Heterogeneous-Agents Macroeconomics in Continuous Time[1]

Lecture Notes[2]

Riccardo A. Cioffi

Preliminary and incomplete

This version: September 28, 2025

Latest version at: https://github.com/ric-cioffi/hact-notes

[1]These lecture notes were developed as part of the teaching material for a PhD class taught at the Paris School of Economics. The material *heavily* borrows from a class taught by Benjamin Moll while I was a student at Princeton, as well as from other classes taught there, most notably by Gianluca Violante and Jonathan Payne. I also received a lot of help from one of my students, Leonardo Endrizzi, as well as other PSE students who provided very useful feedback. While these notes would not have been possible without them, it goes without saying that all errors are exclusively mine. If you find error, typos, or would simply like to discuss the material in these notes please write me an email at riccardo.cioffi@psemail.eu

Contents

Chapter 1

Deterministic Optimal Control

This lecture serves mostly as an introduction to continuous time modeling.[1] Students who have already worked in continuous time can likely skip it, with the only possible exception being section 1.2.2 where we go over the numerically approximation of a system of linear ordinary differential equations using a finite-difference method.

Also, we hereby use the neoclassical growth model as a running example, but our analysis of the model itself is quite minimalistic. If you're not familiar with this specific model and would like to know more, Acemoglu (2008, Chapter 8) has a more in-depth treatment.

1.1. Hamiltonian

In general, a deterministic continuous-time optimal control problems can be written as:

$$v(x_0) = \max_{\alpha(t)} \int_0^\infty e^{-\rho t} r(x(t), \alpha(t)) dt$$

$$\text{s.t. } \dot{x}(t) = f\left(x(t), \alpha(t)\right)$$

for $t \geq 0$ and $x(0) = x_0$ given.[2]

The value function $v(x_0)$ is the maximum with respect to a control vector, $\alpha(t) \in A \subseteq \mathbb{R}^k$, of an instantaneous return function, $r : X \times A \to \mathbb{R}$, which depends on the state vector, $x(t) \in X \subseteq \mathbb{R}^m$, and the control vector, discounted at a constant rate $\rho \geq 0$.

The state vector x is assumed to evolve according to some differential equation $\dot{x}(t) = f(x(t), \alpha(t))$ for $t \geq 0$, where $f : X \times A \to \mathbb{R}^m$ is called the law of motion for the state. Then, to close the problem, we also usually need an initial (or terminal) condition on the state. In this specification $x(0) = x_0$ is the initial condition.

[1] Due to the nature of this document being lecture notes, chapters will sometimes be equivalently referred to as "lectures".

[2] Dotted variable are generally used to indicate the derivative with respect to time, with one dot indicating the first derivative $\dot{x}(t) \equiv \frac{\partial x(t)}{\partial t}$, two dots indicating the second derivative etc.

In order to solve this problem, we can write down the so-called current value Hamiltonian $\mathcal{H}$:

$$\mathcal{H}\left(x(t), \alpha(t), \lambda(t)\right) = r\left(x(t), \alpha(t)\right) + \lambda(t) f\left(x(t), \alpha(t)\right)$$

which is given by the sum of the instantaneous return function, $r(x(t), \alpha(t))$, and the product of the law of motion for the state, $f(x(t), \alpha(t))$, and a so-called co-state vector $\lambda(t) \in \mathbb{R}^m$.[3]

From this, we can obtain a set of conditions that are both necessary (under some regularity conditions) and sufficient (under regularity and concavity conditions) for optimality. These conditions are:

$$\mathcal{H}_\alpha(x(t), \alpha(t), \lambda(t)) = 0 \tag{1.1}$$

$$\mathcal{H}_x(x(t), \alpha(t), \lambda(t)) = -\dot{\lambda}(t) + \rho\lambda(t) \tag{1.2}$$

$$\mathcal{H}_\lambda(x(t), \alpha(t), \lambda(t)) = \dot{x}(t) \tag{1.3}$$

that is,

- Equation (1.1) requires that the derivative of the Hamiltonian $\mathcal{H}$ with respect to the control vector $\alpha(t)$ equals 0, and is the equivalent of the standard FOC for maximization.
- Equation (1.2) requires that the derivative of $\mathcal{H}$ with respect to the state vector $x(t)$ equals minus the time derivative of the co-state vector plus the discount rate times the co-state vector.[4]
- Equation (1.3) requires that the derivative of $\mathcal{H}$ with respect to the co-state vector $\lambda(t)$ equals the law of motion for the state $\dot{x}(t)$ and it simply boils down to rewriting the constraint $\dot{x} = f(x, \alpha)$

This gives us a system of three differential equations in three unknowns which, together with boundary conditions for the state and co-state variables, will determine the solution to our problem. When it comes to the boundary conditions, we already have an initial condition for the state variable $x(0) = x_0$ which we now pair with a transversality condition for the co-state variable $\lambda(t)$:

$$\lim_{t\to\infty} e^{-\rho t}\lambda(t)x(t) = 0 \tag{1.4}$$

which requires that the discounted product between state and co-state variables equals zero. In other words, the transversality condition requires that neither the state nor the co-state variable "explodes" to infinity.[5]

[3] Just as the multiplier in standard Lagrangean maximization, the co-state vector $\lambda(t)$ is simply an auxiliary variable that is useful to solve the problem, though as we will see it will often also have a natural economic interpretation. Notice also that the co-state vector $\lambda(t)$ has the same dimensionality of the state vector $x(t)$.

[4] In other disciplines (especially physics) eq. (1.2) is often written as $\mathcal{H}_x(\cdot) = -\dot{\lambda}(t)$. The difference is due to the way that we defined the Hamiltonian and the fact that we are expressing everything in "current-value" terms. Naturally, the solution is exactly identical.

[5] Where by "explode" we mean that their product cannot grow at a rate bigger than ρ.

It is also important to note that, while we have an *initial* condition for the state $x(t)$, we effectively only have a *terminal* condition for the co-state $\lambda(t)$ in eq. (1.4). This means that the value of the co-state at time $t = 0$, $\lambda(0)$, is actually free (i.e. not predetermined). As we will see, this is particularly important because it is what will allow us to get a (stable) solution to our problem.

1.2. The Neoclassical Growth Model

To make our approach operational, we can apply it to the standard Neoclassical Growth Model.[6] The model consists of a representative household with utility function:

$$U = \int_0^\infty e^{-\rho t} u(c(t)) dt$$

where $u(c(t))$ is the instantaneous utility function of (per-capita) consumption $c(t) \geq 0$ and $\rho \geq 0$ is the discount rate (as opposed to the discount factor β in the corresponding discrete-time model).[7]

The technology in the economy is:

$$y(t) = F(k(t)) \tag{1.5}$$

with $y(t)$ being output, $k(t) \geq 0$ the capital stock, and $F(k(t))$ the production function – which is assumed to be increasing, $F'(k) > 0$, and concave, $F''(k) < 0$. The law of motion for the capital stock is given by:

$$\dot{k}(t) = i(t) - \delta k(t) \tag{1.6}$$

where $i(t)$ represents investment and δ is the capital depreciation rate. The household's budget constraint is given by:

$$c(t) + i(t) = y(t) \tag{1.7}$$

Finally, to close the model we assume that the economy start at period $t = 0$ with an initial capital stock $k(0) = k_0$.

We can then write the problem in sequential form as:

$$\begin{aligned} v(k_0) = \max_{c(t)} & \int_0^\infty e^{-\rho t} u(c(t)) dt \\ & \dot{k}(t) = F(k(t)) - \delta k(t) - c(t) \\ \text{s.t.} & \\ & k(0) = k_0 \end{aligned}$$

where, in terms of the notation introduced before, we have:

[6] For a more detailed explanation of the model, see Acemoglu 2008, Chapter 8.

[7] If you are new to continuous-time models, recall that the discount factor and the discount rate are inversely related: a higher discount factor means that the household is more patient, where instead a higher discount rate instead means that the household is *less* patient.

- Consumption $c(t)$ is the control vector $\alpha(t)$.
- The utility function $u(c(t))$ is the return function $r(x(t), \alpha(t))$.
- The capital stock $k(t)$ is our state vector $x(t)$, which also means the law of motion for capital gives us the law of motion for the state:

$$f(x(t), \alpha(t)) = F(k(t)) - \delta k(t) - c(t)$$

In order to simplify notation, from now on whenever not strictly necessary we will avoid writing the time dependence of variables; it is implied that all variables are a function of time so for example when writing k we really mean $k(t)$.

We now have all the elements to apply the Hamiltonian approach of the previous section. Just as before, we start by writing the present-value Hamiltonian $\mathcal{H}$ as the return function plus the law of motion for the state multiplied by the co-state λ:

$$\mathcal{H}(k, c, \lambda) = u(c) + \lambda \left[F(k) - \delta k - c\right].$$

The derivatives with respect to c, k and λ are given by:

$$\begin{aligned}
\mathcal{H}_c(k, c, \lambda) &= u'(c) - \lambda \\
\mathcal{H}_k(k, c, \lambda) &= \lambda \left[F'(k) - \delta\right] \\
\mathcal{H}_\lambda(k, c, \lambda) &= F(k) - \delta k - c
\end{aligned}$$

and, after imposing the necessary and sufficient conditions for optimality (and rearranging) we get:

$$\begin{aligned}
\lambda &= u'(c) \\
\dot{k} &= F(k) - \delta k - c \\
\dot{\lambda} &= \rho\lambda - \lambda \left[F'(k) - \delta\right]
\end{aligned}$$

where, just to reiterate, the last two equations essentially give us back the laws of motion for the state and co-state variables, respectively.

To solve the model, we then have to combine the necessary and sufficient conditions for optimality with the initial condition for the state $k(0) = k_0$ and the transversality condition for the co-state $\lim_{t\to\infty} e^{-\rho t}\lambda(t)k(t) = 0$. Together with the FOCs, we therefore have a system of first-order ordinary differential equations (ODE) that can be easily solved to find the optimal consumption and capital stock paths.

In order to gain some intuition about our problem, the first thing that we will do is to draw a so-called phase diagram of the system, which is possible because we are in the one dimensional case (the state variable $x(t)$ is a scalar). The phase diagram is particularly useful because it describes the evolution of the system, which will deliver us some insights on the dynamics of the model (and on its solution). Generally, phase diagrams are often drawn in the (state, co-state) space. However, in order to have a more direct interpretation,

in this specific example it will be useful to draw it in the (c, k) space instead.[8]

Let's therefore start by rearranging the system of ODEs to have them in terms of c and k (as opposed to λ and k). From the FOC $\lambda = u'(c)$, using the chain rule, we can obtain the derivative of λ with respect to t as:

$$\dot{\lambda}(t) = u''(c(t))\dot{c}(t)$$

and we can rewrite the law of motion for the co-state variable as:

$$u''(c)\dot{c} = u'(c)(\rho + \delta - F'(k)) \tag{1.8}$$

which the alert reader might recognize as nothing else than the consumption Euler equation and which is, with the appropriate adjustments, exactly the same as it would be in the discrete time formulation of the model. Similarly, we can also rewrite the transversality condition as:

$$\lim_{t\to\infty} e^{-\rho t}u'(c(t))k(t) = 0 \tag{1.9}$$

Finally, to further simplify the system, recall that the coefficient of relative risk aversion is simply given by:

$$\gamma(c) = -\frac{u''(c)c}{u'(c)}$$

which means we can rewrite the Euler equation (1.8) as:

$$\frac{\dot{c}}{c} = \frac{1}{\gamma(c)}(F'(k) - \rho - \delta).$$

and the system that we are going to solve is therefore given by:

$$\frac{\dot{c}}{c} = \frac{1}{\gamma(c)}(F'(k) - \rho - \delta) \tag{1.10}$$

$$\dot{k} = F(k) - \delta k - c \tag{1.11}$$

together with the initial condition $k(0) = k_0$ and the transversality condition (1.9).

1.2.1. Phase Diagram

Figure 1.1 plots the phase diagram in the consumption-capital space. It represents how any point in the (c, k)-space evolves over time according to our system given by eqs. (1.10) and (1.11). The colors of the arrows indicate the speed at which each point moves in the state space (i.e. essentially the values of $\dot{c}$ and $\dot{k}$): darker colors (blue and violet) represent lower speeds, and lighter colors (red and yellow) higher speeds. So, by taking any point in the phase diagram as an initial condition and following the arrows, the phase diagram tells us how consumption and capital will evolve over time.[9]

[8] If you are not familiar with phase diagrams, section A.1 describes how to draw one.

[9] In fact, by simply looking at fig. 1.1 we can already see that there is something going on in the middle of the graph (at about $k \approx 5$ and $c \approx 1.4$) as there seems to be a sort of basin where the direction of the

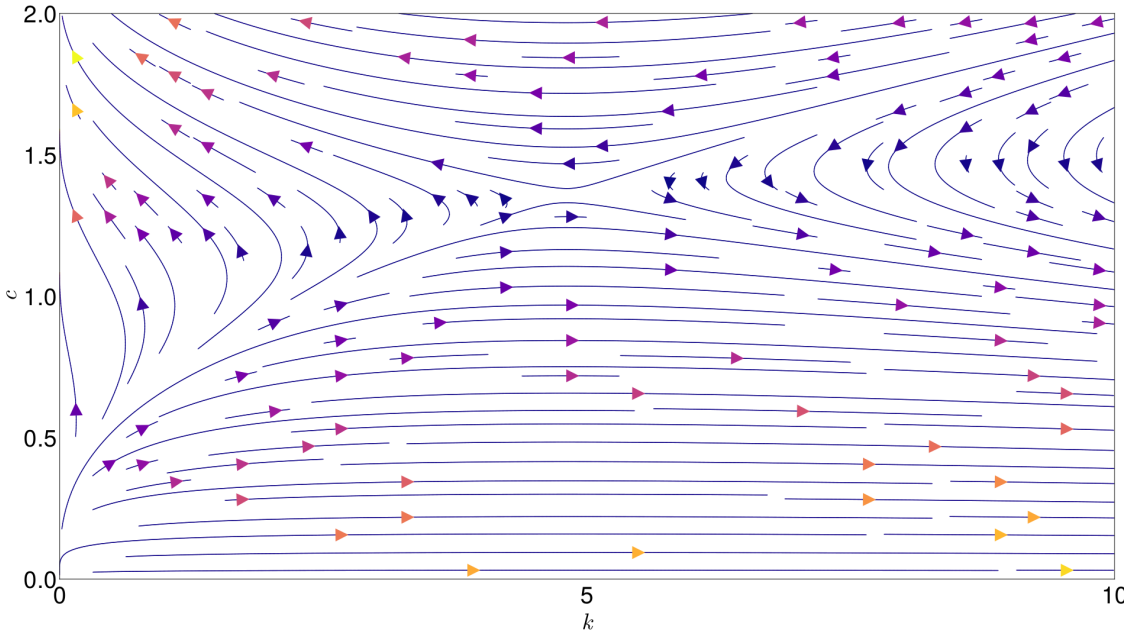

Figure 1.1: Phase diagram

Before looking at the actual solution of the system, it is instructive to better understand what the phase diagram tells us:

1. If the system were to start from a sufficiently low level of initial consumption $c(0)$ (for any level of capital $k(0)$), the phase diagram tells us that the consumption-capital pair will start going to the right and eventually consumption will start to decrease while capital continues to increase.[10] In fact, it can be shown that, starting from such initial condition, consumption will eventually reach 0 in finite time, $c(T) = 0$, while capital will converge to a finite level, $k(T) = k_{\max}$. However, this cannot be a solution to our system because it would violate the transversality condition eq. (1.9). So any pair of points (c_0, k_0) in which initial consumption is "too low" (recall that $k(0) = k_0$ is given) cannot be a solution of our problem because it would eventually violate the transversality condition; what we mean by "too low" will be evident shortly.

2. If instead the system were to start from a sufficiently high level of consumption $c(0)$, the phase diagram tells us that the consumption-capital pair will start going up and eventually capital will start to decrease while consumption continues to increase. In fact, it can be shown that, starting from such initial condition, capital will eventually reach 0 while consumption remains positive. However, this is not possible because it would violate the feasibility condition (zero capital implies zero production which, by the budget constraint, requires zero consumption). So any pair of points (c_0, k_0) in

arrows change. As we will see shortly, this is indeed the steady state of our system.

[10] While it is a bit hard to see from fig. 1.1, all the arrows in the south-east quadrant of the figure are actually slightly pointing downward.

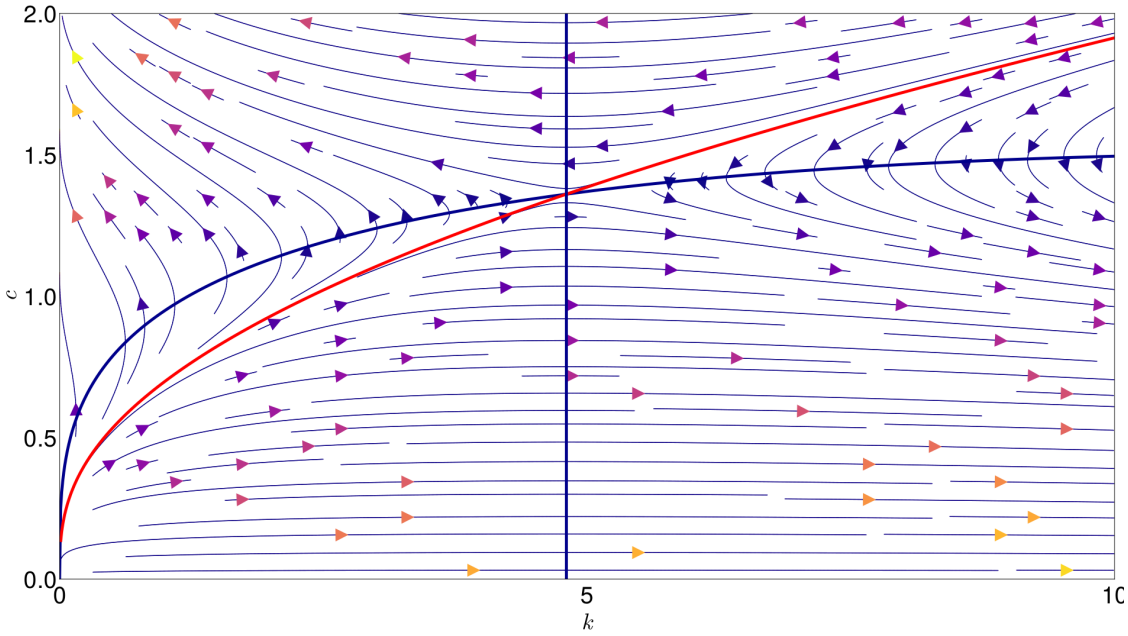

Figure 1.2: Phase diagram with saddle path

which initial consumption is "too high" cannot be a solution of our problem because it would eventually violate the feasibility condition.

Hence, our phase diagram tells us that initial consumption has to be neither too low nor too high, or the system will eventually violate either the transversality or the feasibility condition, respectively. Nonetheless, as we will see very soon, the system will in fact converge to the (unique) solution of our system provided that initial consumption is "just right". Figure 1.2 plots in red all values of initial consumption (for each level of initial capital) that converge to the solution of our system. The set of all the initial pairs (c_0, k_0) that eventually converge to the solution is also called the equilibrium path or saddle path. Before seeing how we can find such equilibrium path, it is useful to first find the steady-state to our system, i.e. the point of convergence for all consumption-capital pairs that lie on the equilibrium path.

By definition the steady state of our system will be such that the time derivatives of both consumption and capital are equal to zero, $\dot{k} = \dot{c} = 0$. Hence, from eqs. (1.10) and (1.11) we immediately get:[11]

$$\dot{c} = 0 \quad \Longrightarrow \quad F'(k^*) = \rho + \delta \tag{1.12}$$

$$\dot{k} = 0 \quad \Longrightarrow \quad c^* = F(k^*) - \delta k^*. \tag{1.13}$$

Figure 1.2 plots the solution to these two equations in blue: The vertical line is the solution to eq. (1.12) and represents the locus of points where $\dot{c} = 0$, while the concave

[11] Notice that this is exactly the same solution of the corresponding model in discrete time just with $\rho = 1/\beta - 1$.

blue line is eq. (1.13) and represents the locus of points where $\dot{k} = 0$.[12] Obviously enough, the intersection of these two lines is therefore the steady-state of our system.

If we assume a functional form for the production function, e.g. $F(k(t)) = Ak(t)^{\alpha}$, we can then find the steady state in closed form:

$$k^* = \left(\frac{\alpha A}{\rho + \delta}\right)^{\frac{1}{1-\alpha}}$$
$$c^* = A\left(\frac{\alpha A}{\rho + \delta}\right)^{\frac{\alpha}{1-\alpha}} - \delta\left(\frac{\alpha A}{\rho + \delta}\right)^{\frac{1}{1-\alpha}}$$

Given our discussion about the initial value of consumption lying above or below the saddle path, one might believe that unless initial consumption happens to lie exactly on the equilibrium path, the system will diverge. This is an incomplete, if not incorrect, characterization of the solution to the system. In fact, it is crucial to point out that the system will *always* converge to steady-state equilibrium and it is not a "knife-edge" scenario.[13] The reason is that, as we mentioned before, the co-state variable λ (and therefore consumption) is *free* at time 0. Hence, λ_0 can always freely adjust to lie exactly on the saddle path. That is to say, at time zero, for any given initial level of capital $k(0) = k_0$, consumption will "jump" to lie exactly on the saddle path. Then $(k(t), c(t))$ will travel monotonically towards the unique steady state of the system. We have therefore shown (rather informally) that there exists a unique equilibrium path, that the system converges monotonically to the unique steady state and that it does so starting from any initial capital stock k_0.

1.2.2. Numerical Solution

To conclude this lecture, let's see how to find the saddle path of the Neoclassical Growth Model using a numerical scheme.[14] Recall that the system of ODEs is given by:

$$\dot{k} = F(k) - \delta k - c$$
$$\frac{\dot{c}}{c} = \frac{1}{\gamma(c)}\left(F'(k) - \delta - \rho\right).$$

The most natural way to solve this system numerically is to simply find a way to approximate the time derivatives and to solve the system of equations forward in time starting from some initial condition (while figuring out a way to ensure the system converges to

[12] Notice in fact that the arrows crossing the $\dot{c} = 0$ line are always parallel to the k axis at that point and, similarly, the arrows crossing the $\dot{k} = 0$ line are always be parallel to the c axis at that point.

[13] A knife-edge case would represent a solution where, as if by "divine intervention", the system will converge to a stable equilibrium only if the initial point happened to be on the saddle-path (and would instead diverge for any other initial condition).

[14] While in this specific case this is obviously not necessary, at least not in order to find the steady state (since the model can be solved in closed form as we just did in section 1.2.1), it will turn out to be quite helpful as it will allow us to introduce some of the concepts that we will be using later on to solve more complicated models in an easier settings.

the steady state). Hence, we will solve the system by approximating the evolution of consumption $c(t)$ and capital $k(t)$ at N discrete points in time $t^n \in \{t^1, t^2 \dots, t^N\}$. Let's also assume that the time grid is uniformly spaced, so that we can simply denote the distance between grid points by Δt, such that $t^{n+1} = t^n + \Delta t$.

By definition the time derivative of, say, capital is given by:

$$\dot{k}(t) \equiv \frac{\partial k(t)}{\partial t} = \lim_{h \to 0} \frac{k(t+h) - k(t)}{h}$$

and identically for consumption. This means that, if we choose a sufficiently small Δt, we can approximate this derivative as:

$$\dot{k}(t^n) \approx \frac{k(t^n + \Delta t) - k(t^n)}{\Delta t}. \tag{1.14}$$

Equation (1.14) is an example of a so-called finite-difference approximation; we will use such approximations extensively throughout this course.

Using the same approximation for $\dot{c}$, we can then write the (approximation of the) entire system of ODEs as:

$$\begin{aligned} \frac{k^{n+1} - k^n}{\Delta t} &= F(k^n) - \delta k^n - c^n \\ \frac{c^{n+1} - c^n}{\Delta t} \frac{1}{c^n} &= \frac{1}{\gamma(c^n)} \left(F'(k^n) - \delta - \rho \right) \end{aligned}$$

where we have used notation $k^n = k(t^n)$ and $c^n = c(t^n)$. We can therefore rewrite this system as:

$$k^{n+1} = \left(F(k^n) - \delta k^n - c^n \right) \Delta t + k^n \tag{1.15}$$

$$c^{n+1} = \frac{c^n}{\gamma(c^n)} \left(F'(k^n) - \delta - \rho \right) \Delta t + c^n \tag{1.16}$$

and it is easy to see that, given a pair (k^n, c^n) we can immediately solve this system for (k^{n+1}, c^{n+1}). Hence, given any initial guess for (k^1, c^1), we can solve this system of equations forward to obtain an approximated time path of consumption and capital.

Now, since the initial condition for capital is already given we can simply set $k^1 = k_0$, and all that is left to do is to ensure that we can find the initial level of consumption such that the initial guess (k^1, c^1) lies on the saddle path. If our initial level of consumption is correct and our approximation is sufficiently precise, the system should then converge to its unique steady-state.

In order to do that, we can use a so-called shooting algorithm, which essentially follows the exact same intuition we applied when analyzing the phase diagram in fig. 1.1 and is presented in algorithm 1.

We start by defining an arbitrarily small tolerance ε, we set $k^1 = k_0$ (recall that $k(0) = k_0$) and we guess c^1. Then we iterate forward the discretized system of eqs. (1.15) and (1.16) to find the path of consumption and capital, $\{k^n, c^n\}_{n=1}^N$, up to a sufficiently

Algorithm 1 Shooting Algorithm

1: Define an arbitrarily small tolerance ε and set $k^1 = k_0$
2: Guess c^1
3: **for** $n = 1$ to N **do**
4: Find (k^{n+1}, c^{n+1}) by solving

$$k^{n+1} = (F(k^n) - \delta k^n - c^n)\,\Delta t + k^n$$
$$c^{n+1} = \frac{c^n}{\gamma(c^n)}\left(F'(k^n) - \delta - \rho\right)\Delta t + c^n$$

5: **if** $k^n \leq 0$ **or** $c^n \leq 0$ **then**
6: Update c^1 and go back to 2
7: **end if**
8: Compute the distance $\varepsilon^n = \|(k^{n+1}, c^{n+1}) - (k^n, c^n)\|$
9: **if** $\varepsilon^n < \varepsilon$ **then**
10: Exit
11: **end if**
12: **end for**

large value of N. At each time-step n we check if the pair (k^n, c^n) violates either the transversality condition, $c^n \leq 0$ – in which case we know the initial guess for consumption was too low and we need to increase it – or the feasibility condition, $k^n \leq 0$ – in which case we know the initial guess for consumption was too high and we need to decrease it.

Finally, to evaluate convergence (i.e., to understand if we have reached the steady state) we can calculate the distance between two subsequent pairs as

$$\varepsilon^n = \|(k^{n+1}, c^{n+1}) - (k^n, c^n)\|$$

and check if it is smaller than our chosen tolerance $\varepsilon^n < \varepsilon$.[15]

[15]Note that we could in principle check this condition just for $n = N$. However, we generally gain in efficiency by checking at every time-step, because this way we can stop the iterations and update our initial guess as soon as one of the two conditions is violated. In fact, we can even do better than that and stop as soon as one of the two elements of the sequence starts decreasing (but not both); this is because, by studying the phase diagram, we know that it happens only off the equilibrium path.

Chapter 2

Deterministic HJB Equations

In this lecture we will continue using the neoclassical growth model as an example. However, we will use a different solution method that directly solves the Hamilton-Jacobi-Bellman (HJB) equation of the model (i.e. the recursive formulation of the sequential problem solved in the first lecture). We will therefore see how to define the HJB equation, as well as how to solve it numerically using a finite-differences (FD) approximation of the value function.

Just as in the first lecture, since this specific model can be solved in closed form (or numerically using a shooting algorithm as we did in chapter 1), writing down the HJB is not strictly necessary to solve the neoclassical growth model *per-se*. However, doing so will allow us to introduce concepts that we will be using later on to solve more complicated models in a familiar setting.

2.1. The Hamilton-Jacobi-Bellman Equation

As a first step, we will provide a "heuristic" approach on how to derive the HJB starting from any deterministic optimal control problem. As already introduced in chapter 1, such problem can be written as

$$\begin{aligned} v(x_0) = \max_{\alpha(t)} & \int_0^\infty e^{-\rho t} r(x(t), \alpha(t)) dt \\ \text{s.t. } & \dot{x}(t) = f\left(x(t), \alpha(t)\right). \end{aligned} \tag{2.1}$$

As we saw, the current-value Hamiltonian $\mathcal{H}$ of this problem is:[1]

$$\mathcal{H}\left(x, \alpha, \lambda\right) = r(x, \alpha) + \lambda f(x, \alpha).$$

Shortly we will provide an alternative derivation of the Hamilton-Jacobi-Bellman equation; however, we can quickly get to it using an quick-and-dirty approach by using the above

[1] We drop the dependence on t to ease notation.

Hamiltonian as:

$$\rho v(x) = \max_{\alpha} r(x,\alpha) + v'(x) f(x,\alpha). \tag{2.2}$$

which makes the connection between the Hamiltonian and the HJB equations very apparent. First, the co-state λ in the Hamiltonian has been replaced by the derivative with respect to the state variable x of the value function, $v'(x)$, in the HJB:[2]

$$\lambda(t) = v'(x(t)).$$

Second, notice that if we write down the maximized Hamiltonian $\hat{\mathcal{H}}(x,p)$

$$\hat{\mathcal{H}}(x,p) \equiv \max_{\alpha} r(x,\alpha) + p f(x,\alpha),$$

then the HJB equation can be written as:

$$\rho v(x) = \hat{\mathcal{H}}(x, v'(x)).$$

That is, the Hamilton-Jacobi-Bellman equation is given by the maximized Hamiltonian in which the co-state variable equals the derivative of the value function.[3]

Let's now apply the same approach to write the HJB of the neoclassical growth model. Recall that the Hamiltonian for that model is given by:

$$\mathcal{H}(k,c,\lambda) = u(c) + \lambda\left[F(k) - \delta k - c\right]$$

which means we can directly write the HJB equation as:

$$\rho v(k) = \max_{c} u(c) + v'(k)\left(F(k) - \delta k - c\right). \tag{2.3}$$

In order to solve the model, in section 2.2 we will see how to discretize the HJB eq. (2.3) directly to find the value function. However, with eq. (2.3) on hand, we can in principle follow the same steps as in discrete time to get to an Euler equation: take first order conditions, apply the envelope condition, and put them together to derive the Euler equation. To do so, we start by taking the derivative with respect to consumption to get the first order condition (FOC):

$$u'(c) = v'(k) \tag{2.4}$$

which, incidentally, also implies that $\lambda = u'(c)$ just as in chapter 1. Then, we write the usual envelope condition:[4]

$$\rho v'(k) = v''(k)\underbrace{\left(F(k) - \delta k - c\right)}_{\dot{k}} + v'(k)\left(F'(k) - \delta\right). \tag{2.5}$$

[2] As we will see in a second, this is in fact also going to be equal to the marginal utility of consumption, just as in chapter 1.

[3] Note also that, in this specific problem, the state variable is one dimensional. If this was not the case, in place of the derivative of the value function, we would have the gradient with respect to the state vector (and the dot product between the gradient and the law of motions for the different states).

[4] If you are not comfortable with the envelope theorem, see section B.1 for a quick review.

In order to put the two eqs. (2.4) and (2.5) together, we can first differentiate both sides of eq. (2.4) with respect to time to obtain:

$$u''(c)\dot{c} = v''(k)\dot{k}.$$

Then, plugging this last equation in the envelope condition (2.5) we get:

$$\rho u'(c) = u''(c)\dot{c} + u'(c)\left(F'(k) - \delta\right)$$

which we can rearrange to obtain the exact same Euler equation that we got using the Hamiltonian approach:[5]

$$\frac{\dot{c}}{c} = -\frac{u'(c)}{u''(c)c}\left(F'(k) - \delta - \rho\right). \tag{2.6}$$

Theoretical results — To conclude this section it is useful to at least mention that, just as there are theoretical results about existence and uniqueness of the solution to discrete-time Bellman equations, there exist also similar results about the HJB equations in continuous time.[6] Since a treatment of these theoretical results is far beyond the scope of these notes, we hereby just briefly mention them:

Theorem 1: *The HJB eq.* (2.2) *has a unique "nice" solution.*

Theorem 2: *The solution to the HJB eq.* (2.2) *equals the value function in eq.* (2.1).

While in this version these theorems are extremely vague (and deliberately so), they are still very important because they ensure that the we can find the solution we are looking for: on the one hand, the first theorem makes sure that the solution to the HJB is unique and has certain desirable properties – specifically, by "nice" we mean that the solution to the HJB is a viscosity solution;[7] on the other, the second theorem ensures us that, by finding the solution to the recursive problem, we also find the solution to our problem of interest (namely, the sequential problem).

2.1.1. An Alternative Derivation

An alternative way to derive the HJB equation which does not make use of the Hamiltonian instead involves simply start from its discrete-time counterpart. While this approach has the advantage of being more intuitive for people used to working with discrete-time models, it may also lead to incorrect results if one is not very careful (especially once we will introduce uncertainty in chapter 3) In any case, since it can also be quite instructive, it is useful to do it at least once.

[5] The Euler Equation (2.6) is obviously identical to eq. (1.8) derived in chapter 1, so we could in theory use the same solution method here.

[6] For a review of such results in discrete time, see Stokey et al. 1989, Chapter 4.

[7] For an introduction to the concept of viscosity solution, the interested reader can refer to Appendix D of Achdou et al. 2022, and the references therein.

Let's start by working with the discrete-time model and by assuming a time period of length Δ.[8] We also assume that the discount factor is in the exponential form $\beta(\Delta) = e^{-\rho\Delta}$. The discrete-time Bellman equation is given by:[9]

$$\begin{aligned} v(k_t) &= \max_{c_t} \ \Delta u(c_t) + e^{-\rho\Delta} v(k_{t+\Delta}) \\ \text{s.t. } & k_{t+\Delta} = \Delta\left(F(k_t) - \delta k_t - c_t\right) + k_t \end{aligned}$$

which obviously boils down to the standard Bellman equation if we set $\Delta = 1$.

For small values of Δ, we can approximate the discount factor as:

$$e^{-\rho\Delta} \approx 1 - \rho\Delta$$

and, as a consequence, the HJB can also be approximated as:

$$v(k_t) \approx \max_{c_t} \Delta u(c_t) + (1 - \rho\Delta)\, v(k_{t+\Delta}).$$

Subtracting $(1 - \rho\Delta)v(k_t)$ and dividing by Δ on both sides leads to

$$\rho v(k_t) \approx \max_{c_t} u(c_t) + u(c_t) + (1 - \rho\Delta)\, \frac{v(k_{t+\Delta}) - v(k_t)}{\Delta}.$$

Finally, multiply and divide the last term on the right-hand side by $k_{t+\Delta} - k_t$ to obtain:

$$\rho v(k_t) \approx \max_{c_t} u(c_t) + (1 - \rho\Delta)\, \frac{v(k_{t+\Delta}) - v(k_t)}{k_{t+\Delta} - k_t} \frac{k_{t+\Delta} - k_t}{\Delta}$$

which, taking the limit for $\Delta \to 0$, gives us the HJB equation as in eq. (2.3):

$$\rho v(k_t) = \max_{c_t} u(c_t) + v'(k_t)\dot{k}_t$$

2.2. Numerical Solution

Except in some specific cases it is often impossible to find an analytical solution to the HJB equation. However, in the same way that there exists a well-developed theory for the *theoretical* properties of HJB equations, there also exists a well-developed theory on how to find the *numerical* solution of the HJB. In particular, this theory tells us that if we define an "appropriate" approximation scheme for the HJB, such scheme will converge to the unique viscosity solution.[10] In terms of the approximation scheme, we will focus on so-called finite-difference (FD) methods, which are arguably the simplest and most

[8] Eventually, we will let Δ go to zero to get the continuous-time limit of our model.

[9] We're changing notation with respect to time by writing $x(t)$ as x_t to stay consistent with the usual discrete time notation.

[10] In order for it to converge to the viscosity solution, the "appropriately" defined approximation scheme needs to satisfy the following three main conditions: monotonicity, consistency, and stability. Of these three conditions, consistency and stability are usually easily satisfied; while monotonicity is sometimes a bit more challenging to prove. See Barles and Souganidis (1991) for more details.

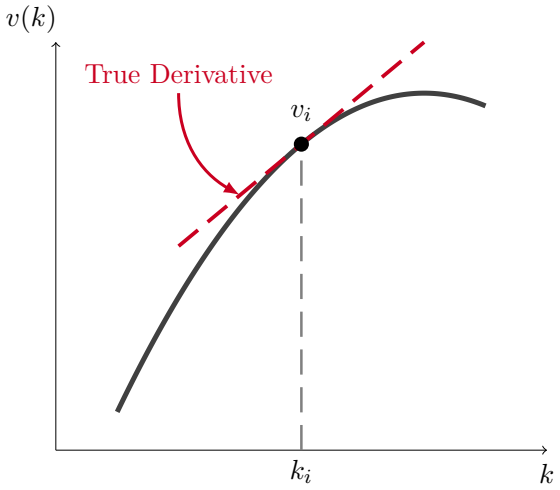

Figure 2.1: True Derivative

easily applicable.[11] However, it is worth mentioning that there exist also other well-developed methods (used mostly in other disciplines) such as finite-element, finite-volume, and spectral methods, among many others.

2.2.1. Finite-differences

When we were looking for a solution to the neoclassical growth model in chapter 1 we wrote down the system of ODEs that characterized our system and discretized it along the time dimension. However, in our HJB eq. (2.3) – which we rewrite here for convenience:

$$\rho v(k) = \max_c u(c) + v'(k)\left(F(k) - \delta k - c\right)$$

we can immediately observe that the value function depends on capital, and not on time. As a consequence, when looking for a solution to eq. (2.3), it seems natural to want to discretize the state space of our value function, i.e. capital k. In particular, since the HJB equation also includes the derivative of the value function with respect to capital, we will need to approximate that derivative. As we will see, much of the discussion in the rest of this chapter will be about how to define such an approximation in an "appropriate" way.

We start by defining a grid for capital $\{k_i\}_{i=1}^I$. Approximating the value function $v(k)$ itself is rather trivial because it suffices to evaluate it at each discrete point on the grid k_i. Approximating the derivative $v'(k)$ requires a bit more thought. In fact, if we want to find such an approximation by using our approximated value function, at least three options come to mind:[12]

- Backward difference: We approximate the derivative at point k_i by using the ap-

[11] A good entry-level reference for finite-difference methods in economics and finance is Tourin (2013).
[12] We hereby use the shorthand notation $v_i \equiv v(k_i)$

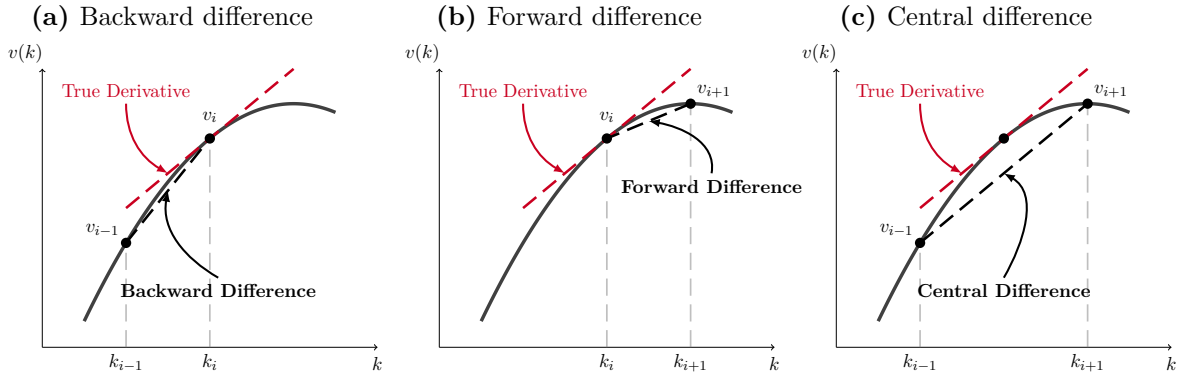

Figure 2.2: Finite Difference Approximations

proximated value function at points k_i and k_{i-1} (see fig. 2.2a):

$$v'_{i,B} = \frac{v_i - v_{i-1}}{\Delta k} \approx v'(k_i)$$

- Forward difference: We approximate the derivative at point k_i by using the approximated value function evaluated at points k_i and k_{i+1} (see fig. 2.2b):

$$v'_{i,F} = \frac{v_{i+1} - v_i}{\Delta k} \approx v'(k_i)$$

- Central difference: We approximate the derivative at point k_i by using the value function evaluated at points k_{i-1} and k_{i+1} (see fig. 2.2c):

$$v'_{i,C} = \frac{v_{i+1} - v_{i-1}}{2\Delta k} \approx v'(k_i)$$

These three approaches are justified by the fact that we can easily evaluate the value function at point k_i – assuming we know it – and then, in order to approximate the derivative at that the same point, we simply "move away" from v_i in a given direction. Because we can in principle move backward, forward, or both we have the above three choices. Figure 2.2 shows graphically how these three approaches work.

Of course our derivative approximation will get more and more accurate as the grid gets finer and finer (i.e., with smaller Δk); however, this obviously comes at the cost of having to evaluate the value function at more points. While we certainly want to find a scheme that converges to the true solution as the grid gets finer, we are also particularly interested in finding a scheme that allows us to have a better approximation of the true derivative *conditional* on the size of the grid.[13]

[13]Since we mentioned grid density, it might be worth noting that, if we know that the value function is going to have more curvature in some specific regions of the state space, it is often useful to use a non-uniform grid that concentrates more points in that region. In order to make the exposition simpler, throughout these notes we will always be using uniform grids; however, most results apply rather straight-

Applying our finite-difference approximations to the HJB eq. (2.3) we have:

$$\rho v_i = u(c_i) + v_i' \cdot (F(k_i) - \delta k_i - c_i) \tag{2.7}$$

where c_i can be "extracted" from the FOC of our problem: $c_i = (u')^{-1}(v_i')$ and v_i' is the approximated derivative of the value function $v'(k)$ at k_i obtained using one of backward, forward or central FD approximations.

There are essentially two problems that we will need to solve in order to find a solution to eq. (2.7): The first one is related to which approximation scheme we should use. The finite difference scheme that we choose needs to satisfy certain monotonicity, consistency, and stability conditions to guarantee convergence of the numerical solution to (2.7) to the true viscosity solution to (2.3). Our chosen approximation scheme will be the so-called upwind scheme, which we cover in section 2.2.2.

The second problem that we face is related to the non-linearity of the HJB equation. Such non-linearity is made apparent for example by the fact that $c_i = (u')^{-1}(v_i')$. Essentially, in eq. (2.7) v_i is defined as a non-linear function of v_i', which is itself however defined as a function of v_i. We will solve this problem by using an iterative scheme, similarly to what we did in the shooting algorithm of chapter 1: we will start with a guess for the value function, which will allow us to compute its derivative and, as a consequence, also consumption. These two together will allow us to update our guess for the value function and to iterate this procedure until it hopefully converges. As we will see there are essentially two ways of doing this: by using an explicit or an implicit method.[14] While each has its own advantages and disadvantages, generally speaking, for these type of heterogeneous-agent models the advantages of using an implicit method far outweigh its possible disadvantages, so we will mostly be focusing on that in section 2.2.4.

2.2.2. Upwind Scheme

The upwind scheme essentially boils down to cleverly choosing when to use the backward- or the forward-difference to approximate the derivative of the value function. As we will see shortly, this choice will be based on the sign of the drift of the state variable. In particular, the upwind scheme will use the forward-difference when the drift is positive and the backward-difference when it is negative. However, while this sounds straightforward in theory, there is one major caveat: the drift itself also depends on which approximation we choose. To see this, recall that the drift of capital is given by:

$$\dot{k} = F(k) - \delta k - c,$$

and that from the FOC we have that $c = (u')^{-1}(v')$. Hence consumption (and therefore $\dot{k}$) is a function of the derivative v'. Because of that, we already have two ways of defining

forwardly in the case of non-uniform grids. In section B.2 we show one way of creating such a non-uniform grid.

[14] While we ignored this issue in our earlier treatment of the shooting algorithm, the same distinction could have been made there as well.

the drift of our state variable: using the forward difference or the backward difference; since the sign of the drift might differ depending on which approximation we use, one has to be a bit careful in the way the scheme is designed.

Let's start by defining the drift of the state variable in these two cases as:

$$s_{i,F} = F(k_i) - \delta k_i - (u')^{-1}(v'_{i,F})$$
$$s_{i,B} = F(k_i) - \delta k_i - (u')^{-1}(v'_{i,B})$$

which for brevity we will refer to as the forward-drift and the backward-drift, respectively (we use the letter s for savings).

As we just said, the upwind scheme tells us to:

- Use the forward difference if the drift is positive
- Use the backward difference if the drift is negative

Intuitively, if the drift is positive, it means that our state variable is increasing and the derivative of our function of interest is likely better approximated using the forward difference (and viceversa if the drift is negative); that is, we use the derivative approximation in the direction in which the state is moving.

In the above definition of the upwind scheme, however, we have not specified whether we refer to the backward- or the forward-drift. In most cases, as long as our grid is sufficiently dense, these two should be sufficiently close to each other as to have the same sign. However, there might be cases in which they are not. To account for all possible cases, what we do is calculate both the backward- and forward-drift, $s_{i,B}$ and $s_{i,F}$, and assess their sign:

- If both are positive
$$s_{i,F} > 0 \quad \text{and} \quad s_{i,B} \geq 0,$$
then we approximate v' using the forward derivative:
$$\dot{k} > 0 \implies v'_i = v'_{i,F}, \quad c_{i,F} = (u')^{-1}(v'_{i,F}), \quad s_i = s_{i,F}.$$
- If both are negative
$$s_{i,F} \leq 0 \quad \text{and} \quad s_{i,B} < 0$$
then we approximate v' using the backward derivative:
$$\dot{k} > 0 \implies v'_i = v'_{i,B}, \quad c_{i,B} = (u')^{-1}(v'_{i,B}), \quad s_i = s_{i,B}$$
- Finally, if the forward drift is negative and the backward drift is positive
$$s_{i,F} \leq 0 \quad \text{and} \quad s_{i,B} \geq 0$$
then, intuitively, we know we should be at a "steady-state" (really, a maximum): if we were to use the forward drift it would tell us to "go backward" and if we were to

use the backward drift it would tell us to "go forward". Since we are at steady-state, our state variable is not moving, i.e. $\dot{k} = 0$, and our drift is zero

$$\dot{k} = 0 \implies s_i = 0, \quad c_{i,0} = F(k_i) - \delta k_i, \quad v_i' = u'(F(k_i) - \delta k_i)$$

that is, capital is constant, which means we can use the law of motion for capital to obtain the value of consumption and then find our approximation of the derivative of the value function directly from the FOC.

To recap, our approximation of the derivative of the value function will be given by the forward difference if the forward-drift is positive, by the backward difference if the backward-drift is negative, and by its steady-state value when the two drifts have opposite signs. In shorter terms:

$$v_i' = v_{i,F}' \cdot \mathbf{1}_{\{s_{i,F}>0\}} + v_{i,B}' \cdot \mathbf{1}_{\{s_{i,B}<0\}} + \bar{v}_i' \cdot \mathbf{1}_{\{s_{i,F}\leq 0 \leq s_{i,B}\}}$$

where $\bar{v}_i' = u'(F(k_i) - \delta k_i)$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function.[15]

The careful reader might have noticed that we did not cover the case when the forward-drift is positive *and* the backward-drift is negative. In general, as long as the value function v is increasing and concave, this should never happen. In fact, if v is concave, we will generally have $v_{i,F}' < v_{i,B}'$ (e.g. compare figs. 2.2a and 2.2b) which implies that the forward drift will generally be less than the backward drift: $s_{i,F} < s_{i,B}$.[16] In principle, however, if the value function is (locally or globally) convex at k_i it could be the case that $s_{i,F} > 0$ and $s_{i,B} < 0$.[17] This obviously begs the question of what to do in this case. One of the nicest properties about the upwind scheme is that it works also in this case (with the appropriate adjustments); all we have to do is to choose in which "direction" to go by looking at the value of the Hamiltonian instead of looking at the sign of the drift. Specifically, after

[15] When searching for references on the upwind scheme, one might find it to be defined the other way around; that is, by using the derivative on the side *opposite* to that of the drift (or flow) of a variable. The reason for this discrepancy is in the direction of time. As we will see in the coming lectures, the HJB is a *backward* equation in time; that is, it can be solved starting from a terminal condition and moving back in time. Because the bias in the upwind scheme needs to change side based on the direction of time, it is defined differently for backward and forward equations (and most general treatments usually start from forward equations).

[16] This is due to the fact that marginal utility is decreasing:

$$(u')^{-1}(v_{i,F}') > (u')^{-1}(v_{i,B}')$$

which, when calculating the drifts, implies:

$$s_{i,F} = F(k_i) - \delta k_i - (u')^{-1}(v_{i,F}') < s_{i,B} = F(k_i) - \delta k_i - (u')^{-1}(v_{i,B}')$$

[17] Since we will always be approximating $v(k)$ with v_i, it is actually possible that along the convergence path our approximation becomes locally convex even if the true objective function is concave.

defining the forward Hamiltonian, $H_{i,F}$, and the backward Hamiltonian, $H_{i,B}$, as

$$H_{i,F} \equiv u(c_i) + v'_{i,F}\left[F(k_i) - \delta k_i - c_i\right] \tag{2.8}$$

$$H_{i,B} \equiv u(c_i) + v'_{i,B}\left[F(k_i) - \delta k_i - c_i\right] \tag{2.9}$$

the upwind scheme can be adjusted by using the forward difference if the forward Hamiltonian is greater than the backward Hamiltonian and the backward difference in the opposite case (however, one also needs to check if setting the drift to zero gives a higher Hamiltonian). In section B.3 we look at one example featuring non-convexities in which this distinction is particularly important.

2.2.3. Discretized HJB Equation

Now that we have an approximation of the derivative of the value function, we can also write the approximation of the HJB equation:

$$\rho v_i = u(c_i) + \frac{v_{i+1} - v_i}{\Delta k} \cdot s_{i,F} \cdot \mathbf{1}_{\{s_{i,F}>0\}} + \frac{v_i - v_{i-1}}{\Delta k} \cdot s_{i,B} \cdot \mathbf{1}_{\{s_{i,B}<0\}}$$

where $c_i = c_{i,F} \cdot \mathbf{1}_{\{s_{i,F}>0\}} + c_{i,B} \cdot \mathbf{1}_{\{s_{i,B}<0\}} + c_{i,0} \cdot \mathbf{1}_{\{s_{i,F}\leq 0 \leq s_{i,B}\}}$ is computed using the same approximation of the derivative, and we have not included the third case ($s_{i,F} \leq 0$ and $s_{i,B} \geq 0$) on the right-hand side because in that case $s_i = 0$ by construction. The approximated HJB can also be written, in shorter notation, as:

$$\rho v_i = u(c_i) + v'_{i,F} \cdot s^+_{i,F} + v'_{i,B} \cdot s^-_{i,B} \tag{2.10}$$

where $x^+ = \max\{x, 0\}$ and $x^- = \min\{x, 0\}$.

It is at this point convenient to write the discretized HJB equation in matrix form as:

$$\rho \mathbf{v} = \mathbf{u} + \mathbf{A}\mathbf{v} \tag{2.11}$$

where $\mathbf{v}$ and $\mathbf{u}$ are I-dimensional column vectors, $\mathbf{A}$ is an $I \times I$ matrix, and I is the number of points in the discretized state space. Zooming in on the i^{th} row of $\mathbf{A}$ and the related components of the $\mathbf{v}$ vector, we have that the only non-zero elements of the i^{th} row are in columns $i-1, i$, and $i+1$:

$$\underbrace{\begin{pmatrix} \underbrace{-\frac{s^-_{i,B}}{\Delta k}}_{\text{inflow}_{i-1}\geq 0} & \underbrace{\frac{s^-_{i,B}}{\Delta k} - \frac{s^+_{i,F}}{\Delta k}}_{\text{outflow}_i \leq 0} & \underbrace{\frac{s^+_{i,F}}{\Delta k}}_{\text{inflow}_{i+1}\geq 0} \end{pmatrix}}_{i^{th} \text{ row of } \mathbf{A}} \begin{pmatrix} v_{i-1} \\ v_i \\ v_{i+1} \end{pmatrix}.$$

It is easy to see that the matrix representation in eq. (2.11) is equivalent to the scalar one

in eq. (2.10) by performing the matrix multiplication on the right-hand side:[18]

$$\begin{aligned}\mathbf{A}_{i,i-1}v_{i-1} + \mathbf{A}_{i,i}v_i + \mathbf{A}_{i,i+1}v_{i+1} &= -\frac{s^-_{i,B}}{\Delta k}v_{i-1} + \left(\frac{s^-_{i,B}}{\Delta k} - \frac{s^+_{i,F}}{\Delta k}\right)v_i + \frac{s^+_{i,F}}{\Delta k}v_{i+1} \\ &= s^-_{i,B}\frac{v_i - v_{i-1}}{\Delta k} + s^+_{i,F}\frac{v_{i+1} - v_i}{\Delta k} \\ &= v'_{i,F}\cdot s^+_{i,F} + v'_{i,B}\cdot s^-_{i,B}.\end{aligned}$$

The $\mathbf{A}$ matrix has several properties that it is useful to make note of:

- Its rows sum to zero.
- It is tridiagonal; that is, its only nonzero elements are on the main diagonal, the first lower diagonal (the first diagonal below the main diagonal), and the first upper diagonal (the first diagonal above the main diagonal). Importantly, this also implies that $\mathbf{A}$ is going to be an extremely sparse matrix.[19]
- It has negative elements on the main diagonal and positive elements off-diagonal. In particular recall that $s^+_{i,F} = \max\{s_{i,F}, 0\} \geq 0$ and $s^-_{i,B} = \max\{s_{i,B}, 0\} \leq 0$.
- It contains information about flows from point i to point $i-1$ and $i+1$, not too differently from a transition matrix in discrete time. These essentially tell us the rate at which households are moving out of point i towards either $i-1$ and/or $i+1$.

Figure 2.3 presents a visual representation of the $\mathbf{A}$ matrix for the neoclassical growth model, where each dot represents a non-zero element of $\mathbf{A}$, with lighter dots corresponding to smaller entries (in absolute value). By analyzing this figure we can see that $\mathbf{A}$ is extremely sparse, that it is tri-diagonal and that the upwind scheme is selecting the forward difference at the low end of the state space (in the top-left of $\mathbf{A}$ there are only elements on the upper and main diagonals) and the backward derivative at the top end of the state space (in the bottom-right of $\mathbf{A}$ there are only elements on the lower and main diagonals). Finally, the colors of the dots also show us that inflows and outflows get larger farther away from the steady state and closer to zero at the steady state (the empty spot at the center of the matrix).

Before moving on to how to handle the non-linearity of eq. (2.7), there is one final issue that needs to be at least mentioned: by construction the upwind scheme uses either the backward difference or the forward difference based on the sign of the drift; however, these are not well defined at the boundaries of the state space. In fact, we cannot really define a backward difference at the first point of our grid without a point to the left of it, and similarly we cannot define a forward difference at the last point of our grid without a point to the right of it (these are sometimes called *ghost nodes*). For the time being, this is actually not going to be an issue since the upwind scheme automatically chooses

[18] We are here making use of the fact that $\mathbf{A}$ is tridiagonal and $\mathbf{A}_{i,j} = 0\ \forall j < i-1, j > i+1$.

[19] In this specific model, out of the I^2 total elements in $\mathbf{A}$, *at most* $3I-2$ will be non-zero. Hence, if we were to discretized capital on a grid with $I = 1{,}000$ points, that would mean that less than 0.3% of the entries of $\mathbf{A}$ would be non-zero.

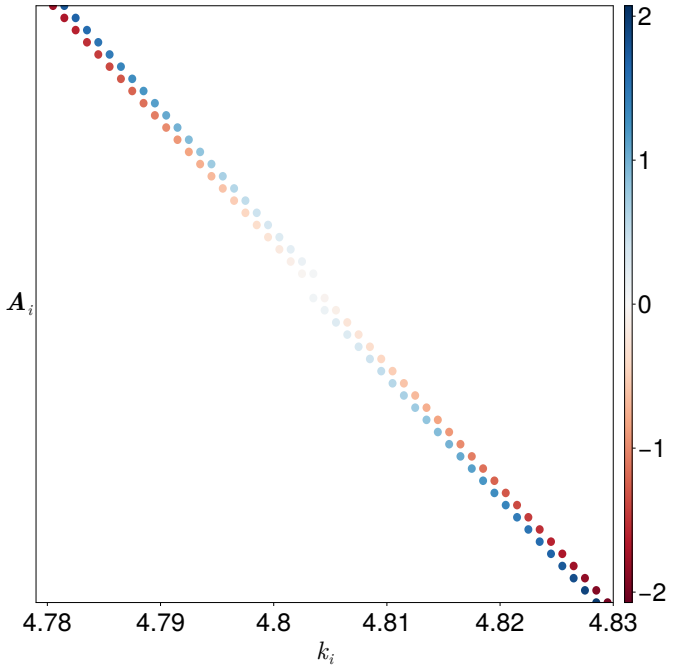

Figure 2.3: Visualization of the **A** matrix for the neoclassical growth model

the forward difference for $i = 1$ and the backward difference for $i = I$ by virtue of the fact that, due to the concavity of our problem, the drift is positive at the beginning of the state space and negative at its end (as long as we choose a sufficiently large k_I). This also makes economic sense: for low level of capital, the household will want to save, $\dot{k} > 0$, and for high level of capital, she will want to dissave, $\dot{k} < 0$. However, this is simply a property of the model we are solving and not of the upwind scheme itself. In fact, when we will go to different models in later lectures, we will have to impose so-called *boundary conditions* which deal specifically with what happens at the boundaries of the state space. We defer the discussion of boundary conditions to **??**.

2.2.4. Explicit and Implicit Methods

Until now, we have deliberately ignored an the non-linearity of the HJB equation. In fact, to be precise, both **A** and **u** are functions of **v** itself: because $s^-_{i,B}$ and $s^+_{i,F}$ are functions of **v**, **A** will also be a function of it, while **u** is clearly related to **v** through the FOC, $c = (u')^{-1}(v')$. As a consequence, a more precise definition of our approximated HJB would be:

$$\rho \mathbf{v} = \mathbf{u}(\mathbf{v}) + \mathbf{A}(\mathbf{v})\mathbf{v}.$$

This means that we cannot generally solve for **v** in "one shot", but we will need to iterate on this equation in order to find a fixed point.

In order to do so, we can use one of two options:

- An explicit method, which is easier to write down and in which each iteration is

Algorithm 2 Explicit Method

Define an arbitrarily small tolerance ε and initial guess $\mathbf{v}^0$
for $n = 0, 1, 2, \ldots$ **do**
  Given a guess v_i^n, find v_i^{n+1} by solving

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^n = u(c_i^n) + (v_i^n)' \cdot (F(k_i) - \delta k_i - c_i^n) \tag{2.12}$$

  Where $(v_i^n)'$ is the FD approximation to $v'(k_i)$ and $c_i^n = (u')^{-1}\left[(v_i^n)'\right]$
  if $\|\mathbf{v}^{n+1} - \mathbf{v}^{\mathbf{n}}\| < \varepsilon$ **then**
    Exit
  else
    Set $n = n + 1$ and go back
  end if
end for

faster to solve, but also less stable and inefficient

- An implicit method, which is slightly harder to write down and in which each iteration is slower, but is also more stable and efficient.

Both methods boil down to iterating on n in the following equation:

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^\bullet = u(c_i^n) + (v_i^\bullet)' \cdot (F(k_i) - \delta k_i - c_i^n)$$

which simply adds a convergence term on the left-hand side that hopefully goes to zero as n grows and ensures that we have reached a fixed point of eq. (2.7).[20] The difference between the explicit and implicit methods will simply be in what $v_i^\bullet$ is, where the explicit method sets $v_i^\bullet = v_i^n$ and the implicit method $v_i^\bullet = v_i^{n+1}$. As we will see, for these types of models the implicit method is generally a much better choice; however, since the explicit method is slightly more intuitive we cover it first.

Explicit method — The explicit method is formalized in algorithm 2. A few things worth noting:

1. The step-size along the "time" dimension, Δ, need not be (and in fact it should not be) the same as the step-size of the state-space discretization Δ_k

2. Equation (2.12) is **not** a contraction mapping, which means that nothing ensures us the algorithm will converge to the true solution

3. Related to the above two points: in order for it to converge, the explicit method usually requires a step-size Δ that is sufficiently small.[21]

[20] We can in fact think of these methods as being equivalent to time-iteration; that is, solving a discretized version of:

$$\dot{v} + \rho v = u + \mathcal{A}v$$

where $\mathcal{A}$ is the infinitesimal generator of the HJB (which we will define later).

[21] In the explicit method, one can in fact derive an expression for the maximum possible Δ that can

Algorithm 3 Implicit Method

Define an arbitrarily small tolerance ε and initial guess $\mathbf{v}^0$
for $n = 0, 1, 2, \ldots$ **do**
Given a guess v_i^n, find v_i^{n+1} by solving

$$\frac{v_i^{n+1} - v_i^n}{\Delta} + \rho v_i^{n+1} = u(c_i^n) + (v_i^{n+1})' \cdot (F(k_i) - \delta k_i - c_i^n) \tag{2.13}$$

Where $(v_i^n)'$ is the FD approximation to $v'(k_i)$ and $c_i^n = (u')^{-1}\left[(v_i^n)'\right]$
Solve the linear system:

$$\frac{1}{\Delta}\left(\mathbf{v}^{n+1} - \mathbf{v}^n\right) + \rho \mathbf{v}^{n+1} = \mathbf{u}^n + \mathbf{A}_n \mathbf{v}^{n+1}$$

$$\Downarrow$$

$$\mathbf{v}^{n+1} = \left(\left(\rho + \frac{1}{\Delta}\right)\mathbf{I} - \mathbf{A}_n\right)^{-1}\left(\mathbf{u}^n + \frac{1}{\Delta}\mathbf{v}^n\right)$$

if $\|\mathbf{v}^{n+1} - \mathbf{v}^{\mathbf{n}}\| < \varepsilon$ **then**
Exit
else
Set $n = n + 1$ and go back to step 2
end if
end for

Implicit method — The implicit method is formalized in algorithm 3, which we can immediately notice is very similar to algorithm 2, the only difference being that we have carefully selected where to substitute our current guess v_i^n with the next guess v_i^{n+1}.[22] Compared to the explicit method, a few differences are worth noting:

1. While eq. (2.12) is of the form $\mathbf{x} = \mathbf{Ab}$ and can therefore be solved at the expense of a single matrix multiplication, eq. (2.13) is of the form $\mathbf{Ax} = \mathbf{b}$ and requires a matrix inversion to be solved (which is obviously more complicated).

2. Related to the above, numerical softwares can usually take advantage of sparsity to ensure the matrix inversion is done efficiently. This means that, since $\mathbf{A}$ is extremely sparse, even the system of the form $\mathbf{Ax} = \mathbf{b}$ can be solved in a fraction of a second.[23,24]

be used as a function of Δ_k. Suffice here to say that, the finer is our state-space approximation (i.e. the smaller the Δ_k), the smaller our Δ will need to be to ensure convergence.

[22] In fact algorithm 2 is technically only a *semi*-implicit method, since a fully-implicit scheme would also replace c_i^n with c_i^{n+1}. However, doing so would imply that we would need to solve a non-linear system of equations at each iteration. On the other hand the semi-implicit scheme defined in algorithm 3 retains many of the useful convergence properties of a fully-implicit method while preserving the linearity of the system. Because at each iteration we now have to solve a system of *linear* equations, this means the semi-implicit method is often faster (and potentially numerically more stable).

[23] In order to take advantage of the sparsity of $\mathbf{A}$ most numerical software generally require it to be *explicitly* declared as sparse (sparse matrices are often stored and handled differently than full matrices).

[24] Note that in most numerical softwares you don't even need to perform the matrix inversion *per-se*,

3. The implicit method is stable "independently" of Δ, which means we can choose a much larger value of Δ than in the implicit method.

Effectively the sparsity of $\mathbf{A}$ means that, while a single iteration of the implicit scheme takes marginally longer than a single iteration of the explicit scheme, the possibility to choose a larger step-size Δ (i.e., larger "convergence steps") means that the system needs a lot fewer steps to converge. As a consequence, while slightly more complicated the implicit method will generally be a lot faster (and more stable) than the explicit method.

Explicit vs. Implicit approximations — To have a better grasp of the differences between the two schemes, let's briefly consider an easier example in the following first-order linear ODE

$$\dot{y}(t) = -\alpha y(t) \tag{2.14}$$

with initial condition $y(0) = 1$. Due to its simplicity we can easily find the analytical solution, which is simply given by:

$$y(t) = e^{-\alpha t}$$

However, even if we know its true solution, we can still approximate this equation numerically using either an explicit or an implicit method just as we did with the HJB.

In both cases we can approximate the time derivative on the left-hand side of eq. (2.14) as

$$\dot{y}(t) \approx \frac{y(t+\Delta) - y(t)}{\Delta},$$

while the difference is going to be in how we handle the RHS (i.e. whether we use $y(t)$ or $y(t+\Delta)$): with an explicit scheme, we use $y(t)$ on the RHS; with an implicit scheme, we use $y(t+\Delta)$.[25]

With the explicit scheme we therefore have:

$$\frac{y(\Delta) - y(0)}{\Delta} = -\alpha y(0)$$

which rearranging and using the initial condition $y(0) = 1$ becomes:

$$y(\Delta) = (1 - \alpha\Delta) \tag{2.15}$$

that is, we are approximating $y(t)$ using a linear approximation.

On the other hand if we use the implicit method on the right hand side we have $y(\Delta)$ and we have:

$$\frac{y(\Delta) - y(0)}{\Delta} = -\alpha y(\Delta)$$

as they often include routines to solve system of equations more efficiently (this is usually done with a so-called *backslash* operator so that the system $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ can be solved directly as $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$)

[25] Finally, since we already have $y(0)$ given, it will make sense to set $t = 0$ in both the implicit and explicit schemes.

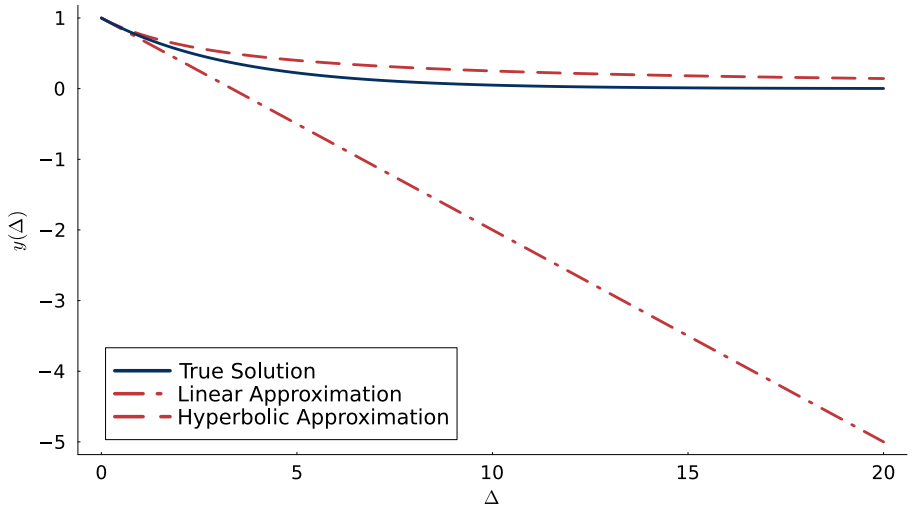

Figure 2.4: Explicit and Implicit methods' comparison

which rearranging and using the initial condition leads to:

$$y(\Delta) = \frac{1}{1 + \alpha\Delta} \tag{2.16}$$

that is, we are approximating $y(t)$ using a hyperbolic approximation.

Figure 2.4 shows how well the two approximations in eqs. (2.15) and (2.16) compare to the true solution in eq. (2.14) for different values of Δ. It should be immediately apparent that, while the linear approximation (i.e., the explicit scheme) is only good for values of Δ that are sufficiently small, the hyperbolic approximation (i.e., the implicit scheme) remains much closer to the true solution also for very large Δ.

2.2.5. Results

Now that we know how to write down the HJB, approximate it with finite-difference method using the upwind scheme, and find the solution using an implicit method we can quickly look at the results. Since the neoclassical growth model is fairly well known, rather than focusing on the economics we will simply look at the numerical properties of the solution and make sure that it indeed coincides with the solution we obtained using the shooting algorithm in chapter 1.

In terms of model specifications we assume that the household has CRRA utility with risk aversion coefficient of 2 and discounts the future at 5% rate, the production function is $F(k) = k^{\alpha}$ with $\alpha = 0.3$, and the depreciation rate δ is set to 5%. We solve the model on a grid for capital with 10,000 points uniformly distributed in $[0.001k_{ss}, 2k_{ss}]$ where k_{ss} is the steady state value of capital computed in closed form as in chapter 1.

Section B.4 includes the code that was used to generate all results in this chapter. The code is written using the `julia` programming language and its main objective is clarity rather than performance. As a consequence it is certainly not optimized for speed (and

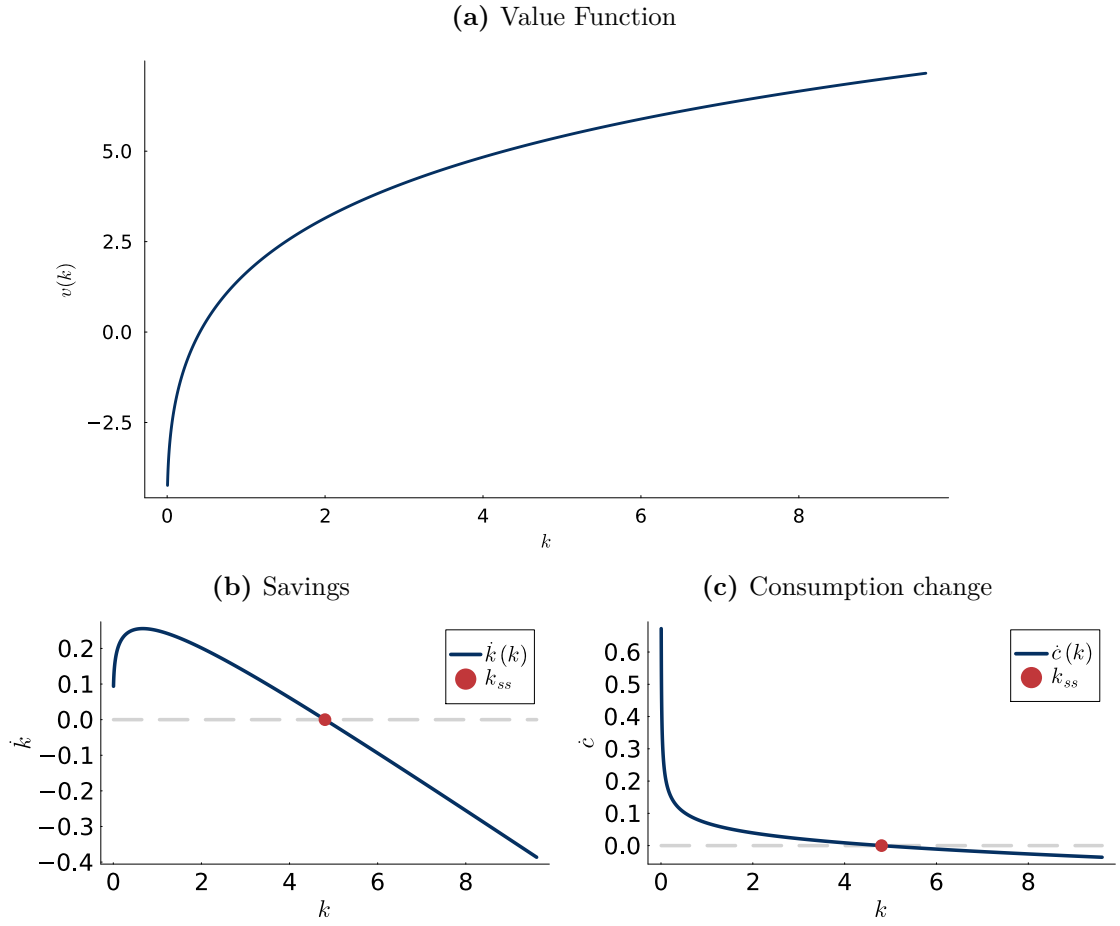

Figure 2.5: Results for the Neoclassical Growth Model

is in fact very "unjulian"). Nonetheless, after the first compilation run, solving the model with 10,000 grid points takes less than a second on a standard laptop.

Figure 2.5 plots the approximated value function in panel (a), which is the solution to eq. (2.7), as well as savings $\dot{k}$ in panel (b) and $\dot{c}$ in panel (c), which is derived from the Euler equation (2.6). The red dot is the computed value for k_{ss}. As we can see both $\dot{k}$ and $\dot{c}$ cross zero exactly at k_{ss}, which indicates that the numerical solution is equivalent to the analytical one.[26]

2.3. Some Practical Considerations

Now that we have a basic understanding of how continuous-time methods work, it is worth taking a short detour and answer the question of why we should care about continuous time in the first place. There are at least two main advantages that surfaced in what we covered so far (and more will be apparent in future lectures): First, it is generally

[26]Notice that nowhere in the HJB solution did we use the solution for $k_s s$, except in the definition of the grid (which could clearly have been done without).

(a) Matlab code

```
% Elements of the A matrix
elem_a = -min(s_B, 0)/dk;   % lower diagonal
elem_x = max(s_F, 0)/dk;    % upper diagonal
elem_b = - elem_a - elem_x; % main diagonal

% Finite-difference A matrix
A = spdiags(elem_a(2:I), -1, I, I) + spdiags(elem_b, 0, I, I) + spdiags([0; elem_x(1:I-1)], 1, I,
↪  I);
```

(b) Julia code

```
# Elements of the A matrix
elem_α = -min.(s_B, 0)/dk   # lower diagonal
elem_ξ = max.(s_F, 0)/dk    # upper diagonal
elem_β = - elem_α - elem_ξ  # main diagonal

# Finite-difference A matrix
A = spdiagm(-1 => elem_α[2:end],       # lower diagonal
             0 => elem_β[:],           # main diagonal
             1 => elem_ξ[1:(end - 1)]) # upper diagonal
```

Figure 2.6: Examples of how to build a sparse matrix

somewhat easier to get analytical and numerical results in continuous time due to the fact that first-order conditions are "static". In fact, just as in the neoclassical growth model we just solved, the FOC $u'(c) = v'(k)$ relates *current* marginal utility of consumption with the *current* marginal value of capital. On the other hand, in discrete time that same FOC would have gotten us a relation between *current* marginal utility of consumption and *future* value of capital. The intertemporal conditions we get in discrete time usually creates quite a few more problems than the intratemporal ones we get in continuous time.

The second main advantage is that, because time is continuous, variables can only move an infinitesimal amount, which means that transition matrices are extremely sparse. Such sparsity is precisely what allows the numerical solution of continuous-time models to be much faster. In fact, it is *fundamental* to leverage such sparsity in order to take advantage of the computational gains. Specifically, in most common programming languages, it is often necessary to explicitly construct matrices as sparse. By doing so, the program can then specialize the backslash operator to solve the system of equations extremely fast. Figure 2.6 provides two examples of how to explicitly construct a (tridiagonal) sparse matrix in both `Matlab` and `julia`.

As we will see in the coming lectures, continuous time also has an obvious application to heterogeneous-agents models because it naturally exploits the connection between the household problem and the problem of finding the distribution – possibly *the* object of interest of heterogeneous-agent models.

Chapter 3

Stochastic Calculus

In this lecture we will cover some basic notions of stochastic calculus, focusing mostly on what is necessary to solve heterogeneous-agent models in continuous time. The main reason we are interested in learning about stochastic calculus is that, just as we do in discrete time, we often have to deal with functions of stochastic processes (e.g. expected values). However, in continuous time we cannot apply the standard calculus rules because the Brownian motion – one of the standard building blocks of uncertainty in continuous time – is nowhere differentiable (although everywhere continuous). As a consequence, in order to deal with stochastic processes in continuous time, we will have to adapt some of the rules of standard calculus.

We will start by covering some basic preliminary notions and introducing the main stochastic processes we will use in the rest of these notes. Then, we will present possibly the most important result in stochastic calculus: Itô's lemma. We will also introduce the Kolmogorov Forward Equation (KFE) – an equation that allows us to characterize the evolution of the probability density function of stochastic processes – and infinitesimal generators, which will later allow us to derive a connection between HJB and KF equations. Finally, we will take a look at how to find a numerical solution to the KFE in order to find the stationary distribution of a specific process.

The objective of these notes is *not* to give an in-depth treatment of stochastic calculus. However, in order to apply many of the methods we are going to use, one needs to be familiar at least with its the basic notions. These notes therefore try to strike a balance of theory and applications. Students who are more interested in directly applying these methods can therefore focus more on the applications, while students who are interested in understanding how and why these methods work will at least find some preliminary notions as well as references to know more. Obviously enough, this means that these notes are in no way a substitute for a book (or a class) on stochastic calculus.

If you are interested in knowing more about stochastic calculus, some useful general references are Øksendal (2014), Karatzas and Shreve (1991) and Karatzas and Shreve (1998), Klebaner (2012), and Kloeden and Platen (1995) (which focuses more on numerical methods for the solution of stochastic differential equations). Other resources that are more economics-focused and you might find useful are Stokey (2009) and Munk (2015)

(the latter is mostly about finance, but with some basic notions of stochastic calculus as well).

3.1. Stochastic Processes

Since the objective of stochastic calculus is to work with stochastic processes, it is a good idea to first define what these are.

Definition 1 [Stochastic Process]**:** *A process that can be described by the change of some random variables over time.*

Definition 2 [Stationary Increments]**:** *For any* $0 < s, t < \infty$*, the distribution of the increments* $W_{t+s} - W_s$ *is the same as that of* $W_t - W_0 =: W_t$*.*

Definition 3 [Independent Increments]**:** *For every choice of non-negative real numbers* $0 \leq s_1 < t_1 \leq s_2 < t_2 \leq \ldots \leq s_n < t_n < \infty$*, the increment random variables* $W_{t_1} - W_{s_1}, W_{t_2} - W_{s_2}, \ldots, W_{t_n} - W_{s_n}$ *are jointly independent.*

Before defining the various types of stochastic processes, a few preliminaries:

- We will sometimes refer to continuous processes and discrete processes. However, this does not refer to the time dimension: time is *always* assumed to be continuous. Hence, when we will talk about continuous processes (as opposed to discrete ones) we mean stochastic processes that have continuous sample-paths. That is, processes in which there are no jumps. Very informally, you can think of a continuous process as one which you could in principle draw with a single stroke of a pen (provided you have sufficient ink).
- Loosely speaking, when we talk about Markov processes, we mean that the evolution of the process can only depend on the history of the process through the *current* value of the process.[1]
- Throughout this document, we will use X_t as shorthand notation for $X(t)$.

3.1.1. Continuous Processes

Brownian Motion

In many of the applications we consider in this class, uncertainty is modeled by the evolution of a so-called standard Brownian motion (or *Wiener process*), one of the basic building blocks on which we will build more complicated stochastic processes. We will usually think of a Brownian motion dW_t as the underlying source of uncertainty to the economy or the individual at time t.

[1] We will provide a more precise definition of this property for continuous-time Markov chains in section 3.1.3, while its formal definition for diffusion processes is beyond the scope of these notes and can be found in Øksendal (2014, Theorem 7.1.2.).

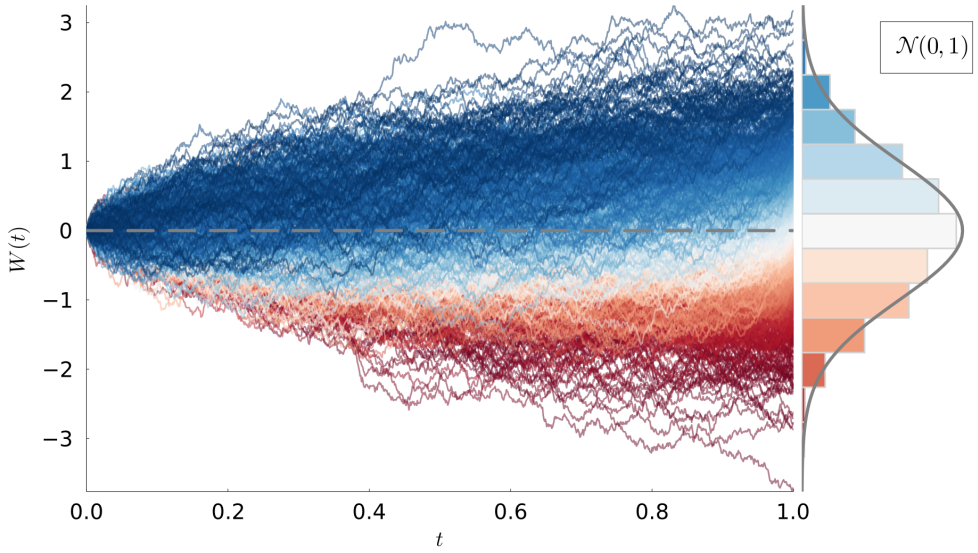

Figure 3.1: Sample paths of a standard Brownian motion process

Definition 4 [Standard Brownian Motion]**:** *A Wiener process is a stochastic process* $W \equiv \{W_t\}_{t \geq 0}$ *with the following properties:*

1. $W_0 = 0$
2. W *has continuous sample paths (the function* $t \to W_t$ *is continuous in* t*)*
3. W *has stationary, independent increments*
4. *For all* $0 \leq t_1 < t_2$*, the random variable* $W_{t_2} - W_{t_1}$ *is distributed* $\mathcal{N}(0, t_2 - t_1)$*.*

Informally, we can write $dW_t \approx W_{t+dt} - W_t \sim \mathcal{N}(0, dt)$, which allows us to map the continuous-time process with its discrete-time counterpart using the following approximation: $W(t + \Delta t) - W(t) = \varepsilon\sqrt{\Delta t}$, where $\varepsilon \sim \mathcal{N}(0, 1)$. In fact, by properties 1 and 4 it is easy to observe that $W_t \sim \mathcal{N}(0, t)$. In other words, a Wiener process is essentially the continuous-time analogue of a random walk, $W_{t+1} = W_t + \varepsilon_{t+1}$ with initial value 0, which also implies that its variance increases with time. While somewhat informal, this approximation will turn out to be particularly valuable when we will need to discretize the process.

Figure 3.1 plots the evolution of a large number of sample paths of a standard Brownian motion, which allows us to directly observe that, as we just argued: the average of the process is zero $\mathbb{E}(W(t)) = 0$, the variance increases with time $\text{Var}(W(t)) = t$, and the distribution is normal.

Properties of Brownian motions — The standard Brownian motion is a martingale, it exhibits the Markov property, it has infinite total variation (over any interval, no matter

how small) and finite quadratic variation.[2] Specifically:

$$\langle W, W\rangle_t^1(\omega) = \infty \qquad \forall\, t > 0$$
$$[W, W]_t = \langle W, W\rangle_t^2(\omega) = t \qquad \forall\, t > 0$$

Intuitively, a process with infinite variation is such that, if we want to draw the sample path over a time interval $t \in [0, \Delta]$, no matter how small Δ gets, we will always need an infinite amount of ink to draw the process.[3]

Additionally, Brownian motion paths are a continuous function of t (by definition), are not monotone (in any interval, no matter how small), and are not differentiable at any point.

Generalized Brownian Motion

By itself, the standard Brownian motion is a rather simple – and therefore not very useful – process. However, as we are about to see, it can be used as a building block for more complex stochastic processes.

For example, a standard Wiener process can be easily generalized to allow for a different drift and volatility. It is in fact often the case that a stochastic process $\{X_t\}_{t\geq 0}$ is defined in terms of a Brownian motion, an initial value X_0, and an equation of the form

$$dX_t = \mu_t dt + \sigma_t dW_t \tag{3.1}$$

where μ_t and σ_t are known at time t. For reference, eq. (3.1) is a stochastic differential equation (SDE) – a differential equation in which one or more terms (in this case dW_t) are stochastic processes.

Taking a step back, let $W \equiv \{W_t\}_{t\geq 0}$ be a standard Brownian motion, and define a new stochastic process $X \equiv \{X_t\}_{t\geq 0}$ as

$$X_t = X_0 + \mu t + \sigma W_t, \quad t \geq 0$$

where X_0, μ, and σ are constants. The change in value of the process between t_1 and t_2, is then given by:

$$X_{t_2} - X_{t_1} = \mu \cdot (t_2 - t_1) + \sigma \cdot (W_{t_2} - W_{t_1}).$$

Just as before, we can then informally think of dX_t as the increment $X_{t+dt} - X_t$ over an "instant" (of length dt) and write the change over such infinitesimally short interval $[t, t + dt]$ (for $dt \to 0$) in differential form:

$$dX_t = \mu dt + \sigma dW_t \tag{3.2}$$

Since dW_t has mean zero and variance dt, we can also (informally) compute the con-

[2] A more complete definition of quadratic variation can be found in section C.2.

[3] In fact, even if we were to zoom in on a sample path of a Brownian motion and we kept zooming on an infinitely small dt interval, the process would keep looking exactly the same.

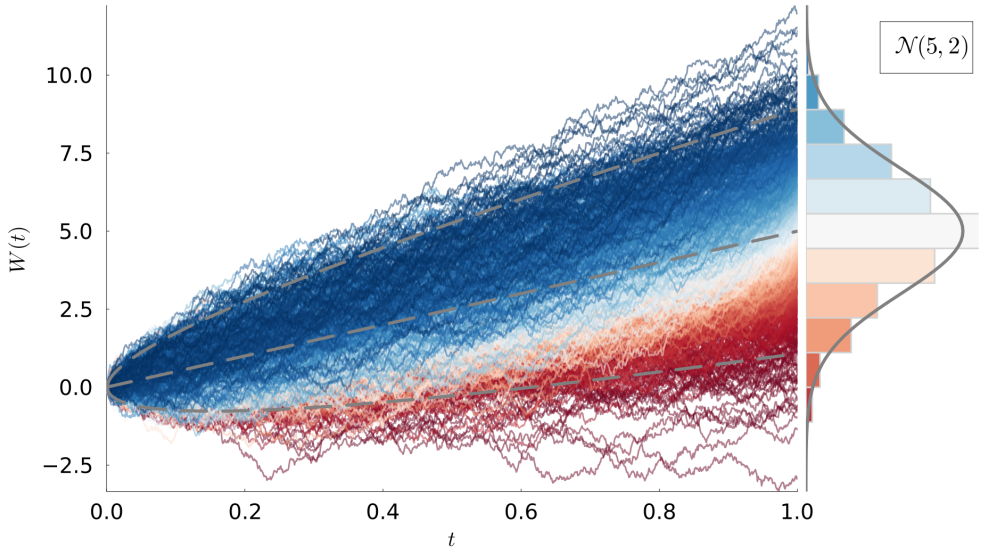

Figure 3.2: Sample paths of a generalized Brownian motion process

ditional mean and variance of dX_t as

$$\mathbb{E}_t\left[dX_t\right] = \mu dt, \qquad \mathrm{Var}_t\left[dX_t\right] = \sigma^2 dt$$

and we can interpret μ and σ^2 as the conditional mean and variance of the change in the value of the process per unit time.

The process X defined as in eq. (3.2) is called a generalized Brownian motion and it is essentially the continuous-time analogue of a random walk with drift. The parameter μ is usually referred to as the *drift* of the process, while the parameter σ reflects uncertainty about the future values of the process and is often referred as the *volatility* of the process.

Figure 3.2 plots the evolution of a large number of sample path of a generalized Brownian motion process with $\mu = 5$ and $\sigma = 2$. From this figure we can observe that the process has a clearly positive (and linear) drift, and that the distribution at time t is $\mathcal{N}(\mu t, \sigma^2 t)$ (the dashed lines above and below the drift represent the 95% confidence interval).

If the parameters μ and σ are allowed to depend on time in a deterministic way, X is called a *time-inhomogeneous* generalized Brownian motion:

$$dX_t = \mu(t)dt + \sigma(t)dW_t \tag{3.3}$$

or, more precisely:

$$X_{t_2} - X_{t_1} = \int_{t_1}^{t_2} \mu(s)ds + \int_{t_1}^{t_2} \sigma(s)dW_s \tag{3.4}$$

where the last integral is a so-called stochastic integral (i.e. an integral which is integrated with respect to a stochastic process).

Diffusion Processes

Because not all processes can be defined using standard and generalized Brownian motions (e.g. if we want to restrict our stochastic process to be non-negative), it is useful to further generalize our definition to diffusion processes:

Definition 5 [Diffusion Process]**:** *A diffusion process is a stochastic process* $X = \{X_t\}_{t\geq 0}$ *for which the drift* μ *and the volatility* σ *are allowed to depend on the current value of the process:*

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

A diffusion process is therefore a continuous-time Markov processes with continuous sample paths and, if both μ and σ are independent of time, the diffusion is said to be *time-homogeneous*.

Just as before, a more rigorous definition of a diffusion process should start by defining the change in the process over an interval $[t_1, t_2]$ as

$$X_{t_2} - X_{t_1} = \int_{t_1}^{t_2} \mu(X_s, s)ds + \int_{t_1}^{t_2} \sigma(X_s, s)dW_s. \tag{3.5}$$

Notice that, because the integrands are now functions of X_s and s, they are generally unknown at time t_1. Nonetheless, since both $\mu(\cdot)$ and $\sigma(\cdot)$ only depend on the current value of the process, a diffusion process is still a Markov process.

Ornstein-Uhlenbeck Process — By choosing appropriate values for μ and σ, we can model almost any continuous stochastic process, which demonstrates their remarkable flexibility. One classic example of a diffusion process often used in economics is the Ornstein-Uhlenbeck process (often referred to, especially in finance, as the Vašiček model). In differential form, an Ornstein-Uhlenbeck process can be expressed as:

$$dX = \eta(\bar{X} - X)dt + \sigma dW$$

where $\bar{X}$ is the long-run mean of the process, σ its instantaneous volatility, and η is the rate of mean reversion. The Ornstein-Uhlenbeck process is the continuous-time analogue of an AR(1) process with autocorrelation $e^{-\eta} \approx 1 - \eta$.[4]

Notice also that, if we ignore risk (i.e. set $\sigma = 0$), the Ornstein-Uhlenbeck process reduces to a first-order ODE:

$$\dot{X} = \eta(\bar{X} - X)$$

with solution $X(t) = \bar{X} + c_0 e^{-\eta t}$ and constant exponential decay η (where c_0 is a constant that will depend on the initial condition $X(0)$).

Figure 3.3 plots the evolution of 3 different sample paths of an Ornstein-Uhlenbeck process with $\bar{X} = 1.2$, $\eta = 1$, and $\sigma = 0.3$. The paths start from different initial points (0, 1, and 2) and they all tend towards the long-run mean of $\bar{X}$. The plot also includes

[4] In fact, it can also be shown that the stationary distribution of the process is $\mathcal{N}\left(\bar{X}, \frac{\sigma^2}{2\eta}\right)$.

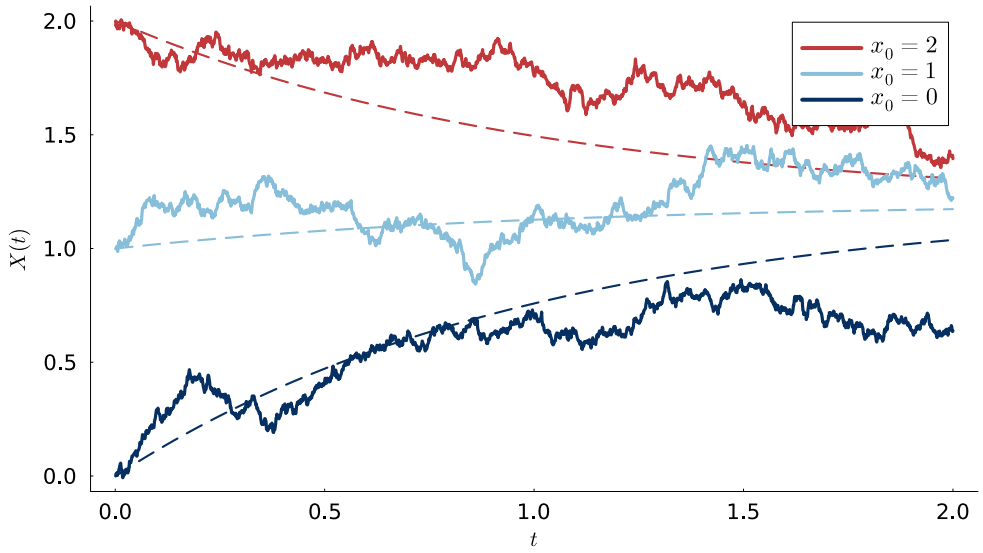

Figure 3.3: Three sample paths of an Ornstein-Uhlenbeck process

the paths of the corresponding deterministic ODEs (dashed lines) starting from the same initial points, which shows how each path on average decays towards the mean at rate η.

Itô Processes

For completeness sake, one can also define continuous processes that are even more general than diffusion processes:

Definition 6 [Itô Process]**:** *An Itô process takes the form:*

$$X_t = X_0 + \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW_s \tag{3.6}$$

where $\mu()$ and $\sigma()$ are themselves stochastic processes.

The associated SDE in differential form is given by:

$$dX_t = \mu_t dt + \sigma_t dW_t.$$

However notice that, in the case of Itô processes, the more rigorous integral representation of the process in eq. (3.6) is usually more appropriate.

It is easy to see that a diffusion process is simply a special case of an Itô process in which the drift and volatility are simply functions of t and X_t. For more general Itô processes, on the other hand, μ_t and σ_t may depend for example on past values of X or of other processes, which implies Itô processes need not be Markov.

In these lectures, we will never actually use Itô processes. However, since some version of theorems – most notably Itô's lemma – found in the wild often refer to Itô processes, it is useful to at least know what they are. In particular, since a diffusion is just a special

case of an Itô process, any such theorem that applies to the latter will also be applicable to the former.[5]

Numerical Solution

To conclude this section it should be noted that, except in few and very specific cases, SDEs have no analytical solutions and it is thus usually necessary to approximate them. Researchers have therefore developed plenty of methods to approximate these solutions.[6] The natural first example to discretize an SDE is the Euler-Maruyama method, a straightforward extension of the Euler method.

Euler-Maruyama Method — Given the following SDE:

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

discretize it along the time dimension to obtain

$$X_{t+\Delta t} - X_t = \mu(X_t, t)\Delta t + \sigma(X_t, t)\Delta W_t.$$

Recalling definition 4, for small Δt we have that $\Delta W_t \sim \mathcal{N}(0, \Delta t)$ and we can therefore approximate (and simulate) $\Delta W_t = \varepsilon\sqrt{\Delta t}$ where $\varepsilon \sim \mathcal{N}(0, 1)$. Once we have chosen a sufficiently small step-size Δt, and given an initial condition on X_0, the problem can be solved forward to get a discretized approximation to one (or more) sample path of the underlying continuous-time process. While the Euler-Maruyama method is often inefficient and does not always work well in more complex cases, its simplicity makes it an obvious candidate for simple applications.[7]

3.1.2. Discrete Processes

Until now, we have only discussed continuous processes (i.e. without jumps). However, since not all stochastic processes are continuous, it is useful to cover jump processes as well.

Definition 7 [Jump Process]**:** *A jump process is a stochastic process that has discrete movements (jumps) with random arrival times.*

Informally, a jump process is simply a stochastic process that has discontinuities happening at random times.

[5] As a side note, the fact that Itô's lemma is often referred to as a theorem is not actually wrong: in fact the reason why it is referred to as a "lemma" is simply historical. As noted in Jarrow and Protter (2004) "The book by H. P. McKean, Jr., published in 1969, had a great influence in popularizing the Itô integral, as it was the first explanation of Itô's and others' related work in book form. But McKean referred to Itô's formula as Itô's lemma, a nomenclature that has persisted in some circles to this day. Obviously this key theorem of Itô is much more important than the status the lowly nomenclature "lemma" affords it [...]"

[6] Kloeden and Platen 1995, is an excellent resource for many of the "classic" methods which covers everything you might want to know at this stage and more.

[7] In fact, all the figures shown above have been obtained using this method.

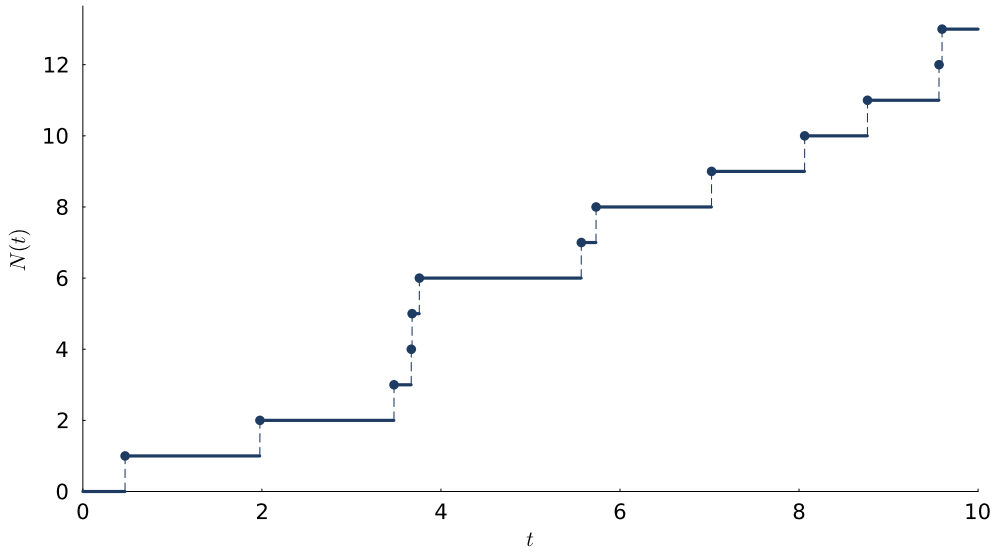

Figure 3.4: Sample path of a simple Poisson process

The most common type of jump process is the Poisson process, which is a type of counting process (i.e. an integer-valued, non-negative, non-decreasing stochastic process which "counts" the number of events that occur in a given time interval).

Poisson Processes

Definition 8 [Simple Poisson Process]**:** *A simple Poisson process (with intensity $\lambda > 0$) is a counting process $\{N_t\}_{t\geq 0}$ with the following properties:*

1. $N_0 = 0$
2. *Independent increments*
3. *Stationary Poisson increments: if $s < t$, then $N_t - N_s \sim \text{Pn}(\lambda(t-s))$:*

$$\mathbb{P}(N_t - Ns = k) = e^{-\lambda(t-s)}\frac{(\lambda(t-s))^k}{k!}, \qquad k \geq 0$$

4. *The paths of N_t are right-continuous with left limits*

In a Poisson process, the intensity λ is the expected number of jumps per unit time, that is:

$$\lambda = \lim_{h\to 0}\frac{1}{h}\mathbb{P}(N_h = 1)$$

Notice that, even though the number of jumps are distributed as a Poisson distribution (which is a discrete probability distribution), the process itself as a function of time is defined on the real line – which obviously makes it it a continuous-time stochastic process.

Figure 3.4 plots one sample path of a simple Poisson process with intensity $\lambda = 1$. We can see that the process starts at 0 and increases by a single integer amount at random

times.

The simple Poisson process is by itself rather limited in its applications. Hence, we will often use more general types of jump processes which build on it. Examples of such processes are the compensated Poisson process, which transforms the simple Poisson process into a martingale and is given by $N_t - \lambda t$; the compound Poisson process, which features random jump sizes (e.g. normally distributed); and other extensions including processes with time and/or state-dependent intensities, random intensities, etc.

Properties of Poisson processes — A Poisson process is a Markov process with finite total variation and finite quadratic variation.[8] In particular, they are both Poisson processes themselves:

$$\begin{aligned} \langle N, N\rangle_t^1(\omega) &= N_t, && \forall t > 0 \\ [N, N]_t = \langle N, N\rangle_t^2(\omega) &= N_t, && \forall t > 0 \end{aligned}$$

Additionally, since the Poisson process has finite variation, we also have that:

$$[W, N]_t = \langle W, N\rangle_t^2(\omega) = 0$$

It can be shown that the probability of observing k jumps in an interval of length h is given by:

$$\begin{aligned} \mathbb{P}(N_h = 0) &= e^{-\lambda h} = 1 - \lambda h + o(h) \\ \mathbb{P}(N_h = 1) &= \lambda h e^{-\lambda h} = \lambda h + o(h) \\ \mathbb{P}(N_h = 2) &= h^2 \frac{\lambda^2}{2} = o(h) \end{aligned}$$

where $o(h) \to 0$ as $h \to 0$.[9] Therefore, for small h, $N_{t+h} - N_t$ behaves like a Bernoulli random variable with parameter λh. That is, a jump happens with probability λh and does not happen with probability $1 - \lambda h$.

The time between jumps is exponentially distributed with parameter λ (i.e. it has mean $1/\lambda$).

3.1.3. Continuous-time Markov Chains

Before moving on to stochastic calculus, it is worth it to also discuss continuous-time Markov chains (CTMC) both because these will be often used in our models and because students who have worked in discrete time will already be familiar with discrete-time

[8] A process has finite variation if its paths (realizations of the process over time) have bounded total variation over any finite interval. The paths of a Poisson process are piecewise constant with jumps of size 1 at each event. Since these jumps are countable and each jump is of fixed size, the total variation is simply the number of jumps in the interval, which is finite for any finite interval.

[9] $o(h)$ is the so-called "little-o notation" and it represents a term that goes to zero faster than h. The first property then simply restates the well know approximation $e^{\lambda h} \approx 1 + \lambda h$ for small h (so we are simply rewriting this approximation in a more formal way). The second and third properties come from the fact that terms in $(\lambda h)^2$ go to zero faster than h (and therefore go into $o(h)$).

Markov chains and with the associated transition matrices. Additionally, since there is a very tight link between transition matrices and infinitesimal generators, treating CTMC separately from other discrete processes will help us gain some intuition about what infinitesimal generators are and how they work in a more familiar context.

In discrete time, a Markov chain remains in any given state for exactly one unit of time before making a transition (possibly to the same state). In continuous time, this assumption is instead relaxed and the chain can remain in any given state for an unspecified amount of time, called *holding time*. While at first sight this might seem to be an issue for the Markov property we are just about to see that, as long as the holding times are exponentially distributed, such property is in fact retained.

In order to define a CTMC, let's first give an appropriate definition of the Markov property in continuous time. Specifically, for a process taking values in a discrete set of states the Markov property requires that:

$$\mathbb{P}\left(X_{t_0+t} = j | X_{t_0} = i, X_{t'} = i'\right) = \mathbb{P}\left(X_{t_0+t} = j | X_{t_0} = i\right) = P_{i,j}(t) \quad \forall\, 0 \leq t' < t_0. \tag{3.7}$$

That is to say: the probability that the process at a time $t_0 + t$ is in state j is only a function of where the process is at time t_0 and of the time interval considered, and does not therefore depend on where the process was at any time $t' < t_0$. Importantly, note that this probability $P_{i,j}(t)$ is (usually) a function of time.

At this point, we are ready to define a continuous-time Markov chain:

Definition 9 [Continuous-time Markov chain]**:** *A stochastic process X taking values in a discrete set of states $i \in \mathcal{X}$ is called a continuous-time Markov chain if for all $t_0 \geq 0, t \geq 0, i \in \mathcal{X}, j \in \mathcal{X}$, $X(t)$ satisfies the Markov property.*[10]

That is, "a continuous-time Markov chains is a stochastic process having the Markovian property that the conditional distribution of the future $X(t+s)$ given the present $X(s)$ and the past $X(u), 0 \leq u < s$, depends only on the present and is independent of the past." (Ross 2014, p. 372) Informally, a CTMC is nothing else than a continuous-time Markov process taking values in a discrete set of states.

To see why this definition must imply that the holding times are exponentially distributed, suppose that a CTMC enters some state i at time 0 and suppose that the process does not leave state i by time s; by definition, the Markovian property requires that the probability that the process will remain in state i for some additional time t must not depend on how much time it has been in state i before (i.e. it must not depend on s). As a consequence the holding time, which is itself a random variable, is *memoryless* and must therefore be exponentially distributed.[11]

Concluding, "a continuous-time Markov chains is a stochastic process that moves from state to state in accordance with a (discrete-time) Markov chain, but it is such that the

[10] We actually also require $X(t)$ to be right-continuous. However, for most practical purposes, this assumption is often not necessary.

[11] A well-known result in probability theory is that a random variable is memoryless if and only if it is exponentially distributed (See Ross 2014, sec. 5.2.2).

amount of time it spends in each state, before proceeding to the next state, is exponentially distributed. In addition, the amount of time the process spends in state i, and the next state visisted, must be independent random variables" (Ross 2014, p. 373).[12]

Properties of CTMC — With our definition on hand, we can also define the transition probability matrix within time t as: $\mathbf{P}(t) = [P_{i,j}(t)]$, i.e. the collection of the transition probabilities from all states $i \in \mathcal{X}$ to all states $j \in \mathcal{X}$ in an interval of length t.

The fact that the matrix $\mathbf{P}(t)$ is parameterized by time means that we can also apply notions from calculus (because time is continuous). For example, we have that:

$$\mathbf{P}(0) \equiv \lim_{t \downarrow 0} \mathbf{P}(t) = \mathbf{I}.$$

To understand what this limit implies think about what the $\mathbf{P}(t)$ matrix is: it represents the matrix of transition probabilities within time t, so if we set $t = 0$, it implies no time at all has passed and the process will have experienced no transitions. The probabilities on the main diagonal of $\mathbf{P}$, $P_{i,i}$ will therefore all equal 1, and all other elements will equal 0. Hence $\mathbf{P}(0)$ will equal the identity matrix.

In the same vein as we defined $\mathbf{P}(t)$ we can also define the state distribution as: $\boldsymbol{\pi}(t) = [\pi_i(t)]' = [\mathbb{P}(X(t) = i)]'$, which is nothing else than the (row) vector of probabilities that the process $X(t)$ is in state i at time t. It immediately follows that:

$$\pi_j(t + \tau) = \sum_i \pi_i(t) P_{i,j}(\tau) \tag{3.8}$$

$$\boldsymbol{\pi}(t + \tau) = \boldsymbol{\pi}(t)\mathbf{P}(\tau). \tag{3.9}$$

Hence, the state distribution at time $t + \tau$ is equal to the distribution at time t times the transition probability matrix within time τ.

A CTMC is therefore defined by two components: a so-called jump chain (or the *embedded Markov chain*) that gives us the transition probabilities $P_{i,j}$; and the holding-time parameters λ_i that control the amount of time spent in each state.

Finally, from eq. (3.9) one can also easily prove the so-called semigroup property (which is the natural analogue of the Chapman-Kolmogorov equations for discrete-time chains):

$$\mathbf{P}(t + s) = \mathbf{P}(t)\mathbf{P}(s) = \mathbf{P}(s)\mathbf{P}(t) \quad \forall\, t, s \geq 0. \tag{3.10}$$

3.2. Stochastic Calculus

One of the main objectives of stochastic calculus is to be able to take derivatives of stochastic objects. However, because the Brownian motion is nowhere differentiable, our usual methods of differentiation does not work anymore. Our "new" method of differentiation will essentially boil down to being a combination of standard calculus plus some simple

[12] Notice that, by our definition of a CTMC, it immediately follows that Poisson processes are CTMC on the state space $\mathbb{Z}_+$ that always move from state i to state $i + 1$ (i.e. $P_{i,i+1} = 1$).

	dt	dW_t	dN_t
dt	0	0	0
dW_t	0	dt	0
dN_t	0	0	dN_t

Table 3.1: Algebraic rules for stochastic calculus

rules. We are now going to (informally) derive some of these rules; specifically, those regarding Brownian motions. However, note that such derivations cannot be directly applied to, for example, Poisson processes. Table 3.1 collects the main rules we care about.

3.2.1. Differential Algebra

In order to prove Itô's lemma in section 3.2.2 we will make use of three rules, which are:

$$(dt)^2 = 0, \qquad dt \cdot dW = 0, \qquad (dW)^2 = dt.$$

To see where these rules are coming from, note the following:

- $(dt)^2 = 0$ – to see why this is the case, recall that, in continuous time, dt is an infinitesimal time interval $dt \to 0$. As a consequence $(dt)^2$ is going to be far smaller than dt itself (essentially 0) and can therefore be safely ignored.

- $dt \cdot dW = 0$ – given the definition of a Wiener process, we know that $dW \sim \mathcal{N}(0, dt)$ so in expectations dW is zero; therefore we also have that $\mathbb{E}[dt \cdot dW] = dt \cdot \mathbb{E}[dW] = 0$. However, $dt \cdot dW$ being zero in expectations is not sufficient to say that $dt \cdot dW = 0$. In order to do so, we need to show that it is *always* zero, which we can do by showing that its variance is also zero. In fact, we have that $\text{Var}[dt \cdot dW] = (dt)^2\text{Var}[dW] = (dt)^3$ which is again much smaller than dt itself and can therefore be safely ignored.

- $(dW)^2 = dt$ – we use the same approach as before: we know that $\mathbb{E}[(dW)^2] = \text{Var}[dW] - \mathbb{E}[dW]^2 = dt$ so we know that $(dW)^2$ is on average equal to dt. However, we need to show that it is always equal to dt. In order to do so, we can again show that its variance is zero:

$$\text{Var}\left[(dW)^2\right] = \underbrace{\mathbb{E}\left[(dW)^4\right]}_{3(dt)^2} - \underbrace{\mathbb{E}\left[(dW)^2\right]^2}_{(dt)^2} = 2(dt)^2$$

  which means that, again, as $dt \to 0$, its variance goes to zero and $(dW)^2$ can be approximated by its mean dt.

With these rules on hand, we are now ready to prove one of the main results of stochastic calculus: Itô's lemma.

3.2.2. Itô's Lemma

Itô's Lemma is an indispensable tool for working with random processes in continuous time. In fact, we often need to describe the behavior of functions of diffusion processes and, in order to describe how they vary over time, we need to resort to the rules of stochastic calculus. Ito's lemma is one of the fundamental results of stochastic calculus and is essentially the analog of the chain rule of standard calculus for stochastic processes.[13]

We start by giving the statement in its integral form which is more general, and then in its differential form which, albeit less precise, is usually enough for most purposes. Then, we will look at an informal proof and discuss some intuition. Finally (and mostly for completeness) we will see its multivariate version in differential form.

Theorem 3 [Itô's lemma, integral form]**:** *Suppose that $f(x)$ is a twice continuously differentiable function, and X_t is an Itô process with*

$$X_t = X_0 + \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s.$$

Then,

$$\begin{aligned} f(X_t) &= f(X_0) + \int_0^t f'(X_s)dX_s + \frac{1}{2}\int_0^t f''(X_s)d[X,X]_s \\ &= f(X_0) + \int_0^t f'(X_s)dX_s + \frac{1}{2}\int_0^t f''(X_s)\sigma_s^2 ds \\ &= f(X_0) + \int_0^t \left(f'(X_s)\mu_s + \frac{1}{2}f''(X_s)\sigma_s^2\right)ds + \int_0^t f'(X_s)\sigma_s dW_s \end{aligned}$$

Theorem 4 [Itô's lemma, differential form]**:** *Suppose that X_t is an Itô process with*

$$dX_t = \mu_t dt + \sigma_t dW_t$$

and $f : \mathbb{R} \to \mathbb{R}$ is a twice continuously differentiable. Then, the process Y defined by $Y_t = f(X_t)$ is itself an Itô process with

$$dY_t = \left(f'(X_t)\mu_t + \frac{1}{2}f''(X_t)\sigma_t^2\right)dt + f'(X_t)\sigma_t dW_t.$$

Proof

The proof of Itô's lemma we are going to present here assumes μ_t and σ_t are constant, it uses a second-order Taylor expansion of f and is a version of those often found in introductory textbooks on stochastic calculus. A more rigorous proof can be found in Øksendal 2014.

[13] One of the obvious implications of Itô's lemma is that any (twice differentiable) function of a diffusion process is itself going to be a diffusion process.

Let's start by applying a second-order Taylor expansion of f:

$$dY = \frac{\partial f(X)}{\partial X} dX + \frac{1}{2} \frac{\partial^2 f(X)}{\partial X^2} dX^2 \tag{3.11}$$

The second term, dX^2 is then:

$$dX^2 = (\mu dt + \sigma dW)^2 = \mu^2 dt^2 + 2\mu\sigma dt dW + \sigma^2 dW^2$$

Now, recalling the rules derived earlier: $dt^2 = 0$, $dt \cdot dW = 0$, and $dW^2 = dt$, we have:

$$dX^2 = \sigma^2 dt$$

and, substituting dX and dX^2 in eq. (3.11), we get:

$$\begin{aligned} dY &= \frac{\partial f(X)}{\partial X}(\mu dt + \sigma dW) + \frac{1}{2} \frac{\partial^2 f(X)}{\partial X^2} \sigma^2 dt \\ &= \left(f'(X)\mu + \frac{1}{2} f''(X)\sigma^2 \right) dt + f'(X)\sigma dW. \end{aligned}$$

Intuition

When applying Itô's Lemma, we encounter an unintuitive feature: a second-order term in the drift coefficient.[14] This second-order term is generally absent in ordinary calculus, and is key to understanding the nature of stochastic processes.

Essentially, the derivation of Itô's lemma revealed us that, in the presence of a Brownian motion, the drift of a function of a stochastic process is affected not just by the determinist drift term (which is analogous to what we'd see in the deterministic calculus) but also by the stochastic diffusion term. The latter is represented by the second-order term that emerges in Itô's calculus, and is absent in ordinary calculus. That is, it is not sufficient to write down

$$dY = \frac{\partial f(X)}{\partial X} dX$$

as we would have done in standard calculus because the second order term dX^2 is non-negligible: when dealing with Brownian motion processes, both the first and second derivatives matter (which is a direct consequence of the fact that Wiener processes have finite quadratic variation). The second-order term in the Itô's lemma is therefore a direct reflection of the stochastic nature of the process we are dealing with.

In order to develop some intuition for this, it might be useful to turn to Jensen's inequality. Recall that Jensen's inequality states that for a convex function, the expectation of the function of a random variable is greater than or equal to the function of the expectation of the random variable.

In the context of Itô's Lemma, this relationship is critical. In fact, the second-order term in Itô's Lemma that enters the drift is a direct consequence of Jensen's inequality acting on the quadratic variation of the stochastic process. Because a stochastic process

[14] We will sometimes refer to such term as Itô- (or variance-) correction term.

has non-zero variance (i.e., it's spread out), the function of the expectation is affected by the shape of the function itself – which is given by its second derivative.

To put it simply, the extra term in Itô's lemma arises because of Jensen's inequality and the stochastic nature of the process: since the variance of a stochastic process is non-zero, the expected value of a function of that process will not simply be the function of the expected value of the process – instead, it will be affected by the "spread" of the process. The second derivative, which measures the curvature of the function, tells us how this spread affects the expected value.

Geometric Brownian motion — Consider, the following example: let X be a so-called geometric Brownian motion (i.e. $dX = \mu X dt + \sigma X dW$) and suppose we want to know what is the evolution of the logarithm of a Brownian motion: $Y(X) = \log(X)$. In order to find that out, we simply need to apply Itô's lemma: the first and second derivative of $Y(X) = \log(X)$ are $Y'(X) = 1/X$ and $Y''(X) = -1/X^2$, respectively. Therefore we obtain:

$$\begin{aligned} dY &= \left((\mu X)\frac{1}{X} - \frac{1}{2}(\sigma X)^2 \frac{1}{X^2} \right) dt + (\sigma X)\frac{1}{X} dW \\ &= \left(\mu - \frac{\sigma^2}{2} \right) dt + \sigma dW \end{aligned}$$

which is a standard Brownian motion with drift and diffusion coefficients given by $\mu - \sigma^2/2$ and σ, respectively. As a consequence Y does will be distributed as:

$$Y(t) \sim \mathcal{N}\left(\left(\mu - \frac{\sigma^2}{2} \right) t, \sigma^2 t \right)$$

that is, the logarithm of a geometric Brownian motion is a Brownian motion.

Multivariate Itô's Lemma

For completeness' sake, we hereby report the multivariate version of Itô's lemma

Lemma 1 [Itô's lemma, multivariate]**:** *Let* $\mathbf{x} \in \mathbb{R}^n$, $\boldsymbol{\mu} : \mathbb{R}^n \to \mathbb{R}^n$ *and* $\boldsymbol{\sigma} : \mathbb{R}^{n \times m} \to \mathbb{R}^n$. *Let* $f : \mathbb{R}^n \to \mathbb{R}$, *with gradient* $\nabla_{\mathbf{x}} f$ *and Hessian* $\nabla^2_{\mathbf{x}} f$ *, then*

$$df(\mathbf{x}) = \left((\nabla_{\mathbf{x}} f)^T \boldsymbol{\mu} + \frac{1}{2} \mathrm{tr}\left(\boldsymbol{\sigma}^T \left(\nabla^2_{\mathbf{x}} f \right) \boldsymbol{\sigma} \right) \right) dt + (\nabla_{\mathbf{x}} f)^T \boldsymbol{\sigma} d\mathbf{W}_t$$

where tr() *is the trace and* $\mathbf{W}_t$ *is an* m*-dimensional Brownian motion.*[15]

[15] The trace of a matrix $\mathbf{A}$, denoted tr($\mathbf{A}$), is defined to be the sum of elements on the main diagonal of $\mathbf{A}$. Notice that the trace is only defined for square matrices.

3.3. The Kolmogorov Forward Equation

Now that we have introduced stochastic processes, and have (briefly) seen how to generate specific sample paths from them, it is only natural to want to look at how a stochastic process is distributed. That is, if we could generate an infinite number of sample paths from a specific stochastic process, how would the *distribution* of these sample paths look like? and additionally, starting from any initial distribution (e.g. a point mass) how would the distribution evolve over time? All of these questions, and more, can be answered by looking at the so-called Kolmogorov Forward equation (KFE), also known as the Fokker-Plack equation. The KFE is nothing else than an equation characterizing the evolution of the probability density function of a stochastic process. As we will see, this equation will be a key tool when solving continuous-time heterogeneous-agent models.

Proposition 1 [Kolmogorov Forward Equation]**:** *Let X_t be an Itô process described by:*

$$dX = \mu(X,t)dt + \sigma(X,t)dW.$$

Then, $g(x,t)$ – the distribution of X at time t – satisfies the following Kolmogorov Forward equation:[16]

$$\frac{\partial g(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[\mu(x,t)g(x,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[\sigma^2(x,t)g(x,t)\right]. \tag{3.12}$$

The KFE is a partial differential equation (PDE) – a differential equation involving a function of two or more variables and its partial derivatives with respect to those variables (i.e. a differential equation that contain partial derivatives) – which defines how the "cross-sectional" distribution of the process, g, evolves over time.[17]

A consequence of eq. (3.12) is that if a stationary distribution of the process, $g(x) \equiv \lim_{t\to\infty} g(x,t)$, exists it solves:

$$0 = -\frac{\partial}{\partial x}\left[\mu(x)g(x)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[\sigma^2(x)g(x)\right]$$

where we have simply used the fact that, because the distribution is not changing over time, $\frac{\partial g(x,t)}{\partial t} = 0$. Like for SDEs, PDEs do not generally have closed-form solutions except in very specific cases and so they need to be solved numerically. Since they involve partial derivatives, in order to numerically solve PDEs, we will have to find a way to approximate such derivatives, just as we did when solving the HJB of the neoclassical growth model (which was also, incidentally, a PDE).

As an example, in fig. 3.5, we use the KFE to look at how the distribution of an

[16] A derivation of the KFE for univariate diffusion processes can be found in section C.1.

[17] The KFE is not to be confused with the Kolmogorov Backward Equation, which is a different, although linked, PDE: the KFE allows us to know what the probability distribution of the state will be at time $s > t$ given an initial condition at time t (the KFE is therefore integrated *forward* in time). On the other hand, the backward equation describes what is the probability at time t of ending up in a "target set" at time $s > t$, and therefore uses a final condition on the PDE (which is then integrated *backward* in time from s to t).

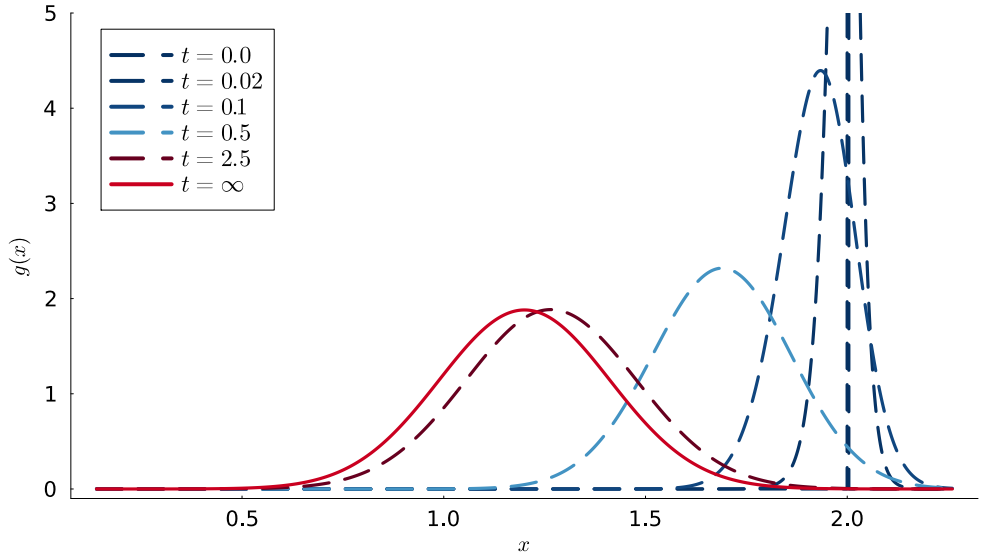

Figure 3.5: Evolution of the distribution for an Ornstein-Uhlenbeck process.

Ornstein-Uhlenbeck process evolves over time. Specifically, we use the same process as in fig. 3.3 – namely we set $\bar{X} = 1.2$, $\eta = 1$, and $\sigma = 0.3$ – and simulate its evolution over time. As initial condition we assume that the process starts from a single mass point at $x_0 = 2$, that is we use the Dirac delta function centered at 2, $\delta_2(x)$. The figure then shows, how the distribution evolves over time (in darker shades of blue as time progresses) until it eventually reaches the stationary distribution – which we know is $\mathcal{N}(\bar{X}, \frac{\sigma^2}{2\eta})$.

Multivariate Case — The KFE (3.12) can easily be generalized to the multivariate case, which we report here for completeness:

Proposition 2 [KFE, multivariate]**:** *Let $\mathbf{x}_t \in \mathbf{R}^n$ be an Itô process described by:*

$$d\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)dt + \boldsymbol{\sigma}(\mathbf{x}_t, t)d\mathbf{W}_t$$

where $\boldsymbol{\mu}$ is an n-dimensional vector, $\boldsymbol{\sigma}$ is an $n \times m$ dimensional matrix and $\mathbf{W}_t$ is an m-dimensional vector of Brownian motions. The KFE for the joint distribution $g(\mathbf{x}, t)$ is:

$$\frac{\partial g(\mathbf{x}, t)}{\partial t} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i} \left[\mu_i(\mathbf{x}, t) g(\mathbf{x}, t)\right] + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j} \left[(\boldsymbol{\sigma}\boldsymbol{\sigma}')_{i,j} g(\mathbf{x}, t)\right] \tag{3.13}$$

3.4. Infinitesimal Generators

As we will see more in details in the next two lectures, infinitesimal generators are particularly useful when mapping stochastic differential equations to their numerical approximations. However, before defining infinitesimal generator for generic continuous-time

processes, it is useful to look at how they are defined (and how they work) for continuous-time Markov chains as they are tightly linked to the more familiar transition matrices.

3.4.1. Generator of a CTMC

Recall we defined the transition probability matrix of a CTMC within time t as the collection of all transition probabilities from states i to states j: $\mathbf{P}(t) = [P_{i,j}(t)]$.

The infinitesimal generator of a CTMC represents the rate of change of these transition probabilities over time and is itself a matrix. Essentially, if we define $\mathbf{Q}$ as the infinitesimal generator of a CTMC, its elements $q_{i,j}$ are nothing else than the time derivative $P'_{i,j}(t)$ evaluated at $t = 0$. More formally, except for pathological cases, the following limit exists and defines the infinitesimal generator $\mathbf{Q}$ of a CTMC:[18]

$$\mathbf{Q} = \mathbf{P}'(0) \equiv \lim_{h \downarrow 0} \frac{\mathbf{P}(h) - \mathbf{I}}{h}. \tag{3.14}$$

Since in the coming lectures when dealing with both CTMC and other processes we will usually work with infinitesimal generators (as opposed to transition probability matrices), and many of the properties of CTMC generators translate one to one to more complex processes it is worth taking a closer look at what they are and how they work.

Properties of CTMC generators — First, notice that since all rows of the transition probability matrix $\mathbf{P}(h)$ must add up to one, it must be the case that the rows of $\mathbf{Q}$ sum to zero. In order to see why this is the case, we can write:

$$\begin{aligned}
\sum_{j \neq i} P_{i,j}(h) &= 1 - P_{i,i}(h) \\
\sum_{j \neq i} \frac{P_{i,j}(h)}{h} &= \frac{1 - P_{i,i}(h)}{h} \\
\lim_{h \downarrow 0} \sum_{j \neq i} \frac{P_{i,j}(h)}{h} &= -\lim_{h \downarrow 0} \frac{P_{i,i}(h) - 1}{h} \\
\sum_{j \neq i} q_{i,j} &= -q_{i,i}
\end{aligned}$$

where $q_{i,j}$ are simply the elements of $\mathbf{Q} = [q_{i,j}]$.

Second, since the rows of $\mathbf{Q}$ must sum up to zero, it must be the case that in any row i some entries are positive and some are negative. Specifically, it turns out to be the case that all off-diagonal entries are positive and all on-diagonal entries are negative. To see why notice that, as we already said, the elements $q_{i,j}$ represent the rate of change in the transition probability from i to j. Hence, since as time progresses the probability of moving from state i to state j goes up, by construction it must be the case that $q_{i,j} \geq 0 \; \forall \; i \neq j$; similarly, since time the probability of staying in state i goes down over time, it must be the case that $q_{i,i} \leq 0$. Informally, we can think of $q_{i,i}$ as the "outflow" from state i and

[18] You should think of the $\mathbf{I}$ on the right-hand side of eq. (3.14) as $\mathbf{P}(0)$.

of $q_{i,j}$ as the "inflow" from state i to state j and that these flows have to net to zero: all flows going out of state i must go to some other state j.

What if we wanted to compute the rate of change in the transition probabilities at any other $t \neq 0$? From the Chapman-Kolmogorov eq. (3.10) we have that

$$\begin{aligned}\mathbf{P}(t+h) - \mathbf{P}(t) &= \mathbf{P}(t)\mathbf{P}(h) - \mathbf{P}(t) = \mathbf{P}(t)\left[\mathbf{P}(h) - \mathbf{I}\right] \\ &= \mathbf{P}(h)\mathbf{P}(t) - \mathbf{P}(t) = \left[\mathbf{P}(h) - \mathbf{I}\right]\mathbf{P}(t).\end{aligned}$$

Dividing both sides by h and taking the limit as $h \downarrow 0$ we find:

$$\frac{d}{dt}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{Q} \tag{3.15}$$

$$= \mathbf{Q}\mathbf{P}(t). \tag{3.16}$$

which are nothing else than the Kolmogorov forward and backward equations, respectively.

When coupled with the initial condition $\mathbf{P}(0) = \mathbf{I}$, the matrix differential eqs. (3.15) and (3.16) have solution:

$$\mathbf{P}(t) = e^{\mathbf{Q}t} \tag{3.17}$$

which, if we forget for one second that we are dealing with matrices, closely resembles the solution to a first-order ODE of the form $\dot{p}(t) = p(t)q$. In fact, it turns out that once we have the generator $\mathbf{Q}$, we can directly solve eq. (3.17) to obtain the transition matrix $\mathbf{P}(t)$ for any duration t (and viceversa). However, because we are dealing with matrices, one has to be careful to apply matrix exponentiation, as opposed to element-wise exponentiation, since the two are *not* the same.[19]

In order to gain some further intuition about these transition rates, we can also look at how they relate to holding times – the amount of time the chain stays in a given state. Defining as T_i the holding time in state i, we saw earlier that T_i is a random variable that, in order for the process to be Markovian, must be exponentially distributed. Let λ_i be the rate parameter of its exponential distribution, i.e.

$$\mathbb{P}\left(T_i > t\right) = e^{-\lambda_i t},\ t \geq 0.$$

By the Markov property we have that $\mathbb{P}\left(T_i > t + \tau | T_i > \tau\right) = P_{i,i}(t)$. If we let $\tau = 0$, we have $\mathbb{P}\left(T_i > t\right) = e^{-\lambda_i t} = P_{i,i}(t)$. Differentiating both sides and evaluating at $t = 0$, we get:

$$\left.\frac{d}{dt}e^{-\lambda_i t}\right|_{t=0} = \left.\frac{d}{dt}P_{i,i}(t)\right|_{t=0}$$

$$-\lambda_i e^{-\lambda_i \cdot 0} = P'_{i,i}(0)$$

$$\lambda_i = -q_{i,i}$$

[19] There are several numerical methods to compute matrix exponentiation but most numerical software usually include at least one such method (as long as you call the right function).

where from the second to third line we have simply used the definition of $\mathbf{Q}$. Hence, $T_i \sim \text{Exp}(-q_{i,i})$ and the mean holding time in state i is simply $\mathbb{E}[T_i] = -\frac{1}{q_{i,i}}$ (recall that $q_{i,i} \leq 0$). In other words, the elements on the main diagonal of the infinitesimal generator of the CTMC are nothing else than (minus) the rate parameter of the holding times and consequently also the (inverse) average holding times.

Finally, it is useful to look at how the generator of a process relates to the state distribution $\boldsymbol{\pi}(t)$. Recall the definition of the state distribution $\boldsymbol{\pi}(t)$ in eq. (3.8), which we can write as:

$$\pi_j(t+h) = \sum_i \pi_i(t) P_{i,j}(h)$$
$$\frac{\pi_j(t+h) - \pi_j(t)}{h} = \pi_j(t)\frac{P_{j,j}(h) - 1}{h} + \sum_{i \neq j} \pi_i(t)\frac{P_{i,j}(h)}{h}$$

where in the second line we have simply subtracted $\pi_j(t)$ and divided by h on both sides. Taking the limit as $h \downarrow 0$ we have:

$$\frac{d}{dt}\pi_j(t) = \sum_i \pi_i(t) q_{i,j} \tag{3.18}$$
$$\frac{d}{dt}\boldsymbol{\pi}(t) = \boldsymbol{\pi}(t)\mathbf{Q}. \tag{3.19}$$

Equation (3.19) can therefore be used to get the state distribution at any time t from any initial distribution $\boldsymbol{\pi}(0)$ simply as $\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)e^{\mathbf{Q}t} = \boldsymbol{\pi}(0)\mathbf{P}(t)$.

Obviously enough, if the steady-state distribution $\boldsymbol{\pi} = \lim_{t\to\infty} \boldsymbol{\pi}(t)$ exists, it must be the case that $\frac{d}{dt}\boldsymbol{\pi} = 0$ and therefore, from eq. (3.19), the steady state distribution must also satisfy the following KFE:[20]

$$\boldsymbol{\pi}\mathbf{Q} = 0$$

which can be rewritten less compactly as

$$0 = \sum_i \pi_i q_{i,j} = \sum_{i \neq j} \pi_i q_{i,j} + \pi_j q_{j,j} = \sum_{i \neq j} \pi_i q_{i,j} + \pi_j \left(-\sum_{i \neq j} q_{i,j} \right).$$

This last way of formulating the KFE should also hopefully clarify the interpretation of the $q_{j,j}$ as outflows from state j and of the $q_{i,j}$ as inflows from state i to state j.

[20] In fact one can also show that, up to a normalization, the steady-state distribution $\boldsymbol{\pi}$ equals the eigenvector associated to the first eigenvalue of $\mathbf{Q}$ (which is 0, by construction).

3.4.2. Generators of General Processes

Definition 10 [Infinitesimal Generator]**:** *The infinitesimal generator, $\mathcal{A}$, of a Markov process is a linear operator that captures the evolution of the process over time.*[21]

$$\mathcal{A}f(x) = \lim_{t \downarrow 0} \frac{\mathbb{E}\left[f(X_t)|X_0 = x\right] - f(x)}{t} \tag{3.20}$$

The basic idea behind the infinitesimal generator is as follows: knowing that the process X_t is at x, how does the evolution of the process affect the value of $f(X_t)$ in an infinitesimal amount of time? In that sense, we can see that the definition of the infinitesimal generator is perfectly consistent (and is in fact the same) with the generator we derived for CTMC.

In section 3.4.1 we discussed the connection between infinitesimal generators for CTMC and transition matrices; the same connection applies here for more general processes. So, if you find definition 10 somewhat confusing, you can keep thinking about the generator as encoding information about the transition *rate* of the process.

As an example, if we restrict our attention to time-homogeneous diffusion processes

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW$$

and considering a generic function $f \in C^2$ (twice continuously differentiable), it is a simple application of Itô's lemma to the expected value in eq. (3.20) to show that the infinitesimal generator of the above diffusion is given by:

$$\left(\mathcal{A}f\right)(x) = \mu(x)f'(x) + \frac{1}{2}\sigma^2(x)f''(x). \tag{3.21}$$

As we will see in the next lecture there is a very tight link between infinitesimal generators, HJB and KF equations. However, before doing so, it is useful to also define the Hermitian adjoint (or simply adjoint) of an operator.

Adjoint Operators

Definition 11 [Hermitian Adjoint]**:** *The Hermitian adjoint of an operator $\mathcal{A}$ is the operator $\mathcal{A}^*$ such that:*

$$\langle \mathcal{A}x, y\rangle = \langle x, \mathcal{A}^* y\rangle \tag{3.22}$$

where $\langle \cdot, \cdot \rangle$ is the inner product on vector spaces: e.g. $\langle u, h\rangle = \int u(x)h(x)dx$.[22]

That is, $\mathcal{A}^*$ must be such that:

$$\langle \mathcal{A}f, g\rangle \equiv \int_{-\infty}^{\infty} \mathcal{A}f(x)g(x)dx = \int_{-\infty}^{\infty} f(x)\mathcal{A}^* g(x)dx \equiv \langle f, \mathcal{A}^* g\rangle.$$

In fact, given an infinitesimal generator $\mathcal{A}$ one can easily (if somewhat tediously) find the

[21] The fact that the infinitesimal generator is a *linear* operator will turn out to be very useful when solving these models numerically.

[22] The inner product is simply a generalization of the dot product to infinite-dimensional spaces.

adjoint $\mathcal{A}^*$ by simply using integration by parts on the above equation.

For the generator of a time-homogeneous diffusion process as in eq. (3.21), it can be shown that its adjoint $\mathcal{A}^*$ is given by:

$$\left(\mathcal{A}^* g\right)(x) = -\frac{\partial}{\partial x}\left[\mu(x)g(x)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[\sigma^2(x)g(x)\right] \tag{3.23}$$

which the careful reader might have already recognized as the right hand side of the KF eq. (3.12). In fact, it is easy to see that we can rewrite that equation more compactly as:

$$\partial_t g = \mathcal{A}^* g \tag{3.24}$$

where $\partial_t g = \frac{\partial g}{\partial t}$.

In order to try and build some intuition about adjoint operators, it is useful to mention that in finite dimensions – where operators can be represented by matrices (e.g. as in the case of CTMC) – the Hermitian adjoint is exactly equivalent to the conjugate transpose. It immediately follows that, if all the elements of a matrix are real, as they usually are in economics applications, the Hermitian adjoint and transpose are the same.[23] In this sense, if the concept of adjoint seem too abstract and unclear, we can simply think of it as the generalization to infinite dimensions of the matrix transpose.

Because when solving our models numerically we will always have to approximate the state space in finite dimensions, this connection between adjoint operators and matrix transposes is going to be extremely useful. In fact, if we can write the finite-dimensional representation of an operator $\mathcal{A}$ we will also get the equivalent representation of its adjoint $\mathcal{A}^*$ "for free" (at the cost of one matrix transposition).

3.5. Example: Ornstein-Uhlenbeck process

In order to better understand the concepts of infinitesimal generator, adjoint, and Kolmogorov Forward Equation, we can use the now familiar Ornstein-Uhlenbeck process as an example and look at how to find its stationary distribution. By definition the process can be written as:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t$$

where $\mu(X_t) = \eta(\alpha - X_t)$ and $\sigma(X_t) = \sigma$.

Using eq. (3.21), we can directly apply the formula to write down the infinitesimal generator of the process:

$$\left(\mathcal{A}f\right)(x) = \eta(\alpha - x)f'(x) + \frac{\sigma^2}{2}f''(x). \tag{3.25}$$

Notice that eq. (3.25) is surprisingly similar to the right hand side of the HJB equation in chapter 2: the first term is given by the drift of the state variable times the first

[23] The conjugate transpose of a matrix is the same as the transpose except that along with switching rows and column elements one also need to take the complex conjugate of all elements.

derivative of the function being considered; while the second term is an additional new one due to the presence of the stochastic term dW_t. In fact, when writing down the finite-difference matrix $\mathbf{A}$ in chapter 2 what we were doing was exactly finding a finite-difference approximation to the infinitesimal generator $\mathcal{A}$.

Alternatively, using eq. (3.23) we can also directly write down the adjoint of $\mathcal{A}$ instead:

$$\left(\mathcal{A}^* g\right)(x) = -\frac{\partial}{\partial x}\left[\eta(\alpha - x)g(x)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[g(x)\sigma^2\right] \tag{3.26}$$

where, as we just argued in the last section, eq. (3.26) is nothing else than the right hand side of the KFE. Hence, in order to find the stationary distribution of our process we will need to find the g such that eq. (3.26) equals zero. In the specific case of the Ornstein-Uhlenbeck process this could in principle be done analytically. However, since this cannot usually be done for more complicated processes, we will instead solve eq. (3.26) numerically. Specifically, just as we solved the HJB in chapter 2, we will write a finite-difference approximation of $\mathcal{A}^*$ and then find a way to set the discretized KFE to zero.

3.5.1. Numerical Solution

We already saw in chapter 2 how to find a finite-difference approximation of the drift term in $\mathcal{A}$ by using an upwind scheme and we previously argued that we can get the FD approximation of $\mathcal{A}^*$ simply as the matrix transpose of (the FD approximation of) $\mathcal{A}$. That is, if we could also find a FD approximation of the diffusion term in $\mathcal{A}$, we would be one matrix transpose away from finding a FD approximation of $\mathcal{A}^*$. Summarizing, in order to numerically approximate $\mathcal{A}^*$, we need to do two things: first, define a finite-difference approximation of $\frac{1}{2}\sigma^2(x)f''(x)$ – which allows us to build our numerical approximation $\mathbf{A}$ of $\mathcal{A}$ – then, to find the approximation of $\mathcal{A}^*$, we simply need to take the transpose of $\mathbf{A}$.[24]

To approximate $\frac{1}{2}\sigma^2(x)f''(x)$ we obviously need an approximation of the second derivative f''. It turns out that the following central difference approximation is a good choice:

$$f''(x) = \lim_{h\to 0}\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \approx \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta_i^2} \tag{3.27}$$

where f_{i-1}, f_i and f_{i+1} are the values of the function f at the points x_{i-1}, x_i and x_{i+1} $\forall i \in 2, \ldots, I-1$, respectively, considering a grid of point on which we approximate the function $\{x_j\}_{j=1}^I$.[25]

By definition the infinitesimal generator is a linear operator; therefore $\mathbf{A}$, our FD approximation of $\mathcal{A}$, will be given by the sum of the FD approximation of the drift term

[24] We could in principle discretize $\mathcal{A}^*$ directly. However, because the "time directions" of $\mathcal{A}$ and $\mathcal{A}^*$ are opposite of each other, the upwind scheme would then need to be modified accordingly. It is therefore a lot simpler (and in line with what we will do when solving HA models) to first approximate $\mathcal{A}$ by $\mathbf{A}$ and then find its adjoint $\mathcal{A}^*$ by taking the transpose $\mathbf{A}^T$.

[25] Notice that the above limit for $f''(x)$ simply comes from

$$f''(x) = \lim_{h\to 0}\frac{f'(x+h) - f'(x)}{h} = \lim_{h\to 0}\frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h} = \lim_{h\to 0}\frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

and the FD approximation of the Ito correction term:

$$\frac{f_{i+1}-f_i}{\Delta x}\mu_{i,F}^{+}+\frac{f_i-f_{i-1}}{\Delta x}\mu_{i,B}^{-}+\frac{1}{2}\frac{f_{i+1}-2f_i+f_{i-1}}{(\Delta x)^2}\sigma_i^2$$

or, in matrix notation,

$$\underbrace{\begin{pmatrix} \underbrace{-\frac{\mu_{i,B}^{-}}{\Delta x}+\frac{1}{2}\frac{\sigma_i^2}{(\Delta x)^2}}_{\text{inflow}_{i-1}\geq 0} & \underbrace{\frac{\mu_{i,B}^{-}}{\Delta x}-\frac{\mu_{i,F}^{+}}{\Delta x}-\frac{\sigma_i^2}{(\Delta x)^2}}_{\text{outflow}_i\leq 0} & \underbrace{\frac{\mu_{i,F}^{+}}{\Delta x}+\frac{1}{2}\frac{\sigma_i^2}{(\Delta x)^2}}_{\text{inflow}_{i+1}\geq 0} \end{pmatrix}}_{i^{th}\text{ row of }\mathbf{A}} \begin{pmatrix} f_{i-1} \\ f_i \\ f_{i+1} \end{pmatrix}$$

where the terms in black are essentially the same as in chapter 2, while the terms in red are the new entries pertaining to our approximation of the correction term $\frac{\sigma^2(x)}{2}f''(x)$.

Boundary Conditions — In principle, we would be at this point ready to find a numerical solution to our KF eq. (3.26). However, just as an ODE requires initial conditions to have a unique solution, a PDE requires boundary conditions (and possibly initial conditions) for uniqueness. Without these conditions, a PDE admits infinitely many solutions. We will discuss boundary conditions more in detail in the next chapter, suffice here to say that in the context of PDEs, boundary conditions express the behavior of the solution on the boundary of its domain.

Why then, were we able to ignore boundary conditions when finding the solution to the HJB equation in chapter 2? To answer this question it is worth noting that the problem we currently face is essentially that – because we are discretizing our state space onto a grid – we need to restrict the domain of our process, which in this example is $(-\infty,\infty)$, to a finite grid. When the process has no diffusion component ($\sigma=0$), as in chapter 2, explicit boundary conditions are sometimes unnecessary when the dynamics naturally keep the state within bounds – the deterministic drift pointed inward at both boundaries, and the upwind scheme automatically handled this by using forward differences at the lower boundary and backward differences at the upper boundary.

However, with diffusion ($\sigma\neq 0$), the situation is fundamentally different. Even when the drift points inward at the boundaries (as in our OU process where the drift pulls toward the mean), the diffusion term creates probability flux at the boundaries that must be explicitly handled. Essentially, there is always a non-zero probability that a sufficiently large Brownian shock pushes the process outside our grid. Without proper boundary conditions, probability mass would "leak" out of our computational domain.

This creates a technical problem: the centered difference approximation of the diffusion term $\frac{1}{2}\frac{\partial^2}{\partial x^2}\left[g(x)\sigma^2\right]$ requires values at $i-1$ and $i+1$. At the boundaries $i=1$ and $i=I$, this would require "ghost nodes" at positions $i=0$ and $i=I+1$ that lie outside our grid.[26] By imposing appropriate boundary conditions, we can specify how to handle probability mass

[26] Simply ignoring these diffusion terms at the boundaries would violate the requirement that rows of $\mathbf{A}$ sum to zero, which, as we saw in section 3.4.1, is a key property of infinitesimal generators.

that would otherwise escape the grid, allowing us to construct a well-defined generator matrix on our finite domain.

Solving the KFE

We now have a way to write $\mathbf{A}$ – our finite-difference approximation of $\mathcal{A}$ – and, in order to find a FD approximation of its hermitian adjoint $\mathcal{A}^*$ it is simply sufficient to take the transpose $\mathbf{A}^T$. To summarize, we have:

- $\mathbf{A}$ is the discretized version of $\mathcal{A}$
- $\mathbf{A}^T$ is the discretized version of $\mathcal{A}^*$

Hence, the KF eq. (3.24) can be approximated as:

$$\frac{\mathbf{g}^{n+1} - \mathbf{g}^n}{\Delta} = \mathbf{A}^T \mathbf{g}^\bullet \tag{3.28}$$

where by $\mathbf{g}^\bullet$ we simply mean that to solve the system we can use either an explicit method and replace $\mathbf{g}^\bullet$ with $\mathbf{g}^n$ on the right-hand side or an implicit method and replace it with $\mathbf{g}^{n+1}$ – which have the same advantages and disadvantages as in chapter 2.

To find the stationary distribution such that $0 = \mathcal{A}^* g$ we can then proceed in two ways: the more intuitive (though slower) way is to simply do time-iteration on the discretized eq. (3.28) starting from an initial guess $\mathbf{g}^0$ until convergence – i.e. solve the initial-value problem – using either an implicit or explicit scheme.[27] This time-iteration approach works because, starting from any initial distribution, the system converges to its stationary distribution as $t \to \infty$.[28] In practice, one typically iterates until $\|\mathbf{g}^{n+1} - \mathbf{g}^n\| < \varepsilon$ for some small tolerance ε (e.g., 10^{-6}). The second approach, which is generally faster since it usually involves one single backslash operator, is to instead directly solve the eigenvalue problem $\mathbf{A}^T \mathbf{g} = 0$. While faster, this approach requires more care numerically. The key insight is that any stationary distribution must be in the null space of $\mathbf{A}^T$, which we can find by solving the eigenvalue problem. However, as we'll see, the singularity of $\mathbf{A}$ requires special handling.

Given the FD approximation of $\mathcal{A}^*$, `AT`, the following code block shows how to solve the initial-value problem in eq. (3.28) using the implicit scheme:[29]

[27] Note that since $\mathbf{g}$ is a distribution, it must always sum to 1 (and so the starting distribution $\mathbf{g}^0$ must also be a density). In theory it can be shown that the fact that the rows of $\mathbf{A}$ sum to zero also implies mass preservation of $\mathbf{g}$ throughout the time-iteration steps. In practice, however, the rows of $\mathbf{A}$ might not sum *exactly* to zero due to numerical inaccuracies. It is therefore usually recommended to apply a normalization at each time step.

[28] Provided that: (i) a unique stationary distribution exists – which requires the continuous-time Markov process to be irreducible and positive recurrent – and (ii) the discretization preserves the key properties of the continuous operator, particularly that $\mathbf{A}$ remains a valid generator matrix (rows sum to zero) and that, if using an explicit scheme, the time step Δ is sufficiently small to satisfy the Courant-Friedrichs-Lewy condition for stability – specifically, $\Delta \leq \frac{(\Delta x)^2}{\sigma^2_{\max}}$ where $\sigma^2_{\max}$ is the maximum diffusion coefficient on the grid. For diffusion processes with reflecting boundaries, the first condition is typically satisfied when the drift points inward at the boundaries (ensuring the process doesn't get "stuck") and the diffusion coefficient is positive in the interior of the domain (ensuring all states communicate); these are clearly satisfied for our OU process.

[29] This is essentially the same code that was used to generate fig. 3.5.

```julia
# Initial value problem
for n = 1:N
    D = SparseArrays.I - Δ*AT

    # Solve system of equations
    g[n + 1] = D\g[n]
    g[n + 1] ./= sum(g[n + 1].*dx) # renormalize to ensure pdf sums to 1
end
```

If instead we are not actually interested in the evolution of g_t starting from some initial distribution g_0 and we only want to find the stationary distribution, we can also directly solve the system $\mathbf{A}^T\mathbf{g} = 0$. This is equivalent to finding the eigenvector associated to the zero eigenvalue; however, exactly because the generator matrix $\mathbf{A}$ has a zero eigenvalue, we also know that it is singular and we cannot just directly use our backslash operator on $\mathbf{A}^T$.[30] There are several ways to handle the singularity of $\mathbf{A}$, the approach we follow here essentially boils down to the following: first, "remove" an equation from our system so that we are left with $I - 1$ equations in I unknowns; second, assign a value for one of the unknowns and solve the system for all others; finally, renormalize so that the density sums to 1.[31,32] "Essentially we remove one of the degrees of freedom by setting one of the elements of π equal to unity and later renormalize to get the correct stationary probability vector" (Stewart 2021, p. 75).

The next code block shows how to implement the "remove an equation" approach to solve for the stationary distribution in eq. (3.28):

```julia
# "Remove an equation" approach

# Choose any one equation to be removed, otherwise matrix is singular
i_fix = 10

# "Remove" equation from the system
AT[i_fix, :] .= 0
AT[i_fix, i_fix] = 1

# Fix "initial" solution for that unknown
b = zeros(Nx)
b[i_fix] = 1.0          # can be any value >0, will be renormalized

# Solve system of equations
g = AT\b
g ./= sum(g.*dx)        # renormalize to ensure pdf sums to 1
```

[30] Incidentally, a homogeneous system of n linear equations in n unknowns has a solution other than the trivial solution ($x_i = 0\ \forall i$) if and only if the coefficient matrix is singular. That is, the fact that $\mathbf{A}$ is singular is specifically what ensures that $\mathbf{A}^T\mathbf{g} = 0$ has a nontrivial solution.

[31] See Stewart (2021, section 2.3.1) for a detailed explanation of this approach as well as others

[32] The choice of which equation to remove (i.e., which i_{fix} to use) typically doesn't matter for well-behaved problems with a unique stationary distribution. However, for numerical stability, it's often best to choose an index where we expect the stationary distribution to have substantial mass.

Chapter 4

Stochastic HJB Equations

Chapter 5

Aggregate Shocks

Appendices

A. Deterministic Optimal Control

A.1. How to draw a phase diagram

Given the system of ODEs as in (1.10) and (1.11) with the associated initial and transversality conditions:

1. Calculate the so called nullclines, by imposing $\dot{c} = 0$ and $\dot{k} = 0$
2. Draw the nullcline associated to $\dot{c} = 0$ in the (c, k)-space and analyze how consumption moves to each side of this line. In order to do that it is usually sufficient to analyze one of the two cases (e.g., $\dot{c} > 0$) and to find for which values of k the inequality holds.
3. Draw the nullcline associated to $\dot{k} = 0$ in the (c, k)-space and analyze how capital moves to each side of this line. As before, it is usually enough to analyze one of the two cases and to find for which values of c the inequality holds.
4. Finally, considering the graph altogether, we will have the direction of every combination of points into the (c, k)-space, and we will be able to draw all the arrows.

For a more general treatment of phase diagrams, see Hirsch et al. (2004, Chapter 9).

B. Deterministic HJB Equations

B.1. The Envelope Theorem

Theorem 5: *Consider the problem:*

$$v(x) = \max_{\alpha} g(x, \alpha)$$

where x represent the state variable of the problem and α the control variable. Then:

$$\frac{dv(x)}{dx} = \frac{\partial g(x, \alpha)}{\partial x}\Big|_{\alpha=\alpha^*(x)}$$

with:

$$\alpha^*(x) = \arg\max_{\alpha} g(x, \alpha)$$

Proof. Let's note first that:

$$v(x) \equiv g(x, \alpha^*(x)) \tag{1}$$

Taking the total derivative of $v(x)$:

$$\frac{dv(x)}{dx} = \frac{\partial g(x, \alpha^*(x))}{\partial x} + \frac{\partial g(x, \alpha^*(x))}{\partial \alpha}\frac{\partial \alpha^*(x)}{\partial x}$$

Note that:

$$\frac{\partial g(x, \alpha^*(x))}{\partial \alpha} = 0$$

by the first-order condition (it is nothing else than the function $g(x, \alpha)$ evaluated at the optimal control $\alpha^*(x)$, and we know that the derivative at that point is 0.) Then:

$$\frac{dv(x)}{dx} = \frac{\partial g(x, \alpha^*(x))}{\partial x} = \frac{\partial g(x, \alpha)}{\partial x}\Big|_{\alpha=\alpha^*(x)}$$

□

In other words what the envelope theorem states is that the derivative of a (value) function with respect to the state variable is equal to the derivative of the function $g(x, \alpha)$ with respect to the state variable, evaluated at the optimal control $\alpha^*(x)$.

In the context of the neoclassical growth model, where we have:

$$\rho v(k) = \max_{c} u(c) + v'(k)\left(F(k) - \delta k - c\right)$$

we can apply the envelope theorem to the value function $v(k)$:

$$\rho\frac{dv(k)}{dk} = \frac{\partial g(k, c)}{\partial k}\Big|_{c=c^*(k)}$$

where $g(k, c) = u(c) + v'(k)\left(F(k) - \delta k - c\right)$ and $c^*(k)$ is the optimal control. Note that

normally we don't write explicitly consumption c as a function of the state variable k but it's clear that this is always the case. We get:

$$\rho v'(k) = v''(k)\left(F(k) - \delta k - c\right) + v'(k)\left(F'(k) - \delta\right)$$

as shown in section 2.1.

B.2. Non-uniform Grids

In the context of heterogeneous-agent models, it is sometimes useful to discretize the state space on a non-uniform grids. This is because it is often the case that the value function present more curvature closer to the borrowing constraint, while at the same time being close to linear far away from it. Since the FD methods introduced in these notes belong to the class of linear interpolation methods (because they essentially interpolate the value function between grid points using a linear function), the approximation error of these methods increase as the curvature of the function to be approximated increases. Non-uniform grids can therefore be particularly useful in these cases: instead of improving the approximation accuracy by using other (e.g. higher order) methods, which can be slower than linear methods, we instead modify the grid to improve the accuracy only where we need to.

One of the most commonly used non-uniform grids is the power-spaced grid where the grid points are spaced according to a power function. We can define the grid as follows:

- Let $[\underline{a}, \bar{a}]$ be the possible range for our state variable a
- Let $\mathcal{Z}$ be a uniformly spaced grid on $[0, 1]$: $\mathcal{Z} = \{z_i : z_{i+1} = z_i + \Delta \ \forall i\}$
- For each grid point $z_i \in \mathcal{Z}$, define $x_i = z_i^\alpha$ for some $\alpha \in (1, \infty)$ to create a non-uniform grid $\mathcal{X}$ on $[0, 1]$. Note that as $\alpha \to \infty$, $\mathcal{X}$ becomes denser and denser closer to 0
- Then, create asset grid $\mathcal{A}$ by rescaling each $x \in \mathcal{X}$

$$a_i = \underline{a} + (\bar{a} - \underline{a})x_i$$

Notice that of course, in the presence of non-uniform grids, our approximation schemes need to be appropriately modified. For example, for the first derivative approximations defined in chapter 2, the scheme becomes:

- Backward difference:
$$v'_{i,B} = \frac{v_i - v_{i-1}}{k_i - k_{i-1}} \approx v'(k_i)$$
- Forward difference:
$$v'_{i,F} = \frac{v_{i+1} - v_i}{k_{i+1} - k_i} \approx v'(k_i)$$
- Central difference:
$$v'_{i,C} = \frac{v_{i+1} - v_{i-1}}{k_{i+1} - k_{i-1}} \approx v'(k_i)$$

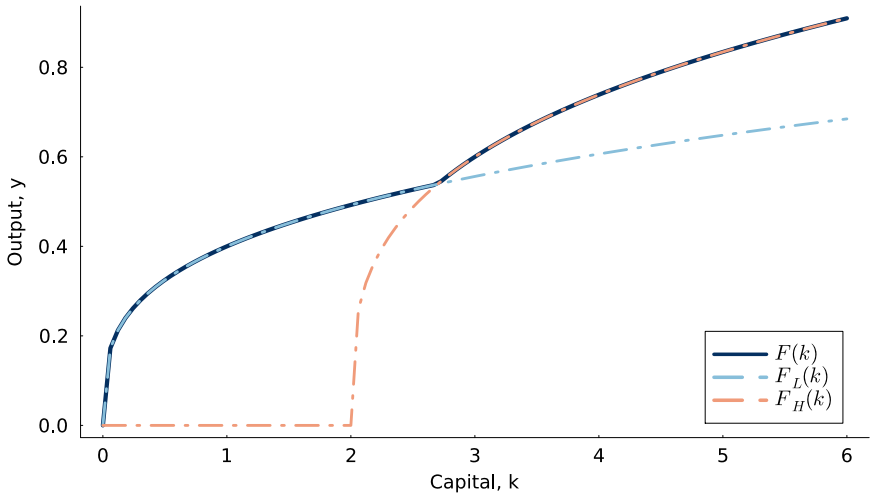

Figure 1: A "butterfly" production function.

B.3. Non-Convexities

In section 2.2.2 we briefly discussed the need to sometimes perform a Hamiltonian "check" if the value function is (locally or globally) convex. Here we look at a classic example of a model featuring non-convexities in the production function and show how the upwind scheme of section 2.2.2 can be very easily extended to cover such cases.

Let's consider again the neoclassical growth model:

$$\rho v(k) = \max_c u(c) + v'(k)\left(F(k) - \delta k - c\right)$$

but drop the assumption that F is strictly concave, which we instead define as follows:

$$\begin{aligned} F(k) &= \max\left\{F_L(k), F_H(k)\right\} \\ F_L(k) &= A_L k^\alpha \\ F_H(k) &= A_H(\max\{k-\kappa, 0\})^\alpha \end{aligned}$$

where $\kappa > 0$ and $A_H > A_L$. Figure 1 shows how the production function looks like under this assumption.

In discrete time the usual first order condition:

$$u'(F(k) - \delta k - k') = \beta v(k')$$

would no longer be sufficient, because the model typically has multiple solutions. In continuous time, on the other hand, nothing changes and we can continue using the exact same algorithm (and code) that we used to solve the standard version of the model. The only thing we need to add is the Hamiltonian "check" mentioned in section 2.2.2.

The code snippet in fig. 2 shows how this check is implemented in practice: if the objective is at a maximum – that is, either if $s_{i,F} \leq 0 \leq s_{i,B}$ or if the steady-state

```
# Upwind scheme under non-convexities

H_F = @. u(c_F) + dv_F*s_F  # Forward Hamiltonian
H_B = @. u(c_B) + dv_B*s_B  # Backward Hamiltonian
H_0 = @. u(c_0)             # Steady-state Hamiltonian

# objective is concave
I_concave = @. ((s_B <  0) && (s_F <= 0)) || ((s_B >= 0) && (s_F > 0))
# objective is convex
I_convex  = @. ((s_B <  0) && (s_F >  0))

# When to choose "steady-state"
I_max     = @. ((s_B >= 0) && (s_F <= 0)) || (H_0 >= max(H_B, H_F)
# When to choose backward difference
I_B = @. ~I_max && (I_concave && (s_B < 0)) || (I_convex && (H_B >= H_F))
# When to choose forward difference
I_F = @. ~I_max && (I_concave && (s_F > 0)) || (I_convex && (H_F >= H_B))

dv_upwind = @. dv_F*I_F + dv_B*I_B + dv_0*I_max
c         = @.  c_F*I_F +  c_B*I_B +  c_0*I_max
# previous line is equivalent to u'_inv.(dv_upwind) but better behaved
```

Figure 2: Julia code for Hamiltonian adjustment

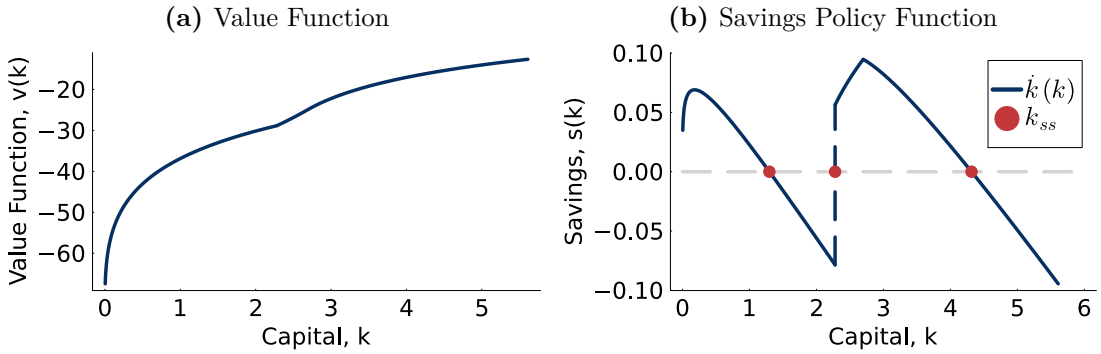

Figure 3: A model with non-convexities

Hamiltonian $H_{i,0} \equiv u(c_{i,0})$ is larger than both $H_{i,B}$ and $H_{i,F}$, as defined in eqs. (2.8) and (2.9) – the updated upwind scheme uses the "steady-state" solution (i.e. zero-drift); otherwise it chooses the backward difference either if the objective is concave and the backward drift is negative or if the objective is convex and the backward Hamiltonian is greater than the forward Hamiltonian; finally, it chooses the forward difference either if the objective is concave and the forward drift is positive or if the objective is convex and the forward Hamiltonian is greater than the backward Hamiltonian. Once this adjustment is in place, the rest of the code remains exactly identical to the one in section B.4 and the model can be solved using the exact same algorithm.

Figure 3 plots both the value function and the savings policy function of our model. The model has been solved using the same parameters as in **??**, and the additional parameters A_H, A_L, and κ have been set to 0.4, 0.6, and 2, respectively. By looking at panel (a) we

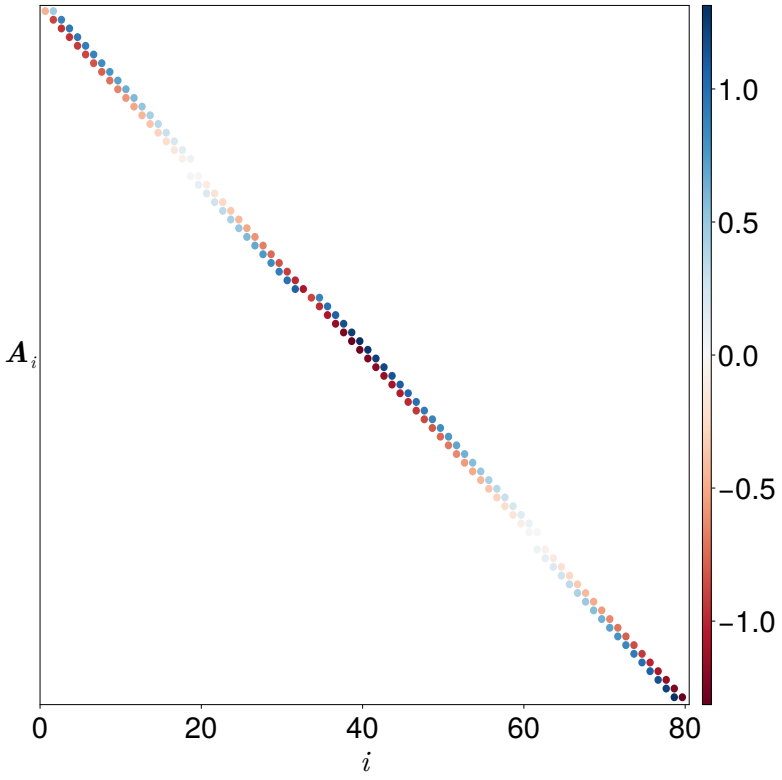

Figure 4: Visualization of the **A** matrix for the model with non-convexities

can immediately see that the non-convexity in the production function translated also to the value function (which is what required us to adapt the upwind scheme). Somewhat more interestingly – at least from an economic perspective – by analyzing panel (b) we can also characterize the behavior of our economy. As mentioned above, our model will feature multiple equilibria; in particular, the model features two stable equilibria and an unstable one. These steady-states can be found as the points in which the savings policy function (the drift of our state variable, $\dot{k}$) crosses the zero line. For sufficiently high levels of initial capital, the system will monotonically converge to the "good" steady state while, if initial capital is too low, the system will monotonically converge to the "bad" steady state – a classical example of a model featuring a poverty trap. The middle equilibrium is instead unstable because, for any infinitesimally small deviations from the "correct" initial condition, the system will diverge towards either the good or the bad steady state. Finally, to further stress the point that the discretized matrix **A** contains *all* relevant information about the problem, fig. 4 plots the usual visualization for this model – which essentially just shows what we just analyzed looking at the savings policy function.[1]

[1] Unlike the rest of the plots in this section, the visualization in fig. 4 has been generated by solving the model on a restricted grid with only 80 points to let all three equilibria feature in the same graph while still being readable.

B.4. Code for the Neoclassical Growth Model

```julia
#==================================#
# Preliminaries

# You might first need to add packages to your environment if they are not available
# You can do so by running: using Pkg; Pkg.add("SparseArrays")
using SparseArrays

#==================================#
# Model Parameters

# Preferences
γ = 2       # RRA
ρ = 0.05    # Discount rate

u(c) = (c^(1 - γ) - 1)/(1 - γ)  # Utility function
u'(c) = c^(-γ)                  # Marginal utility function
u'_inv(x) = x^(-1/γ)            # Inverse of marginal utility function

# Technology
Z = 1       # TFP
α = 0.3     # Capital share
δ = 0.05    # Depreciation

# Production function
F(k) = Z*k^α

# Steady state values
k_ss = (Z*α/(ρ + δ))^(1/(1 - α))
c_ss = F(k_ss) - δ*k_ss

#==================================#
# Grids

# Capital
k_min = k_ss/1000
k_max = 2*k_ss
Nk = 10_000

k_grid = range(k_min, k_max, Nk)
dk = k_grid[2] - k_grid[1]

#==================================#
# Solver parameters

max_iter = 1_000
tol = 1e-6
Δ = 1000
iter = 0

#==================================#
# Initialization

A = SparseMatrixCSC{Float64, Int64}(undef, Nk, Nk)

v0 = @. u(F(k_grid) - δ*k_grid)/ρ
pol_c = similar(v0)
pol_s = similar(v0)

#==================================#
```

```
# Model Solution

while iter <= max_iter
    iter += 1

    v_diff = diff(v0)/dk

    # Forward difference
    dv_F = [v_diff; u'(F(k_max) - δ*k_max)]
    c_F  = u'_inv.(dv_F)
    s_F  = @. F(k_grid) - δ*k_grid - c_F

    # Backward difference
    dv_B = [u'(F(k_min) - δ*k_min); v_diff]
    c_B  = u'_inv.(dv_B)
    s_B  = @. F(k_grid) - δ*k_grid - c_B

    # "Steady-state"
    c_0  = @. F(k_grid) - δ*k_grid
    dv_0 = u'.(c_0)

    # Upwind scheme chooses forward or backward differences based on the sign of the drift
    I_F = s_F .> 0          # Positive drift -> Forward difference
    I_B = s_B .< 0          # Negative drift -> Backward difference
    I_0 = 1 .- I_F .- I_B   # Steady state

    dv_upwind = @. dv_F*I_F + dv_B*I_B + dv_0*I_0
    pol_c = u'_inv.(dv_upwind)
    pol_s .= @. F(k_grid) - δ*k_grid - pol_c
    u_vec = u.(pol_c)

    # Elements of the A matrix
    elem_α = -min.(s_B, 0)/dk   # lower diagonal
    elem_ξ = max.(s_F, 0)/dk    # upper diagonal
    elem_β = - elem_α - elem_ξ  # main diagonal

    # Finite difference A matrix
    A = spdiagm(-1 => elem_α[2:end],        # lower diagonal
                 0 => elem_β,               # main diagonal
                 1 => elem_ξ[1:(end - 1)])  # upper diagonal

    # Check if A matrix is proper
    maximum(abs.(sum(A, dims = 2))) <= 1e-9 || error("Not all rows of A matrix sum to 0")

    # Implicit scheme
    B = (ρ + 1/Δ)*SparseArrays.I - A
    d = u_vec + v0/Δ

    # Solve system of equations
    v = B\d

    hjb_diff = maximum(@. abs(v - v0)/(1 + abs(v0)))
    v0 = copy(v)

    println("Iteration #", iter, "; Distance = ", hjb_diff)
    hjb_diff < tol && break
end
```

C. Stochastic Calculus

C.1. KFE Derivation

Suppose we have a stochastic process $X_t \in \mathbb{R}$ that evolves according to the following diffusion:

$$dX = \mu(X,t)dt + \sigma(X,t)dW \tag{2}$$

where W_t is an independent 1-dimensional Brownian motion process.

We define as G_t the distribution of state X at time t, and as g_t the corresponding probability density. The goal is to derive a differential equation that defines how the density, g_t, evolves over time. Such a differential equation is the so-called Kolmogorov Forward Equation (KFE) or Fokker-Planck Equation. Recall proposition 1:

Proposition 1 [Kolmogorov Forward Equation]**:** *Let X_t be an Itô process described by:*

$$dX = \mu(X,t)dt + \sigma(X,t)dW.$$

Then, $g(x,t)$ – the distribution of X at time t – satisfies the following Kolmogorov Forward equation:[2]

$$\frac{\partial g(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[\mu(x,t)g(x,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[\sigma^2(x,t)g(x,t)\right]. \tag{3.12}$$

Proof

Step 1: Setup

We start by defining an arbitrary function of our process $f(x)$. This function is completely arbitrary and only serves the purpose of being able to apply Itô's lemma. In fact, exactly because it is arbitrary, we will eventually want to get rid of it. We assume that f is twice continuously differentiable, that it has compact support (i.e. it vanishes outside its support), and that its first and second derivatives also have the same support. That is there exists x_1 and x_2 such that:

$$f(x) = f_x(x) = f_{xx}(x) = 0 \;\forall\; x \notin [x_1, x_2].$$

To ease notation, whenever there is no confusion, we will drop the dependencies of functions on x and t, so for example $\mu(x,t)$ and $\sigma(x,t)$ will simply be written as μ and σ.

Step 2: Apply Itô's Lemma

By applying Itô's lemma we have the usual

$$df = \left(f_x\mu + \frac{1}{2}f_{xx}\sigma^2\right)dt + f_x\sigma dW$$

[2] A derivation of the KFE for univariate diffusion processes can be found in section C.1.

Taking expectations on both sides with respect to the density of x we have:

$$\begin{aligned}\mathbb{E}[df] &= \mathbb{E}\left[\left(f_x\mu + \frac{1}{2}f_{xx}\sigma^2\right)dt + f_x\sigma dW\right] \\ &= \mathbb{E}\left[f_x\mu + \frac{1}{2}f_{xx}\sigma^2\right]dt\end{aligned}$$

where in the second line we have used the fact that in the "cross-section", the expectation of the Brownian motion term is zero (remember that expectations are nothing else than the integral with respect to the probability density of x).[3]

Next, we exchange the order of integration and differentiation on the left-hand side and divide both terms by dt to get:[4]

$$\frac{d}{dt}\mathbb{E}[f] = \mathbb{E}\left[f_x\mu + \frac{1}{2}f_{xx}\sigma^2\right].$$

At this point, let's expand the expectation term as the integral with respect to the probability distribution of x, $g(x,t)$ (which remember is the main object of interest):

$$\frac{d}{dt}\int_{-\infty}^{\infty} f(x)g(x,t)dx = \int_{-\infty}^{\infty}\left(f_x(x)\mu(x,t) + \frac{1}{2}f_{xx}(x)\sigma^2(x,t)\right)g(x,t)dx \tag{3}$$

$$= \int_{-\infty}^{\infty} f_x(x)\mu(x,t)g(x,t)dx + \frac{1}{2}\int_{-\infty}^{\infty} f_{xx}(x)\sigma^2(x,t)g(x,t)dx. \tag{4}$$

Step 3: Integration by Parts

The "problem" that we have now is that on the right-hand side we have the derivatives of f, which we want to get rid of (remember f is just an auxiliary function we will want to discard). In order to do so, we can apply the integration by parts formula, which remember can be stated as:

$$\int udv = uv - \int vdu$$

Let's start with the first term on the right-hand side of eq. (4). Set $dv = f_x(x)dx$ and $u = \mu(x,t)g(x,t)$, then integrate dv to get $v = f(x)$ and differentiate u to get $du = \frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right)dx$, and plug both into the integration by parts formula:

$$\begin{aligned}\int_{-\infty}^{\infty} f_x(x)\mu(x,t)g(x,t)dx &= f(x)\mu(x,t)g(x,t)\big|_{x=-\infty}^{x=\infty} - \int_{-\infty}^{\infty} f(x)\frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right) \\ &= -\int_{-\infty}^{\infty} f(x)\frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right)dx.\end{aligned}$$

where in the second equality we have used the fact that by assumption the function f vanishes at both ends of the interval and the probability tails down, so the boundary terms vanish.[5]

[3] Notice that, if dW was a common noise term, its "cross-sectional" expectation would not be zero.

[4] This can be justified by invoking the dominated or bounded convergence theorem.

[5] Note that even if we do not know the formula for the probability density we can justify this by invoking

Now for the second term we will need to apply the same procedure (twice), first we set $dv = f_{xx}(x)dx$ and $u = \sigma^2(x,t)g(x,t)$, then we integrate dv to get $v = f_x(x)$ and differentiate u to get $du = \frac{\partial}{\partial x}\left(\sigma^2(x,t)g(x,t)\right)dx$:

$$\begin{aligned}\int_{-\infty}^{\infty} f_{xx}(x)\sigma^2(x,t)g(x,t)dx &= f_x(x)\sigma^2(x,t)g(x,t)\big|_{x=-\infty}^{x=\infty} - \int_{-\infty}^{\infty} f_x(x)\frac{\partial}{\partial x}\left(\sigma^2(x,t)g(x,t)\right)\\ &= -\int_{-\infty}^{\infty} f_x(x)\frac{\partial}{\partial x}\left(\sigma^2(x,t)g(x,t)\right)dx.\end{aligned}$$

where again the first term vanishes due to the boundary conditions we imposed on f and we are left with the second term, which now has been "lowered" by one degree and only includes the first-derivative of f. We can apply the usual steps once more to get rid of it, which means we eventually have:

$$\int_{-\infty}^{\infty} f_{xx}(x)\sigma^2(x,t)g(x,t)dx = \frac{1}{2}\int_{-\infty}^{\infty} f(x)\frac{\partial^2}{\partial x^2}\left(\sigma^2(x,t)g(x,t)\right)dx.$$

Putting everything together we have:

$$\frac{d}{dt}\int_{-\infty}^{\infty} f(x)g(x,t)dx = -\int_{-\infty}^{\infty} f(x)\frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right)dx + \frac{1}{2}\int_{-\infty}^{\infty} f(x)\frac{\partial^2}{\partial x^2}\left(\sigma^2(x,t)g(x,t)\right)dx.$$

Step 4: Differentiate wrt t

Now on the left-hand side we exchange the order of integration and differentiation to get:

$$\int_{-\infty}^{\infty} f(x)\frac{\partial g(x,t)}{\partial t}dx = -\int_{-\infty}^{\infty} f(x)\frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right)dx + \frac{1}{2}\int_{-\infty}^{\infty} f(x)\frac{\partial^2}{\partial x^2}\left(\sigma^2(x,t)g(x,t)\right)dx.$$

We can now combine the integrals on both sides of the equation:

$$\int_{-\infty}^{\infty} f(x)\left[\frac{\partial g(x,t)}{\partial t} + \frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right) - \frac{1}{2}\frac{\partial^2}{\partial x^2}\left(\sigma^2(x,t)g(x,t)\right)\right]dx = 0.$$

Since f was chosen arbitrarily, in order for the left hand side to be zero, it must be the case that the term in square brackets is equal to zero, which gives us the Kolmogorov Forward Equation:

$$\frac{\partial g(x,t)}{\partial t} + \frac{\partial}{\partial x}\left(\mu(x,t)g(x,t)\right) - \frac{1}{2}\frac{\partial^2}{\partial x^2}\left(\sigma^2(x,t)g(x,t)\right) = 0. \tag{5}$$

C.2. p-Variation

A large part of stochastic calculus is concerned about constructing integrals of stochastic processes. In order to do so, however, one needs to have a formal way of articulating how

the fact that the probability that a "particle" travels infinite distance in a infinitesimal amount of time is going to be zero.

much "variation" a stochastic process has.[6]

Definition 12 [Partition]**:** *The set of points* $\mathcal{P} = \{t_0, \ldots, t_n\}$ *with* $0 = t_0 < t_1 < \ldots < t_n = t$ *is a partition of the interval* $[0, t]$*. The norm of the partition is* $l(\mathcal{P}) = \max|t_j - t_{j-1}|$.

Definition 13 [p-Variation]**:** *Let* $X : \Omega \times \mathcal{T} \to \mathbb{R}$ *be a stochastic process. For any* $p > 0$*, we can define the* p*-th variation process of* X_t *as the following:*

$$\langle X, X \rangle_t^p(\omega) = \lim_{l(\mathcal{P}) \to 0} \sum_{j=0}^{n-1} |X_{t_{j+1}}(\omega) - X_{t_j}(\omega)|^p.$$

That is, $\langle X, X \rangle_t^p(\omega)$ is a measure of how much a stochastic process "moves around", and it is weakly decreasing in p (i.e. it is easier to have finite p-variation as p increases). The intuition is simply that, as p increases, the term $|X_{t_{j+1}}(\omega) - X_{t_j}(\omega)|^p$ gets smaller and smaller as the partition gets finer (because $|X_{t_{j+1}}(\omega) - X_{t_j}(\omega)| \to 0$ as $l(\mathcal{P}) \to 0$).

For $p = 1$, this is called total variation (or simply variation) process; for $p = 2$, this is the quadratic variation process. In stochastic calculus quadratic variation and covariation are particularly important. When there is no risk of confusion, we will denote quadratic variation as $[X]_t$ or $[X, X]_t$.

We can also define the quadratic covariation between two stochastic processes X, Y as:

$$[X, Y]_t = \lim_{l(\mathcal{P}) \to 0} \sum_{j=0}^{n-1} \left(X_{t_{j+1}} - X_{t_j}\right)\left(Y_{t_{j+1}} - Y_{t_j}\right)$$

Properties of Quadratic Variation[7]

1. Linearity: For stochastic processes X, Y, Z:

$$[\alpha X + \beta Y, Z]_t = \alpha [X, Z]_t + \beta [Y, Z]_t$$

2. Polarisation identity:

$$[X, Y]_t = \frac{1}{2}\left([X + Y, X + Y]_t - [X, X]_t - [Y, Y]_t\right)$$

3. Jumps of the quadratic covariation process occur only at points where both processes have jumps:

$$\Delta[X, Y]_t = \Delta X_t \Delta Y_t$$

4. If one of the processes X or Y is of finite variation, then

$$\Delta[X, Y]_t = \sum_{s \leq t} \Delta X_s \Delta Y_s$$

[6] Just as X_t is shorthand notation for $X(t)$, $[X]_t$ will be used as shorthand for $[X](t)$.
[7] For more details, see Klebaner 2012, section 8.5.

5. If X is a continuous stochastic process and Y is a stochastic process with finite variation, then:

$$[X, Y]_t = 0$$

Bibliography

Acemoglu, D. (2008). *Introduction to Modern Economic Growth*. Princeton university press.

Achdou, Y., J. Han, J.-M. Lasry, P.-L. Lions, and B. Moll (2022). "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach". In: *The Review of Economic Studies* 89.1, pp. 45–86.

Barles, G. and P. E. Souganidis (1991). "Convergence of Approximation Schemes for Fully Nonlinear Second Order Equations". In: *Asymptotic Analysis* 4.3, pp. 271–283.

Hirsch, M. W., S. Smale, and R. L. Devaney (2004). *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press. 433 pp.

Jarrow, R. and P. Protter (2004). "A Short History of Stochastic Integration and Mathematical Finance: The Early Years, 1880-1970". In: *Lecture Notes-Monograph Series* 45, pp. 75–91.

Karatzas, I. and S. E. Shreve (1998). *Methods of Mathematical Finance*. Vol. 39. Springer.

— (1991). *Brownian Motion and Stochastic Calculus*. 2nd ed. Springer.

Klebaner, F. C. (2012). *Introduction to Stochastic Calculus with Applications*. 3rd Ed. London: Imperial College Press.

Kloeden, P. E. and E. Platen (1995). *Numerical Solution of Stochastic Differential Equations*. Springer.

Munk, C. (2015). *Financial Asset Pricing Theory*. Oxford University Press.

Øksendal, B. (2014). *Stochastic Differential Equations: An Introduction with Applications*. 6th Ed. Springer.

Ross, S. M. (2014). *Introduction to Probability Models*. 10th ed. Academic press.

Stewart, W. J. (2021). *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press.

Stokey, N. L., R. E. Lucas Jr., and E. C. Prescott (1989). *Recursive Methods in Economic Dynamics*. Harvard University Press.

Stokey, N. L. (2009). *The Economics of Inaction: Stochastic Control Models with Fixed Costs*. Princeton University Press.

Tourin, A. (2013). "Chapter 13 - Introduction to Finite Differences Methods". In: *Optimal Stochastic Control, Stochastic Target Problems, and Backward SDE*. Ed. by N. Touzi. New York, NY: Springer New York, pp. 201–212.