有限差分法

伊藤 幹夫

平成 14 年 6 月 24 日

1 はじめに

有限差分法[1]は偏微分方程式の数値解を得る手段である．差分法は大変強力で柔軟性に富むテクニックからなる．正しく適用されるならば, 物理あるいは金融工学に現れる多くの偏微分方程式に対する精密な数値解を求めることができる．いうまでもないが，ここで与えるのは手短な入門であるので，差分法の基礎に触れるだけである．しかし，根本的なアイデアは，多くのより複雑な問題に対してかなりの程度そのままの形で一般化できる．

よくしられているように Black–Scholes 方程式が拡散方程式にいったん帰着されたならば，その厳密解を見つけてファイナンスの変数に変換するのは相対的に簡単である．これは，もちろん，拡散方程式が Black–Scholes 方程式と比べてはるかに単純だからである．このため，Black–Scholes 方程式を直接解くのと比較して，拡散方程式の数値解を見つけて，変数変換によって Black–Scholes 方程式の数値解に変換する方がずっと簡単である．ここでは，以上の理由から，差分法を用いて拡散方程式を解くことに集中する．これによって，可能な限り整然と，差分法の基本的なアイデアを示すことができる．

Black–Scholes 方程式に直接差分法を用いてはいけないと言っているのではない．(例えばマルチファクター・モデルなどの) 多くの問題においては，問題を (1 次元あるいは多次元の) 定数係数の拡散方程式に帰着することがふさわしくないか不可能である．この場合は Black–Scholes 方程式の一般化に対して差分法を適用する以外に選択の余地があまりない．差分法を直接的に Black–Scholes 方程式に応用するのは演習に回すことにする．

ヨーロピアン・コール・オプションに対する Black–Scholes 方程式

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0, \tag{1}$$

$$C(0,t) = 0, \qquad C(S,t) \sim S \quad \text{as } S \to \infty,$$

$$C(S,T) = \max(S-E, 0).$$

に対して変数変換

$$x = \log\frac{S}{E}, \tau = \frac{\sigma^2}{2}(T-t),$$

$$u(x,\tau) = \frac{e^{(\frac{2r}{\sigma^2}-1)x/2 + (\frac{2r}{\sigma^2}+1)^2\tau/4}}{E} C(S,t)$$

[1]以下，単に差分法と記す.

を用いると拡散方程式と境界条件

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}$$

$$u(x,0) = \max[e^{(\frac{2r}{\sigma^2}-1)x/2} - e^{(\frac{2r}{\sigma^2}+1)x/2}, 0]$$

$$\lim_{x\to\infty} u(x,\tau) = \infty$$

$$\lim_{x\to-\infty} u(x,\tau) = 0$$

に変換できる．

演習 1 上のことを確認せよ．

この方程式を解いたのち解 $u(x,\tau)$ に対して逆変換をほどこすと、オプションの価値 $V(S,t)$ が

$$V = E^{\frac{1}{2}(1+k)} S^{\frac{1}{2}(1-k)} e^{\frac{1}{8}(k+1)^2\sigma^2(T-t)} u\left(\log(S/E), \tfrac{1}{2}\sigma^2(T-t)\right)$$

ともとまる．ここで $k = r/\frac{1}{2}\sigma^2$ である．

2 差分近似

差分法の根本的なアイデアは偏微分方程式の偏微分係数を，関心のある点における Taylor 級数を用いた近似値で置き換えることである．例えば，偏微分係数 $\partial u/\partial\tau$ は増分の極限

$$\frac{\partial u}{\partial \tau}(x,\tau) = \lim_{\delta\tau\to 0} \frac{u(x,\tau+\delta\tau) - u(x,\tau)}{\delta\tau}$$

により定義される．$\delta\tau \to 0$ に関する極限をとるかわりに， $\delta\tau$ をゼロでない小さな量とすれば，近似

$$\frac{\partial u}{\partial \tau}(x,\tau) \approx \frac{u(x,\tau+\delta\tau) - u(x,\tau)}{\delta\tau} + O(\delta\tau) \tag{2}$$

を得る．これが $\partial u/\partial\tau$ の**差分近似**(finite-difference approximation) である．というのはこの近似が独立変数 u の小さいが無限小でない増分に関係するからである．この差分近似を特に**前進差分**(forward difference) とよぶ．それは，差分を τ が増加する方向で計算するからである． $O(\delta\tau)$ が示すように，$\delta\tau$ が小さいほど近似は正確になる

演習 2 $O(\delta\tau)$ の項は $u(x,\tau+\delta\tau)$ の (x,τ) における *Taylor* 展開で表せる．$u(x,\tau+\delta\tau)$ を (x,τ) の周りで *Taylor* 展開することを考えることにより，前進差分 *(2)* が，ある $0 \le \lambda \le 1$ に対して

$$\frac{u(x,\tau+\delta\tau) - u(x,\tau)}{\delta\tau} = \frac{\partial u}{\partial \tau}(x,\tau) + \frac{\partial^2 u}{\partial \tau^2}(x,\tau+\lambda\,\delta\tau)\,\delta\tau$$

を満たすことを示せ．後退差分近似 *(3)* についても同様の結果を示せ.

演習 3 $u(x,\tau+\delta\tau)$ と $u(x,\tau-\delta\tau)$ を (x,τ) の周りで *Taylor* 展開せよ．そして中心差分近似 *(4)* と *(5)* が本当に $O\Big((\delta\tau)^2\Big)$ の精度であることを示せ.

演習 4 中心対称差分近似 *(7)* が $O\Big((\delta x)^2\Big)$ の精度であることを示せ.

演習 5 直接的なアルゴリズム *(11)* において時間に関する *1* ステップあたり最低いくつの算術演算 *(*割り算と掛け算*)* が必要であるか.

他方

$$\frac{\partial u}{\partial \tau}(x,\tau) = \lim_{\delta\tau\to 0} \frac{u(x,\tau) - u(x,\tau-\delta\tau)}{\delta\tau}$$

から得る近似

$$\frac{\partial u}{\partial \tau}(x,\tau) \approx \frac{u(x,\tau) - u(x,\tau-\delta\tau)}{\delta\tau} + O(\delta\tau) \tag{3}$$

も同様に $\partial u/\partial\tau$ の差分近似としてみなせる．これを**後進型差分**(backward difference) とよぶ.

さらに

$$\frac{\partial u}{\partial \tau}(x,\tau) = \lim_{\delta\tau\to 0} \frac{u(x,\tau+\delta\tau) - u(x,\tau-\delta\tau)}{2\,\delta\tau}$$

に注意すると，この式から得られる近似

$$\frac{\partial u}{\partial \tau}(x,\tau) \approx \frac{u(x,\tau+\delta\tau) - u(x,\tau-\delta\tau)}{2\,\delta\tau} + O\Big((\delta\tau)^2\Big) \tag{4}$$

を**中央差分**(central differences) と定義できる．図 1 に，以上の 3 種類の差分の幾何的な解釈を示す．中央差分は (小さな $\delta\tau$ に対して) 前進差分と後進型差分より精密である．このことも図 1 は示唆している．(差分近似の精密さに関しては，

拡散方程式に，前進差分と後進型差分を $\partial u/\partial\tau$ に対して適用する場合，それぞれ **直接的な**(explicit) 差分スキームと**完全に間接的な**(fully implicit) 差分スキームが導かれる． (4) の形の中央差分を実際に用いることはない．というのは質の悪い数値スキーム (詳しく述べると，そもそも不安定なスキーム) が必ずできるからである．中央差分として

$$\frac{\partial u}{\partial \tau} \approx \frac{u(x,\tau+\delta\tau/2) - u(x,\tau-\delta\tau/2)}{\delta\tau} + O\Big((\delta\tau)^2\Big) \tag{5}$$

の形式を取るものは，**Crank–Nicolson** 差分スキームで用いる.

u の x に関する偏微分係数の差分近似もまったく同様に定義できる．例えば中央差分近似

$$\frac{\partial u}{\partial x}(x,\tau) \approx \frac{u(x+\delta x,\tau) - u(x-\delta x,\tau)}{2\,\delta x} + O\Big((\delta x)^2\Big) \tag{6}$$

となることが容易に分る.

注意 1 τ または t の偏微分係数に関しては *(4)* の形式の中央差分は用いられないが，x または S に関する偏微分係数の *(6)* の形式の中央差分は用いられる.

2 階の偏微分係数，例えば $\partial^2 u/\partial x^2$ に関しては，1 階の偏微分の後進型差分の前進差分として，あるいは 1 階の前進型差分の後進差分として，対称な差分を定義できる．どちらの場合も，**対称中央差分**(symmetric central-difference) 近似

$$\frac{\partial^2 u}{\partial x^2}(x,\tau) \approx \frac{u(x+\delta x,\tau) - 2u(x,\tau) + u(x-\delta x,\tau)}{(\delta x)^2} + O\Big((\delta x)^2\Big) \tag{7}$$

を得る．他の近似法もあるが，$\partial^2 u/\partial x^2$ に関しては，上の近似がよく使われる．というのは対称性が 2 階の微分の反転に関する対称性を保存するからである．すなわち， $x \mapsto -x$ の形式の反転に関して不変だからである．また，他の近似より精度が高い.

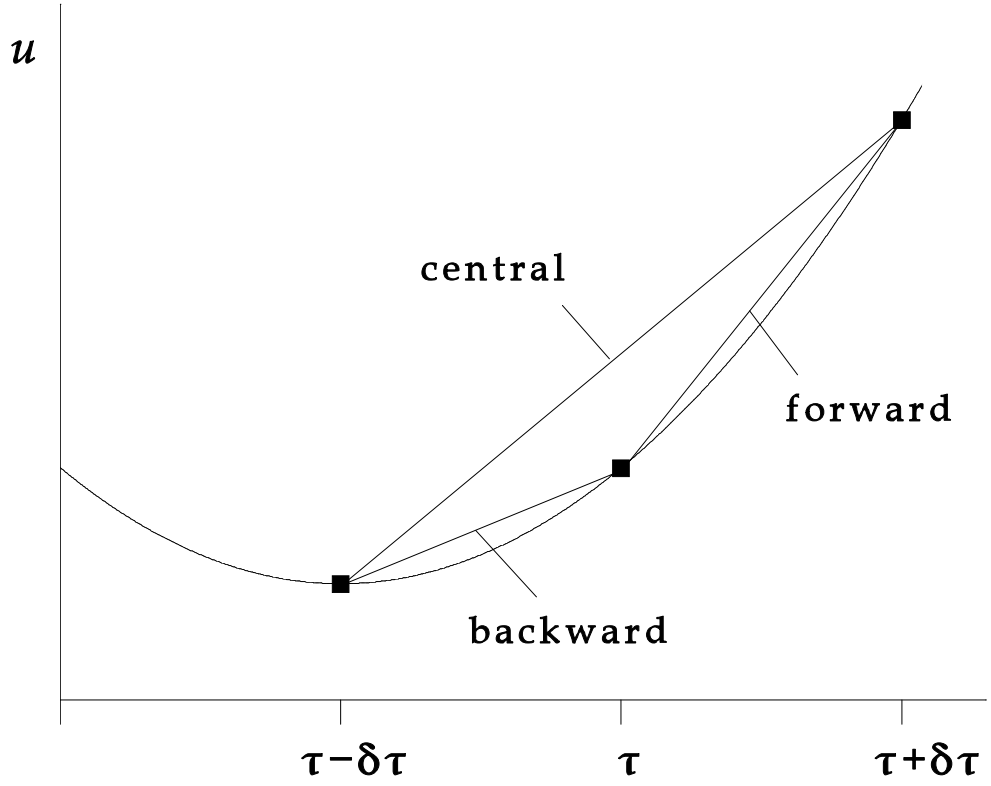
u
central
forward
backward
τ−δτ
τ
τ+δτ

図 2: 有限差分近似のメッシュ.

図 3: ある固定された時間のステップにおける有限差分近似.

3　差分のメッシュ

拡散方程式に差分近似を適用するために，x-軸を等距離 δx 離れた**ノード** (nodes) に分割し， τ を等距離 δt 離れたノードに分割する．これによって (x, τ) 平面はメッシュに分割され，**メッシュの点**(mesh points) は図 2 にあるように $(n\,\delta x, m\,\delta\tau)$ の形式をしている．この状況で $u(x, \tau)$ の点 $(n\,\delta x, m\,\delta\tau)$ における値だけに関心をもつ．図 3 を見よ．ここで $u(x, \tau)$ のメッシュの点 $(n\,\delta x, m\,\delta\tau)$ における値を

$$u_n^m = u(n\,\delta x, m\,\delta\tau) \tag{8}$$

と書く.

図 4: 直接的な差分による離散化

4 直接的な差分法

ヨーロピアン・オプションの価値に関する変換後の Black–Scholes モデルの一般形

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}$$

を境界条件と初期条件

$$\begin{aligned} u(x,\tau) \sim u_{-\infty}(x,\tau), \quad u(x,\tau) \sim u_{\infty}(x,\tau) \quad (x \to \pm\infty \text{ のとき}) \\ u(x,0) = u_0(x) \end{aligned} \tag{9}$$

の下で考えよう．ここで $u_{-\infty}(\tau)$, $u_{\infty}(\tau)$, $u_0(x)$ と記すのは，以下の議論が関係する特定の境界条件と初期条件に依存しないことを強調するためである．

u のメッシュの点における値だけに注意して，$\partial u/\partial \tau$ に関して前進差分 (2) を用い，$\partial^2 u/\partial x^2$ に関して対称中央差分 (7) を用いると，拡散方程式は

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2) \tag{10}$$

となる．ここで $O(\delta\tau)$ と $O\big((\delta x)^2\big)$ の項を無視して，整理すると，差分方程式

$$u_n^{m+1} = \alpha u_{n+1}^m + (1-2\alpha)u_n^m + \alpha u_{n-1}^m \tag{11}$$

が得られる．ここで

$$\alpha = \frac{\delta\tau}{(\delta x)^2} \tag{12}$$

である．((10) は厳密であるが，誤差はあいまいであり，(11) は近似にすぎない．)

もしステップ m においてすべての n の値について u_n^m が分っているならば，u_n^{m+1} が直接的に計算できる．これが，この方法を直接的な差分法とよぶ理由である．図 4 に示すように，u_n^{m+1} は u_{n+1}^m と u_n^m, u_{n-1}^m にしか依存しない．また，図 4 から (11) が通常の格子状を動くランダム・ウォークとみなすことができる．u_n^m は，マーカーがステップ m において位置 n にいる確率を意味する．右あるいは左に 1 単位進む確率をそれぞれ α，その位置にとどまる確率を $(1-2\alpha)$ とみなす．

xに関する間隔δxを一定に選ぶとき，xに関するステップを無限個用いないとすべての$-\infty < x < \infty$に対して問題を解くことはできない．この問題を切りぬけるために有限個の，しかし十分大きな個数の，xに関する刻みをとる．そして区間

$$N^-\delta x \le x \le N^+\delta x$$

だけに着目する．ここでN^-は絶対値が大きな負の整数，N^+は大きな正の整数である．

オプション価格の差分解を得るために，次元に依存しないように変換した，オプションの満期までの時間$\frac{1}{2}\sigma^2 T$をMで割って，時間に関する間隔とする．すなわち

$$\delta\tau = \frac{1}{2}\sigma^2 T/M$$

とする．このとき$N^- < n < N^+$と$0 < m \le M$に対して，差分方程式 (11) が求まる．そして，(9) の境界条件を用いて$u^m_{N^+}$と$u^m_{N^-}$を

$$\begin{aligned} u^m_{N^-} &= u_{-\infty}(N^-\delta x, m\,\delta\tau), \quad 0 < m \le M, \\ u^m_{N^+} &= u_{\infty}(N^+\delta x, m\,\delta\tau), \quad 0 < m \le M \end{aligned} \tag{13}$$

と定める．反復操作を始めるために (9) の初期条件を用いて

$$u^0_n = u_0(n\,\delta x), \quad N^- \le n \le N^+ \tag{14}$$

と定める．方程式はu^{m+1}_nをu^m_nによって直接的に定めているので，この反復プロセスを計算上のコードとして簡単に書き下すことができる．コード[2]を図5に示した．

演習 6 *Black–Scholes* 方程式を、すでに示した変換を使って拡散方程式に変換し、それを直接的な方法で数値解を求め、逆の変換を施すことで *Black–Scholes* モデルの数値解を求めよ．ただし、$x \to \pm\infty$ のときの u の値に注意すること．

表1において，ヨーロピアン・プットに対する直接的な差分解法を用いた解の計算結果と厳密なBlack–Scholesの公式を用いた計算結果を比較している．なお，(**??**) を用いて元の金融変数に変換している．) ここで意図して，δx と $\delta\tau$ をすぐ上で示した方法によって決めるのではなく，α と $\delta\tau$ を変化させている．これによって，きわめて重要な問題が明らかになる．$\alpha = 0.25$ と $\alpha = 0.5$ のときは直接的な差分法を用いた計算結果と厳密解を用いた計算結果はきわめて近い値をとる．ところが $\alpha = 0.52$ のときは，差分法によってとんでもない結果が出ている．このことが差分法の解の**安定性**(stability) の問題である．

安定性の問題が生じるのは，**有限精度の計算**(finite precision computer arithmetic) を用いて差分方程式 (11) を解いているからである．このことが (11) の**数値解**(numerical solution) に**まるめ誤差**(rounding errors) を引き起こす．システム (11) が**安定**(stable) であるというのは，まるめ誤差が反復ごとに増大されないことをいう．(11) は**不安定**(unstable) であるとは，まるめ誤差が解を求める反復の度に大きくなっていくことをいう．

システム (11) に関して

- $0 < \alpha \le \frac{1}{2}$ ならば安定 (**安定性の条件**)

[2]アルゴリズムを示すためだけの C に似た仮想言語のコードをいう．C 言語に精通した読者は，アルゴリズムを誤解することはないだろう．

```
explicit_fd( values,dx,dt,M,Nplus,Nminus )
{
  a = dt/(dx*dx);

  for( n=Nminus; n<=Nplus; ++n )
    oldu[n] = pay_off( n*dx );

  for( m=1; m<=M; ++m )
  {
    tau = m*dt;

    newu[Nminus] = u_m_inf( Nminus*dx,tau );
    newu[ Nplus] = u_p_inf(  Nplus*dx,tau );

    for( n=Nminus+1; n<Nplus; ++n )
      newu[n] = oldu[n]
              + a*(oldu[n-1]-2*oldu[n]+oldu[n+1]);

    for( n=Nminus; n<=Nplus; ++n )
      oldu[n] = newu[n];
  }

  for( n=Nminus; n<=Nplus; ++n )
    values[n] = oldu[n];
}
```

図 5: 拡散方程式に関する直接的な有限差分解を得るためのコード．`a` $= \alpha$, `tau` $= \tau$, `Nminus` $= N^-$, `Nplus` $= N^+$ という対応になっている． u_n^m の値を配列 `oldu`[ ] に， u_n^{m+1} の値を配列 `newu`[ ] に格納する．初期値 u_n^0 の値は `oldu`[] に事前に格納しておく．いったん u_n^{m+1} のすべての値が求められたら，`oldu`[ ] にコピーする．そして，このプロセスはすべての時間ステップが完了するまで繰り返す．数値解は `values`[] にコピーし，このルーチンを呼び出したプログラムに返す．

S	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 0.52$	厳密解
0.00	9.7531	9.7531	9.7531	9.7531
2.00	7.7531	7.7531	7.7531	7.7531
4.00	5.7531	5.7531	5.7531	5.7531
6.00	3.7532	3.7532	2.9498	3.7532
7.00	2.7567	2.7567	−17.4192	2.7568
8.00	1.7986	1.7985	95.3210	1.7987
9.00	0.9879	0.9879	350.5603	0.9880
10.00	0.4418	0.4419	625.0347	0.4420
11.00	0.1605	0.1607	−457.3122	0.1606
12.00	0.0483	0.0483	−208.9135	0.0483
13.00	0.0124	0.0123	40.5813	0.0124
14.00	0.0028	0.0027	−15.2150	0.0028
15.00	0.0006	0.0005	−3.1582	0.0006
16.00	0.0001	0.0001	0.7365	0.0001

表 1: Black–Scholes 方程式の厳密解を用いた計算結果と直接的な差分法を用いた計算結果の比較．ここではヨーロピアン・プットで，$E = 10$, $r = 0.05$, $\sigma = 0.20$ で満期までの期間が 6 か月のものを考えている．$\alpha > \frac{1}{2}$ とすることの効果に注目せよ．

- $\alpha > \frac{1}{2}$ ならば不安定 (**不安定性の条件**)

であることを示すことができる．

演習 7 上のことを示せ．

直接的な差分方程式がランダム・ウォークを表すと解釈すると，不安定な場合，負の確率が現れてしまう．(具体的には，マーカーが同じ値をとる確率 $1 - 2\alpha$ が負になってしまう．)

安定性の条件を満たすために，時間ステップの大きさには厳格な制限がある．安定であるためには

$$0 < \frac{\delta\tau}{(\delta x)^2} \leq \frac{1}{2}$$

あるメッシュでの計算が安定であるとき，精度を上げるために x に関するメッシュの点を例えば 2 倍にすると，時間に関する間隔のサイズを 4 分の 1 倍しなければいけない．空間に関してメッシュを 2 倍細かくすることは，空間方向の計算時間が 2 倍になる一方で，安定性を確保するために時間ステップを 4 倍細かくする必要もあり，時間方向の計算時間が 4 倍になってしまう．従って，x に関するメッシュを 2 倍に細かくすると解を得るのに全部で 8 倍時間がかかることになる．

拡散方程式に対する差分方程式の解が，$\delta x \to 0$ でかつ $\delta\tau \to 0$ とするときに

$$u_n^m \to u(n\,\delta x, m\,\delta\tau)$$

という意味で厳密解に収束するための必要十分条件は直接的な差分法が安定であることが証明できる．

以上の方法は境界条件，初期条件によらないので，直接的な差分法はより一般的な二値オプションとバリアー・オプションを扱うのにも適している

図 6: 間接的な有限差分法による離散化.

5 間接的な差分法

間接的な差分法は，安定性を確保するために直接的な方法で必要となる制限 $0 < \alpha \leq \frac{1}{2}$ から生じる限界を克服するために用いる．間接的な方法を用いると，時間のステップをとんでもなく小さくすることなく，x に関するメッシュを細かくすることができる．

間接的な方法は**連立 1 次方程式系**の解を必要とする．LU 分解と SOR 法のテクニックを用いて，方程式の数値解を求める．これらのテクニックを用いれば，時間に関する 1 ステップごとに必要とする算術的な操作の数に関して，直接的な方法と間接的な方法は時間に関するステップごとにそれぞれ $O(2N)$ と $O(4N)$ の算術操作を必要となる．ここで N は x のグリッドの数である．時間に関して必要なステップの数が少ないほど，間接的な差分法は直接的な差分法より通常，効率的となる．ここでは完全に間接的な方法と Crank–Nicolson 法の両方について考える．

6 完全に間接的な方法

完全に間接的な差分スキームは，単に**間接的な差分法**(implicit finite-difference method) として知られており，$\partial u/\partial \tau$ に関して後進型差分 (3)，そして $\partial^2 u/\partial x^2$ に関して対称中央差分 (7) を用いる．それによって方程式

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} + O\left((\delta x)^2\right)$$

を得る．ここで，これまでと同じ記法を用いた．さらに $O(\delta\tau)$ と $O((\delta x)^2)$，さらに高次の無限小を，無視して整理すると間接的な差分方程式

$$-\alpha u_{n-1}^{m+1} + (1+2\alpha)u_n^{m+1} - \alpha u_{n+1}^{m+1} = u_n^m \tag{15}$$

を得る．以前と同様に，空間の間隔と時間のステップは (12) で定まるパラメータ α を通して関係している．間接的な差分方程式 (15) においては，u_n^{m+1} と u_{n-1}^{m+1}, u_{n+1}^{m+1} がすべて u_n^m に**間接的に**依存している．新しい値は相互に関連していて，古い値だけから直接計算することができない．このスキームは図 6 に図示してある．

前節まで議論したヨーロピアン・オプションの問題を考えよう．無限個のメッシュを $x = N^-\delta x$ と $x = N^+\delta x$ で切り捨てて，N^- と N^+ を十分大きくして，重大な誤差が生じないようにしよう．

以前と同様に，u_n^0 は (14) を用いて， $u_{N^-}^{m+1}$ と $u_{N^+}^{m+1}$ は (4) によって定める．問題は，(15) から $m \geq 0$ かつ $N^- < n < N^+$ である (m,n) に対して u_n^{m+1} を求めることである．

(15) は線形システムとして

$$\begin{pmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & 0 \\ 0 & -\alpha & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & -\alpha \\ 0 & 0 & & -\alpha & 1+2\alpha \end{pmatrix} \begin{pmatrix} u_{N^-+1}^{m+1} \\ \vdots \\ u_0^{m+1} \\ \vdots \\ u_{N^+-1}^{m+1} \end{pmatrix} = \begin{pmatrix} u_{N^-+1}^{m} \\ \vdots \\ u_0^{m} \\ \vdots \\ u_{N^+-1}^{m} \end{pmatrix} + \alpha \begin{pmatrix} u_{N^-}^{m+1} \\ 0 \\ \vdots \\ 0 \\ u_{N^+}^{m+1} \end{pmatrix} = \begin{pmatrix} b_{N^-+1}^{m} \\ \vdots \\ b_0^{m} \\ \vdots \\ b_{N^+-1}^{m} \end{pmatrix} \tag{16}$$

と書くことができる．この方程式の真ん中の項の右側は，(15) において $n = N^+ - 1$ とおいて得られる終端の方程式

$$(1+2\alpha)u_{N^+-1}^{m+1} - \alpha u_{N^+-2}^{m+1} = u_{N^+-1}^{m} + \alpha u_{N^+}^{m+1}$$

から来ている．方程式 (16) はよりコンパクトな形で

$$\mathrm{M}\mathbf{u}^{m+1} = \mathbf{b}^m \tag{17}$$

と書ける．ここで $\mathbf{u}^{m+1}$ と $\mathbf{b}^m$ は $(N^+ - N^- - 1)$ 次元ベクトル

$$\mathbf{u}^{m+1} = (u_{N^-+1}^{m+1}, \ldots, u_{N^+-1}^{m+1}), \quad \mathbf{b}^m = \mathbf{u}^m + \alpha(u_{N^-}^{m+1}, 0, 0, \ldots, 0, u_{N^+}^{m+1})$$

であり，(16) の中に与えられた M は $(N^+ - N^- - 1)$ 次対称行列である．もし $\alpha \geq 0$ ならば， M が可逆であることが証明できる．

演習 8 α は正であることは、問題の性質上わかっているが、$\alpha \geq 0$ ならば， M が可逆であることを M が *3* 元であるときに確認せよ．一般の次元の場合、帰納法を使って示せ．

従って原理的には，M の逆行列 M^{-1} を用いて

$$\mathbf{u}^{m+1} = \mathrm{M}^{-1}\mathbf{b}^m \tag{18}$$

と解くことができる．よって，$\mathbf{u}^m$ から定められる $\mathbf{b}^m$ に対して $\mathbf{u}^{m+1}$ を見つけることができる．初期条件が $\mathbf{u}^0$ を定めるので，すべての $\mathbf{u}^{m+1}$ を順々に求めることができる．

実用上は逆行列を求めてから計算するより，はるかに効率のよい解法がある．行列 M は**三重対角型**(tridiagonal) である．すなわち，対角成分と上対角 (super-diagonal) 成分，下対角 (sub-diagonal) 成分を除いてゼロとなる行列である．この形のおかげで，いくつかの重要な結果が得られる．

まず，すべてのゼロは格納させる必要がなく，ゼロでない成分だけを格納すればよい．M の逆行列 M^{-1} は三重対角でなくて，より多くの記憶容量を必要とする．

注意 2 N がシステムの次元とすると，M^{-1} を記憶させるのに N^2 個の実数を必要とする．他方，M のゼロでない成分を記憶するのには $3N-2$ 必要である．さらに，M の逆行列を求める最も効率的な方法は $O(N^2)$ 個の操作を必要とする．$M^{-1}\mathbf{b}^m$ を求めるために必要な行列の積には $O(N^2)$ 個の操作を必要とする．

次に，M が三重対角型であることに着目すると，(17) を解く計算の非常に速いアルゴリズムがある．その場合 $O(N)$ オーダーの回数の算術操作が必要となる．($4N$ 回の操作が必要となる．) そのための 2 つのアルゴリズム，具体的には LU **分解**と SOR **法**を紹介する．

6.1 LU 分解

LU 分解では，行列 M が下三角行列 L と上三角行列 U の積，すなわち M = LU

$$\begin{pmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & \vdots \\ 0 & -\alpha & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -\alpha \\ 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{N^-+1} & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \ell_{N^+-2} & 1 \end{pmatrix} \begin{pmatrix} y_{N^-+1} & z_{N^-+1} & 0 & \cdots & 0 \\ 0 & y_{N^-+2} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & z_{N^+-2} \\ 0 & \cdots & 0 & 0 & y_{N^+-1} \end{pmatrix} \tag{19}$$

となる分解を求める．ℓ_n, y_n と z_n の値を，とりあえず計算で決定するために，(19) の右辺の 2 つの行列を単純に掛けて，その計算結果を左辺と比較する．単純な計算の結果，

$$\begin{aligned} y_{N^-+1} &= (1+2\alpha), \\ y_n &= (1+2\alpha) - \alpha^2/y_{n-1}, \\ n &= N^-+2, \ldots, N^+-1 \\ z_n &= -\alpha, \\ \ell_n &= -\alpha/y_n, \quad n = N^-+1, \ldots, N^+-2 \end{aligned} \tag{20}$$

(21)

を得る．この計算によって，計算して保存する必要があるのは

$$y_n, n = N^-+1, \ldots, N^+-1$$

の値だけであることも同時に示される．

演習 9 上の y_n が *1* を下界にもつ減少列であることを確認せよ．つまり、各 $y_n > 1$ である．

もともとの問題 $\mathrm{M}\mathbf{u}^{m+1} = \mathbf{b}^m$ は $\mathrm{L}(\mathrm{U}\mathbf{u}^{m+1}) = \mathbf{b}^m$ と書くことができる．この式は，途中の計算のために必要な便宜的なベクトル $\mathbf{q}^m$ を用いて 2 つの単純な問題

$$\mathrm{L}\mathbf{q}^m = \mathbf{b}^m, \quad \mathrm{U}\mathbf{u}^{m+1} = \mathbf{q}^m$$

に分解できる．下三角行列から ℓ_n と上三角行列から z_n を (21) を用いて消去すると，解法の手続きは 2 つの部分的な問題

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\dfrac{\alpha}{y_{N^-+1}} & 1 & 0 & & \vdots \\ 0 & -\dfrac{\alpha}{y_{N^-+2}} & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\dfrac{\alpha}{y_{N^+-2}} & 1 \end{pmatrix} \begin{pmatrix} q^m_{N^-+1} \\ q^m_{N^-+2} \\ \vdots \\ q^m_{N^+-2} \\ q^m_{N^+-1} \end{pmatrix} = \begin{pmatrix} b^m_{N^-+1} \\ b^m_{N^-+2} \\ \vdots \\ b^m_{N^+-2} \\ b^m_{N^+-1} \end{pmatrix} \tag{22}$$

$$\begin{pmatrix} y_{N^-+1} & -\alpha & 0 & \cdots & 0 \\ 0 & y_{N^-+2} & -\alpha & & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & & \ddots & y_{N^+-2} & -\alpha \\ 0 & \cdots & 0 & 0 & y_{N^+-1} \end{pmatrix} \begin{pmatrix} u^{m+1}_{N^-+1} \\ u^{m+1}_{N^-+2} \\ \vdots \\ u^{m+1}_{N^+-2} \\ u^{m+1}_{N^+-1} \end{pmatrix} = \begin{pmatrix} q^m_{N^-+1} \\ q^m_{N^-+2} \\ \vdots \\ q^m_{N^+-2} \\ q^m_{N^+-1} \end{pmatrix} \tag{23}$$

に帰着される．

操作上定義された便宜的な q^m_n の値は，逐次に代入していけば求まる．$n = N^- + 1$ 以外の場合は q^m_n と q^m_{n-1} が方程式によって互いに関連するが，$q^m_{N^-+1}$ の値は直接読み取ることができる．もしシステムを添字 n が増加する順に解くとすると，q^m_n を求めるときに q^m_{n-1} を使うことができる．したがって，q^m_n を

$$q^m_{N^-+1} = b^m_{N^-+1}, \quad q^m_n = b^m_n + \frac{\alpha q^m_{n-1}}{y_{n-1}}, \quad n = N^- + 2, \ldots, N^+ - 1 \tag{24}$$

により容易に求めることができる．同様に，便宜的な q^m_n の値がすでに求められているから，(23) を解くと u^m_n を n が減少する方向に求めていけばよい．この場合は直接わかる $u^{m+1}_{N^+-1}$ の値を始点として，添字 n が減少する方向に u^m_n が

$$u^{m+1}_{N^+-1} = \frac{q^m_{N^+-1}}{y_{N^+-1}}, \quad u^{m+1}_n = \frac{q^m_n + \alpha u^{m+1}_{n+1}}{y_n}, \quad n = N^+ - 2, \ldots, N^- + 1 \tag{25}$$

によって逐次求まる．(21), (24), (25) が LU アルゴリズムを次のように定める．

- (21) を用いて y_n を求める[3]
- ベクトル $\mathbf{b}^m$ が与えられているときに，(24) を用いてベクトル $\mathbf{q}^m$ を求める．(25) を用いて $\mathbf{u}^{m+1}$ を求める．

図 7 のアルゴリズムにあるコードを見れば以上のアルゴリズムの具体的内容がわかる．(上で述べたアイデアは，かなり一般の上対角成分と下対角成分が一定でなく添字によって異なる三重対角行列に対しても適用される．Black–Scholes 方程式に対して直接に間接的な差分法を適用するときに，そのような行列が現れる．)

演習 10 間接的な方法では、直接的な方法と違って安定性の問題が生じない．なぜか．LU 分解を念頭において考えてみよ．

6.2 SOR 法

前節で議論した LU 法は，1 回の計算で解を厳密に求めるという意味で，システム (17) を解くための**直接的な**方法である．別の戦略として**反復法**(iterative method) がある．反復法は直接的な方法と違って，解を推測することから始めて厳密解に収束するまで (あるいは厳密解に十分近くなるまで) 繰り返し計算を実行する．直接的な方法では，解は反復計算を用いないで求める．反復法が直接的な方法より優れているのは，直接法では簡単にはできない，アメリカン・オプションの問題や取引費用が加わる非線型モデルにそのまま適用できる点にある．プログラムを書くのがより簡単

[3] このプロセスは行列 M にのみ依存し，$\mathbf{b}^m$ には依存しない．システム $\mathrm{M}\mathbf{u}^{m+1} = \mathbf{b}^m$ を多くの時間ステップに関して解くことになるはずだが，実際 y_n を求めるのは一度だけでよく，各時間ステップに関して共通となってしまう．

```
lu_find_y( y,a,Nminus,Nplus )
{
  asq = a*a;
  y[Nminus+1] = 1+2*a;

  for( n=Nminus+2; n<Nplus; ++n )
  {
    y[n] = 1+2*a - asq/y[n-1];
    if (y[n]==0) return(SINGULAR);
  }
  return( OK );
}

lu_solver( u,b,y,a,Nminus,Nplus )
{
/* Must call lu_find_y before using this */

  q[Nminus+1] = b[Nminus+1];

  for( n=Nminus+2; n<Nplus; ++n )
    q[n] = b[n]+a*q[n-1]/y[n-1];

  u[Nplus-1] = q[Nplus-1]/y[Nplus-1];

  for( n=Nplus-2; n>Nminus; --n )
    u[n] = (q[n]+a*u[n+1])/y[n];
}
```

図 7: 3 重対角行列による LU 分解の解法のコード．変数は a $= \alpha$, asq $=\alpha^2$, Nplus $= N^+$, Nminus $= N^-$, b[n] $= b_n^m$, q[n] $= q_n^m$, u[n] $= u_n^{m+1}$, y[n] $= y_n$ という対応になっている．このルーチンは，問題を Nminus $+ 1 \leq$ n $\leq$ Nplus $- 1$ の場合に限って解く．Nplus と Nminus は，このルーチンを呼び出したプログラムによって設定される．ルーチン lu_solver を呼び出すプログラムは呼び出す前に lu_solver を呼び出して y[] を設定しなければならない．

なこともう一つの利点である．他方，反復法の欠点として，ヨーロピアン・オプションを扱う場合，直接的な方法よりいくらか計算速度が遅いことがある[4]

SOR は Successive Over-Relaxation [5]の略である．SOR アルゴリズムは反復法の一種で，**Gauss–Seidel** 法として知られる方法の精密化である．なお Gauss–Seidel 法自体は **Jacobi** 法を発展させたものである．SOR 法を説明するには，これらの 2 つのより単純な方法を最初に記述するのが一番よいだろう．これら 3 種類の方法は，どれも，システム (15) (または (16) または (17)) を形式

$$u_n^{m+1} = \frac{1}{1+2\alpha}\Big(b_n^m + \alpha(u_{n-1}^{m+1} + u_{n+1}^{m+1})\Big) \tag{26}$$

に書くことができることを基礎とする．この式は (15)，(16) または (17) の左辺における対角成分を取り出して整理したものである．

Jacobi 法の背後にあるアイデアは， $N^- + 1 \le n \le N^+ - 1$ を満たす u_n^{m+1} の初期の推測値を選び (よい推測値は，一つ前のステップの u，すなわち u_n^m である)，(26) の右辺に代入して得られる左辺の値を u_n^{m+1} に対する新たな推測値とすることである．この操作を近似計算が変化しなくなるまで (あるいは近似値が有意な大きさの変化をしなくなるまで）繰り返す．そのとき解が得られたと考える．

形式的には **Jacobi** 法を次のように定める．$u_n^{m+1,k}$ を u_n^{m+1} に対する k 番目の反復操作から得られる値としよう．従って，最初の推測値を $u_n^{m+1,0}$ と記すことにすると，$k \to \infty$ のとき $u_n^{m+1,k} \to u_n^{m+1}$ となることを期待する．すなわち，$u_n^{m+1,k}$ が与えられると，$u_n^{m+1,k+1}$ を求めるのに (26) を修正した

$$u_n^{m+1,k+1} = \frac{1}{1+2\alpha}\Big(b_n^m + \alpha(u_{n-1}^{m+1,k} + u_{n+1}^{m+1,k})\Big), \quad N^- < n < N^+ \tag{27}$$

を用いる．このプロセス全体を，誤差を測る尺度，例えば

$$\|\mathbf{u}^{m+1,k+1} - \mathbf{u}^{m+1,k}\|^2 = \sum_n \big(u_n^{m+1,k+1} - u_n^{m+1,k}\big)^2$$

が十分小さくなり，それ以上の反復が必要ないと考えられるまで繰り返す．このとき $u_n^{m+1,k+1}$ を u_n^{m+1} の値とする．任意の $\alpha > 0$ に対して，この方法で定まる数列は収束することが知られている．しかし，その議論の詳細は，この本のレベルを越える．

Gauss–Seidel 法は Jacobi 法を発展させたものである．(27) における $u_n^{m+1,k+1}$ を計算する時点において $u_{n-1}^{m+1,k+1}$ が求まっているという事実による．Gauss–Seidel 法では，この値を $u_{n-1}^{m+1,k}$ のかわりに用いる．従って，Gauss–Seidel 法は Jacobi 法における (27) のかわりに

$$u_n^{m+1,k+1} = \frac{1}{1+2\alpha}\Big(b_n^m + \alpha(u_{n-1}^{m+1,k+1} + u_{n+1}^{m+1,k})\Big), \quad N^- < n < N^+ \tag{28}$$

を用いる．Gauss–Seidel 法では最新の推測値を使用可能になったらすぐに使うのに対して，Jacobi 法では最新の推測値は**すべて**が使用可能になった時点で用いるのが，Jacobi 法と Gauss–Seidel 法の違いである．最新の情報 ($u_n^{m+1,k}$ より $u_n^{m+1,k+1}$) を用いる実際上の利点は，Gauss–Seidel 法の方が Jacobi 法に比べて速く収束するので効率的であることである．実際，Gauss–Seidel 法のほうがずっと効率的である．というのは，Gauss–Seidel 法では推測値の更新が，古い反復データを上書きする形で行われる．これに対して Jacobi 法ではすべての新しい反復データが出揃うまで，古い反復データと新しいものを同時に保存しなければならない (出揃ったときに古いデータを上書きする)．Gauss–Seidel 法の場合も同様に，すべての正の α に対して厳密な解に収束することが証明できるが，これもこの本のレベルを越える．

[4]前節で記述した LU アルゴリズムの場合，時間ステップごとに $4N$ 回の演算がある．本節で解説する SOR 法の場合は $4N \times$ (反復回数) である．大抵の場合，反復の回数は二桁か三桁である．

[5](訳注) 逐次過緩和法と訳される場合がある．

```
SOR_solver( u,b,Nminus,Nplus,a,omega,eps,loops )
{
  loops = 0;
  do
  {
    error = 0.0;
    for( n=Nminus+1; n<Nplus; ++n )
    {
      y = ( b[n]+a*(u[n-1]+u[n+1]) )/(1+2*a);
      y = u[n]+omega*(y-u[n]);
      error += (u[n]-y)*(u[n]-y);
      u[n]=y;
    }
    ++loops;
  }
  while ( error > eps );
  return(loops);
}
```

図 8: ヨーロピアン・オプション問題に対する SOR アルゴリズムのコード．ここでは a $= \alpha$, Nplus $= N^+$, Nminus $= N^-$, b[n] $= b_n^m$, u[n] $= u_n^{m+1,k}$, $u_n^{m+1,k+1}$, y $= y_n^{m+1,k+1}$, omega $= \omega$ と対応しており， eps が要求される誤差の許容範囲を表す．ルーチンは，新しい反復が始まるとすぐに古い反復に上書きする．従って，ループ内では任意の n に対して u[n $-$ 1], u[n $-$ 2] などが $u_{n-1}^{m+1,k+1}$, $u_{n-2}^{m+1,k+1}$, $\cdots$ を含む．他方，u[n $+$ 1], u[n $+$ 2], $u_{n+1}^{m+1,k}$, $u_{n+2}^{m+1,k}$, $\cdots$ を含む．アルゴリズムは，$u_n^{m+1,k+1}$ と $u_n^{m+1,k}$ の差の平方の n に関する和が誤差の許容範囲 eps より小さくなった段階で，収束したと考える．この時点で配列 u[ ] に u_n^{m+1} の SOR 解が格納されている．このルーチンは問題を Nminus $+ 1 \leq$ n $\leq$ Nplus $- 1$ の場合にだけ解く．Nplus と Nminus の値は，このルーチンを呼び出したプログラムによって設定される．このルーチンは実行された反復の回数を loops に返す．これによって，呼び出す側のルーチンが反復回数を最小化するように omega を調整できる．

SOR アルゴリズムは Gauss–Seidel 法を精密化したものである．まず自明な

$$u_n^{m+1,k+1} = u_n^{m+1,k} + (u_n^{m+1,k+1} - u_n^{m+1,k})$$

から始めよう．反復列 $u_n^{m+1,k}$ は $k \to \infty$ のとき u_n^{m+1} に収束することが期待されるので，$(u_n^{m+1,k+1} - u_n^{m+1,k})$ を $u_n^{m+1,k}$ に加えた右辺を u_n^{m+1} の厳密な値に近づくための修正項と考える．ここで過剰に修正を加えると反復列がより速く収束する可能性が浮かびあがる．実際，k が増加するとき $u_n^{m+1,k} \to u_n^{m+1}$ の収束が振動するのでなく，単調であればそのとおりになる．Gauss–Seidel 法と SOR 法の両方でも，そうなっている．すなわち

$$\begin{aligned} y_n^{m+1,k+1} &= \frac{1}{1+2\alpha}\left(b_n^m + \alpha(u_{n-1}^{m+1,k+1} + u_{n+1}^{m+1,k})\right) \\ u_n^{m+1,k+1} &= u_n^{m+1,k} + \omega(y_n^{m+1,k+1} - u_n^{m+1,k}) \end{aligned} \tag{29}$$

とおく．ここで $\omega > 1$ は**加速パラメータ**(over-correction parameter, over-relaxation parameter) とよばれる．($y_n^{m+1,k+1}$ の項は Gauss–Seidel 法における $u_n^{m+1,k+1}$ に対応する．SOR 法では $y_n^{m+1,k+1} - u_n^{m+1,k}$ を $u_n^{m+1,k+1}$ を得るために $u_n^{m+1,k}$ に加える修正項とみなす．) SOR アルゴリズムでは $\alpha > 0$ のとき $0 < \omega < 2$ を仮定すれば，反復列が (15) の厳密解に収束することが証明できる．($0 < \omega < 1$ のときアルゴリズムは過緩和 (over-relaxation) というよりは，未緩和 (under-relaxation) とよばれている．過緩和は $1 < \omega < 2$ のとき用いられ，このとき他の ω の値に比べてはるかに速く収束する．ω の最適な値は問題とする行列の次元と関係する．より一般には，問題とする行列の詳しい性質に依存する．) ω の最適値を計算，評価する方法もあるが，多くの計算を要するので，各時間ステップで SOR の反復ループの回数を最小化するように ω を変えた方が速い．図 8 には拡散方程式から得られた完全に間接的な差分方程式に対する SOR アルゴリズムを示した．

6.3 間接的な差分アルゴリズム

間接的な差分法は，(17), (あるいは (15) か (16)) を各時間ステップについて 6.1 節の LU 解法ルーチンあるいは前節の SOR 法を用いて解く．これによって，オプションの現在価値を計算することができる．LU 法を用いたアルゴリズムを図 9 で，SOR 法を用いたアルゴリズムを図 10 で示す.

表 2 では，3 か月の満期，行使価格 $E = 10$，ボラティリティー $\sigma = 0.4$，無リスク利子率 $r = 0.1$ としたときのヨーロピアン・プットの価値に関するものである．そこでは，間接的な差分法を用いて計算した解と厳密な Black–Scholes の公式を用いて計算した解を比較している．直接的な方法の場合と同様に x に関するメッシュの間隔をまず決めた上で，α を変えたとき時間が対応して変わるようにしてある．$\alpha = 0.5$, $\alpha = 1.0$, $\alpha = 5.0$ のいずれの場合も差分法で計算された解は厳密な公式から計算した解にかなり一致する．そして $\alpha > \frac{1}{2}$ であるときも数値解が不安定であるという兆しはない.

これは，直接的な差分法を用いるスキームが不安定な場合 ($\alpha > \frac{1}{2}$) でも間接的な差分法を用いるスキームが安定であることをよく表している．実際，間接的な差分法が任意の $\alpha > 0$ に対して安定であることを証明できる．その結果，間接的なアルゴリズムを用いれば，拡散方程式を直接的な方法と比べてより長い時間ステップで解くことができる．このおかげで，より効率的な数値解を得ることができる．すなわち，間接的な方法では各時間ステップで若干長い計算時間を必要とするが，時間ステップ数が少なくて済むので，結果として計算時間が短くなる．間接的な差分近似が偏

```
implicit_fd1( values,dx,dt,M,Nminus,Nplus )
{
  a = dt/(dx*dx);

  for( n=Nminus; n<=Nplus; ++n )
    values[n] = pay_off(n*dx);

  lu_find_y( y,a,Nminus,Nplus );

  for( m=1; m<=M; ++m )
  {
    tau = m*dt;

    for( n=Nminus+1; n<Nplus; ++n )
      b[n] = values[n];

    values[Nminus] = u_m_inf( Nminus*dx, tau );
    values[ Nplus] = u_p_inf(  Nplus*dx, tau );
    b[Nminus+1] += a*values[Nminus];
    b[ Nplus-1] += a*values[ Nplus];

    lu_solver( values,b,y,a,Nminus,Nplus );
  }
}
```

図 9: LU 分解を用いた間接的な差分法による解法のコード．M は時間ステップ数であり，a $= \alpha$ と対応している．ルーチン lu_solver を使うときは，その前に一度 (しかも一度だけ) lu_find_y を呼び出さなければならないことに注意しよう．そのとき，初期値を配列 values[] に格納し， lu_solver を時間ステップごとに満期時まで繰り返し呼び出す．境界条件に関する修正は終端の値 b[Nminus $+ 1$] と b[Nplus $- 1$] に対して施されることに注意しよう．

```
implicit_fd2( values,dx,dt,M,Nminus,Nplus )
{
  a = dt/(dx*dx);
  eps = 1.0e-8;
  omega = 1.0;
  domega = 0.05;
  oldloops = 10000;

  for( n=Nminus; n<=Nplus; ++n )
    values[n] = pay_off(n*dx);

  for( m=1; m<=M; ++m )
  {
    tau = m*dt;

    for( n=Nminus+1; n<Nplus; ++n )
      b[n] = values[n];

    values[Nminus] = u_m_inf( Nminus*dx, tau );
    values[ Nplus] = u_p_inf(  Nplus*dx, tau );

    SOR_solver( values,b,Nminus,Nplus,a,omega,eps,loops );
    if ( loops > oldloops ) domega *= -1.0;
    omega += domega;
    oldloops = loops;
  }
}
```

図 10: SOR 法を用いる間接的な差分解法のコード．M が時間ステップ数であり， a $= \alpha$ と対応しており，eps は誤差の許容範囲を表す．前のコードと同様に，ルーチンは最初に初期値を配列 values[ ] に代入して SOR_solver を各時間ステップごとに満期時まで繰り返し呼び出す．終端の b[Nminus $+ 1$] と b[Nplus $- 1$] において境界値に関する補正を行う必要はない．というのは SOR ルーチンが自動的にこれを行うからである．

S	$\alpha = 0.50$	$\alpha = 1.00$	$\alpha = 5.00$	厳密解
0.00	9.7531	9.7531	9.7531	9.7531
2.00	7.7531	7.7531	7.7531	7.7531
4.00	5.7531	5.7531	5.7530	5.7531
6.00	3.7569	3.7570	3.7573	3.7569
8.00	1.9025	1.9025	1.9030	1.9024
10.00	0.6690	0.6689	0.6675	0.6694
12.00	0.1674	0.1674	0.1670	0.1675
14.00	0.0327	0.0328	0.0332	0.0326
16.00	0.0054	0.0055	0.0058	0.0054

表 2: Black–Scholes 方程式の厳密解を用いた計算結果と完全に間接的な有限差分解法を用いた計算結果の比較．ここではヨーロピアン・プットを $E = 10$, $r = 0.1$,$\sigma = 0.4$ として，3 か月満期で考えている．$\alpha = 5.0$ であっても，結果が小数第 2 位まで精密である．

微分方程式の解の厳密解に収束することが証明できる．そして直接的な差分スキームと同様に，数値解が収束することと安定であることとは同値である．

7 Crank–Nicolson 法

Crank–Nicolson 法を用いると，解の安定性と収束のために何らかの制限が必要となる直接的な差分法がもつ限界を克服し，$O((\delta\tau)^2)$ の収束速度をもつようにすることができる．(間接的な方法と直接的な方法では解の収束速度は $O(\delta\tau)$ である．)

Crank–Nicolson 法の差分スキームは本質的に間接的な方法と直接的な方法の中間である．時間に関する偏微分について前進差分近似を用いれば，直接的な差分スキーム

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O\left((\delta x)^2\right)$$

を得る．そして後進型差分を用いると間接的な差分スキーム

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} + O\left((\delta x)^2\right)$$

を得る．この 2 つの差分方程式の中間が

$$\frac{u_n^{m+1} - u_n^m}{\delta\tau} \quad + \quad O(\delta\tau) = \tag{30}$$

$$\frac{1}{2}\left(\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2}\right) + O\left((\delta x)^2\right) \tag{31}$$

である．実際，(31) において精度は $O(\delta\tau)$ より良くて $O((\delta\tau)^2)$ であることがわかる．誤差項を無視すれば， Crank–Nicolson のスキーム

$$u_n^{m+1} \quad - \quad \tfrac{1}{2}\alpha(u_{n-1}^{m+1} - 2u_n^{m+1} + u_{n+1}^{m+1}) \tag{32}$$

$$= \quad u_n^m + \tfrac{1}{2}\alpha(u_{n-1}^m - 2u_n^m + u_{n+1}^m)$$

を得る．ここでは，以前と同様に，$\alpha = \delta\tau/(\delta x)^2$ である．u_n^{m+1}, u_{n-1}^{m+1} そして u_{n+1}^{m+1} がすべての u_n^m, u_{n+1}^m , u_{n-1}^m によって間接的に決定されていることに注意しよう．

この方程式を解くことは，原理的に，間接的なスキームにおける (15) を解くことと変わりない．これは，ステップ m における各 u_n^m が既知ならば (33) の右辺のすべてが直接的に評価できるからである．したがって，問題は最初に Z_n^m に関する直接的な公式

$$Z_n^m = (1-\alpha)u_n^m + \frac{1}{2}\alpha(u_{n-1}^m + u_{n+1}^m) \tag{33}$$

を計算して，次に

$$(1+\alpha)u_n^{m+1} - \frac{1}{2}\alpha(u_{n-1}^{m+1} + u_{n+1}^{m+1}) = Z_n^m \tag{34}$$

を解くことに帰着される．この 2 番目の問題は本質的に (15) を解くことと同じである．

ここで以前と同様に無限個のメッシュを $x = N^-\delta x$ と $x = N^+\delta x$ において切断しても，N^- の絶対値と N^+ を十分大きくすれば有意な誤差が生じないと仮定する．以前と同じく，(14) を用いて u_n^0 が計算でき，境界条件 (4) を用いれば $u_{N^-}^{m+1}$ と $u_{N^+}^{m+1}$ を定めることができる．