```
--> /* Appendix Maxima の最適化関数 */

--> if numer#false then numer:false else numer:true;

--> fpprec:7;

--> fpprintprec:7;

--> /* A.1 多変量最適化問題 */

--> black_scholes(S,K,r,sigma,T):=
    (
    load(distrib),
    d1 : (log (S/K) + (r+sigma^2/2) *T) / (sigma*sqrt (T)),
    d2 : d1 - sigma*sqrt (T),
    C0 : S*cdf_normal (d1,0,1) - exp (-r*T) *K*pdf_normal (d2,0,1)
    );

--> err2(S,K,r,sigma,T,MktPrice):=
    (
    /* var (r,sigma) とする */
     /*MktPrice と K はベクトル*/
    MktPrice-black_scholes (S,K,r,sigma,T)
    );

--> err2s(S,K,r,sigma,T,MktPrice):=
    (
    /* var (r,sigma) とする */
     /*MktPrice と K はベクトル*/
    sum((MktPrice[i]-black_scholes (S,K[i],r,sigma,T))^2, i, 1, length(MktPrice))
    );

--> /*K と MktPrice にはベクトルを与える*/
    K_sample :[80,90,100,110,120];
    Mkt_sample :[22.75,15.1,8.43,4.72,3.28];

--> M: transpose(matrix(K_sample, Mkt_sample));

--> load (lsquares);

--> lsquares_estimates (M, [K,MktPrice],
    err2(100,K,r,sigma,1,MktPrice)=0, [r,sigma]);
```

```
--> load(minpack);/* 最小化と根のためのパッケージ*/


--> minpack_lsquares(
    err2(100, K_sample, r, sigma, 1, Mkt_sample), [r,sigma], [0.02, 0.2]);

--> err2(100, K_sample, r, sigma, 1, Mkt_sample);

--> load (lbfgs);

--> lbfgs(err2s(100,K_sample,r,sigma,1,Mkt_sample), [r,sigma], [0.01,0.1], 1e-7, [0,0]);

--> /* A.2
    最適化問題としての有効フロンティア */


--> /* 期待収益率を定義 */
    mu :[0.1,0.05,0.15];

--> /* 分散共分散行列を定義 */
    V : matrix([0.2^2,-0.015,0.01],[-0.015,0.1^2,-0.02],[0.01,-0.02,0.3^2]);
```

```
--> /* 不等式制約付き最小化パッケージfmin_coblya読み込み */
    load(fmin_cobyla);
    /* fmin_cobyla(
         F(x1,...,xn),
         [x1,...,xn],
         [a1,...,an],
         constraints=[g1(x1,...,xn),
                  ..., gm(x1,...,xn)])

    F(x1,...,xn)は最小化目的関数
    [a1,...,an]は変数(x1,...,xn)初期値
     g1(x1,...,xn), ..., gm(x1,...,xn)) は不等号,等号制約式   */
    /* 戻り値
    [[最小化変数リスト], 最小値，評価回数,エラーコード]
    エラーコード=0は，エラーなし   */

    fmin_cobyla(
    [w1,w2,w3].V.transpose([w1,w2,w3]) /* 目的関数：分散最小化 */,
    [w1,w2,w3] /* 変数: 投資比率 */,
    [0.3,0.3,0.4] /* 変数初期値 */,
    constraints=[
    [1,1, 1].[w1,w2,w3]=1 /* 比率の和=1 */,
    mu.[w1,w2,w3]=0.1      /* 期待収益率=0.1*/,
    w1>=0,w2>=0,w3>=0 /* 空売り制約 */
    ]);

--> m: floor((lmax(mu)-lmin(mu))/0.001)
    /* 束縛条件の mu_p を muの[最小値，最大値]で0.001 刻みで設定
    floor(x)はx以下の最大整数を返す*/ ;
    mu_p: makelist(lmin(mu)+i*0.001, i, 0, m);
    if mu_p[m] < lmax(mu) then mu_p: endcons(lmax(mu), mu_p);
    /* muの(最小値-最大値)が0.01で割り切れなかった場合の処理 */
    m:length(mu_p);

--> var_p:makelist(0,m) /* 最小分散リスト初期化 */;
    weight: zeromatrix(m,3) /* 最小分散投資比率リスト初期化*/;
```

```
--> for i:1 thru m do (
    result: fmin_cobyla(
    [w1,w2,w3].V.transpose([w1,w2,w3]),
    [w1,w2,w3],
    [0.3,0.3,0.4],
    constraints=[
    [1,1, 1].[w1,w2,w3]=1,
    mu.[w1,w2,w3]=mu_p[i],
    w1>=0,w2>=0,w3>=0
    ]),
    weight[i]: result[1], var_p[i]: result[2]
    );

--> sigma_p: sqrt(var_p);

--> load(draw);
    scatterplot(
    transpose(matrix(sigma_p,mu_p)), points_joined=true,
    xlabel="sigma", ylabel="mu",
    title="Minimum variance boundary") ;
```