Optimization of Active Rendezvous Trajectories by Genetic Algorithms

Marie Emmanuelle Ricour
Embry-Riddle Aeronautical University - Daytona Beach

Follow this and additional works at: https://commons.erau.edu/db-theses
Part of the Aerospace Engineering Commons

Scholarly Commons Citation
https://commons.erau.edu/db-theses/235
Optimization of active rendezvous trajectories
by Genetic Algorithms

by Marie-Emmanuelle Ricour

A thesis submitted to the Graduate Studies Office in partial fulfillment of the requirements for the Degree of Master of Science in Aerospace Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida
Fall 2006
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
Optimization of active rendezvous trajectories by Genetic Algorithms

by Marie-Emmanuelle Ricour

This thesis was prepared under the direction of the candidate's thesis committee chairman, Dr. Yechiel Crispin, Department of Aerospace Engineering at Embry-Riddle aeronautical university and has been approved by the members of the thesis committee.

THESIS COMMITTEE:

Dr. Yechiel Crispin
Chairman

Dr. Ossama Abdelkhalik

Dr. Eric Perrell

Dr. Habib Eslami
Department Chairman, Aerospace Engineering

Dr. Christina Frederick-Recascino
Associate Provost

Date: 12/14/06
Acknowledgements

I would like to express my gratitude to the individuals who have helped me accomplish this thesis.

My greatest thanks and respect go to Dr. Yechiel Crispin, Chair of my committee, for his guidance, his encouragement, knowledge of the topic and his patience.

I would like to express my gratitude to my thesis committee members, Dr. Abdelkhalik and Dr. Perrell.

I would also like to thank every faculty member of the Department of Aerospace Engineering for their kindness and support, particularly Dr. Habib Eslami, Mary Balistreri and Kathleen Dileo.

I would also like to thank Alain Jeneveau, Guy Creutin, Rene Schmitt, Dr. James Cunningham, Judith Assad and Gisela Bouzon who made possible my dual degree between EPF, Ecole d'ingenieurs and Embry-Riddle Aeronautical University.

Finally, I would like to thank both my family and my friends for their support and encouragement.
ABSTRACT

Author: Marie-Emmanuelle Ricour
Title: Optimization of active rendezvous trajectories by Genetic Algorithms
Institution: Embry-Riddle Aeronautical University
Degree: Master of Science in Aerospace Engineering
Year: 2006

Trajectory optimization is and will remain a hot topic in the engineering field. Because analytical or exact solutions are often difficult and sometimes impossible to compute, there is a need for alternative and efficient methods. UnderWater Vehicles (UWV) trajectories and rendezvous trajectories of continuous low-thrust spacecraft are examined. One of the methods to solve such problems is the Genetic Algorithm (GA) method. In this work, a GA has been developed using Matlab®. It treats possible solutions to the studied problems as individuals and eventually converges to an optimal or near optimal solution. Genetic Algorithms have been used previously to solve chaser-target type of rendezvous trajectories. Here, active rendezvous trajectories have been successfully solved using Genetic Algorithms.
# TABLE OF CONTENTS

Acknowledgements ................................................................................................................................. iii

Abstract ................................................................................................................................................ iv

List of Tables .......................................................................................................................................... vii

List of Figures ......................................................................................................................................... viii

List of Symbols ....................................................................................................................................... x

Chapter 1: Introduction ............................................................................................................................ 1
  1.1 Rendezvous trajectories problem ................................................................................................... 1
  1.2 Literature Review .......................................................................................................................... 3
    1.2.1 Rendezvous trajectories optimization - classical methods ..................................................... 3
    1.2.2 Spacecraft trajectories optimization - Genetic Algorithm (GA) ........................................... 4

Chapter 2: Methods of solution of optimal control problems .............................................................. 8
  2.1 Classical method ............................................................................................................................ 8
  2.2 Direct methods .............................................................................................................................. 8
  2.3 Biologically inspired methods ....................................................................................................... 10
    2.3.1 Particle Swarm Optimization (PSO) ....................................................................................... 10
    2.3.2 Description of the Genetic Algorithm ................................................................................. 11
  2.4 Advantages and disadvantages of GAs compared to the other techniques .................. 20

Chapter 3: Genetic Algorithm implementation .................................................................................. 22
  3.1 Pseudo code of the GA .................................................................................................................. 22
    3.1.1 Pseudo code for the GA's main program ............................................................................... 22
    3.1.2 Pseudo code of the generate function ................................................................................. 23
  3.2 Improvements ............................................................................................................................... 24
    3.2.1 Initialization ......................................................................................................................... 24
    3.2.2 Creation of the new generation ........................................................................................... 24

Chapter 4: Rendezvous between many vehicles .................................................................................. 25
  4.1 Description of the problem .......................................................................................................... 25
    4.1.1 Description of the type of vehicles ....................................................................................... 25
List of Tables

Table 1 : De Jong's example - unsorted population ........................................... 16
Table 2 : De Jong's example - sorted population ............................................. 16
Table 3: De Jong's example - child 1 ............................................................... 17
Table 4: De Jong's example - child 2 ............................................................... 18
Table 5 : Reference parameters ....................................................................... 27
Table 6 : set of parameters - Single vehicle - simple case ................................ 39
Table 7 : Optimum solution characteristics - Single vehicle - simple case ........... 42
Table 8 : Optimum solution characteristics - Single vehicle - simple case ........... 45
Table 9 : Optimum solution characteristics - Single vehicle - simple case ........... 49
Table 10 : set of parameters - 2-vehicle rendezvous ........................................... 50
Table 11 : Optimum solution characteristics- 2-vehicle rendezvous ...................... 53
Table 12 : set of parameters - 2-vehicle rendezvous - Dive ................................. 54
Table 13 : Optimum solution characteristics - 2-vehicle rendezvous - Dive ........... 57
Table 14 : set of parameters - shallow depth - close rendezvous point .................... 57
Table 15 : Optimum solution characteristics - 2-vehicle rendezvous .................... 60
Table 16 : Reference parameters ....................................................................... 63
Table 17 : Set of parameters - 2-spacecraft rendezvous ...................................... 72
Table 18 : Optimum solution characteristics - 2-spacecraft rendezvous ............... 75
Table 19 : Set of parameters - Chebyshev approach .......................................... 81
Table 20 : Optimum solution characteristics - Chebyshev approach ...................... 82
Table 21 : Set of parameters - Degenerate case 1 .............................................. 86
Table 22 : Optimum solution characteristics - Degenerate case 1 ......................... 87
Table 23 : Set of parameters - 3-vehicle rendezvous .......................................... 88
Table 24 : Optimum solution characteristics - 3-vehicle rendezvous ...................... 89
List of Figures

Figure 1: De Jong F3 function ................................................................. 13
Figure 2: Best and worst solutions of De Jong's F3 function .................. 19
Figure 3: Convergence for De Jong's F3 function - best and average fitness ...... 19
Figure 4: Forces acting on the vehicle ................................................... 26
Figure 5a: GA Convergence - Single vehicle ........................................... 39
Figure 5b: GA convergence Single vehicle ............................................. 40
Figure 6: Best trajectories convergence ................................................ 41
Figure 7: Best trajectory - Single vehicle ................................................ 41
Figure 8: Optimum \( \gamma(t) \) for a single vehicle trajectory ....................... 42
Figure 9a: GA convergence - Single vehicle dive .................................... 44
Figure 9b: GA convergence - Single vehicle dive .................................... 44
Figure 10: Best trajectories convergence - Single vehicle dive ................. 44
Figure 11: Best trajectory - Single vehicle dive ....................................... 45
Figure 12: Optimum \( \gamma(t) \) for a single vehicle dive trajectory .................. 46
Figure 13a: GA convergence - Single vehicle - Too close final location ....... 47
Figure 13b: GA convergence - Single vehicle - Too close final location ....... 47
Figure 14: Trajectories convergence - Single vehicle trajectory ............... 47
Figure 15: Optimum trajectory ............................................................. 48
Figure 16: Optimum \( \gamma \) history ........................................................... 48
Figure 17a: GA convergence - raw fitness .............................................. 51
Figure 17b: GA convergence - modified fitness ...................................... 51
Figure 18: Optimum trajectory for a simple 2-vehicle rendezvous ............. 52
Figure 19: Zoom of the ending points area of the optimum trajectories ......... 52
Figure 20: Optimum \( \gamma \) history for a simple 2-vehicle rendezvous .......... 52
Figure 21: Fitness convergence - Dive type trajectory ............................... 54
Figure 22: Optimum trajectory - Dive type trajectory ............................... 55
Figure 23: Zoom of the optimum trajectory - Dive type trajectory ............. 55
Figure 24: Optimum \( \gamma \) history - Dive type trajectory ............................ 56
Figure 25: Fitness convergence shallow depth - close rendezvous point ...... 58
Figure 26: Optimum trajectory - shallow depth - close rendezvous point .... 59
Figure 27: Zoom of optimum trajectory - shallow depth - close rendezvous point. 59
List of Symbols

Lower case

\( a_n \)  \( coefficient \) of a regular polynomial
\( d \)  \( reference \) diameter
\( f \)  \( fitness \) function
\( f_{stop} \)  \( fitness \) stopping criterion
\( g \)  \( gravity \) constant
\( i \)  \( individual \) counter
\( j \)  \( vehicle \) counter
\( k \)  \( bit \) counter
\( m \)  \( mass \)
\( m \)  \( mass \) flow rate
\( n \)  \( generation \) counter
\( n^i \)  \( number \) of bits used to encode one design variable
\( n_{pop} \)  \( number \) of individuals in a population
\( n_{var} \)  \( number \) of design variable
\( p_{mut} \)  \( probability \) of mutation
\( p_{Xover} \)  \( probability \) of cross over
\( r \)  \( radius \)
\( t \)  \( time \)
\( u \)  \( radial \) component of the velocity
\( u_e \)  \( velocity \) of the propellant at the exit of a spacecraft's nozzle
\( v \)  \( normal \) component of the velocity
\( x \)  \( x-coordinate \)
\( x^i \)  \( design \) variable
\( x_{min}^i \)  lower limit of the design variable \( x^i \)
\( x_{max}^i \)  upper limit of the design variable \( x^i \)
\( y \)  \( y-coordinate \)

Upper case

\( A_c \)  \( coefficient \) of a Chebyshev polynomial
\( C_D \)  \( drag \) coefficient
\( D \)  \( drag \)
\( I_{sp} \) specific impulse
\( L_c \) typical hydrodynamic length
\( L_{ch} \) number of bits in a chromosome
\( N \) number of time steps
\( N_{gen} \) total number of generations
\( N_v \) number of vehicles
\( Re \) Reynolds number
\( S \) reference area
\( T \) thrust
\( T_c \) \( c^{th} \) Chebyshev polynomial of the first kind
\( V \) velocity vector
\( X \) position vector of a vehicle

Greek
\( \alpha \) true anomaly angle
\( \beta \) weighing parameter
\( \gamma \) thrust direction of an UWV
\( \gamma_{\text{min}} \) lower limit for \( \gamma \)
\( \gamma_{\text{max}} \) upper limit for \( \gamma \)
\( \theta \) thrust direction of a spacecraft
\( \theta_{\text{min}} \) lower limit for \( \theta \)
\( \theta_{\text{max}} \) upper limit for \( \theta \)
\( \mu \) medium viscosity OR heliocentric gravitational constant
\( \rho \) medium density
\( \tau \) nondimensionalized thrust
\( \Omega \) design space
\( \Delta t \) time step
\( \Delta x^i \) interval between two consecutive discrete values of the design variable \( x^i \)

Subscript
\( 0 \) initial condition
\( f \) final condition
\( j \) vehicle's number
superscript

* reference parameter

\bar{x} nondimensionalized x

Abbreviations and Acronyms

pop 3-D array of the population
fit 1-D array of the fitness values of each individual
DCNLP Direct Collocation with Non Linear Programming
GA Genetic Algorithm
\mu GA Micro Genetic Algorithm.
IVP Initial Value Problem
NLP Non Linear Programming
ODE Ordinary Differential Equation
PSO Particle Swarm Optimization
SA Simulated Annealing
TPBVP Two-Point Boundary Value Problem
UAV Unmanned Aerial Vehicle
UWV UnderWater Vehicle
Chapter 1

Introduction

1.1 Rendezvous trajectories problem

Determining a good trajectory is a very common problem and although the solution might sometimes look trivial, the most obvious solution is not always the best one or has to be corrected. For example, imagine you need to compute the trajectory of a vehicle in a two-dimensional domain, like space for a spacecraft or water for a robot. We might want to compute a minimum time or a minimum fuel trajectory between two locations in the domain. The equations of motion are given. They correspond to the dynamical constraints of the system. The starting location or initial condition and the ending location, a terminal constraint, are also given. Some other constraints are known, like the maximum velocity of the vehicle and the borders of the domain. A careful analysis of the problem is then required, depending on the optimality criteria and the given constraints.

Classical methods used to solve optimal control problems are based on the calculus of variations. With the increase of computing power, some numerical methods have been developed in the past decades. These methods will be presented in more detail in the next section. For now, let us introduce the optimal control problem:

To solve an optimal control problem, one has to determine the time histories of the controls, as well as the state variables history, that will optimize (meaning, maximize or minimize, depending on the formulation of the problem) a performance index over a given time. The state variables are subject to dynamical constraints. To simulate a continuous system on a digital computer, the system can be discretized by dividing the total time into a finite number $N$ of time intervals, over which the controls are kept constant. Discretizing the problem simplifies it a lot compared to the continuous time approach: the ordinary differential equations can be reduced to difference equations and the integral performance index can be reduced to a finite sum over the discrete time counter (Bryson, 1999).
To plan the trajectory of a vehicle, one needs first to determine what criterion or criteria have to be optimized. We may want to minimize the fuel consumption of a thrusted body, the time of travel, the error on the final location, etc. In this study, the total time of travel will always be given, as well as the starting and ending locations of the vehicle.

The constraints have then to be identified. The dynamical constraints are the equations of motion of the vehicle. The system might also be subject to terminal constraints, such as the final location of the vehicle.

Finally, the appropriate optimization method has to be chosen. Obviously, a method that will find the exact solution to the problem would be desirable. However, some problems are impossible to solve analytically. In this case, taking into account the required time to solve the problem and the expected accuracy of the solution, we have to choose a method that will provide an approximate but as good as possible solution.

First, we are going to study single vehicle's trajectory optimizations. Starting from a given initial location, the vehicle has to go to a prescribed final location in a given time. Its trajectory is determined by its thrust direction, which will thus be the control variable.

Then, we will study multiple vehicle rendezvous trajectories, with all the vehicles being active or cooperative, as opposed to an active-passive rendezvous, also called chaser-target rendezvous. For example, for a 2-vehicle active-passive rendezvous, one of them will not thrust at all to modify its trajectory, but "wait" passively for the other vehicle to rendezvous. In the cooperative rendezvous case, starting from different initial locations, all the vehicles will have to thrust in order to get to the prescribed final location.

The rendezvous problem is of a greater level of difficulty than a single vehicle trajectory optimization. Indeed, two or more trajectories have to be determined and additional terminal constraints have to be considered to ensure the vehicles are in the vicinity of each other. Some other terminal constraints can also be added, such as velocities alignment, minimum and maximum magnitudes of the final velocities, etc.

The problem of rendezvous trajectories optimization has been treated in the literature, using various techniques. These techniques are summarized in section 1.2. In Chapter 2 we briefly review the different optimization methods that have been or can be used to solve rendezvous problems. Finally, in section 2.4, we will compare the Genetic
Algorithm with these methods and explain why the GA technique has been chosen to solve rendezvous problems.

Two types of vehicles will be considered: vehicles that are moving in a viscous fluid (Chapter 4), with a small time of travel; they can be seen, for example, as small robots moving in water or UnderWater Vehicles (UWV), or as Unmanned Aerial Vehicles (UAV) then continuous low-thrust spacecraft in Chapter 5.

1.2 Literature Review

Some of the following papers deal with cooperative rendezvous, but optimum solutions have always been obtained using classical methods. Other papers treat the optimization of trajectories for rendezvous problems using genetic algorithms, but none of them considered the two vehicles to be active.

1.2.1 Rendezvous trajectories optimization - classical methods

Coverstone-Carroll and Prussing (1992) obtained analytical solutions for a minimum fuel rendezvous between two active power-limited spacecraft with propellant constraints. They first studied the rendezvous problem assuming a Hill-Clohessy-Wiltshire linearized gravity field. Then, they obtained solution for the nonlinearized inverse-square gravity field. In this case, a direct method using Direct Collocation with NonLinear Programming (DCNLP) was used.

A similar problem is studied for two power-limited spacecraft in the linearized Hill-Clohessy-Wiltshire gravity field, but this time for spacecraft on neighboring circular orbits only (Coverstone-Carroll and Prussing, 1993).

Rendezvous between two spacecraft in the inverse-square gravitational field have also been studied (Coverstone-Carroll and Prussing, 1994), for spacecraft in coplanar orbits. In the case of equal initial power-to-mass ratios and circular initial orbits, cooperative rendezvous allows saving a significant amount of propellant compared to the classical chaser/target rendezvous.

Several optimal three dimensional orbital transfer problems are solved by Pourtakdoust and Jalali (1995) for a thrust-limited spacecraft also by a direct optimization scheme. The Jacobian and Hessian matrices are solved analytically using DCNLP.
Another study of optimal low-thrust rendezvous using an extremum variational problem with constraints formulation has been conducted by Marinescu (1976), but for a chaser-target problem.

A recently presented method using generating functions (Park, Scheeres and Guibout, 2005) determines the optimal feedback control and trajectory of a continuous thrust rendezvous problem. The Hamiltonian system for the state and adjoints with split boundary conditions is derived. Generating functions are then used to find the optimal solution, considering the two point boundary value problem (TPBVP) as a canonical transformation. The main advantage of this method is that it does not require the guess of the initial or terminal adjoints to solve the problem.

Jezewski (1992) presents an optimal rendezvous trajectories problem subject to arbitrary perturbations and constraints, using primer vector theory as the basis for the optimal formulation. The solutions to the constrained nonlinear parameter problem is found using NLP. However the rendezvous is impulsive and for a chaser-target situation.

To summarize, active and passive rendezvous trajectories have been studied extensively in the literature using classical methods. In this work, the inherent difficulties of using a classic approach are being avoided by using an evolutionary method.

1.2.2 Spacecraft trajectories optimization - Genetic Algorithm (GA)

A nonlinear discrete-time optimal control problem with terminal constraints is treated by Crispin (2006) using a combination of genetic search which finds the control sequence with a solution of the initial value problem for the state variables. This method proved to be very efficient because it completely avoids solving the two point boundary value problem, and compared favorably with analytical and gradient based solutions.

Rauwolf, G.A. and Coverstone-Carroll, (1996) provided interesting conclusions about the use of Genetic Algorithms to generate low-thrust orbit transfers. In particular, they present two trajectories, one with constant and the other with variable thrust. The near optimal solutions obtained proved to be accurate enough to be at least used for preliminary mission planning, or as initial guesses for direct optimization techniques.
Rendezvous trajectories of the chaser-target type in the presence of disturbing forces are studied by Carpenter and Jackson (2003). If the Clohessy-Wiltshire equations describe the motion of orbiting bodies accurately enough for preliminary mission planning, they can lead to significant error in actual use, due to the presence of disturbing forces. Here, a Genetic Algorithm is used to minimize the range error after an impulsive maneuver, using the Clohessy-Wiltshire equations to compute solutions that will be used to initialize the GA population. This study leads to conclusive and encouraging improvements of impulsive rendezvous trajectories.

A similar problem has been treated by Kim, Y.H. and Spencer (2002), where the minimal fuel solution of the optimal impulsive rendezvous of two spacecraft is sought. Once again, however, the problem is of the chaser-target type. The algorithm used in this study proved to be very efficient at solving orbit transfer trajectories and solved a two-impulse rendezvous problem relatively accurately.

Olsen, C. and Fowler W., (2004) also present some encouraging results for the use of Genetic Algorithm to generate near optimal rendezvous trajectories. In a reasonable computational time, the algorithm used provided solutions that closely matched the reference optimal trajectories. Like in the work of Rauwolf, G.A. and Coverstone-Carroll, (1996), the authors emphasized the interest of near optimal solutions generated by a Genetic Algorithm that can be used as an initial guess for a calculus of variations method.

For engineering optimization, the quality of a solution often depends on more than one parameter. For example, criteria used to define a good trajectory might be the fuel consumption of a vehicle, the energy path, to be minimized, the error made on the final state, also to be minimized, or all of the given criteria at once. Hence, there is a need for efficient multi-objective optimization methods. In many studies involving trajectories optimization using Genetic Algorithms, multi-objective optimization techniques have thus been used. Rather than searching for a solution whose single objective value is the global optimal value, the "best" solution is found by simultaneously optimizing several objectives. These types of optimization problems have traditionally been solved by assigning a weighting factor to each objective, then combining the weighted objectives into a single scalar objective. This eliminates the need for a complex multi-objective
algorithm, but introduces new parameters: the weighting factors themselves. Determination of the correct weighting factors can, in fact, become an optimization process in its own right.

To deal with multi objective problems, a Pareto algorithm can be used. The concept of Pareto optimization has been described by Hartmann, J.W. (1999) as:

"A Pareto optimal solution is not unique, but is a member of a set of such points which are considered equally good in terms of the vector objective. This space may be viewed as a space of compromise solutions in which each objective could be improved upon, but if it were, it would be improved at the expense of at least one other objective."

The intent of the Pareto optimization is to derive a set of elite solutions from a larger population of candidate solutions through simultaneous comparison of several criteria.

For example, in the work of Rauwolf, G.A. and Friedlander, A. (1999), a Pareto Genetic Algorithm has been used in conjunction with a calculus-of-variations optimizer (as a local improvement procedure). The algorithm formulated was applied to three different interplanetary trajectory optimizations: Earth-Mars flyby, Earth-Mars rendezvous, and Earth-Mercury rendezvous. The fitness function of the GA is proportional to the squared errors made on the final location and velocities. Families of optimal trajectories were obtained in all test cases, with family members related through continuous Pareto curves, but, as trajectory complexity increased, populations were distributed less evenly over apparent Pareto curves. However, the algorithm proved useful in producing novel trajectories. The new solutions discovered possessed both non-intuitive structures and very high performance.

This is another advantage of a Genetic Algorithm: the absence of any preconceptions with regard to solution structure allows the GA to produce inventive solutions. On the other hand, the results presented required a large computational time. To reduce it, one might have to add some heuristics to guide the Genetic Algorithm search, thus losing the lack of preconceptions aspect.

Finally, micro Genetic Algorithms (µGA) are investigated by Coverstone-Carroll, V. (1997) to determine near-optimal low-thrust trajectories. Basically, micro Genetic Algorithms are Genetic Algorithms with populations typically fewer than 20 individuals, whereas classical Genetic Algorithms have populations typically ranging from 50 to 200
individuals. GAs are particularly robust methods for unconstrained optimization problems. Here, two ways of taking constraints into account are studied. The first method enforces constraints through equality constraints appended to the objective function and the second method treats the constraints as inequality constraints. It appears that μGAs converge faster than classical GAs when using the inequality constraints approach and were inefficient when the boundary conditions are treated as equality constraints.

It has been observed that all those previous studies using a classical approach provide optimal solutions but of course are strongly dependent on each particular problem and performance indices. The equations of motion to be used are determined by the studied type of vehicle. Then, to find an analytical optimal solution in a reasonable time, simplifications and approximations have to be made that are again strongly dependent on each particular case. One of the main purposes of this thesis is to develop a method which would allow a very broad range of applications.
Chapter 2

Methods of solution of optimal control problems

2.1 Classical method

We will go over various optimization methods that could be used to solve this problem. Even though the defined problem is actually a constrained optimal control problem, which involves the dynamics of the studied system, most of the general optimization methods can be used to solve it, for example gradient based methods.

The most classical method to solve dynamic optimization problems is the calculus of variations method. This is an indirect method. It provides an analytical solution. However, because it is sometimes very hard to solve the Euler Lagrange equations, it cannot always be used and the process can take an unreasonably long time. Furthermore, if one aspect of the problem is slightly changed, the whole problem has to be solved again. When using a numerical algorithm, and not an analytical approach, parameters just have to be changed and a new simulation can be run. Finally, when used to solve an optimal control problem, the calculus of variations method can sometimes lead to a difficult two point boundary value problem.

2.2 Direct methods

Non Linear Programming methods (NLP) can be used to solve an optimal control problem. Basically, it reformulates the dynamic problem as a static optimization problem. They have to start at a reasonable guess for the optimum solution. Then, the objective function and its derivatives are computed at that point. Starting from the present values the whole solution is moved to another point of the design space while satisfying the constraints. The process is repeated iteratively. These methods represent thus an organized search in the design space. They are also called direct methods of optimization. However, NLP can lead to stability and convergence issues, as will be discussed in the following paragraphs.
To solve the static optimization problem, various methods have been developed on the basis of gradient methods. Gradient methods search the design space using a search direction which is opposite to the gradient of the objective function. Thus, it ensures that, once the convergence criterion is met, the solution corresponds at least to a local optimum. The disadvantage of a gradient based method is that the objective function and its derivative have to be continuous. The other problem of gradient based methods is that they do not guaranty that the solution found corresponds to a global optimum. This is strongly apparent for multi-modal functions. It makes the method highly sensitive to the initial location of the search, which has to be as close as possible to the global optimum solution.

Newton's method and its variants approximate the objective function by a second order polynomial function around the current search point. The optimum solution for the approximated objective function is then computed using a one-dimensional search technique. They should thus be more accurate and require less iterations than basic gradient based methods, since they not only use the gradient of the objective function, but also its second derivatives to ensure an accurate enough approximation. On the other hand, it adds a new restriction on the objective function. It has now to be of class $C^2$, meaning that the objective function, its derivative and second derivative have to be continuous. Furthermore, the inverse or an approximation of the inverse of the objective function Hessian matrix has to be computed at each iteration. This can slow down the overall convergence of the method in the sense that it can require a relatively long computational time, or might sometimes be impossible when the matrix to be inverted is singular or becomes close to singular. Finally, just as gradient based methods, they are very sensitive to the initial guess and can easily lead to a local and not global optimum solution. To summarize, gradient based method cannot handle discontinuous functions or functions with discontinuous derivatives and do not guaranty a global optimum solution.

Direct Collocation Non Linear Programming (DCNLP) is a numerical method that has been used to solve many aerospace optimization problems. This procedure transcribes the continuous equations of motion into a finite number of nonlinear equality constraints. These constraints must be satisfied at designated collocation points, if the discrete approximation is to accurately represent the actual states of the system. This method was originally developed by Dickmanns and Well (Dickmanns, E. D. and Well, H., 1975.) and used by Hargraves and Paris (Hargraves, C. R. and Paris, S. W., 1987) to solve several
atmospheric trajectory optimization problems. DCNLP was utilized to determine the optimal cooperative and active-passive rendezvous trajectories.

Other modern methods include probabilistic, non gradient based methods, which have been developed quite recently, based on physical or biological analogies. For example, Simulated Annealing (SA) is an analogy with a manufacturing process. To cool a heated metal, this technique is sometimes used to make sure the metal does not become too brittle. By cooling it gradually through a particular device, the atoms align themselves to form crystals. This configuration represents the minimum energy state of the material.

2.3 Biologically inspired methods

Some methods based on biological analogies are Particle Swarm Optimization (PSO) (see for example Venter, G. and Sobieszczanski-Sobieski, J. (2002) and Crispin, Y. (2005)), and Genetic Algorithm (GA).

2.3.1 Particle Swarm Optimization (PSO)

PSO mimics the social behavior of a swarm, like a swarm of bees. Whenever bees find food, they will "dance" or fly with a specific pattern to inform the rest of the swarm. Thus a bee searching for food will use its own memory as well as information constantly provided by the swarm. This bee will remember where it has found food, transmit this information to the swarm and receive information from other bees about other locations where food has been found. Thus, it will "smartly" search the space for food by exploiting all these inputs. A bee, isolated from its swarm, is not very efficient. However, the whole swarm constitutes some kind of an "intelligent" entity. The large number of individuals allows a very big portion of the food search space to be explored, even though the search is not exhaustive. By cooperating constantly and working together, bees can search, find and get food very efficiently. The PSO algorithm is initialized with "particles" randomly distributed throughout the design space. Each particle will then start searching the surrounding space. To do so, its velocity vector is calculated as a function the best locations it has found so far, as well as the best locations found by the rest of the swarm. When all the particles have gathered at one design space location, the algorithm has converged.
This algorithm is easy to program and does not require continuity of the objective function in the problem definition. Furthermore, the particles will explore a large portion of the design space, and have thus a better chance to converge to a global or near global optimum than gradient based methods. It is also very convenient to solve discrete, combinatorial or discontinuous types of problems.

2.3.2 Description of the Genetic Algorithm

Finally, one other biologically inspired method is the Genetic Algorithm. Its advantages and disadvantages are described here. We will start by explaining where the name Genetic Algorithm comes from. Then we will describe the algorithm.

The analogy with nature and Darwin's theory

The concept of GA is attributed to the mathematician John Holland in the 1960's, first published in 1975 in *Adaptation in Natural and Artificial Systems*. It has also been used by David Goldberg, mainly to solve pipeline control problems (Goldberg, 1989).

Basically, GAs are a paradigm of Darwin's theory of evolution. According to this theory, a species naturally adapts to its environment to perdure. The best, strongest or fittest individuals in the given environment will have a better chance to reproduce than the weakest individuals. Their genome will thus be remembered in the next generation since it will be completely or partially transmitted to their offspring. It ensures that good genetic materials will not be completely forgotten from one generation to the other. Since the best individuals are more likely to mate, the new generation will be mainly made of children of two fit parents. They are thus likely to be themselves very fit individuals in their generation. Eventually, after a reasonable number of generations, the average fitness of the population should improve, and the fittest individual in the last generation should be particularly well adapted to its environment.

Essentially, a GA will treat solutions to a problem as individuals, part of a population of other solutions. The parameters defining a solution are encoded, for example in a binary string, to constitute a "chromosome". This chromosome thus contains all the necessary information to describe the solution completely. The solution itself can be seen as the genome of the individual. A fitness value will be assigned to each individual according to the solution they define. If the solution is good, i.e. it does not violate the constraints and performs well regarding the optimality criteria, its fitness will be good.
Starting from an initial population generated randomly, basic reproduction operators will be applied to these individuals to create a new generation. The individuals with the highest fitness will have a higher probability of being selected for reproduction. This process is repeated until some convergence criterion is met. The algorithm as it has been described above mimics indeed the Darwinian concept of natural selection, hence the name Genetic Algorithm.

In the next paragraphs, the mechanisms of a GA, briefly described above, are detailed, starting with the fitness function. To show how to formulate a problem in order to use a GA as the optimization method, we will use an example studied by (De Jong, 1975). De Jong's work has been particularly important to the development of GAs because he has very carefully tested the algorithm from a function optimization point of view only, even though he was himself interested in using GAs in other domains.

**Formulation of a problem using GAs**

**Fitness function or how to evaluate the quality of a solution**

The fitness function basically corresponds to the objective function associated to the problem. It may also contain the constraints of the problem. Depending on the formulation of the problem, it will have to be maximized or minimized.

For an engineering problem, the objective function is not necessarily given and has to be defined. If the problem involves many design variables and if the optimality criteria do not depend explicitly on the design variables, defining the objective function may be very sensitive. It has to be chosen carefully because the behavior of the algorithm, its convergence, will strongly depend on it. The reproduction selection is indeed based on the fitness value of each individual. If this fitness does not represent the quality of a solution well, the algorithm will probably be inefficient. So it has to reflect the quality of a solution, without assigning very high value to only relatively good solutions, to make sure that other regions of the design space will keep being explored.

The functions that De Jong studied do not include any engineering considerations, but allow a very rigorous analysis of the behavior of a GA from the mathematical point of view.
Let us consider De Jong's example function $F_3$, in a 2-D version for graphical interpretation:

\[
\text{maximize } f(x) = \sum_{i=1}^{2} \text{integer}(x_i), \; x \in \Omega
\]

with \( \Omega = \{ x = (x_1, x_2)^T : -5.12 \leq x_1, x_2 \leq 5.12 \} \)

In this case, the objective function is explicitly given and has to be maximized. It is obvious that the larger \( x_1 \) and \( x_2 \) are, the larger \( f \) will become, as can be seen in Figure 1. Even though maximizing \( f \) is in this case trivial, it will help understanding how a GA works throughout this chapter and also shows GA's efficiency at solving optimization problems. Indeed, as it appears very clearly, we would not be able to solve this problem were we to use a classical gradient based method, since the objective function is discontinuous.

Figure 1: De Jong F3 function
Representation of a solution - coding

The natural parameter set of the optimization problem has to be coded as a finite-length string over some finite alphabet. The simplest representation is binary: a number \( n \) of bits will be allocated to each design variable \( x_i \). Then, for each parameter, boundaries have to be defined, in order to be able to encode and decode the actual parameter value and its binary representation. This is how the design space domain or the set of feasible points is defined when using a GA.

In a \( n \)-bit long binary string, there are \( 2^n \) different combinations of 0 and 1. Thus, for one design variable \( x_i \) encoded on \( n \) bits, for \( x_i \in (x_{i\text{min}}, x_{i\text{max}}) \), \( x_i \) will be represented by a set of \( 2^n \) discrete binary strings, equally spaced between an all-0 string and an all-1 string, which correspond respectively to \( x_{i\text{min}} \) and \( x_{i\text{max}} \). In the real design space, the interval between two consecutive discrete values is given by:

\[
\Delta x_i = \frac{x_{i\text{max}} - x_{i\text{min}}}{2^n - 1}
\]

The complete solution known as the chromosome, will be represented by a larger string, containing all the smaller binary strings (genes) encoding the design variables values.

For De Jong's function, we can use for example 4 bits to encode each component of \( x \). A point of the design space is represented by an 8-bit long binary string, since \( \Omega \) is of dimension 2.

Reproduction analogy

Before starting mimicking the natural selection processes described by Darwin, we need to create the initial population of solutions. It will be generated randomly, knowing a few parameters like the population size and the total length of a chromosome. Then, the fitness value of each solution can be determined by calling a function that will decode the binary values of the design variables of the chromosome into the actual variables value of the design space and compute the objective function value corresponding to the such defined solution. The reproduction process can then start.
Selection

The most common selection operator for a GA is called the roulette wheel selection. Basically, a "slice" of an imaginary roulette wheel will be attributed to each solution of the current generation. The slice size will be proportional to each solution's fitness. A larger part of the wheel will thus be attributed to a better fit solution than to a poor solution. The wheel is then artificially spun. A solution will correspond to the area where the wheel stopped relative to a fixed pointer. This solution is then selected for mating, thus placed in the "mating pool". Since fit solutions correspond to larger portions of the wheel, they will have a higher chance to be selected. The selection is repeated a prescribed number of times.

Then, pairs of parents solution are selected randomly in the mating pool. We need now to introduce another parameter of a GA: the probability of reproduction. The behavior of the GA will strongly depend on its value. It is a real number between 0 and 1 that will determine how often parents selected for reproduction will actually reproduce. For each pair of parents, a random number between 0 and 1 is generated. If the probability of reproduction is less than this number; the two parents will not reproduce. Two other parents will then be selected, until a pair is really chosen for reproduction.

In this study, we will use a modified version of the roulette wheel. It has been described because it helps to visualize the selection process. We select two individuals $i_1$ and $i_2$ in the population randomly, without taking their fitness into account. A randomly generated number $p$, between 0 and 1, will be used as the reproduction probability. Only then will we use the fitness of the two selected parents. The population is sorted according to the fitness value, in a descending order (meaning, the fittest individual will be ranked 1 in the population). According to the ranks of the parents in the population, we compute a probability of cross over, the reproduction operator explained in the next section. For this particular pair of individuals:

$$p_{X_{over}} = 1 - \frac{rank(i_1) + rank(i_2)}{2n_{pop}}$$

(2.2)

where $rank(i)$ is the rank of individual $i$ in the population and $n_{pop}$ is the size of the population. If $p_{X_{over}}$ is greater than the randomly generated number $p$, they will be selected for reproduction.
For example, imagine that we have a population of 6 solutions to the De Jong's function:

<table>
<thead>
<tr>
<th>Solution #</th>
<th>chromosome</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0101 0101</td>
<td>-1.71</td>
<td>-1.71</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>1100 0011</td>
<td>3.07</td>
<td>-3.07</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>0111 1010</td>
<td>0.34</td>
<td>1.71</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0110 1111</td>
<td>-1.02</td>
<td>5.12</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1101 1100</td>
<td>3.75</td>
<td>3.07</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0101 0101</td>
<td>-5.12</td>
<td>-2.39</td>
<td>9</td>
</tr>
</tbody>
</table>

Let us now sort this population according to the fitness values:

<table>
<thead>
<tr>
<th>Rank</th>
<th>chromosome</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1101 1100</td>
<td>3.75</td>
<td>3.07</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0110 1111</td>
<td>-1.02</td>
<td>5.12</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0111 1010</td>
<td>-0.34</td>
<td>1.71</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1100 0011</td>
<td>3.07</td>
<td>-3.07</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>0101 0101</td>
<td>-1.71</td>
<td>-1.71</td>
<td>-4</td>
</tr>
<tr>
<td>6</td>
<td>0101 0101</td>
<td>-5.12</td>
<td>-2.39</td>
<td>-9</td>
</tr>
</tbody>
</table>

Let us say that solutions 2 and 5 are selected randomly as parents for the next generation. Their probability of cross over is, according to Equation (2.2):

\[
P_{\text{cross}} = 1 - \frac{2 + 5}{(2)(6)}
\]

\[
P_{\text{cross}} = 0.4167
\]

Let us assume the random number $p$ is 0.6. The probability of cross over for the two selected parents is less than $p$, thus the two parents will be discarded from the mating pool. Two new parents have to be randomly selected, for example 1 and 3. A new value for $p$ has to be generated, let us say 0.4, and $P_{\text{cross}}$ is given by:

\[
P_{\text{cross}} = 1 - \frac{1 + 3}{(2)(6)}
\]

\[
P_{\text{cross}} = 0.6667
\]

Since $P_{\text{cross}} > p$, the cross over operator will be used on the two parents 1 and 3.
To keep the size of the population constant, we will need to repeat this process until as many parents as there are individuals in the population are selected for cross over.

**Cross Over**

Let us now focus on the cross over operator, which allows "genetic" material to be mixed to create new solutions. The new solutions, or children, will be part of the next generation.

The simplest cross over one can think of is the single point cross over, where the two parents' chromosomes will be split in two. A child will inherit the first part of its chromosome from one parent, and the second part from the other parent. Two new solutions can thus be created.

In our study, we will use a slightly more elaborate method: instead of a one-point cross over, we will use a random multiple point cross over, by generating a mask. Basically, the mask is a binary string of the same size as a solution chromosome, where zeros and ones are randomly distributed. Then, the genetic material of the two parents will be mixed bit by bit using information from the mask. Whenever the mask contains a 1 in a given bit, child 1 will receive the binary value of parent 1 for this bit, and child 2 will receive the bit of parent 2. Whenever the mask contains a 0, child 1 will inherit from parent 2 and child 2 from parent 1.

Let us go back to De Jong's function example to illustrate the cross over process: parents 1 and 3 have been selected for cross over.

Then, generating a random binary string of size 8, we get the following mask:

\[ mask = 10111010 \]

We can now create two new solutions from parents 1 and 3:

<table>
<thead>
<tr>
<th>Table 3: De Jong's example - child 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
</tr>
<tr>
<td>parent 1</td>
</tr>
<tr>
<td>parent 2</td>
</tr>
<tr>
<td>child 1</td>
</tr>
</tbody>
</table>
Table 4: De Jong's example - child 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>1011 1010</td>
</tr>
<tr>
<td>parent 1</td>
<td>1101 1100</td>
</tr>
<tr>
<td>parent 2</td>
<td>0111 1010</td>
</tr>
<tr>
<td>child 2</td>
<td>0111 1110</td>
</tr>
</tbody>
</table>

**Mutation**

Just like the cross over operation, mutation is a phenomenon that exits in nature. However, mutation happens very rarely. To keep the analogy with nature consistent, thus, the probability of mutation $p_{mut}$ is often set to a very small value. For a binary string, when mutation occurs, the value contained in a bit will be changed to the other binary value.

To determine which bits have to mutate, a random number $p(k)$ is generated for each bit $k$ of the children. If $p(k)$ is smaller than $p_{mut}$, the bit's value is changed.

For example, let us assume that $p_{mut}$ is set to 0.1. If $p(k)$ is greater than $p_{mut}$ for $k \in (0,7)$ for child 1, then its first seven bits will remain the same. Now, if $p(8) < p_{mut}$, then the value in the last bit will change from 0 to 1:

$child_1 = 1101 1001$

**Results**

Using Matlab® to code the GA, we tested it with De Jong's F3 function. The pseudo code for the most important functions of the algorithm is given in Section 3.

The F3 function is plotted in black and white. The red and green dots represent the worst and best solutions respectively of a generation.
Figure 2: Best (green) and worst (red) solutions of De Jong’s F3 function

Figure 3: Convergence for De Jong’s F3 function - best and average fitness
As can be seen very clearly on Figure 2, the GA made the solutions "climb" all the way to the optimum value. Figure 3 shows the rate of convergence of the GA. The opposite of the best and average fitness of each generation are plotted. After about 35 generations, the optimum solution has been found.

2.4 Advantages and disadvantages of GAs compared to the other techniques

For optimal control problems, GAs might be a powerful tool to avoid solving a complicated two point boundary values problem, by transforming it to many Initial Value Problems (IVP) and avoid the initialization problem of gradient based methods. They have been used to solve a large number of control, orbit transfers, trajectories and rendezvous problems.

A GA has the advantage of being particularly robust compared to conventional methods. It is based on a directed random search. Thus, it guaranties that a large region of the design space will be explored, without being an exhaustive search. This increases the probability to find a global optimum solution to the problem. It does not search for an optimum in the neighborhood of a given point, but rather searches from a population of points. It can handle continuous or discontinuous functions and does not require derivatives knowledge. It only uses the objective function, which determines the fitness of an individual solution. This makes the GA a very general method, since this information is available in any problem. It can solve inherently discrete, combinatorial problems, since it does not work with the design variables themselves, but a coding of the variable set. Each variable is represented by a string. The GA will manipulate the strings coding the variables, thus exploiting similarities among high performance strings.

However, GAs do not guaranty that the global optimum solution will be found. If the algorithm converges too fast, it decreases the probability of exploring some regions of the design space, since most of the solution will have similarities with the best solution of the current generation. Some methods can help preventing the algorithm to converge too quickly, such as fitness scaling, an increase of the probability of mutation, redefinition of the fitness function, etc. Basically, diversity of the population should be maintained for a reasonable number of generations.
From previous studies, we can be very confident that GAs can solve a trajectory optimization problem. The robustness of a GA and its capacity to explore unusual regions of the design space are two of the important aspects that motivated our decision of using a GA to solve the problems. A major advantage of the GA is that it does not require the computation of first or second derivatives of the objective function. This is very important since calculating those derivatives can take a long time if the problem is complicated or have many design variables. Furthermore, the objective function does not even have to be continuous. GAs can handle discontinuous functions as efficiently as continuous functions. Even though the solution found by the algorithm might be only near-optimal, the short computational time is a very good advantage. The solution could be used to initialize another optimization procedure that requires a good initial guess. Finally, the very concept of GA was of interest. This technique is not conventional, not fully analyzed yet, and represented a challenge for the author.
Chapter 3
Genetic Algorithm implementation

3.1 Pseudo code of the GA

We are going to give here a pseudo code for a GA, applied to trajectory optimization. The first pseudo code is the main program. The \texttt{initpop} function will simply return a randomly generated initial population. The \texttt{getfit} function will calculate the fitness value of each individual of the population. Since the fitness depends on the treated problem, the pseudo code for this function will not be given. It cannot be generalized. It is important to remember, however, that the population will be sorted according to the individuals' fitness in the \texttt{getfit} function. The \texttt{stats} function calculates the best, average and worst fitness values in the current population.

The \texttt{generate} function will be given after the main program's. Indeed, it is the core of the Genetic Algorithm. In this pseudo code, the fitness goes from a positive value to 0 when the quality of a solution is increased.

3.1.1 Pseudo code for the GA's main program

\begin{verbatim}
N_v; % define the number of vehicles
X_0; \texttt{V_0}; t_0 % define initial conditions
X_f; \texttt{V_f}; % define terminal constraints
t_f; % define total time of travel
n_var; % define the number of design variables
n^i; % define the lengths of the binary strings encoding the design variables
n_pop; % define population size
N_gen; % define total number of generations

pop = \texttt{initpop}(n_{var}, n^i, n_{pop}); % generate initial population randomly
fit = \texttt{getfit}(pop, n_{var}, n^i, n_{pop}) % determine the fitness of each individual
[maxfit, avgfit, minfit] = \texttt{stats}(fit); % get statistics on the initial population
\end{verbatim}
\( n = 1; \)  \hspace{1em} \% generation counter

while \( n < N_{\text{gen}} \) do

\[ \text{[pop, fit]} = \text{generate}(\text{pop, fit, } n_{\text{pop}}, n^*); \] \hspace{1em} \% generate the next population

\[ \text{[max fit, avg fit, min fit]} = \text{stats}(\text{fit}); \] \hspace{1em} \% get statistics of new generation

\( n = n + 1; \)

end while

output results

3.1.2 Pseudo code of the generate function

for \( i = 1 \) to \( n_{\text{pop}} \) \% create \( n_{\text{pop}} \) children for the next generation

\[ \text{[parents]} = \text{select}(n_{\text{pop}}); \] \hspace{1em} \% select 2 parents randomly

\( p_{\text{Xover}} = \text{prob}(\text{parents, } n_{\text{pop}}); \) \% calculate their probability of cross over

\( p = \text{random}(1) \) \% generate random number between 0 and 1

if \( p_{\text{Xover}} > p \) \% if the 2 parents are selected for cross over

\[ \text{[children]} = \text{cross}(\text{parents}); \] \hspace{1em} \% create 2 new children with cross over

\[ \text{[children]} = \text{mutate}(\text{children}); \] \hspace{1em} \% apply mutation operator

\[ \text{[fitc]} = \text{getfit}(\text{children, } n_{\text{var}}, n^*); \] \hspace{1em} \% get the fitness of the children

\% if child 1 is better than the worst individual in its parents' generation

if \( \text{fitc}(1) < \text{fit}(n_{\text{pop}}) \)

\% locate the rank of child 1 according to its fitness

\( \text{rank} = \text{locatechild}(\text{fitc}(1), \text{fit}) \)

\% insert child 1 in the population and delete the worst individual

\( \text{insert}(\text{children}(1), \text{rank}); \)

end if

\% if child 2 is better than the worst individual in its parents' generation

if \( \text{fitc}(2) < \text{fit}(n_{\text{pop}}) \)

\% locate the rank of child 2 according to its fitness

\( \text{rank} = \text{locatechild}(\text{fitc}(2), \text{fit}) \)

\% insert child 2 in the population and delete the worst individual

\( \text{insert}(\text{children}(2), \text{rank}); \)

end if

end for
3.2 Improvements

Genetic Algorithms can always be improved by taking into account additional information on the specific problems it is to solve. In the present study, improvements have been made to the basic GA to decrease the required computational time. Since most of them depend on the problem, they will be given and explained in the next sections.

Other improvements are not necessarily problem-dependent and are described in the next paragraphs.

3.2.1 Initialization

The initialization of a GA is supposed to be completely random. However, to make sure that every possible solution can be created from the initial population, two individuals in the population were not initialized randomly. One chromosome is initialized with zeros only, and the other one with ones only. From those two chromosomes and using the mask technique previously described for crossover, every combination of bits can be created. This partially random initialization has been tested on the De Jong F3 function for a population of 50 individuals. The GA proved to be more efficient in terms of computational time, since most of the population is still initialized randomly.

3.2.2 Creation of the new generation

As can be seen on the pseudo code of the generate function in the previous section, when two children are created, they are not automatically added to the population. They will only be added if they actually improve the average fitness of the population. To do so, the fitness of child 1 is compared to the worst individual's fitness. If the child's fitness is better, then the worst individual is discarded from the current population and child 1 will be added at the proper rank according to its fitness. The same process is repeated for child 2. Since individuals with the worst fitness value have an extremely low probability of being selected for reproduction, discarding them should not affect significantly the exploration of the design space. Again, this has been tested on De Jong F3 function and the convergence rate was significantly improved as expected.
Chapter 4
Rendezvous between many vehicles

4.1 Description of the problem

In this section, we are going to study trajectories optimization for vehicles moving in an incompressible viscous fluid in a 2-dimensional domain. We will first present the types of vehicles we are considering, then describe the medium and external forces acting on the vehicles. From this information, we will state the simplifying assumptions that have been made to solve the problem and justify them. Finally, the problem will be formulated mathematically. We will then present a solution to the problem using a GA.

4.1.1 Description of the type of vehicles

This section deals with a relatively general problem, which can be applied to different kind of vehicles, such as small robots or UAVs. Robots are growing in complexity and their use in industry is becoming more widespread. So far, an important use of robots has been in the automation of mass production industries, where the same tasks must be performed repeatedly in exactly the same way. Robots and UAVs are also being used in environments that are dangerous or unreachable for humans. They can perform various tasks such as mines disposal, space exploration, rescue missions and exploration and mapping of unknown environments. For example unknown environments include underwater and areas that have been polluted by dangerous toxins. Therefore, the study of the dynamics and controls of such vehicles is relevant.

4.1.2 Medium and external forces

We treat the case where the medium in which the vehicles are moving is incompressible and viscous.

A gravitational field will exert an attractive force on any object. One of the forces acting on a vehicle is thus its weight, positive when acting downward (see Figure 4). The vehicle has a propulsion system that delivers thrust of constant magnitude and variable direction. Finally, since the fluid in which the vehicle is moving is viscous, a drag force is also acting on the vehicle, opposite to its velocity.
The control variable of the problem is the thrust direction $\gamma$. The angle $\gamma$ is measured positive counterclockwise from the horizontal as shown on Figure 4 below:

![Figure 4: Forces acting on the vehicle](image)

We need to determine the history of $\gamma$ between $t_0$ and $t_f$. Since GAs deal with discrete variables, we will have to discretize the values of $\gamma$. This will be detailed later in this section.

### 4.1.3 Simplifying assumptions

For the type of vehicles and the type of missions considered, it is reasonable to assume that the mass of the vehicles will remain constant. For an electric system of propulsion, its mass actually will remain constant. For other types of propulsion, the mass may slightly vary, but not significantly in the given time $t_f$. We choose the time length such as the assumption of a constant mass remains valid. We will also assume the velocity and the thrust to be parallel. If the thrust is changed quickly, we assume that the velocity vector will adjust to that direction. Finally, we will assume that the thrust magnitude is constant at all times and that its direction determines the vehicle's trajectory.

### 4.1.4 Mathematical formulation of the problem

**Equations of motion**

The motion of the vehicle is governed by Newton's second law
\[
\frac{d}{dt}(m\vec{V}) = m\vec{g} + \vec{T} + \vec{D}
\]  
(4.1)

where \( \vec{D} \) is the drag force acting on the body.

Since we assumed \( m \) to be constant,

\[
\frac{d\vec{V}}{dt} = \vec{g} + \frac{\vec{T}}{m} + \frac{\vec{D}}{m}
\]  
(4.2)

Projecting this equation on the direction tangent to the vehicle's path, it follows that:

\[
\frac{dV}{dt} = g\sin \gamma + \frac{T}{m} - \frac{D}{m}
\]  
(4.3)

\( D \) can be expressed in the typical form:

\[
D = \frac{1}{2} \rho V^2 S C_D
\]  
(4.4)

where \( \rho \) is the fluid density, \( S \) a typical cross-section area of the vehicle and \( C_D \) its drag coefficient that depends on the Reynolds number \( R_e = \rho V d/\mu \).

Equation (4.4) becomes:

\[
\frac{dV}{dt} = g\sin \gamma + \frac{T}{m} - \frac{1}{2m}\rho V^2 S C_D
\]  
(4.5)

This equation is then nondimensionalized, using the following reference parameters:

| Table 5: Reference parameters |
|-----------------------------|-----------------------|
| Unit                         | Parameter | Value            |
| \( m \)                      | \( L_c \)   | \( \frac{2m}{\rho S C_D} \) |
| \( kg \)                     | \( m_0 \)   | Constant mass of vehicle |
| \( s \)                      | \( t^* \)   | \( \sqrt{\frac{2m}{\rho \rho S C_D}} = \sqrt{\frac{L_c}{g}} \) |
| \( m.s^{-1} \)               | \( v^* \)   | \( \sqrt{L_c g} \) |
| \( N \text{ or } m.s^{-2} \) | \( F^* \)   | \( m_0 g \) |

where \( L_c \) is a very typical hydrodynamic reference length.
Equation (4.5) becomes:
\[
\frac{d\bar{V}}{dt} \frac{\sqrt{L_c g}}{\sqrt{L_c / g}} = g\sin\gamma + \frac{T m_0 g}{m m_0} - \frac{\rho S C_D}{2m} \bar{V}^2 (\sqrt{L_c / g})^2
\]
\[
\frac{d\bar{V}}{dt} = \sin\gamma + \frac{T}{m} - \bar{V}^2
\]  
(4.6)

or, rearranging the equation in separated variables form,
\[
\frac{d\bar{t}}{\bar{t}} = \frac{d\bar{V}}{\sin\gamma + \frac{T}{m} - \bar{V}^2}
\]  
(4.7)

The other equations of motion are:
\[
\frac{dx}{dt} = V \cos\gamma 
\]  
(4.8)
\[
\frac{dy}{dt} = V \sin\gamma 
\]  
(4.9)

Again, using Table 5, we can nondimensionalize Equations (4.8) and (4.9):
\[
\frac{d\bar{x}}{dt} = \bar{V} \cos\gamma 
\]  
(4.10)
\[
\frac{d\bar{y}}{dt} = \bar{V} \sin\gamma 
\]  
(4.11)

Now, if we are given the value of \(\gamma\) at all time \(t\), Equation (4.7) can be integrated between \(t_0 = 0\) and \(t_f\), knowing the initial conditions and terminal constraints.

For all the examples described later in this section, we will use:
\[
\begin{align*}
V(0) &= 0 \\
x(0) &= x_0 \\
y(0) &= y_0
\end{align*}
\]  
(4.12)

For all cases studied, the initial velocity of the vehicle or vehicles will be 0.

The terminal constraints correspond to the prescribed final location:
\[
\begin{align*}
x(t_f) &= x_f \\
y(t_f) &= y_f
\end{align*}
\]  
(4.13)
Discretization

Now, notice that Equation (4.7) cannot be integrated directly, since the right hand side depends on $V$ and $\gamma$. Since $\gamma$ is a continuous function of time, the right hand side does not depend on $V$ only, but also on the time $t$. However, if we divide the trajectory into $N$ straight segments of equal time duration $\Delta t$, we can assume that $\gamma$ is kept constant during one time step $i$, allowing us to integrate Equation (4.7). Also notice that $\gamma$ would have been discretized anyway to be able to use a GA.

The time segment $\Delta t$ is given by:

$$\Delta t = \frac{t_f}{N}$$

Or, using nondimensional notation, with $\bar{t}_f = \frac{t_f}{\sqrt{L_a/g}}$, we have $\Delta \bar{t} = \frac{\bar{t}_f}{N}$. From now on, for simplicity, we will drop the bar notation indicating nondimensional variables.

Integration

We can now integrate Equation (4.7) during one time step where $\gamma$ is held constant. For Equation (4.7) to be truly in separate variable form, thus, it has to be integrated between $t_i$ and $t_{i+1}$. The left hand side is simply:

$$\int_{t_i}^{t} dt = t - t_i$$

and the right hand side becomes:

$$\int_{V_i}^{V} \frac{dV}{\sin \gamma + \frac{X}{m} - V^2} = \frac{1}{b_t} \left[ \tanh^{-1} \left( \frac{V}{b_t} \right) - \tanh^{-1} \left( \frac{V_i}{b_t} \right) \right]$$

with $b_t = \sqrt{\sin \gamma + \frac{X}{m}}$

We have thus:

$$\int_{t_i}^{t} dt = \frac{1}{b_t} \left[ \tanh^{-1} \left( \frac{V(t)}{b_t} \right) - \tanh^{-1} \left( \frac{V_i}{b_t} \right) \right]$$

Rearranging the equation, we get:

$$V(t) = b_t \tanh \left[ b_t (t - t_i) + \tanh^{-1} \left( \frac{V_i}{b_t} \right) \right] \quad (4.14)$$
We can then calculate $V_{i+1}$ iteratively for all $i \in [0, N - 1]$ using:

$$V_{i+1} = b_i \tanh \left[ b_i \Delta t + \tanh^{-1} \left( \frac{V_i}{b_i} \right) \right]$$  \hspace{1cm} (4.15)

Now, we can integrate Equation (4.10). Separating variables, we have:

$$dx = V \cos \gamma_i \, dt$$

Substituting Equation (4.15) into this equation, it follows that:

$$dx = \cos \gamma_i \tanh \left[ b_i (t - t_i) + \tanh^{-1} \left( \frac{V_i}{b_i} \right) \right] (b_i \, dt)$$  \hspace{1cm} (4.16)

We can now integrate both sides of Equation (4.16). The left hand side is simply:

$$\int_{x_i}^{x} dx = x - x_i$$

For the right hand side, let us set:

$$a_i = \tanh^{-1} \left( \frac{V_i}{b_i} \right)$$

$$u = b_i (t - t_i) + a_i$$

Notice that $a_i$ is constant between $t_i$ and $t_{i+1}$. Therefore:

$$du = b_i \, dt$$

Changing variables for the integration of the right hand side of Equation (4.16) yields:

$$\text{RHS} = \cos \gamma_i \int_{u_i}^{u} \tanh (u) \, du = \cos \gamma_i \int_{u_i}^{u} \frac{e^u - e^{-u}}{e^u + e^{-u}} \, du$$  \hspace{1cm} (4.17)

Let us set:

$$v = e^u + e^{-u}$$

$dv$ becomes:

$$dv = (e^u - e^{-u}) \, du$$

Changing variables in Equation (4.17):

$$\text{RHS} = \cos \gamma_i \int_{v_i}^{v} \frac{dv}{v} = \cos \gamma_i [\ln v]_{v_i}^{v}$$
Finally, the right hand side of Equation (4.16) is:

\[ \text{RHS} = (\cos \gamma_i) \left[ \ln \left( e^{b_i(t_i-t)+a_i} + e^{-(b_i(t_i-t)+a_i)} \right) \right]_{t_i}^t \]

\[ = (\cos \gamma_i) \{ \ln[2\cosh(b_i(t_i-t)+a_i)] \}^t_{t_i} \]

Overall, Equation (4.16) becomes:

for \( t \in [t_i, t_{i+1}] \), \( x - x_i = (\cos \gamma_i) \{ \ln(2\cosh(b_i(t_i-t)+a_i)) \}^t_{t_i} \)

We can now find an expression for \( x_{i+1} \):

\[ x_{i+1} = x_i + (\cos \gamma_i) \{ \ln(2\cosh(b_i(t_i-t)+a_i)) \}^t_{t_i} \]

\[ = x_i + (\cos \gamma_i) \cdot \ln \left| \frac{\cosh(b_i\Delta t + a_i)}{\cosh(a_i)} \right| \] \quad (4.18)

Similarly,

\[ y_{i+1} = y_i + (\sin \gamma_i) \cdot \ln \left| \frac{\cosh(b_i\Delta t + a_i)}{\cosh(a_i)} \right| \] \quad (4.19)

**Objective function**

Now that we know how to determine the final location of one vehicle for a given \( \gamma \) time history, we can evaluate the quality of the vehicle's trajectory with respect to the terminal constraints. Basically, we want to minimize the error made between the prescribed final location and the actual final location of the vehicle.

For problems with more than one vehicle, we want to minimize this error for each vehicle. We can thus formulate the following objective function, for a given number of vehicles \( N_v \) and a prescribed final location \( \bar{x}_f \):

\[ f = \frac{1}{N_v} \sum_{j=1}^{N_v} \left\| \bar{x}_j(t_f) - \bar{x}_f \right\|^2 \] \quad (4.20)

where \( \bar{x}_j \) is the position vector of the vehicle:

\[ \bar{x} = \{ x \ y \} \]
Formulation of the optimal control problem

We now have all the information required to formulate the problem:

Find the optimum $\gamma$ time history:

$$\gamma(i) \quad \text{for } i \in [0, N - 1]$$

to minimize

$$f = \frac{1}{N_0} \sum_{j=1}^{N_0} \left\| x_j(t_f) - x_f^j \right\|^2$$ (4.20)

subject to:

– the state equations:

$$V_{i+1} = b_i \tanh \left( b_i \Delta t + \text{argtanh} \left( \frac{V_i}{b_i} \right) \right)$$ (4.15)

$$x_{i+1} = x_i + (\cos \gamma_i) \cdot \ln \left| \frac{\cosh(b_i \Delta t + a_i)}{\cosh(a_i)} \right|$$ (4.18)

$$y_{i+1} = y_i + (\sin \gamma_i) \cdot \ln \left| \frac{\cosh(b_i \Delta t + a_i)}{\cosh(a_i)} \right|$$ (4.19)

– the initial conditions

$$\begin{cases}
V(0) = 0 \\
x(0) = x_0 \\
y(0) = y_0
\end{cases}$$ (4.12)

– and the terminal constraints

$$\begin{cases}
x(t_f) = x_f \\
y(t_f) = y_f
\end{cases}$$ (4.13)
4.2 Formulation of the problem for GA and Matlab®

We are going to describe the type of variables that have been used to program a GA with Matlab® that will solve the described problem. We will start by presenting how a solution to the problem is encoded to a chromosome, then what problem-dependant improvements of the algorithm have been made compared to the basic GA.

4.2.1 Chromosome or solution

The design variables of the problem are the $N$ discrete values that $\gamma$ takes during each time step $\Delta t$ for each vehicle. They describe the solution completely and allow us to determine the final locations of the vehicles, and thus the fitness of this solution. So a chromosome will have to contain the $N \times N_v$ values of $\gamma$.

Depending on the accuracy of the desired solution, we now have to choose the number of bits $n^i$ on which each $\gamma(i)$ will be encoded. Also, according to Equation (2.1), the accuracy depends on the given range of $\gamma$. Depending on the initial conditions and terminal constraints, this range may be adjusted. Finally, the size of each design variable $n^i$ as well as the number of time steps $N$ will strongly influence the computational time. Hence, $n^i$ and $N$ must be chosen carefully, in order to compute an accurate enough solution in a reasonable time.

Overall, the total length of the chromosome will be:

$$L_{ch} = n^i \times N \times N_v$$

Essentially, a chromosome will look like:

$$\gamma_{(i \rightarrow j)}$$

4.2.2 Improvements

For the type of studied problem, we have been able to increase the rate of convergence of the algorithm by improving the basic algorithm.

Monotonicity of $\gamma$

For the kind of trajectory we are trying to compute, $\gamma$ is actually a continuous monotonic function of time. Thus, instead of waiting for the algorithm to converge
towards a monotonic $\gamma$, we are going to sort the values of $\gamma$ of each individual before calculating its fitness.

**Smoothing $\gamma$**

We do not want $\gamma$ to vary abruptly neither. This could lead to a contradiction with one of the assumptions we made, stating that the thrust is always acting tangent to the vehicle's path. If $\gamma$ does not vary smoothly, the time of the transition phase required for the thrust to become tangent to the path might be non negligible compared to the time of a step. We will thus use functions implemented in Matlab® to compute the coefficients of a fourth-order polynomial that fits best the discrete values of $\gamma$. The values of the polynomial at the $N$ discrete time points are then used as the current values of $\gamma$.

### 4.3 Results

#### 4.3.1 A simplified case: single vehicle trajectory

We first studied the trajectory of a single vehicle to test the algorithm.

**Reference parameters**

We need to determine first some reasonable values for the nondimensionalized parameters. Even though those values might slightly change for the next test cases, we will keep the same order of magnitude.

We will use some typical reference parameters for an UWV. The reference parameters were listed in Table 5. A small UWV could have the following characteristics

\[
m^* = 10 \text{ kg} \\
S = 1 \text{ m}^2
\]

The typical cross sectional area of the vehicle $S$ has to be of the same order of magnitude as its maximum cross sectional area. If the cross section is circular, the diameter $d$ of the vehicle will be:

\[d \approx .32 \text{ m}\]
We will also set the typical velocity to:

\[ v^* \approx 1 \text{ m/s} \]

Now, for water, we have:

\[ \rho = 1,000 \text{ kg/m}^3 \]
\[ \mu \approx 10^{-3} \text{ N.s/m}^2 \]

The water viscosity is given at \( T \approx 20^\circ \text{C} \).

We can now determine the Reynolds number \( R_e \):

\[
R_e = \frac{\rho v^* d}{\mu} \\
R_e = \frac{(1,000)(1)(.32)}{10^{-3}} \\
R_e = 3.2 \times 10^5
\]

From previous experiments, for a smooth sphere and the above Reynolds number, the drag coefficient is approximately:

\[ C_D = 1 \]

We can now determine \( L_c \) (see Table 5):

\[
L_c = \frac{2m}{\rho SC_D} \\
L_c = \frac{2(10)}{(1000)(1)(1)} \\
L_c = .02 \text{ m}
\]

Then the reference time can be calculated, using \( g = 9.81 \text{ m/s}^2 \):

\[
t^* = \sqrt{\frac{L_c}{g}} \\
t^* = \sqrt{\frac{.02}{9.81}} \\
t^* \approx 0.045 \text{ s}
\]
The reference velocity becomes:

\[ v^* = \sqrt{L_c g} \]
\[ v^* = \sqrt{(.02)(9.81)} \]
\[ v^* \approx .443 \text{ m/s} \]

And finally, the reference force is given by:

\[ F^* = m^* g \]
\[ F^* = (10)(9.81) \]
\[ F^* \approx 98.1 \text{ N} \]

We can estimate reasonable values for the nondimensionalized parameters.

**Case 1: Standard trajectory**

**GA Parameters**

For this problem, we choose:

- initial conditions

\[ \begin{cases} V(0) = 0 \\ x(0) = 0 \\ y(0) = 0 \end{cases} \]

- terminal constraints:

\[ \begin{cases} x(t_f) = 4 \\ y(t_f) = 2 \end{cases} \]

In dimensionalized form, we have \( x_f = 200 \text{ m} \) and \( y_f = 100 \text{ m} \) with:

\[ t_f = 5 \]

This corresponds to an actual final time of approximately 111 s.

For this set of terminal constraints, an appropriate range for \( \gamma \) would be:

\[ \gamma \in \left[ 0, \frac{\pi}{2} \right] \]
Since the mass of the vehicle is constant, the ratio \( \frac{T}{m} \) is actually equal to \( T \). A typical thrust for a UWV is in the range 50 \( \sim \) 100 \( N \). By nondimensionalizing this range using the reference force \( F^* \), we get the following approximate range for \( T \):

\[
T \in [0.5, 1]
\]

For this example, we will use:

\[
T = 0.6
\]

Now, we also need to choose a reasonable number of time steps. We chose:

\[
N = 40
\]

Thus, for \( t_f = 5 \), \( \gamma \) will be held constant during 0.125 in nondimensional time unit, which small enough.

The given set of parameters corresponds to physical properties of the problem. We now need to choose the design parameters associated with the Genetic Algorithm.

First, we need to choose the lengths of the "genes" where each discrete value of \( \gamma \) will be encoded. In the present case, there is no need to vary the values of the \( n^i \): for each value of \( \gamma \), the same number of bits \( n^i \) will be used. Nothing would justify that a better accuracy is required for one particular discrete value of \( \gamma \). Thus, \( \forall i \in [0, N - 1] \), \( n^i = \text{const} \). In the present case, we chose:

\[
n^i = 8 \, \text{bits}
\]

The interval between two consecutive possible values of \( \gamma \) is given by Equation (2.1):

\[
\Delta \gamma = \frac{\gamma_{\text{max}} - \gamma_{\text{min}}}{2^n - 1}
\]

\[
\Delta \gamma = \frac{\frac{\pi}{2} - 0}{2^8 - 1}
\]

\[
\Delta \gamma \approx 0.0062 \, \text{rad} \, \text{or} \, 0.35 \, \text{deg}
\]

And we can compute the length of a chromosome, using Equation (4.20):

\[
L_{ch} = n^i \, N \, N_v
\]

\[
L_{ch} = (8) \, (40) \, (1)
\]

\[
L_{ch} = 320 \, \text{bits}
\]
We need then to determine a reasonable size for the population of solutions. It is typically in the range

\[ n_{\text{pop}} \in [50, 200] \]

For this simple problem, there is no need for a particularly large population. After trying different values for \( n_{\text{pop}} \), we chose:

\[ n_{\text{pop}} = 50 \]

Finally, we need to determine the number of generations \( N_{\text{gen}} \) we want to create. Once again, since the problem is relatively simple, the convergence should be quick. We initially set \( N_{\text{gen}} \) to:

\[ N_{\text{gen}} = 40 \]

It is large enough to ensure that the best solution of the last generation has indeed a good fitness. We will use a generation counter to stop the algorithm when it reaches 40 generations. However, this stopping criterion is not a convergence criterion. We defined then another stopping criterion, depending on the fitness of the best individual in the population. In this simulation we keep creating new generations while:

\[ f_{\text{best}} > f_{\text{stop}} \]

with \( f_{\text{stop}} = .001 \)

This ensures that the error made on the final location is very small and insignificant for both \( x \)- and \( y \)- coordinates. This stopping criterion has been used for the final simulation.

Also, the mutation probability is set to a low value for the reasons mentioned previously:

\[ p_{\text{mut}} = .05 \]

The complete set of parameters for this simulation is summarized in Table 6.
Table 6: set of parameters
- Single vehicle - simple case -

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_v$</td>
<td>1</td>
</tr>
<tr>
<td>$n_{pop}$</td>
<td>50</td>
</tr>
<tr>
<td>$n_l$</td>
<td>8</td>
</tr>
<tr>
<td>$N$</td>
<td>40</td>
</tr>
<tr>
<td>$p_{mut}$</td>
<td>.05</td>
</tr>
<tr>
<td>$f_{stop}$</td>
<td>.001</td>
</tr>
<tr>
<td>$\gamma_{min}$</td>
<td>0</td>
</tr>
<tr>
<td>$\gamma_{max}$</td>
<td>\frac{\pi}{2}</td>
</tr>
<tr>
<td>$t_0$</td>
<td>0</td>
</tr>
<tr>
<td>$t_f$</td>
<td>5</td>
</tr>
<tr>
<td>$x_0$</td>
<td>0</td>
</tr>
<tr>
<td>$y_0$</td>
<td>0</td>
</tr>
<tr>
<td>$V_0$</td>
<td>0</td>
</tr>
<tr>
<td>$x_f$</td>
<td>4</td>
</tr>
<tr>
<td>$y_f$</td>
<td>2</td>
</tr>
</tbody>
</table>

The results of this simulation are presented in the following paragraphs.

Convergence

The convergence of the algorithm can be seen by plotting the worst, average and best fitness values vs. the generation counter:

![Figure 5a: GA Convergence - Single vehicle](image)

Figure 5a: GA Convergence - Single vehicle
From Figure 5a, it seems that the GA converges very fast, in about 5 or 10 generations, but keep in mind that relatively good improvements from one solution to the other can become hard to notice on this graph, since their fitness values are both very close to 0. On Figure 5b, we plot:

\[ y = \frac{1}{1 + f_n} \]

where \( n \) is the generation counter, for the best, average and worst fitness values of each generation \( n \).

![Figure 5b: GA convergence - Single vehicle](image)

To meet the convergence criterion, 22 generations were required. For this simple case, the algorithm converges in only 13.6 seconds on Pentium 4 - 2 GHz computer operating on Windows XP Professional.

On Figure 6, the fittest trajectory of the initial population is plotted. Then, every time the algorithm finds a new fittest trajectory from one generation to the other, this trajectory is added to the graph. It appears very clearly that improvements are made between generations 5 and 22.
Figure 6: Best trajectories convergence

Best trajectory

The best trajectory found by the algorithm once it has converged is plotted on the next figure.

Figure 7: Best trajectory - Single vehicle
The characteristics of this optimum solution are given on the following table:

<table>
<thead>
<tr>
<th>Table 7: Optimum solution characteristics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Single vehicle - simple case -</td>
<td></td>
</tr>
<tr>
<td>( N_{\text{gen}} )</td>
<td>22</td>
</tr>
<tr>
<td>CPU time</td>
<td>13.6 s</td>
</tr>
<tr>
<td>prescribed ( x_f )</td>
<td>4</td>
</tr>
<tr>
<td>actual ( x_f )</td>
<td>3.9808</td>
</tr>
<tr>
<td>error on ( x_f )</td>
<td>0.480%</td>
</tr>
<tr>
<td>prescribed ( y_f )</td>
<td>2</td>
</tr>
<tr>
<td>actual ( y_f )</td>
<td>2.0162</td>
</tr>
<tr>
<td>error on ( y_f )</td>
<td>0.808%</td>
</tr>
<tr>
<td>fitness ( f )</td>
<td>( 6.03 \times 10^{-4} )</td>
</tr>
</tbody>
</table>

This solution corresponds to the following \( \gamma \) history:

![Figure 8: Optimum \( \gamma(t) \) for a single vehicle trajectory](image)

**Analysis of the results**

The final optimum trajectory found by the algorithm is very encouraging. Its characteristics correspond to what we expected. Since gravity is acting downward, a straight line between the initial and final locations would not have been the optimum
trajectory. The vehicle should indeed take advantage of its own weight to sink deeper. The optimum trajectory (Figure 7) is slightly curved as expected. This case is not as trivial as it seemed. Indeed, in the given time of travel, the distance between the initial and final prescribed location is very close to the maximum distance the vehicle could have covered in the direction defined by the initial and final locations.

"Difficult" cases

We can now test the GA for more complicated or degenerate cases.

We will test two relatively difficult cases. The first one is an almost perfect dive and for the second case, we will intentionally choose the final location of the vehicle too close to its starting point. The reasons why we considered those two cases to be challenging to solve with a Genetic Algorithm will also be given.

Vertical dive

We will use the parameters as in the previous case, except for the terminal constraints and the stopping criterion: since the case is more difficult, we relaxed it a little bit. We chose:

\[
\begin{align*}
X_f &= .5 \\
y_f &= 5.55 \\
f_{stop} &= .0015
\end{align*}
\]

We did not try to compute the trajectory for a perfect dive, because its solution is included in the population when it is initialized, as explained in the "Improvements" section 4.2.2. Indeed, the population is not initialized completely randomly. To make sure that every combination of bit values can be created from the initial population, two chromosomes are initialized with only zeros and only ones. In this case, for a perfect dive, the thrust direction should be kept constant and equal to \( \frac{\pi}{2} \). Since \( \gamma_{max} = \frac{\pi}{2} \), the global optimum solution to the perfect dive problem would already be in the initial population.

For this case, it took 140 iterations or 72.7 seconds to reach convergence. This was not unexpected, since we kept the same range for \( \gamma \):

\[
\gamma \in \left[ 0, \frac{\pi}{2} \right]
\]
and we want the algorithm to find a solution where $\gamma$ is very close to $\frac{\pi}{2}$ for all time steps. It is a difficult case because the algorithm has to find a solution where $\gamma$ always has to be in a relatively small range compared to the actual given range. The oversized given range increases the size of the design space a lot. It is not surprising, thus, that more generations than for the first case were required to reach convergence.

We obtained the following results:

![Figure 9a: GA convergence - Single vehicle dive](image1)

![Figure 9b: GA convergence - Single vehicle dive](image2)

![Figure 10: Best trajectories convergence - Single vehicle dive](image3)
As can be seen on the previous figures, the algorithm converges very quickly to a reasonable solution, then converges more slowly to a better and better solution. Overall, it took 140 generations to create a solution whose fitness is less that $f_{stop} = 0.002$.

The characteristics of the optimum solution are given in the following table 8:

| Table 8 : Optimum solution characteristics  
- Single vehicle - simple case - |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{gen}$</td>
</tr>
<tr>
<td>CPU time</td>
</tr>
<tr>
<td>prescribed $x_f$</td>
</tr>
<tr>
<td>actual $x_f$</td>
</tr>
<tr>
<td>error on $x_f$</td>
</tr>
<tr>
<td>prescribed $y_f$</td>
</tr>
<tr>
<td>actual $y_f$</td>
</tr>
<tr>
<td>error on $y_f$</td>
</tr>
<tr>
<td>fitness $f$</td>
</tr>
</tbody>
</table>

Both errors are reasonable. The solution is near optimal but very encouraging. The trajectory corresponds to the time history of $\gamma$ plotted on Figure 12.
shallow depth - close prescribed final location

In this problem, we intentionally chose the prescribed final location close to the initial location. Since $t_f$ is fixed, the vehicle will have to "loiter" around the final location and only reach it when $t = t_f$.

For this case, thus, we increased the range of $\gamma$ to give more freedom to the vehicle:

$$\gamma \in \left[ -\frac{\pi}{2}, \frac{\pi}{2} \right]$$

The vehicle still starts at the origin of the coordinate system, and has to end at:

$$\begin{cases} x(t_f) = 2 \\ y(t_f) = 2 \end{cases}$$

Also, we used the same stopping criterion as in the first case:

$$f_{stop} = .001$$

The algorithm converged very quickly in 17 generations, in 10.4 seconds.
Figure 13a: GA convergence
Single vehicle - Too close final location

Figure 13b: GA convergence
Single vehicle - Too close final location

Figure 14 shows the fittest trajectory at the current generation as a function of the generation counter. This trajectory shows that the vehicle has to maneuver more because it has to dive deeper than the prescribed final depth first and then go up to be in the vicinity of the final location in the given total time.

Figure 14: Trajectories convergence - Single vehicle trajectory
The optimum trajectory and the corresponding $\gamma$ history are plotted in the next figures:

Figure 15: Optimum trajectory

Figure 16: Optimum $\gamma$ history
The algorithm in this case converged very quickly. This illustrates an important aspect of the GA. Indeed, by running the same calculation many times, the computed solution might be slightly different or found in a different number of generations. This is a direct consequence of the role of randomness in a GA. The GA is initialized partially randomly. The selection is also partially random, since a pair of solutions is selected for reproduction is their cross over probability is larger than a random number. Then the operation of cross over is itself random, since the mask is generated randomly. Therefore, each simulation will converge differently. It can also happen that a very good solution is randomly created in the initial population, in which case the algorithm will converge much more quickly.

In any case, the solution for the problem is again very encouraging regarding the efficiency of the algorithm. The characteristics of the solution are given in Table 9.

<table>
<thead>
<tr>
<th>Table 9 : Optimum solution characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Single vehicle - simple case -</td>
</tr>
<tr>
<td>( N_{\text{gen}} )</td>
</tr>
<tr>
<td>CPU time</td>
</tr>
<tr>
<td>prescribed ( x_f )</td>
</tr>
<tr>
<td>actual ( x_f )</td>
</tr>
<tr>
<td>error on ( x_f )</td>
</tr>
<tr>
<td>prescribed ( y_f )</td>
</tr>
<tr>
<td>actual ( y_f )</td>
</tr>
<tr>
<td>error on ( y_f )</td>
</tr>
<tr>
<td>fitness ( f )</td>
</tr>
</tbody>
</table>

### 4.3.2 Multiple vehicles Rendezvous

Now that the algorithm has been tested for a simple case, we can start studying rendezvous trajectory optimization problems. The same code has been used to solve rendezvous problems, by setting the number of vehicles \( N_v \) to 2 or more. We will start by solving a simple case. It will be our reference case. Then we will try to solve a dive-type problem. Finally, we will study a case where the prescribed final location is too close to the original points, following the same scheme as in the previous section.
Reference case

Since we are now studying rendezvous problem, the variable $N_v$ is this time relevant. We are going to solve a 2-vehicle rendezvous problem, starting from different initial location given in Table 10. The total time of travel remains unchanged. Furthermore, the prescribed final location will be close to the single vehicle reference case presented previously. One of the vehicles (here vehicle 1) will start from the same initial location as in the above case. This gives us a rough idea of what its trajectory should like.

GA Parameters

All the parameters used for this simulation are given in Table 10

<table>
<thead>
<tr>
<th>Table 10 : set of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-vehicle rendezvous</td>
</tr>
<tr>
<td>$N_v$</td>
</tr>
<tr>
<td>$n_{pop}$</td>
</tr>
<tr>
<td>$n^i$</td>
</tr>
<tr>
<td>$N$</td>
</tr>
<tr>
<td>$p_{mut}$</td>
</tr>
<tr>
<td>$f_{stop}$</td>
</tr>
<tr>
<td>$\gamma_{min}$</td>
</tr>
<tr>
<td>$\gamma_{max}$</td>
</tr>
<tr>
<td>$t_0$</td>
</tr>
<tr>
<td>$t_f$</td>
</tr>
</tbody>
</table>

Convergence

The best, worst and average fitness values are plotted on Figure 17a. The modified fitness values are plotted on Figure 17b. It appears clearly that the convergence to the optimum solution happened reasonably quickly, in 232 generations.
The plot of the best trajectories found until convergence is reached is not given here. Indeed, a new and better trajectory was found at almost every generation. The plot was therefore hard to read.

**Best trajectory**

The final and optimum trajectory for this reference case in plotted on Figure 18. Figure 19 shows a zoom on the region of the prescribed final location, represented by the red cross. We considered that any solution for which the two vehicles were in the vicinity of the prescribed final location is acceptable. The vicinity itself is defined by the box surrounding the prescribed ending point on Figure 21. The acceptable error margin in both directions is 5 % with respect to the final location $x$ and $y$ coordinates. The solution found by the GA makes the two vehicles' final locations lie in the vicinity of the prescribed final location.

Notice that, as in the last single vehicle case, the vehicles have to dive deeper than the actual prescribed depth, since the total time of travel is prescribed. If the vehicle's thruts were constantly aiming at the prescribed final location, they would both reach it for $t < t_f$. 

Figure 17a: GA convergence - raw fitness  
Figure 17b: GA convergence - modified fitness
The solution $\gamma(t)$ corresponding to the optimum trajectory is plotted on Figure 20.

**Results**

The characteristics of the optimum solution are given in Table 1.
The maximum error is of 1.14 %, for the final $y$ coordinate of vehicle 1 with respect to the prescribed final $y_f$, which is very reasonable. Furthermore, convergence was reached in less than 8 minutes.

<table>
<thead>
<tr>
<th>Table 11 : Optimum solution characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 2-vehicle rendezvous -</td>
</tr>
<tr>
<td>$N_{gen}$</td>
</tr>
<tr>
<td>CPU time</td>
</tr>
<tr>
<td>prescribed $x_f$</td>
</tr>
<tr>
<td>actual $x_{1f}$</td>
</tr>
<tr>
<td>error on $x_{1f}$</td>
</tr>
<tr>
<td>actual $x_{2f}$</td>
</tr>
<tr>
<td>error on $x_{2f}$</td>
</tr>
<tr>
<td>prescribed $y_f$</td>
</tr>
<tr>
<td>actual $y_{1f}$</td>
</tr>
<tr>
<td>error on $y_{1f}$</td>
</tr>
<tr>
<td>actual $y_{2f}$</td>
</tr>
<tr>
<td>error on $y_{2f}$</td>
</tr>
<tr>
<td>fitness $f$</td>
</tr>
</tbody>
</table>

### Degenerate cases

#### Dive-type

For this problem, the two vehicles start very close to each other and have to meet at a very deep location. As mentioned before for this kind of trajectory, for a single vehicle, trying to compute a perfect dive is irrelevant since the perfect dive solution is added to the initial population as an improvement. Table 12 gives the set of parameters used to initialize the GA. Notice that for this case, the values of $\gamma_{\text{min}}$ and $\gamma_{\text{max}}$ are different for each vehicle, to reduce the computational time. The parameters that changed compared to the reference case are in red.
Table 12: set of parameters
- 2-vehicle rendezvous - Dive -

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_v$</td>
<td>2</td>
<td>$t_0$</td>
</tr>
<tr>
<td>$n_{pop}$</td>
<td>100</td>
<td>$t_f$</td>
</tr>
<tr>
<td>$n_i$</td>
<td>8</td>
<td>$x_{10}$</td>
</tr>
<tr>
<td>$N$</td>
<td>40</td>
<td>$y_{10}$</td>
</tr>
<tr>
<td>$p_{mut}$</td>
<td>.05</td>
<td>$V_{10}$</td>
</tr>
<tr>
<td>$f_{stop}$</td>
<td>.002</td>
<td>$x_{20}$</td>
</tr>
<tr>
<td>$\gamma_{1 \min}$</td>
<td>0</td>
<td>$y_{20}$</td>
</tr>
<tr>
<td>$\gamma_{1 \max}$</td>
<td>$\pi$</td>
<td>$V_{20}$</td>
</tr>
<tr>
<td>$\gamma_{2 \min}$</td>
<td>$\pi$</td>
<td>$x_f$</td>
</tr>
<tr>
<td>$\gamma_{2 \max}$</td>
<td>$\pi$</td>
<td>$y_f$</td>
</tr>
<tr>
<td>$T$</td>
<td>.6</td>
<td>$N_{gen}$</td>
</tr>
</tbody>
</table>

This case is obviously more difficult than the reference case. Hence it took 397 generations for the best fitness to become less than the stopping criterion of 0.001. The next figure shows how the fitness values converge. The steep steps from one generation to the other are very characteristic of the way the fitness values are supposed to behave.

Figure 21: Fitness convergence - Dive type trajectory -

Rendezvous trajectories are more complex to solve than single vehicle trajectories. They require thus more generations to find the optimum solution. Just like for the
reference case for rendezvous, then, the best trajectories plot contained too many information and will not be shown.

The next two graphs are the optimum trajectory as well as a zoom of the area around the final location. Again, the two vehicles end in the vicinity of the prescribed final location. The maximum error made on the prescribed final $x$ and $y$ coordinates is of about 2.5%. The trajectory looks exactly like what one would expect. This is one more time very encouraging regarding the efficiency and robustness of the algorithm.

![Figure 22: Optimum trajectory](image)

![Figure 23: Zoom of the optimum trajectory](image)

The optimum solution, $\gamma(t)$ is plotted below. Notice that for this case, the values of $\gamma_2$ have obviously not been sorted in descending order, as specified in the Improvements section. For this special case, it is trivial that $\gamma$ is going to increase, globally speaking.

Also notice that even though we sort the values of $\gamma$ to make $\gamma(t)$ a monotonic function, it is then smoothed, by approximating it by a fourth order polynomial. This is why $\gamma_1$ here is not perfectly monotonic. Nevertheless, the overall trend of $\gamma_1$ is to decrease.
It appears very clearly that the two curves are almost symmetric with respect to 90 degrees. The results of simulation are recapitulated below. Notice that the CPU computational time is close to 2 min and 13 seconds on a T2400 - 2 GHz computer. It is larger than the reference case, as expected, but in a very reasonable range.

---

1All the simulations have been run on a Pentium 4 - 2 GHz computer using Windows XP Professional except this case and the next one.
Table 13: Optimum solution characteristics
- 2-vehicle rendezvous - Dive -

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{gen}$</td>
<td>190</td>
</tr>
<tr>
<td>CPU time</td>
<td>132.75 s</td>
</tr>
<tr>
<td>prescribed $x_f$</td>
<td>1</td>
</tr>
<tr>
<td>actual $x_{1f}$</td>
<td>0.9750</td>
</tr>
<tr>
<td>error on $x_{1f}$</td>
<td>2.501%</td>
</tr>
<tr>
<td>actual $x_{2f}$</td>
<td>0.9918</td>
</tr>
<tr>
<td>error on $x_{2f}$</td>
<td>0.819%</td>
</tr>
<tr>
<td>prescribed $y_f$</td>
<td>5.4</td>
</tr>
<tr>
<td>actual $y_{1f}$</td>
<td>5.4180</td>
</tr>
<tr>
<td>error on $y_{1f}$</td>
<td>0.333%</td>
</tr>
<tr>
<td>actual $y_{2f}$</td>
<td>5.4366</td>
</tr>
<tr>
<td>error on $y_{2f}$</td>
<td>0.677%</td>
</tr>
<tr>
<td>fitness $f$</td>
<td>$1.18 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

**Shallow depth - close rendezvous point**

Like in the previous section for a single vehicle trajectory, we are now going to study the case where the prescribed final location is relatively close to the starting points of both vehicles. The parameters of the simulation are given in Table 14. Again, the parameters that differ from the reference case are listed in red.

Table 14: set of parameters
- shallow depth - close rendezvous point -

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_v$</td>
<td>2</td>
</tr>
<tr>
<td>$n_{pop}$</td>
<td>100</td>
</tr>
<tr>
<td>$n^*$</td>
<td>8</td>
</tr>
<tr>
<td>$N$</td>
<td>40</td>
</tr>
<tr>
<td>$p_{mut}$</td>
<td>.05</td>
</tr>
<tr>
<td>$f_{stop}$</td>
<td>.001</td>
</tr>
<tr>
<td>$\gamma_{min}$</td>
<td>$-\frac{\pi}{2}$</td>
</tr>
<tr>
<td>$\gamma_{max}$</td>
<td>$\frac{\pi}{2}$</td>
</tr>
<tr>
<td>$t_0$</td>
<td>0</td>
</tr>
<tr>
<td>$t_f$</td>
<td>5</td>
</tr>
<tr>
<td>$T$</td>
<td>.6</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>0</td>
</tr>
<tr>
<td>$y_{10}$</td>
<td>0</td>
</tr>
<tr>
<td>$V_{10}$</td>
<td>0</td>
</tr>
<tr>
<td>$x_{20}$</td>
<td>1</td>
</tr>
<tr>
<td>$y_{20}$</td>
<td>0</td>
</tr>
<tr>
<td>$V_{20}$</td>
<td>0</td>
</tr>
<tr>
<td>$x_f$</td>
<td>3</td>
</tr>
<tr>
<td>$y_f$</td>
<td>2</td>
</tr>
<tr>
<td>$N_{gen}$</td>
<td>0</td>
</tr>
</tbody>
</table>

The behavior of the worst, best and average fitness values from one generation to the other is again typical for a GA. The fitness of the solutions improves very quickly at first
until optimum regions are found in the design space. The algorithm then searches locally to eventually find the global minimum of the fitness function.

![Fitness convergence - shallow depth - close rendezvous point -](image)

In this case too, the graph of the best trajectories was not very useful because it contained too much information. The optimum trajectory is plotted on Figure 26. It has been found in 157 generations or 107 seconds on a T2400 - 2GHz computer. Figure 27 shows a zoom of the vicinity, as defined previously, of the final prescribed location.
Finally, the optimum solution is plotted on the figure below.

Table 15 gives all the characteristics of the solution.
Table 15: Optimum solution characteristics
- 2-vehicle rendezvous - Too close final location -

<table>
<thead>
<tr>
<th>$N_{gen}$</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>107 s</td>
</tr>
<tr>
<td>prescribed $x_f$</td>
<td>3</td>
</tr>
<tr>
<td>actual $x_{1f}$</td>
<td>2.9981</td>
</tr>
<tr>
<td>error on $x_{1f}$</td>
<td>0.063%</td>
</tr>
<tr>
<td>actual $x_{2f}$</td>
<td>3.0293</td>
</tr>
<tr>
<td>error on $x_{2f}$</td>
<td>0.978%</td>
</tr>
<tr>
<td>prescribed $y_f$</td>
<td>2</td>
</tr>
<tr>
<td>actual $y_{1f}$</td>
<td>2.0171</td>
</tr>
<tr>
<td>error on $y_{1f}$</td>
<td>0.854%</td>
</tr>
<tr>
<td>actual $y_{2f}$</td>
<td>2.0137</td>
</tr>
<tr>
<td>error on $y_{2f}$</td>
<td>0.683%</td>
</tr>
<tr>
<td>fitness $f$</td>
<td>$6.7 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

For all the above studied cases, the developed Genetic Algorithm managed to find at least near optimal solutions in a reasonable computational time. This is encouraging regarding the way it has been programmed. The next step is now to study spacecraft rendezvous, which will be presented in the next Chapter.
Chapter 5
Continuous low-thrust rendezvous between spacecraft

5.1 Description of the problem

We are now going to focus on spacecraft trajectories optimization, and more specifically on rendezvous trajectories. Two spacecraft, starting from the same circular orbit around an attracting body have to meet on a different circular orbit around the same body at a given location and in a given time, coplanar to the initial one so that the problem is two-dimensional. The location of a vehicle is defined by the two polar coordinates \((r, \alpha)\) in an inertial frame, as shown on the figure below.

![Figure 29: Spacecraft's problem notation](image)

The state of the vehicle is characterized by its distance from the center of attraction \(r(t)\), its true anomaly angle \(\alpha(t)\), measured counterclockwise with respect to the \(x\)-axis, its radial component of velocity \(u(t)\), and its velocity component \(v(t)\) perpendicular to \(r\).

5.1.1 Type of vehicles

We are going to study continuous low-thrust rockets, for example ion-thrusted. Impulsive maneuvers have been studied extensively and are much easier to solve
analytically. After the impulse, a spacecraft is passive on a Keplerian orbit and its trajectory can be easily determined. In our case, all spacecraft are active at all time during the maneuver. This complicates the problem a lot since we cannot treat the maneuver as an impulsive maneuver. We are dealing with a cooperative problem, as opposed to a chaser-target kind of problem. Of course, cooperative rendezvous only makes sense if both vehicles have comparable size and propulsive capability.

5.1.2 Simplifying assumptions

First, we are going to assume that there is no perturbation of the gravity field (for example oblateness of the attracting body). We will also assume that there is no attraction between the two spacecraft as it is negligible with respect to the main attracting body: we are solving a two-body problem.

Also, without loss of generality, we will assume that the two spacecraft are identical, meaning that the spacecraft's initial mass are the same, as well as their (constant) thrust magnitude. Furthermore, the mass flow rate of the engines \( \dot{m} \) is going to be held constant.

5.1.3 Mathematical formulation of the problem

Equations of motion

The spacecraft are governed by the following equations of motion, derived in a polar coordinate system. Bryson (1999) gives the polar form of the equations of motion as:

\[
\begin{align*}
\frac{dr}{dt} &= u \\
\frac{m}{r} \frac{dv}{dt} &= m \frac{v^2}{r} - m \frac{\mu}{r^2} + T \sin \theta \\
\frac{m}{r} \frac{dv}{dt} &= -m \frac{uv}{r} + T \cos \theta \\
\frac{d\alpha}{dt} &= \frac{v}{r}
\end{align*}
\]
These equations are then nondimensionalized, using the following reference parameters:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>( r_0 )</td>
<td>Initial circular orbit radius</td>
</tr>
<tr>
<td>( kg )</td>
<td>( m_0 )</td>
<td>Initial mass of a vehicle</td>
</tr>
<tr>
<td>( s )</td>
<td>( t^* )</td>
<td>( \sqrt{\frac{r_0^3}{\mu}} )</td>
</tr>
<tr>
<td>( m.s^{-1} )</td>
<td>( V_0 )</td>
<td>( \sqrt{\frac{\mu}{r_0}} )</td>
</tr>
<tr>
<td>( N ) or ( m.s^{-2} )</td>
<td>( T_0 )</td>
<td>( \frac{\mu m_0}{r_0^2} )</td>
</tr>
</tbody>
</table>

Notice that the reference velocity is the initial speed of each spacecraft, when they are passive on the initial circular orbit with no radial velocity, before starting the maneuver.

Thus, we have:

\[
\begin{align*}
  r &= r_0 \tau \\
  m &= m_0 \tilde{m} \\
  t &= \sqrt{\frac{r_0^3}{\mu} t} \\
  \left\{ \begin{array}{l}
    u = \sqrt{\frac{\mu}{r_0 u}} \\
    v = \sqrt{\frac{\mu}{r_0 v}} 
  \end{array} \right. \\
  T &= \frac{\mu m_0}{r_0^2} \tau
\end{align*}
\]

where \( \tau \) is nondimensionalized thrust.

Combining Equations (5.1), (5.5), (5.7) and (5.8), we get:

\[
\frac{r_0}{\sqrt{\frac{r_0^3}{\mu}}} \left( \frac{d\tau}{dt} \right) = \sqrt{\frac{\mu}{r_0 u}}
\]
That simplifies into:

\[ \frac{d\tau}{dt} = \bar{u} \quad (5.10) \]

Then, combining Equations (5.2), (5.6), (5.7), (5.8) and (5.9), we get:

\[ m_0 \bar{m} \sqrt{\frac{\mu}{r_0}} \frac{d\bar{u}}{dt} = m_0 \bar{m} \frac{\mu}{r_0} \frac{\bar{v}^2}{\bar{r}} - m_0 \bar{m} \frac{\mu}{r_0^2 \sin^2 \theta} + \frac{\mu m_0}{r_0^2} \tau \sin \theta \]

We can simplify both sides by \( \frac{\mu m_0}{r_0^2} \). We get:

\[ \frac{d\bar{u}}{dt} = \frac{\bar{v}^2}{\bar{r}} - \frac{\bar{m}}{\bar{r}^2} + \tau \sin \theta \quad (5.11) \]

Notice that in this case, we do not consider the mass of the vehicles to be constant.

The mass can be expressed as:

\[ m(t) = m_0 + \dot{m} t \]

where \( \dot{m} < 0 \).

Using Equations (5.6) and (5.7), it follows that:

\[ m_0 \bar{m}(\bar{t}) = m_0 + \dot{m} \sqrt{\frac{r_0^3}{\mu} \bar{t}} \]

\[ \bar{m}(\bar{t}) = 1 - \frac{\dot{m} \sqrt{r_0^3 \bar{t}}}{m_0 \mu} \quad (5.12) \]

Let us define the following nondimensional parameter:

\[ B = \frac{\dot{m}}{m_0} \sqrt{\frac{r_0^3}{\mu}} \quad (5.13) \]

Equation (5.12) becomes:

\[ \bar{m}(\bar{t}) = 1 - B \bar{t} \quad (5.14) \]
Substituting Equation (5.14) into Equation (5.11) and rearranging the equation, we get:

\[
\frac{d\bar{u}}{dt} = \frac{\bar{v}^2}{\bar{r}} - \frac{1}{\bar{r}^2} + \frac{\tau}{1 - Bt} \sin \theta \tag{5.15}
\]

Similarly, Equations (5.3) and (5.4) become:

\[
\frac{d\bar{v}}{dt} = -\frac{\bar{u}\bar{v}}{\bar{r}} + \frac{\tau}{1 - Bt} \cos \theta \tag{5.16}
\]

\[
\frac{d\bar{\alpha}}{dt} = \frac{\bar{v}}{\bar{r}} \tag{5.17}
\]

Now, we also know the following initial conditions for each spacecraft \( j \):

\[
\begin{align*}
\bar{r}_j(0) &= 1 \\
\bar{u}_j(0) &= 0 \\
\bar{v}_j(0) &= 1 \\
\bar{\alpha}_j(0) &= \alpha_{j0}
\end{align*}
\tag{5.18}
\]

For the reasons explained in the Objective function Section, we also have the following terminal constraints, with \( \bar{t}_f \) given:

\[
\begin{align*}
\bar{r}_j(\bar{t}_f) &= \bar{r}_f \\
\bar{u}_j(\bar{t}_f) &= 0 \\
\bar{v}_j(\bar{t}_f) &= \frac{1}{\sqrt{\bar{r}_f}}
\end{align*}
\tag{5.19}
\]

Where \( \bar{u}_f \) and \( \bar{v}_f \) correspond to the radial and tangent velocities, respectively, that the spacecraft must have in order to stay on the circular orbit of radius \( \bar{r}_f \).

**Objective function**

There are many different ways of formulating a rendezvous problem in this situation. We might try to minimize the transfer time \( t_f \) knowing the final radius \( r_f \) and true anomaly \( \alpha_f \). Equivalently, we could attempt to maximize the radius \( r_f \) in a given time \( t_f \). Here, \( r_f \) and \( t_f \) are prescribed, and we want to minimize the difference between \( r_j(t_f) \) and \( r_f \), where \( j \in [1, N] \) and minimize the difference between \( \alpha_{j1}(t_f) \) and \( \alpha_{j2}(t_f) \) for all
(j_1, j_2) \in \{j_1 \in [1, N_v], j_2 \in [1, N_u]; j_1 < j_2\}. Basically, we are trying to minimize the error between the actual final radii of the vehicles and the prescribed final radius, and at the same time minimize the difference in true anomaly angle between the vehicles. This ensures that all the vehicles have to be on the prescribed final orbit and that they are all close to each other. Now, for spacecraft rendezvous, not only do we want the vehicles to be close to each other on a prescribed orbit, but we also constrain their tangent and radial velocities. Indeed, we do not only want them to meet but to stay on the circular orbit defined by the final radius. The final velocities \( u_f \) and \( v_f \) depend on \( r_f \). We want to minimize the difference between \( u_j(t_f) \) and \( u_f \) and between \( v_j(t_f) \) and \( v_f \) for all \( j \in [1, N_v] \).

Overall, the objective function can be formulated as:

\[
f = \frac{\beta_1}{\sum_k \beta_k} \sum_{j=1}^{N_u} [r_f - r_j(t_f)]^2 + \frac{\beta_2}{\sum_k \beta_k} \sum_{j=1}^{N_u} [u_f - u_j(t_f)]^2 \\
+ \frac{\beta_3}{\sum_k \beta_k} \sum_{j=1}^{N_u} [v_f - v_j(t_f)]^2 + \frac{\beta_4}{\sum_k \beta_k} \sum_{j_1=2}^{N_1} \sum_{j_2=1}^{j_1-1} [\alpha_{j_1}(t_f) - \alpha_{j_2}(t_f)]^2
\]

The errors are squared to make the objective function even more sensitive. \( \beta_k \) for \( k \in [1, 4] \) are weight corresponding to the design variables \( r, u, v \) and \( \alpha \) respectively.

For example, for \( N_v = 2 \):

\[
f = \frac{\beta_1}{\sum_k \beta_k} \left\{ [r_f - r_1(t_f)]^2 + [r_f - r_2(t_f)]^2 \right\} \\
+ \frac{\beta_2}{\sum_k \beta_k} \left\{ [u_f - u_1(t_f)]^2 + [u_f - u_2(t_f)]^2 \right\} \\
+ \frac{\beta_3}{\sum_k \beta_k} \left\{ [v_f - v_1(t_f)]^2 + [v_f - v_2(t_f)]^2 \right\} \\
+ \frac{\beta_4}{\sum_k \beta_k} [\alpha_1(t_f) - \alpha_2(t_f)]^2
\]
Formulation of the optimal control problem

We now have all the information required to formulate the problem:
Find the optimum \( \theta_j(t) \) for \( t \in [0, \tau_f] \) and \( j \in [1, N_v] \) to minimize

\[
    f(\theta_j) = \frac{\beta_1}{\sum_{k} \beta_k} \sum_{j=1}^{N_v} [r_f - r_j(t_f)]^2 + \frac{\beta_2}{\sum_{k} \beta_k} \sum_{j=1}^{N_v} [u_f - u_j(t_f)]^2
\]

\[
+ \frac{\beta_3}{\sum_{k} \beta_k} \sum_{j=1}^{N_v} [v_f - v_j(t_f)]^2 + \frac{\beta_4}{\sum_{k} \beta_k} \sum_{j=1}^{N_v} \sum_{j=2}^{N_v} [\alpha_{j1}(t_f) - \alpha_{j2}(t_f)]^2
\]

subject to: the state equations for each vehicle \( j \):

\[
    \frac{d\vec{r}}{dt} = \vec{u}
\]

\[
    \frac{d\vec{u}}{dt} = \frac{\vec{v}^2}{\tau} - \frac{1}{\tau^2} + \frac{\tau}{1 - B\tau} \sin \theta
\]

\[
    \frac{d\vec{v}}{dt} = -\frac{\vec{uv}}{\tau} + \frac{\tau}{1 - B\tau} \cos \theta
\]

\[
    \frac{d\alpha}{dt} = \frac{\hat{v}}{\hat{r}}
\]

the initial conditions for each vehicle \( j \):

\[
    \begin{align*}
        r_{f_j}(0) &= 1 \\
        u_{j}(0) &= 0 \\
        v_{j}(0) &= 1 \\
        \alpha_{j}(0) &= \alpha_{j0}
    \end{align*}
\]

- and the terminal constraints:

\[
    \begin{align*}
        r_{j}(\tau_f) &= r_f \\
        u_{j}(\tau_f) &= 0 \\
        v_{j}(\tau_f) &= \frac{1}{\sqrt{r_f}}
    \end{align*}
\]
Method of integration of the ODEs

For this problem we will use a continuous approach and integrate the equations of motion for one spacecraft using the Runge-Kutta 4 method implemented in Matlab®, between \( t_0 = 0 \) to a given \( t_f \) for the initial conditions (5.18).

Note however that to use a GA to formulate the problem, the discrete approach is inherent. Indeed, some discrete values of \( \theta \) will have to be picked to constitute the chromosome solution. But, as we previously did for the UWV problems, the values of \( \theta \) are going to be smoothed and approximated by a polynomial. During the integration, the values of \( \theta \) at any time \( t \) (not only at a discrete time step) can be calculated exactly knowing the polynomial coefficients.

5.2 Formulation of the problem for GA and Matlab®

5.2.1 Choice of chromosome

As mentioned before, the equations of motion are going to be integrated using a 4th order Runge Kutta method, but we need to discretize the values of \( \theta \) to formulate the problem for a GA. We will represent a continuous history of \( \theta \) by \( N \) discrete values that \( \theta \) takes at \( N \) times \( t_i \) uniformly distributed between 0 and \( t_f \).

As in the previous case, we also have to choose the number of bits \( n^i \) used to encode each value of \( \theta \).

5.2.2 Handling the control histories

When studying UWV, the monotonic variation of \( \gamma \) was obvious. In the present case, however, we do not know beforehand how \( \theta \) will behave. It strongly depends on the prescribed travel duration \( t_f \). Therefore, we are only going to smooth \( \theta \) using functions implemented in Matlab® to compute the coefficients of a polynomial that best fits the discrete values of \( \theta \). The values of the polynomial at any time \( t \) can then be computed using the coefficients calculated by Matlab and used during the integration.
5.2.3 Reference parameters and constraints

We need then to determine some reasonable values for the nondimensionalized parameters. Even though those values might slightly change for the next test cases, we will keep the same order of magnitude.

We are going to use reference parameters corresponding to a transfer from Earth orbit, nearly circular and Mars orbit. Thus:

\[ r_0 = 1 \text{AU} = 1.4959787 \times 10^8 \text{km} \]

\[ \mu = \mu_{\text{Sun}} = GM_{\text{Sun}} \]

where \( G \) is the universal gravity constant and \( M_{\text{Sun}} \) is the mass of the Sun. It follows that:

\[ \mu = 1.3271244 \times 10^{11} \text{km}^3/\text{s}^2 \]

Regarding the initial mass of the vehicles, using an example of (Bryson, 1999), we will choose a value of:

\[ m_0 = 10,000 \text{lb}_m = 4,536 \text{kg} \]

For the given initial orbit (circular Earth orbit around the Sun), we can also compute:

- the reference time:

\[ t^* = \sqrt{\frac{r_0^3}{\mu}} \]

\[ t^* = \sqrt{\frac{(1.4959787 \times 10^8)^3}{1.3271244 \times 10^{11}}} \left( \frac{\text{km}^{3/2}}{\text{km}^{3/2}/\text{s}^{2/2}} \right) \]

\[ t^* = 5.022643 \times 10^6 \text{ s} = 58.13 \text{ days} \]

- the reference velocity:

\[ V_0 = \sqrt{\frac{\mu}{r_0}} \]

\[ V_0 = \sqrt{\frac{1.3271244 \times 10^{11}}{1.4959787 \times 10^8}} \left( \frac{\text{km}^{1/2}}{\text{km}^{3/2}/\text{s}^{2/2}} \right) \]

\[ V_0 = 29.7847 \text{ km/s} \]
- the characteristic thrust:

\[
T_0 = \frac{\mu \dot{m}_0}{r_0^2}
\]

\[
T_0 = \frac{(1.3271244 \times 10^{11})(4,536) \ (km^3/s^2)(kg)}{(1.4959787 \times 10^8)^2 \ km^2}
\]

\[
T_0 = 2.69 \times 10^{-2} \ km.km/s \ or \ 26.9 \ N
\]

A typical value for the actual constant thrust of a 10,000-lb spacecraft would actually be, according to (Bryson, 1999):

\[
T = 0.85 \ lb \ or \ 3.778 \ N
\]

Thus, the nondimensionalized thrust is:

\[
\tau = \frac{T}{T_0}
\]

\[
\tau = \frac{3.778}{26.9}
\]

\[
\tau = 0.1405
\]

We also need to estimate the mass flow rate \( \dot{m} \). We know that:

\[
T = \dot{m} u_e
\]

where \( u_e \) is the exhaust velocity of the propellant at the exit of the spacecraft's nozzle.

We also know that:

\[
I_{sp} = \frac{u_e}{g}
\]

For a spacecraft with a specific impulse \( I_{sp} \approx 5,700 \ s \) according to (Bryson, 1999), we thus have:

\[
 u_e = I_{sp} g
\]

\[
 u_e = (5,700 \ s)(9.8 \ m/s^2)
\]

\[
 u_e = 55,860 \ m/s
\]

Thus,

\[
 \dot{m} = \frac{T}{u_e}
\]

\[
 \dot{m} = \frac{3.778}{55,860} \ \left( \frac{N}{m/s} \right)
\]

\[
 \dot{m} = 6.763 \times 10^{-5} \ kg/s
\]
Knowing $m$, we can finally calculate the parameter $B$:

$$B = \frac{|m|}{m_0} \sqrt{\frac{r_0^3}{\mu}}$$

$$B = \left( \frac{6.763 \times 10^{-5} \text{ kg/s}}{4,536 \text{ kg}} \right) \sqrt{\frac{(1.4959787 \times 10^8 \text{ km})^3}{1.3271244 \times 10^{11} \text{ km}^3/\text{s}^2}}$$

$$B = 0.0749$$

We can now choose reasonable values for the terminal constraints. As mentioned before, we are studying a transfer maneuver from Earth to Mars orbit. Thus $r_f$ is:

$$r_f = 1.5237 \text{ AU} = 2.2794 \times 10^8 \text{ km}$$

and

$$\bar{r}_f = 1.5237 \quad (5.21)$$

We know that:

$$u_f = 0 \text{ m/s}$$

thus:

$$\bar{u}_f = 0 \quad (5.22)$$

Also, $\bar{v}_f$ can be calculated using Equation (5.19):

$$\bar{v}_f = \frac{1}{\sqrt{\bar{r}_f}}$$

$$\bar{v}_f = \frac{1}{\sqrt{1.5237}}$$

$$\bar{v}_f = 0.8101 \quad (5.23)$$

Finally, we need to choose a reasonable time of travel $\bar{t}_f$. In (Bryson, 1999), the total time of transfer of one vehicle from the Earth to the Mars orbit is set to 3.3155 nondimensional units. In our case, we are going to choose a larger value of $\bar{t}_f$ first, since the rendezvous problem is much more complex. Then we will try to compute the optimal rendezvous trajectories for smaller values of $\bar{t}_f$ until the GA fails to find a good solution. This will correspond to the minimum time for the transfer.
We will start with:

\[ \overline{t_f} = 5.5 \]

We have now all the information required to initialize the GA for the reference case. Starting from now, we will drop the nondimensional bar notation.

### 5.3 Two spacecraft rendezvous

The set of parameters for this first case is given in Table 17.

We are going to study a rendezvous maneuver between two spacecraft, starting from Earth orbit at different true anomaly angle for the two spacecraft \( \alpha_{10} \) and \( \alpha_{20} \).

They have to rendezvous on Mars orbit with no prescribed value for their final true anomaly. They are however subject to the true anomaly rendezvous constraint explained in the Objective function section \( (\alpha_{1f} = \alpha_{2f}) \)

The given range for \( \theta \) is the largest possible range: \( \theta \in [-\pi, \pi] \) to give the maximum freedom to the vehicles, since we do not know how \( \theta \) will behave.

We will use \( N = 40 \) discrete values of \( \theta \) to represent a solution for the Genetic Algorithm.

<table>
<thead>
<tr>
<th>Table 17 : Set of parameters</th>
<th>- 2-spacecraft rendezvous -</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_v )</td>
<td>2</td>
</tr>
<tr>
<td>( n_{pop} )</td>
<td>80</td>
</tr>
<tr>
<td>( n_i )</td>
<td>6</td>
</tr>
<tr>
<td>( N )</td>
<td>40</td>
</tr>
<tr>
<td>( p_{mut} )</td>
<td>.05</td>
</tr>
<tr>
<td>( f_{stop} )</td>
<td>.0002</td>
</tr>
<tr>
<td>( \theta_{min} )</td>
<td>( -\pi )</td>
</tr>
<tr>
<td>( \theta_{max} )</td>
<td>( \pi )</td>
</tr>
<tr>
<td>( \beta_1 )</td>
<td>1</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>0.3</td>
</tr>
<tr>
<td>( \beta_3 )</td>
<td>1.5</td>
</tr>
<tr>
<td>( \beta_4 )</td>
<td>1.5</td>
</tr>
</tbody>
</table>

In the graph below, the fitness convergence of the simulation is plotted. Notice that the convergence is extremely quick and that the stopping criterion is met after only 61 generations. Other simulations with the same set of initialization parameters have been
run and their rates of convergence were overall much slower than this one. The average CPU time was 1,051.6 s $\approx 17.53$ min (average made on 30 simulations on a Pentium 4 - 2 GHz computer, Windows XP Professional). This is one of the characteristics of GAs: since the initial population is generated randomly, the rate of convergence may vary dramatically when running the same simulations many times, because none of the initial populations are going to be the same. This case was solved particularly fast, in 6 minutes (see Table 18).

![Fitness convergence](image)

**Figure 30: Fitness convergence**

- 2-spacecraft rendezvous -

The optimal trajectories of the two spacecraft are given in Figure 31. The black dotted lines represent the initial and final circular orbits. The green and blue dots represent the starting locations of vehicles 1 and 2, respectively. The green and blue lines are the two spacecraft's trajectories. Finally, the two + represent the spacecraft's final locations.

The optimum history of $\theta$ is then plotted on Figure 32.
Figure 31: Optimum trajectory - 2-spacecraft rendezvous -

Figure 32: Optimum $\theta$ history
As it appears very clearly on Figure 32, the thrust direction of vehicle 2 is almost monotonic. This heuristic could be added to the algorithm in the future. But it will be shown later that the behavior of the thrust direction is strongly related to the given total time of travel, which makes this information irrelevant in our study.

The characteristics of the simulation and of the optimum solution found by the generic algorithm are given in Table 18. The errors on \( r_f \) for both spacecraft are less than 0.5% which is very reasonable. Also, the difference between the two spacecraft\'s true anomalies is less than 0.5 deg, the radial velocities of the two vehicles less than .022 (to be compared to 0) and the errors on the tangent velocities less than 2.13 %. Overall, thus, all errors are in a reasonable range and we can assume the trajectories to be valid.

<table>
<thead>
<tr>
<th>Table 18 : Optimum solution characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-spacecraft rendezvous -</td>
</tr>
<tr>
<td>( N_{\text{gen}} )</td>
</tr>
<tr>
<td>CPU time</td>
</tr>
<tr>
<td>prescribed ( r_f )</td>
</tr>
<tr>
<td>( r_1(t_f) )</td>
</tr>
<tr>
<td>error on ( r_{1f} )</td>
</tr>
<tr>
<td>( r_2(t_f) )</td>
</tr>
<tr>
<td>error on ( r_{2f} )</td>
</tr>
<tr>
<td>( \alpha_1(t_f) )</td>
</tr>
<tr>
<td>( \alpha_2(t_f) )</td>
</tr>
<tr>
<td>( \Delta \alpha )</td>
</tr>
<tr>
<td>prescribed ( u_f )</td>
</tr>
<tr>
<td>( u_1(t_f) )</td>
</tr>
<tr>
<td>( u_2(t_f) )</td>
</tr>
<tr>
<td>prescribed ( v_f )</td>
</tr>
<tr>
<td>( v_1(t_f) )</td>
</tr>
<tr>
<td>error on ( v_{1f} )</td>
</tr>
<tr>
<td>( v_2(t_f) )</td>
</tr>
<tr>
<td>error on ( v_{2f} )</td>
</tr>
<tr>
<td>best fitness ( f )</td>
</tr>
</tbody>
</table>

The next step is now to run more simulations while decreasing the total time of travel, in order to determine the minimum time for the rendezvous maneuver.
5.4 Minimum time problem

There is no direct way to use the GA as it was programmed to compute the minimum time of the maneuver. Not knowing the total time of travel makes the integration of the equations of motion very hard and will not be explored. The alternative is to use the same code and decrease the given $t_f$, as explained before, until the GA fails to find a solution to the problem. This will correspond to the minimum time of travel for the studied case, and a simple way to overcome the difficulty of solving the minimum time problem directly. This process is unfortunately time consuming and represents relatively little interest in the scope of this thesis, since it does not require any additional analysis of the problem. Hence, we will not solve the problem here, while knowing it can easily be done.

Another solution to solve the minimum time problem would be to include the total time of travel in the design variables, or the starting time of each vehicle.

5.5 Reference case using Chebyshev polynomials to approximate $\theta$

5.5.1 Redefinition of the design variables

So far, we have used a discrete representation of the variation of $\theta$ as a solution to the problems for the GA implementation. For this approach to be valid, a relatively large number of $\theta$ values were required. The main disadvantage is that the corresponding chromosomes are large too. To reduce the size of the chromosomes as well as to use an exact representation of the variations of $\theta$, we are going to use polynomial coefficients as the design variables. We will still assume that $\theta(t)$ is a fourth order polynomial of the form:

$$\theta(t) = \sum_{n=0}^{4} a_n t^n$$

If we know the coefficients $a_n$, the value of $\theta$ at any time $t$ can be calculated. The idea is thus to use these coefficients as the design variables, which will considerably reduce the dimension of the design space. However, we are not going to use the $a_n$ directly as the design variables, but rather the coefficients of Chebyshev polynomials, described in the next section.
We chose to use a Chebyshev polynomial representation instead of a regular polynomial representation because, after trying both representations, we noticed that the lower and upper limits for the $a_n$ define a very large design space. One way to reduce its size would be to use different ranges for the $a_n$. Also, to be able to accurately define $\theta$, it was necessary to increase the number of bits $n_i$ used to encode each design variable. In this case, the new definition of the design variables was not significantly decreasing the size of a chromosome. The Chebyshev polynomials $T_c$ on the other hand can define complicated variations of $\theta$ for a narrower range of their coefficients.

5.5.2 Presentation of the Chebyshev polynomials

Chebyshev polynomials are a unique family of polynomials having very specific properties. They define a sequence of orthogonal polynomials. We are going to use the Chebyshev polynomials of the first kind $T_c$ to represent the variations of $\theta$. For a given $N_C$ number of Chebyshev polynomials, a function can be approximated by:

$$p(x) = \sum_{c=0}^{N_C} A_c T_c(x) \text{ for } x \in [-1, 1]$$

(5.1)

The Chebyshev polynomials can be defined by the contour integral:

$$T_c(z) = \frac{1}{4\pi i} \oint \frac{(1 - t^2) t^{-c-1}}{1 - 2tz + t^2} dt$$

The first Chebyshev polynomials of the first kind are:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$

and can be recursively defined by the following relation:

$$T_{c+1}(x) = 2xT_c(x) - T_{c-1}(x) \text{ for all } c \geq 1$$

(5.2)

Now, $\forall x \in [-1, 1], \forall c \in [0, N_c], -1 \leq T_c(x) \leq 1$. To be able to use Chebyshev polynomials to represent the variations of $\theta$ for $t \in [0, t_f]$ so that $\theta_{min} \leq \theta \leq \theta_{max}$, we need to change variables and rescale all $T_c$. We want an expression for $\theta$ of the form:

$$\theta(t) = \sum_{c=0}^{N_c} A_c T_c(t)$$

(5.3)
Let us define:

\[ x = 2 \left( \frac{t - t_0}{t_f - t_0} \right) - 1 \]  

(5.4)

Since \( t_0 \) will always be 0 in our case, this equation simplifies to:

\[ x = \frac{2t}{t_f} - 1 \]  

(5.5)

To ensure that \( x \in [-1, 1] \), \( \theta_{\text{min}} \leq \theta \leq \theta_{\text{max}} \), we will also have to find the appropriate upper and lower values for the \( A_c \) coefficients. This will be explained in the next section where the way Chebyshev polynomials have been used for the GA is presented.

### 5.5.3 Implementation of Chebyshev polynomials in the GA

We are going to describe how to use the \( A_c \) coefficients as the design variables of the Genetic Algorithm. Since we do not know anything about the behavior of \( \theta \), it is not necessary anymore to use actual values of \( \theta \) to represent a solution. Note that for the UWV cases, we were using the monotonicity of \( \gamma \) as an improvement. In that case, it was much easier to represent a solution by a set of discrete values of \( \gamma \). Here, the use of Chebyshev polynomials will allow a substantial decrease of the length of each chromosome depending on the choice of \( N_c \) and \( n^i \) (recall that \( n^i \) is the number of bits used to encode each design variables). For given \( N_c \) and \( n^i \), the length of a chromosome will be given by the following equation, similar to equation (4.20):

\[ L_{ch} = n^i N_c N_v \]

If there is no reason to change the order of magnitude of \( n^i \), we can however reasonably assume that a chromosome will be shorter in this case, because a lot of discrete values \( N \) were required to represent a solution accurately in the previous formulation of the GA.

We will use the first 5 Chebyshev polynomials, so that \( \theta \) will stay a degree 4 polynomial.

We now need to determine a range for the values of \( A_c \), just as we did for the values of \( \gamma \) or \( \theta \) in the previous cases. To do so, we determined the polynomial of the form (5.3)
using the transformation given by equation (5.5) that approximates the best the optimum \( \theta \) history found in Section 5.3. A Matlab® program was coded to solve for the values of \( A_c \) that give this best approximation. Then, using the maximum and minimum values of the computed \( A_c \), we know the order of magnitude we should give to the maximum and minimum values for the Chebyshev coefficients. We found:

\[
(A_c)_{\text{min}} = -1.4863 \\
(A_c)_{\text{max}} = 0.4140
\]

Those values have been found using a code developed in Matlab that solves for the Chebyshev coefficients, given a set of coefficients of a regular polynomial of any degree \( n \), so that the two polynomials (regular and Chebyshev) represent exactly the same function.

We are actually going to use the largest absolute value between \( (A_c)_{\text{min}} \) and \( (A_c)_{\text{max}} \) as the absolute values of the upper and lower limit, meaning that we have:

\[
A_{\text{min}} = -1.5 \\
A_{\text{max}} = 1.5
\]

A chromosome will thus be made of \( N_c \) large binary strings, each one containing a sequence of \( N_c \) binary strings of length \( n^i \), encoding values of \( A_c \in [A_{\text{min}}, A_{\text{max}}] \).

Then, to use the GA with as few modifications as possible, we simply added a function that will compute the value of \( \theta \) at a given time \( t \) for a given set of Chebyshev polynomials coefficients \( A_c \). This function is used in the integration function at each integration step of the Runge Kutta 4 method.

### 5.5.4 Results

As mentioned in the previous section, we will use the first five Chebyshev polynomials to represent the variations of \( \theta \). The design variables are therefore \( A_0, A_1, A_2, A_3 \) and \( A_4 \).

We first tried with \( n^i = 8 \) as in the previous cases. We got a very reasonable result and close to the previous one. In less than 300 generations, we got the following \( \theta \) history (plain lines).
Figure 33: Comparison of the Chebyshev and the discrete approach - $n^i = 8$

It appears very clearly that the optimum solution found using the Chebyshev polynomials coefficients as the design variables tends to be close to the solution found in the previous problem that uses a discrete approach (dotted lines). The results are not however identical.

We previously computed the Chebyshev polynomial coefficients corresponding to the solution found using the discrete approach. By using $n^i = 8$ with the Chebyshev polynomial approach, the GA can find a solution close to the previous one, but the $\Delta x^i$ for each coefficient which can be found using Equation (2.1) is quite large (relatively to the total range).

We thus tried to increase $n^i$ to 12 to check if the algorithm will indeed converge to a solution close to the previous one. The set of parameters used to initialize the GA is given in Table 19.
Table 19: Set of parameters - Chebyshev approach -

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_v$</td>
<td>2</td>
</tr>
<tr>
<td>$n_{pop}$</td>
<td>80</td>
</tr>
<tr>
<td>$n^i$</td>
<td>12</td>
</tr>
<tr>
<td>$N_c$</td>
<td>5</td>
</tr>
<tr>
<td>$p_{mut}$</td>
<td>0.05</td>
</tr>
<tr>
<td>$f_{stop}$</td>
<td>0.0002</td>
</tr>
<tr>
<td>$A_{min}$</td>
<td>-1.5</td>
</tr>
<tr>
<td>$A_{max}$</td>
<td>1.5</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>0.3</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>1.5</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Once again, the convergence of the algorithm, characterized by the fitness values at each generation, is very typical of a GA convergence (see next figure).

![Figure 34: $f$ convergence - spacecraft rendezvous - Chebyshev polynomials](image)
It took 192 generations for the best fitness to be below $f_{stop}$, in approximately 13 minutes. The maximum error is less than 1.8 %, for $r_1(t_f)$.

| $N_{gen}$ | 192  |
| CPU time  | 814 s |
| prescribed $r_f$ | 1.52 |
| $r_1(t_f)$ | 1.49751 |
| error on $r_1(t_f)$ | 1.719 % |
| $r_2(t_f)$ | 1.51956 |
| error on $r_2(t_f)$ | 0.272 % |
| $\alpha_1(t_f)$ | 6.0367 |
| $\alpha_2(t_f)$ | 6.0294 |
| $\Delta \alpha$ | 0.0073 rad or 0.4183 deg |
| prescribed $u_f$ | 0 |
| $u_1(t_f)$ | −0.02319 |
| $u_2(t_f)$ | −0.00699 |
| prescribed $v_f$ | 0.81 |
| $v_1(t_f)$ | 0.8209 |
| error on $v_1(t_f)$ | 1.330 % |
| $v_2(t_f)$ | 0.81078 |
| error on $v_2(t_f)$ | 0.081 % |

Overall, the solution computed by the algorithm seems to be very good and could definitely be used for a preliminary mission planning. The actual $\theta$ histories are plotted on Figure 35. By simply looking at the graph, it appears very clearly that the solution is very close to the one found using the discrete approach. The two solutions (four $\theta$ histories) are plotted on Figure 38.
The next graphs are the trajectories of the two spacecrafts and their trajectory with vectors representing the thrust, respectively.

Figure 35: Optimum $\theta(t)$ - Chebyshev polynomials

Figure 36: Optimum trajectories - Chebyshev polynomials
On the following graph, the variation of $\theta$ for vehicle 1 may look surprising: at all time $t$, the thrust is acting in the direction of the attracting body. However, because vehicle 1 starts by thrusting to decrease its velocity significantly, it gets relatively close to the Sun on a trajectory that may become hyperbolic if it stopped thrusting. This allows vehicle 1 to travel faster than vehicle 2 which is required for the rendezvous to take place on the final orbit.

![Graph showing direction of thrust - Chebyshev polynomials](image)

Figure 37: Direction of thrust - Chebyshev polynomials

Figure 38 shows $\theta(t)$ for vehicles 1 and 2 of the previous problem, using the discrete approach and for the current problem, using Chebyshev polynomials coefficients as the design variables. The two solutions are not perfectly identical, but notice that the optimum solutions' fitness values were not identical either (see Table 18 and 20).
The optimum trajectories of the two solutions to be compared are plotted on the next figure.

Figure 39: Comparison of trajectories for the discrete and the Chebyshev approach
From these results, we can reasonably conclude that the computed solution is definitely near-optimal since the GA converged to it using two different methods of representing the solution. Furthermore, recall that the results presented in the previous section had been found in a very surprisingly short CPU time. This time was not fully representative of the average time required when using this method. Therefore, even though the time required to compute the solution using the Chebyshev polynomials method is greater than the CPU time of the previous case, it is still less than the average CPU time required in the previous case.

5.6 Degenerate Cases

So far we have tested only one case of spacecraft rendezvous and improved the GA so that it does not use a discrete set of $\theta$ values as the design variables. In this section, the GA will be tested for more complicated cases.

5.6.1 Maximum initial true anomaly between the two spacecraft

In this case the spacecraft start from two opposite locations on the initial orbit. To give more freedom to the algorithm, the total time of travel has been increased from 5.5 s to 6 s. The GA parameters are given in the next table.

| Table 21 : Set of parameters | - Degenerate case 1 - |
|---|---|---|---|
| $N_v$ | 2 | $\tau$ | .1405 |
| $n_{pop}$ | 100 | $t_0$ | 0 |
| $n^i$ | 8 | $t_f$ | 6 |
| $N_c$ | 7 | $r_0$ | 1 |
| $p_{mut}$ | .05 | $\alpha_{10}$ | 0 |
| $f_{stop}$ | .001 | $\alpha_{20}$ | $\pi$ |
| $A_{min}$ | $-1$ | $u_0$ | 0 |
| $A_{max}$ | 1 | $v_0$ | 1 |
| $\beta_1$ | 3 | $r_f$ | 1.52 |
| $\beta_2$ | 2 | $u_f$ | 0 |
| $\beta_3$ | 1 | $v_f$ | 0.81 |
| $\beta_4$ | 2 | |

The characteristics of the computed solution are given in Table 22
Table 22: Optimum solution characteristics
- Degenerate case 1 -

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{gen}$</td>
<td>91</td>
</tr>
<tr>
<td>CPU time</td>
<td>798 s</td>
</tr>
<tr>
<td>prescribed $r_f$</td>
<td>1.52</td>
</tr>
<tr>
<td>$r_1(t_f)$</td>
<td>1.49502</td>
</tr>
<tr>
<td>error on $r_1(t_f)$</td>
<td>1.882 %</td>
</tr>
<tr>
<td>$r_2(t_f)$</td>
<td>1.52189</td>
</tr>
<tr>
<td>error on $r_2(t_f)$</td>
<td>0.119 %</td>
</tr>
<tr>
<td>$\alpha_1(t_f)$</td>
<td>7.3918</td>
</tr>
<tr>
<td>$\alpha_2(t_f)$</td>
<td>6.4013</td>
</tr>
<tr>
<td>$\Delta \alpha$</td>
<td>0.0094 rad or 0.5409 deg</td>
</tr>
<tr>
<td>prescribed $u_f$</td>
<td>0</td>
</tr>
<tr>
<td>$u_1(t_f)$</td>
<td>0.06778</td>
</tr>
<tr>
<td>$u_2(t_f)$</td>
<td>-0.00091</td>
</tr>
<tr>
<td>prescribed $v_f$</td>
<td>0.81</td>
</tr>
<tr>
<td>$v_1(t_f)$</td>
<td>0.80604</td>
</tr>
<tr>
<td>error on $v_1(t_f)$</td>
<td>0.504 %</td>
</tr>
<tr>
<td>$v_2(t_f)$</td>
<td>0.80006</td>
</tr>
<tr>
<td>error on $v_2(t_f)$</td>
<td>1.242 %</td>
</tr>
<tr>
<td>best fitness $f$</td>
<td>$9.5 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

The computed trajectories are plotted on the next figure:

Figure 40: Degenerate case 1 optimum trajectories
For the given problem, this case is the most difficult case for a 2-vehicle rendezvous. The GA converged relatively quickly to a very good solution. The next step is to test the algorithm for a rendezvous with more than two vehicles.

5.6.2 3-vehicle Rendezvous

The next challenging case that has been tried to test the GA is the 3-vehicle rendezvous. Once again, the total time of travel has been slightly increased. The following initializing parameters have been used:

<table>
<thead>
<tr>
<th>Table 23 : Set of parameters - 3-vehicle rendezvous -</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_v$</td>
</tr>
<tr>
<td>$n_{pop}$</td>
</tr>
<tr>
<td>$n^1$</td>
</tr>
<tr>
<td>$N_e$</td>
</tr>
<tr>
<td>$p_{mut}$</td>
</tr>
<tr>
<td>$f_{stop}$</td>
</tr>
<tr>
<td>$A_{min}$</td>
</tr>
<tr>
<td>$A_{max}$</td>
</tr>
<tr>
<td>$\beta_1$</td>
</tr>
<tr>
<td>$\beta_2$</td>
</tr>
<tr>
<td>$\beta_3$</td>
</tr>
<tr>
<td>$\beta_4$</td>
</tr>
</tbody>
</table>

Notice that this case "includes" the previous degenerate case since two of the spacecraft start on opposite sides of the initial orbit. The required computational time is much higher in this case, but eventually the GA converged to a very reasonable solution whose characteristics are given on the next table.
Table 24: Optimum solution characteristics
- 3-vehicle rendezvous

<table>
<thead>
<tr>
<th>$N_{gen}$</th>
<th>483</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>1814 s</td>
</tr>
<tr>
<td>prescribed $r_f$</td>
<td>1.52</td>
</tr>
<tr>
<td>$r_1(t_f)$</td>
<td>1.51031</td>
</tr>
<tr>
<td>error on $r_{1f}$</td>
<td>0.879 %</td>
</tr>
<tr>
<td>$r_2(t_f)$</td>
<td>1.53403</td>
</tr>
<tr>
<td>error on $r_{2f}$</td>
<td>0.678 %</td>
</tr>
<tr>
<td>$r_3(t_f)$</td>
<td>1.54538</td>
</tr>
<tr>
<td>error on $r_{3f}$</td>
<td>1.423 %</td>
</tr>
<tr>
<td>$\alpha_1(t_f)$</td>
<td>7.361</td>
</tr>
<tr>
<td>$\alpha_2(t_f)$</td>
<td>7.363</td>
</tr>
<tr>
<td>$\alpha_3(t_f)$</td>
<td>7.358</td>
</tr>
<tr>
<td>$(\Delta \alpha)_{max}$</td>
<td>0.005 rad or 0.2991 deg</td>
</tr>
<tr>
<td>prescribed $u_f$</td>
<td>0</td>
</tr>
<tr>
<td>$u_1(t_f)$</td>
<td>0.09579</td>
</tr>
<tr>
<td>$u_2(t_f)$</td>
<td>0.00810</td>
</tr>
<tr>
<td>$u_3(t_f)$</td>
<td>0.06777</td>
</tr>
<tr>
<td>prescribed $v_f$</td>
<td>0.81</td>
</tr>
<tr>
<td>$v_1(t_f)$</td>
<td>0.79867</td>
</tr>
<tr>
<td>error on $v_1(t_f)$</td>
<td>1.414 %</td>
</tr>
<tr>
<td>$v_2(t_f)$</td>
<td>0.82670</td>
</tr>
<tr>
<td>error on $v_2(t_f)$</td>
<td>2.046 %</td>
</tr>
<tr>
<td>$v_3(t_f)$</td>
<td>0.83746</td>
</tr>
<tr>
<td>error on $v_3(t_f)$</td>
<td>3.375 %</td>
</tr>
<tr>
<td>best fitness $f$</td>
<td>$9.7 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

The computed trajectories are plotted on Figure 41:
The GA conclusively solved a difficult 3-body problem. Overall, the Chebyshev approach proves to be efficient and can solve degenerate cases without increasing the computational time dramatically.
Chapter 6
Conclusion

In this work, rendezvous for UWV and continuous low-thrust spacecraft has been investigated. The optimum thrust direction of the vehicles has been determined using the developed GA.

6.1 Accomplishments

1. Development of a simple genetic algorithm with Matlab® to optimize explicit functions, for analysis on the behavior of a GA from the mathematical point of view. The algorithm has been used to solve De Jong's F3 function as an example.

2. Development of a GA for dynamic systems using a discrete approach. A continuous thrust UWV motion has been solved and its trajectory optimized for a given time of travel and a prescribed final location.

3. Rendezvous of UWV has then been investigated using a modified version of the previous GA to extend the problem to more than one vehicle's trajectory optimization. It was demonstrated than the GA is able to find optimum or at least near optimal solutions to the rendezvous problem, even for degenerate cases in a very reasonable computational time.

4. A Runge-Kutta 4 method has then been used to solve a spacecraft rendezvous problem, using first a discrete set of thrust direction values to represent a solution, followed by a Chebyshev polynomials representation of the thrust direction variations that allows using the Chebyshev polynomials coefficients as the design variables. The last method reduces the length of a chromosome, even though the decrease is not as substantial as expected. The equations of motion were integrated using a Runge Kutta 4 method. Both techniques converged to very similar solutions, which is very encouraging regarding the quality of the computed thrust direction history.

5. Conclusive test of the developed GA for a degenerate case and 3-spacecraft rendezvous.
6.2 Recommendations for future work

Some problems could have been studied using the developed Matlab® code without any modifications, except initial conditions. Those problems are for example the minimum time problem, as mentioned in Section 5.4, and rendezvous between more than two vehicles. Rendezvous between more than two vehicles has been investigated in this thesis, but not in detail, because of the significant increase of required computational time.

The code could be easily modified to study more complicated problems. For example, instead of assuming that the thrust magnitude is constant, we could add to the design variables the history of the thrust magnitude, to at least allow it the thrust to be "on" or "off". Also, a more accurate model of the gravitational field could be used, for example taking into account the oblateness of the Earth, the Sun, etc.
List of references


Cripsin, Y., 2005, "Cooperative control of a robot swarm with network communication delay", ICINCO conference


De Jong, K., 1975, "An Analysis of the Behavior of a Class of Genetic Adaptative Systems"


Hartmann, J.W., 1999, "Low thrust trajectory optimization using stochastic optimization methods", Master of Science in Aeronautical and Astronautical Engineering Thesis


