Credit Suisse Case Study

CET-1 ratio parameter estimation

```
DataB = Import["Wolfram Mathematica/Data/Credit_Suisse_Data_17-23.xlsx",
        导入                                                    偏导
   {"Data", 2, Table[i, {i, 2, 26}], 4}];
                表格

DataJ = Differences[Tan[Pi * ((1 - 2 * DataB) / 2)] + Cot[Pi * (1 - DataB[[1]])]];
        差分        正切 圆周率                          余切 圆周率
(* solvency shocks *)
UnitRootTest[DataJ] (* stationarity check *)
单位根检验
% < 0.05
```

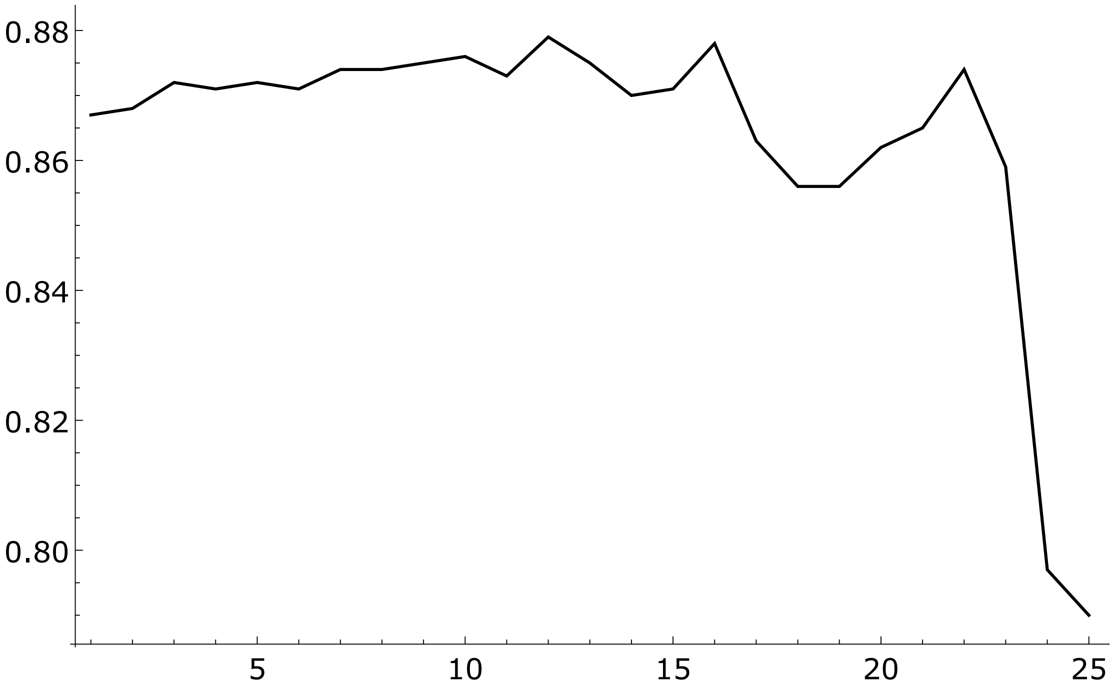
```
0.00073546
```

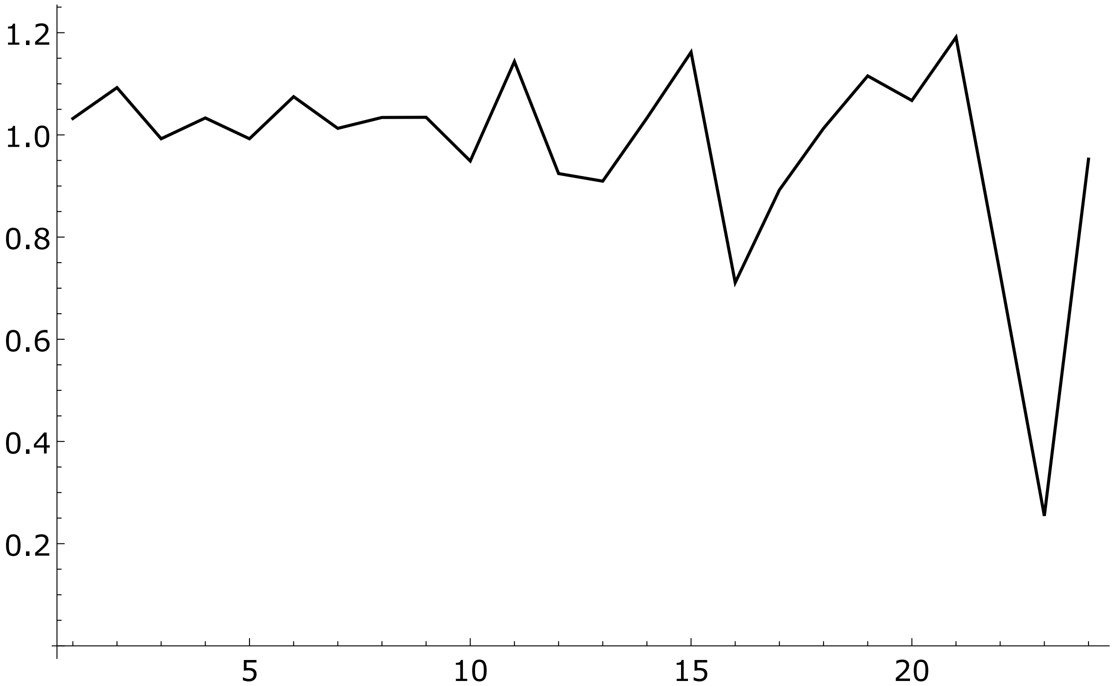
```
True
```

```
DataH = DataJ - Quantile[SmoothKernelDistribution[DataJ], 0];
                分位数   平滑核分布
(* positive transformation *)
nH = Length[DataH]
     长度
```

```
24
```

```
dp = ListLinePlot[1 - DataB, PlotStyle -> Black, AxesStyle -> Black,
       绘制点集的线条                  绘图样式         黑色      坐标轴样式        黑色
  TicksStyle -> Directive["Label", 15], ImageSize -> Large,
  刻度样式         指令         标签            图像尺寸          大
 PlotRange -> Full]
 绘制范围       全范围
dp1 = ListLinePlot[DataH, PlotStyle -> Black, AxesStyle -> Black,
        绘制点集的线条                绘图样式         黑色      坐标轴样式        黑色
  TicksStyle -> Directive["Label", 15], ImageSize -> Large, PlotRange -> Full]
                 指令         标签            图像尺寸          大       绘制范围         全范围
```



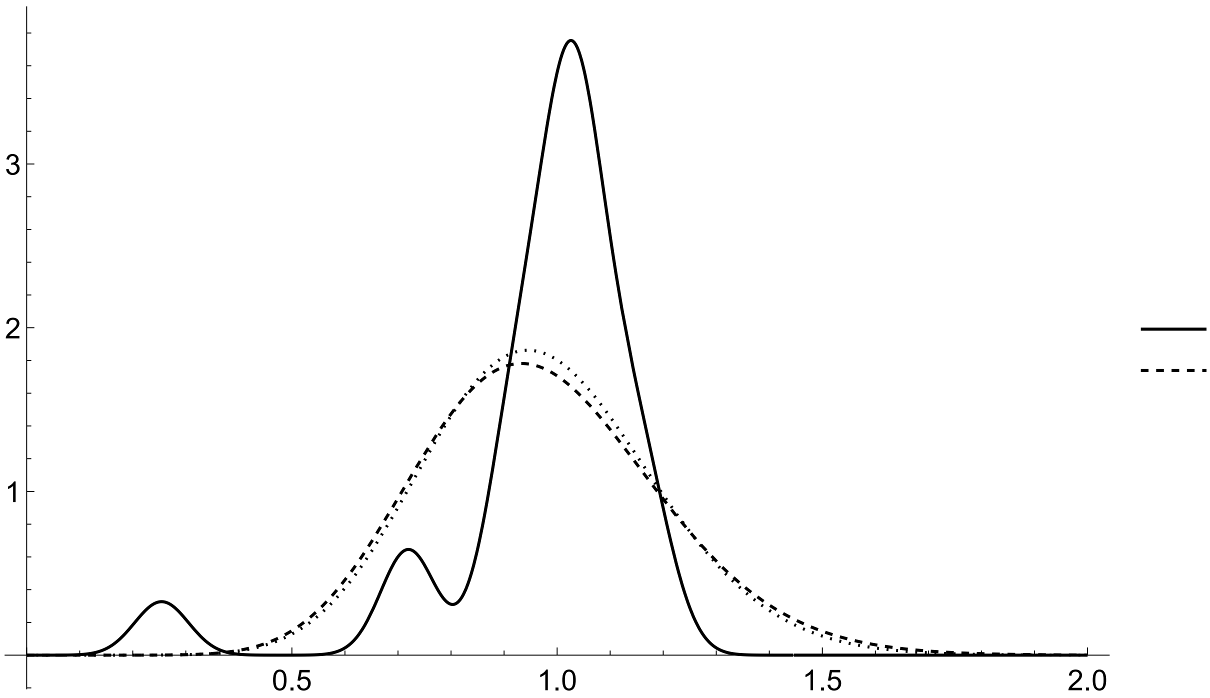


```
(* Kernel density estimation *)
kernelH = SmoothKernelDistribution[DataH];
            平滑核分布
fH[x_] := PDF[kernelH, x]; (* kernel density *)
            概率密度函数

(* Model PDFs continuous part *)
f[λ_, α_, β_, t_, x_] :=
  Exp[(-λ) * t] * (λ * β^α * t * x^(α - 1) * (Exp[(-β) * x] / (α - 1)!) *
  指数形式                                    指数形式
     HypergeometricPFQ[{}, Table[1 + i / α, {i, 1, α}],
     广义超几何函数                表格
   λ * t * (β * (x / α))^α]); (* general PDF *)
                                     概率密度函数
f0[λ_, β_, t_, x_] := Exp[(-λ) * t] * (Exp[(-β) * x] * Sqrt[λ * β * (t / x)] *
                      指数形式          指数形式           平方根
     BesselI[1, 2 * Sqrt[λ * β * t * x]]); (* relaxed PDF *)
                    平方根                        概率密度函数


T = 1/4; (* quarterly frequency *)
relaxedest =
  Maximize[{Sum[Log[f0[λ, β, T, DataH[[i]]]], {i, 1, nH}], λ > 0, β > 0},
            求和 对数
   {λ, β}];  (* MLE (relaxed) *)
λhat0 = λ /. relaxedest[[2]];
βhat0 = β /. relaxedest[[2]];
Print[{λhat0, βhat0}]
打印

{147.835, 37.9971}

λhat0 > 15 (* if false use a different method for reestimation *)

True

genest = For[α = 1, α <= 5, α++,
         For循环
  {est[α] = Maximize[{Sum[Log[f[λ, α, β, T, DataH[[i]]]], {i, 1, nH}],
            最大点值    求和 对数
      λ > 0, β > 0}, {λ, β}],
  Print[est[α]]}] (* MLE *)
  打印

{1.94434, {λ → 147.835, β → 37.9971}}

{2.51877, {λ → 116.83, β → 60.0564}}

{2.79354, {λ → 106.477, β → 82.1014}}

{2.95453, {λ → 101.293, β → 104.139}}

{3.06027, {λ → 98.1804, β → 126.174}}
```

```
αhat =
  Sum[If[est[α][[1]] == Max[Table[est[a][[1]], {a, 1, 5}]], α, 0], {α, 1, 5}];
  求和 如果                  …   表格
λhat = Sum[If[est[α][[1]] == Max[Table[est[a][[1]], {a, 1, 5}]],
       求和 如果                   …   表格
    λ /. est[α][[2]], 0], {α, 1, 5}];
βhat = Sum[If[est[α][[1]] == Max[Table[est[a][[1]], {a, 1, 5}]],
       求和 如果                   …   表格
    β /. est[α][[2]], 0], {α, 1, 5}];
Print[{λhat, αhat, βhat}]
打印
```

```
{98.1804, 5, 126.174}
```

```
p1 = Plot[{fH[x], f0[λhat0, βhat0, T, x], f[λhat, αhat, βhat, T, x]},
     绘图
  {x, 0, 2}, PlotStyle → {Black, Dashed, Dotted},
             绘图样式      黑色    虚线     点线
  PlotTheme → "Monochrome", TicksStyle → Directive["Label", 15],
  绘图主题                   刻度样式      指令        标签
  PlotLegends → {"Fitted", "Fitted (α = 1)"}, ImageSize → Large, PlotRange → Full]
  绘图的图例                                   图像尺寸     大       绘制范围     全范围
```



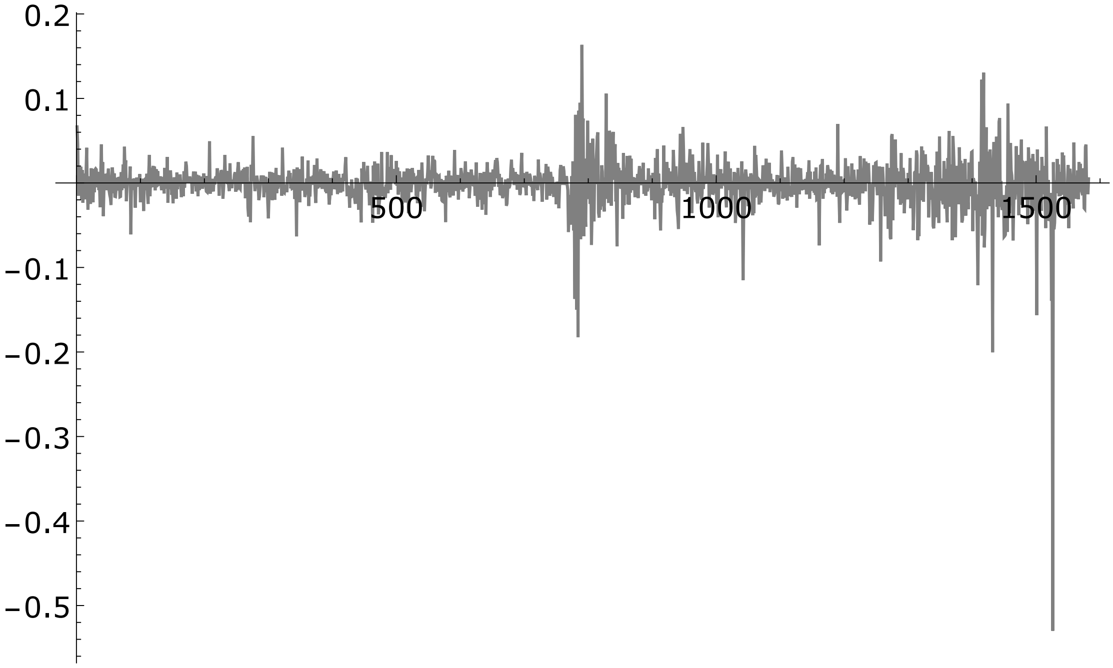
Stock price parameter estimation

```
Rets = Import["Wolfram Mathematica/Data/Credit_Suisse_Data_17-23.xlsx",
       导入                                              偏导
   {"Data", 1, Table[i, {i, 2, 1583}], 7}];
               表格
```

```
nR = Length[Rets]
      长度
```

```
1582
```

```
dp2 = ListLinePlot[Rets, PlotStyle -> Gray, AxesStyle -> Black,
      绘制点集的线条          绘图样式      灰色     坐标轴样式     黑色
  TicksStyle -> Directive["Label", 15], ImageSize -> Large, PlotRange -> Full]
  刻度样式        指令        标签         图像尺寸       大      绘制范围       全范围
```




```
UnitRootTest[Rets] (* stationarity check *)
单位根检验
```

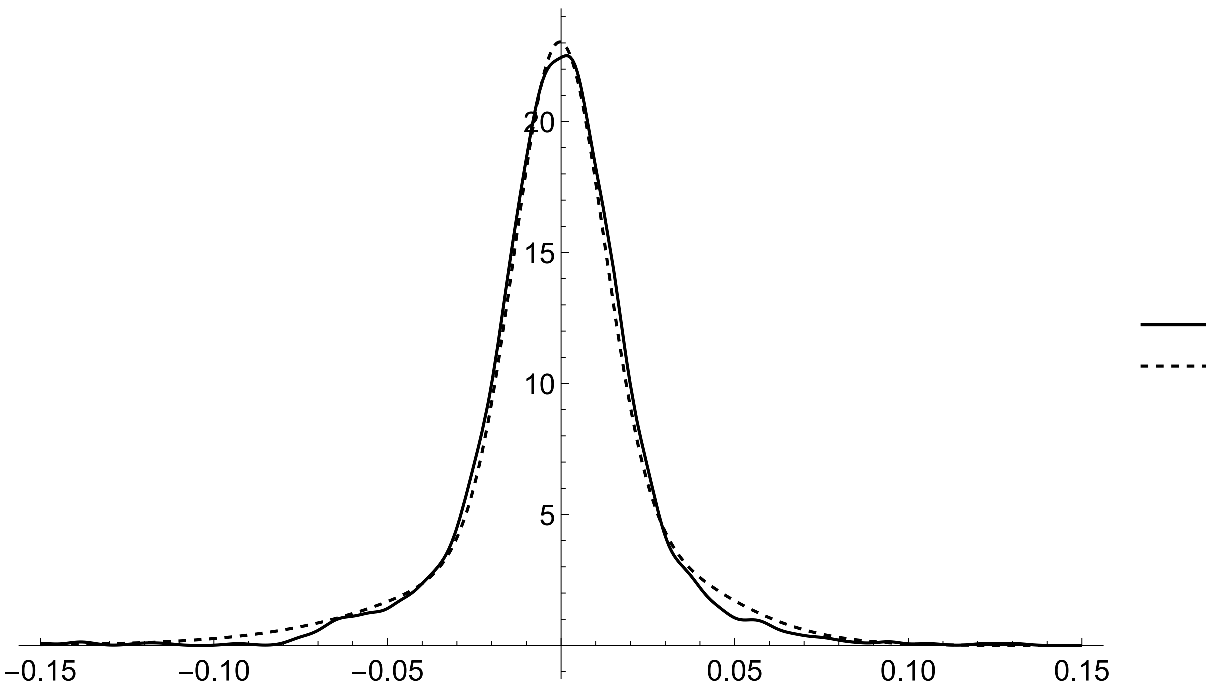
```
% < 0.05
```

7.9505×10^{-19}

```
True
```

```
(* Kernel density estimation *)
kernelX = SmoothKernelDistribution[Rets];
          平滑核分布
fRets[x_] := PDF[kernelX, x]; (* kernel density *)
             概率密度函数
(* Model PDFs (at least daily freq.) *)
fX[λ1_, α_, β_, μ_, σ_, λ2_, μV_, σV_, η_, Δ_, x_] :=
  Module[{ν = μ + μV / Δ, υ = Sqrt[σ^2 + σV^2 / Δ]},
                              平方根
   Ψ = (Δ^((α - 1) / 2) / Sqrt[2 * Pi * υ^2]) * (β * (υ / η))^α * Exp[-(x - ν * Δ)^2 /
                         平方根    圆周率                             指数形式
        (2 * υ^2 * Δ) + (η * (x - ν * Δ) + β * υ^2 * Δ)^2 / ((2 * η * υ)^2 * Δ)] *
     ParabolicCylinderD[-α, (η * (x - ν * Δ) + β * υ^2 * Δ) / (η * υ * Sqrt[Δ])];
     抛物柱面函数                                                   平方根
   λ1 * λ2 * Δ^2 * Ψ + λ2 * Δ * ((1 - λ1 * Δ) / Sqrt[2 * Pi * υ^2 * Δ]) *
                                                平方根    圆周率
     Exp[-(x - ν * Δ)^2 / (2 * υ^2 * Δ)] + λ1 * Δ * (1 - λ2 * Δ) * Ψ +
     指数形式
    (1 - λ1 * Δ) * ((1 - λ2 * Δ) / Sqrt[2 * Pi * σ^2 * Δ]) *
                                    平方根    圆周率
     Exp[-(x - μ * Δ)^2 / (2 * σ^2 * Δ)]];
     指数形式


Δt = 1 / 252; (* daily frequency *)
λ1hat = λhat;
genretest =
 Maximize[{Sum[Log[fX[λ1hat, αhat, βhat, μ, σ, λ2, μV, σV, η, Δt, Rets[[i]]]],
 最大点值      求和 对数
    {i, 1, nR}], -1 < μ < 1, 0 < σ < 1,
   λ2 > 0, -1 < μV < 1, 0 < σV < 1, 0 < η}, {μ, σ, λ2, μV, σV, η}]
(* constrained MLE *)


{3744.67, {μ → -0.123329, σ → 0.202993,
  λ2 → 2.31387, μV → 0.067659, σV → 0.0179964, η → 1.79029}}

μhat = μ /. genretest[[2]];
σhat = σ /. genretest[[2]];
λ2hat = λ2 /. genretest[[2]];
μVhat = μV /. genretest[[2]];
σVhat = σV /. genretest[[2]];
ηhat = η /. genretest[[2]];
Print[{μhat, σhat, λ2hat, μVhat, σVhat, ηhat}]
打印

{-0.123329, 0.202993, 2.31387, 0.067659, 0.0179964, 1.79029}
```

```
p2 = Plot[{fRets[x], fX[λ1hat, αhat, βhat, μhat,
      绘图
    σhat, λ2hat, μVhat, σVhat, ηhat, Δt, x]}, {x, -0.15, 0.15},
  PlotStyle -> {Black, Dashed}, PlotTheme -> "Monochrome",
  绘图样式          黑色     虚线        绘图主题
   TicksStyle -> Directive["Label", 15],
   刻度样式          指令          标签
  PlotLegends -> {"Kernel", "Fitted"}, ImageSize -> Large, PlotRange -> Full]
  绘图的图例                               图像尺寸     大      绘制范围     全范围
  (* density comparison *)
```



CoCo valuation

```
(* Distributional formulas *)
(* whole PDF of Erlang-CPP *)
          概率密度函数
fh[λ_, α_, β_, t_, x_] :=
  Exp[(-λ) * t] * (DiracDelta[x] + λ * β^α * t * x^(α - 1) * (Exp[(-β) * x] / (α - 1) !) *
  指数形式          狄拉克δ函数                                   指数形式
   HypergeometricPFQ[{}, Table[1 + i / α, {i, 1, α}], λ * t * (β * (x / α)) ^α]);
   广义超几何函数            表格
(* continuous part of PDF *)
                     概率密度函数
fc[λ_, α_, β_, t_, x_] := λ * β^α * t * x^(α - 1) * (Exp[(-λ) * t - β * x] / (α - 1) !) *
                                                     指数形式
    HypergeometricPFQ[{}, Table[1 + i / α, {i, 1, α}],
                          表格
  λ * t * (β * (x / α)) ^α];
(* time derivative of continuous part *)
tdfc[λ_, α_, β_, t_, x_] :=
  λ * β^α * x^(α - 1) * (Exp[(-λ) * t - β * x] / (α - 1) !) *
                         指数形式
   (λ * t * ((β * x) ^α / Pochhammer[1 + α, α]) *
                           波赫汉默
       HypergeometricPFQ[{}, Table[2 + i / α, {i, 1, α}], λ * t * (β * (x / α)) ^α] -
                              表格
   (λ * t - 1) * HypergeometricPFQ[{}, Table[1 + i / α, {i, 1, α}],
                 广义超几何函数             表格
         λ * t * (β * (x / α)) ^α]);
(* expectation of negative part of CCPP *)
I1[λ_, α_, β_, (t_) ?NumericQ] := I1[λ, α, β, t] =
                       数值量判定
    NIntegrate[(1 - β * (y / (λ * α * t))) * fc[λ, α, β, t, y], {y, 0, λ * α * (t / β)},
    数值积分
  AccuracyGoal -> 3];
  准确度目标
(* a rewarding probability *)
I2[λ_, α_, β_, (t_) ?NumericQ, (x_) ?NumericQ] := I2[λ, α, β, t, x] =
                       数值量判定       数值量判定
    NIntegrate[tdfc[λ, α, β, t, y], {y, 0, x}, AccuracyGoal -> 3];
    数值积分                                      准确度目标
(* supremum probability of CCPP *)
SupPr[λ_, α_, β_, t_, x_] :=
   (1 - Exp[(-λ) * t] - NIntegrate[fc[λ, α, β, t, y], {y, 0, x + λ * α * (t / β)}]) +
        指数形式          数值积分
  NIntegrate[I1[λ, α, β, t - s] *
  数值积分
      ((β / (λ * α)) * (I2[λ, α, β, s, x + λ * α * (s / β)] - λ * Exp[(-λ) * s]) +
                                                                指数形式
    fc[λ, α, β, s, x + λ * α * (s / β)]), {s, 0, t}, AccuracyGoal -> 3];
                                                     准确度目标
```

```
(* Simplified distributional formulas: α==1 *)
fh0[λ_, β_, t_, x_] := Exp[(-λ) * t] * (DiracDelta[x] +
                       指数形式          狄拉克δ函数
    Exp[(-β) * x] * Sqrt[λ * β * (t / x)] * BesselI[1, 2 * Sqrt[λ * β * t * x]]);
                    平方根                   第一类修正贝塞…     平方根
fc0[λ_, β_, t_, x_] := Exp[(-λ) * t - β * x] * Sqrt[λ * β * (t / x)] *
                       指数形式                 平方根
  BesselI[1, 2 * Sqrt[λ * β * t * x]];
  第一类修正贝塞…  平方根
tdfc0[λ_, β_, t_, x_] := λ * β * Exp[(-λ) * t - β * x] *
                                 指数形式
   (HypergeometricOF1Regularized[1, λ * β * t * x] -
    正则化的合流超几何函数0F1
  λ * t * HypergeometricOF1Regularized[2, λ * β * t * x]);
          正则化的合流超几何函数0F1
I10[λ_, β_, (t_)?NumericQ] :=
                 数值量判定
  I1[λ, β, t] = NIntegrate[(1 - β * (x / (λ * t))) * fc0[λ, β, t, x],
                数值积分
    {x, 0, λ * (t / β)}, AccuracyGoal -> 3];
                         准确度目标
I20[λ_, β_, t_, (x_)?NumericQ] := I2[λ, β, t, x] =
                     数值量判定
  NIntegrate[tdfc0[λ, β, t, y], {y, 0, x}, AccuracyGoal -> 3];
  数值积分                                  准确度目标
SupPr0[λ_, β_, t_, x_] :=
  (1 - Exp[(-λ) * t] - NIntegrate[fc0[λ, β, t, y], {y, 0, x + λ * (t / β)}]) +
       指数形式         数值积分
 NIntegrate[I10[λ, β, t - s] *
 数值积分
    ((β / λ) * (I20[λ, β, s, x + λ * (s / β)] - λ * Exp[(-λ) * s]) +
                                                   指数形式
      fc0[λ, β, s, x + λ * (s / β)]), {s, 0, t},
  AccuracyGoal ->
  准确度目标
    3];
```

Market parameters (with artificial values)

```
W = 0.07; (* watermark *)
C0 = 0.11; (* current CET1 ratio *)
S0 = 15; (* current stock price *)
K = 10; (* principal *)
c = 6.75/2; (* semiannual coupons *)
w = 0.75; (* write-down porportion *)
r = 0.0168; (* risk-free rate *)
q = 0.02; (* dividend yield *)
T = 5; (* maturity *)
M = 2 * T; (* coupon number *)
m = 0.03; (* net profit *)
l = 0.04; (* loan growth s.t. CET1 ratio equals watermark at infinity *)
n0 = 0.8; (* current one minus CET-1 ratio *)

(* converted trigger barrier *)
Jbar = Tan[Pi * ((1 - 2 * W) / 2)] + Cot[Pi * (1 - DataB[[1]])]
        正切 圆周率                     余切 圆周率
Jbar0 = Jbar;
```

```
2.22136
```

```
(* Govt. intervention-related functionals *)
```

```
E1[κ1_, ς1_, t_, u_] := Exp[κ1^2 * (t/ (2 * ς1^2)) *
                        指数形式
    (Tanh[Sqrt[2 * ς1^2 * t^2 * u]] / Sqrt[2 * ς1^2 * t^2 * u] - 1)] *
     双…  平方根                        平方根
   Sqrt[Sech[Sqrt[2 * ς1^2 * t^2]]];
   平方根 双…  平方根
(* continuous part expectation *)

ac[μV_, σV_, i_] :=
  Which[i == 1, 1 - (1/2) * Erfc[(μV + σV) / Sqrt[2 * σV^2]], i == 2,
  Which循环                      补余误差函数        平方根
  (1/2) * (Erfc[(μV + σV) / Sqrt[2 * σV^2]] - Erfc[(μV + 2 * σV) / Sqrt[2 * σV^2]]),
           补余误差函数      平方根              补余误差函数               平方根
   i == 3,
  (1/2) * (Erfc[(μV + 2 * σV) / Sqrt[2 * σV^2]] -
           补余误差函数          平方根
      Erfc[(μV + 3 * σV) / Sqrt[2 * σV^2]]), i == 4,
      补余误差函数          平方根
   (1/2) * Erfc[(μV + 3 * σV) / Sqrt[2 * σV^2]], True, None];
           补余误差函数          平方根             真     无
E2[κ2_, ς2_, λ2_, λ30_, μV_, σV_, t_, u_] :=
  Exp[(-u) * λ30 * ((1 - Exp[(-κ2) * T]) / κ2) +
  指数形式                指数形式
   (λ2 / κ2) * Sum[ac[μV, σV, i] * Exp[(-i) * ς2 * (u / κ2)] * (ExpIntegralEi[
               求和                 指数形式                      指数积分Ei
          i * u * (ς2 / κ2)] - ExpIntegralEi[i * u * ς2 * (Exp[(-κ2) * t] / κ2)]),
                               指数积分Ei                  指数形式
    {i, 1, 4}] - λ2 * t]; (* jump part expectation *)
```

Write-down CoCo

```
(* write-off *)
Timing[
计算时间
 WDZ = K * Exp[(-r) * T] * (1 - SupPr[λhat, Ceiling[αhat], βhat, T, Jbar]) +
           指数形式                       向上取整
  Sum[c * Exp[(-r) * k * (T / M)] *
  求和   指数形式
     (1 - SupPr[λhat, Ceiling[αhat], βhat, k * (T / M), Jbar]), {k, 1, M}]]
                      向上取整
```

```
{2.96875, 41.2376}
```

```
(* write-
 down (quadrature rule applied for fast approximation with NN steps ) *)
NN = 4 * T;
κ1hat = 0.09;
ς1hat = 1.4;
κ2hat = 3.2;
ς2hat = 0.9;
λ30hat = 0; (* artificial values *)
Timing[WD =
计算时间
  WDZ + (1 - w) * K * Sum[Exp[(-r) * (T / NN) * j] * E1[κ1hat, ς1hat, j * (T / NN), 1] *
                       求和 指数形式
      E2[κ2hat, ς2hat, λ2hat, λ30hat, μVhat, σVhat, j * (T / NN), 1] *
    (SupPr[λ1hat, Ceiling[αhat], βhat, (j + 1) * (T / NN), Jbar] -
                 向上取整
        SupPr[λ1hat, Ceiling[αhat], βhat, j * (T / NN), Jbar]), {j, 1, NN}]]
                     向上取整
```

```
{10.2188, 41.238}
```

Convertible AT1 bond pricing

```
(* quadrature rule applied for fast approximation with NN steps *)
ψ1[p_] := (1 + ηhat * (p / βhat)) ^ (-αhat) - 1;
ψ2[p_] := Exp[μVhat + σVhat^2 / 2] - 1; (* exponential correcting terms *)
          指数形式
Q[p_] := p * q + (1 - p) * r + (1 / 2) * p * (1 - p) * σhat^2 +
   λ1hat * (p * ψ1[1] - ψ1[p]) + λ2hat (p * ψ2[1] - ψ2[p]);
(* power-adjusted dividend *)
CV[p_] :=
  WDZ + w * K * Sum[Exp[(-Q[p]) * (T / NN) * j] * E1[κ1hat + p * σhat * ς1hat, ς1hat,
                求和 指数形式
      j * (T / NN), 1] * E2[κ2hat, ς2hat, λ2hat * (1 + ψ2[p]),
   μVhat + p * σVhat^2, σVhat, j * (T / NN), 1] * (SupPr[λhat * (1 + ψ1[p]),
        Ceiling[αhat], βhat + ηhat * p, (j + 1) * (T / NN), Jbar] -
        向上取整
   SupPr[λhat * (1 + ψ1[p]), Ceiling[αhat], βhat + ηhat * p, j * (T / NN), Jbar]),
                          向上取整
     {j, 1, NN}];
```

```
DiscretePlot[CV[p], {p, 0, 1, 0.1}, Joined -> True,
离散图                                连接点     真
 Filling -> None, ImageSize -> Large, PlotRange -> All]
 填补         无       图像尺寸        大      绘制范围        全部
```