

AEROSPACE DESIGN OPTIMIZATION USING A REAL CODED GENETIC
ALGORITHM

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

John David Dyer

Certificate of Approval:

John Burkhalter
Professor Emeritus
Aerospace Engineering

Roy J. Hartfield, Chair
Professor
Aerospace Engineering

Robert Gross
Associate Professor
Aerospace Engineering

Brian Thurow
Assistant Professor
Aerospace Engineering

Joe F. Pittman
Interim Dean
Graduate School

AEROSPACE DESIGN OPTIMIZATION USING A REAL CODED GENETIC
ALGORITHM

John David Dyer

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 10th, 2008

AEROSPACE DESIGN OPTIMIZATION USING A REAL CODED GENETIC
ALGORITHM

John David Dyer

Permission is granted to Auburn University to make copies of this thesis at its discretion,
upon request of individuals or institutions and at their expense. The author reserves all
publication rights

Signature of Author

Date of Graduation

VITA

John David Dyer was born on July 13, 1983, in New Orleans, Louisiana to Mark and Cetta Dyer. He began attending Auburn University after graduating from De La Salle High School in 2001. While at college, he was a member of Tau Beta Pi engineering society, and was a co-op for Chromalloy Florida in Ft. Walton Beach, Florida. John graduated Cum Laude in May 2006 with a Bachelor of Aerospace Engineering degree. He began graduate school at Auburn University where he pursued a master's degree in aerospace engineering.

ABSTRACT

AEROSPACE DESIGN OPTIMIZATION USING A REAL CODED GENETIC
ALGORITHM

John David Dyer

Master of Science, May 10, 2008
(B.A.E., Auburn University, 2006)

114 Typed Pages

Directed by Roy J. Hartfield

This study demonstrates the advantages of using a real coded genetic algorithm (GA) for aerospace engineering design applications. A real coded GA was written from first principles for this study and the source code can be seen in Appendix A. The GA runs steady state, meaning that after every function evaluation a tournament scheme determines the worst performer and that worst performer is then thrown out and replaced by a new member that has been evaluated. The new member is produced by mating two successful parents through a crossover routine, and then mutating that new member. For this study three different preliminary design studies were conducted using both a binary and a real coded GA including a Single Stage Solid Propellant Missile Systems Design, a Two Stage Solid Propellant Missile Systems Design and a Single Stage Liquid Propellant Missile Systems Design.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Roy Hartfield and Dr. Gerry Dozier for their guidance with this thesis as well as Dr. John Burkhalter for his help with technical missile matters. The author also wishes to thank Dr. Murray Anderson, the author of the IMPROVE 3.1 Genetic Algorithm. The author would also like to thank his friends, family, and loving fiancé for being patient with him as he worked to complete this thesis.

Style manual of journal used:

The American Institute of Aeronautics and Astronautics Journal

Computer software used:

Improve 3.1 Genetic Algorithm, Tecplot 360, Compaq Visual Fortran Compiler,

Microsoft Excel, Microsoft Word

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES.....	x
NOMENCLATURE.....	xii
1 INTRODUCTION.....	1
2 REAL CODED GA METHODOLOGY.....	5
2.1 CROSSOVER.....	6
2.2 MUTATION.....	8
2.3 STEADY STATE GA.....	10
2.4 VARIABLE MUTATION	13
3 MISSILE SYSTEM DESCRIPTIONS	14
4 REAL GA CONVERGENCE TESTING	25
5 MISSILE SYSTEM DESIGNS GA COMPARISONS	31
5.1 SINGLE STAGE SOLID MISSILE SYSTEM DESIGN.....	31
5.1.1 SINGLE STAGE SOLID 100,000 FT 2,500 LB GOAL	35
5.1.2 SINGLE STAGE SOLID 350,000 FT 4,000 LB GOAL	39
5.1.3 SINGLE STAGE SOLID 2,500 LB 240 SEC TOF GOAL.....	45
5.2 TWO STAGE SOLID MISSILE SYSTEM DESIGN.....	50
5.2.1 TWO STAGE SOLID 100,000 FT 6,000 LB GOAL	51
5.2.2 TWO STAGE SOLID 250,000 FT 7,000 LB GOAL	57
5.3 SINGLE STAGE LIQUID MISSILE SYSTEM DESIGN.....	63
5.3.1 SINGLE STAGE LIQUID 450K FT 55K LB WEIGHT GOAL ...	64
5.3.2 SINGLE STAGE LIQUID 700K FT 55K LB WEIGHT GOAL ...	68
6 REAL CODED GA CONVERGENCE TESTING REVISITED.....	74
7 CONCLUSIONS AND RECOMMENDATIONS	80
REFERENCES.....	83
APPENDIX A: Real Coded GA Source File.....	86
APPENDIX B: Single Stage Solid Real GA Input File.....	94
APPENDIX C: Single Stage Solid Binary GA Input File	95
APPENDIX D: Two Stage Solid Real GA Input File.....	97
APPENDIX E: Two Stage Solid Binary GA Input File	99
APPENDIX F: Single Stage Liquid Real GA Input File.....	101
APPENDIX G: Single Stage Liquid Binary GA Input File.....	102

LIST OF TABLES

Table 1. General Form of the initial constants input file for the Missile Design Codes	15
Table 2. Single Stage Solid GA Variable Definition	17
Table 3. Two Stage Solid GA Variable Definition	18
Table 4. Single Stage Liquid GA Variable Definition	22
Table 5. Mutation Operator Fitness Comparison	27
Table 6. Population Testing Fitness Comparison	28
Table 7. Single Stage Solid Optimization Results	35
Table 8. Single stage solid 100,000 ft 2,500 lb Final Design Variables	36
Table 9. Single stage solid 350,000 ft 4,000 lb Final Design Variables	40
Table 10. Single stage solid 2,500 lb 240 sec Final Design Variables	45
Table 11. Two Stage Solid Optimization Results	50
Table 12. Two stage solid 100,000 ft 6,000 lb Final Design Variables	51
Table 13. Two stage solid 250,000 ft 7,000 lb Final Design Variables	57
Table 14. Single Stage Liquid Optimization Results	63
Table 15. Single Stage Liquid 450,000 ft 55,000 lb Final Design Variables	64
Table 16. Single Stage Liquid 700,000 ft 55,000 lb Final Design Variables	69
Table 17. Two Goal Mutation Testing Results	75
Table 18. One and Two goal GA Parameter Comparison	77
Table 19. Overall Final Fitness Comparison	81

LIST OF FIGURES

Figure 1. Tournament Selection Mutation and Crossover Flow Chart	6
Figure 2. Blend X Crossover.....	7
Figure 3. Single-Point Crossover	7
Figure 4. Uniform Crossover	8
Figure 5. Real Coded Steady State GA Program Flow	10
Figure 6. Steady State Detailed Flow Chart.....	12
Figure 7. Missile System Design Code Program Flow	16
Figure 8. Solid Propellant Grain Cross-Section	20
Figure 9. Nose Design Schematic	20
Figure 10. Nozzle Design Schematic	21
Figure 11. Tail Design Schematic	21
Figure 12. Liquid Missile Design Schematic	24
Figure 13. 1 Stage Solid Missile Crossover Testing 400,000 ft (Single) Goal	26
Figure 14. Single Stage Solid Missile Mutation Testing 400,000 ft Goal	27
Figure 15. Single Stage Solid Missile Population Testing 400,000 ft Goal	28
Figure 16. 1 Stage Solid Variable Mutation Convergence Plot	30
Figure 17. Single Stage Solid 700,000 ft Preliminary Test Convergence History ..	32
Figure 18. Single Stage Solid 250,000 ft Preliminary Test Convergence History ..	33
Figure 19. Single stage solid 100,000 ft 2,500 lb convergence	37
Figure 20. Single Stage Solid Missile Body Diameter Convergence History	38
Figure 21. Single Stage Solid 3D Model-Real GA	39
Figure 22. Single Stage Solid 3D Model-Binary GA	39
Figure 23. Single stage solid 350,000 ft 4,000 lb convergence	41
Figure 24. Single Stage Solid Missile Body Diameter Convergence History	42
Figure 25. Single Stage Solid Missile Initial Launch Angle Convergence History.	43
Figure 26. Single Stage Solid 3D Model-Real GA	44
Figure 27. Single Stage Solid 3D Model-Binary GA	44
Figure 28. Single stage solid 2,500 lb 240sec convergence history.....	46
Figure 29. Single Stage Solid Missile Body Diameter Convergence History	47
Figure 30. Single Stage Solid Missile Initial Launch Angle Convergence History.	48
Figure 31. Single Stage Solid 3D Model-Real GA	49
Figure 32. Single Stage Solid 3D Model-Binary GA	49
Figure 33. Two stage solid 100,000 ft 6,000 lb convergence history	53
Figure 34. Two Stage Solid Missile Upper Body Diameter Convergence History .	53
Figure 35. Two Stage Solid Missile Initial Launch Angle Convergence History....	55
Figure 36. Two Stage Solid 3D Model-Real GA	56
Figure 37. Two Stage Solid 3D Model-Binary GA	56
Figure 38. Two stage solid 250,000 ft 7,000 lb convergence history	59
Figure 39. Two Stage Solid Missile Upper Stage Diameter Convergence History .	60

Figure 40. Two Stage Solid Missile Initial Launch Angle Convergence History....	61
Figure 41. Two Stage Solid 3D Model-Real GA	62
Figure 42. Two Stage Solid 3D Model-Binary GA	62
Figure 43. Single Stage Liquid 450,000 ft 55,000 lb Convergence Plot	65
Figure 44. Single Stage Liquid Missile Body Diameter Convergence History	66
Figure 45. Single Stage Liquid Missile Tail Leading Edge Angle Convergence History	67
Figure 46. Single Stage Liquid 3D Model-Real GA.....	68
Figure 47. Single Stage Liquid 3D Model-Binary GA	68
Figure 48. Single Stage Liquid 700,000 ft 55,000 lb Convergence Plot	70
Figure 49. Single Stage Liquid Missile Body Diameter Convergence History	71
Figure 50. Single Stage Liquid Missile Burn Time Convergence History	72
Figure 51. Single Stage Liquid 3D Model-Real GA.....	73
Figure 52. Single Stage Liquid 3D Model-Binary GA	73
Figure 53. Two Goal Crossover Convergence History	74
Figure 54. Two Goal Mutation Testing Convergence History	75
Figure 55. Two Goal Population Test Convergence History	76
Figure 56. Two Stage Solid Match 100k ft 6k lb Re-Run.....	78
Figure 57. Two Stage Solid Match 250k ft 7k lb Re-Run.....	78

NOMENCLATURE

μ	Mutation Rate
σ	Mutation Amount
Ae	Nozzle Exit Area
A*	Nozzle Throat Area
b2tail	Tail Semi-Span
b2wing	Wing Semi-Span
crtail	Tail Root Chord
crwing	Wing Root Chord
dbody	Diameter of Body
dnose	Nose Diameter
dstar	Throat Diameter
eps	Epsilon-Grain
f	Propellant Grain Fillet Radius
GA	Genetic Algorithm
lbody	Length of Body
LE	Leading Edge
lnose	Length of Nose
N	Number of Star Points-Grain
rbody	Radius of Body
Ri	Propellant Inner Grain Radius
rnose	Radius of Nose
Rp	Propellant Outer Grain Radius
TE	Trailing Edge
TOF	Time of Flight
xLEt	X-Location of Tail Leading Edge
xLEw	X-Location of Wing Leading Edge

1 INTRODUCTION

Optimization of aerospace engineering applications using GA's such as spacecraft controls^{1,2}, turbines³, helicopter controls⁴, flight trajectories⁵, wings and airfoils^{6,7}, missiles^{8,9,12,11}, rockets^{12,13,14}, propellers¹⁵ and inlets¹⁶ have been performed with great success. In some cases, real coded genetic algorithm's have been shown to produce better results than binary coded GA's^{17,18,19}. This success is the primary motivation for the current study involving aerospace applications.

The goal of engineering design optimization is to find an optimum solution to a design problem. Optimization has evolved throughout the years from classical methods to modern evolutionary algorithms and modern computers with their exceedingly fast computational times have enabled optimizers to become extremely efficient at solving very complex problems.

The first mathematical optimization methods were developed by Newton. Newton formulated that by taking the first derivative of a function and setting it equal to zero one could find the local maxima or minima of a piecewise continuous, finite solution. Later Siddal²⁰ showed that when using multiple functions to describe a system, Newton's method produces multiple non-linear equations that must be solved simultaneously. Because of the complexity of solving a system of non-linear equations other optimization methods were developed. The next major step in optimization came in the 1800's with the gradient methods. Gradient decent (or ascent) works on the method of taking steps proportional to the negative (or positive for ascent) of the gradient of the

function. Steps continue until a local minimum has been reached. The main problem with the gradient methods is they are very susceptible to converging to local minima or maxima instead of converging to a global minima or maxima.

The next breakthrough in optimization came in the mid 1900's from George B. Dantzig with linear programming²¹. Linear programming works on the principle of optimizing a linear objective function with a set of linear constraints. The optimization is set up to either maximize or minimize the objective function. The simplex method developed by Dantzig works in polyhedron space by constructing a solution on the vertex of the polyhedron and then walking along the edges of the polyhedron until the optimal solution is reached. The algorithm is quite efficient and in practice can be set up to only converge to global optimums if the proper constraints are used. This method does not work for complex multi-variable problems because the constraints and objective function must be linear. Nonlinear programming addresses this issue.

Genetic algorithms in theory can overcome many of the issues that the early optimization schemes encountered. All GA's are based on the principles developed by John Holland in his book "Adaptation in Natural and Artificial Systems"²². Holland outlined the methods for successfully implementing population based adaptive optimizers. Holland's methods operate on the principle of survival of the fittest. In a computational sense, candidate solutions are assembled in a population and compared to one another, the weak die off and the strong are left to reproduce and mutate to produce better children.

The search for a more efficient optimizer for some of these aerospace applications arose due to the extended run times associated with long range missiles when solved

using binary coded genetic algorithms, such as IMPROVE[®] (Implicit Multi-objective PaRameter Optimization Via Evolution)²³. The IMPROVE[®] software was shown to be a very versatile and robust method for optimization. A primary disadvantage of the binary coded GA comes from the fact that because all of the variables must be converted into a single bit string, the solution accuracy is dependant on the number of bits that can be used for the string.

Because of the large ranges associated with many on the genes the smallest resolution was limited to 0.01 while the real coded GA is only limited to a double precision number. The two stage solid system used in this study has 46 genes making up each individual. Because of this large number of genes, the resolution of each gene is limited in order to reduce the size of the bit string. This translates into a reduced level of accuracy because each gene is limited to a specific number of significant digits because of the nature of the binary GA. Resolution is not a significant issue with a real coded GA because all of the variables remain real double precision variables. Dozier^{24,25,26} demonstrated the ability for a real coded GA to achieve shorter run times as well as more accurate solutions for some applications. Another problem associated with the binary GA is hamming cliffs. Hamming cliffs can also pose a problem for the binary GA because all of the design parameters are converted into a single bit string. For example if two integers 15, and 16 were represented by the bit strings 10000 and 01111 respectively, the GA would have to change all of the bits simultaneously to change from 15 to 16. Mutation and crossover do not always solve this problem. Hamming cliffs are not possible with the real GA because the design variables remain real coded.

Another advantage of the real GA created for this study is that it uses steady state optimization unlike the generational optimization used by the binary GA. The key difference is that in the steady state GA for each generation only the worst performer is thrown out and replaced by a new member, whereas for the generational GA all of the members of the population are thrown out and replaced (except in elitist mode when the best member remains in the next generation) using a similar tournament routine. For the steady state GA, once the survivors have had a chance to crossover (i.e. pass genetic material back and forth through their variables), the new member replacing the worst performer is run back through the objective function. This process continues, with the parents producing on average better offspring, until the maximum number of generations (user specified) is reached. There are proofs^{27,28} which show why this process produces increasingly superior performers in a population, but a simplistic view is that a good parent mated with another good parent, is more likely to produce good offspring than two poor parents when mated. This is not to say that two good parents cannot produce poor performers. Rather, when two good performers exchange genes, statistically the resulting offspring have a higher chance of outperforming their parents.

2 REAL CODED GA METHODOLOGY

The real and binary GA's used in this study both operated using the same tournament style evolution of a population. They each work with a number of candidate solutions to solve a particular problem. A data structure known as an individual is used to represent each candidate solution. Each individual has a fitness and a chromosome. Each chromosome is made up of genes, in this case the genes used are the GA variables in the design variable input (GANNL.DAT) files associated with each code used. (see Appendix B through G) A group of individuals makes up a population. In order to create a new population, individuals called parents are selected based on their fitness and allowed to create children using a tournament selection process. Details of the binary GA operation can be found in references 20 and 27. For the real coded GA, parents are chosen in groups of two and the parent with the better fitness survives to produce children. This process is repeated to select a second parent. The tournament selection process is shown in Figure 1. Both parents then use a crossover routine to mix their genes in order to produce a child.

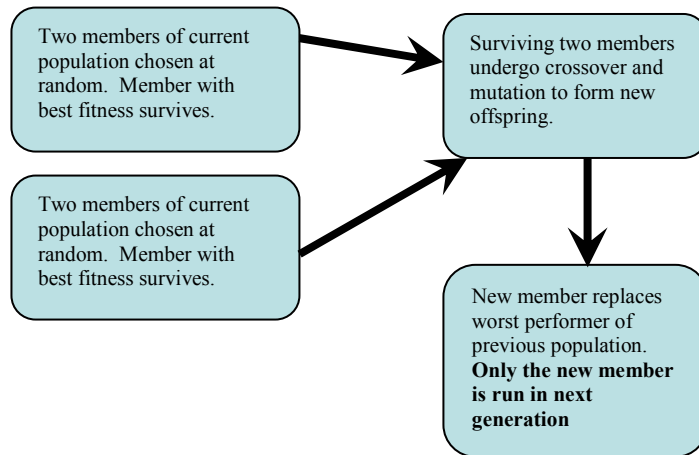


Figure 1. Tournament Selection Mutation and Crossover Flow Chart

2.1 CROSSOVER

Three different crossover routines were used for the real coded GA, Blend X, Uniform and Singlepoint crossover. For Blend X²⁹ crossover each design parameter of the two parents selected is subtracted and then multiplied by a random number (between 0 and 1), and added to the smaller parent as shown below. Blend X crossover allows for the most mutation of all the crossover routines. Blend X crossover creates children unique from their parents, unlike uniform or single-point crossover, whose children are the same values from either one parent or the other. It should be noted while Blend X crossover does produce a unique child from the two parents, the child cannot be outside the range of the values of the parents, and for this reason mutation is necessary.

For $i = 1$ to number of design parameters:

$$Child(i) = abs(Parent1(i) - Parent2(i)) \times Random\# + \min(Parent1(i) - Parent2(i))$$

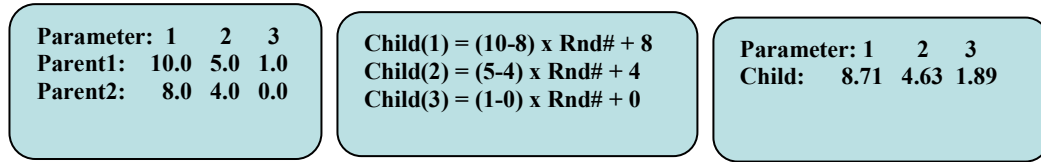


Figure 2. Blend X Crossover

Both Single-point and Uniform crossover swap design parameters between parents. For Single-point crossover a random number is used to define a cut point where in the first parents design parameters are used up to that cut point, and the second parents design parameters are used after that cut point, as shown in Figure 3.

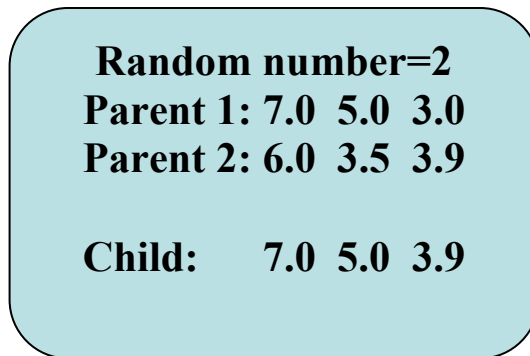


Figure 3. Single-Point Crossover

Uniform crossover works very similarly to single-point crossover except each design parameter in the child is randomly chosen from either parent 1 or parent 2, this allows a child to be made up of any combination of both parents design parameters as shown in Figure 4.

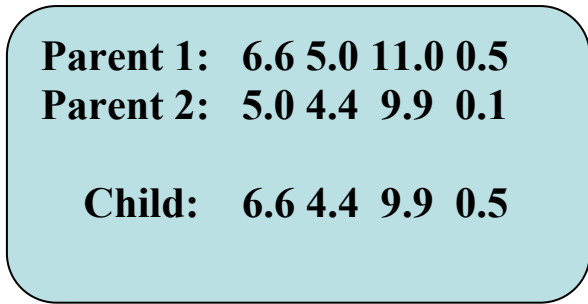


Figure 4. Uniform Crossover

2.2 MUTATION

After a child has been created it can be mutated using a Gaussian mutation routine. Two main operators control the mutation, mutation amount and mutation rate. Mutation rate, μ determines how many genes of each child will be mutated, this operator is set between 0.0 and 1.0, with 1.0 mutating every gene within the child. The second operator used in mutation is the mutation amount σ . The mutation amount determines how much each gene will be mutated, this usually ranges from 0.5 (high mutation) to 0.005 (very low mutation). A Gaussian distributed random number is used with the mutation amount in order to mutate the child as shown below.

For i = 1 to number of genes:

$$Child(i) = \sigma * [x_{max}(i) - x_{min}(i)] * gaussian_random_number + Child(i)$$

Where $x_{\max}(i)$ and $x_{\min}(i)$ are the maximum and minimum values specified by the user for each gene in the GA variable input file (GANNL.DAT). If any of the mutated child's design parameters are larger than the maximum or smaller than the minimum value then that design parameter is set to the maximum or the minimum value. It has been shown that mutating with a Gaussian distributed random number with a zero mean and unit variance works well for optimizations³⁰.

After the parents design parameters have been crossed and mutated to produce a new child or individual, that individual is evaluated and given a fitness. In order for the new individual to become a member of the population, one member must die in order for the new member to take its place. The member that dies off is the member in the population with the worst fitness. This method of allowing individuals to create new children while also killing off members with bad fitness' is known as natural selection³¹. This process of creating new children and killing off members with bad fitness' continues on until a set number of iterations have been satisfied. The entire program flow for the real coded GA is shown in Figure 5.

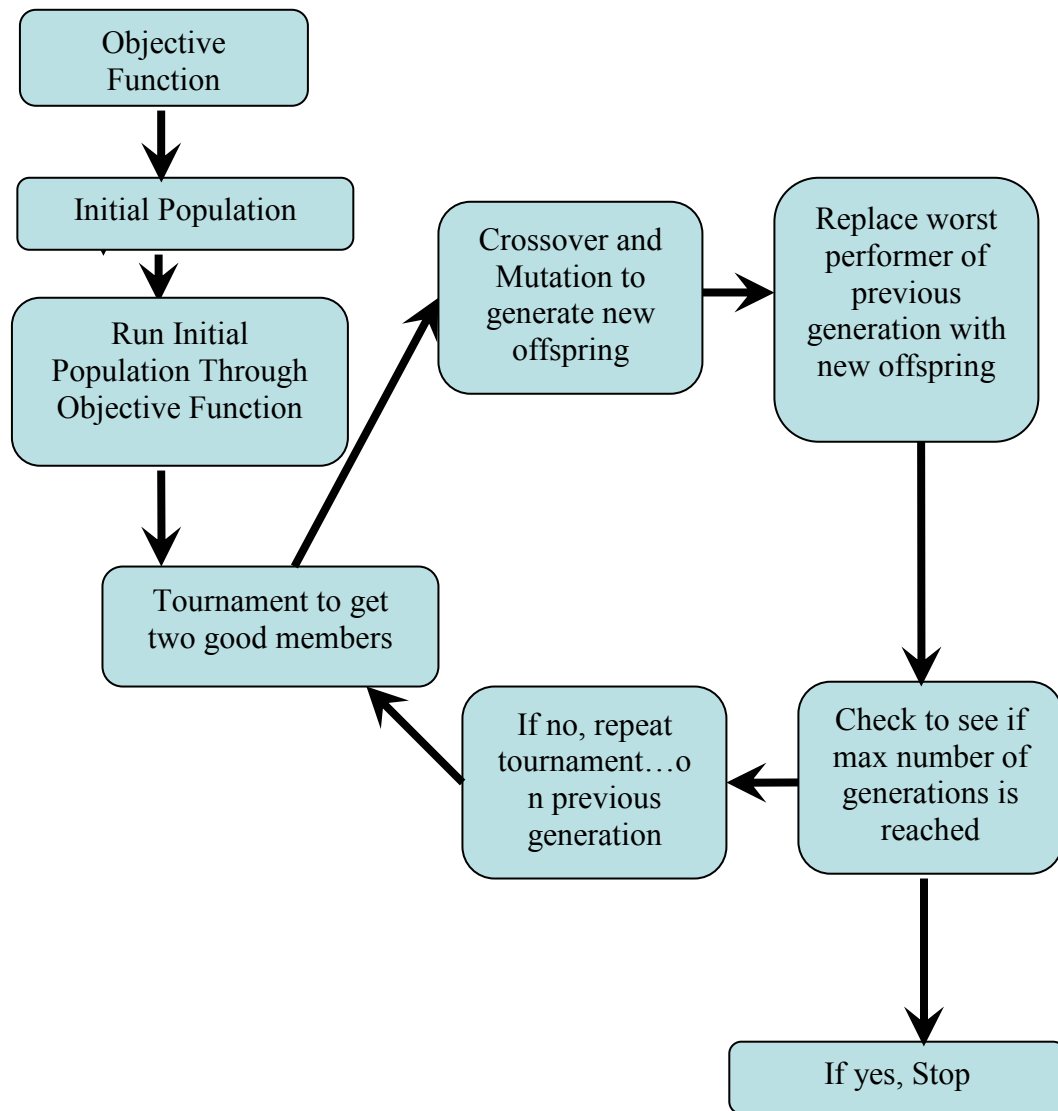


Figure 5. Real Coded Steady State GA Program Flow

2.3 STEADY STATE GA

The real GA developed for this study operates in steady state as opposed to generational as the binary GA. The main difference between steady state and generational is the number of members after each generation that are evaluated by the objective function. For a generational GA all of the members of a population, usually at least 50, are evaluated by the objective function. Their fitnesses are then evaluated and

sorted and then a tournament scheme replaces each member of the population by crossover and mutation, and all of the new members are evaluated by the objective function, except for the best performer.

For a steady state GA after the initial population is evaluated by the objective function, only one member is replaced by a tournament scheme using crossover and mutation. That new member is evaluated by the objective function and then replaces the previous generation's worst performer. This can lead to quick and very accurate convergence due to the fact that the member immediately becomes part of the mating pool making a shift toward an optimal fitness possible early in the optimization³¹. A flow chart showing the steady state process in detail is shown in Figure 6. The main drawback to steady state is that it does not have the large number of random guesses that the generational GA can obtain. Since each member that is created is composed of 2 good members from the previous generation the steady state GA can quickly converge to a good solution, however if none of the members of the initial population are good members the steady state GA can only rely on mutation of the one member each generation to find a good fitness. For this reason BlendX crossover was included to enable more mutation for each new member.

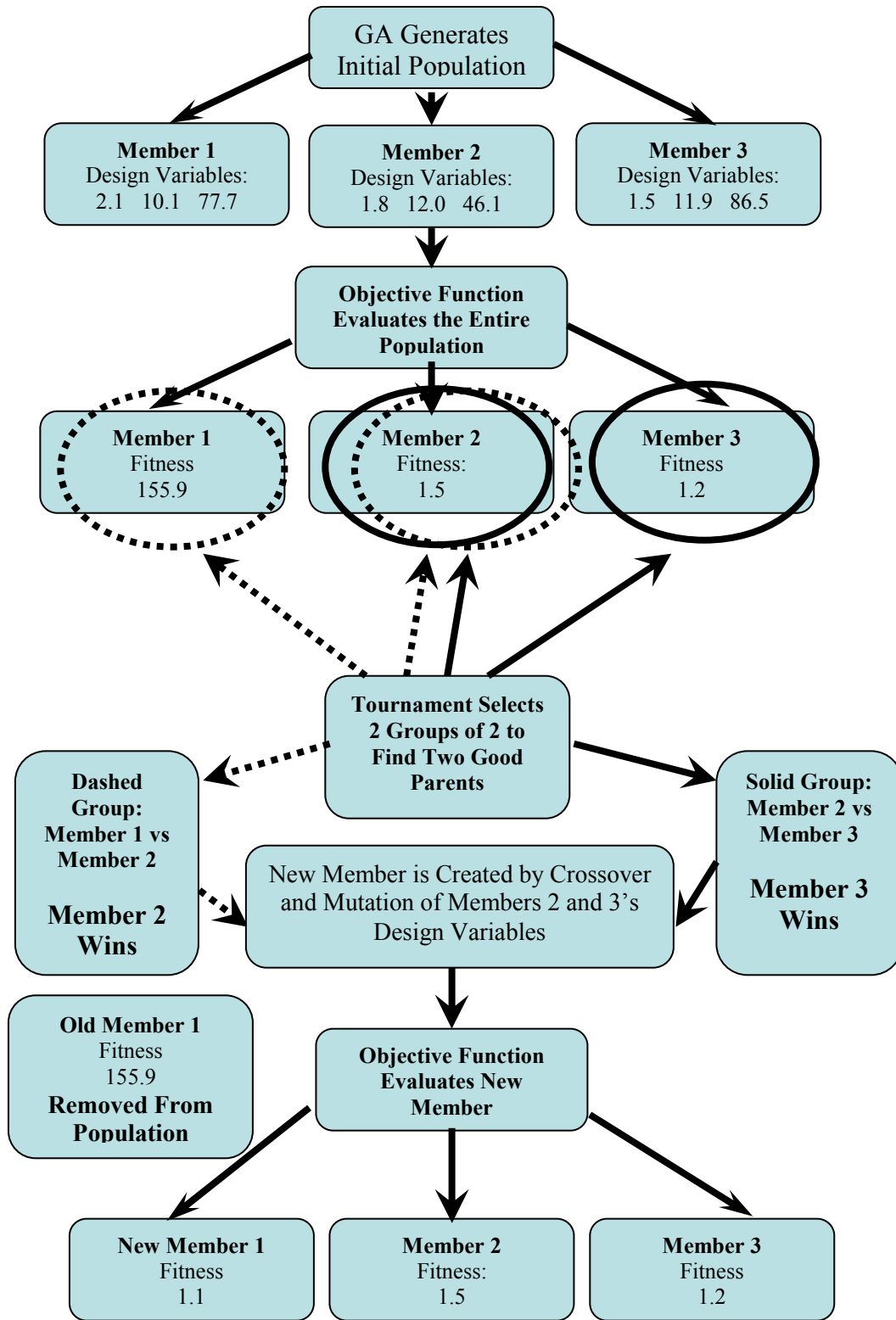


Figure 6. Steady State Detailed Flow Chart

2.4 VARIABLE MUTATION

Variable mutation is used to obtain quick convergence for a given function. The mutation rate μ is allowed to vary throughout the GA's run. The code is set up to keep track of how many individuals were better than the previous population. After a specified number of function evaluations, if the number of successful mutations is less than $1/5^{\text{th}}$ of the total number of mutations then the mutation rate is reduced by 20%, similarly if more than $1/5^{\text{th}}$ of the individuals are better then the mutation rate is increased by 20%. This $1/5^{\text{th}}$ rule was pioneered by Rechenberg³² and has been shown to provide quick convergence. Initial testing of variable mutation demonstrated the ability of quick convergence, however in many instances a local minimum was achieved.

3 MISSILE SYSTEM DESCRIPTIONS

Three different missile system design codes were used in this study to compare the real and binary GA's. The missile system design code¹⁷ creates and flies preliminary design level models for missiles powered by one or two stage solid propellant rocket propulsion or liquid rocket propulsion using a set of approximately 26-46 critical design variables. The GA is used to optimize the missile to meet certain goals. Burhalter, Jenkins, Hartfield, Anderson and Sanders have developed the programs used to design the physical model of the missile that they described in their paper "Missile Systems Design Optimization Using Genetic Algorithms"¹². Further research has been conducted on missile design optimization by Burhalter, Jenkins and Hartfield in their paper, "Optimizing a Solid Rocket Motor Boosted Ramjet Powered Missile Using a Genetic Algorithm"¹⁴. The missile design codes have been shown to be a very good tool for preliminary design and have been extensively used as a design tool.

All three missile system design codes follow a similar program flow. In the main program, the first task is to read in two input files that contain constants including densities masses and moments of inertia. The block of information in these files is divided into major sections as listed in Table 1. The number listed to the right is the number of variables included in each of the major sections. Initially, the table must be generated with known values for each of the variables.

Table 1. General Form of the initial constants input file for the Missile Design Codes

1. densities 30
2. masses 30
3. center of gravity 27
4. moments of inertia 60
5. lengths and limits 30
6. external geometry 30
7. required computed data from aero 30
8. other dimensions 11
9. internal solid rocket grain variables 12
10. nozzle and throat variables 14
11. other computed stage variables 8
12. program lengths, limits and constants 10
13. initiation of launch data 14
14. target data 6
15. GA goals (outdata variables) 20
16. auxillary variables to be used as needed 10
17. list of GA variables passed to setup, etc. 29
18. Total missile variables 10

After the initial constants input files are read in, the GA input file is read in, which contains the GA variables to be optimized. For each member of a population a new missile is built using the data from the initial constants input file and the design parameters that the GA generates using the GA input file. After the initial parameters are stored, the program designs and flies the missile using a series of subroutines to determine the propulsion, mass properties and aerodynamics. This missile is then flown in the 6-DOF routine as shown in Figure 7.

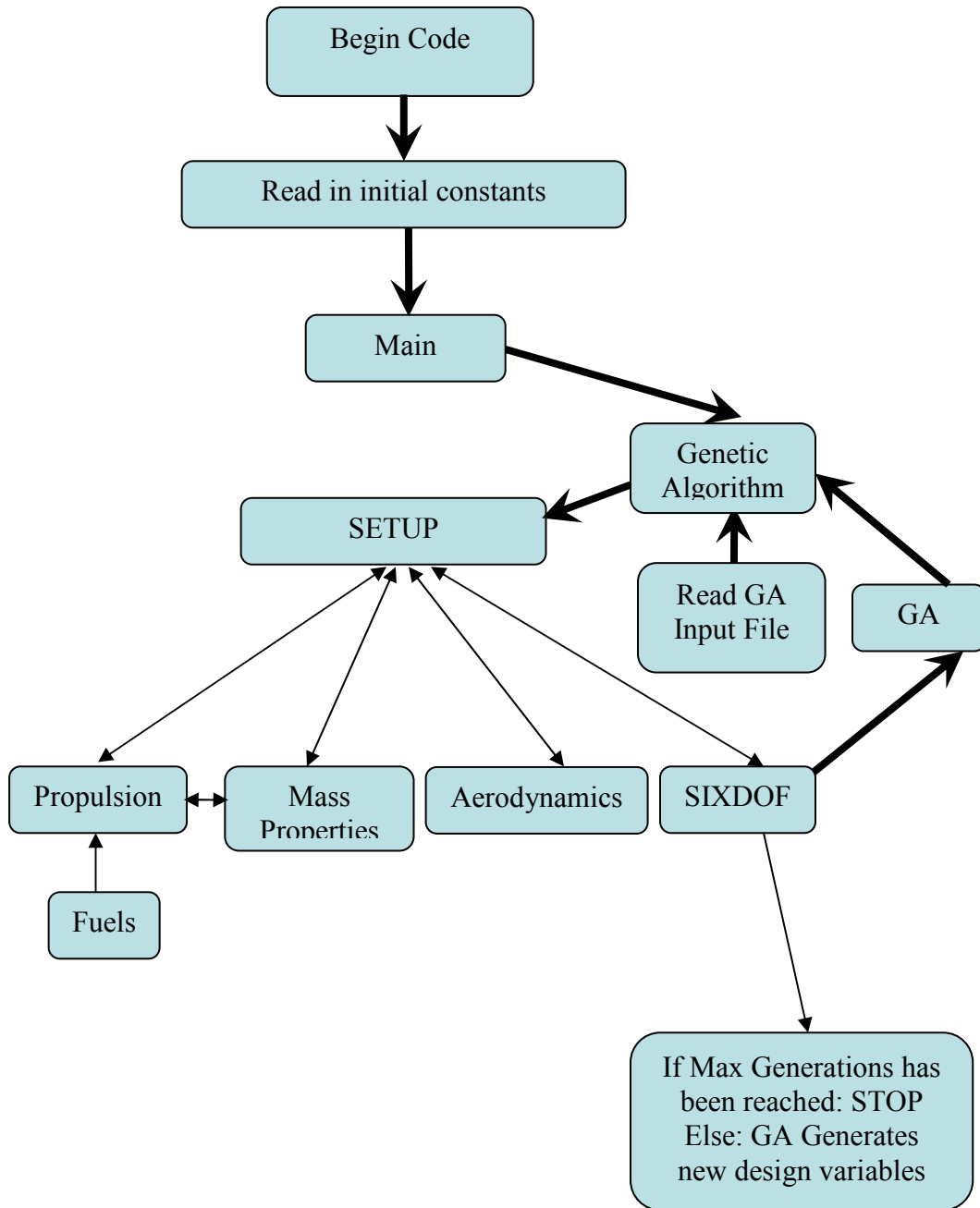


Figure 7. Missile System Design Code Program Flow

The first system tested was the single stage solid missile design. The single stage solid missile design uses 29 design variables in order to produce a physical model of the system. An overview of the 29 design variables is shown in Table 2.

Table 2. Single Stage Solid GA Variable Definition

Geometry	Propellant	Auto Pilot
1.nose radius ratio = r_{nose}/r_{body}	3. fuel type	24. auto pilot on delay time
2.nose length ratio = L_{nose}/d_{body}	4. propellant out rad $rpvar=(rp+f)/r_{body}$	25. auto pilot time constant-tau, f.c.s. time const
10. fractional nozzle length ratio = f/rp	5.propellant inner radius ratio= ri/rp	26. auto pilot damping -zeta, f.c.s. damping
11. throat diameter / D_{body}	6. number of star points	27. cross over freq - w_{cr} , f.c.s. crossover freq
12. total length of stage1/ D_{body}	7. fillet radius ratio= f/rp	28. pronav gain - n-prime, guidance gain
13. dia of stage1 center section d_{body} (in)	8. epsilon - star width	
14.wing exposed semi-span = b_{2w}/d_{body}	9. star point angle	
15. wing root chord = crw/d_{body}		
16. wing taper ratio = ctw/crw		
17. LE sweep angle deg		
18. $xLEw/l_{body}$		
19. tail exposed semi-span = b_{2t}/d_{body}		
20. tail root chord = crt/d_{body}		
21. tail taper ratio = ctt/crw		
22. LE sweep angle deg		
23. $xTEt/l_{body}$		
29. Initial Launch Angle (deg)		

These 29 GA variables in conjunction with the initial constants in the constants input file are used to design and fly the missile through the 6-DOF. The performance is then stored and used to compute the overall fitness depending on the goal chosen for the optimization.

The second system tested was the two stage solid propellant missile design. The two stage solid missile design operates in a similar fashion to the single stage solid code, however it uses 46 design variables in order to produce a physical model of the system instead of the 29 used in the single stage solid code. An overview of the 46 design variables is shown in Table 3.

Table 3. Two Stage Solid GA Variable Definition

Geometry (Upper Stage)	Propellant (Upper Stage)	Auto Pilot (Upper Stage)
1.nose radius ratio = r_{nose}/r_{body}	3. fuel type	36. auto pilot on delay time
2.nose length ratio = L_{nose}/d_{body}	4. propellant out rad $r_{pvar}=(r_p+f)/r_{body}$	37. auto pilot time constant-tau, f.c.s. time const
10. fractional nozzle length ratio = f/r_p	5.propellant inner radius ratio= r_i/r_p	38. auto pilot damping - zeta, f.c.s. damping
11. throat diameter / D_{body}	6. number of star points	39. cross over freq - wcr, f.c.s. crossover freq
12. total length of stage1/ D_{body}	7. fillet radius ratio= f/r_p	40. pronav gain - n-prime, guidance gain
13. dia of stage1 center section d_{body} (in)	8. epsilon - star width	
14.wing exposed semi-span = b_{2w}/d_{body}	9. star point angle	
15. wing root chord = crw/d_{body}		
16. wing taper ratio = ctw/crw		
17. LE sweep angle deg		
18. $xLEw/l_{body}$		
Geometry (Bottom Stage)	Propellant (Bottom)	Auto Pilot (Bottom)

	Stage)	Stage)
26. fractional nozzle length ratio = f/rp	19. fuel type	41. auto pilot on delay time
27. throat diameter / D_{body}	20. propellant out rad $rpvar=(rp+f)/rbody$	42. auto pilot time constant- τ , f.c.s. time const
28. total length of stage1/ D_{body}	21. propellant inner radius ratio= ri/rp	43. auto pilot damping - zeta, f.c.s. damping
29. dia of stage1 center section d_{body} (in)	22. number of star points	44. cross over freq - wcr , f.c.s. crossover freq
30. wing exposed semi-span = $b2w/d_{body}$	23. fillet radius ratio= f/rp	45. pronav gain - n -prime, guidance gain
31. wing root chord = crw/d_{body}	24. epsilon - star width	
32. wing taper ratio = ctw/crw	25. star point angle	
33. LE sweep angle deg		
34. $xLEw/l_{body}$		
35. Upper Stage Separation Time		
46. Initial Launch Angle (deg)		

The second column in Table 2 and Table 3 is used to generate the solid propellant grain. An example of the cross-section of a solid propellant grain is shown in Figure 8. The third column for all three missile design codes is used for autopilot controls. For the tests used in this study the autopilot was not activated leaving only ballistic trajectory missiles.

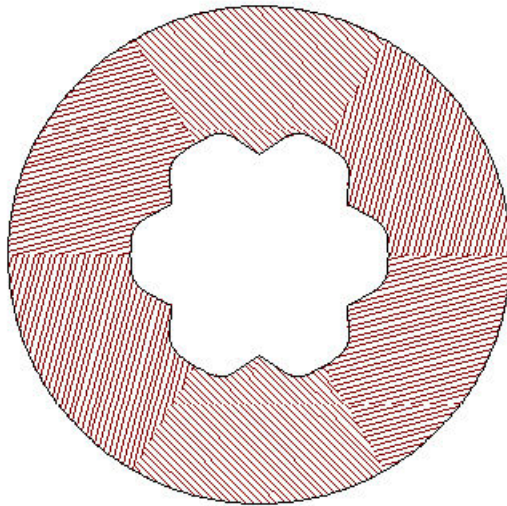


Figure 8. Solid Propellant Grain Cross-Section³³

The geometry (1st column in Table 2 and Table 3) is used to physically design the exterior of the missile including the nose, nozzle, tails and body as shown in Figure 9, Figure 10, and Figure 11.

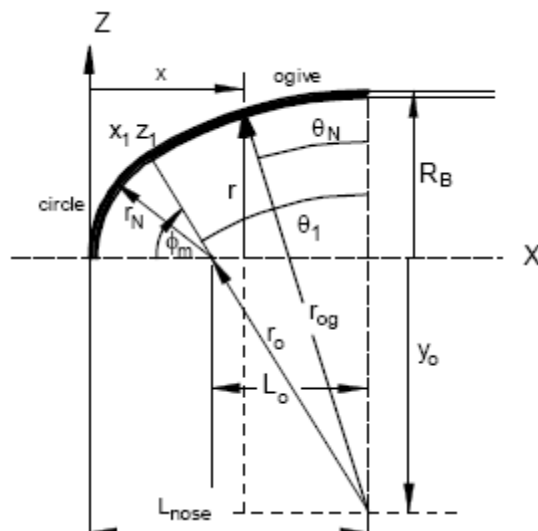


Figure 9. Nose Design Schematic³³

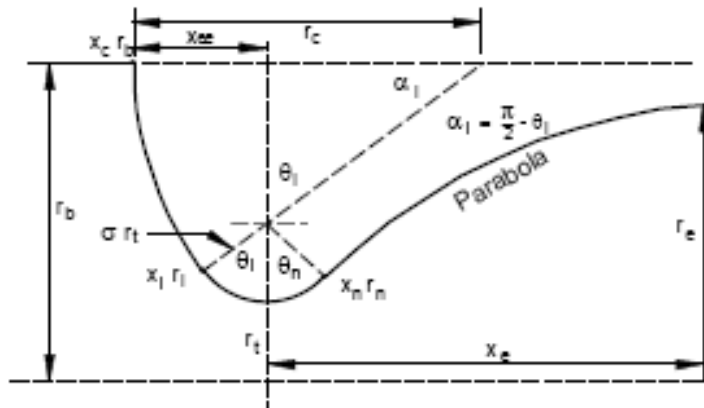


Figure 10. Nozzle Design Schematic³³

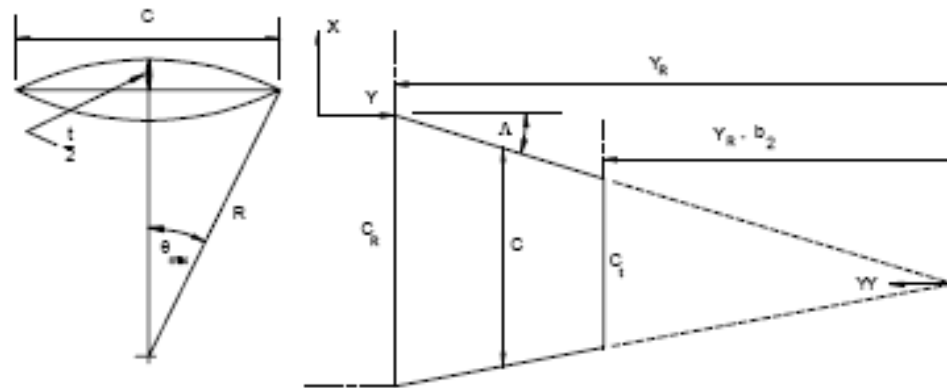


Figure 11. Tail Design Schematic³³

The third missile design code tested was the liquid propulsion missile design code. The liquid code follows a slightly different program flow than the two solid codes. The basic performance of the liquid code is based on the assumption that combustion takes place under constant pressure and constant temperature. The initial constants input file is set up similarly to the solid input files, however the GA variables input file is comprised of a different set of 26 design parameters. An overview of the 26 design variables is shown in Table 4. For the liquid system it is assumed to have a single stage

comprised of a single engine, where the fuel and oxidizer tanks are modeled to be cylindrical tanks with hemispherical endcaps.

Table 4. Single Stage Liquid GA Variable Definition

Geometry	Propellant	Auto Pilot
1. body diameter	2. propellant type	21. auto pilot on delay time
5. nose dia ratio = d_{nose}/DB	3. oxidizer to fuel ratio	22. auto pilot time constant- τ , f.c.s. time const
6. length of nose ratio = bl_{nose}/d_{nose}	4. chamber pressure	23. auto pilot damping - ζ , f.c.s. damping
7. nozzle throat dia ratio = d_{star}/d_{body}		24. cross over freq - ω_{cr} , f.c.s. crossover freq
8. nozzle expansion ratio = A_e/A^*		25. pronav gain - n -prime, guidance gain
9. fractional nozzle length		
11. wing root chord ratio = cr_{wing}/DB		
12. wing taper ratio		
13. wing b/2 ratio = b_{2wing}/DB		
14. wing le angle		
15. wing X loc = $x_{LEwing}/totlen$		
16. tail root chord ratio = cr_{tail}/DB		
17. tail taper ratio		
18. tail b/2 ratio = b_{2tail}/DB		
19. tail le angle deg		
20. tail X loc = $x_{TEtail}/totlen$		
10. burn time		
26. Initial Launch Angle (deg)		

The geometry for the liquid system is used to generate the physical model of the missile in the same way that it was developed for the two solid systems as shown in Figure 12. The main difference between the solid systems and the liquid systems is the propellant. For the liquid system, the code generated models the tanks used to store the fuel and oxidizer. The second column in Table 4 is used to determine the fuel-oxidizer combination as well as the initial chamber pressure. These inputs along with the geometric inputs are used to design the liquid propellant missile.

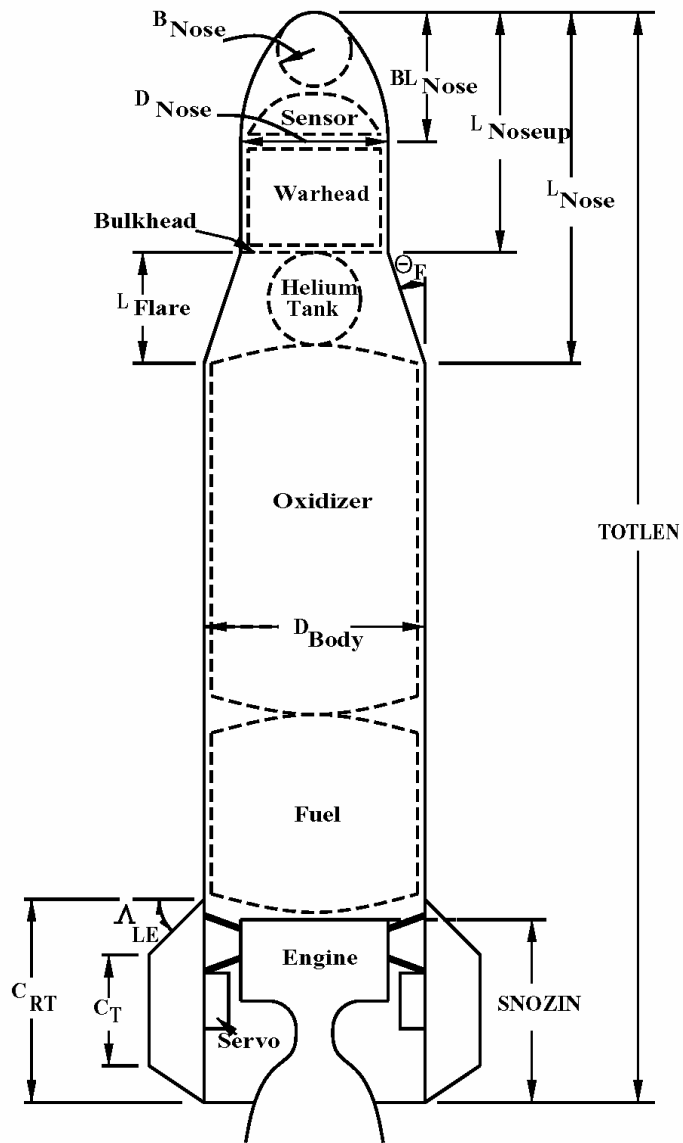


Figure 12. Liquid Missile Design Schematic³³

4 REAL GA CONVERGENCE TESTING

The binary GA IMPROVE[®] used in this study has been used for many years and has proven to be very robust. Because the binary GA has been proven to be successful with its current GA parameters, no testing was done in order to optimize the binary GA parameters, for this study they remained constant. These parameters can be seen in Appendix B-G. In order to compare the binary and the real GA's, the parameters for the real GA needed to be tested. The parameters tested were the type of crossover (blendx, single-point and uniform), mutation rate, mutation amount and population size. Changing any one of these parameters can cause the GA to converge faster or slower to a solution. For these tests the single stage solid missile system design code was used to match a range of 400,000 ft.

The first parameter tested was crossover. Each type of crossover was tested using the single stage solid code and run out to 10,000 function evaluations to determine which type would achieve the best fitness. All three types of crossovers performed well in this test, with the BlendX converging more rapidly to the best fitness. This coupled with the fact that the binary GA had single-point crossover and uniform crossover made BlendX the crossover type chosen for this study. The fitness for each test was calculated as follows:

$$Fitness = \frac{abs(Range - DesiredRange)}{DesiredRange}$$

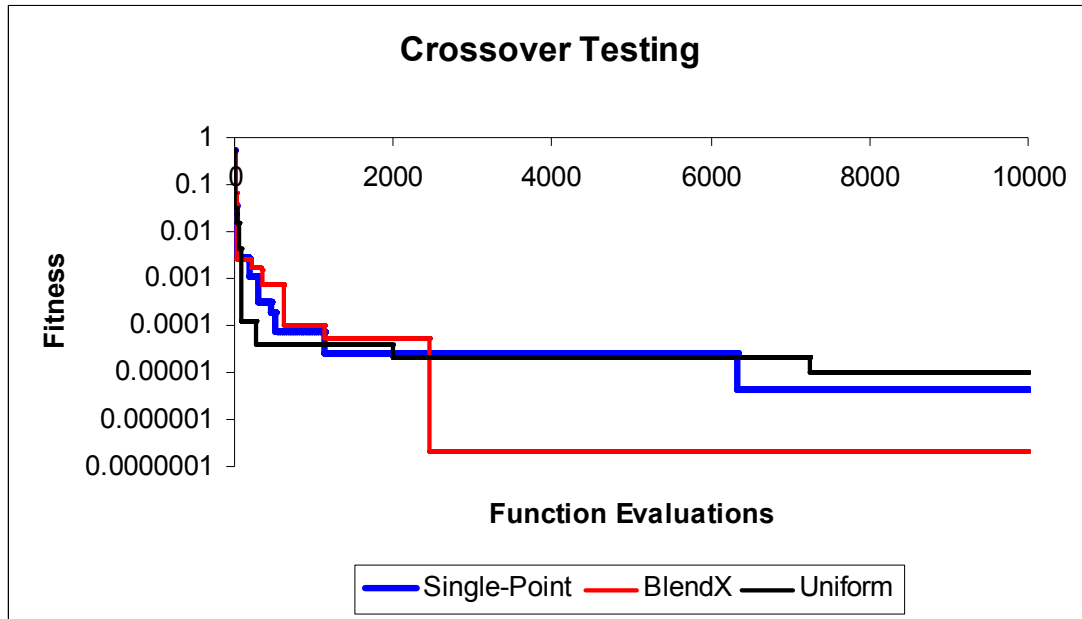


Figure 13. 1 Stage Solid Missile Crossover Testing 400,000 ft (Single) Goal

The next parameters studied were the mutation operators. Both mutation rate and mutation amount were tested in an attempt to find a trade off between quick convergence and not converging to a local minimum. Four different sets of mutation operators were tested using the single stage solid missile system design code matching a range of 400,000 ft. All GA parameters were held constant except the mutation rate and mutation amount. Table 5 shows the different sets of mutation operators and their converged fitness.

Table 5. Mutation Operator Fitness Comparison

Mutation Testing			
Mutation Operators	Mutation Rate	Mutation Amount	Fitness
Test_1	1.00	0.05	3.98E-06
Test_2	0.20	0.10	6.54E-09
Test_3	0.05	1.00	8.03E-12
Test_4	0.50	0.05	4.26E-06

Of the cases explored, the best converged fitness was achieved using a mutation rate of 0.05 and a mutation amount of 1.00. This means that the best convergence was obtained by only mutating 5% of the genes in each individual, but those 5% were mutated the most amount possible. Figure 14 shows the convergence data for each of the four cases tested.

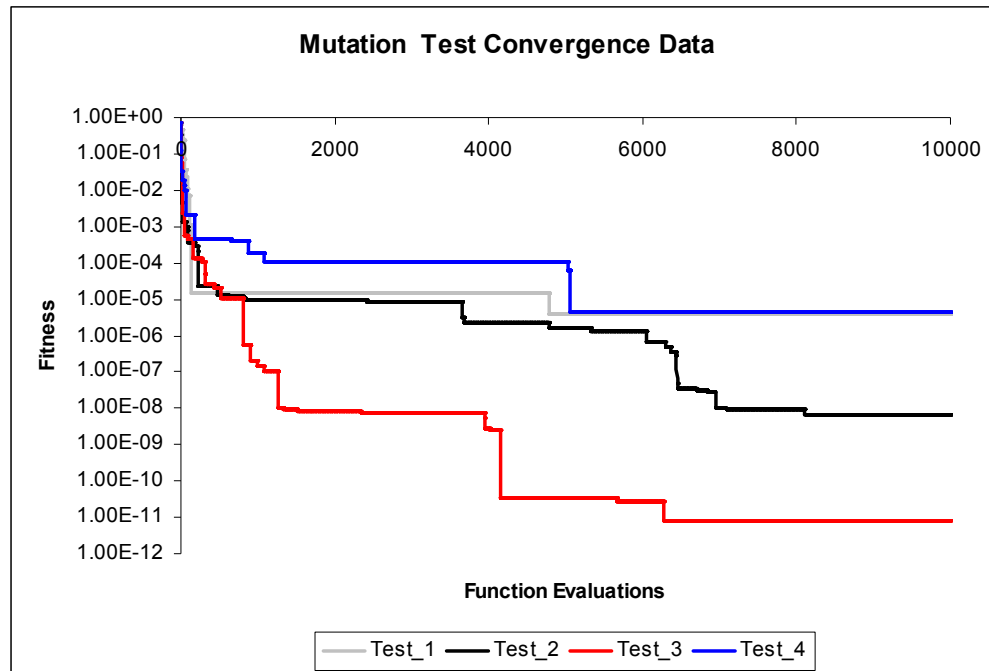


Figure 14. Single Stage Solid Missile Mutation Testing 400,000 ft Goal

The last parameter studied was the population size. This parameter is able to change the amount of selection pressure on good members of the population. The fewer the number of members in a population the higher probability that two good members could be chosen for the tournament. This allows for quicker convergence, yet again at the cost of potential premature convergence on a local minimum.

Table 6. Population Testing Fitness Comparison

Population Testing	Population Size	Fitness
Test_1	3	8.03E-12
Test_2	5	3.79E-06
Test_3	7	6.52E-11
Test_4	10	2.40E-06
Test_5	30	6.56E-11

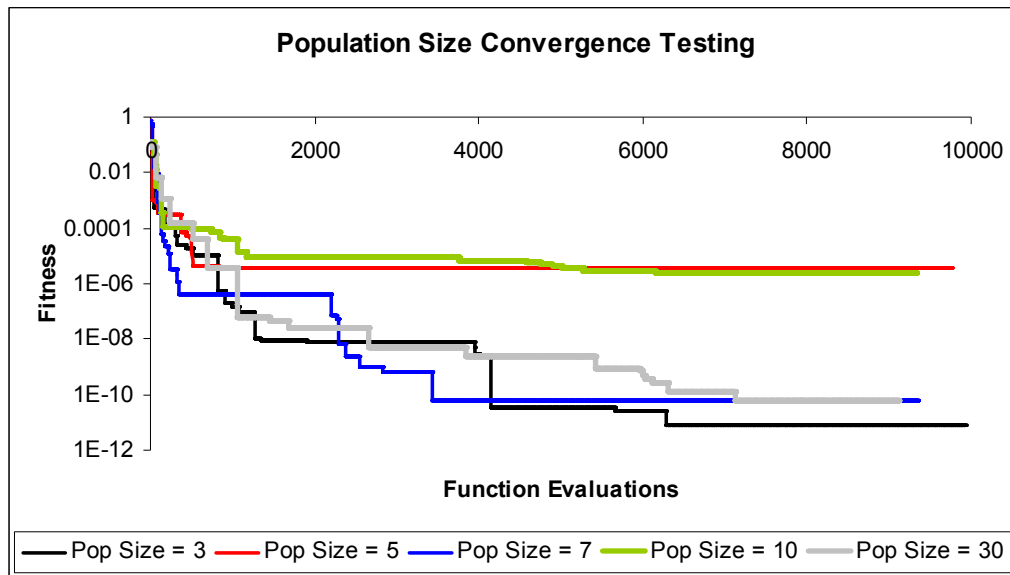


Figure 15. Single Stage Solid Missile Population Testing 400,000 ft Goal

Table 6 shows the results for the population testing. Of the five tests completed Test_1 having a population size of 3 was able to achieve the best fitness. Population sizes of 30 and 7 were also able to achieve surprisingly low fitness. Figure 15 shows the

convergence data for the 5 population tests conducted. A population size of three is the clear winner having a converge fitness of 8.03×10^{-12} in just 10,000 function evaluations.

The results of the GA convergence testing have shown that using a BlendX crossover with a mutation rate of 0.05, mutation amount of 1.0 and a population of 3 yielded the best results for a single stage solid missile design system with a single 400,000 ft match range goal. Therefore for the binary versus real GA comparisons these parameters will be used for all cases. Changing each parameter for each different GA run could certainly yield better results, however in an effort to compare for this comparison one set of parameters was chosen for each GA and held constant for each test.

Variable mutation was also tested to demonstrate its ability for quick convergence. Figure 16 shows the convergence history for a single stage solid with a goal of matching a 400,000 ft range. The fitness (miss distance) is plotted versus the number of function evaluations. For this test after every 40 function evaluations the number of better individuals was checked and the mutation rate was adjusted. The constant mutation rate real-coded GA was able to converge to a very good fitness of 9.85×10^{-5} in just 10,000 function evaluations. This corresponded to a range of 400,003.94 ft, with a miss distance of 3.94 ft. Keep in mind that this is a numerical result and is not intended to represent the accuracy of the physical model. For this test the fitness was determined using the equation shown below:

$$Fitness = \frac{abs(Range - DesiredRange)}{DesiredRange}$$

With variable mutation turned on the GA was able to converge to a fitness of 10^{-8} in under 2,000 function evaluations. The variable mutation allowed the GA to converge

to a better solution in 80% less function evaluations compared to the constant mutation rate real-coded GA. After 10,000 function evaluations the variable mutation GA was able to converge to a fitness of 1.31×10^{-15} , corresponding to a miss distance of 5.24×10^{-11} ft.

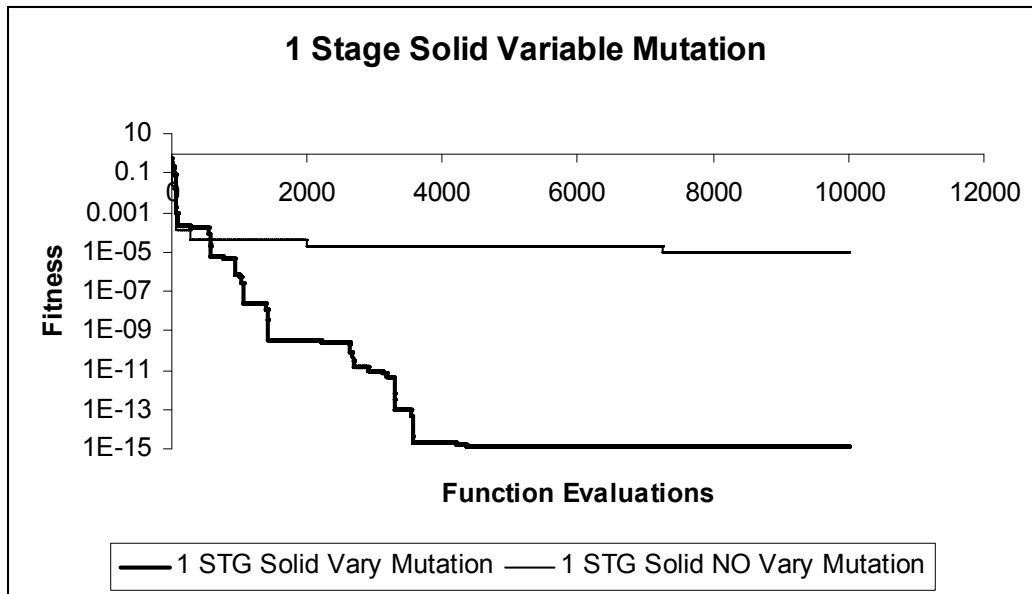


Figure 16. 1 Stage Solid Variable Mutation Convergence Plot

The variable mutation GA was able to obtain its final fitness after 4,000 function evaluation, 50% less than the regular real coded GA. Variable mutation gives the real coded GA the possibility to converge to a very accurate solution very rapidly however it also has the tendency to get stuck on a local minimum and converge to a non-optimum solution. For this reason variable mutation was not used for the binary versus real GA comparison tests.

5 MISSILE SYSTEM DESIGNS GA COMPARISONS

Comparisons of the real and binary GA were conducted using all three missile system design codes outlined in Chapter 3. Initially both GA's were setup to optimize a single goal, match a range. These tests are shown below in Chapter 5 Section 1. It was later determined that a more robust comparison would be two optimize two goals for each GA, match a range and match a weight, or any two goal combination that would drive each GA to a similar global optimum. Optimizing two goals for the missile system design codes drives the GA's to a particular solution because it limits the possibilities for good missiles. For the match range goal, there are a wide variety of missiles that can match the range, however if a match range and match weight goals are both in effect the number of possible solutions is limited.

5.1 SINGLE STAGE SOLID MISSILE SYSTEM DESIGN

For both the real and binary GA's, each variable is given a maximum allowable value, and a minimum allowable value. For the binary GA, each variable is also given a resolution, this resolution is limited by the number of bits as described in Chapter 1.

The initial excitement for the real coded GA came from preliminary comparisons of the real and binary GA's for single stage solid propellant missile design using a single goal, match range. The first two tests conducted demonstrated the real GA's potential for quick and accurate convergence when compared to the binary GA. The convergence plots for both tests are shown in Figure 17 and Figure 18.

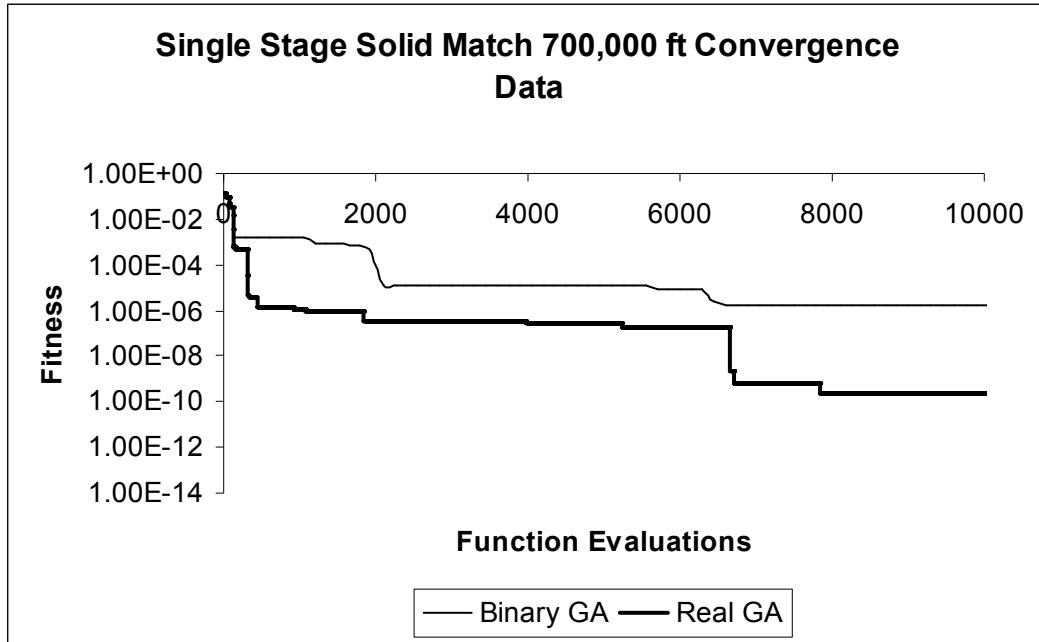


Figure 17. Single Stage Solid 700,000 ft Preliminary Test Convergence History

The first preliminary test was to design a single stage solid missile that could hit a target 700,000 ft down range. The fitness is calculated as follows:

$$Fitness = \frac{abs(Range - DesiredRange)}{10}$$

The real-coded GA was able to achieve a better fitness than the binary GA in just over 2,000 function evaluations, and when let run to the full 10,000 function evaluations was able to converge to a solution 6 orders of magnitude more accurate. It should be noted that for this problem a miss distance of a foot or less is practically considered to be a hit. Therefore the increased precision is for academic purposes only.

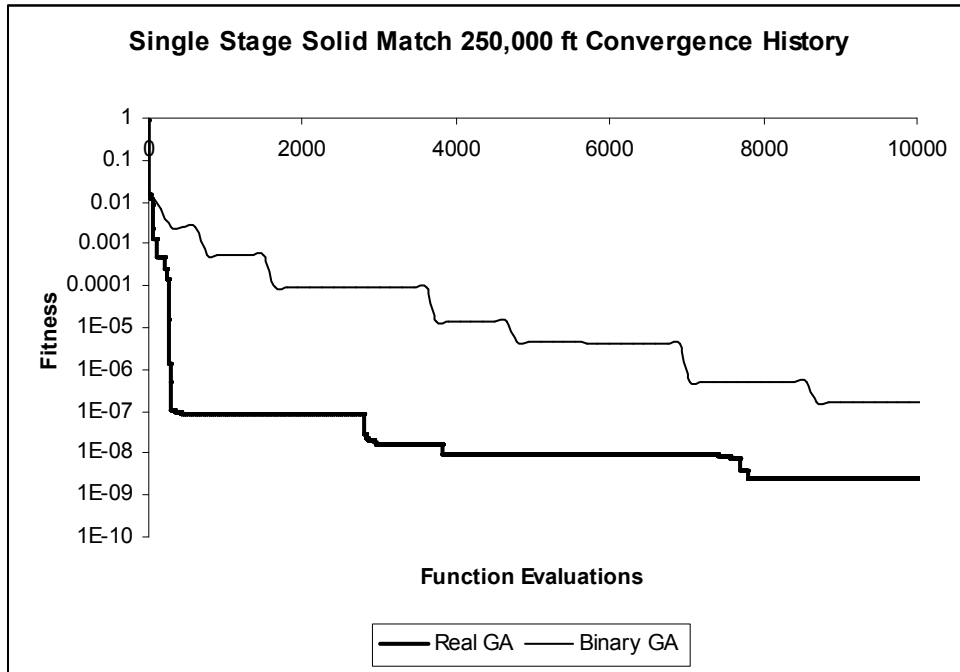


Figure 18. Single Stage Solid 250,000 ft Preliminary Test Convergence History

The second preliminary test was to design a single stage solid missile that could hit a target 250,000 ft down range. The fitness is calculated as follows:

$$Fitness = \frac{abs(Range - DesiredRange)}{10}$$

The real GA was able to achieve a better fitness than the binary GA in just over 300 function evaluations, and when let run to the full 10,000 function evaluations was able to converge to a solution 2 orders of magnitude more accurate. The success of both

preliminary tests lead to more detailed and complex goals to better compare the two GA's.

Three more complex cases were tested in order to compare the binary and real coded GA's: match range 100,000 ft while also matching a 2,500 lb take-off weight, match range 350,000 ft while also matching a 4,000 lb take-off weight, and match a 2,500lb take-off weight with a 240 second time of flight. By adding a second goal to each test both GA's would be forced to a global solution in order to minimize each goal. Both missile design codes were identical with the exception of the GA's. The GA input file (GANNL.DAT) can be seen in Appendix B for the real coded GA and Appendix C for the binary coded GA. Fitness' for the single stage solid system were calculated using the following equations. The real and binary GA's both summed up the fitnesses for multiple goals, therefore the optimization only consisted of a single number to minimize as shown in the equations below.

$$Fitness = \frac{abs(Range - DesiredRange)}{10} + \frac{abs(Mass - DesiredMass)}{10}$$

$$Fitness = \frac{abs(Weight - DesiredWeight)}{10} + \frac{abs(TOF - DesiredTOF)}{10}$$

A summary of the single stage solid optimization results is shown in Table 7. For the single stage solid propellant missile design code the binary coded GA was able to converge to better solutions more rapidly than the real GA for all three cases tested. For two of the three cases tested both the real and binary GA's converged to very similar results.

Table 7. Single Stage Solid Optimization Results

GA Fitness Comparison	Real GA		Binary GA	F.E.
1 Stg Solid				
Match 100kft 2500lb weight	50.61		46.69	10,000
Match 350kft 4000lb weight	48.67		3.98	10,000
Match 2500 lbs 240 sec	16.65		16.61	10,000

5.1.1 SINGLE STAGE SOLID 100,000 FT 2,500 LB GOAL

For the first test the binary coded genetic algorithm was able to produce a missile that could hit a target 100,000.53 ft down range with a 2,969.91 lb initial take-off weight using 10,000 function evaluations with a converged solution having a fitness of 46.69.

Table 8. Single stage solid 100,000 ft 2,500 lb Final Design Variables

Real GA	Binary GA	Design Variable Definition	
0.4667	0.4732	1	rnose/rbody
2.7097	2.1902	2	lnose/dbody
2.6000	3.6958	3	(1)fuel type
0.5857	0.5955	4	(1)star out R
0.7067	0.5538	5	(1)star inner ratio
6.6000	10.2099	6	(1)number of star points
0.0627	0.0665	7	(1)fillet radius ratio
0.6111	0.6622	8	(1)eps
5.0000	4.7193	9	(1)star point angle deg
0.7386	0.8935	10	(1)fractional nozzle length
0.2825	0.2996	11	(1)Dia throat/Dbody
12.0000	12.1720	12	(1)Fineness Ratio
1.3410	1.3245	13	(1)dia stage 1 ft
0.0386	0.0495	14	(1)wing semispan / dbody
0.0271	0.0114	15	(1)wing root chord/dbody
0.9900	0.9708	16	(1)taper ratio
25.8571	15.8560	17	(1)wing LE sweep angle deg
0.2000	0.2105	18	(1)xLEw/lbody
1.3333	1.3965	19	(1)tail semispan/dbody
0.9000	1.0415	20	(1)tail root chord/dbody
0.9591	0.8318	21	(1)tail taper ratio
29.0794	12.8401	22	(1)LE sweep angle deg
0.9500	0.9642	23	(1)xTEt/lbody
4000.0000	4000.1293	24	(1)autopilot delay time sec
0.4000	0.7345	25	(1)auto pilot time constant sec
0.7622	0.4595	26	(1)auto pilot damping
54.2857	64.1508	27	(1)crossover freq hz
4.1613	3.1442	28	(1)pronav gain
62.1429	56.4049	29	initial launch angle deg

This corresponded to a miss distance of 0.53 ft and a miss weight of 469.91 lb. Given the same inputs the real coded GA was able to achieve a fitness of 50.61 after 10,000 function evaluations. This corresponded to a miss distance of 0.00 ft and a miss weight of 506.10 lb. Table 8 lists all of the final design parameters chosen by both GA's. Both GA's converged to different propellant types and grain geometry as shown in design variables 3-9. The only other major differences in the converged design parameters were tail geometries shown in variables 19-23. The body diameter converged to very similar values for both GA's, and can be seen in variable 13.

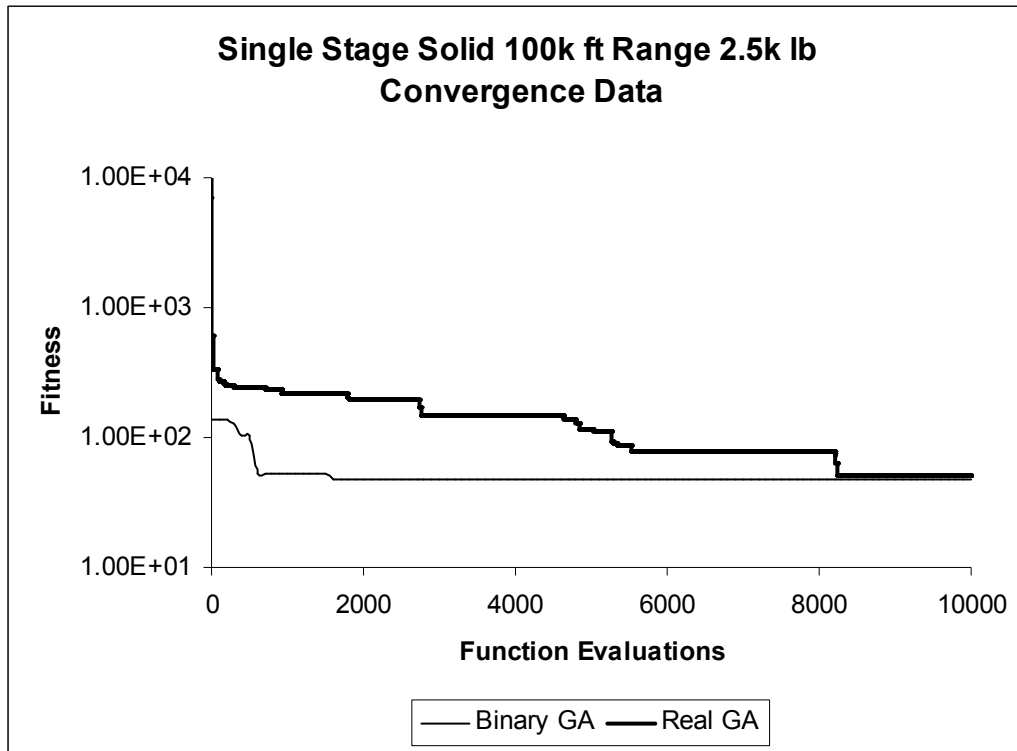


Figure 19. Single stage solid 100,000 ft 2,500 lb convergence

Even though there are major design differences between the two missiles, both were able to meet their goals with similar fitness's. A comparison of the performance of the two GA's is shown in Figure 19. The binary coded GA was able to produce a more accurate converged solution quicker than its real counterpart.

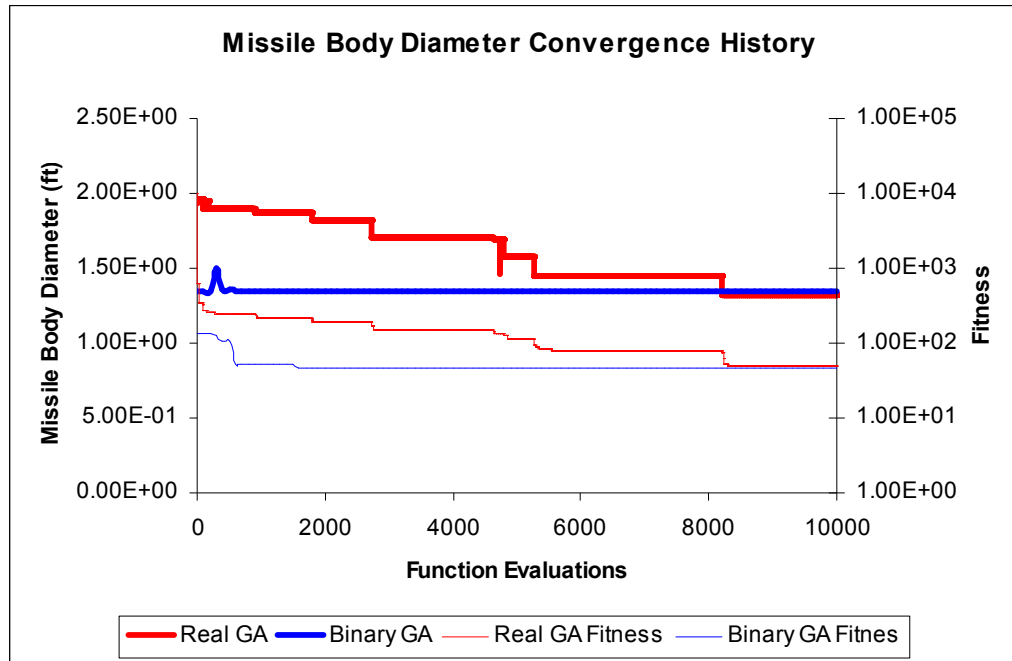


Figure 20. Single Stage Solid Missile Body Diameter Convergence History

Figure 20 shows how both GA's changed the missile body diameter as they converged to their final solution. The binary GA started out with a small missile body diameter for its initial guess, this contributed to its quicker convergence compared to the real coded GA. The real GA's initial guess for body diameter was much larger than the binary GA's initial guess. It should be noted that the overall convergence of the real GA closely follows the minimizing of the missile body diameter. The initial body diameter for the real GA was 1.94 ft, while the binary GA's initial diameter was 1.34 ft. After 10,000 function evaluations the real GA converged to a body diameter of 1.32 ft, while the binary GA's diameter remained at 1.34 ft. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 21 and Figure 22.

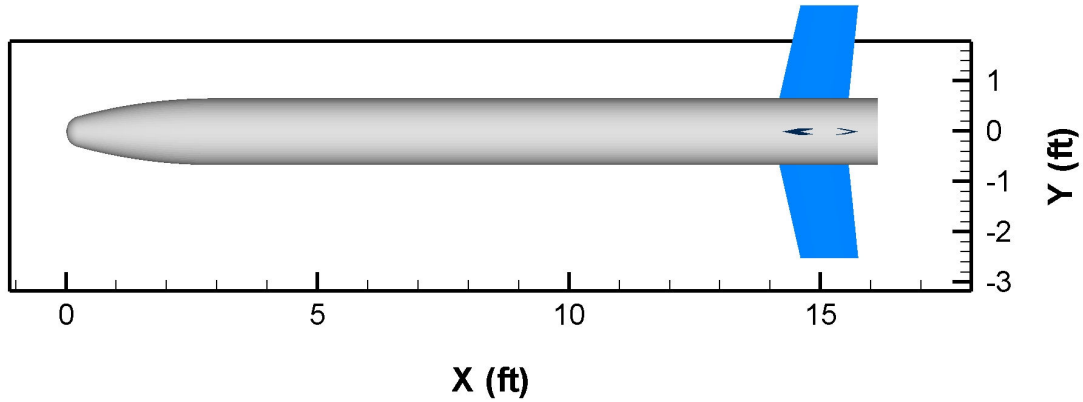


Figure 21. Single Stage Solid 3D Model-Real GA

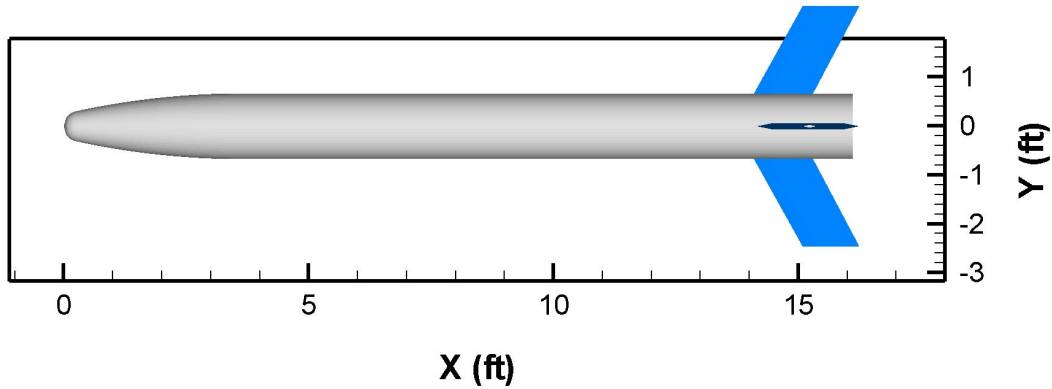


Figure 22. Single Stage Solid 3D Model-Binary GA

5.1.2 SINGLE STAGE SOLID 350,000 FT 4,000 LB GOAL

For the second test the binary coded genetic algorithm was able to produce a missile that could hit a target 350,000.05 ft down range with a 4,039.75 lb initial take-off weight using 10,000 function evaluations with a converged solution having a fitness of 3.982. This corresponded to a miss distance of 0.05 ft and a miss weight of 39.75 lb. Given the same inputs the real coded GA was able to achieve a fitness of 48.67 after 10,000 function evaluations. This corresponded to a miss distance of 0.00 ft and a miss weight of 485.6 lb. Table 9 shows the final design variables chosen by both GA's.

Table 9. Single stage solid 350,000 ft 4,000 lb Final Design Variables

Real GA	Binary GA	Design Variable Definition	
0.5550	0.4000	1	rnose/rbody
2.1030	1.8387	2	lnose/dbody
7.6170	2.0667	3	(1)fuel type
0.5464	0.4429	4	(1)star out R
0.8000	0.5667	5	(1)star inner ratio
7.6293	10.2000	6	(1)number of star points
0.0751	0.0907	7	(1)fillet radius ratio
0.6707	0.7833	8	(1)eps
3.8307	3.2857	9	(1)star point angle deg
0.8320	0.7543	10	(1)fractional nozzle length
0.2803	0.2619	11	(1)Dia throat/Dbody
12.6541	14.3333	12	(1)Fineness Ratio
1.7177	1.4312	13	(1)dia stage 1 ft
0.0204	0.0386	14	(1)wing semispan / dbody
0.0332	0.0157	15	(1)wing root chord/dbody
0.9230	0.9360	16	(1)taper ratio
5.4306	23.0952	17	(1)wing LE sweep angle deg
0.2000	0.2000	18	(1)xLEw/lbody
1.3111	1.2000	19	(1)tail semispan/dbody
1.0726	0.9000	20	(1)tail root chord/dbody
0.7168	0.6620	21	(1)tail taper ratio
23.5479	1.9206	22	(1)LE sweep angle deg
0.9754	1.0000	23	(1)xTEt/lbody
3999.6107	4000.0000	24	(1)autopilot delay time sec
0.5869	0.4000	25	(1)auto pilot time constant sec
0.8690	0.7819	26	(1)auto pilot damping
47.5653	48.0952	27	(1)crossover freq hz
3.3209	4.0323	28	(1)pronav gain
74.2459	44.2857	29	initial launch angle deg

For this case almost all of the final converged design variables differed for both GA's. The propellant type and grain geometry converged to different values as shown in variables 3-9. The wing and tail geometries, variables 14-23, also converged to different values. The initial launch angle, variable 29 and the body diameter, variable 13 also converged to very different values. These parameters played a very important role in the convergence of this missile system, as will be shown in later sections. A comparison of the performance of the two GA's is shown in Figure 23.

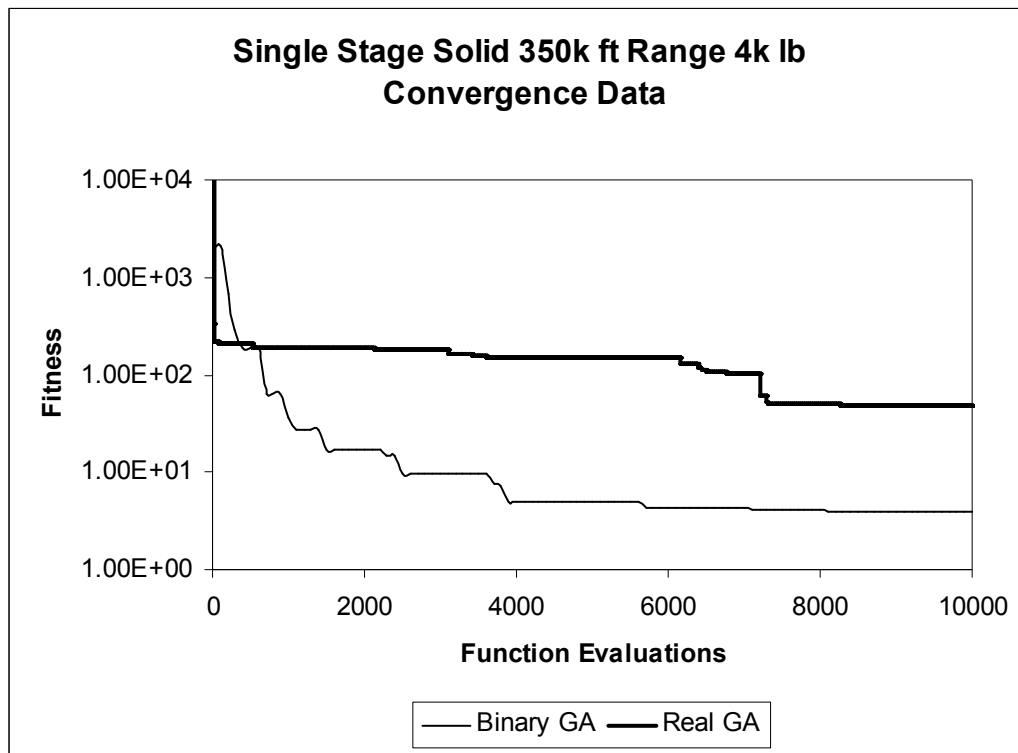


Figure 23. Single stage solid 350,000 ft 4,000 lb convergence

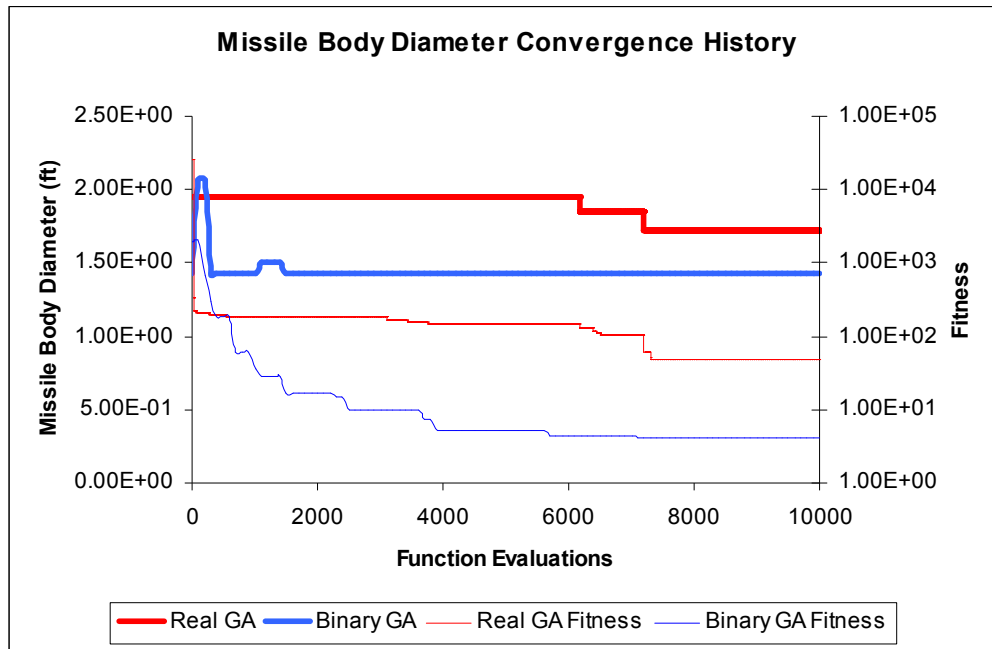


Figure 24. Single Stage Solid Missile Body Diameter Convergence History

Figure 24 shows the how both GA’s changed the missile body diameter as they converged to their final solution. The binary GA was able to quickly converge to a small missile body diameter, this contributed to its quicker convergence compared to the real coded GA. The real GA’s initial guess for body diameter was much larger than the binary GA’s initial guess. The initial body diameter for the real GA was 1.95 ft, while the binary GA’s initial diameter was 1.43 ft. After 10,000 function evaluations the real GA converged to a body diameter of 1.72 ft, while the binary GA’s diameter remained at 1.43 ft.

It should be noted that the overall convergence of the real GA closely follows the minimizing of the missile body diameter, while the binary GA’s overall convergence does not. Figure 25 shows the convergence of another GA design variable, initial launch angle. Figure 25 shows that the convergence of this design variable contributed more for

the overall convergence of the binary GA. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 26 and Figure 27.

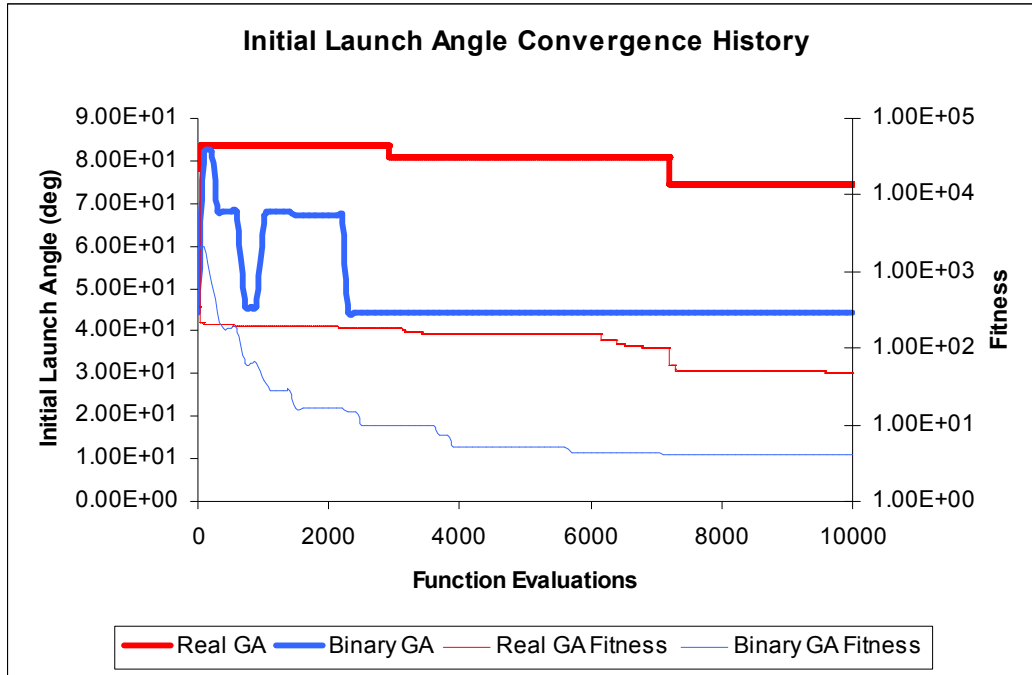


Figure 25. Single Stage Solid Missile Initial Launch Angle Convergence History

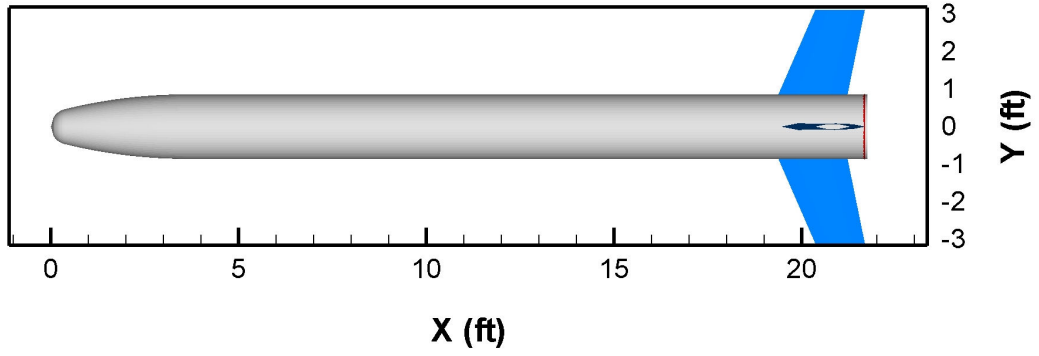


Figure 26. Single Stage Solid 3D Model-Real GA

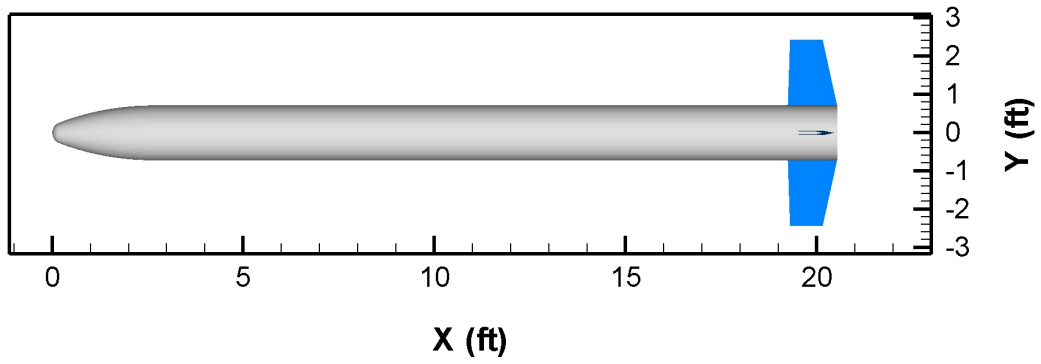


Figure 27. Single Stage Solid 3D Model-Binary GA

5.1.3 SINGLE STAGE SOLID 2,500 LB 240 SEC TOF GOAL

For the third test the binary coded genetic algorithm was able to produce a missile with a 2,500.00 lb initial take-off weight and a 406.05 second flight time using 10,000 function evaluations with a converged solution having a fitness of 16.61. This corresponded to a miss weight of 0.00 lb and miss time of 166.05 seconds.

Table 10. Single stage solid 2,500 lb 240 sec Final Design Variables

Real GA	Binary GA	Design Variable Definition	
0.4000	0.4000	1	rnose/rbody
1.7483	2.4194	2	lnose/dbody
2.9399	2.0667	3	(1)fuel type
0.6144	0.5143	4	(1)star out R
0.7426	0.6600	5	(1)star inner ratio
10.1264	5.0000	6	(1)number of star points
0.0300	0.0533	7	(1)fillet radius ratio
0.6000	0.6000	8	(1)eps
4.8342	1.0000	9	(1)star point angle deg
0.7293	0.7124	10	(1)fractional nozzle length
0.2500	0.2683	11	(1)Dia throat/Dbody
12.0881	14.0000	12	(1)Fineness Ratio
1.0447	0.9579	13	(1)dia stage 1 ft
0.0401	0.0100	14	(1)wing semispan / dbody
0.0234	0.0329	15	(1)wing root chord/dbody
0.9852	0.9720	16	(1)taper ratio
10.5898	18.0317	17	(1)wing LE sweep angle deg
0.2010	0.2071	18	(1)xLEw/lbody
1.3036	1.3333	19	(1)tail semispan/dbody
0.9641	0.9667	20	(1)tail root chord/dbody
0.7347	0.5077	21	(1)tail taper ratio
30.0000	29.0794	22	(1)LE sweep angle deg
0.9562	0.9857	23	(1)xTEt/lbody
3999.8384	3999.0000	24	(1)autopilot delay time sec
0.5699	0.6286	25	(1)auto pilot time constant sec
0.7403	0.4945	26	(1)auto pilot damping
40.0000	51.4286	27	(1)crossover freq hz
3.0626	3.3226	28	(1)pronav gain
85.0000	85.0000	29	initial launch angle deg

Given the same inputs the real coded GA was able to achieve a fitness of 16.65 after 10,000 function evaluations. This corresponded to a miss weight of 0.00 lb and a

miss time of 166.49 seconds. Table 10 shows the final converged design variables for each GA. Both the real and binary GA's produced similar missiles for this test. When analyzing Table 11 the only real difference between the two missiles is the fineness ratio, variable 12, which is defined as the length of the missile divided by the diameter of the body. The binary GA converged to a larger fineness ratio of 14 while the real GA converged to a value of just above 12. This effectively made the binary GA longer as shown in Figure 28. A comparison of the performance of the two GA's is shown in Figure 28. The binary coded GA was able to produce a slightly more accurate converged solution quicker than its real counterpart.

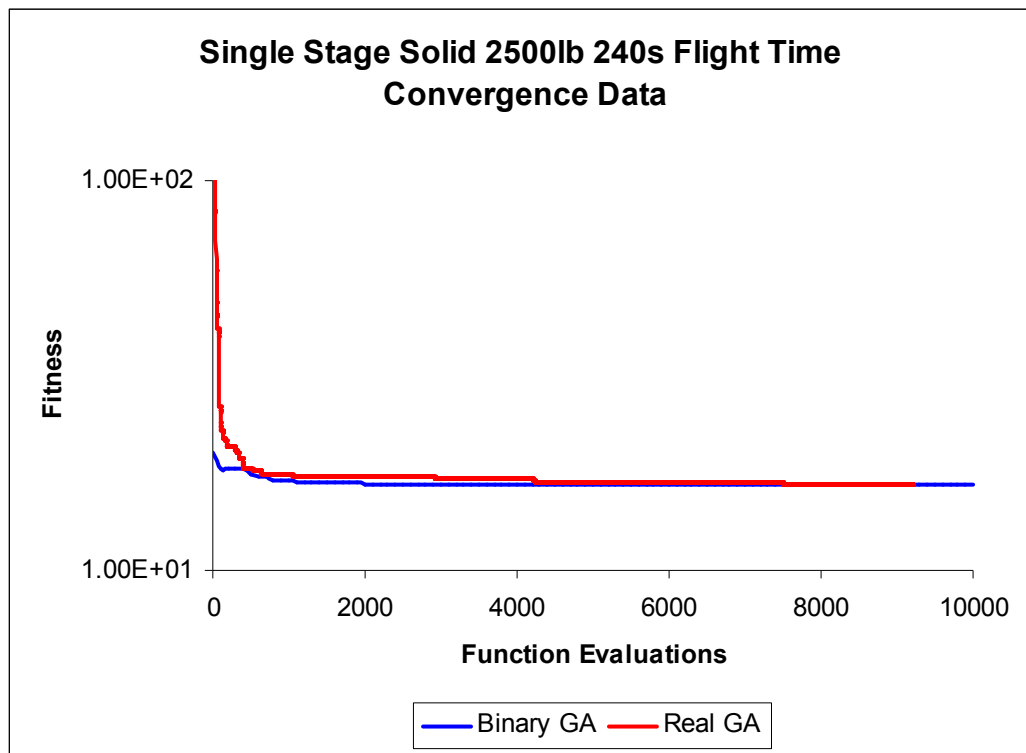


Figure 28. Single stage solid 2,500 lb 240sec convergence history

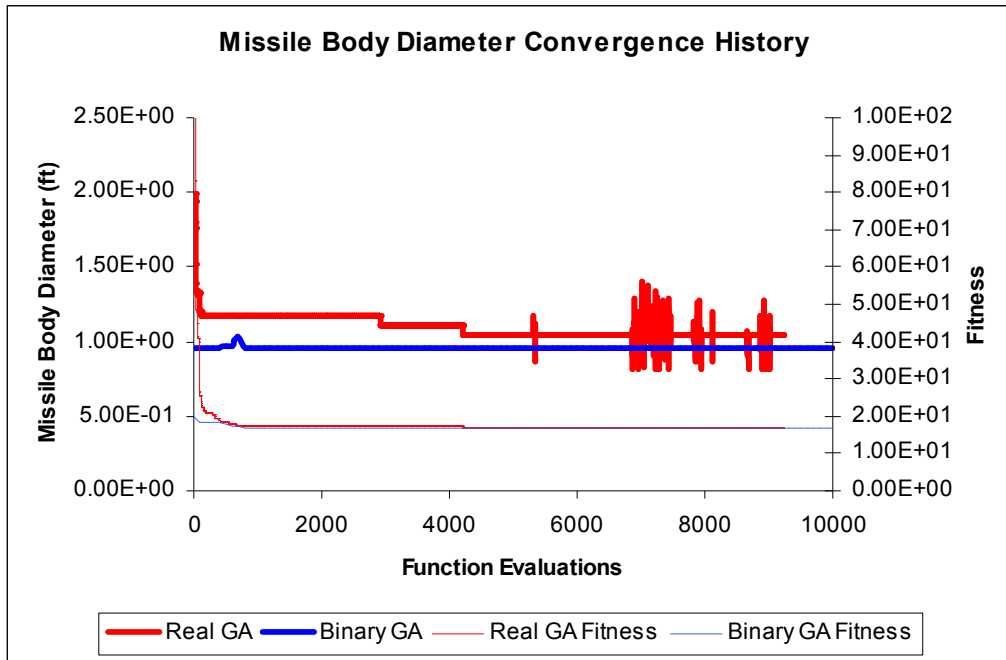


Figure 29. Single Stage Solid Missile Body Diameter Convergence History

Figure 29 shows the how both GA's changed the missile body diameter as they converged to their final solution. Body diameter did not contribute as much to the overall fitness for the binary GA as it did for the real GA. The binary GA's diameter started off at a small value and remained there throughout the GA run. The diameter did however change dramatically for the real GA. The initial drop in the fitness of the real GA was largely contributed by the initial decrease in body diameter from 1.93 ft to 1.18 ft in the first 100 function evaluations. Since the second goal for this test was match a flight time, it is not surprising that the initial launch angle played a major role in the overall fitness of both GA's. A plot of the initial launch angle vs. function evaluations can be seen in Figure 30.

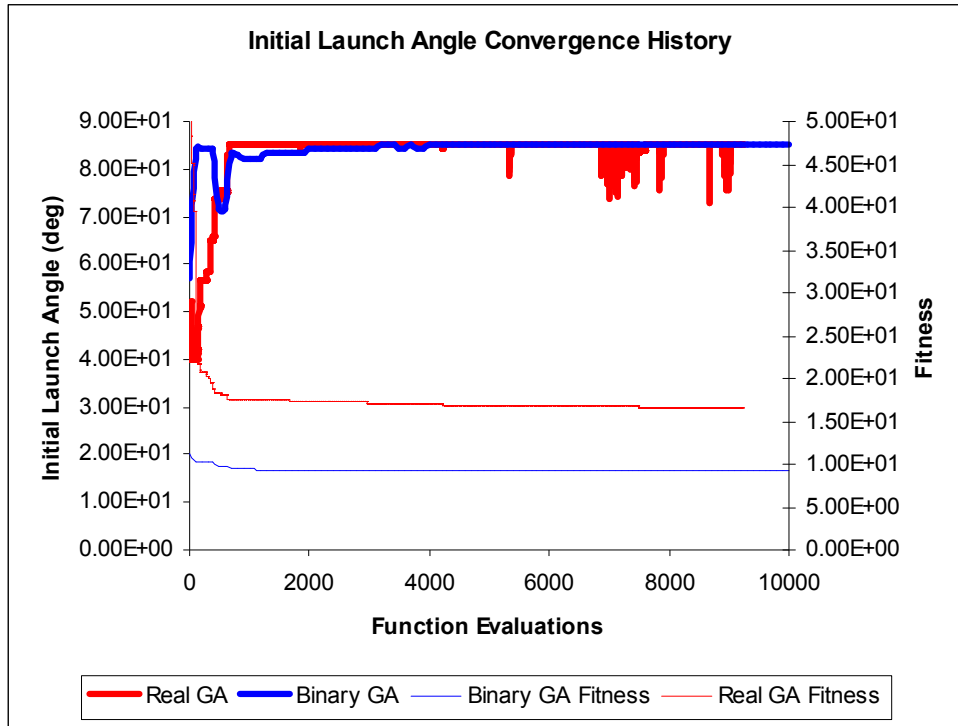


Figure 30. Single Stage Solid Missile Initial Launch Angle Convergence History

Figure 30 shows that the convergence of the initial launch angle contributed to the overall convergence of both GA's. The real and binary GA's both started with relatively low initial launch angles and quickly increased the angle to increase the time of flight while maintaining a low take-off weight. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 31 and Figure 32.

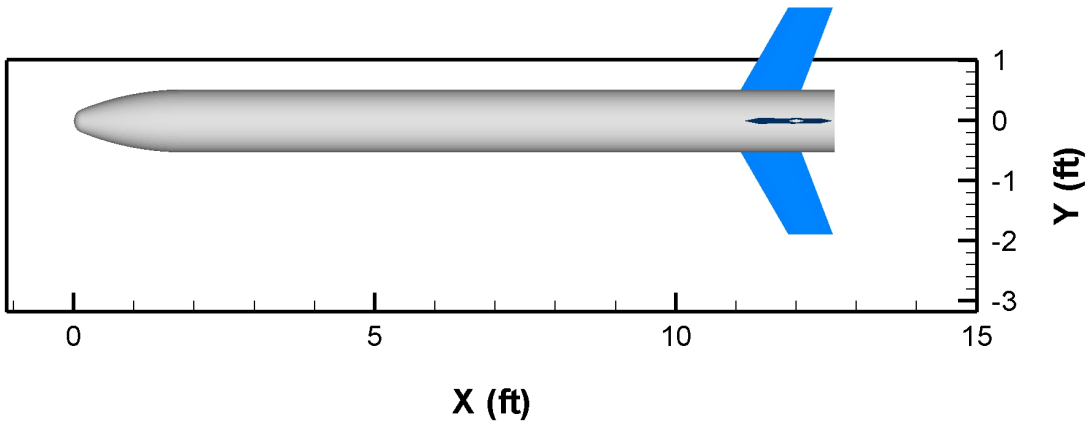


Figure 31. Single Stage Solid 3D Model-Real GA

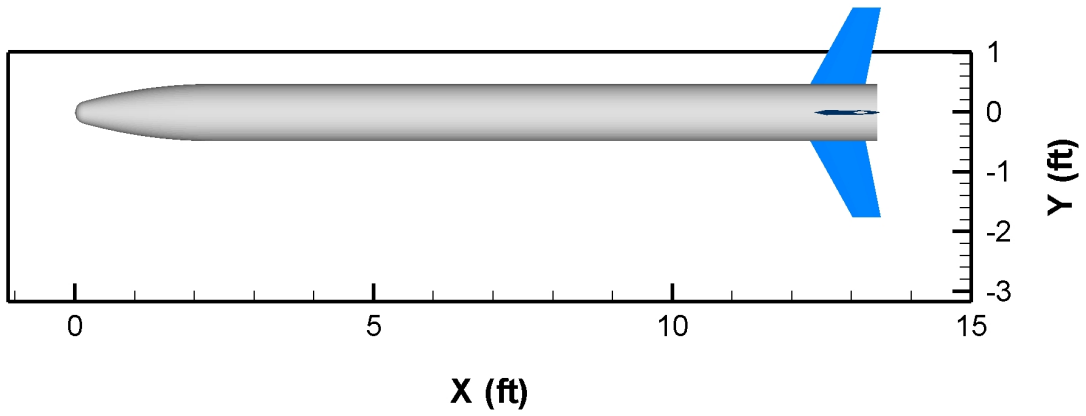


Figure 32. Single Stage Solid 3D Model-Binary GA

5.2 TWO STAGE SOLID MISSILE SYSTEM DESIGN

Two cases were tested in order to compare the binary and real coded GA's for the two stage solid propellant missile system: match range 100,000 ft with a 6,000 lb initial take-off weight and match range 250,000 ft with a 7,000 lb initial take-off weight. Both missile design codes were identical with the exception of the GA's. The GA input file (GANNL.DAT) can be seen in Appendix D for the real coded GA and Appendix E for the binary coded GA. The fitness' for the two stage system were calculated as shown below.

$$Fitness = \frac{abs(Range - DesiredRange)}{10.0}$$

$$Fitness = \frac{abs(Weight - DesiredWeight)}{10.0}$$

A summary of the 2 stage solid optimization results is shown in Table 11. For the two stage solid propellant missile design code the real coded GA was able to converge to a better solution more rapidly for one case, while the binary GA was able to converge to a better solution for the other case tested.

Table 11. Two Stage Solid Optimization Results

GA Fitness Comparison	Real GA		Binary GA	F.E.
2 Stg Solid				
match 100k ft 6000lbm weight	408.6		414.1	10,000
match 250k ft 7000lbm weight	5148		314.1	10,000

5.2.1 TWO STAGE SOLID 100,000 FT 6,000 LB GOAL

For the first test the binary coded genetic algorithm was able to produce a missile that could hit a target 101,387.69 ft down range with a 8,758.00 lb initial take-off weight using 10,000 function evaluations with a converged solution having a fitness of 414.1. This corresponded to a miss distance of 1,387.69 ft and a miss weight of 2,758.00 lb. Given the same inputs the real coded GA was able to achieve a fitness of 408.6 after 10,000 function evaluations. This corresponded to a miss distance of 1,427.14 ft and a miss weight of 2,658.80 lb. Table 12 shows the final design variables that both the real and binary GA's converged to.

Table 12. Two stage solid 100,000 ft 6,000 lb Final Design Variables

Real GA	Binary GA		Design Variable Definition
0.4000	0.5333	1	nose radius ratio
3.0000	2.1774	2	nose length ratio
6.0916	6.1000	3	(1)fuel type
0.5000	0.5000	4	(1)propellant outer radius ratio
0.5022	0.5000	5	(1)propellant inner radius ratio
9.0000	9.1000	6	(1)number of star points
0.0896	0.0957	7	(1)fillet radius ratio
0.8000	0.9000	8	(1)epsilon
10.0000	11.1063	9	(1)star point angle
0.9369	0.9313	10	(1)fractional nozzle length ratio
0.2800	0.2933	11	(1)throat diameter/dbody
11.9668	10.1000	12	(1)total length of stg 1 in
1.5359	1.8331	13	(1)dia of stg 1 center body ft
1.2000	1.2000	14	(1)exposed semi span/dbody1
0.9000	0.9000	15	(1)root chord/dbody1
0.9600	0.9814	16	(1)taper ratio
39.3637	40.0000	17	(1)LE sweep angle deg
0.9938	1.0000	18	(1)xTE/lbody1
6.0000	6.1000	19	(2)fuel type
0.4000	0.4000	20	(2)propellant outer radius ratio
0.6000	0.6000	21	(2)propellant inner radius ratio
9.0948	9.0000	22	(2)number of star points
0.1000	0.1000	23	(2)fillet radius ratio
0.8756	0.8667	24	(2)epsilon
10.0000	10.0000	25	(2)star point angle
0.9900	0.9900	26	(2)fractional nozzle length ratio

0.3200	0.3200	27	(2)throat dia/dbody2
7.0000	7.0000	28	(2)total length of booster stage 2 ft
1.4000	1.4000	29	(2)dia of booster center section stg 2 ft
1.4000	1.4000	30	(2)exposed semi-span/dbody2
1.0956	1.1000	31	(2)root chord/dbody2
0.9600	0.9814	32	(2)taper ratio
2.0000	15.4839	33	(2)LE sweep angle deg
1.0000	0.9933	34	(2)xTE/lbody2
2.4092	2.2000	35	time to separate stage 1 sec
4001.0000	4000.0000	36	(1)autopilot on delat time stage 1
0.5570	0.5714	37	(1)autopilot time constant stage 1
0.6376	0.6190	38	(1)autopilot damping stage 1
60.0000	59.3333	39	(1)crossover freq hz stage 1
5.6430	5.8000	40	(1)pronavgain stage 1
4000.5150	4001.0000	41	(2)autopilot on delat time stage 2
0.6493	0.4857	42	(2)autopilot time constant stage 2
0.6000	0.8048	43	(2)autopilot damping stage 2
51.2369	50.6667	44	(2)crossover freq hz stage 2
3.0000	3.6000	45	(2)pronavgain stage 2
45.0906	45.0000	46	initial launch angle deg

Both the real and binary GA's produced similar missiles for this test. When analyzing Table 12 the only real difference between the two missiles was the top stage body diameter variable 13. The binary GA converged to a larger body diameter of 1.83 ft while the real GA converged to a value of just above 1.5 ft. This effectively increased the binary GA's weight increasing the fitness as shown in Figure 33. A comparison of the performance of the two GA's is shown in Figure 33.

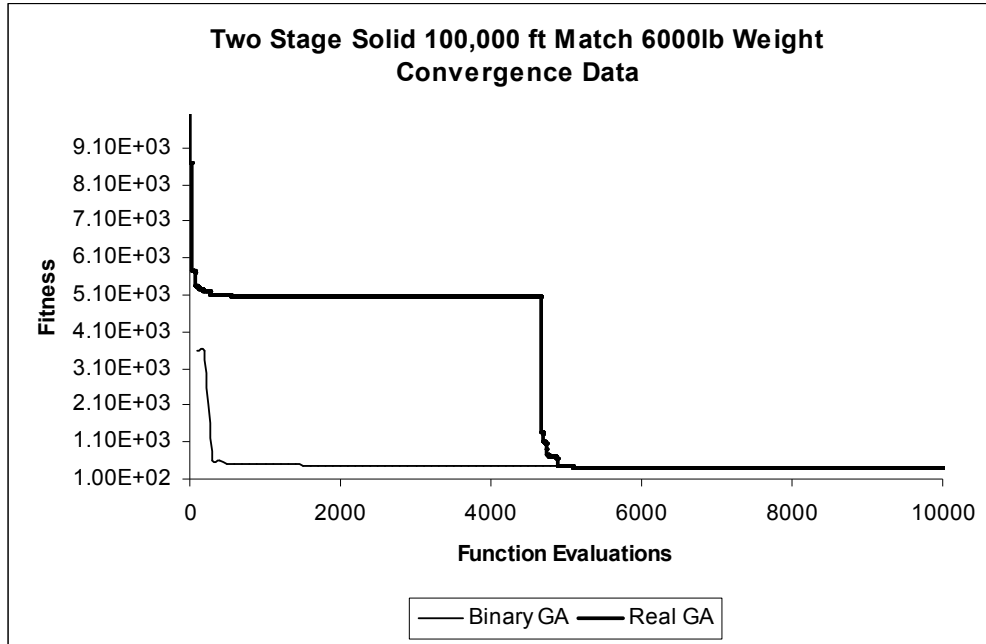


Figure 33. Two stage solid 100,000 ft 6,000 lb convergence history

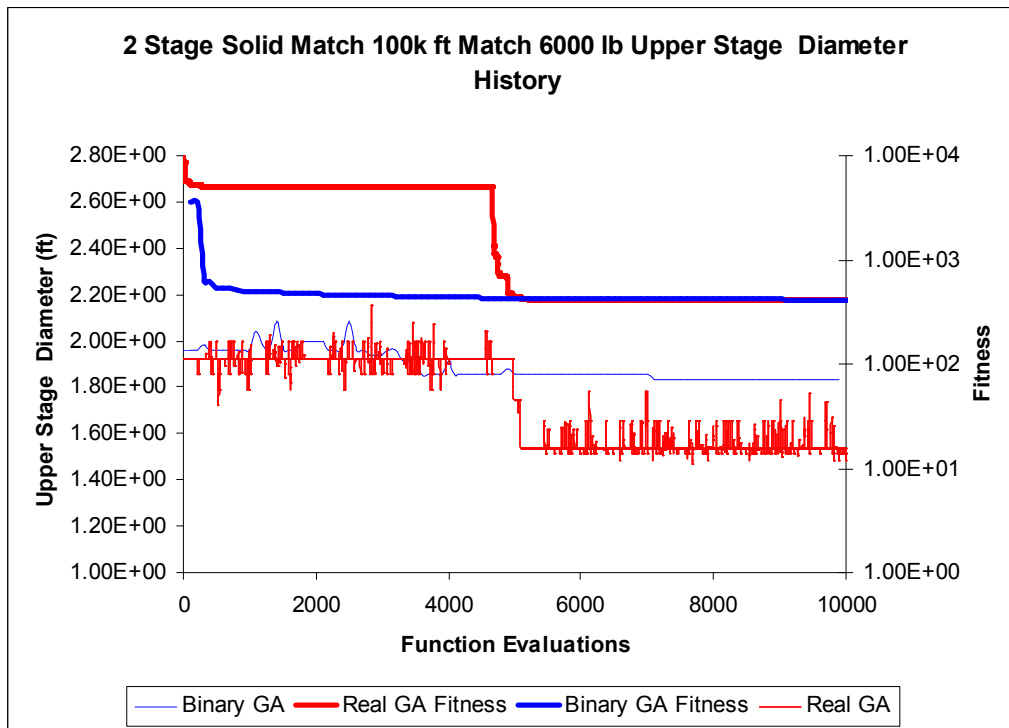


Figure 34. Two Stage Solid Missile Upper Body Diameter Convergence History

The real coded GA was able to produce a slightly more accurate converged solution than its binary counterpart. Figure 34 shows the how both GA's changed the

upper stage missile body diameter as they converged to their final solution. The body diameter is very important when trying to minimize the weight of a missile. For the two stage solid system the body diameter is the only dimensional parameter, all of the other parameters are non-dimensional therefore it is easy to see the overall scale of the missile when looking at the body diameter. For this test the upper stage body diameter could vary between 2.62 and 1.31 feet. The binary GA started with an initial upper stage body diameter of 1.96 ft and converged to a body diameter of 1.83 ft after 10,000 function evaluations. When comparing the overall convergence plot with the body diameter plot there does not seem to be a correlation between the body diameter and the overall convergence for the binary GA. There does seem to be a correlation between the overall convergence and the body diameter for the real GA. After approximately 5000 function evaluations the real GA's body diameter was reduced from 1.92 ft to 1.54 ft, this reduced the weight of the missile and helped to reach a better solution.

Another very important design parameter is the initial launch angle. The launch angle is very important when trying to match a range for a ballistic trajectory. Figure 35 shows the convergence of another GA design variable, initial launch angle. When comparing the overall convergence plot with the launch angle convergence plot it becomes clear that for both the real and binary GA's the launch angle was a large contributor to the overall convergence. The binary GA started off with an initial launch angle of 63.9 degrees and rapidly converged to a launch angle of 45 degrees within the first 1000 function evaluations. The real GA started with an initial launch angle of 85 degrees and fluctuated slightly for the first 5000 function evaluations and then rapidly decreased to 45 degrees after 5000 function evaluations. The max value allowed for the

launch angle was 85 degrees and the minimum value was 45 degrees, therefore better fitness' could possibly have been obtained with a broader range of initial launch angles. The 3D models of the two stage solid designed by the real and binary coded GA's are shown in Figure 36 and Figure 37.

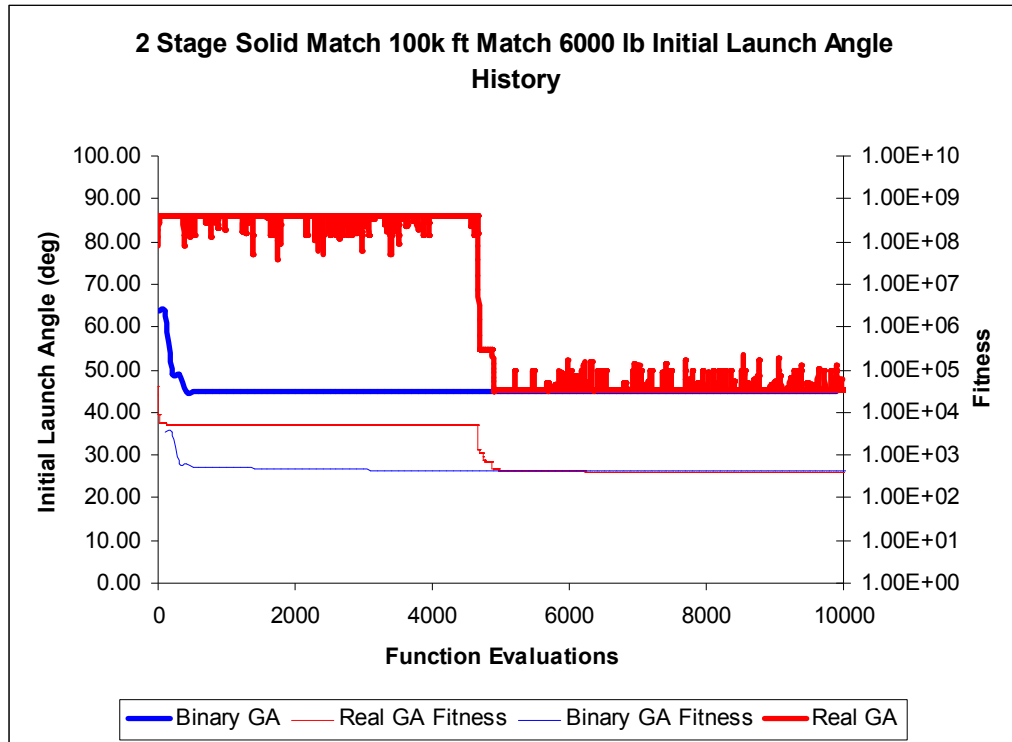


Figure 35. Two Stage Solid Missile Initial Launch Angle Convergence History

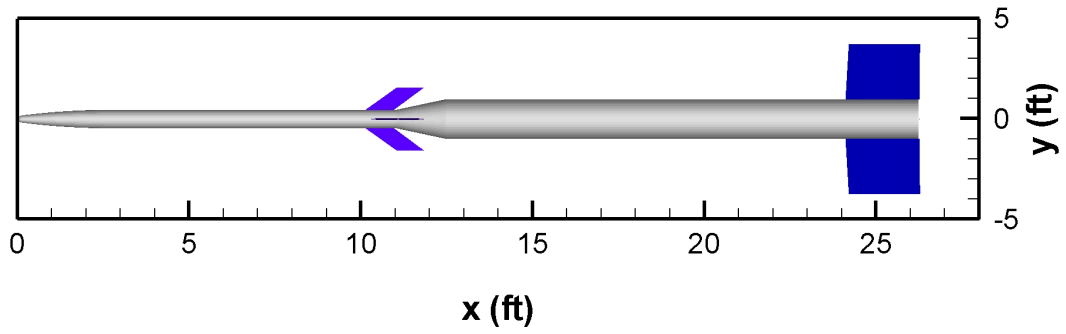


Figure 36. Two Stage Solid 3D Model-Real GA

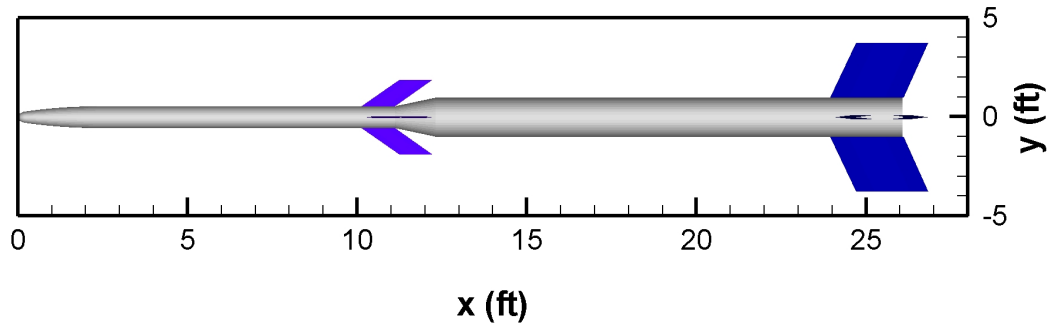


Figure 37. Two Stage Solid 3D Model-Binary GA

5.2.2 TWO STAGE SOLID 250,000 FT 7,000 LB GOAL

For the second test the binary coded genetic algorithm was able to produce a missile that could hit a target 251,144.44 ft down range with a 9,391.07 lb initial take-off weight using 10,000 function evaluations with a converged solution having a fitness of 314.1. This corresponded to a miss distance of 1,144.44 ft and a miss weight of 2,391.07 lb. Given the same inputs the real coded GA was able to achieve a fitness of 5,148 after 10,000 function evaluations. This corresponded to a miss distance of 50,377.97 ft and a miss weight of 1,105.80 lb.

Table 13. Two stage solid 250,000 ft 7,000 lb Final Design Variables

Real GA	Binary GA	Design Variable Definition	
0.5764	0.5333	1	rnose/rbody
2.0296	2.1774	2	lnose/dbody
6.0069	6.1000	3	(1)fuel type
0.4949	0.5000	4	(1)star out R
0.5126	0.5000	5	(1)star inner ratio
9.0962	9.1000	6	(1)number of star points
0.0944	0.0957	7	(1)fillet radius ratio
0.8083	0.9000	8	(1)eps
10.0145	11.1063	9	(1)star point angle deg
0.9567	0.9313	10	(1)fractional nozzle length
0.2905	0.2933	11	(1)Dia throat/Dbody
10.6759	10.1000	12	(1)Fineness Ratio
1.6277	1.8331	13	(1)body dia ft
1.3963	1.2000	14	(1)wing semispan / dbody
1.0155	0.9000	15	(1)wing root chord/dbody
0.9877	0.9814	16	(1)taper ratio
26.1889	40.0000	17	(1)wing LE sweep angle deg
0.9944	1.0000	18	(1)xLEw/lbody
6.1000	6.1000	19	(2)fuel type
0.6000	0.4000	20	(2)star out R
0.5000	0.6000	21	(2)star inner ratio
9.0181	9.0000	22	(2)number of star points
0.1000	0.1000	23	(2)fillet radius ratio
0.8000	0.8667	24	(2)eps
10.0648	10.0000	25	(2)star point angle deg
0.9900	0.9900	26	(2)fractional nozzle length

0.3100	0.3200	27	(2)Dia throat/Dbody
7.0000	7.0000	28	(2)Fineness Ratio
1.9685	1.9685	29	(2)body dia ft
1.2000	1.4000	30	(2)wing semispan / dbody
1.1000	1.1000	31	(2)wing root chord/dbody
0.9843	0.9814	32	(2)taper ratio
29.1717	15.4839	33	(2)wing LE sweep angle deg
0.9800	0.9933	34	(2)xLEw/lbody
2.5789	2.2000	35	time to separate stage 1 sec
4000.1986	4000.0000	36	(1)autopilot delay time sec
0.4000	0.5714	37	(1)auto pilot time constant sec
0.8700	0.6190	38	(1)auto pilot damping
52.6969	59.3333	39	(1)crossover freq hz
3.4299	5.8000	40	(1)pronav gain
4000.0869	4001.0000	41	(12)autopilot delay time sec
0.4120	0.4857	42	(2)auto pilot time constant sec
0.6182	0.8048	43	(2)auto pilot damping
53.3991	50.6667	44	(2)crossover freq hz
4.1287	3.6000	45	(2)pronav gain
86.0000	45.0000	46	initial launch angle deg

When comparing the final converged design variables for the real and binary GA's the main differing variables are the upper stage body diameter, initial launch angle, and both the upper stage(1) and bottom stage(2) wing LE angles as shown above in Table 13. A comparison of the performance of the two GA's is shown in Figure 38.

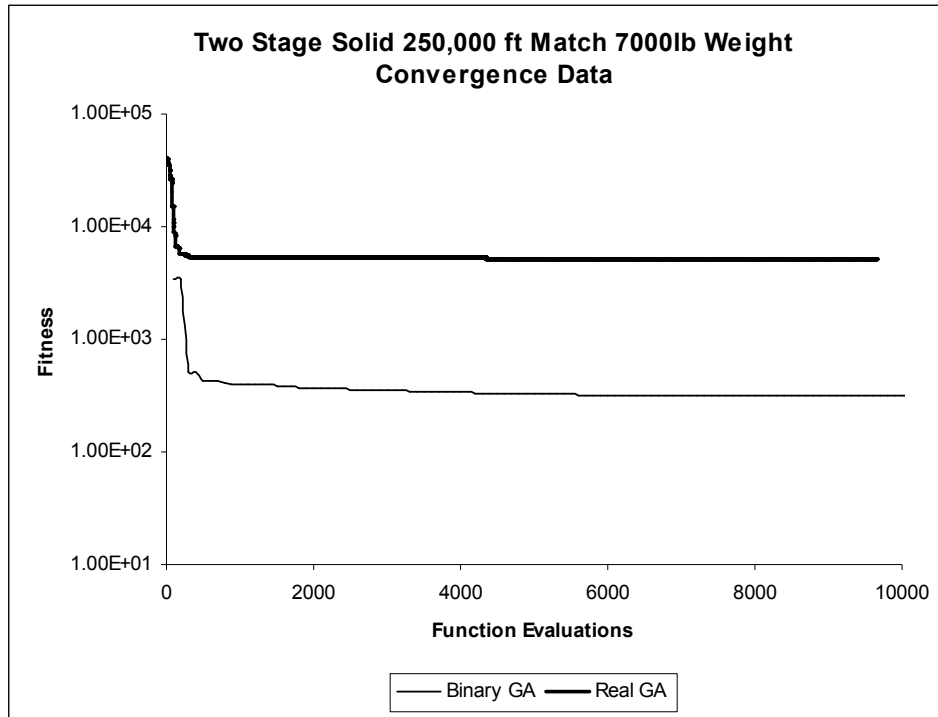


Figure 38. Two stage solid 250,000 ft 7,000 lb convergence history

The binary coded GA was able to produce a much more accurate converged solution than its real counterpart. While the binary GA was able to produce a better overall fitness than the real GA, it was not able to match the weight as accurately as the real GA. The real GA was able to get over 1,000 lbs closer to the desired weight of 7,000 lbs than the binary GA. The weight difference can be accounted for by the body diameter. Figure 39 shows that the real GA was able to converge to a smaller upper stage body diameter than the binary GA.

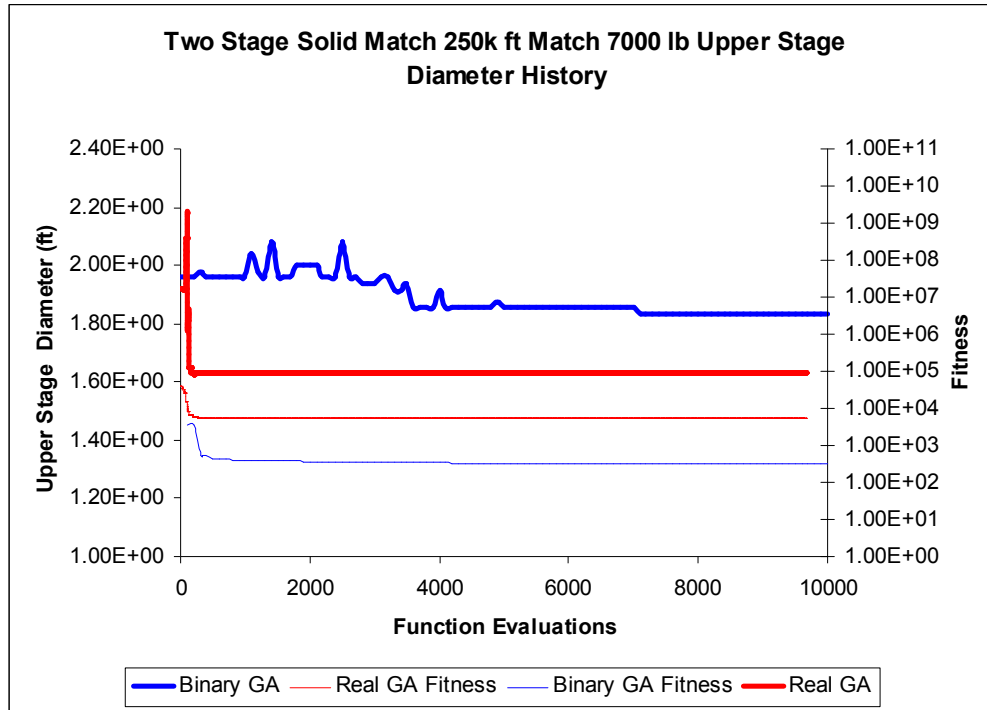


Figure 39. Two Stage Solid Missile Upper Stage Diameter Convergence History

For this test the upper stage body diameter could be allowed to vary between 2.62 and 1.31 feet, the same ranges as the previous two stage test. The binary GA started with an initial upper stage body diameter of 1.96 ft and converged to a body diameter of 1.83 ft after 10,000 function evaluations. When comparing the overall convergence plot with the body diameter plot there does not seem to be a correlation between the body diameter and the overall convergence for the binary GA. There does seem to be a correlation between the overall convergence and the body diameter for the real GA. In the first few hundred function evaluations the real GA's body diameter was reduced from 1.92 ft to 1.63 ft, this reduced the weight of the missile and helped to reach a better solution.

While the weight of the real GA was closer to the desired weight than the binary GA, the binary GA was able to match the range much more accurately than the real GA,

giving the binary GA a converged fitness almost an order of magnitude better than the real GA. When the design parameters are analyzed it becomes clear that the reason for the poor performance of the real GA was the initial launch angle. Figure 40 shows the initial launch angle for both the real and binary GA's plotted against function evaluations.

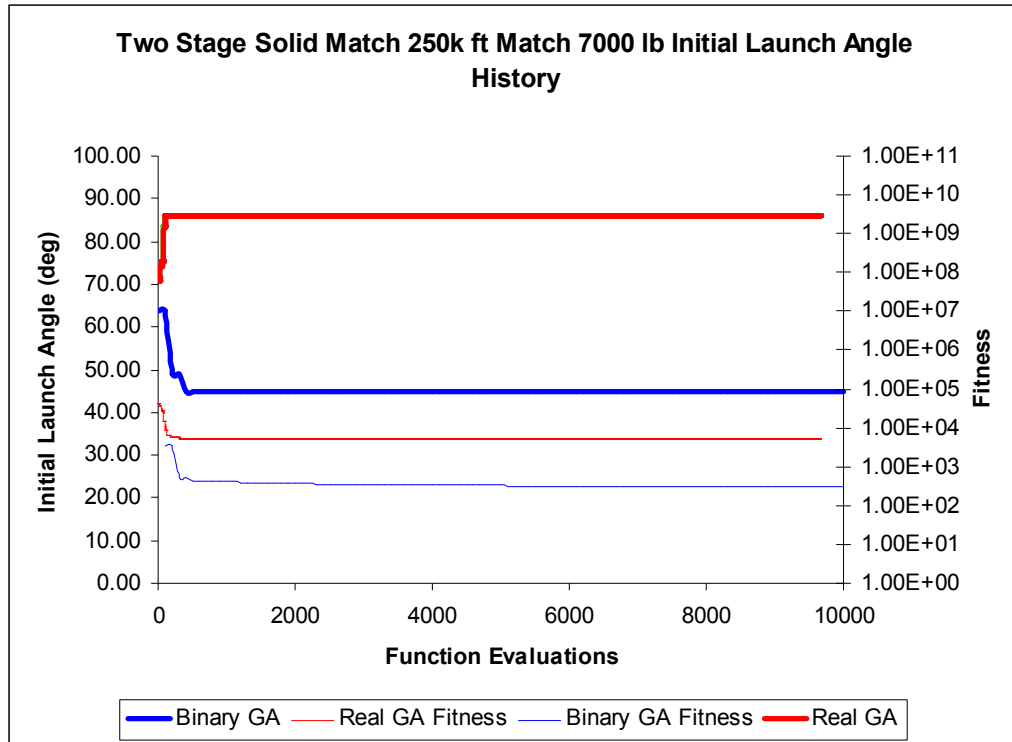


Figure 40. Two Stage Solid Missile Initial Launch Angle Convergence History

The real GA started with an initial launch angle of 75 degrees and quickly rose to 85 degrees, where it remained to the remainder of the function evaluations. This may be due to the fact that the real GA's smaller body diameter missile needed to have a high initial launch angle in order to coast to the target because of the decreased propellant mass associated with a smaller diameter. Any lower launch angle could have come up far short of the target. The binary GA was able to start of with a 65 degree launch angle and it converged to a 45 degree launch angle. The binary GA was able to achieve a better overall fitness by trading off weight for range. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 41 and Figure 42.

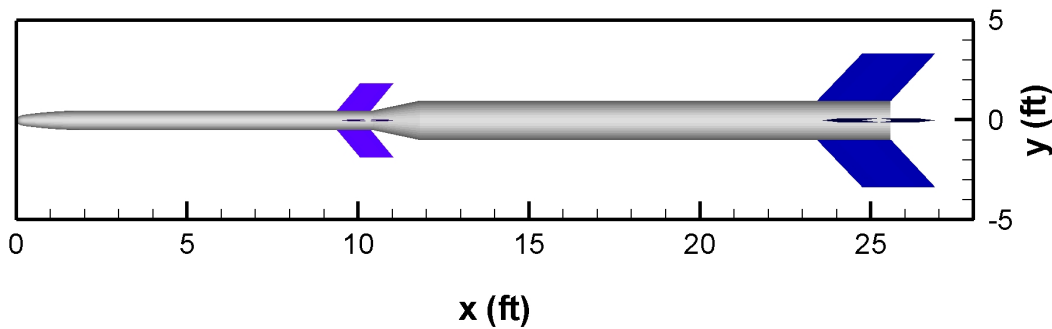


Figure 41. Two Stage Solid 3D Model-Real GA

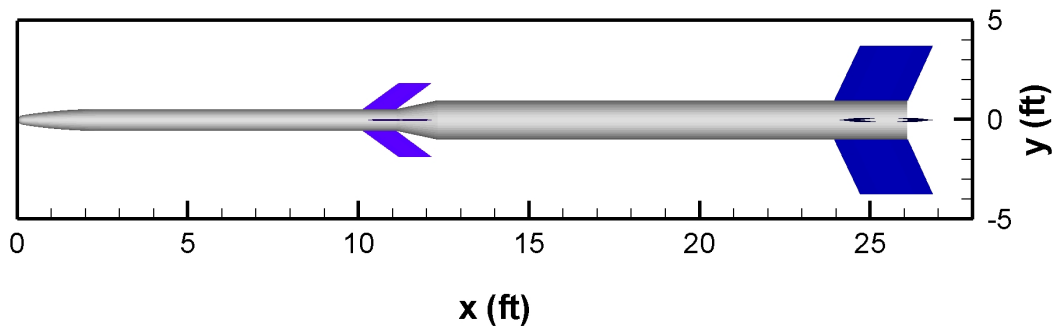


Figure 42. Two Stage Solid 3D Model-Binary GA

5.3 SINGLE STAGE LIQUID MISSILE SYSTEM DESIGN

Two cases were tested in order to compare the binary and real coded GA's for the single stage liquid propellant missile tests: match range 450,000 ft while also matching a 55,000lb initial take-off weight and match range 700,000 ft while also matching a 55,000lb initial take-off weight. Both missile design codes were identical with the exception of the GA's. The GA input file (GANNL.DAT) can be seen in Appendix F for the real coded GA and Appendix G for the binary coded GA. The fitness' for the liquid system were calculated as shown below.

$$Fitness = \frac{abs(Range)}{10.0} \quad Range = abs(DesiredRange - ActualRange)$$

$$Fitness = \frac{abs(Weight - DesiredWeight)}{10.0}$$

A summary of the single stage liquid optimization results is shown in Table 14. For the single stage liquid missile design code the real coded GA was able to converge to a better solution in one of the two tests. The binary GA was able to converge to better solutions in the remaining test however the fitness's were on the same order of magnitude.

Table 14. Single Stage Liquid Optimization Results

GA Fitness Comparison	Real GA	Binary GA	F.E.
1 Stg Liquid			
Match 450kft 55000lb weight	6.97	276.16	10,000
Match 700kft 55000lb weight	94.00	72.03	10,000

5.3.1 SINGLE STAGE LIQUID 450K FT 55K LB WEIGHT GOAL

For the first test the binary coded genetic algorithm was able to produce a missile that could hit a target 450,088.15 ft down range with an initial take-off weight of 57,812.7 lbs using 10,000 function evaluations with a converged solution having a fitness of 276.2. This corresponded to a miss distance of 88.15 ft and a miss weight of 2,812.7 lbs. Given the same inputs the real coded GA was able to achieve a fitness of 6.973 after 10,000 function evaluations. This corresponded to a miss distance of 19.97 ft and a miss weight of 49.76 lbs.

Table 15. Single Stage Liquid 450,000 ft 55,000 lb Final Design Variables

Real GA	Binary GA	Design Variable Definition	
5.1644	5.5677	1	body diameter ft
4.1000	4.1000	2	propellant type
0.6928	0.7000	3	equivalence ratio
2457.9795	2340.1575	4	chamber pressure psi
0.5460	0.7143	5	nose dia ratio
1.9389	1.7333	6	nose length ratio
0.1040	0.1013	7	nozzle throat dia/dbody
20.7429	10.3922	8	nozzle expansion ratio
0.6359	0.7600	9	fractional nozzle length
87.6968	87.6378	10	burn time sec
0.0229	0.0400	11	wing root chord ratio
0.8968	0.8900	12	wing taper ratio
0.0259	0.0457	13	wing b/2 ratio
3.8352	2.1429	14	wing le angle deg
0.5000	0.3095	15	XLEwing/lbody
1.0551	1.1000	16	tail root chord ratio
0.5000	0.5656	17	tail taper ratio
1.0128	1.0667	18	tail b/2 ratio
20.4071	25.5484	19	tail le angle deg
0.9809	0.9786	20	tail x loc
4999.0000	4999.0000	21	autopilot delay time sec
0.6716	0.3094	22	autopilot time const
0.5849	0.9206	23	autopilot damping coeff
40.0672	60.0000	24	cross freq hz
5.5797	4.4194	25	pronav gain
83.0333	83.7419	26	initial launch angle deg

When analyzing the final converged design parameters for both GA's, all of the parameters are very similar except for the missile body diameter, variable 1, the nozzle expansion ratio, variable 8 and variable 19, the tail leading edge angle. Since both GA's converge to different fitness's it is logical that for a match weight goal the diameters would be different, and similarly for a match range goal the tails would also be different for two missiles that fly out to differing ranges. A comparison of the performance of the two GA's is shown in Figure 43.

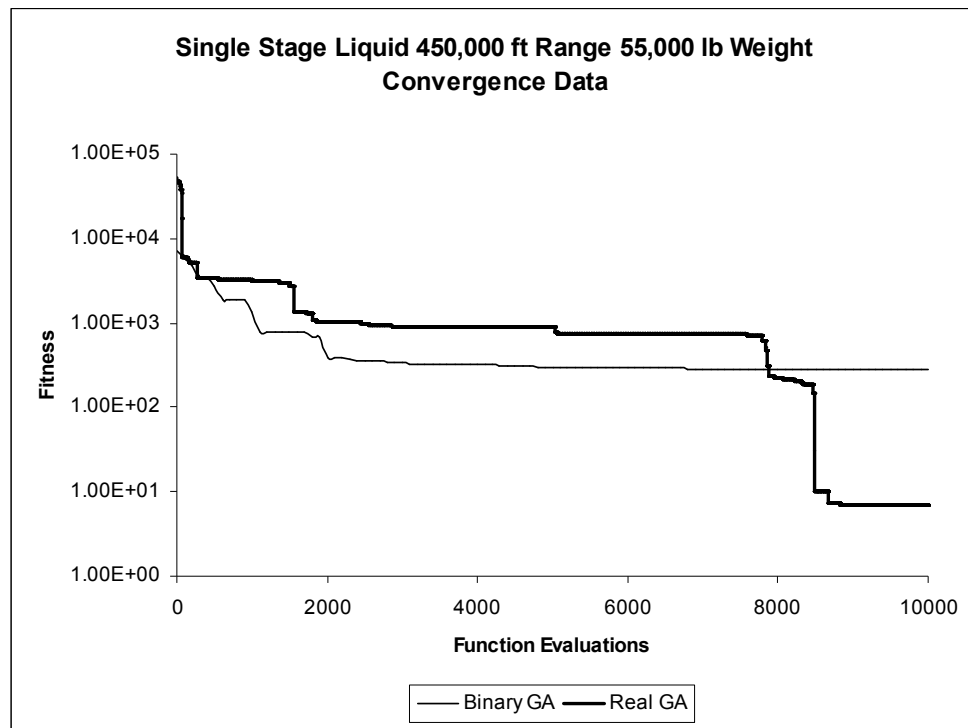


Figure 43. Single Stage Liquid 450,000 ft 55,000 lb Convergence Plot

The real coded GA was able to produce a much more accurate converged solution than its binary counterpart. The weight difference can be accounted for by the body diameter. Figure 44 shows that the real GA was able to converge to a smaller body diameter than the binary GA. For the single stage liquid the body diameter was allowed to vary between 6.6 ft and 5.0 ft. The binary GA started out with a body diameter of 6.19

ft and converged to a 5.52 ft body diameter. The real GA was able to converge to a 5.14 ft body diameter, allowing for a lower initial take-off weight.

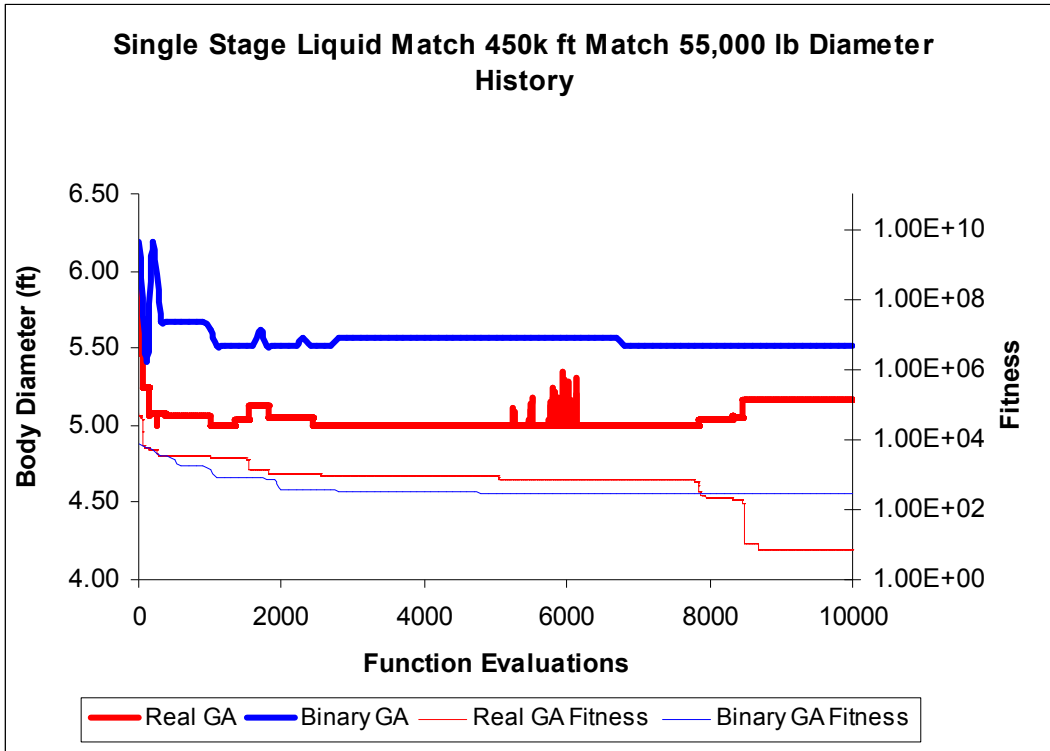


Figure 44. Single Stage Liquid Missile Body Diameter Convergence History

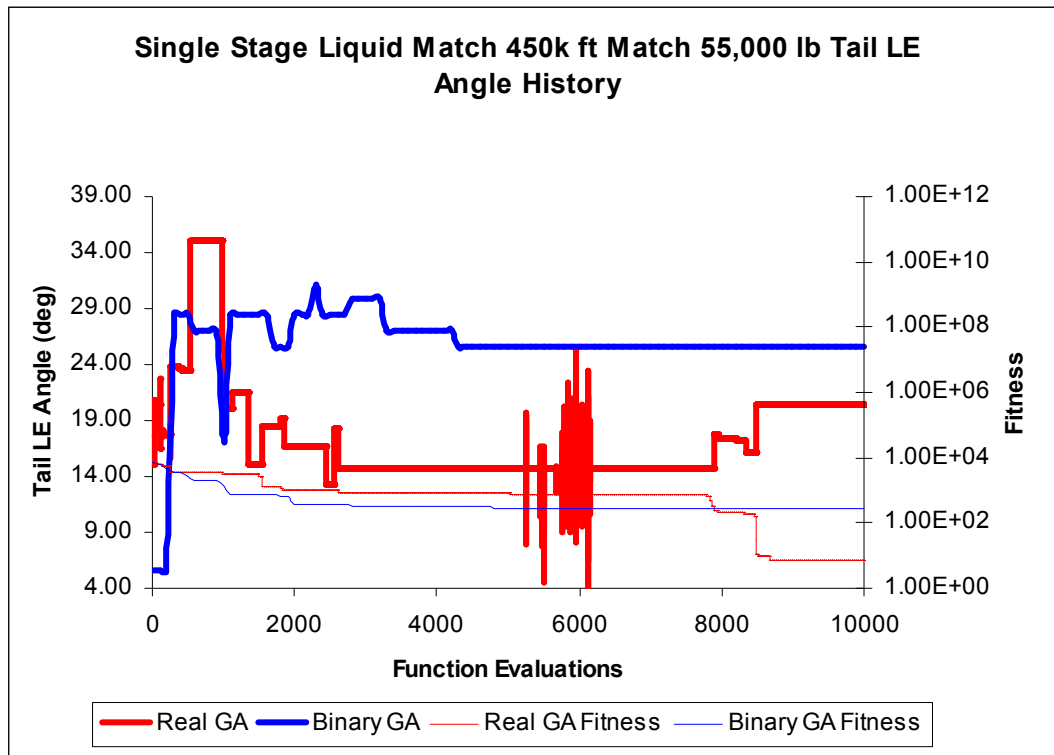


Figure 45. Single Stage Liquid Missile Tail Leading Edge Angle Convergence History

While the diameter is important for the overall convergence, it does not seem to be contributing to the final drop in the fitness for the real coded GA. When the convergence history was analyzed it was discovered that the final decreases in fitness for the real coded GA was due to the tail leading edge angle. The angle was allowed to vary between 44.0 and 0.0 degrees. After 7,000 function evaluations the real GA had converged to a 14.6 degree tail leading edge angle, in the remaining 3,000 function evaluations the real GA increased the angle to 20.6 degrees aided in converging closer to the target giving the real GA the better overall fitness. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 46 and Figure 47.

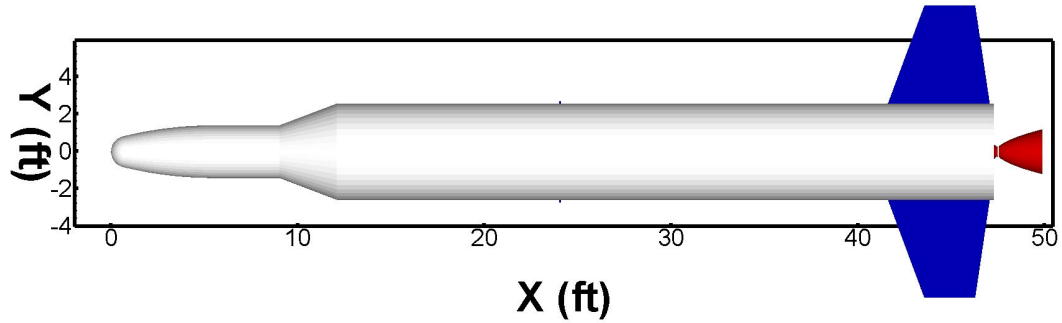


Figure 46. Single Stage Liquid 3D Model-Real GA

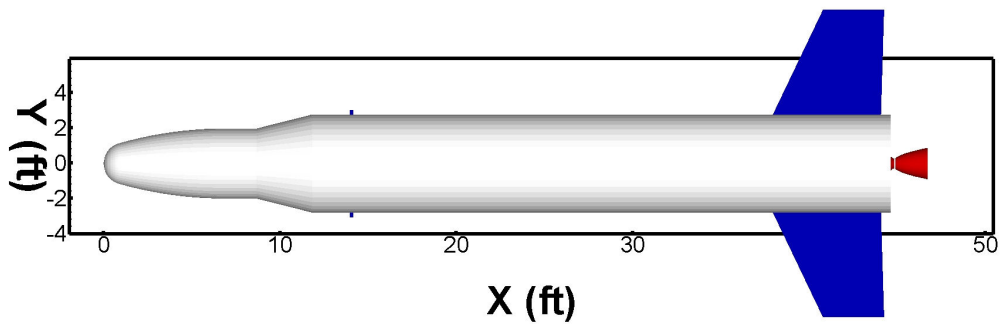


Figure 47. Single Stage Liquid 3D Model-Binary GA

5.3.2 SINGLE STAGE LIQUID 700K FT 55K LB WEIGHT GOAL

For the second test the binary coded genetic algorithm was able to produce a missile that could hit a target 700,417.41 ft down range with an initial take-off weight of 55,302.29 lbs using 10,000 function evaluations with a converged solution having a fitness of 72.03. This corresponded to a miss distance of 417.41 ft and a miss weight of 302.29 lbs. Given the same inputs the real coded GA was able to achieve a fitness of

94.00 after 10,000 function evaluations. This corresponded to a miss distance of 848.21 ft and a miss weight of 91.80 lbs.

Table 16. Single Stage Liquid 700,000 ft 55,000 lb Final Design Variables

Real GA	Binary GA		Design Variable Definition
5.4645	5.0021	1	body diameter ft
4.0000	4.0936	2	propellant type
0.6000	0.6998	3	equivalence ratio
2018.8976	2496.5987	4	chamber pressure psi
0.7143	0.5000	5	nose dia ratio
1.7333	1.9850	6	nose length ratio
0.1173	0.1159	7	nozzle throat dia/dbody
8.5294	21.2879	8	nozzle expansion ratio
0.8200	0.6176	9	fractional nozzle length
73.7795	73.8050	10	burn time sec
0.0333	0.0316	11	wing root chord ratio
0.8900	0.9188	12	wing taper ratio
0.0500	0.0322	13	wing b/2 ratio
2.1429	4.2167	14	wing le angle deg
0.2206	0.3488	15	XLEwing/lbody
1.0000	1.0410	16	tail root chord ratio
0.8048	0.5000	17	tail taper ratio
1.0667	1.0026	18	tail b/2 ratio
17.0323	10.6517	19	tail le angle deg
0.9643	0.9763	20	tail x loc
5000.0000	4999.3174	21	autopilot delay time sec
0.7890	0.4717	22	autopilot time const
0.8318	0.7489	23	autopilot damping coeff
53.5484	52.4772	24	cross freq hz
6.2258	5.3116	25	pronav gain
82.1936	78.0266	26	initial launch angle deg

When analyzing the final converged design parameters for both GA's, all of the parameters are very similar except for the missile body diameter, variable 1, the nozzle expansion ratio, variable 8 and variable 19, the tail leading edge angle. The initial launch angles were also different by a few degrees which could have contributed to the increased accuracy for the match range goal. A comparison of the performance of the two GA's is shown in Figure 48.

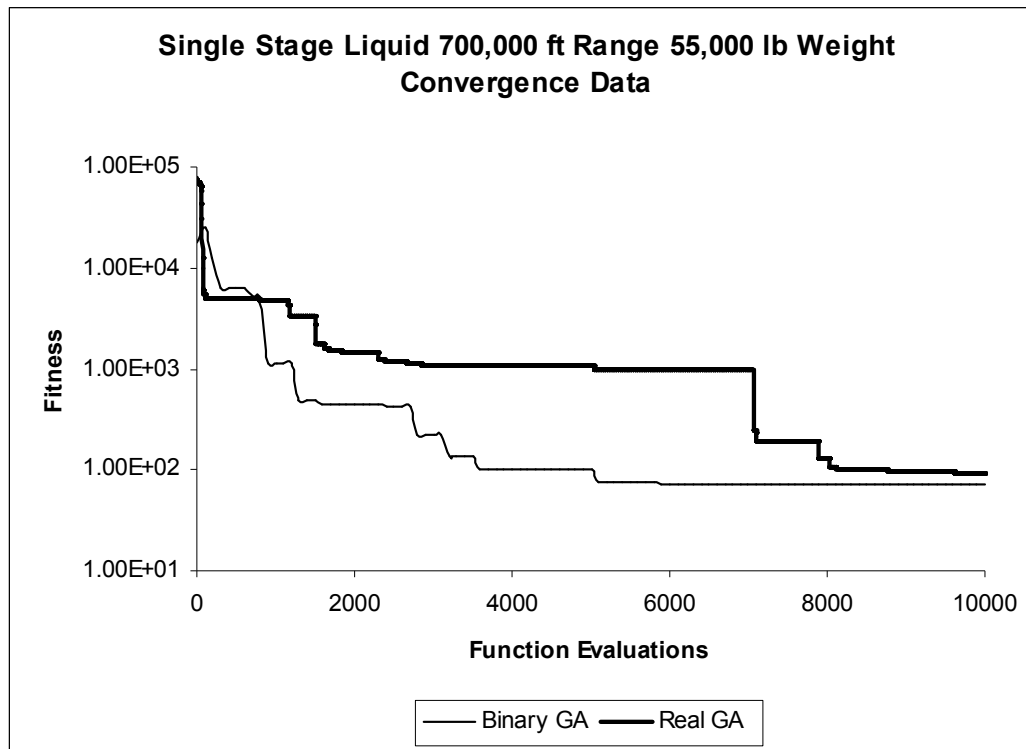


Figure 48. Single Stage Liquid 700,000 ft 55,000 lb Convergence Plot

The binary coded GA was able to produce a slightly more accurate converged solution than its real counterpart. The real GA was able to converge to a lower weight than the binary GA, however it missed the target by a greater distance, therefore the binary GA converged to a lower overall fitness. The real GA's ability to converge to a lower weight can be account for by its body diameter convergence as shown in Figure 49. The limits allowed for the body diameter were a maximum value of 6.6 ft with a 5.0 ft minimum. The real GA started off with an initial body diameter of 6.05 ft and quickly converged to the minimum allowable value for the diameter of 5.0 ft, in order to minimize the weight. The binary GA also started off with a body diameter above 6.0 ft, however the binary GA's body diameter only converged to 5.50 ft.

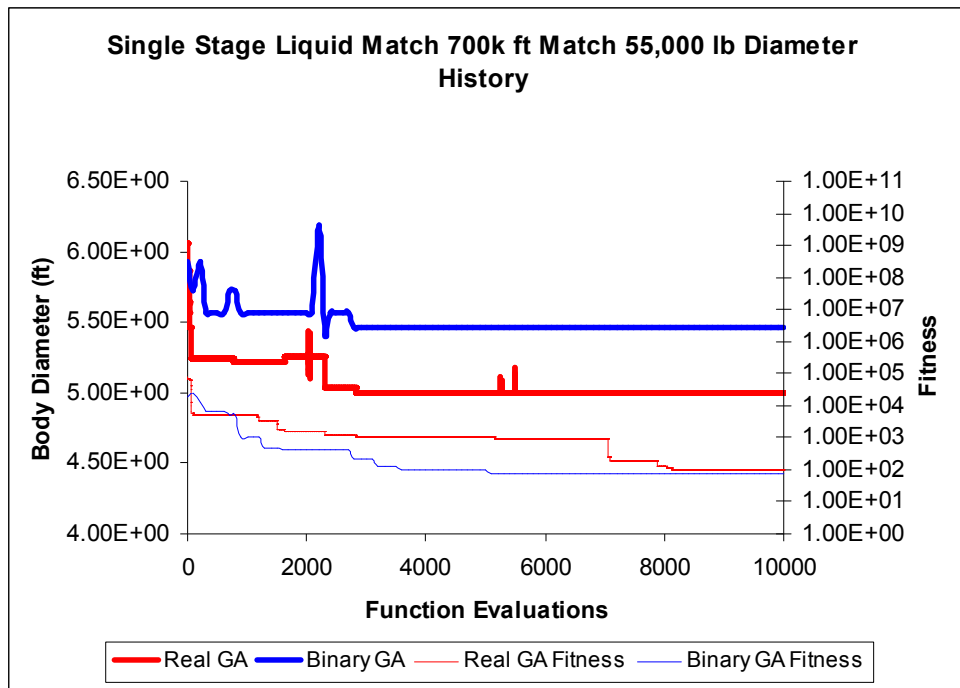


Figure 49. Single Stage Liquid Missile Body Diameter Convergence History

Both GA's body diameters were converged after only a few thousand function evaluations. The binary GA's overall fitness followed the body diameter convergence, the real GA on the other hand continued to converge well after the body diameter had converged to the minimum allowable value. The second design variable shown to play an important role in the convergence of this case was the burn time. A plot of the burn time versus function evaluations is shown in Figure 50.

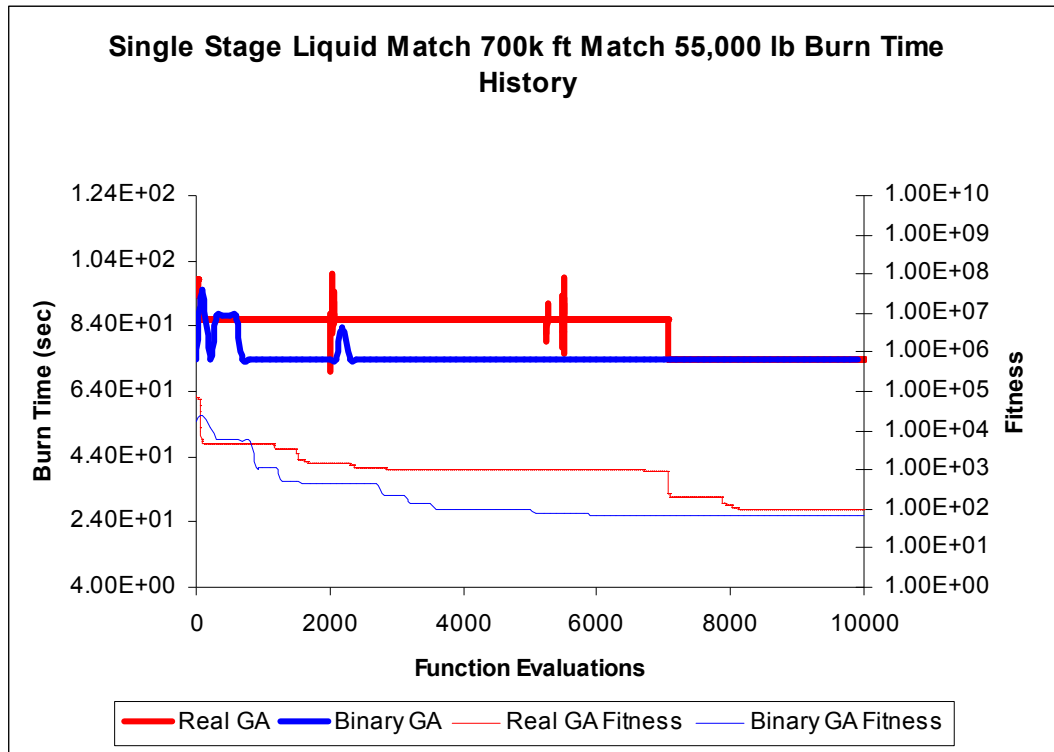


Figure 50. Single Stage Liquid Missile Burn Time Convergence History

The burn time was allowed to vary between 150 and 70 seconds for the liquid missile systems. The initial burn times for both GA's was over 80 seconds. The binary GA converged to a burn time of 73.8 seconds in just over 2,000 function evaluations. The real GA converged to the same burn time, however it took just over 7,000 function evaluations. Three dimensional models of the single stage solid designed by the real and binary coded GA's are shown in Figure 51 and Figure 52.

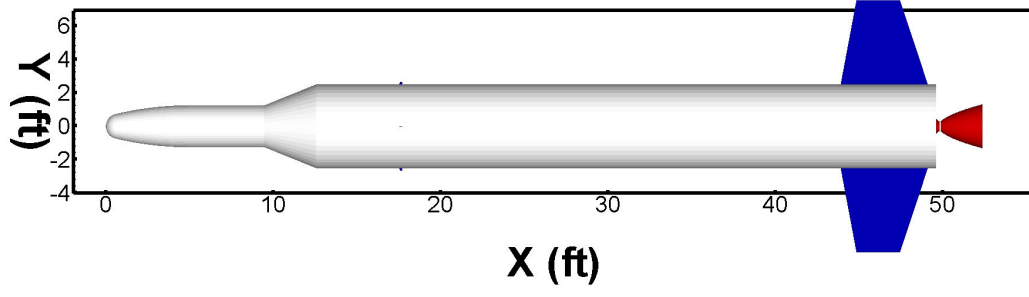


Figure 51. Single Stage Liquid 3D Model-Real GA

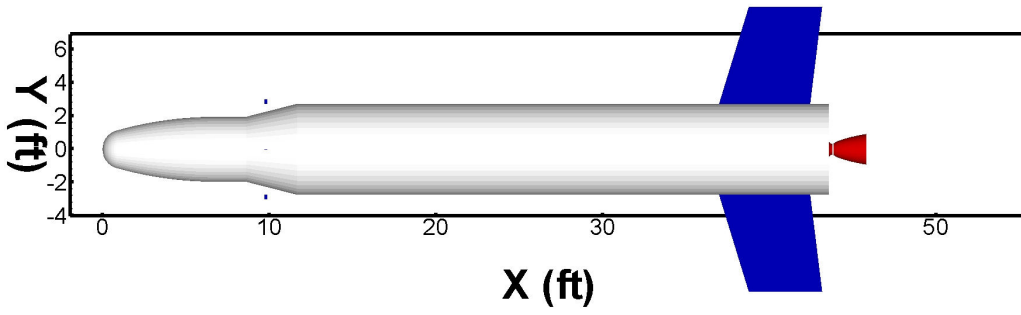


Figure 52. Single Stage Liquid 3D Model-Binary GA

6 REAL CODED GA CONVERGENCE TESTING REVISITED

The GA parameters (mutation rate, mutation amount, population size etc) for both GA's were held constant throughout all of the comparison tests analyzed in Chapter 5. The initial convergence testing for the real coded GA demonstrated its ability to converge to accurate solutions very rapidly for single goal cases. The same GA parameters were used for the two goal cases discussed in Chapter 5. In order to get a better understanding of the effect of a second goal on the mutation parameters, a second set of convergence testing for the real coded GA was conducted.

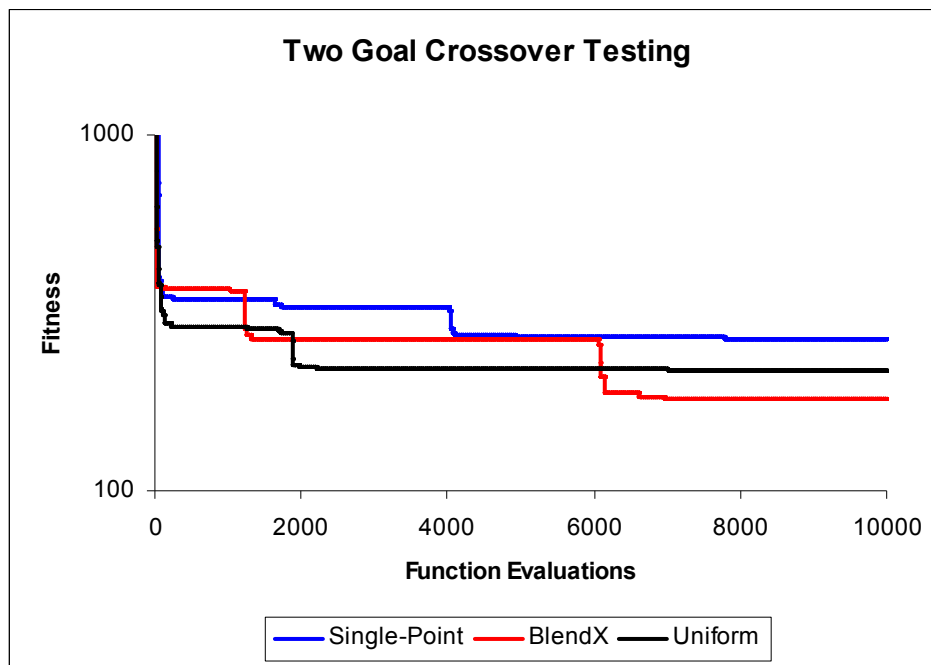


Figure 53. Two Goal Crossover Convergence History

For the two goal convergence tests the single stage solid code was used, with goals of match range 150,000 ft, and a match weight of 1,500 lbs. BlendX crossover was

able to achieve the best fitness. The next GA parameter tested was the mutation operators, mutation rate and mutation amount. Table 17 shows the final fitness' achieved for the various mutation operator combinations. It should be noted that for the two goal case the best operator combination was not Test_3 as it was in the 1 goal test.

Table 17. Two Goal Mutation Testing Results

2 Goal Mutation Testing			
Mutation Operators	Mutation Rate	Mutation Amount	Fitness
Test_1	1.00	0.05	190.49
Test_2	0.20	0.10	181.33
Test_3	0.05	1.00	288.03
Test_4	0.50	0.05	190.25

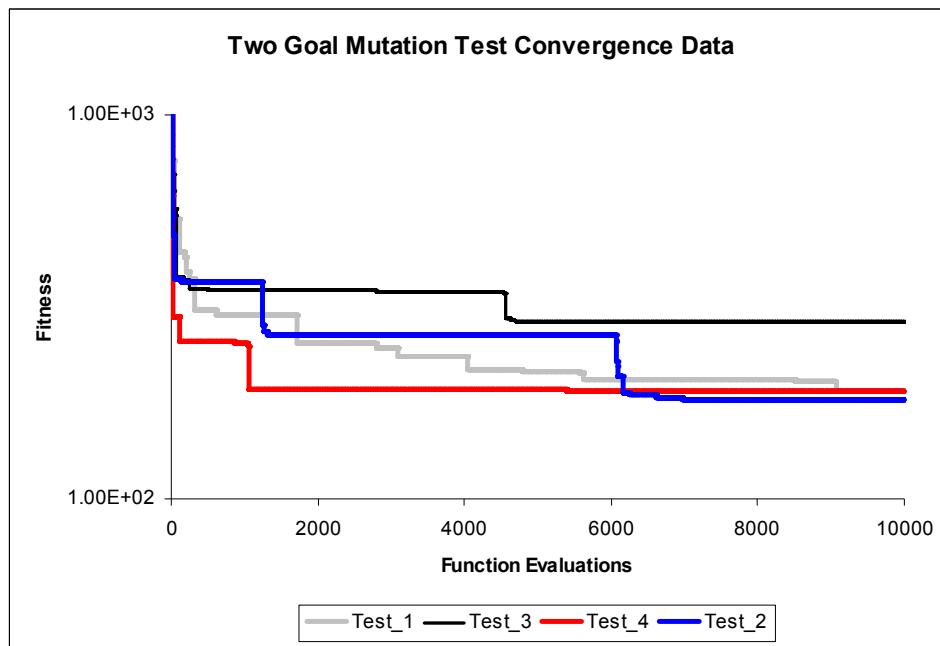


Figure 54. Two Goal Mutation Testing Convergence History

Figure 54 shows the overall convergence of the two goal mutation test for the single stage solid. The operator combination used in the comparisons in Chapter 5 was Test_3, which was the worst performer for this test. The third and final test conducted was for population size. The same population sizes were tested for the two goal test that were tested in Chapter 4 for the one goal test.

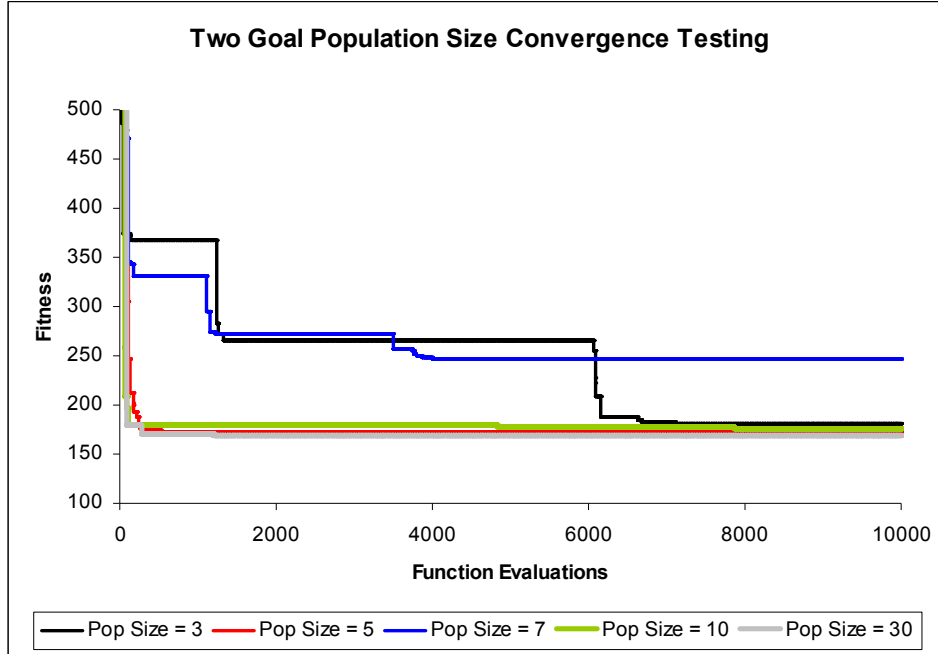


Figure 55. Two Goal Population Test Convergence History

Figure 55 shows that a population size of 30 was able to converge to the best fitness for the two goal case. The two goal testing demonstrates how sensitive the real coded GA is to minor changes in any of the GA parameters. Table 18 shows all of the GA parameters with their achieved fitness values for the one goal and two goal tests.

Table 18. One and Two goal GA Parameter Comparison

	One Goal			Two Goal		
Crossover Testing	Crossover Type	Fitness		Crossover Type	Fitness	
Test_1	Blend X	2.10E-07		Blend X	181.330895	
Test_2	Single-Point	9.85E-06		Single-Point	266.909582	
Test_3	Uniform	4.23E-06		Uniform	216.382506	
Mutation Testing	One Goal			Two Goal		
Mutation Operators	Mutation Rate	Mutation Amount	Fitness	Mutation Rate	Mutation Amount	Fitness
Test_1	1.00	0.05	3.98E-06	1.00	0.05	190.49
Test_2	0.20	0.10	6.54E-09	0.20	0.10	181.33
Test_3	0.05	1.00	8.03E-12	0.05	1.00	288.03
Test_4	0.50	0.05	4.26E-06	0.50	0.05	190.25
	One Goal			Two Goal		
Population Testing	Population Size	Fitness		Population Size	Fitness	
Test_1	3	8.03E-12		3	181.330895	
Test_2	5	3.79E-06		5	172.961289	
Test_3	7	6.52E-11		7	246.371314	
Test_4	10	2.40E-06		10	176.498353	
Test_5	30	6.56E-11		30	169.005173	

Due to the differences in the mutation operators and the population size it was deemed necessary to re-run all of the binary vs real comparisons from Chapter 5 with the new GA parameters to see if the real GA would produce different results. All of the re-run cases achieved similar fitness to the first runs, with the exception of the two stage solid. The two stage solid using the new parameters was able to achieve a better fitness than the binary GA.

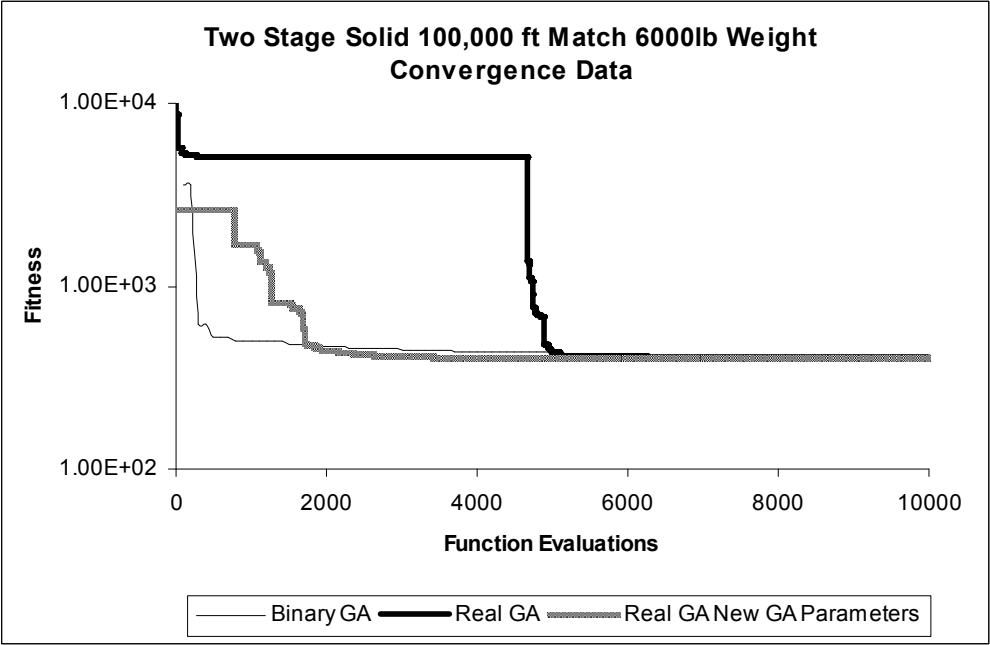


Figure 56. Two Stage Solid Match 100k ft 6k lb Re-Run

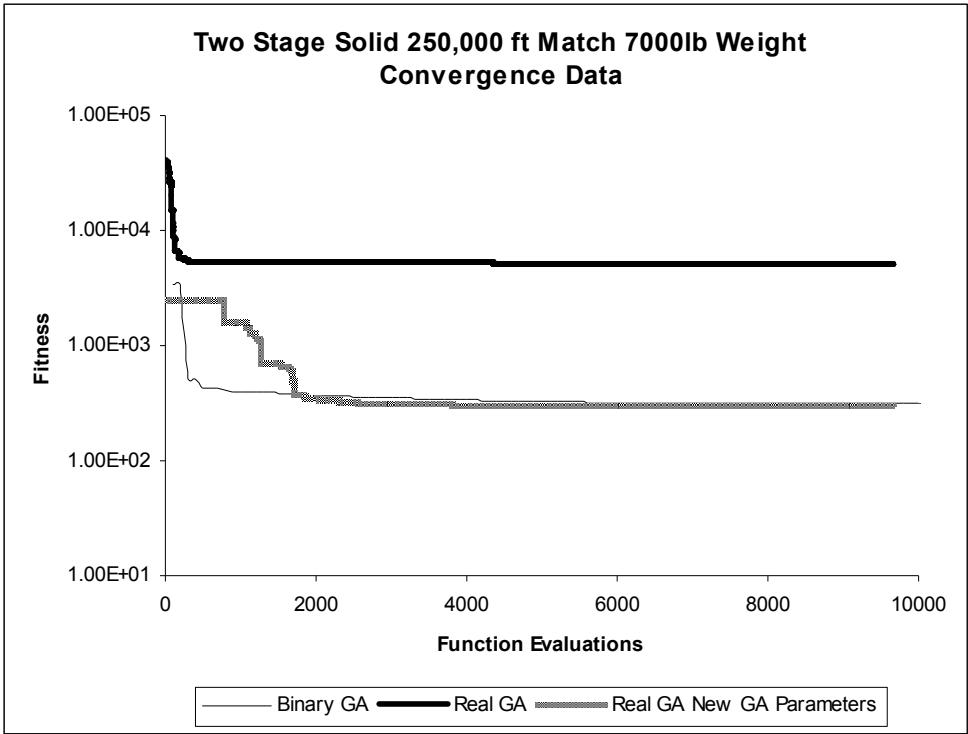


Figure 57. Two Stage Solid Match 250k ft 7k lb Re-Run

For both tests, the real GA was able to achieve a better fitness than the initial real GA and the binary GA. The remaining five tests for the single stage solid and single stage liquid produced similar results to the initial tests.

7 CONCLUSIONS AND RECOMMENDATIONS

A real coded genetic algorithm has been written from first principles. The real GA operators have been tested to find an optimal set of parameters for quick convergence with minimal convergence on local minimums. The real coded GA was then coupled with three different design codes, a single stage solid missile system design code, two stage solid missile system design code and a single stage liquid missile system design code. Three tests were performed for the single stage solid code while two tests were conducted for both the two stage solid code and the single stage liquid code giving a total of seven test cases. For each of the seven tested cases both the real and binary GA's were run for 10,000 function evaluations in order to compare the converged fitness's. The real coded GA demonstrated that it is a viable optimization tool when compared to the already robust binary GA. Of the seven comparative tests, the real GA was able to converge to a better fitness in two tests. Table 19 shows an overview of the final fitness's achieved by both the real and binary GA's for the seven cases tested.

Table 19. Overall Final Fitness Comparison

GA Fitness Comparison	Real GA	Binary GA	F.E.
1 Stg Solid			
Match 100kft 2500lb weight	50.61	46.69	10,000
Match 350kft 4000lb weight	48.67	3.98	10,000
Match 2500 lbs 240 sec	16.65	16.61	10,000
2 Stg Solid			
match 100k ft 6000lbm weight	408.6	414.1	10,000
match 250k ft 7000lbm weight	5148	314.1	10,000
1 Stg Liquid			
Match 450kft 55000lb weight	6.97	276.2	10,000
Match 700kft 55000lb weight	94.0	72.03	10,000

More research needs to be conducted on the GA parameters themselves to determine either an overall optimum set of GA parameters for the real coded GA, or a way to vary all of the parameters throughout the GA run to account for the differing parameters needed for differing problems. A second GA could be used to optimize the GA operators to get a set of operators that is truly optimized over a wide variety of tests.

The steady state real coded GA was able to beat the binary GA for simple 1 goal tests cases. For these tests the real coded GA was able to very rapidly converge to much more accurate fitness than the binary GA. This is due to the fact that the steady state GA did not have to waste function evaluations on bad members. The steady state real GA has proven that it is very effective at optimizing a system that is fairly well known, however for a very complex unknown system the generational binary GA with its large population size is better.

While the binary GA converged to better fitness's for 5 of the 7 tests conducted for this study, the real GA can still be much more effective than the binary GA for more

complex problems involving 60 or more design variables because of the bit limitation inherent with the binary GA.

The real GA is inherently better than the binary GA for more complex optimizations because the real coded GA has no limit on resolution other than what a double precision real number can handle. Also since the real-coded GA does not have to convert real numbers into a single binary bit string it is not susceptible to hamming cliffs therefore making more complex optimizations with more design parameters and finer resolutions possible. Because the real-coded GA is operating directly on the design parameters rather than bits of design parameters it is much easier to gain an understanding of how the GA is modifying the parameters. This could allow the user to track which parameters are affecting the convergence the most and select them for more or less mutation throughout the optimization to increase the convergence. This would be much more difficult for the binary GA because all of the design parameters are represented by a single bit string.

In the future a real coded generational GA could be written and coupled with the real coded steady state GA. The generational GA could be used at the beginning of the optimization, and after a set number of generations the GA could be switched to steady state to home in on a very accurate fitness, therefore using the best characteristics of both types of genetic algorithms. Other types of population based optimizers should also be researched such as particle swarm optimization (PSO) and compared to both binary and real genetic algorithms for aerospace optimization applications.

REFERENCES

1. Karr, C.L., Freeman, L.M., and Meredith, D.L., "Genetic Algorithm based Fuzzy Control of Spacecraft Autonomous Rendezvous," NASA Marshall Space Flight Center, Fifth Conference on Artificial Intelligence for Space Applications, 1990.
2. Krishnakumar, K., Goldberg, D.E., "Control System Optimization Using Genetic Algorithms", Journal of Guidance, Control, and Dynamics, Vol. 15, No. 3, May-June 1992.
3. Krishnakumar, K., Goldberg, D.E., "Control System Optimization Using Genetic Algorithms", Journal of Guidance, Control, and Dynamics, Vol. 15, No. 3, May-June 1992.
4. Perhinschi, M.G., "A Modified Genetic Algorithm for the Design of Autonomous Helicopter Control System," AIAA-97-3630, Presented at the AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, August 1997.
5. Mondoloni, S., "A Genetic Algorithm for Determining Optimal Flight Trajectories", AIAA Paper 98-4476, AIAA Guidance, Navigation, and Control Conference and Exhibit, August 1998.
6. Anderson, M.B., "Using Pareto Genetic Algorithms for Preliminary Subsonic Wing Design", AIAA Paper 96-4023, presented at the 6th AIAA/NASA/USAF Multidisciplinary Analysis and Optimization Symposium, Bellevue, WA, September 1996.
7. Perez, R.E., Chung, J., Behdinan, K., "Aircraft Conceptual Design Using Genetic Algorithms", AIAA Paper 2000-4938, Presented at the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 2000.
8. Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., "Design of an Air to Air Interceptor Using Genetic Algorithms", AIAA Paper 99-4081, presented at the 1999 AIAA Guidance, Navigation, and Control Conference, Portland, OR, August 1999.
9. Anderson, M.B., Burkhalter, J.E., and Jenkins, R.M., "Intelligent Systems Approach to Designing an Interceptor to Defeat Highly Maneuverable Targets", AIAA Paper 2001-1123, presented at the 39th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2001.
10. Chernyavsky, B., Stepanov, V., Rasheed, K., Blaize, M., and Knight, D., "3-D Hypersonic Inlet Optimization Using a Genetic Algorithm", AIAA Paper 98-3582, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 1998.

11. Torella, G., Blasi, L., "The Optimization of Gas Turbine Engine Design by Genetic Algorithms", AIAA Paper 2000-3710, 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 2000.
12. J.E. Burkhalter, R.M. Jenkins, and R.J. Hartfield, M. B. Anderson, G.A. Sanders, "Missile Systems Design Optimization Using Genetic Algorithms," AIAA Paper 2002-5173, Classified Missile Systems Conference, Monterey, CA, November, 2002
13. Hartfield, Roy J., Jenkins, Rhonald M., Burkhalter, John E., "Ramjet Powered Missile Design Using a Genetic Algorithm," AIAA 2004-0451, presented at the forty-second AIAA Aerospace Sciences Meeting, Reno NV, January 5-8, 2004.
14. Jenkins, Rhonald M., Hartfield, Roy J., and Burkhalter, John E., "Optimizing a Solid Rocket Motor Boosted Ramjet Powered Missile Using a Genetic Algorithm", AIAA 2005-3507 presented at the Forty First AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Tucson, AZ, July 10-13, 2005.
15. Riddle, David B., "Design Tool Development for Liquid Propellant Missile System," MS Thesis, Auburn University, May 10, 2007.
16. Burger, Christoph and Hartfield, Roy J., "Propeller Performance Optimization using Vortex Lattice Theory and a Genetic Algorithm", AIAA-2006-1067, presented at the Forty-Fourth Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan 9-12, 2006.
17. Vukovic, S., Sopta, L., "Binary Coded and Real Coded Genetic Algorithm in Pipeline Flow", AMS Paper 35-42, 7th International Applied Mathematical Communications Conference, 1998
18. Elliot, L., Ingham, D. B., Kyne, K.G., "A Real Coded Genetic Algorithm for the Optimization of Reaction Rate Parameters for Chemical Kinetic Modelling in a Perfectly Stirred Reactor", Proceeding of Genetic and Evolutionary Computational Conference, New York, 2002
19. Sugala, S. V., Bhattacharaya, P.K., "Real Coded Genetic Algorithm for Optimization of Pervaporation Process Parameters for Removal of Volatile Organics from Water", Ind. Eng. Chem. Res. 2003, Volume 42, No.13, 3118-3128, November 2003
20. Siddal, J.N., Optimal Engineering Design: Principles and Applications, Merzell Dekker Inc., New York, NY, 1982.
21. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 29.3: The simplex algorithm, pp.790–804.
22. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press; Reprint edition 1992 (originally published in 1975)
23. Anderson, M.,B., "Users Manual for IMPROVE[®] Version 3.0", Sverdrup Technology Inc./TEAS Group

24. Unsal , E. , Dane , J. H. , and Dozier , G. V. (2005). "A Genetic Algorithm for Predicting Pore Geometry Based on Air Permeability Measurements" ,*The Vadose Zone Journal*, 4:389-397 (2005), Soil Science Society of America, Madison, WI
25. Dozier , G. , Cunningham , H. , Britt , W. , and Zhang , F. (2004). "Distributed Constraint Satisfaction , Restricted Recombination , and Hybrid Genetic Search ," *The Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO-2004)* ,LNCS pp. 1078-1087 , June 2004 , Seattle , WA. Springer
26. G. Dozier, A. Homaifar, E. Tunstel, and D. Battle, "An Introduction to Evolutionary Computation" (Chapter 17), *Intelligent Control Systems Using Soft Computing Methodologies*, A. Zilouchian & M. Jamshidi (Eds.), pp. 365-380, CRC press
27. Eshelman, L.J., Schaffer, J.D., "Foundations of Genetic Algorithms 2", Real Coded Genetic Algorithms and Interval Schemata, 5-17, San Mateo, CA, 1992
28. G. Dozier, A. Homaifar, E. Tunstel, and D. Battle, "An Introduction to Evolutionary Computation" (Chapter 17), *Intelligent Control Systems Using Soft Computing Methodologies*, A. Zilouchian & M. Jamshidi (Eds.), pp. 365-380, CRC press.
29. L. J. Eshelman and J. D. Schaffer, "*Real-coded genetic algorithms and interval-schemata*," in *Foundations of Genetic Algorithms-2*. San Mateo, CA: Morgan Kaufman, 1993, pp. 187--202.
30. Fogel, David B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, 1995.
31. Vavak, F., Fogarty, T. *Comparison of Steady State and Generational Genetic Algorithms for Use in Non-Stationary Environments*, IEEE Press, 1996.
32. I. Rechenberg, *Evolutionstrategie*, Frommann-Hozboog, Stuttgart, 1973
33. Hartfield, Roy J., Burkhalter, John E., Jenkins, Rhonald M., Dyer, John, and McDavid, Brian., "Genetic Algorithm Developments for Multiple Stage Missile Analysis", submitted to Missile and Space Intelligence Center Redstone Arsenal, Alabama 35898, February 2008, Reference Contract No. PAN 70084-07.

APPENDIX A: Real Coded GA Source File

```

Subroutine GAMAIN
implicit double precision (a-h,o-z)
character names*14
parameter (npr=200,mstr=750,mpop=400,npls=20)
dimension xmax(npr),xmin(npr),resolution(npr),names(npr)
dimension niche_par(npr),var(mstr,mpop),storevar(mstr)
dimension ans(npls),fitness(npls,mpop),totalfit(mpop)
dimension child(mstr),genchild(mstr,mpop)
integer iBestParent(mpop)
real mue
integer iarray(mstr*2)
character answ*1,fname*50,filename*50
logical micro,elitist,restart,maximize,disrupt,pareto,
& creep,remove_dup,uniform,check_avg,steady_state,
& niche,phenotype,niche_par
common/pass/yy(1,15,30),ys(10,29)
common/draw2/lastgen,mempops1,maxgen1,member1
c
      do i=1,40
        filename(i:i)=' '
      enddo
      fname='gannl.dat'
28      format(/,1x,'input the parameter (min/max/res)',
&        ' filename, def= ',a12,' > ',,$)
        filename=fname
        open(unit=43,file=filename,status='old')
        write(*,*)'input file successfully opened'
        write(*,*)

          read(43,*) ipopulation
          read(43,*) iuniform
          read(43,*) iblend
          read(43,*) isnglpoint
          read(43,*) ivar_mutation
          read(43,*) iK_check
          read(43,*) xmutation_rate
          read(43,*) xmutation_amount
          read(43,*) ngoals

          ys(4,5)=dble(float(ngoals))

        read(43,*) no_para

          do j=1,no_para
            read(43,*) names(j),xmax(j),xmin(j)
          enddo

        read(43,*) mempops
        read(43,*) maxgen

```

```

        close(43)
        icount=0

        mue=xmutation_rate
        sigma=xmutation_amount

open(unit=41,file='ga_out1.dat',status='unknown')
open(unit=42,file='population.dat',status='unknown')
open(unit=45,file='mutation_check.dat',status='unknown')
open(unit=56,file='Pop_Tracking.dat',status='unknown')

write(45,*)'Gen#,Amountbetter,NewSigma'
write(56,712)(names(i),i=1,no_para),('fitness')

k=0          !number of func evals b4 you check 1/5 rule
ibetter=0

        write(*,*)'***STEADY STATE REAL GA SELECTED***'

!   Generate Initial Population
T1=1.0d0
idum=SECNDS(T1)
do j=1,mempops
    do jk=1,no_para
        random=ran1(idum)
        var(jk,j)=(xmax(jk)-xmin(jk))*random+xmin(jk)
    end do
enddo

!   Complete Function evaluations for first generation
do i=1,mempops

    do j=1,no_para
        storevar(j)=var(j,i)
    enddo

    do jk=1,no_para
        ys(9,jk)=storevar(jk)
    enddo
    ys(4,2)=i
    call setup

    do j=1,npls
        fitness(j,i)=sngl(ys(7,j))
    enddo

end do

if(ipopulation.eq.-1) then
    write(42,*)'Generation#',1
else

```

```

        write(42,*)'Population check must be set to true in gannl.dat
& to record population data '
    end if

    do j=1,mempops
        iarray(j)=j
    end do

    if(ipopulation.eq.-1) then
        write(42,712) (names(i),i=1,no_para), ('fitness')
712    format(7x,120(a14,1x))
        do i=1,mempops
            write(42,713) iarray(i), (var(ii,iarray(i)),ii=1,no_para),
&                (fitness(ii,iarray(i)),ii=1,ngoals)
713    format(1x,i3,1x,120(e14.8,1x))
        end do
        write(42,*)' '
    end if

    ! First Generation now complete

    do i=1,mempops
        totalfit(i)=0.0d0
        do j=1, ngoals
            totalfit(i)=totalfit(i)+fitness(j,i)
        end do
        totalfit(i)=totalfit(i)
    end do

    Bestfitness=1.0d6
    do i=1,mempops
        if(totalfit(i) .lt. Bestfitness) then
            Bestfitness=totalfit(i)
            ielite=i
        end if
    end do

    write(41,*)'Steady State GA'
    write(41,*)1,Bestfitness

    ! Main Generation Loop
    do l=2,maxgen

        !Get Parent1
        randum=ran1(idum)
        parent1=randum*mempops
        if(parent1.lt.1.d0)parent1=mempops
        iparent1=int(parent1)
        randum=ran1(idum)
        iparent2=randum*mempops
        if(parent2.lt.1.d0)parent2=mempops
        iparent2=int(parent2)
        !Perform Tournament Selection to get best parent1
        if(totalfit(iparent1).lt.totalfit(iparent2)) then
            iBestParent1=iparent1
        else

```

```

        iBestParent1=iparent2
    end if

    !Get Parent2
    randum=ran1(idum)
    parent1=randum*mempops
    if(parent1.lt.1.d0)parent1=mempops
    iparent1=int(parent1)
    randum=ran1(idum)
    iparent2=randum*mempops
    if(parent2.lt.1.d0)parent2=mempops
    iparent2=int(parent2)
    !Perfom Tournament Selection to get best parent2
    if(totalfit(iparent1).lt.totalfit(iparent2)) then
        iBestParent2=iparent1
    else
        iBestParent2=iparent2
    end if

c    Check to see which type of crossover to use

    !Perform Uniform Crossover using BestParent 1 & 2
    if(iuniform.eq.-1)then
        do j=1,no_para
            rnd=ran1(idum)
            if(rnd.ge.0.5d0)child(j)=var(j,iBestParent1)
            if(rnd.lt.0.5d0)child(j)=var(j,iBestParent2)
        end do
    end if

    !Perform Single Point Crossover using BestParent 1 & 2
    if(isnglpoint.eq.-1)then

        rnd=ran1(idum)
        ipoint=rnd*no_para

        do j=1,ipoint
            child(j)=var(j,iBestParent1)
        end do

        if(ipoint.eq.no_para)goto 876

        do j=ipoint+1,no_para
            child(j)=var(j,iBestParent2)
        end do

876        continue

    end if

    !Perform Blend-Crossover Using BestParent 1 & 2
    if(iblend.eq.-1)then
        do j=1,no_para
            child(j)=abs(var(j,iBestParent1)-var(j,iBestParent2))
            &
            *ran1(idum)+dmin1(var(j,iBestParent1),var(j,iBestParent2))

```

```

        end do
    end if

    if(iuniform.eq.0 .and. iblend.eq.0 .and. isnglpoint.eq.0) then
        write(*,*)'*****NO CROSSOVER SELECTED*****'
        write(*,*)'*****SET UNIFORM OR BLEND TO TRUE IN GANNL.DAT'
        STOP
    end if
c  end of crossover

!Perform Mutation on Children
do j=1,no_para
    rnd=ran1(idum)
    if(rnd.lt.mue) then
        child(j)=sigma*(xmax(j)-xmin(j))*gasdev(idum)+
&         child(j)
    end if
end do

    do j=1,no_para
        if(child(j).gt.xmax(j)) child(j)=xmax(j)
        if(child(j).lt.xmin(j)) child(j)=xmin(j)
    end do

!Make New Population Replacing Worst Preformer with Child
!Replace Worst Preformer with Child
!Find worst Fitness for the Generation
Worstfitness=0.0d0
do i=1,mempops
    if(totalfit(i) .gt. worstfitness) then
        worstfitness=totalfit(i)
        iworst=i
    end if
end do

do j=1,no_para
    var(j,iworst)=child(j)
enddo

do jk=1,no_para
    ys(9,jk)=var(jk,iworst)
enddo

open(unit=47,file='SaveCurrentVariables.dat',status='unknown')
write(47,*)0
write(47,*)'Current: Gen#,Member#',1,iworst
do jk=1,no_para
    write(47,*)var(jk,iworst)
end do
close(47)

ys(4,3)=1
ys(4,2)=iworst

call setup

```

```

        do j=1,ngls
            fitness(j,iworst)=sngl(ys(7,j))
        enddo

if(ipopulation.eq.-1) then
    write(42,*)'Generation#',1
end if

        do j=1,mempops
            iarray(j)=j
        end do

if(ipopulation.eq.-1) then
    write(42,712) (names(i),i=1,no_para), ('fitness')
write(42,713) iarray(iworst), (var(ii,iarray(iworst)),ii=1,no_para),
&
    (fitness(ii,iarray(iworst)),ii=1,ngoals)

    write(42,*)' '
end if

        do i=1,mempops
            totalfit(i)=0.0d0
            do j=1,ngoals
                totalfit(i)=totalfit(i)+fitness(j,i)
            end do
            totalfit(i)=totalfit(i)
        end do

!Test to see if mutation amount needs to be adjusted

if(ivar_mutation.eq.-1)then
    k=k+1

        do i=1,mempops
            if(totalfit(i).lt. bestfitness)then
                ibetter=ibetter+1
                write(*,*)'ibetter',ibetter,totalfit(i),bestfitness
            end if
        end do

if(k.eq.iK_check)then
    better=float(ibetter)
    pop=float(k)

    amountbetter=better/pop

    if(amountbetter.gt.0.2d0) sigma=sigma*1.2d0
    if(amountbetter.lt.0.2d0) sigma=sigma*.8d0

    if(sigma.gt.0.8) sigma=0.8
        write(45,*)1,amountbetter,sigma
        k=0
        ibetter=0
    end if
end if
!end of variable mutation amount routine

```

```

!Find Best Fitness for the Generation
do i=1,mempops
  if(totalfit(i) .lt. bestfitness) then
    bestfitness=totalfit(i)
    ielite=i
  end if
end do

!Store Variables for best Population
  open(unit=46,file='Best_POP.dat',status='unknown')
  write(46,*)0
  write(46,*)'Best Member:
Gen#,Member#,Fit',1,ielite,bestfitness
  do i=1,no_para
    write(46,*)var(i,ielite)
  end do
  close(46)

  write(56,717)1,ielite,(var(ii,ielite),ii=1,no_para),
&          (fitness(ii,ielite),ii=1,ngoals)
717  format(1x,i7,1x,i3,1x,120(e14.8,1x))

  write(41,*)1,ielite,bestfitness

end do
close(56)
end

```

```

FUNCTION gasdev(idum)
INTEGER idum
REAL gasdev
CU  USES ran1
INTEGER iset
REAL fac,gset,rsq,v1,v2,ran1
SAVE iset,gset
DATA iset/0/
if (idum.lt.0) iset=0
if (iset.eq.0) then
1  v1=2.*ran1(idum)-1.
  v2=2.*ran1(idum)-1.
  rsq=v1**2+v2**2
  if(rsq.ge.1..or.rsq.eq.0.)goto 1
  fac=sqrt(-2.*log(rsq)/rsq)
  gset=v1*fac
  gasdev=v2*fac
  iset=1
else
  gasdev=gset
  iset=0
endif
return
END

```



```

FUNCTION ran1(idum)
INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
REAL ran1,AM,EPS,RNMX
PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
*NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER j,k,iv(NTAB),iy
SAVE iv,iy
DATA iv /NTAB*0/, iy /0/
if (idum.le.0.or.iy.eq.0) then
  idum=max(-idum,1)
  do 11 j=NTAB+8,1,-1
    k=idum/IQ
    idum=IA*(idum-k*IQ)-IR*k
    if (idum.lt.0) idum=idum+IM
    if (j.le.NTAB) iv(j)=idum
11  continue
  iy=iv(1)
endif
k=idum/IQ
idum=IA*(idum-k*IQ)-IR*k
if (idum.lt.0) idum=idum+IM
j=1+iy/NDIV
iy=iv(j)
iv(j)=idum
ran1=min(AM*iy,RNMX)
return
END

```

APPENDIX B: Single Stage Solid Real GA Input File

```

.true.                ;steady_state ONLY 1 GA TYPE TRUE
.false.              ;population check true writes popul
.false.              ;uniform x (50% parent1 and parent2
.true.               ;Blend x (blend of parents)
.false.              ;singlepointx
.false.              ;var_mutation true allows mutation
500                  ;kcheck # of gen before 1/5 rule ck
0.05                 ;xmutation_rate how much mutation
1.0                  ;xmutation_amount % of variables muta
1                    ;ngoals
29                   ;no_para
'rnos/rbod' 0.6 0.4 ;xmax-xmin
'lnos/dbod' 3.0 1.5 ;xmax-xmin
'kfuel__3' 9.0 1.0 ;xmax-xmin
'rpvar__4' 0.8 0.3 ;xmax-xmin
'rivar__5' 0.8 0.1 ;xmax-xmin
'nsp____6' 11.0 5.0 ;xmax-xmin
'fvar____7' 0.1 0.03 ;xmax-xmin
'eps____8' 0.95 0.6 ;xmax-xmin
'ptang__9' 5.0 1.0 ;xmax-xmin
'fn1____10' 0.99 0.66 ;xmax-xmin
'dth/Db_11' .30 .25 ;xmax-xmin
'Lb/Db_12' 15.0 10.0 ;xmax-xmin
'dbody__13' .64 .25 ;xmax-xmin
'b2w/DB_14' .05 .01 ;xmax-xmin
'crw/DB_15' .05 0.01 ;xmax-xmin
'trw____16' 0.99 0.90 ;xmax-xmin
'wleswe_17' 30.0 1.0 ;xmax-xmin
'xLEw__18' .25 0.20 ;xmax-xmin
'b2t/DB_19' 1.4 1.2 ;xmax-xmin
'crt/DB_20' 1.1 0.9 ;xmax-xmin
'trt____21' 0.99 0.50 ;xmax-xmin
'tleswp_22' 30.0 1.0 ;xmax-xmin
'xTEt__23' 1.00 0.95 ;xmax_xmin
'APdly__24' 4001.00 3999.00 ;xmax-xmin
'APtau__25' .8 .4 ;xmax-xmin
'APzeta_26' .9 0.4 ;xmax-xmin
'APwcr__27' 70.0 40.0 ;xmax-xmin
'pngain_28' 5.0 3.0 ;xmax-xmin
'thet0__29' 85.0 40.0 ;xmax-xmin
1
3
10000

```

APPENDIX C: Single Stage Solid Binary GA Input File

```

.false.                ;micro
.false.                ;pareto
.false.                ;steady_state
.false.                ;maximize
.true.                 ;elitist
.true.                 ;creep
.false.                ;uniform
.false.                ;restart
.true.                 ;remove_dup
.false.                ;niche
.false.                ;phenotype
0.04                   ;niche_diversity_percent_goal
67741                  ;iseed
0.9                   ;pcross
0.002                  ;pmutation
0.05                   ;pcreep
2                      ;ngoals
1.0,1.0               ;xgls(j+1),xgls(j+2) . . .
1.                    ;domst
2550                   ;convrng_chk
29                    ;no_para
'rnos/rbod' 0.6 0.4 0.1 .false. ;xmax-xmin-resolution-niche_par
'lnos/dbod' 3.0 1.5 0.1 .false. ;xmax_xmin_resolution_niche_par
'kfuel__3' 9.0 1.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'rpvar__4' 0.8 0.3 0.1 .false. ;xmax_xmin_resolution_niche_par
'rivar__5' 0.8 0.1 0.1 .false. ;xmax_xmin_resolution_niche_par
'nsp____6' 11.0 5.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'fvar____7' 0.1 0.03 0.01 .false. ;xmax_xmin_resolution_niche_par
'eps____8' 0.95 0.6 0.01 .false. ;xmax_xmin_resolution_niche_par
'ptang__9' 5.0 1.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'fn1____10' 0.99 0.66 0.01 .false. ;xmax_xmin_resolution_niche_par
'dth/Db_11' .30 .25 0.002 .false. ;xmax_xmin_resolution_niche_par
'Lb/Db_12' 15.0 10.0 .5 .false. ;xmax_xmin_resolution_niche_par
'dbody__13' .64 .25 .002 .false. ;xmax_xmin_resolution_niche_par
'b2w/DB_14' .05 .01 0.01 .false. ;xmax_xmin_resolution_niche_par
'crw/DB_15' .05 0.01 0.01 .false. ;xmax_xmin_resolution_niche_par
'trw____16' 0.99 0.90 0.01 .false. ;xmax_xmin_resolution_niche_par
'wleswe_17' 30.0 1.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'xLEw__18' .25 0.20 0.01 .false. ;xmax_xmin_resolution_niche_par
'b2t/DB_19' 1.4 1.2 0.1 .false. ;xmax_xmin_resolution_niche_par
'crt/DB_20' 1.1 0.9 0.1 .false. ;xmax_xmin_resolution_niche_par
'trt____21' 0.99 0.50 0.01 .false. ;xmax_xmin_resolution_niche_par
'tleswp_22' 30.0 1.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'xTEt__23' 1.00 0.95 0.01 .false. ;xmax_xmin_resolution_niche_par
'APdly_24' 4001.00 3999.00 1.00 .false. ;xmax-xmin-resolution-niche_par
'APtau_25' .8 .4 0.1 .false. ;xmax_xmin_resolution_niche_par
'APzeta_26' .9 0.4 0.01 .false. ;xmax_xmin_resolution_niche_par
'APwcr_27' 70.0 40.0 1.0 .false. ;xmax_xmin_resolution_niche_par
'pngain_28' 5.0 3.0 0.1 .false. ;xmax_xmin_resolution_niche_par
'thet0__29' 85.0 40.0 1.0 .false. ;xmax-xmin-resolution-niche_par

```

1
100
100

APPENDIX D: Two Stage Solid Real GA Input File

```

.false.                ;population check true writes popul
.false.                ;uniform x (50% parent1 and parent2
.true.                 ;Blend x (blend of parents)
.false.                ;singlepointx
.false.                ;var_mutation true allows mutation
500                    ;kcheck # of gen before 1/5 rule ck
0.05                   ;xmutation_rate how much mutation
1.0                    ;xmutation_amount % of variables muta
2                      ;ngoals
46                     ;no_para
'rnos/rbd 1' 0.6 0.4 ;xmax xmin
'lnos/dbd 2' 3.0 1.5 ;xmax xmin
'kfuel 3' 6.1 6.0 ;xmax xmin
'rpvar 4' 0.5 0.4 ;xmax xmin
'rivar 5' 0.6 0.5 ;xmax xmin
'nsp 6' 9.1 9.0 ;xmax xmin
'fvar 7' 0.1 0.07 ;xmax xmin
'eps 8' 0.9 0.8 ;xmax xmin
'ptang 9' 10.1 10.0 ;xmax xmin
'fnl 10' 0.99 0.88 ;xmax xmin
'dth/Db 11' .30 .28 ;xmax xmin
'Lb1/Db 12' 14.1 10.0 ;xmax xmin
'dbody1 13' .8 .4 ;xmax xmin
'b2t/DB 14' 1.4 1.2 ;xmax xmin
'crt/DB 15' 1.1 0.9 ;xmax xmin
'lrt 16' 0.99 0.96 ;xmax xmin
'tleswp 17' 40.0 2.0 ;xmax xmin
'xTEt/Lb 18' 1.00 0.98 ;xmax xmin
'kfuel 19' 6.1 6.0 ;xmax xmin
'rpvar 20' 0.6 0.4 ;xmax xmin
'rivar 21' 0.6 0.5 ;xmax xmin
'nsp 22' 9.1 9.0 ;xmax xmin
'fvar 23' 0.1 0.07 ;xmax xmin
'eps 24' 0.9 0.8 ;xmax xmin
'ptang 25' 10.1 10.0 ;xmax xmin
'fnl 26' 0.99 0.88 ;xmax xmin
'Dth/Db 27' .32 .31 ;xmax xmin
'Lb2/Db 28' 10.0 7.0 ;xmax xmin
'dbody2 29' 0.9 0.60 ;xmax xmin
'b2t/DB 30' 1.4 1.2 ;xmax xmin
'crt/DB 31' 1.1 0.9 ;xmax xmin
'lrt 32' 0.99 0.96 ;xmax xmin
'tLeswp 33' 40.0 2.0 ;xmax xmin
'xTEt/Lb 34' 1.00 0.98 ;xmax xmin
'tsep2 35' 2.6 1.6 ;xmax xmin
'APdly1 36' 4001.0 4000.0 ;xmax xmin
'APtc1 37' .7 .4 ;xmax xmin
'APdmp1 38' .9 0.6 ;xmax xmin
'cohz1 39' 60.0 50.0 ;xmax xmin

```

```
'pngain1 40' 6.0 3.0 ;xmax xmin  
'APdly1 41' 4001.0 4000.0 ;xmax xmin  
'APtc2 42' .7 .4 ;xmax xmin  
'APdmp2 43' .9 0.6 ;xmax xmin  
'cohz2 44' 60.0 50.0 ;xmax xmin  
'pngain2 45' 6.0 3.0 ;xmax xmin  
'thet0 46' 86.0 45.0 ;xmax xmin  
1 ;freq  
3 ;no members  
10000 ;no generations
```

APPENDIX E: Two Stage Solid Binary GA Input File

```

.false.                ; micro
.false.                ; pareto
.false.                ; steady_state
.false.                ; maximize
.true.                 ; elitist
.true.                 ; creep
.false.                ; uniform
.false.                ; restart
.true.                 ; remove_dup
.false.                ; niche
.false.                ; phenotype
0.04                   ; niche diversity percentile goal
67742                  ; iseed
0.9                    ; pcross
0.002                  ; pmutation
0.05                   ; pcreep
2                       ; ngoals
1.0,1.0                ; xgls(j)
1.                     ; domst
2550                   ; convrg_chk (end of group2)
46                     ; no_para
'rnos/rbd 1' 0.6 0.4 0.1 .false. ;xmax xmin resolution niche_par
'lnos/dbd 2' 3.0 1.5 0.1 .false. ;xmax xmin resolution niche_par
'kfuel 3' 6.1 6.0 0.1 .false. ;xmax xmin resolution niche_par
'rpvar 4' 0.5 0.4 0.1 .false. ;xmax xmin resolution niche_par
'rivar 5' 0.6 0.5 0.1 .false. ;xmax xmin resolution niche_par
'nsp 6' 9.1 9.0 0.1 .false. ;xmax xmin resolution niche_par
'fvar 7' 0.1 0.07 0.01 .false. ;xmax xmin resolution niche_par
'eps 8' 0.9 0.8 0.01 .false. ;xmax xmin resolution niche_par
'ptang 9' 14.1 10.0 0.1 .false. ;xmax xmin resolution niche_par
'fnl 10' 0.99 0.88 0.01 .false. ;xmax xmin resolution niche_par
'dth/Db 11' .30 .28 0.01 .false. ;xmax xmin resolution niche_par
'Lb1/Db 12' 10.1 10.0 .1 .false. ;xmax xmin resolution niche_par
'dbody1 13' .8 .4 .01 .false. ;xmax xmin resolution niche_par
'b2t/DB 14' 1.4 1.2 0.1 .false. ;xmax xmin resolution niche_par
'crt/DB 15' 1.1 0.9 0.1 .false. ;xmax xmin resolution niche_par
'rt 16' 0.99 0.96 0.01 .false. ;xmax xmin resolution niche_par
'tleswp 17' 40.0 2.0 2.0 .false. ;xmax xmin resolution niche_par
'xTEt/Lb 18' 1.00 0.98 0.01 .false. ;xmax xmin resolution niche_par
'kfuel 19' 6.1 6.0 0.1 .false. ;xmax xmin resolution niche_par
'rpvar 20' 0.6 0.4 0.1 .false. ;xmax xmin resolution niche_par
'rivar 21' 0.6 0.5 0.1 .false. ;xmax xmin resolution niche_par
'nsp 22' 9.1 9.0 0.1 .false. ;xmax xmin resolution niche_par
'fvar 23' 0.1 0.07 0.01 .false. ;xmax xmin resolution niche_par
'eps 24' 0.9 0.8 0.01 .false. ;xmax,xmin,resolution,niche_par
'ptang 25' 10.1 10.0 .1 .false. ;xmax xmin resolution niche_par
'fnl 26' 0.99 0.88 0.01 .false. ;xmax xmin resolution niche_par
'Dth/Db 27' .32 .31 0.01 .false. ;xmax xmin resolution niche_par
'Lb2/Db 28' 10.0 7.0 .5 .false. ;xmax xmin resolution niche_par

```

```

'dbody2 29' 0.9 0.6 .01 .false. ;xmax xmin resolution niche_par
'b2t/DB 30' 1.4 1.2 0.1 .false. ;xmax xmin resolution niche_par
'crt/DB 31' 1.1 0.9 0.1 .false. ;xmax xmin resolution niche_par
'trt 32' 0.99 0.96 0.01 .false. ;xmax xmin resolution niche_par
'tLeswp 33' 40.0 2.0 2.0 .false. ;xmax xmin resolution niche_par
'xTEt/Lb 34' 1.00 0.98 0.01 .false. ;xmax xmin resolution niche_par
'tsep2 35' 2.6 1.6 0.1 .false. ;xmax xmin resolution niche_par
'APdly1 36' 4001.0 4000.0 1.0 .false. ;xmax xmin-resolution-niche_par
'APtc1 37' .7 .4 .1 .false. ;xmax xmin resolution niche_par
'APdmp1 38' .9 0.6 0.01 .false. ;xmax xmin resolution niche_par
'cohz1 39' 60.0 50.0 1.0 .false. ;xmax xmin resolution niche_par
'pngain1 40' 6.0 3.0 0.5 .false. ;xmax xmin resolution niche_par
'APdly1 41' 4001.0 4000.0 1.0 .false. ;xmax xmin-resolution-niche_par
'APtc2 42' .7 .4 .1 .false. ;xmax xmin resolution niche_par
'APdmp2 43' .9 0.6 0.01 .false. ;xmax xmin resolution niche_par
'cohz2 44' 60.0 50.0 1.0 .false. ;xmax xmin resolution niche_par
'pngain2 45' 6.0 3.0 0.5 .false. ;xmax xmin resolution niche_par
'thet0 46' 86.0 45.0 1.0 .false. ;xmax xmin resolution niche_par
1 ;freq
100 ;no members
100 ;no generations

```


APPENDIX F: Single Stage Liquid Real GA Input File

```

.true.                ;steady_state ONLY 1 GA TYPE TRUE
.false.              ;population check true writes popul
.false.              ;uniform x (50% parent1 and parent2
.true.               ;Blend x (blend of parents)
.false.              ;singlepointx
.false.              ;var_mutation true allows mutation
500                  ;kcheck # of gen before 1/5 rule ck
0.05                 ;xmutation_rate how much mutation
1.0                  ;xmutation_amount % of variables muta
1                    ;ngoals
26                   ;no_para
'dbody 1' , 6.6 , 5.0 ;xmax,xmin
'kprop 2' , 4.1 , 4.0 ;xmax,xmin
'eqr 3' , .7 , .6 ;xmax,xmin
'po 4' , 2800. , 2000. ;xmax,xmin
'dnosDB 5' , 1.0 , .5 ;xmax,xmin
'blnosDB 6' , 2.0 , 1.00 ;xmax,xmin
'dstarDB 7' , .16 , .08 ;xmax,xmin
'eps 8' , 30. , 5. ;xmax,xmin
'fnl 9' , 0.9 , 0.6 ;xmax,xmin
'tb 10' , 150.0 , 70.00 ;xmax,xmin
'crwDB 11' , .04 , .02 ;xmax,xmin
'trw 12' , 0.92 , 0.89 ;xmax,xmin
'b2wDB 13' , 0.05 , 0.02 ;xmax,xmin
'angLE1 14' , 5.0 , 1.00 ;xmax,xmin
'xLEwTL 15' , 0.50 , 0.10 ;xmax,xmin
'crtDB 16' , 1.10 , 1.00 ;xmax,xmin
'trt 17' , 0.99 , 0.50 ;xmax,xmin
'b2ftDB 18' , 1.20 , 1.00 ;xmax,xmin
'angLE2 19' , 44.0 , 0.00 ;xmax,xmin
'xTEtTL 20' , 1.00 , 0.95 ;xmax,xmin
'Apdly 21' , 5000.0 , 4999.0 ;xmax,xmin
'tau 22' , 0.80 , 0.10 ;xmax,xmin
'zeta 23' , 0.99 , 0.50 ;xmax,xmin
'Cohz 24' , 60.00 , 40.0 ;xmax,xmin
'pronvg 25' , 7.0 , 3.0 ;xmax,xmin
'theta0 26' , 88.0 , 76.0 ;xmax,xmin
1                    ; ifreq
3                    ; mempops
10000                ; maxgen

```

APPENDIX G: Single Stage Liquid Binary GA Input File

```

.false.                ; micro
.false.                ; pareto
.false.                ; steady_state
.false.                ; maximize
.true.                 ; elitist
.true.                 ; creep
.false.                ; uniform
.false.                ; restart
.true.                 ; remove_dup
.true.                 ; niche
.true.                 ; phenotype
0.5                    ; niche diversity percentile goal
61732                  ; iseed
0.9                    ; pcross
0.002                  ; pmutation
0.05                   ; pcreep
1                      ; ngoals
1.0                    ; xgls(j)
2550                   ; convrg_chk (end of group2)
26                     ; no_para
'dbody 1' , 6.6 , 5.0 , 0.1 , .false. ;xmax,xmin,resolution,niche_par
'kprop 2' , 4.1 , 4.0 , 0.1 , .false. ;xmax,xmin,resolution,niche_par
'eqr 3' , .7 , .6 , 0.1 , .false. ;xmax,xmin,resolution,niche_par
'po 4' , 2800. , 2000. , 10.0 , .true. ;xmax,xmin,resolution,niche_par
'dnosDB 5' , 1.0 , .5 , .1 , .false. ;xmax,xmin,resolution,niche_par
'blnosDB 6' , 2.0 , 1.00 , .1 , .false. ;xmax,xmin,resolution,niche_par
'dstarDB 7' , .16 , .08 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'eps 8' , 30. , 5. , 0.2 , .false. ;xmax,xmin,resolution,niche_par
'fnl 9' , 0.9 , 0.6 , 0.05 , .false. ;xmax,xmin,resolution,niche_par
'tb 10' , 150.0 , 70.00 , 1.0 , .false. ;xmax,xmin,resolution,niche_par
'crwDB 11' , .04 , .02 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'trw 12' , 0.92 , 0.89 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'b2wDB 13' , 0.05 , 0.02 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'angLE1 14' , 5.0 , 1.00 , 1.00 , .false. ;xmax,xmin,resolution,niche_par
'xLEwTL 15' , 0.50 , 0.10 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'crtDB 16' , 1.10 , 1.00 , 0.10 , .false. ;xmax,xmin,resolution,niche_par
'trt 17' , 0.99 , 0.50 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'b2ftDB 18' , 1.20 , 1.00 , 0.10 , .false. ;xmax,xmin,resolution,niche_par
'angLE2 19' , 44.0 , 0.00 , 2.00 , .false. ;xmax,xmin,resolution,niche_par
'xTEtTL 20' , 1.00 , 0.95 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'Apdly 21' , 5000.0 , 4999.0 , 1.0 , .false. ;xmax,xmin,resolution,niche_par
'tau 22' , 0.80 , 0.10 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'zeta 23' , 0.99 , 0.50 , 0.01 , .false. ;xmax,xmin,resolution,niche_par
'Cohz 24' , 60.00 , 40.0 , 1.00 , .false. ;xmax,xmin-resolution,niche_par
'pronvg 25' , 7.0 , 3.0 , 0.20 , .false. ;xmax,xmin,resolution,niche_par
'theta0 26' , 88.0 , 76.0 , 1.00 , .false. ;xmax,xmin,resolution,niche_par
1                      ; ifreq
100                    ; mempops
100                    ; maxgen

```