Optimization

Introduction

Pierre de Fermat

- Pierre de Fermat, 1601 (Beaumont-de-Lomagne, near Montauban) – 1665 (Castres)
- method *maximis et minimis*
- Early developments that led to infinitesimal calculus

The problem is to solve

$$(P)\begin{cases} Min\ f(x) \\ x \in \mathbf{R}^n \end{cases}$$

Automatique differentiation

▼ Code

```julia
using LinearAlgebra
using ForwardDiff
A = [-1 2;1 4]; b=[1,1]; c=1
f(x) = 0.5*x'*A*x + b'*x +c

analytic_∇f(x) = 0.5*(A' + A)*x+b
analytic_∇²f(x) = 0.5*(A' + A)

∇f(x) = ForwardDiff.gradient(f, x)
∇²f(x) = ForwardDiff.hessian(f, x)

x0 = [1,-1]
println("∇f(x0) = ", ∇f(x0))
println("analytic_∇f(x0) = ", analytic_∇f(x0))
println("analytic_∇f(x0) - ∇f(x0) = ", analytic_∇f(x0)- ∇f(x0))

println("∇²f(x0) = ", ∇²f(x0))
println("analytic_∇²f(x0) = ", analytic_∇²f(x0))
println("analytic_∇²f(x0) - ∇²f(x0) = ", analytic_∇²f(x0)- ∇²f(x0))
```

```
∇f(x0) = [-1.5, -1.5]
analytic_∇f(x0) = [-1.5, -1.5]
analytic_∇f(x0) - ∇f(x0) = [0.0, 0.0]
∇²f(x0) = [-1.0 1.5; 1.5 4.0]
analytic_∇²f(x0) = [-1.0 1.5; 1.5 4.0]
analytic_∇²f(x0) - ∇²f(x0) = [0.0 0.0; 0.0 0.0]
```

Newton's Algorithm

Solve $\nabla f(x) = 0$ by Newton's method

Require $f\colon \mathbf{R}^n \rightarrow \mathbf{R}$, $x_0 \in \mathbf{R}^n$ (initial point)

- $k \leftarrow 0$
- continue = true
- **While** continue $\#$ See Section Section 1.3.1
  - $d_k $ solution of $\nabla^2 f(x_k)\, d = -\nabla f(x\_k)$ $\#$ Newton's direction
  - $x_{k+1} \leftarrow x_k + d_k$ # Mise à jour de l'itéré
  - $k \leftarrow k+1$
  - continue = stop_function(∇f_k,x_k,x_{k+1},f_k,f_{k+1},AbsTol,RelTol,ε)
- **EndWhile**

Stop criteria

Stop criteria, Stagnation criteria are more restrictive ($\varepsilon = 0.01$, for example).

Criteria	Formula
$\|\nabla f(x_{k+1})\| = 0$	$\|\nabla f(x_{k+1})\| < \max(\mathrm{RelTol}\|\nabla f(x_0)\|, \mathrm{AbsTol})$
Stagnation of the iterate	$\|x_{k+1} - x_k\| < \varepsilon \max(\mathrm{RelTol}\|x_{k+1}\|, \mathrm{AbsTol})$
Stagnation of the function	$\|f(x_{k+1}) - f(x_k)\| < \varepsilon \max(\mathrm{RelTol}\|f(x_{k+1})\|, \mathrm{AbsTol})$
Maximum number of iteration	$k + 1 = \mathrm{max\_iter}$

Exercice

▼ Code

```julia
using LinearAlgebra
using ForwardDiff

"""
   Solve by Newton's algorithm the optimization problem Min f(x)
   Case where f is a function from R^n to R
"""


function algo_Newton(f,x0::Vector{<:Real};AbsTol= abs(eps()), RelTol = abs(eps(
# to complete

    # flag = 0 if the program stop on the first criteria
    # flag = 2 if the program stop on the second criteria
    # ...
    return xₖ, flag, fₖ, ∇fₖ , k
end
```

```
algo_Newton (generic function with 1 method)
```

▼ Code

```julia
include("src/MyOptims.jl")

A = [1 0 ; 0 9/2]
b = [0,0]; c=0.
f1(x) = 0.5*x'*A*x + b'*x +c
x0 = [1000,-20]
println("Results for Newton on f1 : ", algo_Newton(f1,x0))
println("eigen value of 0.5(A^T+A) = ", 0.5*eigen(A'+A).values)
using Plots;
x=range(-10,stop=10,length=100)
y=range(-10,stop=10,length=100)
f11(x,y) = f1([x,y])
p1 = plot(x,y,f11,st=:contourf)

A = [-1 0 ; 0 3]
f2(x) = 0.5*x'*A*x + b'*x +c
println("Results for Newton on f2 : ", algo_Newton(f2,x0))
println("eigen value of 0.5(A^T+A) = ", 0.5*eigen(A'+A).values)
f21(x,y) = f1([x,y])
p2 = plot(x,y,f21,st=:contourf)
# Rosenbrock function
x=range(-1.5,stop=2,length=100)
y=range(-0.5,stop=3.,length=100)
f3(x) = (1-x[1])^2 + 100*(x[2]-x[1]^2)^2
f31(x,y) = f3([x,y])
p3 = plot(x,y,f31,st=:contourf)
x0 = [1.1,1.2]

println("Results for Newton on f3 : ", algo_Newton(f3,[1.1,1.2]))

println("Results for Newton on f3 : ", algo_Newton(f3,[3.,0.]))
plot(p1,p2,p3)
```

```
Results for Newton on f1 : ([0.0, 0.0], 0, 0.0, [0.0, 0.0], 1)
eigen value of 0.5(A^T+A) = [1.0, 4.5]
Results for Newton on f2 : ([0.0, 0.0], 0, 0.0, [0.0, 0.0], 1)
eigen value of 0.5(A^T+A) = [-1.0, 3.0]
Results for Newton on f3 : ([1.0, 1.0], 0, 0.0, [-0.0, 0.0], 7)
Results for Newton on f3 : ([1.0, 1.0], 0, 0.0, [-0.0, 0.0], 5)
```