Part VII

LQ Control

CHAPTER

FORTYEIGHT

LQ CONTROL: FOUNDATIONS

Contents

In addition to what's in Anaconda, this lecture will need the following libraries:

```
!conda install -y quantecon
```

48.1 Overview

Linear quadratic (LQ) control refers to a class of dynamic optimization problems that have found applications in almost every scientific field.

This lecture provides an introduction to LQ control and its economic applications.

As we will see, LQ systems have a simple structure that makes them an excellent workhorse for a wide variety of economic problems.

Moreover, while the linear-quadratic structure is restrictive, it is in fact far more flexible than it may appear initially.

These themes appear repeatedly below.

Mathematically, LQ control problems are closely related to *the Kalman filter*

- Recursive formulations of linear-quadratic control problems and Kalman filtering problems both involve matrix **Riccati equations**.
- Classical formulations of linear control and linear filtering problems make use of similar matrix decompositions (see for example this lecture and this lecture).

In reading what follows, it will be useful to have some familiarity with

- matrix manipulations
- vectors of random variables
- dynamic programming and the Bellman equation (see for example *this lecture* and *this lecture*)

For additional reading on LQ control, see, for example,

- [LS18], chapter 5
- [HS08], chapter 4
- [HLL96], section 3.5

In order to focus on computation, we leave longer proofs to these sources (while trying to provide as much intuition as possible).

Let's start with some imports:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (11, 5)  #set default figure size
import numpy as np
from quantecon import LQ
```

48.2 Introduction

The "linear" part of LQ is a linear law of motion for the state, while the "quadratic" part refers to preferences.

Let's begin with the former, move on to the latter, and then put them together into an optimization problem.

48.2.1 The Law of Motion

Let x_t be a vector describing the state of some economic system.

Suppose that x_t follows a linear law of motion given by

$$x_{t+1} = Ax_t + Bu_t + Cw_{t+1}, \qquad t = 0, 1, 2, \ldots \tag{1}$$

Here

- u_t is a "control" vector, incorporating choices available to a decision-maker confronting the current state x_t
- $\{w_t\}$ is an uncorrelated zero mean shock process satisfying $\mathbb{E} w_t w_t' = I$, where the right-hand side is the identity matrix

Regarding the dimensions

- x_t is $n \times 1$, A is $n \times n$
- u_t is $k \times 1$, B is $n \times k$
- w_t is $j \times 1$, C is $n \times j$

Example 1

Consider a household budget constraint given by

$$a_{t+1} + c_t = (1 + r)a_t + y_t$$

Here a_t is assets, r is a fixed interest rate, c_t is current consumption, and y_t is current non-financial income.

If we suppose that $\{y_t\}$ is serially uncorrelated and $N(0, \sigma^2)$, then, taking $\{w_t\}$ to be standard normal, we can write the system as

$$a_{t+1} = (1 + r)a_t - c_t + \sigma w_{t+1}$$

This is clearly a special case of (1), with assets being the state and consumption being the control.

Example 2

One unrealistic feature of the previous model is that non-financial income has a zero mean and is often negative.

This can easily be overcome by adding a sufficiently large mean.

Hence in this example, we take $y_t = \sigma w_{t+1} + \mu$ for some positive real number μ.

Another alteration that's useful to introduce (we'll see why soon) is to change the control variable from consumption to the deviation of consumption from some "ideal" quantity $\bar{c}$.

(Most parameterizations will be such that $\bar{c}$ is large relative to the amount of consumption that is attainable in each period, and hence the household wants to increase consumption.)

For this reason, we now take our control to be $u_t := c_t - \bar{c}$.

In terms of these variables, the budget constraint $a_{t+1} = (1 + r)a_t - c_t + y_t$ becomes

$$a_{t+1} = (1 + r)a_t - u_t - \bar{c} + \sigma w_{t+1} + \mu \tag{2}$$

How can we write this new system in the form of equation (1)?

If, as in the previous example, we take a_t as the state, then we run into a problem: the law of motion contains some constant terms on the right-hand side.

This means that we are dealing with an *affine* function, not a linear one (recall *this discussion*).

Fortunately, we can easily circumvent this problem by adding an extra state variable.

In particular, if we write

$$\begin{pmatrix} a_{t+1} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 + r & -\bar{c} + \mu \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_t \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix} u_t + \begin{pmatrix} \sigma \\ 0 \end{pmatrix} w_{t+1} \tag{3}$$

then the first row is equivalent to (2).

Moreover, the model is now linear and can be written in the form of (1) by setting

$$x_t := \begin{pmatrix} a_t \\ 1 \end{pmatrix}, \quad A := \begin{pmatrix} 1 + r & -\bar{c} + \mu \\ 0 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad C := \begin{pmatrix} \sigma \\ 0 \end{pmatrix} \tag{4}$$

In effect, we've bought ourselves linearity by adding another state.

48.2.2 Preferences

In the LQ model, the aim is to minimize flow of losses, where time-t loss is given by the quadratic expression

$$x_t' R x_t + u_t' Q u_t \tag{5}$$

Here

- R is assumed to be $n \times n$, symmetric and nonnegative definite.
- Q is assumed to be $k \times k$, symmetric and positive definite.

Note: In fact, for many economic problems, the definiteness conditions on R and Q can be relaxed. It is sufficient that certain submatrices of R and Q be nonnegative definite. See [HS08] for details.

Example 1

A very simple example that satisfies these assumptions is to take R and Q to be identity matrices so that current loss is

$$x_t' I x_t + u_t' I u_t = \|x_t\|^2 + \|u_t\|^2$$

Thus, for both the state and the control, loss is measured as squared distance from the origin.

(In fact, the general case (5) can also be understood in this way, but with R and Q identifying other – non-Euclidean – notions of "distance" from the zero vector.)

Intuitively, we can often think of the state x_t as representing deviation from a target, such as

- deviation of inflation from some target level
- deviation of a firm's capital stock from some desired quantity

The aim is to put the state close to the target, while using controls parsimoniously.

Example 2

In the household problem *studied above*, setting $R = 0$ and $Q = 1$ yields preferences

$$x_t' R x_t + u_t' Q u_t = u_t^2 = (c_t - \bar{c})^2$$

Under this specification, the household's current loss is the squared deviation of consumption from the ideal level $\bar{c}$.

48.3 Optimality – Finite Horizon

Let's now be precise about the optimization problem we wish to consider, and look at how to solve it.

48.3.1 The Objective

We will begin with the finite horizon case, with terminal time $T \in \mathbb{N}$.

In this case, the aim is to choose a sequence of controls $\{u_0, \dots, u_{T-1}\}$ to minimize the objective

$$\mathbb{E}\left\{\sum_{t=0}^{T-1} \beta^t (x_t' R x_t + u_t' Q u_t) + \beta^T x_T' R_f x_T\right\} \tag{6}$$

subject to the law of motion (1) and initial state x_0.

The new objects introduced here are β and the matrix R_f.

The scalar β is the discount factor, while $x' R_f x$ gives terminal loss associated with state x.

Comments:

- We assume R_f to be $n \times n$, symmetric and nonnegative definite.
- We allow $\beta = 1$, and hence include the undiscounted case.
- x_0 may itself be random, in which case we require it to be independent of the shock sequence $w_1, \dots, w_T$.

48.3.2 Information

There's one constraint we've neglected to mention so far, which is that the decision-maker who solves this LQ problem knows only the present and the past, not the future.

To clarify this point, consider the sequence of controls $\{u_0, \dots, u_{T-1}\}$.

When choosing these controls, the decision-maker is permitted to take into account the effects of the shocks $\{w_1, \dots, w_T\}$ on the system.

However, it is typically assumed — and will be assumed here — that the time-t control u_t can be made with knowledge of past and present shocks only.

The fancy measure-theoretic way of saying this is that u_t must be measurable with respect to the σ-algebra generated by $x_0, w_1, w_2, \dots, w_t$.

This is in fact equivalent to stating that u_t can be written in the form $u_t = g_t(x_0, w_1, w_2, \dots, w_t)$ for some Borel measurable function g_t.

(Just about every function that's useful for applications is Borel measurable, so, for the purposes of intuition, you can read that last phrase as "for some function g_t")

Now note that x_t will ultimately depend on the realizations of $x_0, w_1, w_2, \dots, w_t$.

In fact, it turns out that x_t summarizes all the information about these historical shocks that the decision-maker needs to set controls optimally.

More precisely, it can be shown that any optimal control u_t can always be written as a function of the current state alone.

Hence in what follows we restrict attention to control policies (i.e., functions) of the form $u_t = g_t(x_t)$.

Actually, the preceding discussion applies to all standard dynamic programming problems.

What's special about the LQ case is that – as we shall soon see — the optimal u_t turns out to be a linear function of x_t.

48.3.3 Solution

To solve the finite horizon LQ problem we can use a dynamic programming strategy based on backward induction that is conceptually similar to the approach adopted in *this lecture*.

For reasons that will soon become clear, we first introduce the notation $J_T(x) = x' R_f x$.

Now consider the problem of the decision-maker in the second to last period.

In particular, let the time be $T-1$, and suppose that the state is x_{T-1}.

The decision-maker must trade-off current and (discounted) final losses, and hence solves

$$\min_u \{x'_{T-1} R x_{T-1} + u' Q u + \beta \, \mathbb{E} J_T(A x_{T-1} + B u + C w_T)\}$$

At this stage, it is convenient to define the function

$$J_{T-1}(x) = \min_u \{x' R x + u' Q u + \beta \, \mathbb{E} J_T(A x + B u + C w_T)\} \tag{7}$$

The function J_{T-1} will be called the $T-1$ value function, and $J_{T-1}(x)$ can be thought of as representing total "loss-to-go" from state x at time $T-1$ when the decision-maker behaves optimally.

Now let's step back to $T-2$.

For a decision-maker at $T-2$, the value $J_{T-1}(x)$ plays a role analogous to that played by the terminal loss $J_T(x) = x' R_f x$ for the decision-maker at $T-1$.

That is, $J_{T-1}(x)$ summarizes the future loss associated with moving to state x.

The decision-maker chooses her control u to trade off current loss against future loss, where

- the next period state is $x_{T-1} = A x_{T-2} + B u + C w_{T-1}$, and hence depends on the choice of current control.
- the "cost" of landing in state x_{T-1} is $J_{T-1}(x_{T-1})$.

Her problem is therefore

$$\min_u \{x'_{T-2} R x_{T-2} + u' Q u + \beta \, \mathbb{E} J_{T-1}(A x_{T-2} + B u + C w_{T-1})\}$$

Letting

$$J_{T-2}(x) = \min_u \{x' R x + u' Q u + \beta \, \mathbb{E} J_{T-1}(A x + B u + C w_{T-1})\}$$

the pattern for backward induction is now clear.

In particular, we define a sequence of value functions $\{J_0, \ldots, J_T\}$ via

$$J_{t-1}(x) = \min_u \{x' R x + u' Q u + \beta \, \mathbb{E} J_t(A x + B u + C w_t)\} \quad \text{and} \quad J_T(x) = x' R_f x$$

The first equality is the Bellman equation from dynamic programming theory specialized to the finite horizon LQ problem.

Now that we have $\{J_0, \ldots, J_T\}$, we can obtain the optimal controls.

As a first step, let's find out what the value functions look like.

It turns out that every J_t has the form $J_t(x) = x' P_t x + d_t$ where P_t is a $n \times n$ matrix and d_t is a constant.

We can show this by induction, starting from $P_T := R_f$ and $d_T = 0$.

Using this notation, (7) becomes

$$J_{T-1}(x) = \min_u \{x' R x + u' Q u + \beta \, \mathbb{E}(A x + B u + C w_T)' P_T (A x + B u + C w_T)\} \tag{8}$$

To obtain the minimizer, we can take the derivative of the r.h.s. with respect to u and set it equal to zero.

Applying the relevant rules of *matrix calculus*, this gives

$$u = -(Q + \beta B' P_T B)^{-1} \beta B' P_T A x \tag{9}$$

Plugging this back into (8) and rearranging yields

$$J_{T-1}(x) = x' P_{T-1} x + d_{T-1}$$

where

$$P_{T-1} = R - \beta^2 A' P_T B (Q + \beta B' P_T B)^{-1} B' P_T A + \beta A' P_T A \tag{10}$$

and

$$d_{T-1} := \beta \, \text{trace}(C' P_T C) \tag{11}$$

(The algebra is a good exercise — we'll leave it up to you.)

If we continue working backwards in this manner, it soon becomes clear that $J_t(x) = x' P_t x + d_t$ as claimed, where $\{P_t\}$ and $\{d_t\}$ satisfy the recursions

$$P_{t-1} = R - \beta^2 A' P_t B (Q + \beta B' P_t B)^{-1} B' P_t A + \beta A' P_t A \quad \text{with} \quad P_T = R_f \tag{12}$$

and

$$d_{t-1} = \beta (d_t + \text{trace}(C' P_t C)) \quad \text{with} \quad d_T = 0 \tag{13}$$

Recalling (9), the minimizers from these backward steps are

$$u_t = -F_t x_t \quad \text{where} \quad F_t := (Q + \beta B' P_{t+1} B)^{-1} \beta B' P_{t+1} A \tag{14}$$

These are the linear optimal control policies we *discussed above*.

In particular, the sequence of controls given by (14) and (1) solves our finite horizon LQ problem.

Rephrasing this more precisely, the sequence $u_0, \ldots, u_{T-1}$ given by

$$u_t = -F_t x_t \quad \text{with} \quad x_{t+1} = (A - BF_t) x_t + C w_{t+1} \tag{15}$$

for $t = 0, \ldots, T-1$ attains the minimum of (6) subject to our constraints.

48.4 Implementation

We will use code from lqcontrol.py in QuantEcon.py to solve finite and infinite horizon linear quadratic control problems.

In the module, the various updating, simulation and fixed point methods are wrapped in a class called `LQ`, which includes

- Instance data:
  - The required parameters Q, R, A, B and optional parameters C, β, T, R_f, N specifying a given LQ model
    - set T and R_f to `None` in the infinite horizon case
    - set `C = None` (or zero) in the deterministic case
  - the value function and policy data
    - d_t, P_t, F_t in the finite horizon case

    * d, P, F in the infinite horizon case

- Methods:
  - `update_values` — shifts d_t, P_t, F_t to their $t-1$ values via (12), (13) and (14)
  - `stationary_values` — computes P, d, F in the infinite horizon case
  - `compute_sequence` —- simulates the dynamics of x_t, u_t, w_t given x_0 and assuming standard normal shocks

48.4.1 An Application

Early Keynesian models assumed that households have a constant marginal propensity to consume from current income.

Data contradicted the constancy of the marginal propensity to consume.

In response, Milton Friedman, Franco Modigliani and others built models based on a consumer's preference for an intertemporally smooth consumption stream.

(See, for example, [Fri56] or [MB54].)

One property of those models is that households purchase and sell financial assets to make consumption streams smoother than income streams.

The household savings problem *outlined above* captures these ideas.

The optimization problem for the household is to choose a consumption sequence in order to minimize

$$\mathbb{E}\left\{\sum_{t=0}^{T-1}\beta^t(c_t-\bar{c})^2+\beta^T q a_T^2\right\} \tag{16}$$

subject to the sequence of budget constraints $a_{t+1}=(1+r)a_t-c_t+y_t,\ t\geq 0$.

Here q is a large positive constant, the role of which is to induce the consumer to target zero debt at the end of her life.

(Without such a constraint, the optimal choice is to choose $c_t=\bar{c}$ in each period, letting assets adjust accordingly.)

As before we set $y_t=\sigma w_{t+1}+\mu$ and $u_t:=c_t-\bar{c}$, after which the constraint can be written as in (2).

We saw how this constraint could be manipulated into the LQ formulation $x_{t+1}=Ax_t+Bu_t+Cw_{t+1}$ by setting $x_t=(a_t\ 1)'$ and using the definitions in (4).

To match with this state and control, the objective function (16) can be written in the form of (6) by choosing

$$Q:=1,\quad R:=\begin{pmatrix}0&0\\0&0\end{pmatrix},\quad \text{and}\quad R_f:=\begin{pmatrix}q&0\\0&0\end{pmatrix}$$

Now that the problem is expressed in LQ form, we can proceed to the solution by applying (12) and (14).

After generating shocks $w_1,\dots,w_T$, the dynamics for assets and consumption can be simulated via (15).

The following figure was computed using $r=0.05, \beta=1/(1+r), \bar{c}=2, \mu=1, \sigma=0.25, T=45$ and $q=10^6$.

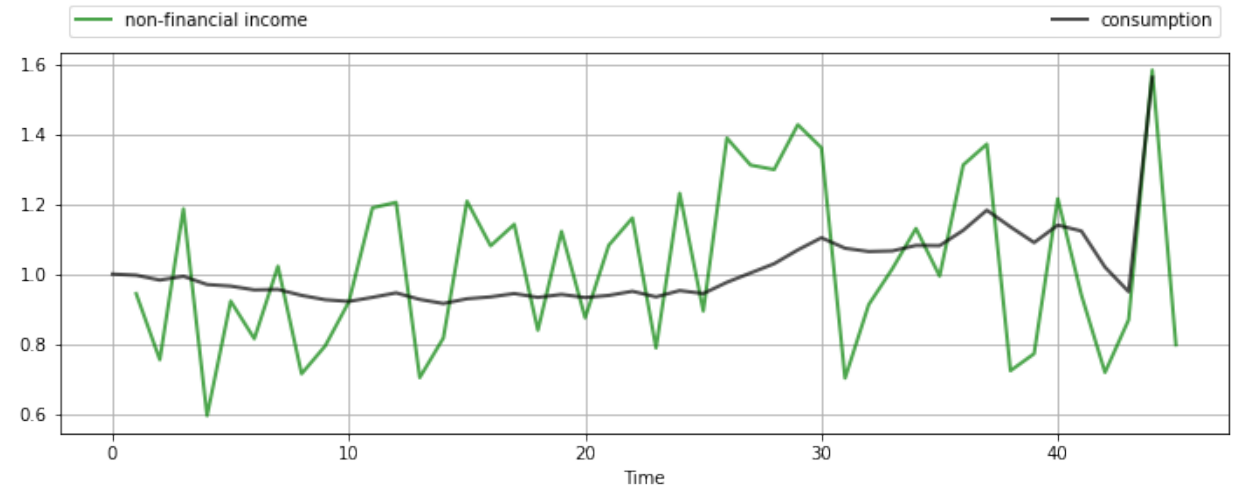
The shocks $\{w_t\}$ were taken to be IID and standard normal.

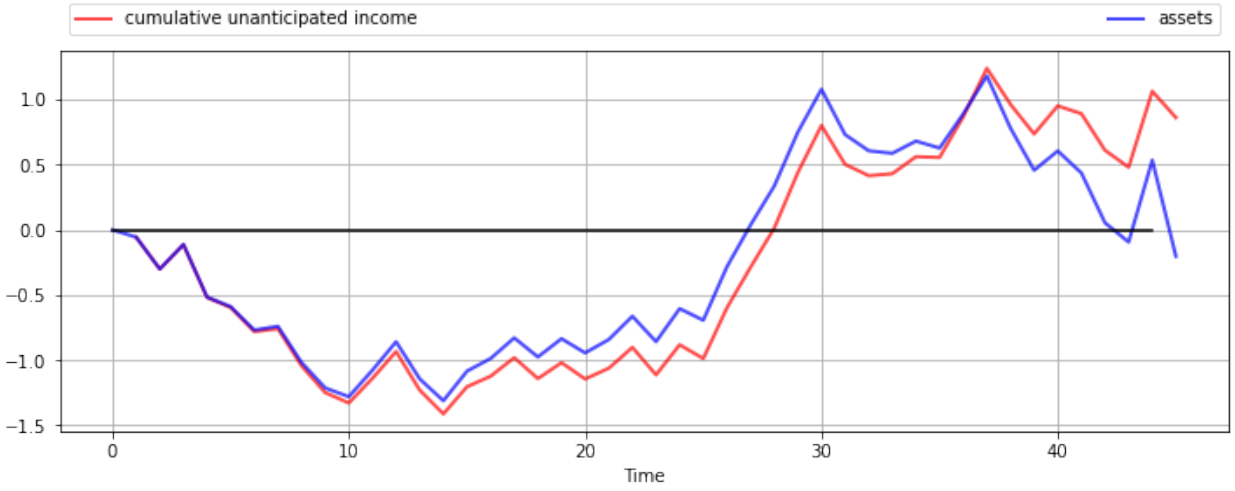
```
# Model parameters
r = 0.05
β = 1/(1 + r)
T = 45
c_bar = 2
σ = 0.25
```

(continues on next page)

(continued from previous page)

```
μ = 1
q = 1e6

# Formulate as an LQ problem
Q = 1
R = np.zeros((2, 2))
Rf = np.zeros((2, 2))
Rf[0, 0] = q
A = [[1 + r, -c_bar + μ],
     [0,               1]]
B = [[-1],
     [ 0]]
C = [[σ],
     [0]]

# Compute solutions and simulate
lq = LQ(Q, R, A, B, C, beta=β, T=T, Rf=Rf)
x0 = (0, 1)
xp, up, wp = lq.compute_sequence(x0)

# Convert back to assets, consumption and income
assets = xp[0, :]            # a_t
c = up.flatten() + c_bar     # c_t
income = σ * wp[0, 1:] + μ   # y_t

# Plot results
n_rows = 2
fig, axes = plt.subplots(n_rows, 1, figsize=(12, 10))

plt.subplots_adjust(hspace=0.5)

bbox = (0., 1.02, 1., .102)
legend_args = {'bbox_to_anchor': bbox, 'loc': 3, 'mode': 'expand'}
p_args = {'lw': 2, 'alpha': 0.7}

axes[0].plot(list(range(1, T+1)), income, 'g-', label="non-financial income",
             **p_args)
axes[0].plot(list(range(T)), c, 'k-', label="consumption", **p_args)

axes[1].plot(list(range(1, T+1)), np.cumsum(income - μ), 'r-',
             label="cumulative unanticipated income", **p_args)
axes[1].plot(list(range(T+1)), assets, 'b-', label="assets", **p_args)
axes[1].plot(list(range(T)), np.zeros(T), 'k-')

for ax in axes:
    ax.grid()
    ax.set_xlabel('Time')
    ax.legend(ncol=2, **legend_args)

plt.show()
```





The top panel shows the time path of consumption c_t and income y_t in the simulation.

As anticipated by the discussion on consumption smoothing, the time path of consumption is much smoother than that for income.

(But note that consumption becomes more irregular towards the end of life, when the zero final asset requirement impinges more on consumption choices.)

The second panel in the figure shows that the time path of assets a_t is closely correlated with cumulative unanticipated income, where the latter is defined as

$$z_t := \sum_{j=0}^{t} \sigma w_t$$

A key message is that unanticipated windfall gains are saved rather than consumed, while unanticipated negative shocks are met by reducing assets.

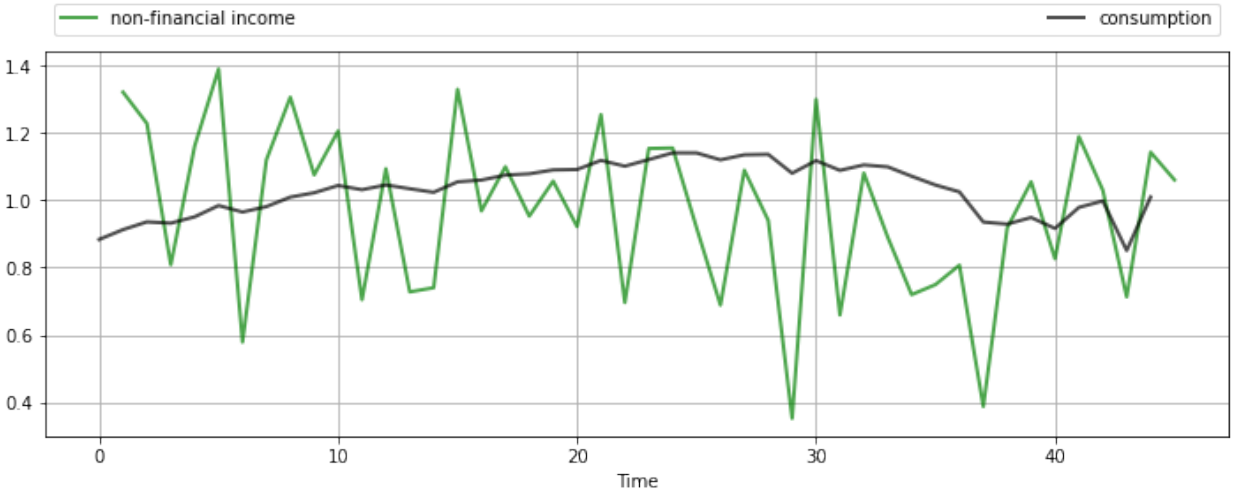
(Again, this relationship breaks down towards the end of life due to the zero final asset requirement.)

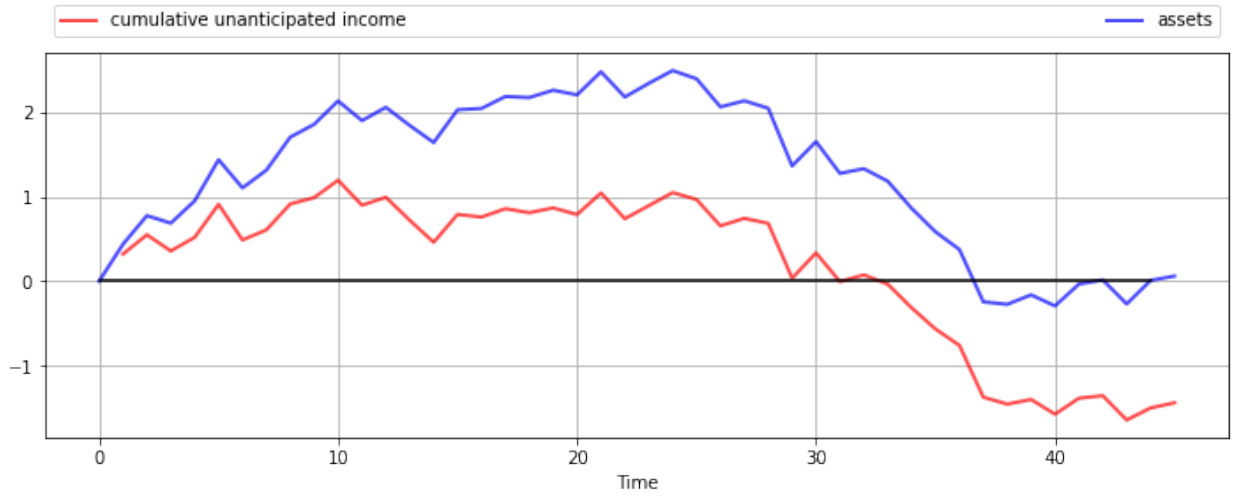
These results are relatively robust to changes in parameters.

For example, let's increase β from $1/(1+r) \approx 0.952$ to 0.96 while keeping other parameters fixed.

This consumer is slightly more patient than the last one, and hence puts relatively more weight on later consumption values.

```
# Compute solutions and simulate
lq = LQ(Q, R, A, B, C, beta=0.96, T=T, Rf=Rf)
x0 = (0, 1)
xp, up, wp = lq.compute_sequence(x0)

# Convert back to assets, consumption and income
assets = xp[0, :]            # a_t
c = up.flatten() + c_bar     # c_t
income = σ * wp[0, 1:] + μ   # y_t

# Plot results
n_rows = 2
fig, axes = plt.subplots(n_rows, 1, figsize=(12, 10))

plt.subplots_adjust(hspace=0.5)

bbox = (0., 1.02, 1., .102)
legend_args = {'bbox_to_anchor': bbox, 'loc': 3, 'mode': 'expand'}
p_args = {'lw': 2, 'alpha': 0.7}

axes[0].plot(list(range(1, T+1)), income, 'g-', label="non-financial income",
             **p_args)
axes[0].plot(list(range(T)), c, 'k-', label="consumption", **p_args)

axes[1].plot(list(range(1, T+1)), np.cumsum(income - μ), 'r-',
             label="cumulative unanticipated income", **p_args)
axes[1].plot(list(range(T+1)), assets, 'b-', label="assets", **p_args)
axes[1].plot(list(range(T)), np.zeros(T), 'k-')

for ax in axes:
    ax.grid()
    ax.set_xlabel('Time')
    ax.legend(ncol=2, **legend_args)

plt.show()
```





We now have a slowly rising consumption stream and a hump-shaped build-up of assets in the middle periods to fund rising consumption.

However, the essential features are the same: consumption is smooth relative to income, and assets are strongly positively correlated with cumulative unanticipated income.

48.5 Extensions and Comments

Let's now consider a number of standard extensions to the LQ problem treated above.

48.5.1 Time-Varying Parameters

In some settings, it can be desirable to allow A, B, C, R and Q to depend on t.

For the sake of simplicity, we've chosen not to treat this extension in our implementation given below.

However, the loss of generality is not as large as you might first imagine.

In fact, we can tackle many models with time-varying parameters by suitable choice of state variables.

One illustration is given *below*.

For further examples and a more systematic treatment, see [HS13], section 2.4.

48.5.2 Adding a Cross-Product Term

In some LQ problems, preferences include a cross-product term $u_t' N x_t$, so that the objective function becomes

$$\mathbb{E} \left\{ \sum_{t=0}^{T-1} \beta^t (x_t' R x_t + u_t' Q u_t + 2 u_t' N x_t) + \beta^T x_T' R_f x_T \right\} \tag{17}$$

Our results extend to this case in a straightforward way.

The sequence $\{P_t\}$ from (12) becomes

$$P_{t-1} = R - (\beta B' P_t A + N)'(Q + \beta B' P_t B)^{-1}(\beta B' P_t A + N) + \beta A' P_t A \quad \text{with} \quad P_T = R_f \tag{18}$$

The policies in (14) are modified to

$$u_t = -F_t x_t \quad \text{where} \quad F_t := (Q + \beta B' P_{t+1} B)^{-1}(\beta B' P_{t+1} A + N) \tag{19}$$

The sequence $\{d_t\}$ is unchanged from (13).

We leave interested readers to confirm these results (the calculations are long but not overly difficult).

48.5.3 Infinite Horizon

Finally, we consider the infinite horizon case, with *cross-product term*, unchanged dynamics and objective function given by

$$\mathbb{E} \left\{ \sum_{t=0}^{\infty} \beta^t (x_t' R x_t + u_t' Q u_t + 2 u_t' N x_t) \right\} \tag{20}$$

In the infinite horizon case, optimal policies can depend on time only if time itself is a component of the state vector x_t.

In other words, there exists a fixed matrix F such that $u_t = -F x_t$ for all t.

That decision rules are constant over time is intuitive — after all, the decision-maker faces the same infinite horizon at every stage, with only the current state changing.

Not surprisingly, P and d are also constant.

The stationary matrix P is the solution to the discrete-time algebraic Riccati equation.

$$P = R - (\beta B'PA + N)'(Q + \beta B'PB)^{-1}(\beta B'PA + N) + \beta A'PA \tag{21}$$

Equation (21) is also called the *LQ Bellman equation*, and the map that sends a given P into the right-hand side of (21) is called the *LQ Bellman operator*.

The stationary optimal policy for this model is

$$u = -Fx \quad \text{where} \quad F = (Q + \beta B'PB)^{-1}(\beta B'PA + N) \tag{22}$$

The sequence $\{d_t\}$ from (13) is replaced by the constant value

$$d := \text{trace}(C'PC)\frac{\beta}{1-\beta} \tag{23}$$

The state evolves according to the time-homogeneous process $x_{t+1} = (A - BF)x_t + Cw_{t+1}$.

An example infinite horizon problem is treated *below*.

48.5.4 Certainty Equivalence

Linear quadratic control problems of the class discussed above have the property of *certainty equivalence*.

By this, we mean that the optimal policy F is not affected by the parameters in C, which specify the shock process.

This can be confirmed by inspecting (22) or (19).

It follows that we can ignore uncertainty when solving for optimal behavior, and plug it back in when examining optimal state dynamics.

48.6 Further Applications

48.6.1 Application 1: Age-Dependent Income Process

Previously we studied a permanent income model that generated consumption smoothing.

One unrealistic feature of that model is the assumption that the mean of the random income process does not depend on the consumer's age.

A more realistic income profile is one that rises in early working life, peaks towards the middle and maybe declines toward the end of working life and falls more during retirement.

In this section, we will model this rise and fall as a symmetric inverted "U" using a polynomial in age.

As before, the consumer seeks to minimize

$$\mathbb{E}\left\{\sum_{t=0}^{T-1}\beta^t(c_t - \bar{c})^2 + \beta^T q a_T^2\right\} \tag{24}$$

subject to $a_{t+1} = (1+r)a_t - c_t + y_t,\ t \geq 0$.

For income we now take $y_t = p(t) + \sigma w_{t+1}$ where $p(t) := m_0 + m_1 t + m_2 t^2$.

(In *the next section* we employ some tricks to implement a more sophisticated model.)

The coefficients m_0, m_1, m_2 are chosen such that $p(0) = 0, p(T/2) = \mu$, and $p(T) = 0$.

You can confirm that the specification $m_0 = 0, m_1 = T\mu/(T/2)^2, m_2 = -\mu/(T/2)^2$ satisfies these constraints.

To put this into an LQ setting, consider the budget constraint, which becomes

$$a_{t+1} = (1 + r)a_t - u_t - \bar{c} + m_1 t + m_2 t^2 + \sigma w_{t+1} \tag{25}$$

The fact that a_{t+1} is a linear function of $(a_t, 1, t, t^2)$ suggests taking these four variables as the state vector x_t.

Once a good choice of state and control (recall $u_t = c_t - \bar{c}$) has been made, the remaining specifications fall into place relatively easily.

Thus, for the dynamics we set

$$x_t := \begin{pmatrix} a_t \\ 1 \\ t \\ t^2 \end{pmatrix}, \quad A := \begin{pmatrix} 1+r & -\bar{c} & m_1 & m_2 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 \end{pmatrix}, \quad B := \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad C := \begin{pmatrix} \sigma \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{26}$$

If you expand the expression $x_{t+1} = Ax_t + Bu_t + Cw_{t+1}$ using this specification, you will find that assets follow (25) as desired and that the other state variables also update appropriately.

To implement preference specification (24) we take

$$Q := 1, \quad R := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad R_f := \begin{pmatrix} q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{27}$$

The next figure shows a simulation of consumption and assets computed using the `compute_sequence` method of `lqcontrol.py` with initial assets set to zero.

Once again, smooth consumption is a dominant feature of the sample paths.

The asset path exhibits dynamics consistent with standard life cycle theory.

Exercise 1 gives the full set of parameters used here and asks you to replicate the figure.

48.6.2 Application 2: A Permanent Income Model with Retirement

In the *previous application*, we generated income dynamics with an inverted U shape using polynomials and placed them in an LQ framework.

It is arguably the case that this income process still contains unrealistic features.

A more common earning profile is where

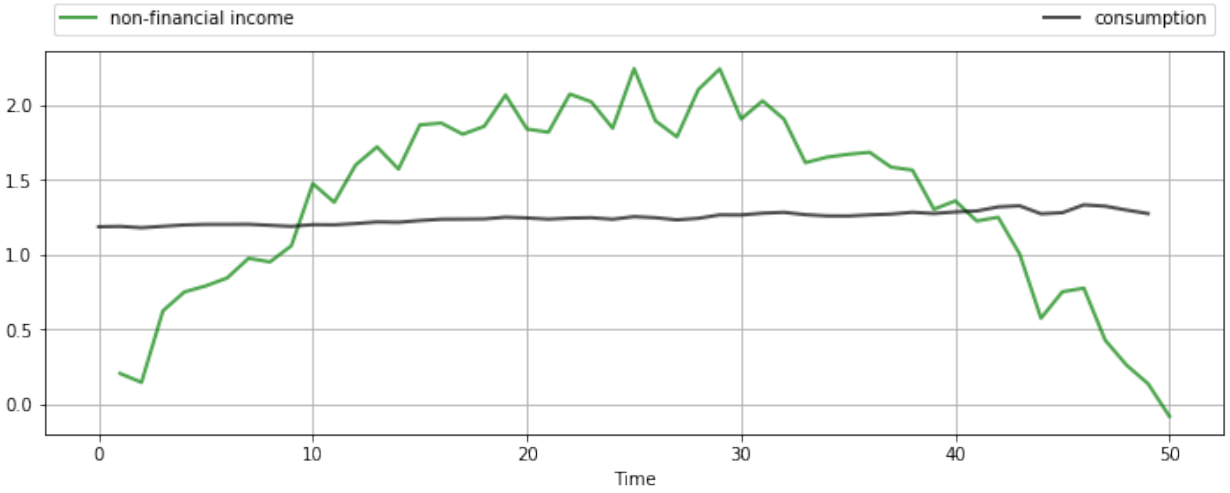
1. income grows over working life, fluctuating around an increasing trend, with growth flattening off in later years
2. retirement follows, with lower but relatively stable (non-financial) income

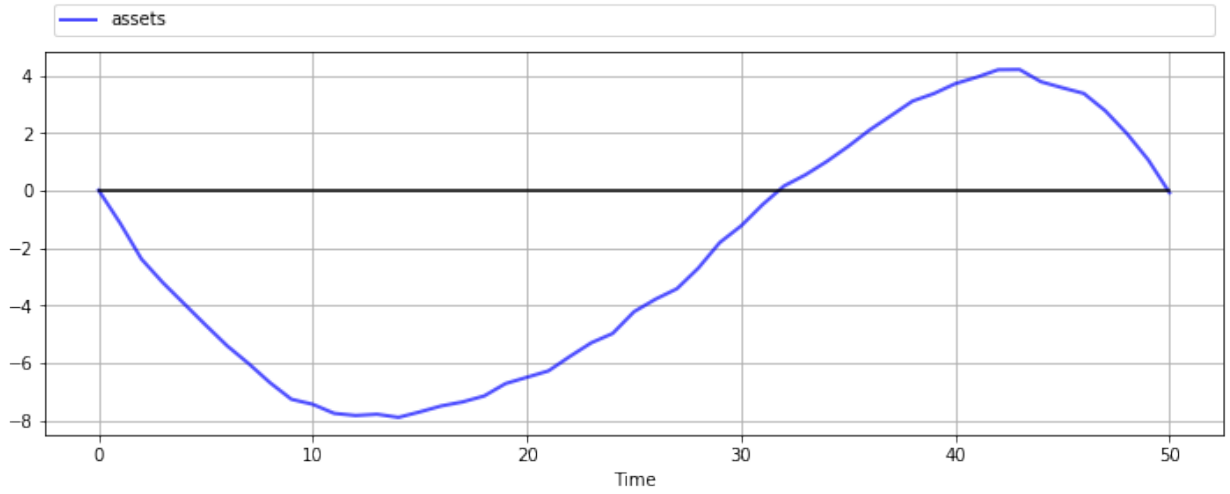
Letting K be the retirement date, we can express these income dynamics by

$$y_t = \begin{cases} p(t) + \sigma w_{t+1} & \text{if } t \leq K \\ s & \text{otherwise} \end{cases} \tag{28}$$

Here

- $p(t) := m_1 t + m_2 t^2$ with the coefficients m_1, m_2 chosen such that $p(K) = \mu$ and $p(0) = p(2K) = 0$
- s is retirement income


non-financial income
consumption
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
Time


assets
4
2
0
−2
−4
−6
−8
0
10
20
30
40
50
Time

We suppose that preferences are unchanged and given by (16).

The budget constraint is also unchanged and given by $a_{t+1} = (1+r)a_t - c_t + y_t$.

Our aim is to solve this problem and simulate paths using the LQ techniques described in this lecture.

In fact, this is a nontrivial problem, as the kink in the dynamics (28) at K makes it very difficult to express the law of motion as a fixed-coefficient linear system.

However, we can still use our LQ methods here by suitably linking two-component LQ problems.

These two LQ problems describe the consumer's behavior during her working life (`lq_working`) and retirement (`lq_retired`).

(This is possible because, in the two separate periods of life, the respective income processes [polynomial trend and constant] each fit the LQ framework.)

The basic idea is that although the whole problem is not a single time-invariant LQ problem, it is still a dynamic programming problem, and hence we can use appropriate Bellman equations at every stage.

Based on this logic, we can

1. solve `lq_retired` by the usual backward induction procedure, iterating back to the start of retirement.
2. take the start-of-retirement value function generated by this process, and use it as the terminal condition R_f to feed into the `lq_working` specification.
3. solve `lq_working` by backward induction from this choice of R_f, iterating back to the start of working life.

This process gives the entire life-time sequence of value functions and optimal policies.

The next figure shows one simulation based on this procedure.

The full set of parameters used in the simulation is discussed in *Exercise 2*, where you are asked to replicate the figure.

Once again, the dominant feature observable in the simulation is consumption smoothing.

The asset path fits well with standard life cycle theory, with dissaving early in life followed by later saving.

Assets peak at retirement and subsequently decline.

48.6.3 Application 3: Monopoly with Adjustment Costs

Consider a monopolist facing stochastic inverse demand function

$$p_t = a_0 - a_1 q_t + d_t$$

Here q_t is output, and the demand shock d_t follows

$$d_{t+1} = \rho d_t + \sigma w_{t+1}$$

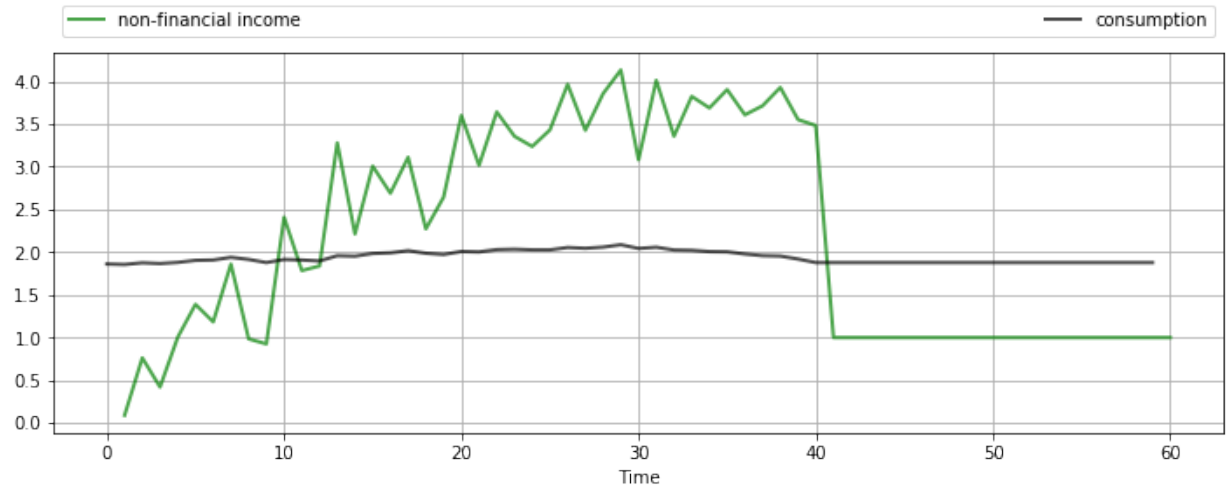
where $\{w_t\}$ is IID and standard normal.

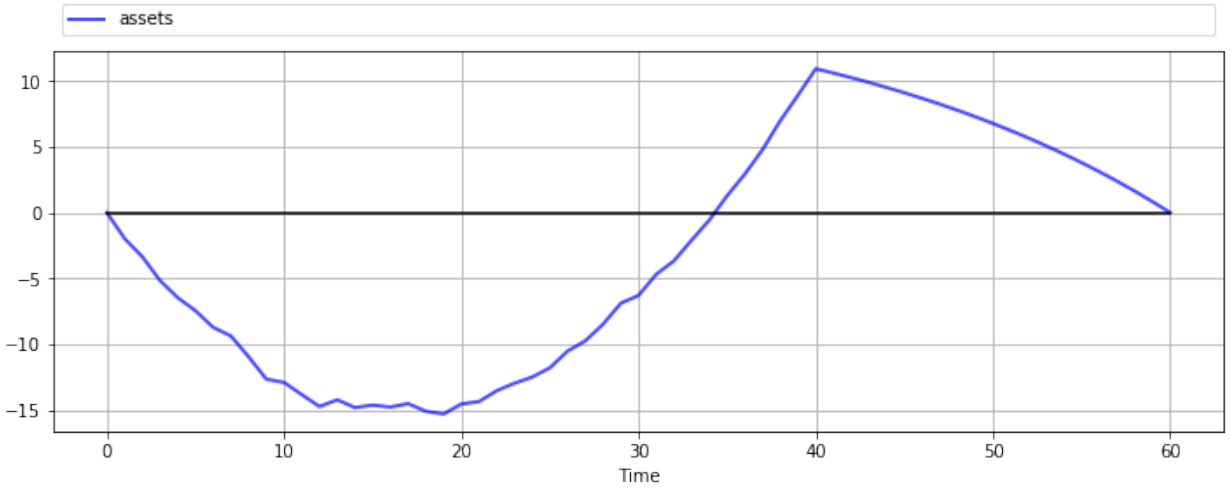
The monopolist maximizes the expected discounted sum of present and future profits

$$\mathbb{E}\left\{\sum_{t=0}^{\infty} \beta^t \pi_t\right\} \quad \text{where} \quad \pi_t := p_t q_t - c q_t - \gamma (q_{t+1} - q_t)^2 \tag{29}$$

Here

- $\gamma (q_{t+1} - q_t)^2$ represents adjustment costs
- c is average cost of production


non-financial income
consumption
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
Time


assets
10
5
0
−5
−10
−15
0
10
20
30
40
50
60
Time

This can be formulated as an LQ problem and then solved and simulated, but first let's study the problem and try to get some intuition.

One way to start thinking about the problem is to consider what would happen if $\gamma = 0$.

Without adjustment costs there is no intertemporal trade-off, so the monopolist will choose output to maximize current profit in each period.

It's not difficult to show that profit-maximizing output is

$$\bar{q}_t := \frac{a_0 - c + d_t}{2a_1}$$

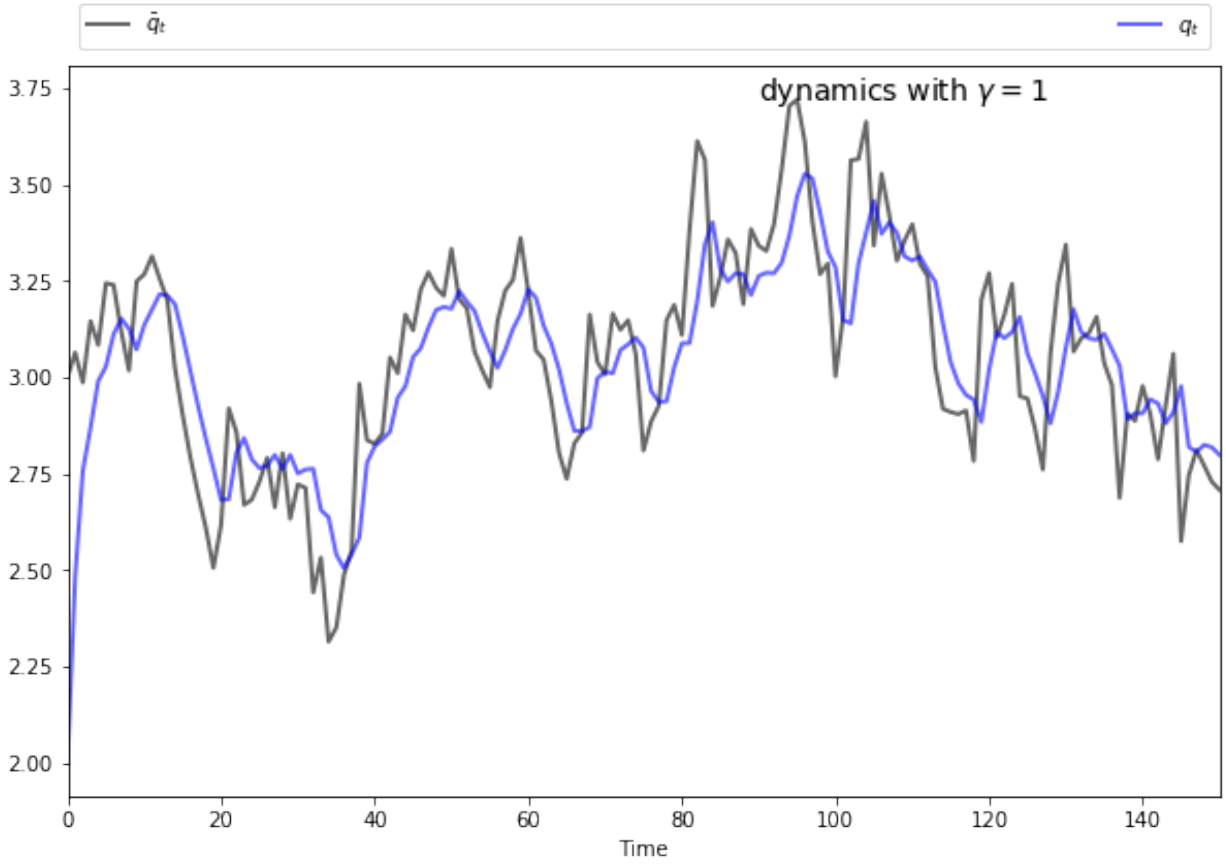
In light of this discussion, what we might expect for general γ is that

- if γ is close to zero, then q_t will track the time path of $\bar{q}_t$ relatively closely.
- if γ is larger, then q_t will be smoother than $\bar{q}_t$, as the monopolist seeks to avoid adjustment costs.

This intuition turns out to be correct.

The following figures show simulations produced by solving the corresponding LQ problem.

The only difference in parameters across the figures is the size of γ

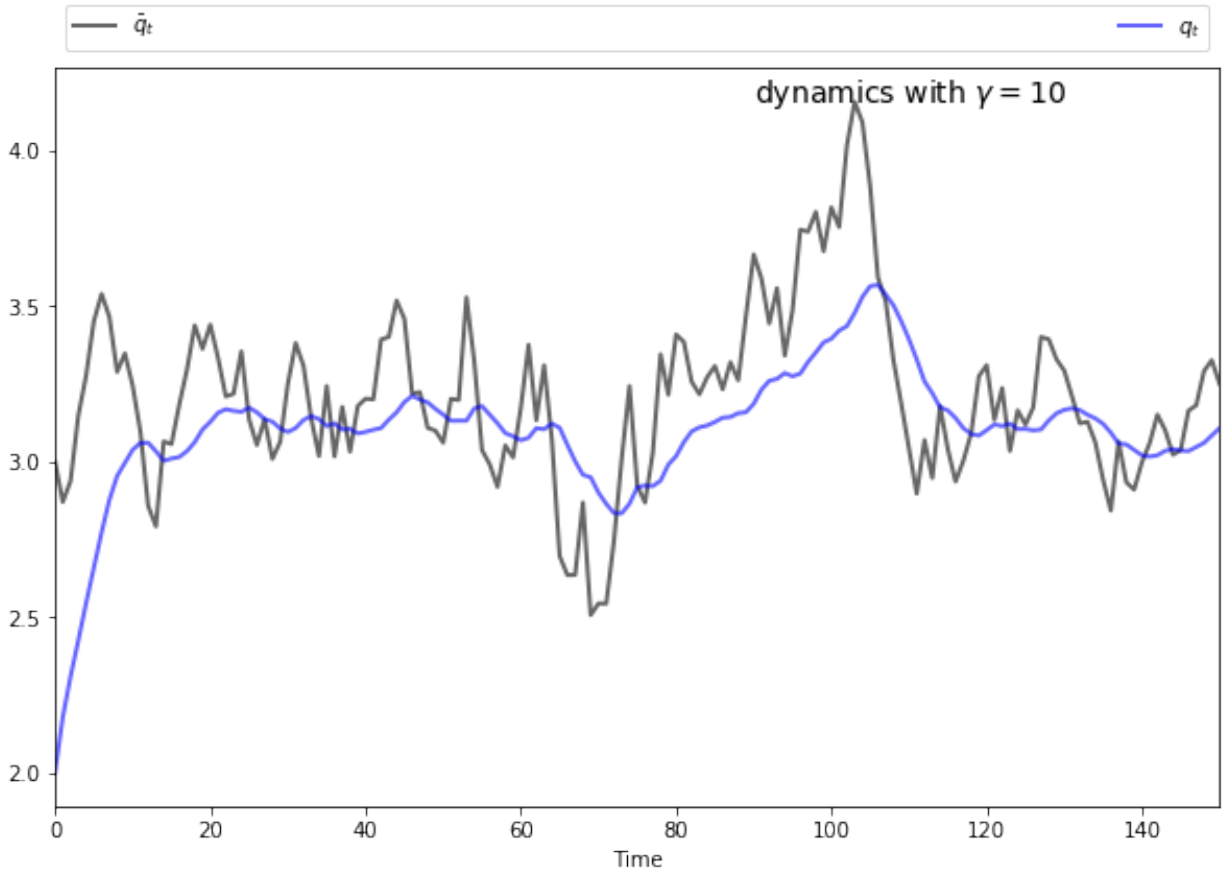


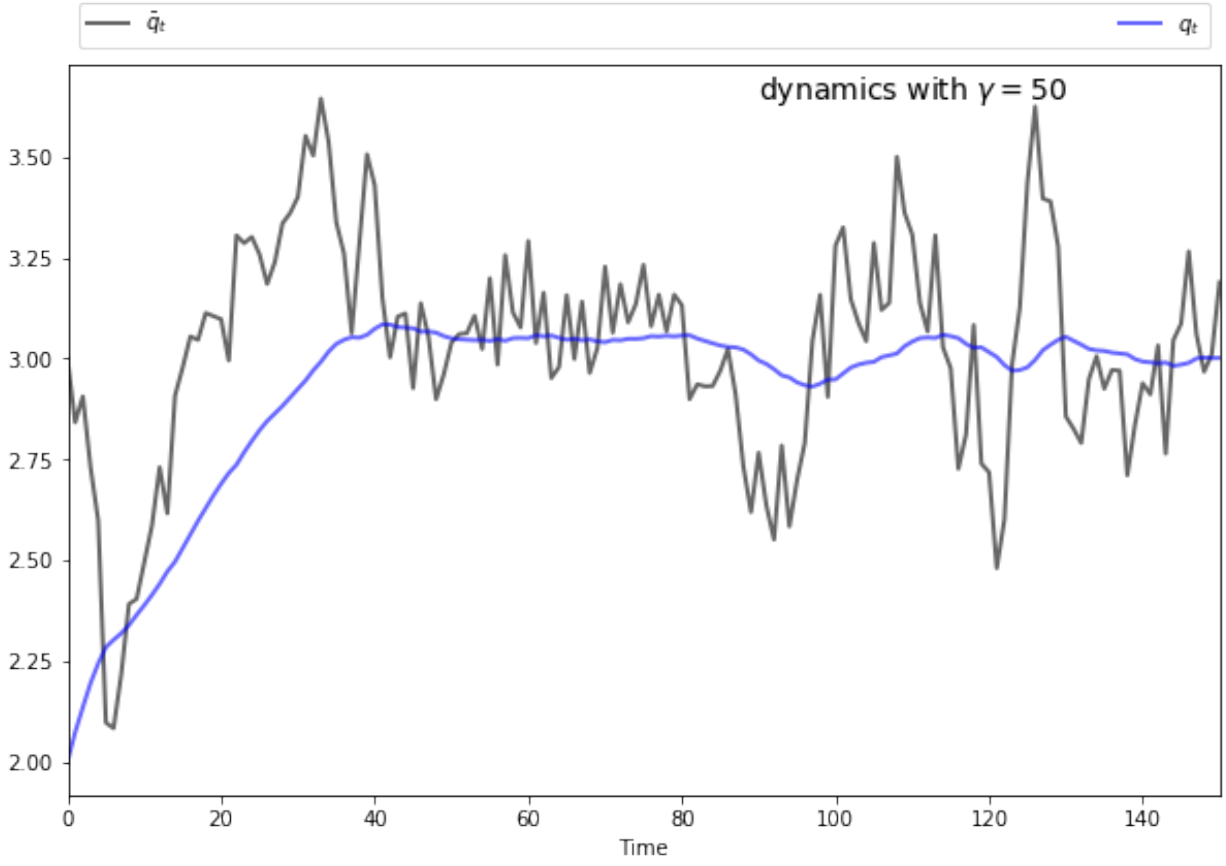
To produce these figures we converted the monopolist problem into an LQ problem.

The key to this conversion is to choose the right state — which can be a bit of an art.

Here we take $x_t = (\bar{q}_t \;\; q_t \;\; 1)'$, while the control is chosen as $u_t = q_{t+1} - q_t$.

We also manipulated the profit function slightly.


$\tilde{q}_t$
q_t
dynamics with $\gamma = 10$
4.0
3.5
3.0
2.5
2.0
0
20
40
60
80
100
120
140
Time


$\bar{q}_t$
q_t
dynamics with $\gamma = 50$
3.50
3.25
3.00
2.75
2.50
2.25
2.00
0
20
40
60
80
100
120
140
Time

In (29), current profits are $\pi_t := p_t q_t - c q_t - \gamma (q_{t+1} - q_t)^2$.

Let's now replace π_t in (29) with $\hat{\pi}_t := \pi_t - a_1 \bar{q}_t^2$.

This makes no difference to the solution, since $a_1 \bar{q}_t^2$ does not depend on the controls.

(In fact, we are just adding a constant term to (29), and optimizers are not affected by constant terms.)

The reason for making this substitution is that, as you will be able to verify, $\hat{\pi}_t$ reduces to the simple quadratic

$$\hat{\pi}_t = -a_1 (q_t - \bar{q}_t)^2 - \gamma u_t^2$$

After negation to convert to a minimization problem, the objective becomes

$$\min \mathbb{E} \sum_{t=0}^{\infty} \beta^t \left\{ a_1 (q_t - \bar{q}_t)^2 + \gamma u_t^2 \right\} \tag{30}$$

It's now relatively straightforward to find R and Q such that (30) can be written as (20).

Furthermore, the matrices A, B and C from (1) can be found by writing down the dynamics of each element of the state.

Exercise 3 asks you to complete this process, and reproduce the preceding figures.

48.7 Exercises

48.7.1 Exercise 1

Replicate the figure with polynomial income *shown above*.

The parameters are $r = 0.05, \beta = 1/(1+r), \bar{c} = 1.5, \mu = 2, \sigma = 0.15, T = 50$ and $q = 10^4$.

48.7.2 Exercise 2

Replicate the figure on work and retirement *shown above*.

The parameters are $r = 0.05, \beta = 1/(1+r), \bar{c} = 4, \mu = 4, \sigma = 0.35, K = 40, T = 60, s = 1$ and $q = 10^4$.

To understand the overall procedure, carefully read the section containing that figure.

Some hints are as follows:

First, in order to make our approach work, we must ensure that both LQ problems have the same state variables and control.

As with previous applications, the control can be set to $u_t = c_t - \bar{c}$.

For `lq_working`, x_t, A, B, C can be chosen as in (26).

- Recall that m_1, m_2 are chosen so that $p(K) = \mu$ and $p(2K) = 0$.

For `lq_retired`, use the same definition of x_t and u_t, but modify A, B, C to correspond to constant income $y_t = s$.

For `lq_retired`, set preferences as in (27).

For `lq_working`, preferences are the same, except that R_f should be replaced by the final value function that emerges from iterating `lq_retired` back to the start of retirement.

With some careful footwork, the simulation can be generated by patching together the simulations from these two separate models.

48.7.3 Exercise 3

Reproduce the figures from the monopolist application *given above*.

For parameters, use $a_0 = 5, a_1 = 0.5, \sigma = 0.15, \rho = 0.9, \beta = 0.95$ and $c = 2$, while γ varies between 1 and 50 (see figures).

48.8 Solutions

48.8.1 Exercise 1

Here's one solution.

We use some fancy plot commands to get a certain style — feel free to use simpler ones.

The model is an LQ permanent income / life-cycle model with hump-shaped income

$$y_t = m_1 t + m_2 t^2 + \sigma w_{t+1}$$

where $\{w_t\}$ is IID $N(0, 1)$ and the coefficients m_1 and m_2 are chosen so that $p(t) = m_1 t + m_2 t^2$ has an inverted U shape with

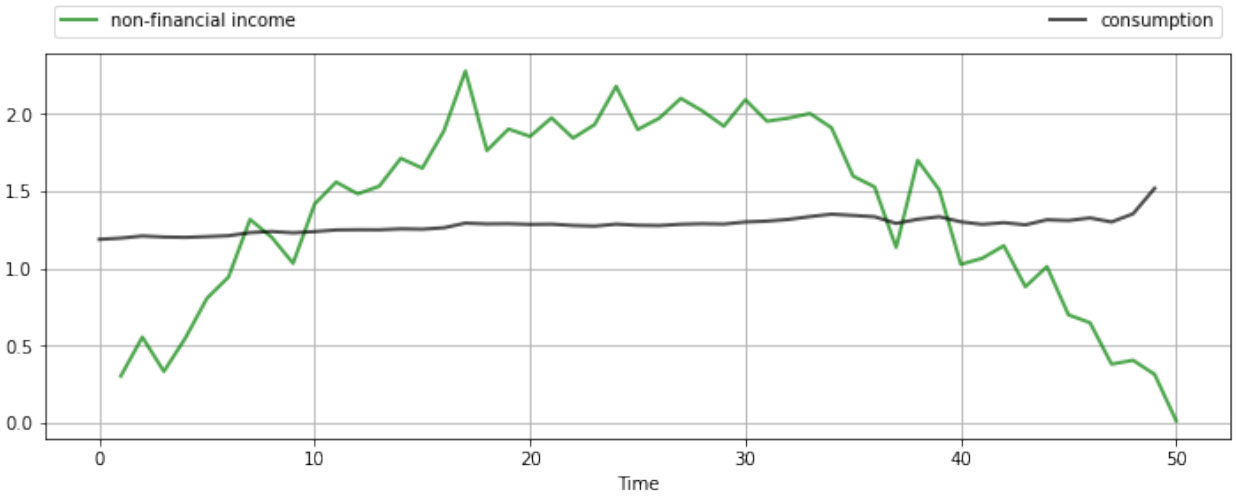
- $p(0) = 0, p(T/2) = \mu$, and
- $p(T) = 0$

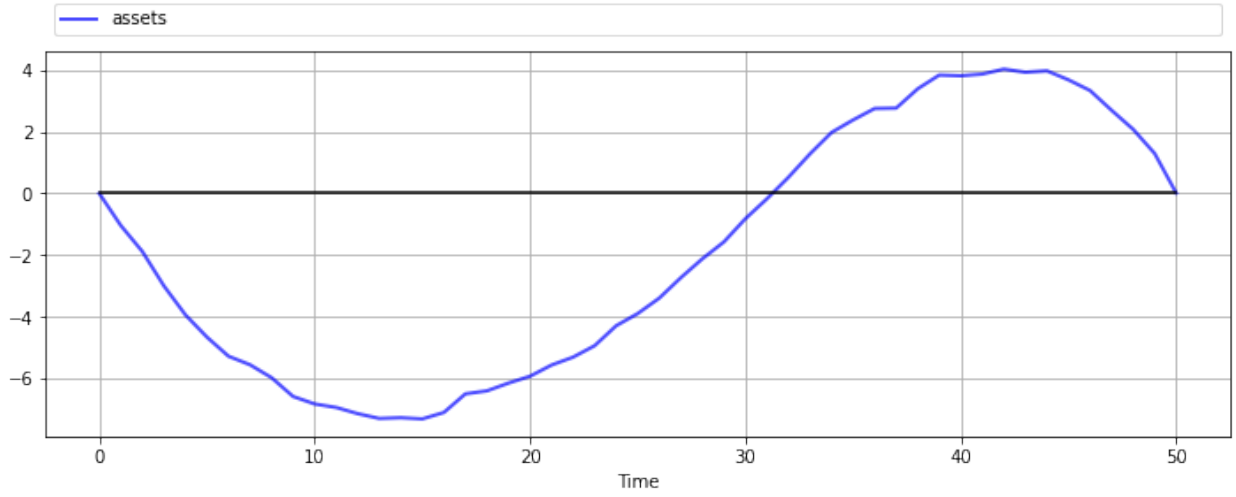
```
# Model parameters
r = 0.05
β = 1/(1 + r)
T = 50
c_bar = 1.5
σ = 0.15
μ = 2
q = 1e4
m1 = T * (μ/(T/2)**2)
m2 = -(μ/(T/2)**2)

# Formulate as an LQ problem
Q = 1
R = np.zeros((4, 4))
Rf = np.zeros((4, 4))
Rf[0, 0] = q
A = [[1 + r, -c_bar, m1, m2],
     [0,           1,  0,  0],
     [0,           1,  1,  0],
     [0,           1,  2,  1]]
B = [[-1],
     [ 0],
     [ 0],
     [ 0]]
C = [[σ],
     [0],
     [0],
     [0]]

# Compute solutions and simulate
```

(continues on next page)

(continued from previous page)

```
lq = LQ(Q, R, A, B, C, beta=β, T=T, Rf=Rf)
x0 = (0, 1, 0, 0)
xp, up, wp = lq.compute_sequence(x0)

# Convert results back to assets, consumption and income
ap = xp[0, :]               # Assets
c = up.flatten() + c_bar    # Consumption
time = np.arange(1, T+1)
income = σ * wp[0, 1:] + m1 * time + m2 * time**2  # Income


# Plot results
n_rows = 2
fig, axes = plt.subplots(n_rows, 1, figsize=(12, 10))

plt.subplots_adjust(hspace=0.5)

bbox = (0., 1.02, 1., .102)
legend_args = {'bbox_to_anchor': bbox, 'loc': 3, 'mode': 'expand'}
p_args = {'lw': 2, 'alpha': 0.7}

axes[0].plot(range(1, T+1), income, 'g-', label="non-financial income",
             **p_args)
axes[0].plot(range(T), c, 'k-', label="consumption", **p_args)

axes[1].plot(range(T+1), ap.flatten(), 'b-', label="assets", **p_args)
axes[1].plot(range(T+1), np.zeros(T+1), 'k-')

for ax in axes:
    ax.grid()
    ax.set_xlabel('Time')
    ax.legend(ncol=2, **legend_args)

plt.show()
```





48.8.2 Exercise 2

This is a permanent income / life-cycle model with polynomial growth in income over working life followed by a fixed retirement income.

The model is solved by combining two LQ programming problems as described in the lecture.

```
# Model parameters
r = 0.05
β = 1/(1 + r)
T = 60
K = 40
c_bar = 4
σ = 0.35
μ = 4
q = 1e4
s = 1
m1 = 2 * μ/K
m2 = -μ/K**2
```

(continues on next page)

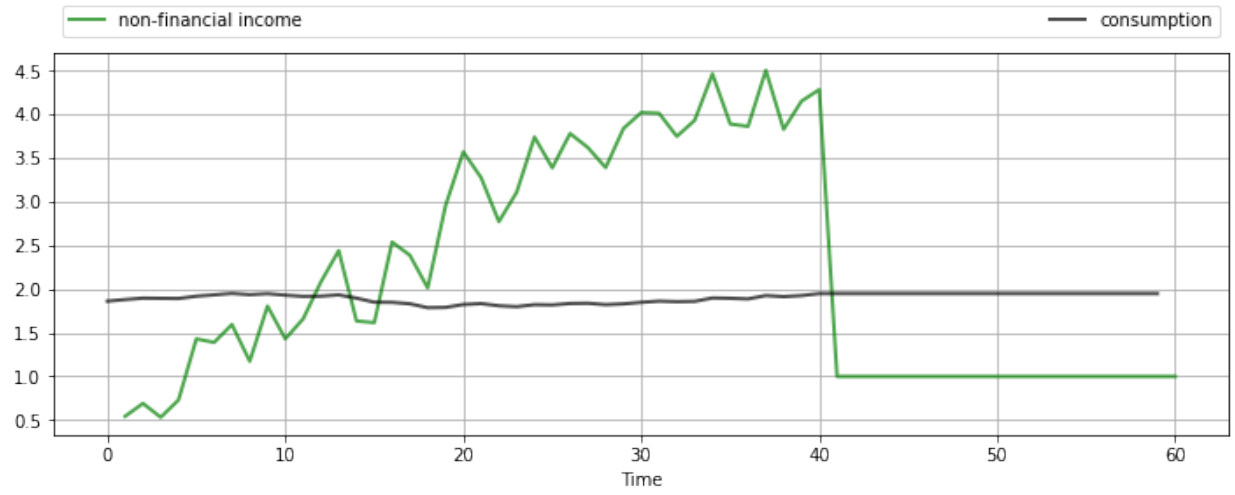
(continued from previous page)

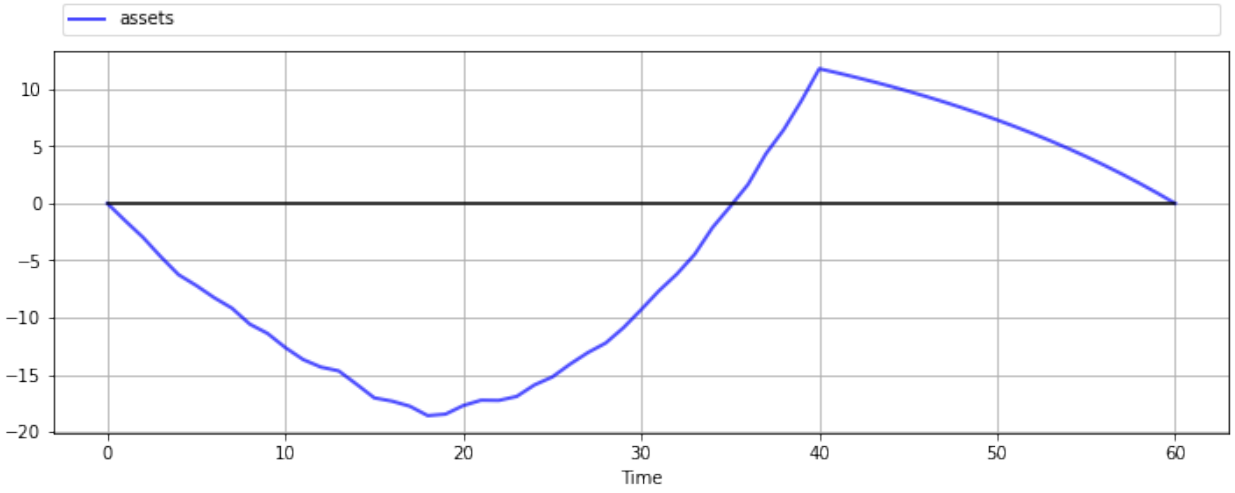
```
# Formulate LQ problem 1 (retirement)
Q = 1
R = np.zeros((4, 4))
Rf = np.zeros((4, 4))
Rf[0, 0] = q
A = [[1 + r, s - c_bar, 0, 0],
     [0,             1, 0, 0],
     [0,             1, 1, 0],
     [0,             1, 2, 1]]
B = [[-1],
     [ 0],
     [ 0],
     [ 0]]
C = [[0],
     [0],
     [0],
     [0]]

# Initialize LQ instance for retired agent
lq_retired = LQ(Q, R, A, B, C, beta=β, T=T-K, Rf=Rf)
# Iterate back to start of retirement, record final value function
for i in range(T-K):
    lq_retired.update_values()
Rf2 = lq_retired.P

# Formulate LQ problem 2 (working life)
R = np.zeros((4, 4))
A = [[1 + r, -c_bar, m1, m2],
     [0,          1,  0,  0],
     [0,          1,  1,  0],
     [0,          1,  2,  1]]
B = [[-1],
     [ 0],
     [ 0],
     [ 0]]
C = [[σ],
     [0],
     [0],
     [0]]

# Set up working life LQ instance with terminal Rf from lq_retired
lq_working = LQ(Q, R, A, B, C, beta=β, T=K, Rf=Rf2)

# Simulate working state / control paths
x0 = (0, 1, 0, 0)
xp_w, up_w, wp_w = lq_working.compute_sequence(x0)
# Simulate retirement paths (note the initial condition)
xp_r, up_r, wp_r = lq_retired.compute_sequence(xp_w[:, K])

# Convert results back to assets, consumption and income
xp = np.column_stack((xp_w, xp_r[:, 1:]))
assets = xp[0, :]                    # Assets

up = np.column_stack((up_w, up_r))
c = up.flatten() + c_bar             # Consumption
```

(continues on next page)

(continued from previous page)

```
time = np.arange(1, K+1)
income_w = σ * wp_w[0, 1:K+1] + m1 * time + m2 * time**2  # Income
income_r = np.full(T-K, s)
income = np.concatenate((income_w, income_r))

# Plot results
n_rows = 2
fig, axes = plt.subplots(n_rows, 1, figsize=(12, 10))

plt.subplots_adjust(hspace=0.5)

bbox = (0., 1.02, 1., .102)
legend_args = {'bbox_to_anchor': bbox, 'loc': 3, 'mode': 'expand'}
p_args = {'lw': 2, 'alpha': 0.7}

axes[0].plot(range(1, T+1), income, 'g-', label="non-financial income",
             **p_args)
axes[0].plot(range(T), c, 'k-', label="consumption", **p_args)

axes[1].plot(range(T+1), assets, 'b-', label="assets", **p_args)
axes[1].plot(range(T+1), np.zeros(T+1), 'k-')

for ax in axes:
    ax.grid()
    ax.set_xlabel('Time')
    ax.legend(ncol=2, **legend_args)

plt.show()
```





48.8.3 Exercise 3

The first task is to find the matrices A, B, C, Q, R that define the LQ problem.

Recall that $x_t = (\bar{q}_t \;\; q_t \;\; 1)'$, while $u_t = q_{t+1} - q_t$.

Letting $m_0 := (a_0 - c)/2a_1$ and $m_1 := 1/2a_1$, we can write $\bar{q}_t = m_0 + m_1 d_t$, and then, with some manipulation

$$\bar{q}_{t+1} = m_0(1-\rho) + \rho\bar{q}_t + m_1\sigma w_{t+1}$$

By our definition of u_t, the dynamics of q_t are $q_{t+1} = q_t + u_t$.

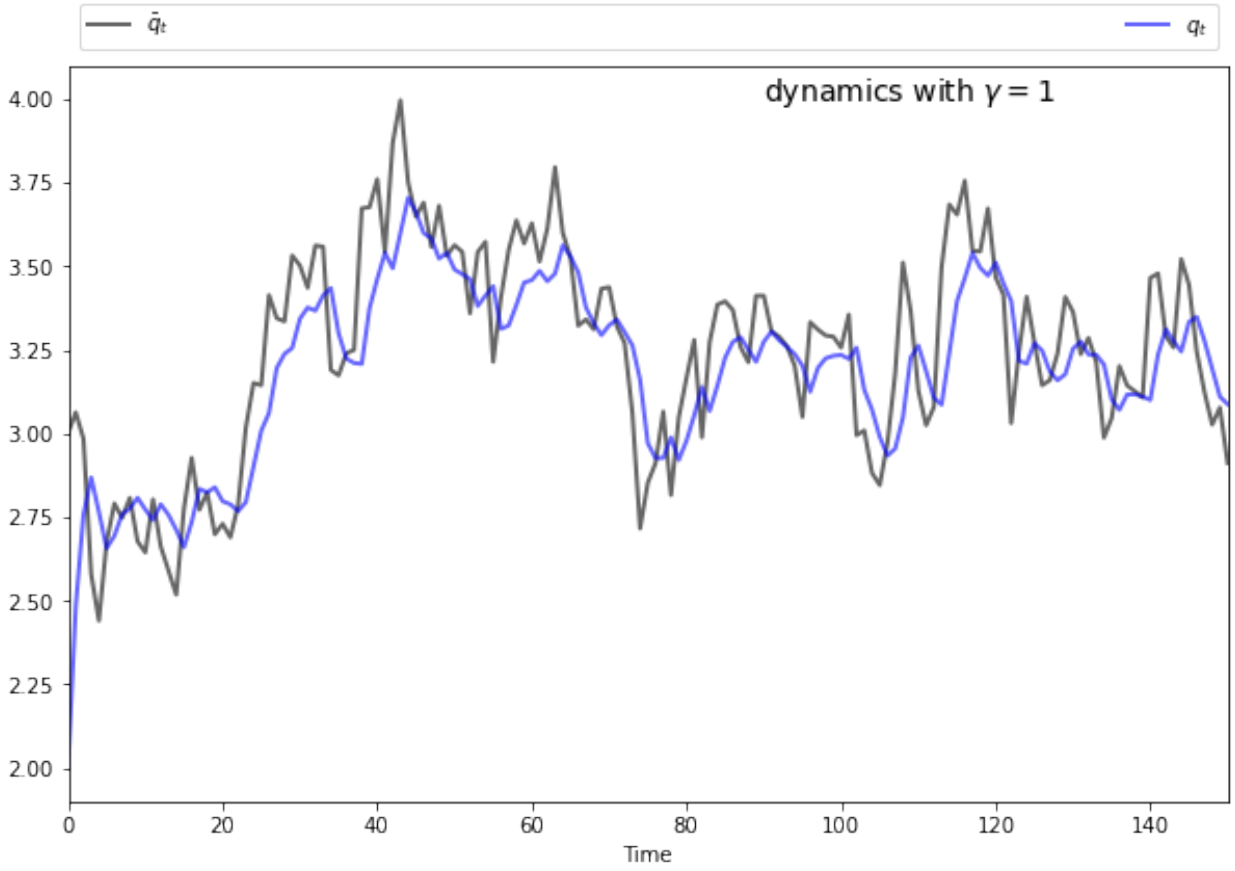
Using these facts you should be able to build the correct A, B, C matrices (and then check them against those found in the solution code below).

Suitable R, Q matrices can be found by inspecting the objective function, which we repeat here for convenience:

$$\min \mathbb{E}\left\{\sum_{t=0}^{\infty} \beta^t a_1(q_t - \bar{q}_t)^2 + \gamma u_t^2\right\}$$

Our solution code is

```
# Model parameters
a0 = 5
a1 = 0.5
σ = 0.15
ρ = 0.9
γ = 1
β = 0.95
c = 2
T = 120

# Useful constants
m0 = (a0-c)/(2 * a1)
m1 = 1/(2 * a1)

# Formulate LQ problem
Q = γ
R = [[ a1, -a1,  0],
     [-a1,  a1,  0],
     [  0,   0,  0]]
A = [[ρ, 0, m0 * (1 - ρ)],
     [0, 1,            0],
     [0, 0,            1]]

B = [[0],
     [1],
     [0]]
C = [[m1 * σ],
     [     0],
     [     0]]

lq = LQ(Q, R, A, B, C=C, beta=β)

# Simulate state / control paths
x0 = (m0, 2, 1)
xp, up, wp = lq.compute_sequence(x0, ts_length=150)
q_bar = xp[0, :]
q = xp[1, :]

# Plot simulation results
fig, ax = plt.subplots(figsize=(10, 6.5))

# Some fancy plotting stuff -- simplify if you prefer
bbox = (0., 1.01, 1., .101)
legend_args = {'bbox_to_anchor': bbox, 'loc': 3, 'mode': 'expand'}
p_args = {'lw': 2, 'alpha': 0.6}

time = range(len(q))
ax.set(xlabel='Time', xlim=(0, max(time)))
ax.plot(time, q_bar, 'k-', lw=2, alpha=0.6, label=r'$\bar q_t$')
ax.plot(time, q, 'b-', lw=2, alpha=0.6, label='$q_t$')
ax.legend(ncol=2, **legend_args)
s = f'dynamics with $\gamma = {γ}$'
ax.text(max(time) * 0.6, 1 * q_bar.max(), s, fontsize=14)
plt.show()
```


$\bar{q}_t$
q_t
dynamics with $\gamma = 1$
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
0
20
40
60
80
100
120
140
Time

CHAPTER

FORTYNINE

THE PERMANENT INCOME MODEL

Contents

In addition to what's in Anaconda, this lecture will need the following libraries:

```
!conda install -y quantecon
```

49.1 Overview

This lecture describes a rational expectations version of the famous permanent income model of Milton Friedman [Fri56].

Robert Hall cast Friedman's model within a linear-quadratic setting [Hal78].

Like Hall, we formulate an infinite-horizon linear-quadratic savings problem.

We use the model as a vehicle for illustrating

- alternative formulations of the *state* of a dynamic system
- the idea of *cointegration*
- impulse response functions
- the idea that changes in consumption are useful as predictors of movements in income

Background readings on the linear-quadratic-Gaussian permanent income model are Hall's [Hal78] and chapter 2 of [LS18].

Let's start with some imports

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (11, 5)  #set default figure size
import numpy as np
import random
from numba import njit
```

49.2 The Savings Problem

In this section, we state and solve the savings and consumption problem faced by the consumer.

49.2.1 Preliminaries

We use a class of stochastic processes called martingales.

A discrete-time martingale is a stochastic process (i.e., a sequence of random variables) $\{X_t\}$ with finite mean at each t and satisfying

$$\mathbb{E}_t[X_{t+1}] = X_t, \qquad t = 0, 1, 2, ...$$

Here $\mathbb{E}_t := \mathbb{E}[\cdot \mid \mathcal{F}_t]$ is a conditional mathematical expectation conditional on the time t *information set* $\mathcal{F}_t$.

The latter is just a collection of random variables that the modeler declares to be visible at t.

- When not explicitly defined, it is usually understood that $\mathcal{F}_t = \{X_t, X_{t-1}, ..., X_0\}$.

Martingales have the feature that the history of past outcomes provides no predictive power for changes between current and future outcomes.

For example, the current wealth of a gambler engaged in a "fair game" has this property.

One common class of martingales is the family of *random walks*.

A **random walk** is a stochastic process $\{X_t\}$ that satisfies

$$X_{t+1} = X_t + w_{t+1}$$

for some IID zero mean *innovation* sequence $\{w_t\}$.

Evidently, X_t can also be expressed as

$$X_t = \sum_{j=1}^{t} w_j + X_0$$

Not every martingale arises as a random walk (see, for example, Wald's martingale).

49.2.2 The Decision Problem

A consumer has preferences over consumption streams that are ordered by the utility functional

$$\mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t) \right] \tag{1}$$

where

- $\mathbb{E}_t$ is the mathematical expectation conditioned on the consumer's time t information
- c_t is time t consumption
- u is a strictly concave one-period utility function
- $\beta \in (0,1)$ is a discount factor

The consumer maximizes (1) by choosing a consumption, borrowing plan $\{c_t, b_{t+1}\}_{t=0}^{\infty}$ subject to the sequence of budget constraints

$$c_t + b_t = \frac{1}{1+r} b_{t+1} + y_t \quad t \geq 0 \tag{2}$$

Here

- y_t is an exogenous endowment process.
- $r > 0$ is a time-invariant risk-free net interest rate.
- b_t is one-period risk-free debt maturing at t.

The consumer also faces initial conditions b_0 and y_0, which can be fixed or random.

49.2.3 Assumptions

For the remainder of this lecture, we follow Friedman and Hall in assuming that $(1+r)^{-1} = \beta$.

Regarding the endowment process, we assume it has the *state-space representation*

$$\begin{aligned} z_{t+1} &= A z_t + C w_{t+1} \\ y_t &= U z_t \end{aligned} \tag{3}$$

where

- $\{w_t\}$ is an IID vector process with $\mathbb{E} w_t = 0$ and $\mathbb{E} w_t w_t' = I$.
- The *spectral radius* of A satisfies $\rho(A) < \sqrt{1/\beta}$.
- U is a selection vector that pins down y_t as a particular linear combination of components of z_t.

The restriction on $\rho(A)$ prevents income from growing so fast that discounted geometric sums of some quadratic forms to be described below become infinite.

Regarding preferences, we assume the quadratic utility function

$$u(c_t) = -(c_t - \gamma)^2$$

where γ is a bliss level of consumption.

Note: Along with this quadratic utility specification, we allow consumption to be negative. However, by choosing parameters appropriately, we can make the probability that the model generates negative consumption paths over finite time horizons as low as desired.

Finally, we impose the *no Ponzi scheme* condition

$$\mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t b_t^2 \right] < \infty \tag{4}$$

This condition rules out an always-borrow scheme that would allow the consumer to enjoy bliss consumption forever.

49.2.4 First-Order Conditions

First-order conditions for maximizing (1) subject to (2) are

$$\mathbb{E}_t[u'(c_{t+1})] = u'(c_t), \qquad t = 0, 1, \ldots \tag{5}$$

These optimality conditions are also known as *Euler equations*.

If you're not sure where they come from, you can find a proof sketch in the *appendix*.

With our quadratic preference specification, (5) has the striking implication that consumption follows a martingale:

$$\mathbb{E}_t[c_{t+1}] = c_t \tag{6}$$

(In fact, quadratic preferences are *necessary* for this conclusion[1].)

One way to interpret (6) is that consumption will change only when "new information" about permanent income is revealed.

These ideas will be clarified below.

49.2.5 The Optimal Decision Rule

Now let's deduce the optimal decision rule[2].

Note: One way to solve the consumer's problem is to apply *dynamic programming* as in *this lecture*. We do this later. But first we use an alternative approach that is revealing and shows the work that dynamic programming does for us behind the scenes.

In doing so, we need to combine

1. the optimality condition (6)
2. the period-by-period budget constraint (2), and
3. the boundary condition (4)

To accomplish this, observe first that (4) implies $\lim_{t\to\infty} \beta^{\frac{t}{2}} b_{t+1} = 0$.

Using this restriction on the debt path and solving (2) forward yields

$$b_t = \sum_{j=0}^{\infty} \beta^j (y_{t+j} - c_{t+j}) \tag{7}$$

Take conditional expectations on both sides of (7) and use the martingale property of consumption and the *law of iterated expectations* to deduce

$$b_t = \sum_{j=0}^{\infty} \beta^j \mathbb{E}_t[y_{t+j}] - \frac{c_t}{1-\beta} \tag{8}$$

Expressed in terms of c_t we get

$$c_t = (1-\beta)\left[\sum_{j=0}^{\infty} \beta^j \mathbb{E}_t[y_{t+j}] - b_t\right] = \frac{r}{1+r}\left[\sum_{j=0}^{\infty} \beta^j \mathbb{E}_t[y_{t+j}] - b_t\right] \tag{9}$$

[1] A linear marginal utility is essential for deriving (6) from (5). Suppose instead that we had imposed the following more standard assumptions on the utility function: $u'(c) > 0$, $u''(c) < 0$, $u'''(c) > 0$ and required that $c \geq 0$. The Euler equation remains (5). But the fact that $u''' < 0$ implies via Jensen's inequality that $\mathbb{E}_t[u'(c_{t+1})] > u'(\mathbb{E}_t[c_{t+1}])$. This inequality together with (5) implies that $\mathbb{E}_t[c_{t+1}] > c_t$ (consumption is said to be a 'submartingale'), so that consumption stochastically diverges to $+\infty$. The consumer's savings also diverge to $+\infty$.

[2] An optimal decision rule is a map from the current state into current actions—in this case, consumption.

where the last equality uses $(1+r)\beta = 1$.

These last two equations assert that consumption equals *economic income*

- **financial wealth** equals $-b_t$
- **non-financial wealth** equals $\sum_{j=0}^{\infty} \beta^j \mathbb{E}_t[y_{t+j}]$
- **total wealth** equals the sum of financial and non-financial wealth
- a **marginal propensity to consume out of total wealth** equals the interest factor $\frac{r}{1+r}$
- **economic income** equals
  - a constant marginal propensity to consume times the sum of non-financial wealth and financial wealth
  - the amount the consumer can consume while leaving its wealth intact

Responding to the State

The *state* vector confronting the consumer at t is $\begin{bmatrix} b_t & z_t \end{bmatrix}$.

Here

- z_t is an *exogenous* component, unaffected by consumer behavior.
- b_t is an *endogenous* component (since it depends on the decision rule).

Note that z_t contains all variables useful for forecasting the consumer's future endowment.

It is plausible that current decisions c_t and b_{t+1} should be expressible as functions of z_t and b_t.

This is indeed the case.

In fact, from *this discussion*, we see that

$$\sum_{j=0}^{\infty} \beta^j \mathbb{E}_t[y_{t+j}] = \mathbb{E}_t \left[\sum_{j=0}^{\infty} \beta^j y_{t+j} \right] = U(I - \beta A)^{-1} z_t$$

Combining this with (9) gives

$$c_t = \frac{r}{1+r} \left[U(I - \beta A)^{-1} z_t - b_t \right] \tag{10}$$

Using this equality to eliminate c_t in the budget constraint (2) gives

$$\begin{aligned} b_{t+1} &= (1+r)(b_t + c_t - y_t) \\ &= (1+r)b_t + r[U(I - \beta A)^{-1} z_t - b_t] - (1+r)U z_t \\ &= b_t + U[r(I - \beta A)^{-1} - (1+r)I] z_t \\ &= b_t + U(I - \beta A)^{-1}(A - I) z_t \end{aligned}$$

To get from the second last to the last expression in this chain of equalities is not trivial.

A key is to use the fact that $(1+r)\beta = 1$ and $(I - \beta A)^{-1} = \sum_{j=0}^{\infty} \beta^j A^j$.

We've now successfully written c_t and b_{t+1} as functions of b_t and z_t.

A State-Space Representation

We can summarize our dynamics in the form of a linear state-space system governing consumption, debt and income:

$$\begin{aligned} z_{t+1} &= Az_t + Cw_{t+1} \\ b_{t+1} &= b_t + U[(I-\beta A)^{-1}(A-I)]z_t \\ y_t &= Uz_t \\ c_t &= (1-\beta)[U(I-\beta A)^{-1}z_t - b_t] \end{aligned} \tag{11}$$

To write this more succinctly, let

$$x_t = \begin{bmatrix} z_t \\ b_t \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} A & 0 \\ U(I-\beta A)^{-1}(A-I) & 1 \end{bmatrix}, \quad \tilde{C} = \begin{bmatrix} C \\ 0 \end{bmatrix}$$

and

$$\tilde{U} = \begin{bmatrix} U & 0 \\ (1-\beta)U(I-\beta A)^{-1} & -(1-\beta) \end{bmatrix}, \quad \tilde{y}_t = \begin{bmatrix} y_t \\ c_t \end{bmatrix}$$

Then we can express equation (11) as

$$\begin{aligned} x_{t+1} &= \tilde{A}x_t + \tilde{C}w_{t+1} \\ \tilde{y}_t &= \tilde{U}x_t \end{aligned} \tag{12}$$

We can use the following formulas from *linear state space models* to compute population mean $\mu_t = \mathbb{E}x_t$ and covariance $\Sigma_t := \mathbb{E}[(x_t - \mu_t)(x_t - \mu_t)']$

$$\mu_{t+1} = \tilde{A}\mu_t \quad \text{with} \quad \mu_0 \text{ given} \tag{13}$$

$$\Sigma_{t+1} = \tilde{A}\Sigma_t\tilde{A}' + \tilde{C}\tilde{C}' \quad \text{with} \quad \Sigma_0 \text{ given} \tag{14}$$

We can then compute the mean and covariance of $\tilde{y}_t$ from

$$\begin{aligned} \mu_{y,t} &= \tilde{U}\mu_t \\ \Sigma_{y,t} &= \tilde{U}\Sigma_t\tilde{U}' \end{aligned} \tag{15}$$

A Simple Example with IID Income

To gain some preliminary intuition on the implications of (11), let's look at a highly stylized example where income is just IID.

(Later examples will investigate more realistic income streams.)

In particular, let $\{w_t\}_{t=1}^{\infty}$ be IID and scalar standard normal, and let

$$z_t = \begin{bmatrix} z_t^1 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & \mu \end{bmatrix}, \quad C = \begin{bmatrix} \sigma \\ 0 \end{bmatrix}$$

Finally, let $b_0 = z_0^1 = 0$.

Under these assumptions, we have $y_t = \mu + \sigma w_t \sim N(\mu, \sigma^2)$.

Further, if you work through the state space representation, you will see that

$$b_t = -\sigma \sum_{j=1}^{t-1} w_j$$

$$c_t = \mu + (1-\beta)\sigma \sum_{j=1}^{t} w_j$$

Thus income is IID and debt and consumption are both Gaussian random walks.

Defining assets as $-b_t$, we see that assets are just the cumulative sum of unanticipated incomes prior to the present date.

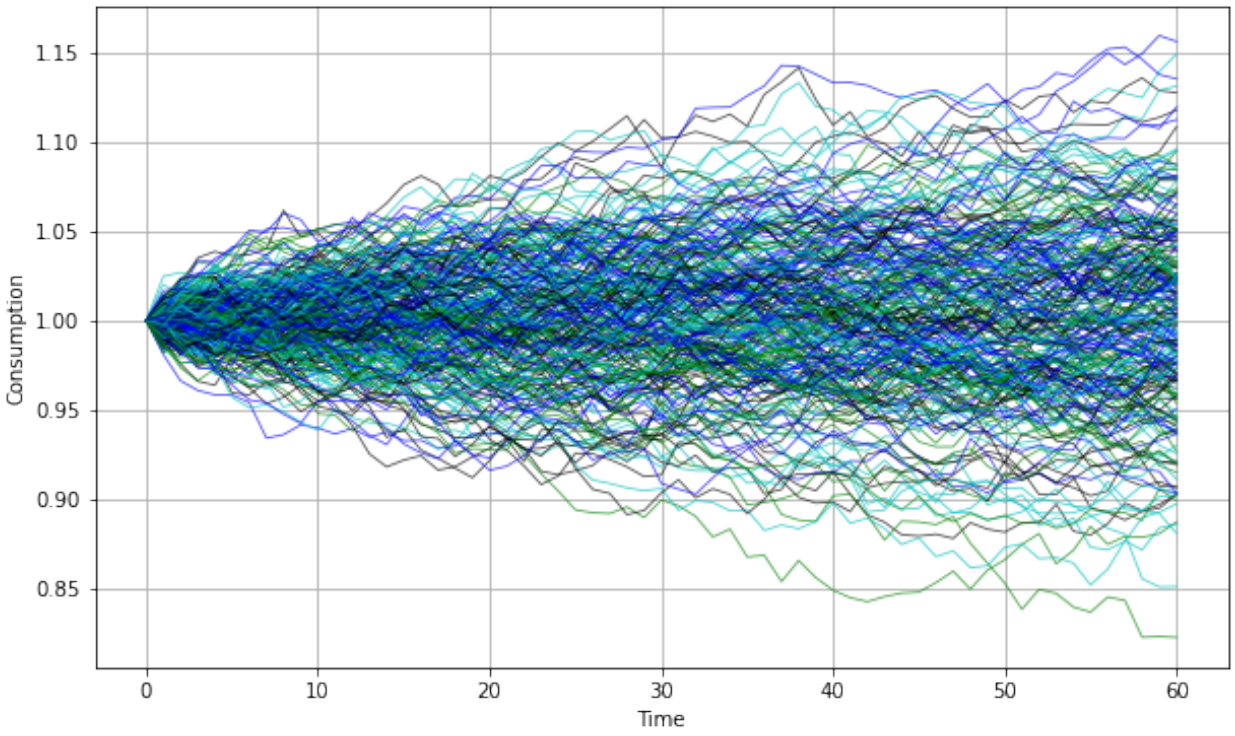
The next figure shows a typical realization with $r = 0.05$, $\mu = 1$, and $\sigma = 0.15$

```
r = 0.05
β = 1 / (1 + r)
σ = 0.15
μ = 1
T = 60

@njit
def time_path(T):
    w = np.random.randn(T+1)   # w_0, w_1, ..., w_T
    w[0] = 0
    b = np.zeros(T+1)
    for t in range(1, T+1):
        b[t] = w[1:t].sum()
    b = -σ * b
    c = μ + (1 - β) * (σ * w - b)
    return w, b, c

w, b, c = time_path(T)

fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(μ + σ * w, 'g-', label="Non-financial income")
ax.plot(c, 'k-', label="Consumption")
ax.plot( b, 'b-', label="Debt")
ax.legend(ncol=3, mode='expand', bbox_to_anchor=(0., 1.02, 1., .102))
ax.grid()
ax.set_xlabel('Time')

plt.show()
```

Observe that consumption is considerably smoother than income.

The figure below shows the consumption paths of 250 consumers with independent income streams

```
fig, ax = plt.subplots(figsize=(10, 6))

b_sum = np.zeros(T+1)
for i in range(250):
    w, b, c = time_path(T)  # Generate new time path
    rcolor = random.choice(('c', 'g', 'b', 'k'))
    ax.plot(c, color=rcolor, lw=0.8, alpha=0.7)

ax.grid()
ax.set(xlabel='Time', ylabel='Consumption')

plt.show()
```



49.3 Alternative Representations

In this section, we shed more light on the evolution of savings, debt and consumption by representing their dynamics in several different ways.

49.3.1 Hall's Representation

Hall [Hal78] suggested an insightful way to summarize the implications of LQ permanent income theory.

First, to represent the solution for b_t, shift (9) forward one period and eliminate b_{t+1} by using (2) to obtain

$$c_{t+1} = (1-\beta)\sum_{j=0}^{\infty}\beta^j \mathbb{E}_{t+1}[y_{t+j+1}] - (1-\beta)\left[\beta^{-1}(c_t + b_t - y_t)\right]$$

If we add and subtract $\beta^{-1}(1-\beta)\sum_{j=0}^{\infty}\beta^j \mathbb{E}_t y_{t+j}$ from the right side of the preceding equation and rearrange, we obtain

$$c_{t+1} - c_t = (1-\beta)\sum_{j=0}^{\infty}\beta^j\left\{\mathbb{E}_{t+1}[y_{t+j+1}] - \mathbb{E}_t[y_{t+j+1}]\right\} \tag{16}$$

The right side is the time $t+1$ *innovation to the expected present value* of the endowment process $\{y_t\}$.

We can represent the optimal decision rule for (c_t, b_{t+1}) in the form of (16) and (8), which we repeat:

$$b_t = \sum_{j=0}^{\infty}\beta^j \mathbb{E}_t[y_{t+j}] - \frac{1}{1-\beta}c_t \tag{17}$$

Equation (17) asserts that the consumer's debt due at t equals the expected present value of its endowment minus the expected present value of its consumption stream.

A high debt thus indicates a large expected present value of surpluses $y_t - c_t$.

Recalling again our discussion on *forecasting geometric sums*, we have

$$\mathbb{E}_t \sum_{j=0}^{\infty} \beta^j y_{t+j} = U(I - \beta A)^{-1} z_t$$
$$\mathbb{E}_{t+1} \sum_{j=0}^{\infty} \beta^j y_{t+j+1} = U(I - \beta A)^{-1} z_{t+1}$$
$$\mathbb{E}_t \sum_{j=0}^{\infty} \beta^j y_{t+j+1} = U(I - \beta A)^{-1} A z_t$$

Using these formulas together with (3) and substituting into (16) and (17) gives the following representation for the consumer's optimum decision rule:

$$\begin{aligned} c_{t+1} &= c_t + (1-\beta) U(I - \beta A)^{-1} C w_{t+1} \\ b_t &= U(I - \beta A)^{-1} z_t - \frac{1}{1-\beta} c_t \\ y_t &= U z_t \\ z_{t+1} &= A z_t + C w_{t+1} \end{aligned} \tag{18}$$

Representation (18) makes clear that

- The state can be taken as (c_t, z_t).
  - The endogenous part is c_t and the exogenous part is z_t.
  - Debt b_t has disappeared as a component of the state because it is encoded in c_t.
- Consumption is a random walk with innovation $(1-\beta) U(I - \beta A)^{-1} C w_{t+1}$.
  - This is a more explicit representation of the martingale result in (6).

49.3.2 Cointegration

Representation (18) reveals that the joint process $\{c_t, b_t\}$ possesses the property that Engle and Granger [EG87] called cointegration.

Cointegration is a tool that allows us to apply powerful results from the theory of stationary stochastic processes to (certain transformations of) nonstationary models.

To apply cointegration in the present context, suppose that z_t is asymptotically stationary[3].

Despite this, both c_t and b_t will be non-stationary because they have unit roots (see (11) for b_t).

Nevertheless, there is a linear combination of c_t, b_t that *is* asymptotically stationary.

In particular, from the second equality in (18) we have

$$(1-\beta) b_t + c_t = (1-\beta) U(I - \beta A)^{-1} z_t \tag{19}$$

Hence the linear combination $(1-\beta) b_t + c_t$ is asymptotically stationary.

Accordingly, Granger and Engle would call $\begin{bmatrix} (1-\beta) & 1 \end{bmatrix}$ a **cointegrating vector** for the state.

When applied to the nonstationary vector process $\begin{bmatrix} b_t & c_t \end{bmatrix}'$, it yields a process that is asymptotically stationary.

[3] This would be the case if, for example, the *spectral radius* of A is strictly less than one.

Equation (19) can be rearranged to take the form

$$(1-\beta)b_t + c_t = (1-\beta)\mathbb{E}_t \sum_{j=0}^{\infty} \beta^j y_{t+j} \tag{20}$$

Equation (20) asserts that the *cointegrating residual* on the left side equals the conditional expectation of the geometric sum of future incomes on the right[4].

49.3.3 Cross-Sectional Implications

Consider again (18), this time in light of our discussion of distribution dynamics in the *lecture on linear systems*.

The dynamics of c_t are given by

$$c_{t+1} = c_t + (1-\beta)U(I-\beta A)^{-1}Cw_{t+1} \tag{21}$$

or

$$c_t = c_0 + \sum_{j=1}^{t} \hat{w}_j \quad \text{for} \quad \hat{w}_{t+1} := (1-\beta)U(I-\beta A)^{-1}Cw_{t+1}$$

The unit root affecting c_t causes the time t variance of c_t to grow linearly with t.

In particular, since $\{\hat{w}_t\}$ is IID, we have

$$\text{Var}[c_t] = \text{Var}[c_0] + t\,\hat{\sigma}^2 \tag{22}$$

where

$$\hat{\sigma}^2 := (1-\beta)^2 U(I-\beta A)^{-1}CC'(I-\beta A')^{-1}U'$$

When $\hat{\sigma} > 0$, $\{c_t\}$ has no asymptotic distribution.

Let's consider what this means for a cross-section of ex-ante identical consumers born at time 0.

Let the distribution of c_0 represent the cross-section of initial consumption values.

Equation (22) tells us that the variance of c_t increases over time at a rate proportional to t.

A number of different studies have investigated this prediction and found some support for it (see, e.g., [DP94], [STY04]).

49.3.4 Impulse Response Functions

Impulse response functions measure responses to various impulses (i.e., temporary shocks).

The impulse response function of $\{c_t\}$ to the innovation $\{w_t\}$ is a box.

In particular, the response of c_{t+j} to a unit increase in the innovation w_{t+1} is $(1-\beta)U(I-\beta A)^{-1}C$ for all $j \geq 1$.

[4] See [JYC88], [LL01], [LL04] for interesting applications of related ideas.

49.3.5 Moving Average Representation

It's useful to express the innovation to the expected present value of the endowment process in terms of a moving average representation for income y_t.

The endowment process defined by (3) has the moving average representation

$$y_{t+1} = d(L)w_{t+1} \tag{23}$$

where

- $d(L) = \sum_{j=0}^{\infty} d_j L^j$ for some sequence d_j, where L is the lag operator[5]
- at time t, the consumer has an information set[6] $w^t = [w_t, w_{t-1}, ...]$

Notice that

$$y_{t+j} - \mathbb{E}_t[y_{t+j}] = d_0 w_{t+j} + d_1 w_{t+j-1} + \cdots + d_{j-1} w_{t+1}$$

It follows that

$$\mathbb{E}_{t+1}[y_{t+j}] - \mathbb{E}_t[y_{t+j}] = d_{j-1} w_{t+1} \tag{24}$$

Using (24) in (16) gives

$$c_{t+1} - c_t = (1-\beta)d(\beta)w_{t+1} \tag{25}$$

The object $d(\beta)$ is the **present value of the moving average coefficients** in the representation for the endowment process y_t.

49.4 Two Classic Examples

We illustrate some of the preceding ideas with two examples.

In both examples, the endowment follows the process $y_t = z_{1t} + z_{2t}$ where

$$\begin{bmatrix} z_{1t+1} \\ z_{2t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_{1t} \\ z_{2t} \end{bmatrix} + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} w_{1t+1} \\ w_{2t+1} \end{bmatrix}$$

Here

- w_{t+1} is an IID 2×1 process distributed as $N(0, I)$.
- z_{1t} is a permanent component of y_t.
- z_{2t} is a purely transitory component of y_t.

49.4.1 Example 1

Assume as before that the consumer observes the state z_t at time t.

In view of (18) we have

$$c_{t+1} - c_t = \sigma_1 w_{1t+1} + (1-\beta)\sigma_2 w_{2t+1} \tag{26}$$

Formula (26) shows how an increment $\sigma_1 w_{1t+1}$ to the permanent component of income z_{1t+1} leads to

[5] Representation (3) implies that $d(L) = U(I - AL)^{-1}C$.

[6] A moving average representation for a process y_t is said to be **fundamental** if the linear space spanned by y^t is equal to the linear space spanned by w^t. A time-invariant innovations representation, attained via the Kalman filter, is by construction fundamental.

- a permanent one-for-one increase in consumption and
- no increase in savings $-b_{t+1}$

But the purely transitory component of income $\sigma_2 w_{2t+1}$ leads to a permanent increment in consumption by a fraction $1-\beta$ of transitory income.

The remaining fraction β is saved, leading to a permanent increment in $-b_{t+1}$.

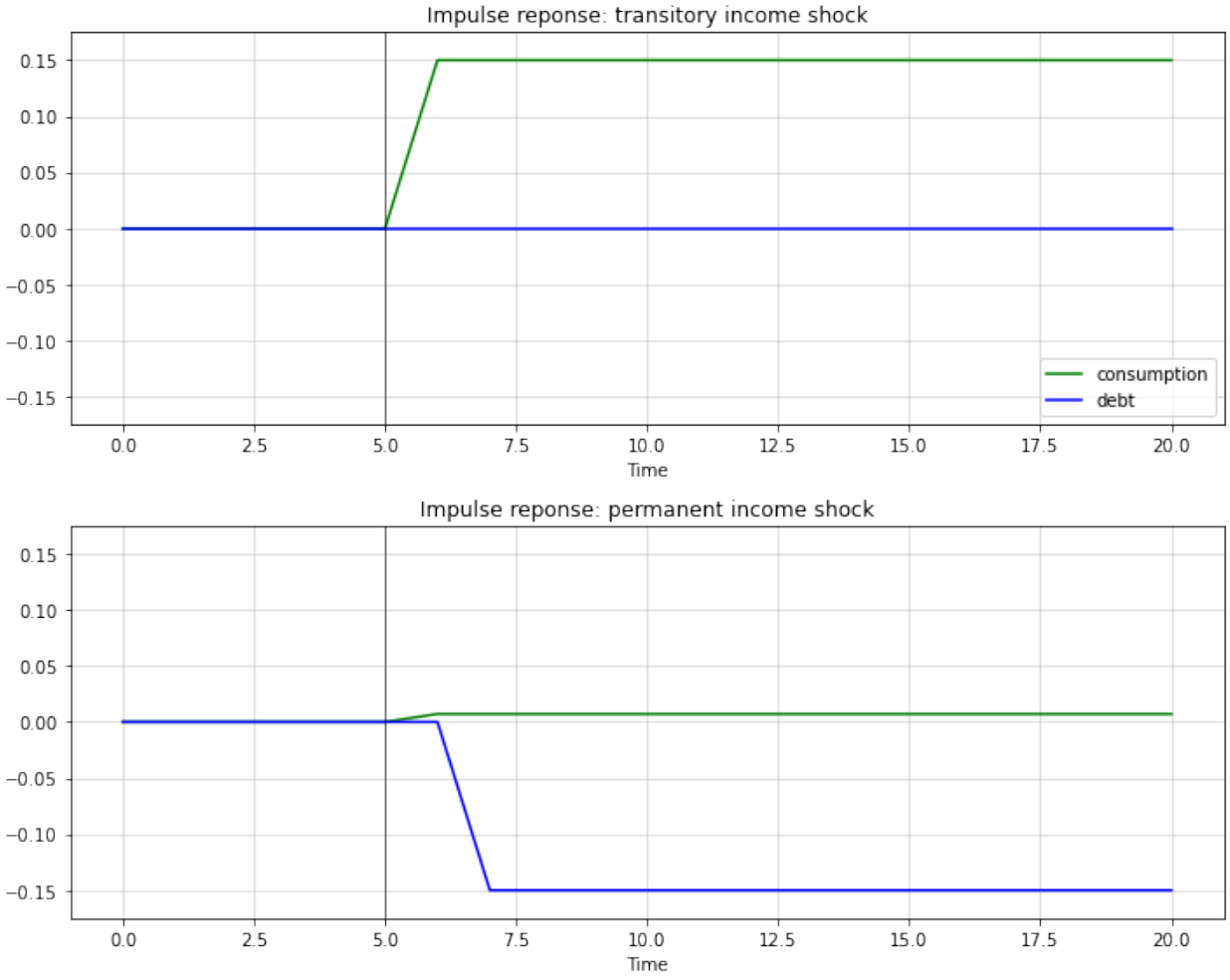
Application of the formula for debt in (11) to this example shows that

$$b_{t+1} - b_t = -z_{2t} = -\sigma_2 w_{2t} \tag{27}$$

This confirms that none of $\sigma_1 w_{1t}$ is saved, while all of $\sigma_2 w_{2t}$ is saved.

The next figure illustrates these very different reactions to transitory and permanent income shocks using impulse-response functions

```
r = 0.05
β = 1 / (1 + r)
S = 5   # Impulse date
σ1 = σ2 = 0.15

@njit
def time_path(T, permanent=False):
    "Time path of consumption and debt given shock sequence"
    w1 = np.zeros(T+1)
    w2 = np.zeros(T+1)
    b = np.zeros(T+1)
    c = np.zeros(T+1)
    if permanent:
        w1[S+1] = 1.0
    else:
        w2[S+1] = 1.0
    for t in range(1, T):
        b[t+1] = b[t] - σ2 * w2[t]
        c[t+1] = c[t] + σ1 * w1[t+1] + (1 - β) * σ2 * w2[t+1]
    return b, c


fig, axes = plt.subplots(2, 1, figsize=(10, 8))
titles = ['transitory', 'permanent']

L = 0.175

for ax, truefalse, title in zip(axes, (True, False), titles):
    b, c = time_path(T=20, permanent=truefalse)
    ax.set_title(f'Impulse reponse: {title} income shock')
    ax.plot(c, 'g-', label="consumption")
    ax.plot(b, 'b-', label="debt")
    ax.plot((S, S), (-L, L), 'k-', lw=0.5)
    ax.grid(alpha=0.5)
    ax.set(xlabel=r'Time', ylim=(-L, L))

axes[0].legend(loc='lower right')

plt.tight_layout()
plt.show()
```



49.4.2 Example 2

Assume now that at time t the consumer observes y_t, and its history up to t, but not z_t.

Under this assumption, it is appropriate to use an *innovation representation* to form A, C, U in (18).

The discussion in sections 2.9.1 and 2.11.3 of [LS18] shows that the pertinent state space representation for y_t is

$$\begin{bmatrix} y_{t+1} \\ a_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & -(1-K) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_t \\ a_t \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} a_{t+1}$$

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} y_t \\ a_t \end{bmatrix}$$

where

- $K :=$ the stationary Kalman gain
- $a_t := y_t - E[y_t \mid y_{t-1}, \ldots, y_0]$

In the same discussion in [LS18] it is shown that $K \in [0, 1]$ and that K increases as σ_1/σ_2 does.

In other words, K increases as the ratio of the standard deviation of the permanent shock to that of the transitory shock increases.

Please see *first look at the Kalman filter*.

Applying formulas (18) implies

$$c_{t+1} - c_t = [1 - \beta(1-K)]a_{t+1} \tag{28}$$

where the endowment process can now be represented in terms of the univariate innovation to y_t as

$$y_{t+1} - y_t = a_{t+1} - (1-K)a_t \tag{29}$$

Equation (29) indicates that the consumer regards

- fraction K of an innovation a_{t+1} to y_{t+1} as *permanent*
- fraction $1-K$ as purely transitory

The consumer permanently increases his consumption by the full amount of his estimate of the permanent part of a_{t+1}, but by only $(1-\beta)$ times his estimate of the purely transitory part of a_{t+1}.

Therefore, in total, he permanently increments his consumption by a fraction $K + (1-\beta)(1-K) = 1 - \beta(1-K)$ of a_{t+1}.

He saves the remaining fraction $\beta(1-K)$.

According to equation (29), the first difference of income is a first-order moving average.

Equation (28) asserts that the first difference of consumption is IID.

Application of formula to this example shows that

$$b_{t+1} - b_t = (K-1)a_t \tag{30}$$

This indicates how the fraction K of the innovation to y_t that is regarded as permanent influences the fraction of the innovation that is saved.

49.5 Further Reading

The model described above significantly changed how economists think about consumption.

While Hall's model does a remarkably good job as a first approximation to consumption data, it's widely believed that it doesn't capture important aspects of some consumption/savings data.

For example, liquidity constraints and precautionary savings appear to be present sometimes.

Further discussion can be found in, e.g., [HM82], [Par99], [Dea91], [Car01].

49.6 Appendix: The Euler Equation

Where does the first-order condition (5) come from?

Here we'll give a proof for the two-period case, which is representative of the general argument.

The finite horizon equivalent of the no-Ponzi condition is that the agent cannot end her life in debt, so $b_2 = 0$.

From the budget constraint (2) we then have

$$c_0 = \frac{b_1}{1+r} - b_0 + y_0 \quad \text{and} \quad c_1 = y_1 - b_1$$

Here b_0 and y_0 are given constants.

Substituting these constraints into our two-period objective $u(c_0) + \beta \mathbb{E}_0[u(c_1)]$ gives

$$\max_{b_1} \left\{ u\left(\frac{b_1}{R} - b_0 + y_0\right) + \beta \, \mathbb{E}_0[u(y_1 - b_1)] \right\}$$

You will be able to verify that the first-order condition is

$$u'(c_0) = \beta R \, \mathbb{E}_0[u'(c_1)]$$

Using $\beta R = 1$ gives (5) in the two-period case.

The proof for the general case is similar.

CHAPTER

FIFTY

PERMANENT INCOME II: LQ TECHNIQUES

Contents

In addition to what's in Anaconda, this lecture will need the following libraries:

```
!conda install -y quantecon
```

50.1 Overview

This lecture continues our analysis of the linear-quadratic (LQ) permanent income model of savings and consumption.

As we saw in our *previous lecture* on this topic, Robert Hall [Hal78] used the LQ permanent income model to restrict and interpret intertemporal comovements of nondurable consumption, nonfinancial income, and financial wealth.

For example, we saw how the model asserts that for any covariance stationary process for nonfinancial income

- consumption is a random walk
- financial wealth has a unit root and is cointegrated with consumption

Other applications use the same LQ framework.

For example, a model isomorphic to the LQ permanent income model has been used by Robert Barro [Bar79] to interpret intertemporal comovements of a government's tax collections, its expenditures net of debt service, and its public debt.

This isomorphism means that in analyzing the LQ permanent income model, we are in effect also analyzing the Barro tax smoothing model.

It is just a matter of appropriately relabeling the variables in Hall's model.

In this lecture, we'll

- show how the solution to the LQ permanent income model can be obtained using LQ control methods.
- represent the model as a linear state space system as in *this lecture*.

- apply QuantEcon's LinearStateSpace class to characterize statistical features of the consumer's optimal consumption and borrowing plans.

We'll then use these characterizations to construct a simple model of cross-section wealth and consumption dynamics in the spirit of Truman Bewley [Bew86].

(Later we'll study other Bewley models—see *this lecture*.)

The model will prove useful for illustrating concepts such as

- stationarity
- ergodicity
- ensemble moments and cross-section observations

Let's start with some imports:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (11, 5)  #set default figure size
import quantecon as qe
import numpy as np
import scipy.linalg as la
```

50.2 Setup

Let's recall the basic features of the model discussed in the *permanent income model*.

Consumer preferences are ordered by

$$E_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{1}$$

where $u(c) = -(c - \gamma)^2$.

The consumer maximizes (1) by choosing a consumption, borrowing plan $\{c_t, b_{t+1}\}_{t=0}^{\infty}$ subject to the sequence of budget constraints

$$c_t + b_t = \frac{1}{1+r} b_{t+1} + y_t, \quad t \geq 0 \tag{2}$$

and the no-Ponzi condition

$$E_0 \sum_{t=0}^{\infty} \beta^t b_t^2 < \infty \tag{3}$$

The interpretation of all variables and parameters are the same as in the *previous lecture*.

We continue to assume that $(1+r)\beta = 1$.

The dynamics of $\{y_t\}$ again follow the linear state space model

$$\begin{aligned} z_{t+1} &= A z_t + C w_{t+1} \\ y_t &= U z_t \end{aligned} \tag{4}$$

The restrictions on the shock process and parameters are the same as in our *previous lecture*.

50.2.1 Digression on a Useful Isomorphism

The LQ permanent income model of consumption is mathematically isomorphic with a version of Barro's [Bar79] model of tax smoothing.

In the LQ permanent income model

- the household faces an exogenous process of nonfinancial income
- the household wants to smooth consumption across states and time

In the Barro tax smoothing model

- a government faces an exogenous sequence of government purchases (net of interest payments on its debt)
- a government wants to smooth tax collections across states and time

If we set

- T_t, total tax collections in Barro's model to consumption c_t in the LQ permanent income model.
- G_t, exogenous government expenditures in Barro's model to nonfinancial income y_t in the permanent income model.
- B_t, government risk-free one-period assets falling due in Barro's model to risk-free one-period consumer debt b_t falling due in the LQ permanent income model.
- R, the gross rate of return on risk-free one-period government debt in Barro's model to the gross rate of return $1 + r$ on financial assets in the permanent income model of consumption.

then the two models are mathematically equivalent.

All characterizations of a $\{c_t, y_t, b_t\}$ in the LQ permanent income model automatically apply to a $\{T_t, G_t, B_t\}$ process in the Barro model of tax smoothing.

See consumption and tax smoothing models for further exploitation of an isomorphism between consumption and tax smoothing models.

50.2.2 A Specification of the Nonfinancial Income Process

For the purposes of this lecture, let's assume $\{y_t\}$ is a second-order univariate autoregressive process:

$$y_{t+1} = \alpha + \rho_1 y_t + \rho_2 y_{t-1} + \sigma w_{t+1}$$

We can map this into the linear state space framework in (4), as discussed in our lecture on *linear models*.

To do so we take

$$z_t = \begin{bmatrix} 1 \\ y_t \\ y_{t-1} \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ \alpha & \rho_1 & \rho_2 \\ 0 & 1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 \\ \sigma \\ 0 \end{bmatrix}, \quad \text{and} \quad U = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

50.3 The LQ Approach

Previously we solved the permanent income model by solving a system of linear expectational difference equations subject to two boundary conditions.

Here we solve the same model using *LQ methods* based on dynamic programming.

After confirming that answers produced by the two methods agree, we apply QuantEcon's LinearStateSpace class to illustrate features of the model.

Why solve a model in two distinct ways?

Because by doing so we gather insights about the structure of the model.

Our earlier approach based on solving a system of expectational difference equations brought to the fore the role of the consumer's expectations about future nonfinancial income.

On the other hand, formulating the model in terms of an LQ dynamic programming problem reminds us that

- finding the state (of a dynamic programming problem) is an art, and
- iterations on a Bellman equation implicitly jointly solve both a forecasting problem and a control problem

50.3.1 The LQ Problem

Recall from our *lecture on LQ theory* that the optimal linear regulator problem is to choose a decision rule for u_t to minimize

$$\mathbb{E} \sum_{t=0}^{\infty} \beta^t \{x_t' R x_t + u_t' Q u_t\},$$

subject to x_0 given and the law of motion

$$x_{t+1} = \tilde{A} x_t + \tilde{B} u_t + \tilde{C} w_{t+1}, \qquad t \geq 0, \tag{5}$$

where w_{t+1} is IID with mean vector zero and $\mathbb{E} w_t w_t' = I$.

The tildes in $\tilde{A}, \tilde{B}, \tilde{C}$ are to avoid clashing with notation in (4).

The value function for this problem is $v(x) = -x' P x - d$, where

- P is the unique positive semidefinite solution of the *corresponding matrix Riccati equation*.
- The scalar d is given by $d = \beta(1-\beta)^{-1} \text{trace}(P \tilde{C} \tilde{C}')$.

The optimal policy is $u_t = -F x_t$, where $F := \beta(Q + \beta \tilde{B}' P \tilde{B})^{-1} \tilde{B}' P \tilde{A}$.

Under an optimal decision rule F, the state vector x_t evolves according to $x_{t+1} = (\tilde{A} - \tilde{B}F) x_t + \tilde{C} w_{t+1}$.

50.3.2 Mapping into the LQ Framework

To map into the LQ framework, we'll use

$$x_t := \begin{bmatrix} z_t \\ b_t \end{bmatrix} = \begin{bmatrix} 1 \\ y_t \\ y_{t-1} \\ b_t \end{bmatrix}$$

as the state vector and $u_t := c_t - \gamma$ as the control.

With this notation and $U_\gamma := \begin{bmatrix}\gamma & 0 & 0\end{bmatrix}$, we can write the state dynamics as in (5) when

$$\tilde{A} := \begin{bmatrix} A & 0 \\ (1+r)(U_\gamma - U) & 1+r \end{bmatrix} \quad \tilde{B} := \begin{bmatrix} 0 \\ 1+r \end{bmatrix} \quad \text{and} \quad \tilde{C} := \begin{bmatrix} C \\ 0 \end{bmatrix} w_{t+1}$$

Please confirm for yourself that, with these definitions, the LQ dynamics (5) match the dynamics of z_t and b_t described above.

To map utility into the quadratic form $x_t' R x_t + u_t' Q u_t$ we can set

- $Q := 1$ (remember that we are minimizing) and
- $R :=$ a 4×4 matrix of zeros

However, there is one problem remaining.

We have no direct way to capture the non-recursive restriction (3) on the debt sequence $\{b_t\}$ from within the LQ framework.

To try to enforce it, we're going to use a trick: put a small penalty on b_t^2 in the criterion function.

In the present setting, this means adding a small entry $\epsilon > 0$ in the $(4, 4)$ position of R.

That will induce a (hopefully) small approximation error in the decision rule.

We'll check whether it really is small numerically soon.

50.4 Implementation

Let's write some code to solve the model.

One comment before we start is that the bliss level of consumption γ in the utility function has no effect on the optimal decision rule.

We saw this in the previous lecture *permanent income*.

The reason is that it drops out of the Euler equation for consumption.

In what follows we set it equal to unity.

50.4.1 The Exogenous Nonfinancial Income Process

First, we create the objects for the optimal linear regulator

```
# Set parameters
α, β, ρ1, ρ2, σ = 10.0, 0.95, 0.9, 0.0, 1.0

R = 1 / β
A = np.array([[1., 0., 0.],
              [α,  ρ1, ρ2],
              [0., 1., 0.]])
C = np.array([[0.], [σ], [0.]])
G = np.array([[0., 1., 0.]])

# Form LinearStateSpace system and pull off steady state moments
μ_z0 = np.array([[1.0], [0.0], [0.0]])
Σ_z0 = np.zeros((3, 3))
Lz = qe.LinearStateSpace(A, C, G, mu_0=μ_z0, Sigma_0=Σ_z0)
μ_z, μ_y, Σ_z, Σ_y, Σ_yx = Lz.stationary_distributions()
```

(continues on next page)

(continued from previous page)

```
# Mean vector of state for the savings problem
mxo = np.vstack([μ_z, 0.0])

# Create stationary covariance matrix of x -- start everyone off at b=0
a1 = np.zeros((3, 1))
aa = np.hstack([Σ_z, a1])
bb = np.zeros((1, 4))
sxo = np.vstack([aa, bb])

# These choices will initialize the state vector of an individual at zero
# debt and the ergodic distribution of the endowment process. Use these to
# create the Bewley economy.
mxbewley = mxo
sxbewley = sxo
```

The next step is to create the matrices for the LQ system

```
A12 = np.zeros((3,1))
ALQ_l = np.hstack([A, A12])
ALQ_r = np.array([[0, -R, 0, R]])
ALQ = np.vstack([ALQ_l, ALQ_r])

RLQ = np.array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 1e-9]])

QLQ = np.array([1.0])
BLQ = np.array([0., 0., 0., R]).reshape(4,1)
CLQ = np.array([0., σ, 0., 0.]).reshape(4,1)
β_LQ = β
```

Let's print these out and have a look at them

```
print(f"A = \n {ALQ}")
print(f"B = \n {BLQ}")
print(f"R = \n {RLQ}")
print(f"Q = \n {QLQ}")
```

```
A =
 [[ 1.          0.          0.          0.        ]
 [10.          0.9         0.          0.        ]
 [ 0.          1.          0.          0.        ]
 [ 0.         -1.05263158  0.          1.05263158]]
B =
 [[0.        ]
 [0.        ]
 [0.        ]
 [1.05263158]]
R =
 [[0.e+00 0.e+00 0.e+00 0.e+00]
 [0.e+00 0.e+00 0.e+00 0.e+00]
 [0.e+00 0.e+00 0.e+00 0.e+00]
 [0.e+00 0.e+00 0.e+00 1.e-09]]
Q =
 [1.]
```

Now create the appropriate instance of an LQ model

```
lqpi = qe.LQ(QLQ, RLQ, ALQ, BLQ, C=CLQ, beta=β_LQ)
```

We'll save the implied optimal policy function soon compare them with what we get by employing an alternative solution method

```
P, F, d = lqpi.stationary_values()  # Compute value function and decision rule
ABF = ALQ - BLQ @ F  #  Form closed loop system
```

50.4.2 Comparison with the Difference Equation Approach

In our *first lecture* on the infinite horizon permanent income problem we used a different solution method.

The method was based around

- deducing the Euler equations that are the first-order conditions with respect to consumption and savings.
- using the budget constraints and boundary condition to complete a system of expectational linear difference equations.
- solving those equations to obtain the solution.

Expressed in state space notation, the solution took the form

$$
\begin{aligned}
z_{t+1} &= Az_t + Cw_{t+1} \\
b_{t+1} &= b_t + U[(I - \beta A)^{-1}(A - I)]z_t \\
y_t &= Uz_t \\
c_t &= (1 - \beta)[U(I - \beta A)^{-1}z_t - b_t]
\end{aligned}
$$

Now we'll apply the formulas in this system

```
# Use the above formulas to create the optimal policies for b_{t+1} and c_t
b_pol = G @ la.inv(np.eye(3, 3) - β * A) @ (A - np.eye(3, 3))
c_pol = (1 - β) * G @ la.inv(np.eye(3, 3) - β * A)

# Create the A matrix for a LinearStateSpace instance
A_LSS1 = np.vstack([A, b_pol])
A_LSS2 = np.eye(4, 1, -3)
A_LSS = np.hstack([A_LSS1, A_LSS2])

# Create the C matrix for LSS methods
C_LSS = np.vstack([C, np.zeros(1)])

# Create the G matrix for LSS methods
G_LSS1 = np.vstack([G, c_pol])
G_LSS2 = np.vstack([np.zeros(1), -(1 - β)])
G_LSS = np.hstack([G_LSS1, G_LSS2])

# Use the following values to start everyone off at b=0, initial incomes zero
μ_0 = np.array([1., 0., 0., 0.])
Σ_0 = np.zeros((4, 4))
```

`A_LSS` calculated as we have here should equal `ABF` calculated above using the LQ model

```
ABF - A_LSS
```

```
array([[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00],
       [-9.51248178e-06,  9.51247915e-08,  0.00000000e+00,
        -1.99999923e-08]])
```

Now compare pertinent elements of `c_pol` and `F`

```
print(c_pol, "\n", -F)
```

```
[[65.51724138  0.34482759  0.        ]]
 [[ 6.55172323e+01  3.44827677e-01 -0.00000000e+00 -5.00000190e-02]]
```

We have verified that the two methods give the same solution.

Now let's create instances of the LinearStateSpace class and use it to do some interesting experiments.

To do this, we'll use the outcomes from our second method.

50.5 Two Example Economies

In the spirit of Bewley models [Bew86], we'll generate panels of consumers.

The examples differ only in the initial states with which we endow the consumers.

All other parameter values are kept the same in the two examples

- In the first example, all consumers begin with zero nonfinancial income and zero debt.
  - The consumers are thus *ex-ante* identical.
- In the second example, while all begin with zero debt, we draw their initial income levels from the invariant distribution of financial income.
  - Consumers are *ex-ante* heterogeneous.

In the first example, consumers' nonfinancial income paths display pronounced transients early in the sample

- these will affect outcomes in striking ways

Those transient effects will not be present in the second example.

We use methods affiliated with the LinearStateSpace class to simulate the model.

50.5.1 First Set of Initial Conditions

We generate 25 paths of the exogenous non-financial income process and the associated optimal consumption and debt paths.

In the first set of graphs, darker lines depict a particular sample path, while the lighter lines describe 24 other paths.

A second graph plots a collection of simulations against the population distribution that we extract from the `LinearStateSpace` instance `LSS`.

Comparing sample paths with population distributions at each date t is a useful exercise—see *our discussion* of the laws of large numbers

```
lss = qe.LinearStateSpace(A_LSS, C_LSS, G_LSS, mu_0=μ_0, Sigma_0=Σ_0)
```

50.5.2 Population and Sample Panels

In the code below, we use the LinearStateSpace class to

- compute and plot population quantiles of the distributions of consumption and debt for a population of consumers.
- simulate a group of 25 consumers and plot sample paths on the same graph as the population distribution.

```
def income_consumption_debt_series(A, C, G, μ_0, Σ_0, T=150, npaths=25):
    """
    This function takes initial conditions (μ_0, Σ_0) and uses the
    LinearStateSpace class from QuantEcon to  simulate an economy
    npaths times for T periods. It then uses that information to
    generate some graphs related to the discussion below.
    """
    lss = qe.LinearStateSpace(A, C, G, mu_0=μ_0, Sigma_0=Σ_0)

    # Simulation/Moment Parameters
    moment_generator = lss.moment_sequence()

    # Simulate various paths
    bsim = np.empty((npaths, T))
    csim = np.empty((npaths, T))
    ysim = np.empty((npaths, T))

    for i in range(npaths):
        sims = lss.simulate(T)
        bsim[i, :] = sims[0][-1, :]
        csim[i, :] = sims[1][1, :]
        ysim[i, :] = sims[1][0, :]

    # Get the moments
    cons_mean = np.empty(T)
    cons_var = np.empty(T)
    debt_mean = np.empty(T)
    debt_var = np.empty(T)
    for t in range(T):
        μ_x, μ_y, Σ_x, Σ_y = next(moment_generator)
        cons_mean[t], cons_var[t] = μ_y[1], Σ_y[1, 1]
        debt_mean[t], debt_var[t] = μ_x[3], Σ_x[3, 3]

    return bsim, csim, ysim, cons_mean, cons_var, debt_mean, debt_var

def consumption_income_debt_figure(bsim, csim, ysim):

    # Get T
    T =  bsim.shape[1]

    # Create the first figure
    fig, ax = plt.subplots(2, 1, figsize=(10, 8))
    xvals = np.arange(T)

    # Plot consumption and income
    ax[0].plot(csim[0, :], label="c", color="b")
```

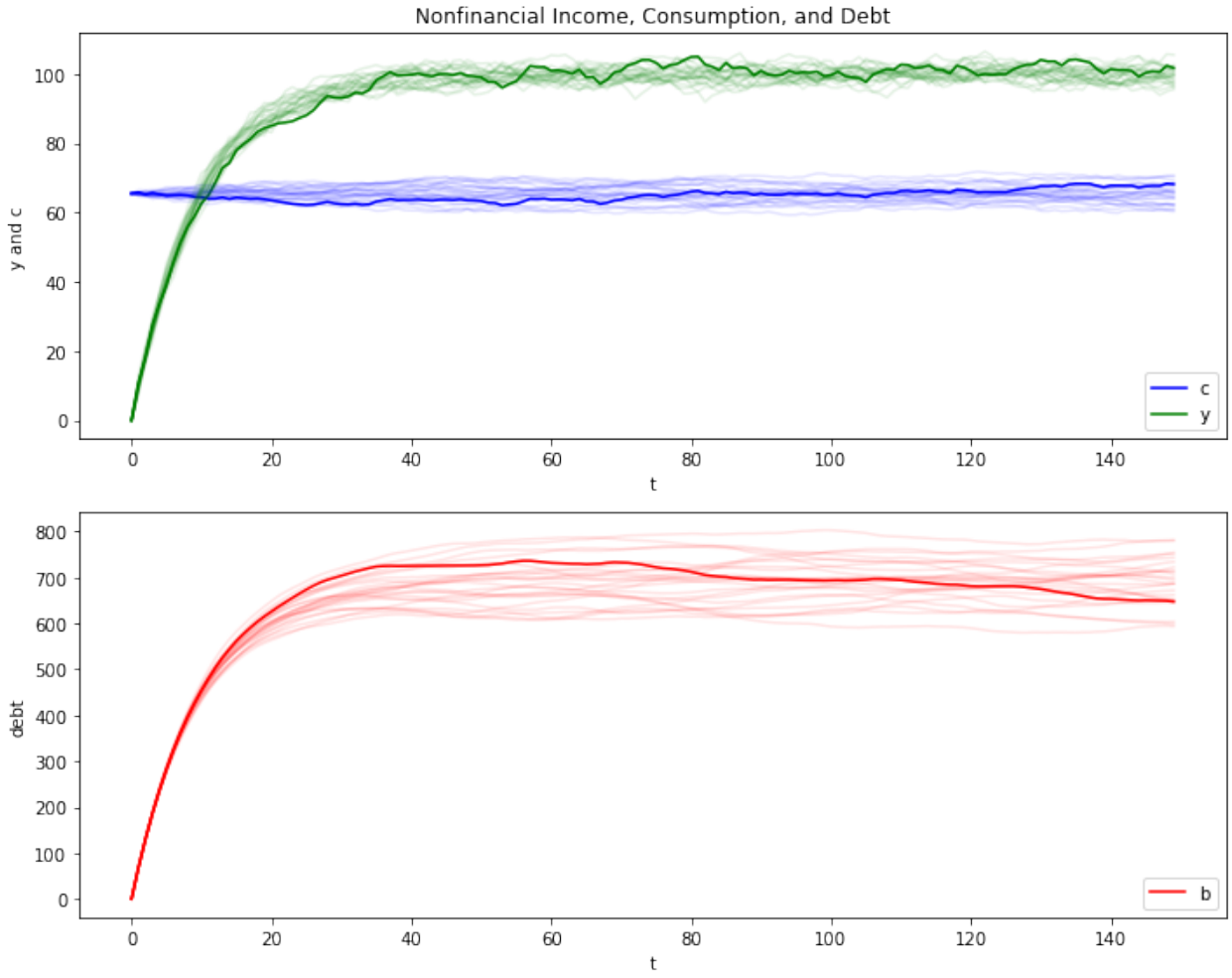
(continues on next page)

(continued from previous page)

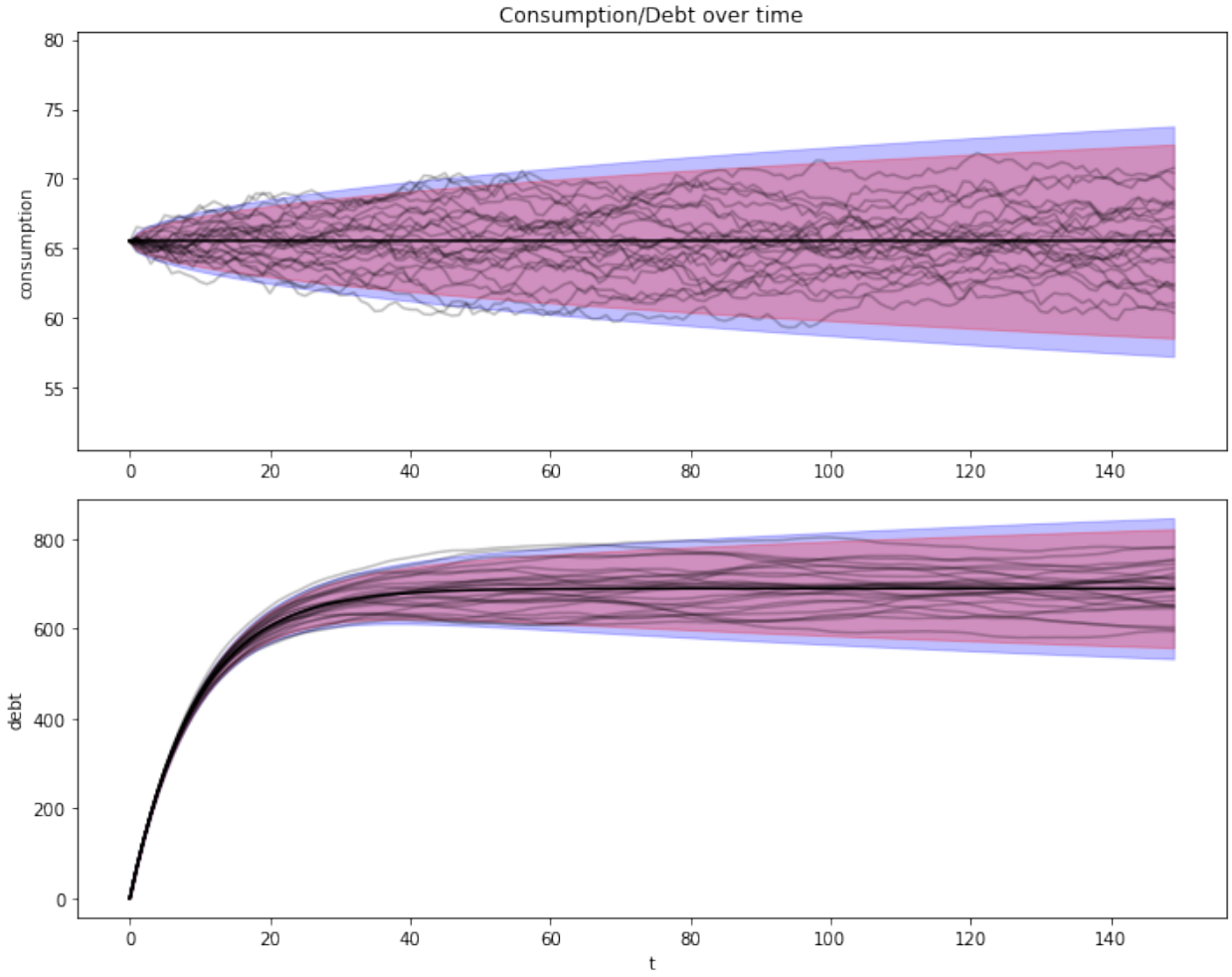
```
    ax[0].plot(ysim[0, :], label="y", color="g")
    ax[0].plot(csim.T, alpha=.1, color="b")
    ax[0].plot(ysim.T, alpha=.1, color="g")
    ax[0].legend(loc=4)
    ax[0].set(title="Nonfinancial Income, Consumption, and Debt",
              xlabel="t", ylabel="y and c")

    # Plot debt
    ax[1].plot(bsim[0, :], label="b", color="r")
    ax[1].plot(bsim.T, alpha=.1, color="r")
    ax[1].legend(loc=4)
    ax[1].set(xlabel="t", ylabel="debt")

    fig.tight_layout()
    return fig

def consumption_debt_fanchart(csim, cons_mean, cons_var,
                              bsim, debt_mean, debt_var):
    # Get T
    T =  bsim.shape[1]

    # Create percentiles of cross-section distributions
    cmean = np.mean(cons_mean)
    c90 = 1.65 * np.sqrt(cons_var)
    c95 = 1.96 * np.sqrt(cons_var)
    c_perc_95p, c_perc_95m = cons_mean + c95, cons_mean - c95
    c_perc_90p, c_perc_90m = cons_mean + c90, cons_mean - c90

    # Create percentiles of cross-section distributions
    dmean = np.mean(debt_mean)
    d90 = 1.65 * np.sqrt(debt_var)
    d95 = 1.96 * np.sqrt(debt_var)
    d_perc_95p, d_perc_95m = debt_mean + d95, debt_mean - d95
    d_perc_90p, d_perc_90m = debt_mean + d90, debt_mean - d90


    # Create second figure
    fig, ax = plt.subplots(2, 1, figsize=(10, 8))
    xvals = np.arange(T)

    # Consumption fan
    ax[0].plot(xvals, cons_mean, color="k")
    ax[0].plot(csim.T, color="k", alpha=.25)
    ax[0].fill_between(xvals, c_perc_95m, c_perc_95p, alpha=.25, color="b")
    ax[0].fill_between(xvals, c_perc_90m, c_perc_90p, alpha=.25, color="r")
    ax[0].set(title="Consumption/Debt over time",
              ylim=(cmean-15, cmean+15), ylabel="consumption")

    # Debt fan
    ax[1].plot(xvals, debt_mean, color="k")
    ax[1].plot(bsim.T, color="k", alpha=.25)
    ax[1].fill_between(xvals, d_perc_95m, d_perc_95p, alpha=.25, color="b")
    ax[1].fill_between(xvals, d_perc_90m, d_perc_90p, alpha=.25, color="r")
    ax[1].set(xlabel="t", ylabel="debt")

    fig.tight_layout()
    return fig
```

Now let's create figures with initial conditions of zero for y_0 and b_0

```
out = income_consumption_debt_series(A_LSS, C_LSS, G_LSS, μ_0, Σ_0)
bsim0, csim0, ysim0 = out[:3]
cons_mean0, cons_var0, debt_mean0, debt_var0 = out[3:]

consumption_income_debt_figure(bsim0, csim0, ysim0)

plt.show()
```




```
consumption_debt_fanchart(csim0, cons_mean0, cons_var0,
                          bsim0, debt_mean0, debt_var0)

plt.show()
```



Here is what is going on in the above graphs.

For our simulation, we have set initial conditions $b_0 = y_{-1} = y_{-2} = 0$.

Because $y_{-1} = y_{-2} = 0$, nonfinancial income y_t starts far below its stationary mean $\mu_{y,\infty}$ and rises early in each simulation.

Recall from the *previous lecture* that we can represent the optimal decision rule for consumption in terms of the **cointegrating relationship**

$$(1-\beta)b_t + c_t = (1-\beta)E_t \sum_{j=0}^{\infty} \beta^j y_{t+j} \tag{6}$$

So at time 0 we have

$$c_0 = (1-\beta)E_0 \sum_{t=0}^{\infty} \beta^j y_t$$

This tells us that consumption starts at the income that would be paid by an annuity whose value equals the expected discounted value of nonfinancial income at time $t = 0$.

To support that level of consumption, the consumer borrows a lot early and consequently builds up substantial debt.

In fact, he or she incurs so much debt that eventually, in the stochastic steady state, he consumes less each period than his nonfinancial income.

He uses the gap between consumption and nonfinancial income mostly to service the interest payments due on his debt.

Thus, when we look at the panel of debt in the accompanying graph, we see that this is a group of *ex-ante* identical people each of whom starts with zero debt.

All of them accumulate debt in anticipation of rising nonfinancial income.

They expect their nonfinancial income to rise toward the invariant distribution of income, a consequence of our having started them at $y_{-1} = y_{-2} = 0$.

Cointegration Residual

The following figure plots realizations of the left side of (6), which, *as discussed in our last lecture*, is called the **cointegrating residual**.

As mentioned above, the right side can be thought of as an annuity payment on the expected present value of future income $E_t \sum_{j=0}^{\infty} \beta^j y_{t+j}$.

Early along a realization, c_t is approximately constant while $(1-\beta)b_t$ and $(1-\beta)E_t \sum_{j=0}^{\infty} \beta^j y_{t+j}$ both rise markedly as the household's present value of income and borrowing rise pretty much together.

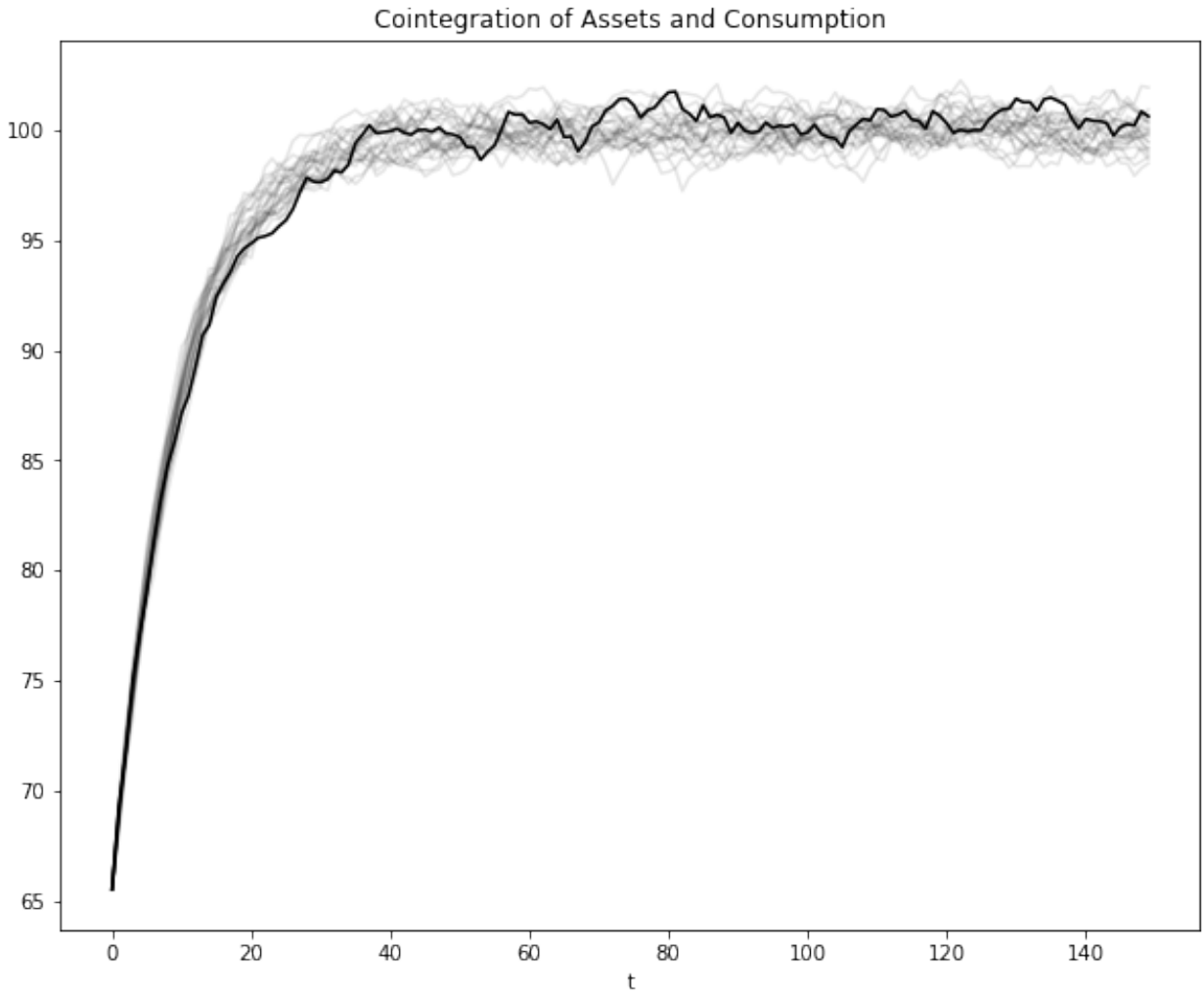
This example illustrates the following point: the definition of cointegration implies that the cointegrating residual is *asymptotically* covariance stationary, not *covariance stationary*.

The cointegrating residual for the specification with zero income and zero debt initially has a notable transient component that dominates its behavior early in the sample.

By altering initial conditions, we shall remove this transient in our second example to be presented below

```
def cointegration_figure(bsim, csim):
    """
    Plots the cointegration
    """
    # Create figure
    fig, ax = plt.subplots(figsize=(10, 8))
    ax.plot((1 - β) * bsim[0, :] + csim[0, :], color="k")
    ax.plot((1 - β) * bsim.T + csim.T, color="k", alpha=.1)

    ax.set(title="Cointegration of Assets and Consumption", xlabel="t")

    return fig
```

```
cointegration_figure(bsim0, csim0)
plt.show()
```



50.5.3 A "Borrowers and Lenders" Closed Economy

When we set $y_{-1} = y_{-2} = 0$ and $b_0 = 0$ in the preceding exercise, we make debt "head north" early in the sample.

Average debt in the cross-section rises and approaches the asymptote.

We can regard these as outcomes of a "small open economy" that borrows from abroad at the fixed gross interest rate $R = r + 1$ in anticipation of rising incomes.

So with the economic primitives set as above, the economy converges to a steady state in which there is an excess aggregate supply of risk-free loans at a gross interest rate of R.

This excess supply is filled by "foreigner lenders" willing to make those loans.

We can use virtually the same code to rig a "poor man's Bewley [Bew86] model" in the following way

- as before, we start everyone at $b_0 = 0$.
- But instead of starting everyone at $y_{-1} = y_{-2} = 0$, we draw $\begin{bmatrix} y_{-1} \\ y_{-2} \end{bmatrix}$ from the invariant distribution of the $\{y_t\}$ process.

This rigs a closed economy in which people are borrowing and lending with each other at a gross risk-free interest rate of $R = \beta^{-1}$.

Across the group of people being analyzed, risk-free loans are in zero excess supply.

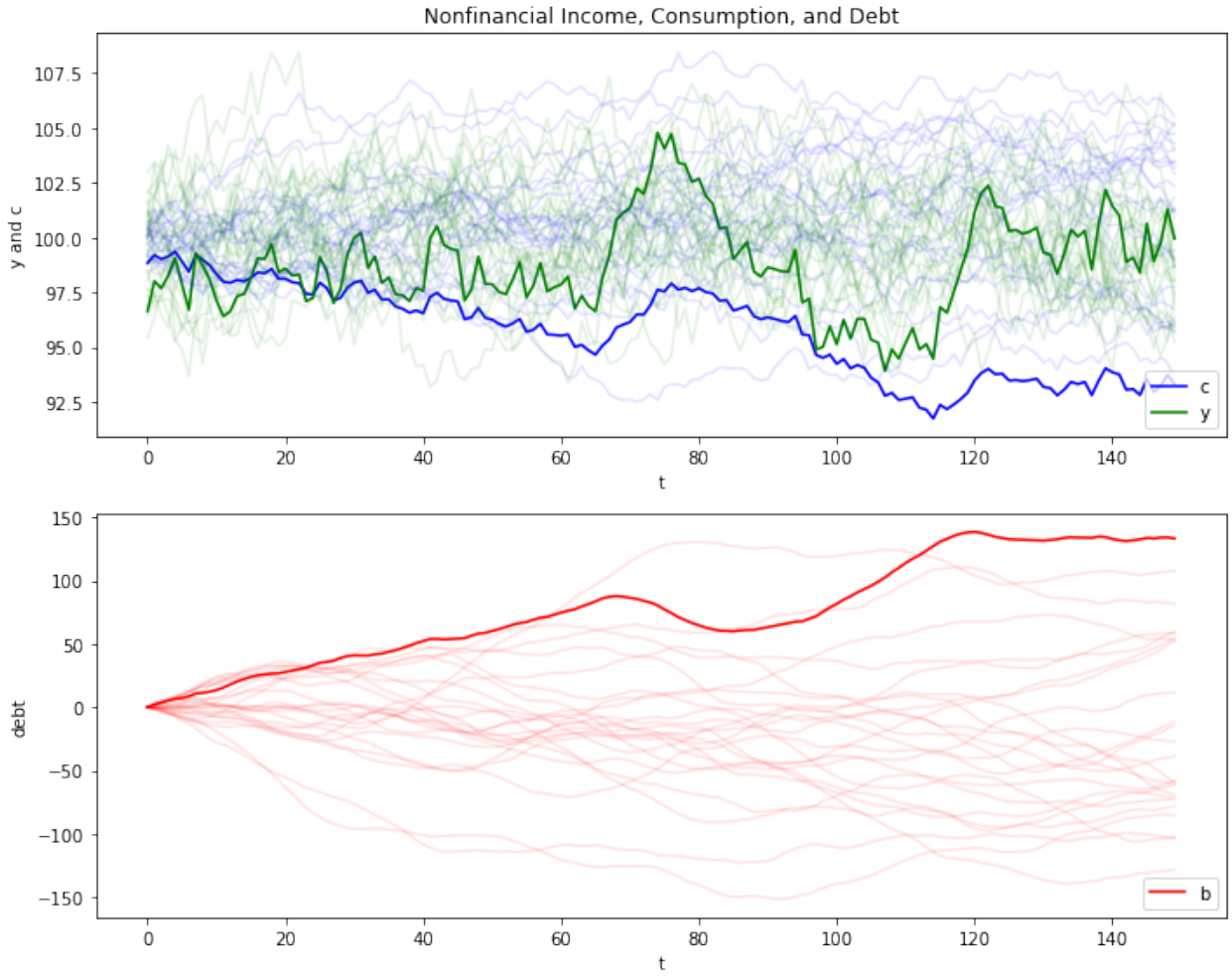
We have arranged primitives so that $R = \beta^{-1}$ clears the market for risk-free loans at zero aggregate excess supply.

So the risk-free loans are being made from one person to another within our closed set of agents.

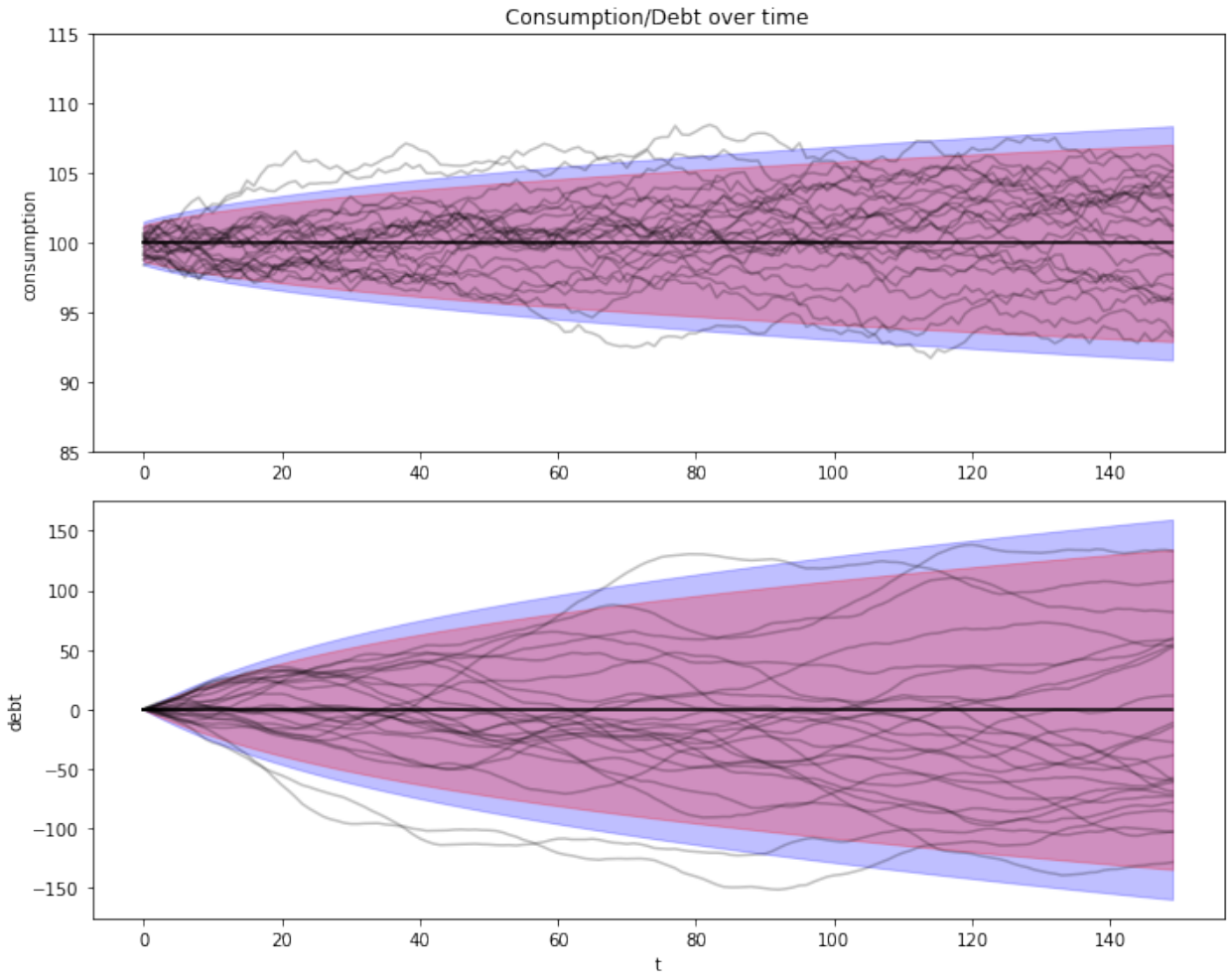
There is no need for foreigners to lend to our group.

Let's have a look at the corresponding figures

```
out = income_consumption_debt_series(A_LSS, C_LSS, G_LSS, mxbewley, sxbewley)
bsimb, csimb, ysimb = out[:3]
cons_meanb, cons_varb, debt_meanb, debt_varb = out[3:]

consumption_income_debt_figure(bsimb, csimb, ysimb)

plt.show()
```




```
consumption_debt_fanchart(csimb, cons_meanb, cons_varb,
                          bsimb, debt_meanb, debt_varb)

plt.show()
```



The graphs confirm the following outcomes:

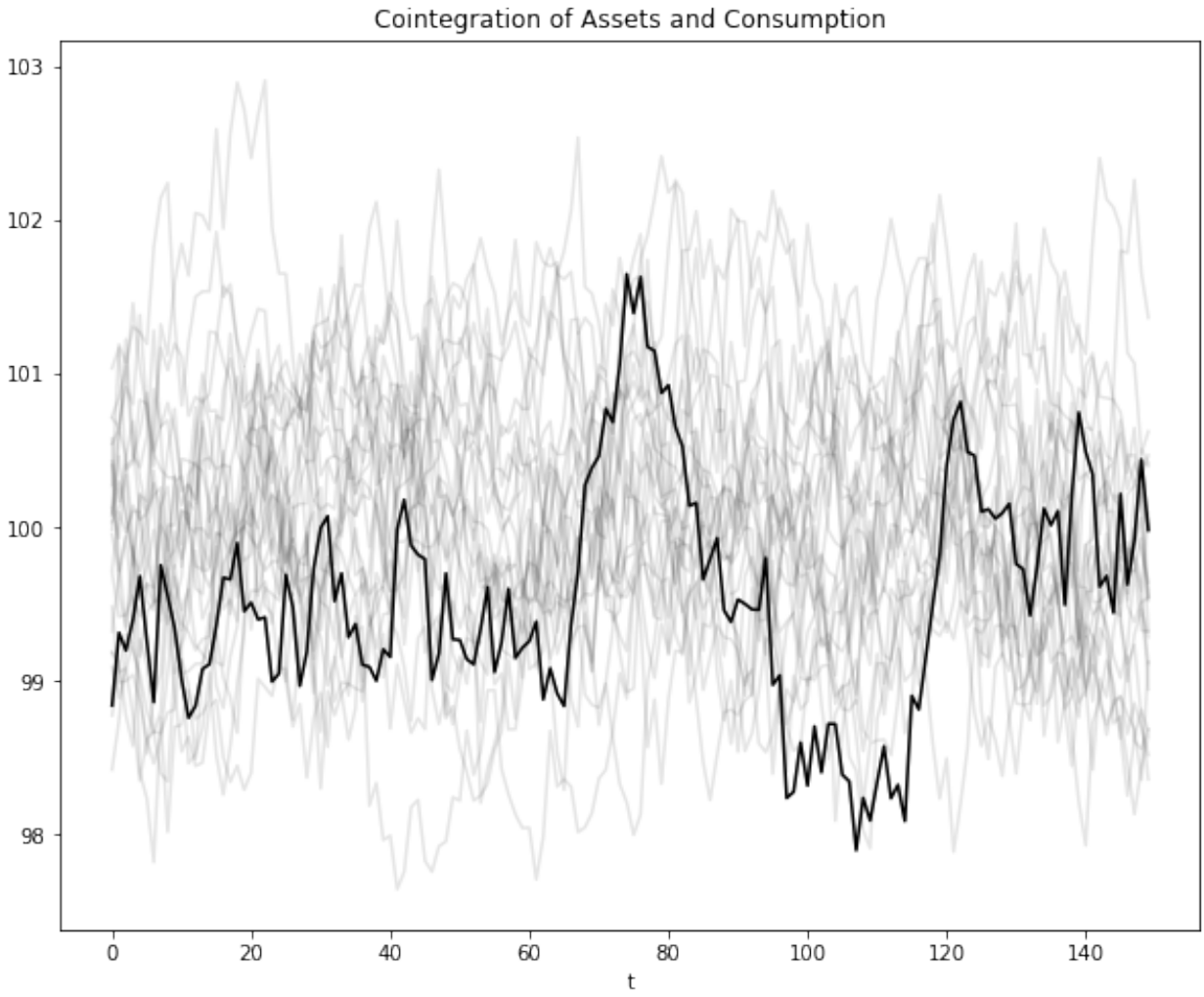
- As before, the consumption distribution spreads out over time.

But now there is some initial dispersion because there is *ex-ante* heterogeneity in the initial draws of $\begin{bmatrix} y_{-1} \\ y_{-2} \end{bmatrix}$.

- As before, the cross-section distribution of debt spreads out over time.
- Unlike before, the average level of debt stays at zero, confirming that this is a closed borrower-and-lender economy.
- Now the cointegrating residual seems stationary, and not just asymptotically stationary.

Let's have a look at the cointegration figure

```
cointegration_figure(bsimb, csimb)
plt.show()
```


Cointegration of Assets and Consumption
103
102
101
100
99
98
0
20
40
60
80
100
120
140
t

CHAPTER

FIFTYONE

PRODUCTION SMOOTHING VIA INVENTORIES

Contents

In addition to what's in Anaconda, this lecture employs the following library:

```
!conda install -y quantecon
```

51.1 Overview

This lecture can be viewed as an application of this *quantecon lecture* about linear quadratic control theory.

It formulates a discounted dynamic program for a firm that chooses a production schedule to balance

- minimizing costs of production across time, against
- keeping costs of holding inventories low

In the tradition of a classic book by Holt, Modigliani, Muth, and Simon [HMMS60], we simplify the firm's problem by formulating it as a linear quadratic discounted dynamic programming problem of the type studied in this *quantecon lecture*.

Because its costs of production are increasing and quadratic in production, the firm holds inventories as a buffer stock in order to smooth production across time, provided that holding inventories is not too costly.

But the firm also wants to make its sales out of existing inventories, a preference that we represent by a cost that is quadratic in the difference between sales in a period and the firm's beginning of period inventories.

We compute examples designed to indicate how the firm optimally smooths production while keeping inventories close to sales.

To introduce components of the model, let

- S_t be sales at time t
- Q_t be production at time t
- I_t be inventories at the beginning of time t
- $\beta \in (0,1)$ be a discount factor
- $c(Q_t) = c_1 Q_t + c_2 Q_t^2$, be a cost of production function, where $c_1 > 0, c_2 > 0$, be an inventory cost function
- $d(I_t, S_t) = d_1 I_t + d_2 (S_t - I_t)^2$, where $d_1 > 0, d_2 > 0$, be a cost-of-holding-inventories function, consisting of two components:
  - a cost $d_1 I_t$ of carrying inventories, and
  - a cost $d_2 (S_t - I_t)^2$ of having inventories deviate from sales
- $p_t = a_0 - a_1 S_t + v_t$ be an inverse demand function for a firm's product, where $a_0 > 0, a_1 > 0$ and v_t is a demand shock at time t
- $\pi\_t = p_t S_t - c(Q_t) - d(I_t, S_t)$ be the firm's profits at time t
- $\sum_{t=0}^{\infty} \beta^t \pi_t$ be the present value of the firm's profits at time 0
- $I_{t+1} = I_t + Q_t - S_t$ be the law of motion of inventories
- $z_{t+1} = A_{22} z_t + C_2 \epsilon_{t+1}$ be a law of motion for an exogenous state vector z_t that contains time t information useful for predicting the demand shock v_t
- $v_t = G z_t$ link the demand shock to the information set z_t
- the constant 1 be the first component of z_t

To map our problem into a linear-quadratic discounted dynamic programming problem (also known as an optimal linear regulator), we define the **state** vector at time t as

$$x_t = \begin{bmatrix} I_t \\ z_t \end{bmatrix}$$

and the **control** vector as

$$u_t = \begin{bmatrix} Q_t \\ S_t \end{bmatrix}$$

The law of motion for the state vector x_t is evidently

$$\begin{bmatrix} I_{t+1} \\ z_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} I_t \\ z_t \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_t \\ S_t \end{bmatrix} + \begin{bmatrix} 0 \\ C_2 \end{bmatrix} \epsilon_{t+1}$$

or

$$x_{t+1} = A x_t + B u_t + C \epsilon_{t+1}$$

(At this point, we ask that you please forgive us for using Q_t to be the firm's production at time t, while below we use Q as the matrix in the quadratic form $u_t' Q u_t$ that appears in the firm's one-period profit function)

We can express the firm's profit as a function of states and controls as

$$\pi_t = -(x_t' R x_t + u_t' Q u_t + 2 u_t' N x_t)$$

To form the matrices R, Q, N in an LQ dynamic programming problem, we note that the firm's profits at time t function can be expressed

$$
\begin{aligned}
\pi_t =& p_t S_t - c\left(Q_t\right) - d\left(I_t, S_t\right) \\
=& \left(a_0 - a_1 S_t + v_t\right) S_t - c_1 Q_t - c_2 Q_t^2 - d_1 I_t - d_2 \left(S_t - I_t\right)^2 \\
=& a_0 S_t - a_1 S_t^2 + G z_t S_t - c_1 Q_t - c_2 Q_t^2 - d_1 I_t - d_2 S_t^2 - d_2 I_t^2 + 2 d_2 S_t I_t \\
=& - \left(\underbrace{d_1 I_t + d_2 I_t^2}_{x_t' R x_t} + \underbrace{a_1 S_t^2 + d_2 S_t^2 + c_2 Q_t^2}_{u_t' Q u_t} \underbrace{- a_0 S_t - G z_t S_t + c_1 Q_t - 2 d_2 S_t I_t}_{2 u_t' N x_t} \right) \\
=& - \Bigg(\begin{bmatrix} I_t & z_t' \end{bmatrix} \underbrace{\begin{bmatrix} d_2 & \frac{d_1}{2} S_c \\ \frac{d_1}{2} S_c' & 0 \end{bmatrix}}_{\equiv R} \begin{bmatrix} I_t \\ z_t \end{bmatrix} + \begin{bmatrix} Q_t & S_t \end{bmatrix} \underbrace{\begin{bmatrix} c_2 & 0 \\ 0 & a_1 + d_2 \end{bmatrix}}_{\equiv Q} \begin{bmatrix} Q_t \\ S_t \end{bmatrix} + 2 \begin{bmatrix} Q_t & S_t \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \frac{c_1}{2} S_c \\ -d_2 & -\frac{a_0}{2} S_c - \frac{G}{2} \end{bmatrix}}_{\equiv N} \begin{bmatrix} I \\ z \end{bmatrix}
\end{aligned}
$$

where $S_c = [1, 0]$.

Remark on notation: The notation for cross product term in the QuantEcon library is N.

The firms' optimum decision rule takes the form

$$u_t = -F x_t$$

and the evolution of the state under the optimal decision rule is

$$x_{t+1} = (A - BF) x_t + C \epsilon_{t+1}$$

The firm chooses a decision rule for u_t that maximizes

$$E_0 \sum_{t=0}^{\infty} \beta^t \pi_t$$

subject to a given x_0.

This is a stochastic discounted LQ dynamic program.

Here is code for computing an optimal decision rule and for analyzing its consequences.

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (11, 5)  #set default figure size
import numpy as np
import quantecon as qe
```

```
class SmoothingExample:
    """
    Class for constructing, solving, and plotting results for
    inventories and sales smoothing problem.
    """

    def __init__(self,
                 β=0.96,            # Discount factor
                 c1=1,              # Cost-of-production
                 c2=1,
                 d1=1,              # Cost-of-holding inventories
                 d2=1,
                 a0=10,             # Inverse demand function
```

(continues on next page)

(continued from previous page)

```
                a1=1,
                A22=[[1,   0],     # z process
                     [1, 0.9]],
                C2=[[0], [1]],
                G=[0, 1]):

        self.β = β
        self.c1, self.c2 = c1, c2
        self.d1, self.d2 = d1, d2
        self.a0, self.a1 = a0, a1
        self.A22 = np.atleast_2d(A22)
        self.C2 = np.atleast_2d(C2)
        self.G = np.atleast_2d(G)

        # Dimensions
        k, j = self.C2.shape         # Dimensions for randomness part
        n = k + 1                    # Number of states
        m = 2                        # Number of controls

        Sc = np.zeros(k)
        Sc[0] = 1

        # Construct matrices of transition law
        A = np.zeros((n, n))
        A[0, 0] = 1
        A[1:, 1:] = self.A22

        B = np.zeros((n, m))
        B[0, :] = 1, -1

        C = np.zeros((n, j))
        C[1:, :] = self.C2

        self.A, self.B, self.C = A, B, C

        # Construct matrices of one period profit function
        R = np.zeros((n, n))
        R[0, 0] = d2
        R[1:, 0] = d1 / 2 * Sc
        R[0, 1:] = d1 / 2 * Sc

        Q = np.zeros((m, m))
        Q[0, 0] = c2
        Q[1, 1] = a1 + d2

        N = np.zeros((m, n))
        N[1, 0] = - d2
        N[0, 1:] = c1 / 2 * Sc
        N[1, 1:] = - a0 / 2 * Sc - self.G / 2

        self.R, self.Q, self.N = R, Q, N

        # Construct LQ instance
        self.LQ = qe.LQ(Q, R, A, B, C, N, beta=β)
        self.LQ.stationary_values()

    def simulate(self, x0, T=100):
```

(continues on next page)

(continued from previous page)

```
        c1, c2 = self.c1, self.c2
        d1, d2 = self.d1, self.d2
        a0, a1 = self.a0, self.a1
        G = self.G

        x_path, u_path, w_path = self.LQ.compute_sequence(x0, ts_length=T)

        I_path = x_path[0, :-1]
        z_path = x_path[1:, :-1]
        □_path = (G @ z_path)[0, :]

        Q_path = u_path[0, :]
        S_path = u_path[1, :]

        revenue = (a0 - a1 * S_path + □_path) * S_path
        cost_production = c1 * Q_path + c2 * Q_path ** 2
        cost_inventories = d1 * I_path + d2 * (S_path - I_path) ** 2

        Q_no_inventory = (a0 + □_path - c1) / (2 * (a1 + c2))
        Q_hardwired = (a0 + □_path - c1) / (2 * (a1 + c2 + d2))

        fig, ax = plt.subplots(2, 2, figsize=(15, 10))

        ax[0, 0].plot(range(T), I_path, label="inventories")
        ax[0, 0].plot(range(T), S_path, label="sales")
        ax[0, 0].plot(range(T), Q_path, label="production")
        ax[0, 0].legend(loc=1)
        ax[0, 0].set_title("inventories, sales, and production")

        ax[0, 1].plot(range(T), (Q_path - S_path), color='b')
        ax[0, 1].set_ylabel("change in inventories", color='b')
        span = max(abs(Q_path - S_path))
        ax[0, 1].set_ylim(0-span*1.1, 0+span*1.1)
        ax[0, 1].set_title("demand shock and change in inventories")

        ax1_ = ax[0, 1].twinx()
        ax1_.plot(range(T), □_path, color='r')
        ax1_.set_ylabel("demand shock", color='r')
        span = max(abs(□_path))
        ax1_.set_ylim(0-span*1.1, 0+span*1.1)

        ax1_.plot([0, T], [0, 0], '--', color='k')

        ax[1, 0].plot(range(T), revenue, label="revenue")
        ax[1, 0].plot(range(T), cost_production, label="cost_production")
        ax[1, 0].plot(range(T), cost_inventories, label="cost_inventories")
        ax[1, 0].legend(loc=1)
        ax[1, 0].set_title("profits decomposition")

        ax[1, 1].plot(range(T), Q_path, label="production")
        ax[1, 1].plot(range(T), Q_hardwired, label='production when  $I_t$ \
            forced to be zero')
        ax[1, 1].plot(range(T), Q_no_inventory, label='production when \
            inventories not useful')
        ax[1, 1].legend(loc=1)
        ax[1, 1].set_title('three production concepts')
```

(continues on next page)

(continued from previous page)

```
        plt.show()
```

Notice that the above code sets parameters at the following default values

- discount factor $\beta = 0.96$,
- inverse demand function: $a0 = 10, a1 = 1$
- cost of production $c1 = 1, c2 = 1$
- costs of holding inventories $d1 = 1, d2 = 1$

In the examples below, we alter some or all of these parameter values.

51.2 Example 1

In this example, the demand shock follows AR(1) process:

$$\nu_t = \alpha + \rho \nu_{t-1} + \epsilon_t,$$

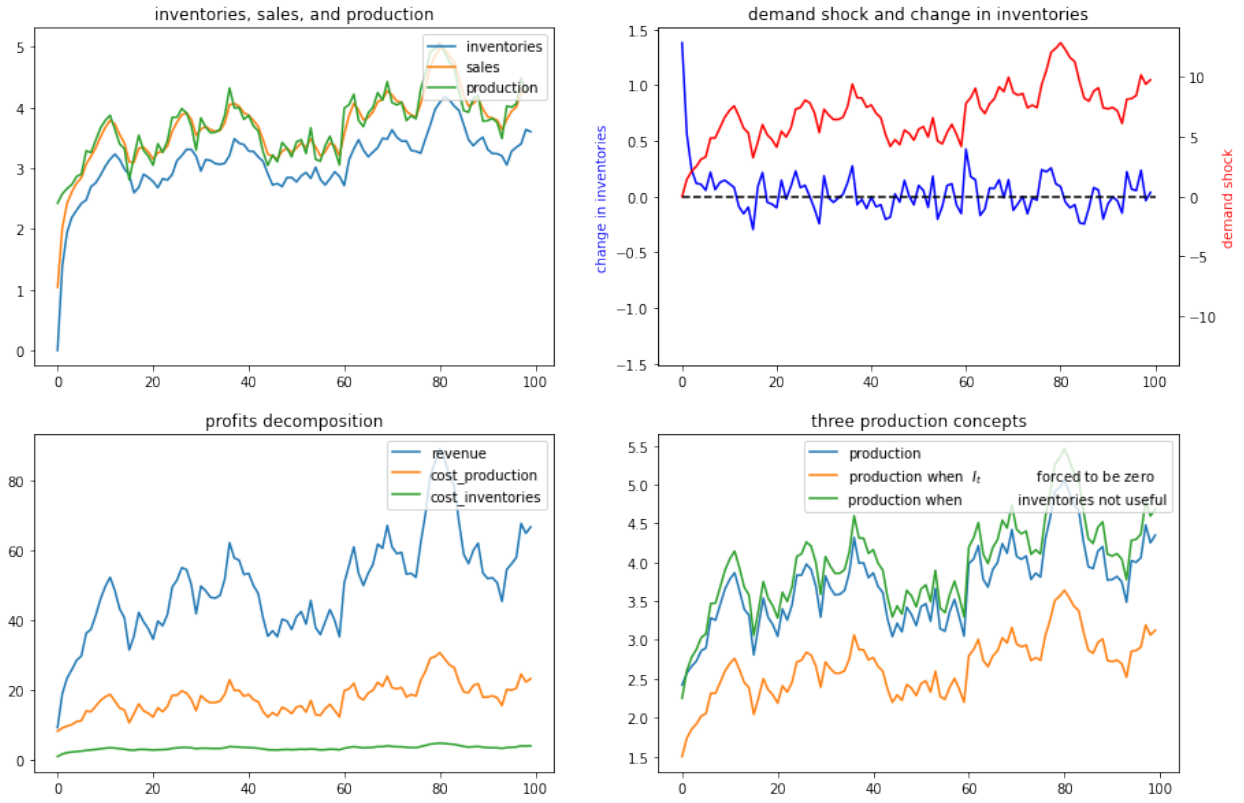
which implies

$$z_{t+1} = \begin{bmatrix} 1 \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \alpha & \rho \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ v_t \end{bmatrix}}_{z_t} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \epsilon_{t+1}.$$

We set $\alpha = 1$ and $\rho = 0.9$, their default values.

We'll calculate and display outcomes, then discuss them below the pertinent figures.

```
ex1 = SmoothingExample()

x0 = [0, 1, 0]
ex1.simulate(x0)
```



The figures above illustrate various features of an optimal production plan.

Starting from zero inventories, the firm builds up a stock of inventories and uses them to smooth costly production in the face of demand shocks.

Optimal decisions evidently respond to demand shocks.

Inventories are always less than sales, so some sales come from current production, a consequence of the cost, $d_1 I_t$ of holding inventories.

The lower right panel shows differences between optimal production and two alternative production concepts that come from altering the firm's cost structure – i.e., its technology.

These two concepts correspond to these distinct altered firm problems.

- a setting in which inventories are not needed
- a setting in which they are needed but we arbitrarily prevent the firm from holding inventories by forcing it to set $I_t = 0$ always

We use these two alternative production concepts in order to shed light on the baseline model.

51.3 Inventories Not Useful

Let's turn first to the setting in which inventories aren't needed.

In this problem, the firm forms an output plan that maximizes the expected value of

$$\sum_{t=0}^{\infty} \beta^t \{p_t Q_t - C(Q_t)\}$$

It turns out that the optimal plan for Q_t for this problem also solves a sequence of static problems $\max_{Q_t} \{p_t Q_t - c(Q_t)\}$.

When inventories aren't required or used, sales always equal production.

This simplifies the problem and the optimal no-inventory production maximizes the expected value of

$$\sum_{t=0}^{\infty} \beta^t \left\{p_t Q_t - C\left(Q_t\right)\right\}.$$

The optimum decision rule is

$$Q_t^{ni} = \frac{a_0 + \nu_t - c_1}{c_2 + a_1}.$$

51.4 Inventories Useful but are Hardwired to be Zero Always

Next, we turn to a distinct problem in which inventories are useful – meaning that there are costs of $d_2(I_t - S_t)^2$ associated with having sales not equal to inventories – but we arbitrarily impose on the firm the costly restriction that it never hold inventories.

Here the firm's maximization problem is

$$\max_{\{I_t, Q_t, S_t\}} \sum_{t=0}^{\infty} \beta^t \left\{p_t S_t - C\left(Q_t\right) - d\left(I_t, S_t\right)\right\}$$

subject to the restrictions that $I_t = 0$ for all t and that $I_{t+1} = I_t + Q_t - S_t$.

The restriction that $I_t = 0$ implies that $Q_t = S_t$ and that the maximization problem reduces to

$$\max_{Q_t} \sum_{t=0}^{\infty} \beta^t \left\{p_t Q_t - C\left(Q_t\right) - d\left(0, Q_t\right)\right\}$$

Here the optimal production plan is

$$Q_t^h = \frac{a_0 + \nu_t - c_1}{c_2 + a_1 + d_2}.$$

We introduce this I_t **is hardwired to zero** specification in order to shed light on the role that inventories play by comparing outcomes with those under our two other versions of the problem.

The bottom right panel displays a production path for the original problem that we are interested in (the blue line) as well with an optimal production path for the model in which inventories are not useful (the green path) and also for the model in which, although inventories are useful, they are hardwired to zero and the firm pays cost $d(0, Q_t)$ for not setting sales $S_t = Q_t$ equal to zero (the orange line).

Notice that it is typically optimal for the firm to produce more when inventories aren't useful. Here there is no requirement to sell out of inventories and no costs from having sales deviate from inventories.

But "typical" does not mean "always".

Thus, if we look closely, we notice that for small t, the green "production when inventories aren't useful" line in the lower right panel is below optimal production in the original model.

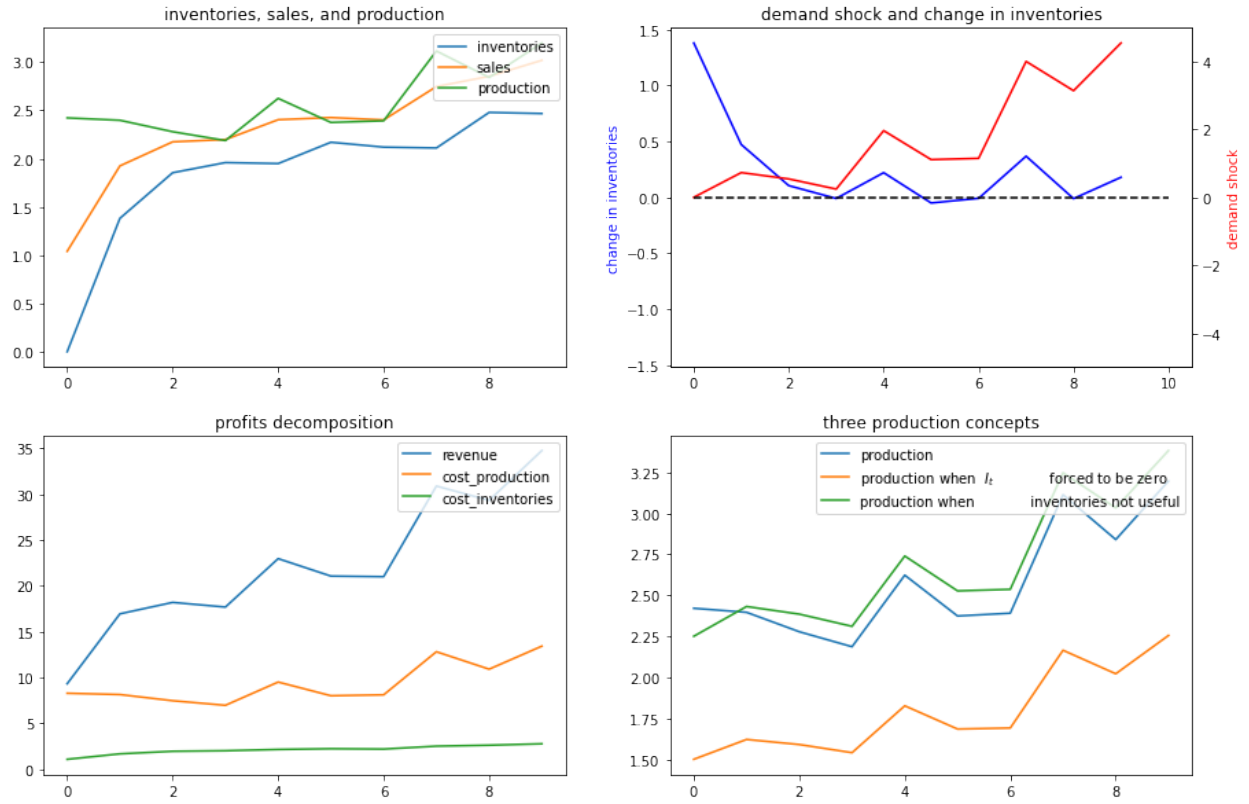
High optimal production in the original model early on occurs because the firm wants to accumulate inventories quickly in order to acquire high inventories for use in later periods.

But how the green line compares to the blue line early on depends on the evolution of the demand shock, as we will see in a deterministically seasonal demand shock example to be analyzed below.

In that example, the original firm optimally accumulates inventories slowly because the next positive demand shock is in the distant future.

To make the green-blue model production comparison easier to see, let's confine the graphs to the first 10 periods:

```
ex1.simulate(x0, T=10)
```



51.5 Example 2

Next, we shut down randomness in demand and assume that the demand shock ν_t follows a deterministic path:

$$\nu_t = \alpha + \rho \nu_{t-1}$$

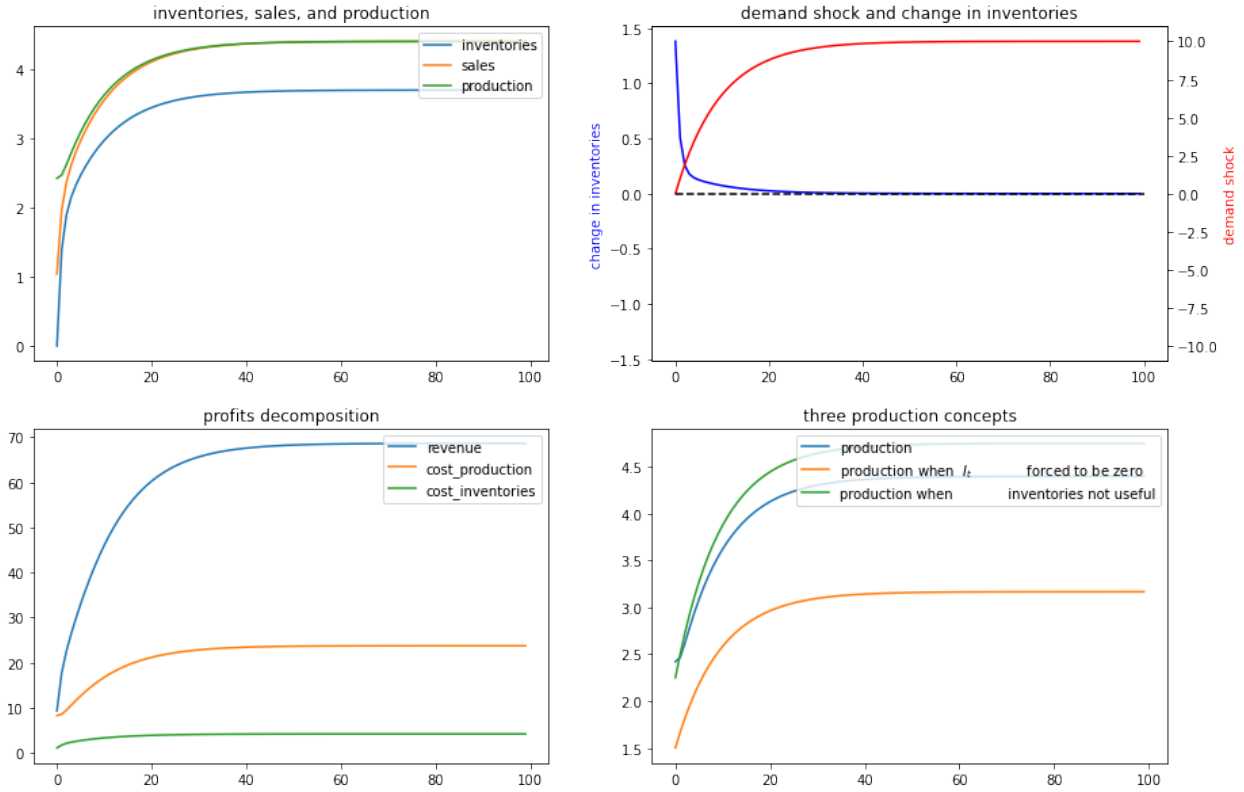
Again, we'll compute and display outcomes in some figures

```
ex2 = SmoothingExample(C2=[[0], [0]])
```

(continues on next page)

(continued from previous page)

```
x0 = [0, 1, 0]
ex2.simulate(x0)
```



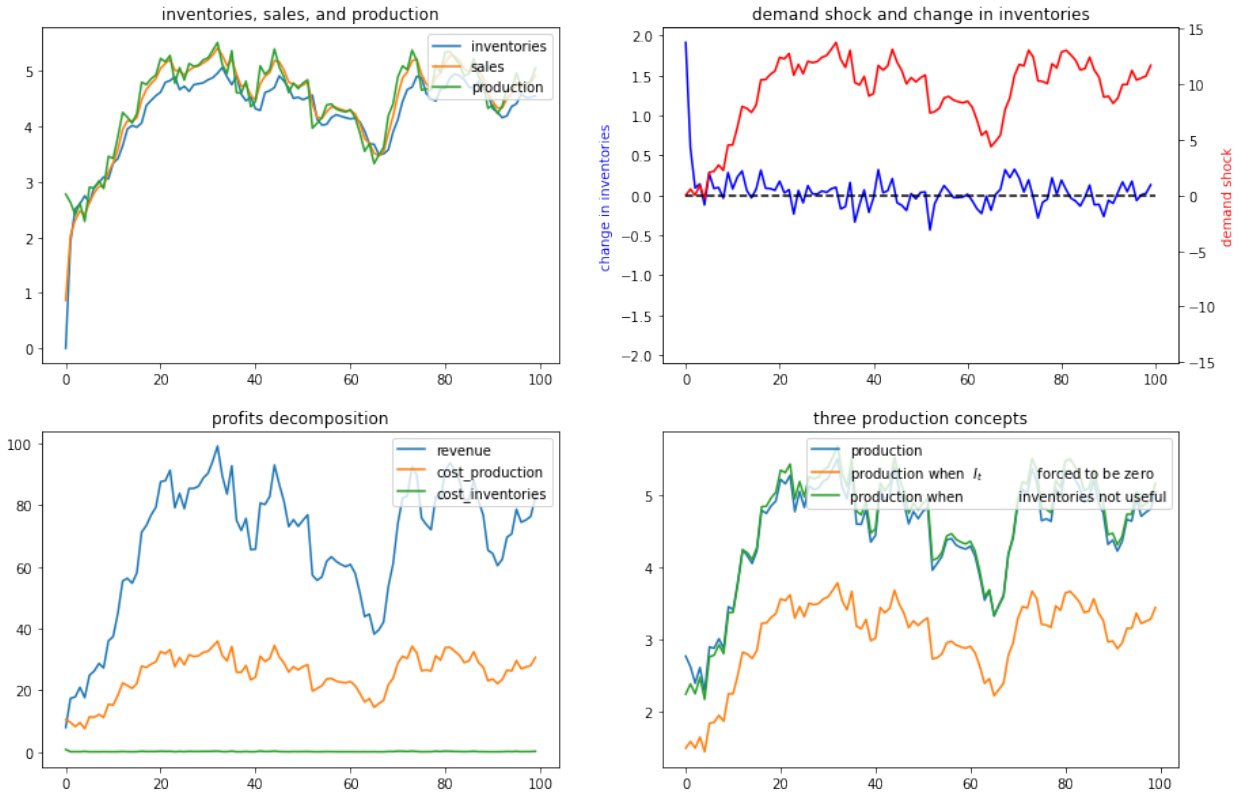
51.6 Example 3

Now we'll put randomness back into the demand shock process and also assume that there are zero costs of holding inventories.

In particular, we'll look at a situation in which $d_1 = 0$ but $d_2 > 0$.

Now it becomes optimal to set sales approximately equal to inventories and to use inventories to smooth production quite well, as the following figures confirm

```
ex3 = SmoothingExample(d1=0)

x0 = [0, 1, 0]
ex3.simulate(x0)
```



51.7 Example 4

To bring out some features of the optimal policy that are related to some technical issues in linear control theory, we'll now temporarily assume that it is costless to hold inventories.

When we completely shut down the cost of holding inventories by setting $d_1 = 0$ and $d_2 = 0$, something absurd happens (because the Bellman equation is opportunistic and very smart).

(Technically, we have set parameters that end up violating conditions needed to assure **stability** of the optimally controlled state.)

The firm finds it optimal to set $Q_t \equiv Q^* = \frac{-c_1}{2c_2}$, an output level that sets the costs of production to zero (when $c_1 > 0$, as it is with our default settings, then it is optimal to set production negative, whatever that means!).

Recall the law of motion for inventories

$$I_{t+1} = I_t + Q_t - S_t$$

So when $d_1 = d_2 = 0$ so that the firm finds it optimal to set $Q_t = \frac{-c_1}{2c_2}$ for all t, then

$$I_{t+1} - I_t = \frac{-c_1}{2c_2} - S_t < 0$$

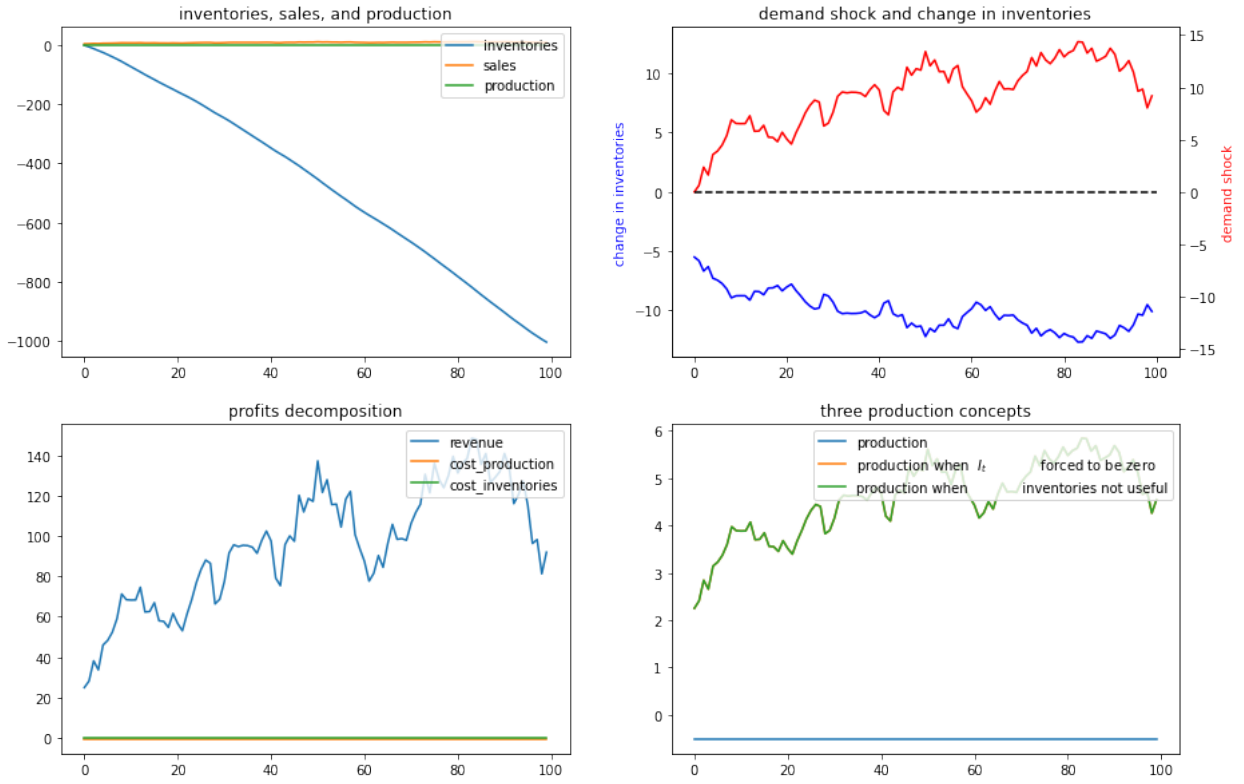
for almost all values of S_t under our default parameters that keep demand positive almost all of the time.

The dynamic program instructs the firm to set production costs to zero and to **run a Ponzi scheme** by running inventories down forever.

(We can interpret this as the firm somehow **going short in** or **borrowing** inventories)

The following figures confirm that inventories head south without limit

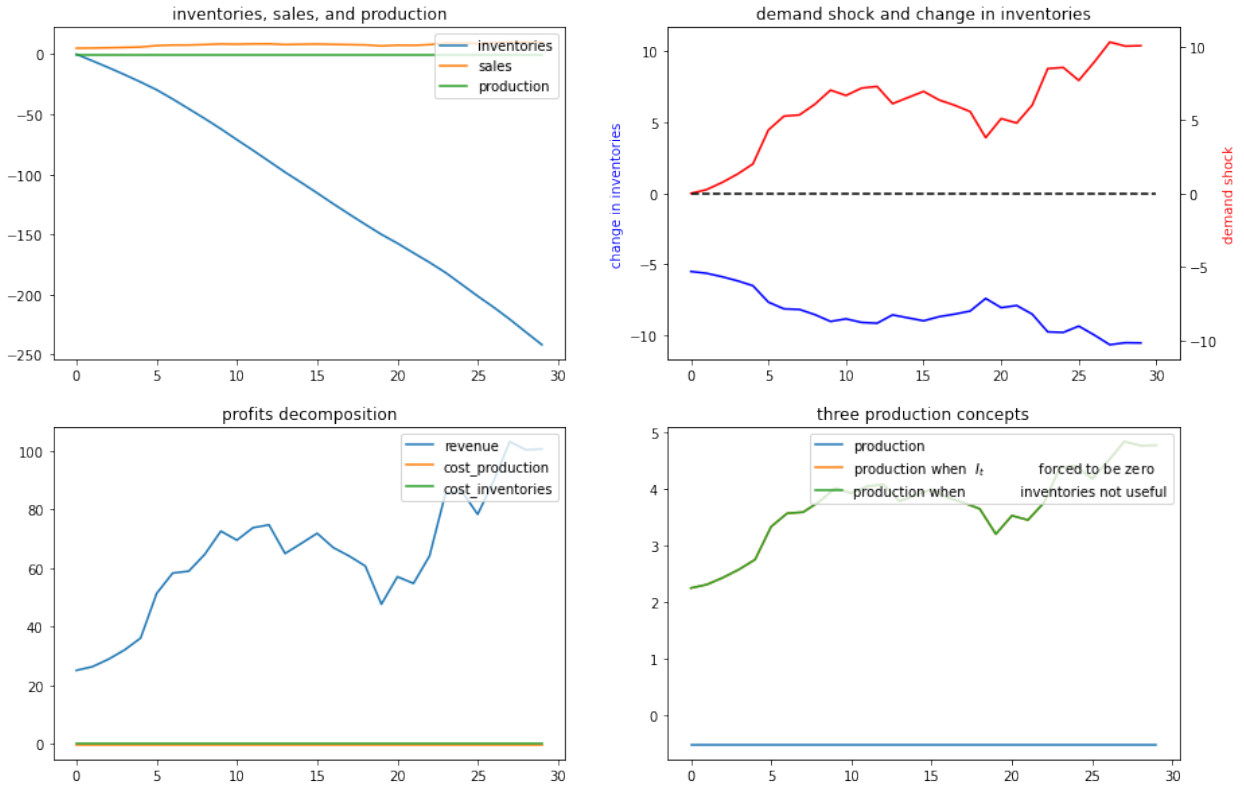
```
ex4 = SmoothingExample(d1=0, d2=0)

x0 = [0, 1, 0]
ex4.simulate(x0)
```



Let's shorten the time span displayed in order to highlight what is going on.

We'll set the horizon $T = 30$ with the following code

```
# shorter period
ex4.simulate(x0, T=30)
```



51.8 Example 5

Now we'll assume that the demand shock that follows a linear time trend

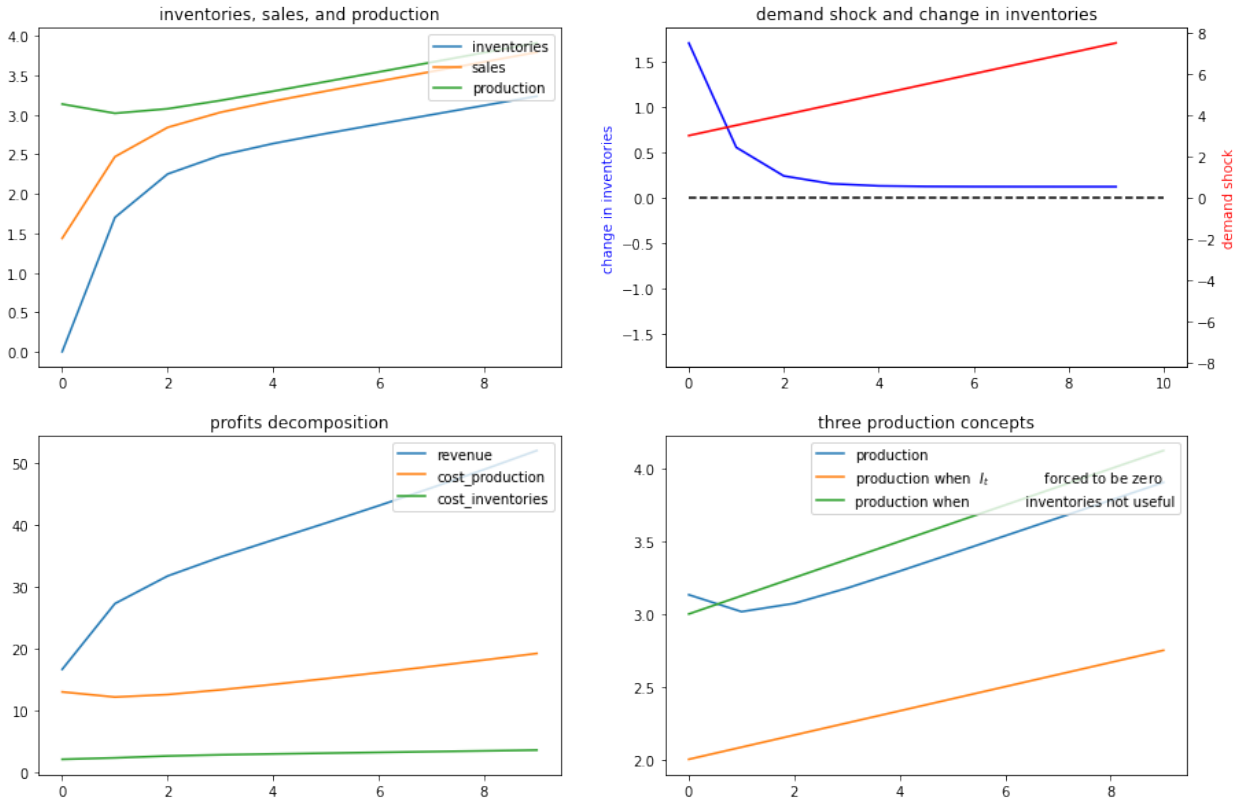
$$v_t = b + at, a > 0, b > 0$$

To represent this, we set $C_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and

$$A_{22} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, G = \begin{bmatrix} b & a \end{bmatrix}$$

```
# Set parameters
a = 0.5
b = 3.
```

```
ex5 = SmoothingExample(A22=[[1, 0], [1, 1]], C2=[[0], [0]], G=[b, a])

x0 = [0, 1, 0] # set the initial inventory as 0
ex5.simulate(x0, T=10)
```



51.9 Example 6

Now we'll assume a deterministically seasonal demand shock.

To represent this we'll set

$$A_{22} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, C_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, G' = \begin{bmatrix} b \\ a \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $a > 0, b > 0$ and

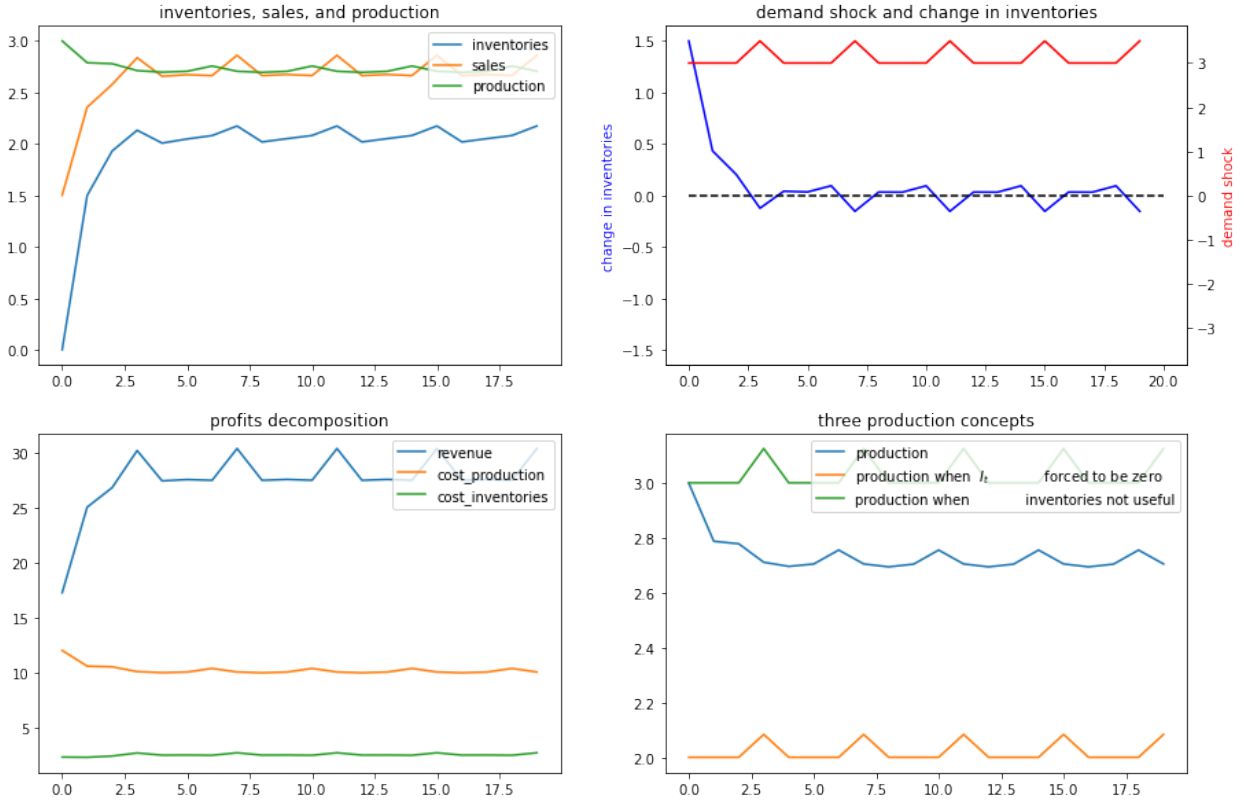
$$x_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

```
ex6 = SmoothingExample(A22=[[1, 0, 0, 0, 0],
                           [0, 0, 0, 0, 1],
                           [0, 1, 0, 0, 0],
                           [0, 0, 1, 0, 0],
                           [0, 0, 0, 1, 0]],
                       C2=[[0], [0], [0], [0], [0]],
```

(continues on next page)

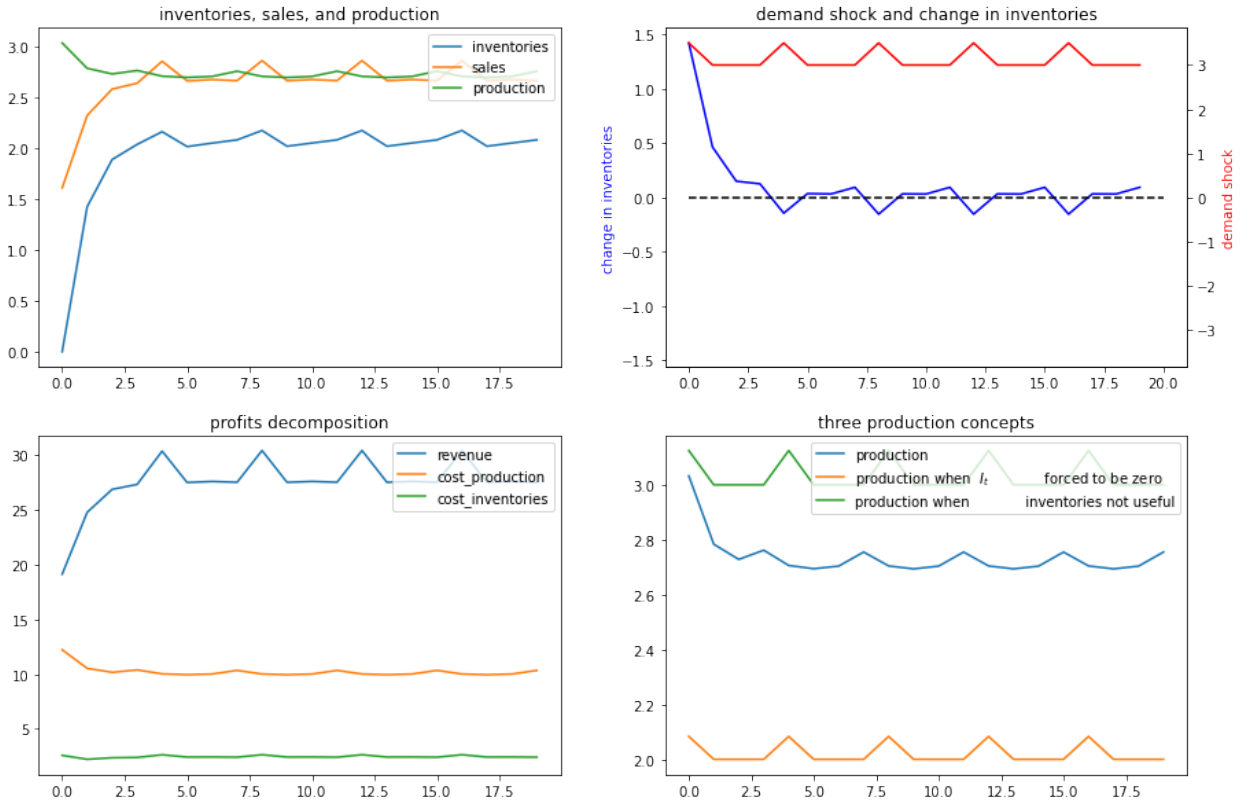
(continued from previous page)

```
                        G=[b, a, 0, 0, 0])

x00 = [0, 1, 0, 1, 0, 0] # Set the initial inventory as 0
ex6.simulate(x00, T=20)
```

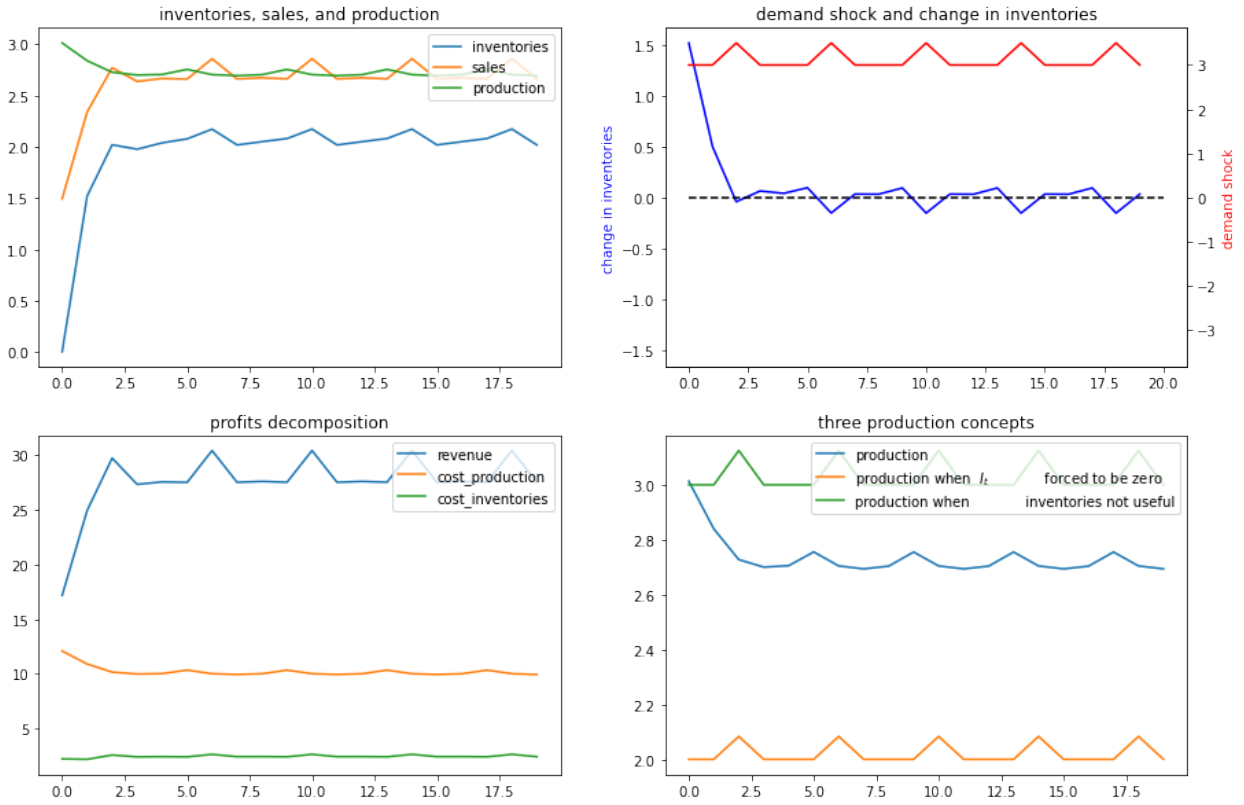


Now we'll generate some more examples that differ simply from the initial **season** of the year in which we begin the demand shock

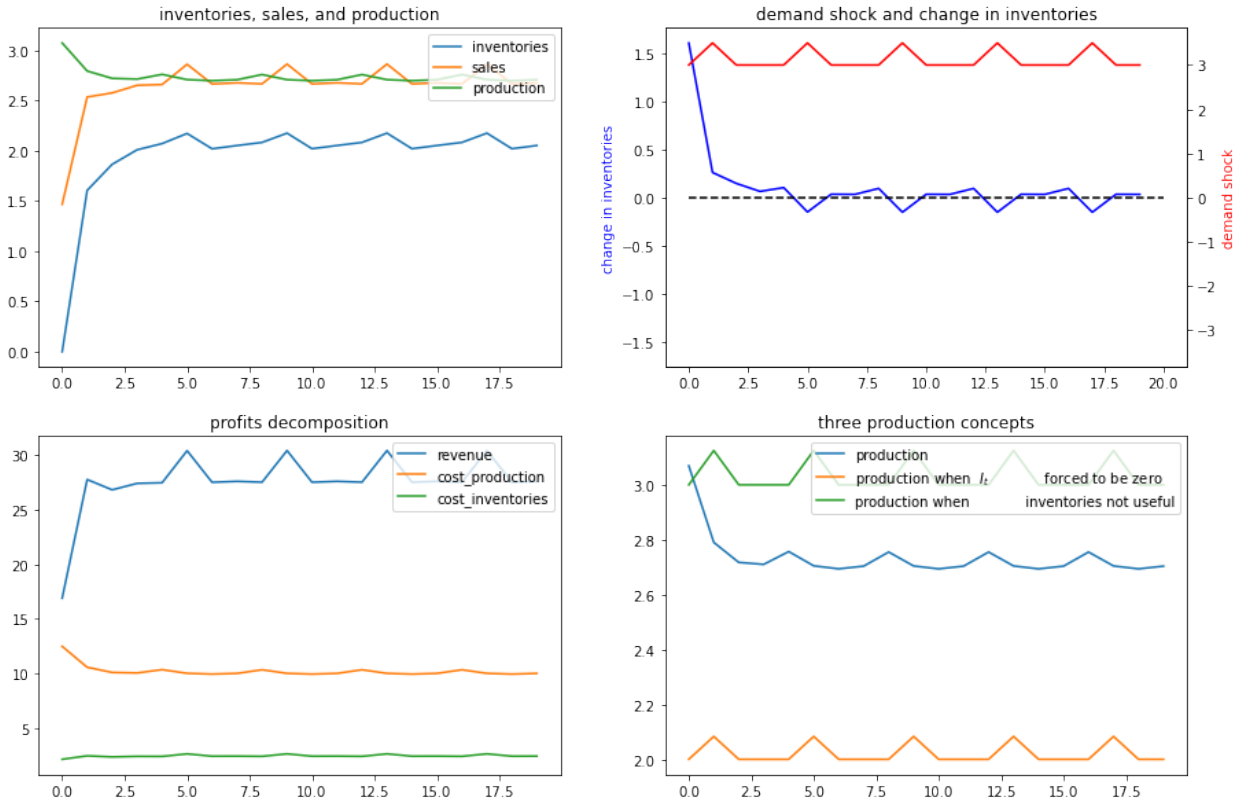
```
x01 = [0, 1, 1, 0, 0, 0]
ex6.simulate(x01, T=20)
```




```
x02 = [0, 1, 0, 0, 1, 0]
ex6.simulate(x02, T=20)
```




```
x03 = [0, 1, 0, 0, 0, 1]
ex6.simulate(x03, T=20)
```



51.10 Exercises

Please try to analyze some inventory sales smoothing problems using the `SmoothingExample` class.

51.10.1 Exercise 1

Assume that the demand shock follows AR(2) process below:

$$\nu_t = \alpha + \rho_1 \nu_{t-1} + \rho_2 \nu_{t-2} + \epsilon_t.$$

where $\alpha = 1$, $\rho_1 = 1.2$, and $\rho_2 = -0.3$. You need to construct $A22$, C, and G matrices properly and then to input them as the keyword arguments of `SmoothingExample` class. Simulate paths starting from the initial condition $x_0 = [0, 1, 0, 0]'$.

After this, try to construct a very similar `SmoothingExample` with the same demand shock process but exclude the randomness ϵ_t. Compute the stationary states $\bar{x}$ by simulating for a long period. Then try to add shocks with different magnitude to $\bar{\nu}_t$ and simulate paths. You should see how firms respond differently by staring at the production plans.

51.10.2 Exercise 2

Change parameters of $C(Q_t)$ and $d(I_t, S_t)$.

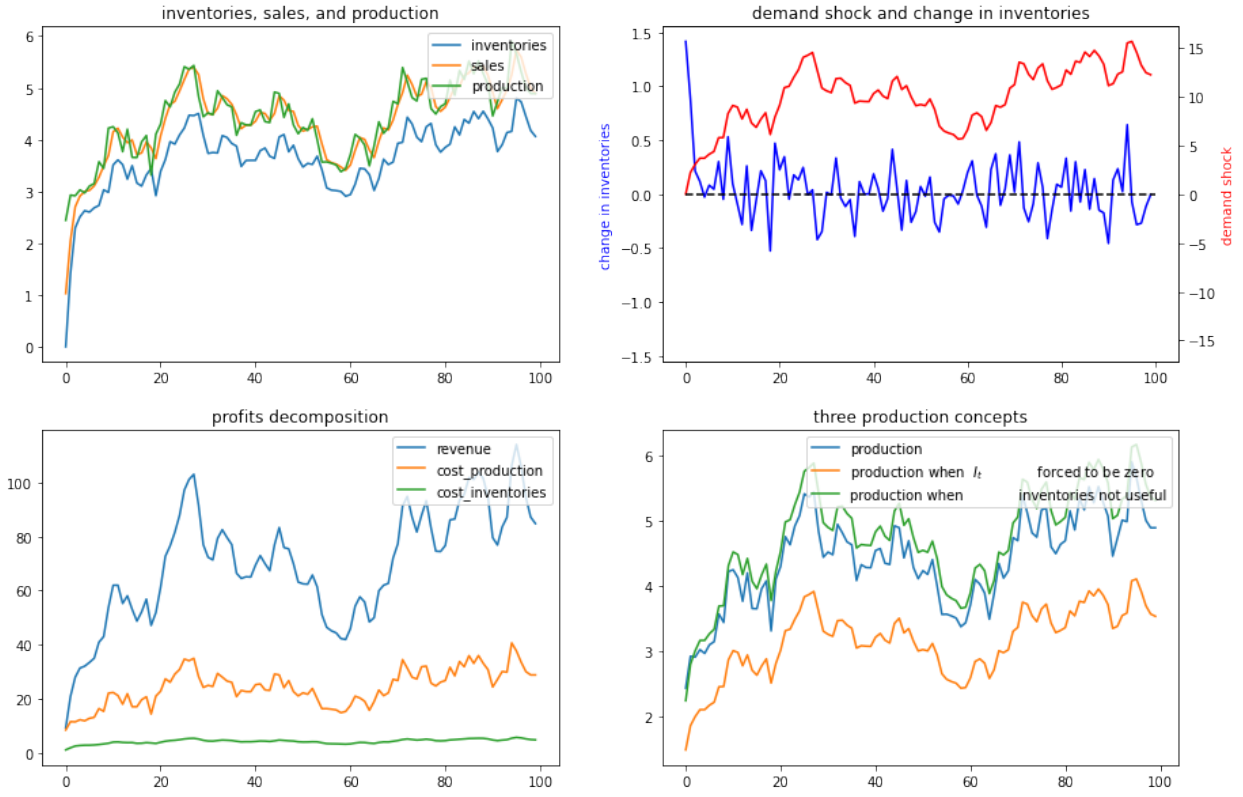
1. Make production more costly, by setting $c_2 = 5$.
2. Increase the cost of having inventories deviate from sales, by setting $d_2 = 5$.

51.10.3 Solution 1

```
# set parameters
α = 1
ρ1 = 1.2
ρ2 = -.3
```

```
# construct matrices
A22 =[[1,  0,  0],
      [1, ρ1, ρ2],
      [0,  1,  0]]
C2 = [[0], [1], [0]]
G = [0, 1, 0]
```

```
ex1 = SmoothingExample(A22=A22, C2=C2, G=G)

x0 = [0, 1, 0, 0] # initial condition
ex1.simulate(x0)
```



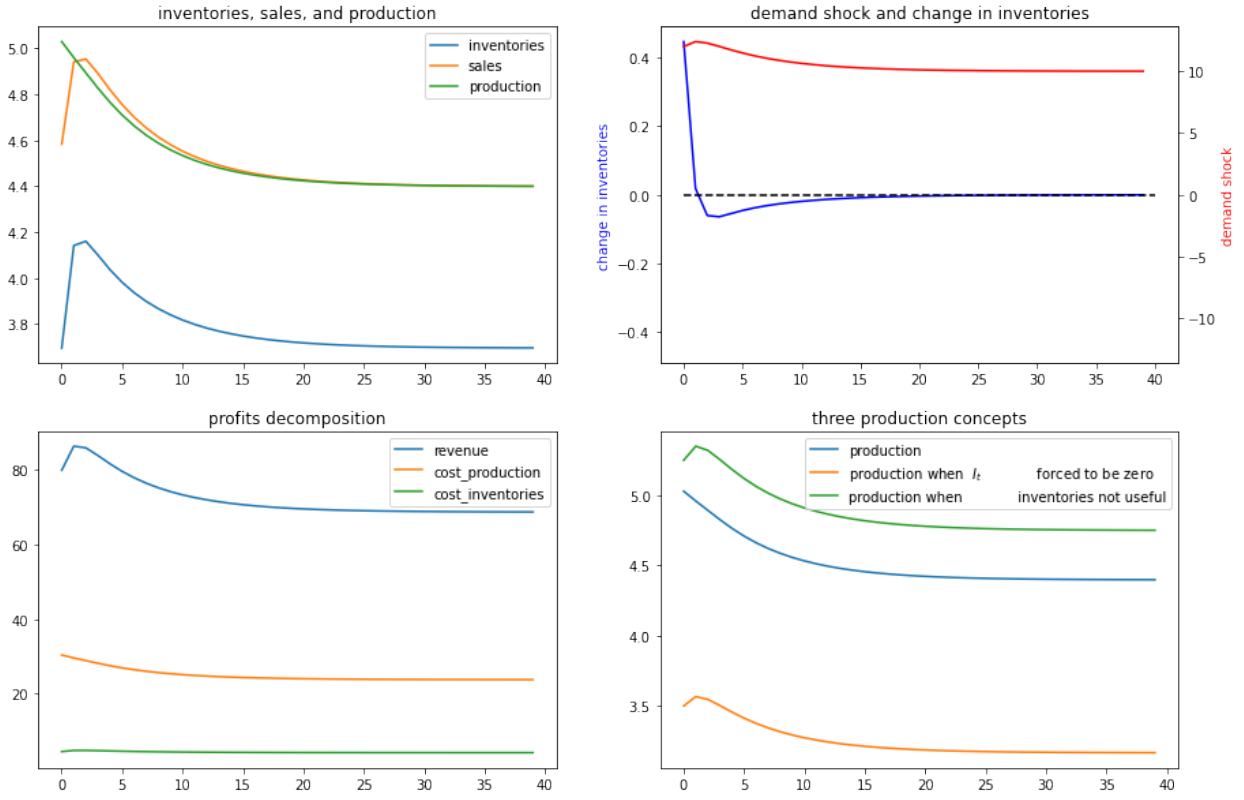
```
# now silence the noise
ex1_no_noise = SmoothingExample(A22=A22, C2=[[0], [0], [0]], G=G)

# initial condition
x0 = [0, 1, 0, 0]

# compute stationary states
x_bar = ex1_no_noise.LQ.compute_sequence(x0, ts_length=250)[0][:, -1]
x_bar
```

```
array([ 3.69387755,  1.        , 10.        , 10.        ])
```

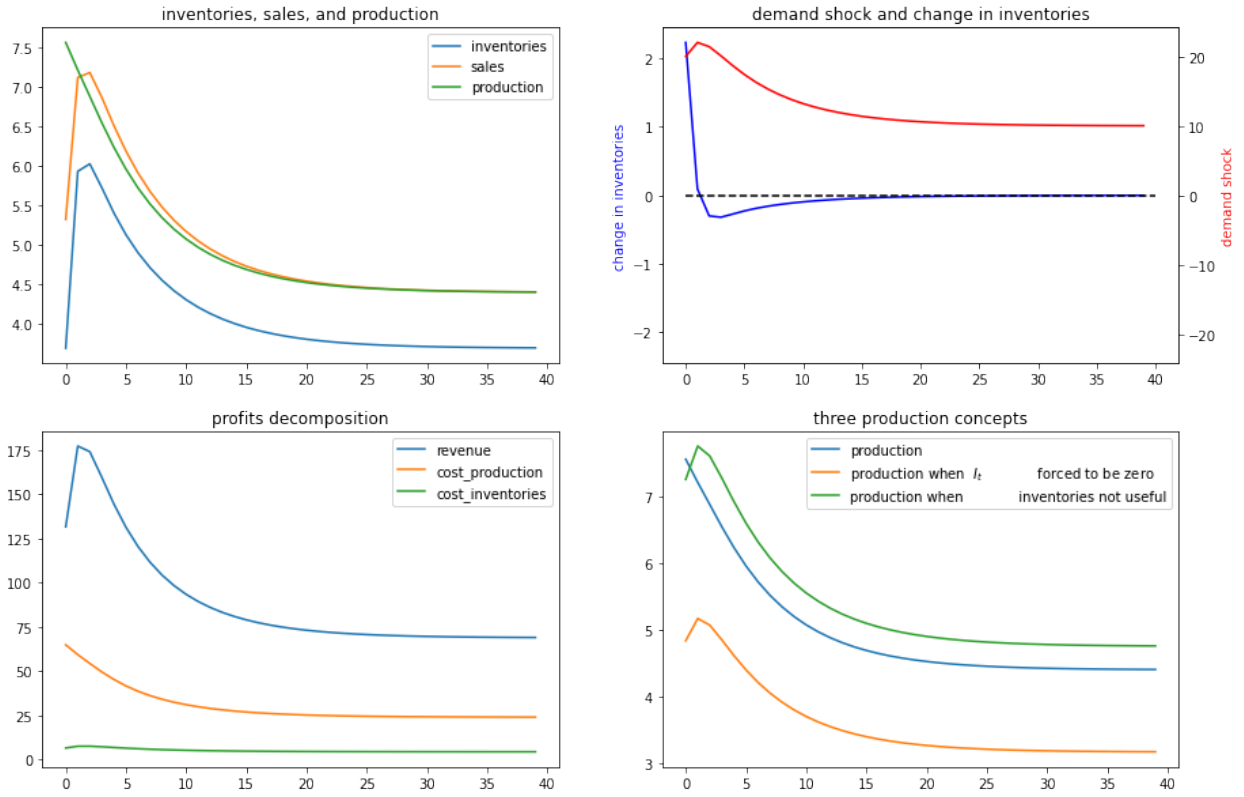
In the following, we add small and large shocks to $\bar{\nu}_t$ and compare how firm responds differently in quantity. As the shock is not very persistent under the parameterization we are using, we focus on a short period response.

```
T = 40
```

```
# small shock
x_bar1 = x_bar.copy()
x_bar1[2] += 2
ex1_no_noise.simulate(x_bar1, T=T)
```



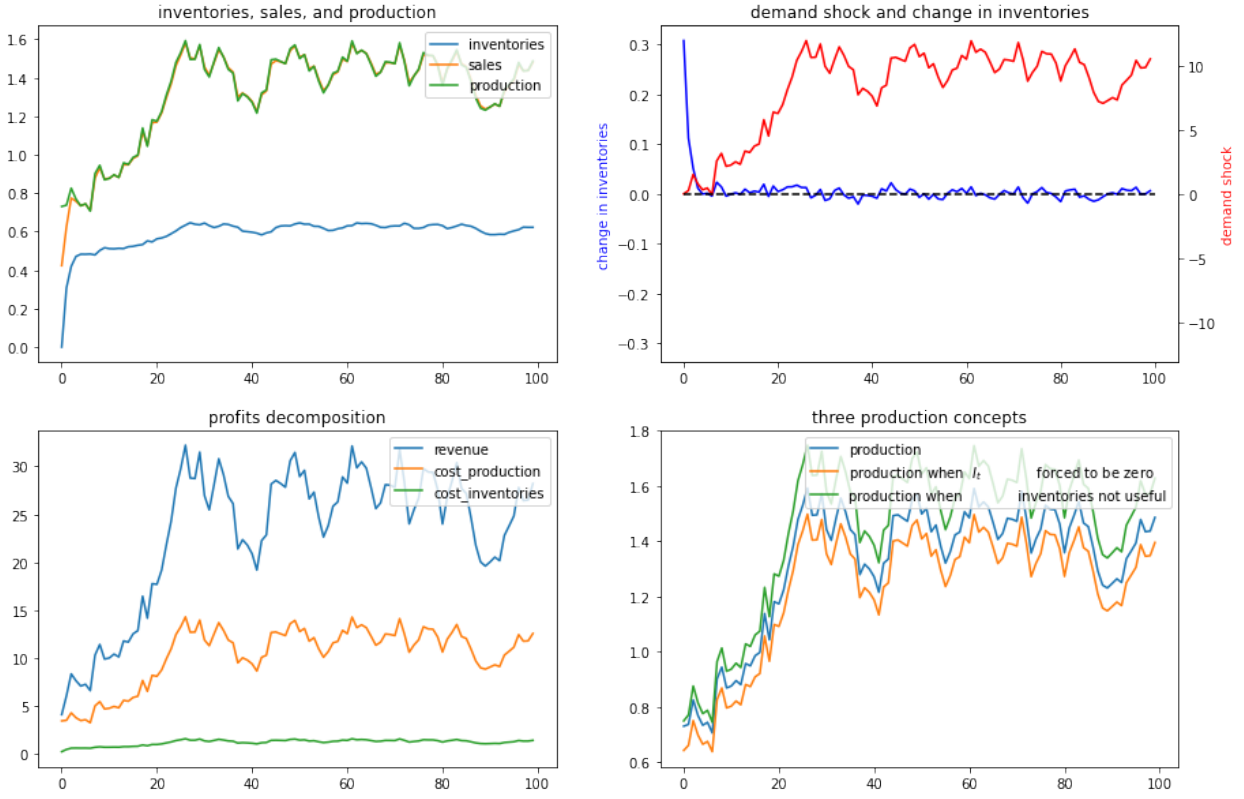

```
# large shock
x_bar1 = x_bar.copy()
x_bar1[2] += 10
ex1_no_noise.simulate(x_bar1, T=T)
```



51.10.4 Solution 2

```
x0 = [0, 1, 0]
```

```
SmoothingExample(c2=5).simulate(x0)
```




```
SmoothingExample(d2=5).simulate(x0)
```