DATA 609 HW 3

Avery Davidowitz

2023-03-06

Ex. 1

Write down Newton's formula for finding the minimum of $f(x) = \frac{3x^4 - 4x^3}{12}$ Solution: Finding the minimum is equivalent to finding the roots for f'(x)=0. Newton's Method can solve this numerically using the iterative step of:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

In our case we have

$$x_{k+1} = x_k - \frac{x_k^2(x_k - 1)}{3x_k^2 - 2x_k)}$$

Implementing this algorithm in R

```
dfdx <- function(x) {
    x^2 * (x - 1)}

df2dx <- function(x) {
    3 * x^2 - 2 * x}

init <- 3
tol <- .00001

newton <- function(dfdx, df2dx, init, tol) {
    x = init
    while (abs(dfdx(x)) > tol) {
        x = x - dfdx(x)/df2dx(x)
        }
    x
    }

newton(dfdx, df2dx, init, tol)
```

```
## [1] 1
```

Ex. 2

Explore optimize() in R and try to solve the previous problem.

```
f <- function (x) (3*x^4 - 4*x^3)/12
xmin <- optimize(f, c(-10, 10), tol = .00001)
xmin$minimum
```

```
## [1] 0.9999986
```

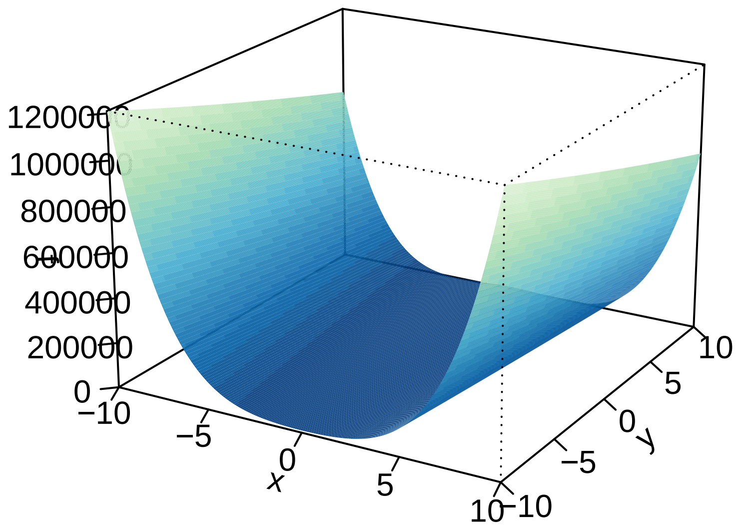
Ex. 3

Use any optimization algorithm to find the minimum of $f(x,y) = (x-1)^2 + 100(y-x^2)^2$ in the domain $-10 \le x, y \le 10$. Discuss any issues concerning the optimization process.

```
library(GA)
```

```
## Loading required package: foreach

## Loading required package: iterators

## Package 'GA' version 3.2.3
## Type 'citation("GA")' for citing this R package in publications.

##
## Attaching package: 'GA'

## The following object is masked from 'package:utils':
##
##     de
```

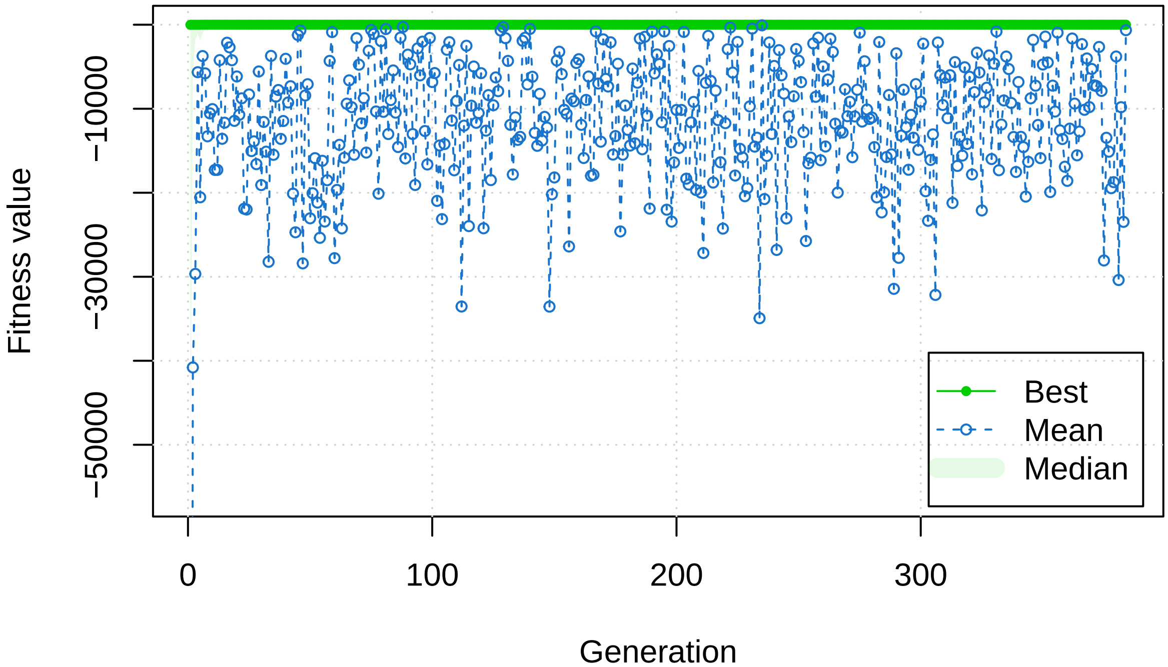
```
fxy <- function(x, y)
{
  (x - 1)^2 + 100*(y - x^2)^2
}

x <- y <- seq(-10, 10, by = 0.1)
f <- outer(x, y, fxy)
persp3D(x, y, f, col.palette = bl2gr.colors)
```




```r
GA <- ga(type = "real-valued",
         fitness =  function(x) -fxy(x[1], x[2]),
         lower = c(-10, -10), upper = c(10, 10),
         popSize = 100, maxiter = 1000, run = 100, seed = 160)
summary(GA)
```

```
## -- Genetic Algorithm -------------------
##
## GA settings:
## Type                  =  real-valued
## Population size       =  100
## Number of generations =  1000
## Elitism               =  5
## Crossover probability =  0.8
## Mutation probability  =  0.1
## Search domain =
##        x1  x2
## lower -10 -10
## upper  10  10
##
## GA results:
## Iterations             = 384
## Fitness function value = -4.729505e-07
## Solution =
##             x1       x2
## [1,] 1.000576 1.001115
```

```
plot(GA)
```



I did not really encounter any issues using the GA package to optimize this problem. I did find it hard to get the solution closer than 3 decimal places even after increasing the maximum iterations and the run parameter (which exits the algorithm after no solution improvement after that many iterations)

Ex. 4

Explore the optimr package for R and try to solve the previous problem.

```
library(optimr)
fxy2 <- function(par)
{
  (par[1] - 1)^2 + 100*(par[2] - par[1]^2)^2
}

optimr(par=c(.5,.5), fxy2, lower=c(-10,-10), upper=c(10,10), method='L-BFGS-B')
```

```
## $par
## [1] 0.9998686 0.9997375
##
## $value
## [1] 1.726918e-08
##
## $counts
```

```
## function gradient
##       46       46
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

It is interesting to note how much faster this bounded BFGS method converged to the solution than the genetic algorithm.