Optimizing methods

Day 2

Dominique Gravel

August 15th, 2017

UNIVERSITÉ DE SHERBROOKE

Optimization

Our objective is to find the set of parameters maximizing the likelihood of a model.

- Brute force/direct search
- Derivative-based methods
- Nelder-mean (simplex)
- Genetic algorithm
- Simulated annealing
- Monte Carlo Markov Chain
- Approximate Bayesian Computing

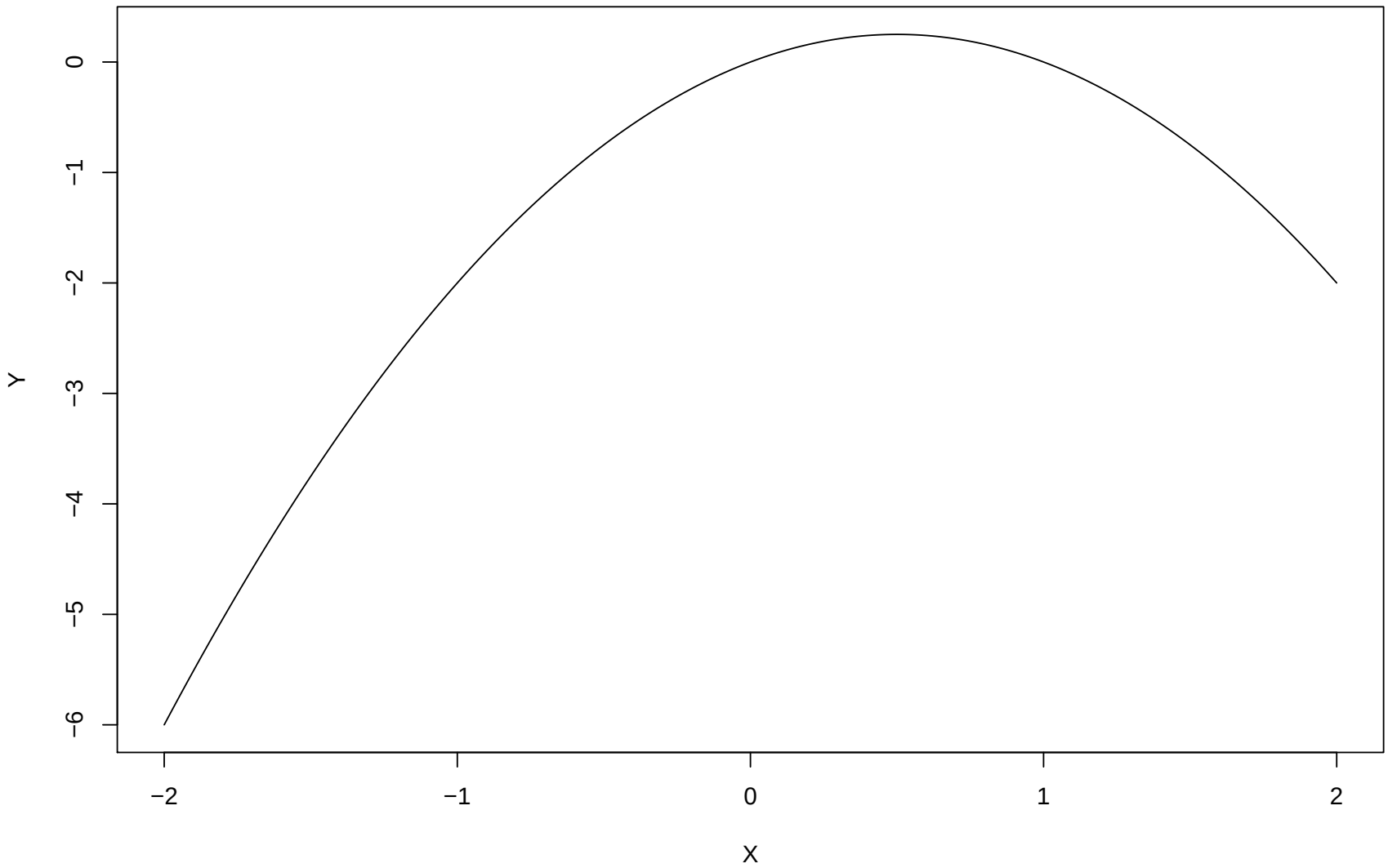
A classic from high school math

Our objective is to find the set of parameters maximizing the likelihood of a model.

$$Y = aX^2 + bX + c$$

A classic from high school math

Which gives



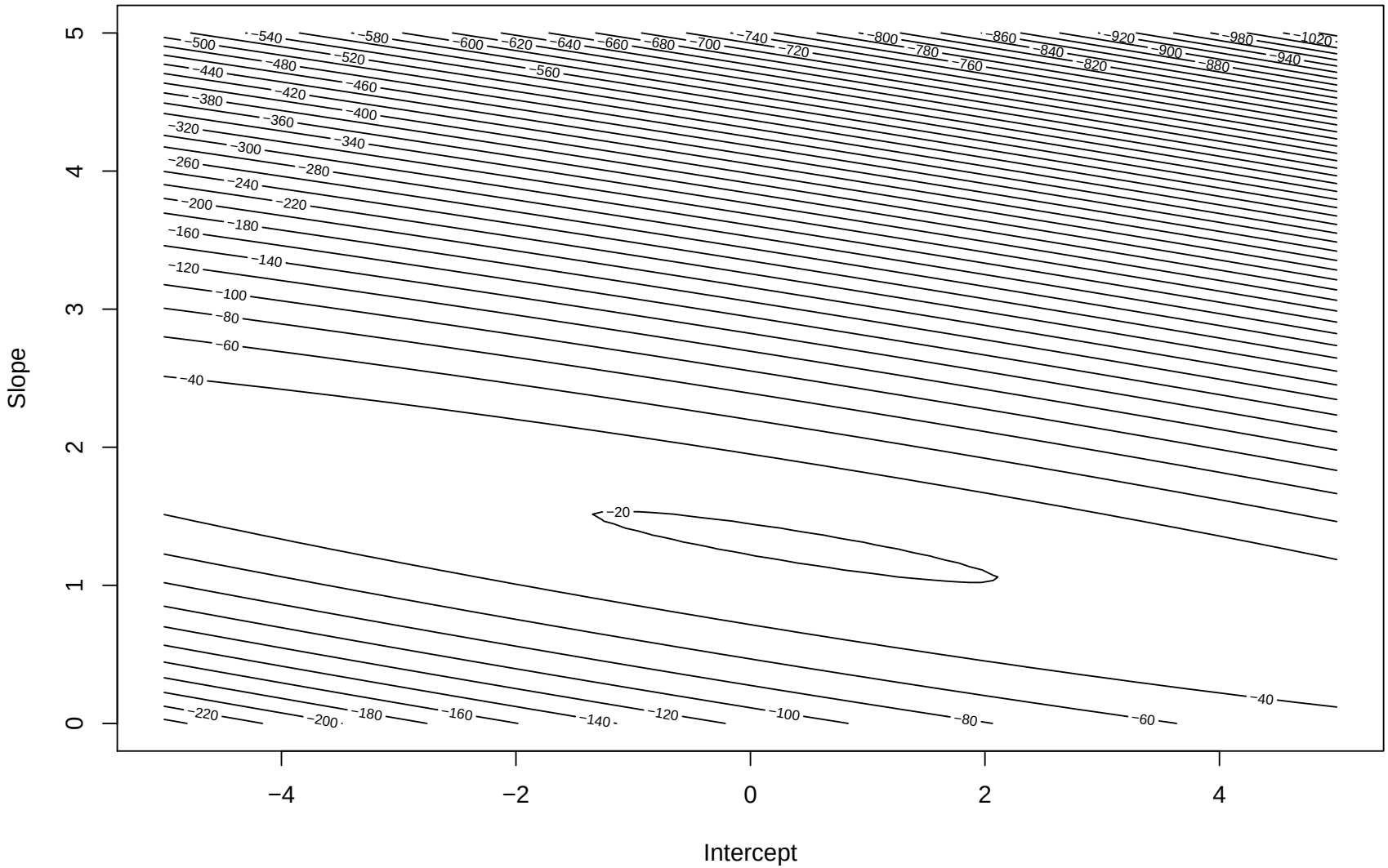
Many algorithms : Which one to pick ?

- The equation to optimize
- The size of the dataset
- Number of parameters
- Likelihood surface
- Area for the optimal solution or precise value ?
- Confidence interval ?

The brute force approach: grid search

```
X <- runif(n = 10, min = 0, max = 10)
Y <- rnorm(n = 10, mean = 0.5 + 1.2*X, sd = 1.5)
ll <- function(X,Y,a,b, sd) {
    pred <- a + b*X
    sum(dnorm(Y, pred, sd, log = TRUE))
}
a_seq <- seq(-5,5,length.out = 1000)
b_seq <- seq(0,5,length.out = 1000)
grid <- expand.grid(a_seq,b_seq)
res <- numeric(1000*1000)
for(i in 1:nrow(grid))
    res[i] <- ll(X,Y,grid[i,1],grid[i,2],sd=2)
z <- matrix(res, nr = 1000, nc = 1000)
contour(a_seq,b_seq,z, xlab = "Intercept",
    ylab = "Slope",nlevels=50)
```

The brute force approach: grid search



Nelder-Mead

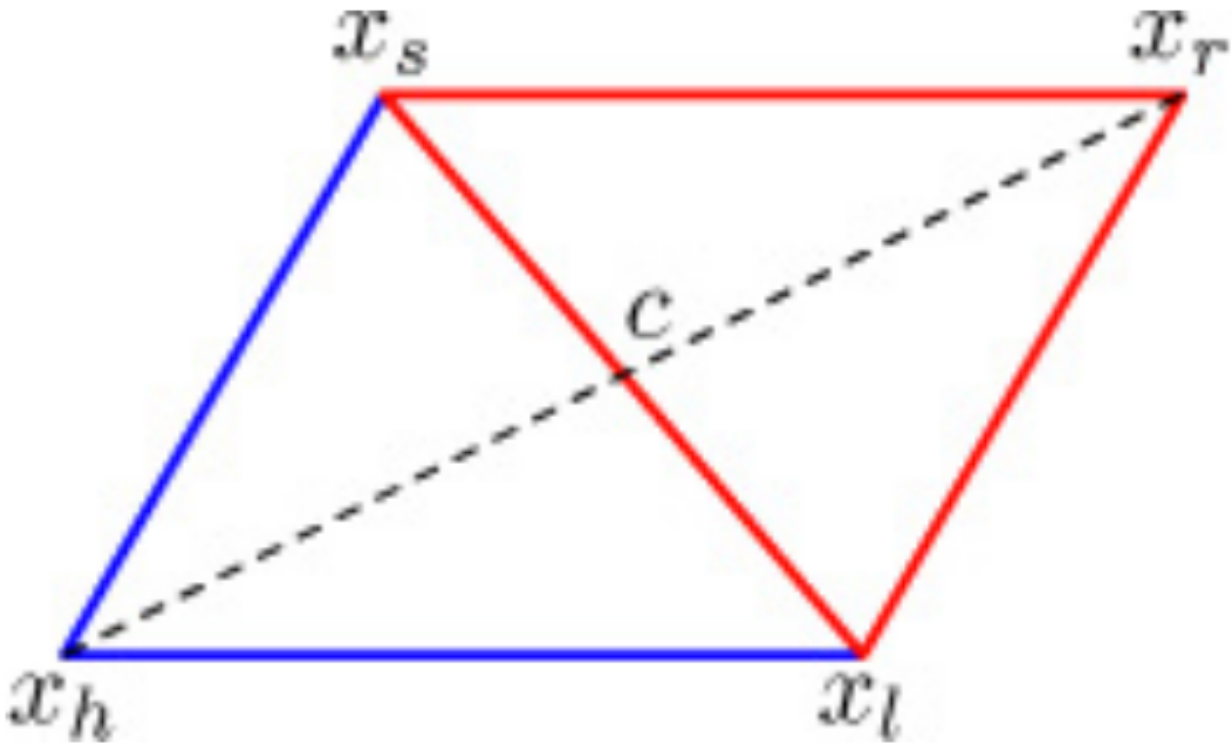
At best, we could simply compute the derivative of the model and solve it (i.e. find the parameter values for which the derivative equals 0).

Sometimes (often) it is just impossible to do it or to look at the entire likelihood surface. In such cases, we could use local optimization techniques such as the Nelder-Mean (simplex) method.

Pseudo-code

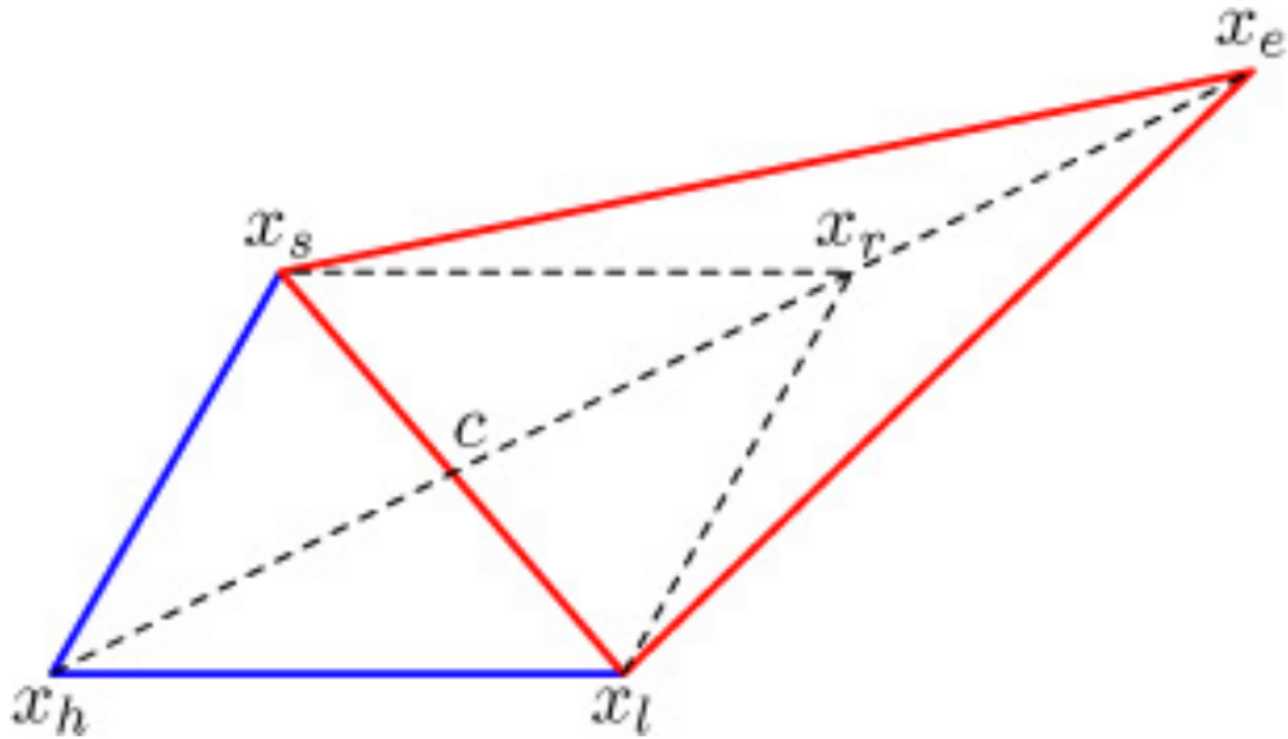
```
DEFINE function to optimize h(X)
DEFINE control parameters
DRAW initial simplex values
REPEAT
    EVALUATE the function h(x_i) for all $i$
    RANK the vertices $x_1, x_2, x_3$
    COMPUTE the centroid between the two best vertices
    PROPOSE new positions by reflection,
        expansion and contraction
  IF it succeeds
    ACCEPT the new vertex
  ELSE
    SHRINK the simplex towars the best vertex
UNTIL nsteps is reached
```

Reflection



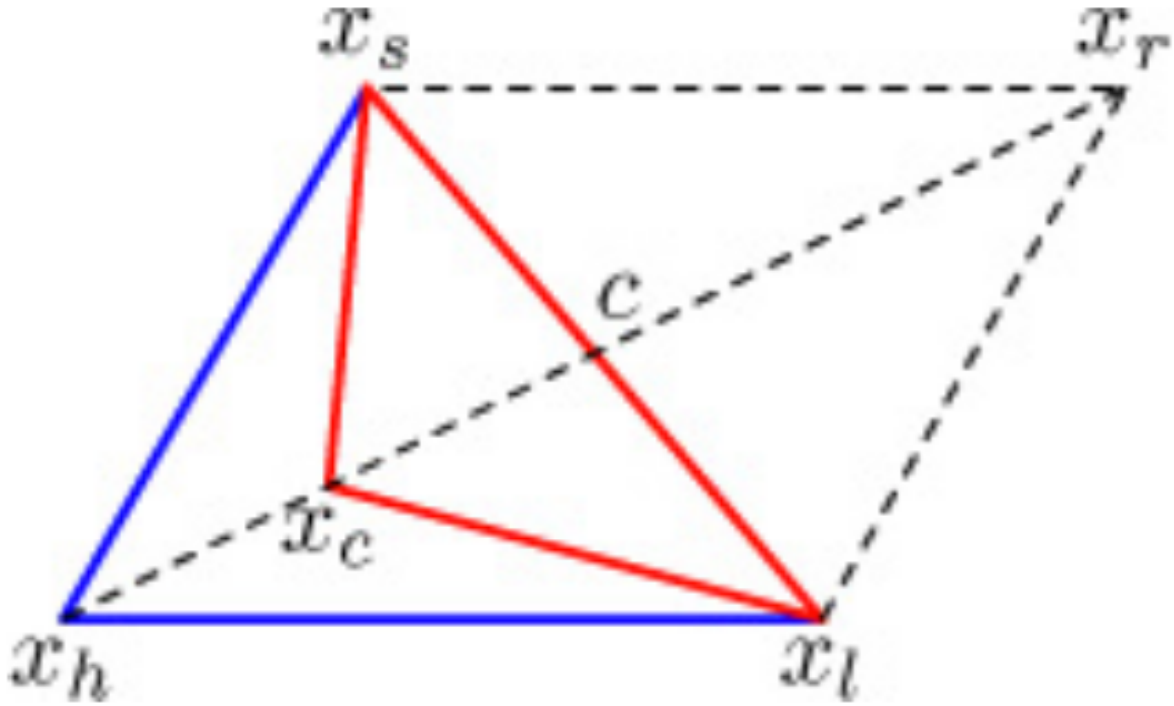
where $x_r = c + \alpha(c - x_h)$

Expansion



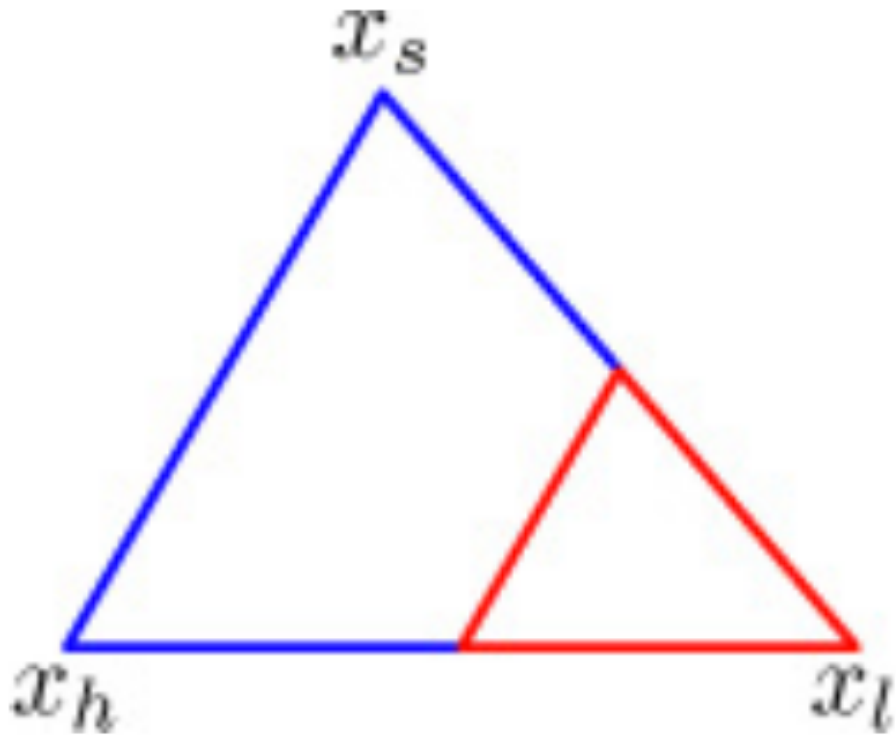
where $x_e = c + \gamma(x_r - c)$

Contraction



where $x_c = c + \beta(x_h - c)$

Shrinkage



where the new vertices are $x_j = x_l + \delta(x_j - x_l)$

Controls

α for reflection (>0, 1 by default)

β for contraction (bounded between 0 and 1, 0.5 by default)

γ for expansion (>1, 2 by default)

δ for shrinkage (bounded between 0 and 1, 0.5 by default)

Number of iterations OR any control parameter

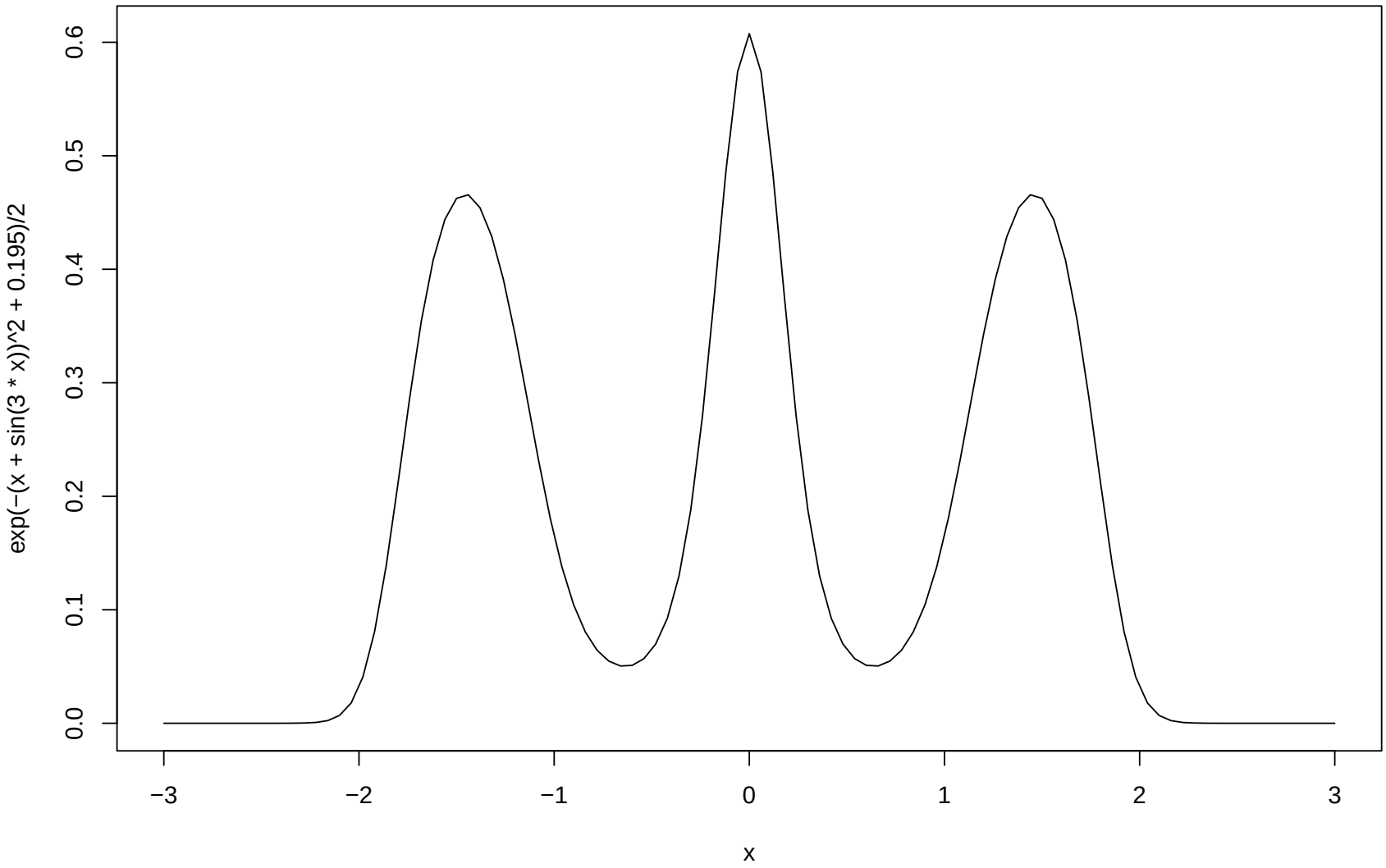
Time to practice

Find the optimal solution to the equation :

$$\frac{e^{-(x+\sin(3x))^2+0.195}}{2}$$

Time to practice

Which looks like :



Simulated annealing

The name of the method is coming from metallurgy, by analogy to the movement of atoms in a hot piece of metal as it cools down. Simulated annealing interprets cooling as a slow decrease in the probability of accepting a bad proposition of parameter values as it explores the solution space. Accepting worse solutions is a fundamental property of the method which allows an extensive search of the solution space and avoid getting stuck on local optima. The method is an adaptation of the Metropolis–Hastings algorithm.

Simulated annealing

Accept the candidate parameter value :

$$\theta_{t+1} = \theta_t + \delta$$

with probability:

$$\rho = \exp(\Delta h / T_t)$$

Pseudo-code

```
DEFINE function to optimize h(X)
DEFINE the sampling function c(X)
DEFINE temperature sequence
```

Pseudo-code

```
REPEAT
    DRAW sample X from c(x)
    COMPUTE difference diff = h(X) - h(X_0)
    COMPUTE acceptance probability p = exp(diff/T)
    DRAW value P from random uniform on (0,1)
    IF P < p
        ACCEPT X
    ELSE
        REJECT X
    UPDATE temperature
UNTIL nsteps is reached
```

Exercise

Take the previous function and find the maximum using simulated annealing

For advance users, compare the performance of the two methods in the precision of the solution and the time to solve it.

For the advance advance users, compute the derivative and solve it analytically!

Function to optimize

Initiating the algorithm

Main loop