Projected gradient method for weighted sphere constrained problem*

*Course: Numerical optimization for large scale problems and Stochastic Optimization

Davide Buoso
Politecnico di Torino
Student ID: s317660

Marco Castiglia
Politecnico di Torino
Student ID: s317381

***Abstract*—In this report we are going to describe the results obtained by minimizing the weighted sphere (or hyper-ellipsoid) constrained optimization problem described by equation 1 introduced in [1] using the projected gradient method exploiting both exact gradient and finite differences to approximate it.**

I. Problem overview

The objective of this report is to analyze the results when looking for a minimum $x_* = (x_1, ..., x_n)^T \in \mathbb{R}^n$, for the problem expressed in Equation 1. Since $f(x)$ is a convex and lower bounded function it has only a single local minimum, therefore, a global minimum. The previous statement implies that the iterative methods will always converge to the same point, independently from the initial guess. Analytically the minimum is $x_{min} = (0, ..., 0)^T$ where $f(x_*) = 0$. Since it is a gradient based method, the analytical gradient was computed as expressed in Equation 2. Also, this will be useful when evaluating the error on the finite differences approximation.

$$f(x) = \sum_{i=1}^{n} i x_i^2, \forall x_i : -5.12 \leq x_i \leq 5.12 \tag{1}$$

$$\frac{\partial f}{\partial x_i} = 2 i x_i, \nabla f(x) = (2x_1, 4x_2, ..., 2n x_n)^T \tag{2}$$

A graphical representation of the function limited to three dimensions can be inspected in Figure 1.

The constrained problem has been tackled using four different scenarios (shown in Fig. 2):

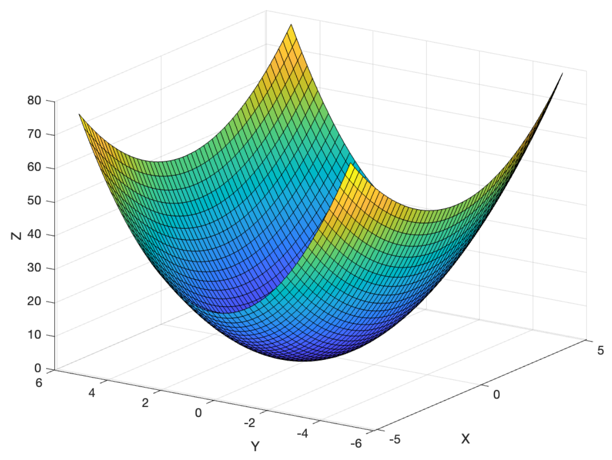
a) $X = [-5.12, 5.12]^n$ (no constraint)
b) $X = [1, 5.12]^n$
c) $X = [-5.12, 5.12] \times [1, 5.12]^{n-1}$
d) $X = [-5.12, 5.12]^{n/2} \times [1, 5.12]^{n/2}$

and with a different number of dimensions: $n = 10^d$ and $d = 3, 4, 5$.

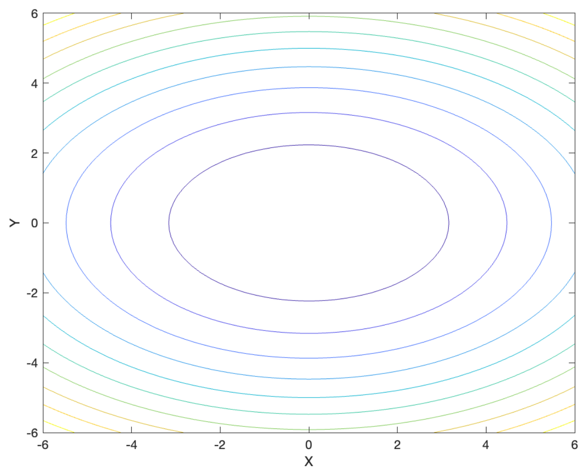
II. Proposed approach

A. Introduction and setup

Our approach is based on the Projected Gradient Descent (PGD). The latter is a numerical optimization method used to find the minimum of a function subject to constraints. It is an iterative method which starts with an initial guess for the solution, then updates the guess in the direction of the negative gradient (steepest descent) until the solution satisfies the constraints. At each iteration, the solution is

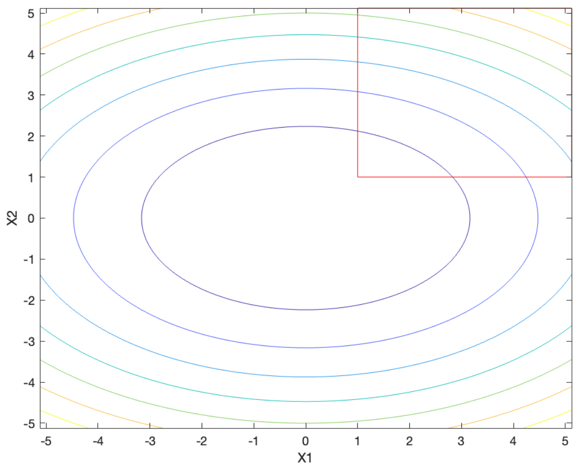


(a) 3D plot of the objective function

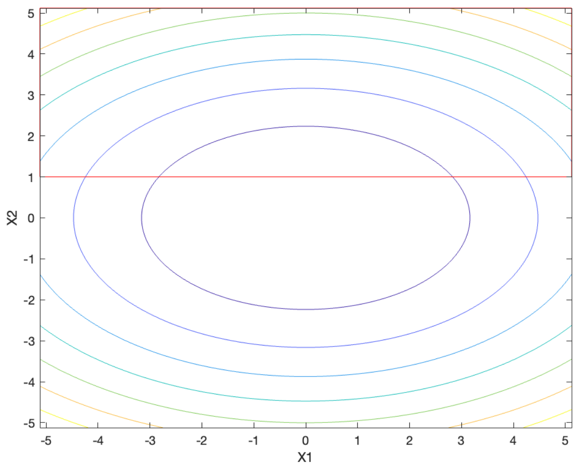


(b) Countour lines of the objective function

Fig. 1: Figure 1a and 1b shows some graphical information about the problem



(a) Contraint b) in 2D



(b) Constraint c) and d) in 2D

Fig. 2: Figure 2a shows the first constraints and 2b the second and the third (they are the same in 2D)

"projected" back onto the feasible region to ensure that it remains within the constraints. The method is commonly used in machine learning, signal processing, and other fields where the objective function is differentiable and constraints are present. PGD is known for its simplicity, efficiency, and ability to handle large-scale optimization problems with many constraints. Given an objective function $f(x)$ and a feasible set $\mathcal{C}$, the objective of PGD is to find x^* that minimizes $f(x)$ subject to $x \in \mathcal{C}$. The algorithm works as follows:

- Initialize $x^{(0)}$.
- For $k = 0, 1, 2, \ldots$ repeat until convergence:

$$\bar{x}^{(k)} = \Pi_{\mathcal{C}} \left(x^{(k)} - \gamma_k \nabla f(x^{(k)}) \right)$$

$$x^{(k+1)} = x^{(k)} + \alpha_k (\bar{x}^{(k)} - x^{(k)})$$

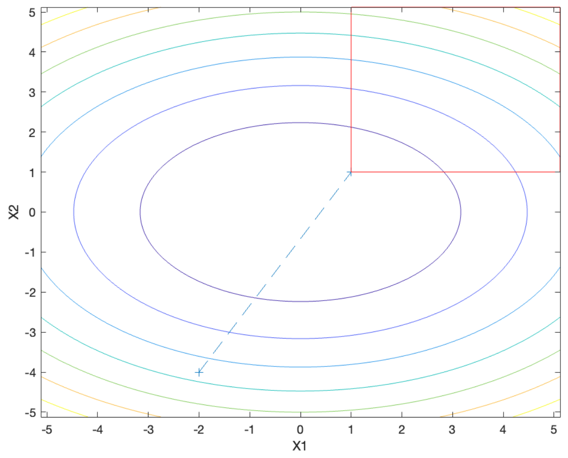


Fig. 3: An example of projection in scenario b) using PGD method.

where $\Pi_{\mathcal{C}}$ is the projection operator that maps $x^{(k)} - \gamma \nabla f(x^{(k)})$ back onto the feasible set $\mathcal{C}$, and α is the step size. An example of the projection is presented in figure 3.

The Projected Gradient Method mainly relies on two scalar parameters γ_k and α_k. The first multiplies the gradient direction before the projection, while the second one multiplies the direction pk towards the projected point. As starting point we used $x_0 = [2, ..., 2]$.
In these tests $\gamma_k = 1$, while α_k is calculated adopting the backtracking strategy, testing the value of α_k against Armijo conditions.
Given an objective function $f(x)$, a feasible set $\mathcal{C}$, the current iterate $x^{(k)}$, and the negative gradient $\nabla f(x^{(k)})$, the Armijo conditions require that the step size α satisfies:

$$f(x^k) - f(x^k + \alpha_k(\bar{x}^k - x^k)) \geq -\sigma \alpha_k \nabla f(x^k)^T (\bar{x}^k - x^k)$$

where $\sigma > 0$ is a fixed constant known as the "sufficient decrease parameter". The parameter ρ used for the backtracking is set to 0.5. Finally, the stopping criterion used were:

- $||x^{k+1} - x^k||_2 \leq 10^{-8}$
- $||\nabla f(x)||_2 \leq 10^{-8}$

B. *Exact derivatives*

We started exploring the problem using analytically computed gradient. The obtained results are presented in Table I. Naturally, increasing the dimensionality longer times were required, especially in the case of no constraint and in the case of the constraint d).
In the scenario a) we obtain the global minimum of the function $x_* = (0, ..., 0)$ while in the other scenarios, it is a different point forced by the constraint:

- Under constraint a) the function converges to: $x_* = (1, ..., 1)$.
- Under constraint b) the function converges to: $x_* = (0, 1, 1, ..., 1)$, a single point to zero and n-1 ones.

- Under constraint c) the function converges to: $x_* = (0, 0, ..., 1, 1)$. In particularly, the first n/2 points are zeros and the last n/2 are ones.

This justifies the high values of the function at convergence. Note: The max number of iterations was set to 10^4.

n	cons	k	f(xk)	norm	Δx	time
10^3	a)	1841	$3.3 \cdot 10^{-12}$	$9.7 \cdot 10^{-6}$	$9.78 \cdot 10^{-9}$	0.11
	b)	2	$500.5 \cdot 10^3$	$3.6 \cdot 10^4$	0	0.012
	c)	3	$500.5 \cdot 10^3$	$3.65 \cdot 10^4$	0	0.010
	d)	10^4	$3.751 \cdot 10^4$	$3.4 \cdot 10^4$	$1.39 \cdot 10^{-6}$	0.51
10^4	a)	9463	$3.76 \cdot 10^{-10}$	$7.48 \cdot 10^{-5}$	$9.95 \cdot 10^{-9}$	10.5
	b)	2	$5 \cdot 10^7$	$1.15 \cdot 10^6$	0	0.004
	c)	3	$5 \cdot 10^7$	$1.15 \cdot 10^6$	0	0.007
	d)	10^4	$3.751 \cdot 10^7$	$1.1 \cdot 10^6$	0.14	9.49
10^5	a)	10^4	$1 \cdot 10^{-2}$	$8.8 \cdot 10^{-2}$	$8.62 \cdot 10^{-6}$	75.9
	b)	2	$5 \cdot 10^9$	$3.65 \cdot 10^7$	0	0.01
	c)	3	$5 \cdot 10^9$	$3.65 \cdot 10^7$	0	0.011
	d)	10^4	$3.751 \cdot 10^9$	$3.41 \cdot 10^7$	0.27	64.4

TABLE I

C. *Inexact derivatives approximation*

The process is similar to the one proposed before but sometimes exact gradient is costly or hardly computable. Therefore, in this section we analyze the previous methodology applying an approximated version of the gradient. This is obtained using the "Centered finite differences" method. The latter involves taking the average of the forward and backward finite differences at a given point to estimate the gradient. The forward finite difference is calculated by subtracting the value of the function at a certain point from the value of the function at a nearby point along the positive direction, while similarly the backward finite difference is calculated by subtracting the value of the function at a certain point from the value of the function at a nearby point along the negative direction. The centered finite difference approximation for the gradient at a given point x is calculated as in 3:

$$\frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

For the parameter $h = 10^{-k}$, different values of k were considered: 2, 6, and 12.
We then decided to analyze only the case where $n = 10^5$ to better notice the differences from the precise analytical gradient. Results are presented in table II.

The difference between using accurate gradient and an approximation in gradient descent lies in the speed and accuracy of convergence. Accurate gradients usually result in faster and more accurate convergence, as they provide more precise updates to the model parameters at each iteration. Approximations, such as finite differences central approximation, have a slower convergence rate and may not converge to the global minimum, however they are simpler to compute and require less computational power. Our study showed that in most cases, convergence using approximations was comparable or slower. The scenario without constraints is easier to analyze since constraints can obstruct the descent of the gradient towards the minimum. In scenario a), the execution time showed a noticeable difference (when n=10^5), with the accurate gradient method taking 75.9 seconds compared to a range of 86.14-125.2 seconds for the approximated method. The execution time increases with the tolerance of the gradient approximation, represented by the parameter h. The final norm of the gradient obtained using the accurate method was one order of magnitude lower than that obtained using the approximation. Despite this, the overall norm was not significantly lower than the previous approach.

cons	h	k	f(xk)	norm	Δx	time
a)	2	10^4	$1 \cdot 10^{-2}$	$8.8 \cdot 10^{-1}$	$8.62 \cdot 10^{-6}$	86.14
	6					92.44
	12					125.2
b)	2	2	$5 \cdot 10^9$	$3.65 \cdot 10^7$	0	0.008
	6					0.008
	12					0.026
c)	2	3	$5 \cdot 10^9$	$3.65 \cdot 10^7$	$2.57 \cdot 10^{-14}$	0.012
	6	3			$1.07 \cdot 10^{-14}$	0.017
	12	10^4			$8.84 \cdot 10^{-5}$	22
d)	2	10^4	$3.75 \cdot 10^9$	$3.41 \cdot 10^7$	0.27	62.65
	6					64.4
	12					64.4

TABLE II

III. Results

In this study, we compared the results of the project gradient method applied to Equation 1 using both finite differences and the exact gradient.

- Using different techniques of the backtracking for the choice of the step size alpha.
- Comparing also the performance of forward/backward finite differences for the approximation of the gradient.

With a maximum of 10^4 iterations, we found that the function consistently converged to the minimum of the constrained set, with satisfactory convergence speed.

References

[1] X.-S. Yang, "Test problems in optimization," p. 2, 08 2010. [Online]. Available: https://arxiv.org/pdf/1008.0549.pdf

Appendix A
Snippets of code

```
clear
close all

load("test_functions_fin.mat")
c1=1e-4;
rho=0.5;
btmax=50;
gamma=1;
tolx=1e-8;
tolgrad=1e-8;
h=1e-12;
n=1e3;

mins0=-5.12*ones(n,1);
maxs0=5.12*ones(n,1);

mins1=1*ones(n,1);
maxs1=5.12*ones(n,1);

mins2=[-5.12;1*ones(n-1,1)];
maxs2=5.12*ones(n,1);

mins3=[-5.12*ones(n/2,1);1*ones(n/2,1)];
maxs3=5.12*ones(n,1);

constrm = mins1;
constrM = maxs1;

kmax=1e4;
proj = @(x) box_projection2(x,constrm, constrM);
f = @(x) weighted_sphere(x);
x0=[-2;-4*ones(n-1,1)];
gradf = @(x) grad_weighted_sphere(x);
%gradf = @(x) iff_grad_weighted_sphere(x,h,'central');

[xk, fk, gradfk_norm, deltaxk_norm, k, xseq, btseq, fseq] = ...
   constr_steepest_desc_bcktrck(x0, f, gradf, ...
   kmax, tolgrad, c1, rho, btmax, gamma, tolx, proj);

[X, Y] = meshgrid(linspace(-5.12, 5.12, 500));
Z = X.^2 + 2*Y.^2;
fig1 = figure();
% Contour plot with curve levels for each point in xseq
[C1, ~] = contour(X,Y,Z);
hold on
plot([x0(1), xseq(1,:)], [x0(2), xseq(2,:)],'--+')
rectangle('Position',[constrm(1:2)', (constrM(1:2)-constrm(1:2))' ], 'EdgeColor','r')
xlabel("X1");
ylabel("X2");

hold off
```

```
figure();
iter = 1:k; % create a vector for the number of iterations
hold on
plot(iter, fseq2(1:k))% plot the function values against the number of iterations
plot(iter, fseq(1:k))
xlabel(’Number␣of␣Iterations’) % label the x-axis
ylabel(’Function␣Value’) % label the y-axis
title(’Function␣Value␣vs␣Number␣of␣Iterations’) % add a title to the plot
legend(’Sequence␣1’, ’Sequence␣2’)
ylim([0 50])


figure();
surf(X, Y, Z, ’EdgeColor’, ’none’)
plot3([x0(1) xseq(1, :)], [x0(2) xseq(2, :)], [f1(x0), f1(xseq)], ’r--*’)


function y = weighted_sphere(x)
   [n, ~] = size(x);
   y = sum([1:n]’ .* (x .^ 2));
end

function grad = grad_weighted_sphere(x)
   [n, ~] = size(x);
   grad = [2:2:2*n]’ .* x;
end

function grad = iff_grad_weighted_sphere(x, h, diff_type)
   h=h*norm(x);
   [n, ~] = size(x);
   seq = (1:n)’;
   switch diff_type
      case ’forward’
         grad = seq .* ((x+h).^2 - (x).^2) ./ h;
      case ’central’
         grad = seq .* ((x+h).^2 - (x-h).^2) ./ (2.*h);
      otherwise
         error(’Requested␣difference␣not␣available’)
   end
end
```