# Nonlinear Optimization Exercise Session 5

Maximilian Stollmayer

In [ ]:
```julia
using LinearAlgebra
using Plots
```

## 33. Variations of the Gradient Algorithm

Solve the minimization problem

$$\min_{x \in \mathbb{R}} f(x)$$

where

$$f : \mathbb{R}^n \to \mathbb{R},\, f(x) = \begin{cases} \frac{1}{c}\|x\| - \frac{1}{2c^2}, & \text{if } \|x\| \geq \frac{1}{c} \\ \frac{1}{c}\|x\|^2, & \text{otherwise} \end{cases},$$

with the gradient algorithm and the fast gradient algorithm

- by considering different values for the dimension $n \in \{1, 10, 50, 500, 5000\}$ and for the parameter $c > 0$
- by using the Nesterov rule and the Chambolle-Dossal rule for the momentum parameters
- by using different values for the starting point $x^0$

Run the two algorithms with step size $\gamma = \frac{1}{L_{\nabla f}}$ for 200 iterations and display $\left(\|x^k\|\right)_{k=0,...,200}$ and $\left(f(x^k)\right)_{k=0,..,200}$ as functions of the number of iterations $k$.

In [ ]:
```julia
function gradientAlgorithmArmijo(
    f::Function,
    g::Function,
    x0::Vector,
    β::Real,
    σ::Real,
    k::Integer
)

    @assert k > 0 "Number of iterations is not positive, got $k."

    x = [x0]
    d = -g(x[1])

    for _ ∈ 1:k
        fx = f(x[end])
        σnorm = σ * norm(d)
        k = 0
        t = 1

        # iterate exponent k until Armijo condition is met
        while f(x[end] + t * d) > fx - t * σnorm
            k = k + 1
            t = β^k
        end

        push!(x, x[end] + t * d)
        d = -g(x[end])
    end

    return x
end
;
```

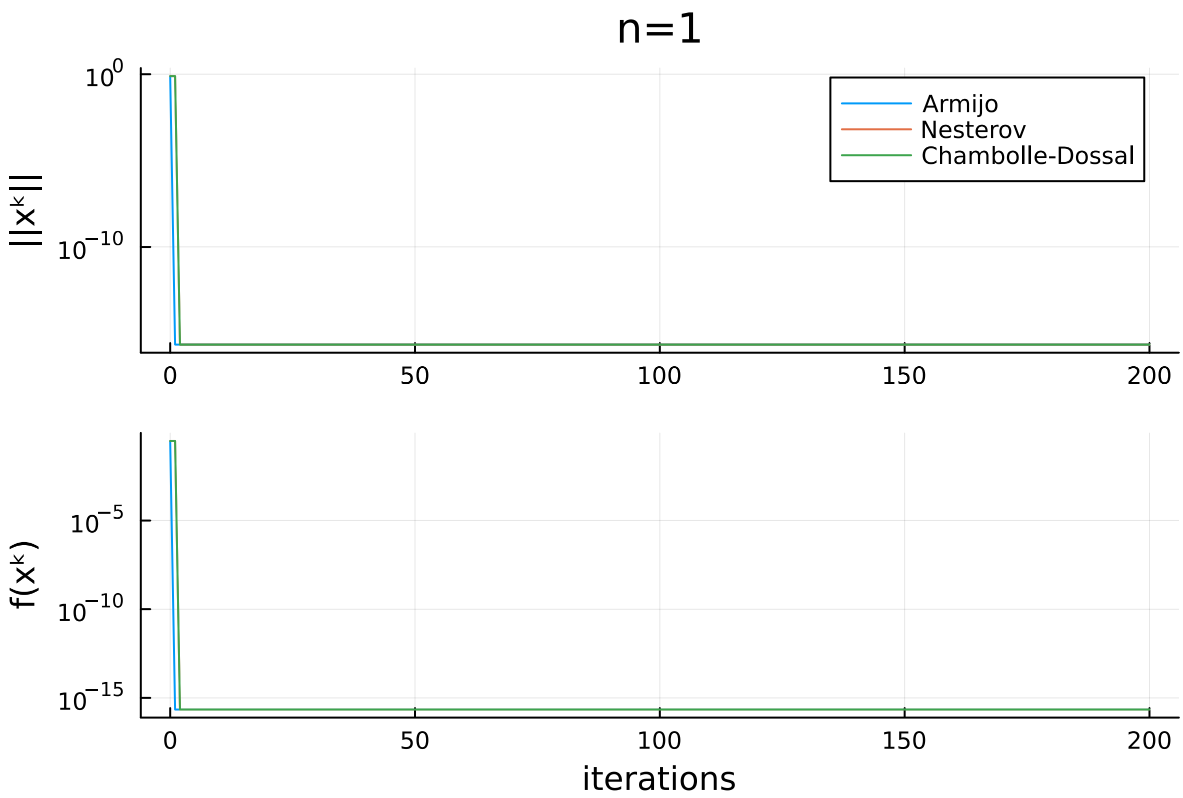
In [ ]:
```julia
function gradientAlgorithmNesterov(
    f::Function,
    g::Function,
    x0::Vector,
    γ::Real,
    k::Integer
)

    @assert k > 0 "Number of iterations is not positive, got $k."

    x = [x0, x0]
    t_old = 1
    d = -g(x[1])

    for _ ∈ 2:k
        t_new = (1 + sqrt(4 * t_old^2 + 1)) / 2
        y = x[end] + (t_old - 1) / t_new * (x[end] - x[end-1])
        push!(x, y - γ * g(y))
        d = -g(x[end])
        t_old = t_new
    end

    return x
end
;
```

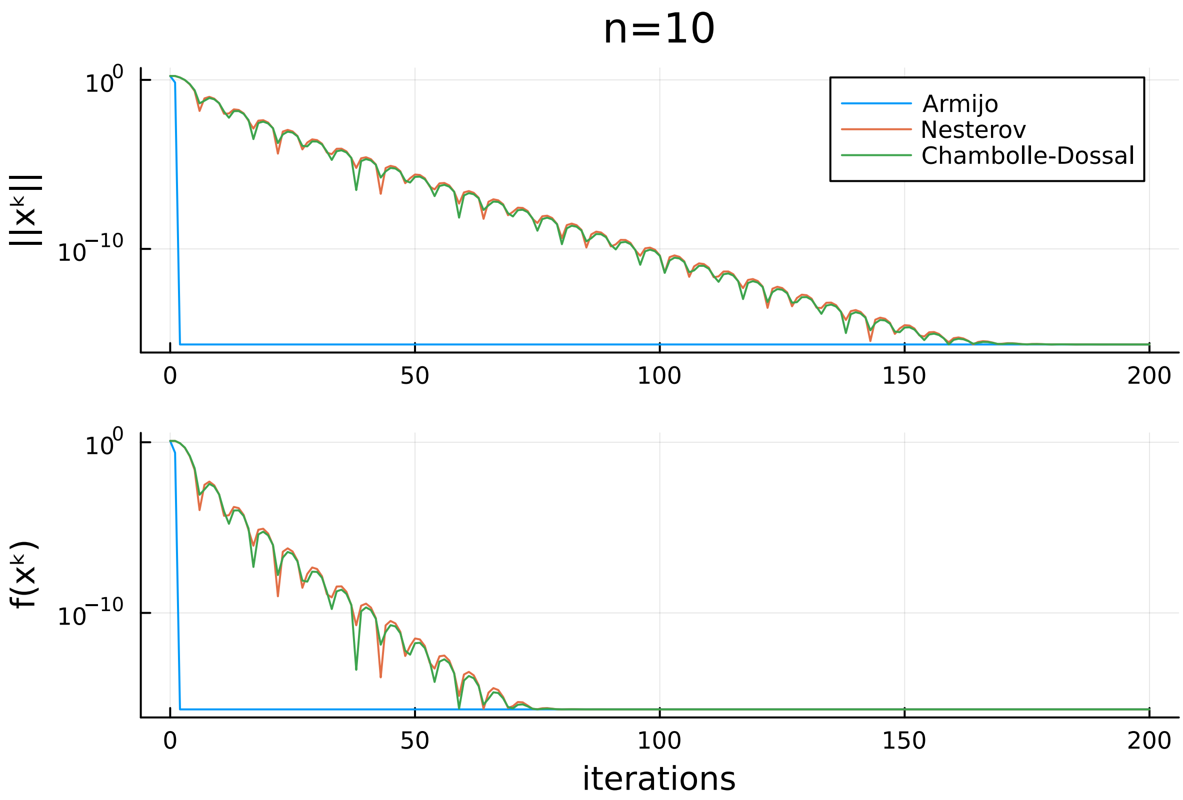
In [ ]:
```julia
function gradientAlgorithmChambolleDossal(
    f::Function,
    g::Function,
    x0::Vector,
    γ::Real,
    α::Real,
    k::Integer
)

    @assert k > 0 "Number of iterations is not positive, got $k."

    x = [x0, x0]
    t_old = 1
    d = -g(x[1])

    for _ ∈ 2:k
        t_new = (length(x) + α - 2) / (α - 1)
        y = x[end] + (t_old - 1) / t_new * (x[end] - x[end-1])
        push!(x, y - γ * g(y))
        d = -g(x[end])
        t_old = t_new
    end

    return x
end
;
```

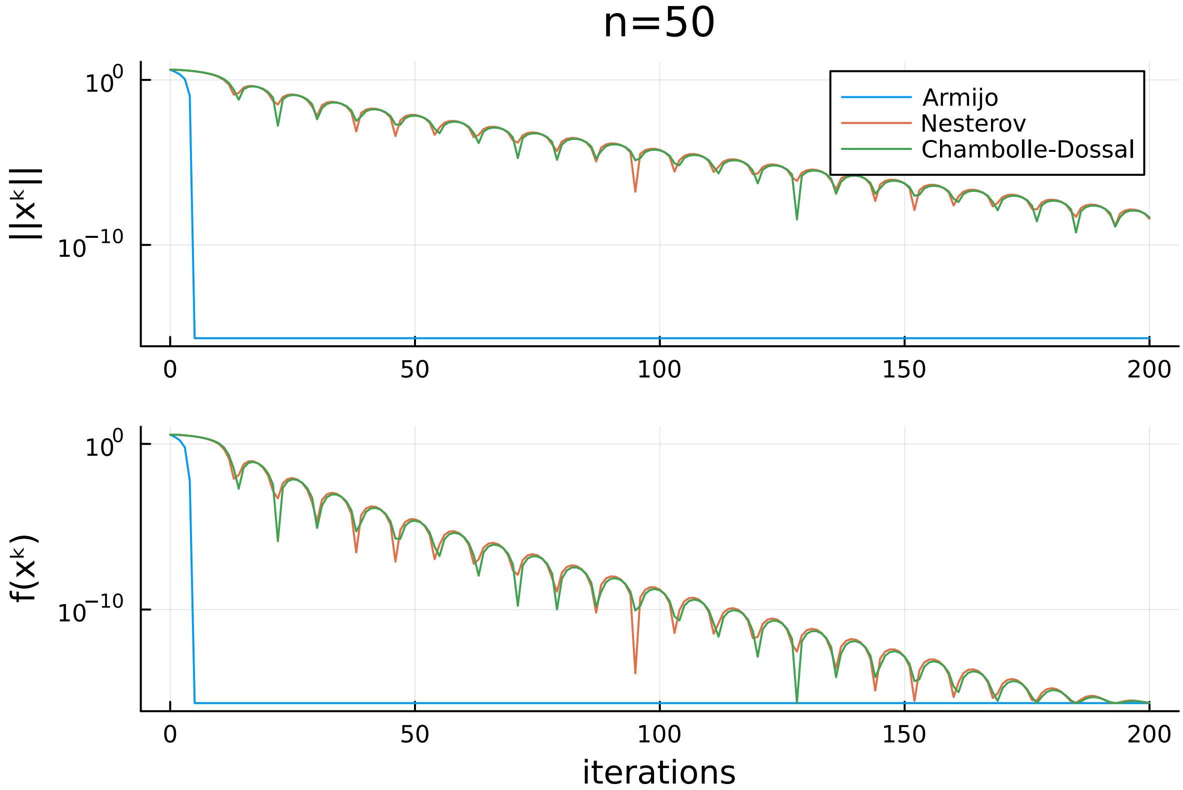
In [ ]:
```julia
function plotIterations(ns::Vector{Int}, c::Real, β::Real, σ::Real, α::Real, k::Intege
    function f(x::AbstractVector)
        nx = norm(x)
        return nx >= 1/c ? nx/c - 1/(2*c^2) : nx^2 / 2
    end

    function g(x::AbstractVector)
        nx = norm(x)
        nx >= 1/c ? x / nx / c : x
    end

    for n ∈ ns
        x = 2*rand(n) .- 1
        γ = 1 / sqrt(n) # 1 / Lipschitz constant
        Armijo = gradientAlgorithmArmijo(f, g, x, β, σ, k)
        Nesterov = gradientAlgorithmNesterov(f, g, x, γ, k)
        ChambolleDossal = gradientAlgorithmChambolleDossal(f, g, x, γ, α, k)
        norms = plot(
            0:k,
            [norm.(Armijo).+eps(), norm.(Nesterov).+eps(), norm.(ChambolleDossal).+eps
            yscale=:log10,
            title="n=$n",
            ylabel="||xᵏ||",
            label=["Armijo" "Nesterov" "Chambolle-Dossal"],
        );
        fs = plot(
            0:k,
            [f.(Armijo).+eps(), f.(Nesterov).+eps(), f.(ChambolleDossal).+eps()],
            yscale=:log10,
            xlabel="iterations",
            ylabel="f(xᵏ)",
            legend=false
        );
        display(plot(norms, fs, layout=(2,1)))
    end
end
;
```

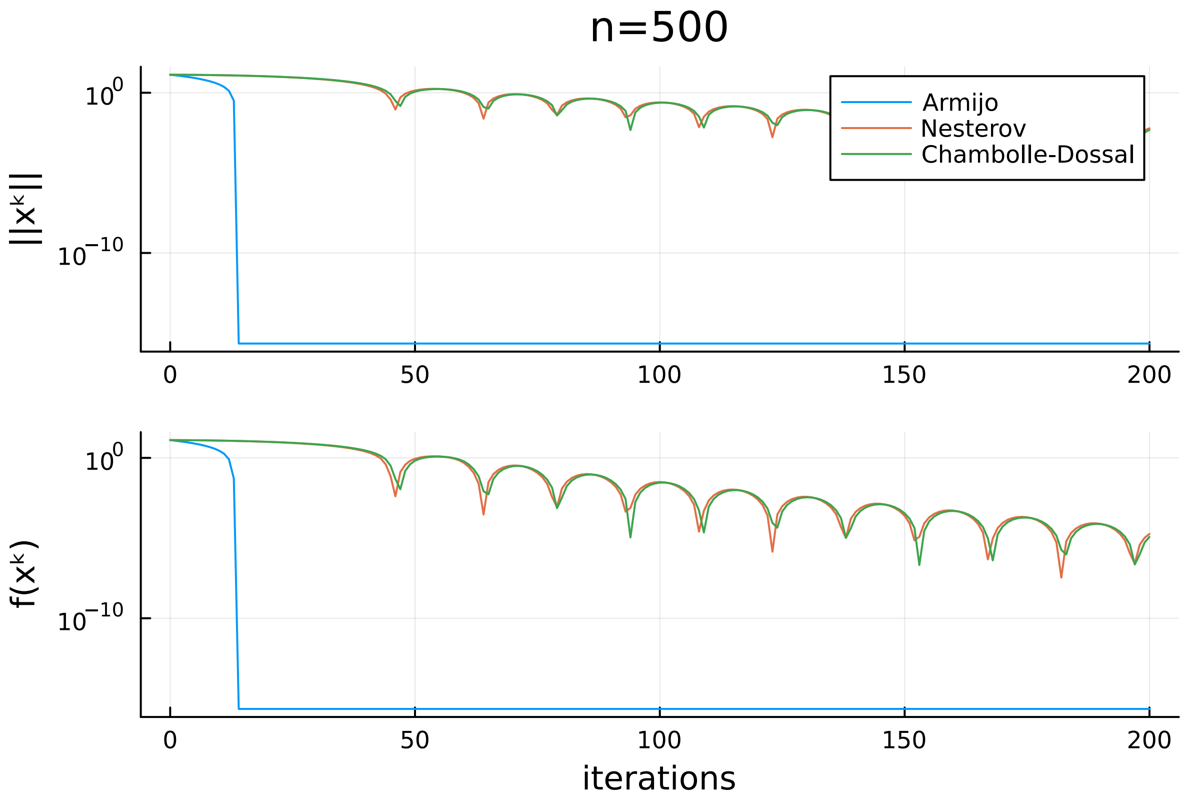
In [ ]:
```
ns = [1, 10, 50, 500, 5000]
β = 0.5  # parameter of Armijo rule
σ = 0.01 # parameter of Armijo rule
α = 3.01 # parameter of Chambolle-Dossal rule
k = 200  # number of iterations
;
```

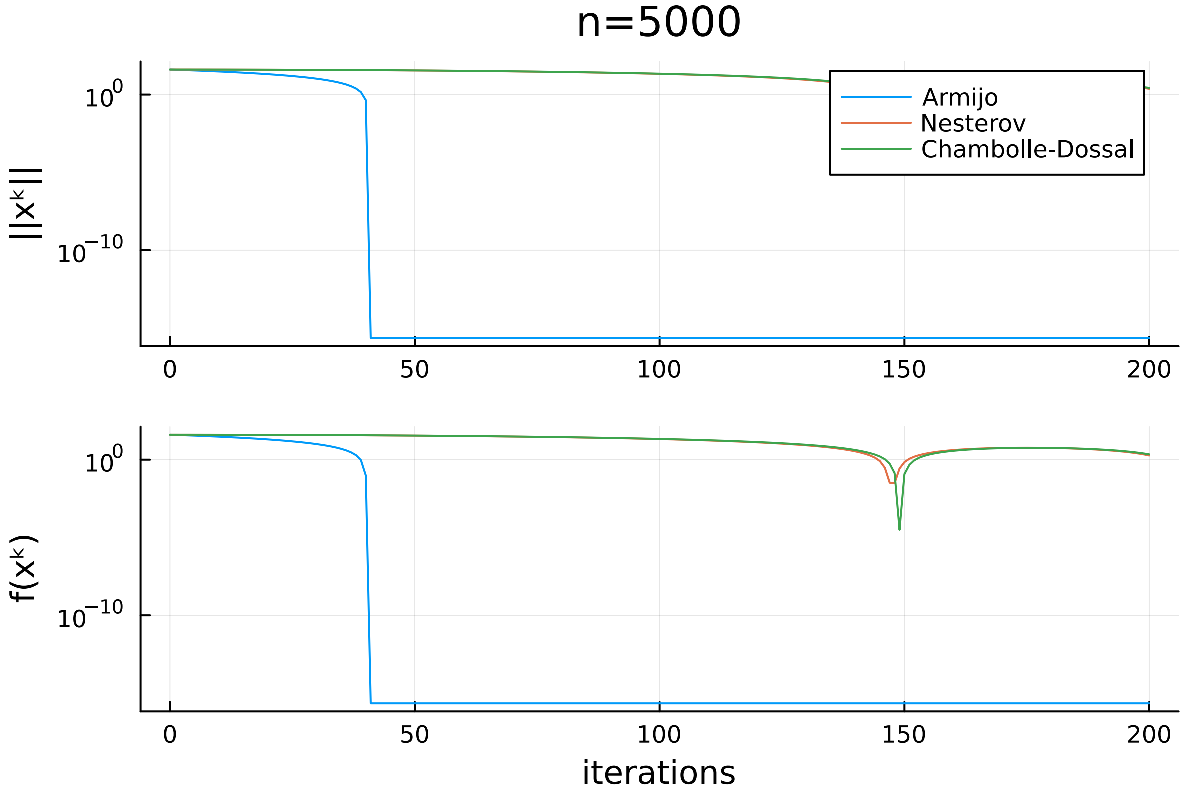
In [ ]:
```
c = 1
plotIterations(ns, c, β, σ, α, k)
```



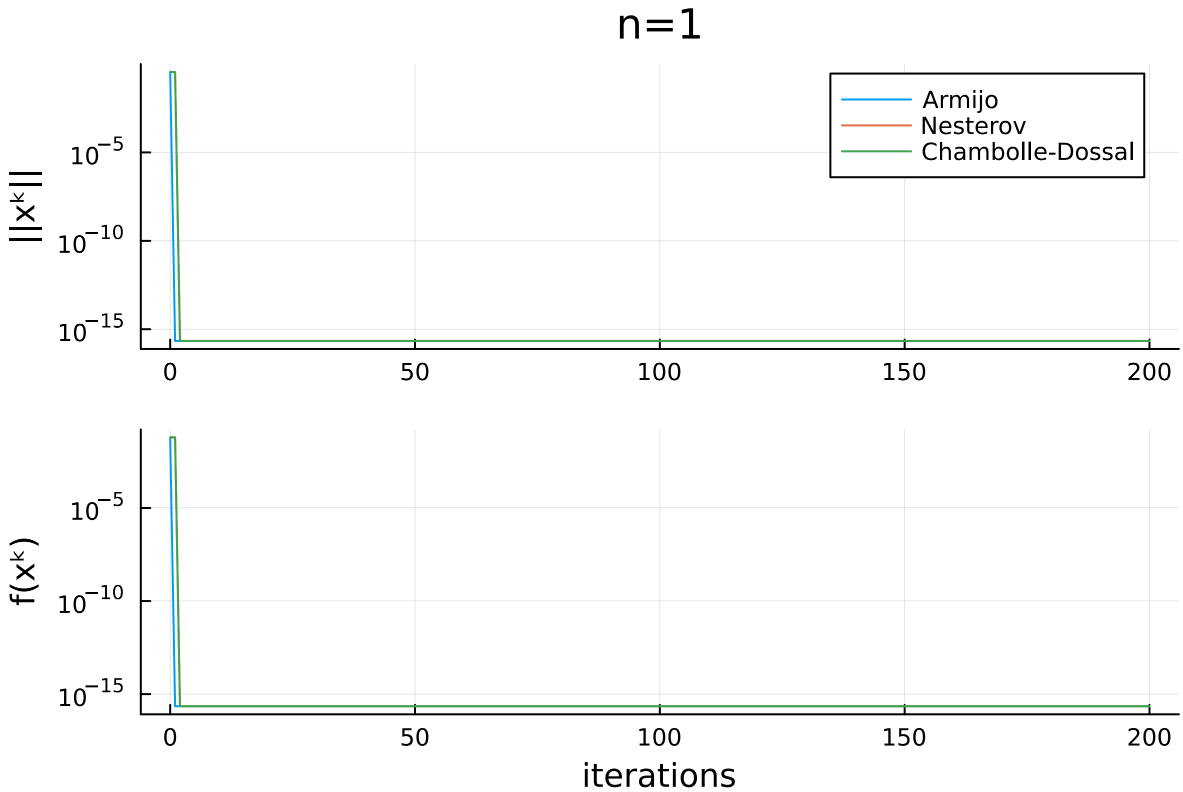



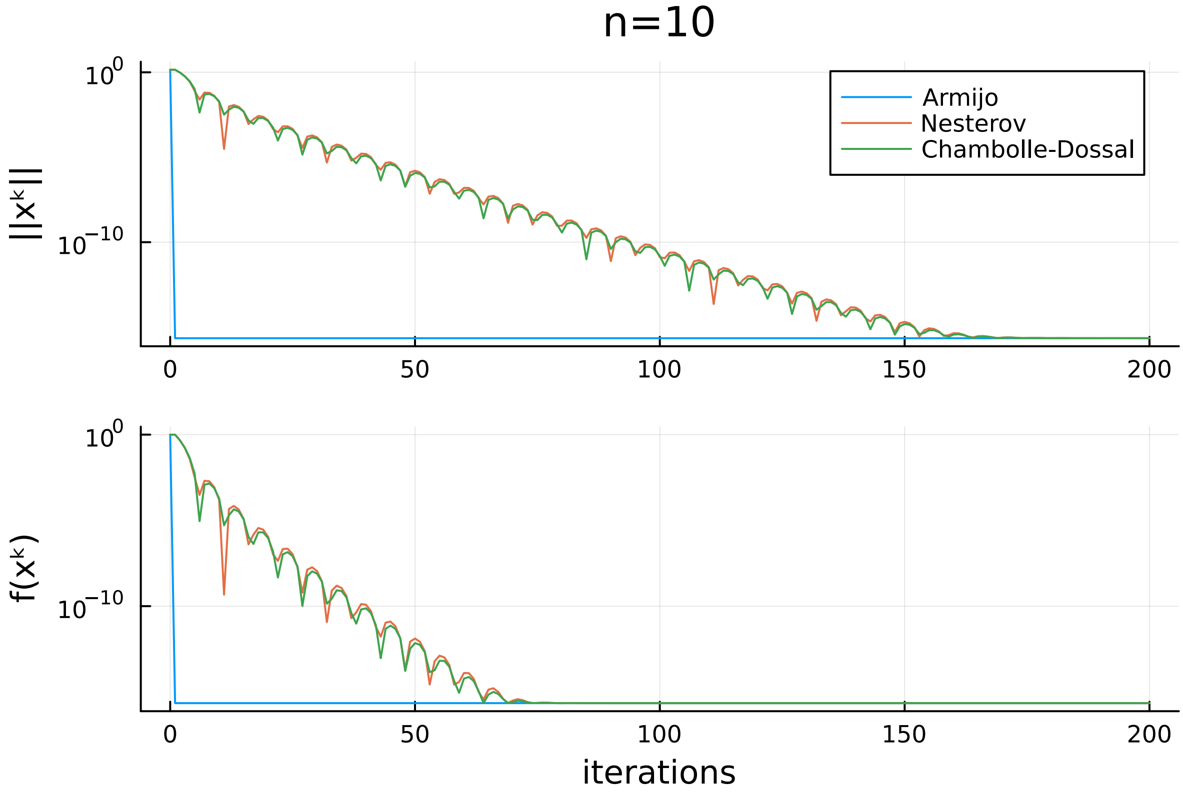

n=50
Armijo
Nesterov
Chambolle-Dossal
$||x^k||$
$f(x^k)$
$10^0$
$10^{-10}$
0
50
100
150
200
iterations

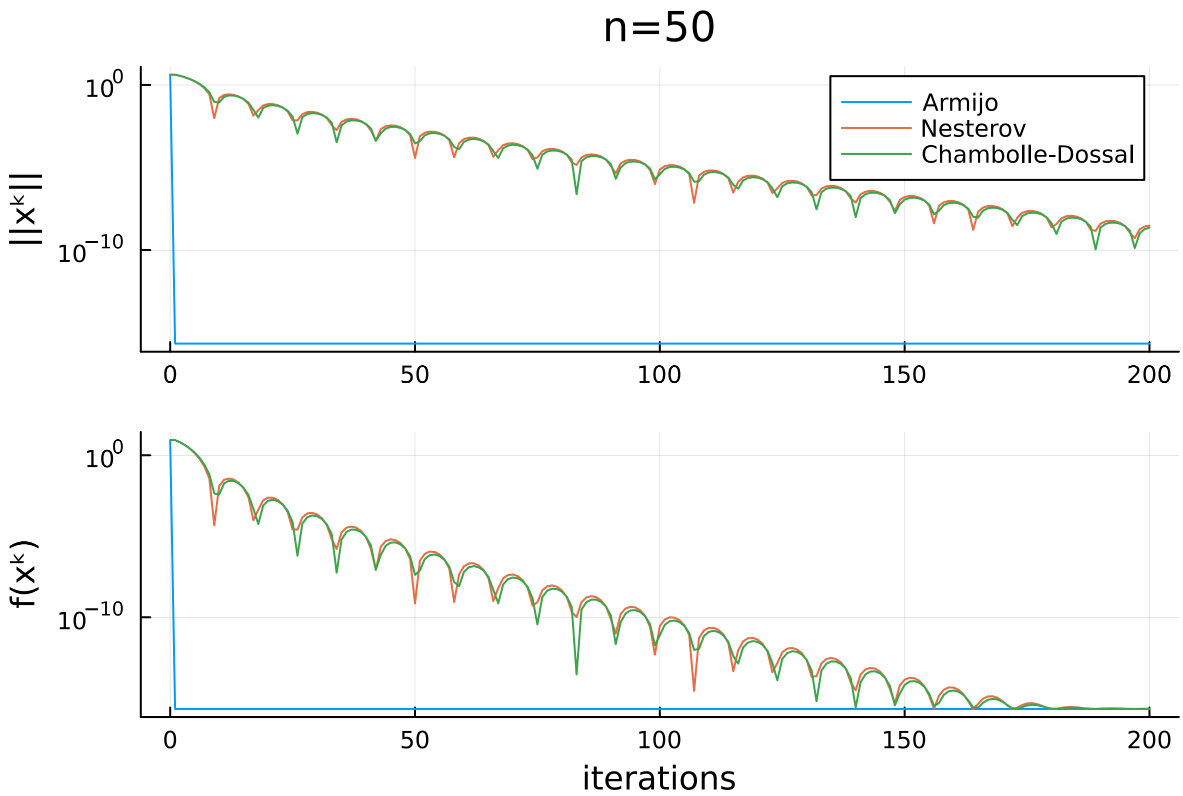

n=500
Armijo
Nesterov
Chambolle-Dossal
$||x^k||$
$f(x^k)$
$10^0$
$10^{-10}$
0
50
100
150
200
iterations

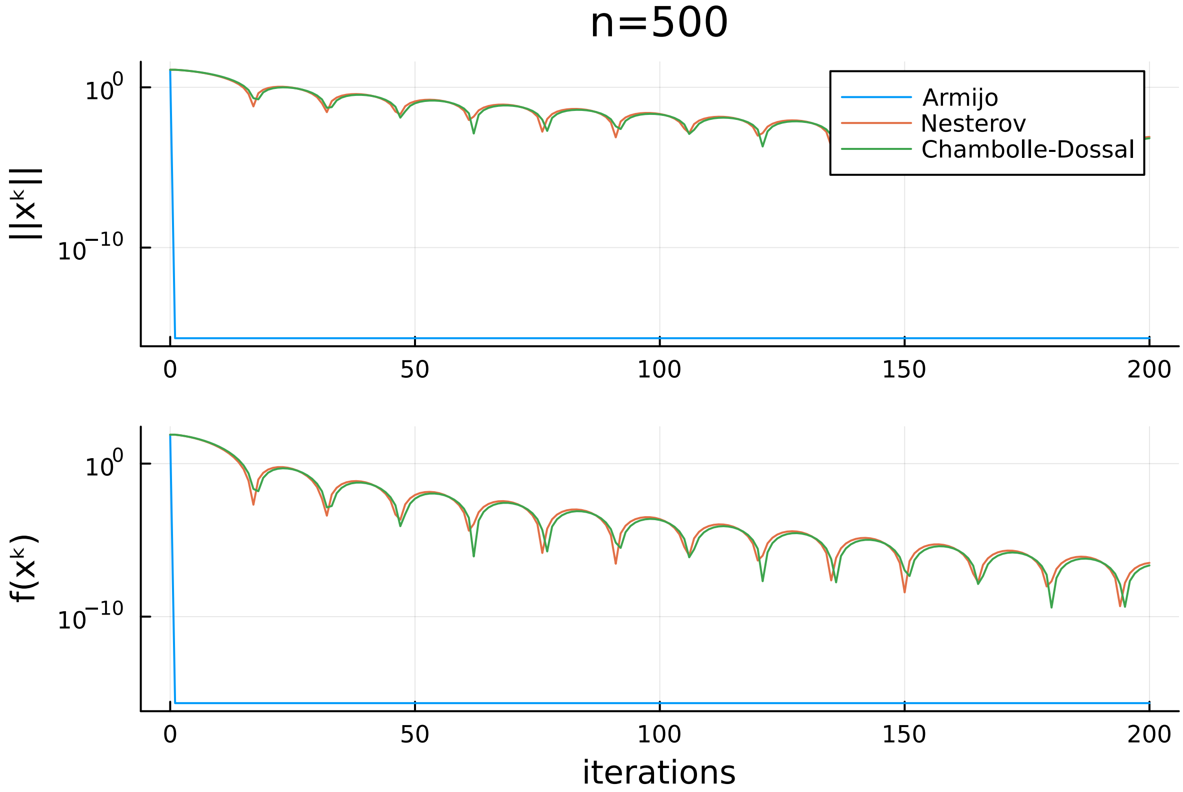


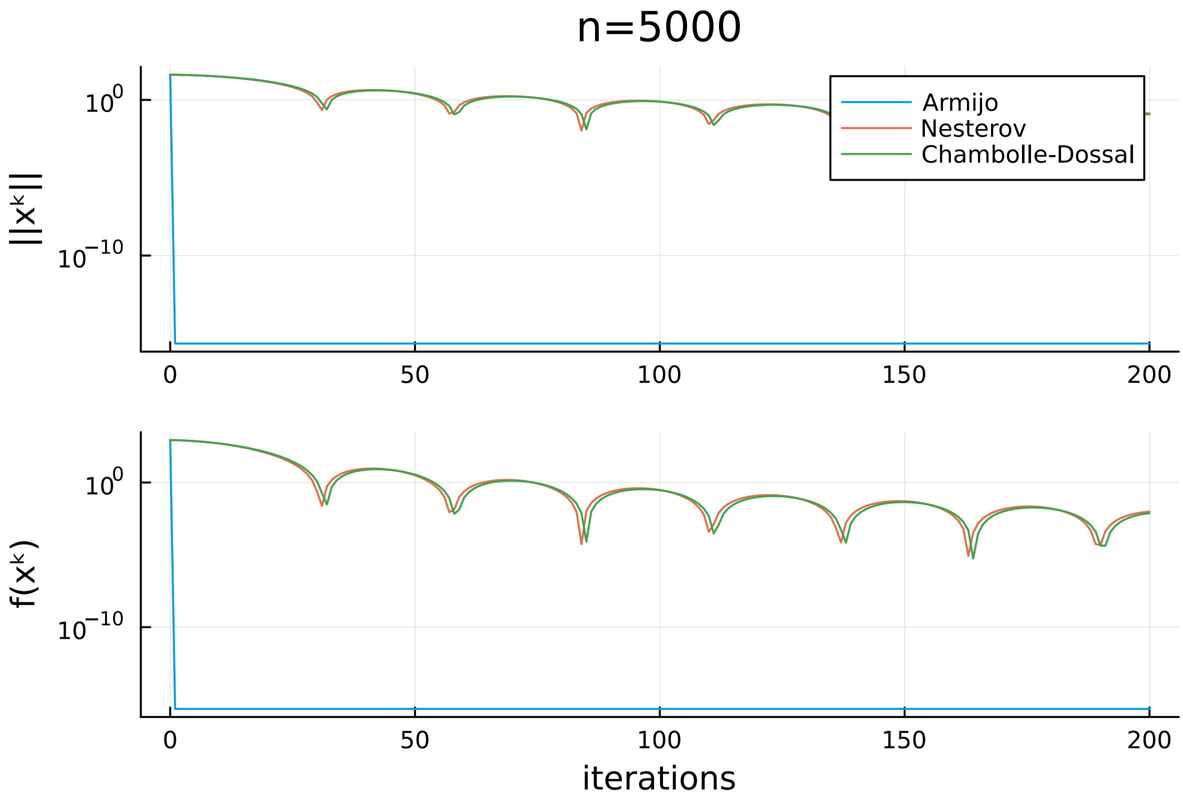

In [ ]:
```
c = 0.01
plotIterations(ns, c, β, σ, α, k)
```



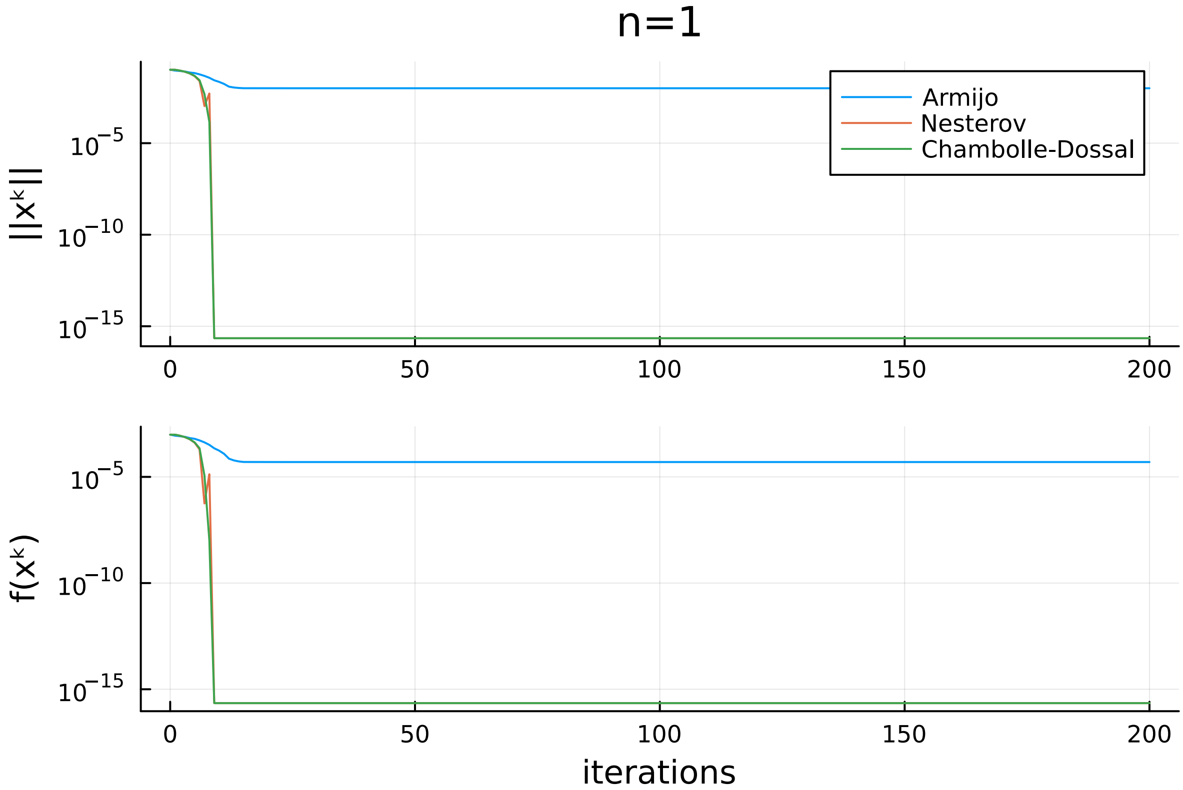

n=10
Armijo
Nesterov
Chambolle-Dossal
||x^k||
f(x^k)
10^0
10^−10
0
50
100
150
200
iterations

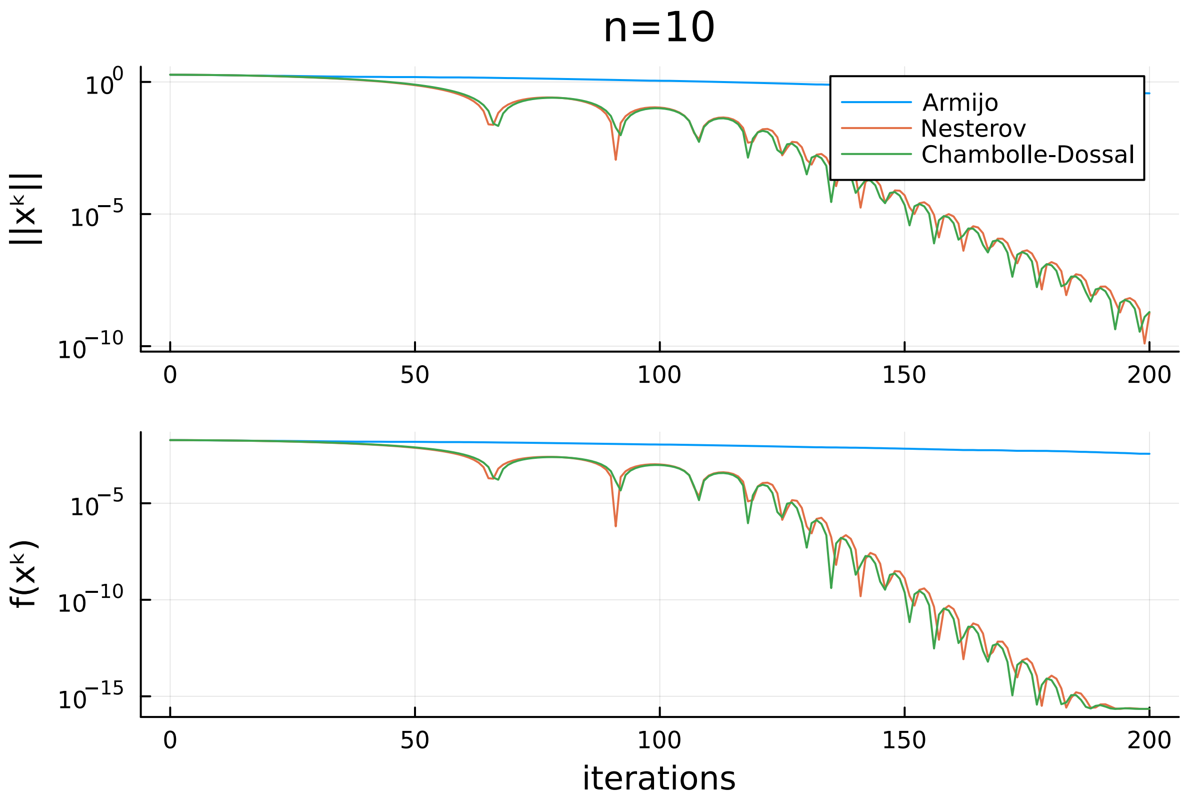

n=50
Armijo
Nesterov
Chambolle-Dossal
||x^k||
f(x^k)
10^0
10^−10
0
50
100
150
200
iterations

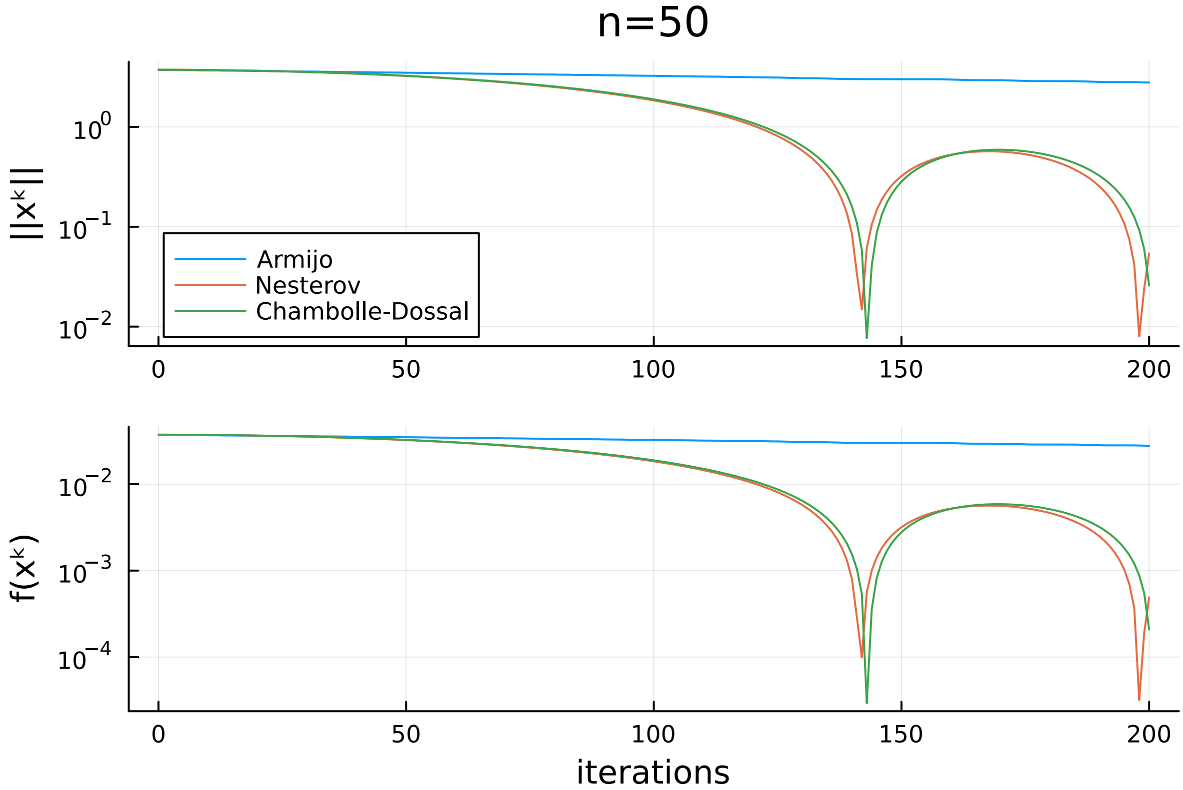


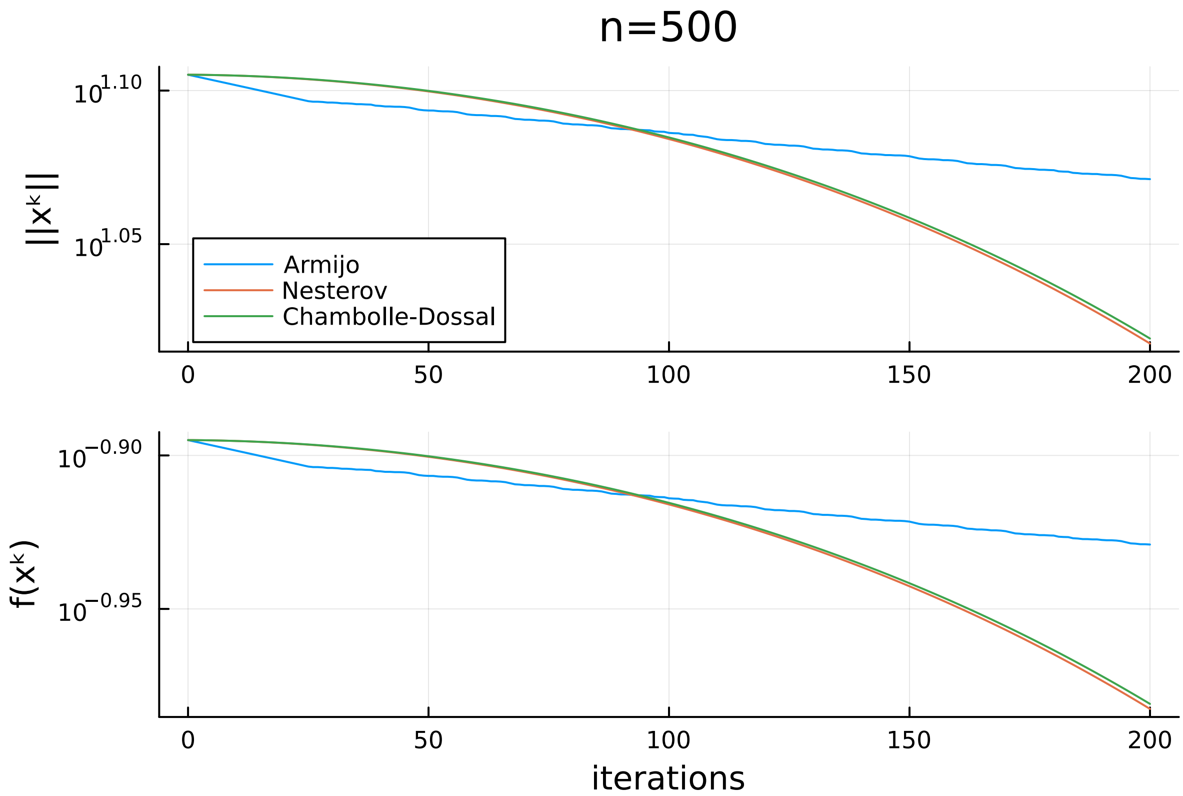



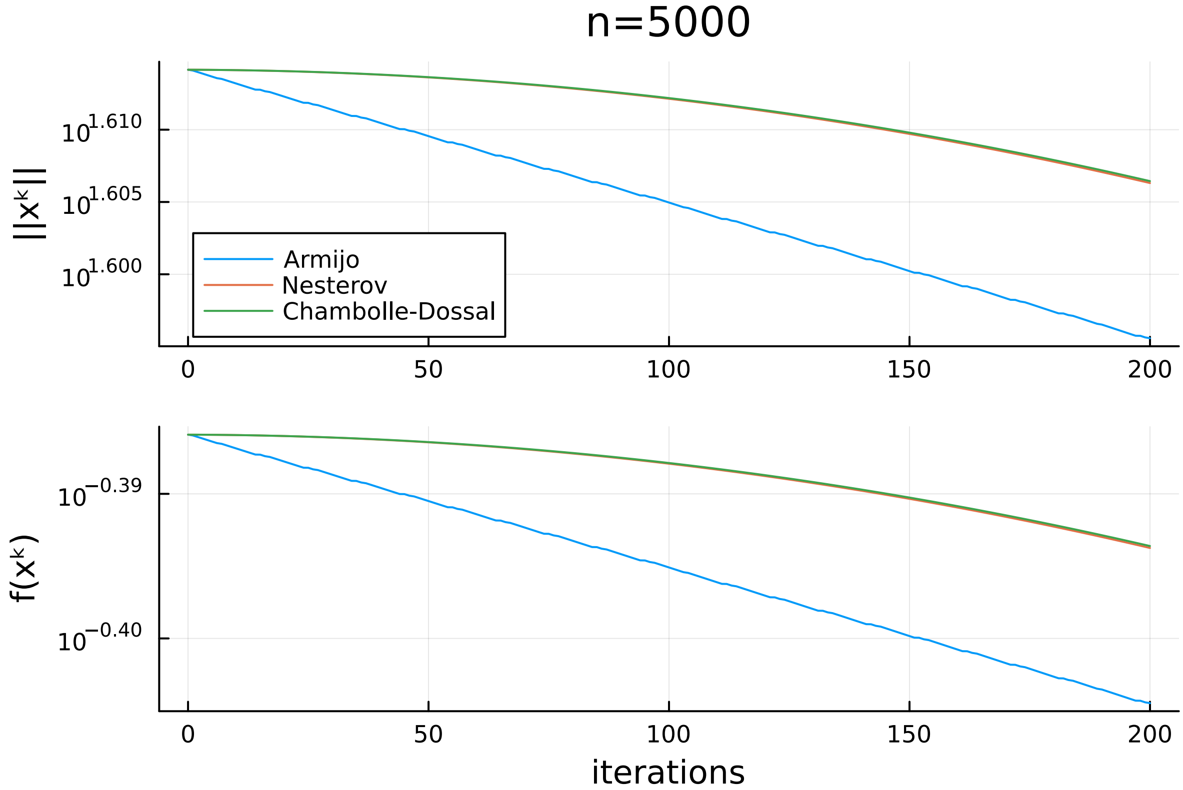

In [ ]:
```
c = 100
plotIterations(ns, c, β, σ, α, k)
```


n=1
Armijo
Nesterov
Chambolle-Dossal
||x^k||
f(x^k)
10^-5
10^-10
10^-15
0
50
100
150
200
iterations


n=10
Armijo
Nesterov
Chambolle-Dossal
||x^k||
f(x^k)
10^0
10^-5
10^-10
10^-15
0
50
100
150
200
iterations


n=50
||x^k||
f(x^k)
Armijo
Nesterov
Chambolle-Dossal
iterations
0
50
100
150
200
10^0
10^-1
10^-2
10^-3
10^-4


n=500
||x^k||
f(x^k)
Armijo
Nesterov
Chambolle-Dossal
iterations
0
50
100
150
200
10^1.10
10^1.05
10^-0.90
10^-0.95




## 34. Implementation of CG Algorithm

Implement the CG algorithm for linear systems (Algorithm 8.2). Use as input data the symmetric and positive definite matrix $A$, the vector $b$, the starting vector $x^0$ and the parameter for the stopping criterion $\varepsilon$. The solution $x^*$ and the number of performed iterations should be returned.

In [ ]:
```julia
function CG(A::AbstractMatrix, b::AbstractVector, x0::AbstractVector, tol::Real)

    n = size(A, 1)

    @assert size(A, 2) == n "Matrix has to be square, got $n by $(size(A, 2))."
    @assert size(b, 1) == n "System vector has to be of size $n, got $(size(b, 1))."
    @assert size(x0, 1) == n "Starting vector has to be of size $n, got $(size(x0, 1)
    @assert tol >= 0 "Tolerance has to be greater than zero, got $tol."

    x_old = x0
    g_old = A * x_old - b
    norm_old = norm(g_old)^2
    d_old = -g_old
    tol = tol^2
    n_iterations = 0

    while norm_old > tol && n_iterations < n
        # calculate new iterates
        Ad = A * d_old
        t = norm_old / dot(d_old, Ad)
        x_new = x_old + t * d_old
        g_new = g_old + t * Ad
        norm_new = norm(g_new)^2
        β = norm_new / norm_old
        d_new = -g_new + β * d_old

        # update old variables
        x_old = x_new
        g_old = g_new
        norm_old = norm_new
        d_old = d_new
        n_iterations += 1
    end

    return x_old, n_iterations
end
;
```

Test the algorithm on the following optimization problems and input data values:

(a) Minimize the function

$$f(x_1, x_2) = 2x_1^2 + x_2^2 - 4x_1 - 2x_2 + 3$$

over $\mathbb{R}^2$, for $x^0 = (5, -5)^\top$ and $\varepsilon = 10^{-3}$.

In [ ]:
```julia
A = [4 0; 0 2]
b = [4; 2]
x0 = [5; -5]
tol = 1e-3
;
```

In [ ]:
```julia
x, n_iterations = CG(A, b, x0, tol)
```

```
([1.0, 1.0000000000000004], 2)
```

In [ ]:
```julia
# optimality condition
all(isapprox.(A * x - b, 0, atol=1e-8))
```

```
true
```

In [ ]:
```julia
# minimum value
dot(x, A, x) / 2 - dot(b, x) + 3
```

```
0.0
```

(b) Minimize the function

$$f(x_1, x_2, x_3) = x_1^2 + 0.3x_1x_2 + 0.975x_2^2 + 0.01x_1x_3 + x_3^2 + 3x_1 - 4x_2 + x_3$$

over $\mathbb{R}^3$, for $x^0 = (0, 0, 0)^\top$ and $\varepsilon = 10^{-8}$.

In [ ]:
```
A = [2 0.3 0.01; 0.3 0.975*2 0; 0.01 0 2]
b = [-3; 4; -1]
x0 = [0; 0; 0]
tol = 1e-8
;
```

In [ ]:
```
x, n_iterations = CG(A, b, x0, tol)
```

```
([-1.8478819339864998, 2.3355715795876666, -0.4907605903300675], 3)
```

In [ ]:
```
# optimality condition
all(isapprox.(A * x - b, 0, atol=1e-8))
```

```
true
```

In [ ]:
```
# minimum value
dot(x, A, x) / 2 - dot(b, x)
```

```
-4.688346355320117
```

In [ ]:
```
```