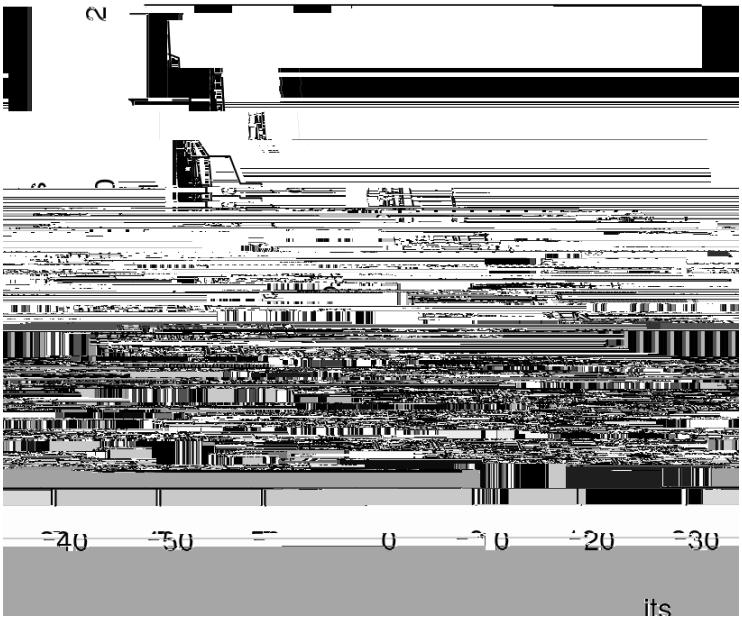
# Optimization

Observe that Step 2 involves its own inner-s"

```
while(1) {
    m <- which.min(y)
    opt <- optim(X[m,], obj.mean, lower=0, upper=1,
        method="L-BFGS-B", gpi=gpi)
    ynew <- f(opt$par)
    if(abs(ynew - y[length(y)]) < 1e-4) break;
    updateGPsep(gpi, matrix(opt$par, nrow=1), ynew)
    mle <- mleGPsep(gpi, param="d", tmin=da$min, tmax=da$max, ab=da$ab)
    X <- rbind(X, opt$par)
    y <- c(y, ynew)
}
```

# Progress

```
prog <- y
for(i in 2:length(y))
    if(prog[i] > prog[i-
```




# How does `optim` compare?

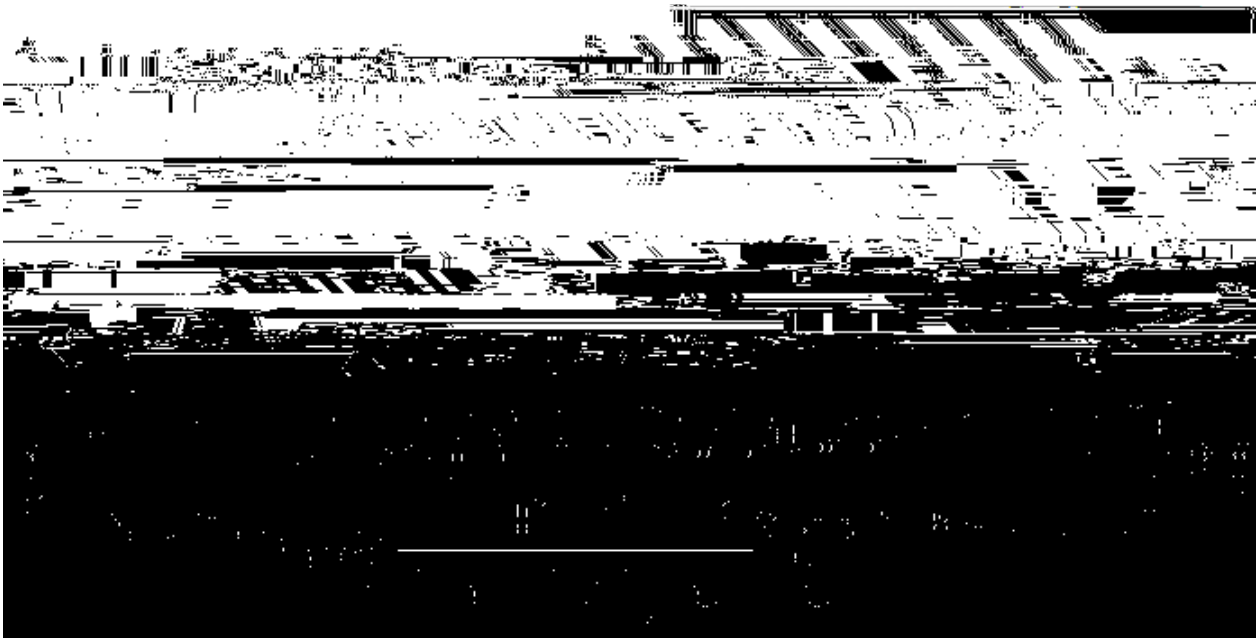
First, we need to modify  so we can keep track of the path of evaluations.

- because `optim` will call  to approximate gradients.

```
fprime <- function(x)
 {
    ynew <- f(x)
    y <<- c(y, ynew)
```

Even when is known, this is hard when is a blackbox.

# Visualizing the toy problem



- may seem uninteresting, but it reminds us that solutions may not exist oay not ex

# AL sequence

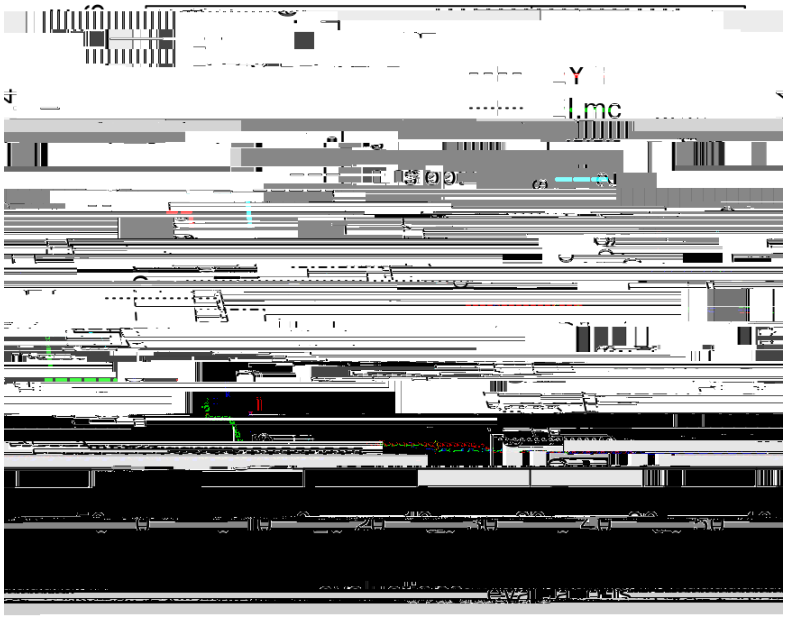
Given ,

1. approximately solve the **subproblem**

2. update:
    - , .
    - If , set ; otherwise, set
3. é then repeat, incrementing .

- Functions and are only evaluated when solving the

# Separated modeling



# Typical behavior

Average results after 100 restarts.