Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

Diplomarbeit

Stochastic Projectmanagement

Applying dynamic programming and CRRA utility to projects subject to cost uncertainty

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

Ao.Univ.Prof. Mag.rer.nat.
Dr.rer.soc.oec. Dr.techn. Thomas Dangl

am Institut für Managementwissenschaften - E330
Forschungsbereich Finanzwirtschaft und Controlling

eingereicht an der Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften
von

Sebastian Rötzer, BSc
e0726857

Weinbergsiedlung 1
A-2465 Höflein

Wien, October 10, 2013

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Behandlung von Projekten mit unsicheren Kosten bei vergleichsweise bekannt angenommen Erträgen als Investitionsproblem. Dazu wird auf die Bellman Methode, auch Dynamische Programmierung genannt, zurückgegriffen und die Entwicklung der Projektkosten als geometrisch Brownsche Bewegung auf einem diskreten Gitternetz modelliert.
Dabei wird, in einem ersten Beispiel, ein Investment-Modell auf dem Gitter aufgesetzt, welches nach dem 'Alles-oder-Nichts' Prinzip agiert. Diese grundlegende, erste Implementierung wurde für ausführlich Untersuchungen zum korrekten Verhalten des Gitters genutzt und anschließend um eine Nutzenfunktion erweitert. Diese sogenannte CRRA Nutzenfunktion wiederum dient dem Zweck menschliches Verhalten der Risikovermeidung in beliebiger Ausprägung in das Modell einfließen zu lassen und ermöglichte erstmal das Vorkommen von internen Maxima d.h. Extremstellen.
Abschließend wurde untersucht ob die Vorgehensweise, die sich zur Evaluierung eines Einzelprojekts mit unsicheren Kosten eignet, auch zur Bewertung und Risikominimierung eines Projektportfolios verwendet werden kann.
Zur Umsetzung des Rechenmodells auf einem Rechner wird die freie Programmiersprache R herangezogen. Die entsprechend verwendeten Programmcodes sind im Appendix C zu finden.

Abstract

The thesis work at hand is concerned with the treatment of projects of uncertain cost and comparatively known revenues as investment-problem. Therefore we resort to the Bellman method also referred to as dynamic programming and model the development of expected project cost as geometric Brownian motion on a discrete computation lattice. In the process we set up an investment-model as first example that operates as 'make-or-break' solution. This fundamental, first implementation was used for extensive investigation on the correct behaviour of the grid and was complemented with an utility function afterwards. The so called CRRA (constant relative risk aversion) utility function serves the purpose of simulating human risk-avoiding behaviour in arbitrary peculiarity within the model and enables the occurrence of internal extrema for the first time.
Concluding we investigated whether the approach that suits the assessment of an individual project of uncertain cost could also be extended to the evaluation and risk-minimisation of project portfolios.
For the implementation of the calculation model on a personal computer the free programming language R was utilized. The respectively developed source codes can be found in appendix C.

Acknowledgement

I want to thank Ao.Univ.-Prof. Mag. DDr. Thomas Dangl for the supervision of my thesis and the good collaboration not only in the development of this project but also in the conduction of the course 'Betriebswirtschaftliche Optimierung' where I got the opportunity to accompany the course as tutor. With my leave from the Vienna University of Technology I wish him all the best for his future endeavours.
Furthermore I would like to express my appreciation to my family who encouraged me to pursue university studies and kept supporting me throughout the past six years.
Last but not least I want to thank my friends, especially Gernot, Klemens, Daniel and Markus, who made me who I am and Bianca who always succeeds in making me smile.

'Mehr als die Vergangenheit interessiert mich die Zukunft, denn in ihr gedenke ich zu leben.'

Albert Einstein

Contents

1. Introduction

Let us define a project as any temporary and unique (social) system whose purpose is to achieve a certain set of given objectives in a specific period of time while subject to a defined number of constraints. These constraints originate from the limitation of resources be it monetary, temporal, social, environmental or quality-wise. An endeavour can be labeled project if it has distinct start- and termination-deadlines, the problem looking for solution appears relatively complex and that the approach is initially unknown.
The afore-mentioned complexity can accrue from

- the large number of possible approaches to find a solution whose effectiveness is not yet assessable
- conflicts of aims
- reciprocity of measures taken to achieve the projects goal
- discrepancy in project understanding of involved parties.

All efforts undertaken to successfully operate such an enterprise can be subsumed as feedback control system of the project. This thesis work is the authors contribution to such control systems by developing a mathematical model that represents projects of uncertain cost and implementing it by application of the free programming language R. Any particular project concept pursued is subject to one of the three different cases illustrated by figures 1.1, 1.2 and 1.3. First there is figure 1.1 where project financing is not even initiated. We see a grey chart with quasi-random behaviour of the projects estimated cost to completion whereupon the colour grey represents the activity of 'waiting' i.e. non-investment. With the passing of time additional information unfolds that can change the projects estimated remaining cost in both directions either up- or downwards. As we will learn later on this can only occur if the expected revenues are to low or vice-versa the assessed project cost estimation is to high for any point in time to form a reasonable investment. Note that the project diagrammed will be terminated at t = 10 regardless of its state of completion.
Heading on to figure 1.2 we observe a project that was kicked-off when the economic environment seemed beneficial and that was on its way to completion when a small disturbance caused it to slip out of the profitable area whereby it was dragged on for some time but eventually terminated unsuccessfully at the deadline.
Lastly we have a prosperous project shown in diagram 1.3. In the strictly mathematical representation chosen for this work a project meets its goals when the estimated remaining cost falls beneath a critical margin for instance one monetary unit. Only if

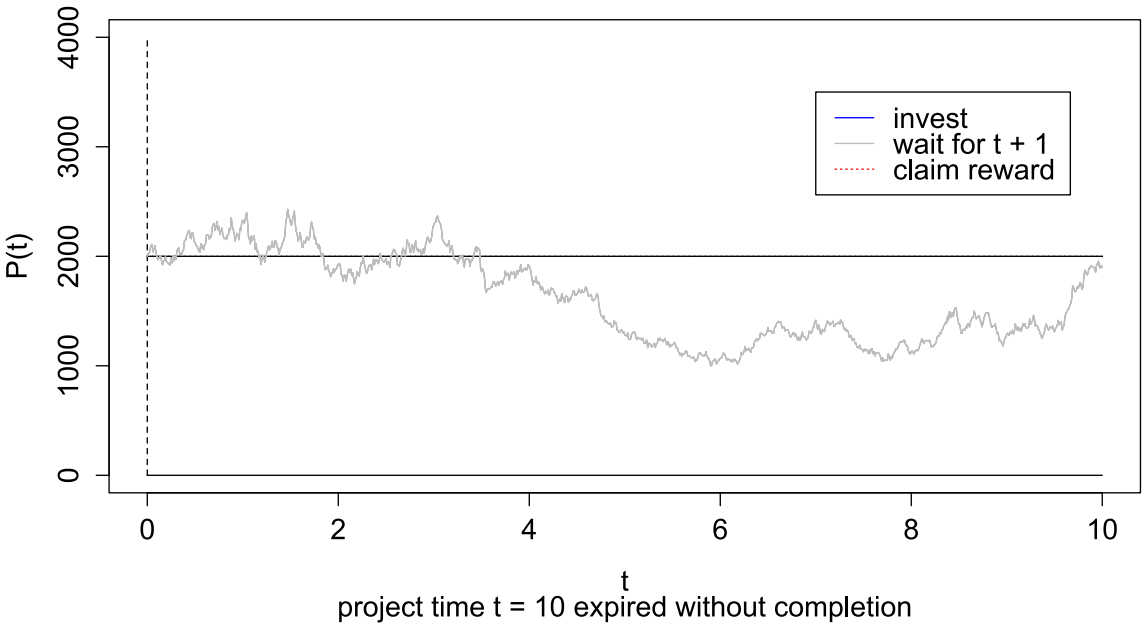

Figure 1.1.: Time horizon expired without project kick-off

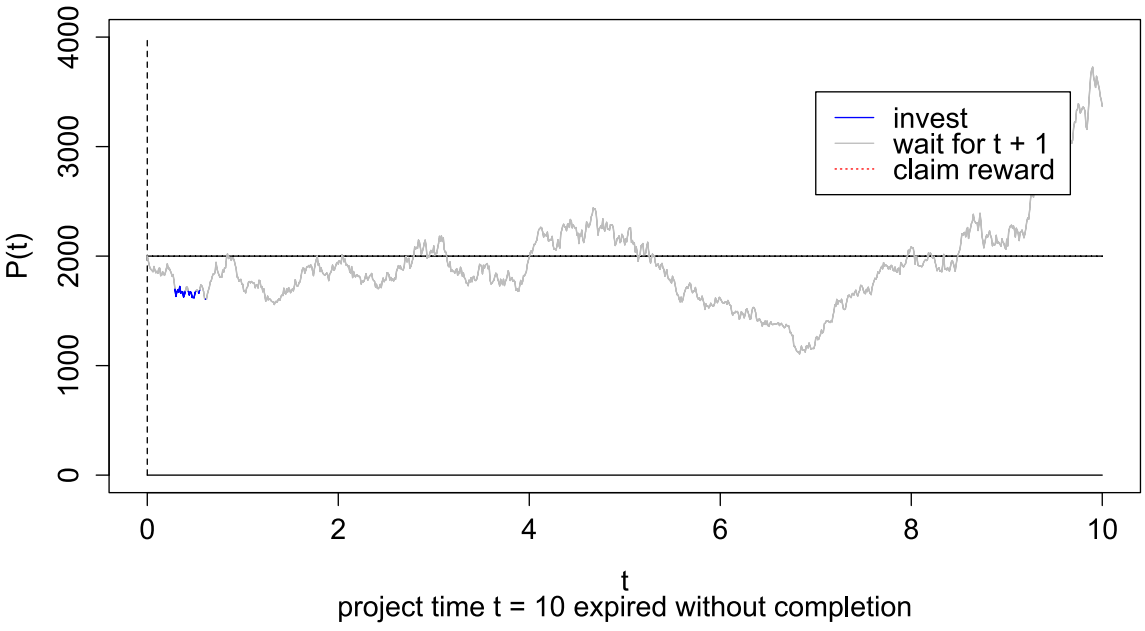

Figure 1.2.: Project cost development outside of profitable area

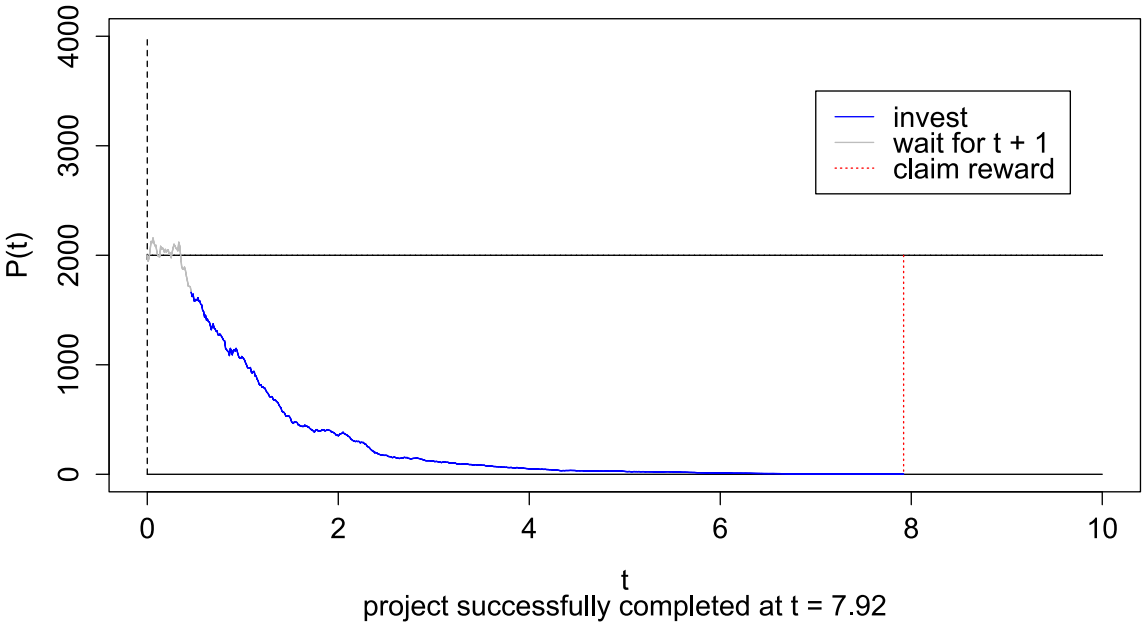

Figure 1.3.: Successful project finish within time horizon

this aim is met within the pre-specified time horizon the project organisation earns itself a financial reward to compensate its prior expenses.
It is now time to switch to a short introduction of this thesis and its contents. In chapter 2 we start out with the basic ideas proposed by Richard E. Bellman in 1957 when he published his book named 'Dynamic programming'[1] . Essentially this is an approach of solving large-scale i.e. high-dimensional optimization problems by breaking them down into a sequence of familiar low-dimensional tasks. We exemplify dynamic programming for a deterministic function first and move on to stochastic systems afterwards.
Following this short theoretical discussion three chapters containing examples of dynamic programming in project investment with increasing complexity await the interested reader. To begin with we establish a general way of representing a project's cost development over time by the application of a geometric Brownian motion in chapter 3. Furthermore we introduce a simple investment model based on the relation of cost to expected gains of investing. Whenever the algorithm finds that the gradient of this relation is positive it will allocate money to the project at the maximum allowable rate. Chapter 4 is an evolution of the preceding one and marks the introduction of the CRRA (constant relative risk aversion) utility function to the investment model. By the employment of this utility function it is possible to simulate a real investors decision more precisely and obtain internal solutions i.e. abandon the initial 'make-or-break' tactics. Therefore it was however necessary to completely revise the model as the CRRA utility is a consumption based concept, i.e., it assigns strictly positive levels of consumption

the utility realized by the decision maker.
The last example found in chapter 5 tries to expand the evaluation of individual investment projects to a portfolio of several (in our case two) projects. With the insights gathered from the one-dimensional case and the studies of Boyle, Evnine and Gibbs [2] a new model is constructed that merges the two correlated random walks into movement described by a single probability value. While initial attempts to find a numeric solution to the portfolio optimization problem were not successful and quite resource demanding for the executing personal computer a later analytic approach turned out to work like a charm.
After concluding the last example a short conclusion follows. An appendix concludes the thesis. This last section of the work is again divided in three parts containing extensive Monte Carlo studies on the results of example A1, a method of determining additional constraints for example B in order to retain real probabilities and finally the source code employed to generate all results present in this book. On the very last page the bibliography with suggestions for further reading is located.

2. Fundamentals on dynamic programming

2.1. Aim and references

Before starting out with the investment optimization problem briefly mentioned in the introductory chapter we will discuss the prospective methods employed. Our starting point will be a chapter in Rangarajan K. Sundaram's *'A First Course In Optimization Theory'* [14] that is focused on the description and solution of finite horizon dynamic programming problems. This reference that was chosen for its clearness will give us a first intuition of dynamic programming in deterministic scenarios and lead us to examples from the classical work *'Dynamic Programming'* [1] presented by Richard E. Bellman in 1957. To further our goal of achieving an insight into dynamic programming the examples described in equations 2.16 to 2.31 are directly taken from Bellman's book. If interested in a mathematical proof please refer to the original publication [1] chapter I - III. Also Bellman's examples will draw a bow from deterministic to stochastic scenarios which are the focus of this thesis' practical part. In addition a reference to Avinash Dixit's and Robert S. Pindyck's book [5] concludes this chapter. This book and an earlier working paper published by Pindyck served as main inspirations for the models that are to be constructed and discussed in later chapters.

2.2. Dynamic programming of deterministic target functions

In this section we focus on problems whose target function is determined by the current state and the action chosen by a decision-maker. The term deterministic connotes that the same combination of state and action will at any time produce the same result. This is different from stochastic target functions which will be discussed afterwards where the result is given as distribution over a probability space. Since the class of problems undergoing investigation share the feature of distinct termination dates their study is named Finite Horizon Dynamic Programming.

2.2.1. An introduction based on Rangarajan K. Sundaram's *A First Course In Optimization Theory*

We define a *Finite Horizon (Markovian) Dynamic Programming Problem* (henceforth referred to as FHDP) as a tuple consisting of the elements $\{S, A, T, (r_t, f_t, \Phi_t)_{t=1}^T\}$. Whereat

- S is the *state space* of the problem, with generic element s.
- A is the *action space* of the problem, with generic element a.
- T, a positive integer, is the *horizon* of the problem.

For each $t \in \{1, \ldots, T\}$,

- $r_t : S \times A \to \mathbb{R}$ is the period-t *reward function*,
- $f_t : S \times A \to S$ is the period-t *transition function*, and
- $\Phi_t : S \to P(A)$ is the period-t *feasible action correspondence*.

An interpretation of the FHDP could read as follows. In the beginning the problem is found in some initial state $s_1 = s \in S$ and the decision-maker is faced with a set of possible action alternatives denoted $\Phi_1(s_1) \subset A$. Once the decision-maker chooses an action $a_1 \in \Phi_1(s_1)$ to proceed an immediate reward $r_1(s_1, a_1)$ is allocated and a state-transformation to $s_2 = f_1(s_1, a_1)$ takes place. This cycle of state analysis, selection of appropriate measures, reward distribution and transformation can be continued all the way to the terminal date T. So far the problem appears rather simple the tricky task of the operator though is to maximize the sum of all rewards accumulated over the problem horizon. This is synonymous to

$$\begin{aligned} \text{Maximize } & \sum_{t=1}^{T} r_t(s_t, a_t) && && (2.1) \\ \text{subject to } & s_1 = s \in S \\ & s_t = f_{t-1}(s_{t-1}, a_{t-1}), && t = 2, \ldots, T \\ & a_t \in \Phi_t(s_t), && t = 1, \ldots, T \end{aligned}$$

While this approach catches with its simplicity Sundaram points out that there are better ways to represent the FHDP as well as the objective if assigned with the analytical solution of such a problem. This leads us to the modelling as histories, strategies and the value function.

One can think of a FHDP problem's solution as an interactive map that describes the exactly best possible way i.e. decisions to be made depending on where the decision-maker

stands i.e. the state of the problem and how it was reached. As mathematical expression of the decision-makers current location would be the history $\eta_t = \{s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t\}$ that consists of the current (s_t) and all preceding $(s_1, \ldots, s_{t-1})$ states as well as the corresponding actions $(a_1, \ldots, a_{t-1})$ taken. Furthermore we can think of $H_1 = S$ and H_t as the set of all possible histories η_t for $t > 1$. Provided with anyone history η_t we identify $s_t[\eta_t]$ as the period-t state under the history η_t. Until now we have concerned ourselves with the description of the decision-makers problem but now we will turn towards possible solutions for the entire problem horizon. Let us define a strategy σ as a sequence of decisions $\{\sigma\}_{t=1}^{T}$ where for each t $\sigma_t : H_t \to A$ specifies the action $\sigma_t(\eta_t) \in \Phi_t(s_t[\eta_t])$ for period t as function of $\eta_t \in H_t$. Consider $\sigma_t(\eta_t) \in \Phi_t(s_t[\eta_t])$ a built-in feasibility requirement that secures the strategies validity. Also let us denote a set of all possible σ for the problem which we name Σ.
Returning to our objective from the preceding section we denote the total reward under strategy σ with starting condition s as

$$W(\sigma)(s) = \sum_{t=1}^{T} r_t(\sigma)(s) \tag{2.2}$$

We define the value function $V : S \to \mathbb{R}$ by

$$V(s) = \underset{\sigma \in \Sigma}{sup} W(\sigma)(s) \tag{2.3}$$

and deem a strategy σ^* an optimal if and only if its eventual total reward from any state s_1 equals the supremum of possible pay-offs from that state. Note that there might be more than one optimal strategy.

$$W(\sigma^*)(s) = V(s), \text{ for all } s \in S \tag{2.4}$$

In a given FHDP (eq. 2.5) there any τ -history η_τ can be taken as sample that leads to a new FHDP which we refer to as the $T - \tau$ - period continuation problem described by equation 2.6. For the purpose of notational simplicity the new problem will also be denoted by $\{S, A, T - \tau, (r_t, f_t, \Phi_t)_{t=\tau+1}^{T}\}$.

$$\{S, A, T, (r_t, f_t, \Phi_t)_{t=1}^{T}\} \tag{2.5}$$

$$\{S, A, T - \tau, (r_t^*, f_t^*, \Phi_t^*)_{t=1}^{T-\tau}\} \tag{2.6}$$

The new problem originates from state $s = s_\tau$ and is subject to the conditions 2.7.

$$\begin{aligned} r_t^*(s,a) &= r_{t+\tau}(s,a), & (s,a) &\in S \times A \\ \Phi_t^*(s) &= \Phi_{t+\tau}(s), & s &\in S \\ f_t^*(s,a) &= f_{t+\tau}(s,a), & (s,a) &\in S \times A \end{aligned} \tag{2.7}$$

Let us now state a bold assumption: All τ-histories η_τ that end in s_τ result in the same continuation $(T-\tau)$ -period problem. This means that the current state s_t and the time period in which this state was reached contain all relevant information regarding continuation possibilities i.e. feasible strategies and attainable rewards. This also incorporates that is it not important how one arrives at a certain state i.e. the problems solution is path independent. We call this feature Markovian behaviour. Let us denote a Markovian strategy as sequence $\{g_1 \ldots, g_T\}$ where an action $g_t(s_t) \in \Phi_t(s_t)$ selected for each t is subject to $g_t : S \to A$. A Markovian strategy that is found to be optimal is called an Markovian optimal strategy and is superior not only to other Markovian approaches but to all other decision sequences.
To verify the existence of an (Markovian) optimal strategy we use a sequence of decisions seen in eq. 2.8 that follows the recommendations of a strategy σ for the initial (t - 1) periods and switches to the command of a Markovian strategy afterwards. Note that the procedure $\{\sigma_1, \ldots, \sigma_{t-1}, g_t, \ldots, g_T\}$ is replaced by γ for notational simplicity in the proof paragraph.

$$\{\sigma_1, \ldots, \sigma_{t-1}, g_t, \ldots, g_T\} \tag{2.8}$$

Furthermore we incorporate Lemma 2.2.1 quoted from *A First Course in Optimization Theory*[14] for this purpose.

Lemma 2.2.1 *Let $\sigma = (\sigma_1, \ldots, \sigma_T)$ be an optimal strategy for the FHDP*

$$\{S, A, T, (r_t, f_t, \Phi_t)_{t=1}^T\}$$

Suppose that for some $\tau \in \{1, \ldots, T\}$, the $(T - \tau + 1)$ - period continuation problem

$$\{S, A, T - \tau + 1, (r_t, f_t, \Phi_t)_{t=\tau}^T\}$$

admits a Markovian optimal strategy $\{g_\tau, \ldots, g_T\}$. Then, the strategy

$$\{\sigma_1, \ldots, \sigma_{t-1}, g_t, \ldots, g_T\}$$

is an optimal strategy for the original problem.

Proof Should the Lemma presented afore turn out to be incorrect, then it would be possible that the optimal strategy σ produces results superior to the rewards attainable under approach γ for any initial state s. So that equation 2.9 becomes true.

$$W(\sigma)(s) > W(\gamma)(s) \tag{2.9}$$

If the declaration in equation 2.9 is to hold we must require the relationship shown in eq. 2.10 since the sum of the $(\tau - 1)$ period rewards are equal to each other by construction.

$$\sum_{t=\tau}^{T} r_t(\sigma)(s) > \sum_{t=\tau}^{T} r_t(\gamma)(s) \tag{2.10}$$

Once we let denote s_τ^* the common period τ state of both strategies and let inequality 2.10 postulates the superiority of σ over γ we run into a contradiction of the optimality of γ for the $T - \tau$ period continuation problem. The correctness of Lemma 2.2.1 is of major importance for our application as it will enable us to solve dynamic programming problems by the method of backward induction. That means one can start from the one-period problem with any $s \in S$ as in eq. 2.11. According to Lemma 2.2.1 we are able to use a Markovian strategy g_T^* without affecting the total rewards in comparison to σ_T^*. No we could go ahead and find the optimal solution for the two period problem constituted by T and (T - 1). This results in a decision sequence (g_{T-1}^*, g_T^*). By inclusion of an induction argument the construction of an optimal strategy can be concluded.

$$\text{Maximize } r_T(s,a) \text{ subject to } a \in \Phi_T(s) \tag{2.11}$$

To ensure validity of the solution a couple of conditions must be met that enforce minimal continuity and compactness of the problems target and transformation functions.

1. For each t, r_t is continuous and bounded on $S \times A$.
2. For each t, f_t is continuous on $S \times A$.
3. For each t, Φ_t is a continuous, compact-valued correspondence on S.

Theorem 2.2.1 *Under 1 - 3, the dynamic programming problem admits a Markovian optimal strategy. The value function V_t of the $(T - \tau + 1)$ period continuation problem satisfies for each $t \in \{1, \ldots, T\}$ and $s \in S$, the following condition, known as the 'Bellman Equation', or the 'Bellman Principle of Optimality':*

$$V_t(s) = \max_{a \in \Phi_t(s)} \{r_t(s,a) + V_{t+1}[f_t(s,a)]\}$$

Proof We will start with the one-period case and develop the two-period example from there. Afterwards a simple induction argument suffices to prove 'Bellman Principle of Optimality' correct. Any strategy g_T can only be optimal for the one-period case if it solves equation 2.12.

$$\max\{r_T(s,a) | a \in \Phi_T(s)\} \tag{2.12}$$

With our assumptions concerning continuity and compact-valued-ness the maximized value turns out to be simply $V_T(s)$. If we no choose an function $g_T^* : S \to A$ in a way that $g_T^*(s) \in \Phi_T^*(s)$ for all $s \in S$ i.e. carry out a selection, g_T^* will advise us an optimal decision for any initial state s. There may exist more than one optimal strategy however the resulting pay-off will always be $V_T(s)$. Should we now take our optimal strategy g_T^* and face a two-period problem and additional decision for the period (T-1) is necessary. Once we decide for any action $a \in \Phi_{T_1}(s)$ an immediate reward $r_{T_1}(s,a)$ is distributed and the problem undergoes a transformation $f_{T-1}(s,a)$ so that a new state s_T is realized.

The maximum reward of this continuation problem is by definition $V_T[f_{T_1}(s,a)]$ attained by implementation of g_T^*. Thus the maximum reward of a two-period example subject to activity a in the first period is given by equation 2.13.

$$r_{T-1}(s,a) + V_T[f_{T_1}(s,a)] \tag{2.13}$$

Under this assumption we can state that $\{g_{T-1}^*, g_T^*\}$ is an optimal strategy for two-period cases if it solves equation 2.14.

$$\max_{a \in \Phi_{T-1}(s)} \{r_{T-1}(s,a) + V_T[f_{T-1}(s,a)]\} \tag{2.14}$$

Under the supposition that we established and verified the following two statements a skilled mathematician can easily prove the correctness of the Bellman equation 2.15 by induction.

1. A Markovian optimal strategy i.e. a decision sequence $\{g_{t+1}^*, \ldots, g_T^*\}$ can be found for the (T - t) period problem,
2. and said problem's value function V_{t+1} is continuous on S.

$$V_t(s) = \max_{a \in \Phi_t(s)} \{r_t(s,a) + V_{t+1}[f_t(s,a)]\} \tag{2.15}$$

2.2.2. A deterministic allocation process example from Richard E. Bellman's *Dynamic Programming*

$$f(x) = \max_{0 \leq y \leq x} [g(y) + h(x-y) + f(ay + b(x-y))] \tag{2.16}$$

Consider a process from any application field whose input is denoted by the quantity x which can be divided into two non-negative parts y and $(x-y)$ and subsequently spent on the two continuous return functions $g(y)$ and $h(x-y)$. If assigned with the optimization of the process's output one would go about to search for the maximum of the analytic equation 2.17 with constraints 2.18.

$$R_1(x,y) = g(y) + h(x-y) \tag{2.17}$$

$$\begin{aligned} 0 \leq y \leq x \\ x \geq 0 \end{aligned} \tag{2.18}$$

Now imagine that employing our process doesn't entirely consume the input resources yet their initial quantities shrink to ay and $b(x-y)$ where $0 \leq a \leq 1$ and $0 \leq b \leq 1$. Since there are unspent resources (quantified by the value x_1 as seen in 2.19) we continue utilizing the process and claim another round of returns.

$$ay + b(x-y) = x_1 = y_1 + (x_1 - y_1) \tag{2.19}$$

As we already possess returns from the first application we add them up to the second application's output and hence have a total two stage yield denoted by equation 2.20. Extended to two allocation stages the process's maximum return can be found be optimizing the function in y and y_1 whilst maintaining the constraints in 2.21.

$$R_2(x, y) = g(y) + h(x - y) + g(y_1) + h(x_1 - y_1) \tag{2.20}$$

$$\begin{aligned} 0 \leq y \leq & x \\ 0 \leq y_1 \leq & x_1 \end{aligned} \tag{2.21}$$

Turning toward an N-stage allocation process the structure of the total return would look like equation 2.22 with user distributable resources quantified by the 2.23 set of equations.

$$R_N(x, y, y_1, \ldots, y_{N-1}) = g(y)+h(x-y)+g(y_1)+h(x_1-y_1)+\ldots+g(y_{N-1})+h(x_{N-1}+y_{N-1}) \tag{2.22}$$

$$\begin{aligned} x_1 &= ay + b(x - y), & 0 \leq y &\leq x \\ x_2 &= ay_1 + b(x_1 - y_1), & 0 \leq y_1 &\leq x_1 \\ &\vdots \\ x_{N-1} &= ay_{N-2} + b(x_{N-2} - y_{N-2}), & 0 \leq y_{N-2} &\leq x_{N-2} \\ & & 0 \leq y_{N-1} &\leq x_{N-1} \end{aligned} \tag{2.23}$$

Common numerical approaches and calculus

For the purpose of better understanding the benefits of dynamic programming compared to ordinary methods we will walk down the paths of a 'brute-force' numerical solution and calculus to evaluate and comment on the problems arising.
If for example the operator refuses the application of any analytic tools and instead insists to compute the maximum in a strictly numerical way he might partition each decision space x_i in a suitable number of lattice points, say 10. Trying to calculate the maximum result of a 10-stage allocation process $R_{10}(y, y_1, \ldots, y_9)$ he would run into a dimension of 10^{10} operations since every lattice point spawns a new decision space again segmented by 10 lattice points. So the order of the problem is given by 2.24 with N the number of allocation stages and m the lattice density.

$$\Theta(m^N) \tag{2.24}$$

Note that for a 20-stage process the number of operations becomes $10^{10} \cdot 10^{10}$. Imagine now that computation, comparison, storage and all processing of one lattice point took one millisecond. Then the numerical solution would require $2.77 \cdot 10^3$ hours. For a

microsecond however the number is 2.77 hours which already sounds fairly reasonable. Unfortunately this approach doesn't yield any information on the underlying structure of the solution, thus if stability or sensitivity analysis' are to be carried out (i.e. set of variables for a,b and x as well as sets of functions for g and h) the problem will run out of hand fairly quick. Even more so if the simple process is to be expanded to more realism which could mean increases in the number of resources or available process choices.
Hence we prospect for another approach and employ calculus as method of choice i.e. consider the whole problem a single stage, multivariate allocation process of N-interdependent variables whose maximum can be found by taking partial derivatives and solving the resulting equation system 2.25. Note however that this approach ignores the possibilities of maxima in the boundary values $y_i = 0$ and $y_i = x_i$ which need to be reviewed additionally not mentioning the necessity of testing all possible combinations yet. To make matters worse one has to ensure that the solution found is, if not unique, a real absolute maximum and not a minimum or a simple local maximum or a saddle point.

$$\begin{aligned} &g'(y_{N-1}) + h'(x_{N-1} - y_{N-1}) && = 0 \\ &g'(y_{N-2}) + h'(x_{N-2} - y_{N-2}) + (a-b)h'(x_{N-1} - y_{N-1}) && = 0 \\ &\vdots \\ &g'(y) + h'(x-y) + (a-b)h'(x_1 - y_1) + \ldots && \ldots = 0 \end{aligned} \tag{2.25}$$

Again the approach is cluttered by the sheer number of calculations and possibilities that have to be taken into consideration. Going back to our initial two-stage problem we can rethink our results so far. Though one is naturally interested in receiving the point $(y, y_1; \ldots, y_N)$ in policy space that yield the maximum return all that is of immediate importance to the individual carrying out the decision is the single choice of y in terms of the available x. Thus the former multivariate, single-stage problem transforms into a N-stage process of one-dimensional choices. This is the basic idea of dynamic programming.

Dynamic programming

Assuming that the optimal decision for every stage of the process is known the resulting maximum only depends on the initially available resource quantity x and the number of allocation steps N. Pursuing our concept of one-dimensionality from the end of the last section we are able to formulate equations 2.26 and the solution for the one-stage case 2.27.

$$f_N(x) = \max_{y,y_i} R_N(x, y, \ldots, y_{N-1}), N = 2, 3, \ldots \tag{2.26}$$

$$f_1(x) = \max_{0 \le y \le x} [g(y) + h(x-y)] \tag{2.27}$$

Next, we will proceed an try to formulate a function that yields the maximum return in a two stage allocation process. In the one-stage process it was clear that the resource

has to be used in the best possible manner if a maximum return is to be achieved. How about a two- or even N-stage allocation? Sure enough there is no difference. To receive a maximum return of the two-stage process we simply take the spare resources x_1 lead them back to our initial allocation process (i.e allocating them as y_1 and $(x_1 - y_1)$) thus gaining a total return posed by eq. 2.28. We can therefore write down the functional equation as recurrence relation shown on eq. 2.29.

$$R_2(x, y, y_1) = g(y) + h(x - y) + f_1(ay + b(x - y)) \tag{2.28}$$

$$f_2(x) = \max_{0 \leq y \leq x} [g(y) + h(x - y) + f_1(ay + b(x - y))] \tag{2.29}$$

Consequently the upcoming and ultimate task is to find the allocation function for a N-stage process. Interpreting the structure we found above we denote the N-stage allocation as eq. 2.30. So if beginning with the initial allocation $f_1(x)$ and thus gaining the first return and spare resources we can employ 2.30 to find $f_2(x)$, then $f_3(x)$ and so on. Note that this approach does not only yield the quantity of the return $f_N(x)$ yet even produces the sequence of decisions labelled $y_N(x)$ for the N-stage allocation.

$$f_N(x) = \max_{0 \leq y \leq x} [g(y) + h(x - y) + f_{N-1}(ay + b(x - y))] \tag{2.30}$$

$$\begin{aligned} \bar{y} &= y_N(x) \\ \bar{y}_1 &= y_{N-1}(a\bar{y} + b(x - \bar{y})) \\ \bar{y}_2 &= y_{N-2}(a\bar{y}_1 + b(x_1 - \bar{y}_1)) \\ &\vdots \\ \bar{y}_{N_1} &= y_1(a\bar{y}_{N-2} + b(x_{N-2} - \bar{y}_{N-2})) \end{aligned} \tag{2.31}$$

Once the sequence of functions denoted by $y_N(x)$ for a specific application is known the sequence of optimal allocation $(\bar{y}, \bar{y}_1, \ldots, \bar{y}_{N-1})$ can be computed by means of 2.31 provided that N and x are known.
Let's recapitulate. We started out with an single N-stage allocation problem and concluded with the transformation into a sequence of N one-dimensional problems. Furthermore as promised in the introduction we succeeded in embedding the initial problem for specific values of N and x within a family of related problems (i.e. arbitrary quantities of N and x). Likewise the obvious advantages concerning computational effort of dynamic programming compared to the common numeric approach should not go unmentioned.

2.3. Dynamic programming of stochastic target functions

We will now turn our introductory discussion of the dynamic programming method to a class of functions more suitable to represent projects under uncertainty - the stochastic

multi-stage decision process.
In contrary to the deterministic case the outcome of stochastic processes is not solely dependent from the operators decision but also influenced by chance in a varying degree. Instead of resulting in a predetermined outcome a decision constitutes or biases a distribution of possible outcomes. This leads to a core-question of optimum decision-making in stochastic processes that is: *'Which mathematical property is chosen as indicator of the results optimality? Since the lack of full control over the process effectively prevents the guaranteed receipt of a maximum return even for optimal policies.* The question of decision-making criteria is, for this case, rather philosophical than mathematical. One could select the policy with the maximum expected-value (commonly adapted) as well as choose a strategy with the highest minimum return (far less widespread). These are referred to as so called mini-max or maxi-min strategies.

2.3.1. A stochastic decision process example from Richard E. Bellman's *Dynamic Programming*

Let us now turn towards an example designed by Richard E. Bellman to introduce dynamic programming to processes of partially random nature. Imagine a gold-mining-procedure taking place in two different claims A and B containing the distinct amounts of gold x and y. Yet the owner of the mine only possesses one rather delicate prospecting-machine. Whenever the machine is put to use in mine A it has a chance of p_1 to excavate an relative amount r_1 of gold (resulting in a total of $r_1 \cdot x$) while on the other hand being faced with a probability $1 - p_1$ to be damaged beyond repair. The same principle is applied for mine B with probability p_2 and ratio r_2. The decision-makers task in this problem would be to determine the sequence of choices that maximized the (expected) amount of gold mined before the machine suffers a total loss.
While the problem might occur trivial for the one-period example seen in equation 2.32 the profits of mining are beyond any deterministic statement for the multi-period case since the remaining gold quantities x and y have to be readjusted after each successful mining step. Therefore and because of the immanent danger of machine destruction we will utilize the expected-value of profits to evaluate our policies. The optimum strategy will provide us with a return given by $\max E(r_1, r_2)$ before any damage to our machine occurs.

$$f_1(x, y) = \max[p_1 \cdot r_1 \cdot x, p_2 \cdot r_2 \cdot y] \tag{2.32}$$

To begin our investigation we assume that our mining process is terminated in any case after N-steps. This allows us to theoretically calculate and compare all possible decision sequences. This approach however doesn't yield us any information about the structure of the solution and is not that smart from a mathematical viewpoint. Considering a process with two decision alternatives over a ten-stage period we end up with $2^{10} = 1024$ possible strategies; if we, by chance, own a third mine this number explodes to $3^{10} = 59049$. With these thoughts in mind we can draw up two simple

estimations for the N+1 stage process. These are equation 2.33 as expected gold return for continued prospecting in mine A, and 2.34 as correspondence for claim B.

$$f_A(x,y) = p_1 \cdot (r_1 \cdot x + f_N((1-r_1)\cdot x, y)) \tag{2.33}$$
$$f_B(x,y) = p_2 \cdot (r_2 \cdot y + f_N(x, (1-r_2)\cdot y)) \tag{2.34}$$

This leads us to a basic N+1 stage approximation given by equation 2.35 and an infinite stage estimation described by eq. 2.36 for the unbound process.

$$f_{N+1}(x,y) = \max[f_A(x,y), f_B(x,y)] = \max[r_1 \cdot x + f_N((1-r_1)\cdot x, y), r_2 \cdot y + f_N(x,(1-r_2)\cdot y)] \tag{2.35}$$

$$f(x,y) = \max[r_1 \cdot x + f((1-r_1)\cdot x, y), r_2 \cdot y + f(x,(1-r_2)\cdot y)] \tag{2.36}$$

As before the infinite process is only an approximation towards finite solutions with large N. This is due to diminishing returns that lead to a convergence of the final result. Two rather simple estimations contained within our policy space are those for an infinite series of decisions either for mine A or B. In the case of $P_1 \cdot p_2 > 0$ both strategies show monotone convergence and lead to the results given in equation 2.37 for mine A and eq. 2.38 for mine B. These are the gold profits if the decision maker chooses mine A or B and sticks to his or her decision forever.

$$f_A(x,y) = \frac{p_1 \cdot r_1 \cdot x}{(1 - p_1 \cdot (1-r_1))} \tag{2.37}$$
$$f_B(x,y) = \frac{p_2 \cdot r_2 \cdot y}{(1 - p_2 \cdot (1-r_2))} \tag{2.38}$$

But how should we proceed and evaluate other approaches? A graphical illustration of the problem can provide us with some intuition towards a possible solution. But first let us state anyone will prefer prospecting of mine A so long $\frac{x}{y} >> 1$ and vice versa stick to mine B if $\frac{y}{x} >> 1$ holds (Of course under the precondition that $0 \leq p_1, p_2 \leq 1$). Therefore we could argue that the strategic decision depends solely on the ratio of $\frac{x}{y}$ since $f(kx, ky) = kf(x,y)$ for $k > 0$. If we imagine the amounts of gold x and y located within the mines as coordinates in a coordinate systems positive quadrant we could establish regions (subsets) where a decision for mine A or B is preferable as well as (at least) one border line where none of both options is superior. Let us assume that only two regions exist and that equations 2.39 and 2.40 provide us with the knowledge of the returns of a first decision for either case A or B. The first term is the expected immediate profit whereas the second term indicates the amounts of gold still to be mined presupposed that the machine is still undamaged.

$$f_A(x,y) = p_1 \cdot r_1 \cdot x + p_1 \cdot f((1-r_1)\cdot x, y) \tag{2.39}$$
$$f_B(x,y) = p_2 \cdot r_2 \cdot y + p_2 \cdot f(x, (1-r_2)\cdot y) \tag{2.40}$$

When equating these to equations as seen in 2.41 we obtain the exact mathematical description fo the bordering line which we will denote L. Unfortunately it still contains a functional term that can neither be ignored nor simple circumvented. Yet we can resolve this problem by adding another pair of observations to or repertoire of formulas. Consider that a decision for mine A while located on the border will decrease the amount of gold x stored within A while y for B remains constant and thus putting us in region B for the second decision. The same principle is true for the inverse case i.e. a first decision for B will lead to an A in the second stage if started from the border L.

$$f_A(x, y) = f_B(x, y) \tag{2.41}$$

With this insight gained we will try to construct the equations for the two-stage process and hopefully be able to eliminate the functional terms from these. Equations 2.42 and 2.43 show the to two-stage gold yields and to our relief we can equate them without obstacles (eq. 2.44).

$$f_{AB}(x, y) = p_1 r_1 x + p_1 p_2 r_2 y + p_1 p_2 f((1 - r_1)x, (1 - r_2)y) \tag{2.42}$$
$$f_{BA}(x, y) = p_2 r_2 y + p_2 p_1 r_1 x + p_2 p_1 f((1 - r_1)x, (1 - r_2)y) \tag{2.43}$$

$$f_A B(x, y) = f_B A(x, y) \tag{2.44}$$

From there it is only a stone's throw to the mathematical description of the bordering line L given by equation 2.45. This border however leads us to an interesting interpretation. Observe that these fractions directly compare the immediate gains $p_1 r_1 x$ of a decision versus the instantaneous losses $(1 - p_1)$. Obviously any rational investor will then settle for the option that maximizes the value of possible profits against corresponding losses i.e. the investor will choose claim A if $\frac{p_1 r_1 x}{(1-p_1)} > \frac{p_2 r_2 y}{(1-p_2)}$ and mine B otherwise. Assuming for instance that $p_1 = p_2 = p$ and $r_1 = r_2 = r$ the sequence of decisions is solely determined by the remaining amounts of gold in each mine. This intriguingly simple criterion tends to occasionally turn up in the dynamic programming theory.

$$L = \frac{p_1 r_1 x}{(1 - p_1)} = \frac{p_2 r_2 y}{(1 - p_2)} \tag{2.45}$$

2.3.2. Some thoughts on dynamic programming in stochastic processes following Dixit's and Pindyck's *Investment Under Uncertainty*

The book of Avinash K. Dixit and Robert S. Pindyck [5] is concerned with investment considerations in uncertain environments. They stress that time plays an important role for investment decisions. Due to its very nature. An investment constitutes a future stream of revenues that accumulate to a monetary amount that ideally should exceed the initial cost. These streams however are influenced by random events and future

actions of rival as well as the investor himself. To secure a decision's desired result the possible developments need to be estimated and evaluated whereupon the possibility of postponing a decision is of special interest. Dixit and Pindyck feature two alternate methods suitable for such an evaluation; the already well known dynamic programming approach and the contingent claims analysis which is not in the scope of this work. According to their description, dynamic programming is the decomposition of a (possibly large) sequence of decisions into just two components: the immediate decision's reward and a value function that contains the information about all subsequent decisions and their consequences. If the planning horizon is finite i.e. the problem has a definite terminal date we have the option to start with the very last sub-problem and apply it to standard static optimization methods. Afterwards the last period's result can be utilized to optimize the penultimate problem and so on. Obviously it is not possible to relocate the same approach to infinite stage processes. Yet the recursive nature of these problems helps to keep even these tasks relatively simple since solving the one period optimization leads to the exactly same problem again with just different starting conditions. Hence we are looking for a stationary solution. This not only facilitates efficient computing but also enables the decision maker to find and establish analytic solutions. We will now start with a two period example and from there develop a multi-period and afterwards an infinite stage dynamic programming method for stochastic processes.
Suppose that an investor has the opportunity to invest into the construction of a factory for a not further specified good. This production site whose price will be denoted by the sunk cost of investment I will produce one piece of its product for evermore upon completion. This item can be sold for a price P_0 in period 0. In the second time-step i.e. period 1 however the price will either rise or fall to a new level P_1 due to a stochastic market shift and will remain the same from there on. Equation 2.46 indicates the possible expected revenues.

$$P_1 = \begin{cases} (1+u)P_0 & \text{with probability } q \\ (1-d)P_0 & \text{with probability } (1-q) \end{cases} \tag{2.46}$$

Furthermore let us assume that all future revenues must be discounted with the risk-less interest r and that the investment can only be carried out in period 0 and there is no possibility to catch up afterwards. With these informations a decision-maker can calculate the value of investment V_0 as shown in equation 2.47.

$$\begin{aligned} V_0 &= P_0 + [q(1+u)P_0 + (1-q)(1-d)P_0]\left[\frac{1}{1+r} + \frac{1}{(1+r)^2} + \dots\right] \\ &= P_0 + 1 + q(u+d) - dP_0\frac{1/(1+r)}{1-1/(1+r)} \\ &= P_0[1 + r + q(u+d) - d]/r \end{aligned} \tag{2.47}$$

With V_0 known computing the net-pay-off of the project Ω_0 to the firm is done according to equation 2.48. Remember though that an investor is only willing to allocate money

to the factor if $V_0 > I$, otherwise the project will be abandoned.

$$\Omega_0 = \max[V_0 - I, 0] \tag{2.48}$$

Let us now presume that the investment opportunity remains intact throughout period 0 and that an investor might wait for the markets development and choose whether to buy the factory or not afterwards. If so we could put our dynamic programming knowledge to use and calculate the value function in period 1 as seen in equation 2.49. Note however that we just use generic variables and have not yet considered the random behaviour of prices.

$$\begin{aligned} V_1 &= P_1 + \frac{P_1}{(1+r)} + \frac{P_1}{(1+r)^2} + \dots \\ &= P_1 \frac{1+r}{r} \end{aligned} \tag{2.49}$$

According to our earlier investigations the value of the option to invest in stage 1 will therefore equal 2.50. Since P_1 can take on two different values the same holds true for V_1 and F_1. Therefore we need to compute an expected value of F_1 as shown in eq. 2.51 in order to reasonable compare the investment opportunities.

$$F_1 = \max[V_1 - I, 0] \tag{2.50}$$

$$\mathcal{E}_0[F_1] = q \max[(1+u)P_0 \frac{1+r}{r} - I, 0] + (1-q) \max[(1-d)P_0 \frac{1+r}{r} - I, 0] \tag{2.51}$$

With the results for period 1 we can proceed to investigate the options in stage 0. All we need to do is take our expected value for waiting, apply the discount rate since it comes into effect one period later and put it in place of the former waiting revenue which was zero. The value of F_0 calculated in eq. 2.52 might differ from the initial Ω_0 if the possibility to wait yield larger profits than an immediate investment. This difference can be referred to as the value of the opportunity to wait.

$$F_0 = \max \left\{ V_0 - I, \frac{1}{1+r} \mathcal{E}_0[F_1] \right\} \tag{2.52}$$

Let us now generalize the two-period example into a method for an unspecified finite number of periods. We will stick to the discrete time setting and use Markov processes as source of stochastic behaviour. The current status of the investment, project, process or any other subject to investigation is indicated by the state variable x that is known for any period or time-step t up to the present one. Future states $x_{t+1}, x_{t+2}, \dots$ are considered random variables under our assumption of Markovian behaviour. There is however one or more control variable(s) i to be chosen be the decision-maker that do not only affect the immediate profit flow $\pi_t(x_t, i_t)$ but also the cumulative probability distribution of future states $\Phi_t(x_{t+1}|x_t, i_t)$. Again we let a discount factor $1/(1+r)$ affect

decisions in later periods, with r indicating the risk-less interest rate. Also the whole project or process is ended at an specific termination date T where a final pay-off $\Omega_T(x_T)$ is distributed. As before our aim is to maximize the total revenue over the whole time horizon.
Similar to the two-period example the dynamic programming algorithm starts at the very end of the time span and the two-part structure of the solution i.e. immediate reward and value function of future decisions in a node remains intact as seen in eq. 2.53.

$$\pi_t(x_t, i_t) + \frac{1}{1+r}\mathcal{E}_t[F_{t+1}(x_{t+1})] \tag{2.53}$$

A decision-maker will strive to maximize the value of his opportunities. Therefore $F_t(x_t)$ denotes the net-present-value of all cash-flows provided that the operator selects all decisions optimally from period t onwards. Equation 2.54 shows the general Bellman Principle of Optimality for all nodes except for the last decision. In words: *An optimal policy has the property that, whatever the initial action, the remaining choices constitute an optimal policy with respect to the subproblem starting at the state that results from the initial actions.*[5]

$$F_t(x_t) = \max_{i_t}\left\{\pi_t(x_t, i_t) + \frac{1}{1+r}\mathcal{E}_t[F_{t+1}(x_{t+1})]\right\} \tag{2.54}$$

We demanded earlier that a termination pay-off will be allocated in the final period. Thus we need to establish a slightly different Bellman equation for the penultimate stage given in eq. 2.55. Here the investor's goal is to maximize the profits from immediate rewards and discounted expected terminal pay-off.

$$F_{T-1}(x_{T-1}) = \max_{i_{T-1}}\left\{\pi(x_{T-1}, i_{T-1}) + \frac{1}{1+r}\mathcal{E}_{T-1}[\Omega_T(x_T)]\right\} \tag{2.55}$$

In dynamic optimization tasks with infinite horizon we face a new problem. Since there is no final stage to work backward from one cannot easily determine a value function for the penultimate stage. While this may seem hugely impractical it is not that troublesome in reality since the problem takes on a recursive structure that facilitates both theoretical as well as numerical analysis. Let us boldly state that an infinite problem horizon leads to nothing else than independence from time as such provided that there are no explicit time dependencies e.g. seasonality. The importance of state variable(s) x_t remains intact though but it does not matter when they occur any more. Assume that we already found a value function common to all periods that is only evaluated at different points x_t. Then we are able to postulate the general infinite horizon Bellman equation in 2.56. Note how we are able to strip the time label from the value function since it is commonly shared for all periods.

$$F(x_t) = \max_{i_t}\left\{\pi(x_t, i_t) + \frac{1}{1+r}\mathcal{E}_t[F(x_{t+1})]\right\} \tag{2.56}$$

We could generalize this equation even more by removing all time labels and replace x_t and x_{t+1} with x and x' since time as such is not of concern any more. Let us refer to this as infinitely, repeating or recursive bellman problem (Eq. 2.57).

$$F(x) = \max_i \left\{ \pi(x,i) + \frac{1}{1+r} \mathcal{E}[F(x')|x,i] \right\} \tag{2.57}$$

This measure leaves us with only one problem that is where to start, i.e., which value function to use. Luckily though, the whole complex can be regarded as functional equation with the function F as its unknown. It can be mathematically proven that one could start with any guess of the right function and the discount term lets all wrong components vanish over time and therefore convergence is guaranteed no matter how bad the initial guess was. Therefore only the correct solution will be left after a number of iterations dependent on the scale of the (positive) discount factor. Note however that this method requires the boundedness of profit flows in order to force the contraction of wrong solution candidates. F can therefore be considered a fixed point of the overall problem. A short, intuitive proof can be found in [5] chapter 4 appendix A.
With this general introduction on dynamic programming being said let us proceed to the examples that were developed to verify the suitability of the method for evaluation of projects under cost uncertainty.

3. Example A1 - Dynamic programming of project investment under uncertainty

3.1. Project representation and variables

Consider a firm with a single development project with an estimated cost to completion of P_t. At any point in time $t = 0, 1, \ldots, T$ the firm decides about the flow of investment I_t which allocates an amount of $I_t \Delta t$ to the project over the next time interval. While investing the amount $I_t \Delta t$ reduces the required costs to completion in expectation by $I_t \Delta t$, development costs are subject to random shocks which are assumed to follow a geometric Brownian motion modelled on a binomial tree. Thus, the transition equation satisfies $E(P(t+1)) = P(t) - I_t$ with $\lim_{\Delta t \to 0} \frac{var(P_{t+1})}{\Delta t} = \beta^2 P_t$.
Furthermore the discussed project is represented by a number of inherent model parameters shown in table 3.1 and control variables seen in 3.2. The difference between these two kinds of values is that control variables can, within certain limits, be chosen by the operator according to strategic decisions while the inherent type is predetermined by the project itself or its environment and stays fixed throughout the optimization process.

formula	source code	value	unit
P_0	priceGuess0	2000	[monetary units]
P_R	rewardMargin	1	[monetary units]
λ	lambda	$\frac{P_R}{P_0}$	[-]
φ	shift	0	[-]
β	beta	0.20	[-]
r	r	0.05	[-]
T	Tmax	10	[time units]
R	reward	3000	[monetary units]

Table 3.1.: Project inherent variables

Model parameters P_0 is the initial estimation of the projects realization cost. If the cost cannot be assessed in a single number of monetary units the variable φ can be used to build a tree that starts not from a single node P_0 but from a whole range of $\varphi + 1$ nodes centered around P_0. P_R represents a specific cost margin below which a reward

formula	source code	value	unit
I	invest	{user}	[monetary units / time unit]
$\frac{I}{P(t)}$	investQprice	$\frac{I}{P(t)}$	[1 / time unit]
Δt	dt	{user}	[time units]
t_{Count}	tCount	$\frac{T}{\Delta t}$	[-]

Table 3.2.: Project control variables

R is to be allocated i.e., if the remaining project cost to completion is at or below P_R, the project is regarded as completed and the final reward is awarded. λ is a quotient of P_R and P_0 necessary to find a suitable grid interval for the computation (see sec. 3.2.3). T marks the allowable time horizon within which the project has to be finished if a reward is to be distributed. To conclude, there are r, which is the risk-less rate of interest by which future earnings and expenses are discounted alike, and β, that comes from the projects risk assessment and acts as the variance parameter of the Brownian motion underlying the model.

Control variables The outcome of our model can be influenced by selecting values for two control variables. First, there is I which is the amount of money an investor is willing to spend on the project when faced with a distinct cost $P(t)$. The quotient of these two is $\frac{I}{P(t)}$. Second there is Δt which defines the spacing of the computational lattice and therefore determines the number of discrete time-steps t_{Count} and in conjunction with φ the amount of nodes to be optimized. The variable Δt has to fulfil certain requirements discussed in sec. 3.2.3 for the model to work properly. Furthermore the product of $I \times \Delta t$ is the specific amount of monetary units invested at a given node in the models grid.

3.2. Modelling of a project under uncertainty

3.2.1. Geometric random walk of estimated cost

We begin our study with the single variable P_0 which is the assessed cost (or base for an estimated range of possible prices) of the project at time t = 0. At every subsequent point in time this initial price guess (and the price guesses that have forked from it) can either rise or fall to a new estimation of remaining cost $P(t)$. This is due to stochastic events that can't be directly influenced by the project management and modelled by a geometric random walk with an underlying Wiener process. The mentioned stochastic behaviour is represented by the probabilities π for an up-move (i.e. a rise in expected costs to completion) and $1 - \pi$ for a down-move (i.e. a decline in expected costs to completion), both depending on the individual project's risk evaluation (β factor) and the management's investment decisions ($I\Delta t$). Equation 3.1 indicates the calculation of the up- and down- multiplier. These factors are then multiplied with the existing cost

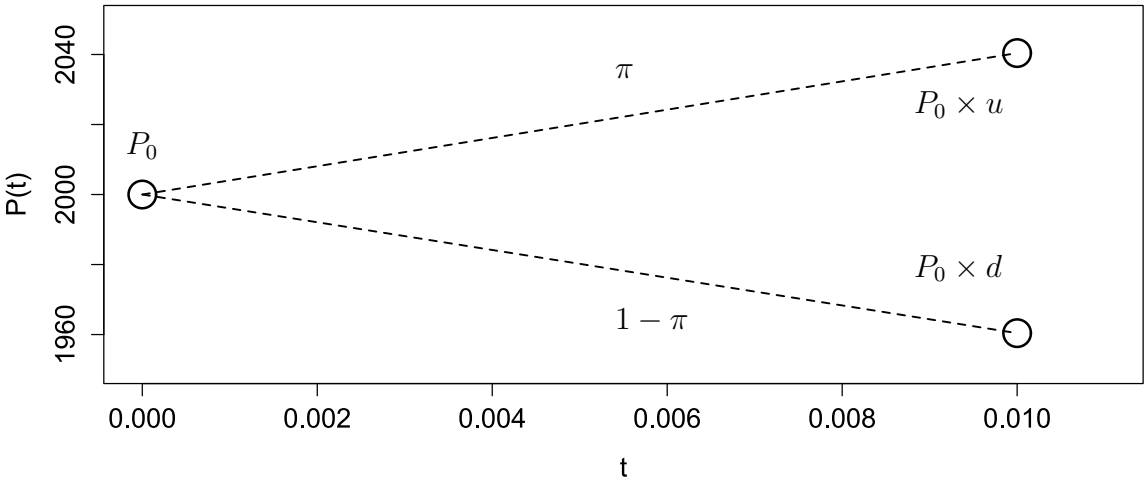

Figure 3.1.: Basic bifurcation of P_0 into two price possibilities

estimation $P(t)$ and generate to new guesses of expected cost to completion $P(t+1) = P(t) \times u$ and $P(t+1) = P(t) \times d$. See figure 3.1 for the basic bifurcation model.

$$u = e^{\beta \times \sqrt{\Delta t}} \tag{3.1}$$
$$d = u^{-1} = e^{-\beta \times \sqrt{\Delta t}}$$

Since cost after the up- or down-moves is only dependent on current expected cost to completion, assessed project risk and amount of time spent between the review points and the formulas contain only multiplications, the commutative property applies and all forks of expected cost in the geometric random walk form a recombining tree (fig. 3.2) similar to those described by Cox, Ross and Rubinstein in their Binomial options pricing model [3]. Our investment problem can furthermore be labelled a Markov decision process since it is is not of concern how a specific node was reached but only if it is profitable to invest at the node's expected cost to completion or not. Therefore an important requirement for the dynamic programming method's backward induction is met.

For a geometric random walk unaffected by investment the probabilities π and $1 - \pi$ are defined solely by the step-sizes u and d. This can be proven by solving $P(t) = \pi \times P(t)u + (1 - \pi) \times P(t)d$ for π under the assumption that $P(t) = E(P(t+1))$. For the driftless geometric random walk it turns out that $\pi = \frac{1-d}{u-d}$. Figure 3.1 illustrates the split of a single node into two successor nodes whose expected value remains just equal to their origin.

The dashed orange line in fig. 3.2 that originates in P_0 divides probability spaces of the

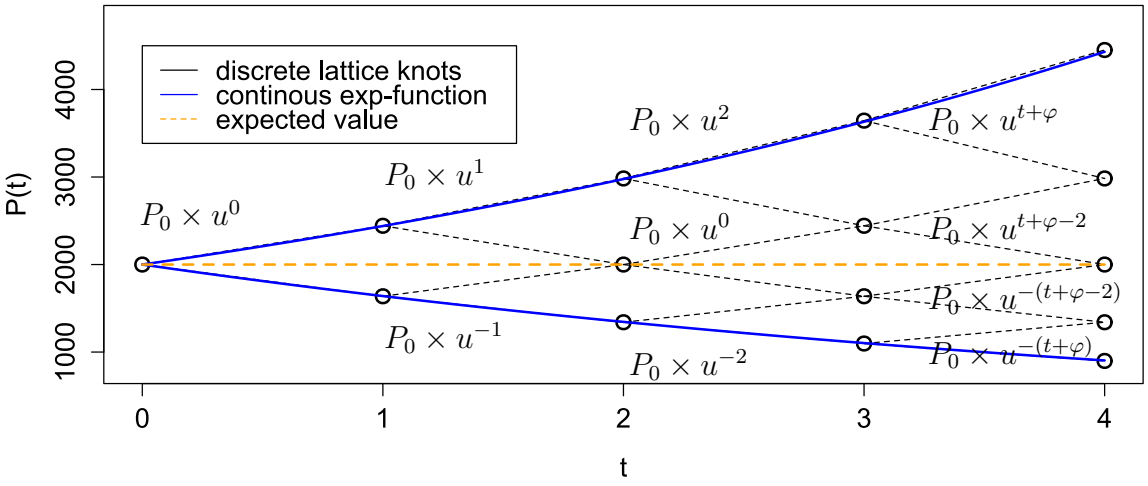

Figure 3.2.: Discrete geometric random walk of P_0 compared to an continuous exponential function of P_0

resulting log-normal distribution in two parts. In the limit of $\Delta t \to 0$ these will both converge to a probability of 0.5. In a discrete setting however the chance π for an up-move will recede but the same number of lattice nodes will sprout above and below that centreline. While the lower half converges against 0 and thus concentrates its probability density, the upper part end strives for $+\infty$ and therefore spreads its probability density over a wide range of possible values. For discrete values of Δt the expected value of the remaining cost to completion distribution will thus shift slightly.
Note how the number and value of the expected cost forks is tied to the time-step they correspond to, i.e., the power of u is represented by a sequence ranging from -(t+φ) to +(t+φ) in increments of 2. For instance for the third time-step with $\varphi = 0$ there are four forks whose powers are -3,-1,+1,+3. The parameter φ can be used to build a tree that starts not from a single node P_0 but from a whole range of $\varphi + 1$ nodes centered around P_0.

3.2.2. Introducing investment activity

Now that the binomial tree's cost values can be calculated in every node we turn toward the modelling of our investment problem. The selection of the control variable $I \times \Delta t$ should reduce the expected remaining project cost for the next time step by exactly the amount spent. Let $P(t)$ denote the current estimated project cost and $I \times \Delta t$ the investment at a discrete step in time. After the passing of a time Δt the price of the project should be $P(t) \times u$ with probability π and $P(t) \times d$ with probability $1 - \pi$. We equate these two statements into the single equation 3.2 which will be the basis for

our investment decisions. Since the cost values of the geometric random nodes cannot be influenced directly our investment activity affects the values of π and $1-\pi$ so that $P(t) - I \times \Delta t = E(P(t+1))$ holds. Thus investment imposes a drift in the stochastic Wiener process.

$$P(t) - I \times \Delta t = \pi \times P(t) \times u + (1-\pi) \times P(t) \times d = E(P(t+1)) \tag{3.2}$$

Solved for π equation 3.2 yields the relationship of investment and stochastic behaviour we have been looking for, as seen in equation 3.3. If the investor decides to wait (and Δt is sufficiently small) chances for an up- or down-move of the expected cost should converge towards $0.50 : 0.50$. Should we on the other hand want to invest, the probability π can assume any value < 0.5 including negative values. Therefore it is important to ensure within the program that π must not become smaller than 0 in order to correspond with real probabilities (i.e $\pi = \max[\pi, 0]$).

$$\pi\left(\frac{I}{P(t)}\right) = \frac{1 - \frac{I}{P(t)} \times \Delta t - d}{u - d} \tag{3.3}$$

3.2.3. Concerning variable constraints

In order to carry out the dynamic programming of investment behavior, two variables must be chosen by human beings. First, the grid resolution represented by the discrete time interval Δt needs to be chosen by the analyst and in order to minimize the impact on the optimization results. Second the investor decides the relation of money invested to the proposed project cost namely $\frac{I}{P(t)}$. Although these variables are user selected they have widespread effects on the whole system of equations and therefore precautions have to be applied. Starting with Δt the requirements in the resolution of the geometric random walk will be discussed before we conclude with the relationship of $\frac{I}{P(t)}$, π and real probabilities.

Lattice spacing Δt

Naturally every project has a finite time horizon, in this case given by T. The discrete dynamic optimization at hand allows the selection of a variable Δt which represents the spacing in the computational grid and thereby dictates the quantity of nodes. The number of available nodes $t_{CountAv(ailable)}$ on the time axis is given by equation 3.4.

$$\frac{T}{\Delta t} = t_{CountAv} \tag{3.4}$$

$$z \cdot \sum_{k=1}^{t_{CountAv}} k = z \cdot \frac{t_{CountAv} \cdot (t_{CountAv} + 1)}{2} \tag{3.5}$$

Due to the structure of the recombining tree, the number of operations scales with the (small) Gaussian sum formula eq. 3.5 and is of the order $O(t^2_{CountAv})$. The value of z is

the count of operations within a single node and thus dependent on the complexity of the investment decision. Considering this possibly large number of optimization problems, it appears desirable to choose Δt as large as possible in order to keep the computing cost low. Yet there is another condition that Δt has to fulfil.
To be more specific: In order to allow the distribution of the project reward R, a certain critical remaining cost level > 0 must be within reach of the lattice. Let λ denote the quotient of the critical value against the initial cost estimation and equal it to an unknown number of consecutive exponential contraction steps (down-moves). Solving equation 3.6 for $t_{CountReq(uired)}$ yields the minimum number of nodes on the time axis to get – at the tree – a positive overall probability that the project can be completed.

$$\lambda = \frac{P_R}{P_0} \equiv e^{-\beta\sqrt{\Delta t}\cdot t_{CountReq}} \tag{3.6}$$

$$-\frac{ln\lambda}{\beta\sqrt{\Delta t}} = t_{CountReq} \tag{3.7}$$

The formulas 3.4 and 3.7 can be used to construct an analytical function which, when solved for Δt leads to the exact largest tolerable Δt_{Max}. For this purpose we subtract the required from the available nodes and let the whole term equal null. Eventually Δt_{Max} is given by eq. 3.9. The functions for $t_{CountAv}$ given in eq. 3.4 and $t_{CountReq}$ seen in 3.7 are illustrated in fig. 3.3. While the left diagram illustrates all possible values of Δt from 0 to T, the right figure is centred around and highlighting the intersection of the two functions. The model parameter Δt has to be chosen smaller or at most equal to Δt_{Max} if the model shall produce valuable results.

$$T/\Delta t_{Max} - (-ln\lambda/\beta\sqrt{\Delta t_{max}}) \equiv 0 \tag{3.8}$$

$$\Delta t_{max} = \left(\frac{-T\cdot\beta}{ln\lambda}\right)^2 \tag{3.9}$$

Investment ratio $\frac{I}{P(t)}$ and real probabilities in π

The policy variable $\frac{I}{P(t)}$ is the mathematical expression of the operators investment strategy. It can be kept stable at the same value as project cost information unfolds or changes if for instance investments are carried out at a constant rate. The purpose $\frac{I}{P(t)}$ is to affect the random walk's stochastic behaviour by manipulating the probability π. Starting from $\pi \sim 0.5$ for $\frac{I}{P(t)} = 0$ the probability dwindles as investments rise and unfortunately without any measures taken π can adopt values beyond the real probability space $\epsilon[0,1]$ and thus lead to deceptive NPV results. Here, two approaches are possible. Both are employed at different stages throughout the development of this work. In this chapter we chose to coerce π to 0 if it stretches to far (see eq. 3.11) and regard the overspending as diminishing returns to investment. When switching to dynamic programming of net utility in chapters 4 and 5 we will limit $\frac{I}{P(t)}$ to a number

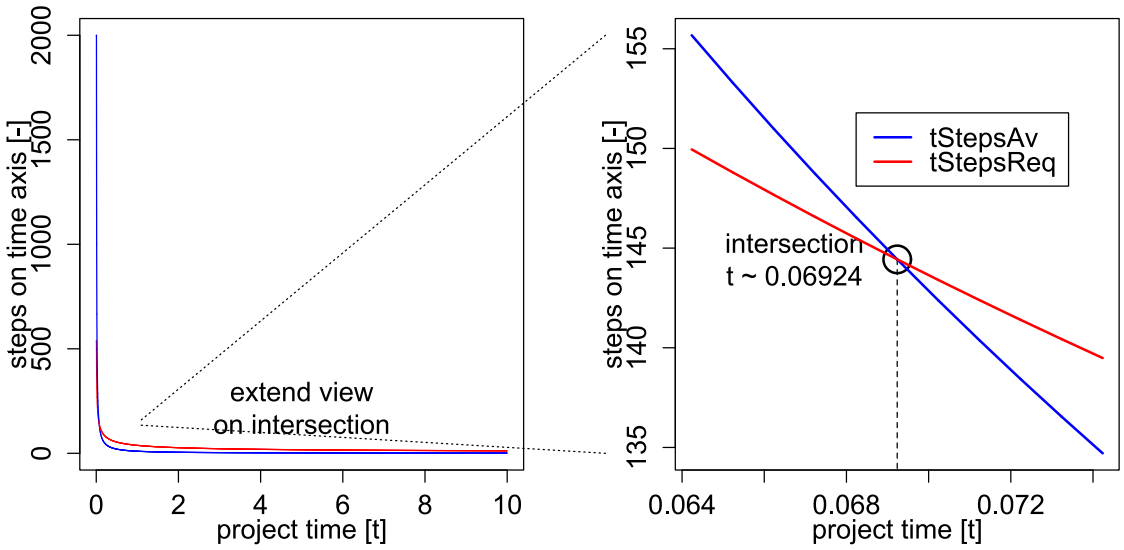

Figure 3.3.: intersection $t_{CountAv}$ and $t_{CountRe}$

given by eq. 3.10 in order to attain well behaved functions for the optimization function, i.e., prevent the final reward from ramping up due to negative probabilities and keep the number of boundary conditions small. With the model parameters of our showcase the limit for $\frac{I}{P(t)}$ would be ~ 2. Figure 3.4 diagrams the development of π over the investment ratio $\frac{I}{P(t)}$ with the mentioned root at ~ 2 and the origin at ~ 0.5.

$$0 \equiv \frac{1 - \frac{I^*}{P(t)} \times \Delta t - d}{u - d} \qquad \frac{I^*}{P(t)} = \frac{1 - d}{\Delta t} \tag{3.10}$$

$$\pi(\frac{I}{P}) = \frac{1 - \frac{I}{P(t)}\Delta t - d}{u - d}, \qquad \pi(\frac{I}{P}) \in [0, 1] \tag{3.11}$$

3.3. Applying the Bellman principle

After setting up all inherent project parameters and control variables according to real world and mathematical constraints we are now able find a number of nodes that satisfy one of the following two conditions

- Projects not finished on deadline must not yield any profits. Therefore the value of any node at t = T whose estimated cost is above the reward margin P_r is $V\{P, T\} = 0$

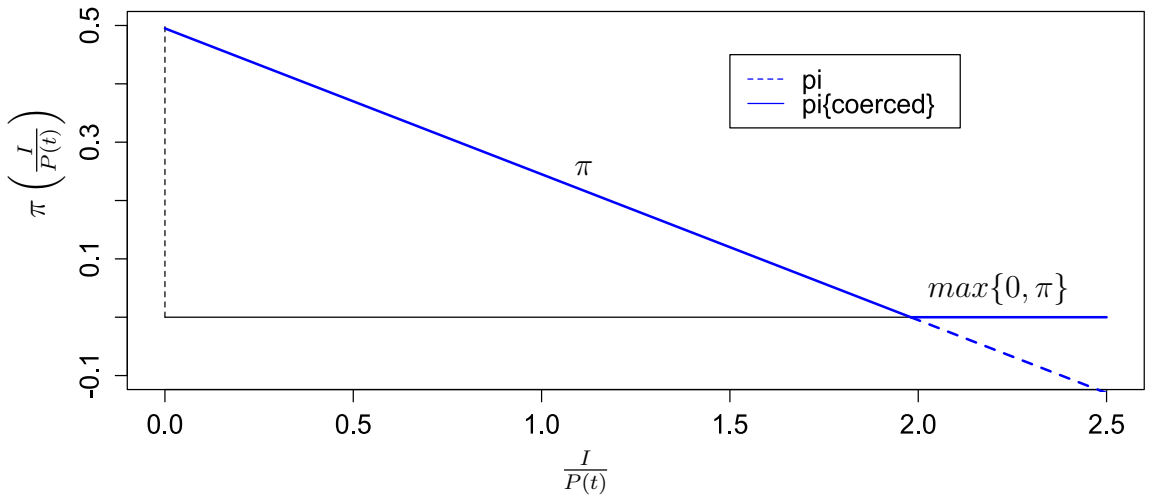

Figure 3.4.: value of probability π linearly dependent on $\frac{I}{P(t)}$

- Projects whose estimated cost to completion is below the reward margin regardless of the time step reviewed allocate their reward. Thus it is possible to finish the project precociously. The value of the option to invest in such a project is equal to the reward i.e. $V\{P_r, t\} = R$

For all other nodes in the discrete computation lattice a basic Bellman equation (3.12) has to be set up that will be used to optimize investment behaviour by backward induction starting from t = T.

$$V\{P, t-1\} = \max_{I \in \mathbb{R}^+}[-I\Delta t + \frac{1}{(1+r)^{\Delta t}} \times [\pi(\frac{I}{P})V\{P \times u, t\} + (1-\pi(\frac{I}{P}))V\{P \times d, t\}]] \quad (3.12)$$

As the value of the option to invest $V\{P, t-1\}$ profits linearly from investment I (equation 3.11) until π is coerced, maximizing the value of $V\{P, t-1\}$ becomes a decision of whether to wait or invest at the full rate the currently employed strategy allows. This can be observed in equation 3.13 as the Bellman equation's gradient is not dependent on the invested sum I. In the underlying implementation the decision is made by comparing the functional values for waiting and full-rate investment. Another possibility is to evaluate eq. 3.13 at 0 and, if positive, assign the maximum tolerable investment, otherwise wait. In consequence of the three different states that have to be considered a case structure is used to aid the net-present-value (NPV) calculation by extending the basic Bellman equation. Formula 3.14 shows the extended dynamic programming model used within the R program. In the upcoming sections three different strategies, their effects on the optimization and their overall turnouts will be discussed. The strategies

are labeled A1-0 for the case of permanent waiting, A1-1 for constant relative and A1-2 for constant absolute investment rates.

$$\nabla V = \frac{\delta V}{\delta I} = -\Delta t + \frac{1}{(1+r)^{\Delta t}} \times \frac{\Delta t}{P(u-d)} \times [V\{P \times d, t\} - V\{P \times u, t\}] \tag{3.13}$$

$$V\{P, t-1\} = \begin{cases} 0 & , t = T \wedge P(t) > P_R \\ R & , t \leq T \wedge P(t) \leq P_R \\ \max\limits_{I \in \mathbb{R}^+}[-I\Delta t + \frac{1}{(1+r)^{\Delta t}} \times \ldots & \\ [\pi(\frac{I}{P})V\{P \times u, t\} + (1 - \pi(\frac{I}{P}))V\{P \times d, t\}]] & , t < T \wedge P(t) > P_R \end{cases} \tag{3.14}$$

3.4. Strategy A1-0 - non-investment

3.4.1. Investment considerations

Strategy A1-0 is only elaborated for explanatory reasons. The intent of this section is to introduce some useful concepts for the evaluation of the strategies A1-1 and A1-2. Thus we set the investment values to null as shown in eq. 3.15.

$$\begin{aligned} I(t) &= const. = \{0\} \\ \frac{I}{P(t)} &= const. = \{0\} \end{aligned} \tag{3.15}$$

Furthermore we will generate a discontinue curve above which the project has no chance to be finished successfully. Thus for P(t) exceeding this critical cost, it is optimal not to invest at all. To achieve this eq. 3.6 respectively 3.16 are slightly altered to equation 3.17. This yields a critical cost for each time-step that is represented by a red line within the dynamic optimization graphs. We will refer to this as the technical border.

$$\frac{P(t)}{P_R} = e^{(t_{CountAv} - \frac{t}{\Delta t}) \times \beta \times \sqrt{\Delta t}} \tag{3.16}$$

$$P_{dis}(t) = P_R \times e^{(t_{CountAv} - \frac{t}{\Delta t}) \times \beta \times \sqrt{\Delta t}} \tag{3.17}$$

3.4.2. Dynamic programming of investment

Running a dynamic programming process without any decisions yields the results illustrated in figures 3.5 and 3.6. Figure 3.5 confirms that no investment activity is taking place, while 3.6 indicates that the odds of attaining a reward is between 0% and 25

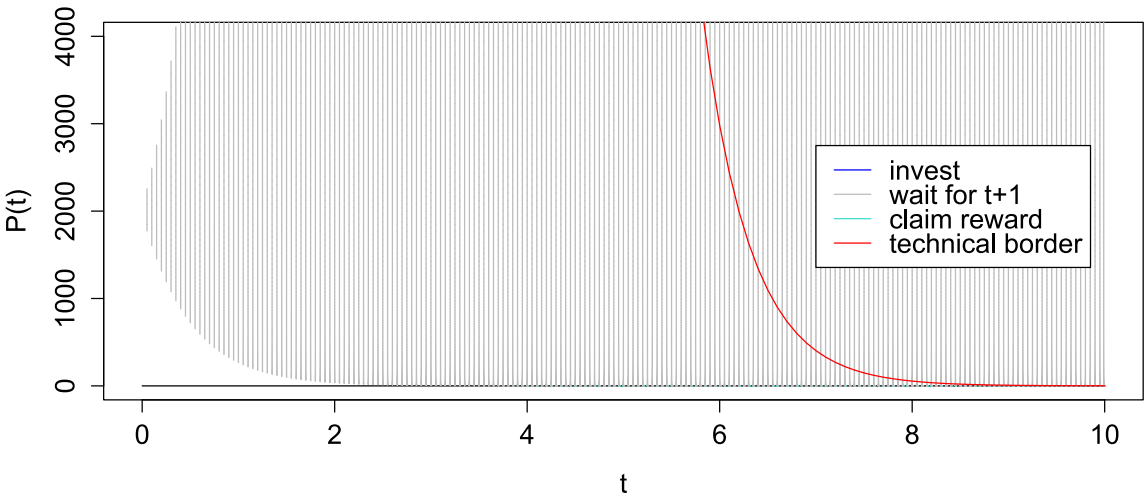

Figure 3.5.: Dynamic programming instruction results without any investments (A1-0)

%. While these graphs of A1-0 seem not very helpful, their computation provides results necessary for the Monte Carlo method simulation of cost-development and project turnout of non-investment i.e. waiting. The NPV (net-present-value) of the project on a time-interval of $\Delta t = 0.01$ without any investment considerations ($\frac{I}{P(t)} = 0$) is $V_{A1-0}(0) = 2.4393 \cdot 10^{-29}$ with a probability of successful project completion of $\Pi_{A1-0}(0) = 1.3164 \cdot 10^{-32}$. As intended the chance of attaining a reward without preceding investment is negligible small.

3.5. Strategy A1-1 - constant relative investment rates

3.5.1. Investment considerations

For strategy A1-1 we decide to keep the ratio of $\frac{I}{P(t)}$ constant at any arbitrary value larger than 0 for the investment and 0 for the waiting nodes. Waiting nodes are points in the geometric random-walk's lattice where, depending on individual remaining cost estimation and time, waiting appears financially more attractive than waiting. Because of the constant $\frac{I}{P(t)}$ we receive the same probabilities π_{act} and $1-\pi_{act}$ in any lattice node where investment takes place. The probabilities for waiting remain the same as in the previous case and are close to $\pi_{wait} \sim 0.5$ and $1-\pi_{wait} \sim 0.5$ with growing deviations for large Δt. Since we decided for a constant relative investment ratio, the absolute value has to be computed for every node in the grid (eq. 3.18). Also remember that either π_{act} must be coerced to a value ≥ 0 or $\frac{I}{P(t)}$ must be forced to $\leq \frac{1-d}{\Delta t}$ in order to compute

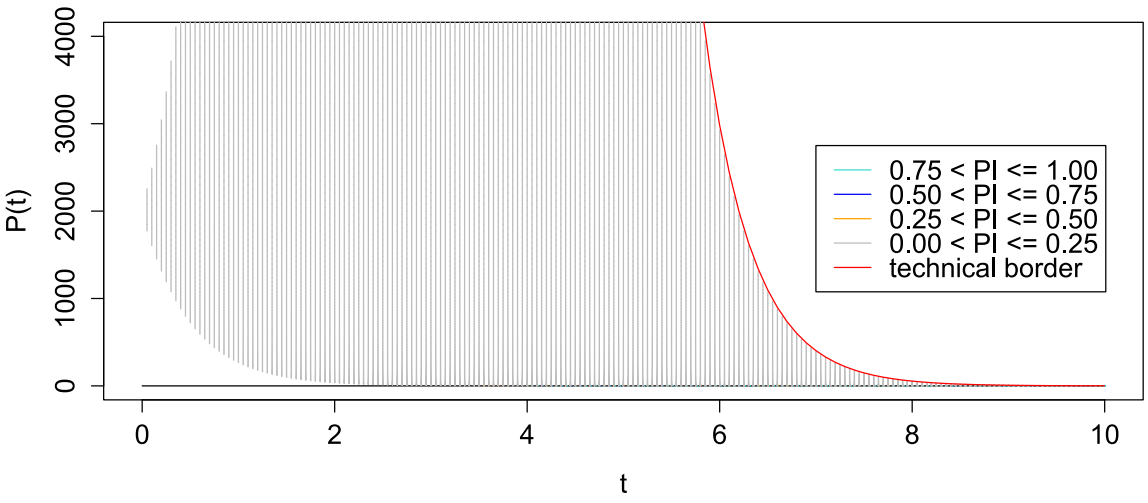

Figure 3.6.: Dynamic programming probability results without any investments (A1-0)

reasonable results. Figure 3.7 shows the development of $\frac{I}{P(t)}$ *(investQprice)* and $I(t)$ *(investment)* assuming $\beta = 0$, i.e., a motion without random shocks.

$$\begin{aligned} \frac{I}{P(t)} \quad &= const. \quad &= \{0, 1\} \\ I(t) \quad &= var. \quad &= \{0, P_0 \times u^n \times \frac{I}{P(t)} \times \Delta t\} \end{aligned} \tag{3.18}$$

3.5.2. Dynamic programming of investment

The NPV (net-present-value) of the project incorporating strategy A1-1, under parameters $\frac{I}{P(t)} = 1, \Delta t = 0.01$, is $\mathrm{V}_{A1-1}(0) = 297.20$ with a probability of successful project completion of $\Pi_{A1-1}(0) = 0.6545$. With constant relative investment rates figure 3.8 exhibits two distinct areas. On the one hand there are the waiting nodes, already seen before, colored in grey. On the other hand there is continuous blue zone that consists of nodes where investment appears optimal. Note the blue area's distinct bulge before it starts to decline towards the horizontal time axis. This is due to an endgame effect where the investor faces a make-or-break decision and may be willing to accept higher estimated remaining costs to completion than before. It turns out that the position and form of the hump is determined by a single parameter. It is the ratio of $\frac{I}{P(t)}$ which is directly related to the geometric random-walk's drift. It appears that for smaller i.e. less favourable downward movements investors prefer to transact the bigger investment

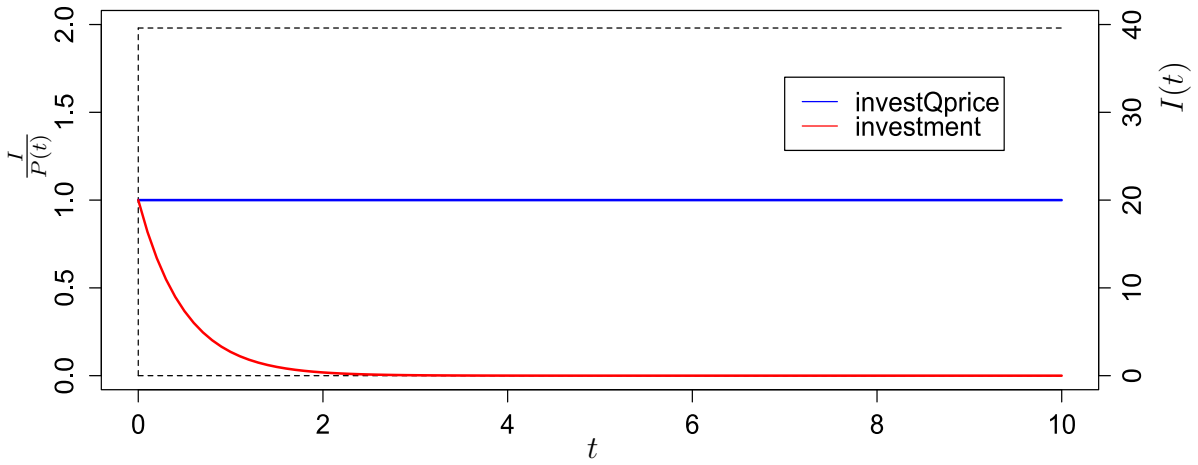

Figure 3.7.: Diagram of strategy parameters in A1: $\frac{I}{P(t)} = const., I(t) = var.$

chunks early on rather than wait and gamble the technical border. If however the imposed drift is large enough i.e. the investment can be considered quite 'secure' decision makers may be willing to invest on the edge of the technical feasibility. Additionally there are factors that may influence the visual appearance of the investment bump within the graph. Investigation shows that we also have to consider the computational grid's resolution Δt. While the time-step's size does neither affect the location nor the amplitude of the bulge it determines the nodes that lie on the technical frontier and thus the border may appear closer to or further away from the investment peak. This is a direct side effect of operating in a discrete calculation space. Figure 3.9 indicates the cumulative chance of actually finishing the project for every node in the lattice. We denote this probability Π(t) and observe that even for costs larger than our initial estimate P_0 there is a realistic chance to complete the project. In fig. 3.10 we see ten slices through the value ridge. The horizontal axis shows the movement within the discrete random-walk as remaining cost to completion of the project P(t). The location of points on the vertical axis represents the net present value of the option to invest in the project at any given time and cost to completion whereupon the point in time is visualized through distinct colouring. Slice t = 9.95 visualizes a sharp skip from an option value of zero to a peak of height R at the remaining cost where the payout margin P_R is located. The adjacent two slices for t = 8.95 and t = 7.95 also show extreme slopes while at the same time a first sign of smooth pasting into the coordinate axis is observed. Reaching out rightward from the a gentle curve below the peak there is a trailing edge. This edge indicates nodes where investment takes place and which are linked to two successor nodes that also recommend investment. Thus such a nodes value is only affected by discount and

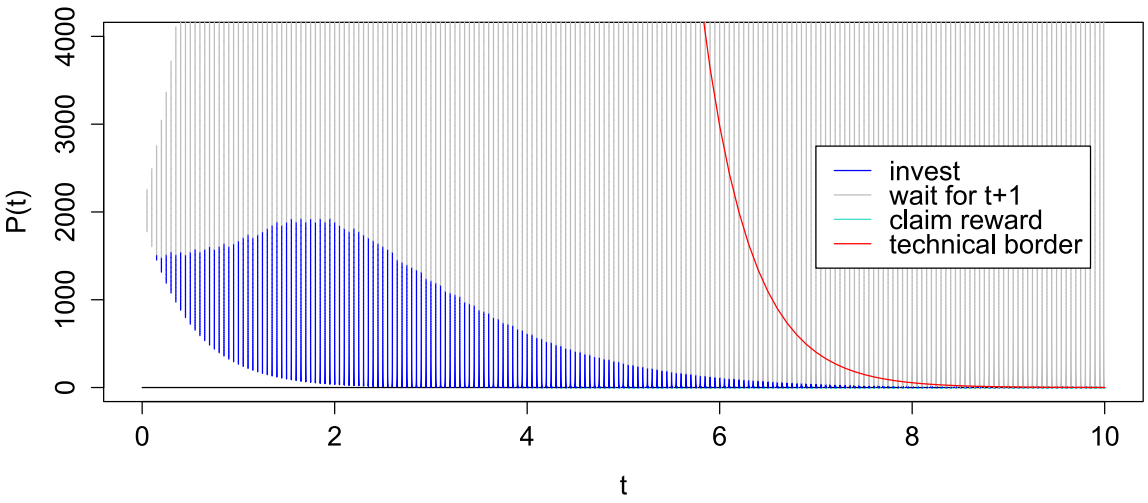

Figure 3.8.: Dynamic programming instruction results with constant relative investment (A1-1)

its individual cost while the probability of missing the reward is neglected. The value of nodes that don't have a successor inside of the investment area quickly dwindles into the zero plane as the curve bows downward and an unrewarded project must be taken into consideration. Note that while the nodes in slice t = 0.95 might not have investment successors directly they can maintain their only slight slope since the endgame bulge is still ahead.

3.6. Strategy A1-2 - constant absolute investment rates

3.6.1. Investment considerations

In strategy A1-2 we choose to settle for an absolute investment value per time-step $I(t)$ *(investment)* and keep it constant throughout the dynamic programming process (eq. 3.19). Therefore the probabilities π_{act} and $1 - \pi_{act}$ need to be re-evaluated at each and every node. From figure 3.11 we can deduce that this modus operandi quickly ends up in significant overspending. Less than two periods in the future, the ratio of investment to remaining cost $\frac{I}{P(t)}$ *(investQprice)* will exceed the maximum possible value that still results in real probabilities and will thus, as a consequence of our decision to limit the chances rather than the investment sum, for the most part vanish into thin air.

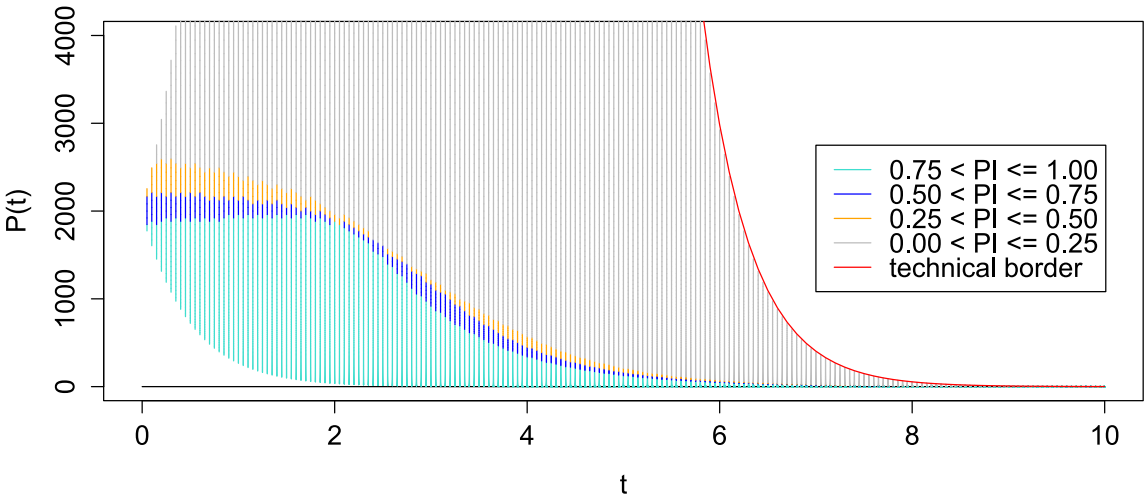

Figure 3.9.: Dynamic programming probability results with constant relative investment (A1-1)

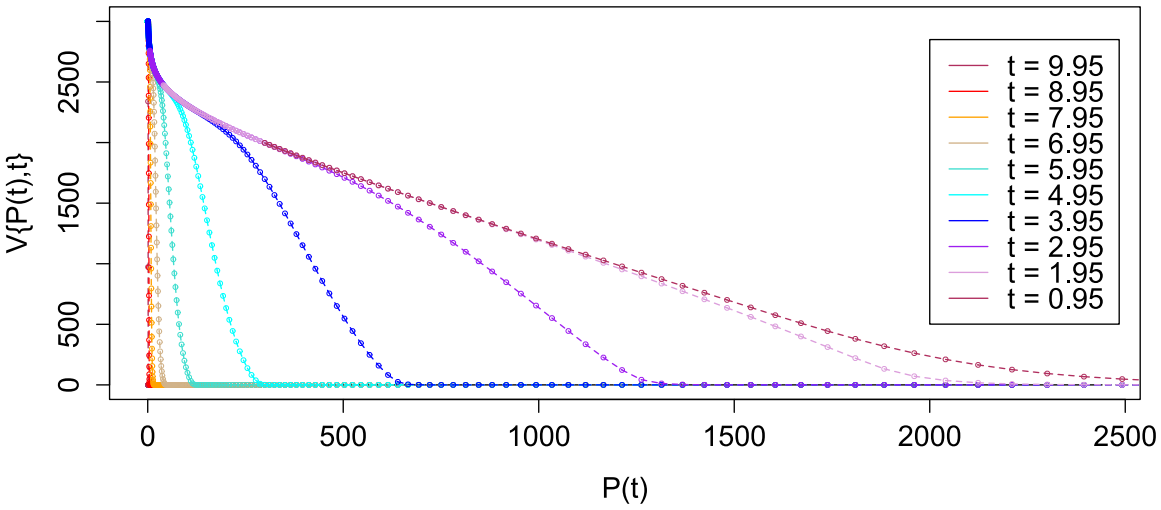

Figure 3.10.: Slices through the value ridge generated with constant relative investment (A1-1)

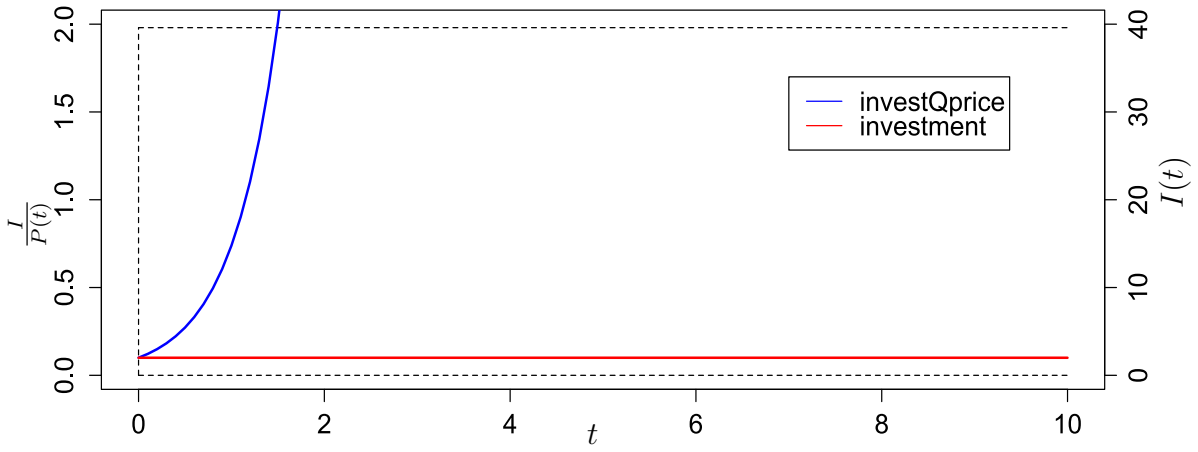

Figure 3.11.: Diagram of strategy parameters in A1-2: $I(t) = const.$, $\frac{I}{P(t)} = var.$

$$\begin{aligned} I(t) &= const. &&= \{0, \frac{P_0}{T}\} \\ \frac{I}{P(t)} &= var. &&= \{0, \frac{I(t)}{P_0 \times u^n}\} \end{aligned} \tag{3.19}$$

3.6.2. Dynamic programming of investment

The NPV (net-present-value) of a project receiving constant payment rates as suggested by strategy A1-2, under parameters $\frac{I}{P(t)} = 1, \Delta t = 0.01$, is $V_{A1-2}(0) = 83.23$ with a probability of successful project completion of $\Pi_{A1-2}(0) = 0.1429$. Incorporating constant absolute investment rates, the (blue colored) invest area of figure 3.12 is way smoother than seen in strategy A1-2. Intuition suggests that the time horizon investigated allows only the expression of the end-game decline and deprives us of the tolerable peak in remaining cost as well as the probably lower threshold earlier on the time horizon. Concerning cumulative probabilities of project completion Π figure 3.13 indicates that chances to complete the project are quite limited to the area where investment takes place. In fig. 3.14 we see ten slices through the value ridge. This time more sharp skips from zero value to the reward peak or the trailing edge after the curve can be identified. Due to over-spending in the lower (left) areas there's a sharp edge between nodes that lead to a reward and those that don't. The nodes which don't carry a chance of reward are identical to those that lie above the technical border in fig. 3.12. Smoothing occurs for t = 6.95 where the first waiting nodes appear inside the feasible area. With constant

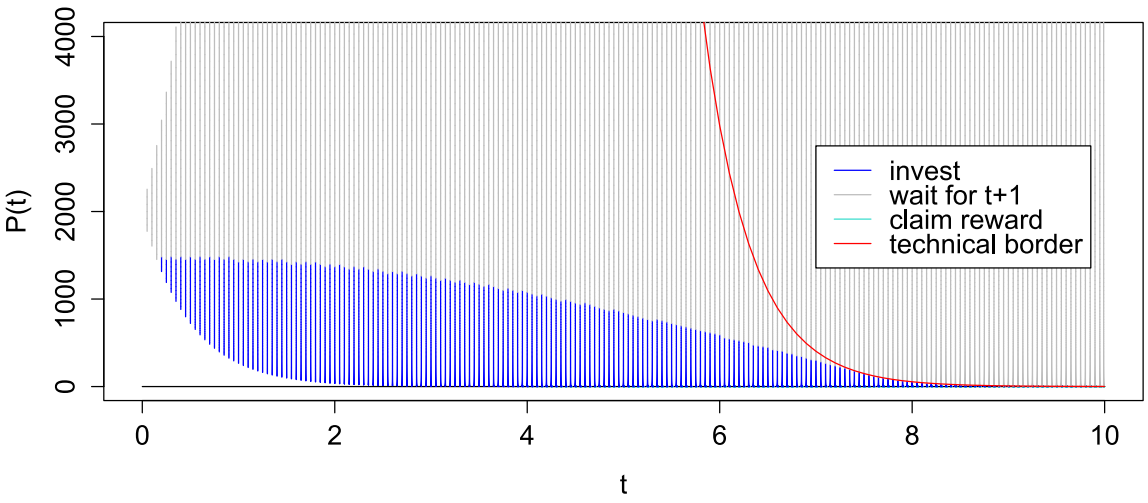

Figure 3.12.: Dynamic programming instruction results with constant absolute investment (A1-2)

rates of investment, at the value investigated the decline of the slope is slower and thus a larger time horizon T might result in a slightly higher option value $V_{A1-2}(0)$.

3.7. Monte Carlo method and comparison of project turnouts

In order to obtain an insight in the distributions of remaining costs to completion and cash-flows, i.e., financial turnouts of the project, a Monte Carlo simulation with sample size $n = 20000$ was carried out for each of the three strategies. Based on the result matrices for investment decisions and the deterministic probabilities for up- and down-movements of remaining cost, a Brownian motion of cost and cash-flow is simulated for every sample. Afterwards several statistical tests based on the produced samples were conducted on cash-flows and cost outcomes which can be found in appendix A. It could be proven that the uninfluenced geometric random walk strives towards an logarithmic normal distribution (fig. 3.15). In this short summary, however, we will limit ourselves to a short comparison and discussion of the results from dynamic programming and Monte Carlo simulation of risk-neutral investment.
Starting out with NPV and success probabilities, seen in table 3.3, we find that constant relative investment (A1-1) produces the highest expected net-present value and has a significantly higher chance of actually finishing a project. Constant absolute investment (A1-2) already marks a significant drop in value as well as conclusion likelihood. Success

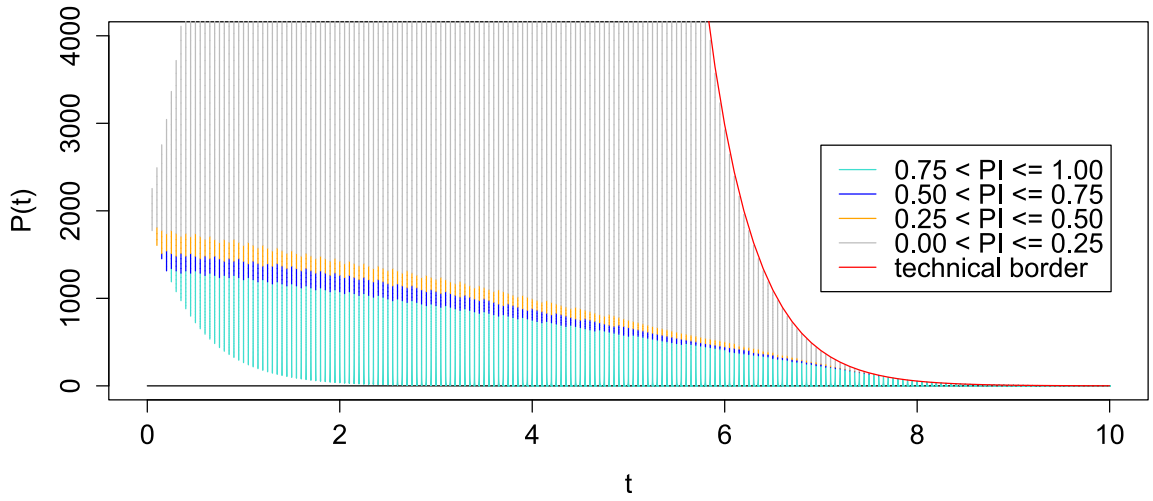

Figure 3.13.: Dynamic programming probability results with constant absolute investment (A1-2)

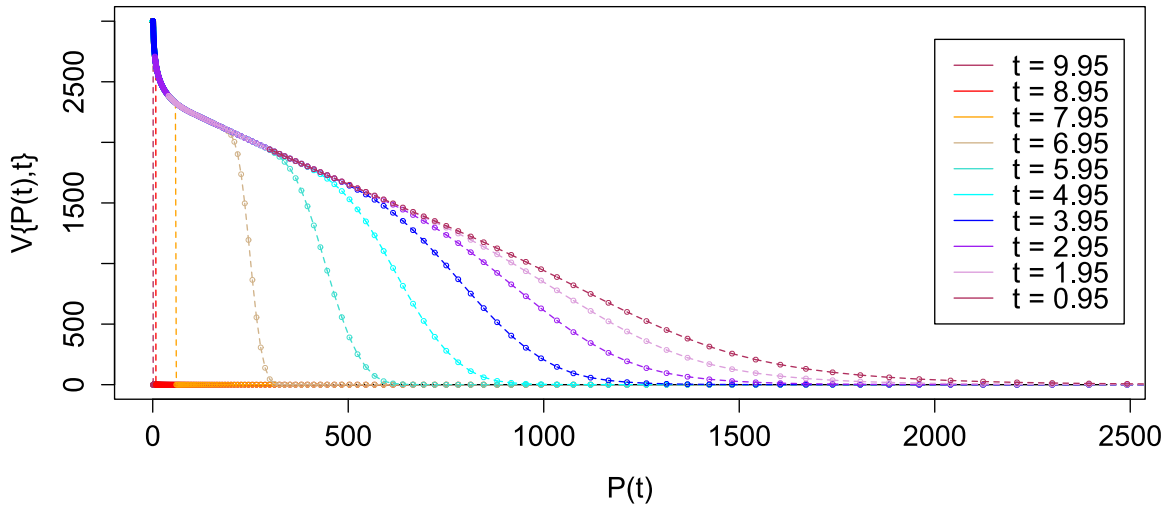

Figure 3.14.: Slices through the value ridge generated with constant absolute investment (A1-2)

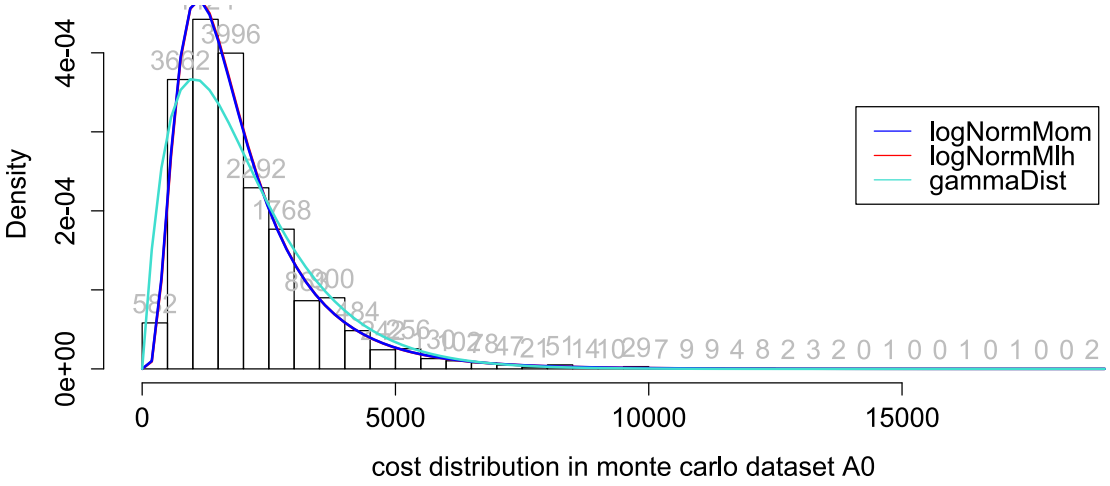

Figure 3.15.: Histogram of simulated remaining cost at t = T for A1-0 (n = 20000)

	formula	A1-0	A1-1	A1-2
Net-present-value	V(0) =	2.439 e-29	297.202	83.231
Success probability	Π(0) =	1.316 e-32	0.655	0.143

Table 3.3.: Dynamic programming comparison by strategy

probability and net-present-value of the non-investment strategy (A1-0) are negligible. These initial statements are substantiated by table 3.4. We find that for a sample size of $n = 20000$ constant relative investment is more successful in actually launching the project within a simulation run than a strategy of fixed payments (A1-3) (67.55 % vs. 34.49%) since profitability can be preserved through higher levels of expected cost to completion.
Turning towards cash-flows, i.e., the sum of the money invested plus possible rewards, seen in table 3.5, it is clear that waiting over the whole project horizon won't result in any exchange of wealth between the project and it's investor. We define a samples profit as $-\sum_{t=0}^{T} I(t) + R(P(T))$. Note that while strategy A1-1 has a higher expected value,

	formula	A1-0	A1-1	A1-2
# simulations total	N =	20000	20000	20000
# projects simulated	n =	0	13510	6899
% projects started	$\frac{n}{N}$ =	0.0000	0.6755	0.3449

Table 3.4.: Monte Carlo comparison by strategy

	A1-0	A1-1	A1-2
mean	-	374.75	124.78
std dev	-	382.79	522.68
median	-	416.92	- 10.00
max	-	1250.74	1372.92
min	-	-2754.99	-1412.00
range	-	4005.74	2784.92

Table 3.5.: Project turnout comparison by strategy

	A1-0	A1-1	A1-2
mean	2007.07	896.43	1838.38
std dev	1409.41	1576.83	1556.50
median	1637.46	0.98	1573.26
max	18786.66	16008.94	18786.66
min	95.67	0.98	0.98
range	18690.99	16007.96	18785.68

Table 3.6.: End remaining cost comparison by strategy

it also bears the risk of larger losses and is humbled by inferior maximum profit. On the other side A1-2 usability suffers heavily from the negative median value and a large spread of outcomes indicated by the standard deviation. Mentioning remaining cost to completion at the terminal date, table 3.6, constant relative investment is more effective in generating the necessary drift towards the critical reward cost than constant absolute investment. Special attention is deserved by the median which is exactly the price node below the reward margin while A1-2 median is far higher. As we would expect the mean of remaining cost to completion in the strategy of non-investment (A1-0) stays very close to the initial value P_0 of the continuous log-normal distribution. The deviation is caused by numerical effects due to discretization and simulation. From our preceding investigations we can deduce that investment at relative rates is superior to constant rates. Also, a long project horizon may turn out as a benefit since the investor might, due to random shocks, be able to avoid the end-game area where the entry to the project is most expensive (in A1-1) or least effective (in A1-2). The location of this end-game phase on the time axis, however, is dependant on the operator's strategic decisions i.e. investment ratio itself.

4. Example A2 - dynamic programming of risk-averse project investment

The example presented in the preceding chapter 3 assumes a risk neutral decision maker, i.e., a decision maker who seeks to maximize expected profit but is not concerned about higher moments of the profit distribution (the riskiness of profits). In this chapter, we model risk averse decision makers who seek to maximize expected utility. For advanced studies of risk-aversion Ingersoll's *'Theory of Financial Decision Making'*[9] or Ljungqvist's *Recursive Macroeconomic Theory*[10] may provide a good starting point. Before, we boldly supposed that an individual is willing to spend any amount of monetary units in return for a chance of profit whose net present value lies above the investment (even with long odds). Unfortunately this presupposition is incorrect for the majority of economic subjects as most of these are considered risk-averse (in contrary to our previous model which can be deemed risk-neutral). From now on we will try to incorporate risk avoiding behaviour into our model starting out with a definition of risk aversion as *the reluctance of a person to accept a bargain with an uncertain payoff rather than another bargain with a more certain, but possibly lower, expected payoff* [1]. In mathematical terms an individual is considered risk-averse if the expected utility of a consumption is smaller than the utility of an expected consumption as seen in equation 4.1. This relation is commonly referred to as Jensen's inequality (for expected values).

$$E(u(c)) < u(E(c)) \tag{4.1}$$

Let us assume that the overall utility of a series of consumptions is time separable and can be modelled as discounted sum of the consumption's utilities over the time-horizon T as seen in equation 4.2. A thorough discussion on this topic can be found in vol. 40 of the 'Journal of Economic Literature'[6].

$$U = U(c_{t=0}, c_1, \ldots) = \sum_{t=0}^{T} \frac{u(c_t)}{1+\kappa}^t \tag{4.2}$$

To carry out our intention we will have to choose a suitable utility function that satisfies a small set of criteria commonly adopted in economics and especially utility theory. These so called *Inada conditions* are referred to in [13].

[1] http://en.wikipedia.org/wiki/Risk_aversion#Relative_risk_aversion

Inada conditions In the neoclassical growth model the Inada conditions ensure the stability of an economic growth[2] path by presupposing a set of six conditions. A thorough discussion on these can be found in [8] and [7].
The conditions are

- the function is continuously differentiable
- the function is strictly increasing $\delta f(x)/\delta x > 0$
- the second derivative of the function is decreasing (i.e. the function is concave); $\delta^2 f(x)/\delta x^2 < 0$
- for x equal to 0 the limit of the first derivative is infinity; $\lim_{x\to 0} \delta f(x)/\delta x = +\infty$
- for x striving to infinity the limit of the first derivative is zero; $\lim_{x\to +\infty} \delta f(x)/\delta x = 0$

4.1. Introducing the CRRA utility function

The constant relative risk aversion (CRRA) utility function satisfies all of these conditions. It is defined as shown in equation 4.3 and can be found in [9] on page 40. Yet, it defines utility in terms of consumption and will therefore require us to revise or model towards an consumption oriented approach. We will return to this point later on.
Figures 4.1 and 4.2 show the trend of the function value. For γ values beneath 1 the result is has no upper bound i.e. the function strives for $+\infty$ while it has no lower bound for γ larger than 1 and converges against an upper border. For γ equal to 1 the CRRA utility function behaves as natural logarithm and is therefore not bounded at both ends. Regardless of the risk-aversion coefficient a fixed point exists at $u(c = 1) = 0$.

$$u(c) = \begin{cases} \frac{c^{1-\gamma}-1}{1-\gamma} & ,\gamma > 0, \gamma \neq 1 \\ \ln(c) & ,\gamma = 1 \end{cases} \tag{4.3}$$

A gradient of the function is easily derived (eq. 4.4) and illustrated in diagrams 4.3 and 4.4. The gradients development is experienced steep between 0 and 1 and plane afterwards with a fixed point of $u'(c = 1) = 1$.

$$u'(c) = \begin{cases} c^{-\gamma} & ,\gamma > 0, \gamma \neq 1 \\ \frac{1}{c} & ,\gamma = 1 \end{cases} \tag{4.4}$$

$$u"(c) = \begin{cases} -\gamma c^{-(\gamma+1)} & ,\gamma > 0, \gamma \neq 1 \\ -\frac{1}{c^2} & ,\gamma = 1 \end{cases} \tag{4.5}$$

Let us define the Arrow-Pratt-De Finetti measure of relative risk-aversion as seen in equation 4.6. With the utility functions gradient (equation 4.4) and second derivative

[2] http://en.wikipedia.org/wiki/Inada_conditions

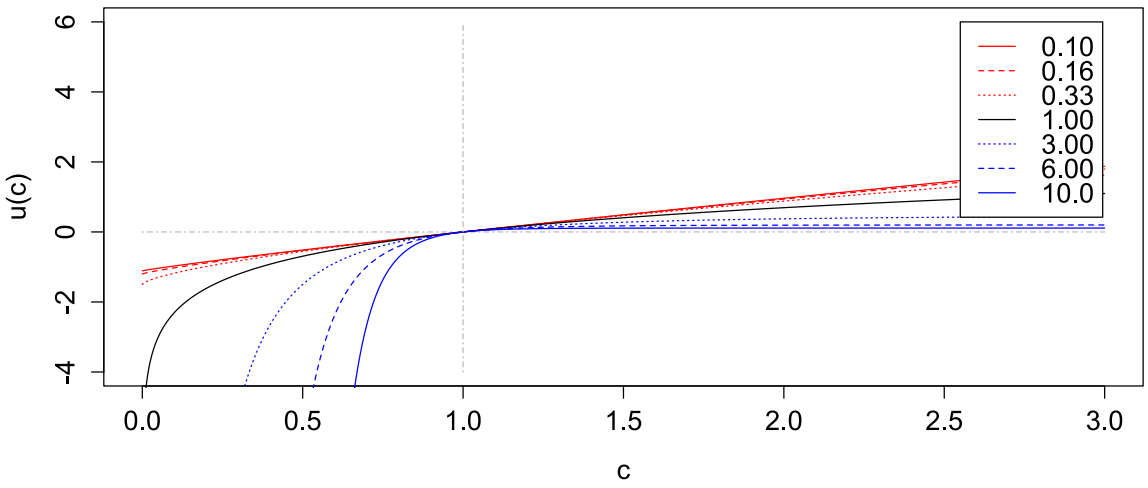

Figure 4.1.: CRRA utility function value (interval 0 to +3)

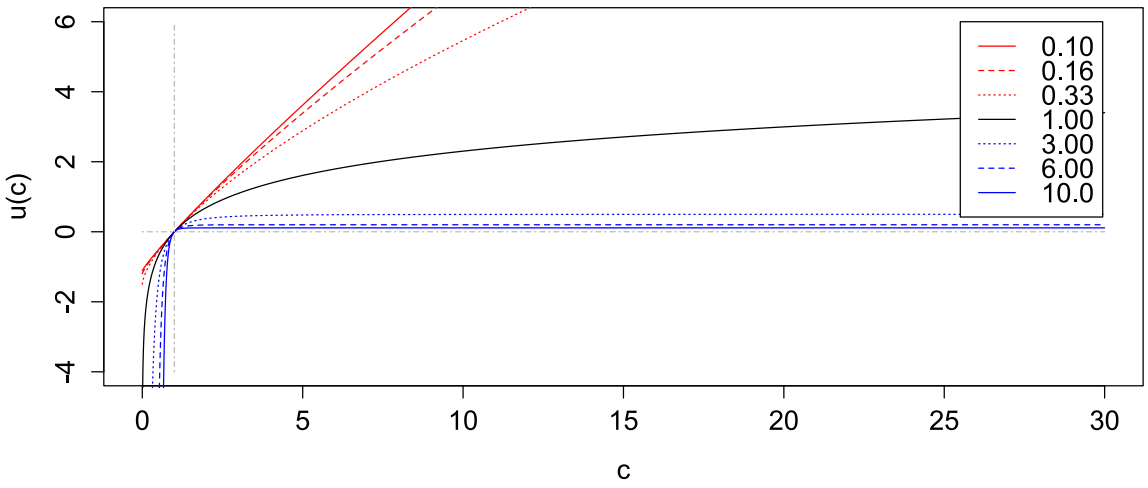

Figure 4.2.: CRRA utility function value (interval 0 to +30)

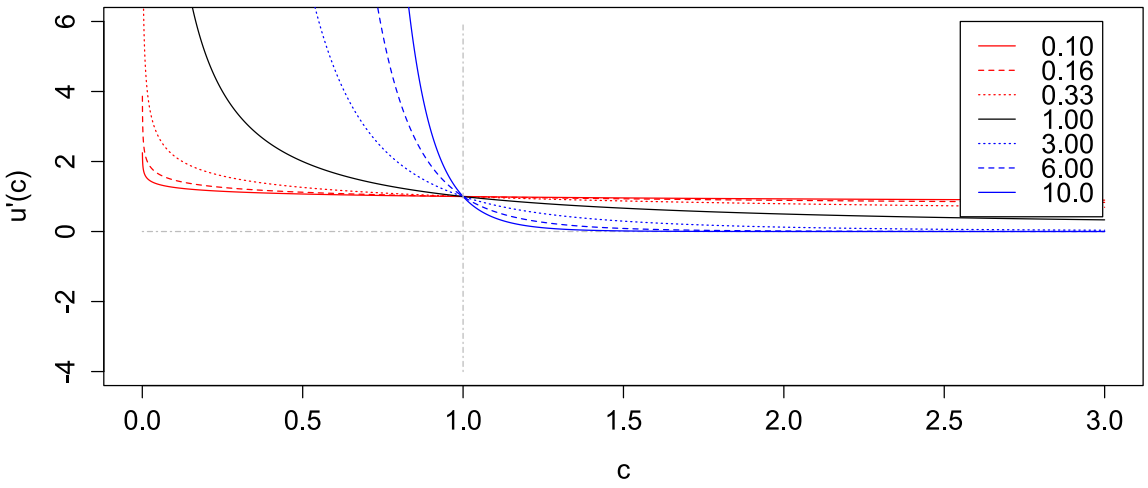

Figure 4.3.: CRRA utility function gradient (interval 0 to +3)

(eq. 4.5) it is easy to prove that the utility function's relative risk aversion is indeed constant and of value γ. Refer to equation 4.7 for the calculation.

$$R(c) = \frac{-cu"(c)}{u'(c)} \tag{4.6}$$

$$R_{CRRA}(c) = \frac{(-c)(-\gamma)c^{-(\gamma+1)}}{c^{-\gamma}} = \gamma \tag{4.7}$$

Typical γ coefficients for the CRRA utility function commonly adopted in literature are somewhat between 3 and 10. For our evaluation we chose the value 1,3 and 6.

4.2. Modelling with risk-aversion

Since the CRRA utility function originates from the consumption theory it is designed with positive values in mind. Therefore we could not simply feed our investment expenses $-I_t$ to the utility function and receive reasonable results. The solution of choice was to set any arbitrary amount of monetary units that an investor is willing to spend per time-step as available capital and subtract the investment from there. In other words, if the investor decides to wait, all reserved capital is ready for consumption, if on the other hand an investment is to be made, its returns must allow for higher consumption later, which must compensate for the loss in utility today Since the immediate losses are weighed higher than the future earning we can achieve any grade of risk-aversion desired by simply altering the value of the utility functions γ coefficient. Equation 4.8

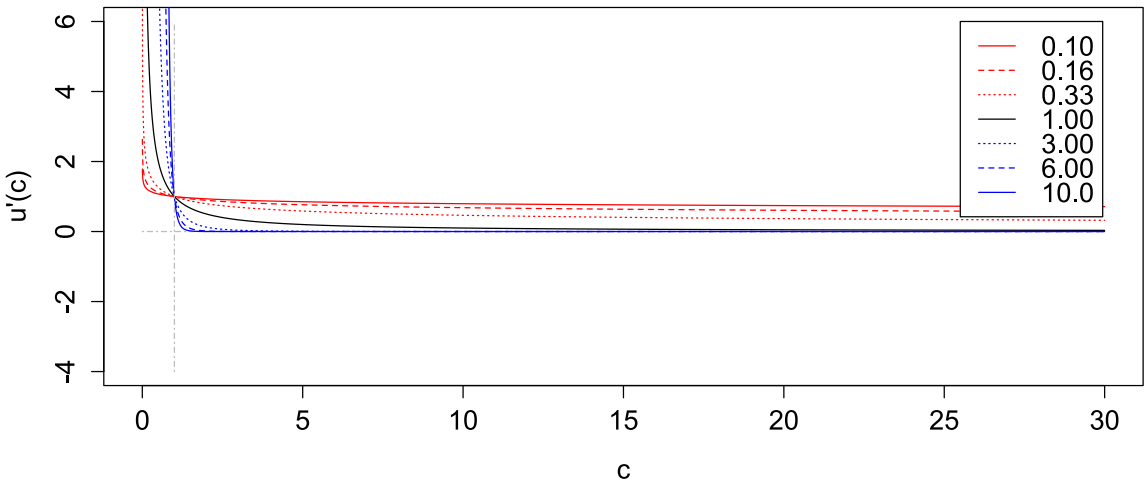

Figure 4.4.: CRRA utility function gradient (interval 0 to +30)

shows the basic function used to model the evaluation of the money invested. While the normalization $\frac{I_{\max_f} - I}{I_{\max_f}}$ may seem unusual for consumption theory it fits our investment purpose perfectly. This representation assumes that an investor does not gain any utility from clinging to the funds he already possesses i.e. the maximum for the immediate consumption term is always 0 (except for nodes where the a reward is distributed). If the decision maker decides to invest his immediate utility reward is negative (between $-\infty$ and 0) and he expects the asset to compensate for this in the future. An illustration can be found in diagram 4.5. The term $I_{\max_{f(inancial)}}$ represents the afore mentioned available amount of monetary units from which the invested capital is subtracted. Note that the remaining amount is again scaled by $I_{\max_f}$ so that the variable fed to the CRRA function can only take values between 0 and 1.

$$u\left(\frac{I_{\max_f} - I}{I_{\max_f}}\right) = \frac{\left(\frac{I_{\max_f} - I}{I_{\max_f}}\right)^{1-\gamma} - 1}{1-\gamma} = \frac{(I_{\max_f} - I)^{1-\gamma}}{(1-\gamma)(I_{\max_f})^{1-\gamma}} - \frac{1}{1-\gamma} \tag{4.8}$$

As said the parameter $I_{\max_f}$ is the financially tolerable amount an investor is willing to spend. For our example we decided to determine this value by discounting the project reward over the full time horizon i.e. from the terminal date t = T to the initial point in time t = 0. Thus $I_{\max_f}$ is given by eq. 4.9. We will see later that $I_{\max_f}$ does not only serve in the utility function but also as boundary condition.

$$I_{\max_f} = R \times \frac{1}{(1+r)^T} \tag{4.9}$$

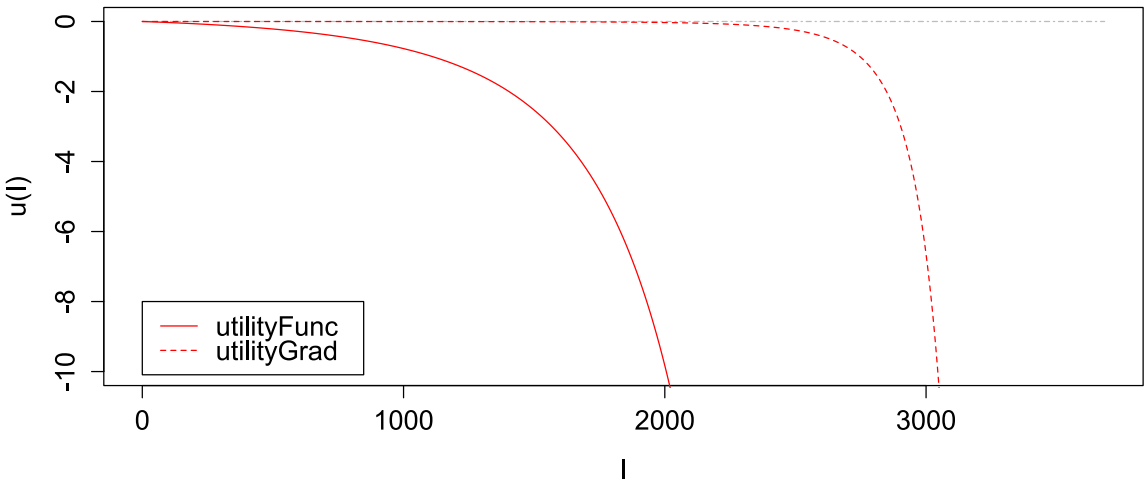

Figure 4.5.: Utility contribution of investment activity scaled with $I_{\max_f}$ (interval 0 to $I_{\max_f}$)

Aside from $I_{\max_f}$ another kind of maximum tolerable investment has to be defined. The variable $I_{\max_{t(echnical)}}$ is the highest amount of monetary units a lattice node can absorb while keeping the probabilities real i.e. $\pi = 0$ and $(1 - \pi) = 1$. The calculation of $I_{\max_t}$ which must be repeated for every node is seen in 4.10.

$$I_{\max_t} = \frac{P(t)}{\Delta t} \times (1 - d) \tag{4.10}$$

The application of $I_{\max_f}$ enables a reasonable consumption for the nodes in every time-step where no investment is taking place i.e. the utility results from the unspent monetary units. In our example this consumption is $\frac{I_{\max_f}}{I_{\max_f}} = 1$ thus making $u(1) = 0$ for every possible level of risk aversion. This measure is especially important for the termination time-step where a zero value would lead to a utility of $-\infty$ which mathematically dominates the whole random walk. Equation 4.11 deducts the gradient of the utility function and figure 4.5 diagrams the function's and its gradient's development over the invested monetary units under the parameters, $R = 6000, \gamma = 1$

$$u'\left(\frac{I_{\max_f} - I}{I_{\max_f}}\right) = \left(\frac{I_{\max_f} - I}{I_{\max_f}}\right)^{-\gamma} \cdot \left(-\frac{1}{I_{\max_f}}\right) \tag{4.11}$$

4.3. Adjusting the Bellman equations

The Bellman equation from chapter 3 can be reused almost without changes. Only two minor adjustments need to be made. First we exchange $-I$ for our new found CRRA utility function. Second we switch r for κ which is the inter-temporal discounting factor of utility and can be chosen arbitrarily. In our examples we will let it equal r. Equation 4.12 depicts the adapted Bellman equation. Note that $\pi(\frac{I}{P})$ is still determined by equation 3.11.

$$V\{P,t-1\} = \max_{I\in\mathbb{R}^+}[u\left(\frac{I_{\max_f}-I}{I_{\max_f}}\right)\Delta t + \frac{1}{(1+\kappa)^{\Delta t}}(\pi(\frac{I}{P})V\{Pu,t\} + (1-\pi(\frac{I}{P}))V\{Pd,t\})] \tag{4.12}$$

As before not every node in the lattice requires an optimization process. For those points in the grid whose time t = T let the utility equal to $u(1)\Delta t = 0$. Nodes that lie beyond the reward margin P_R on the other hand shall receive the utility of the consumption of the money available to invest plus the utility of the reward regardless of their time value t. All other points are subject to the dynamic programming process as can be seen in eq. 4.12. The case structure in 4.13 illustrates the complete algorithm.

$$V\{P,t-1\} = \begin{cases} u\left(\frac{I_{\max_f}}{I_{\max_f}}\right)\times\Delta t = 0 & ,t = T \wedge P(t) > P_R \\ u\left(\frac{I_{\max_f}}{I_{\max_f}}\right)\times\Delta t + u\left(\frac{R}{P_0}\right) & ,t \leq T \wedge P(t) \leq P_R \\ \max\limits_{I\in\mathbb{R}^+}[u\left(\frac{I_{\max_f}-I}{I_{\max_f}}\right)\times\Delta t + \frac{1}{(1+\kappa)^{\Delta t}}\times\dots & \\ (\pi(\frac{I}{P})V\{P\times u,t\} + (1-\pi(\frac{I}{P}))V\{P\times d,t\})] & ,t < T \wedge P(t) > P_R \end{cases} \tag{4.13}$$

Turning towards the optimization of an individual node the introduction of the CRRA function facilitates the existence of internal solutions. Yet these solution does not necessarily occur within the admissible range of values which we specify as in 4.14. Therefore the investment value I has to be chosen within the space of $\epsilon[0, I_{\max}]$. To find the optimal value for I the gradient of the Bellman equation 4.12 is derived (eq. 4.15) and analysed in specific points. This is possible as the second derivative of the Bellman equation (4.16) retains only the term from the investment function which is ≤ 0. The gradient can hence be considered strictly monotonic decreasing and allows us to find a maximum by application of the first order condition.

$$0 \leq I \leq I_{\max} = \min[I_{\max_f}, I_{\max_t}] \tag{4.14}$$

$$\nabla V = \frac{\delta V}{\delta I} = \left(\frac{-1}{I_{\max_f}}\right)\left(\frac{I_{\max_f}-I}{I_{\max_f}}\right)^{-\gamma}\Delta t + \frac{1}{(1+\kappa)^{\Delta t}}\times\frac{1}{u-d}\times\frac{\Delta t}{P}\times[V\{Pd,t\} - V\{Pu,t\}] \tag{4.15}$$

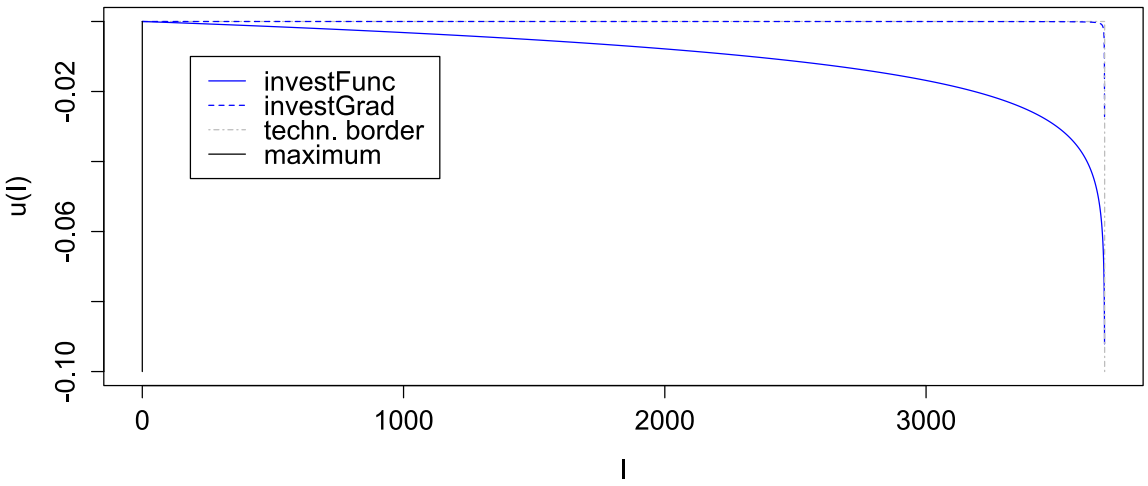

Figure 4.6.: Optimization of a single node - negative gradient at zero - wait

$$\frac{\delta^2 V}{\delta I^2} = \frac{-\gamma}{(I_{\max_f})^2} \left(\frac{I_{\max_f} - I}{I_{\max_f}} \right)^{-\gamma-1} \Delta t \leq 0 \tag{4.16}$$

To find the optimal solution for each node, the gradient of the Bellman equation has to be computed in two known points. Depending on the computation results we can determine three cases of possible decisions. In two of these the optimum strategy is immediately known i.e. waiting or investment at full rate. In the third case an internal maximum exists and its location needs to be calculated.

Case 1: $\nabla V \leq 0$ for $I = 0$

Should the gradient 4.15 of the investment function be less than 0 for $I = 0$ the allowable maximum of $V\{P, t-1\}$ is to be found exactly there, i.e., the allocation of money to the project is not recommended. This is possible as the Bellman equation's second-order derivative is strictly ≤ 0. Figure 4.6 illustrates the case with both functions (utility and its gradient) strictly declining for a waiting node in the random-walk's grid. The CRRA utility is represented as continuous, blue line whil its gradient is drawn in dashed blue. The grey vertical line marks the technical tolerable amount of investment, i.e., $I_{\max_t}$. The black line indicates that the maximum utility is to be found at I(t) = 0.

Case 2: $\nabla V > 0$ for $I = 0$ & $\nabla V \geq 0$ for $I = I_{\max}$

If however the gradient at the origin is positive and does not drop below zero for $I_{\max}$ then investment at maximum rates is preferred. This case can only occur if $I_{\max_T} < I_{\max_f}$

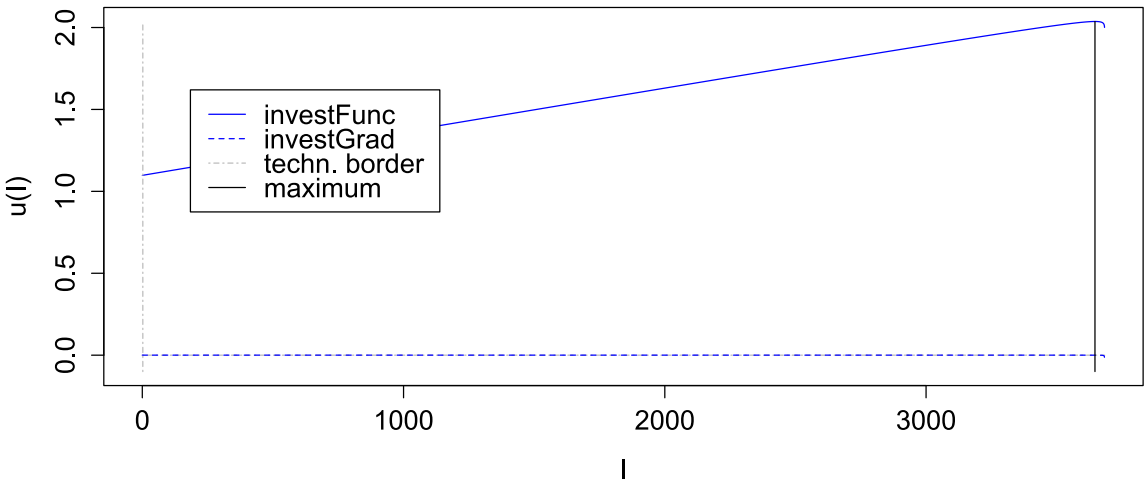

Figure 4.7.: Optimization of a single node - positive gradient at barrier - full invest

since the gradient and function value of $I_{\max_f}$ equal $-\infty$. The invested value is then set to $I = I_{\max_T}$. For better perception diagram 4.7 is included. Note that the technical barrier is far to the left while the internal maximum would occur close to $I_{\max_f}$.

Case 3: $\nabla V > 0$ **for** $I = 0$ **&** $\nabla V < 0$ **for** $I = I_{\max}$

Finally if the two extrema of I have different leading signs the intermediate value theorem applies and an internal solution must exist. The optimal value can be calculated by means of equation 4.17. This analytic three step solution is preferable over numeric attempts to find the optimum since it is not only faster but also way more reliable. While the Brent algorithm performed quite well in locating the optima, gradient based methods as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm had substantial difficulties with the extreme flatness of the gradient function.

$$\frac{\delta V}{\delta I} = 0 \rightarrow I = I_{\max_f} - \left\{ I_{\max_f}^{1-\gamma} \frac{1}{(1+\kappa)^{\Delta t}} \times \frac{1}{u-d} \times \frac{1}{P} \times [V\{P \times d, t\} - V\{P \times u, t\}] \right\}^{-\frac{1}{\gamma}} \tag{4.17}$$

4.4. Dynamic programming of investment

For the dynamic programming process most of the variables stay as they were in chapter 3. The only alterations are the introduction of an inter-temporal utility discount factor κ which we let equal to 0.05 and the alteration of the reward to $R = 6000$. This

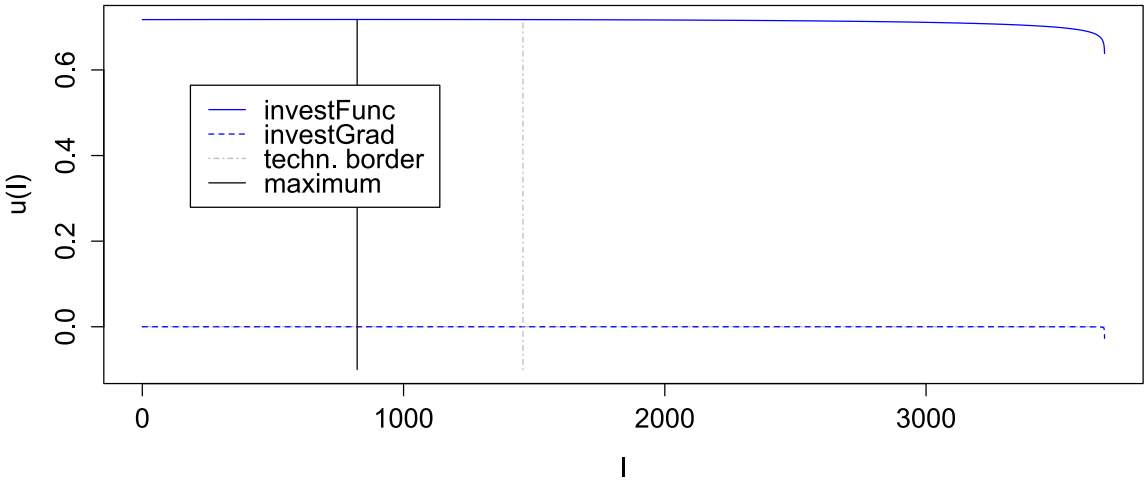

Figure 4.8.: Optimization of a single node - inner solution - partial invest

second change is due to better visibility of the optimization instruction graphs as with a reward of 3000 the coloured areas would have lingered in the low lying areas of the diagram. Observe that therefore the Monte Carlo methods results are NOT quantitatively comparable to the findings from before. Also note that the slice diagrams of the NPV ridges exhibit a different scaling on the y-axis and that the utility values in the upcoming examples are not inter-comparable as the risk-aversion coefficient is altered between the experiments. Table 4.1 shows the extended project model parameters.

formula	source code	value	unit
P_0	priceGuess0	2000	[monetary units]
P_R	rewardMargin	1	[monetary units]
λ	lambda	$\frac{P_R}{P_0}$	[-]
φ	shift	0	[-]
β	beta	0.20	[-]
κ	kappa	0.05	[-]
r	r	0.05	[-]
T	Tmax	10	[time units]
R	reward	6000	[monetary units]

Table 4.1.: Project inherent variables

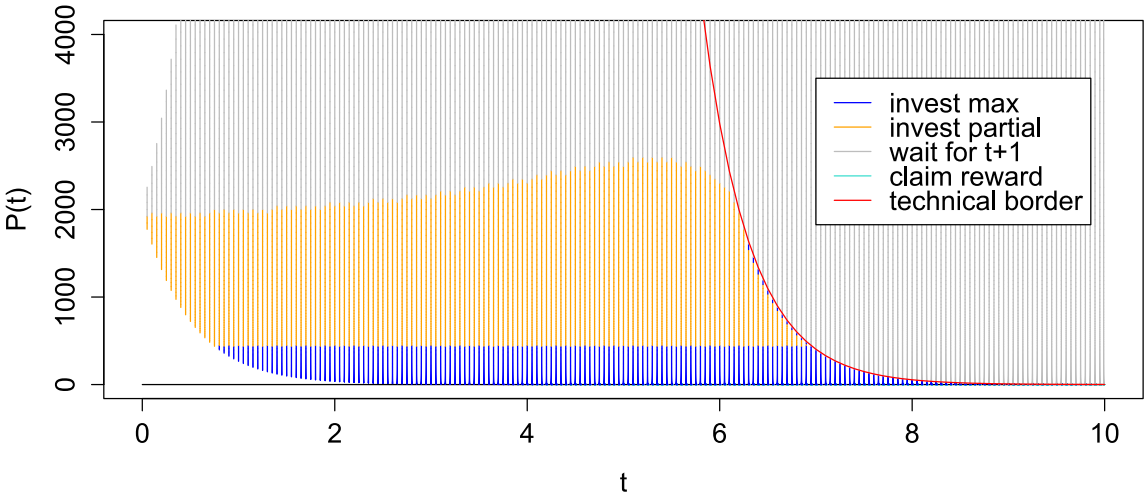

Figure 4.9.: Dynamic programming instruction results with $\gamma = 1$

4.4.1. Experiment A2-1, risk-aversion($\gamma = 1$)

Dynamic programming of investment under the assumption of risk-aversion results in a net-present-value of utility V(0) = 0.3057 with a project success probability of $\Pi(0)$ = 0.9142. A risk-averse investor with $\gamma = 1$ is called myopic, since the income effect and the substitution effect of different return expectations completely cancel out. The instruction graphs in this chapter distinguish between 'invest max', which is synonymous to investment of the full technically allowable amount $I_{\max_t}$, and 'invest partial', that highlights an internal solution between zero and $I_{\max}$. The 'invest max' area is sharply confined somewhere around $P(t) \sim 500$ and shows a significant end-game effect between t = 5 and 6. Figure 4.9 diagrams the instruction graph for risk-aversion while 4.10 adds a review of success chances and 4.11 contributes the slices of the utility ridge. We state that risk-aversion under current parameters has a quite low threshold for investment and more or less immediately starts to dispense monetary units to the project. In the comparison section we will find that this results in the highest number of launched projects but also that myopic investors suffer the largest losses.

4.4.2. Experiment A2-2, risk-aversion ($\gamma = 3$)

With a risk-aversion coefficient of $\gamma = 3$ we experience a significant drop in both the net-present-value and the finishing probability. While the present value of utility V(0) = 0.0128 is not comparable to the experiment conducted before the chance of success $\Pi(0)$ = 0.1806 shrinks by roughly 80% stacked up against experiment A2-1. A γ-factor of 3 implies in a utility cap of $u(c) = 0.5$ for a consumption of $c = +\infty$. Therefore the utility

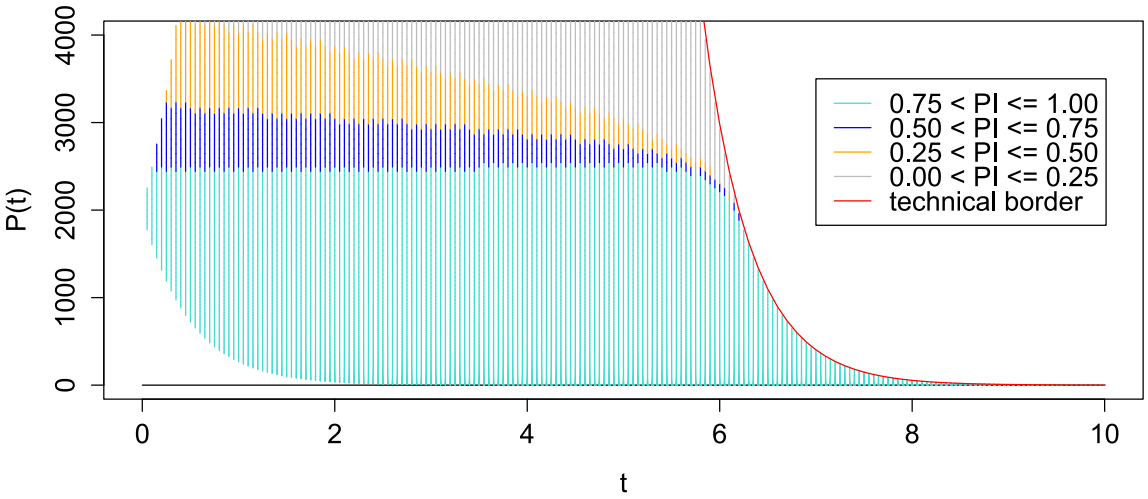

Figure 4.10.: Dynamic programming probability results with $\gamma = 1$

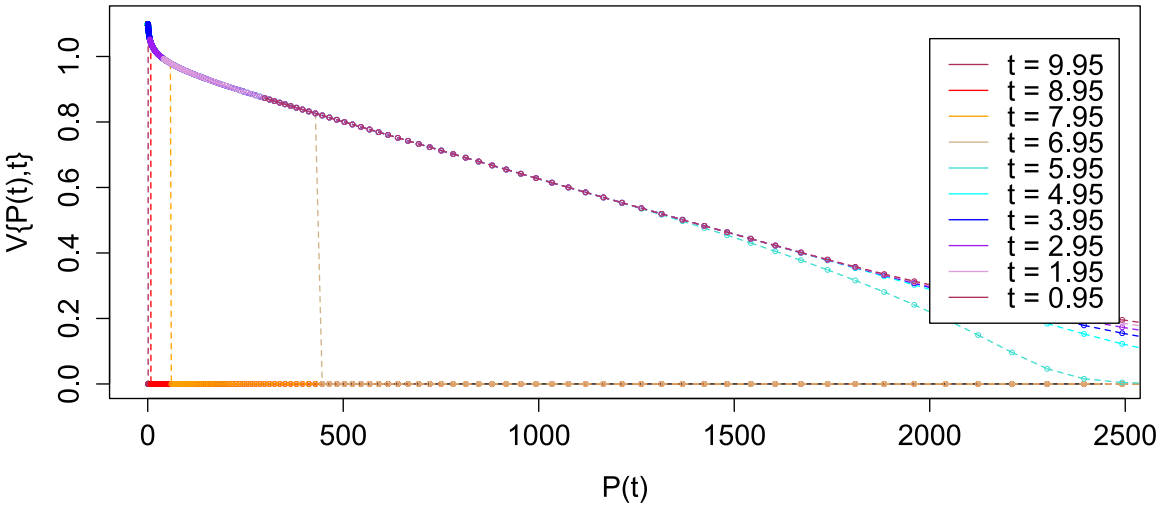

Figure 4.11.: Slices through the utility ridge generated with $\gamma = 1$

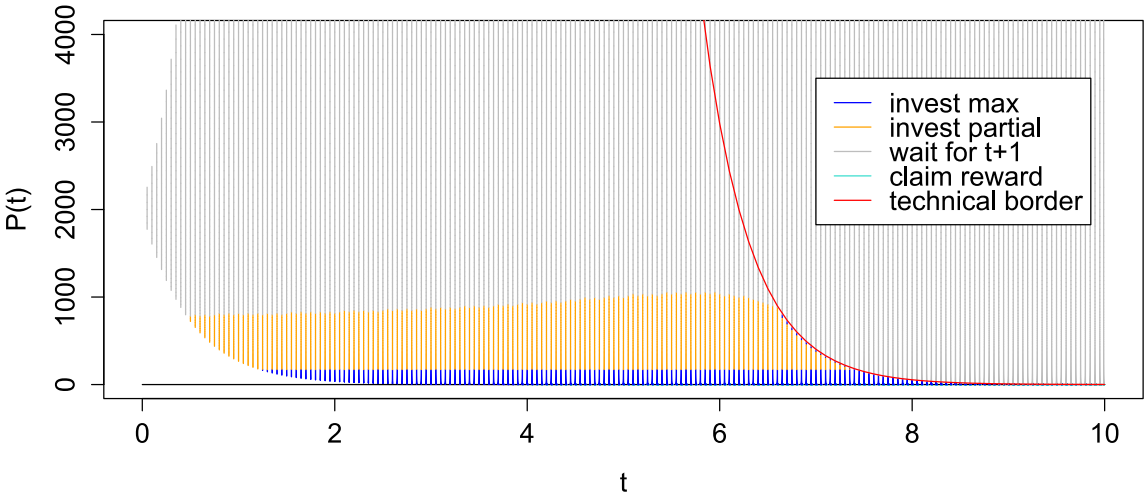

Figure 4.12.: Dynamic programming instruction results with $\gamma = 3$

of consumption of the reward is considerably smaller than in dynamic programming with risk-neutrality (0.4444444 versus 1.098612). Also the actually required amounts of money to finance the project are valued higher as they have a larger (negative) effect on immediate consumption. Both these effects cause the coloured areas in the instruction graph 4.12 and the probabilities diagram 4.13 to appear somewhat compressed. The domain of 'invest max' also shrinks to a border around $P(t) \sim 250$. In the utility figure 4.14 the difference is seen by a slight shift to the left of the whole figure. Also remember the different scale of the y-axis that originates from the boundedness of the utility function.

4.4.3. Experiment A2-3, risk-aversion ($\gamma = 6$)

$\Pi(0) = 0.00789338843501965$
The last experiment is carried out with a γ-factor of 6, a value approximately in the middle of the commonly assumed risk-aversion area of 3 to 10. The dynamic programming process results in $\Pi(0) = 0.0079$ with a net utility value of $V(0) = 0.0002$. We will find in the Monte Carlo section that only a almost negligible number of cases the project is actually launched under these circumstances. Thus the hazard of financial losses is hardly present. The utility of the un-discounted reward is 0.199177 (compared to 1.098612 with myopic risk-averse investment). With the even steeper (negative) trajectory for allocated monetary units this results in even more compressed 'invest max' and 'invest partial' areas. Figures 4.15 and 4.16 illustrate the dynamic programming results. In diagram 4.17 the utility ridge is shown; again shifted to the left i.e. a number of lattice nodes downwards.

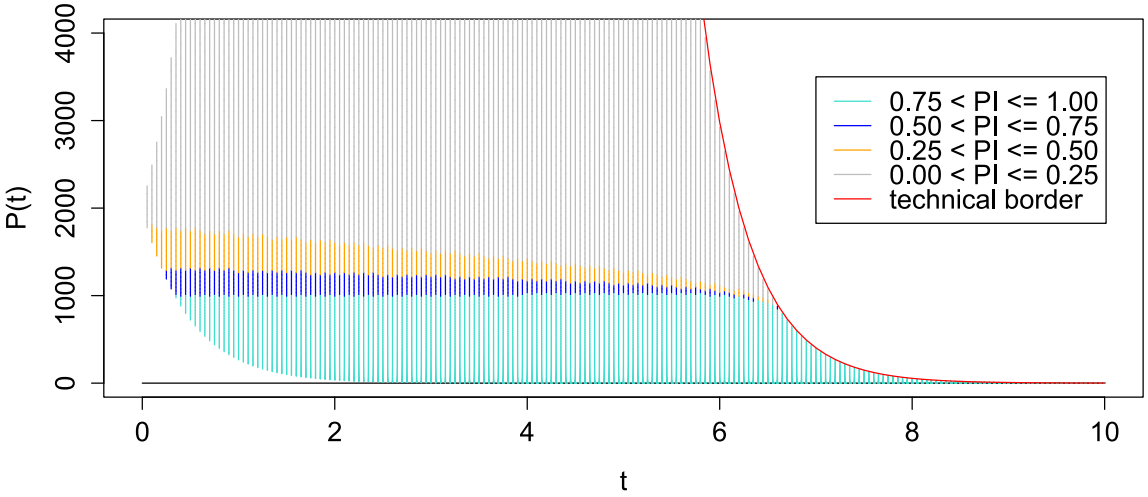

Figure 4.13.: Dynamic programming probability results with $\gamma = 3$

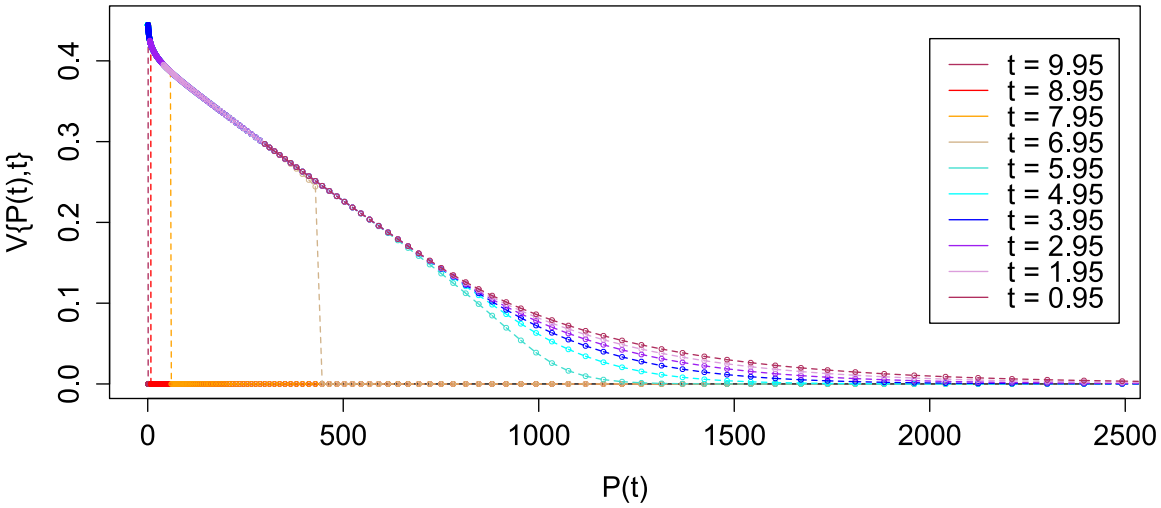

Figure 4.14.: Slices through the utility ridge generated with $\gamma = 3$

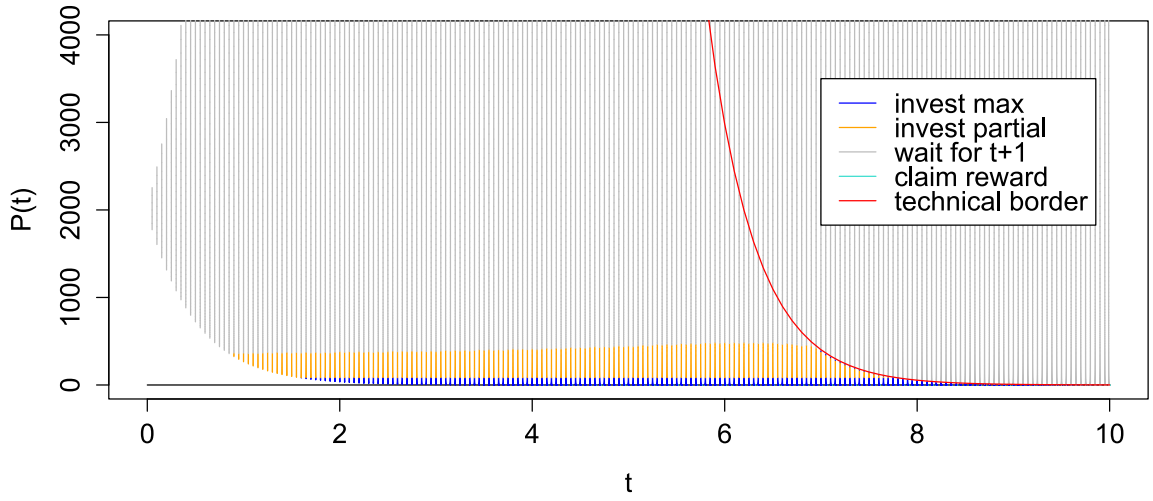

Figure 4.15.: Dynamic programming instruction results with $\gamma = 6$

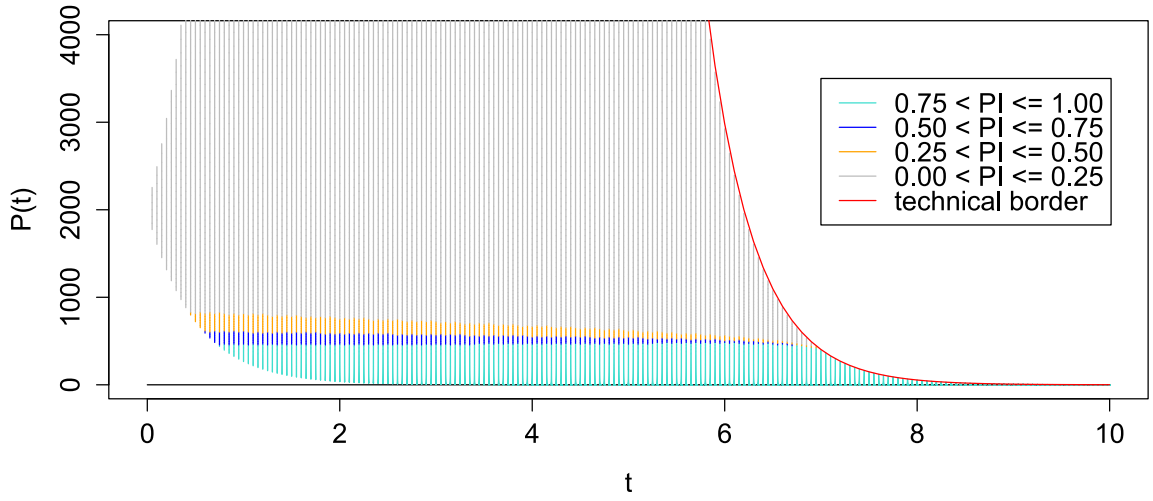

Figure 4.16.: Dynamic programming probability results with $\gamma = 6$

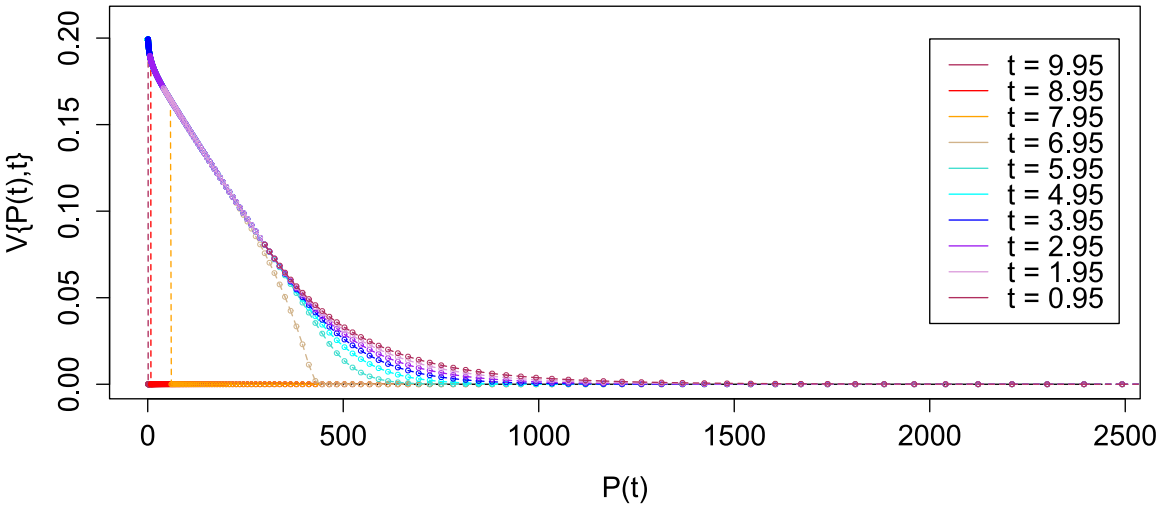

Figure 4.17.: Slices through the utility ridge generated with $\gamma = 6$

4.5. Monte Carlo simulation and comparison

Here we reiterate our Monte Carlo method with the matrices obtained from dynamic programming in experiments A2-1 to A2-3. Subsequently the projects cash-flow turnouts and remaining cost to completion are evaluated. Table 4.2 repeats the values already showcased while table 4.3 poses the numbers gained from the simulation results. The percentage of cases where project financing is started corresponds roughly to the probability of successful completion calculated earlier. Thus in most of the simulations run where the project is initiated it is also concluded satisfactory below the reward margin.

	formula	A2-1	A2-2	A2-3
Net-present-value	V(0) =	0.306	0.013	0.0002
Success probability	Π(0) =	0.914	0.181	0.0079

Table 4.2.: Dynamic programming comparison by strategy

Reviewing the cash-flows in table 4.4 from the different experiments only minor differences can be found in the maximum outputs. Yet in the minima there is a factor $\frac{-1179.59}{-39.72} = 29.69763$ between the highest and the lowest computed loss. Hence the risk-aversion introduced through the CRRA utility function is remarkably eligible to reduce possible losses of an investor. Of course there is also a down-side to risk-averse behaviour. The number of simulation runs where the project is started by an investor with $\gamma = 6$ is very small with a percentage range somewhere around 1%. This can be seen

	formula	A2-1	A2-2	A2-3
# simulations total	N =	20000	20000	20000
# projects simulated	n =	19487	4500	202
% projects started	$\frac{n}{N}$ =	0.974	0.225	0.010

Table 4.3.: Monte Carlo comparison by strategy

in table 4.3 as well as in the remaining costs to completion, seen in table 4.5, which are hardly affected by the investment strategy. Remember that the expected value without any intervention is 2000 monetary units for the cost at t = T.
We end this section here and move on to optimization of portfolio investment under uncertainty.

	A2-1	A2-2	A2-3
mean	2688.26	2511.20	2852.23
std dev	915.68	1248.69	1340.34
median	2957.57	2964.65	3425.80
max	4026.35	4200.19	3953.04
min	-1179.59	- 336.05	- 39.72
range	5205.95	4536.24	3992.76

Table 4.4.: project turnout comparison by strategy

	A2-1	A2-2	A2-3
mean	327.17	1847.89	1990.96
std dev	1218.97	1562.49	1407.32
median	0.98	1637.46	1637.46
max	17342.28	28026.41	18786.66
min	0.98	0.98	0.98
range	17341.30	28025.43	18785.68

Table 4.5.: End remaining cost comparison by strategy

5. Example B - dynamic programming of portfolio investment under uncertainty

In the concluding chapter of this work we turn towards optimization of investment in a portfolio of possibly correlated projects. This correlation of random shocks affecting the remaining R&D expenditures is denoted by the variable ρ and can take on values from -1 e.g. cases where a positive development for one project immediately implies a negative shock for the other to $+1$ where positive results complement each other. However as long as $\rho < 1$, the projects in the portfolio compete for resources e.g. financing. A paper from Boyle, Evnine and Gibbs titled 'Numerical Evaluation of Multivariate Contingent Claims' [2] acts as starting point for our research. Slightly altering their probability model for multivariate distributions on discrete lattices we can expand our dynamic programming effort to a two dimensional portfolio of projects.

5.1. Modelling of portfolio investment

Our first task is to adapt the probability model's drift parameters to the investment model. Therefore we develop the relationship between a general Geometric Brownian motion (eq. 5.1) and our special investment case with induced drift. Let dx in equation 5.1 be the alteration of x over an infinitesimal time interval.

$$dx = axdt + \sigma x dz \tag{5.1}$$

Then the change of the estimated project cost dP for continuous time is given by eq. 5.2.

$$dP = -Idt + \beta P dz \tag{5.2}$$

Expanding the equation's drift term by $\frac{P(t)}{P(t)}$ the parameters of our investment model can be identified as drift of $-\frac{I}{P(t)}$ and β as variance of the geometric Brownian motion. This is coherent with our first example's assumptions for investment strategies. Moving from continuous time to a discrete setting we can establish an iterative pattern. Let $P(t+1)$ be the sum of $P(t)$ and the alteration of estimated cost over a discrete time interval ΔP $(= \beta P - I)$. The iterative scheme is given in equation 5.3.

$$P(t+1) = P(t) \times (1+\beta) + (-I) = P(t) \times \left(1 + \left(-\frac{I}{P(t)}\right) + \beta\right) \tag{5.3}$$

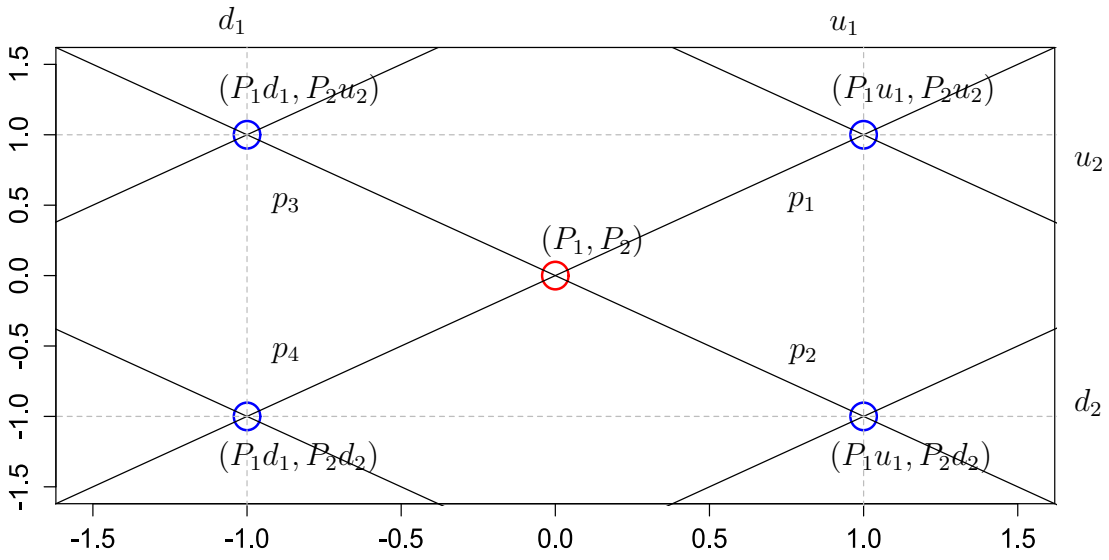

Figure 5.1.: Possible development of expected cost in a two project portfolio

Equipped with the new found drift and variance parameters we are able to manipulate the probability equations derived by Boyle, Evnine and Gibbs in [2] p 246 (11a) - (11d) for our cause. The four probabilities which model the development of estimated project cost (one case for decline of cost in both, one for rise of cost in both, two for decline in one and rise in the respective other) are shown in eq. 5.4 and visualized in diagram 5.1. For the purpose of clarity $P_i(t)$ is simplified to P_i.

$$
\begin{aligned}
p_1 &= \frac{1}{4}\left\{1+\rho+\sqrt{\eta}\left(\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2}\right)\right\} &&= \frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \\
p_2 &= \frac{1}{4}\left\{1-\rho+\sqrt{\eta}\left(\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2}\right)\right\} &&= \frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \\
p_3 &= \frac{1}{4}\left\{1-\rho+\sqrt{\eta}\left(-\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2}\right)\right\} &&= \frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \\
p_4 &= \frac{1}{4}\left\{1+\rho+\sqrt{\eta}\left(-\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2}\right)\right\} &&= \frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\}
\end{aligned}
\tag{5.4}
$$

To ensure convergence of the model we demand that all probabilities lie within the real probability space $p_i \epsilon [0,1]$ as indicated by equation 5.5.

$$0 \leq p_i \leq 1 \tag{5.5}$$

Since all parameters except the drift term are inherent to the model or at least subject to a minimum requirement in the case of Δt we will have to stick with $-\frac{I}{P(t)}$ in order to keep probabilities real. This is a clear contradiction to the base model from the literature which suggested small Δt values to control probabilities. While in our representation a certain minimum number of time-steps is necessary it is not recommendable to boost the lattice resolution to far beyond that magnitude since the amount of lattice nodes $\left(\frac{T}{\Delta t}\right)^{1+N}$ scales exponentially with the number of projects N.
To reduce the solution space to real probabilities we introduce the bounding factor ψ that restricts investment to a maximum technically tolerable value $I_{\max_t}$. .

$$0 \leq \frac{I_{1_{\max_t}}}{P_1}, \frac{I_{2_{\max_t}}}{P_2}, \ldots, \frac{I_{\max_t}}{P} \leq \psi \tag{5.6}$$

See equation 5.6 for the general form whereupon ψ is given by eq. 5.7. An elaborate discussion on the derivation of ψ can be found in Appendix B.

$$\psi = \min[1, \psi_1^0, \psi_2^0, \psi_3^0] = \min[1, \frac{1+\rho}{\sqrt{dt}} \times \frac{\beta_1\beta_2}{\beta_1+\beta_2}, \frac{1-\rho}{\sqrt{dt}}\beta_1, \frac{1-\rho}{\sqrt{dt}}\beta_2] \tag{5.7}$$

5.2. Application of the CRRA utility function

With the experience earned in chapter 4 we have no difficulties in drawing up an utility function for the momentary consumption i.e. investment (eq. 5.8). Both financial asset opportunities add up to a cumulative negative consumption which has to be made up by the future earnings (under the assumption of risk-aversion). Diversification arises if sufficient capital is available for both projects and the gradient is still positive after the first enterprise is fully financed. We will prove this theorem later on. Equation 5.9 points the utility function for an n-dimensional investment portfolio.

$$u\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right) \tag{5.8}$$

$$u\left(\frac{I_{\max_f} - I_1 - I_2 - \ldots - I_n}{I_{\max_f}}\right) \tag{5.9}$$

The constructed utility function is subject to a couple of constraints. In this work we will limit ourselves to the two-dimensional case thus the invested capital is restricted to values allowed by equations 5.10.

$$\begin{aligned} I_1 + I_2 &\leq I_{\max_f} \\ I_1 &\leq I_{1_{\max_t}} = \psi \times P_1 \\ I_2 &\leq I_{2_{\max_t}} = \psi \times P_2 \\ I_1 &\geq 0 \\ I_2 &\geq 0 \end{aligned} \tag{5.10}$$

Whereupon $I_{\max_f}$ is denoted by eq. 5.11. In principle the calculation of $I_{\max_f}$ is an arbitrary task in the sense that an investor is free to choose how to calculate that value or name it directly. A way of computing a reasonable value for $I_{\max_f}$ was shown in chapter 4.

$$I_{\max_f} = \min[I_{1_{\max_f}}, I_{2_{max_f}}] \tag{5.11}$$

Equation 5.12 reveals the extension of constraints to the n-dimensional case.

$$\begin{aligned} I_1 + I_2 + \ldots + I_n &\leq \min[I_{1_{\max_f}}, \ldots, I_{n_{max_f}}] \\ I_1 &\leq I_{1_{\max_t}} = \psi \times P_1 \\ \vdots &\quad \vdots \\ I_n &\leq I_{n_{\max_t}} = \psi \times P_n \\ I_1 &\geq 0 \\ \vdots &\quad \vdots \\ I_n &\geq 0 \end{aligned} \tag{5.12}$$

5.3. Derivation of probabilities and Bellman principle

The four possible cases that can develop from the current state of the project portfolio can be merged into a single vector $\vec{p}$ (eq.5.13) whose sum $\sum \vec{p} = 1$. Differentiating this vector with respect to I_1 and I_2 the probabilities gradient is obtained. In our two dimensional case it is represented by a 4×2 matrix (eq. 5.14).

$$\vec{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} \tag{5.13}$$

$$\nabla \vec{p} = \begin{pmatrix} \frac{\delta p_1}{\delta I_1} & \frac{\delta p_1}{\delta I_2} \\ \frac{\delta p_2}{\delta I_2} & \frac{\delta p_2}{\delta I_2} \\ \frac{\delta p_3}{\delta I_3} & \frac{\delta p_3}{\delta I_3} \\ \frac{\delta p_4}{\delta I_4} & \frac{\delta p_4}{\delta I_4} \end{pmatrix} = \frac{1}{4}\sqrt{\Delta t} \begin{pmatrix} \frac{-1}{P_1\beta_1} & \frac{-1}{P_2\beta_2} \\ \frac{-1}{P_1\beta_1} & \frac{1}{P_2\beta_2} \\ \frac{1}{P_1\beta_1} & \frac{-1}{P_2\beta_2} \\ \frac{1}{P_1\beta_1} & \frac{1}{P_2\beta_2} \end{pmatrix} \tag{5.14}$$

With the understanding acquired in chapter 4 the Bellman equation 5.15 for the two-dimensional case is easily constructed. Again this model is only applied if all other cases shown in equation 5.16 are dormant. A graphical illustration can be found in figure 5.2.

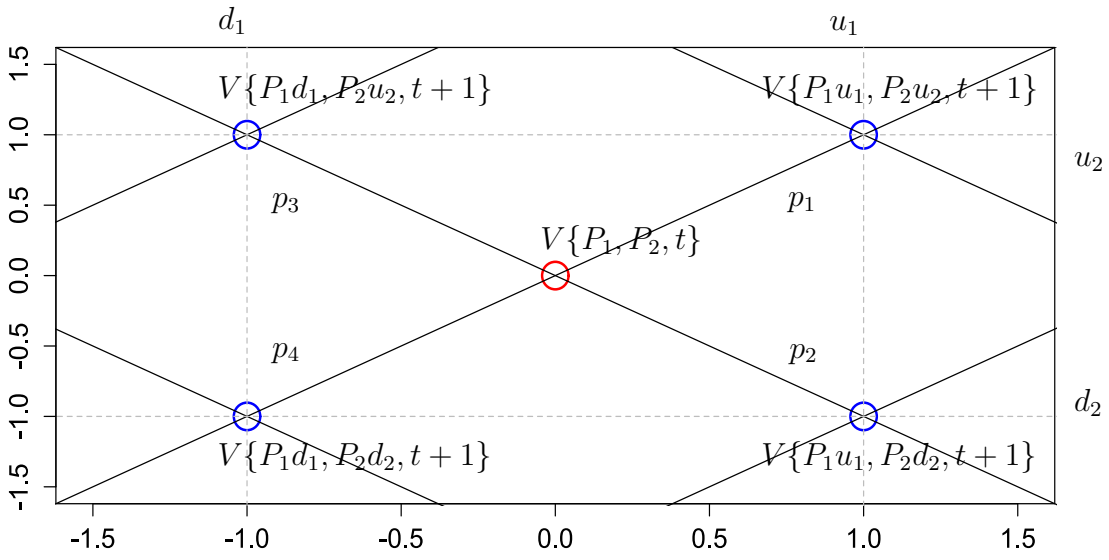

Figure 5.2.: Backward induction of option value from four subsequent lattice nodes

$$
\begin{aligned}
V\{P_1, P_2, t-1\} = \max_{I_i \in \mathbb{R}^+} [u\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right) \times \Delta t + + \frac{1}{(1+\kappa)^{\Delta t}} \times \dots \\
\times (p_1 \times V\{P_1u_1, P_2u_2, t\} + p_2 \times V\{P_1u_1, P_2d_2, t\} \\
+ p_3 \times V\{P_1d_1, P_2u_2, t\} + p_4 \times V\{P_1d_1, P_2d_2, t\})]
\end{aligned}
\tag{5.15}
$$

$$
V\{P_1, P_2, t-1\} = \begin{cases}
u\left(\frac{I_{\max_f}}{I_{\max_f}}\right) \times \Delta t & , t = T \wedge P_1 > P_{1_R} \\
 & \wedge P_2 > P_{2_R} \\
u\left(\frac{I_{\max_f}}{I_{\max_f}}\right) \times \Delta t + u\left(\frac{R_1}{P_{0_1}}\right) & , t \leq T \wedge P_1 \leq P_{1_R} \\
u\left(\frac{I_{\max_f}}{I_{\max_f}}\right) \times \Delta t + u\left(\frac{R_2}{P_{0_2}}\right) & , t \leq T \wedge P_2 \leq P_{2_R} \\
\max\limits_{I_i \in \mathbb{R}^+} [u\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right) \times \Delta t + + \frac{1}{(1+\kappa)^{\Delta t}} \times \dots & \\
\times (p_1 \times V\{P_1u_1, P_2u_2, t\} + p_2 \times V\{P_1u_1, P_2d_2, t\} & \\
+ p_3 \times V\{P_1d_1, P_2u_2, t\} + p_4 \times V\{P_1d_1, P_2d_2, t\})] & , t < T \wedge P_1 > P_{1_R} \\
 & \wedge P_2 > P_{2_R}
\end{cases}
\tag{5.16}
$$

If the Bellman equation is differentiated with respect to I_1 and I_2 we receive a gradient for the portfolio given in equations 5.17 and 5.18. Let us then establish that these differ

only in the algebraic sign of the contributing subsequent lattice nodes. Furthermore this optimization problem cannot be solved directly by calculus as the equation system is under-determined. Instead an iterative scheme as seen in chapter 4 has to be applied. The three point analysis introduced before is extended to an investigation of maximum five points. Three of those are inherent by the financial and technical constraints while the other two represent possible internal solutions.

$$\begin{aligned}\frac{\delta V}{\delta I_1} = \left(\frac{-1}{I_{\max_f}}\right)\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right)^{-\gamma}\Delta t + \frac{1}{(1+\kappa)^{\Delta t}} \times \frac{1}{4} \times \sqrt{\Delta t} \times \ldots \\ (-\frac{1}{P_1\beta_1} \times V\{P_1u_1, P_2u_2, t\} - \frac{1}{P_1\beta_1} \times V\{P_1u_1, P_2d_2, t\} + \ldots \\ \ldots \frac{1}{P_1\beta_1} \times V\{P_1d_1, P_2u_2, t\} + \frac{1}{P_1\beta_1} \times V\{P_1d_1, P_2d_2, t\})\end{aligned} \tag{5.17}$$

$$\begin{aligned}\frac{\delta V}{\delta I_2} = \left(\frac{-1}{I_{\max_f}}\right)\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right)^{-\gamma}\Delta t + \frac{1}{(1+\kappa)^{\Delta t}} \times \frac{1}{4} \times \sqrt{\Delta t} \times \ldots \\ (-\frac{1}{P_1\beta_1} \times V\{P_1u_1, P_2u_2, t\} + \frac{1}{P_1\beta_1} \times V\{P_1u_1, P_2d_2, t\} - \ldots \\ \ldots \frac{1}{P_1\beta_1} \times V\{P_1d_1, P_2u_2, t\} + \frac{1}{P_1\beta_1} \times V\{P_1d_1, P_2d_2, t\})\end{aligned} \tag{5.18}$$

Obviously the first point undergoing research will be zero i.e. no investment in neither project. But where should an investor proceed afterwards? The Hessian matrix shown in equation 5.19 provides us an helpful answer. In contrary to the linear problem seen in risk-neutral optimization, $\mathbb{H}V \neq 0$. Also the Hessian matrix is indefinite which tells us that the optimum is to be found on the edges of the solution space. Since the second derivative or the 'gradients gradient' has the same value ≤ 0 for every possible direction the gradients components develop in the same manner and the difference in values accrues exclusively from the follow-up nodes contribution. The original function may thus be optimized by following the strongest gradient.

$$\mathbb{H}V = \left(\frac{-1}{I_{\max_f}}\right)^2(-\gamma)\left(\frac{I_{\max_f} - I_1 - I_2}{I_{\max_f}}\right)^{-\gamma-1} \times \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \leq \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{5.19}$$

Before pursuing our search for the function's optimum we define $\mathbb{V}$. Its purpose is represent the elements of the vector-like gradient function sorted by their numeric value. Let the elements of $\mathbb{V}$ be equal to those of the Bellman equations gradient δV with $\mathbb{V}^1 = \max[\delta V]$, $\mathbb{V}^2$ holds the second largest element and so on. Note that this order need to be established only once e.g. at $I_1(t) = I_2(t) = 0$ and stays the same from there on since the Hessian matrix has the same numeric value in every direction for any combination of I_1 and I_2. Additionally we define I^1 and I^2 as investments in the projects with strongest or second-strongest gradient. For the purpose of clarity an illustration can be found in diagram 5.3. The figure shows only the linear portion of the gradient

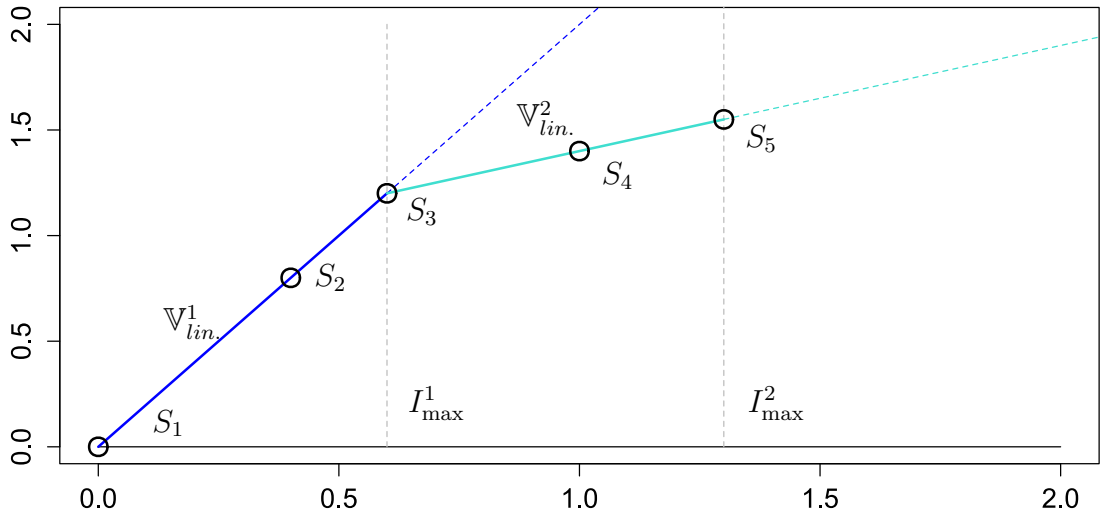

Figure 5.3.: Linear contribution of subsequent lattice points

resulting from subsequent grid points. Inner solution points occur where the tangent on the CRRA utility functions curve is parallel to the straight line obtained from the follow-up nodes.

$S_1 : \mathbb{V}^1 \leq 0$ **for** $\sum_{n=1}^{2} I^n = 0$ If the maximum element of the Bellman functions gradient δV denoted by $\mathbb{V}^1$ is less than or equal to zero for an investment of zero monetary units in the corresponding project waiting i.e. non-investment is recommended as within the constraints no other maximum can be found.

$S_2 : \mathbb{V}^1 > 0$ **for** $\sum_{n=1}^{2} I^n = 0$ **&** $\mathbb{V}^1 < 0$ **for** $\sum_{n=1}^{2} I^n = I^1_{\max}$ Once the gradient in the origin is positive the highest possible value for the according asset is tested. This can be either given by $I_{\max_f}$ or $I_{\max_t}$. If the gradient is less than zero at the boundary the optimum must lie on the I^1 axis. Since the investment value in the project with the second largest gradient I^2 is still zero the optimum number of monetary units for the most promising project can be directly calculated by equation 5.20. Note how $\frac{1}{P^1 \beta^1}$ and its algebraic sign $\pm$ is determined by the gradient itself.

$$\mathbb{V}^1 = 0 \rightarrow I^1 = I_{\max_f} - \{I_{\max_f}^{1-\gamma} \frac{1}{(1+\kappa)^{\Delta t}} \times \frac{1}{4} \times \frac{1}{\sqrt{\Delta t}} \times \dots \tag{5.20}$$
$$(-\frac{1}{P^1\beta^1} \times V\{P_1u_1, P_2u_2, t\} \pm \frac{1}{P^1\beta^1} \times V\{P_1u_1, P_2d_2, t\} \dots$$
$$\pm\frac{1}{P^1\beta^1} \times V\{P_1d_1, P_2u_2, t\} + \frac{1}{P^1\beta^1} \times V\{P_1d_1, P_2d_2, t\})\}^{-\frac{1}{\gamma}}$$

$S_3 : \mathbb{V}^1 > 0$ **for** $\sum_{n=1}^{2} I^n = 0$ **&** $\mathbb{V}^1 \geq 0$ **&** $\mathbb{V}^2 \leq 0$ **for** $\sum_{n=1}^{2} I^n = I_{\max}^1$ In case that the maximum gradient is positive at $I^1 = 0$ and still is at $I^1 = I_{\max}^1$ we need to evaluate the second largest gradient since further extension in the first direction is prohibited. If this second largest gradient is already less than null at the given amount invested the optimization process is stopped with $I^1 = I_{\max}^1$ and $I^2 = 0$ as best possible achievement. This point lies at a corner point of the solution space.

$S_4 : \mathbb{V}^1 > 0$ **for** $\sum_{n=1}^{2} I^n = 0$ **&** $\mathbb{V}^1, \mathbb{V}^2 > 0$ **for** $\sum_{n=1}^{2} I^n = I_{\max}^1$ **&** $\mathbb{V}^2 < 0$ **for** $\sum_{n=1}^{2} I^n = I_{\max}^2$ If however the gradient $\mathbb{V}^2$ is greater than 0 for $I^1 = I_{\max}^1$ testing of the function at the point $\sum_{n=1}^{2} I^n = I_{\max}^2$ is necessary. To perform this test we set $I^1 = I_{\max}^1$ and $I^2 = \min[I_{\max_t}^2, I_{max_f} - \sum_{n=1}^{1} I^n (= I^1)]$. When the gradient turns out to be negative at the point of interest we proclaim that an optimum on an edge with a I^2 coordinate $< I_{\max}^2$ exists. We stated earlier that the system of equations was under-determined and could not be solved for two unknown variables under current circumstances. With the successive approach tough it is possible since we already set $I^1 = I_{\max}^1$ one variable is already fixed and we can use it in the function of $\mathbb{V}^2$ to find the optimum value for I^2 as seen in equation 5.21.

$$\mathbb{V}^2 = 0 \rightarrow I^2 = I_{\max_f} - I_{(\max)}^1 - \{I_{\max_f}^{1-\gamma} \frac{1}{(1+\kappa)^{\Delta t}} \times \frac{1}{4} \times \frac{1}{\sqrt{\Delta t}} \times \dots \tag{5.21}$$
$$(-\frac{1}{P^2\beta^2} \times V\{P_1u_1, P_2u_2, t\} \pm \frac{1}{P^2\beta^2} \times V\{P_1u_1, P_2d_2, t\} \dots$$
$$\pm\frac{1}{P^2\beta^2} \times V\{P_1d_1, P_2u_2, t\} + \frac{1}{P^2\beta^2} \times V\{P_1d_1, P_2d_2, t\})\}^{-\frac{1}{\gamma}}$$

$S_5 : \mathbb{V}^1 > 0$ **for** $\sum_{n=1}^{2} I^n = 0$ **&** $\mathbb{V}^1, \mathbb{V}^2 > 0$ **for** $\sum_{n=1}^{2} I^n = I_{\max}^1$ **&** $\mathbb{V}^2 \geq 0$ **for** $\sum_{n=1}^{2} I^n = I_{\max}^2$ The last case worth mentioning is the one where both gradients are positive at null and the respective end points i.e. $\mathbb{V}^1 > 0$ for $I^1 = I_{\max_{(t)}}^1$ and $\mathbb{V}^2 > 0$ for $I^2 = I_{\max_{(t)}}^2$. As indicated by the parentheses this can only occur if the maximum tolerable investment is appointed by the technical values $I_{\max_t}^n$ rather than the financial limit $I_{\max_f}$. Thus it must satisfy $\sum_{n=1}^{2} I_{\max_t}^n < I_{\max_f}$.
Note: If all i.e both gradients return the same value for a testing point one is indifferent an can chose either one. In the underlying R implementation in such cases the gradient is chosen at random.

formula	source code	value	unit
P_0	priceGuess0	1000, 1000	[monetary units]
P_R	rewardMargin	10, 10	[monetary units]
λ	lambda	$\frac{P_R}{P_0}$	[-]
φ	shift	0	[-]
β	beta	0.20, 0.20	[-]
ρ	rho	+0.50	[-]
κ	kappa	0.05	[-]
r	r	0.05	[-]
T	Tmax	10	[time units]
R	reward	5000, 6000	[monetary units]

Table 5.1.: Project inherent variables

5.4. Dynamic programming results of portfolio investment

We are now eligible to perform a dynamic programming process of a portfolio consisting of two correlated projects. Therefore we extend all project-specific parameters (P_0, P_R, β, R) to a vector consisting of two values each. Furthermore we introduce the correlation factor ρ that incorporates the mathematical relation between the projects geometric random walks. We decide to settle $\rho = +0.50$ which indicates a light positive interrelationship of the projects compromising the portfolio. Also the value for project 2's reward is chosen slightly higher in order to avoid the arising of confusion by the random number generator if the gradients where equal to each other.
Concerning the selection of Δt this value is set to 0.05. This results in an overall time-step resolution of 200 which means that 201 nodes are to be found on the X-axis. We take this measure in order to keep computing time and, of even greater importance, memory consumption low. Since the number of lattice nodes is of the order $O(t^3_{Count})$ we already look at a pile of $\sim 8e+06$ points to calculate. Not to mention the $\sim 1e+09$ nodes if t_{Count} was kept at 1000. Calculations where performed with a risk-aversion coefficient of $\gamma = 1$.
The dynamic programming process results in a net-present-value of V(0) = 0.1953 with an overall chance of finishing any one project $\Pi(0) = 0.4359$. For the individual projects the success probabilities are $\Pi_1(0) = 0.0577$ and $\Pi_2(0) = 0.3872$. We note that the likelihood of completing project 2 dominates the chances of enterprise 1 because the optimization algorithm prefers the higher reward to estimated cost ratio and even more important suggests investing in project 2 right from the start.
Similar to previous chapters we continue to illustrate the optimized investment behaviour by the application of diagrams. First figures 5.4 and 5.5 are three dimensional scatter plots that present the suggested investors behaviour. On the t-axis we have the portfolio's time span (from zero to the terminal date $t = T = 10$) while the P1(t)- and

P2(t)-axes expand the field of possible random shock developments by one dimension each. So the P1(t)-axis corresponds to the remaining costs to completion of project 1 and reaches from $P_0^1 u_1^- 200 \sim 0.13$ to $P_0^1 u_1^+ 200 \sim 7.68e+06$. The side area of the P1(t)-axis contains point coloured in red. Those are the lattice nodes in which it is advised to invest into project 1. The same statements can be applied to project 2 and the P2(t)-axis i.e. a range of 400 steps that translates into an interval from 0.13 to $7.68e+06$ and yellow investment nodes. The three colours not mentioned yet represent the cases of waiting for a better opportunity to invest denoted by grey, transparent points, the suggestion of investment in both projects indicated by a purple, bluish coloring and the claim of reward in turquoise color. Note that only a subset of 20 out of the 200 time-steps is visualized in the pyramid graphs.
Diagrams 5.6 to 5.15 show nine profiles of the pyramid for different points in time. Obviously project 2 is preferred whenever possible due to its higher attainable reward. Figure 5.12 is an augmented view on the investment area for t = 6.25 and reveals the inversion of investment preferences for nearly finished portfolios. This is due to the fact that the dynamic programming algorithm is allowed to cash the reward from both projects but only from one point in the grid per time-step.

5.5. Monte Carlo study of portfolio investment

To conclude our short investigation of portfolio investment a Monte Carlo study is performed. Thanks to [2] we are able to operate with a single uniformly distributed random number. The dynamic programming procedure earned us a matrix containing probabilities for the movement in four different directions corresponding to the possible cases of up- and down-moves for two correlated projects. All that needs to be done is dividing up the uniform probability space $\in [0,1]$ to intervals representing the desired odds. Let ω be a uniformly distributed random number and p_1, p_2, p_3 and p_4 the probabilities of the different cases. Then the stochastic behaviour of the remaining costs to completion of the correlated projects is determined by the cases shown in equation 5.22.

$$\begin{aligned}
P_1u_1, P_2u_2 : &\quad 0 &&\leq \omega < &&\sum_{i=1}^{1} p_i \\
P_1u_1, P_2d_2 : &\quad \sum_{i=1}^{1} p_i &&\leq \omega < &&\sum_{i=1}^{2} p_i \\
P_1d_1, P_2u_2 : &\quad \sum_{i=1}^{2} p_i &&\leq \omega < &&\sum_{i=1}^{3} p_i \\
P_1d_1, P_2d_2 : &\quad \sum_{i=1}^{3} p_i &&\leq \omega \leq &&\sum_{i=1}^{4} p_i = 1
\end{aligned} \tag{5.22}$$

With this random number generation approach our Monte Carlo simulation was performed with results shown in tables 5.2 to 5.5.

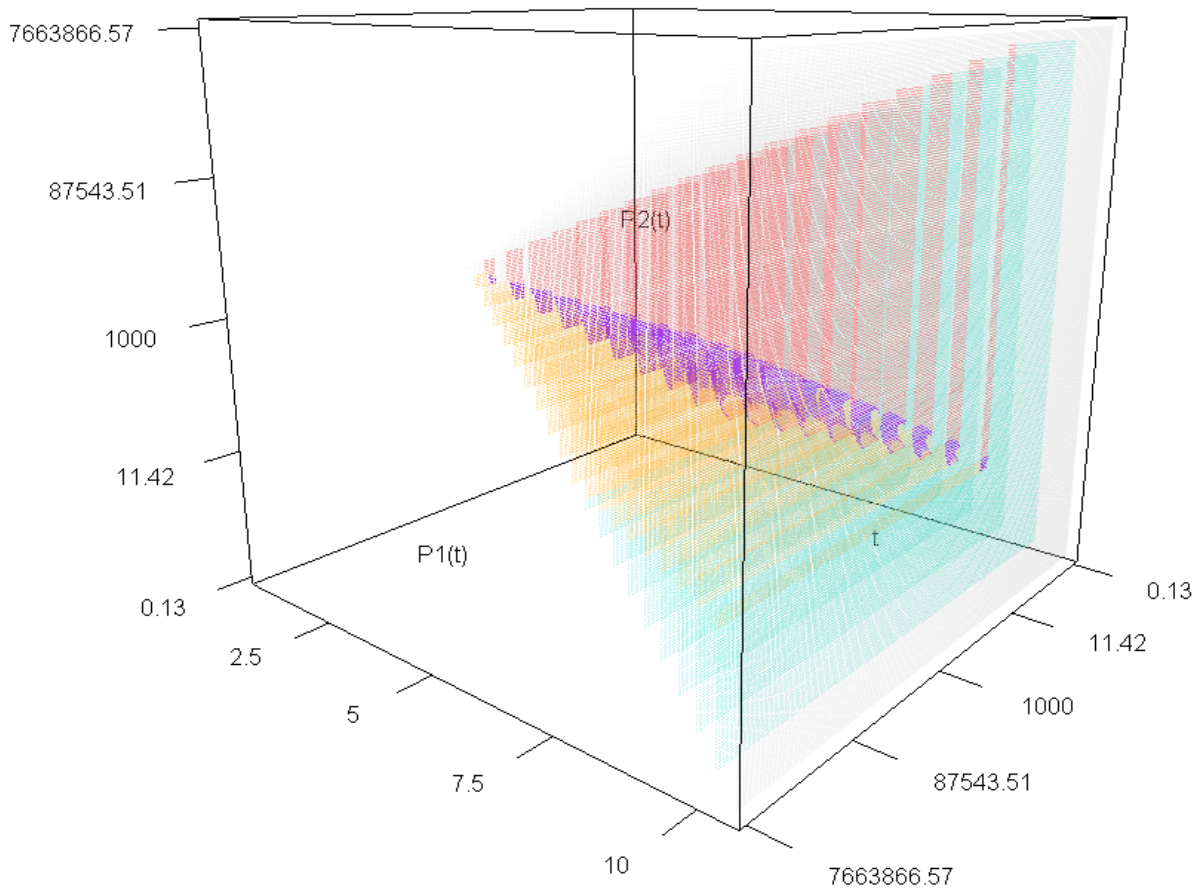

Figure 5.4.: Dynamic programming instruction results (tree illustration 1)

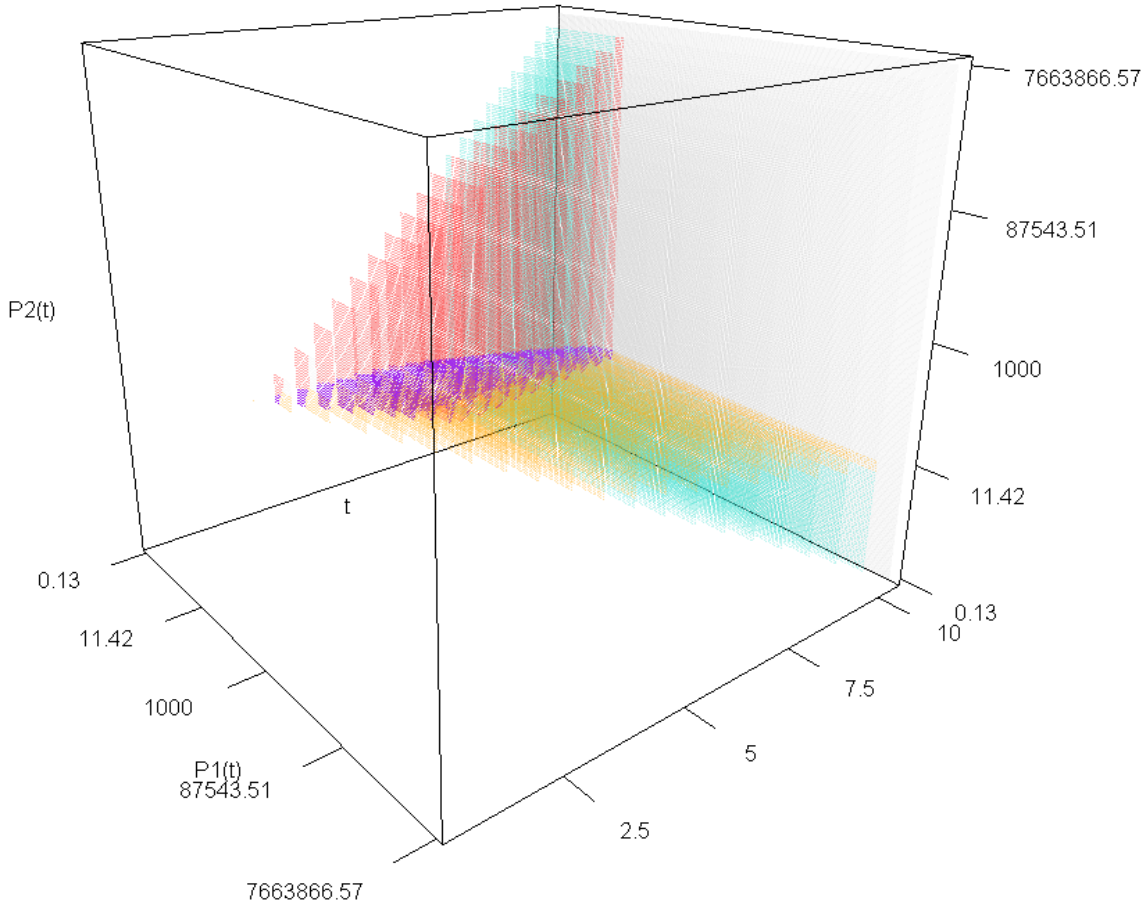

Figure 5.5.: Dynamic programming instruction results (tree illustration 2)

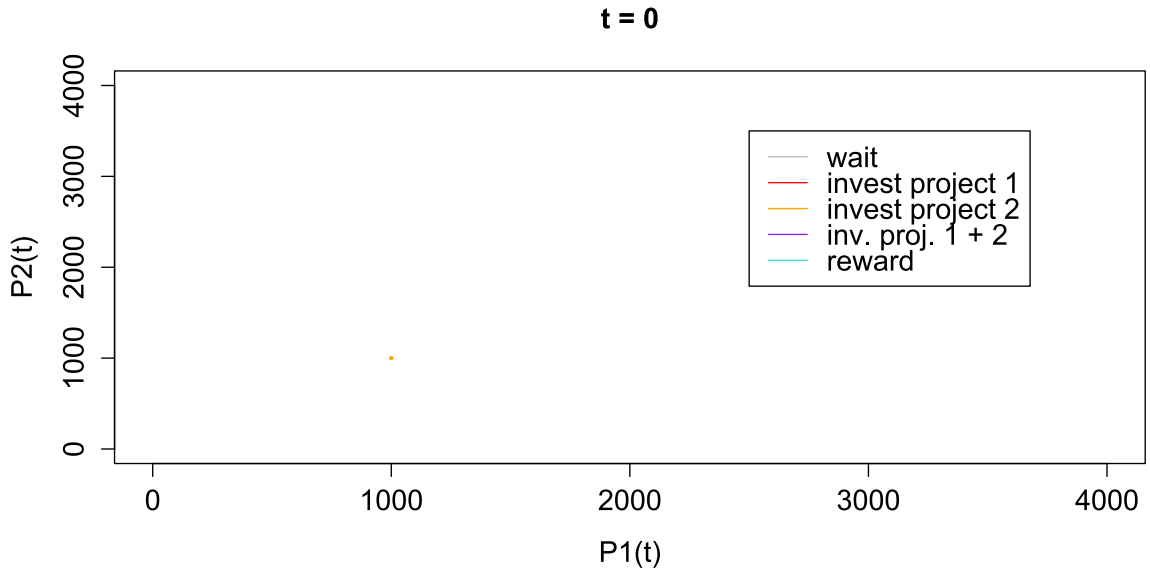

Figure 5.6.: Dynamic programming instruction results (t = 0)

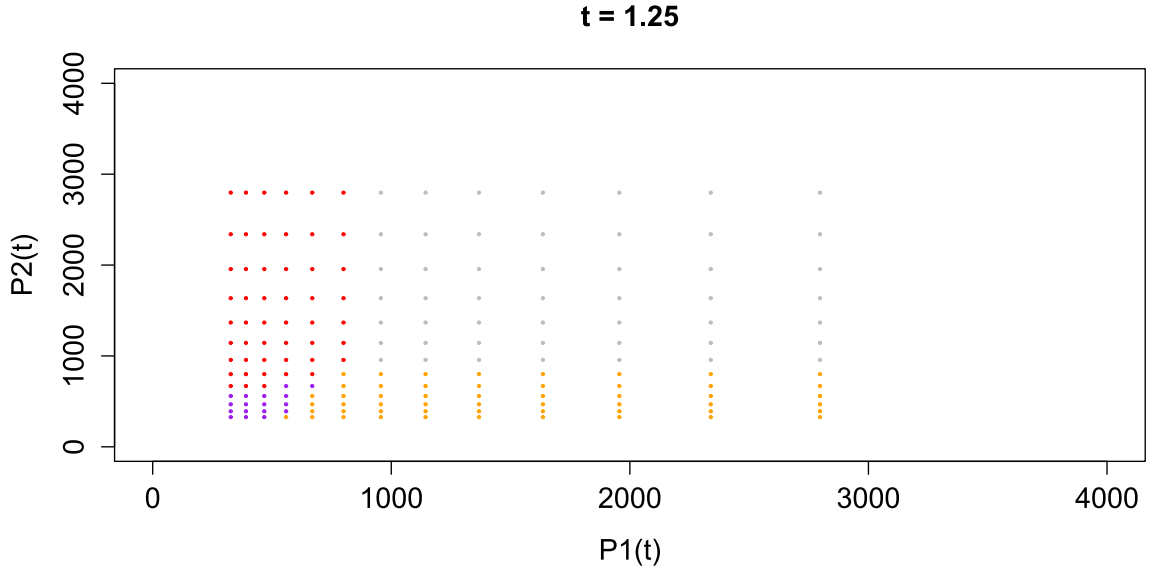

Figure 5.7.: Dynamic programming instruction results (t = 25)

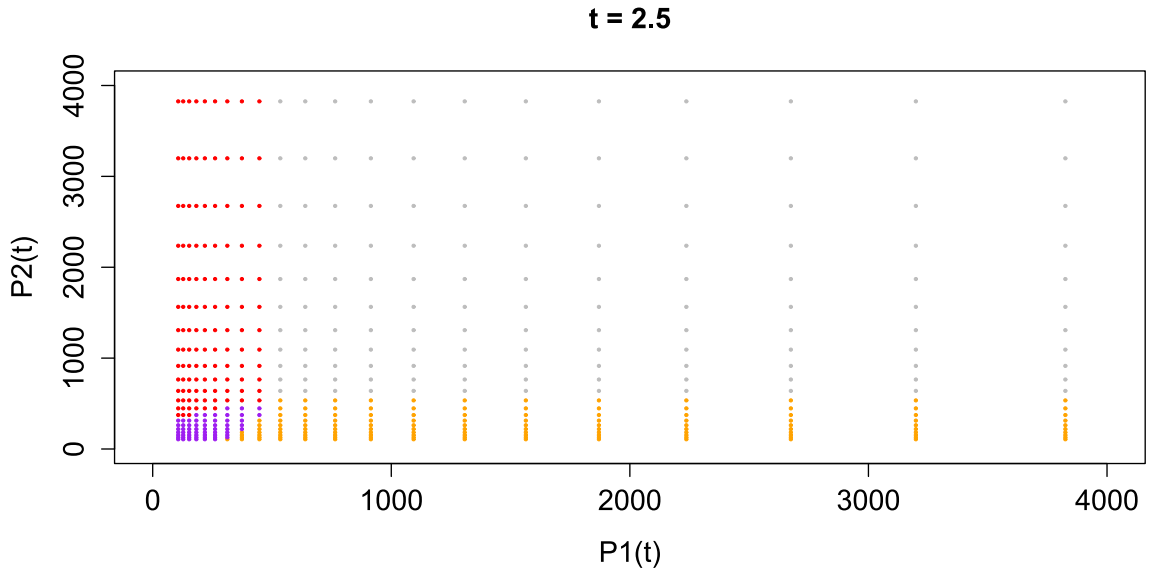

Figure 5.8.: Dynamic programming instruction results (t = 50)

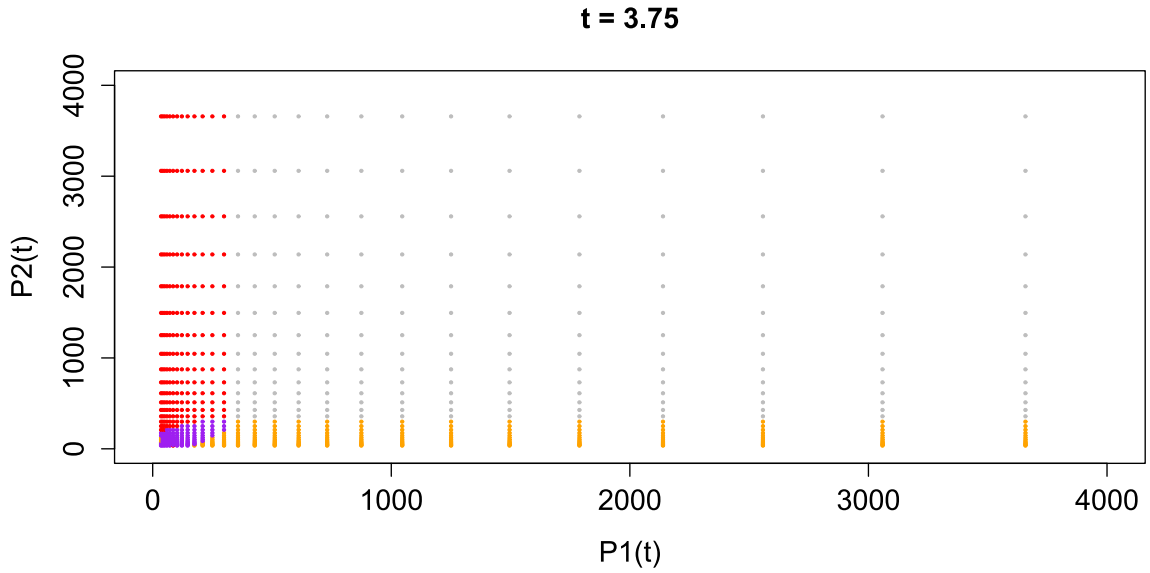

Figure 5.9.: Dynamic programming instruction results (t = 75)

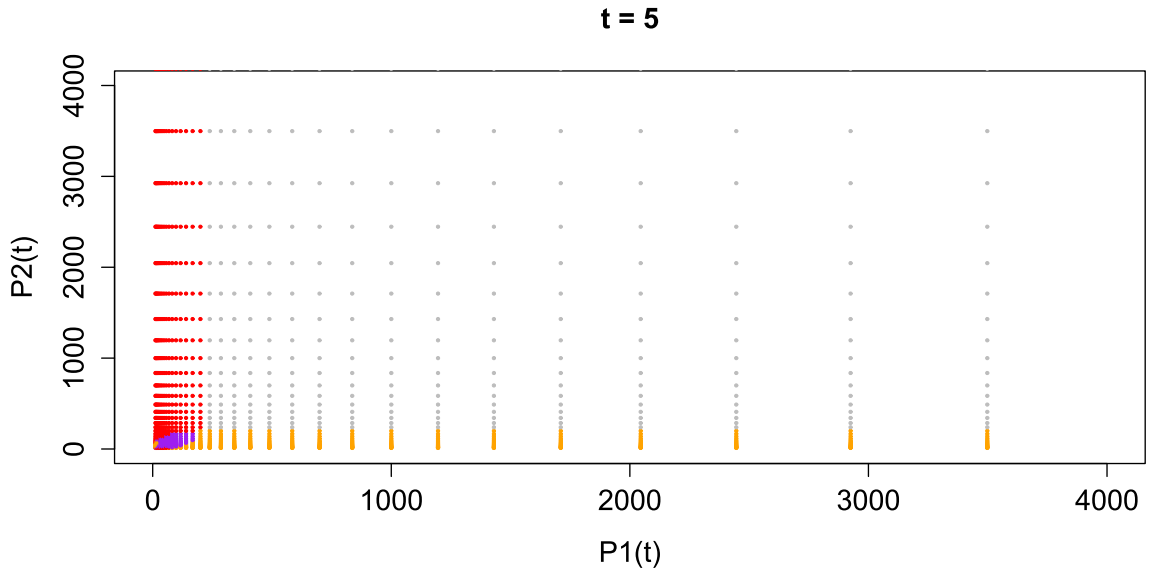

Figure 5.10.: Dynamic programming instruction results (t = 100)

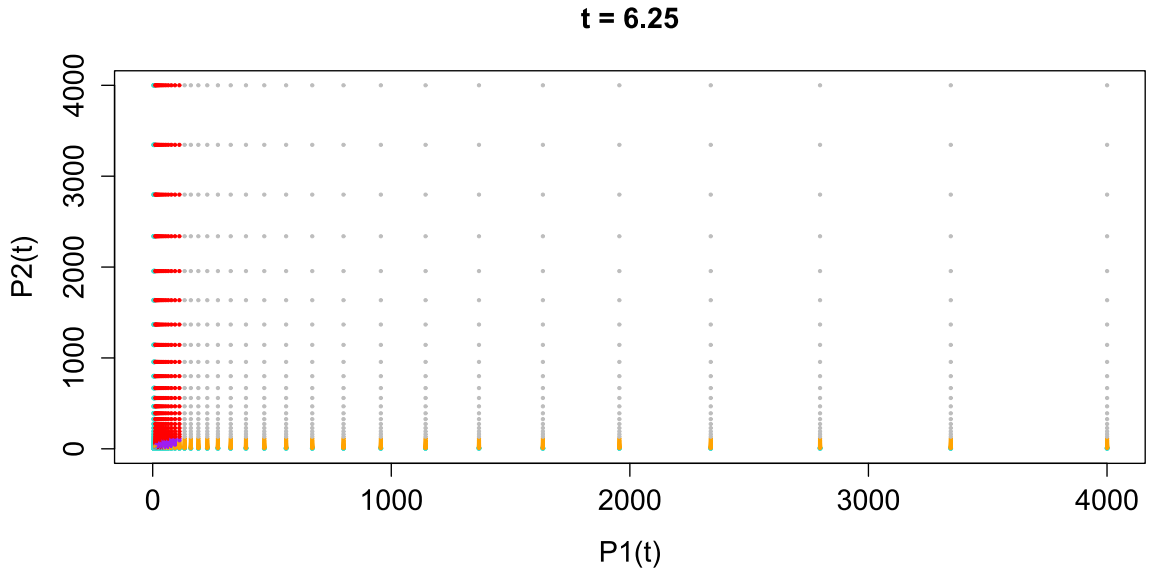

Figure 5.11.: Dynamic programming instruction results (t = 125)

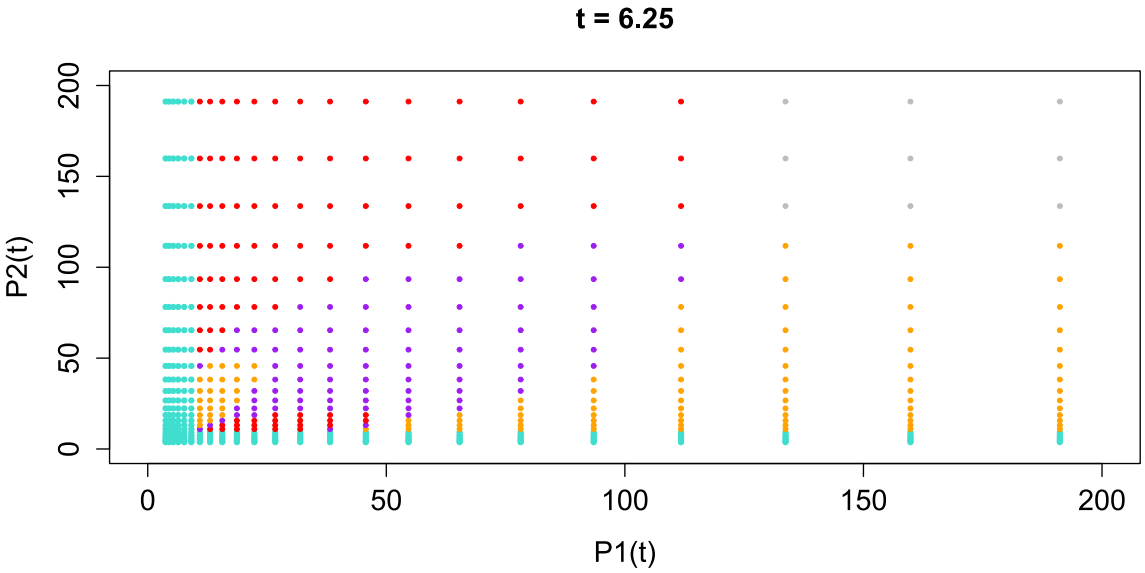

Figure 5.12.: Augmented view of investment area (t = 125)

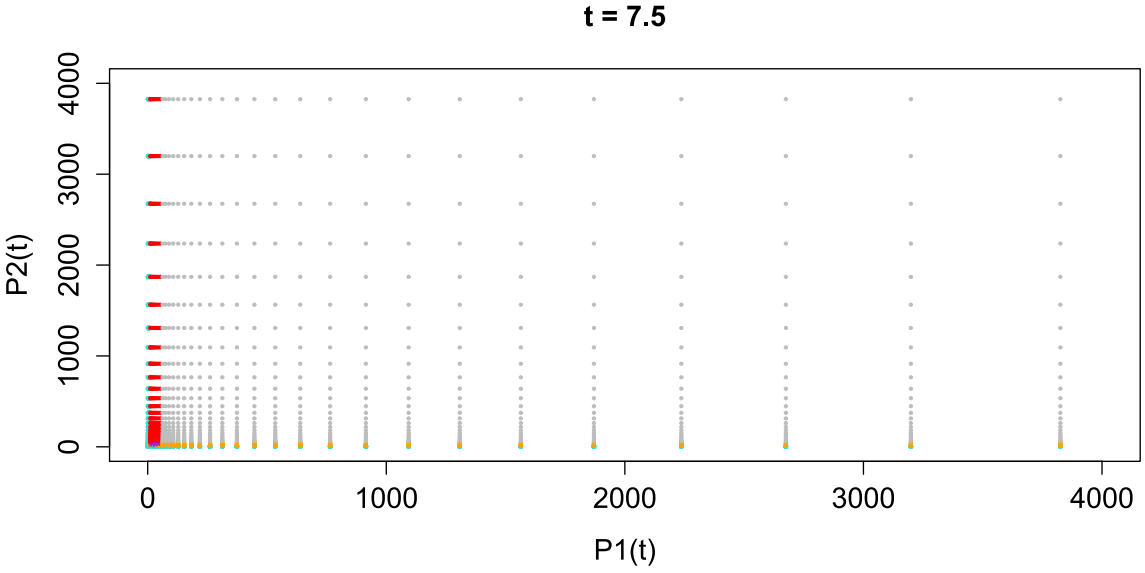

Figure 5.13.: Dynamic programming instruction results (t = 150)

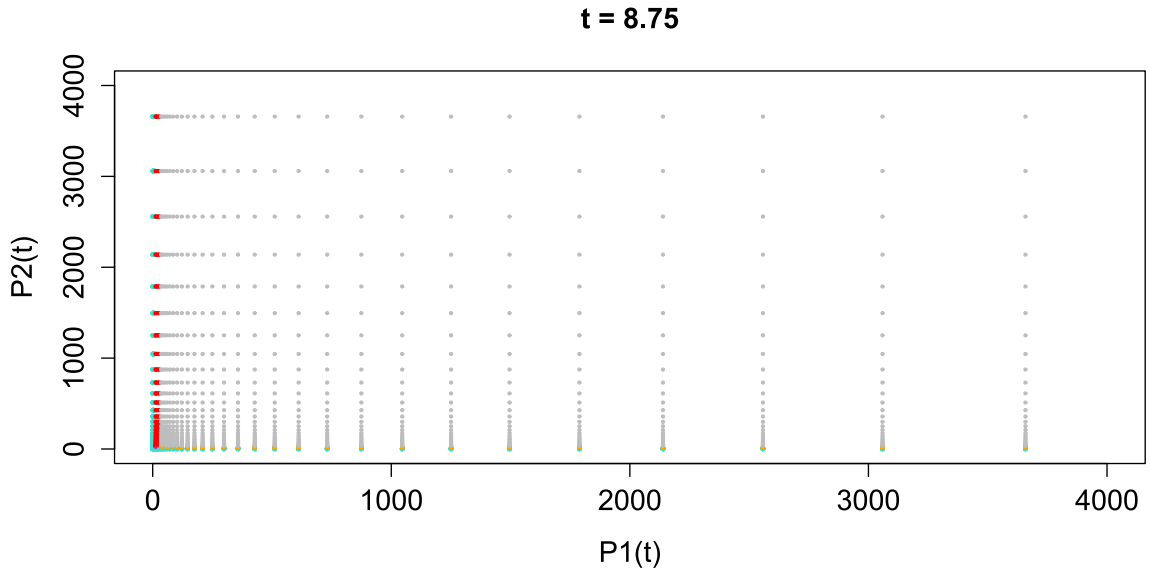

Figure 5.14.: Dynamic programming instruction results (t = 175)

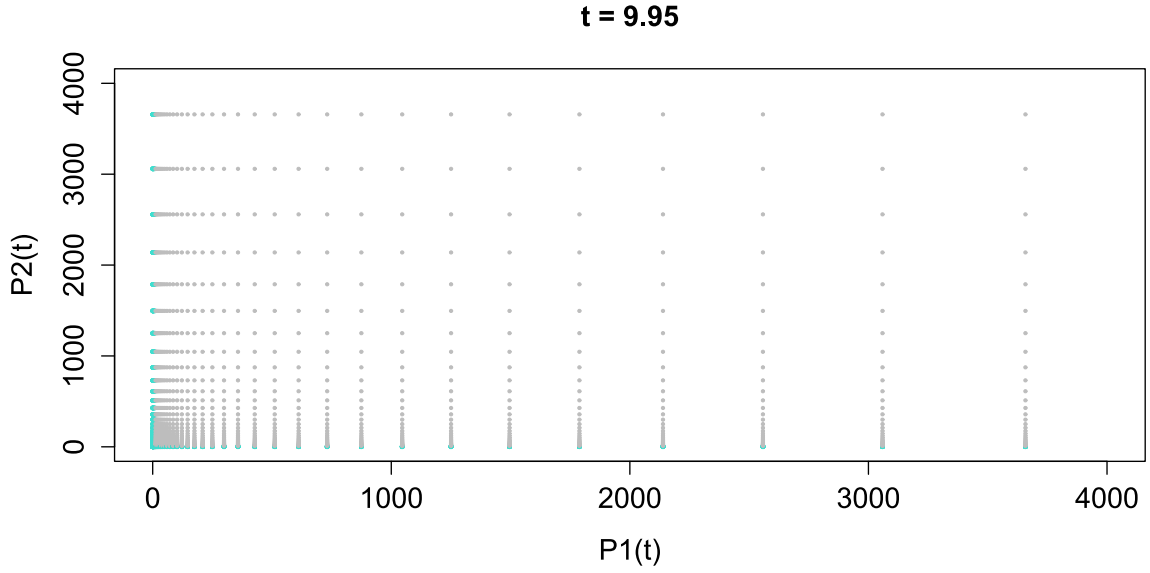

Figure 5.15.: Dynamic programming instruction results (t = 199)

	formula	value
Net-present-value	$V(0) =$	0.1953
Success probability	$\Pi(0) =$	0.4359
Success prob. proj. 1	$\Pi_1(0) =$	0.0577
Success prob. proj. 2	$\Pi_2(0) =$	0.3872

Table 5.2.: Dynamic programming results for portfolio investment

	formula	value
# simulations total	N =	20000
# projects simulated	n =	20000
% projects started	$\frac{n}{N} =$	1.00

Table 5.3.: Monte Carlo results for portfolio investment

With the suggestion of investing in project 2 from the very starting point of the dynamic optimization it is clear that the percentage of simulated projects equals 100%. The examination of project turnouts shows an ambivalent picture. While the cash-flow's mean is positive the median still indicates a loss and that more than half of the projects don't meet the reward margin in time. Also the standard deviation is quite high. A look at the remaining costs to completion reveals that while both projects are capable of reaching the reward distribution state (min value), project 2 is far more likely to achieve this (median). Supposedly this is due to its higher reward that permits the dynamic programming algorithm to carry the investment area to higher levels of remaining cost.

	value
mean	736.69
std dev	1997.82
median	- 568.08
max	6434.79
min	-2075.09
range	8509.89

Table 5.4.: project turnouts for portfolio investment

	project 1	project 2
mean	1082.79	200.27
std dev	891.02	487.60
median	914.44	14.94
max	10231.91	10231.91
min	9.99	9.99
range	10221.92	10221.92

Table 5.5.: End remaining cost for portfolio investment

6. Conclusion

Eventually it is time to close and recapitulate the findings of this work. First and most important is that time discrete dynamic programming models can act as a valid tool of evaluating projects or even portfolios of projects. The quality of the obtained predictions is however largely dependant on the foregoing modelling process. While example A1 shown in chapter 3 quickly returned reasonable results, the approach was rather simplistic and ignored the risk-avoiding aspect of human behaviour. It could however be proven that project funding with constant investment rates is, in terms of efficiency, inferior to relative funding. Also we could show that in a risk-neutral setting the optimum point in time to invest in a project is dependent on the intended rate of investment itself.
In the following section 4 a utility model, namely the CRRA function, was introduced to regard for risk aversion in decision making. Though this is a very promising approach it raised a few obstacles on its own. These sprouted from the newly accrued possibility of internal solutions since in the former model the only points of interest were waiting and investment at full rates. Finding these internal extrema turned out more complex than anticipated due to, at least in the beginning, poor modelling and issues with numeric optimization algorithms. Finally a revised mathematical model and a move from numeric to an analytic solution could overcome the problems. The techniques provide a method for evaluating future projects that is both simple in handling and a source of valuable information.
Lastly with example B found in chapter 5 an implementation of dynamic programming for portfolios of projects was discussed. This optimization problem is a logical extension of the single dimensional investment considerations shown in example A1 and A2. Figures 5.4 and 5.5 illustrate that there is a number of nodes where investment in both projects takes place. Furthermore we observed areas (figures 5.12) where the algorithm modified its investment pattern in order to claim the reward for both projects. While this outcome appears quite exciting one has to trade off the additional information in exchange for increased modelling and calculation effort.
Concerning future enhancement of the model let us state that there is plenty of room for further improvement. As for now the project representation in the models covers the aspect that Pindyck referred to as 'input cost uncertainty' in his 1993 publication [11] in the 'Journal of Financial Economics'. 'Technical uncertainty' and unknown project duration are not yet implemented in the model and will raise considerable amount of new questions upon their introduction. One of these would for instance be the abolishment of the Markov property i.e. path dependency by adoption of 'technical uncertainty' .
For now though let us conclude the discussion with the statement that the dynamic programming approach can be a very suitable tool for reviewing ventures of uncertain

cost and that this field of study still offers a vast jungle of unexplored possibilities for further investigation.

A. Appendix - Monte Carlo simulation and distribution analysis

In this chapter we employ the result matrices from dynamic programming to compute extensive Monte Carlo simulations ([4] sec. 12) with a sample size of $n = 20000$. Both the end remaining project cost at t = T and the overall cash-flows are analysed to evaluate the strategies. Moreover there will be statistical tests on the simulation outcomes to find out whether prices and/or cash-flows follow a certain statistical pattern (in our case logarithmic normal or gamma distribution). For this purpose we will conduct test with mathematical methods as well as use visualization techniques to judge the outcomes of the simulation.

Instruments of analysis

- Shapiro-Wilk test [12] for normality (i.e. normal distribution of the logarithm of remaining cost or cash-flows),
- Kolmogorov-Smirnov test ([4] section 10.6) used for logarithmic normal and gamma distribution,
- Q-Q plots ([4] section 11.3) as means of diagramming for normal distribution of the logarithm of remaining cost or cash-flows,
- Histograms ([4] sec. 3.7 and 6.2) for gamma, log-normal distribution plus normal distributions of the logarithm of the values.

A.1. Parameter estimation for continuous distributions

In order to test the obtained simulation samples for certain kinds of distributions the parameters of the generalised continuous density functions need to be estimated. Utilizing the maximum-likelihood estimator and the method of moments we can produce two sets of values for the log-normal distribution for each simulation sample. Parameters for a gamma function can also be computed. A comprehensive discussion of the tested distribution can be found in [4] chapter 5.

A.1.1. Logarithmic normal distribution

Maximum-likelihood Estimator

Found in [4] section 7.5.

$$\begin{aligned} \mu_{mlh} &= \frac{1}{N} \times \sum_{n=1}^{N} \ln\{x_i\} \\ \sigma^2_{mlh} &= \frac{1}{N} \times \sum_{n=1}^{N} (\ln\{x_i\} - \mu_{mlh})^2 \end{aligned} \tag{A.1}$$

Method of moments

Detailed in [4] section 7.6.

$$\begin{aligned} E &= \frac{1}{N} \times \sum_{n=1}^{N} \{x_i\} \\ Var &= \frac{1}{N} \times \sum_{n=1}^{N} (\{x_i\} - E)^2 \\ \mu_{mom} &= \ln\left(E^2 \frac{1}{\sqrt{Var + E^2}}\right) \\ \sigma^2_{mom} &= \sqrt{\ln\left(\frac{Var}{E^2} + 1\right)} \end{aligned} \tag{A.2}$$

A.1.2. Gamma distribution

Equations A.3 for the computation of the distribution parameters are derived from theorem 5.7.5 in [4].

$$\begin{aligned} E &= \frac{1}{N} \times \sum_{n=1}^{N} \{x_i\} \\ Var &= \frac{1}{N} \times \sum_{n=1}^{N} (\{x_i\} - E)^2 \\ p &= \frac{E^2}{Var} \\ b &= \frac{E}{Var} \end{aligned} \tag{A.3}$$

A.2. Cash-flow analysis of actually launched projects

Now we will devote some time to the investigation of cash-flows in actual projects. For strategy A1-0 cash-flow analysis is omitted as no investments are undertaken and therefore no projects are finished. The possibility of earning the reward without investment is neglected.

A.2.1. Logarithmic normal distribution of cash-flows

Hypothesis

- H_0 : The random variable cf_x is $\mathcal{LN}(\mu, \sigma^2)$ distributed, w. $\mu \in \mathbb{R}$ and $\sigma^2 > 0$
- H_1 : The random variable cf_x is <u>not</u> $\mathcal{LN}(\mu, \sigma^2)$ distributed

Maximum-likelihood method	formula	A1-0	A1-1	A1-2
mean	μ_{mlh}	-	6.696131	7.018453
standard deviation	σ_{mlh}	-	0.399790	0.511219
variance	σ^2_{mlh}	-	0.159832	0.261345

Table A.1.: Cash-flow parameters for log-norm distribution by strategy

Maximum-likelihood method	formula	A1-0	A1-1	A1-2
mean	μ_{mom}	-	6.688090	7.048836
standard deviation	σ_{mom}	-	0.417944	0.401789
variance	σ^2_{mom}	-	0.174677	0.161434

Table A.2.: Cash-flow parameters for log-norm distribution by strategy

Logarithmic normal distribution	formula	A1-0	A1-1	A1-2
p - value	p_{mlh}	-	2.2 e-16	2.2 e-16
	p_{mom}	-	2.2 e-16	2.2 e-16

Table A.3.: p - value for log-norm distribution of cash-flows by strategy

Viewing at histogram A.1 we look at a well behaved distribution of profits with a minor risk of loosing the monetary units invested. One might be tempted to consider the spreading of results as some kind of inverted log-normal distribution, yet it has to be proven mathematically. The histogram diagrammed in figure A.2 on the other hand is far of from a log-normal or gamma distribution. The Kolmogorov-Smirnov method corroborates our notion that neither of the strategies cash-flow follow a logarithmic normal distribution. Due to the relatively small artificial drift for high price ranges

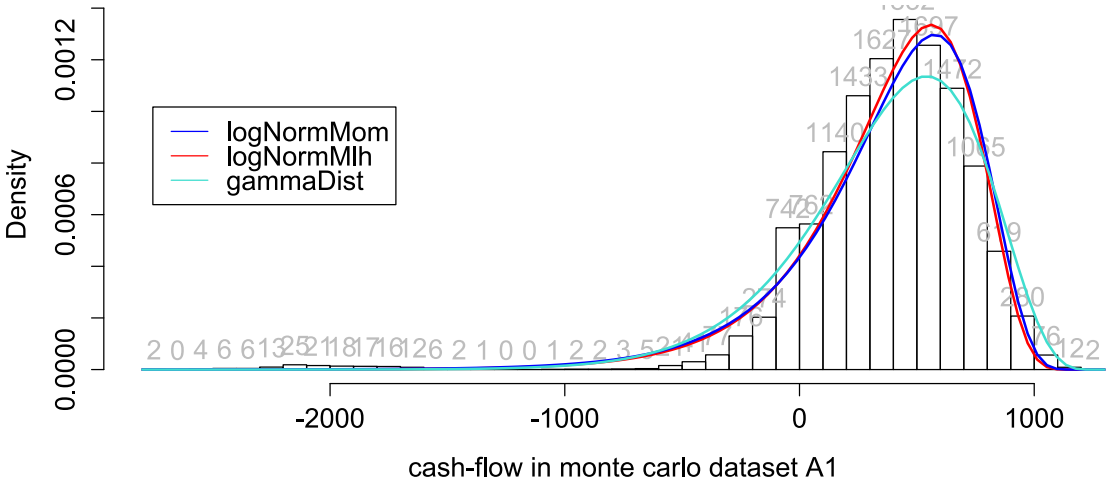

Figure A.1.: Histogram of simulated project cash-flow for A1-1 (n = 20000)

there is a significant chance of loosing the invested amount of monetary units. Also the frequency of losses is as big or even higher than the occurrence of profits. Concluding the investigation on histograms of project turnouts we state that constant absolute rates of investment are less useful the relative investments. This is due to over-investment in the areas where the project is nearly finished and to little activity in the beginning, which is crucial for project success. Especially the peak of the histogram shortly below zero is an indicator of poorly dimensioned investment volume.

A.2.2. Normal distribution of logarithm of cash-flows

Hypothesis

- H_0 : The random variable $\log\{cf_x\}$ is $\mathcal{N}(\mu, \sigma^2)$ distributed, with $\mu \in \mathbb{R}$ and $\sigma^2 > 0$
- H_1 : The random variable $\log\{cf_x\}$ is <u>not</u> $\mathcal{N}(\mu, \sigma^2)$ distributed

Normal distribution	formula	A1-0	A1-1	A1-2
p - value	p	-	2.2 e-16	2.2 e-16

Table A.4.: p - value for normal distribution of log{cash-flows} by strategy

As the R implementation of the Shapiro-Wilk test runs without precomputed values calculation of μ and σ^2 is neglected. The tests result for $\mathcal{N}(\mu, \sigma^2)$ are carried out with a sample size of n = 2000 with replacement. The p value is 2.2 e-16 for all tested

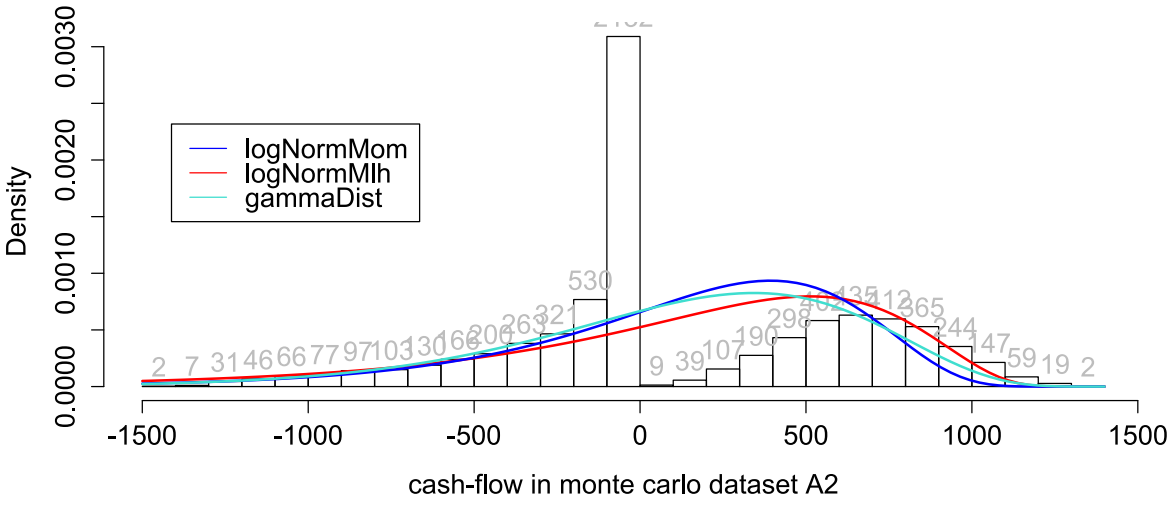

Figure A.2.: Histogram of simulated project cash-flow for A1-2 (n = 20000)

strategies → H_0 is discarded and H_1 accepted. The accompanying figures A.3 and A.5 give rise to ambivalent thoughts. While the histogram again may be misleading towards an assumption of normal distribution of the values the Q-Q plot clearly indicates that the logarithm of the simulated cash flow is not normally distributed. Figures A.4 and A.6 convey the same information on strategy A1-2. Both, the Q-Q plot and the histogram, indicate that the resulting distribution of cash-flows is certainly not normal distributed.

A.2.3. Gamma distribution of cash-flows

Hypothesis

- H_0 : The random variable cf_x is $\gamma(p, b)$ distributed, with $p > 0$ and $b > 0$
- H_1 : The random variable cf_x is not $\gamma(p, b)$ distributed

The Kolmogorov-Smirnov tests for $\gamma(p, b)$ result in p = $2.2e-16$ and therefore could not yield a significant evidence for gamma distribution → H_0 discarded and H_1 accepted.

parameter	formula	A1-0	A1-1	A1-2
p - value	p	-	5.239394	5.707914
b - value	b	-	0.005980	0.004572

Table A.5.: Cash-flow parameters for log-norm distribution by strategy

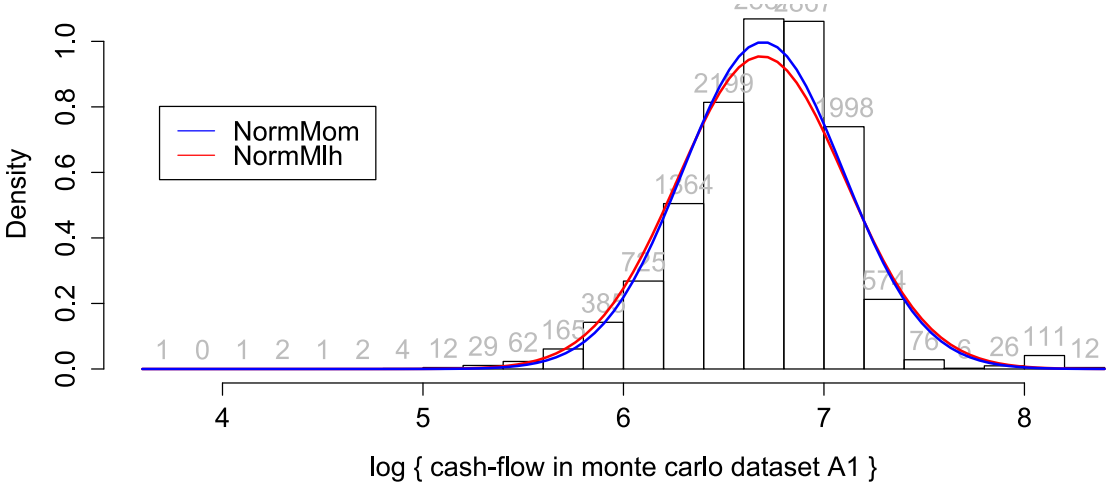

Figure A.3.: Histogram of logarithm of simulated project cash-flow for A1-1 (n = 20000)

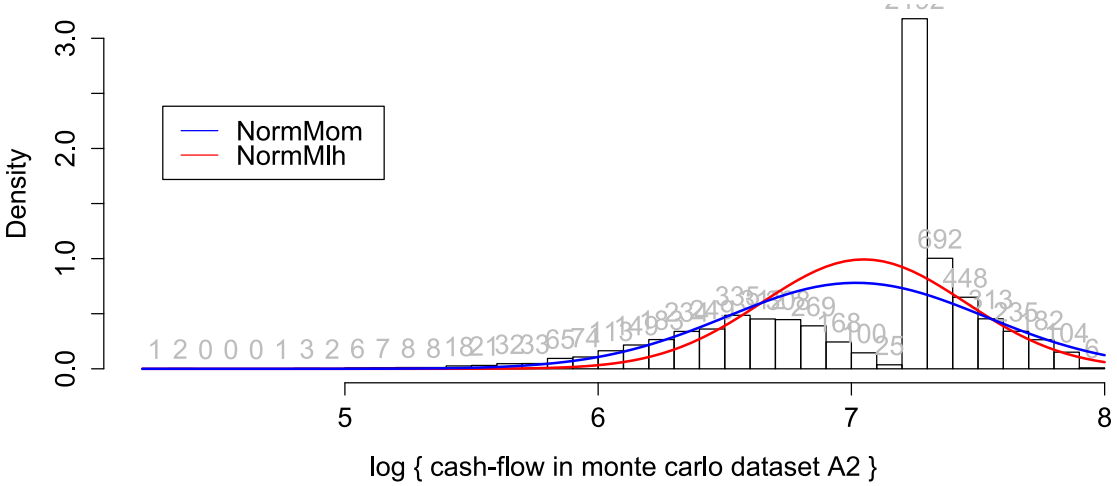

Figure A.4.: Histogram of logarithm of simulated project cash-flow for A1-2 (n = 20000)

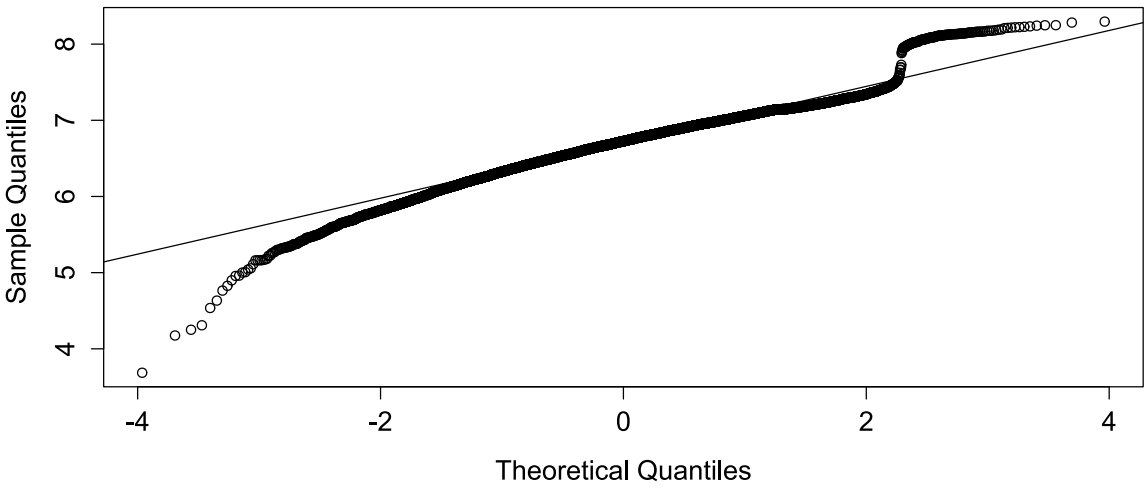

Figure A.5.: Histogram of logarithm of simulated project cash-flow for A1-1 (n = 20000)

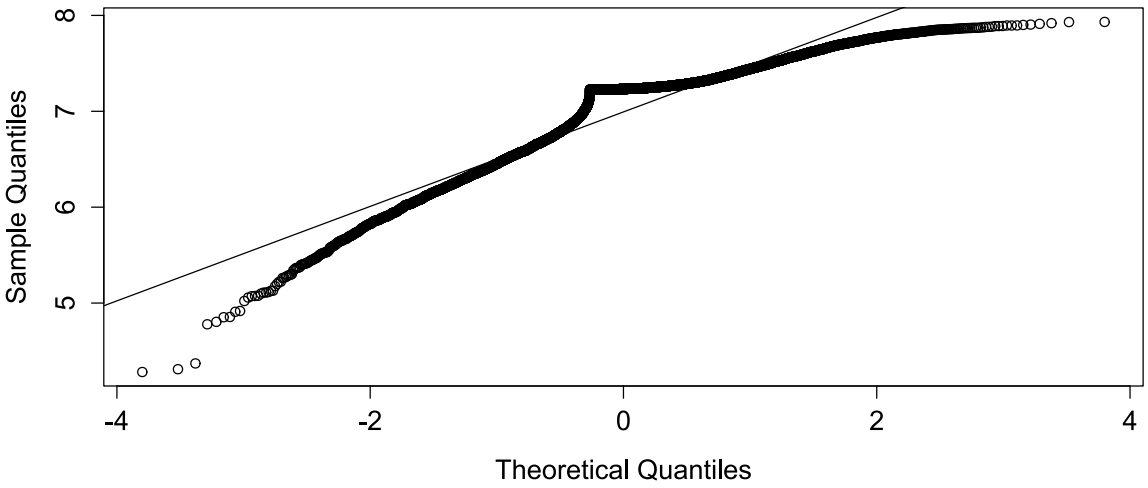

Figure A.6.: Histogram of logarithm of simulated project cash-flow for A1-2 (n = 20000)

Gamma distribution	formula	A1-0	A1-1	A1-2
p - value	p	-	2.2 e-16	2.2 e-16

Table A.6.: p - value for normal distribution of log{cash-flows} by strategy

A.3. Analysis of remaining cost at t = T

In the following section the simulated remaining cost at t = T will undergo different test procedures in order to allow us a review of the behaviour of the geometric random walk with and without management interference. Tests for logarithmic normal distribution (direct and indirect as test for normality of the logarithm of the simulated values) as well as tests for gamma distribution are conducted.

A.3.1. Logarithmic normal distribution of end remaining cost

Hypothesis

- H_0 : The random variable $P(T)_x$ is $\mathcal{LN}(\mu, \sigma^2)$ distributed, w. $\mu \in \mathbb{R}$ and $\sigma^2 > 0$
- H_1 : The random variable $P(T)_x$ is <u>not</u> $\mathcal{LN}(\mu, \sigma^2)$ distributed

Figures A.7 to A.9 illustrate the simulated cost values as histograms and also display the continuous distributions the obtained samples are tested for. Parameters for the continuous functions are derived from the samples by means of maximum-likelihood-method and method of moments as introduced above. Graphs for the gamma distribution which we will test later on are also present in this graph. For A.7 we are able to state that all three continuous distributions follow the histogram quite well with the gamma distribution being most deviant. Concerning diagram A.8 it is obvious that our intervention by financing the project warped the initial log-normal distribution of end remaining cost values seen in A.7 beyond recognition. Figure A.9 is kind of in the middle between the former two histograms. The deviations are not as distinct as with constant relative investment, yet the distribution can't be regarded as logarithmic normal any more.

Maximum-likelihood method	formula	A1-0	A1-1	A1-2
mean	μ_{mlh}	7.405894	2.616356	6.437208
standard deviation	σ_{mlh}	0.629455	3.675115	2.637931
variance	σ^2_{mlh}	0.396214	13.506470	6.958678

Table A.7.: End remaining cost parameters for log-norm distribution by strategy

From the simulation results we conclude that the uninfluenced geometric random walks strives towards a logarithmic normal distribution provided the numbers of samples and time-steps per simulation are large enough. Financial intervention by the project management on the other hand had a huge impact on the simulated outcomes of remaining

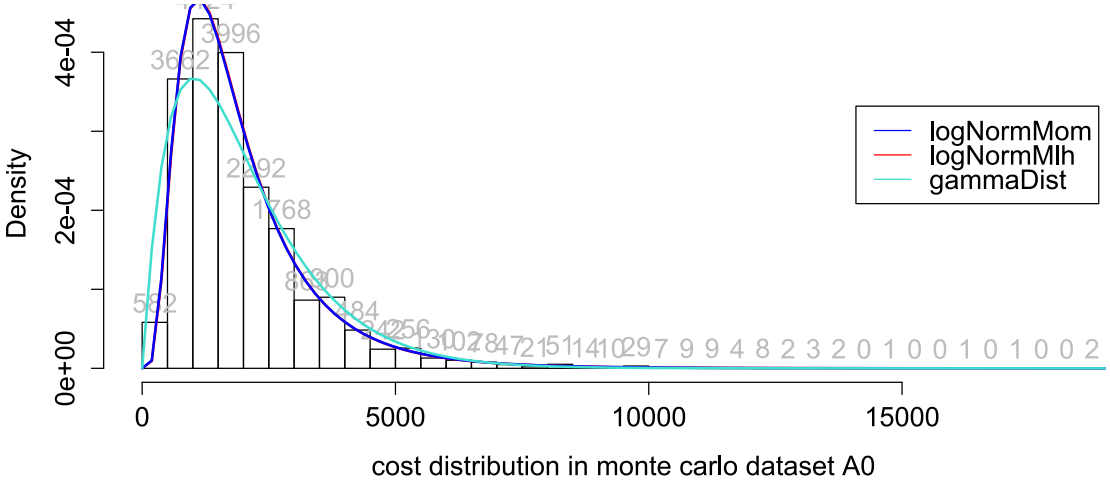

Figure A.7.: Histogram of simulated remaining cost at t = T for A1-0 (n = 20000)

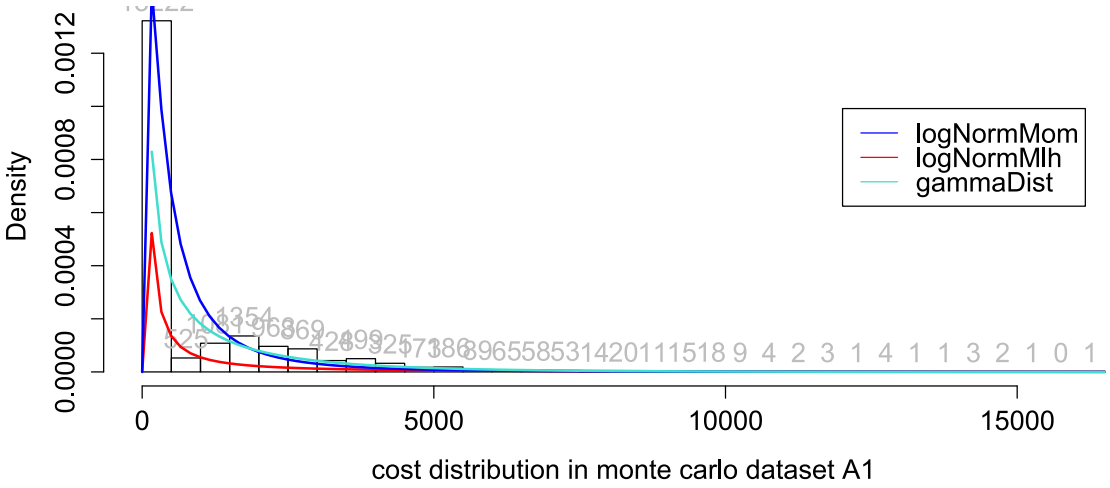

Figure A.8.: Histogram of simulated remaining cost at t = T for A1-1 (n = 20000)

Maximum-likelihood method	formula	A1-0	A1-1	A1-2
mean	μ_{mom}	7.403997	6.093647	7.246390
standard deviation	σ_{mom}	0.633139	1.187244	0.735184
variance	σ^2_{mom}	0.400865	1.409548	0.540494

Table A.8.: End remaining cost parameters for log-norm distribution by strategy

Logarithmic normal distribution	formula	A1-0	A1-1	A1-2
p - value	p_{mlh}	0.0001062	2.2 e-16	2.2 e-16
	p_{mom}	4.003 e-05	2.2 e-16	2.2 e-16

Table A.9.: p - value for log-norm distribution of end remaining cost by strategy

cost at t = T. The resulting distribution is far off from random and thus the artificially imposed drift of the Wiener-Process appears to have worked quite well. We note that the effects of constant absolute investments are not as distinct as with relative rates while they still invoke the desired drift of the process.

A.3.2. Normal distribution of logarithm of end remaining cost

Hypothesis

- H_0 : The random variable $\log\{P(T)_x\}$ is $\mathcal{N}(\mu, \sigma^2)$ distributed, with $\mu \in \mathbb{R}$ and $\sigma^2 > 0$
- H_1 : The random variable $\log\{P(T)_x\}$ is <u>not</u> $\mathcal{N}(\mu, \sigma^2)$ distributed

Since the R implementation of the Shapiro-Wilk test doesn't require precomputed parameters the calculation of μ and σ^2 is omitted. The test result of strategy A1-0 for $\mathcal{N}(\mu, \sigma^2)$ with a sample size of n = 2000 with replacement is p = 0.1572 $\rightarrow$ H_0 cannot be discarded. Diagrams A.10 to A.12 present histograms of the logarithm of the simulated data while A.13 to A.15 add easily assessable Q-Q plots. Figures A.10 and A.13 show that the logarithm of the simulated cost follows a normal distribution quite well. Again strategy A1-1 appears to be very much distorted from the initial normal distribution while investment with constant absolute rates is the compromise between the two extremes.

Normal distribution	formula	A1-0	A1-1	A1-2
p - value	p	0.1572	2.2 e-16	2.2 e-16

Table A.10.: p - value for normal distribution of log{cash-flows} by strategy

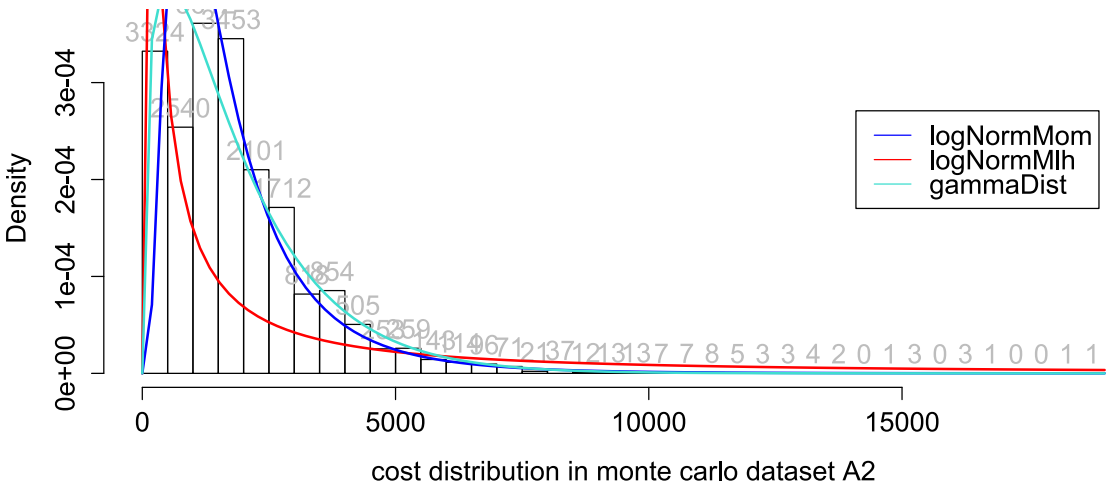

Figure A.9.: Histogram of simulated remaining cost at t = T for A1-2 (n = 20000)

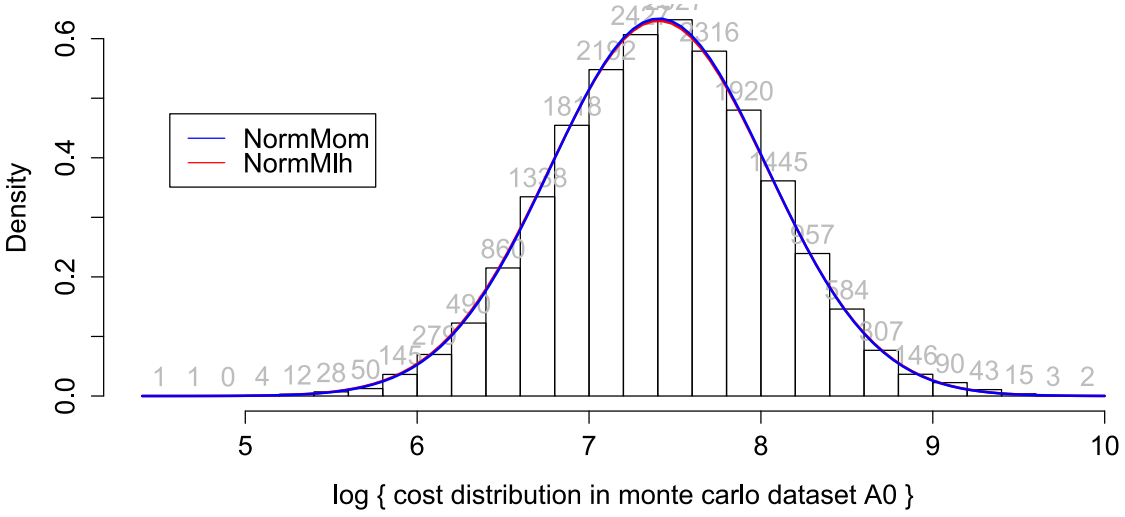

Figure A.10.: Histogram of logarithm of simulated remaining cost at t = T for A1-0 (n = 20000)

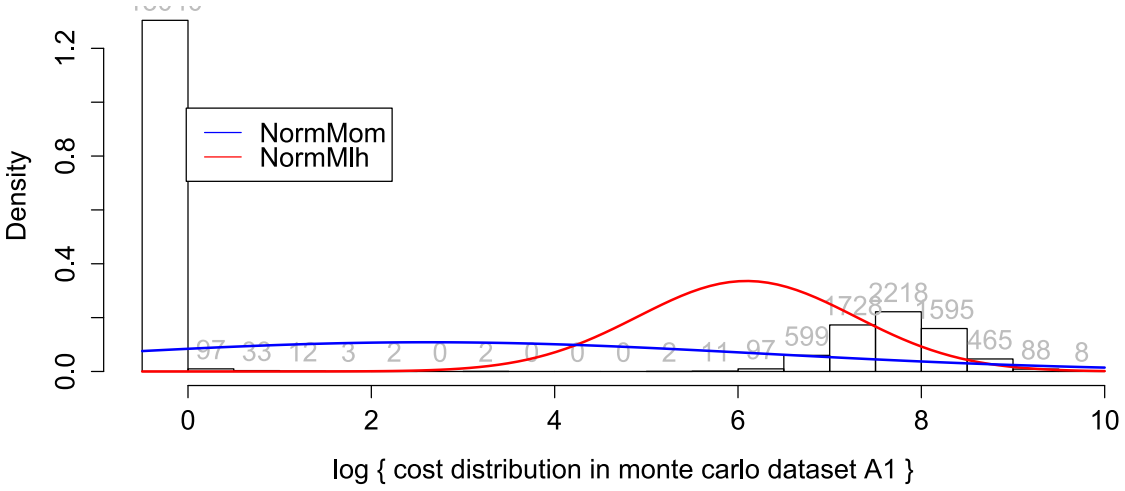

Figure A.11.: Histogram of logarithm of simulated remaining cost at t = T for A1-1 (n = 20000)

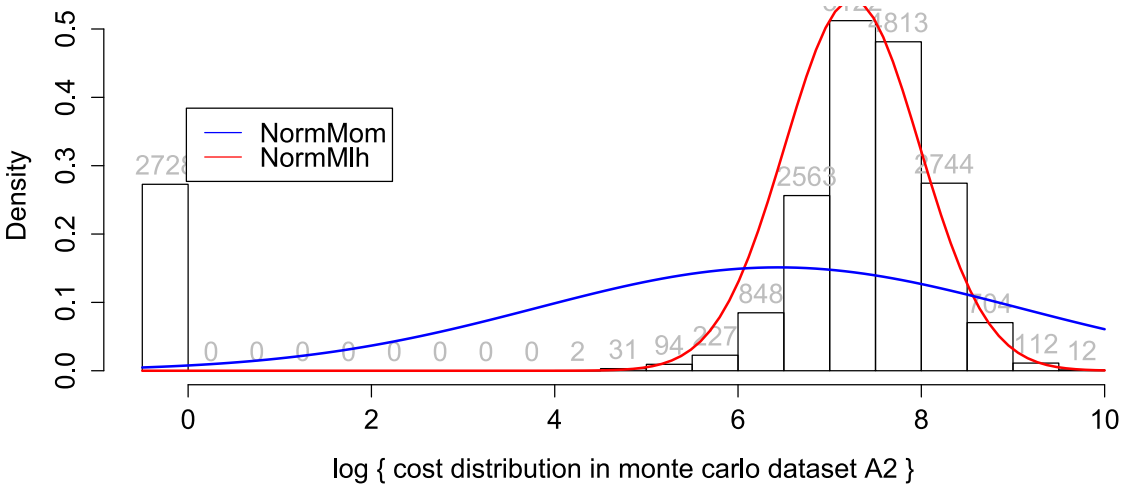

Figure A.12.: Histogram of logarithm of simulated remaining cost at t = T for A1-2 (n = 20000)

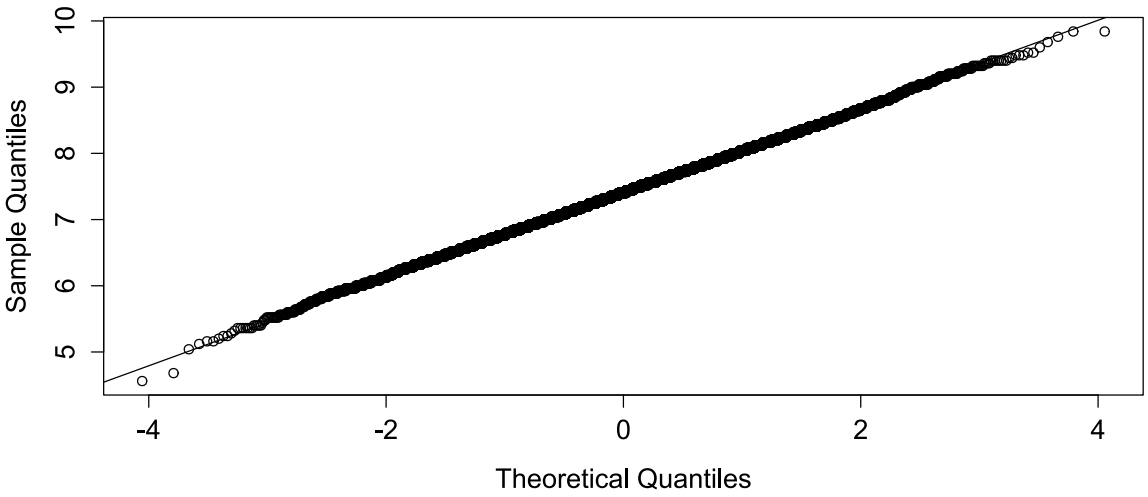

Figure A.13.: Q-Q plot of logarithm of simulated remaining cost at t = T for A1-0 (n = 20000)

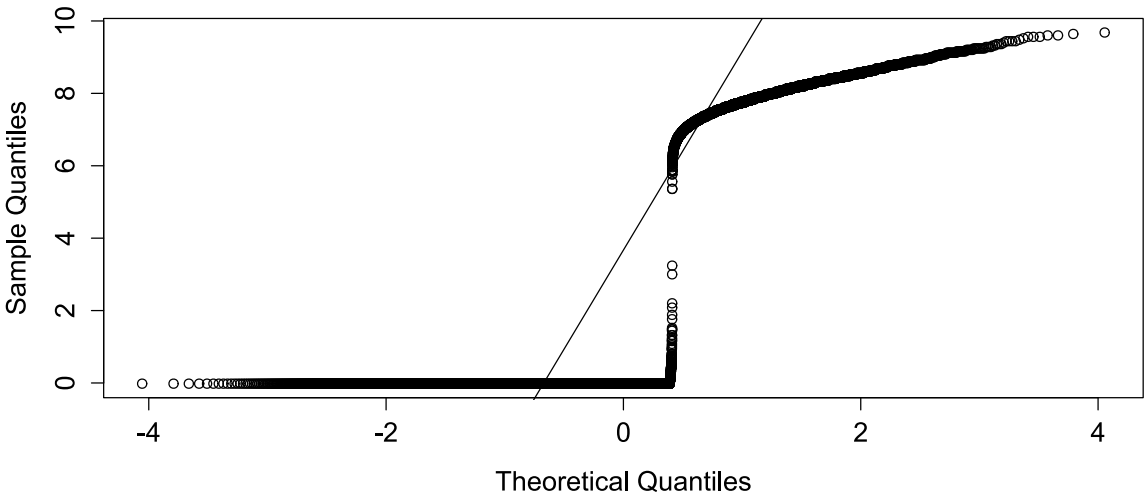

Figure A.14.: Q-Q plot of logarithm of simulated remaining cost at t = T for A1-1 (n = 20000)

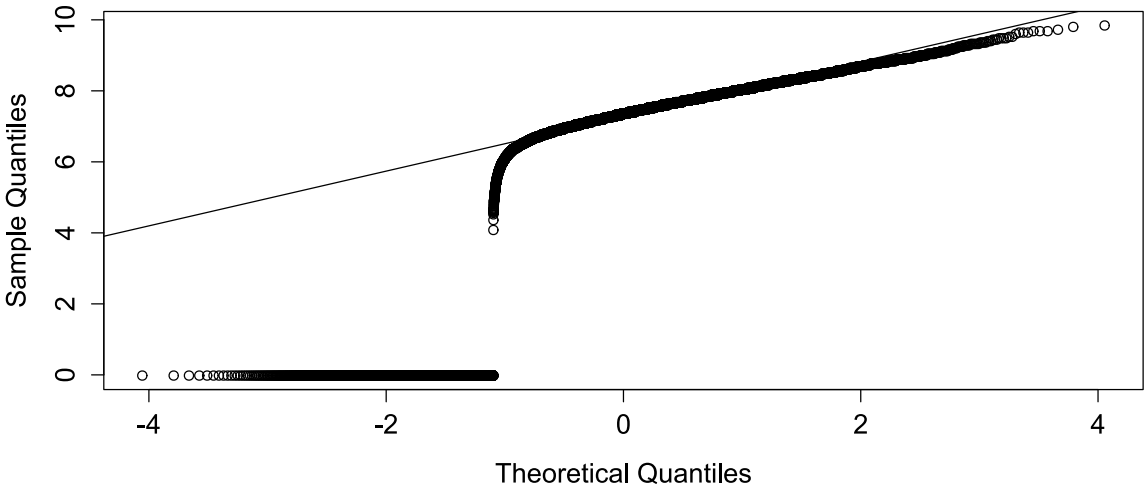

Figure A.15.: Q-Q plot of logarithm of simulated remaining cost at t = T for A1-2 (n = 20000)

A.3.3. Gamma distribution of end remaining cost

Hypothesis

- H_0 : The random variable $P(T)_x$ is $\gamma(p, b)$ distributed, with $p > 0$ and $b > 0$
- H_1 : The random variable $P(T)_x$ is <u>not</u> $\gamma(p, b)$ distributed

The Kolmogorov-Smirnov tests for $\gamma(p, b)$ result in p = $2.2e - 16$ for each and every strategy employed. Thus we refuse the assumption that the remaining cost at t = T might be gamma distributed $\rightarrow H_0$ discarded and H_1 accepted.

parameter	formula	A1-0	A1-1	A1-2
p - value	p	2.027921	0.323195	1.394980
b - value	b	0.001010	0.000360	0.000759

Table A.11.: Cash-flow parameters for log-norm distribution by strategy

Gamma distribution	formula	A1-0	A1-1	A1-2
p - value	p	2.2 e-16	2.2 e-16	2.2 e-16

Table A.12.: p - value for normal distribution of log{cash-flows} by strategy

B. Appendix - Derivation of bounding factor ψ

In chapter 5 we raised the demand that the probabilities of all four knots subsequent to the currently investigated must be real and thus satisfy B.1.

$$0 \leq p_i \leq 1 \tag{B.1}$$

We will now solve all four probabilities for they invested cash-flows I_1 and I_2 and in a second step derive a bounding factor ψ, which purpose is to limit the solution space to an area where only chances that meet B.1 can arise.

B.1. Probability p_1

Basic equation found in [2] [11.a]

$$p_1 = \frac{1}{4}\left\{1+\rho+\sqrt{\eta}\left(\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2}\right)\right\} = \frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \tag{B.2}$$

Solved for $p_1 \leq 1$

$$\begin{aligned}
&\frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 1 && |\times(4) \\
&\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 4 && |-1-\rho \\
&\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq 3-\rho && |/\sqrt{\Delta t} \\
&\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq \frac{3-\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.3}$$

Solved for $p_1 \geq 0$

$$\begin{aligned}
&\frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |\times(4) \\
&\left\{1+\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |-1-\rho \\
&\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq -1-\rho && |/\sqrt{\Delta t} \\
&\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq \frac{-1-\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.4}$$

B.2. Probability p_2

Basic equation found in [2] [11.b]

$$p_2 = \frac{1}{4}\left\{1-\rho+\sqrt{\eta}\left(\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2}\right)\right\} = \frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \tag{B.5}$$

Solved for $p_2 \leq 1$

$$\begin{aligned}
&\frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 1 && |\times(4) \\
&\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 4 && |-1+\rho \\
&\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq 3+\rho && |/\sqrt{\Delta t} \\
&\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq \frac{3+\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.6}$$

Solved for $p_2 \geq 0$

$$\begin{aligned}
&\frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |\times(4) \\
&\left\{1-\rho+\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |-1+\rho \\
&\sqrt{\Delta t}\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq -1+\rho && |/\sqrt{\Delta t} \\
&\left(-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq \frac{-1+\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.7}$$

B.3. Probability p_3

Basic equation found in [2] [11.c]

$$p_3 = \frac{1}{4}\left\{1-\rho+\sqrt{\eta}\left(-\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2}\right)\right\} = \frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \tag{B.8}$$

Solved for $p_3 \leq 1$

$$\begin{aligned}
&\frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 1 && |\times(4) \\
&\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 4 && |-1+\rho \\
&\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq 3+\rho && |/\sqrt{\Delta t} \\
&\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq \frac{3+\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.9}$$

Solved for $p_3 \geq 0$

$$\begin{aligned}
&\frac{1}{4}\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |\times(4) \\
&\left\{1-\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |-1+\rho \\
&\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq -1+\rho && |/\sqrt{\Delta t} \\
&\left(\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq \frac{-1+\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.10}$$

B.4. Probability p_4

Basic equation found in [2] [11.d]

$$p_4 = \frac{1}{4}\left\{1+\rho+\sqrt{\eta}\left(-\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2}\right)\right\} = \frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} \tag{B.11}$$

Solved for $p_4 \leq 1$

$$\begin{aligned}
&\frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 1 && |\times(4) \\
&\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \leq 4 && |-1-\rho \\
&\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq 3-\rho && |/\sqrt{\Delta t} \\
&\left(+\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \leq \frac{3-\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.12}$$

Solved for $p_4 \geq 0$

$$\begin{aligned}
&\frac{1}{4}\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |\times(4) \\
&\left\{1+\rho+\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right)\right\} && \geq 0 && |-1-\rho \\
&\sqrt{\Delta t}\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq -1-\rho && |/\sqrt{\Delta t} \\
&\left(\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right) && \geq \frac{-1-\rho}{\sqrt{\Delta t}}
\end{aligned} \tag{B.13}$$

B.5. Evaluation of extrema

Let the ratio of the technically allowable investment in a project $I_{i_{\max_t}}$ against it's estimated cost P_i be subject to the following constraint B.14.

$$0 \leq \frac{I_{1_{\max_t}}}{P_1}, \frac{I_{2_{\max_t}}}{P_2}, \ldots, \frac{I_{\max_t}}{P} \leq 1 \tag{B.14}$$

Then we will solve the inequalities given in B.1 to B.4 for the worst-case scenarios that still must satisfy the conditions. This is done by either setting $\frac{I_i}{P_i}$ to zero or one in order to achieve a maximum for the lesser-equal relations (and vice-versa a minimum for greater-equal constraints). Hence we will deduce a bounding factor ψ that further constrains the tolerable solution space and thus ensure that only non-negative probabilities smaller than one occur.

Derived from $p_i <= 1$

$$\begin{aligned}
\max\left[-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= 0 &\leq \frac{3-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_1^1 \leq \infty \\
\max\left[-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= \frac{1}{\beta_2} &\leq \frac{3+\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_2^1 \leq \frac{3+\rho}{\sqrt{\Delta t}}\beta_2 \\
\max\left[+\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= \frac{1}{\beta_1} &\leq \frac{3+\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_3^1 \leq \frac{3+\rho}{\sqrt{\Delta t}}\beta_1 \\
\max\left[+\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= \frac{\beta_2+\beta_1}{\beta_1\beta_2} &\leq \frac{3-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_4^1 \leq \frac{3-\rho}{\sqrt{\Delta t}}\frac{\beta_1\beta_2}{\beta_2+\beta_1}
\end{aligned} \tag{B.15}$$

Derived from $p_i >= 0$

$$\begin{aligned}
\min\left[-\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= -\frac{\beta_2+\beta_1}{\beta_1\beta_2} &\geq \frac{-1-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_1^0 \leq \frac{1+\rho}{\sqrt{\Delta t}}\frac{\beta_1\beta_2}{\beta_2+\beta_1} \\
\min\left[-\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= -\frac{1}{\beta_1} &\geq \frac{+1-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_2^0 \leq \frac{1-\rho}{\sqrt{\Delta t}}\beta_1 \\
\min\left[+\frac{I_1}{P_1}\frac{1}{\beta_1}-\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= -\frac{1}{\beta_2} &\geq \frac{+1-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_3^0 \leq \frac{1-\rho}{\sqrt{\Delta t}}\beta_2 \\
\min\left[+\frac{I_1}{P_1}\frac{1}{\beta_1}+\frac{I_2}{P_2}\frac{1}{\beta_2}\right] &= 0 &\geq \frac{-1-\rho}{\sqrt{\Delta t}} \rightarrow &\quad \psi_4^0 \geq -\infty
\end{aligned} \tag{B.16}$$

Conclusion

Since the coefficient of correlation ρ is restricted to values between or equal to $+1$ and -1 the inequalities given in the lower section will be determining for the bounding factor ψ. This is also due to the sum of p_1, p_2, p_3 and p_4 must always equal one and therefore the more closeness of all probabilities to zero than to one. As we don't wish to invest more than necessary we will also consider one as bounding factor and chose the minimum of the available factors. For simplicity we omit the upper section. Equation B.17 illustrates the selection of a suitable bounding factor ψ.

$$\psi = \min[1, \psi_1^0, \psi_2^0, \psi_3^0] = \min[1, \frac{1+\rho}{\sqrt{dt}}\frac{\beta_1\beta_2}{\beta_1+\beta_2}, \frac{1-\rho}{\sqrt{dt}}\beta_1, \frac{1-\rho}{\sqrt{dt}}\beta_2] \tag{B.17}$$

C. Appendix - R Code

Below the R scripts for parameter definition and the dynamic programming routines are provided. These and all program codes utilized to create simulations and figures are also available on the author's public SVN repository https://riouxsvn.com/svn/projectspublic. Interested readers might also receive the codes upon request from the author Sebastian Rötzer, BSc. (roetzer.sebastian@gmail.com) or his supervisor Prof. Dr. Dangl (thomas.dangl@tuwien.ac.at).

C.1. Definitions

R/000–definitions.R

```
#------------------------------------------------------------------------------
#stochastic projectmanagement - dynamic optimization of investment under cost-uncertainty
#------------------------------------------------------------------------------
#program 0 - definitions of global variables
#Sebastian Roetzer
#e0726857
#05-04-2013
#Vienna University of Technology

#definition of global functions
#------------------------------------------------------------------------------


## CRRA - Constant Relative Risk Aversion - Function definition

utilityFunc <- function(consumption,gamma)
{
  if(gamma == 1)
  {
    uc <- log(consumption)
  }
  else
  {
    uc <- ((consumption)^(1 - gamma) - 1)/(1 - gamma)
  }
  return(uc)
}

## CRRA - Gradient function definition

utilityGrad <- function(consumption,gamma)
{
  if(gamma == 1)
  {
    ucg <- 1/consumption
  }
  else
  {
    ucg <- (consumption)^(-gamma)
  }
  return(ucg)
}

#definition of global variables
#------------------------------------------------------------------------------

##inherent variables
#------------------------------------------------------------------------------

#initial price estimation from which the geometric random walk is unfolded
priceGuess0 <- c(1000,1000)
```

```
53 #reward is distributed as the margin of remaining cost is reached
   rewardMargin <- c(10,10)#,1)#0.05 * priceGuess0
55
   #relation of rewardMargin to the initial priceGuess0
57 lambda <- rewardMargin/priceGuess0

59 #allow the a left shift of the whole cost-grid -> a bandwidth of initial price guesses will be
       calculated
   shift <- 0
61 shift <- max(0,shift)

63 #volatility (per time-unit) of the examined project
   beta <- c(0.20,0.20)
65
   #discount rate
67 r <- 0.05

69 #utility discount rate  (intertemporal elasticity of substitution)
   kappa <- 0.05
71
   #maximum available time for completion
73 Tmax <- 10

75 #reward for project completion
   reward <- c(5000,6000)
77
   #risk aversion parameter for the utility function
79 gamma <- 1

81
   ##arbitrary variables
83 #-------------------------------------------------------------------------------------------

85 #desired time interval for the computational grid
   dt <- 1/20#1/100
87
   #maximal feasible time resolution in order to reach the reward margin within Tmax (dt < dtmax is
       preferable)
89 dtMax <- (-Tmax*beta/log(rewardMargin/priceGuess0,))^2

91 #shoud dt exceed dtMax -> coerce to dtMax, send a warning message
   if(dt > min(dtMax))
93 {
     dt <- min(dtMax)
95   print (paste("WARNING! dt coerced to:",dt,sep=""))
   }
97
   #resolution of the computational grids time-axis
99 tCount <- Tmax/dt

101
   ##derived variables
103 #-------------------------------------------------------------------------------------------

105 #preparation of the u(p-) and d(own-ward) movements in the geometric random walk
   #the standard dev of p becomes beta whilst the expected value stays E[p(t+1)] = p(t)
107 u <- exp(beta * sqrt(dt))
   d <- 1/u
109

111 utilReward <- utilityFunc(reward/priceGuess0,gamma)
```

C.2. 1-dimensional dynamic programming

R/101–optimization–of–investment–behavior.R

```
1 #-------------------------------------------------------------------------------------------
   #stochastic projectmanagement - dynamic optimization of investment under cost-uncertainty
3 #-------------------------------------------------------------------------------------------
   #program 1 - dynamic optimization of investment in a single project
5 #
   #Sebastian Roetzer
7 #e0726857
   #03-01-2013
9 #Vienna University of Technology

11
   #definition of local functions
13 #-------------------------------------------------------------------------------------------

15 ## pi Probability

17 pi1DFunc <- function(I,pF,dt)
   {
19   return(max(0,(1 - d - (I/pF) * dt)/(u - d)))
```

```
}


## function of investment utility

invest1DFunc <- function(I,ImaxF,ti,fo,pF)
{
  uVFunc <- utilityFunc((ImaxF - I)/sc,gamma) * dt  +
            1/(1 + kappa)^dt * (((1 - d[1] - (I/pF) * dt)/(u[1] - d[1])) * V[pOffset+fo+1,tOffset+ti
                +1] +
            (1 - ((1 - d[1] - (I/pF) * dt)/(u[1] - d[1]))) * V[pOffset+fo-1,tOffset+ti+1])

  return(uVFunc)
}

## gradient of investment utility

invest1DGrad <- function(I,ImaxF,ti,fo,pF)
{
  uVGrad <- utilityGrad((ImaxF - I)/sc,gamma) * dt * (-1/sc) +
            1/(1 + kappa)^dt * 1/(u[1] - d[1]) * (dt/pF) * (V[pOffset+fo-1,tOffset+ti+1] -
                                                            V[pOffset+fo+1,tOffset+ti+1])

  return(uVGrad)
}

## gradient inner solution

invest1DSolInt <- function(ImaxF,ti,fo,pF)
{
  I <- ImaxF - ((sc)^(1-gamma) * (1/(1 + kappa)^dt) * (1/(u[1] - d[1])) * (1/pF) *
               ((V[pOffset+fo-1,tOffset+ti+1] - V[pOffset+fo+1,tOffset+ti+1])))^(-1/gamma)
  I <- max(0,I)

  return(I)
}


#definition of local variables
#-------------------------------------------------------------------------------

#strategy case selector
strategy <- 3
#0: no investment activity
#1: constant relative investment activity
#2: constant absolute investment activity
#3: use CRRA utility function and gradient based optimization

#boolean strategy selector variable (affected by strategy 3)
useOptim       <- FALSE

switch(strategy+1,
       {investment   <- 0},
       {investQprice <- 1},#tolerable maximum is (1-d)/dt), larger iQp has no effect
       {investment   <- priceGuess0[1]/Tmax},
       {useOptim     <- TRUE}
        )

#calculation of probability pi for the waiting case
piWait <- (1 - d[1])/(u[1] - d[1])

#calculate the maximum tolerable investment
ImaxF <- reward[1]/(1 + r)^Tmax

#set the scaling factor for the CRRA utility function of investment
sc            <- ImaxF

#the utility of unspent monetary units in a timestep (i.e. consumation)
utilWait      <- utilityFunc(ImaxF/sc,gamma) * dt # = 0


#necessary offset for the matrix calculation (set the price pointer to the grids centre)
pOffset <- tCount+shift+1
tOffset <- 1

#create matrices for the value function [V] and suggested behavior [a]
#also create matrices for the up-move probability [p] and accumulated success probability [P]
V   <- matrix(NA,nrow=2*(tCount+shift)+1,ncol=tCount+1)
a   <- matrix(NA,nrow=2*(tCount+shift)+1,ncol=tCount+1)
pi  <- matrix(NA,nrow=2*(tCount+shift)+1,ncol=tCount+1)
PI  <- matrix(NA,nrow=2*(tCount+shift)+1,ncol=tCount+1)

#fill the last column as preparation for dynamic programming
#as the time-border is reached without passing the reward margin
#the value function becomes 0; in case of strategy 3 the available cash is consumed
V[,tOffset+tCount]  <- switch(1 + useOptim,0,utilWait)
a[,tOffset+tCount]  <- 0
pi[,tOffset+tCount] <- 0
```

```
109 PI[,tOffset+tCount] <- 0

111 #initialize with zeor errors
    error <- 0
113
    #dynamic optimization of investment behavior
115 #-------------------------------------------------------------------------------

117 #backward induction from Tmax to t = 0
    for(t in (tCount-1):(0))
119 {
      #display the time-step currently under calculation
121   print (t*dt)

123   #number of different price-forks in the current time-step
      forks <- seq(from = -(t+shift), to = (t+shift), by = 2)
125
      #compute the values for all price-forks
127   for(f in forks)
      {
129     #calculate the estimated remaining project cost in the fork
        priceFork <- priceGuess0[1] * u[1]^f
131
        #should the reward margin exceed the remaining estimated cost set the value function to reward
133     if(priceFork <= rewardMargin[1])
        {
135       if(useOptim == FALSE)
          {
137         V[pOffset+f,tOffset+t] <- reward[1]
          }
139       else
          {
141         V[pOffset+f,tOffset+t] <- utilWait + utilReward[1]
          }
143
          a[pOffset+f,tOffset+t]  <- 0
145       pi[pOffset+f,tOffset+t] <- 0
          PI[pOffset+f,tOffset+t] <- 1
147     }
        #if no reward is distributed to as follows
149     else
        {
151       if(useOptim == FALSE)
          {
153         #dynamic programming using policy guided decisions

155         #compute the value of waiting for the next fork to unfold without any investment activity
            waitingValue <- 0 + 1/(1+r)^dt * (piWait * V[pOffset+f+1,tOffset+t+1] +
157                                             (1-piWait) * V[pOffset+f-1,tOffset+t+1])

159         #chose depending on strategy selector
            switch(strategy+1,
161         {},
            {investment   <- priceFork*investQprice},
163         {},
            {})
165
            #due to investment activity a new probability pi has to be computed
167         piAct <- pi1DFunc(investment,priceFork,dt)

169         #compute the value of this fork with respect to the investment activity undertaken
            actingValue <- -investment * dt + 1/(1+r)^dt * (piAct * V[pOffset+f+1,tOffset+t+1] +
171                                                           (1-piAct) * V[pOffset+f-1,tOffset+t+1])

173         #determine whether to wait or to invest in this node
            if(waitingValue > actingValue)
175         { #set the value function to the value of waiting, plot a grey coloured line to the overlying
                node
              V[pOffset+f,tOffset+t]  <- waitingValue
177           a[pOffset+f,tOffset+t]  <- 0
              pi[pOffset+f,tOffset+t] <- piWait
179           PI[pOffset+f,tOffset+t] <- piWait * PI[pOffset+f+1,tOffset+t+1] +
                                         (1-piWait) * PI[pOffset+f-1,tOffset+t+1]
181         }
            else
183         { #set the value function to th value of investing, plot a blue coloured line to the
                overlying node
              V[pOffset+f,tOffset+t]  <- actingValue
185           a[pOffset+f,tOffset+t]  <- -investment * dt
              pi[pOffset+f,tOffset+t] <- piAct
187           PI[pOffset+f,tOffset+t] <- piAct * PI[pOffset+f+1,tOffset+t+1] +
                                         (1-piAct) * PI[pOffset+f-1,tOffset+t+1]
189         }
          }
191       else
          {
193         #dynamic programming using an utility function and gradient based optimization

195         #calculate ImaxT
            ImaxT <- priceFork * ((1-d[1])/dt)
```

```
197
          Imax  <- min(ImaxF,ImaxT)
199
          #if(invest1DGrad(0,ImaxF,t,f,priceFork)>0){print(paste("+0 :",f,sep=""))}
201       #if(invest1DGrad(Imax,ImaxF,t,f,priceFork)>0){print(paste("+max :",f,sep=""))}

203       gr0 <- invest1DGrad(0,ImaxF,t,f,priceFork)
          grm <- invest1DGrad(Imax,ImaxF,t,f,priceFork)
205
          if(gr0 < 0)
207       {
            #print("wait")
209
            #compute the value of waiting for the next fork to unfold without any investment activity
211         waitingValue <- utilWait + 1/(1+kappa)^dt * (piWait * V[pOffset+f+1,tOffset+t+1] +
                                                         (1-piWait) * V[pOffset+f-1,tOffset+t+1])
213
            V[pOffset+f,tOffset+t]  <- waitingValue
215         a[pOffset+f,tOffset+t]  <- 0
            pi[pOffset+f,tOffset+t] <- piWait
217         PI[pOffset+f,tOffset+t] <- piWait * PI[pOffset+f+1,tOffset+t+1] +
                                       (1-piWait) * PI[pOffset+f-1,tOffset+t+1]
219       }
          else
221       {
            if(grm > 0)
223         {
              #print("invest_max")
225
              piAct <- pi1DFunc(Imax,priceFork,dt)
227
              V[pOffset+f,tOffset+t]  <-  invest1DFunc(Imax,ImaxF,t,f,priceFork)
229           a[pOffset+f,tOffset+t]  <-  -Imax * dt
              pi[pOffset+f,tOffset+t] <-  piAct
231           PI[pOffset+f,tOffset+t] <-  piAct * PI[pOffset+f+1,tOffset+t+1] +
                                          (1-piAct) * PI[pOffset+f-1,tOffset+t+1]
233         }
            else
235         {
              #print("invest_partial")
237
              investment <- invest1DSolInt(ImaxF,t,f,priceFork)
239
              actingValue <- uniroot(invest1DGrad,interval = c(0,Imax),ImaxF = ImaxF,ti = t,fo = f,pF =
                   priceFork)
241           investmentAlt <- actingValue$root

243           if(abs(investment - investmentAlt)>1e-03)
              {
245             print("ERROR")
                error <- error + 1
247           }

249           piAct <- pi1DFunc(investment,priceFork,dt)


251
              V[pOffset+f,tOffset+t]  <-  invest1DFunc(investment,ImaxF,t,f,priceFork)
253           a[pOffset+f,tOffset+t]  <-  +investment * dt
              pi[pOffset+f,tOffset+t] <-  piAct
255           PI[pOffset+f,tOffset+t] <-  piAct * PI[pOffset+f+1,tOffset+t+1] +
                                          (1-piAct) * PI[pOffset+f-1,tOffset+t+1]
257         }
          }
259
        }
261


263   }
    }
265 }

267
    print(paste("V.0 = ",max(V[,1],na.rm=TRUE),sep=""))
269
    print(paste("PI.0 = ",max(PI[,1],na.rm=TRUE),sep=""))
271
    error
```

C.3. 2-dimensional dynamic programming

R/211–portfolio–optim.R

```
#-------------------------------------------------------------------------------
2 #stochastic projectmanagement - dynamic optimization of investment under cost-uncertainty
  #-------------------------------------------------------------------------------
```

```
#program 1 - dynamic optimization of investment in a single project
#
#Sebastian Roetzer
#e0726857
#03-01-2013
#Vienna University of Technology

#definitions of local variables
#-------------------------------------------------------------------------------------------

#library(rgl)

#FALSE to skip diagram title (for export), TRUE to display it
displayTitle  <- FALSE

#reward is distributed as the margin of remaining cost is reached
#rewardMargin <- priceGuess0 * 0.1

#discount rate
#r <- 0.05

#volatility (per time-unit) of the examined project
#beta <- c(0.20,0.20)

rho <- 0.5

#maximum available time for completion
#T <- 5

#desired time interval for the computational grid
#dt <- 1/40#100

#maximal feasible time resolution in order to reach the reward margin within T (dt < dtmax is preferable)
#dtMax <- (-T*beta/log(rewardMargin/priceGuess0,))^2

#shoud dt exceed dtMax -> coerce to dtMax, send a warning message
#if(dt>dtMax)
#{
#  dt <- dtMax
#  print(paste("WARNING! dt coerced to:",dt,sep=""))
#}

#resolution of the computational grids time-axis
#tCount <- T/dt

#preparation of the u(p-) and d(own-ward) movements in the geometric random walk
#the standard dev of p becomes beta whilst the expected value stays E[p(t+1)] = p(t)
#u <- exp(beta * sqrt(dt))
#d <- 1/u

#bounding factor psi
psi <- min(1,((1 + rho)/sqrt(dt))*((beta[1]*beta[2])/(beta[1] + beta[2])),beta[1]*((1 - rho)/sqrt(dt)),beta[2]*((1 - rho)/sqrt(dt)))

#ImaxF
ImaxF <- min(reward/(1 + r)^Tmax)

sc <- ImaxF

utilWait      <- utilityFunc(ImaxF/sc,gamma) * dt # = 0

#####insert######
#strategy <- 1
#calculate the quotient of investment against remaining price that ensures a down-move (i.e. remaining price
#is reduced); the possibility to pay more than estimated is neglected, thus for dt->0 a down-move cannot be assured
#investQprice <- min(1,(1-d)/dt)


#allow the a left shift of the whole cost-grid -> a bandwidth of initial price guesses will be calculated
#shift <- 0
#shift <- max(0,shift)

#necessary offset for the matrix calculation (set the price pointer to the grids centre)
pOffset <- tCount+shift+1
tOffset <- 1

fOffset <- 2

#create matrices for the value function [V] and suggested behavior [a]
V <- array(c(NA,NA,NA),c(tCount+1,2*(tCount+shift)+1,2*(tCount+shift)+1))

```

```
a <- array(c(NA,NA,NA,NA),c(tCount+1,2*(tCount+shift)+1,2*(tCount+shift)+1,2))
#also create matrices for the up-move probability [pi] and accumulated success probability [PI]
pi <- array(c(NA,NA,NA,NA),c(tCount+1,2*(tCount+shift)+1,2*(tCount+shift)+1,4))

PI <- array(c(NA,NA,NA,NA),c(tCount+1,2*(tCount+shift)+1,2*(tCount+shift)+1,3))

#break

#fill the last column -> as the time-border is reached without passing the reward margin the value
    function becomes 0
V[tOffset+tCount,,] <- utilWait
a[tOffset+tCount,,,] <- c(0,0)
pi[tOffset+tCount,,,] <- c(0,0,0,0)
PI[tOffset+tCount,,,] <- c(0,0,0)


error <- FALSE
#d <- array(c(NA,NA,NA,NA),c(2,2,1,2))
#d[1,1,1,] <- c(1,2)

## pi Probability function definition
#-------------------------------------------------------------------------------------------------

pi2DFunc <- function(I,pF,beta,rho)
{
  prob <- c(0,0,0,0)
  prob[1] <- 0.25 * (1 + rho + sqrt(dt) * ((-I[1]/(pF[1]*beta[1]))+(-I[2]/(pF[2]*beta[2]))))
  prob[2] <- 0.25 * (1 - rho + sqrt(dt) * ((-I[1]/(pF[1]*beta[1]))-(-I[2]/(pF[2]*beta[2]))))
  prob[3] <- 0.25 * (1 - rho + sqrt(dt) * ((+I[1]/(pF[1]*beta[1]))+(-I[2]/(pF[2]*beta[2]))))
  prob[4] <- 0.25 * (1 + rho + sqrt(dt) * ((+I[1]/(pF[1]*beta[1]))+(+I[2]/(pF[2]*beta[2]))))

  #if(sum(prob) != 1)
   # {print(paste("ERROR, pi = ",prob," sum = ",sum(prob),sep = ""))
    #}
  if(max(prob) > 1){print("ERROR")}

  return(prob)
}

#pi2DFunc <- function(I,pF,beta,rho)
piWait <- pi2DFunc(c(0,0),c(1,1),beta,rho)

## investment function definition - dynamic programming of utility function
#-------------------------------------------------------------------------------------------------

invest2DFunc <- function(I,ImaxF,sc,ti,fo,br,pF)
{
  pr <- pi2DFunc(I,pF,beta,rho)

  UValue <- utilityFunc((ImaxF - I[1] - I[2])/sc,gamma)*dt + 1/(1 + kappa)^dt * (pr[1] * V[tOffset+ti
      +1,pOffset+fo+1,pOffset+br+1] + pr[2] * V[tOffset+ti+1,pOffset+fo+1,pOffset+br-1] + pr[3] * V[
      tOffset+ti+1,pOffset+fo-1,pOffset+br+1] + pr[4] * V[tOffset+ti+1,pOffset+fo-1,pOffset+br-1])

  return(UValue)
}


## GRADIENT
#-------------------------------------------------------------------------------------------------

invest2DGrad <- function(I,ImaxF,sc,br,pF,rho,dt,beta,gamma,kappa,Vold,fOffset,pOffset,utilityFunc,
    utilityGrad)
{
  UIgrad <- c(0,0)

  UIgrad[1] <- utilityGrad((ImaxF - I[1] - I[2])/sc,gamma) * dt * (-1/sc) +
    1/(1 + kappa)^dt * (0.25/(pF[1]*beta[1])) * sqrt(dt) *(-Vold[fOffset+1,pOffset+br+1] - Vold[
        fOffset+1,pOffset+br-1] + Vold[fOffset-1,pOffset+br+1] + Vold[fOffset-1,pOffset+br-1])

  UIgrad[2] <- utilityGrad((ImaxF - I[1] - I[2])/sc,gamma) * dt * (-1/sc) +
    1/(1 + kappa)^dt * (0.25/(pF[2]*beta[2])) * sqrt(dt) *(-Vold[fOffset+1,pOffset+br+1] + Vold[
        fOffset+1,pOffset+br-1] - Vold[fOffset-1,pOffset+br+1] + Vold[fOffset-1,pOffset+br-1])

  return(UIgrad)
}

## GRADIENT inner solution
#-------------------------------------------------------------------------------------------------

invest2DIntSol <- function(I,ImaxF,sc,br,pF,rho,dt,beta,gamma,kappa,Vold,fOffset,pOff,utilityFunc,
    utilityGrad)
{
  Inner <- c(0,0)

  Inner[1] <- ImaxF - I[2] - sc * ((sc/(1 + kappa)^dt) * (0.25/(pF[1]*beta[1])) * (1/sqrt(dt)) *(-
      Vold[fOffset+1,pOffset+br+1] - Vold[fOffset+1,pOffset+br-1] + Vold[fOffset-1,pOffset+br+1] +
      Vold[fOffset-1,pOffset+br-1]))^(-1/gamma)

  Inner[2] <- ImaxF - I[1] - sc * ((sc/(1 + kappa)^dt) * (0.25/(pF[2]*beta[2])) * (1/sqrt(dt)) *(-
      Vold[fOffset+1,pOffset+br+1] + Vold[fOffset+1,pOffset+br-1] - Vold[fOffset-1,pOffset+br+1] +
```

```
        Vold[fOffset-1,pOffset+br-1]))^(-1/gamma)

170   return(Inner)
}
172


174
## Five-point optimization
176 #-----------------------------------------------------------------------------------------

178 optimMultiDim <- function(ImaxT,ImaxF,sc,br,pF,rho,dt,beta,gamma,Vold,fOffset,pOffset,utilityFunc,
      utilityGrad,invest2Dgrad,invest2DIntSol)
{
180   gradDim <- length(ImaxT)
    #gradDim
182   blockIndex <- NULL
    #blockIndex
184   I <- rep(0,gradDim)
    #I
186   gIndex <- NULL
    gStart <- NULL
188

190   for(i in 1:gradDim)
    {
192     gStart <-invest2DGrad(I,ImaxF,sc,br,pF,rho,dt,beta,gamma,kappa,Vold,fOffset,pOffset,utilityFunc,
          utilityGrad)

194     #print(gStart)

196     if(length(blockIndex) != 0)
      {
198       #gStmp <- gStart[-blockIndex]
        gStart[blockIndex] <- -Inf
200       #ImTtmp <- ImaxT[-blockIndex]
        ImaxT[blockIndex] <- 0
202     }else
      {
204       #gStmp <- gStart
        #ImTtmp <- ImaxT
206     }
      #gStmp
208     #ImTtmp
      #gStart
210     #ImaxT

212     #gIndex <- which(gStmp==max(gStmp))
      gIndex <- which(gStart == max(gStart))
214
      if(length(gIndex) > 1)
216   {gIndex <- gIndex[1+round(runif(1)*(length(gIndex)-1),0)]}
      gIndex
218
      #print(gStart[gIndex])
220   if(gStart[gIndex] <= 0)
      {
222     #gStart
        return(I)
224     #break
      }else
226   {
        Itmp <- I
228     Itmp[gIndex] <- min(ImaxT[gIndex],ImaxF - sum(I))

230     gEnd <- invest2DGrad(Itmp,ImaxF,sc,br,pF,rho,dt,beta,gamma,kappa,Vold,fOffset,pOffset,
          utilityFunc,utilityGrad)

232     #gEtmp <- gEnd[-blockIndex]

234     if(gEnd[gIndex] > 0)
        {
236       #print("full invest")

238       I[gIndex] <- Itmp[gIndex]

240       blockIndex <- c(blockIndex,gIndex)

242       #ImaxT <- ImaxT - sum(I)
        }
244     else
        {
246       #print("partial invest")

248       Itmp <- invest2DIntSol(I,ImaxF,sc,br,pF,rho,dt,beta,gamma,kappa,Vold,fOffset,pOffset,
            utilityFunc,utilityGrad)

250       I[gIndex] <- Itmp[gIndex]

252       blockIndex <- c(blockIndex,gIndex)
```

```
254           break
255         }
256       }
257     }
258     #I
259     return(I)
260 }
261
262
263 #dynamic optimization of investment behavior
264 #-------------------------------------------------------------------------------
265
266 #backward induction from T to t = 0
267 for(t in (tCount-1):(0))
268 {
269   #display the time-step currently under calculation
270   #print (t*dt)
271
272   #number of different price-forks in the current time-step
273   forks <- seq(from = -(t+shift), to = (t+shift), by = 2)
274   branches <- forks
275   #determine whether to plot a line in the solution graph or not (based on t, T and dt)
276   #plotEnable <- round(t/plotDist) == t/plotDist || t == (tCount-1)
277
278   #print(paste(forks,branches,sep=""))
279   if(error == TRUE){break}
280
281   #compute the values for all price-forks
282   for(f in forks)
283   {
284     #calculate the estimated remaining project cost in the fork
285     priceFork <- priceGuess0[1]*u[1]^f
286
287     print(paste("(",t*dt,"|",f,")",sep=""))
288
289     if(error == TRUE){break}
290
291     Vold <- V[tOffset+t+1,(pOffset+f-1):(pOffset+f+1),]
292
293     for(b in branches)
294     {
295       priceBranch <- priceGuess0[2]*u[2]^b
296
297       #print(paste("(",t*dt," | ",f," | ",b,")",sep=""))
298
299       pF <- c(priceFork,priceBranch)
300
301
302       #should the reward margin exceed the remaining estimated cost set the value function to reward
303       if(priceFork>rewardMargin[1] && priceBranch>rewardMargin[2])
304       {
305         ImaxT <- pF*psi
306
307         investment <- optimMultiDim(ImaxT,ImaxF,sc,br=b,pF=pF,rho,dt,beta,gamma,Vold,fOffset,pOffset,
308               utilityFunc,utilityGrad,invest2DGrad,invest2DIntSol)
309
310         V[tOffset+t,pOffset+f,pOffset+b] <- invest2DFunc(investment,ImaxF,sc,ti=t,fo=f,br=b,pF)
311
312         a[tOffset+t,pOffset+f,pOffset+b,] <- -investment*dt
313
314         pi[tOffset+t,pOffset+f,pOffset+b,] <- pi2DFunc(investment,pF,beta,rho)
315
316
317         PI[tOffset+t,pOffset+f,pOffset+b,1] <- pi[tOffset+t,pOffset+f,pOffset+b,1] * PI[tOffset+t+1,
318               pOffset+f+1,pOffset+b+1,1] + pi[tOffset+t,pOffset+f,pOffset+b,2] * PI[tOffset+t+1,pOffset
319               +f+1,pOffset+b-1,1] + pi[tOffset+t,pOffset+f,pOffset+b,3] * PI[tOffset+t+1,pOffset+f-1,
320               pOffset+b+1,1] + pi[tOffset+t,pOffset+f,pOffset+b,4] * PI[tOffset+t+1,pOffset+f-1,pOffset
321               +b-1,1]
322         PI[tOffset+t,pOffset+f,pOffset+b,2] <- pi[tOffset+t,pOffset+f,pOffset+b,1] * PI[tOffset+t+1,
323               pOffset+f+1,pOffset+b+1,2] + pi[tOffset+t,pOffset+f,pOffset+b,2] * PI[tOffset+t+1,pOffset
324               +f+1,pOffset+b-1,2] + pi[tOffset+t,pOffset+f,pOffset+b,3] * PI[tOffset+t+1,pOffset+f-1,
325               pOffset+b+1,2] + pi[tOffset+t,pOffset+f,pOffset+b,4] * PI[tOffset+t+1,pOffset+f-1,pOffset
326               +b-1,2]
327         PI[tOffset+t,pOffset+f,pOffset+b,3] <- pi[tOffset+t,pOffset+f,pOffset+b,1] * PI[tOffset+t+1,
328               pOffset+f+1,pOffset+b+1,3] + pi[tOffset+t,pOffset+f,pOffset+b,2] * PI[tOffset+t+1,pOffset
329               +f+1,pOffset+b-1,3] + pi[tOffset+t,pOffset+f,pOffset+b,3] * PI[tOffset+t+1,pOffset+f-1,
330               pOffset+b+1,3] + pi[tOffset+t,pOffset+f,pOffset+b,4] * PI[tOffset+t+1,pOffset+f-1,pOffset
331               +b-1,3]
332
333         if(abs(V[tOffset+t,pOffset+f,pOffset+b]) > max(abs(V[tOffset+t+1,pOffset+f+1,pOffset+b+1]),
```

```
                                                                    abs(V[tOffset+t+1,pOffset+f+1,pOffset+b-1]),
332                                                                 abs(V[tOffset+t+1,pOffset+f-1,pOffset+b+1]),
                                                                    abs(V[tOffset+t+1,pOffset+f-1,pOffset+b-1])))
334           {
                error <- TRUE
336             print("ERROR detected")
                print(paste("t = ",t," f = ",f," b = ",b,sep=""))
338             print(paste("V(t) = ",V[tOffset+t,pOffset+f,pOffset+b],sep=""))
                print(paste("V(t,f+,b+) = ",V[tOffset+t+1,pOffset+f+1,pOffset+b+1],sep=""))
340             print(paste("V(t,f+,b-) = ",V[tOffset+t+1,pOffset+f+1,pOffset+b-1],sep=""))
                print(paste("V(t,f-,b+) = ",V[tOffset+t+1,pOffset+f-1,pOffset+b+1],sep=""))
342             print(paste("V(t,f-,b+) = ",V[tOffset+t+1,pOffset+f-1,pOffset+b-1],sep=""))

344             print(paste("invest = ",investment,sep=""))
                print(paste("piAct = ",pi2DFunc(investment,pF,beta,rho),sep=""))
346             print(paste("piWait = ",piWait,sep=""))

348             break
              }
350
            }
352         #if no Bellman equation needs to be solved, the reward margin is reached -> distribute a
                renumeration
            else
354         {
              payout <- utilWait
356
              PI[tOffset+t,pOffset+f,pOffset+b,1] <- 0
358
              PI[tOffset+t,pOffset+f,pOffset+b,2] <- 0
360
              if(priceFork <= rewardMargin[1])
362           {
                payout <- payout + utilReward[1]
364
                PI[tOffset+t,pOffset+f,pOffset+b,1] <- 1
366           }

368           if(priceBranch <= rewardMargin[2])
              {
370             payout <- payout + utilReward[2]

372             PI[tOffset+t,pOffset+f,pOffset+b,2] <- 1
              }
374

376           V[tOffset+t,pOffset+f,pOffset+b] <- payout

378           a[tOffset+t,pOffset+f,pOffset+b,] <- c(0,0)

380           pi[tOffset+t,pOffset+f,pOffset+b,] <- c(0,0,0,0)

382           PI[tOffset+t,pOffset+f,pOffset+b,3] <- 1

384         }
          }
386


388     }
    }
390

392 print(paste("V.0 = ",max(V[1,,],na.rm=TRUE),sep=""))

394 print(paste("PI.0 = ",max(PI[1,,,3],na.rm=TRUE)," (",max(PI[1,,,1],na.rm=TRUE),", ",max(PI[1,,,2],na.
        rm=TRUE),")",sep=""))
```

Bibliography

[1] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[2] Phelim P. Boyle, Jeremy Evnine, and Stephen Gibbs. Numerical evaluation of multivatiate contingent claims. *The Review of Financial Studies*, Vol. 2(Nr. 2):pp. 241 – 250, 1989.

[3] John C. Cox, Stephen Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, Vol. 7:pp. 229 – 263, 1979.

[4] Morris H. DeGroot and Mark j. Schervish. *Probability and Statistics*. Pearson, 2011.

[5] Avinash K. Dixit and Robert S. Pindyck. *Investment Under Uncertainty*. Princeton University Press, 1994.

[6] Shane Frederick, George Loewenstein, and Ted O'Donoghue. Time discounting and time preference: A critical review. *Journal of Economic Literature*, Vol. 40:pp. 351 – 401, 2002.

[7] Paolo Guasoni and Walter Schachermayer. Necessary conditions of the existence of utility maximizing strategies under transaction costs. Working paper, Boston University, University of Pisa and Vienna University of Technology, 2004.

[8] Ken-Ichi Inada. On a two-sector model of economic growth. *The Review of Economic Studies*, Vol. 30(Nr. 2):pp. 119 – 127, 1963.

[9] Jonathan E. Ingersoll. *Theory of Financial Decision Making*. Rowman and Littlefield Pubslishers, 1987.

[10] Lars Ljungqvist. *Theory of Financial Decision Making*. Massachusetts Institute of technology, 2000.

[11] Robert S. Pindyck. Investments of uncertain cost. *Journal of Financial Economics*, Vol. 34(Nr. 1):pp. 53 – 76, 1993.

[12] Sam S. Shapiro and Martin Bradbury Wilk. An analysis of variance test for normality. *Biometrika*, 1965.

[13] Richard M. H. Suen. Bounding the crra utility functions. Mpra paper, University Library of Munich, Germany, 2009.

[14] Rangarajan K. Sundaram. *A First Course In Optimization Theory*. Cambridge University Press, 1996.