C_Assignment_2 Report

Assignment 2: projected gradient method

Consider the following problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & 0 \leq x_i \leq 5 \qquad \forall i \end{aligned}$$

where

$$f(x) = \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n-1} x_i x_{i+1}$$

Use your own implementation of the projected conjugate gradient method to solve the problem with $n = 10^4$ and $n = 10^6$, both using exact derivatives and using finite differences to approximate the gradient. Compare the behavior of the two implementations, using the following values for the increment h:

$$h = 10^{-k}\hat{x}, \qquad k = 2, 4, 6, 8, 10, 12, 14$$

where $\hat{x}$ is the point at which the derivatives have to be approximated. The comparison should be made, for example, in terms of number of iterations and computing time.

Write a report summarizing the results with tables and/or figures, commenting the results obtained.

Part 1: Code

The problem has been solved using **MATLAB** software. The code is composed by the following scripts:

- **Script: `finDiff.m`**

```
function[fk] = finDiff(f, x, h, type)

fk = zeros(size(x));
n = length(x);

switch type
    case 'fw'
        for i=1:n
            fk(i) = (f([x(1:i-1); x(i) + h(i); x(i+1:end)]) - f(x))/h(i);
        end
    case 'c'
        for i=1:n
            fk(i) = (f([x(1:i-1); x(i) + h(i); x(i+1:end)]) - f([x(1:i-1);...
                x(i) - h(i); x(i+1:end)]))/h(i);
        end
    case 'bw'
        for i=1:n
            fk(i) = (f(x) - f([x(1:i-1); x(i) - h(i); x(i+1:end)]))/h(i);
        end
end

end
```

- **Script: constraints_projection.m**

```
%function to project the points into the constrains (a box in this case)
function [x_projected] = constraints_projection(x, mins, maxs)

x_projected = max(min(x, maxs), mins);

end
```

- **Script: projected_gradient.m**

```
function [fk, k, btiters] = ...
    projected_gradient(x0, f, ...
    gradf, kmax, tolgrad, tolx, projectionf, ...
    gamma, alpha0, c1, rho, lsmax, t, type)

xk = x0;
fk = f(xk);
k = 0;

deltaxknorm = tolx + 1;
btiters = [];
flag = 0;

% fin diff
if isempty(gradf)
        disp("Approximation of g")
        h = (10 ^ (-t) .* xk);
        gradf = @(x) finDiff(f, x, h, type);
        flag = 1;
end

pkVal = gradf(xk);

gradnorm = norm(pkVal);

while k < kmax && gradnorm >= tolgrad && deltaxknorm >= tolx
    % selection of the descent direction
    pk = - pkVal;
    % Reset of the value of alpha
    alpha = alpha0;

    xbark = xk + gamma * pk;

    % application of the projectionf function handle.
    xbark_projected = projectionf(xbark);

    % "modified" descent direction
    deltax_projection = (xbark_projected - xk);

    xnew = xk + alpha * deltax_projection;

    fnew = f(xnew);
    armijo_c1 = c1 * pkVal' * deltax_projection;
    farmijo = fk + armijo_c1 * alpha;

    i = 0;
    % Backtracking Iterations (2nd condition is the armijo condition not
    % satisfied) this method is the second for the steplength
    while i <= lsmax && fnew > farmijo
        % Reduce the value of alpha
        alpha = alpha * rho;
        % Computation of the new x_(k+1) and its value for f
        xnew = xk + alpha * deltax_projection;
        fnew = f(xnew);
        farmijo = fk + armijo_c1 * alpha;

        i = i + 1;
    end
```

```
    % updating
    deltaxknorm = norm(xnew - xk);
    btiters = [btiters; i];
    xk = xnew;
    fk = f(xk);
    gradnorm = norm(pkVal);
    k = k + 1;

    % updating gradient at each iteration because h changes with xk
    if flag == 1
        h = (10 ^ (-t) * xk);
        gradf = @(x) finDiff(f, x, h, type);
    end
    pkVal = gradf(xk);
end

end
```

- **Main script `test.m`**

```
clear
clc
close all

% Dimension of x
n = 100;

f = @(x) sum(x(:).^2) - sum(x(1:n-1).*x(2:n));

% Constraints
mins = zeros(n,1);
maxs = ones(n,1) * 5;

% check that x0 is in the box
x0 = rand(n,1)*5;

% build the system
kmax = 10000;
tolgrad = 1e-3;
tolx = 1e-6;
projectionf = @(x) constraints_projection(x, mins, maxs);
gamma = 0.5;
alpha0 = 1;
rho = 0.8;
lsmax = 25;
c1 = 1e-4;

for c = 1:4

    if c == 1
        disp('Exact derivates computed')
        gradf = @(x) [(2 * x(1) - x(2));2 * x(2:n-1) - x(1:n-2) - x(3:n), ...
         ;2 * x(n) - x(n-1)];
        tic
        [fk, k] = ...
            projected_gradient(x0, f, ...
            gradf, kmax, tolgrad, tolx, projectionf, ...
            gamma, alpha0, c1, rho, lsmax, 0, '')
        time = toc
        f_save = fk;
        time_save = time;
        iters_save = k;

    elseif c == 2
        figure
        disp('Finite differences case fw')
        valk = [];
        fval = [];
        times = [];
        iters = [];
```

```
    i = 0;
    % variable t as k
    for t = 2:2:14
        disp('Value of k : ')
        disp(t)
        gradf = [];
        tic
        [fk, k] = ...
            projected_gradient(x0, f, ...
            gradf, kmax, tolgrad, tolx, projectionf, ...
            gamma, alpha0, c1, rho, lsmax, t, 'fw')
        elapsed_time = toc
        i = i + 1;
        valk(i) = t;
        fval(i) = fk;
        times(i) = elapsed_time;
        iters(i) = k;
        plot(t,f_save, 'b*')
        hold on
        plot([t t], [f_save fk], '--');
    end
    plot(valk,fval,'-om');
    title('Value of the minimum with respect to k, case forward')
    xlabel('K');
    ylabel('Minima');
    saveas(gcf, 'fw1.png')
    figure
    plot(valk, times, '-om')
    hold on
    for t = 2:2:14
        plot(t,time_save, 'b*')
        hold on
        plot([t t], [time_save times(t/2)], '--');
    end
    title('Elapsed time with respect to k, case forward')
    xlabel('K');
    ylabel('Time');
    saveas(gcf, 'fw2.png')
    figure
    plot(valk, iters, '-om')
    hold on
    for t = 2:2:14
        plot(t,iters_save, 'b*')
        hold on
        plot([t t], [iters_save iters(t/2)], '--');
    end
    title('Iterations with respect to k, case forward')
    xlabel('K');
    ylabel('Iterations');
    saveas(gcf, 'fw3.png')
elseif c == 3
    figure
    valk = [];
    fval = [];
    times = [];
    iters = [];
    i = 0;
    disp('Finite difference centred case')
    for t = 2:2:14
        disp('Value of k : ' )
        disp(t)
        gradf = [];
        tic
        [fk, k] = ...
            projected_gradient(x0, f, ...
            gradf, kmax, tolgrad, tolx, projectionf, ...
            gamma, alpha0, c1, rho, lsmax, t, 'c')
        elapsed_time = toc
        i = i + 1;
        valk(i) = t;
```

```
        fval(i) = fk;
        times(i) = elapsed_time;
        iters(i) = k;
        plot(t,f_save, 'r*')
        hold on
        plot([t t], [f_save fk], '--');
    end
    plot(valk,fval,'-ob');
    title('Value of the minimum with respect to k, case centered')
    xlabel('K');
    ylabel('Minima');
    saveas(gcf, 'c1.png')
    figure
    plot(valk, times, '-ob')
    hold on
    for t = 2:2:14
        plot(t,time_save, 'b*')
        hold on
        plot([t t], [time_save times(t/2)], '--');
    end
    title('Elapsed time with respect to k, case centered')
    xlabel('K');
    ylabel('Time');
    saveas(gcf, 'c2.png')
    figure
    plot(valk, iters, '-ob')
    hold on
    for t = 2:2:14
        plot(t,iters_save, 'b*')
        hold on
        plot([t t], [iters_save iters(t/2)], '--');
    end
    title('Iterations with respect to k, case centered')
    xlabel('K');
    ylabel('Iterations');
    saveas(gcf, 'c3.png')
elseif c == 4
    figure
    disp('Finite difference case bw')
    valk = [];
    fval = [];
    times = [];
    iters = [];
    i = 0;
    for t = 2:2:14
        disp('Value of k : ')
        disp(t)
        gradf = [];
        tic
        [fk, k] = ...
            projected_gradient(x0, f, ...
            gradf, kmax, tolgrad, tolx, projectionf, ...
            gamma, alpha0, c1, rho, lsmax, t, 'bw')
        elapsed_time = toc
        i = i + 1;
        valk(i) = t;
        fval(i) = fk;
        times(i) = elapsed_time;
        iters(i) = k;
        plot(t,f_save, 'r*')
        hold on
        plot([t t], [f_save fk], '--');
    end
    plot(valk,fval,'-ok');
    title('Value of the minimum with respect to k, case backward')
    xlabel('K');
    ylabel('Minima');
    saveas(gcf, 'bw1.png')
    figure
    plot(valk, times, '-ok')
```

```
        hold on
        for t = 2:2:14
            plot(t,time_save, 'b*')
            hold on
            plot([t t], [time_save times(t/2)], '--');
        end
        title('Elapsed time with respect to k, case backward')
        xlabel('K');
        ylabel('Time');
        saveas(gcf, 'bw2.png')
        figure
        plot(valk, iters, '-ok')
        hold on
        for t = 2:2:14
            plot(t,iters_save, 'b*')
            hold on
            plot([t t], [iters_save iters(t/2)], '--');
        end
        title('Iterations with respect to k, case backward')
        xlabel('K');
        ylabel('Iterations');
        saveas(gcf, 'bw3.png')
    end
end
```

Part 2: Summary of results

The function in analysis is a **convex function** and the constraint set is a **convex set** (because it is a n-dimensional hyper-cube) so we expect the **warranty of convergence**.

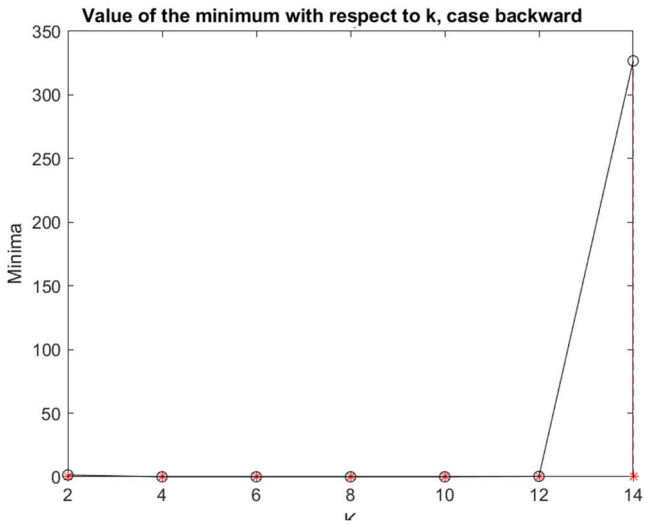
First, we analize the behaviour of the algorithm using the 3 cases of *finite differences*.

In order to decrease the time of elaboration we run the algorithm with **n=100**.

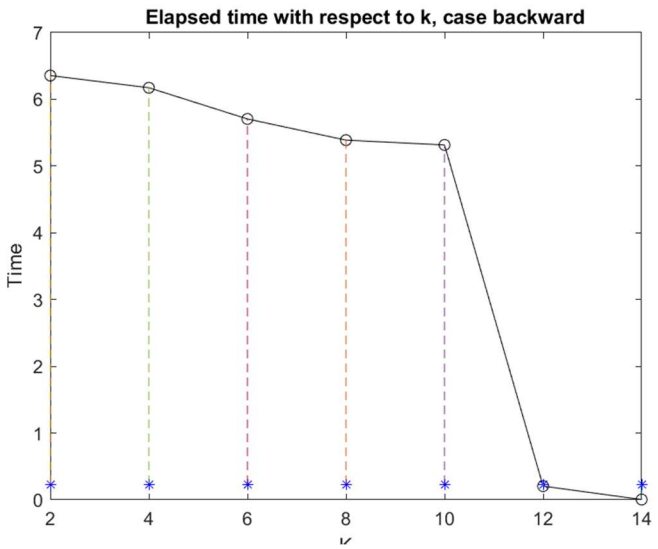
The results are the following:

(**Note:** in the plots, the value marked with '*' represents the *exact solution obtained using derivates*)

Backward finite difference $f'(x) = \frac{f(x+h)-f(x)}{h}$

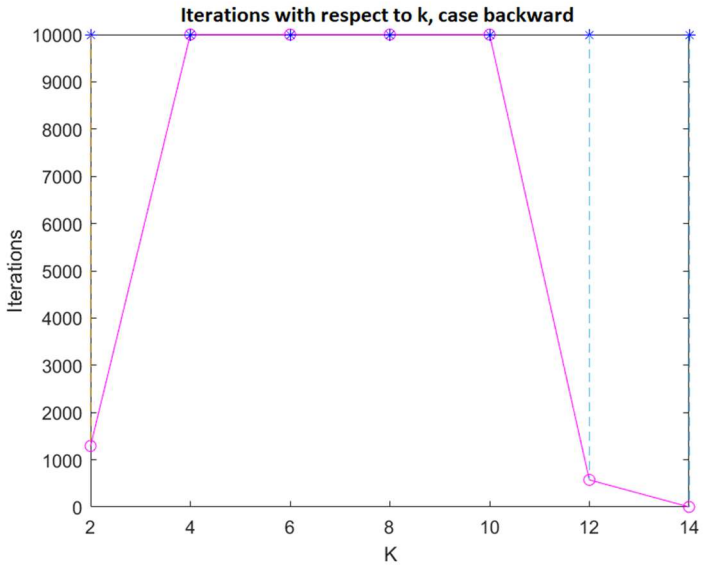


As we expected, the algorithm behaves quite well and **it is able to find the minimum with values of k smaller or equal than 12**, while going beyond this value leads to errors due to **numerical cancellation** and the algorithm does not converge anymore.



The computation time is quite similar for k: [2,10].

It drops down for k=12 (*maybe beacuse the approximation of the derivatives is very close to the exact derivative*) and for k=14 (due to *numerical cancellation*, the method performs very few iterations).



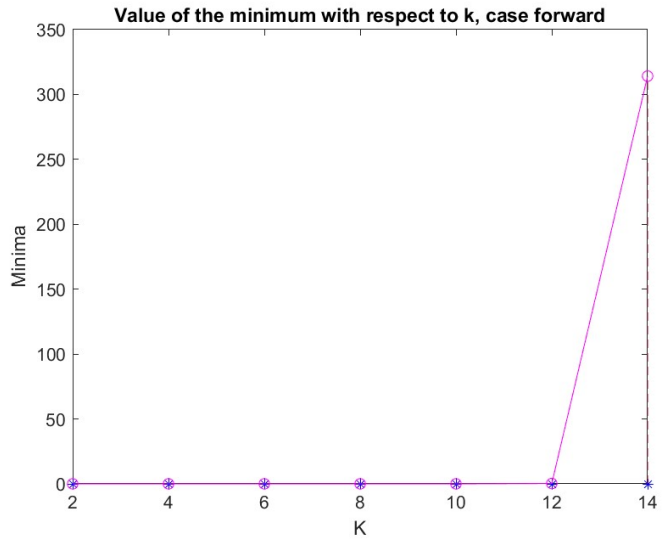
The explanation for this plot is stricly correlated with the plot above.

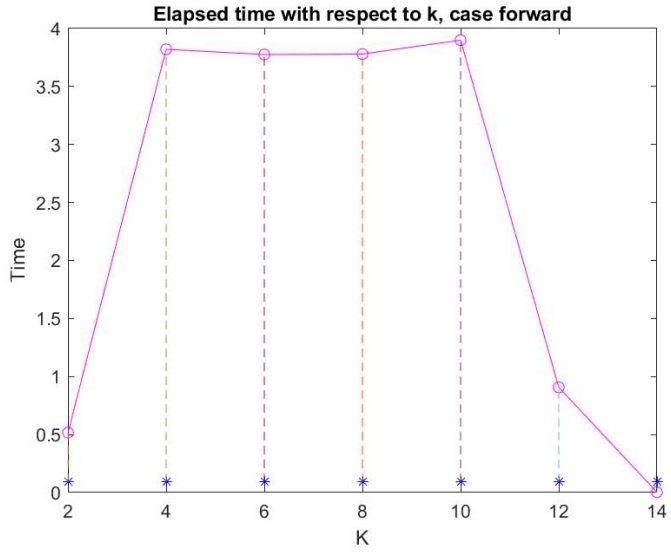
Note that the exact derivatives method reaches the maximum number of iterations (*kmax*) all the times, while for k : [2,12] the algorithm converges in fewer iterations.

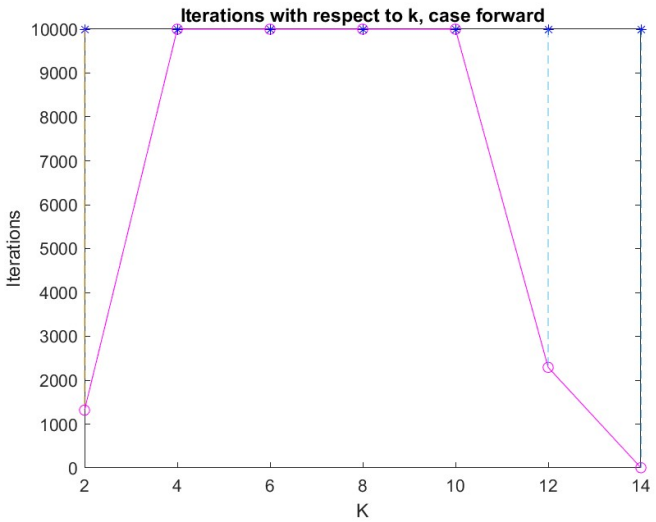
k=12 seems to be the **best choice** both in terms of computational time and number of iterations.

Forward finite difference $f'(x) = \frac{f(x)-f(x-h)}{h}$

This approximation practically leads to the same results of the backward case, **being two quite similar approximations**.

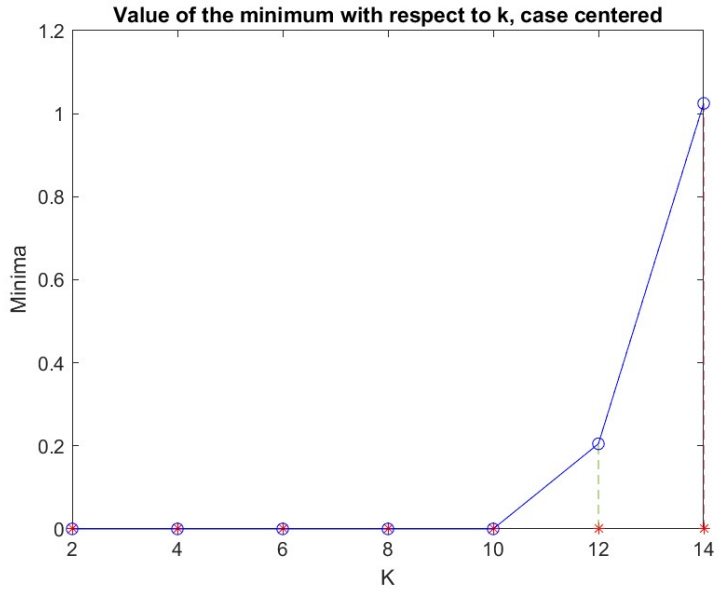






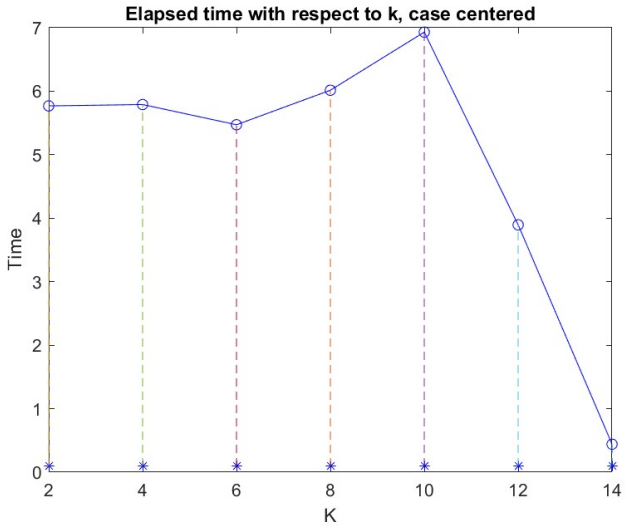
Centered finite difference $f'(x) = \frac{f(x+h)-f(x-h)}{2h}$

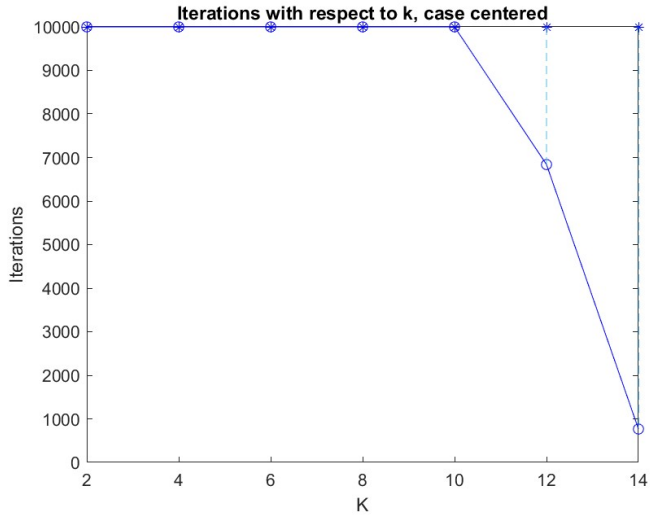
The plots of the results are different from previous cases:



The method is able to find the right minimum until **k ≤ 10.**

As we expected, in this case, being **h** multiplied by a factor **2**, we incur in **numerical cancellation** with greater values of k.



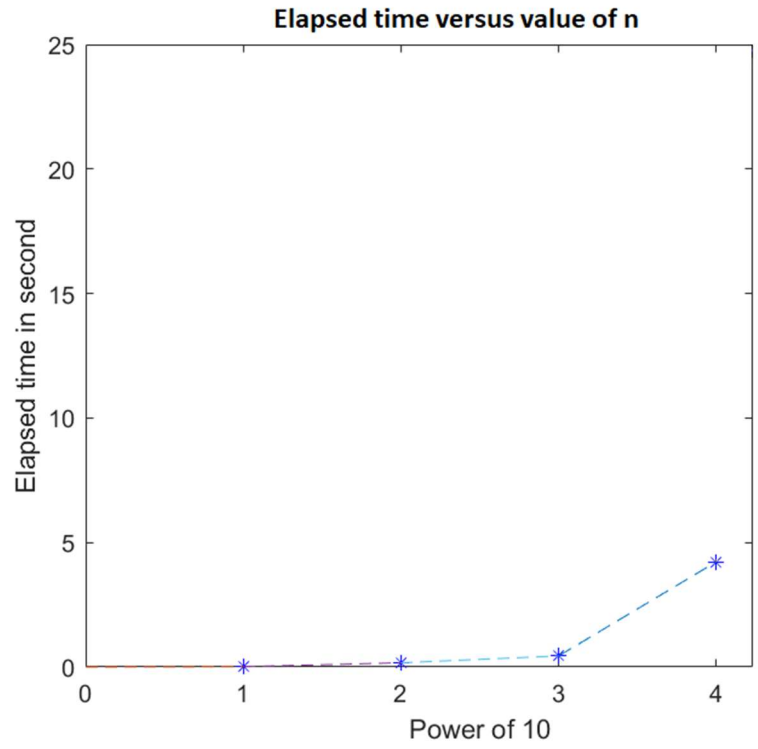


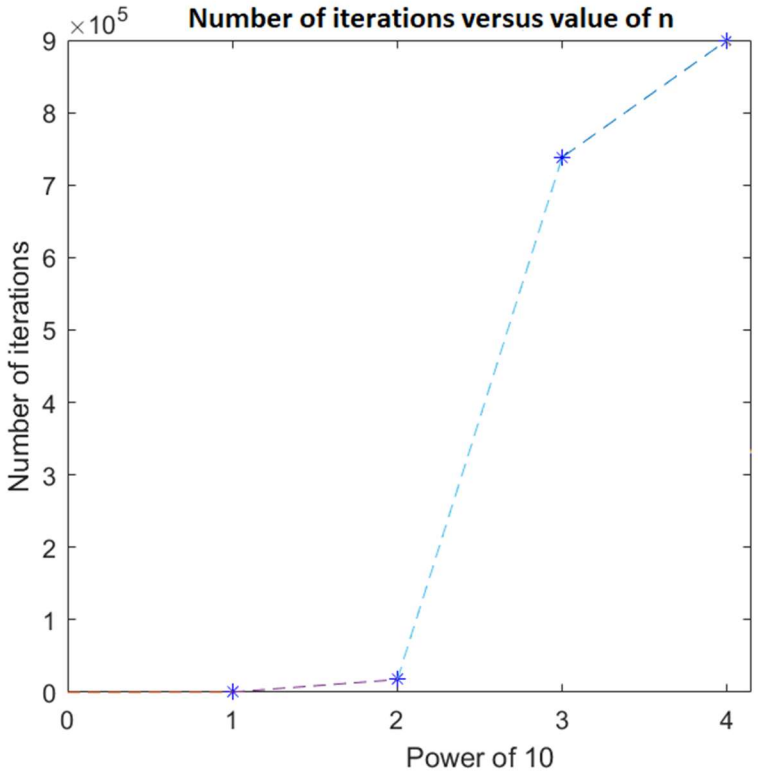
We can notice as before that, due to numerical cancellation for values of **k > 10**, the computational time and the maximum number of iterations **dropped down** but **without reaching the minimum**.

The elapsed time plot also shows that **k=4** is the best choice in terms of **speed**.

Considerations about exact derivatives method

Comparison between the exact derivatives method with different values of **n**, from 10 to 1e4.





The elapsed time and the number of iterations increases exponentially to the dimension of the problem, **as we expected**.

Final Considerations

Summarizing, until the value of **h** is **not too small**, the **algorithm performs very well** in terms of precision using finite differences, but **it is very much slower** if compared to the case where exact derivatives are used.

Indeed, **this is not a problem** if the **dimension** of the system **is not too big**, but the exponentially growing computational time makes unfeasible apply this technique to problem with $\mathbf{n>10^4}$.

We also report that in this case, with the minimum of the function within the boundaries of the constraints, **the algorithm has never pierced the boundaries**, so it has not performed any projection: minimization algorithms for unconstrained optimization should work well too.

Students

Cavagnero Niccolò

Mascarello Andrea

Serra Alessio