第1章 二叉树期权定价的数值实验

通过二叉树期权定价模型的数值实验，可以进一步强化学生对二叉树期权定价模型的理解，使学生不但能够掌握期权定价理论方法，而且可以熟练地进行手工计算，此外还可以通过软件及自主编程来完成 N 期二叉树期权定价模型的计算（手工计算无法完成 N 较大的情况）。特别是在现代经济和金融环境下，期权定价模型日益复杂，无论是研究领域还是业界，都是通过计算软件和编程来计算这些期权定价模型。而计算成本的下降，更为模型的算法实现提供了可能。因此，相应的编程技能成为学生适应现代社会特别是从事相关领域研究和实际工作的必备技能。

考虑到不同专业、不同学校学生的计算机能力和编程偏好存在较大差异，本章的模型采用 Excel、MATLAB、Python 和 C++四种金融领域主流建模工具依次实现。衍生品定价，一方面需要高效快速的代码，另一方面需要兼顾代码开发的速度。在衍生品定价的研究与实务中，常用的编程语言包括以下几种：

（1）Excel/VBA：Excel/VBA 是微软为 Office 用户开发的一种基于 Visual Basic 语言的、可在 Office 软件上直接运行的编程语言。用户可以通过在 Office 中内置的 VBA 编辑器直接编写宏或者函数，并在 Excel 中调用。Excel/VBA 的优点是开发速度快、检验方便，缺点是大量使用 VBA 会使表格速度变慢。

（2）MATLAB：MATLAB 是由 MathWorks 公司推出的用于数值计算、可视化及编程的高级语言和交互式环境。MATLAB 由于其矩阵运算能力强、编程开发速度快等特性也备受金融开发人员青睐。

（3）Python：Python 作为一个开源编程语言问世于 1990 年，由于它提供了高效的高级数据结构，能简单有效地面向对象编程，并且有强大的第三方库的支持，近年来逐渐成为财经领域，特别是金融领域的热门编程工具。随着量化投资领域的主流量化平台纷纷选择 Python 语言，Python 更是稳坐金融领域编程工具的首把交椅。这也是我们在编写本书第二版时候增加 Python 这部分内容的原因。

（4）C/C++/C#：C 语言是一种通用的计算机语言，被广泛用于系统和应用软件的开发。C++在 C 语言基础上实现了面向对象的特点。C 类语言由于其高效、灵活、功能丰富、表达力强、移植性强等特性，在金融行业中使用广泛。

本章主要研究二叉树期权定价模型——CRR（Cox，Ross，Rubinstein）模型的算法实现，将主要通过 Excel、MATLAB、Python 和 C++与 Excel-Addin 结合使用四种主流方法来实现。Excel 简单直观，运用宏后功能也很强大，对于不太喜欢编程的读者来说更适合。MATLAB 是主流科学计算软件，有强大的金融类工具箱支撑，近年来在财经领域的占有率越来越高，特别适合有编程背景，现在或将来从事金融工程、量化投资等领域的读者学习掌握。Python 语言作为金融领域的首选编程语言，简单易学，借助其高效的数据结构和强

大的第三方库，在期权定价领域大有用武之地。C++与 Excel-Addin 结合使用，不但保证了计算速度，而且是目前金融、量化投资领域的主流编程语言，有强大的函数库的支持。同时 Excel-Addin 的使用更有利于结果的清晰展示，便于非编程人员使用。

1.1 理 论 基 础

1.1.1 无套利均衡分析与状态价格定价技术

无套利均衡分析的出现具有里程碑的意义，它标志着金融学和经济学的正式分离，是金融学从经济学中独立出来的标志。无套利均衡分析是衍生品定价的核心思想，起源于 1958 年提出的 MM 理论，在推导无税情况下有财务杠杆企业和无财务杠杆企业的价值时，米勒和莫迪里亚尼用自制财务杠杆复制了企业的财务杠杆，从而得出在无税情况下有财务杠杆企业和无财务杠杆企业的价值相同，进一步得出在无税情况下资本结构与企业价值无关的结论。由此可见，无套利均衡分析的关键是复制。在 MM 理论的推导中由于复制是一次性的，因此被称为静态复制。当我们需要引入模型来描述标的资产价格的动态变化时，相应的复制技术就称为动态复制技术。在随后的章节里我们阐述的都是动态复制。

严格的状态价格过程需要给出与之相关的一系列数学背景知识与假设条件。我们这里仅给出最简要的版本。设 $(\Omega,\mathcal{F},\mathbb{P})$ 为一个装备了 $\sigma-$ 代数流的概率空间，设 $\mathcal{F}$ 是一个递增的 $\sigma-$ 代数，也就是域流，可看作到时间 n 为止所获得的信息。随着时间的推移，所获得的信息越来越多，因此有

$$\mathcal{F}_n \subseteq \mathcal{F}_{n+1},\ n=0,1,\cdots,N-1$$

由此可见，这种信息结构具有不断扩展的树图的特征。这里的 N 就是这棵“树”的末期。通俗地讲，状态价格过程就是用来描述资产价格动态变化的过程，也就是构造树形图的过程。这里我们假设资产价格在未来只有两种变化状态：上升或是下降，上升用 u 来表示，下降用 d 来表示。假设有一个金融市场，模型为一期模型（严格地说，一期模型中的复制是静态复制，这里之所以选择一期模型是为了计算简便，而多期模型也并不复杂，仅是单期模型的自然扩展，在后续的算法实现章节，我们处理的都是多期模型），市场上有两种风险资产，均为债券，其中债券 A 的初始价格为 $P_{\mathrm{A}}=100$，设 $u=1.25$，$d=0.8$，则债券在一段时间 $\Delta t=0.25$ 后的价格变化如图 1-1 所示。

假设债券 B 在一段时间 $\Delta t=0.25$ 后的价格变化如图 1-2 所示。

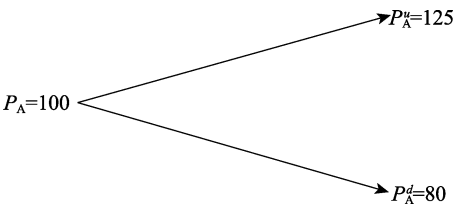


图 1-1 债券 A 的价格变化

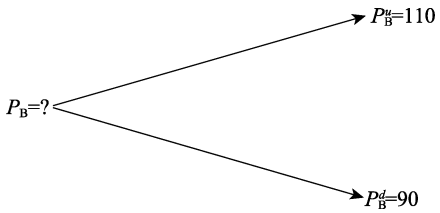


图 1-2 债券 B 的价格变化

那么，债券 B 的期初价格是多少？假设无风险证券的利率为 4%，采用连续复利计算。在对债券 B 进行定价之前，我们给出基础证券的概念。基础证券又称阿罗–德布鲁证券，是一种虚拟证券，它在状态价格定价技术过程中起着关键的作用。每次我们介绍阿罗–德布鲁证券并运用该证券进行动态复制时，都深深地折服于这一概念的构想者，真是简洁而又精妙。此概念既蕴含了金融市场完备性的内容，即两种状态需要两种证券复制这一关键思想，同时又隐含着二进制的内容，当世界复杂多变时，也许最好的出发点就是研究两种状况。阿罗–德布鲁证券是一组证券，一般记为 π_u 和 π_d ， π_u 和 π_d 好似一对开关型证券。其状态价格变化如图 1-3 和图 1-4 所示。

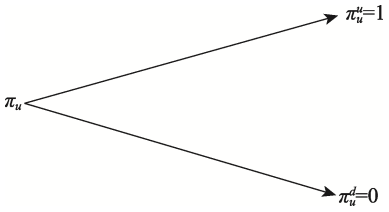


图 1-3　基础债券 π_u 的价格变化

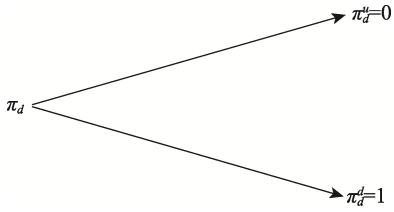


图 1-4　基础债券 π_d 的价格变化

现在，我们需要运用两个基础证券去复制债券 A。我们所说的复制是指对现金流的复制。构造组合 V 如下： $\pi_u \times P_{\mathrm{A}}^u + \pi_{\mathrm{d}} \times P_{\mathrm{A}}^d$ 。组合 V 的价格变化如图 1-5 所示。

由图 1-5 可知，组合 V 复制了债券 A 的期末现金流。由无套利原理可知，当组合 V 在期末等于债券 A 的期末现金流时，期初两者的价值也应该相等，否则就会有套利发生。由此可知：

$$\begin{cases} \pi_u \times P_{\mathrm{A}}^u + \pi_d \times P_{\mathrm{A}}^d = P_{\mathrm{A}} \\ \pi_u \times u + \pi_d \times d = 1 \end{cases} \tag{1-1}$$

由于有两个未知数而只有一个方程，因此无法求出 π_u 和 π_d 。现考察另一组合 $\pi_u + \pi_d$ 的 V^*，如图 1-6 所示。

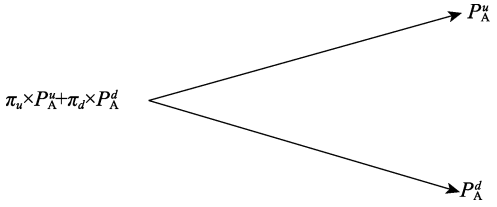


图 1-5　组合 V 的价格变化

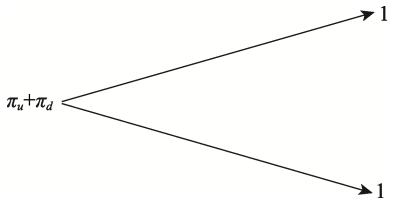


图 1-6　组合 V^*的价格变化

由图 1-6 可知，无论是上升状态还是下降状态，组合 V^*的价值均等于 1。由此可见，组合 V^*是无风险证券，于是得到另一个方程：

$$(\pi_u + \pi_d) \times \mathrm{e}^{r\Delta t} = 1 \tag{1-2}$$

联立方程（1-1）和方程（1-2），可以求得：

$$\pi_u = \frac{e^{r\Delta t} - d}{e^{r\Delta t}(u-d)} \tag{1-3}$$

$$\pi_d = \frac{u - e^{r\Delta t}}{e^{r\Delta t}(u-d)} \tag{1-4}$$

带入式（1-3）和式（1-4）中可得 $\pi_u = 0.462\,1$ 和 $\pi_d = 0.527\,9$。有了 π_u 和 π_d 这两个基础证券，我们就可以为由这两个基础证券“统治”的金融市场上的任何证券定价。这里需要指出的是，虽然 π_u 和 π_d 是由债券 A 的价格推导出来的，但 π_u 和 π_d 是独立于债券 A 的。如果把 π_u 和 π_d 比作两个点的话，两点决定一条直线。而这里的“直线”就是指由基础证券所控制的金融市场。也就是说，一对 π_u 和 π_d 控制一个金融市场，为其上的资产定价，换一组 π_u 和 π_d，控制的则是另一个不同的金融市场。当然这里所说的金融市场是理论上的金融市场而不是现实中的金融市场。现在，我们利用 π_u 和 π_d 为债券 B 定价：$P_B = \pi_u \times P_B^u + \pi_d \times P_B^d = 0.462\,1 \times 110 + 0.527\,9 \times 90 = 98.34$。我们再用 π_u 和 π_d 的值来验证一下债券 A 的价格：$P_A = \pi_u \times P_A^u + \pi_d \times P_A^d = 0.462\,1 \times 125 + 0.527\,9 \times 80 = 99.99 \approx 100$。正好和我们预先给定的债券 A 的初始价格 100 一致。

虽然基础证券是如此美妙而强大，但毕竟它们只是虚拟证券，无法在真实的证券市场中找到，因此在真实的金融市场中，我们需要寻找其他真实的证券参与复制过程来为其他证券定价。在上述为债券 B 定价的过程中，我们自然而然地会想到价格已知的债券 A，而另一个债券我们选择无风险证券 L。构造组合 V 如下：$\Delta \times P_A + L$ 复制债券 B 的现金流量，从而得到如下方程组：

$$\begin{cases} \Delta \times uP_A + e^{r\Delta t} \times L = P_B^u \\ \Delta \times dP_A + e^{r\Delta t} \times L = P_B^d \end{cases} \tag{1-5}$$

解方程组（1-5）可以求得

$$\begin{cases} \Delta = \dfrac{P_B^u - P_B^d}{P_A^u - P_A^d} \\ L = \dfrac{P_B^d \times P_A^u - P_B^u \times P_A^d}{e^{r\Delta t} \times (P_A^u - P_A^d)} \end{cases} \tag{1-6}$$

代入（1-6）中可得

$$\Delta = \frac{110-90}{125-80} = \frac{4}{9} = 0.444\,4$$

$$L = \frac{90 \times 125 - 110 \times 80}{e^{0.04 \times 0.25} \times (125-80)} = \frac{2\,450}{45.452\,2} = 53.90$$

现在，我们利用债券 A 和无风险证券 L 为债券 B 定价：$P_B = \Delta \times P_A + L = \frac{4}{9} \times 100 + 53.90 = 98.34$。由此可见，无论是利用基础证券 π_u 和 π_d 还是利用债券 A 和无风险证券 L 都可以得到相同的债券 B 的价格。以后我们将采用真实金融市场中的金融工具来复制其他金融工具

从而对其定价。

1.1.2 动态复制定价方法

现在，我们将状态价格定价技术中的动态复制定价方法用于期权定价。如果不特别指出，本章研究的都是欧式看涨期权。由于仅考虑上升和下降两种状态，状态价格过程看上去像一个具有两个枝杈的树形图，因此这种期权定价的离散模型一般被称为二叉树期权定价模型。二叉树期权定价模型有很多种，当研究的模型是多期模型时，如果先上升后下降和先下降后上升交汇于一点的话，我们把这样的二叉树期权定价模型称为交叉树。对于一个 N 期模型来说，树的交叉使得最后一期树的节点的数量为 N+1，这使树的状态增加得比较慢，从而具有计算成本上的优势。

假设是 B-S（Bond-Stock）市场，也就是由一种风险资产股票和一种无风险证券构成的金融市场。现在我们研究一个二期模型。假设股票 S 价格的变化如图 1-7 所示，期权的执行价格为 28，无风险利率为 $r_f = 4\%$，连续复利，期权的有效期为 $T = 0.5$ 年，现在利用动态复制定价方法给欧式看涨期权定价。

动态复制的核心是现金流，用于期权定价的时候，复制的是终端支付（payoff）。在期权定价中，终端支付是确定的，而期初价格是不确定的，因此期权也被称为“未定权益”。正因为如此，期权定价是一个如假包换的倒向问题，这一特征在二叉树期权定价模型中体现得非常明显，因为我们总是从终端出发，逆推回期初。为了清楚明了，图 1-8 给出了期权的终端支付。

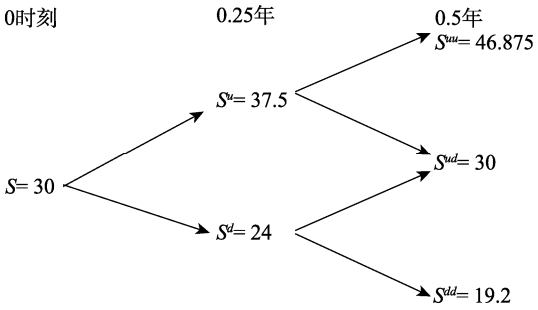


图 1-7　股票 S 的价格变化

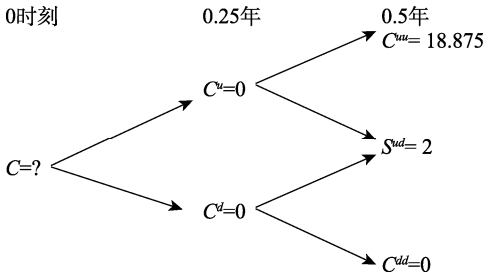


图 1-8　期权的终端支付

在为期权 C 定价的过程中，与前面的做法一致，我们选择标的资产 S 和无风险证券 L。构造组合 A 如下：$\Delta \times P_{\mathrm{A}} + L$ 复制期权的终端支付。这里我们先从右上方的树开始做起，得到如下方程组：

$$\begin{cases} \Delta^u \times S^{uu} + \mathrm{e}^{r\Delta t} \times L^u = C^{uu} \\ \Delta^u \times S^{ud} + \mathrm{e}^{r\Delta t} \times L^u = C^{ud} \end{cases} \tag{1-7}$$

解方程组（1-7）可以求得：

$$\begin{cases}\Delta^u = \dfrac{C^{uu} - C^{ud}}{S^{uu} - S^{ud}} \\ L^u = \dfrac{u \times C^{ud} - d \times C^{uu}}{\mathrm{e}^{r\Delta t} \times (u - d)}\end{cases} \tag{1-8}$$

将数据代入（1-8）中可得

$$\begin{cases}\Delta^u = \dfrac{18.875 - 2}{46.875 - 30} = 1 \\ L^u = \dfrac{1.25 \times 2 - 0.8 \times 18.875}{\mathrm{e}^{0.04 \times 0.25} \times 0.45} = \dfrac{-12.6}{0.454\,5} = -27.72\end{cases}$$

由此可得 $C^u = \Delta^u \times S^u + L^u = 1 \times 37.5 - 27.72 = 9.78$。

同理，我们可以得到

$$\Delta^d = \frac{2 - 0}{30 - 19.2} = 0.185\,2$$

$$L^d = \frac{1.25 \times 0 - 0.8 \times 2}{\mathrm{e}^{0.04 \times 0.25} \times 0.45} = \frac{-1.6}{0.454\,5} = -3.52$$

由此可得 $C^d = \Delta^d \times S^d + L^d = 0.185\,2 \times 24 - 3.52 = 0.924\,8$

进一步计算可以得到

$$\Delta = \frac{9.78 - 0.924\,8}{37.5 - 24} = 0.655\,9$$

$$L = \frac{1.25 \times 0.924\,8 - 0.8 \times 9.78}{\mathrm{e}^{0.04 \times 0.25} \times 0.45} = \frac{-6.668}{0.454\,5} = -14.67$$

由此可得 $C = \Delta \times S + L = 0.655\,9 \times 30 - 14.67 = 5.01$。

动态复制定价方法用于期权定价的具体内容到此似乎已经接近尾声，但故事还没有结束。在前面介绍二叉树期权定价模型的过程中还遗留了一个关键问题，那就是如何选择 u、d，是不是只需要满足无套利条件 $d < \mathrm{e}^{r\Delta t} < u$ 就可以了？事情没有这么简单。从期权定价模型的发展来看，我们都知道最先出现的是基于几何布朗运动的连续时间模型即 BSM 模型，它出现于 1973 年。而这里的二叉树期权定价模型（也称 CRR 模型）出现于 1979 年，它的理论基础在于二项分布的极限是对数正态分布，从而保证二叉树期权定价模型是 BSM 模型的离散版本，这必然导致二叉树期权定价模型的参数选择需要匹配 BSM 模型的参数。鉴于本教程不打算详细阐述连续时间模型，因此这一问题也无法在这里进行深入探讨。[①]已经有不少学者给出了不同的参数匹配方案，在这些方案里二叉树期权定价模型的参数选择最为简单，从而使该模型得到了最广泛的使用。假设 BSM 模型下，波动率为 σ，时间间隔为 Δt，则在二叉树期权定价模型下，参数 u、d 应该这样选择：$u = \mathrm{e}^{\sigma\sqrt{\Delta t}}, d = \mathrm{e}^{-\sigma\sqrt{\Delta t}}$。

① 感兴趣的读者可以参阅作者主编的《期权与期货定价》相关章节，其中讨论了其他二叉树期权定价模型下参数的匹配方案，本书由中国人民大学出版社于 2021 年 10 月出版。

1.1.3 风险中性定价方法

本部分阐述的是二叉树期权定价模型的风险中性定价方法。在不少衍生品定价的经典教材中，为了降低数学难度，多从动态复制定价方法的数学变形引出风险中性定价，然后直接给出风险中性概率的定义和计算公式。这样的处理方法的确有一定的好处，绕过了很多艰深的数学概念。但在多年的教学过程中我们发现这样处理的直接后果就是绝大多数同学认为风险中性定价方法是动态复制定价方法的数学简便运算解决途径，这实在是大大扭曲了风险中性定价方法的本质和地位。这里我们试图用简洁的语言和最少的篇幅给出风险中性定价方法的真实全貌。在这一过程中鞅（这里指的是离散鞅）和金融资产定价基本原理是最核心的内容。

1. 鞅

设 $(\Omega,\mathcal{F},\mathbb{P})$ 为一个装备了 σ 代数流的概率空间，设 $\mathcal{F}$ 是一个递增的 σ 代数。如果 $\forall n, S_n$ 关于 $\mathcal{F}_n$ 可测（简记为 $S_n \in \mathcal{F}_n$），则称 $\{S_n : n \geqslant 0\}$ 是适应的。对于适应过程我们写为 $(S_n,\mathcal{F}_n)$，$n \geqslant 0$。如果 $\forall n, S_n \in \mathcal{F}_{n-1}$，则称 S_n 是可料的。关于可测、可料等概念的详细内容，感兴趣的同学可以查阅关于随机分析的书籍，这里不再详述。

下面给出鞅的定义。称 $(S_n,\mathcal{F}_n)_{n\geqslant 0}$ 为鞅（上鞅，下鞅），如果 $\forall n, S_n$ 可积，且

$$E(S_{n+1} \mid \mathcal{F}_n) = S_n \quad (\text{相应地，} \leqslant S_n\text{，} \geqslant S_n) \tag{1-9}$$

假设风险资产股票用 S_n 来表示，无风险证券用 B_n 来表示。而资产的贴现价格过程用 $\overline{S}_n$ 来表示。等鞅测度 Q 是定义在 $(\Omega,\mathcal{F},\mathbb{P})$ 上且满足如下条件的概率测度：

（1）对于所有的 $w \in \Omega$，均有 $Q(w) > 0$ 和 $\sum_{\bar{\omega}} Q(w) = 1$；

（2）对于任一 $n \geqslant 0$ 在 Q 测度下，贴现价格过程 $\overline{S_n}$ 是一个鞅。

2. 金融资产定价基本原理

需要声明的是，我们这里给出的金融资产定价基本原理是一个简单版本由三部分组成：

（1）对于有限离散时间的金融市场，如果市场无套利，那么存在一个等鞅测度；

（2）如果市场是完备的，那么等鞅测度是唯一的；

（3）在这个等鞅测度下，任一风险资产的贴现价格过程都是一个鞅。

如果金融资产定价基本原理成立，那么期权的价格等于在等鞅测度下的终端支付的期望的折现值。

要想运用风险中性定价方法计算二叉树期权定价模型，那么，确定风险中性概率是问题的关键。由于其是离散模型，因此问题并不太复杂。我们自然而然的出发点就是寻找一个“概率”，虽然这一风险中性概率不是真实概率，但它既然被称为概率，就应该符合概率的基本要求。我们首先想到的是，它的取值范围应该在 0 和 1 之间。回顾前已述及的在无套利条件下，u、d 应该满足如下条件：

$$d < \mathrm{e}^{r\Delta t} < u \tag{1-10}$$

在不等式的两端同时减去 d，得到

$$0 < e^{r\Delta t} - d < u - d \tag{1-11}$$

在不等式两端同时除以一个正数 $u-d$，得到

$$0 < \frac{e^{r\Delta t} - d}{u - d} < 1 \tag{1-12}$$

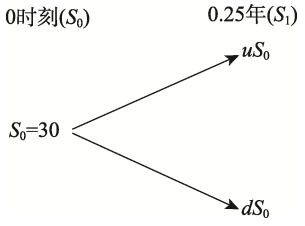


图 1-9　股票价格变化（一期模型）

由（1-12）可知，$\frac{e^{r\Delta t} - d}{u - d}$ 是一个 0 到 1 之间的数。现在我们将其定义为 p^*。

现在我们需要验证 p^* 是否就是我们想要寻找的风险中性概率。为了简单起见，我们以一期模型进行验证。参数取值和上一节一致。股票价格变化如图 1-9 所示。

首先我们验证一下存在性，这比较显而易见，因为我们已经找到这一概率。现在我们验证股票贴现过程是否为鞅。具体分析如下：

$$\begin{aligned} E^*(e^{-r\Delta t}S_1) &= e^{-r\Delta t}E^*(S_1) \\ &= e^{-r\Delta t}\left[\frac{e^{r\Delta t} - d}{u - d} \times uS_0 + \left(1 - \frac{e^{r\Delta t} - d}{u - d}\right)dS_0\right] = S_0 \end{aligned}$$

由此可见，股票贴现价格过程在该概率测度下是鞅。接下来需要验证的是测度的唯一性，假设存在另一个等鞅测度 $\overline{p} \neq \frac{e^{r\Delta t} - d}{u - d}$，则

$$\begin{aligned} E^*(e^{-r\Delta t}S_1) &= e^{-r\Delta t}E^*(S_1) \\ &= e^{-r\Delta t}\left[\overline{p} \times uS_0 + (1 - \overline{p})dS_0\right] \\ &\neq e^{-r\Delta t}\left[\frac{e^{r\Delta t} - d}{u - d} \times uS_0 + \left(1 - \frac{e^{r\Delta t} - d}{u - d}\right)dS_0\right] \\ &\neq S_0 \end{aligned}$$

由此可知，当 $\overline{p} \neq \frac{e^{r\Delta t} - d}{u - d}$ 时，贴现价格过程不是鞅，从而反证出该概率测度的唯一性。

到现在为止，金融资产定价基本原理的三个条件都已经具备了，因此我们可以使用风险中性定价——鞅定价方法为期权定价，也就是说，欧式期权的价格等于风险中性概率测度下期权终端支付的折现值。由于是风险中性概率测度，因此折现率采用无风险利率。继续采用上面给出的参数，风险中性定价过程如下：

①计算风险中性概率：

$$p^* = \frac{e^{r\Delta t} - d}{u - d} = \frac{e^{0.04 \times 0.25} - 0.8}{1.25 - 0.8} = \frac{0.210\,05}{0.45} = 0.466\,8$$

②计算期权的价格：

$$
\begin{aligned}
C &= e^{-2r\Delta t}[p^{*2}C^{uu} + 2p^{*}(1-p^{*})C^{ud} + (1-p^{*})^{2}C^{dd}] \\
&= 0.980\,2\times(0.217\,9\times 18.875 + 0.995\,6 + 0) \\
&= 0.980\,2\times 5.108\,5 \\
&= 5.01
\end{aligned}
$$

通过风险中性定价方法得到的欧式看涨期权的价格为 5.01，和前面运用动态复制定价得到的欧式看涨期权的价格相同。与动态复制定价方法相比，风险中性定价方法不但在理论基础上更为高端，从计算形式上也更加简洁。因此，在离散的二叉树期权定价模型计算中，风险中性定价方法更为普遍。下面我们给出一个 N 期模型的欧式看涨期权风险中性定价的一般表达式：

$$
C = \mathrm{e}^{-nr\Delta t}\sum_{j=0}^{n}\frac{n!}{j!(n-j)!}p^{*j}(1-p^{*})^{n-j}C^{u^{j}d^{(n-j)}} \qquad (1\text{-}13)
$$

以上我们讨论的都是欧式看涨期权，现在我们探讨美式期权。无论是采用动态复制定价方法还是风险中性方法，都需要在从末期向前推算期权价格的过程中，在每个节点上比较期权的理论价格和期权立即执行的支付的大小，保留较大的那个值，并在此基础上进一步往前推算。由此可见，即使运用风险中性定价方法计算美式期权，也不能采用式（1-13）的一般表达式，而需要逐步计算。这里需要指出的是，理论上可以证明，在没有股利支付的情况下，美式看涨期权不会提前实施，因此美式看涨期权价格和欧式看涨期权的价格相等。但是在股价较低的情况下，美式看跌期权会提前实施，因此美式看跌期权的价格高于欧式看跌期权。由于在后面的算法实现中我们将系统计算欧式期权和美式期权的价格，这里就不再给出计算美式期权价格的例子了。

1.2 基于 Excel 的二叉树期权定价的数值实验

1.2.1 实验目的

要求学生使用 Excel 运用动态复制定价和风险中性定价两种方法分别计算二叉树期权定价模型框架下的欧式看涨期权、欧式看跌期权、美式看涨期权、美式看跌期权。要求学生使用 Excel 计算 BSM 期权定价模型框架下的欧式看涨期权、欧式看跌期权，并与二叉树期权定价模型计算结果进行对比分析。

1.2.2 编程准备

Excel 中的函数其实是一些预定义的公式，它们使用一些称为参数的特定数值按特定的顺序或结构进行计算。用户可以直接用它们对某个区域内的数值进行一系列运算。下面给出一些算法实现中需要调用的主要函数。

1. IF 函数

【主要功能】根据对指定条件的逻辑判断的真假结果，返回相对应的内容。

【使用格式】IF(Logical,Value_if_true,Value_if_false)

【参数说明】Logical 代表逻辑判断表达式；Value_if_true 表示当判断条件为逻辑“真（TRUE）”时的显示内容，如果忽略，则返回“TRUE”；Value_if_false 表示当判断条件为逻辑“假（FALSE）”时的显示内容，如果忽略，则返回“FALSE”。

2. MAX 函数

【主要功能】求出一组数中的最大值。

【使用格式】MAX（number1，number2……）

【参数说明】number1，number2……代表需要求最大值的数值或引用的单元格（区域）。

3. ABS 函数

【主要功能】求出相应数字的绝对值。

【使用格式】ABS（number）

【参数说明】number 代表需要求绝对值的数值或引用的单元格。

4. MOD 函数

【主要功能】求出两数相除的余数。

【使用格式】MOD（number，divisor）

【参数说明】number 代表被除数；divisor 代表除数。

5. EXP 函数

【主要功能】返回 e 的 *n* 次幂。

【使用格式】EXP（number）

【参数说明】number 为底数 e 的指数。

1.2.3 实验数据

由于二叉树期权定价计算属于数值实验，因此实验所需数据是通过二叉树也就是二项分布的演变形成计算所需的标的资产价格树的，实验所需的其他参数是考虑该参数的真实取值区间并考虑到计算的便利而外生给定的。所需的主要数据包括股票的初始价格、期权的执行价格、无风险利率、股票的上升与下降幅度、股票价格的波动率、期权的有效期和二叉树期权定价模型的期数。

1.2.4 实验过程

1. 动态复制定价方法

在利用 Excel 建立二叉树期权定价模型时需要重点解决以下三个问题：

（1）如何建立股票价格树，并且使得整棵“树”形状呈现二叉树型；

（2）在建立期权价格树时如何体现不同期权类型的特征；

（3）在建立期权价格树时如何应用动态复制定价方法与风险中性定价方法。

下面给出期权定价建模的基本资料，一般情况下，如不特殊说明，在 Excel、MATLAB

和 Python 算法实现部分我们均采用该例资料。

【例 1-1】 期初股价 $S_0=30$，执行价格 $K=28$，无风险利率 $r=0.04$，连续复利计算，期权有效期 $T=1$，股价上升 $u=1.25$，下降为 $d=0.8$。$n=10$，即该期权定价模型为 10 期模型。采用二叉树期权定价模型中的动态复制定价方法，在 Excel 中计算出以该股票为标的资产的欧式看涨期权的价格。

1）建立基础数据表格

在建立具体的二叉树期权定价模型前，我们需要设计建立基础数据输入表格和结果输出表格，以便在建立模型时引用数据，同时也增强其在不同基础数据下的适用性。表格中需要包括 6 个基础数据输入项：期初股价 S_0，执行价格 K，无风险利率 r，期权有效期 T，股价上升 u，下降为 d。（二叉树期权定价模型中，$u=e^{\sigma\sqrt{\Delta t}}$，$d=e^{-\sigma\sqrt{\Delta t}}$，$ud=1$），同时还需要预留结果输出空间。由于我们需要使用动态复制定价和风险中性定价两种方法，因此安排两种方法的输出结果并列显示，从而便于比较分析。具体的基础数据表格布局可参见图 1-10。

	A	B	C	D
1				
2		基础数据		
3			动态复制	风险中性
4		S		
5		K		
6		T		
7		r		
8		u		
9		d		
10		结果		
11		欧式看涨		
12		欧式看跌		
13		美式看涨		
14		美式看跌		

图 1-10　基础数据输入与结果输出的 Excel 表格布局

2）建立股票价格树

首先我们需要在新工作表中建立坐标系，如图 1-11 中所示。目的是方便将工作区域划分为特征不同的四个区块，即图 1-11 中的白色、红色、灰色、蓝色区块，进而我们可以针对区块的不同特征编写不同的代码。

图 1-11　Excel 中坐标系建立与各区域示意

在进行具体的代码编写前，我们要先明确不同颜色区块的特征。白色区块：此区块内

单元格纵坐标的绝对值大于横坐标的。从而我们可以使用 IF 函数实现从整个工作区域中划分出此区块。条件为横坐标的绝对值大于纵坐标的，当条件判断为真时，单元格值为空，在 Excel 中可以使用“”表示空。当条件判断为假时，我们需要继续使用 IF 函数实现下一个区块的划分。红色区块：我们需要将此区块内单元格细分为三部分。（0，0）点的值应该为初始股票价格。红色区块的上半支特征为纵坐标的绝对值等于横坐标的，其值应该为 $S_0 \times d^{\text{该单元格横坐标}}$。红色区块的下半支特征为纵坐标的绝对值与横坐标的绝对值互为相反数，其值应该为 $S_0 \times d^{\text{该单元格横坐标}}$。蓝色区块：此区块单元格左侧第二个单元格一定有数字，因此可利用 IF 函数划分出该区块，其数值等于 左侧第二个单元格$\times u \times d$（左侧第二个单元格*u*d）。灰色区块：此区块单元格左侧第二个单元格一定为空或者为 0，同样可用 IF 函数划分出来。

将公式应用于工作区域内全部单元格，我们便得到了股票价格树，如图 1-12 所示。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2			0	1	2	3	4	5	6	7	8	9	10
3		10											279.40
4		9										223.52	
5		8									178.81		178.81
6		7								143.05		143.05	
7		6							114.44		114.44		114.44
8		5						91.55		91.55		91.55	
9		4					73.24		73.24		73.24		73.24
10		3				58.59		58.59		58.59		58.59	
11		2			46.88		46.88		46.88		46.88		46.88
12		1		37.50		37.50		37.50		37.50		37.50	
13		0	30.00		30.00		30.00		30.00		30.00		30.00
14		-1		24.00		24.00		24.00		24.00		24.00	
15		-2			19.20		19.20		19.20		19.20		19.20
16		-3				15.36		15.36		15.36		15.36	
17		-4					12.29		12.29		12.29		12.29
18		-5						9.83		9.83		9.83	
19		-6							7.86		7.86		7.86
20		-7								6.29		6.29	
21		-8									5.03		5.03
22		-9										4.03	
23		-10											3.22

图 1-12　Excel 中股票价格演化树形图

3）建立期权价格树

在新的工作表中参考股票价格树的思路建立期权价格树。同样，我们需要建立自己的坐标系，将工作区域划分为四个区块，如图 1-13 所示。

白色区块：可参考股票价格树白色区块的划分方式。红色区块：对于第 10 列单元格，我们需要将其划分为红色部分和灰色部分，红色部分单元格的横、纵坐标之和为偶数，其值应该为期末的支付，即 $\max(S_T - K, 0)$。又因为例 1-1 中的期权为欧式看涨期权，期末支付即为对应股票价格树单元格的值减去期权的执行价格。而剩下的灰色部分单元格横纵坐标之和为奇数，其值应该为空。

灰色区块：该区块单元格右上方一定为空，从而我们可以利用 IF 函数划分出此区块。

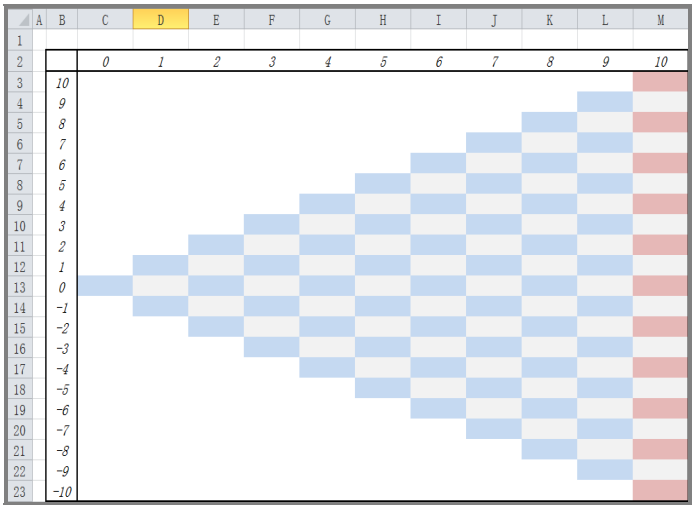


图 1-13　Excel 中期权价格树坐标系建立与各区域示意

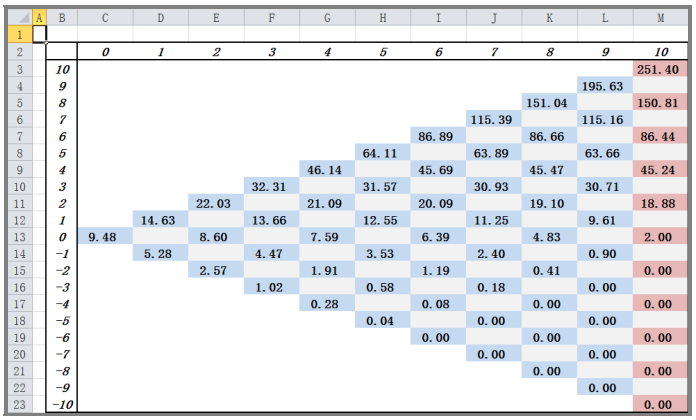
蓝色区块：该区块单元格右上方不为空，其值可以根据二叉树动态复制定价方法得出，具体可表示为其右上方和右下方两个单元格及对应股票价格树单元格的表达式。以 L4 单元格为例，其值可表达为：

```
股票价格树!L4*(M3-M5)/(股票价格树!M3-股票价格树!M5)+(M3-(M3-M5)/(股票价格树!M3-股票价格树!M5)*股票价格树!M3)/EXP(输入与输出!$C$6*输入与输出!$C$5/10)
```

例 1-1 期权价格树单元格所填的具体公式可参考如下代码（以 E13 单元格为例）：

```
=IF(ABS($B13)>E$2,"",IF(AND(E$2=10,MOD($B13,2)=0),MAX(股票价格树!E13-输入与输出!$C$4,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),股票价格树!E13*(F12-F14)/(股票价格树!F12-股票价格树!F14)+(F12-(F12-F14)/(股票价格树!F12-股票价格树!F14)*股票价格树!F12)/EXP(输入与输出!$C$6*输入与输出!$C$5/10),""))
```

将公式应用于工作区域内全部单元格，即得到了例 1-1 的期权价格树。



	0	1	2	3	4	5	6	7	8	9	10
10											251.40
9										195.63	
8									151.04		150.81
7								115.39		115.16	
6							86.89		86.66		86.44
5						64.11		63.89		63.66	
4					46.14		45.69		45.47		45.24
3				32.31		31.57		30.93		30.71	
2			22.03		21.09		20.09		19.10		18.88
1		14.63		13.66		12.55		11.25		9.61	
0	9.48		8.60		7.59		6.39		4.83		2.00
-1		5.28		4.47		3.53		2.40		0.90	
-2			2.57		1.91		1.19		0.41		0.00
-3				1.02		0.58		0.18		0.00	
-4					0.28		0.08		0.00		0.00
-5						0.04		0.00		0.00	
-6							0.00		0.00		0.00
-7								0.00		0.00	
-8									0.00		0.00
-9										0.00	
-10											0.00

图 1-14　Excel 中欧式看涨期权价格树形图

4）输出结果数据

期权价格树中（0，0）点的值即为例 1-1 的答案，因此最后一步就是将该单元格的值赋给第一步建立的基础数据表格中。

【例 1-2】 假设例 1-1 中其他条件不变，试利用 Excel 计算欧式看跌期权的价格。

对于例 1-2 的计算，我们仍将其划分为四个步骤，即建立基础数据表格、建立股票价格树、建立期权价格树，以及输出结果数据。而其中的第一步、第二步以及第四步都与例 1-1 相同，仅第三步有所差别，因此我们不妨继续在例 1-1 的工作簿中继续进行例 1-2 的计算。

与例 1-1 相比，例 1-2 唯一的不同在于期权类型变为欧式看跌期权。因此，我们只需要在例 1-1 的期权价格树中针对最后一期的支付做出修改即可。对于欧式看跌期权，期末支付应该等于 $MAX(K-S_T,0)$，我们只需要将例 1-1 代码中的 MAX（股票价格树!E13-输入与输出!$C$4,0）更改为 MAX（输入与输出!$C$4-股票价格树!E13,0），即得到例 1-2 的代码。例 1-2 的期权价格树参考图 1-15。

	0	1	2	3	4	5	6	7	8	9	10
10											0.00
9										0.00	
8									0.00		0.00
7								0.00		0.00	
6							0.00		0.00		0.00
5						0.00		0.00		0.00	
4					0.23		0.00		0.00		0.00
3				0.95		0.42		0.00		0.00	
2			2.27		1.55		0.77		0.00		0.00
1		4.14		3.38		2.50		1.42		0.00	
0	6.38		5.72		4.93		3.95		2.61		0.00
-1		8.29		7.69		6.98		6.07		4.79	
-2			10.49		10.05		9.54		8.98		8.80
-3				12.88		12.67		12.49		12.53	
-4					15.33		15.35		15.49		15.71
-5						17.65		17.84		18.06	
-6							19.69		19.91		20.14
-7								21.37		21.60	
-8									22.74		22.97
-9										23.86	
-10											24.78

图 1-15　Excel 中欧式看跌期权价格树形图

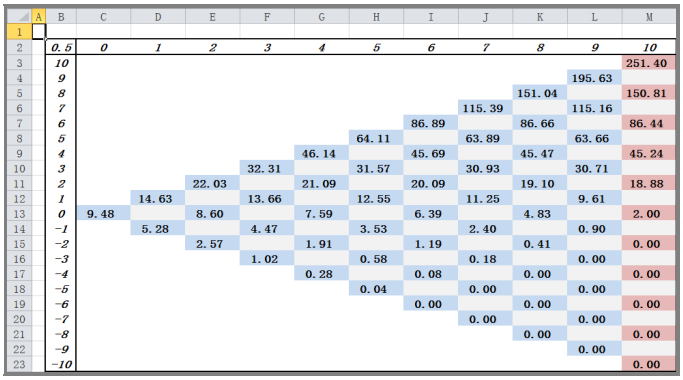
【例 1-3】 假设例 1-1 中其他条件不变，试利用 Excel 计算美式看涨期权的价格。

例 1-3 同例 1-1 相比，期权类型变为美式期权，对于美式期权来说，其每期的期权价格都需要与即时执行的支付相比并取较大值，所以只需在例 1-1 解法的第三步做适当修改即可。

在例 1-1 代码的基础上做修改，引入一个 max 函数，即得到如下代码：

```
=IF(ABS($B13)>E$2, "",IF(AND(E$2=10,MOD($B13,2)=0),MAX(股票价格树!E13-输入与输出!$C$5,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),MAX(股票价格树!E13-输入与输出!$C$5,股票价格树!E13*(F12-F14)/(股票价格树!F12-股票价格树!F14)+(F12-(F12-F14)/(股票价格树!F12-股票价格树!F14)*股票价格树!F12)/EXP(输入与输出!$C$7*输入与输出!$C$6/10)),")))
```

最终的计算结果与欧式看涨期权价格相同，这与理论结果一致，也就是无股利支付的美式看涨期权的价格等于欧式看涨期权的价格，具体建立的期权价格树如图 1-16 所示。



	0.5	0	1	2	3	4	5	6	7	8	9	10
10												251.40
9											195.63	
8										151.04		150.81
7									115.39		115.16	
6								86.89		86.66		86.44
5							64.11		63.89		63.66	
4						46.14		45.69		45.47		45.24
3					32.31		31.57		30.93		30.71	
2				22.03		21.09		20.09		19.10		18.88
1			14.63		13.66		12.55		11.25		9.61	
0		9.48		8.60		7.59		6.39		4.83		2.00
−1			5.28		4.47		3.53		2.40		0.90	
−2				2.57		1.91		1.19		0.41		0.00
−3					1.02		0.58		0.18		0.00	
−4						0.28		0.08		0.00		0.00
−5							0.04		0.00		0.00	
−6								0.00		0.00		0.00
−7									0.00		0.00	
−8										0.00		0.00
−9											0.00	
−10												0.00

图 1-16　Excel 中美式看涨期权价格树形图

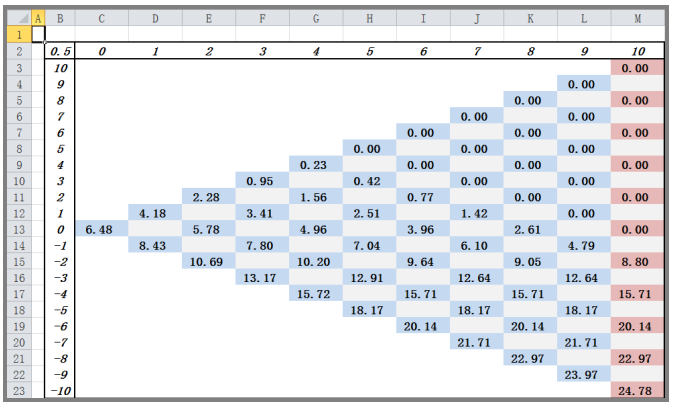
【例 **1-4**】 假设例 1-1 中其他条件不变，试利用 Excel 计算美式看跌期权的价格。

例 1-4 同例 1-2 相比期权类型变为了美式，同例 1-3 相比期权类型变为看跌期权。因此，我们可以结合例 1-2 和例 1-3 的代码特点，对第三步做适当修改。

在例 1-3 代码的基础上做修改，改变支付的形式，即得到如下代码：

```
=IF(ABS($B13)>E$2,"",IF(AND(E$2=10,MOD($B13,2)=0),MAX(-股票价格树!E13+输入与输出!$C$5,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),MAX(输入与输出!$C$5-股票价格树!E13,股票价格树!E13*(F12-F14)/(股票价格树!F12-股票价格树!F14)+(F12-(F12-F14)/(股票价格树!F12-股票价格树!F14)*股票价格树!F12)/EXP(输入与输出!$C$7*输入与输出!$C$6/10)),")))
```

最终计算结果与欧式看跌期权价格不一致，期权价格略大于欧式看跌期权，具体建立的期权价格树如图 1-17 所示。



	0.5	0	1	2	3	4	5	6	7	8	9	10
10												0.00
9											0.00	
8										0.00		0.00
7									0.00		0.00	
6								0.00		0.00		0.00
5							0.00		0.00		0.00	
4						0.23		0.00		0.00		0.00
3					0.95		0.42		0.00		0.00	
2				2.28		1.56		0.77		0.00		0.00
1			4.18		3.41		2.51		1.42		0.00	
0		6.48		5.78		4.96		3.96		2.61		0.00
−1			8.43		7.80		7.04		6.10		4.79	
−2				10.69		10.20		9.64		9.05		8.80
−3					13.17		12.91		12.64		12.64	
−4						15.72		15.71		15.71		15.71
−5							18.17		18.17		18.17	
−6								20.14		20.14		20.14
−7									21.71		21.71	
−8										22.97		22.97
−9											23.97	
−10												24.78

图 1-17　Excel 中美式看跌期权价格树形图

2. 风险中性定价方法

【**例 1-5**】 期初股价 $S_0=30$，执行价格 $K=28$，无风险利率 $r=0.04$，连续复利计算，期权有效期 $T=1$，股价上升 $u=1.25$，下降 $d=0.8$。$n=10$，即该期权定价模型为 10 期模型。采用二叉树期权定价模型中的风险中性定价方法，在 Excel 中计算以该股票为标的资

产的欧式看涨期权的价格。

例 1-5 与例 1-1 的区别表现在其应用的是风险中性定价方法。解题思路仍是在例 1-1 的基础上做出相应的修改，将第三步需做出适当改变。例 1-5 的风险中性定价方法通过蓝色区块单元格的变化而实现。风险中性定价方法的应用首先需要得到风险中性概率，套用风险中性概率的计算公式得到其数值，不妨将其表达在工作区域的左上角单元格中。风险中性定价方法导致蓝色区块单元格的值应当等于其右上角和右下角两个单元格的数学表达式。以 L4 格为例，其值可表达为：

$$\max\left[\frac{P\times L3+(1-P)\times L5}{\mathrm{e}^{r\times\Delta t}},0\right]$$

从而以 k_{13} 单元格为例，例 1-5 的公式可参考如下代码：

```
=IF(ABS($B13)>E$2,",IF(AND(E$2=10,MOD($B13,2)=0),MAX(股票价格树!E13-输入与输出!$C$5,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),($B$2*F12+(1-$B$2)*F14)/EXP(输入与输出!$C$7*输入与输出!$C$6/10),")))
```

最终结果与例 1-1 的结果相同。

【例 1-6】 假设例 1-6 中其他条件不变，试利用 Excel 计算欧式看跌期权的价格。

例 1-6 与例 1-5 的区别在于期权类型变为看跌期权，只需将例 1-5 代码中的期权支付表达式做出更改即可。具体代码如下：

```
=IF(ABS($B13)>E$2,",IF(AND(E$2=10,MOD($B13,2)=0),MAX(-股票价格树!E13+输入与输出!$C$5,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),($B$2*F12+(1-$B$2)*F14)/EXP(输入与输出!$C$7*输入与输出!$C$6/10),")))
```

最终结果与例 1-2 的结果相同。

【例 1-7】 假设例 1-5 中其他条件不变，试利用 Excel 计算美式看涨期权的价格。

例 1-7 与例 1-1 的区别表现在其应用的是风险中性定价方法并且是美式期权。解题思路仍是在例 1-1 的基础上做出相应的修改。第一步、第二步、第四步仍可套用例 1-1 中的模式。第三步需做出适当改变。

例 1-7 风险中性定价方法及美式期权特征的体现都可以通过蓝色区块单元格的变化而实现。风险中性定价方法的应用首先需要得到风险中性概率，套用风险中性概率的计算公式得到其数值，不妨将其表达在工作区域的左上角单元格中。风险中性定价方法导致蓝色区块单元格的值应当等于其右上角和右下角两个单元格的数学表达式，而美式期权的特征又决定了上一步的值还需要与即时执行的支付相比并取较大值。

例 1-7 的公式可参考如下代码：

```
=IF(ABS($B13)>E$2,",IF(AND(E$2=10,MOD($B13,2)=0),MAX(股票价格树!E13-输入与输出!$C$4,0),IF(AND(E$2<10,MOD($B13+E$2,2)=0),MAX(($B$2*F12+(1-$B$2)*F14)/EXP(输入与输出!$C$6*输入与输出!$C$5/10),股票价格树!E13-输入与输出!$C$4),")))
```

最终结果与例 1-3 的结果相同。

【例 1-8】假设例 1-5 中其他条件不变，试利用 Excel 计算美式看跌期权的价格。

例 1-8 与例 1-7 相比只是期权类型发生了变化，由看涨期权变为看跌期权，因此我们可在例 1-7 期权价格树中做出适当修改。第一步、第二步、第四步省略。因为例 1-8 是一个看跌期权，所以只需要将期权支付的形式改为 $\max(K-S_T,0)$ 即可。例 1-8 的结果与例 1-4 的结果相同。最终结果如图 1-18 所示。

	A	B	C	D
1				
2		基础数据		
3			动态复制	风险中性
4		S	30.00	
5		K	28.00	
6		T	1.00	
7		Rf	0.04	
8		u	1.25	
9		d	0.80	
10		结果		
11		欧式看涨	9.48	9.48
12		欧式看跌	6.38	6.38
13		美式看涨	9.48	9.48
14		美式看跌	6.48	6.48

图 1-18　运用风险中性定价方法和动态复制定价方法计算的四种期权的价格

也许有的读者认为无须对风险中性定价方法和动态复制定价方法的算法实现进行如此详细的阐述。我们之所以这样做，是因为虽然风险中性定价方法优于动态复制定价方法，但这两种定价方法是期权定价中最为重要的两种方法，无论是从理论上还是算法实现上都应该全面、熟练地掌握，同时这也有利于大家在后续学习中理解连续时间框架下这两种方法的区别所在。动态复制定价方法在连续时间模型中得到的是期权的偏微分方程，而风险中性定价方法借助的是鞅论，无论从理论上还是从所用数学上工具上两者都存在很大差异，但最后都能得到相同的 BSM 期权定价公式。而在二叉树期权定价模型中两者差别比较明显，因此使用了较多的篇幅进行阐述。期望通过这样的阐述，使学生一方面熟练掌握有关建模的过程，另一方面通过建模加深对两者之间区别的理解。

为了增加模型算法实现的适用性，我们将二叉树期权定价模型的期数也作为输入变量。有了前面期数固定二叉树期权定价模型的建模基础，我们可以很快得到期数可变的二叉树期权定价模型。

	A	B	C
1			
2		基础数据	
3			动态复制
4		S	
5		K	
6		T	
7		Rf	
8		u	
9		d	
10		n	
11		结果	
12		案例9	

图 1-19　数据输入和输出的 Excel 布局——适用于期数可变的 CRR 模型

【例 1-9】假设例 1-1 中其他条件不变，利用 Excel 计算不同期数二叉树期权定价模型下期权的价格。

同上述例子不同的是，本例的期权类型并没有发生改变，而只是要求二叉树期权定价模型的期数 n 从常数变为变量。我们仍然沿着例 1-1 的思路建立 Excel，但需要做出适当改变。

1）建立基础数据表格

本例由于引入了一个新的变量 n，所以基础数据表格中将会增加一行，具体形式如图 1-19 所示。

2）建立股票价格树

首先是建立自己的坐标系，但是坐标的最大值不再是常数 10，而是我们自己设定的期数 n 所能取到的

最大值，不妨将其设置为 30。现在的坐标平面可以划分为四个部分，即以期数 *n* 为界的矩形区块，以及该区块上方区块、下方区块及右侧区块。我们可以使用 IF 函数实现这些区块的划分，比如，如果单元格的横坐标大于给定的 *n* 值，则该单元格划分在右侧区块中。以期数 *n* 为界的矩形区块单元格的代码可套用例 1-1 的代码，只是把常数 10 更改为期数 *n* 所在的单元格即可。具体代码可参考（以 E33 为例）：

```
    =IF(E$2>例 1-9_1!$C$10,",IF(ABS($B33)>E$2,",IF($B33=E$2,例 1-9_1!$C$4*例
1-9_1!$C$8^E$2,IF($B33=-E$2,例 1-9_1!$C$4*例 1-9_1!$C$9^E$2,IF(OR(C33=",
C33=0),",C33*例 1-9_1!$C$8*例 1-9!$C$9)))))
```

股票价格树形式如图 1-20 所示（以 *n*=8 为例）。

10								
9								
8								178.81
7							143.05	
6						114.44		114.44
5					91.55		91.55	
4				73.24		73.24		73.24
3			58.59		58.59		58.59	
2		46.88		46.88		46.88		46.88
1	37.50		37.50		37.50		37.50	
0	30.00	30.00		30.00		30.00		30.00
-1	24.00		24.00		24.00		24.00	
-2		19.20		19.20		19.20		19.20
-3			15.36		15.36		15.36	
-4				12.29		12.29		12.29
-5					9.83		9.83	
-6						7.86		7.86
-7							6.29	
-8								5.03
-9								
-10								

图 1-20　Excel 中股票价格演化树形图（*n*=8）

3）建立期权价格树

同样建立如同第二步中的坐标系，利用 IF 函数划分出四个区块，其中上侧、下侧、右侧区块均为空，而中间区块则是期权价格树所处的位置。中间部分单元格的代码可在例 1-1 代码基础上做稍许修改，即将常数 10 改为期数 *n* 所处的单元格。具体代码参考如下：

```
    =IF(OR(ABS($B33)>例 1-9_1!$C$10,E$2>例 1-9_1!$C$10),"",IF(ABS($B33)>E$2,",
IF(AND(E$2=例 1-9_1!$C$10,MOD($B33,2)=MOD(例 1-9_1!$C$10,2)),MAX(例 1-9_
2!E33-例 1-9_1!$C$5,0),IF(AND(E$2<例 1-9_1!$C$10,MOD($B33+E$2,2)=0),例 1-9_
2!E33*(F32-F34)/(例 1-9_2!F32-例 1-9_2!F34)+(F32-(F32-F34)/(例 1-9_ 2!F32-
例 1-9_2!F34)*例 1-9_2!F32)/EXP(例 1-9_1!$C$7*例 1-9_1!$C$6/例 1-9_ 1!$C$ 10),"))))
```

最终期权价格树结果的一部分如图 1-21 所示。

4）输出结果

此部分可完全参照例 1-1 的做法，将 C33 作为最终结果输入基础数据表格的结果输出位置。最终结果参考图 1-22 所示。

10										150.81
9									115.19	
8								86.72		86.44
7							63.97		63.69	
6						45.80		45.52		45.24
5					31.70		31.01		30.73	
4				21.22		20.19		19.15		18.88
3			13.77		12.64		11.32		9.64	
2		8.69		7.67		6.44		4.86		2.00
1			4.52		3.57		2.43		0.91	
0				1.94		1.20		0.41		0.00
-1					0.59		0.19		0.00	
-2						0.08		0.00		0.00
-3							0.00		0.00	
-4								0.00		0.00
-5									0.00	
-6										0.00
-7										
-8										
-9										
-10										

图 1-21　期数可变 CRR 模型的期权价格树

例 1-9 计算的是期数可变二叉树期权定价模型下应用动态复制定价方法得到的欧式看涨期权的价格，而对于动态复制定价方法下欧式看跌期权来说，做法实际上是例 1-2 与例 1-9 的结合，只需要将支付形式作出改变即可。同样，利用期数可变二叉树期权定价模型计算其他类型的期权价格时，也是例 1-9 与之前对应例的结合，做法相对简单，这里不再赘述。

	A	B	C
1			
2		基础数据	
3			动态复制
4		S	30.00
5		K	28.00
6		T	1.00
7		Rf	0.04
8		u	1.25
9		d	0.80
10		n	8.00
11		结果	
12		案例9	8.69

图 1-22　期数可变模型的数据输入与输出界面示意

1.2.5　实验结果

本实验基于 Excel 采用动态复制定价方法和风险中性定价方法计算了欧式看涨期权、欧式看跌期权和美式看涨期权、美式看跌期权的价格，通过实验发现无论是动态复制定价方法还是风险中性定价方法借助 Excel 建模都可以快速计算出期权的价格。从实验过程中可以发现与动态复制定价方法相比，风险中性定价方法更为简洁，有利于提高计算速度，降低建模难度。而期数可变 CRR 模型的构建，提升了模型的扩展性。

1.3　基于 MATLAB 的二叉树期权定价的数值实验

1.3.1　实验目的

MATLAB 是主流科学计算软件，有强大的金融类工具箱支撑，近年来在财经领域的占有率越来越高，特别适合有编程背景，现在或将来从事金融工程、量化投资等领域的读者学习掌握。要求学生使用 MATLAB 运用动态复制定价方法和风险中性定价方法分别计算 CRR 模型下的欧式看涨期权、欧式看跌期权、美式看涨期权、美式看跌期权。要求学生使用 MATLAB 计算 BSM 模型下的欧式看涨期权、欧式看跌期权并与 CRR 模型计算结果进行对比分析，并通过图形展示 CRR 模型的收敛性。

1.3.2 编程准备

这部分我们介绍 CRR 模型的 MATLAB 编程实现，与 Excel 相似，其建模重点是利用二叉树本身的性质，构造出股票价格路径的变化，也就是我们所说的“树”。在 MATLAB 中我们将股票价格在矩阵中合理布局，并利用循环嵌套，快速、准确地计算出期权的价格。为了增加程序的拓展性，并有利于图形用户界面（Graphical User Interfaces，GUI）的设计，我们的 MATLAB 程序都将以函数的形式构造。

1.3.3 实验数据

由于二叉树期权定价计算属于数值实验，因此实验所需数据是通过二叉树也就是二项分布的演变形成计算所需的标的资产价格树的，实验所需的其他参数是考虑该参数的真实取值区间并考虑到计算的便利而外生给定的。所需的主要数据包括股票的初始价格、期权的执行价格、无风险利率、股票的上升与下降幅度、股票价格的波动率、期权的有效期，以及二叉树期权定价模型的期数等。

1.3.4 实验过程

1. 风险中性定价方法

与前面的参数保持一致，期初股价 S_0，执行价格 K，无风险利率 r，期权有效期 T，股价上升 u，下降 d（CRR 模型中，$u=\mathrm{e}^{\sigma\sqrt{\Delta t}}, d=\mathrm{e}^{-\sigma\sqrt{\Delta t}}, ud=1$），二叉树的期数 N，美式或欧式 AoE，看涨看跌 type 这几个变量作为输入变量，期权价格作为输出变量，并将函数命名为 Optionprice。其代码如下：

```
function [Price] =Optionprice(S0,K,r,T,u,N,AoE,type)
%OptCRR - Price an option via Cox-Ross-Rubenstein tree
% Inputs:
%
%   S0      Initial asset price
%   K       Strike Price
%   r       Interest rate
%   T       Time to maturity of option
%   u       a constent number of the increasing stock per t
%   N       Number of points in time grid to use
%   AoE     type of the option true for a European and false for an
%           American option
%   type    True (default) for a call, false for a put
```

注意，MATLAB 中的百分号起到注释作用，一个完整编排的代码中，清晰的注释是必不可少的，不过在以后的程序中，为了避免重复，我们会把重复的注释删去，只保留关于新引入变量必需的注释。编程时，应当把应用到的变量名称一一加注，这种良好的编程习惯有利于提高效率。

下面予以详细介绍模型建立的三个步骤。

（1）将除输入变量外的其他变量及模型中多次利用到的中间计算式计算出来。由输入变

量并结合 $ud=1$，我们可以很容易计算出股票价格每次下跌百分比 d，以及风险中性概率 p。

（2）建立股票价格的二叉树并获得终端支付（payoff），由对于终端时刻 T 的股票价格情况，其价格按照二叉树期权定价模型又有如下规律：股票最高价格是其初始价格上涨 N 次的价格，随后价格递减并且按照 d 的二次幂成等比数列，这样我们可以很方便地在一个一维行向量 S 中建立股票终端价格的分布，然后根据期权的性质确定看涨看跌期权的终端支付，这里用一个一维向量 P 储存。下面显示其代码实现过程：

```
%   type    True (default) for a call, false for a put

dt = T/N;

d = 1/u;
a = exp(r*dt);
p = (a-d)/(u-d);

% Create final Returns on the tree
S{N+1} = S0*u^N*d.^(0:2:2*N);
if type
    %for call option type =1
    P{N+1} = max(S{N+1}-K,0);
else
    P{N+1} = max(K-S{N+1},0);
end% Now move back through time and calculate the expected return at previous
% nodes on the tree.
```

（3）根据期权终端支付计算期权价格，这是期权定价计算最重要的一步。这里有两种方案：其一，构建一个二维矩阵将期权每期价格列示；其二，在同一个一维向量中，由后向前根据期权二叉树期数倒推，并且不断更改它的数值，直至计算完成。考虑到我们的目的只是计算出期权价格，并且如果采用第一种方法，其中间变量难免也是二维矩阵，对于初次接触 MATLAB 矩阵运算的读者可能理解起来有些吃力，这里我们将采用第二种方法演示，这种方法也在一定程度上减轻了计算压力。对中间变量比较关注或者对编程有兴趣的读者可以尝试采用第一种方法进行编程。

关于期权价格具体的求解，我们是通过两个 for 循环实现的，外部的 for 循环主要起到控制计算期数的作用，同时对内部 for 循环加以限制，内部 for 循环起到计算各节点股票价格和期权价格的作用。对于有一定 C（C++）语言基础的读者来说，这样的循环应该不难。程序中，我们用 V 实现显示每期的节点各股票价格，用 E 显示每期各节点的期权价格，通过中间变量 I 和 P 来实现变量的比较和储存。当然，这些中间计算过程变量的设计主要是针对美式期权灵活的执行日期考虑的，在编程时，思路应该从欧式期权的角度开始，然后逐步扩大到美式期权。

由上述计算方案可知，期权价格向量 P 第一期的第一个数值就是我们所求解的期权的二叉树期权定价模型下的理论价格，即输出变量。二叉树期权定价模型在 MATLAB 中的实现比在 Excel 中实现还要简单，因为这里我们只要求计算出结果，对变量计算过程要求不高。具体代码如下：

```
% nodes on the tree.
```

```
for ii = N:-1:1
    Q = zeros(1,ii);
    V = zeros(1,ii);
    for jj = 1:ii
        V(jj) = S0*u^(ii-1)*d^(2*(jj-1));
        % Expected value of option due if we continue to hold
        E = p*P{ii+1}(jj)/a+(1-p)*P{ii+1}(jj+1)/a;
        % Value of early exercise
        if type
            % call option
            I = max(V(jj)-K,0)  ;
        else
            I = max(K-V(jj),0);
        end

        if AoE  %european option
            Q(jj)=E;
        else
            Q(jj) = max(E,I);
        end
    end

    P{ii} = Q;
end

Price = P{1};
```

我们将写好的函数拿到命令窗口下执行，当输入参数正确的时候，点击回车键运行，期权价格将很快显示在命令行窗口中（如图 1-23 所示）。如果不加以特殊说明，以后的期权价格计算的演示我们都会以 S_0=30，K=28，r=0.04，T=1，u=1.25（$\sigma=0.7056$），N=10，AoE=1，type=1 为例。这里需要指出的是，由于我们事先给出了 u、d 的值，而二叉树期权定价模型的期数为 N，T=1，由此推算出来 $\sigma=0.705\,6$。如果二叉树期权定价模型的期数和期权有效期发生变化，波动率 σ 也会随之变化。当然如果发生变化我们会特别指出。也许有的读者会觉得将二叉树期权定价模型作为 BSM 连续时间模型的离散版本，逻辑上应该是由波动率 σ 推算出 u、d。事实也应如此。只是若事先给定波动率 σ，如果不是很凑巧，推算出的 u、d 的数值会比较复杂，从而不利于二叉树期权定价模型的计算。特别是在进行手工计算时，计算过程会比较烦琐。当运用软件进行编程时，也是相对比较简单的 u、d 取值有利于股票价格路径变化的构造，同时也有利于直观理解。

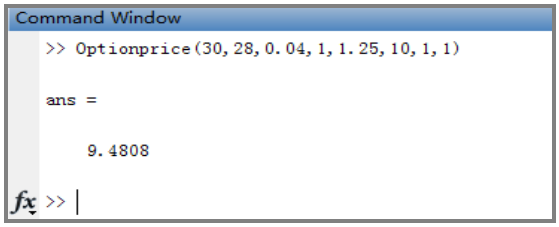


图 1-23　运行函数 Optionprcice 的命令行窗口

建模小技巧：

①当你需要重复运行某一条指令时，可以找到并双击历史命令窗口（Command History）中的该条指令；

②当你需要简单更改上一条指令中的某一个或几个参数时，只需按键盘方向键的向上键↑，调出上一条指令，并在此基础上进行修改。

掌握了 MATLAB 程序的技巧，将会给程序运行与修改带来极大便利，读者应该在学习中注意积累和应用。

下面我们将分步研究代码运行的过程，这里我们取 S_0=30，K=28，r=0.04，T=1，u=1.25，N=10，AoE=1，type=1；并删去函数名称这行代码使代码能够在命令窗口中执行，这时我们将计算过程中第一个 for 循环中“for ii = N：-1：1”变为“for ii = N”，使之固定为 N，此时虽然不能计算出期权价格，但却能够帮助我们分析其运算过程。

此时我们关注的是工作空间（Workspace）窗口，如图 1-24 所示。

双击变量名称能够显示出变量矩阵的具体形式及数值。例如，关注倒数第二期股票价格变化时，双击变量 V，将得到如图 1-25 所示的结果。

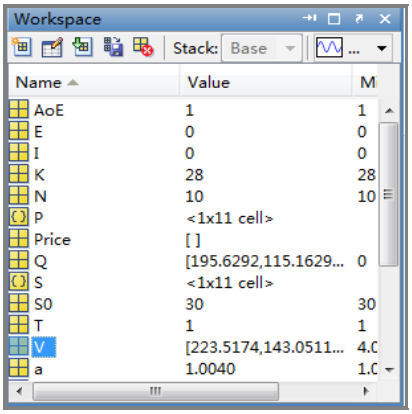


图 1-24　工作空间窗口

V <1x10 double>

	1	2	3	4	5	6	7	8	9	10
1	223.5174	143.0511	91.5527	58.5938	37.5000	24.0000	15.3600	9.8304	6.2915	4.0265

图 1-25　变量 V 矩阵的形式和数值——倒数第二期股票价格变化

如果更改最外层 for 循环的数值，就能够得到类似 Excel 中的一个树状图，倒数第三期股票价格变化如图 1-26 所示。

V <1x9 double>

	1	2	3	4	5	6	7	8	9	10
1	178.8139	114.4409	73.2422	46.8750	30.0000	19.2000	12.2880	7.8643	5.0332	

图 1-26　变量 V 矩阵的形式和数值——倒数第三期股票价格变化

这里共有 9 个数，依此类推，每向前一期价格减少一个股价。

前面提到过两种计算期权价格的方法，我们采用的方法是计算快捷、占用空间较少的一维向量法。然而在这种情况下却无法一次形成一个完整的“树”，当代码有误时，采用一维向量法很难找到出错的位置。在用二维矩阵法计算期权价格时，则很容易形成一棵“二叉树”，并比较容易找到出错的位置，因此这两种方法各有利弊。

2. 动态复制定价方法

运用风险中性定价方法给期权定价，虽然优于动态复制方法，但这两种方法都很重要，都应熟练掌握。下面我们针对一个二维矩阵承装运算过程，采用动态复制定价方法进行算法实现，读者可以比照两种方法，在编程中交叉利用，在熟悉 MATLAB 运算过程与原理的同时，熟练掌握这两种定价方法。

变量选择和中间变量的计算，不再做具体说明。这里将函数命名为 Optionprice2，代码如下：

```
function [Price] =Optionprice2(S0,K,r,T,u,N,AoE,type)

dt = T/N;
d = 1/u;
```

中间的过程与在 Excel 中的实现是一致的，我们用 *S* 矩阵承装股票价格走势，用 *P* 矩阵承装每期的期权价格，*V* 矩阵是用于对比的每期支付（payoff）矩阵。它们的输入过程比较简单。唯一需要注意的是 j 行 i 列对应的具体数值，尤其是 *S* 矩阵的设计，即 S(j,i)= S0*(u^(i-1))*d^(2*(j-1))需要特别注意。

```
S=zeros(N+1,N+1);
P=zeros(N+1,N+1);
for i=N+1:-1:1
    for j=1:1:i
        S(j,i)=S0*(u^(i-1))*d^(2*(j-1));
    end
end
if type
   V=S-K;
   P(:,N+1)=max(S(:,N+1)-K,0);
else
   V=K-S;
   P(:,N+1)=max(K-S(:,N+1),0);
end
```

P 矩阵的填充是关键的一步，需要事先计算出动态复制的系数公式，然后按照系数公式编写运算程序，这部分代码在程序设计方面并没有太大难度。

```
if AoE
  for i=N:-1:1
    for j=1:1:i  %下面公式较长在排版中出现分行情况，在源程序中应该在同一行
      P(j,i)=(S(j,i)*(P(j,i+1)-P(j+1,i+1))/(S(j,i+1)-S(j+1,i+1)))+(S(j,i+
1)*P(j+1,i+1)-S(j+1,i+1)*P(j,i+1))*exp(-r*dt)/(S(j,i+1)-S(j+1,i+1));
    end
  end
else
    for i=N:-1:1
    for j=1:1:i
      P(j,i)=max(V(j,i),(S(j,i)*(P(j,i+1)-P(j+1,i+1))/(S(j,i+1)-S(j+1,i+
1)))+(S(j,i+1)*P(j+1,i+1)-S(j+1,i+1)*P(j,i+1))*exp(-r*dt)/(S(j,i+1)-S(j+
1,i+1)));    %同上此处也应当在一行内输入
```

```
        end
        end
    end
    Price=P(1,1);
```

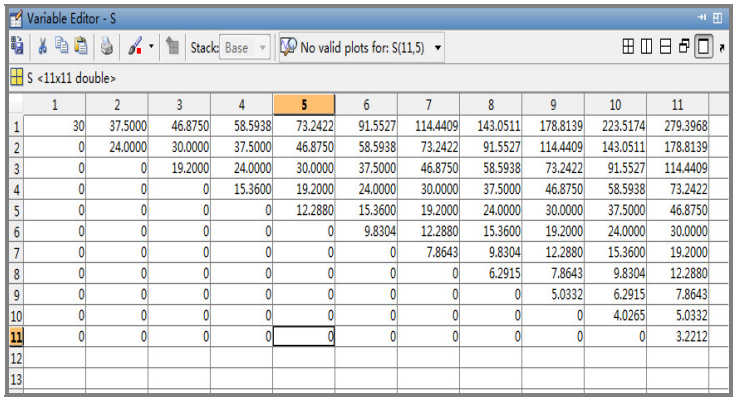
在后续的学习中，我们会发现一个函数可以有多个输出变量，这时我们便可以把风险中性定价方法和动态复制定价方法整合到一个函数中，形成如下形式的函数：

```
    function [Price1,Price2] =Optionprice(S0,K,r,T,u,N,AoE,type)
```

其函数内部编写与上面的函数编写类似，只是两种方法输出分别为 Price1 和 Price2，在运行函数时能够直接输出两个结果。由于这两种方法得出的结果完全一致，实际应用中我们没有必要将函数写成上面输出两个变量的形式，这里只出于理论的深化及 MATLAB 函数功能拓展在这里加以介绍。

接下来对这一函数计算的内部过程加以简单探究。赋予各变量数值，在命令窗口中运行，我们可以得到各个变量的数值，并且以树的形式出现。

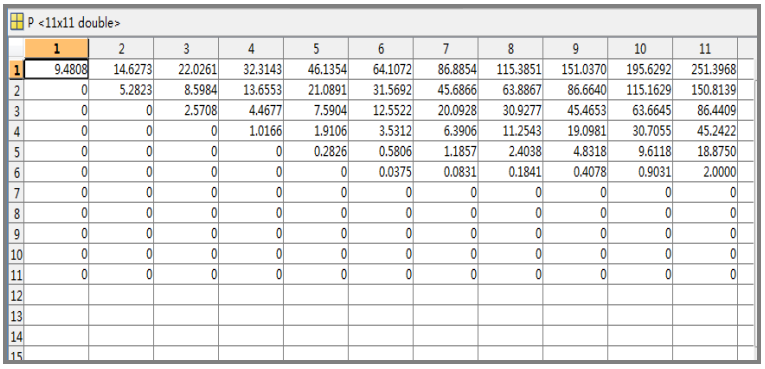
股票价格矩阵 S 的变化如图 1-27 所示。



	1	2	3	4	5	6	7	8	9	10	11
1	30	37.5000	46.8750	58.5938	73.2422	91.5527	114.4409	143.0511	178.8139	223.5174	279.3968
2	0	24.0000	30.0000	37.5000	46.8750	58.5938	73.2422	91.5527	114.4409	143.0511	178.8139
3	0	0	19.2000	24.0000	30.0000	37.5000	46.8750	58.5938	73.2422	91.5527	114.4409
4	0	0	0	15.3600	19.2000	24.0000	30.0000	37.5000	46.8750	58.5938	73.2422
5	0	0	0	0	12.2880	15.3600	19.2000	24.0000	30.0000	37.5000	46.8750
6	0	0	0	0	0	9.8304	12.2880	15.3600	19.2000	24.0000	30.0000
7	0	0	0	0	0	0	7.8643	9.8304	12.2880	15.3600	19.2000
8	0	0	0	0	0	0	0	6.2915	7.8643	9.8304	12.2880
9	0	0	0	0	0	0	0	0	5.0332	6.2915	7.8643
10	0	0	0	0	0	0	0	0	0	4.0265	5.0332
11	0	0	0	0	0	0	0	0	0	0	3.2212
12											
13											

图 1-27　股票价格矩阵

期权价格矩阵 P 如图 1-28 所示。



	1	2	3	4	5	6	7	8	9	10	11
1	9.4808	14.6273	22.0261	32.3143	46.1354	64.1072	86.8854	115.3851	151.0370	195.6292	251.3968
2	0	5.2823	8.5984	13.6553	21.0891	31.5692	45.6866	63.8867	86.6640	115.1629	150.8139
3	0	0	2.5708	4.4677	7.5904	12.5522	20.0928	30.9277	45.4653	63.6645	86.4409
4	0	0	0	1.0166	1.9106	3.5312	6.3906	11.2543	19.0981	30.7055	45.2422
5	0	0	0	0	0.2826	0.5806	1.1857	2.4038	4.8318	9.6118	18.8750
6	0	0	0	0	0	0.0375	0.0831	0.1841	0.4078	0.9031	2.0000
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0
12											
13											
14											
15											

图 1-28　期权价格矩阵

我们还可以用 MATLAB 金融工具包中的 binprice 函数，给出这一问题的具体结果。binprice 函数是 MATLAB 自带的期权计算函数，它本身就是通过二叉树期权定价模型计算期权价格的。关于 binprice 函数的具体内容，可以通过 MATLAB 帮助或者在 MATLAB 安装文件夹 C:\Program Files\MATLAB\R2009b\toolbox\finance\finance 下找到金融工具包，双击 binprice.m 查看函数具体内容，事实上这一函数也是按照风险中性定价方法进行定价的。在以后我们应用到 MATLAB 工具箱中函数时，都可以通过这种方式查看其内部代码以了解其具体原理。

应用 binprice 函数，我们首先应该计算出期权波动率为 0.705 64，代入函数公式中：

```
[AssetPrice,OptionValue]=binprice(30,28,0.04,1,0.1,0.705 641 867,1)
```

单击运行，就会在命令窗口得到股票价格的二叉树及期权价格的二叉树。这里我们截取 WorkSpace 窗口两个输出变量的结果，如图 1-29 和图 1-30 所示。

AssetPrice <11x11 double>

	1	2	3	4	5	6	7	8	9	10	11
1	30	37.5000	46.8750	58.5937	73.2422	91.5527	114.4409	143.0511	178.8139	223.5174	279.3968
2	0	24.0000	30.0000	37.5000	46.8750	58.5937	73.2422	91.5527	114.4409	143.0511	178.8139
3	0	0	19.2000	24.0000	30.0000	37.5000	46.8750	58.5937	73.2422	91.5527	114.4409
4	0	0	0	15.3600	19.2000	24.0000	30	37.5000	46.8750	58.5937	73.2422
5	0	0	0	0	12.2880	15.3600	19.2000	24.0000	30.0000	37.5000	46.8750
6	0	0	0	0	0	9.8304	12.2880	15.3600	19.2000	24.0000	30
7	0	0	0	0	0	0	7.8643	9.8304	12.2880	15.3600	19.2000
8	0	0	0	0	0	0	0	6.2915	7.8643	9.8304	12.2880
9	0	0	0	0	0	0	0	0	5.0332	6.2915	7.8643
10	0	0	0	0	0	0	0	0	0	4.0265	5.0332
11	0	0	0	0	0	0	0	0	0	0	3.2212
12											

图 1-29　股票价格矩阵——MATLAB 自带函数 binprice 运行生成

OptionValue <11x11 double>

	1	2	3	4	5	6	7	8	9	10	11
1	9.4808	14.6273	22.0261	32.3143	46.1354	64.1072	86.8854	115.3851	151.0370	195.6292	251.3968
2	0	5.2823	8.5984	13.6553	21.0891	31.5692	45.6866	63.8867	86.6640	115.1629	150.8139
3	0	0	2.5708	4.4677	7.5904	12.5522	20.0928	30.9277	45.4653	63.6645	86.4409
4	0	0	0	1.0166	1.9106	3.5312	6.3906	11.2543	19.0981	30.7055	45.2422
5	0	0	0	0	0.2826	0.5806	1.1857	2.4038	4.8318	9.6118	18.8750
6	0	0	0	0	0	0.0375	0.0831	0.1841	0.4078	0.9031	2
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0
12											

图 1-30　期权价格矩阵——MATLAB 自带函数 binprice 运行生成

3. 二叉树期权定价模型的收敛性示意

当我们应用二叉树期权定价模型时，会发现二叉树期权定价模型计算出来的期权价格

是与计算期数或者时间间隔有关的，这是由于二叉树期权定价模型模拟的股票价格，理论上只有当 dt 趋近于 0 时，股票价格才服从对数正态分布。因此，理论上当 dt 趋近于 0，即计算期数趋近于无穷大时，二叉树期权定价模型计算出来的价格与 BSM 公式计算出来的价格是一致的。下面我们将通过绘图具体感知这一收敛过程，具体如图 1-31 所示。我们可以利用已经建好的二叉树程序或者直接调用 MATLAB 金融工具包里的 binprice 函数。这里我们采用后者，并直接在命令窗口运行如下程序。代码如下：

```
%Example: Note requires the financial toolbox function blsprice
%and blnprice
S=30; K=28; sigma=0.705642; r=0.04; T=1; tic
counter=0;
for k=5:1:200
counter=counter+1;
a(counter)=binprice(S,K,r,T,T/k,0.705641867,1);
b(counter)=blsprice(S,K,r,T,sigma);
end
k=5:1:200;
plot(k,a,'b');
hold on
plot (k,b,':r');
hold off
title('Option price as a function of the number of steps');
ylabel('Option price');
xlabel('Number of steps, n');   toc
```

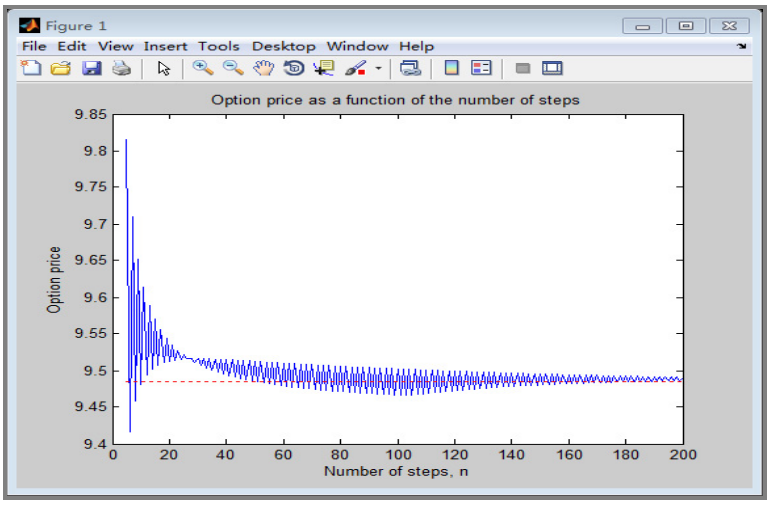


图 1-31　期权价格随着二叉树期权定价模型期数增加收敛到解析解示意

从图 1-31 中可以看出，上面的理论分析是正确的，当 n 趋近于 200 时，期权价格逐渐接近 BSM 公式计算出来的价格，并且波动性逐渐减弱。我们引入计时函数 tic toc（在函数起始位置添加 tic，在末尾添加 toc，都不用加分号），可以显示执行 196 个计算并画图累计用时 2.932 360 秒，平均每个计算不到 0.015 秒。而当 n=200 时，用时 0.39 秒，也在可接受范围之内。因而，在利用二叉树期权定价模型计算期权价格时，可以让 n 尽可能大，以

便使结果尽可能精确。关于二叉树期权定价模型的收敛性，有很多种观点。通过增大计算期数的方法属于其中一种，除此之外还可以利用三叉树及构造新型二叉树等方法。然而这些方法各有利弊，大多是通过提高计算复杂程度来改进计算结果收敛速度的。有兴趣的读者可以通过自主学习去了解有关其他新型二叉树期权定价模型和三叉树期权定价模型的知识。

1.3.5 实验结果

通过实验可以发现，无论是动态复制定价方法，还是风险中性定价方法，都可借助 MATLAB 建模快速计算出期权价格。从实验过程中还可以发现，与动态复制定价方法相比，风险中性定价方法更为简洁，更有利于提高计算速度，降低建模难度。与 Excel 实现手段相比，MATLAB 更适合计算期数较多时的二叉树期权定价模型，而 Excel 受到计算能力的限制一般只能计算 30 期以内的二叉树期权定价模型。由于 MATLAB 计算二叉树期权定价模型时，二叉树期权定价模型的期数本身就是作为定价函数的输入参数之一，因此基于 MATLAB 的二叉树期权定价模型自然可以处理二叉树期权定价期数可变的情况，因此没有必要单独讨论期数可变的情况。就二叉树期权定价模型的收敛性进行分析，实验结果表明当 n 趋近于 200 时，期权价格逐渐接近 BSM 公式计算出来的价格并且波动性逐渐减弱。因此，在一般情况下，采用二叉树期权定价模型计算期权价格时，n 可以选择在 200 附近，这种选择已经可以保证价格计算的精确度，并且计算成本也不高。

1.4 基于 Python 的二叉树期权定价的数值实验

1.4.1 实验目的

Python 诞生于 1990 年，其具有简洁性、易读性及可扩展性强的特点，能够非常高效地分析和处理数据，当前已被广泛应用于机器学习、人工智能系统等各种技术领域中，并被广泛应用于金融领域。读者应能够熟练使用 Python，运用动态复制定价方法和风险中性定价方法分别计算二叉树期权定价模型框架下的欧式看涨期权、欧式看跌期权、美式看涨期权、美式看跌期权。

1.4.2 编程准备

这部分我们介绍二叉树期权定价模型的 Python 编程实现，与 Excel 和 MATLAB 相似，其建模重点是利用二叉树本身的性质，构造出股票价格路径的变化，也就是我们所说的“树”。在 Python 中我们将股票价格合理布局在矩阵中，并利用循环嵌套，快速、准确地计算出期权的价格。

1.4.3 实验数据

由于二叉树期权定价计算属于数值实验，因此实验所需数据是通过二叉树也就是二项

分布的演变形成计算所需的标的资产价格树的，实验所需的其他参数是考虑该参数的真实取值区间并考虑到计算的便利而外生给定的。所需的主要数据包括股票的初始价格、期权的执行价格、无风险利率、股票的上升与下降幅度、股票价格的波动率、期权的有效期以及二叉树期权定价模型的期数等。

1.4.4 实验过程

1. 风险中性定价方法

与前面的参数保持一致，期初股价 S_0，执行价格 K，无风险利率 r，期权有效期 T，股价上升 u，下降 d 二叉树的期数 N、美式或欧式期权 AoE、看涨看跌 type 等几个变量作为输入变量，期权价格作为输出变量，并将函数命名为 price_risk_neutral。通过 MATLAB 部分的内容，在计算期权价格的过程中有一维向量法和二维矩阵法，由于二维矩阵法“二叉树”更为直观，Python 代码的展示将全部基于二维矩阵法。此外，Python 代码的逻辑与 MATLAB 高度相似，这里不再赘述，仅给出 Python 语言本身的一些注意事项。

首先，将使用到的包进行 import 导入，并且为其起一个别名以方便后续使用。在本例中需要导入 NumPy 包，并将其记作“np”，这是 Python 中科学计算的基本包，Python 数据分析的绝大多数内容都是基于 NumPy 和在 NumPy 基础之上构建的库。它提供的功能包括：快速高效的多维数组对象 ndarray，用数组或数组之间的数学运算来执行元素计算的函数，用于读取和写入基于数组的数据集到磁盘的工具，线性代数运算、傅里叶变换、随机数生成等。

其次，Python 中用#开头进行单行注释，进行多行注释时通常用三个单引号或三个双引号将注释部分包裹起来。在 def 定义函数后，通常用多行注释的方式为函数编写说明文档，用来解释函数的主要功能、参数和返回值的数据类型及含义等，增强函数的可读性和可用性。

最后，在代码编写时可以用“if __name__ == '__main__': ”将测试代码分离出来。一个 Python 文件通常有两种使用方法，一种是直接作为脚本执行，另一种是被其他脚本 import 调用执行。“if __name__ == '__main__': ”这段语句起到的作用就是控制在这两种情况下执行代码的过程，在添加该语句的情况下，该语句下的代码只有在文件作为脚本直接执行时才会被执行，被其他脚本 import 调用时不会被执行。代码如下：

```
import numpy as np

def price_risk_neutral(S0, K, r, T, u, N, AoE, type):
    '''
    功能：利用风险中性定价方法计算期权价格
    参数：
        S0:股票的期初价格
        K :期权的执行价格
        r :无风险利率
        T : 有效期
```

```
        u : 股价每次上涨幅度
        N :判断点个数
        AoE:1 表示欧式期权，0 表示美式期权
        type:1 表示看涨，0 表示看跌
    '''
    d = 1 / u
    dt = T / N
    a = np.exp(r*dt)

    # 风险中性概率
    p = (a-d) / (u-d)
    # S 记录每期各节点的股票价格
    S = np.zeros([N+1, N+1])
    # P 记录每期各节点的期权价格
    P = np.zeros([N+1, N+1])

    # Create terminal payoff on the tree
    S[:, N] = S0 * np.power(u, N) * np.power(d, range(0, 2*N+1, 2))
    if type:
        P[:, N] = np.maximum(S[:, N]-K, 0)
    else:
        P[:, N] = np.maximum(K-S[:, N], 0)

    # nodes on the tree
    for j in range(N, 0, -1):
        for i in range(0, j):
            S[i][j-1] = S0 * np.power(u, j-1) * np.power(d, 2*i)
            # E 记录每期各节点的期权价值
            E = (p*P[i][j] + (1-p)*P[i+1][j]) / a
            if type:
                I = max(S[i][j-1]-K, 0)
            else:
                I = max(K-S[i][j-1], 0)

            if AoE:
                P[i][j-1] = E
            else:
                P[i][j-1] = max(E, I)
    print("the price of the option is %.4f" % P[0, 0])
return S, P

if __name__ == '__main__':
    S0 = 30
    K = 28
    r = 0.04
    T = 1
    u = 1.25
    N = 10
    AoE = 1
    type = 1
```

```
    price_risk_neutral(S0, K, r, T, u, N, AoE, type)
```

运行结果如下：

```
the price of the option is 9.4808
```

2. 动态复制定价方法

介绍完运用风险中性定价方法给期权定价，接下来将对动态复制定价方法进行展示，此处我们将函数命名为 price_dynamic_replication，代码如下：

```
import numpy as np

def price_dynamic_replication(S0, K, r, T, u, N, AoE, type):
    '''
    功能：利用动态复制定价方法计算期权价格
    参数：
        S0:股票的期初价格
        K :期权的执行价格
        r :无风险利率
        T : 有效期
        u : 股价每次上涨幅度
        N :判断点个数
        AoE:1 表示欧式期权，0 表示美式期权
        type:1 表示看涨，0 表示看跌
    '''
    d = 1 / u
    dt = T / N
    # S 记录每期各节点的股票价格
    S = np.zeros([N+1, N+1])
    # P 记录每期各节点的期权价格
    P = np.zeros([N+1, N+1])

    # Create terminal payoff on the tree
    for i in range(0, N+1):
        for j in range(i, N+1):
            S[i, j] = S0*pow(u, j-i)*pow(d, i)
    # V 记录每期各节点的支付
    if type:
        V = S-K
        P[:, N] = np.maximum(S[:, N]-K, 0)
    else:
        V = K-S
        P[:, N] = np.maximum(K-S[:, N], 0)

    if AoE:
        # j 控制列，i 控制行
        for j in range(N-1, -1, -1):
            for i in range(0, j+1):
                delta = (P[i, j+1]-P[i+1, j+1]) / (S[i, j+1]-S[i+1, j+1])
```

```
                L = (u*P[i+1, j+1] - d*P[i, j+1]) / np.exp(r*dt) / (u-d)
                P[i, j] = delta*S[i, j] + L
    else:
        for j in range(N-1, -1, -1):
            for i in range(0, j+1):
                delta = (P[i, j+1]-P[i+1, j+1]) / (S[i, j+1]-S[i+1, j+1])
                L = (u*P[i+1, j+1] - d*P[i, j+1]) / np.exp(r*dt) / (u-d)
                P[i, j] = max(V[i, j], delta*S[i, j] + L)
    print("the price of the option is %.4f" % P[0, 0])
return S, P

if __name__ == '__main__':
    S0 = 30
    K = 28
    r = 0.04
    T = 1
    u = 1.25
    N = 10
    AoE = 1
    type = 1
    price_dynamic_replication(S0, K, r, T, u, N, AoE, type)
```

运行结果如下：

```
the price of the option is 9.4808
```

我们可以通过 Data Viewer（这是 VScode 编辑器 debug 过程中提供的变量查看工具，对于其他编辑器查看变量具体数据的方法，读者可以自行探索）查看记录股票价格节点数据的 S、记录期权价格节点数据的 P 等变量的变化，如图 1-32 和图 1-33 所示。

	0	1	2	3	4	5	6	7	8	9	10
0	30	37.5	46.875	58.59375	73.24218…	91.55273…	114.4409…	143.0511…	178.8139…	223.5174…	279.3967…
1	0	24	30	37.5	46.875	58.59375	73.24218…	91.55273…	114.4409…	143.0511…	178.8139…
2	0	0	19.2	24	30	37.5	46.875	58.59375	73.24218…	91.55273…	114.4409…
3	0	0	0	15.36	19.2	24	30	37.5	46.875	58.59375	73.24218…
4	0	0	0	0	12.288	15.36	19.2	24	30	37.5	46.875
5	0	0	0	0	0	9.8304	12.288	15.36	19.2	24	30
6	0	0	0	0	0	0	7.86432	9.8304	12.288	15.36	19.2
7	0	0	0	0	0	0	0	6.291456	7.86432	9.8304	12.288
8	0	0	0	0	0	0	0	0	5.0331648	6.291456	7.86432
9	0	0	0	0	0	0	0	0	0	4.026531…	5.0331648
10	0	0	0	0	0	0	0	0	0	0	3.221225…

图 1-32　股票价格矩阵

	0	1	2	3	4	5	6	7	8	9	10
0	9.480840…	14.62726…	22.02612…	32.31434…	46.13543…	64.10717…	86.88535…	115.3851…	151.0370…	195.6291…	251.3967…
1	0	5.282280…	8.598429…	13.65531…	21.08905…	31.56924…	45.68662…	63.88672…	86.66402…	115.1629…	150.8139…
2	0	0	2.570834…	4.467654…	7.590428…	12.55217…	20.09277…	30.92774…	45.46529…	63.66451…	86.44091…
3	0	0	0	1.016598…	1.910607…	3.531165…	6.390583…	11.25434…	19.09810…	30.70552…	45.24218…
4	0	0	0	0	0.282624…	0.580642…	1.185660…	2.403785…	4.831812…	9.611776…	18.875
5	0	0	0	0	0	0.037542…	0.083141…	0.184129…	0.407779…	0.903082…	2
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

图 1-33　期权价格矩阵

1.4.5 实验结果

本实验，基于 Python 语言，采用动态复制定价方法和风险中性定价方法计算了欧式看涨期权、欧式看跌期权、美式看涨期权、美式看跌期权的价格。通过实验发现，无论是动态复制定价方法还是风险中性定价方法，借助 Python 编程都可以快速计算出期权的价格。可以发现，通过 Python 语言编程计算得到的期权价格、股票价格矩阵、期权价格矩阵等和利用 MATLAB 得到的结果是一致的，由此可见编程语言仅仅是工具。我们可以选择熟悉的工具解决问题，但通过多种编程语言得到的结果之间可以相互比照、相互印证，更能够感受到不同编程语言的特点和优势。

1.5 基于 C++与 Excel-Addin 的二叉树期权定价的数值实验

1.5.1 实验目的

本章主要研究二叉树期权定价模型的算法实现，C++与 Excel-Addin 结合使用，因为有强大的函数库的支持，不但保证了计算的速度，而且可以清晰地展示结果，对非编程人员比较友好，因此成为金融、量化投资领域的主流编程语言。本实验使用 Excel-Addin 在 Excel 中实现欧式期权与美式期权的二叉树期权定价模型算法，并与解析解进行对比。由于二叉树期权定价模型算法在数值上可以看作特殊的显性差分算法，因此从后面关于有限差分收敛性的理论分析可知，显性差分方法的均方误差应以 $O(h^2+k)$的速度进行收敛，则当 $k=h$ 时，显性差分方法的均方误差应收敛于 $O(h)$。若对两边取对数，则均方误差的对数应为 $\log(h)$ 的线性函数，并且其斜率为 1。由此可知，二叉树期权定价模型算法的均方误差的对数也应约为节点数的对数 $\log(h)$的线性函数，并且其斜率为 1。实验中将使用 Excel-Addin 对二叉树期权定价模型算法进行误差分析，来观察二叉树期权定价模型算法的渐近收敛性。

1.5.2 编程准备

我们着重介绍如何在 Windows 环境下通过 Microsoft Visual Studio 实现一个最简单的 Excel-Addin——在 Excel 中插入一个欧式期权定价函数。在本书的编写过程中，我们使用的是 Microsoft Visual Studio Express 10.0 版本及 Microsoft Office 2007 套装系列。使用 C++编译 Excel-Addin 最简单、最直接的方法是使用微软为 C++编程人员免费提供的 Microsoft Office Excel 2007 XLL Software Development Kit (SDK)插件。该插件可在微软官方网站免费下载。

使用 C++与 Excel-Addin 的最大优点在于，编程人员一方面可以享受到 C++高效的计算速度，另一方面可以方便地使用 Excel 进行数据处理等操作，从而极大地提高了开发效率。

1. QuantLib 概述

QuantLib 是一个基于 C++的金融衍生品定价的免费开源库，在学术界与业界被广泛使

用。目前，QuantLib 涵盖了股票、利率、商品等衍生品的定价引擎，提供泛型 PDE 与蒙特卡罗的求解器，以及一些金融领域常用的数学工具，如优化函数、方程求解、插值等。

然而，QuantLib 也有一定的缺点。QuantLib 最大的缺点在于其过于复杂的系统架构（尤其对于编程初学者而言），使许多编程人员对其无从入手——即便要计算一个最简单的欧式期权，在 QuantLib 都要涉及至少十个对象。

要在 Visual Studio 下编译 QuantLib，首先需要编译 Boost。Boost 是一个开源的 Boost 库，是一个可移植、提供源代码的 C++库。作为标准库的后备，Boost 为 C++语言标准库提供了许多扩展。例如，uBLAS 为用户提供了一个矩阵运算的库，允许用户进行一些简单的矩阵运算以及求解矩阵等工作。

2. Boost 编译

编译 Boost，首先需要从其官方网站下载源代码，并解压放置于某个位置。与其他库不同的是，Boost 里的大部分函数是通过.hpp 的方式实现的，不需要编译即可直接使用。但 QuantLib 里的一部分函数是必须编译后才能使用的函数，因此我们还是要按照正常程序进行 Boost 编译。

Boost 编译的方式在其官方网站上已经有详细描述，在此不再赘述。但为了方便读者使用本书，我们将编译好的库文件（.lib）文件放于随书附送的代码里以备查用。

3. QuantLib 编译

有了 Boost 后，我们即可进行 QuantLib 编译。针对 Microsoft Visual Studio 10.0，QuantLib 已经具备解决方案的能力，用户只需打开 QuantLib 根目录下的 QuantLib_vc10 解决方案设置编译即可。

1.5.3 实验数据

由于二叉树期权定价计算属于数值实验，因此实验所需数据是通过二叉树也就是二项分布的演变形成计算所需的标的资产价格树的，实验所需的其他参数是考虑该参数的真实取值区间并考虑到计算的便利而外生给定的。所需的主要数据包括股票的初始价格、期权的执行价格、无风险利率、股票的上升与下降幅度、股票价格的波动率、期权的有效期，以及二叉树期权定价模型的期数等。

1.5.4 实验过程

由于前面我们已经基于 Excel 定价、MATLAB 和 Python 运用动态复制方法和风险中性定价方法计算了期权价格，为了节省篇幅，这里不再采用动态复制定价方法而仅使用风险中性定价方法。

1. 编写 C++代码

由于欧式期权与美式期权不涉及路径依赖的情况，为了节省内存，我们可以通过使用向量（而非矩阵或二维数组）进行储存，并在每一步都倒向更新向量。而美式期权与欧式期权的区别仅在于是否需要在倒向计算的每一步中寻求支付的最优值。

下面的代码是一个二叉树期权定价模型求解欧式期权的算法实现，可求解的期权包括欧式看涨期权、欧式看跌期权、数字看涨期权、数字看跌期权。

```
    using namespace QuantLib;

    DLLEXPORT double BinomialOptionPrice(char *optionType, double dSpot,
double dStrike,
        double dRate, double dDiv, double dVol, double dDaysToMaturity, int
iNodes)
    {
        try{
            std::vector<double> vS(iNodes,0.0);         // underlying price
            std::vector<double> vU(iNodes,0.0);        // option price
            double dYears = dDaysToMaturity / (double)TRADEDAY_COUNTER;
            double dt = dYears / (double)(iNodes-1);   // time step size
            double dUp = std::exp(dVol*std::sqrt(dt)); // up movement
            double dDown = 1/dUp; // down movement
            double dPu = (std::exp((dRate-dDiv)*dt) - dDown)/(dUp - dDown);
// probability of up movement
            double dPd = 1.0 - dPu; // probability of down movement

            /******** Make Option ********/
            for ( int j = 0; j < iNodes; j++)
            {
                  vS[j] = dSpot*std::pow(dUp,iNodes-1-j)*std::pow(dDown,j);
                  if ( !lstrcmpi(optionType,"Call") )
                        vU[j] = std::max(vS[j]-dStrike,0.0);
                  else if ( !lstrcmpi(optionType,"Put") )
                        vU[j] = std::max(dStrike-vS[j],0.0);
                  else if ( !lstrcmpi(optionType, "DigitCall")  ){
                        if ( vS[j] >= dStrike ) vU[j] = 1.0;}
                  else if ( !lstrcmpi(optionType, "DigitPut") ){
                        if ( vS[j] <= dStrike ) vU[j] = 1.0;}
                  else
                        QL_FAIL("failed");
            }

            /******** start solving Trees ********/
            for ( int i = iNodes-2; i >= 0; i--)
                  for ( int j = 0; j <= i; j++)
                        vU[j] = std::exp(-dRate*dt)*(dPu*vU[j] + dPd*vU[j+1]);

            return vU[0];
        }
        catch (const std::exception &e) {
              std::ostringstream err;
              err <<"Error loading: "<< e.what();
         }
    }
```

为了能在 Excel-Addin 中调用，我们还需要向 Excel 注册该函数，并在 DLLEXPORT int xlAutoOpen()中添加 registerBinomialOptionPrice(xDll)函数：

```
    void registerBinomialOptionPrice(XLOPER &xDll){
         // 参数量 = 10 + 函数参数量
         EXCEL(xlfRegister, 0, 18, &xDll,
             // function code name
             TempStrNoSize(" BinomialOptionPrice"),
             // parameter codes: First is OUTPUT, others are parms resp.
             TempStrNoSize(" BFBBBBBBJ"),
             // function display name
             TempStrNoSize(" BinomialOptionPrice"),
             // comma-delimited list of parameter names

             TempStrNoSize("\xFF""OptionType,Spot,Strike,Rate,Dividend,   Vo-
latility,DaysToMaturity,Nodes"),
             // function type (0 = hidden function, 1 = worksheet function, 2 =
command macro)
             TempStrNoSize(" 1"),
             // function category
             TempStrNoSize(" Pricing Engine - Vanilla"),
             // shortcut text (command macros only)
             TempStrNoSize(""),
             // path to help file
             TempStrNoSize(""),
             // function description
        TempStrNoSize("\xFF""Compute prices and greeks of vanilla options"),
             // parameter descriptions
             TempStrNoSize("\xFF""{Call,Put,DigitCall,DigitPut}.  Case  in-
sensitive"),
             TempStrNoSize("\xFF""Spot price of the underlying"),
             TempStrNoSize("\xFF""Strike of the option"),
             TempStrNoSize("\xFF""Risk-free Rate. In terms of percentage, e.g. 0.04
or 4%"),
             TempStrNoSize("\xFF""Dividend of the underlying. In terms of
percentage, e.g. 0.02 or 2%"),
             TempStrNoSize("\xFF""Annualized Volatility of the underlying.
In terms of percentage, e.g. 0.3 or 30%"),
             TempStrNoSize("\xFF""Days left to maturity. e.g. 60 = 60 days.
Day counter is 240/yr."),
             TempStrNoSize("\xFF""Number of nodes."));
    }
```

在二叉树期权定价模型的美式期权定价上，我们需要对每一步都进行最优化的判断处理。因此，在倒向计算的每一步中，我们都需要重新计算标的资产价格在各个节点的值，并由此计算支付。下面是二叉树期权定价模型在 C++中的代码实现：

```
    DLLEXPORT double BinomialAmericanOptionPrice(char *optionType, double
dSpot, double dStrike,
        double dRate, double dDiv, double dVol, double dDaysToMaturity, int
iNodes)
    {
        try{
            std::vector<double> vS(iNodes,0.0); // underlying price
```

```
            std::vector<double> vU(iNodes,0.0); // option price
            double dYears = dDaysToMaturity / (double)TRADEDAY_COUNTER;
            double dt = dYears / (double)(iNodes-1);    // time step size
            double dUp = std::exp(dVol*std::sqrt(dt)); // up movement
            double dDown = 1/dUp; // down movement
            double dPu = (std::exp((dRate-dDiv)*dt) - dDown)/(dUp - dDown);
// probability of up movement
            double dPd = 1.0 - dPu; // probability of down movement

            /******** Make Option ********/
            for ( int j = 0; j < iNodes; j++)
            {
               vS[j] = dSpot*std::pow(dUp,iNodes-1-j)*std::pow(dDown,j);
               if ( !lstrcmpi(optionType,"Call") )
                    vU[j] = std::max(vS[j]-dStrike,0.0);
               else if ( !lstrcmpi(optionType,"Put") )
                    vU[j] = std::max(dStrike-vS[j],0.0);
               else if ( !lstrcmpi(optionType, "DigitCall")  ){
                    if ( vS[j] >= dStrike ) vU[j] = 1.0;}
               else if ( !lstrcmpi(optionType, "DigitPut") ){
                    if ( vS[j] <= dStrike ) vU[j] = 1.0;}
               else
                    QL_FAIL("failed");
    }

            /******** start solving Trees ********/
            for ( int i = iNodes-2; i >= 0; i--)
            {
                  for ( int j = 0; j <= i; j++)
                  {
                       vS[j] = dSpot*std::pow(dUp,i-j)*std::pow(dDown,j);
                       vU[j] = std::exp(-dRate*dt)*(dPu*vU[j] + dPd*vU[j+1]);
                       if ( !lstrcmpi(optionType,"Call") )
                            vU[j] = std::max(vS[j] - dStrike, vU[j]);
                       else if ( !lstrcmpi(optionType,"Put") )
                            vU[j] = std::max(dStrike - vS[j], vU[j]);
                       else if ( !lstrcmpi(optionType, "DigitCall")){
                            if ( vS[j] >= dStrike ) vU[j] = 1.0;}
                       else if ( !lstrcmpi(optionType, "DigitPut") ){
                            if ( vS[j] <= dStrike ) vU[j] = 1.0;}
                       else
                            QL_FAIL("failed");
                  }
               }
               return vU[0];
          }
          catch (const std::exception &e) {
               std::ostringstream err;
               err <<"Error loading: "<< e.what();
          }
    }
```

同样地，为了能在 Excel-Addin 中调用，我们还需要向 Excel 注册该函数，并在

DLLEXPORT int xlAutoOpen()中添加 registerBinomialAmericanOptionPrice(xDll)函数：

```
    void registerBinomialAmericanOptionPrice(XLOPER &xDll){
        // 参数量 = 10 + 函数参数量
        EXCEL(xlfRegister, 0, 18, &xDll,
            // function code name
            TempStrNoSize(" BinomialAmericanOptionPrice"),
            // parameter codes: First is OUTPUT, others are parms resp.
            TempStrNoSize(" BFBBBBBBJ"),
            // function display name
            TempStrNoSize(" BinomialAmericanOptionPrice"),
            // comma-delimited list of parameter names

    TempStrNoSize("\xFF"                                          "Option-
Type,Spot,Strike,Rate,Dividend,Volatility,DaysToMaturity,Nodes"),
            // function type (0 = hidden function, 1 = worksheet function, 2
= command macro)
            TempStrNoSize(" 1"),
            // function category
            TempStrNoSize(" Pricing Engine - Vanilla"),
            // shortcut text (command macros only)
            TempStrNoSize(""),
            // path to help file
            TempStrNoSize(""),
            // function description
            TempStrNoSize("\xFF""Compute prices and greeks of vanilla op-
tions"),
            // parameter descriptions
            TempStrNoSize("\xFF""{Call,Put,DigitCall,DigitPut}. Case insen-
sitive"),
            TempStrNoSize("\xFF""Spot price of the underlying"),
            TempStrNoSize("\xFF""Strike of the option"),
            TempStrNoSize("\xFF""Risk-free Rate. In terms of percentage, e.g. 0.04
or 4%"),
            TempStrNoSize("\xFF""Dividend of the underlying. In terms of
percentage, e.g. 0.02 or 2%"),
            TempStrNoSize("\xFF""Annualized Volatility of the underlying. In
terms of percentage, e.g. 0.3 or 30%"),
            TempStrNoSize("\xFF""Days left to maturity. e.g. 60 = 60 days. Day
counter is 240/yr. "),
            TempStrNoSize("\xFF""Number of nodes."));
    }
```

2. 利用 Excel-Addin 进行定价

在加载宏编译成功后，我们即可将其作为下一步在 Excel 中的定价分析工具。利用二叉树期权定价模型求解欧式期权和美式期权，除了需要用到通常的欧式期权和美式期权定价参数，还需要确定用于计算的节点数量。

首先建立期权定价的参数表，并为相应的参数单元格命名，然后使用 Excel-Addin 中的函数对欧式期权和美式期权进行定价。其中，option()和 AmericanOption()分别为欧式期权的解析解和美式期权的近似解函数，感兴趣的读者可在书中附带的程序中找到源代码。

```
    option(OptionType,Spot,Strike,Rate,Dividend,Volatility,DaysToMaturity,
Output)
    AmericanOption(OptionType,Spot,Strike,Rate,Dividend,Volatility,
DaysToMaturity,Output,"Analytic")
    BinomialOptionPrice(OptionType,Spot,Strike,Rate,Dividend,Volatility,
DaysToMaturity,Nodes)
     BinomialAmericanOptionPrice(OptionType,Spot,Strike,Rate,Dividend,
Volatility,DaysToMaturity,Nodes)
```

1.5.5 实验结果

实验结果表明运用风险中性定价方法使用 Excel-Addin 在 Excel 可以快速计算出经典欧式期权、经典美式期权和数字期权的价格。通过与其解析解进行对比，节点数选择 201 时，二叉树期权定价模型算法的计算结果非常接近解析解。具体结果如图 1-34 所示。

参数表	
Spot	100
Rate	4%
Dividend	0%
Volatility	20%
Strike	105
DaysToMaturity	240
Nodes	201
Output	Price

欧式期权	Call	Put	DigitCall	DigitPut
显式解	7.57	8.45	0.43	0.54
二叉树	7.57	8.45	0.44	0.52

美式期权	Call	Put	DigitCall	DigitPut
近似解	7.57	9.06	0.82	1.00
二叉树	7.57	9.10	0.79	1.00

图 1-34　二叉树期权定价模型的欧式期权和美式期权定价对比

实验中通过计算在不同标的资产价格下取不同节点数的有限差分法，并计算与解析解相比的均方误差可以看到，在误差分析表中，均方误差的对数与节点数量的对数确实成反比的线性关系，其斜率为–1.05，与我们之前的分析基本吻合。由此可见，二叉树期权定价模型的渐近收敛速度和节点数量成正比关系。具体结果如图1-35所示。

参数表	
OptionType	Call
Output	Price
Spot	100
Rate	0%
Dividend	0%
Volatility	20%
Strike	100
DaysToMaturity	240

误差分析表						
节点数量（对数）	1.00	1.30	1.60	1.90	2.20	2.51
均方误差（对数）	-0.19	-0.54	-0.86	-1.13	-1.50	-1.79
斜率拟合						-1.05

图 1-35　二叉树期权定价模型的欧式期权定价误差分析