Numerical Solutions to Principal-Agent Problems

Nam Dang

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Numerical Solutions to Principal-Agent Problems

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft University of Technology

Nam Dang

April 6, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

A principal-agent problem is a mathematical framework for modelling contractual relationships, where a principal delegates tasks to an agent in exchange for reward. Many real life contracts are of this type. The agent does not have the same objectives as the principal and he possesses hidden information. Two types of hidden information is generally considered: moral hazard (hidden action) and adverse selection (hidden type). This asymmetry of information can be exploited by the agent for his own benefits, which can reduce the principal's utility. The goal is find a mechanism, in the form of a contract, that aligns the principal's and agent's objectives and maximises the principal's utility.

Previous economic studies focuses on finding analytical solutions on specific cases. In this study we consider numerical solution approaches for a principal agent-problem with moral hazard, and a generalised principal-agent problem with moral hazard and adverse selection problem. The moral hazard problem is a bi-level programming problem and it is solved by assuming the agent's optimal effort is an implicit function dependent on the contract.

The generalised principal agent problem is an infinite programming problem. For that we propose two numerical solution methods: the discretisation method and the basis function method. Both are tested on a option contract problem with moral hazard and adverse selection. From the experiment, the discretisation method seems to be the better performer.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many contract design problems, e.g. in insurance, employment and outsourcing agreements, can be modelled as a principal-agent problem, The parties that are involved in the contract are: the principal, referred to as "she", and the agent, referred to as "him". The principal is the contractee, who delegates tasks to the agent. She offers a contract to the agent who may or may not agree on it. If the agent accepts the contract, then he performs actions that generates a payoff for both parties. The contract induces a game where the agent is the player, who act in such a way that maximises his own pay off. The goal of every principal-agent problem is to design an optimal contract that maximises the principals utility. Before going further with principal-agent problems, we explain some relevant game theoretic concepts. For a more in-depth introduction into game theory, we refer the reader to [1].

1-1 Game Theory

Game theory is a formal study of conflict and cooperation. A game is a model of a situation where players can interact, which results in an outcome. Players are parties that can make decisions in a game and each player has their own preferences regarding the outcome of the game. A common assumption is that each player is *rational*. A rational player makes decisions such that he gets the best possible outcome according to his preferences, while considering other player's decisions. Players have preferences between possible outcomes of the game. How much a player values a certain outcome is quantified with the *utility* or *payoff*. The general convention is that the higher the utility, the more a player prefers the outcome. Examples of utilities are the amount of chips in a poker game or the number of goals in a football match. With the moves of all other players fixed, we say a player's move is a *best response*, if he cannot improve the outcome by changing his move. If there is a situation where all players have decided on a move and all theses moves are best responses, then we have a *Nash equilibrium*. This concept is first introduced by John Nash [2]. To clarify these concepts further, we give the following example of a game:

Example 1 (Chicken)**.** *Consider two drivers A and B going straight towards each other. If none of them changes course, then they will end up in a head on collision. If both of them want to survive,*

then at least on of them have to swerve. If one driver swerves, then the other one would prefer driving straight. We have the following possible outcomes and corresponding utilities:

1. *If they both swerve, then we have a tie and both drivers gain* 0 *utility.*
2. *If one swerves and the other goes straight, then the swerver gets* -1 *utility and the driver who drives straight gets* $+1$ *utility.*
3. *If both drivers remain their course, then they both crash and they receive* -10 *utility.*

The outcomes and utilities are shown in the payoff matrix in Table 1-1. When B swerves, then A's best response is to go straight. If B goes straight, then A's best response is to swerve. Similarly, if A swerves, then B should go straight, and if A goes straight, then B should swerve. We can see that the Nash equilibria are (swerve, straigt) and (straight, swerve). It is possible to have multiple Nash equilibria.

Table 1-1: Payoff matrix for the Chicken game. The utility pair (x, y) means that A receives x utility and B gets y utility.

		Player B	
		Swerve	Straight
Player A	Swerve	$(0,0)$	$(-1,+1)$
	Straight	$(+1,-1)$	$(-10,-10)$

1-2 Hidden Information

In a principal-agent problem, we want to find a mechanism that maximises the principal's (expected) utility, while assuming that the agent's set of actions in the induced game is a Nash equilibrium. Principal-agent problems are characterized by the misalignment of the principal's and agent's objectives and the asymmetry of information. The agent possesses information that is hidden to the principal. Two types of hidden information are considered: the first one is the private *type* of the agent before a contract is proposed and the second one is the inability to verify the agent's actions after the contract is signed. The first kind of hidden information is referred to as *adverse selection* and the latter as *moral hazard*. The agent can exploit the information advantage that he has over the principal to gain more utility.

In case of adverse selection, the agent's type is not known to the principal. The agent's type is any kind of relevant hidden information the agent possess, before a contract is signed. Take for example the situation where the principal is a health insurance company and she does not know the medical history of a client (agent). The patients's history will be his type. Another example is when a company (principal) hires a contractor (agent) to perform a certain job. The hiring company does not know the true costs of the contractor. The contractor's cost is his type and he would like to exaggerate this if he can. The final example is an investor (principal) and an investee (agent). The type is the investment's profitability and the investee might exaggerate future outlooks to get a higher investment.

When there is moral hazard, the principal cannot verify and control the agent's actions after a contract is signed. An example is when a car insurance company (principal) can not check how

reckless a client (agent) drives. Another example is an employer (principal) that can not see if a worker (agent) is slacking off. Principal-agent problems can have both adverse selection and moral hazard or only one of them. A principal-agent problem with both adverse selection and moral hazard is called a *generalised principal-agent problem*.

When the agent's strategy to truthfully declare their hidden information is a Nash-equilibrium, then we say that the mechanism is *incentive compatible*. We will restrict ourselves to finding optimal mechanisms that are incentive compatible. It turns out that the optimal incentive compatible mechanism is also optimal in the general class of mechanisms [3].

1-3 Real World Examples

In this section we give some examples of real world contracts that are studied by modelling it as a principal-agent problem. In agriculture, one type of contract is a sharecropping agreement. In this agreement the landowner (the principal) allows a tenant (the agent) to use her farmland in return for a share of the crops. Stiglitz [4] studied the sharecropping system using principal-agent models. He considered the cases with symmetric information and with moral hazard.

Principal-agent models are often applied to employment contracts. Throughout human history the majority of labour transactions are coercive. Think of slavery and forced labour. Chwe [5] and Acemoglu and Wolitzy [6] used principal-agent models to study these coercive relationships.

Krinsky and Mehrez [7] studied a principal-agent problem where the principal (the lessor) leases equipment to an agent (the lessee). During use, the equipment suffers from wear and could breakdown before the end of the contract. The lessee can do preventive maintenance during the lease time to extend the expected operation time of the equipment. If a breakdown happens before the end of contract, it can have a negative impact on the lessee's profits. When the equipment breaks down, the lessor has to do a costly overhaul maintenance. If no breakdown happens, then the cost to bring it back to a standard level is much lower. The lessor's problem is to find the optimal lease time that maximizes his expected profit.

1-4 Problem Statement

In current literature, principal-agent problems under moral hazard and adverse selection are only studied for analytical solutions under restrictive convexity assumptions. To cover more general cases, we develop numerical methods for the moral hazard problem and generalised principal agent problem under the following assumptions:

1. The agent's type and action spaces is a closed continuous interval.
2. The agent's expected utility is non-convex in action and type.

In chapter 2 we give an overview of existing literature. chapter 3, we develop numerical solution methods for a moral hazard problem and we test it on a option contract problem given in [8]. In chapter 4 we introduce numerical solution methods for the generalised problem and we test it on a option contract problem with adverse selection and moral hazard. We end with conclusion, recommendations and future research in chapter 5.

Chapter 2

Literature

2-1 Analytical Solutions to Moral Hazard Problems

Let V_P and V_A be the principal's and agent's expected utility, respectively. The agent's action space is $\mathscr{A}$. The outcome space is $\mathscr{X}$ and $w : \mathscr{X} \to \mathbb{R}$ is the reward function that determines how much reward (often money) the agent gets depended on the outcome. The value v_0 is the participation constraint and the agent will only accept the contract if his expected utility is at least v_0. A contract is the pair $(a, w(\cdot))$, where a is the recommended action. The moral hazard optimisation problem is:

$$\max_{a \in \mathscr{A}, w(\cdot)} V_P(a, w) \tag{2-1a}$$

$$\text{s.t.} V_A(a, w) \geq v_0 \tag{2-1b}$$

$$a \in \arg\max_{\hat{a} \in \mathscr{A}} V_A(\hat{a}, w). \tag{2-1c}$$

Constraint (2-1b) is called the participation constraint and it ensures that the agent will accept the contract. Constraint (2-1c) is the incentive compatibility constraint and it ensures that the recommended action is also the agent's best response.

Since (2-1b) is also an optimisation problem, we say that (2-1) is a *non-linear bilevel optimization problem*. A common approach is to replace (2-1c) with the first-order necessary conditions for optimality. This called the *first order approach*. The original bilevel program 2-1 is replaced by:

$$\max_{a \in A, w(.)} V_P(a, w) \tag{2-2a}$$

$$\text{s.t.} V_A(a, w) \geq v_0 \tag{2-2b}$$

$$\frac{\partial V_A}{\partial a'}(a) = 0. \tag{2-2c}$$

We say the first-order approach is *valid* if 2-1 and 2-2 have the same solution. This is not always the case; see for example [9] or the option contract in [8]. A sufficient condition for validity is that V_A needs to be concave in a and the action space $\mathscr{A}$ is an open interval in $\mathbb{R}$. Special cases that assume this condition are studied in [10], [11] and [12].

2-2 Analytical Solutions to Adverse Selection Problems

Let $\mathscr{T}$ be the type space, $\mathscr{A}$ be the action space and $\mathscr{X}$ be the outcome space. At the start of the contracting game, the principal asks an agent with true type $\tau \in \mathscr{T}$ to report his type. The agent knows which contract he gets depending on the type he reports and he might not report his true type. Based on the agent's report $\tau' \in \mathscr{T}$, the principal will propose a contract that depends on the reported type: $\left(\alpha\left(\tau'\right), w\left(\cdot, \tau'\right)\right)$, where $\alpha : \mathscr{T} \to \mathscr{A}$ generates the recommended action and $w : \mathscr{X} \times \mathscr{T} \to \mathbb{R}$ is the reward function. Different from the moral hazard problem is that the agent will surely perform the recommended $\alpha\left(\tau'\right)$.

An alternative, but equivalent mechanism is one where the principal does not directly ask for the type, but instead, she offers a menu of contracts. Each contract is assigned to a specific type and the agent will select the contract that will give him the highest expected utility. The menu of contracts is incentive compatible, if the agent selects the contract that is designed for his true type.

Let the principal's expected utility be V_{P}. The agent with type τ has an expected utility V_τ and his reported type is τ'. For better readability, we use the following abbreviations: $\alpha_\tau = \alpha(\tau)$ and $w_\tau = w(\tau)$. The optimisation problem under adverse selection is:

$$\max_{\alpha(.), w(.,.)} \; V_{\mathrm{P}}(\alpha, w) \tag{2-3a}$$

$$\text{s.t. } V_\tau\left(\alpha_\tau, w_\tau\right) \geq v_0 \tag{2-3b}$$

$$V_\tau\left(\alpha_\tau, w_{\tau'}\right) \geq V_\tau\left(\alpha_{\tau'}, w_{\tau'}\right) \tag{2-3c}$$

$$\forall \tau, \tau' \in \mathscr{T}$$

If the size of the type space $\mathscr{T}$ is finite than (2-3c) represents a finite number of constraints and (2-3) is a regular non-linear program. If $\mathscr{T}$ is infinite is size, for example an interval in $\mathbb{R}$, then we have infinite many constraints.

Chade [13] looked at a adverse selection insurance contract where the outcome is either a "win" or "lose". The agent's type is the probability of getting a "lose". If the outcome is "lose", then the principal has to pay a certain amount of money to the agent and otherwise nothing. The agent reports his type and the principal will offer a contract that states how much premium the agent has to pay and how much he gets in case of a "lose".

2-3 Analytical Solutions to Moral Hazard and Adverse Selection Problems

In the generalised principal-agent problem, there is both adverse selection and moral hazard. The agent declares a type $\tau' \in \mathscr{T}$ and the principal proposes a contract $\left(\alpha\left(\tau'\right), w\left(x, \tau'\right)\right)$. The action $\alpha\left(\tau'\right)$ is only a recommendation. The agent with true type τ will report a type τ' and select an action a that maximises his expected utility.

Let $\mathscr{T}$ be the type space and $\mathscr{A}$ be the action space. The principal's expected utility is V_{P}. An agent

with type τ has expected utiltiy V_τ. The optimisation problem is:

$$\max_{\alpha(\cdot),w(\cdot,\cdot)} V_P(\alpha(\cdot), w(\cdot,\cdot)) \tag{2-4a}$$

$$\text{s.t.} V_\tau(\alpha_\tau, w_\tau) \geq v_0 \tag{2-4b}$$

$$V_\tau(\alpha_\tau, w_\tau) \geq V_\tau\left(a, w'_\tau\right) \tag{2-4c}$$

$$\forall \tau, \tau' \in \mathscr{T}, \quad \forall a \in \mathscr{A}, \tag{2-4d}$$

Picard [14], and Guesnerie, Picard and Rey [15] studied the combined problem with risk neutral principal and agent. Risk neutrality means that the utility increases linear with the reward. Here, the output x depends on the effort a as $x = a + \epsilon$, where ϵ is a random variable with zero mean. It turns out that in many cases, optimal pure adverse selection contracts (where e is directly observable) are also optimal in the combined problem.

When the outcome is either a *succes* or a *failure*, Li [16] found an optimal contract when the principal and agent are risk neutral. An agent gets paid more if the outcome is a success. Li shows that a capable agent can be identified by the fact that he would be more willing to select a high reward and high punishment contract (high risk) instead of a low reward and low punishment contract (low risk).

Theilen [17] studied the case with a risk averse agent and found an optimum in the class of linear contracts. Contracts are called linear if the payment to the agent is of the form $w(x,\tau) = c_1(\tau)x + c_2(\tau)$. The reward is linear in the outcome x and the parameters c_1 and c_2 depends on the agent's declared type τ. The outcome x is normally distributed with the mean depended on the effort a. Linear contracts are easy to understand which makes them an attractive choice to use in real world contracts. We are interested to know, if there are non-linear incentive compatible contracts that will give a better expected value to the principal.

Faynzilberg and Kumar [18] applied the first-order approach to solve the generalised problem. This means that the incentive compatibility constraint 4-3c is replaced by the first order conditions

$$\frac{\partial V_\tau(\alpha(\hat{\tau}), w(\hat{\tau}))}{\partial \hat{\tau}}\Big|_\tau = 0 \tag{2-5}$$

$$\frac{\partial V_\tau(a, w(\tau))}{\partial a}\Big|_{\alpha(\tau)} = 0. \tag{2-6}$$

They showed that the sufficient conditions that made the first-order approach valid for the moral hazard case are not always valid for the generalised problem. The first order condition is valid for the generalised problem, if the probability density of the outcome is of the form:

$$p(x|a,\tau) = (a+\tau) f_H(x) + (1-(a+\tau)) f_L(x), \tag{2-7}$$

with $0 \leq a + \tau \leq 1$. The functions f_H and f_L are probability densities, such that f_H has first order stochastic dominance over f_L. According to Formula 2-7 the probability of getting an outcome x is affine in the effort a, which means that the expectating of outcome x is also affine in a.

2-4 Limitations of the Principal-Agent Model

In the principal-agent model, human behaviour is assumed to be rational. It is clear that not all individuals act rationally. Taking irrationality into consideration, with knowledge from behavioural economics can be topic for future research.

Most people are social creatures who don't always act selfishly. In business there are firms pay more wages than industry standard and have workers who work harder than the minimum requirements. This interaction can be considered as form of gift exchange [19]. By getting paid and being treated well, workers are willing to work harder, although they could get away by slacking off a little bit.

Here it is assumed that the agent's utility only depends on the money he receives. Self-fulfilment and prestige are non-monetary factors that can play a big role in motivating people. Another factor are fairness. The agent's utility not only depends on how much money he receives, but also on how much he is rewarded compared to what others get. Desiraju and Sappington (2007) [20] considers a multi-agent problem where the utility does depend on how much a principal earns compared to the other agents. A possible explanation why people compare their reward to others, is that they do not know how they ought te be treated, which comes from the fact that they are not perfectly rational.

2-5 Numerical Solution Methods

For the moral hazard problem, Cecchini [21] assumed that the agent's optimal action is an implicit function of the contract. The bi-level program becomes a non-linear optimisation problem and is then solved with the ellipsoid method. Cecchini tested the ellipsoid method on wage contracts for car dealer and the algorithm did find the global optimal solutions. No information on convergence speed or number of iterations until convergence were provided.

2-6 Non-Linear Bilevel Optimization

The moral hazard problem (2-1) is a bi-level optimisation problem. In this section we explore several solution methods to solve bi-level programs. The standard form of a non-linear bilevel program is:

$$\min_{a,\theta} F(a,\theta) \tag{2-8a}$$

$$\text{s.t. } G(a,\theta) \leq 0 \tag{2-8b}$$

$$a \in \arg\min_{a'} f\left(a',\theta\right) \tag{2-8c}$$

$$\text{s.t. } g\left(a',\theta\right) \leq 0. \tag{2-8d}$$

The functions F and f are referred to as the *upper-level* and *lower-level objective functions* respectively. Similarly, G and g are the *upper-level* and *lower-level contraints*. The program 2-8c with 2-8d is referred to as the *lower-level program*.

A difficulty with bilevel optimisation is when the lower-level program is non-convex. In that case, optimality of the lower-level program and feasability of the bi-level program cannot be guaranteed.

2-6-1 Descent Methods

For some θ, the *lower-level rational reaction set* is:

$$R(\theta) = \left\{ a : a \in \arg\min_{a'} f\left(a',\theta\right) \text{ s.t. } g\left(a',\theta\right) \leq 0 \right\}.$$

We say every $a \in R(\theta)$ is a *rational response* to θ. If $R\left(\hat{\theta}\right)$ is a singleton, then a can be considered to be an implicit function of θ, i.e. $a = a(\theta)$. In this case, descent methods are possible choices to solve the bilevel program. An iteration of a descend method will go as follows.

1. Given a feasible point $(a(\theta),\theta)$, find a feasible direction d along which F decreases.
2. Find a feasible step length such that

$$F(a(\theta+\gamma d),\theta+\gamma d) < F(a(\theta),\theta)$$

and then return to step 1 with the new feasible point $(a(\theta+\gamma d),\theta+\gamma d)$.

One difficulty is that if the lower-level program is non-convex, then there is no guarantee that the computed $a(\theta)$ is a global optimum.

Descent Direction

The steepest descent direction is $d = -\nabla_\theta F(a(w),\theta)$. Like in [21], we can numerically approximated the gradient with the central difference method. First define:

$$\theta_{i,+h} = (\theta_1,\ldots,\theta_{i-1},\theta_i+h,\theta_{i+1},\ldots,\theta_n)$$
$$\theta_{i,-h} = (\theta_1,\ldots,\theta_{i-1},\theta_i-h,\theta_{i+1},\ldots,\theta_n),$$

and let:

$$a_{i,+h} \in R\left(\theta_{i,+h}\right)$$
$$a_{i,-h} \in R\left(\theta_{i,-h}\right)$$

The vector entry i of the gradient $\nabla_\theta F$ is:

$$\frac{\partial F}{\partial \theta_i} \approx \frac{F(a_{i,+h},\theta_{i,+h}) + F(a_{i,-h},\theta_{i,-h})}{2h}, \tag{2-9}$$

with $h > 0$ a small constant.

Central differencing requires two lower level optimizations to compute one vector entry. We can reduce the amount of optimizations to one, by using forward or backwards differencing. The truncation error with forward and backward differencing will be of order $\mathcal{O}(\delta)$ while central differencing has a truncation error of order $\mathcal{O}\left(\delta^2\right)$. The higher errors might not be a problem if δ is taken small enough.

Another method to compute $\nabla_w F(a(\theta),\theta)$ was proposed by Kolstad and Lasdon (1990) [22]. Computing the gradient does require one optimisation of the lower-level program, but unlike the finite difference method, the whole gradient vector is derived instead of just one entry. The method is tested on a bilevel program with no upper level constraints, 30 lower-level constraints and 230 variables. Convergence is reached in 18 iterations and the CPU time is less than a minute.

If there are no upper level constraints, Savard and Gauvin (1994) [23] found the steepest descent direction by solving a *linear-quadratic bilevel program*. The linear-quadratic bilevel can be solved by exact algorithms like the one by Bard and Moore (1990) [24]. It was demonstrated, step by step, that the method could find the global solution of a quadratic bilevel program.

2-6-2 Barrier Function Method

With the barrier function method, we replace the lower level constraint program:

$$\begin{aligned} &\min_{a} f(a,\theta) \\ &\text{s.t. } g(a,\theta) \leq 0, \end{aligned}$$

with the unconstrained program

$$\min_{a} \hat{f}(a,\theta) = \min_{a} f(a,\theta) + r\phi(a), \tag{2-10}$$

with constant $r > 0$ and ϕ a continues barrier function that satisfies

$$\begin{aligned} &\phi(a) > 0 && \text{if } a \in \operatorname{int}\{a : g(a,\theta) \leq 0\} \\ &\phi(a) \to +\infty && \text{as } a \to \partial\{a : g(a,\theta) \leq 0\}. \end{aligned}$$

If (2-10) is a convex, then it is replaced by the first order optimality condition:

$$\frac{\partial \hat{f}(a,\theta)}{\partial a} = 0.$$

Aioyoshi and Shimizo [25] showed that the sequence of approximate solutions converges to the solution of the original bilevel program as $r \to 0$. The algorithm was tested on a bilevel program, with a convex lower-level program. There number of upper- and lower-level constraints were 8 and 9 respectively. The known optimal upper-level objective function value is 6600 and the approximate value of that is 6586, where $r = 0.001$ was used. The approximate value is 0.21% from the analytical solution. The CPU time was around 50 seconds on a Burroughs B-5700. An alternative to the barrier function 2-10 is to use a penalty function:

$$\hat{f}(a,\theta) = f(a,\theta) + r\sum \max\big(g_j(a,\theta), 0\big). \tag{2-11}$$

2-6-3 Trust Region Method

For the case that the upper-level constraint only depends on θ, i.e. $G(a,\theta) = G(\theta)$, Colso et al. (2005) [26] developed a trust region algorithm. The trust region method is an iterative method. An iteration will go as follows: at an iteration point (a_k, θ_k), we compute linear approximations $\big(\tilde{F}, \tilde{G}, \tilde{g}\big)$ of $\big(F, G, g\big)$ and a quadratic approximation $\tilde{f}$ of f. The following linear-quadratic program is then solved

$$\min_{a,\theta} \tilde{F}(a,\theta) \tag{2-12a}$$

$$\text{s.t. } \tilde{G}(\theta) \leq 0 \tag{2-12b}$$

$$a \in \arg\min_{a'} \tilde{f}\left(a', \theta\right) \tag{2-12c}$$

$$\text{s.t. } \tilde{g}\left(a', \theta\right) \leq 0. \tag{2-12d}$$

Let $(\tilde{a}_k, \tilde{\theta}_k)$ be the solution of 2-12. The quality of the linear and quadratic models is determined by the ratio of the actual reduction

$$F(a_k, \theta_k) - F\left(a_k^*, \tilde{\theta}_k\right),$$

with $a_k^* \in R(\tilde{\theta}_k)$ and the predicted reduction

$$\tilde{F}(a_k, \theta_k) - \tilde{F}\left(\tilde{a}_k, \tilde{\theta}_k\right)$$

of the upper-level objective function F. This ratio is

$$\rho_k = \frac{F(a_k, \theta_k) - F\left(a_k^*, \tilde{\theta}_k\right)}{\tilde{F}(a_k, \theta_k) - \tilde{F}\left(\tilde{a}_k, \tilde{\theta}_k\right)}. \tag{2-13}$$

If the original lower-level program is non-convex, global optimality of a_k^* is generally not guaranteed. Depending on ρ_k, the algorithm updates the current iterate and the trust-region radius, and the process is repeated until convergence.

The algorithm is tested on 19 problems including some toll-setting and network design problems. Some of the problems do not have convex lower-level programs. The algorithm found all optimal solutions in less then 50 iterations and most problems converge in less then 10 iterations.

2-7 Semi-Infinite Programming

For the adverse selection and generalised problem, we have to find the optimal functions $\alpha(\cdot)$ and $w(\cdot,\cdot)$. To optimise in the parameter space, we do a basis function approximation of α with n basis function and w with m 2-dimensional basis functions. Let $\theta \in \mathbb{R}^{nm}$, be the vector that stores the basis coefficients of both α and w. By using a basis function approximation, we can turn 2-3 into a semi-infinite programming problem. A semi-infinite programming problem has infinitely many constraints and finitely many decision variables. The standard semi-infinite program (SIP) is in the form:

$$\min_{\theta} F(\theta) \tag{2-14a}$$

$$\text{s.t. } g(\theta, y) \leq 0 \tag{2-14b}$$

$$y \in Y,$$

where $Y = \left[\overline{y}, \underline{y}\right] \subseteq \mathbb{R}$ is a closed interval. It is easy to see that the SIP program 2-14 is equivalent to:

$$\min_{\theta} F(\theta) \tag{2-15a}$$

$$\text{s.t. } f(\theta) = \max_{y \in Y} g(\theta, y) \leq 0. \tag{2-15b}$$

If the maximisation problem in Equation 2-15b is concave, then the SIP program 2-15 can be formulated into a programming problem with mathematical constraints (MPCC), see [27]. There are algorithms to solve MPCC programs. For a study on MPCC check [28].

2-7-1 Discretization

We can relax program 2-3, by replacing Y with a finite subset $\tilde{Y} \subset Y$. The set of incentive compatibility constraints described by 2-3c is then finite and the new discretized program is a finite non-linear constraint program. The solution of the discretized program is denoted as $\tilde{\theta}$. The algorithm goes as follows:

1. Solve the finite discretized program and interpolate $\tilde{\theta}$. The interpolation of $\tilde{\theta}$ is denoted as $\tilde{\theta}_{\text{ip}}$.
2. Check on a "super fine" grid, if the incentive compatibility constraints 2-3c of the original semi-infinite program are satisfied within a fixed accuracy:

   $$g(\theta, y) \leq \epsilon,$$

   where $\epsilon > 0$ a small constant. If these constraints are not satisfied, then retry step 1 with a finer grid on Y, otherwise stop.

Consider a sequence of grids $\left\{\tilde{Y}_k \subset Y\right\}_{k=1}^{\infty}$, such that the grids gets finer as the index k increases. The feasible set when using a grid $\tilde{Y}_k$ is denoted as $Z(\tilde{Y}_k)$. If $Z(\tilde{Y}_1)$ is compact, then the solutions of the discretized program, with grid $\tilde{Y}_k$, converges to the solution of the original semi-infinite program as $k \to \infty$. For details check [29].

If $|\tilde{Z}|$ is the size of the set $\tilde{Z}$, then the number of incentive compatibility constraints is $|\tilde{Z}|^2$. Having a fine grid can become computational expensive to solve.

A nice thing about the discretization method is that we do not need to do a basis function approximation for $\alpha(\cdot)$ and $w(\cdot,\cdot)$. Since we have finite number of types in the discretized problem, $\alpha(\cdot)$ and $w(\cdot,\cdot)$ can be represented as finite length vectors.

2-7-2 Adaptive Convexification

Another numerical method to solve SIP programs is adaptive convexification [30, 31]. Instead of taking finitely many elements from the space Y as in the discretization method, we can take finitely many sub intervals. For $\underline{y} = y_0 < y_1 \ldots < y_n = \overline{y}$, define the subintervals $Y_k = \left[y_{k-1}, y_k\right]$. All elements in $\{Y_k\}_1^n$ form a tessellation of Y. The idea is to construct overestimators g_k of g that are concave on Y_k. We will then replace 2-15b by the finitely many concave programs

$$f_k = \max_{y \in Y_k} g_k(\theta, y) \leq 0$$
$$k \in \{1, \ldots, n\}.$$

The new SIP program can be solved by reformulating it into a MPCC problem. Since $g_k\left(\theta, y\right) \geq g\left(\theta, y\right)$ the new solution is also feasible in the original SIP program.

As the tessellation gets finer, the solution of the adapted program will converge to the original SIP. It has also been shown that the algorithm will terminate in finite amount of steps. In [30], the algorithm is tested on a design centering problem and the runtime is 3 minutes.

2-8 Conclusion

Principal-agent problems are bi-level programs or semi-infinite programs. We have that the optimisation parameters are solutions of a lower-level optimisation problem. Convex lower-level programs can be replaced by the KKT-conditions. For non-convex lower-level problems, we consider the solution to be a function that is defined implicitly by the numerical solution of the lower level program. For problems under adverse selection, the discretization method is nice because it transforms the function optimisation problem into a parametric. So we do not need to do a basis function approximation.

Chapter 3

Moral Hazard

3-1 The Contracting Game

We first define the following:

- The *action space* $\mathscr{A} = [\underline{a}, \overline{a}]$ is an interval in $\mathbb{R}$.
- We have an *outcome space* $\mathscr{X} \subseteq \mathbb{R}$. The *outcome* is a random variable X, that takes values in $\mathscr{X}$. The probability density of the outcome is $p(x|a)$, and it depends on the agent's action $a \in \mathscr{A}$.
- The agent's *reward* is a function $w : \mathscr{X} \to \mathbb{R}$.
- The agent's *contract* is the pair $(a, w(.))$, where $a \in \mathscr{A}$ is the recommended action and $w(.)$ is te reward function.

The actions and time line of the contracting game is shown in Figure 3-1 and it goes as follows:

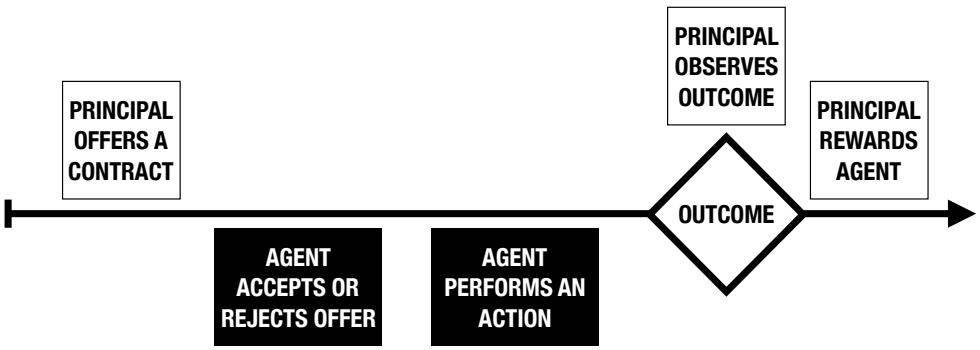


Figure 3-1: Time line of the contracting game with moral hazard.

1. The principal offers a contract $w(.)$ to the agent.
2. The agent will either accept or reject the offer.
3. If the offer is accepted, the agent will perform some action $a \in A$.
4. A random outcome $x \in \mathscr{X}$ takes place, that is conditionally depended on the agents action a.
5. The principal observes the outcome x and rewards the agent $w(x)$.

At the and of the contracting game, the agent will have an utility v_{A} and the principal will have an utility v_{P}:

$$\begin{aligned} v_{\mathrm{A}}\left(a, w, x, t'\right) &= u_{\mathrm{A}}\left(w\left(x\right)\right) - h(a) \\ v_{\mathrm{P}}\left(a, w, x, t'\right) &= x - w\left(x\right), \end{aligned}$$

with $\frac{\partial h}{\partial a} \geq 0$, $u'_{\mathrm{A}}(.) \geq 0$ and $u''_{\mathrm{A}}(.) \leq 0$. The agent's utility goes down in a, so a higher action is modelled as bringing dis-utility to the agent, and it can be interpreted as the amount of effort the agent puts in.

Since the outcome X is random, we have to maximise expected utilities. The agent's expected utility is:

$$V_{\mathrm{A}}(a, w) = \int_{\mathscr{X}} u_{\mathrm{A}}\left(w\left(x\right)\right) p\left(x|a\right)\ dx - h(a). \tag{3-1}$$

The agent's reservation utility is v_0 and it is the utility he has if he declines the contract. In step 4 of the contracting game, the agent will only accept the offer if $V_{\mathrm{A}} \geq v_0$. If the agent does action a, then the principal's expected utility is:

$$V_{\mathrm{P}}(a, w) = \int_{\mathscr{X}} (x - w(x)) p(x|a)\ dx. \tag{3-2}$$

3-2 The Moral Hazard Optimisation Problem

In finding the optimal contract $w(.)$ that maximises the principal's utility, we have to take into account that the agent will select an optimal $a^* \in \mathscr{A}$, such that

$$a^* \in \arg\max_{a \in \mathscr{A}} V_{\mathrm{A}}(a, w(x)).$$

The optimization problem is

$$\max_{a \in \mathscr{A}, w(.)} V_{\mathrm{P}}(a, w) \tag{3-3a}$$

$$\text{s.t.} V_{\mathrm{A}}(a, w) \geq v_0 \tag{3-3b}$$

$$a \in \arg\max_{\hat{a} \in \mathscr{A}} V_{\mathrm{A}}(\hat{a}, w). \tag{3-3c}$$

Constraint (3-3c) is called the *incentive compatibility* constraint. The incentive compatibility constraint ensures that the principal's expected utility depends on an effort that maximises the agent's

expected utility. If the action set $\mathscr{A}$ is a finite set, then 3-3c turns into a finite set of inequality constraints and we get a regular non-linear optimisation problem. See for example [32].

We can find the optimal $w(\cdot)$ in a class of all functions from $\mathscr{X}$ to $\mathbb{R}$, or we assume a predefined structure on $w(\cdot)$. If we take a predefined structure of $w(\cdot)$, then $w(\cdot)$ is a function that depends on constants parameter $\theta_1,\ldots,\theta_n$. Finding the optimal $w(\cdot)$, is reduced to finding the optimal $\theta_1^*,\ldots,\theta_n^*$. An example of this is the linear contract $w(x)=\theta_1 x+\theta_2$, where need to find the optimal $\left(\theta_1^*,\theta_2^*\right)$.

3-3 Bilevel Optimisation

The standard form of a non-linear bilevel program is:

$$\min_{a,\theta} F(a,\theta) \tag{3-4a}$$

$$\text{s.t. } G(a,\theta)\leq 0 \tag{3-4b}$$

$$a\in \arg\min_{a'} f\left(a',\theta\right) \tag{3-4c}$$

$$\text{s.t. } g\left(a',\theta\right)\leq 0. \tag{3-4d}$$

The functions F and f are referred to as the *upper-level* and *lower-level objective functions* respectively. Similarly, G and g are the *upper-level* and *lower-level contraints*. Program (3-4c) with (3-4d) is referred to as the *lower-level program*.

We can see that the moral hazard problem (3-3) is a standard bilevel-optimisation problem (3-4), by setting:

$$\begin{aligned} F(a,\theta) &= -V_{\mathrm{P}}(a,\theta)\\ G(a,\theta) &= v_0 - V_{\mathrm{A}}(a,\theta)\\ f(a,\theta) &= -V_{\mathrm{A}}(a,\theta)\\ g_1(a,\theta) &= \underline{a}-a\\ g_2(a,\theta) &= a-\overline{a}. \end{aligned}$$

For some $\hat{\theta}(x)$, the *lower-level rational reaction set* is:

$$R\left(\hat{\theta}\right)=\left\{a\in A\,\middle|\, a\in \arg\min_{a'} f\left(a',\hat{\theta}\right) \text{ s.t. } g\left(a',\hat{\theta}\right)\leq 0\right\}.$$

Every $a\in R\left(\hat{\theta}\right)$ is called a *rational response* to $\hat{\theta}$. Note that a higher effort level will increase the principal's utility. Assuming that the agent is *cooperative*, he will select effort level $a^*(\theta)=\max R\left(\hat{\theta}\right)$. If $\max R\left(\hat{\theta}\right)$ is unique, then $a^*(\theta)$ is an function implicitly depended on θ. In that case, the bi-level optimisation problem (3-4) tcan be considered as the non-linear program:

$$\min_{\theta} F\left(a^*(\theta),\theta\right) \tag{3-5a}$$

$$\text{s.t. } G\left(a^*(\theta),\theta\right)\leq 0. \tag{3-5b}$$

For the computation of $a^*(\theta)$ and the optimisation of (3-5) we can use non-linear constraint optimisation methods like *Sequential Quadratic Programming (SQP)*. Non-linear optimisation methods like SQP requires the gradients of F and G. If an analytical expression of the gradient is not known a numerical approximation is needed. This can be done by Formula (2-9).

Other methods to solve the bilevel programs is the barrier function method [25] and the trust-region method [26]. The barrier function method by Aioyoshi and Shimizo only works when the inner program is convex. The trust-region method is for problems where the upper-level constraint only depends on θ.

3-4 Example: Option Contract

In this section we solve the option contract problem given by Lambert [8]. In this option contract, the agent receives an *European call option* in addition to a fixed wage. A European call option is a financial contract where after the *expiration date* the holder has the option to buy a number of company shares at a *strike price*. The strike price is a fixed value and does not change regardless of the current stock price. If the stock price is higher than the strike price, then the agent would surely want the exercise his right to buy the shares. On the other hand, he will not buy if the stock price is lower than the strike price, because he is not obliged to buy.

Let c_1 be a constant fixed wage, c_2 the number of shares, x the stock price at the expiration date and K the exercise price. The agent's total wage is

$$w(x) = c_1 + c_2 \max(x - K, 0).$$

The vector $\theta = (c_1, c_2, K)$ stores the contract parameters and defines the contract. In this example, we assume the stock price x is a realisation of a random variable X, that is normal distributed with mean $\mu = 100 + a$ and variance σ^2. The value $a \in \mathscr{A} = [0, 100]$ is the effort level of the agent. More effort will increase the probability of getting a higher stock price.

In this example, the disutility in effort is -100^2. The agent's utility is then:

$$v_{\mathrm{A}} = w(x) - 100a^2 = \alpha + \beta \max(x - K, 0) - 100a^2.$$

With the help of Appendix A, we can show that the utility is:

$$\begin{aligned} V_{\mathrm{A}}(\theta, a) &= \mathbb{E}[v_{\mathrm{A}}] = c_1 + c_2 \mathbb{E}[\max(X - K, 0)] - 100a^2. \\ &= c_1 + c_2 \left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y \left(1 - F\left(-\mu_y/\sigma\right)\right) \right] - 100a^2, \end{aligned}$$

with $\mu_y = 100 + a - K$ and $F(\cdot)$ the cumulative density function of the standard normal. Example plots of the agent's expected utility is shown in Figure 3-2. We can see that V_{A} is non-convex with two local maxima. Also, the global optimal effort can jump between regions of lower and higher effort. This implies that the optimal effort is non-continuous in θ.

The principal's expected utility is:

$$\begin{aligned} V_{\mathrm{P}}(\theta, a) &= \mathbb{E}[Lx - c_2 \max(x - K, 0) - c_1] \\ &= L(100 + a) - c_2 \left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y \left(1 - F\left(-\mu_y/\sigma\right)\right) \right] - c_1, \end{aligned} \tag{3-6}$$

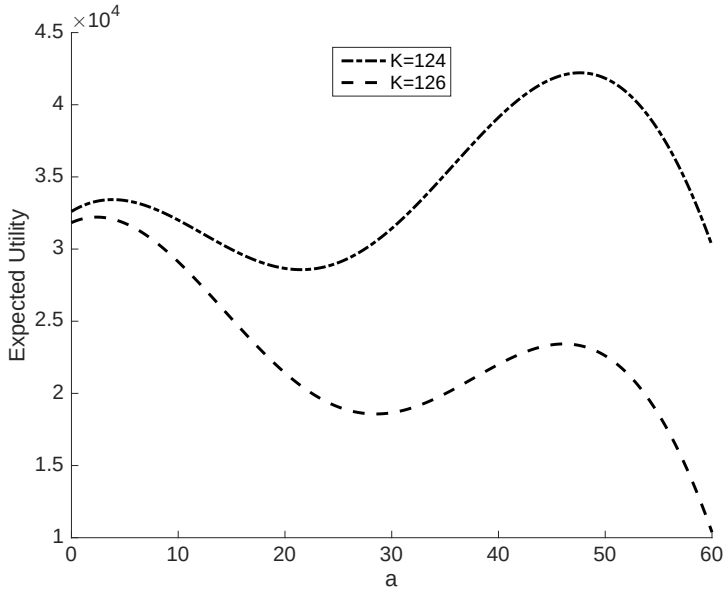


Figure 3-2: Agent's expected utility plotted over the effort. Parameters are $\sigma^2 = 200$, $\alpha = 30000$, $\beta = 10000$ and $K = 124$ or $K = 126$.

where L is a fixed number and it is the total amount of stocks the principal owns and $F(\cdot)$ is the cumulative distribution of the standard normal distribution.

Defining $\theta = (\alpha, \beta, K)$, we want to find the solution of the following program:

$$\max_{\theta,a} L(100+a) - \beta\left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y\left(1 - F\left(-\mu_y/\sigma\right)\right)\right] - \alpha \tag{3-7a}$$

$$\text{s.t.} \left(\alpha + \beta\left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y\left(1 - F\left(-\mu_y/\sigma\right)\right)\right] - 100a^2\right) \geq v_0 \tag{3-7b}$$

$$0 \leq \theta \leq \bar{\theta} \tag{3-7c}$$

$$a \in \arg\max_{\hat{a}\in A}\left(\alpha + \beta\left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\hat{\mu}_y^2}{2\sigma^2}} + \hat{\mu}_y\left(1 - F\left(-\hat{\mu}_y/\sigma\right)\right)\right] - 100\hat{a}^2\right), \tag{3-7d}$$

with $\hat{\mu}_y = 100 + \hat{a} - K$. We introduce an artificial upper bound $\overline{\theta}$ so that Matlab multi-start solver can select random starting points between 0 and $\overline{\theta}$. The bound will be increased if $\theta \leq \overline{\theta}$ is equal for some entries in θ.

The fixed parameters in the problem are shown in Table 3-1. The optimisation parameters θ are bounded by:

$$0 \leq \theta \leq \overline{\theta} = (40000; 0.49L; 300).$$

We bound the fixed wage by 40000, because it does not make sense to give him guaranteed payment higher than his reservation utility. The number of shares the agent can buy β, can not be more than halve the principal's total shares. For multi-start optimisation we bound the strike price to 300, which hopefully will be sufficiently high. If this constraint turns out to be active, we will increase the bound.

Table 3-1: Constant parameters of the option contract problem.

Description	Symbol	Value
Action space	$\mathscr{A}$	$[0,100]$
Outcome space	$\mathscr{X}$	$\mathbb{R}$
Participation utility	v_0	40000
Number of shares	L	50000
Volatility	σ^2	200

Table 3-2: Five examples of optimal option contracts.

Fixed wage c_1^*	Number of shares c_2^*	Strike price K^*
11780	22999	155.3
36039	21754	153.9
1542	23028	154.9
5468	20229	148.9
10892	22692	154.7

The optimisation is done with the SQP solver in Matlab, with multi-start. The systems specifications are in B. The solver selects 50 uniform and random starting points, the algorithm finishes successfully in 68 seconds. There can be many different optimal contracts. Examples of optimal contracts are given in Table 3-2. In each optimal contract, the principal receives a utility of $V_{\mathrm{P}}^* = 8960000$, the agent's optimal effort is $a^* = 100$ and the agent's utility is $V_{\mathrm{A}}^* = 40000$. The principal makes the agent do his maximum effort, while only paying the reservation utility. We can conclude that in this case there is no other type of reward function better then the option contract.

3-5 Basis Function Approximation

The reward function $w(\cdot)$ does not need to have a predefined structure like in the option contract. Instead, we can consider $w(\cdot)$ to be any function. We do get a functional optimisation problem that is difficult the solve numerically. To transform the functional optimisation problem into a parametric one, a basis function approximation of $w(\cdot)$ can be done:

$$w(\cdot) = \sum_{i=1}^{n} \theta_i \phi_i(\cdot),$$

where θ_i is the i-th basis function coefficient and $\phi_i(\cdot)$ is the i-th basis function. After setting the type and amount of basis functions, a parameter optimisation of the coefficients $\theta = (\theta_1, \ldots, \theta_n)$ can be performed. With an appropriate set of basis functions, any function can be approximated with arbitrary precision.

3-5-1 Radial Basis Functions

One type of basis functions are the *radial basis functions (RBFs)*. An RBF is a function that satisfies: $\phi_i(x, x_i) = \phi_i(\|x - x_i\|)$, where x_i is a constant that is called the *centre* of $\phi_i(x, x_i)$ and $\|\cdot\|$ the

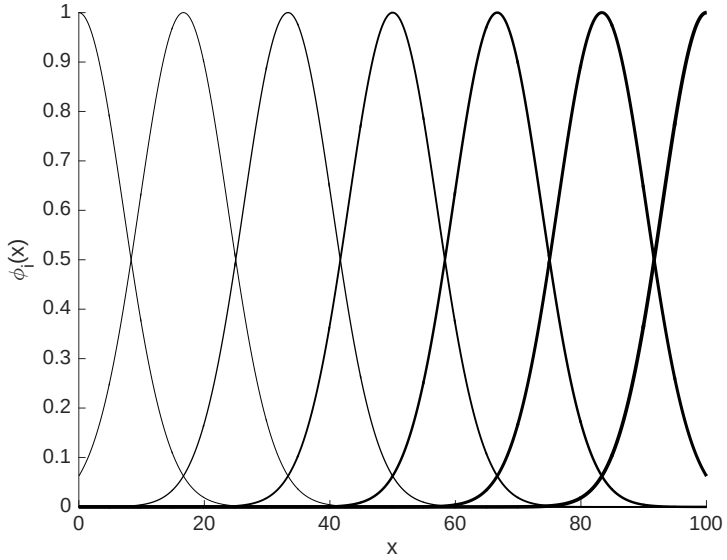


Figure 3-3: Example of 7 Gaussian basis functions

euclidean norm, usually the 2-norm. The value $\phi_i(x, x_i)$ only depends on the distance between x and the centre x_i.

An example of an RBF is the Gaussian

$$\phi_i(x) = e^{-(\epsilon|x-x_i|)^2}. \tag{3-8}$$

Figure 3-3 shows some examples of Gaussian basis functions.

3-5-2 B-Splines

Another type of basis functions are *B(asis)-splines*. The contents of this section is based on [33]. A B-spline basis function is denoted as $\phi_{i,k}$, where i is the i-th basis function and k is the order. A linear combinations of B-splines:

$$w(\cdot) = \sum_{i=1}^{n} \theta_i \phi_{i,k}(\cdot)$$

is a piecewise polynomial of degree $k-1$. For example, if $k = 2$, then $w(x)$ is a piece wise linear function.

To construct the B-splines, we start by defining a non-decreasing knot sequence $(t_1, t_2, \ldots, t_m)$. The first-order B-spline is:

$$\phi_{i,1}(x) = \begin{cases} 1, & \text{if } t_i \leq x < t_{i+1} \\ 0, & \text{otherwise.} \end{cases}$$

Higher order B-splines of order k are then defined recursively as

$$\phi_{i,k}(\cdot) = w_{i,k}(\cdot)\phi_{i,k-1}(\cdot) + \big(1 - w_{i+1,k}(\cdot)\big)\phi_{i+1,k-1}(\cdot),$$

with

$$w_{i,k}(x) := \begin{cases} \frac{x-t_i}{t_{i+k-1}-t_i}, & \text{if } t_{i+k-1} \neq t_i \\ 0, & \text{otherwise.} \end{cases}$$

We can see that if $\phi_{i,k-1}(\cdot)$ is a piecewise polynomial of degree $k-2$ and if $w_{i,k}(\cdot)$ is a polynomial of degree 1, then $\phi_{i,k}(\cdot)$ is a piecewise polynomial of degree $k-1$. Secondly if $\phi_{i,k-1}(\cdot)$ and $\phi_{i+1,k-1}(\cdot)$ have support on $[t_i, t_{i+k-1})$ and $[t_{i+1}, t_{i+k})$ respectively, then $\phi_{i,k}(\cdot)$ will have support on $[t_i, t_{i+k})$.

Each B-spline $\phi_{i,k}(\cdot)$ is uniquely defined by the $k+1$ knots $(t_i, t_{i+1}, \ldots, t_{i+k})$. From this we can see the following relationship between the number of knots, the number of basis functions and the order of each B-spline:

$$\#\text{knots} = \#\text{basis functions} + \text{B-spline order}. \tag{3-9}$$

We are going to give a few examples of knot sequences and basis functions.

An example of a knot sequence on the interval $[0, 100]$ is

$$t = \left(0, 0, \frac{100}{6}, \frac{100}{3}, 50, \frac{200}{3}, \frac{500}{6}, 100, 100\right).$$

Note that it is allowed to have $t_i = t_{i+1}$ as long as sequence t is non-decreasing. If the number of basis functions is seven, then according to (3-9) we will get a set of seven second-order B-splines. These are plotted in Figure 3-4. They are also known as *linear basis functions*, which are often used in for example finite element methods.

Another example is the knot sequence

$$t = (0, 0, 0, 0, 0, 0, 0, 100, 100, 100, 100, 100, 100, 100),$$

which has 14 knots on the interval $[0, 100]$. If we use seven seventh-order B-splines on this knot sequence, we get what is called a *Bernstein basis* [34]. The functions are plotted in Figure 3-5. On an interval $[a, b]$, any polynomial of degree less than k can be written as a linear combination of Bernstein basis functions of order k, that are constructed from the knot sequence:

$$t = (\underbrace{a, \ldots, a}_{k \text{ times}}, \underbrace{b, \ldots, b}_{k \text{ times}}).$$

The Bernstein basis is an alternative for the power basis and Lagrange basis for polynomials.

3-5-3 Assessment of Basis Functions

To get an efficient algorithm for the Principal-Agent problem, we want to have a good function approximations with small number of basis function coefficients.

To get an idea what kind of basis functions might work well we design the following test problem. We approximate the function

$$f(x) = \frac{100}{1 + 0.01(x-40)^2} \tag{3-10}$$

by a basis function approximation

$$\hat{f}(x) = \sum_{i=1}^{n} \theta_i \phi_i(x)$$

over the interval $[0, 100]$. The Function (3-10) is plotted in Figure 3-6 and it is a slight modification of the *Runge function* function used in *Runge's example* [35]. In this example, Runge demonstrated that using polynomial interpolation to approximate the Runge function will not lead to a converging sequence of polynomials. By approximating (3-10), we want to illustrate a scenario where using

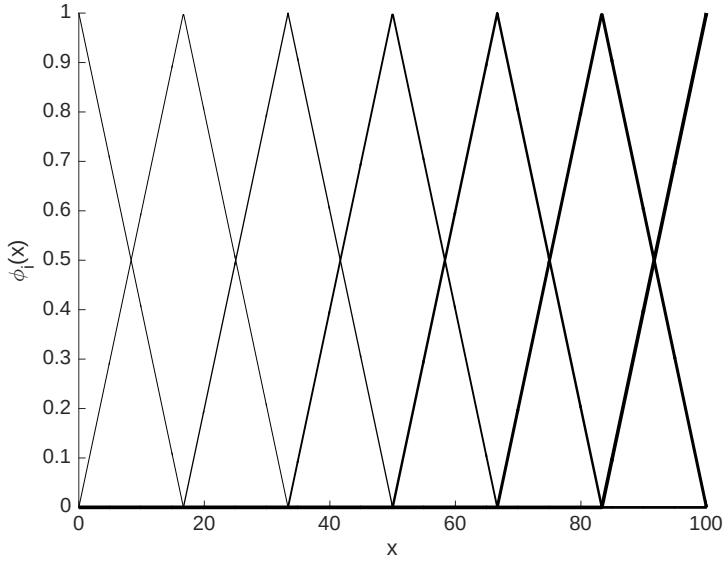


Figure 3-4: Seven 2nd order B-spline basis functions.

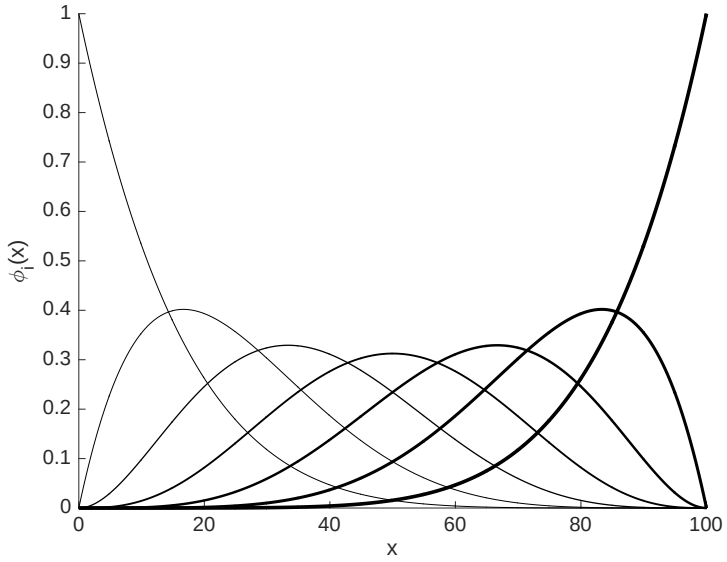


Figure 3-5: Bernstein basis of order 7.

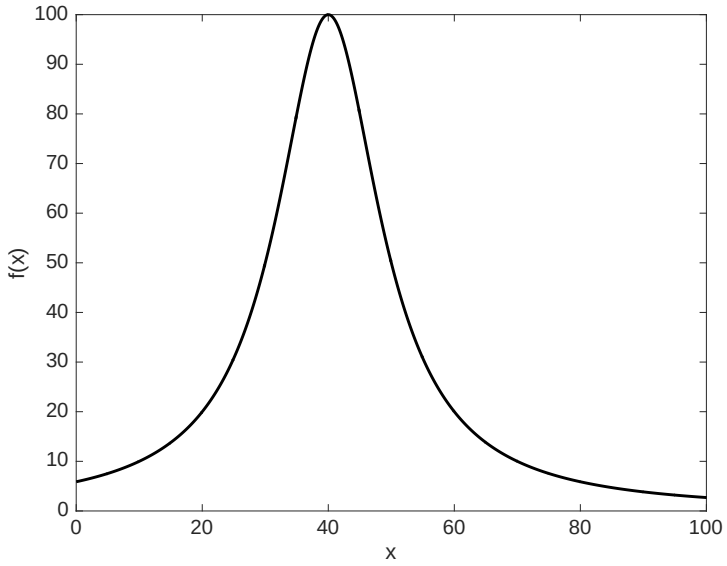


Figure 3-6: Modified Runge function

high degree piecewise polynomial approximation can lead to slow convergence and oscillations. Oscillations are particular bad in principal-agent problems, because it results in many local maxima in the inner program.

We define $\mathbf{f} = \big(f(0), f(1), \ldots, f(100)\big)$ and $\hat{\mathbf{f}} = \big(\hat{f}(0), \hat{f}(1), \ldots, \hat{f}(100)\big)$. The optimal basis function coefficients will then minimise:

$$\min_{\theta} \|\mathbf{f} - \hat{\mathbf{f}}\|, \tag{3-11}$$

where $\|\cdot\|$ is the 2-norm and $\mathbf{f} - \hat{\mathbf{f}}$ is the error. For the optimisation of program (3-11) the Matlab solver `Multistart` and `fmincon` set to the SQP algorithm was used. We introduce artificial bounds $0 < \theta_i < 400$, for all i, and generate 10 random starting points with uniform distribution between the artificial bounds. The computer specifications are in B.

We are going to compare different spline orders and increase the number basis functions. If the number of basis functions is n and the spline order is p, then the following knot sequence is used to generate the B-splines:

$$t = (\underbrace{0, \ldots, 0}_{p \text{ times}}, \frac{100}{n-p+1}, \frac{2 \cdot 100}{n-p+1}, \ldots, \frac{(n-p)100}{n-p+1}, \underbrace{100, \ldots, 100,}_{p \text{ times}}). \tag{3-12}$$

Optimisation results for third, fifth and n-th order splines are shown in Table 3-3.

Note that the n-th order B-splines with n basis functions are Bernstein basis polynomials and these approximations are full polynomials on $[0, 100]$. The third-order and fifth-order B-spline approximations are second-degree and fourth-degree piecewise polynomials. For the Runge function, using the third order B-splines gives the best fitting and fastest convergence. We do see that going from 13 to 15 basis functions gives a lesser fitting. That is because of the way the knot sequence is defined. When the number of knots is increased, the locations of the knots also changes and all the basis functions will be different.

Approximations of the best fits of each order are plotted in Figure 3-7. We notice that the 14-degree polynomial approximations oscillates close to the edges. This might be because these n-th order

Table 3-3: Global optimisation results of fitting piecewise polynomial approximations with varying spline orders and number of basis functions. The best local optimisation run has a starting error of size $\|\mathbf{f}-\hat{\mathbf{f}}_{\mathbf{0}}\|$ and a final error of size $\|\mathbf{f}-\hat{\mathbf{f}}\|$.

B-Spline order	# Basis functions n	$\|\mathbf{f}-\hat{\mathbf{f}}_{\mathbf{0}}\|$	$\|\mathbf{f}-\hat{\mathbf{f}}\|$	Run time (s)
3	5	528.5	172.4	43.04
	7	401.0	116.3	94.35
	9	326.0	53.81	115.4
	11	483.6	12.97	132.2
	13	462.4	7.453	163.3
	15	482.1	11.15	207.4
5	5	426.7	181.8	47.68
	7	375.8	146.5	83.64
	9	314.5	107.9	139.1
	11	413.8	58.06	206.9
	13	437.7	13.72	262.7
	15	513.8	7.42	335.6
n	5	284.8	181.8	74.12
	7	338.0	145.1	110.8
	9	445.9	114.5	199.5
	11	461.1	94.24	242.2
	13	394.0	84.1	369.0
	15	467.8	72.83	433.8

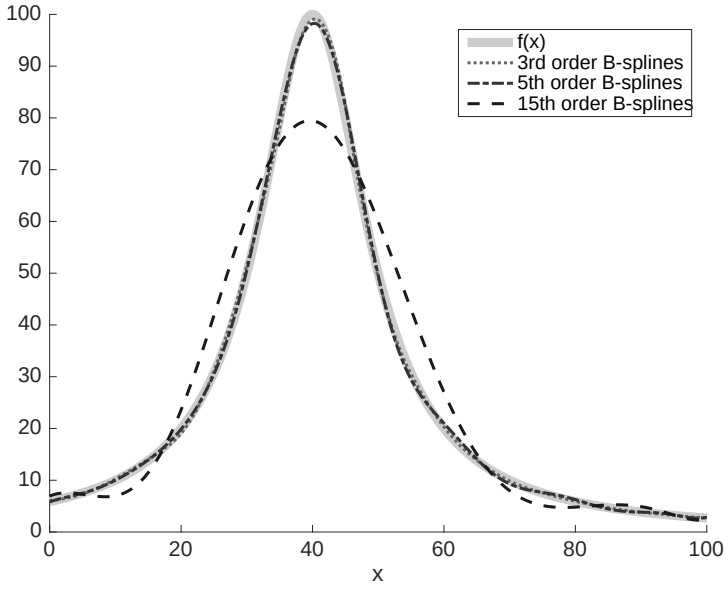


Figure 3-7: Plots of the best B-spline fits. The 3rd order B-splines uses 13 basisfunctions and the other two 15 basis functions.

basis functions have full support on $[0,100]$. We want to avoid oscillations, because using them as reward functions can cause multiple local maxima in the agents' utility function. To approximate functions that do not resemble polynomials, it might be better to use lower order B-splines that have support on smaller intervals.

We also did a test with the Gaussian basis functions (3-8). In case of n basis functions, the centres are placed at $x_1 = 0$ and $x_n = 100$ and subsequent centres x_i and x_{i+1} are placed equidistant $x_{i+1} - x_i = d$ apart. To select the width, we set $\epsilon = \frac{2\sqrt{\ln(2)}}{d}$. This ensures that subsequent basis functions are not too far apart, otherwise we get intervals in $[0,100]$ that are almost zero. A plot of 7 basis functions is shown in Figure 3-3.

Optimisation results are shown in Table 3-4. For a lower number of basis functions, the Gaussian is slightly better than using B-splines. Compared to the lower order B-splines it converges more slowly when the number of basis functions is increased.

The difficulty with Gaussian basis functions or other radial basis functions, is selecting the width and centre of each basis function. If not selected well, oscillations can occur as shown in Figure 3-8. We could turn the centre and width into optimisation parameters, but this would increase the complexity.

3-6 A Principal-Agent Problem with a General Reward Function

In this section we solve a moral hazard principal agent problem using general reward functions. The outcome $x \in \mathscr{X}$ is deterministic, and it is equal to the agent's action, so $\mathscr{X} = \mathscr{A} = [0,100]$ and

Table 3-4: Optimisation results of fitting with Gaussian basis functions. The objective function value with the starting point is $\|\mathbf{f}-\hat{\mathbf{f}}_\mathbf{0}\|$, and the objective function value with the final solution is $\|\mathbf{f}-\hat{\mathbf{f}}\|$.

# Basis functions n	$\|\mathbf{f}-\hat{\mathbf{f}}_\mathbf{0}\|$	$\|\mathbf{f}-\hat{\mathbf{f}}\|$	Run time (s)
5	402.0	163.4	8.819
7	392.0	103.0	23.94
9	495.6	43.6	37.83
11	277.6	23.62	67.03
13	417.3	27.33	79.56
15	478.3	25.79	104.9

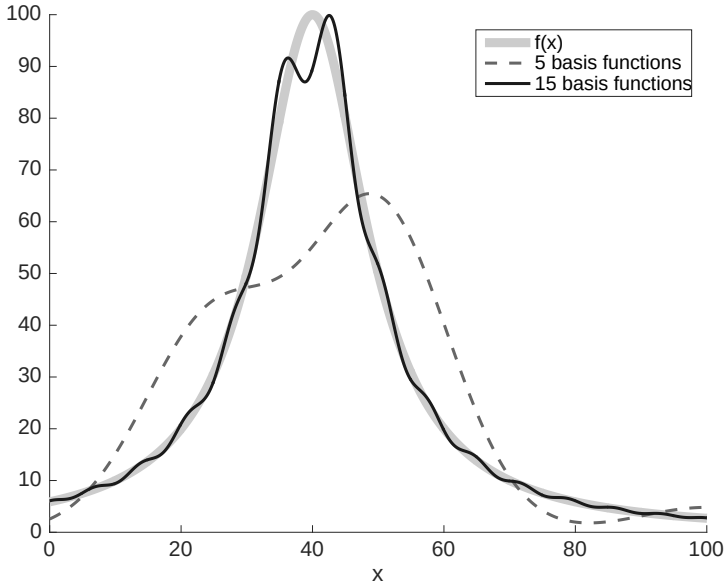


Figure 3-8: Fitting with Gaussian basis functions.

$x = a \in \mathscr{A}$. The whole optimisation problem is:

$$\max_{a\in A, w(.)} \; V_\mathrm{P} = 50\ln(1+a) - w(a) \tag{3-13a}$$

$$\text{s.t.} \;\; V_\mathrm{A} = 20\ln(1+w(a)) - 0.01a^2 \geq v_0 = 30 \tag{3-13b}$$

$$w(a) \geq 0 \tag{3-13c}$$

$$a \in \arg\max_{\hat{a}\in A} \; 20\ln(1+w(\hat{a})) - 0.01\hat{a}^2. \tag{3-13d}$$

Constraint (3-13b) is the participation constraint and it ensures that the agent receives at least reservation utility v_0. We impose (3-13c), so all payments are positive. Negative payments can be interpreted as fines, and we do not consider those kind payments in this example.

To turn (3-13) into a parametric optimisation problem, the reward $w(\cdot)$ is approximated by:

$$w(\cdot) = \sum_{i=1}^{n} \theta_i \phi_{i,k}(\cdot),$$

Table 3-5: Principal's utility of the best 3 and worst 3 local solutions. Solutions are indexed from best to worst

solution	Initial V_{P0}	Final V_{P^*}	V_A^*	a^*
1	-50.0	182.2	30.00	52.56
2	-981.6	124.0	81.10	75.37
3	-1577.0	59.03	86.79	19.11
47	-1495.0	-1092.0	141.0	9.829
48	-1289.0	-1165.0	139.1	21.51
50	-1315.0	-1314.0	141.9	19.91

Table 3-6: Starting points of the 3 best solutions. Solutions are indexed from best to worst

solution	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	50.0	50.0	50.0	50.0	50.0	50.0
2	981.6	13.12	884.2	1340.0	1957.0	585.5
3	1577.0	109.5	339.8	469.3	1226.0	846.2

with $\phi_{i,k}(\cdot)$ a k-th order B-spline generated by the knot sequence (3-12).

For the first experiment, the number of basis functions is $n = 6$ and the spline order is $k = 3$. For the optimisation procedure we use the Matlab solvers `Multistart` with `fmincon` set to the SQP algorithm. The number of starting points is 50 and we manually specify one of the starting points to be $\theta_0 = (50, 50, 50, 50, 50, 50)$. In this case, the reward $w(\cdot)$ is a constant payment that surely satisfies the participation constraint (3-13b).

The computation time is 5025 seconds. The principal's utilities for the best 3 and worst 3 local solutions are shown in Table 3-5. We notice a big difference between the worst and best solutions. As shown by the third solution, the objective function value of the initial starting point is not always indicative of the final objective function value. The starting points of the 3 best solutions are shown in Table 3-6. Our own guess turns out to be the best initial point, which converges to the solution

$$\theta^* = (2.03, 1.27, 1.07, 27.49, 24.91, 22.06). \tag{3-14}$$

In this case, the agent's optimal effort is $a^* = 52.56$ and his utility is exactly his reservation utility of 30.

Next, we use the matlab solver `GlobalSearch` instead of `Multistart`. Since we have an idea in what order of magnitude the best solution might be, we will set the initial point to the best known solution (3-14) and `GlobalSearch` will likely look in that region for better starting points. The solver `GlobalSearch` needs an artificial upper bound and that is set to 200.

The solver evaluated 26 more starting points and it turns out that solution (3-14) is again the optimal solution. The computation time is 2065 seconds. The utilities for the best 3 and worst 3 solutions are shown in Table 3-7. The difference between the best and worst values is much smaller now. The top 5 best reward functions are plotted in Figure 3-9. We can see that close to the optimal effort $a^* = 52.52$ there is hardly any difference between the reward functions. There is more variation further away from $a^* = 52.52$, but that does not affect the principal's utility, if it does not change optimal effort and reward to the agent.

Table 3-7: Principal's and agent's utility of the best 3 and worst 3 local solutions. Solutions are indexed from best to worst

solution	Initial V_{P0}	Final V_{P^*}	V_A^*	a^*
1	182.2	182.2	30.0	52.52
2	182.2	182.2	30.0	52.52
3	182.2	182.2	30.0	52.56
23	154.0	154.7	48.66	52.56
24	154.2	154.2	48.99	52.35
26	154.0	154.0	49.08	52.35

The principal's and agent's utilities are plotted in Figure 3-10 and we can see in the plot that the solution that the agent's effort is indeed the maxima of his utility function.

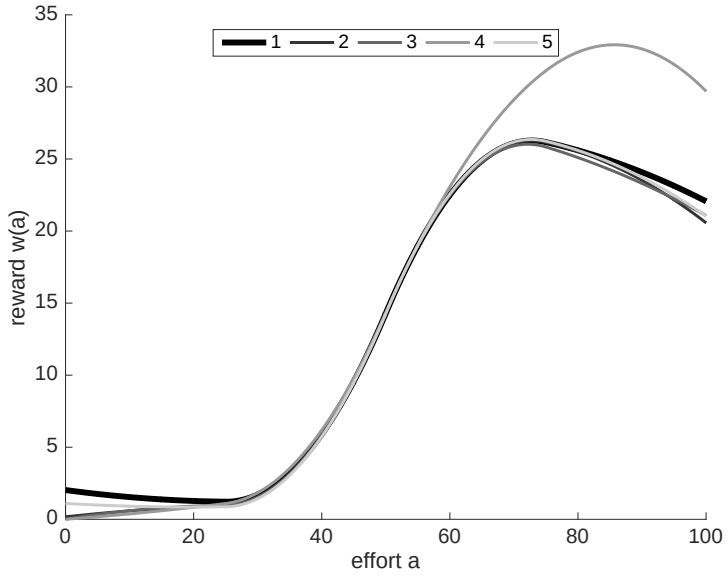


Figure 3-9: Plot of the top five best utility function.

Next, we perform optimisations with 4 and 10 basis functions. We again use the Matlab solver `GlobalSearch` with the SQP algorithm as local solver. We can specify one of the starting points and we select the best solution from the optimisation with 6 basis functions. To use the previous solution as starting point with different basis functions, we do a fitting like in section 3-5-3 to find an approximation as close as possible.

Results of the three best solution for each number of basis functions are shown in Table 3-8. We can see that using 4 basis functions is slightly worst than 6. Going from 6 to 10 does not yield an improvement in the solution.

Figure 3-11 shows a plot of the best solution for 4, 6 and 10 basis functions. The solution with 4 basis functions is quite different from 6 and 10. Solutions 6 and 10 are very similar in the region around the optimal effort $a^* = 52.51$. Although they behave differently further away from $a^* = 52.51$, the principal will get the same utility, because the agent's optimal effort and reward is the

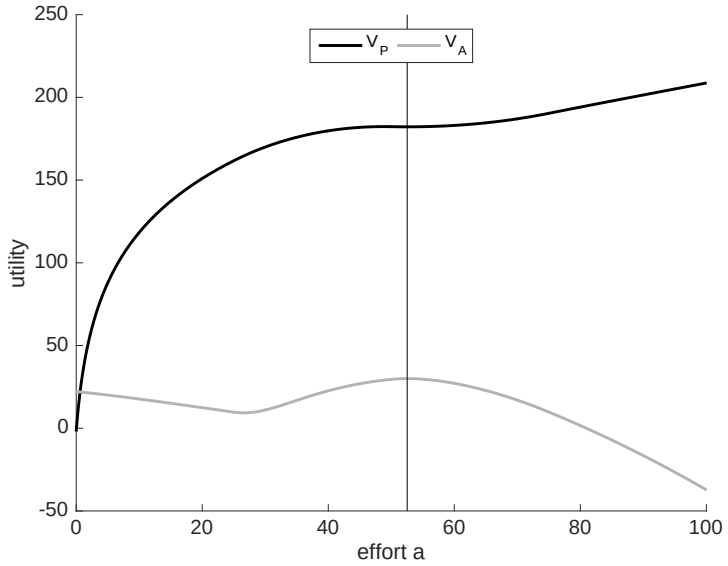


Figure 3-10: The principal's utility V_P and agent's utility V_A as function of the effort a. The vertical line indicates is the agent's best response $a^* = 52.52$.

same.

Table 3-8: The best three local solutions for 4, 6 and 10 basis functions.

#Basis Functions	Solution	Initial V_{P0}	Final V_{P^*}	V_A^*	a^*
4	1	162.6	178.7	30.0	42.66
	2	160.7	178.0	30.1	41.77
	3	162.6	176.0	30.6	39.57
6	1	182.2	182.2	30.0	52.52
	2	182.2	182.2	30.0	52.52
	3	182.2	182.2	30.0	52.56
10	1	182.2	182.2	30.0	52.51
	2	182.2	182.2	30.0	52.51
	3	182.2	182.2	30.0	52.51

3-7 Conclusion

In this chapter we introduce the moral hazard principal-agent problem. The moral hazard problem is a bi-level optimisation problem, where the lower-level program is possibly non-convex. We consider the agent's optimal action as a function that is implicitly defined by the numerical solution of the lower-level program, which depends on the contract parameters. This way, the bi-level problem transforms into a standard single level non-linear programming problem, which is solved by the Matlab SQP solver. We tested this method on an option contract problem and it is successfully solved in around a minute.

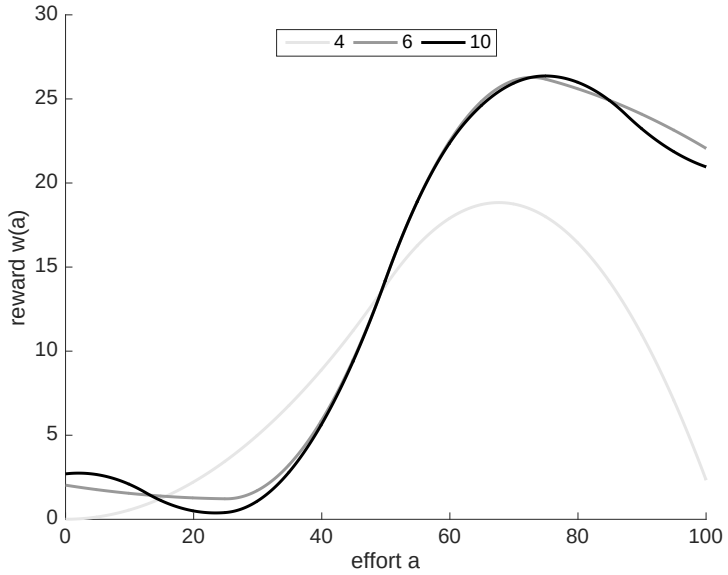


Figure 3-11: The best reward functions for 4, 6 and 10 basis functions

Next, we solve a moral hazard problem, without a structure imposed on the reward function. We approximate the reward function with a basis function approximation and third order B-spline basis functions. The program successfully converges to solution.

We have showed that these two particular problems can be numerically solved in a reasonable amount of time, and we think that this is true for most real world moral hazard problems. It would be interesting to test other problems in future research.

Chapter 4

Moral Hazard and Adverse Selection

In this chapter we add adverse selection to the moral hazard principal-agent problem. We will call this the *generalised principal-agent problem*. If there is a moral hazard, then the principal cannot control the agent's actions after the contract is signed. In case of adverse selection, the agent possesses relevant hidden information before a contract is signed. This hidden information is called the agent's *type*.

4-1 The Contracting Game

The principal-agent model in this chapter is loosely based of the one in [18]. We start by introducing the following definitions:

- The *type space* $\mathscr{T} = \left[\underline{t}, \overline{t}\right]$ is an interval in $\mathbb{R}$ and the *type* T is random variable [1] taking values in $\mathscr{T}$, with probability density $\pi(\cdot)$.
- The *action space* $\mathscr{A} = \left[\underline{a}, \overline{a}\right]$ is an interval in $\mathbb{R}$.
- We have an *outcome space* $\mathscr{X} \subseteq \mathbb{R}$ and the *outcome* X is a random variable taking values in $\mathscr{X}$, with probability density $p(\cdot|a,t)$, where $a \in \mathscr{A}$ is the agent's action and $t \in \mathscr{T}$ is the agent's true type.
- The *recommended action* $\alpha : \mathscr{T} \to \mathscr{A}$ is the action that the principal recommends to the agent and it depends on the agent's reported type.
- The *reward function* is $w : \mathscr{X} \times \mathscr{T} \to \mathbb{R}$, that depends on the outcome and agent's reported type.

[1] In this report, there is a notational difference between random variables and realisations of random variables. A random variable X is denoted with a capital letter. A realisation of X is a particular outcome of the random variable and is denoted with a small letter, for example x.

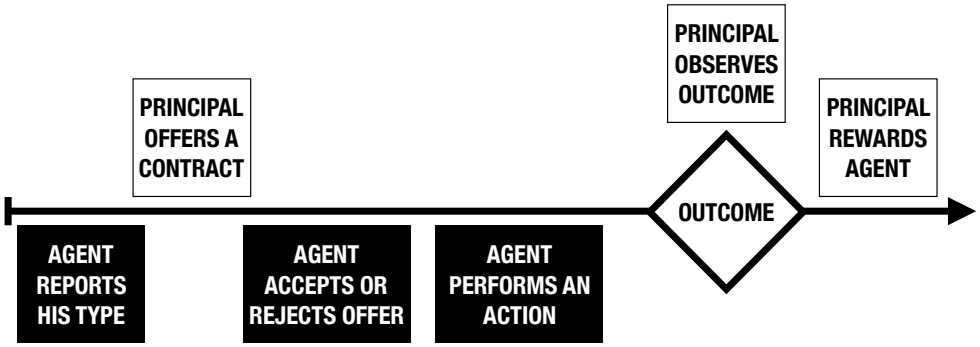


Figure 4-1: Time line of the contracting game with moral hazard and adverse selection.

- The *contract* is $\left(\alpha\left(t'\right), w\left(x, t'\right)\right)$, where $t' \in \mathscr{T}$ is the agent reported type and $x \in \mathscr{X}$ is a random outcome.

The actions and time line of the contracting game is shown in Figure 4-1 and it goes as follows:

1. Nature randomly assigns a type $t \in \mathscr{T}$ to the agent.
2. The agent, with type $t \in \mathscr{T}$, reports a type $t' \in \mathscr{T}$ to the principal.
3. The principal offers a contract $\left(\alpha\left(t'\right), w\left(x, t'\right)\right)$ to the agent.
4. The agent will either accept or reject the offer.
5. If the offer is accepted, the agent will perform some action $a \in \mathscr{A}$.
6. There is a random outcome $x \in \mathscr{X}$ conditionally dependent on the agents action a and his true type t.
7. The principal observes the outcome x and rewards the agent $w\left(x, t'\right)$.

Note that because of moral hazard and adverse selection, the reported type t' does not need to be equal to the agent's true type and the agent's action a, does not need to be equal to the recommendation $\alpha(t')$. If the agent always reports $t = t'$ we say that the agent is *honest*, and if the agent always does $a = \alpha(t')$, then the agent is *obedient*. The contracting game here is said to have a *direct mechanism*, because the agent directly reports his type t' and receives a recommendation $\alpha\left(t'\right)$. The contract is *incentive compatible* if the agent maximises his expected utility by being honest and obedient.

If the only goal is to maximise the principal's expected utility, then it might not be apparent why incentive compatibility is relevant. By the revelation principle [3], it turns out that for any contract that maximises the principal's expected utility, there exists an incentive compatible contract with the same expected utility for the principal. By requiring an incentive compatibility constraint, the agent's behaviour can easily be predicted without needing to consider all kind of reporting strategies.

At the end of the contracting game, an agent of type t will have an utility v_t and the principal has a utility v_{P}:

$$\begin{aligned} v_t\left(a, w, x, t'\right) &= u\left(w\left(x, t'\right)\right) - h(a, t) \\ v_{\mathrm{P}}\left(a, w, x, t'\right) &= x - w\left(x, t'\right), \end{aligned}$$

where we assume that $\frac{\partial h}{\partial a} \geq 0$, $u'(.) \geq 0$ and $u''(.) \leq 0$. Note that the agent's utility goes down in the action a, so a higher action is modelled as bringing more dis-utility to the agent. The conditions on u have to do with the fact that we assume that the agent is risk averse.

Since the agents true type t and the outcome x are realisations of random variables, we have to maximise expected utilities. An agent with true type t has an expected utility

$$V_t\left(a, w, t'\right) = \int_{\mathscr{X}} u\left(w\left(x, t'\right)\right) p\left(x|a, t\right)\ dx - h(a, t). \tag{4-1}$$

The agent's reservation utility is v_0 and it is the utility he gets if he declines the contract. In step 4 of the contracting game, the agent will only accept the offer if $V_t \geq v_0$. If the agent is honest and obedient, then the principal's expected utility is

$$V_{\mathrm{P}}(\alpha, w) = \int_{\mathscr{T}} \left[\int_{\mathscr{X}} x - w(x, t) p(x|\alpha(t), t)\ dx \right] \pi(t)\ dt. \tag{4-2}$$

4-2 The Optimisation Problem

We start of by making an assumption that the reward function w is structured as

$$w(x, t) = w\left(x, c(t)\right),$$

where $c : \mathscr{T} \to \mathscr{C} \subseteq \mathbb{R}^m$ and $\mathscr{C}$ is some parameter set. An example is $c(t) = (c_1(t), c_2(t))$ and $w(x, c(t)) = c_1(t)x + c_2(t)$.

Now, instead of finding the optimal contract $(\alpha(.), w(.,.))$, we want to find the optimal $(\alpha(.), c(.))$. The agent's and principal's expected utility only depends on $(\alpha(.), c(.))$, i.e.

$$\begin{aligned} &V_{\mathrm{P}}(\alpha(\cdot), c(\cdot)) \\ &V_{\mathrm{A}}(\alpha(\cdot), c(\cdot)). \end{aligned}$$

We want to find an incentive compatible contract $(\alpha(.), c(.))$ that maximises the principal's expected utility V_{P}. The optimisation problem is:

$$\max_{\alpha(.), c(.)} V_{\mathrm{P}}(\alpha(\cdot), c(\cdot)) \tag{4-3a}$$

$$\text{s.t. } V_t(\alpha(t), c(t)) \geq v_0 \tag{4-3b}$$

$$V_t(\alpha(t), c(t)) \geq V_t\left(a', c\left(t'\right)\right) \tag{4-3c}$$

$$\forall\, t, t' \in \mathscr{T}, \quad \forall\, a' \in \mathscr{A}$$

Equation (4-3b) is the participation constraint and if that is satisfied, then the agent will surely sign the contract. Equation (4-3c) is the incentive compatibility constrained and it ensures that the agent would report his true type and select the action recommended by the principal. The optimisation problem (4-3) is an infinite program, because it has an infinite number of constraints and optimisation parameters.

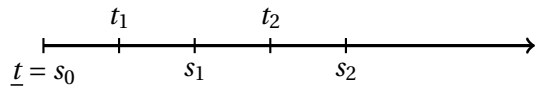


Figure 4-2: An illustration on how the type space is discretised.

4-3 Numerical Solution Methods

Numerical solution methods to solve (4-3) has to deal with the fact that the program has infinite many constraints and optimisation parameters. In this section we will consider two methods: the discretisation- and the basisfunction method. We will then try out and evaluate both methods on a test problem.

4-3-1 Discretisation

In the discretisation method, the continuous type space $\mathscr{T}$ is discretised into

$$\mathscr{T}_{\mathrm{d}} = \{t_1, \ldots, t_n\} \subset \mathscr{T}.$$

We only consider a finite number of n types and each agent type is indexed by indices in the set $i \in I_{\mathrm{d}} = \{1, \ldots, n\}$. To set the values of t_i, first define a grid $\{s_0, \ldots, s_n\}$, where $\underline{t} = s_0 < s_1 < \ldots < s_n = \overline{t}$. Each value t_i is then $t_i = (s_{i-1} + s_i)/2$ and the probability that an agent is of type t_i is $\pi_i = \int_{s_{i-1}}^{s_i} \pi(t)\,dt$. Figure 4-2 illustrates how the discretisation looks like.

For simplicity, we use the abbreviation $V_i = V_{t_i}$ and $c^i = c(t_i)$. Because c^i can be a vector, we put the index i, which refers to agent i, in the superscript of c, so i here is not a power. If an agent is of type i, reports j, and receives contract with parameters c^j, then his action $\alpha_i(j)$ is the solution of:

$$\alpha_i(j) \in \arg\max_{a \in \mathscr{A}} V_i\left(a, c^j\right). \tag{4-4}$$

Define $\boldsymbol{c} = \left(c^1, \ldots, c^n\right)$. If all agents i truthfully report their types, then the principal's expected utility for the discretised type space $\mathscr{T}_{\mathrm{d}}$ is:

$$V_{\mathrm{Pd}}(\alpha, \boldsymbol{c}) = \sum_{i=1}^{n} \left[\int_{\mathscr{X}} \left(x - w\left(x, c^i\right)\right) p\left(x | \alpha_i(i), t_i\right)\, dx \right] \pi_i.$$

By discretising the type space, we approximate the solution of (4-3), with a program that has finite number of constraints. The discretisation algorithm goes as follows;

1. Solve the non-linear program with a finite number of constraints and optimisation variables.

$$\max_{\boldsymbol{c}} V_{\mathrm{Pd}}(a, \boldsymbol{c}) \tag{4-5a}$$

$$\text{s.t. } V_i\left(\alpha_i(i), c^i\right) \geq v_0 \tag{4-5b}$$

$$V_i\left(\alpha_i(i), c^i\right) \geq V_i\left(\alpha_i(j), c^j\right) \tag{4-5c}$$

$$\forall i, j \in I_{\mathrm{d}}$$

2. Interpolate $\boldsymbol{c}$ over $\mathscr{T}$ to get the function $c_{\text{ip}}(\cdot)$.

3. Let ϵ_1 and ϵ_2 be two (small) positive constants. Take $N > n$ and construct a finer grid $\{\tilde{t}_1, \ldots, \tilde{t}_N\} = \mathscr{T}_{\text{d2}} \subset \mathscr{T}$ and check whether

$$V_i\left(\alpha_i(i), c^i_{\text{ip}}\right) + \epsilon_1 \geq v_0$$
$$V_i\left(\alpha_i(i), c^i_{\text{ip}}\right) + \epsilon_2 \geq V_i\left(\alpha_i(j), c^j_{\text{ip}}\right)$$
$$\forall i, j \in I_{\text{d2}}$$

If yes, then stop. Otherwise, go to step 1 with a finer grid for $\mathscr{T}_{\text{d}}$.

Regarding complexity, if $c(t)$ is a vector of size m, then $\boldsymbol{c}$ is a vector of size nm, so the number of optimisation parameters is nm; The total number of constraints is $n + n(n-1)$, where n is the number of participation constraints (4-5b) and $n(n-1)$ is the number of incentive compatibility constraints (4-5c). The number of constraints grows quadratically in n, and to check each constraint, we have to perform a global optimisation of (4-4). This can lead to long computational time, if n is large.

4-3-2 Basis Function Method

Reformulating the Optimisation problem

With the basis function method we approximate each vector entry $c_i(t)$ of $c(t)$ as a sum of finite many basis functions:

$$c_i(\cdot) = \sum_{j=1}^{n_i} \theta_{ij}\phi_{ij}(\cdot),$$

where θ_{ij} is the basis function coefficient of the basis function $\phi_{ij}(\cdot)$. With an appropriate selection of basis functions, any function c_i can be approximated with arbitrary precision by increasing the number of basis functions. Instead of solving a functional optimisation problem for $c(\cdot)$, we can use a basis function approximation and perform parameter optimisation for θ_{ij}. We write c_θ as the vector of basis function approximations of $(c_1, \ldots, c_m)$, with basis function coefficients:

$$\left\{\theta_{ij} | i = 1, \ldots, m; j = 1, \ldots, n_i\right\}.$$

It is also possible to optimise for parameters in the basis functions $\phi_{ij}(\cdot)$. In that case, we not only optimise the weight of each basis functions, but also the shape.

Next, we rewrite (4-4) as a tri-level optimisation problem. First, the effort $\alpha(t)$ is considered as an implicit function of t, and the value is the solution of:

$$\alpha_\theta(t) = \arg\max_{a \in \mathscr{A}} V_t(a, c_\theta(t)).$$

Since the participation constraint (4-3b) has to hold for all types, it also has to hold for the type where the agent's expected utility is the lowest. So constraint (4-3b) can be written as:

$$v_0 - V_{\hat{t}}\left(\hat{a}, c_\theta\left(\hat{t}\right)\right) \leq 0 \tag{4-6a}$$
$$\hat{t} \in \arg\min_{\hat{\tau}} V_{\hat{\tau}}\left(\hat{a}, c_\theta\left(\hat{\tau}\right)\right) \tag{4-6b}$$
$$\text{s.t. } \hat{a} \in \arg\max_{\hat{\alpha}} V_{\hat{\tau}}\left(\hat{\alpha}, c_\theta\left(\hat{\tau}\right)\right). \tag{4-6c}$$

The whole formulation (4-6) is abbreviated as:

$$g_{\text{par}}(\theta) \leq 0, \tag{4-7}$$

where $g_{\text{par}}(\theta) = v_0 - V_{\hat{t}}\left(\hat{a}, c_\theta\left(\hat{t}\right)\right)$, and $\left(\hat{a}, \hat{t}\right)$ satisfy (4-6b) and (4-6c).

For the incentive compatibility constraint (4-3c), define a function $z : \mathscr{T} \times \mathscr{A} \times \mathscr{T} \times \mathscr{A} \to \mathbb{R}$, such that:

$$z_\theta\left(t, a, t', a'\right) = V_t\left(a', c_\theta\left(t'\right)\right) - V_t(a, c_\theta(t)).$$

Constraint (4-3c) can be rewritten as:

$$z_\theta\left(t, \tilde{a}, t', \tilde{a}'\right) \leq 0 \tag{4-8a}$$
$$\left(t, t'\right) \in \arg\max_{(\tau,\tau') \in \mathscr{T}^2} z_\theta\left(\tau, \tilde{a}, \tau', \tilde{a}'\right) \tag{4-8b}$$
$$\text{s.t. } \tilde{a} \in \arg\max_{\tilde{\alpha} \in \mathscr{A}} V_\tau\left(\tilde{\alpha}, c_\theta\left(\tau\right)\right) \tag{4-8c}$$
$$\tilde{a}' \in \arg\max_{\tilde{\alpha}' \in \mathscr{A}} V_\tau\left(\tilde{\alpha}', c_\theta\left(\tau'\right)\right). \tag{4-8d}$$

For an agent t, the action $\tilde{a}$ is the optimal action if the agent truthfully reports his type t and $\tilde{a}'$ is the optimal action if the agent reports a type t'. The value $z_\theta\left(t, \tilde{a}, t', \tilde{a}'\right)$ is the maximum difference of V_t when an agent is truthful or untruthful. The whole formulation (4-8) is abbreviated as:

$$g_{\text{inc}}(\theta) \leq 0, \tag{4-9}$$

where $g_{\text{inc}}(\theta) = z_\theta\left(t, \tilde{a}, t', \tilde{a}'\right)$, and $\left(t, \tilde{a}, t', \tilde{a}'\right)$ satisfy (4-8b), (4-8c) and (4-8d).

The principal-agent problem with basis function approximation and the alternative constraint formulations (4-7) and (4-9) is:

$$\min_\theta V_{\text{P}}(\theta) \tag{4-10a}$$
$$\text{s.t. } g_{\text{par}}(\theta) \leq 0 \tag{4-10b}$$
$$g_{\text{inc}}(\theta) \leq 0 \tag{4-10c}$$

Gradient methods on tri-level optimisation problems

The tri-level optimisation problem (4-10) is of the form

$$\min_{\theta,x,y} f^1(\theta, x, y) \tag{4-11a}$$
$$\text{s.t. } g^1(\theta, x, y) \leq 0 \tag{4-11b}$$
$$x \in \arg\min_{\hat{x}} f^2\left(c, \hat{x}, y\right) \tag{4-11c}$$
$$\text{s.t. } g^2\left(\theta, \hat{x}, y\right) \leq 0. \tag{4-11d}$$
$$y \in \arg\min_{\hat{y}} f^3(\theta, \hat{x}, \hat{y}) \tag{4-11e}$$

Here, the numbers i on f^i and g^i are indices and not powers.

As described in section 3-3, we already know how to apply gradient methods to bi-level optimisation problems. So we can assume that for some fixed c, we can find the solution of

$$\min_{x,y} f^2(c,x,y) \tag{4-12a}$$

$$\text{s.t. } g^2(c,\hat{x},y) \leq 0. \tag{4-12b}$$

$$y \in \arg\min_{\hat{y}} f^3(c,\hat{x},\hat{y}) \tag{4-12c}$$

If (x,y) are unique solutions of (4-12) for every θ, then (x,y) can be considered as implicit functions of θ. Define:

$$\theta_{i,+h} = (\theta_1,\ldots,\theta_{i-1},\theta_i+h,\theta_{i+1},\ldots,\theta_n)$$
$$\theta_{i,-h} = (\theta_1,\ldots,\theta_{i-1},\theta_i-h,\theta_{i+1},\ldots,\theta_n).$$

The gradient $\nabla_\theta f^1(\theta,x,y)$ can be approximated by the formula

$$\frac{\partial f^1(\theta,x,y)}{\partial \theta_i} \approx \frac{f^1(\theta_{i,+h},x_{i,+h},y_{i,+h}) - f^1(\theta_{i,-h},x_{i,-h},y_{i,-h})}{2h}, \tag{4-13}$$

where $(x_{i,+h},y_{i,+h})$ and $(x_{i,-h},y_{i,-h})$ are the solutions of (4-12) with the parameter vectors $\theta_{i,+h}$ and $\theta_{i,-h}$ respectively. With the gradient of f^1, we can use gradient based non-linear optimisation algorithms like the SQP-algorithm.

4-4 Example: Option Contract with Moral Hazard and Adverse Selection

In this section we modify the problem in Section 3-4 to include adverse selection. We then apply the discretisation method and basis function method to test the algorithms and compare the results.

Same as in 3-4, the outcome is the stock price, which is a random variable X over $\mathscr{X} = \mathbb{R}$. The outcome is normally distributed with mean $100 + a \in \mathscr{A}$ and variance σ^2, where a is the agent's effort and the σ^2 is the volatility. Let c_1 be the fixed wage, c_2 the number of stock options and K the strike price. If $x \in \mathscr{X}$ is a realisation of X then the principal rewards the agent the amount:

$$w(x) = c_1 + c_2 \max(x-K,0).$$

What is different from Section 3-4 is that the disutility is $h_{\text{MhAs}} = ta^2$ with $t \in \mathscr{T}$. If an agent t reports t' to the principal, then he receives the contract $(c_1(t'),c_2(t'),K(t'))$ and his expected utility is

$$\begin{aligned} V_t(t') = \mathbf{E}[v_t] &= c_1(t') + c_2(t')\,\mathbf{E}\left[\max\left(X-K(t'),0\right)\right] - t\alpha_t(t')^2. \\ &= c_1(t') + c_2(t')\left[\frac{\sigma}{\sqrt{2\pi}}e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y\left(1-F\left(-\mu_y/\sigma\right)\right)\right] - t\alpha_t(t')^2, \end{aligned}$$

Table 4-1: Constant parameters of the option contract problem.

Description	Symbol	Value
Type space	$\mathscr{T}$	$[90,200]$
Action space	$\mathscr{A}$	$[0,100]$
Outcome space	$\mathscr{X}$	$\mathbb{R}$
Participation utility	v_0	40000
Number of shares	L	50000
Volatility	σ^2	200
Probability density type space	$\pi(t)$	$\frac{1}{110}$

where $F(\cdot)$ is the cumulative distribution of the standard normal and $\mu_y = 100 + a - K\left(t'\right)$. For more details on the derivation, check Appendix A. The action $\alpha_t\left(t'\right)$ is an implicit function of $\left(t, t'\right)$ it is the solution of

$$\alpha_t\left(t'\right) \in \arg\max_{a \in \mathscr{A}} \left(c_1\left(t'\right) + c_2\left(t'\right) \left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y \left(1 - F\left(-\mu_y/\sigma\right)\right) \right] - t a^2 \right).$$

If an agent t truthfully reports type for all true t, then the principal's expected utility is

$$V_{\mathrm{P}}\left(\alpha(\cdot), c_1(\cdot), c_2(\cdot), K(\cdot)\right) = \int_{\mathscr{T}} \mathbf{E}\left[Lx - c_2(t)\max(x-K,0) - c_1(t)\right]\pi(t)\,dt,$$

where

$$\begin{aligned}\mathbf{E}\left[Lx - \beta\max(x-K,0) - c_1(t)\right] = & L(100 + \alpha_t(t)) - \\ & \beta(t)\left[\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y\left(1 - F\left(-\mu_y/\sigma\right)\right) \right] - c_1(t),\end{aligned}$$

$F(\cdot)$ is the cumulative distribution of the standard normal, $\mu_y = 100 + \alpha_t(t) - K$, and L is a fixed constant, that is the number of shares the principal owns at the beginning of the contracting game. The values of all fixed parameters in this example are shown Table 4-1.

Additionally, we introduce the following bound constraints;

$$0 \le c_1 \le v_0 \tag{4-14}$$

$$0 \le c_2 \le 0.49L. \tag{4-15}$$

Constraint (4-14) is imposed, because it is not needed to pay a guaranteed fixed wage that is more than the reservation utility. We have constraint (4-15), because the principal does not want to give more than 50 percent of her shares. For both constraints negative values do not make sense.

4-4-1 Scenario with only Moral Hazard

For comparison purposes we look at how much utility the principal gets for each agent type t, if there is no adverse selection. The type space is discretised into a really fine grid, and for each type the optimal moral hazard contract is computed. For the gobal optimisation, we use 50 random starting from an uniform distribution. For the non-linear solver, we use the Matlab solvers

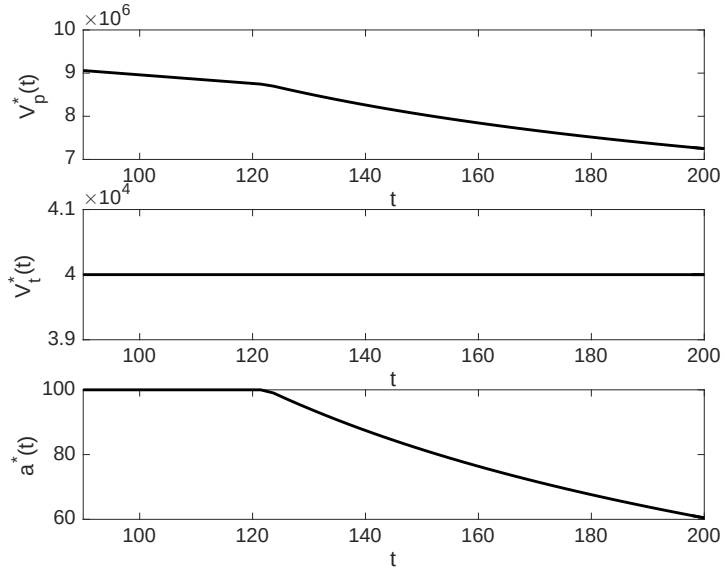


Figure 4-3: Solution of the discretised problem (circles) with the piece-wise linear interpolation when using 15 grid points.

`MulitStart` and `fmincon`, set to the SQP algorithm. The parallelisation functionality is turned on. The computer specifications are in Appendix B. The total run time is 42 minutes.

The expected utilities are plotted in Figure 4-3 and the contracts are shown in Figure 4-4. The principal will receive a higher utility if the agent is more efficient (lower t). Each agent gets exactly their reservation utility. Agents from $t = 90$ too around $t = 121$ will select their maximum effort of $\overline{a} = 100$. After $t = 121$, the agent's optimal effort decreases. Figure 4-5 compares the utility curves of an efficient and inefficient agent. We can see that the optimal effort of the inefficient agent is moved to the left. If the probability density of the agent's type is uniform, then the principal will receive a expected utility of $V_{\mathrm{p}} = 8.1787 \cdot 10^6$.

4-4-2 Applying the Discretisation Method

In this section, the discretisation method is applied. We will perform a global optimisation with multiple random starting points. The starting points are random variables with an uniform distribution over a closed interval. This means that in addition to the constraints (4-14) and (4-15), we impose the following (partly) artificial constraint for K:

$$0 \leq K \leq 200.$$

The lower bound is required because we cannot have negative strike prices. The upper bound for K can be increased, if that constraint turns out to be tight in the optimal solution. For the non-linear solver we use the Matlab solvers `MultiStart` and `fmincon`, set to the SQP-algorithm. The parallelization functionality is turned on. The specifications of the computer used can be found in Appendix B. After finding the discretised solution, we perform a piecewise linear interpolation. The maximum constraint violation of the interpolated solution is 100.

The number of starting points is 100. One starting point is a predefined input the rest are random variables, with an uniform distribution over the bounds. For the predefined starting point, we want

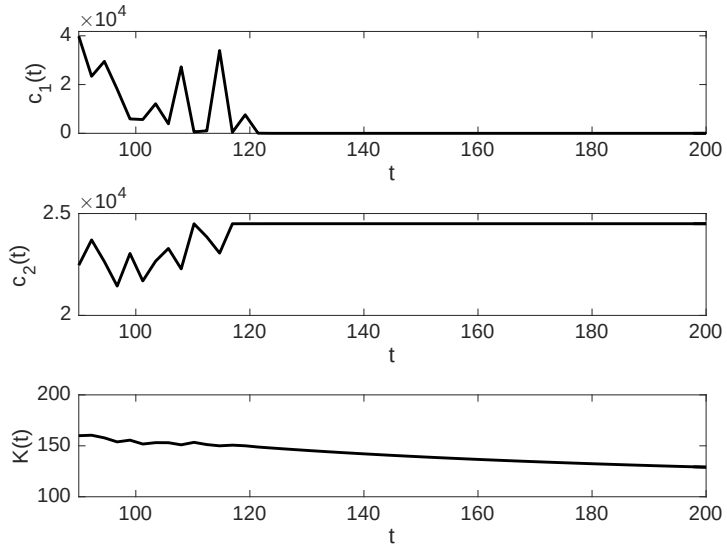


Figure 4-4: Solution of the discretised problem (circles) with the piece-wise linear interpolation when using 15 grid points.

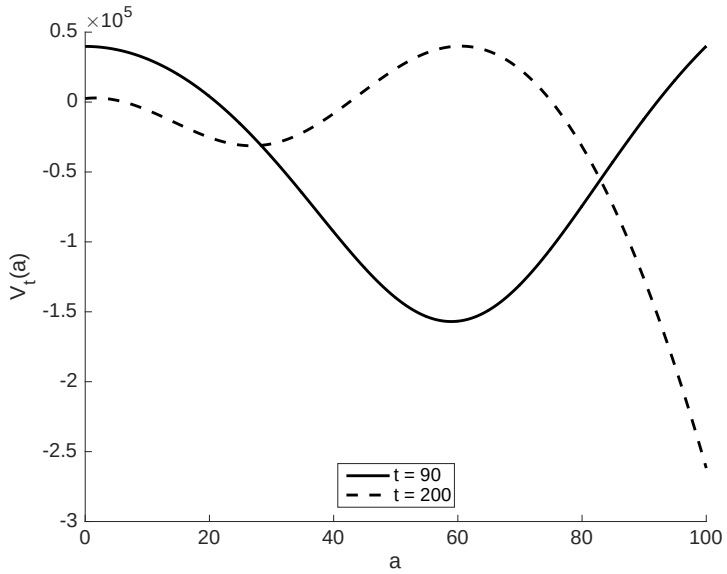


Figure 4-5: Example of utility curves for an efficient agent, $t = 90$, and inefficient agent $t = 200$.

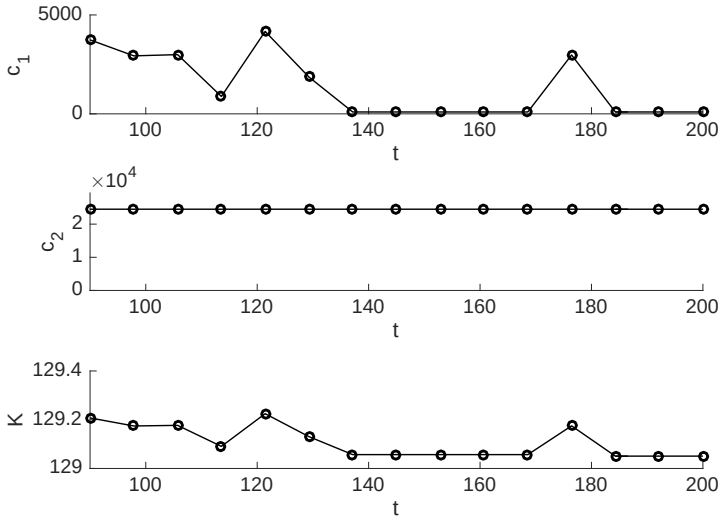


Figure 4-6: Plot of the interpolated optimal contract with 15 grid points. The discrete solution is represented by the circles.

to select a point that is feasible and not the trivial solution:

$$c_1(t) = 40000, \quad c_2(t) = 0, \quad K(t) = 0.$$

A possibility is:

$$c_1(t) = 0, \quad c_2(t) = 24500, \quad K(t) = 130.$$

It is easy to see that a contract that is constant over all t should satisfy the incentive compatibility constraint. To be sure that the contract also satisfies the participation constraints, we take the values similar to the contract given to the least efficient agent from the optimisation results in Section 4-4-1.

Optimisations are done for 5, 10 and 15 number of grid points. The top 10 best results for each number of grid points are presented in Table 4-2. In each group the results go from best to worst. The best solution is found with 15 grid points. Considering the maximum constraint violation of 100 for the interpolated solution, we see that it is more likely to find feasible solutions with a lower number of grid points. Note that the top 3 solutions from the 10 and 15 grid points do vary, so we should try more starting points, to have a higher confidence that the best local solution is also the global one.

The best contract is with 15 grid points and is plotted in Figure 4-6. The principal's utility, the agent's utility and the agent's optimal effort for each type is plotted in Figure 4-6. If we compare this to the utilities in Figure 4-4, then we see that the principal receives less utility and the agent gets more in case of adverse selection. The more efficient the agent is, the more he gets.

We did an additional run where the starting points are uniform and randomly selected from the moral hazard solution in section 4-4-1. The number of grid points are 5, 10 and 15, and for each number of grid points, we do a global optimisation with 30 random starting points. The solutions are not better than the 100 random points. Because of time constraints, we took less starting points.

Table 4-2: The top 10 best results from a global optimisation run with 100 random starting points. The principal's utility with the discretised and interpolated solutions are V_{pd}^* and V_{pi}^* respectively. The values $\|g_{\text{d}}\|_\infty$ and $\|g_{\text{i}}\|_\infty$ are the highest constraint function values for the discretised and interpolated solutions respectively.

Grid number	Run time [h]	$V_{\text{pd}}^*\,[10^6]$	$V_{\text{pi}}^*\,[10^6]$	$\|g_{\text{d}}\|_\infty$	$\|g_{\text{i}}\|_\infty$
5	10.9	7.8321	7.8541	1.18e-06	1.18e-06
		7.8321	7.8541	5.25e-07	7.22e-01
		7.8321	7.8541	4.34e-05	3.84e+00
		7.8321	7.8541	4.66e-10	7.81e+00
		7.8320	7.8540	6.75e-09	4.34e+01
		7.8319	7.8539	8.53e-06	5.35e+01
		7.8319	7.8539	6.98e-10	6.58e+01
		7.8318	7.8539	1.53e-03	7.89e+01
		7.8318	7.8539	1.57e-05	7.73e+01
		7.8318	7.8539	5.05e-05	6.98e+01
10	94.6	7.8427	7.8539	9.36e-04	3.07e+00
		7.8344	7.8487	2.38e-03	7.37e+02
		7.7848	7.7896	7.67e-05	1.12e+03
		7.7839	7.7892	8.52e-06	4.20e+02
		7.7839	7.7889	9.59e-06	6.66e+01
		7.7761	7.7818	5.02e-04	8.51e+02
		7.6146	7.6245	3.37e-02	1.94e+03
		7.5675	7.5594	8.94e-03	3.92e+02
		7.5047	7.4779	1.99e-04	8.24e+03
		7.4105	7.4315	3.16e+00	3.46e+05
15	253	7.8478	7.8525	3.49e-01	1.54e+00
		7.7774	7.7801	9.95e-04	1.83e+02
		7.7770	7.7798	5.09e-03	1.31e+03
		7.7690	7.7718	2.19e+01	2.17e+03
		7.5882	7.5841	7.98e-05	5.76e+02
		7.5877	7.5833	4.10e-04	3.88e+03
		7.5758	7.5770	1.98e-06	7.10e+03
		7.5193	7.5446	3.66e-04	3.33e+03
		6.9514	6.8905	1.12e-02	3.95e+04
		6.8425	6.9084	1.08e+02	3.26e+05

Table 4-3: The top 10 best results from a global optimisation run with 30 starting points. Note that for the case with 10 and 15 grid points, only 8 and respectively 2 feasible solutions were found by the solver. Starting points were randomly selected from the moral hazard solution. The principal's utility with the discretised and interpolated solutions are V_{pd}^* and V_{pi}^* respectively. The values $\|g_{\text{d}}\|_\infty$ and $\|g_{\text{i}}\|_\infty$ are the highest constraint function values for the discretised and interpolated solutions respectively.

Grid number	Run time [h]	$V_{\text{pd}}^*\,[10^6]$	$V_{\text{pi}}^*\,[10^6]$	$\|g_{\text{d}}\|_\infty$	$\|g_{\text{i}}\|_\infty$
5	6.37	7.8321	7.8541	1.13e-06	1.13e-06
		7.8320	7.8541	1.92e-02	2.08e+01
		7.8320	7.8541	1.93e-05	2.62e+01
		7.8320	7.8540	1.93e-06	2.74e+01
		7.8314	7.8540	2.35e-05	4.72e+01
		7.8067	7.8167	3.21e-08	8.51e+01
		7.7727	7.7854	-4.47e-07	5.76e+01
		7.6864	7.6946	1.28e-09	7.81e-05
		7.4675	7.3551	4.66e-10	3.84e+04
		7.3855	7.4476	1.05e-09	3.04e+04
10	28.4	7.8430	7.8541	4.10e-06	6.24e-05
		7.8426	7.8539	2.42e-01	5.12e+01
		7.8405	7.8524	5.90e-05	1.62e+02
		7.8328	7.8441	1.52e+00	1.12e+01
		7.7840	7.7889	1.35e-03	7.49e+02
		7.6313	7.6844	2.16e-01	1.63e+04
		6.9949	6.9065	4.76e-01	2.43e+04
		5.7556	5.5162	1.66e-01	3.03e+04
15	56.2	7.7219	7.7332	1.73e-01	1.40e+03
		5.7486	5.5836	4.98e-01	2.82e+04

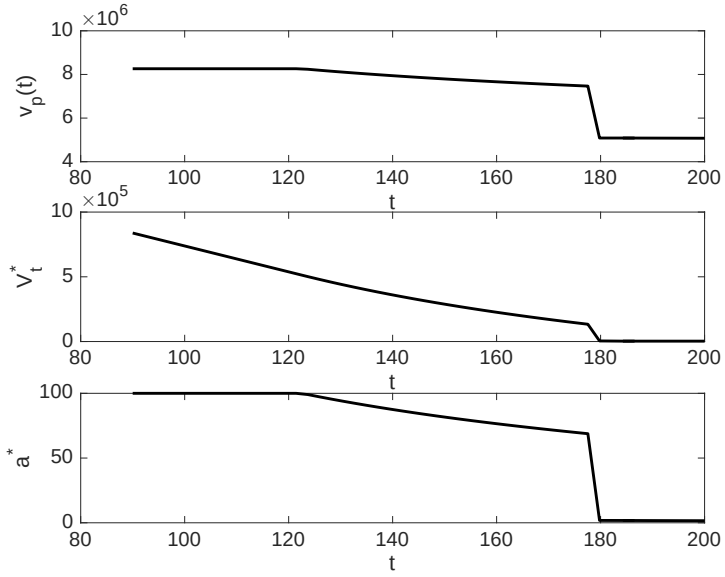


Figure 4-7: The principal's utility, the agent's utility, and the agent's optimal effort for each type.

4-4-3 Applying the Basis Function Method

With the gradient method we approximate each $\left(c_1\left(t'\right), c_2\left(t'\right), K\left(t'\right)\right)$ by a linear combination of B-splines. Advantages of B-spline approximation, is that it does not oscillate and we can perfectly construct straight lines.

Same as with the discretisation method, we use the Matlab solvers `MultiStart` and `fmincon`, set to the SQP-algorithm. Parallelization is turned on and the system specifications are in Appendix B.

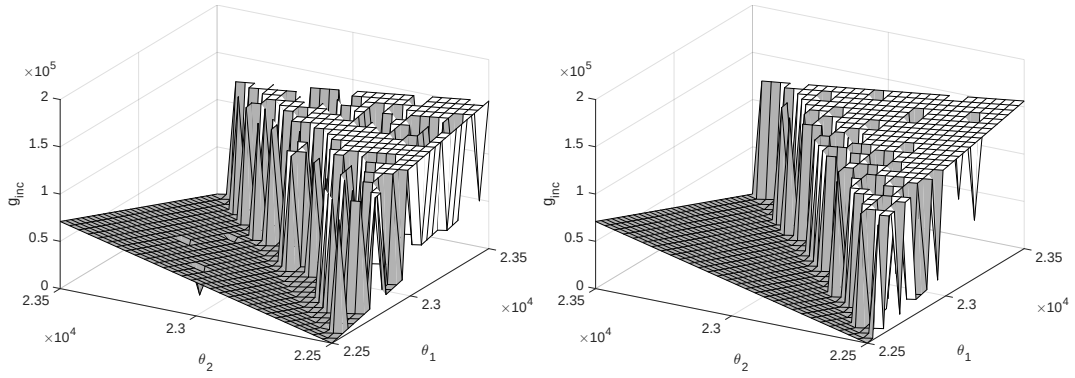
Checking Feasibility

To compute g_{par} and g_{inc}, we have to perform two global optimisations. The global optimisation is done with a SQP-algorithm and multiple random starting points. It is important to use a sufficient amount of starting points, to get accurate results. See for example Figure 4-8. We have plotted $g_{\text{inc}}(\theta)$, where $c_1(t) = 35000$, $K(t) = 129$ and $c_2(t)$ is a linear function. The function $c_2(t)$ is structured as:

$$c_2(t) = \theta_1\phi_1(t) + \theta_2\phi_2(t),$$

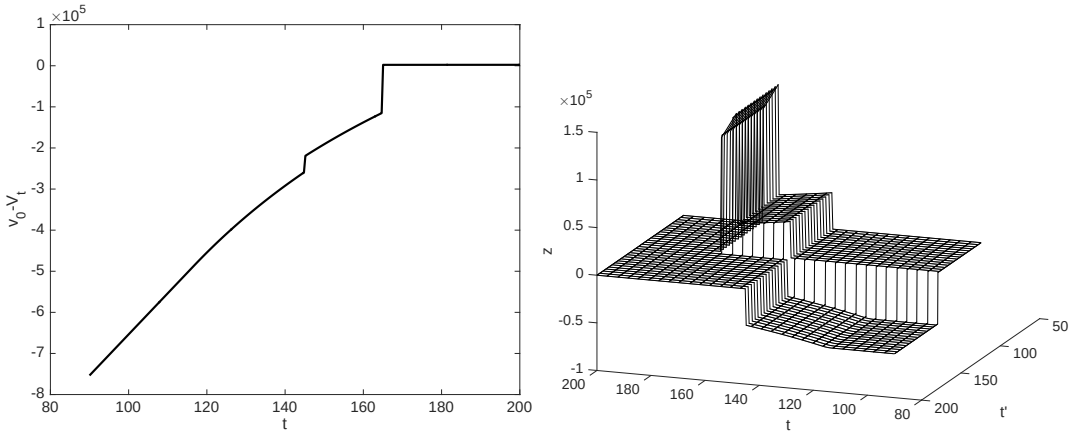
where $\phi_i(\cdot)$ is a second order B-spline. We have plotted g_{inc} over (θ_1, θ_2). We compare two plots where g_{inc} is computed with 20 and 50 starting points. Note that there are many dips and they are located differently in both plots. This suggest that often times the program cannot find the global optimum and needs more starting points.

To get an idea how many starting points we need we create the following experiment; Take the same contract as above and set $(\theta_1, \theta_2) = (23360, 22600)$. The plots of $v_0 - V_t(\alpha_\theta(t), c_\theta(t))$ and $z_\theta\left(t, \tilde{a}, t', \tilde{a}'\right)$ are shown in Figure 4-9. We run 100 global optimisations and check how often it converges to the true global optimum. The results are shown in Table 4-4. In this example, the participation constraint,can be found with only a few starting points. To check incentive compatibility



(a) 20 starting points **(b)** 50 starting points

Figure 4-8: Two plots of g_{ic}. Comparisons when using 20 or 50 starting points.



(a) Participation constraint over t **(b)** Incentive compatibility constraint over (t, t')

Figure 4-9: Checking constraints over all types.

we need 100 starting points and 5-6 minutes of computation time. This makes the basis function algorithm impractically slow on the machine we are working on.

Optimisation

We test a run with one user specified starting point:

$$\begin{aligned} c_1(t) &= 40000 \\ c_2(t) &= 24000 \\ K(t) &= 128. \end{aligned}$$

The principal's expected utility at the starting point is $V_{\text{P0}} = 7.359232 \cdot 10^6$. We first use a basis function approximation with one first order B-spline basis for each entry in $c(t) = (c_1(t), c_2(t), K(t))$, so $c(t)$ is constant in t. In the second test, we use two second order B-spline basis functions for each entry in $c(t)$; Each entry in $c(t)$ is a linear function in t. The optimisation results are in Table 4-5. Looking at the results, the case with one basis function ends prematurely. The optimal V_{P}^* is an

Table 4-4: The fraction that converges too the true global optimum. Results are taken from an experiment with 100 global optimisation runs.

Constraint	# Starting points	Convergence fraction	Avg. optimisation time [s]
g_{par}	1	1	0.64
	3	1	1.64
	5	1	2.65
g_{inc}	5	0.23	32.4
	10	0.43	66.1
	20	0.66	104
	50	0.97	214
	100	1	351

Table 4-5: Optimisation results with the basis function method.

#Basis functions	c^*	$V_{\text{P}}^* \left[\cdot 10^6\right]$	g_{par}	g_{inc}	Run time [h]
1	(40000, 24012, 128.14)	7.361404	-3482.8	0	1.85
2	No convergence	NA	NA	NA	8.68

improvement compared to V_{P0}, but since the participation g_{par} is not tight, we can still improve the solution by reducing c_1. The second test does not converge at all.

To find possible reasons why the program shows poor converge, we could look deeper into the SQP algorithm. The SQP algorithm finds a feasible descent direction d by generating and solving a quadratic program. For that we need the Hessian in θ of the Lagrangian:

$$L(\theta, \lambda) = V_{\text{P}}(\theta) + \lambda_1 g_{\text{par}}(\theta) + \lambda_2 g_{\text{inc}}(\theta),$$

or an approximation thereof. The Matlab implementation in the Optimization Toolbox uses a quasi-Newton approximation of the Hessian, which could result in inaccurate findings of the feasible descent direction. Computing the Hessian with regular finite differencing takes a lot of computation time, particularly in this problem. Note that even with the analytical solution of the Hessian, the direction d is not guaranteed to be a descend direction, see Ex. 12.25 in [36]

4-4-4 Conclusion

We have tested the discretisation method and basis function method on the option contract problem with moral hazard and adverse selection. The best solutions was found with the discretisation method with 15 grid points and 100 starting points. There was still some variation in the top 3 local solution, so we need more starting points to be more confident that the best local solution is also the global one.

The basis function method is really slow. On our machine it requires 5-6 minutes to check for feasibility. In gradient based methods optimisation methods like the SQP-algorithm, we need 10 minutes to compute a single vector entry of ∇g_{inc}. The algorithm is not practical and we conclude that is better to use the discretisation method. Although we might need many starting points and

computation time in the discretisation method to find a global solution, we can find feasible local optimum solution in relative short amount of time.

Chapter 5

Conclusion and Recommendations

We have given a Moral Hazard and Generalised Principal-Agent problem, and we have For the moral hazard problem, we have tested it on an option contract problem and a problem where the reward function is unstructured. The option contract problem can be solved in a around a minute and the problem with the unstructured reward function in around 30 minutes. We suspect that most moral hazard can be numerically solved in a reasonable amount of time. In the near future, it would be interesting to test more kind of problems.

With the generalised problem, we have developed the discretisation method and basis function method, and we have tested it on an option contract problem. The option contract problem is a modification of the problem with only moral hazard. The basis function method is too slow and impractical, because checking for feasibility takes around 5-6 minutes on our machine. The current best solution is found in around a week with the discretisation method. To be certain that it is the global optimal solution, we should use more starting points. For the optimisation, we have to do a lot of independent local optimisations with different starting points. This procedure can be parallelised, so we could reduced the computation time by several factors, if the algorithm is performed on a multi-core desktop workstation. Again, to strengthen the validity of our claims, we would like to test it on more problems in the near future.

5-1 Extensions

5-1-1 Multidimensional Models

In the current models, the agent's action is a single parameter a and he is rewarded based on a single outcome x. This type of model is called *single dimensional*.

In a *multidimensional* principal-agent model, there can be multiple effort parameters $a_1, \ldots, a_n$. The payment also does not need to depend on the outcome x only, but also on additional *performance measures* $y_1, \ldots, y_m$. The payment to the agent is then $w(x, y_1, \ldots, y_n)$. The outcome and

performance measures are random variables conditionally depended on the actions $a_1, \ldots, a_n$ and the agent's type τ, and has a distribution:

$$p\left(x, y_1, \ldots, y_m \middle| a_1, \ldots, a_n, \tau\right).$$

Nowhere in the numerical optimisation methods, do we require the our principal agent model to be single dimensional, so an extension to a multidimensional model should be possible with little difficulties.

There are analytical solutions for a class of moral hazard models called the *linear-exponential-normal* (LEN) formulation, see [37]. Each performance measure y_i is normal distributed and the payment $w\left(x, y_1, \ldots, y_n\right)$ is linear in the outcome x and each performance measure y_i.

5-1-2 Dynamic Models

For now, we have considered static principal-agent problems, where the contracting game ends after one period. We can extend this to a dynamic principal-agent model with multiple periods. The key aspect of a dynamic game is that the agent's past actions and outcomes influences future outcomes. Let $k = 1, 2, \ldots$ be discrete time periods. We say period is a time horizon if a player's total utility over time does not depend on what is happening after the time horizon. At a time period k we will start a contracting game and the agent and principal will act such that they maximise their expected utility over their whole time horizon. When the principal has a different, often longer, time horizon than the agent, then this will lead to misalignment of incentives. Dynamic effects that can be incorporated into the model are:

- Time preference. For the same amount of utility, the agent and principle would rather receive it as soon as possible.
- Past dependence of outcome. The outcome $x(k)$ at time k depends on all previous actions $\{a(i) \in \mathscr{A} : 0 \leq i \leq k\}$ and previous outcomes $\{x(i) \in \mathscr{X} : 0 \leq i \leq k\}$. The action can be multi dimensional, where some actions have an immediate effect on the next outcome and others only affects future outcomes. TThese kind of actions can be interpreted as short term- and long term investments respectively. Interesting scenario is when the agent has to make a compromise between long term and short term investments.
- Forward looking performance measures. To encourage long term investments, the principal needs to reward the agent based on an additional performance measure, that measures potential future earnings.
- Time depended type. The type is the private information the agent posses. In one context, the type is the agent's skill and his skill can improve over time as the agent gains more experience.

  In another context the type is the agent's insight on the profitability of a project he is contracted to work on. Since the agent works close on the project, he has the most knowledge about the projects progress. After each time period, the principal will ask the agent to report on the projects status. The agent's reports influences the forward looking performance measure.

There has been some previous work done to find analytical solutions to dynamic principal-agent problems. Spear and Srivastave [38] and Sannikov [39, 40] looked at a dynamic moral hazard problem, where the outcome at a certain time period only depends on the last action. Jarque [41] extended the model in [38] such that the outcome does depend on all past efforts. The agent's action space is single dimensional and high effort has a positive effect on all future outcomes. In fact, Jarque showed that the model in [41] can be translated to the model in [38]. Cvitanic, Wan and Yang [42] extended the model in [39] with adverse selection. The agent type is assumed to be a constant in time and unknown to the principal.

5-1-3 Multi-Agent Models

At this moment, we have only considered one agent. We can expand this to multiple agents. The goal is to find a contract that maximises the principal's utility, such that all the agent's actions are in a Nash equilibrium. A field similar to principal-agent problems that deals with multiple agent is *mechanism design*.

In mechanism design, there are multiple agents and each agent has a type only known to him. The agents report their type, which results into an outcome and a payment to each agent. The goal is to design a mechanism that maximises a social well-fare function depended on the outcome. If the social well-fare function is defined to be the principal's utility, then we get a adverse selection principal-problem. Similar as with principal-agent problems, the revelation principle applies. This means that we want to find an incentive compatible mechanism, that maximises the social welfare funcion. There has been research done on for example auction design [43, 44, 45] and job scheduling [46]. For a elaborate review on mechanism design check [47].

Appendix A

Expected Value of an Option Contract

Let K be the strike price and a be the agent's action. The random variable $X \sim \mathcal{N}\left(100-a,\sigma^2\right)$ is the value of one company share at the maturity date. The holder's profit of the option contract is the random variable:

$$\max(X-K,0).$$

We want to compute the expected value $\mathbf{E}[\max(X-K,0)]$. First, define $Y = X-K$. We then have $Y \sim \mathcal{N}(\mu_y,\sigma^2)$, where $\mu_y = 100+a-K$.

$$\begin{aligned}\mathbf{E}[\max(X-K,0)] &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\max\left(y,0\right)e^{-\frac{\left(y-\mu_y\right)^2}{2\sigma^2}}dy \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{0}^{\infty}ye^{-\frac{\left(y-\mu_y\right)^2}{2\sigma^2}}dy\end{aligned}$$

by doing the substitution $u = y-\mu_y$, we have $u \sim \mathcal{N}\left(0,\sigma^2\right)$ and

$$\begin{aligned}\mathbf{E}[\max(x-K,0)] &= \frac{1}{\sigma\sqrt{2\pi}}\int_{0}^{\infty}ye^{-\frac{\left(y-\mu_y\right)^2}{2\sigma^2}}dy \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\mu_y}^{\infty}ue^{-\frac{u^2}{2\sigma^2}}du + \frac{1}{\sigma\sqrt{2\pi}}\int_{-\mu_y}^{\infty}\mu_y e^{-\frac{u^2}{2\sigma^2}}du \\ &= \frac{1}{\sigma\sqrt{2\pi}}\left[-\sigma^2 e^{-\frac{u^2}{2\sigma^2}}\right]_{-\mu_y}^{\infty} + \mu_y\mathbf{P}\left(u \geq -\mu_y\right) \\ &= \frac{\sigma}{\sqrt{2\pi}}e^{-\frac{\mu_y^2}{2\sigma^2}} + \mu_y\left(1-F\left(-\mu_y/\sigma\right)\right), \qquad \text{(A-1)}\end{aligned}$$

with $F(\cdot)$ the cumulative distribution function of the standard normal distribution.

Appendix B

System specification

Table B-1: Specifications of the computer that is used in all the calculations.

Description	Value
Computer	MacBook Pro 13.3" Retina, mid 2014
CPU	2.6GHz dual-core Intel Core i5-4278U
GPU	Intel Iris Graphics 5100
RAM	8GB, 1600MHz DDR3L
Storage	256GB PCIe-based flash storage
Operating system	macOS 10.12.x
Matlab version	R2015b + Optimisation Toolbox

Bibliography

[1] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT press, 1994.

[2] J. F. Nash *et al.*, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[3] R. B. Myerson, "Optimal coordination mechanisms in generalized principal–agent problems," *Journal of Mathematical Economics*, vol. 10, no. 1, pp. 67 – 81, 1982.

[4] J. E. Stiglitz, "Incentives and risk sharing in sharecropping," *The Review of Economic Studies*, pp. 219–255, 1974.

[5] M. S.-Y. Chwe, "Why were workers whipped? pain in a principal-agent model," *The Economic Journal*, pp. 1109–1121, 1990.

[6] D. Acemoglu and A. Wolitzky, "The economics of labor coercion," *Econometrica*, vol. 79, no. 2, pp. 555–600, 2011.

[7] I. Krinsky and A. Mehrez, "Principal-agent maintenance problem," *Naval Research Logistics (NRL)*, vol. 36, no. 6, pp. 817–828, 1989.

[8] R. A. Lambert, "Agency theory and management accounting," in *Handbook of Management Accounting Research*, vol. 1, pp. 247–268, Elsevier, 2007.

[9] J. A. Mirrlees, "The theory of moral hazard and unobservable behaviour: Part i," *The Review of Economic Studies*, vol. 66, no. 1, pp. 3–21, 1975.

[10] B. Holmström, "Moral hazard and observability," *The Bell journal of economics*, vol. 10, no. 1, pp. 74–91, 1979.

[11] W. P. Rogerson, "The first-order approach to principal-agent problems," *Econometrica: Journal of the Econometric Society*, vol. 53, no. 6, pp. 1357–1367, 1985.

[12] I. Jewitt, "Justifying the first-order approach to principal-agent problems," *Econometrica: Journal of the Econometric Society*, vol. 56, no. 5, pp. 1177–1190, 1988.

[13] H. Chade and E. Schlee, "Optimal insurance with adverse selection," *Theoretical Economics*, vol. 7, no. 3, pp. 571–607, 2012.

[14] P. Picard, "On the design of incentive schemes under moral hazard and adverse selection," *Journal of public economics*, vol. 33, no. 3, pp. 305–331, 1987.

[15] R. Guesnerie, P. Picard, and P. Rey, "Adverse selection and moral hazard with risk neutral agents," *European Economic Review*, vol. 33, no. 4, pp. 807–823, 1989.

[16] J. Li, "Optimal incentive mechanism design under adverse selection and moral hazard," in *Management and Service Science (MASS), 2010 International Conference on*, pp. 1–4, IEEE, 2010.

[17] B. Theilen, "Simultaneous moral hazard and adverse selection with risk averse agents," *Economics Letters*, vol. 79, no. 2, pp. 283–289, 2002.

[18] P. S. Faynzilberg and P. Kumar, "On the generalized principal-agent problem: decomposition and existence results," *Review of Economic Design*, vol. 5, no. 1, pp. 23–58, 2000.

[19] G. A. Akerlof, "Labor contracts as partial gift exchange," *The Quarterly Journal of Economics*, vol. 97, no. 4, pp. 543–569, 1982.

[20] R. Desiraju and D. E. Sappington, "Equity and adverse selection," *Journal of Economics & Management Strategy*, vol. 16, no. 2, pp. 285–318, 2007.

[21] M. Cecchini, J. Ecker, M. Kupferschmid, and R. Leitch, "Solving nonlinear principal-agent problems using bilevel programming," *European Journal of Operational Research*, vol. 230, no. 2, pp. 364–373, 2013.

[22] C. D. Kolstad and L. S. Lasdon, "Derivative evaluation and computational experience with large bilevel mathematical programs," *Journal of optimization theory and applications*, vol. 65, no. 3, pp. 485–499, 1990.

[23] G. Savard and J. Gauvin, "The steepest descent direction for the nonlinear bilevel programming problem," *Operations Research Letters*, vol. 15, no. 5, pp. 265–272, 1994.

[24] J. F. Bard and J. T. Moore, "A branch and bound algorithm for the bilevel programming problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 2, pp. 281–292, 1990.

[25] E. Aiyoshi and K. Shimizu, "Hierarchical decentralized systems and its new solution by a barrier method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 444–449, 1981.

[26] B. Colson, P. Marcotte, and G. Savard, "A trust-region method for nonlinear bilevel programming: algorithm and computational experience," *Computational Optimization and Applications*, vol. 30, no. 3, pp. 211–227, 2005.

[27] O. Stein, "How to solve a semi-infinite optimization problem," *European Journal of Operational Research*, vol. 223, no. 2, pp. 312–320, 2012.

[28] S. Leyffer, "Complementarity constraints as nonlinear equations: Theory and numerical experience," in *Optimization with Multivalued Mappings*, pp. 169–208, Springer, 2006.

[29] M. López and G. Still, "Semi-infinite programming," *European Journal of Operational Research*, vol. 180, no. 2, pp. 491–518, 2007.

[30] C. A. Floudas and O. Stein, "The adaptive convexification algorithm: a feasible point method for semi-infinite programming," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1187–1208, 2007.

[31] O. Stein and P. Steuermann, "The adaptive convexification algorithm for semi-infinite programming with arbitrary index sets," *Mathematical programming*, vol. 136, no. 1, pp. 183–207, 2012.

[32] S. J. Grossman and O. D. Hart, "An analysis of the principal-agent problem," *Econometrica: Journal of the Econometric Society*, vol. 51, no. 1, pp. 7–45, 1983.

[33] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag New York, 1978.

[34] G. G. Lorentz, *Bernstein polynomials*. University of Toronto Press, 1953.

[35] J. F. Epperson, "On the Runge example," *The American Mathematical Monthly*, vol. 94, no. 4, pp. 329–341, 1987.

[36] U. Faigle, W. Kern, and G. Still, *Algorithmic principles of mathematical programming*, vol. 24. Springer Science & Business Media, 2002.

[37] R. A. Lambert, "Contracting theory and accounting," *Journal of accounting and economics*, vol. 32, no. 1, pp. 3–87, 2001.

[38] S. E. Spear and S. Srivastava, "On repeated moral hazard with discounting," *The Review of Economic Studies*, vol. 54, no. 4, pp. 599–617, 1987.

[39] Y. Sannikov, "A continuous-time version of the principal-agent problem," *The Review of Economic Studies*, vol. 75, no. 3, pp. 957–984, 2008.

[40] Y. Sannikov, "Contracts: The theory of dynamic principal-agent relationships and the continuous-time approach," in *10th World Congress of the Econometric Society*, 2012.

[41] A. Jarque, "Repeated moral hazard with effort persistence," *Journal of Economic Theory*, vol. 145, no. 6, pp. 2412–2423, 2010.

[42] J. Cvitanic, X. Wan, and H. Yang, "Dynamics of contract design with screening," *Management Science*, vol. 59, no. 5, pp. 1229–1244, 2013.

[43] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[44] E. H. Clarke, "Multipart pricing of public goods," *Public choice*, vol. 11, no. 1, pp. 17–33, 1971.

[45] T. Groves, "Incentives in teams," *Econometrica: Journal of the Econometric Society*, pp. 617–631, 1973.

[46] R. Lavi, "Computationally efficient approximation mechanisms," *Algorithmic Game Theory*, pp. 301–329, 2007.

[47] N. Nisan, "Introduction to mechanism design (for computer scientists)," *Algorithmic game theory*, pp. 209–242, 2007.