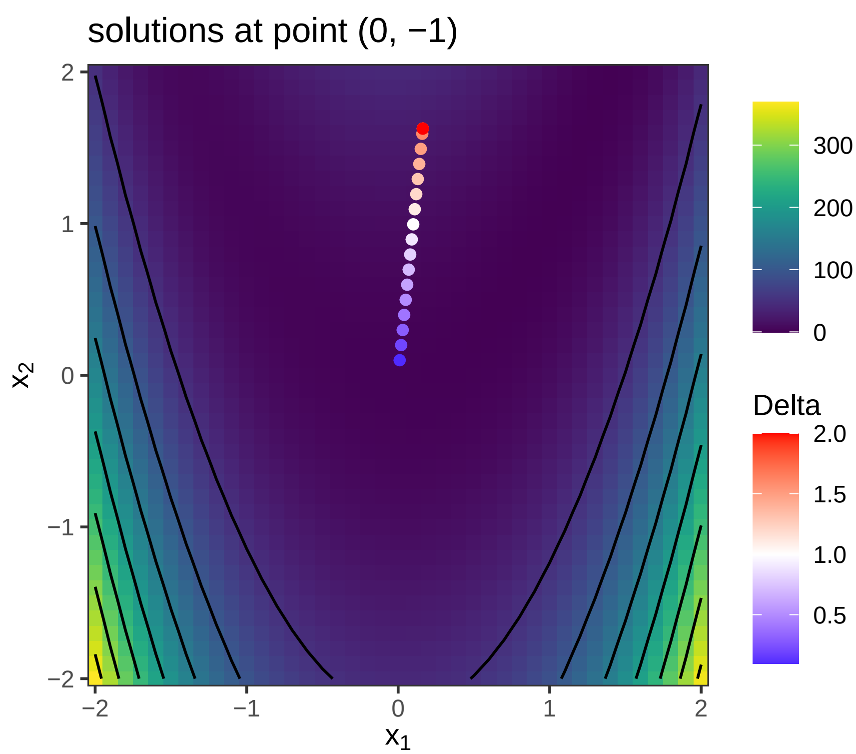
HW4

John Koo

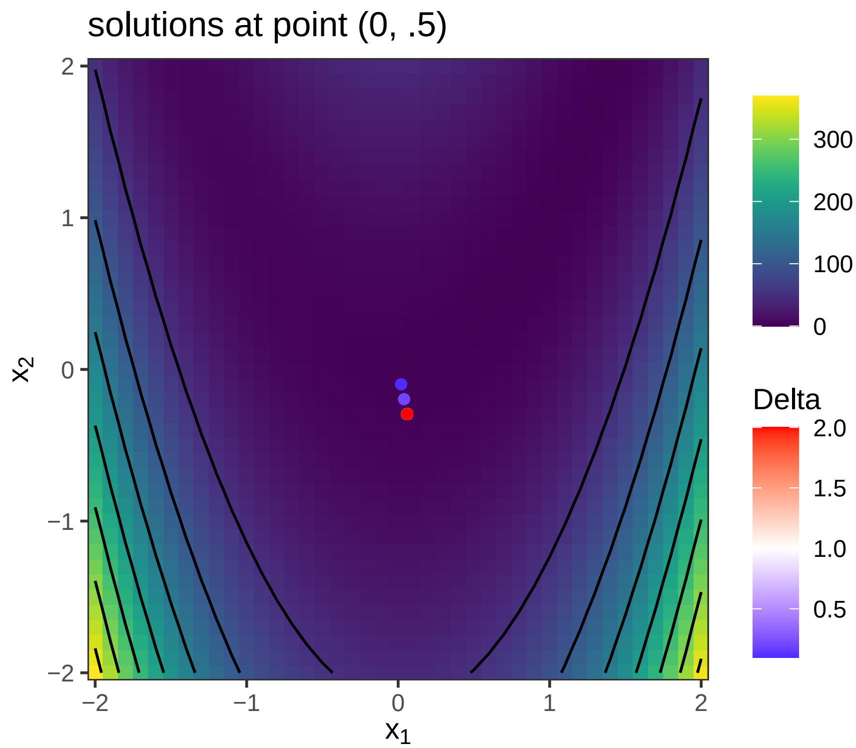
4.1

```
x1 <- seq(-2, 2, .1)
x2 <- seq(-2, 2, .1)
deltas <- seq(.1, 2, .1)

f <- function(x1, x2) 10 * (x2 - x1 ** 2) ** 2 + (1 - x1) ** 2
g <- function(x1, x2) {
  c(40 * x1 * (x1 ** 2 - x2) + 2 * (x1 - 1),
    20 * (x2 - x1 ** 2))
}
B <- function(x1, x2) {
  matrix(c(40 * (3 * x1 ** 2 - x2) + 2, -40, -40, 20),
         nrow = 2, ncol = 2)
}

cauchy <- function(x1, x2, Delta) {
  f.k <- f(x1, x2)
  g.k <- g(x1, x2)
  B.k <- B(x1, x2)

  g.k.norm <- sqrt(sum(g.k ** 2))
  gBg <- as.numeric(t(g.k) %*% B.k %*% g.k)

  if (gBg <= 0) {
    tau <- 1
  } else {
    tau <- min(g.k.norm ** 3 / Delta / gBg, 1)
  }

  return(-tau * Delta / g.k.norm * g.k)
}

solutions.1.df <- sapply(deltas, function(d) cauchy(0, -1, d)) %>%
  t() %>%
  as.data.frame() %>%
  dplyr::transmute(Delta = deltas, x1 = V1, x2 = V2)

solutions.2.df <- sapply(deltas, function(d) cauchy(0, .5, d)) %>%
  t() %>%
  as.data.frame() %>%
  dplyr::transmute(Delta = deltas, x1 = V1, x2 = V2)
```

```r
main.plot <- expand.grid(x1 = x1, x2 = x2) %>%
  dplyr::mutate(y = f(x1, x2)) %>%
  ggplot() +
  coord_fixed() +
  viridis::scale_fill_viridis() +
  geom_tile(aes(x = x1, y = x2, fill = y)) +
  geom_contour(aes(x = x1, y = x2, z = y),
               colour = 'black') +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  labs(x = expression(x[1]), y = expression(x[2]),
       fill = NULL)

main.plot +
  geom_point(data = solutions.1.df,
             aes(x = x1, y = x2, colour = Delta)) +
  ggtitle('solutions at point (0, -1)') +
  scale_colour_gradient2(low = 'blue', mid = 'white', high = 'red',
                         midpoint = 1)
```




```r
main.plot +
  geom_point(data = solutions.2.df,
             aes(x = x1, y = x2, colour = Delta)) +
  ggtitle('solutions at point (0, .5)') +
  scale_colour_gradient2(low = 'blue', mid = 'white', high = 'red',
                         midpoint = 1)
```



4.3

(Assuming 7.2 should be 4.1)

```
f <- function(x) {
  n <- length(x) / 2
  sapply(seq(n), function(i) {
    (1 - x[2 * i - 1]) ** 2 + 10 * (x[2 * i] - x[2 * i - 1] ** 2) ** 2
  }) %>%
    sum()
}
g <- function(x) {
  n <- length(x) / 2
  out <- rep(NA, 2 * n)
  for (i in seq(n)) {
    out[2 * i - 1] <-
      40 * x[2 * i - 1] * (x[2 * i - 1] ** 2 - x[2 * i]) +
      2 * (x[2 * i - 1] - 1)
    out[2 * i] <- 20 * (x[2 * i] - x[2 * i - 1] ** 2)
  }
  return(out)
}
B <- function(x) {
  n <- length(x) / 2
  out <- matrix(0, nrow = 2 * n, ncol = 2 * n)
  for (i in seq(n)) {
    out[2 * i - 1, 2 * i - 1] <- 40 * (3 * x[2 * i - 1] ** 2 + x[2 * i]) + 2
    out[2 * i, 2 * i] <- 20
    out[2 * i - 1, 2 * i] <- -40
    out[2 * i, 2 * i - 1] <- -40
```

```r
  }
  return(out)
}
```

```r
m <- function(p, x, f, g, B, ...) {
  f(x, ...) + t(g(x, ...)) %*% p + .5 * t(p) %*% B(x, ...) %*% p
}

rho <- function(x, p, f, g, B, ...) {
  z <- rep(0, length(x))
  (f(x, ...) - f(x + p, ...)) / (m(z, x, f, g, B, ...) - m(p, x, f, g, B, ...))
}

trust.region <- function(f, g, B,
                         x0,
                         method = 'dogleg',
                         Delta.hat = .01,
                         Delta0 = .001,
                         eta = .1,
                         maxiter = 1e3,
                         eps = 1e-4,
                         ...) {
  x <- x0
  Delta <- Delta0

  function.vals <- rep(NA, maxiter)

  iter <- 0

  while (sum(g(x) ** 2) > eps) {
    iter <- iter + 1
    if (iter > maxiter) {
      warning('failed to converge')
      break
    }

    g.k <- g(x, ...)
    g.k.norm <- sqrt(sum(g.k ** 2))
    B.k <- B(x, ...)

    # 4.12
    tau <- ifelse(t(g.k) %*% B.k %*% g.k <= 0,
                  1,
                  min(g.k.norm ** 3 / (Delta * t(g.k) %*% B.k %*% g.k), 1))
    tau <- as.numeric(tau)

    if (method == '4.2') {
      # 4.11
      p <- -tau * Delta / g.k.norm * g.k
    } else if (method == 'dogleg') {
      # 4.16
      p.U <- -as.numeric(g.k.norm ** 2 / (t(g.k) %*% B.k %*% g.k)) * g.k
      p.B <- -solve(B.k, g.k)
      if (tau <= 1) {
```

```r
        p <- tau * p.U
      } else {
        p <- p.U + (tau - 1) * (p.B - p.U)
      }
    }

    # 4.4
    rho.k <- as.numeric(rho(x, p, f, g, B, ...))

    # pg 69
    if (rho.k < .25) {
      Delta <- .25 * Delta
    } else {
      if ((rho.k > .75) & (sum(p ** 2) == Delta ** 2)) {
        Delta <- min(2 * Delta, Delta.hat)
      }
    }

    if (rho.k > eta) {
      x <- x + p
    }
    function.vals[iter] <- f(x, ...)
  }

  function.vals <- function.vals[!is.na(function.vals)]

  return(list(x = x,
              function.vals = function.vals))
}
```

```r
# x.start <- runif(20, -2, 2)
x.start <- rnorm(20, 1, .5)
# x.start <- rep(1.2, 20)
out <- trust.region(f, g, B, x.start, method = '4.2')
summary(out$x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.9914  0.9956  0.9973  0.9985  0.9994  1.0131 
```

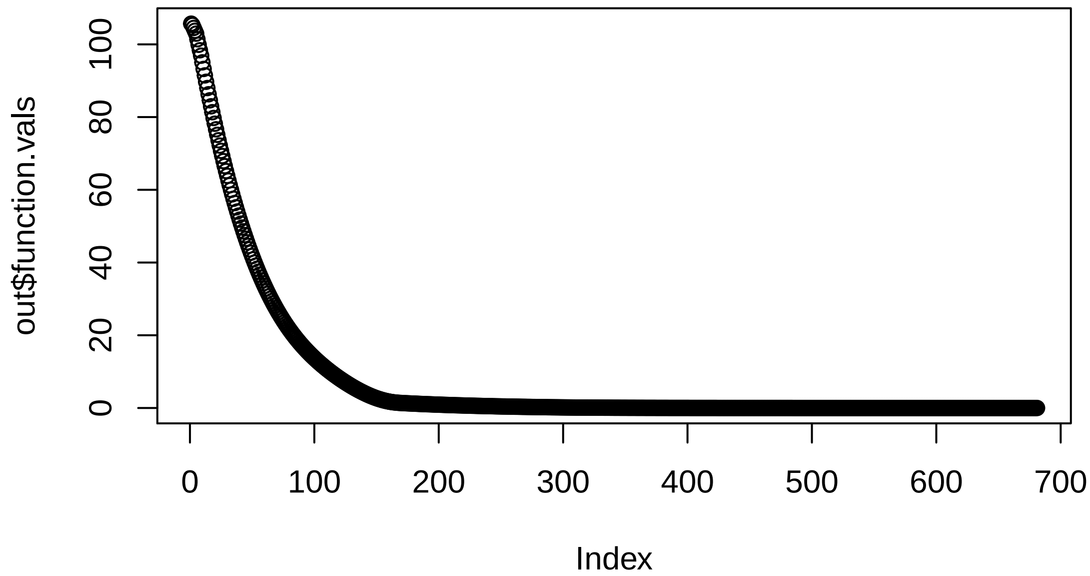
```r
f(out$x)
```

```
[1] 0.0001115101
```

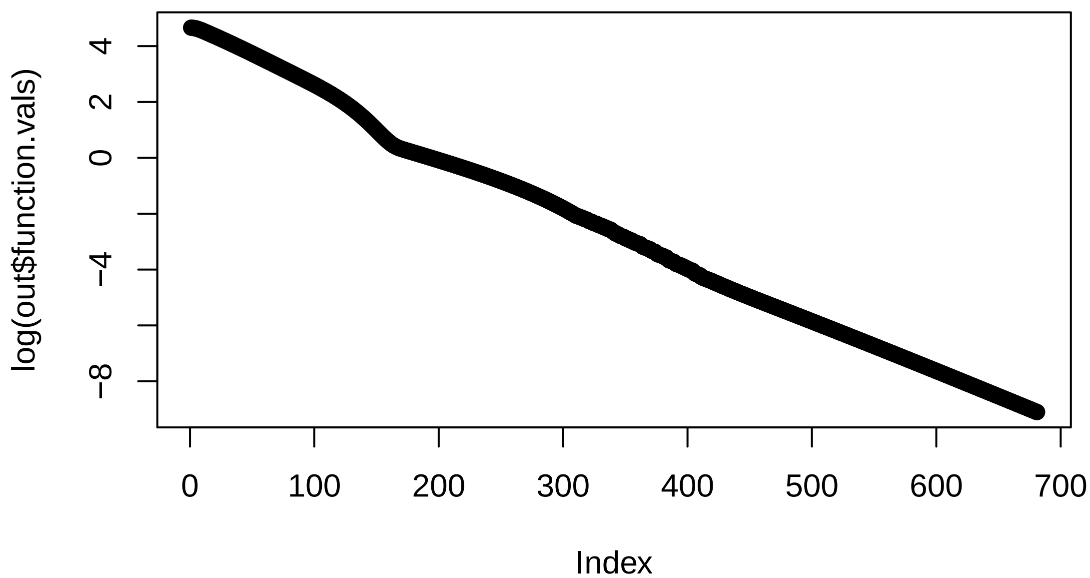
```r
sum(g(out$x) ** 2)
```

```
[1] 9.812568e-05
```

```r
plot(out$function.vals)
```




```
plot(log(out$function.vals))
```




```
# x.start <- runif(50, -2, 2)
x.start <- rnorm(50, 1, .5)
# x.start <- rep(1.2, 50)
out <- trust.region(f, g, B, x.start, method = '4.2')
summary(out$x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9942  0.9977  0.9993  1.0001  1.0028  1.0091
```

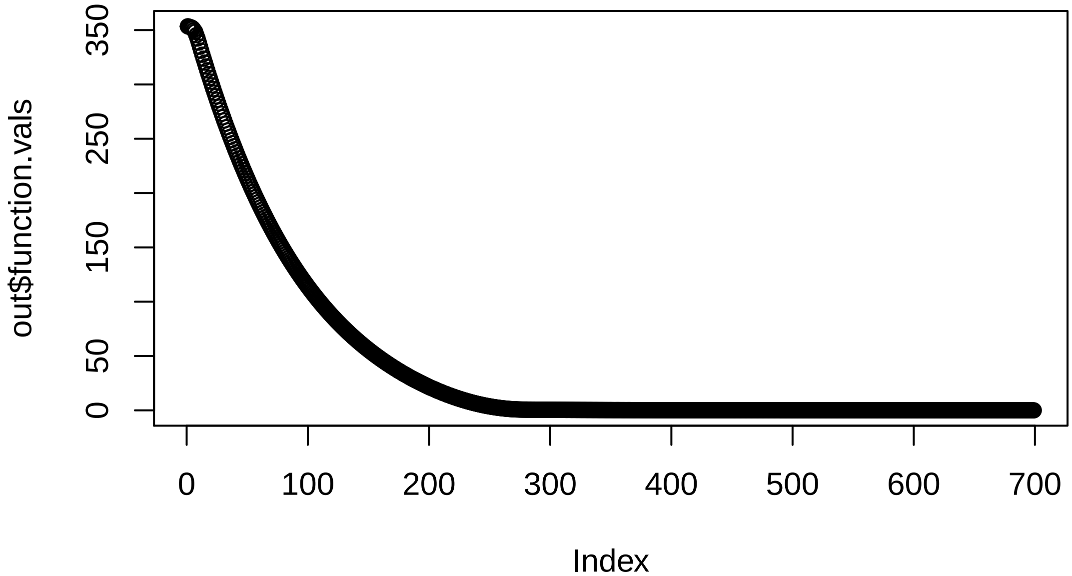
```
f(out$x)
```

```
[1] 0.0001179856
```

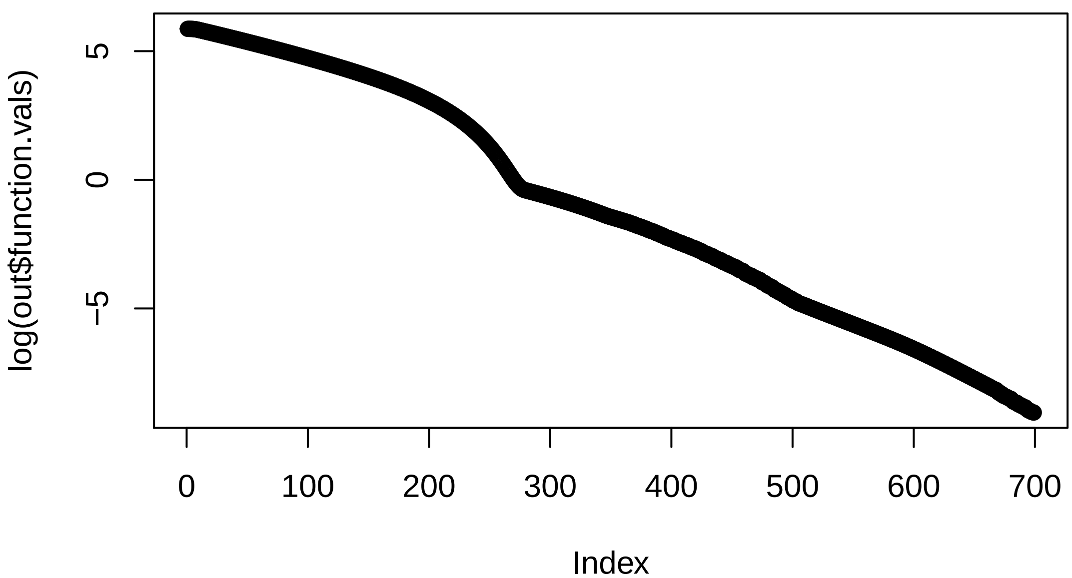
```
sum(g(out$x) ** 2)
```

```
[1] 9.274455e-05
```

```r
plot(out$function.vals)
```




```r
plot(log(out$function.vals))
```



4.4

Since $\{x_k\}$ is bounded in set $\mathcal{B}$ and $\liminf ||g_k|| = 0$, there exists a subsequence $||g_{k_j}||$ that is monotone decreasing to 0. $g_k = g(x_k)$ (and g is continuous), so there must be an accompanying subsequence $\{x_{k_j}\}$ such that $||g(x_{k_j})|| \to 0$.

4.5

$m_k(-\tau\frac{\Delta_k}{||g_k||}g_k) = f_k + g_k^\top(-\tau\frac{\Delta_k}{||g_k||}g_k) + \frac{1}{2}\frac{\Delta_k^2 g_k^\top B_k g_k}{||g_k||^2}\tau^2$

To minimize w.r.t. τ:
$\partial_\tau m_k = -\Delta_k||g_k|| + \frac{\Delta_k^2 g_k^\top B_k g_k}{||g_k||^2}\tau = 0$

$\rightarrow \tau = \frac{||g_k||^3}{\Delta_k g_k^\top B_k g_k}$

4.6

By matching, we can say $u = \frac{B^{1/2}g}{\sqrt{g^\top Bg}}$ and $v = \frac{B^{-1/2}g}{\sqrt{g^\top B^{-1}g}}$ to get the expression on the left-hand side.
Then the right hand side becomes $(u^\top u)(v^\top v) = \frac{(g^\top Bg)(g^\top B^{-1}g)}{\sqrt{(g^\top Bg)^2(g^\top B^{-1}g)^2}}$
$= 1$

4.9

Since our solution must be in $span(g, B^{-1}g)$, we can write it in the form $ag + bB^{-1}g$ where $a, b \in \mathbb{R}$.
$m(ag + bB^{-1}g) = f + g^\top(ag + bB^{-1}g) + \frac{1}{2}(ag + bB^{-1}g)^\top B(ag + bB^{-1}g)$
$= f + ||g||^2 a + g^\top B^{-1}gb + \frac{g^\top Bg}{2}a^2 + ||g||^2 ab + \frac{g^\top B^{-1}g}{2}b^2$

To minimize w.r.t. a and b, we take the partial derivatives and set them to 0 to obtain this system of linear equations:

$$\begin{bmatrix} g^\top Bg & ||g||^2 \\ ||g||^2 & (g^\top B^{-1}g) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -||g||^2 \\ -g^\top B^{-1}g \end{bmatrix}$$

4.10

Since B is symmetric, it can be decomposed as $B = VDV^\top$ where $VV^\top = I$ and D is a diagonal matrix. Then we can write $\lambda I = V\Lambda I$ where $\Lambda = diag_n(\lambda)$. So $B + \lambda I = V(D + \Lambda)V^\top$ where $D_{ii} + \lambda$ are the eigenvalues of $B + \lambda I$. Setting a large enough λ will then ensure that $D_{ii} + \lambda > 0 \ \forall i$.