```
--> /* 9 偏微分方程式によるデリバティブプライシング */

--> /* ［数値処理］→［自動的に数値で出力］*/
    if numer#false then numer:false else numer:true;
    /*浮動小数点の桁数=7に指定 */
    fpprec : 7;
    /* 浮動小数点表示桁数を7に指定*/
    fpprintprec:7;

--> /* 9.1 陽解法*/
```

```
--> PDEex_call (S,K,r,sigma,T,M,N) :=
    (
    /* 2*N=原資産価格分割数
         M=時間分割数  */
    /* 注: N>Mとする*/
    Smax : 20*S
    /* 解くべき領域を原資産価格の 20 倍に設定 */,
    xmax : log (Smax)
    /*S_max に対応する x_max を計算*/,
    dt : T/M
    /* 時間 [0,T] を M 個に分割 */,
    dx : (xmax-log (S))/N
    /* xmax から x=log (S) までを N 個に分割 */,
    xmin : log (S) -N*dx
    /* xmin か ら x=log (S) ま で を N 個 に 分 割 す る よ う に */
    /* xmin を設定 */,
    NN : 2*N,
    x : makelist(xmin+i*dx, i, 0, NN)
    /* 2N+1 次元ベクトル, x[1], x[2N+1] が境界, x[N+1]がlog(S) */,
    dd : dt/ (dx^2) /* delta を定義する */,
    a : (dd/2) * (sigma^2-dx* (r-sigma^2/2)),
    b : (1-dd*sigma^2),
    c : (dd/2) * (sigma^2+dx* (r-sigma^2/2)),
    /* 初期条件 */
    f : makelist(max (exp (x[i]) -K,0), i, 1,NN+1)
    /* t=T (満期)のペイオフを代入 */,
    g : f /*1 ステップ前の情報を記憶するために g を用意*/,
    /*再帰計算*/
    /* t->0 (j->M) に向かって解いていく*/
    for j:1 thru M do (
    f: makelist(a*g[i-1]+b*g[i]+c*g[i+1], i, 2, NN),
    g:f,
    NN:NN-2),
    /* リストfの長さは, 初項, 最終項の２項について, ２つ減少*/
    C0 : exp (-r*T) *f[N-M+1],
    /* 原資産価格 S に対応するデリバティブ価格 */
    /* BS 式計算 */
    load(distrib),
    d1 : (log (S/K) + (r+sigma^2/2) *T) / (sigma*sqrt (T)),
    d2 : d1 - sigma*sqrt (T),
    C0_A : S*cdf_normal (d1,0,1) - exp (-r*T) *K*cdf_normal (d2,0,1),
    print("PDE=", C0,", ", "BS=",C0_A)
    );

--> PDEex_call (100,110,0.01,0.2,1,50,50);

--> PDEex_call (100,110,0.01,0.2,1,100,100);
```

```
--> /* 9.2 陰解法 */
    PDEim_call (S,K,r,sigma,T,M,N):=
     (
    Smax : 20*S
    /* 解くべき領域を原資産価格の 20 倍に設定 */,
    xmax : log (Smax)
    /* Smax に対応する xmax を計算 */,
    dt : T/M
    /* 時間 [0,T] を M 個に分割 */,
    dx : (xmax-log (S))/N
    /* xmax から x=log (S) までを N 個に分割 */,
    xmin : log (S) -N*dx
    /* xmin か ら x=log (S) ま で を N 個 に 分 割 す る よ う に */
    /* xmin を設定 */,
    NN : 2*N,
    x : makelist(xmin+i*dx,i, 0, NN),
    dd : dt/ (dx^2) /* delta を定義する */,
    a : -dd* (sigma^2-dx* (r-sigma^2/2))/2,
    b : (1+dd*sigma^2),
    c : -dd* (sigma^2+dx* (r-sigma^2/2))/2,
    /* 関数  diagmatrix (n, x) */
    /* 対角要素すべてが xに等しくて、 サイズが n×nの対角線行列を返す */
    /* 関数 zeromatrix (m, n) */
    /* 要素すべてがゼロの m行 n列行列を返す*/
    /* 関数: addcol (M, list_1, ..., list_n) */
    /*1 つか複数のリスト(または行列)で与えられる列を行列 Mに追加する */
    /* 関数: addrow (M, list_1, ..., list_n) */
    /*1 つか複数のリスト(または行列)で与えられる行を行列 Mに追加する */
    Da : addcol (addrow (zeromatrix(1, NN-2),diagmatrix(NN-2, a)),zeromatrix(NN-1,1)),
    Db : diagmatrix (NN-1, b),
    Dc : addcol (zeromatrix(NN-1,1), addrow(diagmatrix(NN-2, c),zeromatrix(1, NN-2))),
    /*  行列 D を生成 */
    D : Da + Db + Dc,
    /* D の逆行列を計算 */
    /* 関数: invert (M) あるは  M^^-1 */
    /* 行列 Mの逆行列を返す */
    Dinv : invert(D),
    /* 初期条件 */
    f0 : makelist(max (exp (x[i]) -K,0), i, 1,NN+1)
    /* t=T (満期)のペイオフを代入 */,
    /* 関数: rest  */
    /*   rest (expr, n)  */
    /*    rest (expr) */
    /* nが正なら、 頭の n個の要素を取り除いた exprを返し、 */
    /* nが負なら、 お尻の n個の要素を取り除いた exprを返す。 */
    /* nが1なら、nを省略可。 */
    /* 1番目の引数 exprは、リスト、行列、他の式を取り得る。*/
    f : rest(rest(f0,-1))
    /*  境界 f0[1] (i=0),f0[NN+1] (i=2N) を除いた 2N-1 次元ベクトル */,
    /*境界条件 */
    g : makelist(0, NN-1)
    /* NN-1 個の要素0からなる数列を生成 */,
    g[1] : a*0
    /* i=0 境界条件を設定*/
    /* t->0 (j->M) に向かって解いていく */,
    for j :1 thru M do (
    g[NN-1] : c*exp (r*j*dt) * f0[NN+1]
    /* i=2N による境界条件を設定 */,
    f : Dinv. (f-g)
    ),
    C0 : exp (-r*T) *f[N][1],
    /* 原資産価格 S に対応するデリバティブ価格 */
    /* BS 式計算 */
    load(distrib),
    d1 : (log (S/K) + (r+sigma^2/2) *T) / (sigma*sqrt (T)),
    d2 : d1 - sigma*sqrt (T),
    C0_A : S*cdf_normal (d1,0,1) - exp (-r*T) *K*cdf_normal (d2,0,1),
    print("PDE=", C0,", ", "BS=",C0_A)
    );
```

```
--> PDEim_call (100,110,0.01,0.2,5,10,10);

--> PDEim_call (100,110,0.01,0.2,5,50,50);

--> PDEim_call (100,110,0.01,0.2,5,100,100);

--> PDEim_call (100,110,0.01,0.2,5,500,500);
```