Nonlinear Optimization Exercise Session 6

Maximilian Stollmayer

In [ ]:
```
using LinearAlgebra, ForwardDiff
```

41

Implement the local Newton algorithm. Use as input data the starting vector x^0, the parameter for the stopping criterion ε and the parameter for the maximal number of allowed iterations *kmax*. The sequence $x^0, x^1, x^2, \ldots$ containing the iteration history and the number of performed iterations should be returned.

In [ ]:
```
function localNewton(f::Function, x0::Vector, tol::Real, kmax::Integer)

    @assert tol >= 0
    @assert kmax > 0

    grad = xk -> ForwardDiff.gradient(f, xk)
    hess = xk -> ForwardDiff.hessian(f, xk)
    x = [x0]
    ng = Inf

    for k ∈ 1:kmax
        g = grad(x[k])
        ng = norm(g)
        if ng <= tol
            return x, k-1, ng
        end

        try
            d = hess(x[k]) \ g
            push!(x, x[k] - d)
        catch e
            if isa(e, SingularException)
                return x, k-1, ng
            else
                throw(e)
            end
        end
    end

    return x, kmax, ng
end
;
```

The implemented algorithm should be tested for $\varepsilon = 10^{-6}$ and *kmax* = 200, and the following functions and starting values:

In [ ]:
```
ε = 1e-6
kmax = 200
;
```

(a)

Rosenbrock function $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ and $x^0 = (-1.2, 1)^\top$

In [ ]:
```
f_a(x::Vector)::Real = (1-x[1])^2 + 100(x[2] - x[1]^2)^2
x0_a = [-1.2, 1.0]

x, k, ng = localNewton(f_a, x0_a, ε, kmax)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
6 iterations
gradient norm = 8.285705791275365e-9
approximated minimum:
[0.9999999999999999, 0.9999999999814724]
```

(b)

trigonometric function $f(x_1, x_2, x_3, x_4) = \sum_{i=1}^{4} \left(4 - \sum_{j=1}^{4} \cos x_j + i(1 - \cos x_i) - \sin x_i\right)^2$ and $x^0 = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})^\top$

In [ ]:
```
f_b(x::Vector)::Real = sum([(4 - sum(cos.(x)) + i*(1 - cos(x[i])) - sin(x[i]))^2 for i ∈ 1:le
x0_b = [1/4, 1/4, 1/4, 1/4]

x, k, ng = localNewton(f_b, x0_b, ε, kmax)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
16 iterations
gradient norm = 2.8054341478443444e-8
approximated minimum:
[0.1454861843128384, 0.16018148574735072, 0.424955120548663, 0.21684658127434417]
```

(c)

Brown function $f(x_1, x_2) = (x_1 - 10^6)^2 + (x_2 - 2 \cdot 10^6)^2 + (x_1 x_2 - 2)^2$ and $x^0 = (1, 1)^\top$

In [ ]:
```
f_c(x::Vector)::Real = (x[1] - 1e6)^2 + (x[2] - 2e6)^2 + (x[1]*x[2] - 2)^2
x0_c = [1.0, 1.0]

x, k, ng = localNewton(f_c, x0_c, ε, kmax)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
28 iterations
gradient norm = 0.0
approximated minimum:
[158.75270665788787, 79.36690167054873]
```

(d)

Wood function
$f(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10(x_2 + x_4 - 2)^2 + \frac{1}{10}(x_2$
and $x^0 = (-3, -1, -3, -1)^\top$

In [ ]:
```
f_d(x::Vector)::Real = 100(x[2] - x[1]^2)^2 + (1 - x[1])^2 + 90(x[4] - x[3]^2)^2 + (1 - x[3])^
x0_d = [-3.0, -1.0, -3.0, -1.0]

x, k, nd = localNewton(f_d, x0_d, ε, kmax)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
13 iterations
gradient norm = 0.0
approximated minimum:
[-0.9679740412300142, 0.9471391719429616, -0.9695162944613986, 0.951247634684588]
```

(e)

$f(x) = \sqrt{1 + x^2}$, $x^0 = 2$, $x^0 = 1$ and $x^0 = \frac{1}{2}$

In [ ]:
```
f_e(x::Vector)::Real = sqrt(1 + x[1]^2)
x0s_e = ([2.0], [1.0], [1/2])

for x0 ∈ x0s_e
    x, k, ng = localNewton(f_e, x0, ε, kmax)
    println("x0 = ", x0[1])
    println("gradient norm = ", ng)
    println(k, " iterations")
    println("approximated minimum: ", x[end][1])
    println("")
end
```

```
x0 = 2.0
gradient norm = 1.0
3 iterations
approximated minimum: -1.3421772800366214e8

x0 = 1.0
gradient norm = 4.1402909377839803e-10
37 iterations
approximated minimum: -4.1402909377839803e-10

x0 = 0.5
gradient norm = 7.450580596923828e-9
3 iterations
approximated minimum: -7.450580596923828e-9
```

In the above example for $x^0 = 2$ the Newton equations could not be solved due to a singular Hessian and therefore the iteration was stopped.

42

Implement the globalized Newton algorithm. Use as input data the starting vector x^0, the parameter for the stopping criterion ε, the parameter for the maximal number of allowed iterations *kmax*, the parameters for the determination of the Armijo step size σ and β, and the parameters ρ and p. The sequence $x^0, x^1, x^2, \ldots$ containing the iteration history and the number of performed iterations should be returned.

In [ ]:
```
function globalNewton(f::Function, x0::Vector, tol::Real, kmax::Integer, σ::Real, β::Real, ρ::

    @assert tol >= 0
    @assert kmax > 0
    @assert 0 < σ < 0.5
    @assert 0 < β < 1
    @assert ρ > 0
    @assert p > 2

    grad = xk -> ForwardDiff.gradient(f, xk)
    hess = xk -> ForwardDiff.hessian(f, xk)
    x = [x0]
    ng = Inf

    for k ∈ 1:kmax

        g = grad(x[k])
        ng = norm(g)
        if ng <= tol
            return x, k-1, ng
        end

        d = -g
        try
            dn = -hess(x[k]) \ g
            if dot(g, dn) <= ρ * norm(dn)^p
                d = dn
            end
        catch e
            if !isa(e, SingularException)
                throw(e)
            end
        end

        # iterate step size until Armijo condition is met
        σnd = σ * norm(d)
        fx = f(x[k])
        t = 1
        while f(x[end] + t * d) > fx - t * σnd
            t *= β
        end

        push!(x, x[k] + t * d)
    end

    return x, kmax, ng
end
;
```

The implemented algorithm should be tested for $\varepsilon = 10^{-6}$, *kmax* = 200, $\rho = 10^{-8}$, $p = 2.1$, $\sigma = 10^{-4}$ and $\beta = 0.5$, and the following functions and starting values:

In [ ]:
```
ε = 1e-6
kmax = 200
ρ = 1e-8
p = 2.1
σ = 1e-4
β = 0.5
;
```

(a) Rosenbrock function

In [ ]:
```
x, k, ng = globalNewton(f_a, x0_a, ε, kmax, σ, β, ρ, p)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
200 iterations
gradient norm = 0.00011871675903483276
approximated minimum:
[0.9999988896181284, 0.9999975093462696]
```

(b) trigonometric function

In [ ]:
```
x, k, ng = globalNewton(f_b, x0_b, ε, kmax, σ, β, ρ, p)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
200 iterations
gradient norm = 9.296209871724122e-5
approximated minimum:
[0.1455339094301942, 0.16024364624568757, 0.42490842133240075, 0.21707430546894338]
```

(c) Brown function

In [ ]:
```
x, k, ng = globalNewton(f_c, x0_c, ε, kmax, σ, β, ρ, p)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
22 iterations
gradient norm = 9.458442548534918e-10
approximated minimum:
[1.2500000000001564e-6, 1.999999999999375e6]
```

(d) Wood function

In [ ]:
```
x, k, ng = globalNewton(f_d, x0_d, ε, kmax, σ, β, ρ, p)

println(k, " iterations")
println("gradient norm = ", ng)
println("approximated minimum:")
println(x[end])
```

```
200 iterations
gradient norm = 0.021943549977087782
approximated minimum:
[-0.9976844506520253, 1.005434721018345, -0.9388457042511992, 0.8928940340625637]
```

(e)

In [ ]:
```
for x0 ∈ x0s_e
    x, k, ng = globalNewton(f_e, x0, ε, kmax, σ, β, ρ, p)

    println("x0 = ", x0[1])
    println("gradient norm = ", ng)
    println(k, " iterations")
    println("approximated minimum: ", x[end][1])
    println("")
end
```

```
x0 = 2.0
gradient norm = 7.450580596923828e-9
4 iterations
approximated minimum: 7.450580596923828e-9

x0 = 1.0
gradient norm = 1.1102230246251565e-16
1 iterations
approximated minimum: 1.1102230246251565e-16

x0 = 0.5
gradient norm = 7.450580596923828e-9
3 iterations
approximated minimum: -7.450580596923828e-9
```

Comparison

(a) Rosenbrock function

In [ ]:
```
x_l, k_l, ng_l = localNewton(f_a, x0_a, ε, kmax)
x_g, k_g, ng_g = globalNewton(f_a, x0_a, ε, kmax, σ, β, ρ, p)

println("solver | iterations | gradient norm | approximated minimum")
println("local  |          $k_l | $(round(ng_l, sigdigits=9)) | $(x_l[end])")
println("global |        $k_g | $(round(ng_g, digits=11)) | $(x_g[end])")
```

```
solver | iterations | gradient norm | approximated minimum
local  |          6 | 8.28570579e-9 | [0.9999999999999999, 0.9999999999814724]
global |        200 | 0.00011871676 | [0.9999988896181284, 0.9999975093462696]
```

(b) trigonometric function

In [ ]:
```
x_l, k_l, ng_l = localNewton(f_b, x0_b, ε, kmax)
x_g, k_g, ng_g = globalNewton(f_b, x0_b, ε, kmax, σ, β, ρ, p)

println("solver | iterations | gradient norm | approximated minimum")
println("local  |         $k_l | $(round(ng_l, sigdigits=9)) | $(round.(x_l[end], digits=5))")
println("global |        $k_g | $(round(ng_g, sigdigits=9)) | $(round.(x_g[end], digits=5))")
```

```
solver | iterations | gradient norm | approximated minimum
local  |         16 | 2.80543415e-8 | [0.14549, 0.16018, 0.42496, 0.21685]
global |        200 | 9.29620987e-5 | [0.14553, 0.16024, 0.42491, 0.21707]
```

(c) Brown function

In [ ]:
```
x_l, k_l, ng_l = localNewton(f_c, x0_c, ε, kmax)
x_g, k_g, ng_g = globalNewton(f_c, x0_c, ε, kmax, σ, β, ρ, p)

println("solver | iterations | gradient norm | approximated minimum")
println("local  |         $k_l |           $ng_l | $(round.(x_l[end], sigdigits=5))")
println("global |         $k_g |      $(round(ng_g, sigdigits=3)) | $(round.(x_g[end], sigdigi
```

```
solver | iterations | gradient norm | approximated minimum
local  |         28 |           0.0 | [158.75, 79.367]
global |         22 |      9.46e-10 | [1.25e-6, 2.0e6]
```

(d) Wood function

In [ ]:
```
x_l, k_l, ng_l = localNewton(f_d, x0_d, ε, kmax)
x_g, k_g, ng_g = globalNewton(f_d, x0_d, ε, kmax, σ, β, ρ, p)

println("solver | iterations | gradient norm | approximated minimum")
println("local  |         $k_l | $(round(ng_l, sigdigits=9)) | $(round.(x_l[end], digits=5))")
println("global |        $k_g | $(round(ng_g, sigdigits=10)) | $(round.(x_g[end], digits=5))")
```

```
solver | iterations | gradient norm | approximated minimum
local  |         13 | 2.20997015e-7 | [-0.96797, 0.94714, -0.96952, 0.95125]
global |        200 | 0.02194354998 | [-0.99768, 1.00543, -0.93885, 0.89289]
```

(e)

In [ ]:
```
println(" x0 | solver | iterations | gradient norm | approximated minimum")
for x0_e ∈ x0s_e
    x_l, k_l, ng_l = localNewton(f_e, x0_e, ε, kmax);
    x_g, k_g, ng_g = globalNewton(f_e, x0_e, ε, kmax, σ, β, ρ, p);

    println("$(x0_e[1]) | local  |         $k_l |  $(round(ng_l, sigdigits=4)) | $(round.(x_l|
    println("$(x0_e[1]) | global |         $k_g |  $(round(ng_g, sigdigits=4)) | $(round.(x_g|
end
```

```
 x0 | solver | iterations | gradient norm | approximated minimum
2.0 | local  |          3 |  1.0 | -1.342e8
2.0 | global |          4 |  7.451e-9 | 7.451e-9
1.0 | local  |         37 |  4.14e-10 | -4.14e-10
1.0 | global |          1 |  1.11e-16 | 1.11e-16
0.5 | local  |          3 |  7.451e-9 | -7.451e-9
0.5 | global |          3 |  7.451e-9 | -7.451e-9
```

In [ ]: