金融財務研究会　セミナー

2 項ツリーとモンテカルロ法を用いたオプション価格入門

（エクセル編）

午後の部

2021 年 6 月 29 日

13:30～16:30

講師：森谷博之

第1部：　ブロアディとグラッサーマンのモンテカルロを用いた
アメリカンオプションの価格評価
9:30～11:00

第2部：　ロングスタッフとシュワルツの最小二乗モンテカルロ法
11:10～12:30

満期より前であればいつでも行使できるオプションの価値は、満期までの原資産の価格の動きとオプション保有者の意志によって左右されます。このような金融派生商品は、アメリカンオプションと呼ばれ、上場市場と店頭市場で取引されています。しかし、このような構造は、理論価格の算出を困難にし、解析解は存在しません。動的計画法、二項モデル、モンテカルロ法などが用いられます。

モンテカルロ法を用いた方法は Boyle(1977)により導入され、二項格子を用いた方法は Cox, Ross, Rubinstein(1979)により紹介されました。モンテカルロ法の計算スピードは状態変数の数とは独立ですが、格子法ではそのようなわけにはいきません。したがって、状態変数が増えるにしたがい、格子法に比べて、モンテカルロ法は有利になります。複雑な問題を解くことを考えると、モンテカルロ法が便利です。しかし、モンテカルロ法にも問題があります。一般的には、時間が進む方向に状態変数が進み、この状態変数の発展と事前に定められた投資方針のもとで、価格の系列は決定されます。前方に向かってアルゴリズムを解きます。各時刻までに得られた情報を最大限に利用した合理的期待形成が意思決定の原動力です。時系列は独立となり、証券価格のバイアスの無い推定を可能にします。一方で早期行使が可能なオプションでは、その価格は後ろ向きアルゴリズムを用いて算出されます。後ろを見て今の決定をすることができるということは、未来が完全に予見できていることになります。時系列が独立であっても、未来が既知であれば、意思決定に影響を及ぼし、証券価格にバイアスをもたらします。その問題の解決の過程を２つの論文を通して学びます。

1） Pricing American-style securities using simulation by Broadie and Glasserman

2） Valuing American Options by Simulation: A Simple Least-Squares Approach by Longstaff and Schwartz

1.単純なモンテカルロ法による価格の算出

シミュレーションをもちいて満期だけに行使が可能なヨーロピアンコールオプションの価値を算出する際には

$$C = E[e^{-rT}(S_T - K)^+]$$

とします。その際にはリスク中立測度を用います。リスク中立測度は、オプションの在庫とそのヘッジポジションから構成されるポートフォリオをもつ投資家がリスクに見合った超過収益を要求せずに、無リスク金利だけを求めることを前提にした概念です。ブラック、ショールズ、マートンにより提唱されました。rは無リスク金利、Tはオプションの満期、S_Tは T時の価格、Kは行使価格です。アメリカンコールオプションの価値は

$$C = \max_{\tau} E[e^{-rT}(S_T - K)^+] \qquad (1)$$

として与えられます。停止時刻は$\tau \leq T$です。時刻は有限な離散時間$0 = t_0 < t_1 < \cdots < t_d = T$とし、価格の時系列もそれに同期して$S_0, S_1, \cdots S_T$とします。価格の時系列は複数生成されます。各時系列についてのオプション価格は割り引かれ、最終的に複数生成された時系列の結果を平均します。ここで、停止時刻の最適化政策は未知としますが、各生成時系列で最適停止時刻を算出します。この分析(試行)を複数回繰り返します

$$\max_{i=0,\cdots,d} e^{-rT}(S_T - K)^+$$

この時系列の推定は完全予見による解であるためにオプションの価値を過大評価します。つぎの図にあるように、最適行使領域(Optimal Exercise Region)になければオプションは満期まで行使されません。

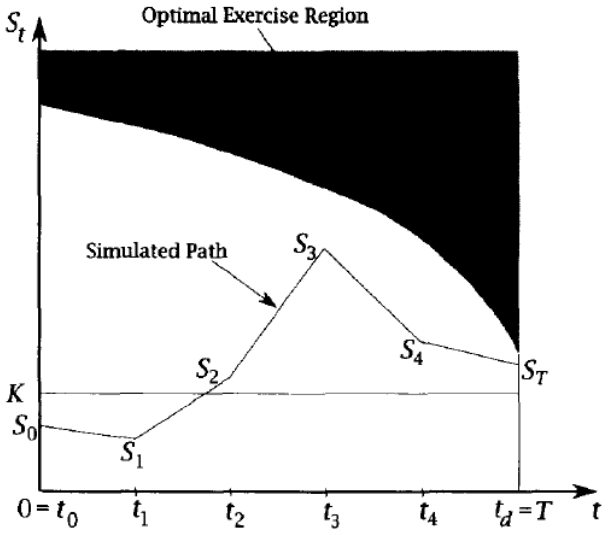

Fig. 1. Sample simulated path.

これでは

$$C = \max_{\tau} E[e^{-rT}(S_T - K)^+]$$ を実現できません。

時系列の生成数を増やしても解決にはなりません。真のオプション価格を求めるために、バイアスの無い２つの推定量(Θ, θ)、分岐点から生成される b 本の枝、限定された行使可能な時刻数を採用することで生成木を構築します。

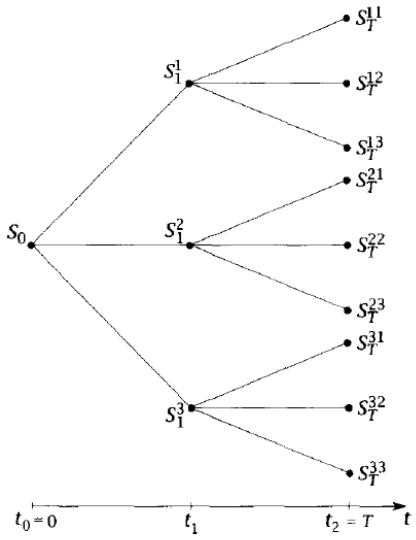

Fig. 2. Simulated tree for $b = 3$.

1.1 高バイアス推定量(Θ)

$S_0 = 101$(初期の株価)
K = 100(行使価格)
T = 1(満期)
r = 0(無リスク金利)
満期になればオプションの価値は既知となります。その他の時刻でも行使価値の最大値とその後のオプションの価値を割り引くことで計算できます。つぎの図は株価の推移です。

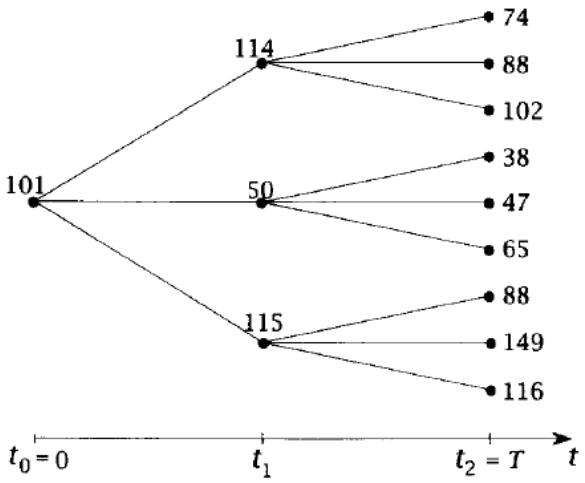

Fig. 3. Stock price tree.

Θは最初の分岐点のオプションの価値の推定値です。つぎの図は各時点のオプションの価値です。

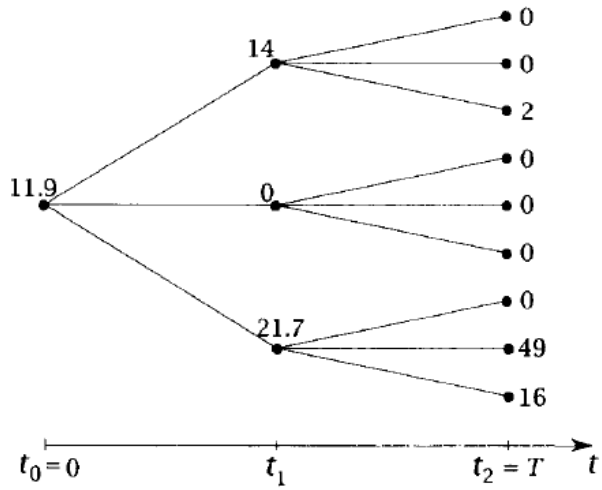

Fig. 4. Θ ('high') estimate.

$t_2 = T$

max(74-100,0)=0
max(88-100,0)=0
max(102-100,0)=2

max(38-100,0)=0
max(47-100,0)=0
max(65-100,0)=0

max(88-100,0)=0

max(149-100,0)=49

max(116-100,0)=16

t_1

max(max(114-100,0),0+0+2/3)=14

max(max(50-100,0),0)=0

max(max(115-100,0),0+49/3+16/3)=21.7

t_0

14/3+0+21.7/3=11.9

Θはつねに$E[\Theta] \geq C$になります。なぜなら生成木は株価の分布を完全に代表することはできません。分岐点のある将来の価格は非常に高いくなり、そのときには動的計画法では今の時点で行使をするという選択をしません。そうすると行使という最適な選択よりもオプションの価値は高くなってしまいます。どうように、将来の価格が低すぎれば、最適な選択が今の時点では行使をしないことでも、行使を選択します。将来の株価を知っているという利点を動的計画法が利用するために、オプションの価値は常に高く見積もられます。ここには

高く見積もるというバイアスがありますが、その

バイアスには一貫性があります。

1.2 低バイアス推定量(θ)

分岐点の枝を 2 つに分けることで、高バイアスの推定量を取り除き、低バイアスの推定量を導入します。枝を 2 つのグループに分けます。枝の第一のグループを行使するかしないかの判断に使い、第 2 のグループを継続の価値の推定に使います。この考え方はつぎの図に表されています。

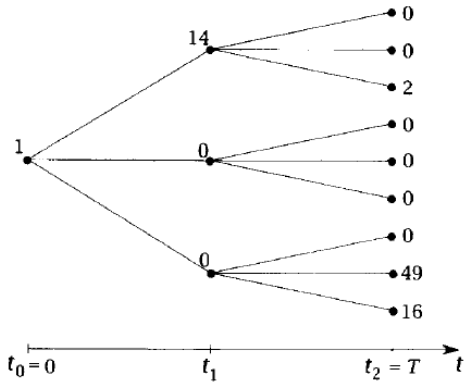

Fig. 5. Simple low estimate.

図 5 は図 3 から作られています。第 2 のグループの枝は行使の判断に使います。最初のグループの枝は継続と判断されたときに使います。t_1の第 3 の分岐点でその場で行使すればその価値は 15 となり、継続と判断すれば、その価値は 0.5×16+0.5×49=32.5 です。したがって、判断は継続となりますが、その価値はt_2の最初の枝を使うので max(88-100,0)= 0 となります。ではどのようにしてこの推定量がつねに低いバイアスをもつといえるのでしょうか？時刻を満期の 1 つ前であるとすると、行使の判断は満期のバイアスの無い情報を使って行われます。この情報から正しい判断をすればバイアスはありません。しかし、限られた標本では、疑似最適な判断をしています。この分岐点の価値は間違った判断に関連して低い価値となりその価値をバイアス無しで推定します。この

分岐点の

期待値は

正しいバイアスの無い推定値

と

バイアスのある低い推定値

の

重みつき平均値

となります。したがって、

低い推定値を

安定して

もたらします。この方法による推定値はつぎのとおりです。

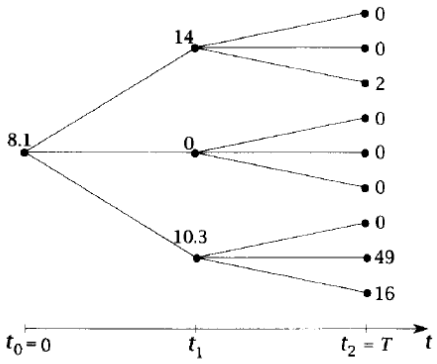

Fig. 6. θ ('low') estimate.

さらにこのアルゴリズムを強化します。枝をグループに分けますが、分け方に重複の無い組み合わせを採用します。つまり枝が 3 つの場合には、

第 1 のグループ	第 2 のグループ
枝 1	枝 2,3
枝 2	枝 1,3
枝 3	枝 1,2

という組み合わせを作り、各組み合わせについての結果を求め、その結果の平均を採用します。詳しくはつぎにプログラムコードを見てください。

1.3 プログラムコード

概要

Function BGmcamrcall(s, k, t, r, b, v As Double, m, branches, n As Long) As Double

For estimator = 1 To 2(推定量 1,2 の算出)

　　For simulation = 1 To n　(n 回の試行)

　　株価時系列の生成

　　Do While j > 0

```
                If j = m Then (満期m)
                    満期の評価
                ElseIf j < m Then  (満期前の分析：後退分析)
                    満期前の評価
                    If estimator = 1 Then (高バイアス推定量)
                        推定量 1 の算出
                    ElseIf estimator = 2 Then (低バイアス推定量)
                        推定量 2 の算出
                    End If
                    If w(j) < branches Then
                        各ノードの満期までの株価の枝の生成
                    End If
                End If
            Loop
            estimatorsum = estimatorsum +  推定値(1 or 2)
        Next simulation
        estimators(estimator) = estimatorsum / n
    Next estimator
    BGmcamrcall = 0.5 * Application.Max(Application.Max((s - k), 0), estimators(2)) + 0.5 * estimators(1)推定量の合成
End Function

Function BGmcamrcall(s, k, t, r, b, v As Double, m, branches, n As Long) As Double
    初期設定
    Dim drift, vsqrdt, discdt As Double
    Dim i, j, i1, i2 As Long
    Dim estimator, simulation As Long
    Dim estimatorsum, sum1, sum2 As Double
    Dim w() As Long
    Dim ss() As Double
    Dim estimators() As Double
    ReDim w(1 To m) As Long
    ReDim ss(1 To branches, 1 To m) As Double
    ReDim estimators(1 To 2) As Double
    drift = (b - v ^ 2 / 2) * t / (m - 1)
    vsqrdt = v * Sqr(t / (m - 1))
    discdt = Exp(-r * t / (m - 1))
```

```
For estimator = 1 To 2  (各推定量の算出)
        estimatorsum = 0
        For simulation = 1 To n  ( n 試行)
            ss(1, 1) = s
            w(1) = 1
            For j = 2 To m  (初期株価時系列の生成)
                ss(1, j) = ss(1, j - 1) * Exp(drift + vsqrdt * Application.NormSInv(Rnd))
                w(j) = 1
            Next j
            j = m
            Do While j > 0
                If j = m Then  (満期の分析)
                    ss(w(j), j) = Application.Max((ss(w(j), j) - k), 0)  (満期の評価)
                    If w(j) < branches Then  (枝の生成)
                        ss(w(j) + 1, j) = ss(w(j - 1), j - 1) * Exp(drift + vsqrdt * Application.NormSInv(Rnd))
                        w(j) = w(j) + 1
                    ElseIf w(j) = branches Then
                        w(j) = 0
                        j = j - 1
                    End If
                ElseIf j < m Then  (満期前の分析：後退分析)
                    If estimator = 1 Then(推定量 1 の算出)
                        sum1 = 0
                        For i1 = 1 To branches
                            sum1 = sum1 + discdt * ss(i1, j + 1)
                        Next i1
                        ss(w(j), j) = Application.Max(Application.Max((ss(w(j), j) - k), 0), sum1 / branches)
                    ElseIf estimator = 2 Then(推定量 2 の算出)
                        sum1 = 0
                        For i1 = 1 To branches(枝の分析)
                            sum2 = 0
                            For i2 = 1 To branches(枝グループの組み合わせ分析)
                                If i2 <> i1 Then sum2 = sum2 + discdt * ss(i2, j + 1)
                            Next i2
                            If Application.Max((ss(w(j), j) - k), 0) >= sum2 / (branches - 1) Then
                                sum1 = sum1 + Application.Max((ss(w(j), j) - k), 0)
                            Else
                                sum1 = sum1 + discdt * ss(i1, j + 1)
                            End If
                        Next i1
                        ss(w(j), j) = sum1 / branches
                    End If
                    If w(j) < branches Then  (枝の生成)
                        If j > 1 Then
                            ss(w(j) + 1, j) = ss(w(j - 1), j - 1) * Exp(drift + vsqrdt * Application.NormSInv(Rnd))
                            w(j) = w(j) + 1
                            For i = j + 1 To m  (満期までの枝の生成)
                                ss(1, i) = ss(w(j), j) * Exp(drift + vsqrdt * Application.NormSInv(Rnd))
                                w(i) = 1
                            Next i
                            j = m
                        Else
                            j = 0
                        End If
                    ElseIf w(j) = branches Then
                        w(j) = 0
                        j = j - 1
```

```
                End If
            End If
        Loop
        estimatorsum = estimatorsum + ss(1, 1)
    Next simulation
    estimators(estimator) = estimatorsum / n
Next estimator
BGmcamrcall = 0.5 * Application.Max(Application.Max((s - k), 0), estimators(2)) + 0.5 * estimators(1)
End Function
```

	A	B	C
1	s	100	
2	k	100	
3	t	1	
4	r	0	
5	b	0	
6	v	0.2	
7	m	5	max<5
8	branches	10	max<20
9	n(試行回数)	100	max<100
10	Bgmcamrcall	8.026405	
11			

2.最小二乗モンテカルロ法

モンテカルロ法を用いたアメリカンオプションの価格推定の強力な方法を紹介します。行使可能時刻で継続を選択したオプション保有者に最小二乗法による条件付期待収益の推定方法を提供します。

まず、どの行使可能時刻においても、アメリカンオプションの保有者は早期行使の収益と継続による期待収益を比較します。そして、早期行使の価値の方が高ければ、行使します。したがって、オプション保有を継続するさいの条件付期待収益をもとに最適化行使戦略は決定されます。簡単に表現するとｎ回の試行で得られる横断的な情報と最小二乗法を用いて、オプション保有者の条件付期待値を推定します。継続による事後の実現収益を状態変数の価値の関数として回帰します。回帰から求められた適合値は条件付期待関数の直接の推定量です。各試行における各行使可能時刻で、この条件付期待値を推定することにより、最適行使戦略の詳細を完全に得ることができます。この詳細はモンテカルロ法によるアメリカンオプションの価値の推定を正確なものにします。この方法を最小二乗モンテカルロ法と呼びます。この方法は２つのバイアスをともなう推定量を用いる方法とは、全く異なります。

2.1　プットオプションを用いた簡単な例

どの行使可能時刻においても、アメリカンオプションの保有者は早期行使の収益と継続による期待収益を比較し、早期行使の判断をします。オプション保有を継続するさいの条件付期待収益をもとに行使するかどうかは決定されます。n (=8)回の試行により生成された株価時系列のもつ横断的な情報を最小二乗法により最適化し、条件付期待関数を特定します。この回帰により求められた適合値は条件付期待値の効率的なバイアスの無い推定値で、オプションの最適停止時刻を正確に推定します。この構造をプットオプションを用いて説明していきましょう。

行使価格= 1.1

満期までの時刻=1,2,3

無リスク金利＝6%

試行の数＝ 8

各試行の株価生成メカニズム=リスク中立測度

Stock price paths

Path	$t=0$	$t=1$	$t=2$	$t=3$
1	1.00	1.09	1.08	1.34
2	1.00	1.16	1.26	1.54
3	1.00	1.22	1.07	1.03
4	1.00	.93	.97	.92
5	1.00	1.11	1.56	1.52
6	1.00	.76	.77	.90
7	1.00	.92	.84	1.01
8	1.00	.88	1.22	1.34

目的は各試行において各行使可能時刻についてのオプションの価値を最大にする停止規則を探すことです。満期(t= 3)まで行使されないという条件のもとで、いくつかの分析を行います。満期まで行使されなければ、オプション保有者に実現される損益は満期における最適戦略によりもたらされます。それはつぎのようなものです。

Cash-flow matrix at time 3

Path	$t = 1$	$t = 2$	$t = 3$
1	—	—	.00
2	—	—	.00
3	—	—	.07
4	—	—	.18
5	—	—	.00
6	—	—	.20
7	—	—	.09
8	—	—	.00

この損益はヨーロピアンオプションのものと同じです。

つぎに時刻２を見てみます。プットがインザマネーにあればオプションの保有者は早期行使をするか、満期まで保有を継続するかを決めなければなりません。インザマネーにある時系列について、

X を株価、

Y を時刻２で継続と判断したときの時刻３の調整後損益

とします。時刻２でインザマネーにある時系列のみを使うのは、この領域における条件付期待関数の推定をより良くする可能性があるからです。時刻２における結果はつぎのとおりです。

Regression at time 2

Path	Y	X
1	.00 × .94176	1.08
2	—	—
3	.07 × .94176	1.07
4	.18 × .94176	.97
5	—	—
6	.20 × .94176	.77
7	.09 × .94176	.84
8	—	—

時刻２の株価をもとにオプション保有による期待収益を推定

します。最小二乗法を用います。

$$\hat{Y} = \hat{a}_1 + \hat{a}_2 X + \hat{a}_3 X^2$$

結果は

$$E[Y|X] = -1.070 + 2.983X - 1.813X^2$$

です。この情報をもとに行使と継続の損益を比較します。

Optimal early exercise decision at time 2

Path	Exercise	Continuation
1	.02	.0369
2	—	—
3	.03	.0461
4	.13	.1176
5	—	—
6	.33	.1520
7	.26	.1565
8	—	—

早期行使の価値はインザマネーにあれば、$1.10 - X$となります。継続の場合は条件付期待関数のXに株価を代入すれば推定値が得られます。早期行使または継続の判断をした後の損益はつぎのとおりです。

Cash-flow matrix at time 2

Path	$t = 1$	$t = 2$	$t = 3$
1	—	.00	.00
2	—	.00	.00
3	—	.00	.07
4	—	.13	.00
5	—	.00	.00
6	—	.33	.00
7	—	.26	.00
8	—	.00	.00

時刻２で早期行使を行えば、時刻３での損益は０になります。この分析を時刻１についても行います。時刻１では５つの時系列(1,4,6,7,8)がインザマネーにあります。時刻２の分析を繰り返すのですが、Yは時刻２の実現損益の割引価値です。期待収益を用いると高いほうのバイアスを生みます。また、行使は各試行で一回しかできません。行使した際の実現損益は時刻１まで割り引かれる必要があります。

Regression at time 1

Path	Y	X
1	$.00 \times .94176$	1.09
2	—	—
3	—	—
4	$.13 \times .94176$	.93
5	—	—
6	$.33 \times .94176$	.76
7	$.26 \times .94176$	.92
8	$.00 \times .94176$	.88

時刻１の株価をもとにオプションを保有し続けることから得られる期待収益を推定するために、

$$\hat{Y} = \hat{a}_1 + \hat{a}_2 X + \hat{a}_3 X^2$$

を再度用います。結果は

$$E[Y|X] = 2.038 - 3.335X + 1.356X^2$$

です。Yの値は時刻２と３から持ってきます。この情報をもとに行使と継続の損益を比較します。

Optimal early exercise decision at time 1

Path	Exercise	Continuation
1	.01	.0139
2	—	—
3	—	—
4	.17	.1092
5	—	—
6	.34	.2866
7	.18	.1175
8	.22	.1533

停止規則はつぎのように決定されます。クロス表の 1 が早期行使の時刻です。

Stopping rule

Path	$t=1$	$t=2$	$t=3$
1	0	0	0
2	0	0	0
3	0	0	1
4	1	0	0
5	0	0	0
6	1	0	0
7	1	0	0
8	1	0	0

停止規則により実現さる損益はつぎの表です。

Option cash flow matrix

Path	$t=1$	$t=2$	$t=3$
1	.00	.00	.00
2	.00	.00	.00
3	.00	.00	.07
4	.17	.00	.00
5	.00	.00	.00
6	.34	.00	.00
7	.18	.00	.00
8	.22	.00	.00

これらの実現損益を時刻 0 に合わせて割り引くことでアメリカンプットオプションの価値を得ることができます。

t=1: (0.17+0.34+0.18+0.22)×0.94=0.855

t=2: $0\times0.94^2=0$

t=3: $0.07\times0.94^3=0.058$

したがって、その値は(0.855+0.058)/8=0.1142 です。

2.2 プログラムコード

```
Sub test()
    Dim s, k, t, r, b, v As Double
    Dim m, n As Long
    s = Application.Cells(2, 2)
    k = Application.Cells(3, 2)
```

```
t = Application.Cells(4, 2)
r = Application.Cells(5, 2)
b = Application.Cells(6, 2)
m = Application.Cells(7, 2)
v = Application.Cells(8, 2)
n = Application.Cells(9, 2)
Dim drift, vsqrdt, discdt, s0, ex, a1, a2, a0, val0 As Double
Dim sumval As Double
Dim i, j, jj, jjj, i1, i2, ii, tt, ttt As Long
Dim simulation, tstep As Long
Dim ss() As Double
ReDim ss(1 To n, 0 To m) As Double
Dim cfv() As Double
ReDim cfv(1 To n, 1 To m) As Double

drift = (b - v ^ 2 / 2) * t / (m - 1)
vsqrdt = v * Sqr(t / (m - 1))
discdt = Exp(-r * t / (m - 1))
For simulation = 1 To n
    ss(simulation, 0) = s
    For j = 1 To m
        ss(simulation, j) = ss(simulation, j - 1) * Exp(drift + vsqrdt * Application.NormSInv(Rnd))
        cfv(simulation, j) = 0
    Next j
Next simulation
For j = 1 To n '満期の最適化
    s0 = ss(j, m)
    cfv(j, m) = Application.Max(s0 - k, 0)
    'Application.Cells(j + 10, 15) = cfv(j, m)
Next j
jjj = 1
For tstep = m - 1 To 1 Step -1
    Dim Y() As Double
    Dim X() As Double
    jj = 1
    For j = 1 To n
        s0 = ss(j, tstep)
        ex = s0 - k
        If ex > 0 Then
            ReDim Y(jj, 2) As Double
            ReDim X(jj, 2) As Double
            val0 = cfv(j, tstep + 1)
            ttt = 1
```

```
                If val0 = 0 Then
                    ttt = 2
                    For tt = tstep + 2 To m
                        val0 = cfv(j, tt)
                        If val0 > 0 Then
                            Exit For
                        End If
                        ttt = ttt + 1
                    Next tt
                End If
                Y(jj, 1) = discdt ^ ttt * val0
                X(jj, 1) = s0
                X(jj, 2) = s0 ^ 2
                jj = jj + 1
            End If

        Next j
        a1 = Application.WorksheetFunction.Index(Application.WorksheetFunction.LinEst(Y, X), 1, 1)
        a2 = Application.WorksheetFunction.Index(Application.WorksheetFunction.LinEst(Y, X), 1, 2)
        a0 = Application.WorksheetFunction.Index(Application.WorksheetFunction.LinEst(Y, X), 1, 2, 1)
        For j = 1 To n
            s0 = ss(j, tstep)
            ex = s0 - k
            If ex > 0 Then
                val0 = a1 * s0 + a2 * s0 ^ 2 + a0
                If ex > val0 Then
                    cfv(j, tstep) = ex
                    cfv(j, tstep + 1) = 0
                End If
            End If
        Next j
        'Application.Cells(jjj, 10) = tstep
        jjj = jjj + 1
    Next tstep
    sumval = 0
    For j = 1 To n
        For i = 1 To m
            val0 = cfv(j, i)
            If val0 > 0 Then
                For ii = i + 1 To m
                    cfv(j, ii) = 0
                Next ii
            End If
```

```
            sumval = sumval + val0 * discdt ^ i
            'Application.Cells(j + 10, i + 10) = val0
        Next i
    Next j
    Application.Cells(12, 2) = sumval / n
End Sub
```