

**OPTIMIZATION OF LOW THURST SPACECRAFT TRAJECTORIES
USING A GENETIC ALGORITHM**

by

JASON COREY EISENREICH, B.S.

THESIS

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements for the Degree of
MASTER OF SCIENCE IN ENGINEERING

The University of Texas at Austin

August 1998

DEM QUANTUM INSPECTED 2

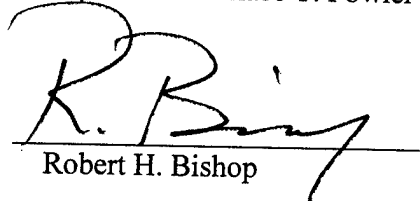
**OPTIMIZATION OF LOW THRUST SPACECRAFT TRAJECTORIES
USING GENETIC ALGORITHMS**

APPROVED BY

SUPERVISING COMMITTEE:



Supervisor Wallace T. Fowler



Robert H. Bishop

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr Wallace T. Fowler for all of his support and guidance during my time at The University of Texas at Austin. Without his comments and suggestions this would never have been accomplished. Also thanks to Dr Robert Bishop for reading my thesis.

I would also like to thank the faculty in the Department of Astronautics at the United States Air Force Academy for my undergraduate education and selecting me for the Graduate Studies Program.

Finally, I give thanks to my wife, Kody Eisenreich, and my parents, J.C. and Linda Eisenreich for all of the sacrifices they have made for me to get this far in my life.

August 5, 1998

OPTIMIZATION OF LOW THRUST SPACECRAFT TRAJECTORIES USING A GENETIC ALGORITHM

by

Jason Corey Eisenreich, M.S.E.

The University of Texas at Austin, 1998

SUPERVISOR: Wallace T. Fowler

This thesis concerns the use of genetic algorithms in the optimization of the trajectories of low thrust spacecraft. Genetic algorithms are programming tools which use the principles of biological evolution and adaptation to optimize processes. These algorithms have been found to be very useful in many different engineering disciplines. The goal of this project is to determine their applicability to the generation and optimization of low thrust spacecraft trajectories. This thesis describes the basic operating principles of genetic algorithms and then applies them to two different missions.

The first problem is an Earth to Mars mission. This mission has been solved many times using both traditional calculus of variations-based optimization techniques and genetic algorithms. Two-dimensional solutions from the literature will provide a baseline and test case to ensure the functionality of the genetic

algorithm. However, in this study, we expand this mission by using a three dimensional model.

The two-dimensional model is a good approximation for the first mission, a Mars test case, but is totally inadequate for the second mission, an asteroid rendezvous. The three dimensional model is needed to obtain a good solution for the second mission, a mission from Earth to the asteroid Eros 433. Eros is a near-earth asteroid with an orbit which is inclined from the Earth-Sun ecliptic. This mission will demonstrate the development of a three dimensional trajectory using the genetic algorithm.

The most important focus of this study is the use of a local coordinate frame known as the trajectory-tangent coordinate frame in stead of a more traditional heliocentric inertial frame. The local frame used will be able to drastically decrease the memory required to operate the genetic algorithm. This is very important in saving computing time as well as genetic algorithm effectiveness.

Table of Contents

List of Illustrations.....	vii
List of Tables.....	viii
Chapter 1: Introduction.....	1
Chapter 2: Background.....	4
2.1 Introduction.....	4
2.2 Optimization Techniques.....	4
2.2.1 Traditional Optimization Techniques.....	4
2.2.2 Genetic Algorithms.....	5
2.3 Earth to Mars Problem Description.....	8
2.4 Earth to Eros Problem Description.....	9
Chapter 3: Problem Formulation.....	12
3.1 Introduction.....	12
3.2 Equations of Motion.....	12
3.3 Genetic Algorithm Formulation.....	16
3.4 Fitness Function Development.....	18
3.5 Genetic Algorithm Parameters.....	19
Chapter 4: Results and Analysis.....	21
4.1 Introduction.....	21
4.2 Mars Test Case Initializations.....	21
4.3 Mars Results and Interpretation.....	21

4.4 Eros Test Case Initializations.....	26
4.5 Eros Results and Interpretation.....	26
4.6 Similarities and Differences.....	32
4.7 Effectiveness of Trajectory-Tangent Coordinate Frame.....	33
Chapter 5: Conclusions and Recommendations.....	35
5.1 Conclusions.....	35
5.2 Recommendations for Further Study.....	36
Appendix A – How to Run Program.....	38
Appendix B – Program Listing.....	42
References.....	68
Vita.....	69

List of Illustrations

Figure 2.1 – Earth and Mars Orbits.....	8
Figure 2.2 – Earth and Eros Orbits.....	10
Figure 3.1 - Coordinate Systems Used for Problem.....	13
Figure 3.2 – Phenotype Structure.....	17
Figure 4.1 – Earth to Mars Trajectory.....	24
Figure 4.2 – Maximum Fitness Value per Iteration.....	24
Figure 4.3 – Median Fitness Value per Iteration.....	25
Figure 4.4 – Earth to Eros Trajectory.....	29
Figure 4.5 – Maximum Fitness Value per Iteration.....	30
Figure 4.6 – Median Fitness Value per Iteration.....	31

List of Tables

Table 2.1 – Eros 433 Orbital Parameters.....	9
Table 4.1 – Earth to Mars Initial Conditions.....	22
Table 4.2 – Earth to Mars Thrust History.....	22
Table 4.3 – Earth to Mars Final Conditions.....	23
Table 4.4 – Earth to Eros Initial Conditions.....	27
Table 4.5 – Earth to Eros Thrust History.....	27
Table 4.6 – Earth to Eros Final Conditions.....	28

Chapter 1: Introduction

Low thrust propulsion is being considered as an efficient propulsion mode for spacecraft flying to celestial bodies such as Mars and Near-Earth asteroids. To date most space missions have used traditional high thrust methods to propel spacecraft into deep space. The primary example of this is the upper stage rocket motor. These engines are primarily chemical rockets such as solid, liquid, or hybrid rockets. Once a spacecraft is placed in a parking orbit by a launcher, the upper stage will fire and propel the spacecraft, causing it to escape the Earth's influence. The typical firing interval is less than 1 percent of the mission length. Low thrust propulsion is quite different. A low thrust engine can burn up to 100 percent of the transfer time. A typical low thrust system might produce up to 20 N, of thrust while a typical chemical rocket might produce up to the 35,000,000 N of thrust.

The reasons for using low thrust propulsion systems (ion engines and solar electric engines) are that they are much more efficient than chemical rockets. The measure of efficiency of a rocket engine is the specific impulse, I_{sp} . I_{sp} is a measure of how many newtons of thrust you get from a kilogram of propellant burned in one second. For chemical rockets specific impulses range from 140 seconds for a monopropellant liquid rocket to 460 seconds for a bipropellant liquid rocket (Humble, 1995). Low thrust engines have a range of

500 sec for electrothermal propulsion systems to 10,000 seconds for electrostatic propulsion systems (Humble, 1995). This increased specific impulse allows for an increased payload mass capacity. However, the low thrust, because of the low resulting acceleration, necessitates an increased amount of time to reach the destination. For this reason, low thrust propulsion is not suitable for transporting humans to destinations such as Mars. However, low thrust propulsion is very promising for use in propelling cargo spacecraft to Mars. Cargo spacecraft can be launched before the astronauts and will reach Mars before the astronauts. By using low thrust technologies, the cargo spacecraft will be able to carry more cargo mass than when propelled by chemical rockets.

High thrust and low thrust rockets also differ in the way they are analyzed. High thrust models can assume that the thrust, and thus the change in velocity, occurs instantaneously. This greatly simplifies the necessary calculations because after the change in velocity the spacecraft simply coasts on a ballistic trajectory. For low thrust trajectories, the analysis is much more involved. For each step in the numerical integration of the trajectory, the thrust, the angles of the thrusters, as well as Newton's laws must be considered.

When trying to optimize the trajectory of low thrust spacecraft these extra variables of integration make for much more difficult calculations. This thesis will show a method of optimization of low thrust trajectories, which will ease the calculation load on the analyst attempting to compute the optimal trajectories.

Genetic algorithms have shown usefulness in many areas optimization. They have been used in high thrust trajectory optimization (Piñon, 1995) as well as low thrust trajectory optimization (Rauwolf, 1995). The genetic algorithm (GA) is an optimization algorithm that mimics the principles of biological evolution and adaptation. The optimal initial values for the integration variables in a low thrust trajectory are very difficult to obtain. Traditional optimization methods are very dependent on the initial and final values of the problem. The GA alleviates this concern by randomly selecting the initial guesses for system parameters and then iterating to find near optimal values for these parameters.

The purpose of this project is to develop an implementation of a genetic algorithm that will determine three-dimensional trajectories for low thrust spacecraft trajectories for various missions. This study will look at two different scenarios. The first is a cargo mission to Mars and the second a mission to the Near-Earth asteroid, Eros. This study will show the viability of a GA in solving the low thrust trajectory optimization. It will be shown that the GA can produce near optimal solutions that can then be used as the basis for a more accurate numerical method if so desired. This study is one stepping stone in showing the potential of genetic algorithms for use in the optimization of spacecraft trajectories.

Chapter 2: Background

2.1 Introduction

This chapter shows the basics of the problem to be solved as well as baseline information on different techniques of optimization. Also, the two trajectories to be analyzed are introduced.

2.2 Optimization Techniques

2.2.1 Traditional Optimization Techniques

Traditional optimization techniques include direct and indirect methods (Kluever, 1997). Indirect methods are those methods based on the calculus of variations, such as the two point boundary value problem. These lead to accurate optimal solutions, but are very sensitive to the initial guess for the initial guess for the costate variables. Determining accurate values for these unknown variables can be very difficult. Direct methods are those which change the control variables at each iteration to continually reduce the performance index. It is usually easier to produce a good guess for the initial conditions using direct methods. Traditional methods have the drawback that the optimal value reached is not necessarily the global optimum. Among traditional methods, the indirect method is the standard which has been used for the majority of spacecraft trajectory optimization.

2.2.2 Genetic Algorithms (GAs)

A GA is an optimization scheme based on the principles of evolution. The GA starts by randomly generating a population of candidate solutions to the problem. These candidate solutions, called chromosomes, contain parameter variable values, which are used to evaluate the performance index selected for the specific problem. The performance index is evaluated using the variables stored in each chromosome. Each chromosome is then assigned a fitness value based on this performance index. The goal can be to either minimize or maximize the performance index. For example, if the maximum performance index is desired, a chromosome with a high performance index will be assigned a high fitness value. The number of variables and the desired precision for each variable determines the length of each chromosome (Piñon 1995). The GA operates for a specified number of iterations, each of which produces a new generation of chromosomes.

Each generation begins with the translation of the binary genetic material into decimal values for parameters to start the integration. These values are then used for the calculation of the spacecraft trajectory. The trajectory is calculated by integrating the equations of motion over a specified time interval. Once the trajectory has been determined, its performance index is calculated and its fitness value is assigned. Once each population member trajectory has a fitness value, the three basic genetic operators are used to generate genetic material for a new

generation. The evolutionary-based operators are selection, crossover, and mutation.

Selection is the process of choosing the parent chromosomes for the next generation. There are two basic methods for selection: the roulette wheel selection method and tournament selection. The roulette wheel selection method assigns a slot on the wheel for each population member. The slot size is proportional to the relative fitness value for each member. Those chromosomes with higher fitness values receive a better chance of reproduction. The wheel is the “spun” to select a parent chromosome. Tournament selection involves randomly selecting two population members and then choosing the chromosome with the highest fitness value. This member goes on to become a parent chromosome. This study uses the tournament selection because of its ease of coding.

Once all of the parent chromosomes have been selected, the crossover operation takes place. Crossover is the mating of two parent chromosomes to produce the next generation. The parent strings are crossed with a fixed probability to produce two child chromosomes. Typical probabilities of crossover range from 0.4 to 0.8 (Piñon 1995).

There are two methods for crossover. The two methods of crossover differ in their method of crossover. In single point crossover the two parents are randomly chosen and then a single point of crossover is randomly chosen. The

two parents then swap genetic material at this point. For example two members '00000' and '11111' could become '00111' and '11000'. In uniform crossover, there is the possibility of swapping genetic material at each point. In this case the parents '00000' and '11111' may become '01010' and '10101'. This project uses single point crossover because of the ease of programming.

The final operation to take place is mutation. Whereas selection and crossover attempt to produce the best possible members from the existing population, mutation attempts to create diversity in the population. Mutation randomly will cause a bit in a chromosome to flip from 0 to 1 or vice versa. The user assigns the probability of mutation. This allows the GA to account for candidate variables that would not be produced by the selection and crossover operators.

One other important topic on the background of GAs is the coding of the chromosomes. To this point binary numbers have been solely discussed as the method of coding. This is because binary numbers are most commonly used. They give the GA the benefits of low-order schemata and the fact that the binary alphabet is the smallest available. Schemata are the building blocks of GAs. They are short lengths of the chromosome that produce similar results in different chromosomes. Schemata are discussed in depth by Piñon and Goldberg. Low-order schemata allow those short pieces of chromosomes that continually produce

good results to grow. The fact that binary is the smallest available alphabet simply gives more schemata to build on.

2.3 Earth to Mars Problem Statement

The first problem to be studied is an Earth to Mars transfer. This problem was selected because the optimal solution for the two dimensional problem has been found through both the calculus of variations (Bryson and Ho, 1975) and through genetic algorithms (Rauwolf, 1995). To best replicate these results using a three-dimensional model, the same values as used in the literature were used for thrust, initial mass, and the mass flow rate. They are as follows:

$$T = 3.787N$$

$$m_0 = 4545.5kg$$

$$\dot{m} = 6.787 \times 10^{-5} kg / sec$$

For this problem, the trajectory starts at the Earth's position and is affected solely by the gravity of the Sun. Throughout the mission, the Sun is the only gravitational force considered. The following figure shows the orbits of the Earth and Mars around the Sun.



Figure 2.1 – Earth and Mars Orbits

As you can see, the orbits of Earth and Mars are very similar. Mars has of course a larger semi-major axis and is very slightly inclined from the Earth-Sun ecliptic as well as having a slightly more eccentric orbit than the Earth.

This is a highly practical problem as low thrust propulsion is a primary candidate for cargo missions in support of a manned mission to Mars. Because of the ability of a spacecraft with a low thrust propulsion source to carry more cargo mass for the same change in velocity as a high thrust propulsion source, more cargo can be carried to Mars at a lower cost.

2.4 Earth to Eros Problem Statement

For this mission, the spacecraft again starts at the Earth's position and is affected only by the thrust of the engine and the Sun's gravitational pull throughout the entire mission. Eros 433 is a Near-Earth asteroid with the following orbital characteristics:

Orbital Parameter	Value
Semi-major Axis	1.4583 AU
Eccentricity	0.2229
Inclination	10.832 degrees
Longitude of Ascending Node	304.497 degrees
Longitude of Perihelion	123.004 degrees

Table 2.1 – Eros 433 Orbital Parameters

Figure 2.2 shows the orbits of the Earth and Eros 433. Notice the contrast between the orbits of the Eros and Mars. Eros's orbit is much more eccentric and more inclined than the orbit of Mars.

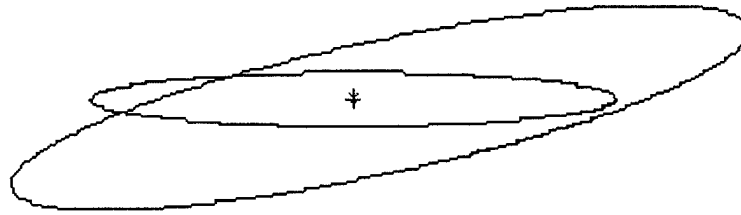


Figure 2.2 – Earth and Eros Orbits

An Earth to Eros mission would primarily serve two purposes. The first mission would be to land a spacecraft on the asteroid and obtain physical evidence as to its composition and mass characteristics. This would then lead to the second mission, which would again land on the asteroid to conduct mining operations. Although not financially viable at the current time, mining of asteroids are being considered as a future means of obtaining raw materials. Both of these missions would need a very large cargo mass capacity because of the many instruments needed and large return cargoes. It is for this reason that the study of the Earth to Eros mission uses a larger initial mass than that of the Earth to Mars mission. The values for thrust, initial mass, and mass flow rate used here are:

$$T = 3.0N$$

$$m_0 = 10,000kg$$

$$\dot{m} = 6.116 \times 10^{-5} kg / sec$$

Notice that the thrust is slightly lower than that used in the Earth to Mars test case. It then follows that the mass flow rate is also lower, as it is directly related to the thrust.

Chapter 3: Problem Formulation

3.1 Introduction

In this chapter the formulation of the problems to be solved is developed. The first step is to define the relevant coordinate frame. Part of this definition will be the introduction of the variables which the GA will use to optimize the fitness. A description of the GA used as well as the definition of the fitness function will follow.

3.2 Equations of Motion

In many previous studies of low thrust propulsion, two-dimensional coordinates have been used. These studies concentrated on missions to other planets in our solar system, such as Mars. For interplanetary missions, a two-dimensional model is adequate because of the negligible difference in the inclinations of the Earth and the other planets (excluding Mercury and Pluto). For example, the Martian orbit is inclined 1.85 degrees from the ecliptic plane. However, near-earth asteroids and comets, can have inclinations that vary greatly from the ecliptic. For example, Eros 433 is inclined 10.8 degrees to the ecliptic and Halley's Comet is inclined 162 degrees (Hamilton 1997). The Rauwolf study (1995) was two-dimensional and used a heliocentric polar coordinate frame. The control variable for this frame is the thrust angle. This frame can be easily changed into three dimensions by adding a second control angle for out of plane

thrust. However, using such a coordinate set is a disadvantage because you must vary the control angles ± 90 degrees. This presents the problem of long genetic strings in order to gain precision. The number of divisions within a range of values corresponds to the number of bits in a chromosome. For example, for the range ± 90 degrees, a chromosome with 4 bits would have 2^4 (16) divisions. Thus the variable would have only about 11 degrees of precision.

In order to increase precision while maintaining small length chromosomes, this study uses the trajectory-tangent coordinate system. This system is described in detail in Ashley (1974). These coordinates are typically used for high thrust missions such a boost from a rotating planet. The coordinate frame is used in this study because its use should lead to better precision while using shorter chromosomes and thus saving computing time. The trajectory-tangent system is rotated and translated from the inertial coordinate frame as illustrated in Figure 3.1.

As can be seen in Figure 3.1, there are four coordinate frames used in the transformation from the inertial frame to the trajectory-tangent frame. The first is the inertial coordinate frame. This has the origin at the planetary center (in this study the center of the Sun) with the Z – axis pointing through the north pole of the body and parallel to the angular momentum vector (Ashley 1974). The second coordinate frame is the planet-fixed central system. This is a rotating coordinate frame with a rotation rate the same rate as the planet. For this study,

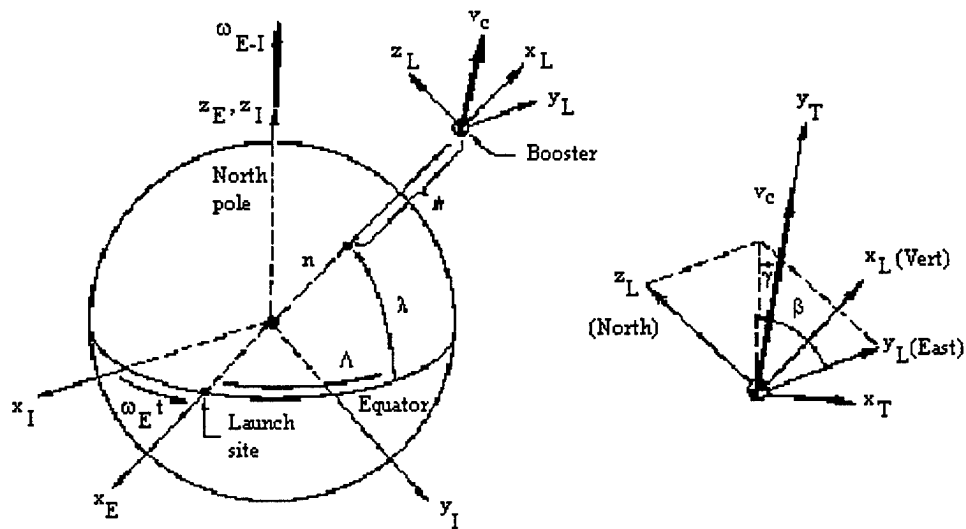


Figure 3.1 – Trajectory Tangent Coordinate System

the rotating planet-fixed coordinate system is unnecessary because the Sun does not rotate. Therefore, for our problem, the inertial and planet-fixed coordinate frames are identical. The third reference system is the locally level frame. It has its origin at the spacecraft center of mass and x-direction, x_L , is the radial direction. The unit vectors y_L and z_L point toward local east and north, respectively. The final coordinate frame is the trajectory-tangent frame. It also has its origin at the spacecraft center of mass. Its y-direction, unit vector y_T , points in the direction of the spacecraft velocity vector, v_c . The unit vector x_T is normal to y_T in the plane containing x_L .

There are four important angles in the coordinate transformation. The east longitude, Λ , and north latitude, λ are measured from the x-axis in the inertial frame. The azimuth angle, β , is measured about x_L from the y_L direction. The

elevation or flight path angle, γ , measures upward from the local horizontal. The azimuth and elevation angles determine the direction of the velocity vector.

The following equations that govern the motion of a spacecraft using the trajectory-tangent frame (Ashley 1974):

$$\dot{r} = v_c \sin \gamma \quad (3.1)$$

$$\dot{\Lambda} = \frac{v_c \cos \gamma \cos \beta}{r \cos \gamma} \quad (3.2)$$

$$\dot{\lambda} = \frac{v_c \cos \gamma \sin \beta}{r} \quad (3.3)$$

$$\dot{v}_c = -\frac{k \sin \gamma}{r^2} + \frac{T_y}{m} \quad (3.4)$$

$$\dot{\gamma} = \frac{v_c \cos \gamma}{r} - \frac{k \cos \gamma}{v_c r^2} + \frac{T_x}{m v_c} \quad (3.5)$$

$$\dot{\beta} = -\left(\frac{v_c \cos \gamma}{r}\right) \cos \beta \tan \lambda + \frac{T_z}{m v_c \cos \gamma} \quad (3.6)$$

These equations include modifications to account for the lack of a rotating central body and the absence of lift and drag. In the equations the quantities T_x , T_y , and T_z represent the components of thrust. These components are determined by two control angles, ϕ and ψ . These angles represent the gimbal angles of the propulsion engine and are typically very small. The thrust components are computed via:

$$T_x = T \sin \phi \quad (3.7)$$

$$T_t = T \cos \phi \cos \psi \quad (3.8)$$

$$T_z = T \cos \phi \sin \psi \quad (3.9)$$

3.3 Genetic Algorithm Formulation

The genetic algorithm used for a spacecraft trajectory optimization must choose a thrust history, which includes thrust direction and possibly coast arcs, which will satisfy performance conditions determined by the user to an acceptable level. These conditions and the acceptable level of performance lead to the definition of the fitness function for the study. For this study, there are four variables that need to be coded into the GA. The two thrust direction angles are obvious choices. A third variable of interest is the time of flight. A fourth variable is necessary to allow for the possibility of coast arcs. Coast arcs are segments of the trajectory where the engine is turned off. A variable called 'onoff' is used to represent if the engine is turned on or off.

It is in coding for the GA that using the trajectory-tangent system gains its advantage. The length of chromosomes is very important in the coding of a GA. The length determines the number of iterations and population members. Therefore, minimizing chromosome length minimizes computing time and maximizes the efficiency of the GA. Using the more traditional polar coordinates, the thrust direction angles are required to vary ± 90 degrees. Each angle would require 8 bits to obtain better than one degree of precision. In the trajectory-tangent coordinate frame, the gimbal angles are typically very small. For this

study, they were allowed to vary ± 7 degrees. This uses only 4 bits for each angle and attains 0.573 degrees of precision.

In order to facilitate the analysis and implementation of the GA, each trajectory is divided into 4 segments. During each segment, the thrust characteristics are assumed to remain constant. Each phenotype (set of parameters, the GA “genetic information”) is made up of 40 bits for the Earth to Eros mission and 38 bits for the Earth to Mars mission. An example of a chromosome for the Earth to Eros test case is seen in Figure 3.2. The difference between the two is the number of days which the time of flight is allowed to vary for each case. Because it has been proven in other studies that the optimal time of flight for an Earth to Mars low thrust mission is 193 days (Bryson and Ho, 1975), our study varied the time of flight from 190 to 221 days. This requires 3 bits. For the Earth to Eros mission the time of flight was allowed to vary from 142 days to 400 days. This is due to the fact that the calculus-based optimum is not known. This mission requires five bits for the time of flight. These bits are located at the end of the chromosome.

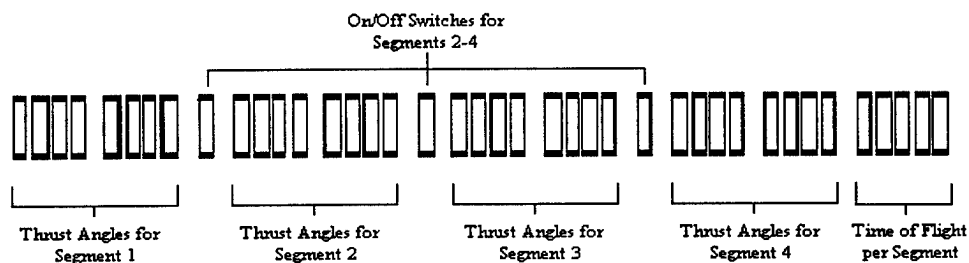


Figure 3.2 – Phenotype structure

Each of the different thrust direction angles, two for each segment, are coded into 4 bits. This gives a total of 32 bits for the thrust angles. Finally, the last three segments each have an onoff bit which precede the thrust angles. The first segment does not contain an onoff bit because it is necessary for the engine to be on to start the mission.

3.4 Fitness Function Development

The same fitness function was used for both mission evaluated. The fitness function combined measures of the relative final position and relative velocity of the spacecraft and the target. The parameters were evaluated in astronomical units. A weighting factor was used to counter the fact that the relative velocity is inherently a much smaller number than the relative position. Equation 3.10 shows the fitness function used for this study.

$$Fitness = \frac{0.01}{MISS} + \frac{0.00001}{VREL} \quad (3.10)$$

$$MISS = \sqrt{(xtgt - xsc)^2 + (ytgt - ysc)^2 + (ztgt - zsc)^2} \quad (3.11)$$

$$VREL = \sqrt{(vxtgt - vxsc)^2 + (vytgt - vysc)^2 + (vztgt - vzsc)^2} \quad (3.12)$$

where

xtgt – x-component of target position
ytgt – y-component of target position
ztgt – z-component of target position
vxtgt – x-component of target velocity
vytgt – y-component of target velocity
vztgt – z-component of target velocity
xsc – x-component of spacecraft position
ysc – y-component of spacecraft position

zsc – z-component of spacecraft position
vxsc – x-component of spacecraft velocity
vyse – y-component of spacecraft velocity
vzsc – z-component of spacecraft velocity

By having the miss distance and the relative velocity in the denominator of the fitness function, the fitness function will grow with increasing accuracy. If either the miss distance or relative velocity were zero, the fitness value would be infinite. This could cause some problems but the GA is not likely to produce such a result as it is not able to obtain such accuracy. The fitness function was designed so that the trajectory is within 1 percent of the position of the target the first component of Equation 3.10 will equal 1. If the velocity of the trajectory is within 0.001 percent of the velocity of the target, the second component of Equation 3.10 will equal 1. For example if the miss distance were 0.02 AU and the relative velocity were 0.00002 JD/AU, the fitness value would be equal to 1.

3.5 Genetic Algorithm Parameters

The GA used in this study was programmed in FORTRAN based of the algorithms found in Goldberg's text (1989). All other subroutines except for those used in integration come from the Mission Design Subroutine Library of the Department of Aerospace Engineering at the University of Texas at Austin. The two subroutines used for integration were modified from the Astro 422 Library from the Department of Astronautics at the United States Air Force Academy. The program ran on Silicon Graphics workstation.

The GA was designed using the basic operators mentioned in Chapter 2 plus one special operator. This special operator is known as elitism. This operation takes place before selection of the new generation. This operator determines the gene with the highest fitness value and ensures that it survives to the next generation. The other operators were chosen mostly for ease of programming as all options produce similar results.

Also necessary in the formulation of the GA are the following parameters: number of individuals, number of generations, crossover probability, and mutation probability. The parameters for this study were chosen based on analyses done by Goldberg and Rauwolf. Both studies cite numbers of generations and individuals similar to the number of bits in each phenotype as sufficient. Therefore, this study chose 50 generation and 50 individuals. A crossover probability of 0.65 was used and a mutation probability of 0.015 was used. These are consistent with studies done by Piñon and Rauwolf.

Chapter 4: Numerical Study

4.1 Introduction

This chapter presents the results of the numerical simulations. The analysis demonstrates the viability of using a genetic algorithm for low thrust trajectory analysis as well as the performance of the trajectory-tangent coordinate frame.

4.2 Mars Test Case Initializations

The most important initialization variable for each of the two test cases is the launch date for the mission. This was determined through a trial and error process. Determining an appropriate launch date is necessary to resolve phasing issues between the Earth and the target. For the Mars test case, this study used a launch date of 15 March 2002. This date was by numerical experimentation over dates ranging from 01 January 2002 to 31 December 2003. For the Mars case, the range of possible times of flight is only between 190 and 221 days, because of the fact that a known optimum of 193 days exists (from calculus of variations solutions to the problem) (Bryson 1975).

4.3 Mars Results and Interpretation

The Mars test case produced some interesting results. The GA was able to converge on all runs of the program. The trajectory was initiated at the position and velocity (in the inertial reference frame) given in Table 4.1.

	X	Y	Z
Position (AU)	-0.98908490	0.10244640	0.00000069
Velocity (AU/JD)	-0.00205368	-0.01717673	-0.00000062

Table 4.1 – Earth to Mars Initial Conditions

The GA produced an optimal solution with a 192 day transfer time. This is only one day off of the calculus of variations solution of 193 days. The GA does not “converge” quadratically, so this is an excellent result. Table 4.2 shows the values of the thrusting angles for each segment of flight.

	Segment 1	Segment 2	Segment 3	Segment 4
ϕ (degrees)	-4.285		-3.571	5.000
ψ (degrees)	5.714		5.714	5.714
Engine On/Off	On	Off	On	On

Table 4.2 – Earth to Mars Thrust History

A most interesting feature of the solution is the fact that a coast arc exists and that the thrust angle, ψ , is constant throughout the flight. During the second segment of the mission, the spacecraft coasts and is influenced only by the attraction of the Sun. The out-of-plane gimbal angle, ψ , which was constant throughout the flight, causes the spacecraft to constantly increase its inclination with respect to the Sun. This causes the spacecraft to approach Mars from the

Martian South Pole. The thrust history above produced the final conditions given in Table 4.3.

	X	Y	Z
Spacecraft Position (AU)	-1.53811280	0.64911999	0.03341085
Target Position (AU)	-1.53233910	0.65136474	0.05135923
Relative Position (AU)	-0.00577370	-0.00224475	-0.01794838
Spacecraft Velocity (AU/JD)	-0.01296442	-0.01094570	0.00007795
Target Velocity (AU/JD)	-0.00492701	-0.01169463	-0.00012363
Relative Velocity (AU/JD)	-0.00803741	0.00074893	0.00020158

Table 4.3 – Earth to Mars Final Conditions

These results show that the GA did very well at finding a solution, which minimized the relative position and velocity of the spacecraft and Mars. The magnitude of the relative position is about 1.5 million miles. At first glance this does not seem to be a very good solution. However, the fact that the GA produced a trajectory that comes this close to Mars is a great success. To realize the success, we must remember the fact that these trajectories are randomly produced. Guessed initial conditions produced trajectories which finished hundred of millions of miles away from the target. The ability of a program to move to the vicinity of Mars is what is what expected. The trajectories obtained are not accurate enough to be used for an actual mission, but they are clearly sufficient as a starting point for quadratically converging optimizer, which is very

sensitive to initial conditions. The following figure shows the flight path of the spacecraft and Mars throughout the time of flight.



Figure 4.1 – Earth to Mars Trajectory

Another interesting facet of the study is the performance of the GA. The Figure 4.2 shows the maximum fitness value in the population for each iteration.

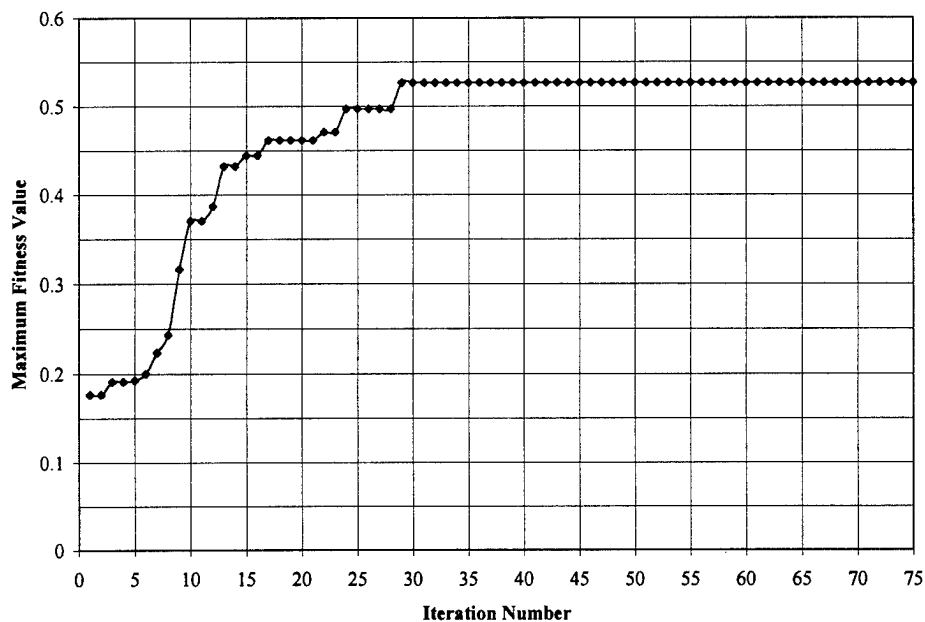


Figure 4.2 – Maximum Fitness Value per Iteration

As seen, the maximum fitness value of about 0.53 is attained after only 29 iterations. From the literature the expected number of iterations required should be equal to the length of the genetic data string. For this case, the length was 38 bits.

Another interesting statistical value is the median value of the population.

This can be seen in Figure 4.3.

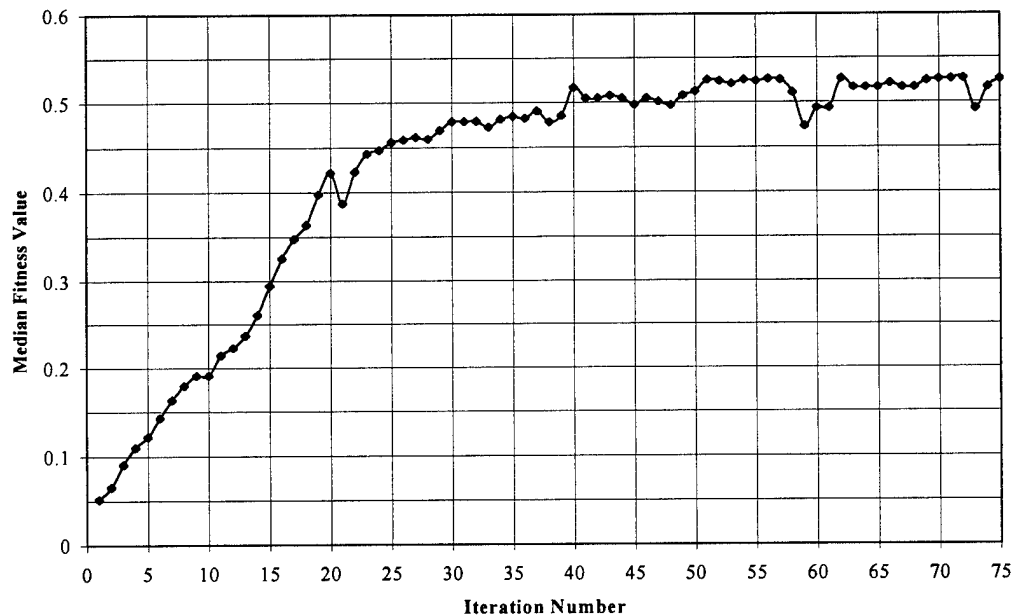


Figure 4.3 – Median Fitness Value per Iteration

This graph shows that the maximum median fitness value is reached at iteration 51. At this time the median value is 0.525368, nearly equal to the maximum. It is this is more informative with respect to the required number of iterations because it shows that a large segment of the population has similar genetic material. This defines “convergence” for a GA. It is also interesting to note that the median goes down after this point. This is due to the fact that mutations alter the genetic mix even after “convergence.”

Mutation introduces a random bit into the genetic material. Mutation can be very important if a maximum is reached that is not the global maximum. Looking at the “convergence” data, we see that 50 – 60 iterations would have been sufficient. Actually 75 iterations were used in this study. It is important to minimize the number of iterations to decrease population size as computing time increases linearly with the number of generations, or iterations (Rauwolf 1995). However, it is important to have sufficient iterations to ensure convergence.

The Mars test case showed the applicability of GAs to the generation of low thrust spacecraft trajectories. The GA was able to converge to an optimal solution that is very near the solution obtained through calculus-based methods.

4.4 Eros Test Case Initializations

As for the Mars case, an important factor is the selection of the launch date. For this case the best launch date was determined to be 15 November 2004. This was based on a trial and error search to find the date which produced the best fitness value over the years of 2003 and 2004.

4.5 Eros Test Case Results and Interpretation

The Earth to Eros test case was also successful. Once again, the GA was able to converge on all runs of the program. There was a noticeable difference in the maximum fitness value reached in the Mars test case. However, the trajectory was able to sufficiently approach Eros. The trajectory again began at the Earth's

position with the Earth's velocity in the heliocentric inertial coordinate frame.

Table 4.4 shows the initial coordinates of the trajectory.

	X	Y	Z
Position (AU)	0.59868739	0.78731214	0.00002972
Velocity (AU/JD)	-0.01397586	0.01034976	0.00000033

Table 4.4 – Earth to Eros Initial Conditions

The GA produced an optimal trajectory with a transfer time of 336 days. This is a much longer transfer time than that of the mission to Mars. This is most likely due to the larger inclination of the orbit of Eros. When calculating the change in velocity for a trajectory, a change in inclination is much more costly than a change in semi-major axis. This is amplified by the fact that it takes a low thrust engine a long time to build up a change in velocity. Table 4.5 shows the optimal thrust history generated by the GA.

	Segment 1	Segment 2	Segment 3	Segment4
ϕ (degrees)	0.000	7.050		
ψ (degrees)	5.714	-5.000		
Engine On/Off	On	On	Off	Off

Table 4.5 – Earth to Eros Thrust History

Table 4.5 shows some interesting results. The first is the fact that there are two coast arcs at the end of the mission. For this trajectory, the engine is engaged

only half of the time. This brings about two possibilities. The trajectory could use only half of the time of flight to get to Eros and then coast with Eros for the last half of the mission. The trajectory could also use the first two segments to get into an orbit that will then fly ballistically to intersect with Eros's orbit.

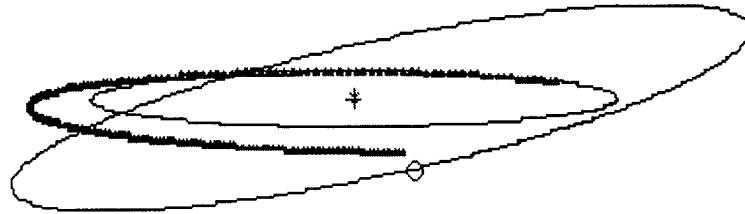
	X	Y	Z
Spacecraft Position (AU)	0.68154751	-1.66538049	-0.01423593
Target Position (AU)	0.67361681	-1.63979014	-0.07147498
Relative Position (AU)	0.00793070	-0.02559035	0.05723905
Spacecraft Velocity (AU/JD)	0.01083091	0.00290918	-0.00003022
Target Velocity (AU/JD)	0.01064372	0.00361664	0.00207037
Relative Velocity (AU/JD)	0.00018719	-0.00070746	-0.00210059

Table 4.6 – Earth to Eros Final Conditions

These values for the variables in the model generate the final conditions in Table 4.6.

These final conditions again do not look very promising upon first glance. The GA generated trajectory finishes 5.9 million miles away from Eros. However, this is again a very good result when the expectations are kept in perspective. One member of the initial population produced a trajectory that finished over 120 million miles away from Eros. This is an improvement of 2000 percent produced without any input from the user. The final conditions also answer the question as to whether the trajectory coast with Eros for the last half of

the mission or if it coasts to Eros. By looking at the z-components of the velocities we see that the trajectory and Eros are going in opposite directions. This shows that the trajectory coasts to an intersection with Eros. The trajectory



produced by the best performer is shown in the following figure.

Figure 4.4 – Earth to Eros Trajectory

Figure 4.4 shows the greater distance between the spacecraft and the target at the end of the trajectory than in the Earth to Mars test case. This was expected from looking at the numerical results. This shows the need for a more accurate optimizer. But the initial conditions produced by the GA can be used to allow the more accurate optimizer to accomplish the task in much less time. Without the initial conditions from the GA, a quadratically converging optimizer either would take a very large amount of time or could not solve the problem.

This case also exhibited some insight into the effectiveness of the GA.

Figure 4.5 shows the maximum fitness value for each iteration.

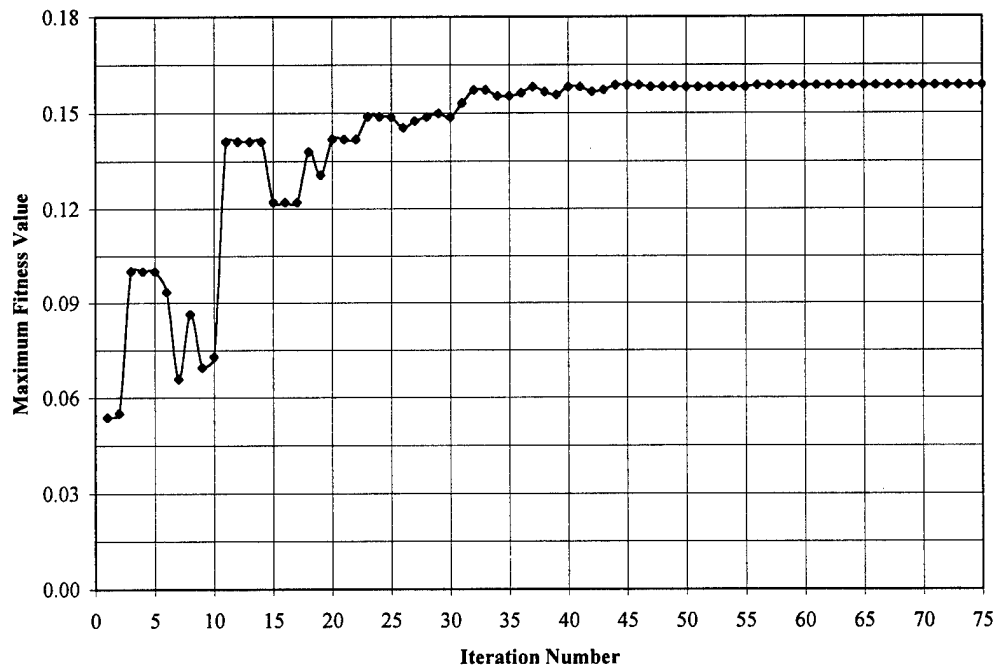


Figure 4.5 – Maximum Fitness Value per Iteration

This case took until iteration 56 to produce the maximum fitness value of 0.15868. This case seems to have stalled at a maximum fitness value slightly lower than this for the previous 10 – 15 iterations. This change suggests that a mutation took place which increased the fitness value. This higher fitness value was then allowed to go throughout the population in subsequent generations. By iteration 63, this member had proliferated enough to reach the highest median fitness value of 0.158176. This is seen in Figure 4.6, which shows the median fitness value for each generation. Again, this shows when the GA has converged

to a point where the best performer occupies the majority of the population members.

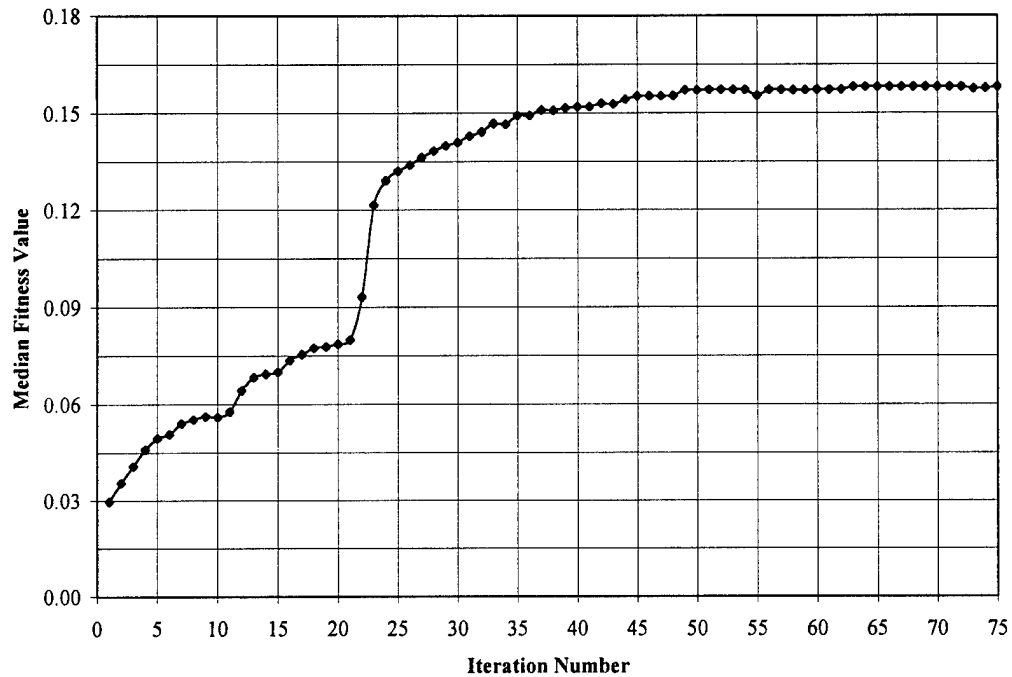


Figure 4.6 – Median Fitness Value per Iteration

This test case shows the importance of employing a sufficient number of iterations. Had the standard benchmark of using a number of iterations equal to the length of the chromosome been used, this test case would not have shown what proved to be the best performer. This test case showed that 60 – 70 iterations would be needed to exhibit convergence.

4.6 Similarities and Differences

This study presented many differences and similarities between the two test cases. In both cases, the GA was successful in generating a good solution to the problem. However, the GA in the Mars test case was able to come much closer to the target than in the Eros case, though in both cases the GA provided marked improvement over the initial populations. Although the final relative distances exceeded one million miles in both cases, the results are more than sufficient for use in a quadratically converging optimizer.

The difference between the two cases is the strategies used to approach the targets. For the Mars case, the GA produced a trajectory which, outside of the on coast arc, was thrusting constantly toward Mars. This trajectory was also approaching Mars using the same out-of-plane angle throughout the flight. For the Eros test case the strategy used was to find a trajectory that would go into an orbit which would ballistically travel towards the target. This is exhibited by the fact that the best trajectory thrusts for the first half of the mission and coasts for the last half.

Another difference between the two test cases was the number of iterations needed for the GA to produce the best performing chromosome. In the Mars test case, the GA produced the best performer in only 29 iterations and converged in 51 iterations. The Eros test case took longer to reach the maximum

value and for convergence. This difference shows the importance of selecting a number of iterations that is large enough to allow for convergence.

4.7 Effectiveness of Trajectory –Tangent Coordinate System

The most important results of this study are the effectiveness of the coordinate frame used. It was hypothesized that by using a local coordinate frame rather than a more traditional inertial frame, the GA would be able to use much smaller chromosomes. This proved to be true. As mentioned before, the chromosomes used for this study are 38 bits long for the Mars test case and 40 bits long for the Eros test case. In both cases, 4 equal length segments were used. Thus there are 8 different thrust angles (2 for each segment) stored in each chromosome. Each thrust angle requires 4 bits to obtain 0.573 degrees of precision. This is due to the fact that each angle need only be varied over the range of ± 7 degrees as measured instantaneously from the velocity vector.

If an inertial frame were used, the thrust angles must be varied at least ± 90 degrees. If 8 bits were used to represent these angles, we would have 0.703 degrees of precision, while if 9 bits are used, the angles have 0.352 degrees of precision. For the sake of discussion, we will assume 8 bits would be used for an inertial frame. Under these conditions, a chromosome for the Mars test case would need to be 70 bits long. A chromosome for the Eros test case would need to be 72 bits long.

Using the local coordinate frame saves 28 bits for each mission.

Decreasing the length of chromosomes is important because computing time is reduced in two ways. First, the computer does not have to deal with as much information when analyzing the chromosome itself, thus it can complete an iteration faster. Second, the length of the chromosome is the primary measuring stick in determining the number of iterations and the number of population members required for successful GA operation. By employing a shorter genetic string, the number of iterations and the number of population members required are decreased. By decreasing both quantities, the required computing time is drastically reduced. The fact that these three different quantities, chromosome length, number of iterations, and number of population members, each lowers the computing time, major time savings have been effected.

This effect is magnified if the number of segments in the trajectory is increased. An increase in segments is necessary when trying to obtain more precise trajectories. For example an Earth-to-Eros trajectory with 50 trajectory segments would require 454 bits when using the trajectory-tangent coordinate frame. This same mission would require 854 bits when using an inertial frame. Using the inertial frame would also require 400 more iterations and population members than the local frame. In such cases, the savings in computing time can make solving an impossible solution possible.

Chapter 5: Conclusions and Recommendations

5.1 Conclusions

This study produced some very useful results. The use of the trajectory-tangent frame proved to be very beneficial in application of a GA to the analysis of low thrust trajectories. The coordinate frame allows for shorter genetic strings, which in turn reduces the computing time. In this basic study, the effects of the trajectory-tangent frame were seen by lowering the chromosome length from 72 bits to 40 bits for the Eros test case.

The Earth to Mars was used as a validation case to ensure that solutions generated by the GA matched those in the literature. The GA was successful in matching these solutions within the expected level of accuracy. The success of the Earth to Mars test case allowed us to use the GA for the second case, a rendezvous with the near-earth asteroid, Eros 433. The importance of this case is that the orbit of Eros is sufficiently inclined from the ecliptic to force the use of a three dimensional model.

The GA was able to converge on a solution in both the Earth to Mars test case and the Earth to Eros test case. The trajectories produced by the GA for both test cases were both within the expected level of accuracy. The GA operated very well in both cases. From analysis of the two test cases, we have surmised that for the chromosomes used, 40 bits long, 50 – 60 iterations should be used. This is

about 50 percent more than is recommended in the literature. However, we feel that the ability to obtain the best solution is more important than the computing time that will be saved by using fewer iterations.

5.2 Recommendations for Further Study

There are four main recommendations for further study. The first is to integrate a more traditional optimization program with the GA to produce a hybrid optimizer. Hybrid optimization has been used to produce results that would be much more difficult to obtain using only the traditional optimization technique or the GA alone (Piñon 1995). The GA is able to obtain a solution with a sufficient accuracy to ensure rapid convergence for higher order optimization techniques.

The second recommendation focuses on the existence of coast arcs in both test cases run in this study. For the Mars test case, each segment is 48 days long and for the Eros test case the segments were 85 days long. The time of flight sets the length of the coast arcs. It is highly probable that in the true optimal solution the coast arcs would not start and stop at exactly the same points as in this study and their lengths would have been different. Therefore additional work needs to be done if optimization of these trajectories is desired.

There are two methods to make coast arcs more accurate. The first and most attractive option is to allow the GA to have a variable trajectory segment length. This would add another variable to the genetic string. A variable length

trajectory segment length would not only allow coast arcs to become the optimal length but would also allow for thrust direction change at optimal times. A second way to address this problem is to simply increase the number of trajectory segments. This method would require less modification to the existing implementation but probably would not result in as good a final solution unless a large number of trajectory segments were used.

The third recommendation is to allow for a variable mission start date rather than using the trial and error method used to determine start dates in this study. This could be accomplished by simply adding another time variable to the chromosome.

A final recommendation would also add more genetic material. The purpose would be to allow variable thrust along the trajectory. Many times it may not be optimal to use the full thrusting capacity of the engine. This change would require slight modifications to the integration subroutines used in this project.

All of these recommendations require adding genetic material to the chromosomes. This is however, not a great problem. Studies have been accomplished successfully with chromosome lengths in the hundreds of bits. Also, these effects are mitigated by the use of the trajectory-tangent coordinate frame.

Appendix A
Program Usage

Program Usage

The program used for this study is very easy to use. For this study, the program was run on a Silicon Graphics workstation at the Center for Space Research at the University of Texas at Austin. The program is written in FORTRAN 77. All subroutines that are not listed in the following appendix are available in the Mission Design Library from the Department of Aerospace Engineering at the University of Texas at Austin.

All modifications that may be needed to use the program are relatively simple. This section will list a few probable modifications and the method to accomplish these changes.

Lengthening the Chromosome or Adding Genetic Material

The first step to lengthening the chromosome is to change the variable, SIZE, in the main program. Also if the length is to be over 100 bits, the variable declaration must be changed. Second, the user must change the BIN2DEC subroutine. This subroutine changes binary value to decimal values. Each variable is listed and an equation is used for the transformation. Finally, the MAP subroutine must be changed. This subroutine maps the decimal values from BIN2DEC into the desired range. The user must change the equations in this subroutine to fit the needs of the study.

Adding Population Members

Adding population members is accomplished by simply changing the value of the variable POP in the main program. If the population is to exceed 100, the variable declaration must be changed. This goes for any subroutine which is called and is passed the variable POP.

Adding Iterations

Adding iterations is completed by changing the value of X in the statement “DO ITER = 1,X”. This statement is in the main program.

Changing the Number of Segments in the Trajectory

To change the number of segments in the trajectory, the user must first add the appropriate genetic material as prescribed above. Second, the value for the counting variable under the comment “Loop Through Each Segment” must be changed to reflect the appropriate number of segments.

Changing the Launch Date

To change the launch date the user must change the values of the array ITIME and the variable STIME.

Changing the Probability of Crossover and Mutation

To change the probability of crossover, enter the appropriate value into the variable XPROB in the subroutine CROSS. To change the probability of mutation, change the value of the variable MUTPROB in the subroutine MUTATE

Output Files

There are four different output files produced by the program. The first EROS.OUT is a listing of the initial conditions for the case and then the final values of the propagated states and the fitness value of each population member. FIT.OUT is a listing of the initial position and velocity in the inertial frame. This is followed by the final position of the position and velocity of the trajectory and the target in inertial frame. MAP.OUT is a listing of the values of each thrust angle for each population member as well as the time of flight and the onoff variable. F.OUT is a listing of the fitness value of each population member.

Changing the Target

The user may change the target by simply changing the final variable in the subroutine call for the subroutine SOLAR each time it is called for the target (this is done both in the main program and the subroutine FITNESS. A word of caution, the subroutine SOLAR is called in the main program to get the initial conditions of the trajectory (these are the conditions of Earth at the launch date and time). Changing this subroutine call will change the launch location of the trajectory.

Appendix B
Program Listing

```

*-----
*           PROGRAM GA
*
* Jason C. Eisenreich                21Oct97
*
* This program is a genetic algorithm which generates flight path
* angles for a space mission to a near Earth asteroid or any other
* celestial body using a low thrust propulsion system.
*
* Variables:
*   SIZE   Length of each bit string
*   POP    Number of population members
*   SEED   Seed for random number generation
*   GENE   Matrix of genetic material
*   XPosition of Earth in Inertial frame      km
*   JD     Julian date at start of mission    days
*   STATE  State matrix
*   STATE(1) Velocity in trajectory tangent frame km/s
*   STATE(2) Flight path angle                rad
*   STATE(3) Azimuth angle                    rad
*   STATE(4) Distance from Sun to spacecraft  km
*   STATE(5) Solar latitude                   rad
*   STATE(6) Solar longitude                  rad
*   STATE(7) Spacecraft mass                  kg
*   TEMP   Temporary vector for matrix rotation
*   PHI    Array of control angles            rad
*   PSI    Array of control angles            rad
*   VL     Velocity in inertial frame          km/s
*   STIME  Seconds at initial time            sec
*   ITIME  Initial time
*   ITIME(1) Month
*   ITIME(2) Day
*   ITIME(3) Year
*   ITIME(4) Hour
*   ITIME(5) Minute
*   DT     Time interval for Runge-Kutte integration sec
*   FIT    Array of fitness values
*   ISTATE Initial value for state matrix
*   TOF    Array of values of time of flight  sec
*   ONOFF  Array of thruster on/off designators
*   I      Counting variable

```

```

*      J      Counting variable
*      K      Counting variable
*      ITER   Iteration counter
*      RA     Initial Position of the asteroid      km
*      RS     Initial Position of the spacecraft    km
*      VA     Initial velocity of the asteroid      km/s
*      VS     Initial velocity of the spacecraft    km/s
*
*
* Constants:
*      PI      3.1415.....
*
* Subroutines:
*      JULDAY   Determines the Julian date given a calender date and
*               time
*      SOLAR    Determines the position and velocity of a body in the
*               heliocentric inertial frame in AU and AU/JD
*      MAG      Determines the magnitude of a vector
*      ROT3     Rotates a coordinate frame about its 3rd axis
*      ROT2     Rotates a coordinate frame about its 2nd axis
*      GENERATE Generates an ititial random population
*      BIN2DEC  Changes a binary string into the appropriate decimal
*               parameter values
*      MAP      Scales the paramter values to valid values
*      RK4      A fourth order Runge-Kutte integrator
*      FITNESS  Determines the fitness value of each population member
*               after integration
*      TOURNAMENT Uses tournament selection operator to select
*               parents of next generation
*      CROSS    Crossover operator to generate children of next
*               generation
*      MUTATE    Mutation operator
*
* References:
*
* -----

```

PROGRAM GA

IMPLICIT NONE

```

REAL*8 RS(4),VS(4),X(6),JD,STATE(7),TEMP(4),PHI(100,4),RA(4)
REAL*8 VL(4),STIME,DT,FIT(100),ISTATE(7),PSI(100,4),VA(4),Y(6)

```

```
INTEGER SIZE,POP,SEED,GENE(100,100),TOF(100),ONOFF(100,4)
INTEGER ITIME(5),K,J,I,ITER
```

```
*-----Initialize Chromosome Values-----
```

```
SIZE = 40
POP = 50
```

```
*-----Initialize Seed Value-----
```

```
SEED = 124687
OPEN(UNIT = 10,FILE='EROS.OUT',STATUS='UNKNOWN')
OPEN(UNIT = 11,FILE='MAP.OUT',STATUS='UNKNOWN')
OPEN(UNIT = 12,FILE='FIT.OUT',STATUS='UNKNOWN')
OPEN(UNIT = 13,FILE='F.OUT',STATUS='UNKNOWN')
```

```
*-----Intialize Launch Date-----
```

```
ITIME(3) = 2004
ITIME(1) = 11
ITIME(2) = 15
ITIME(4) = 0
ITIME(5) = 0
STIME = 0.0D0
CALL JULDAY(ITIME,STIME,JD)
```

```
*-----Initialize State Matrix-----
```

```
*-----Get Earth Position and Velocity Vector-----
```

```
CALL SOLAR(X,JD,3)
CALL SOLAR(Y,JD,10)
DO J = 1,3
  RS(J) = X(J)
  VS(J) = X(J+3)
  RA(J) = Y(J)
  VA(J) = Y(J+3)
ENDDO
CALL MAG(RS)
CALL MAG(VS)
CALL MAG(RA)
```

```

CALL MAG(VA)
WRITE(12,*)'Initial Position and Velocity'
WRITE(12,25)RS
WRITE(12,26)RA
WRITE(12,27)VS
WRITE(12,28)VA
DO J = 1,3
  RS(J) = X(J)*149597870.0D0
  VS(J) = X(J+3)*149597870.0D0/86400.0D0
  RA(J) = Y(J)*149597870.0D0
  VA(J) = Y(J+3)*149597870.0D0/86400.0D0
ENDDO
CALL MAG(RS)
CALL MAG(VS)
CALL MAG(RA)
CALL MAG(VA)

```

*-----Determine Initial Solar Lat and Lon-----

```

ISTATE(5) = DATAN(RS(2)/RS(1))
ISTATE(6) = DASIN(RS(3)/RS(4))
ISTATE(4) = RS(4)

```

*-----Determine Initial Azimuth Angle-----

```

CALL ROT3(VS,ISTATE(5),TEMP)
CALL ROT2(TEMP,ISTATE(6),VL)
ISTATE(3) = DATAN(VL(3)/VL(2))
ISTATE(2) = DATAN(VL(1)/VL(4))
ISTATE(1) = VL(4)

```

*-----Initialize Initial Mass -----

```

ISTATE(7) = 10000.0D0
WRITE(10,*)'Initial State'
WRITE(10,20)ISTATE

```

*-----Generate Initial Population-----

```

CALL GENERATE(SEED,POP,SIZE,GENE)

```

```

*-----75 Iterations of GA-----

DO ITER = 1,75

*-----Convert From Binary into Decimal Values-----
WRITE(10,*)'Iteration #',ITER
WRITE(11,*)'Iteration #',ITER
WRITE(12,*)'Iteration #',ITER
WRITE(13,*)'Iteration #',ITER
CALL BIN2DEC(POP,GENE,PHI,PSI,TOF,ONOFF)
CALL MAP(POP,PHI,PSI,TOF)
DO I = 1,POP
  WRITE(11,21)PHI(I,1),PHI(I,2),PHI(I,3),PHI(I,4)
  WRITE(11,22)PSI(I,1),PSI(I,2),PSI(I,3),PSI(I,4)
  WRITE(11,*)'OnOff = ',ONOFF(I,1),ONOFF(I,2),ONOFF(I,3),
&      ONOFF(I,4)
  WRITE(11,*)'TOF = ',TOF(I),' days per segment'
ENDDO

*-----Loop through each Population Member-----

DO I = 1,POP

*-----Integrate Each Population Member-----

  DT = TOF(I)*86400.0D0/50.0D0
  DO J = 1,7
    STATE(J) = ISTATE(J)
  ENDDO

*-----Loop Through Each Segment-----
  DO J = 1,4
    DO K = 1,50
      CALL RK4(DT,STATE,PHI(I,J),PSI(I,J),ONOFF(I,J))
    ENDDO
  ENDDO
  WRITE(10,*)'Member #',I
  WRITE(10,20)STATE
  WRITE(12,*)'Member #',I
  CALL FITNESS(STATE,TOF(I),JD,FIT(I),ITER)
  WRITE(10,23)FIT(I)
  WRITE(13,*)FIT(I)

```

ENDDO

*-----Generate next Population Member-----

CALL TOURNAMENT(POP,SIZE,GENE,FIT,SEED)
CALL CROSS(POP,SIZE,GENE,SEED)
CALL MUTATE(POP,SIZE,GENE,SEED)
ENDDO

CLOSE(10)
CLOSE(11)
CLOSE(12)
CLOSE(13)

20 FORMAT('V = ',F8.5,/, 'Gamma = ',F9.6,/, 'Beta = ',F13.10,/, 'r = ',
& F15.4,/, 'Lat = ',F9.6,/, 'Lon = ',F13.10,/, 'Mass = ',F12.6)
21 FORMAT('Phi = ',4(F9.6,3X))
22 FORMAT('Psi = ',4(F9.6,3X))
23 FORMAT('Fitness = ',F20.8)
25 FORMAT('RS = ',4(F12.8,3X))
26 FORMAT('RA = ',4(F12.8,3X))
27 FORMAT('VS = ',4(F11.8,3X))
28 FORMAT('VA = ',4(F11.8,3X))

STOP
END

*-----

SUBROUTINE GENERATE

* Jason C. Eisenreich 21Oct97

* Edited 14Jan98

* This subroutine generates an initial population.

* Variables:

* POP Population size
* SIZE Length of each chromosome
* I Counting variable
* J Counting variable
* RAND Random number
* GENE Matrix containing each chromosome with form

```

*          (chromosome #,bit #)
*   SEED   Integer seed for random number generation
*
* Constants:
*   None
*
* Coupling:
*   RANDOM This subroutine generates a uniformly distributed random
*           number between 0.0 and 1.0
*
* References:
*

```

```

*-----

```

```

SUBROUTINE GENERATE(SEED,POP,SIZE,GENE)

```

```

IMPLICIT NONE

```

```

REAL*8 RAND

```

```

INTEGER POP,I,SIZE,J,GENE(100,100),SEED

```

```

*-----Loop to generate population-----

```

```

DO I = 1,POP
DO J = 1,SIZE
CALL RANDOM(SEED,RAND)
IF (RAND .LE. 0.5) THEN
GENE(I,J) = 0
ELSE
GENE(I,J) = 1
ENDIF
ENDDO
ENDDO

```

```

RETURN

```

```

END

```

```

*-----

```

```

*          SUBROUTINE BIN2DEC

```

```

*

```

```

* Jason C. Eisenreich

```

```

21Oct97

```

```

*

```

```

Edited 14Jan98

```

```

*

```

```

* This subroutine converts a binary number into its appropriate decimal
* equivalent for my low thrust mission. The total chromosome is 39
* bits. The first 7 bits determine the flight path angle for the first
* flight segment. The final 8 bits determine the time of flight with a
* minimum of 90 days and a maximum of 345 days. In between these two
* segments are segments of eight for each other flight segment. Of
* these eight, the first bit determines if the engine is engaged or not
* and the other seven bits determine the flight path angle.

```

```

* Variables:

```

```

*   GENE Matrix of chromosomes
*   I      Current chromosome
*   TEMP   Temporary variable
*   PHI
*   PSI
*   J      Counting variable
*   ONOFF  Matrix of on/off data for engine
*   TOF    Time of flight from one node to next      days

```

```

* Constants:

```

```

* Coupling:

```

```

* References:

```

```

*-----
SUBROUTINE BIN2DEC(POP,GENE,PHI,PSI,TOF,ONOFF)

```

```

IMPLICIT NONE
REAL*8 PHI(100,4),PSI(100,4)
INTEGER I,GENE(100,100),POP,J,ONOFF(100,4),TOF(100)

```

```

*-----Determine flight path angle for each segment-----

```

```

DO I = 1,POP
DO J = 0,3
  PHI(I,J+1) = 8*GENE(I,J*9+1)+4*GENE(I,J*9+2)+2*GENE(I,J*9+3)+
&             GENE(I,J*9+4)
  PSI(I,J+1) = 8*GENE(I,J*9+5)+4*GENE(I,J*9+6)+2*GENE(I,J*9+7)+
&             GENE(I,J*9+8)
ENDDO
ONOFF(I,1) = 1

```

```

DO J = 2,4
  ONOFF(I,J) = GENE(I,9*(J-1))
ENDDO

```

*-----Determine time of flight-----

```

TOF(I) = 16*GENE(I,36)+8*GENE(I,37)+4*GENE(I,38)+2*GENE(I,39)+
&      GENE(I,40)
ENDDO

```

```

RETURN
END

```

*-----

* SUBROUTINE TOURNAMENT

*-----

* Jason C. Eisenreich

21Oct97

* This subroutine uses tournament selection to create the next generation

* Variables:

```

*   I      Counting variable
*   POP    Population size
*   RAND1   Random number
*   RAND2   Random number
*   X      Chromosome number of first compared value
*   Y      Chromosome number of second compared value
*   FIT    Column matrix of fitness values for each chromosome
*   J      Counting variable
*   NEW    Matrix of new generation chromosomes
*   GENE    Matrix of old generation chromosomes

```

* Constants:

*

* Coupling:

*

* References:

*

*-----

SUBROUTINE TOURNAMENT(POP,SIZE,GENE,FIT,SEED)

```

      IMPLICIT NONE
      REAL*8 RAND1,RAND2,FIT(100)
      INTEGER I,J,POP,X,Y,GENE(100,100),SEED,SIZE,NEW(100,100)

```

```

*-----Loop through population size-----

```

```

      DO I = 1,POP
      CALL RANDOM(SEED,RAND1)
      CALL RANDOM(SEED,RAND2)
      X = 1 + INT(RAND1*POP)
      Y = 1 + INT(RAND2*POP)

```

```

*-----Compare fitness values-----

```

```

      IF (FIT(X) .GT. FIT(Y)) THEN
      DO J = 1,SIZE
      NEW(I,J) = GENE(X,J)
      ENDDO
      ELSE
      DO J = 1,SIZE
      NEW(I,J) = GENE(Y,J)
      ENDDO
      ENDIF
      ENDDO

```

```

*-----Fill Gene matrix with new generation-----

```

```

      DO I = 1,POP
      DO J = 1,SIZE
      GENE(I,J) = NEW(I,J)
      ENDDO
      ENDDO

```

```

      RETURN
      END

```

```

*-----

```

```

*                               SUBROUTINE CROSS

```

```

*

```

```

*   Jason C. Eisenreich

```

```

*                               21Oct97

```

```

*

```

* This subroutine does a crossover operation on the new generation. It
 * looks at each pair of chromosomes. To change the probability of
 * crossover, one simply needs to change the value of the variable XPROB.

*

* Variables:

* POP Population size
 * SIZE Length of chromosome
 * I Counting variable
 * XPROB Probability of crossover
 * J Counting variable
 * X Position of crossover
 * GENE Matrix of chromosomes
 * TEMP Temporary matrix

*

* Constants:

*

* Coupling:

*

* References:

*

*-----

SUBROUTINE CROSS(POP,SIZE,GENE,SEED)

IMPLICIT NONE

REAL*8 XPROB,RAND,RAND1

INTEGER I,J,X,POP,GENE(50,20),TEMP(50,20),SEED,SIZE

*-----Declare probability of crossover-----

XPROB = 0.65D0

*-----Loop through each pair-----

DO I = 1,POP-1,2

CALL RANDOM(SEED,RAND)

*-----Check if crossover happens-----

IF (RAND .LE. XPROB) THEN

*-----Generate position of crossover-----

```

CALL RANDOM(SEED,RAND1)
X = 1+INT(RAND1*SIZE)

```

```

*-----Crossover from X to end of chromosome-----

```

```

      DO J = X,SIZE
        TEMP(I,J) = GENE(I+1,J)
        GENE(I+1,J) = GENE(I,J)
        GENE(I,J) = TEMP(I,J)
      ENDDO
    ENDIF
  ENDDO

RETURN
END

```

```

*-----

```

```

*          SUBROUTINE MUTATE

```

```

*

```

```

* This subroutine checks each bit in a population for mutation

```

```

*

```

```

* Jason C. Eisenreich

```

```

10Nov97

```

```

*

```

```

* Variables:

```

```

*   POP

```

```

*   SIZE

```

```

*   GENE

```

```

*   SEED

```

```

*   I

```

```

*   J

```

```

*   MUTPROB

```

```

*   RAND

```

```

*

```

```

* Constants:

```

```

*

```

```

* Coupling:

```

```

*

```

```

* References:

```

```

*

```

```

*-----

```

SUBROUTINE MUTATE(POP,SIZE,GENE,SEED)

IMPLICIT NONE

REAL*8 RAND,MUTPROB

INTEGER POP,SIZE,GENE(100,100),SEED,I,J

MUTPROB = 0.015

DO I = 1,POP

DO J = 1,SIZE

CALL RANDOM(SEED,RAND)

IF (RAND .LE. MUTPROB) THEN

IF (GENE(I,J) .EQ. 0) THEN

GENE(I,J) = 1

ELSE

GENE(I,J) = 0

ENDIF

ENDIF

ENDDO

ENDDO

RETURN

END

*-----

* SUBROUTINE RANDOM

*

* Jason Eisenreich

21 Oct97

*

* This subroutine generates random values between 0.0 and 1.0 using
* an integer seed. This subroutine is taken from "FORTRAN 77 with
* Numerical Methods" by D.M. Etter pg. 306.

*

* Variables:

* SEED The integer seed for the random number

* RAND The random number which is generated

*

* Constants:

* None

*

* Coupling:

* None

```

*
* References:
*   1) Etter, D.M. "FORTRAN 77 with Numerical Methods for
*       Scientists and Engineers". Redwood City, CA: The
*       Benjamin/Cummings Publishing Company; 1992.
*

```

```

-----
SUBROUTINE RANDOM(SEED,RAND)

```

```

    IMPLICIT NONE
    REAL*8 RAND
    INTEGER SEED

```

```

    SEED = 2045*SEED+1
    SEED = SEED - (SEED/1048576)*1048576
    RAND = REAL(SEED+1)/1048576.0D0

```

```

    RETURN
    END

```

```

-----
*
*       SUBROUTINE MAP
*
*
*
-----

```

```

SUBROUTINE MAP(POP,PHI,PSI,TOF)

```

```

    IMPLICIT NONE
    REAL*8 PHI(100,4),PSI(100,4),PI
    INTEGER POP,I,J,TOF(100)

```

```

    DO I = 1,POP
    PI = DACOS(-1.0D0)
    DO J = 1,4
        PHI(I,J) = 10.0D0*PI*PHI(I,J)/14.0D0/180.0D0-5.0D0*PI/180.0D0
        PSI(I,J) = 10.0D0*PI*PSI(I,J)/14.0D0/180.0D0-5.0D0*PI/180.0D0
    ENDDO
    TOF(I) = TOF(I)*2+38
    ENDDO

```

```

    RETURN

```

END

```
*-----
*
*           SUBROUTINE FITNESS
*
* Jason C. Eisenreich                      17Feb98
*
* This subroutine determines the fitness value of a population memeber.
* The fitness function is a weighted combination of the relative
* distance and velocity of the spacecraft and the target.
*
* Variables:
*   STATE      The state matrix after integration
*   JD         The Julian Date at the start of the mission    days
*   JDF        The Julian Date at the end of the mission      days
*   X          The position and velocity of the asteroid at
*              the end of the mission
*   RAST       The position of the asteroid at the end of the
*              mission                                         km
*   VAST       The velocity of the asteroid at the end of
*              the mission                                     km/s
*   RSC        The position of the spacecraft at the end of
*              the mission                                     km
*   VSC        The velocity of the spacecraft at the end of
*              the mission                                     km/s
*   VL         The velocity of the spacecraft in the
*              trajectory tangent frame                       km/s
*   TEMP       Temporary vector
*   MISS       The distance between the spacecraft and the
*              asteroid at the end of the mission            km
*   VREL       The relative velocities of the spacecraft and
*              the asteroid at the end of the mission        km/s
*   FIT        The fitness of the population member
*   TOF        Time of flight                                 days
*   I          Counting variable
*   ITER       The iteration number
*
* Constants:
*
* Coupling:
*
```

* References:

*

*-----

SUBROUTINE FITNESS(STATE,TOF,JD,FIT,ITER)

IMPLICIT NONE

REAL*8 STATE(7),JD,JDF,RAST(4),VAST(4),RSC(4),VSC(4),VL(4)

REAL*8 TEMP(4),MISS,VREL,FIT,X(6)

INTEGER TOF,I,ITER

*-----Determine Position and Velocity of Asteroid-----

JDF = JD+TOF*4.0D0

CALL SOLAR(X,JDF,10)

DO I = 1,3

RAST(I) = X(I)

VAST(I) = X(I+3)

ENDDO

CALL MAG(RAST)

CALL MAG(VAST)

*-----Determine Position and Velocity of Spacecraft-----

RSC(1) = STATE(4)*DCOS(STATE(6))*DCOS(STATE(5))

RSC(2) = STATE(4)*DCOS(STATE(6))*DSIN(STATE(5))

RSC(3) = STATE(4)*DSIN(STATE(6))

CALL MAG(RSC)

*-----Velocity in Locally Level Frame-----

VL(1) = STATE(1)*DSIN(STATE(2))

VL(2) = STATE(1)*DCOS(STATE(2))*DCOS(STATE(3))

VL(3) = STATE(1)*DCOS(STATE(2))*DSIN(STATE(3))

CALL MAG(VL)

*-----Transform into Inertial Frame-----

CALL ROT2(VL,-STATE(6),TEMP)

CALL ROT3(TEMP,-STATE(5),VSC)

DO I = 1,3

RSC(I) = RSC(I) /149597870.0D0

```

        VSC(I) = VSC(I)*86400.0D0/149597870.0D0
        ENDDO
        CALL MAG(RSC)
        CALL MAG(VSC)
        WRITE(12,25)RSC
        WRITE(12,26)RAST
        WRITE(12,27)VSC
        WRITE(12,28)VAST

*-----Determine Miss Distance-----

        MISS = DSQRT((RAST(1)-RSC(1))**2+(RAST(2)-RSC(2))**2+
&      (RAST(3)-RSC(3))**2)

*-----Determine Relative Velocity-----

        VREL = DSQRT((VAST(1)-VSC(1))**2+(VAST(2)-VSC(2))**2+
&      (VAST(3)-VSC(3))**2)

*-----Determine Fitness Value-----

        FIT = 0.01D0/MISS+0.000001/VREL

25  FORMAT('RS = ',4(F12.8,3X))
26  FORMAT('RA = ',4(F12.8,3X))
27  FORMAT('VS = ',4(F11.8,3X))
28  FORMAT('VA = ',4(F11.8,3X))

        RETURN
        END

*-----
*      SUBROUTINE ROT1
*
*  This subroutine performs a rotation about the 1st axis
*
*  Author    : Capt Dave Vallado  USAFA/DFAS  719-472-4109  12Aug88
*
*  Inputs    :

```

```

* Vec      - Input vector
* XVal     - Angle of rotation          rad
*
* Outputs  :
* OutVec   - Vector Result
*
* Locals   :
* c        - Cosine of angle XVal
* s        - Sine of angle XVal
* Temp     - Temporary REAL value
*
* Coupling :
* None.
*
* -----

```

```

SUBROUTINE ROT1(Vec,XVal,OutVec)
  IMPLICIT NONE
  REAL*8 Vec(4),XVal,OutVec(4)

```

```

* ----- Locals -----
  REAL*8 C,S,Temp

```

```

* ----- Implementation -----
  Temp = Vec(3)
  c = DCos(XVal)
  s = DSin(XVal)

```

```

  OutVec(3) = c*Vec(3)-s*Vec(2)
  OutVec(2) = c*Vec(2) + s*Temp
  OutVec(1) = Vec(1)
  OutVec(4) = Vec(4)

```

```

  RETURN
  END

```

```

* -----
* SUBROUTINE ROT2
*
* This subroutine performs a rotation about the 2nd axis
*

```

* Author : Capt Dave Vallado USAFA/DFAS 719-472-4109 12Aug88

*

* Inputs :

* Vec - Input vector

* XVal - Angle of rotation rad

*

* Outputs :

* OutVec - Vector Result

*

* Locals :

* c - Cosine of angle XVal

* s - Sine of angle XVal

* Temp - Temporary REAL value

*

* Coupling :

* None.

*

*-----

SUBROUTINE ROT2(Vec,XVal,OutVec)

IMPLICIT NONE

REAL*8 Vec(4),XVal,OutVec(4)

*----- Locals -----

REAL*8 C,S,Temp

*----- Implementation -----

Temp = Vec(3)

c = DCos(XVal)

s = DSin(XVal)

OutVec(3) = c*Vec(3) + s*Vec(1)

OutVec(1) = c*Vec(1) - s*Temp

OutVec(2) = Vec(2)

OutVec(4) = Vec(4)

RETURN

END

*-----

* SUBROUTINE ROT3

```

*
* This subroutine performs a rotation about the 3rd axis
*
* Author      : Capt Dave Vallado USAFA/DFAS 719-472-4109 12Aug88
*
* Inputs      :
*   Vec        - Input vector
*   XVal        - Angle of rotation                rad
*
* Outputs     :
*   OutVec      - Vector Result
*
* Locals      :
*   c          - Cosine of angle XVal
*   s          - Sine of angle XVal
*   Temp       - Temporary REAL value
*
* Coupling    :
*   None.
*
* -----

```

```

SUBROUTINE ROT3(Vec,XVal,OutVec)
  IMPLICIT NONE
  REAL*8 Vec(4),XVal,OutVec(4)

```

```

* ----- Locals -----
  REAL*8 C,S,Temp

```

```

* ----- Implementation -----
  Temp = Vec(2)
  c = DCos(XVal)
  s = DSin(XVal)

  OutVec(2) = c*Vec(2) - s*Vec(1)
  OutVec(1) = c*Vec(1) + s*Temp
  OutVec(3) = Vec(3)
  OutVec(4) = Vec(4)

```

```

RETURN
END

```

```

*-----
*               SUBROUTINE MAG
*
* This subroutine finds the magnitude of a vector. The tolerance is
* set for 0.000001, thus the 1.0D-12 for a squared test of underflows
*
* Author       : Capt Dave Vallado USAFA/DFAS 719-472-4109 20Sep90
*
* Inputs      :
*   Vec        - Vector
*
* Outputs     :
*   Vec(4)     - Answer stored in fourth component
*
* Locals      :
*   Temp        - Temporary REAL value
*
* Coupling    :
*   None.
*
*-----

```

```

SUBROUTINE MAG(Vec)
  IMPLICIT NONE
  REAL*8 Vec(4)

```

```

*----- Locals -----
  REAL*8 Temp

```

```

*----- Implementation -----
  Temp = Vec(1)**2 + Vec(2)**2 + Vec(3)**2
  IF (DABS(Temp).gt.1.0D-12) THEN
    Vec(4) = DSQRT(Temp)
  ELSE
    Vec(4) = 0.0D0
  ENDIF
  RETURN
END

```

```

*-----

```

```

*           SUBROUTINE RK4
*
* This subroutine is a fourth order Runge-Kutta integrator for a 7
* dimension First Order Differential Equation. This subroutine was
* modified from the A422LIB used at the United States Air Force Academy
* in the Astro 422 class.
*
* Author      : Capt Dave Vallado USAFA/DFAS 719-472-4109 20Sep90
*                               05Aug91
* Edited      : Lt Jason Eiserneich          30Jan98
*
* Inputs:
*  ITIME      - Intial time                sec
*  DT         - Step size                  sec
*  X          - State vector at intitial time
*
* Outputs:
*  X          - State vector at new time
*
* Locals:
*  XDOT       - Derivative of state vector
*  K          - Storage
*  TEMP       - Storage
*  J          - Index
*
* Constants:
*  None.
*
* Coupling
*  DERIV      Subroutines of derivatives of Equations of Motion
*
* References:
*  James, et al., "Numerical Methods" pg. 461-466, eqtn pg 463.
*  BMW pg 414-415
*  A422lib.for
*
*-----
*           SUBROUTINE RK4(DT,X,PHI,PSI,ONOFF)
*
*           IMPLICIT NONE
*           REAL*8 DT,X(7),PHI,PSI

```

INTEGER ONOFF

*-----Locals-----

REAL*8 XDOT(7),K(7,3),TEMP(7)
INTEGER J

*-----Evaluate 1st Taylor Series Term-----

CALL DERIV(X,XDOT,PHI,PSI,ONOFF)

*-----Evaluate 2nd Taylor Series Term-----

DO J = 1,7
K(J,1) = DT*XDOT(J)
TEMP(J) = X(J)+0.5D0*K(J,1)
ENDDO
CALL DERIV(TEMP,XDOT,PHI,PSI,ONOFF)

*-----Evaluate 3rd Taylor Series Term-----

DO J = 1,7
K(J,2) = DT*XDOT(J)
TEMP(J) = X(J) + 0.5D0*K(J,2)
ENDDO
CALL DERIV(TEMP,XDOT,PHI,PSI,ONOFF)

*-----Evaluate 4th Taylor Series Term-----

DO J = 1,7
K(J,3) = DT*XDOT(J)
TEMP(J) = X(J) + K(J,3)
ENDDO
CALL DERIV(TEMP,XDOT,PHI,PSI,ONOFF)

*-----Update the State Vector, Perform Integration-----

DO J = 1,7
X(J) = X(J)+(K(J,1)+2.0D0*(K(J,2)+K(J,3))+DT*XDOT(J))/6.0D0
ENDDO

RETURN
END

```
*-----  
*           SUBROUTINE DERIV  
*  
* This subroutine contains the EOMs for the trajectory tangent  
* coordinate system.  
*  
* Jason C. Eisenreich           30JAN98  
*  
* Variables  
*  
* 0 Constants  
*  
* Coupling  
*  
* References  
*  
*-----
```

SUBROUTINE DERIV(X,XDOT,PHI,PSI,ONOFF)

IMPLICIT NONE
REAL*8 ISP,MDOT,X(7),XDOT(7),PHI,PSI,T(4),K
INTEGER ONOFF

```
*-----Declare Constants-----
```

K = 1.32712438D11

```
*-----Booster Performance-----
```

ISP = 5000.0D0
IF (ONOFF .EQ. 1) THEN
 T(4) = 100*0.00003D0
ELSE
 T(4) = 0.0D0
ENDIF
MDOT = -T(4)/ISP/0.009807D0

```
*-----Determine Thrust Components-----
```

```

T(1) = T(4)*DSIN(PHI)
T(2) = T(4)*DCOS(PHI)*DCOS(PSI)
T(3) = T(4)*DCOS(PHI)*DSIN(PSI)

```

*-----Initialize Derivatives-----

```

XDOT(1)=(-k*DSIN(X(2))/X(4)**2+T(2)/X(7))
XDOT(2)=(X(1)*DCOS(X(2))/X(4)-k*
&      DCOS(X(2))/(X(1)*X(4)**2)+T(1)/(X(7)
&      *X(1)))
XDOT(3)=-(X(1)*DCOS(X(2))/X(4))*DCOS
&      (X(3))*DTAN(X(6))+T(3)/(X(7)*X(1)*DCOS(X(2)))
XDOT(4)=X(1)*DSIN(X(2))
XDOT(5)=(X(1)*DCOS(X(2))*DCOS(X(3))/(X(4)
&      *DCOS(X(6))))
XDOT(6)=X(1)*(DCOS(X(2))*DSIN(X(3))/X(4))
XDOT(7)=MDOT

```

```

RETURN
END

```

REFERENCES

- Ashley, Holt, 1974. Engineering Analysis of Flight Vehicles, New York: Dover Publications, Inc.
- Bryson, Arthur E., and Yu-Chi Ho, 1975. Applied Optimal Control, New York: Hemisphere Publishing Corporation.
- Goldberg, David E. 1989a. Genetic Algorithms in Search, Optimization & Machine Learning, Reading, Ma.: Addison-Wesley Publishing Company, Inc.
- Hamilton, Calvin J. 1997. "Halley's Comet", <http://www.hawast.soc.org/solar/eng/halley.htm>.
- Humble, Ronald W., Gary N. Henry, and Wiley J. Larson, Space Propulsion Analysis and Design, 1995. New York: McGraw-Hill, Inc.
- Piñon, Elfego III, 1995. An Investigation of the Applicability of Genetic Algorithms to Spacecraft Trajectory Optimization, Ph.D. diss., The University of Texas at Austin.
- Rauwolf, Gerald, 1995 Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm, M.S. Thesis, The University of Illinois at Urbana-Champaign.
- Sellers, Jerry Jon. 1994. Understanding Space: An Introduction to Astronautics, New York: McGraw Hill, Inc.

VITA

Jason Corey Eisenreich was born in Gordon, Nebraska, on April 26, 1974, the son of Joe C. and Linda J. Eisenreich. After graduation from Gordon High School in Gordon, Nebraska, in May 1993, he entered the United States Air Force Academy. After graduation with a Bachelor of Science degree in Space Operations and commissioning as a 2nd Lieutenant in the United States Air Force, in May 1998, he enrolled in the graduate school of The University of Texas at Austin. In July 1997, he married the former Kody Lanae Benson of Gordon, Nebraska. Jason is a member of AIAA. After completion of his master's degree he will be stationed at Los Angeles Air Force Base in El Segundo, California.

Permanent Address: 707 N. Main St.

Gordon, NE 69343

This thesis was typed by the author.