Literature Survey on Methodologies for On-Line Trajectory Generation

by

GARTEUR FM(AG14)

GARTEUR aims at stimulating and co-ordinating co-operation between Research Establishments, Industry and Academia in the areas of Aerodynamics, Flight Mechanics, Helicopters, Structure & Material and Propulsion Technology.
Literature Survey on Methodologies for On-Line Trajectory Generation

by

GARTEUR FM(AG14)

This report has been published under auspices of the Flight Mechanics Group of Responsables of the Group for Aeronautical Research and Technology in EUROpe (GARTEUR)
LIST OF AUTHORS

Dr. Ing. Vittorio Di Vito - CIRA SCpA, Italian Aerospace Research Centre
Dr. Ing. Bernardino Ricci - CIRA SCpA, Italian Aerospace Research Centre
SUMMARY

This document is a literature survey on the methodologies employed in the field of trajectory generation, with particular regard to on-line applications. It refers to the Work Package 3, “Methods selection”, of GARTEUR FM(AG14).

In this work, a wide summary and description of both numerical methods and problem formulations most commonly used to consider the optimization problem of trajectory generation are provided. Furthermore, an extended analysis of literature works referred to the problem in consideration is furnished and some relevant conclusions are emphasized.

In particular, section 2 is devoted to the analysis of numerical methods employed in solving optimal trajectory generation problems, being this analysis essential in order to analyze, in section 3, problem formulations. This is due to the fact that the numerical method is a function of the problem formulation adopted, hence the critical topic in trajectory generation problems is problem formulation.

With reference to numerical methods, both classical calculus based and modern soft computing based methods are analyzed. From a formulation point of view, geometric, dynamic and optimal control problem formulations are considered.

Many papers have been examined in order to carry out this literature survey: in section 4 several applications found in literature are described; some of these literature applications, the most significant in our opinion, are also analyzed in details in appendix.

From the brief theoretical overview performed in sections 2 and 3 and the literature analysis summarized in section 4, we have derived some observations with reference to the methodologies for on-line trajectory generation; these conclusions are reported in section 5.

In section 6, finally, a list of references is reported. This list covers all the methodologies and approaches described in this document and contains references to both papers found in literature and specialized books.

This literature survey, of course, does not cover all possible aspects of a very wide subject such as the optimal trajectory generation, nevertheless it may constitute an useful outline and description of the state-of-the-art in this field.
Contents

List of Authors i
Summary ii
List of Figures v
List of Tables vi
List of Symbols and Abbreviations vii
Distribution List ix

1. Introduction 1

2. Numerical optimization methods employed in the field of trajectory generation 3
   2.1 Numerical optimization methods for linear programming 5
   2.2 Numerical optimization methods for non linear programming 10
   2.3 Evolutionary algorithms 28
   2.4 Fuzzy methods 32
   2.5 Neural networks 36

3. Formulation based classification of on-line trajectory generation problems 43
   3.1 Geometric formulation 45
   3.2 Dynamic formulation 47
   3.3 Optimal control formulation 50
4. Review of applications in literature

4.1 Level I applications

4.2 Level II applications

4.3 Level III applications

4.4 Level IV applications

4.5 Other applications

5. Conclusions

6. References

Appendix: paper evaluation reports
LIST OF FIGURES

Figure 1 – Structure of section 2 ................................................................. 4
Figure 2 – Example of membership function [78] ........................................ 33
Figure 3 – Biological neuron ................................................................... 37
Figure 4 – Artificial neuron [79] ............................................................. 38
Figure 5 – Some activation functions [79] ................................................... 39
Figure 6 – Schematic examples of feedforward (or acyclic) NN [79] ............... 40
Figure 7 – Schematic examples of feedback (or recurrent or cyclic) NN [79] ......... 40
Figure 8 – Schematic example of feedforward multilayer NN .......................... 40
Figure 9 – Structure of section 3 ............................................................... 44
Figure 10 – Trajectory generation problems for UAVs applications ................. 57
Figure 11 – Structure of section 4 ............................................................. 58
Figure 12 – Examples of trajectories obtained from the application of EA to trajectory
  generation problems [45] ....................................................................... 64
Figure 13 – Operation and training mode for NN guidance and control function [68] .... 68
Figure 14 - Two degree of freedom design .................................................. 70
Figure 15 – An example of electrostatic potential field .................................... 80
Figure 16 - System architecture for a single UAV [12] .................................. 103
Figure 17- Example of Voronoi path [12] .................................................... 104
Figure 18 - Trajectory planning problem in the original and transformed space [63] .... 105
Figure 19 - Architecture considered in [64] ............................................... 110
LIST OF TABLES

Table 1 – Performances of NN based UAVs compared to way point based UAV [68] ............ 68
## List of Symbols and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUV</td>
<td>Autonomous Underwater Vehicle</td>
<td></td>
</tr>
<tr>
<td>BFGS</td>
<td>Broyden-Fletcher-Goldfarb-Shanno method</td>
<td></td>
</tr>
<tr>
<td>BM</td>
<td>Barrier Methods</td>
<td></td>
</tr>
<tr>
<td>CGM</td>
<td>Conjugate Gradient Method</td>
<td></td>
</tr>
<tr>
<td>CLF</td>
<td>Control Lyapunov Function</td>
<td></td>
</tr>
<tr>
<td>DMRP</td>
<td>Direct Method for Rapid Prototyping</td>
<td></td>
</tr>
<tr>
<td>DP</td>
<td>Dynamic Programming</td>
<td></td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
<td></td>
</tr>
<tr>
<td>EPF</td>
<td>Electrostatic Potential Field</td>
<td></td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
<td></td>
</tr>
<tr>
<td>FMP</td>
<td>Fuzzy Mathematical Programming</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithms</td>
<td></td>
</tr>
<tr>
<td>GARTEUR</td>
<td>Group for Aeronautical Research and Technology in EURope</td>
<td></td>
</tr>
<tr>
<td>GPM</td>
<td>Gradient Projection Method</td>
<td></td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Programming</td>
<td></td>
</tr>
<tr>
<td>KKT</td>
<td>Karush-Kuhn-Tucker</td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming</td>
<td></td>
</tr>
<tr>
<td>LQ</td>
<td>Linear Quadratic</td>
<td></td>
</tr>
<tr>
<td>MILP</td>
<td>Mixed Integer Linear Programming</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>Mathematical Programming</td>
<td></td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
<td></td>
</tr>
<tr>
<td>NLP</td>
<td>Non Linear Programming</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>Neural Networks</td>
<td></td>
</tr>
<tr>
<td>NTG</td>
<td>Nonlinear Trajectory Generation</td>
<td></td>
</tr>
<tr>
<td>OCT</td>
<td>Optimal Control Theory</td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>Penalty Methods</td>
<td></td>
</tr>
</tbody>
</table>
PMP  Pontryagin’s Maximum Principle
PVTOL  Planar Vertical TakeOff and Landing
QNM  Quasi-Newton Methods
QP  Quadratic Programming
RGM  Reduced Gradient Method
RHC  Receding Horizon Control
SAM  Surface to Air Missile
SCGRA  Sequential Conjugate Gradient Restoration Algorithm
SQP  Sequential Quadratic Programming
UAV  Unmanned Aerial Vehicle
UCAV  Uninhabited Combat Air Vehicle
DISTRIBUTION LIST

The GARTEUR FM AG14 Group.
1. **INTRODUCTION**

Main task of GARTEUR FM(AG14) project is autonomy in Unmanned Aerial Vehicles (UAVs).

Autonomous vehicles with nonlinear dynamics (such as UAVs) need to have a planned reference trajectory and a feedback controller to accomplish the task of travelling from a starting point to a goal point, thus one of the main functionalities of UAVs autonomy is the capability of on-line trajectory generation. Main purpose of this survey is try to present the state-of-the-art in the field of on-line trajectory generation.

In the analysis of methodologies for on-line trajectory generation, which is in practice an optimization problem, we underline that the focus must be centered on the problem formulations most widely used today rather than on the solution methods. This is due to the fact that the formulations are specific of on-line trajectory generation problems while solution methods are substantially the same which are used in general in other optimization problems.

The structure of this review, therefore, provides first for a preliminary analysis of the numerical methods used in general optimization problems and in particular trajectory planning and, then, the focus is centered on an attempt to classify on-line trajectory generation problems, based on the formulation used. The overview on numerical optimization methods employed in the field of trajectory generation is performed in section 2 and regards both calculus based and soft computing techniques, while the formulation based classification of on-line trajectory generation problems is reported in section 3.

This literature survey on methodologies for the on-line trajectory generation has been obtained by critical analysis of many papers, so several literature works are described, in section 4, also in order to provide examples of applications for the methodologies and formulations briefly described in section 2 and 3. Furthermore, to make more complete this document, some relevant papers are analyzed in detail in appendix.

Moreover, with reference to numerical methods, they can be simply regarded as tools for optimal trajectory generation problems solution and several techniques are currently available. These techniques, in general, refer to any optimization problem and the choice of the method to be used depends on the problem formulation considered, which is the true discriminating factor in on-line trajectory generation.
problems classification. The solution techniques usually proposed in literature may be divided between classical mathematical optimization techniques (as the methods used for linear and non-linear programming) and most recent soft computing techniques (as evolutionary algorithms, fuzzy methods and neural networks). In section 2 we will attempt to describe some benefits and deficiencies in the most used approaches.

With reference to the problem formulation, which is the main concern of this literature survey, an high level classification is between geometric and dynamic formulation. This is due to the fact that usually the planning stage has either been done from a purely geometrical point of view, without regarding the dynamic constraints of the system, or by numerical optimization including the dynamic constraints and input bounds. Furthermore, another way to mathematically formulate a trajectory generation problem is the one consisting in using the optimal control approach, which includes, in the most general case, the consideration of both dynamic constraints and controls acting on the vehicle. In order to describe these different approaches to on-line trajectory generation problem formulation, section 3 of this document is devoted to classify, describe and analyze the formulations resulting from the literature review.

The literature review has itself been performed by analyzing many papers, regarding applications as well as theoretical considerations. Hence, in section 4, several works from literature are described and, in appendix, the ones regarded as particularly significant are considered more in details.

Based on the analysis performed in this survey, in section 5 some conclusions are done and observations related to the problem of trajectory tracking are pointed out. The certainty of convergence and the computational quickness are essential requirements which must be taken into account in evaluating the effectiveness of formulations and solution methods employed in the field of on-line trajectory generation. The certainty of convergence and the time required for the computation, of course, are strongly dependent from the problem formulation considered and from the numerical method employed, so the most significant criterion to evaluate the effectiveness of formulations and solution techniques is the one based on the compliance with the earlier described requirements.

Section 6 contains a list of references, which may be regarded as an annotated bibliography and, finally, in appendix some relevant papers, resulting from the literature analysis, are analyzed.
2. **Numerical optimization methods employed in the field of trajectory generation**

Mathematical techniques used in the field of optimal trajectory planning are the usual ones employed in order to solve optimization problems in general. A survey on these techniques is a very hard task, since they are really various and many classification frameworks are possible.

The solution algorithms are function of the optimization problem formulation but a fairly general way for provide a very high level classification of these is according to the typology of the framework examined or according to the solution search method used.

Based on the framework in which a solution method operates, it is possible to distinguish between deterministic and stochastic methods. The first ones are used in the case in which the optimization problem is considered in a deterministic frame while the second ones refer to a probabilistic approach in the problem solving.

Based on the solution search employed, the mathematical methods used in order to solve optimization problems may be divided between exhaustive and iterative methods. The first ones consider a closed form of the problem solution while the second ones make use of iterative algorithms, in order to reach an approximate solution by means of step by step procedures.

Really, a third way to provide an high level classification of solution methods used in optimization problems is to distinguish between classical methods and modern methods. This is due to the fact that historically the most used solution techniques in the optimization field have been developed in order to solve classical problems, such as linear programming and non linear programming problems, hence we make use of the expression “classical methods” to refer to these techniques. On the other hand, in the late 1950s Bellman has introduced the dynamic programming theory and most recently soft computing techniques, such as genetic algorithms, fuzzy logic and neural networks, have been employed in order to solve optimization problems very difficult to be solved by means of the classical methods; we use the expression “modern methods” to indicate these techniques.

As emphasized by Betts in [1] and by Ross and Fahroo in [60], classical calculus based methods used in the field of trajectory generation may also be classified by
means of the distinction between two categories: direct and indirect methods. These papers will be described in appendix.

In this section, we briefly illustrate a description of both classical calculus based methods and soft computing techniques; to make simpler the reading, in Figure 1 a schematic structure of this section is reported.

![Figure 1 - Structure of section 2](image-url)
2.1 Numerical optimization methods for linear programming

A linear programming (LP) problem is characterized by linear functions of the unknowns: the objective function is linear in the unknowns and the constraints are linear equalities or linear inequalities in the unknowns. Any LP problem can be transformed into the following standard form [74]:

\[
\begin{align*}
\text{minimize} & \quad \mathbf{c}^T \mathbf{x} \\
\text{subject to} & \quad \mathbf{A} \mathbf{x} = \mathbf{b} \\
& \quad \mathbf{x} \geq 0
\end{align*}
\]

(1)

in which \( \mathbf{x} \) is the \( n \)-dimensional column vector of the real unknowns to be determined, \( \mathbf{c}^T \) is an \( n \)-dimensional row vector of fixed real constants (cost coefficients), \( \mathbf{A} \) in an \( m \) by \( n \) matrix of fixed real constants and \( \mathbf{b} \) is an \( m \)-dimensional column vector of fixed real constants. Notice that the terms of \( \mathbf{b} \) are always assumed non negative and the unknowns \( \mathbf{x} \) also are assumed non negative.

As well known, usually in linear programming problems it is assumed that \( n > m \), which implies that the number of unknowns exceeds the number of equality constraints, as obvious in optimization problems. Moreover, it is also assumed that the rows of the matrix \( \mathbf{A} \) are linearly independent, which means the independence of \( m \) equations. These assumptions imply that the equation system \( \mathbf{A} \mathbf{x} = \mathbf{b} \) will always have a solution and at least one basic solution; the question at this point is if the solution of the system \( \mathbf{A} \mathbf{x} = \mathbf{b} \) is feasible (i.e. satisfies also the constraints \( \mathbf{x} \geq 0 \)) and if it is optimal (i.e. satisfies the whole LP problem (1)).

The fundamental theorem of the linear programming states that with reference to the problem (1), with the assumptions that we have adopted, if there is a feasible solution, there is a basic feasible solution and, moreover, if there is an optimal feasible solution, there is an optimal basic feasible solution. Based on this theorem, the methods developed to solve LP problems consider, in the search for an optimal feasible solution, only basic feasible solutions.

Based on this observation, the main idea used in classical solution methods for solving LP problems is to proceed from one basic feasible solution to another, in such a way as to continually decrease the value of the objective function until a minimum is reached.
**Simplex method**

This idea is applied by means of the *simplex method*: starting from a basic feasible solution of the LP problem, some non basic variables are increased of a small value over zero and the values of the current basic variables are modified in order to again satisfy the linear equality constraints. If the change in the objective function due to a unit increase in the non basic variable considered and to the corresponding required changes in the values of the basic variables (which is defined *relative cost coefficient* related to the considered non basic variable) is negative, the objective function value decreases. Repeating this procedure until none relative cost coefficient is negative, the solution of the LP problem is reached. A detailed explanation of the simplex method can be found in [74].

**Dual problem**

For any linear programming problem it is possible to develop a related *dual* LP problem. The dual problem is related to the original (*primal*) problem in such a way as, if the primal LP problem requires a minimization, then the dual problem requires a maximization and vice versa, although the optimal values of the corresponding objective functions, if finite, are the same.

The dual form of the standard primal LP problem (1) is:

\[
\begin{align*}
\text{maximize} & \quad \lambda^T b \\
\text{subject to} & \quad \lambda^T A \leq c^T 
\end{align*}
\]

being \( \lambda \) the unknown variables vector of the dual problem (\( \lambda^T \) is an \( m \)-dimensional row vector). The transformation from the primal to the dual problem can be found in [74].

The usefulness of introduction of the dual LP problem results from the fact that it is possible to demonstrate that:

- if \( \mathbf{x} \) and \( \lambda \) are feasible for (1) and (2) respectively, then \( c^T \mathbf{x} \geq \lambda^T \mathbf{b} \);
- if \( \mathbf{x}_o \) and \( \lambda_o \) are feasible for (1) and (2) respectively and \( c^T \mathbf{x}_o = \lambda_o^T \mathbf{b} \), then they are optimal for their respective problems.
This implies that, if a pair of feasible vectors can be found to primal and dual problems with equal objective values, then these are both optimal; this issue can be used in order to develop methods for solving LP problems.

**Integer and mixed integer linear programming**

A LP problem may contains all integer variables, in which case the problem is defined as integer linear programming (ILP) problem. A such problem is expressed in standard form as:

$$\begin{align*}
\text{minimize} & \quad \mathbf{c}^T \mathbf{x} \\
\text{subject to} & \quad \mathbf{A} \mathbf{x} = \mathbf{b} \\
& \quad \mathbf{x} \geq 0, \text{ integer.}
\end{align*}$$  \hspace{1cm} (3)

Another alternative is constituted from a mixed integer linear programming (MILP) problem, in which some variables are integer and the others are real. A MILP problem is expressed in standard form as:

$$\begin{align*}
\text{minimize} & \quad \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y} \\
\text{subject to} & \quad \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} = \mathbf{b} \\
& \quad \mathbf{x} \geq 0, \text{ integer} \\
& \quad \mathbf{y} \geq 0.
\end{align*}$$  \hspace{1cm} (4)

In general, integer or mixed integer problems are more complicated than LP problems in the real domain (1). Besides, differently from the case of real LP, in the case of integer or mixed integer problems the effectiveness of the solution method is related to the problem formulation.

A method for solution of ILP problems consists in the use of the same techniques developed in the field of real LP problems, by means of the linear relaxation. Starting from an integer linear programming problem in the standard form (3), the linear relaxation of this problem is:
\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0, \text{ real}
\end{align*}
\]  

(5)

in which, practically, the constraint of \(x\) to be integer has been eliminated. It is possible to demonstrate that, since the objective function in the relaxed problem (5) is the same as in the integer problem (3), if the optimal solution of the relaxed problem is integer, then it is the optimal solution for the ILP problem also. Similar observations can be made for MILP problems too, hence sometimes it is possible to solve an integer or mixed integer linear programming problem by means of solution techniques developed with reference to real LP problems. It’s worth to emphasize yet that in general it’s wrong to solve an ILP problem by solving the relaxed problem and rounding off the solution, if it is not integer: the use of the linear relaxation technique means to an exact solution for the integer linear programming problem only if the optimal solution of the relaxed formulation is integer! This implies that the linear relaxation technique is not always practical.

Another method employed in order to solve integer linear programming problems is the Gomory technique [75] but nowadays it is not much used, since this technique is very slow in the convergence to the solution, hence its effectiveness is weak.

Once again, we emphasize that the problem formulation is fundamental in order to solve optimization problems, therefore we regard the classification of the on-line trajectory generation methods based on the problem formulation as the main concern of this literature survey. Notice in this connection that linear relaxation and Gomory technique may be regarded as methods for improving the initial formulation of the problem to be solved, which once more underlines that the problem formulation is essential.

The most employed techniques for the solution of integer LP problems are the branch & bound algorithm, which is based on an iterative procedure, and the Lagrangian relaxation; these methods can be found in [75].

Example of application of MILP to aircraft trajectory planning are provided, for instance, in [31] and [32]; these papers will be described in section 4.
Combined use of mixed integer linear programming and receding horizon control

The MILP techniques have been applied also in association with receding horizon control (RHC), as in [34]. In section 4, this work will be discussed but we emphasize here some theoretical considerations regarding the methodology employed in the paper and some concepts with reference to the receding horizon control approach.

Improvements in UAV capabilities make it possible for UAVs to perform longer and more complicated missions and scenarios. In these missions and scenarios, optimal path navigation through complicated environments is crucial to mission success. As more vehicles and more targets are involved in the mission, the complexity of the trajectory design problem grows rapidly, increasing the computation time to obtain the optimal solution [31].

Using MILP to design a whole trajectory with a planning horizon fixed at the goal is very difficult to perform in real time because the computational effort required grows rapidly with the length of the route and the number of obstacles to be avoided. This limitation can be overcome by using a receding planning horizon in which MILP is used to form a shorter plan that extends towards the goal, but does not necessarily reach it. This overall approach is known as either model predictive control (MPC) or receding horizon control [34].

In practice, the RHC uses a plant model and an optimization technique to design an input trajectory that optimizes the plant’s output over a period of time called the “planning horizon”. A portion of the input trajectory is then implemented over the shorter execution horizon, and the optimization is performed again starting from the state that is to be reached. If the control problem is not completed at the end of the planning horizon, the cost incurred past the planning horizon must be accounted for in the cost function.
2.2 Numerical Optimization Methods for Non Linear Programming

Linear programming problems, whether real or integer or mixed integer, are a subset of the most general set of non linear programming (NLP) problems.

In order to perform a survey on the solution methods applied in the field of NLP problems it is advisable to distinguish between unconstrained and constrained non linear programming problems [74].

Case of unconstrained problems

Let us consider a generic unconstrained optimization problem of the form:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in \Omega
\end{align*}
\]

(6)

where \( f \) is a real valued function and \( \Omega \), the feasible set, is a subset of \( \mathbb{R}^n \). With reference to the existence of a solution for the problem (6), the theorem of Weierstrass states that if \( f \) is continuous and \( \Omega \) is compact a solution exists. As well known, in a problem such as (6) it is possible to distinguish two kinds of solution points: local minimum points and global minimum points. Obviously the solution required for the optimization problem (6) is a global solution but practical reality, both from the theoretical and computational viewpoint, dictates that in many circumstance it is impractical to obtain a global minimum point and only a local minimum point can be determined. Based on this practical observation, usually the search for a relative minimum point is implicitly assumed as aim of the optimization problem solution.

To derive necessary conditions satisfied by a relative minimum point \( x^* \), the basic idea is to consider movement away from the point in some given direction. This means to derive some simple conditions satisfied by relative minimum points.

The first order necessary conditions for the unconstrained case are that if \( f \in C^1 \) in \( \Omega \), \( x^* \) is a relative minimum point of \( f \) over \( \Omega \) and \( x^* \) is an interior point of \( \Omega \), then:

\[
\nabla f(x^*) = 0.
\]

(7)
These necessary conditions in the pure unconstrained case, based on making a first order approximation to the function $f$ in the neighborhood of the relative minimum point, lead to $n$ equations in $n$ unknowns. They in many cases can be solved to determine the solution but, usually, solution methods for NLP unconstrained problems solve directly the optimization problem without explicitly attempting to solve these equations.

The second order necessary conditions for the unconstrained case are that if $x^*$ is an interior point of the set $\Omega$ and is a relative minimum point over $\Omega$ of the function $f \in C^2$, then:

$$\nabla f(x^*) = 0$$  \hspace{1cm} (8)

and

$$d^T \nabla^2 f(x^*) d \geq 0, \quad \forall d \in \mathbb{R}^n.$$  \hspace{1cm} (9)

The condition (9) states that the Hessian matrix of $f$, evaluated in $x^*$, is positive semidefinite and this condition plays a fundamental role in iterative methods for solving unconstrained NLP problems.

The conditions so far exposed are necessary and referred only to the case in which the point $x^*$ is interior to the feasible region $\Omega$. The second order sufficient conditions for the unconstrained case, once again referred to the case in which the relative minimum point is interior to the subset $\Omega$, is that, if $f \in C^2$ on $\Omega$ and (8) and (9) are satisfied, then $x^*$ is a strict relative minimum point of $f$.

The algorithms designed to solve non linear programming problems have the common feature of all being iterative descent algorithms. In other words, they are iterative in the sense that the algorithm generates a series of points each one is calculated on the basis of the points preceding it and are descent in the sense that, as each new point is generated by the algorithm, the corresponding value of some function evaluated at the most recent point decreases.

Before briefly describe the most relevant methods used in order to solve NLP unconstrained problems, we emphasize that there is a fundamental structure for almost all these descent algorithms. This structure is:

- one starts at an initial point,
- determines, according to a fixed rule, a direction of movement,
- moves in that direction to a relative minimum of the objective function on that line,
It must be noted that the primary differences between algorithms rest with the rule by which successive directions of movement are selected. Once the selection is made, all algorithms call for movement to the minimum point on the corresponding line and the process of determining the minimum point on a given line is called line search. There are several approach to the line search phase, as Fibonacci and golden section search or curve fitting procedures; a detailed analysis of these techniques and of NLP problems in general can be found in [74] and in [75].

Another relevant characteristic of iterative solution algorithms is that their convergence to a solution may be dependent by the starting point. If, for arbitrary starting points, the algorithm is guaranteed to generate a sequence of points converging to a solution, then the algorithm is said to be globally convergent. Unfortunately, not all algorithms have this desirable property.

An high level classification of the algorithms for NLP unconstrained problems solution may be:

- **gradient method** (first order algorithm), which makes use of a first order truncated Taylor series approximation of the objective function;
- **Newton’s method** (second order algorithm), which makes use of a second order truncated Taylor series approximation of the objective function;
- **conjugate direction methods**;
- **quasi-Newton methods**.

**Gradient method**

The gradient method, also called steepest descent method, is one of the oldest and most widely known methods for minimizing a function of several variables. We provide here a basic explanation of the principle on which the gradient method is based. For a complete analysis of the gradient method, one can make use of specialized textbooks, as for instance [74] and [75].

Let \( f \) have continuous first partial derivatives on \( \mathbb{R}^n \) and define, for convenience, the gradient of \( f \), which is a \( n \)-dimensional row vector, as a \( n \)-dimensional column vector. This is obtained by means of the following position:

\[
g(x) = \nabla f(x)^T. \tag{10}\]

The gradient method is then defined by the iterative algorithm:

\[
x_{k+1} = x_k - \alpha_k g(x_k), \tag{11}\]
where the subscript \( k \) is the index which defines the current step of the algorithm and \( \alpha_k \) is a non negative scalar minimizing the function \( f[x_k - \alpha g(x_k)] \).

In other words, the concept is that, from the point \( x_k \), the algorithm search, along the direction of the negative gradient \( -g(x_k) \), to a minimum point on this line; this minimum point, reached by means of the line search methods above mentioned, is taken to be \( x_{k+1} \).

It is possible to demonstrate that the gradient method is characterized by global convergence under weak conditions and that it converges linearly. However, the circumstances that the direction used in this method is of best descent and that the method is globally convergent does not be sufficient to assure the practical effectiveness of this method. The performance of the method of steepest descent is dependent from the formulation employed, in particular from the choice of variables \( x \) used for the problem definition. If the problem is ill conditioned, the method may converge very slowly, so their effectiveness is poor. Near the solution point, besides, the steepest descent method may become very slow, since its may proceed in an oscillatory manner.

**Newton’s method**

We provide here a basic explanation of the principle on which Newton’s method is based. For a complete analysis of this method, one can make use of specialized textbooks, as for instance [74] and [75].

Newton’s method is based on the idea that the function \( f \) being minimized is approximated locally by a quadratic function and this approximate function is then minimized exactly.

Thus, near \( x_k \) it is possible approximate \( f \) by the second order truncated Taylor series:

\[
  f(x) \approx f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)
\]  \hspace{1cm} (12)

The right-hand side of (12) is minimized at

\[
  x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)^T,
\]  \hspace{1cm} (13)

and this equation is the pure form of Newton’s method.
It is possible to demonstrate that Newton’s method has very good convergence properties if started near the solution point but, on the other hand, it is not globally convergent. This means that Newton’s algorithm require some modifications in order to obtain better convergence properties if it starts from a point which is not sufficiently close to the solution point. A very frequently used modification of the Newton’s algorithm (13) to do this is to introduce a search parameter $\alpha$ so that the method takes the form:

$$x_{k+1} = x_k - \alpha_k \left[ \nabla^2 f(x_k) \right]^{-1} \nabla f(x_k)^T,$$

(14)

where $\alpha_k$ is selected in order to minimize $f$. Other modifications exists in order to improve the convergence characteristics of this algorithm but the most relevant characteristic of Newton’s algorithm is its very good local convergence property. In other words, it is possible to demonstrate that if Newton’s method converges, it very quickly converges, with order two convergence.

In conclusion, Newton’s method, despite very good local convergence properties, has some considerable drawbacks, as:

- it may be impossible to invert the Hessian matrix;
- in some points, the possible presence of negative eigenvalues in the Hessian matrix may entail that the direction generated by the algorithm does not be a descent;
- even if the Hessian matrix is positive semidefinite, the evaluation of this matrix at every step of the procedure is computationally expensive and, if the matrix is ill conditioned, the algorithm may become numerically unstable.

This means that Newton’s method is rarely used in practice on large problems.

**Conjugate direction methods**

Conjugate direction methods are intermediate between the gradient and Newton’s method and their aim is to accelerate the slow convergence of the gradient algorithm while avoiding the computationally expensive evaluation and inversion of the Hessian matrix, required in Newton’s method.

We provide here a basic explanation of the principles on which conjugate direction methods are based. For a complete analysis of these methods, one can make use of specialized textbooks, as for instance [74] and [75].

All conjugate direction methods refers to the purely quadratic problem:
minimize $\frac{1}{2}x^TQx - b^Tx$, \hspace{1cm} (15)

where $Q$ is an $n$ by $n$ symmetric positive definite matrix and $b$ is an $n$-dimensional column vector. Taking into account that near the solution point every problem may be approximated by means of a quadratic form, the techniques developed for this problem are then extended, by approximation, to more general problems.

Conjugate direction methods are iterative procedures which, applied to a quadratic function of $n$ variables as (15), lead to the solution at most in $n$ steps.

Given a symmetric matrix $Q$, two vectors $d_1$ and $d_2$ are defined as “conjugate” with respect to $Q$ if they satisfy the condition $d_1^TQd_2 = 0$. The principle of conjugate direction methods is to start from an initial point $x_0$ and minimize the quadratic objective function in $n$ steps, along $n$ directions linearly independent and mutually conjugates with respect to the quadratic form.

The main conjugate direction method is the conjugate gradient method (CGM). This technique is very effective in dealing with general objective functions and is considered as a very good general purpose method. The CGM is based on the idea of selecting the successive direction vectors as a conjugate version of the successive gradients obtained step by step. This implies that the directions are determined sequentially as the method progresses, in such a way as at each step the current negative gradient vector is evaluated as a linear combination of the previous direction vectors is added to this gradient in order to determine a new conjugate direction vector along which to move.

If we define a generic point $x_i \in \mathbb{R}^n$ as the starting point of the numerical procedure and define moreover the initial direction $d_0 = -g_0 = b - Qx_0$, the algorithm of CGM can be summarized by means of the rule [74]:

$$x_{k+1} = x_k + \alpha_k d_k$$ \hspace{1cm} (16)

where

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$$ \hspace{1cm} (17)

$$d_{k+1} = -g_{k+1} + \beta_k d_k$$ \hspace{1cm} (18)

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$$ \hspace{1cm} (19)
and

\[ g_k = Qx_k - b. \]  \hspace{1cm} (20)

The analysis of the algorithm shown that the first step is the same as the gradient method (see (16) and (11)) but the direction of motion in each succeeding step is a linear combination of the current gradient and the preceding direction vector. Moreover, for updating the direction vector, the CGM makes use of the simple formulae (18) and (19) and does not require matrix updating. Finally, the method converges in \( n \) steps. These are the advantages of the conjugate gradient method, which is in practice only slightly more complicated than the gradient method.

**Quasi-Newton methods**

Since the evaluation and use of the Hessian matrix is impractical or computational expensive, the principle of the quasi-Newton methods (QNM) consists in the use of an approximation to the inverse Hessian matrix in place of the true inverse, which is required in Newton’s method. There are several QNM, which are different with respect to the approximation used for the inverse Hessian matrix. For instance, this matrix can be kept fixed throughout the iterative process or can be varied and this choice involves different types of quasi-Newton algorithms.

We provide here a basic explanation of the principles on which QNM are based. For a complete analysis of these methods, one can make use of specialized textbooks, as for instance [74] and [75].

The quasi-Newton methods are in general based on the observation that the Taylor expansion of the gradient of the objective function \( f \) leads to:

\[ \nabla f(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k). \] \hspace{1cm} (21)

If we define:

\[ y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \] \hspace{1cm} (22)

\[ s_k = x_{k+1} - x_k \] \hspace{1cm} (23)

we obtain that the approximation (21) can be rewritten as:
\[ \nabla^2 f(x_k) s_k = y_k. \]  \hspace{1cm} (24)

QNM generate at each step a matrix \( B_k \) which satisfies the relation:
\[ B_k s_k = y_k \]  \hspace{1cm} (25)
so the algorithm is:
\[ x_{k+1} = x_k - B_k^{-1} \nabla f(x_k). \]  \hspace{1cm} (26)

Obviously, the equation (25) does not be sufficient in order to determine the \( n^2 \) elements of the matrix \( B_k \) and some degrees of freedom are then used in order to impose the matrix to be symmetric and positive definite.

One of the most employed methods for the definition of \( B_k \) is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) technique, in which the updating of the matrix is obtained as:
\[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \]  \hspace{1cm} (27)

Many QNM have very good properties of local convergence and in some cases does not be used the matrix \( B_k \) but matrices \( H_k \) are defined in such a way as:
\[ s_k = H_k y_k \]  \hspace{1cm} (28)
so the descent algorithm becomes:
\[ x_{k+1} = x_k - H_k \nabla f(x_k). \]  \hspace{1cm} (29)

Notice that the algorithm (29) does not require the solution of an equation system in order to determine the descent direction.
CASE OF CONSTRAINED PROBLEMS

Let us consider a generic constrained NLP problem of the form:

\[
\begin{aligned}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad h(x) = 0 \\
& \quad g(x) \leq 0 \\
& \quad x \in \Omega
\end{aligned}
\]  

(30)

where \( f \) is a real valued function and \( \Omega \), the feasible set, is a subset of \( \mathbb{R}^n \). In this optimization problem the constraints \( h(x) = 0 \) and \( g(x) \leq 0 \) are referred to as functional constraints and in particular \( h \) is an \( m \)-dimensional vector-valued function \((m \leq n, \text{ of course})\) and \( g \) is a \( p \)-dimensional vector-valued function; all the functions in these vectors and the function \( f \) are considered to be continuous up to the second partial derivatives.

The constraint \( x \in \Omega \subset \mathbb{R}^n \) is defined as set constraint but in most cases it is assumed that \( \Omega = \mathbb{R}^n \), so this constraint is neglected. Anyway, a point \( x \in \Omega \) that satisfies also the functional constraints is said to be feasible. Moreover, an inequality constraint \( g_i(x) \leq 0 \) is said to be active at a feasible point \( x \) if \( g_i(x) = 0 \) and inactive at the same point if \( g_i(x) < 0 \); obviously, any equality constraint \( h_i(x) = 0 \) is said active at any feasible point.

For constrained problems in the most general form (30), it is possible to derive necessary and sufficient conditions for the minimum point.

To do it, first it must define the concept of regular point: if \( x^* \) is a point which satisfies the constraints \( h(x^*) = 0, \ g(x^*) \leq 0 \) and the gradients of the active constraints evaluated in this point are linearly independent, then \( x^* \) is defined as a regular point for the constraints.

Now we can derive the necessary first order conditions for constrained non linear optimization problems (also defined as “Lagrange first order necessary conditions”). These conditions, which usually are said to be Kuhn-Tucker conditions, state that if \( x^* \) is a relative minimum point for the NLP constrained problem (30) and is a regular point for the constraints, then there are a vector \( \lambda \in \mathbb{R}^m \) and a vector \( \mu \in \mathbb{R}^p \) with \( \mu \geq 0 \) such that:

\[
\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) = 0
\]  

(31)
\[ \mu^T g(x^*) = 0. \] (32)

Moreover, it is possible to demonstrate the necessary and sufficient second order conditions for constrained non linear optimization problems. These conditions require a more sophisticated mathematical development, which is out of the main subject of this literature survey; for a complete explanation of necessary and sufficient second order optimality conditions with reference to constrained NLP problems, one can refer to specialized literature or books, as [74] and [75].

In this section, we provide a general presentation of the algorithms designed to solve constrained NLP problems. These algorithms can be firstly classified by means of a very high level scheme: assigned the constrained minimization problem, which have \( n \) variables and \( m \) constraints, the algorithms for solving this problem can work in spaces of dimensions \( n-m \) (primal methods), \( n \) (penalty and barrier methods), \( m \) (dual methods) or \( n+m \) (Lagrange methods).

**Primal methods**

With reference to the minimization NLP problem (30), we assume that all the functions involved in this problem have continuous partial derivatives of order three. The constraints can be regarded in such a way as they characterize a specified region of feasibility in \( \mathbb{R}^n \), so the minimization problem can be considered as that of minimizing the objective function \( f \) over this region.

The aim of a primal method for solving a minimization problem is to work on the original problem directly by searching through the feasible region for the optimal solution. Each point in this search process is then feasible and the value of the objective function constantly decreases. Primal methods, hence, are direct methods, in the sense that they work directly on the original problem, and, for a problem with \( n \) variables and \( m \) constraints \((m<n)\), they work in the feasible space, which is a \( n-m \) dimensional space.

An example of primal methods are the feasible direction methods, which can be regarded as an extension of unconstrained NLP descent methods to constrained NLP problems. The principle of these methods is to take steps through the feasible region of the form:

\[ x_{k+1} = x_k + \alpha_k d_k \] (33)
where \( \mathbf{d}_k \) is a direction vector and \( \alpha_k \) is a non-negative scalar, selected in order to minimize the objective function \( f \) in such a way as the point \( \mathbf{x}_{k+1} \) and the segment joining \( \mathbf{x}_k \) and \( \mathbf{x}_{k+1} \) are feasible. It is evident that each step of feasible direction methods is the composition of selecting a feasible direction and a constrained line search.

Another example of primal methods are the active set methods, which are based on the idea of to partition inequality constraints in two groups: those that are to be treated as active and those that are to be treated as inactive, which are essentially ignored. If we concentrate over a constrained problem which have only inequality constraints, as:

\[
\begin{align*}
\text{minimize} & \quad f(\mathbf{x}) \\
\text{subject to} & \quad \mathbf{g}(\mathbf{x}) \leq 0
\end{align*}
\]  \tag{34}

and if we define \( A \) the index set of active constraints, then the necessary conditions for this problem are:

\[
\nabla f(\mathbf{x}) + \sum_{i \in A} \lambda_i \nabla g_i(\mathbf{x}) = 0 \\
g_i(\mathbf{x}) = 0, \quad i \in A \\
g_i(\mathbf{x}) < 0, \quad i \notin A \\
\lambda_i \geq 0, \quad i \notin A \\
\lambda_i = 0, \quad i \in A.
\]  \tag{35}

In these necessary conditions, the first two lines correspond to the necessary conditions of equality constrained problem obtained by requiring the active constraints to be zero and the next line guarantees that the inactive constraints are satisfied, so, if the active set were known, the original problem could be replaced by the corresponding problem having equality constraints only.

The main idea of active set methods is to define at each step of an algorithm a set of constraints (said the working set) that is to be treated as active set. The working set is selected as a subset of the constraints that are actually active at the current point so this point is feasible for the selected working set and the algorithm
proceeds to move on the surface defined by the working set to an improved point, in which the working set may be changed. There are several methods for determining the movement on the surface defined by the working set.

The most practical primal methods are the gradient projection method (GPM) and the reduced gradient method (RGM). Both of these methods can be regarded as an application of the gradient method for unconstrained NLP problems on the surface defined by the active constraints and RGM can be considered as more effective than GPM.

The main idea of the GPM is that at a feasible point $\mathbf{x}_k$ the active constraints are determined and the negative gradient is projected onto the subspace tangent to the surface determined by these constraints. The direction for the next step is then determined by this vector, if it is non zero; in general, yet, the direction determined by this vector may be not feasible, for instance in the case in which the constraint surface is highly curved, so the method must be modified in order to correct the direction in such a case [74].

The RGM is based on the transformation of the original problem in the standard form:

$$
\begin{align*}
\text{minimize} & \quad f(\mathbf{x}) \\
\text{subject to} & \quad \mathbf{h}(\mathbf{x}) = 0 \\
& \quad \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}
\end{align*}
$$

where $\mathbf{h}(\mathbf{x})$ is an $m$-dimensional vector. This form can be always obtained starting from a general constrained NLP problem by introducing some slack variables and selecting appropriate values for the vectors $\mathbf{a}$ and $\mathbf{b}$. The detailed explanation of this procedure is not the main concern of this literature survey, so one can refer to specialized books as, for instance, [74] and [75].

To sum up primal methods, it must be noted that they have some advantages:

- each point generated in search procedure is feasible, so, if the process is terminated before reaching the solution, the terminating point is feasible;
- the characteristics of global convergence are often satisfactory;
- most primal methods are applicable to general NLP problem, without regard to the problem structure.
On the other hand, the primal methods have some drawbacks:

- they require a pre-processing procedure in order to obtain an initial feasible point;
- they involve computational difficulties arising from the necessity to remain in the feasible region as the method progresses.

**Penalty and barrier methods**

For a minimization constrained NLP problem in the generic form:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in S 
\end{align*}
\]  

(37)

where \( f \) is a continuous function on \( \mathbb{R}^n \) and \( S \) is a constraint set in \( \mathbb{R}^n \), the *penalty methods* (PM) are based on the substitution of the constrained problem with a sequence of unconstrained problems in such a way as the sequence of their optimal points converges at the optimal point of the constrained original problem.

In order to define these unconstrained problems, a penalty function \( P(x) \) is used, defined as:

\[
P(x) \begin{cases} 
0 & \text{if } x \in S \\
> 0 & \text{if } x \not\in S.
\end{cases}
\]  

(38)

The unconstrained problem which is then considered is:

\[
\text{minimize} \quad [f(x) + cP(x)],
\]  

(39)

where \( c \) is a positive constant, and this new problem is solved by using an algorithm for unconstrained NLP problems. Hence, the procedure for solving a generic constrained non linear problem such as (37) making use of a penalty method is this: let \( \{c_k\} \), \( k = 1, 2, \ldots \), be a positive sequence tending to infinity such that, for each \( k \), is \( c_{k+1} > c_k > 0 \); for each \( k \) solve the unconstrained problem (39), obtaining a solution point \( x_k \). Ideally, if \( k \to \infty \) (also \( c \) tends to infinity, of course),
the solution point of the penalty problem will converge to a solution of the
constrained original problem.

In the case in which the minimization problem has only inequality constraints, in
such a way as the minimization original NLP problem is in the form:

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, 2, ..., p,
\end{align*}
\] (40)

an alternative to penalty methods are the \textit{barrier methods} (BM). They make use of
a barrier function \(B(x)\) defined as a continuous positive function which tend to
infinity as \(x\) approaches the boundary of the feasible region. The most common
barrier function is the one defined as:

\[
B(x) = -\sum_i \log[-g_i(x)]
\] (41)

The procedure of barrier methods is similar to the one of PM. The original
constrained NLP problem (40) is replaced by a sequence of unconstrained problems
defined as:

\[
\text{minimize} \quad [f(x) + mB(x)]
\] (42)

and the procedure is: let \(\{m_k\}, k = 1, 2, ...,\) be a positive sequence tending to zero
such that, for each \(k\), is \(0 < m_{k+1} < m_k\); for each \(k\) solve the unconstrained problem
(42), obtaining a solution point \(x_k\). Ideally, if \(k \to \infty\), the solution point of the
barrier problem will converge to a solution of the constrained original problem.

Briefly, the idea of penalty and barrier methods is to approximate constrained
NLP problems by means of sequences of unconstrained NLP problems, which makes
possible to use the algorithms developed for solving unconstrained problems for
constrained problems too. In the case of penalty methods, this approximation is
obtained by adding to the objective function a term that prescribes an high cost for
violation of the constraints. In the case of barrier methods, the approximation is
obtained by adding to the objective function a term that favours points interior to
the feasible region over those near the boundary.
For an optimization NLP problem with $n$ variables and $m$ constraints, penalty and barrier methods works directly in the $n$-dimensional space of variables.

An advantages of PM is that, under weak conditions, it is possible to demonstrate that, if at each iteration the solution $x_k$ of the unconstrained problem (39) is a global solution, then the sequence of $\{x_k\}$ converges to the global optimal solution for the original problem, as $k$ tends to infinity. A disadvantage of penalty methods, yet, is that the transformation from the constrained problem to the unconstrained problem does not be always possible and, moreover, as the parameter $c_k$ increases, the solution of the unconstrained problem (39) becomes difficult. Similar characteristics may be associated to barrier methods.

**Dual methods**

Dual methods are based on the use of Lagrange multipliers, which are the fundamental unknowns associated with a constrained NLP problem. These methods, hence, do not work on the original constrained problem directly but work instead on an alternate problem (dual problem) whose unknowns are the Lagrange multipliers of the original problem. This implies that, if the original problem has $n$ variables and $m$ equality constraints ($m<n$), dual methods work in the $m$-dimensional space of Lagrange multipliers.

If we consider a generic constrained NLP problem as (30) and suppose that $x^*$ is a regular point of constraints and a local solution of this optimization problem, then there are Lagrange multipliers $\lambda^*$ and $\mu^* \geq 0$ such that [74]:

$$
\nabla f(x^*) + (\lambda^*)^T \nabla h(x^*) + (\mu^*)^T \nabla g(x^*) = 0
$$

$$
(\mu^*)^T g(x^*) = 0.
$$

(43)

If we impose the local convexity assumptions that the Hessian of the Lagrangian:

$$
L(x^*) = F(x^*) + (\lambda^*)^T H(x^*) + (\mu^*)^T G(x^*)
$$

(44)

is positive definite and, for $\lambda$ and $\mu \geq 0$ near $\lambda^*$ and $\mu^*$, we define the dual function:
\[
\varphi(\lambda, \mu) = \min \left[ f(x) + \lambda^T h(x) + \mu^T g(x) \right],
\]

where the minimum is taken locally near \( x^* \), then it is possible to demonstrate that \( \varphi \) achieves a local maximum with respect to \( \lambda, \mu \geq 0 \) at \( \lambda^*, \mu^* \).

This means that, locally, the original constrained NLP problem (30) is equivalent to unconstrained NLP problem local maximization of the dual function (45). Hence it is possible to locally establish a "dualization", that is an equivalence between a constrained problem in \( x \) (the original minimization problem (30)) and an unconstrained problem in \( \lambda, \mu \) (the problem of maximization of the dual function (45)).

This is only an introductory exposition of the main idea on which dual methods are based. For an exhaustive explanation of more sophisticated dual methods, as for instance the augmented Lagrangian methods, one can refer to specialized textbooks, as [74] and [75].

**Lagrange methods**

The Lagrange methods have the aim of solving constrained NLP problems directly solving the Lagrange first order necessary conditions (31) and (32).

For instance, if the NLP problem has only equality constraints, so it is in the form:

\[
\begin{cases}
\text{minimize} & f(x) \\
\text{subject to} & h(x) = 0,
\end{cases}
\]

the Lagrange methods solve the equation system:

\[
\begin{cases}
\nabla f(x) + \lambda^T \nabla h(x) = 0 \\
h(x) = 0
\end{cases}
\]
with respect to $\mathbf{x}$ and $\lambda$. This implies that, since the necessary conditions are a system of $n+m$ equations in the $n+m$ unknowns, which are the components of $\mathbf{x}$ and $\lambda$, Lagrange methods work in an $(n+m)$-dimensional space.

In the particular case of problems in which the objective function is a quadratic form ($quadratic$ $programming$ $problems$) and it contains linear equality constraints only, so the problem is expressed as:

$$\begin{cases} 
\text{minimize} & \frac{1}{2} \mathbf{x}^T \mathbf{Qx} + \mathbf{c}^T \mathbf{x} \\
\text{subject to} & \mathbf{Ax} = \mathbf{b},
\end{cases} \quad (48)$$

with $\mathbf{A}$ a matrix of full rank and $\mathbf{Q}$ positive definite, a unique solution exists and the Lagrange necessary first order conditions are:

$$\begin{align*}
\mathbf{Qx} + \mathbf{A}^T \lambda + \mathbf{c} &= 0 \\
\mathbf{Ax} - \mathbf{b} &= 0,
\end{align*} \quad (49)$$

so the problem consists in the solving this $(n+m)$-dimensional linear system of equations, by means of standard linear procedures.

If the quadratic programming problem has inequality constraints, it is almost always solved by using an active set method, in which the direction of movement is toward the solution of the corresponding equality constrained problem and the solution is reached in a finite number of steps.

In the most general case of NLP problems, the Lagrange first order necessary equations can be solved by means of several direct approach, which may be regarded as an extension of first order methods as conjugate gradient method and Newton’s method. For instance, one class of direct solution methods for general NLP problems consists of first order methods that move in a direction related to the residual in the equations. A second class of methods is an extension of the conjugate direction methods to non positive definite systems. Finally, a third class of methods is based on Newton’s method for solving systems of non linear equations or for solving a linearized version of the system at each iteration. Others direct methods are also possible, however for a complete exposition of direct Lagrange methods one can refers to specialized literature and NLP books.
Some applications in literature of LP and NLP to trajectory generation problems are the ones reported in [3], [18], [27], [31], [32] and [34]; these applications will be described in section 4 and paper [3] will be also described more in details in appendix.
2.3 Evolutionary Algorithms

A brief general description of evolutionary algorithms has been furnished by Betts in [1]. In this paper, the author emphasize that the ability to define convergence is a fundamental property of calculus-based methods. All of the numerical methods for LP and NLP problems solution have well-defined termination criteria, so it is possible to decide whether a candidate solution is in fact an answer by evaluating the appropriate analytical conditions. In contrast, when the variables are discrete, calculus-based methods do not apply. In general, for problems with discrete variables the only way to decide if a candidate solution is in fact an answer is by comparison with all other possible candidates. Unfortunately, this is a combinatorial problem that is computationally prohibitive for all but the smallest applications.

To avoid direct comparison of all possible solutions, it is necessary to introduce randomness at some point in the optimization process and abandon a definitive convergence criterion. The basic notion of genetic algorithms (GA), simulated annealing, tabu search, and evolutionary or Monte Carlo methods is to randomly select values for the unknown problem variables. After a finite number of random samples, the best value is considered the answer. For some applications, notably those with discrete variables, algorithms of this type are the only practical alternative. However, in the opinion of the author [1], since trajectory optimization problems in general are not characterized by discrete variables, there is no reason to use in this field an evolutionary method. Nevertheless, the author admits, methods of this type have attracted the interest of many analysts, presumably because they are incredibly simple to apply without a detailed understanding of the system being optimized.

In confirmation of this observation, in literature several works appear in which evolutionary techniques are used as a search or optimization method [45], [58]. In general, in the group of so called evolutionary algorithms, genetic algorithms are the best known, nevertheless these methods belong to a larger group called heuristic methods or soft computing methods, which are techniques strongly based on intuition instead of hard mathematical derivations.

Whereas gradient based methods proceed by deterministically improving an iteration point, genetic algorithms use a random population of solution candidates. The features of the best candidates are used for generating new populations with the intent of producing new and better candidates. The basic approach uses three
operators on the population: selection, crossover and mutation. Selection chooses, possibly in a random fashion, the best candidates for the new population. Crossover combines features of the selected members of the population and mutation introduces random variability into the members.

The convergence of the repeated selection–crossover–mutation procedure to the optimal solution is based on the “schema theorem”. A schema is a similarity template fixing some values of the elements of the decision variable vector and leaving others free. The schema theorem states that the number of candidates in the population matching a schema increases if those candidates are above average.

Genetic algorithms are hence a global search strategy that generally starts with the initial population randomly dispersed into the search domain. They have a fairly good probability of locating the global optimum from among local optima, in contrast to gradient based methods, which converge to the local optimum closest to the given starting point.

These algorithms, hence, are a class of search methods with remarkable balance between exploitation of the best solutions and exploration of the search space. They combine elements of directed and stochastic search and, therefore, are more robust than existing directed search methods [45]. Additionally, they may be easily tailored to the specific application of interest, taking into account the special characteristics of the problem under consideration. The natural selection process is simulated in genetic algorithms using a number (population) of individuals (solutions to the problem) to evolve through certain procedures. Each individual is represented through a string of numbers (bit strings, integers or floating point numbers) in a similar way with chromosomes in nature. Each individual’s quality is represented by a fitness function tailored to the problem to be optimized.

Classical GA use binary coding for the representation of the genotype. However, floating point coding moves the GA closer to the problem space, allowing the operators to be more problem specific and providing, for instance in the case of trajectory generation, a better physical representation of the path line control points and easier control of the space constraints. Additionally, two points that are close to the physical space are also close in the representation space (the genotype encoding), and vice versa. With this type of encoding, directed search techniques gain physical representation and they are easily applicable.

The GA starts by generating, randomly, the initial chromosome population with their genes taking values inside the desired constrained space. After the evaluation of each individual’s fitness function, operators are applied to the population,
simulating the according natural processes. Applied operators include various forms of selection, recombination and mutation, which are used in order to provide the next generation chromosomes. The process of a new generation evaluation and creation is successively repeated, providing individuals with high values of fitness function. Each chromosome consists of the same (fixed) number of genes.

Several operators can be applied to the selected chromosomes [45]. The classical one-point crossover scheme is: two randomly selected chromosomes are divided in the same (random) position, while the first part of the first one is connected to the second part of the second one, and vice-versa. This operator is used in order to provide information exchange between different potential solutions to the problem.

The classic uniform mutation scheme is an operator which alters a randomly selected gene of a chromosome in such a way as the new gene takes its random value from the constrained space, determined in the beginning of the process. The mutation operator is used in order to introduce some extra variability into the population.

In order to provide fine local tuning, non-uniform mutation and heuristic crossover are used, along with the classic mutation and crossover schemes. The non-uniform mutation operator chooses randomly, with a predefined probability, the gene of a chromosome to be mutated. Contrary to the uniform mutation described above, the search space for the new gene is not fixed, but it shrinks close to the previous value of the corresponding gene as the algorithm converges. The search is uniform initially, but very local at later stages.

The heuristic crossover operator, finally, generates a single offspring \( x_3 \) from two parents \( x_1 \) and \( x_2 \). If \( x_1 \) is not worse than \( x_2 \), then the offspring is given as:

\[
x_3 = r(x_2 - x_1) + x_2
\]

where \( r \) is a random number between 0 and 1. In this way, a search direction is adopted, providing fine local tuning and search in the most promising direction (for instance, in a trajectory generation problem, the chromosome genes can be regarded as physical coordinates so the direction of search actually assumes a physical meaning).

The one reported in this section, of course, is an introductory explanation of evolutionary algorithms and genetic algorithms in particular. In order to obtain a
more detailed analysis of these methods, one can refer to specialized literature and books, as for instance [76].

Some applications in literature of evolutionary algorithms with reference to trajectory generation problems are the ones reported in [45] and [58]. These applications will be described in section 4 and, more in details, in appendix.
2.4 Fuzzy methods

The most relevant soft computing techniques are genetic algorithms, fuzzy methods and neural networks. The first ones have been briefly described in the previous section whereas neural networks will be examined in the next section. In this section, fuzzy methods are summarily illustrated.

Fuzzy methods are based on “if ... then ... else” rules and represents a subset of knowledge based processing (expert systems), in which the reason engine (also said inference engine) applies fuzzy logic (FL). The main idea of fuzzy logic is that any proposition is never completely true nor completely false, but it has a truthfulness level (degree of membership) on which will be based the application of the appropriate decision rule. The truthfulness level is described by the membership of the considered input to a specified set, where the possible sets are not definitely distinct but overlapped. This is due to the fact that the membership functions are usually triangular or trapezoidal, so an input can belong to two sets at the same time, but with different degrees of membership.

Briefly, a general fuzzy logic based procedure may be summarized by means of three steps:

- the first operation in application of fuzzy methods is the transformation of the input data in fuzzy form, by means of the definition of membership functions and sets;
- once determined the fuzzy form of data, appropriate “if ... then ... else” rules must be defined, in order to obtain the fuzzy outputs starting from the fuzzy inputs;
- from the fuzzy outputs determined by means of early defined rules, it is necessary to convert output data in fuzzy form to numerical outputs. In this operation may arise perhaps some conflict, that is situations in which a same fuzzy output is assigned to different sets, so appropriate techniques in order to solve these conflict must be used, for instance the “center of gravity method”.

An example of membership functions is shown in Figure 2 [78].
Fuzzy methods may be applied in control field as well as in optimization field.

For instance, as well known, the purpose of control is to influence the behavior of a system by changing an input or inputs to that system according to a rule or set of rules that model how the system operates. Classic control theory uses a mathematical model to define a relationship that transforms the desired state and observed state of the system into an input or inputs that will alter the future state of the system. The most common example of a control model is the PID controller, which takes the output of the system and compares it with the desired state of the system, so adjusting the input value based on the difference between the two values. The major drawback of such a system is that it usually assumes that the system being modeled is linear and well known, so as the complexity of the system increases it becomes more difficult to formulate the mathematical model.

Fuzzy control replaces the mathematical model with another that is build from a number of smaller rules that in general only describe a small section of the whole system. The process of inference considers these rules together in order to produce the desired outputs. In this way, the inputs and outputs of the system remain unchanged but the mathematical model is replaced by a fuzzy model.

With reference to the optimization field, as well known, mathematical programming (MP) problems, such as the ones considered in sections 2.1 and 2.2, form a subclass of decision-making problems, where preferences between alternatives are described by means one objective function defined on the set of alternatives. The value of the objective function describes the effects of the choice of the alternatives and the set of alternatives is described by means of constraints (equations or inequalities or both), representing relevant relationship between alternatives. In any case, the results of the analysis using given formulation of the mathematical programming problem depend largely upon how adequately various
factors of the real system are reflected in the description of the objective function
and of the constraints.

Descriptions of the objective function and of the constraints in a MP problem
usually include some parameters. The nature of those parameters depends, of
course, on the approximation accepted for the model representation, and their
values are considered as data that should be known. Clearly, the values of such
parameters depend on multiple factors not included into the formulation of the
problem. Trying to make the model more representative, often it must to include
more complex relations into it, causing that the model becomes more cumbersome
and analytically unsolvable.

Moreover, it can happen that such attempts to increase the precision of the
model will be of no practical value due to the impossibility of measuring the
parameters accurately. On the other hand, the model with some fixed values of its
parameters may be too crude, since these values are often chosen in a quite
arbitrary way.

An intermediate approach is based on the introduction into the model of the
parameters in the form of fuzzy sets of their possible values. The resultant model,
although not taking into account many details of the real system in question could
be a more adequate representation of the reality than that with more or less
arbitrarily fixed values of the parameters.

On this way we obtain a new type of MP problems containing fuzzy parameters.
Treating such problems requires the application of fuzzy-set-theoretic tools in a
logically consistent manner. Such treatment forms an essence of fuzzy
mathematical programming (FMP), investigated in specialized literature and books.

FMP and related problems have been extensively analyzed and many papers
have been published displaying a variety of formulations and approaches. Most
approaches to FMP problems are based on the straightforward use of the
intersection of fuzzy sets representing goals and constraints and on the subsequent
maximization of the resultant membership function. This approach has been
mentioned by Bellman and Zadeh already in their paper published in the early
seventies. Later on many papers have been devoted to the problem of
mathematical programming with fuzzy parameters, known under different names,
mostly as fuzzy mathematical programming.

The fuzzy mathematical programming is examined in specialized literature or
books, as for instance [78].
In this section we have provided a very brief and introductory description of the main idea on which fuzzy methods are based. A complete explanation of fuzzy logic and methods is not the main subject of this literature survey, so we emphasize that in order to obtain more information about these very interesting methods one can refers to specialized literature and books, as for instance [77] and [78].

Fuzzy logic is nowadays widely employed in control system, primarily in cases in which the high complexity of the system examined does not allow a classical control theory application. In the field of trajectory generation also some applications have been made and reported in literature, as for instance [46] and [55]; these applications are described in section 4 and the paper [55] in particular will be analyzed more in details in appendix.
2.5 **NEURAL NETWORKS**

The third soft computing technique which is used in literature with reference to trajectory generation problems is the one of neural networks. This is due to the fact that the properties of neural networks (NN) make them promising candidates for customizing navigation and control systems, hence their application in the field of on-line trajectory generation is not surprising.

Neural networks have the capability of estimating various mathematical functions, including highly nonlinear functions and, furthermore, they can in many cases be trained to adapt to changing input-output relationship. These features give to NN a great potential in control systems also for highly nonlinear dynamics, which makes these techniques suitable in the field of control systems for unmanned aerial, undersea and ground vehicles.

Really, several authors consider NN as a very powerful tool, whereas others are very skeptical, because they regard negatively the lack of formalized methods and analyses of neural network controller systems, especially regarding the stability of the controllers.

Nevertheless, without doubt NN have several properties which make them suitable for control purposes [69]:
- nonlinearity;
- parallel structure;
- hardware implementation;
- multivariable nature.

More into details, the nonlinear nature of neural networks makes them particularly well suited for solving complex nonlinear control problems and their parallel structure facilitates the construction of parallel implementation of control systems, which can results in robust and fast processing systems. Furthermore, NN can be easily implemented in hardware and their ability to correctly map functions with many inputs and outputs makes neural networks interesting for the control of multivariable systems.

Historically, neural networks development is due to the attempt of reproduce typical activities of the human brain. The human brain may be regarded as a very complex, nonlinear and parallel system, nevertheless constituted by very simple elements: the neurons (Figure 3). The brain, furthermore, has the capability of change the connections among neurons and is fault tolerant, in the sense that it works even if some connections are damaged and its performances deteriorate gradually. This means that, in order to reproduce human brain capabilities, the
neural networks must be a structure of very simple elements, distributed, parallel and capable to learn and generalize. The concept of generalization is very important and refers to the fact that the NN must furnish an appropriate output even in the presence of inputs not examined in the training process.

![Biological neuron](image)

*Figure 3 – Biological neuron*

The main element of a neural network is the artificial neuron (Figure 4), which has several inputs and only one output. To each input a weight is associated, which determines the connecting capability of the input channel. The neuron activation is a function of the weighted sum of the inputs.

With reference to the artificial neuron model, we have \( n \) input channels \( x_1, \ldots, x_n \) to each a weight \( w_i \) is associated. The weight may be positive (excitation channel) or negative (inhibitory channel) and the absolute value of this weight represents the permeability of the input channel, that is the strength of the connection.

The output \( y \) (state) of the neuron is the signal by which the neuron transfers its activity outside and is evaluated by applying the activation (or transfer) function \( f \) to the weighted sum of the inputs \( \xi \) (said excitation level):

\[
y = f(\xi) = f\left(\sum_{i=1}^{n} w_i x_i\right).\tag{51}
\]
In the artificial neuron model shown in Figure 4 a threshold value \( h \) equivalent to a bias value \( w_0 = -h \) is represented, which can be interpreted as the weight associated to a constant input \( x_0 \) always equal to 1 and reduces the excitation level value, so (51) becomes:

\[
y = f(\xi) = f\left(\sum_{i=0}^{n} w_i x_i - h\right) = f\left(\sum_{i=0}^{n} w_i x_i \right).
\]

(52)

The activation function defines the output of the neuron as a function of the excitation level and may produces as output a real number or a Boolean value. Some examples of activation functions are the hard limiter:

\[
y = f(\xi) = \begin{cases} 
1 & \text{if } \xi \geq 0 \\
0 & \text{if } \xi < 0
\end{cases}
\]

(53)

or other well known functions as liner, piecewise linear:

\[
y = f(\xi) = \begin{cases} 
1 & \text{if } \xi \geq 1 \\
\xi & \text{if } 0 \leq \xi \leq 1 \\
0 & \text{if } \xi < 0
\end{cases}
\]

(54)
and standard sigmoid (also said logistic function):

\[ y = f(\xi) = \frac{1}{1 + \exp(-\xi)} \]  

These functions are shown in Figure 5.

![Figure 5 – Some activation functions [79]](image)

Notice that the sigmoid function assumes all the values between 0 and 1, whereas classical threshold functions as hard limiter assume only the values 0 and 1; furthermore, the standard sigmoid is differentiable.

With reference to the architecture of NN, it is possible to distinguish two majors kinds of network topology:

- feedforward (or acyclic) NN;
- feedback (or recurrent or cyclic) NN.

In feedforward NN, the connections between neurons do not form cycles, while in feedback (or recurrent) NN there are cycles in the connections. Some schematic examples of feedforward NN are shown in Figure 6, while some schematic examples of feedback NN are shown in Figure 7.
From Figure 6, it may be noted that the neurons on the first layer have the only task to transfer the input signals to next layers and the signals go from the input towards the output (feedforward). In Figure 8 is shown a stratified feedforward layer in which an hidden layer, that is a layer of neurons which does not communicate directly with outside, is outlined:

In general, a NN may contains several hidden layers, so the connections among neurons will be represented by a number of matrices equal to the number of adjoining layers pairs.
The performances of the NN depend from the system architecture, that is from
the number of layers, from the number of neurons in each layer, from the transfer
function of neurons and, finally, from the weights associated to communication
channels among neurons. The number of layers, the number of neurons for each
layer and the typology of the activation function are fixed, so it must provide an
optimal choice of the weight values. This is obtained by means of the *learning
phase*.

The most common method to train the neural network is the one consisting in
introducing to the network several examples (said the *training set*). The answer of
the network for each example is compared to the exact answer, so the

The number of players, the number of neurons for each
layer and the typology of the activation function are fixed, so it must provide an
optimal choice of the weight values. This is obtained by means of the *learning
phase*.

The most common method to train the neural network is the one consisting in
introducing to the network several examples (said the *training set*). The answer of
the network for each example is compared to the exact answer, so the
corresponding error is evaluated and, based on this error, the weights used in the
network connections are adjusted. This process is repeated over the whole training
set, until the answers obtained from the NN produce an error lower than a fixed
value. This process is said supervised learning but, in general, the two main kinds
of learning algorithms are:

- supervised learning;
- unsupervised learning.

In *supervised learning*, the correct results (target values) are known and are
given to the NN during training, so that the NN can adjust its weights to try match
its outputs to the target values. After the training, the NN is tested by giving it only
input values, not target values, and seeing how close it comes to outputting the
correct target values. In the test phase also the generalization capabilities of the
NN are examined and it’s worth to emphasize that the performance of the network
are strongly dependent from the training set used in the learning phase, which is
problem specific.

The most usual rule employed in supervised learning is the *delta rule*, also said
*Widrow-Hoff rule*. If \( x=(x_1, ... , x_n) \) is the input set to the neuron and \( t, y \) are
respectively the exact and the neural outputs, the error of the NN in evaluating the
input set is:

\[
\delta = t - y. \tag{56}
\]

The delta rule states that the correction to a weight \( (\Delta w_i) \) in this situation is:

\[
\Delta w_i = \eta \cdot \delta \cdot x_i, \tag{57}
\]
where $\eta$ is a real number between 0 and 1, said the learning rate, which determines the speed of learning of the neuron. Notice that the Widrow-Hoff technique modifies only the weights of the neuron which have contributed to the error, because if a neuron has $x_i=0$, so this does not contribute to the error, the adjustment for this neuron is 0.

The main limitation of the delta rule is that it is impractical for multilayer networks. This is because the delta rule is based on the difference between desired output and real output, so this rule is useful only for the adjustment of the weights related to neurons on the output layer but does not adjust the weights related to neuron on hidden layers. This problem is overcome by means of the backpropagation algorithm. This algorithm is the most used in NN learning phase and is based on the following steps:

- first, the weights are initialized by means of random values and, for each example of the training set, the signals go from input to output in order to evaluate the answer of the NN;
- then, the error signals are propagated back, on the same channels in which the signals have been transmitted in the first step and in this second step the weights are adjusted.

Usually, in multilayer networks the sigmoid activation function is employed, because it is differentiable and this is a requirement of the backpropagation algorithm.

In unsupervised learning, instead, the NN is not provided with the correct results during training.

The presentation of neural networks theory here provided is, of course, only qualitative and introductory. For a complete examination of these soft computing techniques, one can refers to specialized literature and texts, such as [72], [79], [80] and [81].

Some applications in literature of neural networks with reference to trajectory generation problems are the ones reported in [68], [69] and [71]. These applications will be described in section 4.
3. FORMULATION BASED CLASSIFICATION OF ON-LINE TRAJECTORY GENERATION PROBLEMS

In general, the trajectory generation may be regarded as a typical optimization problem. Regardless of the particular kind of the trajectory generation problem examined, a first high level classification of the optimization problems in general may be identified based on:

- the presence or absence of constraints (constrained or unconstrained problems);
- number of allowable solutions (determined or over-determined problems);
- type of variables (continuous, discrete or mixed problems);
- analytical form of the model considered (linear or non linear problems);
- number of variables considered (small scale, for instance up to 5 variables, intermediate scale, for instance from 5 to 200/400 variables or large scale, for instance over 200/400 variables, problems).

In the preceding section we have emphasized the numerical methods employed in the field of trajectory generation, so we have implicitly covered the full spectrum of classification earlier mentioned, analyzing methods for both constrained and unconstrained problems, in linear and non linear form, with continuous, discrete or mixed variables and so on.

As we have before pointed out, however, the main concern of this survey on methodologies for the on-line trajectory generation is not the complete analysis of the solution methods used in this field but is the description of the problem formulations most usually employed in literature, since the solution method to be used is function of the problem formulation.

From this point of view, an high level classification is between geometric and dynamic formulation. This is due to the fact that usually the planning stage has either been done from a purely geometrical point of view without regarding the dynamic constraints of the system or by numerical optimization including the dynamic constraints and input bounds. To these two categories, nevertheless, it must be added a third, the one which refers to trajectory problems formulated as optimal control problems, which includes, in the most general case, the consideration of both dynamic constraints and controls acting on the vehicle.
Of course, these categories does not comprise all the possible situation which can be encountered in literature, yet they are anyway a very useful classification scheme from the point of view of trajectory generation problem formulation.

In conclusion, we have adopted the following formulation based classification of trajectory generation problems:

- geometric formulation (trajectory generation problem is based on a purely geometrical point of view, without regarding the dynamic constraints of the system);
- dynamic formulation (trajectory generation problem takes into account the dynamic constraints of the system);
- optimal control formulation (trajectory generation problem is formulated as an optimal control problem, taking into account the dynamic constraints of the system as well as the controls which act on the vehicle).

In the rest of this section, we propose a description of these categories; to make simpler the reading, in Figure 9 a schematic structure of this section is reported.

---

Figure 9 – Structure of section 3
3.1 **GEOMETRIC FORMULATION**

*Geometric formulations* of trajectory generation problems are based on a purely geometrical point of view, without regarding the dynamic constraints of the system.

Traditionally, path planning methods have used *computational geometry* to find obstacle-free geometric paths that may not be feasible given the dynamics of the plant. The planned path is only constrained with respect to such factors as the terrain obstacles or hostile areas. Given such a path, the tracking task of the controllers (either an autonomous controller or a human pilot), can be difficult if not impossible.

On the other hand, by considering the dynamics of the plant at the planning stage, the planned path or further the planned trajectory becomes less demanding for the controller to track but, however, including the dynamic constraints at the planning stage often leads to a time-consuming nonlinear dynamic optimization that may not be practical especially if the dynamic model is complex.

The main purpose of purely geometrical formulations, hence, is to obtain quickly and easily a trajectory joining a starting point and an ending point, with regards to geometrical respects (such as, for instance, the presence of obstacles between starting and ending point and so on) but with no regards to the dynamic capabilities of the vehicle to track the selected trajectory (such as, for instance, the minimum turning radius and so on). These observations clearly shown that the advantage of geometric classification is the consideration of an optimization problem which is in general more simple in the formulation, so probably also more simple in solving. This implies that, probably, to solve a purely geometric trajectory generation problem requires a small computational time, which makes this formulation particularly suitable for real time applications. However, geometric formulation has the serious drawback that the trajectory selected as optimal may be unfeasible, because it may be inconsistent with the unavoidable constraints affecting the dynamic behavior of the vehicle.

In order to eliminate or at least mitigate this problem, geometric formulations with some dynamic constraints, that are *mixed geometric-dynamic formulations*, can be employed. Examples of this mixed formulation are reported in [12] and [59] and another example of this approach, but in an interesting probabilistic frame, is provided in [67]. These papers will be described in section 4 and, more in details, in appendix.
Furthermore, some applications found in literature of the geometric formulation of trajectory generation problems are the ones reported in [56] and [57]; these applications will be described in section 4.
3.2 Dynamic formulation

Dynamic formulations of trajectory generation problems take into account the dynamic constraints of the system.

In contrast to the geometric planning methods, these numerical optimization planning methods directly include the dynamic system model while finding a path which optimizes a performance criterion. Realistic input/output and state bounds can be included in the optimization routine due to its numerical nature. Assuming the system model is accurate enough, the resulting path is optimized with respect to the given criteria and can be easily applied to a real system because it satisfies the dynamic constraints.

Though variations exist, as emphasized by Betts in [1], all calculus based methods utilize a Newton-based iteration to adjust a finite set of variables. The author points out that optimal solutions of this type to practical engineering problems seem only to be found numerically. Since the dynamic model of the system is included in the path generation routine, the scheme is fundamentally complicated and numerically intensive. Furthermore, analytical exact solutions of dynamical optimization problems are only possible when the system equations, the performance index and the constraints of the problem are very simple.

If the trajectory generation problem is expressed by means of a dynamic formulation, the use of calculus based methods to solve this optimization problem determines the input necessary to satisfy the optimality criteria given certain constraints. The complexity of the numerical method can be relieved, for instance, by assuming the input to be a parameterized sinusoidal, polynomial or piece-wise constant function, then applying it to a system model transformed into an appropriate form. Although the resulting trajectory is not optimal, an exact analytic result can be quickly obtained.

These observations clearly show that dynamic formulation is more consistent with physical behavior of aerial vehicles (and moving objects in general) than geometric formulation. This is due to the circumstance that dynamic formulation takes into account the dynamic constraints which affect the vehicle, so the optimal trajectory determined is certainly feasible for the airplane.

Of course, the dynamic formulation can be considered with different detail levels, in the sense that one can use a more accurate and comprehensive formulation of dynamic constraints or can use a more approximated formulation. In the first case,
the formulation (and, hence, the solution method to be used) will be more complicated and computational expensive than in the second case, even if the physical behavior of the system will be better taken into account.

The drawback of this formulation, yet, is the more complicated analytical form of the optimization problem, which forces to make use of complicated algorithms, such as the ones described in the previous section, to solve the problem. In accordance with the complexity of the dynamic formulation adopted, moreover, the optimization problem may be impossible to solve, for instance because the solution algorithm does not converge, and the time required for the convergence is anyway not foreseeable. These drawbacks make dynamic formulations unadvisable for real time applications, in which the certainty of the convergence, that is the certainty of the determination of a trajectory for the vehicle, and the reliable estimate of the time (as small as possible, of course) required to determine this trajectory are the predominant concerns.

These drawbacks have stimulated the consideration of approach which, although take into account the dynamical constraints related to physical behavior of vehicles, allow however the certain determination of an optimal or near optimal trajectory for the vehicle and require an almost fixed and small computational time to reach this result, making them suitable for real time applications. These approach to dynamic formulation of trajectory generation problems may be for instance the ones consisting in use soft computing techniques, such as genetic algorithms and so on. This is only a working hypothesis, nevertheless the interest in these topics is growing.

**Receding Horizon Control approach to solve trajectory generation problems using dynamic formulation**

At this point it’s worth to emphasize some concepts with reference to the receding horizon control (RHC) approach.

Improvements in UAV capabilities make it possible for UAVs to perform longer and more complicated missions and scenarios. In these missions and scenarios, optimal path navigation through complicated environments is crucial to mission success. As more vehicles and more targets are involved in the mission, the complexity of the trajectory design problem grows rapidly, increasing the computation time to obtain the optimal solution [31].
One alternative to overcome this computational burden is to use receding horizon control. The RHC uses a plant model and an optimization technique to design an input trajectory that optimizes the plant’s output over a period of time called the “planning horizon”. A portion of the input trajectory is then implemented over the shorter execution horizon, and the optimization is performed again starting from the state that is to be reached. If the control problem is not completed at the end of the planning horizon, the cost incurred past the planning horizon must be accounted for in the cost function.

The selection of the terminal penalty in RHC design is a crucial factor in obtaining reasonable performance, especially in the presence of obstacles and no-fly zones. In general, the cost function of a receding horizon optimization problem estimates the cost-to-go from a selected terminal state to the goal. The length of a path to the goal made up of straight line segments can be used as its cost-to-go. This is a good approximation for minimum time of arrival problems since the true minimum distance path to the goal will typically touch the corners of obstacles that block the vehicle’s path.

In order to connect the detailed trajectory designed over the planning horizon and the coarse cost map beyond it, the RHC selects an obstacle corner that is associated with the best path. This approach has another advantage in terms of real-time applications. The cost map not only gives a good prediction of vehicle behavior beyond the planning horizon, but because it is very coarse, it also can be rapidly updated when the environment and/or situational awareness changes.

An application of RHC to solve dynamic formulated trajectory generation problems is presented in [64]; this paper will be described in section 4 and, more in details, in appendix.

Some applications in literature of the dynamic formulation of trajectory generation problems are the ones reported in [27], [31], [32] and [34]; these applications will be described in section 4.
3.3 **Optimal control formulation**

Another way to formulate the trajectory generation problem is the one consisting in using an optimal control framework. In optimal control formulations, the trajectory generation problem is formulated as an optimal control problem, taking into account the dynamic constraints of the system as well as the controls which act on the vehicle.

Optimal control (also referred to as dynamic optimization) nowadays is a widely used tool and framework for various types of analyses, as for instance the ones regarding aerospace applications in general and trajectory optimization problems in particular. The theory and framework of optimal control allows analysis of problems in which a dynamic system is to be controlled in an optimal manner, according to some performance index. The dynamics describes the evolution of the system state and how the controls affect it; the performance index is a functional of the state and the control and gives the cost to be minimized or utility to be maximized.

In order to formalize the general concept of optimal control just expressed, let system dynamic be modeled by the vectorial differential equation and constraints [84]:

\[
\begin{align*}
\dot{x} &= f(x(t), u(t), t) \\
\text{subject to} & \quad X_{\text{min}} \leq x \leq X_{\text{max}} \\
& \quad U_{\text{min}} \leq u \leq U_{\text{max}} \\
& \quad x(t_0) = x_0 \\
& \quad t \in [t_0, t_f]
\end{align*}
\]

(58)

where \( x \in \mathbb{R}^n \) is the state variables vector and \( u \in \mathbb{R}^m \) is the control variables vector. If we define the cost functional:

\[
J(u) = \varphi[x(t_f), t_f] + \int_{t_0}^{t_f} L[x(\tau), u(\tau), \tau] d\tau,
\]

(59)
where $L$ is a scalar valued function representing the performance index and $\varphi$ is a
terminal boundary conditions, the aim of the optimal control problem is to minimize $J$ with respect to $u$, subject to constraints (58).

The two main theoretical approaches to control optimization problems are [85]:

- Bellman’s dynamic programming method (DP);
- Pontryagin’s maximum principle (PMP).

Of course, the aim of this survey and of this section in particular is not the
detailed explication of solution methods for dynamic optimization, so one can refers
for an analytic exposure of these methods to specialized literature (such as, for
instance, [51] and [85]) or books (such as, for instance, [82], [83] and [84]). We
here just emphasize some aspects regarding optimal control theory in general and
applications to trajectory generation problems.

The DP is a method based on the Bellman’s principle of optimality, illustrated in
the remainder of this section. This method was originally developed for discrete-
time problems and then extended to continuous-time problems. In the latter form,
it is often referred to as the Hamilton-Jacobi-Bellman theory and its major
disadvantages lies in the large computer memory requirements.

The PMP is in practice an extension and application of the classical calculus of
variations to the optimal control problem. This method was originally developed for
continuous-time problems and then has been extended to discrete-time problems.
Major disadvantages of Pontryagin’s maximum principle method is that it provides,
in general, only local necessary conditions for optimality. Moreover, its
computational requirements, although non trivial, are not as severe as those
associated with dynamic programming.

In [51] a very interesting introduction to historical development of optimal
control theory (OCT) is provided. In this work, the author point out that the two
different methodologies used in studying optimal control problems have both
advanced from the analytical tools provided by “nonsmooth analysis”. In aerospace
applications, non-smoothness appears in examples such as launch problems where
discontinuity of state variables (mass) or dynamics are expected. In optimization
problems any cost function that depends on something as simple and natural as the
distance function requires nonsmooth formulation. Nonsmooth analysis,
therefore, is motivated not just by natural occurrence of nonsmooth (non-
differentiable) phenomena, but also by its usefulness in analyzing smooth problems
in a more general setting.
Regarding in particular the dynamic programming, it refers to the solution method developed by Bellman [82], based on the principle of optimality, which states that on an optimal path each control is optimal for the state at which it is executed, regardless of how that state was arrived at and is valid for continuous as well as for discrete time problems.

The dynamic programming method, when extended to continuous time problems, leads to the Hamilton-Jacobi-Bellman equation, which is a partial differential equation defining the optimal cost to go function, i.e., performance index value from current time to the end, on the optimal trajectory. For continuous time optimal control problems, the necessary conditions of optimality are provided by the Pontryagin maximum principle [36], which relates the optimality of the control to minimizing or maximizing the Halmiltonian function of the problem at each time instant by the control value subject to control constraints. These necessary conditions define a two point boundary value problem for the dynamics of the system and the adjoint states, also known as “co-states”. Analytical as well as indirect numerical methods for solving optimal control problems are based on the maximum principle.

The solution of an optimal control problem can be found either by solving the boundary value problem formulated by the optimality conditions of the maximum principle, by application of dynamic programming, or by direct optimization of the objective functional.

For example, if the dynamics of a discrete time problem with finite number of time steps is given in state space form, the problem can easily be written as a parameter optimization problem with the state transition equations as constraints, or the dynamics can be inserted into the objective function, so that only controls are to be chosen. For a continuous time problem, the solution can be found approximately by representing the state and control functions by a finite number of parameters, and thus transcribing the problem into a parameter optimization problem, as shown for instance in [7].

Finding a solution in closed form for an optimal control problem, by any method, is usually possible only for very simple problems. Also, the solution is often available only in the form of open loop control, i.e., the solution applies only to a given initial state instead of being in feedback form in which the control is related to the current state. A special situation is if the dynamics is linear, the objective functional is quadratic and there are no constraints, except for fixed initial state. Then, a closed loop formulation of the control is available, it is a linear function of
the state and the problem and solution are referred to as \textit{linear quadratic (LQ)} problem and control [30].

A computationally efficient technique for the numerical solution of optimal control problems is discussed in [20]. This method utilizes tools from nonlinear control theory to transform the optimal control problem to a new, lower dimensional set of coordinates. It is hypothesized that maximizing the relative degree in this transformation is directly related to minimizing the computation time. Firm evidence of this hypothesis is given by numerical experiments.

\textbf{Direct and indirect methods for solving optimal control problems}

In the so-called \textit{direct method of solution of optimal control problems}, either the state variable time history or the control variable time history, or both, of the continuous problem are discretized. The problem then becomes a parameter optimization problem. The system-governing equations may be satisfied by explicit numerical integration or implicitly, by including nonlinear constraints, which are in fact quadrature rules.

A method termed \textit{differential inclusion} has been recommended [8] for the solution of certain classes of such problems because it reduces the size of the parameter optimization problem. It does this by removing bounded control variables in favor of bounds on attainable time rates of change of the states. The smaller problem is then in principle solved more quickly and reliably. In [8] it is demonstrated analytically and with several computed problem solutions that differential inclusion, because it requires the use of an implicit quadrature rule with the lowest possible order of accuracy, i.e., Euler’s rule, yields larger rather than smaller nonlinear programming problems than direct methods, which retain the control variables but use much more sophisticated implicit quadrature rules.

In practice, direct methods for the solution of continuous optimal control problems are methods that do not explicitly employ the necessary conditions for optimality. They are becoming widely used for many reasons, the principal one being that the indirect approach, which introduces Lagrange multipliers (or co-states) and yields a two-point-boundary-value problem, is very difficult to solve for all but the simplest problems. In the direct method the control time history and/or the state variable time history is discretized and the satisfaction of the system-governing equations can be ensured in several ways. The system-governing equations are integrated forward numerically using the discrete control time
history. The problem becomes a Nonlinear Programming problem (see section 2) if, as it is typically the case, the initial and/or terminal constraints or the performance index are nonlinear functions of the states and controls.

*Collocation-based methods* discretize both the control and state variable time histories, i.e. the states and controls are known only at discrete points, typically the beginning and end of each time segment into which the total time is subdivided (referred to as the nodes of the discretization). The system-governing equations are satisfied by including nonlinear constraint equations (the defects) for each state and in each time segment with the constraints representing quadrature across the time segment. The collocation methods thus integrate the system equations implicitly. Collocation methods typically use many NLP variables and constraint equations, but because many thousands of trajectories do not have to be integrated numerically significant execution, time is saved.

*Indirect methods* are characterized by explicitly solving the optimality conditions stated in terms of the adjoint differential equations, the maximum principle and associated boundary (transversality) conditions. Using the calculus of variations, the optimal control necessary conditions can be derived by setting the first variation of the Hamiltonian function to zero. The indirect approach usually requires the solution of a nonlinear multipoint boundary value problem. By analogy, an indirect method for optimizing a function of n variables would require analytically computing the gradient and then locating a set of variables using a root-finding algorithm such that the gradient is zero. There are three primary drawbacks to this approach in practice:

- first, it is necessary to derive analytic expressions for the necessary conditions, and for complicated nonlinear dynamics this can become quite daunting;
- second, the region of convergence for a root-finding algorithm may be surprisingly small, especially when it is necessary to guess values for the adjoint variables that may not have an obvious physical interpretation;
- third, for problems with path inequalities it is necessary to guess the sequence of constrained and unconstrained subarcs before iteration can begin.

In contrast, a direct method does not require an analytic expression for the necessary conditions and typically does not require initial guesses for the adjoint variables; instead, the dynamic (state and control) variables are adjusted to directly optimize the objective function.
Some applications in literature of the dynamic formulation of trajectory generation problems are the ones reported in [23], [62] and [63]. In [49] and [52], furthermore, theoretical methods for solving a wide variety of optimal control problems are presented. Paper [63] will be examined in section 4 and paper [49] will be described in appendix.
4. Review of Applications in Literature

In this section, a wide review of applications found in literature is provided. This review covers all the categories which we have identified in the previous sections, regarding both numerical optimization methods and formulations. In this section, however, we attempt to organize the wide literature devoted to trajectory generation problems from a point of view which emphasizes the typology of the application considered. For each field of application that we characterize, we furnish also some examples from the literature.

From our literature analysis, we have identified some relevant typologies of applicative problems. In general, a trajectory generation problem always consists in practice in finding the path which connects two points and, at the same time, allows to minimize a certain cost function. The points connected from this trajectory may be the starting and ending points of the whole path considered or may be specified waypoints defining the whole path of interest. The various trajectory optimization problems in literature differs in complexity because the cost function considered and the constraints imposed to the optimization problem may involve different levels of complexity.

In this section, we emphasize that trajectory generation problems may consider only one UAV or more UAVs; furthermore, the trajectory generation problem may consider or not the presence of obstacles, which may be fixed obstacles or dynamic threats. Based on these observations, in this section we attempt to classify the applications found in literature depending on the complexity level considered.

Hence, we distinguish four classes of applications, described as:

- **Level I** - The trajectory generation problem consists in connecting two specified points minimizing a cost function, considering a single UAV and without obstacles;
- **Level II** - The trajectory generation problem consists in connecting two specified points minimizing a cost function, considering a single UAV in presence of fixed obstacles;
- **Level III** - The trajectory generation problem consists in connecting two specified points minimizing a cost function, considering a single UAV in presence of dynamic threats;
- **Level IV** - The trajectory generation problem consists in connecting two specified points minimizing a cost function, considering multiple UAVs and taking into account collision avoidance and team coordination.
A schematic representation of the UAV trajectory generation applications is reported in Figure 10.

<table>
<thead>
<tr>
<th>Level I</th>
<th>Single UAV obstacle free trajectory optimization between two fixed points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level II</td>
<td>Single UAV trajectory optimization between two fixed points in presence of fixed obstacles</td>
</tr>
<tr>
<td>Level III</td>
<td>Single UAV trajectory optimization between two fixed points in presence of dynamic threats</td>
</tr>
<tr>
<td>Level IV</td>
<td>Multiple cooperative UAVs trajectory optimization with collision avoidance</td>
</tr>
</tbody>
</table>

**Figure 10 – Trajectory generation problems for UAVs applications**

Furthermore, some applications not directly referred to UAVs are also considered in this section, because they may be interesting in unmanned aerial vehicles field too.

In order to summarize the point of view that we have described and present the structure of this section, in Figure 11 a schematic representation is shown.

The attempt which we have made is to present the review of literature works in an homogeneous manner, although, of course, there are in literature a number of different approach to the problem of trajectory generation, so it is not always possible to classify exactly an approach. The schematic organization which we furnish in this document, nevertheless, is surely a satisfactory way to comprise and organize most of the papers in literature.
Applications of trajectory generation

UAVs

Level I
(Section 4.1)

Level II
(Section 4.2)

Level III
(Section 4.3)

Level IV
(Section 4.4)

Others
(Section 4.5)

Figure 11 – Structure of section 4
4.1 **LEVEL I APPLICATIONS**

As earlier said, level I applications are the ones in which a single UAV is considered and the problem of optimal trajectory generation is examined without collision avoidance.

**Level I application n.1**

An example of this application in literature is provided in [27]. In this paper a single UAV which flies from a starting point to an ending point in obstacle free environment is considered. The formulation employed to formalize the problem is the dynamic one (see section 3.2) and in the paper is presented a way to generate a feasible trajectory for an input-output linearizable system while satisfying the dynamic constraints, that is input and state constraints.

The system model is transformed into a linear form, then suitable bounds are obtained for the linear coordinates. Within the linear coordinates, an optimal trajectory is found using LP techniques (see section 2.1) and this trajectory is then transformed back to the original nonlinear coordinates without violating the bounds given on the original nonlinear system. Although the resulting trajectory is not optimal due to the transformation and the limiting choice of the bounds, one can easily and quickly obtain a reasonable solution, as is shown in an helicopter trajectory generation example performed in the paper.

It must be noted that systems such as spacecrafts, helicopters, airplanes and cars are often input-output linearizable, so the described method may be used to find feasible trajectories for these and other autonomous systems. When the method works, it can be used to quickly find trajectories which connect any pairs of points. If a geometric planner is used to find a set of waypoints which connect the start and goal points and are obstacle-free, the planning method presented in [27] could be used to find feasible trajectories connecting these waypoints.

**Level I application n.2**

Another paper which proposes a level I trajectory generation application is [55]. In this paper, the application considered consists in the trajectory generation and guidance for an UAV in order to connect two assigned waypoints, in accordance with specified constraints in terms of crossing speed and heading angle and without obstacles. In this work is presented a fuzzy logic based controller (see section 2.4)
devoted to the guidance of the UAV and the proposed procedure is based on the assumption that the aircraft is equipped by an autopilot system which takes the references from a fuzzy controller.

In particular, the problem formulation considers an aircraft provided by a flight control system with an inner autopilot loop for tracking commanded velocity, flight path and heading angle. This inner control system is assumed to be able to track desired velocity, flight path and heading angle with a first order dynamic behavior. The state variables of the autopilot model are the velocity, the flight path angle and the heading angle whereas the inputs of the autopilot control system are the desired values of these state variables. The fuzzy controller is an outer loop which generates these references for the autopilot control system in order to reach fixed waypoints with fixed crossing heading and velocity. The constraints of this guidance problem are the angular rate limitations of the aircraft.

The solution method proposed in [55] is based on a fuzzy guidance system, which provides the references to the inner autopilot loop. This fuzzy guidance system is based on Takagi-Sugeno fuzzy sets, in order to obtain an acceptable computational load. In the application proposed in the paper, it uses membership functions which are combination of Gaussian curves. The fuzzy rules are defined according to the desired approach direction and the angular rate constraints of the UAV. Since the aircraft is considered equipped by three closed loop inner autopilots, one for the velocity another for the flight path angle and the last for the heading angle, the outer fuzzy controller is constituted by three independent controllers, one for each reference: the first generates the desired flight path angle based on the altitude error, the second the velocity reference based on the velocity error and the third generates the desired heading angle based on the position errors in the reference horizontal plane and on the heading angle error.

The velocity fuzzy controller uses three fuzzy sets for both the velocity error input and the velocity reference correction output. The flight path angle controller uses four fuzzy sets for both the altitude error input and the desired path angle output. The heading angle controller uses five fuzzy sets for the error input referred to the X-axis of the reference horizontal plane, three sets for the Y-axis error input and seven fuzzy sets for the heading error angle input; in the application, seventy fuzzy rules are defined for the heading angle controller.

The authors state that the numerical applications performed on both a linear decoupled aircraft model and a fully non linear autopiloted aircraft model have
shown the effectiveness of the proposed fuzzy approach. The problem examined refers to the trajectory generation in order to reach fixed waypoints with fixed specifications in terms of crossing heading and velocity, without considering in detail the problem of trajectory tracking except for the inclusion of dynamic constraints (angular rate limitations) in the trajectory path generation problem.

This paper will be described also in appendix.

**Level I application n.3**

Another example of level I application is the one provided in paper [57]. In this work the authors consider the problem of trajectory generation in order to connect assigned waypoints with the only constraint of continuity. In practice, this approach can be applied for instance to aircraft maneuvering to land at an airport.

In [57] a geometric formulation (see section 3.1) is considered, consisting in following predetermined discrete waypoints for aircraft landing maneuvers by optimizing cubic splines to minimize acceleration, curvature or to obtain constant velocity. Although the method suggests that the trajectory is more efficient than conventional straight-line flight trajectories, the advantage of the cubic splines may need more investigation due to its geometry oriented approach.

The authors point out that, for certain aircraft, the trajectory may be hard to follow, suggesting that the minimum turning radius of the target aircraft must be small enough to follow the path at all.

The main limitation of these geometric planning methods, hence, is that they do not take the dynamic model of the vehicle into account; only a geometrically feasible path is given. This means that, before it can be used by a dynamic system such as an airplane, the feasibility of the devised path needs to be investigated. In some cases, the geometric path may be simply impossible to follow for systems with certain dynamic constraints.

At most, it is true that, if the system is controllable, any geometric path can be approximated arbitrarily closely by a feasible path which satisfies the dynamic constraints, nevertheless, since this methods relies on a given geometric path as the targeting reference, the resulting feasible path may be awkward to follow for a real system.
4.2 LEVEL II APPLICATIONS

Level II applications are the ones in which a single UAV is considered and the problem of optimal trajectory generation is examined in association with the one of collision avoidance in presence of fixed obstacles; a particular class of these applications is the one of terrain following problems, where the vehicle must follow the terrain profile.

Level II application n.1

A first example of level II application is the one considered in paper [45], in which an evolutionary algorithm (see section 2.3) based framework is utilized to design an offline/online path planner for unmanned aerial vehicles autonomous navigation. The path planner calculates a curved path line with desired characteristics in a three-dimensional (3-D) rough terrain environment, represented using B-Spline curves, with the coordinates of its control points being the evolutionary algorithm artificial chromosome genes. Given a 3-D rough environment and assuming flight envelope restrictions, two problems are solved:

- UAV navigation using an offline planner in a known environment;
- UAV navigation using an online planner in a completely unknown environment.

The offline planner produces a single B-Spline curve that connects the starting and target points with a predefined initial direction. The online planner, based on the offline one, is given on-board radar readings which gradually produces a smooth 3-D trajectory aiming at reaching a predetermined target in an unknown environment; the produced trajectory consists of smaller B-Spline curves smoothly connected with each other.

As usual, in the trajectory generation problems, the final destination (end-point coordinates) is known and the UAV must follow an as smooth as possible trajectory (imitating real flight restrictions), planned and re-planned in real-time, avoiding static (mountains) and moving obstacles, given its initial position and initial flight direction. The vehicle is assumed to be a point.

More in detail, in the problem of UAV navigation using an offline planner considering a known 3-D environment, the offline planner generates collision free paths in environments with known characteristics and flight restrictions. The derived path line is a single continuous 3-D B-Spline curve, while the solid boundaries are interpreted as 3-D rough surfaces. The air vehicle course is actually...
a curve with curvature continuity that cannot be modeled using straight-line segments (which is the usual practice for ground robots). Therefore, B-Spline curves based path representation is utilized having the advantage of being described using a small amount of data (the coordinates of their control points), although possibly producing very complicated curves.

In the problem of UAV navigation using an online planner considering a completely unknown 3-D environment, the online planner uses acquired information from the UAV on-board sensors, that scan the area within a certain range from the UAV. The online controller rapidly generates a near optimum path that will guide the vehicle safely to an intermediate position within the sensors range. The process is repeated until the final position is reached and the path line from the starting point to the final goal is a smooth, continuous 3-D line that consists of successive B-Spline curves, smoothly connected to each other.

The UAV path-planning problem is considered within an evolutionary algorithm (EA) context, in which the path line is represented using B-Spline curves, with the coordinates of its control points being the EAs artificial chromosome genes. The reasons behind choosing EAs as an optimization tool for the path-planning problem are their high robustness compared to other existing directed search methods, their ease of implementation in problems with a relatively high number of constraints, and their high adaptability to the special characteristics of the problem under consideration. As examples of application of the evolutionary procedure applied to a trajectory generation problem, in Figure 12 some trajectories resulting from this procedure are shown, which demonstrate the ability of the method in dealing with difficult terrain shapes.

This paper will be described also in appendix.
Level II application n.2

Another example of level II application is the one reported in [59]. The problem examined in this paper is the on-line trajectory optimization for an UAV, with particular reference to the optimization of the terrain following trajectory, taking into account the threat avoidance constraint. The authors propose two formulations for the optimization problem described above. In both formulations, which are different regarding to the problem approximation level, the aircraft is considered as a point of mass with 3 degrees of freedom and the dynamic system consists in the kinematics equations of motion.

In the first (simplified) formulation, the equations of motion are written in a local level frame, considering only the horizontal component of velocity and assuming that the vertical component is small. In the second formulation, in which the aircraft is considered flying in the plane tangent to the local terrain, this assumption is eliminated and the total velocity is taken into account. In both formulations, a constant energy approximation is used, instead of the more usual constant velocity approximation. The control variable is the local heading angle and the objective function is formulated accounting for the flying time, the terrain and threat, allowing a choice of the preferred optimization criterion by which a weighting parameter. The limitation of the turn rate of the vehicle is taken into account as the only constraint and is formulated in two different ways.

For the problem solution an indirect method based on the application of the Pontryagin Maximum Principle is proposed (see section 3.3). This method leads to
different systems of differential equations to be solved according to the different problem and constraint formulations.

Paper [59] will be also examined in appendix.

**Level II application n.3**

Paper [67] provides a third example of level II application, in which the problem of optimal trajectory generation for an UAV flying from a starting point to an ending point avoiding fixed obstacles is considered. The one presented in this paper is an interesting application of the mixed geometric-dynamic formulation (see section 3.1) in a probabilistic frame. In this paper the problem examined is the mission planning for an UAV flying through an area in the presence of a multiple source of threat, in order to go to a target location following the shortest possible path with an acceptable probability of getting disabled or an acceptably long path with the lowest possible probability of getting disabled.

The author considers simply a point of mass which follows a path. In the path planning the heading angle is used as control variable and the maximum heading angle allowed is the only constraint taken into account. Two point of view are regarded: the length and the risk of the path, which may be taken into account at different levels. The proposed solution method is based on the evaluation of the conditional probability that the UAV is disabled when following a certain path, considering the upper limit of this conditional probability as performance index in the optimal path search.

For the solution the author proposes a multi-step method, based on the evaluation of the probability of getting disabled along different directions, choosing the step in the direction corresponding to the smaller probability and respecting the constraint on the maximum heading angle. The author emphasizes that, if the maximum heading angle allowed is too small, the path generated by this strategy might get in a limit cycle close to the target position but not close enough to attain it. In order to avoid this situation, the author proposes a modification in the strategy, which is explained in the paper.

Paper [67] is analyzed more in details in appendix.

**Level II application n.4**

Another example of level II application is the one proposed in [63]. In this work, the authors refer to trajectory planning of dynamic systems to satisfy explicitly the
dynamic equations and inequalities on states and inputs; this problem is considered making use of the optimal control formulation (see section 3.3). As example of application of the trajectory planning methodology proposed in the paper, the trajectory planning problem for a planar vertical takeoff and landing (PVTOL) aircraft is considered.

The author states that it is well known that most approaches to optimal control for nonlinear systems are computational intensive and are usually not suited for real-time implementation. The optimal control problem is here applied to differentially flat systems: these include controllable linear systems as well as nonlinear systems linearizable by static and dynamic feedback. Previous studies on differentially flat systems were restricted to cases where the motion wasn’t subject to inequality constraints. This paper extends the planning in presence of auxiliary constraints.

Paper [63] will be discussed more in details in appendix.
4.3 Level III Applications

Level III applications are the ones in which the problem consists in optimal trajectory generation for a single UAV, which flies from an assigned point to another specified point in presence of dynamic obstacles.

Level III application n.1

In [68] an interesting level III application is considered. In this work the problem examined is the one of the path planning for an UAV in order to fly from a starting point to a target in presence of dynamic threats and the application of neural networks to this problem is proposed.

Really, this paper is only descriptive and considers a NN-based UAV which navigates in an hostile environment and reacts to unexpected dynamic treats. In order to achieve this objective, critical is the development of an intelligent autonomous system capable to operate in an unpredictable environment. The main task of this intelligent system is to modify on-line the trajectory of the UAV in order to route the vehicle around unexpected threats, so enhancing the mission success of UAVs flying straight lines between way points.

In the paper a NN system is described which is organized in such a way as the guidance and control function receives sensor information about the environment and the vehicle status. The sensor inputs are navigation information describing current and desired positions and the radar system which may detect the surface to air missiles (SAM). The self status inputs include fuel, weapon status and system damage. The outputs of NN are course and speed corrections and the control of other instruments on-board which are needed for the planned maneuvers and actions.

Of course, as described in section 2.5, two mode of operation are involved with NN. These mode of operation, as shown in Figure 13, are the “training mode”, performed on the mission planning workstation, and the “operating mode”, which may be either a performance evaluation on a workstation or the use on-board the UAV.
In the training mode, available data are applied to the network, which then uses NN learning algorithms to adjust the NN weights. In the testing, the network no longer learns, but rather is producing outputs based on its inputs to compare with known cases. The final trained NN is then ready for field or simulator use.

In the operating mode, the neural network evaluates the situation at each step and provides course corrections and other controls.

In simulation mode, the authors have performed a quantitative estimate of mission success based on probability of kill and this estimate has demonstrated the effectiveness of the NN in improving the probability of success of the UAV in performing missions in hostile environments. In Table 1 the results of simulations performed by the authors in [68] are shown, in which two different behavioral characteristics of the NN based UAV (corresponding to the columns “aggressive driving” and “careful driving”) are compared with a simple way point flying UAV (column “way point driving”); these simulations demonstrate that the NN based UAV performs better than a simple way point based UAV.

<table>
<thead>
<tr>
<th>Low Density Air Defense</th>
<th>Aggressive Driving</th>
<th>Careful Driving</th>
<th>Way Point Driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Missions</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Successful Returns</td>
<td>42</td>
<td>48</td>
<td>18</td>
</tr>
<tr>
<td>Reach Target Only</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>UAV Lost</td>
<td>8</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>Probability of Success</td>
<td>84.0%</td>
<td>96.0%</td>
<td>36.0%</td>
</tr>
<tr>
<td>Probability of Reaching Target</td>
<td>94.0%</td>
<td>98.0%</td>
<td>46.0%</td>
</tr>
</tbody>
</table>

Table 1 – Performances of NN based UAVs compared to way point based UAV [68]
Level III application n.2

Another level III application found in literature is the one reported in [64]. In this paper the problem considered is the one of trajectory planning, shooting and tracking for an UAV flying in presence of multiple dynamic threats, taking into account constraints for heading rate and velocity.

The proposed approach is applied to a simulation scenario, where an UAV is assigned to transition through several targets in the presence of multiple dynamic threats. The inherent properties of fixed-wing UAVs impose the input constraints of positive minimum velocity due to the stall conditions of the aircraft, bounded maximum velocity due to thrust limitations and saturated heading rate due to roll angle and pitch rate constraints. The architecture considered in this paper consists of five layers:

- the waypoint path planner;
- the dynamic trajectory smoother;
- the trajectory tracker;
- the longitudinal and lateral autopilots;
- the unmanned aerial vehicle.

With the UAV equipped with standard autopilots, the UAV model is assumed to be first order in heading and velocity and second order in altitude; suitable constraints are imposed on minimum and maximum values of velocity and heading rate. The authors believe that the utility of RHC is limited by its computation requirements and stability concerns, so they propose an approach based on a control Lyapunov function (CLF).

This paper will be analyzed more in particular in appendix.

Level III application n.3

Work [18] proposes another application in which the problem of optimal trajectory generation for a single UAV, in presence of dynamic threats, is considered. The methodology proposed in this paper would be used in the control of an Uninhabited Combat Air Vehicle (UCAV).

The desired objective of the UCAV would be commanded by the operator or preprogrammed without operator intervention. The UCAV would be required to generate trajectories in real-time and stabilize about them to meet the objectives. The objective might be to engage a target in a dynamically changing environment.
or to determine a trajectory that minimizes the cross-section of the UCAV in order to evade radar in a reconnaissance mission.

A scale model of an highly maneuverable UCAV is used as a test-bed to validate real-time trajectory generation methodology. The desired objective, similar to that of a real UCAV, is to track real-time commanded positions and velocities of the test vehicle in minimum time. This must be achieved while ensuring that the test vehicle is stabilized and operating within all flight and actuation constraints.

A simple approach to tracking a target trajectory using a feedback control system is to subtract the current state of the tracking system from the target and feed this error into a controller. This approach is frequently referred to as a one degree of freedom design. It is well known, for a large class of nonlinear mechanical systems with constraints, that this approach generally does not work well, since we are likely tracking a drifting equilibrium configuration which is an infeasible trajectory of the system. Furthermore, achieving high performance while guaranteeing stability in the presence of constraints is difficult.

Another approach to tracking a target is the two degree of freedom design depicted in Figure 14.

![Figure 14 - Two degree of freedom design](image)

The two-degree of freedom design consists of a trajectory generator and a linear feedback controller. For the experiment in [18], real-time trajectory generation has been combined with a feedback controller to form the two degree of freedom control system shown in Figure 14. The trajectory generator provides a feasible feed-forward control and reference trajectory in the presence of system and environment constraints. Given inherent modeling uncertainty, a feedback
controller (tracker) is necessary to provide stability around the reference trajectory. An advantage of this approach is that a stabilizing controller is provided with the feasible trajectory, not just the trajectory itself. Furthermore, it is possible to make the reference trajectory as aggressive as is allowed by the model.
4.4 **Level IV Applications**

Level IV applications are the most complicated ones and refers to multiple UAVs trajectory generation and collision avoidance, in presence of fixed as well as dynamic obstacles and threats.

**Level IV application n.1**

A first example of level IV application in literature is the one reported in [32]. This work is a thesis in which methods for finding optimal trajectories for vehicles subjected to collision avoidance and team coordination requirements are presented.

The major application areas considered for the methods proposed in [32] are aircrafts and spacecrafts. In the first case, the possible applications of the methods proposed in [32] are for instance minimum flight-time problems, air traffic management and coordination of autonomous UAVs team. In the second case, the possible applications of the methods proposed in [32] may be for instance minimum fuel optimization, international space station rendezvous and satellite cluster configuration.

The problem formulation used in this work is in practice the dynamic formulation (see section 3.2). In this thesis is firstly emphasized that optimizing a kinematically and dynamically constrained path is a significant problem in controlling autonomous vehicles and has received attention in the fields of robotics [25], [26], [27], [28], [29], underwater vessels [56], and aerial vehicles. Planning trajectories that are both optimal and dynamically feasible is complicated by the fact that the space of possible control actions is extremely large and non-convex and that simplifications reducing the dimensionality of the problem without losing feasibility and optimality are very difficult to achieve.

Starting from these observations, in [32] the author proposes the use of mixed-integer linear programming (see section 2.1) in off-line trajectory design for vehicles under various dynamic and kinematic constraints, since MILP allows the inclusion of non-convex constraints and discrete decisions in the trajectory optimization. Binary decision variables, for instance, allow the choice of whether to pass left or right of an obstacle and so on.

The particular feature of the problems examined in [32], hence, is the inclusion of non-convex constraints, which arise from two requirements: avoidance and assignment. Avoidance, as well known, is the requirement to remain outside certain regions of the solution space, such as those which would lead to collisions.
Assignment refers to the inclusion of variable boundary conditions (initial or terminal), subject to logical constraints.

Assignment problems involve discrete decisions, such as “go to point A or point B”, and therefore lead to non-convex constraints. Their non-convexity makes these problems intrinsically difficult to solve, because they are essentially combinatorial and therefore can lead to very large solution spaces. Furthermore, the combined problems of path-planning and assignment are tightly coupled: the assignment is strongly dependent on the paths taken and to decouple the problems would require computing paths for all possible assignments, which is prohibitive.

The applications in aircrafts field possible for the procedure proposed in [32] are air traffic management and autonomous unmanned aerial vehicles and both areas require path-planning methods for multiple vehicles, avoiding obstacles and each other. Future air traffic concepts involve “free-flight,” in which flight planning and conflict resolution are performed on-board. UAV problems often involve the additional complexity of waypoint assignment.

For instance, for small groups of vehicles which operate autonomously, with the high-level goal of visiting a set of waypoints, the allocation of waypoints is to be determined, according to capability and timing constraints. This assignment is required to optimize some metric for the mission, such as minimum time, maximum reward, or minimum risk. All of these objectives are highly dependent on the paths taken, since this determines the order of events, the flight time and the risk exposure. This makes the two problems of assignment and path-planning strongly coupled.

The procedure proposed in [32] in order to solve the problems discussed considers three major steps:

- representation of non-convex trajectory optimizations as integer programs;
- development of efficient approximations to enable these optimizations to be solved quickly;
- use of the optimizations in a real time, model predictive control (MPC) scheme to compensate for uncertainty, such as noise and disturbance.

Solving a single MILP optimization generates an optimal trajectory for the problem seen at a particular instant. In practice, this “snapshot” will be subject to uncertainty and the actual problem to be solved will change as time progresses. Furthermore, the model of the system used in the optimization will be imperfect.
and the system state will not evolve exactly as predicted. The single MILP is an open-loop solution and cannot account for these uncertainties.

For application to real-time control, the MILP trajectory optimizations are embedded within MPC scheme. This is a scheme in which trajectory optimization is performed at each time step, finding a solution to complete the problem from the current position. Only the first step of the resulting control sequence is implemented and the process is then repeated. This incorporates feedback in the control, allowing it to compensate for uncertainties such as model error, disturbances and noise. It also introduces new challenges into the problem: it is required to run in real-time and the stability of the resulting controller must be demonstrated.

The MILP approach has two major drawbacks:

- the intensive nature of the computation, which is centralized and scales poorly with problem size;
- the restriction to linear problems.

With reference to the first one, it must be emphasized that a MILP representation of trajectory problems can involve many binary variables, typically thousands, which results in a very large computational time. This implies that it must simplify the MILP trajectory optimization problem in order to accelerate the solution process. In [32], methods are presented for using prior knowledge of the solution to identify redundant or inactive constraints before solving the problem.

In conclusion, in [32] the author demonstrates that MILP approach to trajectory optimization can solve realistic problems involving practical computation times and that linear constraints provide good approximations to the system of interest.

**Level IV application n.2**

Another level IV application is the one proposed in [34]. This work also is a thesis, in which the problem of trajectory generation with collision avoidance and coordination of UAVs cooperative team is considered.

In this work is employed a dynamic formulation (see section 3.2) of the trajectory generation problem and, to solve it, is used a MILP (see section 2.1) approach, which simplifies the dynamic model to a linearized form.

More in details, this work deals with detailed trajectory optimization using receding horizon control (RHC), and a low-level waypoint follower. MILP is applied
to both the task allocation and trajectory design problems to encode logical constraints and discrete decisions together with the continuous vehicle dynamics. The MILP-RHC uses a simple vehicle dynamics model in the near term and an approximate path model in the long term. This combination gives a good estimate of the cost-to-go and greatly reduces the computational effort required to design the complete trajectory, but discrepancies in the assumptions made in the two models can lead to infeasible solutions.

The primary contribution of [34] is to extend the previous stable RHC formulation to ensure that the on-line optimizations will always be feasible. Graph search algorithms are also integrated with the MILP-RHC, and the resulting controller is analytically shown to guarantee finite-time arrival at the goal. The control algorithms acting on four different levels of the hierarchy were also integrated and tested on two hardware testbeds (three small ground vehicles and a hardware-in-the-loop simulation of three aircraft autopilots) to verify real-time operation in the presence of real-world disturbances and uncertainties. Experimental results, reported in this thesis, have shown the successful control loop closures in various scenarios, including operation with restricted environment knowledge based on a realistic sensor and a coordinated mission by different types of UAVs.

The use of MILP together with RHC enables the application of the real-time trajectory optimization to scenarios with multiple ground and aerial vehicles that operate in dynamic environments with obstacles.

The “stability” of the trajectory designer proposed in [34] is defined as a guaranteed arrival at the goal. A straight line approximation gives a good cost estimate, but it can require too tight a turn because it does not account for the vehicle dynamics. Replanning is usually able to find a dynamically feasible path around the line segment path. However, in extreme cases, the large heading discontinuities, when the line segments join, leads to an infeasible problem in the detailed trajectory design phase. The trajectory optimization problem can become infeasible if there is a large difference between the straight line approximation and the kinodynamically feasible trajectory that has a minimum turning radius.

One way to avoid this situation is to prune, before constructing a cost map, all of the corners that require a large turn or have other obstacles around them. This ensures that any incoming direction at every corner can result in a feasible turn. However, this could lead to an overly conservative path selection (e.g., “go around all the obstacles”). Another approach is to place a turning circle at each corner
when constructing a cost map and enforce the rule that the vehicle moves towards the arc of the circle, not the corner.

The approach presented in [34] ensures the existence of a kinodynamically feasible turn at each corner, by applying a modified Dijkstra’s algorithm when constructing the cost map. When searching for the shortest path, this algorithm rejects a sequence of nodes if the turning circles cannot be suitably placed (i.e. a kinodynamically infeasible sequence). The generated tree of nodes gives the shortest distance from each node to the goal, along the straight lines in the regions where a feasible path is guaranteed to exist.

**Level IV application n.3**

A third level IV application found in literature is the one proposed in [31]. This paper too considers the problem of trajectory generation for multiple aircrafts avoiding collisions. The problem is described by means of a dynamic formulation (see section 3.2) and is solved by using mixed integer linear programming techniques (see section 2.1).

In practice, in this paper a method for finding optimal trajectories for multiple aircrafts avoiding collisions is described, in which the problem of trajectory optimization including collision avoidance is written as a linear program subject to mixed integer constraints. An approximate model of aircraft dynamics using only linear constraints is developed, enabling the MILP approach to be applied to aircraft collision avoidance. The formulation can also be extended to include multiple waypoint path-planning, in which each vehicle is required to visit a set of points in an order chosen within the optimization. A minimum time formulation is combined with the collision avoidance constraints and extended to include aircraft dynamics.

The ability of MILP to include discrete decisions in the optimization allows some very flexible mission problems to be solved. In particular, in [31] the extension of the formulation to include multiple waypoints path-planning is performed. In practice, instead of having a single terminal point, each aircraft is required to visit a number of points and the order of those visits is selected within the optimization to give the shortest overall flight time. [31] begins with the formulation of the trajectory optimization, including the dynamics model, collision avoidance constraints and multiple waypoints extension. An example is then presented to demonstrate that the approximation of the dynamics performs acceptably. Further examples demonstrate cooperative planning for multiple aircrafts and the flexibility of the multiple waypoints method.
Moreover, from a point of view of problem formulation and system modeling, it has been shown in [31] by theory and example that a constant speed, limited turn rate vehicle, such as an aircraft, can be modeled as a point mass with limited speed and subject to limited force, in a minimum time problem.

**Level IV application n.4**

Another level IV application found in literature is reported in [12]. This paper examines the problem of team cooperative UAVs path generation with collision avoidance, in presence of dynamic threats.

The one proposed in this work is an intermediate formulation between geometric and dynamic ones. This typology of formulation makes use of a substantially geometric approach but with the inclusion of some simple dynamic limits, in order to obtain solutions which are probably feasible.

The proposed application is referred to a battlefield scenario, where M unmanned air vehicles are assigned to strike N known targets, in the presence of dynamic threats. The problem is decomposed into subproblems, regarding in particular the cooperative target assignment, the coordinated UAV intercept, the path planning and the generation of a feasible trajectory. The main idea proposed in this paper is to use a nonlinear filter that has a mathematical structure similar to the kinematics of the vehicle to generate trajectories that smooth through the waypoints in real-time, while the vehicle traverses the trajectory, assuming constant altitude maneuvers and considering suitable heading rate constraints.

Paper [12] is analyzed more in details in appendix.

**Level IV application n.5**

Finally, another level IV application found in literature is the one proposed in [71]. This work considers the problem of UAVs team cooperative control, that is to generate trajectories for each UAV team member that enable the team to accomplish a specified mission, in presence of known threats, with a desired level of optimality.

In [71] the problem of team cooperative control of an UAV team is formulated as a constrained linear programming problem (see section 2.1) and is solved by means the use of classic LP numerical methods (see section 2.1) and of a neural network (see section 2.5), in order to compare these two approach to solution of the same problem.
This is due to the fact that, as well known, to solve the considered LP optimization problem by means of classic LP numerical methods, as the ones presented in section 2, may be critical, in particular with reference to on-line applications; hence, the authors propose to solve this optimization problem using a NN. From the numerical applications performed, the authors deduce that the use of NN in optimization problems, such as the one of the UAVs cooperation, is effective, particularly in on-line applications.
4.5 OTHER APPLICATIONS

From the literature survey also some papers, which are not directly referred to the problem of trajectory generation for aerial vehicles but, on the other hand, may be interesting also in this field, have been selected. In the next some of these paper are described.

**Underwater vehicles**

In paper [69], for instance, the problem examined is not referred directly to UAV applications but is referred to autonomous underwater vehicles (AUV). Nevertheless, the analysis of these applications may be useful in the field of unmanned aerial vehicles, because the AUV also are characterized by highly nonlinear and coupled system dynamics, so the autonomous control problems are in practice the same.

The paper is a survey on neural networks (see section 2.5) controllers for AUVs. The authors point out the advantage and drawbacks of the two typologies of network training: the off-line and the on-line training.

In the case of off-line training, the neural network controller is first trained (which can be compared to the tuning of a conventional controller) and then put to use. The primary advantage of offline learning of neural networks controllers is the speed of the resulting network. As no weight adjustments take place during the running of the controller, the response of the controller can be very rapid. The major drawback of this NN controller, yet, is that it is not adaptive. Hence, inaccuracies in the network weights or changes in system parameters are likely to result in poor performance of the controller system.

In the case of on-line learning, the weights of the NN are continuously updated as the neural controller works. Usually, a measure of the system performance is set up and one attempts to adjust the controller weights in a manner which improves this performance. The main challenge in this process is calculating the optimal weight changes from the system input and output as well as the reference trajectory for the system. Furthermore, ensuring the stability of such controllers is a challenging area where substantial work still remains to be done.

Another application in the field of underwater vehicles is presented in the work [56], where the problem of collision free path generation for an unmanned underwater vehicle, connecting a starting point with an ending point in presence of fixed obstacles, is considered.
This paper describes the optimal trajectory generation problem considered by means of a geometric path planning formulation (see section 3.1) and makes use of an algorithm to define and smooth out the boundary of obstacles. The algorithm is able to eliminate local minima to prevent the trajectory from getting stuck, a common problem encountered with the potential field method. A numerical optimization method (sequential quadratic programming) is then used to find the shortest path between the start and goal point.

Robots

Another interesting application found in literature is the one proposed in the paper [46], in which the problem of real-time path generation for a robot which must move in a plane surface in presence of dynamic obstacles is considered.

This paper proposes an application of fuzzy concepts (see section 2.4) to trajectory generation problems and, in particular, an electrostatic potential field (EPF) path planner is combined with a two-layered fuzzy logic inference engine and implemented for real-time mobile robot navigation in a 2-D dynamic environment.

In order to create an electrostatic potential field, the environment is first mapped into a resistor network, then a current injection into the network creates the electrostatic potential field. An example of electrostatic potential field between a positive electric charge (which represents, for instance, the fixed starting point of the trajectory to be determined) and a negative electric charge (which represents, for instance, the ending point of the trajectory to be determined) is shown in Figure 15.

![Figure 15 – An example of electrostatic potential field](image)
The path of maximum current through the network corresponds to the approximately optimum path in the environment.

The navigation control for the mobile robot is generated through a two-layered fuzzy logic inference engine. The first layer is primarily responsible for obstacle detection and performs sensor data fusion for sonar sensor readings. Each reading is represented as two different fuzzy variables, the one related to the direction, the other related to the distance from the obstacle. In particular, the fuzzy variable related to the direction can assume four different values that describe the sensor’s membership in four cardinal relative directions: the front collision, the left collision, the right collision and the back collision. Each of the collision values is represented as a fuzzy variable which can assume three values: not possible, possible and high. Any obstacle that is close to the mobile robot in any direction is represented, by means of this fuzzy set, by means of the first layer of fuzzy logic inference engine. The second layer, then, receives the output of the first one, representing the immediate collision possibilities, as well as the output of the artificial potential field and the speed of the mobile robot as input and generates the control, consisting in the values of two fuzzy output variables, the one related to the change of speed and the other related to the steering.

The EPF and the two-layered FL inference engine operate in tandem, in order to provide the selection of the approximated optimum path in the environment and the collision avoidance.

From the point of view of implementation, an environment occupancy map is first created and then mapped onto a resistor network used to derive and interpret the EPF. The EPF goal-driven navigation generates in real-time minimum occupancy trajectories. The FL inference engine provides a comprehensive sensor fusion approach that interprets the potential field results in light of the current local environment situation, thus allowing both reactive and reflexive navigation. The obstacle detection module outputs the position and the degree of possibility with which any collision may occur. This information is combined in the motion control module with the path planner output. If an obstacle blocks the planned path, a collision avoidance decision that affects the robot heading and speed is taken.

The solution to the navigation problem may be compared to the flow of electric current within a sheet of conducting material. A set mapping is performed from the real environment into a discrete electric circuit (resistance) and, in this resistor network, obstacles also are mapped. The path of minimum resistance within the circuit maps into a path of minimum occupancy within the environment.
The algorithm to create the natural potential field follows three steps:

- create an occupancy map of the environment;
- create the resistor network;
- solve the resistor network to obtain the potential field.

In this application the trajectory generation problem is in practice solved by means of the electrostatic potential field approach. Several advantages of this approach may be emphasized:

- in a network composed solely of resistors with positive resistance values, the system of equations that describe the network, based on Kirchhoff’s Laws, is linearly independent;
- the collision-free paths, generated from the electrostatic potential field, necessarily lead to the goal position;
- the electrostatic potential field approach generates an optimal, minimum occupancy path;
- the solution proposed in [46] accomplishes real-time path generation.

The fuzzy logic controller, on the other hand, interprets the potential path and performs sensor fusion in order to increase the ability of the vehicle to react to dynamic obstacles.

In general, the EPF path planner performs best in cluttered environments. If the potential field has several obstacles grouped in one location of the grid, the large open area of the network tends to pull the path further away from the obstacles and into the open area. Relatively thin obstacles do not provide the EPF with enough information to consistently generate collision-free paths. The FL controller incorporates omni-directional sensing, which allows the vehicle to detect and react quickly to any obstacle in the environment, regardless of the placement of the obstacle relative to the robot.

Other interesting works, which refer, respectively, to application of neural networks and genetic algorithms to the field of collision free trajectory generation and time optimal trajectory planning for mobile robots are [70] and [73].

**Descent/landing on aircraft carriers**

An application of real-time trajectory generation to the problem of descent/landing on aircraft carriers in automatic mode is proposed in paper [3].
The problem is solved by parameterizing a certain class of near optimal trajectories for a finite number of representative maneuvers. The parameterized set of solutions can then be used as zero-order approximations for real-time refinement of near optimal solutions corresponding to maneuvers not included in the initial parameter set.

The main thrust of [3] is real-time automatic trajectory generation (for increased efficiency and safety during landing), that takes explicitly into account environment and aircraft constraints. The work described bears important connections with the general problem of avionics system development to aid aircraft pilots in the execution of demanding tasks. Limitations can hardly be overemphasized, given the fact that approximately 40% of all emergencies and accidents today occur because of pilot errors. The key features of the autonomous carrier landing systems thus developed will include significantly increased reliability and safety when executing landing maneuvers as compared to auto-landing systems in use today.

A critical element in the development of such systems is the requirement that all algorithms developed be executed in real-time on the computers installed on-board the aircraft. Thus, the algorithms must be capable of solving in real-time boundary value trajectory optimization problems for a given set of tactical conditions and aircraft dynamics. Furthermore, they must provide autonomous tracking capabilities. It’s of great importance for carrier landing, because near-optimal trajectory during landing maneuver should be calculated several times, taking into account carrier’s position change.

This paper will be described also in appendix.

Trajectory tracking

A paper in which the problem of integrated design of guidance and control systems for autonomous vehicles is considered is [65]. The application here is the design and implementation of a nonlinear trajectory tracking controller for an UAV. The linear position of an autonomous vehicle is given in terms of its location with respect to the closest point on a desired trajectory, together with the arc length of an imaginary curve traced along that trajectory. Tracking of the nominal trajectory by the vehicle at a fixed speed is then converted into the problem of driving a generalized error vector, which implicitly includes the distance to the trajectory to zero. So the problem of trajectory tracking is posed and solved in the framework of gain scheduled control theory, leading to a new technique for integrated design of guidance and control systems for UAVs.
The key idea of the approach proposed in this paper is to reduce the problem to the design of a tracking controller for a linear time-invariant plant utilizing a simple nonlinear transformation that inverts the vehicle kinematics: this poses no robustness concerns. The application of the non linear transformation results in a nonlinear plant, whose linearization along trimming trajectories is time invariant. The methodology selected for linear control system design was $H_\infty$ synthesis.

Starting from a family of linear controllers with integral action designed for linearization of the nonlinear equations of motion described in an appropriate state space, the method produces a gain scheduled controller that preserves the input-output properties of the original linear closed-loop systems as well as the closed-loop eigenvalues. The key feature of the method is the ability to automatically reconfigure the control inputs of the vehicle to provide for proper control action as the body tracks an inertial trajectory in free space while maintaining constant airspeed, also in presence of wind disturbance.

The paper [65] will be analyzed more in details in appendix.

**Space plane’s reentry**

In paper [58] is proposed a method for trajectory optimization by using genetic algorithms, applied to space plane’s reentry problem. More in details, the space plane’s reentry trajectory generation problem is an highly nonlinear problem, consisting in choosing the angle of attack and the bank angle such that the final cross-range is maximized.

In this paper the proposed procedure makes use of a genetic algorithm (see section 2.3) in combination with calculus based methods, in order to find the best initial guess for iterative procedures such as the steepest descent method (see section 2.2).

Since typical GA solves the unconstrained optimization problems, the penalty function has usually been used as the fitness to handle the constraints. However, the convergence property of this conventional method is sensitive to the penalty parameter. In the case that the penalty parameter is substantially larger than its optimum value, which cannot be estimated a priori, the solutions tend to converge on a local minimum. On the other hand, in the case that the penalty parameter is smaller than its optimum value, the solutions probably cannot converge on the feasible region. Thus, some attempts have been made to remedy these drawbacks of the penalty function for GA. For example the penalty parameter is adaptively adjusted based on the population’s mean and the best value of the penalty...
function. On the other hand, in [58] a new selection method for GA is proposed, that robustly achieves the global optimization of the objective function and the feasibility search, even with a fixed large penalty parameter. In this method, the selection of the surviving individuals is carried out based on multiple criteria, i.e. the distance between the individuals, the objective function and the penalty function.

The proposed GA is applied to the optimization of a space plane’s reentry trajectory problem, which has a complexity of practical interest. It is observed that the solution of the proposed GA approaches to the vicinity of the optimal solution and it is reliable enough as an initial solution to the gradient-based trajectory optimization.

This paper will be described also in appendix.
5. Conclusions

There are many techniques for numerically solving on-line trajectory generation and optimization problems and several ways to formulate these problems. This document is intended as an excursus of the most important techniques and formulations that have received relevance in literature and as an attempt to classify trajectory generation problems based on their typology.

In recent years there has been an upsurge of purported new methods for trajectory generation and optimization each promising many advantages over the others. As pointed out by Betts in [1], frequently one has to deal with new, and sometimes confusing, terminologies but, in general, trajectory generation problems can be referred to the most comprehensive category of optimization problems.

This implies that the methods used in order to solve trajectory generation problems are substantially the same used in the field of optimization problems; an overview on these methods, both calculus-based and soft computing-based, is provided in section 2 of this document. As we have emphasized at various times, this overview must be regarded only as introductory, so one can refers to specialized literature and books if wants completely investigate these topics. The list of references reported in the next section can be regarded as an indication, although not exhaustive, regards to literature and books which can be useful in the field of trajectory generation.

In section 3 an attempt of formulation-based classification of trajectory generation problems is furnished. This is due to the fact that the solution method to be used is function of the problem formulation, which is the true critical topic in trajectory generation problems, in particular in the case of on-line applications.

From a most general point of view, however, the problem of trajectory generation is intrinsically linked to the problem of trajectory tracking. In other words, once an optimal or near optimal trajectory is determined with reference to the examined problem and this trajectory is physically feasible, it must have a control system on board of the vehicle which is able to track the specified trajectory.

Furthermore, in the real world, disturbances act on the vehicle and its behavior can be different from what is planned. In an uncertain environment, a change in the situational awareness might make the existing plan obsolete and the result of a stochastic event might require a change in the entire approach to the mission.
Replanning at the various levels of the control hierarchy can be used to compensate for these different types of uncertainties.

In conclusion, the problem of autonomous flying by manned and unmanned aircraft can be broken down into two tasks:

- the first is to compute in real-time optimal (or near optimal) trajectories from any initial condition to an appropriate final condition;
- the second is to track the computed (reference) trajectories in the presence of external disturbances and modeling uncertainties.

The complexity of a such problem defies an analytical solution. Therefore, research in this area has concentrated on developing suitable numerical techniques and modern methods, based for instance on the use of soft computing techniques. In developing these techniques with particular regard to the first topic of the problem, that is the real-time optimal trajectory generation, consideration must be devoted to two aspect: the trajectory must be feasible and the algorithm which determines it must converge surely and must involve a computational time short enough.

In this literature survey, main advantages and drawbacks of both numerical methods and problem formulations used in the field of on line trajectory generation have been emphasized. Furthermore, in section 4, we have provided a classification of the applications found in literature, based on the typology of the examined problem; in that section, several of the many papers examined in this literature survey have been described and analyzed and some of these, the most significant, have been also analyzed more in details in appendix. This analysis, of course, cannot be exhaustive, because this matter is the subject of ongoing researches, nevertheless it certainly represents a comprehensive outline of the state-of-the-art.

From the analysis performed and taking into account the observations earlier reported, in this section we summarize some conclusions regarding the approaches proposed in literature with reference to the problem of on-line trajectory generation and, briefly, regarding the problem of trajectory tracking also.

The problem of determining the optimal trajectory for an UAV is regarded in general as the problem of characterizing a trajectory able to drive the vehicle from a specified starting point to a specified ending point, optimizing a certain performance index in compliance with specified constraints related to aerodynamic and structural characteristics of the vehicle. In general, an algorithm devoted to this optimal trajectory generation will be iterative, so it may involve large
computational time, due for instance to the high nonlinearity of the UAV mathematical model.

Furthermore, the trajectory generation must be implemented on-line, since an off-line evaluation may be impractical because, for instance, the environmental conditions in general may change with respect to the ones considered in off-line evaluation or the UAV is required to fly through an unknown scenario.

As earlier emphasized, nevertheless, in on-line applications the certainty of the convergence and the shortness of the computational time are essential requirements, so in the choice of the problem formulation and of the solution technique this respect must be taken into account. In order to meet these requirements, in literature the most employed strategies are:

- the simplification of the UAV mathematical model;
- the parametrization of the solution, in such a way as quick algorithms can be used.

Of course, these strategies have some advantage as well as drawbacks. In particular, the use of a simplified model of the vehicle, i.e. for instance the use of a purely geometric problem formulation (see section 3.1), allows a significant reduction of the computation effort but, on the other hand, does not assure the determination of a solution which is really feasible, because for instance this solution may be incoherent with respect to aerodynamic and structural limits of the vehicle. A way out for this disadvantage may be, for instance, the introduction of some dynamic constraints in the simplified model, as in the case of the intermediate geometric-dynamic formulations, which we have described in section 3.1.

The parametrization of the solution has the disadvantage that it may lead to feasible solutions which does not be optimal but only sub-optimal; on the other hand, this approach allows computational times which are assured and short.

Some approaches proposed in literature with regard to on-line trajectory generation problems are based on the discretization of the optimization time horizon, in such a way as the original not finite dimensional problem is reduced to a finite dimensional problem, which can be handled by means of mathematic programming techniques such as the ones briefly reminded in sections 2.1 and 2.2. Nevertheless, these LP and NLP techniques are usually iterative, hence the computational time (and perhaps even the convergence, depending on the
technique employed) cannot be assured; this is the main limitation of mathematic programming in on-line applications.

In order to improve the computational effectiveness, several other approaches have been proposed in literature. For instance, dynamic formulations (see section 3.2) in which the model of the UAV is opportunely simplified have been presented. An example of these approaches is reported in paper [27], described in details in section 4, in which an optimization technique applicable to input-output linearizable systems is proposed and applied to a minimum time trajectory optimization problem for an helicopter.

Other approaches directly search for sub-optimal solution of the trajectory generation problem, by using the receding horizon approach, described in section 3.2. For instance, in [34] a method which combines RHC and MILP (see section 2.1) techniques, applied to a simplified model of the vehicle, is proposed. This methodology has the major drawbacks consisting in the use of a point of mass model of the vehicle and in the uncertainty regarding the computational time required.

Another possible approach, consisting in parametrizing the solutions by means of parametric curves, such as splines, is proposed for instance in [17]. In [45], furthermore, the application of genetic algorithms (see section 2.3) is proposed; this paper is described in details in section 4 and in appendix.

As earlier emphasized, the problem of trajectory generation is linked to the one of trajectory tracking, in the sense that the control system on board the UAV must be able to track the optimal or sub-optimal trajectory planned in the trajectory generation stage. With regard to the problem of trajectory tracking, the control system must generate thrust and control surface deflection commands in order to track the optimal trajectory. The major difficulties related to this problem are due to the high nonlinearity and coupling of the vehicle dynamics.

The trajectory tracking problem has been approached in different manners in literature. With reference to manned vehicles, the control system consists in two feedback loops: an inner loop which generates, based on set points regarding the attitude angles and their derivatives, the commands for the vehicle and an outer loop which is devoted to tracking the variables which specify the desired trajectory.

It is also possible to consider at the same time the optimal trajectory generation and tracking, by means of problem formulations such as the ones of optimal control (see section 3.3), but this approach is really complicated.
The problem of trajectory tracking, even if related to the one of optimal trajectory generation, is not the main concern of this literature survey; it is in any case exposed in specialized literature and books, as for instance [87].
6. REFERENCES


[28] Balkcom D. J, Mason M. T., "Extremal Trajectories for Bounded Velocity Mobile Robots"

[29] Hsu D., Kindel R., Latombe J. C., Rock S. “Randomized Kinodynamic Motion Planning with Moving Obstacles”


Literature Survey on Methodologies for On-line Trajectory Generation

Vittorio Di Vito, Bernardino Ricci – CIRA, Italian Aerospace Research Centre


[34] Kuwata Y., "Real-Time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon", MIT June 2003


[40] Li S. M., Boskovic J. D., Seereeram S., Prasanth R., Amin J., Mehre R. K., Beard R. W., McLain T. W., "Autonomous Hierarchical Control of Multiple Unmanned Combat Air Vehicles (UCAVs)", Proceedings of the American Control Conference, Anchorage, May 8-10, 2002


[69] Lorentz J., Yuh J., "A Survey and Experimental Study of Neural Network AUV Control"


APPENDIX: PAPER EVALUATION REPORTS

In this section some evaluation reports referred to the most significant papers examined during this literature survey are reported. These reports consist in three sections: first provide only the identification of the document, second furnish a critical analysis of this and third express an evaluation concerning the relevance of the document with respect to the problem of on line trajectory generation.

Of course, these reports does not be regarded as judgments but only as an useful way to summarize and arrange the wide literature which we have examined.

The reports are arranged as:

<table>
<thead>
<tr>
<th>Report n.</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Survey of Numerical Methods for Trajectory Optimization</td>
</tr>
<tr>
<td>2</td>
<td>A perspective on methods for trajectory optimization</td>
</tr>
<tr>
<td>3</td>
<td>Coordinated Target Assignment and Intercept for Unmanned Air Vehicles</td>
</tr>
<tr>
<td>4</td>
<td>Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities</td>
</tr>
<tr>
<td>5</td>
<td>Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation</td>
</tr>
<tr>
<td>6</td>
<td>Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints</td>
</tr>
<tr>
<td>7</td>
<td>On-Line Trajectory Optimization for Autonomous Air Vehicles</td>
</tr>
<tr>
<td>8</td>
<td>Near Optimal Trajectory Generation for Autonomous Aircraft Landing</td>
</tr>
<tr>
<td>9</td>
<td>Trajectory Optimization via Modified Genetic Algorithm</td>
</tr>
<tr>
<td>10</td>
<td>Direct Trajectory Optimization by a Chebyshev Pseudospectral Method</td>
</tr>
<tr>
<td>11</td>
<td>Trajectory Tracking for Autonomous Vehicles: An Integrated Approach to Guidance and Control</td>
</tr>
</tbody>
</table>
12 Probabilistic Path Planning for UAVs
A. Dogan [67]

13 Guidance of Unmanned Air Vehicles Based on Fuzzy Sets and Fixed Waypoints
M. Innocenti, L. Pollini and D. Turra [55]
Section 1: Identification of the Document

Title: Survey of Numerical Methods for Trajectory Optimization

Author(s): John T. Betts

Year: 1998


Section 2: Critical Analysis of the Document

Examined problem: The problem examined in the paper is the trajectory optimization of an aircraft or a space vehicle. In particular, a wide survey on the numerical methods applied in the field of the trajectory optimization is performed.

Problem Formulation: The examined problem is posed in the most general way. In particular, the dynamic system is modeled as an explicit system of first-order equations (state equations) and the trajectory optimization is intended as a constrained optimization problem, in which a generic performance index is considered as objective function.

Proposed solution method: The author does not propose a method for the trajectory optimization problem solution but he compares many methods, based both on classical mathematical formulations and on modern approaches. Regarding to the classical mathematical formulations, the author distinguishes direct methods and indirect methods, emphasizing that all the methods utilize a Newton-based iteration. Regarding to the modern approaches, the author briefly describe the dynamic programming and the genetic algorithms approach. In short, the author states that the genetic algorithms have the disadvantage that they are not computationally competitive with the classical methods, the dynamic programming approach involves a very big amount of data storage and the disadvantages of the indirect methods are substantially lower than the ones related to the direct methods.

Comments: In the paper the problem formulation is very general, so the author does not specify in details the control variables and constraints to be preferred in the practical implementation of a trajectory optimization algorithm. The importance of the paper is in the comparison of the numerical methods used in the solution of a generic trajectory optimization problem rather than in the problem formulation, which is virtually referred to a fairly general optimization problem. Only the trajectory generation problem is examined, without reference to the possible trajectory tracking problem.
### Section 3: Overall Interest of the Document for the On-Line Trajectory Generation and Degree of Confidence

|------------------------|---------|-----------------|--------|----------------|
SECTION 1: IDENTIFICATION OF THE DOCUMENT

<table>
<thead>
<tr>
<th>Title:</th>
<th>A perspective on methods for trajectory optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s):</td>
<td>I. Michael Ross and Fariba Fahroo</td>
</tr>
<tr>
<td>Year:</td>
<td>2002</td>
</tr>
<tr>
<td>Bibliographic reference:</td>
<td>AIAA</td>
</tr>
</tbody>
</table>

SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT

Examined problem: In the paper a classification of the methods used in the field of trajectory optimization is drawn.

Problem Formulation: The authors does not specify a problem formulation but in the paper a general optimization problem is considered, which may be in particular regarded as a generic trajectory optimization problem.

Proposed solution method: The authors does not propose a particular solution method, since their paper is intended for classify existing methods and clarify some aspect regarding these methods. More in detail, in the paper a distinction between transformations and discretizations of the original optimization problem is made. The authors point out that the transformations allow to "dualize" the problem but, strictly speaking, they not constitute a solution method being only an equivalent representation of the original optimization problem. The discretizations allow to convert infinite dimensional problems to finite dimensional problems and they are preliminary in order to numerically solve the problem.

In the paper the discretization methods are classified as "direct" if referred to the original optimization problem or "indirect" if referred to the transformed problem. The authors emphasize that usually direct methods are preferred over indirect methods in the case of complex problems, since they does not require transformations but, on the other hand, direct methods are less accurate than indirect ones. Finally the authors define the concept of algorithm and underline the difference between the convergence of the discretization and the one of the algorithm.

Comments: In this paper the problem of the trajectory optimization is not explicitly examined but a classification of the methods for trajectory optimization is pointed out. The main concern of the authors is to emphasize that a numerical trajectory optimization method is the totality of discretization method and the solution algorithm, so they examine in detail the transformations, discretizations and algorithms without reference to a specific trajectory optimization problem.
### Section 3: Overall Interest of the Document for the On-Line Trajectory Generation and Degree of Confidence

|-------------------|---------|-----------------|--------|----------------|
Section 1: Identification of the Document

Title: Coordinated Target Assignment and Intercept for Unmanned Air Vehicles
Author(s): Beard, McLain, Goodrich
Year: 2002

Section 2: Critical Analysis of the Document

Application of the Examined Problem: The application here is a very high level: there is a battlefield scenario where M unmanned air vehicles are assigned to strike N known targets, in the presence of dynamic threats. Some simulation results are presented.

Problem Formulation: The problem is decomposed into sub-problems:
- cooperative target assignment;
- coordinated UAV intercept;
- path planning;
- feasible trajectory generation.

In Figure 16 the system architecture proposed in [12] for a single UAV is shown.

The idea is to use a nonlinear filter that has a mathematical structure similar to the kinematics of the vehicle to generate...
trajectories that smooth through the waypoints in real-time, while the vehicle traverses the trajectory. There is also the simplified assumption of constant altitude maneuvers.

The heading rate constraint is indeed considered as: 
\[-c \leq u \leq c\]
where \(u\) indicates the value of the heading rate.

**Proposed solution method:**

Dashed line is not constrained to go through the waypoint, solid line is constrained to pass through the waypoint. It can be shown, using optimal control techniques, that the approach minimizes the time that the vehicle deviates from the Voronoi path.

![Voronoi path example](image)

*Figure 17: Example of Voronoi path [12]*

---

**SECTION 3: OVERALL INTEREST OF THE DOCUMENT FOR THE ON-LINE TRAJECTORY GENERATION AND DEGREE OF CONFIDENCE**

|-------------------|----------|-------------|--------|----------------|
**Section 1: Identification of the Document**

<table>
<thead>
<tr>
<th>Title</th>
<th>Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Faiz, Agrawal</td>
</tr>
<tr>
<td>Year</td>
<td>2001</td>
</tr>
<tr>
<td>Bibliographic reference</td>
<td>Journal of Guidance, Control and Dynamics</td>
</tr>
</tbody>
</table>

**Section 2: Critical Analysis of the Document**

**Application of the Examined Problem:**
This paper deals with trajectory planning of dynamic systems to satisfy explicitly the dynamic equations and inequalities on states and inputs. The author states that it is well known that most approaches to optimal control for nonlinear systems are computationally intensive and are usually not suited for real-time implementation. The method proposed is illustrated by two examples: an hardware implementation on a spring-mass-damper system to demonstrate real-time capabilities during pursuit and trajectory planning of a planar VTOL aircraft trajectory to illustrate application to nonlinear problems.

**Problem Formulation:**

![Figure 18 - Trajectory planning problem in the original and transformed space [63]](image)

The optimal control problem is here applied to differentially flat systems: these include controllable linear systems as well as nonlinear systems linearizable by static and dynamic feedback. Previous studies on differentially flat systems were restricted to cases where the motion wasn’t subject to inequality constraints. This paper extends the planning in presence of auxiliary constraints.

**Proposed solution method:**
By the use of the theory of semi-infinite optimization the volume of a polytope with a given number of facets is maximized. The author states that this procedure is called polytopic approximation and can be implemented using nonlinear programming.

**Comments:**

Literature Survey on Methodologies for On-line Trajectory Generation

*Vittorio Di Vito, Bernardino Ricci – CIRA, Italian Aerospace Research Centre*
### Section 3: Overall Interest of the Document for the On-Line Trajectory Generation and Degree of Confidence

|-------------------|---------|------------|--------|----------------|
SECTION 1: IDENTIFICATION OF THE DOCUMENT

Title: Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation

Author(s): Ioannis K. Nikolos, Kimon P. Valavanis, Nikos C. Tsourveloudis and Anargyros N. Kostaras

Year: 2003


SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT

Examined problem: In the paper the problem of path planning for UAVs is examined, with reference to both offline and online planning. In the offline planning a known environment is considered while in the online planning the environment is considered completely unknown. The aim of the planning is to determine an optimal or near optimal curved three-dimensional path line which connects a specified start point with a specified end point, allowing the UAV to avoid the obstacles in a rough terrain and the moving ones. This path line must be feasible with reference to the capabilities of the UAV and must involve the minimum possible length.

Problem Formulation: The formulation of both the examined problems, offline and online path planning, is substantially the same: the UAV is considered as a point of mass, subject to constraints which are an enforced maximum flight height and a minimum turn radius. Moreover, the UAV must to avoid the obstacles being the three-dimensional rough terrain and moving objects. The optimization problem considers an objective function which is slightly different in the case of offline and online planning.

In the first case the objective function considers the feasibility of the curve with reference to the allowed 3D space, the length of the path curve, the safety distance from the obstacles and the minimum curvature radius. In the second case, which is the online planning, the procedure is a step by step planning and the objective function considers the feasibility of the curve with reference to the allowed 3D space, the value of a potential field between the starting and the ending point, the length of the path curve, the safety distance from the obstacles, the minimum curvature radius, a coefficient designed to prevent the UAV being trapped in a local optimum region and another potential field developed in a small area around the final target.

The output of the offline path planning problem is a single three-dimensional spline curve, while the output of the online path planning problem is a continuous three-dimensional line that consists of successive spline curves smoothly connected.
to each other. In this second case, each curve is a near optimal solution, so the final curve also is a near optimal solution, because in the online planning the environment is considered unknown and the UAV is assumed to be equipped with a set of on-board sensors through which it can sense step by step its surroundings and position.

Proposed solution method:

The proposed solution method is a genetic algorithm. In the offline path planning problem, in which a single curved optimal path connecting the specified starting and ending points is determined, each chromosome constituting the population considered in the evolutionary algorithm represents the whole trial path examined, the genes of this chromosome being the 3D coordinates of the control points which characterize the spline curve. The fitness function associated to each chromosome of the population evolving in the genetic procedure is obviously the objective function, which incorporates the kinetic and dynamic constraints.

The evolutionary procedure considered in the online problem is substantially the same of the offline procedure but with the difference that in the online planning the genetic algorithm determines each spline curve constituting the final path curve. The selection scheme considered makes use of evolutionary operators which are the classical one-point crossover, the classic uniform mutation, the non uniform mutation and the heuristic crossover, with an elitist component which assures the permanence of the best individual in the next generation. The selection scheme used is a combination of the truncation model and the traditional roulette wheel selection. Since the computational time involved from the online procedure must be lower than the one involved from the offline procedure, the population size, the chromosomes length and the maximum number of generations is lower in the online procedure than in the offline one.

Comments:

Essentially, in the offline path planning problem a pure trajectory planning problem is considered while in the online path planning a mixed problem of both trajectory planning and sensing avoiding is examined. Because the terms in the objective functions are weighted by appropriate coefficients and the path feasibility is the main concern, the term of the objective function linked to this aspect has a dominant role.

The authors state that the numerical applications have demonstrated the effectiveness and the low computational time of the proposed procedure, although the online procedure may perhaps produce paths which are not compatible with the UAV capabilities. The opinion of the authors is that the evolutionary algorithms are ideal for the path planning problem because of their robustness compared to other direct search methods, their ease of implementation in problems with a relatively high number of constraints and their high adaptability to the considered problem.
### Section 3: Overall Interest of the Document for the On-Line Trajectory Generation and Degree of Confidence

|-------------------|----------|-------------|--------|----------------|
Problem Formulation:
The inherent properties of fixed-wing UAVs impose the input constraints of positive minimum velocity due to the stall conditions of the aircraft, bounded maximum velocity due to thrust limitations and saturated heading rate due to roll angle and pitch rate constraints.

The architecture considered in this paper consists of five layers:

![Architecture diagram](image)

With the UAV equipped with standard autopilots, the UAV model is assumed to be first order in heading and velocity, and second order in altitude.

Proposed solution method:
A Control Lyapunov Function (CLF) approach is proposed.
Comments: The authors believe that the utility of RHC is limited by its computation requirements and stability concerns.

Section 3: Overall interest of the document for the on-line trajectory generation and degree of confidence

|-------------------|----------|-------------|--------|----------------|
**Section 1: Identification of the Document**

**Title:** On-Line Trajectory Optimization for Autonomous Air Vehicles  
**Author(s):** Shannon Twigg, Anthony Calise and Eric Johnson  
**Year:** 2003  
**Bibliographic reference:** AIAA Guidance, Navigation and Control Conference and Exhibit, 11-14 August 2003, Austin, Texas (USA)

**Section 2: Critical Analysis of the Document**

**Examined problem:** The problem examined in the paper is the on-line trajectory optimization for UAVs, with particular reference to the optimization of the terrain following trajectory taking into account the threat avoidance constraint.

**Problem Formulation:** The authors propose two formulations for the optimization problem described above. In both formulation, which are different regarding to the problem approximation level, the aircraft is considered as a point of mass with 3 degrees of freedom and the dynamic system consists in the kinematics equations of motion.

In the first (simplified) formulation, the equations of motion are written in a local level frame considering only the horizontal component of velocity and assuming that the vertical component is small. In the second formulation, in which the aircraft is considered to fly in the plane tangent to the local terrain, this assumption is eliminated and the total velocity is taken into account.

In both formulations, a constant energy approximation is used, instead of the more usual constant velocity approximation. The control variable is the local heading angle and the objective function is formulated accounting for the flying time, the terrain and threat, allowing a choice of the preferred optimization criterion by which a weighting parameter. The limitation of the turn rate of the vehicle is taken into account as the only constraint and is formulated in two different ways.

**Proposed solution method:** For the problem solution an indirect method based on the application of the Pontryagin Maximum Principle is proposed. This method leads to different systems of differential equations to be solved according to the different problem and constraint formulations. In the numerical application the authors does not take into account the constraint and emphasize that, even if in the numerical applications proposed in the paper the two problem formulations lead in practice to the same optimal trajectory, this does not imply that in general they are equivalent.
Comments: The authors believe that the employment of a constant energy instead of a constant velocity approximation in the formulation of the trajectory optimization problem is more realistic and leads to optimized trajectory which are more convenient in terms of the trajectory tracking problem. Anyway, in the document only the problem of the trajectory optimization is regarded, without explicit reference to the problem of the trajectory tracking.

Section 3: Overall interest of the document for the Online Trajectory Generation and degree of confidence

|-------------------|----------|-------------|--------|-----------------|
**SECTION 1: IDENTIFICATION OF THE DOCUMENT**

**Title:** Near Optimal Trajectory Generation for Autonomous Aircraft Landing  
**Author(s):** Yakimenko, Kaminer  
**Year:** 1999  

**SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT**

**Application of the Examined problem:** Here the application is the problem of autonomous landing on aircraft carrier. In order to succeed, two main features need to be satisfied:  
- real-time computing of optimal or near-optimal trajectories, starting from an arbitrary initial condition to the appropriate final condition;  
- tracking the optimal trajectory computed before, with disturbances and uncertainties introduced by the model.  
No experimental flights carried on, only on-line simulations.

**Problem Formulation:** The problem of optimal trajectory generation is reduced to the one of optimal control. It is provided a formal definition of the aircraft optimal control problem for a set of admissible trajectories, satisfying:  
- system of ordinary differential equations with control inputs and aircraft physical parameters;  
- initial and terminal conditions in terms of trajectories and control inputs;  
- restrictions on the state space, on control inputs and on their derivatives.  
So the optimal control problem is to find an optimal trajectory that minimizes either integral functional $J$ or a function of current states and control inputs.

**Proposed solution method:** The authors believe that such kind of problems cannot be solved analytically, therefore, the only way is to find out a numerical solution. They also states that the effectiveness of numerical solution relies substantially on the theoretical formulation of the optimal control problem. In this sense, they have a bad opinion on direct methods in terms of computational weight and scarce numeric convergence (classical calculus of variations, Pontryagin’s maximum principle, Bellman’s Dynamic Programming).  
As a matter of fact they propose a direct method (namely mathematical programming, similar to Ritz-Galerkin) to address trajectory optimization problem: in this way near optimal trajectories are represented by polynomials.

**Comments:** Seems to be a really promising approach.
### Section 3: Overall Interest of the Document for the On-Line Trajectory Generation and Degree of Confidence

|-------------------|----------|-------------|--------|----------------|

Literature Survey on Methodologies for On-line Trajectory Generation

*Vittorio Di Vito, Bernardino Ricci – CIRA, Italian Aerospace Research Centre*
PAPER EVALUATION REPORT N. 9

SECTION 1: IDENTIFICATION OF THE DOCUMENT

Title: Trajectory Optimization via Modified Genetic Algorithm
Author(s): Yokoyama, Suzuki
Year: 2003
Bibliographic reference: AIAA Guidance, Navigation and Control Conference and Exhibit

SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT

Application of the Examined problem: The method proposed (genetic algorithm) is applied to two simple numerical test problems: the brachistochrone problem and the optimization of spaceplane’s reentry trajectory. Thanks to these examples, the good convergence property of the proposed method was demonstrated as well as the effectiveness for a practical problem solution.

Problem Formulation: First of all, is better for us to identify, what’s the difference between direct and indirect methods in the author’s opinion:

- indirect methods (e.g. Pontryagin) - Here the two point or multi point boundary value problem of the state variables and the adjoint variables is solved by a gradient algorithm such as SCGRA (Sequential Conjugate Gradient-Restoration Algorithm), MQA (Modified Quasi linearization Algorithm), Indirect Multiple Shooting;
- direct methods - Here the trajectory optimization (optimal control) problem is converted into mathematical programming such as NLP and is solved by NLP techniques: Sequential Quadratic Programming (SQP), Direct Collocation (DC), Direct Multiple Shooting.

The authors believe that Non Linear Programming Techniques are sensitive to initial solutions in some cases. As a matter of fact, they state that, compared to indirect methods, direct methods have advantages in terms of the robustness of convergence and the flexible applicability to practical complex problems. In any case there is a problem on direct methods: optimization results of the direct methods are still sensitive to initial solution in some cases, i.e. a poor initial solution may converge on infeasible solution or local optimum in highly nonlinear or multimodal problem.

Proposed solution method: In this paper genetic algorithms have been applied as a preliminary search method to find initial solution to gradient based direct optimization methods. Of course, genetic algorithms are not computationally competitive against the gradient based methods, but it is useful to find appropriate initial solutions for the gradient based methods due to GA global search capability: in detail, this study proposes a new selection method for GA that robustly achieves the global optimization of the objective function and the feasibility
search even with a fixed large penalty parameter; here the selection of the surviving individuals was carried out based on multiple criteria i.e. the distance between the individuals, the objective function and the penalty function.

**Comments:** This article shows in a quite clear way in its introduction which role is played by genetic algorithms into optimal control problems (initial guess). It is still unknown, however, if genetic algorithms may be used not only for computing initial guesses, but also to solve some other parts of the formulated problem.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Degree of confidence:</strong></td>
<td>[3] High</td>
<td><strong>[2] Average</strong></td>
<td>[1] Low</td>
<td></td>
</tr>
</tbody>
</table>
SECTION 1: IDENTIFICATION OF THE DOCUMENT

Title: Direct Trajectory Optimization by a Chebyshev Pseudospectral Method
Author(s): Fahroo, Ross
Year: 2002
Bibliographic reference: Journal of Guidance, Control and Dynamics, Vol 25, No 1

SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT

Application of the Examined problem: The application of the examined problem is theoretical and is connected to optimal control problem. Three numerical examples are produced to solve brachistochrone problem and moon-landing problem.

Problem Formulation: The formulation is the classical one of optimal control problems: in order to solve it, the exposed method transforms the optimal control problem into a Non Linear Programming (NLP) problem.

Proposed solution method: The authors believe that direct methods shows better convergence properties with respect to the classical indirect methods (e.g. Pontryagin). The advantage of the direct methods stands in the fact that necessary conditions aren't derived (instead of Pontryagin) and this feature let direct methods to be applicable to many practical trajectory optimization problems.

The solution method proposed reduces the optimal control problem into a NLP problem. Usually, in order to solve direct collocation schemes, that discretize both controls and state variables, the unknowns are the values of controls and state variables at the nodes (boundary conditions). Cost function and state equations are expressed in terms of these values. Then is commonly used a sort of interpolation form that uses interpolating polynomials (linear or cubic splines) that should satisfy the state differential equations by some integration scheme, such as Hermite Simpson, Gauss Lobatto or Jacobi.

In truth, instead of interpolating polynomials, globally orthogonal polynomials can be used, such as Legendre, Chebyshev or Lagrange.

For instance, in Chebishev, both state variables and controls are expanded in Fourier series with generalized coefficients: in this way, state equations, performance indexes and boundary conditions are converted into algebraic or transcendent equations in terms of these unknown generalized coefficients.

Here is presented a solution method that seems to be an evolution of direct collocation methods and is called Chebyshev Pseudospectral Method. The advantage of this
method with respect to classical direct collocation is an high accuracy level offered by the pseudospectral approximation, this thanks to a precise choice of the nodes on where boundary conditions are applied. Moreover Chebyshev polynomials have the computational advantage of being able to be evaluated in a closed form, without the need of advanced numerical techniques of linear algebra.

Comments: This technique has an high potential for use in optimal guidance algorithms that require a corrective maneuver from the perturbed trajectory. Further tests and analysis are necessary to investigate the stability and accuracy of the method.

### Section 3: Overall interest of the document for the On-Line Trajectory Generation and degree of confidence

|-------------------|----------|-------------|---------|----------------|
Section 1: Identification of the Document

Title: Trajectory Tracking for Autonomous Vehicles: An Integrated Approach to Guidance and Control

Author(s): Kaminer, Pascoal, Hallberg, Silvestre

Year: 1998

Bibliographic reference: Journal of Guidance, Control and Dynamics, Vol 21, No 1, January-February

Section 2: Critical Analysis of the Document

Application of the Examined problem: The application here is the design and implementation of a nonlinear trajectory tracking controller for the UAV Bluebird (unmanned air vehicle laboratory of the US Naval Postgraduate School). Numerical simulations using a full set of nonlinear equations of motion of the vehicle show the effectiveness of the proposed techniques.

Problem Formulation: The linear position of an autonomous vehicle is given in terms of its location with respect to the closest point on a desired trajectory, together with the arc length of an imaginary curve traced along that trajectory. Tracking of the nominal trajectory by the vehicle at a fixed speed is then converted into the problem of driving a generalized error vector, which implicitly includes the distance to the trajectory to zero. So the problem of trajectory tracking is posed and solved in the framework of gain scheduled control theory, leading to a new technique for integrated design of guidance and control systems for UAVs.

Proposed solution method: The key idea is to reduce the problem to the design of a tracking controller for a linear time-invariant plant utilizing a simple nonlinear transformation that inverts the vehicle kinematics: this poses no robustness concerns. It is important to point out that the application of the nonlinear transformation results in a nonlinear plant, whose linearization along trimming trajectories is time invariant. The methodology selected for linear control system design was $H_{\infty}$ synthesis. Starting from a family of linear controllers with integral action designed for linearization of the nonlinear equations of motion described in an appropriate state space, the method produces a gain scheduled controller that preserves the input output properties of the original linear closed-loop systems as well as the closed-loop eigenvalues. The key feature of the method is the ability to automatically reconfigure the control inputs of the vehicle to provide for proper control action as the body tracks an inertial trajectory in free space while maintaining constant airspeed (presence of wind disturbance is contemplated).
**Comments:** The methodology herein has two main advantages over traditional ones:
- the resulting trajectory tracking system achieves zero steady state tracking error about any trimming trajectory;
- the design methodology explicitly addresses the problem of stability of the combined guidance and control systems.

<table>
<thead>
<tr>
<th>SECTION 3:</th>
<th>OVERALL INTEREST OF THE DOCUMENT FOR THE ON-LINE TRAJECTORY GENERATION AND DEGREE OF CONFIDENCE</th>
</tr>
</thead>
</table>
PAPER EVALUATION REPORT N. 12

SECTION 1: IDENTIFICATION OF THE DOCUMENT

<table>
<thead>
<tr>
<th>Title:</th>
<th>Probabilistic Path Planning for UAVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s):</td>
<td>Atilla Dogan</td>
</tr>
<tr>
<td>Year:</td>
<td>2003</td>
</tr>
<tr>
<td>Bibliographic reference:</td>
<td>AIAA</td>
</tr>
</tbody>
</table>

SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT

Examined problem: The problem examined in the paper is the mission planning for UAVs flying through an area in the presence of a multiple source of threat. This planning is processed by means of a probabilistic approach, in order to go to a target location following the shortest possible path with an acceptable probability of getting disabled or an acceptably long path with the lowest possible probability of getting disabled.

Problem Formulation: The problem is formulated in a probabilistic frame. The author does not emphasize the aircraft modeling since he considers simply a point of mass which follows a path. In the path planning the heading angle is used as control variable and the maximum heading angle allowed is the only constraint taken into account. The objective of the path planning considered in the paper is to go to a specified target location along the shortest possible path with an acceptably probability of getting disabled or along an acceptably long path with the lowest probability of getting disabled. This implies that two point of view are regarded: the length and the risk of the path. Furthermore, the risk of the path may be taken into account at different levels, considering that the UAV is disabled when detected or hit or shut down.

Proposed solution method: The proposed solution method is based on the evaluation of the conditional probability that the UAV is disabled when following a certain path, considering the upper limit of this conditional probability as performance index in the optimal path search. In order to use this method, the probabilistic map of the area of operations must be available, by means of the knowledge of the probability density functions which link the exposure to the sources of threat to the aircraft position. For the solution, the author proposes a multi-step method, based on the evaluation of the probability of getting disabled along different directions, choosing the step in the direction corresponding to the smaller probability and respecting the constraint on the maximum heading angle (Local Minimum Strategy with Heading Constraint). The author emphasizes that, if the maximum heading angle allowed is too small, the path generated by this strategy might get in a limit cycle close to the target position but not close enough to attain it.
In order to avoid this situation, the author proposes a modification in the strategy (Limit Cycle Detection and Handling).

Comments: The author believes that the proposed method involves a computational time very low. In the paper only the high level path planning problem is considered, without reference to the problem of trajectory tracking.

SECTION 3: OVERALL INTEREST OF THE DOCUMENT FOR THE ON-LINE TRAJECTORY GENERATION AND DEGREE OF CONFIDENCE

|-------------------|---------|------------|--------|----------------|


**PAPER EVALUATION REPORT N. 13**

**SECTION 1: IDENTIFICATION OF THE DOCUMENT**

<table>
<thead>
<tr>
<th>Title:</th>
<th>Guidance of Unmanned Air Vehicles Based on Fuzzy Sets and Fixed Waypoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s):</td>
<td>Mario Innocenti, Lorenzo Pollini and Demetrio Turra</td>
</tr>
<tr>
<td>Year:</td>
<td>2003</td>
</tr>
<tr>
<td>Bibliographic reference:</td>
<td>Journal of Guidance, Vol. 27 No. 4, Engineering Notes</td>
</tr>
</tbody>
</table>

**SECTION 2: CRITICAL ANALYSIS OF THE DOCUMENT**

**Examined problem:** In the paper a guidance scheme for UAVs path planning and trajectory computation by specifying the waypoint position in space, crossing heading and velocity is examined. The proposed procedure is based on the assumption that the aircraft is equipped by an autopilot system which takes the references from a fuzzy controller. In particular, the synthesis of this fuzzy controller in order to execute the path planning and trajectory computation task earlier described is the focus of the paper.

**Problem Formulation:** The problem formulation considers an aircraft provided by a flight control system with an inner autopilot loop for tracking commanded velocity, flight path and heading angle. This inner control system is assumed to be able to track desired velocity, flight path and heading angle with a first order dynamic behavior. The state variables of the autopilot model are the velocity, the flight path angle and the heading angle whereas the inputs of the autopilot control system are the desired values of these state variables. The fuzzy controller is an outer loop which generates these references for the autopilot control system in order to reach fixed waypoints with fixed crossing heading and velocity. The constraints of this guidance problem are the angular rate limitations of the aircraft.

**Proposed solution method:** The proposed solution method is based on a fuzzy guidance system, which provides the references to the inner autopilot loop. This fuzzy guidance system is based on Takagi-Sugeno fuzzy sets, in order to obtain an acceptable computational load. As well known, the fuzzy controller is based on a blend of if-then rules and, in the application proposed in this paper, it uses membership functions which are combination of Gaussian curves. The fuzzy rules are defined according to the desired approach direction and the angular rate constraints of the UAV.

Since the aircraft is considered equipped by three closed loop inner autopilots, one for the velocity another for the flight path angle and the last for the heading angle, the outer fuzzy
controller is constituted by three independent controllers, one for each reference: the first generates the desired flight path angle based on the altitude error, the second the velocity reference based on the velocity error and the third generates the desired heading angle based on the position errors in the reference horizontal plane and on the heading angle error.

The velocity fuzzy controller uses three fuzzy sets for both the velocity error input and the velocity reference correction output. The flight path angle controller uses four fuzzy sets for both the altitude error input and the desired path angle output. The heading angle controller uses five fuzzy sets for the error input referred to the X-axis of the reference horizontal plane, three sets for the Y-axis error input and seven fuzzy sets for the heading error angle input; in the application, seventy fuzzy rules are defined for the heading angle controller.

Comments: The authors state that the numerical applications performed on both a linear decoupled aircraft model and a fully nonlinear autopiloted aircraft model have shown the effectiveness of the proposed fuzzy approach. The problem examined refers to the trajectory generation in order to reach fixed waypoints with fixed specifications in terms of crossing heading and velocity, without considering in detail the problem of trajectory tracking except for the inclusion of dynamic constraints (angular rate limitations) in the trajectory path generation problem.

SECTION 3: OVERALL INTEREST OF THE DOCUMENT FOR THE ON-LINE TRAJECTORY GENERATION AND DEGREE OF CONFIDENCE

|-------------------|---------|-------------|--------|----------------|