Genetic Algorithm Optimization of Inlet Geometry for a Hypersonic Jet Engine with Mode Transition

Ashley E. Micks[1]
NASA Academy, NASA Glenn Research Center, Cleveland, OH, 44070
Massachusetts Institute of Technology, Cambridge, MA, 02139

Project Mentor:
Dr. Meng-Sing Liou[2]
NASA Glenn Research Center, Cleveland, OH 44070

[Abstract] A genetic algorithm (GA) program is developed to simulate the "evolution" of a "population" of hypersonic jet engine inlet geometries over many generations, with "natural selection" favoring better total pressure recovery. The algorithm mimics biological evolution to produce increasingly desirable designs with each generation. The result is an improved design from a small-scale run, plus the functional algorithm, complete with auxiliary scripts for use in future work such as time- and computation-intensive large-scale runs.

Contents

[1] NASA Academy Research Associate, Research and Technology Directorate, M.S. 5-11.
[2] Aerospace Engineer, Research and Technology Directorate, M.S. 5-11.

Glossary

allele = in a GA, the value of a given design variable for a potential design
blended crossover = in a GA, a process that produces children whose alleles are weighted averages of the parents' alleles
CFD = computational fluid dynamics
child = in a GA, a new individual placed into the next generation
cowl = a flap in a hypersonic inlet that closes the turbojet flow path at high speeds
chromosome = in a GA, a string of information defining an individual's design
crossover = in a GA, the process of swapping or blending alleles between parents to form children
entrance plane = a plane just before the beginning of the inlet walls, where freestream conditions are set
fitness = in a GA, a measure of an individual's performance, usually from the objective function
GA = genetic algorithm
generation = in a GA, the set of individuals generated by one round of reproduction, analogous to a generation in a species of organisms
hypersonic = speeds significantly above the speed of sound, usually Mach 5 and higher
individual = in a GA, a potential design considered by the algorithm
IPG = initial population generator
key point = a point along the inlet wall whose coordinates are used as design variables
L/IMX = Large-Scale Inlet Mode Transition, an inlet design project at NASA Glenn
objective function = a measure of a design's performance as a function of its design parameters
Overflow = the CFD program used for this project
parent = in a GA, an individual chosen for use in generating children
population = in a GA, the set of individuals in the current generation
$p_{t,enter}$ = stagnation pressure at an inlet's entrance plane
$p_{t,exit}$ = stagnation pressure at an inlet's exit plane
ramp = the wall opposite the cowl in a hypersonic inlet
total pressure recovery = $p_{t,exit}/p_{t,enter}$

I. Introduction

IN the aircraft design process, computational fluid dynamics (CFD) is a rapidly growing field that allows engineers to test new designs quickly and easily from their desktop computers rather than having to spend as much time and money on wind tunnel tests. While CFD is no substitute for experimental data, it can help engineers decide how they want to budget their wind tunnel time, calculating optimized designs that the engineers would then want to test and validate.

To fulfill this function, a wealth of optimization tools has been built around and with CDF software. However, for design problems with complicated, nonlinear objective functions and many design variables, the most common, gradient-based optimization techniques are very limited. An objective function provides a measure of how well a given design performs, as a function of its values for the design variables. Gradient-based methods will find the local maximum in the objective function closest to the baseline design, but will not be able to go farther, since they only move "uphill."

The emerging field of genetic algorithms (GAs) offers an alternative. Based on biological evolution, a GA's search is global, exploring the entire design space instead of simply climbing hills. First, a GA will generate an initial "population" of potential designs, randomly scattered across the design space. It assigns each design—each individual in the population—a "fitness" value by evaluating the individual using the objective function. Individuals with higher fitness are more likely to "reproduce," so that their traits continue to the next generation. As in biological reproduction, crossover and mutation of "chromosomes"—which in this case are lists of design variable values or "alleles"—bring diversity to the population so that the algorithm can consider more possibilities. Since a GA searches from multiple points in the design space—multiple individuals in a population—parallel processing fits in naturally with one individual per processor per generation, dramatically decreasing computing time.

The baseline design chosen for this optimization is the Large-scale Inlet Mode Transition (L/IMX) inlet currently undergoing tests with the Inlet and Nozzle Branch at NASA Glenn Research Center in the 1'x1' wind tunnel. This is an inlet designed for a hypersonic aircraft that will travel up to Mach 7 from Mach 0. Able to fly at such a range of speeds, the L/IMX inlet is a stepping stone on the way to a single-stage to orbit vehicle.

To make flight across this range of speeds possible for an air-breathing engine, mode transition is necessary. At high-Mach speeds, the stagnation temperature of the air is too hot for a turbojet engine to handle, so a ramjet—or for even higher speeds, a scramjet—must be used instead. Since ramjets and scramjets don't have meltable turbine or compressor machinery, they can stand the high speeds, but they're also dependent on the high speeds to compress the air without the help of compressor blades. That means the vehicle must have both types of engines (or rocket propellant) to reach hypersonic speeds: a turbojet for subsonic and slightly supersonic speeds and a ramjet or scramjet for higher speeds—mode transition. The L/IMX inlet has a turbojet and a scramjet path, with cowl flaps to open and close the different flow paths as they are needed.

II. Problem Definition

The problem of inlet geometry optimization looks at the interactions between the shape of the inlet walls, the train of shocks down the inlet, the boundary layers, the total pressure, and ultimately how changes in inlet shape can improve the inlet's contribution to the engine's efficiency as a whole. Total pressure, or stagnation pressure, reflects how much work can be extracted from a fluid, so a loss in total pressure indicates an increase in entropy and a loss of ability to do work. Total pressure recovery is a measure of the change in total pressure from one end of the inlet to the other, so it represents the efficiency of the inlet.

With the goal to maximize total pressure recovery, obstacles include shock losses as well as flow separation, since boundary layers can increase entropy and shock strength, therefore increasing shock losses in total pressure. The inlet geometry must then be designed to keep the shocks as weak as possible and the flow separation minimal.

III. Objective

The goal of this project is to implement a GA to design an optimum geometry for the turbojet flow path of a hypersonic jet engine inlet at Mach 4, using the L/IMX inlet as a baseline. "Optimum" is defined here as having the maximum possible total pressure recovery, averaged across the inlet cross-section to account for non-uniform flow at the exit plane. Represented as a function of parameters determining inlet geometry, then, this cross-sectional average total pressure recovery is the GA's objective function.

IV. Approach

In order to implement a GA for the purpose outlined above, key components and subroutines of the GA (alleles, chromosomes, fitness, parent selection, crossover, mutation, and design constraints) need to be defined. Next, a plan needs to be developed for how the different programs involved in the GA (an initial population generator, a grid generator, a CFD program called Overflow, the GA itself) interact.

A. Alleles

Alleles are design parameter values that define an individual, a design considered by the GA. In this project, there are three types of alleles contained in each chromosome.

1) *Coordinates of key points along the inlet walls.* Both the x and y coordinates of the cowl lip were used as design variables, but only the y coordinates of the other key points were varied. Those points' x coordinates remained constant to simplify the constraints necessary to keep the inlet walls of newly generated individuals from intersecting themselves or stretching to infinity. X is positive in the ramp-to-

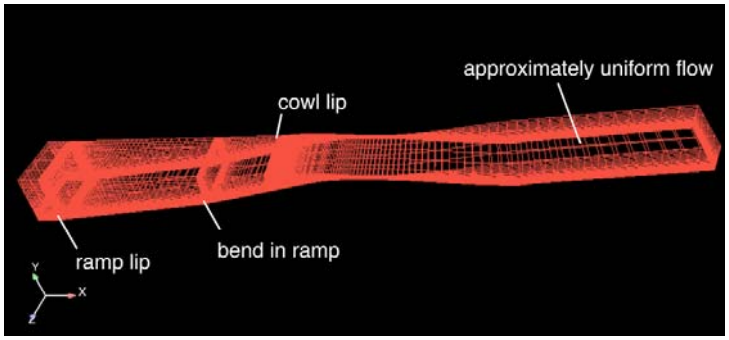


Figure 2. Geometry of the Baseline Inlet. *This grid represents the turbojet flow path geometry in the L/IMX inlet, using the xyz coordinate system employed in the GA.*

This hypersonic inlet has two flow paths, one through a turbojet engine and one through a scramjet engine. The cowls open and close according to Mach number so that the air follows the path appropriate for the current speed.

cowl direction, parallel to the inlet's entrance and exit planes, and y is defined positive in the direction of oncoming flow perpendicular to the entrance plane. The origin is at the ramp lip (see Figure 2).

2) *Bias values.* This is one element of Hermite interpolation, which is used to determine the shape of the nonlinear inlet wall sections. There is a bias value for each wall section where Hermite interpolation is used, and it affects which direction the wall bends as it goes through the key points at either end of the wall section.
3) *Tightness values.* This is another element of Hermite interpolation. Like bias values, there is one for each Hermite-interpolated wall section. These numbers determine how sharply the wall bends.

B. Chromosomes

Each individual's chromosome is simply a list of its alleles. The x and y coordinates of the cowl lip are listed first, followed by the y coordinates of the other key points, the bias values, and finally the tightness values.

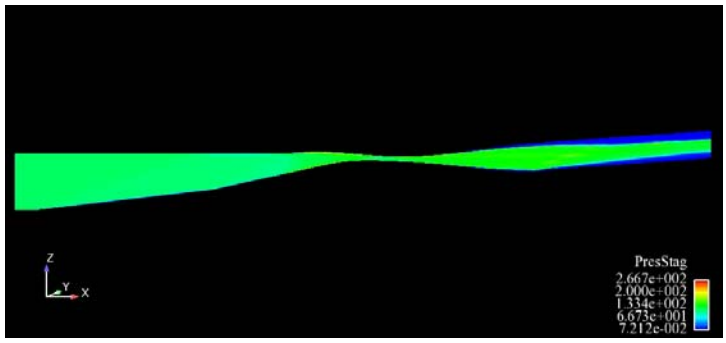


Figure 3. Total Pressure along the Baseline Inlet. *This is a side view of the L/IMX turbojet flow path, colored according to the nondimensionalized total pressure (or stagnation pressure) of the flow. The coordinate axes are different from in Figure 2 because Overflow, which calculated this flow pattern, requires the use of a different axis system than the one used by the GA.*

C. Fitness

As stated above, an individual's fitness equals its total pressure recovery, but that value must also accommodate the non-uniform flow at the exit plane due to boundary layer effects. Total pressure is defined as uniform everywhere on the entrance plane, but it has different values at different y-positions on the exit plane. Since Overflow only calculates values for discrete points—the intersections of gridlines on the grid representing the inlet's geometry—the Riemann sums of total pressure at the entrance and exit planes are each divided by the area of the respective plane to find average total pressures at the entrance and exit. Total pressure recovery is then the average at the exit divided by the average at the entrance. Also, for each gridline intersection, Overflow only outputs the nondimensionalized density, specific heat ratio, and velocity vector components, so the GA calculates nondimensionalized total pressure from that information.

D. Initialization

The initial population is randomly distributed across the design space, with all possible values of a given allele having equal likelihood. The L/IMX inlet is always included as one individual in this first generation, so that the GA can take advantage of the optimization that has already gone into this baseline design.

E. Parent Selection

"Natural selection" for better fitness occurs when the individuals with higher fitness values have a better chance of passing on their traits to the next generation. Here, Roulette Wheel selection is used, so the probability of choosing a particular individual to become a parent is directly proportional to that individual's fitness, and all individuals have some chance of being selected. An individual can be selected as a parent more than once, since its probability of selection does not change as more pairs of parents are selected.

F. Crossover

Once two parents are chosen, there is a 0.7 chance of crossover. If crossover does not occur, the parents are simply copied into the next generation. When crossover occurs, however, some alleles will be swapped or blended between the two parent chromosomes to form two children. Because the chromosomes are arranged by type of allele (coordinates, bias, tightness), alleles are chosen at random to be crossed over, as opposed to the traditional method of crossing over all alleles beyond a randomly chosen point in both parent chromosomes. This alternative creates more diversity among children, allowing the GA to explore more of the design space. When the parent chromosomes undergo crossover, the alleles at a given chromosome location have a 1/3 chance of being swapped between parents, a 1/3 chance of blended crossover, and a 1/3 chance of remaining unchanged. Swapped alleles

have the same values that were present in the parent chromosomes, except that they are now on opposite chromosomes from where they were before. Blended alleles, however, are weighted averages of the corresponding alleles on the parent chromosomes. A random weighting factor, γ, is chosen between 0 and 1 so that

$$a_{child1} = \gamma a_{parent1} + (1-\gamma)a_{parent2} \quad (1)$$
$$a_{child2} = \gamma a_{parent2} + (1-\gamma)a_{parent1} \quad (2)$$

for blended crossover at chromosome location a. The process of parent selection and crossover is repeated until the population of the next generation has been filled.

G. Mutation

Mutation happens to the children after they are produced by either crossover or by copying the parents into the next generation. It occurs at a random allele on every child, causing that allele to change its value to any value in the design space with equal probability. This is another technique designed to introduce variety into the population so that it does not converge to one small area in the design space without exploring the alternatives.

H. Elitism

So that the best fitness value in the population never decreases from one generation to the next, the best individual from the current generation is always copied directly into the next generation.

I. Design Constraints

There are two constraints imposed on inlet designs generated by the GA:

1.) The inlet walls cannot intersect.
2.) The cowl lip must be upstream of the shock wave coming off the ramp lip.

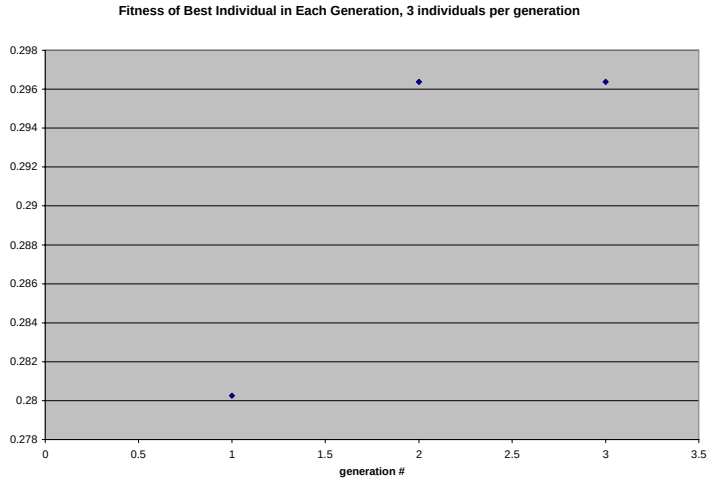
If a new individual violates a design constraint, the process that produced the individual (crossover, mutation, or random generation in the case of the first generation) is redone until a viable individual is produced.

J. Validation of the GA

Several test cases have validated that the GA does find better designs that approach the optimum. Initially, so that time-consuming Overflow runs would not be necessary, a dummy fitness function simply summed the allele values for an individual and used the sum as the x value in

$$f(x) = | \cos((x-400)/25) * e^{-|(x-400)/400|} | \quad (3)$$

which became the fitness value. This function is always positive, but has multiple local maxima to test whether the GA's search is global, as it should be. The optimum is at 400, which is relatively central in the design space, so that the initial population will fall on both sides of the optimum along the x-axis.



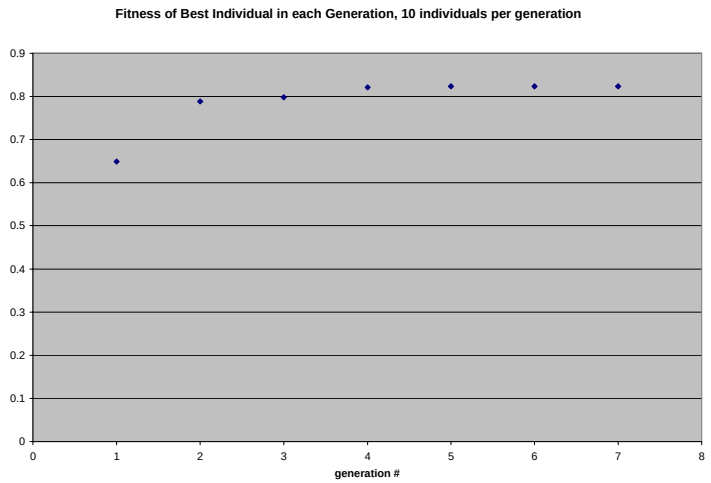


Figure 4. Progress of GA Optimization in Initial Tests. *These two tests used a dummy fitness function to evaluate the designs produced by the GA without requiring CFD calculations. The performance with small population of three individuals is very limited, but a test with a population of ten had better results, demonstrating the validity of the GA.*

The first test used three individuals per generation. With such a small population, a high-scoring individual quickly took over the "gene pool," making further improvement near impossible after three generations, since the whole population had the same chromosome. The best fitness achieved this way was also not very high, 0.296 out of an optimum 1, since the fitness function had its optimum at x=0 and all the initial designs were on one side of the optimum on the x-axis, making it hard to converge to the absolute max. A second test used ten individuals per generation and the fitness function in equation (3) to achieve better optimization. The best fitness value after seven generations was 0.823 out of 1, improved from the initial population's best score of 0.649. Slight additional improvements in performance came from spreading the random distribution of the initial population and mutated allele values evenly across the design space and increasing the rate of mutation to keep the population from getting stuck at a suboptimal point in the design space for lack of diverse alleles. Additional improvements in performance can be achieved through future work on this project, experimenting with alternative arrangements for crossover, mutation, parent selection, and elitism.

The calculation of fitness from CFD result files was validated using results for the baseline design. The output was a cross-sectional average total pressure recovery of 0.4377 for the L/IMX inlet—a reasonable value.

K. Analysis Tools

For successful implementation of the GA, tools other than the GA and IPG programs developed for this project are necessary.

1.) *Overflow.* This is the CFD software that calculates flow behavior for each potential design so that a fitness value for that design can be calculated. For the purposes of this project, Overflow's input is a Plot3D-format 3D grid representing an inlet's geometry and an input script specifying the boundary conditions, the number of iterations Overflow will use to converge, and whether or not convergence will begin from assumed uniform flow or from a "restart" file, i.e. from where a previous run of Overflow left off. The boundary conditions used here are uniform flow perpendicular to the entrance plane at Mach 4, Reynolds number 2.276 million (derived from L/IMX wind tunnel tests), and specific heat ratio γ=1.4, inviscid flow along boundaries parallel to the flow leading up to the ramp and cowl lips, and viscid flow along the inlet walls. There are no constraints on the conditions at the exit plane, which are a function of inlet geometry. The results of 1000 iterations on the L/IMX inlet is used as an initial restart file, and then the results file from Overflow is copied into a replacement restart file every time Overflow is run.

2.) *Grid generation.* A grid generation subroutine included in the GA outputs Plot3D-format 2D grid files for the inlet designs, and then a script called "makegrid" inputs this 2D grid file and outputs a corresponding 3D grid file that matches Overflow's input requirements. All inlet designs are approximated as 2D to decrease the runtime of Overflow to an amount appropriate for the time allotted to this project, ten weeks. The grid geometries input to Overflow are nominally 3D as required by the program, but they are uniform in the third dimension. There are 300 vertical gridlines, which are parallel to the entrance and exit planes and are concentrated near the areas where the flow is most complicated: bends in the ramp, the ramp lip, and the cowl lip—areas where shocks and boundary layers originate. There are 100 horizontal gridlines distributed between the top and bottom walls according to a Gaussian function, with more lines near the walls so that calculations regarding the viscid flow near the walls are more accurate.

3.) *EnSight.* This is a visualization tool that reads Plot3D grid files and Overflow results files, producing images of inlet geometries that it can color according to the flow properties, such as temperature, density, and total pressure.

L. Interactions Between Programs

There are a few different programs and a multitude of data files and directories necessary to run the GA and record its results. These components are spread across two machines: NASA's mass storage machine, Lou, and the Columbia supercomputer at NASA Ames Research Center. Columbia houses all the executable files, including Overflow, the GA itself, and an initial population generator (IPG). Lou contains the archives: the chromosomes, geometry grid files, Overflow flow behavior result files, and total pressure recovery values for every individual in every generation, as well as summary files stating the best individual in each generation and generational data files that each contain all the chromosomes from a generation so that the GA could be restarted from that generation if necessary. Calling the GA once only steps the process forward one generation, so a batch job script manages the whole, multigenerational process by calling the IPG at the beginning, calling the GA every generation, sending each individual to a different processor for the CFD calculations, and moving data files around as necessary.

V. Results

A final, small-scale run of the algorithm on the supercomputer yielded a design improved from the baseline. After two generations with a population size of four, the best individual had a total pressure recovery of 0.4450, compared to the baseline design's total pressure recovery of 0.4377. This new design is undoubtedly suboptimal because the population used to find it was so small and the number of generations so few, but it was necessary to keep the scale of the run small due to time constraints.

The other products of this project are a functional GA, IPG, fitness calculation, and random number generator. The GA and fitness calculation were validated as described previously in the section labeled, "Validation of the GA." These tests also validated the IPG, which needs to be called for every trial run of the algorithm, and requires no changes between runs using the dummy fitness function and runs using the actual fitness function. The grid generator within the GA has been validated by viewing the grids with EnSight. Images from EnSight of the designs also verified that the IPG and GA successfully checked themselves against the design constraints, so none of the designs from the final programs showed inlet walls intersecting in the visualization. Finally, throughout these test runs of the algorithm as well as separate tests of the random number generator only, the random numbers have been shown to be neither identical between runs of the program nor identical between random numbers generated in one run of the program. Unfortunately, there was not time in the ten weeks allotted to this project to run the large-scale optimizations for which the algorithm is intended. Such optimizations would generate upwards of a hundred individuals per generation and require multiple days to go through the Columbia supercomputer's job queue.

VI. Conclusion

The GA developed here does live up to the expectation for GAs to explore designs far from the baseline design, to make use of parallel computing, and to be adaptable to adjustments in the design problem, but it required considerable time and computing power to run. Parallel computing speeds up computations by a factor of the number of individuals in a generation, with one processor per individual, but the shear number of designs considered and the calculations required for each put a full-scale run of the algorithm out of scope for this ten-week project. The largest time requirements are for the CFD calculations and the generation of a new grid for every individual, including the hundreds of constraint-violating designs that the algorithm tries before finding working designs to insert into the population. These time and resource limitations factor into why GAs are not more widely used. Nonetheless, given how quickly computing power increases, the thoroughness of a genetic algorithm's search through the design space and the robustness of its natural-selection-inspired approach will soon be within reach for a much wider variety of projects.

Appendix

A. Algorithm Sequence

The following is a sequential breakdown of what occurs during a run of the algorithm.

Key:
Scripts and pieces of code
Folders, directories
Files

$num_ind means "insert the number of the current individual here"
$num_gen means "insert the number of the current generation here"
$num_minus_1 is one less than $num_gen

1. **The batch script** calls an **initial population generator called IPG**. The **IPG** is separate from the **GA** so that a random population isn't generated every time the **GA** is called, and the rest of the **GA** doesn't run before the **batch script** gets a chance to do anything else.
2. **IPG** generates the initial population, outputting each new individual's chromosome as a new **fort.i** file for the i^{th} individual. After each individual, it uses **Gengrid** to check that the design constraints are obeyed.
3. **Gengrid** (a subroutine in the **IPG** that takes integer arguments m and n, which set it to input **fort.m** and output **fort.n**) reads **fort.i** and outputs a **fort.1** grid file, replacing the **fort.1** for the previous individual each time.

4. **IPG** reads **fort.1** each time and sees if the walls intersect or if the shock comes before the cowl lip. If so, it tries again with a different individual.
5. Once the initial population is generated, **IPG** outputs the entire generation's chromosomes as individual **fort.1, fort.2, fort.3 . . .** files and as one file **gen.dat** so the generation can be recovered by the **GA**.
6. **Gengrid** reads each **fort.1, fort.2, fort.3 . . .** file and outputs grid files **fort.1, fort.2, fort.3 . . .** in their places.
7. **The batch script** copies **gen.dat** to **gen$num_gen.dat** in the **summary** folder on **Lou.**
8. **The batch script** copies each **fort.$num_ind** to **fort.$num_gen** in the respective **inlet$num_ind** folder on **Lou** and to **inlet.fmt** in the respective **inlet$num_ind** folder on **Columbia.**
9. **The batch script** calls **makegrid** in each individual's **inlet$num_ind** folder, using the 2D **inlet.fmt** to create a new 3D **grid.in** input file for **Overflow**.
10. **The batch script** copies **inlet.inp,** an input file, to **over.namelist**, **Overflow**'s other input file, in each individual's **inlet$num_ind** folder.
11. **The batch script** calls **overflow < over.namelist** in each individual's **inlet$num_ind** folder, running **Overflow** with **over.namelist** as input.
12. **Overflow** runs each individual on a different processor. A new **q.save** file appears in each **inlet$num_ind** folder. The **inlet.inp** file in each run folder has a .T. next to the "run from **q.restart**" statement, so Overflow's calculations converge starting from a previous inlet's calculations instead of from uniform flow, the default. **q.restart** was originally placed there ahead of time from the results for the baseline design.
13. **The batch script** copies each **q.save** to **q.restart** in each **inlet$num_ind** folder.
14. **The batch script** copies each **q.save, resid.out** and **grid.in** to **q.$num_gen, resid.$num_gen** and **grid.$num_gen** in the respective **inlet$num_ind** folder on **Lou**.
15. **The batch script** copies each **q.save** file to **fort.$num_ind** in the main directory where the **GA** is.
16. num_gen = num_gen + 1
17. **The batch script** calls the **GA.**
18. **The GA** reads the chromosomes from **gen.dat** into an array called oldpop
19. **The GA** finds a fitness for each individual using the result files now at **fort.1, fort.2, fort.3. . .** and **Gengrid**'s gridline spacing functions to integrate total pressure without needing the grid files.
20. **The GA** outputs the chromosome and fitness for best individual in that generation to **best.dat**, which contains a statement at the beginning of the file saying what the number of that best individual is, so that the corresponding q and grid files can be found.
21. **The GA** outputs **scores.dat**, a list of fitnesses next to the numbers of the individuals they correspond to.
22. **The GA** generates a new population, again checking each new individual by replacing **fort.i** with the chromosome, **Gengrid** generating a grid at **fort.1** based on that, and then the **GA** checking for wall intersections and cowl position.
23. **The GA** outputs the new population's chromosomes into replacement **fort.1, fort.2, fort.3. . .** and **gen.dat** files.
24. **Gengrid** (a subroutine in the **GA** identical to the one in the initial population generator) replaces **fort.1, fort.2, fort.3. . .** with the corresponding 2D grid files.
25. **The batch script** copies **gen.dat** to **gen$num_gen.dat** in the **summary** folder on **Lou.**
26. **The batch script** copies **best.dat** and **scores.dat** to **best$num_minus_1.dat** and **scores$num_minus_1.dat** in the **summary** folder on **Lou.**
27. **The batch script** copies each **fort.$num_ind** in the main **Columbia** directory to **fort.$num_gen** in the respective **inlet$num_ind** folder on **Lou** and to **inlet.fmt** in the respective **inlet$num_ind** folder on **Columbia.**
28. Go back up to step 9, unless the last generation you wanted to complete just had its **best.dat** and **scores.dat** moved to **Lou.**
29. Now, each inlet folder on **Lou** should contain:
    - A 3D binary Plot3D grid file for each individual **(grid.1, grid.2, grid.3 . . . one for the individual of the same number as the folder from each generation)**
    - A 3D binary Plot3D result file for each individual **(q.1, q.2, q.3 . . .)**
    - A 2D ASCII Plot3D grid file for each individual **(fort.1, fort.2, fort.3 . . .)**

And the summary folder on **Lou** should contain:

- A list of all the individuals' chromosomes, minus their fitness values, for each generation so that the algorithm can be restarted from that generation **(gen1.dat, gen2.dat, gen3.dat . . .)**

- The number (index) and chromosome of the best individual in each generation, including the fitness value **(best1.dat, best2.dat . . .)**
- A list of fitnesses for all the individuals in each generation, each fitness labeled with the individual's number **(scores1.dat, scores2.dat . . . one file per generation)**
- **resid.1, resid.2 . . .** files showing how much each individual converged

B. Directory Layout

The following is a schematic of how the directories were arranged for the algorithm across two NASA Advanced Supercomputing (NAS) machines: Columbia and Lou. * denotes an executable file.

Home directory on Columbia

GA*
IPG*
GA.f
IPG.f
gen.dat
best.dat
scores.dat
run_batch*
overflow
    over2.0aa
        bin
            overflow*
            makegrid*
Fort.1
        Fort.2
        .
        .
        .
Inlet1
    q.save
    grid.in
    q.restart
    inlet.fmt
    inlet.inp
    over.namelist
    resid.out

Inlet2
    (same as Inlet1)
Inlet3
    (same as Inlet1)
.
.
.

Home directory on Lou

Inlet1
    q.1
    q.2
    .
    .
    grid.1
    grid.2
    .
    .

fort.1
fort.2
.
.
resid.1
resid.2
.
.

Inlet2
(same as Inlet1)
Inlet3
(same as Inlet1)
.
.
.
Summary
gen1.dat
gen2.dat
.
.
best1.dat
best2.dat
.
.
scores1.dat
scores2.dat
.
.

Acknowledgments

Special thanks to Meng-Sing Liou, May-Fun Liou, Dave Saunders, MaryJo Long-Davis, Jay Horowitz, and Mary Vickerman for their mentorship and support. Thanks also to the Massachusetts Space Grant Consortium for making the author's work with NASA possible, to Dr. David Kankam for directing the NASA Glenn Academy, and to Kyle Gaiser for his friendship and collaboration.

References

[1]Bourke, Paul, "Interpolation Methods," URL: http://local.wasp.uwa.edu.au/~pbourke/other/interpolation/ [cited 16 July 2007].

[2]"CFD & Hypersonic Mode Transition." A PowerPoint presentation, NASA Glenn Research Center, Research and Technology, Inlet and Nozzle Branch, November 15 2006.

[3]Davis, Lawrence, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[4]Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[5]Oyama, Akira, and Liou, Meng-Sing, "Multiobjective Optimization of Rocket Engine Pumps Using Evolutionary Algorithm," AIAA 2001-2581, 2001.

[6]Walatka, Pamela P., Buning, Pieter G., Pierce, Larry, and Elson, Patricia A., *PLOT3D User's Manual*, NASA Technical Memorandum 101067, 1990.

[7]Whitlock, Sarah T., Boman, Erik, and Terry, Steven H., "Fortran Tutorial," URL: http://gershwin.ens.fr/vdaniel/Doc-Locale/Langages-Program-Scientific/Fortran/Tutorial/index.htm [cited 14 June 2007].