

I. Cravero* - M. Introna^{†*}

A STUDY ON OPTIMISATION METHODS IN CELESTIAL MECHANICS

Abstract. This text explores advanced methods for solving the N -body problem, particularly through variational approaches and optimization techniques. The focus is on finding critical points of the action functional, which correspond to potential orbits. Various optimization strategies, including trust region methods, constrained optimization, and metaheuristic methods, are discussed for their effectiveness in identifying minima, saddle points, and other critical points. The text emphasizes the importance of choosing efficient algorithms and the challenges in distinguishing between local and global solutions in nonlinear optimization problems. The goal is to gather a wide range of solutions to gain deeper insights into the system's behaviour and uncover new properties.

1. Introduction

The N -body problem in the celestial mechanics consists in the study of the motion of N point particles (the bodies) in the d -dimensional space, when just the gravitational forces induced by the other $N - 1$ act on each particle. To formalize this dynamical system, we first define the masses of the bodies as $m_1, \dots, m_N > 0$ and their respective positions as $x_1(t), \dots, x_N(t)$. These positions can be represented by the vector $x(t) = (x_1(t), \dots, x_N(t)) \in \mathbb{R}^{dN}$. The gravitational force exerted by mass m_j on mass m_i is given by

$$F_{ij}(x) = m_i m_j G \frac{x_j - x_i}{|x_j - x_i|^3}$$

with G being the gravitational constant. For simplicity, we assume $G = 1$. The cumulative force on the i -th mass is

$$F_i = \sum_{j \neq i, j=1}^n F_{ij}$$

and, according to Newton's law, it is represented by

$$m_i \ddot{x}_i(t) = F_i.$$

It is worth noting that F_i is not a central force field. By dividing both sides of the previous equation by m_i , we derive the N -body dynamical system

$$\begin{cases} \ddot{x}_i(t) = \sum_{j \neq i, j=1}^n m_j \frac{x_j - x_i}{|x_j - x_i|^3} \\ i = 1, \dots, n. \end{cases} \quad (1)$$

This system is valid for $x \in \mathbb{R}^{dN} \setminus \Delta$, where Δ denotes the collision set

$$\Delta = \left\{ x \in \mathbb{R}^{dN} : x_i = x_j \text{ for some } i \neq j \right\}.$$

*Department of Mathematics, University of Turin, Via Carlo Alberto 10, Torino TO, 10123, Italy.

[†]Department of Physics, University of Trento, Via Sommarive 14, Povo TN, 38123, Italy.

By introducing the force vector

$$F(x) = (F_1(x), \dots, F_N(x)) \in \mathbb{R}^{dN}$$

and the diagonal-block matrix of dimension $dN \times dN$

$$M = \text{diag}(m_1 I_d, \dots, m_N I_d),$$

we can represent system [1] in a compact form as

$$M\ddot{x}(t) = F(x(t)).$$

This dynamical system [1] conserves energy, and the C^1 scalar function

$$V(x) = \sum_{i < j} \frac{m_i m_j}{|x_i - x_j|} \quad x \in \mathbb{R}^{dN} \setminus \Delta$$

is such that

$$M\ddot{x}(t) = \nabla V(x(t)).$$

So, if $x : I \subset \mathbb{R} \rightarrow \mathbb{R}^{Nd}$ is a solution of the dynamical system [1], then the *total energy* is conserved along the motion x , that is

$$\frac{1}{2} \langle M\dot{x}(t), \dot{x}(t) \rangle - V(x(t)) = h, \quad \text{for any } t \in I, \quad (2)$$

for some constant $h \in \mathbb{R}$. The function

$$K(p) := \frac{1}{2} \langle Mp, p \rangle, \quad p \in \mathbb{R}^{Nd}$$

is the *kinetic energy*, while the *potential energy* is $-V(q)$, $q \in \mathbb{R}^{Nd} \setminus \Delta$.

Analysing the general N -body problem, one can prove the existence of some special periodic solutions (relative equilibria and homographic solutions, starting from central configurations). In recent years, a variational approach to the problem has been used which has brought significant progress in the field and new possible orbits have been determined. We refer to the following articles and references therein: [2, 3, 21, 22]. For this purpose, we introduce $\mathbb{T}$, the unit circle of length T , and the space of centred configurations $\mathcal{X}$, a subspace of $\mathbb{R}^{Nd}$ formed by the points $x = (x_1, x_2, \dots, x_N)$ with centered center of mass $\sum_{i=1}^N m_i x_i = 0$. Let $\Lambda = H^1(\mathbb{T}, \mathcal{X})$ the Sobolev space of the L^2 loops $\mathbb{T} \rightarrow \mathcal{X}$ with L^2 derivatives. The new method consists in the minimization of the Lagrangian action functional $\mathcal{A}$:

$$\mathcal{A}(x) = \int_0^T \frac{1}{2} \langle M\dot{x}(t), \dot{x}(t) \rangle + V(x(t)) dt, \quad (3)$$

on a space of loops Λ where $x_i(t) \in \mathbb{R}^d$, with $d = 2$ or $d = 3$. The action functional $\mathcal{A}$ is of class C^1 on the subspace $\hat{\Lambda} \subset \Lambda$ of the collisionless loops.

We can now consider the restriction of the action functional $\mathcal{A}$ to a functional defined on a subspace of G -equivariant loops, where G denotes an appropriate finite group. In this case, the problem can be defined in a simpler way: it becomes a *Bolza*-conditioned problem. As a matter of fact, the action functional is analysed on a particular subdomain, the so called fundamental domain, and, thanks to the group of symmetries G , the result can be copied symmetrically, obtaining the entire orbit. Furthermore, the initial conditions become conditions that are only restricted to the border of this fundamental domain, therefore *Bolza* conditions. This is a fundamental step towards the minimisation problem, since it allows to switch from the analysis of periodic loops to the minimisation among fixed-end points which share initial and final conditions fixed on extrema of such subdomains. For more details, see [4].

So, while in the classic problem the solutions were sought as solutions to differential systems, in this new approach the solution trajectories are seen as the search for the critical points of the action functional.

At this point, it becomes important to use effective and efficient optimisation methods to determine such trajectories. optimisation is the process of minimizing or maximizing a function subjected to constraints on its variables. The first step to tackle is therefore to identify the real variables and constraints related to the problem. We give a brief overview of how this process can be carried out. We consider [3], where $N, d, m_1, \dots, m_N > 0$ are input values. Our goal is to reckon the N trajectories $x_1(t), \dots, x_N(t)$, with $x_i(t) \in \mathbb{R}^d$, $i = 1, \dots, N$ as the minima of the action functional $\mathcal{A}$.

We have two possible choices for modeling our problem. The first one consists in discretizing the integral operator using an appropriate quadrature technique and subsequently using finite differences for the approximation of derivatives. In particular, we consider an uniform grid $0 = t_0 < t_1 < \dots < t_{M+1} = T$ so t_k represents the k -th instant of the interval of time $I = [0, T]$. We put

$$G(t) = G(x_1(t), x_2(t), \dots, x_N(t)) = \sum_{i=1}^N \frac{m_i}{2} |\dot{x}_i|^2 + \sum_{i<j} \frac{m_i m_j}{|x_i - x_j|}$$

and we consider a composite trapezoidal rule on the $M+1$ intervals

$$\mathcal{A} = \int_0^T G(t) dt \sim \frac{h}{2} \left[G(t_0) + G(t_{M+1}) + 2 \sum_{k=1}^M G(t_k) \right]$$

where $h = T/(M+1)$, considering $y_i^k \sim x_i(t_k) \in \mathbb{R}^d$, $i = 1, \dots, N$, $k = 0, \dots, M+1$ and together with a forward difference formula we obtain

$$\begin{aligned} \mathcal{A} &\sim \sum_{k=1}^M \left[\sum_{i=1}^N \frac{m_i}{2} \frac{|y_i^{k+1} - y_i^k|^2}{h} + h \sum_{i<j} \frac{m_i m_j}{|y_i^k - y_j^k|} \right] \\ &+ \sum_{i=1}^N \frac{m_i}{4h} \left[|y_i^1 - y_i^0|^2 + |y_i^{M+2} - y_i^{M+1}|^2 \right] \\ &+ \frac{h}{2} \sum_{i<j} m_i m_j \frac{|y_i^0 - y_j^0| + |y_i^{M+1} - y_j^{M+1}|}{|y_i^0 - y_j^0| |y_i^{M+1} - y_j^{M+1}|} = \mathcal{A}_h^{(1)} \end{aligned} \quad (4)$$

where we introduce opportune ghost points y_i^{M+2} . Consequently, we have to optimise the object function

$$f(y_i^k) = \mathcal{A}_h^{(1)}, \quad i = 1, \dots, N, \quad k = 1, \dots, M \quad (5)$$

with constraints $y_i^0 \in H_0$ and $y_i^{M+1} \in H_1$, $i = 1, \dots, N$, where H_0, H_1 are subspaces of $\mathbb{R}^d$ and the numerical solution is a block real matrix $(y_i^k)_{i,k} \in \mathbb{R}^d$, $i = 1, \dots, N$ and $k = 1, \dots, M$, with dimension $d \times N \times M$. For further details, see [9–12].

An alternative strategy relies on a Fourier expansion of the trajectories; related methods can be found, for instance, in [25–28]. Following the approach adopted in [4, 24], the trajectory is decomposed into a linear interpolant connecting the boundary configurations and a finite Fourier correction. More precisely, we write

$$x(t) = x_0 + \frac{t}{\pi} (x_1 - x_0) + \psi(t), \quad (6)$$

for $t \in [0, \pi]$, where the remainder term satisfies the homogeneous boundary conditions $\psi(0) = \psi(\pi) = 0$. The function $\psi(t)$ is approximated by a truncated Fourier sine series. Let $F \in \mathbb{N}$ denote the number of retained modes; then

$$\psi(t) = \sum_{k=1}^F A_k \sin(kt), \quad t \in [0, \pi],$$

with coefficients $A_k \in \mathbb{R}^{d \times n}$. The choice of sine functions guarantees compatibility with the prescribed boundary conditions. Inserting the ansatz [6] into the action functional [3] yields a reduced functional of the form

$$f_2(x_0, x_1, A_k) = \mathcal{A}_h^{(2)}, \quad k = 1, \dots, F. \quad (7)$$

This formulation leads to an optimisation problem involving $(F + 2) \times d \times n$ real variables, which can be solved using suitable numerical minimisation techniques. Additional implementation details are discussed in [4, 24].

In both models, we have introduced scalar functions f_i , $i = 1, 2$, given by [5] or by [7], in the variables $x = y_i^k$ or $x = A_k$ where the vector x represents the unknowns of our problem and some constraints. The functions $f = f_i$, $i = 1, 2$, to be maximized or minimized is usually referred to as the objective function.

The various methods and approaches presented in this work are used to collect as many critical points as possible of the action functional, defined in [4]. These critical points include minima, maxima and saddle points. Once obtained this large set, it is fundamental to be able to distinguish them or to consider them equal when they give the same orbit. As a matter of fact, an overview of a large scale of possible orbits of the N -body dynamical system would give us more information on how this system develops. Moreover, using the orbits obtained numerically, we might be able to discover new properties and previously hidden patterns.

There are no universal algorithms to solve a specific problem, but rather a collection of algorithms, each of which is tailored to a particular optimisation problem.

To set the notation, let us consider x as a vector of variables, which are the parameters or unknowns. Then f is a scalar function called the objective function that represents the function we want to minimize or maximize. Finally, we have two sets of scalar functions which are the constraint functions, denoted by c_i and d_i that define certain equations or inequalities, respectively, that must be satisfied.

Optimisation algorithms are always iterative methods that are distinguished based on the technique used to move from one iteration to the next. Roughly speaking, we can distinguish between methods that obtain the next iteration using the values of f , the values of the constraint functions $d_i(x)$, and possibly their first or second derivatives, or methods that take advantage of the information accumulated in previous steps, or even local information derived from the current point.

Choosing an algorithm is not a simple matter but important, as it determines how quickly or slowly a problem can be solved and above all how satisfactory the solution can be. An algorithm must be efficient in terms of memory usage or computational time, and accurate, meaning it must be able to determine the solution with precision, without being too sensitive to errors introduced in the data or rounding errors introduced once the algorithm is implemented on a computer, remembering, however, that these goals often conflict with each other. (optimality conditions, sensitivity analysis?)

We underline the importance of the regularity of the functions involved both to be able to exploit information on constraints and objects at nearby points and to establish the nature of the solutions found. We speak of local or global solutions. Many algorithms for nonlinear optimisation problems lead to a local solution or points where the objective function is smaller compared to the value of the function at nearby points. It is not always easy to find a global solution that represents the point where the objective function takes the smallest value over the entire domain. In cases where f is a convex function, global solutions are also local. In general, nonlinear problems, both constrained and unconstrained, have local solutions that are not global and a large number of local solutions must be sought, to which global optimisation algorithms can then be applied.

The aim of our work is to present a brief survey of the possible optimisation methods. With the advent of artificial intelligence, particularly in the realms of machine learning and deep learning, the optimisation techniques have gained renewed interest and importance while, historically, optimisation techniques have been pivotal in areas ranging from operations research to economics. As a consequence, the motivation of this work is to highlight the research and improvements made in the last years. As a matter of fact, we have not considered several well known methods, such as the *Gradient Methods* and the *Newton's Methods*. Both categories are considered as fundamental techniques used in optimisation and they have been generalised and improved during the past years. In the upcoming sections, we delve into various optimisation techniques and their applications. Section 2 focuses on trust region methods, laying the groundwork for more specialized topics; see for example [5–7, 31]. In Section 3, we explore constrained optimisation, with particular

attention given to penalization methods, nonlinear optimisation, and the integral roles of merit functions and filters in the optimisation process. Lastly, in Section 4, we pivot to metaheuristic methods, discussing techniques such as tabu search, simulated annealing, and genetic algorithms. Each section aims to provide comprehensive insights into these respective methods and their implications.

All the described methods mainly focus on the research of minima. If one is interested in saddle points and critical points of Morse index major than 2, they can on the one side exploit these methods to find minima and then use, for example, bisection algorithms to search for mountain pass critical points (i.d. saddle points of Morse index equal to 1); on the other side, they can impose as objective function to minimise the gradient, so to find critical points, which may include saddle points. In general, the metaheuristic methods can be seen as the most versatile ones for the search of critical points which are not global or local minima.

2. Trust-Region Methods

A general algorithm for minimization, starting from an initial point x_0 produces a series of iterations, denoted as $\{x_k\}_{k=0}^{\infty}$ which conclude either when further improvements seem unattainable or when the solution appears to be approximated with adequate precision. To transition from one iteration x_k to the next, these algorithms utilize data about the function f at the point x_k and potentially also from prior iterations like $x_0, x_1, \dots, x_{k-1}$. Leveraging this data, they identify a subsequent iteration x_{k+1} that has a more favorable function value than x_k . There are two principal strategies for moving from the current point x_k to a new iterate x_{k+1} . In the line search approach, the algorithm picks a direction, denoted as p_k , for example $p_k = -\nabla f(x_k)$ in the gradient methods or $p_k = -H_k(x_k)^{-1}\nabla f(x_k)$ in the Newton methods, where $H_k(x_k)$ denotes the Hessian of f in x_k , and then searches along this direction from the current point x_k to find a new point with a better function value. The extent to which one should move in the direction p_k is determined by attempting to solve a one-dimensional optimisation problem to identify an optimal step size, α , given by:

$$\min_{\alpha > 0} f(x_k + \alpha p_k).$$

Ideally, by solving this problem precisely, one would maximize the benefits from the chosen direction p_k . However, finding the exact solution might be computationally intensive and often is not required. Instead, the line search method tests a few potential step sizes until it identifies one that provides a good approximation to the minimum of the problem. Once this new position is established, the algorithm determines a new search direction and step size, and the cycle continues.

In the trust region approach, the information collected about the function f is utilized to create a model function, denoted as m_k , that mimics the behaviour of f around the current point x_k . However, since m_k might not accurately represent f at points distant from x_k , the search for an optimal point for m_k is limited to a vicinity around

x_k . Specifically, the potential step p is determined by trying to solve:

$$\min_p m_k(x_k + p) \quad \text{subjected to } x_k + p \text{ being within the trust region.}$$

If this proposed step does not lead to a significant reduction in f , it implies that our trust in the region's size might be misplaced. Consequently, the region is reduced in size, and the problem is tackled again. Typically, the trust region is a sphere described by $\|p\|_2 \leq \Delta$, where $\Delta > 0$ is the radius of this trust region. Other shapes, like ellipsoids or rectangular regions, can also be employed.

Furthermore, we consider m_k to be based on the Taylor-series expansion of our objective function f at x_k , therefore

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T H_k(x_k) p + \mathcal{O}(\|p\|^3),$$

where $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$ and t is a scalar in $(0, 1)$. We consider an approximation B_k of the Hessian $H_k(x_k)$ in the second-order term, therefore we can define m_k as

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p,$$

where B_k is a symmetric matrix. One can note that $m_k(p)$ and $f(x_k + p)$ differ of a factor $\mathcal{O}(\|p\|^2)$ that is small when p is small. In the trust region Newton methods, we make the following choice $B_k = H_k(x_k)$, which leads to an approximation error of $\mathcal{O}(\|p\|^3)$, therefore we obtain an accurate model when $\|p\|$ is small.

Trust region methods are considered optimal and general, as few properties, only symmetry and uniform boundedness, are imposed on B_k , making it more usable in several problems.

A step is obtain finding a solution to the following problem

$$\min_{p_k \in \mathbb{R}^n} m_k(p_k) = f_k + g_k^T p_k + \frac{1}{2} p_k^T B_k p_k \quad \text{s.t. } \|p_k\| \leq \Delta_k, \quad (8)$$

where $\Delta_k > 0$ is the trust-region radius; therefore the solution p_k^* of [8] can be seen as the minimiser of m_k in the ball of radius Δ_k . One can verify that p_k^* is a solution of [8] if it solves the following

$$(B_k + \lambda_k I) p_k^* = -g_k$$

for a $\lambda_k \geq 0$ with $B_k + \lambda_k I$ semidefinite positive and the condition $\lambda_k(\Delta_k - \|p_k^*\|) = 0$.

OBSERVATION. One can note that when B_k is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, we can easily find the solution of [8] as the unconstrained minimum $p_k^B = -B_k^{-1} g_k$ of the quadratic $m_k(p)$. In this specific case, p_k^B is defined as *full step*.

Radius Δ_k of the trust-region. The key element in a trust region method is the choice of the radius Δ_k at each iteration. The agreement between the model function

m_k and the objective function f at previous iterations is used to make an accurate choice of the current trust region. Given a step p_k , one can define the following ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}, \quad (9)$$

where the numerator is known as the *actual reduction* while the denominator as the *predicted reduction*. When ρ_k is negative, which means $f(x_k)$ is smaller than the current value $f(x_k + p_k)$, one rejects the step. However, when $\rho_k \rightarrow 1$, the model m_k and the function can be considered similar over the step, therefore one can enlarge the trust region in the following iteration. On the other hand, when $0 < \rho_k < 1$ one does not modify the trust region; in case ρ_k near to 0 then one reduces the trust region at the next iteration.

Algorithm 1: Trust Region Method

Input: Set $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$ and $\eta \in [0, \frac{1}{4})$.

Output: Approximation of the solution.

for $k = 0, 1, 2, \dots$ **do**

 Reckon p_k by solving [8] in an approximate way;

 Evaluate ρ_k using [9];

if $\rho_k < \frac{1}{4}$

then

$\Delta_{k+1} = \frac{1}{4} \Delta_k$;

else

if $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

then

$\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$;

else

$\Delta_{k+1} = \Delta_k$;

if $\rho_k > \eta$

then

$x_{k+1} = x_k + p_k$;

else

$x_{k+1} = x_k$.

return the current trial solution.

We denote $\hat{\Delta}$ as an overall bound on the step lengths.

OBSERVATION. One increases the radius only in the case in which $\|p_k\|$ reaches the boundary of the trust region. When the step does not reach the boundary the current value of Δ_k is considered not to be interfering with the progress of the algorithm, therefore it is not altered.

The Cauchy Point. Though in theory one looks for the optimal solution of the subproblem [8], for purpose of global convergence, it is sufficient to find an approximate

solution p_k , which is in the trust region, and provide a *considerable reduction* in the model. One can quantify this considerable reduction using the Cauchy point, which can be denoted by p_k^c . We define this point using a particular procedure. We compute a vector p_k^s which is the vector that solves a linear version of [8], which is

$$p_k^s = \underset{p_k \in \mathbb{R}^n}{\operatorname{argmin}} f_k + g_k^T p_k \quad \text{s.t. } \|p_k\| \leq \Delta_k; \quad (10)$$

Then we find the scalar $\tau_k > 0$ that minimises $m_k(\tau p_k^s)$ which has to satisfies the trust-region bound, that is

$$\tau_k = \underset{\tau \geq 0}{\operatorname{argmin}} m_k(\tau p_k^s) \quad \text{s.t. } \|\tau p_k^s\| \leq \Delta_k; \quad (11)$$

and then we set $p_k^c = \tau_k p_k^s$.

The Cauchy step p_k^c can be calculated in an inexpensive way and it is fundamental when one has to decided whether to accept or not an approximate solution of the trust-region subproblem. In detail, one can prove that a trust-region method globally converges if its steps p_k give a reduction in the model m_k that is at least some fixed positive multiple of the decrease attained by the Cauchy step.

OBSERVATION. One does not take the Cauchy point as our step because otherwise it would be like implementing the steepest descent method using a particular choice of step length. It is known that steepest descent method performs poorly even if an optimal step length is used at each iteration.

Therefore, when using trust region methods, one usually computes the Cauchy point and then makes an improvement of it. In order to do so, one designs the algorithm so that the full step $p_k^B = -B_k^{-1} g_k$ is selected when B_k is positive definite and $\|p_k^B\| \leq \Delta_k$. Let us see now two methods to approximately solve [8] which are based on the strategy just described, i.e.

1. Doglet Method: it is used if the model Hessian B_k is positive definite.
2. Two-dimensional-subspace-minimisation: it is used when B_k is indefinite, although an estimate of the most negative eigenvalue of B_k is needed.

2.1. The Doglet Method

As previously mention, one can use this method when B_k is positive definite. It is important to see how the trust-region radius Δ_k effects the solution $p_k^*(\Delta_k)$ of the subproblem [8]. When B_k is positive definite, we know that the unconstrained minimiser of m_k is $p_k^B = -B_k^{-1} g_k$. Of course, when this point is considered feasible, it is a solution; therefore one has

$$p_k^*(\Delta_k) = p_k^B, \quad \text{when } \Delta_k \geq \|p_k^B\|.$$

In the case in which Δ_k is small in relation to p_k^B , the constraint $\|p_k\| \leq \Delta_k$ assures us that the solution of [8] is not so effected by the quadratic term in m_k . For such choice of Δ_k , one can reckon an approximation to $p_k(\Delta_k)$ that is

$$p_k^*(\Delta) \approx -\Delta_k \frac{g_k}{\|g_k\|}, \quad \text{when } \Delta_k \text{ is small.}$$

For intermediate values of Δ_k , one can note that the solution $p_k^*(\Delta_k)$ usually follows a curved trajectory.

The doglet method looks for an approximate solution by substituting this curved trajectory for $p_k^*(\Delta_k)$ with a path, which consists of two line segments. The former one starts from the origin until the minimiser of m_k is reached, following the steepest descent direction, which can be written as

$$p_k^U = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k;$$

while the latter one starts from p_k^U until p_k^B . It can be summarise as the trajectory $\tilde{p}(\tau)$ with $\tau \in (0, 2)$ that is defined as

$$\tilde{p}_k(\tau) = \begin{cases} \tau p_k^U, & 0 \leq \tau \leq 1, \\ p_k^U + (\tau - 1)(p_k^B - p_k^U), & 1 \leq \tau \leq 2. \end{cases}$$

In conclusion, the doglet method selects p_k to minimise the model m_k , following this path constrained to the trust-region bound. One can prove that this minimum can be easily found.

2.2. Two-dimensional-subspace-minimisation

When B_k is positive definite, one can improve the doglet method; as a matter of fact, the search for p_k can be widened to the entire two-dimensional subspace spanned by p_k^U and p_k^B , i.e. spanned by g_k and $-B_k^{-1}g_k$. So one can substitute the subproblem [8] with the following one

$$\min_{p_k} m_k(p_k) = f_k + g_k^T p_k + \frac{1}{2} p_k^T B_k p_k \quad \text{s.t. } 0 \leq \|p_k\| \leq \Delta_k, \quad p_k \in \text{span}[g_k, B_k^{-1}g_k]. \quad (12)$$

One can deduce that the Cauchy point p_k^c is feasible for [12]; therefore, we know that we would get the same reduction, given by the optimal solution of this subproblem, in m_k as the one given by the Cauchy point, which leads to global convergence of the algorithm.

One can modify this strategy to adapt it to the case of B_k being indefinite. We only present here the general idea, without going into details. When B_k has two negative eigenvalues, the two-dimensional subspace is altered in [12] into

$$\text{span}[g_k, (B_k + \alpha I)^{-1}g_k] \quad \text{for some } \alpha \in (-\lambda_1, -2\lambda_1], \quad (13)$$

where we denote λ_1 as the most negative eigenvalue of B_k . One makes this choice for α so that $B_k + \alpha I$ is positive definite and the range of α gives more freedom in terms of methods to compute it. When $\|(B_k + \alpha I)^{-1} g_k\| \leq \Delta_k$, the subspace search of [12] and [13] is substituted with the following definition of the step to be

$$p_k = -(B_k + \alpha I)^{-1} g_k + v,$$

where v defines the vector that satisfies $v^T (B_k + \alpha I)^{-1} g_k \leq 0$, so one has that $\|p_k\| \geq \|(B_k + \alpha I)^{-1} g_k\|$.

On the other hand, when B_k has zero eigenvalues but no negative ones, the step to be is defined as the Cauchy point $p_k = p_k^c$.

3. Constrained optimisation

In mathematical optimisation, *constrained optimisation* can be defined as the process of optimising an objective function with respect to some variables in the presence of constraints on those variables. The objective function can be either a cost function or energy function, which is to be minimized, otherwise a reward function or utility function, which is to be maximized. Moreover, one can have different possible constraints. They can be either *hard* constraints, which set conditions for the variables that are required to satisfy, or *soft* constraints, which have some variable values that are penalized in the objective function if the conditions on the variables are not satisfied.

Let us consider the problem in the following form:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subjected to} \quad & c_i(x) = 0 \quad \text{for } i \in \mathcal{J} \quad \text{equality constraints} \\ & d_i(x) \geq 0 \quad \text{for } i \in \mathcal{E} \quad \text{inequality constraints,} \end{aligned} \quad (14)$$

where the objective function f and the constraint functions are all smooth, real-valued functions on a subset of $\mathbb{R}^n$.

Let us give some fundamental definitions:

DEFINITION 1. *We define the feasible set Γ to be the set of points x that satisfy the constraints; that is,*

$$\Gamma = \{x \mid c_i(x) = 0, i \in \mathcal{J} \text{ and } d_i(x) \geq 0, i \in \mathcal{E}\}.$$

DEFINITION 2. *The active set $\Omega(x)$ at any feasible x consists of the equality constraint indices from $\mathcal{J}$ together with the indices of the inequality constraints i for which $d_i(x) = 0$; that is:*

$$\Omega(x) = \mathcal{J} \cup \{i \in \mathcal{E} \mid d_i(x) = 0\}.$$

At a feasible point x , the inequality constraint $i \in \mathcal{E}$ is said to be active if $d_i(x) = 0$ and inactive if the strict inequality $d_i(x) > 0$ is satisfied.

In other words, an active constraint is a constraint that "binds" or "restricts" the solution at that point; while an inactive constraint does not impact the solution at that point.

In constrained optimisation, it is crucial to determine which constraints are active and which are not, as this affects the direction in which one can move to improve the current solution. Specifically, when using active-set based methods, the idea is to start with a guess about the active constraint set and solve a simpler problem based only on these constraints. If the guess turns out to be correct, you have an optimal solution. Otherwise, the active set is updated, and the process is repeated.

DEFINITION 3. *Given the point x and the active set $\Omega(x)$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla d_i(x), i \in \Omega(x)\}$ is linearly independent.*

DEFINITION 4. *We define the Lagrangian function for the general problem [14] as*

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{I}} \lambda_i c_i(x) - \sum_{i \in \mathcal{E}} \lambda_i d_i(x). \quad (15)$$

THEOREM 2. *Suppose that x^* is a local solution of [14], that the functions f, c_i and d_i in [14] are continuously differentiable, and that the LICQ holds at x^* . Then there is a Lagrange multiplier vector λ^* , with components λ_i^* , $i \in \mathcal{I} \cup \mathcal{E}$, such that the following conditions are satisfied at*

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, \\ c_i(x^*) &= 0, \quad i \in \mathcal{I} \\ d_i(x^*) &\geq 0, \quad i \in \mathcal{E} \\ \lambda_i^* &\geq 0, \quad i \in \mathcal{E} \end{aligned} \quad (16)$$

$$\lambda_i^* d_i(x^*) \geq 0, \quad i \in \mathcal{E} \cup \mathcal{I}. \quad (17)$$

The conditions [16] are often known as the Karush–Kuhn–Tucker conditions, or KKT conditions for short. The conditions [17] are complementarity conditions; they imply that either constraint i is active or $\lambda_i^ = 0$, or possibly both.*

3.1. The Penalty Methods

The penalty methods are used to transform constrained optimisation problems into unconstrained ones by adding penalty terms to the objective function. In particular, considering [14], we introduce the penalty function

$$Q(x, \mu) = f(x) + \mu \sum_{i \in \mathcal{I}} \psi(c_i(x)) + \mu \sum_{i \in \mathcal{E}} \phi(d_i(x))$$

where μ is the penalty parameter and ψ, ϕ are functions that penalize constraint violations. The various methods differ based on the choice of functions ψ and ϕ . Generally, smooth functions are chosen, and the penalty function is minimized over

an increasing sequence of penalty parameter values μ until sufficient accuracy of the constrained problem is achieved. Another possibility is to use exact penalty functions, sacrificing the smoothness of such functions. In literature, one defines a penalty function to be *exact* when, due to particular choices of their penalty parameters, the minimisation with respect to x can yield to the exact solution of the nonlinear programming problem; see for example [14, 20]. It is an advantageous property because we get a penalty method which depends less on the scheme to update the penalty parameter. One can tackle the problem from several points of view. We concentrate on three methods:

1. The *quadratic penalty* methods add a multiple of the square of the violation of each constraint to the objective function to penalize solutions that violate constraints. The quadratic-penalty-functions are smooth and differentiable; they facilitate analysis and computation but they may require very large penalty parameter values to enforce feasibility.
2. In *nonsmooth exact penalty* methods, the original constrained problem is replaced by an unconstrained problem, instead of a sequence, providing exact solutions through the use of penalty functions without the need to increase the penalty parameter. It introduces penalty terms that are exact, i.e., solutions to the penalized problem which match those of the original constrained problem. In the end, one deals with a single unconstrained minimization problem, although it being nonsmooth may add complications. An example of these penalty functions is the ℓ_1 function. With the advent of large datasets and complex models, such as those in machine learning, nonsmooth optimisation has become increasingly relevant. Algorithms for nonsmooth optimisation have seen significant development see, for example [18].
3. The *method of multipliers* or *augmented Lagrangian method* combines the Lagrangian function with quadratic penalty terms. It can be seen as a different kind of exact penalty approach. In this case, one avoids the ill-conditioning, which is inherent in the quadratic-penalty-function, using explicit Lagrange multiplier estimates. As it converges quickly, it can handle both equality and inequality constraints but it requires solving unconstrained problems at each iteration.

We recall also the *interior penalty functions*; these functions add a penalty term to the original objective function to "punish" solutions that approach the boundaries of the constraints, pushing the solutions towards the interior of the feasible region. The primary goal of interior penalty functions is to keep the solution within the feasible region, avoiding the boundaries of the constraints. Therefore, they introduce a penalty term that becomes infinite as the solution approaches the boundaries of the constraints. A common example of an interior penalty function is the *log barrier function*, where the penalty is based on the logarithm of the constraints.

Interior penalty functions are particularly useful when one wants to ensure that the solution remains within the feasible region. This can be important in appli-

cations such as linear and quadratic programming.

A major disadvantage of these methods is the requirement of an initial solution to already be within the feasible region. Moreover, since the penalty term becomes infinite at the boundaries of the constraints, the objective function can become very steep, making the search for a solution more challenging.

The Quadratic Penalty Method

The quadratic penalty methods substitute a constrained optimisation problem with a single function, the quadratic-penalty-function, that consists of the original objective function and one additional term for each constraint, which is positive when the current point x violates that constraint and zero otherwise; that is the reason why it is called *quadratic penalty method*.

A classical way to approach the problem is to define a sequence of such penalty functions, in which one multiplies the penalty terms for the constraint violations by a positive coefficient. The larger this coefficient is, the more severely one penalises constraint violations; consequently, the minimiser of the penalty function is pushed closer to the feasible region for the constrained problem.

The simplest example of quadratic-penalty-function is when the penalty terms are the squares of the constrained violations. Let us analyse first the equality-constrained-problem, i.e.:

$$\min_x f(x) \quad \text{subjected to } c_i(x) = 0, \quad i \in \mathcal{I}. \quad (18)$$

One can define the quadratic-penalty-function $Q(x; \mu)$ as follows:

$$Q(x; \mu) := f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} c_i^2(x), \quad (19)$$

where $\mu > 0$ is named the *penalty parameter*. As previously said, the larger μ is (i.e. $\mu \rightarrow \infty$), the more severely one penalises constraint violations; consequently, the minimiser of the penalty function is pushed closer to the feasible region for the constrained problem. Therefore, it is logical to consider a sequence of values $\{\mu_k\}$ with $\mu_k \rightarrow \infty$, when $k \rightarrow \infty$, and to search for the approximate minimiser x_k of $Q(x; \mu_k)$ for each k . Thanks to the regularity of the terms in equation [19], one can use unconstrained optimisation techniques to seek x_k .

For general constrained optimisation problem [14], the quadratic-penalty-function is defined as follows:

$$Q(x; \mu) := f(x) + \frac{\mu}{2} \sum_{i \in \mathcal{I}} c_i^2(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} ([d_i(x)]^-)^2, \quad (20)$$

where $[y]^-$ denotes $\max\{-y, 0\}$.

OBSERVATION. Due to this definition, Q may be less regular than the objective and constrained functions.

We can write a general framework for the algorithm which uses the quadratic-penalty-function as follows:

Algorithm 2: Quadratic Penalty Method

Input: Set $\mu_0 > 0$, a non-negative sequence $\{\tau_k\}$ with $\tau_k \rightarrow 0$ and a starting point x_0^s .
Output: x_k, μ_k .
for $k = 0, 1, 2, \dots$ **do**
 Find an approximate minimiser x_k of $Q(\cdot; \mu_k)$, starting at x_k^s and terminating when $\|\nabla_x Q(x; \mu_k)\| \leq \tau_k$;
 if *final convergence test is satisfied* **then**
 | **stop** with approximate solution x_k ;
 Choose new penalty parameter $\mu_{k+1} > \mu_k$;
 Choose new starting point x_{k+1}^s ;
return *the current trial solution*.

One can choose the parameter sequence $\{\mu_k\}$ accordingly to the difficulty of minimising the penalty function at each iteration. For instance, the more expensive the minimisation of $Q(x; \mu_k)$ tends to be, the less larger μ_{k+1} is in relation to μ_k .

OBSERVATION. Unfortunately, it could happen that the iterates may move away from the feasible region when the penalty parameter is not large enough; therefore, the stop rule $\|\nabla_x Q(x; \mu_k)\| \leq \tau_k$ may not be satisfied. In order to avoid this, one has to include safeguards that increase the penalty parameter when either the decreasing speed of the constraint violation is not high enough or when the iterates appear to be diverging.

OBSERVATION. Although $Q(x; \mu_k)$ is smooth when only equality constraints are considered, it is important to note that as μ_k becomes large, the minimization of $Q(x; \mu_k)$ becomes challenging. As a matter of fact, the Hessian $\nabla_{xx}^2 Q(x; \mu_k)$ becomes arbitrary ill conditioned near the minimiser. As a result, even if we could employ algorithms for unconstrained minimization (such as Newton's, quasi-Newton's methods, or conjugate gradient), their performance might be suboptimal. To mitigate this, a judicious choice of the starting point x_{k+1}^s can be made or one could set $x_{k+1}^s = x_k$ and choose μ_{k+1} to be slightly larger than μ_k .

For more details on convergence theorems and on how to reformulate the problem when there is ill conditioning for the Hessian see [13, 16].

Nonsmooth Penalty Functions.

A well known nonsmooth penalty function used in general non linear programming problems is the ℓ_1 *penalty function*, which can be written as

$$Q_1(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{I}} |c_i(x)| + \mu \sum_{i \in \mathcal{E}} [d_i(x)]^-. \quad (21)$$

One can prove that the ℓ_1 penalty function is exact.

In more general settings, one can define exact nonsmooth penalty functions in terms of other norms. For example, one can define

$$Q(x; \mu) = f(x) + \mu \|d_{\mathcal{E}}(x)\| + \mu \|c_{\mathcal{J}}(x)\|^{-1},$$

where $\|\cdot\|$ is any vector norm and the vector functions $d_{\mathcal{E}}$ and $c_{\mathcal{J}}$ define all the equality and inequality constraints, respectively.

The most common norms used in practice are ℓ_1 , ℓ_∞ and ℓ_2 (not squared).

Note that, although Q_1 is not differentiable, one can reckon a directional derivative $D(Q_1(x; \mu); p)$ along any direction p and a *stationary point* for the penalty function $Q_1(x; \mu)$ is a point $\hat{x} \in \mathbb{R}^n$ such that $D(Q_1(\hat{x}; \mu); p) \geq 0$, for all $p \in \mathbb{R}^n$. It is important to characterize stationary points for Q_1 as one minimises the penalty function when using penalty methods.

We introduce the function measure of infeasibility as

$$h(x) = \sum_{i \in \mathcal{J}} |c_i(x)| + \mu \sum_{i \in \mathcal{E}} [d_i(x)]^-. \quad (22)$$

One can write the following algorithm:

Algorithm 3: Classic ℓ_1 Penalty Method

Input: Set $\mu_0 > 0$, tolerance $\tau > 0$ and a starting point x_0^s .

Output: x_k approximation of the minimiser of Q_1 .

for $k = 0, 1, 2, \dots$ **do**

 Find an approximate minimiser x_k of $Q_1(x; \mu_k)$, starting at x_k^s ;

if $h(x_k) \leq \tau$ **then**

stop with approximate solution x_k ;

 Choose new penalty parameter $\mu_{k+1} > \mu_k$;

 Choose new starting point x_{k+1}^s ;

return the current trial solution.

The minimisation of $Q_1(x; \mu_k)$ is made difficult by the nonsmoothness of the function.

Augmented Lagrangian Method.

In the following paragraph, we explain the approach named as the *method of multipliers* or the *augmented Lagrangian method*. It differs from the previous ones because, with the introduction of explicit Lagrange multiplier estimates into the function to be minimised, ill conditioning is less possible.

Moreover, the augmented-Lagrangian-function largely preserves smoothness and one can implement this method using unconstrained optimisation methods.

Let us study the equality-constrained-problem [18]. We know that the quadratic-penalty-function defined in [19] penalises constraint violations by squaring the infeasibilities and scaling them by $\frac{\mu}{2}$. Nevertheless, one can notice that the approximate minimizers x_k of $Q(x; \mu_k)$ do not satisfy the feasibility conditions $c_i(x) = 0$, $i \in \mathcal{J}$. Instead, they are perturbed so to have that $c_i(x_k) \rightarrow 0$, when $\mu_k \rightarrow \infty$. Once noticed that, one could be incline to change the function $Q(x; \mu_k)$ to avoid this systematic perturbation.

The augmented-Lagrangian-function $\mathcal{L}_A(x, \lambda; \mu)$ succeeds in this thanks to the inclusion of an explicit estimate of the Lagrange multipliers λ , based on the perturbation done on $c_i(x_k)$, previously mentioned.

DEFINITION. We define the *augmented-Lagrangian-function* $\mathcal{L}_A(x, \lambda; \mu)$ as

$$\mathcal{L}_A(x, \lambda; \mu) := f(x) - \sum_{i \in \mathcal{J}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{J}} c_i^2(x). \quad (23)$$

The augmented Lagrangian is different from the standard Lagrangian [15] due to the presence of the squared terms; on the other hand, it is different from the quadratic-penalty-function [19] due to the presence of the summation term involving λ . Consequently, it can be seen as the result of combining the Lagrangian function with the quadratic-penalty-function.

Let us now make some considerations in order to build an algorithm that fixes the penalty parameter μ to some value $\mu_k > 0$ at its k -th iteration; moreover it fixes λ at the current estimate λ^k and minimises in relation to x . Thanks to the optimality conditions for unconstrained problems, one has

$$0 \approx \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k) = \nabla f(x_k) - \sum_{i \in \mathcal{J}} [\lambda_i^k - \mu_k c_i(x_k)] \nabla c_i(x_k).$$

Using the optimality condition for unconstrained problem, it follows that

$$\lambda_i^* \approx \lambda_i^k - \mu_k c_i(x_k), \quad \forall i \in \mathcal{J},$$

where λ^* is the optimal Lagrange multiplier vector. Rewriting the previous expression, one has

$$c_i(x_k) \approx -\frac{1}{\mu_k} (\lambda_i^* - \lambda_i^k), \quad \forall i \in \mathcal{J}.$$

Therefore, when $\lambda^k \rightarrow \lambda^*$, the infeasibility in x_k is much smaller than $\frac{1}{\mu_k}$ and not proportional as it would happen with the quadratic-penalty-function. Consequently, one can deduce a formula which improves the current estimate λ^k of the Lagrange multiplier vector, by the use of the approximate minimizer x_k just calculated. It follows:

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_k), \quad \forall i \in \mathcal{J}. \quad (24)$$

OBSERVATION. One can show that this method converges without increasing μ indefinitely. Therefore, one does not encounter a big problem with ill conditioning, so the choice of starting points x_{k+1}^s is less critical.

Algorithm 4: Augmented Lagrangian Method-Equality Constraints

Input: Set $\mu_0 > 0$, tolerance $\tau_0 > 0$ and starting points x_0^s and λ^0 .
Output: Approximation of the minimiser of $\mathcal{L}_A$.
for $k = 0, 1, 2, \dots$ **do**
 Find an approximate minimiser x_k of $\mathcal{L}_A(\cdot, \lambda^k; \mu_k)$, starting at x_k^s ;
 and terminate when $\|\nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k)\| \leq \tau_k$;
 if a convergence test for the problem [18] is reached **then**
 stop with approximate solution x_k ;
 Using formula [24], update the Lagrange multipliers, so to obtain λ^{k+1} ;
 Choose new penalty parameter $\mu_{k+1} \geq \mu_k$;
 Choose new starting point $x_{k+1}^s = x_k$;
 Select a new tolerance τ_{k+1} ;
return the current trial solution.

One can prove two main properties of the augmented Lagrangian:

1. When one exactly knows the Lagrange multiplier vector λ^* , the solution x^* of the problem [18] is a strict minimizer of $\mathcal{L}_A(x, \lambda^*; \mu)$ for all μ sufficiently larger. Moreover, in the practical case that the exact value of λ^* is unknown, a good estimate of x^* by minimizing $\mathcal{L}_A(x, \lambda; \mu)$ can be obtained even when μ is not particularly large, although one needs that λ is a reasonably good estimate of λ^* ;
2. When $\lambda \neq \lambda^*$, one can impose some conditions in order to prove the existence of a minimizer of $\mathcal{L}_A(x, \lambda; \mu)$ which lies close to x^* ; furthermore one can provide error bounds on both x_k and the updated multiplier estimate λ^{k+1} , obtained at the k -th iteration from solving the problem.

From this result, one deduces that the augmented Lagrangian method provides two ways to improve the accuracy of x_k , while the quadratic penalty method gives only one way. As a matter of fact, the former states that x_k will be close to x^* if λ_k is accurate *or* if the penalty parameter μ_k is large; the latter states that x_k will be close to x^* only by increasing μ_k .

Conclusion. In literature, one can find many examples of practical uses of the quadratic penalty approach when the number of constraints is small. This approach is especially used as an important mechanism to regularise other algorithms, such as the sequential quadratic programming.

On the other hand, the augmented Lagrangian method is generally the one chosen due to its simplicity. As a matter of fact, the subproblems are not likely to be more difficult to solve and introducing multipliers estimates reduces the chance of needing large values of μ to reach good feasibility and accuracy. Doing that, ill conditioning of the subproblems is avoided.

In relation to the ℓ_1 penalty function, this approach has lately been developed. It has

proven to be particularly useful to solve difficult problems, such as mathematical programs with complementarity constraints, which do not satisfy standard qualifications. In this case, one transform these problematic constraints as a penalty term, extending the range of applicability of these other methods.

3.2. Nonlinear Constrained optimisation

In this section, we analyse algorithms which find local solutions for the problem [14] in the case of an objective function that is nonlinear, subjected to a set of constraints that can also be nonlinear. The main difficulty in this situation lies in deciding which of the inequality constrained can be considered active at the solution and which inactive. The foundation of active-set methods is to start by making a guess of the optimal active set Ω^* , therefore the set of constraints that are satisfied as equalities at a solution. If we name the guess set the *working set* W , then we can solve a problem in which we impose the constraints in the working set as equalities, while we ignore the constraints that do not belong to W . Next, we study whether there is a possible choice of Lagrange multipliers so that the solution x^* obtained for this particular W satisfies the KKT conditions [16]. If this is the case, x^* is accepted as a local solution of the problem [14]. On the other hand, a different choice of W is made and we repeat the process.

It is important to notice that one can choose the working set W in several ways, to be precise $2^{|\mathcal{E}|}$. As a matter of fact, one can distinguish only two possible choices for each $i \in \mathcal{E}$: either it belongs to W or it does not. Therefore, it is easy to note that the number of possible working sets grows exponentially in relation to the number of inequalities and it is known as *combinatorial difficulty* of nonlinear programming. As a consequence, designing an algorithm that takes into account all the possible choices of W is unthinkable.

Furthermore, thanks to some educated guesses for the working set based on the properties of the functions which define the problem, one can reduce the number of choices of W , eliminating the ones which would lead nowhere. For a practical example, see [17].

Reducing the number of variables

In order to obtain a simpler problem with fewer degree of freedom, it is common to take advantages of the constraints so to reduce the number of variables. It is important to point out that, when dealing with elimination techniques, precaution is needed otherwise one might alter the problem or introduce ill conditioning. As a matter of fact, when one uses nonlinear equations to reduce the number of variables, they may create errors which are not easy to trace. Most optimisation algorithms tend to avoid nonlinear elimination. Instead, one usually linearises the constraints and then applies elimination techniques to reduce the difficulty of the problem.

We introduce now a procedure to eliminate variables using linear constraints. Let us

consider the following problem

$$\min f(x) \quad \text{subjected to} \quad Ax = b, \quad (25)$$

where A is an $m \times n$ matrix, with $m \leq n$ and with full row rank. Therefore, there is a subset of m columns of A which is linearly independent. We can consider a new matrix B made of these $m \times m$ and then consider a $n \times n$ permutation matrix P , which puts these columns to the first m column positions in A , so it follows that:

$$AP = [B|N],$$

where $N \in \mathbb{R}^{m \times (n-m)}$ denotes the $n - m$ remaining columns of A . Let us define the following subvectors $x_B \in \mathbb{R}^m$ and $x_N \in \mathbb{R}^{n-m}$ as:

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = P^T x,$$

and we define x_B as the *basic variables* and B as the *basis matrix*. As one has $PP^T = I$, one can formulate the constraints $Ax = b$ as follows:

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

It is easy to deduce that we can write the basic variables as:

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

Then, a feasible point for the constraints $Ax = b$ can be computed selecting any value of x_N and writing x_B using the previous formula. Consequently, we can solve

$$\min_{x_N} f \left(P \begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix} \right).$$

that it is an equivalent unconstrained formulation for the problem [25].

This method is known as the *simple elimination of variables*.

One can interpret the simple elimination of variables in a compelling way. We can assume that $P = I$, which means that the basic columns appear in the first m positions. From the previous formulas, one can deduce that we can write any feasible point x for the linear constraints as

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = x = Yb + Zx_N,$$

where

$$Y = \begin{bmatrix} B^{-1} \\ 0_{(n-m) \times m} \end{bmatrix} \in \mathbb{R}^{n \times m}, \quad Z = \begin{bmatrix} -B^{-1}N \\ I_{(n-m) \times (n-m)} \end{bmatrix} \in \mathbb{R}^{n \times (n-m)}.$$

One can notice that Z has $n - m$ linearly independent columns and that it satisfies $AZ = 0$. Consequently, Z can be seen as a *basis for the null space* of A .

Rewording, thanks to the simple elimination technique, one can write feasible points as the sum of a particular solution of $Ax = b$ plus a displacement along the null space of constraints. From the previous relations, we can obtain the particular Yb solution by holding $n - m$ components of x at zero, while relaxing the other m components until they reach the constraints.

OBSERVATION. Although simple elimination is inexpensive, it can lead to numerical instabilities. As a matter of fact, choosing the best basis can be challenging as one might encounter the need of an excessively large coordinate relaxation step. To avoid this situation, the particular solution Yb can be defined as the minimum-norm step to the constraints.

Generalising the simple elimination of variables. Let us now consider the matrices $Y \in \mathbb{R}^{n \times m}$ and $Z \in \mathbb{R}^{n \times (n-m)}$ with the following properties:

$$[Y|Z] \in \mathbb{R}^{n \times n} \text{ is nonsingular, } AZ = 0.$$

Then, any solution of the linear constraints $Ax = b$ can be rewritten as:

$$x = Yx_Y + Zx_Z,$$

for some vectors $x_Y \in \mathbb{R}^m$ and $x_Z \in \mathbb{R}^{n-m}$. If one substitutes this in the constraints $Ax = b$, one has

$$Ax = (AY)x_Y = b;$$

thanks to the nonsingularity of AY , one can write x_Y explicitly

$$x_Y = (AY)^{-1}b.$$

It leads to the fact that any vector of the form

$$x = Y(AY)^{-1}b + Zx_Z$$

has the property of satisfying the constraints $Ax = b$ for any choice of $x_Z \in \mathbb{R}^{n-m}$. Consequently, we have achieved a different formulation for the problem [25] as an unconstrained problem

$$\min_{x_Z} f(Y(AY)^{-1}b + Zx_Z).$$

As one can notice from this formulation, it would be preferable to have Y in such a way that AY is as well conditioned as possible. One can use the QR factorization* of A^T , which can be expressed as

$$A^T \Pi = [Q_1 \quad Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $[Q_1 \quad Q_2]$ is orthogonal. One can see that the submatrices Q_1 and Q_2 have orthonormal columns and are of dimension $n \times m$ and $n \times (n-m)$; R , instead, is $m \times m$ upper triangular and nonsingular and Π in an $m \times m$ permutation matrix. Then one can define

$$Y = Q_1, \quad Z = Q_2,$$

*A QR decomposition is a decomposition of a matrix A into a product $A = QR$ of an orthonormal matrix Q and an upper triangular matrix R .

in order to have that the columns of Y and Z form an orthonormal basis of $\mathbb{R}^n$. From the previous considerations, it follows that

$$AY = \Pi R^T, \quad AZ = 0.$$

It follows that Y and Z have now the desired properties. Therefore, one can express any solution of $Ax = b$ as

$$x = Q_1 R^{-T} \Pi^T b + Q_2 x_Z,$$

for some vector x_Z . In this setting we have the advantage that one carries out the computation of $R^{-T} \Pi^T b$ inexpensively, paying only the cost of a single triangular substitution.

OBSERVATION. If one carries out a simple computation, one can deduce that the particular solution $Q_1 R^{-T} \Pi^T b$ can be expressed as $A^T (AA^T)^{-1} b$. This vector can be seen as the solution of the following problem

$$\min \|x\|^2 \quad \text{subjected to} \quad Ax = b;$$

that is known as the *minimum-norm* solution of $Ax = b$.

As the main cost associated with elimination via orthogonal basis is computing the QR factorization, this reduction strategy is highly suitable for numerical stability. Unluckily, in the cases where A is large and sparse, computing the QR factorization can be more costly than simple elimination reduction strategy. In literature, one can find other elimination strategies that compromise these two techniques.

3.3. Merit Functions and Filters

Sometimes an algorithm produces a step that, on one side, lowers the objective function, but, at the same time, highers the violation of the constraints. Should the step be accepted or not? An equilibrium can be obtained using merit functions and filters. In the general setting of a constrained optimisation algorithm, one accepts a step p only if it sufficiently reduces the so-called merit function Q or when it is accepted by the filter. The real advantage of using the merit function becomes apparent in complex and/or nonlinear problems, where the analytical approach becomes impractical and numerical methods are employed. In such situations, the merit function and the filters help guide the optimisation algorithm towards a solution that not only minimizes the objective but also satisfies the constraints.

Merit functions. A merit function is a scalar function designed to measure the "quality" or "merit" of a candidate solution, taking into account both the objective function value and the degree to which the constraints are satisfied (or violated). The goal is to minimize the merit function, and a solution that minimizes the merit function will also be a solution to the original constrained optimisation problem. In literature, one can notice that the natural choice for the merit function is the

objective function in unconstrained problem. This can be true even for constrained optimisation when the starting point and all subsequent iterations satisfy all the problem's constraints; for example see [15]. However, when iterations are allowed to violate the constraints, methods are needed to evaluate the quality of the steps and the iterations. In this particular case, a merit function is built using the objective combined with measures of constraint violation.

The ℓ_1 penalty function [21] is a potential choice for the merit function. In this case μ , the *penalty parameter*, determines the weight given to constraint satisfaction relative to the minimization of the objective. It is important to point out that the ℓ_1 merit function Q_1 is exact but not differentiable.

DEFINITION. A merit function $Q(x; \mu)$ is *exact* if there is a positive scalar μ^* such that for any $\mu > \mu^*$, any local solution of the nonlinear programming problem [14] is a local minimiser of $Q(x; \mu)$.

One could prove that, if some assumptions are met, the ℓ_1 merit function $Q_1(x; \mu)$ is exact and its threshold value μ^* is given by

$$\mu^* = \max\{|\lambda_i^*| \text{ with } i \in \mathcal{I} \cup \mathcal{E}\},$$

where λ_i^* are the Lagrange multipliers associated with an optimal solution x^* .

In case of equality-constrained-problems, where we define $c(x)$ as the vector of the equality constraints, one could choose as merit function the exact ℓ_2 defined by

$$Q_2(x; \mu) = f(x) + \mu \|c(x)\|_2,$$

which is not differentiable and its derivative is not defined at x so that $c(x) = 0$.

In order to obtain merit functions with smoothness and exactness, one has to add terms to the merit functions. In relation to equality-constrained-problems, one has the *Fletcher's augmented Lagrangian*, which is defined as

$$Q_F(x; \mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2} \mu \sum_{i \in \mathcal{E}} c_i(x)^2,$$

where $\mu > 0$ is the penalty parameter and

$$\lambda(x) = [A(x)A(x)^T]^{-1} A(x) \nabla f(x),$$

where $A(x)$ is the Jacobian matrix of $c(x)$.

OBSERVATION. It can be proven that this merit function has some interesting theoretical properties; however, it is not often used due to expense of solving $\lambda(x)$.

Another possible example of merit function is given by the *standard augmented Lagrangian* in x and λ , which for the equality-constrained-problems is defined as follows

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \lambda^T c(x) + \frac{1}{2} \mu \|c(x)\|_2^2.$$

OBSERVATION. The standard augmented Lagrangian is not considered a merit function in the sense that a solution (x^*, λ^*) of the nonlinear programming is usually only a stationary point, not a minimiser. One can modify μ and λ in some sequential quadratic programming methods in order to use $\mathcal{L}_A$ as a merit function.

In the case of non differentiable merit functions, one might use, when they exist, their directional derivatives $D(Q(x; \mu); p)$, such as in the case of the ℓ_1 and ℓ_2 merit functions that are not differentiable but they have a directional derivative. For example, in a line search method, one can deduce that the step length parameter $\alpha > 0$ must be small enough to ensure that the following inequality is satisfied:

$$Q(x + \alpha p; \mu) \leq Q(x; \mu) + \eta \alpha D(Q(x; \mu); p),$$

for some $\eta \in (0, 1)$; in the trust region method, one often uses a quadratic model $q(p)$ to reckon the value of the merit function Q after a step p . In this case, the sufficient decrease condition can be written as

$$Q(x + p; \mu) \leq Q(x; \mu) - \eta (q(0) - q(p)),$$

for some $\eta \in (0, 1)$.

Filters. Filters are an alternative technique to merit functions for constrained problems. While a merit function combines the value of the objective function and the measure of constraint violation into a single scalar value, filters maintain a list of acceptable combinations of objective values and measures of constraint violation. Let us consider the measure of infeasibility written in the following way, which is similar to the definition in [22] with the difference of having $\mu = 1$:

$$h(x) = \sum_{i \in \mathcal{J}} |c_i(x)| + \sum_{i \in \mathcal{E}} [d_i(x)]^-.$$

The problem is to minimize both $f(x)$ and $h(x)$. While merit functions combine both issues into a single minimization problem, filters allow the two objectives to be addressed separately. In brief, filter methods accept a trial step x^+ as a new iterate if the pair $(f(x^+), h(x^+))$ is not "*dominated*" by a previous pair $(f_l, h_l) = (f(x_l), h(x_l))$ generated by the algorithm. More precisely,

- DEFINITION.**
- a. A pair (f_k, h_k) is said to *dominate* another pair (f_l, h_l) if both $f_k \leq f_l$ and $h_k \leq h_l$.
 - b. A *filter* is a list of pairs (f_l, h_l) such that no pair dominates any other.
 - c. An iterate x_k is said to be *acceptable* to the filter if (f_k, h_k) is not dominated by any pair in the filter.

It is common to add (f_k, h_k) to the filter and delete any pairs that are dominated by (f_k, h_k) , when an iterate x_k is acceptable to the filter. In *line search methods*,

when a trial step $x^+ = x_k + \alpha_k p_k$ gives a pair (f^+, h^+) which is considered acceptable to the filter, one usually sets $x_{k+1} = x^+$; otherwise one performs a backtracking line search. However, in the case of *trust-region methods*, if the step is evaluated as not acceptable to the filter, one reduces the trust region and computes a new step.

If one has the goal of reaching global convergence and good practical performance, several adjustments to this filter technique need to be performed. Firstly, it is important not to accept a point whose (f, h) pair is very near to the current pair (f_k, h_k) or to another pair in the filter. In order to obtain this, one has to modify the acceptability criterion and a sufficient decrease condition needs to be imposed. We consider a trial iterate x^+ to be acceptable to the filter if, for all pairs (f_j, h_j) in the filter, one has that

$$f(x^+) \leq f_j - \beta h_j \quad \text{or} \quad h(x^+) \leq h_j - \beta h_j,$$

with $\beta \in (0, 1)$. In practical using, with $\beta = 10^{-5}$, this method is considered to be effective; however, if one has the purpose of analysis, one usually modifies the first inequality with

$$f(x^+) \leq f_j - \beta h^+.$$

One may encounter some problematic aspects of the filter method. It may occur that the search directions generated by line search methods require arbitrarily small steplengths α_k so to be accepted by the filter. When this happens, the algorithm might stall and therefore fail. If a steplength, generated by a line search algorithm, is smaller than a given threshold $\alpha_{\min}$, the algorithm enters a *feasibility restoration phase*. In a similar way, if a sequence of trial steps, generated by a trust-region algorithm, is not accepted by the filter, the trust region might reduce to an extent that the trust-region subproblem becomes infeasible. Even in this case, one has that the algorithm enters a *feasibility restoration phase*. In detail, the only aim of the feasibility restoration phase is to perform a reduction on the constraint violation; in other words, we search an approximate solution to the problem $\min_x h(x)$.

Although one might notice that $h(x)$ is not smooth, there are several techniques to minimise it, thanks to smooth constrained optimisation subproblems. Let us see now a general algorithm where the iterates are generated by a trust-region method:

4. Metaheuristics

It is important to point out that some problems may be so complicated that it might not be possible to solve them finding an optimal solution. Consequently, new methods have been found to provide a general structure and strategy guidelines to solve the problem, the so-called metaheuristic methods; see for example [8, 19, 23].

DEFINITION. A *heuristic method* is a procedure that is likely to discover a very feasible solution, therefore a solution that satisfies all the problem's constraints, but not necessarily an optimal solution for the specific problem being considered. A heuristic method can be considered well-designed when it provides a solution that is

Algorithm 5: General Filter Method

Input: Set a starting point x_0^s and initial trust-region of radius Δ_0 .
Output: Approximation of the solution.
Set $k \leftarrow 0$;
Repeat until a convergence test is satisfied
if the step-generation subproblem is infeasible **then**
 | Compute x_{k+1} using the feasibility restoration phase;
else
 | Compute a trial iterate $x^+ = x_k + p_k$;
 | **if** (f^+, h^+) is acceptable to the filter
 | **then**
 | | Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
 | | Choose Δ_{k+1} so that $\Delta_{k+1} \geq \Delta_k$;
 | | Remove all pairs from the filter that are dominated by (f_{k+1}, h_{k+1}) ;
 | **else**
 | | Reject the step, set $x_{k+1} = x_k$;
 | | Choose $\Delta_{k+1} < \Delta_k$;
 | $k \leftarrow k + 1$;
return the current trial solution.

at least nearly optimal (when this is not the case, one should come to the conclusion that no such solution exists).

Moreover, they are usually developed following simple common-sense ideas to find suitable solutions. It is fundamental to carefully tailor these ideas so to describe the problem of interest.

DEFINITION. A *metaheuristic method* is a general solution method that provides a general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problems.

When one deals with challenging nonlinear programming problems, one can opt for a simple heuristic method to achieve a *local improvement procedure*. Such a procedure is organised as follows: it starts with an initial trial solution; then, at each iteration, it searches in the neighbourhood of the current trial solution to find a better trial solution.

In order to better understand it, one can visualise it as a *low-downhill procedure* that continues descending lower on the graph of the objective function (assuming the objective is minimisation) until it essentially reaches the lowest point of the valley. In general, it is known that a well built local improvement procedure is successful in converging to a local minimum; however, it does not continue in the search, but rather stops although this found local minimum is not a global one. Therefore, the main drawback in opting for this technique is converging to a local minimum and

then concluding the search. On the other hand, these techniques tend to converge faster to a local solution, especially when analysing difficult problems.

What is next? Considering the previous ideas, one needs to better structure the used approach; as a matter of fact, one should gather as much information on the problem as possible, to guide the search towards a global minimum.

DEFINITION. The *nature of metaheuristic*: a metaheuristic method is a general kind of solution that exploits the interaction between local improvement procedures and higher level strategies to build a process that is able to escape from local minima and to conduct a thorough search on a feasible region.

In summary, the *role of metaheuristic methods* is to deal with problems that are out of reach when solved using exact algorithms. For applications, see [1, 29, 30]. We analyse three different methods:

1. *Tabu Search*;
2. *Simulated Annealing*;
3. *Genetics Algorithms*.

4.1. Tabu Search

It is a well-known metaheuristic method that, based on some common-sense ideas, allows the search process not to stop in a local minimum.

In brief, it is similar to a local improvement procedure, with the only exception that it might not need that each new trial solution must be better than the proceeding trial solution. As a matter of fact, it includes a local search procedure as a subroutine, which is developed in order to fit the problem taken into consideration.

The distinctive aspect of this method is the perseverance of the search as non-improving moves to the best solutions are allowed in the neighbourhood of the local minimum. In general, it can be known as the **steepest ascent/mildest descent approach**, as at each iteration, it chooses the move that goes furthest down the valley, or, when this is not possible, it opts for a move that goes least down the valley.

One can immediately spot a possible danger when using this method: once we have moved away from a local minimum, the process will conduct our search once more in that same local minimum, and so on, going back and forth from that local minimum. Therefore, in order to avoid this, a tabu search temporarily forbids moves that would return to a solution recently visited.

DEFINITION. A *tabu list* keeps records of the forbidden tabu moves.

Moreover, one can improve this method by adopting two different strategies. Firstly, one could intensify the search, exploring more thoroughly a particular portion of the feasible region that has been shown to be promising for containing good

solutions.

Secondly, one can diversify the search by forcing it in unknown or previously unexplored areas of the feasible region.

Algorithm 6: Tabu Search

Input: Select a feasible initial trial solution

Output: Approximate the minimum, using a *stopping rule*

for each iteration do

Use an appropriate procedure to define the feasible moves into the local neighbourhood of the current trial solution;
Do not take into account any move on the current tabu list (unless it would lead to a better solution);
Choose between the remaining moves which one would provide the best solution;
Adopt this solution;
Update the tabu list;

return *the current trial solution.*

Once this algorithm has been chosen, one still needs to properly adapt it to the problem. As a matter of fact, one should define:

- the local search procedure;
- the way in which this chosen procedure defines the neighbourhood structure which distinguishes which solutions are considered immediate neighbours of any current trial solution;
- how tabu moves should be listed in the tabu list;
- what tabu move should be added to the tabu list at each iteration;
- the time of permanence of a tabu move in the tabu list;
- the stopping rule.

4.2. Simulated Annealing

It is a method that concentrates in the search for the highest hill or lowest valley (depending on the problem considered). Since the location of this hill/valley is unknown, firstly this method focuses on taking steps in random direction so that it can inspect a vast area of the feasible region. Secondly, due to the fact that the majority of the accepted steps are upwards or downwards (always depending on the problem considered), this method progressively concentrates its search in those area of the feasible region where one can find the optimum or minimum. After some time, it converges to the highest hill or lowest valley.

More specifically, at each iteration, this method passes from the current trial solution

to an immediate neighbour in the local neighbourhood of this solution. It differs from other methods in the process of the selection of this immediate neighbour to be the next trial solution.

Let

- z_c = objective function value for the current trial solution;
- z_n = objective function value for the current candidate to be the next trial solution;
- T = parameter for the tendency to accept the current candidate to be the next trial solution if this candidate is not an improvement on the current trial solution.

One can define the *move selection rule* as follows:

DEFINITION. The process of randomly choosing, among all the immediate neighbours of the current trial solution, a candidate to become the next trial solution is called as *move selection rule*. More precisely, if our goal is a minimisation, one has:

- if $z_n \leq z_c$ always accept this candidate;
- if $z_n > z_c$ accept the candidate with the following probability:

$$\text{Prob}\{\text{acceptance}\} = e^x \quad \text{with} \quad x = \frac{z_c - z_n}{T}.$$

While when our goal is a maximisation, one has:

- if $z_n \geq z_c$ always accept this candidate;
- if $z_n < z_c$ accept the candidate with the following probability:

$$\text{Prob}\{\text{acceptance}\} = e^x \quad \text{with} \quad x = \frac{z_n - z_c}{T}.$$

OBSERVATION. The value of T over time is crucial: as a matter of fact, it provides more flexibility in contrast to the tabu search method. In particular, it controls the degree of randomness in the process of allowing upwards steps (when we have a minimisation) or downwards steps (when we are in the case of a maximisation).

One can make an analogy between this definition of the *move selection rule*, especially the probability of acceptance, to a random number between 0 and 1.

- *Minimisation*: if the random number $> \text{Prob}\{\text{acceptance}\} \implies$ accept an upwards step. Otherwise reject the step.
- *Maximisation*: if the random number $< \text{Prob}\{\text{acceptance}\} \implies$ accept a downwards step. Otherwise reject the step.

Furthermore, one can notice that the formula of the *move selection rule* is based on the physical annealing process.

Another key aspect of this method is the *temperature schedule* to use. Firstly one needs to select an initial, quite large value of T ; secondly, it is important to choose an appropriate sequence of progressively smaller values of T ; lastly, one should decide the number of iterations to perform at each specific value of T .

Algorithm 7: Simulated Annealing

Input: Select a feasible initial trial solution.

Output: Approximate the minimum/optimum, using the *move selection rule*.

for each iteration do

Use the move selection rule to define the next trial solution (when none of the immediate neighbours of the current trial solution is accepted, the algorithm is ended).

Check the temperature schedule: once one has performed the scheduled amount of moves at the current value of T , decrease T to the next value in the temperature schedule and start again the iterations at this new value.

Stopping rule: when one has performed the scheduled amount of moves at the lowest value of T , in the temperature schedule, **stop**.

return the best trial solution found at any iteration (including the one for high values of T as the final solution.)

Once this algorithm has been chosen, one still needs to properly adapt it to the problem. As a matter of fact, one should define:

- the way in which one selects the initial trial solution;
- the neighbourhood structure which distinguishes which solutions are considered immediate neighbours (in the sense of being reachable with one single iteration) of any current trial solution;
- how, in the move selection rule, one should randomly select one of the immediate neighbours of the current trial solution as the next candidate;
- a temperature schedule that fits the problem.

Hybrid Algorithm. It is important to point out that one could combine advanced features of the tabu search method with simulated annealing in a successful way. For example:

1. one could apply the *strategic oscillation* feature of the tabu search to the temperature schedule of simulated annealing. As a matter of fact, this particular feature modifies the temperature schedule by lowering the temperature in a faster way than scheduled; however, then, it strategically changes the temperature values back and forth across levels where the best solution were found;

2. one could apply the *candidate-list strategies* feature of the tabu search to the move selection rule of simulated annealing. In details, one would look through several candidates to verify if an improving move can be found before applying the randomised rule used to accept or reject the current candidate as the next trial solution.

4.3. Genetic Algorithms

It is known that genetic algorithms are used with significant results when there is the need to explore several parts of the feasible region, and gradually approaching the best feasible solutions. The theory of evolution and the survival of the fittest are the basic concepts on which genetic algorithms are founded. Let us see how.

Natural Selection: one can draw an analogy between feasible solutions for a studied problem and members of a specific species, in the sense that the value of the objective function measures the fitness of each member. In addition, one analyses an entire population of trial solutions. For each iteration or generation of a genetic algorithm, we can consider that the *current population* is made of the set of trial solutions, which are at that moment analysed. One can see these trial solutions as the current living members of the species. As in nature, some of the youngest members of the population (in particular the fittest members) survive into adulthood and become, paired randomly, *parents*. Then, they have *children*, who can be seen as new trial solutions, and who share some of the features, or biologically speaking genes, of both parents. Since there is a greater probability of the fittest members of the population to become parents, it is known that genetic algorithms tend to produce *improving populations* of trial solutions as it evolves. Furthermore, always based on the natural way of life, *mutations* are taken into consideration and they may seldom happen; consequently, some of the children might possess some features (even desirable ones) that they do not share with either parent.

In conclusion, following a pattern similar to the "survival of the fittest" should induce a genetic algorithm to a trial solution, that is the best of any considered, which is at least nearly optimal.

Once this algorithm has been chosen, one still needs to properly adapt it to the problem. As a matter of fact, one should define:

- the size of the population;
- the criteria to select members of the current population to become parents;
- how the children's features should come from the parents' ones;
- the stopping rule to use.

Algorithm 8: Genetic Algorithms

Input: Select an initial population of feasible trial solution. Evaluate the fitness (that is the value of the objective function for each member of this current population).

Output: Approximate the minimum/optimum.

for each iteration do

Use a random process that is biased towards the fittest members of the current population; then select an even number of these members to become parents. Pair up the parents in a random way; each couple of parents gives birth to two children, which are the *new feasible solutions*, whose features or genes are a random mixture of the features of the parents, except for occasional mutations. In the case of a child being an infeasible solution, it is considered a miscarriage.

Now retain enough children of the fittest members of the current population in order to form for the next iteration the new population, which will be of the same size. Evaluate the fitness for each new member of this new current population.

Stopping rule: use some stopping rule (such as a fixed number of iterations);

return the best trial solution.

5. Conclusions

In this survey, we have explored a variety of optimization techniques, emphasizing their applications. We began with an introduction of the fundamentals, setting the stage for a deeper dive into the subject matter.

In Section 2, we examined 'Trust-Region Methods,' highlighting the comparison between the Dogleg Method and two-dimensional subspace minimization. These methods offer solid frameworks for tackling challenges encountered with gradient-based methods, especially for ill-conditioned problems.

In Section 3, we examine penalty methods and their distinctions between quadratic, nonsmooth, and augmented Lagrangian techniques. These techniques are fundamental in addressing the complex aspects of nonlinear constrained optimization, and we discuss the practical implications of reducing the number of variables, as well as the role of merit functions and filters.

In Section 4, we explore Metaheuristic methods, showcasing the flexibility and adaptability of approaches such as Tabu Search, Simulated Annealing, and Genetic Algorithms. These methods surpass traditional optimization boundaries by utilizing stochastic processes and evolutionary concepts to discover near-optimal solutions in extensive and intricate search spaces.

In conclusion, we can see that the field of optimization is varied and constantly evolving. The methods described provide professionals with various ways to address a wide range of problems, with each method being particularly suited to certain types

of situations. However, challenges remain, especially regarding the ability to handle large-scale problems, the assurance of reaching the best solution, and the complexity of the necessary calculations. Future research may focus on the development of combined methods, the use of optimization techniques in emerging fields such as quantum computing, and the relentless pursuit of greater efficiency."

Moreover, the intersection of optimization with machine learning and data science presents a fertile ground for innovation. The synergy between these disciplines promises to unlock new capabilities and methodologies that will propel the field forward.

In summary, the optimization methods we have discussed in this survey give us ways to approach complex problem-solving. With the growth of computational power and the emergence of new theories, we are able to optimize more and more, which opens up promising possibilities for innovation and progress.

Acknowledgements.

The authors would like to express very special thanks to Susanna Terracini for several helpful conversations and pieces of advice.

Moreover this work is partially supported by INdAM group G.N.A.M.P.A..

This paper has been conducted during and with the support of the Italian national inter-university PhD programme in Space Science and Technology.

References

- [1] T. Akan, A. M. Anter, A. Ş. Etaner-Uyar, D. Oliva: *Engineering Applications of Modern Metaheuristics*. Springer Nature Switzerland AG, (2023).
- [2] A. Chenciner, R. Montgomery: *A remarkable periodic solution of the three-body problem in the case of equal masses*. Ann. Math. (2) 152, 881-901 (2000).
- [3] A. Chenciner, A. Venturelli: *Minima de l'intégrale d'action du problème newtonien de 4 corps de masses égales dans $\mathbb{R}^3$: orbites "hip-hop"*. Celest. Mech. Dyn. Astron. 77, 139-152 (2000).
- [4] D. Ferrario, S. Terracini: *On the Existence of Collisionless Equivariant Minimizers for the Classical n -body Problem*. John Wiley & Sons, (2005).
- [5] E. Geminiani, G. Marra, I. Moustaki: *Single- and Multiple-Group Penalized Factor Analysis: A Trust-Region Algorithm Approach with Integrated Automatic Multiple Tuning Parameter Selection*. Psychometrika, vol. 86, no. 1, 65-95 (March 2021).
- [6] J. C. Gross, P. Seshadri, G. Parks: *Optimisation with Intrinsic Dimension Reduction: A Ridge Informed Trust-Region Method*. Session: Emerging Methods, Algorithms, and Software Development in MDAO (AIAA 2020-0157).
- [7] J. Grudzien Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, Y. Yang: *Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning*. Published as a conference paper at ICLR (2022).
- [8] S. Gupta, H. Abderazek, B. S. Yıldız, A. R. Yıldız, S. Mirjalili, S. M. Sait: *Comparison of metaheuristic optimisation algorithms for solving constrained mechanical design optimisation problems*. Expert Systems with Applications, vol. 183 (November 2021).
- [9] Bert C. Rust and John R. Lewis, *Finite Difference Discretizations by Differential Quadrature Techniques*, Communications in Numerical Methods in Engineering, **15**(11), 823-833, 1999.
- [10] Yukun Huang and Adam Oberman, *Numerical Methods for the Fractional Laplacian: A Finite Difference-Quadrature Approach*, SIAM Journal on Numerical Analysis, **52**(6), 3056-3084, 2014.

- [11] Jason E. Hicken and David W. Zingg, *Summation-By-Parts Operators and High-Order Quadrature*, SIAM Journal on Scientific Computing, **33**(2), 893–922, 2011.
- [12] Kjartan Kari Gardarsson Myrdal, *Some Theoretical and Numerical Aspects of the N-Body Problem*, Bachelor's thesis, Lund University Faculty of Science, Centre for Mathematical Sciences, Mathematics, 2013.
- [13] E.S. Hillier, G.J. Lieberman: *Introduction to operations research*. Applied Mathematics Letters 17, 68.3-689 (2004).
- [14] B. Liu, W. Zhao: *New Exact Penalty Functions for Nonlinear Constrained Optimization Problems*. Hindawi Publishing Corporation Abstract and Applied Analysis, vol. 2014, Article ID 738926, (2014).
- [15] Z. Meng, Q. Hu, C. Dang, X. Yangn: *An Objective Penalty Function Method for Nonlinear Programming*, McGrawHill Higher Education (2005).
- [16] J. Nocedal, S. J. Wright: *Numerical optimisation*. Second edition, Springer Series in Operations Research, Springer (2006).
- [17] M. B. Oguntola, R. J. Lorentzen: *Ensemble-based constrained optimisation using an exterior penalty method*. Journal of Petroleum Science and Engineering, vol. 207, 109165 (December 2021).
- [18] G. Z. Oztas, S. Erdem: *A penalty-based algorithm proposal for engineering optimisation problems*. Neural Computing and Applications, 35:7635–765, (2023).
- [19] J. A. Parejo, A. Ruiz-Corte's, S. Lozano, P. Fernandez: *Metaheuristic optimisation frameworks: a survey and benchmarking*. Springer-Verlag (2011).
- [20] Y. Qian, S. Pan, L. Xiao: *Error bound and exact penalty method for optimisation problems with nonnegative orthogonal constraint*. IMA Journal of Numerical Analysis 00, 1–37, (2023).
- [21] E. Serra, S. Terracini: *Collisionless periodic solutions to some three-body problems*. Arch. Ration. Mech. Anal. 120, 305-325 (1992).
- [22] E. Serra, S. Terracini: *Noncollision solutions to some singular minimization problems with Keplerian-like potentials*. Nonlinear Anal. 22, 45-62 (1994).
- [23] K. Sörensen: *Metaheuristics—the metaphor exposed*. Intl. Trans. in Op. Res. 22 (2015) 3–18 (November 2012).
- [24] Vivina Barutello, Mattia G. Bergomi, Gian Marco Canneori, Roberto Ciccirelli, Davide L. Ferrario, Susanna Terracini, Pietro Vertechi, *Equivariant optimisation for the gravitational n-body problem: a computational factory of symmetric orbits*, *Communications in Nonlinear Science and Numerical Simulation*, **152** (2026), 109180.
- [25] Carles Simó, *New Families of Solutions in N-Body Problems*, European Congress of Mathematics, Birkhäuser Basel, 101–115, 2001.
- [26] Carles Simó, *Periodic Orbits of the Planar N-Body Problem with Equal Masses and All Bodies on the Same Path*, IoP Publishing, Bristol, 265–284, 2001.
- [27] Carles Simó, *Dynamical properties of the figure eight solution of the three-body problem*, Semantic-scholar, 2002.
- [28] G. Fusco, G.F. Gronchi, P. Negrini, *Platonic polyhedra, topological constraints and periodic solutions of the classical N-body problem*, *Inventiones mathematicae*, **185**, 283–332, 2011.
- [29] M. Vasile, M. Locatelli: *A hybrid multiagent approach for global trajectory optimisation*. Springer Science and Business Media, LLC., (2008).
- [30] M. Vasile, E. Minisci, M. Locatelli: *An Inflationary Differential Evolution Algorithm for Space Trajectory optimisation*. IEEE, vol. 15, no. 2, (2011).
- [31] Y. Wu, E. Mansimov, S. Liao, R. Grosse, J. Ba: *Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation*. 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA (NIPS 2017).

AMS Subject Classification: 90C, 65K, 70F

Isabella CRAVERO,
Department of Mathematics, University of Turin
Via Carlo Alberto 10, 10123 Torino, Italy
e-mail: isabella.cravero@unito.it

Margaux INTRONA,
Department of Physics, University of Trento
Via Sommarive 14, Povo TN, 38123, Italy.
Department of Mathematics, University of Turin
Via Carlo Alberto 10, 10123 Torino, Italy.
e-mail: margaux.introna@unitn.it

Lavoro pervenuto in redazione il 01.10.2024.