A Monetarist Theory of Price Levels

Overview

我们将首先使用线性代数来解释，然后通过一些实验来验证“**货币主义价格水平理论(monetarist theory of price levels)**”。

经济学家之所以称其为“货币”或“货币主义”的价格水平理论，是因为价格水平的影响是通过中央银行决定印钞供应量来实现的。

- 政府的财政政策决定了其*支出*是否超过其*税收*
- 如果其支出超过税收，政府可以指示中央银行通过*印钞*来弥补差额
- 这会导致价格水平的变化，因为价格水平路径会调整，以使货币供应量与货币需求相等

有时，这种理论也被称为“财政价格水平理论”，以强调财政赤字在塑造货币供应变化中的重要性。

在财政价格水平理论中起作用的基本力量有助于理解这些事件。

根据这一理论，当政府持续支出超过税收收入并印钞来弥补赤字（“赤字”称为“政府赤字”）时，它会对价格水平产生上行压力并产生持续的通货膨胀。

“货币主义”或“财政价格水平理论”认为：

- 要*开始*持续的通货膨胀，政府开始持续运行货币融资的政府赤字
- 要*停止*持续的通货膨胀，政府停止持续运行货币融资的政府赤字

本次讲座中的模型是菲利普·卡根[卡根，1956]用来研究恶性通货膨胀的货币动态的“理性预期”（或“完美预见”）版本的模型。

虽然卡根没有使用该模型的“理性预期”版本，但托马斯·萨金特[萨金特，1982]在研究第一次世界大战后欧洲四大通货膨胀的终结时使用了。

我们现在描述了一个不强加“理性预期”而是使用卡根和他的老师米尔顿·弗里德曼所说的“适应性预期”的模型版本

- 当前理性预期版本的模型代数运算较简单
- 代数复杂性的差异可以追溯到以下来源：适应性预期版本的模型有更多的内生变量和更多的自由参数

我们与理性预期版本的模型的一些定量实验旨在说明财政理论如何解释那些大通货膨胀的突然结束。

在这些实验中，我们将遇到有时伴随成功的通货膨胀稳定化计划的“速度红利”。

为了便于使用线性矩阵代数作为我们的主要数学工具，我们将使用模型的有限期版本。

我们的数学工具是矩阵乘法和矩阵求逆。

Structure of the Model

模型包括：

- 一个函数，表达政府印制货币的**实际余额需求**，作为公众预期通货膨胀率的逆函数
- 货币供应量的**外生**增长率序列。货币供应量增长是因为政府印钞来支付商品和服务
- 一个平衡条件，将**货币需求等同于供应**
- 一个“**完美预见(Perfect Foresight)**”假设，即**公众预期的通货膨胀率等于实际通货膨胀率**。

为了正式表示模型，设

- m_t 为名义货币余额的对数；
- $\mu_t = m_{t+1} - m_t$ 为名义货币余额的净增长率；
- p_t 为价格水平的对数；
- $\pi_t = p_{t+1} - p_t$ 为 t 和 $t+1$ 之间的净通货膨胀率；
- π_t^* 为公众预期的 t 和 $t+1$ 之间的通货膨胀率；
- T 为**Horizon**——即模型将确定 p_t 的最后一个时期；
- π_{T+1}^* 为时间 T 和 $T+1$ 之间的最终通货膨胀率。

实际余额需求 $\exp\left(m_t^d - p_t\right)$ 由以下版本的卡根需求函数控制

$$m_t^d - p_t = -\alpha \pi_t^* \text{ , } \alpha > 0; \quad t = 0, 1, \ldots, T. \tag{14.1}$$

这个方程断言实际余额需求与公众预期的通货膨胀率成反比。

人们通过解决预测问题，以某种方式获得**完美预见**。

这让我们可以设定

$$\pi_t^* = \pi_t, \quad \forall t \tag{14.2}$$

同时将货币需求等同于供应让我们可以设定 $m_t^d = m_t, \forall t \geq 0$。

前面的方程则意味着

$$m_t - p_t = -\alpha(p_{t+1} - p_t) \tag{14.3}$$

为了详细说明私人代理具有完美预见的含义，我们将时间 t 的方程 (14.3) 减去 $t+1$ 时的同一方程得到

$$\mu_t - \pi_t = -\alpha\pi_{t+1} + \alpha\pi_t,$$

我们将其重写为 π_s 的前瞻性一阶线性差分方程，以 μ_s 作为"驱动变量"：

$$\pi_t = \frac{\alpha}{1+\alpha}\pi_{t+1} + \frac{1}{1+\alpha}\mu_t, \quad t = 0, 1, \ldots, T$$

其中 $0 < \frac{\alpha}{1+\alpha} < 1$。

设 $\delta = \frac{\alpha}{1+\alpha}$，让我们将前面的方程表示为

$$\pi_t = \delta\pi_{t+1} + (1-\delta)\mu_t, \quad t = 0, 1, \ldots, T$$

将这个 $T+1$ 方程组写成单一矩阵方程

$$\begin{bmatrix} 1 & -\delta & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\delta & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -\delta & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & -\delta & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -\delta \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \\ \vdots \\ \pi_{T-1} \\ \pi_T \end{bmatrix} = (1-\delta) \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{T-1} \\ \mu_T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \delta\pi_{T+1}^* \end{bmatrix} \tag{14.4}$$

通过将方程 (14.4) 的两边乘以左侧矩阵的逆，我们可以计算

$$\pi \equiv \begin{bmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \\ \vdots \\ \pi_{T-1} \\ \pi_T \end{bmatrix}$$

结果表明

$$\pi_t = (1-\delta)\sum_{s=t}^{T} \delta^{s-t}\mu_s + \delta^{T+1-t}\pi_{T+1}^* \tag{14.5}$$

我们可以将方程

$$m_{t+1} = m_t + \mu_t, \quad t = 0, 1, \ldots, T$$

表示为矩阵方程

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_T \\ m_{T+1} \end{bmatrix} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{T-1} \\ \mu_T \end{bmatrix} + \begin{bmatrix} m_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{14.6}$$

将方程 (14.6) 的两边乘以左侧矩阵的逆将给出

$$m_t = m_0 + \sum_{s=0}^{t-1} \mu_s, \quad t = 1, \ldots, T+1 \tag{14.7}$$

方程 (14.7) 显示，时间 t 的货币供应量对数等于初始货币供应量 m_0 的对数加上时间 0 和 T 之间货币增长率的累积。

Continuation values

为了确定继续通货膨胀率 π_{T+1}^*，我们将在时间 $t = T+1$通过应用以下无限期版本的方程 (14.5)：

$$\pi_t = (1-\delta)\sum_{s=t}^{\infty} \delta^{s-t}\mu_s, \tag{14.8}$$

并且假设在T之后 μ_t 沿以下路径运动：

$$\mu_{t+1} = \gamma^* \mu_t, \quad t \geq T.$$

将前面的方程插入到 $t = T + 1$ 的方程 (14.8) 并重新排列，我们可以推断出

$$\pi^*_{T+1} = \frac{1-\delta}{1-\delta\gamma^*}\gamma^* \mu_T \tag{14.9}$$

其中我们要求 $|\gamma^* \delta| < 1$。

让我们实现并解决这个模型。

像往常一样，我们将首先导入一些 Python 模块。

In [ ]:
```
import numpy as np
from collections import namedtuple
import matplotlib.pyplot as plt
```

首先，我们将参数存储在一个 `namedtuple` 中：

In [ ]:
```
# Create the rational expectation version of Cagan model in finite time
CaganREE = namedtuple("CaganREE",
                      ["m0",    # initial money supply
                       "μ_seq", # sequence of rate of growth
                       "α",     # sensitivity parameter
                       "δ",     # α/(1 + α)
                       "π_end"  # terminal expected inflation
                      ])

def create_cagan_model(m0=1, α=5, μ_seq=None):
    δ = α/(1 + α)
    π_end = μ_seq[-1]    # compute terminal expected inflation
    return CaganREE(m0, μ_seq, α, δ, π_end)
```

现在我们可以使用上述矩阵方程来计算 $t = 1, \ldots, T+1$ 时的 π_t、m_t 和 p_t：

In [ ]:
```
def solve(model, T):
    # Unpack model parameters
    m0, π_end, μ_seq, α, δ = (model.m0, model.π_end,
                              model.μ_seq, model.α, model.δ)

    # Create matrix representation for the first equation
    A1 = np.eye(T+1, T+1) - δ * np.eye(T+1, T+1, k=1)
    # Create matrix representation for the second equation
    A2 = np.eye(T+1, T+1) - np.eye(T+1, T+1, k=-1)

    # Construct the right-hand side vector for the first equation
    b1 = (1-δ) * μ_seq + np.concatenate([np.zeros(T), [δ * π_end]])
    # Construct the right-hand side vector for the second equation
    b2 = μ_seq + np.concatenate([[m0], np.zeros(T)])

    # Solve the first equation for π_seq using matrix inversion
    π_seq = np.linalg.solve(A1, b1)
    # Solve the second equation for m_seq using matrix inversion
    m_seq = np.linalg.solve(A2, b2)

    # Append the end value of π to the sequence
    π_seq = np.append(π_seq, π_end)
    # Append the initial value of m to the sequence
    m_seq = np.append(m0, m_seq)

    # Calculate the price level sequence using the money supply and inflation sequences
    p_seq = m_seq + α * π_seq

    # Return the sequences for inflation, money supply, and price level
    return π_seq, m_seq, p_seq
```

一些定量实验

在下面的实验中，我们将使用公式 (14.9) 作为预期通货膨胀的终端条件。

在设计这些实验时，我们将对 $\{\mu_t\}$ 做出与公式 (14.9) 一致的假设。

我们描述了几个这样的实验。

在所有实验中，

$$\mu_t = \mu^*, \quad t \geq T_1$$

这样，根据我们的符号和上面的 π^*_{T+1} 公式，$\tilde{\gamma} = 1$。

实验 1：预期的突然稳定化

在这个实验中，我们将研究当 $\alpha > 0$ 时，预期的通货膨胀稳定化如何影响其之前的通货膨胀。

我们将研究一种情况，即货币供应量的增长率从 $t = 0$ 到 $t = T_1$ 时是 μ_0，然后在 $t = T_1$ 时永久性地下降到 μ^*。

因此，设 $T_1 \in (0, T)$。

所以当 $\mu_0 > \mu^*$ 时，我们假设

$$\mu_{t+1} = \begin{cases} \mu_0, & t = 0, \ldots, T_1 - 1 \\ \mu^*, & t \geq T_1 \end{cases}$$

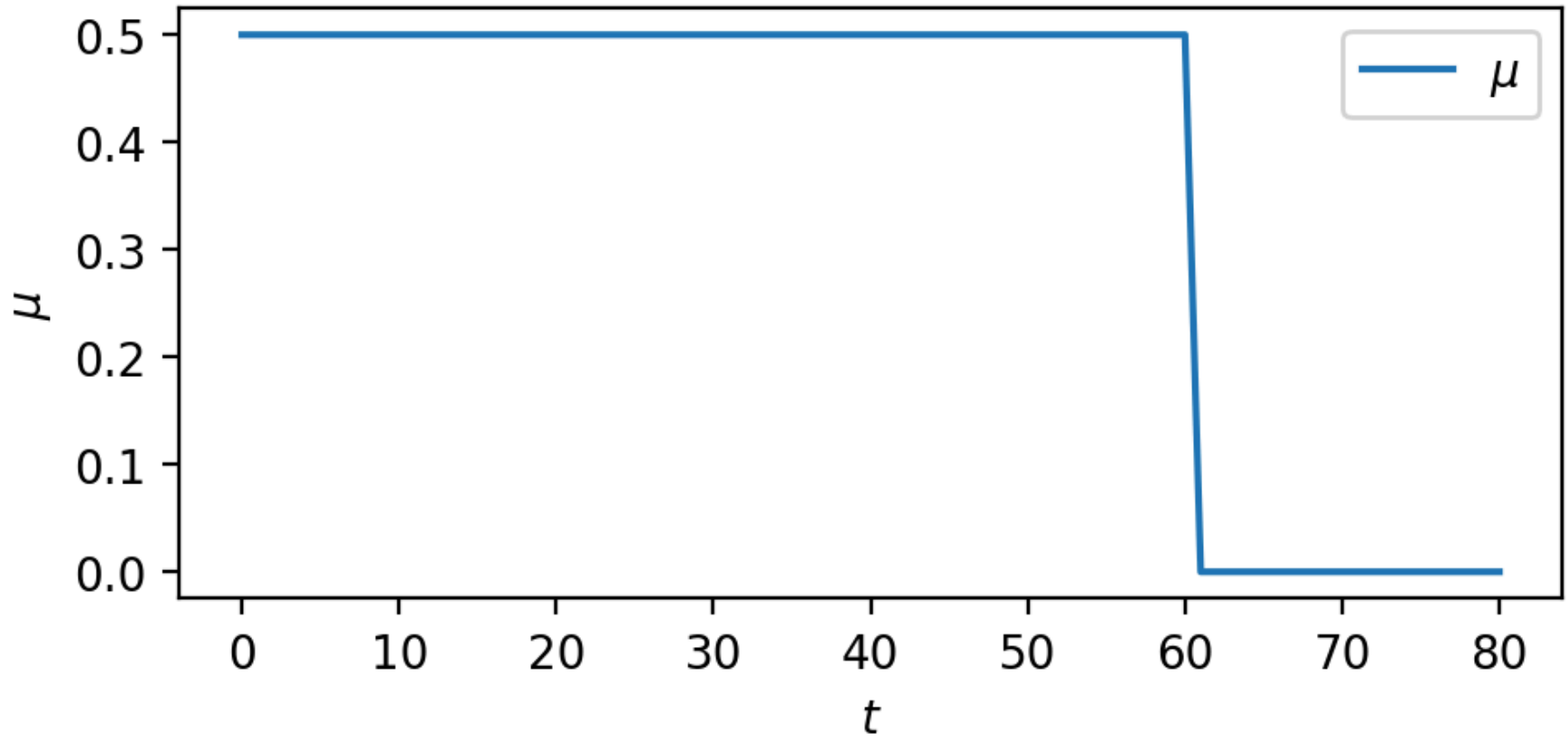
我们将首先执行一个版本的“实验 1”，在这个版本中，政府在时间 T_1 实施了一个*预期的*货币创造速率的突然永久性减少。

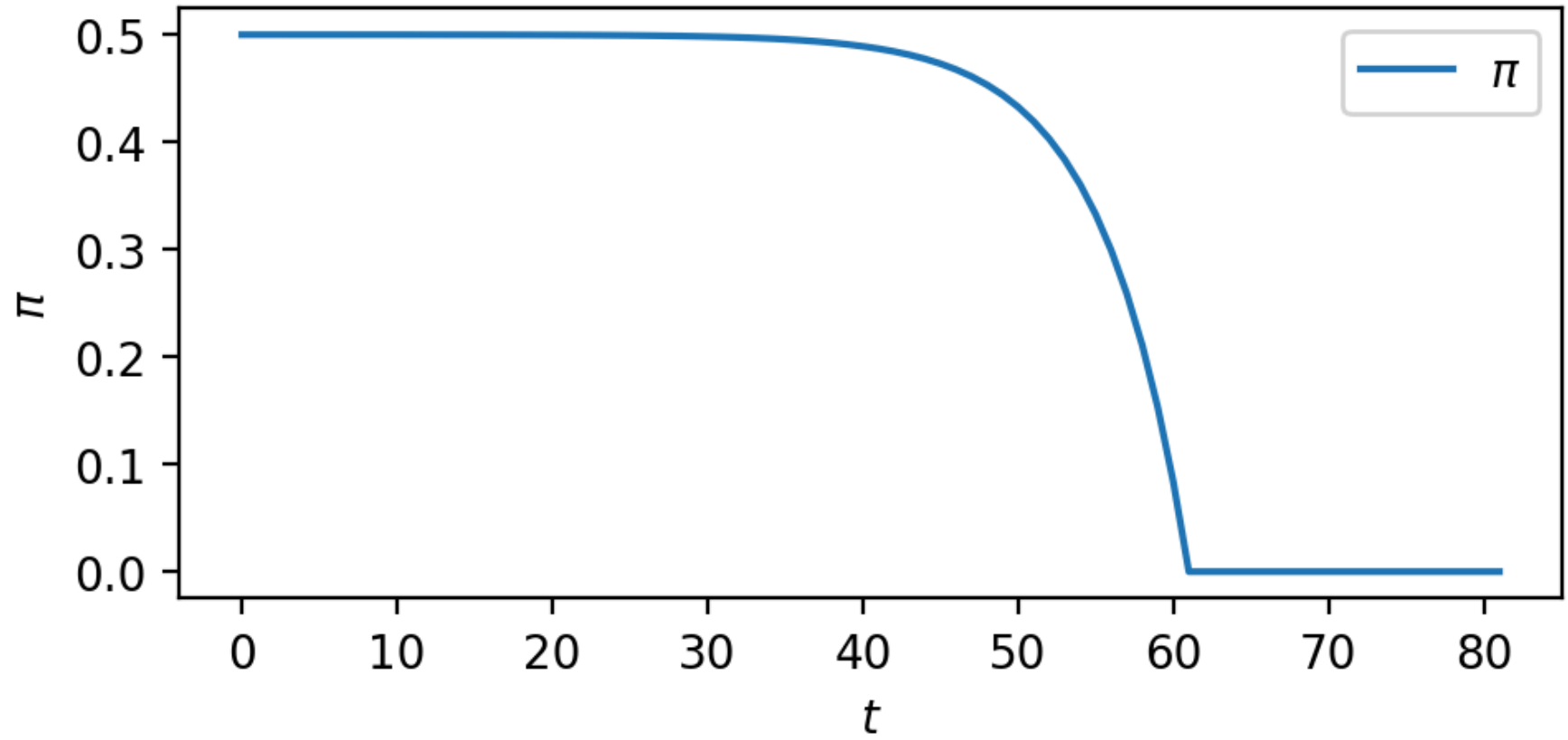
让我们用以下参数进行实验

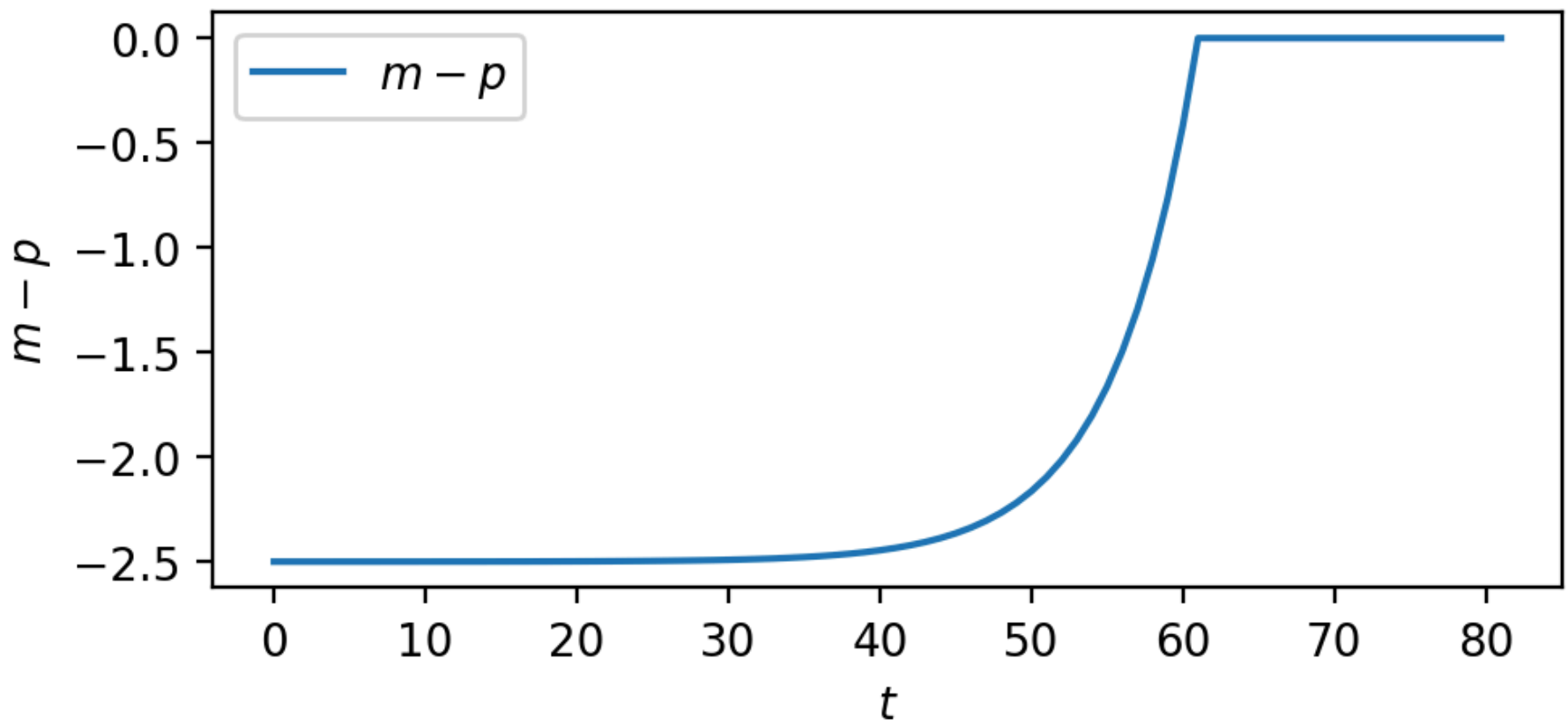
In [ ]:
```
T1 = 60
μ0 = 0.5
μ_star = 0
T = 80
# this step is just the given exogenous mu sequence above
μ_seq_1 = np.append(μ0*np.ones(T1+1), μ_star*np.ones(T-T1))

cm = create_cagan_model(μ_seq=μ_seq_1)

# solve the model
π_seq_1, m_seq_1, p_seq_1 = solve(cm, T)
```

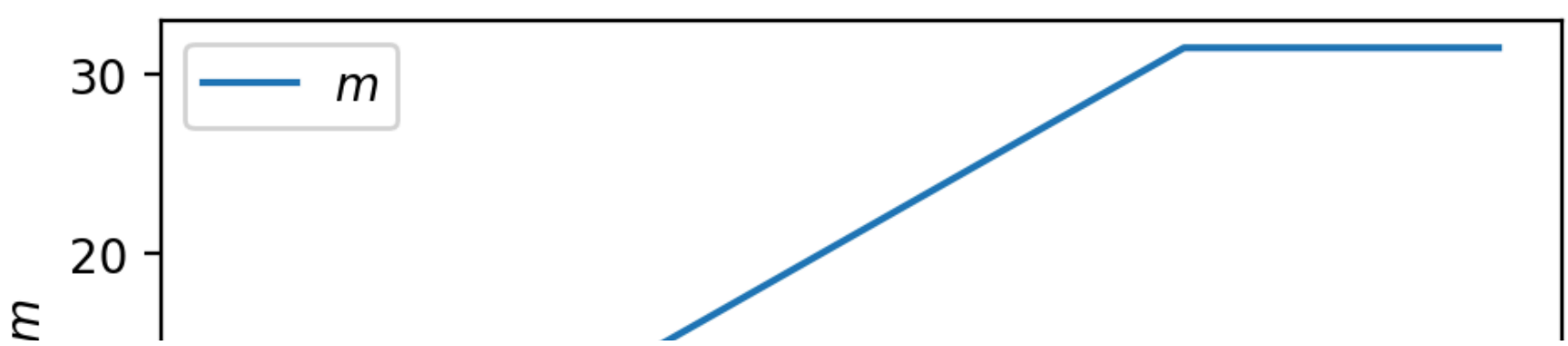
现在，我们使用以下函数绘制结果图

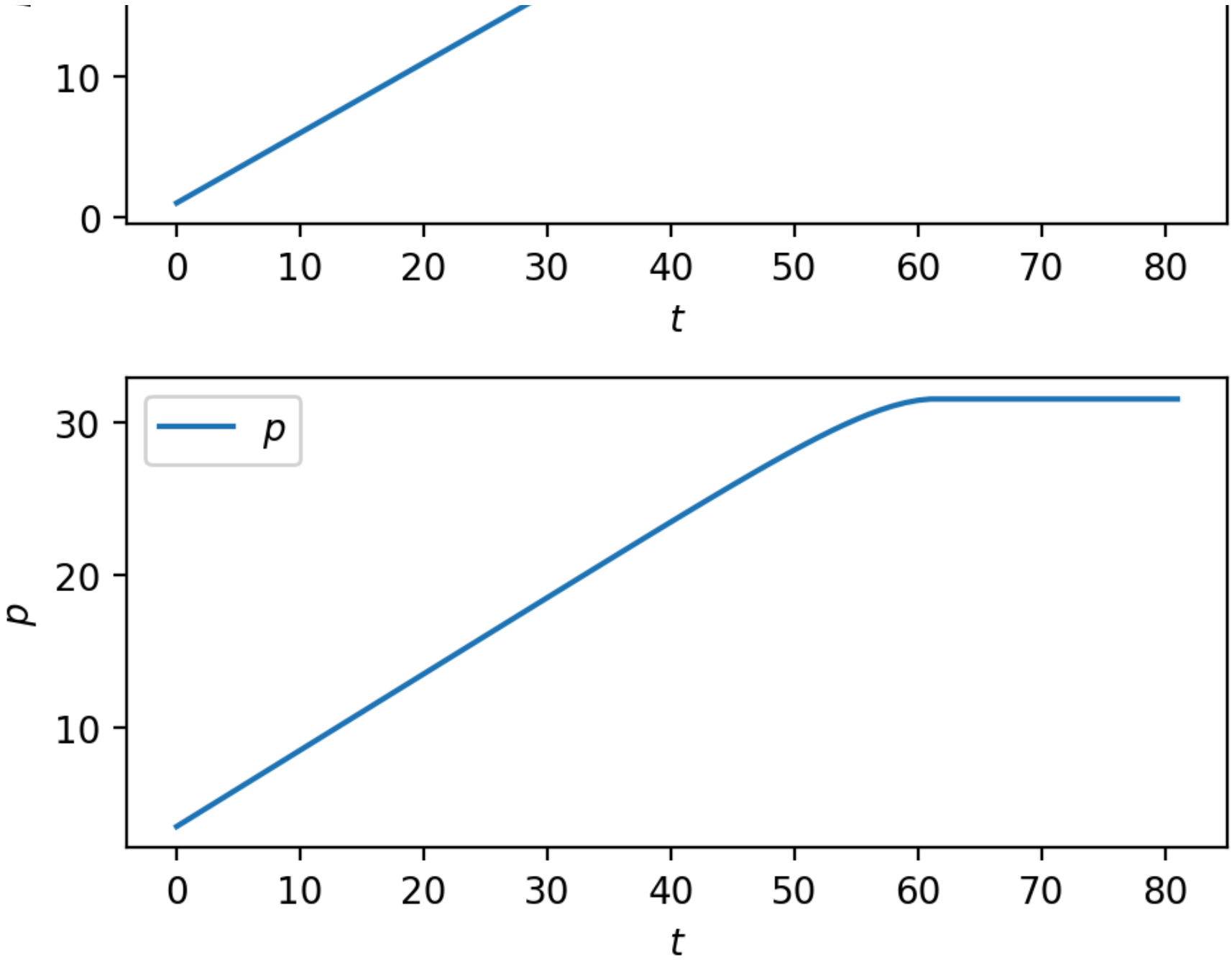
In [ ]:
```
def plot_sequences(sequences, labels):
    fig, axs = plt.subplots(len(sequences), 1, figsize=(5, 12), dpi=200)
    for ax, seq, label in zip(axs, sequences, labels):
        ax.plot(range(len(seq)), seq, label=label)
        ax.set_ylabel(label)
        ax.set_xlabel('$t$')
        ax.legend()
    plt.tight_layout()
    plt.show()

sequences = (μ_seq_1, π_seq_1, m_seq_1 - p_seq_1, m_seq_1, p_seq_1)
plot_sequences(sequences, (r'$\mu$', r'$\pi$', r'$m - p$', r'$m$', r'$p$'))
```


0.5
0.4
0.3
0.2
0.1
0.0
μ
μ
0
10
20
30
40
50
60
70
80
t


0.5
0.4
0.3
0.2
0.1
0.0
π
π
0
10
20
30
40
50
60
70
80
t


0.0
−0.5
−1.0
−1.5
−2.0
−2.5
$m - p$
$m - p$
0
10
20
30
40
50
60
70
80
t


30
20
m
m



货币增长率 μ_t 的图表在顶部面板中描绘了从 0.5 突然减少到 0 的情况，这一变化发生在时间 $T_1 = 60$。

这导致通货膨胀率 π_t 逐渐减少，这种减少发生在时间 T_1 的货币供应增长率减少之前。

注意通货膨胀率是如何在 T_1 时平滑（即连续）下降到 0 的——与货币增长率不同，它在 T_1 时并没有突然"跳"下来。

这是因为在 T_1 时 μ 的减少从一开始就被预见到了。

虽然底部面板中描绘的对数货币供应在 T_1 处有一个折点，但对数价格水平却没有——它是"平滑的"——这再次是因为 μ 的减少已经被预见到了。

为了为我们的下一个实验做准备，我们想更深入地研究价格水平的决定因素。

对数价格水平

我们可以使用方程 (14.1) 和 (14.2) 来发现价格水平的对数满足

$$p_t = m_t + \alpha \pi_t \tag{14.10}$$

或者，通过使用方程 (14.5)，

$$p_t = m_t + \alpha \left[(1-\delta) \sum_{s=t}^{T} \delta^{s-t} \mu_s + \delta^{T+1-t} \pi^*_{T+1} \right] \tag{14.11}$$

在我们的下一个实验中，我们将研究一个"意外"的永久性货币增长变化，这在之前是完全没有预料到的。

在时间 T_1 当"意外"的货币增长率变化发生时，为了满足 方程 (14.10)，实际余额的对数向上跳变 *上升*，而 π_t 向下跳变 *下降*。

但为了使 $m_t - p_t$ 跳变，哪个变量跳变，m_{T_1} 还是 p_{T_1}？

我们先来探讨这个有趣的问题。

什么会跳变?

在时间 T_1 会发生什么跳变?

是 p_{T_1} 还是 m_{T_1}?

如果我们坚持货币供应量 m_{T_1} 被锁定在其从过去继承的值 $m^1_{T_1}$，那么公式 (14.10) 意味着价格水平在时间 T_1 向下跳变，以与 π_{T_1} 的向下跳变相一致。

关于货币供应水平的另一种假设是，作为“通货膨胀稳定化”的一部分，政府根据以下公式重新设定 m_{T_1}：

$$m^2_{T_1} - m^1_{T_1} = \alpha(\pi^1_{T_1} - \pi^2_{T_1}), \tag{14.12}$$

这描述了政府如何根据与货币稳定化相关的预期通货膨胀的跳变，在 T_1 重新设定货币供应。

这样做将使价格水平在 T_1 保持连续。

通过根据方程 (14.12) 让货币跳变，货币当局防止了在未预料到的稳定化到来时价格水平的 *下降*。

在关于高通货膨胀稳定化的多篇研究论文中，由方程 (14.12) 描述的货币供应的跳变被称为政府从实施维持永久性较低通货膨胀率的制度变化中获得的“**货币流通速度红利(the velocity dividend)**”。

关于 p 或 m 在 T_1 是否跳变的技术分析

我们已经注意到，如果预期的前向序列 $\mu_s = \bar{\mu}$ 对于 $s \geq t$ 是恒定的，那么 $\pi_t = \bar{\mu}$。

一个结果是在 T_1，m **或** p **必须在** T_1 **“跳变”**。

我们将研究这两种情况。

m_{T_1} 不跳变。

$$\begin{aligned} m_{T_1} &= m_{T_1-1} + \mu_0 \\ \pi_{T_1} &= \mu^* \\ p_{T_1} &= m_{T_1} + \alpha\pi_{T_1} \end{aligned}$$

简单地将序列 $t \leq T_1$ 和 $t > T_1$ 连接起来。

m_{T_1} 跳变。

我们重新设定 m_{T_1}，使得 $p_{T_1} = (m_{T_1-1} + \mu_0) + \alpha\mu_0$，其中 $\pi_{T_1} = \mu^*$。

然后，

$$m_{T_1} = p_{T_1} - \alpha\pi_{T_1} = (m_{T_1-1} + \mu_0) + \alpha\left(\mu_0 - \mu^*\right)$$

然后，我们计算剩余的 $T - T_1$ 期，其中 $\mu_s = \mu^*, \forall s \geq T_1$ 以及上述的初始条件 m_{T_1}。

我们现在技术上已经准备好讨论我们的下一个实验。

实验 2：一个未预见的突然稳定化

这个实验稍微偏离了我们“完美预见”假设的纯版本，假设像实验 1 中分析的那样，μ_t 的突然永久性减少是完全未预见的。

这种完全未预见的冲击通常被称为“MIT 冲击”。

心理实验涉及在时间 T_1 从初始的“延续路径” $\{\mu_t, \pi_t\}$ 切换到另一个路径，该路径涉及永久性较低的通货膨胀率。

初始路径： $\mu_t = \mu_0$ 对于所有 $t \geq 0$。因此，这个路径是 $\{\mu_t\}_{t=0}^{\infty}$；相关的 π_t 路径有 $\pi_t = \mu_0$。

修订的延续路径 其中 $\mu_0 > \mu^*$，我们通过设置 $\mu_s = \mu^*$ 对于所有 $s \geq T_1$ 来构建一个延续路径 $\{\mu_s\}_{s=T_1}^{\infty}$。对于 π 的完美预见延续路径是 $\pi_s = \mu^*$

为了捕捉时间 T_1 对 $\{\mu_t\}$ 过程的“完全未预见的永久性冲击”，我们只需将路径 2 下 $t \geq T_1$ 出现的 μ_t, π_t 粘接到路径 1 下 $t = 0, \ldots, T_1 - 1$ 出现的 μ_t, π_t 路径。

我们可以通过手工完成大部分 MIT 冲击计算。

因此，对于路径 1，$\pi_t = \mu_0, \forall t \in [0, T_1 - 1]$，而对于路径 2，$\mu_s = \mu^*, \forall s \geq T_1$。

我们现在转向实验 2，我们的“MIT 冲击”，完全未预见的突然稳定化。

我们这样设置，描述突然稳定化的 $\{\mu_t\}$ 序列与实验 1，预见的突然稳定化的那些序列是相同的。

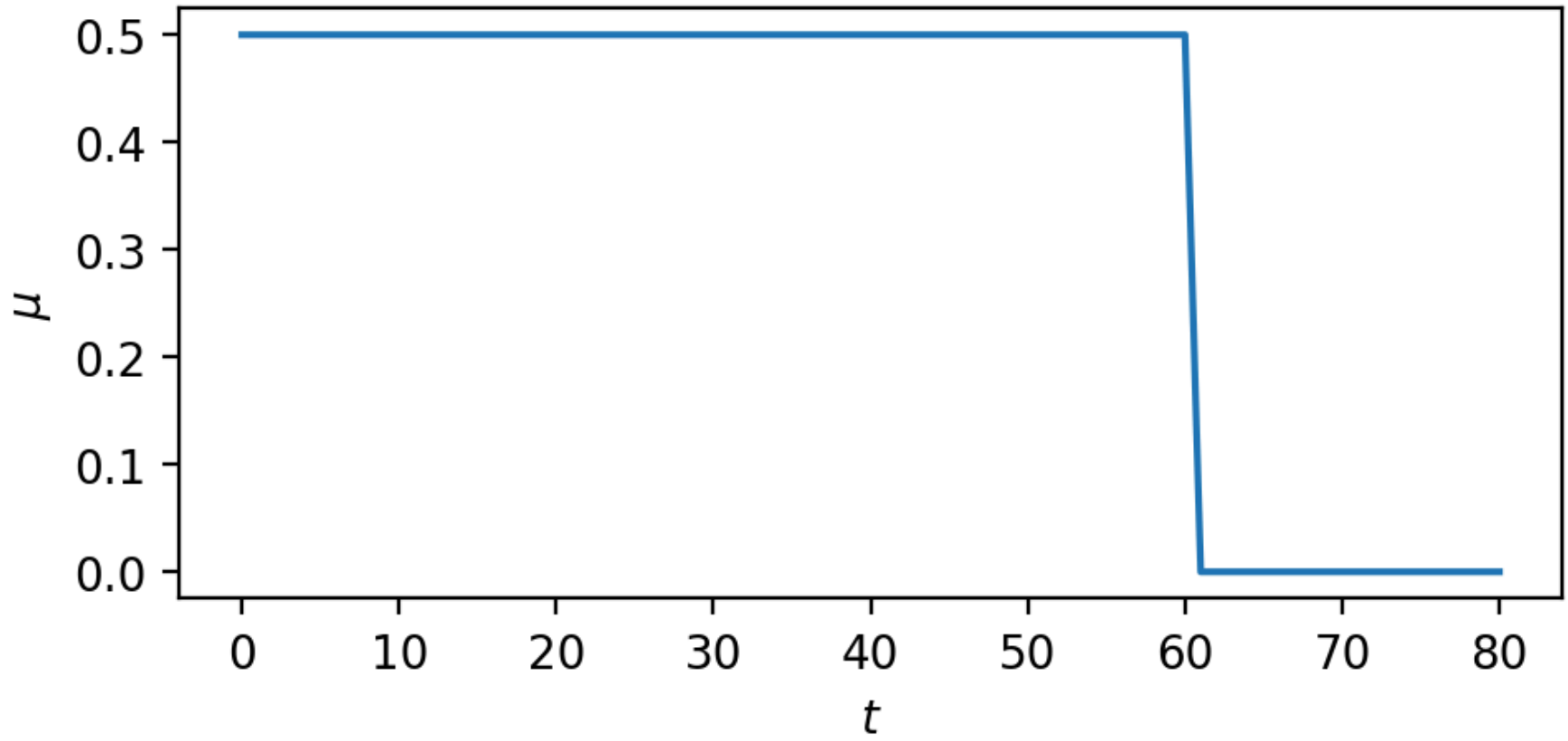
以下代码进行计算并绘制结果。

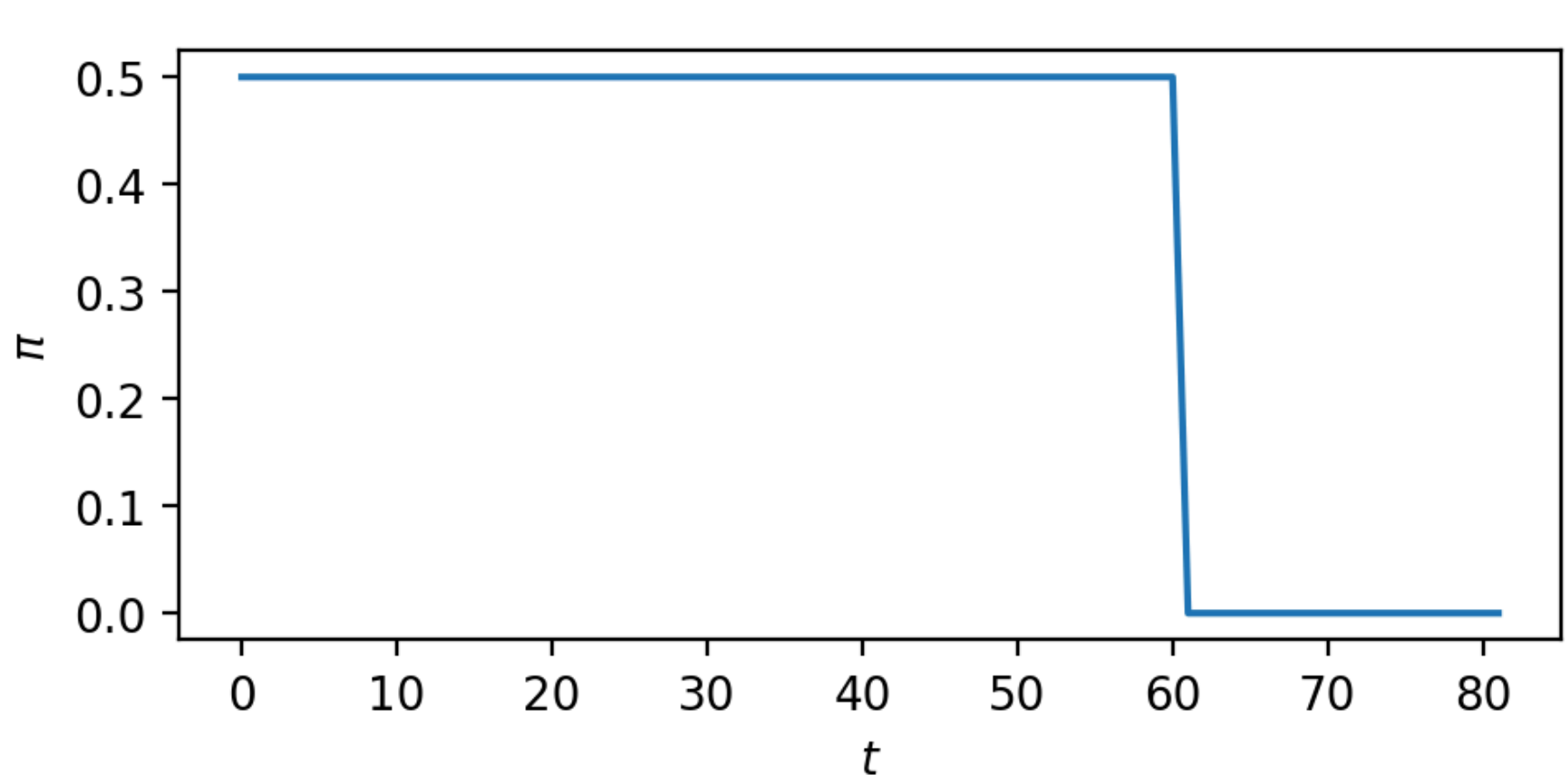
In [ ]:
```
# path 1
μ_seq_2_path1 = μ0 * np.ones(T+1)

cm1 = create_cagan_model(μ_seq=μ_seq_2_path1)
π_seq_2_path1, m_seq_2_path1, p_seq_2_path1 = solve(cm1, T)

# continuation path
μ_seq_2_cont = μ_star * np.ones(T-T1)

cm2 = create_cagan_model(m0=m_seq_2_path1[T1+1],
                         μ_seq=μ_seq_2_cont)
π_seq_2_cont, m_seq_2_cont1, p_seq_2_cont1 = solve(cm2, T-1-T1)
```

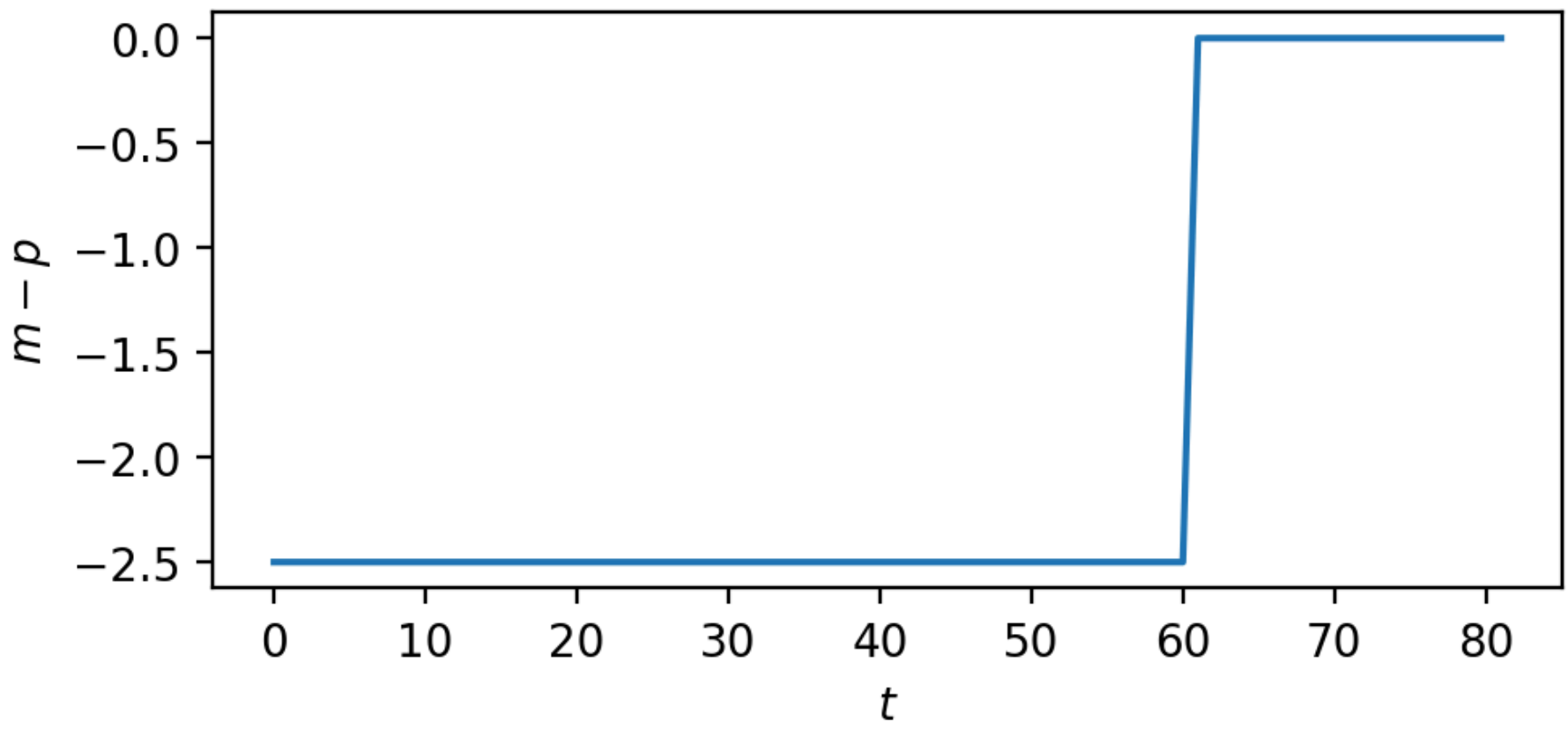
```
# regime 1 - simply glue π_seq, μ_seq
μ_seq_2 = np.concatenate((μ_seq_2_path1[:T1+1],
                          μ_seq_2_cont))
π_seq_2 = np.concatenate((π_seq_2_path1[:T1+1],
                          π_seq_2_cont))
m_seq_2_regime1 = np.concatenate((m_seq_2_path1[:T1+1],
                                  m_seq_2_cont1))
p_seq_2_regime1 = np.concatenate((p_seq_2_path1[:T1+1],
                                  p_seq_2_cont1))

# regime 2 - reset m_T1
m_T1 = (m_seq_2_path1[T1] + μ0) + cm2.α*(μ0 - μ_star)

cm3 = create_cagan_model(m0=m_T1, μ_seq=μ_seq_2_cont)
π_seq_2_cont2, m_seq_2_cont2, p_seq_2_cont2 = solve(cm3, T-1-T1)

m_seq_2_regime2 = np.concatenate((m_seq_2_path1[:T1+1],
                                  m_seq_2_cont2))
p_seq_2_regime2 = np.concatenate((p_seq_2_path1[:T1+1],
                                  p_seq_2_cont2))
```

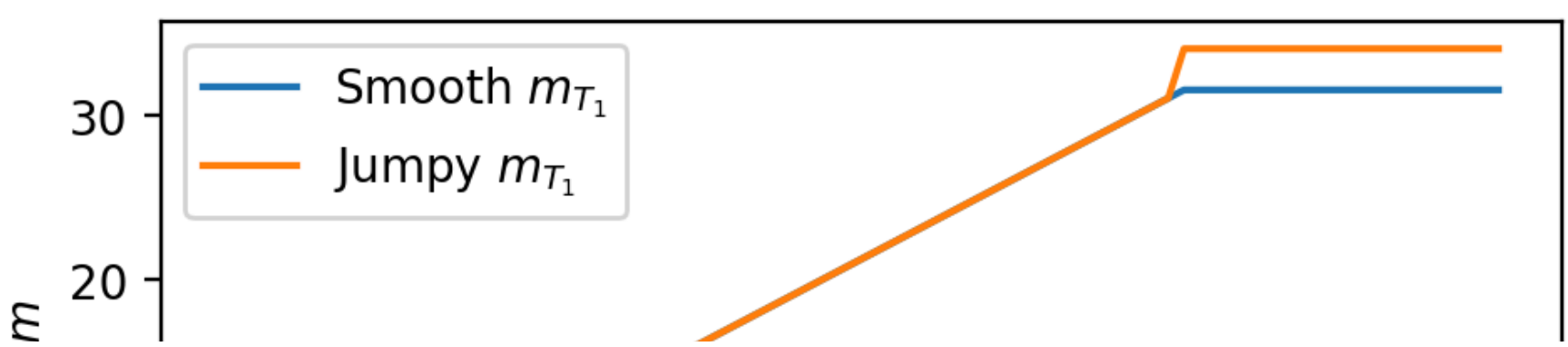
In [ ]:

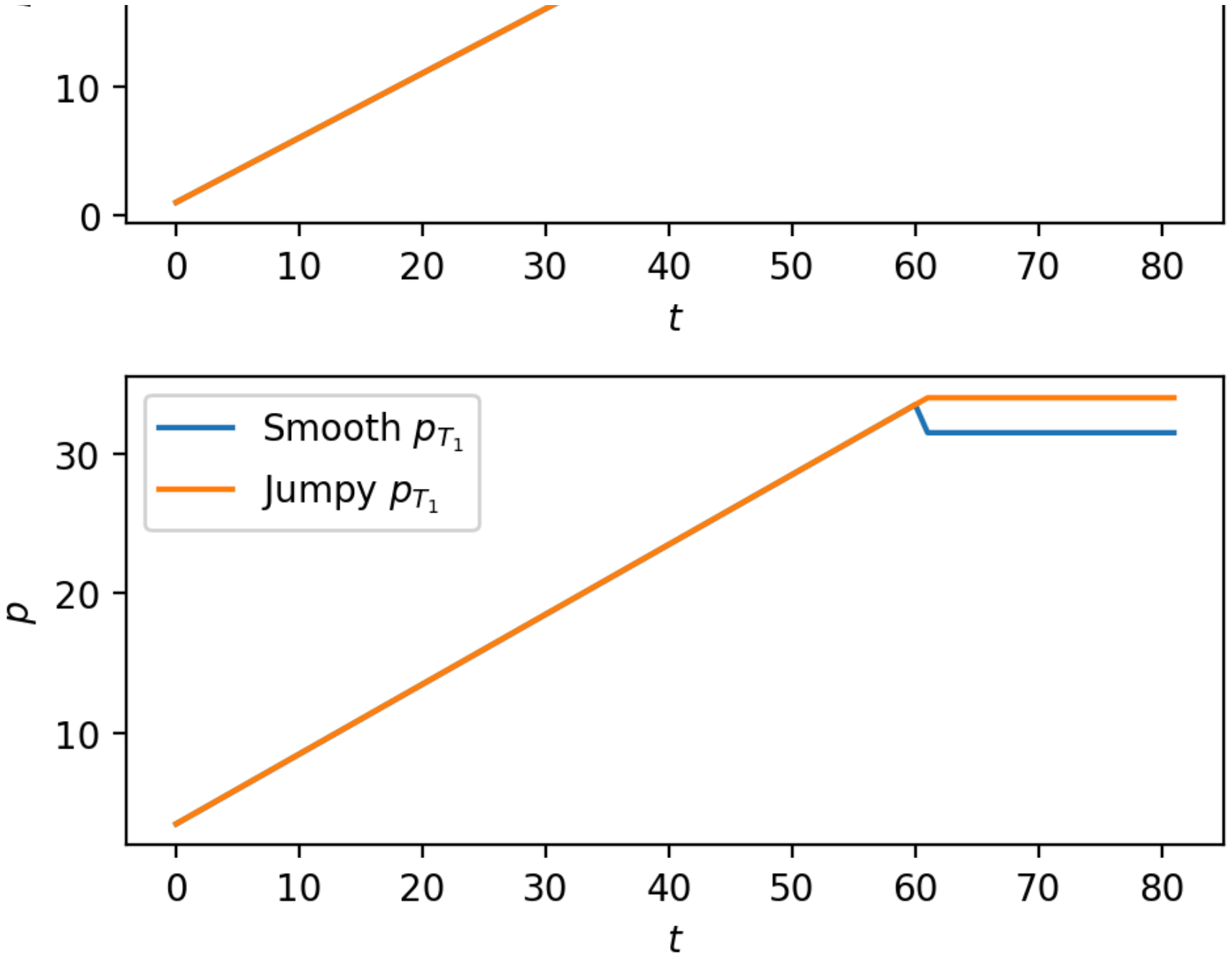
```
T_seq = range(T+2)

# plot both regimes
fig, ax = plt.subplots(5, 1, figsize=(5, 12), dpi=200)

# Configuration for each subplot
plot_configs = [
    {'data': [(T_seq[:-1], μ_seq_2)], 'ylabel': r'$\mu$'},
    {'data': [(T_seq, π_seq_2)], 'ylabel': r'$\pi$'},
    {'data': [(T_seq, m_seq_2_regime1 - p_seq_2_regime1)],
     'ylabel': r'$m - p$'},
    {'data': [(T_seq, m_seq_2_regime1, 'Smooth $m_{T_1}$'),
              (T_seq, m_seq_2_regime2, 'Jumpy $m_{T_1}$')],
     'ylabel': r'$m$'},
    {'data': [(T_seq, p_seq_2_regime1, 'Smooth $p_{T_1}$'),
              (T_seq, p_seq_2_regime2, 'Jumpy $p_{T_1}$')],
     'ylabel': r'$p$'}
]

def experiment_plot(plot_configs, ax):
    # Loop through each subplot configuration
    for axi, config in zip(ax, plot_configs):
        for data in config['data']:
            if len(data) == 3:  # Plot with label for legend
                axi.plot(data[0], data[1], label=data[2])
                axi.legend()
            else:  # Plot without label
                axi.plot(data[0], data[1])
        axi.set_ylabel(config['ylabel'])
        axi.set_xlabel(r'$t$')
    plt.tight_layout()
    plt.show()

experiment_plot(plot_configs, ax)
```


0.5
0.4
0.3
0.2
0.1
0.0
μ
0
10
20
30
40
50
60
70
80
t


0.5
0.4
0.3
0.2
0.1
0.0
π
0
10
20
30
40
50
60
70
80
t


0.0
−0.5
−1.0
−1.5
−2.0
−2.5
$m - p$
0
10
20
30
40
50
60
70
80
t


Smooth m_{T_1}
Jumpy m_{T_1}
30
20
m



我们邀请您将这些图表与上述实验 1 中分析的预见性稳定化的相应图表进行比较。

注意第二面板中的通货膨胀图表现在与顶部面板中的货币增长率图表完全相同，以及第三面板中描绘的实际余额对数在时间 T_1 向上跳变。

底部两个面板绘制了 m 和 p 在两种可能的方式下的表现，这些方式是 m_{T_1} 可能的调整方式，以满足 T_1 时 $m - p$ 的向上跳变的要求。

- 橙色线让 m_{T_1} 向上跳变，以确保对数价格水平 p_{T_1} 不会下降。
- 蓝色线让 p_{T_1} 下降，同时阻止货币供应量跳变。

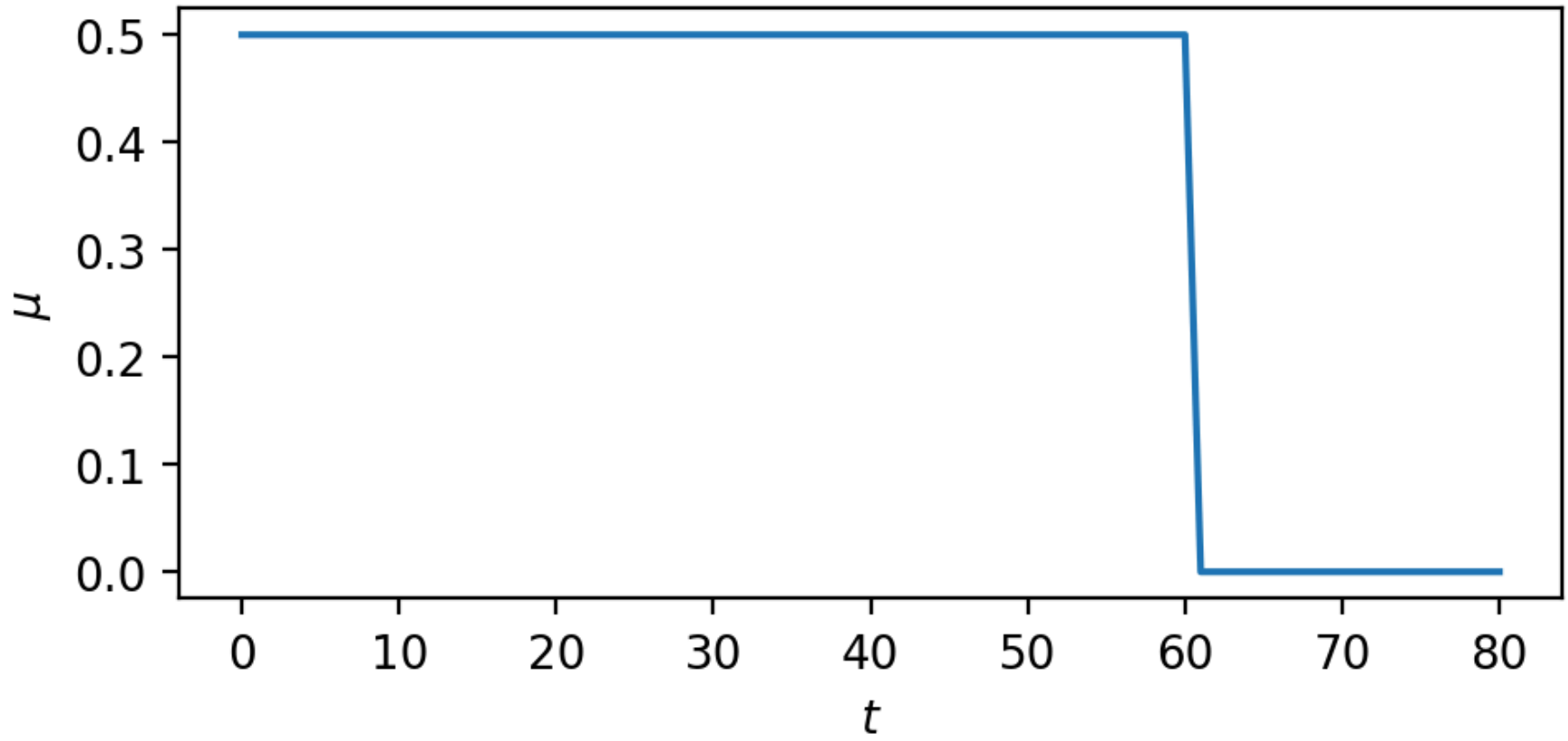
当橙色线政策实施时政府在做什么？

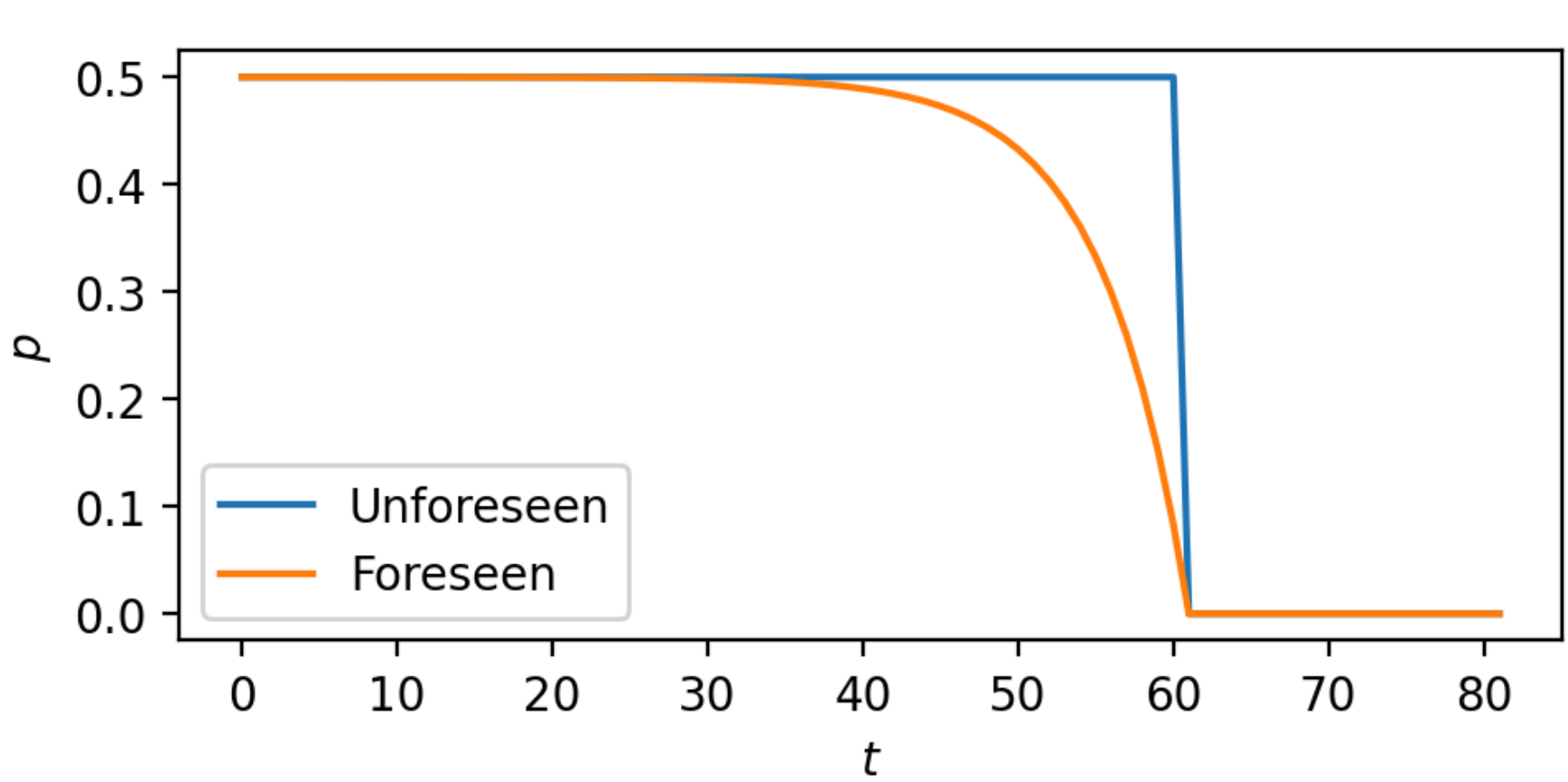
政府印制货币来为支出提供资金，它从货币供应增长率永久性减少引起的实际余额需求增加中获得的“货币流通速度红利”。

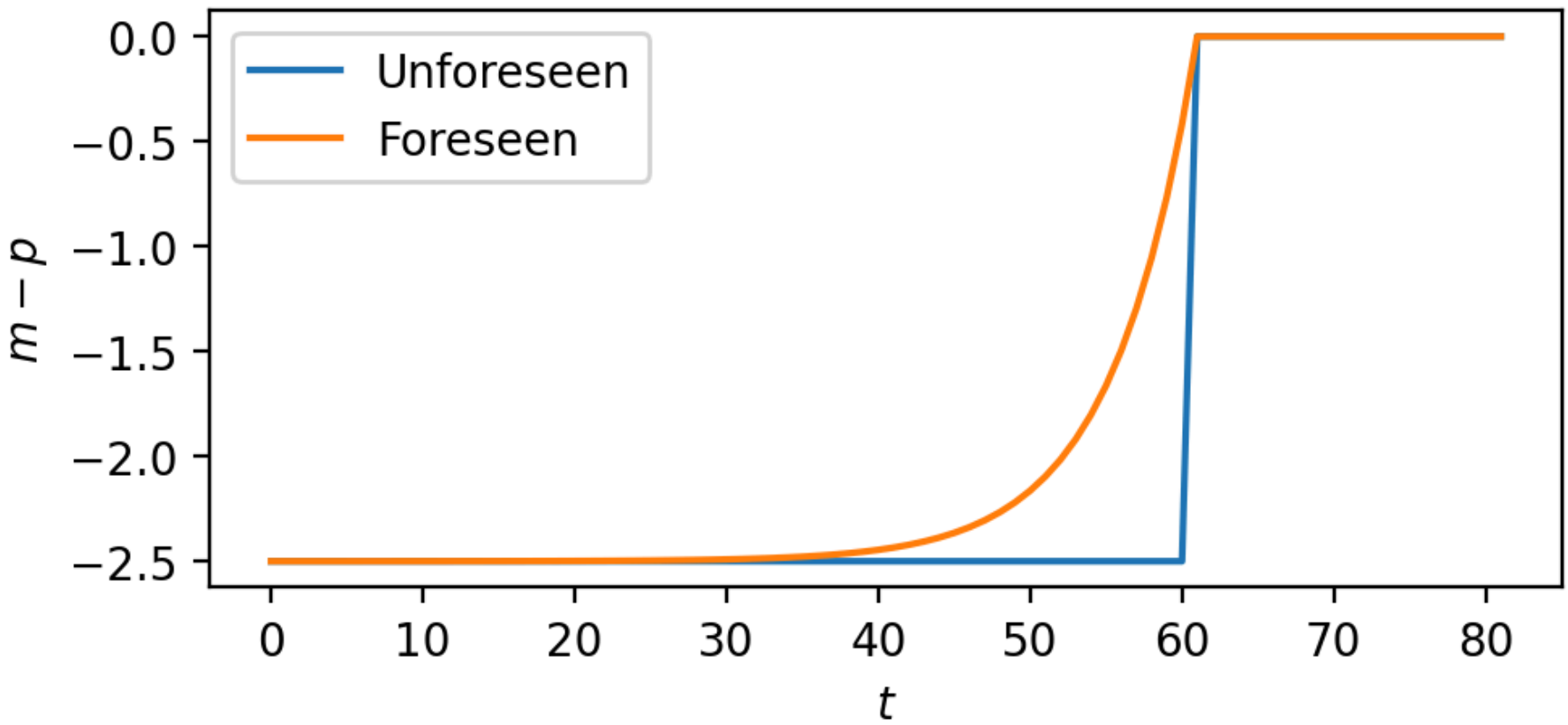
接下来的代码生成了一个多面板图表，其中包括了实验 1 和实验 2 的结果。

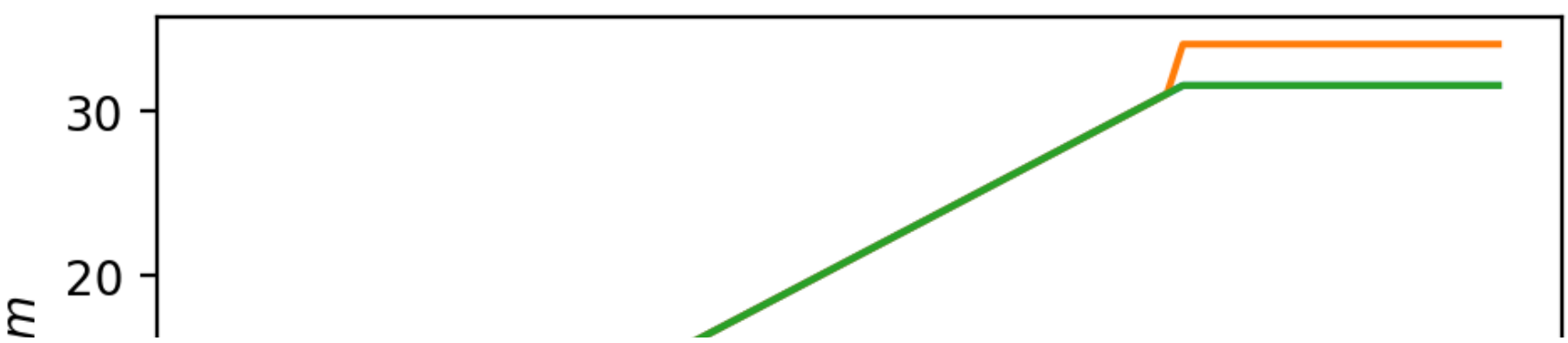
这使我们能够评估理解 $t = T_1$ 时 μ_t 的突然永久性下降是否被完全预见，如实验 1 中那样，或者完全未预见，如实验 2 中那样。

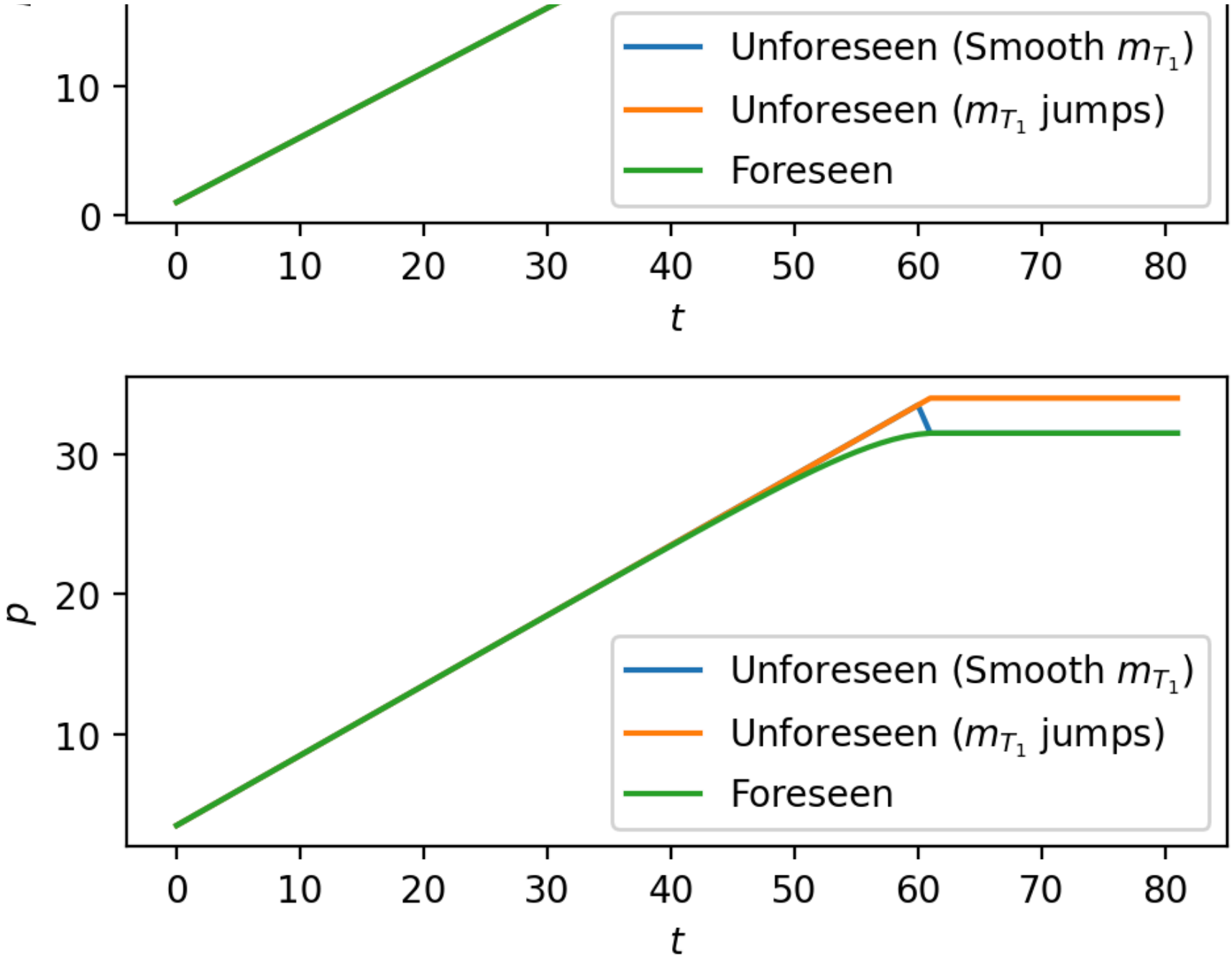
In [ ]:
```
# compare foreseen vs unforeseen shock
fig, ax = plt.subplots(5, figsize=(5, 12), dpi=200)

plot_configs = [
    {'data': [(T_seq[:-1], μ_seq_2)], 'ylabel': r'$\mu$'},
    {'data': [(T_seq, π_seq_2, 'Unforeseen'),
              (T_seq, π_seq_1, 'Foreseen')], 'ylabel': r'$p$'},
    {'data': [(T_seq, m_seq_2_regime1 - p_seq_2_regime1, 'Unforeseen'),
              (T_seq, m_seq_1 - p_seq_1, 'Foreseen')], 'ylabel': r'$m - p$'},
    {'data': [(T_seq, m_seq_2_regime1, 'Unforeseen (Smooth $m_{T_1}$)'),
              (T_seq, m_seq_2_regime2, 'Unforeseen ($m_{T_1}$ jumps)'),
              (T_seq, m_seq_1, 'Foreseen')], 'ylabel': r'$m$'},
    {'data': [(T_seq, p_seq_2_regime1, 'Unforeseen (Smooth $m_{T_1}$)'),
          (T_seq, p_seq_2_regime2, 'Unforeseen ($m_{T_1}$ jumps)'),
          (T_seq, p_seq_1, 'Foreseen')], 'ylabel': r'$p$'}
]

experiment_plot(plot_configs, ax)
```


0.5
0.4
0.3
0.2
0.1
0.0
μ
0
10
20
30
40
50
60
70
80
t


0.5
0.4
0.3
0.2
0.1
0.0
p
Unforeseen
Foreseen
0
10
20
30
40
50
60
70
80
t


0.0
−0.5
−1.0
−1.5
−2.0
−2.5
$m - p$
Unforeseen
Foreseen
0
10
20
30
40
50
60
70
80
t


30
20
m



实验 3 预见的逐渐稳定化

与实验 1 中研究的预见的突然稳定化不同，研究预见的逐渐稳定化的后果也很有趣。

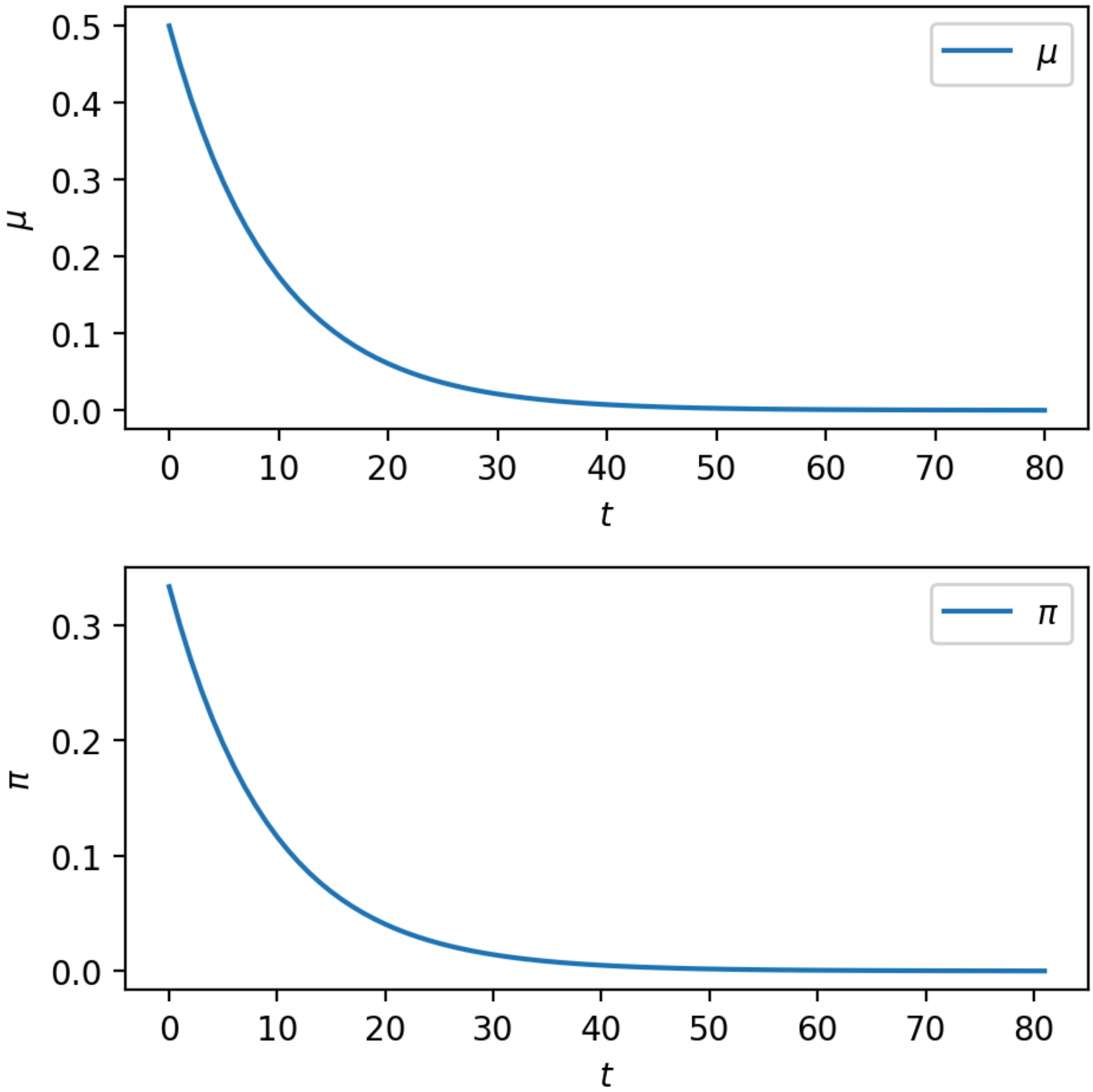
因此，假设 $\phi \in (0,1)$，$\mu_0 > \mu^*$，并且对于 $t = 0, \ldots, T-1$

$$\mu_t = \phi^t \mu_0 + (1 - \phi^t)\mu^*.$$

接下来我们进行一个实验，在该实验中，货币供应增长率的下降是**完全预见的 *逐渐* 减少**。

以下代码进行计算并绘制结果。

In [ ]:
```
# parameters
ϕ = 0.9
μ_seq_stab = np.array([ϕ**t * μ0 + (1-ϕ**t)*μ_star for t in range(T)])
μ_seq_stab = np.append(μ_seq_stab, μ_star)

cm4 = create_cagan_model(μ_seq=μ_seq_stab)

π_seq_4, m_seq_4, p_seq_4 = solve(cm4, T)

sequences = (μ_seq_stab, π_seq_4,
             m_seq_4 - p_seq_4, m_seq_4, p_seq_4)
plot_sequences(sequences, (r'$\mu$', r'$\pi$',
                           r'$m - p$', r'$m$', r'$p$'))
```


0.5
0.4
0.3
0.2
0.1
0.0
μ
0
10
20
30
40
50
60
70
80
t
μ
0.3
0.2
0.1
0.0
π
0
10
20
30
40
50
60
70
80
t
π
0.0
−0.5
−1.0
−1.5
$m - p$
0
10
20
30
40
50
60
70
80
t
$m - p$
6
5
4
m