Suboptimal Path Planning and Control for AERCam/Sprint: Final Report *

J.C. Trinkle and S. Sudarsky
Department of Computer Science
College Station, TX 77843-3112

July 6, 1998

Abstract. This paper describes a technique for planning collision-free, fuel-efficient paths and thruster controls for moving Sprint from an initial to a goal location and attitude among multiple obstacles. The planning problem is formulated as a nonlinear optimization problem whose parameters are the locations of control points of splines representing Sprint's path and the time interval between the control points. Using Matlab as a test bed, many paths were found for an environment designed to approximate a space shuttle.

Key Words. Uniform nonrational B-splines, inverse dynamics, internal force, positive definite, linear programming, motion planning.

*This work was partially funded by grants from NASA/JSC (NAG9-893) and NSF (IRI-9334047). Any recommendations and conclusions expressed herein, are not necessarily those of the funding agencies. The authors would like to thank John Junkins and Hanspeter Schaub for helpful discussions on topics related to this work.

1 Introduction

NASA JSC is developing a "flying" semi-autonomous "eyeball" to allow astronauts to visually inspect space vehicles or structures on orbit without the cost of a space walk. The "eyeball," Sprint, is primarily a camera enclosed in a padded spherical structure fitted with 12 thrusters and telemetry equipment. Its simplest mode of operation would require an astronaut to "fly" it using a joy-stick with limited visual feedback to "fly" Sprint from one inspection point to another.

The objective of this project was to develop routines for planning safe, efficient, paths for Sprint to follow autonomously. Thereby freeing the astronaut for other tasks. The planner discussed herein can generate time- and energy-efficient, collision-free paths and associated thrust histories to desired configurations. Figure 1 below shows an approximation of a space shuttle formed by the union of three ellipsoids, and a possible path that would relocate Sprint from a position near the tail to near the middle of the fuselage. The arrows with their tails on the path are the angular velocity vectors of Sprint at the corresponding points along the path.

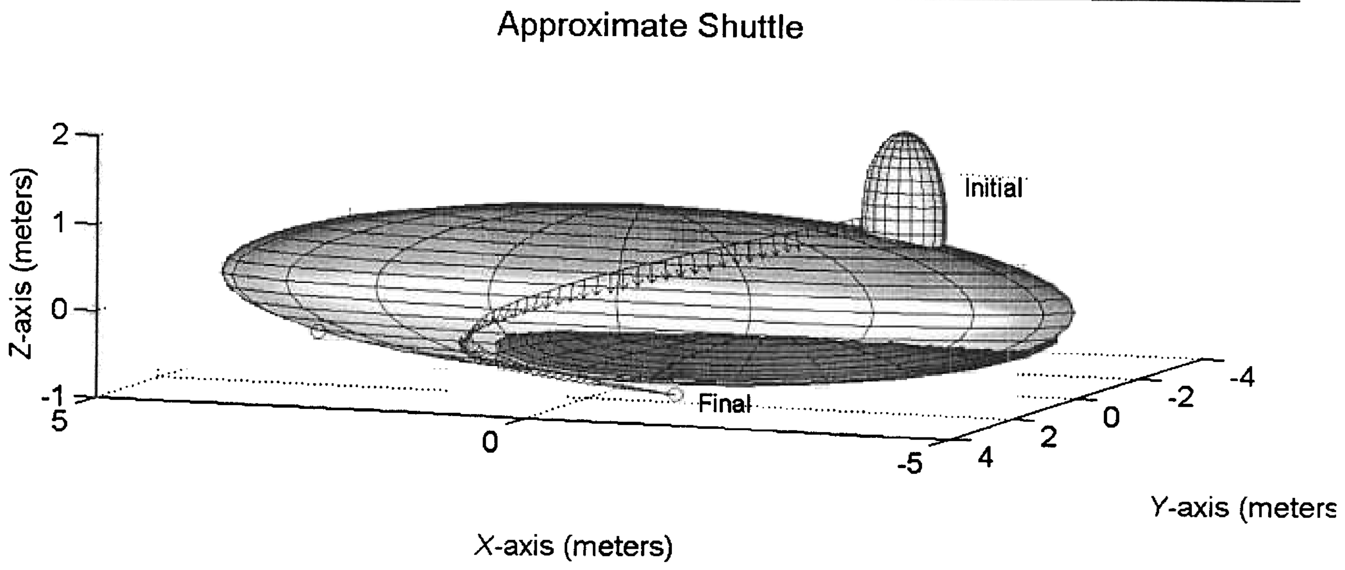

Figure 1: A smooth, collision-free path connecting two inspection locations.

In what follows, we will present the dynamic model of Sprint, a method for representing Sprint's position and attitude trajectories as sets of spline curves (one set for each of Sprint's 6 degrees of freedom), a method for computing thrust histories to follow a given sprint trajectory, an objective function to minimize to produce the optimal trajectory, simulated experimental results, and suggestions for further improvement of the method.

1.1 Dynamic Model of Sprint

Let the column vector $\boldsymbol{p}(t) = [x(t)\ y(t)\ z(t)\ ]^T$ (the superscript ^{T} denotes the transposition operator) represent the position trajectory of the center of mass of Sprint with respect to an inertial frame. Similarly, let the vector $\boldsymbol{\sigma}(t) = [\sigma_x(t)\ \sigma_y(t)\ \sigma_z(t)]^T$ represent the attitude of a right-handed coordinate frame fixed to Sprint with its origin coincident with Sprint's center of mass. These two vectors together completely specify the configuration of Sprint. Note that while any parameterization of attitude could be used, we have chosen to use modified Rodriguez parameters, because they provide a singularity-free representation in the range $\pm 360°$ [1]. The rotation matrix, $C(\boldsymbol{\sigma}) \in SO(3)$, describes the orientation of the body-fixed frame with respect to the inertial frame and can be written as follows:

$$C(\sigma) = \frac{1}{(1+\sigma^T\sigma)^2} \begin{bmatrix} 4(\sigma_x^2 - \sigma_y^2 - \sigma_z^2) + \Sigma^2 & 8\sigma_x\sigma_y + 4\sigma_z\Sigma & 8\sigma_x\sigma_z - 4\sigma_y\Sigma \\ 8\sigma_x\sigma_y - 4\sigma_z\Sigma & 4(-\sigma_x^2 + \sigma_y^2 - \sigma_z^2) + \Sigma^2 & 8\sigma_y\sigma_z + 4\sigma_x\Sigma \\ 8\sigma_z\sigma_x + 4\sigma_y\Sigma & 8\sigma_y\sigma_z - 4\sigma_x\Sigma & 4(-\sigma_x^2 - \sigma_y^2 + \sigma_z^2) + \Sigma^2 \end{bmatrix} \tag{1}$$

where sigma is defined as $\Sigma = (1 - \sigma^T\sigma)$.

In order to write the dynamic equations, let us define the generalized six-dimensional velocity, $\dot{q}(t)$. It is composed of two three-vectors representing the linear and angular velocity of Sprint, $v(t)$ and $\omega(t)$, defined with respect to the inertial frame. The linear velocity is just the time derivative of the position of the center of mass of Sprint. The relationship between the angular velocity ω and the time rate of change of the attitude parameters $\dot{\sigma}$ is more complicated:

$$\omega = B^{-1}\dot{\sigma} \tag{2}$$

where $B(\sigma)$ is defined as follows:

$$B(\sigma) = \frac{1}{4} \begin{bmatrix} (1 + \sigma_x^2 - \sigma_y^2 - \sigma_z^2) & 2(\sigma_x\sigma_y - \sigma_z) & 2(\sigma_x\sigma_z + \sigma_y) \\ 2(\sigma_x\sigma_y + \sigma_z) & (1 - \sigma_x^2 + \sigma_y^2 - \sigma_z^2) & 2(\sigma_y\sigma_z - \sigma_x) \\ 2(\sigma_z\sigma_x - \sigma_y) & 2(\sigma_z\sigma_y + \sigma_x) & (1 - \sigma_x^2 - \sigma_y^2 + \sigma_z^2) \end{bmatrix} \tag{3}$$

Finally, the generalized velocity can be written as:

$$\dot{q}(t) = \left[\begin{array}{cc} v^T(t) & \omega^T(t) \end{array}\right]^T \tag{4}$$

Similarly, the generalized acceleration $\ddot{q}(t)$ can be written as:

$$\ddot{q}(t) = \left[\begin{array}{cc} \dot{v}^T(t) & \dot{\omega}^T(t) \end{array}\right]^T \tag{5}$$

where $\dot{\omega}$ is given by:

$$\dot{\omega} = B^{-1}(\ddot{\sigma} - \dot{B}\omega). \tag{6}$$

Sprint's equations of motion can now be written in the following form:

$$M\ddot{q} - h(q, \dot{q}) = DWc \tag{7}$$

$$c \geq 0 \tag{8}$$

where c is n_t-vector of (unidirectional) thrusts with one element for each thruster (note that $n_t = 12$ for the current version of Sprint, but a smaller number could still control Sprint effectively), W is the (6×12) wrench matrix that maps the thruster forces into forces and moments expressed with respect to the body-fixed frame, the (6×6), block diagonal mass matrix computed with respect to the inertial frame is given by $M = diag\{mE, CIC^{-1}\}$, where E is the (3×3) identity matrix and m is the mass of Sprint, D is the block diagonal matrix defined as $D = diag\{C, C\}$, and h is the vector of velocity product terms defined as:

$$h = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\omega \times CIC^{-1}\omega \end{bmatrix} \tag{9}$$

where $\boldsymbol{I}$ is the constant (3×3) inertia tensor computed with respect to the body-fixed frame. data provided in [2], $m = 15.69kg$, and $\boldsymbol{W}$ and $\boldsymbol{I}$ are:

$$\boldsymbol{W} = \begin{bmatrix} 1.0 & 1.0 & -1.0 & -1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & -1.0 & -1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & -1.0 & -1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -.102 & .102 & .102 & -.102 \\ -.102 & .102 & .102 & -.102 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -.102 & .102 & .102 & -.102 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{10}$$

where the elements in the first three rows are dimensionless and those in the last three rows have units of meters, and

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

where $I_{xx} = 0.159kg\,m^2$, $I_{yy} = 0.168kg\,m^2$, $I_{zz} = 0.156kg\,m^2$, $I_{xy} = 0.0043$, $I_{xz} = -0.0040$, and $I_{yz} = -0.0060kg\,m^2$.

Note that the dynamic equations defined here assume that the structure to be avoided is fixed in inertial space. The effects due to the fact that the structure is actually in orbit are not accounted for here, nor are they in our implementation. However, including these effects (by augmenting the dynamic model with the well-known CW equations) will not affect the performance of the optimization scheme.

1.2 Trajectory Generation

In order to find the "best" trajectory for Sprint, we need a compact method for representing trajectories and a means to evaluate each one. We have chosen to represent trajectories as cubic, uniform, nonrational B-splines. As such, a trajectory is completely specified by a small set of control points and a time interval between the points. These splines are continuous in their first and second derivatives (assuming that consecutive control points are distinct, as they are for our application). Trajectory evaluation will be discussed in the section "Trajectory Optimization."

Each degree of freedom of Sprint's configuration will be represented by n_s spline segments. For example, given $n_s + 3$ control values $[x_{-2}\ x_{-1}\ \ldots\ x_{n_s}]$, the x-component of Sprint's configuration trajectory is defined as follows:

$$x(t) = \sum_{k=1}^{n_s} x_k(t)$$

where

$$x_k(t) = \begin{cases} \boldsymbol{b}_k \boldsymbol{Q} \boldsymbol{u}_k & t_k \leq t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$x_k(t)$ is the k^{th} spline segment and $\boldsymbol{u}_k^T = [(t-t_k)^3\ \ (t-t_k)^2\ \ (t-t_k)\ \ 1]$ The control values for $x_k(t)$ are $\boldsymbol{b}_k = [x_{k-3}\ x_{k-2}\ x_{k-1}\ x_k]$, and the weighting matrix $\boldsymbol{Q}$ is:

$$\boldsymbol{Q} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Given $n_s + 3$ positional control points, $\boldsymbol{P} = [\boldsymbol{p}_{-2} \cdots \boldsymbol{p}_{n_s}]$, the trajectories for all three translational degrees of freedom of sprint can be written as follows:

$$\boldsymbol{p}(t) = \sum_{k=1}^{n_s} \boldsymbol{p}_k(t)$$

where

$$\boldsymbol{p}_k(t) = \begin{cases} \boldsymbol{P}_k \boldsymbol{Q} \boldsymbol{u}_k & t_k \leq t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

and $\boldsymbol{P}_k$ is

$$\boldsymbol{P}_k = \begin{bmatrix} x_{k-3} & x_{k-2} & x_{k-1} & x_k \\ y_{k-3} & y_{k-2} & y_{k-1} & y_k \\ z_{k-3} & z_{k-2} & z_{k-1} & z_k \end{bmatrix}$$

Similarly, given $n_s + 3$ attitudinal control points, $\boldsymbol{S} = [\boldsymbol{\sigma}_{-2} \ldots \boldsymbol{\sigma}_{n_s}]$, the trajectories for all three rotational degrees of freedom can be written as:

$$\boldsymbol{\sigma}(t) = \sum_{k=1}^{n_s} \boldsymbol{\sigma}_k(t)$$

where

$$\boldsymbol{\sigma}_k(t) = \begin{cases} \boldsymbol{A}_k \boldsymbol{Q} \boldsymbol{u}_k & t_k \leq t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\boldsymbol{A}_k = \begin{bmatrix} (\sigma_x)_{k-3} & (\sigma_x)_{k-2} & (\sigma_x)_{k-1} & (\sigma_x)_k \\ (\sigma_y)_{k-3} & (\sigma_y)_{k-2} & (\sigma_y)_{k-1} & (\sigma_y)_k \\ (\sigma_z)_{k-3} & (\sigma_z)_{k-2} & (\sigma_z)_{k-1} & (\sigma_z)_k \end{bmatrix}$$

From these cubic splines, we can determine linear velocity and acceleration of the center of gravity of Sprint, $\dot{\boldsymbol{p}}(t)$ and $\ddot{\boldsymbol{p}}(t)$ as:

$$\begin{aligned} \dot{\boldsymbol{p}}(t) &= \sum_{k=1}^{n_s} \dot{\boldsymbol{p}}_k(t) \\ \ddot{\boldsymbol{p}}(t) &= \sum_{k=1}^{n_s} \ddot{\boldsymbol{p}}_k(t) \end{aligned}$$

where

$$\dot{\boldsymbol{p}}_k(t) = \begin{cases} \boldsymbol{P}_k \boldsymbol{Q} \dot{\boldsymbol{u}}_k & t_k \leq t < t_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

and

$$\ddot{\boldsymbol{p}}_k(t) = \begin{cases} \boldsymbol{P}_k \boldsymbol{Q} \ddot{\boldsymbol{u}}_k & t_k \leq t < t_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

The angular velocity and acceleration of sprint can be computed using equations (2), (6), (25), and (26):

$$\begin{aligned} \dot{\boldsymbol{\sigma}}(t) &= \sum_{k=1}^{n_s} \dot{\boldsymbol{\sigma}}_k(t) \\ \ddot{\boldsymbol{\sigma}}(t) &= \sum_{k=1}^{n_s} \ddot{\boldsymbol{\sigma}}_k(t) \end{aligned}$$

where

$$\dot{\sigma}_k(t) = \begin{cases} A_k Q \dot{s}_k & t_k \le t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\ddot{\sigma}_k(t) = \begin{cases} A_k Q \ddot{s}_k & t_k \le t < t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

1.3 Thrust History Computation

Given a set of control points and a time interval, the position and attitude trajectories and their first and second time derivatives can be computed as indicated above. From these trajectories, one can determine $\boldsymbol{D}(t)$, $\boldsymbol{M}(t)$, $\boldsymbol{h}(t)$, and $\ddot{q}(t)$. Substituting these functions into the dynamic equation (7) and rearranging yields the resultant thruster force and moment (the product $\boldsymbol{Wc}$) as functions of time:

$$\boldsymbol{D}^{-1}(\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{h}) = \boldsymbol{Wc}.$$

Note that the resultant force and moment are represented with respect to Sprint's body-fixed frame. Assuming no errors, if these resultant force and moment trajectories can be achieved by the coordinated control of the thrusters, then Sprint will follow the trajectory represented by the given points and time interval.

Since $\boldsymbol{D}^{-1}$ exists and is unique, the resultant force and moment to be applied by the thrusters is known uniquely. However, what is needed, is the vector of thrust histories, $\boldsymbol{c}(t)$. Fortunately, $\boldsymbol{W}$ is full row rank, so a solution for $\boldsymbol{c}$ exists for every time t, but $\boldsymbol{W}$ has a nontrivial null space, so the solution is not unique. This nonuniqueness provides the opportunity to determine the solution which minimizes the rate of fuel expenditure at each instant of time, and thus minimizes the total fuel use for a given set of splines defining the motion of Sprint.

The general solution for the thrust vector is:

$$\boldsymbol{c}(t) = \boldsymbol{W}^{+}\boldsymbol{D}^{-1}(\boldsymbol{M}\ddot{\boldsymbol{q}} \quad \boldsymbol{h}) + \boldsymbol{N}\boldsymbol{\xi}$$

where $\boldsymbol{W}^{+}$ is the generalized inverse of $\boldsymbol{W}$, $\boldsymbol{N}$ is a matrix whose columns form a basis for the null space of $\boldsymbol{W}$, and $\boldsymbol{\xi}$ is an arbitrary vector function of time.[1] The first term on the left hand side of this equation is known as the particular solution $\boldsymbol{c}_p = \boldsymbol{W}^{+}\boldsymbol{D}^{-1}(\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{h})$. It is responsible for ensuring that the thrusters' resultant force and moment are correct. The second term, the homogeneous solution, does not contribute to the resultant force and moment. Rather, varying $\boldsymbol{\xi}$ effects only the internal stresses generated, which of course, should be minimized.

For a given thruster, the rate of fuel expenditure is proportional to the thrust. Therefore, the desired vector of thrusts at any instant of time, is the one that minimizes the sum of the elements of $\boldsymbol{c}$, while still achieving the correct resultant force and moment and satisfying $\boldsymbol{c} \ge 0$ (since the thrusters are unidirectional). These constraints define the following linear program in $\boldsymbol{\xi}$:

$$\begin{aligned} \text{Minimize:} \quad & \boldsymbol{n}^T\boldsymbol{\xi} \\ \text{subject to:} \quad & \boldsymbol{N}\boldsymbol{\xi} \ge -\boldsymbol{c}_p \end{aligned} \tag{31}$$

where $\boldsymbol{n}^T$ is the row vector formed by summing all the rows of $\boldsymbol{N}$, and the variable $\boldsymbol{\xi}$ is unconstrained. Solving this linear program at a given instant of time yields the most fuel-efficient thrusts for that time instant. In the present implementation, the position and attitude trajectories are

[1]We remind the reader that all terms in equation (30) with the exception of $\boldsymbol{W}$ are functions of time. However, we have abandonded showing this dependence explicitly for brevity.

discretized in time and the above linear program is solved at each time. This approach is simple to implement, but considerably less efficient than an approach based on sensitivity theory of linear programming.

1.4 Trajectory Optimization

We formulate the planning problem as a nonlinear optimization problem whose parameters are the elements of the control points contained in $\boldsymbol{P}$ and $\boldsymbol{S}$ and the time interval between the control points, $\Delta = t_{k+1} - t_k$. Our objective is to find the spline curves $\boldsymbol{q}(t;\, \boldsymbol{P}, \boldsymbol{S}, \Delta)$ with the following attributes:

1. The flight time of Sprint should be short.
2. Sprint may not collide with the structure.
3. The fuel consumption should be small.
4. The resultant force and moment required should never exceed what can be produced by the thrusters.

Trajectories minimizing the objective function ϕ (defined next) will exhibit these attributes:

$$\begin{aligned}
\phi(t, \boldsymbol{\alpha}) \;&=\; w_1 (n_s\Delta)^2 \;-\; w_2\rho \;+\; \int_0^{n_s\Delta} \boldsymbol{g}(t)^T \boldsymbol{R}_1\, \boldsymbol{g}(t)\, dt \\
&+\; \int_0^{n_s\Delta} w_3 \max\left((\boldsymbol{f}(t)^T \boldsymbol{f}(t) \;-\; ||\boldsymbol{f}||^2_{max}) \right) dt \\
&+\; \int_0^{n_s\Delta} w_4 \max\left((\boldsymbol{\tau}(t)^T \boldsymbol{\tau}(t) \;-\; ||\boldsymbol{\tau}||^2_{max}) \right) dt.
\end{aligned}$$

where w_1, w_2, w_3, and w_4 are nonnegative weights, $\boldsymbol{f}(t)$ and $\boldsymbol{\tau}(t)$ are the resultant force and moment trajectories, $\boldsymbol{g}$ is the six-vector of resultant generalized force, *i.e.*, $\boldsymbol{g} = [\boldsymbol{f}^T \boldsymbol{\tau}^T]^T$, and $||\boldsymbol{f}||_{max} = 0.49N$ and $||\boldsymbol{\tau}||_{max} = 0.0049Nm$ are the norms of the (approximate) maximum resultant force and moment of the thrusters. The term multiplied by w_1 is the square of the total traversal time. Increasing w_1 tends to generate trajectories that require less time to execute, but use more fuel. The term multiplied by w_2 is the collision penalty (described below in detail). A set of splines that would result in collision would have a negative value of ρ. Thus collisions work against minimization. Large values of w_2 imply that collision avoidance is very important. In the first integral term, $\boldsymbol{R}$ is a positive definite matrix that weights the relative importance of the resultant force and moment applied by the thrusters. While it is not a direct measure of fuel expenditure, larger values of $\mathbf{c}$ increase this integral and the objective function. Thus when $\boldsymbol{R}$ has large (eigen-) values fuel economy becomes very important. The second and third integral terms are the differences between the amounts of resultant force and moment required by the splines and that which can be produced by the thrusters. These terms only inflict a penalty if the required force and moment are more than can be produced. Also, note that this is an approximate measure since in our chosen optimization approach, we do not work with the thrusts directly. Nonetheless, the term produces the desired effect as will be seen in the results following this section.

The penetration penalty function ρ is defined as follows. Let D be a set of set of N test times distributed in the interval $[0, n_s\Delta]$, with elements denoted by δ_j, $j = 1 \ldots N$, and let n_e be the number of ellipsoids whose union form the geometric model of the structure. Define the penalty

$\rho_{ij}(\delta_j)$ as follows:

$$\rho_{ij} = \min\left(0, \left(\frac{x_j - a_{ix}}{r_{ix}}\right)^2 + \left(\frac{y_j - a_{iy}}{r_{iy}}\right)^2 + \left(\frac{z_j - a_{iz}}{r_{iz}}\right)^2 - 1\right)$$

where a_{ix}, a_{iy}, and a_{iz} are the offsets of the center of the center of the i^{th} ellipsoid, and r_{ix}, r_{iy}, r_{iz} are the lengths of the principle radii of the i^{th} ellipsoid, and $x_j = x(\delta_j)$, $y_j = y(\delta_j)$, and $z_j = z(\delta_j)$ are the coordinates of a test point on the position spline to be tested for intersection with the structure. Note that ρ_{ij} is negative when the test point is inside the i^{th} ellipsoid and zero otherwise. The penalty, ρ can now be written as:

$$\rho = \sum_j \min(0, \rho_{1,j}, \rho_{2,j}, \quad , \rho_{N,j}).$$

Thus we see that ρ is the sum of the "deepest" penetrations in each of the ellipsoids for each of the N penetration test times.

One other relevant point is that the trajectory found by the optimization procedure can be scaled easily. If, for example, the path for Sprint is acceptable, but more fuel savings is desired (at the expense of more traverse time), the time can be scaled. This is an extremely simple procedure that takes only about 1 second of cpu time. If one increases the traverse time by a factor, the fuel use is (approximately) reduced by the same factor. Thus tripling the traverse time will reduce the fuel requirement to one third of the original requirement.

2 Results

We have implemented our trajectory planning method described above in MATLAB 5.2 Release 10 using a quasi-Newton method for unconstrained minimization. The convergence parameters were set to a tolerance of 0.1 for changes of the coordinates of the control points and the time interval. In addition, the maximum number of evaluations of the objective function was set to 1200.

The examples below use a space structure approximated by the union of three ellipsoids. The equations of the ellipsoids represented in the inertial frame are:

$$f_i(x,y,z) = \left(\frac{x - a_{ix}}{r_{ix}}\right)^2 + \left(\frac{y - a_{iy}}{r_{iy}}\right)^2 + \left(\frac{z - a_{iz}}{r_{iz}}\right)^2 - 1; \quad \text{for } i = 1,2,3$$

where x, y, and z are the inertial coordinates, $a_{i\alpha}; \alpha \in \{x,y,z\}$, are the offsets from the origin of the inertial frame to the center of the ith ellipsoid in the α directions, and $r_{i\alpha}; \alpha \in \{x,y,z\}$ are the lengths of the primary axes of the i^{th} ellipsoid. A given point, $[x,y,z]$, is inside the i^{th} ellipsoid if $f_i(x,y,z)$ is negative, on the boundary if $f_i(x,y,z)$ is zero, and outside otherwise.

For the following examples the ellipsoid parameters were taken to be:

a_{1x}	=	0.0	a_{1y}	=	0.0	a_{1z}	=	0.0
a_{2x}	=	−1.5	a_{2y}	=	0.0	a_{2z}	=	−0.0
a_{3x}	=	−3.0	a_{3y}	=	0.0	a_{3z}	=	1.0
r_{1x}	=	5.0	r_{1y}	=	1.0	r_{1z}	=	1.0
r_{2x}	=	3.0	r_{2y}	=	4.0	r_{2z}	=	0.2
r_{3x}	=	0.5	r_{3y}	=	0.2	r_{3z}	=	1.0

These ellipsoids form a crude model of a space shuttle.

2.1 Example 1

In this example, a trajectory was planned parallel to the inertial x-direction along the fuselage also oriented along the inertial x-direction, but far enough away such that a straight-line path would not cause collisions. Sprint was to begin and end at rest. The initial and final configurations were defined as:

$$\begin{bmatrix} p_{init} \\ s_{init} \end{bmatrix} \begin{bmatrix} -3.0 \\ -2.0 \\ 1.1 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \qquad \begin{bmatrix} p_{final} \\ s_{final} \end{bmatrix} \begin{bmatrix} 3.0 \\ -2.0 \\ 1.1 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \tag{40}$$

Note that in both configurations, the three MRPs (modified Rodriguez parameters) are zero. In this case, the body axes of sprint are parallel to the axes of the inertial frame. These configurations were chosen for testing purposes, because intuitively, the optimal trajectory should be one which simply translates Sprint six meters in the x-direction. Since Sprint's x-direction thrusters are aligned with the direction of translation, we expected symmetrical firings of the x-direction thrusters and no firings of the other eight thrusters. As will be seen below, the trajectory found exhibits these characteristics.

The constraint that Sprint will begin and end at rest was enforced by placing the initial and final configurations in the matrices, $\boldsymbol{P}$ and $\boldsymbol{A}$, in triplicate as shown below:

$$\begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{A} \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} -3.0 & -3.0 & -3.0 & -1.5 & 0.0 & 1.5 & 3.0 & 3.0 & 3.0 \\ -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 \\ 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}\right]. \tag{41}$$

The initial and final configurations account for the first and last three columns of matrix in equation (41). These columns are fixed throughout the optimization procedure, since the only candidate trajectories must have Sprint at rest at the beginning and end. The interior three columns of the matrix are the coordinates of the three adjustable control points. These 18 values were adjusted by the optimization algorithm to minimize the objective function. The initial values in the three adjustable columns were found by linear interpolation between the initial and final configurations (as a convenient way to automatically initialize the optimization algorithm.

The remaining initial parameter required to start optimization is the time step between the control points, *i.e.*, the knot times. The program always set the initial value to 10 seconds. For a trajectory with three adjustable control points, 10 seconds corresponds to a trajectory traverse time of one minute. The knot vector for this problem is:

$$\boldsymbol{kt} = [0\ 10\ 20\ 30\ 40\ 50]. \tag{42}$$

Finding an optimal trajectory for this example required optimization over 19 parameters, the elements of the three adjustable control points, and the time interval between the control points. This was done several times using different weights in the objective function (33).

Example 1: First Run

For the first run, the weights were taken to be: $w_1 = 1$, $w_2 = 200$, $w_3 = 30$, $w_4 = 1$, and $\boldsymbol{R} = diag\{5, 5, 5, 1, 1, 1\}$. These values enforced a moderate penalty on collisions and exceeding thrust capacities, and placed a moderate penalty on large forces and long traverse times. The expected trajectory then is one that will use the close to the maximum available thrust yielding a relatively short traverse time. The remaining parameter needed to specify the objective function of the distribution of test times over along the trajectory to test for collisions. Our code used 10 test points on each spline segment, so for this example, with 6 spline segments, N was 60.

The results reported below were obtained in 274 seconds of cpu time on a Pentium PC running at 120 MHz. The optimization algorithm evaluated the objective function 285 times in finding the optimal path shown in Figure 2. This path is a straight line parallel to the fuselage, as expected. The initial and final positions are indicated by the circles at the end points. The three circles interior to the path are the positions of the optimal control points. Their values are given in the three central columns of the follow array:

$$\begin{bmatrix} \boldsymbol{P}^* \\ \boldsymbol{A}^* \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} -3.0 & -3.0 & -3.0 & -1.97 & -0.001 & 1.96 & 3.0 & 3.0 & 3.0 \\ -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 \\ 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}\right] \tag{43}$$

Nte the symmetry in the adjustable control points. The corresponding optimal value of the time interval between the control points was found to be 4.67 seconds.

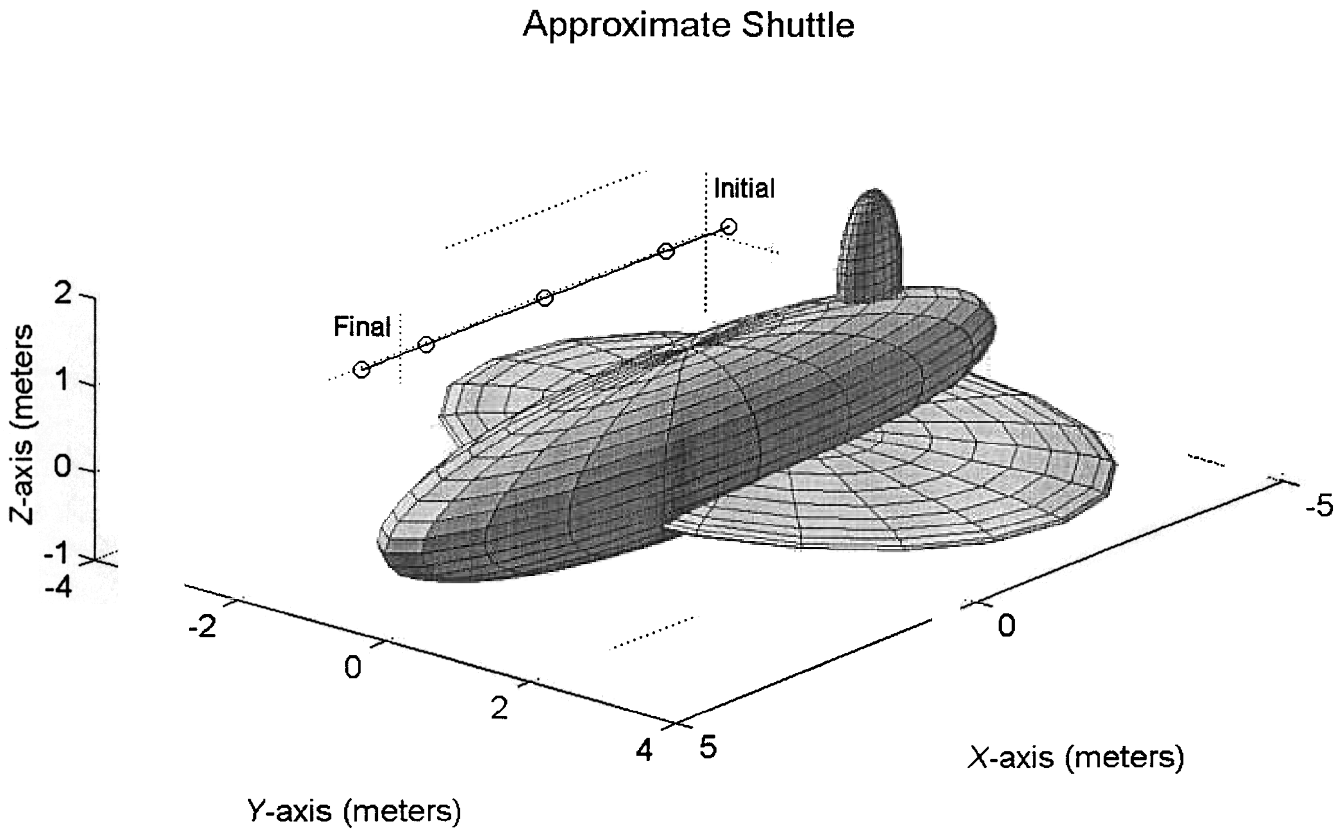

Figure 2: A straight-line path connecting two configurations.

Figure 3 shows that the x-coordinate of Sprint's position varies smoothly, monotonically increasing from its initial value of -3 to its final value $+3$ in 28 seconds. Further calculations projected fuel expenditure of $0.0185kg$, which is 6.6% of the fuel capacity, $0.281kg$. The y- and z-coordinates remain constant as would be expected, since any deviation from the straight line path would require additional fuel.

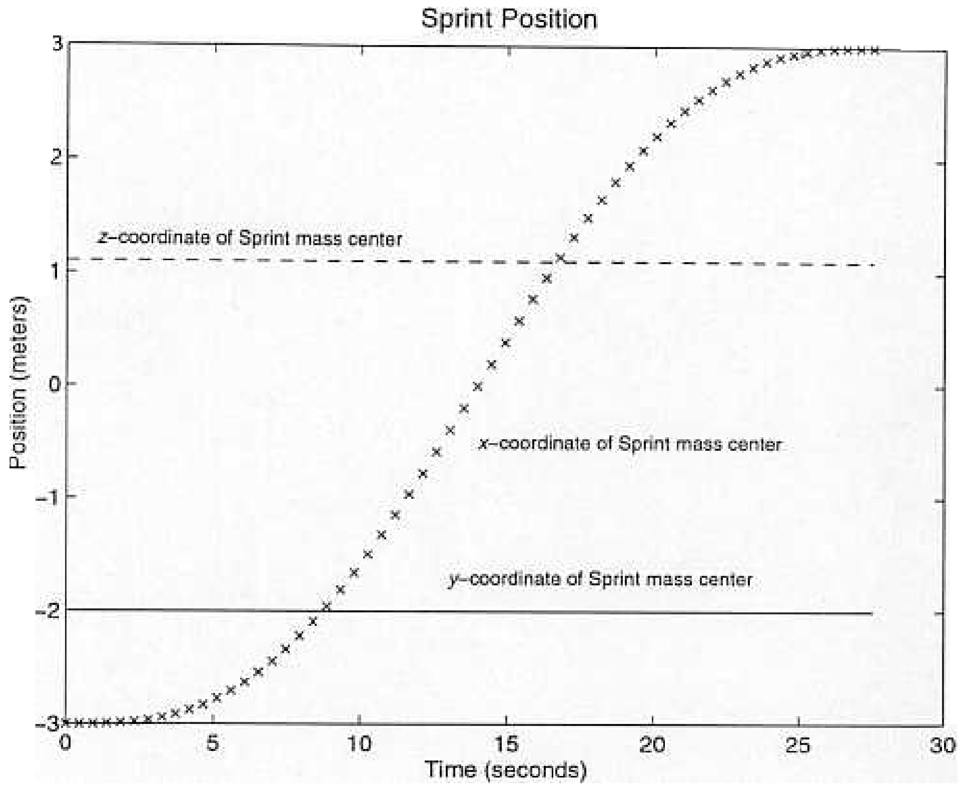

Figure 3: Sprint position coordinate trajectories.

Figure 4 shows Sprint's velocity components. Note that the x-component begins and ends at zero and varies smoothly and symmetrically between the end points. The y- and z- components are zero throughout the trajectory. Note that the position trajectories can be show to be piecewise-cubic curves. Therefore, the linear velocity trajectories must be piecewise-quadratic curves and the linear acceleration trajectories must be piecewise-linear curves (see Figure 5).

Since the linear acceleration of Sprint's center of mass is proportional to the resultant force of the thrusters, it is not surprising that the plots in Figure 6 are scaled versions of those in the previous figure. Notice that in the first half of the trajectory both aft x-direction thrusters are acting in unison to produce translational motion in the positive x-direction without producing any rotation. One of the thruster's values are represented by circles, the other by plus signs. During this time, the two forward x-direction thrusters are off. In the second half of the trajectory the forward thrusters act together to null the velocity while the aft thrusters are off. Note, however, that the maximum required thrust, $0.38N$, exceeds the $0.349N$ capacity of the thrusters. This is due to use of penalty terms in the objective function to limit thrust rather than using inequality constraints in the optimization algorithm. To correct the problem, one can scale the trajectory in time, increasing it by a factor of $\sqrt{\frac{0.38}{0.349}} = 1.04$. This would increase the traverse time from 28.0 to 29.2 seconds.

Finally, plots of the orientation, angular velocity and acceleration, and thrust histories from the

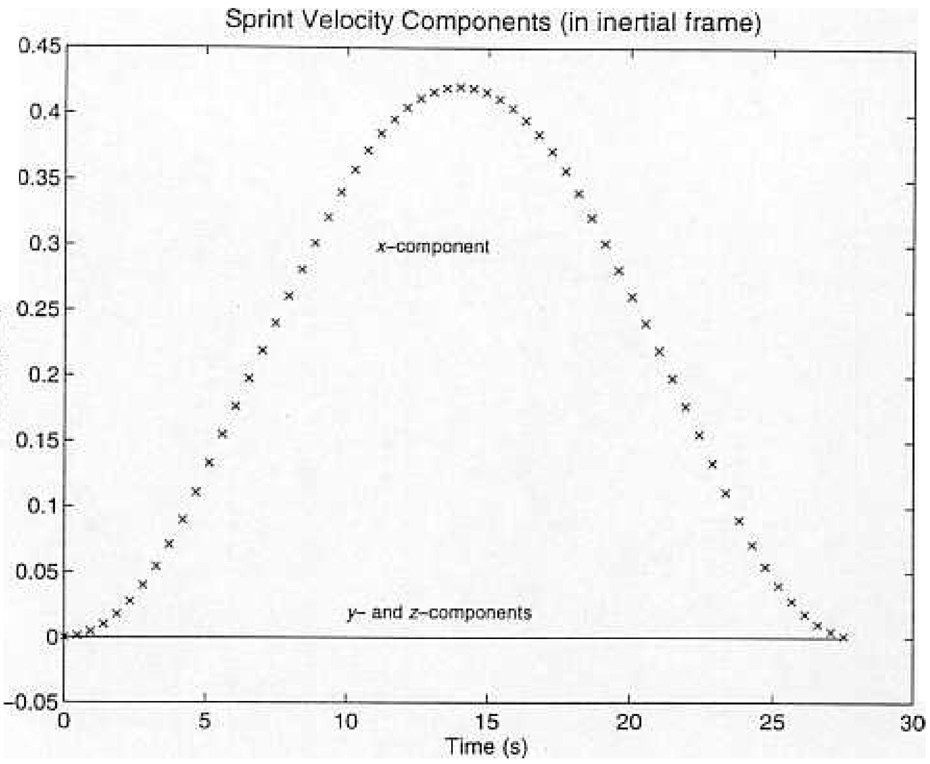

Figure 4: Sprint linear velocity component trajectories.

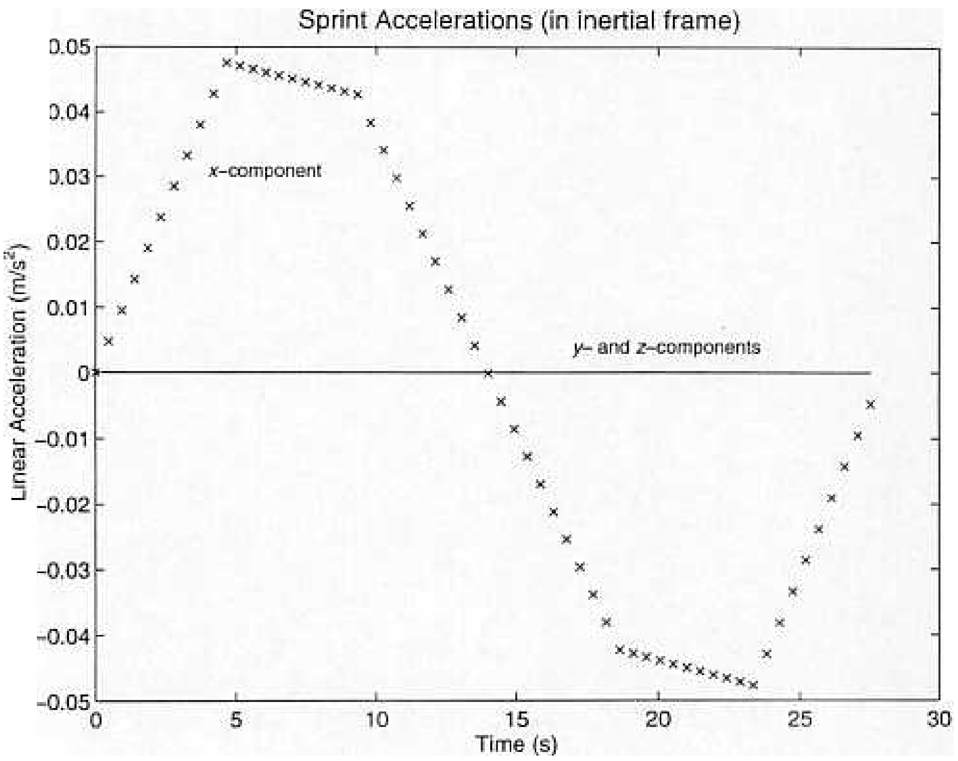

Figure 5: Sprint linear acceleration component trajectories.

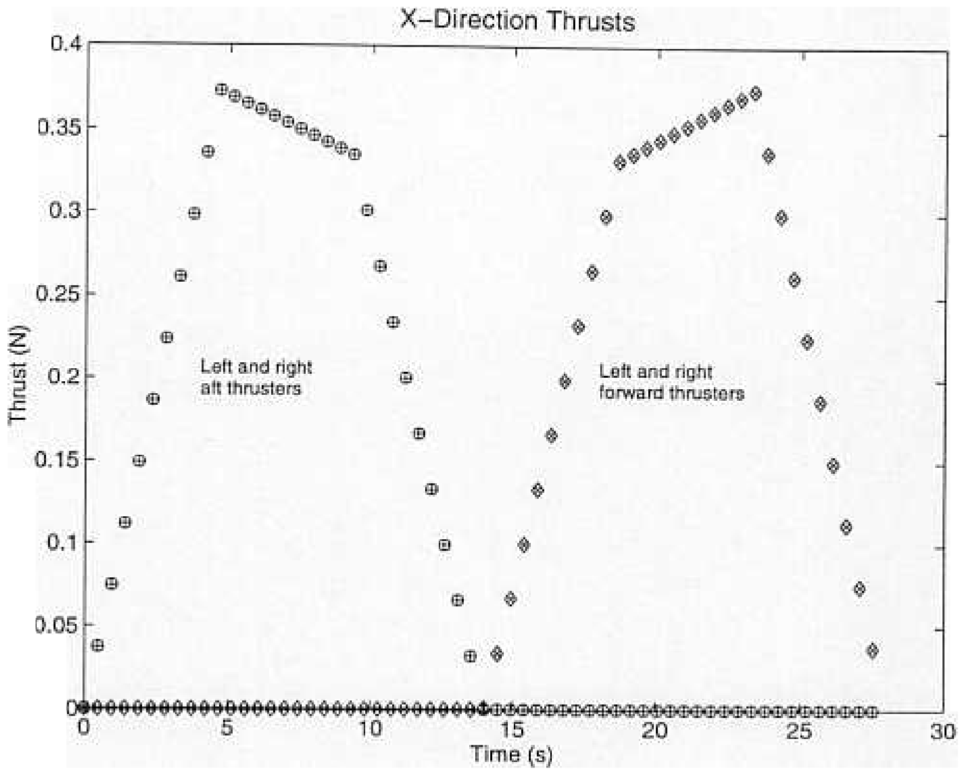

Figure 6: Sprint thruster trajectories.

other eight thrusters are not shown here, since they are identically zero throughout the trajectory; again, as expected.

2.1.2 Example 1: Second Run

Only one parameter was changed for this run: the weight, w_5, was changed from 5 to 500. This placed a large penalty on fuel usage. The optimizer converged after 109 function evaluations in 110 cpu seconds finding a trajectory with a 48.6 second traverse time (74% longer) and requiring just 0.0092 kilograms of fuel (50% less). None of the plots are shown here, because they are qualitatively similar to those for the first run. The primary differences of note are that the maximum linear velocity for the second run was reduced by 50% to $0.21 m/s$ and the maximum thrust was reduced by 58% to $0.16N$.

$$\begin{bmatrix} \boldsymbol{P}^* \\ \boldsymbol{A}^* \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} -3.0 & -3.0 & -3.0 & -1.68 & -0.00 & 1.68 & 3.0 & 3.0 & 3.0 \\ -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 & -2.0 \\ 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 & 1.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}\right] \tag{44}$$

The corresponding optimal value of the time interval control points was found to be 8.10 seconds.

2.2 Example 2

In this example, a trajectory was planned starting from beside the tail on the top of Sprint to under the wing on the opposite side. A straight-line path connecting these configurations would pass through the center of the structure. The weights used to obtain this trajectory were: $w_1 = 1$, $w_2 = 1000$, $w_3 = 30$, $w_4 = 1$, and $\boldsymbol{R} = diag\{10, 10, 10, 1, 1, 1\}$. This enforces a moderate penalty on collisions and on using more thrust than is The initial and final configurations were defined as:

$$\begin{bmatrix} \boldsymbol{p}_{init} \\ \boldsymbol{s}_{init} \end{bmatrix} = \begin{bmatrix} -3.0 \\ -0.4 \\ 1.1 \\ 0.2 \\ 0.2 \\ 0.7 \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{p}_{final} \\ \boldsymbol{s}_{final} \end{bmatrix} = \begin{bmatrix} -0.8 \\ 1.3 \\ -1.0 \\ 0.8 \\ 0.5 \\ 0.1 \end{bmatrix} \tag{45}$$

The path shown in Figure 7 was found in about 7 minutes of cpu time using only one adjustable control point, whose final optimal value was: $\boldsymbol{p}^* = [2.615\ 3.490\ -0.186]^T$ (shown as a circle near the nose of the shuttle.), and $\boldsymbol{s}^* = [0.498\ 0.349\ 0.401]^T$ Traversing this path will require $49s$ and less than 15% of the fuel capacity. As mentioned previously, the fuel expenditure could be approximately halved if the traverse time were doubled.

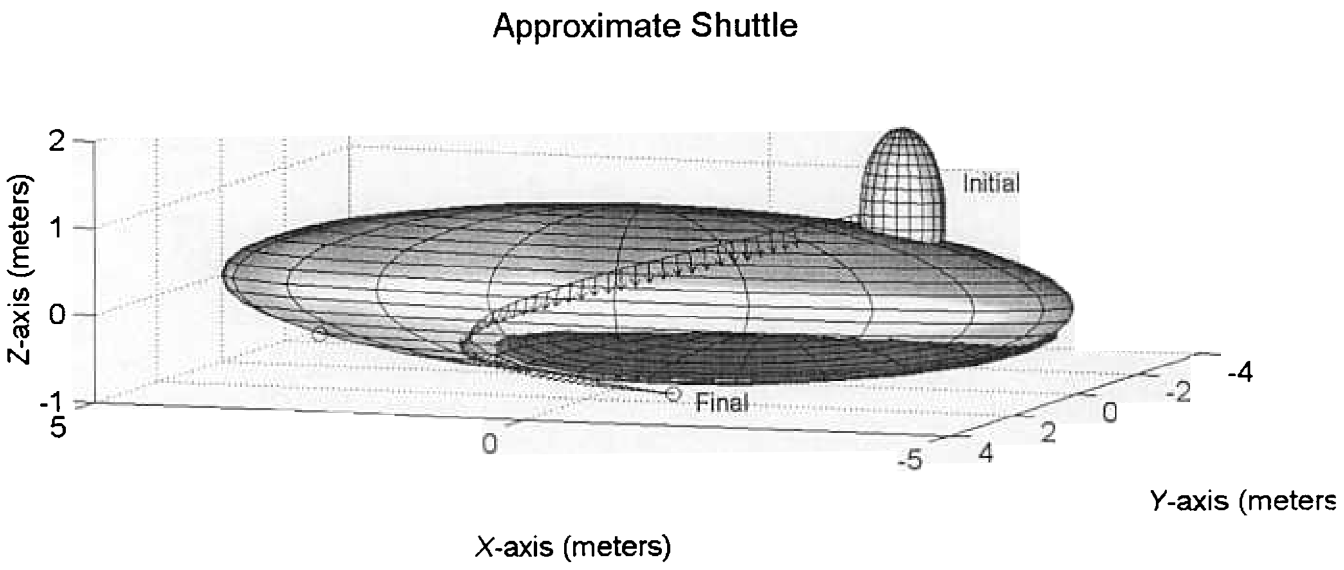

Figure 7: Trajectory looping around the fuselage and wing.

The remaining figures contain plots of various physical quantities for this trajectory. Only the most important aspects of which will be pointed out.

Note the nonmonotonic behavior of the x- and y-components of Sprint's position. This is due to the fact that the path must go around the wing and fuselage to avoid collisions.

The thrust plots show a mismatch between the aft and forward pairs of thrusters that was not present in the first example. This mismatch is due to the fact that Sprint must change its orientation, which can only be accomplished by applying a resultant moment. Note that the mismatch is greatest at the beginning and end of the trajectory implying that a large moment impulse is being used initially to initiate the proper rotation and then a large moment impulse is

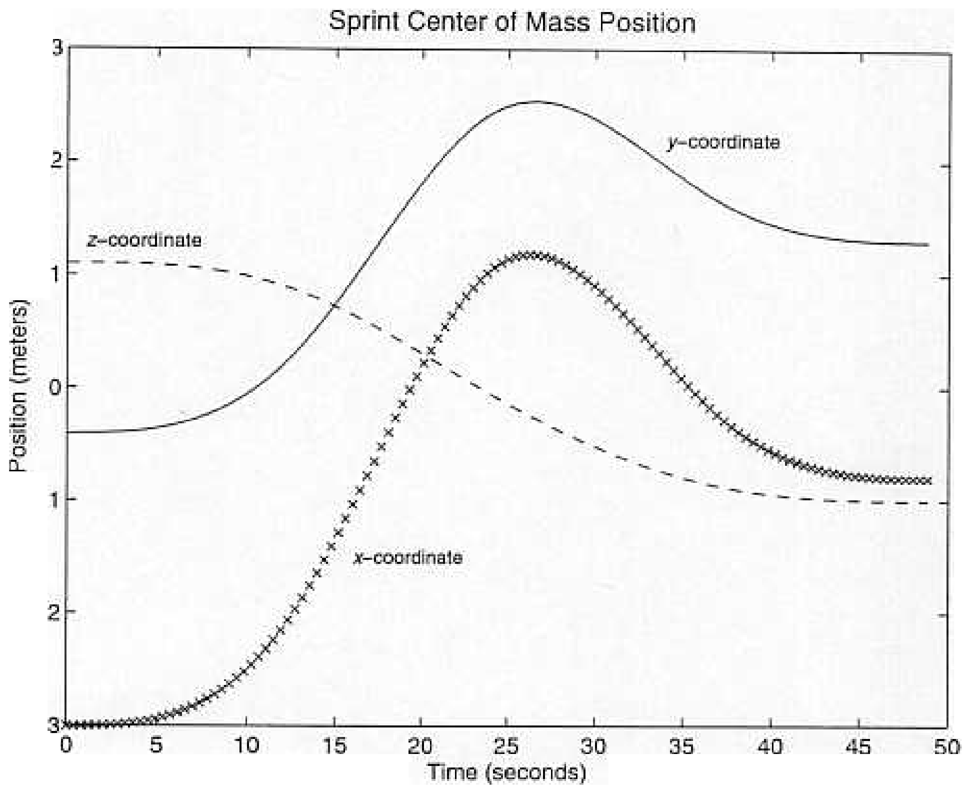

Figure 8: Sprint position component trajectories.

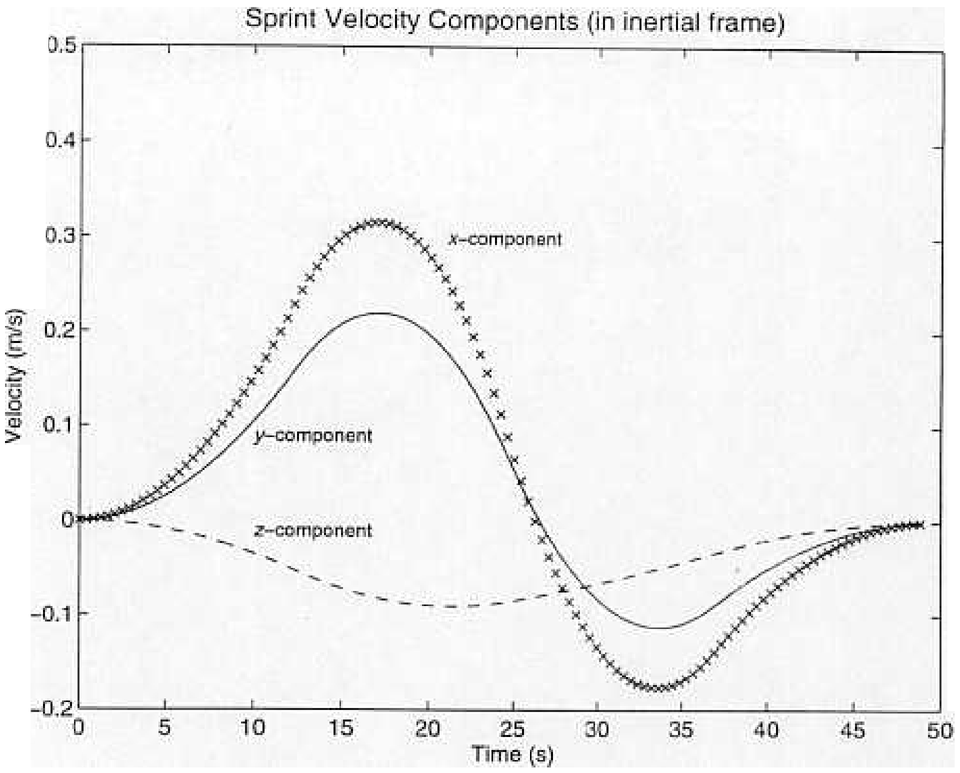

Figure 9: Sprint linear velocity component trajectories.

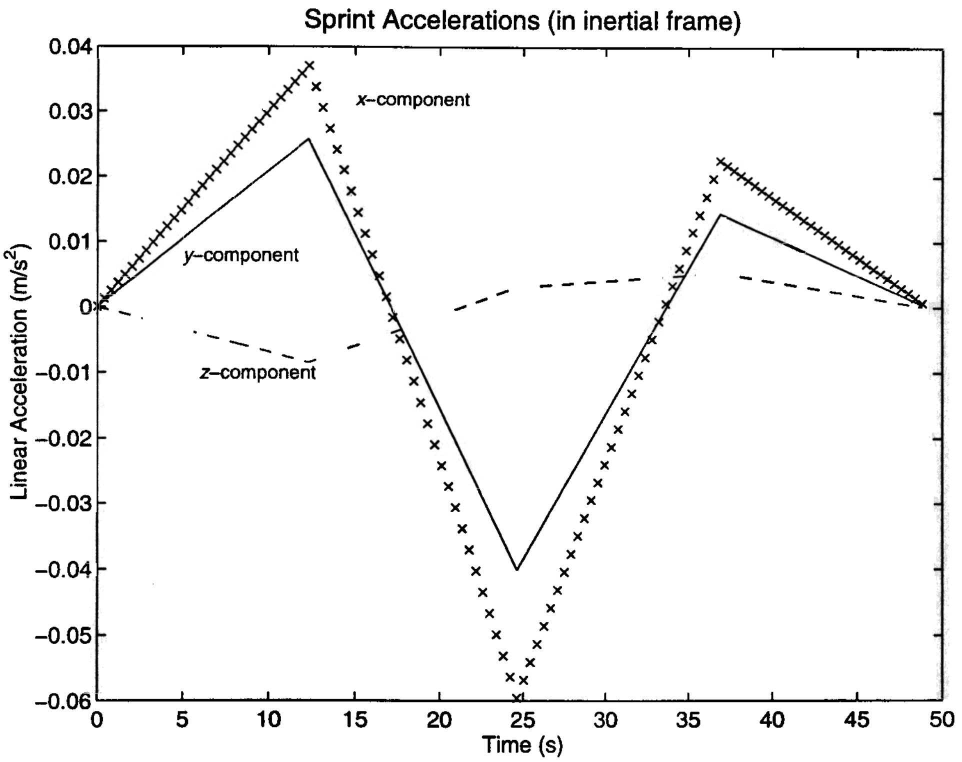

Figure 10: Sprint linear acceleration component trajectories.

applied again at the end of the trajectory to arrest the rotation. Also, the mismatches between the thrusters are distributed across the three thrust plot figures. The plots show that the rotation is initiated primarily by a pair of y-direction thrusters and arrested primarily by x-direction thrusters. This is explained by the fact that Sprint is reorienting during the trajectory.

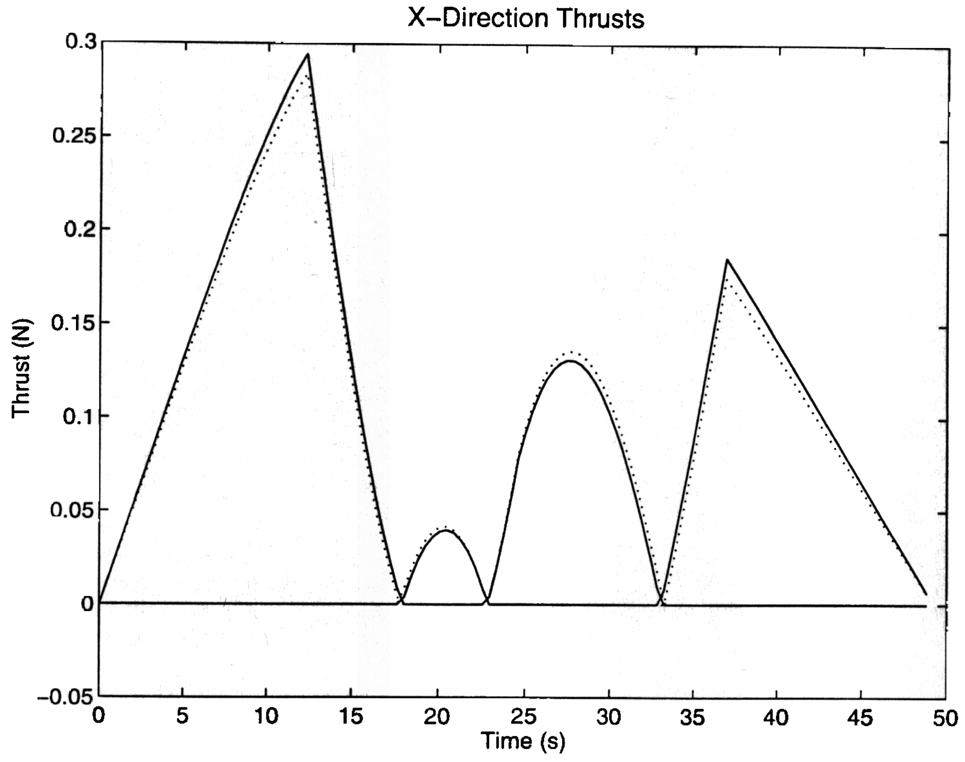

Figure 11: Trajectories for x-direction thrusters.

3 Possible Improvements

Numerous improvements could be made to the planning software, but even without improvements, one could easily find a computer that would run the Matlab code 5 times faster than a Pentium/120 used for the experiments discussed here. For further improvement, the planner could be coded carefully in C to obtain another factor of 10 speed-up. With a speed-up factor of 50, nearly all plans would complete in less than 30 seconds. Even complicated ones requiring three to five control points.[2] It would be useful to automate the selection of the collision weight w_2 in the objective function and N the number of collision test points, as it sometimes takes several runs with increasing value of w_2 to generate a path that avoids collisions.

It would also be possible to create a user interface to allow direct interaction with the splines and control points in real-time. The benefit would derive from using a user's intelligence and experience to generate good initial guesses for the locations of the control points, and to allow adjustments of the trajectory after convergence of the optimization algorithm.

[2]In numerous experiments in simple environments, we have never found a trajectory with four control points that could be significantly improved with five control points.

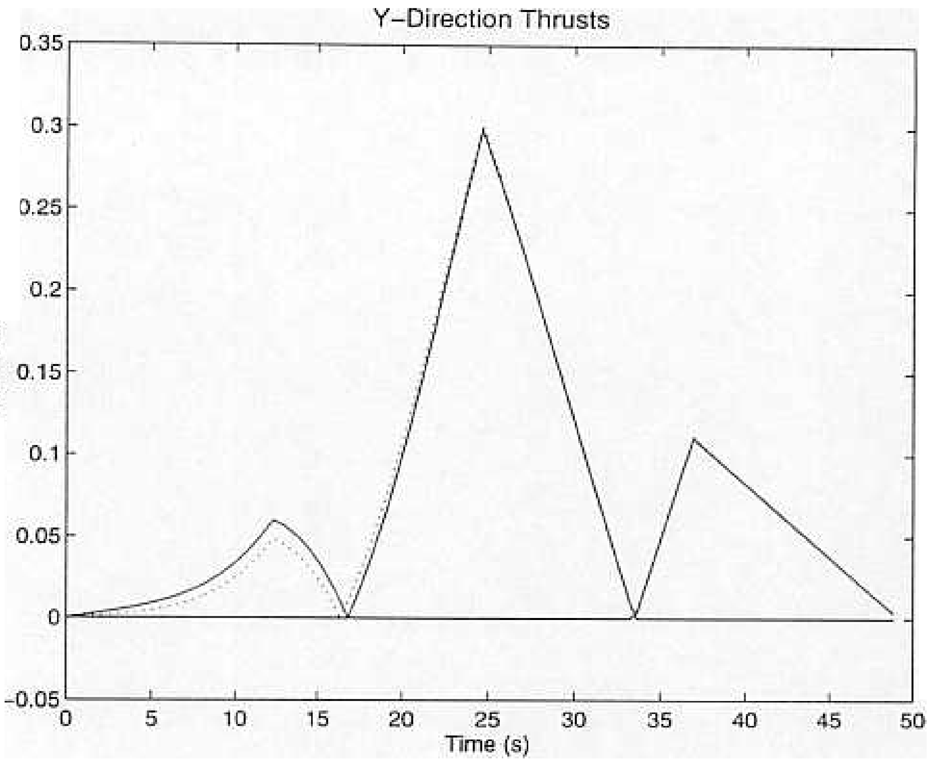

Figure 12: Trajectories for y-direction thrusters.

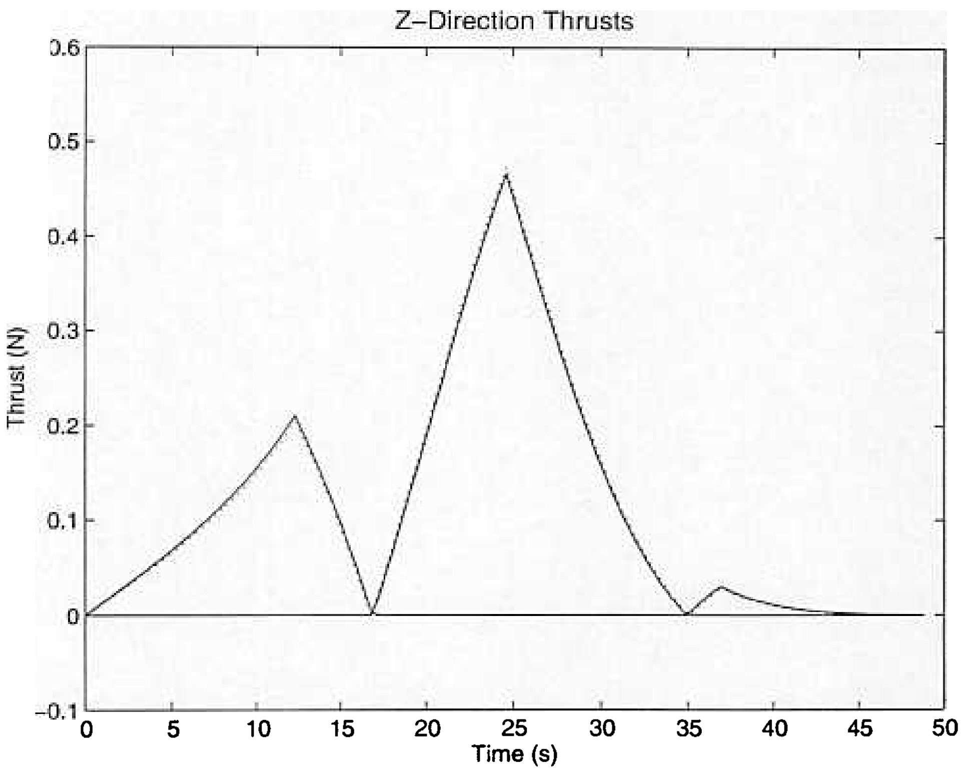

Figure 13: Trajectories for z-direction thrusters.

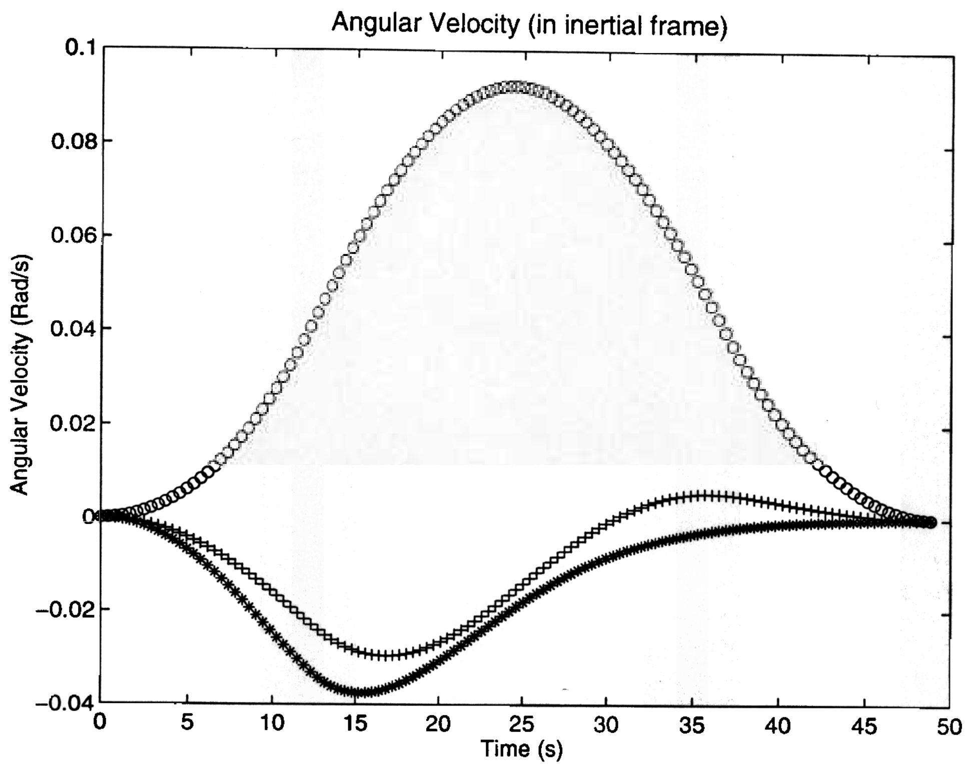

Figure 14: Sprint angular velocity component trajectories.

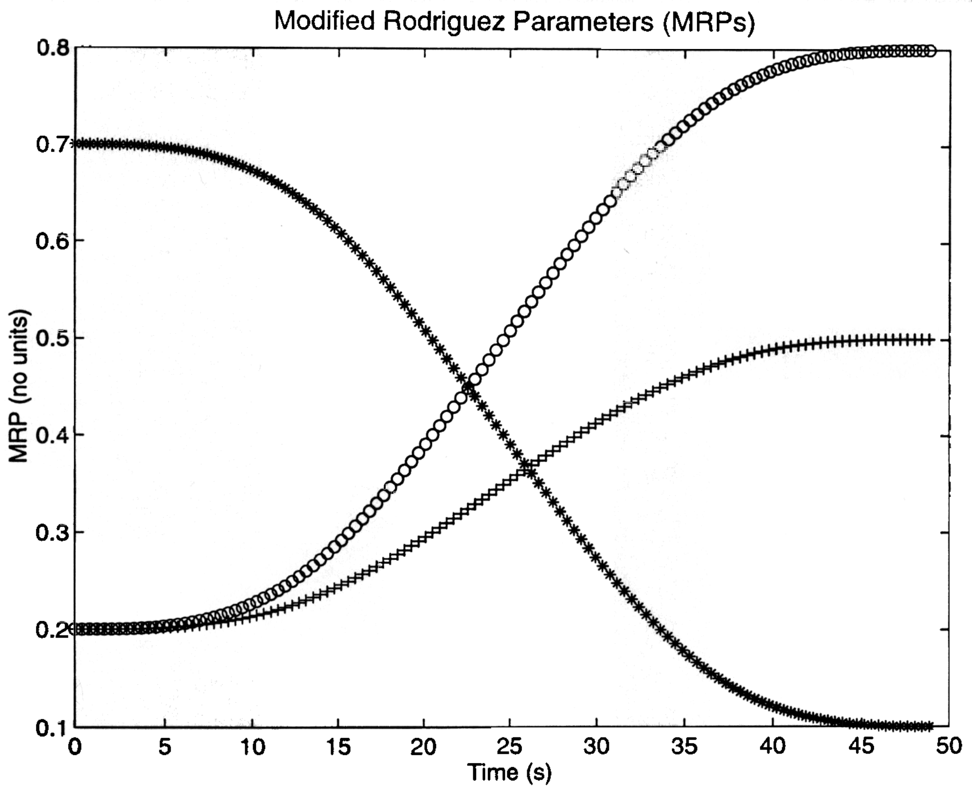

Figure 15: Modified Rodriguez parameter trajectories.

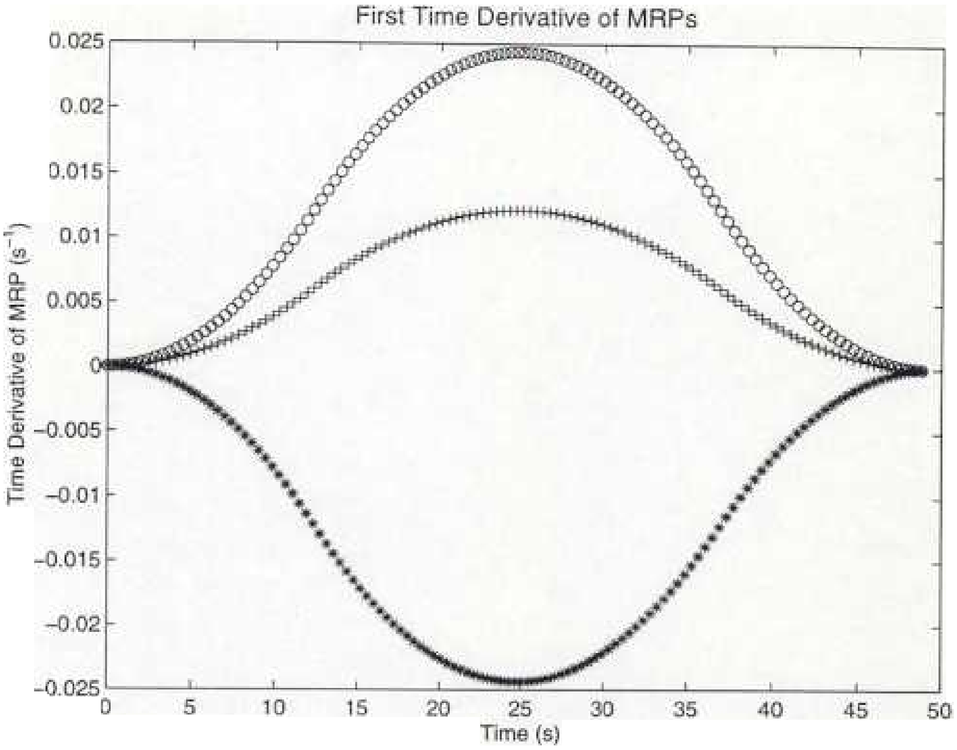

Figure 16: Trajectories of the first time derivatives of modified Rodriguez parameters.

The most useful enhancement of the planner would be to create a postprocessor that would take the continuous thrust trajectories generated by our planner and produce pulse-width-modulated thrust histories for execution on Sprint. Such an algorithm would be straight forward to implement.

References

[1] P. Tsiotras, J.L. Junkins, and H. Schaub. Higher-order cayley transforms with applications to attitude representations. *Journal of Guidance, Control and Dynamics*, 20(3):528–535, May 1997.

[2] T. Williams and S. Tanygin. On-orbit engineering tests of aercam sprint robotic camera vehicle. In *AAS/AIAA Space Flight Mechanics Meeting*, February 1998.