CQF 教材 中译本

保罗·维尔默特论
数量金融

（Paul Wilmott on Quantitative Finance）

第六部分
数值方法与程序

（Part Six Numerical Methods and Programs）

Paul Wilmott 著

李 敬 译

前　言

这份译稿是我个人进行金融工程学习的一个副产品。对于我来说，在学习这种外文专著的过程中将其翻译出来，一方面可以作为个人的一份资料存档，另一方面也可以加深对技术细节的理解。在时间条件允许的状况下，这是一种很好的学习方法。

我之所以要将这份译稿发布出来，一方面是希望有业内的资深人士能够就此给我一定指导，同时也希望它能成为我与同好们进行交流的一个契机；另一方面是考虑目前量化投资的方向逐步的为国内投资界所接受，也有很多人正在开展这方面的学习，这份译稿或许能够帮助到一些人。

事实上维尔默特教授的这本专著在国内已经有了一个初步的译本，《金融工程与风险管理技术》，但是这个译本只是摘译本，而且翻译的水平实在难以恭维。我想，至少对于第六部分来说，多一个译本的选择也是好的。

我接下来的计划是学习并翻译本书的第三部分“固定收益模型与衍生品”。

由于我个人学术水平和文字能力所限，译文中难免会有错误，一些表达也不是很通顺。所以读者在使用我这份译稿的时候，最好是与原文参照使用。我的信箱是：minker97@hotmail.com。我十分盼望对于译稿以及其它各个方面的指教与建议。另我的 MSN 与信箱地址相同，欢迎各位同仁交流。不过由于我工作性质的原因，白天在单位不能上网，所以只能在晚间回复信息。

最后感谢我敬爱的母亲大人在我工作过程中给予我的鼓励与支持，她老人家做的饭菜十分美味。

李　敬

2010 年 3 月 24 日

目　　录

保罗·维尔默特论数量金融

第六部分 数值方法与程序

本书的最后部分涉及数值方法方面的内容，这些方法是在实现本书其他部分所描述的各种模型时所需要的。应用数学和物理学领域发展出来的很多技术，目前看来在解决金融问题方面也取得了成功。

尽管笔者在论述中尽量做到详细与完整，但这一部分的内容必然仍旧只是一个概述。我鼓励读者在这方面进行更为广泛的阅读。

第 76 章：数值方法概述。准备性的一章，并对自学计划提出一些建议。

第 77 章：单因素模型有限差分方法。我们之前在书中所见到的大部分模型都会导出一些形式的偏微分方程，并且，在通常情况下（并不是所有情况下），这些偏微分方程是抛物型的。抛物型的偏微分方程是一种非常易于用数值方法进行求解的方程。本章讨论了有限差分网格和导数近似方面的基本知识。在本章的最后，我们讨论求解微分方程的显式法。

第 78 章：单因素模型有限差分方法进阶。比显式法更为复杂的是隐式法。这种方法比较难以程序化，但是通过它可以获得更快以及更为准确的结果。我们将探讨另外的一些技术，这些方法将被使用于美式以及路径依赖期权的定价上。

第 79 章：双因素模型有限差分方法。当模型中有两个随机因素时，我们最终将获得一个偏微分方程，这个方程具有与两个自变量相关的二阶导数项。我们仍然可以使用显式法，但为了计算速度我们可能要使用一个可用的隐式法进

行求解。

第 80 章：蒙特卡洛模拟。对于一些路径依赖问题，或者当我们面对一些高维问题时，我们可能需要使用模拟法进行定价。

第 81 章：数值积分。在一些特殊情况下，数值积分法是非常有用的。其具体的实现与蒙特卡洛路径模拟方法有所相关。

第 82 章：有限差分法程序。本章包含几个 Visual Basic 程序，用以对前面所描述的大部分数值技术进行描述。

第 83 章：蒙特卡洛法程序。本书的最后包含一些用 Visual Basic 编写的蒙特卡洛程序。

第 76 章　数值方法概述

本章内容:

- 有限差分方法
- 蒙特卡罗模拟
- 数值积分
- 学习计划

76. 1 简介

到现在为止，我们的讨论还是停留在理论上，数值方法将把我们导入实践层面。

本章的目的，是令我们在接触数值方法的具体内容前，可以了解这些技术的背景、应用范围，并对这些技术的实现方法有一个初步的认识。我还将对这些技术在计算时间上的性能进行阐述，并给出一个学习计划，以帮助你们建立自己的实践经验。

76. 2 有限差分方法

有限差分方法被用来寻找微分方程的数值解。通过使用有限差分网格而不是二叉树，我们将可以在股票的（价格-时间）空间的所有点上解出合约的价值。在数量金融领域里，微分方程一般是扩散型或抛物型的，这些偏微分方程仅在以下几个方面有所不同:

- 维度
- 系数的函数形式
- 边界与终值条件
- 判决条款
- 线性或非线性

维度

一份期权合约是建立在单一资产的基础上还是多个资产的基础上？其回报是否有强烈的路径依赖？如我们在第 24 章所见，对这些问题的回答决定了维度。在实际状况中，我们至少面对两个维度：资产价格或利率，以及时间。有限差分法在处理低维度问题时性能较好（最高不超过四维）。超过这个维度，它的时间性能就会大幅度的下降。虽然有限差分法可以很方便的推广到你想要的任何维度，但我们将仅考察三维以内的有限差分法的应用——也就是说，一个时间维度再加上另外两个。

系数的函数形式

一个股票期权问题和一个单因素利率期权问题的主要区别，就在于其漂移率率与波动率的函数形式的不同上。它在数学上体现为对偏微分方程系数的调整。一个标准的股票模型是对数正态的，但在固定收益方面，用得更多的是标准正态模型。这会给我们造成什么问题吗？不会的。除非在方程的系数形式很简单，以至于你要去尝试解出方程的显式解时，它才会有影响。如果你使用数值方法对方程进行求解，它不会造成任何的问题。在我们考察有限差分法的细节问题时，我们不会预设任何的系数函数形式。

边界与终值条件

在数值求解的过程中，一个看涨期权和一个看跌期权间的主要不同就在于其

终值条件上。你需要告诉有限差分过程如何开始。不过很奇特地，在金融应用方面，我们是从合约的到期日开始进行有限差分过程，向前逆推到当前状态。边界条件是我们告诉有限差分过程的，类似于敲出障碍在哪儿这样的信息。当我们进行编码的时候，希望代码尽可能的通用，并具有可重用的性质。这就是说，当我们从一个合约或模型转移到另一个时，我们不需要对代码进行太多的改动。所以我们应当将终值条件之类的东西放到某些外部函数中去，以便于修改。

判决条款

提前执行，分期付费，选择权条款，这些都是奇异型合约中嵌入式判决条款的例子。使用有限差分数值方法去处置这些问题是非常简易的——就象是这种数值技术是天生要来处置这些奇异合约的一般。在一个有限差分程序中，欧式期权和美式期权间的差别仅限于大约三行左右的代码——进行这种修改仅需不到一分钟的操作。

线性或非线性

绝大多数金融模型是线性的，所以你可以通过分别计算各个合约然后加总的方式计算一个期权组合。一些更现代的模型是非线性的——我们在本书中已经看到过几个。当我们使用有限差分法时，线性或非线性不会造成太大的不同。因此，选择这个数值方法将会给你在模型的使用上带来很大的弹性。

76. 2. 1 运行效率

有限差分法非常适于处置低维问题，并且是处置内嵌判决条款合约的首选方法。同时它们在处置非线性微分方程方面也是很出色的。

这里我仅仅给出一个式子，以便计算一个为期权组合进行定价的有限差分格

式的典型执行时间。它的具体推导过程可以参见第 79 章。

为一个期权组合定价，并计算其对基础资产价格和时间的敏感度，需要的时间是：

$$O(M\varepsilon^{-1-d/2})$$

其中 M 是组合中期权的数目，ε 是我们所期望的精度，d 是除时间外的维度。

76. 2. 2 学习计划

如果你是一个数值分析方面的新手，并且你想要学习这些技术，以便实现我在前面描述的那些模型的话，那么你需要一个学习计划。这里我会给你一些关于如何进行数值方法学习的建议。

- **显式法/欧式看涨、看跌以及二值期权：** 在一开始，你应当学习显式法在求解欧式期权的布莱克-舒尔斯方程中的应用。这非常易于编程，你不会在其中犯很多的错误。
- **显式法/美式看涨、看跌以及二值期权：** 美式期权方面的应用并没有难多少。
- **克朗克-尼科尔森法/欧式看涨、看跌以及二值期权：** 一旦你拿下了显式法后，你应当学习克朗克-尼科尔森隐式法。这种方法编程难一些，但是你会获得更好的精度。
- **克朗克-尼科尔森法/美式看涨、看跌以及二值期权：** 为美式期权定价并不比为欧式期权定价要多做多少努力。
- **显式法/路径依赖期权：** 现在，你已经是非常精通数值方法了，是时候为路径依赖型合约定价了。你可以从离散采样的亚式期权开始，然后尝试在连续采样的亚式期权上开展工作。最后，你应当尝试回望式期权。
- **利率产品：** 在非路径依赖与路径依赖的利率产品方面重复以上的工作。首先定价利率上限与利率下限，然后是指数递减比率互换。
- **双因素显式法：** 通过用显式法定价一个可转换债券来开始双因素方面的工

作。在这里股价和即期利率都是随机的。

- **双因素隐式法：**在最后，使用本书中所描述的方法去实现一个应用于可转债的双因素隐式数值方法。

76. 3 蒙特卡罗方法

蒙特卡罗方法模拟金融模型中基础资产的随机性行为。因此，它听起来抓住了问题的核心。一定要记得，尽管在定价时你必须模拟风险中性随机游走，但合约的价值是所有现金流的现值期望。当实现蒙特卡罗方法时，你需要注意以下几点：

- 维度
- 系数的函数形式
- 边界与终值条件
- 判决条款
- 线性或非线性

再来！

维度

对于每个随机因素，你都必须模拟一个时间序列，显然这将花费比较长的时间。但计算时间仅仅是与因素数量成正比的关系，这已经不错了。当处理较高维问题时，有限差分法已经不再适用，而蒙塔卡罗法就成为一个理想的方法了。

系数的函数形式

如同有限差分法一样，只要你不想去寻找方程的显式解，漂移率与波动率的具体函数形式就不会造成大的影响。

边界与终值条件

与有限差分法中的状况类似的，终值条件是合约的回报函数，而边界条件则是我们去执行触发条款的位置。

判决条款

当你处置一个带有嵌入式判决条款的合约时，蒙塔卡罗方法将变得笨重。这是模拟方法的一个主要缺点。你会发现在使用蒙特卡罗法时，你只能求出在当前时间与股票价格下的期权价值。但是如果要为美式期权正确定价的话，你必须要了解在时-价空间中每一点上的期权价值。我们无法在蒙特卡罗法中找到相关的部分。

线性或非线性

模拟法同样也不善于处理非线性问题。一些模型并没有对于概率或期望的可用的说明，所以你使用以随机模拟为基础的方法去处理它们是很困难的。

76. 3. 1 运行效率

如果在模型中具有 d 种基础资产并且期望的计算精度为 ε，则计算时间为:

$$O(d\varepsilon^{-3})$$

在确定各种对冲系数（Greeks）时，蒙特卡罗方法会使用较长的时间，不过从另一个方面来说，我们可以在同一时间里为多个期权进行定价了。

76. 3. 2 学习计划

这里是蒙特卡罗路径模拟方法的学习计划:

- **单一股票基础上的欧式看涨、看跌以及二值期权:** 模拟单一股票的路径，得出单一期权或期权组合的收益，最后计算收益期望及其现值以便为合约定价。
- **单一股票基础上的路径依赖期权:** 为障碍期权、亚式期权及回望期权定价。
- **多个股票基础上的期权:** 通过模拟多个相关的随机游走，来为一份多基础资产合约定价。你会看到维度是如何影响计算时间的。
- **利率衍生品、即期利率模型:** 这并不比股票方面的内容难多少，只是要记得在计算所有路径的总期望之前先要得出沿着每条已实现路径的利率值。
- **HJM 模型:** 更进一步的工作是 HJM 利率模型，首先考虑单因素的状况，然后是双因素状况等。
- **BGM 模型:** HJM 模型的离散版本。

76. 4 数值积分

偶尔的，我们可以以多重积分的形式写出一个期权定价问题的解。这是因为此时你可以通过收益期望的方式去表示期权价值，同时这个收益期望可以表示为收益函数与一个概率密度函数的乘积的积分。这仅仅是在一些特殊的情况下才是可能的: 期权必须是欧式的，描述基础资产行为的随机微分方程必须是显式可积的（对数正态随机游走在这方面性质很好），然后收益通常不应是路径依赖的。如果以上这些都成为可能的话，那定价工作就容易了——你拥有一个公式，唯一的困难是向这个公式代入数字。这就是数值积分技术的实际主题。注意下面的问题:

- 你是否可以用积分的形式写出一个期权的价值?

这可是一个“坚果”。

76. 4. 1 运行效率

我们在第 81 章里描述了多种数值积分方法，但其中两种最为通用的依旧是基于随机数生成的。其中一种使用正态分布数，而另一种使用所谓的低差异序列（确定性超均匀分布序列，low discrepancy sequences）。低差异序列的好处在于它们在表面上具有随机性的行为，但不会出现真实随机序列中不可避免的聚簇现象。

使用简单的正态分布数，如果我们为 M 个期权定价，并希望获得 ε 的精度，则计算时间是

$$O(M\varepsilon^{-2})$$

如果使用低差异序列，计算时间将会是:

$$O(M\varepsilon^{-1})$$

可见这个方法的计算速度很快，但不幸的是，它并非经常可用的。

76. 4. 2 学习计划

这里是数值积分的学习计划

- **使用正态序列计算单一股票基础上的欧式看涨、看跌以及二值期权:** 非常简单，你只需计算一个单积分。
- **使用正态序列计算多个对数正态股票基础上的，回报非路径依赖的任意欧式期权:** 你只需改变一个函数。
- **使用低差异序列计算多个对数正态股票基础上的，回报非路径依赖的任意欧式期权:** 只需在前面的代码中改变随机数的生成方式。

76. 5 总结

	有限差分	蒙特卡罗	数值积分
低维	好的	无效率的	好的
高维	慢的	杰出的	好的
路径依赖	看情况	杰出的	不好的
对冲系数	杰出的	不好的	杰出的
组合	无效率的	非常好的	非常好的
判决条款	杰出的	坏的	非常坏的
非线性	杰出的	坏的	非常坏的

如果你完成了整个学习计划，那么你将成为一名高等级的专家了。

扩展阅读:

我们所能使用的数值方法并不只有限差分法和蒙特卡罗法，Topper (2005) 描述了有限元方法。这种方法一般用于结构科学，但现在正被引入金融学中。

第 77 章　单因素模型有限差分方法

本章内容:

- 有限差分网格
- 如何逼近一个函数的导数
- 如何将布莱克-舒尔斯偏微分方程转化为差分方程
- 显式有限差分法——二叉树方法的一般化

77. 1 简介

很少有我们能求出期权价值的显式解的情况，除非这个问题非常的简单。事实上我们经常不得不采用数值方法去解决一个问题。在前面的章节中我曾经介绍过期权定价的二叉树方法。它使用一个有限的树结构，这个树结构从当前时间的资产价格开始，不断地产生演化分支，直到期权到期日为止。对二叉树的一种看法是将其视为偏微分方程的一种解法。虽然比之于“树”，我们更倾向于谈论**“网格”**，但其实有限差分法仅仅是二叉树的一个平凡的推广。当我们要用数值方法对方程进行求解时，使用有限差分网格要比用二叉树容易得多。因为当“网格”或“树”的结构比较规则时，将一个微分方程（如布莱克-舒尔斯方程）转化为差分方程要容易一些。另外，有很多很多的方法可以提升有限差分法的性能，使得其更快、更精确，但二叉树方法就没有这样的弹性。最后，在有限差分法和其他一些方法上，有丰富的数学/数值分析方面的文献，如果我们忽略了这些的话，那可不太好了。

有限差分法与二叉树方法之间最大的不同在于，在二叉树方法中，扩散的行为方式和波动率是固化在树的结构中的；而在有限差分法中，“树”本身被固定

了，但是我们可以通过更改参数的方式来反映扩散行为的变动。如果你是数值分析方面的新手，我会在一开始就告诉你：你将发现用数值方法去求解抛物型微分方程是很容易的事情；如果你并非新手，已经在二叉树方法上获得了一定的教育，那现在是你丢弃它们的时候了。从我个人的情况来说，在我解决实际问题的过程中，有大约 75%的时间是在使用有限差分法的，蒙特卡罗模拟占了25%，其余的时间我会使用显式的公式。我所使用的公式基本上都是看涨/看跌期权的正规的布莱克-舒尔斯公式。我从来不在障碍期权上使用显式公式，因为对于这种期权来说，使用固定的波动率是高度危险的。只有一次我曾严肃地使用过二叉树方法，相对于数值分析来说，它在建模方面的帮助更大些。

在本章中，我会展示如何使用网格法去逼近导数，以及如何将布莱克-舒尔斯方程改写为差分方程。我会示范多种方法去完成此项工作，并比较它们各自的优点。在下一章中，我会展示如何扩展这一思想，以便为一个带有提前执行条款的合约及一个奇异期权合约定价。当我讨论这些数值方法时，我会经常使用布莱克-舒尔斯方程作为例子，但事实上这些方法都可以被应用于其他问题上，如随机利率问题。我假定我的读者有足够的智力水平，当他们学习了有限差分法在股票、货币以及商品领域的应用后，可以自主地将这些方法推广于固定收益领域。相信你不会令我失望。

77. 2 概述

本章是讨论有限差分法方法族的第一部分。我们将从对网格的介绍开始，然后是使用离散数据点集去逼近导数的方法——它是通过网格格点上的值导出的。

在我们考察完微分后，我们将看到如何将关于期权价值的已知信息集结合进来。这些信息包括期权的回报函数，以及期权价值在价格高位和低位上的表现。我们将使用它们作为我们的终值和边界条件。

最后，也是最重要的，我们将会看到，定义在连续时间和连续价格空间上的布莱克-舒尔斯偏微分方程是如何被近似为一个定义在离散时间和离散价格空间上的——也就是说，定义在我们的网格上的——差分方程的。

然后我们就可以解出期权的价值与对冲系数了。

77. 3 网格

图 77.1 是第 15 章所介绍的二叉树的图形。这种结构通常被用来给简单的、非路径依赖的合约定价。其基本思想在第 15 章已经作了解释。在有限差分领域，我们使用图 77.2 中所示的网格。第一张图形中，**节点**被放置在时间等距和对数价格等距的位置上，而有限差分网格通常使用相等的**时间步长**以及相等的价格或对数价格步长。当然，如果我们愿意，我们也可以使用任意形状的网格。

我们将仅仅讨论使用相等时间与资产步长的有限差分法，这样做有利有弊。当我们求解一个布莱克-舒尔斯方程时，使用一个具有固定对数价格步长的网格看起来更加吸引人，因为基础资产是遵循对数正态随机游走的。当然如果我们想要去使用这种网格的话，一个更简单的方法是换元。在新的变量 $x = \log S$ 的基础上重写布莱克-舒尔斯方程，这样一来固定对数价格步长就转化为固定的 x 步长了。你也可以做得更多，将式子转化为更规整的热传导方程。这样做的一个坏处是固定对数价格步长意味着在低价区间会集中大量的网格节点，而在这里通常不会发生什么重要的事情。当我使用数值方法解题时，我很少进行任何的方程变换。其主要原因是，一些种类的合约是要去引用真实的金融变量的：将类似于障碍期权这样的合约的方程转化为热传导方程会造成一定的问题；而对于另外一些问题，这种转化则是不可能的——如对于那些基础资产价格的波动率对时间和价格具有依赖关系的合约，以及利率产品等。所以我们倾向于在解题时使用真实金融变量。

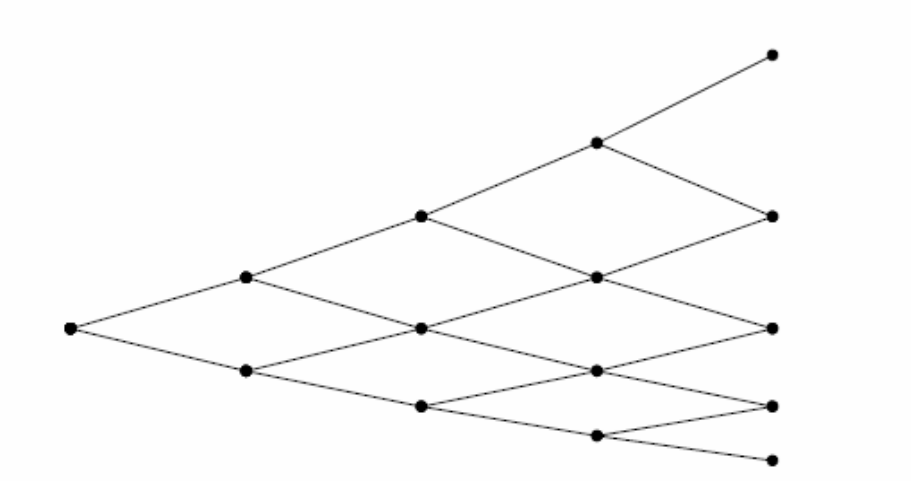


图 77.1 二叉树

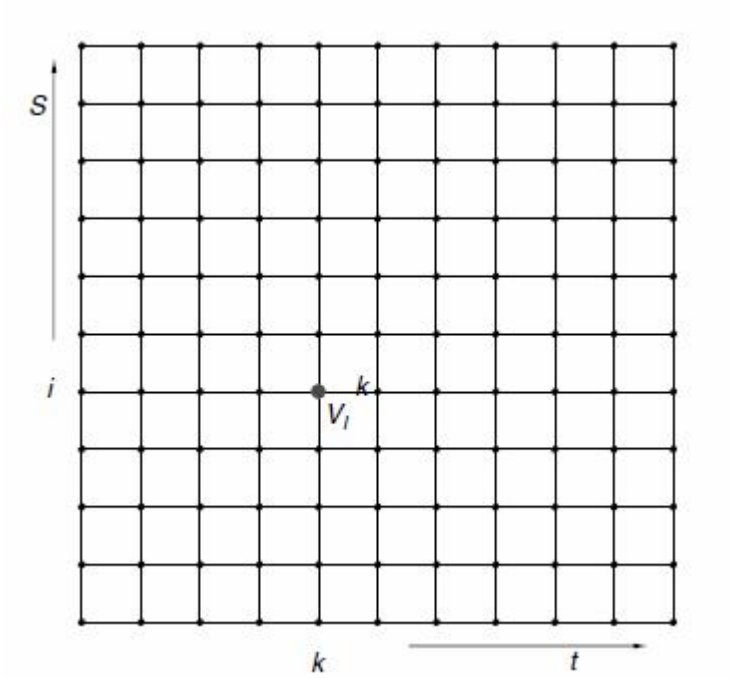


图 77.2 有限差分网格

我同时会将论述集中于后向抛物线形方程上，虽然大部分偏微分方程和数值分析方面的书籍都使用前向方程去说明有限差分法。前向与后向在这里并没有什么大的区别，仅仅是将时间的正负符号改变一下而已。(注意确认你是将初值条件加在前向方程上而将终值条件加在后向方程上。)

77. 4 使用网格进行微分

让我们首先介绍一些符号：时间增量写做δt，资产增量写做δS，这两者都是常量。形成网格的那些格点在资产价格上的位置是：

$$S = i\delta S$$

在时间上的位置是：

$$t = T - k\delta t$$

这里$0 \le i \le I$并且$0 \le k \le K$，这就是说，我们解题时所涉及的资产价值区间是从0到$I\delta S$。考虑到布莱克-舒尔斯方程在S上的定义域是$0 \le S < \infty$，所以我们是用去$I\delta S$近似无穷大。在实践中，这个上限没必要搞得太大。一个典型的设定是期权执行价格的3到4倍，或者，更一般的，一些关键价位的3到4倍。从这个意义上来说，障碍期权更易于用数值方法进行求解，因为你不用对整个的价格区间进行计算。对于一个上升敲出期权来说，不需要去构造障碍以上的网格。

我将以如下形式写出每个格点上的期权价值：

$$V_i^k = V(i\delta S, T - k\delta t)$$

这里上标是时间变量而下标是资产变量。注意我是如何改变时间的方向的，当k增加时，时间是递减的。

假设我们知道每一格点上的期权价值，我们可以由此推出期权价值关于价格和时间的导数吗？也就是说，我们可以找出那些将要带入布莱克-舒尔斯方程的参量吗？

77. 5 θ的近似

V关于t的一阶导数的定义很简单：

$$\frac{\partial V}{\partial t}=\lim_{h\to 0}\frac{V(S,t+h)-V(S,t)}{h}$$

很自然的，我们可以在网格的基础上用下式去近似时间导数:

$$\frac{\partial V}{\partial t}(S,t)\approx\frac{V_i^k-V_i^{k+1}}{\delta t} \qquad (77.1)$$

这就是为我们对期权的 Theta 值的近似，它使用图 77.3 中所示的两点上的期权价值。

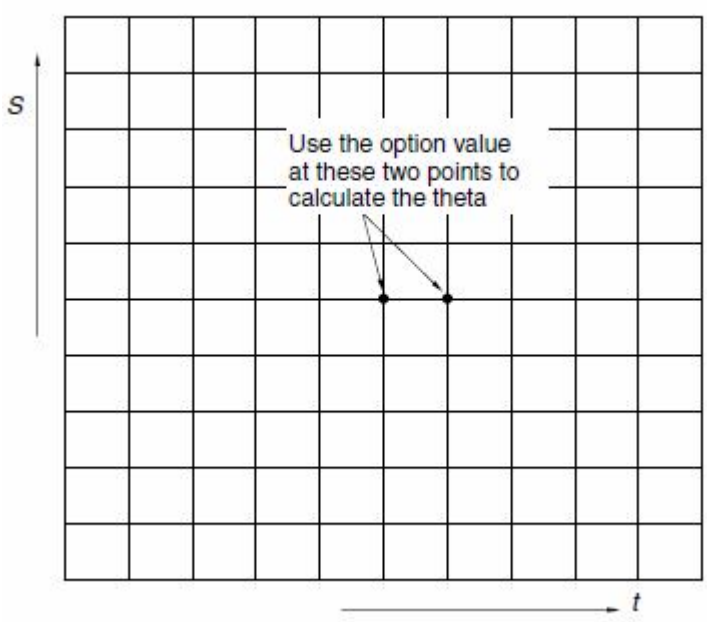


图 77.3 Theta 的近似

这个近似的精度如何？我们可以通过泰勒公式在(S,t)点上针对δt展开期权价值的表达式，如下:

$$V(S,t-\delta t)=V(S,t)-\delta t\frac{\partial V}{\partial t}(S,t)+O(\delta t^2)$$

将其转化为格点值，就是:

$$V_i^k=V_i^{k+1}+\delta t\frac{\partial V}{\partial t}(S,t)+O(\delta t^2)$$

整理上式，得:

$$\frac{\partial V}{\partial t}(S,t)=\frac{V_i^k-V_i^{k+1}}{\delta t}+O(\delta t)$$

我们的问题有了回答，误差是$O(\delta t)$。当然，也有可能比这更精确，误差取决于对t的二阶导数的数量级。在这个问题上我不会继续讨论细节问题。

还有其他的方法可以去近似期权价值的时间导数，但我们目前先使用这个。

77. 6 Δ的近似

同样的思想可被用于近似对S的一阶导数，也就是 Delta，但现在我首先要展示一些选择。让我们考察网格在一个时间点上的截面，如图 77.4 所示。图上有三种东西：我们要去近似的函数（曲线），在格点上的函数值（点）和对一阶导数的可能的近似（三条直线）。

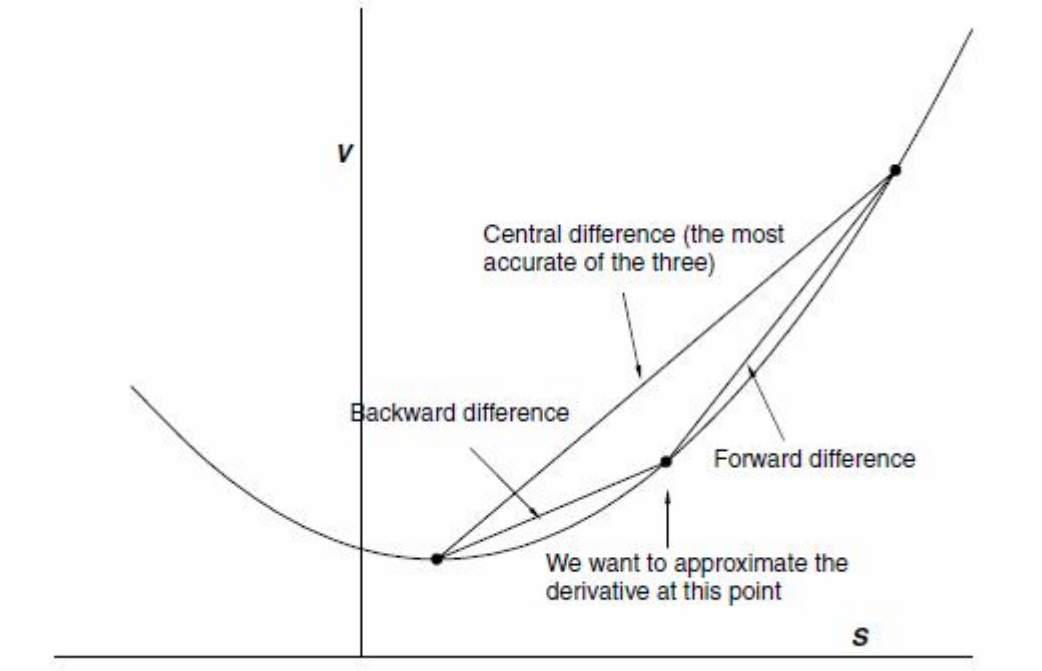


图 77.4 对 Delta 的近似

这三种近似分别是：

$$\frac{V_{i+1}^k - V_i^k}{\delta S}、\frac{V_i^k - V_{i-1}^k}{\delta S}和\frac{V_{i+1}^k - V_{i-1}^k}{2\delta S}$$

它们相应地被称为是：**前向差分**，**后向差分**和**中央差分**。

其中的一种近似比另外两种要好，这在图上是显而易见的。在(S,t)点针对δS对期权价值进行泰勒展开，我们得到：

$$V(S+\delta S,t)=V(S,t)+\delta S\frac{\partial V}{\partial S}(S,t)+\frac{1}{2}\delta S^2\frac{\partial^2 V}{\partial S^2}(S,t)+O(\delta S^3)$$

类似的：

$$V(S-\delta S,t)=V(S,t)-\delta S\frac{\partial V}{\partial S}(S,t)+\frac{1}{2}\delta S^2\frac{\partial^2 V}{\partial S^2}(S,t)+O(\delta S^3)$$

用其中一个减去另外一个，除以$2\delta S$，然后整理可得：

$$\frac{\partial V}{\partial S}(S,t)=\frac{V_{i+1}^k-V_{i-1}^k}{2\delta S}+O(\delta S^2)$$

中央差分的误差是$O(\delta S^2)$，相形之下前向和后向差分的误差都太大了：$O(\delta S)$。中央差分之所以可以取得较好的精度，是由于关于S的差分在定义上的对称性，使我们可以幸运地消去二阶项。

77. 6. 1 单边差分

对S点上的差分需要知道$S+\delta S$和$S-\delta S$点上的期权价值。但有些情况下我们并不知道其中的一个，例如当我们处理区间边缘（也就是$i=0$或$i=I$）的情况时，出于对稳定性的考虑（这是我们后面要讨论的一个重点）这里使用单边导数会好一些。如果我们需要一个单边导数的话，是否我们就必须使用简单的前向或后向差分？有更好的办法么？

简单的前向或后向差分仅使用两个点去计算导数。如果我们使用三个点，我们可以取得更好的精度。为了找出使用三个点的最好的近似，我们需要再次使用泰勒展开。

假设我想要用S，$S+\delta S$，$S+2\delta S$这三个点去计算期权的 Delta 值，如何才尽可能的精确呢？首先，在$S+\delta S$和$S+2\delta S$点上对期权价值进行泰勒展开:

$$V(S+\delta S,t)=V(S,t)+\delta S\frac{\partial V}{\partial S}(S,t)+\frac{1}{2}\delta S^2\frac{\partial^2 V}{\partial S^2}(S,t)+O(\delta S^3)$$

和

$$V(S+2\delta S,t)=V(S,t)+2\delta S\frac{\partial V}{\partial S}(S,t)+2\delta S^2\frac{\partial^2 V}{\partial S^2}(S,t)+O(\delta S^3)$$

如果我计算这个线性组合

$$-4V(S+\delta S,t)+V(S+2\delta S,t)$$

我会得到:

$$-3V(S,t)-2\delta S\frac{\partial V}{\partial S}(S,t)+O(\delta S^3)$$

这样，二阶导数项$O(\delta S^2)$就都被消去了，因此:

$$\frac{\partial V}{\partial S}(S,t)=\frac{-3V(S,t)+4V(S+\delta S,t)-V(S+2\delta S,t)}{2\delta S}+O(\delta S^2)$$

这个近似具有和中央差分同样的精度，但比简单的前向差分要精确得多。它不使用S以下格点上的V值。如果我们要使用更好的后向差分去计算 Delta，我们可以选择:

$$\frac{\partial V}{\partial S}(S,t)=\frac{3V(S,t)-4V(S-\delta S,t)+V(S-2\delta S,t)}{2\delta S}+O(\delta S^2)$$

大部分情况下我会使用中央差分去近似 Delta，但有时我也需要去使用单边近似。

77. 7 Γ的近似

Gamma 是期权价值对基础资产价格的二阶导数。计算它的一种方法是通过前向差分去估计 Delta，然后使用后向差分去估计 Delta，然后用这两个估计值的

差分除以格点间的距离去估计 Gamma。

前向差分是:

$$\frac{V_{i+1}^k - V_i^k}{\delta S}$$

你可以将它看作是在 $S + \frac{1}{2}\delta S$ 点上的 Delta 的中央差分近似。后向差分是:

$$\frac{V_i^k - V_{i-1}^k}{\delta S}$$

同样可将其视为一个 Delta 的中央差分近似,不过这个是在 $S - \frac{1}{2}\delta S$ 点上的。

对这两个 Delta 取差分，并除以 δS，也就是两个中点间的距离，我们可以很自然地得到对 Gamma 的近似:

$$\frac{\partial^2 V}{\partial S^2}(S,t) \approx \frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2}$$

同样的，我们可以通过考察泰勒级数来检验这个结果，近似的误差是 $O(\delta S^2)$。我将把具体的论证作为一个练习留给读者。

77. 8 例子

图 77.5 展示了在网格上的一些期权价值，时间步长是 0.1，资产价格步长是 2。从这些数据中我们可以估计 Theta:

$$\frac{12-13}{0.1} = -10$$

Delta 的近似是:

$$\frac{15-10}{2\times 2} = 1.25$$

以及 Gamma 的近似是:

$$\frac{15-2\times 12+10}{2\times 2} = 0.25$$

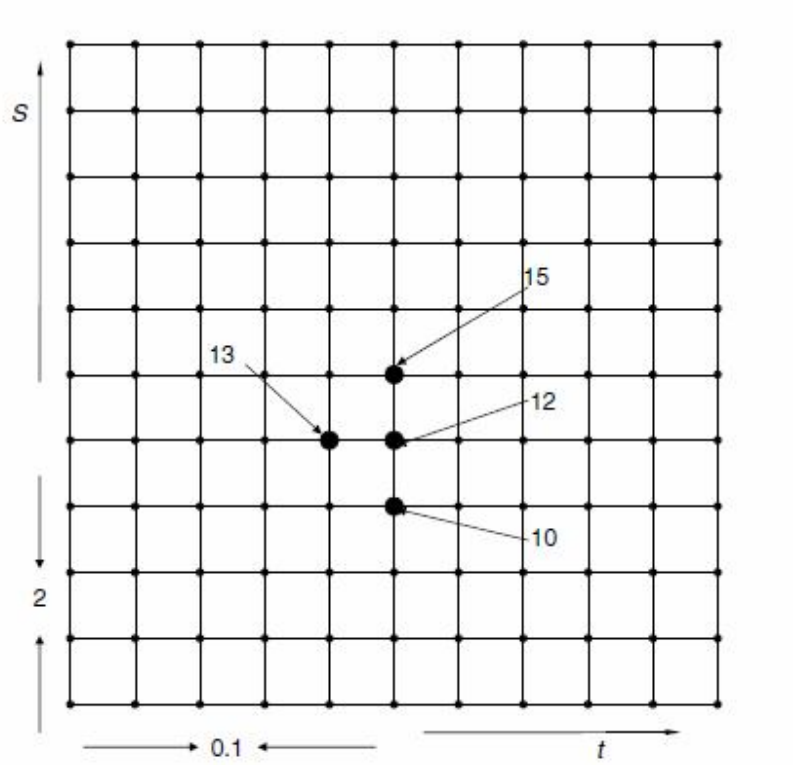


图 77.5 对冲系数的计算

77. 9 终值条件与收益

我们知道，在到期日，期权的价值就是其收益函数。也就是说，我们在时间点T（到期日）上不需要求解任何的东西，我们有:

$$V(S,T)=\text{Payoff}(S)$$

或者，用我们的有限差分符号表示:

$$V_i^0=\text{Payoff}(i\delta S)$$

方程右侧是一个已知函数。例如，当我们在为一个看涨期权定价，有:

$$V_i^0=\max(i\delta S-E,0)$$

终值条件是有限差分过程的起始，这与二叉树方法中的反向过程是一致的。

77. 10 边界条件

当我们在下一节中开始用数值方法对布莱克-舒尔斯方程进行求解时，我们必须在求解区间的边缘指定期权的价值。也就是说，我们必须在$S=0$和$S=I\delta S$的位置上指定期权的价值。如何进行这种指定取决于期权的类型。我会给出一些例子:

例 1:

假定我们为一个看涨期权定价，我们知道在$S=0$的位置上期权价值永远是0，因此我们有:

$$V_0^k = 0$$

例 2:

对于一个比较大的S的取值，看涨期权的价值渐进于$S-Ee^{-r(T-t)}$（加上一个小的指数项）。因此我们的上边界条件应当是:

$$V_I^k = I\delta S - Ee^{-r(T-t)}$$

对于有股息的状况，式子会有一点小小的不同。

例 3:

对于看跌期权，在$S=0$的位置上有$V=Ee^{-r(T-t)}$，因此:

$$V_0^k = Ee^{-rk\delta t}$$

例 4:

看跌期权在S取值比较大时变得毫无价值，因此:

$$V_I^k = 0$$

例 5:

对于大部分的期权（包括看涨与看跌期权），一个可用在$S=0$位置上的最为有用的边界条件，是扩散项和漂移项都消失了。这就是说，在$S=0$的位置上，收益是确定性的，这导致如下边界条件:

$$\frac{\partial V}{\partial t}(0,t) - rV(0,t) = 0$$

将其数值化，可得:

$$V_0^k = (1 - r\delta t)V_0^{k-1}$$

例 6:

当 S 取值比较大时，期权的收益对于基础资产价格具有近似线性的关系，这导致我们可以使用如下的上边界条件:

$$\frac{\partial^2 V}{\partial S^2}(S,t) \to 0\text{，当 } S \to \infty \text{ 时}$$

几乎所有常见的合约都具有这样的特性，其在有限差分中的表示是:

$$V_I^k = 2V_{I-1}^k - V_{I-2}^k$$

这一关系非常有用，因为它与你想要定价的合约无关。这使得你的有限差分程序不必太过智能化。

77. 10. 1 其它边界条件

我们的问题在基础资产价格上的边界经常是有限的，或不为 0 的。这意味着我们解题的区域不一定要向下延伸到 0 或者向上延伸到无限。障碍期权是这种合约中最为常见的一类。

举个例子，假设我们要为一个上升敲出看涨期权定价，当基础资产价格触及 S_u 时，这个期权将会变得毫无价值。显然

$$V(S_u,t) = 0$$

如果我们使用数值方法去解决这个问题，我们应当如何引入边界条件?

第一种思想是: 选择适当的资产价格步长，使得障碍 $S = S_u$ 正好处于格点上也就是说 $S_u / \delta S$ 应当是整数。这确保了如下的边界条件:

$$V_I^k = 0$$

是对事实边界条件的精确表示。注意我们不再将资产价格范围扩展到比较大的S取值。上边界条件$S = S_u$可能会接近于当前资产价格水平。从某种意义上来说，这使得障碍期权问题更容易去求解，它的计算区间永远会比那些非障碍期权的计算区间要小。

有时，令我们的网格构造与障碍位置匹配是不可能的。例如，当障碍本身会移动时就会造成这种情况。如果发生了这种情况，我们必须寻找对边界条件的近似。将距离障碍最近的网格的期权价值设定为0，这种事情你一定不能去做。这种近似实在是太不精确了（$O(\delta S)$），它将摧毁你的数值解法。克服这个问题的一个方法是建立一个**虚拟点**，如图77.6。点$i = I - 1$是一个在解题区域内的真实的点，而点$i = I$则已经超越了障碍。

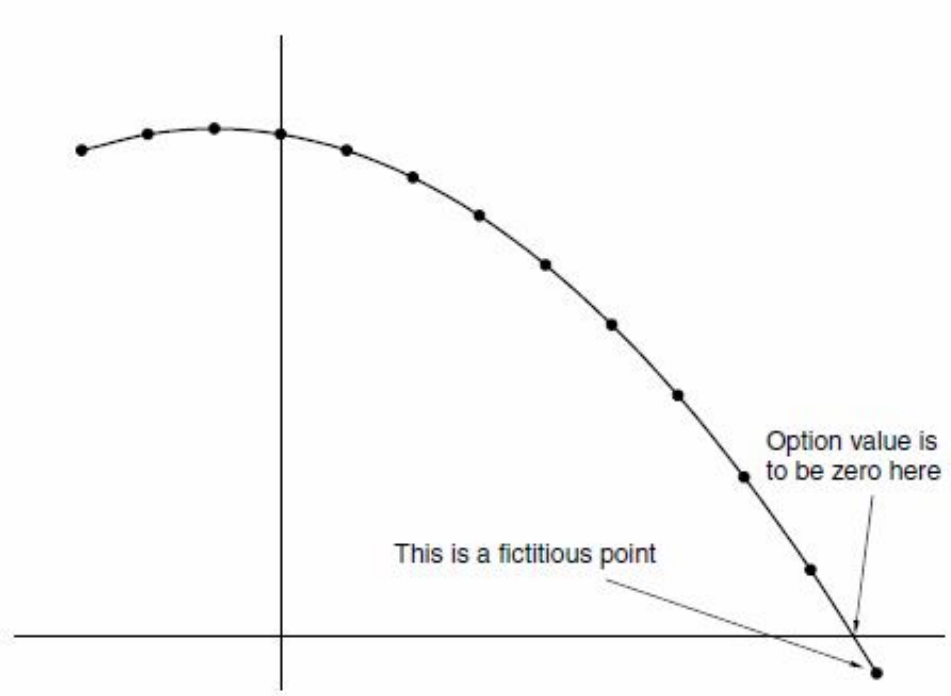


图77.6 虚拟点，为了保证障碍期权边界条件的精度而引入

例 7:

假设我们有一个边界条件:

$$V(S_u,t)=f(t)$$

当我们持有一个敲出期权时，f 将可能是 0 或者回佣的价格。如果我们拥有一个敲入期权，那么 f 将会是由障碍期权所转化的那个期权的价值。

我们可以这样建立虚拟点：令其与障碍内最近点间的连线在穿越障碍时，其取值刚好是 f。因此，关于边界条件的一个好的离散表示是:

$$V_I^k=\frac{1}{\alpha}(f-(1-\alpha)V_{I-1}^k)$$

这里

$$\alpha=\frac{S_u-(I-1)\delta S}{\delta S}$$

其精度是: $O(\delta S^2)$，与关于 S 的导数的近似精度同阶。

我已经建立了解出一些问题时所需要的所有基础条件。记住，迄今为止，我们所做的所有事情中没有困难的，一切都只是泰勒级数的简单应用。

77. 11 显式有限差分法

布莱克-舒尔斯方程是:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2S^2\frac{\partial^2V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0$$

我会将它改写为更一般的形式:

$$\frac{\partial V}{\partial t}+a(S,t)\frac{\partial^2V}{\partial S^2}+b(S,t)\frac{\partial V}{\partial S}+c(S,t)V=0$$

以强调有限差分方法的广泛适用性。我们将并不仅仅用它来为那些建立于对数正态股票基础上的期权定价。对于系数的唯一约束就在于: 如果我们是在求解

一个后向方程（也就是说，使用终值条件），我们必须有 $a>0$。

我们将对那些导数进行近似，然后将它们代入方程：

$$\frac{V_i^k - V_i^{k+1}}{\delta t}$$

$$+a_i^k\left(\frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2}\right)$$

$$+b_i^k\left(\frac{V_{i+1}^k - V_{i-1}^k}{2\delta S}\right)$$

$$+c_i^k V_i^k = O(\delta t, \delta S^2)$$

对于原来方程的每一项，我们都列出单独一行。

注意如下几点：

- 时间导数使用在 k 和 $k+1$ 时间点上的期权价值，而其它项仅使用 k 时间点上的值。
- Gamma 项使用的是中央差分，不会再使用别的方法
- Delta 项使用的是中央差分，不过这一项经常会出现使用单边导数更好的情况，我们会在后面看到例子。
- 资产价格依赖或时间依赖的函数 a，b 和 c 在 $S_i = i\delta S$ 和 $t = T - k\delta t$ 的位置上应当有显式的赋值。
- 方程的误差是 $O(\delta t, \delta S^2)$

我将把这个**差分方程**的 $k+1$ 项都整理到方程的左边：

$$V_i^{k+1} = A_i^k V_{i-1}^k + (1 + B_i^k)V_i^k + C_i^k V_{i+1}^k \qquad (77.2)$$

这里：

$$A_i^k = v_1 a_i^k - \frac{1}{2}v_2 b_i^k$$

$$B_i^k = -2v_1 a_i^k + \delta t c_i^k$$

以及:

$$C_i^k = v_1 a_i^k + \frac{1}{2} v_2 b_i^k$$

并且:

$$v_1 = \frac{\delta t}{\delta S^2} \text{ 和 } v_2 = \frac{\delta t}{\delta S}$$

这里的误差是$O(\delta t^2, \delta t \delta S^2)$；我等下会讨论这个问题。微分方程的近似误差被称为**局部截断误差**。

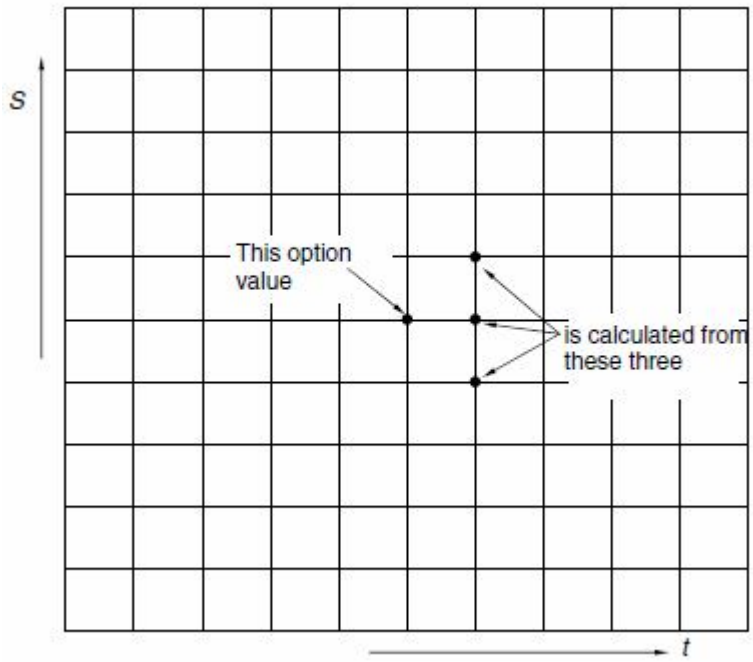


图 77.7 显式法中期权价格间的关系

方程（77.2）只在内部格点，也就是$i = 1, \ldots, I-1$上有意义，因为V_{-1}^k和V_{I+1}^k都没有定义。因此这里有$I+1$个未知数但只有$I-1$个方程。其余两个方程来自$i=0$和$i=I$上的两个边界条件。这两个端点将被分别对待。

如果我们在所有的i点上都知道了V_i^k，则方程(77.2)将告诉我们V_i^{k+1}。当我们知道V_i^0，也就是收益函数，我们可以很容易的计算V_i^1——到期日前一个时间步长上的期权价值。这样我们可以一步一步地反推格点值，直到我们想要的任

意远的时间点。由于时间点$k+1$上的期权价值是点k上期权价值的简单函数，所以这个方法被称为**显式有限差分法**。方程（77.2）中所示的期权价值间的关系如图 77.7 所示。

方程(77.2)只能用来计算$1 \le i \le I$上的期权价值，因为这个计算需要知道$i+1$和$i-1$点上的期权价值。这里就需要引入边界条件了。一般来说，我们要么指定在$i=0$和$i=I$上的V_i^k值；要么，如同上面所建议的，指定端点值和内部点值间的关系。这一思想体现在以下的 Visual Basic 代码片段。这些代码不包含所有的变量声明，也没有任何的返回值。我将很快给出一个完整的函数，但此时我希望你集中注意于终值条件和有限差分时间循环的建立上。

数组 v(i,k)用来存储期权价值。除非我们本身就想要存储所有时间步上的所有期权价值，否则这种做法在编程的时候并非最有效率的做法。我等下将会描述一个更好的做法。

首先建立终值条件，也就是收益函数:

```
For i = 0 To NoAssetSteps
    S(i) = i * AssetStep
    V(i, 0) = CallPayoff(S(i)) ' 建立终值条件
Next i
```

现在我们可以用如下的时间循环进行逆推了:

```
' 时间循环
For k = 1 To NoTimesteps
    RealTime = Expiry - k * Timestep
    For i = 1 To NoAssetSteps - 1
        V(i, k + 1) = A(S(i), RealTime) * V(i - 1, k) + B(S(i), RealTime) * _
        V(i, k) + C(S(i), RealTime) * V(i + 1, k)
    Next i
    ' S=0上的边界条件
    V(0, k + 1) = 0
    ' “无穷大”上的边界条件
    V(NoAssetSteps, k + 1) = 2 * V(NoAssetSteps - 1, k + 1) - _
        V(NoAssetSteps - 2, k + 1)
Next k
```

显式有限差分算法非常的简单，函数 A(S(i), RealTime)，B(S(i), RealTime)

和 C(S(i), RealTime)将在其它地方定义，并与资产价格 S(i)和时间 RealTime 相关。因为我在这里是为一个看涨期权定价，所以在 S=0 位置上的边界条件是 V(0, k + 1) = 0。我在上边界 i = NoAssetSteps 上使用的边界条件是 Gamma = 0。

77. 11. 1 布莱克-舒尔斯方程

对于有股息的布莱克-舒尔斯方程，其系数形式是：

$$A_i^k = \frac{1}{2}\left(\sigma^2 i^2 - (r-D)i\right)\delta t$$

$$B_i^k = -\left(\sigma^2 i^2 + r\right)\delta t$$

和

$$C_i^k = \frac{1}{2}\left(\sigma^2 i^2 + (r-D)i\right)\delta t$$

这里使用 $S = i\delta S$。如果波动率，利率和股息是固定的，则这些系数是非时间或 k 依赖的。

77. 12 显式法的收敛性

在最后的时间点 K 上，我们可以将任意 i 点上的期权价值写作：

$$V_i^K = V_i^0 + \sum_{k=0}^{K-1}\left(V_i^{k+1} - V_i^k\right)$$

这个累加中的每一项的误差都是：$O\left(\delta t^2, \delta t \delta S^2\right)$，这意味着在期权终值上的误差是：

$$O\left(K\delta t^2, K\delta t\delta S^2\right)$$

因为这里共有 K 项进行相加。如果我们在有限的 T 上对期权进行估值，则 $K = O(\delta t^{-1})$。因此，期权终值上的误差就是：$O\left(\delta t, \delta S^2\right)$。

虽然显式法易于实现，但是它并非总是收敛的。其收敛性取决于时间步长的大小，资产价格步长的大小，和系数 a、b 和 c 的大小。

通常被用来证明这种收敛性的方法很有意思，我将向你示范如何去做。这种方法并不是十分严格的，但确实可用。

考虑如下问题："如果一个小的误差被引入求解过程，那么它是会被数值方法逐步的放大，还是会被衰减？"如果一个小的误差——如舍入误差——会被逐步的放大，那么这个方法就是没有用处的。一个对稳定性的常用的分析方法是考察方程的一个如下形式的解:

$$V_i^k = \alpha^k e^{2\pi i\sqrt{-1}/\lambda} \qquad (77.3)$$

换句话说，我们将去考察一个具有波长 λ 的振荡解。如果我们有 $|\alpha|>1$ 的话，那么方程就是不稳定的。请注意，我并不关心振荡如何开始。我可以将这个特解理解为一个傅里叶级数分析的一部分。

将（77.3）代入（77.2），我们有:

$$\alpha = \left(1 + c_i^k \delta t + 2a_i^k v_1 \left(\cos\left(2\pi/\lambda\right) - 1\right)\right) + \sqrt{-1} b_i^k v_2 \sin\left(2\pi/\lambda\right)$$

出于对稳定性，也就是 $|\alpha|<1$ 的要求，我们需要:

$$c_i^k \le 0$$

$$2v_1 a_i^k - \delta t c_i^k \le 1$$

以及

$$\frac{1}{2} v_2 \left|b_i^k\right| \le v_1 a_i^k$$

为了得到这个结果，我假定所有的系数对 δS 的步长不敏感。

在一个金融问题中，我们几乎总是有负的 c，它一般就简单地是 $-r$，也就是负的无风险利率。另外两个约束是对显式法应用范围的真正限制。

通常我们会选择 v_1 为 $O(1)$，在这个情况下，第二项约束可被近似为（忽略 a 的上标和下标）:

$$v_1 \le \frac{1}{2a}$$

这是对时间步长的严格限制:

$$\delta t \le \frac{\delta S^2}{2a}$$

如果我们要通过将资产价格步长减半的方式来提升精确度的话，相应的，我们必须将时间步长减少为原来的四分之一。于是计算时间就变为原来的八倍。这样我们获得的精度就是原来的四倍，因为显式有限差分法的精度是$O\left(\delta t, \delta S^2\right)$。

在布莱克-舒尔斯方程中，时间步长的约束是:

$$\delta t \le \frac{\delta S^2}{2a} = \frac{\delta S^2}{\sigma^2 S^2} = \frac{1}{\sigma^2}\left(\frac{\delta S}{S}\right)^2$$

这个约束依赖于资产价格。由于δt应当不依赖于S，实际的约束受网格中最大的S的限制。如果有I个等长间距资产格点，则约束可简单地写为:

$$\delta t \le \frac{1}{\sigma^2 I^2}$$

如果时间步长太大，从而其约束不被满足，那么在结果中将会呈现出明显的不稳定。这种不稳定是如此的严重，如此的具有振荡性，以至于它们很容易引起注意。如果你使用显式法，你不太可能会得到一个错误，但可信的结果。

最后的一项约束同样是一个严格的限制。它可以被写做:

$$\delta S \le \frac{2a}{|b|} \qquad (77.4)$$

如果我们是在求解布莱克-舒尔斯方程，除非波动率特别的小，否则这个限制不会带来太大的不同。在另外一些问题里，这个约束会变得非常重要，稍后我将会展示如何满足这个约束。

77. 13 代码#1，欧式期权

下面的 VB 代码将会输出一个看涨或看跌（依赖于 PType 是 C 还是 P）期权的价值。时间步长被硬性确定，并依赖于资产价格步长。在满足v_1约束的前提下，它被选为尽可能大的值，这个格式应当是稳定的。

在本例中，期权价值对股票价格和时间的整个函数数组都会被输出。

```
Function Option_Value_3D(Vol, Int_Rate, PType, Strike, Expiration, NAS)
' NAS 是资产价格上的步数
    ReDim S(0 To NAS) As Double ' 资产数组
    dS = 2 * Strike / NAS ' “无穷大”是敲定价格的两倍
    dt = 0.9 / Vol ^ 2 / NAS ^ 2 ' 为了稳定性
    NTS = Int(Expiration / dt) + 1 ' 时间上的步数
    dt = Expiration / NTS ' 为了确保到期日刚好在整数个时间步长上
    ReDim V(0 To NAS, 0 To NTS) As Double ' 期权价值数组
    q = 1

    If PType = "P" Then q = -1 ' 测试是看涨还是看跌期权

    For i = 0 To NAS
        S(i) = i * dS ' 建立S数组
        V(i, 0) = Application.Max(q * (S(i) - Strike), 0) ' 建立收益函数
    Next i

    For k = 1 To NTS ' 时间循环
        For i = 1 To NAS - 1 ' 资产循环，端点分别处理
            Delta = (V(i + 1, k - 1) - V(i - 1, k - 1)) / 2 / dS ' 中央差分
            Gamma = (V(i + 1, k - 1) - 2 * V(i, k - 1) + V(i - 1, k - 1)) / _
                dS / dS ' 中央差分
            Theta = -0.5 * Vol ^ 2 * S(i) ^ 2 * Gamma - Int_Rate * S(i) * _
                Delta + Int_Rate * V(i, k - 1) ' 布莱克-舒尔斯方程
            V(i, k) = V(i, k - 1) - dt * Theta
        Next i

        V(0, k) = V(0, k - 1) * (1 - Int_Rate * dt) ' S=0上的边界条件
        V(NAS, k) = 2 * V(NAS - 1, k) - V(NAS - 2, k) ' S=infinity上的边界条件
    Next k
```

```
    Option_Value_3D = V ' 输出数组
End Function
```

设 $\sigma = 0.2$， $r = 0.05$， $E = 100$， $T = 1$ 以及 $NAS = 20$，对于一个看涨期权，程序的结果如表 77.1 和图 77.8 所示。

表 77.2 和图 77.9 展示在同样参数下，一个看跌期权的输出。

表 77.1，显式有限差分代码输出的看涨期权价值，股票价格从 60 到 100，时间从 0 到 1

	0.000	0.111	0.222	0.333	0.444	0.556	0.667	0.778	0.889	1.000
60	0.000	0.000	0.000	0.000	0.001	0.004	0.011	0.023	0.041	0.066
70	0.000	0.000	0.001	0.008	0.028	0.067	0.126	0.207	0.308	0.430
80	0.000	0.000	0.037	0.141	0.310	0.534	0.799	1.096	1.416	1.754
90	0.000	0.128	0.592	1.182	1.812	2.450	3.080	3.700	4.306	4.899
100	0.000	2.253	3.819	5.054	6.109	7.054	7.925	8.741	9.515	10.255
110	10.000	10.671	11.587	12.535	13.455	14.337	15.180	15.990	16.770	17.523
120	20.000	20.555	21.159	21.826	22.529	23.247	23.967	24.683	25.391	26.089
130	30.000	30.555	31.109	31.680	32.272	32.882	33.504	34.134	34.768	35.402
140	40.000	40.555	41.106	41.658	42.213	42.777	43.348	43.926	44.509	45.095

表 77.2，显式有限差分代码输出的看跌期权价值，股票价格从 60 到 100，时间从 0 到 1

	0.000	0.111	0.222	0.333	0.444	0.556	0.667	0.778	0.889	1.000
60	40.000	39.445	38.894	38.345	37.801	37.261	36.728	36.204	35.688	35.182
70	30.000	29.445	28.894	28.353	27.827	27.323	26.843	26.387	25.955	25.546
80	20.000	19.445	18.930	18.486	18.110	17.790	17.516	17.276	17.063	16.871
90	10.000	9.573	9.486	9.527	9.612	9.706	9.798	9.880	9.953	10.015
100	0.000	1.699	2.713	3.399	3.909	4.311	4.642	4.921	5.162	5.372
110	0.000	0.116	0.481	0.880	1.254	1.593	1.898	2.171	2.417	2.639
120	0.000	0.000	0.052	0.171	0.328	0.503	0.684	0.864	1.038	1.205
130	0.000	0.000	0.003	0.025	0.071	0.138	0.221	0.315	0.415	0.518
140	0.000	0.000	0.000	0.002	0.013	0.033	0.065	0.106	0.156	0.211

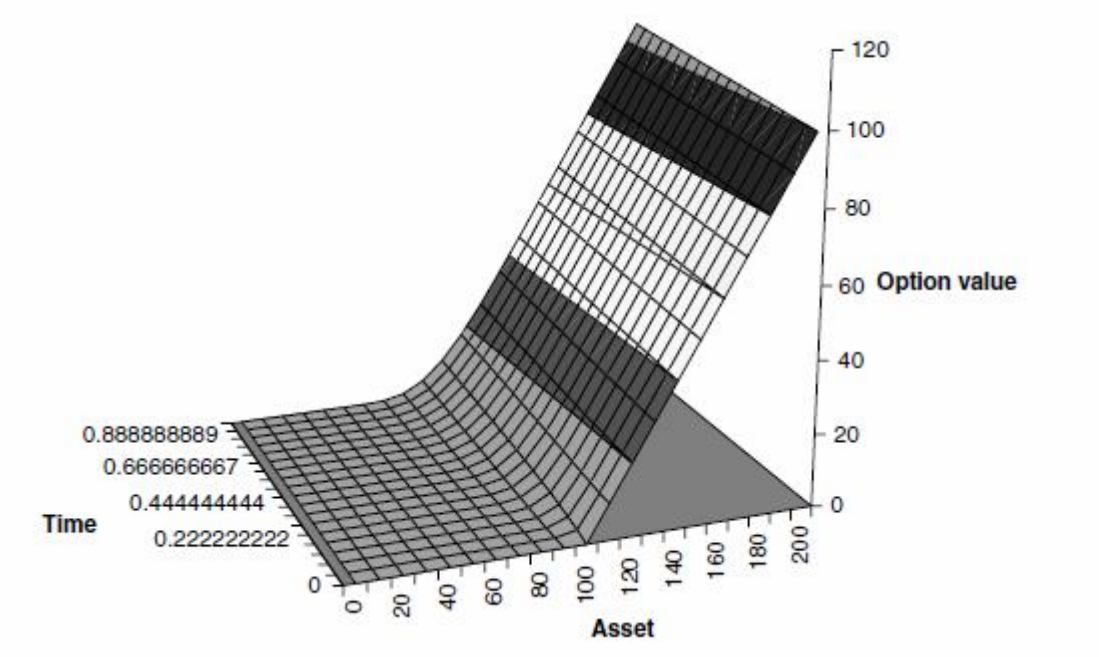


图 77.8 显式有限差分代码输出的看涨期权价值

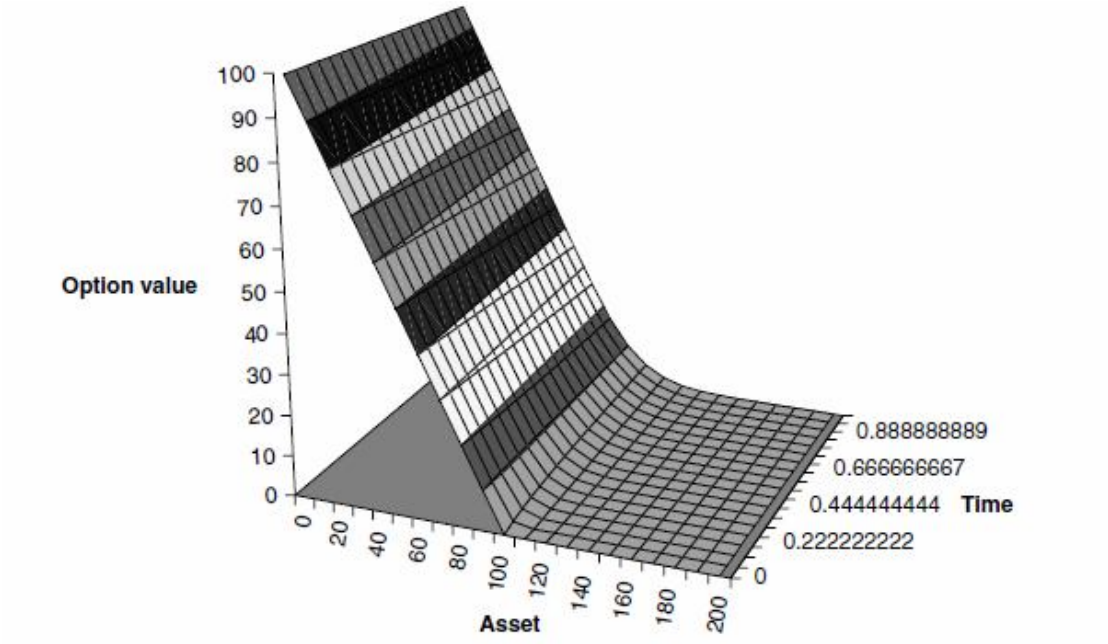


图 77.9 显式有限差分代码输出的看跌期权价值

显式有限差分程序的结果和准确的布莱克-舒尔斯公式的结果间的误差对基础资产价格的函数如图 77.10 所示。这里资产步数是 50，波动率是 20%，利率是 10%，距离到期日还有一年，并且敲定价格是 100，看涨期权。

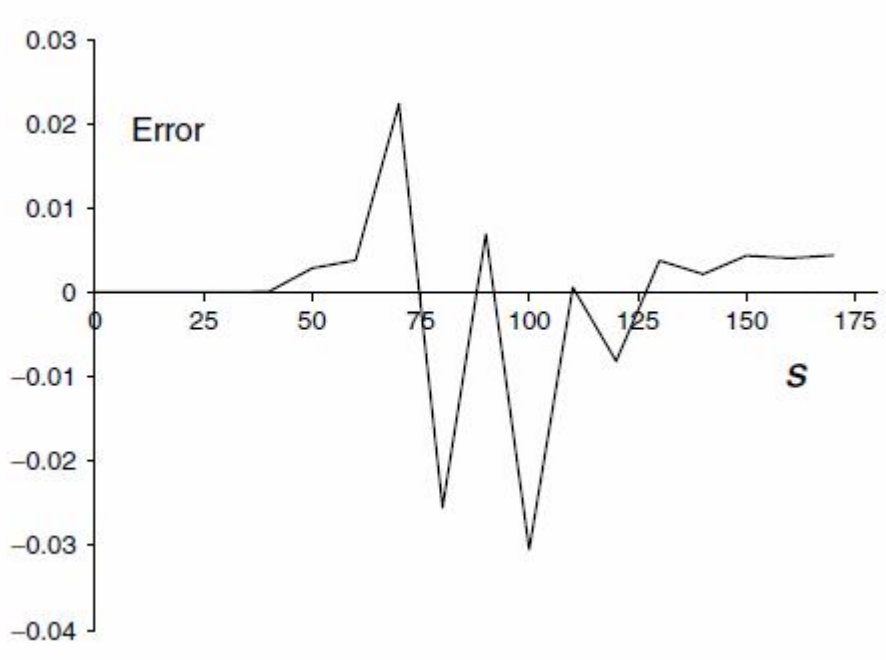


图 77.10 有限差分格式误差对资产价格的函数

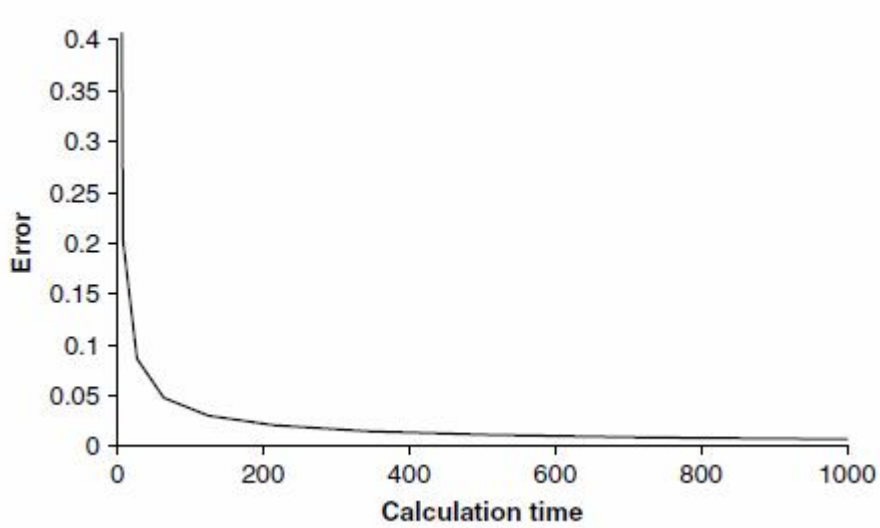


图 77.11 有限差分格式中 $\log\left(\text{Error}\right)$ 对 $\log\left(\delta S^2\right)$ 的函数

绝对误差的对数与资产价格步长的平方的对数间的函数关系如图 77.11 所示。$O\left(\delta S^2\right)$ 的行为是很明显的。在这些计算里我保持 v_1 不变。

图 77.12 展示了误差对计算时间的函数，时间单位依赖于你的计算机。（译者注：这里图 77.11 和图 77.12 似乎颠倒了。原书如此，译者不作改动，仅在

这里注明)。

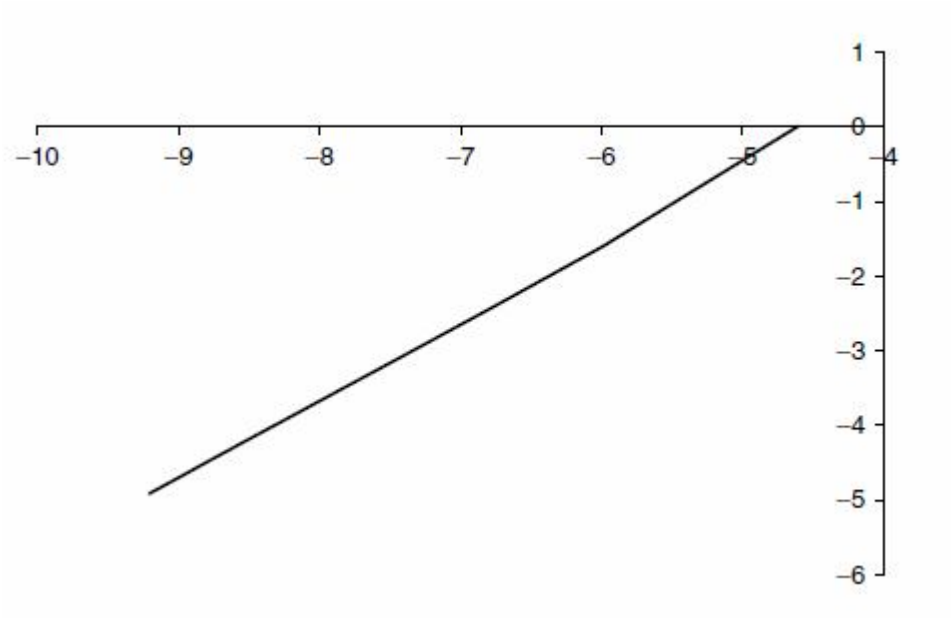


图 77.12 有限差分格式误差对计算时间的函数

77. 14 代码#2，美式期权

在有提前执行条件时，我们对程序进行一个平凡的修改。

```
Function Option_Value_3D_US(Vol, Int_Rate, PType, Strike, Expiration, _
EType, NAS)
' NAS 是资产价格上的步数
    ReDim S(0 To NAS) As Double ' 资产数组
    ReDim Payoff(0 To NAS) As Double ' 收益数组
    dS = 2 * Strike / NAS ' “无穷大”是敲定价格的两倍
    dt = 0.9 / Vol ^ 2 / NAS ^ 2 ' 为了稳定性
    NTS = Int(Expiration / dt) + 1 ' 时间上的步数
    dt = Expiration / NTS ' 为了确保到期日刚好在整数个时间步长上
    ReDim V(0 To NAS, 0 To NTS) As Double ' 期权价值数组
    q = 1

    If PType = "P" Then q = -1 ' 测试是看涨还是看跌期权

    For i = 0 To NAS
```

```
        S(i) = i * dS ' 建立S数组
        V(i, 0) = Application.Max(q * (S(i) - Strike), 0) ' 建立收益函数
        Payoff(i) = V(i, 0) ' 存储收益函数
    Next i

    For k = 1 To NTS ' 时间循环
        For i = 1 To NAS - 1 ' 资产循环，端点分别处理
            Delta = (V(i + 1, k - 1) - V(i - 1, k - 1)) / 2 / dS ' 中央差分
            Gamma = (V(i + 1, k - 1) - 2 * V(i, k - 1) + V(i - 1, k - 1)) / _
                dS / dS ' 中央差分
            Theta = -0.5 * Vol ^ 2 * S(i) ^ 2 * Gamma - Int_Rate * S(i) * _
                Delta + Int_Rate * V(i, k - 1) ' 布莱克-舒尔斯方程
            V(i, k) = V(i, k - 1) - dt * Theta
        Next i

        V(0, k) = V(0, k - 1) * (1 - Int_Rate * dt) ' S=0上的边界条件
        V(NAS, k) = 2 * V(NAS - 1, k) - V(NAS - 2, k) ' S=infinity上的边界条件

        If EType = "Y" Then ' 提前执行检查
            For i = 0 To NAS
                V(i, k) = Application.Max(V(i, k), Payoff(i))
            Next i
        End If
    Next k

    Option_Value_3D_US = V ' 输出数组
End Function
```

表 77.3 和图 77.13 是在美式看跌期权上的结果，参数与上面相同。

表 77.3 有限差分代码输出的美式看跌期权价值，股票价格从 60 到 100，时间从 0 到 1

	0.000	0.111	0.222	0.333	0.444	0.556	0.667	0.778	0.889	1.000
60	40.000	40.000	40.000	40.000	40.000	40.000	40.000	40.000	40.000	40.000
70	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000
80	20.000	20.000	20.000	20.000	20.000	20.000	20.000	20.000	20.000	20.000
90	10.000	10.000	10.000	10.060	10.226	10.434	10.653	10.869	11.074	11.267
100	0.000	1.726	2.810	3.549	4.102	4.551	4.935	5.275	5.581	5.859
110	0.000	0.116	0.491	0.909	1.305	1.666	1.994	2.294	2.571	2.827
120	0.000	0.000	0.053	0.174	0.338	0.523	0.714	0.905	1.093	1.276
130	0.000	0.000	0.003	0.025	0.073	0.143	0.230	0.328	0.434	0.544
140	0.000	0.000	0.000	0.002	0.013	0.034	0.067	0.110	0.162	0.221

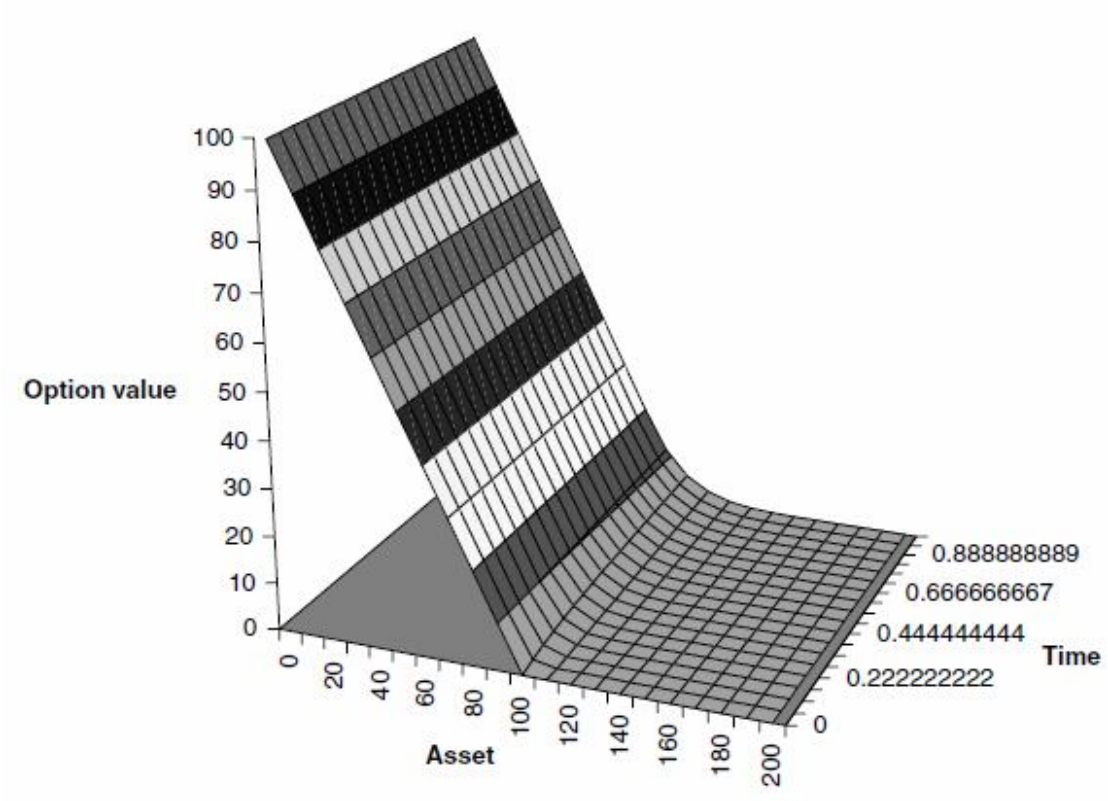


图 77.13 有限差分代码输出的美式看跌期权价值

77. 15 代码#3，二维输出

如果你想要得到整个的$V(S,t)$函数，上面的代码就很好了。但是如果你仅仅是想要知道今天的曲线或价值，那这样做就太没效率了，因为它存储了太多的东西。下面的代码只使用 VOld 和 VNew 来存储信息，这些信息足以去计算当前的价值曲线了。这个代码同时输出对冲系数。

```
Function Option_Value_2D_US(Vol, Int_Rate, PType, Strike, Expiration, _
EType, NAS)
' NAS 是资产价格上的步数
    ReDim S(0 To NAS) As Double ' 资产数组
    ReDim VOld(0 To NAS) As Double ' 一个期权数组
    ReDim VNew(0 To NAS) As Double ' 第二个期权数组
    ReDim Payoff(0 To NAS) As Double ' 收益数组
    ReDim Dummy(0 To NAS, 1 To 6) As Double ' 用来存储输出数据
```

```
dS = 2 * Strike / NAS ' “无穷大”是敲定价格的两倍
dt = 0.9 / Vol ^ 2 / NAS ^ 2 ' 为了稳定性
NTS = Int(Expiration / dt) + 1 ' 时间上的步数
dt = Expiration / NTS ' 为了确保到期日刚好在整数个时间步长上
q = 1

If PType = "P" Then q = -1 ' 测试是看涨还是看跌期权

For i = 0 To NAS
    S(i) = i * dS ' 建立S数组
    VOld(i) = Application.Max(q * (S(i) - Strike), 0) ' 建立收益函数
    Payoff(i) = VOld(i) ' 存储收益函数
    Dummy(i, 1) = S(i) ' Dummy 数组的第一列是股票价格
    Dummy(i, 2) = Payoff(i) ' Dummy 数组的第二列是收益
Next i

For k = 1 To NTS ' 时间循环
    For i = 1 To NAS - 1 ' 资产循环，端点分别处理
        Delta = (VOld(i + 1) - VOld(i - 1)) / 2 / dS ' 中央差分
        Gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / dS / dS _
            ' 中央差分
        Theta = -0.5 * Vol ^ 2 * S(i) ^ 2 * Gamma - Int_Rate * S(i) * _
            Delta + Int_Rate * VOld(i) ' 布莱克-舒尔斯方程
        VNew(i) = VOld(i) - dt * Theta ' 更新期权价值
    Next i

    VNew(0) = VOld(0) * (1 - Int_Rate * dt) ' S=0上的边界条件
    VNew(NAS) = 2 * VNew(NAS - 1) - VNew(NAS - 2) ' S=infinity上的边界条件

    For i = 0 To NAS ' 用新值替换旧值
        VOld(i) = VNew(i)
    Next i

    If EType = "Y" Then ' 提前执行检查
        For i = 0 To NAS
            VOld(i) = Application.Max(VOld(i), Payoff(i))
        Next i
    End If
Next k
```

表 77.4 有限差分代码输出的看涨期权价值和对冲系数

Stock	Payoff	Option	Delta	Gamma	Theta
0	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000
15	0.000	0.000	0.000	0.000	0.000
20	0.000	0.000	0.000	0.000	0.000
25	0.000	0.000	0.000	0.000	0.000
30	0.000	0.000	0.000	0.000	0.000
35	0.000	0.000	0.000	0.000	0.000
40	0.000	0.000	0.000	0.000	−0.001
45	0.000	0.001	0.000	0.000	−0.005
50	0.000	0.003	0.002	0.000	−0.023
55	0.000	0.016	0.005	0.001	−0.086
60	0.000	0.058	0.016	0.003	−0.256
65	0.000	0.173	0.038	0.006	−0.617
70	0.000	0.437	0.078	0.010	−1.235
75	0.000	0.953	0.139	0.015	−2.113
80	0.000	1.832	0.222	0.019	−3.168
85	0.000	3.174	0.321	0.021	−4.255
90	0.000	5.044	0.429	0.022	−5.224
95	0.000	7.461	0.536	0.021	−5.962
100	0.000	10.403	0.636	0.019	−6.425
105	5.000	13.816	0.723	0.016	−6.625
110	10.000	17.628	0.795	0.013	−6.614
115	15.000	21.764	0.852	0.010	−6.459
120	20.000	26.149	0.896	0.007	−6.225
125	25.000	30.722	0.928	0.005	−5.965
130	30.000	35.430	0.951	0.004	−5.712
135	35.000	40.235	0.968	0.003	−5.488
140	40.000	45.107	0.979	0.002	−5.301
145	45.000	50.023	0.986	0.001	−5.153
150	50.000	54.969	0.991	0.001	−5.039
155	55.000	59.935	0.994	0.001	−4.954
160	60.000	64.913	0.997	0.000	−4.892
165	65.000	69.900	0.998	0.000	−4.848
170	70.000	74.892	0.999	0.000	−4.817
175	75.000	79.886	0.999	0.000	−4.796
180	80.000	84.883	0.999	0.000	−4.781
185	85.000	89.881	1.000	0.000	−4.770
190	90.000	94.880	1.000	0.000	−4.761
195	95.000	99.878	1.000	0.000	−4.754
200	100.000	104.877	1.000	0.000	−4.754

```
For i = 1 To NAS - 1
        Dummy(i, 3) = VOld(i) ' Dummy 数组的第三列是期权价值
```

```
        Dummy(i, 4) = (VOld(i + 1) - VOld(i - 1)) / 2 / dS _
            ' Dummy 数组的第四列是Delta
        Dummy(i, 5) = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / dS / dS _
            ' Dummy 数组的第五列是Gamma
        Dummy(i, 6) = -0.5 * Vol ^ 2 * S(i) ^ 2 * Dummy(i, 5) - Int_Rate * _
            S(i) * Dummy(i, 4) + Int_Rate * VOld(i) _
            ' Dummy 数组的第六列是Theta
    Next i

    Dummy(0, 3) = VOld(0)
    Dummy(NAS, 3) = VOld(NAS)
    Dummy(0, 4) = (VOld(1) - VOld(0)) / dS _
        ' 端点的Delta, Gamma 和 Theta需要分开处理
    Dummy(NAS, 4) = (VOld(NAS) - VOld(NAS - 1)) / dS
    Dummy(0, 5) = 0
    Dummy(NAS, 5) = 0
    Dummy(0, 6) = Int_Rate * VOld(0)
    Dummy(NAS, 6) = -0.5 * Vol ^ 2 * S(NAS) ^ 2 * Dummy(NAS, 5) - Int_Rate * _
        S(NAS) * Dummy(NAS, 4) + Int_Rate * VOld(NAS)
    Option_Value_2D_US = Dummy ' 输出数组
End Function
```

表 77.4 展示了这个代码的结果，参数与上面相同，但是资产步数是 40。

77. 16 双线性插值

假设我们有对网格点上的期权价值或其导数的一个估计，我们应如何估计那些网格之间的点上的值呢？最简单的方法是使用一个二维插值方法，它被称为**双线性插值**。这种方法是很容易通过图 77.14 解释。

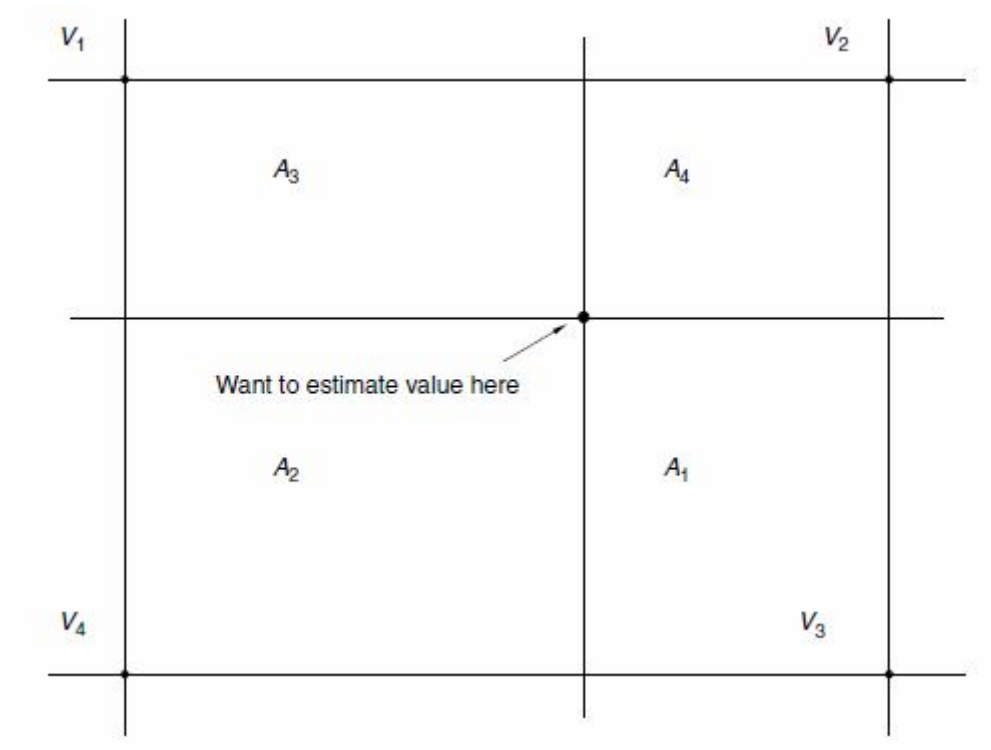


图 77.14 双线性插值

例如，我们要估计图中的内部点上的期权价值。四个近邻点上的期权价值为 V_1、V_2、V_3和V_4，这是在前面章节中的符号的简写。由四个角上的点和内部点所框出的四个矩形面积分别被标注为 A_1、A_2、A_3和 A_4。注意具有相同下标的面积与期权价值所处的位置是相反的。对于内部点的期权价值的近似是:

$$\frac{\sum_{i=1}^{4} A_i V_i}{\sum_{i=1}^{4} A_i}$$

77. 17 逆风差分

如果我们在期权价值对资产价格的一阶导数的近似中使用单边差分以替代中央差分的话，约束（77.4）是可以被规避的。如我在第 6 章所述，对 S 的一阶导数表现为漂移项。这个漂移具有一个与漂移项本身关联的特定方向。例如当 t 递减，也就是时间从到期日向前逆推的时候，漂移指向的方向是较小的 S。从某

种意义上来说，这使得（对于漂移方向来说）前向的资产价格是一个用起来更好的变量。而数值格式可以通过一个单边差分来利用这一特性。这也是布莱克-舒尔斯方程所面临的状况。更一般地，我们对方程中的 Delta 的近似方法可以依赖于b的符号，例如如下的格式:

$$\text{如果} b(S,t)\geq 0\text{，则}\frac{\partial V}{\partial S}(S,t)=\frac{V_{i+1}^k-V_i^k}{\delta S}$$

以及

$$b(S,t)<0\text{，则}\frac{\partial V}{\partial S}(S,t)=\frac{V_i^k-V_{i-1}^k}{\delta S}$$

这将消除（77.4）对时间步长的限制，从而提升稳定性。但是由于这些单边差分的精度只有$O(\delta S)$，所以这个数值方法是不够精确的。

单边差分的方向取决于系数b的符号，这种数值格式被称为逆风差分。一个技术上的小改进在于对函数b取值的选择上:

$$\text{如果} b(S,t)\geq 0\text{，则}b(S,t)\frac{\partial V}{\partial S}(S,t)=b_{i+\frac{1}{2}}^k\frac{V_{i+1}^k-V_i^k}{\delta S}$$

以及

$$b(S,t)<0\text{，则}b(S,t)\frac{\partial V}{\partial S}(S,t)=b_{i-\frac{1}{2}}^k\frac{V_i^k-V_{i-1}^k}{\delta S}$$

注意我是如何使用b的格点间值的。

下面的 Visual Basic 代码段使用依赖于漂移项符号的单边差分。这个代码段可以用于利率产品，因为它是用来求解如下方程的:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma(r)^2\frac{\partial^2 V}{\partial r^2}+\left(\mu(r)-\lambda(r)\sigma(r)\right)\frac{\partial V}{\partial r}-rV=0$$

注意这里设定:

$$\sigma(r)=\texttt{ Volatility(IntRate(i))}$$

和

$$\mu(r)-\lambda(r)\sigma(r)=\texttt{RiskNeutralDrift(IntRate(i))}$$

这个代码段仅仅涉及时间步进部分。在它上边应当有变量声明和收益函数的设定，在它的下边应当有输出语句。它在 `i = 0` 和 `i = NoIntRateSteps` 的位置上没有实现任何的边界条件。这将取决于被估值合约的具体状况而定。

```
For i = 1 To NoIntRateSteps - 1
    If RiskNeutralDrift(IntRate(i)) > 0 Then
        Delta(i) = (VOld(i + 1) - VOld(i)) / dr
        RNDrift = RiskNeutralDrift(IntRate(i) + 0.5 * dr)
    Else
        Delta(i) = (VOld(i) - VOld(i - 1)) / dr
        RNDrift = RiskNeutralDrift(IntRate(i) - 0.5 * dr)
    End If

    Gamma(i) = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / (dr * dr)
    VNew(i) = VOld(i) + Timestep * (0.5 * Volatility(IntRate(i)) * _
        Volatility(IntRate(i)) * Gamma(i) + RNDrift * Delta(i) - _
        IntRate(i) * VOld(i))
Next i
```

如果想要在单边差分格式下取得与中央差分同样的精度 $O\left(\delta S^2\right)$，你可以使用第 77.6 节中讨论的近似方法。

我们已经看到，显式有限差分法需要承受对于网格步长的限制。显式法从原理上来说类似于二叉树方法，而上面所说的限制可以通过风险中性概率来解释。项 A，B 和 C 是与到达 $i-1$，i 和 $i+1$ 点上的风险中性概率相关的。不稳定的情况等同于这些概率中的某个取到了负值，而我们是不能允许负概率的存在的。一些更复杂的数值方法可以不再承受这些限制，接下来我将开始讨论它们。

显式法的优点

- 易于编程而且很难出错
- 如果它进入不稳定的状态，通常是很容易发现的
- 它善于处置系数依赖于资产价格和时间的状况

- 易于引入较为精确的单边差分

显式法的缺点

- 它在时间步长上受到约束，所以它比其它的数值格式要慢一些

77. 18 小结

扩散方程已经出现了很长一段时间。解决扩散方程的数值方法也已经有很长时间的历史了——虽然不如方程本身那样长，但肯定比布莱克-舒尔斯方程和二叉树方法的历史长得多。这意味着，在一般抛物型方程的高效解法中有很多的学术文献。我向您介绍了显式法方面的主题，在下一章中，我向你介绍更为复杂的数值方法。

扩展阅读

- 对于一般的数值方法，参见Johnson & Riess (1982)和 Gerald & Wheatley (1992)。
- 有限差分法在金融中的首次应用应归于Brennan & Schwartz (1978)。其在利率模型中的应用参见Hull & White (1990)。
- 一本金融数值方法方面的杰出著作是Ahmad (2006)。

第 78 章　单因素模型有限差分方法进阶

本章内容:

- 隐式有限差分方法（包括克朗克-尼科尔森法）
- 道格拉斯格式
- 理查德森外推
- 美式期权上的应用
- 奇异期权

78. 1 简介

我们在单因素数值分析方面继续，开始讨论在编程上稍难一些的隐式法。其卓越的稳定性抵消了额外的那一点复杂性。我还会展示如何去扩展有限差分法以便其处置提前执行和路径依赖合约。

78. 2 隐式有限差分法

全隐式方法使用如图 78.1 所示的点集去计算期权价值。这个格式表面上看起来和显式法计算期权价值很像，但它的 Delta 和 Gamma 现在是在 $K+1$ 的时间点上。格点上期权价值间的关系可简单地写出:

$$\frac{V_i^k - V_i^{k+1}}{\delta t}$$

$$+a_i^{k+1}\left(\frac{V_{i+1}^{k+1}-2V_i^{k+1}+V_{i-1}^{k+1}}{\delta S^2}\right)$$

$$+b_i^{k+1}\left(\frac{V_{i+1}^{k+1}-V_{i-1}^{k+!}}{2\delta S}\right)$$

$$+c_i^{k+1}V_i^{k+1}=O(\delta t,\delta S^2)$$

（系数a，b和c是在时间点k还是$k+1$上被估值并不重要。）这个方法的精度依旧是$O(\delta t,\delta S^2)$。

这个式子可被改写为:

$$A_i^{k+1}V_{i-1}^{k+1}+(1+B_i^{k+1})V_i^{k+1}+C_i^{k+1}V_{i+1}^{k+1}=V_i^k \qquad (78.1)$$

这里:

$$A_i^{k+1}=v_1a_i^{k+1}-\frac{1}{2}v_2b_i^{k+1}$$

$$B_i^{k+1}=-2v_1a_i^{k+1}+\delta tc_i^{k+1}$$

以及:

$$C_i^{k+1}=v_1a_i^{k+1}+\frac{1}{2}v_2b_i^{k+1}$$

并且:

$$v_1=\frac{\delta t}{\delta S^2}\text{ 和 }v_2=\frac{\delta t}{\delta S}$$

同样的，方程（78.1）在$i=0$和$i=I$上无效，边界条件提供了剩下的两个方程。

事实上这个格式和显式有限差分格式有很大的差别，其间的两个主要差别包括方法的稳定性和求解过程。

隐式法不再有对时间步长的限制。资产步长变小和时间步长变大不会再导致稳定性的问题。

对差分方程的求解不再那么直接。从V_i^k中解出V_i^{k+1}需要求解一个线性方程组。每一个V_i^{k+1}都和它在同一时间点上的两个邻点直接相关，因而，也就和这个

时间点上的每一点间接相关。

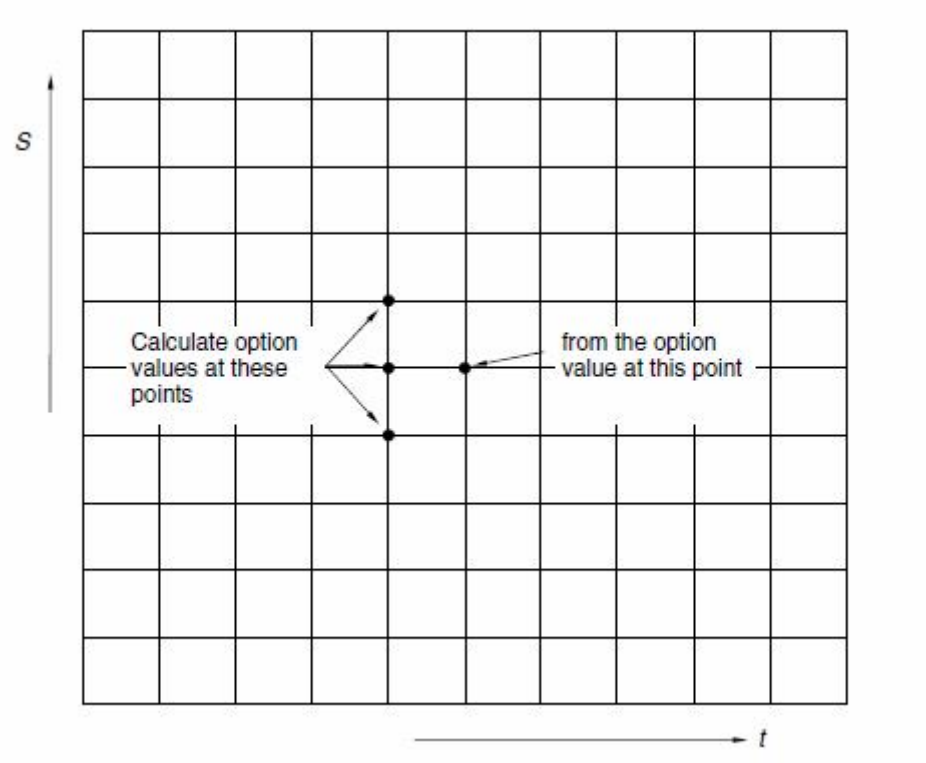


图 78.1 全隐式方法中期权价值间的关系

我不会再对全隐式法进行更多的批评，因为这个方法可以显著地提升性能，而在计算上却几乎没有额外的负担。

78. 3 克朗克-尼科尔森法

克朗克-尼科尔森法可以被看做是显式法和全隐式法的一个折衷。它使用在图 78.2 上所示的六个点：

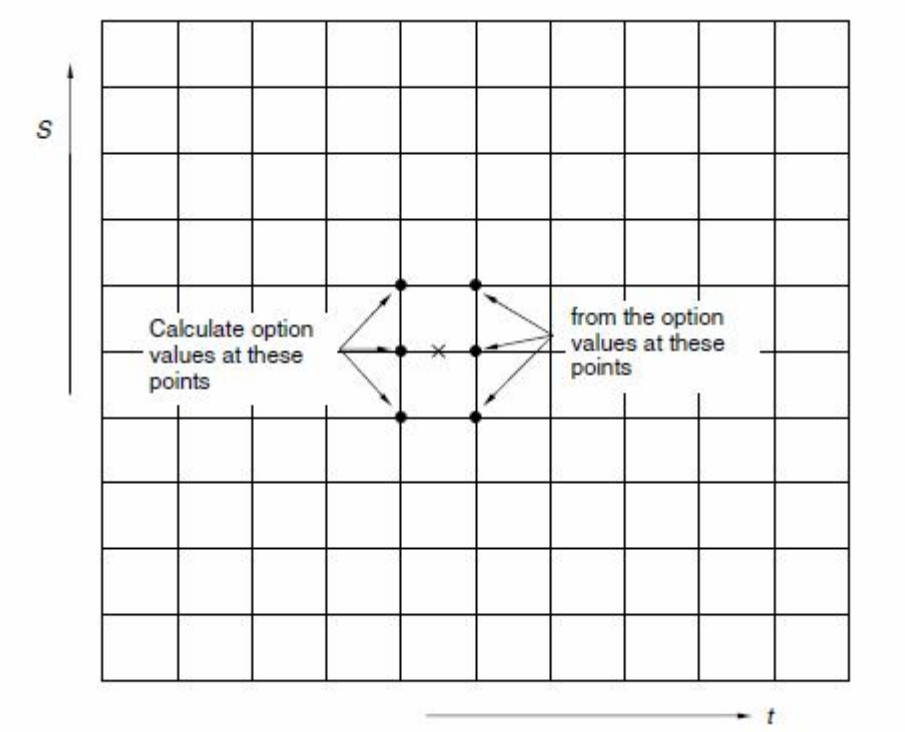


图 78.2 克朗克-尼科尔森法中的期权价值关系

克朗克-尼科尔森格式是:

$$\frac{V_i^k - V_i^{k+1}}{\delta t}$$

$$+\frac{a_i^{k+1}}{2}\left(\frac{V_{i+1}^{k+1} - 2V_i^{k+1} + V_{i-1}^{k+1}}{\delta S^2}\right) + \frac{a_i^k}{2}\left(\frac{V_{i+1}^k - 2V_i^k + V_{i-1}^k}{\delta S^2}\right)$$

$$+\frac{b_i^{k+1}}{2}\left(\frac{V_{i+1}^{k+1} - V_{i-1}^{k+1}}{2\delta S}\right) + \frac{b_i^k}{2}\left(\frac{V_{i+1}^k - V_{i-1}^k}{2\delta S}\right)$$

$$+\frac{1}{2}c_i^{k+1}V_i^{k+1} + \frac{1}{2}c_i^k V_i^k = O(\delta t^2, \delta S^2)$$

它被改写为

$$-A_i^{k+1}V_{i-1}^{k+1} + (1 - B_i^{k+1})V_i^{k+1} - C_i^{k+1}V_{i+1}^{k+1} = A_i^k V_{i-1}^k + (1 + B_i^k)V_i^k + C_i^k V_{i+1}^k \qquad (78.2)$$

这里:

$$A_i^k = \frac{1}{2}v_1 a_i^k - \frac{1}{4}v_2 b_i^k$$

$$B_i^k = -v_1 a_i^k + \frac{1}{2}\delta t c_i^k$$

以及:

$$C_i^k = \frac{1}{2}v_1 a_i^k + \frac{1}{4}v_2 b_i^k$$

这个方程只对区间$1 \le i \le I$有效，边界条件再次提供了另外两个方程。它们比在显式法中的那些方程处置起来要难一些，我们等下要深入讨论。

虽然这些方程看上去有点混乱，并且难以求解，但这种方法的优点在于其稳定性和准确性。如同全隐式方法一样，该方法在收敛性方面并没有对时间步长做出特殊的限制。更好的是，该方法比迄今为止的另两种方法更为准确。这个方法的误差为$O\left(\delta t^2, \delta S^2\right)$。所以，现在我们可以使用更大的时间步长，而依旧可获得准确的解。通过在点$(S, t+\frac{1}{2}\delta t)$(即图 78.2 上的 X 点)上展开方程(78.2)，可以看到误差对时间步长的关系现在是$O(\delta t^2)$了。

克朗克-尼科尔森法可以被写为矩阵格式:

$$\begin{pmatrix} -A_1^{k+1} & 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & . \\ 0 & -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & 0 \\ . & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & -C_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_0^{k+1} \\ V_1^{k+1} \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \\ V_I^{k+1} \end{pmatrix}$$

$$=\begin{pmatrix} A_1^k & 1+B_1^k & C_1^k & 0 & . & . & . \\ 0 & A_2^k & 1+B_2^k & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1+B_{I-2}^k & C_{I-2}^k & 0 \\ . & . & . & 0 & A_{I-1}^k & 1+B_{I-1}^k & C_{I-1}^k \end{pmatrix} \begin{pmatrix} V_0^k \\ V_1^k \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^k \\ V_I^k \end{pmatrix} \qquad (78.3)$$

这两个矩阵有 $I-1$ 行和 $I+1$ 列。这是对个 $I-1$ 方程和 $I+1$ 个未知数的一种反映。

两个剩下的方程来自于边界条件。利用这些条件，我们可以将方程（78.3）转换为一个包含方阵的方程。最终的目的是将方程写做:

$$\mathrm{M}_L^{k+1}\mathrm{v}^{k+1}+\mathrm{r}^k=\mathrm{M}_R^k\mathrm{v}^k$$

其中方阵 M_L^{k+1} 和 M_R^k 以及向量 r^k 是已知的。并且关于边界条件的细节都已经被包含其中。

我会根据边界条件的实现形式将情况分为三种进行考虑。我只讨论在 $i=0$ 上的边界条件，$i=I$ 上的边界条件可以通过类似的方式处置。

78. 3. 1 第一类边界条件: 已知 V_0^{k+1}

有时我们会知道期权在边界 $i=0$ 或 $i=I$ 上的价值。例如，如果我们有一个欧式看跌期权，那么我们知道 $V(0,t)=Ee^{-r(T-t)}$。将其转换为 $V_0^{k+1}=Ee^{-r(k+1)\delta t}$，我们可以将:

$$\begin{pmatrix} -A_1^{k+1} & 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & . \\ 0 & -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & 0 \\ . & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & -C_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_0^{k+1} \\ V_1^{k+1} \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \\ V_I^{k+1} \end{pmatrix}$$

写为:

$$\begin{pmatrix} 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & \\ -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & \\ & 0 & . & . & . & 0 \\ & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} \\ & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} \end{pmatrix} \begin{pmatrix} V_1^{k+1} \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \end{pmatrix}$$

$$+ \begin{pmatrix} -A_1^{k+1} V_0^{k+1} \\ 0 \\ 0 \\ . \\ . \\ 0 \\ . \end{pmatrix} = \mathrm{M}_L^{k+1} \mathrm{v}^{k+1} + \mathrm{r}^k$$

我们所做的所有事情就是将矩阵的顶行和底行乘了出来。这里矩阵 M_L 是一个 $I-1$ 阶的方阵，而 $I-1$ 阶向量 r^k 只在顶部（以及底部）有非零元素。这些非零元素是完全确定的，因为它们只依赖于函数 A 和预设的边界值 V。

78. 3. 2 第二类边界条件：已知V_0^{k+1}和V_1^{k+1}的关系

如果我们有一个障碍期权，并且网格点与障碍不相吻合，我们必须使用第77.10节中提到的近似：

$$V_0^{k+1}=\frac{1}{\alpha}(f-(1-\alpha)V_1^{k+1})$$

在一个“下”障碍中，这个关系是V_0^{k+1}与V_1^{k+1}之间的，而在一个“上”障碍中，这个关系是V_I^{k+1}和V_{I-1}^{k+1}之间的。另外，或许我们会知道在大的或小的资产价格S上的期权价值的斜率，这给了我们一个梯度边界条件，其形式也是如上所述的。如果我们使用单边差分去近似导数的话，边界条件也可以写成最后一个网格点与倒数第二个网格点间的关系。一般地，假设我们有：

$$V_0^{k+1}=a+bV_1^{k+1}$$

这样方程（78.3）的左边可以写作：

$$\begin{pmatrix} 1-B_1^{k+1} & -C_1^{k+1} & 0 & . & . & \\ -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & \\ & 0 & . & . & . & 0 \\ & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} \\ & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} \end{pmatrix}\begin{pmatrix} V_1^{k+1} \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \end{pmatrix}+\begin{pmatrix} -A_1^{k+1}(a+b)V_0^{k+1} \\ 0 \\ 0 \\ . \\ . \\ 0 \\ . \end{pmatrix}$$

同样的，这是将矩阵的顶行和底行乘出来后得出的结果。将最右边的向量中的项重新归入矩阵，我们得到：

$$\begin{pmatrix} 1-B_1^{k+1}-bA_1^{k+1} & -C_1^{k+1} & 0 & . & . & & \\ -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & & \\ & 0 & . & . & . & 0 & \\ & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & \\ & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & \end{pmatrix} \begin{pmatrix} V_1^{k+1} \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \end{pmatrix} + \begin{pmatrix} -aA_1^{k+1} \\ 0 \\ 0 \\ . \\ . \\ 0 \\ . \end{pmatrix}$$

再次得到如下的形式:

$$\mathrm{M}_L^{k+1}\mathrm{v}^{k+1}+\mathrm{r}^k$$

但是 M_L^{k+1} 和 r^k 与前面的不同了。这里矩阵和向量都是已知的。

78. 3. 3 第三类边界条件: 已知 $\partial^2 V/\partial S^2=0$

更复杂，但依旧易于处理的，是边界条件:

$$\frac{\partial^2 V}{\partial S^2}=0$$

这种情况是特别有用，因为它不依赖于合约的类型，只要该合约的回报函数与基础资产的价格大致呈线性即可。在中央差分形式下，这一条件可写做:

$$V_0^{k+1}=2V_1^{k+1}-V_2^{k+1}$$

因此，这里我们可以将方程（78.3）的左边写为:

$$\begin{pmatrix} 1-B_1^{k+1}-2A_1^{k+1} & -C_1^{k+1}+A_1^{k+1} & 0 & . & . & & \\ -A_2^{k+1} & 1-B_2^{k+1} & . & . & . & & \\ & 0 & . & . & . & 0 & \\ & . & . & . & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} & \\ & . & . & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} & \end{pmatrix} \begin{pmatrix} V_1^{k+1} \\ . \\ . \\ . \\ . \\ . \\ V_{I-1}^{k+1} \end{pmatrix}$$

在这里我们不需要额外的引入已知向量。我们矩阵方程的左边依旧是:

$$\mathrm{M}_L^{k+1}\mathrm{v}^{k+1}+\mathrm{r}^k$$

但此处矩阵 M_L^{k+1} 与前面不同，而向量 r^k 为零。

读者可以尝试将其它的边界条件引进来，这种练习是有益的。

78. 3. 4 矩阵方程

不管我们有什么样的边界条件，克朗克-尼科尔森格式在考虑边界条件后，是:

$$\mathrm{M}_L^{k+1}\mathrm{v}^{k+1}=\mathrm{M}_R^{k}\mathrm{v}^{k}-\mathrm{r}^k \qquad (78.4)$$

这里 r^k 是一个具有 $I-1$ 个项: V_i^k 的列向量（$i=1$ 是第一行），M_L^{k+1} 和 M_R^{k} 分别是一个 $I-1$ 阶的方阵。

记住 v^k 我们是知道的，（78.4）式的右边，矩阵的乘法和向量的加法很容易完成，但我们应当如何去计算 v^{k+1}？这个向量方程具有 $I-1$ 个方程和 $I-1$ 个未知数，是一个线性联立方程组。从原理上来说，可以求出 M_L^{k+1} 的逆矩阵，然后可得:

$$\mathrm{v}^{k+1}=\left(\mathrm{M}_L^{k+1}\right)^{-1}\left(\mathrm{M}_R^{k}\mathrm{v}^{k}-\mathrm{r}^k\right)$$

但是矩阵求逆是非常耗时的，从计算的角度来说效率非常低。两个更为有效的求解（78.4）方法被称为 LU 分解和超松弛迭代法。

78. 3. 5 LU 分解

矩阵 M_L^{k+1} 具有特殊的形式，它是**三对角线矩阵**，只有在矩阵的对角线和上下

副对角线上才有非零元素。这意味着可以很容易的将这个矩阵分解为另外两个矩阵的乘积，其中的一个在对角线和下副对角线上有非零元素，另一个在对角线和上副对角线上有非零元素。我们相应地将其称为L矩阵和U矩阵。在本节的其余部分我们将省略一切不重要的上标及下标。因此我们可以写下:

$$M = LU \tag{78.5}$$

我将首先展示如何进行**LU分解**，然后是如何用这个方法去解出方程(78.4)。LU 分解是“直接法”的一个特例，其目的是一次性的解出方程的精确解。唯一的误差来源于舍入误差。

方程（78.5）是:

$$\begin{pmatrix} 1-B_1 & -C_1 & 0 & . & . & . & . \\ -A_2 & 1-B_2 & -C_2 & 0 & . & . & . \\ 0 & -A_3 & 1-B_3 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1-B_{I-3} & -C_{I-3} & 0 \\ . & . & . & 0 & -A_{I-2} & 1-B_{I-2} & -C_{I-2} \\ . & . & . & . & 0 & -A_{I-1} & 1-B_{I-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & . & . & . & 0 \\ l_2 & 1 & 0 & 0 & . & . & . \\ 0 & l_3 & 1 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & 1 & 0 & 0 \\ . & . & . & 0 & l_{I-2} & 1 & 0 \\ . & . & . & . & 0 & l_{I-1} & 1 \end{pmatrix} \begin{pmatrix} d_1 & u_1 & 0 & . & . & . & 0 \\ 0 & d_2 & u_2 & 0 & . & . & . \\ 0 & 0 & d_3 & . & . & . & . \\ . & 0 & . & . & . & 0 & . \\ . & . & . & . & d_{I-3} & u_{I-3} & 0 \\ . & . & . & 0 & 0 & d_{I-2} & u_{I-2} \\ . & . & . & . & 0 & 0 & d_{I-1} \end{pmatrix}$$

这里，不失一般性的，我们可以将矩阵L的对角元素都设为1。我将请读者自行证明:

$$d_1 = 1 - B_1$$

以及:

$$l_i d_{i-1} = -A_i,\quad u_{i-1} = -C_{i-1} \text{和} d_i = 1 - B_i - l_i u_{i-1} \text{当} 2 \le i \le I-1 \qquad (78.6)$$

注意我们是如何从$i=1$到$i=I$进行推导的。

在下面的这段Visual Basic代码中，矩阵M的对角线，上副对角线，下副对角线被存储于数组 Diag()，SuperDiag()和 SubDiag()中。它们的长度都是NoElements。代码计算d，u和l。

```
Dim d(1 To NoElements) As Double
Dim u(1 To NoElements) As Double
Dim l(1 To NoElements) As Double
d(1) = Diag(1)

For i = 2 To NoElements
    u(i - 1) = SuperDiag(i - 1)
    l(i) = SubDiag(i) / d(i - 1)
    d(i) = Diag(i) - l(i) * SuperDiag(i - 1)
Next i
```

对矩阵的分解是第一步，现在我们使用这个分解对方程（78.4）进行求解。

假设我们已经将方程（78.4）的右端整理完毕，从而给出:

$$Mv=q$$

（所有的上标和下标都被省略），我们可以写出

$$LUv=q \qquad (78.7)$$

向量q包含在时间点k上关于旧的期权价值和边界条件的所有细节信息。

我们分两步去求解方程（78.7），首先解出:

$$Lw=q$$

于是:

$$Uv=w$$

这就是所有的工作。

第一步给出:

$$w_1 = q_1$$

以及

$$w_i = q_i - l_i w_{i-1} \text{ 当 } 2 \le i \le I-1$$

我们必须从 $i=2$ 到 $i=I-1$ 逐步递推。第二步使用反向的递推从 $i=I-2$ 到 $i=1$：

$$v_{I-1} = \frac{w_{I-1}}{d_{I-1}}$$

以及

$$v_i = \frac{w_i - u_i v_{i+1}}{d_i} \text{ 当 } I-2 \ge i \ge 1$$

以下这段 Visual Basic 代码使用刚才计算的 l 和 d，以及方程右边的向量 q，以计算 v。

```
Dim v(1 To NoElements) As Double
Dim w(1 To NoElements) As Double
w(1) = q(1)

For i = 2 To NoElements
    w(i) = q(i) - l(i) * w(i - 1)
Next i

v(NoElements) = w(NoElements) / d(NoElements)

For i = NoElements - 1 To 1 Step -1
    v(i) = (w(i) - u(i) * v(i + 1)) / d(i)
Next i
```

如果我们的矩阵方程是时不变的，也就是说，a，b 和 c 仅仅是 S 的函数，那么我们可以只进行一次 LU 分解并存储必要的结果。因此这个方法在求解系数仅与 S 相关的经典布莱克-舒尔斯方程时工作较好。不过现今它们通常都是时间依赖的。无论在股票类问题还是在固定收益类问题中，漂移项和波动率项都具有一定的项结构。这意味着系数们都是具有很强的时间依赖的。这种时间依赖意味着我们在每一时间步上都要重新进行 LU 分解，这将显著地降低计算速度。

LU 分解法的优点

- 很快
- 当矩阵不具有时间依赖时，只需要进行一次分解计算

LU 分解法的缺点

- 不能直接用于美式期权
- 在矩阵具有时间依赖时，对于每一时间步都要进行一次分解操作

78. 3. 6 超松弛迭代法，SOR

我们现在讨论“间接方法”的一个例子。在这个方法中我们使用迭代来对方程求解。虽然最终的解永远不会是完全精确的，但我们可以得到任何我们想要的精确度。只要这种算法能够很快地得出结果，那它就是足够好的。间接方法适用于更广泛的问题——我们的左边矩阵无须是三对角线型的——所以我将更具一般性的描述这个思想。这意味着我将丢弃所有的上/下标，而符号也会略有改变。

考虑矩阵方程中的矩阵 M :

$$Mv=q$$

具有元素 M_{ij}，则方程组可以写做:

$$M_{11}v_1 + M_{12}v_2 + \cdots + M_{1N}v_N = q_1$$

$$M_{21}v_1 + M_{22}v_2 + \cdots + M_{2N}v_N = q_2$$

$$\cdots$$

$$M_{N1}v_1 + M_{N2}v_2 + \cdots + M_{NN}v_N = q_N$$

其中 N 是方程的数量。也是矩阵的阶数。

将它进行重写:

$$M_{11}v_1 = q_1 - (M_{12}v_2 + \cdots + M_{1N}v_N)$$

$$M_{22}v_2 = q_2 - (M_{21}v_1 + \cdots + M_{2N}v_N)$$

$$\cdots$$

$$M_{NN}v_N = q_N - (M_{N1}v_1 + \cdots)$$

这个方程组可以很容易地进行迭代求解:

$$v_1^{n+1} = \frac{1}{M_{11}}\left(q_1 - \left(M_{12}v_2^n + \cdots + M_{1N}v_N^n\right)\right)$$

$$v_2^{n+1} = \frac{1}{M_{22}}\left(q_2 - \left(M_{21}v_1^n + \cdots + M_{2N}v_N^n\right)\right)$$

$$\cdots$$

$$v_N^{n+1} = \frac{1}{M_{NN}}\left(q_N - \left(M_{N1}v_1^n + \cdots\right)\right)$$

这里上标 n 表示迭代的步数，而不是时间的步数。这个迭代过程从初始猜想 v^0 开始。(在实际解题时我们通常使用前一个时间步上的期权价值作为当前时间步上的初始猜想。) 这个方法被称为**雅可比方法**。

我可以将矩阵 M 写为对角阵 D、对角线为 0 的上三角矩阵 U 以及对角线为 0 的下三角矩阵 L 的和:

$$\mathrm{M} = \mathrm{D} + \mathrm{U} + \mathrm{L}$$

我可以用这个表示法很漂亮地改写雅可比方法与其他方法。这里雅可比方法改写为:

$$\mathrm{v}^{n+1} = \mathrm{D}^{-1}\left(\mathrm{q} - \left(\mathrm{U} + \mathrm{L}\right)\mathrm{v}^n\right)$$

当实际执行雅可比方法时，我们可以首先获取某些 v_i^{n+1} 的值。在**高斯一赛德**

尔方法中，我们可以在新的值算出来后马上就使用它们，这个方法被写为:

$$v_i^{n+1}=\frac{1}{M_{ii}}\left(q_i-\sum_{j=1}^{i-1}M_{ij}v_j^{n+1}-\sum_{j=i+1}^{N}M_{ij}v_j^{n}\right)$$

注意右边的一些项有上标$n+1$，这些的值之前已被算了出来，但与正在计算的值属于同一迭代步。这一方法可以更简洁地写为:

$$\mathrm{v}^{n+1}=\left(\mathrm{D}+\mathrm{L}\right)^{-1}\left(\mathrm{q}-\mathrm{U}\mathrm{v}^{n}\right)$$

当矩阵M是来源于对一个抛物型方程的有限差分离散时（对于大部分金融问题都是如此）上面的迭代方法通常从一个侧面逼近精确解。这意味着当n增加时，$v_i^{n+1}-v_i^n$会保持相同的符号，其具体的正负取决于初始猜想在精确解的哪一侧。这一特性在**超松弛迭代法（SOR）**中被用来加速收敛。

这个方法可以写做:

$$v_i^{n+1}=v_i^{n}+\frac{\omega}{M_{ii}}\left(q_i-\sum_{j=1}^{i-1}M_{ij}v_j^{n+1}-\sum_{j=i+1}^{N}M_{ij}v_j^{n}\right)$$

同样的，v_i的新值在计算出来后立刻就被使用，但是现在，被称为**加速因子**或**超松弛参数**的因子ω被包含了进来，这个参数必须取值在1到2之间，它可以加速逼近真实解的收敛过程。

在矩阵格式下我们可以写出:

$$\mathrm{v}^{n+1}=\left(\mathrm{I}+\omega\mathrm{D}^{-1}\mathrm{L}\right)^{-1}\left(\left(\left(1-\omega\right)\mathrm{I}-\omega\mathrm{D}^{-1}\mathrm{U}\right)\mathrm{v}^{n}\right)+\omega\mathrm{D}^{-1}\mathrm{q}\Big)$$

这里I是单位矩阵。

下面是一个 Visual Basic 代码段，它可以用来在给定矩阵M和向量 q 的情况下计算 v。这个代码假设M是一个三对角线矩阵，其对角线为 MDiag(i)，上副对角线为 MSuperDiag(i)，下副对角线为 MSubDiag(i)，迭代过程一直持续到均方差 Error 小于预设的公差 tol 为止。注意我在跟踪需要的迭代步数，它是 NoIts。

```
Dim q(1 To N) As Double
Dim v(0 To N + 1) As Double
Dim temp(1 To N) As Double
```

```
Dim Diag(1 To N) As Double
Dim SuperDiag(1 To N + 1) As Double
Dim SubDiag(0 To N) As Double
SuperDiag(N + 1) = 0
SubDiag(0) = 0
NoIts = 0

While Err > tol
    Err = 0
    For i = 1 To N
        temp(i) = v(i) + omega * (q(i) - SuperDiag(i) * v(i + 1) - Diag(i) * _
            v(i) - SubDiag(i) * v(i - 1)) / Diag(i)
        Err = Err + (temp(i) - v(i)) * (temp(i) - v(i))
                        v(i) = temp(i) ' 计算出来后马上就被使用
    Next i
    NoIts = NoIts + 1
Wend
```

78. 3. 7 ω的最优选择

设v是精确解，则误差为$e^n=v^n-v$，它满足:

$$e^{n+1}=\left(I+\omega D^{-1}L\right)^{-1}\left((1-\omega)I-\omega D^{-1}U\right)e^n$$

SOR 方法的收敛性需要以下条件的保证: SOR 矩阵

$$\left(I+\omega D^{-1}L\right)^{-1}\left((1-\omega)I-\omega D^{-1}U\right)$$

的特征值的最大模，也就是所谓的**谱半径**小于1。理论上存在一个加速因子ω的最优值，使得谱半径最小，从而使 SOR 方法的收敛速度最快。ω最优值搜索方面的深入讨论参见 Smith (1985)。在实践中，有一个简单的迭代法来搜索ω的最优值。其做法如下:

首先设$\omega=1$，在第一个时间步达到所需精度并结束后，计算其所需要的迭代步数。也就是 Visual Basic 代码中的 NoIts。对于下一时间步，给ω上增加一个微小增量，比如0.05。再次记录达成需要精度时的迭代步数。如果这个数

字比第一个时间步上的迭代步数少，则在第三个时间步上再将ω增加0.05。对于每一个时间步都对ω进行累加，直到迭代步数开始增加为止，我们可以选择那个令迭代步数最少的ω作为其最优值。如果 SOR 矩阵是时不变的，那么你不需要再对参数进行修改。如果这个矩阵具有强烈的时间依赖，那么你可能需要重新测试ω的最优值。

SOR 方法的优点

- 比 LU 分解更易编程
- 可以很容易地应用于美式期权

SOR 方法的缺点

- 在处理欧式期权时比 LU 分解稍慢一点

78. 4 有限差分法的比较

现在我们已经学习了显式的，全隐式的以及克朗克-尼科尔森有限差分方法。我们可以从实际中，而不仅是从理论上去考察这些方法的误差。下面的两张图展示了三种格式在求解一个平值欧式看涨期权时的误差。这个期权的敲定价格是20，到期日三个月，波动率20%且利率为5%。图78.3展示了误差与资产价格步长之间关系的双对数图，这三条线，如期望的，其斜率都是2，也就是说，这三种方法的误差都随δS^2减少。图78.4中展示了误差与时间步长之间关系的双对数图。现在我们可以看到三种方法的不同，显式法和全隐式法的误差随δt减少，而克朗克-尼科尔森法更好些，它的误差随δt^2减少。

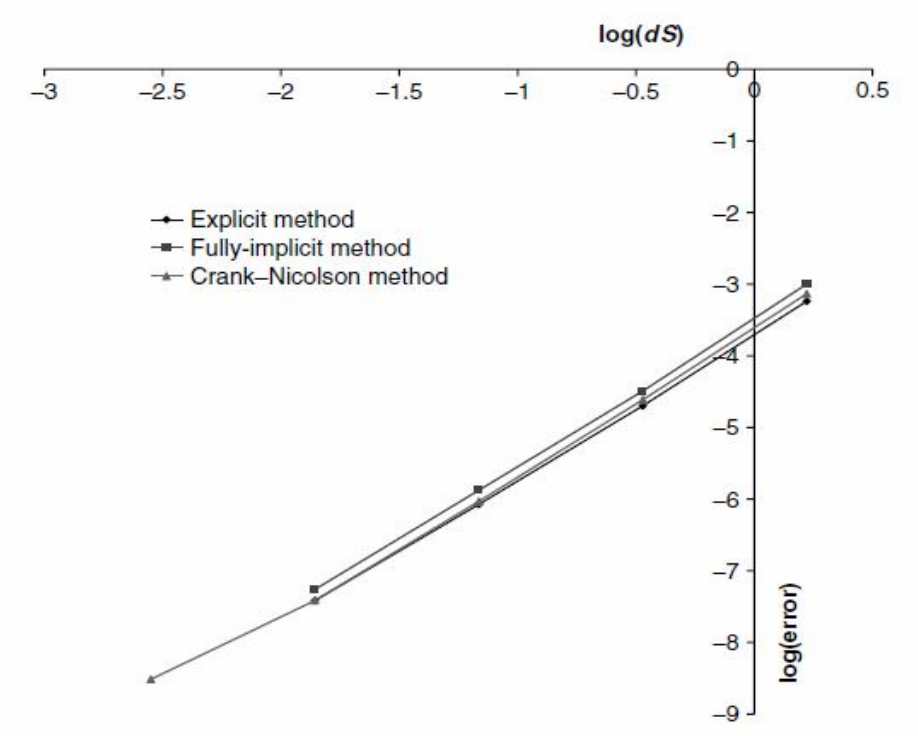


图 78.3 三种差分格式下 log(Error) 对 log(δS) 的函数关系

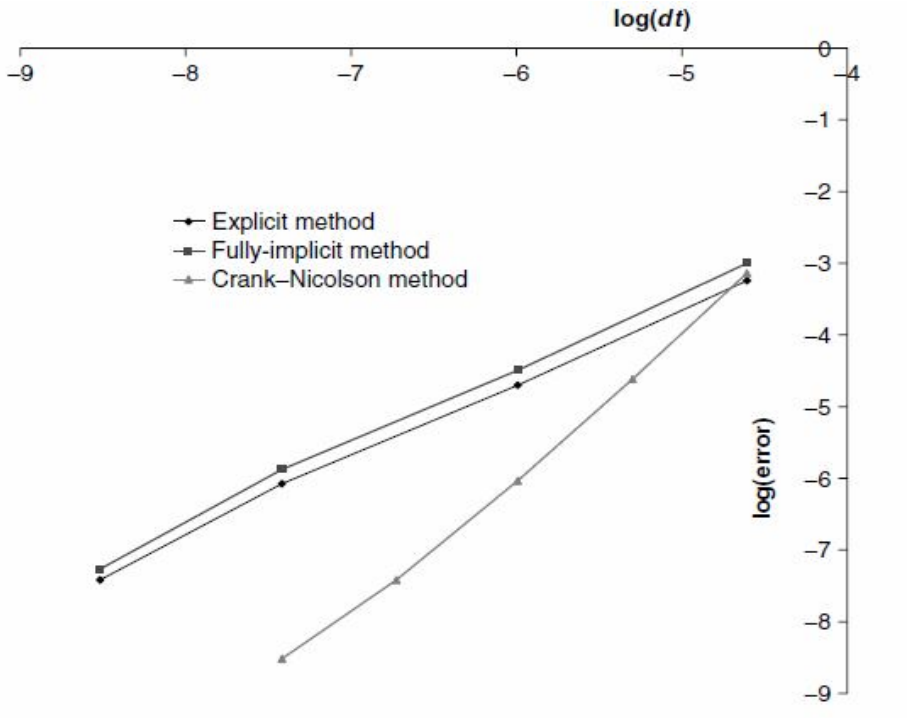


图 78.4 三种差分格式下 log(Error) 对 log(δt) 的函数关系

78. 5 其它方法

以上所讨论的方法仅仅是过去一个时期计算流体力学高速发展所提炼出的诸多格式中的三个例子。其它的格式可能不止使用三个资产点和两个时间点。对于任何的方法，都存在如下问题:

- 在这个方法中，误差和δS，δt间的关系是什么?
- 对时间步长和资产价格步长的限制是什么?
- 所否可以快速地对所得的差分方程进行求解?

我们需要回答这三个问题以确定针对特定问题哪个方法才是最好的。还有另外一个问题，它可能是重要的，也可能不是，这取决于你打算在编程中倾注多少精力:

- 方法是否具有足够的灵活性，使其足以应对系数和边界条件的改变。

也就说，在合约发生轻微的变动时，你是不得不从打草稿开始重新进行你的工作，还是只需简单地改写一个子程序就足以应对新的合约?一个重要的例子是提前执行条款，我们上面讨论的方法能否处理美式期权?我等下会讨论这个问题。但首先，我想讨论有限差分思想的一点扩展，并以一个非常简单的，但却可以加速收敛的技术收尾。它并不总是可用，但它值得尝试，因为它并不引入额外的编程工作。

78. 6 道格拉斯格式

一个最近被金融研究者重新发现的方法是**道格拉斯格式**，这个方法具有局部截断误差$O\left(\delta S^4,\delta t^2\right)$，而它的计算花费与克朗克-尼科尔森格式相同。一个想法是: 如果要获得比克朗克-尼科尔森格式更高的精度的话，需要在S方向上使用更多的格点，但这个方法不是这样。

我会使用基本扩散方程来描述这个方法。(关于将这个方法扩展至非定常系

数的对流-扩散方程的细节，参见本章结尾处的阅读清单。）

基本扩散方程是：

$$\frac{\partial V}{\partial t}+\frac{\partial^2 V}{\partial S^2}=0$$

它通常并不都写成这种格式，但是我希望我们使用的方程尽可能地采取紧凑形式。

显式法在这个方程上的应用是：

$$\frac{V_i^{k+1}-V_i^k}{\delta t}=\frac{V_{i+1}^k-2V_i^k+V_{i-1}^k}{\delta S^2}$$

类似地，全隐式法是：

$$\frac{V_i^{k+1}-V_i^k}{\delta t}=\frac{V_{i+1}^{k+1}-2V_i^{k+1}+V_{i-1}^{k+1}}{\delta S^2}$$

克朗克-尼科尔森格式是这两种方法的平均，但是如果我们将这两种方法做有权重的平均，会有什么好处吗？这导致了θ方法的思想，对显式法和全隐式法取权重平均，我们得到：

$$\frac{V_i^{k+1}-V_i^k}{\delta t}=\theta\left(\frac{V_{i+1}^{k+1}-2V_i^{k+1}+V_{i-1}^{k+1}}{\delta S^2}\right)+(1-\theta)\left(\frac{V_{i+1}^k-2V_i^k+V_{i-1}^k}{\delta S^2}\right)$$

当$\theta=\frac{1}{2}$时，我们回到了克朗克-尼科尔森方法。对于一般的θ取值，局部截断误差是：

$$O\left(\frac{1}{2}\delta t-\frac{1}{12}\delta S^2-\theta\delta t,\delta S^4,\delta t^2\right)$$

当$\theta=0$，$\frac{1}{2}$或1时，我们会得到已经见过的结果，但是如果：

$$\theta=\frac{1}{2}-\frac{\delta S^2}{12\delta t}$$

则局部截断误差被改进。这个方法比起克朗克-尼科尔森格式来并不更难实现。

78. 7 三时间平面方法

数值格式并没有只使用两个时间平面的约束。我们可以构建很多的使用三个或更多时间平面的算法。只要它能提供更好的局部截断误差或更好的收敛特性，我们会愿意做这样的事情。出于简洁性的考虑，我依旧针对基本扩散方程展开讨论。

第一个明显的方法是在显式格式中对时间导数使用中央差分。毕竟它比我们之前在显式格式中使用的时间导数精确多了。在这个时间导数的近似下我们得到:

$$\frac{V_i^{k+1}-V_i^{k-1}}{2\delta t}=\frac{V_{i+1}^k-2V_i^k+V_{i-1}^k}{\delta S^2}$$

虽然这个近似的精度更高，但是它并非对所有的时间步都是稳定的。在寻找好的差分格式的过程中，这是可能出现缺陷的地方。

一个对所有时间步都稳定的显式格式是:

$$\frac{V_i^{k+1}-V_i^{k-1}}{2\delta t}=\frac{V_{i+1}^k-V_i^{k+1}-V_i^{k-1}+V_{i-1}^k}{\delta S^2}$$

由此导出:

$$(1+2\nu_1)V_i^{k+1}=2\nu_1\left(V_{i+1}^k+V_{i-1}^k\right)+(1-2\nu_1)V_i^{k-1}$$

对于这个，以及其它三时间平面方法，在其起始位置上需要一个初值条件以及第一个时间平面上的数据，后者必须通过一个与我们使用的三时间平面方法具有相同精度的双时间平面方法来求得。

一种隐式方法具有与克朗克-尼科尔森方法同阶的精度，但它在非连续初始数据方面的表现更好，它是:

$$-2\nu_1\left(V_{i+1}^{k+1}+V_{i-1}^{k+1}\right)+(3+2\nu_1)V_i^{k+1}=4V_i^k-V_i^{k-1}$$

最后，理所当然的，三时间平面道格拉斯方法可以提升精度特性:

$$(3-24\nu_1)V_{i+1}^{k+1}+(30+48\nu_1)V_i^{k+1}+(3-24\nu_1)V_{i-1}^{k+1}$$

$$= 4V_{i-1}^{k} + 40V_{i}^{k} + 4V_{i+1}^{k} - V_{i+1}^{k-1} - 10V_{i}^{k-1} - V_{i-1}^{k-1}$$

78. 8 理查德森外推

显式法的误差是$O\left(\delta t, \delta S^2\right)$，如果我们假设，当时间步长和资产价格步长都趋向于0时，对精确解的逼近过程在某种意义上是“正则的”，则我们可以设:

$$\text{approximate solution} = \text{exact solution} + \varepsilon_1 \delta t + \varepsilon_2 \delta S^2 + \varepsilon_3 \delta t^2 + \cdots$$

对于一些系数ε_i，假设我们有两个使用不同格子尺寸的近似解V_1和V_2，我们可以写出:

$$V_1 = \text{exact solution} + \varepsilon_1 \delta t_1 + \varepsilon_2 \delta S_1{}^2 + \varepsilon_3 \delta t_1{}^2 + \cdots$$

$$= \text{exact solution} + \delta S_1^2 \left(\varepsilon_1 \frac{\delta t_1}{\delta S_1^2} + \varepsilon_2 \right) + \cdots$$

和

$$V_2 = \text{exact solution} + \varepsilon_1 \delta t_2 + \varepsilon_2 \delta S_2{}^2 + \varepsilon_3 \delta t_2{}^2 + \cdots$$

$$= \text{exact solution} + \delta S_2^2 \left(\varepsilon_1 \frac{\delta t_2}{\delta S_2^2} + \varepsilon_2 \right) + \cdots$$

如果我们选择:

$$\frac{\delta t_1}{\delta S_1^2} = \frac{\delta t_2}{\delta S_2^2}$$

也就是说，v_1是固定的——这里下标表示在求解V_1和V_2时使用的步长尺寸——则我们可以通过估计上述两式中的第一个误差项来获得比V_1和V_2更好的解。一个更好的近似由下式给出:

$$\frac{\delta S_2^2 V_1 - \delta S_1^2 V_2}{\delta S_2^2 - \delta S_1^2}$$

这个方法的精度是：$O\left(\delta t^2, \delta S^3\right)$。

现在回头看看图 78.3 和图 78.4，在那些图中，你可以看到误差与时间步长及资产价格步长平方的相关关系是多么的好。理查德森外推在这种情况下工作得很好。

为了给出一个例子，我们使用前面章节中给出的显式格式 Visual Basic 代码去计算一个看涨期权。设其敲定价格是100，一年到期，当前资产价格是100，波动率20%，利率是10%，此时精确的布莱克-舒尔斯价值是13.269。设资产方向上格点数是20，则上述程序给出的结果是13.067，因为计算的数量与$\delta t^{-1}\delta S^{-1}$呈正比，所以时间花费是8000个时间单位。在30个资产格点的情况下，计算出的价值提升至13.183，而时间花费是27000个时间单位。在这两个值上使用理查德森外推，得出的值是13.275，总的时间花费是$8000+27000=35000$，但是误差只有0.006。直接使用显式法达成这一精度需要100个资产格点，计算时间是1000000，是理查德森外推法的30倍。

Method	BS	Numerical	Error	Time taken
20 asset steps	13.269	13.067	0.202	8,000
30 asset steps	13.269	13.183	0.086	27,000
Richardson	13.269	13.275	0.006	35,000
100 asset steps	13.269	13.276	0.007	1,000,000

同样的外推原理可应用于本章讨论的任何方法。但是要小心使用这个方法，因为没有任何东西能保证我们一定能将解写成有意义的泰勒展开。所以这个方法有可能将事情搞坏。另外，外推法可能会增加舍入误差。

78. 9 自由边界问题与美式期权

我在接下来的两章中要展示的东西，其背后的理论具有重要的意义。这个理

论显然是重要的，因为它使得我接下来的工作是“有效的”。但是相对于其简明的最终结果而言，这个理论是晦涩的。所以我将直接给出结果。(译者注：这个神秘兮兮的理论大概就是所谓的“无套利原理”吧)。

美式期权的价值必须一直大于其收益，否则这里就存在着套利机会:

$$V(S,t) \geq \text{Payoff}(S)$$

收益函数也可能是时间依赖的。例如，对于百慕大期权来说，它只能在特定的日期执行。在这些日期中，它的收益是基础资产价格的预设函数，而在其他日期里，它的收益函数是0。这样我只要写出函数 $\text{Payoff}(S,t)$ 就可以了，而不需要再关心百慕大期权本身。

美式期权是“自由边界问题”的例子。我们必须在一个未知的边界下求解偏微分方程。与预设好的边界比起来，这里边界的位置是由多个边界条件所决定的。在美式期权问题中，我们知道，期权价值函数和它的 Delta 在与收益函数连接处是连续的。这个条件被称为**平滑过渡条件**。求解一个边界位置未知的问题是非常复杂的，如果我们通过一个直接的方法去解决这个问题的话，它实现起来恐怕并不那么简单。

需要注意的第一个问题是：我将在固定的 S 范围内求解。当然我会确认这个范围能够包含自由边界。并且，我不会尝试用任何直接的方法去找出这个边界的位置。

78. 9. 1 提前执行与显式法

让我来告诉你如何将提前执行约束引入到显式有限差分格式中来。并且我将解释为什么这种做法是明智的。

假设我们对于时间步 k 上（例如到期日上）的所有 i 都解出了 V_i^k，使用有限差分格式求解时间步 $k+1$ 上的期权价值，如下式:

$$V_i^{k+1} = A_i^k V_{i-1}^k + \left(1 + B_i^k\right) V_i^k + C_i^k V_{i+1}^k$$

直到解出所有的V_i^{k+1}为止，我们不必考虑是否违反了美式期权的约束。现在让我们检查新的期权价值是大于还是小于当时的收益，如果它小于当时的收益，则我们拥有了一个套利机会。我们在任何i点上都不能允许这种状况的发生：期权价值的值导致套利机会。所以我们会将该点上的期权价值替换为其上的收益。这就是要做的所有事情。

关于代码方面，只需将程序中 VOld(i)更新的代码行替换为如下的代码：

```
VOld(i) = max(VNew(i), Payoff(S(i), RealTime))
```

很明显的，这个有限差分方程的解最终会收敛到一个函数，这个函数在某个点上是与收益函数连续的。但是 Delta 连续条件的满足就不是那么明显的了——不过事实确实如此。显式法仅仅是二叉树方法的一种良好形式，而使用收益函数进行的简单替换也与二叉树方法中的做法完全相同。解的精度依旧是$O\left(\delta t,\delta S^2\right)$。

提前执行合约的边界的位置在两个资产价格之间，其一是期权价值等于收益的位置，其二是期权价值稍微超过收益的位置。在实践中，一个以上的这种边界是很有可能的，上面所述的方法可以将它们都找到而不需要额外的工作。

78. 9. 2 提前执行与克朗克-尼科尔森方法

在克朗克-尼科尔森方法中实现美式期权的约束比在显式法中要难一点。但其报酬是计算的精度。这个方法的精度依旧是$O\left(\delta t^2,\delta S^2\right)$，并且在时间步长上没有限制。

这里复杂性的增加仅仅因为克朗克-尼科尔森方法是隐式法。所有在时间步$k+1$上的期权价值都与该时间步上的其它价值相关。因此在计算出所有的期权价值后再用收益去进行替换就不够好了，替换必须与期权价值计算同时进行：

$$v_i^{n+1}=\max\left(v_i^n+\frac{\omega}{M_{ii}}\left(q_i-\sum_{j=1}^{i-1}M_{ij}v_j^{n+1}-\sum_{j=i+1}^{N}M_{ij}v_j^n\right),\text{Payoff}\right)$$

i和$k+1$点上的收益已经被显式地估了出来，这个方法被称为投影超松弛迭代法。

下面的 Visual Basic 代码与前面的 SOR 代码相同，除了有一行代码用来检查期权价值是否大于收益 Payoff。如果不是，则收益将被用作新的价值。

```
Dim q(1 To N) As Double
Dim v(1 To N) As Double
Dim temp(1 To N) As Double
Dim MDiag(1 To N) As Double
Dim MSuperDiag(1 To N + 1) As Double
Dim MSubDiag(0 To N) As Double
MSuperDiag(N + 1) = 0
MSubDiag(0) = 0
NoIts = 0

While Error < tol
    Error = 0
    For i = 1 To N
        temp(i) = v(i) + omega * (q(i) - MSuperdiag(i + 1) * v(i + 1) - _
            MDiag(i) * v(i) - MSubDiag(i - 1) * v(i - 1)) / MDiag(i)
        temp(i) = max(temp(i), Payoff(S(i)))
        Error = Error + (temp(i) - v(i)) * (temp(i) - v(i))
        v(i) = temp(i) ' 计算出来后立刻被使用
    Next i
    NoIts = NoIts + 1
Wend
```

78. 10 阶跃条件

偏微分方程所需要满足的，除了终值条件（收益）、边界条件（位于零，无穷大和障碍上）外，经常还会有阶跃条件。该条件的引入可能是由于在除息日基础资产价格的跃动，息票支付，以及路径依赖离散采样量上的跃动等等。其中的一种情况非常的简单，而另外两种情况在实现上基本相同。

78. 10. 1 离散现金流

如果一项合约允许我们在特定的日期离散地取得收益（如债券息票），则在现金流实现的那一刻，合约价值必然产生阶跃。在连续时间中，我们有：

$$V\left(S,t_d^-\right)=V\left(S,t_d^+\right)+C \qquad (78.8)$$

现金流C可能是基础资产价格的函数，或者甚至是期权价格的函数。当你为利率产品估值时，你经常会遇到与S相关的r。

从数值方法的角度来说，一个理想的状况是：产生现金流的日期t_d刚好落在格点上。也就是说，从这个日期到到期日刚好是整数个时间步。如果是这种情况的话，则对（78.8）的实现就是很直接的了。用任意的数值方法计算期权价值，直到日期t_d（包括t_d本身），在计算下一个时间步（也就是真实时间的前一步）之前，将每个期权价值都加上C。

当产生现金流的日期并不符合格点位置时，问题就复杂了。一个最简单的办法是在这个日期之后（真实时间之前）的第一个时间步上加C。这样做的精度是$O(\delta t)$，与显式法相符合。如果你使用更加精确的数值方法，如克朗克-尼科尔森方法，你需要使用对（78.8）更精确的实现。如果你在求解基本的布莱克-舒尔斯方程，并且现金流是一个常数，不依赖于基础资产价格和期权价值，则可以在某一时间步上对C取现值，以获得一个等价的现金流。如果现金流是与基础资产价格呈正比的，则（当没有基础资产的除息时）在最近的时间步上使用该阶跃条件也是没有问题的。因为基础资产价格本身就满足布莱克-舒尔斯方程。通过一个明晰的，但有时是冗长的方法，可以将现金流精确地引入计算，类似这种情况还有很多简单的例子。但是当现金流的状况更加复杂，或者在求解一个利率产品时，最好的方法还是还是调整紧随现金流发生时之后（真实时间）的那个时间步的步长，使得发生现金流的日期刚好落在网格上。这个方法可以保证现金流以与数值方法同阶的精度引入计算。

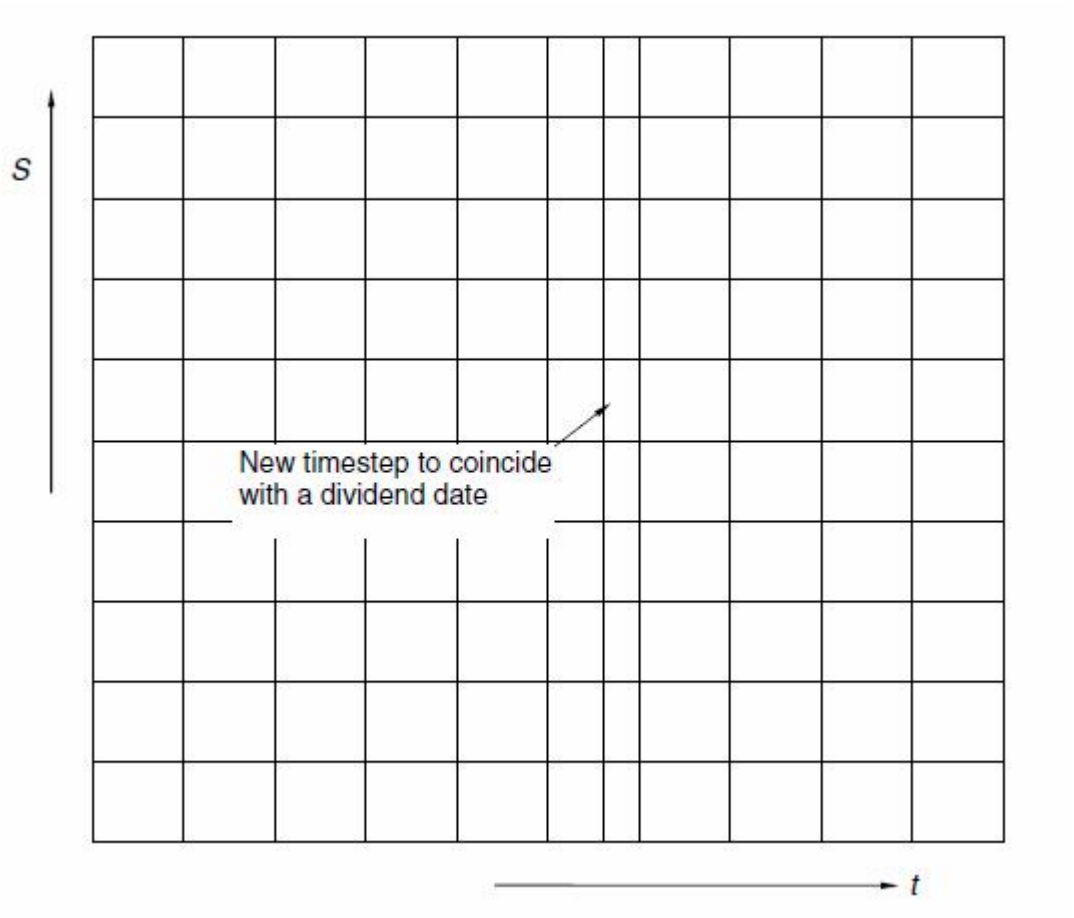


图 78.5 离散现金流或除息日附近的网格

78. 10. 2 离散除息

当某个独立变量发生阶跃的时候，一些更有趣味的阶跃条件出现了。如果在基础资产价格上出现了一个阶跃，则我们必须实现一个更加复杂的条件。我将仅针对一个实例进行讨论，其它的同类状况可以用相同的方式进行处理。

假设股息 $D(S)$ 在日期 t_d 上支付。越过这个除息日，资产价格将会下跌 $D(S)$（这样就没有套利了），但是期权价值不会改变。这在第 8 章已经解释过。在数学上我们可以写出：

$$V\left(S,t_d^-\right)=V\left(S-D(S),t_d^+\right)$$

当我们进行数值求解时，我们会发现在方程的右边有从到期日步进过来的时间。在继续对时间进行反向步进前，我们需要实现式（78.9）。这里我假设t_d位于一个时间步上，或者在这里时间步长已经被调整以便满足这个假设。也就是说，在实现（78.9）时，我们只需要关心与有限的资产价格步长相关的精度问题。

我将示范如何用线性插值来实现这个阶跃条件，这个方法的精度是$O\left(\delta S^2\right)$，与前面所述的其它方法相同。

通过有限差分格式，我们可以知道所有i上的V_i^{k+1}。这是刚刚经过了除息日后的也就是位于t_d^+上的期权价值。为了实现阶跃条件，我们必须找出将点$S-D$夹在中间的那些格点，然后我们会通过对这两点的插值来计算股息支付前期权价值的精确值。

下面是实现了该阶跃条件的Visual Basic代码段。支付的股息是`Div(S(i))`，它是基础资产价格的一个函数。除息日后的期权价值存储于`VOld(i)`中，之前的期权价值则储存于`VNew(i)`中。`dummy`是点`S(i) - Div(S(i))`与其最近的格点间的距离比上资产价格步长`AssetStep`。

```
For i = 0 To NoAssetSteps
    inew = Int((S(i) - Div(S(i))) / AssetStep)
    dummy = (S(i) - Div(S(i)) - inew * AssetStep) / AssetStep
    VNew(i) = (1 - dummy) * VOld(inew) + dummy * VOld(inew + 1)
Next i
```

这里有一个稍微复杂的状况，我之前假设$0 \leq$ `inew` $\leq$ `NoAssetSteps`。如果当点`S(i) - Div(S(i))`位于我们原来的网格之外，情况就不是这样了。为了处置这种情况，通常要对网格之外的V的行为进行推想。依据具体的问题，这个推想值通常是常数或者基础资产价格的线性函数。相应的代码修改是很简单的。

78. 11 路径依赖期权

我在第二部分讨论过的路径依赖合约具有简单的偏微分方程形式。一般这种问题是三维的，基础资产（资产，利率等），时间和路径依赖量。这些奇异期权所带来的三维问题具有两种形式，一种形式不会在方程中增加新的项，因为路径依赖量是离散采样的；另一种形式会导致方程中出现额外的项，这个项是对新的连续采样变量的一阶导数。

很明显，路径依赖问题最重要的方面就是这个额外的维度。我们必须在三维状态下求解问题，因此我们引入新的期权价值:

$$V_{i,j}^{k}$$

上标 k 依旧代表时间，下标 i 代表资产，而 j 代表新的变量。

上面为读者提供的信息，已经足以给出求解新问题的方法了。我会简单地指出这些方法，并给出两个具体的例子。

78. 11. 1 离散采样路径依赖量

当路径依赖量是离散采样的情况下，偏微分方程中不会引入新的项。因此在其近似的差分方程中也不会有新的项。这使得求出有限差分解不会太难，并且上面所述的任何方法都是可用的。但是记住，你现在面对的是一个三维网格，你必须对每一个 j，也就是路径依赖量的取值解出相应的期权价值。在这个方法中，你要使用反向时间步进，直到你遇到一个采样日期。在越过这个采样日期时，需要使用一个阶跃条件。对这个条件的实现可以达成与离散除息问题中同样的精度。

假设阶跃条件是:

$$V\left(S,I,t_d^-\right)=V\left(S,F\left(S,I\right),t_d^+\right)$$

它同样可以使用线性插值来实现。并且如果设δI为路径依赖量I的步长的话，则实现精度为$O\left(\delta I^2\right)$。

下面的 Visual Basic 代码段中，路径依赖量为 P，其步长为 PDQStep。如果 jnew 的位置跑到了范围 0 到 NoPDQSteps 之外，那么我们需要使用外推法。注意我对格点之间的阶跃条件进行了插值。

```
For j = 0 To NoPDQSteps
    For i = 0 To NoAssetSteps
        jnew = Int(F(S(i), P(j)) / PDQStep)
        dummy = (F(S(i), P(j)) - jnew * PDQStep) / PDQStep
        VNew(i, j) = (1 - dummy) * VOld(i, jnew) + dummy * VOld(i, jnew + 1)
    Next i
Next j
```

78. 11. 2 连续采样路径依赖量

当路径依赖量是连续采样的时候，通常偏微分方程中会引入新的项，并且其近似差分方程中也会出现新的项:

$$\frac{\partial V}{\partial t}+a\left(S,I,t\right)\frac{\partial^2 V}{\partial S^2}+b\left(S,I,t\right)\frac{\partial V}{\partial S}+f\left(S,I,t\right)\frac{\partial V}{\partial I}+c\left(S,I,t\right)V=0$$

（在实际情况下，系数很少会依赖于I）

除显式有限差分外，其它任何方法的实现都太复杂了。所以我在这里将只讨论显式法。

格式的更新，边界和终值条件的实现都是很直观的，在这里唯一需要注明的问题是对导数:

$$\frac{\partial V}{\partial I}$$

的离散格式的选择。

因为在I的方向上不存在扩散现象（也就是说，方程中没有期权价值对I的二阶导数项），所以差分格式的选择是很重要的。对I的一阶导数使得方程在I的方向上呈现出对流特性，数值格式必须与此相容。这导致必须要使用单边差分

格式。系数$f(S,I,t)$的正负改变必须在差分格式的选择上得到反映，所以这里必须要使用逆风差分格式。下面是对差分格式的一种可能的选择:

$$f(S,I,t)\frac{\partial V}{\partial I}(S,I,t)=f^k_{i,j+\frac{1}{2}}\frac{V^k_{i,j+1}-V^k_{i,j}}{\delta I} \text{ 当 } f(S,I,t)\geq 0$$

以及

$$f(S,I,t)\frac{\partial V}{\partial I}(S,I,t)=f^k_{i,j-\frac{1}{2}}\frac{V^k_{i,j}-V^k_{i,j-1}}{\delta I} \text{ 当 } f(S,I,t)<0$$

另外，为了取得更好的精度，可以使用某种三点差分格式。

关于三维奇异期权问题数值解的最后一点是: 如果存在一个近似消元法可以将问题降为二维的话，那么这个方法应当得到使用。它不仅可以极大地加速求解过程，同时也可以扩展差分格式的选择范围，令其实现更加容易。

78. 12 小结

如果你成功地完成了对本章中各种方法的学习和实践练习工作的话，那么你已经达到了一个远高于二叉树方法的专家水平……当然，还有更高的目标在等着你。

扩展阅读

- 偏微分方程数值解方面的杰出书籍是Morton & Mayers (1994) 和 Smith (1985)。Richtmyer & Morton (1976) 和 Mitchell & Griffiths (1980) 在非定常系数偏微分方程方面写得非常好，这也是我们在金融问题中经常要用到的内容。
- Roache (1982) 是针对计算流体力学中的数值方法的一项很有趣味的说明。
- 克朗克-尼科尔森方法的原文参见 Crank & Nicolson (1947)。

- 对克朗克-尼科尔森格式的评论参见 Duffy (2004)。
- 对于本章上述各种格式的更多细节参见 Ahmad (2006)

第 79 章　双因素模型有限差分方法

本章内容:

- 双因素模型显式有限差分法
- 交替方向隐式法（ADI）和跳房子法

79. 1 简介

当前流行的许多金融模型都具有两个随机因素，可转换债券的定价通常要受随机的基础资产价格和随机的违约风险或利率的影响。股票的奇异型衍生品经常要参考随机的波动率进行定价，对于障碍型期权来说尤为如此。有限差分法非常适于解决此类问题。

当你面对三个因素，亦即三个随机性的来源，有限差分法开始变得缓慢而且笨拙。当有四个随机因素时——如果可能的话，你应当使用蒙特卡罗法来进行定价，这个方法将在第 80 章讨论。蒙特卡罗法在高维的情况下工作得更好些。

有限差分法在低维的情况下工作良好，并且在处置提前执行问题方面非常高效。一些路径依赖问题也可以很容易得到处理。

在本章中，我将讨论应用于两个或多个随机因素状况下的有限差分法的最简形式。

79. 2 双因素模型

在整个章节中，我都会围绕以下的双因素方程展开讨论:

$$\frac{\partial V}{\partial t}+a(S,I,t)\frac{\partial^2 V}{\partial S^2}+b(S,I,t)\frac{\partial V}{\partial S}+c(S,I,t)V$$

$$+d(S,I,t)\frac{\partial^2 V}{\partial r^2}+e(S,I,t)\frac{\partial V}{\partial S\partial r}+f(S,I,t)\frac{\partial V}{\partial r}=0 \qquad (79.1)$$

我在前面章节中所写的每种方法，可以良好地适用于任何问题，而不论其基础变量是什么。我将要解决的的问题写成了（79.1）的形式，是认为读者能够将具体的“资产”和“利率”看做是更加抽象的 x 和 y。事实上，当我们考虑第 33 章中所描述的双因素可转换债券问题的解法时，这种抽象的思考将极大地帮助我们。这是因为这种问题包含很多重要的，但是可变的特性，例如对利率模型的选择，如何将其离散化，以及提前执行条件等。对于一般的双因素问题，如果要求它是抛物型的，则我们需要有:

$$e(S,r,t)^2<4a(S,r,t)d(S,r,t)$$

如同单因素领域里一样，变量必须被离散化，也就是说，我们会在如下的一个三维网格上进行求解:

$$S=i\delta S,\quad r=j\delta r \text{ 和 } t=T-k\delta r$$

到期日是 $t=T$ 或 $k=0$，对于 i 和 j，其指标范围相应的是 0 到 I 或 J。我在这里假设利率模型仅定义在 $r\geq 0$ 的范围上。但有些状况并非如此，对于一些简单的利率模型如 Vasicek 模型来说，它在负的 r 上也有良好的定义。如果我们有这样的一个模型则我们应当重新定义 r。我在上一章中对利率模型和边界条件所做的评述同样适用于双因素领域。

合约价值可以写做:

$$V(S,r,t)=V_{ij}^k$$

对于任何要去求解的问题，我们都必须提交各种确定的条件。首先，我们必须定义终值条件，它就是合约的收益函数，它会告诉我们在到期日合约的价值如何。

假设我们为一个具有看涨收益的长期权证定价，这是双因素模型的一个很好例子。我们可能要考虑引入一个随机性的利率，因为在长期条件下，我们不会认为定常利率会是一个好的设想。对于上面的问题，终值条件是：

$$V(S,r,T)=V_{ij}^{0}=\max(S-E,0)$$

和终值条件相同的，我们必须在环绕计算区域的位置上提交边界条件。S 和 r 的区域展示于图 79.1 中，边界条件必须在所有由黑点所标记的格点上进行提交。边界条件是依赖于合约的。记住，图上还有一个时间轴是指向读者的，但是没有画出。

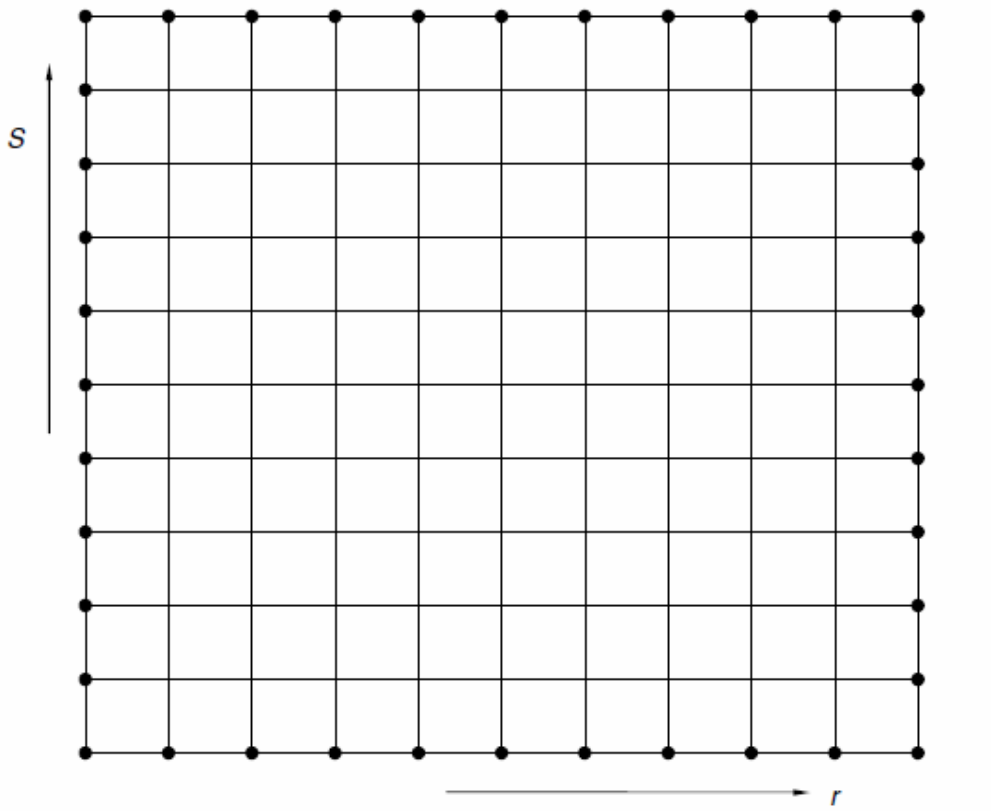


图 79.1 S，r 区域和边界条件

79. 3 显式法

单因素显式法可以很容易地扩展为双因素的。事实上，其易于编程的特性使

得它成为处理这一新课题的良好方法。我将对式（79.1）中的所有导数使用对称的中央差分。这是对二阶导数求近似的最好方法，但对一阶导数可能并非如此。我将在后面讨论这个。对于所有的项，我们都曾学习过如何去使用中央差分，除了对S和r的交叉二阶导数项:

$$\frac{\partial^2 V}{\partial S \partial r}$$

我们可以将其近似为:

$$\frac{\partial\left(\frac{\partial V}{\partial r}\right)}{\partial S} \approx \frac{\frac{\partial V}{\partial r}(S+\delta S, r, t) - \frac{\partial V}{\partial r}(S-\delta S, r, t)}{2\delta S}$$

而

$$\frac{\partial V}{\partial r}(S+\delta S, r, t) \approx \frac{V^k_{i+1,j+1} - V^k_{i+1,j-1}}{2\delta r}$$

这显示一个适用的离散化方法可能是:

$$\frac{\frac{V^k_{i+1,j+1} - V^k_{i+1,j-1}}{2\delta r} - \frac{V^k_{i-1,j+1} - V^k_{i-1,j-1}}{2\delta r}}{2\delta S}$$

$$= \frac{V^k_{i+1,j+1} - V^k_{i+1,j-1} - V^k_{i-1,j+1} + V^k_{i-1,j-1}}{4\delta S \delta r}$$

它具有特别的好处，因为它不仅具有与其它导数逼近方法同阶的误差水平，而且它还保持了特性:

$$\frac{\partial^2 V}{\partial S \partial r} = \frac{\partial^2 V}{\partial r \partial S}$$

最终的显式有限差分格式是:

$$\frac{V_{i,j}^k - V_{i,j}^{k+1}}{\delta t} + a_{i,j}^k \left(\frac{V_{i+1,j}^k - 2V_{i,j}^k + V_{i-1,j}^k}{\delta S^2} \right) + b_{i,j}^k \left(\frac{V_{i+1,j}^k - V_{i-1,j}^k}{2\delta S} \right) + c_{i,j}^k V_{i,j}^k$$

$$+ d_{i,j}^k \left(\frac{V_{i,j+1}^k - 2V_{i,j}^k + V_{i,j-1}^k}{\delta r^2} \right) + e_{i,j}^k \left(\frac{V_{i+1,j+1}^k - V_{i+1,j-1}^k - V_{i-1,j+1}^k + V_{i-1,j-1}^k}{4\delta S \delta r} \right)$$

$$+ f_{i,j}^k \left(\frac{V_{i,j+1}^k - V_{i,j-1}^k}{2\delta r} \right) = O\left(\delta t, \delta S^2, \delta r^2\right)$$

我们可以将其按如下格式重写:

$$V_{i,j}^{k+1} = \cdots$$

这里方程右边是如图 79.2 所示的九个期权价值的线性函数。在时间步 k 上这九个值的系数与 a，b 等相关。将方程整理成这种形式对我们的工作帮助不是很大，因为实际实现的方法通常要比它明晰得多。注意在一般性的情况下所有的九个点 (i,j)，$(i\pm1,j)$，$(i,j\pm1)$，$(i\pm1,j\pm1)$ 都被用于格式中，如果没有交叉导数项的话，则只有五个点 (i,j)，$(i\pm1,j\pm1)$ 被使用。这会简化一些方法。

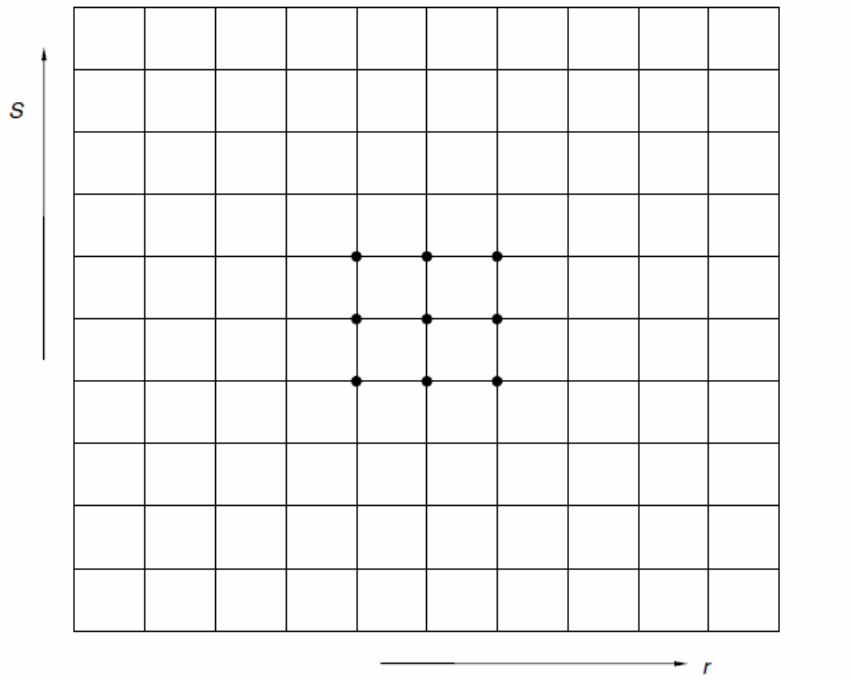


图 79.2 显式法时间步进所需要的九个期权价值

现在我会给出一个例子。下面是一段为可转换债券进行定价的 Visual Basic 代码，基础资产价格是 S(i)，利率是 r(j)：

```
For j = 1 To MR - 1
    For i = 0 To MS - 1
        VS = (VOld(i + 1, j) - VOld(i - 1, j)) / (2 * dS)
        Vrp = (VOld(i, j + 1) - VOld(i, j)) / dr
        Vrm = (VOld(i, j) - VOld(i, j - 1)) / dr
        Vr = Vrm

        If RDdrift(r(j), RealTime) > 0 Then Vr = Vrp

        VSS = (VOld(i + 1, j) - 2 * VOld(i, j) + VOld(i - 1, j)) / (dS * dS)
        Vrr = (VOld(i, j + 1) - 2 * VOld(i, j) + VOld(i, j - 1)) / (dr * dr)
        VSr = (VOld(i + 1, j + 1) - VOld(i - 1, j + 1) - VOld(i + 1, j - 1)+ _
            VOld(i - 1, j - 1)) / (4 * dS * dr)
        VNew(i, j) = VOld(i, j) + dt * (0.5 * S(i) * S(i) * vol * vol * VSS _
            + 0.5 * ratevol * ratevol * Vrr + rho * vol * ratevol * S(i) * _
            VSr + r(j) * (S(i) * VS - VOld(i, j)) - div * S(i) * VS _
            + RNDrift(r(j), RealTime) * Vr)
    Next i
Next j
```

注意期权价值 VOld(i, j)的导数是如何依次定义的。我对每个导数都使用了中央差分，除了对利率的导数外。它被定义为一个单边差分，使用逆风差分格式。我使用了一个不够精确（误差为$O(\delta r)$）的单边差分，但它可以如第 77 章中所建议的那样进行精度的提升。之所以在对利率的一阶导数上使用逆风差分，是因为利率的波动率通常是非常小的。扩散现象使得数值解法变得较为容易，而它对利率的影响很不明显，这使问题变得更像是双曲型的而不是抛物型的。

基础资产价格的波动率是 vol，利率的波动率是 ratevol，它们之间的相关度是 rho，波动率和相关系数都是常数。

我们使用的利率模型是：

$$dr = \mathtt{RNDrift}(r,t)\,dt + \mathtt{ratevol}\,dX$$

我将风险中性漂移率 RNDrift 设为利率和时间 RealTime 的共同函数。这个关于时

间依赖的设定使得该代码段可以和利率期限结构模型一起使用。

对于这个代码段，我还想讨论的最后一点是关于边界条件的。利率下标 j 的范围是从 0 到 MR，在满足 j = 0 和 j = MR 以及任意 i 的所有点上都必须提交边界条件。对于资产价格，其下标 i 的范围是从 0 到 MS，其对边界条件提交的要求是一样的。在这个代码段中，请注意我在 i=0 的点上，也就是一些边界点上所使用的时间步进算法。这并不通用，我只能在一些特殊条件下使用它。在计算 i=0，也就是资产价格 $S=0$ 的位置上方程的解的时候，我们允许 VOld(i, j)寻址 i=-1 的位置，而这个结果是有效的。这是因为在 $S=0$ 的位置上扩散已经消失。换句话来说，也就是点 $S=0$ 是奇异的。

在应用了相应的边界条件之后，开始计算下一时间步之前，必须将产品的可转换特性包含进来。对于显式法这很容易完成，下面的代码会保证这里没有套利机会:

```
For i = 0 To MS
    For j = 0 To MR
        VOld(i, j) = max(VNew(i, j), ConvertRate * S(i))
    Next j
Next i
```

这段代码将旧价值替换为新的价值 VNew(i, j)，同时要保证债权的价值大于可转换收益 convertrate * S(i)。通常这种转换只允许在特定日期上或特定日期之间进行，所以这个约束只在那些允许进行转换的时间步上得到应用。

79. 3. 1 显式法的稳定性

显式法的一个优点是易于编程，其主要的缺点来源于稳定性和速度。这个方法只在足够小的时间步长下是稳定的，而时间步长的上限严重地限制了计算速度。

我们可以再次通过考察差分方程的解在 S 和 r 方向上的振荡性来对稳定性进行分析。这意味着要考察如下形式的解:

$$V_{i,j}^{k}=\alpha^{k}e^{2\pi\sqrt{-1}(i/\lambda_S+j/\lambda_r)}$$

并假设所有的系数 a，b，c 等相对于 δS 和 δr 的尺度是缓慢变化的。我将跳过细节，简单地给出在一个特殊但却非常重要的情况下的解：如果我们有一个纯粹的扩散问题而没有对流或衰减项，所有的变量都是非相关的，并且只有系数 a 和 d 是非零的，则稳定性条件是：

$$a\frac{\delta t}{\delta S^2}+d\frac{\delta t}{\delta r^2}\le\frac{1}{2}$$

对比单因素问题显式法的状况，这是一个严厉得多的约束。

为了规避这个时间步长上的约束，我们可以尝试隐式格式，很快我们就会谈到它。

79. 4 计算时间

解出一个偏微分方程以便为一个期权定价，需要花费多少时间？

让我们来考察应用于期权定价问题的显式有限差分法。由于显式有限差分法的应用并不仅限于一维或二维的情况，所以我们在这里考察 d 维状况下的求解过程。假设我们预期的精度是 ε。这将要花费多少计算时间呢？

显式法的误差是 $O(\delta t)$ 和 $O(\delta S^2)$。如果我们希望它们和 ε 都是同阶的，那么：

$$\text{time step}=O(\varepsilon)$$

以及

$$\text{asset step}=O(\varepsilon^{1/2})$$

计算时间与格点的数目呈正比，其阶次是：

$$\frac{1}{\delta t}\times\frac{1}{\delta S}\times\cdots\times\frac{1}{\delta S}$$

这里有 d 个资产价格步长项。因此计算时间是：

$$O(\varepsilon^{-1}\varepsilon^{-d/2})=O(\varepsilon^{-1-d/2})$$

计算对冲系数不会花费更多的时间，因为我们已经将所有格点上的期权价值都计算出来了。但是如果我们要为 M 个不同的期权进行定价，则计算时间是：

$$O\left(M\varepsilon^{-1-d/2}\right)$$

提前执行条款不会增加很多的计算时间。

79. 5 交替方向隐式法

有很多的隐式方法可用于单因素问题上。而对于双因素问题，则有着更多的隐式方法。好的数值方法应当是快速的，并且其存储需求要尽可能的少。我将只讨论两个应用于双因素问题的隐式有限差分法，它们都使用与前面一章中所述相同的思想。第一种方法被称为**交替方向隐式法**，或 ADI。

我们将尝试对克朗克-尼科尔森方法进行双因素扩展，以便解出 $(I-1)(J-1)$ 个方程（与未知数的数目等同）组成的方程组。这里的问题是其最终得到的矩阵的形式并不好，对其进行求解将会是复杂并且耗费时间的。如果我们想要保持隐式法在稳定性方面的优点和显式法在易于求解方面的长处，则我们可以尝试在一个因素上进行隐式求解而在另外一个因素上进行显式的求解。这就是 ADI 方法背后的思想。我将首先用语言去解释这个思想。

与 $V_{i,j}^{k}$ 相同的，我们在这里引入一个中间变量 $V_{i,j}^{k+(1/2)}$。从时间步 k 到中间步 $k+\frac{1}{2}$，我们在 S 方向上使用显式差分，在 r 方向上使用隐式差分进行求解。因为是有一个方向是隐式的，所以在这里使用 LU 分解和 SOR 方法求解并不比在单因素状况下更难。当我们求得 $V_{i,j}^{k+(1/2)}$，并继续步进到时间步 $k+1$ 上时，我们在 S 方向上使用隐式差分，而在 r 方向上使用显式差分。在这个“半时间步”上，相对于前一个“半时间步”，我们改变了显式法和隐式法的方向。最终的矩阵方程是简单的。这个方法对于所有的时间步长都是稳定的，并且其误差是

$O\left(\delta t^2,\delta S^2,\delta r^2\right)$。

现在让我们看一下在实践中这个方法是如何实现的。我将使用如下的简单方程进行示例：

$$\frac{\partial V}{\partial t}+a\left(S,r,t\right)\frac{\partial^2 V}{\partial S^2}+d\left(S,r,t\right)\frac{\partial^2 V}{\partial r^2}=0 \qquad (79.2)$$

在一个完整的，具有一阶导数项的方程上使用 ADI 方法就并不更难，只要差分格式可以被分解为分别关于S和r的导数项。交叉导数项会给 ADI 的基本实现形式造成一些问题。这些问题是可以克服的，但是过程比较冗长。

S方向上显式，r方向上隐式的离散化结果是：

$$\frac{V_{i,j}^k-V_{i,j}^{k+(1/2)}}{\frac{1}{2}\delta t}+a_{i,j}^k\left(\frac{V_{i+1,j}^k-2V_{i,j}^k+V_{i-1,j}^k}{\delta S^2}\right)$$

$$+d_{i,j}^{k+(1/2)}\left(\frac{V_{i,j+1}^{k+(1/2)}-2V_{i,j}^{k+(1/2)}+V_{i,j-1}^{k+(1/2)}}{\delta r^2}\right)=0$$

将所有时间步$k+\frac{1}{2}$上的项移到左边，将时间步k上的项移到右边，我们得到：

$$V_{i,j}^{k+(1/2)}-\frac{1}{2}d_{i,j}^{k+(1/2)}\left(\frac{V_{i,j+1}^{k+(1/2)}-2V_{i,j}^{k+(1/2)}+V_{i,j-1}^{k+(1/2)}}{\delta r^2}\right)\delta t$$

$$=V_{i,j}^k+\frac{1}{2}a_{i,j}^k\left(\frac{V_{i+1,j}^k-2V_{i,j}^k+V_{i-1,j}^k}{\delta S^2}\right)\delta t$$

如果我们知道所有时间步k上的项，则我们可以通过求解一个联立方程组来求得$V_{i,j}^{k+(1/2)}$。这与第 78 章中提到的全隐式格式相同。而这个二维版本的唯一不同在于，必须对所有的i进行求解。对于每一个i，其对应方程组的求解与一维状况下的复杂度是相同的，不过这里多了几套方程组而已。这个隐式格式可以写为一个与三对角线矩阵相关的矩阵方程。它可以通过 LU 分解或 SOR 法进行求解。

当我们解出$V_{i,j}^{k+(1/2)}$后，我们继续时间步进以解出$V_{i,j}^{k+1}$。这里我们将显式法和

隐式法的方向进行互换:

$$\frac{V_{i,j}^{k+1}-V_{i,j}^{k+(1/2)}}{\frac{1}{2}\delta t}+a_{i,j}^{k+1}\left(\frac{V_{i+1,j}^{k+1}-2V_{i,j}^{k+1}+V_{i-1,j}^{k+1}}{\delta S^2}\right)$$

$$+d_{i,j}^{k+(1/2)}\left(\frac{V_{i,j+1}^{k+(1/2)}-2V_{i,j}^{k+(1/2)}+V_{i,j-1}^{k+(1/2)}}{\delta r^2}\right)=0$$

它可以重写为:

$$V_{i,j}^{k+1}+\frac{1}{2}a_{i,j}^{k+1}\left(\frac{V_{i+1,j}^{k+1}-2V_{i,j}^{k+1}+V_{i-1,j}^{k+1}}{\delta S^2}\right)\delta t$$

$$=V_{i,j}^{k+(1/2)}-\frac{1}{2}d_{i,j}^{k+(1/2)}\left(\frac{V_{i,j+1}^{k+(1/2)}-2V_{i,j}^{k+(1/2)}+V_{i,j-1}^{k+(1/2)}}{\delta r^2}\right)\delta t$$

同样的，这是一个通过$V_{i,j}^{k+(1/2)}$求解$V_{i,j}^{k+1}$的全显式（译者注：隐式？）格式。

79. 6 跳房子法

我想要讨论的最后一个方法是**跳房子法**。之所以这样称呼它，是出于它在时间步进时使用格点的方式——参见图 79.3。同样的，出于简单性的考虑，我们仍以方程（79.2）为例。

对于每个时间步，我们执行两个步骤。对于$k=1$以及所有的奇数时间步，我们首先计算图 79.3 中用圆圈标出的格点的值。圆圈标出的格点的定义是$i+j$为奇数。这可以使用显式法:

$$V_{i,j}^{k+1}=V_{i,j}^{k}+a_{i,j}^{k}\left(\frac{V_{i+1,j}^{k}-2V_{i,j}^{k}+V_{i-1,j}^{k}}{\delta S^2}\right)\delta t+d_{i,j}^{k}\left(\frac{V_{i,j+1}^{k}-2V_{i,j}^{k}+V_{i,j-1}^{k}}{\delta r^2}\right)\delta t$$

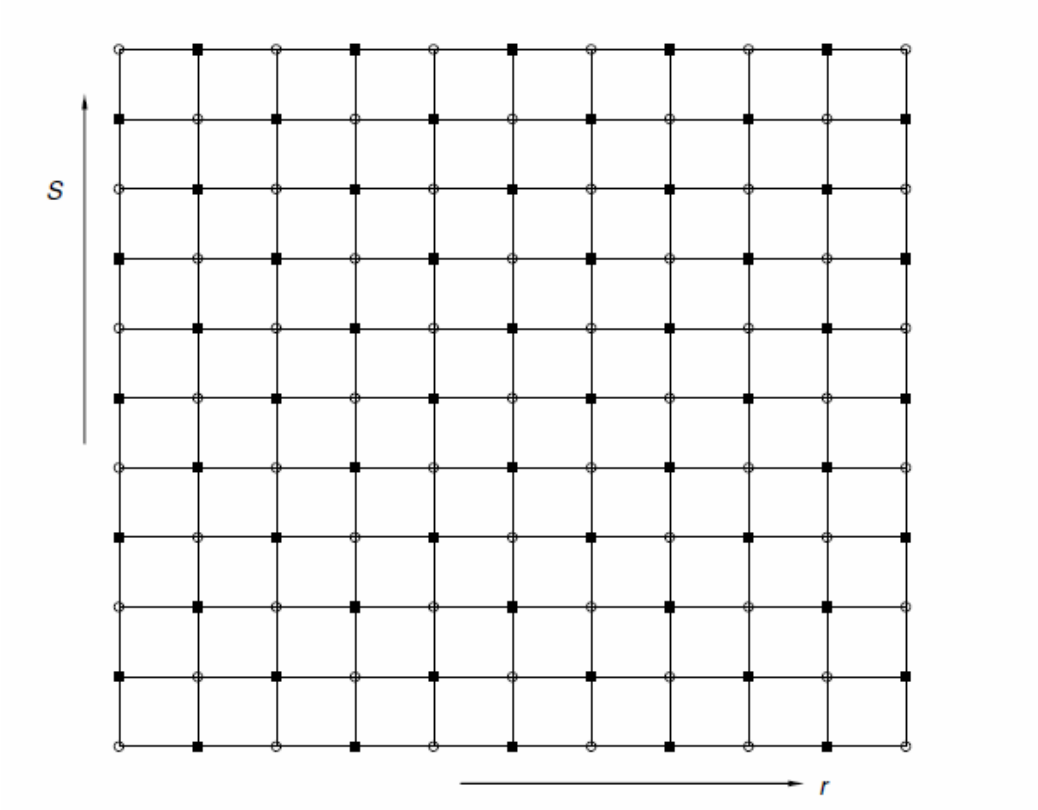


图 79.3 跳房子法对格点的使用

现在我们执行第二个步骤，在同一时间平面上，对用方块标出的格点使用相同的格式，但是这里代入的是刚刚计算出的值$V_{i,j}^{k+1}$

$$V_{i,j}^{k+1}=V_{i,j}^{k+1}+a_{i,j}^{k+1}\left(\frac{V_{i+1,j}^{k+1}-2V_{i,j}^{k+1}+V_{i-1,j}^{k+1}}{\delta S^2}\right)\delta t+d_{i,j}^{k+1}\left(\frac{V_{i,j+1}^{k+1}-2V_{i,j}^{k+1}+V_{i,j-1}^{k+1}}{\delta r^2}\right)\delta t$$

虽然这个格式在技术上是隐式的，但是它不需要解任何的联立方程组。这个方法的误差是$O\left(\delta t,\delta S^2,\delta r^2\right)$。

当我们开始计算下一时间步，也就是所有的偶数时间步时，用圆圈标出的格点和用方块标出的格点需要做一个角色互换。

如果在微分方程中存在交叉导数项，则格式中的显式性会消失。

79. 7 小结

对于具有两个或三个随机因素的金融问题，有限差分法是适用的。对于随机因素更多的状况，蒙特卡罗法可能是一个更好的方法。有限差分法特别适用于具有提前执行条款、或者看涨看跌可选的合约。随机波动率（见第 51 章）下的可转换债券或美式期权是两个明显的例子。由于单纯的显式有限差分法类似于二叉树法，所以本章中的方法至少不会比二叉树方法差。而一些有限差分方法则要好得多。ADI 方法和跳房子方法虽然难于编程，但是作为那些简单但却缓慢的方法的替代，它们还是值得去尝试一下的。

扩展阅读

- 偏微分方程的数值解法参见 Morton & Mayers (1994) 和 Mitchell & Griffiths (1980)
- Roache (1982) 讨论了双因素或多因素状况下更多的数值方法
- 带有相关性的 ADI 方面的工作参见 McKee, Wall & Wilson (1996)
- 多因素方法方面的更多细节参见 Ahmad (2006)

第 80 章　蒙特卡洛模拟

本章内容:

- 期权价值与股票、货币、商品及指数的期望值之间的关系
- 随机利率情况下衍生产品与利率期望间的关系
- 在计算衍生品价格和查看衍生品投机结果方面如何进行蒙特卡罗模拟
- 如何在高维问题上使用乔里斯基分解（Cholesky Factorization）进行模拟

80. 1 简介

衍生品定价理论的基础是资产价格和利率等的随机游走。我们在第 3 章看到关于股票，货币和商品的随机行为基础理论；并在第 5 章之后看到由此产生的期权定价理论。这就是导出了布-舒抛物型偏微分方程的布莱克-舒尔斯理论。我们还在第 10 章中看到，一个随机变量的随机微分方程模型是如何导出关于其概率密度函数的类似方程的。在那一章，我揭示了期权定价和转移概率密度之间的关系。本章中，我们将利用这种关系去了解如何从对资产价格或利率的随机行走的特定模拟中发现衍生品的价格。概括地讲，期权的价值就是其收益的现值期望——这里的核心是对“期望”的精确定义。

我将要区分以股票、指数、货币及商品为基础资产的期权在确定性利率条件下的估值和在随机利率条件下的估值。首先，我将揭示衍生品价值与在确定性利率条件下的收益期望的关系。

80. 2 对股票、指数、货币和商品等的模拟与衍生品估值间的关系:

回忆一下第 10 章的内容，在布莱克-舒尔斯理论中，一个期权的公允价值是其在到期日的收益的现值期望，这个期望是在基础资产价格风险中性随机游走的条件下取得的。

S 的风险中性随机游走是:

$$dS = rSdt + \sigma SdX$$

因此我们可以写出:

$$\text{option value}=e^{-r(T-t)}E\left[\text{payoff}(S)\right]$$

它提示期望值是与风险中性随机游走相关的，而非真实的价格波动。

这一结果导致了对期权价值进行估计的以下简单步骤:

1. 按下文的讨论模拟风险中性随机游走，从资产 S_0 在当前的价格开始，覆盖所需要的时间范围。这一期间从当前开始一直持续到期权的到期日。这是基础资产价格路径的一个实现。
2. 在这个实现的基础上计算期权回报。
3. 在既定的时间范围内实现更多的价格路径。
4. 计算所有实现的平均回报。
5. 计算这个平均回报的现值，这就是期权的价值。

80. 3 生成路径

该算法的初始部分，需要首先在一个标准正态分布（或一些适当近似）下生成随机数。我们将在下面讨论这个问题，目前先假设我们可以生成这样的一个具有足够元素数量的序列。然后我们应当在每一时间步上使用这个随机序列作

为增量来更新资产价格。这里，在如何更新 S 上，我们有一个选择。

一个显而易见的选择是使用：

$$\delta S = rS\delta t + \sigma S\sqrt{\delta t}\phi$$

这里 ϕ 是从一个标准正态分布得出的。这种对进行时间序列模拟的离散方法被称为**欧拉方法**。简单的将最新的 S 值代入方程右边去计算 δS，然后再用其求出下一个 S 的值。这个方法可以很方便地应用于任何随机微分方程。这种离散化方法的误差是：$O(\delta t)$。

	A	B	C	D	E	F	G	H	I
1	Asset	100		Time	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5
2	Drift	5%		0	100.00	100.00	100.00	100.00	100.00
3	Volatility	20%		0.01	100.15	101.27	100.79	100.16	98.54
4	Timestep	0.01		0.02	99.80	100.84	102.72	102.31	101.66
5	Int. rate	5%		0.03	97.35	103.77	105.27	102.00	105.64
6				0.04	96.50	103.08	104.72	97.65	105.35
7			= D3+$B$4	0.05	101.25	101.61	102.37	102.76	103.63
8				0.06	97.53	100.49	104.47	106.86	99.04
9				0.07	97.41	103.09	107.70	105.73	99.20
10		=E3*EXP(($B$5-0.5*$B$3*$B$3)*$B$4+$B$3*SQRT($B$4)*NORMSINV(RAND()))							.77
11				0.09	85.74	100.79	109.07	106.01	97.95
12				0.1	81.32	100.99	105.13	105.40	100.32
94				0.92	102.25	105.44	88.51	96.74	96.08
95				0.93	100.68	105.48	90.44	97.04	95.36
96				0.94	102.26	104.01	92.40	99.26	94.67
97				0.95	102.10	103.47	88.99	95.27	97.09
98				0.96	100.11	103.36	88.95	92.74	96.30
99				0.97	101.34	104.06	89.26	93.59	97.19
100				0		= MAX($B$104-F102,0)		= MAX(G102-$B$104,0)	99
101		=AVERAGE(E104:IV104)		0.99	103.71	102.73			98
102				1	104.94	104.47	91.86	95.05	98.79
103									
104	Strike	105	CALL	Payoff	0.00	0.00	0.00	0.00	0.00
105	=D105*EXP(-		Mean	8.43					
106	$B$5*$D$102)		PV	8.02					
107			PUTP	ayoff	0.06	0.53	13.14	9.95	6.21
108			Mean	8.31					
109			PV	7.9					
110									
111									

图 80.1 使用蒙特卡罗模拟对看涨/看跌期权进行估值的电子表格

上述算法如图 80.1 所示。股票价格在时间点 $t=0$ 具有起始值100以及波动率20%。电子表格同时计算看涨和看跌期权的值。它们距离到期日都为一年，并

且敲定价格是105。利率是5%。在电子表格中我们看到的是对S的随机游走的大量蒙特卡罗模拟中的一小部分。这里的漂移率是5%。时间步长被选为0.01。对于每一个实现，股票价格的终值在行102上（行13-93被隐藏了）。期权收益在行104和行107中显示。所有这些模拟的收益的均值，在行105和行108中显示。在行106和行109中我们将看到这些均值的现值，也就是期权价值。在实际的期权估值中你不能只用一张电子表格进行模拟。对于目前的例子，我采取了数量相对较少的样本路径。

	A	B	C	D	E	F	G	H	I
1	Asset	100		Time	Sim 1	Sim 2	Sim 3	Sim 4	Sim 5
2	Drift	5%		0	100.00	100.00	100.00	100.00	100.00
3	Volatility	20%		0.01	98.62	97.68	99.73	99.42	102.98
4	Timestep	0.01		0.02	100.69	96.45	101.13	101.28	101.36
5	Int. rate	5%		0.03	99.60	99.67	102.62	99.37	101.95
6				0.04	99.19	101.15	104.14	98.60	99.51
7				0.05	104.10	100.00	105.03	98.97	96.86
8				0.06	104.71	99.11	103.22	96.93	98.89
9									
10									
11				0.09	110.57	100.93	101.59	99.34	98.75
12				0.1	114.50	100.24	99.36	95.67	99.88
13				0.11	114.43	101.32	100.22	94.92	102.90
93				0.91	101.02	111.09	119.38	77.82	85.23
94				0.92	101.54	109.58	118.06	80.13	83.75
95				0.93	101.38	108.20	118.49	79.96	83.54
96				0.94	103.38	107.87	119.79	82.21	83.12
97				0.95	107.58	108.43	116.24	81.58	84.69
98						109.39	115.79	81.61	88.50
99				0.97	107.20	112.81	115.56	83.07	90.72
100				0.98	109.18	113.45			
101					110.49	114.04			
102					113.23	117.67	120.05	81.49	93.27
103									
104				Average	105.98	106.95	109.21	87.43	97.22
105									
106	Strike	105	ASIAN	Payoff	0.98	1.95	4.21	0.00	0.00
107			Mean	4.79					
108			PV	4.55					
109									
110									

= D3+$B$4

=E3*EXP(($B$5-0.5*$B$3*$B$3)*$B$4+$B$3*SQRT($B$4)*NORMSINV(RAND()))

=AVERAGE(E2:E102)

=AVERAGE(E106:IV106)

= MAX(G104-$B$106,0)

=D107*EXP(-$B$5*$D$102)

图 80.2 使用蒙特卡罗模拟对亚式期权进行估值的电子表格

该方法尤其适合路径依赖的期权。在图 80.2 的电子表格中，我展示了如何对亚式期权进行估值。该合约的支付额是 $\max(A-105,0)$，这里 A 是合约一年存续期中资产价格的均值。关于基础资产的其它细节与前例一样。思考以下问题：如果只考虑合约存续期中最后六个月的资产价格均值，那么电子表格应当如何修改？

80. 4 对数正态基础资产，非路径依赖期权

对于对数正态随机游走，非常幸运地，我们能够找到一种简单，准确的时间步进算法。我们可以以如下形式写出 S 的风险中性随机微分方程：

$$d(\log S)=\left(r-\frac{1}{2}\sigma^2\right)dt+\sigma dX$$

本式可以被积分为：

$$S(t)=S(0)\exp\left(\left(r-\frac{1}{2}\sigma^2\right)t+\sigma\int_0^t dX\right)$$

或者，基于时间步长 δt 的：

$$S(t+\delta t)=S(t)+\delta S=S(t)\exp\left(\left(r-\frac{1}{2}\sigma^2\right)\delta t+\sigma\sqrt{\delta t}\phi\right) \quad (80.1)$$

注意这里并不要求小的 δt，因为这个表达式是精确的。由于这个方法的精确和简单，它是最好的时间步进算法。此外，因为它是准确的，所以如果期权的收益只依赖于资产终值，也就是说是欧式的和非路径依赖的，那么我们可以直接对资产终值进行模拟，即使用时间步长 T。

如果该期权是路径依赖的，那么我们必须回到一般的小时间增量方法。

80. 5 蒙特卡罗模拟的优点

现在我们对蒙特卡罗仿真如何与期权定价发生联系有了一定认识，使用这种模拟方法的一些优点是:

- 执行蒙特卡罗模拟对你的数学基础只有很基本的要求。
- 随机变量的相关性可以很容易地建模。
- 有很多的软件可用，至少电子表格函数可以满足大部分的状况。
- 执行更多的模拟可以取得更好的精度。
- 在求解一些问题时需要花费的工作很少。
- 通常不需要太多工作就可以修改模型。
- 复杂的路径依赖性可以很容易地引入。
- 人们普遍接受这个技术，并相信你的结果。

80. 6 使用随机数

我们所见到的布莱克-舒尔斯理论建立于如下的假设之上：二叉树模型，也就是资产价格的简单上下移动，或是正态分布的回报。当我们模拟一个随机游走时，只要时间步长足够的小，而资产价格路径中包含足够大量的时间步，我们就不必太过关心在随机增量上使用的是什么分布。我们所需要的只是分布的方差是有限的和固定的。(这种必要的定常性使得年化的波动率，也就是说，年化的回报标准差是一个精确的值。特别是，它(译者注：波动率？)必须与 $\delta t^{1/2}$ 的值相匹配。)当时间步长的大小趋向于0时，在有限时间尺度上的模拟具有相同的概率特性，而不必考虑在无穷小时间尺度上的分布特性。这是中心极限定理的一个结果。

不过，最准确的模型是具有正态收益的对数正态模型。由于一个人不得不考虑模拟足够数量的资产路径以获得一个足够精确的期权价格，他不希望多过地

考虑时间步长的长度问题。如我在上面所述，最好的方法是使用精确的公式（80.1），由此对于时间步长的选择就不会去影响随机游走的精度了。在一些情况下我们可以只使用一个时间步，因为此时时间步进算法是精确的。如果我们用这么大的时间步长，那么我们必须生成正态分布随机变量。我将在下面讨论这些，我将讨论保克斯-穆勒方法。

如果时间步长是δt，对于更为复杂的产品，如路径依赖产品来说，我们依旧可能引入误差$O(\delta t)$，这是对连续事件的离散近似造成的。这里的一个例子是连续障碍期权。如果我们使用有限的时间步，我们可能会丢失时间点之间障碍被触发的状况。一般来说，由有限时间步数所引入的误差是：$O(\delta t)$。

由于只能在无限的资产价格路径中模拟其中有限的部分，在我们实现N个资产路径时，误差是$O(N^{-1/2})$。

在对一个衍生品进行估价时，所需要的运算总数是$O(N/\delta t)$。这是对定价过程中所花费的计算时间的一个衡量。定价过程中的误差是：

$$O(\max(\delta t, \frac{1}{\sqrt{N}}))$$

也就是说，由离散时间步和有限个路径实现所造成的误差中最坏的那一个。为了将其最小化，当我们将计算时间固定为$O(N/\delta t)=K$时，我们必须选择：

$$N=O(K^{2/3}) \text{ 和 } \delta t=O(K^{-1/3})$$

80. 7 生成正态变量

一些随机数生成器是好的，而另一些则是坏的——它们会在有限个样本后就开始自我重复，或者显现出序列自相关的特性。它们也可以是快的或者慢的。一个易于用电子表格实施的，快的并且非常有用的分布是下面的正态分布近似：

$$\left(\sum_{i=1}^{12}\psi_i\right)-6$$

这里ψ_i是独立随机变量，其分布是从0到1的均匀分布。这个分布接近于正态分布，具有均值0，标准差1，以及三阶矩0。它的四阶矩和更高阶的矩则和正态分布有所不同。当我使用小的时间步长去生成资产价格路径时，我会在电子表格中使用这个方法。

一个更好的近似是：你可以对更多的均匀分布随机变量进行求和。一般的公式是：

$$\sqrt{\frac{12}{N}}\left(\left(\sum_{i=1}^{N}\psi_i\right)-\frac{N}{2}\right)$$

它具有均值0和标准差1。当N更大时这个近似也就更好。但是当N太大时，我们不妨使用下面的技术生成真正的正态分布数字。

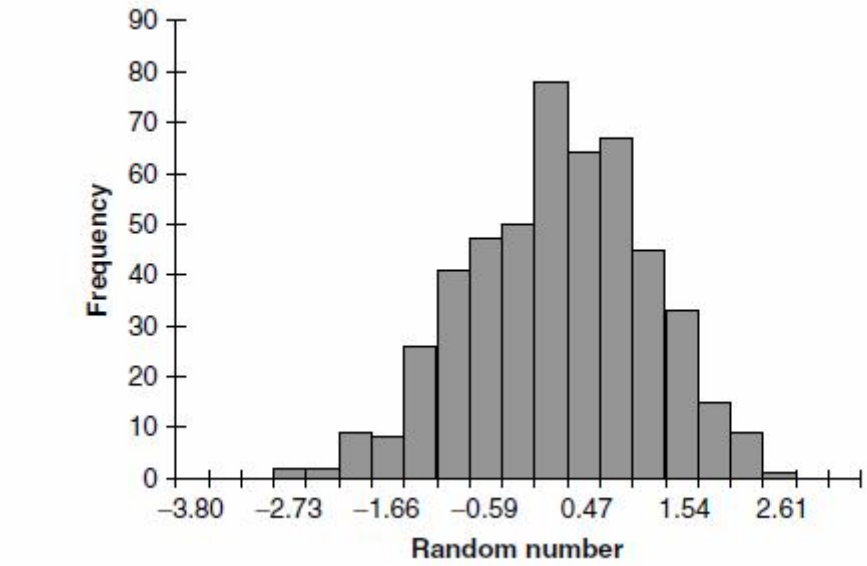


图 80.3 使用500个均匀分布数据点和保克斯-穆勒方法对正态分布进行的近似

80. 7. 1 保克斯-穆勒方法（Box-Muller）

如果你需要生成真正的正态分布随机数，那么最简单的技术是：**保克斯-穆勒方法**。这个方法使用均匀分布变量，并将其转化为正态分布的。基本的均匀分布数可以用很多的方法生成，一些算法可参见杂志 *et al*（1992）。保克斯-穆勒方法使用两个从0到1之间的均匀分布随机数 x_1 和 x_2，并由它们计算出两个数 y_1 和 y_2。这两个数都是正态分布的：

$$y_1 = \sqrt{-2\ln x_1}\cos(2\pi x_2) \text{ 和 } y_2 = \sqrt{-2\ln x_1}\sin(2\pi x_2)$$

这里有一个用来输出正态分布变量的 Visual Basic 函数：

```
Function BoxMuller()
    Randomize
    Do
        x = 2 * Rnd() - 1
        y = 2 * Rnd() - 1
        dist = x * x + y * y
    Loop Until dist < 1
    BoxMuller = x * Sqr(-2 * Log(dist) / dist)
End Function
```

图 80.3 是使用 500 个均匀分布数据点和保克斯-穆勒方法对正态分布进行的近似。

对于更新的和更有效的正态分布随机变量生成算法，请参见 Jäckel（2002）。

80. 8 实际情况与风险中性，投机与对冲

图 80.4 中展示了对风险中性资产价格随机游走的几个实现，参数是利率 5%，波动率 20%。它们是虚线。图中实线是相应的实际随机游走，它使用相同的随机数，但这里使用 20% 的漂移率以代替 5% 的利率漂移。虽然我在这里一直强调蒙特卡罗模拟在期权估值中的使用，但我们也可以在持有一个未对冲期权

头寸时用它来评估收益分布。在这种情况下，我们不仅仅是对收益的均值或期望值有兴趣，而且对整个的收益分布（及其现值）有兴趣。这是因为当我们持有一个未对冲期权头寸时，我们无法像在持有已对冲头寸时那样（理论上的）有一个可保证的回报。因此，获取一个真实的漂移率以便将其作为参数引入，是有效并且重要的。使用风险中性漂移率去评估一个未对冲头寸的回报概率密度函数是错误的。

在图 80.5 中，我展示了在如下条件下估计出的概率密度函数和积累分布函数：看涨期权，到期日一年，敲定价格105，$\mu = 20\%$，$\sigma = 20\%$。概率密度曲线没有显示零收益时的情况，在到期时没有收益的概率大约是25%。

在图 80.6 中，我展示了在如下条件下估计出的概率密度函数和积累分布函数：看跌期权，到期日一年，敲定价格105，$\mu = 20\%$，$\sigma = 20\%$。概率密度曲线没有显示零收益时的情况，在到期时没有收益的概率大约是75%。

在投机状态下为期权估值是第 59 章的主题。

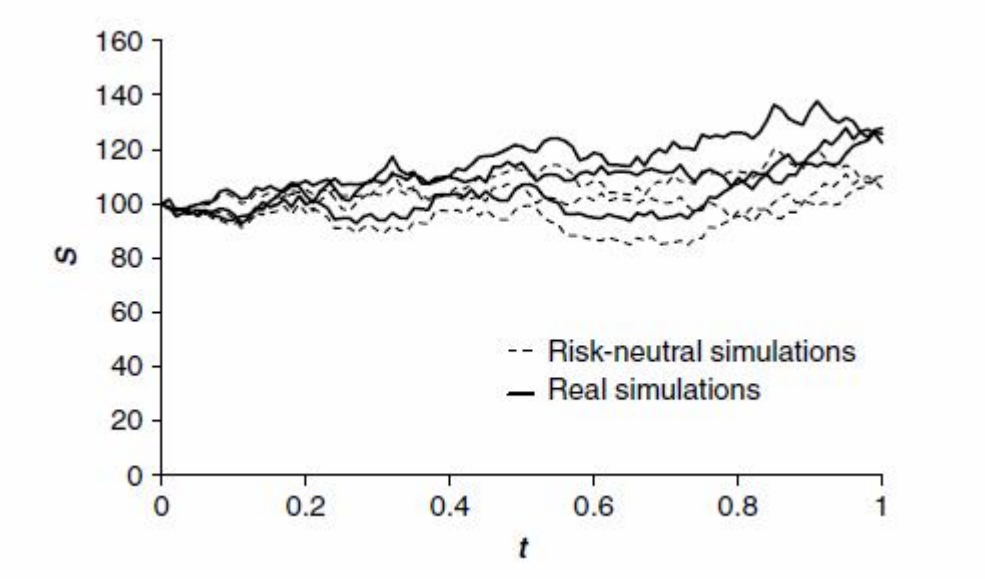


图 80.4 资产价格随机游走的一些实现

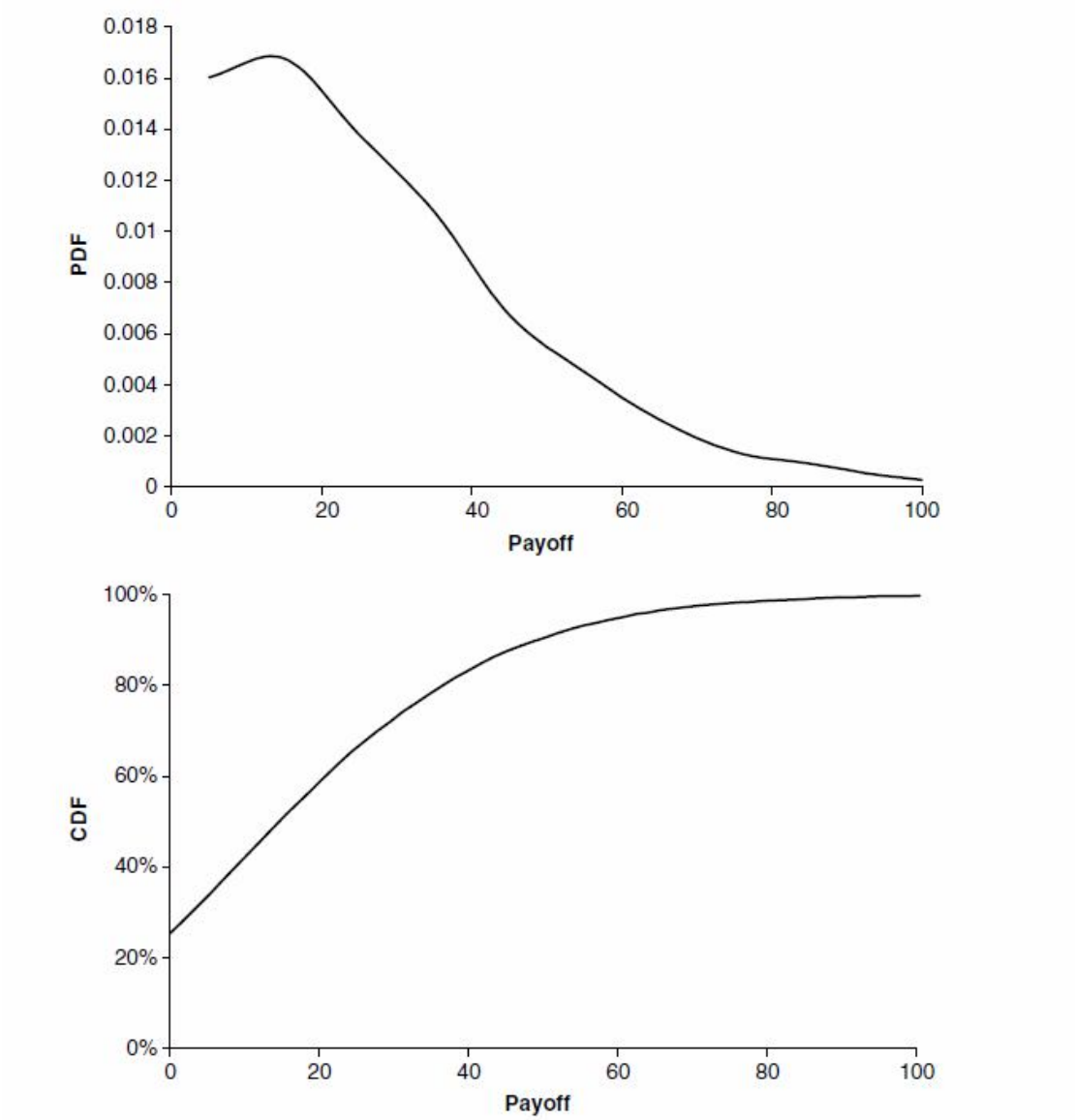


图 80.5 看涨期权收益的真实概率密度函数和积累分布函数

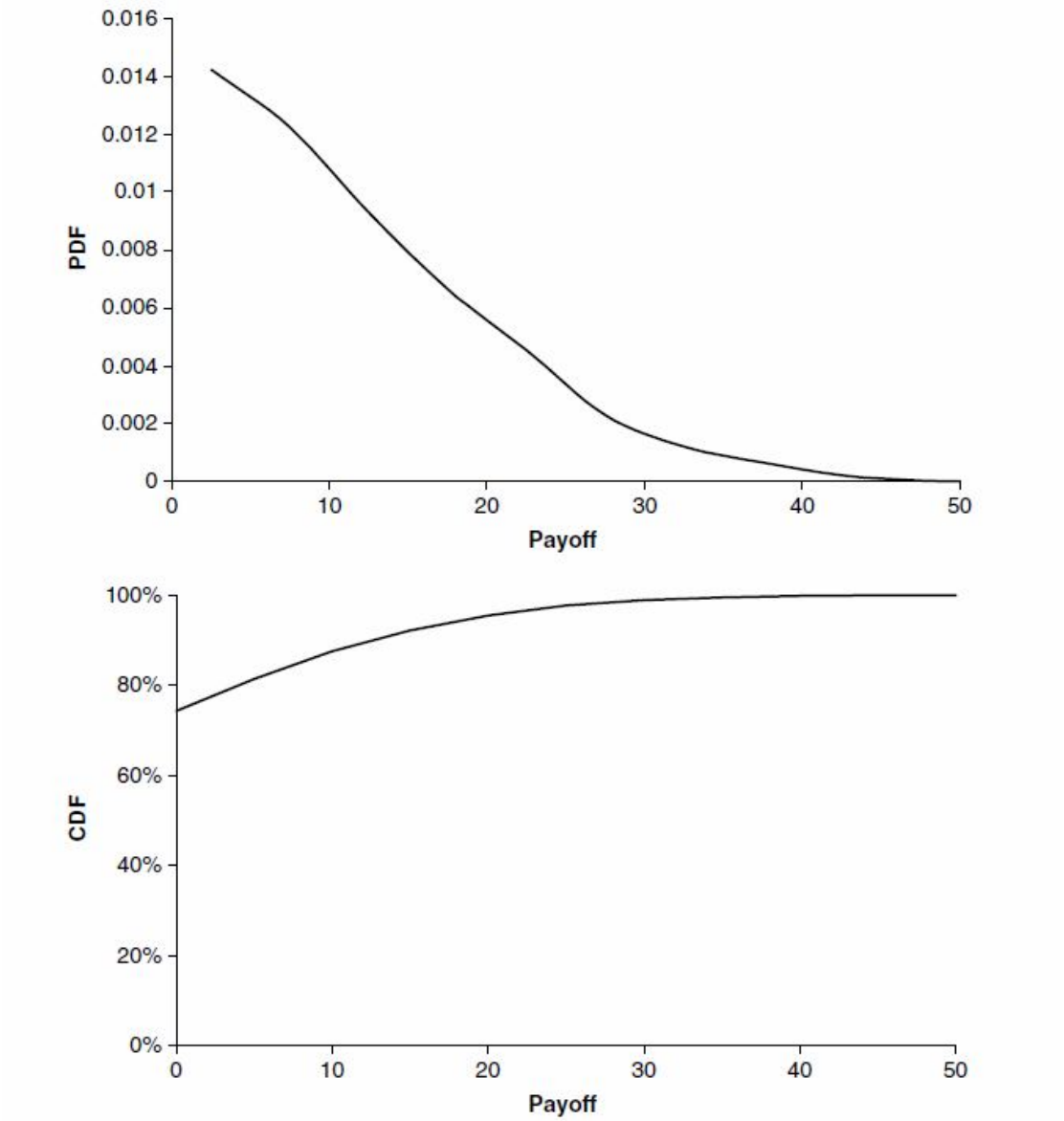


图 80.5 看跌期权收益的真实概率密度函数和积累分布函数

80. 9 利率产品

在短期利率呈现随机性的情况下，期权价值和它的期望收益之间的关系会稍微复杂一些，因为存在这样的问题：应当采用什么样的折现率。

在随机利率条件下对期权价值进行估计的正确方法如下:

1. 对风险调整后的即期利率的随机游走进行模拟，如下面所讨论的。从即期利率在今天的值开始，覆盖所需要的时间范围。这一期间从今天开始一直持续到期权的到期日。这是即期利率路径的一个实现。
2. 对于这个实现计算两个量: 收益和直到收益实现为止的利率平均。
3. 执行更多的实现。
4. 对于的随机游走的每一个实现，用这个实现的平均利率计算折现率，并计算收益的现值。
5. 对所有的实现计算收益现值的平均，这就是期权的价值。

换句话来说:

$$\text{option value} = E\left[e^{-\int_t^T r(\tau)d\tau} \text{payoff}(r) \right]$$

为什么这里与确定性利率的状况不同？为什么使用平均利率计算折现率？我们之所以使用平均利率折现所有的现金流，是因为这是货币市场客户所接受的利率。并且，在风险中性领域所有的资产具有相同的无风险增长速度。考虑在银行中的现金增长为:

$$\frac{dM}{dt} = r(t)M$$

其解为:

$$M(t) = M(T)e^{-\int_t^T r(\tau)d\tau}$$

它包含与期权价值相同的折现率。

对即期汇率模型离散化方法的选择通常局限于欧拉方法:

$$\delta r = \left(u(r,t) - \lambda(r,t)w(r,t)\right)\delta t + w(r,t)\sqrt{\delta t}\phi$$

很少有即期利率方程可以被精确积分的。

在下面的电子表格（图 80.7）中，我示范了一个蒙特卡罗方法，它用来计算一个具有回报 $\max(r-10\%,0)$ 的合约。该合约到期日是一年。模拟使用的模型

是具有固定参数的 Vasicek 模型。即期利率的初值是10%。期权价值是最后一行中所显示的平均现值。

	A	B	C	D	E	F	G	H
1	Spot rate	10%		Time	Sim 1	Sim 2	Sim 3	Sim 4
2	Mean rate	8%		0	10.00%	10.00%	10.00%	10.00%
3	Reversion rate	0.2		0.01	9.99%	10.04%	10.10%	9.88%
4	Volatility	0.007		0.02	9.95%	10.07%	10.14%	9.87%
5	Timestep	0.01		0.03	9.88%	10.08%	10.08%	9.81%
6				0.04	9.78%	10.12%	10.18%	9.84%
7			=D3+$B$5	0.05	9.87%	10.17%	10.15%	9.75%
8				0.06	9.87%	10.18%	10.10%	9.82%
9				0.07	9.73%	10.20%	10.11%	9.81%
10				0.08	9.72%	10.21%	10.27%	9.82%
11				0.09	9.60%	10.22%	10.25%	9.73%
12							9%	9.79%
13							4%	9.77%
14							7%	9.75%
15				0.13	9.57%	10.15%	10.38%	9.73%
95				0.93	8.85%	10.44%	10.21%	9.27%
96				0.94	8.81%	10.47%	10.23%	9.21%
97				0.95	8.83%	10.49%	10.17%	9.10%
98				0.96	8.81%	10.62%	10.19%	9.01%
99				0.97	8.73%	10.63%	10.29%	9.14%
100				0.98	8.68%	10.71%	10.28%	9.18%
101				0.99	8.56%	10.68%	10.15%	9.09%
102				1	8.42%	10.64%	10.33%	9.22%
103								
104				Mean rate	9.19%	10.32%	10.18%	9.56%
105								
106	Strike	10%		Payoff	0.0000	0.0064	0.0033	0.0000
107			PV'd		0.0000	0.0058	0.0030	0.0000
108			Mean	0.001352				
109								
110								
111								
112								
113								
114								

=F5+$B$3*($B$2-F5)*$B$5+$B$4*SQRT($B$5)*(RAND()+RAND()+RAND()+RAND()+RAND()+RAND()+RAND()+RAND()+RAND()+RAND()+RAND()-6)

=AVERAGE(E2:E102)

=MAX(E102-$B$106,0)

=E106*EXP(-$D$102*E104)

=AVERAGE(E107:IV107)

图 80.7 使用蒙特卡罗模拟对具有回报 $\max(r-10\%,0)$ 的合约进行估值的电子表格

80. 10 计算对冲系数

一个最简单的计算期权的 Delta 的方法是使用两次蒙特卡罗方法对期权进行估值。Delta 是期权价值对于基础资产价格的导数:

$$\Delta = \lim_{h\to 0} \frac{V(S+h,t)-V(S-h,t)}{2h}$$

这是在第 77 章中讨论过的中央差分。它对一阶导数的估值误差是$O(h^2)$:。但是在蒙特卡罗模拟中，在点$S+h$和点$S-h$上对期权进行估值的误差要比前面的求导误差要大得多。当h变小时，蒙特卡罗估值的误差会变大，最终的误差是:$O(1/hN^{1/2})$。为了克服这个问题，可以在对$S+h$和$S-h$两点进行估值时，使用同样的随机数序列。这样蒙特卡罗模拟的误差就可以对消了。同样的原理可用于计算期权的 Theta 和 Gamma。

当我们处置对数正态随机游走的状况时，事情变得非常简单。首先在股票价格为S的点上模拟许多对数正态随机路径。然后设想在股票价格为$(1+\varepsilon)S$的点上，你是否需要对所有的路径进行重采样？不，你需要做的所有事情就是将股票的终值，也就是到期日的价格乘上一个系数$1+\varepsilon$——当然这只对对数正态随机游走有效。然后用这些调整后的价格去计算期权的收益及其价值。使用系数$1-\varepsilon$重复上面的过程。将这两个新的期权价值取差分，并除以$2\varepsilon S$就是你要求出的 Delta 值了。这里几乎没有额外的计算时间。

最后，我们可以通过观察 Delta 需要满足的方程来求它的值。

布莱克-舒尔斯方程:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0 \qquad (80.2)$$

的解可以被解释为风险中性随机游走下收益的现值期望。

方程中的第一个'r'代表了

$$dS = rSdt+\sigma SdX$$

中的'r'；第二个'r'代表了现值。

将微分方程（80.2）对S求导，得:

$$\frac{\partial \Delta}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 \Delta}{\partial S^2}+\left(r+\sigma^2\right)S\frac{\partial \Delta}{\partial S}=0$$

这里

$$\Delta=\frac{\partial V}{\partial S}$$

将其与布莱克-舒尔斯方程进行对比可知，上面方程的解代表了在随机游走

$$dS=\left(r+\sigma^2\right)Sdt+\sigma SdX$$

下Δ终值（一个阶跃函数）的期望值。现在我们可以使用与计算期权价值时相同的那些S路径了。但此时它们被转换为与

$$Se^{\int_0^t \sigma^2(\tau)d\tau}$$

相关的。可以使用这个转换后的随机游走来计算这个阶跃函数(到期日的 Delta)的期望值。因为在 Delta 的偏微分方程中没有折现项，所以这里不需要取现值。

80. 11 高维：乔里斯基分解（CHOLESKY FACTORIZATION）

对于基于多种资产的欧式合约的定价，蒙特卡罗方法是一个自然的选择。假设我们有一个期权，它的收益是$S_1,S_2,\ldots,S_d$的函数，则从理论上来说，我们可以写出一个具有$d+1$个自变量的偏微分方程。这样的问题在计算时间上将会呈现难以接受的状况。上面所讨论的模拟方法可以轻易地进行扩展以便解决这种问题。我们所需要做的所有事情就是去模拟:

$$S_i\left(t+\delta t\right)=S_i\left(t\right)\exp\left(\left(r-\frac{1}{2}\sigma_i^2\right)\delta t+\sigma_i\sqrt{\delta t}\phi_i\right)$$

问题在于ϕ_i是相关的。

$$E\left[\phi_i\phi_j\right]=\rho_{ij}$$

我们如何生成相关的随机变量？这就是**乔里斯基分解**的由来。

让我们设想现在可以生成 d 个不相关的正态分布变量 $\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_d$。我们可以通过以下变换从中得到具有相关性的随机变量：

$$\phi=\mathrm{M}\varepsilon \qquad (80.3)$$

这里 ϕ 和 ε 是列向量，ϕ_i 和 ε_i 是它们在第 i 行上的元素。矩阵 M 是特定的，并且必须满足：

$$\mathrm{MM}^T=\Sigma$$

这里 Σ 是相关矩阵。

要证明这个变换的正确性很简单，从式（80.3）中我们可以得到：

$$\phi\phi^T=\mathrm{M}\varepsilon\varepsilon^T\mathrm{M}^T \qquad (80.4)$$

对这个矩阵方程的每一项取期望，可得：

$$E\left[\phi\phi^T\right]=\mathrm{M}E\left[\varepsilon\varepsilon^T\right]\mathrm{M}^T=\mathrm{MM}^T=\Sigma$$

我们之所以在这里可以通过矩阵的乘法取得期望值，是因为式（80.4）的右边对项 $\varepsilon_i\varepsilon_j$ 是线性的。

将相关矩阵分解为两个矩阵的乘积的方式并不是唯一的。乔里斯基分解提供了一个在分解中进行选择的途径。它将使矩阵 M 是一个下三角矩阵。这里有一个进行这种分解的算法。

矩阵 `Sigma` 包含了一个 `n` 维的相关矩阵，输出矩阵存储于 M 中：

```
Function cholesky(Sigma As Object)
    Dim n As Integer
    Dim k As Integer
    Dim i As Integer
    Dim j As Integer
    Dim x As Double
    Dim a() As Double
```

```
    Dim M() As Double
    n = Sigma.Columns.Count
    ReDim a(1 To n, 1 To n)
    ReDim M(1 To n, 1 To n)

    For i = 1 To n
        For j = 1 To n
            a(i, j) = Sigma.Cells(i, j).Value
            M(i, j) = 0
        Next j
    Next i

    For i = 1 To n
        For j = i To n
            x = a(i, j)
            For k = 1 To (i - 1)
                x = x - M(i, k) * M(j, k)
            Next k

            If j = i Then
                M(i, i) = Sqr(x)
            Else
                M(j, i) = x / M(i, i)
            End If
        Next j
    Next i

    cholesky = M
End Function
```

一个警告是：当存在任何的完全相关关系时（译者注：相关系数为 1），乔里斯基分解将会变得不稳定。一些更深入的观察和其它的分解方法参见 J$\ddot{a}$ckel(2002)。

80. 12 计算时间

如果我们有d种基础资产（维度），并希望获得精度ε，则我们为一个期权

进行定价的时间会是多长呢？

由于精度正比于时间步长 δt，反比于模拟路径数 N 的平方根，（译者注：这里的“精度”似乎称“误差”更好些，不过原书如此）因此我们必须有：

$$\delta t = O(\varepsilon) \text{ 和 } N = O(\varepsilon^{-2})$$

因此总的计算时间是：

$$O(\delta t^{-1} \times N \times d) = O(\varepsilon^{-1} \times \varepsilon^{-2} \times d) = O(d\varepsilon^{-3})$$

这里出现了系数 d，因为我们必须模拟 d 种相关资产。

注意在这个计算中我们没有能够获得对冲系数。如果我们想要计算对冲系数的话，我们将不得不将这个计算重复很多次，直到获得我们所需要的数据为止。但是，从另一方面来说，我们可以同时为多个不同的期权定价，而不需要花费太多的额外时间，因为时间主要是花费在模拟路径的生成，而不是计算收益上。

80. 13 加速收敛

在三维及三维以下的状况中，与有限差分法相比，蒙特卡罗法是没有效率的。这很自然地导致了一个问题：如何才能加速它的收敛呢？这里有几个常用方法，我将描述其中的两个。

80. 13. 1 对偶变量法

在这项技术里，我们将使用一组随机数计算两个期权的估值。我们首先使用正态分布随机数生成资产价格路径的一个实现，并计算期权收益和它的现值；现在我们使用同一组随机数，但是改变它们的正负符号，也就是从 ϕ 变换为 $-\phi$，重新生成一个实现，并计算期权的收益和现值。对期权价值的估计就是这两个值的平均。我们将这个过程重复执行多次，以获得对期权价值的精确估值。

这项技术之所以适用，是由于正态分布的对称性。在模拟中，这个对称性通过

使用对偶变量得到了保证。

80. 13. 2 控制变量技术

假设我们有两个类似的衍生品，其中的一个我们想用模拟法去估值，而另外一个具有与前者类似（但是更好）的结构，我们可以用一个显式的公式来计算它的价值。使用同一组价格路径实现去为这两个期权估值，设这两个通过蒙特卡罗方法估计出来的价值为$V_1^{'}$和$V_2^{'}$。如果第二个期权的精确价值是V_2的话，那么对于第一个期权，一个$V_1^{'}$比更好的估计是:

$$V_1^{'}-V_2^{'}+V_2$$

这个方法背后的思想是：估计值$V_1^{'}$的误差应当与$V_2^{'}$的误差相等，而后者是已知的。

这一技术的一个改进是**鞅方差削减**。在这个方法中，我们可以同时模拟同一基础资产路径的多个因变量。新随机变量的选择依据是：在每一时间步上都具有零期望。这个新的变量（控制变量）将被加到期权价值上。因为它具有零期望，所以它不会将估值搞得更坏，而如果我们精心地选择这个变量，它将可以显著地减少误差的方差。

让我们看一下在使用一个控制变量时是如何操作的。假设我们要通过模拟

$$\delta S=\mu S\delta t+\sigma S\sqrt{\delta t}\phi$$

来为合约定价。现在引入控制变量y，令其满足:

$$\delta y=f\left(S,t\right)\left(\delta S-E\left[\delta S\right]\right)$$

并具有零初值。注意这个控制变量已经是零期望的了。对$f\left(S,t\right)$的选择将在后面讨论。对期权价值的新的估计就是

$$\overline{V} = \alpha e^{-r(T-t)} \overline{y}$$

这里$\overline{V}$是一般的蒙特卡罗估值，而$\overline{y}$是控制变量在所有路径实现到期日上的值的平均。对α的选择很简单，它应使得误差的方差最小，也就是说，它应使下式取得最小值:

$$E\left[\left(V - \alpha e^{-r(T-t)} y\right)^2\right]$$

具体的细节留给读者去完成。

函数$f(S,t)$该如何选择呢？一个自然的想法是使用期权的 Delta，因为它是与这个期权问题紧密相关的，这里要求它有一个封闭形式解。这个选择对应于 Delta 对冲的一个近似形式，并可以在每个模拟路径上都减少合约的价值波动。

80. 14 蒙特卡罗模拟的长处和短处

很显然的，蒙特卡罗技术是一项通用而且有力的技术。这个概念很容易扩展到奇异或路径依赖期权上去——无非是模拟随机游走以及相应的现金流，估计平均收益并取它的现值。

它主要的缺点有两个。其一，和偏微分方程的有限差分解法比起来，这个方法有些慢——这种说法在问题是三维到四维的情况下通常是适用的。当有四个或四个以上的随机或路径依赖量时，蒙特卡罗法会变得相对有效率一些。其二，其在美式期权方面的应用远未明朗。在美式期权方面应用的主要问题在于: 要考虑提前执行的最优化问题。要知道何时是期权的最佳执行时间，必须要计算到到期日为止关于所有S和t的期权价值，以保证在任何时候没有任何套利机会。然而，基本形式的蒙特卡罗法只能用来在S,t空间中的一个点上对期权价值进行估计——也就是当前的价值。

因为蒙特卡罗法是用随机数序列生成有限数量的资产价格路径实现，所以在

每次模拟后所得到的期权价值会有所不同。大致来说，蒙特卡罗估计与正确的期权价格之间的误差与模拟次数的平方根的倒数相关。更确切地说，如果一次模拟所得到的期权估值的标准差是 ε，则在 N 次模拟后，误差的标准差就是 $\varepsilon/\sqrt{N}$。如果要将精度提升10倍的话，我们必须将同样的模拟执行100倍。

80. 15 美式期权

将蒙特卡罗方法应用于欧式合约的估值是很简单的，但是将它们应用于美式期权则比较困难。问题在于在求解时时间的方向。我们看到，在偏微分框架下的工作，从到期日到当前逆向进行计算是很自然的事情。如果我们使用数值方法来开展这样的工作的话，我们可以获得从当前到到期日之间每个网格点上的合约价值。也就是说，通过这种方式我们可以保证这里没有套利机会，从而保证了提前执行的约束得以满足。

当我们使用基本形式的蒙特卡罗方法为欧式期权估值时，我们只在一个点上获取期权价值，也就是当前的资产价格水平和当前时间点。我们没有获得在其它资产价格水平和其它时间点上期权价值的任何信息。所以如果我们的合约是美式的，我们无法获知在未来的某处我们是否违反了提前执行约束。

从原理上来说，我们可以使用蒙特卡罗法在资产-时间空间中的每一点上获取期权价值。在每一个需要知道其期权价值的资产价格及时间位置上，我们都进行模拟。但是在面对一个提前执行条款时，我们必须在资产-时间空间中的大量点位上开展工作，以便跟踪约束是否被违反。如果发现在资产-时间空间的某个点上，期权的价值低于它的当前收益，则我们应当把这个点标记为期权的执行点。因此，所有通过这个点的资产价格路径都会在这个点上得到期权执行的结果——如果这些路径不是在前面的点已经执行了的话。这个过程是可行的，但是其执行时间会随着估值点位的增加而呈指数增长。

在第 11 章中，我曾讲过，蒙特卡罗方法在应用于提前执行条款方面，没有

什么好的算法。当然，这并不完全正确，我们将会在下一节看到相关内容。

80. 16 朗斯塔夫和施瓦茨回归，处理美式期权的方法

在使用蒙特卡罗方法进行美式期权定价方面，有几个算法，细节可以参见 Jäckel(2002)和 Glasserman (2003)。但是在专业人员中使用最多的是方法是 Longstaff & Schwartz (2001)中提出的。这个方法包含对股票价格路径从开始到到期日的前向模拟，以获得各个路径上现金流的现值；但在每一个时间步上它又会通过一个股票价格的简单回归来检查立即执行与继续持有策略的优劣。让我们通过一个例子来说明这个算法的细节。

考虑一个敲定价格为$100的美式看跌期权，有效期为一年。股票的当前价格是$100，其波动率为20%，无风险利率为5%。

第一步：从当前到到期日，使用时间步长δt模拟大量的（N个）资产价格路径实现。如果随机游走是对数正态的，则我们将模拟：

$$S_{j+1} = S_j \exp\left(\left(r - \frac{1}{2}\sigma^2\right)\delta t + \sigma\sqrt{\delta t}\phi\right)$$

这里ϕ是正态分布随机变量，S_j是经过j个时间步后的股票价格。我们在$j=0$上开始每个价格路径，其起始值是$S_0 = 100$。图80.8中展示了10个这样的路径，其中从当前到到期日是5个时间步。它们的具体数值在图80.9中给出。

第二步：对这十个路径分别计算它们在到期日上的收益。其结果在图80.10中显示。注意有五条路径结束于实值状态而另外五条结束于虚值状态。当然，这里假设我们在到期日前没有执行这个看跌期权。我们等下会讨论这个问题。如果我们对这些收益的平均取现值的话，我们就获得了对欧式期权的一个估值。

上面的讨论告诉我们如果只在到期日执行期权将会得到什么，但是有可能在那之前我们就将期权执行了。让我们回到第四步，也就是时间0.8的位置上，考

察每一路径在这个时间点上是否应当执行期权。以下我将使用朗斯塔夫和施瓦茨符号。

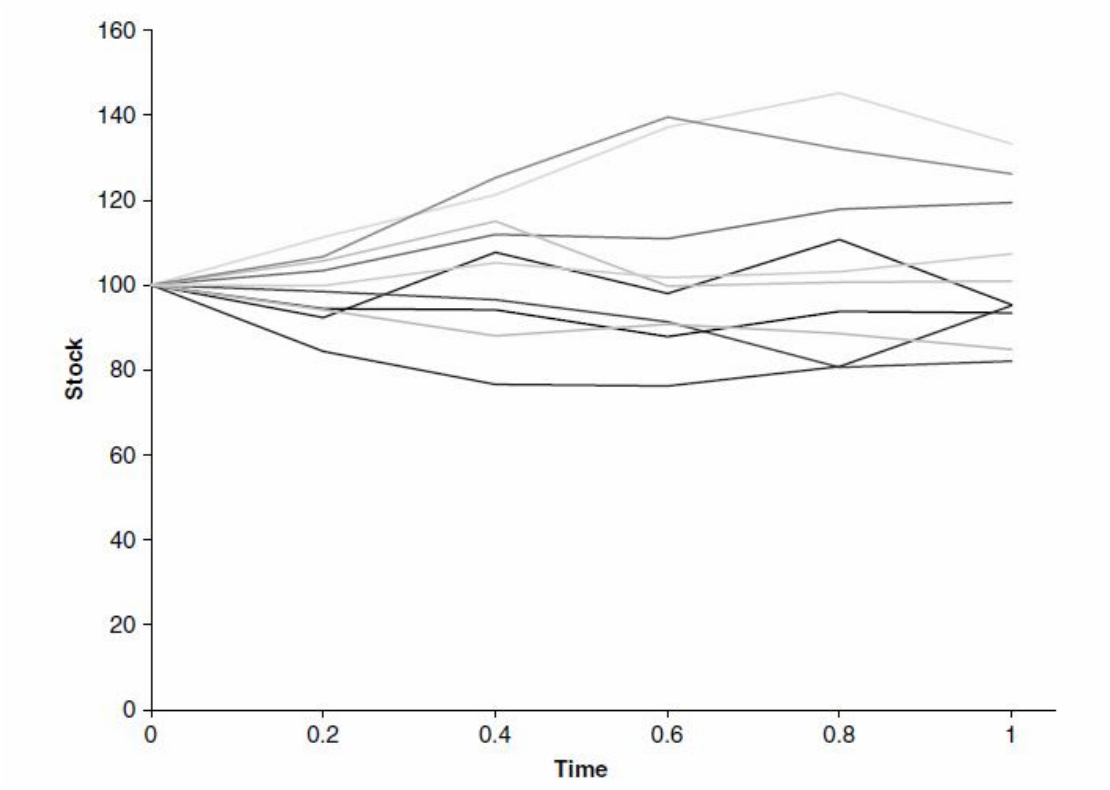


图 80.8 股票价格的十个实现

Realization	0	0.2	0.4	0.6	0.8	1
1	100	92.30759	107.7357	98.04343	110.7416	95.34586
2	100	103.4446	111.9465	110.9322	117.8379	119.4419
3	100	111.2298	121.2417	137.1683	145.1687	133.1789
4	100	105.7152	115.0572	99.73054	100.6804	100.9471
5	100	98.47278	96.5825	91.32007	80.63689	82.1163
6	100	94.40168	94.16078	87.83702	93.84797	93.45847
7	100	106.7042	125.264	139.4822	132.0177	126.2041
8	100	84.37568	76.60055	76.21345	80.85454	95.19434
9	100	94.21698	88.00477	90.81541	88.63676	84.80556
10	100	99.81029	105.2631	101.747	103.1483	107.3703

图 80.9 股票价格

在图 80.11 中有 X 列和 Y 列。X 列是时间点 0.8 上具有实值状态的那些路径的股票价格，Y 列是这些路径的收益在时间点 0.8 上的折现值。当时间点 0.8 上股

票的价格是这种情况时，我们想要计算继续持有期权的现金流状况。为了这个目的，我们将Y对取二次多项式回归。在本例中，通过最小二乘拟合我们得到：

$$Y = -0.1472X^2 + 25.347X - 1075.2 \qquad (80.5)$$

Realization	0	0.2	0.4	0.6	0.8	1	Payoff
1	100	92.30759	107.7357	98.04343	110.7416	95.34586	4.654138
2	100	103.4446	111.9465	110.9322	117.8379	119.4419	0
3	100	111.2298	121.2417	137.1683	145.1687	133.1789	0
4	100	105.7152	115.0572	99.73054	100.6804	100.9471	0
5	100	98.47278	96.5825	91.32007	80.63689	82.1163	17.8837
6	100	94.40168	94.16078	87.83702	93.84797	93.45847	6.541526
7	100	106.7042	125.264	139.4822	132.0177	126.2041	0
8	100	84.37568	76.60055	76.21345	80.85454	95.19434	4.805663
9	100	94.21698	88.00477	90.81541	88.63676	84.80556	15.19444
10	100	99.81029	105.2631	101.747	103.1483	107.3703	0

图 80.10 股票价格和收益

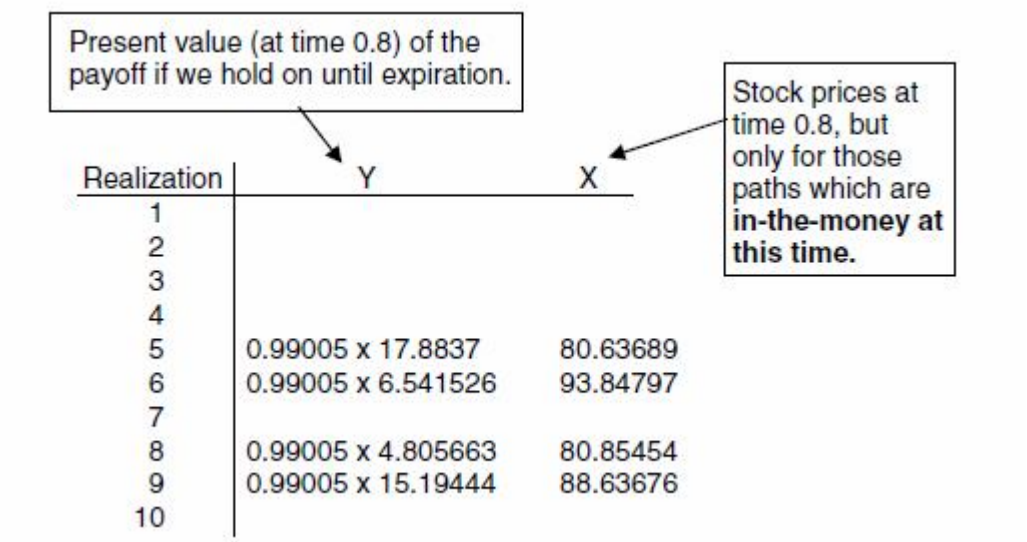


图 80.11 收益折现和股票价格，仅对于时间点0.8上的实值路径，折现率0.99005

现在，对于同一个“时间点0.8实值路径”我们比较立即执行期权的价值和继续持有期权的价值（使用式（80.5）计算）（见图 80.12）。这些数据显示，对于路径 5 和路径 8，立即执行是更优选择。

下一步是建立一个现金流矩阵，如图 80.13。它显示了当我们假设在时间点0.8之前不会执行期权时的最佳选择。注意如果在时间点0.8列上有正值，则说

明我们将在这里执行期权（如果没有更早执行的话），因此在后面的时间里现金流被设定为 0 。

很明显我们需要继续反向工作，接下来的问题是：对于每一个路径，在时间点 0.6 上执行期权是否是更优的？图 80.14 是该时间点 0.6 上的现金流，Y 列是该点上的收益折现，X 列是该点上实值路径的股票价格。注意这里有两个折现率，因为一些现金流是在一个时间步上折现而另外一些现金流是在两个时间步上折现的。现在可以给出回归：

$$Y = -0.0361X^2 + 5.6613X - 203.95$$

然后再次的，我们比较立即执行的收益 $\max(100 - X, 0)$ 和继续持有的价值 $Y = -0.0361X^2 + 5.6613X - 203.95$。之后我们可以得到一个新的现金流矩阵。

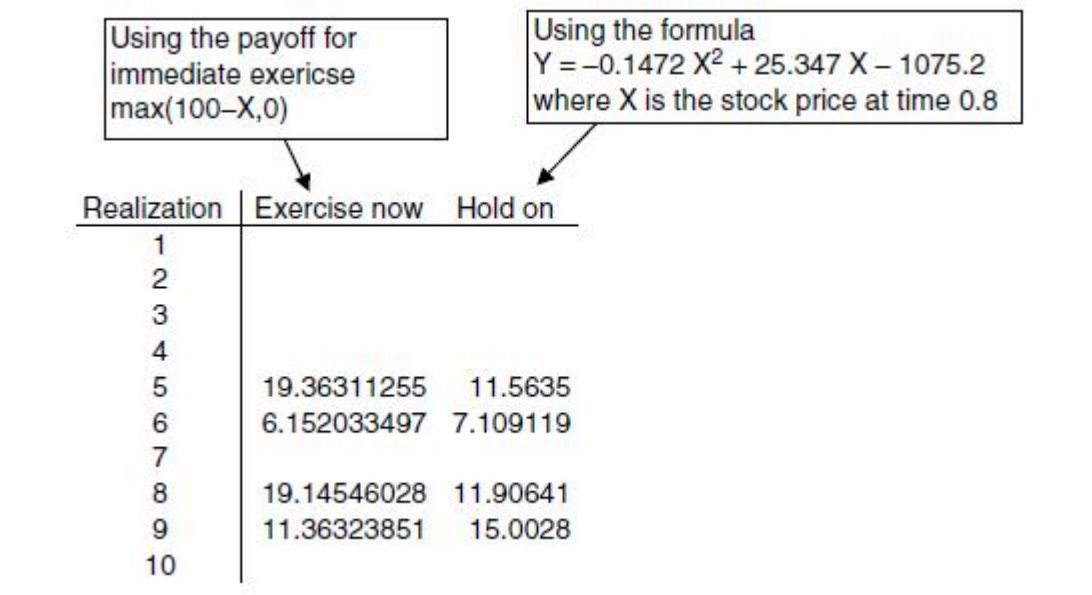
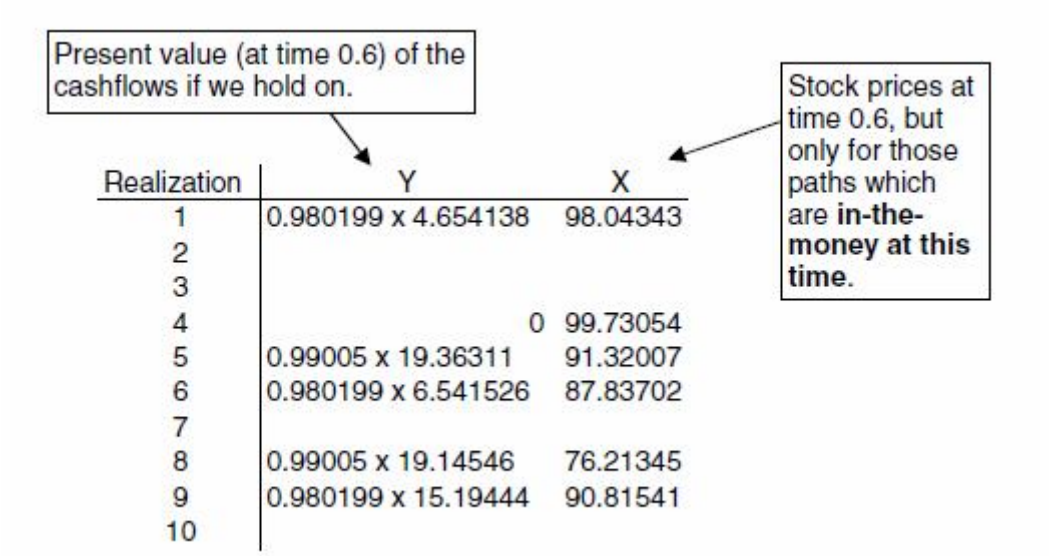


图 80.12 立即执行和继续持有的收益比较

Realization	0.2	0.4	0.6	0.8	1
1				0	4.654138
2				0	0
3				0	0
4				0	0
5				19.36311	0
6				0	6.541526
7				0	0
8				19.14546	0
8				0	15.19444
10				0	0

图 80.13 现金流矩阵



Realization	Y	X
1	0.980199 x 4.654138	98.04343
2		
3		
4	0	99.73054
5	0.99005 x 19.36311	91.32007
6	0.980199 x 6.541526	87.83702
7		
8	0.99005 x 19.14546	76.21345
9	0.980199 x 15.19444	90.81541
10		

图 80.14 时间点 0.6 上的回归

继续这个过程直到时间点 0，你会得到如图 80.15 所示的最终现金流矩阵。期权定价的最后一步是取这些现金流在时间点 0 上的现值，并取它们的平均。最终的结果是：6.409。有限差分过程给出的结果是：6.092

本算法的 VB 代码在第 83 章里。

Realization	0.2	0.4	0.6	0.8	1
1	0	0	0	0	4.6541375
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	19.363113	0
6	0	0	0	0	6.541526
7	0	0	0	0	0
8	0	0	23.786554	0	0
9	0	11.995234	0	0	0
10	0	0	0	0	0

图 80.15 最终的现金流矩阵

80. 17 基函数

在回归中我们使用了简单的二次多项式基函数，因为它易于解释，并且性能还不错。当然，你可以使用更好的基函数。朗斯塔夫和施瓦茨的原著讨论了这个问题。当在高维状况下使用这个方法时（毕竟是要考虑的），对基函数的选择就变得很重要了。

80. 18 小结

模拟是金融的核心内容。通过模拟你可以探索未知的未来，并采取相应的行动。模拟也可以用来为期权定价，虽然未来是不确定的，但是对期权进行对冲的结果在理论上是有保证的。

扩展阅读

- 蒙特卡罗模拟在衍生品定价方面的最初应用参见 Boyle (1977)
- Duffie (1992)介绍了蒙特卡罗模拟有效性背后的重要理论，并对如何提升

其效率给出了一些线索

- 在 Vose （1997）中对蒙特卡罗模拟方面的课题进行了清晰和详细的介绍。
- 蒙特卡罗方法在美式期权方面的应用概览参见 Boyle, Broadie & Glasserman (1995)
- 对蒙特卡罗模拟的一个非常技术性的考察参见 Jäckel(2002)
- 美式期权算法参见Longstaff & Schwartz (2001)，这是一项非常好的作品，精彩的表达，非常好的可理解性，以及很多具有说服力的例子。如果所有的论文都能写成这样就好了。

第 81 章　数值积分

本章内容:

- 如何在高维情况下进行数值积分以便计算一篮子期权的价格

81. 1 简介

期权的公允价值经常可以解析地写成一个积分。这当然是针对依赖于 d 种对数正态基础资产的非路径依赖欧式期权而言的。对此我们有:

$$V = e^{-r(T-t)}\left(2\pi\left(T-t\right)\right)^{-d/2}\left(\mathrm{Det}\,\Sigma\right)^{-1/2}\left(\sigma_1\cdots\sigma_d\right)^{-1}$$

$$\int_0^\infty\cdots\int_0^\infty \frac{\mathrm{Payoff}\left(S_1^{'}\cdots S_d^{'}\right)}{S_1^{'}\cdots S_d^{'}}\exp\left(-\frac{1}{2}\alpha^T\Sigma^{-1}\alpha\right)dS_1^{'}\cdots dS_d^{'}$$

这里

$$\alpha_i = \frac{1}{\sigma_i\left(T-t\right)^{1/2}}\left(\log\left(\frac{S_i}{S_i^{'}}\right)+\left(r-D_i-\frac{\sigma_i^2}{2}\right)\left(T-t\right)\right)$$

Σ 是 d 种资产间的相关矩阵，$\mathrm{Payoff}\left(\cdots\right)$ 是收益函数。有时一些路径依赖合约的价值也可以被写为多重积分的形式。不过，美式期权很少能采用如此简单的表达方式。

如果我们对一个期权的价值已经有了这样的一种表示，那么接下来我们要做的所有事情就是对这个多重积分进行估值了。让我们来看看如何去做这样的事情。

81. 2 规则格网

我们可以通过在 d 维资产空间的均匀格网上估计函数值的方式来进行多重积分的计算。当我们总共使用 N 个格点时，在每一个方向上都会有 $N^{1/d}$ 个格点。假设我们使用梯形或中点规则，则对积分的估值误差是 $O\left(N^{-2/d}\right)$。并且，由于这里有 N 次的函数估值，所以计算时间近似为 $O(N)$。当维数增长时，这个方法将会变得极端的缓慢。注意由于被积函数通常并不是光滑的，所以除非你确实要去寻找导数不连续的位置，否则在这里使用简单的中点法则和使用更高阶的方法相比没有什么不同。为了克服这个“维度诅咒”，我们可以使用蒙特卡罗积分或者低差异序列。

81. 3 基本蒙特卡罗积分

假设我们要在一定区间上计算积分:

$$\int\cdots\int f\left(x_1,\ldots,x_d\right)dx_1\ldots dx_d$$

我们可以简单地通过蒙特卡罗模拟对其进行估值。其背后的思想是：积分可以用如下方式写出:

$$\int\cdots\int f\left(x_1,\ldots,x_d\right)dx_1\ldots dx_d = \text{volume of region of integration}\times\text{average}\,f$$

这里对 f 的平均是遍及于整个被积区域的。为了简化描述，我们可以将被积区域缩放为一个单位超立方体，这样我们有:

$$\int_0^1\cdots\int_0^1 f\left(x_1,\ldots,x_d\right)dx_1\ldots dx_d = \text{average}\,f$$

这种缩放操作在金融问题中显然是必要的，因为积分的范围通常是从零到无穷大。稍后我将讨论这一点。

我们可以通过在 d 维空间中使用均匀分布随机数的蒙特卡罗采样来对 f 的

均值进行估计。在进行 N 个采样后，我们有:

$$\text{average}\, f \approx \frac{1}{N}\sum_{i=1}^{N} f\left(x_i\right) \qquad (81.1)$$

这里 x_i 是向量 $x_1,\ldots,x_d$ 在第 i 次采样中的值。当 N 增加时，其精度也会提升。式（81.1）仅仅是一个逼近。误差的大小可以通过采样平均对实际平均的标准差来衡量，也就是:

$$\sqrt{\frac{1}{N}\left(\overline{f^2}-\overline{f}^2\right)}$$

（这里必须乘上被积区域的超体积），其中:

$$\overline{f}=\frac{1}{N}\sum_{i=1}^{N} f\left(x_i\right)$$

以及

$$\overline{f^2}=\frac{1}{N}\sum_{i=1}^{N} f^2\left(x_i\right)$$

因此，当我们使用的采样点的个数为 N 时，蒙特卡罗模拟对积分进行估值的误差是 $O\left(N^{-1/2}\right)$，它不依赖于维度数。由于这里需要 N 次函数估值，所以计算时间是 $O\left(N\right)$。当我们的问题具有五维或更高的维度时，蒙特卡罗模拟比之均匀网格法具有高得多的精度。

我已经说明了在 d 维单位超立方体上的蒙特卡罗积分。在实际的金融问题中，积分的范围通常是从零到无穷大。从 0 到 1 区间转换到 0 到 ∞ 区间的方法，我们应针对当时的问题给出具体的方案。让我们考虑 d 种具有相关随机游走的资产。依据它们在时间 t 时的初始价值，它们于此时的风险中性价值可以写做:

$$S_i\left(T\right)=S_i\left(t\right)e^{\left(r-D_i-\frac{1}{2}\sigma_i^2\right)(T-t)+\sigma_i\phi_i\sqrt{T-t}}$$

随机变量 ϕ_i 是正态分布的并且是相关的。现在我们可以将欧式期权的价值写做:

$$e^{-r(T-t)}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\text{Payoff}\left(S_1(T),\ldots,S_d(T)\right)p\left(\phi_1,\ldots,\phi_d\right)d\phi_1\ldots d\phi_d$$

这里 $p\left(\phi_1,\ldots,\phi_d\right)$ 是 d 个相关正态随机变量的概率密度函数，且具有零均值和单位标准差。我不会写出 p 的显式表达式，因为只要我们能够在这个分布下产生随机数，我们就没必要去知道它的具体函数形式。事实上，我在这里所做的所有事情就是将对数正态分布的资产价格转换为正态分布的资产回报。

现在，为了给期权估值，我们必须要生成适用的正态变量。第一步是生成非相关的变量，然后是将它们转化为相关的变量。这两步在上面都已经说明过了，它们使用保克斯-穆勒法和乔里斯基分解。这样，期权的价值就可以用所有已生成随机数上收益的均值来进行估计了。

以下是一个简单的代码段，它使用 `NoPts` 个点计算 `NDim` 种资产下的欧式期权价值。利率是 `IntRate`，股息收益率是 `Div(i)`，波动率是 `Vol(i)`，到期日是 `Expiry`。资产价值的初始值是 `Asset(i)`。正态分布变量是 `x(i)`，`S(i)`是资产价格的预期值，它是对数正态分布的。

```
a = Exp(-IntRate * Expiry) / NoPts
suma = 0

For k = 1 To NoPts
    For i = 1 To NDim
        If test = 0 Then
            Do
                y = 2 * Rnd() - 1
                z = 2 * Rnd() - 1
                dist = y * y + z * z
            Loop Until dist < 1

            x(i) = y * Sqr(-2 * Log(dist) / dist)
            test = 1
        Else
            x(i) = z * Sqr(-2 * Log(dist) / dist)
            test = 0
        End If
    Next i
```

```
    For i = 1 To NDim
        S(i) = Asset(i) * Exp((IntRate - Div(i) - 0.5 * Vol(i) * Vol(i)) * _
            Expiry + Vol(i) * x(i) * Sqr(Expiry))
    Next i

    term = Payoff(S(1), S(2), S(3), S(4), S(5))
    suma = suma + term
Next k

Value = suma * a
```

这个代码段是蒙特卡罗法的最基本的形式，并且不使用下面说明的任何技巧。这些技巧中的某些实现起来比较琐碎——特别是那些不依赖于特定期权类型的技巧。

81. 4 低差异序列

基本蒙特卡罗积分的一个显著缺点是我们无法确保在 d 维空间中生成的随机点具有“良好”的分布。事实上，这里必然会有大量的聚簇现象。针对这个问题的一个方法是使用具有更好分布特性的非随机点序列。

让我们从一个蒙特卡罗法的例子开始讨论低差异序列。假设我们想要去计算一个二维积分的值，因此我们使用蒙特卡罗模拟法生成了大量的二维点，以便用它们去采样被积函数。所选出的点类似于图 81.1 的状况，请注意各点并不是均匀分散的。

现在假设我们想增加几百个点，以便提高精度。我们应当在何处放置新点？如果在其他点的中间放置新点，那么我们可以提高积分的精度。如果将新点放在距离原有点很近的地方，那么我们有可能将事情搞得更糟。

上面的状况说明，我们应当有一个选择点的方法，使得这些点不会太过的聚集，而具有良好的分散性。同时我们也希望可以在之后增加更多的点而不至于将分布搞差。很明显的，蒙特卡罗法在分布的均匀性上很差，而均匀网格在我

们加入任意数目的点后也不会继续保持均匀。**低差异序列**或**伪随机序列**具有我们需要的特性。

这里有两种低差异序列：开放的和封闭的。开放序列建立在如下假设上：我们在后面有可能要加入更多的点。而封闭序列针对给定的样本大小进行了优化，在一定数目采样点的条件下给出对积分的最佳估值。规则格网是封闭低差异序列的一个例子。我将在这里讨论开放序列。

这项技术在金融领域的第一次应用是 Barrett, Moore & Wilmott (1992).2。低差异序列有很多种，如：**Sobol'**，**Faure**，**Haselgrove** 和 **哈尔顿（Halton）**等。我将在这里讨论哈尔顿序列，因为它在目前是最易于描述的。

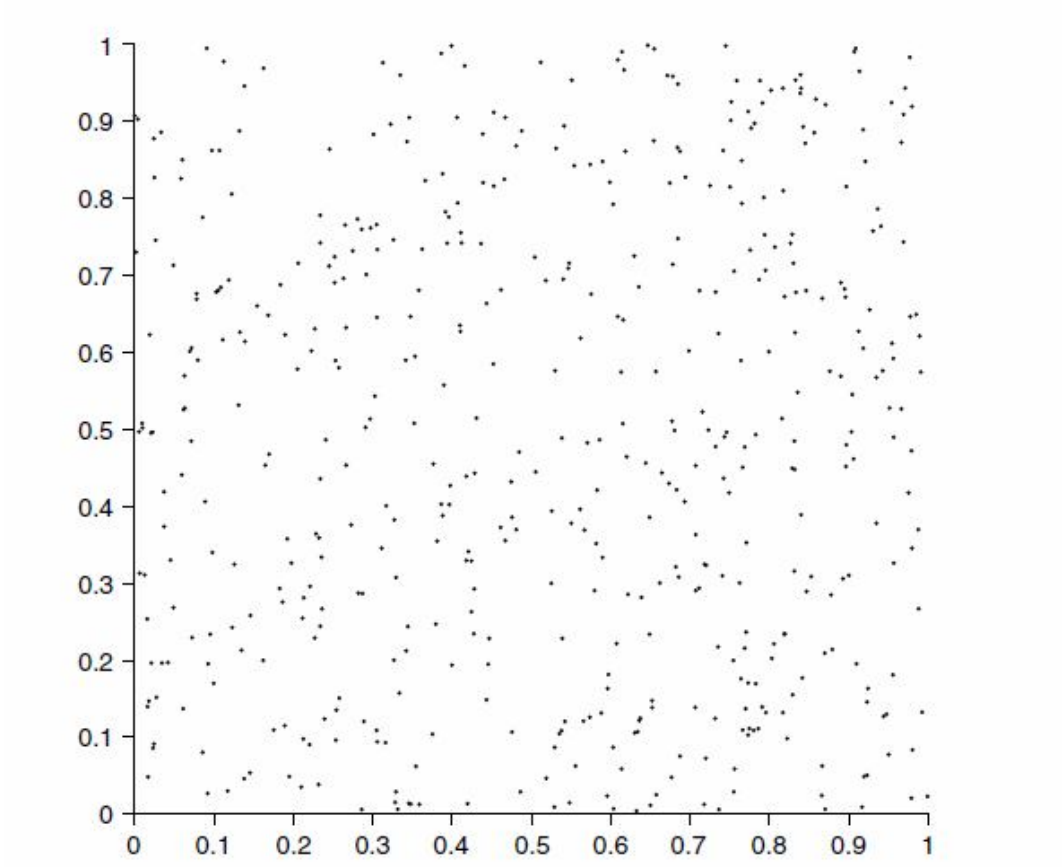


图 81.1 二维蒙特卡罗采样

哈尔顿序列是如下的数序列：$h(i;b)$，对于$i=1,2,\ldots$。整数b是基。这些数

都位于0到1之间。它们是使用如下方法构造的。首先选择你的基，让我们在这里选择2。现在我们在基2下按升序写出正整数，例如1,10,11,100,101,110,111等等。基2的哈尔顿序列是这些正整数关于小数点的镜面映射。如下图:

Integers base 10	Integers base 2	Halton sequence base 2	Halton number base 10
1	1	$1 \times \frac{1}{2}$	0.5
2	10	$0 \times \frac{1}{2} + 1 \times \frac{1}{4}$	0.25
3	11	$1 \times \frac{1}{2} + 1 \times \frac{1}{4}$	0.75
4	100	$0 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8}$	0.125
...	...	...	...

这就是所谓的关于小数点的数字反射。如果你对哈尔顿序列进行顺序描点，你会看到下一个数总是尽可能的去远离前一个点。一般地，整数可以在基b下写做:

$$i = \sum_{j=1}^{m} a_j b^j$$

这里$0 \le a_j < b$。哈尔顿数可以通过如下方法给出:

$$h(i;b) = \sum_{j=1}^{m} a_j b^{-j-1}$$

下面是一个可以计算出任意基哈尔顿数的算法。在基`b`下的哈尔顿序列中的第`n`项通过`Halton(n, b)`给出。

```
Function Halton(n, b)
    Dim n0, n1, r As Integer
    Dim h As Double
    Dim f As Double
    n0 = n
    h = 0
    f = 1 / b
```

```
    While (n0 > 0)
        n1 = Int(n0 / b)
        r = n0 - n1 * b
        h = h + f * r
        f = f / b
        n0 = n1
    Wend

    Halton = h
End Function
```

由此产生的序列性能很好，因为当我们增加越来越多的数字、越来越多的“点”的时候，我们是越来越精细地在填充0到1这个范围。

图81.2展示了使用500个点的哈尔顿序列和保克斯-穆勒方法对正态分布进行的逼近。可以将这个分布与图 80.3 中的分布进行比较。(如果你混合使用舍选保克斯-穆勒法和低差异序列的话，你会得到一个错误的结果。参见Jäckel（2002）。)

考虑在两个维度上生成计算点，你可以选择——例如基2和基3的哈尔顿序列。此时被积函数在点 $(h(i,2),h(i,3))$ 上被计算，这里 $i=1,\ldots,N$。两个序列的基应当是素数。由此产生的这些点的分布见于图 81.3，可将其与前面的图做一对比。

对于 d 维积分的估计是:

$$\int_0^1 \cdots \int_0^1 f\left(x_1,\ldots,x_d\right) dx_1 \ldots dx_d$$

它是:

$$\frac{1}{N}\sum_{i=1}^{N} f\left(h(i;b_1),\ldots,h(i;b_n)\right)$$

这里 b_j 是不同的素数。

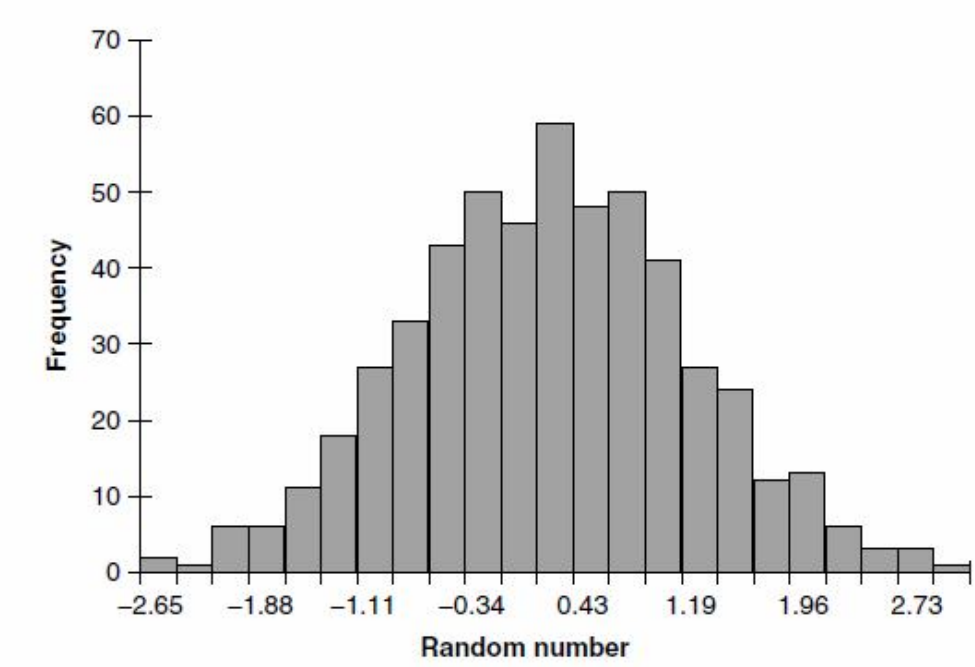


图 81.2 使用 500 个点的哈尔顿序列和保克斯-穆勒方法对正态分布进行的逼近

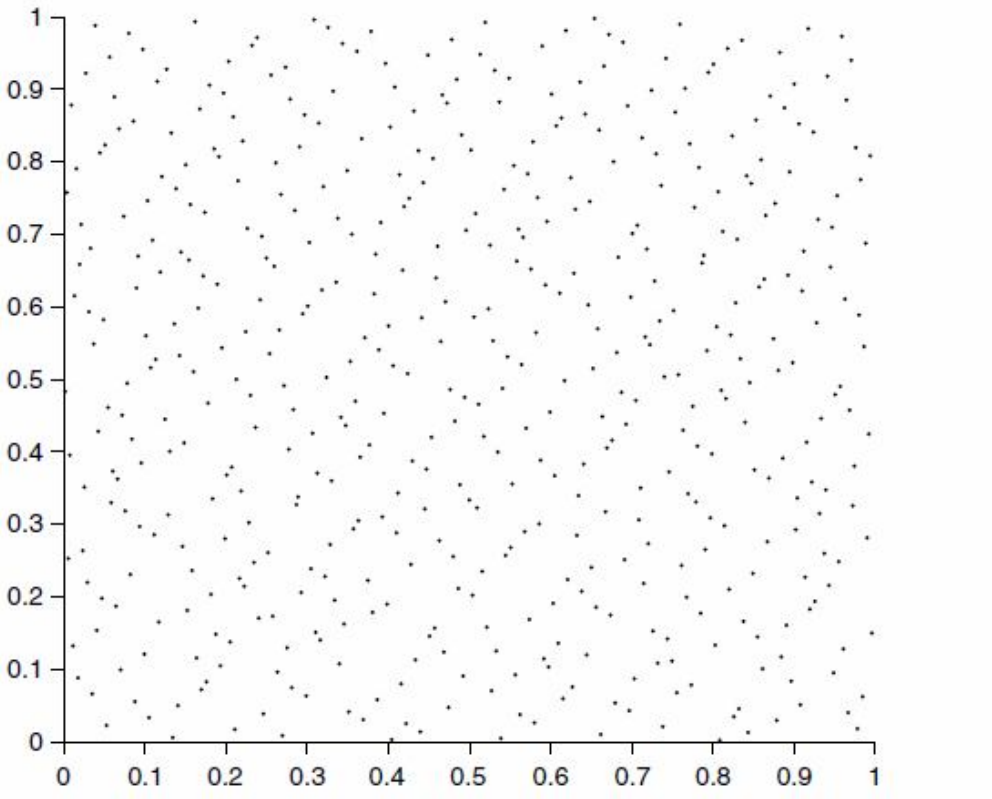


图 81.3 二维哈尔顿点

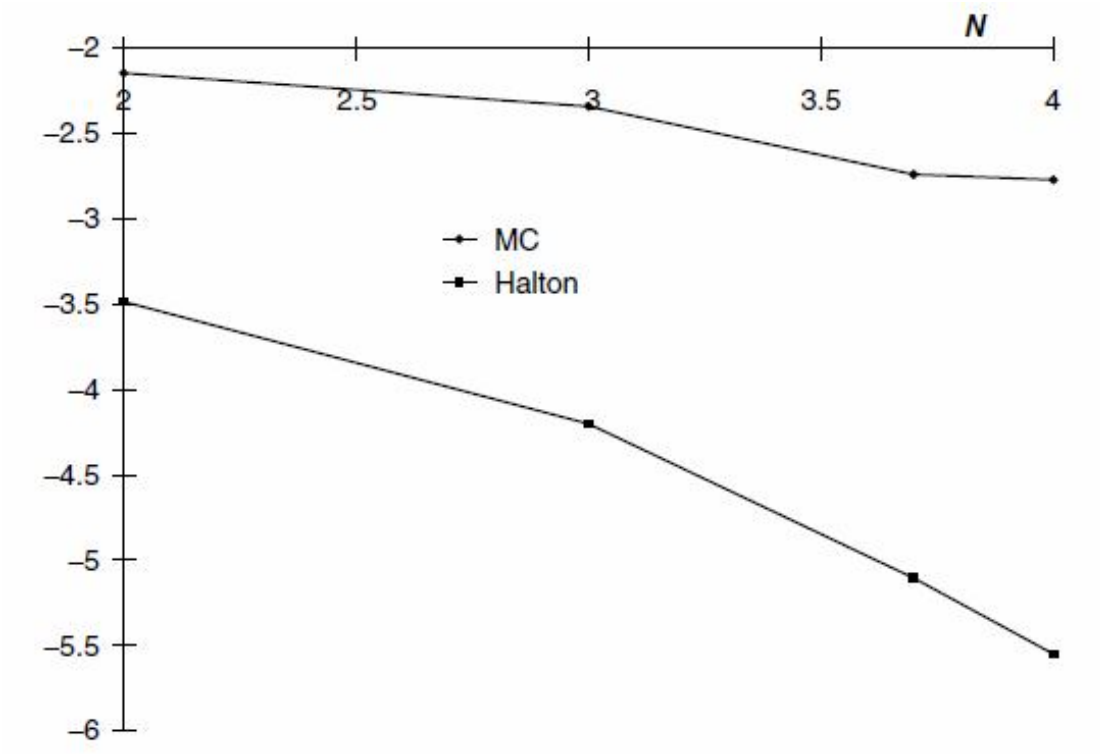


图 81.4 使用基本蒙特卡罗法和低差异序列

对一个五维合约进行估值的误差估计

伪随机法的误差是:

$$O\left((\log N)^d N^{-1}\right)$$

这在任意维度的情况下都优于蒙特卡罗法。误差表达式中的系数依赖于具体使用的低差异序列。Sobol'序列一般被认为是最好的序列,但是它非常的难于解释。生成 Sobol'序列的代码可以参见 *et al*(1995)。在三维或三维以上的状况下,低差异序列法会优于均匀网格法。它的执行时间是: $O(N)$。

很明显的,误差对维数 d 很敏感。对误差的要求反过来决定了 N 的下限,要充分认识到,这会导致一个对计算的非常大的需求。当然,即使在实践中计算的点位少一些,低差异序列法至少可以取得和蒙特卡罗方法同等的误差水平。

图 81.4 中展示了使用基本蒙特卡罗法和低差异序列对一个五维合约进行估值的误差估计。误差对哈尔顿序列采样数的反比关系十分明显。

低差异序列的另一个优点是，当你将这些点投影于一个较低的维度上(例如，将一个二维平面上的所有点投影在横轴上)，它们不会重合，它们不会投影于另一个点的投影上。这意味着如果比起其他自变量，计算结果强烈地依赖于某个特定的自变量，那么这个方法依旧可以取得一个比较精确的结果，因为低差异序列在低维上的分布特性依旧很好。

81. 5 高级技术

有几个可以用来改善蒙特卡罗和相关的数值积分法收敛性的高级技术。它们通常可以归类于**方差减少技术**，从而也提高了准确性。所有这些技术都不能提升与 N 相关的收敛性（例如，对蒙特卡罗法来说，误差依旧保持于 $O\left(N^{-1/2}\right)$)。但它们都可以显著地减小误差项中的系数。

上文所述的**对偶变量法**非常容易地应用于数值积分。它应当总是被使用，因为它不会带来任何的负面影响，并且完全独立于被估价的产品。

控制变量法可以在上述完全相同的情况下使用。与顺向模拟定价相同的，这个方法的应用依赖于是否能够得到对产品的一个具有解析形式的良好近似。

重要性采样法背后的思想是：改变自变量，从而使得被积函数尽可能的取常值。在极端情况下，如果被积函数精确地转换为常值函数，则问题的解就简单地成为新自变量的被积区间的超体积。当然，通常我们是不可能获得这么好的状况的。但是被积函数越是趋向于一个常值，那么结果的精度也就越好。这个方法很少应用于金融领域。

分层采样法主要是将被积区间划分为更小的子区间。每个子区间所分配到的采样点数可以根据每个子区间的积分的方差进行优化分配。在一维以上的情况下，如何去划分子区间，以及如何去布置网格并不总是显而易见的，有可能这个方法的应用反而削弱了蒙特卡罗法的效果。这个方法的性能可以通过**递归分层采样法**进行提升。在这个方法中，一个区间是否要被划分，取决于在这个区

间上的积分方差。分层采样法很少应用于金融领域。

81. 6 小结

低差异序列，数论与高级金融的碰撞。

扩展阅读

- 网格优化方面的细节参见 Sloan & Walsh (1990) 和 Stetson, Marshall & Loeball (1995)
- Haselgrove 方法在期权领域应用的细节参见 Barrett, Moore & Wilmott (1992)，更为详细的方法介绍参见 Haselgrove (1961)
- 一个抵押贷款证券定价的实例参见Ninomiya & Tezuka (1996)
- 更多的金融实例参见Paskov & Traub (1995), Paskov (1996) and Traub & Wozniakowski (1994)
- 对低差异序列的深入探讨参见Niederreiter (1992)
- 随机数生成和数值积分的示例代码参见杂志*et al*（1992）。你需要确认使用的是最新版，第一版上的随机数生成器并不好。他们还讨论了更先进的积分技术。

第 82 章　有限差分法程序

本章内容:

- 上述模型和有限差法方法的几个示例代码

82. 1 简介

当你开始在有限差分法方面开展工作时，可能会遇到一些困难。所以这里收集了一些基于书中所述思想的示例程序，以备参考。

82. 2 柯尔莫戈洛夫方程

以下代码计算一个随机波动率在时间 TMax 之前超出（向上或向下）SigMin 到 SigMax 之间范围的概率。波动率的现值是 SigNow，其模型是:

$$d\sigma = \nu^2 \sigma^{2\gamma-1}\left(\gamma - \frac{1}{2} - \frac{1}{2a^2}\log\left(\frac{\sigma}{\bar{\sigma}}\right)\right)dt + \nu\sigma^{\gamma}dX$$

我将时间步长和波动率步长设为固定的，你可以考虑改变这个设定。

```
Function Kolmogorov(SigNow, SigMax, SigMin, TMax, nu, SigBar, a, g)
    Dim VOld(500) As Double
    Dim VNew(500) As Double
    Dim alph(500) As Double
    Dim bet(500) As Double
    Dim Sig(500) As Double
    Dim dt As Double
    Dim ds As Double
    Dim NTS As Long
```

```
Dim NSS As Long
Dim SigNowInt As Long
Dim i As Long
Dim j As Long
Dim Delta As Double
Dim Gamma As Double
Dim frac As Double
Dim Answer As Double
dt = 0.001
ds = 0.01
NTS = Int(TMax / dt)
NSS = Int((SigMax - SigMin) / ds)
SigNowInt = Int((SigNow - SigMin) / ds)
frac = (SigNow - SigMin) / ds - SigNowInt

For i = 0 To NSS
    VOld(i) = 0
    Sig(i) = SigMin + i * ds
    alph(i) = nu * nu * (Sig(i) ^ (2 * g - 1)) * (g - 0.5 - ((Log(Sig(i) _
        / SigBar)) / (a * a * 2)))
    bet(i) = nu * (Sig(i) ^ g)
Next i
VOld(0) = 1
VOld(NSS) = 1
For j = 0 To NTS
    For i = 1 To NSS - 1
        Delta = (VOld(i + 1) - VOld(i - 1)) / 2 / ds
        Gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / ds / ds
        VNew(i) = VOld(i) + dt * (alph(i) * Delta + 0.5 * bet(i) * _
            bet(i) * Gamma)
    Next i

    VNew(0) = 1
    VNew(NSS) = 1

    For i = 0 To NSS
        VOld(i) = VNew(i)
    Next i
Next j
```

```
    If SigNow > SigMax Then
        Answer = 1
    ElseIf SigNow < SigMin Then
        Answer = 1
    Else
        Answer = frac * VOld(SigNowInt + 1) + (1 - frac) * VOld(SigNowInt)
    End If

    Kolmogorov = Answer
End Function
```

82. 3 可转换债券的显式单因素模型

下面的Visual Basic代码段展示了方程（33.1）的一个显式有限差分解法。在这里，方程是基于一个具有间歇可转换期的可转换债券的。下列代码允许有多个可转换期。每个可转换期的起始和终结点上的时间步分别被存储于数列ConvertDateStart和ConvertDateEnd中。在这些时间步之间的转换比率被存储于ConvertRate中。当时间步穿越值ConvertDateStart时，整数itest会被加一，并且程序将会开始搜索下一个可转换期。在真实时间中，这个可转换期其实是更早的，因为我们是反向进行时间步进的。

债券的价值被存储于 VOld 中，并通过 VNew 得到更新。注意我们在资产价格范围的顶端使用边界条件:

$$\frac{\partial^2 V}{\partial S^2} = 0$$

在程序中它是VNew(M) = 2 * VNew(M - 1) - VNew(M - 2)。在$S = 0$处我们有:

$$\frac{\partial V}{\partial t} = rV$$

也就是: VNew(0) = VOld(0) - dt * IntRate * VOld(0)

为了将这个代码写成一个完整的函数，我们还需要在它的开始及最后加上诸如变量声明等内容。你需要确认对时间步长 dt 的选择要足够小，以便满足稳定

性的约束。

```
For j = 1 To N
    If (test = 0) Then
        itest = itest + 1
        test = 1
    End If

    For i = 1 To M - 1
        gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / dS / dS
        sdelta = S(i) * (VOld(i + 1) - VOld(i - 1)) / 2 / dS
        cash = sdelta - VOld(i)
        VNew(i) = VOld(i) + dt * (0.5 * S(i) * S(i) * sig * sig * gamma + _
            IntRate * cash - div * sdelta)
    Next i

    VNew(0) = VOld(0) - dt * IntRate * VOld(0)
    VNew(M) = 2 * VNew(M - 1) - VNew(M - 2)

    For i1 = 0 To M
        VOld(i1) = VNew(i1)
        If j >= ConvertDateEnd(itest) And j <= ConvertDateStart(itest) Then
            VOld(i1) = max(VOld(i1), ConvertRate(itest) * S(i1))
            If j = ConvertDateStart(itest) Then test = 0
        End If
    Next i1
Next j
```

82. 4 美式看涨期权，隐式法

下列代码通过隐式格式及超松弛迭代法为一个美式看涨期权定价。当前资产价值是 Asset，波动率是 Sigma，瞬时利率是 IntRate，股息收益率是 Div，期权到期日是 ExpTime，执行价格是 Strike。这段代码同时接受资产价格步数和时间步长参数：NoAssetSteps 和 Timestep。通过实验，确定求解范围是从 $S=0$ 到 $S=3$Asset。注意我们在求解范围周边应用的边界条件。

```
Function ImplicitUSCall(Asset, ExpTime, Sigma, IntRate, Div, Strike, _
NoAssetSteps, Timestep)
    Dim VNew(0 To 1000), VOld(0 To 1000), g(0 To 1000), _
        a(0 To 1000), b(0 To 1000), c(0 To 1000) As Double
    M = NoAssetSteps
    dS = Strike / M    '资产步长
    frac = (Asset - M * dS) / dS
    M = 3 * M
    dt = Timestep
    kmax = Int(ExpTime / dt + 0.5)
    dt = ExpTime / kmax
    w = 1.5   '松弛参数

    ' 设置看涨收益下的初始条件
    For i = 0 To M
        temp = i * dS
        VOld(i) = 0
        If (temp > Strike) Then VOld(i) = temp - Strike
        g(i) = VOld(i)
    Next i

    ' 对于固定的波动率，利率，股息率和时间步长，预设三对角线矩阵的元素
    For i = 1 To M - 1
        a(i) = dt * 0.5 * i * (Sigma * Sigma * i + IntRate - Div)
        b(i) = 1 + dt * (Sigma * Sigma * i * i + IntRate)
        c(i) = dt * 0.5 * i * (Sigma * Sigma * i - IntRate + Div)
    Next i

    ' 开始时间步进
    For j = 1 To kmax
        t = j * dt
        For i = 1 To M - 1 ' 对于当前时间步，设置初始猜想
            VNew(i) = VOld(i)
        Next i

        k = 0 ' 迭代计数器初始化
        ' 设置边界条件
        VNew(0) = 0
        VNew(M) = ds * M * Exp(-Div * t) - Strike * Exp(-IntRate * t)
10      norm = 0 ' 设置初始方差为0
```

```
        For i = 1 To M - 1
            y = (VOld(i) + a(i) * VNew(i + 1) + c(i) * VNew(i - 1)) / b(i)
            temp = y - VNew(i)
            norm = norm + temp * temp
            VNew(i) = VNew(i) + w * temp
            If VNew(i) < g(i) Then VNew(i) = g(i)
        Next i

        k = k + 1
        If (norm > 0.0001) And (k < 50) Then GoTo 10

        For i = 1 To M - 1
            VOld(i) = VNew(i)
        Next i
    Next j
    ImplicitUSCall = frac * VNew(M / 3 + 1) + (1 - frac) * VNew(M / 3)
End Function
```

82. 5 巴黎式期权，显式法

下列代码实现了一个显式方法，这个方法是应用于一种路径依赖合约——也就是巴黎式期权的。这里期权的价值是基础资产价格和路径依赖量的函数，我们需要一个二维数组以便保持对期权价值的跟踪。所有的函数输入都是明晰的，除了 BarTime 之外。它是为了触发障碍条件而必须满足的资产价格持续于障碍位之外的时间长度。思考一下这是哪一类巴黎式期权？上升还是下降？敲入还是敲出？

```
Function ParisianPut(Asset, ExpTime, Sigma, IntRate, Div, BarTime, Barrier, _
Strike, NoAssetSteps)
    Dim VNew(0 To 300, 0 To 500), VOld(0 To 300, 0 To 500), _
    q(0 To 300, 0 To 500) As Double
    dS = Strike / NoAssetSteps ' 资产价格步长
    x = Int(Barrier / dS + 0.5)
    y = Int(Strike / dS + 0.5)
    z = Int(Asset / dS)
```

```
l = 2 * y + 10 ' 资产价格方向上的点数
dt = 0.9 / (l * l * Sigma * Sigma)
kmax = Int(ExpTime / dt + 1#)
kmin = Int(BarTime / dt + 0.5)
start1 = x
end1 = l - 1
start2 = 0
end2 = x

' 设置看跌收益下的初始条件
For k = 0 To kmax
    For i = 0 To l
        temp = i * dS
        VOld(i, k) = 0#
        If (k <= kmin) And (temp < Strike) Then VOld(i, k) = Strike - temp
        q(i, k) = 0#
    Next i
Next k

' 在 tau=kmin 线上重设边界条件
For i = start2 To end2
    VOld(i, kmin) = 0#
Next i

' 开始时间步进
For j = 1 To kmax - 1
    t = j * dt

    ' 在 1-d 区域里求解B-S方程
    For i = start1 To end1
        VNew(i, 0) = VOld(i, 0) + dt * (0.5 * i * i * Sigma * Sigma * _
            (VOld(i + 1, 0) - 2 * VOld(i, 0) + VOld(i - 1, 0)) + 0.5 * _
            (IntRate - Div) * i * (VOld(i + 1, 0) - VOld(i - 1, 0)) _
            - IntRate * VOld(i, 0))
    Next i

    ' 在 2-d 区域里求解修改后的B-S方程
    For k = 0 To kmin - 1
        For i = start2 + 1 To end2 - 1
            VNew(i, k) = VOld(i, k + 1) + dt * (0.5 * i * i * Sigma * Sigma_
```

```
                    * (VOld(i + 1, k + 1) - 2 * VOld(i, k + 1) + _
                    VOld(i - 1, k + 1)) + 0.5 * (IntRate - Div) * i _
                    * (VOld(i + 1, k + 1) - VOld(i - 1, k + 1)) _
                    - IntRate * VOld(i, k + 1))
            Next i
        Next k

        ' 在线 k=kmin 上输出 q 数组的数据
        For i = 0 To l
            VNew(i, kmin) = q(i, j)
        Next i

        ' 在 tau>0, s=x 的情况下输出边界条件
        For k = 1 To kmin - 1
            VNew(x, k) = VNew(x, 0)
        Next k

        For k = 0 To kmin
            For i = 0 To l
                VOld(i, k) = VNew(i, k)
            Next i
            VOld(l, k) = 0#
            If (k <= kmin) Then VOld(0, k) = Strike * Exp(-IntRate * t)
        Next k
    Next j

    ParisianPut = VNew(z, 0)
End Function
```

82. 6 护照期权（Passport Option）

平凡的护照期权是基于一个交易账户结余的期权。合约的持有者交易基础资产（交易额的上限是 Limit）直到到期日 Expiry。对于一个看涨期权，在到期日他会得到收益 max(Balance, 0)，而对于看跌期权，收益是 max(-Balance, 0)。这里 Balance 是最终结余。

```
Function Passport(Asset, PType, Vol, Balance, N, Expiry, Limit)
    Dim dt As Double
    Dim dz As Double
    Dim VOld(-500 To 500) As Double
    Dim VNew(-500 To 500) As Double
    Dim zminus, zplus, s As Integer
    Dim i, Tim As Integer
    Dim z(-500 To 500) As Double
    Dim zmax As Double
    Dim Arg As Integer
    Dim frac As Double
    Dim Answer As Double
    dt = Expiry / N
    zmax = 2
    dz = 1.1 * Sqr(Vol * Vol * (1 + zmax) * (1 + zmax) * dt)
    zplus = Int(zmax / dz)
    zminus = -zplus
    q = 1

    If PType = "P" Then q = -1

    For i = zminus To zplus
        VOld(i) = Application.Max(q * i * dz, 0)
    Next i

    For Tim = 1 To N
        For i = zminus + 1 To zplus - 1
            VNew(i) = VOld(i) + 0.5 * dt * Vol * Vol * (1 + Abs(i * dz)) * _
                (1 + Abs(i * dz)) * (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) _
                / (dz * dz)
        Next i

        VNew(zminus) = 2 * VNew(zminus + 1) - VNew(zminus + 2)
        VNew(zplus) = 2 * VNew(zplus - 1) - VNew(zplus - 2)

        For i = zminus To zplus
            VOld(i) = VNew(i)
        Next i
    Next Tim
```

```
    Arg = Int(Balance / Asset / dz)
    frac = Balance / Asset / dz - Arg
    Answer = frac * VOld(Arg + 1) + (1 - frac) * VOld(Arg)
    Passport = Limit * Asset * Answer
End Function
```

82. 7 选择者护照期权（Chooser Passport Option）

选择者护照期权允许持有者选择实现看涨收益还是看跌收益。选择必须在预设的选择时间时做出。这个选择时间 ChooseTime 早于到期日 Expiry。

```
Function ChooserPassport(Asset, Expiry, ChooseTime, Vol, Balance, Limit, NTS)
    Dim Result As Double
    Dim dt As Double
    Dim dz As Double
    Dim VOld(-500 To 500) As Double
    Dim zminus, zplus, s As Integer
    Dim i, Tim As Integer
    Dim z(-500 To 500) As Double
    Dim zmax As Double
    Dim VNew(-500 To 500) As Double
    Dim payoff(-500 To 500) As Double
    Dim Arg As Integer
    Dim frac As Double
    Dim Answer As Double
    Dim N1 As Integer
    Dim N2 As Integer
    dt = Expiry / NTS
    N2 = Int(ChooseTime / dt)
    N1 = NTS - N2
    zmax = 0.5
    dz = 1.2 * Vol * (1 + zmax) * Sqr(dt)
    zplus = Int(zmax / dz)
    zminus = -zplus

    For i = zminus To 0
        VOld(i) = 0
    Next i
```

```
For i = 1 To zplus
    VOld(i) = i * dz
Next i

For Tim = 1 To N1
    For i = zminus + 1 To zplus - 1
        VNew(i) = VOld(i) + 0.5 * dt * Vol * Vol * (1 + Abs(i * dz)) _
            * (1 + Abs(i * dz)) * (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1))_
            / (dz * dz)
    Next i

    VNew(zminus) = 2 * VNew(zminus + 1) - VNew(zminus + 2)
    VNew(zplus) = 2 * VNew(zplus - 1) - VNew(zplus - 2)

    For i = zminus To zplus
        VOld(i) = VNew(i)
    Next i
Next Tim

For i = zminus To 0
    VOld(i) = VOld(i) - i * dz
Next i

For Tim = N1 + 1 To NTS
    For i = zminus + 1 To zplus - 1
        VNew(i) = VOld(i) + 0.5 * dt * Vol * Vol * (1 + Abs(i * dz)) _
            * (1 + Abs(i * dz)) * (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1))_
            / (dz * dz)
    Next i

    VNew(zminus) = 2 * VNew(zminus + 1) - VNew(zminus + 2)
    VNew(zplus) = 2 * VNew(zplus - 1) - VNew(zplus - 2)

    For i = zminus To zplus
        VOld(i) = VNew(i)
    Next i
Next Tim

Arg = Int(Balance / Asset / dz)
```

```
    frac = Balance / Asset / dz - Arg
    Answer = frac * VOld(Arg + 1) + (1 - frac) * VOld(Arg)
    ChooserPassport = Limit * Asset * Answer
End Function
```

82. 8 随机波动率下的显式法

这里是一个双因素程序，它在随机波动率条件下为欧式看涨期权定价。请思考我在这里用的是什么风险中性波动率过程。

```
Function stochvol(Expry, Strike, IntRate, SMax, SigMax, dt)
    Dim ds, dsig, a, b, c, d, rho, DeltaS, DeltaSig, DeltaSig1, DeltaSig2, _
        GammaS, GammaSig, XGamma, Theta As Double
    Dim N, i, j, k As Integer
    ds = Strike / SMax * 2
    dsig = 1 / SigMax
    N = Int(Expry / dt) + 1
    dt = Expry / N
    a = 0.2
    b = 1
    c = 1
    d = 1
    rho = 0
    ReDim S(0 To SMax) As Double
    ReDim Sig(0 To SigMax) As Double
    ReDim VOld(-1 To SMax, -1 To SigMax) As Double
    ReDim VNew(-1 To SMax, -1 To SigMax) As Double

    For i = 0 To SMax
        S(i) = ds * i
    Next i

    For j = 0 To SigMax
        Sig(j) = dsig * j
    Next j

    For i = 0 To SMax
        For j = 0 To SigMax
```

```
        VOld(i, j) = Application.Max(S(i) - Strike, 0)
    Next j
Next i

For k = 1 To N
    For i = 0 To SMax - 1
        For j = 0 To SigMax - 1
            DeltaS = (VOld(i + 1, j) - VOld(i - 1, j)) / 2 / ds
            DeltaSig1 = (VOld(i, j + 1) - VOld(i, j)) / dsig
            DeltaSig2 = (VOld(i, j) - VOld(i, j - 1)) / dsig
            DeltaSig = DeltaSig1
            If (a - b * Sig(j)) < 0 Then DeltaSig = DeltaSig2
            GammaS = (VOld(i + 1, j) - 2 * VOld(i, j) + VOld(i - 1, j)) / _
                ds / ds
            GammaSig = (VOld(i, j + 1) - 2 * VOld(i, j) + VOld(i, j - 1)) _
                /  dsig / dsig
            XGamma = (VOld(i + 1, j + 1) - VOld(i + 1, j - 1) - _
                VOld(i - 1, j + 1) + VOld(i - 1, j - 1)) / 4 / ds / dsig
            Theta = 0.5 * Sig(j) * Sig(j) * S(i) * S(i) * GammaS + IntRate_
                * S(i) * DeltaS - IntRate * VOld(i, j) + 0.5 * d * d * _
                Sig(j) * GammaSig + c * (a - b * Sig(j)) * DeltaSig + rho _
                * Sig(j) * S(i) * d * Sqr(Sig(j)) * XGamma
            VNew(i, j) = VOld(i, j) + dt * Theta
        Next j
    Next i

    For j = 0 To SigMax - 1
        VNew(SMax, j) = 2 * VNew(SMax - 1, j) - VNew(SMax - 2, j)
    Next j

    For i = 0 To SMax
        VNew(i, SigMax) = 2 * VNew(i, SigMax - 1) - VNew(i, SigMax - 2)
    Next i

    For i = 0 To SMax
        For j = 0 To SigMax
            VOld(i, j) = VNew(i, j)
        Next j
    Next i
Next k
```

```
    stochvol = VOld
End Function
```

82. 9 不确定性波动率

这里要稍微容易一些。在程序进程中，计算完 Gamma 后，开始计算 Theta 之前，你只需要加入如下代码行:

```
Vol = VolMax
If Gamma > 0 Then Vol = VolMin
```

这就是要做的所有事情。

82. 10 崩盘模型

下面的 Visual Basic 代码段是应用于第 58 章所讨论的崩盘模型的。

这里 S(i)是资产价格数组，它关于当前股票价格水平是呈对数正态分布的，也就是说 S(i) = Exp(Log(Asset) + i * step)。这与我们通常使用的资产价格等距的网格格式不同。之所以在这里使用对数等距网格会更好一些，是因为我们假设在崩盘时，资产价格是呈百分比跌落的。衍生品由布莱克-舒尔斯方程所决定的价值被存储于数组 BSOld 中，它是通过另外的有限差分格式或公式计算出来的。ik 是崩盘时资产价格跌落的网格数，亦即 ik = Int(-Log(1 - Crash) / Step)，这里 Crash 是容许的资产价格跌落百分比。因此 kstar = 1 - Exp(-ik * Step)。变量 testing 用来考察崩盘是否导致更大的期权价值。数组 VOld 和 VNew 用来存储期权价值的旧值和新值。Delta 是最优对冲系数，它可能是由布莱克-舒尔斯模型导出的，也可能是由崩盘模型导出的。

```
For i = -N + ik To N
    testing = VOld(i + 1) + (S(i) - S(i + 1) - kstar * S(i)) * _
        (VOld(i + 1) - VOld(i - 1)) / (S(i + 1) - S(i - 1))
```

```
    If BSOld(i - ik) > testing Then
    ' *************** Delta对冲
        Delta(i) = (VOld(i + 1) - VOld(i - 1)) / (S(i + 1) - S(i - 1))
        VNew(i) = (VOld(i + 1) + (S(i) - S(i + 1) + IntRate * S(i) * dt) * _
            (VOld(i + 1) - VOld(i - 1)) / (S(i + 1) - S(i - 1))) / _
            (1 + IntRate * dt)
    Else
    ' *************** 崩盘对冲
        Delta(i) = (BSOld(i - ik) - VOld(i + 1)) / (S(i) - S(i + 1) - kstar _
            * S(i))
        VNew(i) = (BSOld(i - ik) + S(i) * (kstar + IntRate * dt) * _
            (BSOld(i - ik) - VOld(i + 1)) / (S(i) - S(i + 1) - kstar * S(i)))_
            / (1 + IntRate * dt)
    End If
Next i
```

你需要在这个代码段的开始和末尾添加一些辅助内容。

考虑崩盘的力度不是某个确定值，而是某个范围的状况，如果要让这个代码段可用于这种状况，你将对代码进行怎样的修改？

82. 11 爱泼斯坦-维尔默特模型的显式解

下面的代码用来求解非确定性利率条件下的非线性一阶爱泼斯坦-维尔默特模型。付息日期参数是 input1，付息金额参数是 input2。当前瞬时利率是 rNow，rMin 和 rMax 是最小和最大的利率可能值。利率上升或下降的最大可能速率是 growth。如果要在最佳状况下为债券估值，你将如何修改程序？

```
Function EpsteinWilmott(input1, input2, rNow, rMin, rMax, growth)
    Dim VNew(500), VOld(500), CouponVal(20), CouponDate(20) As Double
    numcoup = input1.Rows.Count
    tMin = 0
    tMax = 0

    For i = 0 To numcoup - 1
        CouponDate(i) = input1(numcoup - i)
                        CouponVal(i) = input2(numcoup - i)
```

```
    tMax = Application.Max(CouponDate(i), tMax)
Next i

m = 1
' 设置可变数据
dr = 1 / 100
dt = dr / growth
rSteps = Int((rMax - rMin) / dr)
tSteps = Int((tMax - tMin) / dt)
rateinteger = (rNow - rMin) / dr
frac = (rNow - rateinteger * dr + rMin) / dr

' 设置数组初值
For i = 0 To rSteps
    VOld(i) = CouponVal(0)
Next i

' 求解方程
For j = 1 To tSteps
    ' 边界值
    If VOld(0) * (1 - rMin * dt) < VOld(1) * (1 - (rMin + 0.5 * dr) * dt)_
    Then
        VNew(0) = VOld(0) * (1 - rMin * dt)
    Else
        VNew(0) = VOld(1) * (1 - (rMin + 0.5 * dr) * dt)
    End If

    If VOld(rSteps) * (1 - rMax * dt) < VOld(rSteps - 1) _* (1 - (rMax - _
        0.5 * dr) * dt) Then
        VNew(rSteps) = VOld(rSteps) * (1 - rMax * dt)
    Else
        VNew(rSteps) = VOld(rSteps - 1) * (1 - (rMax - 0.5 * dr) * dt)
    End If

    ' 内部格点
    Spacer = rMin + dr

    For i = 1 To rSteps - 1
        If VOld(i) * (1 - Spacer * dt) < VOld(i - 1) _
            * (1 - (Spacer - 0.5 * dr) * dt) _
```

```
                And VOld(i) * (1 - Spacer * dt) < VOld(i + 1) _
                * (1 - (Spacer + 0.5 * dr) * dt) Then
                    VNew(i) = VOld(i) * (1 - Spacer * dt)
            ElseIf VOld(i - 1) * (1 - (Spacer - 0.5 * dr) * dt) < VOld(i) _
                * (1 - Spacer * dt) And VOld(i - 1) * _
                (1 - (Spacer - 0.5 * dr) * dt) < VOld(i + 1) _
                * (1 - (Spacer + 0.5 * dr) * dt) Then
                    VNew(i) = VOld(i - 1) * (1 - (Spacer - 0.5 * dr) * dt)
            Else
                VNew(i) = VOld(i + 1) * (1 - (Spacer + 0.5 * dr) * dt)
            End If

            Spacer = Spacer + dr
        Next i

        ' 付息日
        If j = Int((tMax - CouponDate(m)) / dt) Then
                                    For i = 0 To rSteps
            VNew(i) = VNew(i) + CouponVal(m)
            Next i
            m = m + 1
        End If

        ' 更新旧价值
        For i = 0 To rSteps
            VOld(i) = VNew(i)
        Next i
    Next j

    EpsteinWilmott = (1 - frac) * VOld(rateinteger) + frac * _
        VOld(rateinteger + 1)
End Function
```

82. 12 带险债券的计算

以下代码为一个具有随机违约风险$(\mathtt{hrate}, p)$的附息债券进行估值。违约率

的模型是一个通常运用于瞬时利率建模的均值回复 CIR 模型:

$$dp = \mathrm{rev}\left(\mathrm{meanh} - p\right)dt + \mathrm{vol}p^{1/2}dX$$

这里本金是1，支付期是 mat。利率 intrate 是常数并且是连续复利。对于接下来 nog 期的息票，可能会有一个滚动保证。并且，这里可能会有一个对于本金的保证。如果本金是被保证的，则有特定数目的得到保证的息票会在支付期支付，否则，这些得到保证的息票会在最开始就得到支付。这些债券是可提前偿还的。

```
Sub rbpricer(hrate, intrate, meanh, rev, vol, mat, coupon, cpa, nog, optpg, _
optcall, callfor, nots, out)
    Dim VOld(-1 To 1000) As Double
    Dim VNew(0 To 1000) As Double
    Dim p(0 To 1000) As Double
    Dim dt As Double
    Dim dp As Double
    Dim pmax As Double
    Dim nopsteps As Double
    Dim inthrate As Integer
    Dim frac As Double
    Dim j As Integer
    Dim realtime As Double
    Dim delta As Double
    Dim gamma As Double
    Dim coupdate As Double
    Dim Q As Double
    Dim Q0 As Double
    Dim Q2 As Double
    Dim k As Integer
    Dim lastone As Integer
    Dim tc As Double
    dt = mat / nots
    pmax = 1
    dp = 3 * 1.1 * Sqr(dt * pmax) * vol
    nopsteps = Int(pmax / dp)
    inthrate = Int(hrate / dp)
    frac = (hrate - dp * inthrate) / dp
```

```
For i = 0 To nopsteps
    p(i) = i * dp
    VOld(i) = 1 + coupon / cpa
Next i

Q0 = 0
lastone = 0

If optpg = True Then
    Q0 = 1
    If nog > 0 Then
        Q0 = Q0 + coupon / cpa
        nog = nog - 1
    End If
Else
    lastone = 1
End If

coupdate = mat - 1 / cpa

For j = 1 To nots
    realtime = mat - j * dt
    Q = Q0 * Exp(-intrate * j * dt)
    For k = 1 To min(nog, lastone + Int(j * dt * cpa))
        tc = j * dt - Int(j * dt * cpa) / cpa + (k - 1) * 1 / cpa
        Q = Q + coupon / cpa * Exp(-intrate * tc)
    Next k

    For i = 0 To nopsteps - 1
        If j * dt >= (nog - 1) / cpa Then
            delta = (VOld(i + 1) - VOld(i)) / dp
            If p(i) > meanh Then delta = (VOld(i) - VOld(i - 1)) / dp
            gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / dp / dp
            VNew(i) = VOld(i) + dt * (0.5 * vol * vol * p(i) * gamma + _
                rev * (meanh - p(i)) * delta - (intrate + p(i)) * VOld(i) _
                + p(i) * Q)
        Else
            If optpg Then
                VNew(i) = Q
            Else
```

```
                    delta = (VOld(i + 1) - VOld(i)) / dp
                    If p(i) > meanh Then delta = (VOld(i) - VOld(i - 1)) / dp
                    gamma = (VOld(i + 1) - 2 * VOld(i) + VOld(i - 1)) / dp / dp
                    VNew(i) = VOld(i) + dt * (0.5 * vol * vol * p(i) * gamma _
                        + rev * (meanh - p(i)) * delta - (intrate + p(i)) * _
                        VOld(i) + p(i) * (Q - Q0 * Exp(-intrate * j * dt)))
                End If
            End If
        Next i

        VNew(nopsteps) = VNew(nopsteps - 1) * VNew(nopsteps - 1) / _
            VOld(nopsteps - 2)

        For i = 0 To nopsteps
            VOld(i) = VNew(i)
        Next i

        If realtime < coupdate And realtime + dt > coupdate Then
            coupdate = coupdate - 1 / cpa
            For i = 0 To nopsteps
                VOld(i) = VNew(i) + coupon / cpa
            Next i
        End If

        If optcall Then
            For i = 0 To nopsteps
                VOld(i) = min(VOld(i), callfor)
            Next i
        End If
    Next j

    out(1) = frac * VOld(inthrate) + (1 - frac) * VOld(inthrate + 1)
    out(2) = frac * (VOld(inthrate + 1) - VOld(inthrate - 1)) / 2 / dp + _
        (1 - frac) * (VOld(inthrate + 2) - VOld(inthrate)) / 2 / dp
    out(3) = Q
End Sub
```

第 83 章　蒙特卡罗法程序

本章内容:

- 几个模拟程序的示例

83. 1 简介

下面是几个 Visual Basic 代码，它们可以帮助你在这个重要的数值方法（蒙特卡罗法）上开展工作。

83. 2 对基于一篮子资产的期权进行蒙特卡罗定价

下面的代码是为一个欧式期权进行估值的，这个欧式期权建立于 NDim 种对数正态分布的基础资产之上。它的收益被简单地定义为所有资产价格的最大值。需要输入的数组有：资产的起始价格 Asset，波动率 Vol 和股息率 DivYld。相关矩阵 Correl 作为一个方形二维数组输入。它将被子程序 NewMat 分解。

```
Function Monte_Carlo_Basket(Asset, Correl, Vol, DivYld, IntRate, Expiry, _
NoEvals, NDim)
    ReDim ux(1 To NDim) As Double
    ReDim cx(1 To NDim) As Double
    ReDim s(1 To NDim) As Double
    ReDim CholM(1 To NDim, 1 To NDim) As Double
    rootexpiry = Sqr(Expiry)
    a = Exp(-IntRate * Expiry) / NoEvals
    suma = 0

    Call NewMat(Correl, CholM, NDim)
```

```
    For k = 1 To NoEvals
        '生成非相关正态随机变量
        For i = 1 To NDim
            If test = 0 Then
                Do
                    y = 2 * Rnd() - 1
                    z = 2 * Rnd() - 1
                    dist = y * y + z * z
                Loop Until dist < 1
                ux(i) = y * Sqr(-2 * Log(dist) / dist)
                test = 1
            Else
                ux(i) = z * Sqr(-2 * Log(dist) / dist)
                test = 0
            End If
        Next i
        '将其转换为相关随机变量
        For i = 1 To NDim
            cx(i) = 0
            For j = 1 To i
                cx(i) = ux(j) * CholM(i, j) + cx(i)
            Next j
        Next i

        For i = 1 To NDim
            s(i) = Asset(i) * Exp((IntRate - DivYld(i) - 0.5 * Vol(i) * Vol(i))_
                * Expiry + Vol(i) * cx(i) * rootexpiry)
        Next i

        term = Application.Max(s)
        suma = suma + term
    Next k

    Monte_Carlo_Basket = suma * a
End Function
```

下面是乔里斯基分解的代码。

```
Sub NewMat(Correl, CholM, NDim)
' 乔里斯基分解
```

```
    Dim x As Double

    For i = 1 To NDim
        For j = 1 To NDim
            CholM(i, j) = 0
        Next j
    Next i

    For i = 1 To NDim
        For j = i To NDim
            x = Correl(i, j)
            For k = 1 To (i - 1)
                x = x - CholM(i, k) * CholM(j, k)
            Next k
            If j = i Then
                CholM(i, i) = Sqr(x)
            Else
                CholM(j, i) = x / CholM(i, i)
            End If
        Next j
    Next i
End Sub
```

83. 3 使用伪随机数对基于一篮子资产的期权进行蒙特卡罗定价

与上面同样的问题，但这里使用哈尔顿数。

```
Function Quasi_Monte_Carlo_Basket(Asset, Correl, Vol, DivYld, IntRate, _
Expiry, NoEvals, NDim)
    Dim prime(1 To 5, 1 To 2) As Integer
    Dim en(1 To 5) As Long
    ReDim ux(1 To NDim) As Double
    ReDim cx(1 To NDim) As Double
    ReDim s(1 To NDim) As Double
    ReDim CholM(1 To NDim, 1 To NDim) As Double
    prime(1, 1) = 2
    prime(1, 2) = 13
    prime(2, 1) = 3
    prime(2, 2) = 17
```

```
prime(3, 1) = 5
prime(3, 2) = 19
prime(4, 1) = 7
prime(4, 2) = 23
prime(5, 1) = 11
prime(5, 2) = 29
rootexpiry = Sqr(Expiry)
a = Exp(-IntRate * Expiry) / NoEvals
suma = 0

Call NewMat(Correl, CholM, NDim)

For k = 1 To NoEvals
    '************************
    ' 生成非相关正态随机变量
    For i = 1 To NDim
        Do
            y = 2 * Halton(en(i), prime(i, 1)) - 1
            z = 2 * Halton(en(i), prime(i, 2)) - 1
            dist = y * y + z * z
            en(i) = en(i) + 1
        Loop Until dist < 1
        ux(i) = y * Sqr(-2 * Log(dist) / dist)
    Next i

    '************************
    ' 将其转换为相关随机变量
    For i = 1 To NDim
        cx(i) = 0
        For j = 1 To i
            cx(i) = ux(i) * CholM(i, j) + cx(i)
        Next j
    Next i

    For i = 1 To NDim
        s(i) = Asset(i) * Exp((IntRate - DivYld(i) - 0.5 * Vol(i) * Vol(i))_
            * Expiry + Vol(i) * cx(i) * rootexpiry)
    Next i

    term = Application.Max(s)
```

```
        suma = suma + term

    Next k
    Quasi_Monte_Carlo_Basket = suma * a
End Function


Function Halton(n, b)
    Dim n0, n1, r As Long
    Dim h As Double
    Dim f As Double
    n0 = n
    h = 0
    f = 1 / b

    While (n0 > 0)
        n1 = Int(n0 / b)
        r = n0 - n1 * b
        h = h + f * r
        f = f / b
        n0 = n1
    Wend

    Halton = h
End Function
```

83. 4 应用于美式期权的蒙特卡罗法

下面的代码实现了朗斯塔夫和施瓦茨最小二乘回归算法。注意代码中调用了 Excel 函数 LinEst 来执行所需要的回归操作。收益函数 Payoff 是在外部定义的，在本节中它实现了一个看跌期权收益。你可以修改这个代码以便实现不同的回归基函数。

```
Function USOptionMC(SToday, Strike, Expn, Vol, IntRate, NTS, NPaths)
    ReDim Stock(0 To NTS, 1 To NPaths)
    ReDim Cashflow(1 To NTS, 1 To NPaths)
    ReDim TempArrayX(1 To NPaths, 1 To 2)
```

```
ReDim TempArrayY(1 To NPaths)
Dim TempArrayXForRegression()
Dim TempArrayYForRegression()
                          ReDim Regress(1 To 3)
TStep = Expn / NTS
Drift = (IntRate - 0.5 * Vol * Vol) * TStep
SD = Vol * Sqr(TStep)
DF = Exp(-IntRate * TStep)

' 模拟股票
For i = 1 To NPaths
    Stock(0, i) = SToday
    For j = 1 To NTS
        Stock(j, i) = Stock(j - 1, i) * Exp(Drift + SD * Norm)
    Next j
Next i

For i = 1 To NPaths
    Cashflow(NTS, i) = Payoff(Stock(NTS, i), Strike)
    ValueEuro = ValueEuro + Cashflow(NTS, i) / NPaths
Next i

ValueEuro = DF ^ NTS * ValueEuro ' 仅仅在你想得到这个值时有用

For j = NTS - 1 To 1 Step -1
    Num = 0
    For i = 1 To NPaths
        If Payoff(Stock(j, i), Strike) > 0 Then Num = Num + 1
    Next i

    ' 回归
    ReDim TempArrayXForRegression(1 To 2, 1 To Num)
    ReDim TempArrayYForRegression(1 To Num)
    k = 1

    For N = 1 To Num
        TempArrayXForRegression(1, N) = 0
        TempArrayXForRegression(2, N) = 0
        TempArrayYForRegression(N) = 0
    Next N
```

```
        For i = 1 To NPaths
            If Payoff(Stock(j, i), Strike) > 0 Then ' ITM
                TempArrayXForRegression(1, k) = Stock(j, i)
                TempArrayXForRegression(2, k) = Stock(j, i) * Stock(j, i)
                For m = 1 To NTS - j
                    TempArrayYForRegression(k) = TempArrayYForRegression(k) _
                        + DF ^ m * Cashflow(j + m, i)
                Next m
                k = k + 1
            End If
        Next i

        ' 使用 Excel 函数 LinEst 计算回归系数
Regress = Application.LinEst(TempArrayYForRegression, TempArrayXForRegression)

        For i = 1 To NPaths
            If Payoff(Stock(j, i), Strike) > Regress(1) * Stock(j, i) * _
                Stock(j, i) + Regress(2) * Stock(j, i) + Regress(3) _
                And Payoff(Stock(j, i), Strike) > 0 Then

                    ' 执行
                    Cashflow(j, i) = Payoff(Stock(j, i), Strike)

                    For m = 1 To NTS - j
                        Cashflow(j + m, i) = 0
                    Next m
            Else
                Cashflow(j, i) = 0
            End If
        Next i
    Next j

    ' 所有现金流的平均现值
    For i = 1 To NPaths
        For j = 1 To NTS
            ValueUS = ValueUS + DF ^ j * Cashflow(j, i) / NPaths
        Next j
    Next i
```

```
    USOptionMC = ValueUS
End Function


Function Payoff(S, E)
    Payoff = 0
    If S < E Then Payoff = E - S
End Function


Function Norm()
    Dim fac As Double, R As Double, X As Double, Y As Double
1   X = 2 * Rnd - 1
    Y = 2 * Rnd - 1
    R = X * X + Y * Y
    If (R >= 1) Then GoTo 1
    Norm = Y * Sqr(-2 * Log(R) / R)
End Function
```