Constrained Optimization

Intuition : Convert the constrained problem to unconstrained problem , then solve the unconstrained problem

Section 1 : Solving problems with equality constraints using the Lagrange method
Section 2 : Solving general non-linear programming problems in R
Section 3 : Penalty function methods & barrier function methods

Lagrange method

Original problem

min $f(x_1, x_2, \ldots, x_n)$

subject to $g_1(x_1, x_2, \ldots, x_n) = b_1$
$g_2(x_1, x_2, \ldots, x_n) = b_2$
$\vdots$
$g_n(x_1, x_2, \ldots, x_n) = b_m$

Step 1 : Formulate the Lagrangian

$\downarrow$ unconstrained optimisation

Optimal solution $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ occurs at

min $L(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m)$

$$= f(x_1, x_2, \ldots, x_n) + \sum_{i=1}^{m} \lambda_i \left[b_i - g_i(x_1, x_2, \ldots, x_n)\right]$$

Step 2: Formulate the necessary optimal condition

$\downarrow$

Necessary optimality condition

$$\frac{\partial L}{\partial \lambda_i} = b_i - g_i(x_1, x_2, \ldots, x_n) = 0$$

Note : for any feasible solution , $\lambda_i = 0$

Step 3: Solve the simultaneous equations and obtain critical points in the original problem

$\downarrow$

$x_1 =$ ___ , $x_2 =$ ___ , ... , $x_n =$ ___ ,
$\lambda_i =$ ___ , $\lambda_2 =$ ___ , .. , $\lambda_n =$ ___

$\Rightarrow$ possible $(x_1, x_2, \ldots, x_n)$ & z -values

Note : The method of Lagrange multiplier is not a powerful procedure. It may be impossible to solve the equations to obtain critical points. Or the number of critical points may be too large that it is not practical to identify the solution to the original problem.

Example 1 (single optimal solution)

$$\min\ z = 2x^2 + y^2 - xy - 8x - 3y$$
subject to $3x + y = 10$

$$L(x,y,\lambda) = 2x^2 + y^2 - xy - 8x - 3y + \lambda(10 - 3x - y)$$

$$\frac{\partial L}{\partial x} = 4x - 8 - 3\lambda = 0$$
$$\frac{\partial L}{\partial y} = 2y - x - 3 - \lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = 10 - 3x - y = 0$$

↓ Simultaneous equation (Gauss Jordan elimination)

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{\lambda} \end{pmatrix} = \begin{pmatrix} \frac{69}{28} \\ \frac{73}{28} \\ -\frac{1}{4} \end{pmatrix}$$

Example 2 (multiple optimal solution)

$$\min\ z = x^2 + 2y$$
subject to $x^2 + y^2 = 1$

$$L(x,y,\lambda) = x^2 + 2y + \lambda(1 - x^2 - y^2)$$

$$\frac{\partial L}{\partial x} = 2x - 2x\lambda = 0$$
$$\frac{\partial L}{\partial y} = 2 - 2y\lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = 1 - x^2 - y^2 = 0$$

↓

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{\lambda} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

↓

Critical points are $(x,y) = (0,1)$, $z = 1$ and
$(x,y) = (0,-1)$, $z = -2$ (solution to the original problem)

A general non-linear programming problem is normally expressed as follows :

$\min f(x) = \min f(x_1, x_2, \dots, x_n)$
subject to
$g_1(x_1, x_2, \dots, x_n) \leq b_1$
$g_2(x_1, x_2, \dots, x_n) \leq b_2$
$\vdots$
$g_m(x_1, x_2, \dots, x_n) \leq b_m$

- "$=$" constraint $g(x)=b \longrightarrow$ replace with $g(x) \leq b$ and $g(x) \geq b$
- "$\geq$" constraint $g(x) \geq b \longrightarrow -g(x) \leq -b$

Use case 1 : portfolio selection

An investor has \$P to invest in a set of n stocks and would like to know how much to invest in each stock. The investor would like a portfolio with a minimum expected return and a small risk, where risk = variance of return on the portfolio.

Notation

σ_i^2 = variance of yearly return from stock i
σ_{ij} = covariance of yearly return from stocks i and j
R_i = expected yearly return from stock i
G = lower bound on expected yearly return from total investment
S_i = upper bound on investment in stock i

Example (3 stocks)

$\min \ \mathrm{var}(x_1 R_1 P + x_2 R_2 P + x_3 R_3 P) = \sigma_1^2 x_1^2 + \sigma_2^2 x_2^2 + \sigma_3^2 x_3^2 + 2\sigma_{12} x_1 x_2 + 2\sigma_{13} x_1 x_3 + 2\sigma_{23} x_2 x_3$
subject to
$x_1 + x_2 + x_3 = 1$ —— all funds must be invested
$x_1 R_1 + x_2 R_2 + x_3 R_3 \geq G$ —— lower bound on expected return
$x_1 \leq S_1, \ x_2 \leq S_2, \ x_3 \leq S_3$ —— upper bound on investment in each stock

Suppose $\sigma_1^2 = (0.20)^2$ $\quad \sigma_2^2 = (0.15)^2$ $\quad \sigma_3^2 = (0.08)^2$ $\quad R_1 = 0.14$ $\quad R_2 = 0.11$ $\quad R_3 = 0.10$ $\quad G = 0.12$
$\sigma_{12} = 0.6 \times 0.20 \times 0.15$ $\quad \sigma_{13} = 0.4 \times 0.20 \times 0.08$ $\quad \sigma_{23} = 0.7 \times 0.15 \times 0.08$

```
eval_f <- function( x ) {
    return( list( "objective" = 0.04*x[1]^2 +
                                0.0225*x[2]^2 + 0.0064* x[3]^2 +
                                1.2*0.20*0.15*x[1]*x[2] +
                                0.8*0.20*0.08*x[1]*x[3] +
                                1.4*0.15*0.08*x[2]*x[3],
                  "gradient" = c( 0.08*x[1] + 1.2*0.20*0.15*x[2] +
                                  0.8*0.20*0.08*x[3],
                                  0.045*x[2] + 1.2*0.20*0.15*x[1] +
                                  1.4*0.15*0.08*x[3],
                                  0.0128*x[3] + 0.8*0.20*0.08*x[1] +
                                  1.4*0.15*0.08*x[2] ) ))}

# greater than equal constraint
eval_geq <- function( x ) {
    geq_constr <- ( -x[1] -x[2] -x[3] + 1)
    geq_grad <- c(-1, -1, -1)
    return( list( "constraints" = geq_constr, "jacobian" = geq_grad) )}

# inequality contraint
eval_gineq <- function( x ) {
    gineq_constr <- ( -0.14*x[1] -0.11*x[2] -0.10*x[3] + 0.12 )
    gineq_grad <-c (-0.14, -0.11, -0.10)
    return( list( "constraints" = gineq_constr, "jacobian" = gineq_grad) )}

# initial values
x0 <- c(0.3, 0.2, 0.5)

opts <- list( "algorithm" = "NLOPT_LD_SLSQP",
              "xtol_rel" = 1.0e-7,
              "maxeval" = 50000,
              "local_opts" = local_opts )

res <- nloptr( x0 = x0, eval_f = eval_f, eval_g_ineq = eval_gineq,
               eval_g_eq = eval_geq, lb = c(0,0,0), ub = c(1,1,1), opts = opts)
print( res )
```

Number of Iterations....: 54
Termination conditions: xtol_rel: 1e-07 maxeval: 50000
Number of inequality constraints: 1
Number of equality constraints: 1
Optimal value of objective function: 0.0148
Optimal value of controls: 0.5 0 0.5

Use case 2 (managerial economics model)

The daily demand for electricity during each period is related as follows:

$D_P = 60 - 0.5P_P + 0.1P_O$

$D_O = 40 - P_O + 0.1P_P$

where D_P and D_O are demands during peak and off-peak periods respectively.
It cost \$10/day to maintain 1 unit of electricity. Factory has max capacity C.
What is the best pricing strategy (P_P & P_O)?

max profit = revenue - cost

$$\text{revenue} = D_P P_P + D_P P_O$$
$$= (60 - 0.5P_P + 0.1P_O)P_P + (40 - P_O + 0.1P_P)P_O$$
$$= -0.5P_P^2 - P_O^2 + 0.2P_O P_P + 60P_P + 40P_O$$

$$\text{cost} = 10C$$

$$\therefore \min\ 0.5P_P^2 + P_O^2 - 0.2P_O P_P - 60P_P - 40P_O + 10C$$

subject to

$$60 - 0.5P_P + 0.1P_O \leq C$$
$$40 - P_O + 0.1P_P \leq C$$

```
eval_f <- function( x ) {
    return( list( "objective" = 0.5*x[1]^2+ x[2]^2 -0.2*x[1]*x[2] -60*x[1] -40*x[2] + 10*x[3],
                  "gradient" = c(x[1] -0.2*x[2] -60,
                                 2*x[2] -0.2*x[1] -40,
                                 10) ))
}

eval_g_ineq <- function( x ) {
    constr <- rbind( -0.5*x[1] + 0.1*x[2] -x[3] +60,
                     0.1*x[1] -x[2] -x[3]+ 40 )
    grad <- rbind( c(-0.5,0.1,-1) , c(0.1,-1,-1) )
    return( list( "constraints" = constr, "jacobian" = grad ) )
}

x0 <- c( 10,20,30 ) # initial values

opts <-list( "algorithm" = "NLOPT_LD_AUGLAG",
             "xtol_rel" = 1.0e-7,
             "maxeval" = 1000,
             "local_opts" = local_opts )

res <- nloptr( x0 = x0, eval_f = eval_f, eval_g_ineq = eval_g_ineq, opts = opts)
print( res )
```

Output:
Number of Iterations....: 149
Termination conditions: xtol_rel: 1e-07 maxeval: 1000
Number of inequality constraints: 2
Number of equality constraints: 0
Optimal value of objective function: -2202.29591593652
Optimal value of controls: 70.30612 26.53061 27.5

Conclusion:
Charge \$70.31 during peak load period.
Charge \$26.53 during off-peak load period
Total of 27.5 kwh capacity required.

Use case 3 (Inventory control)

▷ 3 types of costs with inventories:
① holding cost — opportunity cost as money is "tied up in the inventory"
② ordering cost — cost incurred per order, independent of order quantity
③ stockout cost — penalty per unit unsatisfied demand (eg forgone sales, deproved reputation)

▷ Find inventory quantity (Q) that min cost ⇐ economic order quantity (EOQ) model
↳ Simplify assumptions:
1. Demand (D) is constant
2. No stockouts allowed ⇒ total cost = ordering cost (C_o) + holding cost (C_h)

$$= C_o \times (\#\text{ orders per unit time}) + C_h \times (\text{average inventory})$$
$$= C_o \times \frac{D}{Q} + C_h \times \frac{Q}{2}$$

subject to $Q \leq$ warehouse capacity

E.g. $C_o = 25$, $C_h = 24\%$, purchase price = \$8/unit, $D = 16000$, warehouse capacity = 1500

```
eval_f <- function( x ) {
    return( list( "objective" = 25*60000/x + 0.24*8*x/2,
                  "gradient" = -25*60000/(x*x) + 0.24*8/2)) }

x0 = 1000

opts <- list("algorithm" = "NLOPT_LD_LBFGS", "xtol_rel" = 1.0e-8)

res <- nloptr( x0 = x0, eval_f = eval_f, opts = opts, ub = 1500)
print(res)
```

```
NLoptsolver status: 1 ( NLOPT_SUCCESS: Generic success return value. )
Number of Iterations....: 8
Termination conditions: xtol_rel: 1e-08
Number of inequality constraints: 0
Number of equality constraints: 0
Optimal value of objective function: 2400
Optimal value of controls: 1250
> eval_f(1250)
$objective
[1] 2400
$gradient
[1] 0
```

Use case 4 (classification by support vector machines)

$$\min \ \tfrac{1}{2}\|w\|^2 = \tfrac{1}{2}(w_1^2 + w_2^2)$$

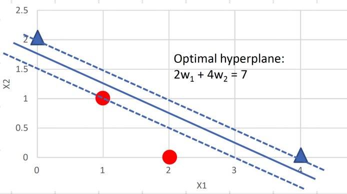
subject to $w_1 X_{k1} + w_2 X_{k2} + b \geq +1$ if $y_k = +1$

$w_1 X_{k1} + w_2 X_{k2} + b \leq -1$ if $y_k = -1$

General QP:
min $\frac{1}{2} x^T \underbrace{D}_{\text{Hessian}} x - \underbrace{d}_{\text{derivative of gradient (linear)}}{}^T x$
subject to $A^T x \geq b$

E.g. $\tfrac{1}{2}\|w\|^2 = \tfrac{1}{2}(w_1^2 + w_2^2)$

subject to
$2w_2 + b \geq +1$ △
$4w_1 + b \geq +1$ △
$w_1 + w_2 + b \leq -1$ ○
$2w_1 + b \leq -1$ ○



M1

```
eval_f <- function(w) {
    w1 <- w[1]
    w2 <- w[2]
    b  <- w[3]
    return(list("objective" = (1/2)*(w1*w1 + w2*w2),"gradient" = c(w1,w2,0) ))}

eval_g_ineq <- function(w) {
    w1 <- w[1]
    w2 <- w[2]
    b  <- w[3]
    constr <- rbind( -2*w2 -b+1, -4*w1 -b+1, w1 + w2 +b+ 1, 2*w1 + b +1 )

grad <- rbind( c(0,-2,-1) , c(-4,0,-1),c(1,1,1),c(2,0,1))
return( list( "constraints" = constr, "jacobian" = grad )) }

w0 <- c( 1,1,1) # initial values
opts <-list( "algorithm" = "NLOPT_LD_SLSQP", "xtol_rel" = 1.0e-7, "maxeval" = 5000)
res <- nloptr( x0=c(1,1,1), eval_f = eval_f, eval_g_ineq = eval_g_ineq, opts = opts)
```

```
Number of Iterations....: 3
Termination conditions: xtol_rel: 1e-07 maxeval: 5000
Number of inequality constraints: 4
Number of equality constraints: 0
Optimal value of objective function: 9.99
```

M2 — as such for all QP problems

```
library(quadprog)
Dmat <- matrix(c(1, 0, 0, 0, 1, 0, 0, 0, 1), 3, 3) # this value for all QP problems
Amat <- rbind(c(0, 2, 1),c( 4, 0, 1),c( -1, -1, -1),c( -2, 0, -1))
dvec <- c(0, 0, 0) # this value for all QP problems
bvec <- c(1, 1, 1, 1)
solve.QP(Dmat,dvec,t(Amat),bvec)
```

```
$solution
[1] 2 4 -7
$Lagrangian
[1] 21 10 38 0
```

$$w = \sum_i^N \alpha_i \underbrace{d_i}_{+1/-1} x_i$$
$$= \lambda_1 \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} - \lambda_3 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \lambda_4 \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$$
$$= 21 \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} + 10 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} - 38 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \underbrace{-\, 0}_{\text{not SV}} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$$
$$= \begin{pmatrix} 2 \\ 4 \\ -7 \end{pmatrix}$$

Penalty function methods

Original constrained problem

min $f(x)$

subject to $g_i(x) \leq 0 \quad i = 1, 2, \dots, m$

$h_i(x) = 0 \quad i = 1, 2, \dots, l$

$\downarrow$

min $f(x) + \mu\alpha(x)$ ⇐ auxillary function, where μ is a large positive number

$\alpha(x) = \sum_{i=1}^{m}[\max\{0, g_i(x)\}]^p + \sum_{i=1}^{l}[h(x)]^p$, for some positive integer p

Step 1 (initialisation): Let $\varepsilon > 0$ be a termination scalar. Choose an initial point, x_1, a penalty parameter μ_1 and a scalar $\beta > 1$. Let $\underbrace{k = 1}_{\text{iteration counter}}$.

Step 2 (optimization):

① Solve min $f(x_k) + \mu_k \alpha(x_k)$.

(i) Find derivative of $f(x_k) + \mu_k \alpha(x_k)$

(ii) Find minimum of $f(x_k) + \mu_k \alpha(x_k)$

Let x_{k+1} be the optimal solution.

② While $\mu_k \alpha(x_{k+1}) \geq \varepsilon$, $\underbrace{\mu_{k+1} = \beta\mu_k}_{\text{start with small penalty, increase gradually}}$, $k = k+1$, repeat ①.

$\Rightarrow \because$ problems with large μ is expensive to compute

The optimal solution of the penalty problem can be made arbitrarily close to the solution of the original problem by increasing the value of μ.

Example 1 (with single $\leq$ constraint)

min x

subject to $-x + 2 \leq 0$

$\downarrow$

min $f(x) + \mu\alpha(x)$, where $\alpha(x) = \begin{cases} 0 & , x \geq 2 \\ (-x+2)^2 & , x < 2 \end{cases}$

$\downarrow$

derivative of $f(x) + \mu\alpha(x) = \begin{cases} 1 & , x \geq 2 \\ 1 - 2\mu(-x+2) & , x < 2 \end{cases}$

$\downarrow$

minimum occurs at $x = 2 - \frac{1}{2\mu}$.

As $\mu \to \infty$, $x \to 2$.

Example 2 (single equality constraint)

$$\min \ (x_1-2)^4 + (x_1-2x_2)^2$$
$$\text{subject to} \quad x_1^2 - x_2 = 0$$

```
# penalty term
pt <- function(mu, x1, x2) {
    mu*(x1*x1 -x2)^2
}

# tolerance
tol <- 10^{-6}

# initial starting point
x0 <- c(2,1)

# initial penalty constant
mu <- 0.1

penalty_term <- pt(mu,x0[1],x0[2])

# initialize iteration counter
i = 0

while ((penalty_term> tol) || (i == 0)) {
    i <- i + 1
    print(i)

    fr <- function(x) {
        x1 <- x[1]
        x2 <- x[2]
        (x1 -2)^4 + (x1 -2*x2)^2 + mu*(x1*x1 -x2)^2
    }

    grr <- function(x) { ## Gradient of 'fr'
        x1 <- x[1]
        x2 <- x[2]
        c(4 *( x1 -2)^3 + 2 * (x1 -2*x2) + 4*mu * (x1*x1 -x2)*x1,
        -4* (x1 -2* x2) -2*mu * (x1*x1 -x2))
    }

    # sol <- optim( x0, fr, grr, method= "BFGS")
    sol <- lbfgs(fr,grr,x0,invisible=1) # another BFGS package from R

    cat ("Current solution ", sol$par[1],sol$par[2], "\n")
    x1 <-sol$par[1]
    x2 <-sol$par[2]
    fx <- (x1 - 2)^4 + (x1 -2*x2)^2

    penalty_term <- pt(mu,x1,x2)
    cat("I = ",i, "mu = ",mu," Penalty = ",penalty_term, " f(x) = ",fx, "\n")

    #update penalty constant and starting point for BFGS
    x0 <-sol$par
    mu <-10*mu
}
```

```
[1] 1
Current solution 1.453875 0.7607625
I = 1 mu = 0.1 Penalty = 0.1830581 f(x) = 0.09353118
[1] 2
Current solution 1.168725 0.7406732
I = 2 mu = 1 Penalty = 0.3909298 f(x) = 0.5752396
[1] 3
Current solution 0.9906151 0.8424581
I = 3 mu = 10 Penalty = 0.1928217 f(x) = 1.520125
[1] 4
Current solution 0.9507638 0.8874683
I = 4 mu = 100 Penalty = 0.02717041 f(x) = 1.891234
[1] 5
Current solution 0.9461094 0.8934415
I = 5 mu = 1000 Penalty = 0.002827601 f(x) = 1.940522
[1] 6
Current solution 0.9456357 0.8940584
I = 6 mu = 10000 Penalty = 0.0002839089 f(x) = 1.945616
[1] 7
Current solution 0.9455883 0.8941203
I = 7 mu = 1e+05 Penalty = 2.840254e-05 f(x) = 1.946127
[1] 8
Current solution 0.9455835 0.8941265
I = 8 mu = 1e+06 Penalty = 2.840366e-06 f(x) = 1.946178
[1] 9
Current solution 0.9455829 0.8941268
I = 9 mu = 1e+07 Penalty = 2.840374e-07 f(x) = 1.946183
```

Example 3 (single equality constraint)

$\min\ x_1^2 + x_2^2$

subject to $x_1 + x_2 - 1 = 0$

$\downarrow$

$\min\ f(x) + \mu\alpha(x) = x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)^2$

$\downarrow$

gradient of $f(x) + \mu\alpha(x)$ is

$2x_1 + \mu(x_1 + x_2 - 1) = 0$

$2x_2 + \mu(x_1 + x_2 - 1) = 0$ at minimum.

$\downarrow$

Solving the simultaneous equations, $x_1 = x_2 = \frac{\mu}{2\mu + 1}$.

As $\mu \to \infty$, $x_1 = x_2 \to \frac{1}{2}$

Barrier function methods

Original problem

$\min\ f(x)$

subject to $g_i(x) \le 0$ $\quad i = 1, 2, \ldots, m$

$\downarrow$

$\min\ f(x) + \mu B(x)$, where $\mu > 0$

Intuition: the function $B(x)$ sets a barrier against leaving the feasible region. It should take 0 value when $g_i(x) < 0$ and approach ∞ on its boundary.

- Default barrier function: $B(x) = \sum_{i=1}^{m} - \frac{1}{g_i(x)}$
- Alternative barrier function: $B(x) = -\sum_{i=1}^{m} \log(-g_i(x))$, logarithmic barrier function

Step 1 (initialisation). Let $\varepsilon > 0$ be a termination scalar. Choose an initial point, x_1, a barrier parameter μ, and a $\beta \in (0, 1)$. Let $k = 1$.

Step 2 (optimization):

① Solve $\min\ f(x_k) + \mu_k B(x_k)$ subject to $g(x) < 0$

(i) Find derivative of $f(x_k) + \mu_k B(x_k)$

(ii) Find minimum of $f(x_k) + \mu_k B(x_1)$

Let x_{k+1} be the optimal solution.

② While $\mu_k B(x_{k+1}) \ge \varepsilon$, $\mu_{k+1} = \beta\mu_k$, $k = k+1$, repeat ①.

Some problems have optimal solutions at the boundary. To ensure that obtained solution is as close to optimal solution as possible, μ_k decreases every iteration.

- There are computational difficulties associated with the barrier methods.
  - ↳ Initial point must be in the interior of feasible region, ie $g(x_0) < 0$. It may not be straightforward to find such a point.
  - ↳ If step size is too large, a line search algorithm may lead to a point outside the feasible region
  - ↳ Some line search algorithms may face difficulties when μ is close to 0 and the iterations are approaching the boundary of the feasible region.

Example 1

min x

subject to $-x+1 \leq 0$

$\downarrow$

$B(x) = -\frac{1}{-x+1}$ for $x \neq 1$

$\theta(x) = f(x) + \mu B(x) = x + \mu\left(-\frac{1}{-x+1}\right)$

$\downarrow$

$\theta'(x) = 1 - \mu(x-1)^{-2} = 0$ for $x > 1$

$x = 1 + \sqrt{\mu}$

$\downarrow$

As $\mu \to \infty$, $x \to 1$ and $\theta(x) \to f(x)$

Example 2 (min variance portfolio)

min $0.09x_1^2 + 0.04x_1x_2 + 0.06x_2^2$ —— min portfolio variance

subject to

$x_1 + x_2 = 1$ —— All funds must be invested

$x_1 \leq 0.75$ —— Upper bound on investment in Stock 1

$x_2 \leq 0.9$ —— Upper bound on investment in Stock 2

```
library(LowRankQP)

Amat <- rbind(c(1,1,0),
              c(0.06,0.02,-1)) # subtract surplus varfrom Const2

Vmat<- rbind(c(0.18, 0.04,0),
             c(0.04,0.12,0),
             c(0,0,0))

dvec <- c(0,0,0)
bvec <- c(1, 0.03)
uvec <- c(0.75, 0.90, 10000) # surplus varhas no bound

LowRankQP(Vmat,dvec,Amat,bvec,uvec,method="LU",niter=100000)
```

```
LowRankQP CONVERGED IN 13 ITERATIONS

Primal Feasibility = 1.2101977e-17
Dual Feasibility = 1.1796120e-16
Complementarity Value = 1.0325263e-12
Duality Gap = 1.0325213e-12
Termination Condition = 9.8763381e-13

$alpha
[,1]
[1,] 0.363636364
[2,] 0.636363636
[3,] 0.004545455
```