Characterization of Numerical Error in the Simulation of Translunar Trajectories Using the Method of Nearby Problems

by

Ashish Ashok Jagat

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 9, 2011

Approved by

Andrew J. Sinclair, Chair, Associate Professor of Aerospace Engineering
David Cicci, Professor of Aerospace Engineering
Andrew Shelton, Assistant Professor of Aerospace Engineering

Abstract

This thesis focuses on analyzing the effect numerical error has on the accurate simulation of translunar trajectories. The method of nearby problems is employed to estimate the numerical error. A simulation is developed to generate translunar trajectories. Analytical curve fit is generated to this numerical solution and this curve fit is used to compute analytical source terms. The addition of these source terms to the governing equations defines a nearby problem, for which the curve fit serves as an exact solution. By solving the nearby problem numerically, the numerical error in it can be calculated. This facilitates the estimation of numerical error in the original problem.

Acknowledgments

This work could not have been completed without the help and support of many individuals. I am especially grateful to my advisor, Dr. Andrew Sinclair, for his constant encouragement, guidance, and support. He went above and beyond to help me succeed. His suggestions from time to time have been imperative to this work. Working with him has been a tremendous learning experience for me. I am very thankful for the insights and valuable time of my committee members Dr. David Cicci and Dr. Andrew Shelton. I would also like to thank Mr. Dave Patrick of the Department of Physics at Auburn University for giving me the opportunity to work as a teaching assistant. I can't imagine life without friends and have no words for their tremendous support both during good and tough times. Finally, the support of my parents, my sister Anagha, and the rest of my family in whatever I do is invaluable to me.

Table of Contents

Appendices

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Numerical simulations play an important role in mathematical modeling of many systems in engineering. They are of utmost importance when it comes to estimating the performance of systems too complex for analytical solutions. Numerical simulations are imperative to the field of orbital mechanics as there are many differential equations and dynamical systems which cannot be solved analytically. Numerical simulations in this field have been used since the era of the Apollo missions.

In numerical simulations, however, one must account for the numerical error as it affects the efficacy of the simulation scheme. Previous studies on the numerical error in orbital mechanics simulations have included three types of numerical error - iteration error, round-off error, and discretization error. Iteration error is the difference between the current iterative solution and the exact solution. In Runge-Kutta simulation of translunar trajectories, however, iterative solutions are not used, and hence iteration error is not considered in this study. Round-off error is caused by the fact that digital computers can store numbers with only a finite precision. Discretization error is the difference between the solution of the discretized equation and the exact solution of the original differential equation.

Many researchers have studied numerical error in orbital mechanics simulations. Some of them have focused on the development of accuracy assessment techniques.

Huang and Innanen [1] showed that the traditional ways of checking the accuracy of the numerical solutions to the dynamical systems by the use of known integrals or the integral invariant relations are neither exact nor reliable due to the tendency of these numerical solutions to keep the integrals constant. They suggested a revised technique to use the integral invariant relations for checking the accuracy of the numerical solutions to the dynamical systems.

Other researchers have focused on surveying the accuracy of various numerical integration schemes, often assessing the accuracy by comparison of the numerical solutions to known exact solutions. Fox [2] did an accuracy based comparative study of various categories of numerical integration methods applied to the solution of two-body problem. Berry and Healy [3] compared the efficacy of various accuracy assessment techniques of numerical integrators using two-body problem with and without perturbations. Montenbruck [4] assessed the usefulness of various methods of numerical integration such as Runge-Kutta, multi-step and extrapolation based methods for generating numerical solutions to the problems involving solar system bodies or artificial satellites. Hadjifotinou and Gousidou-Koutita [5] proposed a new method called the recurrent power series (RPS) method for the integration of the system of n satellites orbiting a point-mass planet.

One approach taken to check the numerical accuracy of a numerical solution is to construct a similar problem to the original problem of interest. In this approach, the similar problem has an exact solution. Therefore, error in the numerical solution to the similar problem is exactly known which is then used to estimate the error in the numerical solution to the original problem of interest. Researchers studying orbital

mechanics as well as fluid dynamics have taken this approach. Roach [6] proposed "method of manufactured solutions (MMS)". MMS involves manufacturing an exact solution to a set of equations which are a modified form of the original differential equations. The solution obtained to this set of modified equations may not have physical significance. Therefore, MMS is used only to verify the mathematics involved in solving the original equations, and does not verify the solution obtained by solving the original equations.

Other researchers have used their similar problem to actually validate the numerical solution to the original problem. All these researchers use curve fit to the numerical solution of the original problem to construct the similar problem. The way in which these curve fits are calculated and are then used to construct the similar problem might differ slightly in each case. Zadunaisky [7, 8, 9, 10] suggested a technique in which he called his similar problem as a "pseudo-system" and applied it to the problems in orbital mechanics. Junkins and Lee [11] constructed "benchmark problem" for hybrid coordinate systems of ordinary/partial differential equations. Hopkins and Roy [12, 13] referred to their similar problem as "nearby problem" and the approach was called as "the method of nearby problems (MNP)". They applied this method to the problems in fluid dynamics.

MNP is based on constructing a problem near the original problem of interest. This nearby problem is constructed in such a way that it is both representative of the original problem and also has an exact known solution. This nearby problem is then solved numerically using the same numerical solution scheme that was used to numerically solve the original problem. Because the exact solution to the nearby problem

is known, error in its numerical solution can be calculated. This information is then used to estimate the error in the numerical solution of the original problem. MNP involves five steps. These steps are explained as follows:

Establishing an Accurate Numerical Solution to the Original Problem

Once the problem of interest is identified, the first step is to discretize this problem and produce an accurate numerical solution.

Generating an Analytical Curve Fit to the Above Numerical Solution

Once the accurate numerical solution is computed in previous step, this step involves generating an analytical curve fit to this numerical solution. One of the many curve fitting techniques is used to generate this curve fit. It should be kept in mind that the technique used for curve fitting should provide a particular order of continuity which is problem dependent. Once the curve fit is generated, it should be examined to see how good the fit approximates the numerical solution. This analytical curve fit will serve as the exact solution to the nearby problem.

Generating Analytical Source Terms

The nearby problem differs from the original problem by (hopefully) small source terms. These source terms are obtained by operating the original equation on the analytic curve fit obtained from the previous step. In the limit, as the magnitude of the source terms approaches zero, the nearby problem approaches the original problem. The nearness of the nearby problem to the original problem can be judged by the magnitude of the source terms.

Numerically Solving the Nearby Problem

The nearby problem consists of original equations plus the analytical source terms. This step involves solving the nearby problem numerically using the same numerical solution scheme that was used to solve the original problem.

Estimating the Numerical Error in the Original Problem

Because both the exact and the numerical solution to the nearby problem are known, error in the numerical solution can be calculated for the nearby problem. This information can then be used to estimate the error in the numerical solution to the original problem.

In this thesis, an effort is made to extend the application of MNP to the problems in orbital mechanics. The objective is to demonstrate the usefulness of MNP in validating the accuracy of the numerical solutions to the problems in orbital mechanics by constructing a nearby problem to the Earth-spacecraft-Moon three-body problem. While this work also uses the curve fit to the numerical solution of the original problem to construct the nearby problem, unlike Zadunaisky, various curve fitting techniques are examined first and then the technique satisfying certain criteria is used to construct the nearby problem. Moreover, the way in which this curve fit is used to construct the nearby problem differs from that of Zadunaisky's.

Chapter 2 discusses the original problem studied in this work. In this chapter, equations of motion of the *n*-body problem are derived first and these equations are then specialized for the case of the three-body problem. System model for the Earth-spacecraft-Moon three-body problem is also discussed and the equations of motion governing the motion of the spacecraft in the cis-lunar space are formed. Chapter 3

discusses the numerical solution to the Earth-spacecraft-Moon three-body problem. Chapter 4 discusses various curve fitting techniques and their feasibility to construct the nearby problem. Chapter 5 discusses the calculation of analytical source terms using the various curve fitting techniques studied in chapter 4. In chapter 6, the nearby problem is constructed and the nearness of this nearby problem to the original problem is established. This allows the exact error in the numerical solution to the nearby problem to be considered as a good estimation of the error in the numerical solution to the original problem. The numerical error estimated by MNP is compared to that estimated by Richardson extrapolation using both global and local order of accuracy. In chapter 7, a conclusion to this thesis is presented.

CHAPTER 2

ORIGINAL PROBLEM

The original problem studied in this work is the Earth-spacecraft-Moon three-body problem. One of the most fundamental problems of orbital mechanics is to accurately describe the motion of *n* gravitationally interacting massive particles also known as the *n*-body problem. In this chapter, we will begin with the general *n*-body problem and then we will specialize to our three-body problem. Newton introduced his three laws of motion in *The Mathematical Principles of General Philosophy*, or, more simply, *the Principia*, in 1687. Newton's law of universal gravitation along with the second law of motion can be used to describe the *n*-body problem. The second law can be expressed mathematically in vector notation as follows:

$$\Sigma \boldsymbol{F} = m\,\ddot{\boldsymbol{r}} \tag{1}$$

In Eq. (1), $\Sigma \boldsymbol{F}$ is the vector sum of all forces acting on the mass m and $\ddot{\boldsymbol{r}}$ is the vector acceleration of the mass measured relative to an inertial reference frame. Similarly, the universal law of gravitation can be expressed mathematically in vector notation as follows:

$$\mathbf{F}_{12} = -\frac{G\,m_1\,m_2}{{r_{12}}^2}\,\frac{\boldsymbol{r}_{12}}{r_{12}} \tag{2}$$

In Eq. (2), $\mathbf{F}_{12}$ is the gravitational force exerted on mass m_2 by mass m_1, $\boldsymbol{r}_{12}$ is the vector from m_1 to m_2 and G is the universal gravitational constant.

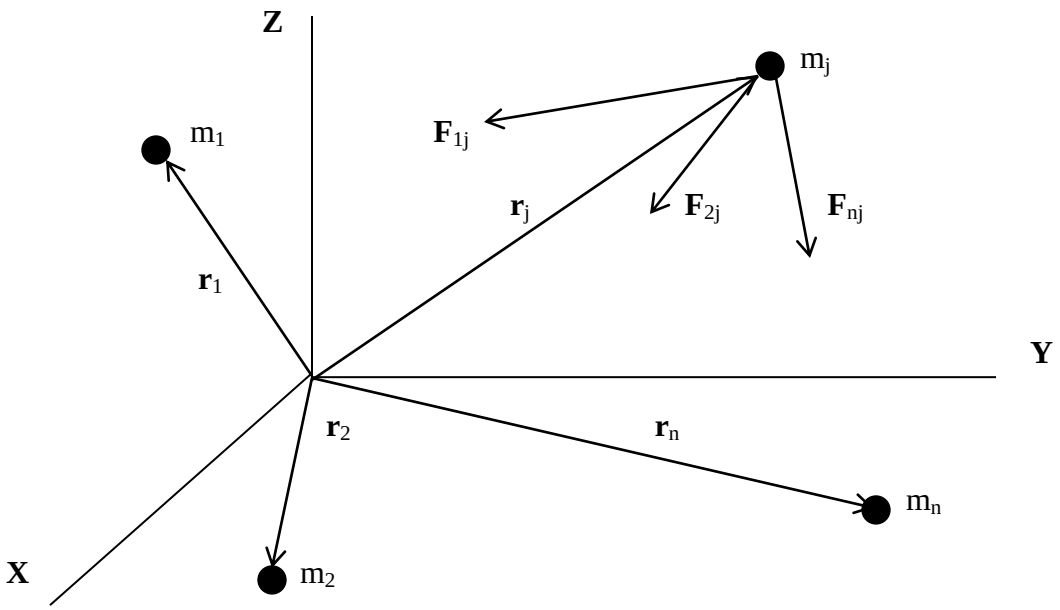

Figure 1. *n*-body Problem [14]

The *n*-body problem is illustrated in figure 1. It shows a system of *n* bodies $m_1, m_2, m_3, \ldots, m_n$. Let us assume an inertial reference frame (*X*, *Y*, *Z*) in which the position vectors of these *n* masses are $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \ldots, \boldsymbol{r}_n$ respectively. We wish to study the motion of one of these bodies. Let us call this body as the *j*th body, m_j. At any given time in its journey, this body is being acted upon by several gravitational masses and may be experiencing other forces such as drag, thrust, and solar radiation pressure. Let us not consider all these other forces for the time being. Applying Newton's law of universal gravitation to the above system, the force $\mathbf{F}_{ij}$ exerted on m_j by m_i is:

$$\mathbf{F}_{ij} = -\frac{G\, m_i\, m_j}{{r_{ij}}^3}\, \boldsymbol{r}_{ij} \tag{3}$$

In Eq. (3), $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$.

The vector sum, **F**, of all such gravitational forces acting on the *j*th body may be written as:

$$\mathbf{F} = -G\, m_j \sum_{i=1}^{n,i\neq j} \left(\frac{m_i}{{r_{ij}}^3}\, \boldsymbol{r}_{ij} \right) \tag{4}$$

Applying Newton's second law of motion,

$$\mathbf{F} = m_j \ddot{\boldsymbol{r}}_j \tag{5}$$

Dividing both sides of Eq. (5) by m_j we get,

$$\ddot{\boldsymbol{r}}_j = -G \sum_{i=1}^{n,i\neq j} \left(\frac{m_i}{{r_{ij}}^3}\, \boldsymbol{r}_{ij} \right) \tag{6}$$

Eq. (6) describes the inertial motion of the *j*th body. For our purposes, we will be interested in describing the motion of a body relative to another body. Consider figure 2.

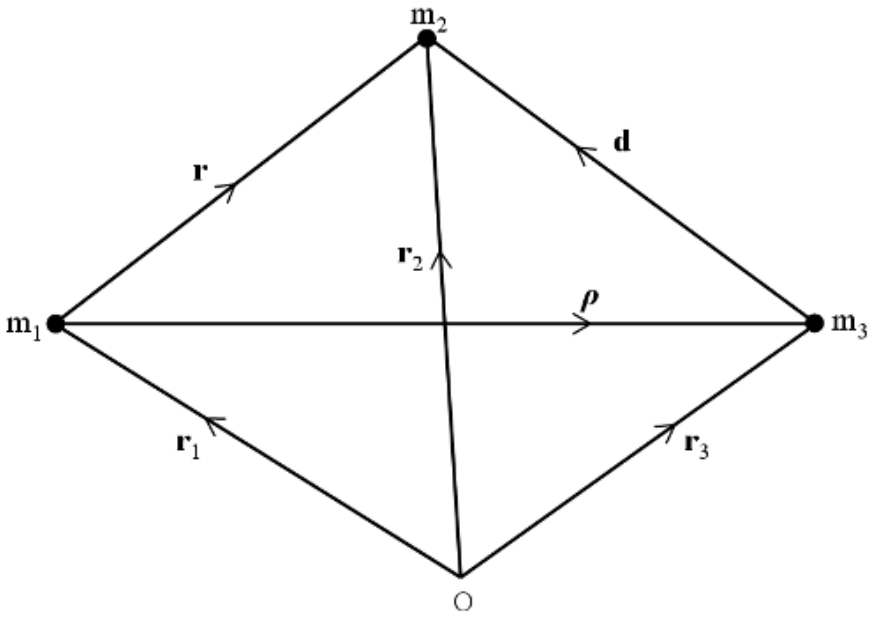

Figure 2. Three-body Problem [16]

Equation of motion of m_1 with respect to the inertial reference frame can be given by:

$$\ddot{\boldsymbol{r}}_1 = -G\left(\frac{m_2}{{r_{21}}^3}\,\boldsymbol{r}_{21}\right) - G\left(\frac{m_3}{{r_{31}}^3}\,\boldsymbol{r}_{31}\right) \tag{7}$$

Also the equation of motion of m_2 with respect to the same reference frame is:

$$\ddot{\boldsymbol{r}}_2 = -G\left(\frac{m_1}{{r_{12}}^3}\,\boldsymbol{r}_{12}\right) - G\left(\frac{m_3}{{r_{32}}^3}\,\boldsymbol{r}_{32}\right) \tag{8}$$

By subtracting Eq. (7) from (8), equation of motion of m_2 relative to m_1 is given as:

$$\ddot{\boldsymbol{r}}_2 - \ddot{\boldsymbol{r}}_1 = -G\left(\frac{m_1}{{r_{12}}^3}\,\boldsymbol{r}_{12} + \frac{m_2}{{r_{12}}^3}\,\boldsymbol{r}_{12} + \frac{m_3}{{r_{32}}^3}\,\boldsymbol{r}_{32} + \frac{m_3}{{r_{13}}^3}\,\boldsymbol{r}_{13}\right) \tag{9}$$

Let $\boldsymbol{r} = \boldsymbol{r}_2 - \boldsymbol{r}_1,\ \boldsymbol{d} = \boldsymbol{r}_2 - \boldsymbol{r}_3,\ \boldsymbol{\rho} = \boldsymbol{r}_3 - \boldsymbol{r}_1$ as shown in figure 2.

Thus, Eq. (9) can be simplified as:

$$\ddot{\boldsymbol{r}} = -G\left(\frac{(m_1 + m_2)}{r^3}\,\boldsymbol{r}\right) - Gm_3\left(\frac{\boldsymbol{d}}{d^3} + \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{10}$$

Similarly, equation of motion of m_2 relative to m_3 is given as:

$$\ddot{\boldsymbol{d}} = -G\left(\frac{(m_3 + m_2)}{d^3}\,\boldsymbol{d}\right) - Gm_1\left(\frac{\boldsymbol{r}}{r^3} - \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{11}$$

Considering $m_2 \ll m_1$ and $m_2 \ll m_3$ above equations can be rewritten as:

$$\ddot{\boldsymbol{r}} = -\frac{G\,m_1}{r^3}\,\boldsymbol{r} - Gm_3\left(\frac{\boldsymbol{d}}{d^3} + \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{12}$$

$$\ddot{\boldsymbol{d}} = -\frac{G\,m_3}{d^3}\,\boldsymbol{d} - Gm_1\left(\frac{\boldsymbol{r}}{r^3} - \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{13}$$

In both the above equations, the first term on the right hand side is the central acceleration due to the primary body, while the second term on the right hand side is the

disturbing acceleration from the perturbing body. As mentioned earlier, several external forces such as drag, thrust, and solar radiation pressure are not considered while deriving these equations of motion. Also, Newton's law of gravitation applies only if the bodies are spherical and the mass is evenly distributed in spherical shells. Thus, the non-spherical shape of the bodies also results in a perturbing force. Therefore, the governing equations derived above are approximations and inclusion of all the perturbing forces could facilitate a more accurate modeling of the trajectory.

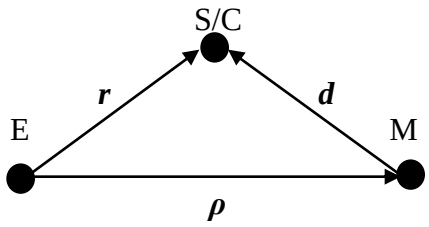

Figure 3. System Model

For the Earth-spacecraft-Moon three-body problem, let m_1 be the Earth, m_2 be the spacecraft and m_3 be the moon. Figure 3 shows the system model for this problem. The spacecraft motion is described either in the Earth-Centered (EC) reference frame or in the Moon-Centered (MC) reference frame. Both EC and MC are non-rotating reference frames. The EC frame has its origin at the center of mass of the Earth while the MC frame has its origin at the center of mass of the Moon. The position vector of the spacecraft in the EC frame is given by $\boldsymbol{r}$ while in the MC frame it is given by $\boldsymbol{d}$. The equation of motion of the spacecraft in vector form in the EC frame is given by Eq. (12) rewritten below. Here, μ_e is the gravitational parameter for the Earth with a value of $3.98600436 \times 10^{14}$ m^3/s^2 and μ_m is the gravitational parameter for the Moon with a value of 4.90266×10^{12} m^3/s^2.

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = -\frac{\mu_e}{r^3}\mathbf{r} - \mu_m\left(\frac{\mathbf{d}}{d^3} + \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{14}$$

In this equation, the Earth is the central body and the Moon causes the perturbative acceleration. The Cartesian components of ***r*** in a non-rotating frame are written as *X* and *Y*, and the scalar components of the equation of motion are as follows.

$$\frac{\mathrm{d}^2 X}{\mathrm{d}t^2} = -\frac{\mu_e}{r^3} X \;\; - \mu_m\left(\frac{X_m}{d^3} + \frac{\rho_x}{\rho^3}\right) \tag{15}$$

$$\frac{\mathrm{d}^2 Y}{\mathrm{d}t^2} = -\frac{\mu_e}{r^3} Y - \mu_m\left(\frac{Y_m}{d^3} + \frac{\rho_y}{\rho^3}\right) \tag{16}$$

Similarly, the equation of motion of the spacecraft in vector form in the MC frame is given by Eq. (13) rewritten below.

$$\frac{\mathrm{d}^2\mathbf{d}}{\mathrm{d}t^2} = -\frac{\mu_m}{d^3}\mathbf{d} - \mu_e\left(\frac{\mathbf{r}}{r^3} - \frac{\boldsymbol{\rho}}{\rho^3}\right) \tag{17}$$

In this equation, the Moon is the central body and the Earth causes the perturbative acceleration. The Cartesian components of ***d*** in a non-rotating frame are written as X_m and Y_m, and the scalar components of the equation of motion are as follows.

$$\frac{\mathrm{d}^2 X_m}{\mathrm{d}t^2} = -\frac{\mu_m}{d^3} X_m - \mu_e\left(\frac{X}{r^3} + \frac{\rho_x}{\rho^3}\right) \tag{18}$$

$$\frac{\mathrm{d}^2 Y_m}{\mathrm{d}t^2} = -\frac{\mu_m}{d^3} Y_m - \mu_e\left(\frac{Y}{r^3} + \frac{\rho_y}{\rho^3}\right) \tag{19}$$

These two sets of equations in the EC and the MC reference frames represent analytically equivalent description of the three-body problem. The numerical solution described in the next chapter will use switching between the two reference frames.

The problem description is completed by defining various initial conditions. The starting location of the Moon is out from the Earth along the positive *X*-axis as shown in figure 3. Because the mean eccentricity of the Moon's orbit is only about 0.0549, it is considered to be a circular orbit with a radius (ρ) of 384,400 km. The spacecraft trajectory is assumed to be coplanar with the Moon's orbit. The initial conditions for the translunar trajectory are specified by the injection criteria relative to the low Earth orbit (LEO): altitude (*alt*), Δv, and angle (θ). These initial conditions are given in table 1 and are illustrated in figure 4. [16]

Parameter	Value(Units)
r_e	6372.797(km)
alt	359750(m)
Δv	3102.13(m/s)
θ	-36.890(degrees)

Table 1. LEO Conditions

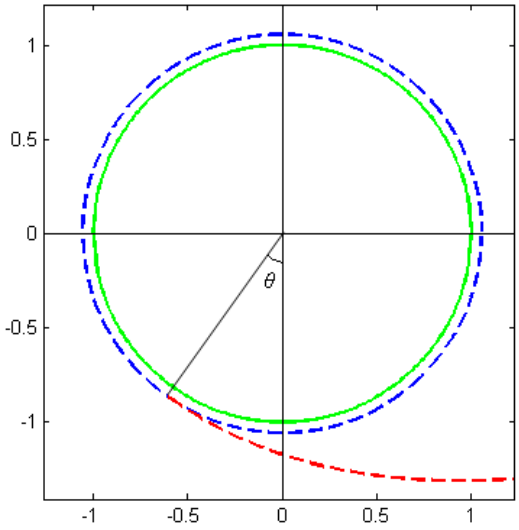

Figure 4. The Earth, LEO, Point of Δv and Initial Trajectory

CHAPTER 3

NUMERICAL SOLUTION TO THE ORIGINAL PROBLEM

To numerically solve the Earth-spacecraft-Moon three-body problem, a MATLAB code (appendix A) implementing the fourth-order Runge-Kutta (RK4) numerical integration scheme is developed to propagate the initial conditions forward in time. [16] A trip time of 3.5 days is considered. An integration time step of 20 seconds is used. Consistent with the RK4 algorithm, the lunar ephemeris are updated every half time step. Choice of the reference frame for describing the spacecraft motion affects the numerical error in the simulation. In order to avoid precision loss due to round-off error, reference frame is switched from the EC to the MC at 2.905 days. [16] After the switch point, equation of motion in the MC frame is integrated for $\boldsymbol{d}$. This solution is then converted to output a solution for $\boldsymbol{r}$ at each instant in time.

Figure 5 illustrates the resulting numerical solution of the original problem for the spacecraft's trajectory along with the Moon's trajectory. Figures 6 and 7 show the plots of X and Y with respect to time. Figures 8 and 9 show the plots of $\dot{X}$ and $\dot{Y}$ with respect to time. Figures 10 and 11 show the plots of $\ddot{X}$ and $\ddot{Y}$ with respect to time. It can be seen from the acceleration plots that the spacecraft experiences a higher acceleration when in proximity to the Earth and the Moon. This is due to the strong nature of the gravitational forces exerted by the Earth and the Moon on the spacecraft. The higher acceleration is also reflected in the position and the velocity plots. The velocity of the spacecraft when

in proximity to the Earth and the Moon is higher, and it also changes rapidly. High curvature of the trajectory during the initial and the final phases of the journey which indicates a rapid change in position during these phases can also be seen by a close examination of figures 5, 6 and 7. This behavior will affect the accuracy of the curve fits calculated in the next chapter.

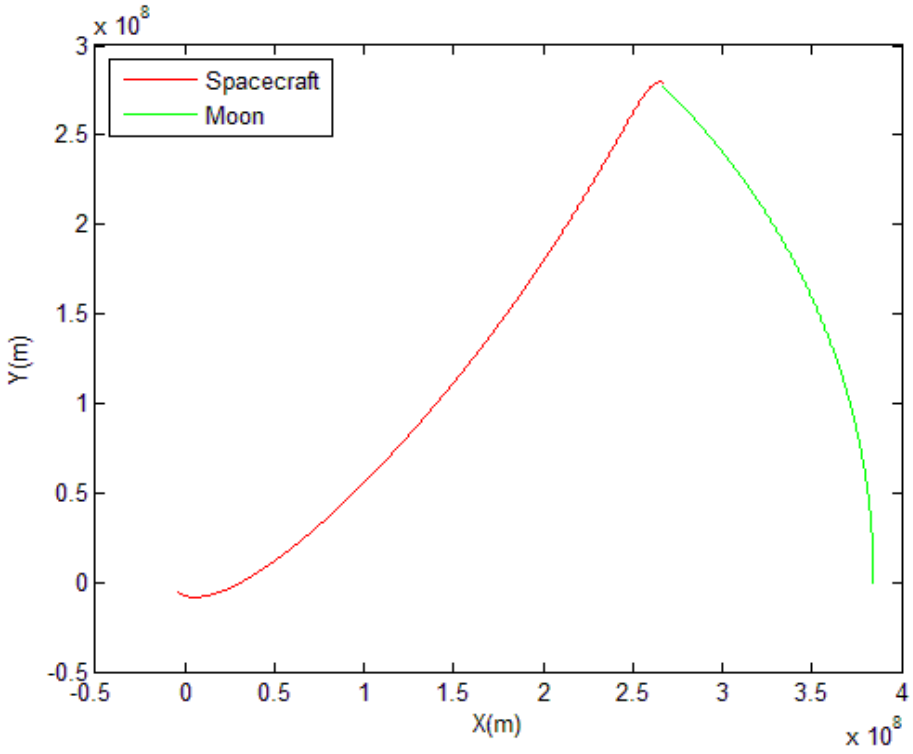

Figure 5. Spacecraft and Moon Trajectories

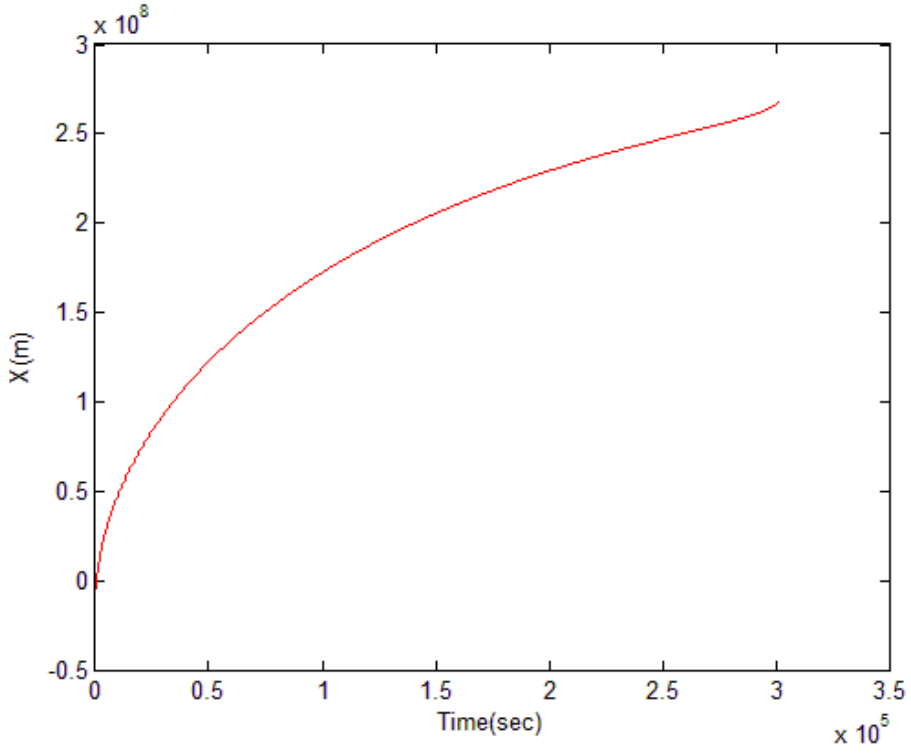

Figure 6. Time vs. *X* Position

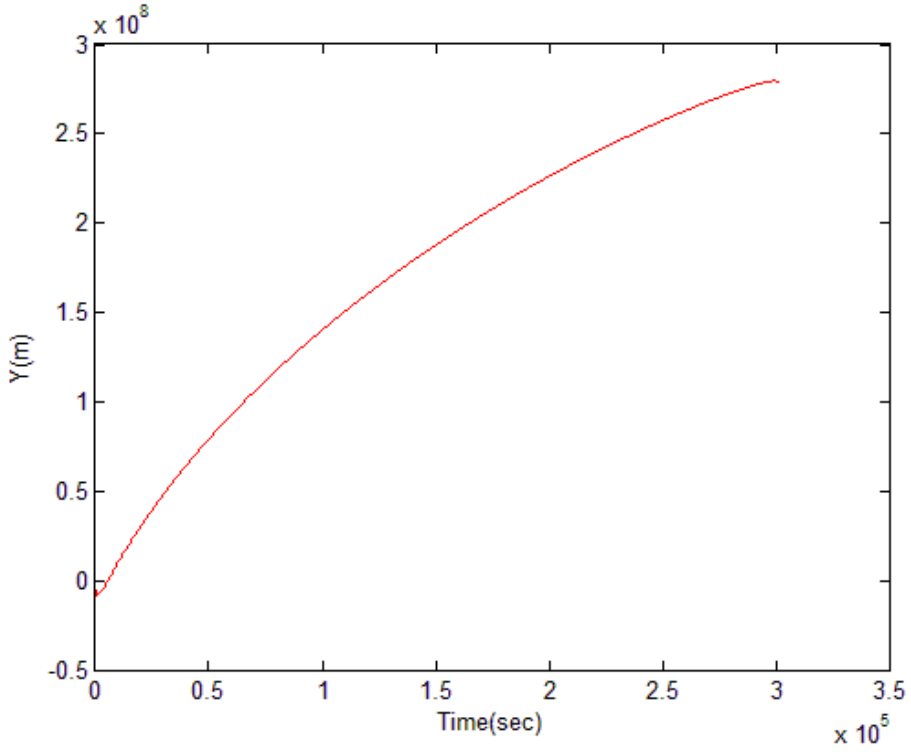

Figure 7. Time vs. *Y* Position

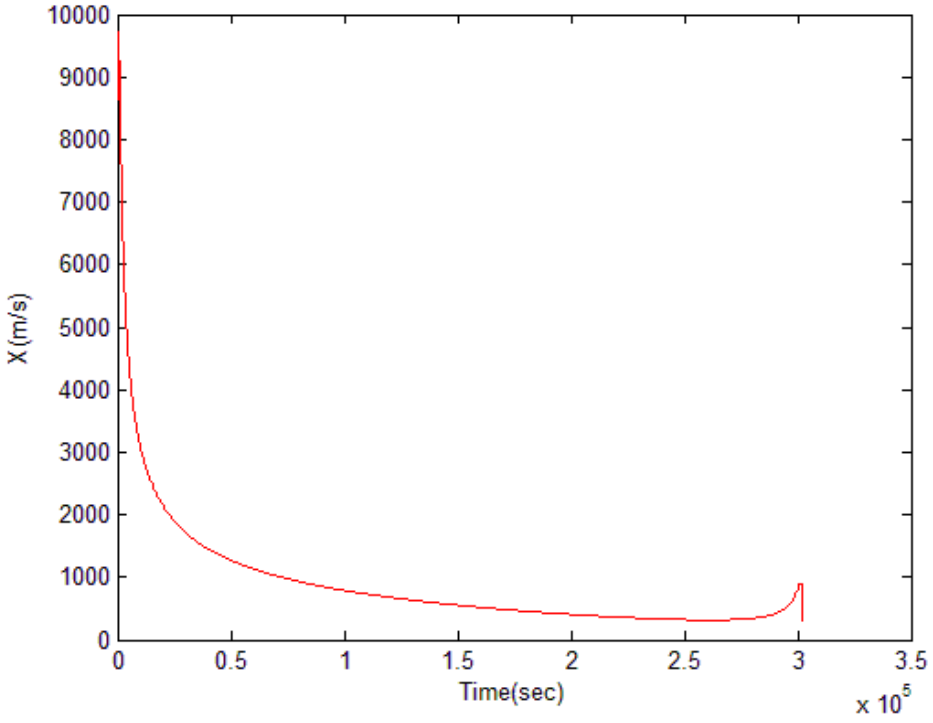

Figure 8. Time vs. *X* Velocity

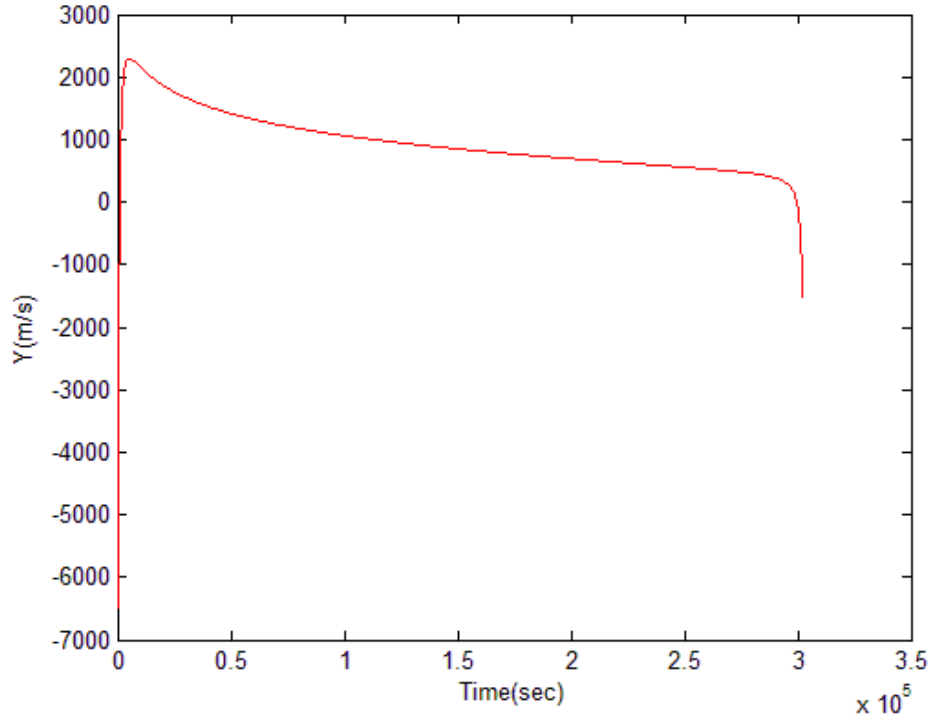

Figure 9. Time vs. Y Velocity

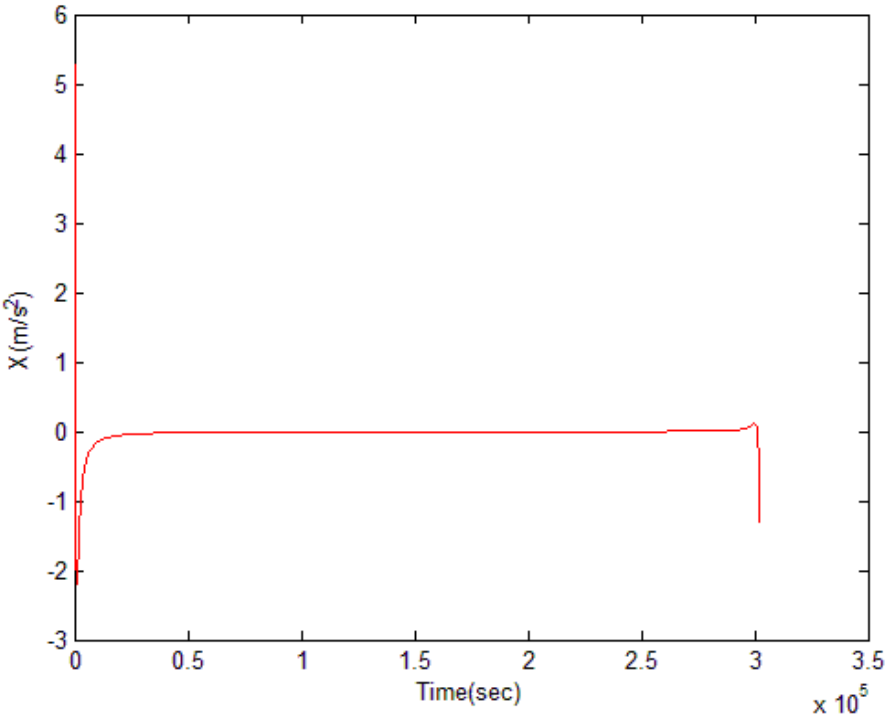

Figure 10. Time vs. *X* Acceleration

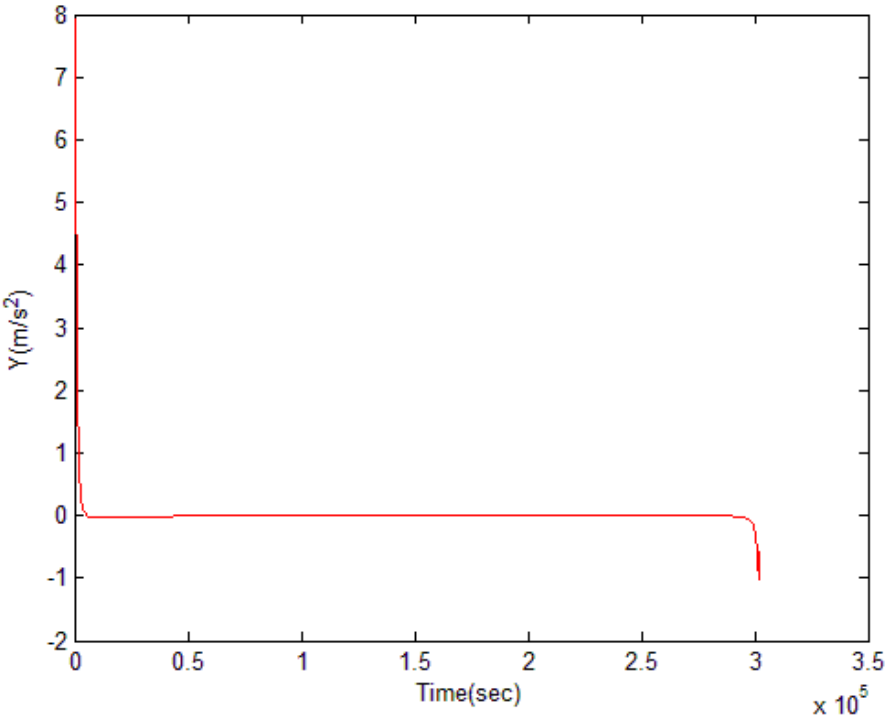

Figure 11. Time vs. *Y* Acceleration

CHAPTER 4

CURVE FITTING METHODS

Generating an accurate curve fit to the numerical solution to the original problem is a critical step in MNP as subsequent analysis is based on this curve fit. The curve-fit accuracy controls the nearness of the nearby problem. An analytical curve fit has to satisfy two criteria in order to be considered accurate. The magnitude of the source terms generated should be small, and the curve fit should maintain a particular order of continuity in order to maintain slope continuity of the source terms. The order of continuity to be maintained is problem dependent. The equations of motion of the three-body dynamics are second order differential equations. Therefore, C^3 continuity is needed in the curve fit for this problem. Various curve fitting methods are available. The methods used for generating curve fits in this study are least squares, cubic splines, and fifth-degree Hermite splines. The accuracy of these curve fits is indicated by both the residuals with respect to the numerical solution and the magnitude of the analytical source terms calculated. The governing equations contain X, Y, $\ddot{X}$, $\ddot{Y}$. However, independent curve fits are only needed for X and Y.

Least Squares

The method of least squares is a standard approach to approximate the solution of over determined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the

squares of the errors made in solving every single equation. Assume that we want to approximate a function *y(t)*, *t* being the independent variable. Assume that there are *m* observations, i.e., values of *y* measured at specific values of *t*. [17]

$$y_i = y(t_i), \ i = 1,...,m \tag{20}$$

The idea is to model *y(t)* by a linear combination of *n* basis functions:

$$y(t) = x_1\phi_1(t) + ... + x_n\phi_n(t) \tag{21}$$

For example, a second degree polynomial can be written as $x_1t^0 + x_2t^1 + x_3t^2$, i.e., as a linear combination of the basis functions t^0, t^1, and t^2. The design matrix H is a rectangular matrix of order $m \times n$ with elements

$$h_{i,j} = \phi_j(t_i) \tag{22}$$

The design matrix generally has more rows than columns. In matrix-vector notation:

$$y = Hx \tag{23}$$

H is not invertible, but a pseudo inverse can be calculated as follows:

$$H^T y = H^T Hx \tag{24}$$

$$x = (H^T H)^{-1} H^T y \tag{25}$$

In this study, the pseudo inverse is not calculated but instead the *polyfit* function in MATLAB is used. The polyfit MATLAB file forms the Vandermonde matrix, *H*, whose elements are powers of *t*:

$$H_{i,j} = t_i^{n-j} \tag{26}$$

It then uses the *backslash* operator to solve the least squares problem shown in Eq. (25). The *backslash* operator selects from a variety of algorithms depending upon the structure of the matrix H. Function *"polyfit (t, y, n)"* finds the coefficients of a polynomial *p(t)* of degree *n* that fits the data *y* best in a least squares sense. Various curve fits and their residuals with respect to the numerical solution are shown in the following figures.

The residuals of these curve fits show that as the degree of the polynomial increases, the curve fits are more accurate. However, as mentioned earlier, in MNP, in order to be considered accurate a curve fit has to satisfy one more criterion, i.e., the analytical source terms generated by these curve fits should be small in magnitude. Source terms are discussed in detail in chapter 5.

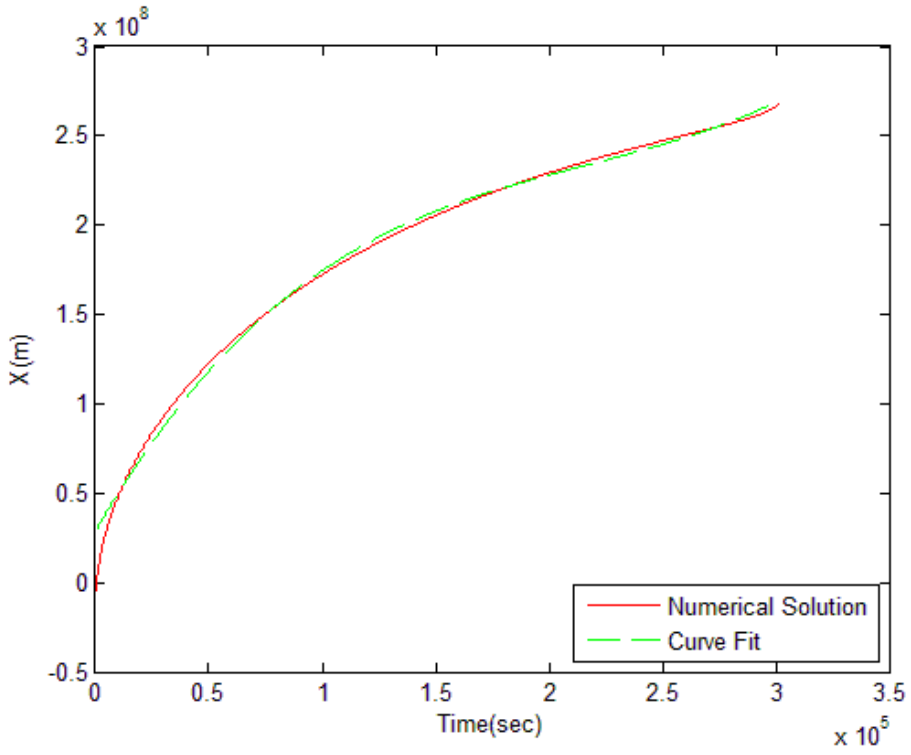

Figure 12. Curve Fit to *X* – Least Squares Using 3rd Degree Polynomial

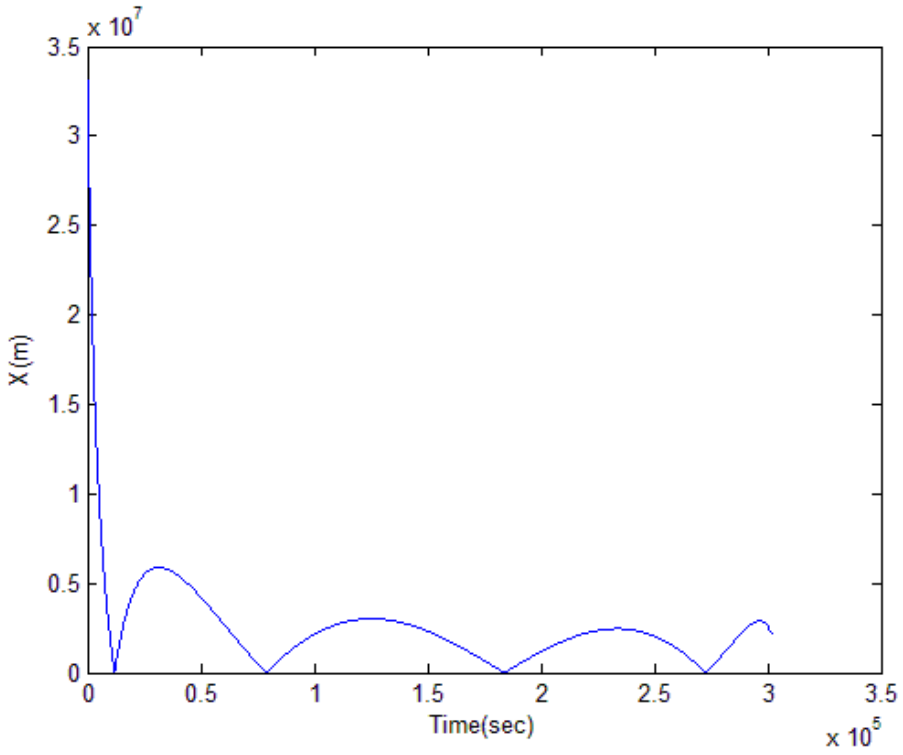

Figure 13. Curve Fit Residuals for *X* - Least Squares Using 3rd Degree Polynomial

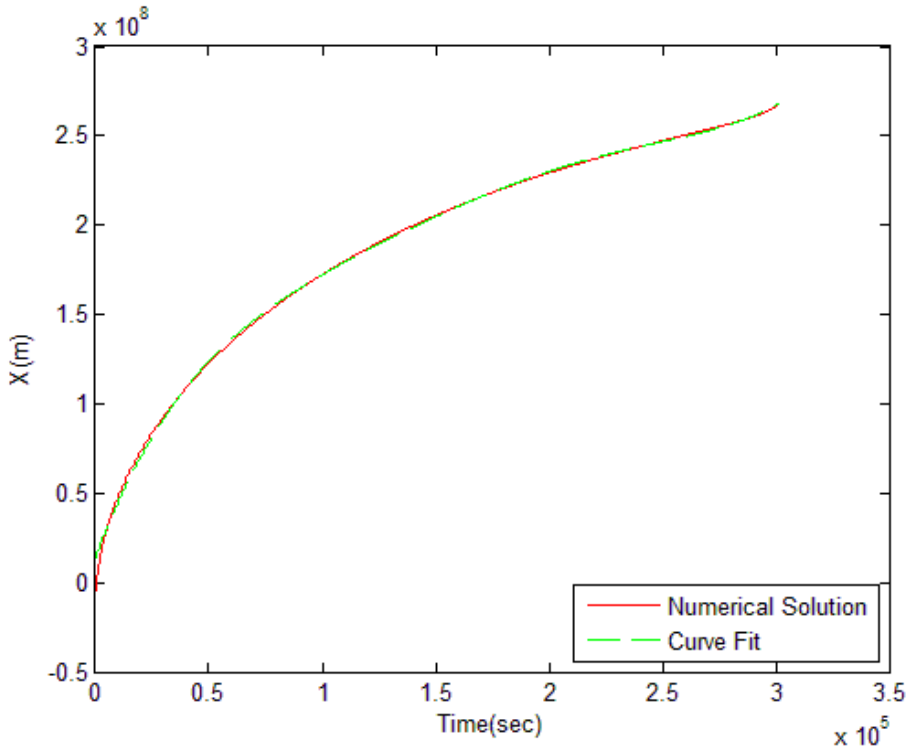

Figure 14. Curve Fit to *X* – Least Squares Using 5th Degree Polynomial

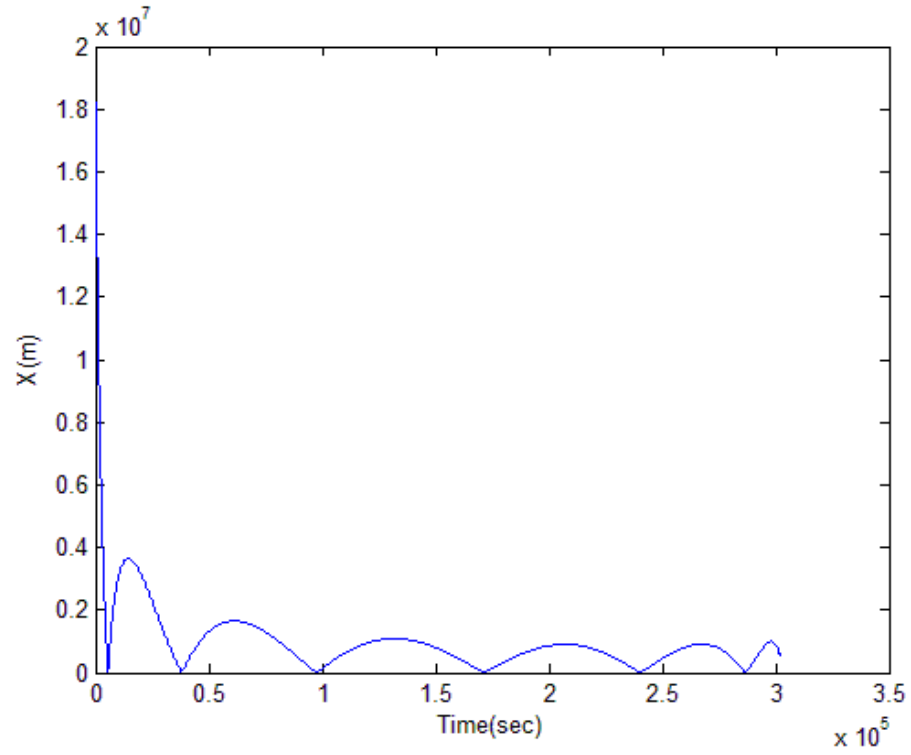

Figure 15. Curve Fit Residuals for *X* - Least Squares Using 5th Degree Polynomial

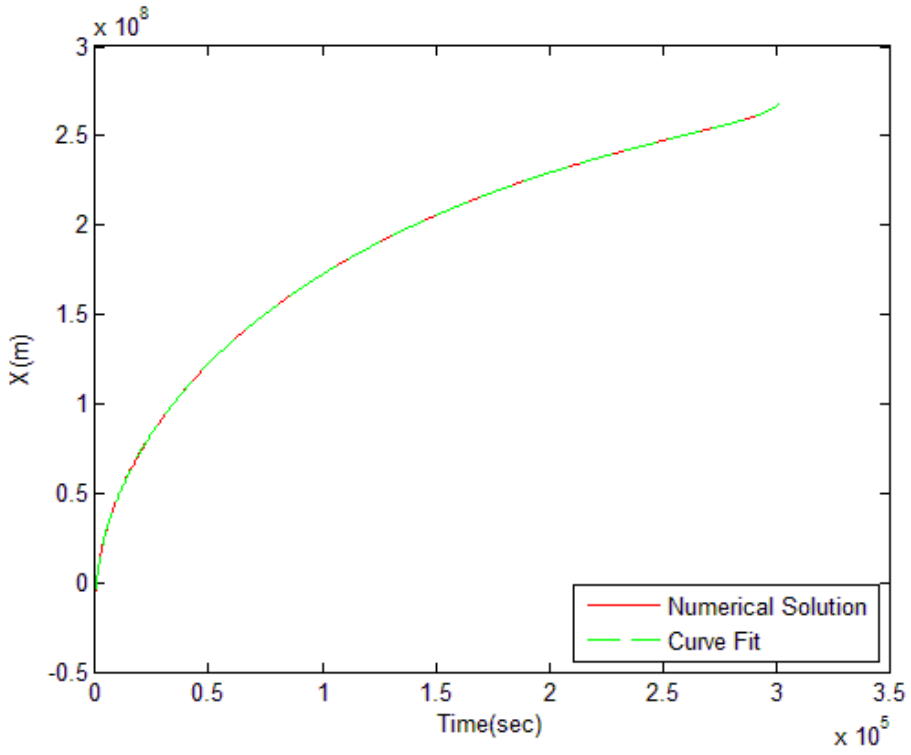

Figure 16. Curve Fit to *X* – Least Squares Using 20th Degree Polynomial

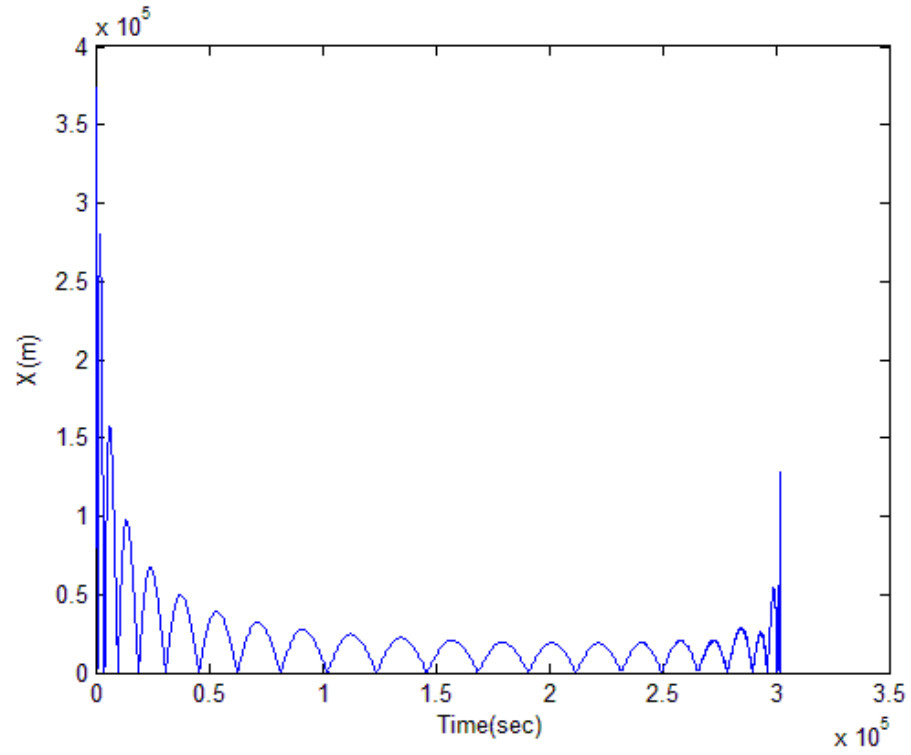

Figure 17. Curve Fit Residuals for *X* - Least Squares Using 20th Degree Polynomial

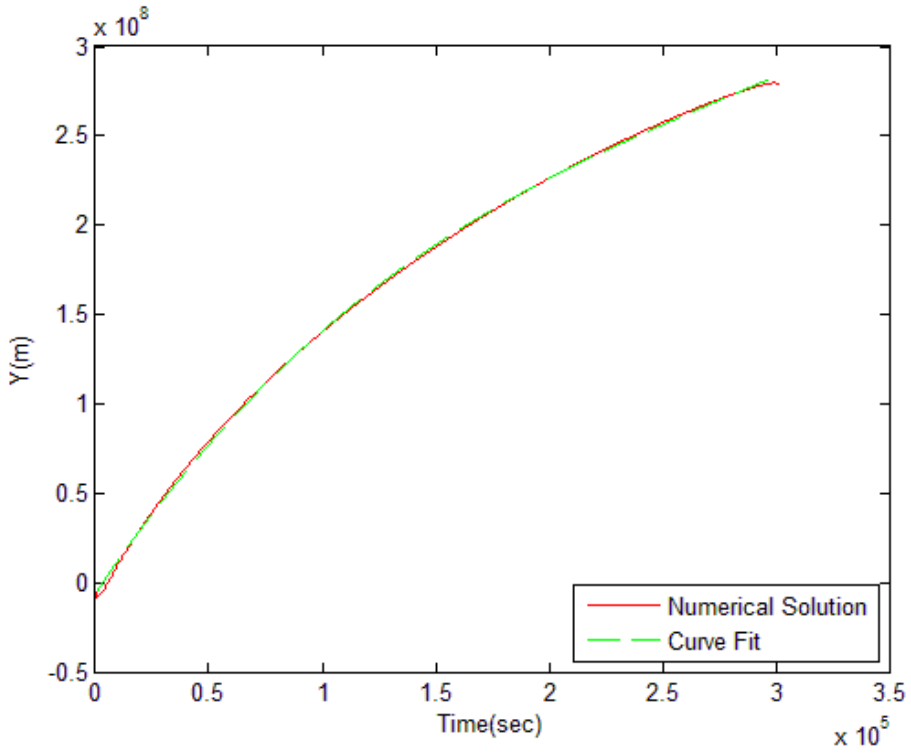

Figure 18. Curve Fit to *Y* – Least Squares Using 3rd Degree Polynomial

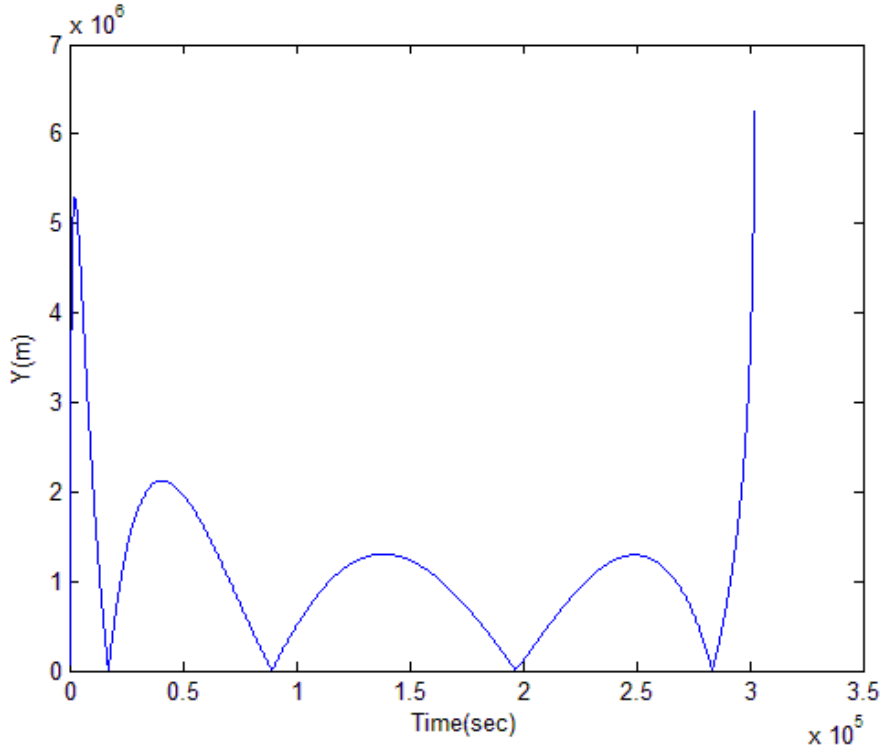

Figure 19. Curve Fit Residuals for *Y* - Least Squares Using 3rd Degree Polynomial

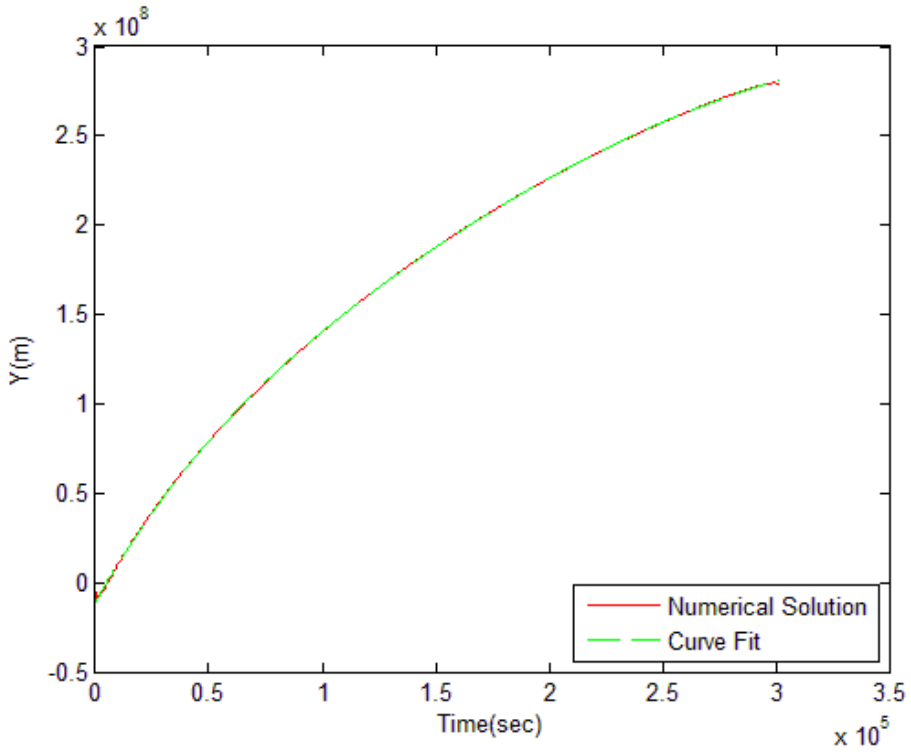

Figure 20. Curve Fit to *Y* – Least Squares Using 5th Degree Polynomial

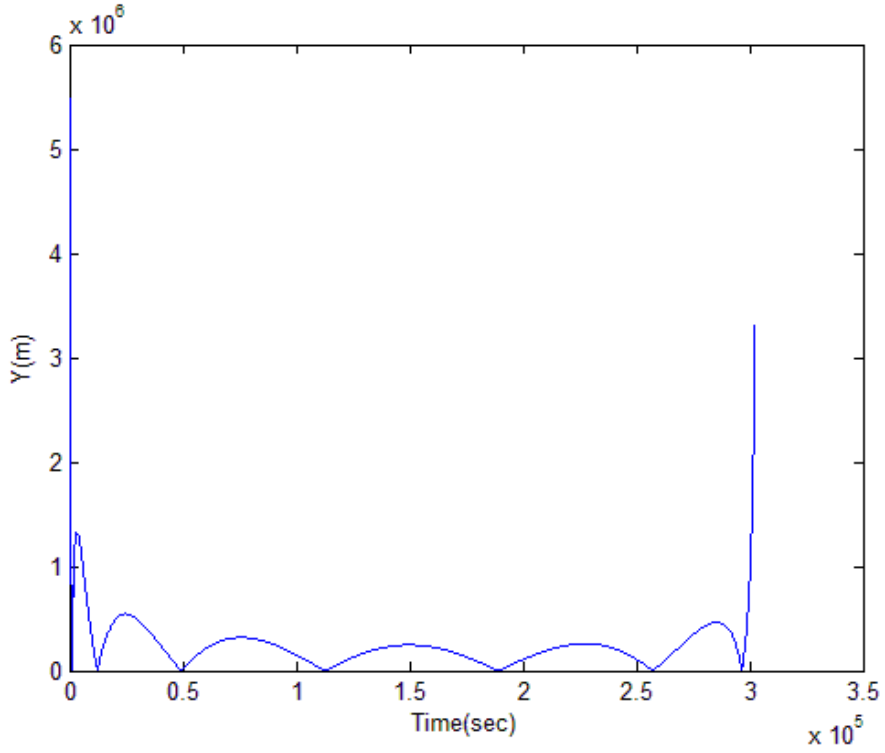

Figure 21. Curve Fit Residuals for *Y* - Least Squares Using 5th Degree Polynomial

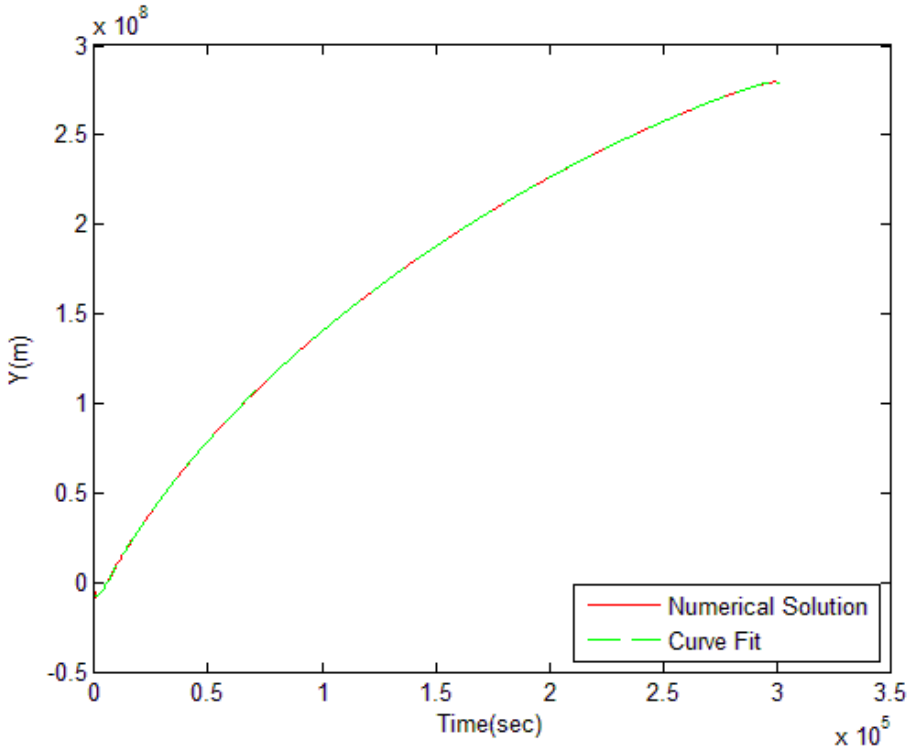

Figure 22. Curve Fit to *Y* – Least Squares Using 20th Degree Polynomial

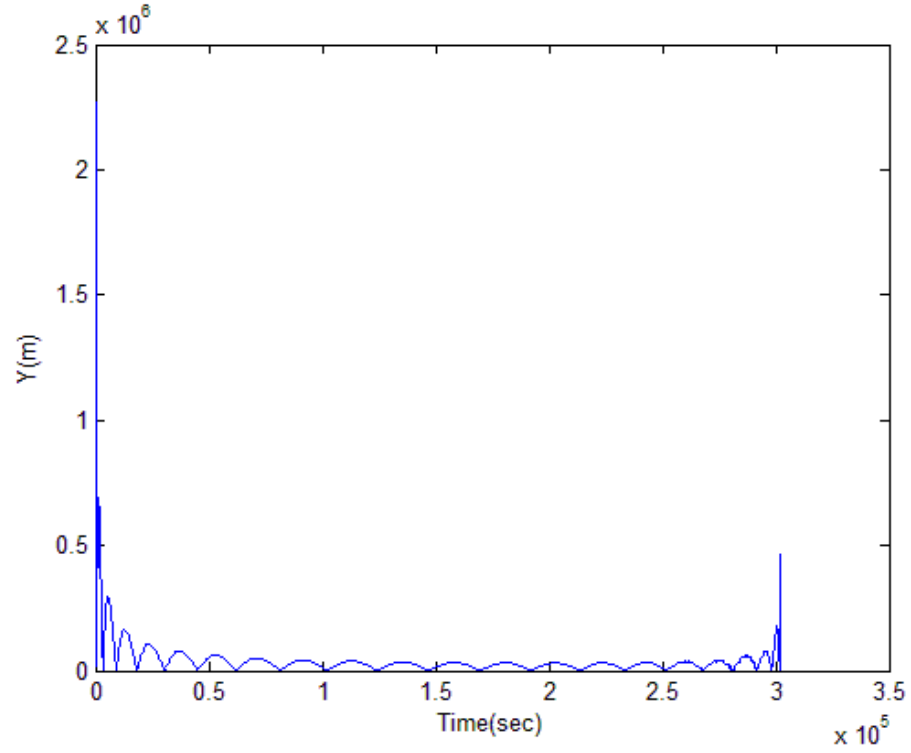

Figure 23. Curve Fit Residuals for *Y* - Least Squares Using 20th Degree Polynomial

Cubic Splines

Spline interpolation is a form of interpolation where piecewise functions called "splines" are used. This has the advantage relative to a global least squares fit that similar accuracy can be achieved using local fits that are each lower order.

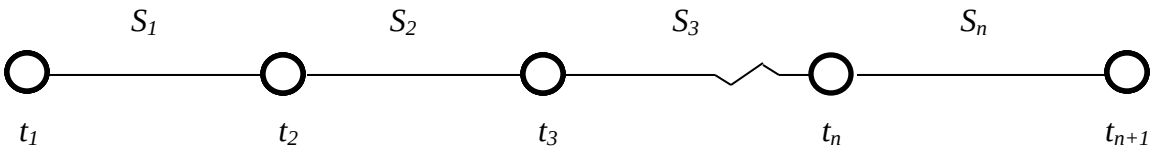

Figure 24. Schematic of Cubic Splines Interpolation

The basic idea of cubic splines is to fit a cubic polynomial on each interval between points t_i and t_{i+1} for $i = 1,\ldots, n$. [18]

$$S_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3 \tag{27}$$

This system has $n+1$ spline points (t_i for $i = 1,\ldots, n+1$) and n spline zones (S_i for $i = 1,\ldots, n$) as shown in figure 24. The conditions that are used to construct these polynomials are explained below for given X, $\dot{X}$, $\ddot{X}$ data.

$$S_i(t_i) = X_i \qquad i = 1,\ldots., n \tag{28}$$

$$S_n(t_{n+1}) = X_{n+1}$$

$$S_i(t_i) = S_{i-1}(t_i) \qquad i = 2,\ldots., n \tag{29}$$

$$\dot{S}_i(t_i) = \dot{S}_{i-1}(t_i) \qquad i = 2,\ldots., n \tag{30}$$

$$\ddot{S}_i(t_i) = \ddot{S}_{i-1}(t_i) \qquad i = 2,\ldots., n \tag{31}$$

$$\ddot{S}_1(t_1) = \ddot{X}_1 \tag{32}$$

$$\ddot{S}_n(t_{n+1}) = \ddot{X}_{n+1}$$

Conditions (28) set the value at each node point. Condition (29) makes the solution 0^{th} order continuous. Condition (30) makes the solution 1^{st} order continuous, while

condition (31) makes the solution 2nd order continuous. Conditions (32) are the two additional end point conditions.

Curve fits to *X* and *Y* using cubic splines are calculated using 8, 63, 504 and 5040 spline zones. The number 5040 corresponds to one third of the total time instances at which the numerical solution is calculated. It was seen that further increase in the number of spline zones did not improve the accuracy of the curve fit. A MATLAB algorithm (appendix C) is developed to calculate the coefficients of these cubic splines. [18] Various curve fits and their residuals with respect to the numerical solution are shown in following figures. The residuals of these curve fits show that as the number of spline zones increases, the curve fits are more accurate.

It is also seen that the residuals are higher in the initial and the final phases of the trajectory. As discussed in chapter 2, the spacecraft experiences higher acceleration during the initial and the final phases of the journey which results in the rapid change in the position during these phases. The rapid change in the position results in high curvature of the trajectory during these phases. Since the above cubic splines fit is calculated using fixed resolution throughout the entire trajectory, it fails to capture these high curvature regions precisely resulting in a less accurate curve fit in these regions. The source terms calculated using cubic splines fit (discussed in chapter 5) indicate lesser overall accuracy of cubic splines fit as compared to other methods discussed later in this chapter. Therefore, no effort has been made to use a multi-resolution cubic splines fit in order to capture the high curvature regions more precisely.

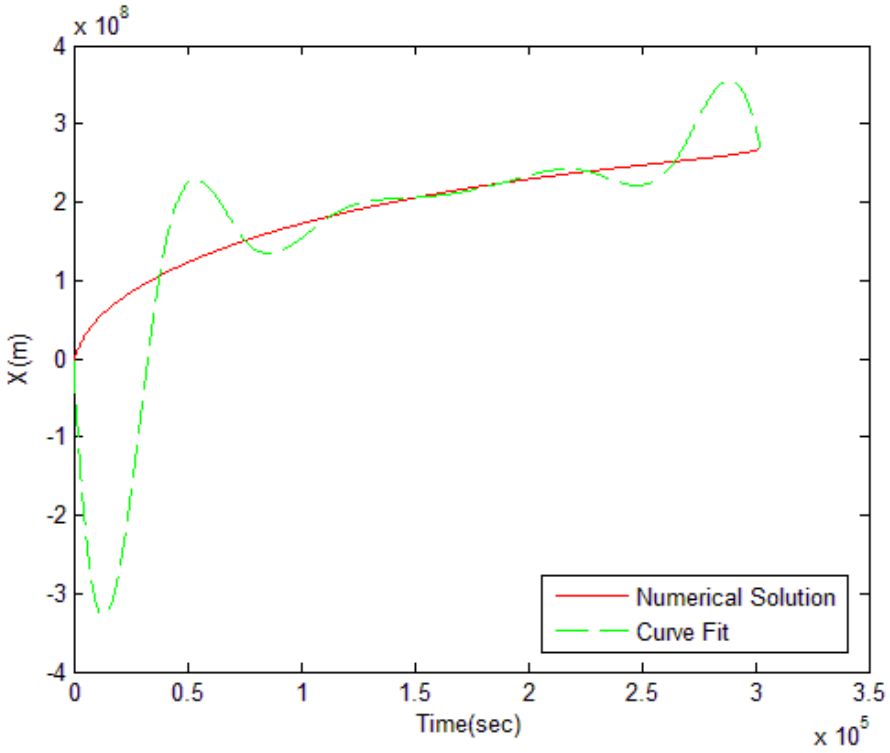

Figure 25. Curve Fit to *X* - Cubic Splines Using 8 Spline Zones

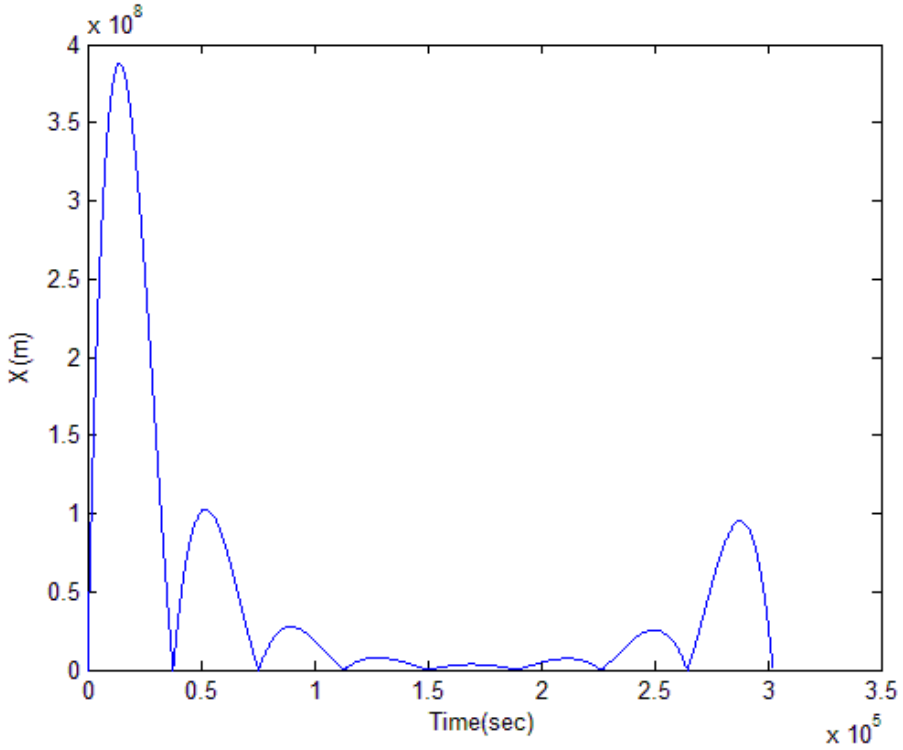

Figure 26. Curve Fit Residuals for *X* - Cubic Splines Using 8 Spline Zones

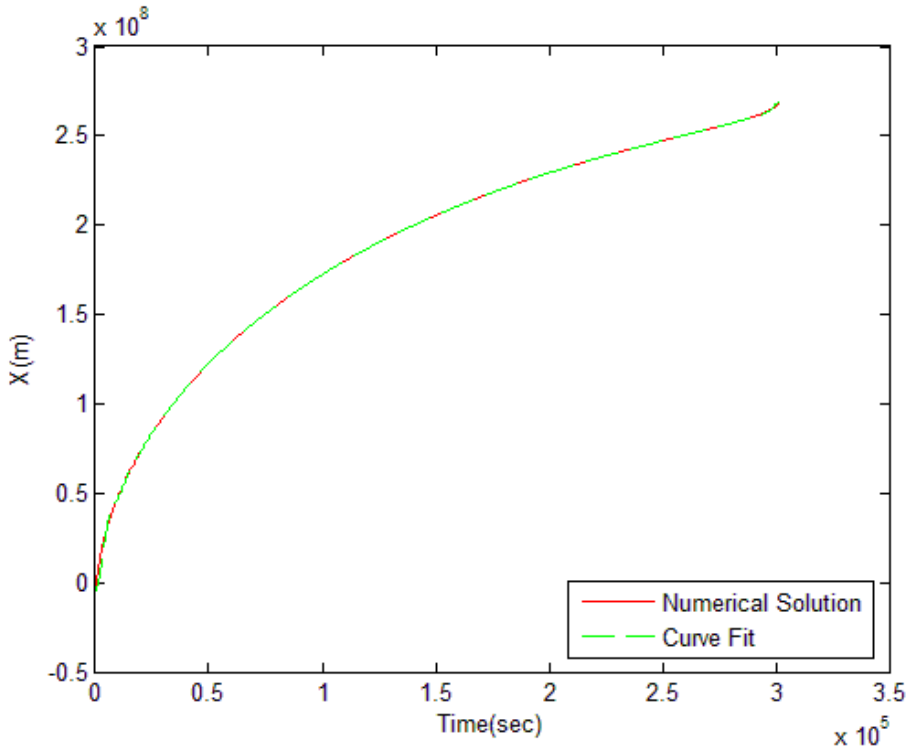

Figure 27. Curve Fit to *X* - Cubic Splines Using 63 Spline Zones

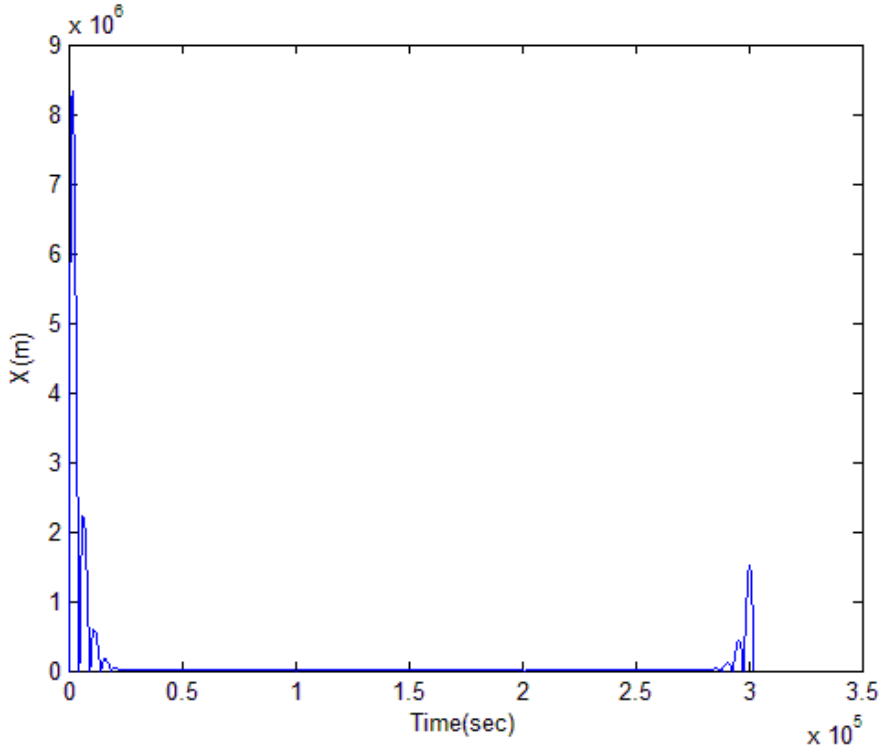

Figure 28. Curve Fit Residuals for *X* - Cubic Splines Using 63 Spline Zones

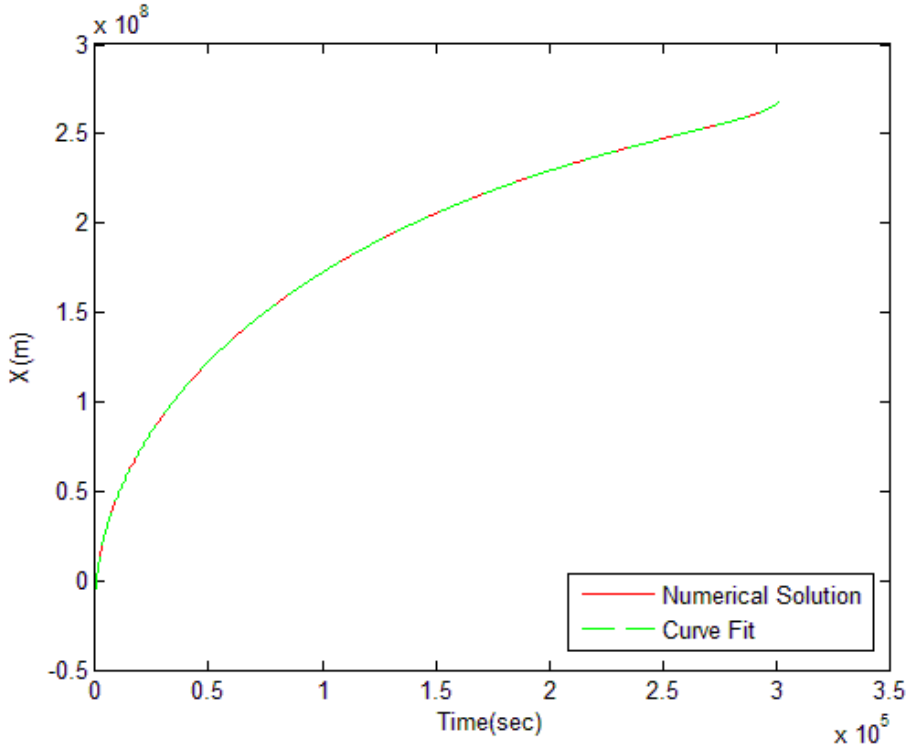

Figure 29. Curve Fit to *X* - Cubic Splines Using 504 Spline Zones

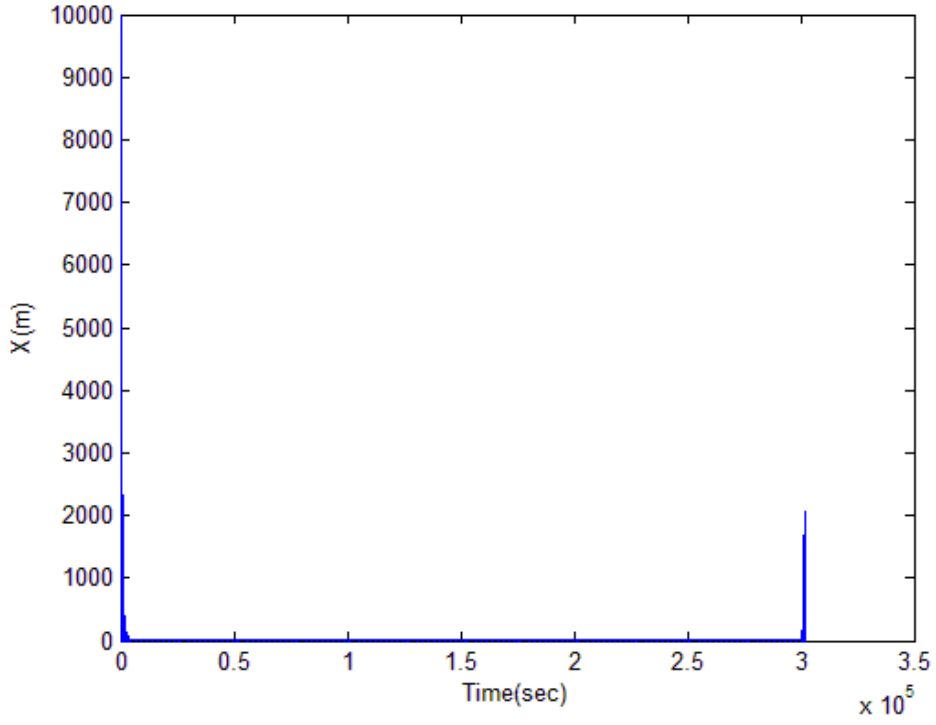

Figure 30. Curve Fit Residuals for *X* - Cubic Splines Using 504 Spline Zones

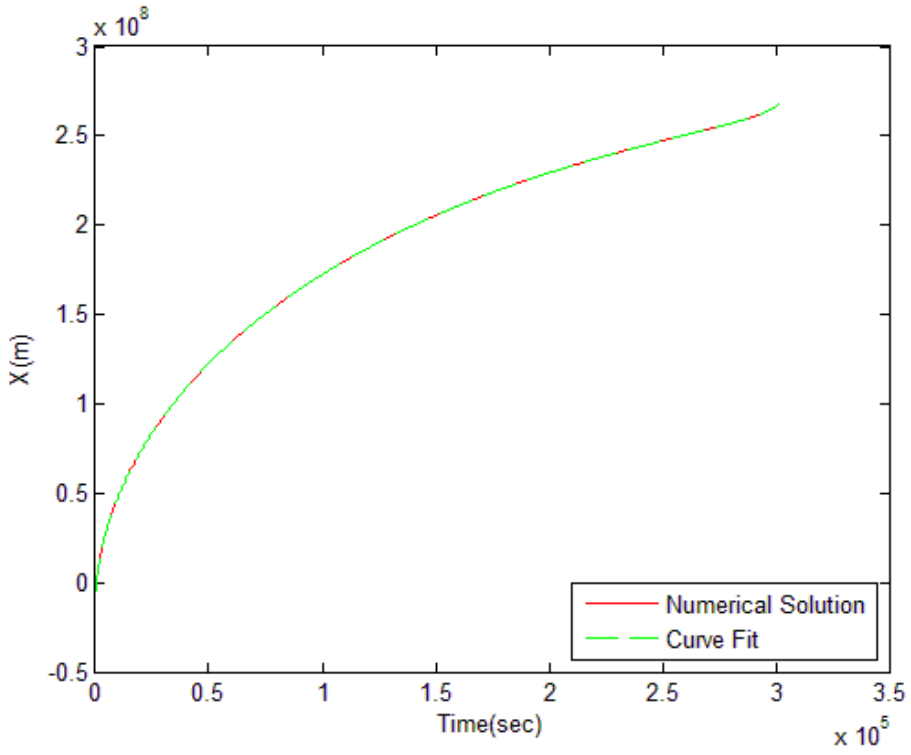

Figure 31. Curve Fit to *X* - Cubic Splines Using 5040 Spline Zones

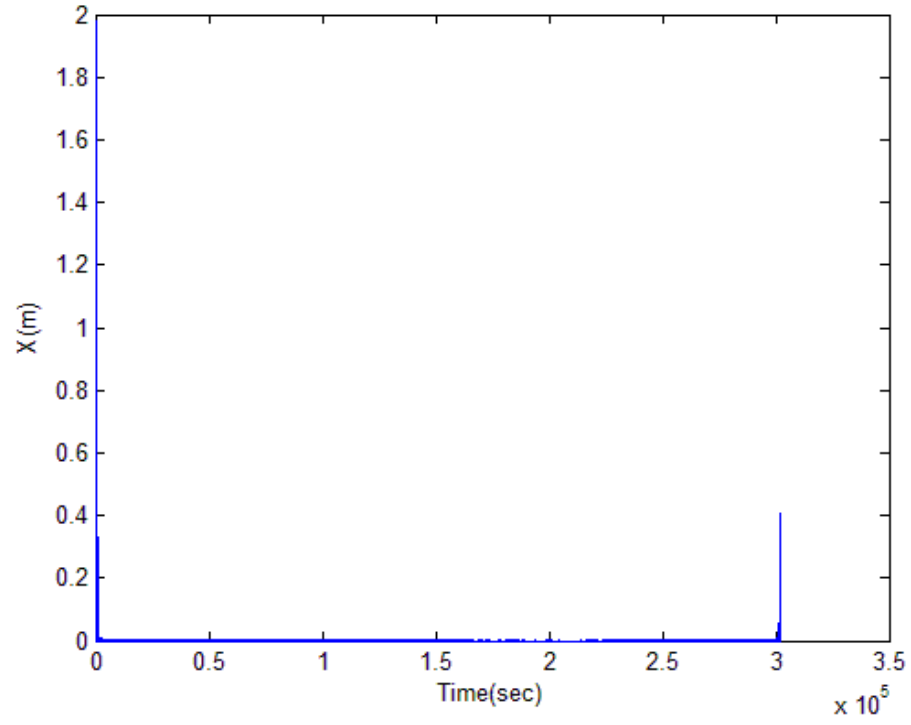

Figure 32. Curve Fit Residuals for *X* - Cubic Splines Using 5040 Spline Zones

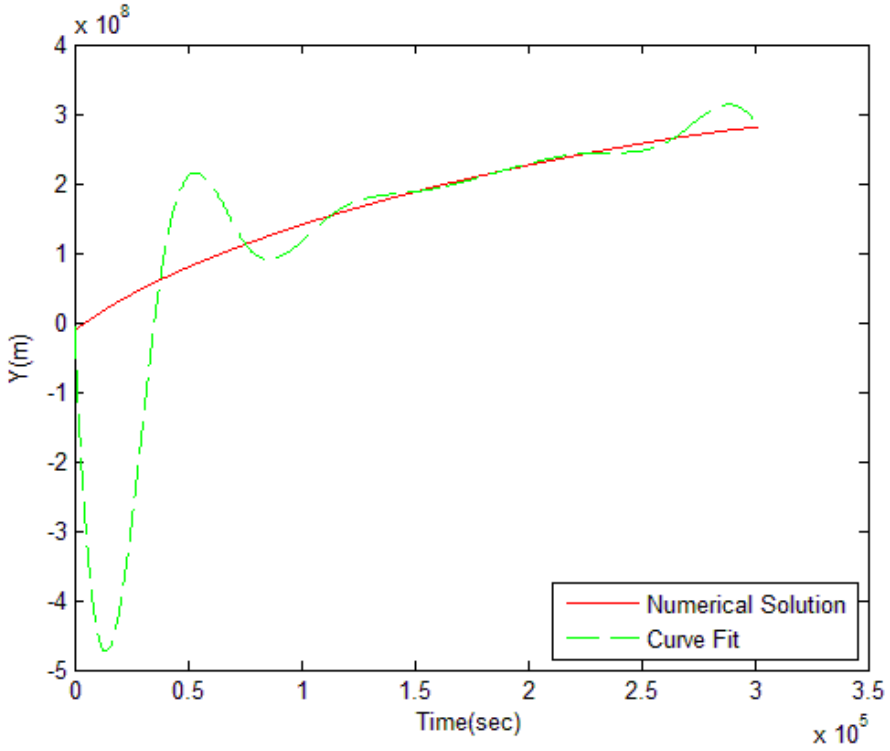

Figure 33. Curve Fit to *Y* - Cubic Splines Using 8 Spline Zones

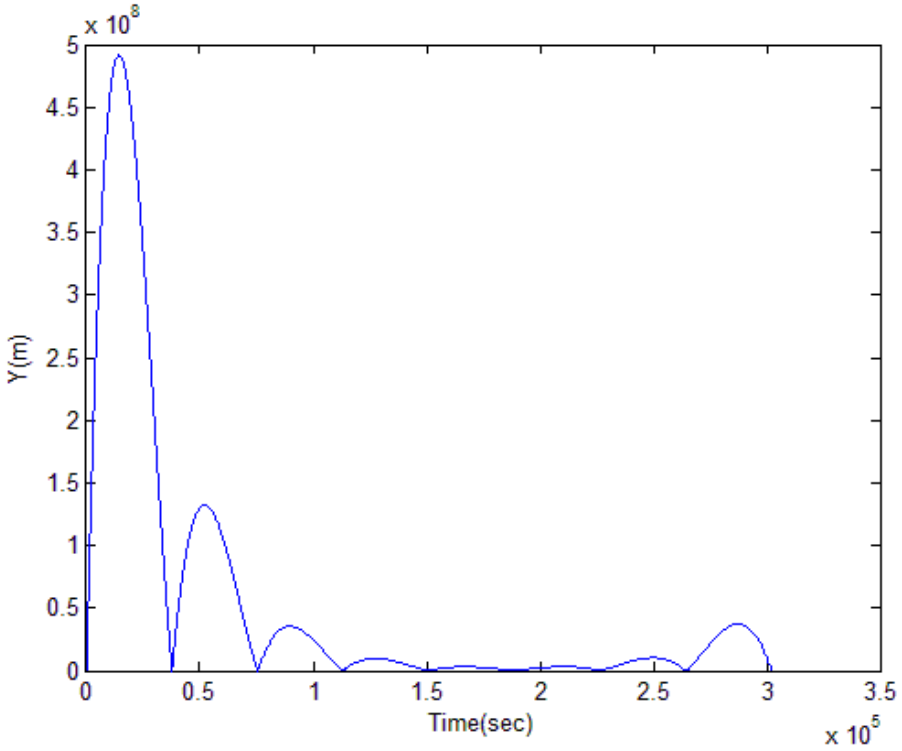

Figure 34. Curve Fit Residuals for *Y* - Cubic Splines Using 8 Spline Zones

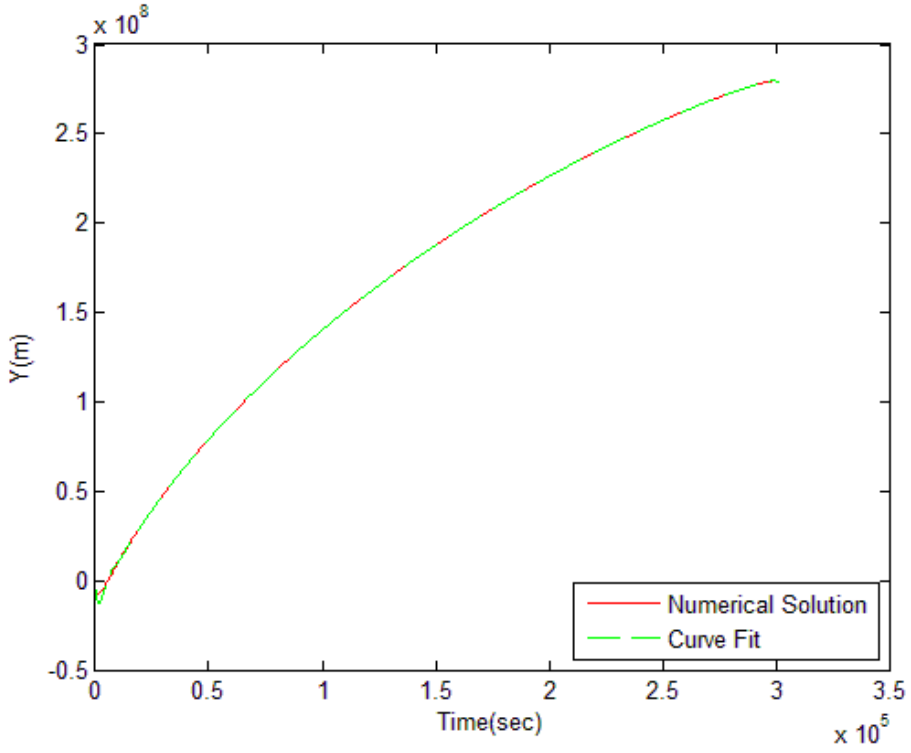

Figure 35. Curve Fit to *Y* - Cubic Splines Using 63 Spline Zones

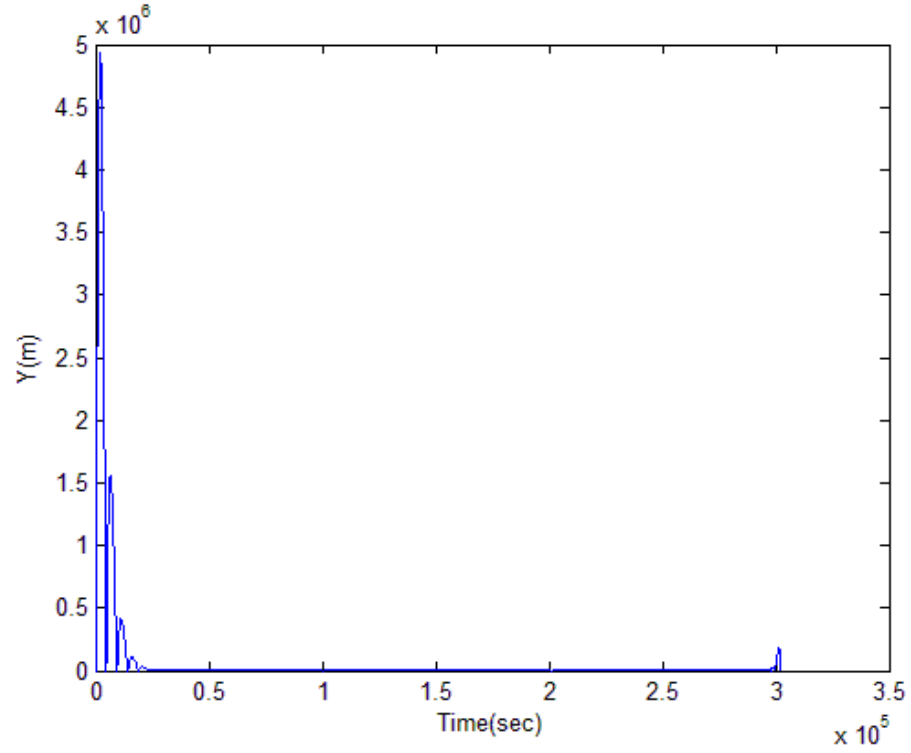

Figure 36. Curve Fit Residuals for *Y* - Cubic Splines Using 63 Spline Zones

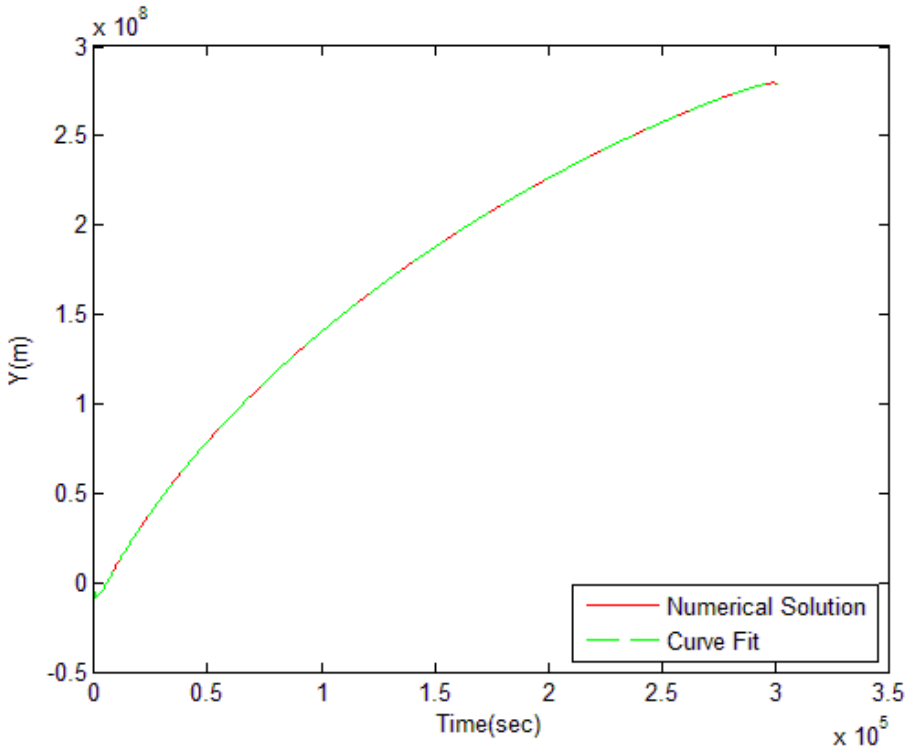

Figure 37. Curve Fit to *Y* - Cubic Splines Using 504 Spline Zones

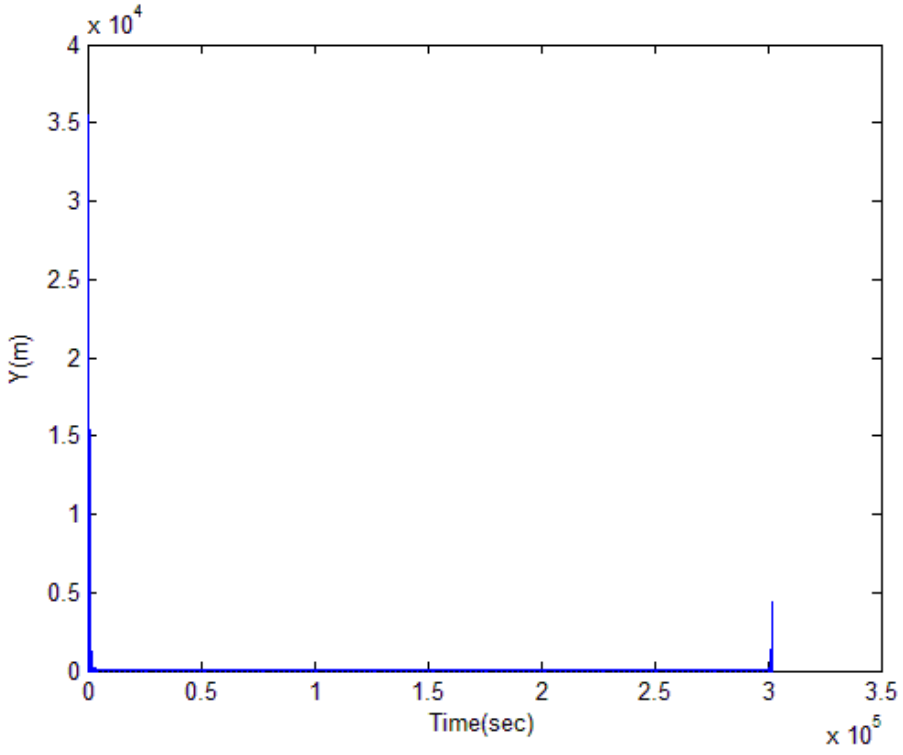

Figure 38. Curve Fit Residuals for *Y* - Cubic Splines Using 504 Spline Zones

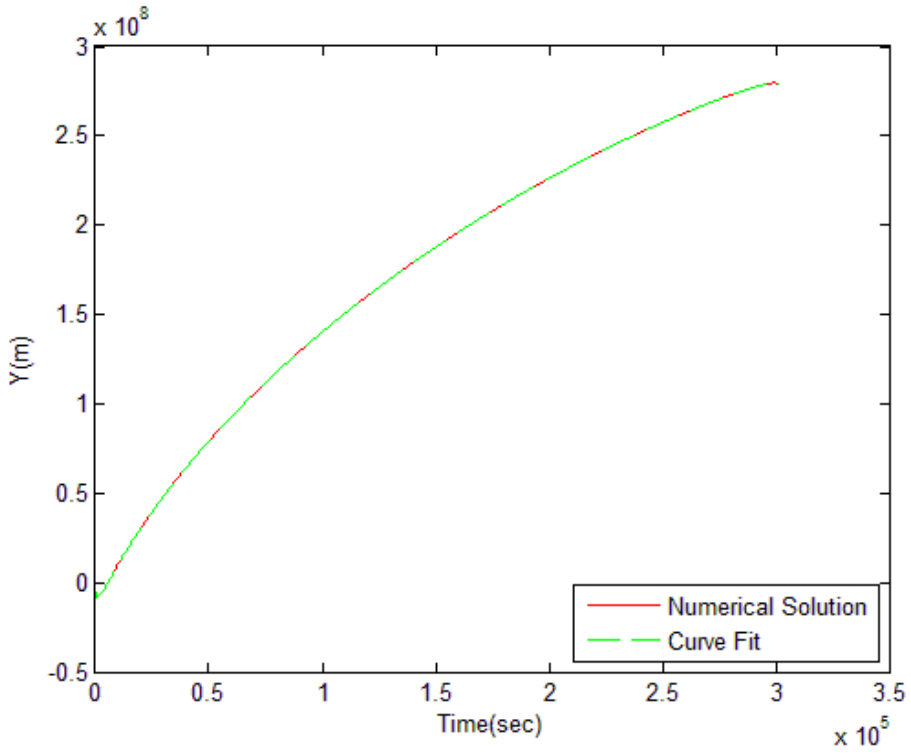

Figure 39. Curve Fit to *Y* - Cubic Splines Using 5040 Spline Zones

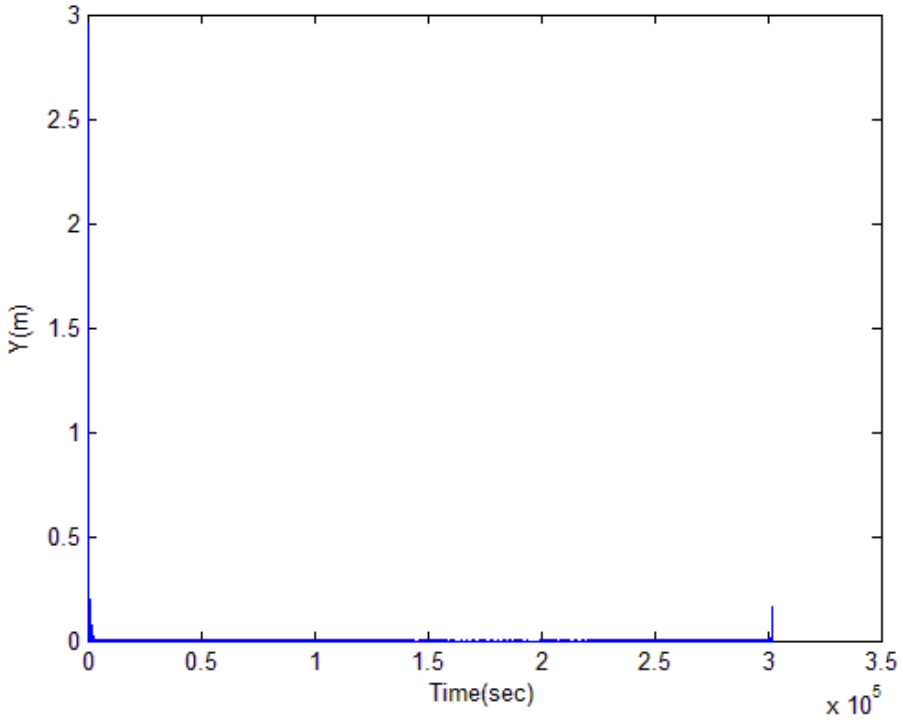

Figure 40. Curve Fit Residuals for *Y* - Cubic Splines Using 5040 Spline Zones

Fifth-degree Hermite Splines

An alternative approach to spline interpolation is to use "fifth-degree Hermite splines" instead of "cubic splines". The schematic of spline interpolation using fifth-degree Hermite splines is as shown in figure 41.

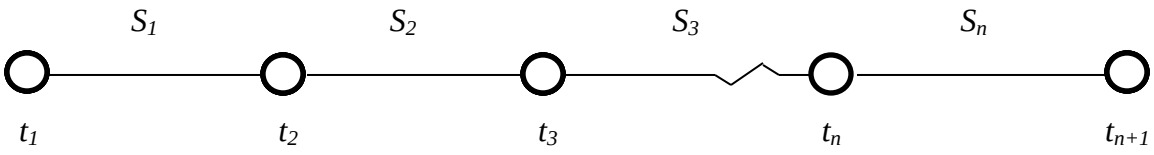

Figure 41. Schematic of Hermite Splines Interpolation

Here, the basic idea is to fit a fifth degree polynomial on each interval between points t_i and t_{i+1} for $i = 1,\ldots, n$. [18]

$$S_i(t) = a_i + b_i(t-t_i) + c_i(t-t_i)^2 + d_i(t-t_i)^3 + e_i(t-t_i)^4 + f_i(t-t_i)^5 \qquad (33)$$

This system has $n+1$ spline points (t_i for $i = 1,\ldots, n+1$) and n spline zones (S_i for $i = 1,\ldots, n$). The conditions that are used to construct these polynomials are explained below for given X, $\dot{X}$, $\ddot{X}$ data.

$$S_i(t_i) = X_i \qquad i = 1,\ldots., n \qquad (34)$$

$$S_n(t_{n+1}) = X_{n+1}$$

$$\dot{S}_i(t_i) = \dot{X}_i \qquad i = 1,\ldots., n \qquad (35)$$

$$\dot{S}_n(t_{n+1}) = \dot{X}_{n+1}$$

$$S_i(t_i) = S_{i-1}(t_i) \qquad i = 2,\ldots., n \qquad (36)$$

$$\dot{S}_i(t_i) = \dot{S}_{i-1}(t_i) \qquad i = 2,\ldots., n \qquad (37)$$

$$\ddot{S}_i(t_i) = \ddot{S}_{i-1}(t_i) \qquad i = 2,\ldots., n \qquad (38)$$

$$\dddot{S}_i(t_i) = \dddot{S}_{i-1}(t_i) \qquad i = 2,\ldots., n \qquad (39)$$

$$\ddot{S}_1(t_1) = \ddot{X}_1 \qquad (40)$$

$$\ddot{S}_n(t_{n+1}) = \ddot{X}_{n+1}$$

Conditions (34) set the value at each node point. Conditions (35) set the first derivative at each node point. Typically, the cubic Hermite spline form consists of two control points and two control tangents at the boundaries for each polynomial. Conditions (34) and (35) make each polynomial of the spline fit to be in "Hermite" form. Here, the additional degrees of freedom in the fifth degree polynomial are used to enforce additional continuity requirements. Condition (36) makes the solution 0^{th} order continuous. Condition (37) makes the solution 1^{st} order continuous. Condition (38) makes the solution 2^{nd} order continuous, while condition (39) makes the solution 3^{rd} order continuous. Conditions (40) are the two additional end point conditions.

Curve fits to *X* and *Y* using fifth-degree Hermite splines are calculated using 8, 63, 504 and 5040 spline zones. A MATLAB algorithm (appendix D) is developed to calculate the coefficients of these fifth-degree Hermite splines. [18] Various curve fits and their residuals with respect to the numerical solution are shown in figures 42 to 57. The residuals of these curve fits show that as the number of spline zones increases, the curve fits are more accurate.

The phenomenon of higher residuals in the initial and the final phases of the trajectory is also seen in the case of fifth-degree Hermite splines curve fit. An effort has been made to address this phenomenon by using an iterative multi-resolution fifth-degree Hermite splines fit in order to capture the high curvature regions more precisely. A MATLAB code (appendix E) is developed for this purpose. In this code, the trajectory is divided into four regions of equal amounts of time with region 1 and 4 covering the initial

and the final phases of the trajectory respectively. To start with, each region is divided into two spline zones (total of eight spline zones). A curve fit is calculated using these spline zones and each region is checked for the maximum value of residual which should be below a selected threshold. If the maximum value of residual in a particular region is below a selected threshold, addition of spline points to that particular region is terminated. One additional spline point is added to each region where the threshold is not met. For each iteration, the spline points within each region are evenly distributed. However, spline points falling in the middle of a time step are rounded to the nearest integer number of time steps. In this manner, the maximum value of residual in each region is made to be somewhat consistent indicating that the curve fit exhibits nearly the same level of accuracy throughout the trajectory.

It is seen that, if the selected threshold is below 10^{-3} meters, the maximum value of residual fails to converge. Therefore, the threshold was selected to be 10^{-3} meters. A reason for this is that the number of spline zones in certain regions began to approach the number of data points. Tables 2 and 3 list, for the curve fits to X and Y, the number of spline zones required in each region so that the maximum value of residual in that region is below the selected threshold. It can be seen that the number of spline zones required by regions 1 and 4 is much higher than that required by regions 2 and 3.

Region	No. of Spline Zones
1	1708
2	24
3	17
4	933
Total	2682

Table 2. Region Wise Number of Spline Zones Using Multi-resolution Fifth-degree Hermite Splines Curve Fit to *X*

Region	No. of Spline Zones
1	1779
2	20
3	15
4	1043
Total	2857

Table 3. Region Wise Number of Spline Zones Using Multi-resolution Fifth-degree Hermite Splines Curve Fit to *Y*

Figures 58 and 59 show the plot of the residuals with respect to time for *X* and *Y* using the above discussed multi-resolution curve fit. It can be seen that, though this curve fit requires less number of total spline zones and also maintains the level of accuracy throughout the trajectory, the overall accuracy of this curve fit is fairly similar to that of the fixed-resolution curve fit discussed earlier.

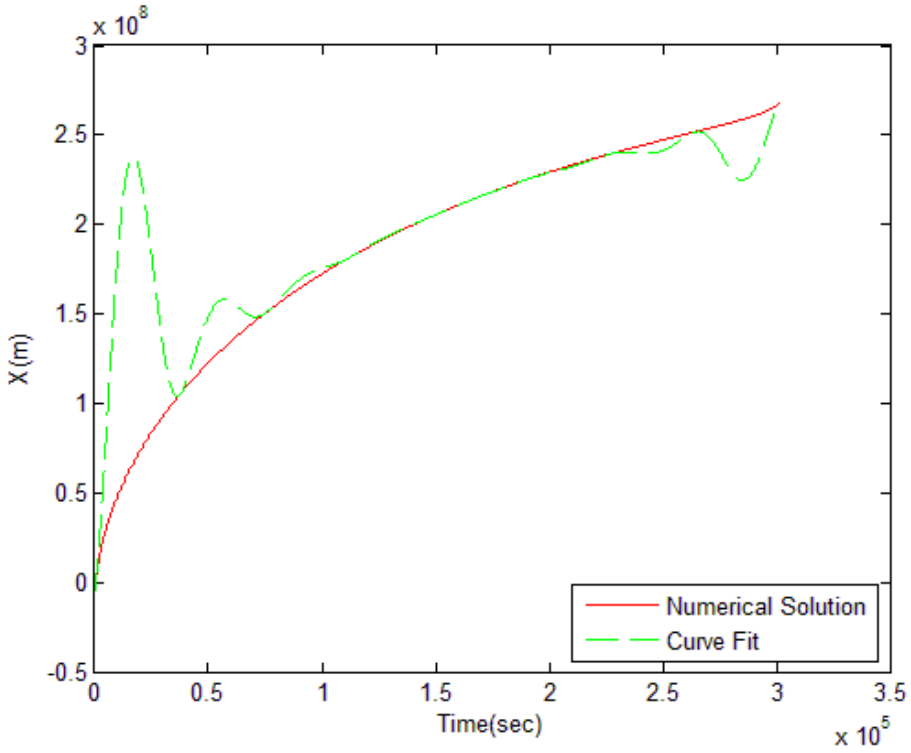

Figure 42. Curve Fit to *X* – Fifth-degree Hermite Splines Using 8 Spline Zones

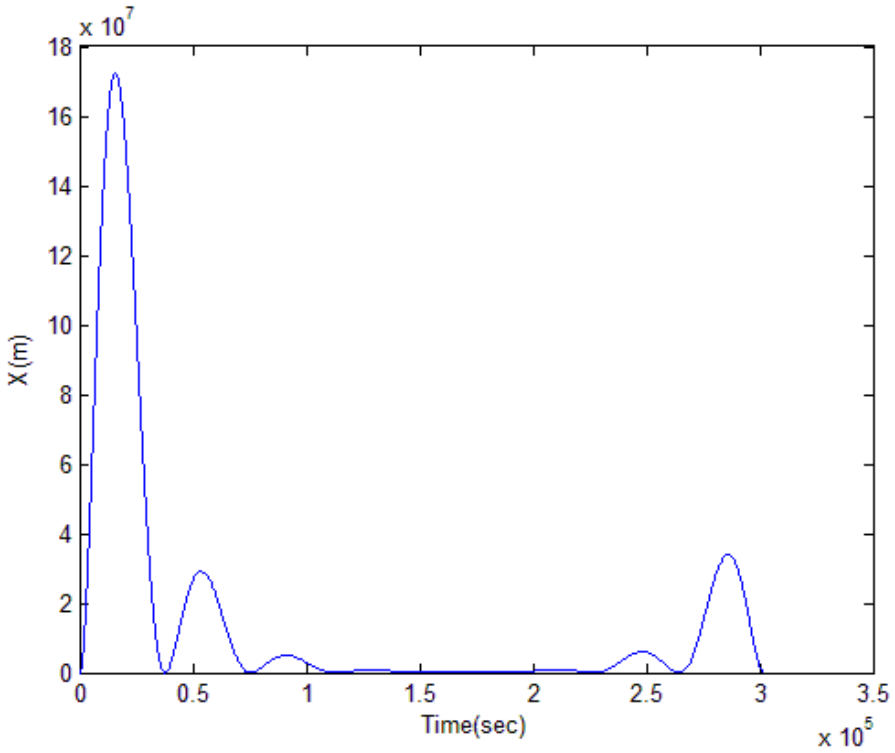

Figure 43. Curve Fit Residuals for *X* - Fifth-degree Hermite Splines Using 8 Spline Zones

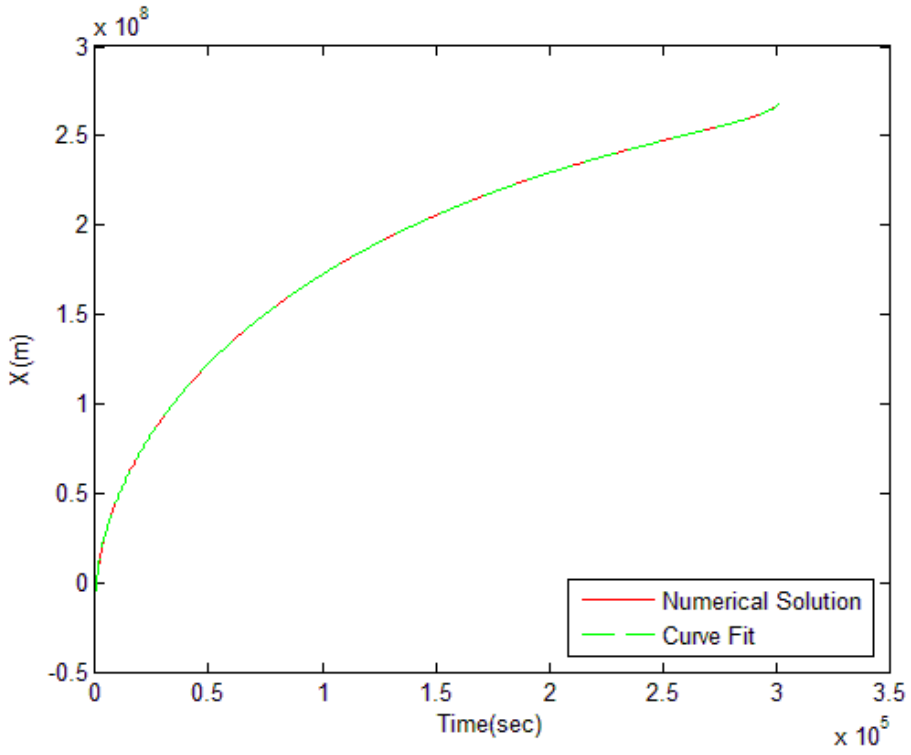

Figure 44. Curve Fit to *X* – Fifth-degree Hermite Splines Using 63 Spline Zones

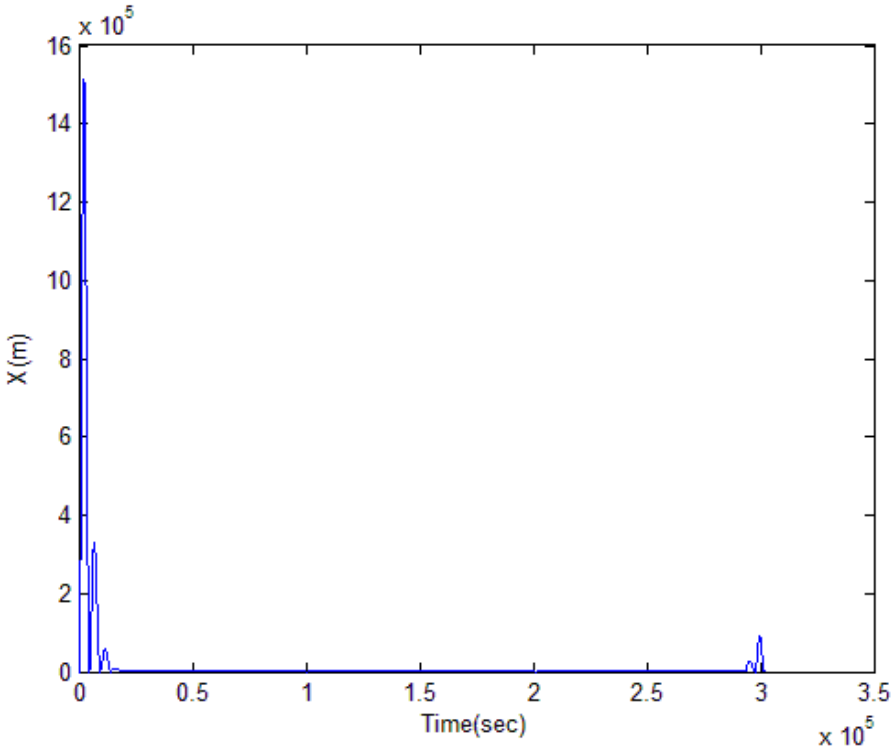

Figure 45. Curve Fit Residuals for *X* - Fifth-degree Hermite Splines Using 63 Spline Zones

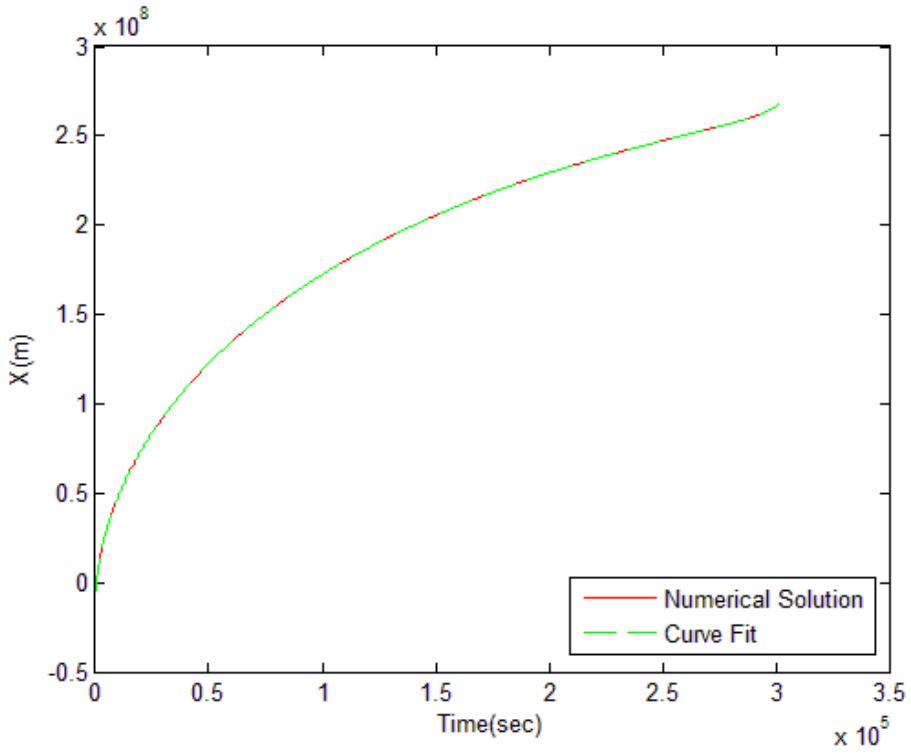

Figure 46. Curve Fit to *X* – Fifth-degree Hermite Splines Using 540 Spline Zones

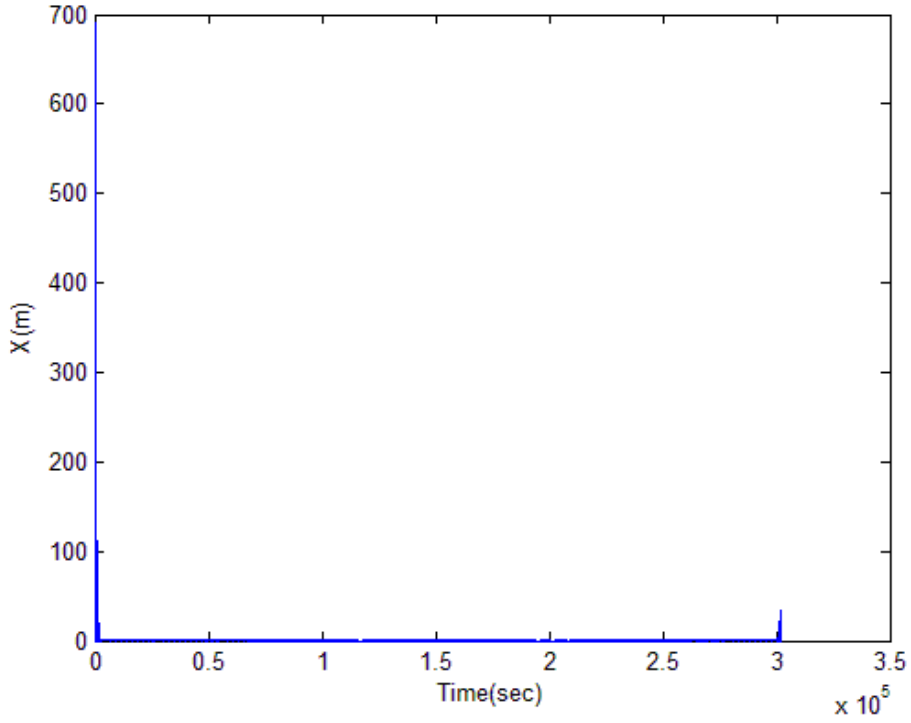

Figure 47. Curve Fit Residuals for *X* - Fifth-degree Hermite Splines Using 540 Spline Zones

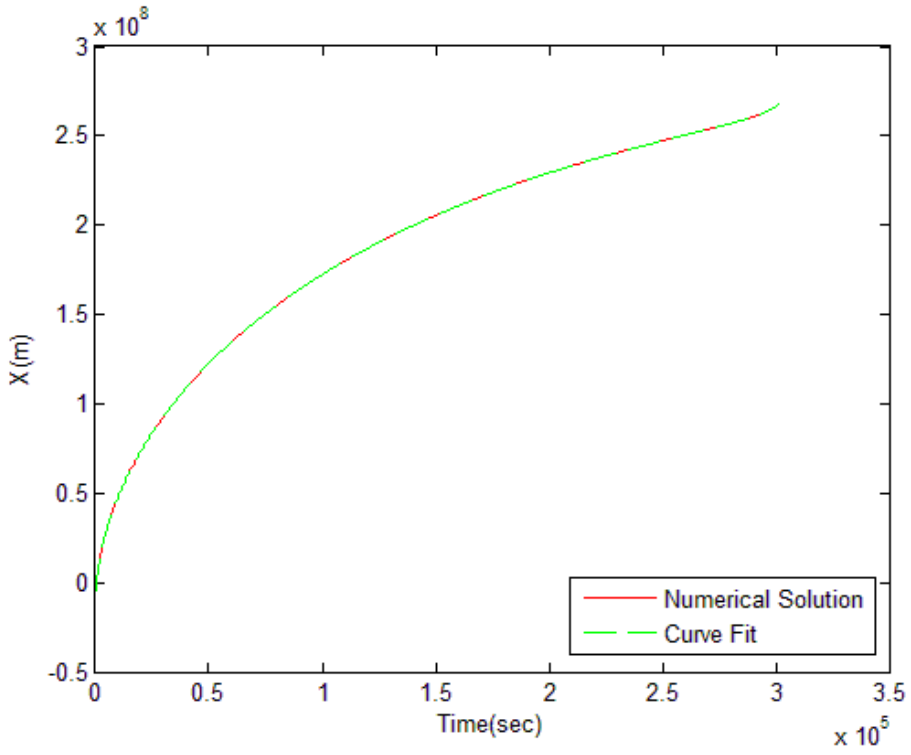

Figure 48. Curve Fit to *X* – Fifth-degree Hermite Splines Using 5040 Spline Zones

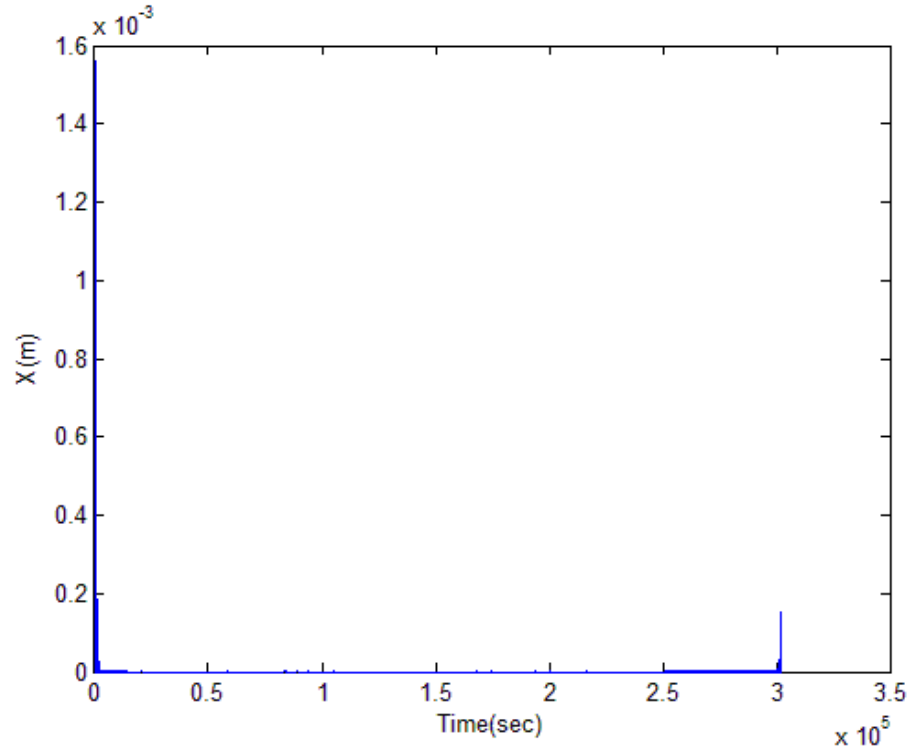

Figure 49. Curve Fit Residuals for *X* - Fifth-degree Hermite Splines Using 5040 Spline Zones

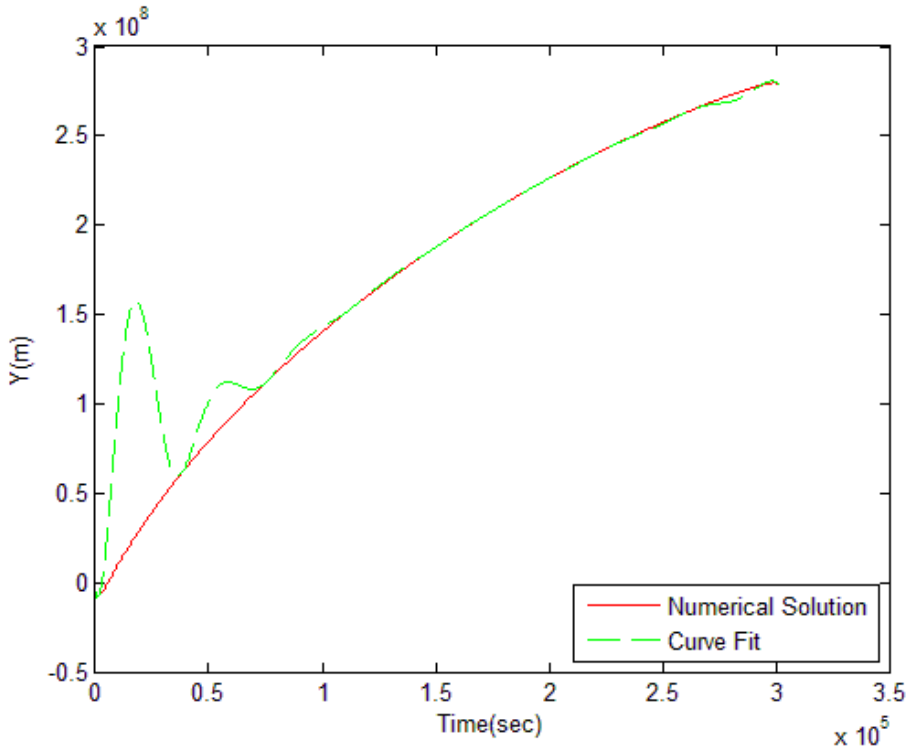

Figure 50. Curve Fit to *Y* – Fifth-degree Hermite Splines Using 8 Spline Zones

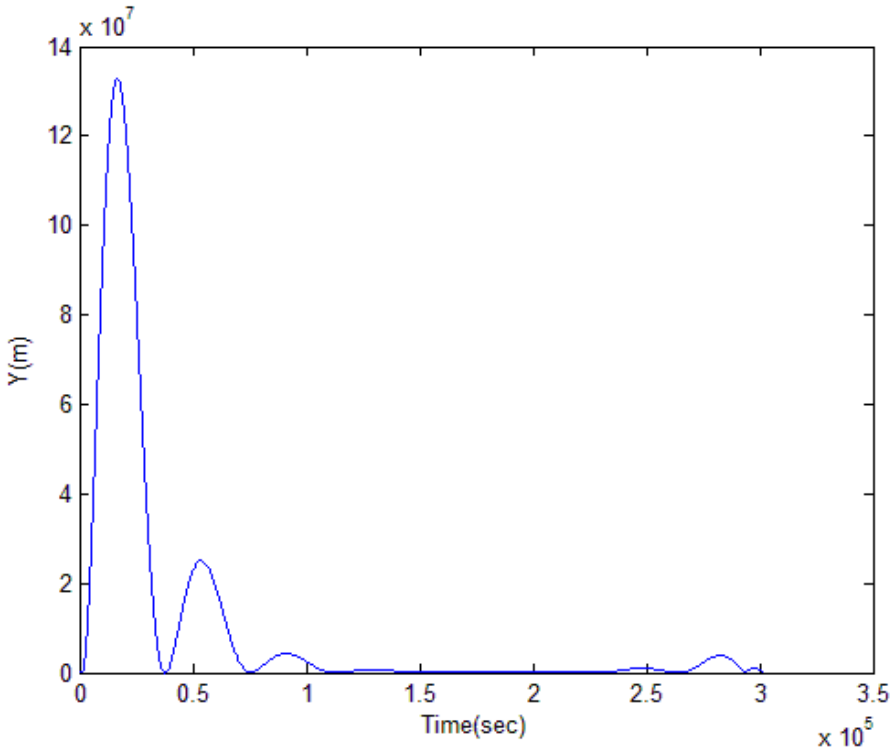

Figure 51. Curve Fit Residuals for *Y* - Fifth-degree Hermite Splines Using 8 Spline Zones

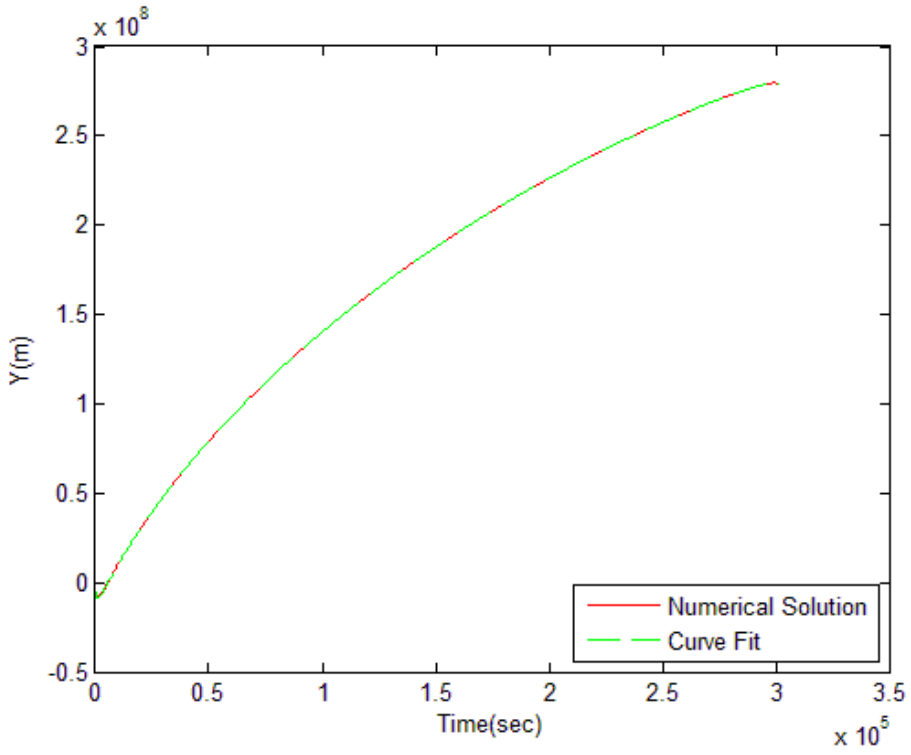

Figure 52. Curve Fit to *Y* – Fifth-degree Hermite Splines Using 63 Spline Zones

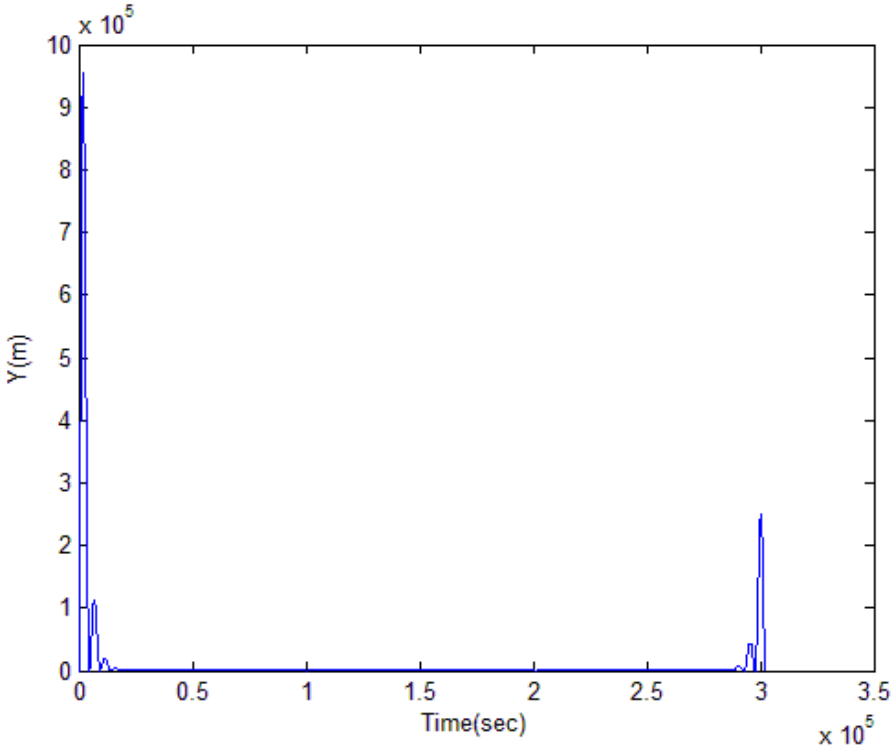

Figure 53. Curve Fit Residuals for *Y* - Fifth-degree Hermite Splines Using 63 Spline Zones

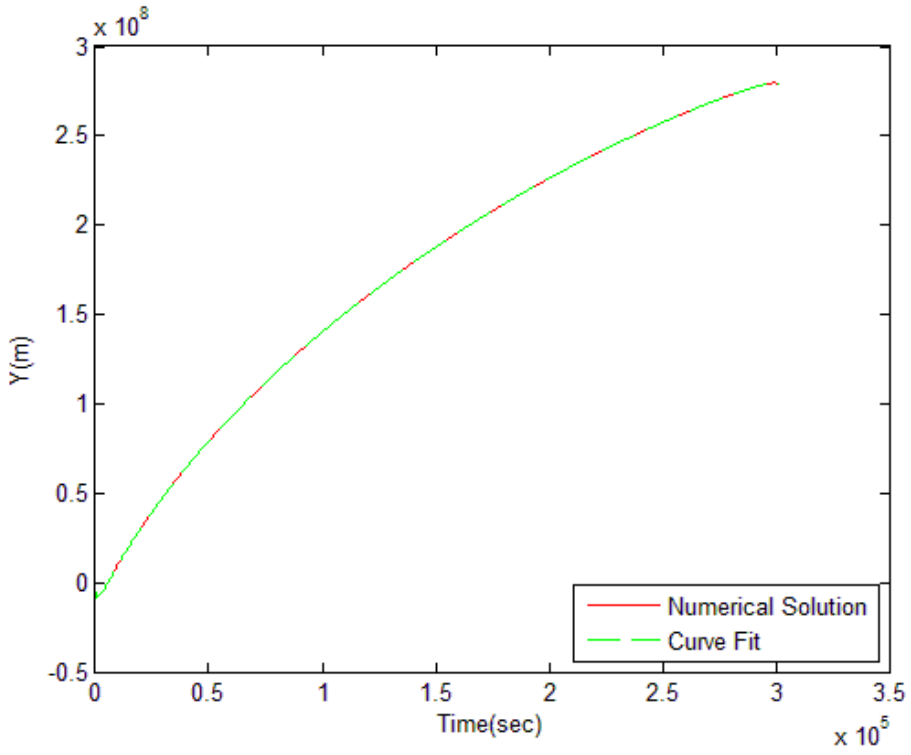

Figure 54. Curve Fit to *Y* – Fifth-degree Hermite Splines Using 504 Spline Zones

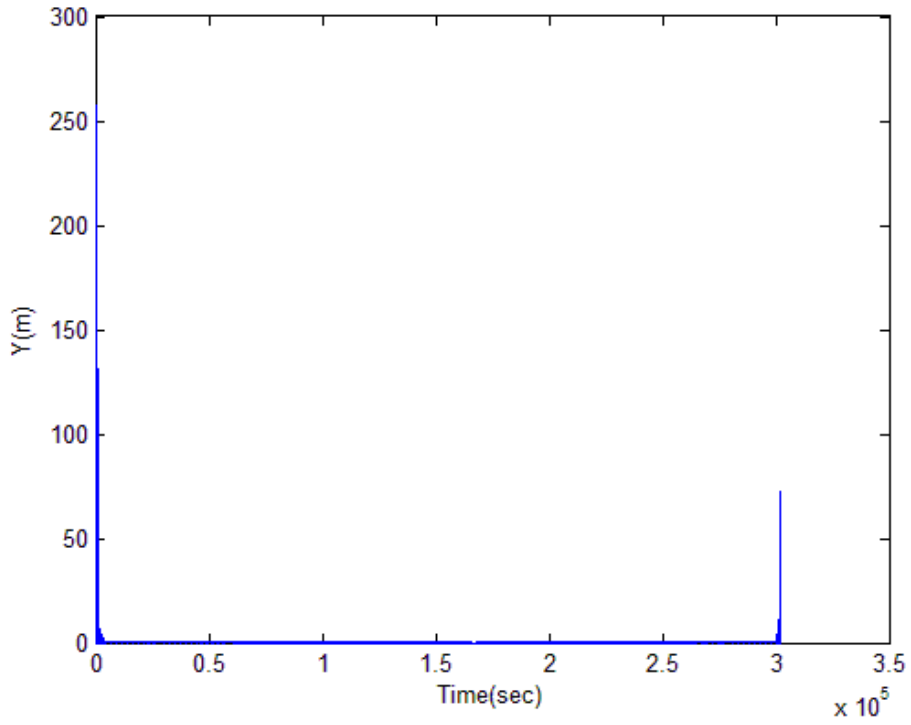

Figure 55. Curve Fit Residuals for *Y* - Fifth-degree Hermite Splines Using 504 Spline Zones

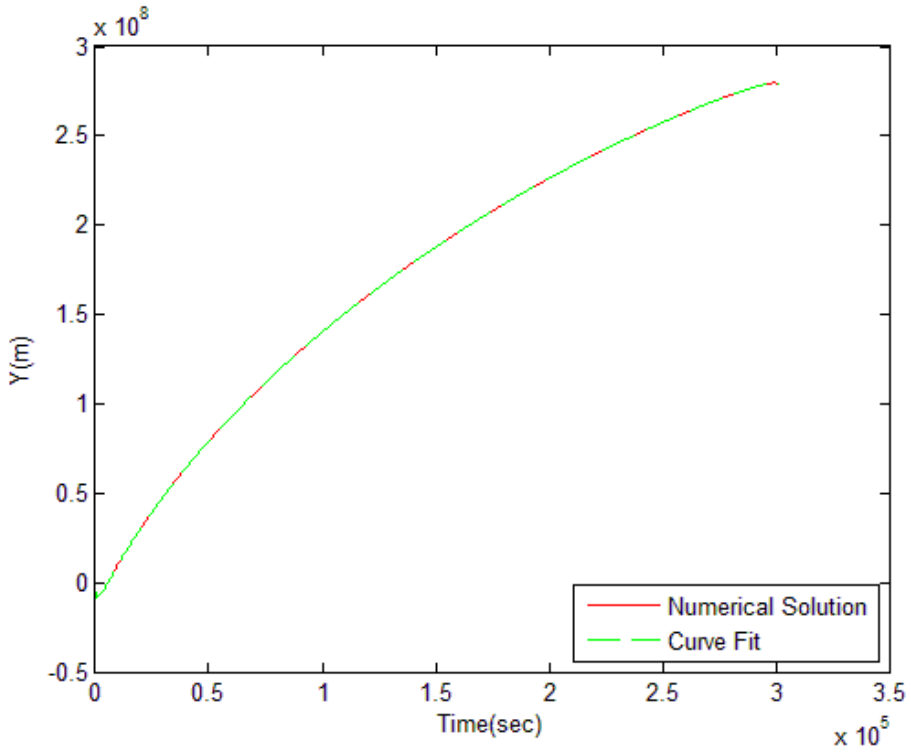

Figure 56. Curve Fit to *Y* – Fifth-degree Hermite Splines Using 5040 Spline Zones

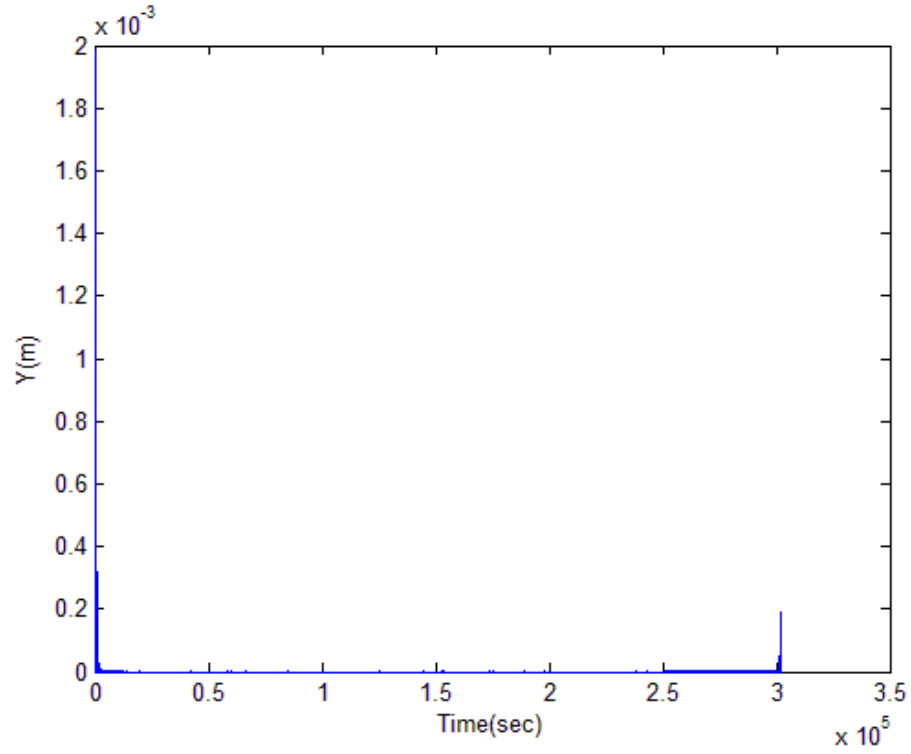

Figure 57. Curve Fit Residuals for *Y* - Fifth-degree Hermite Splines Using 5040 Spline Zone

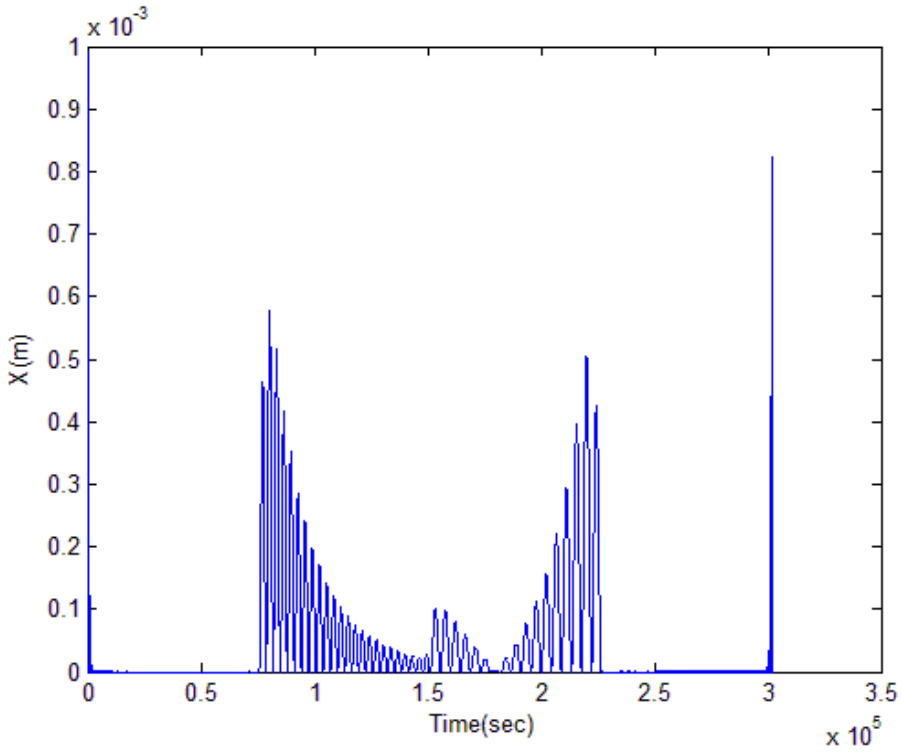

Figure 58. Curve Fit Residuals for *X* – Multi-resolution Fifth-degree Hermite Splines Fit

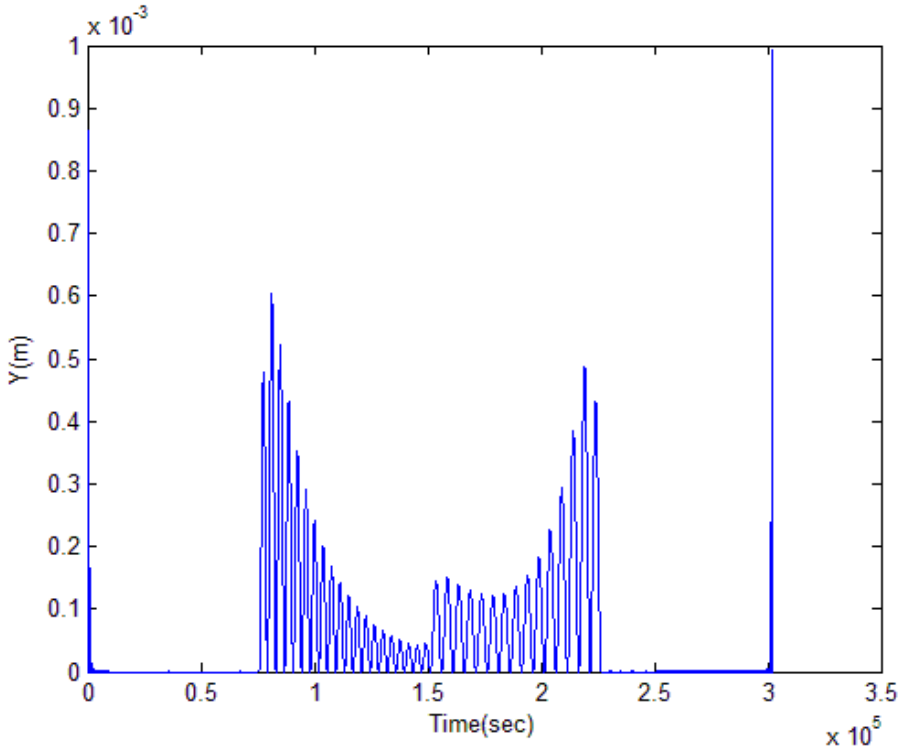

Figure 59. Curve Fit Residuals for *Y* - Multi-resolution Fifth-degree Hermite Splines Fit

CHAPTER 5

GENERATING ANALYTICAL SOURCE TERMS

As mentioned earlier, in MNP, for a curve fit to be considered accurate the analytical source terms generated by it should be small in magnitude. The smaller the source terms, the nearer the nearby problem is to the original problem. Source terms are calculated for all the curve fits discussed in the previous chapter. Consider rewriting Eq. (14) in an operator form.

$$\mathbf{L}(\mathbf{r}(t),t) = \frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} + \frac{\mu_e}{r^3}\mathbf{r} + \mu_m\left(\frac{\mathbf{d}}{d^3} + \frac{\boldsymbol{\rho}}{\rho^3}\right) = \mathbf{0} \tag{41}$$

The curve fits for X and Y generated in the previous chapter can be assembled into a curve fit for the position vector, labeled $\bar{\mathbf{r}}(t)$. Operating the original problem on this curve fit defines the source terms (appendix H), $\mathbf{s}(t)$.

$$\mathbf{L}(\bar{\mathbf{r}}(t),t) \equiv \mathbf{s}(t) \tag{42}$$

By construction, the curve fit is the exact solution of a modified equation, which is the nearby problem.

$$\mathbf{L}(\mathbf{r}(t),t) - \mathbf{s}(t) = \mathbf{0} \tag{43}$$

Therefore, the source terms are equivalent to a set of time-dependent perturbing accelerations that result in a trajectory following the curve fit. In Eq. (43), as $\mathbf{s}(t)$ approaches zero, the nearby problem approaches the original problem.

We will now see the magnitude of the analytical source terms generated using various curve fits discussed in the previous chapter.

Analytical Source Terms Using Least Squares

Following figures show the source-term histories for both X and Y for the least-squares curve fits using varying degrees of polynomials. It can be seen that the magnitude of the source terms increases as the degree of the polynomial increases. Note that, though the curve fit using a higher-degree polynomial is more accurate than that using a lower-degree polynomial; the source terms generated by using the higher-degree polynomial curve fit are larger in magnitude than those generated by using the lower-degree polynomial curve fit. This can be attributed to the fact that the least-squares solution is trying to minimize errors only in position. The derivatives of the position do not play any role in calculating the curve fit. The larger magnitude of the source terms indicates that the least squares is not a feasible option for the construction of the nearby problem.

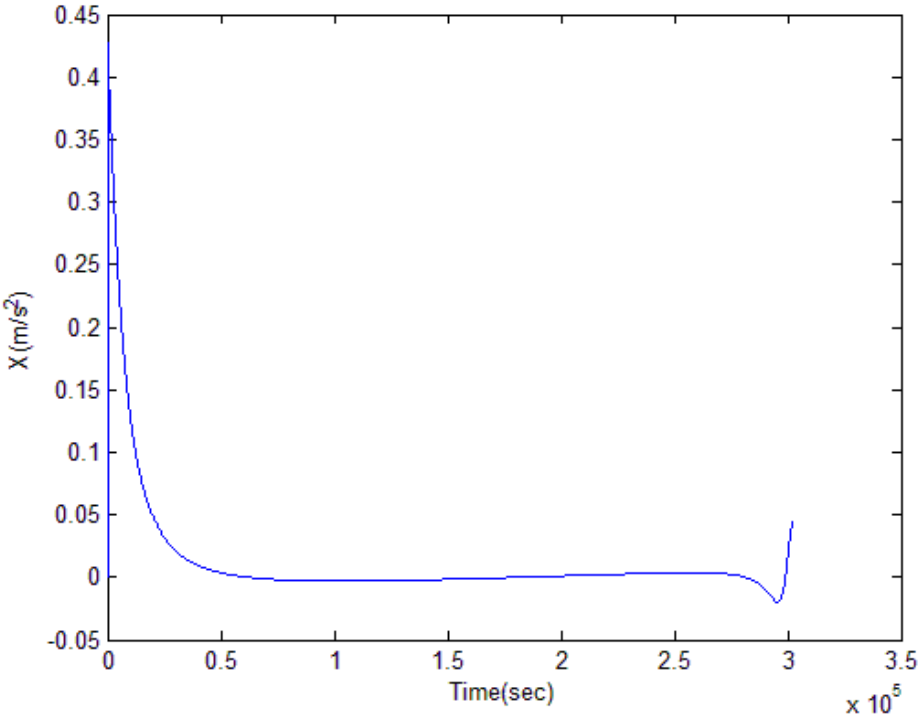

Figure 60. Source Term Histories for *X* - Least Squares Using 3rd Degree Polynomial

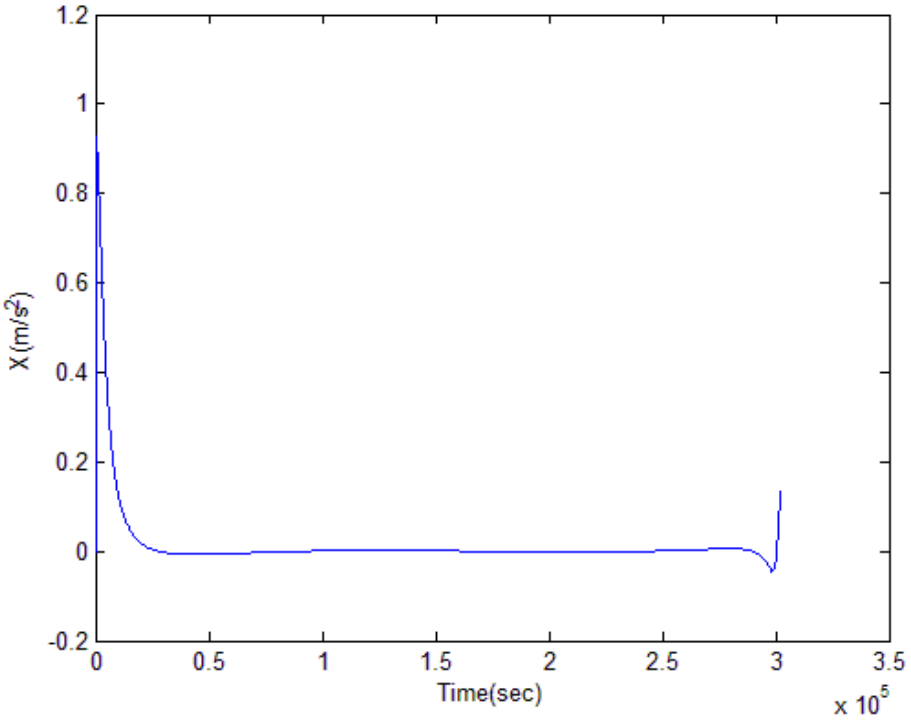

Figure 61. Source Term Histories for *X* - Least Squares Using 5th Degree Polynomial

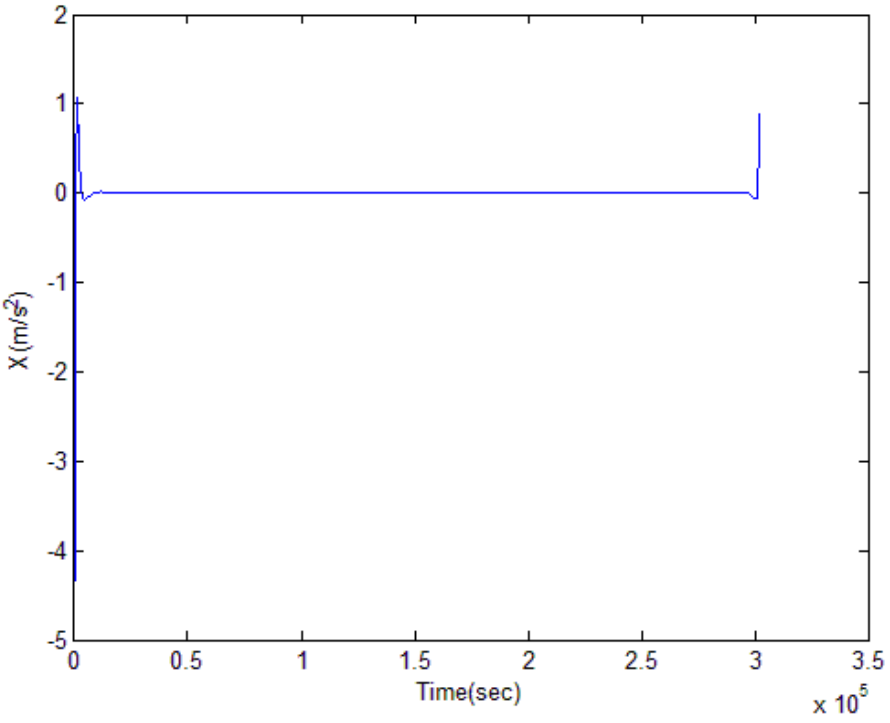

Figure 62. Source Term Histories for *X* - Least Squares Using 20th Degree Polynomial

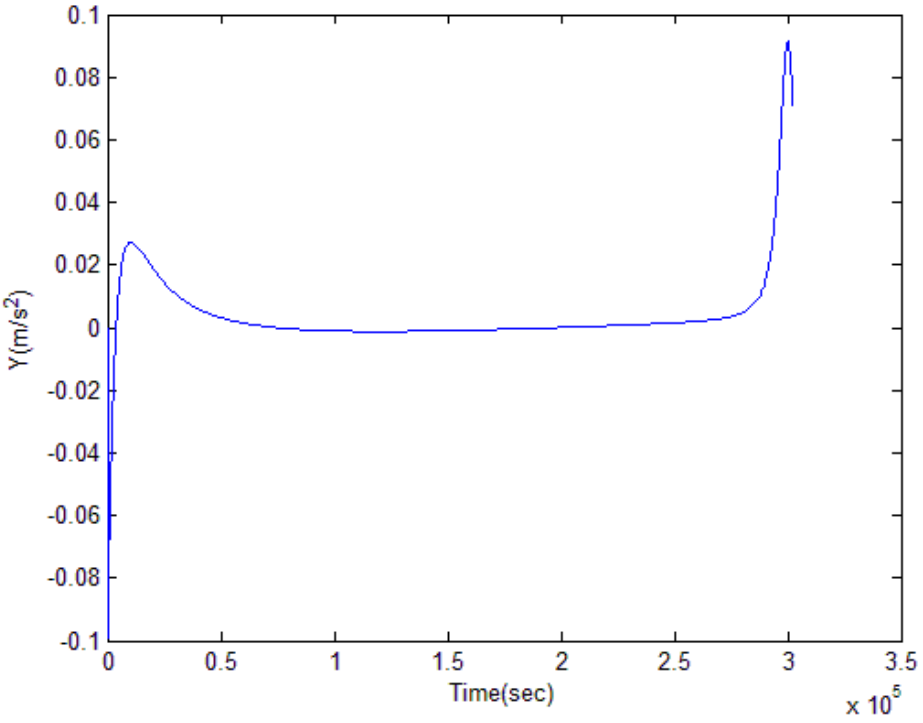

Figure 63. Source Term Histories for *Y* - Least Squares Using 3rd Degree Polynomial

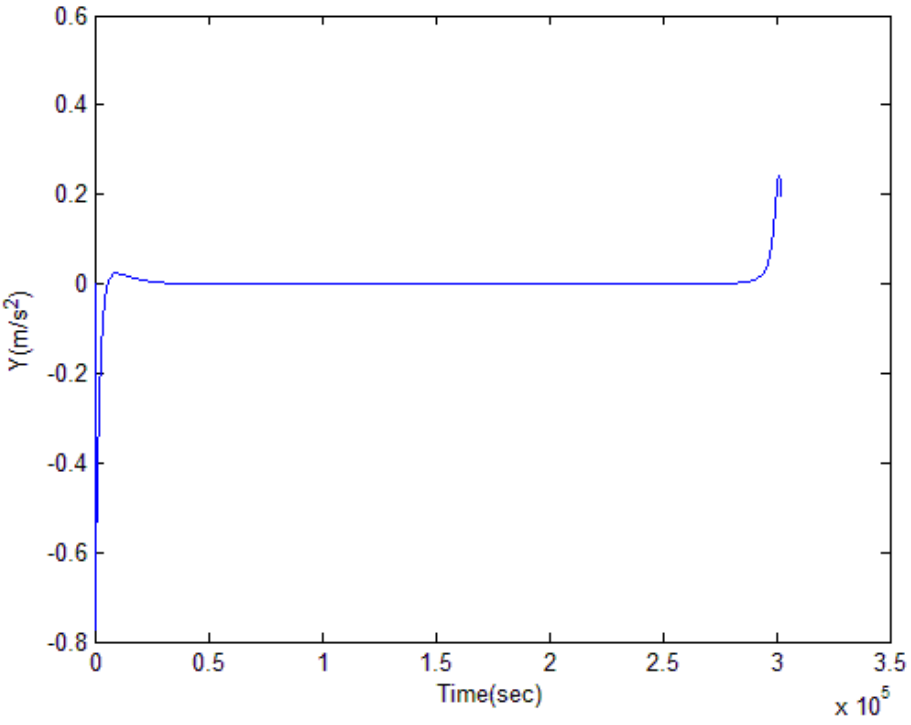

Figure 64. Source Term Histories for *Y* - Least Squares Using 5th Degree Polynomial

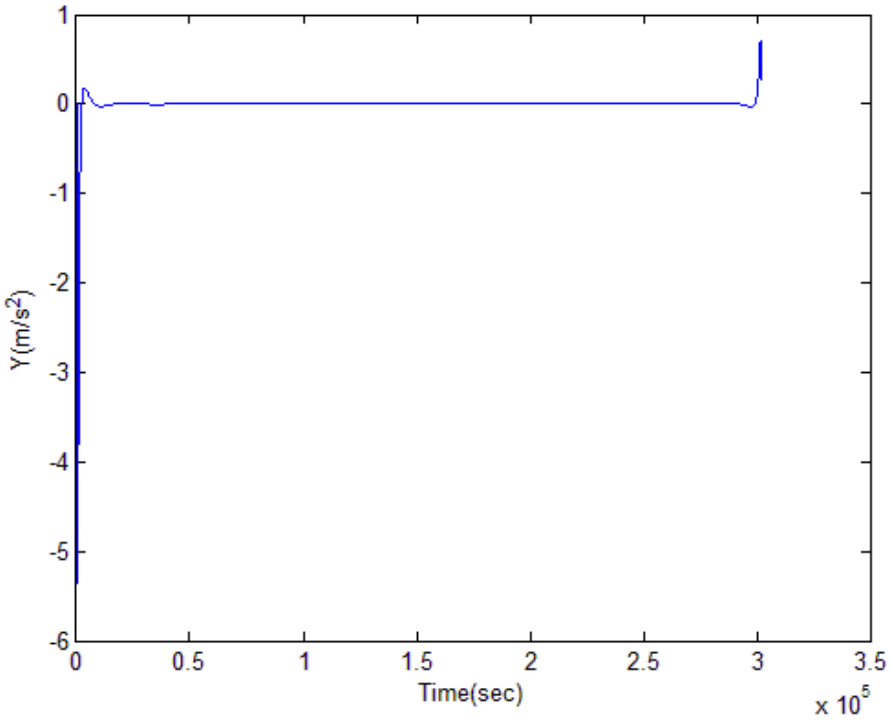

Figure 65. Source Term Histories for *Y* - Least Squares Using 20th Degree Polynomial

Analytical Source Terms Using Cubic Splines

Following figures show the source-term histories for both X and Y for the cubic splines curve fits using varying numbers of spline zones. It can be seen that the magnitude of the source terms decreases as the number of spline zones increases. Using moderate number of spline zones, the magnitude of the source terms generated is small. However, close examination of following figures shows that the source terms exhibit slope discontinuities at the spline points. This is because cubic splines are only C^2 continuous. As mentioned earlier, in MNP, the curve fit should maintain a particular order of continuity in order to maintain the slope continuity of the source terms. The equations of motion of three-body dynamics are second order differential equations and demand C^3 continuity in the curve fit. Since the continuity criterion is not satisfied, cubic splines cannot be used to construct the nearby problem.

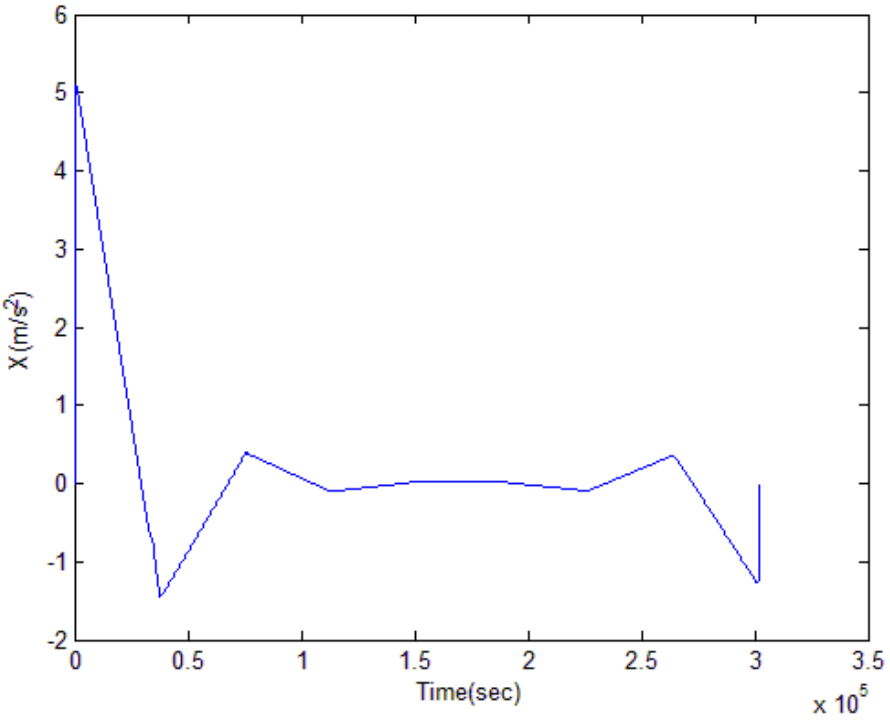

Figure 66. Source Term Histories for *X* - Cubic Splines Using 8 Spline Zones

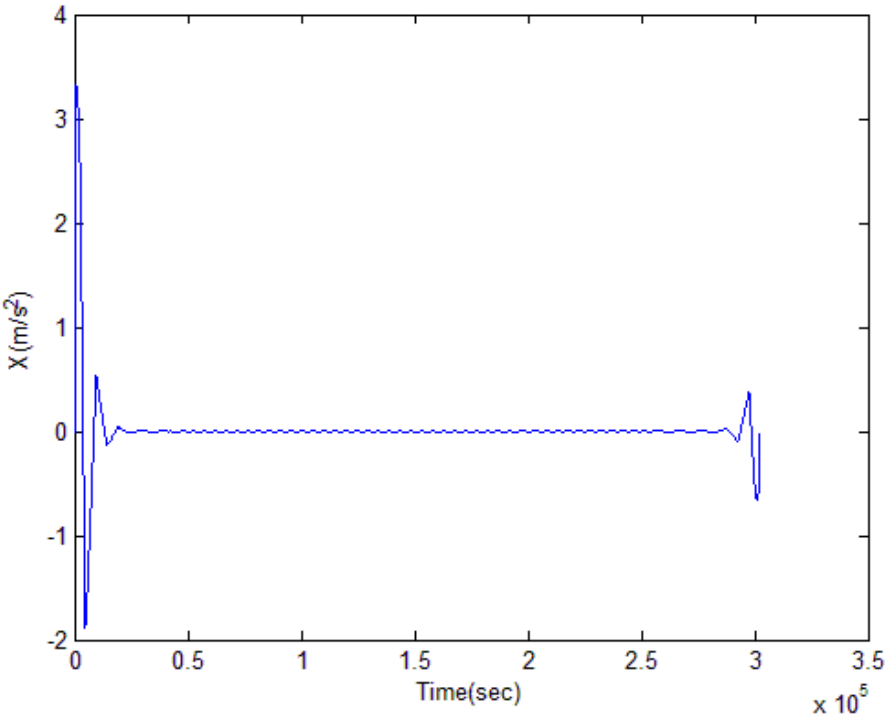

Figure 67. Source Term Histories for *X* - Cubic Splines Using 63 Spline Zones

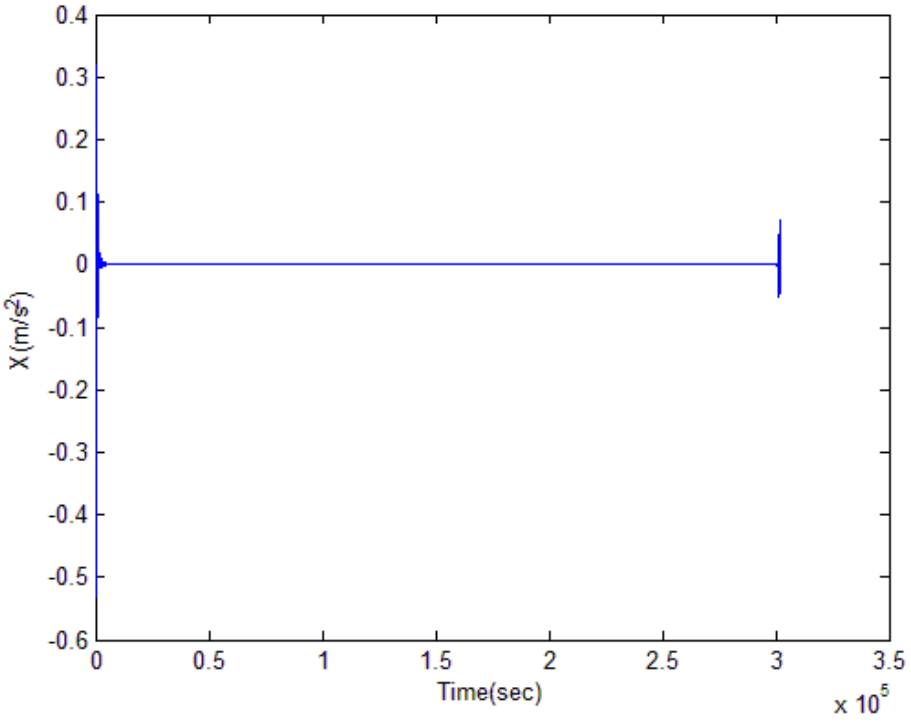

Figure 68. Source Term Histories for *X* - Cubic Splines Using 504 Spline Zones

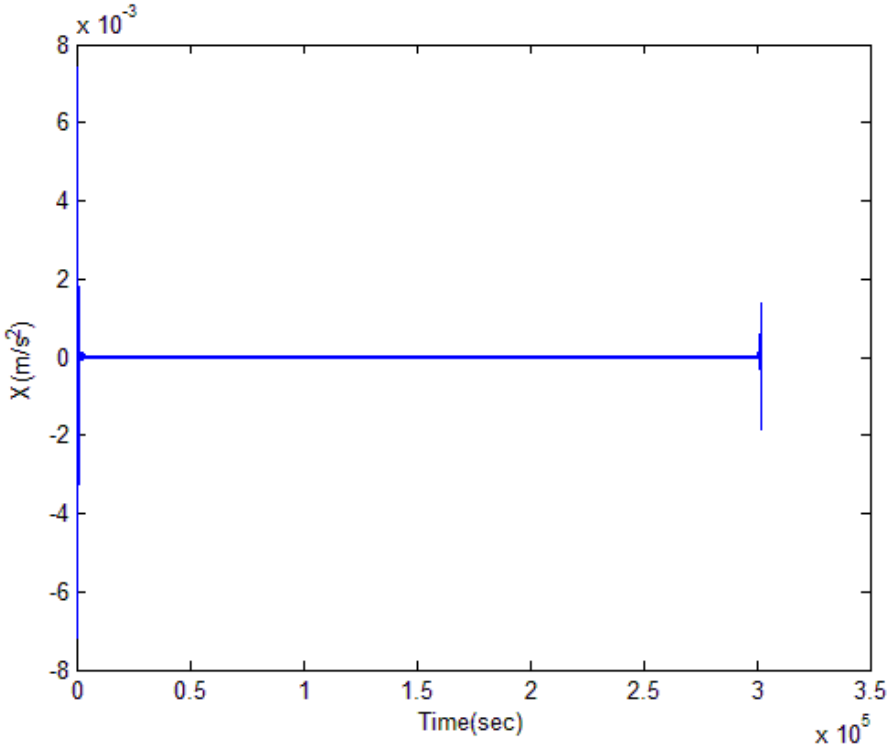

Figure 69. Source Term Histories for *X* - Cubic Splines Using 5040 Spline Zones

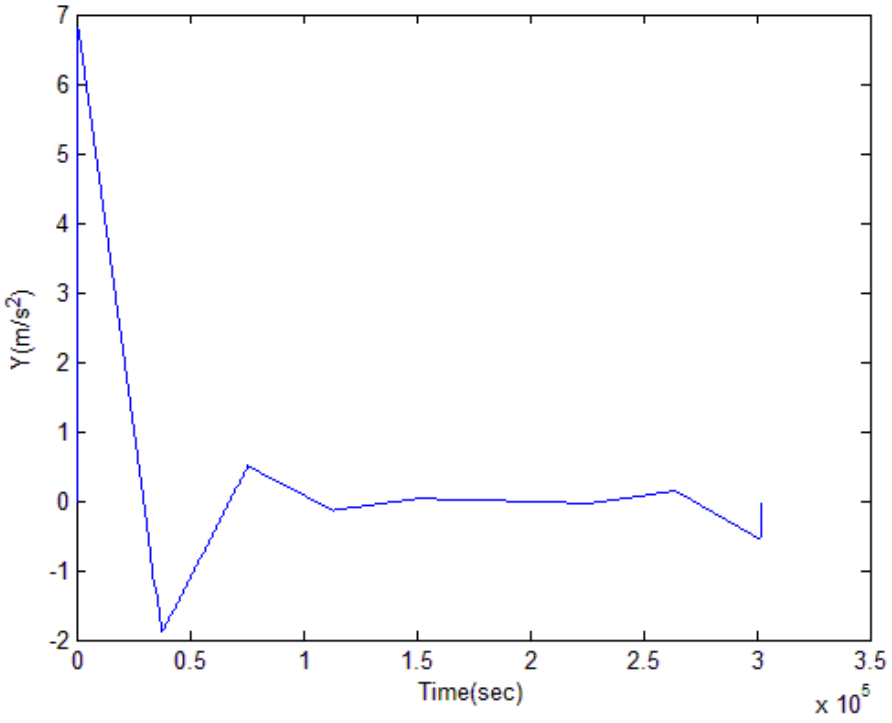

Figure 70. Source Term Histories for *Y* - Cubic Splines Using 8 Spline Zones

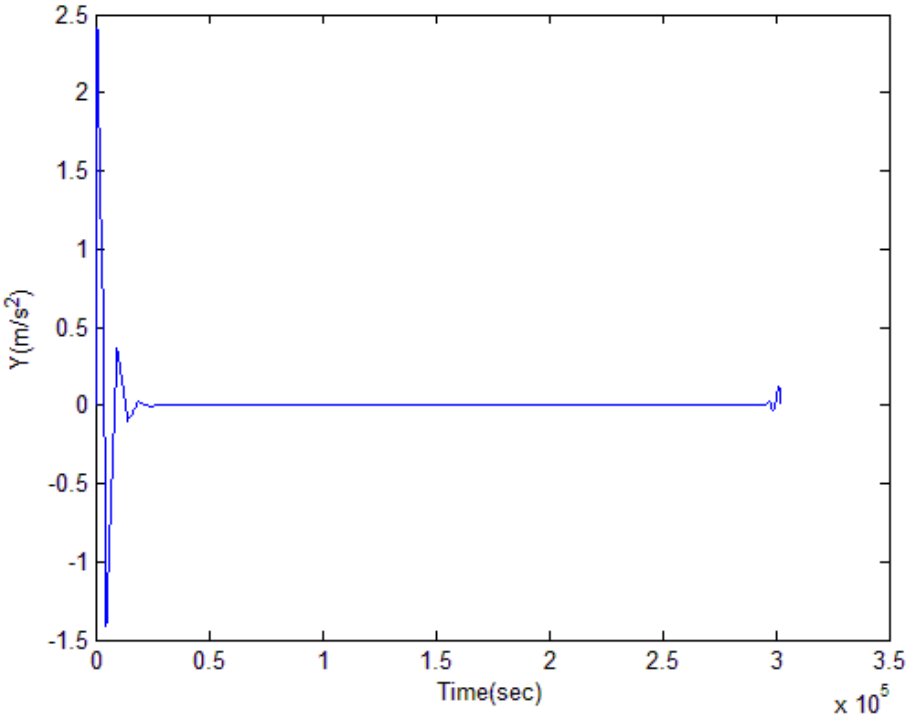

Figure 71. Source Term Histories for *Y* - Cubic Splines Using 63 Spline Zones

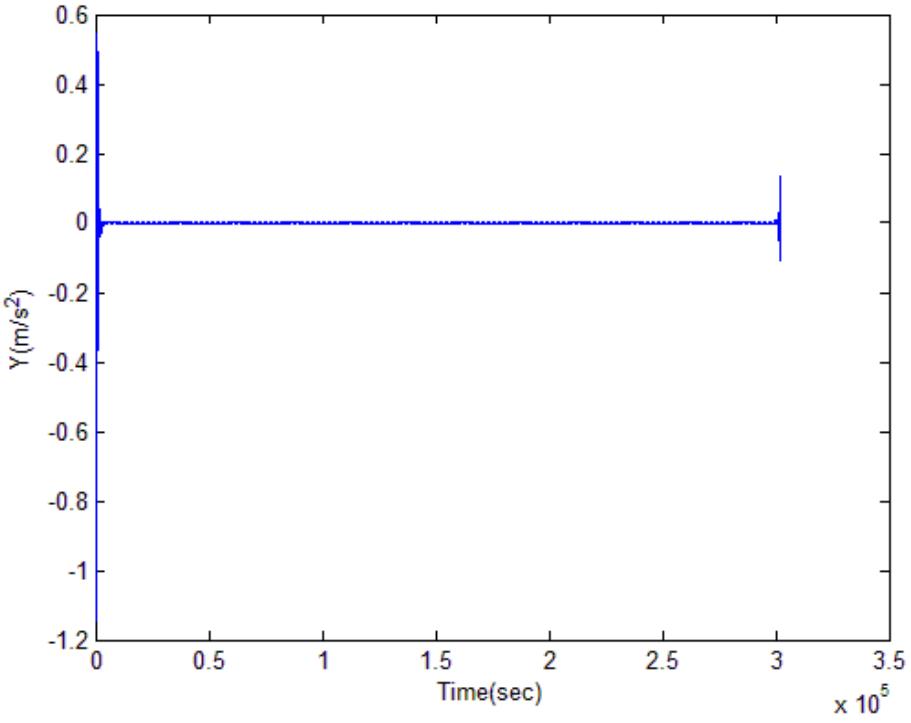

Figure 72. Source Term Histories for *Y* - Cubic Splines Using 504 Spline Zones

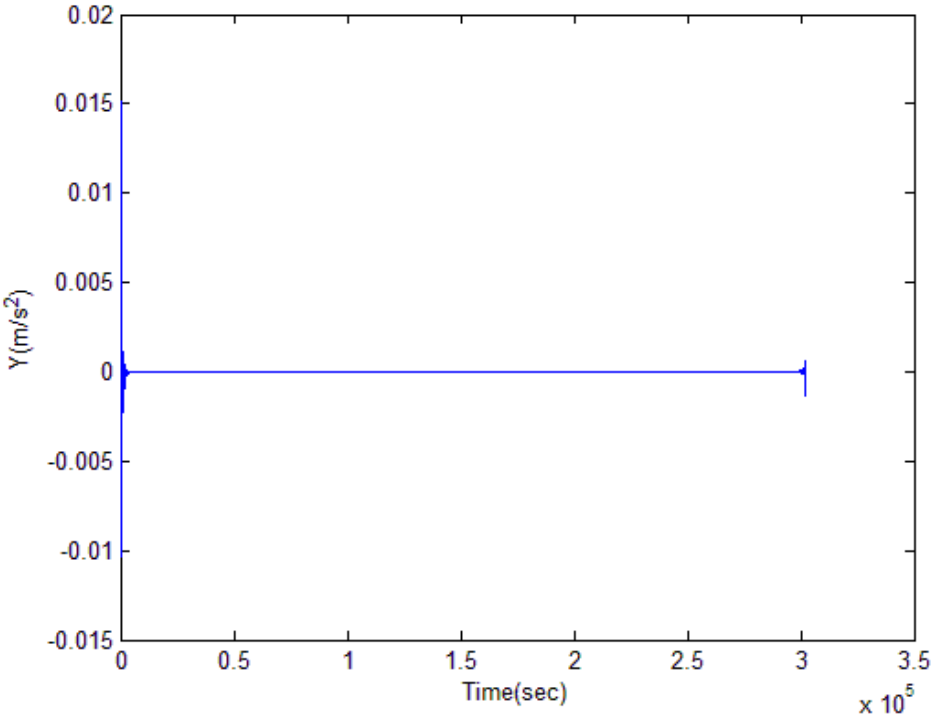

Figure 73. Source Term Histories for *Y* - Cubic Splines Using 5040 Spline Zones

Analytical Source Terms Using Fifth-degree Hermite Splines

Following figures show the source-term histories for both X and Y for the fifth-degree Hermite splines curve fits using varying numbers of spline zones. It can be seen that the magnitude of the source terms decreases as the number of spline zones increases. Using moderate number of spline zones, the magnitude of the source terms generated is extremely small. Since the fifth-degree Hermite splines are C^3 continuous, the source terms are found to be slope continuous; thus satisfying the continuity criterion required by MNP. Extremely small magnitude of source terms indicates that the nearby problem constructed using these source terms can be considered to be a good representation of our original problem. Therefore, the fifth-degree Hermite splines are a feasible option for constructing the nearby problem. Because small source terms were achieved using a reasonable number of fixed-resolution spline zones, for simplicity this approach was used instead of the multi-resolution approach. Construction of the nearby problem using these fifth-degree Hermite splines is discussed in chapter 6.

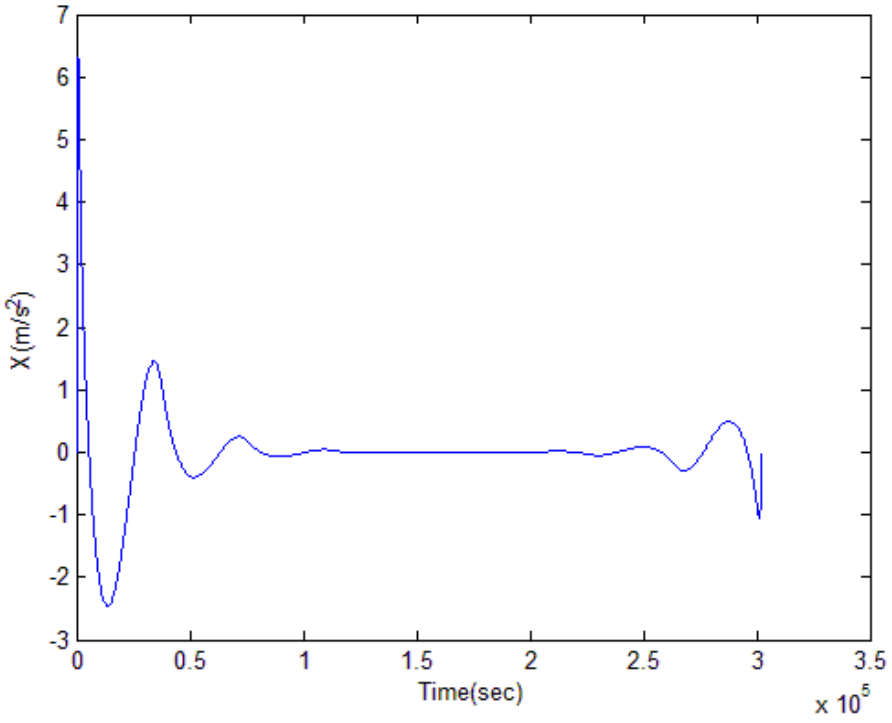

Figure 74. Source Term Histories for *X* – Fifth-degree Hermite Splines Using 8 Spline Zones

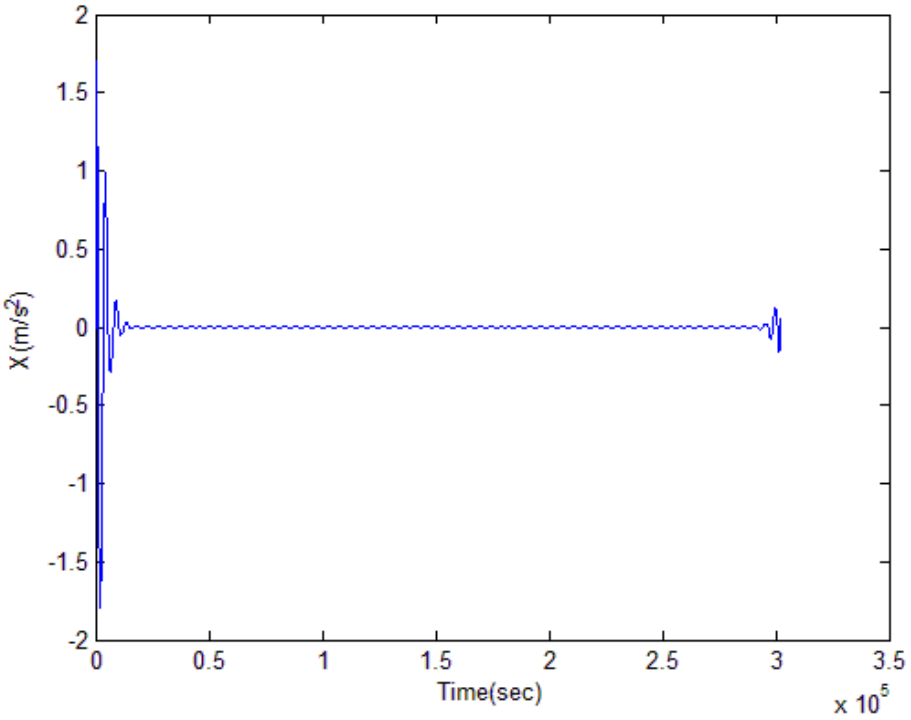

Figure 75. Source Term Histories for *X* – Fifth-degree Hermite Splines Using 63 Spline Zones

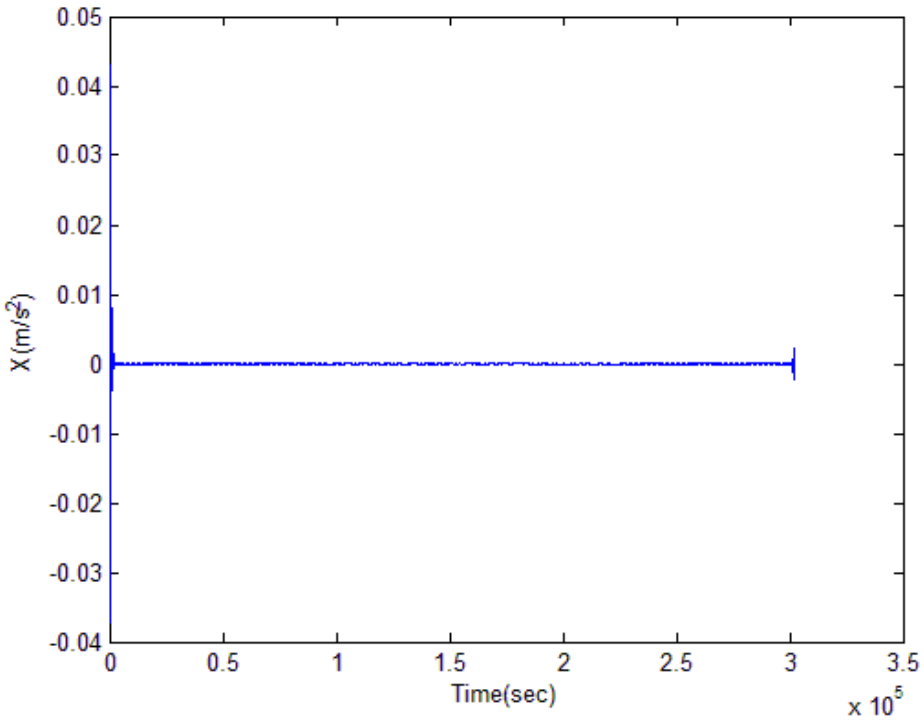

Figure 76. Source Term Histories for *X* – Fifth-degree Hermite Splines Using 504 Spline Zones

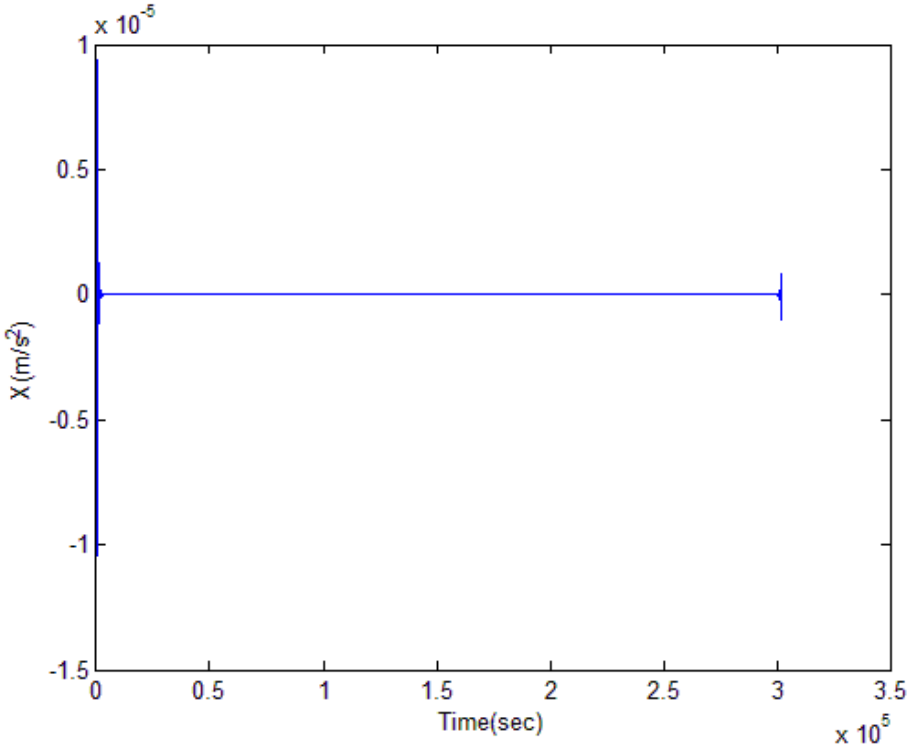

Figure 77. Source Term Histories for *X* – Fifth-degree Hermite Splines Using 5040 Spline Zones

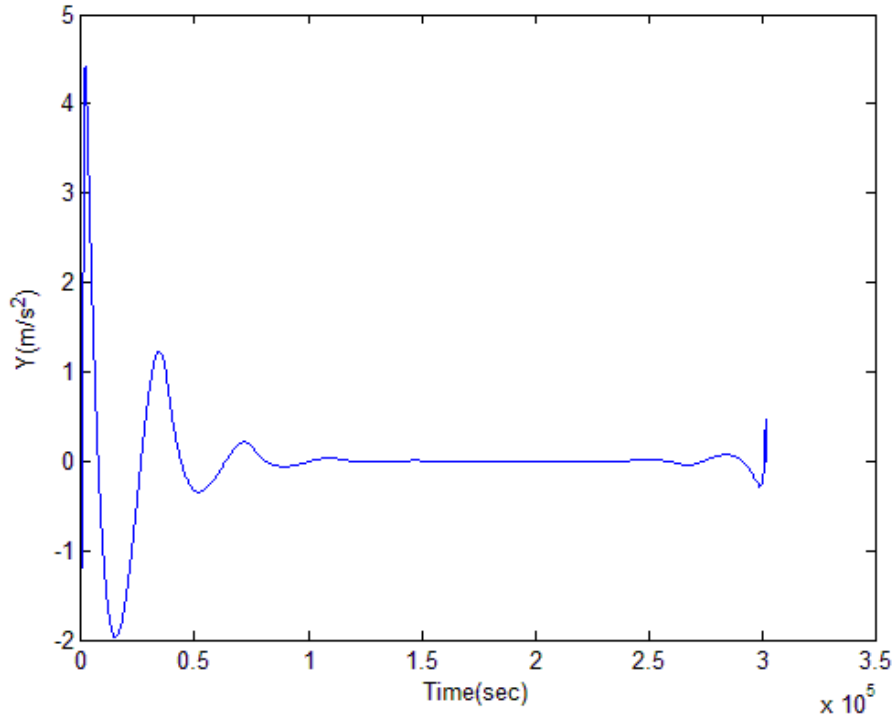

Figure 78. Source Term Histories for *Y* – Fifth-degree Hermite Splines Using 8 Spline Zones

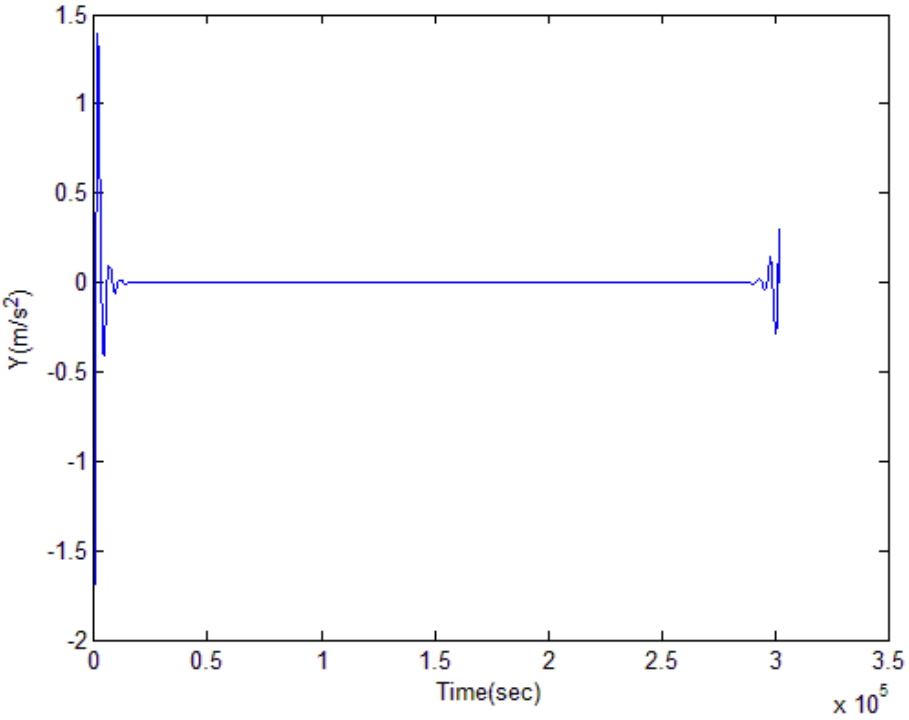

Figure 79. Source Term Histories for *Y* – Fifth-degree Hermite Splines Using 63 Spline Zones

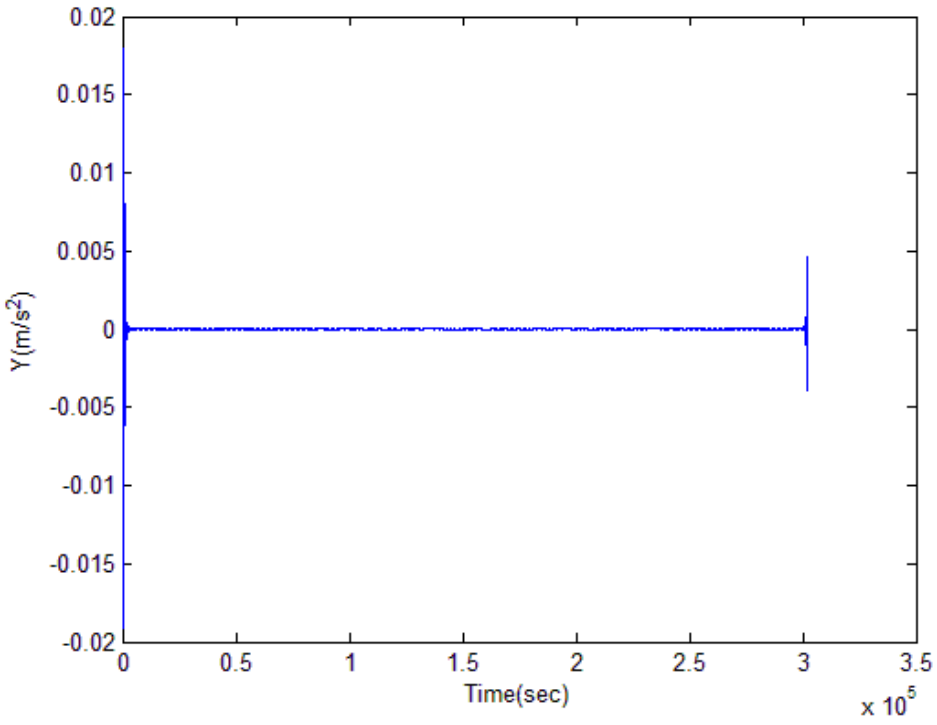

Figure 80. Source Term Histories for *Y* – Fifth-degree Hermite Splines Using 504 Spline Zones

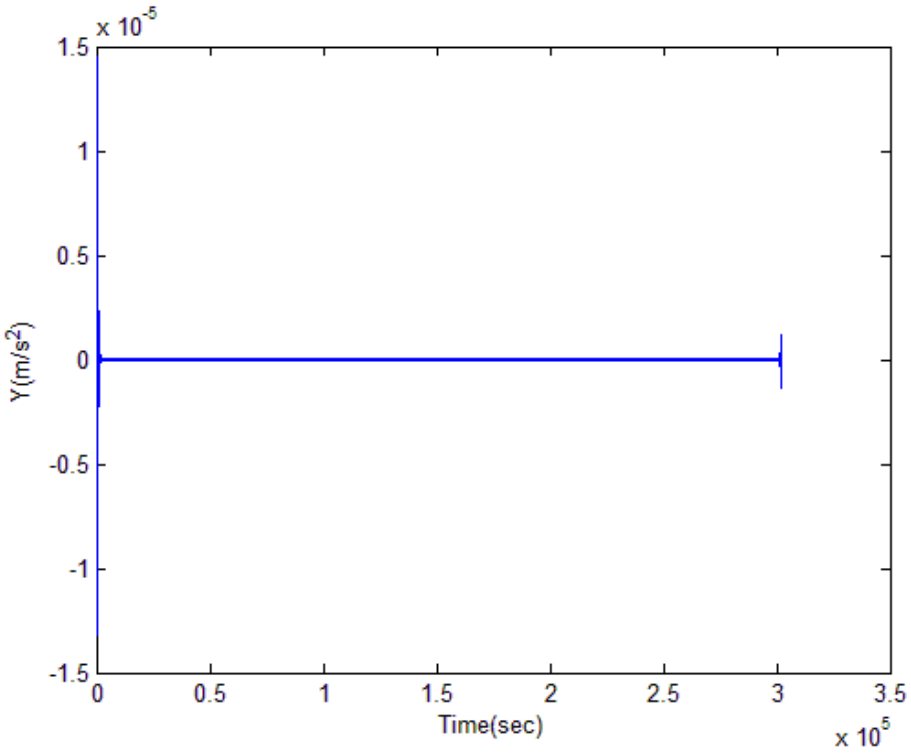

Figure 81. Source Term Histories for *Y* – Fifth-degree Hermite Splines Using 5040 Spline Zones

CHAPTER 6

NEARBY PROBLEM TO THE ORIGINAL PROBLEM

A nearby problem to our original problem of Earth-spacecraft-Moon three-body dynamics is constructed by adding the source terms obtained from the curve fit using the Fifth-degree Hermite splines to the governing equations of the original problem as shown in Eq. (43). As discussed earlier, the curve fit serves as an exact solution to this nearby problem. In order to be able to calculate the error in the numerical solution to the nearby problem, it is solved numerically using the same numerical scheme that was used for solving the original problem (appendix H). Note that solving the nearby problem using RK4 with a time step of 20 seconds requires evaluating the source terms every 10 seconds.

Following figures show the construction of various nearby problems using Fifth-degree Hermite splines with varying numbers of spline zones. It is seen that as the number of spline zones increases, both the exact and the numerical solution to the nearby problem start nearing the solution of the original problem.

The nearness of the nearby problem constructed using the Fifth-degree Hermite splines with moderate number of spline zones has already been established through previous chapters. Therefore, this nearby problem can be considered to be a good representation of the original problem of interest. Thus, the exact numerical error in the nearby problem can be used as an estimate of the numerical error in the original problem.

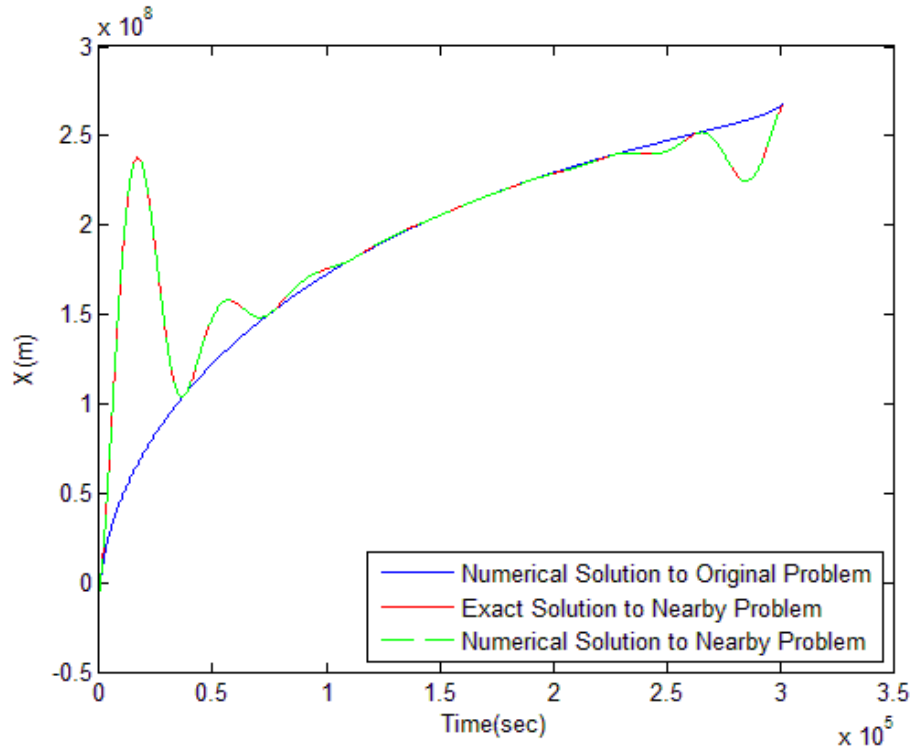

Figure 82. Nearby Problem to *X* - Using Fifth-degree Hermite Splines with 8 Spline Zones

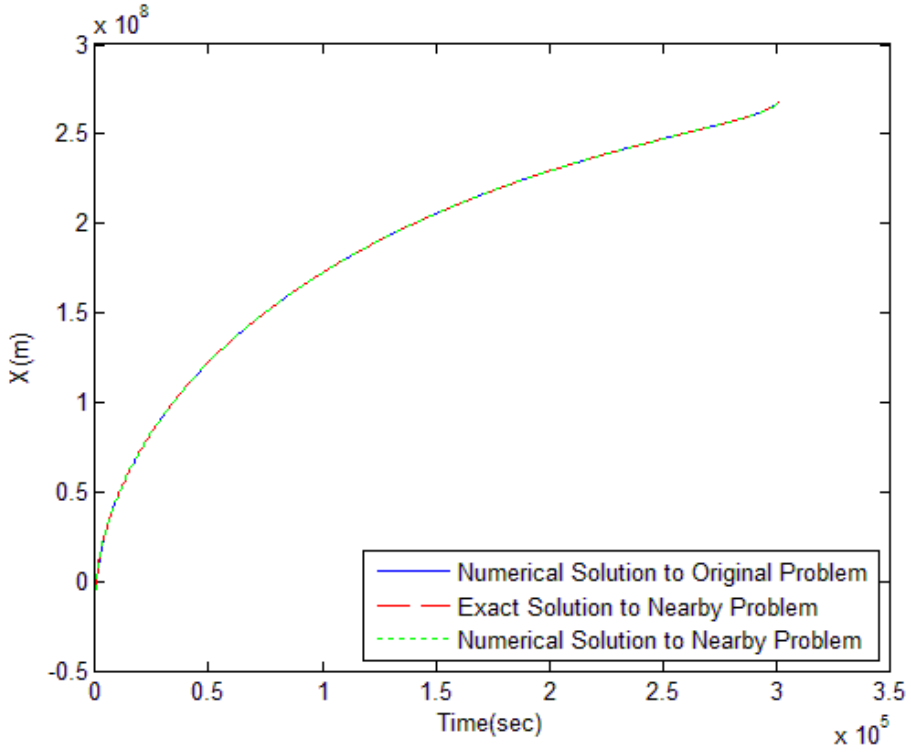

Figure 83. Nearby Problem to *X* - Using Fifth-degree Hermite Splines with 63 Spline Zones

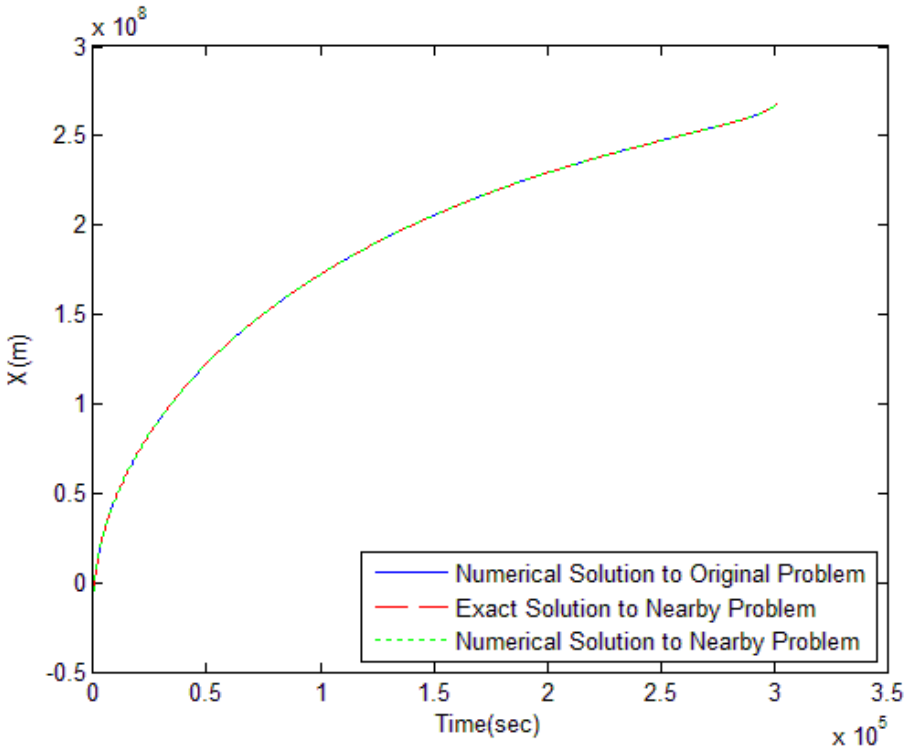

Figure 84. Nearby Problem to *X* - Using Fifth-degree Hermite Splines with 504 Spline Zones

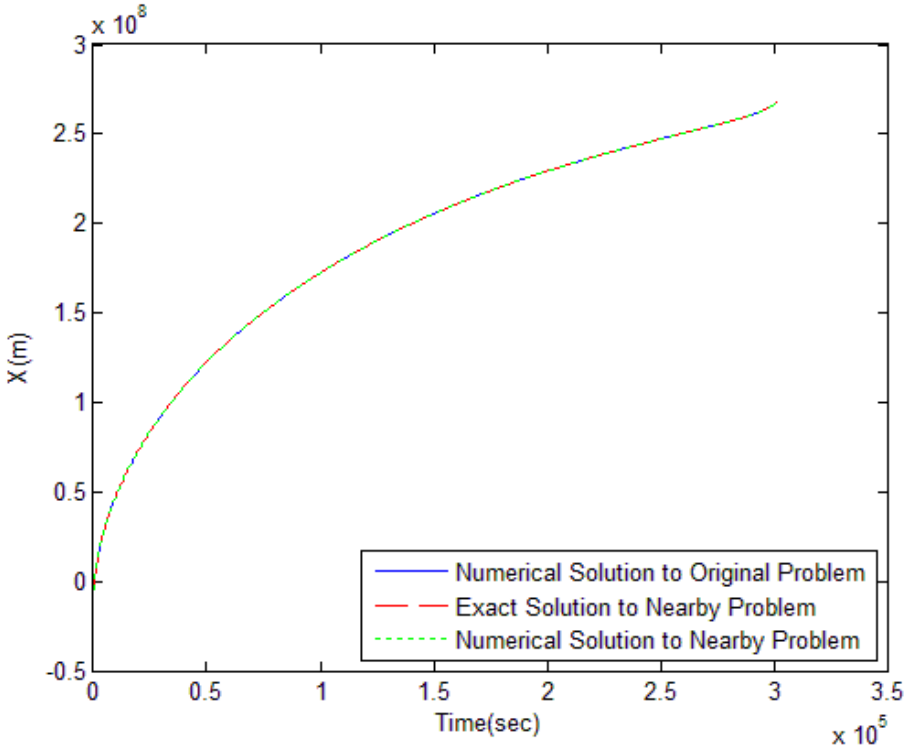

Figure 85. Nearby Problem to *X* - Using Fifth-degree Hermite Splines with 5040 Spline Zones

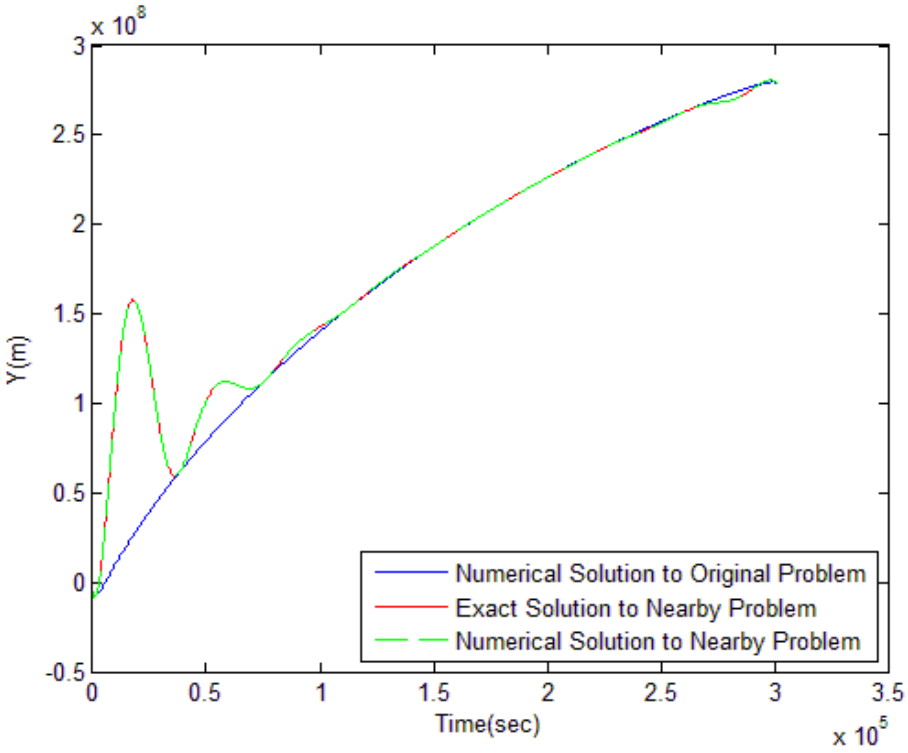

Figure 86. Nearby Problem to *Y* - Using Fifth-degree Hermite Splines with 8 Spline Zones

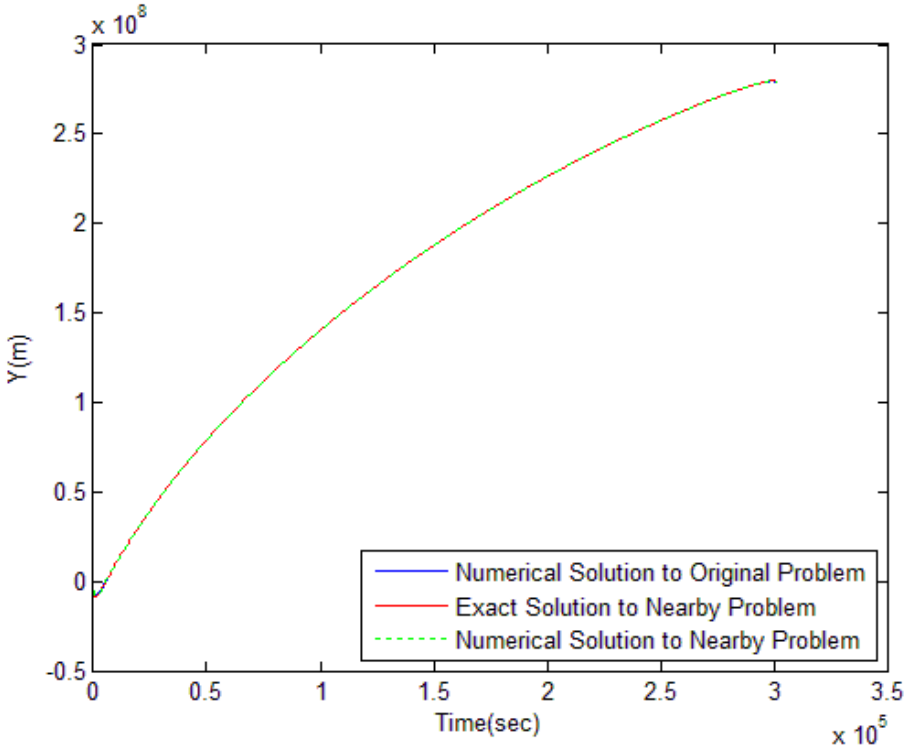

Figure 87. Nearby Problem to *Y* - Using Fifth-degree Hermite Splines with 63 Spline Zones

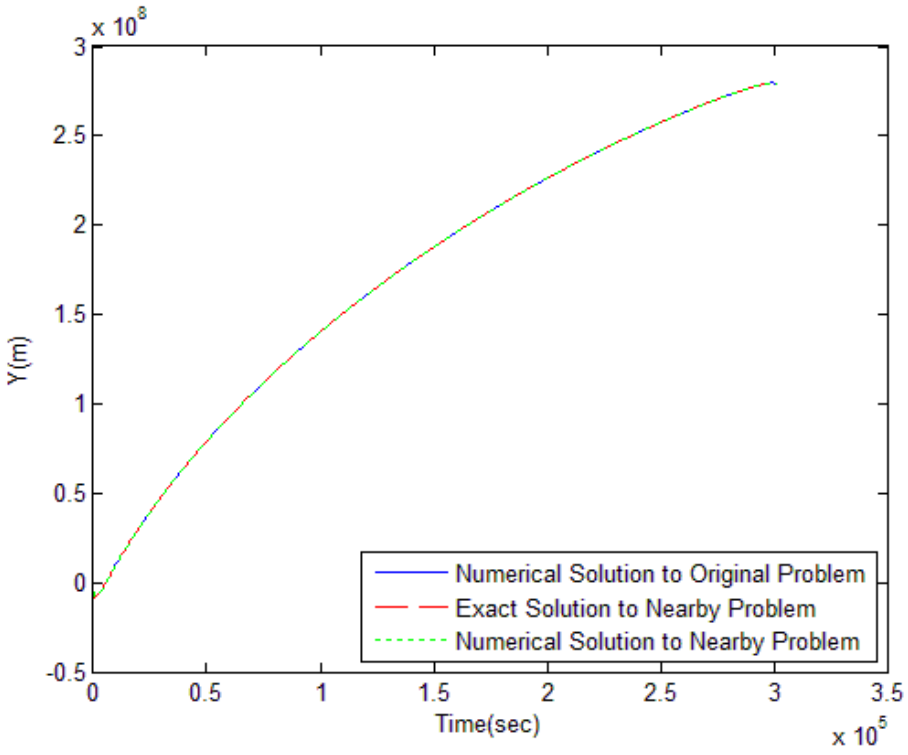

Figure 88. Nearby Problem to *Y* - Using Fifth-degree Hermite Splines with 504 Spline Zones

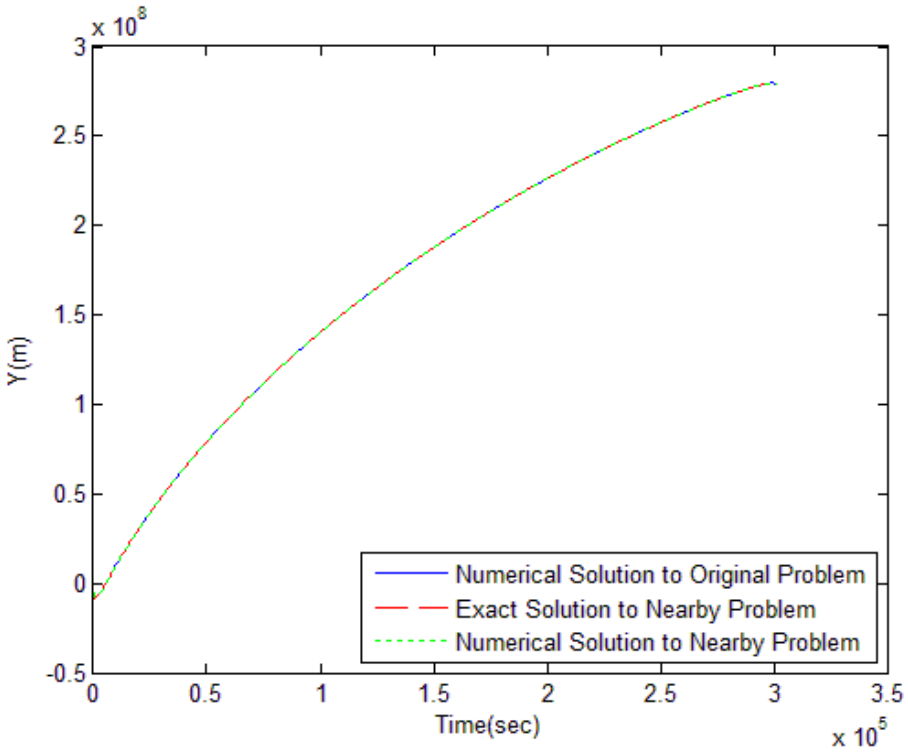

Figure 89. Nearby Problem to *Y* - Using Fifth-degree Hermite Splines with 5040 Spline Zones

Figures 90 and 91 show, for *X* and *Y*, the exact error in numerical solutions to the various nearby problems discussed above. It is seen that using varying numbers of spline zones the exact solution to the nearby problem differs, but the error in the numerical solution to the nearby problem remains fairly consistent. In general, the error tends to grow as the numerical solution is propagated forward in time. In particular, the error tends to change quickly as the spacecraft reaches the Moon in the terminal portion of the trajectory. From the combined results for *X* and *Y*, the accumulated error is approximately of the order of 10 meters.

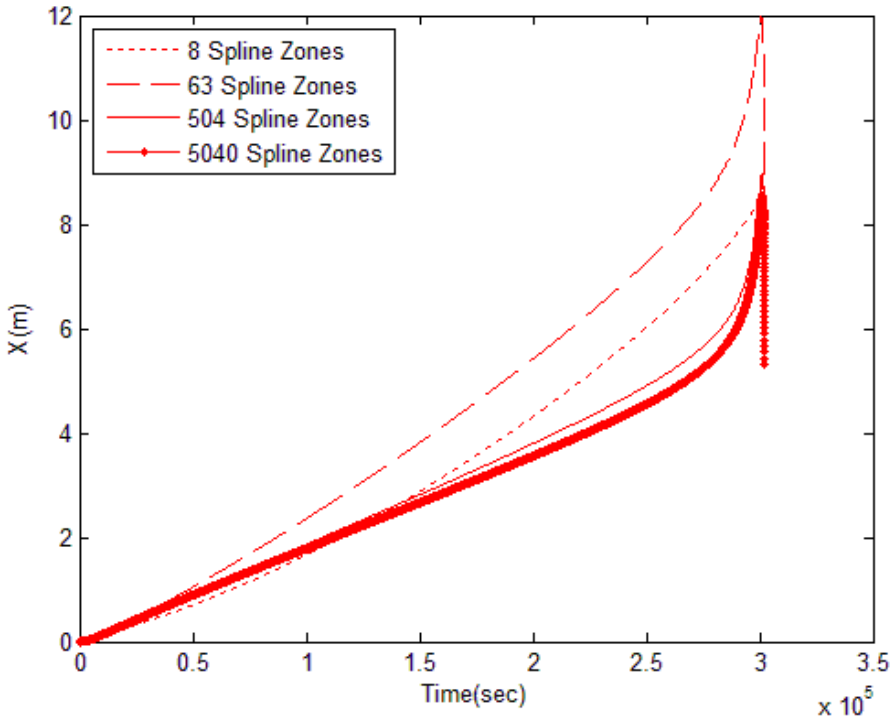

Figure 90. Exact Error in Nearby Problem to *X* Using Fifth-degree Hermite Splines

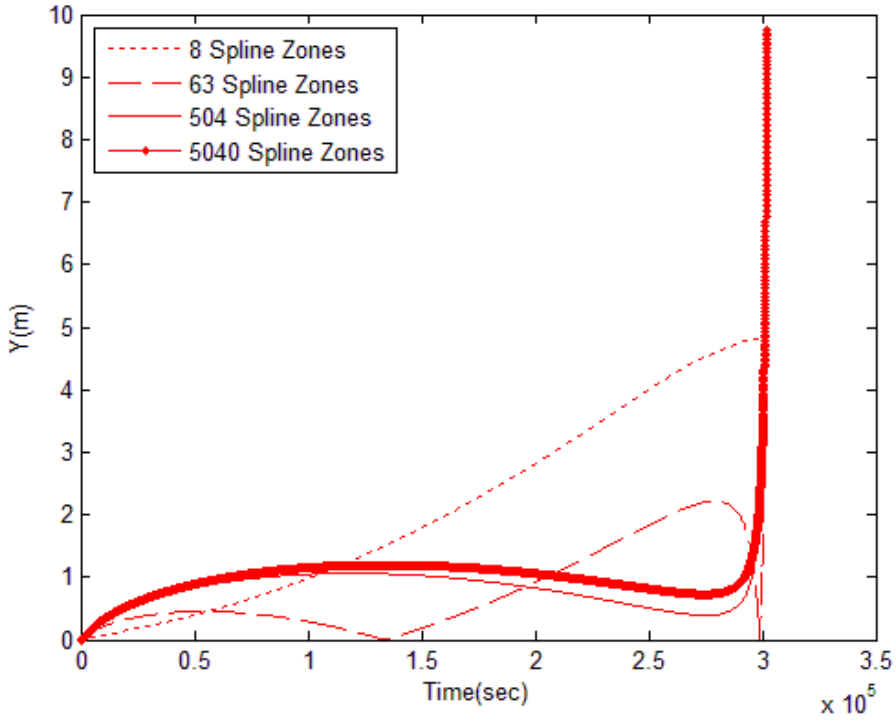

Figure 91. Exact Error in Nearby Problem to *Y* Using Fifth-degree Hermite Splines

It is imperative to examine the reliability of these error estimates. This could be done by comparing these estimates to the estimates given by some other methods. Checking for the reliability of the numerical scheme used is also a good idea.

One of the best ways to verify the numerical scheme is to calculate the observed order of accuracy and see how well it matches the formal order of accuracy. The RK4 numerical integration scheme is a 4^{th} order accurate method i.e. the formal order of accuracy of RK4 method is four. The observed order of accuracy, p, is computed for the various numerical solutions of the nearby problem (constructed using fifth-degree Hermite splines with 5040 spline zones) using different meshes. A mesh is indicated by the step size used. Thus, the fine mesh uses a smaller step size than the coarse mesh. Since the exact solution is known, the observed order of accuracy can be computed by the following relation [19]:

$$p = \frac{ln\left(\frac{E2}{E1}\right)}{ln(r)} \tag{44}$$

In Eq. (44), r is the grid refinement factor (the ratio between the coarse mesh and the fine mesh). One of the tests used by Berry and Healy [3] to verify the accuracy of the numerical integrators is the step-size halving test. For the step-size halving test, the reference integration is produced with the same integrator but with the step-size cut in half. On this basis, here, a value of $r=2$ is used. $E2$ is the L_2 norm of the errors between the exact solution and the numerical solution calculated at each instant in time using the coarse mesh while $E1$ is the L_2 norm of the errors between the exact solution and the numerical solution calculated at each instant in time using the fine mesh. Figure 92 shows the observed order of accuracies for various numerical solutions to the nearby problem (constructed using fifth-degree Hermite splines with 5040 spline zones) using various step sizes. It can be seen that as the step size is reduced the observed order of accuracy approaches four (which is the formal order of accuracy). This suggests that the numerical solution is reliable.

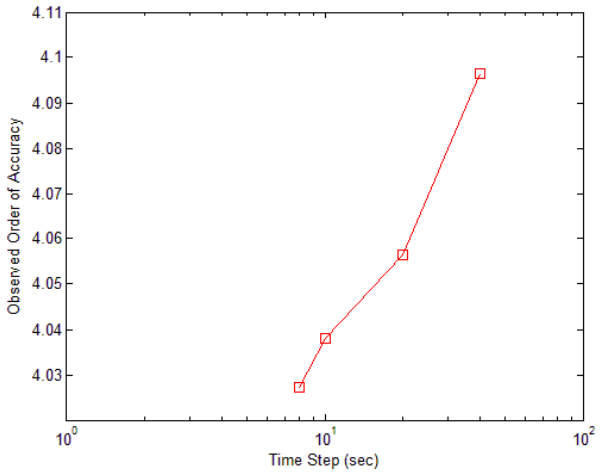

Figure 92. Observed Order of Accuracies for Various Numerical Solutions to the Nearby Problem using Different Step Sizes

Richardson extrapolation (RDE) can also be used to get an estimate of the exact solution. [20, 21] RDE involves computation of numerical solutions on two or more meshes. Solutions on these different meshes are then used to compute a higher-order estimate of the exact solution. This estimate of the exact solution can then be used to estimate the numerical error. For a *p*th order accurate scheme with solutions on a fine mesh (X_1) and a coarse mesh (X_2) with refinement factor *r*, X_{exact} can be approximated as: [19]

$$X_{exact} = X_1 + \frac{X_1 - X_2}{r^p - 1} \tag{45}$$

X_{exact} can be calculated using both global and formal order of accuracies. The error on the fine mesh can then be given as:

$$E = X_1 - X_{exact} \tag{46}$$

Figures 93 and 94 show the comparison of error estimates using MNP, RDE with formal order of accuracy, and RDE with observed order of accuracy. For MNP, a time step of 20 seconds is used and therefore for RDE the fine mesh uses a time step of 20 seconds while the coarse mesh uses a time step of 40 seconds. It can be seen that the error estimates using MNP and RDE match in the order of magnitude.

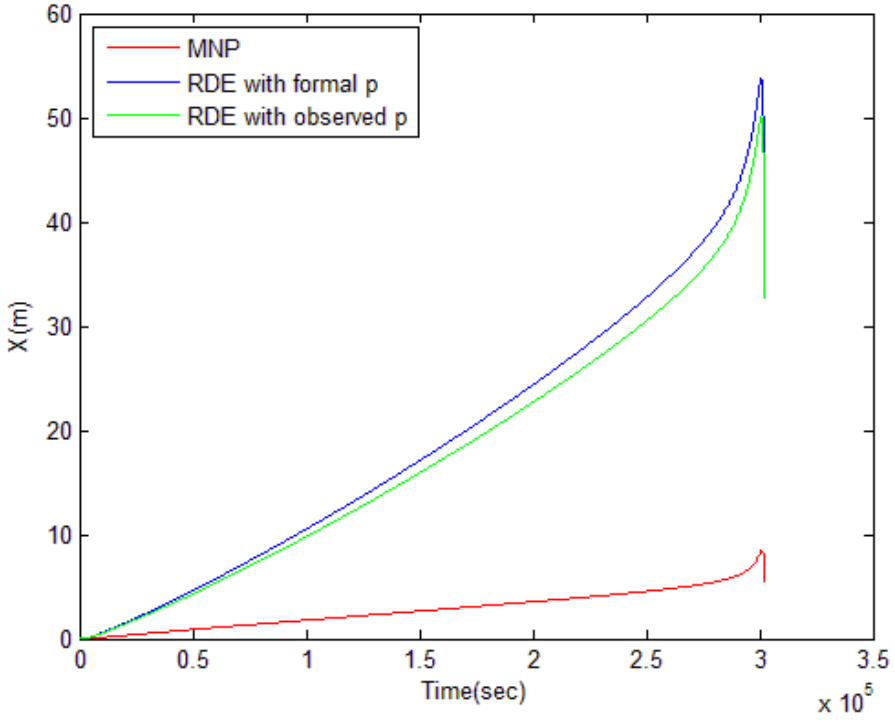

Figure 93. Numerical Error Estimates for *X* Using Various Methods

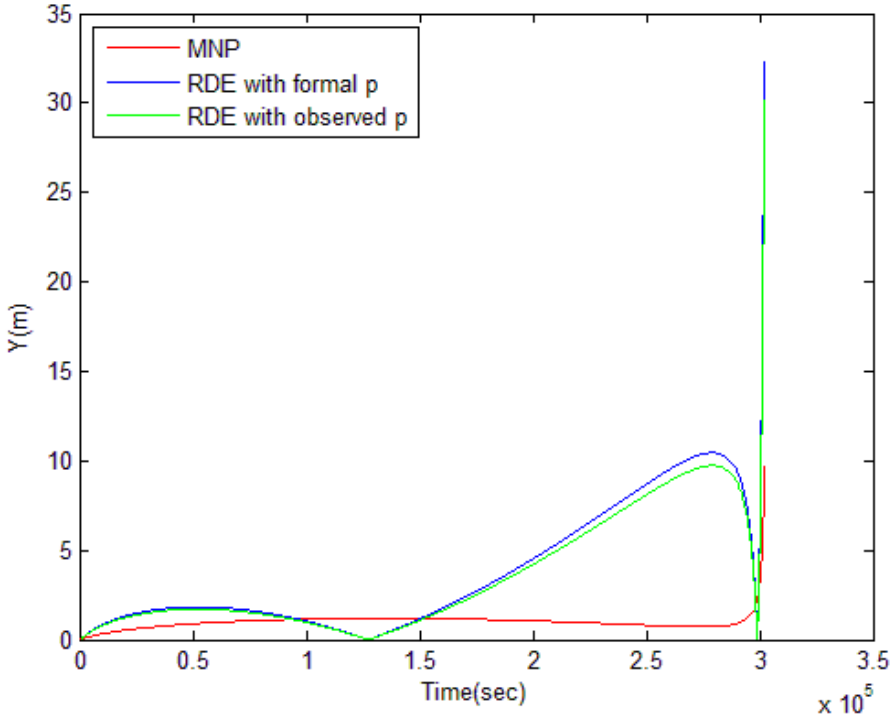

Figure 94. Numerical Error Estimates for *Y* Using Various Methods

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

In orbital mechanics, analytical solutions to many systems do not exist and thus, an accurate numerical solution is imperative to mission planning. One such system is the Earth-spacecraft-Moon three-body dynamics or the translunar trajectories. The translunar trajectories are chaotic in nature. They are extremely sensitive when the spacecraft is in proximity to the Earth and the Moon. So the numerical solution needs to be as accurate as possible otherwise a small error can get escalated later on. MNP is employed to validate the accuracy of this numerical solution by estimating the numerical error in it. A nearby problem (having an exact solution to it) to the Earth-spacecraft-Moon three-body problem is constructed. Fifth-degree Hermite splines are found to be a feasible option to construct this nearby problem as it satisfies all the conditions necessary to demonstrate its nearness to the original problem. This allows the exact error in the numerical solution to this nearby problem to be considered as a good estimation of the error in the numerical solution to the original problem.

The MNP estimate of the magnitude of the numerical error in the simulation of translunar trajectories is of the order of 10 meters. This accuracy may be sufficient for many aspects of mission planning; however, for critical mission phases higher accuracy may be desired. Also, this error is accumulated when the propagation time is 3.5 days. The propagation time for other missions could range from a few days to even years.

During such long propagation times, substantial amount of error can be accumulated. This could lead to inaccurate results. Of course, accurate simulation also depends on accurate modeling of the system dynamics. Mission planning requires both the development of accurate models and the accurate numerical solution of those models.

This thesis demonstrates the usefulness of MNP in providing reliable estimates of the error in the numerical solutions to the problems in orbital mechanics. These error estimates, however, are dependent on the numerical scheme used and the type of problem studied. The reliability of MNP can further be verified by using various numerical integration schemes and/or by studying problems involving forces of a more complex nature.

REFERENCES

1. Huang, T.-Y. and Innanen, K. A., "The accuracy check in numerical integration of dynamical systems," Astronomical Journal, Vol. 88, No. 6, 1983, pp. 870–876, June 1983.

2. Fox, K., "Numerical integration of the equations of motion of celestial mechanics," *Celestial Mechanics*, Vol. 33, No. 2, 1984, pp. 127-142, June 1984.

3. M. Berry, L. Healy, "Comparison of accuracy assessment techniques for numerical integration," 13th AAS/AIAA Space Flight Mechanics Meeting, Ponce, Puerto Rico, 9-13 February 2003.

4. Montenbruck, O., "Numerical Integration Methods for Orbital Motion," Celestial Mechanics and Dynamical Astronomy, Vol. 53, pp. 59–69, 1992.

5. Hadjifotinou, K. G. and Gousidou-Koutita, M., "Comparison of Numerical Methods for the Integration of Natural Satellite Systems," Celestial Mechanics and Dynamical Astronomy, Vol. 70, No. 2, 1998, pp. 99–113, 1998.

6. Roache, P.J., "Code Verification by the Method of Manufactured Solutions," *ASME Journal of Fluids Engineering*, Vol. 124, March 2002, pp. 4-10.

7. Zadunaisky, P. E., "A Method for the Estimation of Errors Propagated in the Numerical Solution of a System of Ordinary Differential Equations," In Contopoulos,

G., editor, *The Theory of Orbits in the Solar System and in Stellar Systems*, pp. 281–287, New York, 1966. International Astronomical Union, Academic Press.

8. Zadunaisky, P. E., “On the Estimation of Errors Propagated in the Numerical Integration of Ordinary Differential Equations,” Numerische Mathematik, Vol. 27, No. 1, 1976, pp. 21–39, 1976.

9. Zadunaisky, P. E., “On the Accuracy in the Numerical Solution of the N-Body Problem,” Celestial Mechanics, Vol. 20, pp. 209–230, 1979.

10. Zadunaisky, P. E., “On the Accuracy in the Numerical Computation of Orbits,” In Giacaglia, G.E. O., editor, Periodic Orbits, Stability and Resonances, pp. 216–227, Dordrecht, Holland, 1970.D. Reidel Publishing Company.

11. Junkins, J.L., and Lee. S., “Validation of Finite-Dimensional Approximate Solutions for Dynamics of Distributed-Parameter Systems,” *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 87-95.

12. M.M. Hopkins and C.J. Roy, "Introducing the Method of Nearby Problems." *European Congress on Computational Methods in Applied Sciences and Engineering,* ECCOMAS 2004, P. Neittaanmaki, T. Rossi, S. Korotov, E. Onate, J. Periaux, and D. Knorzer (eds.), July 24-28, 2004.

13. Roy, C. J., and Hopkins, M. M., “Discretization Error Estimates using Exact Solutions to Nearby Problems,” AIAA Paper 2003-0629, January 2003

14. Bate R., Mueller D., White J., Fundamentals of Astrodynamics, Dover Publications, New York, 1971.

15. V. Chobotov, "Orbital Mechanics," *AIAA Education Series*, Virginia 2002.

16. M.P. Vautier and A.J. Sinclair, “Coordinate Switching for Accurate Simulation of Chaotic Trajectories”, *AAS 08-271, F. Landis Markley Astronautics Symposium, Cambridge, Maryland,* 29 June – 2 July 2008.

17. C.B. Moler, "Numerical Computing with MATLAB." *SIAM*, Philadelphia 2004.

18. G.E. Mulleges and F. Uhlig, "Numerical Algorithms with Fortran," *Springer-Verlag*, Berlin 1996

19. Roy, C. J., “Review of Code and Solution Verification Procedures for Computational Simulation” *Journal of Computational Physics*, Vol. 205, 2005, pp. 131-156

20. L.F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations with an application to the stresses in a masonry dam, Trans. Royal Society London, Ser. A 210 (1910) 307–357.

21. L.F. Richardson, The deferred approach to the limit, Trans. Royal Society London, Ser. A 226 (1927) 229–361.

APPENDICES

A. MATLAB CODE TO NUMERICALLY SOLVE THE ORIGINAL PROBLEM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%     NUMRICAL SOLUTION TO THE ORIGINAL PROBLEM     %%%%%%%%%
%%%%%%%%%%%           PROPOGATION IN THE EC->MC FRAME         %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;
format long;
format compact;

%%%%%%%%%%%%%%%%%%%%%%%%% SETTING UP THE PROBLEM %%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% USER INPUTS %%%%%
alt = 359750; %meters
angle = -36.890; %degrees
dt = 20; %secs
fprintf('alt = %6.0f, ang = %6.3f, dt = %6.3f \n\n', alt, angle, dt);
deltav = 3102.13; %m/s
t0 = 0; %secs
tf = 3.5*86400; %302400 secs (1 day = 86400 secs)
SP = 2.905*86400; % Switch Point at 250992 secs
folder = char(['alt' int2str(alt) 'ang' int2str(abs(angle*1000)) 'dt'
int2str(dt)]);
fid = fopen([folder 'State_SP' '.txt'], 'w');
f = fopen([folder 'Moon_State_SP' '.txt'], 'w');

%%%%% CONSTANTS %%%%%
GMm = 4.90266e12; %m3/s2
GMe = 3.98600436e14; %m3/s2
radE = 6372797; %meters
EMdist = 384400000; %meters
omega = sqrt((GMe+GMm)/EMdist^3); %rad/sec
%Tm = 2*pi/omega = 27d 6h 49m 50.34879957310977s
alpha = angle*pi/180; %radians
h = radE + alt; %meters
Vorbit = sqrt(GMe/h); %m/s
V = Vorbit + deltav; %m/s

%%%%% INITIAL CONDITIONS %%%%%
%%%%% For the Spacecraft %%%%%
rxi = h*sin(alpha); %meters
```

```
ryi = -h*cos(alpha); %meters
vxi = V*cos(alpha); %m/s
vyi = V*sin(alpha); %m/s
x = [rxi;ryi;vxi;vyi];
fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16e nn', t0, x);
%%%%% For the Moon %%%%%
xmt = EMdist*cos(omega*t0);
ymt = EMdist*sin(omega*t0);
fprintf(f, '% 6.0f, % .16f, % .16f nn', t0, xmt, ymt);

%%%%%%%%%%%%%%%%%%%%%%%% PERFORM THE PROPOGATION %%%%%%%%%%%%%%%%%%%%%%

%%%%% EARTH CENTERED FRAME %%%%%
i = 1;
for t = t0:dt:SP-dt
    %Use the moon position at the end of the previous interval as the
    %position at the start of this interval.
    xm = xmt;
    ym = ymt;
    %Calculate the moon position at the middle and end of this
    %interval.
    xmh = EMdist*cos(omega*(t + 0.5*dt));
    ymh = EMdist*sin(omega*(t + 0.5*dt));
    xmt = EMdist*cos(omega*(t + dt));
    ymt = EMdist*sin(omega*(t + dt));
    t1 = t+dt;
    fprintf(f, '% 6.0f, % .16f, % .16f nn', t1, xmt, ymt);
    k1 = RK4_EC(x, xm, ym, GMm, GMe);
    k2 = RK4_EC(x+(dt/2)*k1', xmh, ymh, GMm, GMe);
    k3 = RK4_EC(x+(dt/2)*k2', xmh, ymh, GMm, GMe);
    k4 = RK4_EC(x+(dt)*k3', xmt, ymt, GMm, GMe);
    x = x + dt/6*(k1'+2*k2'+2*k3'+k4');
    fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16enn', t1, x);
    i = i+1;
end

%%%%% SWITCH POINT %%%%%
%Convert state vector from EC to MC
rho = [xmt; ymt];
xmt_dot = -EMdist*omega*sin(omega*(t+dt));
ymt_dot = EMdist*omega*cos(omega*(t+dt));
rho_dot = [xmt_dot; ymt_dot];
d_state = x - [rho; rho_dot];

%%%%% MOON CENTERED FRAME %%%%%
j = 1;
for t = SP:dt:tf-dt
    %Use the moon position at the end of the previous interval as the
    %position at the start of this interval.
    xm = xmt;
    ym = ymt;
    %Calculate the moon position at the middle and end of this
    %interval.
    xmh = EMdist*cos(omega*(t + 0.5*dt));
    ymh = EMdist*sin(omega*(t + 0.5*dt));
    xmt = EMdist*cos(omega*(t + dt));
```

```
    ymt = EMdist*sin(omega*(t + dt));
    t2 = t+dt;
    fprintf(f, '% 6.0f, % .16f, % .16f nn', t2, xmt, ymt);
    %Propogating the states forward in time
    k1 = RK4_MC(d_state, xm, ym, GMm, GMe);
    k2 = RK4_MC(d_state+(dt/2)*k1', xmh, ymh, GMm, GMe);
    k3 = RK4_MC(d_state+(dt/2)*k2', xmh, ymh, GMm, GMe);
    k4 = RK4_MC(d_state+(dt)*k3', xmt, ymt, GMm, GMe);
    d_state = d_state + dt/6*(k1'+2*k2'+2*k3'+k4');
    %Compute the moon state at the end of interval
    rho = [xmt; ymt];
    rho_dot = EMdist*omega*[-sin(omega*(t + dt)); cos(omega*(t + dt))];
    %Convert state vector from MC to EC
    x = d_state + [rho; rho_dot];
    fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16enn',t2,x);
    j = j+1;
end
fclose(fid);
fclose(f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B. SUBROUTINES USED IN THE CODE GIVEN IN APPENDIX A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%              SUBROUTINE TO INTEGRATE                  %%%%%%%%%%
%%%%%%%%%%         THE EQUATION OF MOTION IN EC FRAME            %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 function x_dot  = RK4_EC(x,xmoon,ymoon,GMm,GMe)

          r = x(1:2);
          v = x(3:4);
          rho = [xmoon; ymoon];
          d = r-rho;
          x_dot(1:2) = v;
          x_dot(3:4) = -GMe*r/norm(r)^3-
                       GMm*(d/norm(d)^3+rho/norm(rho)^3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%              SUBROUTINE TO INTEGRATE                  %%%%%%%%%%
%%%%%%%%%%         THE EQUATION OF MOTION IN MC FRAME            %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d_dot = RK4_MC(d_state,xmoon,ymoon,GMm,GMe)

         d = d_state(1:2);
         v = d_state(3:4);
         rho = [xmoon; ymoon];
         r = d+rho;
         d_dot(1:2) = v;
         d_dot(3:4) = -GMm*d/norm(d)^3-GMe*(r/norm(r)^3-
                      rho/norm(rho)^3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C. MATLAB CODE TO CALCULATE THE COEFFICIENTS OF CUBIC SPLINES CURVE FIT

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%  MATLAB CODE TO CALCULATE THE COEFFICIENTS  %%%%%%%%%%%
%%%%%%%%%%%%%%%%         OF CUBIC SPLINES CURVE FIT          %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [X_fit Xd_fit Xdd_fit] = fCurveFit_Cu(time10,time20,X,Xdd)

        % time10 => Time matrix with time step of 10 sec
        % time20 => Time matrix with time step of 20 sec
        % X => X position obtained from the simulation given in
               Appendix A
        % Xdd => X acceleration obtained from the simulation given in
                 Appendix A

        S = size(time20); m = S(2); j = 1;
        %Calculating Coefficients a
        for i = 1:3:m
            x(j) = time20(i);
            y(j) = X(i);
            a(j) = y(j);
            j = j+1;
        end
        S = size(x); n = S(2);
        %Calculating Coefficient c
        c = zeros(1,n); A = zeros(n-2);
        alpha = Xdd(1); beta = Xdd(m);
        c(1) = 0.5*alpha; c(n) = 0.5*beta;
        for i = 1:n-2
            h1 = x(i+1)-x(i);
            h2 = x(i+2)-x(i+1);
            A(i,i) = 2*(h1+h2);
        end
        for i = 2:n-2
            h2 = x(i+1)-x(i);
            A(i-1,i) = h2;
            A(i,i-1) = h2;
        end
        for i = 1:n-2
            h2 = x(i+2)-x(i+1); h1 = x(i+1)-x(i);
            a3 = a(i+2); a2 = a(i+1); a1 = a(i);
            if i == 1
               g(i) = (3/h2*(a3-a2))-(3/h1*(a2-a1))-h1*(alpha/2);
            elseif i == n-2
               g(i) = (3/h2*(a3-a2))-(3/h1*(a2-a1))-h1*(beta/2);
            else
               g(i) = (3/h2*(a3-a2))-(3/h1*(a2-a1));
            end
        end
        C = fTRIDIAG(Amd,Ad,g,n-2);
        for i = 2:n-1
            c(i) = C(i-1);
```

```
end
%Calculating Coefficients b and d
for i = 1:n-1
    h1 = x(i+1)-x(i);
    a2 = a(i+1); a1 = a(i);
    c2 = c(i+1); c1 = c(i);
    b(i) = (1/h1*(a2-a1))-(h1/3*(c2+2*c1));
    d(i) = (1/(3*h1))*(c2-c1);
end
%Evaluating Spline Fit
S = size(time10); m = S(2);
for i = 1:m
    for j = 1:n-1
        if time10(i) >= x(j) && time10(i) <= x(j+1)

            X_fit(i) = d(j)*(time10(i)-x(j))^3 +
                       c(j)*(time10(i)-x(j))^2 +
                       b(j)*(time10(i)-x(j)) + a(j);

            Xd_fit(i) = 3*d(j)*(time10(i)-x(j))^2 +
                        2*c(j)*(time10(i)-x(j)) + b(j);

            Xdd_fit(i) = 6*d(j)*(time10(i)-x(j)) + 2*c(j);
       end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D. MATLAB CODE TO CALCULATE THE COEFFICIENTS OF FIFTH-DEGREE HERMITE SPLINES CURVE FIT

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%     MATLAB CODE TO CALCULATE THE COEFFICIENTS     %%%%%%%%%%
%%%%%%%%%%     OF FIFTH-DEGREE HERMITE SPLINES CURVE FIT     %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [X_fit Xd_fit Xdd_fit] = fCurveFit_HS(time10,time20,X,Xd,Xdd)

        % time10 => Time matrix with time step of 10 sec
        % time20 => Time matrix with time step of 20 sec
        % X => X position obtained from the simulation given in
               Appendix A
        % Xd => X velocity obtained from the simulation given in
                Appendix A
        % Xdd => X acceleration obtained from the simulation given in
                 Appendix A

        S = size(time20); m = S(2); j = 1;
        %Calculating Coefficients a and b
        for i = 1:3:m
            x(j) = time20(i);
            y(j) = X(i);
            yPR(j) = Xd(i);
            a(j) = y(j);
            b(j) = yPR(j);
            j = j+1;
        end
        S = size(x); n = S(2);
        %Calculating Coefficient c
        c = zeros(1,n); A = zeros(n-2);
        c(1) = 0.5*Xdd(1); c(n) = 0.5*Xdd(m);
        alpha = 1; Beta1 = Xdd(1)/(2*(x(2)-x(1)));
        Beta2 = Xdd(m)/(2*(x(n)-x(n-1)));
        for i = 1:n-2
            h1 = x(i+1)-x(i); h2 = x(i+2)-x(i+1);
            if i == 1
               Amd(i) = 3*(alpha/h1+1/h2);
            elseif i == n-2
               Amd(i) = 3*(1/h1+alpha/h2);
            else
               Amd(i) = 3*(1/h1+1/h2);
            end
            A(i,i) = Amd(i);
        end
        for i = 2:n-2
            h2 = x(i+1)-x(i);
            Ad(i-1) = -1/h2;
            A(i-1,i) = Ad(i-1);
            A(i,i-1) = Ad(i-1);
        end
        for i = 1:n-2
            a3 = a(i+2); a2 = a(i+1); a1 = a(i);
```

```
    b3 = b(i+2); b2 = a(i+1); b1 = b(i);
    h2 = x(i+2)-x(i+1); h1 = x(i+1)-x(i);
    if i == 1;
       g(i) = 10*((a3-a2)/h2^3-(a2-a1)/h1^3)+4*(b1/h1^2-
              1.5*b2*(1/h2^2-1/h1^2)-b3/h2^2)+Beta1;
    elseif i == n-2
       g(i) = 10*((a3-a2)/h2^3-(a2-a1)/h1^3)+4*(b1/h1^2-
              1.5*b2*(1/h2^2-1/h1^2)-b3/h2^2)+Beta2;
    else
       g(i) = 10*((a3-a2)/h2^3-(a2-a1)/h1^3)+4*(b1/h1^2-
              1.5*b2*(1/h2^2-1/h1^2)-b3/h2^2);
    end
end
C = fTRIDIAG(Amd,Ad,g,n-2);
for i = 2:n-1
    c(i) = C(i-1);
end
%Calculating Coefficients d
for i = 1:n
    if i < n
       h1 = x(i+1)-x(i);
       a2 = a(i+1); a1 = a(i);
       b2 = b(i+1); b1 = b(i);
       c2 = c(i+1); c1 = c(i);
       d(i) = 10/h1^3*(a2-a1)-2/h1^2*(2*b2+3*b1)+1/h1*(c2-
              3*c1);
   else
       d4 = d(i-1); h4 = x(n)-x(n-1); b5 = b(i); b4 = b(i-1);
       c5 = c(i); c4 = c(i-1);
       d(i) = d4-2/h4^2*(b5-b4)+2/h4*(c5+c4);
    end
end
%Calculating Coefficients e and f
for i = 1:n-1
    h1 = x(i+1)-x(i); b2 = b(i+1); b1 = b(i);
    c2 = c(i+1); c1 = c(i); d2 = d(i+1); d1 = d(i);
    e(i) = 0.5/h1^3*(b2-b1)-1/h1^2*c1-0.25/h1*(d2+5*d1);
    e1 = e(i);
    f(i) = 0.1/(h1^3)*(c2-c1-3*d1*h1-6*e1*h1^2);
end
%Evaluating Spline Fit
S = size(time10); m = S(2);
for i = 1:m
    for j = 1:n-1
        if time10(i) >= x(j) && time10(i) <= x(j+1)

            X_fit(i) = f(j)*(time10(i)-x(j))^5 +
                       e(j)*(time10(i)-x(j))^4 +
                       d(j)*(time10(i)-x(j))^3 +
                       c(j)*(time10(i)-x(j))^2 +
                       b(j)*(time10(i)-x(j)) + a(j);

           Xd_fit(i) = 5*f(j)*(time10(i)-x(j))^4 +
                       4*e(j)*(time10(i)-x(j))^3 +
                       3*d(j)*(time10(i)-x(j))^2 +
                       2*c(j)*(time10(i)-x(j)) + b(j);
```

```
            Xdd_fit(i) = 20*f(j)*(time10(i)-x(j))^3 +
                         12*e(j)*(time10(i)-x(j))^2 +
                         6*d(j)*(time10(i)-x(j)) + 2*c(j);
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

E. MATLAB CODE TO CALCULATE THE COEFFICIENTS OF MULTI-RESOLUTION FIFTH-DEGREE HERMITE SPLINES CURVE FIT

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%      MATLAB CODE TO CALCULATE THE COEFFICIENTS      %%%%%%%
%%%%%%%%%%%%%%    OF FIFTH-DEGREE HERMITE SPLINES CURVE FIT        %%%%%%%
%%%%%%%%%%%%%%               WITH ADAPTIVE ALGORITHM               %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
close all;
format long E;
format compact;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt = 20;
[time10 time20 X Y Xd Yd Xdd Ydd] = fSimulation(dt);
%Subroutine fSimulation is the same as Appendix A with dt as the
parameter.
S = size(time20); m20 = S(2);
S = size(time10); m10 = S(2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
REGIONS = [0 75600 151200 226800 302400]; %4 regions
%Initial No. of zones in each region
nzones1 = 2; nzones2 = 2; nzones3 = 2; nzones4 = 2;
iREGIONS = 1/dt*REGIONS; S = size(iREGIONS); n = S(2);
for i = 1:n-1
    diff(i) = iREGIONS(i+1)-iREGIONS(i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Max_accept = 10^-3;
p = Max_accept + 1;
Max1 = p; Max2 = p; Max3 = p; Max4 = p;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while Max1 >= Max_accept || Max2 >= Max_accept || Max3 >= Max_accept ||
      Max4 >= Max_accept
    %%%% SPLINE POINTS %%%%%
    [x1 y1 yPR1] =
fSplinePoints(time20,diff(1),iREGIONS(1),iREGIONS(2),nzones1,X,Xd,1);
    [x2 y2 yPR2] =
fSplinePoints(time20,diff(2),iREGIONS(2),iREGIONS(3),nzones2,X,Xd,2);
    [x3 y3 yPR3] =
fSplinePoints(time20,diff(3),iREGIONS(3),iREGIONS(4),nzones3,X,Xd,2);
    [x4 y4 yPR4] =
fSplinePoints(time20,diff(4),iREGIONS(4),iREGIONS(5),nzones4,X,Xd,2);
    x = [x1 x2 x3 x4];
    y = [y1 y2 y3 y4];
    yPR = [yPR1 yPR2 yPR3 yPR4];
    %%%%% CALCULATING CURVE FIT %%%%%
    [iX_fit iXd_fit iXdd_fit] =
    fCurveFit_HS(time10,time20,x,y,yPR,Xdd);
```

```
    k = 1;
    for i = 1:2:m10
        X_fit(k) = iX_fit(i);
        k = k+1;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    X1 = X(iREGIONS(1)+1:iREGIONS(2)+1);
    X_fit1 = X_fit(iREGIONS(1)+1:iREGIONS(2)+1);
    X2 = X(iREGIONS(2)+2:iREGIONS(3)+1);
    X_fit2 = X_fit(iREGIONS(2)+2:iREGIONS(3)+1);
    X3 = X(iREGIONS(3)+2:iREGIONS(4)+1);
    X_fit3 = X_fit(iREGIONS(3)+2:iREGIONS(4)+1);
    X4 = X(iREGIONS(4)+2:iREGIONS(5)+1);
    X_fit4 = X_fit(iREGIONS(4)+2:iREGIONS(5)+1);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    S = size(X1); m1 = S(2); S = size(X2); m2 = S(2);
    S = size(X3); m3 = S(2); S = size(X4); m4 = S(2);
    %%%%% CALCULATING RESIDUALS %%%%%
    %%%%% Entire Curve Fit %%%%%
    [Residualx_CF RMSEx_CF Max_CF] = fResiduals(X,X_fit,m20);
    %%%%% In Each Region %%%%%
    [Residualx1 RMSEx1 Max1] = fResiduals(X1,X_fit1,m1);
    [Residualx2 RMSEx2 Max2] = fResiduals(X2,X_fit2,m2);
    [Residualx3 RMSEx3 Max3] = fResiduals(X3,X_fit3,m3);
    [Residualx4 RMSEx4 Max4] = fResiduals(X4,X_fit4,m4);
    %%%%% CHECKING FOR MAXIMUM RESIDUAL IN EACH REGION %%%%%
    nzones1 = fCheck(Max1,Max_accept,nzones1);
    nzones2 = fCheck(Max2,Max_accept,nzones2);
    nzones3 = fCheck(Max3,Max_accept,nzones3);
    nzones4 = fCheck(Max4,Max_accept,nzones4);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

F. SUBROUTINES USED IN THE CODE GIVEN IN APPENDIX E

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x y yPR] = fSplinePoints(time20,diff,m1,m2,nzones,X,Xd,k)

div = diff/nzones;
i = 1;
for j = m1:div:m2
    j = round(j);
    ix(i)   = time20(j+1);
    iy(i)   = X(j+1);
    iyPR(i) = Xd(j+1);
    i = i+1;
end
if k == 1
    for i = 1:1:nzones+1
        x(i)   = ix(i);
        y(i)   = iy(i);
        yPR(i) = iyPR(i);
    end
else
    for i = 2:1:nzones+1
        x(i-1)   = ix(i);
        y(i-1)   = iy(i);
        yPR(i-1) = iyPR(i);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Residual RMSE Max] = fResiduals(X,X_fit,m)

        k = 1;
        for i = 1:m
            Residual(k) = abs(X(k)-X_fit(k));
            k = k+1;
        end
        RMSE = sqrt(sum((X(:)-X_fit(:)).^2)/(m));
        Max = max(Residual);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function nzones = fCheck(Max,Max_accept,nzones)

        if Max <= Max_accept
           nzones = nzones + 0;
        else
           nzones = nzones + 1;
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

G. TRIDIAGONAL SOLVER

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                    TRIDIAGONAL SOLVER                      %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function x = fTRIDIAG(d,f,b,n)

        alpha(1) = d(1);
        gamma(1) = f(1)/alpha(1);
        for i = 2:1:n-1
            alpha(i) = d(i)-f(i-1)*gamma(i-1);
            gamma(i) = f(i)/alpha(i);
        end
        alpha(n) = d(n)-f(n-1)*gamma(n-1);
        z(1) = b(1);
        for i = 2:1:n
            z(i) = b(i)-gamma(i-1)*z(i-1);
        end
        for i = 1:1:n
            c(i) = z(i)/alpha(i);
        end
        x(n) = c(n);
        for i = n-1:-1:1
            i;
            x(i) = c(i)-gamma(i)*x(i+1);
        end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

H. MATLAB CODE TO NUMERICALLY SOLVE THE NEARBY PROBLEM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%       NUMERICAL SOLUTION TO THE NEARBY PROBLEM        %%%%%%
%%%%%%%%%%          USING FIFTH-DEGREE HERMITE SPLINES           %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
close all;
format long E;
format compact;

%%%%%%%%%%%%%%% NUMERICAL SOLUTION TO THE ORIGINAL PROBLEM %%%%%%%%%%%%%%

dt = 20;
[time10 time20 SP X Y Xd Yd Xdd Ydd Xmoon Ymoon Xdd_moon Ydd_moon] =
fSimulation(dt);
%Subroutine fSimulation is the same as Appendix A with dt as the
parameter.

%%%%%%%%%%% GENERATING CURVE FIT TO THE NUMERICAL SIMULATION %%%%%%%%%%
            %%%%%% USING FIFTH-DEGREE HERMITE SPLINES %%%%%%

[X_fit Xd_fit Xdd_fit] = fCurveFit_HS(time10,time20,X,Xd,Xdd);
[Y_fit Yd_fit Ydd_fit] = fCurveFit_HS(time10,time20,Y,Yd,Ydd);
S10 = size(time10); m10 = S10(2);
S20 = size(time20); m20 = S20(2);

%%%%%%%%%%%%%%%%%%%%%%%% CALCULATING SOURCETERMS %%%%%%%%%%%%%%%%%%%%%%%%

for k = 1:1:m10
    GMm = 4.90266e12; GMe = 3.98600436e14;
    rho = [Xmoon(k) Ymoon(k)];
    r   = [X_fit(k) Y_fit(k)];
    d   = r-rho;
    if k <= SP
        acc = -GMe*r/norm(r)^3-GMm*(d/norm(d)^3+rho/norm(rho)^3);
        acc1(k) = acc(1); acc2(k) = acc(2);
        Sourcetermx(k) = Xdd_fit(k)-acc1(k);
        Sourcetermy(k) = Ydd_fit(k)-acc2(k);
    else
        acc = -GMm*d/norm(d)^3-GMe*(r/norm(r)^3-rho/norm(rho)^3);
        acc3(k) = acc(1); acc4(k) = acc(2);
        Sourcetermx(k) = Xdd_fit(k)-Xdd_moon(k)-acc3(k);
        Sourcetermy(k) = Ydd_fit(k)-Ydd_moon(k)-acc4(k);
    end
end

%%%%%%%%%%%%%%% NUMERICAL SOLUTION TO THE NEARBY PROBLEM   %%%%%%%%%%

%%%%% PROPOGATION IN THE EC->MC FRAME %%%%%

%%%%% SETTING UP THE PROBLEM %%%%%
```

```
%%%%% USER INPUTS %%%%%
alt = 359750; %meters
angle = -36.890; %degrees
fprintf('alt = %6.0f, ang = %6.3f, dt = %6.3f \n\n', alt, angle, dt);
deltav = 3102.13; %m/s
t0 = 0; %secs
tf = 3.5*86400; %302400 secs (1 day = 86400 secs)
SP = 2.905*86400; %Switch Point at 250992 secs
folder = char(['alt' int2str(alt) 'ang' int2str(abs(angle*1000)) 'dt'
int2str(dt)]);
fid = fopen([folder 'State_SP' '.txt'], 'w');
f = fopen([folder 'Moon_State_SP' '.txt'], 'w');

%%%%% CONSTANTS %%%%%
radE = 6372797; %meters
EMdist = 384400000; %meters
omega = sqrt((GMe+GMm)/EMdist^3); %rad/sec
%Tm = 2*pi/omega = 27d 6h 49m 50.34879957310977s
alpha = angle*pi/180; %radians
h = radE + alt; %meters
Vorbit = sqrt(GMe/h); %m/s
V = Vorbit + deltav; %m/s

%%%%% INITIAL CONDITIONS %%%%%
%%%%% For Spacecraft %%%%%
rxi = h*sin(alpha); %meters
ryi = -h*cos(alpha); %meters
vxi = V*cos(alpha); %m/s
vyi = V*sin(alpha); %m/s
x = [rxi;ryi;vxi;vyi];
fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16e nn', t0, x);
%%%%% For the Moon %%%%%
xmt = EMdist*cos(omega*t0);
ymt = EMdist*sin(omega*t0);
fprintf(f, '% 6.0f, % .16f, % .16f nn', t0, xmt, ymt);

%%%%% PERFORM THE PROPOGATION %%%%%

%%%%% EARTH CENTERED FRAME %%%%%
i = 1; m = 1;
for t = t0:dt:SP-dt
    %Use the moon position at the end of the previous interval as the
    %position at the start of this interval.
    xm = xmt;
    ym = ymt;
    %Calculate the moon position at the middle and end of this
    interval.
    xmh = EMdist*cos(omega*(t + 0.5*dt));
    ymh = EMdist*sin(omega*(t + 0.5*dt));
    xmt = EMdist*cos(omega*(t + dt));
    ymt = EMdist*sin(omega*(t + dt));
    t1 = t + dt;
    fprintf(f, '% 6.0f, % .16f, % .16f nn', t1, xmt, ymt);
    %Propogating the states forward in time
    k1 = RK4_EC_NP(x, xm, ym, GMm, GMe, Sourcetermx(m),
```

```
        Sourcetermy(m));
    k2 = RK4_EC_NP(x+(dt/2)*k1', xmh, ymh, GMm, GMe, Sourcetermx(m+1),
        Sourcetermy(m+1));
    k3 = RK4_EC_NP(x+(dt/2)*k2', xmh, ymh, GMm, GMe, Sourcetermx(m+1),
        Sourcetermy(m+1));
    k4 = RK4_EC_NP(x+(dt)*k3', xmt, ymt, GMm, GMe, Sourcetermx(m+2),
        Sourcetermy(m+2));
    x = x + dt/6*(k1'+2*k2'+2*k3'+k4');
    fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16enn', t1, x);
    i = i+1;
    m = m+2;
end

%%%%% SWITCH POINT %%%%%
%Convert state vector from EC to MC
rho = [xmt; ymt];
xmt_dot = -EMdist*omega*sin(omega*(t+dt));
ymt_dot = EMdist*omega*cos(omega*(t+dt));
rho_dot = [xmt_dot; ymt_dot];
d_state = x - [rho; rho_dot];

%%%%% MOON CENTERED FRAME %%%%%
j = 1; n = m;
for t = SPt:dt:tf-dt
    %Use the moon position at the end of the previous interval as the
    %position at the start of this interval.
    xm = xmt;
    ym = ymt;
    %Calculate the moon position at the middle and end of this
    interval.
    xmh = EMdist*cos(omega*(t + 0.5*dt));
    ymh = EMdist*sin(omega*(t + 0.5*dt));
    xmt = EMdist*cos(omega*(t + dt));
    ymt = EMdist*sin(omega*(t + dt));
    t2 = t + dt;
    fprintf(f, '% 6.0f, % .16f, % .16f nn', t2, xmt, ymt);
    %Propogating the states forward in time
    k1 = RK4_MC_MNP(d_state, xm, ym, GMm, GMe, Sourcetermx(n),
        Sourcetermy(n));
    k2 = RK4_MC_MNP(d_state+(dt/2)*k1', xmh, ymh, GMm, GMe,
        Sourcetermx(n+1), Sourcetermy(n+1));
    k3 = RK4_MC_MNP(d_state+(dt/2)*k2', xmh, ymh, GMm, GMe,
        Sourcetermx(n+1), Sourcetermy(n+1));
    k4 = RK4_MC_MNP(d_state+(dt)*k3', xmt, ymt, GMm, GMe,
        Sourcetermx(n+2), Sourcetermy(n+2));
    d_state = d_state + dt/6*(k1'+2*k2'+2*k3'+k4');
    %Compute moon state at the end of interval
    rho = [xmt; ymt];
    rho_dot = EMdist*omega*[-sin(omega*(t + dt)); cos(omega*(t + dt))];
    %Convert state vector from MC to EC
    x = d_state + [rho; rho_dot];
    fprintf(fid, '% 6.0f, % .16e, % .16e, % .16e, % .16enn',t2,x);
    j = j+1; n = n+2;
end
fclose(fid); fclose(f);
```

%%

I. SUBROUTINES USED IN THE CODE GIVEN IN APPENDIX H

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% SUBROUTINE TO INTEGRATE THE EQUATION OF MOTION IN EC FRAME %%%%
%%%%%%%                    FOR THE NEARBY PROBLEM                     %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function x_dot  =
         RK4_EC_NP(x,xmoon,ymoon,GMm,GMe,Sourcetermx,Sourcetermy)

        r = x(1:2);
        v = x(3:4);
        rho = [xmoon; ymoon];
        d = r-rho;
        Sourceterm = [Sourcetermx; Sourcetermy];
        x_dot(1:2) = v;
        x_dot(3:4) = -GMe*r/norm(r)^3-
                     GMm*(d/norm(d)^3+rho/norm(rho)^3)+Sourceterm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% SUBROUTINE TO INTEGRATE THE EQUATION OF MOTION IN MC FRAME %%%%
%%%%%%%                    FOR THE NEARBY PROBLEM                     %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d_dot =
         RK4_MC_NP(d_state,xmoon,ymoon,GMm,GMe,Sourcetermx,Sourcetermy)

        d = d_state(1:2);
        v = d_state(3:4);
        rho = [xmoon; ymoon];
        r = d+rho;
        Sourceterm = [Sourcetermx; Sourcetermy];
        d_dot(1:2) = v;
        d_dot(3:4) = -GMm*d/norm(d)^3-GMe*(r/norm(r)^3-
                     rho/norm(rho)^3)+Sourceterm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```