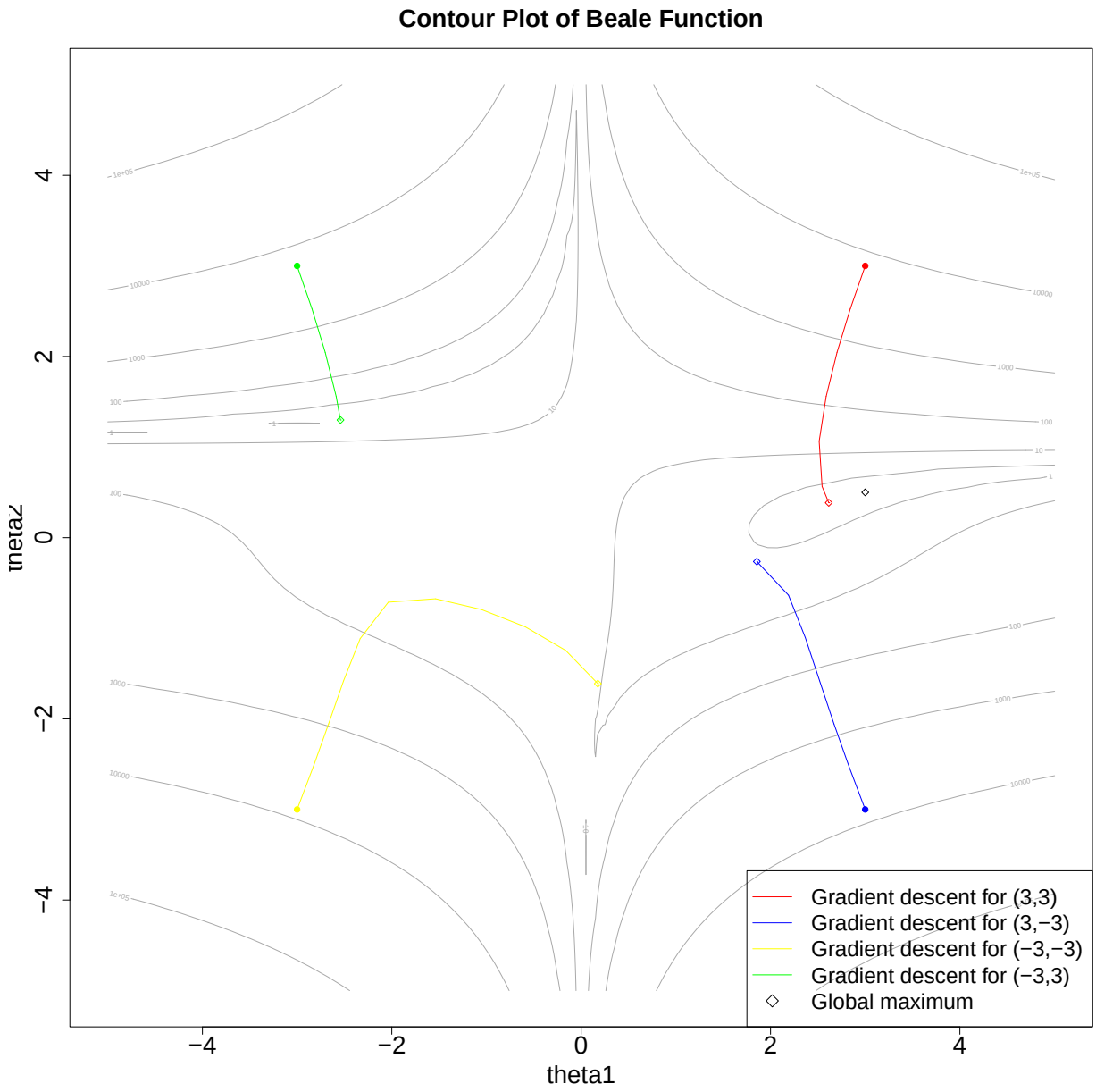
a2q2

```r
beale_rho <- function(theta) {
  theta1 <- theta[1]
  theta2 <- theta[2]
  return((1.5 - theta1 + theta1*theta2)^2 +
         (2.25 - theta1 + theta1*theta2^2)^2 +
         (2.625 - theta1 + theta1*theta2^3)^2)
}

beale_gradient <- function(theta) {
  theta1 <- theta[1]
  theta2 <- theta[2]
  grad1 <- 2*theta1*theta2^6 + 2*theta1*theta2^4 - 4*theta1*theta2^3 -
    2*theta1*theta2 - 4*theta1*theta2 + 6*theta1 + 3*theta2 + 4.5*theta2^2 - 12.75
  grad2 <- 6*theta1^2*theta2^5 - 6*theta1^2*theta2^2 + 4*theta1^2*theta2^3 -
    2*theta1^2*theta2 + 9*theta1*theta2 - 2*theta1^2 + 3*theta1 + 15.75
  return(c(grad1, grad2))
}

# Grid Line Search
gridLineSearch <- function(theta, rhoFn, d, lambdaStepsize = 0.01, lambdaMax = 1) {
  ## grid of lambda values to search
  lambdas <- seq(from = 0, by = lambdaStepsize, to = lambdaMax)
  ## line search
  rhoVals <- sapply(lambdas, function(lambda) {
    rhoFn(theta - lambda * d)
  })
  ## Return the lambda that gave the minimum
  lambdas[which.min(rhoVals)]
}

# Test Convergence
testConvergence <- function(thetaNew, thetaOld, tolerance = 1e-10, relative = FALSE) {
  sum(abs(thetaNew - thetaOld)) < if (relative)
    tolerance * sum(abs(thetaOld)) else tolerance
}


# d)

# Modified Gradient Descent
gradientDescent <- function(theta = 0, rhoFn, gradientFn, lineSearchFn, testConvergenceFn,
                            maxIterations = 100, tolerance = 1e-06, relative = FALSE, lambdaStepsize = 0
                            lambdaMax = 0.5) {

  converged <- FALSE
  i <- 0
```

```
  xpath <- c(theta[1])
  ypath <- c(theta[2])

  while (!converged & i <= maxIterations) {
    g <- gradientFn(theta)  ## gradient
    glength <- sqrt(sum(g^2))  ## gradient direction
    if (glength > 0)
      d <- g/glength

    lambda <- lineSearchFn(theta, rhoFn, d, lambdaStepsize = lambdaStepsize,
                           lambdaMax = lambdaMax)

    thetaNew <- theta - lambda * d
    converged <- testConvergenceFn(thetaNew, theta, tolerance = tolerance,
                                   relative = relative)
    theta <- thetaNew
    i <- i + 1
    xpath[i + 1] <- theta[1]
    ypath[i + 1] <- theta[2]
  }

  ## Return path
  list(xpath = xpath, ypath = ypath)
}

################################################################################

n_pts <- 100 # number of grid points per dimension
t1_surf <- seq(from = -5, to = 5, length.out = n_pts)
t2_surf <- seq(from = -5, to = 5, length.out = n_pts)

cont_mat <- matrix(0, nrow = n_pts, ncol = n_pts)

for (i in 1:n_pts) {
  for (j in 1:n_pts) {
    cont_mat[i, j] <- beale_rho(c(t1_surf[i], t2_surf[j]))
  }
}

levels <- c(10^0, 10^1, 10^2, 10^3, 10^4, 10^5)



# i.
path1 <- gradientDescent(theta = c(3, 3), rhoFn = beale_rho, gradientFn = beale_gradient,
                lineSearchFn = gridLineSearch, lambdaStepsize = 1e-03,
                testConvergenceFn = testConvergence, maxIterations = 500)

# ii.
path2 <- gradientDescent(theta = c(3, -3), rhoFn = beale_rho, gradientFn = beale_gradient,
                lineSearchFn = gridLineSearch, lambdaStepsize = 1e-03,
                testConvergenceFn = testConvergence, maxIterations = 500)
```

```r
# iii.
path3 <- gradientDescent(theta = c(-3, -3), rhoFn = beale_rho, gradientFn = beale_gradient,
                lineSearchFn = gridLineSearch, lambdaStepsize = 1e-03,
                testConvergenceFn = testConvergence, maxIterations = 500)

# iv.
path4 <- gradientDescent(theta = c(-3, 3), rhoFn = beale_rho, gradientFn = beale_gradient,
                lineSearchFn = gridLineSearch, lambdaStepsize = 1e-03,
                testConvergenceFn = testConvergence, maxIterations = 500)
```

```r
contour(
  x = t1_surf,
  y = t2_surf,
  z = cont_mat,
  levels = levels,
  col = "darkgrey",
  main = "Contour Plot of Beale Function",
  xlab = "theta1",
  ylab = "theta2",
  cex = 2,
  cex.lab = 2,
  cex.axis = 2,
  cex.main = 2
)

make_segments <- function(path, colour) {
  i <- 1
  points(x = path$xpath[1], y = path$ypath[1], col = colour, pch = 19)
  while (i != length(path$xpath)) {
    x0 <- path$xpath[i]
    y0 <- path$ypath[i]
    x1 <- path$xpath[i + 1]
    y1 <- path$ypath[i + 1]
    segments(x0 = x0, y0 = y0, x1 = x1, y1 = y1, col = colour, lwd = 1, lty = 1)
    i <- i + 1
  }
  points(x = path$xpath[i], y = path$ypath[i], col = colour, pch = 23)
}
############################################## ADD LABELS FOR Z VALUE OF ENDPOINTS
make_segments(path1, "red")
make_segments(path2, "blue")
make_segments(path3, "yellow")
make_segments(path4, "green")
global_min <- c(3, 0.5)
points(x = global_min[1], y = global_min[2], col = "black", pch = 23)
legend("bottomright",
       legend = c("Gradient descent for (3,3)",
                  "Gradient descent for (3,-3)",
                  "Gradient descent for (-3,-3)",
                  "Gradient descent for (-3,3)",
                  "Global maximum"),
       col = c("red", "blue", "yellow", "green", "black"),
       pch = c(NA, NA, NA, NA, 23), lty = c(1, 1, 1, 1, NA), lwd = 1, cex = 1.75)
```


Contour Plot of Beale Function
theta2
theta1
−4
−2
0
2
4
Gradient descent for (3,3)
Gradient descent for (3,−3)
Gradient descent for (−3,−3)
Gradient descent for (−3,3)
Global maximum