摘　要

本报告选择 2019 年 12 月 2 日作为起始日期，根据上证 50ETF 欧式看涨期权和上证 50ETF 欧式看跌期权的市场历史数据计算出不同行权价格对应的隐含波动率，画出波动率微笑曲线，并通过线性插值的方法得到不同行权价格对应的隐含波动率。

本报告的主题是隐含二叉树的构建。第一步从理论出发，在构建隐含二叉树的过程中，以中心节点为界，将每一层枝干分为上半部分树和下半部分树。首先分别讨论了节点数为奇数和偶数两种情况下中心节点股票价格的计算方法，接下来分别对上半部分树和下半部分树各节点上的股票价格计算公式进行讨论，并证明了股票价格的计算公式，同时讨论了当计算得到的股票价格不满足无套利条件时对其的处理方法。第二步从实证出发，根据得到的隐含波动率，对 2019 年 12 月 2 日的上证 50ETF 收盘价构建隐含二叉树并简要讨论了隐含二叉树的作用。

关键词: 隐含波动率, 隐含二叉树

目 录

第一章　引言

本报告第二章首先对期权定价相关概念（一维布朗运动、*Itô* 公式、一般二叉树模型和隐含波动率）进行简要介绍。

第三章主要介绍了隐含二叉树的理论，首先对构建隐含二叉树所需要的 Arrow-Debreu 价格进行了介绍；接下来分别介绍了枝干层节点数为奇数和偶数时中心节点股票价格的计算公式，根据中心节点将树分为上半部分树和下半部分树，分别对它们节点上的股票价格的计算公式进行介绍，得到计算隐含二叉树各节点股票价格的一般公式；最后讨论了当得到的股票价格不满足无套利条件时对股票价格的处理。

第四章是实证分析部分，根据已有的数据，计算上证 50EFT 欧式看涨期权和欧式看跌期权的隐含波动率，并通过线性插值的方法得到不同行权价格对应的隐含波动率，然后具体介绍了第一层至第三层枝干的构建过程，最后给出了构建的从 2019 年 12 月至 2021 年 12 月的以月为时间间隔的隐含二叉树结果，并对隐含二叉树的作用进行简单讨论。

第二章　相关理论简介

2.1　一维布朗运动

定义 2.1.1 (一维标准布朗运动). 给定概率空间 $(\Omega, \mathcal{F}, P)$ 及其上的 σ-域流 $\mathbb{F}$，定义在该空间的满足以下条件的连续时间一维适应过程 $(B_t)_{t\in R_+}$：

1) $B_0 = 0, a.s.$;

2) B 是独立增量过程，即对 $\forall s \le t$，$B_t - B_s$ 与 $B_u (u \le s)$ 独立；且 $B_t - B_s$ 独立于 $\mathcal{F}_s$;

3) B 具有平稳增量，即对 $\forall s \le t$，$B_t - B_s \overset{d}{\sim} B_{t-s}$，且 $B_t - B_s \sim N(0, t-s), 0 \le s \le t$;

为一维标准布朗运动或维纳过程。

定义 2.1.2 (算数布朗运动). 若随机过程 $\{X_t\}_{t\ge 0}$ 可以表示为：

$$dX_t = \mu dt + \sigma dB_t, \tag{2.1}$$

其中，μ, σ 为常数，$\{B_t\}_{t\ge 0}$ 为一维标准布朗运动，则称 $\{X_t\}_{t\ge 0}$ 为算数布朗运动。

随机微分方程2.1的解为：

$$X_t = X_0 + \mu t + \sigma B_t.$$

证明. 由式 $dX_t = \mu dt + \sigma dB_t$, 可得

$$\int_0^t dX_s = \int_0^t \mu ds + \int_0^t \sigma dB_s$$
$$X_t - X_0 = \mu t + \sigma (B_t - B_0)$$

由于 $B_0 = 0$，因此整理可得

$$X_t = X_0 + \mu t + \sigma B_t.$$

定义 2.1.3 (几何布朗运动). 若随机过程 $\{X_t\}_{t\ge 0}$ 可以表示为：

$$dX_t = \mu X_t dt + \sigma X_t dB_t, \tag{2.2}$$

其中，μ, σ 为常数，$\{B_t\}_{t\ge 0}$ 为一维标准布朗运动，则称 $\{X_t\}_{t\ge 0}$ 为几何布朗运动。

随机微分方程2.2的解为:

$$X_t = X_0 \exp\left\{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma B_t\right\}.$$

证明.

$$dX_t = \mu X_t dt + \sigma X_t dB_t$$

由 *Itô* 引理可得

$$\begin{aligned} d\ln X_t &= \frac{1}{X_t}dX_t - \frac{1}{2X_t^2}d\langle X\rangle_t \\ &= \mu dt + \sigma dB_t - \frac{\sigma^2}{2}dt \\ &= \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dB_t \\ \int_0^t d\ln X_s &= \int_0^t \left(\mu - \frac{\sigma^2}{2}\right)ds + \int_0^t \sigma dB_s \\ \ln X_t - \ln X_0 &= \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma(B_t - B_0) \end{aligned}$$

由于 $B_0 = 0$，因此整理可得

$$X_t = X_0 \exp\left\{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma B_t\right\}.$$

定理 2.1.1 (布朗运动的基本性质). 设 $(B_t)_{t\in R_+}$ 是一维标准布朗运动，则

1. $(B_t)_{t\in R_+}$ 是 Gauss 过程。即对某个指标集 $\mathcal{T}$，设 $B = \{B_t\}_{t\in\mathcal{T}}$ 是带流概率空间 $(\Omega, \mathcal{F}, \mathbb{F}, P)$ 上的一个随机过程，对 $\forall(t_1, t_2, \cdots, t_n) \in \mathcal{T}, n \geq 1$ 都有

$$\left(B_{t_1}, B_{t_2}, \cdots, B_{t_n}\right) \sim N(\boldsymbol{\mu}, \Sigma).$$

其中 $\boldsymbol{\mu}, \Sigma$ 分别是随机向量 $\left(B_{t_1}, B_{t_2}, \cdots, B_{t_n}\right)$ 的期望向量和协方差矩阵。

2. 对 $\forall t < s$,

$$\begin{aligned} P(y, s|x, t) :=& P(B_s \leq y|B_t = x) = P\left(B_s - B_t + B_t \leq y|B_t = x\right) \\ =& P\left(B_s - B_t \leq y - x|B_t = x\right) = P\left(B_{s-t} \leq y - x\right) \\ =& \Phi\left(\frac{y-x}{\sqrt{s-t}}\right). \end{aligned}$$

其中，Φ 为标准正态分布的分布函数。

2.2 *Itô* 公式和 *Itô* 积分

定义 2.2.1 (*Itô* 过程). 给定带流概率空间 $(\Omega, \mathcal{F}, \mathbb{F}, P)$ 上的布朗运动 B，假设随机过程 X 具有连续轨道，X 被称为伊藤过程，如果它具有如下的积分形式：

$$X_T = X_0 + \int_0^T \mu(t)dt + \int_0^T \sigma(t)dB_t,\ T \geq 0. \tag{2.3}$$

其中，$\mu(t), \sigma(t)$ 是关于 t 的函数，B_t 是一维标准布朗运动。

伊藤过程 X 的微分形式为：

$$\mathrm{d}X_t = \mu(t)dt + \sigma(t)\mathrm{d}B_t \tag{2.4}$$

定理 2.2.1 (*Itô* 公式). 设 X 是具有形式 (3.1) 的伊藤过程，$f(t,x)$ 是一个二元可微实值函数，则过程 $f(t, X_t)$ 仍为 *Itô* 过程，且对 $\forall t \geq 0$,

$$\mathrm{d}f(t,X_t) = \frac{\partial f}{\partial t}(t,X_t)\mathrm{d}t + \frac{\partial f}{\partial X_t}(t,X_t)\mathrm{d}X_t + \frac{1}{2}\frac{\partial^2 f}{\partial X_t^2}(t,X_t)\mathrm{d}\langle X\rangle_t.$$

其中，$\langle X\rangle_t$ 为 X 的可料二次变差过程（也称尖括号过程）。

引理 2.2.1 (*Itô* 引理).

$$\mathrm{d}X_t = a(t,X_t)\mathrm{d}t + b(t,X_t)\mathrm{d}X_t.$$

设 $f(t,x)$ 是一个二元可微实值函数，令 $f_t := f(t,X_t)$，则

$$\begin{aligned} df_t &= \left[\frac{\partial f_t}{\partial t} + \frac{1}{2}b^2(X_t,t)\frac{\partial^2 f_t}{\partial X_t^2}\right]dt + \frac{\partial f_t}{\partial X_t}dX_t \\ &= \left[\frac{\partial f_t}{\partial t} + a(t,X_t)\frac{\partial f_t}{\partial X_t} + \frac{1}{2}b^2(t,X_t))\frac{\partial^2 f_t}{\partial X_t^2}\right]dt + b(t,X_t)\frac{\partial f_t}{\partial X_t}dB_t. \end{aligned}$$

2.3 股票价格过程

假设在风险中性测度下，股票价格 S_t 服从几何布朗运动，即

$$dS_t = \mu S_t dt + \sigma S_t dB_t, \tag{2.5}$$

其中，μ 为漂移项（也称为期望回报率），σ 为波动率，$\sigma \geq 0$，且 μ, σ 都为常数。B_t 服从一维标准布朗运动。

令 $f_t = f(S_t, t) = \ln S_t$，则

$$\frac{\partial f_t}{\partial t} = 0,\ \frac{\partial f_t}{\partial S_t} = \frac{1}{S_t},\ \frac{\partial^2 f_t}{\partial S_t^2} = -\frac{1}{S_t^2},$$

根据伊藤引理得，

$$\begin{aligned} d\ln S_t &= df_t \\ &= \left(\frac{\partial f_t}{\partial t} + \mu S_t \frac{\partial f_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 f_t}{\partial S_t^2}\right) dt + \sigma S_t \frac{\partial f_t}{\partial S_t} dB_t \\ &= \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dB_t. \end{aligned} \tag{2.6}$$

式（2.6）与式（2.4）具有相同的形式，因此，$f_t = \ln S_t$ 也是一个伊藤过程，则，

$$\begin{aligned} \ln S_T &= \ln S_0 + \int_0^T (\mu - \frac{1}{2}\sigma^2) dt + \int_0^T \sigma dB_t \\ &= \ln S_0 + (\mu - \frac{1}{2}\sigma^2) T + \sigma B_T. \end{aligned} \tag{2.7}$$

于是，

$$S_T = S_0 e^{(\mu - \frac{1}{2}\sigma^2)T + \sigma B_T}.$$

由于 B_T 是标准布朗运动，则

$$\ln\left(\frac{S_T}{S_0}\right) \sim N\left(\mu - \frac{\sigma^2}{2}, \sigma^2 T\right). \tag{2.8}$$

将式 (2.7) 转换为离散形式，得到

$$\ln S_{t+\delta t} - \ln S_t = \left(\mu - \frac{1}{2}\sigma^2\right)\delta t + \sigma B_{\delta t}.$$

2.4 Black-Scholes 模型

Black-Scholes 模型假设市场是完备的，即市场满足以下条件：

1) 假设股票价格 S_t 服从几何布朗运动，既满足式 (2.5) 的随机微分方程；

2) 无风险利率 r 是已知的常数；

3) 股票无股息派发；

4) 不存在无风险套利机会；

5) 对冲投资组合时没有交易价格；

6) 股票交易连续，且股票价格的变动也是连续；

7) 允许卖空，且股票可分。

假设 $V_t = V(S_t, t, E, T)$ 是一个欧式期权的价值，其中，t 是期权的起始日期，T 是期权的终止日期，E 期权的交割价格，股票 S_t 为期权的标的资产并且无股息派发，V_t 是一个二元可微实值函数。

在 t 时刻做多一份期权合约 V_t，同时做空 a 股 S_t，从而构造组合 Π_t,

$$\Pi_t = V_t - aS_t.$$

由伊藤引理可得，

$$dV_t = \left(\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2}\right)dt + \frac{\partial V_t}{\partial S_t}dS_t,$$

则，

$$\begin{aligned} d\Pi_t =& dV_t - adS_t \\ =& \left(\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2}\right)dt + \left(\frac{\partial V_t}{\partial S_t} - a\right)dS_t. \end{aligned} \tag{2.9}$$

式 (2.9) 可以看作由确定项和随机项组成，随机项为

$$\left(\frac{\partial V_t}{\partial S_t} - a\right)dS_t.$$

为了消除随机项，令

$$\Delta := a = \frac{\partial V_t}{\partial S_t}.$$

这就是 Δ 对冲，并且通过这种方法构造的组合 Π_t 为 Delta 风险中性的组合。

又由无套利假设可得，

$$d\Pi_t = r\Pi dt.$$

因此可以得到 Black-Scholes 方程：

$$\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + rS_t \frac{\partial V_t}{\partial S_t} - rV_t = 0.$$

2.5 二叉树模型

二叉树模型是期权定价的一种有效方法，它建立在以下假设基础之上：

1) 市场是无摩擦的，即无税、无交易成本，所有资产可无限细分，无卖空限制；

2) 投资者和购买者之间无信息差，信息充分共享；

3) 允许完全使用卖空得到的资金；

4) 允许以无风险利率借入和借出资金；

5) 假设股票价格 S_t 服从几何布朗运动，既满足式 (2.5) 的随机微分方程；

6) 不存在无风险套利的机会。

用 $V(S, t_0, K, T)$ 表示一个标的资产为 S 的期权价格，简记为 V，S 无股息派发，期权的生效日期为当前时刻 $t_0 = 0$, 到期日为 T. 二叉树模型的基本思想是将时间区间 $[0, T]$ 均分为 n 份，即 $0 < \frac{T}{n} < \frac{2T}{n} < \cdots < \frac{nT}{n} = T$, 讨论每一个 $\frac{T}{n}$ 时间间隔的资产价格的状态。

二叉树模型分为单期二叉树和多期二叉树模型，但是它们的构建思路是基本一致的，接下来对单期二叉树模型进行简要介绍。

当 $n = 1$ 时，下图展示了单期二叉树的结构，其中 p_u 表示期初价格为 S 的股票在 T 时刻价格上升到 S_u 的转移概率，反之 p_d 表示期初价格为 S 的股票在 T 时刻价格下降到 S_d 的转移概率。同时，为了保证市场是无套利机会的，u 和 v 要满足 $u > e^{rT} > d$，其中 r 表示无风险利率。

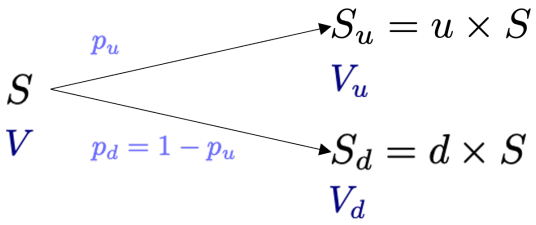


图 2.1. 单期二叉树模型结构示意图

构建一个投资组合，做空一份 V，做多 Δ 股 S，则投资组合的起初价值为：

$$\Phi(0) = -V + \Delta S.$$

在 T 时刻，若股票价格上升到 S_u, 投资组合到期日的价值为：

$$\Phi_u(T) = -V_u + \Delta S_u.$$

在 T 时刻，若股票价格下降到 S_d, 投资组合到期日的价值为：

$$\Phi_d(T) = -V_d + \Delta S_d.$$

由无套利假设可以得到，不论股票的价格是上涨还是下跌，投资组合的价值是等价的，因此

$$\begin{aligned}
\Phi_u(T) &= \Phi_d(T) \\
-V_u + \Delta S_u &= -V_d + \Delta S_d \\
\Delta &= \frac{V_u - V_d}{S_u - S_d}.
\end{aligned}$$

同样由无套利假设得到 $e^{rT}\Phi(0) = \Phi_u(T) = \Phi_d(T)$. 则

$$\begin{aligned}
e^{rT}\left(-V + \Delta S\right) &= -V_u + \Delta S_u \\
&= -V_u + \frac{V_u - V_d}{S_u - S_d} S_u \\
&= \frac{-S_u V_u + S_d V_u + S_u V_u - S_u V_d}{S_u - S_d}.
\end{aligned}$$

令 $S_u := u \cdot S$, $S_d := d \cdot S$, 则

$$e^{rT}\left(-V + \frac{V_u - V_d}{u - d}\right) = \frac{dV_u - uV_d}{u - d}.$$

化简，得

$$\begin{aligned}
V &= \frac{V_u - V_d}{u - d} - e^{-rT}\frac{dV_u - uV_d}{u - d} \\
&= e^{-rT}\left[\frac{e^{rT} - d}{u - d}V_u + \frac{u - e^{rT}}{u - d}\right] \\
&\triangleq e^{-rT}\left[p_u V_u + p_d V_d\right].
\end{aligned}$$

其中，

$$\begin{aligned}
p_u &= \frac{e^{rT} - d}{u - d} \\
p_d &= 1 - p_u.
\end{aligned}$$

因此，只要知道 u 和 d 的值，就可以计算期权在期初的价值。

由于股票价格 S 服从几何布朗运动，既满足式 (2.5) 的随机微分方程。则在相同时间间隔下，二叉树模型下股票价格的变化率的一阶矩和二阶矩应与布朗运动下股票价格的变化率的一阶矩和二阶矩相同。

由式 (2.8) 可知

$$\begin{cases} p_u u + (1-p_u)d = e^{rT} \\ pu^2 + (1-p)d^2 - e^{2rT} = e^{2rT}\left(e^{\sigma^2 T} - 1\right). \end{cases}$$

方程组中包含了 3 个未知量和 2 个方程，为了得到解，引入 CRR 假定：

$$u = \frac{1}{d}.$$

解方程组可以得到，

$$\begin{cases} u = \dfrac{1}{d} = e^{\sigma\sqrt{T}} \\ p_u = \dfrac{e^{rT} - d}{u - d}. \end{cases}$$

证明. 由 $p_u u + (1-p_u)d = e^{rT}$ 可得

$$p_u = \frac{e^{rT} - d}{u - d}.$$

同时将 $u = \frac{1}{d}$ 和 $p_u = \frac{e^{rT}-d}{u-d}$ 带入方程 $pu^2 + (1-p)d^2 - e^{2rT} = e^{rT}\left(e^{\sigma^2 T} - 1\right)$ 中，得

$$\begin{aligned} \frac{u^3 e^{rT} - u^2}{u^2 - 1} + \frac{u - e^{rT}}{u^3 - u} &= e^{2rT+\sigma^2 T} \\ \frac{\left(u^2+1\right)\left(u^2-1\right)e^{rT} - u\left(u^2-1\right)}{u(u^2-1)} &= e^{2rT+\sigma^2 T} \\ \frac{(u^2+1)e^{rT} - u}{u} &= e^{2rT+\sigma^2 T}. \end{aligned}$$

整理可得

$$e^{rT}u^2 - \left(e^{2rT+\sigma^2 T} + 1\right)u + e^{rT} = 0.$$

解方程得到

$$u = \frac{e^{2\left(2r+\sigma^2\right)T} + 1 + \sqrt{\left(e^{\left(2r+\sigma^2\right)T} + 1\right)^2 - 4e^{2rT}}}{2e^{rT}}.$$

泰勒展开得到

$$u = 1 + \sigma\sqrt{T} + O\left(\sqrt{T}\right).$$

因此可得

$$u = e^{\sigma\sqrt{T}}.$$

以及

$$d = \frac{1}{u} = e^{-\sigma\sqrt{T}}.$$

证毕。

2.6 隐含波动率

2.6.1 隐含波动率介绍

隐含波动率是指将根据期权定价模型（如 Black-Scholes 模型）得到的理论期权价格 V 与期权的市场价格 V_m 建立一个等式关系，来反解得到的波动率。

波动率曲线描述率期权的行权价格和隐含波动率之间的关系，以行权价格为横轴、隐含波动率为纵轴。波动率微笑是指波动率曲线呈现出两边高中间低、开口向上的形状。

在波动率曲线的基础上进一步考虑期限维度，即可得到隐含波动率曲面，隐含波动率曲面可以描述不同执行价格和不同期限结构所对应的隐含波动率。

2.6.2 隐含波动率的求解

本报告在实证分析的部分使用牛顿-拉普森迭代法计算期权的隐含波动率，因此这部分以欧式看涨期权为例对牛顿-拉普森迭代法进行检验介绍。

用 $C(S,t,K,T)$ 表示无股息派发的标的资产为 S，起始时刻为 t，执行价格为 K，终止日期为 T 的欧式看涨期权的理论价格，用 r 表示无风险利率，σ 表示波动率，则有

$$\begin{cases} C(S,t,K,T) = SN(d_1) - Ke^{-r(T-t)}N(d_2) \\ d_1 = \dfrac{\ln\frac{S}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \\ d_2 = d_1 - \sigma\sqrt{T-t} \end{cases}. \tag{2.10}$$

其中，$N(\cdot)$ 表示标准正态分布的分布函数。

假设 $f(\sigma)$ 是波动率为 σ 时的欧式看涨期权的理论价格 $C(S,t,K,T)$ 与欧式看涨期权的市场价格 $\hat{C}$ 之间的差异，即

$$\begin{aligned} f(\sigma) &= C(S,t,K,T) - \hat{C} \\ &= SN(d_1) - Ke^{-r(T-t)}N(d_2) - \hat{C}. \end{aligned}$$

于是可以求得

$$\begin{aligned} v = f^{'}(\sigma) &= SN^{'}(d_1)\sqrt{T-t} \\ &= \frac{S}{\sqrt{2\pi}} e^{-\frac{d_1^2}{2}} \sqrt{T-t}. \end{aligned}$$

利用牛顿-拉普森迭代法计算隐含波动率时，首先需要给波动率赋予一个初值 σ_0（一般通过历史波动率给出一个初值的预测）。通过式 (2.10) 计算得到欧式看涨期权的初始理论价格 $C_0(S,t,K,T)$, 于是得到误差

$$\epsilon_0 = |C_0(S,t,K,T) - \hat{C}|.$$

设定误差的阈值为 ϵ，若 $\epsilon_0 \geq \epsilon$, 则令

$$\sigma_1 = \sigma_0 - \frac{f(\sigma_0)}{f^{'}(\sigma_0)}.$$

则通过计算得到 $C_1(S,t,K,T)$，以及误差

$$\epsilon_1 = |C_1(S,t,K,T) - \hat{C}|.$$

若 $\epsilon_1 < \epsilon$，则 σ_1 即为所求的隐含波动率；若 $\epsilon_1 \geq \epsilon$，则继续重复上述步骤，直至误差低于阈值 ϵ，对应的波动率即为所求的隐含波动率。

第三章　隐含二叉树

本章首先对构建隐含二叉树所需要的 Arrow-Debreu 价格进行简要介绍。在构建隐含二叉树的过程中，将隐含二叉树以中心节点为界，分为上半部分树和下半部分树两个部分。因此接下来首先分别讨论当枝干层节点数为奇数和偶数时中心节点的股票价格公式，然后分别给出上半部分树和下半部分树的计算公式，最后讨论当得到的股票价格不满足无套利条件时合适的处理方法。

3.1　Arrow-Debreu 价格

假设起始时刻 $t = 0$，构造二叉树时每一步的步长为 Δt，则第 n 个节点所处的时刻为 $t_n = n\Delta t$. 用 $S_{n,i}$ 表示标的资产在第 n 个节点处于状态 i 时的价格，用 (n, i) 表示标的资产所处的位置，用 $p_{n,i}$ 表示标的资产从 (n, i) 转移到 $(n+1, i+1)$ 的转移概率，用 r 表示无风险收益率。

则从第 n 步到第 $n+1$ 步的二叉树结构如下：

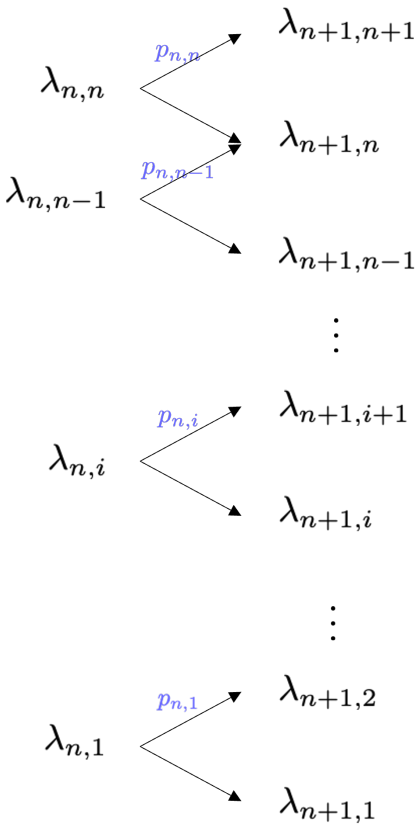


图 3.1. 第 n 步到第 $n+1$ 步的二叉树结构示意图

假设存在一个证券，该证券的支付规则为：若标的资产处于状态 i，则支付 1 元；若标的资产处于其他状态，则支付 0 元。用 $\lambda_{n,i}$ 表示该证券在时刻 n 的价格，

$$\lambda_{0,0} = 1,$$

$$\lambda_{n+1,i} = \begin{cases} e^{-r\Delta t}\left(p_{n,n}\lambda_{n,n}\right), & i = n+1 \\ e^{-r\Delta t}\left(p_{n,i-1}\lambda_{n,i-1} + (1-p_n,i)\,\lambda_{n,i}\right), & 1 \le i \le n \\ e^{-r\Delta t}\left(1-p_{n,0}\right)\lambda_{n,0}, & i = 0 \end{cases}$$

则 $\lambda_{n,i}$ 就被称为 Arrow-Debreu 价格。

以一个步长为 Δt 的两步二叉树为例。

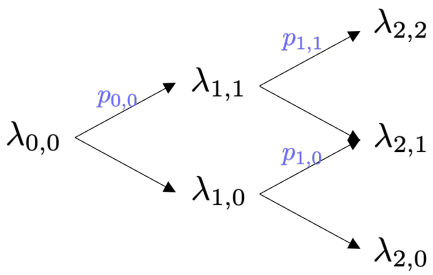


图 3.2. Arrow-Debreu 价格的两步二叉树示例

则可以计算得到各个节点上的 Arrow-Debreu 价格：

$$
\begin{aligned}
\lambda_{0,0} &= 1, \\
\lambda_{1,1} &= e^{-r\Delta t} p_{0,0} \lambda_{0,0}, \\
\lambda_{1,0} &= e^{-r\Delta t} (1 - p_{0,0}) \lambda_{0,0}, \\
\lambda_{2,2} &= e^{-r\Delta t} p_{1,1} \lambda_{1,1}, \\
\lambda_{2,1} &= e^{-r\Delta t} \left[p_{1,0} \lambda_{1,0} + (1 - p_{1,1}) \lambda_{1,1} \right], \\
\lambda_{2,0} &= e^{-r\Delta t} (1 - p_{1,0}) \lambda_{1,0}.
\end{aligned}
$$

3.2 中心节点

假设起始时刻 $t = 0$，对应树的第一层，也称为根节点，对应的股票价格为 $s_{0,0}$。构造二叉树时每一步的步长为 Δt，则第 n 层枝干所处的时刻为 $t_n = (n-1)\Delta t$. 假设已经构造了前 n 步二叉树节点。

3.2.1 当 n 为偶数时

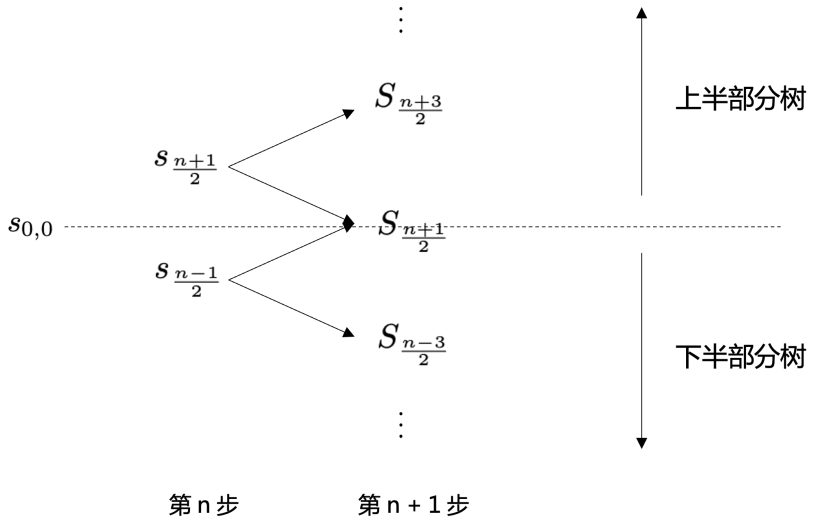


图 3.3. n 为偶数时隐含二叉树中心节点情况

此时，中心节点 $S_{\frac{n+1}{2}}$ 的取值为：

$$S_{\frac{n+1}{2}} = s_{0,0}.$$

3.2.2 当 n 为奇数时

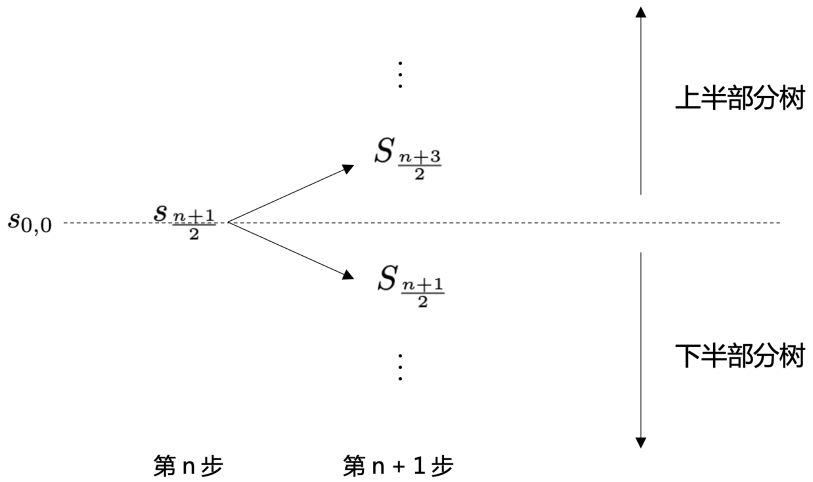


图 3.4. n 为奇数时隐含二叉树中心节点情况

此时 $s_{\frac{n+1}{2}} = s_{0,0}$，而第 $n+1$ 步的中心需要在 CRR 假设下进行计算，具体计算方法在后续介绍。

3.3 其他节点

构建隐含二叉树时，需要根据中心节点的值对上下两部分树的节点值进行递推，当 n 为偶数时，由上面的讨论可以知道第 $n+1$ 层中心点的值与初值相同，讨论起来比较直观，因此首先对 n 为偶数时的情况进行讨论，再对 n 为奇数时的情况进行讨论。

3.3.1 当 n 为偶数时

同样假设起始时刻 $t = 0$，构造二叉树时每一步的步长为 Δt，则第 n 个节点所处的时刻为 $t_n = n\Delta t$.

假设已经构造了前 n 步二叉树节点，n 为奇数，用 $s_i(1 \le i \le n)$ 表示已经通过计算得到的第 n 步处于状态 i 的标的资产的价格，用 $S_i(1 \le i \le n+1)$ 表示未知的第 $n+1$ 步处于状态 i 的标的资产的价格，用 $\lambda_i(1 \le i \le n)$ 表示已经通过计算得到的第 n 步处于状态 i 的 Arrow-Debreu 价格，用 $F_i = e^{r\Delta t}s_i$ 表示标的资产 s_i 在 t_{n+1} 时刻的远期合约价格，用 $p_i(1 \le i \le n)$ 表示未知的从节点 (n,i) 转移到节点 $(n+1,i+1)$ 的转移概率。

表 3.1. 符号及其含义

符号	含义
r	已知的无风险利率
$s_{0,0}$	已知的起初时刻标的资产价格
$s_i,\quad 1 \le i \le n$	已知的处于节点 (n,i) 的标的资产价格
$F_i,\quad 1 \le i \le n$	已知的处于节点 (n,i) 的远期合约价格
$\lambda_i,\quad 1 \le i \le n$	已知的处于节点 (n,i) 的 Arrow-Debreu 价格
$S_i,\quad 1 \le i \le n+1$	未知的处于节点 $(n+1,i)$ 的标的资产价格
$p_i,\quad 1 \le i \le n$	未知的从节点 (n,i) 转移到节点 $(n+1,i+1)$ 的转移概率

当 n 为奇数时，树的第 $n+1$ 步的中心与标的资产起初的价格应该在同一水平线上。第 $n+1$ 步的树上有 $2n+1$ 个未知数：分别是 n 个转移概率和 $n+1$ 个标的资产的价格状态。需要通过解方程组来得到这 $2n+1$ 个量的值。

用 $C(S,t_0,s_i,t_{n+1})$ 和 $P(S,t_0,s_i,t_{n+1})$ 分别表示起始日期为当前时刻 t_0，标的资产为 S，

行权价格为 $s_i(1 \le i \le n)$，到期日为 t_{n+1} 的欧式看涨期权和欧式看跌期权的价格。

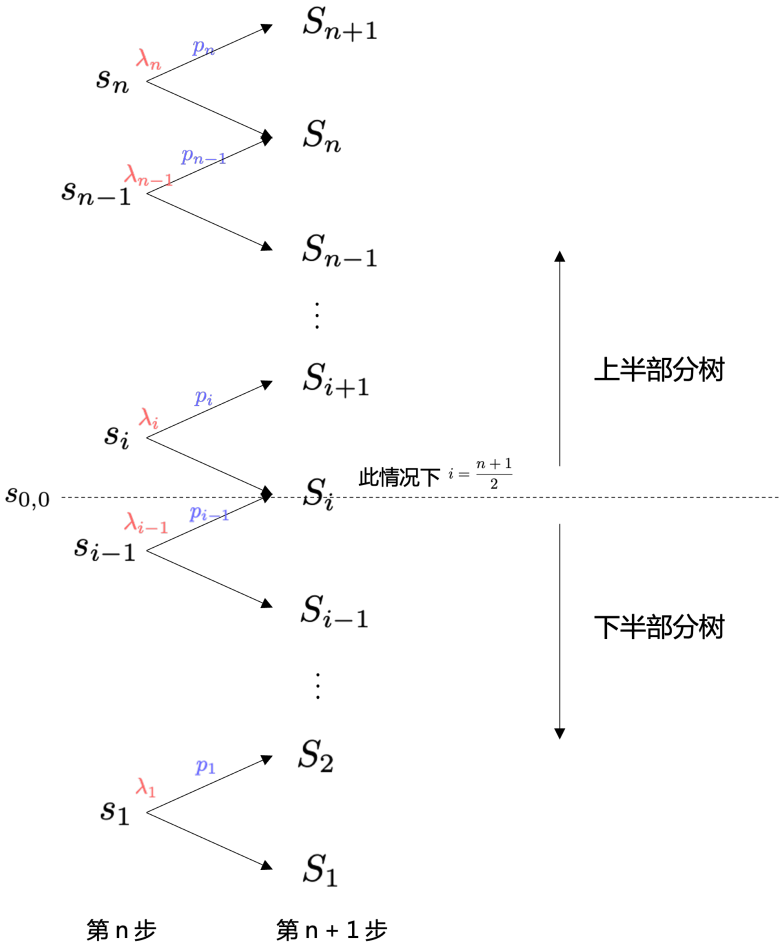


图 3.5. n 为偶数时的隐含二叉树结构

1. **上半部分树**

对于上半部分树，可列方程组

$$\begin{cases} F_i = p_i S_{i+1} + (1 - p_i) S_i \\ C(S, t_0, s_i, t_{n+1}) = e^{-r\Delta t} \sum_{j=1}^{n} \left[\lambda_j p_j + \lambda_{j+1}(1 - p_{j+1}) \right] \max(S_{j+1} - s_i, 0). \\ \lambda_{n+1} = p_{n+1} = 0 \end{cases}$$

得到解为

$$\begin{cases} p_i = \dfrac{F_i - S_i}{S_{i+1} - S_i} \\ S_{i+1} = \dfrac{S_i\left[e^{r\Delta t}C(S,t_0,s_i,t_{n+1}) - \Sigma\right] - \lambda_i s_i(F_i - S_i)}{\left[e^{r\Delta t}C(S,t_0,s_i,t_{n+1}) - \Sigma\right] - \lambda_i(F_i - S_i)}. \end{cases} \tag{3.1}$$

其中，

$$\Sigma := \sum_{j=i+1}^{n} \lambda_j(F_j - s_i).$$

证明. 由 $F_i = p_i S_{i+1} + (1-p_i)S_i$ 可得

$$p_i = \frac{F_i - S_i}{S_{i+1} - S_i}.$$

化简方程 $C(S,t_0,s_i,t_{n+1}) = e^{-r\Delta t}\sum_{j=1}^{n}\left[\lambda_j p_j + \lambda_{j+1}(1-p_{j+1})\right]\max(S_{j+1} - s_i, 0)$,

$$\begin{aligned} e^{r\Delta t}C(S,t_0,s_i,t_{n+1}) = &\sum_{j=i}^{n}\left[\lambda_j p_j + \lambda_{j+1}(1-p_{j+1})\right]\max(S_{j+1} - s_i, 0) \\ =&\left[\lambda_i p_i + \lambda_{i+1}(1-p_{i+1})\right](S_{i+1} - s_i) \\ &+ \left[\lambda_{i+1}p_{i+1} + \lambda_{i+2}(1-p_{i+2})\right](S_{i+2} - s_i) \\ &+ \cdots \\ &+ \lambda_n p_n(S_{n+1} - s_i) \\ =&\lambda_i p_i(S_{i+1} - s_i) \\ &+ \lambda_{i+1}\left\{\left[(1-p_{i+1})S_{i+1} + p_{i+1}S_{i+2}\right] - s_i\right\} \\ &+ \cdots \\ &+ \lambda_n\left\{\left[(1-p_n)S_n + p_n S_{n+1}\right] - s_i\right\} \\ =&\lambda_i p_i(S_{i+1} - s_i) \\ &+ \lambda_{i+1}\left[F_{i+1} - s_i\right] \\ &+ \cdots \\ &+ \lambda_n\left[F_n - s_i\right] \\ =&\lambda_i p_i(S_{i+1} - s_i) + \Sigma. \end{aligned}$$

则

$$\begin{aligned} S_{i+1} - s_i =&\frac{e^{r\Delta t}C(S,t_0,s_i,t_{n+1}) - \Sigma}{\lambda_i p_i} \\ =&\frac{e^{r\Delta t}C(S,t_0,s_i,t_{n+1}) - \Sigma}{\lambda_i(F_i - S_i)}(S_{i+1} - S_i). \end{aligned}$$

进一步化简得

$$\frac{e^{r\Delta t}C(S,t_0,s_i,t_{n+1})-\Sigma-\lambda_i(F_i-S_i)}{\lambda_i(F_i-S_i)}S_{i+1}=\frac{S_i\left[e^{r\Delta t}C(S,t_0,s_i,t_{n+1})-\Sigma\right]-\lambda_i s_i(F_i-S_i)}{\lambda_i(F_i-S_i)}.$$

因此，

$$S_{i+1}=\frac{S_i\left[e^{r\Delta t}C(S,t_0,s_i,t_{n+1})-\Sigma\right]-\lambda_i s_i(F_i-S_i)}{\left[e^{r\Delta t}C(S,t_0,s_i,t_{n+1})-\Sigma\right]-\lambda_i(F_i-S_i)}.$$

证毕。

2. **下半部分树**

对于下半部分树，可列方程组

$$\begin{cases}F_i=p_iS_{i+1}+(1-p_i)S_i\\P(S,t_0,s_i,t_{n+1})=e^{-r\Delta t}\sum_{j=0}^{n}\left[\lambda_jp_j+\lambda_{j+1}(1-p_{j+1})\right]\max(s_i-S_{j+1},0).\\\lambda_0=p_0=0\end{cases}$$

得到解为：

$$\begin{cases}p_i=\dfrac{F_i-S_i}{S_{i+1}-S_i}\\S_i=\dfrac{S_{i+1}\left[e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1\right]+\lambda_i s_i(F_i-S_{i+1})}{\left[e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1\right]+\lambda_i(F_i-S_{i+1})}.\end{cases}\tag{3.2}$$

其中，

$$\Sigma_1=\sum_{j=1}^{i-1}\lambda_j(s_i-F_j).$$

证明. 由 $F_i=p_iS_{i+1}+(1-p_i)S_i$ 可得

$$p_i=\frac{F_i-S_i}{S_{i+1}-S_i}.$$

则

$$1-p_i=\frac{S_{i+1}-F_i}{S_{i+1}-S_i}.$$

$$
\begin{aligned}
e^{r\Delta t}P(S,t_0,s_i,t_{n+1}) &= \sum_{j=0}^{n}\left[\lambda_j p_j + \lambda_{j+1}(1-p_{j+1})\right]\max(s_i - S_{j+1},0)\\
&=\lambda_1(1-p_1)(s_i-S_1)\\
&\quad+\left[\lambda_1 p_1+\lambda_2(1-p2)\right](s_i-S_2)\\
&\quad+\left[\lambda_2 p_2+\lambda_3(1-p3)\right](s_i-S_3)\\
&\quad+\cdots\\
&\quad+\left[\lambda_{i-1}p_{i-1}+\lambda_i(1-p_i)\right](s_i-S_i)\\
&=\lambda_1(s_i-F_1)+\lambda_2(s_i-F_2)+\cdots\\
&\quad+\lambda_{i-1}(s_i-F_{i-1})+\lambda_i(1-p_i)(s_i-S_i)\\
&=\lambda_i(1-p_i)(s_i-S_i)+\sum_{j=1}^{i-1}\lambda_j(s_i-F_j)\\
&=\lambda_i(1-p_i)(s_i-S_i)+\Sigma_1.
\end{aligned}
$$

则，

$$
\begin{aligned}
s_i - S_i &= \frac{e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1}{\lambda_i(1-p_i)}\\
&= \frac{e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1}{\lambda_i(S_{i+1}-F_i)}(S_{i+1}-S_i).
\end{aligned}
$$

进一步化简得

$$
\frac{e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1-\lambda_i(S_{i+1}-F_i)}{\lambda_i(S_{i+1}-F_i)}S_i = \frac{S_{i+1}\left[e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1\right]-\lambda_i s_i(S_{i+1}-F_i)}{\lambda_i(S_{i+1}-F_i)}.
$$

因此，

$$
S_i = \frac{S_{i+1}\left[e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1\right]+\lambda_i s_i(F_i-S_{i+1})}{\left[e^{r\Delta t}P(S,t_0,s_i,t_{n+1})-\Sigma_1\right]+\lambda_i(F_i-S_{i+1})}.
$$

证毕。

因此可以通过式 (3.1) 和 (3.2) 计算每个节点的 S_j 和转移概率 p_j.

3.3.2 当 n 为奇数时

当第 n 层枝干的节点数为奇数时，其中心为 $s_{\frac{n+1}{2}} = s_{0,0}$.

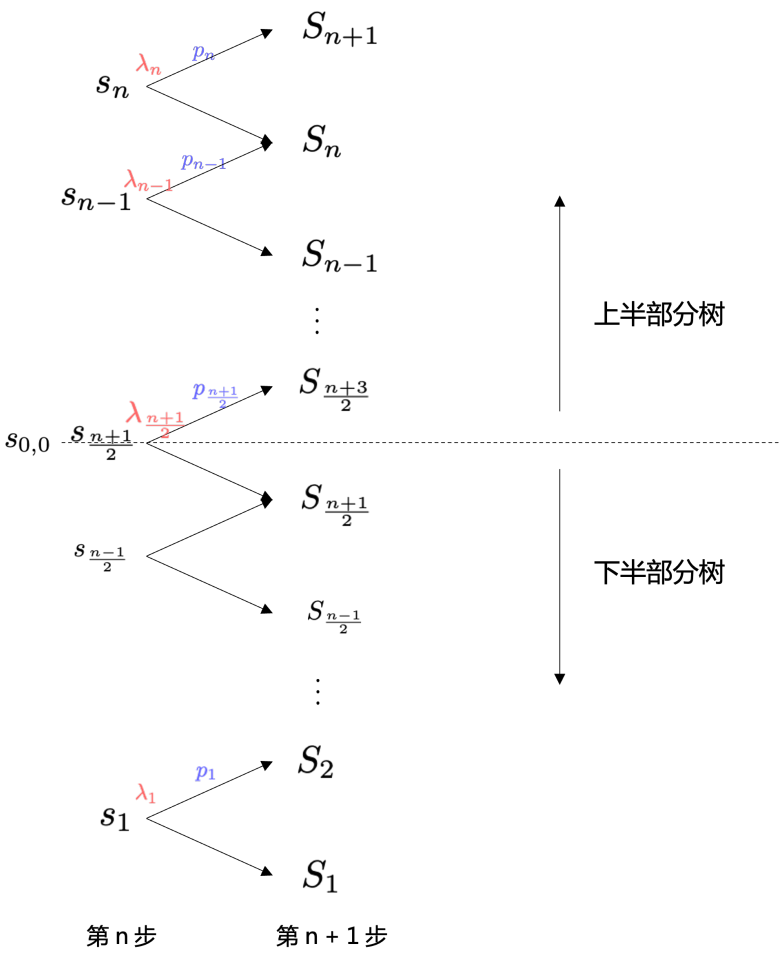


图 3.6. n 为奇数时的隐含二叉树结构

引入 CRR 二叉树的假设，即假设标的资产价格 S 在间隔 Δt 之后向上转移到 $u \times S$，向下转移到 $d \times S$，其中 u, d 满足的条件为

$$\begin{cases} u = \dfrac{1}{d}, \\ u > d. \end{cases}$$

利用 CRR 二叉树的假设来确定 $S_{\frac{n+1}{2}}$ 和 $S_{\frac{n+3}{2}}$，即

$$S_{\frac{n+1}{2}} \times S_{\frac{n+3}{2}} = s_{0,0}^2.$$

可以得到

$$\begin{cases} p_i = \dfrac{F_i - S_i}{S_{i+1} - S_i} \\ S_{\frac{n+3}{2}} = \dfrac{s_{0,0}\left[e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \lambda_{\frac{n+1}{2}}s_{0,0} - \Sigma\right]}{\lambda_{\frac{n+1}{2}}F_{\frac{n+1}{2}} - e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \Sigma} \\ S_{\frac{n+1}{2}} = \dfrac{s_{0,0}\left(\lambda_{\frac{n+1}{2}}F_{\frac{n+1}{2}} - e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \Sigma\right)}{e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \lambda_{\frac{n+1}{2}}s_{0,0} - \Sigma}. \end{cases} \tag{3.3}$$

其中，

$$\Sigma := \sum_{j=\frac{n+3}{2}}^{n} \lambda_j(F_j - s_i).$$

证明. 由式 (3.1) 可得

$$S_{i+1}\left[e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) - \Sigma\right] - \lambda_i S_{i+1}(F_i - S_i) = S_i\left[e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) - \Sigma\right] - \lambda_i s_i(F_i - S_i).$$

其中，$i = \frac{n+1}{2}$，$i+1 = \frac{n+3}{2}$ 又由于

$$e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) = \lambda_i p_i(S_{i+1} - s_{0,0}) + \Sigma,$$

$$F_i = p_i S_{i+1} + (1-p_i)S_i,$$

可以得到

$$\begin{aligned} S_{i+1}\left[e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) - \Sigma\right] - \lambda_i S_{i+1}F_i &= -\lambda_i s_{0,0}^2 + S_i\left(\lambda_i p_i(S_{i+1} - s_{0,0})\right) - \lambda_i s_i(F_i - S_i) \\ &= -\lambda_i s_{0,0}^2 \\ &\quad + \lambda_i p_i s_{0,0}^2 - \lambda_i p_i S_i s_{0,0} \\ &\quad - \lambda_i p_i S_{i+1} s_{0,0} + \lambda_i p_i S_i s_{0,0} \\ &= -\left[\lambda_i s_{0,0}^2 + \lambda_i p_i s_{0,0}(S_{i+1} - s_{0,0})\right] \\ &= -\left[\lambda_i s_{0,0}^2 + s_{0,0}\left(e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) - \Sigma\right)\right]. \end{aligned}$$

将 $i = \frac{n+1}{2}$，$i+1 = \frac{n+3}{2}$ 代入，整理可得

$$S_{\frac{n+3}{2}} = \frac{s_{0,0}\left[e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \lambda_{\frac{n+1}{2}}s_{0,0} - \Sigma\right]}{\lambda_{\frac{n+1}{2}}F_{\frac{n+1}{2}} - e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \Sigma}.$$

最后，由 $S_{\frac{n+3}{2}} \times S_{\frac{n+1}{2}} = s_{0,0}^2$ 可得

$$S_{\frac{n+1}{2}} = \frac{s_{0,0}\left(\lambda_{\frac{n+1}{2}}F_{\frac{n+1}{2}} - e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \Sigma\right)}{e^{r\Delta t}C(S,t_0,s_{0,0},t_{n+1}) + \lambda_{\frac{n+1}{2}}s_{0,0} - \Sigma}.$$

证毕。

在得到树中心的两个资产价格的初始化价格后，可分别通过式 (3.1) 和式 (3.2) 求上半部分树和下半部分树其他节点上的标的资产价格状态和转移概率。

3.4 无套利条件

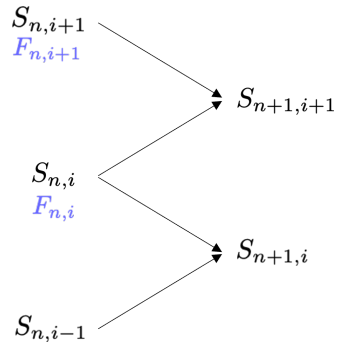


图 3.7. 无套利条件示意图

由无套利假设可得每个节点上的转移概率都必须在 $[0,1]$ 内。若从节点 (n,i) 到节点 $(n+1,i+1)$ 转移概率大于 1，则上半部分树的节点 $(n+1,i+1)$ 上的价格 $S_{n+1,i+1} < F_{n,i}$. 若从节点 (n,i) 到节点 $(n+1,i)$ 转移概率小于 0，则下半部分树的节点 $(n+1,i)$ 上的价格 $S_{n+1,i} > F_{n,i}$. 这两种情况都存在无风险套利的机会。因此必须要求计算得到的股票价格落在对应的两个远期价格之间，即

$$F_{n,i} < S_{n+1,i+1} < F_{n,i+1}.$$

若根据隐含波动率计算出来的股票价格不满足无套利条件，就要推翻通过隐含波动率得到的期权价格所计算得到的节点的股票价格，并将股票价格设定在上述不等式区间内，本报告采取的方法是使 $S_{n+1,i+1}$ 与 $S_{n+1,i}$ 之间的距离与 $S_{n,i}$ 与 $S_{n,i-1}$ 之间的距离保持一致，即将 $S_{n+1,i+1}$ 设置为

$$S_{n+1,i+1} = S_{n+1,i} + \left(S_{n,i} - S_{n,i-1}\right).$$

第四章　实证分析

本报告首先根据已有数据计算 2019 年 12 月 2 日生效、有效期限为一个月的上证 50ETF 欧式看涨和看跌期权的隐含波动率，得到两类期权在 2019 年 12 月 2 日的期限为一个月的波动率微笑，由于数据的限制，假设不同期限结构的波动率微笑形状相同，然后通过线性插值的方法对未知的行权价格所对应的隐含波动率进行插值处理。

接下来通过式 (3.1)、式 (3.2) 和式 (3.3) 计算隐含二叉树中各个节点的股价。对每一层枝干，首先计算中心节点的股票价格，再自下而上得到上半部分树节点的股票价格，最后自上而下得到下半部分树节点的股票价格。

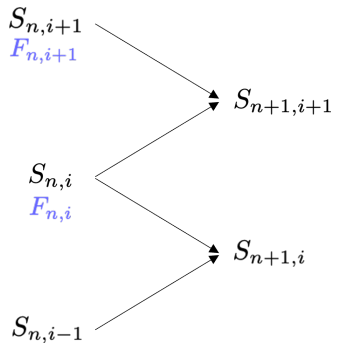


图 4.1. 无套利条件示意图

在得到每一个节点的股票价格后，都需要检查其是否满足无套利条件，如果不满足无套利条件（即上图中 $S_{n+1,i+1}$ 不满足 $F_{n,i} < S_{n+1,i+1} < F_{n,i+1}$ 这个条件），则将 $S_{n+1,i+1}$ 设置为

$$S_{n+1,i+1} = S_{n+1,i} + \left(S_{n,i} - S_{n,i-1}\right),$$

再继续递推计算剩余节点的股票价格，最终得到一个隐含二叉树。

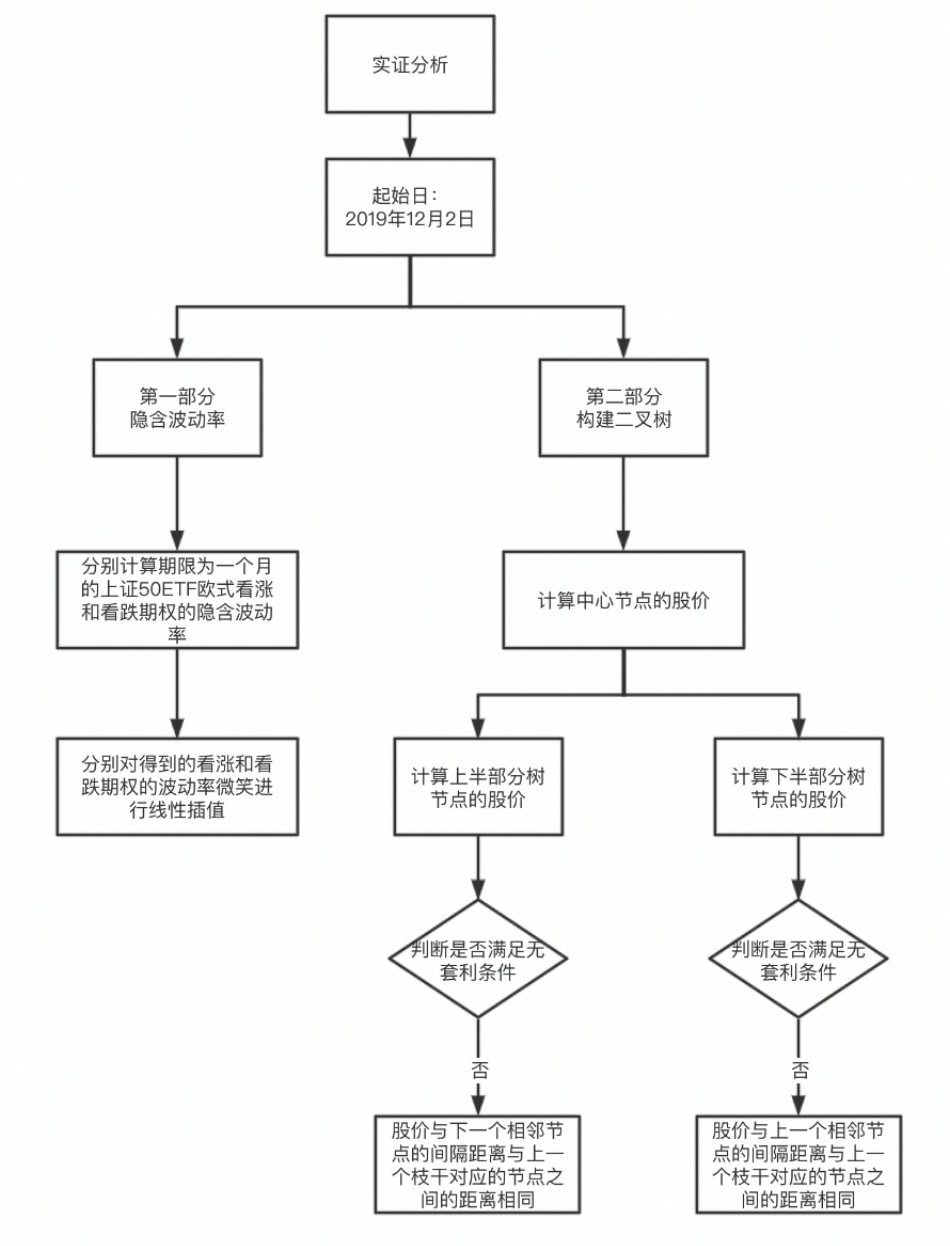


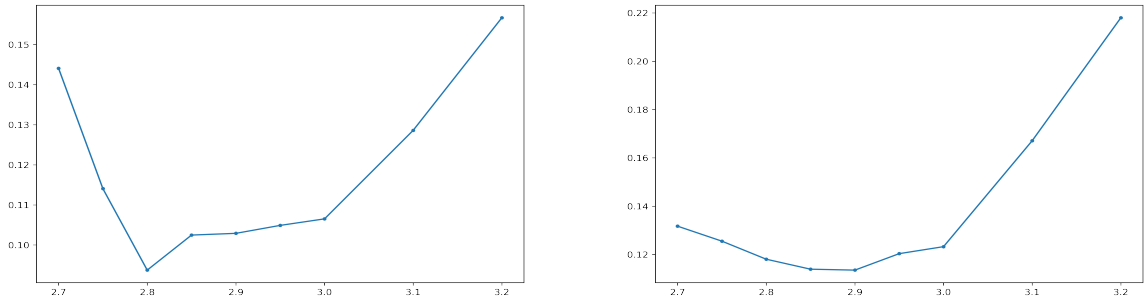
图 4.2. 实证分析流程图

4.1 求解隐含波动率

由于上证 50EFT 期权的合约到期月份为当月、下月及随后两个季月，行权日期为到期月的第四个星期三，因此理论上根据于 2019 年 12 月 2 日生效的期权的数据可以求得 6 个月的 9 种不同行权价格对应的隐含波动率。由于要构建 24 个月的隐含二叉树，根据已有的数据无法得到隐含波动率的期限结构，为简便运算，假设不同期限对应的波动率

微笑相同。

2019 年 12 月 2 日上证 50ETF 的收盘价为 $s = 2.899$, 将 s 以及设置的波动率初始值带入欧式期权的 B-S 公式得到期权的理论价格，设置收敛阈值为 $\epsilon = 1 \times 10^{-5}$，及当期权理论价格与市场价格之间的差距小于阈值时，认为算法收敛，此时对应的波动率即为求解得到的隐含波动率，运用牛顿迭代法求 2019 年 12 月 2 日期限为一个月的看涨和看跌期权的隐含波动率。



(a) 上证 50ETF 欧式看涨期权波动率微笑　(b) 上证 50ETF 欧式看跌期权波动率微笑

图 4.3. 上证 50ETF 欧式期权波动率微笑

而在计算隐含二叉树中各节点的股票价格时，需要用到一些不同上图中已知的 9 个行权价格所对应的隐含波动率，因此需要估计其他行权价格对应的隐含波动率。为简便运算，对行权价格保留三位小数，在 2.7～3.2 之间进行线性插值，在小于 2.7 和大于 3.2 的部分，假设行权价格每变化 0.001，隐含波动率相应变化 0.00025. 这样得到隐含波动率之后，接下来构建隐含二叉树。

4.2 隐含二叉树的构建

设当前时刻为 $t = 0$，用 n 表示树的枝干层数，r 表示年化无风险利率，T 表示一个月的时间间隔。每一层枝干的节点编号从上至下按照从大到小的顺序进行排列，即第 n 层枝干的节点编号从上至下依次从 n 至 1 递减。

由于从第一层树生长到第二层树，以及从第二层树生长到第三层树，这两种情况包含了奇数个节点和偶数个节点的情况，因此接下来以这两种情况为例具体介绍计算过程。

4.2.1 构建过程

1. 当 $n = 1$ 时

第一层枝干也被称为根节点，在这个点上已知上证 50ETF 的收盘价为 $s_0 = 2.899$，以及 Arrow-Debreu 价格为 1，同时可以通过公式

$$F_1 = e^{rT} s_0$$

得到远期合约的价格。

表 4.1. 根节点已知数据

节点编号	股价	Arrow-Debreu 价格	远期合约价格
1	2.899	1	2.9050

2. 当 $n = 2$ 时

第二层枝干有两个节点，需要根据式 (3.3) 来计算这两个节点的股价。

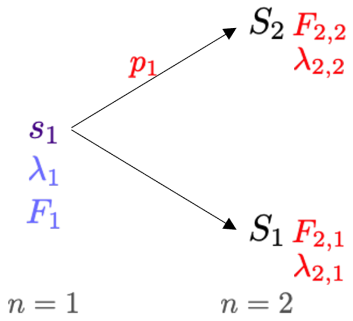


图 4.4. 第 1 层至第 2 层树的示意图

首先从波动率微笑中得到行权价格为 $s_1 = 2.899$ 的欧式看涨期权的隐含波动率，并计算对应的看涨期权价格。

表 4.2. 行权价格为 $s_1 = 2.899$ 时的隐含波动率及期权价格

看涨期权隐含波动率	看涨期权价格 $C(s_0, 0, s_1, T)$
0.102892	0.037416

此时的已知量有：

表 4.3. $n = 2$ 时的已知量

含义	符号	值
根节点股价	s_1	2.899
远期合约价格	F_1	2.9050
Arrow-Debreu 价格	λ_1	1

于是根据公式：

$$\begin{cases} S_2 = \dfrac{s_1\left[e^{rT}C(s_0,0,s_1,T) + \lambda_1 s_1\right]}{\lambda_1 F_1 - e^{rT}C(s_0,0,s_1,T)} \\ S_1 = \dfrac{s_1^2}{S_2}. \end{cases}$$

可以得到

$$\begin{cases} S_2 = 2.969 \\ S_1 = 2.831 \end{cases}.$$

在得到 S_2 和 S_1 后，通过公式计算出这一期远期合约的价格：

$$\begin{cases} F_{2,2} = e^{rT}S_2 = 2.9752 \\ F_{2,1} = e^{rT}S_1 = 2.8369 \end{cases},$$

以及风险中性概率：

$$p_1 = \frac{F_1 - S_1}{S_2 - S_1},$$

以及这一期的 Arrow-Debreu 价格：

$$\begin{cases} \lambda_{2,2} = e^{-rT}p_1\lambda_1 = 0.5354 \\ \lambda_{2,1} = e^{-rT}(1-p_1)\lambda_1 = 0.4624 \end{cases}.$$

于是，$S_1, S_2, F_{2,1}, F_{2,2}, \lambda_{2,1}, \lambda_{2,2}, p_1$ 成为第二层树到第三层树的已知量。

3. 当 $n = 3$ 时

用 s_1, s_2 表示上一层得到的股价 S_1, S_2，而用 S_1, S_2, S_3 表示第三层要求的未知股价，上一层得到的风险中性概率 p_1 在这一层的计算中不再需要使用，因此这一层用 p_2, p_1 表示未知的风险中性概率，其余已知量的符号保持不变。

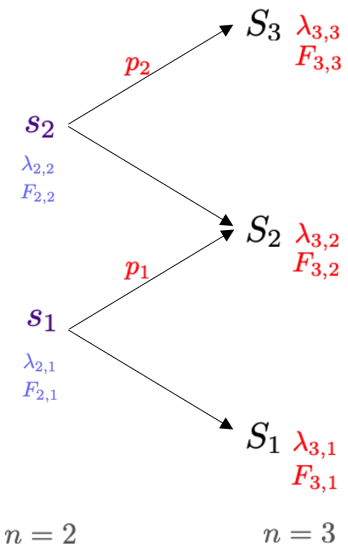


图 4.5. 第 2 层至第 3 层树的示意图

这一层树的中心节点 S_2 的值应该与期初的股价相同，因此

$$S_2 = s_0 = 2.899.$$

首先考虑上半部分树，从波动率微笑中得到行权价格为 $s_2 = 2.969$ 的欧式看涨期权的隐含波动率，并计算对应的看涨期权价格。

表 4.4. 行权价格为 $s_2 = 2.969$ 时的隐含波动率及期权价格

看涨期权隐含波动率	看涨期权价格 $C(s_0, 0, s_2, 2T)$
0.105505	0.026646

于是通过计算可以得到 S_3 的值：

$$S_3 = \frac{S_2\left[e^{rT}C(s_0, 0, s_2, 2T)\right] - \lambda_{2,2}s_2\left(F_{2,2} - S_2\right)}{e^{rT}C(s_0, 0, s_2, 2T) - \lambda_{2,2}\left(F_{2,2} - S_2\right)} = 3.102\ .$$

再考虑下半部分树，从波动率微笑中得到行权价格为 $s_1 = 2.831$ 的欧式看跌期权的隐含波动率，并计算对应的看跌期权价格。

表 4.5. 行权价格为 $s_1 = 2.831$ 时的隐含波动率及期权价格

看跌期权隐含波动率	看跌期权价格 $P(s_0, 0, s_1, 2T)$
0.115473	0.023030

于是通过计算可以得到 S_1 的值：

$$S_1 = \frac{S_2\left[e^{rT}P(s_0,0,s_1,2T)\right] + \lambda_{2,1}s_1\left(F_{2,1}-S_2\right)}{e^{rT}P(s_0,0,s_1,2T) + \lambda_{2,1}\left(F_{2,1}-S_2\right)} = 2.553\ .$$

在得到 S_3、S_2 和 S_1 后，通过公式计算出这一期远期合约的价格：

$$\begin{cases} F_{3,3} = e^{rT}S_3 = 3.1085 \\ F_{3,2} = e^{rT}S_2 = 2.9050\ , \\ F_{3,1} = e^{rT}S_1 = 2.5583 \end{cases}$$

以及风险中性概率：

$$\begin{cases} p_2 = \dfrac{F_{2,2}-S_2}{S_3-S_2} = 0.3753 \\ p_1 = \dfrac{F_{2,1}-S_1}{S_2-S_1} = 0.8205 \end{cases},$$

以及这一期的 Arrow-Debreu 价格：

$$\begin{cases} \lambda_{3,3} = e^{-rT}p_2\lambda_{2,2} = 0.2006 \\ \lambda_{3,2} = e^{-rT}(p_1\lambda_{2,1} + (1-p_2)\lambda_{2,2}) = 0.7125\ . \\ \lambda_{3,1} = e^{-rT}(1-p_1)\lambda_{2,1} = 0.0828 \end{cases}$$

因此，$S_1, S_2, S_3, F_{3,1}, F_{3,2}, F_{3,3}, \lambda_{3,1}, \lambda_{3,2}, \lambda_{3,3}, p_1, p_2$ 成为第三层树到第四层树的已知量。

按照上述步骤依次计算每一层树的节点股价即可，但是在计算出每一个节点股价之后，要对其进行无套利条件的检验。如果不满足无套利条件（即图 (??) 中 $S_{n+1,i+1}$ 不满足 $F_{n,i} < S_{n+1,i+1} < F_{n,i+1}$ 这个条件），则将 $S_{n+1,i+1}$ 设置为

$$S_{n+1,i+1} = S_{n+1,i} + \left(S_{n,i} - S_{n,i-1}\right),$$

再继续递推计算剩余节点的股票价格，最终构建出隐含二叉树。

4.2.2 构建结果

由于篇幅限制，报告中只画出前 12 层树，完整的隐含二叉树的数据将于附件展示。

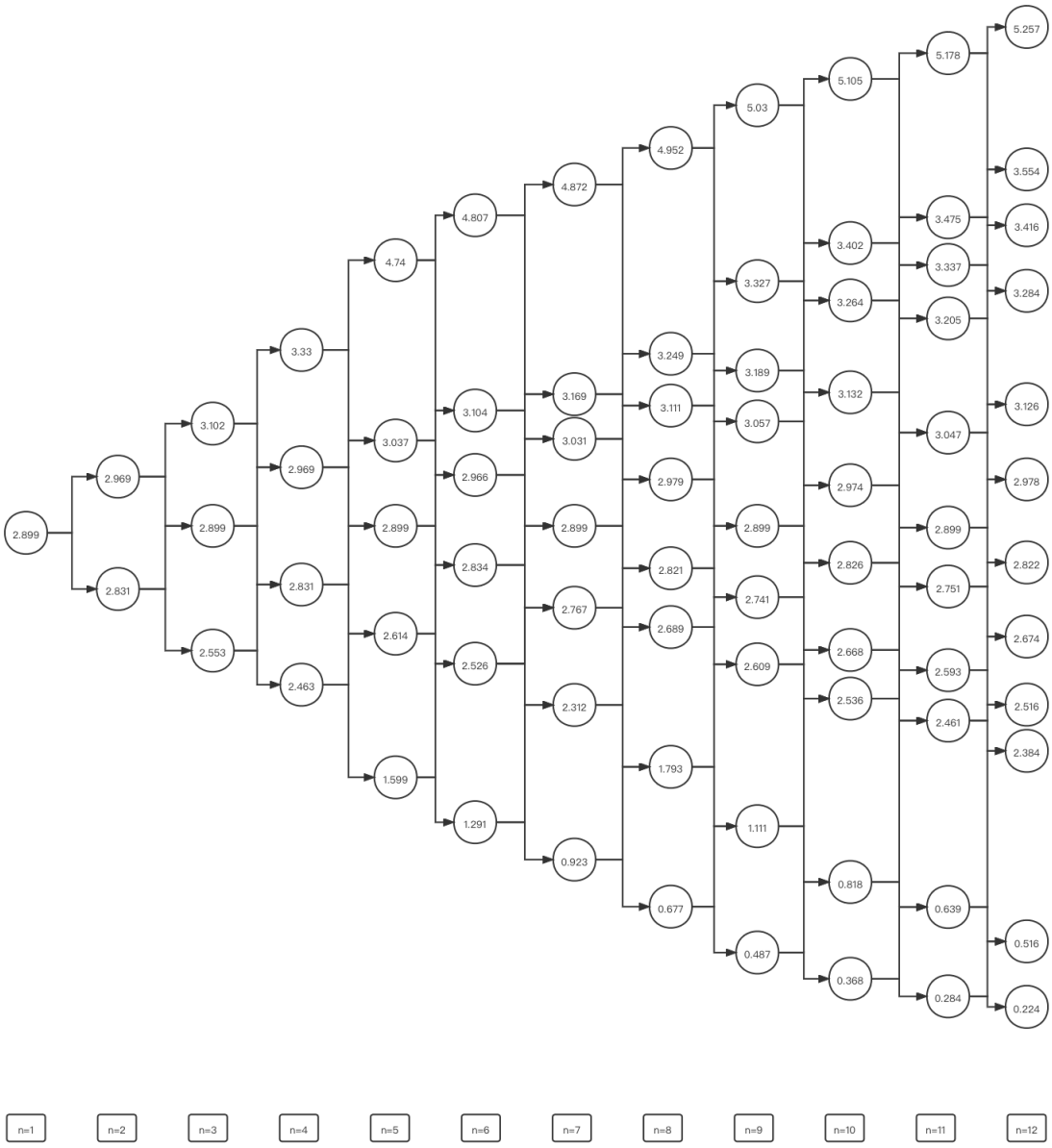


图 4.6. 得到的股票价格二叉树的一部分

可以看到隐含二叉树与一般的二叉树的结构有所不同，前者一般不会出现完全对称的结构，而后者的结构是完全对称的。

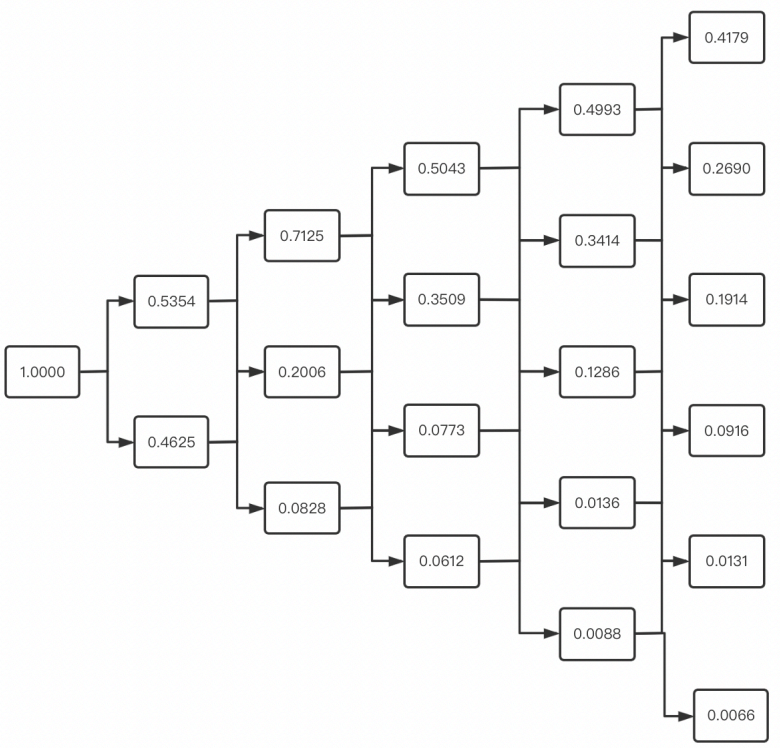


图 4.7. 得到的 Arrow-Debreu 树的一部分

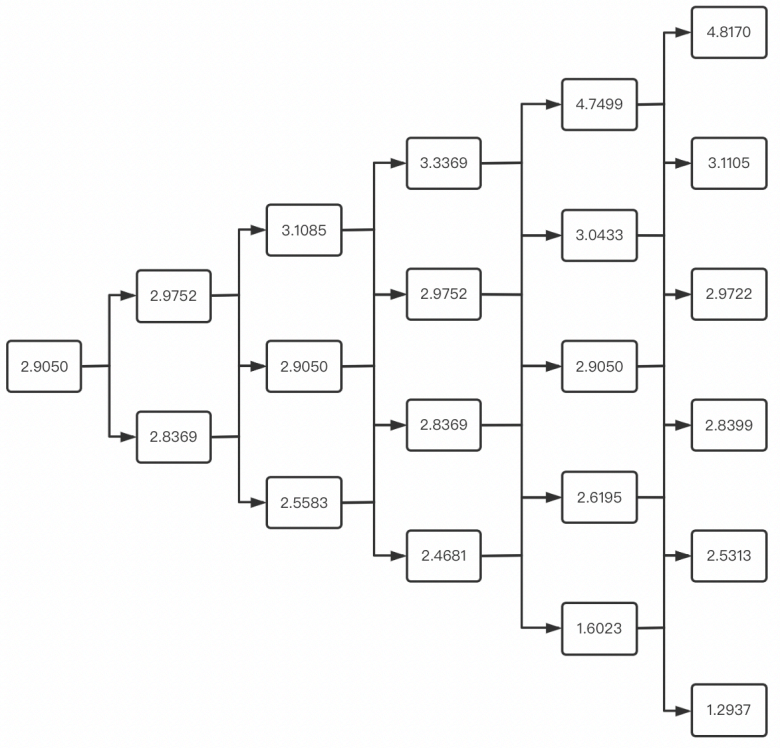


图 4.8. 得到的远期合约价格树的一部分

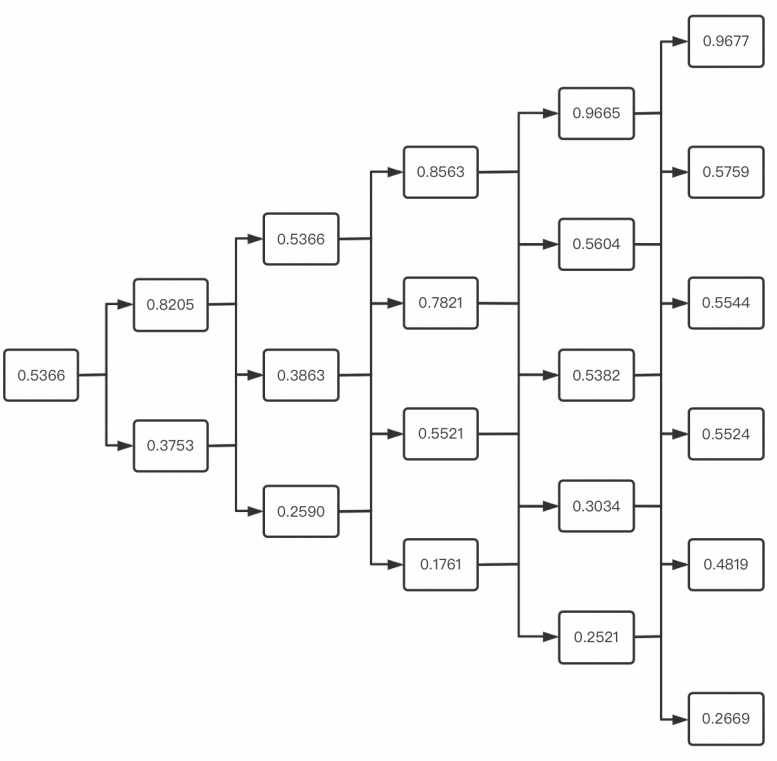


图 4.9. 得到的风险中性概率树的一部分

二叉树中每一步的转移概率是一致的，但是通过得到的数据可以看到，隐含二叉树的风险中性概率并没有这个特点。

4.2.3 隐含二叉树的作用

传统的 Black-Scholes 模型假设波动率是常数，隐含二叉树是对 Black-Scholes 模型的一个拓展，它引入从市场中得到的隐含波动率，使得构建出来的树与市场之间具有一致性。

由于隐含波动率是市场期权价格的一个内含值，而隐含二叉树的实质是根据隐含波动率计算得到的股票未来价格的离散分布，因此得到的股票价格的分布是在风险中性概率测度下的分布。而在得到了股票价格未来的在风险中性概率测度下的分布之后，有以下几种作用：

1) 不仅可以都欧式期权进行定价，还可以对其他类型的期权，如：美式期权、障碍期权、其他路径依赖型期权进行定价，可以应用的范围比较广；

2) 可以与蒙特卡洛模拟方法进行结合，通过随机模拟生成股票价格的变化路径，分别

对每一条路径进行期权支付规则的分析，得到每一条路径的收益现值，对所有路径的现值取平均即可得到期权的价格。

第五章　小结

本报告从理论出发，依次介绍期权定价相关理论、一般二叉树模型和隐含二叉树模型，并在实证分析部分以 2019 年 12 月 2 日为起始日期，以一个月为间隔时间单位，对上证 50ETF 构造了 24 层的隐含二叉树。

本报告首先假设波动率微笑在这两年的结构是一致的，然后利用牛顿迭代法求得上证 50ETF 欧式看涨和看跌期权的隐含波动率，并对不同行权价格对应的隐含波动率进行线性插值。

首先假设隐含波动率的期限结构不变以及用线性插值的方法估计隐含波动率的好处是简便运算，但是隐含波动率的期限结构不变是与市场不符的，这会导致得到的股票价格分布与风险中性测度下的股票价格分布有一定的差异，从而进一步导致计算期权价格时的不准确。因此应该对市场的数据进行隐含波动率曲面进行建模，将波动率的偏度和期限结构同时考虑到模型内，这样才能与市场信息更为相符。

其次直接对波动率微笑进行线性插值是比较死板的方法，并没有考虑在已知的行权价格之间隐含波动率其他的变化方式。

在得到了波动率微笑之后，本报告构建了 24 层的隐含二叉树，以第一层到第三层为例介绍了分别介绍了奇数节点层和偶数节点层中心节点价格的计算，并分别对上半部分树和下半部分树进行构建，给出了各个节点构造的详细过程，最后展示了树的部分结果。

参考文献

[1] MAJMIN L. Local and stochastic volatility models: An investigation into the pricing of exotic equity options[D]. [S.l.]: University of the Witwatersrand, 2005.

[2] DERMAN E, KANI I. The volatility smile and its implied tree[J]. Goldman Sachs Quantitative Strategies Research Notes, 1994, 2: 45–60.

[3] DERMAN E, KANI I, GOLDMAN N C. Implied trinomial trees of the volatility smile[C] // Journal of Derivatives. 1996.

[4] 郭姝辛. 美式期权的正则隐含二叉树定价新法 [D]. [S.l.]: 西南财经大学, 2013.

程序代码

一、计算隐含波动率和期权价格的两个函数

```
from pandas import read_csv
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np


#1. 求隐含波动率的函数
#key：是call或put
#price：期权的市场价格
#S：标的资产当前价格
#sigma：设置的初始波动率
#res：收敛阈值
def ImpliedVolatility(key,price,S,E,T,r,q,sigma,res= 1e-5):
    d1 = (np.log(S/E)+(r-q+(sigma**2)/2)*T)/(sigma*np.sqrt(T))
    d2 = d1-sigma*np.sqrt(T)
    if key == 'call':
    f = S*np.exp(-q*T)*norm.cdf(d1)-E*np.exp(-r*T)*norm.cdf(d2) - price
    elif key == 'put':
    f = E*np.exp(-r*T)*norm.cdf(-d2) - S*np.exp(-q*T)*norm.cdf(-d1) - price
    f_delta = S*np.exp(-q*T)*norm.pdf(d1)*np.sqrt(T)
    iters = 0
    while abs(f)>res:
    iters += 1
    d_sigma = -f/f_delta
    sigma = sigma+d_sigma
    d1 = (np.log(S/E)+(r-q+sigma**2/2)*T)/(sigma*np.sqrt(T))
    d2 = d1-sigma*np.sqrt(T)
    if key == 'call':
    f = S*np.exp(-q*T)*norm.cdf(d1)-E*np.exp(-r*T)*norm.cdf(d2) - price
    elif key == 'put':
    f = E*np.exp(-r*T)*norm.cdf(-d2) - S*np.exp(-q*T)*norm.cdf(-d1) - price
    f_delta = S*np.exp(-q*T)*norm.pdf(d1)*np.sqrt(T)
    return sigma
```

```
35
36  #2. 欧式看涨、看跌期权的B-S公式
37  def EurOption(key,S,K,T,r,sigma):
38      d1 = (np.log(S/K)+(r+sigma**2/2)*T)/(sigma*np.sqrt(T))
39      d2 = d1 - sigma*np.sqrt(T)
40      if key == 'call':
41      price = S * norm.cdf(d1) - K * np.exp(- r*T) * norm.cdf(d2)
42      elif key == 'put':
43      price = K * np.exp(-r*T) * norm.cdf(-d2) - S * norm.cdf(-d1)
44      return price
```

二、计算隐含波动率并进行插值处理

```
1   #2019-12-02的上证50指数的价格
2   Pri0 = 2.899
3   #期限
4   T = 1/12
5   #无风险利率
6   r = 2.5/100
7
8   #2019-12-02期限为一个月的欧式看涨期权的价格
9   bri1 = list(range(10002029,10002038,1))
10  file_path = "/Users/yangyuxin/Desktop/PKU/2021-2022秋季/衍生工具模型/期末大作业/
        Data/上市以来50ETF期权的日度数据/1d/"
11  data1 = {}
12  for i in bri1:
13      path = file_path + str(i) + ".csv"
14      print(path)
15      try:
16          a = read_csv(path)
17      except:
18          continue
19      df = a[['date','strike_price','close']]
20      data1[i] = df
21
22  DF1 = data1[10002029]
23  for i in list(data1.keys())[1:]:
24      DF1 = DF1.append(data1[i],ignore_index=True)
25
```

```
DF1 = DF1.set_index(['date'])
DF3 = DF1.loc['2019-12-02']

#计算隐含波动率
ImpliedVolCall = []
for j in range(len(DF3.index)):
    if j in [0,1]:
        res = 1e-2
    else:
        res = 1e-5
    print(res)
    E = DF3['strike_price'][j]
    price = DF3['close'][j]
    v = ImpliedVolatility('call',price,Pri0,E,T,r,0,0.5,res)
    print('result:',j,E,v)
    ImpliedVolCall.append(v)

#欧式看涨期权波动率微笑
ImpVolCall = pd.DataFrame(ImpliedVolCall,index=DF3.strike_price,columns=['
    ImpliedVolatility'])
plt.figure(figsize = (9,6),dpi=480)
plt.plot(ImpVolCall,marker='.')

#隐含波动率线性插值
strike_call=[]
vol_call=[]
for j in [0.001*i for i in range(int(2.7/0.001))]:
    strike = round(j,3)
    vol = (2.7-j)/0.001*0.00025 + float(ImpVolCall.iloc[0])
    strike_call.append(strike)
    vol_call.append(vol)
for j in range(len(ImpVolCall.index)-1):
    delta = 0.001
    n = int(round((ImpVolCall.index[j+1] - ImpVolCall.index[j])/delta,0))
    for k in range(n+1):
        x = round(delta * (k),3)
        vol = ImpVolCall.iloc[j]+(ImpVolCall.iloc[j+1]-ImpVolCall.iloc[j])/(ImpVolCall.
            index[j+1]-ImpVolCall.index[j])*x
        vol = round(float(vol),6)
        strike = round(ImpVolCall.index[j]+x,3)
        strike_call.append(strike)
        vol_call.append(vol)
```

```
for j in range(10000):
    strike = round(ImpVolCall.index[-1].tolist()+j*0.001,3)
    vol = round(float(ImpVolCall.iloc[-1])+j*0.00025,3)
    strike_call.append(strike)
    vol_call.append(vol)

ImpliedVolCall = pd.DataFrame(vol_call,index=strike_call,columns=['ImpliedVolatility'])

#2019-12-02期限为一个月的欧式看跌期权的价格
bri2 = list(range(10002038,10002047,1))
data2 = {}
for i in bri2:
    path = file_path + str(i) + ".csv"
    print(path)
    try:
        a = read_csv(path)
    except:
        continue
    df = a[['date','strike_price','close']]
    data2[i] = df
DF2 = data2[10002038]
for i in list(data2.keys())[1:]:
    DF2 = DF2.append(data2[i],ignore_index=True)
DF2 = DF2.set_index(['date'])
DF4 = DF2.loc['2019-12-02']

#计算隐含波动率
ImpliedVolPut = []
for j in range(len(DF3.index)):
    res=1e-5
    print(res)
    E = DF4['strike_price'][j]
    price = DF4['close'][j]
    v = ImpliedVolatility('put',price,Pri0,E,T,r,0,0.5,res)
    print('result:',j,E,v)
    ImpliedVolPut.append(v)
#欧式看跌期权波动率微笑
ImpVolPut = pd.DataFrame(ImpliedVolPut,index = DF4.strike_price,columns=['
    ImpliedVolatility'])
plt.figure(figsize = (9,6),dpi=480)
plt.plot(ImpVolPut,marker='.')

```

```
#隐含波动率线性插值
strike_put=[]
vol_put=[]
for j in [0.001*i for i in range(int(2.7/0.001))]:
    strike = round(j,3)
    vol = (2.7-j)/0.001*0.00025 + float(ImpVolPut.iloc[0])
    strike_put.append(strike)
    vol_put.append(vol)
for j in range(len(ImpVolPut)-1):
    delta = 0.001
    n = int(round((ImpVolPut.index[j+1] - ImpVolPut.index[j])/delta,0))
    for k in range(n+1):
        x = round(delta * (k),3)
        vol = ImpVolPut.iloc[j]+(ImpVolPut.iloc[j+1]-ImpVolPut.iloc[j])/(ImpVolPut.
            index[j+1]-ImpVolPut.index[j])*x
        vol = round(float(vol),6)
        strike = round(ImpVolPut.index[j]+x,3)
        strike_put.append(strike)
        vol_put.append(vol)
for j in range(10000):
    strike = round(ImpVolPut.index[-1].tolist()+j*0.001,3)
    vol = round(float(ImpVolPut.iloc[-1])+j*0.00025,3)
    strike_put.append(strike)
    vol_put.append(vol)
ImpliedVolPut = pd.DataFrame(vol_put,index=strike_put,columns=['ImpliedVolatility'])
```

三、构建隐含二叉树

```
#exp{rT}
fac = np.exp(r*T)

#n=1
#上证ETF收盘价
s0 = 2.899
#Arrow-Debreu价格
lam1 = 1
#ForwardPrice
F1 = fac * s0

```

```
#Stock Price 的结果列表
StockPrice = [[s0]]
#Arrow-Debreu价格 的结果列表
ArrowDe = [[lam1]]
#Forward Price 的结果列表
ForPri = [[F1]]
#Probability 的结果列表
Prob = []

print('第',1,'个节点')
print('StockPrice:',StockPrice)
print('Arrow-Debreu:',ArrowDe)
print('ForwardPrice:',ForPri)

for k in range(2,25):
    #已知量:
    lam = ArrowDe[k-2]
    F = ForPri[k-2]
    s = StockPrice[k-2]
    #未知量:
    S = [0 for i in range(k)]
    p = [0 for i in range(k-1)]
    lam_k = [0 for i in range(k)]
    F_k = [0 for i in range(k)]

    if k\%2 == 1:

        #center: S[(k-1)/2]
        center = int((k-1)/2)
        S[center] = s0

        #upper:
        for j in range(center+1,k):
        ##计算Sigma值
        if j == k-1:
            Sigma = 0
        else:
            a = pd.Series(lam) * pd.Series(F) - s[j-2] * pd.Series(lam)
            Sigma = a[j-1:].sum()
        ##调隐含波动率并计算看涨期权的价值
        impvol_call = ImpliedVolCall['ImpliedVolatility'][ImpliedVolCall.index == s[j-1]].
            tolist()[0]
```

```
53          Call = EurOption('call',s0,s[j-1],(k-1)*T,r,impvol_call)
54
55          S[j] = round((S[j-1]*(fac * Call - Sigma)-lam[j-1]*s[j-1]*(F[j-1]-S[j-1]))/(fac * Call -
                Sigma - lam[j-1]*(F[j-1]-S[j-1])),3)
56
57          ##判断是否满足无套利条件;
58          if j!= k-1:
59              if S[j]<F[j] and S[j]>F[j-1]:
60                  continue
61              else:
62                  S[j] = S[j-1]+s[j-1]-s[j-2]
63                  S[j] = round(S[j],3)
64          else:
65              if S[j]>F[j-1]:
66                  continue
67              else:
68                  S[j] = S[j-1]+s[j-1]-s[j-2]
69                  S[j] = round(S[j],3)
70
71          #lower:
72          for j in range(center-1,-1,-1):
73          ##计算Sigma
74              if j == 0:
75                  Sigma1 = 0
76              else:
77                  a = pd.Series(lam) * s[j] - pd.Series(lam) * pd.Series(F)
78                  Sigma1 = a[:j].sum()
79          ##调隐含波动率并计算看跌期权的价值
80          impvol_put = ImpliedVolPut['ImpliedVolatility'][ImpliedVolPut.index == s[j]].
                tolist()[0]
81          Put = EurOption('put',s0,s[j],(k-1)*T,r,impvol_put)
82
83          S[j] = round((S[j+1]*(fac*Put-Sigma1) + lam[j]*s[j]*(F[j]-S[j+1]))/(fac*Put-
                Sigma1+lam[j]*(F[j]-S[j+1])),3)
84
85          ##判断是否满足无套利条件;
86          if j!= 0:
87              if S[j]<F[j] and S[j]>F[j-1]:
88                  continue
89              else:
90                  S[j] = S[j+1]-(s[j+1]-s[j])
91                  S[j] = round(S[j],3)
```

```
92              else:
93                  if S[j]<F[0]:
94                      continue
95                  else:
96                      S[j] = S[j+1]-(s[j+1]-s[j])
97                      S[j] = round(S[j],3)
98
99          elif k%2 ==0:
100
101             #center:
102             center_up = int(k/2)
103             center_down = center_up - 1
104             impvol = ImpliedVolCall['ImpliedVolatility'][ImpliedVolCall.index == s[center_up
                    -1]].tolist()[0]
105             Call0 = EurOption('call',s0,s[center_up-1],(k-1)*T,r,impvol)
106             if k == 2:
107                 Sigma0 = 0
108             else:
109                 a = pd.Series(lam) * pd.Series(F)- s[center_down] * pd.Series(lam)
110                 Sigma0 = a[center_up:].sum()
111             S[center_up] = (s[center_up-1]*(fac*Call0 + lam[center_up-1]*s[center_up-1]-
                    Sigma0))/(lam[center_up-1]*F[center_up-1]-fac*Call0+Sigma0)
112             S[center_up] = round(S[center_up],3)
113             S[center_down] = round(s[center_up-1]**2 / S[center_up],3)
114
115             #upper
116             for j in range(center_up+1,k):
117                 ##计算Sigma值
118                 if j == k-1:
119                     Sigma = 0
120                 else:
121                     a = pd.Series(lam) * pd.Series(F) - s[j-2] * pd.Series(lam)
122                     Sigma = a[j-1:].sum()
123                 ##调隐含波动率并计算看涨期权的价值
124             impvol_call = ImpliedVolCall['ImpliedVolatility'][ImpliedVolCall.index == s[j-1]].
                    tolist()[0]
125             Call = EurOption('call',s0,s[j-1],(k-1)*T,r,impvol_call)
126
127             S[j] = round((S[j-1]*(fac * Call - Sigma)-lam[j-1]*s[j-1]*(F[j-1]-S[j-1]))/(fac * Call -
                    Sigma - lam[j-1]*(F[j-1]-S[j-1])),3)
128
129             ##判断是否满足无套利条件;
```

```
130             if j!= k-1:
131                 if S[j]<F[j] and S[j]>F[j-1]:
132                     continue
133                 else:
134                     S[j] = S[j-1]+s[j-1]-s[j-2]
135                     S[j] = round(S[j],3)
136             else:
137                 if S[j]>F[j-1]:
138                     continue
139                 else:
140                     S[j] = S[j-1]+s[j-1]-s[j-2]
141                     S[j] = round(S[j],3)
142 
143             #lower:
144             for j in range(center_down-1,-1,-1):
145                 ##计算Sigma
146                 if j == 0:
147                     Sigma1 = 0
148                 else:
149                     a = pd.Series(lam) * s[j] - pd.Series(lam) * pd.Series(F)
150                     Sigma1 = a[:j].sum()
151             ##调隐含波动率并计算看跌期权的价值
152             impvol_put = ImpliedVolPut['ImpliedVolatility'][ImpliedVolPut.index == s[j]].
                    tolist()[0]
153             Put = EurOption('put',s0,s[j],(k-1)*T,r,impvol_put)
154 
155             S[j] = round((S[j+1]*(fac*Put-Sigma1) + lam[j]*s[j]*(F[j]-S[j+1]))/(fac*Put-
                    Sigma1+lam[j]*(F[j]-S[j+1])),3)
156 
157             ##判断是否满足无套利条件;
158             if j!= 0:
159                 if S[j]<F[j] and S[j]>F[j-1]:
160                     continue
161                 else:
162                     S[j] = S[j+1]-(s[j+1]-s[j])
163                     S[j] = round(S[j],3)
164             else:
165                 if S[j]<F[0]:
166                     continue
167                 else:
168                     S[j] = S[j+1]-(s[j+1]-s[j])
169                     S[j] = round(S[j],3)
```

```
170
171         print()
172         print('第',k,'个节点')
173         print('StockPrice:',S)
174
175         #probability
176         for m in range(len(p)):
177         p[m] = (F[m]-S[m])/(S[m+1]-S[m])
178         print('Probability:',p)
179
180         #Arrow-Debreu
181         for m in range(len(lam_k)):
182         if m == len(lam_k)-1 :
183         lam_k[m] = 1/fac * p[m-1] * lam[m-1]
184         elif m == 0:
185         lam_k[m] = 1/fac * (1-p[m]) * lam[m]
186         else:
187         lam_k[m] = 1/fac * (p[m-1] * lam[m-1] + (1-p[m]) * lam[m])
188         print('Arrow-Debreu:',lam_k)
189
190         #ForwardPrice
191         for m in range(len(F_k)):
192         F_k[m] = fac * S[m]
193         print('ForwardPrice:',F_k)
194
195         #将得到的4个未知量加入到相应的列表中
196         StockPrice.append(S)
197         ArrowDe.append(lam_k)
198         ForPri.append(F_k)
199         Prob.append(p)
```

四、存数据

```
1  #股票价格数据
2  StockPriceSort = [sorted(i,reverse = True) for i in StockPrice ]
3  StockTree = pd.DataFrame(StockPriceSort).transpose()
4  StockTree.to_csv("/Users/yangyuxin/Desktop/PKU/2021-2022秋季/衍生工具模型/期末大
       作业/Data/StockTree.csv")
5  StockTree
```

```
6
7  #ArrowDebreu价格数据
8  ArrowDeSort = [sorted(i,reverse = True) for i in ArrowDe ]
9  ArrowDeTree = pd.DataFrame(ArrowDeSort).transpose()
10 ArrowDeTree.to_csv("/Users/yangyuxin/Desktop/PKU/2021-2022秋季/衍生工具模型/期
       末大作业/Data/ArrowDeTree.csv")
11 ArrowDeTree
12
13 #风险中性概率数据
14 ProbSort = [sorted(i,reverse = True) for i in Prob]
15 ProbTree = pd.DataFrame(ProbSort).transpose()
16 ProbTree.to_csv("/Users/yangyuxin/Desktop/PKU/2021-2022秋季/衍生工具模型/期末大
       作业/Data/ProbabilityTree.csv")
17 ProbTree
18
19 #远期合约价格数据
20 ForPriSort = [sorted(i,reverse = True) for i in ForPri ]
21 ForPriTree = pd.DataFrame(ForPriSort).transpose()
22 ForPriTree.to_csv("/Users/yangyuxin/Desktop/PKU/2021-2022秋季/衍生工具模型/期末
       大作业/Data/ForwardPriceTree.csv")
23 ForPriTree
```