David Šiška School of Mathematics Sciences, University of Edinburgh

2018/19 Semester 1

Object Oriented Programming with Applications

Problem Sheet 7 - Wednesday 14th November 2018[1]

1 Minimisation

Consider a function $f : \mathbb{R}^d \to \mathbb{R}$ that has a local minimum $x \in \mathbb{R}^d$ (local in some neighbourhood of x denoted D).

Our aim is find this local minimum numerically, from an initial guess $x_0 \in D$. We wish to use the BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno algorithm, see [1, Section 4.4] if you are interested in details). We will use this algorithm as a *black box* method by taking a library implementation provided by Alglib[2].

1.1 Code and exercise

You will need to download a template solution from the course website:

https://github.com/OOPA2018/Problem-sheets/tree/master/ProblemSheet7

Exercise 7.1. Open "VasicekCalibrationAndPricingEmpty.sln". Right-click on "MinimisationWithAlglib" project and select "Set as SetUp Project".

Modify **Example.cs** to find:

a) the minimum $x = (0, 0)^T$ of the function

$$f(x) := x_1^2 + x_2^2$$

when the algorithm is started from:

i) $x_0 = (-4, -3)^T$.
ii) $x_0 = (100, -100)^T$.

Has it converged? If not throw an appropriate exception or make this clear in program output.

b) the minimum $x = (1, \dots, 1)^T$ of the *Rosenbrock function*

$$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

when the algorithm is started from:

i) $x_0 = (3, -3, 3.3)^T$ (this fixes d to be equal to 3 here).
ii) $x_0 = (3, -3, 33000)^T$.

Has it converged? If not throw an appropriate exception or make this clear in program output.

[1] Last updated 14th November 2018

[2] See `http://www.alglib.net/` for completeness, no need to download anything from this website - download the package from the course website.

2 Minimisation and model calibration example

We will now use the above BFGS algorithm to calibrate a simple financial model. The model in question is the Vasicek short rate model and we will use prices of options on zero coupon bonds to calibrate the model.

Before doing any exercises please download the template solution from course website and read Section 2.4.

2.1 Interest rate derivatives under short rate models

The theory here is quite involved but for our purposes it boils down to a couple of formulæ. You will either see the theory and proofs in gory details the Risk-Neutral Asset Pricing course or you can read about it in e.g. Filipovic [2].

A zero coupon bond (ZCB) is a financial asset that pays one unit of currency at its maturity, which we denote T here[3]. Let $p(t,T)$ denote the price of a ZCB at some time $t \leq T$. Clearly $p(T,T) = 1$ since by definition a ZCB pays one unit of currency at maturity T. The question is: what is the price p for $t < T$.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. Let $(\mathcal{F}_t)$ be a filtration generated by some $\mathbb{R}$-valued Wiener process W. Let (r_t) be some stochastic process called the "short rate". Assume that there is a risk-free bank account that evolves like

$$dB_t = r_t B_t dt.$$

Then clearly

$$B_t = B_s \exp\left(\int_s^t r_u du\right).$$

It can be shown that under some "risk-neutral" measure $\mathbb{Q}$ the price of a ZCB is given by

$$p(t,T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r_u du}\middle|\mathcal{F}_t\right]. \tag{1}$$

A European call option on a ZCB is a derivative which gives holder the right (without any obligation) to buy a ZCB at a pre-agreed date $S \leq T$ (exercise date) for a pre-agreed strike price K. Let $v(t;S,K,T)$ denote the value of a ZCB option with exercise date S, strike K on a T-maturity ZCB. It can be shown that

$$v(t;S,K,T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^S r_u du}\left[p(S,T)-K\right]_+\middle|\mathcal{F}_t\right]. \tag{2}$$

In particular

$$v(0;S,K,T) = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^S r_u du}\left[p(S,T)-K\right]_+\right].$$

Other options (or more generally contingent claims) with exercise date S and payoff given by some square integrable and $\mathcal{F}_S$ measurable ξ then have time t value

$$\mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^S r_u du}\xi\middle|\mathcal{F}_t\right].$$

This suggests that we only need to know what the process (r_t) is. A popular *short rate* model is the *Vasicek model*.

[3] Despite the name and popularity in textbooks ZCBs are actually rarely traded. But they are building block for more complicated interest rate assets and hence they are worth studying.

2.2 Vasicek short rate model

Let $W^{\mathbb{Q}}$ be a Wiener process under the aforementioned risk-neutral measure $\mathbb{Q}$.

The Vasicek model assumes that (r_t) satisfies

$$dr(t) = k[\theta - r(t)]dt + \sigma dW^{\mathbb{Q}}(t).$$

It has a number of attractive properties including mean reversion and the fact that $r(t)$ can be negative with non-zero probability. Moreover under the model we have simple formulae for prices of ZCBs and European call / put options on ZCBs.

In particular the price of a ZCB (1) is just

$$P(t,T) = A(t,T)e^{-B(t,T)r(t)}, \tag{3}$$

where

$$A(t,T) = \exp\left(\left[\theta - \frac{\sigma^2}{2k^2}\right](B(t,T) - T + t) - \frac{\sigma^2}{4k}B(t,T)^2\right), \tag{4}$$

$$B(t,T) = \frac{1}{k}\left(1 - e^{-k(T-t)}\right). \tag{5}$$

The price of an European call option on a ZCB (2) is then just

$$v(t;S,K,T) = P(t,T)N(h) - KP(t,S)N(h - \sigma_p), \tag{6}$$

where $x \mapsto N(x)$ is the distribution (the cumulative density) function of a standard normal random variable and

$$\sigma_p := \sigma\sqrt{\frac{1 - e^{-2k(S-t)}}{2k}}B(S,T), \;\; h := \frac{1}{\sigma_p}\ln\frac{P(t,T)}{P(t,S)K} + \frac{\sigma_p}{2}. \tag{7}$$

Exercise 7.2. Write a C# class implementing prices of ZCBs and calls on ZCB options in the Vasicek model.

2.3 Calibration of Vasicek model to market prices

We will only be interested in (6) and (7) in the case when $t = 0$ for the purposes of calibration.

We will also assume that $r(0)$ is known (e.g. the EONIA or SONIA) rate. Thus we have the following model parameters to calibrate: k, θ and σ. We will need the prices of at least three financial derivatives (e.g. options on ZCBs) to calibrate these parameters. If we have fewer than there will be infinitely many choices of parameter values to match these market prices. If we have more than three then we can only hope to choose parameters that minimise some error function.

Note that the v given by (6) is in fact a function of k, θ and σ also. Thus

$$v(0;S,K,T) = v(0;S,K,T,k,\theta,\sigma).$$

Exercise 7.3. Assume that we observe the call option prices in the market given by Table 7.3.

Use the BFGS algorithm from previous exercise to find k, θ, σ which minimise the following function:

$$f(k,\theta,\sigma) = \sum_{i=1}^{N}\left[v_i - v(0;S_i,K_i,T_i,k,\theta,\sigma)\right]^2.$$

In other words we are minimising the *mean square error between the observed prices and the model prices*.

ZCB maturity T_i	Option exercise S_i	Strike K_i	Price v_i
0.5	0.25	0.991	0.00376
1	0.50	0.976	0.00922
1.5	0.75	0.957	0.01572
2	1.00	0.937	0.02291

Table 1: Market prices of call options on ZCBs.

2.4 Code structure

The Solution you download from the website contains of a lot of code and you only need to fill in parts of it. This is actually fairly representative of how things work when you work inside a company: you would rarely start a completely new and independent project.

There are five Projects inside the Solution.

- `AlgLib`: This project produces a dynamically linked library that contains the AlgLib code, in particular the BFGS algorithm.

  You do *not* need to modify this project.

- `MinimisationWithAlglib`: Template code used in the first exercise.

- `VasicekCalibrationAndPricing`: This project also produces a dynamically linked library. This is where you should complete Exercise 7.2 (the `VasicekModel` class) and Exercise 7.3 (the `VasicekCalibrator` class).

  The places to modify are where the `NotImplementedException` are thrown. You will need to add other appropriate methods and member variables.

- `VasicekCmdLine`: This project produces an executable console application. You can use it to test that the `VasicekModel` class and the `VasicekCalibrator` are behaving correctly.

  You will need to modify the `Examples` class a tiny bit. The place to modify is where the `NotImplementedException` is thrown.

To run with the command line interface: select `VasicekCmdLine`, go to the `Project` menu and choose `Set as StartUp Project`.

References

[1] J. F. Bonnans et. al., *Numerical optimization theoretical and practical aspects*. Springer, 2006.

[2] D. Filipovic, *Term-Structure Models: A Graduate Course*. Springer, 2009.