Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

Households Portfolio Choice Problem

Author:
Matteo Mario Di Venti

Supervisor:
Dr Harjoat S. Bhamra

Submitted in partial fulfillment of the requirements for the BSc degree in Mathematics with Statistics for Finance of Imperial College London

June 2021

Abstract

Portfolio choice problems are the basis of modern asset pricing theory. This paper develops and solves an intertemporal household portfolio problem with constant wage income and stochastic market volatility given by the Heston model. The advanced mathematical methods required to describe the problem in a stochastic control framework are introduced together with the stochastic calculus machinery later required to derive the associated PDE. Feynman-Kac theorem and a finite difference scheme are finally used to solve for the optimal controls. The numerical method is then implemented in Python to yield the portfolio solution. Coded examples are also presented in chapter 2 and 3 to exemplify numerical methods to solve Black-Scholes PDE and calibrate the control problem parameters.

Contents

Chapter 1

Introduction

Finance as discipline is the study of money and investments. A fertile and important branch of the subject is dedicated to the study of the systems of asset allocation that can yield the best possible outcome for an investor.
The fundamental question pertaining how to allocate limited personal resources to maximise a certain definition of utility or satisfaction through consumption is a problem that everyone faces daily in their lives. Time, food, sleep or energies expenditure during the day are all types of ubiquitous decision making that everyone faces and tries to optimize for their general well-being. Furthermore, they are all examples of problems that do not arise and resolve in a limited time frame but carry on during our entire lives. On a broader scale, it could be argued that intertermporal asset management is part of those activities that define the life of any conscious human being, whether through career planning, raising a family or granting their own future welfare.

Of utmost importance would therefore be the possibility of modeling the problem in a framework that grants consistent, rational and optimal solutions. This is the problem that the Nobel prize Robert Merton decided to face in two articles published in 1969 and 1971 that introduced his celebrated Portfolio Choice solution [1][2]. The problem arose in trying to consider the best possible investment in continuous time rather than just a number of discrete dates like the mathematician Frank Ramsey did in a pioneering 1920 paper [3]. The investor seeks to maximise through its wealth consumption and proportion of investment in a risky (i.e not yielding deterministic returns) portfolio the expected value of the utility of the consumption. Surprisingly, through the powerful framework of stochastic control and Bellman optimal principle a closed form optimal solution can be found for Merton's formulation. The original paper revealed to be very seminal and defined the field of inter-temporal asset pricing.

Various declinations of Merton's problem have been the object of study in finance and economy. Introducing different utility functions, bequest values or different cash flows the framework can be used to represent more complex and diverse settings. Constantinides [4] considers time varying utility function in which agents get "used" to their utility and Campbell and Cochrane ([5]) expand this idea to take

into account utility changing with the utility of other agents in a "Keeping up with the Joneses" fashion. Obstfeld [6] develops a recursive utility function for which the utility of the household depends at any point in time on the expected value of future utilities. Bhamra and Uppal [7] extend this model to stochastic investment opportunities and demonstrate that in this case the allocation to the portfolio can depend on both risk aversion and the elasticity of intertemporal substitution.

The goal of this thesis will be to develop the mathematical tools and framework necessary to solve portfolio choice problems. In particular, a portfolio choice problem will be solved similar to the one of [7] with stochastic investment opportunities but under the simplifying assumption that elasticity of inter-temporal rate ϵ is equal to $\frac{1}{1-\gamma}$ thus returning a Constant Relative Risk Aversion (CRRA) utility function. To enhance the similarity to reality and make the model more interesting a source of stable income Y will be considered in the household budget. The resulting non-linear PDE will be solved through a numerical method to yield the optimal ratios of consumption and risk investment for an household.

The paper is organised as follows. Finite difference schemes (FTCS, BTCS and Crank-Nicholson) methods are introduced in Chapter 2. To exemplify their utility in finance the BTCS scheme is used to solve Black-Scholes PDE for the value of a European call option (code in Appendix A). Chapter 3 deals with the fundamental stochastic calculus machinery to correctly model and operate on stochastic processes. Ito lemma and Girsanov theorem are proved for future use in Chapter 5, Feynman-Kac theorem is also introduced. The popular Heston model is described and calibrated according to the code in Appendix B through call options on the SP500. Chapter 4 introduces the framework of control problems both in discrete, continuous time and the stochastic version. The fundamental Hamilton-Jacobi-Bellman equation is there derived. Finally, Chapter 5 makes use of all the theory previously described to solve the household portfolio choice problem with income and stochastic volatility. A numerical method to solve the non-linear PDE will be derived and implemented (Appendix C) to give the optimal controls.

Chapter 2

Finite difference schemes

2.1 Introduction

Finite difference schemes are a type of numerical method used to solve continuous PDEs through discrete approximation. The discrete approximation means that the PDE is evaluated only at a finite number of points called nodes on a specified mesh. In general increasing the number of points improves resolution and accuracy of the solution. The method relies on replacing the derivatives in the PDE with their finite differences formulas to get a discrete differential equation. The discrete differential equation is then evaluated at all points of the mesh and the resulting algebraic equations can be efficiently solved by linear algebra methods.
Technically, a finite difference scheme is defined as the specific combination of finite difference formulas. As we will see in a later chapter finite difference schemes play a crucial role in solving the non-linear PDEs associated with stochastic control problems.

2.2 Mesh

Consider a PDE for the function $\phi(x,t)$ defined for $0 \leq x \leq L$ and $0 \leq t \leq T$. For all finite difference schemes, the user must first specify the mesh. The mesh is usually taken to be uniformly spaced, therefore determined by N_x, N_t the number of nodes for the two variables. The discretization is given by

$$\Delta x = \frac{L}{N_x - 1} \text{ and } \Delta t = \frac{T}{N_t - 1} \tag{2.1}$$

Giving the mesh nodes (x_i, t_m) where $x_i = i(i-1)\Delta x$ **and** $t_m = (m-1)\Delta t$ for $i = 1, 2, ..., N_x$ and $m = 1, 2, ..., N_t$.

2.3 Difference Formulas

Difference formulas have general form

$$\frac{f(x+a)-f(x+b)}{b-a}$$

and it can be clearly seen that as $b-a \to 0$ the above expression will be the derivative of the function f. By Taylor's theorem, replacing the derivative with the finite difference approximation gives an error whose order depends on the order of the derivative approximated. The main three functions that we are going to use in a second order differential equation are

- First order Forward Difference

$$\frac{\partial f}{\partial x}|_{x_i} = \frac{f_{i+1}-f_i}{\Delta x} + \mathcal{O}(\Delta x) \tag{2.2}$$

- First order Backward Difference

$$\frac{\partial f}{\partial x}|_{x_i} = \frac{f_i - f_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \tag{2.3}$$

- Second Order Central Difference

$$\frac{\partial^2 f}{\partial x^2}|_{x_i} = \frac{f_{i+1}+2f_i+f_{i-1}}{\Delta x^2} + \mathcal{O}(\Delta x^2) \tag{2.4}$$

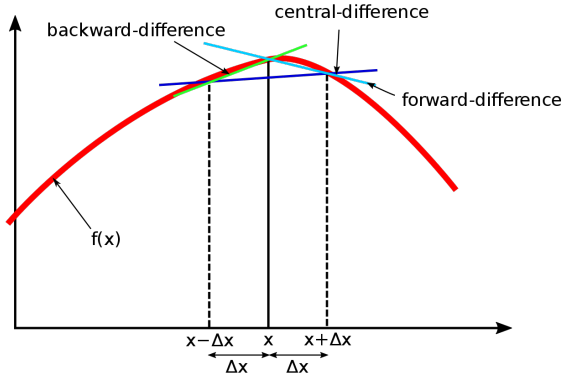

Figure 2.1: Finite Difference Formulas, image by Wikipedia

2.4 Finite difference schemes

The partial differential equations that occur in portfolio choice and asset pricing are second order elliptic or parabolic equations. Elliptic equations occur in infinite-horizon problems, where time plays no role and parabolic equations occur in finite-horizon problems where time is relevant. When the number of spatial state variables is small, such partial differential equations can be solved via finite difference

schemes, popular examples of which are FTCS (forward-time, centred space), BTCS (backward-time, centred space) and Crank-Nicholson, which differ in terms of computational effort, stability and convergence properties. The schemes will be illustrated by applying them to the Black-Scholes equation, a fundamental equation in option pricing, which is essentially the heat equation and functions as prototype for the parabolic partial differential equations which are common in asset pricing.

2.4.1 Black-Scholes Equation

The development of the Black-Scholes equation has been considered a turning point in the development of mathematical finance and option trading. It gave the first mathematical model for the pricing and hedging of derivatives. Although the assumptions of the model are not all empirically valid it is still currently used for its ease of calculation and robust mathematical framework.
The Black-Scholes partial equation is a parabolic partial differential equation with one time variable and one space variable and is given by

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{2.5}$$

where $t \in (0, T]$ and $S \in [0, \infty)$.V is the price of the derivative security, t is the current time, T is the expiry time of the derivative and S is the price of the underlying stock, r is the risk-free rate, σ is the volatility of the return of the underlying stock. The type of the derivative to be priced pins down the time-T boundary condition. For example, with a European call which has strike price, K, and expires at time T, $V(S_T, T) = \max(S_T - K, 0)$. Although in this particular case, an explicit solution for the Black-Scholes PDE can be found, numerical methods are widely used in practice to price more exotic derivatives.

Hereafter V_i^m will denote the value that the function $V(S, t)$ takes on grid at time m and stock price i, while ΔS and Δt will denote the mesh sizes.

2.4.2 FTCS

Forward Time, Centered Space scheme makes use of the forward equation to approximate the first order derivative in time and uses second order central difference to approximate the stock price derivatives.
The scheme is explicit as V^{m+1} can be easily represented as the matrix multiplication of a tridiagonal matrix and V^m. This simple representation makes it the easiest one to be implemented in code, however this come at a significant price as the scheme can yield unstable solutions if Δt is too large. [8]

$$\frac{V_i^{m+1} - V_i^m}{\Delta t} + \frac{1}{2}\sigma^2 S^2 \frac{V_{i+1}^m + 2V_i^m + V_{i-1}^m}{\Delta S^2} + rS_i \frac{V_{i+1}^m - V_i^m}{\Delta S} - rV_i^{n+1} = 0 \tag{2.6}$$

2.4.3 BTCS

Backward Time Centered Space scheme makes use of central difference for dealing with the second derivative and backward difference for the time. The scheme is slightly more difficult to implement as it is implicit but has the significant advantage of being unconditionally convergent. In the following section it will be shown how to construct the tridiagonal matrix and implement the method to solve the PDE.

$$\frac{V_i^{m+1} - V_i^m}{\Delta t} - \frac{1}{2}\sigma^2(i\Delta S)^2 \frac{V_{i+1}^{m+1} - 2V_i^{m+1} + V_{i-1}^{m+1}}{\Delta S^2} - ri\Delta S \frac{V_{i+1}^{m+1} - V_i^{m+1}}{\Delta S} + rV_i^{m+1} = 0 \tag{2.7}$$

2.4.4 Crank-Nicholson

Another widely used scheme is Crank-Nicholson [9] which makes use of backward time difference and the average between central difference evaluated at the current and previous time step. This scheme is implicit and stable as BTCS but more computationally expensive. The major advantage is that it provides quadratic errors both in time and stock price. Define $V_i^{m+\frac{1}{2}} = \frac{1}{2}(V_i^{m+1} - V_i^m)$

$$\frac{V_i^{m+1} - V_i^m}{\Delta S} + ri\Delta S \frac{V_{i+1}^{m+\frac{1}{2}} - V_{i-1}^{m+\frac{1}{2}}}{2\Delta S} + \frac{1}{2}\sigma^2 i^2 \Delta S^2 \frac{V_{i+1}^{m+\frac{1}{2}} + 2V_i^{m+\frac{1}{2}} - V_{i-1}^{m+\frac{1}{2}}}{2\Delta S^2} - rV_i^{m+\frac{1}{2}} = 0 \tag{2.8}$$

2.5 An application to Black-Scholes PDE

To implement the BTCS scheme an implicit equations system is developed

$$\mathbf{A}\mathbf{V}^{m+1} = \mathbf{V}^{\mathbf{m}} \tag{2.9}$$

which can be readily solved with standard techniques like LU decomposition.
The BTCS schemes states:

$$V_i^m = V_i^{m+1} + r\Delta t V_i^{m+1} - \frac{1}{2}\sigma^2 i^2 \Delta t[V_{i+1}^{n+1} + V_{i-1}^{n+1} + 2V_i^{n+1}] - \Delta tri[V_{i+1}^{n+1} - V_i^{n+1}] \tag{2.10}$$

Rearranging the scheme as follows it can be put in tridiagonal matrix form

$$V_i^{m+1}(1 + r\Delta t + \Delta t\sigma^2 i^2 + \Delta tri) + V_{i+1}^{m+1}(-\frac{1}{2}\Delta t\sigma^2 i^2 - \Delta tri) + V_{i-1}^{m+1}(-\frac{1}{2}\Delta t\sigma^2 i^2) = V_i^m \tag{2.11}$$

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} V_1^{m+1} \\ V_2^{m+1} \\ V_3^{m+1} \\ \dots \\ V_{N-1}^{m+1} \\ V_N^{m+1} \end{bmatrix} = \begin{bmatrix} V_1^m \\ V_2^m \\ V_3^m \\ \dots \\ V_{N-1}^m \\ V_N^m \end{bmatrix} \tag{2.12}$$

$a_i = -\frac{1}{2}\Delta t\sigma^2 i^2$
$b_i = 1 + r\Delta t + \Delta t\sigma^2 i^2 + \Delta tri$
$c_i = -\frac{1}{2}\Delta t\sigma^2 i^2 - \Delta tri$
Fixing σ the long term variance, which in Black-Scholes is fixed for all times, the prices can be computed with the numerical scheme above. The MATLAB code for the implementation of the BTCS scheme on Black-Scholes European call option PDE can be found in appendix A.

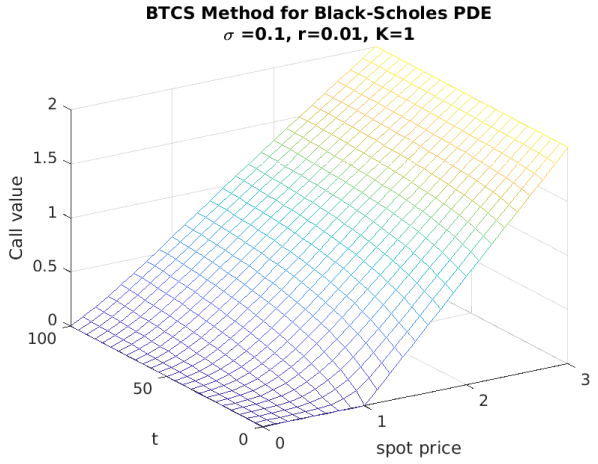

Figure 2.2: Surface of Call Option values for the Black-Scholes BTCS implementation

Chapter 3

Stochastic Models

3.1 Introduction to stochastic processes

The use of stochastic processes to model the stock price can be traced back to Louis Bachelier's 1900 thesis "Theory of Speculation" [10] which developed the diffusion processes framework in modelling options. The work fell into obscurity until Merton [11] and Samuelson [12] pioneered the application of continuous time stochastic processes to solve problems in financial economics. After the seminal works of Black and Scholes [13], this is the preferred framework in the theory of asset pricing.

3.1.1 Brownian Motion

Define a Stochastic process as a collection of random variables indexed by a set T defined on a probability space $(\Omega, \mathcal{F}, P)$ and taking values from some σ-algebra measurable space (S, Σ). For a continuous stochastic process take the index set T and the path of the process to be continuous and define a diffusion process as a continuous stochastic process that satisfies the Markov property at time t.
Define a Wiener Process also called Brownian Motion as a stochastic process satisfying the following properties:

- $W_0 = 0$
- W has independent increments: $\forall t, u, s > 0, W_{t+u} - W_t \geq 0$ are independent of past values $W_s, s \leq t$
- $\forall t, u > 0, W_{t+u} - W_t \sim \mathcal{N}(0, u)$
- W has continuous paths: W_t is continuous in t.

Diffusion Processes

A category of widely popular processes in finance are diffusion processes. These processes have almost surely continuous paths which makes them particularly apt to describe financial instruments that do not experience significant and frequent discontinuities.
Examples of diffusion processes are the Vasicek[14], Hull-White [15] and Black-Karasinski [16] models for the term structure of interest rates. These are also used to price interest rates derivatives like swaptions, floors and caps. Diffusion processes can also be used jointly to create more complex models like Heston model which will be described and employed later.
Diffusion processes, while still very popular in finance, struggle to describe processes with have significant and frequent jumps, therefore a popular alternative are mixed jump-diffusion models introduced by Merton [17]. However, the choice between the two models depends heavily on the the financial process considered and its timescale [18].

The general form for a diffusion process is

$$dx(t) = \mu(x(t), t)dt + \sigma(x(t), t)dz \tag{3.1}$$

where $\mu(x(t), t)$ and $\sigma(x(t), t)$ are time dependent but deterministic functions of Brownian motion $z(t)$. The two terms that determine the process are denoted respectively as the drift and volatility.
Since $dz = \sqrt{dt}d\epsilon \sim \mathcal{N}(0, dt)$ then we have $dx \sim \mathcal{N}(\mu(x(t), t)dt, \sigma^2(x(t), t))dt$. So the diffusion process is instantaneously normally distributed. However, for a finite interval the process is not normally distributed and to determine the distribution it will be necessary to calculate the Ito integral of $\mu(x(t), t)$ and $\sigma(x(t), t)$.

3.1.2 Ito integral

Ito's integral is the fundamental tool of Ito calculus which extends Riemann integral to stochastic processes. [19] The Ito's integral for a right continuous bounded process H with respect to a Brownian motion B and up to time t is the random variable to which the following sum converges in probability:

$$\int_0^t HdB = \lim_{n\to\infty} \sum_{[t_{i-1},t_i]\in\pi_n} H_{t_i-1}\Delta B_i \tag{3.2}$$

for a sequence of partitions π_n of $[0, t]$ with vanishing mesh. Note for the integral to be well defined the process H must be adapted which means that the value it takes at time t only depends on other information available at time t. As the diffusion process is adapted we can write its increments as

$$\int_0^T dX = \int_0^T \mu[x(t), t]dt + \int_0^T \sigma[x(t), t]dz \tag{3.3}$$

3.2 Ito's Lemma

Ito's lemma is a central result in stochastic calculus so much that it is also called the Fundamental Theorem of Stochastic Calculus. The name is well-deserved as it has ubiquitous applications and it will be later used to derive a framework to deal with with stochastic control problems. The lemma permits to calculate the infinitesimal change of a function of stochastic processes. For its role in stochastic calculus it can be rightly considered the stochastic equivalent of the chain rule.

Let the variable $x(t)$ follow the stochastic differential equation $dx(t) = \mu(x(t), t)dt + \sigma(x(t), t)dz$ (where dz Brownian motion) also let $F(x(t), t)$ be twice-differentiable function. Then the differential of F(x,t) is

$$dF = \partial_x F dx + \partial_t F dt + \frac{1}{2}\partial^2_{xx} F(dx)^2 \tag{3.4}$$

then substituting in the the SDE and noting $(dx)^2 = \sigma(x, t)^2 dt$ get

$$dF = \left[\partial_x F \mu(x(t), t) + \partial_t F + \frac{1}{2}\partial^2_{xx}\sigma(x(t), t)\right]dt + \partial_x \sigma(x(t), t)dz \tag{3.5}$$

Here we give an overview of the proof, a more rigorous approach can be found in Merton book on continuous time finance. [11]
The first step is to expand function according to the the Taylor series

$$F(x(t+\Delta t), t+\Delta t) = F(x(t), t)+\partial_x F\Delta x+\partial_t F\Delta t+\frac{1}{2}\partial^2_{xx}F(\Delta x)^2+\partial^2_{xt}F\Delta x\Delta t+\frac{1}{2}\partial^2_{tt}F(\Delta t)^2+o(3) \tag{3.6}$$

Then by substituting the discrete approximation

$$\Delta x = \mu(x, t)\Delta t + \sigma(x, t)\sqrt{dt}\epsilon \tag{3.7}$$

and taking the limit as $\delta \to dt$ and $\Delta f \to dF$ we would expect,in a deterministic scenario, that all second order terms disappear.
However, this is not the case for a function of stochastic processes because of the nature of $dz = \sqrt{\Delta t}d\epsilon$ which yields dz for the first order and dz^2 which converges to dt.
This last limit can be seen by proving the Ito integral for $\int_0^T [dz(t)]^2 = T$.
First consider the expectation of the following sum:

$$E_0[(\sum_{i=1}^{n} \Delta z_i)^2] = \sum_{i=1}^{n} E_0[(\Delta z_i)^2] \tag{3.8}$$

Since by definition dz is Brownian then

$$\Delta z_i \sim \mathcal{N}(0, t_i - t_{i-1}) \tag{3.9}$$

and by the fact that it has independent increments find

$$E_0[(\Delta z_i)^2] = Var[z_i - z_{i-1}] = t_i - t_{i-1} \tag{3.10}$$

which ultimately gives us

$$E_0[(\sum_{i=1}^{n} \Delta z_i)^2] = t \tag{3.11}$$

therefore

$$\int_0^t dz^2 = tdz^2 = dt \tag{3.12}$$

Finally we can rewrite the Taylor expansion as

$$dF = \partial_x F(\mu(x,t))dt + \partial_x F(\sigma(x,t))dz + \partial_t F\Delta t + \frac{1}{2}\partial_{xx}^2 F\sigma^2(x,t)dt \tag{3.13}$$

proving the lemma.

3.3 Girsanov's theorem

Another important instrument in our stochastic "toolbox" will be Girsanov's theorem. Girsanov's theorem plays a fundamental role in all mathematical finance as it permits to calculate how a process is modified under a change in probability measure. In particular it enables to change the probability measure for a stock price to a risk-free equivalent martingale measure.[20]

Assume we have a discrete Brownian motion

$$dX = adt + \sigma dz = adt + \sigma\sqrt{t}d\epsilon \tag{3.14}$$

with $d\epsilon$ being normally distributed $\sim \mathcal{N}(0,1)$ under P with $p(x)$ the probability density function.
Now assume we want a new probability measure P' where $d\epsilon \sim \mathcal{N}(\frac{\nu\sqrt{dt}}{\sigma}, 1)$ with probability density $p'(x) = p(x - \frac{\nu\sqrt{dt}}{\sigma})$ by standard probability result.
The questions at hand is what happens to the expectation of the process when we change the measure?
The definition of expectation for the Brownian motion at time t tells us

$$E_t^{\mathbf{P}'}[d\epsilon] = \int_{-\infty}^{\infty} p'(x)xdx \tag{3.15}$$

This can be conveniently rewritten in terms of $p'(x)$ as

$$\int_{-\infty}^{\infty} p(x)\frac{p'(x)}{p(x)}xdx = E_t[\frac{p'(d\epsilon)}{p(d\epsilon)}d\epsilon] \tag{3.16}$$

where $\frac{p'}{p}(x)$ is a random variable that distorts expectations into what is required.

Therefore through this ratio we can derive a stochastic process

$$\frac{M_{t+dt}}{M_t} = \frac{p'(dZ_t/\sqrt{dt})}{p(dZ_t/\sqrt{dt})} \tag{3.17}$$

which is an exponential martingale for measure $\mathbf{P}'$
As in the previous comment, we can see that $\frac{M_{t+dt}}{M_t}$ will help us distort expectations for the process. In particular it acts on the Brownian motion term

$$E_t^{\mathbf{P}'}[d\epsilon] = E_t^{\mathbf{P}'}[\frac{M_{t+dt}}{M_t}d\epsilon] \tag{3.18}$$

However we see that, nevertheless acting on the stochastic part, the distortion changes the deterministic drift

$$E_t^{\mathbf{P}'}[dx(t)] = E_t[\frac{M_{t+dt}}{M_t}dx(t)] = E_t[dx(t)] + E_t[\frac{dM_t}{M_t}d\epsilon] \tag{3.19}$$

Finally this is Girsanov's theorem.

3.4 Feynman-Kac

The final and most important tool that will be used to tackle control problems is Feynman-Kac theorem. This powerful theorem was developed to establish a connection between parabolic PDEs and expectations of stochastic processes [21]. In physics, this enables to solve PDEs by simulating random paths of stochastic processes for example with Monte Carlo methods.
In finance this enables to transform PDEs like Black-Scholes or Hamilton-Jacobi-Bellman equations into more tractable expectations of stochastic processes. However, it does also permit to use the opposite result and find expectations of some stochastic processes by an associated PDE.

Consider the partial differential equation

$$\partial_t U(x,t) + \mu(x,t)\partial_x U(x,t) + \frac{1}{2}\sigma(x,t)^2\partial_{xx}^2 U(x,t) - v(x,t)U(x,t) + f(x,t) = 0 \tag{3.20}$$

subject to terminal condition $U(x,t) = \Psi(x)$ where $\mu(x,t), \sigma(x,t), \Psi(x), v(x,t), f(x,t)$ are known functions and $U : R \times [0,T] \to R$ then Feynman-Kac theorem states that the solution to the above PDE is given by the following expectation of a stochastic process

$$U(x,t) = E^Q\Big[\int_t^T e^{-\int_t^\tau V(X_\tau,\tau)d\tau} f(X_r,r) + \int_t^T e^{-\int_t^T V(X_\tau,\tau)d\tau}\Psi(X_T)|X_t = x\Big] \tag{3.21}$$

under probability measure Q such that X is an Ito's process driven by the equation $dX_t = \mu(X,t)dt + \sigma(X,t)dZ^Q$ where dZ^Q is Brownian motion under Q and with initial condition X(t)=x.

3.5 Heston model

Heston model was developed by Steven Heston in 1993 [22], so it is a relatively new model that tries to overcome some of the problematic assumptions of Black-Scholes. In particular since it prescribes a stochastic process for volatility it no longer assumes it to be constant. This assumption greatly helps the model to better simulate the unpredictability of markets. In the model, price S_i and volatility v_t of asset i are described through two stochastic processes:

$$dS_i = \mu S_t dt + \sqrt{v_t} S_t dW_t^S \tag{3.22}$$

$$dv_t = k(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^v \tag{3.23}$$

subject to the two Brownian motions W_t^S and W_t^v with correlation ρ and rate of return μ. T other parameters are θ the long run variance, κ the rate for which volatility reverts to its long run value, ξ the volatility of volatility which determines variance of v_t and v_0 the initial volatility.
If $2\kappa\theta > \xi^2$ then the process satisfies Feller condition and it is can be shown to be strictly positive.[23]

3.5.1 Calibrating the Heston Model

The Heston model is quite flexible and currently enjoys a strong reputation in the industry thanks to its realistic assumption. However, the model comes with the downside of needing substantial parameter calibration which can be done by numerical methods. Various methods can be used to solve these non-linear fitting problem. In the calibration code in Appendix B, which follows the implementation of [24], some of the most popular algorithms i.e. Levenberg-Marquardt, Least Squares, Differential Evolution and Basin Hoppping have been employed.

The following section on the household portfolio choice problem motivates us to use the Heston model to describe the return of an Exchange Traded Fund (ETF). The parameters are therefore calibrated on the SP500 call options.
It was a specific aim of the author to calculate the parameters from the most recent data from the options market to later implement it in the portfolio code. However, the current very dynamic and unpredictable economic situation make the estimation very imprecise.
Inspecting the latest financial data used from Yahoo Finance it seems that the option market expects a lot of uncertainty. The range of strikes from here to December changes so dramatically that it is difficult to find consistent and not sparse prices matrix. It has been found that all four methods applied on this data seem to yield almost reasonable results but with very significant errors of estimation.
Processing the matrix by reducing outliers reduced significantly the error of estimation but at the cost of having unrealistic parameters. Testing the code with previous data from 2016/2017 given by the author of [24] gives consistent and correct results.

Therefore, this dataset has been used in calibrating the Heston model for the portfolio problem. In wait for less uncertain times, the code for retrieving and fitting options data to Heston model is given in Appendix B.

Table 3.1: Estimated model parameters SP500 call options data

Parameters	June 21 Raw Data	June 21 Preprocessed	2016/2017 Data
θ	0.000015	1.751837	0.122251
κ	10.072485	0.037471	4.996804
ξ	0.010340	0.67546	0.849266
ρ	-0.916820	-0.988886	-0.637706
v_0	0.001545	0.00562 1	0.079484
error	135805.64	95.36	2.87608

Chapter 4

Control problems

4.1 Introduction

A central topic to this thesis is the problem of optimising the choice that an agent can make in several time steps to get the best possible result. This is a well-explored topic in applied mathematics literature where it takes the name of dynamical programming. Dynamical programming refers to a method of solving temporal problems by breaking it down in several time steps and solving in a recursive manner. This is the famous "Bellman's principle of Optimality" that lays the foundations of Bellman equation, a powerful tool to solve many discrete inter-temporal optimization problems and its continuous time equivalent the Hamilton-Jacobi-Bellman (HJB) equation.

4.2 Dynamic programming framework

First of all, describing the problem requires a brief Dramatis Personae. Here will be defined the main 'actors' of the problem at hand:

- **Objective function** prescribes the objective of the optimization problem over the time frame considered which can be maximising utility of consumption, minimising cost or other complex requirements.

- **Value function** $V(x)$ is the value of the optimised objective function.

- **State variables** $S(x_t, c_t)$ keeps track at every time the current situation.

- **Control variables** c_t given the state variables at any point in time, the control variables permit the agent to influence or choose the following path. Therefore, the control variables are the optimizers of the problem. The goal of the problem is to find expressions for the control variables that satisfy the objective function and depend on the state variables at time t.

4.3 Bellman equation

The fundamental principle on which dynamic programming relies is the so called "Bellman Principle of Optimality":

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. [25]

The infinite time horizon dynamic programming is given by

$$V(x_0) = \max_{\{c_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \delta^t P(x_t, c_t) \tag{4.1}$$

with $P(x_t, c_t)$ the payoff at time t of the current state and controls, the discount factor $\beta^t < 0$, $\beta < 1$ which is introduced to take into account impatience. The problem is subject to constraint $c_t \in \Gamma(x_t)$ and evolution law $x_{t+1} = S(x_t, c_t)$ which determines the new state.
Following the principle of optimality, setting the first decision (following the constraints):

$$\max_{a_0} \left\{ F(x_0, c_0) + \delta \Big[\max_{\{c_t\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \delta^{t-1} P(x_t, c_t) \Big] \right\} \tag{4.2}$$

This will permit to rewrite the problem as

$$V(x_0) = \max_{c_0} \left\{ P(x_0, c_0) + \delta v(x_1) \right\} \tag{4.3}$$

Since this framework can be started at any point in time the time subscripts can be dropped to get Bellman Equation:

$$V(x) = \max_{c \in \Gamma(x)} \left\{ P(x, c) + \delta v(S(x, c)) \right\} \tag{4.4}$$

4.4 Hamilton-Jacobi-Bellmann

HJB equation plays a crucial role in this thesis as it enables to treat and solve continuous time stochastic control problems. The equation plays an analogue role to the Bellman equation in discrete time.
To derive the HJB equation consider a continuous time stochastic control problem of form

$$V_T(x(0), 0) = \max_{c} \Big\{ \int_0^T P[x(t), u(t)] dt \Big\} \tag{4.5}$$

subject to $\dot{x}(t) = S[x(t), c(t)]$.
Using Bellman's principle of optimality over an infinitesimal increase in time it becomes

$$V(x(t),t) = \max_c \Big\{ V(x(t+dt),t+dt) + \int_t^{t+dt} P(x(s),c(s))ds \Big\} \tag{4.6}$$

whose value function can be Taylor expanded to give

$$V(x(dt+t),t+dt) = V(x(t),t) + \partial_t V(x,t)dt + S(x(t),c(t))\partial_x V(x,t)dt + o(dt) \tag{4.7}$$

Now, subtracting $V(x(t),t)$ from both sides and dividing by dt get

$$0 = \max_c \Big\{ \partial_t V(x,t) + S(x(t),c(t))\partial_x V(x,t) + o(dt)/dt + \frac{1}{dt}\int_t^{t+dt} P(x(s),c(s))ds \Big\} \tag{4.8}$$

Finally, taking the limit as dt tends to zero we get the HJB equation:

$$\partial_t V(x,t) + \max_c \big\{ S(x,c)\partial_x V(x,t) + P(x,c) \big\} = 0 \tag{4.9}$$

4.5 Stochastic control problems

A natural kind of extension for dynamic programming is the case where state variable changes according to some stochastic law. The framework and the principle developed by Bellman are still applicable with only minor changes necessary. As in the last example we have a continuous time dynamic problem with objective function:

$$V_T(x(0),0) = \sup_c E_0\Big\{ \int_0^T e^{-\rho t} U(x(t),u(t))dt \Big\} \tag{4.10}$$

evolving according to the law

$$dx(t) = \mu(x(t),c(t))dt + \sigma(x(t),c(t))dz \tag{4.11}$$

The discount factor $e^{-\rho t}$ was introduced for impatience was introduced due to its utility in economic applications. For consistency with later chapters, we now denote U(x(t),t) for the payoff function.
Similarly to the previous case the Hamilton-Jacobi-Bellman equation can be derived to solve the problem.

$$0 = \max_{c(t)} P(s(t),c(t)) - \rho V(s(t),t) + V'(s(t),t)f(s(t),c(t)) + \frac{1}{2}V''(s(t))(\sigma(s(t),c(t))) \tag{4.12}$$

To derive it we will make use of Ito's lemma.
First derive the Bellman equation using the principle of optimality

$$V(x(t),t) = \sup_{c(t)_{t\in[t,t+dt)}} E_t\big[E_t \int_t^{t+dt} e^{-\rho(u-t)} U(s(u),c(u))du + e^{-\rho dt} V(s(t+dt),t) \big] \tag{4.13}$$

Applying first order Taylor expansion to the terms yields:

$$E_t \int_t^{t+dt} e^{-\rho(u-t)} U(s(u), c(u)) du = U(s(t), c(t)dt + o(t) \tag{4.14}$$

$$e^{-\rho dt} = 1 - \rho dt + o(t) \tag{4.15}$$

$$dt E_t V(s(t+dt), t) = dt V(s(t)) + o(t) \tag{4.16}$$

Finally recombining get Bellman's equation

$$V(s(t), t) = \sup_{c(u)_{u \in [t, t+dt)}} U(s(t), c(t))dt + E_t V(s(t+dt)) - \rho dt V(s(t)) + o(dt) \tag{4.17}$$

Now Ito's lemma for the function states

$$v(s(t+dt)) = V(s(t)) + dV(s(t)) \tag{4.18}$$

The last term gives in accordance with Ito's Lemma

$$E_t V(s(t+dt)) = V(s(t)) + V_s(s(t))\mu(s(t), c(t))dt + \frac{1}{2}\partial^2_{ss} V(s(t))\sigma^2(s(t), c(t))dt + o(dt) \tag{4.19}$$

Reinserting this last result back into the Bellman's equation and taking the limit as $dt \to 0$ we get the HJB equation for the problem:

$$0 = \sup_{c(t)} U(s(t), c(t)) - \rho V(s(t)) + V'(s(t))\mu(s(t), c(t)) + \frac{1}{2} V''(s(t))\sigma^2(s(t), c(t)) \tag{4.20}$$

Notice how in the stochastic version the second order term persists. This, like in Ito's lemma, is due to the quadratic variation of the Brownian motion.
To solve the stochastic control problem now it is just necessary to find the solutions to the first order conditions with respect to the controls and substituting the expression for the optimal control

$$0 = U(s(t), c^*(t)) - \rho V(s(t)) + V'(s(t))\mu(s(t), c^*(t)) + \frac{1}{2} V''(s(t))\sigma^2(s(t), c(t)) \tag{4.21}$$

The resulting equation is a non-linear PDE which in general does not have a closed-form solution and in general has to be solved by numerical methods. Finite difference schemes are a very popular choice in the field.
Furthermore, since it does not have boundary conditions does not satisfy the usual solution to a differential equation and it requires a new concept solution called viscosity solutions[26]. A detailed discussion of the properties of these solutions is beyond the scope of this paper but Barles book on the topic[27] can be consulted for further details.

Chapter 5

Portfolio Choice

5.1 Introduction and Merton's original setting

A question is ever present in finance: "How and how much to invest?" If considered in a single time frame one of the most popular frameworks for answering this question is the CAPM model developed by Sharpe in 1964 [28] which prescribes investing in a combination of a risk-less asset like a bond or a bank account and in a special portfolio of risky assets called the market portfolio which should represent the entire market. The quantity to be invested in each however is dependent on the risk aversion of the investor.

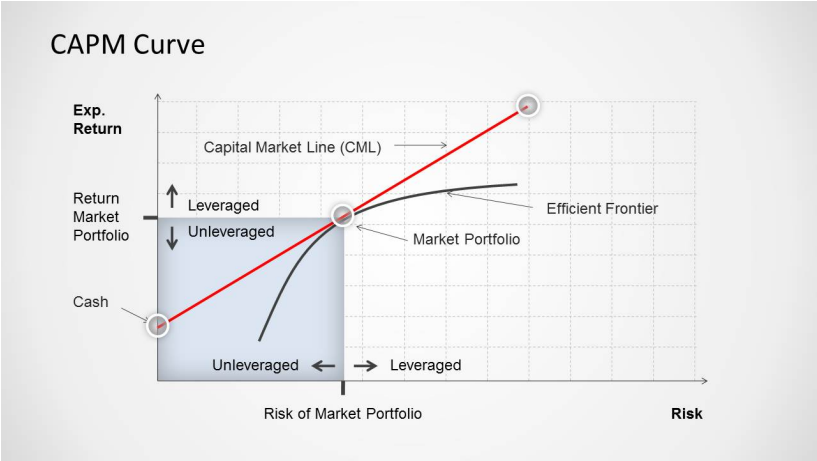

Figure 5.1: CAPM optimal portfolio principle [29]

If we consider multiple time frames or a continuous time frame the question can be nicely framed in a stochastic control problem. The framework introduced by Merton in [1] is a particularly powerful one and incredibly flexible. In his paper, Merton used stochastic control and HJB equation to solve the inter-temporal asset allocation problem by maximising utility through consumption and investment in a risky and risk less portfolio. Quite strikingly the model and the HJB equation can be solved to yield a closed-form solution. This was the basis for his Intertemporal CAPM model which divides portfolio allocation in a myopic and intertemporal part.

The framework is particularly powerful because it permits several modification to take into account income, more complex stock returns, pensions, bequest and other financial flows. The model also found theoretical confirmation in the work of Cox, Ingersoll, Ross [30] which starting from more primitive assumptions and developing their economy model derived results in accordance with Merton.
Following the fruitful literature of expanding the stochastic control framework to more realistic settings, we will here solve the problem for a household receiving an income and investing in a ETF subject to Heston model.

5.2 Households portfolio choice problem

Consider the following stochastic optimal control problem in which an household is trying to optimize the utility of their consumption by investing in a portfolio comprising a risk-less and risky assets. The wealth of the household at time t is given by W_t and it evolves according to

$$W_{t+dt} = (W_t + (Y - C_t)dt)(1 + dR_{p,t})$$

where C_t denotes the consumption of the household at time t, Y denotes the income and $dR_{p,t} = (1 - \phi_t)rdt + \phi_t dR_t$ the instantaneous return of the portfolio. The consumption is a control variable and therefore decided by the Household at each time.

In accordance with CAPM theory, it will be assumed that the household can invest in a combination of a risk-less asset and a market ETF. The return of the ETF will modeled by

$$dR_t = (r + \lambda v_t)dt + \sqrt{v_t}dZ_t \tag{5.1}$$

and

$$dv_t = \kappa(\theta - v_t)dt + \epsilon\sqrt{v_t}dZ_{v,t} \tag{5.2}$$

which is the previously described Heston Model.
If ϕ_t denotes the control variable of the ratio invested in the risky asset then we get the following continuous stochastic law also called the dynamic inter-temporal budget constraint

$$dW_t = (Y - C_t)dt + W_t[rdt + \phi_t(\lambda v_t dt + \sqrt{v_t}dZ_t)] \tag{5.3}$$

which is subject to the two controls C_t and ϕ_t.
This can be expressed in terms of the total stock of wealth at time t by the following transformation:

$$V_t = W_t + (1 - e^{-r(T-t)})\frac{Y}{r} \tag{5.4}$$

Note that if we consider an infinite time frame $T = \infty$ it becomes:

$$V_t = W_t + \frac{Y}{r} \tag{5.5}$$

Which takes into account the present value of future labor income flows. The presence of an income enables the households to scale up their investment in the ETF by a factor.

$$\hat{\phi}_t = \frac{V_t}{W_t}\phi_t \tag{5.6}$$

This relation enables to ignore the effect of the income for the sake of the control problem and calculate the optimal controls as scaled version of the controls for the incomeless problem.

5.2.1 Hamilton-Jacobi-Bellman equation

Therefore in the framework of the stochastic control problem the household seeks to maximise through consumption and risk asset investment rate

$$\sup_{(C_s)_{s\geq t},(\phi_s)_{s\geq t}} E_t \int_0^\infty e^{-\delta(s-t)} u(C_s) ds \tag{5.7}$$

which is the present expected utility of the future consumption where

$$u(x) = \frac{x^{1-\gamma}}{1-\gamma} \tag{5.8}$$

is the CRRA utility function. The dynamic budget constraint is given by

$$dW_t = -C_t dt + W_t[rdt + \phi_t(\lambda v_t dt + \sqrt{v_t} dZ_t)] \tag{5.9}$$

Now from the theory of optimal control introduced in the previous chapter we can derive the Hamilton-Jacobi-Bellman equation for the problem as:

$$0 = \sup_{C_t} u(C_t) - \delta J_t + \sup_{\phi_t} E\left[\frac{dJ_t}{J_t}\right] \tag{5.10}$$

HJB equation provides a useful tool for solving the inter-temporal problem. However, it is not the unique method of solution. Pontryagin maximum principle and hamiltonians also yield the solution to the problem. The two methods are connected and can be reciprocally derived. [31]
A significant difference is that Pontryagin does not assume time consistency (i.e. following the same strategy until end time) which is helpful when the agent might change strategy or management like corporations or central banks.

Using Ansatz $J_t = H(v_t)^\gamma U(W_t)$ we derive the following PDE through HJB equation:

$$0 = 1 - k(v)H(v) + \mu_v^{'}(v)H'(V) + \frac{1}{2}\sigma_v^2(v)H''(v) \tag{5.11}$$

and

$$k(v) = \frac{\delta}{\gamma} + (1-\frac{1}{\gamma})\Big(r + \frac{1}{2}\gamma v\Big(\lambda^2 - \gamma^2\epsilon^2(1-\rho^2)\Big(\frac{H'(v)}{H(v)}\Big)^2\Big)\Big) \tag{5.12}$$

$$\mu_v^{'}(v) = k'(\theta' - v) \tag{5.13}$$

$$\sigma_v^2(v) = \epsilon^2 v \tag{5.14}$$

5.2.2 Finite difference scheme using a Markov process

To solve the PDE a recursive scheme will be formulated for H as in [32]. Here the recursive scheme method will be outlined. First of all, applying Feynman-Kac theorem with the following identifications $V(x,t) \equiv k(v_t)$ and $\mu(x,t) \equiv \mu'_v(v_t)$, the PDE can be transformed in an expectation depending on the risk-free measure P'.

$$H(v_t) = E_t^{P'}\Big[\int_t^{\infty} e^{-\int_t^u k_s ds} du\Big] \tag{5.15}$$

with volatility evolving according to the law:

$$dv_t = \mu'_v(v_t) + \sigma_v(v_t) dZ_{v,t}^{P'} \tag{5.16}$$

with $dZ_{v,t}^{P'}$ standard Brownian motion under P.
The change in probability measure had induced a change in the drift term which can be calculated by Girsanov's theorem as follows. First, since the stochastic process for volatility must be equal under the two measures, and applying Girsanov's theorem

$$\mu'_v(v_t) = \mu_v(v_t)dt + E\Big[\frac{dM'_t}{M'_t} dv_t\Big] \tag{5.17}$$

The next step is identifying M'_t which is an exponential martingale under P. Plugging back in the definition for the drift get

$$\kappa'(\theta' - v_t) - \kappa(\theta - v_t)dt = E\Big[\frac{dM'_t}{M'_t} dv_t\Big] \tag{5.18}$$

Since $\kappa'\theta' = \kappa' \frac{\kappa}{\kappa'}\theta = \kappa$ it can be further simplified to $E\Big[\frac{dM'_t}{M'_t} dv_t\Big] = -(\kappa' - \kappa)v_t dt = \frac{(\gamma-1)\lambda\rho\epsilon}{\gamma} v_t dt$
This way we can define P' through M' and an event A that might or might not take place at date T.

$$E_t^{P'}[I_A] = E_t^P\Big[\frac{M'_T}{M'_t} I_A\Big] \tag{5.19}$$

Finally can derive a recursive scheme using the following formula
Splitting the integral, recognising the $H(v_{t+dt})$ form and considering the first order approximations, we see that

$$H(v_t) = E_t^{P'}\Big[\int_t^{t+dt} e^{-\int_t^u k_s ds} du\Big] + E_t^{P'}\Big[\int_{t+dt}^{\infty} e^{-\int_t^u k_s ds} du\Big] \tag{5.20}$$

$$= dt + E_t^{P'}\Big[\int_{t+dt}^{\infty} e^{-\int_t^u k_s ds} du\Big] + o(dt) \tag{5.21}$$

$$= dt + E_t^{P'} e^{-\int_t^{t+dt} k_s ds} E_{t+dt}\Big[\int_{t+dt}^{\infty} e^{-\int_{t+dt}^u k_s ds} du\Big] + o(dt) \tag{5.22}$$

$$= dt + E_t^{P'} e^{-\int_t^{t+dt} k_s ds} H(v_{t+dt}) + o(dt) \tag{5.23}$$

$$= dt + e^{-k_t dt} E_t^{P'}[H(v_{t+dt})] + o(dt) \tag{5.24}$$

$$= dt + (1 - k_t dt) E_t^{P'}[H(v_{t+dt})] + o(dt) \tag{5.25}$$
$$= E_t^{P'}[dt + (1 - k_t dt) H(v_{t+dt})] + o(dt) \tag{5.26}$$
$$\tag{5.27}$$

$$H(v_t) = dt + e^{-k_t dt} E_t^{P'}[H(v_{t+dt})] = E_t^{P'}[dt + (1 - k_t dt) H(v_{t+dt})] \tag{5.28}$$

From here we can set up a discretization of the state space in volatility $\{v_1, ..., v_{N+1}\}$. This will be done using a continuous time Markov chain to approximate volatility under the new measure. This Markov technique was recently outlined by Bhamra [7] and this is one of the first implementations of this method to portfolio choice problems.
The Markov chain is set up with generator matrix

$$S = \begin{bmatrix} -p_{1,2} & p_{1,2} & 0 & \dots & 0 & 0 \\ p_{2,1} & -(p_{1,2} + p_{2,3}) & p_{2,3} & 0 & \dots & 0 \\ 0 & p_{3,2} & -(p_{3,2} + p_{3,4}) & p_{3,4} & \dots & \dots \\ \dots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & p_{N,N-1} & -(p_{N,N-1} + p_{N,N+1}) & p_{N,N+1} \\ 0 & 0 & \dots & 0 & p_{N+1,N} & -p_{N+1,N} \end{bmatrix} \tag{5.29}$$

where

$$p_{n,n+1} = \frac{\mu_v'(v_n)}{\Delta v} I_{\mu_v'(v_n)>0} + \frac{\sigma_v^2(v_n)}{2(\delta)^2} \tag{5.30}$$

$$p_{n,n+1} = -\frac{\mu_v'(v_n)}{\Delta v} I_{\mu_v'(v_n)<0} + \frac{\sigma_v^2(v_n)}{2(\delta)^2} \tag{5.31}$$

Here the chain built moves along the volatility according to the measure P' with $p_{n,n+1}dt$ being the probability of volatility increasing by one discretization after time dt. The form the generator takes is also the typical one of a birth death process in fact the chain increases and decays by unit increments (dv) instantaneously. It follows that the transition probabilities matrix is given by e^{Sdt} and the recursive scheme

$$H = e^{Sdt}[1dt + (\mathbf{I} - \mathbf{K}dt)H] \tag{5.32}$$

where a non-linear operator (as it depends on the value taken by $K(H)$) is applied to H to give H. Therefore H represents the fixed point of this operator and a recursive scheme can be set up to converge. However, it will first be necessary to discretize time and invert e^{Sdt}. Using a Taylor series expansion get

$$(e^{-S\Delta t}) = \mathbf{I} + \sum_{n=1}^{\infty} (-1)^n \frac{S^n (\Delta t)^n}{n!} \approx \mathbf{I} - \mathbf{S}\Delta t \tag{5.33}$$

Giving the final form for the recursive scheme as $\bar{H} = \mathbf{I} - \mathbf{S}\Delta t[1dt + (\mathbf{I} - \mathbf{K}dt)H] + o(\Delta t)$
Finally let $H_n' = \frac{H(v_{n+1}) - H(v_n)}{\delta v}$ and

$$k(v_n) = \frac{\delta}{\gamma} + (1 - \frac{1}{\gamma})(r + \frac{1}{2}\gamma v_n(\lambda^2 - \gamma^2\epsilon^2(1 - \rho^2)\big(\frac{H_n'}{H(v_n)} I_{\mu_v'(v_n)>0} + \frac{H_{n-1}'}{H(v_n)} I_{\mu_v'(v_n)<0}\big)) \tag{5.34}$$

with $H(v_0), H(v_{N+1}) = 0$ Starting from an initial guess for $\bar{H}$ and applying the recursive scheme the estimate will eventually uniformly converge to the true value of $\bar{H}$.
Now the optimal portfolio and expenditure for the income-less problem are

$$\phi_t = \Big[\frac{\lambda}{\gamma} - \rho\epsilon\frac{H'(v_t)}{H(v_t)}\Big] \tag{5.35}$$

$$C_t = \frac{W_t}{H(v_t)} \tag{5.36}$$

Giving the optimal controls for the income version as:

$$\phi_t = \frac{W_t + \frac{Y}{r}}{W_t}\Big[\frac{\lambda}{\gamma} - \rho\epsilon\frac{H'(v_t)}{H(v_t)}\Big] \tag{5.37}$$

$$C_t = \frac{1}{H(v_t)}\big(W_t + \frac{Y}{r}\big) \tag{5.38}$$

Implementing the method in the code in appendix C the following plots have been derived for the case of a low risk-averse investor ($\gamma = 2$) and market given by the Heston parameters calibrated in chapter 3 with the only modification of having ρ=0.5 for a more realistic model.
As can be seen by (5.36)(5.37) the model suggests the investor to calibrate his portfolio according to the volatility of the market. This can be done by employing the VIX index as our $\sqrt{v_t}$ and calibrating the portfolio according $\phi(v_t)$ and the changes in the VIX index.

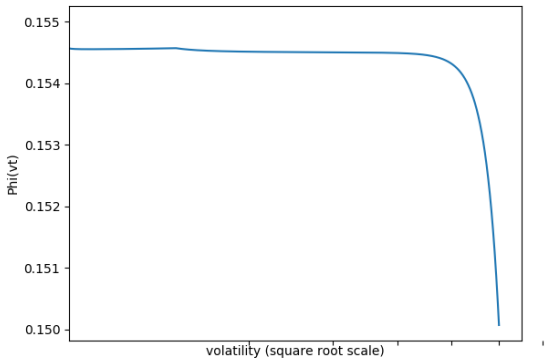

Figure 5.2: Optimal risky asset allocation ratio as a function of $\sqrt{vt}$, the x-axis spans from 0.001 to $\sqrt{1.5}$, risk-aversion =2

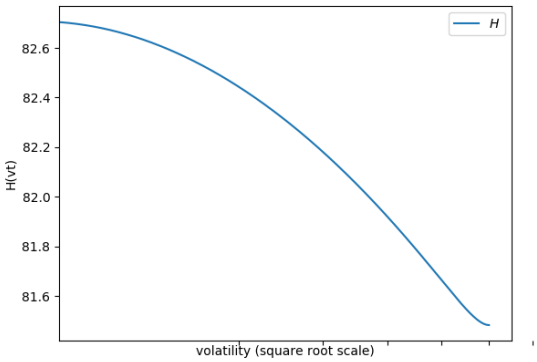

Figure 5.3: $H(v_t)$ against $\sqrt{vt}$, the x-axis spans from 0.001 to $\sqrt{1.5}$, risk-aversion $=2$

Chapter 6

Conclusion

This paper presents and builds the numerical methods, stochastic calculus and control theory necessary for portfolio choice problems with the aim of solving an household problem with income and stochastic volatility returns. Numerical methods are implemented to find solutions to the Black-Scholes equation in Appendix A and Appendix B deals with calibrating the Heston model through call options on the SP500. The final chapter sets up and solves for the optimal controls of the household problem through a finite difference method implemented in the code in Appendix C.

Solving the household problem, the main finding is that the quantity invested in the ETF increases with excess return λ , current wealth W_t and income Y while decreases with increasing risk aversion γ , interest rates r and volatility of volatility ϵ. This is consistent with real markets and economy as high net-worth individuals tend to make a larger use of financial products, and higher excess returns on the market encourage more investments. Note that the ρ term decides and scales the contribution of the second term.
Similarly the result from the optimal expenditure are consistent with the real world as from the table above we can see that $H(v_n)$ decreases with volatility and therefore the household "flights" to safety as the volatility of the market increases.
Finally the optimal portfolio ratio formula (5.37) can be analysed in terms of the myopic $\frac{\lambda}{\gamma}$ and intertemporal part $-\rho\epsilon\frac{H'(v_t)}{H(v_t)}$. This breakdown tells us the ratio that is allocated to the current best return of the market (the myopic term) and the part that is used to hedge changes in the state variable i.e. volatility.

To conclude, the author would like to introduce an idea that he recently developed while learning about the topic of stochastic controls. In light of the various modifications of the utility function, he believes an interesting development to the theory would be considering utility that it is exogenous and depends on a competitive environment. Interest in the topic is motivated by the coursework project done for the module "Dynamics of Games" which employed no regret learning to solve Colonel Blotto allocation problems and in a US elections context.
The framework envisioned would be the following:

- The agent tries to win a series of m zero-sum battles leading to winning or losing a war depending on winning a majority of the battles.
- Each battle can be modeled as a Colonel Blotto game with b equivalently valued battlefields. To win the battle it would be required to win a majority of battlefields. For a start the number of battlefields can be taken to be $b = 1$.
- The agent and the opponent have a starting predetermined discrete number of troops A_0 and O_0. However, only the agent knows also the starting number of troops of the opponent.
- At each battle the two participants are called to decide how many troops "consume" by employing them in battle. The remaining would be invested in a portfolio. For an initial enquiry the portfolio would be taken to be purely risk-free.
- The aim of the two players would be to maximise the expected utility of their consumption, i.e. maximise the probability of winning.
- To make the game more interesting the opponent initial resources will be assumed to be more than the agent ones.

The first step would be to determine the evolution of the utility function through time, this would determine if Bellman's optimality principle is applicable. If it is, then the problem should be solved recursively and a corresponding Bellman equation should be set up and potentially solved through the methods outlined in this paper.

For this project, in addition to the theory here exposed, the papers in sequential games by Klumpp, Konrad [33] and Solomon [34] would be a good starting point for analysing how the strategy changes without investment income.
For determining how the players would allocate resources in multiple battlefields an excellent method is given by the paper of Behnezhad et. al. [35] which gives an extremely efficient algorithm.

Chapter 7

Appendix A

The code is implemented in MATLAB. The LU solving functions are given by [8]

```
% Input: nt = number of steps. Default: nt = 10;
% nx = number of mesh points in x direction. Default: nx=20
% alpha = diffusion coefficient. Default: alpha = 0.1
% L = length of the domain. Default: L = 1;

nt = 25;
nx = 25;
sigma = 0.1;
L = 3;
tmax = 100;
K = 1;
r = 0.01;
% --- Compute mesh spacing and time step
dx = L/(nx-1);
dt = tmax/(nt-1);
% --- Create arrays to save data for export
x = linspace(0,L,nx)';
t = linspace(0,tmax,nt);
U = zeros(nx,nt);
% --- Set IC and BC
U(:,1) = max(0,(x-K));
u0=0;
uL=L-K;
% --- Coefficients of the tridiagonal system

kappa = @(m) (sigma^2*m.^2)./2;
lambda=@(m) (r*m);
```

```
a = ones(nx,1);
for i = 1:length(a)
    a(i)=-0.5*dt*sigma^2*i^2;
    % subdiagonal a: coefficients of phi(i-1)
end
c = ones(nx,1);
for i = 1:length(c)
    c(i)=a(i)-r*i*dt; % superdiagonal c: coefficients of phi(i+1)
end
b = ones(nx,1);
for i = 1:length(b)
    b(i)=-2*a(i)+r*i*dt+1+r*dt; % diagonal b: coefficients of phi(i)
end

b(1) = 1; c(1) = 0; % Fix coefficients of boundary nodes
b(end) = 1; a(end) = 0;
[e,f] = tridiagLU(a,b,c); % Get LU factorization of coefficient matrix
%--- Loop over time steps
for m=2:nt
d = U(:,m-1); % update right hand side
d(1) = u0; d(end) = uL; % overwrite BC values
U(:,m) = tridiagLUSolve(d,a,e,f,U(:,m-1)); % solve the system
end
%plot the surface
[xx,tt]=meshgrid(x,t);
h=mesh(xx,tt,U.');
%set(h,'LineStyle','none')
ylabel('t'),
xlabel('spot price')
zlabel('Call value')


function v = tridiagLUSolve(d,a,e,f,v)
% tridiagLUsolve Solve (LU)*v = d where L and U
%are LU factors of a tridiagonal matrix
%
% Synopsis: v = tridiagLUsolve(d,e,f)
% v = tridiagLUsolve(d,e,f,v)
%
% Input: d = right hand side vector of the system of equatoins
% e,f = vectors defining the L and U factors of the tridiagonal matrix.
% e and f are obtained with the tridiagLU function
% v = solution vector. If v is supplied, the elements of v are over-
% written (thereby saving the memory allocation step). If v is not
% supplied, it is created. v is used as a scratch vector in the
```

```
% forward solve.
%
% Output: v = solution vector
n = length(d);
if nargin<5, v = zeros(n,1); end
%--- Forward substitution to solve L*w = d
v(1) = d(1)/e(1);
for i=2:n
v(i) = (d(i) - a(i)*v(i-1))/e(i);
end
%--- Backward substitution to solve U*v = w
for i=n-1:-1:1
v(i) = v(i) - f(i)*v(i+1);
end
end
function [e,f] = tridiagLU(a,b,c)
% tridiagLU Obtain the LU factorization of a tridiagonal matrix
%
% Synopsis: [e,f] = tridiag(a,b,c)
%
% Input: a,b,c = vectors defining the tridiagonal matrix. a is the
% subdiagonal, b is the main diagonal, and c is the superdiagonal
%
% Output: e,f = vectors defining the L and U factors
%of the tridiagonal matrix
n = length(a);
e = zeros(n,1); f = e;
e(1) = b(1);
f(1) = c(1)/b(1);
for i=2:n
e(i) = b(i) - a(i)*f(i-1);
f(i) = c(i)/e(i);
end
end
```

Chapter 8

Appendix B

The code is implmented in Python The heston calibration was implemented as in [24]

```
"""
the implemntation got financial data from yahho finance
through the yahoo_fin python package on date 04 June 2021


from yahoo_fin import options
Chain = options.get_options_chain("spy")
dates= options.get_expiration_dates("spy")

dates = ["6/7/2021", "6/9/2021", "6/11/2021",
                    "6/14/2021", "6/16/2021", "6/18/2021",
                    "6/21/2021", "6/23/2021", "6/25/2021",
                    "6/28/2021", "6/30/2021", "7/2/2021",
                    "7/6/2021", "7/9/2021", "7/16/2021",
                    "8/20/2021", "9/17/2021", "9/30/2021",
                    "10/15/2021", "11/19/2021", "12/17/2021",
                    "12/31/2021"]

info = {}
for date in dates:
    print(date)
    info[date] = options.get_options_chain("spy",date)
"""

 ######################################################
import numpy as np


dates=[
"June 7, 2021",
"June 9, 2021",
"June 11, 2021",
"June 14, 2021",
"June 16, 2021",
"June 18, 2021",
"June 21, 2021",
"June 23, 2021",
```

```
"June 25, 2021",
"June 28, 2021",
"June 30, 2021",
"July 2, 2021",
"July 6, 2021",
"July 9, 2021",
"July 16, 2021",
"August 20, 2021",
"September 17, 2021",
"September 30, 2021",
"October 15, 2021",
"November 19, 2021",
"December 17, 2021"
,"December 31, 2021"
]

strikes={}
for date in dates:
    strikes[date]= info[date]["calls"]["Strike"]


longlist=[]
for date in dates:
    for i in strikes[date].tolist():
     longlist.append(i)
strikes=list(set(longlist))
strikes.sort()

#create matrix of value of call based on strike and time
Mat=np.ones([len(strikes),len(dates)])
for j in range(len(dates)):
    for i in range(len(strikes)):
        #print(strikes[i])
        if strikes[i] in list(info[dates[j]]["calls"]["Strike"]):
            ind=list(info[dates[j]]["calls"]["Strike"]).index(strikes
                                                    [i])
            Mat[i,j]=info[dates[j]]["calls"]["Last Price"][ind]
        else:
            Mat[i,j]=0
Mat2=[]
strikes2=[]
for i in range(len(strikes)):
    if sum(Mat[i,:]==0)<1:
        strikes2.append(strikes[i])
        Mat2.append(Mat[i,:])
strikes=strikes2

import QuantLib as ql
from math import pow, sqrt
import numpy as np
from scipy.optimize import root

expiration_dates = [
                    ql.Date(7,6,2021),
                    ql.Date(9,6,2021),
```

```
                    ql.Date(11,6,2021),
                    ql.Date(14,6,2021),
                    ql.Date(16,6,2021),
                    ql.Date(18,6,2021),
                    ql.Date(21,6,2021),
                    ql.Date(23,6,2021),
                    ql.Date(25,6,2021),
                    ql.Date(28,6,2021),
                    ql.Date(30,6,2021),
                    ql.Date(2,7,2021),
                    ql.Date(6,7,2021),
                    ql.Date(9,7,2021),
                    ql.Date(16,7,2021),
                    ql.Date(20,8,2021),
                    ql.Date(17,9,2021),
                    ql.Date(30,9,2021),
                    ql.Date(15,10,2021),
                    ql.Date(19,11,2021),
                    ql.Date(17,12,2021)
                    ,ql.Date(31,12,2021)
                    ]

data = np.transpose(Mat2)
data=data.tolist()

"""
Heston model calibration code usign Quantlib by Goutham Balaraman
https://gouthamanbalaraman.com/blog/heston-calibration-scipy-optimize
                                    -quantlib-python.html
accessed in date 20/05/2021
"""
day_count = ql.Actual365Fixed()
calendar = ql.UnitedStates()
calculation_date = ql.Date(6, 1, 2021)

spot = 420.04
ql.Settings.instance().evaluationDate = calculation_date

risk_free_rate = 0.01
dividend_rate = 0.0137
yield_ts = ql.YieldTermStructureHandle(
    ql.FlatForward(calculation_date, risk_free_rate, day_count))
dividend_ts = ql.YieldTermStructureHandle(
    ql.FlatForward(calculation_date, dividend_rate, day_count))



def setup_helpers(engine, expiration_dates, strikes,
                  data, ref_date, spot, yield_ts,
                  dividend_ts):
    heston_helpers = []
    grid_data = []
    for i, date in enumerate(expiration_dates):
        for j, s in enumerate(strikes):
```

```
            t = (date - ref_date )
            p = ql.Period(t, ql.Days)
            vols = data[i][j]
            helper = ql.HestonModelHelper(
                p, calendar, spot, s,
                ql.QuoteHandle(ql.SimpleQuote(vols)),
                yield_ts, dividend_ts)
            helper.setPricingEngine(engine)
            heston_helpers.append(helper)
            grid_data.append((date, s))
    return heston_helpers, grid_data

def cost_function_generator(model, helpers,norm=False):
    def cost_function(params):
        params_ = ql.Array(list(params))
        model.setParams(params_)
        error = [h.calibrationError() for h in helpers]
        if norm:
            return np.sqrt(np.sum(np.abs(error)))
        else:
            return error
    return cost_function

def calibration_report(helpers, grid_data, detailed=False):
    avg = 0.0
    if detailed:
        print ("%15s %25s %15s %15s %20s" % (
            "Strikes", "Expiry", "Market Value",
             "Model Value", "Relative Error (%)"))
        print ("="*100)

    for i, opt in enumerate(helpers):
        err = (opt.modelValue()/opt.marketValue() - 1.0)
        date,strike = grid_data[i]
        if detailed:
            print ("%15.2f %25s %14.5f %15.5f %20.7f " % (
                strike, str(date), opt.marketValue(),
                opt.modelValue(),
                100.0*(opt.modelValue()/opt.marketValue() - 1.0)))
        avg += abs(err)
    avg = avg*100.0/len(helpers)
    if detailed: print ("-"*100)
    summary = "Average Abs Error (%%) : %5.9f" % (avg)
    print (summary)
    return avg


def setup_model(_yield_ts, _dividend_ts, _spot,
                init_condition=(0.10,0.2,1,-0.5,0.1)):
    theta, kappa, sigma, rho, v0 = init_condition
    process = ql.HestonProcess(_yield_ts, _dividend_ts,
                           ql.QuoteHandle(ql.SimpleQuote(_spot)),
                           v0, kappa, theta, sigma, rho)
    model = ql.HestonModel(process)
    engine = ql.AnalyticHestonEngine(model)
```

```
    return model, engine
summary= []

model2, engine2 = setup_model(
    yield_ts, dividend_ts, spot,
    init_condition=(0.02,0.2,0.5,0.1,0.01))
heston_helpers2, grid_data2 = setup_helpers(
    engine2, expiration_dates, strikes, data,
    calculation_date, spot, yield_ts, dividend_ts
)
initial_condition = list(model2.params())


cost_function = cost_function_generator(model2, heston_helpers2)
sol = root(cost_function, initial_condition, method='lm')
theta, kappa, sigma, rho, v0 = model2.params()
print ("theta = %f, kappa = %f, sigma = %f, rho = %f, v0 = %f" % \
    (theta, kappa, sigma, rho, v0))
error = calibration_report(heston_helpers2, grid_data2)
summary.append(["Scipy LM1", error] + list(model2.params()))

from scipy.optimize import least_squares

model3, engine3 = setup_model(
    yield_ts, dividend_ts, spot,
    init_condition=(0.09,2,0.5,0.1,0.5))
heston_helpers3, grid_data3 = setup_helpers(
    engine3, expiration_dates, strikes, data,
    calculation_date, spot, yield_ts, dividend_ts
)
initial_condition = list(model3.params())


cost_function = cost_function_generator(model3, heston_helpers3)
sol = least_squares(cost_function, initial_condition)
theta, kappa, sigma, rho, v0 = model3.params()
print ("theta = %f, kappa = %f, sigma = %f, rho = %f, v0 = %f" % \
    (theta, kappa, sigma, rho, v0))
error = calibration_report(heston_helpers3, grid_data3)
summary.append(["Scipy LS1", error] + list(model3.params()))

from scipy.optimize import differential_evolution

model4, engine4 = setup_model(yield_ts, dividend_ts, spot)
heston_helpers4, grid_data4 = setup_helpers(
    engine4, expiration_dates, strikes, data,
    calculation_date, spot, yield_ts, dividend_ts
)
initial_condition = list(model4.params())
bounds = [(0,1),(0.01,15), (0.01,1.), (-1,1), (0,1.0) ]


cost_function = cost_function_generator(
    model4, heston_helpers4, norm=True)
sol = differential_evolution(cost_function, bounds, maxiter=100)
```

```
theta, kappa, sigma, rho, v0 = model4.params()
print ("theta = %f, kappa = %f, sigma = %f, rho = %f, v0 = %f" % \
    (theta, kappa, sigma, rho, v0))
error = calibration_report(heston_helpers4, grid_data4)
summary.append(["Scipy DE1", error] + list(model4.params()))



from scipy.optimize import basinhopping

class MyBounds(object):
     def __init__(self, xmin=[0.,0.01,0.01,-1,0], xmax=[1,15,1,1,1.0]
                                                    ):
         self.xmax = np.array(xmax)
         self.xmin = np.array(xmin)
     def __call__(self, **kwargs):
         x = kwargs["x_new"]
         tmax = bool(np.all(x <= self.xmax))
         tmin = bool(np.all(x >= self.xmin))
         return tmax and tmin
bounds = [(0,1),(0.01,15), (0.01,1.), (-1,1), (0,1.0) ]

model5, engine5 = setup_model(
    yield_ts, dividend_ts, spot,
    init_condition=(0.02,0.2,0.5,0.1,0.01))
heston_helpers5, grid_data5 = setup_helpers(
    engine5, expiration_dates, strikes, data,
    calculation_date, spot, yield_ts, dividend_ts
)
initial_condition = list(model5.params())


mybound = MyBounds()
minimizer_kwargs = {"method": "L-BFGS-B", "bounds": bounds }
cost_function = cost_function_generator(
    model5, heston_helpers5, norm=True)
sol = basinhopping(cost_function, initial_condition, niter=5,
                   minimizer_kwargs=minimizer_kwargs,
                   stepsize=0.005,
                   accept_test=mybound,
                   interval=10)
theta, kappa, sigma, rho, v0 = model5.params()
print ("theta = %f, kappa = %f, sigma = %f, rho = %f, v0 = %f" % \
    (theta, kappa, sigma, rho, v0))
error = calibration_report(heston_helpers5, grid_data5)
summary.append(["Scipy BH1", error] + list(model5.params()))

model5, engine5 = setup_model(
    yield_ts, dividend_ts, spot,
    init_condition=(0.07,0.5,0.1,0.1,0.1))
heston_helpers5, grid_data5 = setup_helpers(
    engine5, expiration_dates, strikes, data,
    calculation_date, spot, yield_ts, dividend_ts
)
initial_condition = list(model5.params())
```

```
mybound = MyBounds()
minimizer_kwargs = {"method": "L-BFGS-B", "bounds": bounds}
cost_function = cost_function_generator(
    model5, heston_helpers5, norm=True)
sol = basinhopping(cost_function, initial_condition, niter=5,
                   minimizer_kwargs=minimizer_kwargs,
                   stepsize=0.005,
                   accept_test=mybound,
                   interval=10)
theta, kappa, sigma, rho, v0 = model5.params()
print ("theta = %f, kappa = %f, sigma = %f, rho = %f, v0 = %f" % \
    (theta, kappa, sigma, rho, v0))
error = calibration_report(heston_helpers5, grid_data5)
summary.append(["Scipy BH2", error] + list(model5.params()))


from pandas import DataFrame
DataFrame(
    summary,
    columns=["Name", "Avg Abs Error","Theta", "Kappa", "Sigma", "Rho"
                                        , "V0"],
    index=['']*len(summary))
```

Chapter 9

Appendix C

The code is implemented in Python.

```
import numpy as np
import time

#number N of points
N=1000
#parameters
#theta is long run variance in Heston Model
theta=0.122251
#mean revert rate
kappa=4.996804
#correlation assume 0<=rho<=1
rho=0.5
#parameter in the diff eq solution; the square of it needs to be
                                    strictly less 2*k*theta aka Feller
                                     condition; it is vol of vol
e=0.849266
#Gamma is parameter in the utility function; it represents risk
                                    aversion
g=2
#max volatility
vmax=1.5
#min volatility
v1=0.001
#volatility discrete increments
dv=(vmax-v1)/(N)
#time increments
dt=0.0001
#lamdba is the excess expected return to return volatility.
l=0.3
#The risk-free rate of return
r=0.01
#is the discount factor for the utility;  it determines the fraction
                                    of wealth to spend and impatience
                                    of investor

delta=0.01
#creates the volatility vector
v=np.arange(v1,vmax+dv/2,dv)
```

```
#modified parameters
kprime=kappa-(g-1)*l*rho*e/g
thetaprime=theta*kappa/kprime

#mu prime
M=kprime*thetaprime-kprime*v
#test for Mu sign for later use
Mplus=M>0
Mminus=M<0
#
sigmaprime=(e**2)*v
#create diagonals
supdiag=Mplus[1:]*M[1:]/dv+0.5*sigmaprime[1:]/(dv**2)
subdiag=-M[:N]*Mminus[:N]/dv+0.5*sigmaprime[:N]/(dv**2)
#create the matrix S
S=np.diag(supdiag,+1)+np.diag(subdiag,-1)
supdiag=np.append(supdiag,0)
subdiag=np.append(0,subdiag)
maindiag=-supdiag-subdiag
S=S+np.diag(maindiag,0)
#inverted matrix for later
inv=np.linalg.inv(np.eye(N+1)-S*dt)

#create initial H guess
#want to go from 100 to 0.1 in N+1 steps
H=np.arange(100,0.1-(99.9/N),-99.9/N)
#H=np.ones([N+1])/delta
#set up testing for iteration
diff=100
test=0.0001
iteration=0
#the main iteration
while diff>test:
 #calculate H difference equation
 Hlong=np.append(H,0)
 Hlong=np.append(0,Hlong)
 Hprime=(Hlong[1:]-Hlong[:N+2])/dv
 #create K for each iteration
 #use vector form for computational speed
 HH=(Hprime[1:]*Mplus+Hprime[:N+1]*Mminus)/H
 K= delta/g+(1-1/g)*(r+0.5*g*v*(l**2-g**2*e**2*(1-rho**2)*(HH**2)))
 #store the last H
 Hprev=H
 K=1-K*dt
 RHS=K*H+dt
 #calculate new H in three steps
 H=inv.dot(RHS)
 #Format the new H to also have zeros at ends
 #check the diff between the two
 diff= max(abs(H-Hprev))
 #just to see the number of the current iteration as it goes
 iteration+=1
 print(iteration)
```

```
print('end')
phi=(l/g)-rho*e*(Hprime[:-1]/H)
intertemp=-rho*e*(Hprime[:-1]/H)


import matplotlib.pyplot as plt

plt.plot(v[1:],H[1:],'b')
plt.xlabel('volatility')
plt.ylabel('H(vt)')
plt.plot(v[1:],phi[1:]-intertemp[1:],'g')
plt.xlabel('volatility')
plt.ylabel('Myopic part')
plt.plot(v[1:],intertemp[1:],'y')
plt.xlabel('volatility')
plt.ylabel('Intertemporal part')
plt.plot(v[1:],phi[1:],'r')
plt.xlabel('volatility')
plt.ylabel('Phi(vt)')



import matplotlib.scale as mscale
import matplotlib.pyplot as plt
import matplotlib.transforms as mtransforms
import matplotlib.ticker as ticker
import numpy as np


###Adapted square root scale code
## Original from Stack Overflow "Square root scale using matplotlib/
                                    python"
###by michal_2am
class SquareRootScale(mscale.ScaleBase):
    """
    ScaleBase class for generating square root scale.
    """

    name = 'squareroot'

    def __init__(self, axis, **kwargs):
        # note in older versions of matplotlib (<3.1), this worked
                                            fine.
        mscale.ScaleBase.__init__(self)

        # In newer versions (>=3.1), you also need to pass in 'axis'
                                            as an arg
        #mscale.ScaleBase.__init__(self, axis)

    def set_default_locators_and_formatters(self, axis):
        axis.set_major_locator(ticker.AutoLocator())
        axis.set_major_formatter(ticker.ScalarFormatter())
        axis.set_minor_locator(ticker.NullLocator())
        axis.set_minor_formatter(ticker.NullFormatter())
```

```
    def limit_range_for_scale(self, vmin, vmax, minpos):
        return  max(0., vmin), vmax

    class SquareRootTransform(mtransforms.Transform):
        input_dims = 1
        output_dims = 1
        is_separable = True

        def transform_non_affine(self, a):
            return np.array(a)**0.5

        def inverted(self):
            return SquareRootScale.InvertedSquareRootTransform()

    class InvertedSquareRootTransform(mtransforms.Transform):
        input_dims = 1
        output_dims = 1
        is_separable = True

        def transform(self, a):
            return np.array(a)**2

        def inverted(self):
            return SquareRootScale.SquareRootTransform()

    def get_transform(self):
        return self.SquareRootTransform()

mscale.register_scale(SquareRootScale)

fig, ax = plt.subplots(1)

ax.plot(phi[1:], label='$Myopic$')
#ax.plot(intertemp[1:], label='$Intertemporal$')

ax.set_xscale('squareroot')
plt.xlabel('volatility (square root scale)')
plt.ylabel('Phi(vt)')
plt.tick_params(
        axis='x',
        which='both',
        labelbottom=False)
#ax.set_xticks(np.arange(0,20,0.1)**2)
#ax.set_xticks(np.arange(0,8.5,0.5)**2, minor=True)

plt.show()
```

Bibliography

[1] Robert C. Merton. Lifetime portfolio selection under uncertainty: The continuous-time case. *The Review of Economics and Statistics*, 51(3):247–257, 1969. pages 1, 19

[2] Robert C Merton. Optimum consumption and portfolio rules in a continuous-time model. *Journal of Economic Theory*, 3(4):373–413, 1971. pages 1

[3] Orazio P Attanasio. Frank ramsey's a mathematical theory of saving. *The Economic journal (London)*, 125(583):269–294, 2015. pages 1

[4] George M Constantinides. Habit formation: A resolution of the equity premium puzzle. *The Journal of political economy*, 98(3):519–543, 1990. pages 1

[5] John Y Campbell and John H Cochrane. By force of habit: A consumption-based explanation of aggregate stock market behavior. *The Journal of political economy*, 107(2):205–251, 1999. pages 1

[6] Maurice Obstfeld. Risk-taking, global diversification, and growth. *The American economic review*, 84(5):1310–1329, 1994. pages 2

[7] Harjoat S. Bhamra and Raman Uppal. The role of risk aversion and intertemporal substitution in dynamic consumption-portfolio choice with recursive utility. *Journal of Economic Dynamics and Control*, 30(6):967–991, 2006. pages 2, 24

[8] Gerald W. Recktenwald. Finite-difference approximations to the heat equation. pages 5, 29

[9] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, 1947. pages 6

[10] Mark Davis and Alison Etheridge. *Louis Bachelier's Theory of Speculation: The Origins of Modern Finance*. Princeton University Press, 2006. pages 8

[11] Robert C. Merton. *Continuous-time finance*. B. Blackwell, Cambridge, Mass. ;, rev. ed. edition, 1992. pages 8, 10

[12] Paul A. Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *The Review of Economics and Statistics*, 51(3):239–246, 1969. pages 8

[13] Agnes Tourin. An introduction to finite diffference methods for pdes in finance. 2011. pages 8

[14] Oldrich Vasicek. An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177–188, 1977. pages 9

[15] John Hull and Alan White. One-factor interest-rate models and the valuation of interest-rate derivative securities. *Journal of financial and quantitative analysis*, 28(2):235–254, 1993. pages 9

[16] Fischer Black and Piotr Karasinski. Bond and option pricing when short rates are lognormal. *Financial analysts journal*, 47(4):52–59, 1991. pages 9

[17] Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125–144, 1976. pages 9

[18] Di Venti Matteo Mario Ahmed Wasim. MLE estimation for Jump-diffusion processes. 2020. pages 9

[19] Kiyosi Itô. Stochastic integral. *Proceedings of the Imperial Academy*, 20(8):519 – 524, 1944. pages 9

[20] I. V. Girsanov. On transforming a certain class of stochastic processes by absolutely continuous substitution of measures. *Theory of Probability & Its Applications*, 5(3):285–301, 1960. pages 11

[21] M. Kac. On distributions of certain wiener functionals. *Transactions of the American Mathematical Society*, 65(1):1–13, 1949. pages 12

[22] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of financial studies*, 6(2):327–343, 1993. pages 13

[23] Hansjörg Albrecher, Philipp Mayer, Wim Schoutens, and Jurgen Tistaert. The little heston trap, 2006. pages 13

[24] Goutham Balaraman. Heston model calibration using quantlib python and scipy optimize. pages 13, 32

[25] Richard. Bellman. *Dynamic programming*. Research study (Rand Corporation). Princeton University Press, Princeton N.J, 1957. pages 16

[26] Agnes Tourin. An introduction to finite difference methods for pdes in finance. pages 18

[27] Guy Barles. *An Introduction to the Theory of Viscosity Solutions for First-Order Hamilton–Jacobi Equations and Applications*, pages 49–109. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. pages 18

[28] Eugene F. Fama and Kenneth R. French. The capital asset pricing model: Theory and evidence. *The Journal of Economic Perspectives*, 18(3):25–46, 2004. pages 19

[29] pages 19

[30] John C Cox, Jonathan E Ingersoll, and Stephen A Ross. An intertemporal general equilibrium model of asset prices. *Econometrica*, 53(2):363–384, 1985. pages 20

[31] Harjoat S. Bhamra. Monetary policy and asset risk phd: Lectures 1,2 and 3, 2015. pages 22

[32] Harjoat S. Bhamra. Derivatives: Assignment 3, part b, 2021. pages 23

[33] Tilman Klumpp and Kai A. Konrad. Sequential Majoritarian Blotto Games. Working Papers tax-mpg-rps-2018-05, Max Planck Institute for Tax Law and Public Finance, June 2018. pages 28

[34] Adam Solomon. Strategy and fundraising in sequential majoritarian elections. *ERN: Other Political Economy: Fiscal Policies Behavior of Economic Agents (Topic)*, 2018. pages 28

[35] Soheil Behnezhad, Sina Dehghani, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, and Saeed Seddighin. Faster and simpler algorithm for optimal strategies of blotto game, 2016. pages 28