HOMEWORK 2

Mathieu Gaillard

January 28, 2019

Problem 0: Homework checklist

I was out of the university during the whole week, I could not collaborate with anybody. I mainly used the class material, Google, Wikipedia and my previous knowledge.

Problem 1: Optimization software

The function we'll study is the Rosenbrock function:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Question 1:

```
using Plots
plotly(size = (1280, 720));

# Consider this function
f = (x, y) -> 100.0*(y - x.^2).^2 + (1 - x).^2;
# Create a meshgrid
x = -2:0.05:2;
y = -1:0.05:3;
X = repeat(x', length(y), 1);
Y = repeat(y, 1, length(x));
# Evaluate each f(x, y)
Z = map(f, X, Y);

# Contour plot
contour(x, y, Z);
# Contour plot with the log of the function (to see more details)
contour(x, y, log.(10.0, Z .+ 1.0), fill=true);
```

The contour plot of the Rosenbrock function is in Figure 1. There is also a contour plot of $\log(1 + f(x))$, which shows more details around the global minimum, in Figure 2.

Question 2:

Gradient and Hessian of the function.

$$\nabla f = \begin{pmatrix} -2(1 - x_1) - 400(x_2 - x_1^2)x_1 \\ 200(x_2 - x_1^2) \end{pmatrix}$$

$$H = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

Question 3:

The minimizer of the Rosenbrock function is the point $x^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

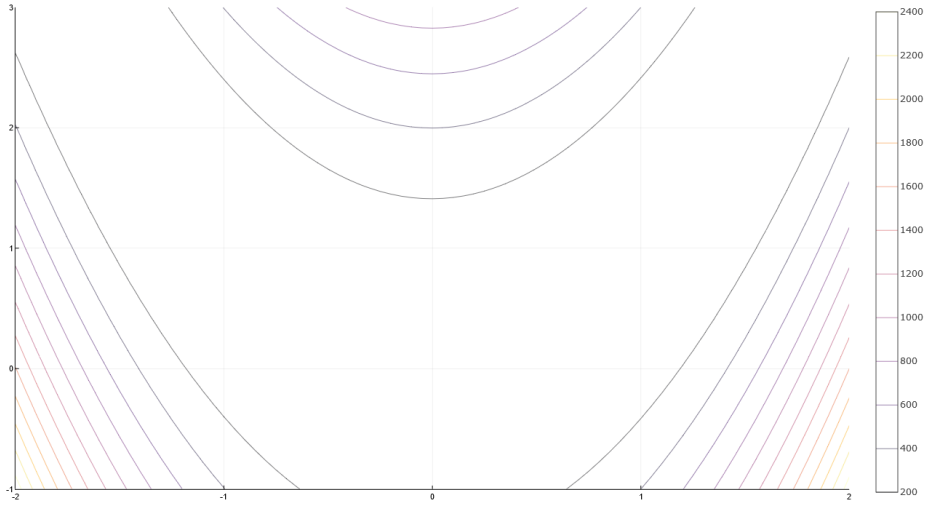

Figure 1: Contour plot of the Rosenbrock function.

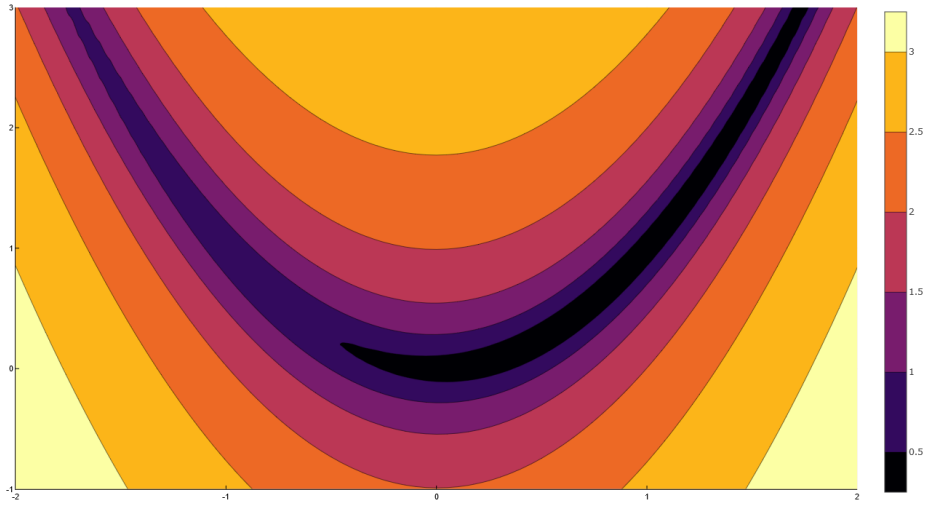

Figure 2: Contour plot of $\log(1 + f(x))$.

Because the Rosenbrock function is a polynomial, it is straightforward that:

$$\forall x \in \mathcal{R}^2, f(x) \geq 0$$

In addition to that, we notice that:

$$f(x^*) = 0$$

Thus, according to the definition, x^* is a global minimizer of the Rosenbrock function. Furthermore, this Wikipedia page states that it is the only global minimum: https://en.wikipedia.org/wiki/Test_functions_for_optimization.

Question 4:

An optimization package would test a sufficient condition for a strict local minimizer at each iteration. There are two conditions: $\nabla f(x^*) = \vec{0}$ and $\mathbf{H}(f(x^*))$

is positive definite. Of course the result of the gradient cannot be exactly 0, it would rather stop when it is within an epsilon of zero. Some implementations may not check that the Hessian matrix is positive definite as it is too time consuming.

Question 5:

```
using Plots, LinearAlgebra, Random, SparseArrays
using Optim

# Rosenbrock function
function f(x)
    return (1.0 - x[1])^2 + 100.0 * (x[2] - x[1]^2)^2;
end

function g!(storage::Vector, x::Vector)
    storage[1] = -2.0 * (1.0 - x[1]) - 400.0 * (x[2] - x[1]^2) * x[1];
    storage[2] = 200.0 * (x[2] - x[1]^2);
end

function h!(H::Matrix, x::Vector)
    H[1, 1] = 2.0 - 400.0 * x[2] + 1200.0 * x[1]^2
    H[1, 2] = -400.0 * x[1]
    H[2, 1] = -400.0 * x[1]
    H[2, 2] = 200.0
end

# Change the initial point if necessary
soln = optimize(f, g!, h!, [0.0, 0.0], GradientDescent())
soln = optimize(f, g!, h!, [0.0, 0.0], BFGS())
soln = optimize(f, g!, h!, [0.0, 0.0], NewtonTrustRegion())
```

Results given by this code are given in Table 1. We can see that Gradient Descent is never converging, the algorithm stops after 1000 iterations. Perhaps, with more iterations it could converge, but I did not try as better methods are available for this problem. BFGS and Newton Trust Region converge every time.

Starting point	Method	Convergence	Calls
$[0, 0]$	GradientDescent	False	2532
$[0, 0]$	BFGS	True	53
$[0, 0]$	NewtonTrustRegion	True	25
$[0, 2]$	GradientDescent	False	2531
$[0, 2]$	BFGS	True	70
$[0, 2]$	NewtonTrustRegion	True	16
$[1.5, 2]$	GradientDescent	False	2548
$[1.5, 2]$	BFGS	True	34
$[1.5, 2]$	NewtonTrustRegion	True	14

Table 1: Convergence of the optimization according to various starting points and methods.

Problem 2: Optimization theory

Suppose that $f : \mathbb{R} \to \mathbb{R}$ (i.e. is univariate) and is four times continuously differentiable. Show that the following conditions imply that x^* is a local minimizer.

i. $f'(x^*) = 0$

ii. $f''(x^*) = 0$

iii. $f'''(x^*) = 0$

iv. $f''''(x^*) > 0$

Because f is four times continuously differentiable, f'''' is still positive in a small neighborhood:

$$\exists r > 0, |x - x^*| < r \implies f''''(x) > 0$$

The fourth order Lagrange form of the Taylor theorem for f in x^* with h in the small neighborhood:

$$\forall |h| < r, \exists \alpha \in [0,1], f(x^*+h) = f(x^*)+\frac{h}{1!}f'(x^*)+\frac{h^2}{2!}f''(x^*)+\frac{h^3}{3!}f'''(x^*)+\frac{h^4}{4!}f''''(x^*+\alpha h)$$

Because the first three derivatives are equal to zero:

$$\forall |h| < r, \exists \alpha \in [0,1], f(x^* + h) = f(x^*) + \frac{h^4}{4!}f''''(x^* + \alpha h)$$

Since $x^* + \alpha h$ is in the small neighborhood in which f'''' is still positive: $f''''(x^* + \alpha\ h) > 0$

$$|(x^* + \alpha h) - x^*| = |\alpha h| < r \implies f''''(x^* + \alpha h) > 0$$

Therefore, x^* is a strict local minimizer:

$$\forall |h| < r, f(x^* + h) > f(x^*)$$

Q.E.D.

Problem 3: Optimization software

The function we'll study is the Booth function:

$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2.$$

```
using Plots
plotly(size = (1280, 720));

# Consider this function
f = (x, y) -> (x + 2.0*y - 7.0)^2 + (2.0*x + y - 5.0)^2;
# Create a meshgrid
x = -1:0.05:3;
y = 1:0.05:5;
X = repeat(x', length(y), 1);
Y = repeat(y, 1, length(x));
# Evaluate each f(x, y)
Z = map(f, X, Y);

# Contour plot
contour(x, y, Z);
```

The contour plot of the Booth function is in Figure 3.

Question 2:

Gradient and Hessian of the function.

$$\nabla f = \begin{pmatrix} 10x_1 + 8x_2 - 34 \\ 8x_1 + 10x_2 - 38 \end{pmatrix}$$

$$H = \begin{bmatrix} 10 & 8 \\ 8 & 10 \end{bmatrix}$$

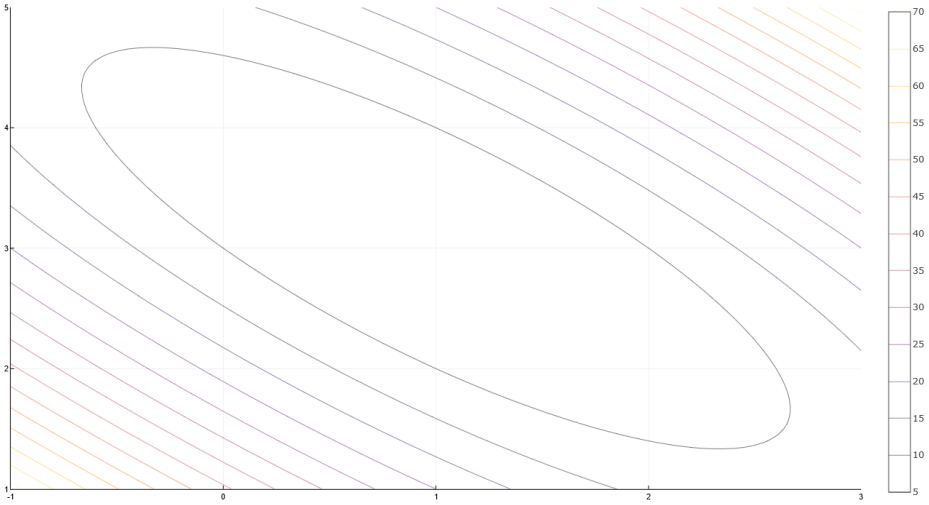

Figure 3: Contour plot of the Booth function.

Question 3:

The minimizer of the Booth function is the point $x^* = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$.

Because the Booth function is a polynomial, it is straightforward that:

$$\forall x \in \mathcal{R}^2, f(x) \geq 0$$

In addition to that, we notice that:

$$f(x^*) = 0$$

Thus, according to the definition, x^* is a global minimizer of the Booth function. Furthermore, this Wikipedia page states that it is the only global minimum: https://en.wikipedia.org/wiki/Test_functions_for_optimization.

Question 4:

See Problem 1 Question 4 as it is the same answer.

Question 5:

```
using Plots, LinearAlgebra, Random, SparseArrays
using Optim

# example of Booth function
function f(x)
    return (x[1] + 2.0*x[2] - 7.0)^2 + (2.0*x[1] + x[2] - 5.0)^2;
end

function g!(storage::Vector, x::Vector)
    storage[1] = 10.0*x[1] + 8.0*x[2] - 34;
    storage[2] = 8.0*x[1] + 10.0*x[2] - 38;
end

function h!(H::Matrix, x::Vector)
    H[1, 1] = 10.0;
    H[1, 2] = 8.0;
    H[2, 1] = 8.0;
```

```
    H[2, 2] = 10.0;
end

# Uncomment the necessary optimizer before running the script
# soln = optimize(f, g!, h!, [3.0, 3.0], NelderMead())
# soln = optimize(f, g!, h!, [3.0, 3.0], SimulatedAnnealing())
# soln = optimize(f, g!, h!, [3.0, 3.0], BFGS())
# soln = optimize(f, g!, h!, [3.0, 3.0], LBFGS())
# soln = optimize(f, g!, h!, [3.0, 3.0], ConjugateGradient())
# soln = optimize(f, g!, h!, [3.0, 3.0], GradientDescent())
# soln = optimize(f, g!, h!, [3.0, 3.0], MomentumGradientDescent())
# soln = optimize(f, g!, h!, [3.0, 3.0], AcceleratedGradientDescent())
# soln = optimize(f, g!, h!, [3.0, 3.0], Newton())
# soln = optimize(f, g!, h!, [3.0, 3.0], NewtonTrustRegion())
```

I tried every method implemented in Optim.jl;

The following methods only require the function:

- NelderMead
- SimulatedAnnealing

The following methods require the function and the gradient:

- BFGS
- LBFGS
- ConjugateGradient
- GradientDescent
- MomentumGradientDescent
- AcceleratedGradientDescent

The following methods require the function, the gradient and the Hessian matrix:

- Newton
- NewtonTrustRegion

The number of iterations when optimizing the Booth function starting from the point $[3, 3]$ with all the methods above are detailed in Table 2. Simulated Annealing does not converge after 1000 iterations. The method that does not require the gradient and takes the most iterations is **NelderMead**. The method that requires only the gradient and takes the most iterations is **MomentumGradientDescent**. The method that requires the gradient and Hessian matrix and that takes the most iterations is **NewtonTrustRegion**.

Method	Convergence	Iterations	Calls
NelderMead	True	36	74
SimulatedAnnealing	False	1000	1001
BFGS	True	2	5
LBFGS	True	2	7
ConjugateGradient	True	2	5
GradientDescent	True	17	51
MomentumGradientDescent	True	19	71
AcceleratedGradientDescent	True	11	40
Newton	True	1	2
NewtonTrustRegion	True	2	3

Table 2: Number of iterations when optimizing the Booth function starting from the point $[3, 3]$.

Problem 4: Convexity

Suppose that $f(x) = x^T Q x$, where Q is an $n \times n$ symmetric positive semi-definite matrix. Show that this function is convex using the definition of convexity, which can be equivalently reformulated:

$$f(y + \alpha(x - y)) - \alpha f(x) - (1 - \alpha) f(y) \leq 0$$

for all $0 \leq \alpha \leq 1$ and all $x, y \in \mathbb{R}^n$.

$$\begin{aligned}
&f(y+\alpha(x-y)) - \alpha f(x) - (1-\alpha) f(y) \\
&\quad = (y + \alpha(x-y))^T Q (y + \alpha(x-y)) - \alpha x^T Q x - (1-\alpha) y^T Q y \\
&\quad = y^T Q y + \alpha y^T Q x - \alpha y^T Q y + \alpha x^T Q y - \alpha y^T Q y \\
&\quad + \alpha^2 x^T Q x - \alpha^2 x^T Q y - \alpha^2 y^T Q x + \alpha^2 y^T Q y \\
&\quad - \alpha x^T Q x - (1-\alpha) y^T Q y \\
&\quad = \alpha(\alpha - 1)(y^T Q y - y^T Q x - x^T Q y + x^T Q x) \\
&\quad = \alpha(\alpha - 1)(x-y)^T Q (x-y)
\end{aligned}$$

Because Q is an $n \times n$ symmetric positive semi-definite matrix:

$$\forall (x, y) \in \mathbb{R}^n, (x-y)^T Q (x-y) \geq 0$$

Because $\alpha \in [0, 1]$:

$$\forall \alpha \in [0, 1], \alpha(\alpha - 1) \leq 0$$

Therefore:

$$\forall \alpha \in [0, 1], \forall (x, y) \in \mathbb{R}^n, f(y + \alpha(x-y)) - \alpha f(x) - (1-\alpha) f(y) \leq 0$$

Q.E.D.