SDFPricing: A Julia Package for Asset Pricing Based on a Stochastic Discount Factor Process

Errikos Melissinos[*]

2024-04-24

Latest Draft

Abstract

I introduce a package in the Julia programming language to perform asset pricing based on a stochastic discount factor in continuous time. Prices are computed through Monte Carlo simulations according to a pricing partial differential equation and the corresponding Feynman-Kac formula. At this stage it is possible to compute a) prices of zero-coupon fixed income securities and b) price-dividend ratios, which also allow the calculation of prices and returns of these securities. The package is focused on ease of use and is meant to be used in research and teaching. I illustrate the functionality of the package with examples and an application. In particular, I show how asset prices react after shifts in economic variables within a consumption-based model, and I discuss to what extent these shifts can be classified as monetary policy shocks or information shocks in connection to monetary policy announcements.

JEL: C63, E43, G12

Keywords: Asset Pricing, Stochastic Discount Factor, Julia, Monte Carlo Simulations, Feynman-Kac Formula, Zero Coupon Bonds

[*]Goethe University Frankfurt & Leibniz Institute for Financial Research SAFE e.V. website: `https://errikos-melissinos.com`

1 Introduction

The stochastic discount factor (SDF) is a fundamental concept in asset pricing. Given a joint process for the SDF and a payoff stream, the corresponding security can be priced. However, there is little software publicly available that is easy to use and allows the calculation of prices based on an SDF. This paper showcases a package in the Julia programming language that I developed to answer this need. The package is called *SDFPricing* and is available at Github In this first release the package has limited functionality compared to other projects that are being developed for more than a decade. Nevertheless, it allows the pricing of zero-coupon fixed income securities, which could then also be combined to price dividend-paying securities. In addition, it is possible to specify a process for a stochastic dividend stream and calculate the price-dividend ratio, which can then be used to calculate the return of the security and its price given the current level of the dividend.[1] While the pricing can be made arbitrarily accurate given the processes of the SDF and the dividend stream, calibration to market data is not directly possible. Therefore, for the time being this tool will not by itself be useful to price derivatives or any security in a way that is consistent with other market prices.

I have chosen to develop this package in Julia for two main reasons. Firstly, Julia being a high-level programming language is easy to use. In addition, its syntax is similar to Python and MATLAB, which are widely used languages, and it is well suited for scientific programming. Secondly, Julia includes the *DifferentialEquations* package, which is arguably the most comprehensive tool for solving differential equations, including stochastic differential equations which are at the core of the pricing problem. Furthermore, there is particular interest in Julia from the finance and economics communities. For example, the QuantEcon Organisation is actively engaged in the language.

The asset pricing package that I introduce is the first within the context of the Julia programming language. In other programming languages, the Computational Finance Suite of MATLAB contains resources that can be used for asset pricing, but unfortunately it requires a license.[2] Apart from MATLAB there is also Quantlib which is a free and open-source C++ library that can be used for asset pricing, while

[1]For the time being the process of the dividend stream needs to obey a stochastic differential equation with no jumps.

[2]Information can be found on the official website `https://www.mathworks.com/solutions/computational-finance/computational-finance-suite.html`

its functionality can also be leveraged by a dedicated Python library.[3] Both Quantlib and the resources available in Matlab are tools tailored to the industry and they they do not focus on the pricing of assets starting from a stochastic discount factor. The aim of my package is to be the go-to resource for those who want to price assets given an SDF as an "input".

In Section 2 I describe the package and its functionality. A central function of the package is *solve*, which takes a variable of type *Problem* and a variable of type *SolutionSettings* and returns a variable of type *Solution*, which contains the solution to the pricing problem described by the inputs. The solution is calculated using Monte Carlo simulations and the Feynman-Kac formula.

In Section 3 I provide concrete examples of how to use the package along with the results. I use both consumption-based and non-consumption-based SDFs. I use SDFs that depend both on one state variable and two state variables, and I compute, zero-coupon bond prices, price-consumption ratios, and returns of dividend-paying securities.

In Section 4 I perform an application using the functionality of the package. In particular, I analyse the effect of general monetary policy shocks on interest rates and asset prices in the context of a simple consumption-based model. This analysis is motivated by the literature on Delphic and Odyssean shocks (Campbell, Evans, Fisher and Lustiniano 2012; Nakamura and Steinsson 2018; Jarociński and Karadi 2020; Andrade and Ferroni 2021; Altavilla, Brugnolini, Gürkaynak, Motto and Ragusa 2019), which are a classification of monetary policy surprises (Gürkaynak, Sack and Swansonc 2005; Kuttner 2001). Both kinds of shocks are caused by actions or announcements of central banks. However, Delphic shocks reveal information about the underlying state of the economy. In contrast, Odyssean shocks reveal information about the future conduct of monetary policy.[4] In the literature the type of shock is identified depending on how financial variables react after the monetary policy announcement. Using a simple consumption-based setup I analyse the possible reactions of interest rates and asset prices to monetary policy announcements. Importantly, monetary policy may not operate through one channel only. While a traditional

[3]Information can be found on the official website `https://www.quantlib.org`.

[4]The term Odyssean is due to the central bank committing to a future policy beforehand, as Odysseus did when he tied himself to the mast of his ship to avoid the Sirens. Originally, in Campbell *et al.* (2012) the emphasis was placed on whether the central bank is actually committed to the future policy or not. In subsequent literature the emphasis for the distinction between the two kinds of shocks was placed more on whether the shocks reveal important information about monetary policy actions as opposed to underlying macroeconomic conditions. In this papers Delphic shocks are identified with information shocks and Odyssean shocks are identified with monetary policy shocks.

monetary policy shock can be justified within a simple consumption-based model, my analysis shows that if monetary policy can also affect consumption volatility, then the response of financial variables can be much more unpredictable and it can give rise to movements in the short-term rate and in asset prices that are usually associated with an information shock. In addition, information shocks may also induce the effects that are associated with a standard monetary policy shock. In particular, this can happen, if the central bank *reveals* the kind of information about the consumption process that is usually assumed to *caused* by a standard monetary policy shock. These results suggest that accurate theoretical models may be required for the classification of monetary policy shocks.

Finally, Section 5 concludes.

2 Package Description

In the following description I assume a single state variable and a single Wiener process for simplicity of the formulas. However, it is also possible to have several state variables and several Wiener processes. The corresponding formulas are given in Appendix A.

2.1 Theory

2.1.1 Zero Coupon Bond

Following Cochrane (2009) the pricing equation for an asset that pays no dividends in continuous time can be written as:

$$\mathrm{E}\Big[\mathrm{d}\Big(\Lambda Q(x_t, t, m)\Big)\Big] = 0, \quad Q(x_t, t, 0) = g(t, x_t) \tag{1}$$

where m is the remaining maturity until the payoff is due, $g(\cdot)$ is a function for the terminal payoff for each time t, which can also depend on the state variable, but which typically is just equal to 1, Λ is the SDF and Q is the price function of the asset in terms of the state variables, current time and the remaining maturity of the asset. If the process for the SDF and the state variables is known, then this equation can be solved to derive the price function $Q(x_t, t, m)$ for any value of x, t, and m.[5] In

[5]Here, I show the dependence on time explicitly. Often it will be the case that there will be no dependence on time and the state variable and remaining maturity will be sufficient to determine the price of assets. The case without explicit time dependence is also easier to compute with the package.

particular, the state variables follow some process that can be written in stochastic differential equation (SDE) form as:

$$\mathrm{d}x = \mu(x_t, t)\mathrm{d}t + \sigma(x_t, t)\mathrm{d}W_t \tag{2}$$

where μ is the drift of the state variable, σ is the diffusion of the state variable, and W_t is the Wiener process. The process for the SDF can be written in SDE form as:

$$\frac{\mathrm{d}\Lambda}{\Lambda} = \mu_\Lambda(x_t, t)\mathrm{d}t + \sigma_\Lambda(x_t, t)\mathrm{d}W_t \tag{3}$$

Next, using Ito's Lemma, it is possible to express the SDE that the price function Q follows. This is given by:

$$\mathrm{d}Q = \left(\frac{\partial Q}{\partial t} - \frac{\partial Q}{\partial m} + \mu(x_t, t)\frac{\partial Q}{\partial x_t} + \frac{1}{2}\sigma(x_t, t)^2\frac{\partial^2 Q}{\partial x_t^2}\right)\mathrm{d}t + \sigma(x_t, t)\frac{\partial Q}{\partial x_t}\mathrm{d}W_t \tag{4}$$

where I have dropped the dependence of Q on the arguments for simplicity, and I have used that $\mathrm{d}m = -\mathrm{d}t$. This expression can now be inserted in Equation (1) to derive the pricing partial differential equation (SDE) that the price function Q follows:

$$\mathrm{E}\left[\frac{\mathrm{d}\left(\Lambda Q\right)}{\Lambda}\right] = 0 \Rightarrow \mathrm{E}\left[\frac{\mathrm{d}\Lambda}{\Lambda}Q + \mathrm{d}Q + \frac{\mathrm{d}\Lambda}{\Lambda}\mathrm{d}Q\right] = 0 \tag{5}$$
$$\Rightarrow -r(x_t, t)Q + \frac{\partial Q}{\partial t} - \frac{\partial Q}{\partial m} + \mu(x_t, t)\frac{\partial Q}{\partial x_t} + \frac{1}{2}\sigma(x_t, t)^2\frac{\partial^2 Q}{\partial x_t^2} + \sigma(x_t, t)\sigma_\Lambda\frac{\partial Q}{\partial x_t} = 0$$

where $r(x_t, t) = -\mathrm{E}[\mathrm{d}\Lambda/\Lambda]$ is the risk-free rate. This PDE can be solved using the Feynman-Kac formula, which states that the solution to the PDE is given by the expected value of the terminal payoff of the asset under the risk-neutral measure. This is given by:

$$Q(x_t, t, m) = \mathrm{E}\left[\exp\left(-\int_t^{t+m} r(\hat{x}_s, s)\mathrm{d}s\right)g(t+m, \hat{x}_{t+m})\middle|\hat{x}_t = x_t\right] \tag{6}$$

where $\hat{x}$ follows the modified process:

$$\mathrm{d}\hat{x}_t = \left(\mu(\hat{x}_t, t) + \rho_{cx}\sigma(\hat{x}_t, t)\sigma_\Lambda\right)\mathrm{d}t + \sigma(\hat{x}_t, t)\mathrm{d}W_t \tag{7}$$

This process has a modified process compared to the original process for the state variable. If the modification is equal to zero then the process for the state variable is the same as the original process, which implies that prices are set by risk-neutral investors (or equivalent to risk-neutral investors). If the modification is not equal to zero then there is a risk premium or a risk discount. By Monte Carlo simulations of the modified process in Equation (7), it is possible to compute the expectation in Equation (6), which gives the value of the zero-coupon bond. This allows to also derive the instantaneous return on bonds ($\mathrm{d}Q/Q$), which follows directly from Ito's Lemma as shown in Equation (4).

Finally, if the zero-coupon bond price is integrated up to infinity then this gives rise to the price of a perpetuity:

$$Z(x_t, t) \equiv \int_t^{\infty} Q(x_t, t, s)\mathrm{d}s \tag{8}$$

2.1.2 Price-Dividend Ratio

Based on the zero-coupon bonds it is also possible to derive the price-dividend ratio of a dividend-paying security. In particular, if at time t the price of the security is U_t and the dividend stream is D_t, then the price-dividend ratio is defined as:[6]

$$u_t(X_t) \equiv \frac{U_t}{D_t} \tag{9}$$

Where D_t follows the process:

$$\frac{\mathrm{d}D_t}{D_t} = \mu_D(x_t, t)\mathrm{d}t + \sigma_D(x_t, t)\mathrm{d}W_{Dt} \tag{10}$$

In order, to show the connection to zero-coupon bonds, I also define the dividend strip $Y_t(T)$, which pays an amount equal to the dividend of the security only at a specific time T ($Y_t(0) = D_t$). Then the strip price ratios are:

$$y_t(m) \equiv \frac{Y_t(m)}{D_t}, \quad y_t(0) \equiv 1 \tag{11}$$

By definition the price of the dividend paying security is the sum of the prices of the dividend strips:

$$U_t = \int_0^{\infty} Y_t(s)\mathrm{d}s \tag{12}$$

[6] Given the homotheticity of preferences, the price-dividend and price-consumption ratios are only functions of the state variable.

The price-dividend ratio of the dividend paying security is then:[7]

$$\frac{U_t}{D_t} \equiv u_t = \int_0^\infty \frac{Y_t(s)}{D_t}\mathrm{d}s = \int_0^\infty y_t(s)\mathrm{d}s \tag{13}$$

Given this setup we can again apply Equation (1) as before:[8]

$$\mathrm{E}\Big[\mathrm{d}\Big(\Lambda Y(x_t,t,m)\Big)\Big] = 0 \quad \forall t \in (t_0,T), \quad Y(x,t,0)=g(t) \quad \forall x \tag{14}$$

This can then be expressed in terms of the ratio y_t as (where I stop showing the explicit dependence on the arguments for simplicity):[9]

$$\begin{aligned}&\mathrm{E}\Big[\mathrm{d}\Big(\Lambda y_t D_t\Big)\Big] = 0\\ \Rightarrow &\mathrm{E}\left[\frac{\mathrm{d}\Lambda}{\Lambda}y + \mathrm{d}y + \frac{\mathrm{d}D}{D_t}y + \frac{\mathrm{d}\Lambda}{\Lambda}\mathrm{d}y + \frac{\mathrm{d}\Lambda}{\Lambda}\frac{\mathrm{d}D}{D}y + \frac{\mathrm{d}D}{D}\mathrm{d}y\right] = 0\end{aligned} \tag{15}$$

In addition, Ito's lemma also applies to the price-dividend ratio, giving rise to an expression similar to Equation (4):

$$\mathrm{d}y_t = \left(\frac{\partial y_t}{\partial t} - \frac{\partial y_t}{\partial m} + \mu(x_t,t)\frac{\partial y_t}{\partial x_t} + \frac{1}{2}\sigma(x_t,t)^2\frac{\partial^2 y_t}{\partial x_t^2}\right)\mathrm{d}t + \sigma(x_t,t)\frac{\partial y_t}{\partial x_t}\mathrm{d}W_{xt} \tag{16}$$

By inserting the expressions for y_t (Equation 16), D_t (Equation 10), and Λ_t (Equation 3) into Equation (15) we get:

$$\begin{aligned}&-r(x_t,t)y_t + \frac{\partial y_t}{\partial t} - \frac{\partial y_t}{\partial m} + \mu(x_t,t)\frac{\partial y_t}{\partial x_t} + \frac{1}{2}\sigma(x_t,t)^2\frac{\partial^2 y_t}{\partial x_t^2} + \mu_D(x_t,t)y_t\\ &+\rho_{cx}\sigma(x_t,t)\sigma_\Lambda\frac{\partial y_t}{\partial x_t} + \rho_{cD}\sigma_D(x_t,t)\sigma_\Lambda y_t + \rho_{xD}\sigma_D(x_t,t)\sigma(x_t,t)\frac{\partial y_t}{\partial x_t} = 0\end{aligned} \tag{17}$$

which is similar to Equation (5) but with additional terms that depend on the dividend process. This PDE can also be solved using the Feynman-Kac formula:

$$y_t(x_t,t,m) = \mathrm{E}\left[\exp\left(-\int_t^{t+m}\tilde{r}(\tilde{x}_s,s)\mathrm{d}s\right)g(t+m)\middle|\tilde{x}_t = x_t\right] \tag{18}$$

[7] A similar expression for discrete time is given in Wachter (2006).

[8] Now, I have adapted the notation as the price of the strip depends on the state variable and time explicitly.

[9] The derivation here is similar to Chen, Cosimano and Himonas (2010)

where

$$\tilde{r}(x_t,t) = r(x_t,t) - \mu_D(x_t,t) - \rho_{cD}\sigma_D(x_t,t)\sigma_\Lambda \tag{19}$$

is adjusted due to the extra terms coming from the dividend stream. And the process for the modified state variable is given by:

$$\mathrm{d}\tilde{x}_{it} = \Big(\sigma(\tilde{x}_{it},t)(\rho_{cx}\sigma_\Lambda + \rho_{xD}\sigma_D) + \mu(\tilde{x}_t,t)\Big)\mathrm{d}t + \sigma(\tilde{x}_t,t)\mathrm{d}W_{xt} \tag{20}$$

Using the ratios of the prices of the dividend strips over the current dividend for each maturity it is possible to integrate over all maturities to get the price-dividend ratio of the dividend-paying security as shown in Equation (13).

Finally, the return of the dividend-paying security can be computed:

$$\begin{aligned}
&\frac{\mathrm{d}U_t}{U_t} + \frac{D_t}{U_t}\mathrm{d}t = \frac{\mathrm{d}(u_tD_t)}{u_tD_t} + \frac{1}{u_t}\mathrm{d}t = \frac{\mathrm{d}u_t}{u_t} + \frac{\mathrm{d}D_t}{D_t} + \frac{\mathrm{d}u_t\mathrm{d}D_t}{u_tD_t} + \frac{1}{u_t}\mathrm{d}t \\
=&\frac{1}{u_t}\Bigg(\bigg(\frac{\partial u_t}{\partial t} + \mu(x_t,t)\frac{\partial u_t}{\mathrm{d}x_t} + \frac{1}{2}\sigma(x_t,t)^2\frac{\partial^2 u}{\partial x^2} + \mu_D(x_t,t)u_t + \frac{1}{2}\sigma_D(x_t,t)\sigma\frac{\partial u_t}{\partial x_t} + 1\bigg)\mathrm{d}t \\
&+\bigg(\sigma(x_t,t)\frac{\partial u_t}{\partial x_t} + \sigma_D(x_t,t)\bigg)\mathrm{d}W_t\Bigg)
\end{aligned} \tag{21}$$

Where Ito's Lemma has been applied to u_t.

2.2 Implementation

The implementation of the package follows closely the logic of the Feynman-Kac formula. In order to get the price of a zero-coupon security, the modified state variable(s) needs to be simulated and the values of these simulations are used to simulate the stochastic integral of the corresponding short rate.[10] The user needs to specify formulas for the drift and diffusion of the state variables, while the for the stochastic the r function needs to be given as a drift and 0 should be given as the diffusion. Each simulation of the stochastic integral is expressed as:[11]

$$\mathcal{I}_i \approx -\int_t^{t+m} \underbrace{r(\hat{x}_s)}_{\text{drift for simulation}} \mathrm{d}s, \quad i = 1, 2, \ldots, N_s \tag{22}$$

[10]In the case of the continuous payoff security the modified short rate should be given by the user as defined in Equation 19.

[11]The simulation of the following integral is handled internally by the StochasticDiffEq.jl package, as is the simulation of the state variables.

where N_s is the number of simulations used. Given a large number of samples the price of the security is approximated by:

$$Q(x_t,t,m) = \mathrm{E}\left[\exp\left(-\int_t^{t+m} r(\hat{x}_s)\mathrm{d}s\right)\right]g(t+m,x_{t+m}) \approx \frac{1}{N_s}\sum_{i=1}^{N_s}\exp\left(\mathcal{I}_i\right)g(t+m,x_{t+m}) \tag{23}$$

This becomes a good approximation for a large enough number of simulations.

The main function of the package, which performs the computation above is *solve* which takes as input a variable of type *Problem* and a variable of type *SolutionSettings*. It then returns a variable of type *Solution*. The *Problem* type contains the information for the drift and the diffusion of the processes to be simulated, it also contains the terminal function for the payoff (typically just equal to one at maturity). The drift and diffusion functions for the processes that are meant to be simulated are given as they would be for the standard DifferentialEquations package in Julia.[12]. The *SolutionSettings* type contains information that is necessary for the solution of the problem, such as the grid for the state variable, the number of simulations to be performed, and the algorithm to be used for the simulations.[13] This type can also be given a specification that a continuous payoff variable is being simulated. In this case, prices of dividend strips are computed[14] and they are then also integrated to give the price-dividend ratio. Finally, the *Solution* type returns the result of the computation that can be called as a normal function. This means that it can be called with specific arguments for time and the state variable(s) to return the price of the zero-coupon security or the price-dividend ratio of the dividend-paying security, if a continuous payoff specification is given.[15]

Finally, a convenience function is also given to compute the derivatives of the price-dividend ratio with respect to the state variable. This can facilitate the computation of the return of the dividend-paying security as shown in Equation (21).[16]

A more detailed technical description of the package is provided in Appendix ref ??, while the following examples illustrate how the package can be used.

[12]Or the more specialised package StochDiffEq.jl for stochastic differential equations.

[13]The algorithm is one of the algorithms offered in the standard DifferentialEquations.jl package. More information can be found in the documentation page.

[14]From the point of view of the code these are exactly the same as zero-coupon securities. The difference should be in the drift of the given process, that should be modified as specified in 13.

[15]The *Solution* type is further subdivided into *SinglePayoffSolution* and *ContinuousPayoffSolution*.

[16]At the point of writing this function can only applies when the problem has one state variable.

3 Examples

3.1 One State Variable

3.1.1 Time-Varying Consumption Drift – Zero-Coupon Security

While the package can be used with any process for theSDF, the examples in this paper are from the context of a consumption-based model, in which the investor has CRRA utility. The specific code and all the results for the examples are shown in Jupyter notebooks that are included in Appendix C.[17] The consumption process is exogenous and given by:

$$\mathrm{d}\log C_t = \mathrm{d}c_t = \mu_c(x_t)\mathrm{d}t + \sigma_c \mathrm{d}W_t, \quad \mu_c(x_t) = \mu_{c0} + x_t \tag{24}$$

By Ito's Lemma and the fact that $\Lambda = e^{-\rho t}C^{-\gamma}$ the process for the SDF is given by:

$$\frac{\mathrm{d}\Lambda}{\Lambda} = \Big(-\rho - \gamma\mu_c(x_t) + \frac{1}{2}\gamma^2\sigma_c^2\Big)\mathrm{d}t - \gamma\sigma_c \mathrm{d}W_t \tag{25}$$

And this also provides the function for the short rate:

$$r(x_t) = -\mathrm{E}\left[\frac{\mathrm{d}\Lambda}{\Lambda}\right]\frac{1}{\mathrm{d}t} = \rho + \gamma\mu_c(x_t) - \frac{1}{2}\gamma^2\sigma_c^2 \tag{26}$$

The process for the state variable is given by:

$$\mathrm{d}x = -\log\phi(\bar{x} - x_t)\mathrm{d}t + \sigma_x \mathrm{d}W_t \tag{27}$$

where $\bar{x}$ is the point at which the process has a drift of zero, which I also call the *stochastic steady state*. So, based on Equation (7), the process for the modified state variable is:

$$\mathrm{d}\hat{x} = \underbrace{\Big(-\log\phi(\bar{x} - \hat{x}_t) - \gamma\rho_{cx}\sigma_x\sigma_c\Big)}_{\text{drift for simulation}}\mathrm{d}t + \underbrace{\sigma_x}_{\text{diffusion for simulation}}\mathrm{d}W_t \tag{28}$$

Apart from computing the regular price of the zero-coupon bond, I also compute the price of the risk-neutral zero-coupon bond. In this case I simulate the unmodified state variable of the problem. Then the term premium can be computed as the difference between the yield and the risk-neutral yield. As can be seen in the results,

[17]These examples are also included as part of the code of the package to facilitate users.

the term structure can be either upward or downward-sloping (positive or negative yield spread), depending on the value of the state variable. This is because the state variable is expected to revert to the stochastic steady state. So, long-term yields which are in some respect combinations of expected future short-rates are higher (lower) compared to short-term yields, when short rates are expected to increase (decrease). In addition, within this stylised model, even though consumption drift is significantly variable, the term premium is negative and tiny.

3.1.2 Time-Varying Consumption Diffusion - Zero-Coupon Security

In this example, instead of having a time-varying consumption drift as in the previous example, I have a time-varying consumption diffusion. The consumption process is given by:

$$\mathrm{d}\log C_t = \mathrm{d}c_t = \mu_c \mathrm{d}t + \sigma_c(x_t)\mathrm{d}W_t, \quad \sigma_c(x_t) = \begin{cases} \frac{2\sigma_{c0}}{1+\exp(-2x)} & \text{if } x < 0 \\ \frac{4\sigma_{c0}}{1+\exp(-x)} - 1 & \text{otherwise} \end{cases} \tag{29}$$

while the state variable follows the same process as in the previous example. The process could have also followed a CIR process and then be used as the consumption diffusion. This would also ensure that the consumption diffusion is positive. However, I use this relatively different process to show that the package can handle processes that are non-affine in terms of the state variable. In addition, I avoid using a simple exponential that would ensure the positivity of the consumption diffusion, because the exponential can increase too fast and make the some paths unstable. The functional form above ensures that the consumption diffusion is bounded between 0 and $3\sigma_{c0}$, where σ_{c0} is the value at the stochastic steady state. The modified state is then given by:

$$\mathrm{d}\hat{x} = \Big(- \log\phi(\bar{x} - \hat{x}_t) - \gamma\rho_{cx}\sigma_x\sigma_c(\hat{x}_t)\Big)\mathrm{d}t + \sigma_x \mathrm{d}W_{ct} \tag{30}$$

As can be seen in the results, the short term rate is slightly decreasing with consumption volatility due to the precautionary savings motive of agents and the term premium is negative and also very small.

3.1.3 Time-Varying Consumption Drift - Dividend-Paying Security

In this example, I show how the package can be used to compute the price-dividend ratio of a dividend-paying security for the same consumption process as in the first example. In general, it is possible for the dividend to follow any process. However, here I assume that the dividend process follows the same process as the consumption process. In this case the price-dividend ratio is called price-consumption ratio, and dividend strips are called consumption strips. Here, the modified process is not the same as in the case with a zero-coupon bond. Following Equation (20) the modified process is given by:

$$\mathrm{d}\tilde{x} = \underbrace{\Big(- \log \phi(\bar{x} - \tilde{x}_t) - (\gamma - 1)\rho_{cx}\sigma_x\sigma_c \Big)}_{\text{drift for simulation}} \mathrm{d}t + \underbrace{\sigma_x}_{\text{diffusion for simulation}} \mathrm{d}W_{ct} \tag{31}$$

Unlike the case of the zero-coupon bond for the consumption strip the short rate function that is used in the simulation is also modified. So, following Equation (19) the modified short rate is given by:

$$\tilde{r}(\tilde{x}_t, t) = \rho + \gamma\mu_c(\tilde{x}_t) - \frac{1}{2}\gamma^2\sigma_c^2 - \mu_c(\tilde{x}_t) + \gamma\sigma_c^2 \tag{32}$$

The results show that when consumption drift is significantly varying the price of the consumption perpetuity is significantly volatile. In addition, it is possible to use the resulting price-consumption ratio calculate its derivatives and then use these to get the return of the security as a function of the state variable.[18] The return of the security is very close to the short term rate verifying the idea behind the equity premium puzzle. In particular, this model would predict a very small equity premium, which is not consistent with the data.

3.1.4 Time-Varying Consumption Diffusion - Dividend-Paying Security

The dividend paying security can also be computed when consumption diffusion is time-varying. Again I assume that the dividend is following the same process as consumption. So, the modified process for the state variable is given by:

$$\tilde{r}(\tilde{x}_t, t) = \rho + \gamma\mu_c - \frac{1}{2}\gamma^2\sigma_c(x_t)^2 - \mu_c + \gamma\sigma_c(x_t)^2 \tag{33}$$

Interestingly for $\gamma = 2$ the function above becomes a constant which makes the price-consumption ratio also a constant. This implies that holding consumption constant and changing consumption volatility actually has no effect on prices and returns. However, the short rate is still a decreasing function of consumption volatility so the premium is increasing in the consumption diffusion.

[18] Here, the derivatives work best when the solution is computed based on an interpolation function that is calculated based on the DataInterpolations package.

3.2 Two State Variables

Finally, the package also works with more than one state variables. In this example, I allow consumption drift and consumption diffusion to vary independently. So, the consumption process is given by:

$$\mathrm{d}c_t = \mu_c(x_{1t})\mathrm{d}t + \sigma_c(x_{2t})\big(1 - |\rho_{cx1}| - |\rho_{cx2}|\big)\mathrm{d}W_{c1t} + \rho_{cx1}\sigma_c(x_{1t})\mathrm{d}W_{x1t} + \rho_{cx2}\sigma_c(x_{2t})\mathrm{d}W_{x2t} \tag{34}$$

Where $\mu_c(\cdot)$ and $\sigma_c(\cdot)$ are the same as in Subsections 3.1.1 and 3.1.2 respectively. And the processes of the state variables are given by:

$$\begin{aligned}
\mathrm{d}x_{1t} &= -\log\phi_1(\bar{x}_1 - x_{1t})\mathrm{d}t + \sigma_{x1}\frac{1}{1+\rho_{12}}\mathrm{d}W_{x1t} + \sigma_{x1}\frac{\rho_{12}}{1+\rho_{12}}\mathrm{d}W_{x2t} \\
\mathrm{d}x_{2t} &= -\log\phi_2(\bar{x}_2 - x_{2t})\mathrm{d}t + \sigma_{x2}\frac{\rho_{21}}{1+\rho_{21}}\mathrm{d}W_{x1t} + \sigma_{x1}\frac{\rho_{21}}{1+\rho_{21}}\mathrm{d}W_{x2t}
\end{aligned} \tag{35}$$

W_{ct}, W_{x1t}, and W_{x2t} are independent Wiener processes, but based on the structure above the correlations between the various components are:

$$\begin{aligned}
\mathrm{E}[\mathrm{d}c_t\mathrm{d}x_{1t}] &= \left(\rho_{cx1}\frac{1}{1+\rho_{12}} + \rho_{cx2}\frac{\rho_{12}}{1+\rho_{12}}\right)\sigma_c(x_{2t})\sigma_{x1}\mathrm{d}t \approxeq \rho_{cx1}\sigma_c\sigma_{x1}\mathrm{d}t \\
\mathrm{E}[\mathrm{d}c_t\mathrm{d}x_{2t}] &= \left(\rho_{cx2}\frac{1}{1+\rho_{21}} + \rho_{cx1}\frac{\rho_{21}}{1+\rho_{21}}\right)\sigma_c(x_{2t})\sigma_{x2}\mathrm{d}t \approxeq \rho_{cx2}\sigma_c(x_{2t})\sigma_{x_2}\mathrm{d}t \\
\mathrm{E}[\mathrm{d}x_{1t}\mathrm{d}x_{2t}] &= \sigma_{x1}\sigma_{x2}\frac{\rho_{21}+\rho_{12}}{1+\rho_{12}+\rho_{21}+\rho_{12}\rho_{21}}\mathrm{d}t \approxeq (\rho_{12}+\rho_{21})\sigma_{x1}\sigma_{x2}\mathrm{d}t
\end{aligned} \tag{36}$$

with the approximate equalities being valid when ρ_{12} and ρ_{21} are small. The benefit of this setup is that consumption diffusion is equal to $\sigma_c(\hat{x}_{2t})$, the diffusion of the state variables is close to σ_{x1} and σ_{x2}, when ρ_{12} and ρ_{21} are small, and a correlation structure can still be maintained between the consumption process and the state variables by an appropriate choice of parameters ρ_{cx1}, ρ_{cx2}, ρ_{12}, and ρ_{21}. For example, a negative correlation between consumption and consumption diffusion can be specified by letting ρ_{cx2} be negative, or a correlation between the two state variables can be specified without introducing correlation with consumption by letting ρ_{12} and ρ_{21} be different from zero. Similar to above the modified process is given by:

$$\begin{aligned}
\mathrm{d}\hat{x}_{1t} &= \big(-\log\phi_1\cdot(\bar{x}_1 - \hat{x}_{1t}) + \rho_{cx1}\sigma_c(x_t)\sigma_{x1}\big)\mathrm{d}t + \sigma_{x1}\frac{1}{1+\rho_{12}}\mathrm{d}W_{x1t} + \sigma_{x1}\frac{\rho_{12}}{1+\rho_{12}}\mathrm{d}W_{x2t} \\
\mathrm{d}\hat{x}_{2t} &= \big(-\log\phi_2\cdot(\bar{x}_2 - \hat{x}_{2t}) + \rho_{cx2}\sigma_c(x_t)\sigma_{x2}\big)\mathrm{d}t + \sigma_{x2}\frac{\rho_{21}}{1+\rho_{21}}\mathrm{d}W_{x1t} + \sigma_{x2}\frac{1}{1+\rho_{21}}\mathrm{d}W_{x2t}
\end{aligned} \tag{37}$$

Finally, the short rate is unmodified and a function of two variables:

$$r(x_{1t}, x_{2t}) = -\mathrm{E}\left[\frac{\mathrm{d}\Lambda}{\Lambda}\right]\frac{1}{\mathrm{d}t} = \rho + \gamma\mu_c(x_{1t}) - \frac{1}{2}\gamma^2\sigma_c(x_{2t})^2 \tag{38}$$

Seen as a function of one state variable at a time the results are not significantly different compared to the examples before. However, the two-variable model can be used to show further moments including cross moments between different financial variables.

4 Application

The package allows the computation of asset prices in response to changes in the state variable and/or in consumption. As also mentioned in the introduction, the literature has focused on two kinds of shocks, Delphic and Odyssean. Delphic shocks reveal information about the underlying state of the economy, and Odyssean shocks introduce an unexpected monetary policy. Performing an analysis in the context of a consumption-based model highlights the fact that interest rates and asset prices ultimately only move when some component of the SDF changes (or is perceived to change). This holds regardless whether the central bank is revealing information or whether it is committing to a different monetary policy. In addition, the analysis highlights the importance of the channel through which monetary policy is conducted. For example, a standard monetary tightening is supposed to decrease output and this increases expected output growth as the economy is expected to revert to the steady state. This implies that the real short-term rate increases to counteract the increased consumption smoothing motive, and asset prices fall due to the higher discount rate. This literature makes the assumption that on the day of a monetary policy announcement the announcement itself is causing the changes in asset prices and not vice versa. And while this is reasonable and in most cases should be true, the observation of an increase in interest rates and a decrease in asset prices does not necessarily imply that the central bank has *caused* this effect by tightening monetary policy, if the central bank could just be revealing information. Indeed the central bank could be directly revealing that output growth has increased for other reasons (in the same way that it would have increased had the actual monetary policy changed), and this by definition should produce exactly the same response of interest rates and asset prices. So, even when we observe the “correct” pattern for monetary policy we cannot exclude the possibility of a Delphic/information shock.

Still one could claim that when the "wrong" pattern occurs, then it *is* due to a Delphic shock. However even in this case, monetary policy could be affecting the economy through different channels, even before the episodes of explicitly unconventional monetary policy. In this section I explore the behaviour of asset prices under when different components of the SDF change, and my results suggest that, whatever the pattern, it is not trivial to classify as Odyssean/monetary shocks and Delphic/information shocks. I analyse two main cases, in the first the state variable is consumption drift and in the second it is consumption diffusion. One could ask whether monetary policy can affect consumption diffusion. And while this channel is less standard, such a relationship can find support in at least two different strands of literature. Firstly, there is literature suggesting the importance of the "risk-taking channel" of monetary policy (Borio and Zhu 2012; Adrian and Shin 2010), and such a channel could be modelled as affecting consumption diffusion in the context of a consumption-based model.[19] Secondly, Vayanos and Vila (2021) has also suggested that the term structure of interest rates is driven by arbitrageurs taking on more or less risk. In their model, this would directly translate to more or less wealth volatility, which can be naturally modelled as consumption volatility within a model that has exogenous consumption.[20]

In the following cases that I examine, I will compare my results to the classification of monetary policy announcements in Jarociński and Karadi (2020) (JK) and Cieslak and Schrimpf (2019) (CS), which I summarize in Table 1. In JK the co-movement of stock prices with the short-term rate is exclusively considered as a criterion to classify shocks into pure monetary policy shocks and information shocks. In the case of CS the yields of long-term bonds are also considered, and essentially the difference is that information shocks are sub-categorised into "growth" and "risk premium" shocks, in the former (latter) short-term (long-term) yields react more aggressively than long-term (short-term) yields. The calibration for both cases is shown in Table

[19]A different approach within a consumption-based model would be to assume that the risk aversion parameter itself can stochastically change as in Lettau and Wachter (2011).

[20]In the original model there is no consumption and arbitrageurs are just optimising the mean and variance of their portfolio value. In particular, arbitrageurs are rather not associated with real consumers but with financial institutions. Here, I use a model with consumption without any explicit wealth, but this could be thought of as modelling either the behaviour of real consumers or the behaviour of financial institutions that use the a consumption-based SDF. While Vayanos and Vila (2021) is mostly associated with the conduct of unconventional monetary policy, it suggests a more general explanation of the term structure. So, assuming that monetary policy has played an important role in shaping the term structure of interest rates even before the introduction of unconventional monetary policies, it is arguable that arbitrageurs would be adjusting their levels of risk even before the introduction of unconventional monetary policies.

2.

	Shock	Yields		Stocks	Stock-Yield
		short	long		Co-movement
Jarociński and Karadi (2020)	Monetary policy:	$\big\uparrow$	-	$\big\downarrow$	-
	Information:	$\big\uparrow$	-	$\big\uparrow$	+
Cieslak and Schrimpf (2019)	Monetary policy:	$\big\uparrow$	$\uparrow$	$\big\downarrow$	-
	Growth:	$\big\uparrow$	$\uparrow$	$\big\uparrow$	+
	Risk premium:	$\downarrow$	$\big\downarrow$	$\big\downarrow$	+

Table 1: This table is partly taken from a corresponding table in Cieslak and Schrimpf (2019), to which I have added the classification in Jarociński and Karadi (2020). In the latter paper the authors do not use long-term yields in the classification and they do not compare the size of the movements. The former paper uses both kinds of yields and compares the size of the movements. This is expressed through the size of the arrows in the table.

4.1 Case 1: Time-Varying Consumption Drift

In the first case that I analyse, the underlying model has a time-varying consumption drift. And the central bank is also able to "externally" affect the variables in the model. I assume that this takes place without adding an extra state variable.[21] So, the processes evolve according to:

$$
\begin{aligned}
\mathrm{d}x_t &= -\log\phi(x_t - x_t)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt} + \mathcal{M}_x \mathrm{d}q_t \\
\mathrm{d}c_t &= (\mu_{c0} + x_t)\mathrm{d}t + \sigma_c \mathrm{d}W_t + \mathcal{M}_c \mathrm{d}q_t
\end{aligned}
\tag{39}
$$

Where $\mathrm{d}q_t$ is a Poisson shock assumed to be activated when the monetary policy announcement occurs. $\mathcal{M}_x$ and $\mathcal{M}_c$ express the size of the effect on the state variable and consumption respectively.[22] I focus on the effect of the effect of the shock on

[21]Technically, the monetary policy variable should also have a distribution. However, for the stylized model I use here, I just assume that monetary policy can affect the economy in an unanticipated way. A different approach would be to introduce monetary policy as a separate state variable or assume that the state variable is already equivalent to monetary policy. The results would not be significantly different.

[22]For mathematical consistency with the rest of the model, as also mentioned in the previous footnote, $\mathrm{d}q_t$ should have 0 intensity, which implies that the probability of such a change is equal to 0.

Parameter	Model	
	Time-varying consumption drift	Time-varying consumption diffusion
γ	1/2	1/2/2.5
ϕ	0.82	0.82
$\bar{x}$	0.0	0.0
ρ	0.02	0.02
μ_{c0}	0.01	-
μ_c	state variable	0.01
σ_{c0}	-	0.06
σ_c	0.01	state variable
σ_x	0.005	0.5
ρ_{cx}	0.3	0.3

Table 2: Calibration for the two cases

yields and on an asset that pays dividends equal to consumption. I call this asset *consumption perpetuity.*

If the announcement only affects the current level of consumption ($\mathcal{M}_x = 0, \mathcal{M}_c \neq 0$), then the short-term rate is unaffected, the price consumption ratio is unaffected, but the price of the consumption perpetuity undergoes a level shift that persists in time.[23] This is because there is only one state variable in the model other than consumption, which by itself does not affect prices and the price-consumption ratios. In addition, the jump in consumption does not dissipate given the process chosen for consumption, which explains the persistence of the effect on the price of the consumption perpetuity. In a more realistic case in which the announcement affects both variables ($\mathcal{M}_x > 0, \mathcal{M}_c < 0$) the results are shown in Figure 1. For both risk aversion values the short-term rate increases, while longer-term yields increase also but less with maturity due to the mean reversion of the steady state. In addition, in both cases there is no significant change in the equity premium. While this change

[23]When discussing how monetary policy announcements affect variables, this could be taking place either by monetary policy literally affecting the variable or by revealing information and making investors aware that a variable has some value. For simplicity I do not explicitly and separately model perceived and real variables.

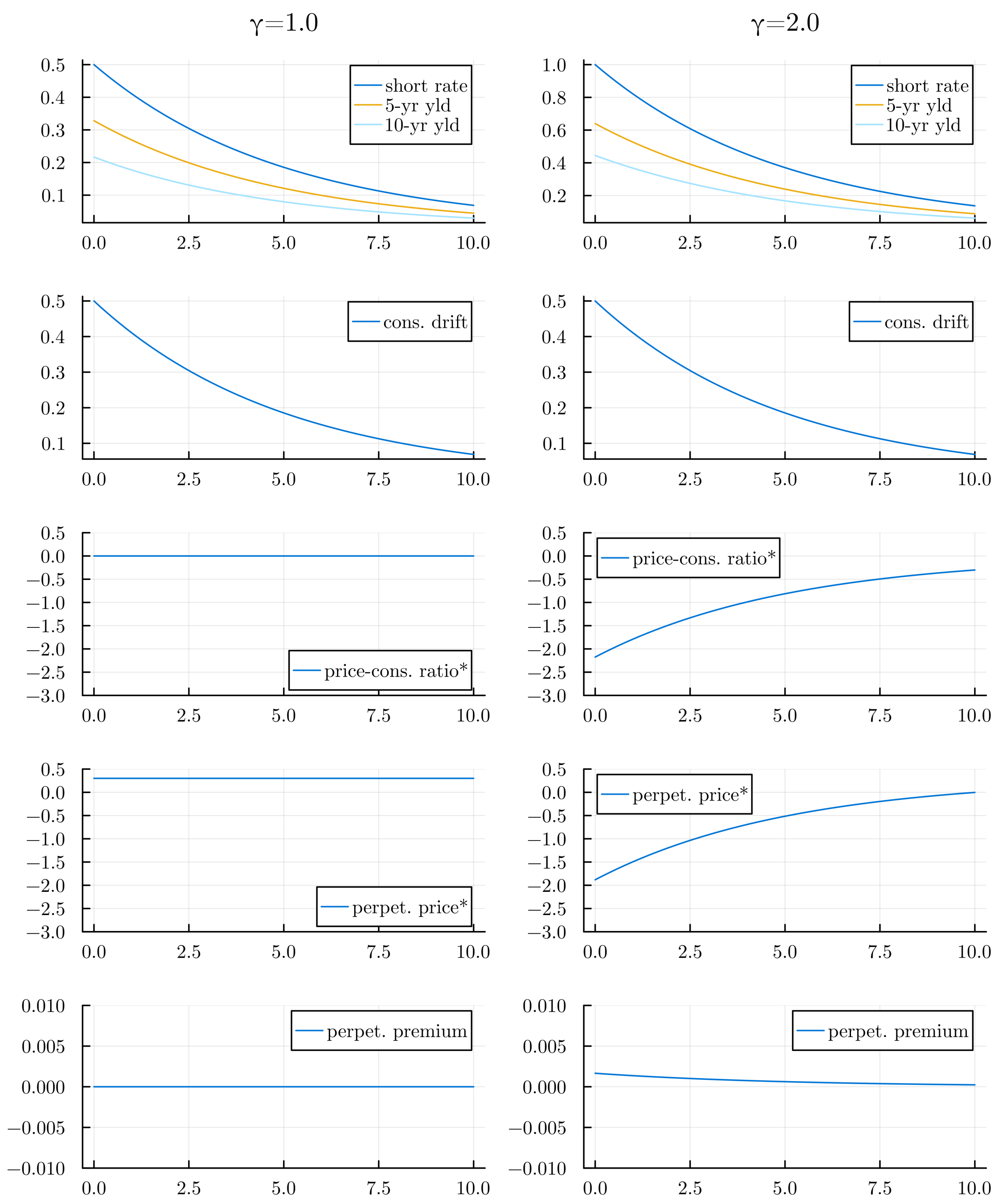


Figure 1: Impulse Responses to Consumption Drift Shock
These are responses after a consumption drift change. The period shown is ten years. The size of the shock corresponds to one standard deviation. I interpret these responses as taking place after a monetary policy announcement, and I assess to which type of shock they correspond. The short-term rate increases, but the response of the asset price is different depending on the risk aversion parameter.
*Plots normally show percent deviation from the stochastic steady state. Plots with starred variables show percent relative deviations from the stochastic steady state.

look more like a traditional monetary policy shock, it turns out that for the special case of $\gamma = 1$ the price-consumption ratio is constant as the higher discounting exactly offsets the higher expected dividends. The price of the asset still undergoes a level increase due to the increase in the dividends, which do not revert to the previous level. When $\gamma = 2.0$ the traditional pattern of monetary policy arises, in which the short rate goes up and the asset price goes down, with both effects subsequently dying out.[24] This is consistent with the monetary policy classification in both JK and CS. Nevertheless, these results highlight that ultimately the effects occur in connection to a shift in the consumption drift. So, if the central bank can affect the economy through an information channel, then the shift could be *caused* or alternatively it could be *revealed* by the central bank. Therefore, even in this conventional monetary policy case, a method is required to distinguish between the two alternatives.

4.2 Case 2: Time-Varying Consumption Diffusion

For the second case, I work on the model with time-varying consumption diffusion. Again I focus on yields and on the consumption perpetuity. Here the processes evolve according to:

$$\begin{aligned} \mathrm{d}x_t &= -\log\phi(x_t - x_t)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt} + \mathcal{M}_x \mathrm{d}q_t \\ \mathrm{d}c_t &= \mu_c \mathrm{d}t + \sigma_c(x_t)\mathrm{d}W_t + \mathcal{M}_c \mathrm{d}q_t \end{aligned} \tag{40}$$

with σ_c defined in Equation (29). The case where only consumption increases are the same as were discussed before. The effect of the monetary policy announcement affecting an increase in consumption volatility ($\mathcal{M}_x > 0, \mathcal{M}_c = 0$) is shown in Figure 2. Here, I use a higher average consumption volatility compared to the previous case, in order for consumption volatility changes to have a significant effect on the short rate and on returns. I also use a positive correlation between consumption volatility changes and consumption changes as was primarily done in Melissinos (2023).[25] For all three values of the risk aversion parameter ($\gamma = 1.0, 2.0.2.5$), the short-term rate decreases albeit with different intensity. This would be associated with an easing of

[24]Even in this case though there is a lasting residual effect on the price of the dividend-paying security that is due to the increase of the dividend which in this model does not revert to the previous level. There is also a tiny increase in the equity premium, significantly less than 0.01%, as the equity premium depends on the state variable, but only very slightly given that consumption volatility is constant.

[25]For this analysis using negative correlation would produce practically the same impulse response functions.

monetary policy. In addition, despite consumption volatility and the equity premium rising considerably in all three cases, the shock is never classified as a risk premium shock of CS because long-term yields react less than short-term yields. This is due to the reversion of the state variable to the steady state, which implies that over the long run, the state variable should revert to its previous value. In addition, consistent with an information shock in JK and a growth shock in CS the price of the asset decreases when $\gamma = 1.0$ when the short-term rate also decreases.[26] Interestingly, this has nothing to do with a growth shock conceptually as by construction only consumption volatility is affected. On the contrary, the case of a growth shock (shift in consumption drift) was analysed in the previous subsection, and it had the effect of a monetary policy shock according to the classifications. Furthermore, there are two more cases for the risk aversion parameter that would imply a different classification. For $\gamma = 2.0$ the price of the asset remains constant, which is an alternative not considered by JK or CS. For $\gamma = 2.5$ the price of the asset increases, when the short-term rate decreases. So, this would actually be classified as a normal monetary policy shock both by JK and CS. The model variations analysed here are highly stylised, but the results are nevertheless suggestive. If monetary policy is inducing investors to take on more risk, by moving the interest rate, then the effect on asset prices may depend on the value of the risk aversion parameter of the investors who are responsible for pricing the assets. And there are some cases, in which, unlike what is assumed in most classifications of monetary policy shocks, the central bank *causes* the short-term rate and asset prices to move in the same direction.

5 Conclusion

The primary goal of this paper is to introduce and explain a Julia package that is able to facilitate asset pricing in settings in which the process of the SDF is known. The package takes advantage of the already available DifferentialEquations.jl package, that is used to simulate SDEs. The package can then generate price functions for fixed income securities. In addition, the price-dividend ratio of dividend-paying securities can also be computed, as long as the joint processes of the SDF and the dividend are given. The package can handle single-variable and multi-variable problems for both fixed income and dividend-paying securities. In the case of single-variable problems the package also facilitates the calculation of the expected return. I illustrate the use of the package in fully worked out examples that have been explained in the paper

[26]This is also consistent with a corresponding result in Bansal and Yaron (2004).

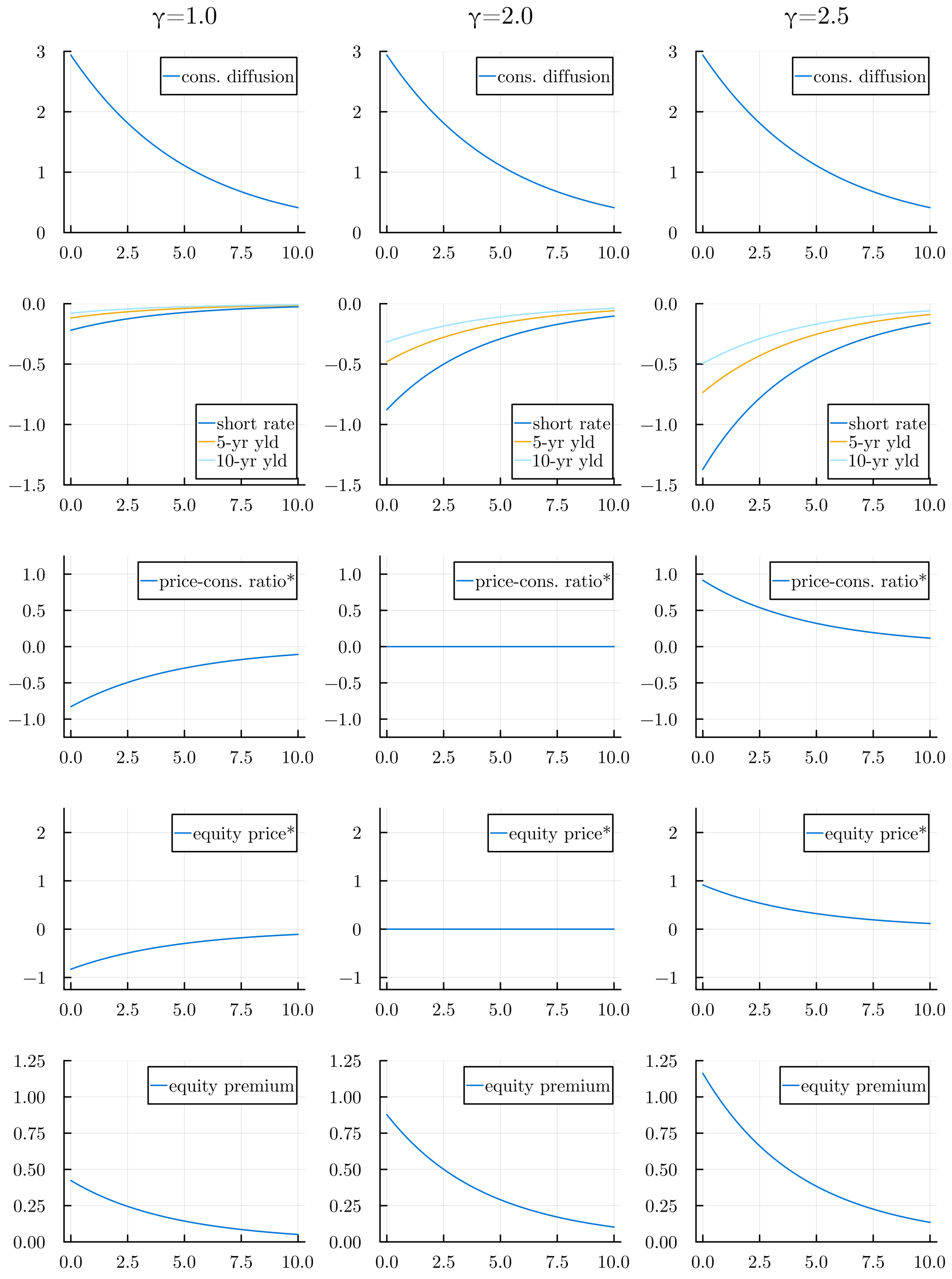


Figure 2: Impulse Responses to Consumption Diffusion Shock
These are responses after a consumption diffusion change. The period shown is ten years. The size of the shock corresponds to one standard deviation. I interpret these responses as taking place after a monetary policy announcement, and I assess to which type of shock they correspond. The short-term rate and the expected excess return (equity premium) always fall, but the response of the asset price is different depending on the risk aversion parameter.
*Plots normally show percent deviation from the stochastic steady state. Plots with starred variables show percent relative deviations from the stochastic steady state.

and whose code is included in the Appendix.[27] The examples include consumption-based models in which the consumption process is exogenous and generates an SDF process. In the examples both one and two state variables are used.

The package is suited for academic research, and I illustrate this with an application, in which I study the possible effects of monetary policy announcements on interest rates and asset prices. While several papers have performed classification of monetary policy announcements, based on responses of financial markets, few provide an explicit theory that shows how these effects are channeled.[28] In general higher interest rates are associated with higher discounting and hence lower asset prices, but if monetary policy increases interest rates via a channel that also increases dividend flows, it becomes non-trivial whether asset prices should increase or decrease. In addition, if monetary policy runs through a risk-taking channel the effects on asset prices can again be ambiguous. My results highlight two things. Firstly, once it is accepted that the central bank can affect the economy by revealing information about the state of the economy, then it becomes difficult to identify a pure monetary policy shock even if financial variables follow the anticipated pattern. Secondly, given unconventional monetary policy or to the extent that monetary policy does not operate through the traditional channel, it is possible that "tightening" ("easing") does not necessarily lead to a decrease (increase) in asset prices. This suggests that the identification of information shocks should also be accompanied by specific theory that outlines the potential channels of monetary policy.

In the future, I am planning to extend the functionality of the package. In particular, as first steps I am planning to a) add the possibility of discrete jumps of the relevant variables, b) include specialised functions to compute the prices of special payoff structures that correspond to different known financial securities, and c) facilitate the choice of underlying solution algorithms, in order to strike the desired balance between speed and accuracy for each kind of problem. Next, functions can be added for the calibration and/or estimation of models to macroeconomic and/or financial data.

[27]The examples are also included in the Github page for the package.

[28]An exception is Cieslak and Schrimpf (2019), who included a non-consumption-based macro-finance model.

References

Adrian, T. and Shin, H. S. (2010). Financial intermediaries and monetary economics. In *Handbook of monetary economics*, vol. 3, Elsevier, pp. 601–650.

Altavilla, C., Brugnolini, L., Gürkaynak, R. S., Motto, R. and Ragusa, G. (2019). Measuring euro area monetary policy. *Journal of Monetary Economics*, **108**, 162–179.

Andrade, P. and Ferroni, F. (2021). Delphic and odyssean monetary policy shocks: Evidence from the euro area. *Journal of Monetary Economics*, **117**, 816–832.

Bansal, R. and Yaron, A. (2004). Risks for the long run: A potential resolution of asset pricing puzzles. *The Journal of Finance*, **59** (4), 1481–1509.

Borio, C. and Zhu, H. (2012). Capital regulation, risk-taking and monetary policy: a missing link in the transmission mechanism? *Journal of Financial stability*, **8** (4), 236–251.

Campbell, J. R., Evans, C. L., Fisher, L. D. and Lustiniano, A. (2012). Macroeconomic effects of federal reserve forward guidance. *Brookings Papers on Economic Activity*.

Chen, Y., Cosimano, T. F. and Himonas, A. A. (2010). Continuous time one-dimensional asset-pricing models with analytic price–dividend functions. *Economic theory*, **42** (3), 461–503.

Cieslak, A. and Schrimpf, A. (2019). Non-monetary news in central bank communication. *Journal of International Economics*, **118**, 293–315.

Cochrane, J. H. (2009). *Asset pricing: Revised edition*. Princeton University Press.

Gürkaynak, R. S., Sack, B. and Swansonc, E. T. (2005). Do actions speak louder than words? the response of asset prices to monetary policy actions and statements. *International Journal of Central Banking*.

Jarociński, M. and Karadi, P. (2020). Deconstructing monetary policy surprises—the role of information shocks. *American Economic Journal: Macroeconomics*, **12** (2), 1–43.

Kuttner, K. N. (2001). Monetary policy surprises and interest rates: Evidence from the fed funds futures market. *Journal of monetary economics*, **47** (3), 523–544.

Lettau, M. and Wachter, J. A. (2011). The term structures of equity and interest rates. *Journal of Financial Economics*, **101** (1), 90–113.

Melissinos, E. (2023). Real term premia in consumption-based models. *Available at SSRN 4582708*.

Nakamura, E. and Steinsson, J. (2018). High-frequency identification of monetary non-neutrality: the information effect. *The Quarterly Journal of Economics*, **133** (3), 1283–1330.

Vayanos, D. and Vila, J.-L. (2021). A preferred-habitat model of the term structure of interest rates. *Econometrica*, **89** (1), 77–112.

Wachter, J. A. (2006). A consumption-based model of the term structure of interest rates. *Journal of Financial Economics*, **79** (2), 365–399.

A Multivariable formulas

- Equation 2:

$$\mathrm{d}x_i = \mu_i(X_t, t)\mathrm{d}t + \sum_{j=1}^{M} \sigma_{i,j}(X_t, t)\mathrm{d}W_{jt}, \quad i = 1, 2, \ldots, N, \quad \mathrm{E}[\mathrm{d}W_{jt}\mathrm{d}W_k(t)] = \rho_{j,k}\mathrm{d}t$$

  where $X_t = (x_1, x_2, \ldots, x_N)$, μ_i is the drift of state variable i, W_{jt} is the j-th Wiener process, M is the number of Wiener processes, $\sigma_{i,j}$ is the diffusion of state variable i with respect to W_{jt}, and $\rho_{j,k}$ is the correlation between the jth and kth Wiener processes.

- Equation 3:

$$\frac{\mathrm{d}\Lambda}{\Lambda} = \mu_\Lambda(X_t, t)\mathrm{d}t + \sum_{j=1}^{M} \sigma_{\Lambda,j}(X_t, t)\mathrm{d}W_{jt}$$

- Equation 4:

$$\begin{aligned}\mathrm{d}Q = &\Bigg(\frac{\partial Q}{\partial t} - \frac{\partial Q}{\partial m} + \sum_{i=1}^{N} \mu_i(X_t, t)\frac{\partial Q}{\partial x_i} \\ &+ \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\sum_{d=1}^{M} \rho_{c,d}\sigma_{a,c}(X_t, t)\sigma_{b,d}(X_t, t)\frac{\partial^2 Q}{\partial x_a \partial x_b}\Bigg)\mathrm{d}t \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{M} \sigma_{i,j}(X_t, t)\frac{\partial Q}{\partial x_i}\mathrm{d}W_{jt}\end{aligned}$$

- Equation 5:

$$\begin{aligned}&\mathrm{E}\left[\frac{\mathrm{d}\big(\Lambda Q\big)}{\Lambda}\right] = 0 \Rightarrow \mathrm{E}\left[\frac{\mathrm{d}\Lambda}{\Lambda}Q + \mathrm{d}Q + \frac{\mathrm{d}\Lambda}{\Lambda}\mathrm{d}Q\right] = 0 \\ \Rightarrow &- r(X, t)Q + \frac{\partial Q}{\partial t} - \frac{\partial Q}{\partial m} + \sum_{i=1}^{N} \mu_i(X_t, t)\frac{\partial Q}{\partial x_i} \\ &+ \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\sum_{d=1}^{M} \rho_{c,d}\sigma_{a,c}(X_t, t)\sigma_{b,d}(X_t, t)\frac{\partial^2 Q}{\partial x_a \partial x_b} \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{M} \rho_{j,k}\sigma_{i,j}(X_t, t)\sigma_{\Lambda,k}\frac{\partial Q}{\partial x_i} = 0\end{aligned}$$

- Equation 6:

$$Q(X_t,t,m) = \mathrm{E}\left[\exp\left(-\int_t^{t+m} r(\hat{X}_s,s)\mathrm{d}s\right)g(t+m)\middle|\hat{X}_t = X_t\right]$$

  where $\hat{X} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$.

- Equation 7:

$$\mathrm{d}\hat{x}_{it} = \left(\sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{i,j}(\hat{X}_t,t)\sigma_{\Lambda,k} + \mu_i(\hat{X}_t,t)\right)\mathrm{d}t + \sum_{j=1}^{M}\sigma_{i,j}(\hat{X}_t,t)\mathrm{d}W_{jt}$$

- Equation 8:

$$Z(X_t,t) \equiv \int_t^{\infty} Q(X_t,t,s)\mathrm{d}s$$

- Equation 9:

$$u_t(X_t) \equiv \frac{U_t}{D_t(X_t)}$$

- Equation 10:

$$\frac{\mathrm{d}D_t}{D_t} = \mu_D(X_t,t)\mathrm{d}t + \sum_{j=1}^{M}\sigma_{D,j}(X_t,t)\mathrm{d}W_{jt}$$

- Equation 14:

$$\mathrm{E}\Big[\mathrm{d}\Big(\Lambda Y(X_t,t,m)\Big)\Big] = 0 \quad \forall t \in (t_0,T), \quad Y(X,t,0) = g(t) \quad \forall X$$

- Equation 16:

$$\mathrm{d}y = \left(\frac{\partial y}{\partial t} - \frac{\partial y}{\partial m} + \sum_{i=1}^{N}\mu_i(X_t,t)\frac{\partial y}{\partial x_i} + \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\sum_{d=1}^{M}\rho_{c,d}\sigma_{a,c}(X_t,t)\sigma_{b,d}(X_t,t)\frac{\partial^2 y}{\partial x_a \partial x_b}\right)\mathrm{d}t$$
$$+ \sum_{i=1}^{N}\sum_{j=1}^{M}\sigma_{i,j}(X_t,t)\frac{\partial y}{\partial x_i}\mathrm{d}W_{jt}$$

- Equation 17:

$$
\begin{aligned}
&-r(X,t)y_t + \frac{\partial y_t}{\partial t} - \frac{\partial y_t}{\partial m} + \sum_{i=1}^{N} \mu_i(X_t,t)\frac{\partial y_t}{\partial x_i} + \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\sum_{d=1}^{M} \rho_{c,d}\sigma_{a,c}(X_t,t)\sigma_{b,d}(X_t,t)\frac{\partial^2 y_t}{\partial x_a \partial x_b} \\
&+ \mu_D(X_t,t)y_t + \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{M} \rho_{j,k}\sigma_{i,j}(X_t,t)\sigma_{\Lambda,k}\frac{\partial y_t}{\partial x_i} + \sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{D,j}(X_t,t)\sigma_{\Lambda,k}y_t \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{D,j}(X_t,t)\sigma_{i,j}(X_t,t)\frac{\partial y_t}{\partial x_i} = 0
\end{aligned}
$$

- Equation 18:

$$
y_t(X_t,t,m) = \mathrm{E}\left[\exp\left(-\int_t^{t+m}\tilde{r}(\tilde{X}_s,s)\mathrm{d}s\right)g(t+m)\middle|\tilde{X}_t = X_t\right]
$$

- Equation 19:

$$
\tilde{r}(X_t,t) = r(X_t,t) + \mu_D(X_t,t) + \sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{D,j}(X_t,t)\sigma_{\Lambda,k}
$$

- Equation 20:

$$
\mathrm{d}\tilde{x} = \left(\sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{i,j}(X_t,t)(\sigma_{\Lambda,k}+\sigma_{D,k}) + \mu_i(\tilde{X}_t,t)\right)\mathrm{d}t + \sum_{j=1}^{M}\sigma_{i,j}(\tilde{X}_t,t)\mathrm{d}W_{jt}
$$

- Equation 21:

$$
\begin{aligned}
\frac{\mathrm{d}U_t}{U_t} + \frac{D_t}{U_t}\mathrm{d}t =& \frac{\mathrm{d}(u_tD_t)}{u_tD_t} + \frac{1}{u_t}\mathrm{d}t = \frac{\mathrm{d}u_t}{u_t} + \frac{\mathrm{d}D_t}{D_t} + \frac{\mathrm{d}u_t\mathrm{d}D_t}{u_tD_t} + \frac{1}{u_t}\mathrm{d}t \\
=& \frac{1}{u_t}\Bigg(\Bigg(\frac{\partial u_t}{\partial t} + \sum_{i=1}^{N}\mu_i(X_t,t)\frac{\partial u_t}{\mathrm{d}x_i} + \frac{1}{2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{M}\sum_{d=1}^{M}\rho_{c,d}\sigma_{a,c}(X_t,t)\sigma_{b,d}(X_t,t)\frac{\partial^2 u}{\partial x_a\partial x_b} \\
&+ \frac{\mu_D(X_t,t)}{D_t}u_t + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{M}\rho_{j,k}\sigma_{D,j}(X_t,t)\sigma_{i,k}\frac{\partial u_t}{\partial x_i}\Bigg)\mathrm{d}t \\
&+ \left(\sum_{i=1}^{N}\sum_{j=1}^{M}\sigma_{i,j}(X_t,t)\frac{\partial u_t}{\partial x_i} + \sum_{j=1}^{M}\sigma_{D,j}(X_t,t)\right)\mathrm{d}W_{jt}\Bigg)
\end{aligned}
$$

B Documentation

C Examples

Example 1 – One state variable

Time-Varying Consumption Drift – Zero-Coupon Bond

The state variable is associated with the consumption drift. Given a CRRA utility function the SDF process can be computed, inserted in the pricing equation and then solved using a Feynman-Kac formula. The modified state variable follows the process:

$$\mathrm{d}\hat{x}_t = \big(-\log\phi(\bar{x}-\hat{x}_t)+\rho_{cx}\sigma_c\sigma_x\big)\mathrm{d}t+\sigma_x\mathrm{d}W_{xt}$$

While the state variable is not modified when there is no correlation between the process for consumption and the process for the state variable:

$$\mathrm{d}x_t = -\log\phi(\bar{x}-x_t)\mathrm{d}t+\sigma_x\mathrm{d}W_{xt}$$

In order to get the price of the zero-coupon security a process for the integral of the short-term rate will also be needed:

$$\mathrm{d}\mathcal{I} = r(\bar{x}_t)\mathrm{d}t$$

Import the packages

```
[1]: import SDFPricing as sdf
     import StochasticDiffEq as sde # this is needed in order to specify the algorithm
```

Define the parameters

```
[2]: cs = (
         phi = 0.92, # mean reversion
         xbar = 0.0, # long-run mean
         rho = 0.01, # time preference parameter
         gamma = 2, # risk aversion
         muc0 = 0.005, # mean of consumption drift
         sigmac = 0.01, # consumption diffusion
         sigmax = 0.005, # state variable diffusion
         rhocx = 0.3 # correlation between consumption and state variable
     );
```

Drift and Diffusion of the processes I also include the unmodified process which will correspond to "risk-neutral pricing". By comparing normal pricing with risk-neutral pricing it is possible to compute excess returns.

```
[3]: mu0(x,c) = -log(c.phi)*(c.xbar-x) # drift of unmodified state
     sigma(x,c) = c.sigmax; # diffusion of both modified and unmodified state
     mu(x,c) = mu0(x,c)-c.rhocx*c.gamma*c.sigmac*sigma(x,c) # drift of modified state
```

```
[3]: mu (generic function with 1 method)
```

Short-term rate function

```
[4]: r(x,c) = c.rho+c.gamma*(c.muc0+x)-c.gamma^2*c.sigmac^2/2;
     r(x) = r(x,cs); # define with one argument for convenience
```

Define setup consistent with SDE package in Julia

```
[5]: function drift(du,u,p,t,c)
         du[1] = mu0(u[1],c)
         du[2] = mu(u[2],c)
         du[3] = r(u[1],c)
         du[4] = r(u[2],c)
     end
     drift(du,u,p,t) = drift(du,u,p,t,cs);
     function diffusion(du,u,p,t,c)
         du[1] = sigma(u[1],c)
         du[2] = sigma(u[2],c)
         du[3] = 0.0
         du[4] = 0.0
     end
     diffusion(du,u,p,t) = diffusion(du,u,p,t,cs);
```

Define the Problem and SolutionSettings variables

```
[6]: prob = sdf.
      ↪Problem(drift=drift,diffusion=diffusion,numNoiseVariables=1,outVariables=[3,4],
     terminalFunction=(ik, x, y, z) -> exp(-x));
     xRange = -0.05:0.01:0.05;
     sett = sdf.SolutionSettings(xRanges=[xRange,], initialValues=[[x, x, 0.0, 0.0]␣
      ↪for x in xRange],
     algorithm=sde.LambaEM(), pathsPerInitialValue=10000, tRange=0.0:1.0:10.0);
```

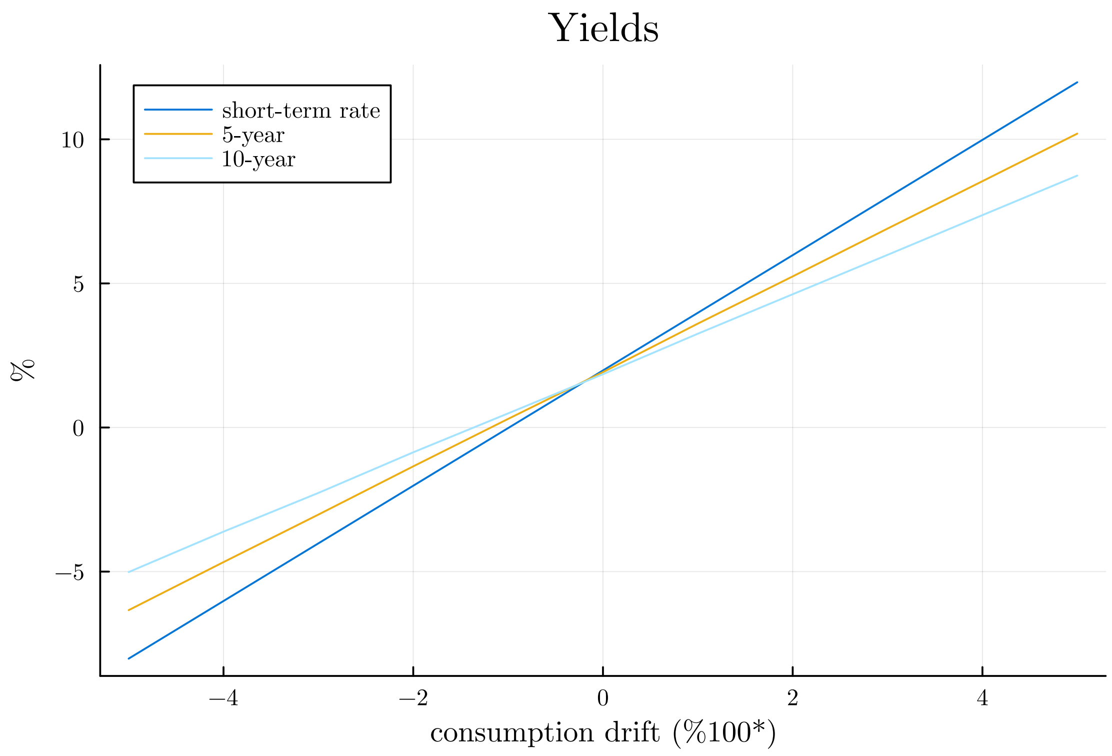
Solve Problem and Define Yield

```
[7]: ((bondPriceRiskNeutral,bondPrice),) = sdf.solve(prob, sett);
     yld(t,x) = -log(bondPrice(t,x))/t;
     yldRiskNeutral(t,x) = -log(bondPriceRiskNeutral(t,x))/t;
```

Plot the yield

```
[8]: # colors: "#0075d6", "#edad14", "#a3e3ff", "#9c0000"
     import Plots as plt
     plt.default(titlefont= (14,"Computer Modern"),legendfont=(8,"Computer Modern"),
         tickfont=(8,"Computer Modern"),guidefont=(10,"Computer Modern"))
     plt.plot(100*xRange, xRange .|> x->100*r(x), title="Yields",
         xlabel="consumption drift (%100*)",label="short-term␣
      ↪rate",color="#0075d6",ylabel="%")
     plt.plot!(100*xRange, 100*yld.(5.0, xRange), label="5-year",color= "#edad14")
     plt.plot!(100*xRange, 100*yld.(10.0, xRange), label="10-year",color="#a3e3ff")
```

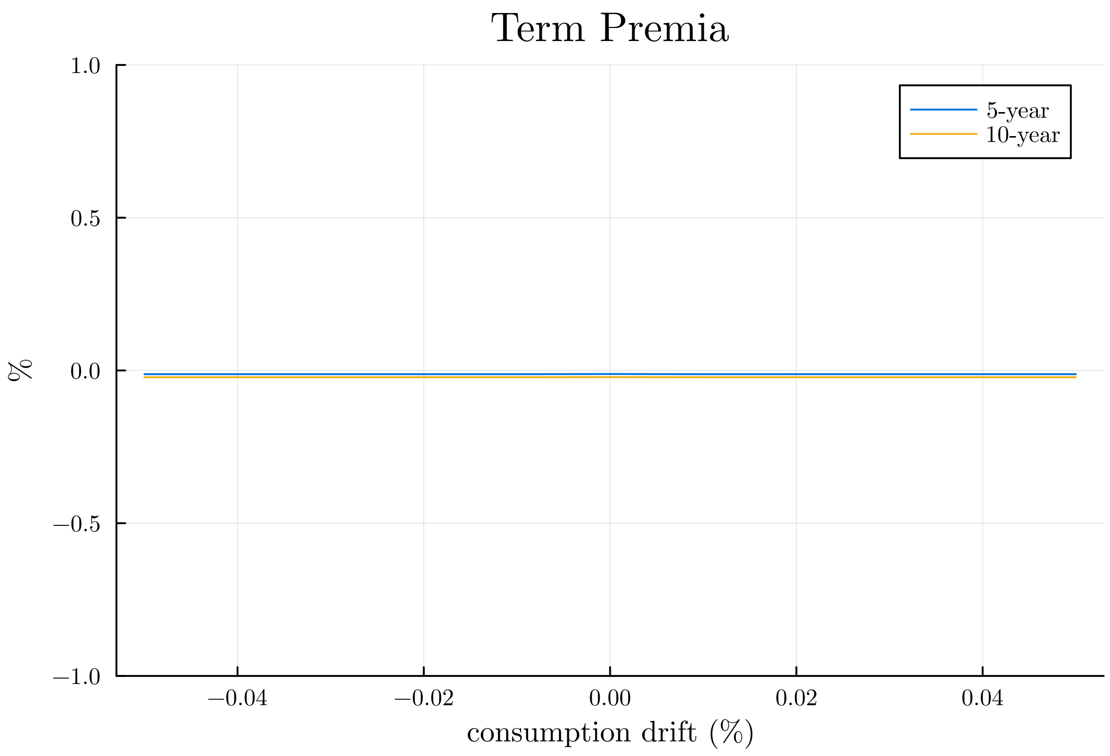
```
[8]:
```



Plot the term premium

[9]:
```
plt.plot(xRange, 100*(yld.(5.0, xRange) .- yldRiskNeutral.(5.0,␣
↪xRange)),title="Term Premia",
    xlabel="consumption drift (%)",label="5-year",ylims=(-0.005,0.
↪005),color="#0075d6",ylabel="%")
plt.plot!(xRange, 100*(yld.(10.0, xRange) .- yldRiskNeutral.(10.0, xRange)),
    label="10-year",ylims=(-1,1),color="#edad14")
```

[9]:



This shows that term premia in such a model with a time-varying consumption drift are negative, very small, and constant.

Example 2 – One state variable

Time-Varying Consumption Diffusion – Zero-Coupon Bond

The state variable is now associated with the consumption diffusion unlike example in which it was associated with consumption drift. Given a CRRA utility function the SDF process can be computed, inserted in the pricing equation and then solved using a Feynman-Kac formula. The modified state variable follows the process:

$$\mathrm{d}\hat{x}_t = \big(- \log \phi(\bar{x} - \hat{x}_t) + \rho_{cx}\sigma_{ct}\sigma_x \big)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt}$$

While the state variable is not modified when there is no correlation between the process for consumption and the process for the state variable:

$$\mathrm{d}x_t = - \log \phi(\bar{x} - x_t)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt}$$

In order to get the price of the zero-coupon security a process for the integral of the short-term rate will also be needed:

$$\mathrm{d}\mathcal{I} = r(\bar{x}_t)\mathrm{d}t$$

Import the packages

[1]:
```
import SDFPricing as sdf
import StochasticDiffEq as sde # this is needed in order to specify the algorithm
```

Define the parameters

[2]:
```
cs = (
    phi = 0.92, # mean reversion
    xbar = 0.0, # long-run mean
    rho = 0.01, # time preference parameter
    gamma = 2, # risk aversion
    muc0 = 0.005, # mean of consumption drift
    sigmac0 = 0.04, # consumption diffusion ###- higher compared to example 1
    sigmax = 0.5, # state variable diffusion ###- higher compared to example 1
    rhocx = -0.3 # correlation between consumption and state variable
);
```

Drift and Diffusion of the processes I also include the unmodified process which will correspond to “risk-neutral pricing”. By comparing normal pricing with risk-neutral pricing it is possible to compute excess returns.

[3]:
```
###- now consumption diffusion is a non-linear function of the state,
###- given that it needs to be positive.
###- I use this function because the simple exponential can get too high for␣
 ↪some samples.
sigmac(x,c) = c.sigmac0*(x<0 ? 2/(1+exp(-2x)) : 4/(1+exp(-x))-1)
mu0(x,c) = -log(c.phi)*(c.xbar-x) # drift of unmodified state
```

```
sigma(x,c) = c.sigmax; # diffusion of modified and unmodified state
mu(x,c) = mu0(x,c)-c.rhocx*c.gamma*sigmac(x,c)*sigma(x,c) # drift of modified␣
 ↪state
```

```
[3]: mu (generic function with 1 method)
```

Short-term rate function

```
[4]: r(x,c) = c.rho+c.gamma*c.muc0-c.gamma^2*sigmac(x,c)^2/2;
     r(x) = r(x,cs);
```

Define setup consistent with SDE solution in Julia

```
[5]: function drift(du,u,p,t,c)
         du[1] = mu0(u[1],c)
         du[2] = mu(u[2],c)
         du[3] = r(u[1],c)
         du[4] = r(u[2],c)
     end
     drift(du,u,p,t) = drift(du,u,p,t,cs);
     function diffusion(du,u,p,t,c)
         du[1] = sigma(u[1],c)
         du[2] = sigma(u[2],c)
         du[3] = 0.0
         du[4] = 0.0
     end
     diffusion(du,u,p,t) = diffusion(du,u,p,t,cs);
```

Define the Problem and SolutionSettings variables

```
[6]: prob = sdf.
      ↪Problem(drift=drift,diffusion=diffusion,numNoiseVariables=1,outVariables=[3,4],
     terminalFunction=(ik, x, y, z) -> exp(-x));
     xRange = -2.0:0.25:2.0;
     sett = sdf.SolutionSettings(xRanges=[xRange,], initialValues=[[x, x, 0.0, 0.0]␣
      ↪for x in xRange],
     algorithm=sde.LambaEM(), pathsPerInitialValue=20000, tRange=0.0:1.0:10.0);
```

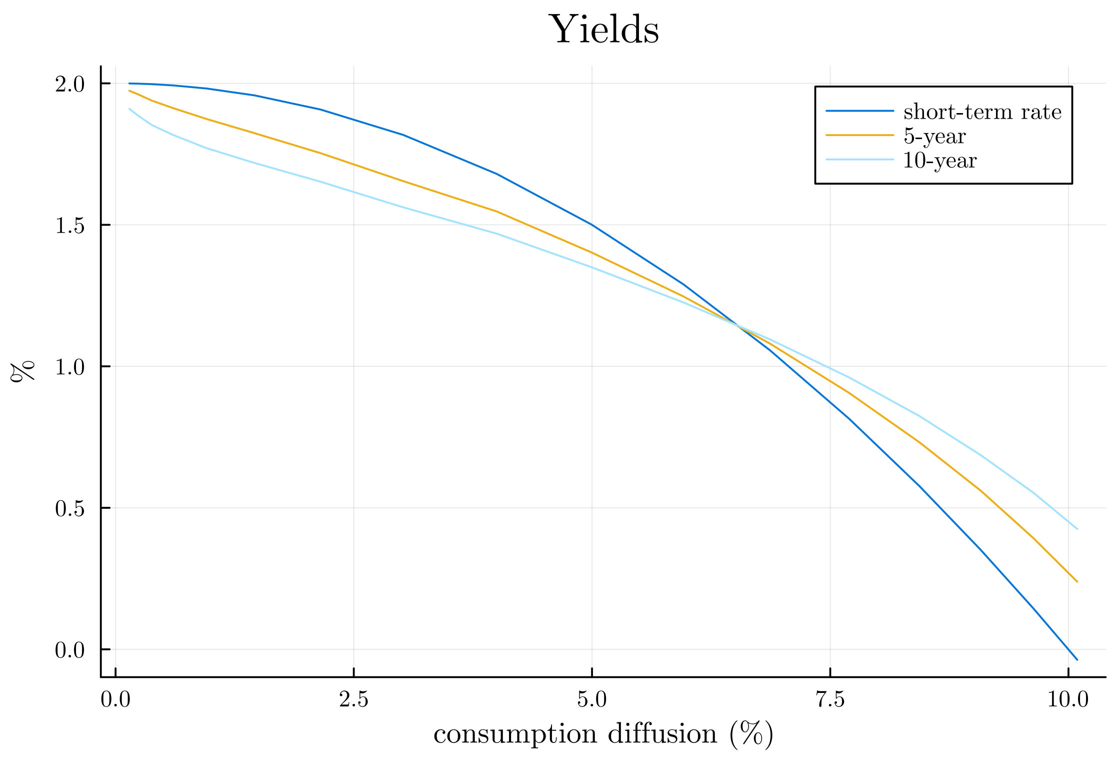
Solve Problem and Define Yield

```
[7]: ((bondPriceRiskNeutral,bondPrice),) = sdf.solve(prob, sett);
     yld(t,x) = -log(bondPrice(t,x))/t;
     yldRiskNeutral(t,x) = -log(bondPriceRiskNeutral(t,x))/t;
```

Plot the yield

```
[8]: # colors: "#0075d6", "#edad14", "#a3e3ff", "#9c0000"
     import Plots as plt
     plt.default(titlefont= (14,"Computer Modern"),legendfont=(8,"Computer Modern"),
         tickfont=(8,"Computer Modern"),guidefont=(10,"Computer Modern"))
     plt.plot(xRange .|>x->100*sigmac(x,cs), xRange .|> x->100*r(x), title="Yields",
         xlabel="consumption diffusion (%)",label="short-term␣
      ↪rate",color="#0075d6",ylabel="%")
     plt.plot!(xRange .|>x->100*sigmac(x,cs), 100*yld.(5.0, xRange),␣
      ↪label="5-year",color= "#edad14")
     plt.plot!(xRange .|>x->100*sigmac(x,cs), 100*yld.(10.0, xRange),␣
      ↪label="10-year",color="#a3e3ff")
```

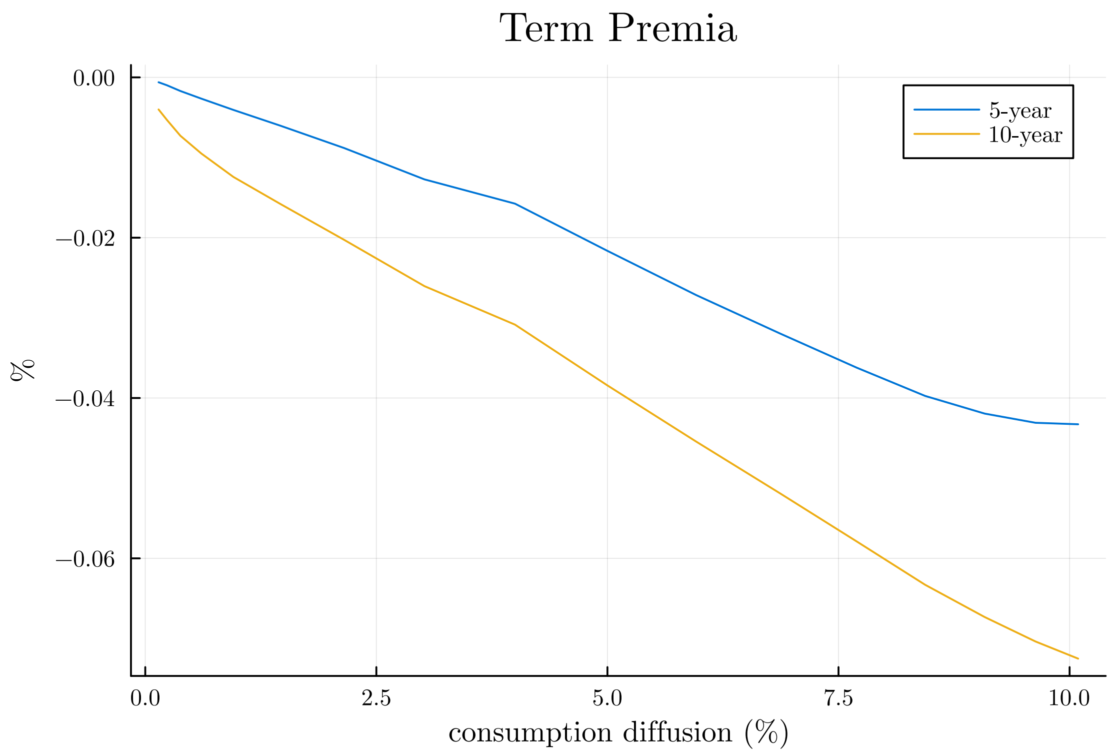
[8]:



Plot the term premium

[9]:
```
plt.plot(xRange .|>x->100*sigmac(x,cs), 100*(yld.(5.0, xRange) .- yldRiskNeutral.
 ↪(5.0, xRange)),title="Term Premia",
    xlabel="consumption diffusion (%)",label="5-year",color="#0075d6",ylabel="%")
plt.plot!(xRange .|>x->100*sigmac(x,cs), 100*(yld.(10.0, xRange) .-␣
 ↪yldRiskNeutral.(10.0, xRange)),
    label="10-year",color="#edad14")
```

[9]:



This shows that term premia are state-dependent when consumption diffusion is time-varying. They can also get larger in absolute value when consumption volatility is relatively high.

Example 3 – One state variable

Time-Varying Drift – Price Consumption Ratio

Here the setup is exactly the same as in example 1, but now I calculate the price consumption ratio instead of the price of zero coupon bond. By changing the values of the parameters it is also possible to compute a more general price-dividend ratio, for an asset that does not have the same dividend process as consumption. The modified process in this case is:

$$\mathrm{d}\tilde{x}_t = \big(- \log\phi(\bar{x} - \tilde{x}_t) + \rho_{cx}\sigma_c\sigma_x + \rho_{xD}\sigma_x\sigma_D \big)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt}$$

In order to get the price of the zero-coupon security a process for the integral of the short-term rate will also be needed:

$$\mathrm{d}\mathcal{I} = r(\tilde{x}_t)\mathrm{d}t$$

Import the packages

[1]:
```
import SDFPricing as sdf
import StochasticDiffEq as sde # this is needed in order to specify the algorithm
```

Define the parameters

[2]:
```
cs = (
    phi = 0.92, # mean reversion
    xbar = 0.0, # long-run mean
    rho = 0.02, # time preference parameter
    gamma = 2, # risk aversion
    muc0 = 0.01, # mean of consumption drift
    sigmac = 0.01, # consumption diffusion
    sigmax = 0.005, # state variable diffusion
    rhocx = 0.3, # correlation between consumption and state variable
    # sigmaD = 0.02, # dividend diffusion ###- added compared to example 1
    # muD = 0.02, # dividend drift ###- added compared to example 1
    # rhoxD = 0.5, # correlation between dividends and state variable ###- added␣
↪compared to example 1
    # rhocD = 0.4 # correlation between dividends and consumption ###- added␣
↪compared to example 1
    sigmaD = 0.01, # dividend diffusion ###- case of consumption perpetuity
    muD = 0.01, # dividend drift ###- case of consumption perpetuity
    rhoxD = 0.3, # correlation between dividends and state variable ###- case of␣
↪consumption perpetuity
    rhocD = 1.0 # correlation between dividends and consumption ###- case of␣
↪consumption perpetuity
);
```

Drift and Diffusion of the processes I also include the unmodified process which will correspond to "risk-neutral pricing". By comparing normal pricing with risk-neutral pricing it is possible to compute excess returns.

[3]:
```
# diffusion of modified state
sigma(x,c) = c.sigmax;
# drift of modified state
mu(x,c) = -log(c.phi)*(c.xbar-x)-c.rhocx*c.gamma*c.sigmac*sigma(x,c)+c.
↪rhoxD*sigma(x,c)*c.sigmaD
```

[3]:
```
mu (generic function with 1 method)
```

Short-term rate function

[4]:
```
r(x,c) = c.rho+c.gamma*(c.muc0+x)-c.gamma^2*c.sigmac^2/2;
r(x) = r(x,cs);
muD(x) = cs.muD+x; #- case of consumption perpetuity
# muD(x) = cs.muD; #- perpetuity with constant dividend drift
rmod(x,c) = r(x,c)-(muD(x)-c.gamma*c.rhocD*c.sigmac*c.sigmaD);
rmod(x) = rmod(x,cs);
```

Define setup consistent with SDE solution in Julia

[5]:
```
function drift(du,u,p,t,c)
    du[1] = mu(u[1],c)
    du[2] = rmod(u[1],c)
end
drift(du,u,p,t) = drift(du,u,p,t,cs);
function diffusion(du,u,p,t,c)
    du[1] = sigma(u[1],c)
    du[2] = 0.0
end
diffusion(du,u,p,t) = diffusion(du,u,p,t,cs);
```

Define the Problem and SolutionSettings variables In the theory I state that the price consumption ratio is computed from the integral over all consumption strip maturities. In practice it is not possible to integrate to infinity. So, I compute consumption strips up to a maturity of 300 years.

[6]:
```
prob = sdf.
↪Problem(drift=drift,diffusion=diffusion,numNoiseVariables=1,outVariables=[2],
terminalFunction=(ik, x, y, z) -> exp(-x));
xRange = -0.05:0.006:0.05;
tRange = 0.0:5.0:300.0;
sett = sdf.SolutionSettings(xRanges=[xRange,], initialValues=[[x, 0.0] for x in␣
↪xRange],
algorithm=sde.LambaEM(), pathsPerInitialValue=5000, tRange=tRange);
# add the settings in order to compmute price-dividend ration of the continuous␣
↪payoff security
sett2 = sdf.SolutionSettings(sett; continuousPayoffVars=[2]);
```

Solve Problem, Get Yield and Price Consumption Ratio

[7]:
```
((consumptionStrip,),(priceConsumptionRatio,)) = sdf.solve(prob, sett2);
```

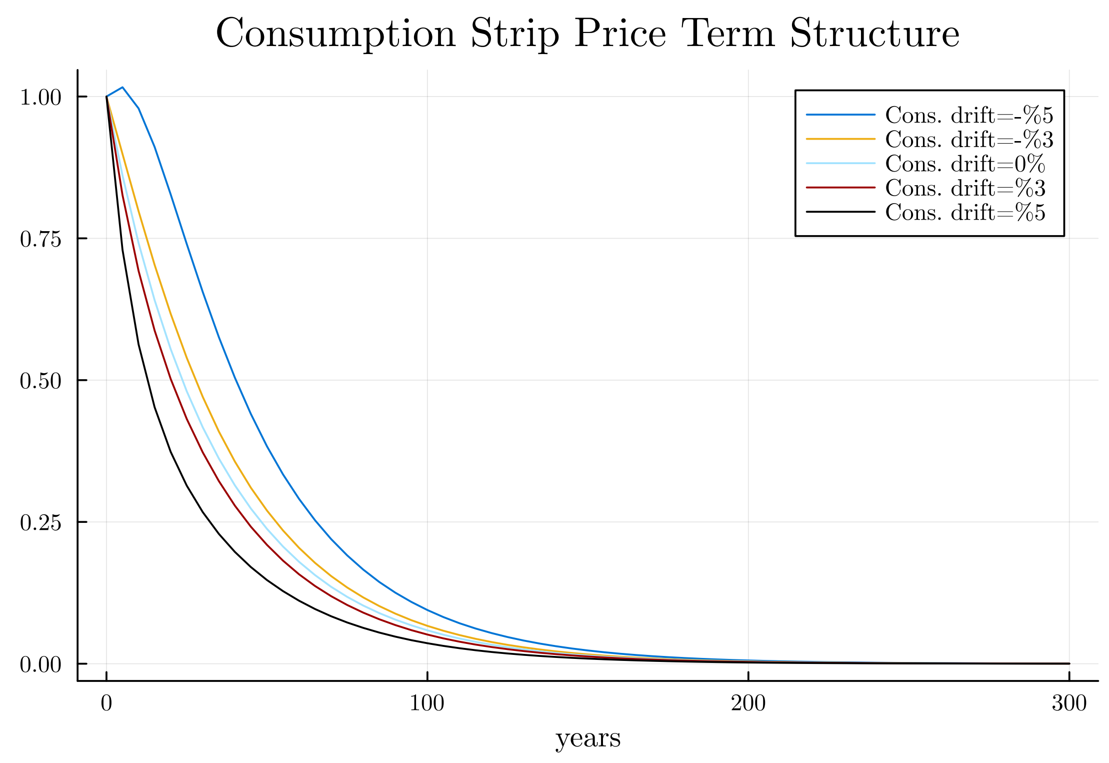
Get the Return of the Consumption Perpetuity The calculation requires the computation of the first and second derivatives of the price consumption ratio with respect to the state of the economy.

[8]:
```
(DPC,D2PC) = sdf.derivatives(priceConsumptionRatio);
ret(x) = (DPC(x)*(mu(x, cs) + sigma(x, cs) * cs.sigmac * cs.rhocx) +
    D2PC(x)* sigma(x, cs)^2/2.0 + 1.0)/priceConsumptionRatio(x) + muD(x);
```

Plot the Consumption Strip Term Structure

[9]:
```
import Plots as plt
# # colors: "#0075d6", "#edad14", "#a3e3ff", "#9c0000", "#000000"
plt.default(titlefont= (14,"Computer Modern"),legendfont=(8,"Computer Modern"),
    tickfont=(8,"Computer Modern"),guidefont=(10,"Computer Modern"))
plt.plot(tRange, consumptionStrip.(tRange, -0.04),color="#0075d6",
    title="Consumption Strip Price Term Structure", xlabel="years",label="Cons.␣
 ↪drift=-%5")
plt.plot!(tRange, consumptionStrip.(tRange, -0.01),label="Cons.␣
 ↪drift=-%3",color="#edad14")
plt.plot!(tRange, consumptionStrip.(tRange, 0.0),label="Cons.␣
 ↪drift=0%",color="#a3e3ff")
plt.plot!(tRange, consumptionStrip.(tRange, 0.01),label="Cons.␣
 ↪drift=%3",color="#9c0000")
plt.plot!(tRange, consumptionStrip.(tRange, 0.04),label="Cons.␣
 ↪drift=%5",color="#000000")
```

[9]:

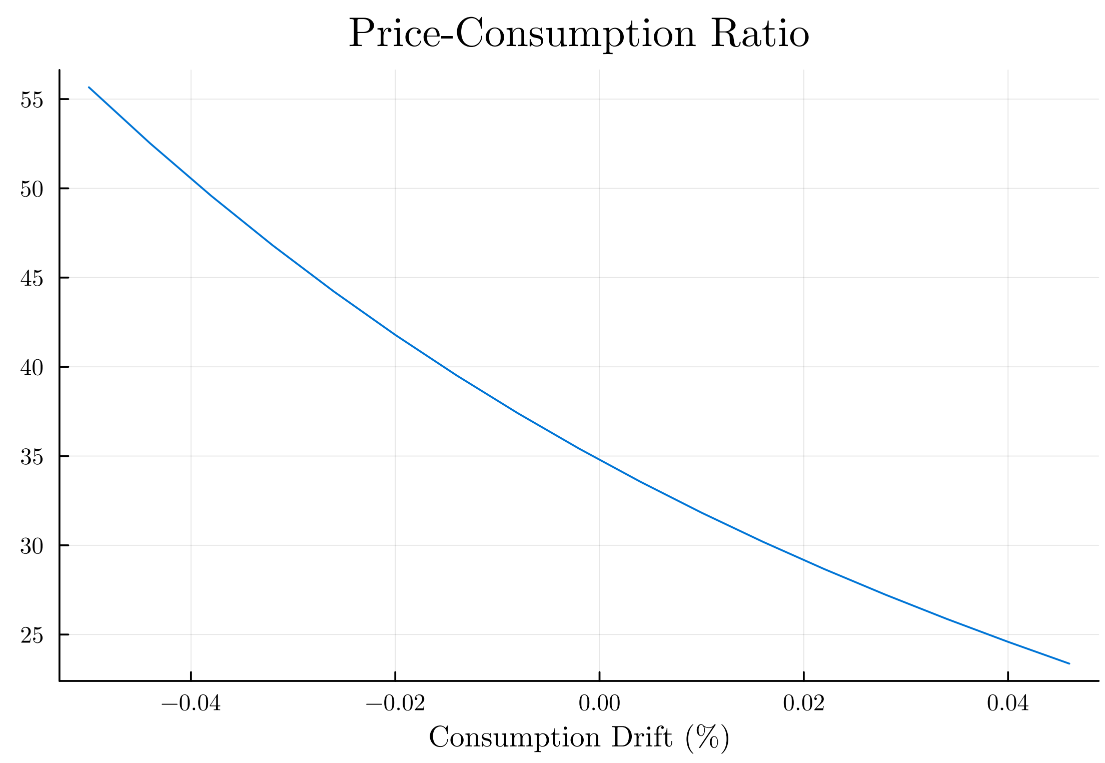


It can be seen that for all values of the state variable the price of the securities comes close to 0 for maturities as long as 300 years.

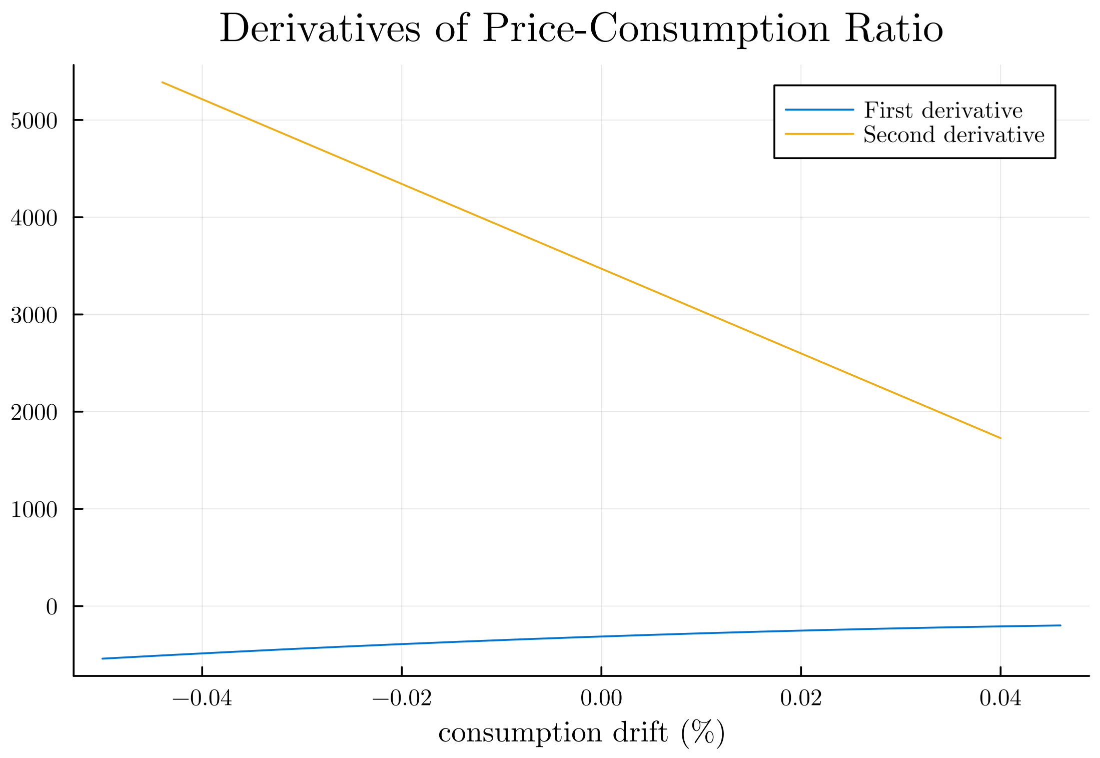
Plot the Price Consumption Ratio

[10]:
```
plt.plot(xRange, priceConsumptionRatio(xRange), legend=false,
    title="Price-Consumption Ratio",color="#0075d6", xlabel="Consumption Drift␣
 ↪(%)")
```

[10]:



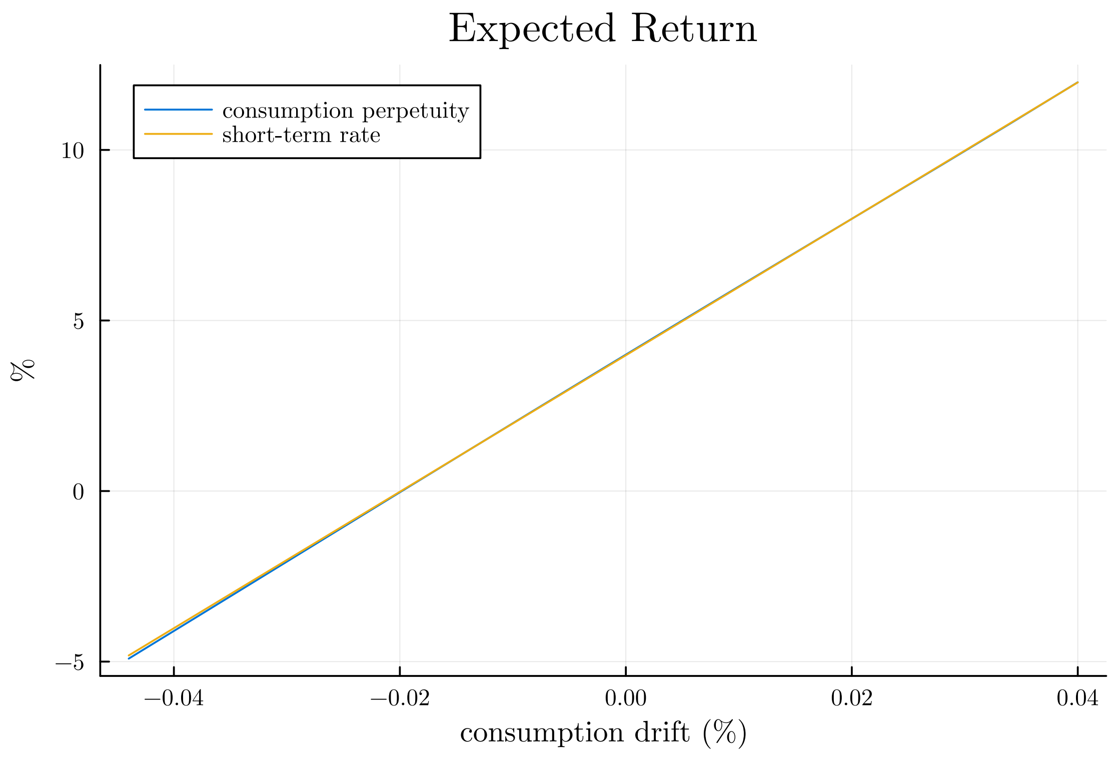
```
[11]: plt.plot(xRange, DPC.(xRange),
          title="Derivatives of Price-Consumption Ratio",label="First␣
       ↪derivative",color="#0075d6")
      plt.plot!(xRange[2:end-1], D2PC.(xRange[2:end-1]),label="Second␣
       ↪derivative",color="#edad14",xlabel="consumption drift (%)")
[11]:
```



Plot the Return

```
[12]: plt.plot(xRange[2:end-1], 100*ret.(xRange[2:end-1]), label="consumption␣
       ↪perpetuity",
          title="Expected Return", color="#0075d6",xlabel="consumption drift␣
       ↪(%)",ylabel="%")
      plt.plot!(xRange[2:end-1], 100*r.(xRange[2:end-1]),
          label="short-term rate",color="#edad14")
```

[12]:



As expected in the standard model with time-varying consumption drift the premium compared to the short-term rate is almost zero.

Example 4 – One state variable

Time-Varying Consumption Drift – Price Consumption Ratio

Here the setup is exactly the same as in example 2, but now I calculate the price consumption ratio instead of the price of zero coupon bond. The modified process in this case is:

$$\mathrm{d}\tilde{x}_t = \big(- \log \phi(\bar{x}_0 - \tilde{x}_t) + \rho_{cx}\sigma_{ct}\sigma_x + \rho_{xD}\sigma_x\sigma_D \big)\mathrm{d}t + \sigma_x \mathrm{d}W_{xt}$$

In order to get the price of the zero-coupon security a process for the integral of the short-term rate will also be needed:

$$\mathrm{d}\mathcal{I} = r(\tilde{x}_t)\mathrm{d}t$$

Import the packages

```
[1]: import SDFPricing as sdf
     import StochasticDiffEq as sde # this is needed in order to specify the algorithm
```

Define the parameters

```
[2]: cs = (
         phi = 0.92, # mean reversion
         xbar = 0.0, # long-run mean
         rho = 0.02, # time preference parameter
         gamma = 2.0, # risk aversion
         muc0 = 0.005, # mean of consumption diffusion
         sigmac = 0.08, # consumption diffusion
         sigmax = 0.5, # state variable diffusion
         rhocx = -0.3, # correlation between consumption and state variable
         # sigmaD = 0.10, # dividend diffusion ###- added compared to example 1
         # muD = 0.01, # dividend drift ###- added compared to example 1
         # rhoxD = -0.5, # correlation between dividends and state variable ###-␣
     ↪added compared to example 1
         # rhocD = 0.4 # correlation between dividends and consumption ###- added␣
     ↪compared to example 1
         sigmaD = 0.08, # dividend diffusion ###- case of consumption perpetuity
         muD = 0.005, # dividend drift ###- case of consumption perpetuity
         rhoxD = -0.3, # correlation between dividends and state variable ###- case␣
     ↪of consumption perpetuity
         rhocD = 1.0 # correlation between dividends and consumption ###- case of␣
     ↪consumption perpetuity
     );
```

Drift and Diffusion of the processes I also include the unmodified process which will correspond to “risk-neutral pricing”. By comparing normal pricing with risk-neutral pricing it is possible to compute excess returns.

[3]:
```
###- now consumption diffusion is a non-linear function of the state,
###- given that it needs to be positive.
###- I use this function because the simple exponential can get too high for␣
 ↪some samples.
sigmac(x,c) = c.sigmac*(x<0 ? 2/(1+exp(-2x)) : 4/(1+exp(-x))-1);
sigmac(x) = sigmac(x,cs);
# sigmaD(x,c) = defineSomeFunctionOf(x,c); #- general dividend diffusion
sigmaD(x,c) = sigmac(x,c); #- case of consumption perpetuity
sigma(x,c) = c.sigmax; # diffusion of modified state
mu(x,c) = -log(c.phi)*(c.xbar-x)-c.rhocx*c.gamma*sigmac(x,c)*sigma(x,c)+c.
 ↪rhoxD*sigma(x,c)*sigmaD(x,c) ; # drift of unmodified state
```

Short-term rate function

[4]:
```
r(x,c) = c.rho+c.gamma*c.muc0-c.gamma^2*sigmac(x,c)^2/2;
r(x) = r(x,cs);
muD(x) = cs.muD; # perpetuity with constant dividend drift
rmod(x,c) = r(x,c)-(muD(x)-c.gamma*c.rhocD*sigmac(x,c)*sigmaD(x,c));
rmod(x) = rmod(x,cs);
```

Define setup consistent with SDE solution in Julia

[5]:
```
function drift(du,u,p,t,c)
    du[1] = mu(u[1],c)
    du[2] = rmod(u[1],c)
end
drift(du,u,p,t) = drift(du,u,p,t,cs);
function diffusion(du,u,p,t,c)
    du[1] = sigma(u[1],c)
    du[2] = 0.0
end
diffusion(du,u,p,t) = diffusion(du,u,p,t,cs);
```

Define the Problem and SolutionSettings variables

[6]:
```
prob = sdf.
 ↪Problem(drift=drift,diffusion=diffusion,numNoiseVariables=1,outVariables=[2],
terminalFunction=(ik, x, y, z) -> exp(-x));
xRange = -2.0:0.4:2.0;
tRange = 0.0:5.0:300.0;
sett = sdf.SolutionSettings(xRanges=[xRange,], initialValues=[[x, 0.0] for x in␣
 ↪xRange],
algorithm=sde.LambaEM(), pathsPerInitialValue=5000, tRange=tRange);
# add the settings in order to compmute price-dividend ration of the continuous␣
 ↪payoff security
sett2 = sdf.SolutionSettings(sett; continuousPayoffVars=[2]);
```

Solve Problem and Define Yield

[7]:
```
((consumptionStrip,),(priceConsumptionRatio,)) = sdf.solve(prob, sett2);
```

Get the Return of the Consumption Perpetuity The calculation requires the computation of the first and second derivatives of the price consumption ratio with respect to the state of the economy.

[8]:
```
(DPC,D2PC) = sdf.derivatives(priceConsumptionRatio);
ret(x) = (DPC(x)*(mu(x, cs) + sigma(x, cs) * sigmac(x,cs) * cs.rhocx) +
    D2PC(x)* sigma(x, cs)^2/2.0 + 1.0)/priceConsumptionRatio(x) + muD(x);
```

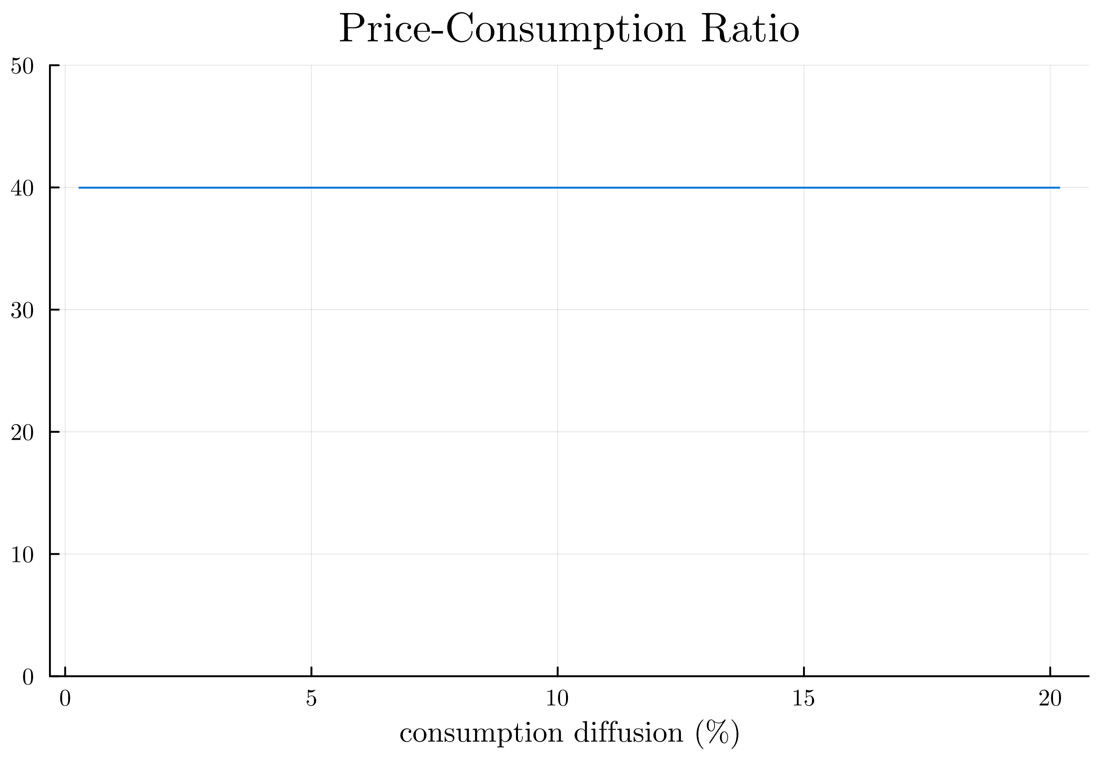
In the theory I state that the price consumption ratio is computed from the integral over all consumption strip maturities. In practice it is not possible to integrate to infinity. So, I compute consumption strips up to a maturity of 300 years.

It can be seen that for all values of the state variable the price of the securities comes close to 0 for maturities as long as 300 years.

Plot the Price Consumption Ratio

[9]:
```
import Plots as plt
plt.default(titlefont= (14,"Computer Modern"),legendfont=(8,"Computer Modern"),
    tickfont=(8,"Computer Modern"),guidefont=(10,"Computer Modern"))
plt.plot(100*sigmac.(xRange), priceConsumptionRatio(xRange), legend=false,
    title="Price-Consumption Ratio",color="#0075d6",ylims = (0.0, 50.0),␣
 ↪xlabel="consumption diffusion (%)")
```

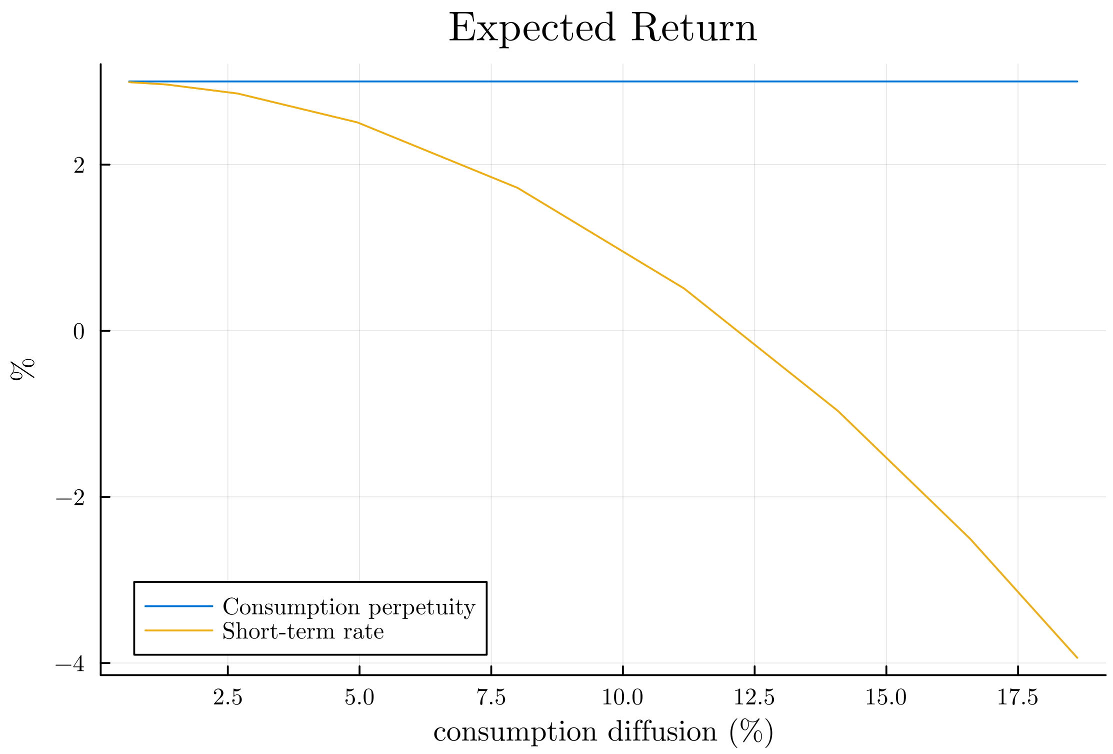
[9]:



Plot the Return

[10]:
```
plt.plot(100*sigmac.(xRange[2:end-1]), 100*ret.(xRange[2:end-1]),␣
↪label="Consumption perpetuity",
    title="Expected Return", color="#0075d6", xlabel="consumption diffusion␣
↪(%)",ylabel="%")
plt.plot!(100*sigmac.(xRange[2:end-1]), 100*r.(xRange[2:end-1]),
    label="Short-term rate",color="#edad14")
```

[10]:



It turns out that for a value of gamma=2 the effects cancel out and the price consumption ratio constant. However, there is still an expected return due to the dividend and the equity premium is large and increasing as volatility increases, due to the falling risk-free rate.

Example 5 – Two state variables

Time-Varying Drift and Diffusion – Zero-Coupon Bond

Here the setup is more complicated, having two state variables. Both consumption drift and diffusion are simultaneously time-varying. The modified processes that will give the price of the zero coupon security are the following (where $\hat{x}_1$ and $\hat{x}_2$ revert to $\bar{x}_1$ and $\bar{x}_2$ respectively):

$$\mathrm{d}c_t = \mu_{ct}\mathrm{d}t+\sigma_{ct}(1-|\rho_{c1}|-|\rho_{c2}|)\mathrm{d}W_{ct}+\sigma_{ct}\rho_{cx1}\mathrm{d}W_{x1t}+\sigma_{ct}\rho_{cx1}\mathrm{d}W_{x2t}\mathrm{d}\hat{x}_{1t} = \big(-\log\phi\cdot(\bar{x}_1-\hat{x}_{1t})+\rho_{cx1}\sigma_{ct}\sigma_{1x}\big)\mathrm{d}t+\sigma_{x1}\frac{1}{1+}$$

where W_{c1}, W_{x1} and W_{x2} are independent and:

$$\mathrm{E}[\mathrm{d}c_t\mathrm{d}\bar{x}_{1t}] = \left(\rho_{cx1}\frac{1}{1+\rho_{12}}+\rho_{cx2}\frac{\rho_{12}}{1+\rho_{12}}\right)\sigma_{ct}\sigma_{x1}\mathrm{d}t \approxeq \rho_{cx1}\sigma_{ct}\sigma_{x1}\mathrm{d}t$$

$$\mathrm{E}[\mathrm{d}c_t\mathrm{d}\bar{x}_{2t}] = \left(\rho_{cx2}\frac{1}{1+\rho_{21}}+\rho_{cx1}\frac{\rho_{21}}{1+\rho_{21}}\right)\sigma_{ct}\sigma_{x2}\mathrm{d}t \approxeq \rho_{cx2}\sigma_{ct}\sigma_{x2}\mathrm{d}t$$

$$\mathrm{E}[\mathrm{d}\bar{x}_{1t}\mathrm{d}\bar{x}_{2t}] = \sigma_{x1}\sigma_{x2}\frac{\rho_{21}+\rho_{12}}{1+\rho_{12}+\rho_{21}+\rho_{12}\rho_{21}}\mathrm{d}t \approxeq (\rho_{12}+\rho_{21})\sigma_{x1}\sigma_{x2}\mathrm{d}t$$

and the approximate equations are valid if ρ_{12} and ρ_{21} are small.

In order to get the price of the zero-coupon security a process for the integral of the short-term rate will also be needed:

$$\mathrm{d}\mathcal{I} = r(\hat{x}_{1t},\hat{x}_{2t})\mathrm{d}t$$

Import the packages

[1]:
```
import SDFPricing as sdf
import StochasticDiffEq as sde # this is needed in order to specify the algorithm
```

```
[ Info: Precompiling SDFPricing
[8f91c045-db67-4ada-b18f-1d80840c3158]
```

Define the parameters

[2]:
```
cs = (
    phi1 = 0.91, # mean reversion
    phi2 = 0.96, # mean reversion
    xbar1 = 0.0, # long-run mean
    xbar2 = 0.0, # long-run mean
    rho = 0.02, # time preference parameter
    gamma = 2.0, # risk aversion
    muc = 0.005, # mean of consumption drift
    sigmac = 0.08, # mean of consumption diffusion
    sigmax1 = 0.005, # volatility
    sigmax2 = 0.2, # volatility
    rhocx1 = 0.0, # correlation parameter
    rhocx2 = -0.6, # correlation parameter
    rho12 = 0.1, # correlation parameter
    rho21 = 0.1 # correlation parameter
    );
```

Drift and Diffusion of the processes I also include the unmodified process which will correspond to "risk-neutral pricing". By comparing normal pricing with risk-neutral pricing it is possible to compute excess returns.

[3]:
```
###- now consumption diffusion is a non-linear function of the state,
###- given that it needs to be positive.
###- I use this function because the simple exponential can get too high for␣
↪some samples.
sigmac(x,c) = c.sigmac*(x<0 ? 2/(1+exp(-2x)) : 4/(1+exp(-x))-1);
muc(x,c) = c.muc + x;
sigmac(x) = sigmac(x,cs);
sigma1_1(x,c) = c.sigmax1/(1+c.rho12);
sigma1_2(x,c) = c.sigmax1*c.rho12/(1+c.rho12);
sigma1(x,c) = sigma1_1(x,c)+sigma1_2(x,c);
sigma2_1(x,c) = c.sigmax2*c.rho21/(1+c.rho21);
sigma2_2(x,c) = c.sigmax2/(1+c.rho21);
sigma2(x,c) = sigma2_1(x,c)+sigma2_2(x,c);
mu1(x,c) = -log(c.phi1)*(c.xbar1-x)-c.gamma*sigmac(x,c)*(sigma1_1(x,c)*c.
↪rhocx1+sigma1_2(x,c)*c.rhocx2); # drift of modified state
mu2(x,c) = -log(c.phi2)*(c.xbar2-x)-c.gamma*sigmac(x,c)*(sigma2_1(x,c)*c.
↪rhocx1+sigma2_2(x,c)*c.rhocx2); # drift of modified state
mu10(x,c) = -log(c.phi1)*(c.xbar1-x); # drift of unmodified state
mu20(x,c) = -log(c.phi2)*(c.xbar2-x); # drift of unmodified state
```

Short-term rate function

[4]:
```
r(x1,x2,c) = c.rho+c.gamma*muc(x1,c)-c.gamma^2*sigmac(x2,c)^2/2;
r(x1,x2) = r(x1,x2,cs);
```

Define setup consistent with SDE solution in Julia

[5]:
```
function drift(du,u,p,t,c)
    du[1] = mu1(u[1],c)
    du[2] = mu2(u[2],c)
    du[3] = mu10(u[3],c)
    du[4] = mu20(u[4],c)
    du[5] = r(u[1],u[2],c)
    du[6] = r(u[3],u[4],c)
end
drift(du,u,p,t) = drift(du,u,p,t,cs);
function diffusion(du,u,p,t,c)
    du[1, 1] = sigma1_1(u[1], c)
    du[1, 2] = sigma1_2(u[1], c)
    du[2, 1] = sigma2_1(u[2], c)
    du[2, 2] = sigma2_2(u[2], c)
    du[3, 1] = sigma1_1(u[3], c)
    du[3, 2] = sigma1_2(u[3], c)
    du[4, 1] = sigma2_1(u[4], c)
    du[4, 2] = sigma2_2(u[4], c)
    du[5, 1] = 0.0
    du[5, 2] = 0.0
    du[6, 1] = 0.0
    du[6, 2] = 0.0
end
diffusion(du,u,p,t) = diffusion(du,u,p,t,cs);
```

Define the Problem and SolutionSettings variables

[6]:
```
prob = sdf.
 ↪Problem(drift=drift,diffusion=diffusion,numNoiseVariables=2,outVariables=[5,6],
terminalFunction=(ik, x, y, z) -> exp(-x),diagonalNoise=false);
xRanges = [-0.03:0.005:0.03,-2.0:0.4:2.0];
tRange = 0.0:1.0:20.0;
sett = sdf.SolutionSettings(xRanges=xRanges, initialValues=vcat([[x, y,x,y, 0.
 ↪0,0.0] for y in xRanges[2] for x in xRanges[1]]),
algorithm=sde.LambaEM(), pathsPerInitialValue=5000, tRange=tRange);
```

Solve Problem and Define Yield

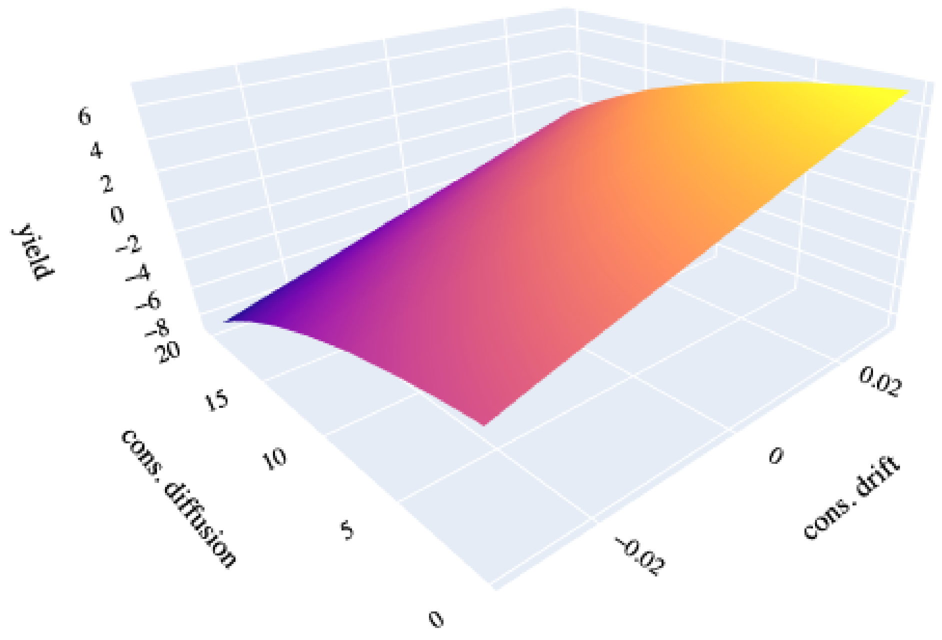
[7]:
```
((bondPrice,riskNeutralPrice),) = sdf.solve(prob, sett);
yld(t,x1,x2) = -log(bondPrice(t,x1,x2))/t;
yldRiskNeutral(t,x1,x2) = -log(riskNeutralPrice(t,x1,x2))/t;
```

Plot the Yield in 3D

[10]:
```
import Plots as plt
import PlotlyJS as pltjs
coordinates = pltjs.surface(
    z=[100*yld(10.0, x1, x2) for x1 in xRanges[1], x2 in xRanges[2]],␣
 ↪x=xRanges[1],
    y=100*sigmac.(xRanges[2]),
    showscale=false)

layout = pltjs.Layout(
    width=800, height=350,
    title_x=0.5,
    titlefont_size="16",
    scene_aspectratio=pltjs.attr(x=1, y=1, z=0.5),
    scene=pltjs.attr(
        xaxis=pltjs.attr(title="cons. drift"),
        yaxis=pltjs.attr(title="cons. diffusion"),
        zaxis=pltjs.attr(title="yield"),
        camera=pltjs.attr(
            center=pltjs.attr(x=0.3, y=0, z=-0.40),
            eye=pltjs.attr(x=-.95, y=-1.25, z=0.65)
        )
    ),
    font=pltjs.attr(family="Computer Modern", size=12, color="black"),
    margin=pltjs.attr(l=0, r=0, b=0, t=0, pad=0))
pltjs.plot([coordinates], layout)
```

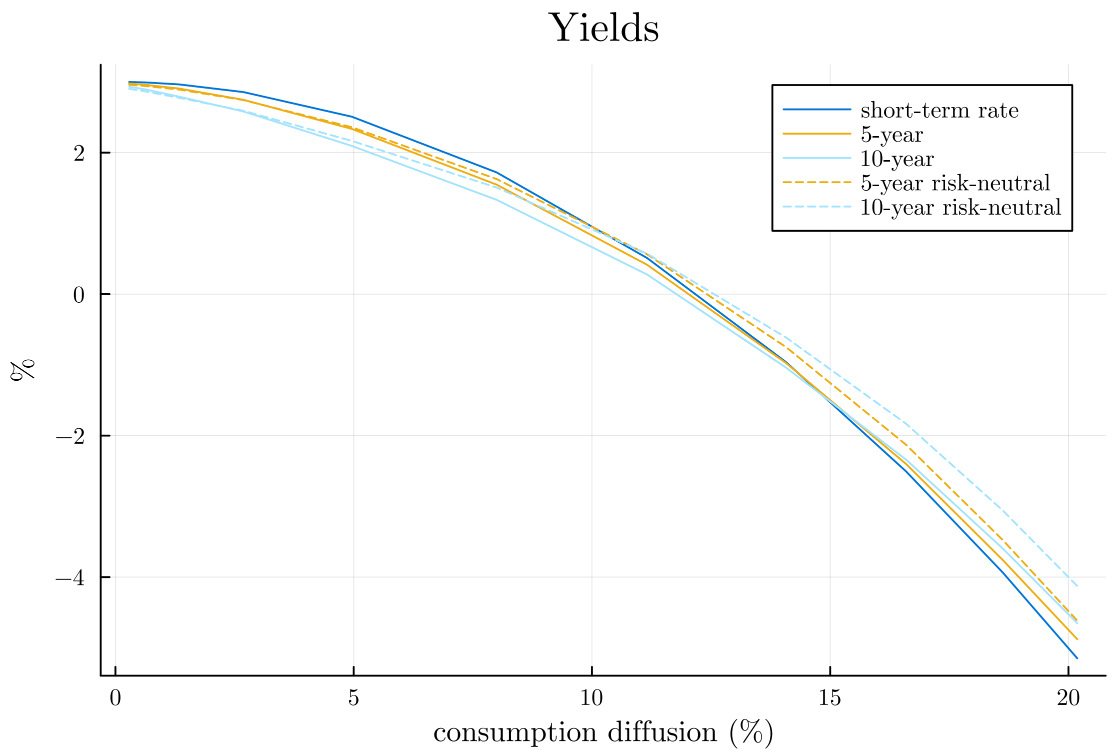
[10]:



[9]:

```
# colors: "#0075d6", "#edad14", "#a3e3ff", "#9c0000"
import Plots as plt
x1 = 0.0
plt.default(titlefont= (14,"Computer Modern"),legendfont=(8,"Computer Modern"),
    tickfont=(8,"Computer Modern"),guidefont=(10,"Computer Modern"))
plt.plot(100*sigmac.(xRanges[2]), xRanges[2] .|> x2->100*r(x1,x2),␣
 ↪title="Yields",
    xlabel="consumption diffusion (%)",ylabel="%",label="short-term␣
 ↪rate",color="#0075d6")
plt.plot!(100*sigmac.(xRanges[2]), 100*yld.(5.0, x1,xRanges[2]),␣
 ↪label="5-year",color= "#edad14")
plt.plot!(100*sigmac.(xRanges[2]), 100*yld.(10.0, x1,xRanges[2]),␣
 ↪label="10-year",color="#a3e3ff")
plt.plot!(100*sigmac.(xRanges[2]), 100*yldRiskNeutral.(5.0, x1,xRanges[2]),␣
 ↪label="5-year risk-neutral",color= "#edad14",style=:dash)
plt.plot!(100*sigmac.(xRanges[2]), 100*yldRiskNeutral.(10.0, x1,xRanges[2]),␣
 ↪label="10-year risk-neutral",color="#a3e3ff",style=:dash)
```

[9]:



Focusing on one state variable shows that the results are similar to the single variable case.