N94- 25666

SYSTEMATIC PROPULSION OPTIMIZATION TOOLS (SPOT)

University of Alabama in Huntsville
Department of Mechanical Engineering
Huntsville, Alabama

Dr. Mark Bower
John Celestian, Teaching Assistant

Abstract

This paper describes a computer program written by senior-level Mechanical Engineering students at the University of Alabama in Huntsville which is capable of optimizing user-defined delivery systems for carrying payloads into orbit. The custom propulsion system is designed by the user through the input of configuration, payload, and orbital parameters. The primary advantages of the software, called Systematic Propulsion Optimization Tools (SPOT), are a user-friendly interface and a modular Fortran 77 code designed for ease of modification.

The optimization of variables in an orbital delivery system is of critical concern in the propulsion environment. The mass of the overall system must be minimized within the maximum stress, force, and pressure constraints. SPOT utilizes the Design Optimization Tools (DOT) program for the optimization techniques.

The SPOT program is divided into a main program and five modules: aerodynamic losses, orbital parameters, liquid engines, solid engines, and nozzles. The program is designed to be upgraded easily and expanded to meet specific user needs. A user's manual and a programmer's manual are currently being developed to facilitate implementation and modification.

Introduction

Improved propulsion system designs and problems and problem solutions are vital to the future of the aerospace industry. To this end, propulsion optimization software can be a valuable tool to the design engineer. However, the software packages currently available are poorly organized and difficult to modify, thus placing the user in an unenviable position. Clearly the problem at hand is to develop a software package which delivers accurate results without sacrificing the user-friendly work environment.

This paper describes the Systematic Propulsion Optimization Tools (SPOT) program written by senior level engineering students at the University of Alabama in Huntsville. Its purpose is to develop the optimum system for delivering a payload into orbit. This is accomplished by optimizing the launch vehicle configuration based on the payload weight, engine types, and orbital parameters entered by the user. The code is written in FORTRAN with a QuickBasic user interface and employs a commercially available optimization routine, Design Optimization Tools (DOT), for all optimization calculations. This allows SPOT to be accurate, user-friendly, and easy to modify.

Scope

SPOT is designed to allow for quick analysis of either the effectiveness of a desired launch hardware combination, or to provide an optimized system to achieve a given orbit. The code allows the user to input any combination of elements within its parameters and to quickly obtain a "useful" answer.

In terms of hardware, SPOT can handle relatively large hardware combinations. The input vehicle can have anywhere from one to four stages. Each stage can be either liquid or solid-fueled and can have up to five engines. Between two and eight solid fuel strap-on engines can be used on the first stage to provide additional thrust. A propulsion system can be designed for payload weights of up to 100,000 pounds.

Some orbital mechanics constraints were also placed on the program in order to help make coding easier. Orbits are circular rather than elliptical. Any desired orbital radius within the feasible limits of the launch vehicle can be used. If the launch vehicle cannot reach the orbit input by the user, the program will relay this information to DOT which will modify the launch systems accordingly.

SPOT Code

The user interface was written in QuickBasic instead of FORTRAN to provide a friendly work environment by offering a system of menus that can be easily navigated by the user. The menu-driven interface offers the option of optimizing any of three parameters: the size of liquid engines per stage, the number of solid engines per stage, or the number of strap-on engines on the first stage. Those variables not targeted for optimization are assigned values by the operator. After the optimization parameters are set, values for the desired orbital altitude, angle of inclination, launch site, and payload weight must be entered. SPOT then exits the user interface and begins executing the code within the main program.

Main Program

The main program serves as the heart of the SPOT project by acting as the driver for the DOT, liquid, solid, and orbital modules. The data returned by these modules is combined with the input values from the data files created by the user interface. This combination is then developed into the objective and constraint functions for the optimization routine. In this case the objective function describes the total vehicle weight. This function is evaluated subject to the system constraints within the DOT routine.

The DOT routine iterates the objective function in search of an optimum value by adjusting the system design variables. The design variables are the number of solid engines, the liquid fuel mass, and the number of liquid engines. When DOT arrives at the optimum configuration, it returns an array to the main program. This array contains the values of the optimized variables. The main program then interprets the DOT array into the optimum mass values for the specified launch sequence.

Liquid Module

The first module called by the main program is the liquid module, which determines the total initial mass of the liquid stages. In order to accomplish this task it must first receive information pertaining to the engine type of each stage, the number of engines used in each stage, and the propellant mass. The routine begins by determining which of the four possible stages were assigned liquid engines. After the liquid stages have been identified, the type of engine on each stage must be known. This allows a set of standard design variables characteristic to each engine type to be initialized. These variables include miscellaneous mass, fuel flow rate, and tank length-to-width ratio among others. The module is now ready to begin its calculations.

The first computation determines the stage burnout time in terms of the fuel flow rate and propellant mass. After the burnout time is found, the volumes of oxidizer and fuel are found. These values allow the respective storage tanks to be sized and their weight computed.

In computing the mass of the fuel and oxidizer tanks the routine takes into account the stresses under a worst-case scenario of 12 G's. This, coupled with a design safety factor of 1.5, is used to design the tank thickness using the equation for hoop stress given below.

$$t = pr/\sigma$$

where t = tank thickness
p = tank pressure
r = tank radius
σ = hoop stress

The total mass of each liquid stage is determined by adding the propellant mass (which is time-variant), the fuel and oxidizer tank masses, and the miscellaneous mass characteristic to the type of engines used. The total mass of the liquid stages is then determined by summation of the individual stage masses. This value is then returned to the main program.

Solid Module

After the initial mass of the liquid stages is calculated, the main program calls the solid module to perform a similar task. The determination of the initial mass of the solid stages is less complicated than that of the liquid engines because the mass of the solid propellant is not a design variable. The routine begins by identifying which stages were assigned solid engines and the type of motors they were designated. Since the propellant mass is a constant dependent on engine type, it is necessary only to multiply the number of engines present by the mass of each engine to determine the initial stage mass. The total initial mass of the solid stages is then simply the summation of the individual initial stage masses. This value can then be returned to the main program for use in the objective function.

In addition to calculating the initial solid engine mass, this module must also determine the time to burnout of each solid stage. This value is simply read from the data file for the corresponding solid engine type.

Orbital Module

The orbital module is concerned with two phases in the rocket's flight, the first being from launch until final stage burnout, and the second being from burnout to orbit (coast). During the first phase the specifics of the flight path are evaluated using the trajectory subroutine. It begins by gathering the values for drag force, thrust, and the system mass at an instantaneous moment during the powered flight of the vehicle. The radial and angular thrust are then determined using the relations below.

$$F_\mu = 1\,(m)(g/r)^2 - F_d(\cos\beta) + \tau(\cos\beta)$$

$$F_\theta = -1(F_d)\,(\sin\beta) + \tau(\sin\beta)$$

where
- m = total mass of vehicle
- g = gravity
- r = distance of vehicle from center of Earth
- F_d = drag force
- τ = rocket thrust
- β = angle of inclination

The next step in the solution process is to determine the radial and angular position, velocity, and acceleration. A Runge-Kutta subroutine is used to determine the velocities and position that are, in turn, used to calculate the acceleration values. This is done for time increments until the final stage burnout occurs. At this point the responsibility for the analysis of the vehicle's flight is transferred to the coast subroutine. This routine also used the general Runge-Kutta routine with the one exception of a thrust value of zero. The velocity and position of the vehicle are continually calculated until either the desired orbit is reached, or the vehicle comes to rest. If the rocket fails to reach the desired orbit, a warning message is returned to the main program.

The orbital module employs the services of three additional subroutines in its computational process. First, an aerodynamics routine is used to evaluate the instantaneous drag force experienced by the vehicle. Next, the orbital module calls the liqminor routine to determine the thrust delivered if a liquid engine is burning. In addition, it calculates the instantaneous mass of the liquid stages. Finally, subroutine solminor is called to find the thrust delivered as the solid engine is burning, as well as the instantaneous solid stage mass.

Conclusion

The driving concept behind SPOT is to provide the aerospace industry with a useful tool with which to evaluate launch and hardware configurations. While the actual program still contains some rough spots, the concept is very solid. With some additional work and modification, we will have achieved our objective and developed a useful design tool.