that the player isn't rational. However, in any experiment, you cannot know which game the player is *truly* playing. In my view, if a player willingly adopts strategies which don't maximize the intended payoffs, that player is simply playing a different game, with different payoffs, and is maximizing those instead.

In penalty kicks, it's generally assumed that the penalty taker and the goalkeeper have conflicting objectives. We assume the penalty taker is solely interested in scoring a goal, and that the goalkeeper is solely interested in preventing it. These are the objectives you would adopt if your sole concern was to win the football match. If this is the case, a rational penalty taker will employ the strategy they believe maximizes the probability of scoring a goal, and a rational goalkeeper will employ the strategy they believe minimizes it. In the absence of match fixing, it looks like we should be confident that players have preferences that align with such objectives. However, it isn't necessarily true.

Players may exhibit behavioural biases, causing them to favour certain actions above others, irrespective of the alternative action yielding a more favourable probability of a goal. For instance, a goalkeeper may be resistant to staying in the middle of the goal, fearing that it will leave the impression that there's a lack of trying, causing backlash from fans and media. In such a case, the goalkeeper has interests that go beyond the sole concern of winning the football match. There's also a concern for reputation. Consequently, the goalkeeper is no longer playing the game of penalty kicks with payoffs that perfectly align with the payoffs that are assumed in the game-theoretical models. It isn't that the player is irrational – it's just that the player's payoffs are different than what is assumed. This is a problem that can lead to a discrepancy between game-theoretical solutions to models, and what we observe empirically.

While such effects will exist, I believe we can be reasonably confident that most players have preferences that at least closely mimic those of a player who seeks to maximize the probability of winning the football match. Just by observing the reactions of players, we can clearly see that penalty takers generally are delighted by scoring goals, and goalkeepers are delighted if they manage to prevent it.

Additionally, if players become more aware of what optimal play looks like, they may be more apt to avoid such behavioural biases. In that sense, solutions to game-theoretical models can help players avoid unconscious biases and help them align their preferences with the intended objective of maximizing or minimizing the probability of a goal.

A simple toy game

To introduce some key game theory concepts, and its applicability to penalty kicks, I believe it will be useful to create and examine a very simple version of the game. There are two players, the penalty taker (PT) and the goalkeeper (GK). The penalty taker has six strategic options: To shoot far left, to shoot left, to shoot high in the middle, to shoot low in the middle, to shoot right, or to shoot far right. The goalkeeper has three strategic options: To dive left, to stay in the middle, or to dive right. The penalty taker and the goalkeeper make their decision simultaneously, meaning that they do not know what the other will do, at the time of making their choice.

If the penalty taker scores a goal, they receive a payoff of 1, and if not, they receive a payoff of 0. If the goalkeeper concedes a goal, they receive a payoff of -1, and if not, they receive a payoff of 0. If there's a non-zero and non-certain probability of a goal, the players receive payoffs equal to the probability. For instance, if a certain combination of choices leaves a 50% chance of a goal, the penalty taker receives a payoff of 0.5, and the goalkeeper receives a payoff of -0.5. This type of game falls within the category of zero sum two-player games, because whatever payoff the penalty taker gains, the goalkeeper loses, so the sum of the payoffs is always equal to zero. For simple simultaneous choice games like this one, it's common to represent the game in the form of a payoff matrix:

		Goalkeeper		
		Dive left	Stay middle	Dive right
Penalty taker	Far left	0.2, -0.2	0.8, -0.8	0.8, -0.8
	Left	0, 0	1, -1	1, -1
	High middle	1, -1	0, 0	1, -1
	Low middle	0.9, -0.9	0, 0	0.9, -0.9
	Right	1, -1	1, -1	0, 0
	Far right	0.8, -0.8	0.8, -0.8	0.2, -0.2

Table 1: The payoff matrix for a simple model of penalty kicks.

Since the penalty taker has six strategic options, and the goalkeeper has three, there are a total of 18 possible outcomes. The payoff matrix summarizes the payoffs for both players in all outcomes of the game. The blue numbers represent the payoffs for the penalty taker, whereas the red numbers are the payoffs for the goalkeeper. If the penalty taker chooses to either "shoot left", "shoot high middle", or "shoot right", we see that the penalty results in a goal 100% of the time, so long as the goalkeeper doesn't choose the best corresponding option (i.e., "dive left" if "shoot left", etc.). In those cases, the payoffs are 1, and -1. If the goalkeeper selects the best corresponding option, there's a 0% chance of a goal, so the payoffs are 0 and 0.

When the penalty taker shoots to the far right or the far left, there's a 20% chance that they will miss the goal entirely. This is reflected in the payoffs. Even if the goalkeeper stays in the middle, or dives in the wrong direction, the penalty taker only receives a payoff of 0.8 (and the goalkeeper receives -0.8), as opposed to 1 (and -1). The upside of shooting far right or far left, is that even if the goalkeeper dives in the correct direction, there's a 20% chance to score regardless. In these instances, the payoffs are 0.2 for the penalty taker, and -0.2 for the goalkeeper.

The final option for the penalty taker is to aim low in the middle. With this option, there's a 10% chance that the goalkeeper manages to stop the shot, despite choosing to dive to the left or the right. Those scenarios give a payoff of 0.9 for the penalty taker, and -0.9 for the goalkeeper. If the penalty taker shoots low in the middle, and the goalkeeper chooses to stay in the middle, the shot is always stopped, meaning that the payoffs are 0 and 0. Notice that the penalty taker also has the options to "shoot high middle". This option performs equally well when the goalkeeper stays in the middle, but better if the goalkeeper dives to either side.

In game theory, if an option always performs worse than another option, regardless of what the opponent does, we call the option strictly dominated. If it sometimes performs worse than the other option, but sometimes equally well, it's weakly dominated. The "shoot low middle" option is therefore weakly dominated by the "shoot high option". An example of a strictly dominated option for the penalty taker could be "shoot outside of the goal". This would result in a payoff of 0, regardless of what the goalkeeper does. Such an option would be strictly dominated by the "shoot far right"- and "shoot far left"-options, as these options always receive a positive payoff. The "shoot left"-, "shoot right"-, "shoot high middle"-, and "shoot low middle"-options would actually only weakly dominate "shoot outside of the goal"-option, as these options sometimes also give a payoff of 0.

Weakly dominated options underperform relative to the alternative if, and only if, the opponent employs a strategy where the non-equal outcomes sometimes occur. So, for example, the "shoot low middle"-option only underperforms relative to the "shoot high middle"-option if the goalkeeper employs a strategy where “dive left” or “dive right” is chosen with a non-zero frequency.

In game theory, a player’s strategy is a set of probabilities that correspond to their available options in the game. Here, the penalty taker has six available options. The strategy for the penalty taker is then a set of six probabilities. The strategy of the goalkeeper, which has three strategic options, is a set of three probabilities. The player must select among the available options, and this means that the probabilities must sum to one. An example of a strategy is the uniform strategy, which picks each option with equal probability. If both players employ this strategy, the penalty taker would select each option with a probability of ⅙, and the goalkeeper would select each option with a probability of ⅓. As an example, the uniform strategy for the penalty taker can be described as [far left: ⅙, left: ⅙, high middle: ⅙, low middle: ⅙, right: ⅙, far right: ⅙].

The uniform strategy is an example of a what’s called a mixed strategy. A mixed strategy is a strategy where at least two options are selected with a non-zero probability. Conversely, a pure strategy is a strategy that selects one option with a 100% frequency, and never selects any of the other options. As an example, if the goalkeeper employs a pure strategy of “stay middle”, the strategy can be described as: [dive left: 0, stay middle: 1, dive right: 0].

If we assume the penalty taker employs the uniform strategy, and the goalkeeper employs the pure strategy of “stay middle”, we can calculate the expected value (EV) of their respective strategies using the formula:

$$EV = \sum_{i,j} (p_i * p_j * EV_{ij})$$

Here, i represents the options available to the penalty taker, and j represents the options available to the goalkeeper.

$$EV_{PT} = \frac{1}{6}(0.8 + 1 + 0 + 0 + 1 + 0.8) = 0.6$$

Given the pure goalkeeper strategy, only the outcomes in the “stay middle”-column of Table 1 can occur – and given the uniform penalty taker strategy, each of those outcomes occur with an equal probability of ⅙. Therefore, the payoffs from the middle column are distributed with equal weighting. The calculation for the expected value of the strategy of the goalkeeper would be similar, but since this is a two-player zero sum game, there is no need to make the calculation. The expected values of the strategies must sum to zero, meaning that:

$$EV_{GK} = -EV_{PT} = -0.6$$

For a moment, let's stay with the assumption that the goalkeeper plays a pure "stay middle"-strategy, and that this strategy is fixed. Given this, what should the penalty taker do in order to maximize their payoff? Both of the choices that involve shooting towards the middle yield a payoff of zero. The options of shooting to the "far left" or the "far right" yield a payoff of 0.8, whereas the options of shooting to the "left" or to the "right" yield a payoff of 1. Therefore, it's clear that to maximize their payoffs, the penalty taker should stick with either shooting "right", or "left". Such a strategy is called the best response strategy. It's the best possible response against a specific strategy – in this case, the goalkeeper's pure "stay middle"-strategy.

There may exist only one best response strategy, or there may exist infinitely many. For there to only exist one best response strategy, one option needs to outperform all others, such that the best response strategy becomes a pure strategy of always selecting that option. In game theory, when two options yield the same payoff, we say that the player is indifferent between the options. In this case, "left" and "right" yield to same payoff, so the penalty taker is indifferent between the options, and there are infinitely many best response strategies. This is because the penalty taker maximizes their payoff regardless of the probability weightings of "left" and "right", as long as those weightings sum to 1, meaning that there are infinitely many combinations.

Let's now assume the penalty taker employs one of the possible best response strategies, namely, to shoot "left" with a probability of 0.5, and to shoot "right" with a probability of 0.5. If we let the goalkeeper update their strategy, what should the goalkeeper do in order to maximize their payoff? Now, "stay middle" is guaranteed to yield a payoff of -1, whereas "dive left" and "dive right" both have a 50% probability of yielding a payoff of 0, and an 50% probability of yielding a payoff of -1. This means that the expected payoff of "dive left" and "dive right" is now equal to -0.5. This is preferable to the guaranteed payoff of -1 when selecting "stay middle". Therefore, the goalkeeper's best response strategy involves only selecting the "dive left"- or "dive right"-options, and again, this means that there's an infinite amount of best response strategies.

We could continue down this path and assume that the goalkeeper now adopts a strategy that selects "dive left" with a probability of 0.5, and "dive right" with a probability of 0.5, and that the penalty taker is free to update their strategy again. The penalty taker would then end up with a single best response strategy of always shooting “high middle”. Given this, if we let the goalkeeper readjust, they would favour to always select “stay middle”. In the next iteration, the penalty taker would be back to being indifferent between “shoot left” and “shoot right”. This back-and-forth would continue indefinitely.

As we have seen, whenever we have perfect knowledge about the strategy the opponent employs, we can find the optimal strategy against that strategy. It’s even rather trivial to do so. The only problem with this is that, in reality, it’s highly unlikely that you find yourself in a situation where you have perfect knowledge about the strategy of the opponent. So, what we really need to answer is: What should we do, assuming that we *don’t* know which strategy the opponent employs?

Nash equilibrium

In 1951, a paper that would revolutionize the field of game theory was published. It was called “Non-Cooperative Games”, and the author was an American mathematician named John Nash. In his paper, Nash introduced and formalized a concept that would later be named after him: the Nash equilibrium. In 1994, he was jointly awarded the Nobel Memorial Prize in Economics for this particular contribution.

The word “equilibrium” has its roots in two Latin words, “aequus” and “libra”. The former means “equal” or “even”, and the latter can be translated to “balance” or “scales”. In general, an equilibrium is a state of stability in a system. This has relevance to many fields beyond game theory. For example, in physics, it can refer to a state in which all forces acting on an object are in balance, resulting in constant velocity, or in ecology, it can refer to a state in which the birth rate equals the death rate, resulting in a stable population.

A Nash equilibrium is a set of strategies that are such that no player can increase their payoff by unilaterally deviating from their current strategy. In other words, we have a Nash equilibrium when all players are playing the best response strategy (to each other’s strategies) simultaneously. So, in game theory, just like in other fields, an equilibrium refers to a state of stability in a system. It refers to a state of a game where all players are maximizing their payoffs, resulting in stable strategies.

In his original paper, Nash proved that every non-cooperative game with a finite number of players, and a finite number of possible actions, has at least one Nash equilibrium. In the game of penalty kicks, there’s only two players. However, if space is continuous, one can argue that we have an infinite number of possible actions. Regardless, when modelling penalty kicks, we can treat space as being discrete, such that the number of options will be finite and countable. Therefore, there will be at least one Nash equilibrium in all models.

In addition to players simultaneously maximizing payoffs, the properties of a Nash equilibrium can also be understood from a dual perspective. Exploitability is a metric that assesses the potential vulnerabilities or weaknesses in a player's strategy, quantifying the degree to which an opponent can deviate from their current strategy and gain an advantage. More specifically, exploitability measures how much expected value an opponent could gain by adopting the best response strategy relative to their current strategy. In a Nash equilibrium, all players are playing best response strategies simultaneously, so all of their strategies have an exploitability of zero. Consequently, a Nash equilibrium is a set of strategies that arise when all players are minimizing the exploitability of their strategy. So, there are two sides to the coin: When all players maximize payoffs, all players minimize exploitability, and vice versa.

From this perspective it becomes natural to view the Nash equilibrium strategies as *maximally defensive* strategies. It's a set of strategies that are impossible to gain an advantage against. In penalty kicks, if two teams with players of equal technical ability alternated between playing as the penalty taker and as the goalkeeper, and one team employed true Nash equilibrium strategies, it would be impossible for the opposing team to achieve an expected value above zero. However, if the opposing team sometimes selects suboptimal options, their expected value will be negative. To illustrate this, let's examine the solution of the simple toy game from Table 1. The Nash equilibrium strategies turn out to be:

Penalty taker: [far left: 0, left: ⅓, high middle: ⅓, low middle: 0, right: ⅓, far right: 0]

Goalkeeper: [dive left: ⅓, stay middle: ⅓, dive right: ⅓]

Given these strategies, the expected payoff for the penalty taker will be ⅔, and -⅔ for the goalkeeper. This is found by taking the probability of each scenario multiplied with the payoff associated with that scenario and adding them all up.

In the solution, both players are employing mixed strategies, meaning that this is a mixed-strategy Nash equilibrium. A player is indifferent between two options if both options yield the same expected value. We also know that the players are maximizing their payoffs. This gives rise to an important corollary: If a player maximizes payoffs, and employs a mixed strategy, the player *must* be indifferent between the options that are chosen at a non-zero frequency. If this wasn't the case, the player could increase their expected payoff by deviating from the current strategy and selecting the more favourable option. So, let's check that all the options a player plays at a non-zero probability actually yield the same expected value:

$$Penalty\ taker: \begin{cases} EV_{left} = \frac{2}{3} \\ EV_{high\ middle} = \frac{2}{3} \\ EV_{right} = \frac{2}{3} \end{cases} \qquad Goalkeeper: \begin{cases} EV_{dive\ left} = -\frac{2}{3} \\ EV_{stay\ middle} = -\frac{2}{3} \\ EV_{dive\ right} = -\frac{2}{3} \end{cases}$$

Once again, the expected values are derived by multiplying the probability of the scenarios with their associated payoffs and add them all up. As we can see, each option for each player yields the same expected payoff, which is also equal to the expected payoff of their overall strategy. This verifies that the players are indifferent between the options. To verify that the set of strategies qualify as a Nash equilibrium, we also need to examine the expected value of the other options available to the players (in this case, only to the penalty taker):

$$Penalty\ taker: \begin{cases} EV_{far\ left} = 0.2 * \frac{1}{3} + 0.8 * \frac{1}{3} + 0.8 * \frac{1}{3} = 0.6 \\ EV_{low\ middle} = 0.9 * \frac{1}{3} + 0 * \frac{1}{3} + 0.9 * \frac{1}{3} = 0.6 \\ EV_{far\ right} = 0.2 * \frac{1}{3} + 0.8 * \frac{1}{3} + 0.8 * \frac{1}{3} = 0.6 \end{cases}$$

As we can see, all of these options are inferior to the played options, as they only yield an expected value of 0.6, as opposed to ⅔. This means that it's impossible for either player to increase their payoff by deviating from their current strategy, verifying that the current strategies qualify as a Nash equilibrium.

The lower expected value associated with "far left", "low middle", or "far right" illustrate how a Nash equilibrium strategy can obtain an advantage against certain opponents. Any opponent who employs a strategy that plays either "far left", "low middle" or "far right" with a non-zero frequency will be at a disadvantage and lose expected value against the Nash equilibrium strategy. In a simple toy game such as this one, it may seem trivial to avoid such options. However, in an actual penalty kick, where the available options are far more complex, it can be very difficult to discern which options will lose expected value. Therefore, adopting Nash equilibrium strategies can be very fruitful, and yield a significant advantage against such opponents.

When up against the Nash equilibrium strategies, in a two-player zero sum game, where both players play both roles equally often, the best you can do is to tie. A common misconception is that to do so, you have to adopt Nash equilibrium strategies yourself. However, it's enough to successfully avoid playing any of the suboptimal options. The frequencies at which the optimal options are selected don't have to align with those of the equilibrium. For instance, in the toy game, you could employ a set of strategies where you always choose "high middle" as the penalty taker, and always "stay middle" as the goalkeeper, and still tie against the Nash equilibrium strategies. This is because, when holding the opponent's strategy constant, the "high middle" option has the same expected value as the strategy of the penalty taker, and the "stay middle" option has the same expected value as the strategy of the goalkeeper. While such a set of strategies would tie against the Nash equilibrium strategies, they would be highly exploitable.

So, there are two classes of exploitable strategies. There are the ones that lose expected value against the Nash equilibrium strategies, simply as a result of selecting suboptimal options with a non-zero frequency. Then there's the other class, which never selects suboptimal options, but are exploitable because the frequencies are suboptimal. Gaining an advantage against this second class of exploitable strategies is only possible by deviating from the Nash equilibrium as well. It requires the adoption of a strategy that actively attempts to take advantage of the weaknesses of the strategy of the opponent. Such strategies are called exploitative strategies.

Unlike Nash equilibrium strategies, which are maximally defensive, exploitative strategies are offensive in nature. As the exploitative strategy is created in an attempt to exploit the vulnerabilities in the opponent's strategy, it deviates from the Nash equilibrium, thereby introducing vulnerabilities of its own. This is risky, because if the opponent's strategy is misjudged, and it actually turns out that the opponent is one step ahead, as opposed to the reverse, you end up losing expected value rather than gaining it. For instance, if the penalty taker believes the goalkeeper never chooses "stay middle", the penalty taker may adopt an exploitative strategy of always choosing "high middle". If the penalty taker is correct, the penalty kick will always result in a goal, meaning that their expected payoff will increase to 1. This is a solid improvement relative to the expected payoff of ⅔, which the penalty taker would be entitled to if both players adopted Nash equilibrium strategies. However, the penalty taker may be mistaken in their assumptions about the goalkeeper's strategy. In the worst case, where the penalty taker is just entirely wrong, the goalkeeper could employ a strategy of selecting "stay middle" with a 100% frequency, which then results in a zero probability of a goal, and expected payoffs of zero to both players.

Once you enter the realm of exploitative strategies, it's really all a question of who can be one step ahead of the other. That's not to say that there's no place for deviating from the Nash equilibrium. Sometimes, you truly have solid information the specific tendencies of the opponent, and it would be foolish not to try to take advantage of it. However, the question I originally raised regarded what to do when there's a lack of information. What do we do if we're completely agnostic about the composition of the opponent's strategy? The answer is that the Nash equilibrium strategies function as the best baseline approach.

Additionally, an improved knowledge about what the Nash equilibrium strategies look like may also help players recognize vulnerabilities in the opponent's gameplan. After all, it's hard to know in which manner the opponent is deviating from the Nash equilibrium, if you don't have an idea about what the equilibrium looks like. Therefore, knowledge about Nash equilibrium strategies is not only useful for players that want to develop an unexploitable gameplan, but also for players that want to actively attack the weaknesses of the opponent.

So, now that it's clear that we want to find the Nash equilibrium strategies, the question becomes: How can we find them? The toy game is simple enough that you can solve it simply by thinking about it, and verifying the solution by hand. However, once there are many more combinations of options, this isn't feasible. In the example, we continuously updated one player's strategy to be the best response strategy to the other player's fixed strategy, alternating between which player's strategy was held fixed, and which player's strategy was being updated. This resulted in an indefinite back-and-forth, and there was no convergence towards an equilibrium. Thankfully, there exists an algorithm we can use.

Counterfactual regret minimization

In a paper called "Regret Minimization in Games with Incomplete Information", Zinkevick et al. (2007) introduced the Counterfactual Regret Minimization (CFR) algorithm. The algorithm is an effective method for approximating Nash equilibrium strategies in games with incomplete information, particularly for two-player zero sum games. The paper demonstrates the algorithm's ability to learn near-optimal strategies for various games. Notably, the algorithm has been used to approximate optimal play in games as complex as No Limit Texas Hold'em (Moravčík et al., 2017), creating AI agents operating at a superhuman level of precision. The success of the algorithm in games like poker has also inspired researchers to explore its potential in various other applications. An example of such an application is negotiation or bargaining scenarios (Baarslag, Kaisers, Gerding, & Jonker, 2017). I'm going to use the CFR-algorithm to approximate Nash equilibrium in penalty kicks. This chapter will serve as an introduction to how the algorithm works.

Counterfactual regret is a key concept to understand in the CFR-algorithm. At any decision point, a player may employ a mixed strategy, meaning more than one option is selected with a non-zero frequency. Nevertheless, every time the player is faced with a decision, they need to pick a specific action. For instance, in a penalty kick, the penalty taker may employ a mixed strategy where they sometimes aim to shoot to the left side and sometimes to the right side, but that doesn't mean that they can aim to shoot to both sides at once. Every time they take a penalty kick, they must choose a specific side. The choice that was made can be viewed as the factual choice, whereas the choices that could have been made, but weren't, are the counterfactual choices. So, when we are talking about counterfactual regret, we are talking about regret associated with possible actions that weren't taken.

All choices have an expected value. The expected value of a choice is the payoff the player will receive on average by making that choice. Usually, this will be subject to which strategy the opponent employs. If the opponent has a strategy that performs well against a choice, that choice will have a lower expected value than if the strategy of the opponent performs poorly against it. For instance, if the penalty taker chooses to aim to shoot to the left, and the goalkeeper's strategy rarely involves diving in that direction, the penalty taker's choice will have a much higher expected value than if the goalkeeper employs a strategy that dives in that direction frequently. We can only know the expected value of the penalty taker's choice if we have perfect information about the strategy of the goalkeeper. To calculate regret, we need to be in possession of that knowledge.

Counterfactual regret is a measure the difference in expected value between choices. It measures the expected value of a counterfactual choice, relative to the factual choice, given a specific opponent strategy. It quantifies how much better or worse off a player would have been if they had chosen a different action. If a counterfactual choice would have resulted in a higher expected value than the factual choice, the regret is positive, and if the opposite is true, the regret is negative. The regret associated with a counterfactual option is simply a comparison of expected value (EV) between that option and the factual option. Mathematically, the counterfactual regret can be expressed as:

$$Regret_{counterfactual\ option} = EV_{counterfactual\ option} - EV_{factual\ option}$$

There's always only one factual option, but there may be many counterfactual options. If a decision point has n possible options, there will be $n - 1$ counterfactual options, and just as many counterfactual regret values.

Now that we understand the concept of counterfactual regret, it's time to focus on how the algorithm works. The algorithm is iterative, meaning that it simulates many rounds of play. To simulate a round of play, actions for both players are chosen randomly, according to the probability weightings of their strategies. In the first iteration, both players are playing strategies where all possible choices are weighted with equal probability. From one iteration to the next, the strategies of the players are updated. The principle which decides the composition of the strategies for the next iteration is called regret matching.

At each iteration, the counterfactual regret values for all possible choices are computed. This gets added to the sum of counterfactual regret values for each option, throughout all previous iterations. Each option has its respective sum of counterfactual regret. These sums form the basis of deciding the strategies in the next iteration. Some sums will be negative, and others will be positive. If the counterfactual regret sum of an option is negative, that option will have zero probability in the next strategy. The strategy for the next iteration will only match the positive regret sums. The probabilities of all options in a strategy needs to sum to one, so to determine the next strategy, the positive regret sums are normalized. As an example, if there are five options, A, B, C, D, and E, and these have the respective counterfactual regret sums of -10, 20, 10, 20, and -10, option A and E will be played with zero probability in the next iteration, since their sums are negative. Option B and D both have sums equal to 20, and should therefore be played equally often, whereas option C, which has a sum of 10, should be played half as often as those. In this case, the next strategy would then be to play option B with a probability of 0.4, option C with a probability of 0.2, and option D with a probability of 0.4.

The counterfactual regret sums have a specific and useful interpretation. The sum for a specific action is a measure of how the player would have performed, had they chosen that action in all iterations, as opposed to the long sequence of mixed strategies that was otherwise chosen throughout the iterations. If the regret sum for an action is positive, it means that the player would have performed better by always choosing that action, and if it's negative, it means that they would have performed worse. So, when the strategies are updated in accordance with the principle of regret matching, we are effectively only selecting actions that would have overperformed in the past, and avoiding the actions that would have underperformed.

One might think that by updating the strategies in accordance with regret matching, the strategies end up converging towards a Nash equilibrium. That's not necessarily the case. The strategies that are updated across iterations are not guaranteed to converge to a Nash equilibrium, because they depend on the opponent's strategy, which is also changing over time. You can easily end up with an endless cycle of updates and counter-updates, which switches between overshooting and undershooting the true Nash equilibrium frequencies. In such cases, the updated strategies can be very exploitable, despite having been arrived at through many iterations.

The CFR-algorithm converges towards a Nash equilibrium solution in another manner. In two-player zero-sum games, the *average* of the strategies used throughout all iterations will converge towards a Nash equilibrium. Even if the current strategy is an endless cycle of overshooting and undershooting equilibrium frequencies, the average strategy will be in-between and over time it approaches Nash equilibrium. To calculate the average strategy, the algorithm keeps track of the sum of all strategies that have been employed throughout all iterations. By normalizing the sum of the strategies, such that the probabilities add up to one, we get the average strategy, and thereby also a Nash equilibrium solution.

In the time since the original paper about the CFR-algorithm was published, others have made various adjustments in order to make the algorithm converge faster. Brown and Sandholm (2019) ran tests on various discounted regret minimization techniques, which are variations of the CFR-algorithm which implement a discount rate. There are several ways to implement discounting, but one simple way of doing it is to discount the earlier iteration's contributions towards the sum of strategies that have been used. This effectively makes later iterations count disproportionately more towards the average strategy. Since the very early iterations can be quite far off from equilibrium strategies, this can make the algorithm converge quicker. However, when the discount rate is applied in this manner it also means that, given enough iterations, the contribution from the early iterations will approach zero. This effectively puts a cap on the number of iterations that can contribute towards the approximate solution. The result is that the algorithm converges faster, but that the approximation cannot be as precise as if the algorithm used a discount rate of zero.

Framework development

In the previous chapter, I introduced some basic game theory, and outlined why Nash equilibrium strategies are useful. I also introduced a method of approximating equilibria – the counterfactual regret minimization algorithm. Now it's time to start developing the framework, creating models that effectively mimics the most important aspects of a real-life penalty kick, allowing us to gain insight about Nash equilibria in the real game. To be able to solve for a Nash equilibrium, the models must be defined in accordance with the technical definition of a game. This means that we define the players of the game, all the possible actions in the game, and the payoffs each player receives for any combination of actions.

In penalty kicks, there's only two players: the penalty taker and the goalkeeper. Furthermore, throughout the framework, I assume that the penalty taker and the goalkeeper both have preferences that are best satisfied by maximizing the likelihood of their team winning the football match. This implies that the penalty taker has the objective of maximizing the probability of scoring a goal, and that the goalkeeper has the objective of minimizing it. Just like in the simple toy game from the previous chapter, the payoffs of the players are connected to a simple metric: goals scored, and goals conceded. When the penalty taker scores, their payoff is 1, and the goalkeeper's payoff is -1. If the penalty kick doesn't result in a goal, both players receive a payoff of zero. However, usually, the outcome from a combination of actions won't be guaranteed to either result in a goal or not. Therefore, for each combination of actions, the penalty taker receives a payoff equal to the probability that the outcome is a goal, and the goalkeeper receives a payoff equal to the negative probability.

The set of possible actions available to each player also needs to be defined. These are defined by the rules of the game, which are underlying assumptions of the models. When studying penalty kicks, it's important to be aware that some assumptions are player-dependent, meaning that they depend on the inherent abilities of the players. For example, some penalty takers will be able to shoot more accurately than others, and some goalkeepers will be able to cover a larger area of the goal than others. On the other hand, some assumptions are player-independent, meaning that they do not depend on the specific attributes of the penalty taker or the goalkeeper. Let's begin by outlining the player-independent assumptions, before moving on to the player-dependent ones.

Simultaneous choice

In reality, the decisions made throughout a penalty kick takes place over several seconds. The penalty taker and the goalkeeper effectively start making strategic decisions from the moment the run-up to the ball begins. For the penalty taker, it can be argued that the decisions end the moment where the ball has been kicked, and it starts travelling towards the goal. For the goalkeeper, decisions are being made up until the point where either the ball has crossed the goal line, or it's clear that the ball cannot cross it any longer. The point is, in reality, both players make decisions throughout a continuous time dimension.

Even every slight tweak in body language counts as a decision, and in the true Nash equilibrium all such possible movements would have to be accounted for. Accounting for such factors would make the system immensely complex, and it would be far beyond our current capabilities. However, I believe we can deduce that some of these options cannot be part of the true Nash equilibrium. Let's examine an example regarding body language: Imagine that the penalty taker runs towards the ball in such a manner that it's revealed that the shot will go towards a specific side. Can such a run-up be part of the Nash equilibrium? Clearly not, because in such a case, the goalkeeper would always be able to move early in the right direction, resulting in a low probability of a goal. The same is true if the goalkeeper dances around in the goal in such a way that it reveals where they will dive. Then the penalty taker would just shoot to the opposite side, resulting in a very high probability of a goal.

From deduction, it's clear that players should conceal information about their action from their opponent. When the penalty taker kicks the ball, they should not have information about which area of the goal the goalkeeper will defend. On the other hand, the penalty taker is forced to reveal where they are shooting eventually – when the ball is kicked, and it starts travelling towards the goal. This means that the goalkeeper will be able to react to the trajectory of the ball.

Another important element in penalty kicks is that both players have a non-zero reaction time. Even if the goalkeeper starts moving shortly before the ball is kicked, the penalty taker won't necessarily be able to react to it. Similarly, the goalkeeper will not be able to react to the trajectory of the ball instantaneously. This effectively means that players have an additional window of time of which their action is concealed, regardless of whether or not the opponent has made their move.

Since players make their decisions without knowing what the opponent will do, penalty kicks can be modelled as a simultaneous choice game, with a single decision point. Notice that the goalkeeper can wait to observe the trajectory of the ball before making their move, but that does not make the decision sequential. This is because the decision to wait is made simultaneously, whereas the movements beyond that point are not. The same is true even if the goalkeeper decides to dive to a side. In such a case, the goalkeeper will already be moving once they realize what the trajectory of the ball is. That doesn't mean they can't make further movements to adjust to the trajectory.

Domains

Football goals have a rectangular frame that consist of two vertical goalposts and a horizontal crossbar that connects them. In 11-a-side senior football, the goal is 24 feet wide and 8 feet tall. In meters, this translates to a width of approximately 7.33 meters, and a height of approximately 2.44 meters. The measurements of a football goal were originally defined using the imperial measurement system, hence the round numbers when measured in feet. Using feet as the unit of measurement will be more convenient and may provide a better understanding and appreciation of the dimensions involved.

The height is measured from the ground and up to the bottom edge of the crossbar, while the width is measured as the distance between the inside edges of the goalposts. To describe the goal, a two-dimensional coordinate system will be used, as depicted in Figure 1. The horizontal axis represents the width of the goal, and the vertical axis represents the height. Depth is not a concern in this framework. The figure presents a cross-sectional view of space along the goal line, from the perspective of the penalty taker. The thick black lines correspond to the goalposts and the crossbar.

With the use of (x, y)-coordinates, we can precisely describe spatial points in and around the goal. The origin of the coordinate system is located at the very bottom left corner of the goal, where the inside of the left goalpost meets the ground. The inside of the right goalpost is located along the line where $x = 24$. It intersects with the bottom edge of the crossbar positioned along the line where $y = 8$.

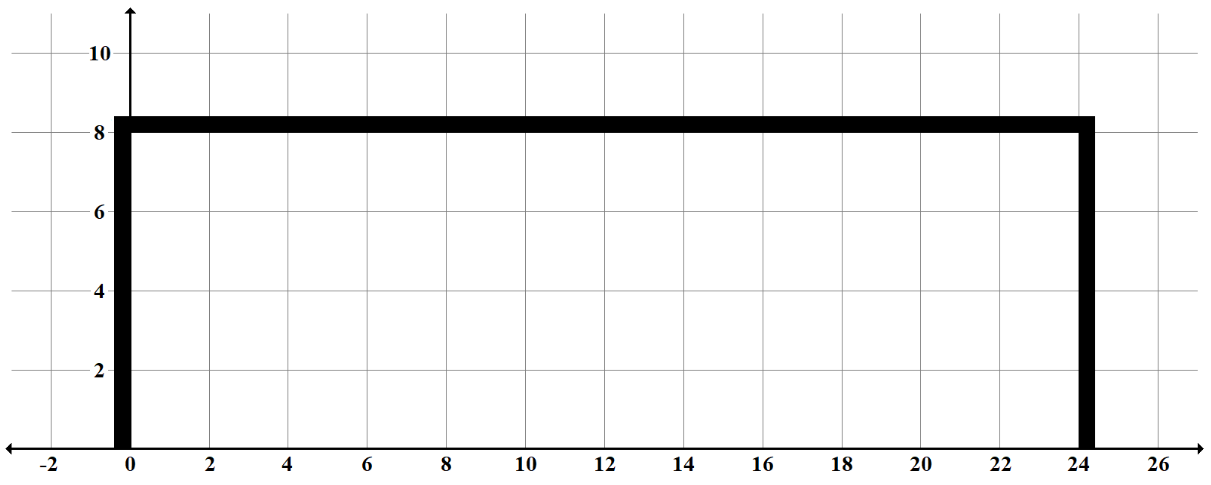


Figure 1: The goal in a two-dimensional coordinate system, from the perspective of the penalty taker.

Coordinates provide a means of specifying the exact point where the ball connects with the goal during a penalty kick. The ball has width and height, so we are actually only describing which point the centre of the ball hits, and not the entire area that the ball occupies. According to FIFA-regulations, the circumference of the ball must be between 27 and 28 inches. If we assume that the ball has a circumference of 27.5 inches, the ball will have a radius of approximately 4.38 inches, or 0.365 feet. In this study, for simplicity, I will assume that the ball remains in its original shape and doesn't deform or change shape due to factors such as impact with the ground. These assumptions imply that all coordinate points where $y < 0.365$ are unfeasible, as it would mean that a portion of the ball is below the ground surface. The domain of possible hit-positions can then be defined as $x \epsilon [-\infty, \infty]$, which means that the x-coordinate can take on any real number value, and $y \epsilon [0.365, \infty]$, which means that the y-coordinate must be greater or equal to 0.365.

According to FIFA-regulations, both the goalposts and the crossbar can have a maximum diameter of 5 inches, or approximately 0.417 feet. In professional leagues and competitions, the goalposts and crossbar are typically at or near the maximum limit, so for the purposes of this study, I'll assume that they are at the maximum allowed diameter. Given the diameter of the goalposts and crossbar, and the radius of the ball, we can identify all coordinate points where there would be overlap between the area of the ball and the area of the frame of the goal. As long as the shot doesn't end up above the goal, there will be contact with the left goalpost if $-0.782 < x < 0.365$, and with the right goalpost if $23.635 < x < 24.782$, and as long as the shot doesn't end up to either side of the goal, there will be contact with the crossbar if $7.635 < y < 8.782$.

Sometimes the ball will come into contact with a goalpost or the crossbar, but still bounce into the goal. I need to define exactly which coordinate points this occurs at. According to FIFA-regulations, both the goalposts and the crossbars should be rounded. The ball is of course also round. For simplicity, I'll disregard spin of the ball as a factor in the framework. We know from physics that if the ball doesn't spin, it should leave the point of contact at the same angle that it entered. Using trigonometry, I can calculate which coordinates result in a goal, absent of goalkeeper intervention. It turns out that a good approximation of the domain of hit-positions that result in a goal are $x \epsilon \langle 0.257, 23.743 \rangle$ and $y \epsilon [0.365, 7.751 \rangle$. The calculation where I derive this domain is presented in Appendix I.

If the ball hits the frame of the goal and rebounds during a penalty shootout, the ball is out of play, meaning that the penalty taker cannot score from a rebound. However, if the ball rebounds from a penalty during normal play, the ball is still in play, so the penalty taker may strike the ball again and manage to score. In this framework, I'll assume that rebounds are out of play. This means that the framework is best suited for penalty shootouts. On the other hand, allowing for scoring on a rebound shouldn't have a massive impact on result, so the framework is still useful for analysing other penalties as well. A simple way of attempting to account for scorable rebounds would be to expand the domain of hit-positions that result in a goal slightly.

Area coverage

Now it's time to move on to the player-dependent assumptions. Throughout all iterations of the framework, the strategic options of the goalkeeper are limited to choosing which area of the goal to defend. Starting out, I will begin with three different area options, and later on two more options will be added. The initial options are "stay middle", "commit left", and "commit right".

It's natural to assume that, in the equilibrium, as the penalty taker is about to kick the ball, the goalkeeper is standing in the middle of the goal. If the goalkeeper would stand to a particular side, they would have a very poor reach to the opposite side, which would likely lead to a higher scoring rate for the penalty taker. Therefore, it should be a safe assumption.

The penalty taker and the goalkeeper make a simultaneous decision of which coordinate to aim for, and which area to defend. The area that corresponds to the "stay middle"-option is meant to represent the area a goalkeeper will be able to defend, if they stay in the middle, waiting to observe the trajectory of the ball. The "commit left"- and "commit right"-options are intended to represent the area a goalkeeper is able to cover if they commit to diving in that direction before observing where the ball is headed. Whether or not an area is defended is decided entirely deterministically, meaning that all coordinates within the area are guaranteed to be defended, and all coordinates outside of it are not.

I looked for data on average save frequencies in different parts of the goal, given a specific goalkeeper action (such as diving to a specific side before observing the trajectory of the ball), but was unable to find something satisfactory. Therefore, the best I could do was to create save-areas that seem reasonable to me. This isn't a big problem, because goalkeeper area coverage is a player-dependent assumption, so there isn't a singular set of area coverage-assumptions that can be viewed as correct. It would have been preferable to know the average save frequencies given a specific action, but it isn't necessary. While the areas I have chosen may not exactly mimic the average save areas across all goalkeepers, they may be a good fit for some specific goalkeeper.

The different area options can be defined within our two-dimensional coordinate system. This is done using inequality expressions. The size and positioning of the areas are dependent on the velocity of the ball. This is accounted for using what I call the velocity-factor, v. Not that the velocity-factor is not intended to mimic a specific velocity measurement, such as meters/second. It's simply there to scale and define the locations of the areas.

A lower velocity means that it takes a longer time for the ball to travel towards the goal. If the goalkeeper has chosen "stay middle", the lower velocity will result in further reach, and an expansion of the area. If the goalkeeper has chosen "commit left" or "commit right", the lower velocity will result in the location of the area to move further away from the from the centre of the goal. Figure 2, Figure 3, and Figure 4 illustrate the area-coverage for the various options, given that the velocity of the ball is high ($v = 10$), medium ($v = 8.5$), and low ($v = 7$), respectively. The blue area represents the area coverage of the "stay middle"-option, whereas the green areas represent the area coverage when the goalkeeper selects "commit left" or "commit right".

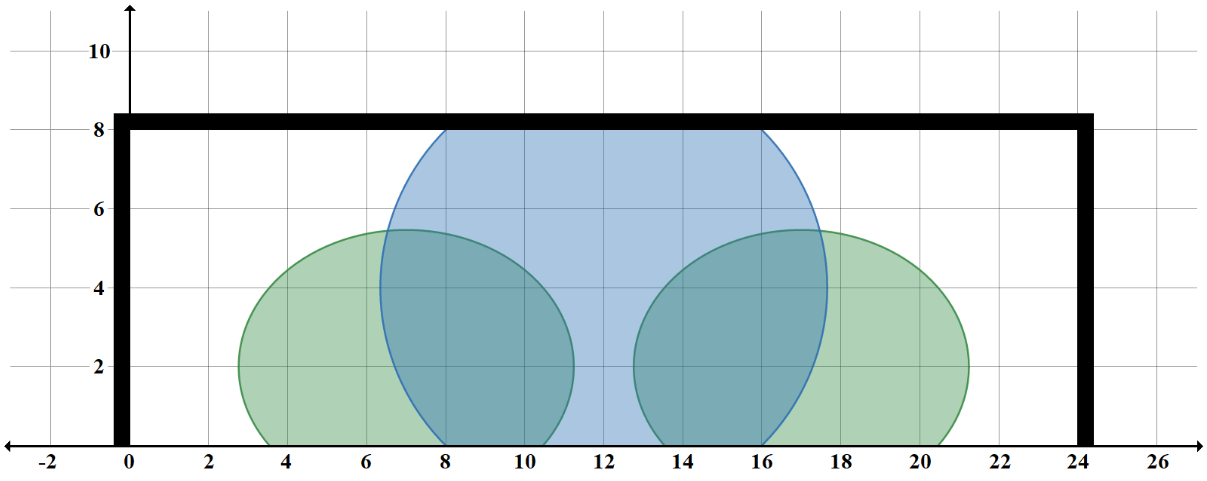


Figure 2: Possible area coverage when the velocity-factor is equal to 10.

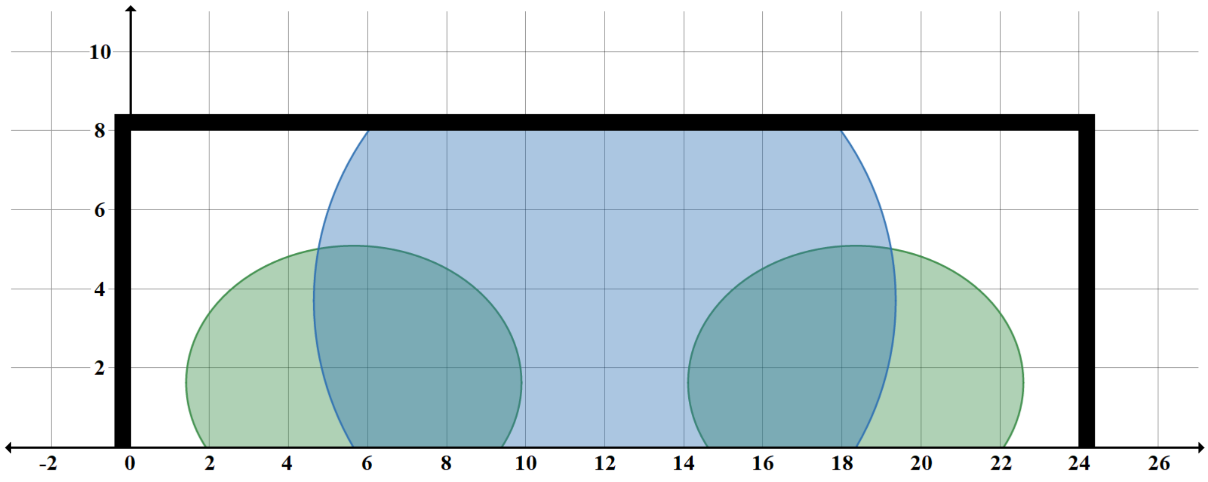


Figure 3: Possible area coverage when the velocity-factor is equal to 8.5.

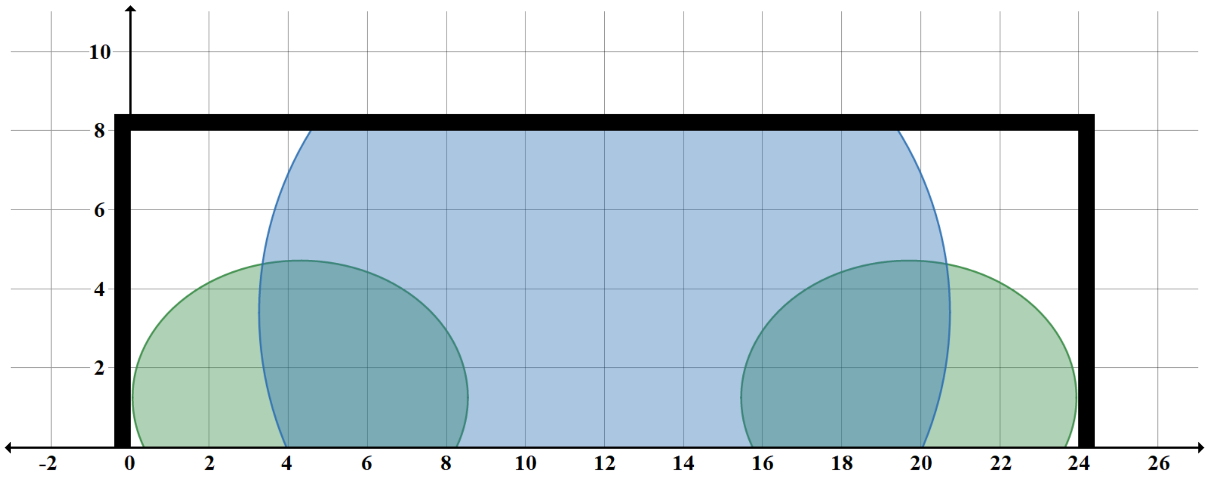


Figure 4: Possible area coverage when the velocity-factor is equal to 7.

The areas are defined mathematically as follows:

$$Stay\ middle: \frac{(x-12)^2}{8} + \frac{(y-2-0.2v)^2}{8} \leq 22.5 - 1.85v$$

$$Commit\ left: \begin{cases} \dfrac{(x+2-0.9v)^2}{3} + \dfrac{(y+0.5-0.25v)^2}{2} \leq 6 & ,7 < v \leq 10 \\ \dfrac{(x-3.4)^2}{3} + \dfrac{(y-3.2)^2}{2} \leq 6 & ,0 \leq v \leq 7 \end{cases}$$

$$Commit\ right: \begin{cases} \dfrac{(x-26+0.9v)^2}{3} + \dfrac{(y+0.5-0.25v)^2}{2} \leq 6 & ,7 < v \leq 10 \\ \dfrac{(x-20.6)^2}{3} + \dfrac{(y-3.2)^2}{2} \leq 6 & ,0 \leq v \leq 7 \end{cases}$$

The velocity-factor, v, has the domain $v\epsilon[0,10]$. This is designed to be such that when $v = 0$, the "stay middle"-option covers the entire goal. When $v = 10$, the penalty taker shoots with the highest velocity they are capable of. The "commit left" and "commit right"-areas are don't change in the when $v \leq 7$. You can imagine that when the velocity-factor is this low, the goalkeeper has already landed on the ground, and their reach doesn't change.

In the first few iterations of the model, the velocity-factor is assumed to be uniform. That is, penalty taker always shoots with the same velocity-factor, regardless of where they are aiming. I will solve the model for when $v = 10$, when $v = 8.5$, and when $v = 7$, as illustrated in the previous figures. In later iterations of the framework, the penalty taker will be allowed to choose between the different velocity-factors as well.

Hit-coordinates

In the initial model, the penalty taker only has one strategic choice: which coordinate point to aim for. Relating to this strategic choice is an assumption that applies to all players: No penalty taker is able to shoot with perfect accuracy. Therefore, the coordinate point that is aimed for isn't necessarily the coordinate point the ball ends up hitting. We can be extremely confident in this assumption. If it wasn't true, we would expect to observe players that would never miss the goal, or players that would always be able to place the ball in areas of the goal that are out of reach of the goalkeeper. This is not what we observe in the real world.

The inaccuracy of the shot can be modelled using a probability distribution. While it's clear that no penalty taker can shoot with perfect accuracy, it's unclear which distribution the inaccuracy follows. Ultimately, this will be player dependent, meaning that some players may shoot in a manner that follows one type of distribution, while others might follow a different distribution. Additionally, as far as I'm aware, there hasn't been any studies that investigate how accurately players are able to shoot, and in which manner they are inaccurate. Empirical data on where penalties have historically hit wouldn't help, because while we can observe where the shot ended up hitting, we do not know where the penalty taker was aiming to hit. So, to effectively study the inaccuracy of penalty takers, you would require the player to specify where they are aiming before they shoot. I would urge future researchers to conduct such a study, as it would help improve the assumptions of the framework.

Throughout the framework, I'm going to assume that the inaccuracy of the players follows a bivariate normal distribution, where the aim coordinate is the mean of the distribution. The distribution is symmetrical, so it means that players are equally likely to be inaccurate in either direction of where they aim. I don't see much of a reason to expect players to for example be inaccurate more often to the left than to the right, so this seems reasonable to me. The degree of inaccuracy can be described by the standard deviation. It could be argued that the standard deviation in the horizontal and vertical directions may not be equal, but as a baseline assumption, I'm going to assume that it is. Figure 5 illustrates an example of a hit-distribution of a penalty kick where the penalty taker aims at coordinate (4, 4) and has a standard deviation of 2 in both the horizontal and vertical direction.

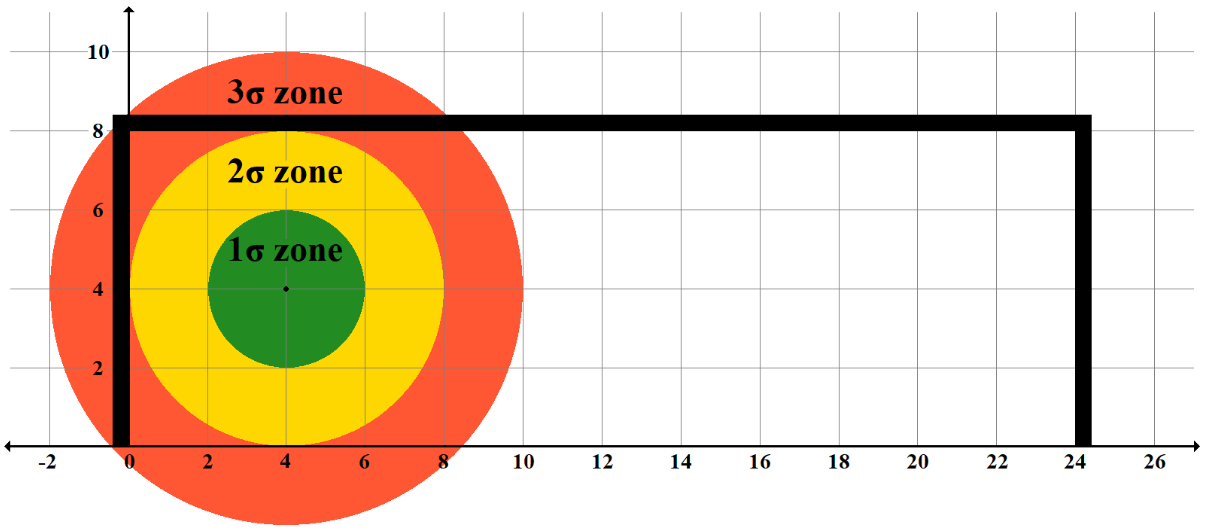


Figure 5: The hit-distribution of a penalty taker with a standard deviation of 2 that aims at coordinate (4, 4).

Here, the green area represents the coordinates that are within one standard deviation of the aim-coordinate. There's roughly a 68.2% chance that the penalty taker hits within this area. The yellow and green areas combined represent the coordinates that are within two standard deviations away from the aim-coordinate. The chance that the shot will hit within these areas is about 95.4%. Finally, the outer edge of the red area is three standard deviations away from the mean. The chance that the ball will hit within the red, yellow, or green areas is about 99.7%.

I established earlier that it's not feasible to hit a y-coordinate that is below $y = 0.365$, as this would imply that part of the ball is below the ground. When drawing hit-positions from the probability distribution, you could end up with y-coordinates below that. In those cases, I move the y-coordinate up to $y = 0.365$ and keep the x-coordinate as its drawn. This often ends up meaning that it's much more likely to end up with a y-coordinate of 0.365 than a slightly higher coordinate of say 0.5. At first glance, this could feel unintuitive, but it's actually completely reasonable. If the ball rolls across the ground, it ends up at coordinates where $y = 0.365$, whereas a coordinate where $y = 0.5$ means that the ball is a very specific height above the ground, which is much more unlikely. One slight caveat is that in reality not all shots which have a drawn y-coordinate below the threshold would end up being in contact with the ground when it crosses the line. The ball could also bounce in the ground and hit a higher y-coordinate in some situations. However, such an effect probably wouldn't impact the results much, so it's not something I'm going to try to account for.

When the ball comes into contact with the ground, there is increased friction, slowing the ball down. I account for this by introducing a friction-factor. If the drawn y-coordinate is below 0.365, the velocity-factor v is reduced in proportion with the friction factor. This means that the hit-velocity can be different from the initial velocity in the system. The hit-velocity is determined by the following equation:

$$hit\ velocity = velocity - (0.365 - y) * friction$$

As an example, if the initial velocity factor is set to 10, the drawn y-coordinate is -1, and the friction factor is 0.1, the hit-velocity will be 9.8635. This effectively functions as a small disincentive for the penalty taker to aim low, as the reduced velocity will have a disadvantageous impact on the goalkeeper area coverage. It's unclear what a realistic friction factor should be, but throughout the framework I'm using a rather small value of 0.1.

Aim-coordinates

As mentioned, the strategic options of the penalty taker are limited to choosing where in the goal to aim. The area inside the goal is a continuous space, meaning that there are infinitely many possible coordinates. However, to solve the model, it's necessary to have a finite number of possible choices. Therefore, I need to pick a set of coordinates that the penalty taker can choose from. In this first solution, my main aim is to illustrate how the model works, so I will keep it fairly simple. The penalty taker gets to choose from 55 different coordinate points, as illustrated in Figure 6.

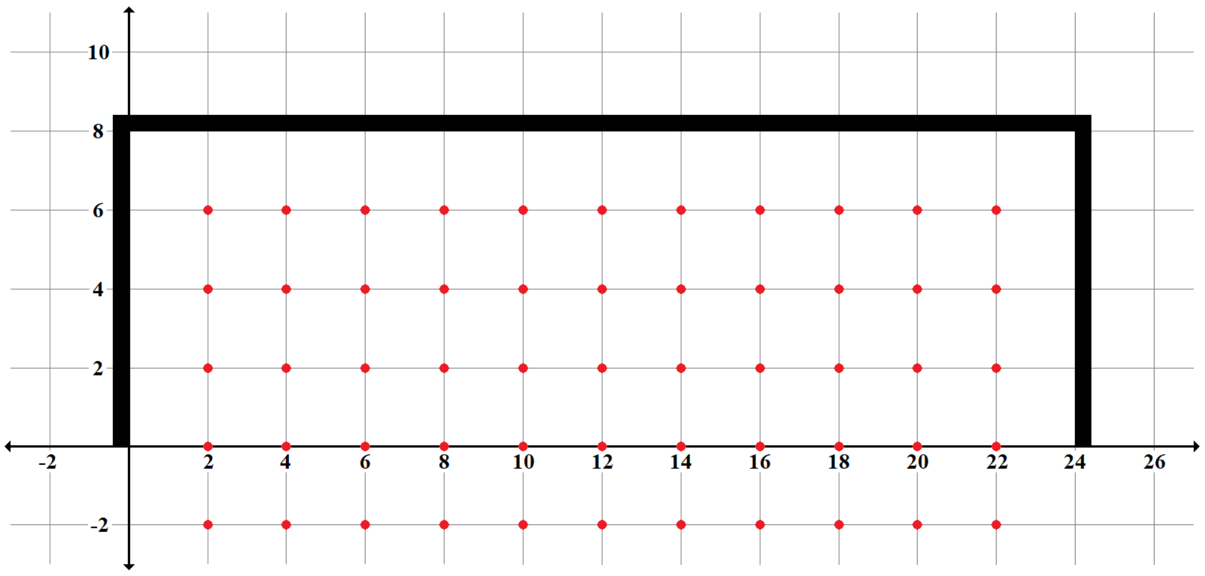


Figure 6: Aim options for the penalty taker in the first iteration of the framework.

Notice that I have allowed for the penalty taker to aim at coordinates where y<0.365. This may feel unintuitive given that the ball cannot hit such coordinates. However, I believe it makes a lot of sense. You could imagine a footballer kicking a ball from the edge of a cliff: Is it possible for the footballer to aim at a point lower than him? Of course, and it would also be possible to hit those points. In a penalty kick, it's also possible to aim at the lower points - the only difference is that the ground prevents the shot from hitting those coordinates. If the penalty taker aims far below the ground, the result will almost certainly be that the ball rolls across the ground as it passes the goal line, which is what happens in the model. Anyone who has experience playing football knows that it's quite easy to achieve a shot that rolls across the ground, so this is realistic.

Payoffs

The penalty taker has 55 different aiming-options, and the goalkeeper has three different area-options. This means that there's a total of 165 possible combinations of aim-options and area-options. Table 1 in the previous chapter, presented a pay-off matrix for a simple toy game with just 18 possible combinations. The pay-off matrix consists of information about the payoffs for each player in all 18 scenarios. Without complete information about the payoffs in each scenario, the game isn't even properly defined, and that's a prerequisite for being able to solve it. Therefore, I need to find the payoffs associated with each of the 165 possible scenarios.

The expected values (payoffs) for both players in each scenario is found by simulation. As an example, let's say we are operating within a system where the velocity-factor is equal to 10, and that the penalty taker's hit-distribution has a standard deviation of 2 in both the vertical and horizontal direction, regardless of which coordinate is being aimed for. Let's then assume the penalty taker aims at coordinate (4, 4), just like the illustration in Figure 5, and that the goalkeeper chooses "commit left", covering the leftmost green area illustrated in Figure 2. This would then be one out of the 165 possible scenarios. To find the probability of a goal in this scenario, I draw hit-coordinates from the hit-distribution and check whether or not the coordinate is (1) within the domain which can result in a goal, and (2) outside of the area that is covered by the goalkeeper. Doing this over a large number of iterations, will provide a good estimate of the probability of a goal, given the specific combination of strategic choices. The payoff for the penalty taker will be equal to the probability of a goal, and the payoff for the goalkeeper will be the negative value of the payoff for the penalty taker.

The same simulation is done for all 165 scenarios. This gives us the complete information about the payoffs for both players in all possible outcomes of the game. I could present this information in a pay-off matrix, just like I did in the toy game in Table 1. However, with 165 combinations, the matrix would be very large, so it wouldn't be very practical. The code used to find the payoffs in each scenario is presented in Appendix II.

Symmetry

In the model I've outlined, all assumptions are perfectly symmetrical. They are all mirrored across the line $x = 12$. When this is the case, nearly all scenarios have a different but symmetrical counterpart. For example, the scenario where the penalty taker aims for (4, 4) and the goalkeeper chooses "commit left", has a symmetrical counterpart where the penalty taker aims for coordinate (20, 4), and the goalkeeper chooses "commit right". The only scenarios that don't have a symmetrical counterpart are the ones where the penalty taker aims at a coordinate where $x = 12$, and the goalkeeper chooses "stay middle".

A scenario and its symmetric counterpart are equivalent. That is, the probability of a goal is the same regardless of if the scenario is (4, 4) and "commit left", or if it's (20, 4) and "commit right". Therefore, the symmetrical scenarios are assigned the average payoffs of the two scenarios, such that they end up with the same payoffs.

Perfectly symmetrical assumptions also have implications with regards to Nash equilibrium strategies. The equivalence between a scenario and its symmetrical counterpart implies that there must exist a set of Nash equilibrium strategies that are symmetrical across $x = 12$ as well. That is, there must exist a Nash equilibrium where "commit left" is played at the same frequency as "commit right", and coordinate (4, 4) is selected as often as coordinate (20, 4), et cetera.

An initial result

By implementing the counterfactual regret minimization (CFR) algorithm, I can locate the Nash equilibrium solution of the model. The essential parts of the code relating to the CFR-algorithm is presented in Appendix III. Let's select a velocity factor of 10, a standard deviation of 2, and a friction factor of 0.1, as the specific assumptions to solve for. Using 1,000,000 iterations to simulate payoffs for each scenario, I arrive at the solution presented in Table 2.

(4, 4): 0.427	(12, 4): 0.146	(20, 4): 0.427
Commit left: 0.461	Stay middle: 0.078	Commit right: 0.467
	Expected values: ±0.724	
	Exploitability: <0.0001 & <0.0001	

Table 2: An initial result. (Rounded to 3 decimals)

In the solution, the penalty taker has an expected value of 0.724, and the goalkeeper has an expected value of -0.724. This is because the probability of a goal is 0.724. The penalty taker selects three out of the 55 possible aiming coordinates with a non-zero frequency. These coordinates are (4, 4), (12, 4), and (20, 4). The penalty taker is indifferent between these coordinates, meaning that they yield the same expected value (which is also 0.724). All other aiming coordinates perform worse, yielding a lower expected value. For the goalkeeper, all three options are viable.

The strategies of the players are exploitable for <0.0001. This means that, assuming the strategy of the opponent is held constant, it's impossible for either player to gain more than 0.0001 in expected value by deviating from their current strategy. At the Nash equilibrium, both strategies will have an exploitability of zero. Since this is an approximation, the value is slightly positive – however, it's very low, meaning that the approximation is a very good one.

While the solution is good, its applicability to penalty kicks is only as good as the assumptions. In this case, the penalty taker had access to 55 different aiming options, whereas in reality, it's possible to aim anywhere in the goal. The 55 different options are spaced 2 feet apart. In the true equilibrium, the penalty taker may prefer aiming at coordinates in between those options.

One way to account for this would be to simply add more options. This method would work, but it wouldn't be very practical. The more options are added, the more computational time is required to find a good solution.

If I want the spacing between the options to be quite small, for example ⅒ of the current distance, the number of aiming options would grow to over 5000, making the required computational time extremely long. Therefore, I want a better way of determining which options should be available to the penalty taker.

Coordinate search

To locate the optimal aiming options, I developed an algorithm which I call coordinate search. The algorithm builds upon the framework I've presented so far but adds an iterative process which adds and removes aiming options based on the solutions the CFR-algorithm arrives at. Appendix IV presents the essential parts of the code relating to the coordinate search algorithm. Here, I will explain the algorithm conceptually.

The first iteration of the coordinate search algorithm is similar to what I have presented earlier. One exception is that the initial aiming options are spaced 1.6 feet apart, with the outer options being space 0.8 feet from the posts or the bar. This means that there are 90 initial aiming options instead of 55, and 270 total scenarios instead of 155. The spacing between the coordinates may seem arbitrary at first, but it's a conscious design choice. When 1.6 feet is divided by two several times, we get distances such as 0.4 feet, 0.2 feet, or 0.1 feet. These are examples of distances between aiming coordinates which I want solutions for.

Once the CFR-algorithm arrives at a solution, the coordinate search algorithm discards some aiming options, and adds some new ones. The options that were chosen at a non-zero frequency are kept, and the other options are discarded. The new coordinates are chosen such that they surround the coordinates that weren't discarded. As an example, if coordinate (4, 4) is played with a non-zero probability in the penalty taker's Nash equilibrium strategy, whereas the surrounding coordinates (2.4, 2.4), (2.4, 4), (2.4, 5.6), (4, 2.4), (4, 5.6), (5.6, 2.4), (5.6, 4), and (5.6, 5.6) are never played, then the coordinate (4, 4) is kept, and the surrounding coordinates are discarded. Furthermore, new coordinates surrounding (4, 4) are added, such that the distance between the points are halved (in this case from 1.6 feet to 0.8 feet). This means that coordinates (3.2, 3.2), (3.2, 4), (3.2, 4,8), (4, 3.2), (4, 4.8), (4.8, 3.2), (4.8, 4), and (4.8, 4.8) are added as new aiming options. If there were two other frequently selected coordinates - perhaps one in the middle, such as (12, 4), and one on the right, such as (20, 4) - we would have a total of 27 aiming options in the next iteration, as these options would also be surrounded by new coordinates.

In the first iteration of the coordinate search, the frequently selected options will always be surrounded by other coordinates (unless the coordinate is one of the outermost options). In the next iterations, it's quite common that a frequently selected option doesn't have any available options on some of its sides. To continue the example, if in the second iteration the coordinate (3.2, 4.8) is a playable option, there won't exist surrounding points to the left of it, or above it. In such a case, new options are added such that (3.2, 4.8) is surrounded, without the distance being halved. The aiming options for the next iteration on this side of the goal would then be (2.4, 4), (2.4, 4.8), (2.4, 5.6), (3.2, 4), (3.2, 4.8), (3.2, 5.6), (4, 4), (4, 4.8), and (4, 5.6). Some of these aiming options were present in the first iteration, but not in the second. The expected payoffs in the scenarios corresponding to some of those coordinates has already been calculated and can therefore be retrieved without additional simulation.

If the solution in the third iteration is such that coordinate (3.2, 4.8) still is being selected with a non-zero probability, the coordinate is kept, and new aim options are added around it. Since the coordinate (3.2, 4.8) is already surrounded by other coordinates (with a spacing of 0.8 feet), the distance is halved, such that the distance between the coordinate and the new coordinates are only 0.4 feet. The new coordinates will then be (2.8, 4.4), (2.8, 4.8), (2.8, 5.2), (3.2, 4.4), (3.2, 5.2), (3.6, 4.4), (3.6, 4.8), and (3.6, 5.2).

In addition to the mechanism I have presented so far, there's another way that new aiming options can be added. At each step of the coordinate search, the expected value of all the discarded options is calculated. This is done with respect to the strategy that the goalkeeper employed in the latest solution. If a previously discarded aiming option yields a higher expected value than the expected value of the strategy of the penalty taker, the coordinate is added back in again. New options are also added around the coordinate, in accordance with the same principles that have already been outlined. By adding new coordinates in between the previously available aiming coordinates, the coordinate search algorithm gradually locates better and better aiming options.

The user defines a desired specificity, which is the acceptable distance between coordinates. For instance, if the user sets the specificity to 0.2 feet, the algorithm will abstain from adding coordinates that are such that the distance to another coordinate will be less than 0.2 feet.

Eventually, the option search algorithm will locate a set of points which are likely to be optimal, given the acceptable specificity. This occurs when (1) all the aiming options that are selected with a non-zero probability are surrounded by other aiming options which are spaced out with a distance equal to the specificity, and (2) none of the discarded aiming options have a higher expected pay-off than the strategy of the penalty taker.

I previously stated that it would require over 5000 aiming options to check for every coordinate if the distance between the coordinates were just 0.2 feet. With the coordinate search algorithm, I effectively achieve the same result, without needlessly having to check thousands of options. This reduces the required computational time immensely, making it feasible to solve the game with such a high degree of coordinate-specificity. However, there is a technical possibility of missing optimal coordinates when using the coordinate search algorithm. The algorithm relies on identifying the optimal coordinates by means of nearby coordinates, assuming that if a coordinate is optimal, nearby coordinates will also be optimal in earlier iterations. Technically, there is the possibility that an in-between coordinate is viable, whereas no available nearby coordinates ever are. In such a case, that point would be missed.

If we imagine that the initial aim options were spaced out with a larger distance, such as 3.2 feet, or even 6.4 feet, this could start becoming likely. However, when the maximum distance between aiming options is as only 1.6 feet, optimal coordinates are very unlikely to be overlooked. I could have made the initial aiming options be spaced out by 0.8 feet instead, but this would increase the computational time, which is a trade-off I believe isn't worth making.

Using the coordinate search algorithm, I'm able to solve the game for a diverse set of assumptions. I used three different velocity-factors: High velocity ($v = 10$), medium velocity ($v = 8.5$), and low velocity ($v = 7$). Furthermore, there are seven different degrees of inaccuracy. Table 3, Table 4, and Table 5 present the solutions. I defined the specificity to be equal to 0.2 feet, meaning that no points can be spaced closer together than that distance. I used 100,000 iterations to simulate the available scenarios in the first step of the coordinate search, 1,000,000 iterations in the subsequent steps, and 10,000,000 iterations in the last step which arrives at the final solution. All strategies in the solutions have an exploitability of <0.00001, meaning that they are excellent approximations of a Nash equilibrium.

	High velocity (v=10)
SD=1	(2.6, 5.4): 0.49 / Commit left: 0.492; (12.0, 5.4): 0.019 / Stay middle: 0.015; (21.4, 5.4): 0.49 / Commit right: 0.492. *Exp. values: ±0.962* *Exploitability: <0.00001 & <0.00001*
SD=1.25	(2.8, 5.4): 0.478 / Commit left: 0.485; (12.0, 5.2): 0.044 / Stay middle: 0.031; (21.2, 5.4): 0.478 / Commit right: 0.485. *Exp. values: ±0.906* *Exploitability: <0.00001 & <0.00001*
SD=1.5	(2.8, 5.0): 0.461 / Commit left: 0.478; (12.0, 5.2): 0.077 / Stay middle: 0.044; (21.2, 5.0): 0.461 / Commit right: 0.478. *Exp. values: ±0.846* *Exploitability: <0.00001 & <0.00001*
SD=1.75	(3.0, 4.8): 0.45 / Commit left: 0.473; (12.0, 4.8): 0.006 / (12.0, 5.0): 0.094 / Stay middle: 0.055; (21.2, 4.8): 0.45 / Commit right: 0.473. *Exp. values: ±0.791* *Exploitability: <0.00001 & <0.00001*
SD=2	(3.2, 4.4): 0.44 / Commit left: 0.468; (12.0, 4.8): 0.12 / Stay middle: 0.064; (20.8, 4.4): 0.44 / Commit right: 0.468. *Exp. values: ±0.742* *Exploitability: <0.00001 & <0.00001*
SD=2.25	(3.6, 4.2): 0.441 / Commit left: 0.464; (12.0, 4.2): 0.001 / (12.0, 4.4): 0.118 / Stay middle: 0.073; (20.4, 4.2): 0.441 / Commit right: 0.464. *Exp. values: ±0.703* *Exploitability: <0.00001 & <0.00001*
SD=2.5	(3.6, 3.8): 0.006 / (3.8, 3.8): 0.436 / Commit left: 0.459; (12.0, 3.8): 0.115 / Stay middle: 0.081; (20.2, 3.8): 0.006 / (20.2, 3.8): 0.436 / Commit right: 0.459. *Exp. values: ±0.67* *Exploitability: <0.00001 & <0.00001*

Table 3: Nash equilibrium solutions when the penalty taker shoots with a high velocity (v=10). (Rounded to 3 decimals)

	Medium velocity (v=8.5)		
SD=1	(2.6, 5.8): 0.491 Commit left: 0.469	(12.0, 4.4): 0.007 (12.0, 4.6): 0.012 Stay middle: 0.061 *Exp. values: ±0.937* *Exploitability: <0.00001 & <0.00001*	(21.4, 5.8): 0.491 Commit right: 0.469
SD=1.25	(2.8, 5.6): 0.488 Commit left: 0.44	(12.0, 4.4): 0.025 Stay middle: 0.119 *Exp. values: ±0.871* *Exploitability: <0.00001 & <0.00001*	(21.2, 5.6): 0.488 Commit right: 0.44
SD=1.5	(2.8, 5.4): 0.488 Commit left: 0.415	(12.0, 4.2): 0.023 Stay middle: 0.171 *Exp. values: ±0.803* *Exploitability: <0.00001 & <0.00001*	(21.2, 5.4): 0.488 Commit right: 0.415
SD=1.75	(3.0, 5.0): 0.448 (3.0, 5.2): 0.044 Commit left: 0.394	(12.0, 4.0): 0.015 Stay middle: 0.213 *Exp. values: ±0.74* *Exploitability: <0.00001 & <0.00001*	(21.0, 5.0): 0.448 (21.0, 5.2): 0.044 Commit right: 0.394
SD=2	(3.0, 4.6): 0.225 (3.2, 4.8): 0.275 Commit left: 0.36	Stay middle: 0.28 *Exp. values: ±0.686* *Exploitability: <0.00001 & <0.00001*	(21.0, 4.6): 0.225 (20.8, 4.8): 0.275 Commit right: 0.36
SD=2.25	(3.0, 4.2): 0.107 (3.2, 4.2): 0.388 (3.2, 4.4): 0.005 Commit left: 0.308	Stay middle: 0.383 *Exp. values: ±0.64* *Exploitability: <0.00001 & <0.00001*	(21.0, 4.2): 0.107 (20.8, 4.2): 0.388 (20.8, 4.4): 0.005 Commit right: 0.308
SD=2.5	(3.2, 3.6): 0.475 (3.4, 3.8): 0.024 Commit left: 0.311	Stay middle: 0.377 *Exp. values: ±0.602* *Exploitability: <0.00001 & <0.00001*	(20.8, 3.6): 0.475 (20.6, 3.8): 0.024 Commit right: 0.311

Table 4: Nash equilibrium solutions when the penalty taker shoots with a medium velocity (v=8.5). (Rounded to 3 decimals)

		Low velocity (v=7)	
SD=1	(1.8, 5.4): 0.32 (2.0, 5.4): 0.18 Commit left: 0.027	Stay middle: 0.946 *Exp. values: ±0.885* *Exploitability: <0.00001 & <0.00001*	(22.2, 5.4): 0.32 (22.0, 5.4): 0.18 Commit right: 0.027
SD=1.25	(2.0, -0.4): 0.025 (2.0, 5.4): 0.475 Commit left: 0.051	Stay middle: 0.898 *Exp. values: ±0.787* *Exploitability: <0.00001 & <0.00001*	(22.0, -0.4): 0.025 (22.0, 5.4): 0.475 Commit right: 0.051
SD=1.5	(2.0, -0.6): 0.025 (2.0, 4.8): 0.475 Commit left: 0.069	Stay middle: 0.863 *Exp. values: ±0.696* *Exploitability: <0.00001 & <0.00001*	(22.0, -0.6): 0.025 (22.0, 4.8): 0.475 Commit right: 0.069
SD=1.75	(2.0, -0.6): 0.027 (2.0, 4.4): 0.473 Commit left: 0.082	Stay middle: 0.835 *Exp. values: ±0.62* *Exploitability: <0.00001 & <0.00001*	(22.0, -0.6): 0.027 (22.0, 4.4): 0.473 Commit right: 0.082
SD=2	(2.0, -0.2): 0.004 (2.0, 3.8): 0.494 (2.0, 4.0): 0.001 Commit left: 0.09	Stay middle: 0.819 *Exp. values: ±0.559* *Exploitability: <0.00001 & <0.00001*	(22.0, -0.2): 0.004 (22.0, 3.8): 0.494 (22.0, 4.0): 0.001 Commit right: 0.09
SD=2.25	(2.0, 3.0): 0.324 (2.2, 3.4): 0.176 Commit left: 0.104	Stay middle: 0.792 *Exp. values: ±0.51* *Exploitability: <0.00001 & <0.00001*	(22.0, 3.0): 0.324 (21.8, 3.4): 0.176 Commit right: 0.104
SD=2.5	(2.2, 1.6): 0.025 (2.2, 1.8): 0.09 (2.2, 2.0): 0.004 (2.2, 2.4): 0.377 (2.2, 2.6): 0.004 Commit left: 0.101	Stay middle: 0.798 *Exp. values: ±0.471* *Exploitability: <0.00001 & <0.00001*	(21.8, 1.6): 0.025 (21.8, 1.8): 0.09 (21.8, 2.0): 0.004 (21.8, 2.4): 0.377 (21.8, 2.6): 0.004 Commit right: 0.101

Table 5: Nash equilibrium solutions when the penalty taker shoots with a low velocity (v=7). (Rounded to 3 decimals)

The increased specificity of the permittable coordinates allows the penalty taker to increase their expected value. This is illustrated when comparing the initial result from Table 2 with the corresponding result in Table 3. In the initial result, the penalty taker ended up aiming for coordinates (4, 4), (12, 4), and (20, 4), as these were the best coordinates available. This yielded an expected value of 0.724. After running the coordinate search (with a specificity of 0.2 feet), I find that the preferred aim-coordinates are (3.2, 4.4), (12.0, 4.8), and (20.8, 4.4). When allowed to aim for those coordinates, their penalty taker's expected value increases to 0.742. In this instance, this effectively means that the penalty taker has managed to increase the probability of a goal by 1.8%, as a result of choosing slightly more optimal coordinates.

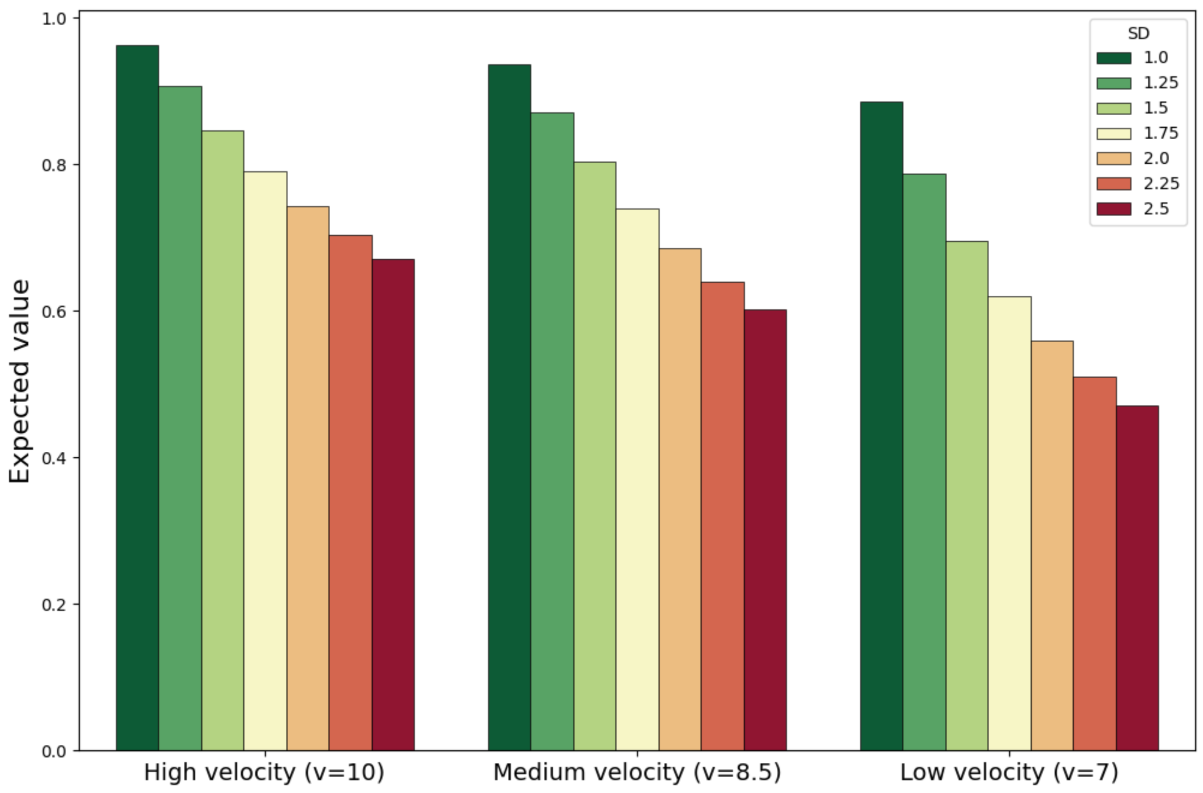


Figure 7: Expected values for the penalty taker in the Nash equilibrium solutions.

Figure 7 illustrates the expected value for the penalty taker for all the combinations of assumptions. We should expect the penalty taker to achieve a higher expected value if they are able to shoot more accurately. The figure demonstrates this, as an increase in standard deviation results in a decrease in expected value. Similarly, with all else being equal, we should expect the penalty taker to have a higher expectation when shooting with a higher velocity. This is because the area the goalkeeper is able to cover is smaller when the velocity is high. Again, this concept is illustrated in the figure. Assuming that the standard deviation is equal, the penalty taker's expected value is highest if the velocity is high, and lowest if the velocity is low. This holds for all the seven values of standard deviation that I solved for.

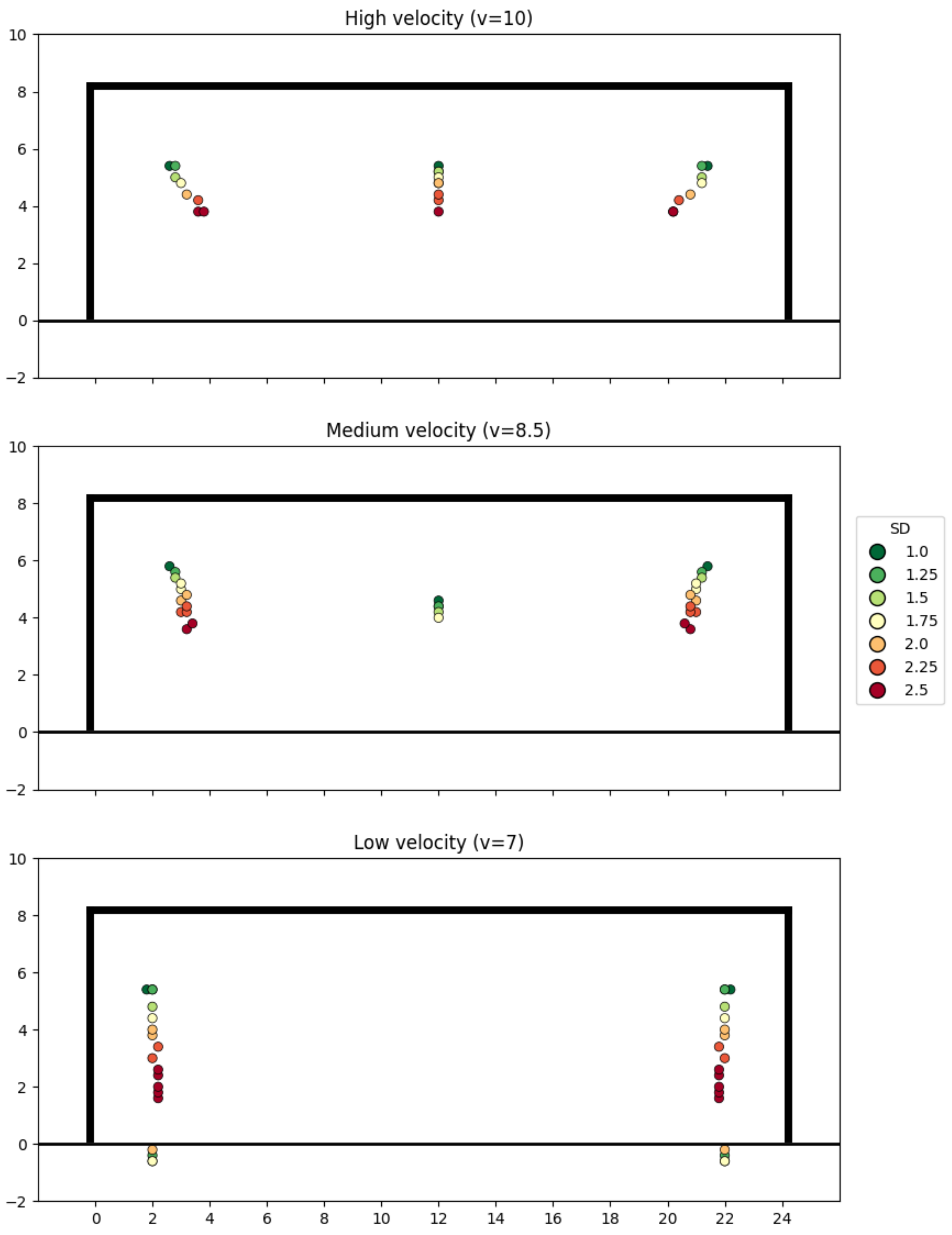


Figure 8: Aiming options utilized by the penalty taker in the Nash equilibrium solutions.

Figure 8 illustrates which aiming options the penalty taker utilizes in the various Nash equilibrium solutions. This is valuable for a couple of reasons. First of all, it demonstrates that there are large general areas of the goal which the penalty taker should avoid aiming for, regardless of the assumptions about inaccuracy and area coverage. Secondly, we see a clear trend between inaccuracy and the exact location of the optimal coordinates. If the inaccuracy increases, the optimal coordinate tends to shift downwards, and also horizontally towards the centre of the goal. This happens because if the inaccuracy of the penalty taker increases, and the same coordinate is aimed for, the chance of missing the goal increases. Therefore, the penalty taker adjusts their aim-coordinate such that there's a larger distance towards the post and the crossbar.

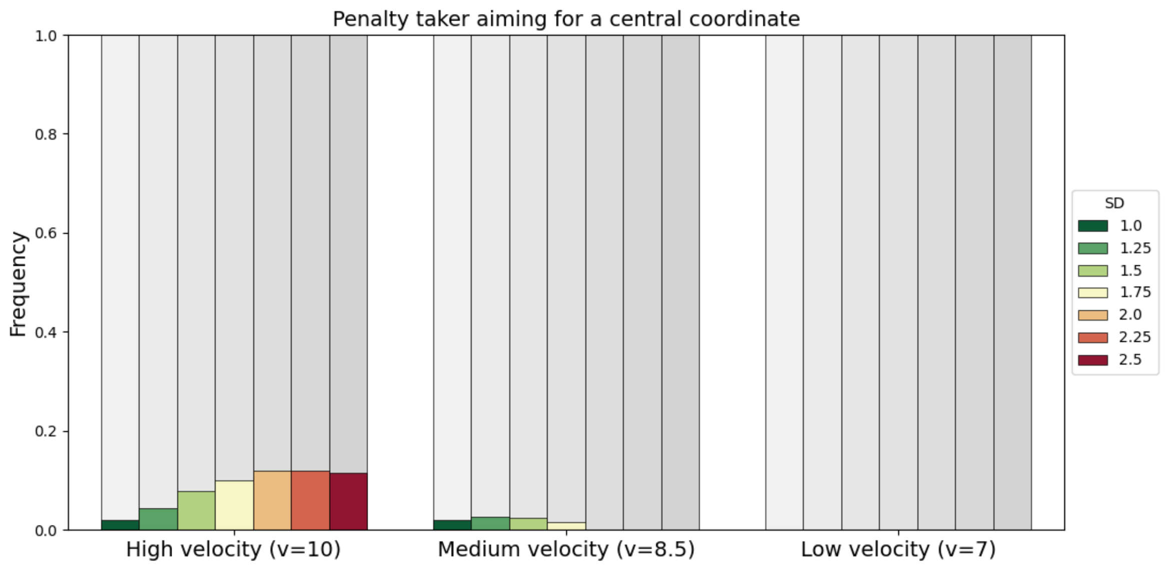


Figure 9: The frequencies of which the penalty taker selects a central coordinate.

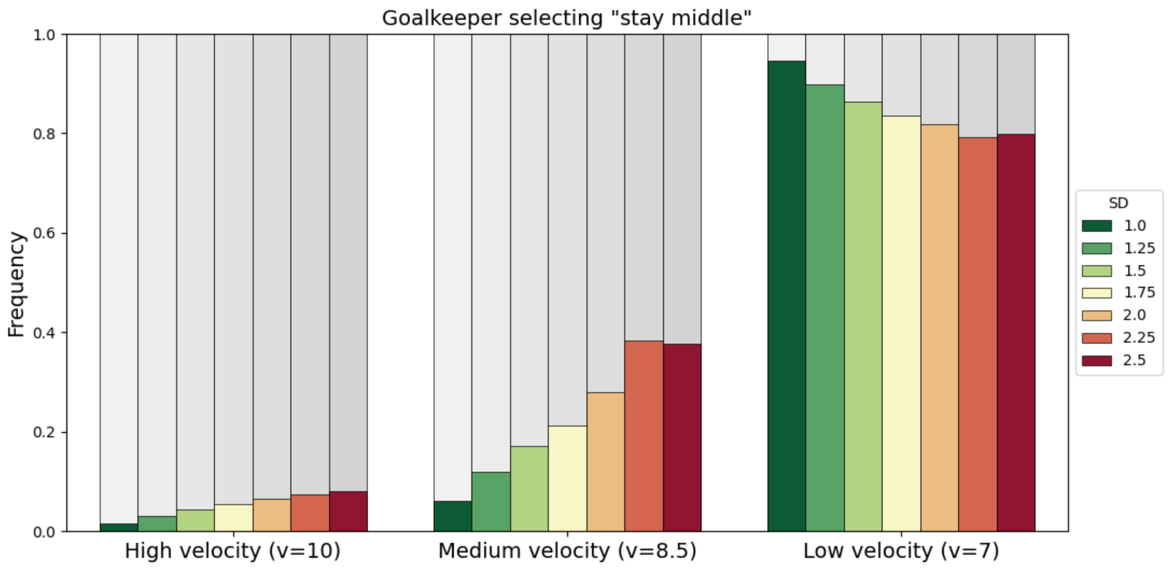


Figure 10: The frequencies at which the goalkeeper selects "stay middle".

Figure 9 and Figure 10 illustrate the frequencies at which the penalty taker and the goalkeeper select their respective central options (i.e., a coordinate at the line $x = 12$, or "stay middle"). These figures illustrate that the penalty taker only selects a central coordinate if the goalkeeper rarely does the same. This is because the goalkeeper virtually always manages to save the shot if both players select a central option. Additionally, we see that the goalkeeper selects "stay middle" more frequently if the velocity is lower. This makes sense, given that "stay middle" provides an increased reach when the velocity is lowered, whereas the "commit left"- and "commit right"-options are just shifter further towards the side.

Leaning options

So far, the goalkeeper has been restricted to three options: "commit left", "stay middle", and "commit right". The "commit left"- and "commit right"-options fully commit the goalkeeper to diving to a specific side, without having observed the trajectory of the ball. In the "stay middle"-option, the goalkeeper stays in the middle and observes the trajectory before moving. While these are options that goalkeepers frequently utilize in real-life penalty kicks, they aren't the only possible options. It's also conceivable that the goalkeeper can start to lean towards a side, without fully committing to diving in a certain direction, such as in the "commit left"- and "commit right"-options.

To model this alternative, I introduce two new options for the goalkeeper: "lean left" and "lean right". These options can be viewed as half-committing to a certain direction. Compared to the "stay middle"-option, the advantage is that the leaning-options allow the goalkeeper to have additional reach in a certain direction, while also covering the central parts of the goal. However, when leaning in a certain direction, the disadvantage is that the reach of the goalkeeper in the opposite direction is reduced. The leaning-options can be described mathematically:

$$Lean\ left: \frac{(x-7.5-0.25v)^2}{7}+\frac{(y-1.5-0.2v)^2}{7} \leq 18-1.4v$$

$$Lean\ right: \frac{(x-16.5+0.25v)^2}{7}+\frac{(y-1.5-0.2v)^2}{7} \leq 18-1.4v$$

Figure 11, Figure 12, and Figure 13 illustrate the possible area coverage for the goalkeeper when the velocity is high ($v = 10$), medium ($v = 8.5$), and low ($v = 7$), with the leaning-options included.

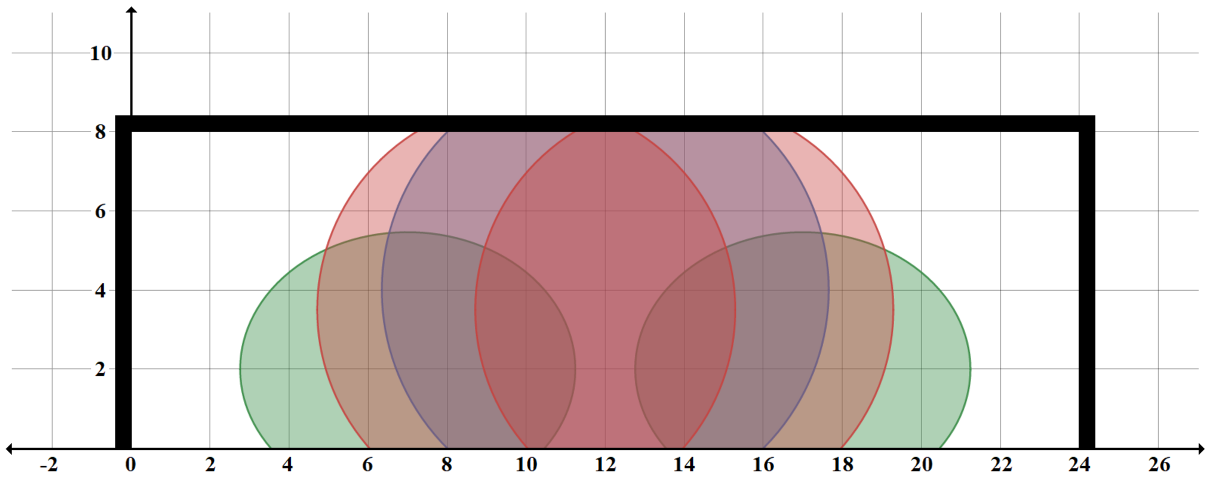


Figure 11: Possible area coverage when the velocity-factor is equal to 10.

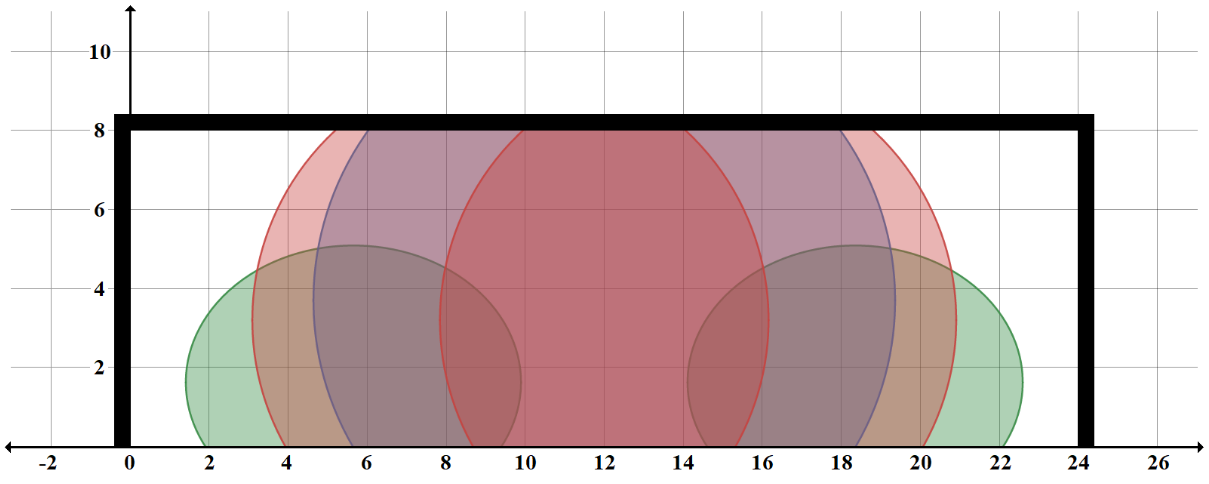


Figure 12: Possible area coverage when the velocity-factor is equal to 8.5.

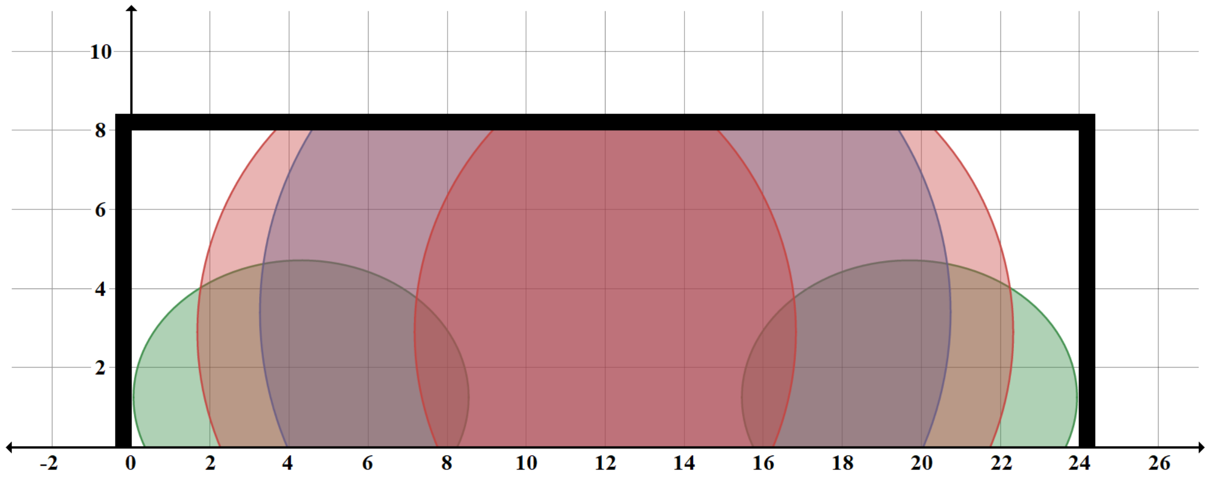


Figure 13: Possible area coverage when the velocity-factor is equal to 7.

Just like before, but with the leaning options included, I can run the coordinate search algorithm to solve for a diverse set of assumptions. Again, the desired specificity is set to 0.2 feet. In the first step of the coordinate search, 100,000 iterations are used to simulate scenario payoffs, whereas in the subsequent steps 1,000,000 iterations are used, and in the final step 10,000,000 iterations are used. All strategies in the solutions have an exploitability of less than 0.00001. The results are presented in Table 6, Table 7, and Table 8.

	High velocity (v=10)		
SD=1	(2.6, 5.4): 0.492 Lean left: 0.008 Commit left: 0.492	(12.0, 5.4): 0.016 *Exp. values: ±0.962* *Exploitability: <0.00001 & <0.00001*	(21.4, 5.4): 0.492 Lean right: 0.008 Commit right: 0.492
SD=1.25	(2.6, 5.2): 0.484 Lean left: 0.016 Commit left: 0.484	(12.0, 5.2): 0.033 *Exp. values: ±0.906* *Exploitability: <0.00001 & <0.00001*	(21.4, 5.2): 0.484 Lean right: 0.016 Commit right: 0.484
SD=1.5	(2.8, 5.0): 0.47 (3.0, 5.2): 0.004 Lean left: 0.023 Commit left: 0.477	(12.0, 5.0): 0.052 *Exp. values: ±0.845* *Exploitability: <0.00001 & <0.00001*	(21.2, 5.0): 0.47 (21.0, 4.2): 0.004 Lean right: 0.023 Commit right: 0.477
SD=1.75	(3.0, 4.8): 0.468 Lean left: 0.03 Commit left: 0.47	(12.0, 4.8): 0.003 (12.0, 5.0): 0.062 *Exp. values: ±0.789* *Exploitability: <0.00001 & <0.00001*	(21.0, 4.8): 0.468 Lean right: 0.03 Commit right: 0.47
SD=2	(3.2, 4.4): 0.018 (3.2, 4.6): 0.444 (3.4, 4.6): 0.003 Lean left: 0.037 Commit left: 0.463	(12.0, 4.6): 0.072 *Exp. values: ±0.74* *Exploitability: <0.00001 & <0.00001*	(20.8, 4.4): 0.018 (20.8, 4.6): 0.444 (20.6, 4.6): 0.003 Lean right: 0.037 Commit right: 0.463
SD=2.25	(3.4, 4.0): 0.033 (3.6, 4.2): 0.426 Lean left: 0.043 Commit right: 0.457	(12.0, 4.4): 0.082 *Exp. values: ±0.7* *Exploitability: <0.00001 & <0.00001*	(20.6, 4.0): 0.033 (20.4, 4.2): 0.426 Lean right: 0.043 Commit right: 0.457
SD=2.5	(3.6, 3.6): 0.004 (3.6, 3.8): 0.001 (3.8, 3.6): 0.009 (3.8, 3.8): 0.443 Lean left: 0.048 Commit left: 0.452	(12.0, 3.8): 0.006 (12.0, 4.0): 0.079 *Exp. values: ±0.668* *Exploitability: <0.00001 & <0.00001*	(20.4, 3.6): 0.004 (20.4, 3.8): 0.001 (20.2, 3.6): 0.009 (20.2, 3.8): 0.443 Lean right: 0.048 Commit right: 0.452

Table 6: Nash equilibrium solutions when the penalty taker shoots with a low velocity (v=10). (Rounded to 3 decimals)

	Medium velocity (v=8.5)		
SD=1	(2.0, 5.2): 0.436 (2.2, 5.4): 0.064 Lean left: 0.488 Commit left: 0.012	*Exp. values: ±0.915* *Exploitability: <0.00001 & <0.00001*	(22.0, 5.2): 0.436 (21.8, 5.4): 0.064 Lean right: 0.488 Commit right: 0.012
SD=1.25	(2.2, 5.0): 0.333 (2.2, 5.2): 0.167 Lean left: 0.43 Commit left: 0.07	*Exp. values: ±0.844* *Exploitability: <0.00001 & <0.00001*	(21.8, 5.0): 0.333 (21.8, 5.2): 0.167 Lean right: 0.43 Commit right: 0.07
SD=1.5	(2.4, 4.8): 0.313 (2.4, 5.0): 0.187 Lean left: 0.352 Commit left: 0.148	*Exp. values: ±0.777* *Exploitability: <0.00001 & <0.00001*	(21.6, 4.8): 0.313 (21.6, 5.0): 0.187 Lean right: 0.352 Commit right: 0.148
SD=1.75	(2.6, 4.6): 0.286 (2.8, 4.6): 0.214 Lean left: 0.297 Commit left: 0.203	*Exp. values: ±0.719* *Exploitability: <0.00001 & <0.00001*	(21.4, 4.6): 0.286 (21.2, 4.6): 0.214 Lean right: 0.297 Commit right: 0.203
SD=2	(3.0, 4.2): 0.325 (3.0, 4.4): 0.175 Lean left: 0.289 Commit left: 0.211	*Exp. values: ±0.673* *Exploitability: <0.00001 & <0.00001*	(21.0, 4.2): 0.325 (21.0, 4.4): 0.175 Lean right: 0.289 Commit right: 0.211
SD=2.25	(3.2, 3.8): 0.171 (3.2, 4.0): 0.329 Lean left: 0.282 Commit left: 0.218	*Exp. values: ±0.635* *Exploitability: <0.00001 & <0.00001*	(20.8, 3.8): 0.171 (20.8, 4.0): 0.329 Lean right: 0.282 Commit right: 0.218
SD=2.5	(3.2, 3.4): 0.004 (3.2, 3.6): 0.429 (3.2, 3.8): 0.067 Commit left: 0.313	Stay middle: 0.374 *Exp. values: ±0.602* *Exploitability: <0.00001 & <0.00001*	(20.8, 3.4): 0.004 (20.8, 3.6): 0.429 (20.8, 3.8): 0.067 Commit right: 0.313

Table 7: Nash equilibrium solutions when the penalty taker shoots with a low velocity (v=8.5). (Rounded to 3 decimals)

	Low velocity (v=7)		
SD=1	(1.6, 5.8): 0.5 Lean left: 0.5	*Exp. values: ±0.761* *Exploitability: <0.00001 & <0.00001*	(22.4, 5.8): 0.5 Lean right: 0.5
SD=1.25	(1.8, 5.4): 0.5 Lean left: 0.5	*Exp. values: ±0.676* *Exploitability: <0.00001 & <0.00001*	(22.2, 5.4): 0.5 Lean right: 0.5
SD=1.5	(2.0, 4.8): 0.478 (2.0, 5.0): 0.022 Lean left: 0.5	*Exp. values: ±0.616* *Exploitability: <0.00001 & <0.00001*	(22.0, 4.8): 0.478 (22.0, 5.0): 0.022 Lean right: 0.5
SD=1.75	(2.4, -0.4): 0.17 (2.4, 4.0): 0.003 (2.4, 4.2): 0.318 (2.4, 4.4): 0.009 Lean left: 0.483 Commit left: 0.017	*Exp. values: ±0.579* *Exploitability: <0.00001 & <0.00001*	(21.6, -0.4): 0.17 (21.6, 4.0): 0.003 (21.6, 4.2): 0.318 (21.6, 4.4): 0.009 Lean right: 0.483 Commit right: 0.017
SD=2	(2.4, 0.2): 0.025 (2.4, 3.4): 0.356 (2.4, 3.8): 0.008 (2.6, 0.0): 0.057 (2.6, 3.6): 0.053 Lean left: 0.386 Commit left: 0.039	Stay middle: 0.148 *Exp. values: ±0.549* *Exploitability: <0.00001*	(21.6, 0.2): 0.025 (21.6, 3.4): 0.356 (21.6, 3.8): 0.008 (21.4, 0.0): 0.057 (21.4, 3.6): 0.053 Lean right: 0.386 Commit right: 0.039
SD=2.25	(2.0, 3.2): 0.343 (2.2, 3.0): 0.107 (2.2, 3.2): 0.05 Commit left: 0.103	Stay middle: 0.794 *Exp. values: ±0.51* *Exploitability: <0.00001 & <0.00001*	(22.0, 3.2): 0.343 (21.8, 3.0): 0.107 (21.8, 3.2): 0.05 Commit right: 0.103
SD=2.5	(2.2, 2.0): 0.181 (2.2, 2.4): 0.318 Commit left: 0.104	Stay middle: 0.792 *Exp. values: ±0.471* *Exploitability: <0.00001 & <0.00001*	(21.8, 2.0): 0.181 (21.8, 2.4): 0.318 Commit right: 0.104

Table 8: Nash equilibrium solutions when the penalty taker shoots with a low velocity (v=7). (Rounded to 3 decimals)

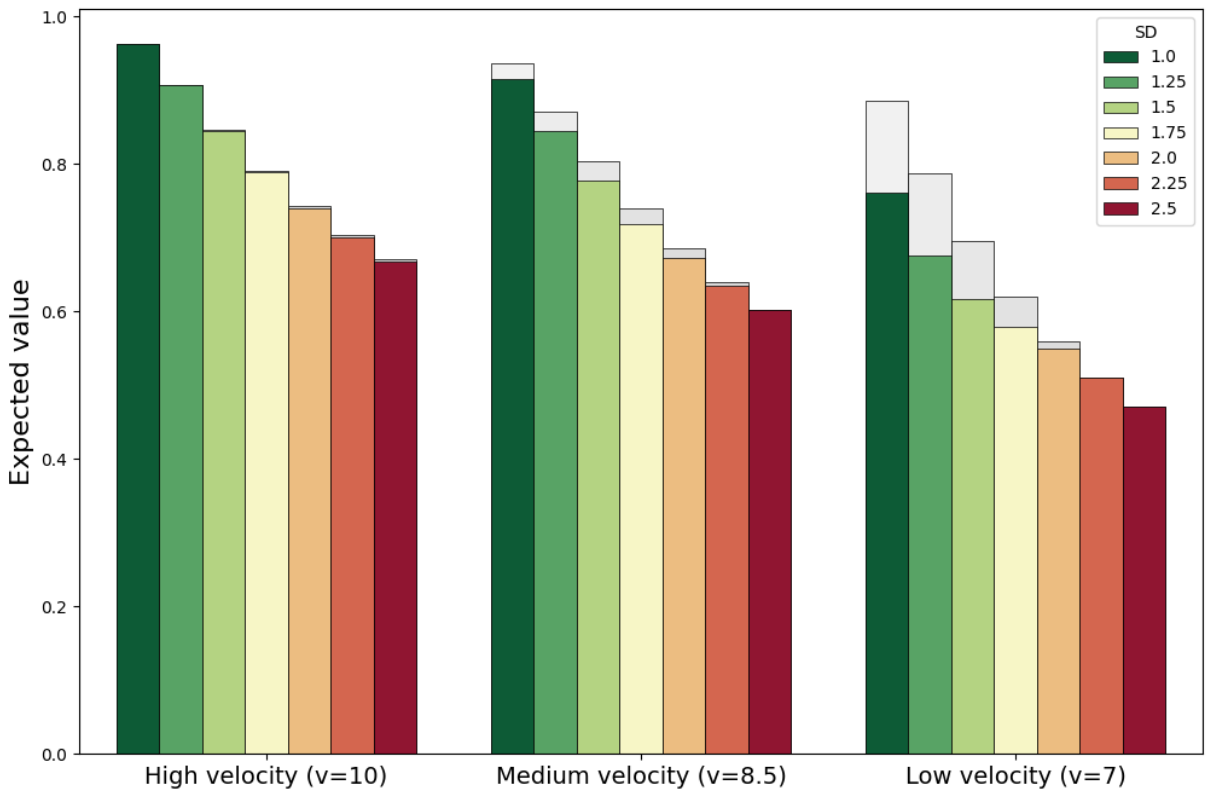


Figure 14: The expected value for the penalty taker in the Nash equilibrium solutions.

Figure 14 illustrates the expected value for the penalty taker in all the different solutions. Once again, we observe that the expected value is correlated with the penalty taker's ability to shoot accurately, and that a higher velocity yields a higher expected value, given the same standard deviation. The figure also illustrates the reduction in the penalty taker's expected value compared to the previous solution, where the goalkeeper only had three available options. The expected values in the previous solutions are illustrated by the grey bars in the background.

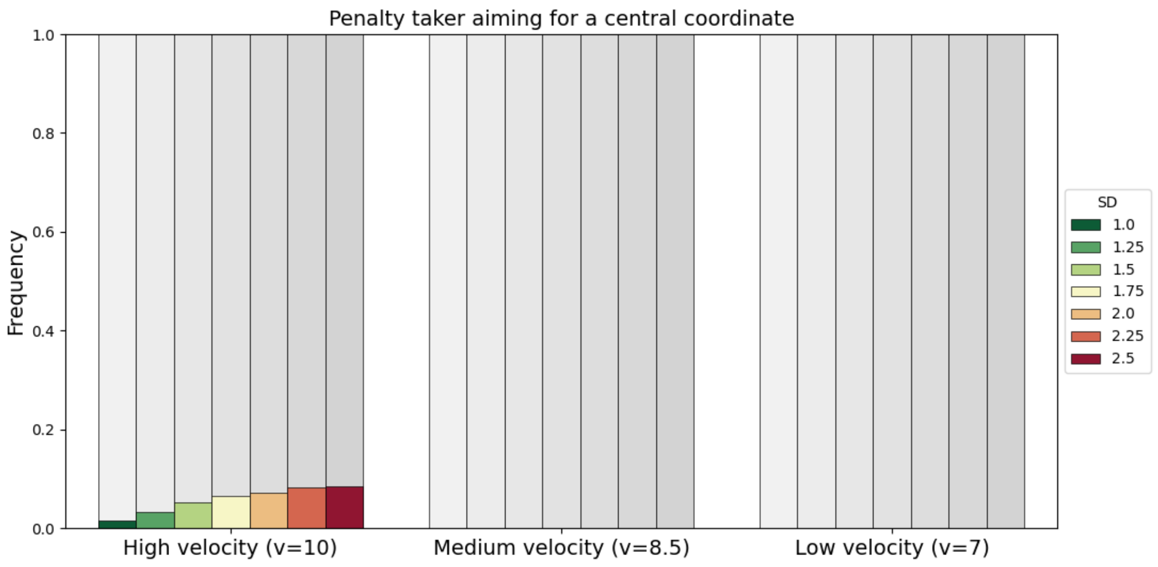


Figure 15: The frequency of which the penalty taker aims at a central coordinate.

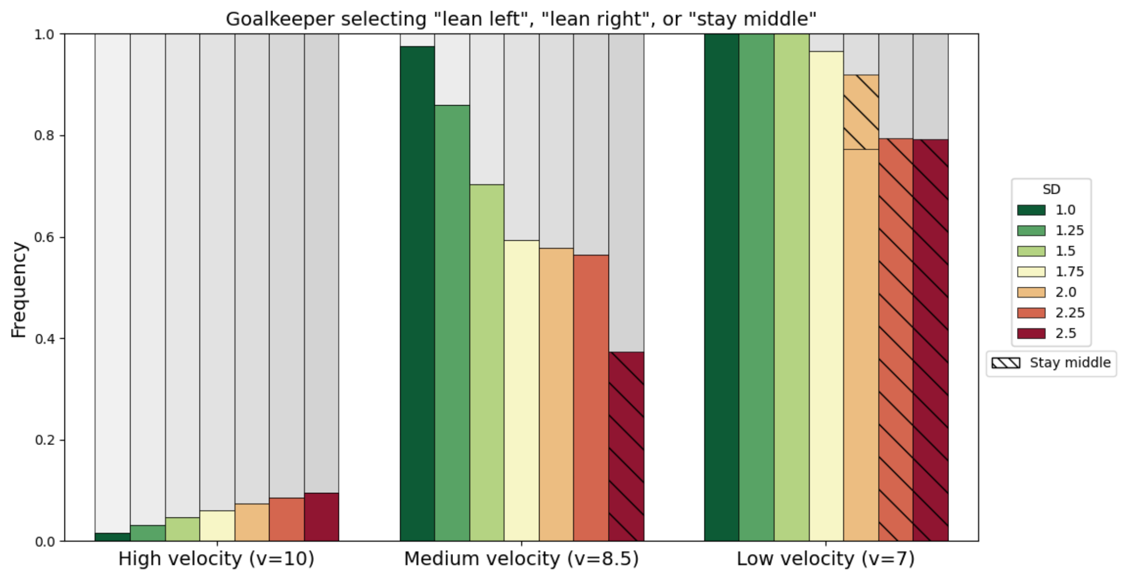


Figure 16: The frequency of which the goalkeeper selects an option that covers the central coordinates.

Figure 15 illustrates the frequencies of which the penalty taker aims at a central coordinate ($x = 12$). Figure 16 illustrates how often the goalkeeper selects an option that can cover those central coordinates. In this new solution, this includes "stay middle", as before, but also "lean left" and "lean right". As in the previous solution, we see that the penalty taker only aims for a central coordinate if the goalkeeper rarely selects those options.

In many combinations of assumptions, the "lean left"- and "lean right"-options are now preferred instead of the "stay middle"-option. The leaning options have a very good save rate assuming that the penalty taker aims at a central coordinate, but also have a better chance of stopping the shot if the penalty taker shoots to the same direction they are leaning. However, if the penalty taker shoots to the opposite direction, the leaning options have virtually no chance of stopping the shot. On the other hand, given most assumptions, the "stay middle"-option can stop the shot to either side, but the probability is quite low. We see that "stay middle" is still being preferred when the velocity is lowered, and standard deviation is quite high. This happens because the "stay middle"-option actually has quite a far reach given those assumptions. However, in most combinations of assumptions, it's better to choose the leaning options and half-commit to one of the sides, and still cover the middle. That said, in a later model, we will see that the leaning options may not be quite as good as they seem in this model.

The reduction expected values illustrated in Figure 14 should also be understood in relation to when the leaning options are being utilized. This and the previous model are identical with the exception of the goalkeeper having two new available options. There are only two outcomes: Either the new options have zero impact, or they increase the goalkeeper's expected value (and thereby reduce the penalty taker's). If the options aren't viable, they will have zero impact on results, but if they are, they will improve the goalkeeper's expectancy. The leaning options are frequently utilized when standard deviation is low and when velocity is medium or low. These are also the combinations of assumptions where the penalty taker's expected value is reduced.

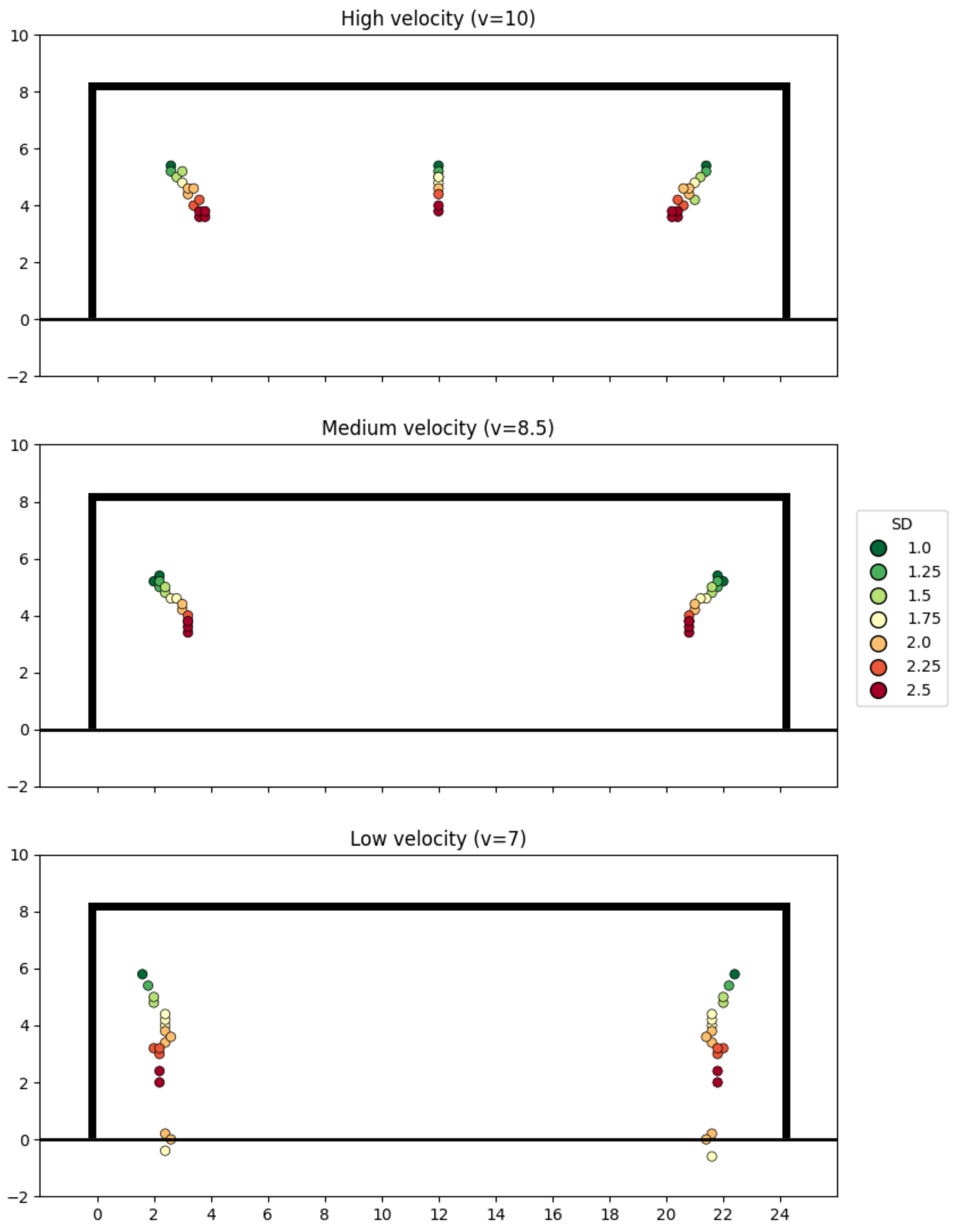


Figure 17: Aiming options utilized by the penalty taker in the Nash equilibrium solutions.

Figure 17 illustrates which aiming options are utilized by the penalty taker in the Nash equilibrium solutions. Again, there are large areas of the goal where the penalty taker should never aim, and there's a clear trend between the exact location of the optimal points and the standard deviation. A higher standard deviation leads to lower aim-coordinates that are horizontally closer to the centre of the goal.

Non-uniform velocity

Up until this point, it has been assumed that the penalty taker is forced to shoot with a specific pre-defined velocity-factor, and that the velocity-factor is uniform across all aiming-coordinates. Additionally, the goalkeeper has had perfect information about which specific velocity-factor that has been available to the penalty taker. These assumptions aren't realistic. In a real-life penalty kick, the penalty taker has the flexibility to vary the velocity-factor for different aiming-coordinates, or even to mix up which velocity-factor is selected for the same coordinate. This adds an additional layer of decision-making to their strategy. The goalkeeper will still have perfect information about which velocity-factors are allowed. However, now there will be potential variability in the selected velocity-factor. This introduces the need for the goalkeeper to design their strategy in a more robust manner, such that it can handle all the possible choices of velocity.

In this iteration of the framework, I will allow the penalty taker to choose between the three different velocity-factors we have become familiar with. That is, high velocity ($v = 10$), medium velocity ($v = 8.5$), and low velocity ($v = 7$). In principle, you could add many more, but for every option that's added, a whole new set of scenarios become possible, increasing the computational time required to solve the game. These three options are sufficient to illustrate the important changes that occur in the equilibrium strategies.

I also considered adding making the achieved velocity follow a probability distribution, such that it could be different from the velocity that's aimed for. This would be similar to how the hit-coordinate differs from the aim-coordinate, and I do believe this would be more realistic. However, unlike with the hit- and aim-coordinates, I don't believe that adding a probabilistic element to determine the achieved velocity adds anything significant to the model, such that results change in any interesting ways. It only increases the variability in results, making the CFR-algorithm converge slower than it otherwise would have. Therefore, I decided to make the achieved velocity be entirely deterministic. This means that, absent from friction with the ground, the velocity that's aimed for is the velocity that's achieved.

Now that there's variability in the possible choices of velocity, we need to account for how these impacts other assumptions in the framework. Although this will be player-dependent, I believe it would generally be safe to assume that the penalty taker's positional accuracy varies based on the velocity-factor. I would expect players to be able to shoot more accurately when shooting with a lower velocity, and vice versa, meaning that there's a trade-off between velocity and accuracy.

I adapted both the standard CFR-algorithm and the coordinate search algorithm to be able to include a choice of velocity. The code is largely still similar to the code presented in Appendix II, Appendix III, and Appendix IV,so it does not warrant its own appendix. However, the complete code can be found at github.com/Monstad/Penalty-Kicks.

In the model, I assume that the penalty taker has a standard deviation of 2 when choosing the high velocity ($v = 10$), a standard deviation of 1.75 when selecting the medium velocity ($v = 8.5$), and a standard deviation of 1.5 when choosing the low velocity ($v = 7$). This encompasses the trade-off between accuracy and velocity. When running a coordinate search with a specificity of 0.2 feet, I get the solution presented in Table 9. In the first step of the coordinate search, the payoffs are simulated using 100,000 iterations, whereas in the subsequent steps it uses 1,000,000 iterations, and in the final step it uses 10,000,000.

v=10, (3.2, 4.2): 0.466	v=7, (12.0, 3.0): 0.027	v=10, (20.8, 4.2): 0.466
Commit left: 0.377	v=7, (12.0, 3.2): 0.041	Commit right: 0.377
Lean left: 0.123		Lean right: 0.123
	Expected values: ±0.751	
	Exploitability: <0.00001 & <0.00001	

Table 9: The Nash equilibrium solution in a model with non-uniform velocity. (Rounded to 3 decimals)

The solution illustrates a concept that can be applied more generally: When aiming in the middle of the goal, it's favourable to shoot with a lower velocity. There are two reasons for this. Firstly, when the penalty taker shoots with a lower velocity, and the goalkeeper selects "commit left" or "commit right", the area covered by the goalkeeper is further away from the centre of the goal. This is because the goalkeeper is already in the air, moving away from the centre, and when the velocity is lower, the ball reaches the goal at a later point in time, meaning that the goalkeeper has moved further away. Secondly, because of the trade-off between accuracy and velocity, the penalty taker is more accurate, meaning that the chance of a goal if the goalkeeper guesses incorrectly is further increased.

From previous models, we know if the goalkeeper chooses "lean left", "stay middle", or "lean right" too often, the expected value of aiming at a central coordinate really suffers. It's only viable for the penalty taker to aim for a central coordinate if the goalkeeper employs a strategy which involves choosing “commit left” or “commit right” quite frequently. Additionally, it’s under these conditions the strategy of shooting with a low velocity when aiming centrally shines. Therefore, we can conclude that if it’s viable to aim for a central coordinate, this should be done using a low velocity.

The solution of Table 9 can be compared with the solution in Table 6 where the penalty taker is forced to shoot with a uniformly high velocity ($v = 10$) and a standard deviation of 2. In both solutions, the penalty taker aims at a central coordinate with approximately the same frequency; 0.072 in the uniform model, and 0.068 in the non-uniform one. However, in the uniform model, the goalkeeper only chooses an option that covers a central coordinate (in this case, “lean left” and “lean right”) with a total frequency of 0.064. In the non-uniform model, “lean left” and “lean right” are chosen at a combined frequency of 0.262. The goalkeeper has to adjust in this manner, because otherwise the success rate of aiming at a central coordinate becomes too good. This is a consequence of how advantageous it is to aim centrally with a low velocity. In the uniform model, when the penalty taker has to shoot with a velocity of 10 when aiming centrally, there is a greater chance of missing the goal (as a consequence of the velocity-accuracy trade-off), and also a greater chance of occasionally stopping the shot despite choosing “commit left” or “commit right” (because of both factors).

When the goalkeeper employs the leaning options more often, it also has consequences for which side-coordinates are optimal to aim for. The “commit left”- and “commit right”-options have the furthest reach, and since the penalty taker knows that these will occur more rarely, it becomes optimal choose an aim-coordinate with a slightly larger margin of error in respect to the post and the crossbar. Therefore, the side-coordinate shift slightly downwards and towards the centre of the goal. Compared to the solution of the uniform model, the additional choice of velocity has shifted the equilibrium in such a way that the expected value for the penalty taker increases from 0.74 to 0.751.

Sequential choices

In all models, I have treated penalty kicks as a simultaneous choice game with one decision point. In reality, a penalty kick is a dynamic event with multiple decision points. It involves strategic interaction that unfolds over time, not merely at one isolated moment.

To some extent, the previous models have implicitly implemented this with regards to the area coverage of the goalkeeper. The goalkeeper makes a simultaneous choice of “commit left”, "lean left", "stay middle", "lean right", or "commit right", without observing the trajectory of the ball. However, once that choice is made, and the trajectory is observed, it's assumed that the goalkeeper will adapt to the information about where the ball is headed. A specific area coverage doesn't really represent an area which the goalkeeper can cover simultaneously. Rather, it represents the area that the goalkeeper can cover after observing the trajectory of the ball, contingent on the choice of area. For example, if the goalkeeper chooses "lean left", the goalkeeper already starts leaning to the left without seeing where the ball is headed. Still, at some point, the goalkeeper will observe where the ball is going, and adapt to it. If the goalkeeper observes the ball going far left, they can start diving to the left, and if they observe the ball going towards the middle, they can start reversing the lean and try to cover the middle. The goalkeeper also has the option of "stay middle", which involves not moving until the trajectory of the ball is observed. So, in previous models, there is an element of sequential choice in the goalkeeper's actions.

On the other hand, it has always been assumed that that penalty taker has zero information about the choice of the goalkeeper when deciding where to aim. In reality, the penalty taker sometimes tries to obtain information about the goalkeeper's intended move before making their final decision. A common tactic is to try to pretend to shoot, and see if the goalkeeper moves. If this is successful, and the goalkeeper starts to move, the penalty taker can shoot in the opposite direction. In this model, I'm going to implement such an option.

It's possible to implement this while also maintaining the structure of a simultaneous choice game with one decision point. I can add the option "fake the shot" to the penalty taker's aim-options. When choosing this option, the penalty taker does the run-up to the ball, as if they are about to shoot, but then stops to see if the goalkeeper reveals their action. Instead of simulating the payoffs in the scenarios associated with this option, these can be added exogenously. For example, we can assume that in the scenarios where the goalkeeper starts to move, i.e., in the scenarios ["fake the shot", "commit left"], ["fake the shot", "lean left"], ["fake the shot", "lean right"], and ["fake the shot", "commit right"], the penalty taker will score with a 100% probability, making the expected payoff equal to 1. This is because the penalty taker can simply shoot in the opposite direction to where the goalkeeper is moving.

In the remaining scenario, ["fake the shot", "stay middle"], the goalkeeper hasn't moved, and will stand a much better chance of stopping the shot. According to FIFA-regulations, the penalty taker is not allowed to step backwards and do the run-up to the ball once again. This effectively means that the penalty taker is forced to shoot from a position of standing still. When this is the case, it rules out shooting with a high velocity, as this would only be feasible with the momentum from a run-up. When the penalty taker has to shoot with a velocity that's lower than usual, the goalkeeper will be able to cover a larger area, and stand a better chance of preventing a goal. This is the drawback of trying to "fake the shot".

If we enter the scenario of ["fake the shot", "stay middle"], one might think that the penalty taker can attempt to "fake the shot" once again. However, this shouldn't be effective. When the shot has been faked once, the penalty taker is already forced to shoot standing still. Therefore, if "fake the shot" is attempted once again, there can be no additional drawback, and the goalkeeper knows this. If the goalkeeper then ever selects anything other than "stay middle", the penalty taker should always continue faking the shot, hoping that the goalkeeper will eventually move. However, if the penalty taker always selects "fake the shot", the goalkeeper should always select "stay middle". Therefore, we can assume that the goalkeeper always selects "stay middle" after the first time the penalty taker selects "fake the shot". This means that we don't have to consider any sort of infinite regress where the penalty taker always selects "fake the shot".

The penalty taker should have a lower expected payoff in the scenario ["fake the shot", "stay middle"], compared to their overall expectancy. In this model, I'm going to assume that the penalty taker manages to score 60% of the time in this scenario, making the expected payoffs 0.6 and -0.6. I adapted the code to accommodate for the inclusion of the "fake the shot" option. The complete code can be found at github.com/Monstad/Penalty-Kicks.

The model builds upon the non-uniform velocity model. This means that it includes the three choices of velocity, i.e., high ($v = 10$), medium ($v = 8.5$), and low ($v = 7$). The trade-off between velocity and accuracy is the same as in the previous model, meaning that for high velocity, the standard deviation is 2, for medium it's 1.75, and for low it's 1.5.

All five options are available to the goalkeeper: "commit left", "lean left", "stay middle", "lean right", and "commit right". When running a coordinate search with a specificity of 0.2 feet, I get the solution presented in Table 10. In the first step of the coordinate search, the payoffs are simulated using 100,000 iterations, whereas in the subsequent steps it uses 1,000,000 iterations, and in the final step it uses 10,000,000.

v=10, (3.2, 3.6): 0.347	Fake the shot: 0.268	v=10, (20.8, 3.6): 0.347
v=10, (3.2, 3.8): 0.007	Stay middle: 0.509	v=10, (20.8, 3.8): 0.007
v=10, (3.4, 3.6): 0.007		v=10, (20.6, 3.6): 0.007
v=10, (3.4, 3.8): 0.005		v=10, (20.6, 3.8): 0.005
Commit left: 0.246	*Expected values: ±0.796*	Commit right: 0.246
	Exploitability: <0.00001 & <0.00001	

Table 10: The Nash equilibrium solution in a model with "fake the shot". (Rounded to 3 decimals)

Allowing the penalty taker to "fake the shot" forces the goalkeeper to select "stay middle" with a relatively high frequency (0.509). This is because "stay middle" is the only option that performs well against "fake the shot". As a consequence of the frequent utilization of "stay middle", the penalty taker never aims for a central coordinate.

Unlike in the previous models, the goalkeeper completely stops utilizing the "lean left"- and "lean right"-options. If the penalty taker selects "fake the shot", the leaning options perform very poorly, as they reveal that one of the sides in the goal will be open. Additionally, if the penalty taker decides to aim for one of the sides, the leaning options underperform compared to the "commit left"- or "commit right"-options. So, in this model, the leaning options are much worse than they seemed in previous models. They are only to be used against penalty takers that rarely or never chooses to "fake the shot".

Compared to the solution of the non-uniform velocity model, the expected value for the penalty taker has increased from 0.751 to 0.796, which is quite a significant increase. This illustrates that "fake the shot" is a quite effective option. It essentially forces the goalkeeper to "stay middle" quite often, which in turn increases the effectiveness of aiming for a side-coordinate. This is much preferable to the previous strategy, which attempted to deter the goalkeeper from choosing "commit left" or "commit right" by sometimes aiming at central coordinates, using a low velocity.

Asymmetry

Throughout the framework, I have operated with assumptions that are perfectly symmetrical and mirrored across the line $x = 12$. In a real-life penalty kick, the player-dependent assumptions may not always be perfectly symmetrical. In the run-up to the ball, the penalty taker usually runs at an angle, creating a curved trajectory as they approach the ball. In other words, the run-up is usually not a straight line, and is therefore asymmetrical. Additionally, the penalty taker kicks the ball, and the foot of the penalty taker isn't symmetrical either. Such factors can potentially lead to other asymmetries in further along in the system. For example, it might be the case that a penalty taker is able to shoot slightly more accurately to one side, compared to the other.

Moreover, the symmetry of a goalkeeper's area coverage can also be called into question. Humans typically show a preference for one hand over the other, with the majority favouring their right hand. This often translates into an unequal distribution of strength, with the preferred hand or arm exhibiting more strength. If the goalkeeper is stronger in one of the arms, they may be able to stop more shots when using that arm, resulting in a slightly larger area coverage when diving to one of the sides.

Despite the potential for asymmetry in penalty kicks, assuming perfect symmetry still serves as a valuable starting point for analysis. However, when introducing asymmetry into the model, the equilibrium will shift away from the perfectly symmetric solutions we've seen so far. I want to showcase the general direction of these equilibrium shifts, and the logic behind them. To do so, I'm going use the framework to solve a simple asymmetric model. For simplicity, let's restrict the penalty taker to a uniform choice of high velocity ($v = 10$). Let's also say that the standard deviation is 1.9 if the penalty taker aims at a coordinate in the leftmost third of the goal, i.e., where $x < 8$. In the middle third of the goal, when $8 \leq x \leq 16$, the standard deviation is 2, and in the rightmost third of the goal, when $x > 16$ the standard deviation is 2.1. The goalkeeper is allowed all five options for area coverage, "commit left", "lean left", "stay middle", "lean right", and "commit right".

When running a coordinate search with a specificity of 0,2 feet, I get the solution presented in Table 11. This was achieved using 100,000 scenario iterations to simulate payoffs in the first step of the algorithm, then 1,000,0000 iterations in the subsequent steps, followed by 10,000,000 iterations in the final step.

(3.0, 4.8): 0.564	(12.2, 4.6): 0.002	(20.6, 4.2): 0.365
Commit left: 0.545	(12.4, 4.6): 0.069	Lean right: 0.072
		Commit right: 0.383
Expected values: ±0.743		
Exploitability: <0.00003 & <0.00002		

Table 11: The Nash equilibrium solution of a model with asymmetric accuracy. (Rounded to 3 decimals)

When the penalty taker is more accurate when shooting to the left, as opposed to the right, the Nash equilibrium becomes asymmetric. The penalty taker aims significantly more often to the left (0.564) than to the right (0.365). Furthermore, when aiming to the left, the penalty taker aims at a coordinate that's higher, and also closer to the post. This is in line with the results found in the first few models, and a necessary adjustment to take full advantage of the better accuracy when aiming to this side.

The goalkeeper also moves more frequently to the left, selecting "commit left" 54.5% of the time. Notice that this is not enough to deter the penalty taker from aiming more often to their more accurate side. This illustrates what how the equilibrium generally shifts when the penalty taker has one side with favourable assumptions. The penalty taker aims more often to the favourable side, and the goalkeeper covers that side more often.

Now, let's solve another model, where the penalty taker has a standard deviation of 2, that's uniform across all coordinates, but the goalkeeper's area coverage is slightly larger when choosing "commit left" than when choosing "commit right". The right side of the inequality is 6.5 for "commit left" and 6 for "commit right":

$$Commit\ left: \begin{cases} \dfrac{(x+2-0.9v)^2}{3} + \dfrac{(y+0.5-0.25v)^2}{2} \leq 6.5 & ,7 < v \leq 10 \\ \dfrac{(x-3.4)^2}{3} + \dfrac{(y-3.2)^2}{2} \leq 6.5 & ,0 \leq v \leq 7 \end{cases}$$

Apart from these changes, the model is similar to the one before. When running the coordinate search with a specificity of 0.2 feet, I get the solution presented in Table 12.

(3.4, 4.6): 0.419	(12.0, 4.6): 0.002	(20.8, 4.4): 0.011
Commit left: 0.0462	(12.0, 4.8): 0.076	(20.8, 4.6): 0.492
		Lean right: 0.07
	Expected values: ±0.733	Commit right: 0.468
	Exploitability: <0.00004 & <0.00002	

Table 12: The Nash equilibrium solution in a model with asymmetric area coverage. (Rounded to 3 decimals)

In this solution, the penalty taker aims more frequently to the right than to the left. Additionally, the goalkeeper chooses to move to the right 53.8% of the time, either by choosing "commit right" or "lean right". This can be understood through the same lens as before. In the previous model, the penalty taker aimed more frequently to the left, because the left side had the favourable attribute of a better accuracy. In this model, the penalty taker aims more frequently to the right, because the right side has the favourable attribute of a smaller area coverage for the goalkeeper.

Conclusion

In the outset of this study, the primary goal was to analyse penalty kicks in football from a decision modelling standpoint. This implies that the game is treated as a decision problem, where the only focus is on strategic choices, and thereby the strategic interaction between players. The central research problem was to develop a framework which allows us to approximate Nash equilibria in penalty kicks. Beyond the development of the framework, the paper seeks to gradually develop our understanding of the characteristics of a Nash equilibrium when applied to penalty kicks. This objective was achieved by initiating with a simple model and progressively integrating additional layers of complexity. With each element introduced, we observed the subsequent shifts in the equilibrium, and thereby gained a deeper understanding about the nature of the Nash equilibrium in penalty kicks.

To locate the Nash equilibrium in the various models, the counterfactual regret minimization (CFR) algorithm was used. Additionally, I developed an algorithm called coordinate search, which locates optimal aiming coordinates for the penalty taker.

In the first model, the velocity and the standard deviation were uniform across all coordinates. The goalkeeper was limited to three options – "commit left", "stay middle", and "commit right". By using the coordinate search algorithm, I solved this model for a diverse set of assumptions of velocity and standard deviation. In doing so, I uncovered that large areas within the goal were never aimed for, regardless of the combination of velocity and standard deviation. The viable aim-coordinates moved slightly depending on the assumptions, and there was a clear trend to the movement. The less accurately the penalty taker is able to shoot, the optimal aim-coordinates tend to move lower, and also horizontally towards the centre of the goal. For the goalkeeper, all three options were viable in all combinations of assumptions.

In the second model, the options available to the goalkeeper are expanded to include two new options – "lean left" and "lean right". Again, I used the coordinate search algorithm and solved the model for a diverse set of assumptions of velocity and standard deviation. The general trend regarding the positioning of the aim-coordinates were consistent with the findings in the first model. However, the leaning options proved to be quite advantageous, often being preferred instead of the original "stay middle"-option. The leaning options were advantageous because they allowed the goalkeeper a better chance at stopping a shot to one of the sides, while still having a good chance at stopping shots in the middle.

In the third model, the options available to the penalty taker were expanded to include a choice of velocity. In previous models, the velocity-factor was set to a single specific value, and was uniform across all coordinates. In the new model, the penalty taker could choose different a velocity-factor for different coordinates. The solution of this model revealed that if it's optimal to aim for a central coordinate, this should be done using a low velocity. There are two reasons for this: First, the ball takes a longer time to reach the goal, and when the goalkeeper dives to either side (by choosing "commit left" or "commit right"), there's more time for them to move away from the centre of the goal. Secondly, there's a trade-off between accuracy and velocity. The penalty taker is able to shoot more accurately when shooting with a low velocity.

This forth model is the most complex and realistic, and provides the best approximation of a Nash equilibrium in a real-life penalty kick. In this model, I introduce an element of sequential choice, where the penalty taker can "fake the shot", and observe if the goalkeeper starts to move in a specific direction. If the goalkeeper starts to move, the penalty taker can easily score by aiming for the opposite side of the goal. Consequently, "stay middle" is the only option that performs well against "fake the shot". This forces the goalkeeper to "stay middle" quite frequently. In turn, this results in that the penalty taker never aims for a central coordinate. The "lean left"- and "lean right"-options perform poorly when the penalty taker chooses "fake the shot". They also underperform compared to the "commit left"- and "commit right"-options when the penalty taker aims for a side-coordinate. Consequently, the leaning-options aren't being utilized at all.

Finally, I briefly examined some cases where the player-dependent assumptions are asymmetrical. In the first case, the penalty taker is able to shoot more accurately to one of the sides, and in the second case, the goalkeeper is able to cover a slightly larger area on one side of the goal. In the first case, the more favourable side for the penalty taker is the side where they are able to shoot more accurately. In the second case, the more favourable side is where the goalkeeper can cover a smaller area. Both of these cases illustrate the same principle: When the assumptions are asymmetrical, the penalty taker aims more often to their favourable side, and the goalkeeper also selects options that cover that side more often.

A real-life penalty kicks scenario is extremely complex, and any model needs to make significant simplifications. The applicability of the results is therefore inherently contingent on how well the models manages to capture the essential dynamics of a penalty kick. Herein lies the main limitation of the study. It's vital to approach the results with a clear understanding of the simplifications, and an appreciation of the potential gaps between the models and a real penalty kick.

Future research into the nature of the player-dependent assumptions would help to strengthen our confidence in the applicability of the solutions to the models. For instance, an empirical study could investigate average area coverage of professional goalkeepers. This could be done by observing historical penalty kicks, and grouping the goalkeeper actions into different categories (i.e., “commit left” if the goalkeeper already fully committed to diving left before observing the trajectory of the ball, etc.), and monitoring for which hit-coordinates the goalkeeper managed to stop the shot. Such a study should also monitor the relationship between velocity and area coverage.

It would also be helpful to have data on the inaccuracy of penalty takers. This cannot be obtained by looking at historical data, because you would need to know the aim-coordinate of the penalty taker, and not just the hit-coordinate. Therefore, such a study would need to collect data, by having professional players specify their aim-coordinate before shooting, and then observing the inaccuracy of the shot. Such a study should also investigate the relationship between velocity and inaccuracy.

Future research could also explore multiple avenues for improving the underlying framework of the models. For instance, transitioning from the 2-dimensional approach to a 3-dimensional one could offer a more comprehensive representation of the game. Furthermore, researchers could also seek to develop a truly sequential model, where the players have more than one decision point as the penalty taker approaches the ball. Such developments could ultimately lead to an even deeper and more nuanced understanding of the most intense and crucial situation in football – the penalty kick.

References

Bar-Eli, M., Azar, O. H., Ritov, I., Keidar-Levin, Y., & Schein, G. (2007). Action bias among elite soccer goalkeepers: The case of penalty kicks. Journal of Economic Psychology, 28(5), 606–621. https://doi.org/10.1016/j.joep.2006.12.001

Baarslag, T., Kaisers, M., Gerding, E., Jonker, C. M., & Gratch, J. (2017). When will negotiation agents be able to represent us? The challenges and opportunities for autonomous negotiators. In C. Sierra (Ed.), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (pp. 4684-4690). International Joint Conferences on Artificial Intelligence. https://doi.org/10.24963/ijcai.2017/653

Brown, N., & Sandholm, T. (2019). Solving Imperfect-Information Games via Discounted Regret Minimization. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 1829-1836. https://doi.org/10.1609/aaai.v33i01.33011829

Chiappori, P. A., Levitt, S., & Groseclose, T. (2002). Testing Mixed-Strategy Equilibria When Players Are Heterogeneous: The Case of Penalty Kicks in Soccer. *American Economic Review*, *92*(4), 1138–1151. https://doi.org/10.1257/00028280260344678

Dalton, K., Guillon, M., & Naroo, S. A. (2015). An Analysis of Penalty Kicks in Elite Football Post 1997. International Journal of Sports Science & Coaching, 10(5), 815–827. https://doi.org/10.1260/1747-9541.10.5.815

Fédération Internationale de Football Association. (2023). Laws of the Game 2023/2024. FIFA. Retrieved from https://digitalhub.fifa.com/m/50518593a0941079/original/khhloe2xoigyna8juxw3-pdf.pdf

Jordet, G., Hartman, E., Visscher, C., & Lemmink, K. A. P. M. (2007). Kicks from the penalty mark in soccer: The roles of stress,skill, and fatigue for kick outcomes. *Journal of Sports Sciences*, *25*(2), 121–129. https://doi.org/10.1080/02640410600624020

Jordet, G., Hartman, E., & Sigmundstad, E. (2009). Temporal links to performing under pressure in international soccer penalty shootouts. *Psychology of Sport and Exercise*, *10*(6), 621–627. https://doi.org/10.1016/j.psychsport.2009.03.004

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., & Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, *356*(6337), 508–513. https://doi.org/10.1126/science.aam6960

MyFootballFacts. (n.d.). Premier League Penalty Statistics. MyFootballFacts. Retrieved from https://www.myfootballfacts.com/premier-league/all-time-premier-league/premier-league-penalty-statistics/

Nash, J. (1951). Non-Cooperative Games. *The Annals of Mathematics*, *54*(2), 286. https://doi.org/10.2307/1969529

Rodrigues-Neto, J. (2014). Game Theory, An Introduction, by Steven Tadelis (Princeton University Press, Princeton, NJ, 2013), pp. xv + 396. *Economic Record*, *90*(291), 551–552. https://doi.org/10.1111/1475-4932.12156

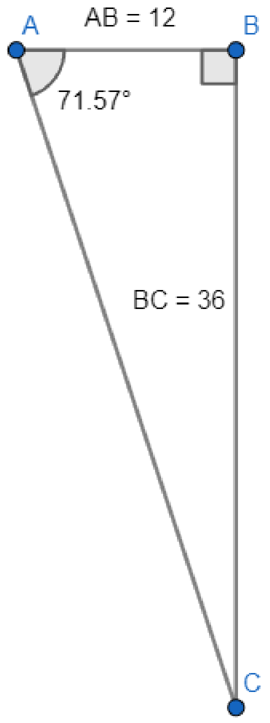
Transfermarkt. (n.d.). Premier League Penalty Statistics. Transfermarkt. Retrieved from https://www.transfermarkt.co.uk/premier-league/topErhalteneElfmeter/wettbewerb/GB1

Visscher, C., Elferink-Gemser, M. T., & Lemmink, K. A. P. M. (2006). Interval Endurance Capacity of Talented Youth Soccer Players. *Perceptual and Motor Skills*, *102*(1), 81–86. https://doi.org/10.2466/pms.102.1.81-86

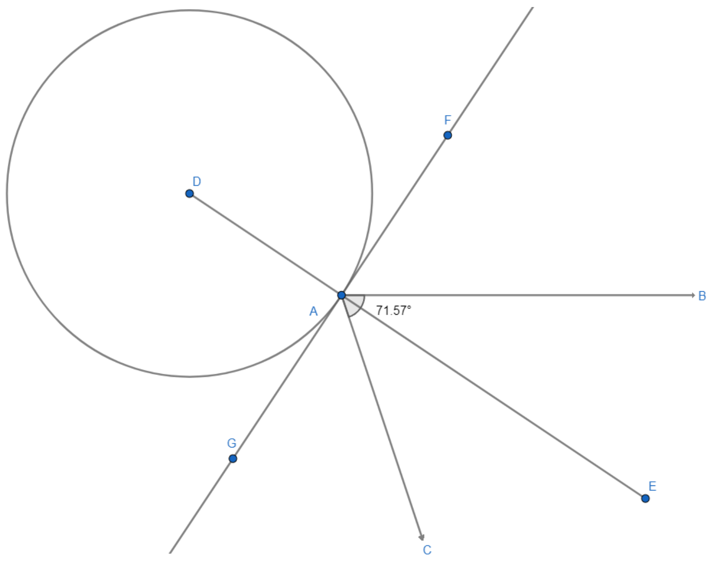
Zinkevich, M., Johanson, M., Bowling, M., & Piccione, C. (2007). Regret Minimization in Games with Incomplete Information. In Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS 2007). Retrieved from https://papers.nips.cc/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf

Appendix I

This appendix presents the calculation that derives a good approximation the domain of hit-coordinates that can result in a goal. The penalty taker shoots from a distance of 11 meters, or approximately 36 feet. In the figure on the right, point C represents the 11-meter mark, point B represents to the centre of the goal, and point A represents the point of contact when a shot hits the left post. The distance between the centre of the goal and the left post is 12 feet. The angle $\angle CAB$ is then given by $\tan^{-1}(36/12) \approx 71.57°$. The AB distance of 12 is technically slightly off, since the point of contact won't be exactly on the edge of the post, but this is a sufficient approximation, as it will hardly change the angle at all.



In the figure below, we zoom in on point A, and add a circle with point D as its centre. The circle represents the left post. The line FG is the tangent line at the point of contact. The angle $\angle CAE$ represents the angle that the ball enters the point of contact. Since the ball will leave the point of contact at the same angle that it entered, $\angle CAE = \angle EAB$. If the sum of these angles is greater than $71.57°$, the ball will bounce into the goal.



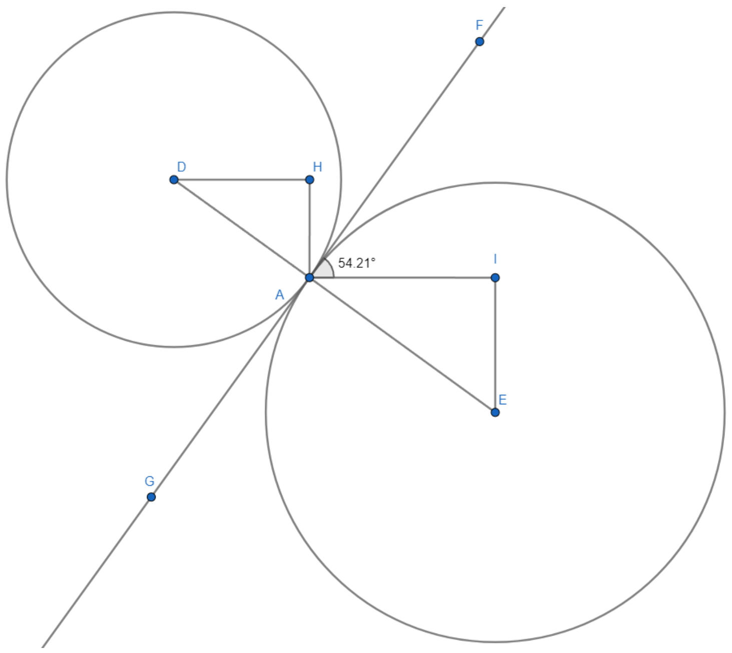
It's apparent that $\angle BAF = 90° - \angle EAB$. I want to find the angle $\angle BAF$ that is such that $\angle CAE + \angle EAB \approx 71.57°$.

That's the case if both angles are half of $71.57°$, meaning that $\angle CAE = \angle EAB \approx 35.79°$.

Therefore, the critical angle is:

$\angle BAF = 90° - 35.79° \approx 54.21°$

At the point of contact, both the ball and the left post will have the same tangent line FG. By giving the tangent line a set angle of $54.21°$ to the goal line, I can find the x-coordinate that is such that the ball bounces in the exact direction of the goal line. The x-coordinates to the right of this will then result in a goal, if they are to the left of the right post and below the crossbar.



In the figure on the right, the circle with a centre of point E represents the ball. Point D is located at a negative x-coordinate equal to that of the radius of the post. The post has a diameter of 0.417 feet, so point D is located at x-coordinate -0.2085. What I need to find is the x-coordinate of point E; the centre of the ball. Since I have the location of point D, I can find the x-coordinate of point E by adding the distances DH and AI.

By normalizing the radius of the left post to 1, we can find the proportional distance of DH to the radius of the circle. Since $\angle IAH = 90°$, $\angle FAH = 90° - 54.21°$, and $\angle FAD = 90°$, the angle $\angle DAH$ must also equal $54.21°$. Since the distance AD is normalized to 1, the distance DH is given by $\sin(54.21°) \approx 0.811$. Similarly, the distance AI will have the same proportion to the radius of the ball, which is 0.365. The critical x-coordinate by the left post is then given by $-0.2085 + (0.2085 + 0.365) * 0.811 \approx 0.257$. On the other side of the goal, the ball needs the same margin, resulting in a critical x-coordinate by the right post of $24 - 0.257 = 23.743$. At these x-coordinates, the ball will leave the post and follow a line perpendicular to the goal line, whereas x-coordinates in between will result in a goal. This gives us the domain of x-values that can result in a goal:

$$x\epsilon\langle 0.257, 23.743\rangle$$

For the y-coordinate, the calculation is exactly the same. The only difference is that the distance from the centre of the goal to the crossbar is 8 feet, instead of 12. This gives us a slightly more open angle to the 11-meter mark of approximately $77.36°$, reducing the required margin slightly. Since only the y-values above or equal to 0.365 are feasible, the following domain can result in a goal:

$$y\epsilon[0.365, 7.751\rangle$$

Appendix II

This appendix covers the essential parts of the code used for simulating the payoffs in each possible outcome of the game. See github.com/Monstad/Penalty-Kicks for the complete code.

```
AIM_OPTIONS = []
x_values = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]
y_values = [-2, 0, 2, 4, 6]
for x_value in x_values:
    for y_value in y_values:
        AIM_OPTIONS.append((x_value, y_value))

AREA_EQUATIONS = {
    'commit_left': lambda x, y, v: ((x + 2 - 0.9*v)**2)/3 + ((y + 0.5 - 0.25*v)**2)/2 <= 6,
    'stay_middle': lambda x, y, v: ((x - 12)**2)/8 + ((y - 2 - 0.2*v)**2)/8 <= 22.5 - 1.85*v,
    'commit_right': lambda x, y, v: ((x - 26 + 0.9*v)**2)/3 + ((y + 0.5 - 0.25*v)**2)/2 <= 6,
}
AREA_OPTIONS = list(AREA_EQUATIONS.keys())
```

AIM_OPTIONS defines the strategic options for the penalty taker, and *AREA_EQUATIONS* and *AREA_OPTIONS* define the strategic options for the goalkeeper. The v-value in the *AREA_EQUATIONS* is the velocity-factor. It defines the size of the area the goalkeeper is able to cover. A larger velocity factor results in a smaller area, and vice versa.

In this example, the penalty taker has 55 different coordinates to aim for, and the goalkeeper has three different areas they can cover. This means that there are 165 different combinations of choices that can occur. I'm going to simulate the expected values (the chance of a goal occurring) in each of these scenarios. The process is exactly the same if there's a different number of scenarios.

```
def getHitCoordinate(aim_coordinate, sd, velocity):
    hit_coordinate = np.random.normal(aim_coordinate, sd, size=2)
    hit_velocity = velocity
    if hit_coordinate[1] < LOWER_BOUND:
        hit_velocity = velocity - (LOWER_BOUND - hit_coordinate[1]) * FRICTION
        hit_velocity = max(hit_velocity, 0)
    hit_coordinate[1] = max(LOWER_BOUND, hit_coordinate[1])
    return hit_coordinate, hit_velocity
```

The *hit_coordinate* is drawn from a bivariate normal distribution. The mean of the distribution is given by the *aim_coordinate*. The standard deviation, which is equal in both the vertical and horizontal direction, is given by *sd*. The *LOWER_BOUND* represents the smallest possible y-value for the *hit_coordinate*. This is the y-value that's such that the ball touches the ground when it crosses the goal line. If the *hit_coordinate* that's drawn is below this threshold, the ball will make contact with the ground during at least some portion of its trajectory towards the goal. This leads to an increase in friction, reducing the velocity factor, and consequently expanding the area that the goalkeeper is able to cover. As the drawn *hit_coordinate* moves further below the threshold, the ball will be in contact with the ground for longer, resulting in even more friction and larger goalkeeper areas. For instance, if the *FRICTION* factor is 0.1 and the *hit_coordinate* is one unit below the *LOWER_BOUND*, the *hit_velocity* will be reduced by 0.1. For a *hit_coordinate* two units below, the *hit_velocity* will decrease by 0.2, and so on. The velocity factor has a domain of $v \epsilon [0, 10]$, so I use the max-function to ensure that the value doesn't drop below zero. Similarly, if the y-value of the *hit_coordinate* is drawn to be below the *LOWER_BOUND*, the max-function sets it equal to the *LOWER_BOUND*.

```
def isInsideScoringArea(hit_coordinate):
    return LEFT_BOUND < hit_coordinate[0] < RIGHT_BOUND and hit_coordinate[1] < UPPER_BOUND
```

The *isInsideScoringArea()*-function returns *True* if the *hit_coordinate* is within the scoring area, and *False* otherwise. The boundaries of the scoring area is defined by the *LEFT_BOUND*, *RIGHT_BOUND*, and *UPPER_BOUND*, as was derived in Appendix I. Since the *hit_coordinate* cannot be below the *LOWER_BOUND*, it's not necessary to include that in the condition.

```
def isSavedShot(hit_coordinate, hit_velocity, area):
    equation = AREA_EQUATIONS.get(area)
    return equation(hit_coordinate[0], hit_coordinate[1], hit_velocity)
```

The *isSavedShot()*-function returns *True* if the *hit_coordinate* is within the area the goalkeeper has decided to cover, and *False* otherwise. By combining the *isSavedShot()*-function with the *isInsideScoringArea()*-function, I can check whether or not a goal has been achieved.

```
def getScenarioEV(iterations, aim_coordinate, area, velocity, std_dev):
    goal = 0
    for _ in range(iterations):
        hit_coord, hit_velocity = getHitCoordinate(aim_coordinate, std_dev, velocity)
        if isInsideScoringArea(hit_coord) and not isSavedShot(hit_coord, hit_velocity, area):
            goal += 1
    return goal/iterations
```

The *getScenarioEV()*-function simulates penalty kicks where the penalty taker aims for a specific *aim_coordinate*, and the goalkeeper covers a specific *area*. By counting the number of goals and dividing by the number of iterations, I get a good approximation of the probability of a goal, given those choices. The probability of a goal is the expected value for the penalty taker, whereas the expected value of the goalkeeper is equal to the negative probability of a goal.

```
def getScenarioData(iterations, aim_options, area_options, velocity, std_dev):
    scenario_data = {}
    for i in range(len(aim_options)):
        for j in range(len(area_options)):
            ev = getScenarioEV(iterations, aim_options[i], area_options[j], velocity, std_dev)
            scenario_data[(aim_options[i], area_options[j])] = ev
    if SYMMETRIC:
        scenario_data = makeScenariosSymmetric(scenario_data, aim_options, area_options)
    return scenario_data
```

The *getSenarioData()*-function employs the *getScenarioEV()*-function to estimate the probability of a goal in all the 165 different scenarios. The resulting data is then saved in a dictionary, where the keys consist of tuples containing all possible combinations of *aim_options* and *area_options*, and the corresponding values are the probabilities of a goal in each scenario.

If all assumptions in the framework are symmetric across the line $x = 12$, the global *SYMMETRIC*-variable should be set to *True*, and otherwise it should be set to *False*. If it's set to *True*, the *makeScenariosSymmetric()*-function will transform the data in such a way that the symmetric scenarios have equal expected values. For instance, assuming perfect symmetry, the scenario where the penalty taker aims for coordinate $(4,4)$ and the goalkeeper chooses *'commit_left'*, is equivalent to the scenario where the penalty taker aims for $(20,4)$ and the goalkeeper chooses *'commit_right'*. In this instance, the *scenario_data* is transformed such that both these scenarios are assigned the average expected value of the two scenarios.

Appendix III

This appendix covers the essential parts of the code relating to the counterfactual regret minimization (CFR) algorithm. See github.com/Monstad/Penalty-Kicks for the complete code.

```
def getRegret(aim_coordinate, area, scenario_data, aim_options, area_options):
    num_options_pt = len(aim_options)
    num_options_gk = len(area_options)
    regret_pt = [0] * num_options_pt
    regret_gk = [0] * num_options_gk
    exp_value_pt = scenario_data[(aim_coordinate, area)]
    exp_value_gk = -exp_value_pt
    for i in range(num_options_pt):
        alt_exp_value_pt = scenario_data[(aim_options[i], area)]
        regret_pt[i] = alt_exp_value_pt - exp_value_pt
    for i in range(num_options_gk):
        alt_exp_value_gk = -scenario_data[(aim_coordinate, area_options[i])]
        regret_gk[i] = alt_exp_value_gk - exp_value_gk
    return regret_pt, regret_gk
```

The *getRegret()*-function calculates all regret-values for both players, given their specific choices of *aim_coordinate* and *area*. The regret for an option is the difference between the expected value that would have been achieved had that option been chosen (*alt_exp_value_pt* or *alt_exp_value_gk*), and the expected value that was achieved with the option that actually was chosen (*exp_value_pt* or *exp_value_gk*), assuming that the opponent's choice remains constant. The regret is positive if the alternative option would have performed better, and negative if it would have performed worse.

```
def getStrategy(regret_sum, strategy_sum, num_options, discount_rate=0):
    strategy = [0] * num_options
    normalizing_sum = 0
    for i in range(num_options):
        if regret_sum[i] > 0:
            strategy[i] = regret_sum[i]
            normalizing_sum += strategy[i]
    for i in range(num_options):
        if normalizing_sum > 0:
            strategy[i] = strategy[i] / normalizing_sum
        else:
            strategy[i] = 1.0 / num_options
        strategy_sum[i] = strategy_sum[i] * (1-discount_rate) + strategy[i]
    return strategy, strategy_sum
```

The *getStrategy()*-function calculates what *strategy* should be used for the next iteration of the CFR-algorithm. It also updates the *strategy_sum*, which is the sum of all strategies used throughout all iterations in the CFR-algorithm. The principle for deciding the next strategy is regret matching. The *regret_sum* keeps track of the sum of regrets throughout all iterations for all options, and is used to perform the regret matching. All options that have a positive *regret_sum* get assigned their *regret_sum* as their *strategy*. Later, the *normalizing_sum* is used to normalize the strategy such that it has a sum of one. The *strategy* is then added to the *strategy_sum*. There's an option to include a *discount_rate* which effectively makes the early iterations of the CFR-algorithm count less towards the *strategy_sum* than the later iterations. Using a discount rate can lead to faster convergence towards an approximate equilibrium, but the approximation may not be as precise as without it.

```
def train(max_exploitability, scenario_data, aim_options, area_options, discount_rate=0):
    num_options_pt = len(aim_options)
    num_options_gk = len(area_options)
    strategy_pt = [1 / num_options_pt] * num_options_pt
    strategy_gk = [1 / num_options_gk] * num_options_gk
    regret_sum_pt = [0] * num_options_pt
    regret_sum_gk = [0] * num_options_gk
    strategy_sum_pt = [0] * num_options_pt
    strategy_sum_gk = [0] * num_options_gk
    optimal_strategy_pt = [0] * num_options_pt
    optimal_strategy_gk = [0] * num_options_gk
    exploitability_pt = 1
    exploitability_gk = 1
    i = 1

    while exploitability_pt > max_exploitability or exploitability_gk > max_exploitability:
        aim = getAction(aim_options, strategy_pt)
        area = getAction(area_options, strategy_gk)

        for j in range(num_options_pt):
            regret_sum_pt[j] += \
                getRegret(aim, area, scenario_data, aim_options, area_options)[0][j]

        for j in range(num_options_gk):
            regret_sum_gk[j] += \
                getRegret(aim, area, scenario_data, aim_options, area_options)[1][j]

        strategy_pt, strategy_sum_pt = \
            getStrategy(regret_sum_pt, strategy_sum_pt, num_options_pt, discount_rate)
        strategy_gk, strategy_sum_gk = \
            getStrategy(regret_sum_gk, strategy_sum_gk, num_options_gk, discount_rate)

        if i % 10000 == 0:
            normalizing_sum_pt = sum(strategy_sum_pt)
            normalizing_sum_gk = sum(strategy_sum_gk)

            for j in range(num_options_pt):
                optimal_strategy_pt[j] = strategy_sum_pt[j] / normalizing_sum_pt

            for j in range(num_options_gk):
                optimal_strategy_gk[j] = strategy_sum_gk[j] / normalizing_sum_gk

            if SYMMETRIC:
                optimal_strategy_pt = makeStrategySymmetricPT(optimal_strategy_pt, aim_options)
                optimal_strategy_gk = makeStrategySymmetricGK(optimal_strategy_gk, area_options)

            _, _, exploitability_pt, exploitability_gk = \
                getExploitability(scenario_data,
                                  aim_options,
                                  area_options,
                                  optimal_strategy_pt,
                                  optimal_strategy_gk)
         i += 1

    return optimal_strategy_pt, optimal_strategy_gk
```

The *train()*-function approximates Nash equilibrium strategies for both the penalty taker and the goalkeeper. The training is done within a *while*-loop, which keeps iterating until the exploitability of the trained strategies are below the threshold defined by *max_exploitability*.

In the first iteration of the training process, both players start out with uniform strategies. Actions are randomly selected for both players, in accordance with their respective strategies. This is done using the *getAction()*-function:

```
def getAction(options, strategy):
    return options[np.random.choice(len(options), p=strategy)]
```

Given the selected actions, the *getRegret()*-function is used to update the regret sums for both players (*regret_sum_pt* and *regret_sum_gk*). The regret sums are then used to update the strategies of the players, in accordance with the principle of regret matching. In this step, it's possible to include a discount rate, putting less weight on earlier iterations when updating the strategies. Once strategies have been updated, new actions are drawn according to those strategies, and the same steps are repeated again and again.

For every 10,000th iteration, the Nash equilibrium strategies for both players (*optimal_strategy_pt* and *optimal_strategy_gk*) are estimated. This is the average strategies used throughout all iterations. To find the strategies, I divide the *strategy_sum_pt* by the *normalizing_sum_pt*, and the *strategy_sum_gk* by the *normalizing_sum_gk*. Both players will have the same normalizing sums unless different discount rates are used for the players.

When all assumptions in the model are perfectly symmetrical across the line $x = 12$, the global *SYMMETRIC*-variable should be set to *True*. When this is the case, the estimated strategies are updated to take on perfectly symmetric frequencies. This implies that the goalkeeper always chooses "commit_left" equally often as "commit_right", and that the penalty taker always chooses to aim for symmetrical coordinates equally often (such as the coordinates (4, 4) and (20, 4)). The symmetrical options are assigned frequencies equal to the average frequencies of both the options.

Once this is done, the exploitability of the strategies is calculated. This is done using the *getExploitability()*-function. If the exploitability of either strategy, *exploitability_pt* or *exploitability_gk,* is above the *max_exploitability* threshold, the while-loop keeps running. Once the acceptable exploitability is reached, the function returns the estimated strategies, *optimal_strategy_pt* and *optimal_strategy_gk*, and the counterfactual regret minimization process is complete.

Since exploitability is the metric which determines when the solution is considered satisfactory, I'm going to explain how it's calculated. At the essence of the calculation is a comparison of expected values.

```
def getStrategyEVs(scenario_data, aim_options, area_options, strategy_pt, strategy_gk):
    exp_value_pt = 0
    for i in range(len(aim_options)):
        for j in range(len(area_options)):
            exp_value_pt += \
                scenario_data[(aim_options[i], area_options[j])] *strategy_pt[i] *strategy_gk[j]
    exp_value_gk = -exp_value_pt
    return exp_value_pt, exp_value_gk
```

The *getStrategyEVs()*-function calculates the expected value of a pair of strategies. Remember that a strategy is only a list of probabilities indicating how often each option is being selected. When we know the strategies of both players, we know the probability of each possible outcome of the game. The *scenario_data* dictionary holds complete information about payoffs in all possible outcomes. By multiplying the probabilities with the corresponding payoffs we can find the expected value for both player's strategies.

```
def getExploitability(scenario_data, aim_options, area_options, strategy_pt, strategy_gk):
    num_options_pt = len(aim_options)
    num_options_gk = len(area_options)
    exp_value_pt, exp_value_gk = \
        getStrategyEVs(scenario_data, aim_options, area_options, strategy_pt, strategy_gk)
    best_response_ev_pt = 0
    best_response_ev_gk = -1

    for i in range(num_options_pt):
        pure_strategy_pt = [0] * num_options_pt
        pure_strategy_pt[i] = 1
        option_ev = getStrategyEVs(scenario_data,
                                   aim_options,
                                   area_options,
                                   pure_strategy_pt,
                                   strategy_gk)[0]
        if option_ev > best_response_ev_pt:
            best_response_ev_pt = option_ev

    for i in range(num_options_gk):
        pure_strategy_gk = [0] * num_options_gk
        pure_strategy_gk[i] = 1
        option_ev = getStrategyEVs(scenario_data,
                                   aim_options,
                                   area_options,
                                   strategy_pt,
                                   pure_strategy_gk)[1]
        if option_ev > best_response_ev_gk:
            best_response_ev_gk = option_ev

    exploitability_pt = best_response_ev_gk - exp_value_gk
    exploitability_gk = best_response_ev_pt - exp_value_pt

    return exp_value_pt, exp_value_gk, exploitability_pt, exploitability_gk
```

The *getExploitability()*-function calculates how much expected value the opponent can gain by deviating from their current strategy, and adopting the best response strategy. The best response strategy is found by testing all possible pure strategies, and finding the one that performs the best. The expected value of the best pure strategy is then assigned to the variables *best_response_ev_pt* and *best_response_ev_gk*. To find the exploitability for both players these values are then compared with the expected values of the current strategies.

Appendix IV

This appendix covers the essential parts of the code relating to the coordinate search algorithm. The complete code can be found at github.com/Monstad/Penalty-Kicks. The algorithm performs an iterative process that locate optimal coordinates. This is done by finding solutions using the CFR-algorithm, and then updating the available aiming-coordinate at every step.

```
INITIAL_AIM_OPTIONS = []
x_values = [0.8, 2.4, 4.0, 5.6, 7.2, 8.8, 10.4, 12.0, 13.6, 15.2, 16.8, 18.4, 20.0, 21.6, 23.2]
y_values = [-0.8, 0.8, 2.4, 4.0, 5.6, 7.2]
for x_value in x_values:
    for y_value in y_values:
        INITIAL_AIM_OPTIONS.append((x_value, y_value))
```

The *INITIAL_AIM_OPTIONS* are initialized such that the penalty taker can aim at coordinates across the goal. The coordinates are evenly spaced out with a distance between the points of 1.6 feet, and the distance from the outermost points to the posts or the bar is 0.8 feet. Throughout the steps of the coordinate search algorithm, these distances will be halved several times, until an acceptable degree of specificity is reached.

After applying the CFR-algorithm to the *INITIAL_AIM_OPTIONS*, we get a Nash equilibrium solution given those options. It's then time to determine which of the aiming-coordinates should be kept for the next iteration of the coordinate search.

```
def getFrequentAimOptions(strategy, aim_options):
    frequent_aim_options = []
    for i in range(len(strategy)):
        if strategy[i] >= 0.0005:
            frequent_aim_options.append(aim_options[i])
    return frequent_aim_options
```

The *getFrequentAimOptions()*-function finds the aiming-options that are chosen more than 0.005% of the time in a *strategy*. When applied to the optimal strategy for the penalty taker (as given by the CFR-algorithm), it gives us the list of aim-coordinates that should be kept.

In the first step of the coordinate search, the only viable options will be the frequent aim-options, because these are the only options that have ever existed. However, after the first step, some *aim_options* get discarded. Even though these aim_options weren't optimal when solving for the set of *INITIAL_AIM_OPTIONS*, they may become optimal when solving for a different set of options. Therefore, at every step, the previously discarded options should be reconsidered. The *getOtherViableOptions()*-function checks if any of the old options would have outperformed the available options, given the goalkeepers optimal strategy against the available options. This is possible because the *scenario_data* for the discarded scenarios is still being stored.

```python
def getOtherViableOptions(scenario_data, aim_options, area_options, strategy_gk, exp_value_pt):
    other_viable_options = []
    for (aim_option, area_option) in scenario_data.keys():
        pure_strategy_pt = [0] * len(aim_options)
        pure_strategy_pt[aim_options.index(aim_option)] = 1
        exp_value_option, _ = getStrategyEVs(scenario_data,
                                             aim_options,
                                             area_options,
                                             pure_strategy_pt,
                                             strategy_gk)
        if exp_value_option >= exp_value_pt:
            other_viable_options.append(aim_option)
    return other_viable_options
```

The function compares the expected value of playing a pure strategy of all previous aiming-coordinates with the expected value of the penalty taker's strategy. It then returns all the coordinates which yielded a higher expected value. This can then be used to add some discarded *aim_options* back in as an available option. Now, let's consider how to add new coordinates that surround the options given by *getFrequentAimOptions()* and *getOtherViableOptions()*.

```python
def getDistanceToNearestCoordinate(aim_coordinate, aim_options):
    nearest_distance = 1.6
    distance = 1.6
    for option in aim_options:
        for i in range(2):
            if option[i] != aim_coordinate[i]:
                distance = round(abs(option[i] - aim_coordinate[i]), 1)
            if nearest_distance > distance:
                nearest_distance = distance
    return nearest_distance
```

The *getDistanceToNearestCoordinate()*-function finds the shortest horizontal or vertical distance to a nearby coordinate. This is useful, because when adding new aiming-coordinates, we need to know how close the coordinates should be to a previous coordinate.

```python
def getNewAimOptions(aim_coord, distance, aim_options):
    values = [-distance, 0, distance]
    is_surrounded = False
    surrounding_coords = \
        [(round(aim_coord[0] -i, 1), round(aim_coord[1] -j, 1)) for i in values for j in values]
    if set(surrounding_coords).issubset(set(aim_options)):
        is_surrounded = True
        if distance > SPECIFICITY:
            values = [-distance/2, 0, distance/2]
    new_aim_options = \
        [(round(aim_coord[0] -i, 1), round(aim_coord[1] -j, 1)) for i in values for j in values]
    new_aim_options = list(set(new_aim_options))
    new_aim_options.sort()
    return new_aim_options, is_surrounded
```

The *getNewAimOptions()*-function finds new aiming-coordinates that surround a specific *aim_coord*. The function checks if the *aim_coord* is surrounded by other coordinates in *aim_options*, spaced out by the given *distance*. If this is the case, it creates a set of *new_aim_options* that surround the *aim_coord* but are half the *distance* away. If it's not the case, it creates *new_aim_options* that are the full *distance* away. Additionally, the function returns a Boolean value, *is_surrounded*, which is used to check if all the coordinates have converged given the desired specificity of the coordinate search.

By combining these functions, we get the general structure of the coordinate search algorithm (although it has been simplified slightly):

```
allAimOptions = INITIAL_AIM_OPTIONS
allNewAimOptions = INITIAL_AIM_OPTIONS

scenarioData = getScenarioData(SCENARIO_ITERATIONS,
                    INITIAL_AIM_OPTIONS,
                    AREA_OPTIONS,
                    VELOCITY,
                    STD_DEV)

while True:
    optimalStrategyPT, optimalStrategyGK = \
        trainCoordinateSearch(scenarioData, allNewAimOptions, AREA_OPTIONS, DISCOUNT_RATE)

    expValuePT, expValueGK, exploitabilityPT, exploitabilityGK = \
        getExploitability(scenarioData,
                          allNewAimOptions,
                          AREA_OPTIONS,
                          optimalStrategyPT,
                          optimalStrategyGK)

    frequentAimOptions = getFrequentAimOptions(optimalStrategyPT, allNewAimOptions)

    otherViableOptions = getOtherViableOptions(scenarioData,
                                              allAimOptions,
                                              AREA_OPTIONS,
                                              optimalStrategyGK,
                                              expValuePT)

    viableOptions = list(set(frequentAimOptions + otherViableOptions))

    allNewAimOptions = []
    num_converged = 0

    for option in viableOptions:
        distance = getDistanceToNearestCoordinate(option, allAimOptions)
        newAimOptions, isSurrounded = getNewAimOptions(option, distance, allAimOptions)
        allNewAimOptions.extend(newAimOptions)

        if (option in frequentAimOptions) and isSurrounded and distance == SPECIFICITY:
            num_converged += 1

    allNewAimOptions = list(set(allNewAimOptions))
    allAimOptions.extend(allNewAimOptions)

    if len(viableOptions) == len(frequentAimOptions) == num_converged:
        print('Completed the option search.')
        break

    scenarioData = getScenarioData(
            SCENARIO_ITERATIONS,
            allNewAimOptions,
            AREA_OPTIONS,
            VELOCITY,
            STD_DEV,
            existing_data=scenarioData,
            replace=False)
```

The first step is to simulate the *scenarioData* for the *INITIAL_AIM_OPTIONS*. Using this *scenarioData*, the *trainCoordinateSearch()*-function then finds the optimal strategies for both players. The *trainCoordinateSearch()*-function is just a CFR-algorithm except it doesn't use exploitability as a stopping-criteria, but rather stops solving once there are either no options in either players strategy that gets selected at a low frequency, or when a certain number of iterations have been reached. The solutions found by the CFR-algorithm, i.e., the *optimalStrategyPT* and *optimalStrategyGK*, are then used with the *getFrequentAimOptions()*- and *getOtherViableOptions()*-functions to locate the viable aiming options. For each viable option, the *getDistanceToNearestCoordinate()*- and *getNewAimOptions()*-functions are used to find new aiming-coordinates that surround the viable options. This gives us a list of options, *allNewAimOptions*, bringing about new scenarios that need to be simulated. Using the *getScenarioData()*-function, and keeping the old *scenarioData* as *existing_data*, the *scenarioData* is updated to include the new scenarios. Once this is complete, the first iteration of the while-loop is complete, and we enter the second step of the coordinate search.

The coordinate search continues until all the coordinates in *frequentAimOptions* are surrounded by coordinates distanced by the desired specificity (for example 0.2 feet). Additionally, all the viable options must be *frequentAimOptions*, as opposed to *otherViableOptions*. Once this is complete, the *frequentAimOptions* will contain the optimal aiming-coordinates for the penalty taker, and we break out of the while-loop. When there are only a few scenarios left, the CFR-algorithm converges to a very low exploitability. We can re-simulate the scenario payoffs using extra many iterations, and run the algorithm again for a final solution:

```
scenarioData = getScenarioData(FINAL_SCENARIO_ITERATIONS,
                               frequentAimOptions,
                               AREA_OPTIONS,
                               existing_data=scenarioData,
                               replace=True)

optimalStrategyPT, optimalStrategyGK = \
    train(MAX_EXPLOITABILITY, scenarioData, frequentAimOptions, AREA_OPTIONS)

expValuePT, expValueGK, exploitabilityPT, exploitabilityGK = \
    getExploitability(scenarioData,
                      frequentAimOptions,
                      AREA_OPTIONS,
                      optimalStrategyPT,
                      optimalStrategyGK)

showStrategies(frequentAimOptions,
               AREA_OPTIONS,
               optimalStrategyPT,
               optimalStrategyGK,
               expValuePT,
               expValueGK,
               exploitabilityPT,
               exploitabilityGK)
```