NASA-CR-197922

(NASA-CR-197922) HIERARCHICAL CONTROL AND TRAJECTORY PLANNING Annual Report, 2 Jan. - 31 Dec. 1994 (Texas Technological Univ.) 201 p

N95-25801
--THRU--
N95-25805
Unclas

G3/63 0044078

ANNUAL REPORT

NASA GRANT #NAG 2-902

AMES RESEARCH CENTER, NASA
Moffett Field, California

submitted by

CENTER FOR APPLIED SYSTEMS ANALYSIS
Texas Tech University
MS 1042
Lubbock, Texas 79409

Title: **Hierarchical Control and Trajectory Planning**

Principal Investigator: Clyde F. Martin, P.W. Horn Professor

Dates: January 2, 1994–December 31, 1994

JAN 20 1995
TO CASI

Most of the time on this project was spent on the trajectory planning problem. As was guessed in the proposal the construction is equivalent to the classical spline construction in the case that the system matrix is nilpotent. If the dimension of the system is n then the spline of degree $2n-1$ is constructed. This gives a new approach to the construction of splines that is more efficient than the usual construction and at the same time allows the construction of a much larger class of splines. All known classes of splines are reconstructed using the approach of linear control theory. Currently one paper is essentially finished and another will be finished before the first of the year. As a numerical analysis tool control theory gives a very good tool for constructing splines. However, for the purposes of trajectory planning it is quite another story.

Consider the following simple situation $\ddot{x} = u(t)$ and $y = x(t)$ and suppose that we want to track a signal that is of the form of a simple step function, 0 to the left of zero and 1 to the right. Denote the function by s(t). Suppose we could track the signal exactly. Then $y(t) = s(t)$, $\dot{y} = \dot{s}$ and finally $u(t) = \ddot{s}$ the derivative of a delta function. We have been able to prove using control theoretic techniques that the spline construction tracks not only the function but its first two derivatives. Thus the control will approximate the derivative of a delta function. Even if we disallow discontinuous trajectories the problem remains. If the function is piecewise analytic and continuous the control will still track a derivative of a delta function or the delta function itself. The problem is that the spline construction is too good. Some improvement can be achieved by matching the initial states and terminal states but there is still a problem.

We're in the process of developing schemes that will relax the constraints that require that the system pass exactly thorough a sequence of points, but only require that the system pass within a window centered at the point. I feel that this will give a satisfactory control. The relation of this scheme to asymptotic model following is interesting. In the classical asymptotic theory a signal is given and the control is constructed in order to bring the system and control together at infinity. Here we are asking that the signal and the control be brought together at a sequence of points and at fixed times. After some thought it is clear that the scheme is going to have problems. If we consider the rather mundane situation of trying to follow another car very closely we know that in order to maintain a very tight margin very high accelerations and decelerations are necessary. The exact same thing is

happening here. I along with the students will develop a theory to relax the constraints that impose the conditions of exact matching at specified times. Acceleration of thirty G's are probably not suitable for a passenger aircraft.

I have enclose four documents which contain reports of work done under this grant. The work is proceeding at a very good pace and I feel that we have made major accoomplishments during this year.

51-63
N95-25802

Splines and Control Theory

Zhimin Zhang, John Tomlinson and Clyde Martin
Department of Mathematics, Texas Tech University, Lubbock, TX 79409

December 19, 1994

Abstract

In this work, the relationship between splines and the control theory has been analyzed. We show that spline functions can be constructed naturaly from the control theory. By establishing a framework based on control theory, we provide a simple and systematic way to construct splines. We have constructed the traditional spline functions including the polynomial splines and the classical exponential spline. We have also discovered some new spline functions such as trigonometric splines and the combination of polynomial, exponential and trigonometric splines. The method proposed in this paper is easy to implement. Some numerical experiments are performed to investigate properties of different spline approximations.

1. Introduction.

Spline functions are well known and are widely used for practical approximation of functions or more commonly for fitting smooth curves through preassigned points. Spline techniques have the advantage over most approximation and interpolation techniques in that they are computational feasible. Most of the published spline algorithms are for polynomial splines and the vast preponderance are for cubic splines. There is a small but excellent literature on the so called exponential splines and there is an even smaller literature on splines with more or less arbitrary nodal functions, [9, 3].

In this paper we will present a common frame work for splines that includes polynomial splines of all orders and generalized exponential splines of all orders. This common frame work is based on ideas from linear control theory. Let's recall some basic ideas from control theory. A linear control system is a differential equation

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t)$$

where $\vec{x} \in \mathbb{R}^n$, $\vec{u} \in \mathbb{R}^m$ and the matrices A and B are constant matrices of compatible dimension. The vector $\vec{x}$ is the state of the system and the vector $\vec{u}$ is the control. The idea is that we can use the control $\vec{u}$ to steer the state from point to point in the state space $\mathbb{R}^n$. We can think of the first component of $\vec{x}$ as representing the position of the system and for appropriate A the second coordinate is the velocity, the third acceleration, etc. A common situation, for example in air traffic control, is to specific the position that the system must be in at a sequence of times. So in fact what we have is a set of points through which the system must traverse at specified times. One could fit these points with a spline curve and then ask for the control that would move the system along that trajectory. In fact this can be done but we will show that the control law can be developed from natural control theoretic principles that will move the system through the points at the desired times and the resulting curve will be piecewise analytic and will have $2n-1$ continuous derivatives, i.e. a generalized spline. With this framework we can construct a wide variety of spline functions. If the matrix A is nilpotent then the resulting construction is just that for polynomial splines. If the matrix is 2×2 and one eigenvalue is zero and the other is a nonzero real number then the spline is the usual exponential spline. In general the nodal functions are the coordinate functions of the matrix function e^{At}.

In this paper we give a unified treatment of all of the common one dimensional spline functions using simple ideas from control theory. It is coming to be understood that there is a large overlap between linear control theory and elementary numerical analysis. Eigenvalue methods are know to be closely related to the theory of the matrix Riccati equation [2], there are close relations between observability and quadrature techniques [8], system identification and Prony's method are very similar [1] and now we see that the spline constructions and basic linear controllability are manifestations of the same phenomena.

In Section 2 we review basic material from the theory of linear control systems that is needed for the development and give a condition that characterizes the optimal control law that generates the spline functions. In Section 3 we give the details of the construction of spline functions using control theory and in Section 4 we classify the possible classes of spline functions that arise from the control theoretic construction. In Section 5 we examine in detail some of the particular classes from Section 4 and finally in Section 6 we present a series of numerical examples comparing the various classes.

2. Some results from the control theory.

In this section we collect a series of results from linear control theory. Most can be found in any control theory textbook. See, for example, the book by Brockett, [5].

Consider the linear system:

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{b}u(t), \quad t \in [0,T], \tag{2.1}$$

with

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_m \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad \vec{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{m-1}(t) \\ x_m(t) \end{pmatrix}, \tag{2.2}$$

and the observation function

$$y(t) = \vec{c}^T\vec{x}(t), \quad \vec{c}^T = (1,0,\cdots,0). \tag{2.3}$$

Let us divide $[0,T]$ into n subintervals as

$$0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T,$$

and define $h_k = t_k - t_{k-1}$, the length of the rth subinterval. Our goal is to find a control law $u \in C^{m-2}[0,T]$ that drives the system (2.1) from $\vec{x}(0) = \vec{x}^0$ to $\vec{x}(T) = \vec{x}^T$ such that the observation function $y(t)$ satisfies the interpolation conditions

$$y(t_k) = \alpha_k, \quad k = 1,\cdots,n-1. \tag{2.4}$$

Furthermore, $u(t)$ minimizes the functional

$$\int_0^T u(s)^2 ds. \tag{2.5}$$

Such a control is called an optimal control.

Definition. The system (2.1) is called controllable if for any $\vec{x}^0$, $\vec{x}^\tau$, and $\tau > 0$, there is a $u(t)$ such that,

$$\vec{x}^\tau = \vec{x}(\tau) = e^{A\tau}\vec{x}^0 + \int_0^\tau e^{A(\tau-s)}\vec{b}u(s)ds.$$

Theorem 2.1 : *The system (2.1) is controllable if and only if*

$$rank\ (\vec{b}, A\vec{b}, \cdots, A^{m-1}\vec{b}) = m. \tag{2.6}$$

For the special matrix A as in (2.2), it is easy to verify that

$$(\vec{b}, A\vec{b}, \cdots, A^{m-1}\vec{b}) = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & * \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & * & * \\ 1 & * & \cdots & * & * \end{pmatrix}, \tag{2.7}$$

and hence the condition (2.6) is satisfied. From Theorem 2.1, the system (2.1) is controllable.

Theorem 2.2 : *The system (2.1) is controllable, if and only if the matrix* $\int_0^T e^{-As}\vec{b}\vec{b}^T e^{-A^T s} ds$ *is invertible.*

For the special matrix in (2.2), we then define

$$M(t) = (\int_0^t e^{-As}\vec{b}\vec{b}^T e^{-A^T s} ds)^{-1}. \tag{2.8}$$

Theorem 2.3 : *When the system (2.1) is controllable, a control that moves the system from* $\vec{x}(\underline{t}) = \vec{\rho}_L$ *to* $\vec{x}(\bar{t}) = \vec{\rho}_R$ *given by*

$$u(t) = \vec{b}^T e^{-A^T t}(\int_{\underline{t}}^{\bar{t}} e^{-As}\vec{b}\vec{b}^T e^{-A^T s} ds)^{-1}(e^{-A\bar{t}}\vec{\rho}_R - e^{-A\underline{t}}\vec{\rho}_L), \tag{2.9}$$

minimizes the functional $J(v) = \int_{\underline{t}}^{\bar{t}} v^2(s) ds$ *among all controls that move the system from* $\vec{x}(\underline{t}) = \vec{\rho}_L$ *to* $\vec{x}(\bar{t}) = \vec{\rho}_R$.

Theorem 2.4 : *When the system (2.1) is controllable, a control* $u \in C^{m-2}[0, T]$ *that moves the system from* $\vec{x}(0) = \vec{x}^0$, *passing through* $\vec{c}^T\vec{x}(t_k) = \alpha_k$, *to* $\vec{x}(T) = \vec{x}^T$ *is given by*

$$u(t) = \sum_{k=1}^{n-1} \beta_k f_k(t) + \sum_{i=1}^{m} \gamma_i g_i(t), \tag{2.10}$$

with

$$f_k(t) = \begin{cases} \vec{e}_1^T e^{A(t_k - t)}\vec{b} & t < t_k, \\ 0 & t \geq t_k, \end{cases} \quad k = 1, \cdots, n-1,$$

$$g_i(t) = \vec{e}_i^T e^{A(t_n - t)}\vec{b}, \quad i = 1, \cdots, m,$$

where $\vec{e}_1^T = (1, 0, \cdots, 0), \cdots, \vec{e}_m^T = (0, \cdots, 0, 1)$, *and* β_k*'s,* γ_i*' are determined by* $n-1$ *interpolation conditions* $\vec{c}^T\vec{x}(t_k) = \alpha_k$ *and* m *boundary conditions* $\vec{x}(T) = \vec{x}^T$. *Moreover, the control*

(2.10) minimizes the functional $J(v) = \int_0^T v^2(s)ds$ *among all functions* $v \in C^{m-2}[0,T]$ *that drives the system (2.1) from* $\vec{x}(0) = \vec{x}^0$, *passing through* $\vec{c}^T\vec{x}(t_k) = \alpha_k$, *to* $\vec{x}(T) = \vec{x}^T$.

The construction of the optimal control is based on Hilbert space techniques and is based on writing the interior constraints in terms of a linear variety defined in terms of the functions $f_k(t)$ and the terminal constraints in terms of the functions $g_i(t)$. Once the constraints are written in terms of linear varieties the form of the optimal control is clear based on the orthogonal complement of the intersection of the varieties. this is a standard technique and is found for example in [7] or [6]. The proof of Theorem 2.4 is based on two facts: (i) $u(t)$ defined by (2.10) is $m-2$ times continuously differentiable; (ii) f_k's and g_i's are $n+m-1$ linearly independent functions. In the following, we verify (i) and (ii).

(i) From the construction (2.10), we only need to show that

$$f_k^{(r)}(t_k) = 0, \quad r = 0, \cdots, m-2, \quad k = 1, \cdots, n-1.$$

Indeed, for all $k = 1, \cdots, n-1$, $r = 0, \cdots, m-2$,

$$\lim_{t \to t_k - 0} f_k^{(r)}(t) = \lim_{t \to t_k - 0} \vec{e}_1^T(-A)^r e^{A(t_k - t)}\vec{b} = (-1)^r \vec{e}_1^T A^r \vec{b} = 0,$$

by virtue of (2.7). Hence $f_k(t)$ is $m-2$ times continuously differentiable, and so is $u(t)$.

(ii) Set $\sum_{i=1}^{m} \gamma_i g_i(t) = 0$ for $t \in [0,T]$, i.e.,

$$F(t) = \sum_{i=1}^{m} \gamma_i \vec{e}_i^T e^{A(t_n - t)}\vec{b} = \vec{\rho}^T e^{A(t_n - t)}\vec{b} = 0,$$

with $\vec{\rho} = \sum_{i=1}^{m} \gamma_i \vec{e}_i$. Then we have $F^{(r)}(t) = 0$ on $[0,T]$, especially

$$F^{(r)}(t_k) = \vec{\rho}^T(-A)^r\vec{b} = 0, \quad r = 0, \cdots m-1.$$

Therefore,

$$\vec{\rho}(\vec{b}, A\vec{b}, \cdots, A^{m-1}\vec{b}) = \vec{0}^T.$$

In light of (2.7), $\vec{\rho}$ is a zero vector and consequently, $\gamma_i = 0$, $i = 1, \cdots, m$. So g_i's are m linearly independent functions.

Next we set

$$\beta_{n-1} f_{n-1}(t) + \sum_{i=1}^{m} \gamma_i g_i(t) = 0, \quad t \in [0,T]. \tag{2.11}$$

ORIGINAL PAGE IS OF POOR QUALITY

If $\beta_{n-1} \neq 0$, then

$$f_{n-1}(t) = -\frac{1}{\beta_{n-1}} \sum_{i=1}^{m} \gamma_i g_i(t). \tag{2.12}$$

By the definition, $f_{n-1}(t) = 0$ on (t_{n-1}, t_n), So $\sum_{i=1}^{m} \gamma_i g_i(t) = 0$ for $t \in (t_{n-1}, t_n)$ which yields $\gamma_i = 0$, $i = 1, \cdots, m$, and hence $f_{n-1}(t) \equiv 0$ by (2.12). This is a contradiction. Then (2.11) yields $\beta_{n-1} = 0$, and consequently $\gamma_i = 0$, $i = 1, \cdots, m$. So f_{n-1} and g_i's are $m+1$ linearly independent functions.

Continue the above procedure by adding f_k's one by one, we are able to show that f_k's and g_i's are $m + n - 1$ linearly independent functions.

3. Construction of splines by the control theory.

Theorem 2.4 implies that an optimal control for the system (2.1) (with A given by (2.2)) is unique. But in general, β_k's and γ_i's in (2.10) are difficult to find, we then introduce a practical procedure to construct a control law that satisfies all the requirements. This control law actually leads us to a construction of spline functions.

By the existence of a control law, there exists a set of points $\vec{x}^1, \cdots, \vec{x}^{n-1}$ with $x_1^k = \alpha_k$, $k = 1, \cdots, n-1$ such that the solution of the system (2.1) satisfies $\vec{x}(t_k) = \vec{x}^k$, $k = 0, 1, \cdots, n-1, n$. By virtue of Theorem 2.3, a control law that satisfies all the requirement can be defined piecewise as

$$u(t)|_{[t_{k-1}, t_k]} = u_k(t), \quad k = 1, \cdots, n, \tag{3.1}$$

where $u_k(t)$ is given by (2.9) with $\underline{t} = t_{k-1}$, $\bar{t} = t_k$, $\vec{\rho}_L = \vec{x}^{k-1}$, and $\vec{\rho}_R = \vec{x}^k$. Then equations to find $(n-1)(m-1)$ unknowns in $\vec{x}^1, \cdots, \vec{x}^{n-1}$ (recall that $x_1^k = \alpha_k$ $k = 1, \cdots, n-1$, are known) come from $(n-1)(m-1)$ continuity conditions on $u(t)$, i.e.,

$$u_k^{(r)}(t_k) = u_{k+1}^{(r)}(t_k), \quad r = 0, \cdots, m-2, \quad k = 1, \cdots, n-1. \tag{3.2}$$

From (2.9),

$$u_k(t_k) = (A^r \vec{b})^T e^{-A^T t_k} \Big(\int_{t_{k-1}}^{t_k} e^{-As} \vec{b}\vec{b}^T e^{-A^T s} ds \Big)^{-1} (e^{-At_k} \vec{x}^k - e^{-At_{k-1}} \vec{x}^{k-1}); \tag{3.3}$$

$$u_{k+1}(t_k) = (A^r \vec{b})^T e^{-A^T t_k} \Big(\int_{t_k}^{t_{k+1}} e^{-As} \vec{b}\vec{b}^T e^{-A^T s} ds \Big)^{-1} (e^{-At_{k+1}} \vec{x}^{k+1} - e^{-At_k} \vec{x}^k). \tag{3.4}$$

Next, we shall simplify (3.3) and (3.4). Toward this end, we introduce a change of variable $s = t_{k-1} + s'$ into

$$\Big(\int_{t_{k-1}}^{t_k} e^{-As} \vec{b}\vec{b}^T e^{-A^T s} ds \Big)^{-1} = \Big(e^{-At_{k-1}} \int_0^{h_k} e^{-As'} \vec{b}\vec{b}^T e^{-A^T s'} ds' e^{-A^T t_{k-1}} \Big)^{-1}$$

$$= e^{A^T t_{k-1}} M(h_k) e^{A t_{k-1}}, \tag{3.5}$$

where $M(h_k)$ is defined by (2.8). Substituting (3.5) into (3.3), we have

$$u_k^{(r)}(t_k) = (A^r \vec{b})^T e^{-A^T h_k} M(h_k)(e^{-A h_k} \vec{x}^k - \vec{x}^{k-1}). \tag{3.6}$$

Similarly,

$$u_{k+1}^{(r)}(t_k) = (A^r \vec{b})^T M(h_{k+1})(e^{-A h_{k+1}} \vec{x}^{k+1} - \vec{x}^k). \tag{3.7}$$

Substituting (3.6) and (3.7) into (3.2) yields a linear system for $(n-1)(m-1)$ unknowns in $\vec{x}^1, \cdots, \vec{x}^{n-1}$:

$$\begin{aligned} &-(A^r \vec{b})^T e^{-A^T h_k} M(h_k) \vec{x}^{k-1} + (A^r \vec{b})^T [e^{-A^T h_k} M(h_k) e^{-A h_k} + M(h_{k+1})] \vec{x}^k \\ &-(A^r \vec{b})^T M(h_{k+1}) e^{-A h_{k+1}} \vec{x}^{k+1} = 0, \quad r = 0, \cdots, m-2, \quad k = 1, \cdots, n-1. \end{aligned} \tag{3.8}$$

By virtue of the existence and uniqueness of the optimal control, the linear system (3.8) has a unique solution and hence its coefficient matrix is invertible.

In order to solve (3.8), The following quantities needs to be calculated, A^r, e^{-Ah} ($e^{-A^T h}$), and $M(h)$. Sometimes it is easier to use the Jordan matrix of A, denoted by Λ. There exists an invertible matrix Q such that $A = Q \Lambda Q^{-1}$, and hence

$$A^r = Q \Lambda^r Q^{-1}, \quad e^{-Ah} = Q e^{-\Lambda h} Q^{-1}, \quad e^{-A^T h} = Q^{-T} e^{-\Lambda^T h} Q^T. \tag{3.9}$$

$$M(h) = (\int_0^h Q e^{-\Lambda s} Q^{-1} \vec{b} \vec{b}^T Q^{-T} e^{-\Lambda^T s} Q^T ds)^{-1} = Q^{-T} \hat{M}(h) Q^{-1}, \tag{3.10}$$

where

$$\hat{M}(h) = (\int_0^h e^{-\Lambda s} Q^{-1} \vec{b} (Q^{-1} \vec{b})^T e^{-\Lambda^T s} ds)^{-1}. \tag{3.11}$$

Substituting (3.9) and (3.10) into (3.8), we then have

$$\begin{aligned} &-(\Lambda^r Q^{-1} \vec{b})^T e^{-\Lambda^T h_k} \hat{M}(h_k) Q^{-1} \vec{x}^{k-1} + (\Lambda^r Q^{-1} \vec{b})^T [e^{-\Lambda^T h_k} \hat{M}(h_k) e^{-\Lambda h_k} + \hat{M}(h_{k+1})] Q^{-1} \vec{x}^k \\ &-(\Lambda^r Q^{-1} \vec{b})^T \hat{M}(h_{k+1}) e^{-\Lambda h_{k+1}} Q^{-1} \vec{x}^{k+1} = 0, \quad r = 0, \cdots, m-2, \quad k = 1, \cdots, n-1. \end{aligned} \tag{3.12}$$

Solving (3.8) or (3.12) for $(n-1)(m-1)$ unknowns in $\vec{x}^1, \cdots, \vec{x}^{n-1}$, we then have the control $u(t)$ defined piecewise by (3.1). The solution of the system (2.1) is thus given by

$$\begin{aligned} \vec{x}(t) &= e^{At} \vec{x}^0 + \int_0^t e^{A(t-s)} \vec{b} u(s) ds \\ &= Q(e^{\Lambda t} Q^{-1} \vec{x}^0 + \int_0^t e^{\Lambda(t-s)} Q^{-1} \vec{b} u(s) ds). \end{aligned} \tag{3.13}$$

ORIGINAL PAGE IS OF POOR QUALITY

Note that $x_i'(t) = x_{i+1}(t)$, $i = 1, \cdots, m-1$. So the continuity of $x_i'(t)$ is continuity of $x_{i+1}(t)$ for $i < m$. Further, continuity of $x_1^{(m+r)}(t)$ is continuity of $u^{(r)}(t)$, $r = 0, \cdots, m-2$. Therefore, the observation function $y(t) = \vec{c}^T\vec{x}(t) = x_1(t)$ is a $2m-2$ times continuously differentiable function that satisfies the boundary conditions

$$y^{(r)}(0) = x_{r+1}^0, \quad y^{(r)}(T) = x_{r+1}^T, \quad r = 0, \cdots, m-1 \tag{3.14}$$

and the interpolation conditions

$$y(t_k) = x_1^k, \quad k = 1, \cdots, n-1. \tag{3.15}$$

Hence $y(t)$ is a spline function. We see that from the control theory, we can derive quite general spline functions. Summing up, we have proved

Theorem 3.1 : *(1) There exists a unique function $y(t) \in C^{m-2}[0,T]$ that satisfies the boundary conditions (3.14) and the interpolation conditions (3.15); (2) $y(t)$ is the first component of the vector function $\vec{x}(t)$ given by (3.13) in which $u(s)$ is defined piecewise on each subinterval $[t_{k-1}, t_k]$, $k = 1, \cdots, n$, by*

$$\begin{aligned} u_k^{(r)}(t_k) &= \vec{b}^T e^{-A^T(t-t_{k-1})} M(h_k)(e^{-Ah_k}\vec{x}^k - \vec{x}^{k-1}) \\ &= (Q^{-1}\vec{b})^T e^{-\Lambda^T(t-t_{k-1})} \hat{M}(h_k)(e^{-\Lambda h_k} Q^{-1}\vec{x}^k - Q^{-1}\vec{x}^{k-1}), \end{aligned} \tag{3.16}$$

where $\vec{x}^k$, $k = 1, \cdots, n-1$ are determined by solving the linear systems (3.8) or (3.12) (Note that $\vec{x}^0$, $\vec{x}^n$ and x_1^k, $k = 1, \cdots, n-1$ are given by the boundary conditions (3.14) and the interpolation conditions (3.15)).

In the next section, we will see that these splines can be piecewise polynomials, trigonometric functions, exponentials or any combination. As special cases, we are able to recover classical polynomial splines (odd order) and exponential splines by properly selecting parameters $a_1, \cdots, a_m$ in (2.2) for the matrix A.

4. Classification of splines.

The type of the splines is determined by its nodal shape functions. From the control theory, we are able to construct the nodal shape functions of splines.

In order to see the kind of interpolation functions in $\vec{x}(t)$, we only need to consider one subinterval. Without loss of generality, we use the first interval $(t_0, t_1) = (0, h)$ where the solution of the system (2.1) is given by

$$\vec{x}(t) = e^{At}\vec{x}^0 + \int_0^t e^{A(t-s)}\vec{b}u(s)ds. \tag{4.1}$$

From Theorem 2.3,

$$u(t) = \vec{b}^T e^{-A^T t}(\int_0^h e^{-As}\vec{b}\vec{b}^T e^{-A^T s}ds)^{-1}(e^{-Ah}\vec{x}^1 - \vec{x}^0). \tag{4.2}$$

Substituting (4.2) into (4.1), we have

$$\begin{aligned}\vec{x}(t) &= e^{At}\vec{x}^0 + \int_0^t e^{A(t-s)}\vec{b}\vec{b}^T e^{-A^T s}ds M(h)(e^{-Ah}\vec{x}^1 - \vec{x}^0) \\ &= Qe^{\Lambda t}[Q^{-1}\vec{x}^0 + \hat{M}(t)^{-1}\hat{M}(h)(e^{-\Lambda h}Q^{-1}\vec{x}^1 - Q^{-1}\vec{x}^0)],\end{aligned} \tag{4.3}$$

where $M(h)$ and $\hat{M}(h)$ are defined by (2.8) and (3.11), respectively.

Theorem 4.1 : *Let A be given by (2.2), let $(p_1(t), \cdots, p_m(t))$ be the first row of the matrix*

$$e^{At}[I - M(t)^{-1}M(h)] = Qe^{\Lambda t}[I - \hat{M}(t)^{-1}\hat{M}(h)]Q^{-1}, \tag{4.4}$$

and let $(q_1(t), \cdots, q_m(t))$ be the first row of the matrix

$$e^{At}M(t)^{-1}M(h)e^{-Ah} = Qe^{\Lambda t}\hat{M}(t)^{-1}\hat{M}(h)e^{-\Lambda h}Q^{-1}. \tag{4.5}$$

Then for $r = 0, \cdots, m-1$,

$$p_i^{(r)}(0) = \delta_{i,r+1}, \quad p_i^{(r)}(h) = 0, \quad i = 1, \cdots, m, \tag{4.6}$$

$$q_j^{(r)}(0) = 0, \quad q_j^{(r)}(h) = \delta_{j,r+1}, \quad j = 1, \cdots, m. \tag{4.7}$$

Proof : From Theorem 2.1 and (2.7), the system (2.1) is controllable. By virtue of Theorem 3.3, a control that moves $\vec{x}(t)$ from $\vec{x}(0) = \vec{x}^0$ to $\vec{x}(h) = \vec{x}^1$ is given by (2.9) (with $\underline{t} = 0$, $\bar{t} = h$, $\vec{\rho}_L = \vec{x}^0$, $\vec{\rho}_R = \vec{x}^1$), and consequently

$$\begin{aligned}y(t) &= \vec{e}_1^T\vec{x}(t) \\ &= \vec{e}_1^T(e^{At}\vec{x}^0 + \int_0^t e^{A(t-s)}\vec{b}u(s)ds) \\ &= \vec{e}_1^T e^{At}[I - M(t)^{-1}M(h)]\vec{x}^0 + \vec{e}_1^T e^{At}M(t)^{-1}M(h)e^{-Ah}\vec{x}^1 \\ &= (p_1(t), \cdots, p_m(t))\vec{x}^0 + (q_1(t), \cdots, q_m(t))\vec{x}^1 \\ &= \sum_{i=1}^m x_i^0 p_i(t) + \sum_{j=1}^m x_j^1 q_j(t).\end{aligned} \tag{4.8}$$

Choose $\vec{x}^0 = \vec{e}_i$, $\vec{x}^1 = \vec{0}$ in (4.8), and we have

$$p_i^{(r)}(t) = y^{(r)}(t), \quad r = 0, \cdots, m-1.$$

ORIGINAL PAGE IS
OF POOR QUALITY

Therefore for $r = 0, \cdots, m-1$,

$$\begin{aligned} p_i^{(r)}(0) &= y^{(r)}(0) = x_1^{(r)}(0) = x_{r+1}(0) = x_{r+1}^0 = \delta_{i,r+1}, \\ p_i^{(r)}(h) &= y^{(r)}(h) = x_1^{(r)}(h) = x_{r+1}(h) = x_{r+1}^1 = 0, \quad i = 1, \cdots, m. \end{aligned}$$

Then we have proved (4.6). The proof of (4.7) is similar. ■

We call p_i, q_i nodal shape functions by the characteristics (4.6) and (4.7). From (4.4) and (4.5), We see that the nodal shape functions are linear combinations of function entries of matrices $e^{\Lambda t}$ and $e^{\Lambda t}\hat{M}(t)^{-1}$. In order to see the type of functions in the spline, we only need to examine the entries of these two matrices.

In the following, we classify the spline functions derived from control theory. This classification is based on the spectrum of the coefficient matrix A of the system (2.1) under different circumstances. We shall concentrate on the case $m = 2$. The reasons are: (1) The general situation for large m is very complicated and is difficult to describe precisely. (2) The case $m = 2$ has almost all features for the general case. (3) From the practical point of view, the case $m = 2$ is the most useful and important case. Let

$$A = \begin{pmatrix} 0 & 1 \\ \beta & 2\gamma \end{pmatrix}, \quad \beta, \gamma \in \mathbb{R}^1.$$

The eigenvalues of A are $\lambda_1 = \gamma + \sqrt{\gamma^2 + \beta}$, $\lambda_2 = \gamma - \sqrt{\gamma^2 + \beta}$.

1. $\gamma^2 + \beta > 0$. There are two distinct real eigenvalues. Then the Jordan matrix Λ of A, the transformation matrix Q and its inverse are given by

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix}, \quad Q^{-1} = \frac{1}{\lambda_2 - \lambda_1} \begin{pmatrix} \lambda_2 & -1 \\ -\lambda_1 & 1 \end{pmatrix}. \tag{4.9}$$

Then

$$e^{\Lambda t} = \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix}. \tag{4.10}$$

From (3.11) (by changing h to t), we have

$$\begin{aligned} \hat{M}(t)^{-1} &= \frac{1}{(\lambda_2 - \lambda_1)^2} \int_0^t \begin{pmatrix} e^{-\lambda_1 s} & 0 \\ 0 & e^{-\lambda_2 s} \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} (-1, 1) \begin{pmatrix} e^{-\lambda_1 s} & 0 \\ 0 & e^{-\lambda_2 s} \end{pmatrix} ds \\ &= \frac{1}{(\lambda_2 - \lambda_1)^2} \begin{pmatrix} (1 - e^{-2\lambda_1 t})/2\lambda_1 & (e^{-(\lambda_1+\lambda_2)t} - 1)/(\lambda_1 + \lambda_2) \\ (e^{-(\lambda_1+\lambda_2)t} - 1)/(\lambda_1 + \lambda_2) & (1 - e^{-2\lambda_2 t})/2\lambda_2 \end{pmatrix}, \end{aligned} \tag{4.11}$$

and hence

$$e^{\Lambda t}\hat{M}(t)^{-1} = \frac{1}{(\lambda_2 - \lambda_1)^2} \begin{pmatrix} (e^{\lambda_1 t} - e^{-\lambda_1 t})/2\lambda_1 & (e^{-\lambda_2 t} - e^{\lambda_1 t})/(\lambda_1 + \lambda_2) \\ (e^{-\lambda_1 t} - e^{\lambda_2 t})/(\lambda_1 + \lambda_2) & (e^{\lambda_2 t} - e^{-\lambda_2 t})/2\lambda_2 \end{pmatrix}. \tag{4.12}$$

1.a. $\gamma \neq 0$, $\beta \neq 0$. In this case $\lambda_1, \lambda_2, -\lambda_1, -\lambda_2$ are all distinct. We then have the exponential spline with basis functions given by linear combinations of $e^{\lambda_1 t}, e^{-\lambda_1 t}, e^{\lambda_2 t}, e^{-\lambda_2 t}$.

1.b. $\gamma = 0$, $\beta > 0$. In this case $\lambda_1 = -\lambda_2 = \sqrt{\beta}$, the basis functions in 1.a. degenerate. However, by applying the following limits

$$\lim_{\lambda_2 \to -\lambda_1} \frac{e^{-\lambda_2 t} - e^{\lambda_1 t}}{\lambda_1 + \lambda_2} = \lim_{\lambda_1 + \lambda_2 \to 0} \frac{e^{\lambda_1 t}(e^{-(\lambda_1+\lambda_2)t} - 1)}{\lambda_1 + \lambda_2} = -te^{\lambda_1 t},$$

$$\lim_{\lambda_2 \to -\lambda_1} \frac{e^{-\lambda_1 t} - e^{\lambda_2 t}}{\lambda_1 + \lambda_2} = \lim_{\lambda_1 + \lambda_2 \to 0} \frac{e^{-\lambda_1 t}(1 - e^{(\lambda_1+\lambda_2)t})}{\lambda_1 + \lambda_2} = -te^{-\lambda_1 t},$$

(4.12) becomes

$$e^{\Lambda t}\hat{M}(t)^{-1} = \frac{1}{4\lambda_1^2} \begin{pmatrix} (e^{\lambda_1 t} - e^{-\lambda_1 t})/2\lambda_1 & -te^{\lambda_1 t} \\ -te^{-\lambda_1 t} & (e^{\lambda_1 t} - e^{-\lambda_1 t})/2\lambda_1 \end{pmatrix}. \tag{4.13}$$

Hence we have the exponential spline with basis functions given by linear combinations of $e^{\sqrt{\beta}t}, e^{-\sqrt{\beta}t}, te^{\sqrt{\beta}t}, te^{-\sqrt{\beta}t}$.

1.c. $\beta = 0$, $\gamma \neq 0$. In this case, $\lambda_1 = 0$ (if $\gamma < 0$), or $\lambda_2 = 0$ (if $\gamma > 0$). Again the basis functions in 1.a. degenerate. Assume that $\lambda_1 = 0$, then $\lambda_2 = \lambda = 2\gamma$. From the limits

$$\lim_{\lambda_1 \to 0} \frac{(e^{\lambda_1 t} - e^{-\lambda_1 t})}{2\lambda_1} = t, \quad \lim_{\lambda_1 \to 0} e^{\lambda_1 t} = 1,$$

we have

$$e^{\Lambda t} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\lambda t} \end{pmatrix}, \quad \hat{M}(t)^{-1} = \frac{1}{\lambda^3} \begin{pmatrix} \lambda t & e^{-\lambda t} - 1 \\ e^{-\lambda t} - 1 & (1 - e^{-2\lambda t})/2 \end{pmatrix}, \tag{4.14}$$

$$e^{\Lambda t}\hat{M}(t)^{-1} = \frac{1}{\lambda^3} \begin{pmatrix} \lambda t & e^{-\lambda t} - 1 \\ 1 - e^{\lambda t} & (e^{\lambda t} - e^{-\lambda t})/2 \end{pmatrix}. \tag{4.15}$$

Therefore we end up with the exponential spline with basis functions given by linear combinations of $1, t, e^{2\gamma t}, e^{-2\gamma t}$. Later we shall further show that this is the classical exponential spline [9].

2. $\gamma^2 + \beta < 0$. There are two complex eigenvalues: $\lambda_1 = \gamma + i\omega$, $\lambda_2 = \gamma - i\omega$, where $\omega = \sqrt{-\gamma^2 - \beta}$.

2.a. $\gamma \neq 0$, $\beta < 0$. Evaluating (4.10) and (4.12), we have the exponential-trigonometric spline with basis functions given by linear combinations of $e^{\gamma t}\sin\omega t$, $e^{\gamma t}\cos\omega t$, $e^{-\gamma t}\sin\omega t$, $e^{-\gamma t}\cos\omega t$.

2.b. $\gamma = 0$, $\beta < 0$. Again this is a degenerated case where $\lambda_1 = \lambda_2 = i\omega = i\sqrt{-\beta}$. Therefore (4.10) is now

$$e^{\Lambda t} = \begin{pmatrix} \cos\omega t + i\sin\omega t & 0 \\ 0 & \cos\omega t + i\sin\omega t \end{pmatrix}. \tag{4.16}$$

Taking the limit $\gamma \to 0$ in (4.12), we then have

$$e^{\Lambda t}\hat{M}(t)^{-1} = \frac{-1}{4\omega^2}\begin{pmatrix} \sin\omega t/\omega & -t(\cos\omega t + i\sin\omega t) \\ -t(\cos\omega t - i\sin\omega t) & \sin\omega t/\omega \end{pmatrix}. \tag{4.17}$$

Hence, we have the polynomial-trigonometric spline with basis functions given by linear combinations of $\sin\sqrt{-\beta}t, \cos\sqrt{-\beta}t, t\sin\sqrt{-\beta}t, t\cos\sqrt{-\beta}t$.

3. $\gamma^2 + \beta = 0$. In this case $\lambda_1 = \lambda_2 = \gamma$.

3.a. $\gamma \neq 0$. We have non-degenerated Jordan form in this case,

$$\Lambda = \begin{pmatrix} \gamma & 1 \\ 0 & \gamma \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & -1/\gamma \\ \gamma & 0 \end{pmatrix}, \quad Q^{-1} = \begin{pmatrix} 0 & 1/\gamma \\ -\gamma & 1 \end{pmatrix}. \tag{4.18}$$

Therefore,

$$e^{\Lambda t} = \begin{pmatrix} e^{\gamma t} & te^{\gamma t} \\ 0 & e^{\gamma t} \end{pmatrix} = e^{\gamma t}\begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}, \tag{4.19}$$

$$\begin{aligned} \hat{M}(t)^{-1} &= \int_0^t e^{-2\gamma s}\begin{pmatrix} 1 & -s \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1/\gamma \\ 1 \end{pmatrix}(1/\gamma, 1)\begin{pmatrix} 1 & 0 \\ -s & 1 \end{pmatrix} ds \\ &= \int_0^t e^{-2\gamma s}\begin{pmatrix} (1/\gamma - s)^2 & 1/\gamma - s \\ 1/\gamma - s & 1 \end{pmatrix} ds. \end{aligned} \tag{4.20}$$

After some more detailed manipulation (see the next section) we can show that this is the exponential spline with basis functions given by linear combinations of $e^{\gamma t}$, $te^{\gamma t}$, $e^{-\gamma t}$, $te^{-\gamma t}$, similar to the case 1.b.

3.b. $\gamma = 0$. In this case, A itself is a Jordan matrix. We compute directly the following quantities:

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad e^{At} = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}, \tag{4.21}$$

$$M(t)^{-1} = \int_0^t \begin{pmatrix} 1 & -s \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}(0,1)\begin{pmatrix} 1 & 0 \\ -s & 1 \end{pmatrix} ds = \begin{pmatrix} t^3/3 & -t^2/2 \\ -t^2/2 & t \end{pmatrix}. \tag{4.22}$$

$$e^{At}M(t)^{-1} = \begin{pmatrix} -t^3/6 & t^2/2 \\ -t^2/2 & t \end{pmatrix}. \tag{4.23}$$

Then we have the polynomial spline with basis functions given by linear combinations of $1, t, t^2, t^3$. In the next section, we shall further show that this is the well-known cubic spline [4].

From the above discussion, we see that we may encounter all kinds of splines by varying parameters β and γ. Two general cases are 1.a. and 2.a. where we have full sized exponential

or exponential-trigonometric splines. Degeneration occurs when zero or multiple eigenvalues appear. The extremal is the case 3.b. when both eigenvalues are zero. It is this extremal case that draws most of the attention. This is evidenced by extensive investigation regarding the cubic spline in the literature. Case 1.c. also has been investigated from a different point of view. But we can hardly find any work regarding the other cases (except 1.c. and 3.b.) listed above.

The situation for $m > 2$ is similar. Let $\lambda_1, \cdots, \lambda_m$ be eigenvalues of A. When $\lambda_1, \cdots, \lambda_m$; $-\lambda_1, \cdots, -\lambda_m$ are all distinct, we have the exponential spline with the basis functions given by linear combinations of

$$e^{\lambda_1 t}, \quad e^{-\lambda_1 t}, \quad \cdots, \quad e^{\lambda_m t}, \quad e^{-\lambda_m t}.$$

See case 1.a. When complex eigenvalues appear, we get the exponential-trigonometric splines with basis functions $e^{\lambda_k t}\sin\omega t$, $e^{-\lambda_k t}\cos\omega t$ (see case 2.a.). If we have multiple eigenvalues, the terms like

$$te^{\lambda t}, \quad t\sin\omega t, \quad te^{-\lambda t}\cos\omega t, \quad t^2 e^{\lambda t}, \quad \cdots$$

will appear in basis functions (see cases 1.b., 2.b. and 3.a.). Finally, zero eigenvalues will introduce polynomials into basis functions (see case 1.c.) and the extremal situation is that all eigenvalues are zero in which case we recover polynomial splines of order $2m-1$ (see case 3.b.).

5. Examples of splines.

In this section, we shall work out in detail some classes of splines. We shall explicitly construct the nodal shape functions and the linear system needed to solve for the unknown parameters.

1. Our first example is the case 3.b. which turns out to be the classical cubic spline. We first construct the nodal shape functions. From Theorem 4.1 we need only to calculate the first row of matrix (4.4) and the first row of matrix (4.5). Let $t = h$ in (4.22), and we have

$$M(h) = \begin{pmatrix} h^3/3 & -h^2/2 \\ -h^2/2 & h \end{pmatrix}^{-1} = \frac{12}{h^4}\begin{pmatrix} h & h^2/2 \\ h^2/2 & h^3/3 \end{pmatrix}.$$

Thus from (4.21) and (4.23), we have

$$e^{At} - e^{At}M(t)^{-1}M(h) = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} -t^3/6 & t^2/2 \\ -t^2/2 & t \end{pmatrix}\frac{12}{h^4}\begin{pmatrix} h & h^2/2 \\ h^2/2 & h^3/3 \end{pmatrix}$$

ORIGINAL PAGE IS OF POOR QUALITY

$$= \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} -2t^3/h^3 + 3t^2/h^2 & t(-t^2/h^2 + 2t/h) \\ \cdot & \cdot \end{pmatrix}.$$

$$e^{At}M(t)^{-1}M(h)e^{-Ah} = \begin{pmatrix} -2t^3/h^3 + 3t^2/h^2 & t(-t^2/h^2 + 2t/h) \\ \cdot & \cdot \end{pmatrix} \begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} -2t^3/h^3 + 3t^2/h^2 & t(t^2/h^2 - t/h) \\ \cdot & \cdot \end{pmatrix}.$$

Hence

$$p_1(t) = 1 + 2(\frac{t}{h})^3 - 3(\frac{t}{h})^2, \quad p_2(t) = t[1 + (\frac{t}{h})^2 - 2(\frac{t}{h})]; \tag{5.1}$$

$$q_1(t) = -2(\frac{t}{h})^3 + 3(\frac{t}{h})^2, \quad q_2(t) = t(\frac{t}{h})[(\frac{t}{h}) - 1]. \tag{5.2}$$

They are precisely the nodal shape functions for the Hermit interpolation.

Next we set in (3.8), $r = 0$, $\vec{x}^k = (\alpha_k, \beta_k)^T$, and $h_k = h_{k+1} = h$ (by this, we are using equally spaced intervals). The equation (3.8) is now

$$-\vec{b}^T e^{-A^T h}M(h)\vec{x}^{k-1} + \vec{b}^T[e^{-A^T h}M(h)e^{-Ah} + M(h)]\vec{x}^k - \vec{b}^T M(h)e^{-Ah}\vec{x}^{k+1} = 0, \tag{5.3}$$

for $k = 1, \cdots, n-1$. Substituting

$$e^{-A^T h}M(h) = \begin{pmatrix} 1 & 0 \\ -h & 1 \end{pmatrix} \frac{12}{h^4} \begin{pmatrix} h & h^2/2 \\ h^2/2 & h^3/3 \end{pmatrix} = \frac{12}{h^3} \begin{pmatrix} 1 & h/2 \\ -h/2 & -h^2/6 \end{pmatrix},$$

$$M(h)e^{-Ah} = [e^{-A^T h}M(h)]^T = \frac{12}{h^3} \begin{pmatrix} 1 & -h/2 \\ h/2 & -h^2/6 \end{pmatrix},$$

$$e^{-A^T h}M(h)e^{-Ah} = \frac{12}{h^3} \begin{pmatrix} 1 & h/2 \\ -h/2 & -h^2/6 \end{pmatrix} \begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix} = \frac{12}{h^3} \begin{pmatrix} 1 & -h/2 \\ -h/2 & h^2/3 \end{pmatrix},$$

into (5.3) yields,

$$(\frac{6}{h^2}, \frac{2}{h}) \begin{pmatrix} \alpha_{k-1} \\ \beta_{k-1} \end{pmatrix} + (0, \frac{8}{h}) \begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} - (\frac{6}{h^2}, -\frac{2}{h}) \begin{pmatrix} \alpha_{k+1} \\ \beta_{k+1} \end{pmatrix} = 0,$$

or

$$\beta_{k-1} + 4\beta_k + \beta_{k+1} = \frac{3}{h}(\alpha_{k+1} - \alpha_{k-1}), \quad k = 1, \cdots, n-1. \tag{5.4}$$

In the matrix form (5.4) is

$$\begin{pmatrix} 4 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 4 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_{n-2} \\ \beta_{n-1} \end{pmatrix} = \frac{3}{h} \begin{pmatrix} \alpha_2 - \alpha_0 \\ \alpha_3 - \alpha_1 \\ \alpha_4 - \alpha_2 \\ \vdots \\ \alpha_{n-1} - \alpha_{n-3} \\ \alpha_n - \alpha_{n-2} \end{pmatrix} - \begin{pmatrix} \beta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \beta_n \end{pmatrix}. \tag{5.5}$$

This is precisely the same linear system as we construct the cubic spline. After solving β_k's from (5.5), the desired cubic spline can be expressed piecewisely on the subinterval $[t_{k-1}, t_k]$, $k = 1, \cdots, n$ as

$$y(t) = \alpha_{k-1}p_1(t - t_{k-1}) + \beta_{k-1}p_2(t - t_{k-1}) + \alpha_k q_1(t - t_{k-1}) + \beta_k q_2(t - t_{k-1}),$$

where p_1, p_2, q_1, q_2 are given by (5.1) and (5.2).

2. The second example is the case 1.c. which is the classical exponential spline [9]. The nodal shape functions are again derived from the first rows of matrices (4.4) and (4.5). We use the Jordan form. let $\lambda_1 = 0$, and $\lambda_2 = \lambda$ (4.9), and we have

$$Q = \begin{pmatrix} 1 & 1 \\ 0 & \lambda \end{pmatrix}, \quad Q^{-1} = \frac{1}{\lambda}\begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix}.$$

Using (4.14) we can calculate (by setting $t = h$),

$$\hat{M}(h) = \lambda^3 \begin{pmatrix} \lambda t & e^{-\lambda t} - 1 \\ e^{-\lambda t} - 1 & (1 - e^{-2\lambda t})/2 \end{pmatrix}^{-1} = C(\lambda) \begin{pmatrix} (1 + e^{-\lambda h})/2 & 1 \\ 1 & \lambda h(1 - e^{-\lambda h})^{-1} \end{pmatrix},$$

where

$$C(\lambda) = \frac{2\lambda^3}{\lambda h(1 + e^{-\lambda h}) - 2(1 - e^{-\lambda h})}.$$

Recall (4.14) and (4.15), and we have

$$\begin{aligned}
& Qe^{\Lambda t}[I - \hat{M}(t)^{-1}\hat{M}(h)]Q^{-1} \\
= & \begin{pmatrix} 1 & 1 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{\lambda t} \end{pmatrix} [I - \frac{C(\lambda)}{\lambda^3} \begin{pmatrix} \lambda t & e^{-\lambda t} - 1 \\ 1 - e^{\lambda t} & (e^{\lambda t} - e^{-\lambda t})/2 \end{pmatrix} \cdot \\
& \begin{pmatrix} (1 + e^{-\lambda h})/2 & 1 \\ 1 & \lambda h(1 - e^{-\lambda h})^{-1} \end{pmatrix}] \frac{1}{\lambda} \begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix} \\
= & \begin{pmatrix} 1 & (e^{\lambda t} - 1)/\lambda \\ 0 & e^{\lambda t} \end{pmatrix} - \frac{C(\lambda)}{\lambda^4} \begin{pmatrix} 1 & 1 \\ 0 & \lambda t \end{pmatrix} \begin{pmatrix} \lambda t & e^{-\lambda t} - 1 \\ 1 - e^{\lambda t} & (e^{\lambda t} - e^{-\lambda t})/2 \end{pmatrix} \cdot \\
& \begin{pmatrix} (1 + e^{-\lambda h})/2 & 1 \\ 1 & \lambda h(1 - e^{-\lambda h})^{-1} \end{pmatrix} \begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix} \\
= & \begin{pmatrix} p_1(\lambda; t) & p_2(\lambda; t) \\ \cdot & \cdot \end{pmatrix},
\end{aligned}$$

where

$$p_1(\lambda; t) = 1 - \frac{\lambda t(1 + e^{-\lambda h}) - (1 - e^{-\lambda h}) - e^{\lambda(t-h)} + e^{-\lambda t}}{\lambda h(1 + e^{-\lambda h}) - 2(1 - e^{-\lambda h})}, \tag{5.6}$$

$$p_2(\lambda;t) = \frac{e^{\lambda t}-1}{\lambda} - \frac{h}{1+e^{-\lambda h}-2\frac{1-e^{-\lambda h}}{\lambda h}}[\frac{1-e^{-\lambda h}}{\lambda h}(\frac{t}{h}+\frac{1-e^{\lambda t}}{\lambda h}) - \frac{1-\frac{\lambda h}{(1-e^{-\lambda h})}}{\lambda h}\cdot\frac{e^{\lambda t}-2+e^{-\lambda t}}{\lambda h}]. \tag{5.7}$$

$$\begin{aligned}
& Qe^{\Lambda t}\hat{M}(t)^{-1}\hat{M}(h)e^{-\Lambda h}Q^{-1} \\
= \; & \frac{C(\lambda)}{\lambda^4}\begin{pmatrix}1 & 1\\ 0 & \lambda\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & e^{\lambda t}\end{pmatrix}\begin{pmatrix}\lambda t & e^{-\lambda t}-1\\ e^{-\lambda t}-1 & (1-e^{-2\lambda t})/2\end{pmatrix}\cdot \\
& \begin{pmatrix}(1+e^{-\lambda h})/2 & 1\\ 1 & \lambda h(1-e^{-\lambda h})^{-1}\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & e^{-\lambda h}\end{pmatrix}\begin{pmatrix}\lambda & -1\\ 0 & 1\end{pmatrix} \\
= \; & \frac{C(\lambda)}{\lambda^4}\begin{pmatrix}1 & 1\\ 0 & \lambda\end{pmatrix}\begin{pmatrix}\lambda t & e^{-\lambda t}-1\\ 1-e^{\lambda t} & (e^{\lambda t}-e^{-\lambda t})/2\end{pmatrix}\begin{pmatrix}(1+e^{-\lambda h})/2 & e^{-\lambda h}\\ 1 & \lambda h(e^{\lambda h}-1)^{-1}\end{pmatrix}\cdot \\
& \begin{pmatrix}\lambda & -1\\ 0 & 1\end{pmatrix} \\
= \; & \begin{pmatrix}q_1(\lambda;t) & q_2(\lambda;t)\\ \cdot & \cdot\end{pmatrix},
\end{aligned}$$

where

$$q_1(\lambda;t) = 1 - p_1(\lambda;t), \tag{5.8}$$

$$q_2(\lambda;t) = -\frac{h}{1+e^{-\lambda h}-2\frac{1-e^{-\lambda h}}{\lambda h}}[\frac{1-e^{-\lambda h}}{\lambda h}(\frac{t}{h}+\frac{1-e^{\lambda t}}{\lambda h}) - \frac{\frac{\lambda h}{e^{\lambda h}-1}-1}{\lambda h}\cdot\frac{e^{\lambda t}-2+e^{-\lambda t}}{\lambda h}]. \tag{5.9}$$

(5.6) - (5.9) are nodal shape functions for the classical exponential spline.

Next we set in (3.12), $r = 0$, $\vec{x}^k = (\alpha_k, \beta_k)^T$, and $h_k = h_{k+1} = h$. The equation (3.12) is now

$$\begin{aligned}
& -(Q^{-1}\vec{b})^T e^{-\Lambda^T h}\hat{M}(h)Q^{-1}\vec{x}^{k-1} + (Q^{-1}\vec{b})^T[e^{-\Lambda^T h}\hat{M}(h)e^{-\Lambda h} + \hat{M}(h)]Q^{-1}\vec{x}^k \\
& -(Q^{-1}\vec{b})^T\hat{M}(h)e^{-\Lambda h}Q^{-1}\vec{x}^{k+1} = 0, \quad k = 1,\cdots,n-1.
\end{aligned} \tag{5.10}$$

Substituting

$$(Q^{-1}\vec{b})^T = \frac{1}{\lambda}(-1,1),$$

$$\begin{aligned}
e^{-\Lambda h}\hat{M}(h) &= C(\lambda)\begin{pmatrix}1 & 0\\ 0 & e^{-\lambda h}\end{pmatrix}\begin{pmatrix}(1+e^{-\lambda h})/2 & 1\\ 1 & \lambda h(1-e^{-\lambda h})^{-1}\end{pmatrix} \\
&= C(\lambda)\begin{pmatrix}(1+e^{-\lambda h})/2 & 1\\ e^{-\lambda h} & \lambda h(e^{\lambda h}-1)^{-1}\end{pmatrix},
\end{aligned}$$

$$\hat{M}(h)e^{-\Lambda h} = [e^{-\Lambda h}\hat{M}(h)]^T = C(\lambda)\begin{pmatrix}(1+e^{-\lambda h})/2 & e^{-\lambda h}\\ 1 & \lambda h(e^{\lambda h}-1)^{-1}\end{pmatrix},$$

$$
\begin{aligned}
e^{-\Lambda h}\hat{M}(h)e^{-\Lambda h} &= C(\lambda)\begin{pmatrix} 1 & 0 \\ 0 & e^{-\lambda h} \end{pmatrix}\begin{pmatrix} (1+e^{-\lambda h})/2 & e^{-\lambda h} \\ 1 & \lambda h(e^{\lambda h}-1)^{-1} \end{pmatrix} \\
&= C(\lambda)\begin{pmatrix} (1+e^{-\lambda h})/2 & e^{-\lambda h} \\ e^{-\lambda h} & \lambda h e^{-\lambda h}(e^{\lambda h}-1)^{-1} \end{pmatrix},
\end{aligned}
$$

into (5.10), canceling $C(\lambda)/\lambda^2$, we have

$$
\begin{aligned}
&-(-1,1)\begin{pmatrix} (1+e^{-\lambda h})/2 & 1 \\ e^{-\lambda h} & \lambda h(e^{\lambda h}-1)^{-1} \end{pmatrix}\begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \alpha_{k-1} \\ \beta_{k-1} \end{pmatrix} \\
&+(-1,1)\begin{pmatrix} 1+e^{-\lambda h} & 1+e^{-\lambda h} \\ 1+e^{-\lambda h} & \lambda h(1+e^{-2\lambda h}(1-e^{-\lambda h})^{-1} \end{pmatrix}\begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \alpha_{k} \\ \beta_{k} \end{pmatrix} \\
&-(-1,1)\begin{pmatrix} (1+e^{-\lambda h})/2 & e^{-\lambda h} \\ 1 & \lambda h(e^{\lambda h}-1)^{-1} \end{pmatrix}\begin{pmatrix} \lambda & -1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \alpha_{k+1} \\ \beta_{k+1} \end{pmatrix} = 0,
\end{aligned}
$$

or

$$
\begin{aligned}
&\frac{1-e^{-2\lambda h}-2\lambda h e^{-\lambda h}}{2(1-e^{-\lambda h})}(\beta_{k-1}+\beta_{k+1})+[\frac{\lambda h(1+e^{-2\lambda h})}{1-e^{-\lambda h}}-(1+e^{-\lambda h})]\beta_k \\
&= \lambda\frac{1-e^{-\lambda h}}{2}(\alpha_{k+1}-\alpha_{k-1}), \quad k=1,\cdots,n-1. \qquad (5.11)
\end{aligned}
$$

This is the linear system for the exponential spline, it can be written in the matrix form as

$$
\begin{aligned}
&\begin{pmatrix} a(\lambda;h) & b(\lambda;h) & 0 & \cdots & 0 & 0 \\ b(\lambda;h) & a(\lambda;h) & b(\lambda;h) & \cdots & 0 & 0 \\ 0 & b(\lambda;h) & a(\lambda;h) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a(\lambda;h) & b(\lambda;h) \\ 0 & 0 & 0 & \cdots & b(\lambda;h) & a(\lambda;h) \end{pmatrix}\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_{n-2} \\ \beta_{n-1} \end{pmatrix} \\
&= \frac{3}{h}\begin{pmatrix} \alpha_2-\alpha_0 \\ \alpha_3-\alpha_1 \\ \alpha_4-\alpha_2 \\ \vdots \\ \alpha_{n-1}-\alpha_{n-3} \\ \alpha_n-\alpha_{n-2} \end{pmatrix}-\begin{pmatrix} b(\lambda;h)\beta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ b(\lambda;h)\beta_n \end{pmatrix}, \qquad (5.12)
\end{aligned}
$$

where

$$
a(\lambda;h)=6\frac{\lambda h(1+e^{-2\lambda h})-(1-e^{-2\lambda h})}{\lambda h(1-e^{-\lambda h})^2}, \quad b(\lambda;h)=3\frac{1-e^{-2\lambda h}-2\lambda h e^{-\lambda h}}{\lambda h(1-e^{-\lambda h})^2}.
$$

Again, after finding β_k's, the spline can be expressed piecewise by the nodal shape functions.

It is interesting to examine the limiting case for the exponential spline obtained above. Applying the L'Hospital's rule three times, we are able to verify that

$$\lim_{\lambda\to 0} a(\lambda;h) = 4, \quad \lim_{\lambda\to 0} b(\lambda;h) = 1. \tag{5.13}$$

We then recover (5.5), the tridiagonal systems for the cubic spline. Further we can verify that (by successively using the L'Hospital's rule)

$$\lim_{\lambda\to 0} q_1(\lambda;h) = \frac{3}{h}t - 1 - (\frac{t}{h}-1)^3 - (\frac{t}{h})^3 = -2(\frac{t}{h})^3 + 3(\frac{t}{h})^2 = q_1(t),$$

$q_1(t)$ is one of the nodal shape functions for the cubic spline given by (5.2). The other three nodal shape functions can be verified similarly. So the cubic spline is the limiting case for the exponential spline 4.1.c. when $\lambda \to 0$. This is not a surprise from the following limit regarding matrix A of the system (2.1),

$$\lim_{\lambda\to 0}\begin{pmatrix} 0 & 1 \\ 0 & \lambda \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \tag{5.14}$$

where the left hand side is the matrix A for the case 1.c., and the right hand side is the matrix A for the case 3.b. We can also examine the limit when $\lambda \to \infty$ where

$$\lim_{\lambda\to\infty} a(\lambda;h) = 6, \quad \lim_{\lambda\to\infty} b(\lambda;h) = 0, \tag{5.15}$$

$$\lim_{\lambda\to\infty} p_1(\lambda;t) = 1 - \frac{t}{h}, \quad \lim_{\lambda\to\infty} q_1(\lambda;t) = \frac{t}{h}, \tag{5.16}$$

$$\lim_{\lambda\to\infty} p_2(\lambda;t) = 0 = \lim_{\lambda\to\infty} q_2(\lambda;t). \tag{5.17}$$

Substituting (5.15) into (5.12), we then have $\beta_k = (\alpha_{k+1} - \alpha_{k-1})/2h$, $k = 1, \cdots, n-1$, which is the central difference scheme. We see that (5.16) gives the nodal shape functions for the linear interpolation. In order to verify (5.17), it is convenient to rewrite

$$\begin{aligned} p_2(\lambda;t) &= -\frac{h}{\lambda h - 2}\cdot\frac{e^{-\lambda h}}{1-e^{-\lambda h}} - \frac{(\lambda h+2)e^{-\lambda h}}{(\lambda h-2)[\lambda h(1+e^{-\lambda h}) - 2(1-e^{-\lambda h})]}(\frac{2}{\lambda} - \frac{h}{1-e^{-\lambda h}}) \\ &- \frac{1}{\lambda} + \frac{C(\lambda)}{2\lambda^3}[-t(1-e^{-\lambda h}) + \frac{e^{-\lambda h}-2}{\lambda} - \frac{e^{\lambda(t-h)}}{\lambda} - \frac{1-e^{-\lambda t}}{\lambda} - \frac{h}{1-e^{-\lambda h}}(e^{-\lambda t}-2)], \\ q_2(\lambda;t) &= \frac{C(\lambda)}{2\lambda^3}[-t(1-e^{-\lambda h}) + \frac{1-e^{-\lambda t}}{\lambda} + \frac{e^{-\lambda h} - e^{\lambda(t-h)}}{\lambda} + \frac{h(e^{\lambda(t-h)} - 2e^{-\lambda h} + e^{-\lambda(t+h)})}{1-e^{-\lambda h}}]. \end{aligned}$$

So the linear spline (the piecewise linear interpolation) is the limiting case for the exponential spline 4.1.c. when $\lambda \to \infty$.

3. The third example is the case 2.b. when $\beta = -\omega^2$, $\gamma = 0$ and

$$A = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}.$$

Denote

$$C_1 = \omega^2 h^2 - \sin^2 \omega h, \quad C_2 = \frac{1}{2}\sin 2\omega h + \omega h, \quad C_3 = \sin^2 \omega h, \quad C_4 = \frac{1}{2}\sin 2\omega h - \omega h.$$

Following the same procedure as the first example, we have the nodal shape functions as following

$$\begin{aligned} C_1 p_1(t) &= C_1 \cos \omega t + C_2(\sin \omega t - \omega t \cos \omega t) - C_3 \omega t \sin \omega t, \\ C_1 p_2(t) &= \omega h^2 \sin \omega t - C_3 t \cos \omega t + C_4 t \sin \omega t, \\ q_1(t) &= 1 - p_1(t), \\ C_1 q_2(t) &= h \sin \omega h(\sin \omega t - \omega t \cos \omega t) - t \sin \omega t(\sin \omega h - \omega h \cos \omega h). \end{aligned}$$

When evenly spaced intervals are used, the tridiagonal system for unknowns β_k is given by

$$b(\omega; h)\beta_{k-1} + a(\omega; h)\beta_k + b(\omega; h)\beta_{k+1} = c(\omega; h)(\alpha_{k+1} - \alpha_{k-1}), \quad k = 1, \cdots, n, \qquad (5.18)$$

where

$$a(\omega; h) = 2\omega h - \sin 2\omega h, \quad b(\omega; h) = \sin \omega h - \omega h \sin \omega h, \quad c(\omega; h) = \omega^2 h \sin \omega h.$$

It is easy to verify that,

$$\lim_{\omega \to 0} \frac{3a(\omega; h)}{(\omega h)^3} = 4, \quad \lim_{\omega \to 0} \frac{3b(\omega; h)}{(\omega h)^3} = 1, \quad \lim_{\omega \to 0} \frac{3c(\omega; h)}{(\omega h)^3} = \frac{3}{h},$$

which are the coefficients for the tridiagonal system of the cubic spline. We can also verify that at the limit $\omega \to 0$, the nodal shape functions $p_i(\omega; t)$ and $q_i(\omega; t)$, $i = 1, 2$, have the relative nodal shape functions of the cubic spline as their limit when $\omega \to 0$. Indeed, we recover the cubic spline from this trigonometric spline when $\omega \to 0$.

4. The fourth example is the case 3.a. where $\beta = -\gamma^2$ and

$$A = \begin{pmatrix} 0 & 1 \\ -\gamma^2 & 2\gamma \end{pmatrix}.$$

Denote

$$C = 1 - 2e^{-2\gamma h} + e^{-4\gamma h} - 4\gamma^2 h^2 e^{-2\gamma h}.$$

Computing the first rows of matrices $e^{At} - e^{At}M(t)^{-1}M(h)$ and $e^{At}M(t)^{-1}M(h)e^{-Ah}$, we have the nodal shape functions

$$\begin{aligned}
Cp_1(t) &= e^{-\gamma(2h+t)}(2\gamma h - 2\gamma^2 h^2 - \gamma t + 2\gamma^2 ht - 1) + e^{-\gamma t}(1+\gamma t) \\
&\quad + e^{-\gamma(2h-t)}(\gamma t + 2\gamma^2 ht - 2\gamma h + 2\gamma^2 h^2 - 1) + e^{-\gamma(4h-t)}(1-\gamma t), \\
Cp_2(t) &= t(e^{-\gamma(4h-t)} + e^{-\gamma t}) + e^{-\gamma(2h-t)}(2\gamma ht - 2\gamma h^2 - t) + e^{-\gamma(2h+t)}(2\gamma h^2 - t - 2\gamma ht), \\
q_1(t) &= 1 - p_1(t), \\
Cq_2(t) &= e^{-\gamma(3h-t)}(h - t - 2\gamma ht) + (e^{-\gamma(3h+t)} + e^{-\gamma(h-t)})(t-h) + e^{-\gamma(h+t)}(2\gamma ht + h - t).
\end{aligned}$$

Evaluating (5.3) in the current case, we get the tridiagonal system (5.18) with

$$\begin{aligned}
a(\omega;h) = a(\gamma;h) &= e^{\gamma h}(1 - e^{-4\gamma h} - 4\gamma h e^{-2\gamma h}), \\
b(\omega;h) = b(\gamma;h) &= e^{-2\gamma h}(1+\gamma h) - (1-\gamma h), \\
c(\omega;h) = c(\gamma;h) &= \gamma^2 h(1 - e^{-2\gamma h}).
\end{aligned}$$

Again we can verify that the cubic spline is the limiting case for this exponential spline when $\gamma \to 0$.

5. Our last example is a case for $m = 3$ when

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.19}$$

The system (2.1) with A given by (5.19) produces the quintic spline. In this case,

$$e^{At} = \begin{pmatrix} 1 & t & t^2/2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{pmatrix}, \tag{5.20}$$

$$M(t)^{-1} = \int_0^t e^{-As}\vec{b}\vec{b}^T e^{-A^T s} ds = \begin{pmatrix} t^5/(20) & -t^4/8 & t^3/6 \\ -t^4/8 & t^3/3 & -t^2/2 \\ t^3/6 & -t^2/2 & t \end{pmatrix}, \tag{5.21}$$

$$M(h) = [M(h)^{-1}]^{-1} = 3\begin{pmatrix} 240/h^5 & 120/h^4 & 20/h^3 \\ 120/h^4 & 64/h^3 & 12/h^2 \\ 20/h^3 & 12/h^2 & 3/h \end{pmatrix}. \tag{5.22}$$

Using (5.20) - (5.22) to compute the first rows of matrices $e^{At} - e^{At}M(t)^{-1}M(h)$ and $e^{At}M(t)^{-1}M(h)e^{-Ah}$, we then have the nodal shape functions for the quintic interpolation:

$$p_1(t) = \frac{(h-t)^3}{h^5}[h^2 + 3ht + 6t^2], \quad q_1(t) = \frac{t^3}{h^5}[h^2 - 3ht + 6t^2],$$

$$p_2(t) = \frac{(h-t)^3}{h^4}[ht+3t^2], \quad q_2(t) = \frac{t^3}{h^4}[h(t-h)-3t^2],$$
$$p_3(t) = \frac{(h-t)^3t^2}{2h^3}, \quad q_3(t) = \frac{t^3(t-h)^2}{2h^3}.$$

In order to compute the parameters for the optimal control, we set in (3.8) $r = 0, 1$ $\vec{x}^k = (\alpha_k, \beta_k, \gamma_k)^T$, and $h_k = h_{k+1} = h$. Then except (5.3), we also have

$$-(A\vec{b})^T e^{-A^T h} M(h)\vec{x}^{k-1} + (A\vec{b})^T[e^{-A^T h}M(h)e^{-Ah} + M(h)]\vec{x}^k - (A\vec{b})^T M(h)e^{-Ah}\vec{x}^{k+1} = 0, \tag{5.23}$$

for $k = 1, \cdots, n-1$. Substituting (5.20) - (5.22) into (5.3) and (5.23), after some tedious symbolic manipulation, we have the following linear system,

$$8(\beta_{k+1} - \beta_{k-1}) + h(-\gamma_{k-1} + 6\gamma_i - \gamma_{k+1}) = \frac{20}{h}(f_{k-1} - 2f_k + f_{k+1}); \tag{5.24}$$

$$7\beta_{k-1} + 16\beta_k + 7\beta_{k+1} + h(\gamma_{k-1} - \gamma_{k+1}) = \frac{15}{h}(f_{k+1} - f_{k-1}), \tag{5.25}$$

for $k = 1, \cdots, n-1$. If we arrange the unknowns as $(\beta_1, h\gamma_1, \cdots, \beta_{n-1}, h\gamma_{n-1})$, we will get a linear system with a banded 6-diagonal coefficient matrix; if we arrange the unknowns as $(\beta_1, \cdots, \beta_{n-1}, h\gamma_1, \cdots, h\gamma_{n-1}) = (\vec{\beta}^T, h\vec{\gamma}^T)$, we will have a linear system with a block tridiagonal coefficient matrix.

$$\begin{bmatrix} S & E^T \\ 8E & H \end{bmatrix} \begin{bmatrix} \vec{\beta} \\ h\vec{\gamma} \end{bmatrix} = \begin{bmatrix} \vec{f} \\ \vec{g} \end{bmatrix}, \tag{5.26}$$

where

$$S = \begin{bmatrix} 16 & 7 & 0 & \cdots & 0 & 0 \\ 7 & 16 & 7 & \cdots & 0 & 0 \\ 0 & 7 & 16 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 16 & 7 \\ 0 & 0 & 0 & \cdots & 7 & 16 \end{bmatrix}, \quad H = \begin{bmatrix} 6 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 6 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 6 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 6 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 6 \end{bmatrix},$$

$$E = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & -1 & 0 \end{bmatrix},$$

$$\vec{f} = \begin{bmatrix} 15(f_2 - f_0)/h - 7\beta_0 - h\gamma_0 \\ 15(f_3 - f_1)/h \\ 15(f_4 - f_2)/h \\ \vdots \\ 15(f_{n-1} - f_{n-3})/h \\ 15(f_n - f_{n-2})/h - 7\beta_n + h\gamma_n \end{bmatrix}, \quad \vec{g} = \begin{bmatrix} 20(f_0 - 2f_1 + f_2)/h + 8\beta_0 + h\gamma_0 \\ 20(f_1 - 2f_2 + f_3)/h \\ 20(f_2 - 2f_3 + f_4)/h \\ \vdots \\ 20(f_{n-3} - 2f_{n-2} + f_{n-1})/h \\ 20(f_{n-2} - 2f_{n-1} + f_n)/h - 8\beta_n + h\gamma_n \end{bmatrix}.$$

In general, if A is a $m \times m$ nilpotent matrix with 1's on the super diagonal and 0's elsewhere, we shall recover all odd degree polynomial splines (with degree $2m - 1$).

6. Numerical experiments.

In this section, we test the behaviors of different splines numerically. Equally spaced intervals are used for all computations.

Example 1. Comparison of the cubic spline with the quintic spline.

Test function 1.

$$f(t) = \begin{cases} 0 & -1 \le t < 0 \\ 1/2 & t = 0 \\ 1 & 0 < t \le 1 \end{cases}$$

For the cubic spline, we pose, in (5.5), the boundary conditions:

$$\beta_0 = 0 = \beta_n;$$

and for the quintic spline, we pose, in (5.25), the boundary condition:

$$\beta_0 = 0 = \beta_n, \quad \gamma_0 = 0 = \gamma_n.$$

Recall that β_i and γ_j are the coefficients for the first and the second derivatives, respectively. The spline functions are then constructed for $h = .2$, $h = .1$, $h = .05$, and $h = .025$. Graphs are plotted in Figure 1(a), 1(b). We see that the qualitative behavior of the two splines are almost same, but the quintic spline has a little better accuracy.

One interesting phenomenon is that the mesh refinement does not effect the maximum overshoot of the spline approximation. Since this is very similar to the Gibbs phenomenon for the Fourier series, we term it as "Gibbs phenomenon" of splines. In fact, all spline functions have this property.

Test function 2.

$$g(t) = e^{-10t^3}, \quad -1 \le t \le 0.$$

For the cubic spline, we pose the boundary conditions:

$$\beta_0 = -30e^{10}, \quad \beta_n = 0;$$

and for the quintic spline, we pose the boundary conditions:

$$\beta_0 = -30e^{10}, \quad \beta_n = 0, \quad \gamma_0 = 960e^{10}, \quad \gamma_n = 0.$$

We use the mesh size $h = .2$, and plot the graphs in Figure 1(c), 1(d). We see that the quintic spline gives much better approximation in the neighborhood of $x = -1$ since it has the correct concavity information at $x = -1$ which the cubic spline does not have.

Example 2. Properties of the classical exponential spline case 1.c.

The test function is the same $f(t)$ as in Example 1. We have observed that for small parameter λ, the behavior is much like the cubic spline. This is not surprise from (5.14). The interesting fact is: For the moderate λ, the graph is very much the same as the cubic spline (Figure 2(a)). If we fix the parameter λ and refine the mesh, we observe Gibbs phenomenon as in the cubic and the quintic splines. But if we fix the mesh (here we choose $h = .1$) and increase the parameter λ, we see that the approximation converges to the piecewise linear function (Figure 2(b), 2(c), 2(d)). This confirms our theoretical analysis made in Section 5.

Example 3. Properties of the exponential spline case 3.a.

We use the same test function $f(t)$ as in the examples 1, 2.

For small parameter γ, the approximating feature of this spline is also like the cubic spline including the Gibbs phenomenon. But when we fix the mesh (here $h = .1$) and increase the parameter γ, an unexpected wiggling appears at $t = 0$ (Figure 3(a)-3(d)).

Example 4. Properties of the exponential spline case 1.a.

Here we choose

$$A = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix},$$

and we have $\lambda_1 = -1$, $\lambda_2 = -2$. Again, the testing function is $f(t)$ as in the previous examples.

We plot the approximation for $h = .1$, $h = .05$, $h = .025$ in Figure 4(a), 4(b), 4(c), respectively. again, we observe the similar behavior as that of the cubic spline.

Conclusions

1. Gibbs phenomenon exists for all splines.
2. The quintic spline is recommended if the concavity is important.

3. From the approximation point of view, the classical exponential spline is preferred when the function has points of discontinuity.

A final remark.

For the discussion purpose, we constructed spline approximation in this paper by introducing the nodal shape functions which is not necessary in practical computation. From the framework we have established based on the control theory in Section 3, all we need to do is: providing the matrix A, the vector $\vec{b}$ to the linear system (3.8), solving (3.8) numerically to obtain $\vec{x}^k$'s, and hence the control law $u(t)$ (see (2.9)). After we have the control $u(t)$, the expected spline function is given by the first component of $\vec{x}(t)$ defined by (3.13). Based on our analysis, we are able to choose different splines by simply selecting entries of the matrix A.

The significance of this investigation is two fold: first, it exposes the relationship between two important fields - control theory and spline approximations. This enables us to discover new spline functions and to investigate, systematically, the properties of the spline approximations. Secondly, it provides a practical way to construct different splines from a same simple framework. From our experience, we feel that this construction is more natural and easier than the traditional approach.

Acknowledgment

This work of the third author was partially supported by NASA Grants NAG 2-902 and NAG 2-899 and a grant from the Texas Tech Leather Research Institute.

References

[1] Ammar, G., Dayawansa, W., Martin, C. (1991): Exponential interpolation: theory and numerical algorithms. Applied Mathematics and Computation **44**, 189-232

[2] Ammar, G., Martin, C. (1986): Geometry of matrix eigenvalue methods. Acta Applicandae Mathematicae **5**, 239-278

[3] Böckmann, C (1995): A modification of the trust-region Gauss-Newton method to solve separable nonlinear least squares problems. Journal of Math. Syst., Est. and Control **5**, 111-115

[4] de Boor, C. (1978): A Practical Guide to Splines. Springer-Verlag

[5] Brockett, R.W. (1970): Finite Dimensional linear systems. John Wiley & Sons

[6] Doolin, B., and Martin, C. (1990): An introduction to differential geometry for control engineers. New York: Marcel Dekker, Inc.

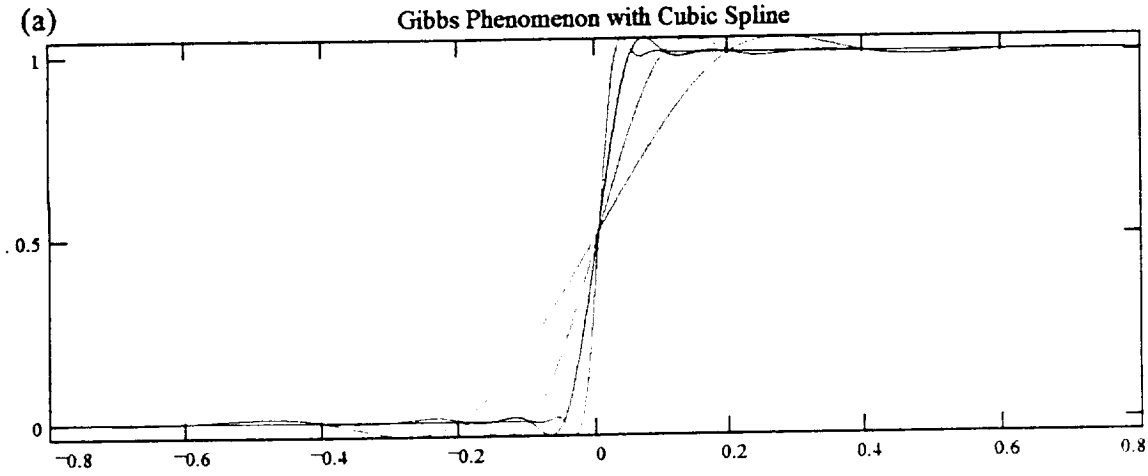
[7] Luenberger, D.G. (1969): Optimization by Vector Space Methods. John Wiley & Sons

[8] Martin, C., Stamp, M., Wang, X. (1991): Discrete observability and numerical quadrature. IEEE Tran. Aut. Control **36**, 1337-1340

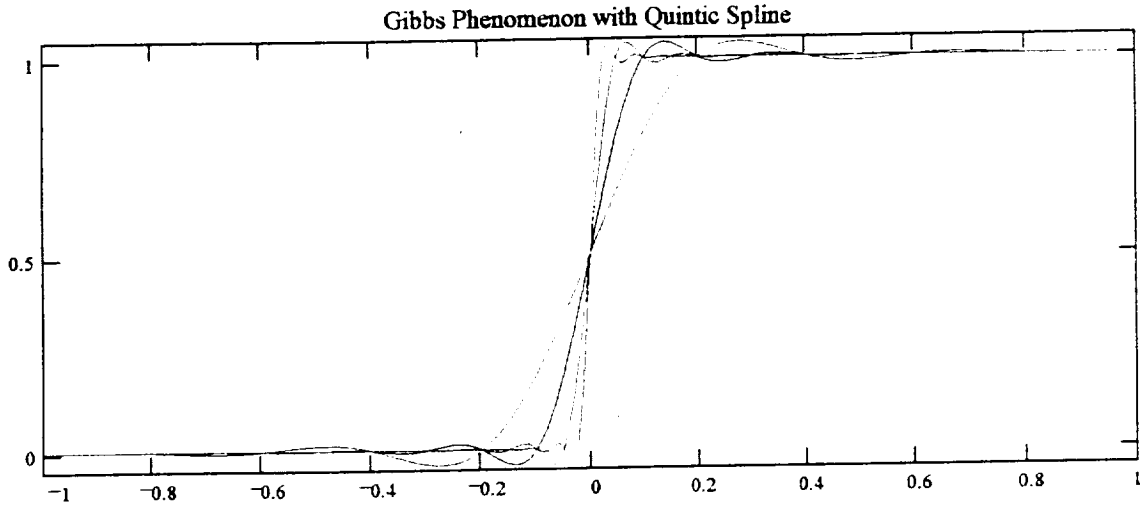
[9] McCartin, B.J. (1991): Theory of exponential splines. Journal of Approximation Theory **66**, 1-23

Figure 1.

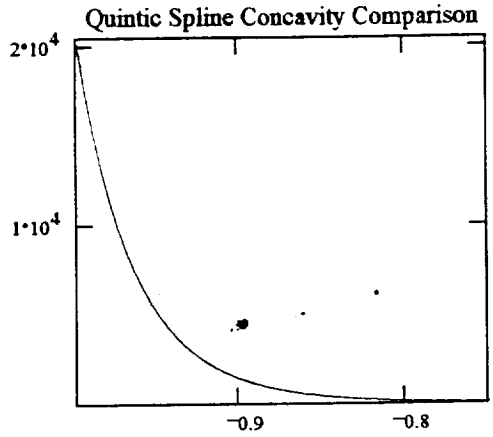
(a)



(b)



(c)



(d)

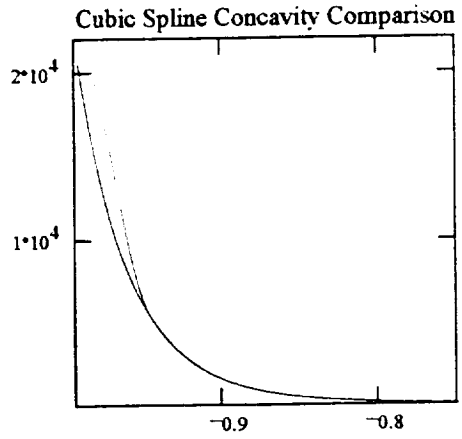
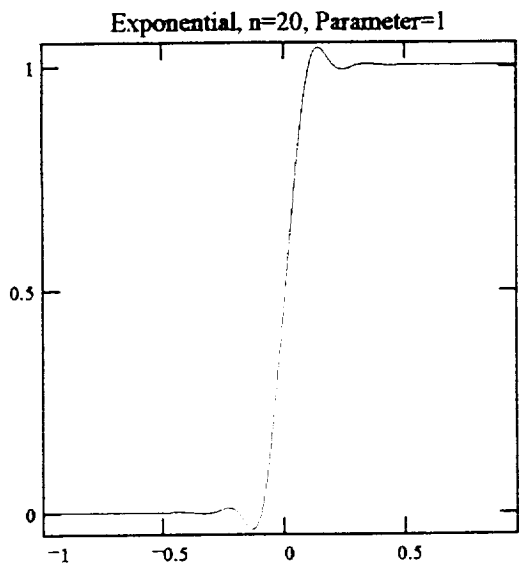
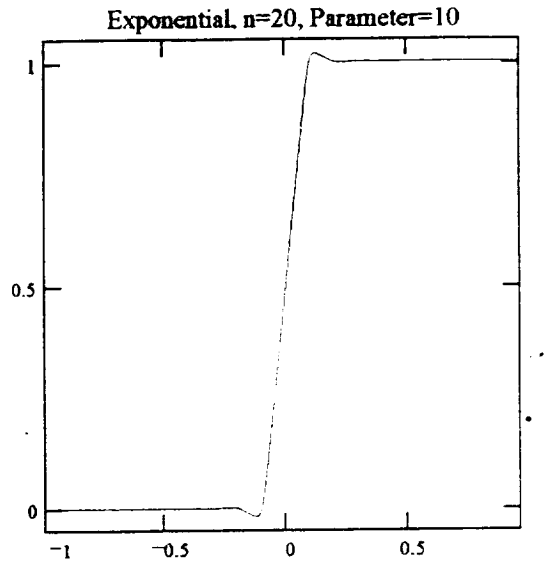


Figure 2.

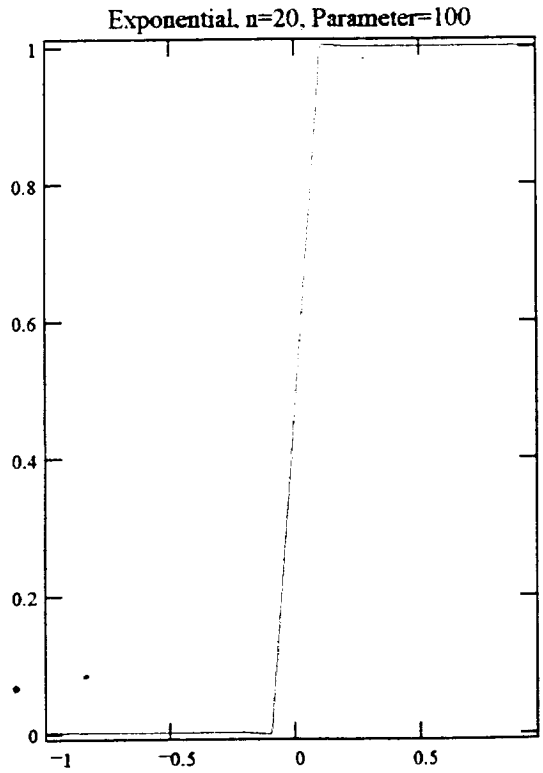
(a)



(b)



(c)



(d)

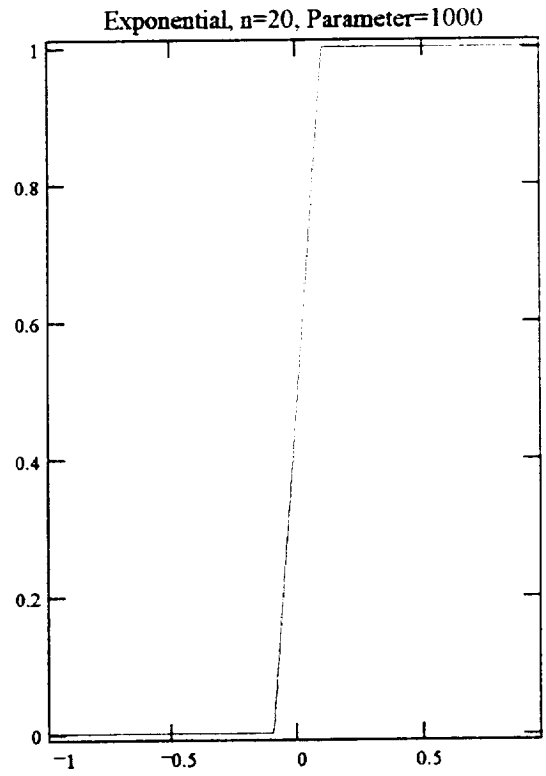
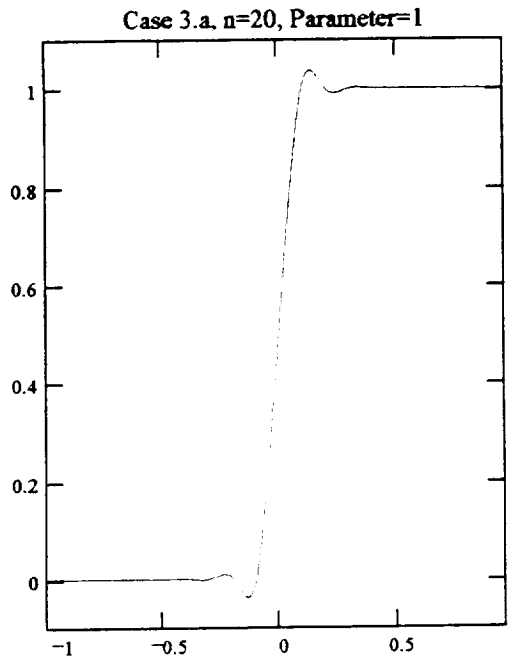
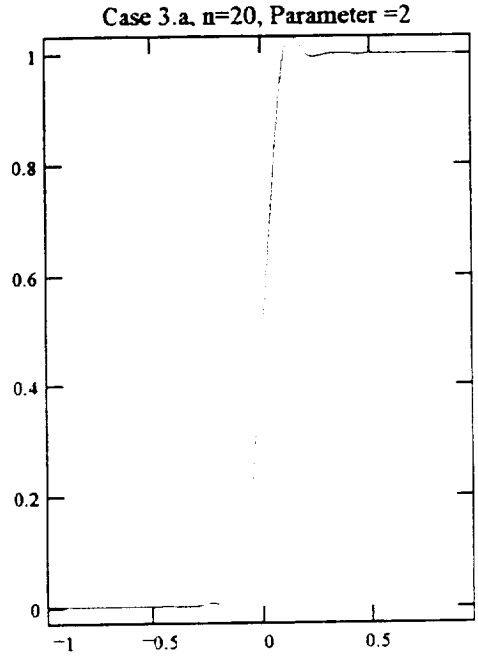


Figure 3.

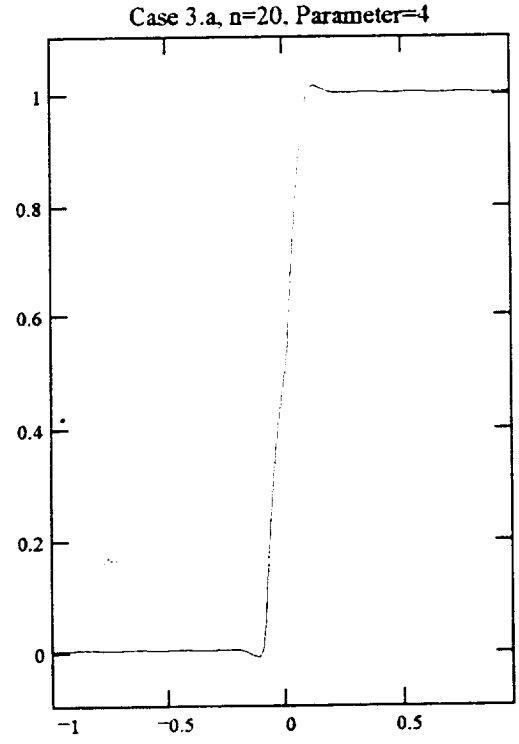
(a)



(b)



(c)



(d)

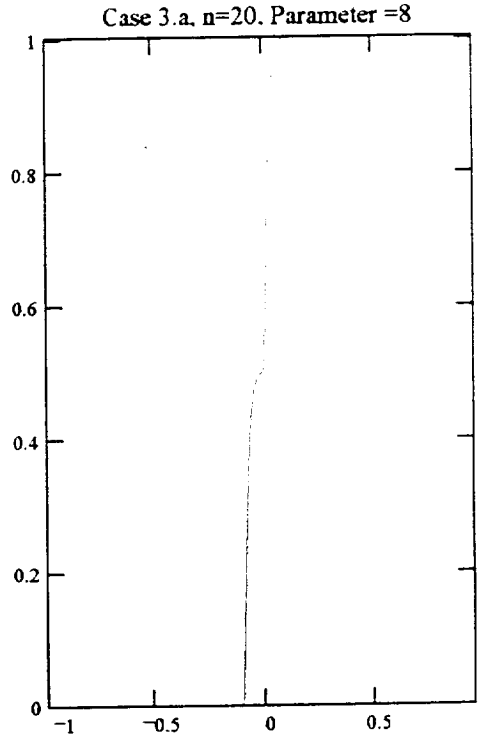
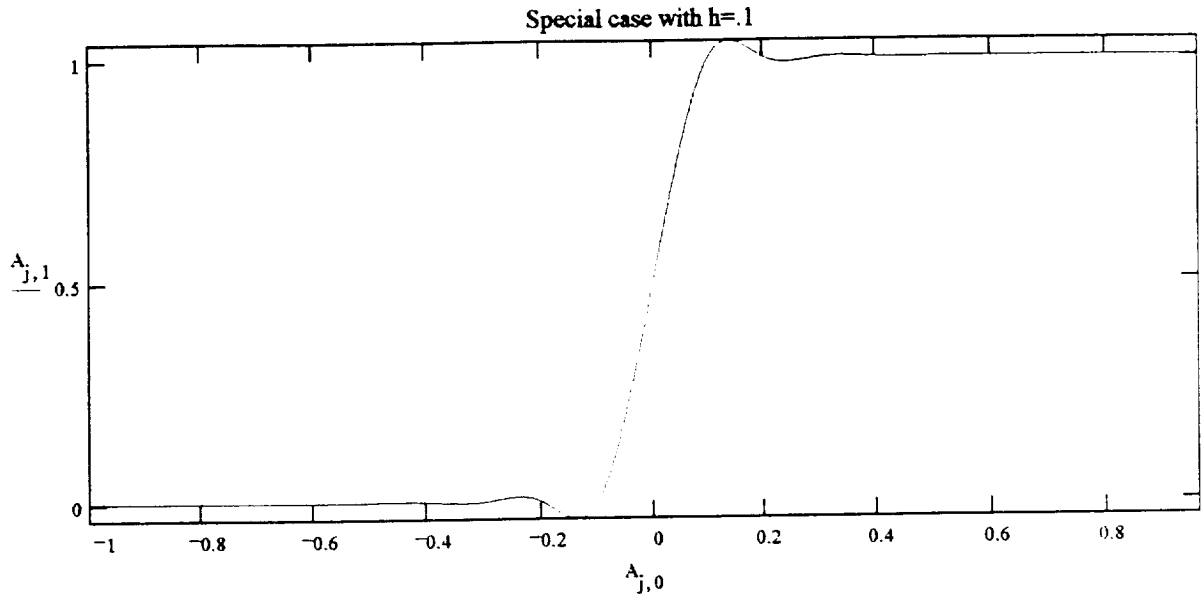
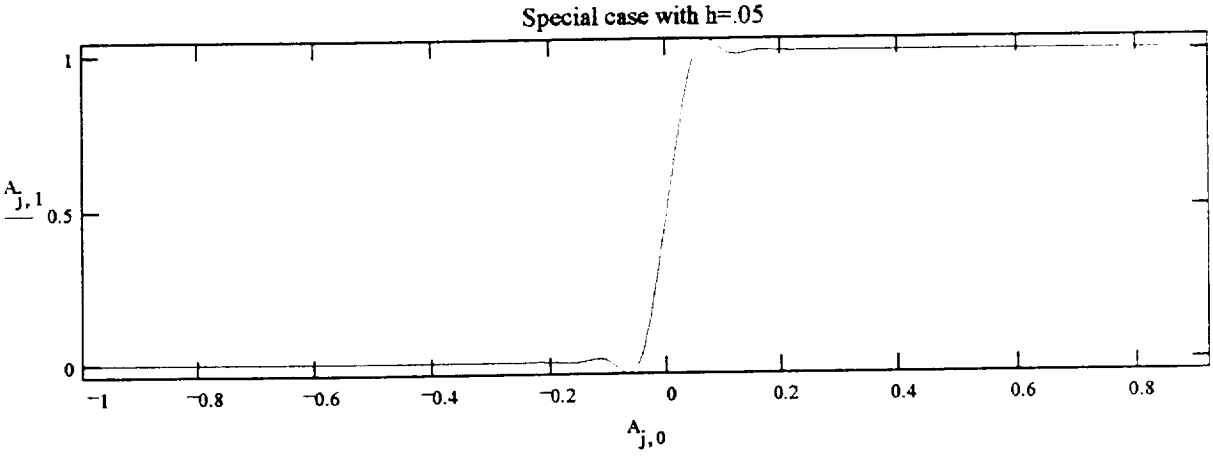


Figure 4.

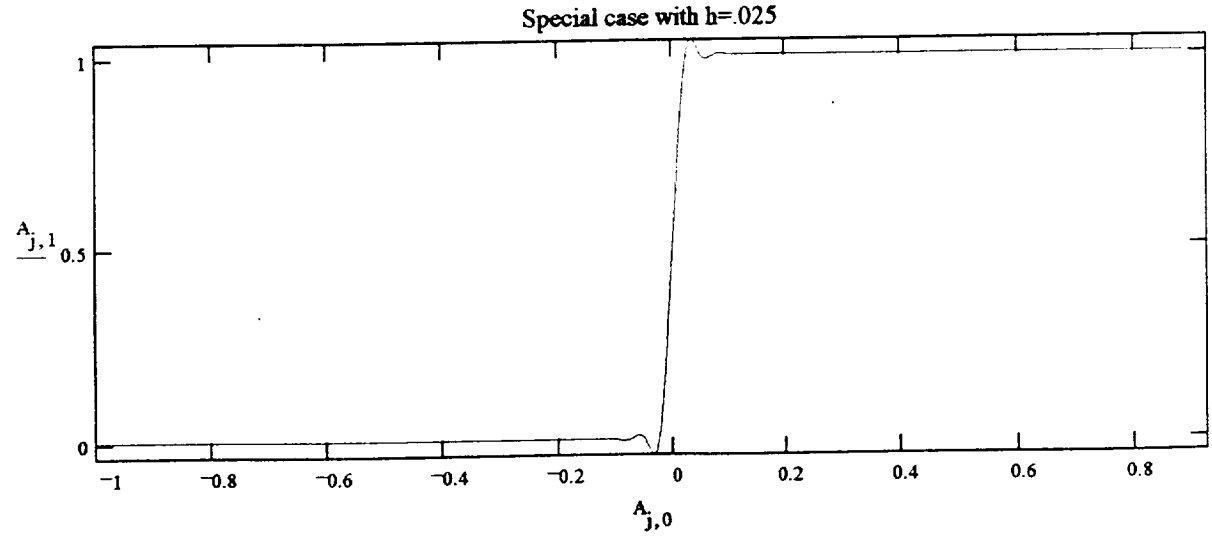
(a)


Special case with h=.1
1
0.5
0
$A_{j,1}$
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
$A_{j,0}$

(b)


Special case with h=.05
1
0.5
0
$A_{j,1}$
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
$A_{j,0}$

(c)


Special case with h=.025
1
0.5
0
$A_{j,1}$
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
$A_{j,0}$

01/10/95 12:41 ☎617 258 8682 LCS/AI MIT EDU →→→ TTU MATH DEPT ☑003

N95-25803

PRECEDING PAGE BLANK NOT FILMED

44080

P-15

A BRIEF SURVEY OF CONSTRAINED MECHANICS AND VARIATIONAL PROBLEMS IN TERMS OF DIFFERENTIAL FORMS

Robert Hermann

Ever since my graduate student days, I have been impressed and influenced by the elegance and systematization of Mechanics and Variational Calculus contained in Elie Cartan's book "Lecons sur les Invariants Integraux". In the period 1959-69, I expended considerable effort in the development of Cartan's point of view in many books and articles. In this paper (which will appear as a Chapter in "Interdisciplinay Mathematics", v. 30), I will give a quick development of some of the material in my books "Differential Geometry and the Calculus of Variations" and "Geometry, Physics and Systems".

Another purpose in developing this geometric form of the Equations of)Mechanics in this Volume is that it fits in with my strategy of investigating mechancis with 'singular' features, such as Delta Functions, Discontinuities, Shocks, etc. As I will show In Volume 30 the C-O-R constructions of Generalized Functions enable one to define 'differential forms with generalized coefficients', thus preparing the ground for the material in this Chapter serving as foundation for Mechanics with Singular Data, the Theory of Splines on nonlinear manifolds, etc. Further, when combined with the Computational Methods under development at the AI Lab of MIT by Gerry Sussman and co-workers this material will be useful in the development of Air Traffic Control methodology.

Another gaol of my work is to develop a general structure for ODE systems, to be used in both 'smooth' and 'generalized' (in the sense of Colombeau, Oberguggenberger and Rosinger) Mechanics, Control and Numerical Analysis. Since Martin, Crouch have shown that, in the linear case, Splines may be constructed from linear control system so attention will, in the future, focus on the Splines associated with Generalized Inpouts to Nonlinear Control Systems. Work of Sastry and Montgomery indicates that imporant examples of such systems will be the Left-Invariant Control Systems on Lie Groups, which have been much studied in recent years by

01/10/95 12:42 ☎617 258 8682 LCS/AI MIT EDU →→→ TTU MATH DEPT 004

researchers interested in Integrable Systems, Robotics, and Aircraft Guidance,

1. Introduction.

There has been considerable interest recently in constrained mechanics and variational problems. This is in part due to Applied interests (such as 'non-holonomic mechanics in robotics') and in other part due to the fact that several schools of 'pure' mathematics have found that this classical subject is of importance for what they are trying to do. I have made various attempts [2, 3, 4, 5, 6, 8, 11, 15, 20, 26, 27] at developing these subjects since my Lincoln lab days of the late 1950's. In this Chapter, I will sketch a Unified point of view, using Cartan's approach with differential forms. This has the advantage from the C-O-R viewpoint being developed in this Volume that the extension from 'smooth' to 'generalized' data is very systematic and algebraic. (I will only deal with the 'smooth' point of view in this Chapter; I will develop the 'generalized function' material at a later point.) The material presented briefly here about Variational Calculus and Constrained Mechanics can be found in more detail in my boooks, "Differential Geometry and the Calculus of Variations" "Lie Algebras and Quantum Mechanics", and "Geometry, Physics and Systems".

Here is the basic set-up. Suppose given the following data:

A smooth paracompact manifold X (1.1)

T(X) is its **tangent vector bundle** (1.2)

A set $\{\theta, \omega^1, ..., \omega^m\}$ of smooth 1-forms on X. (1.3)

θ is called the **action form**, $\{\omega^1, ..., \omega^m\}$ are the **constraint forms.**

Let us suppose given a curve in X:

$$\mathbf{x} = \{t \dashrightarrow x(t) \in X: a \leq t \leq b\}\}: [a, b] \dashrightarrow X \tag{1.4}$$

$$dx/dt = \mathbf{v} = \{t \dashrightarrow dx/dt(t) \in T(X): a \leq t \leq b\}\}: [a, b] \dashrightarrow T(X) \tag{1.5}$$

is its **tangent vector** or **velocity curve.**

(In this Chapter, I suppose all such curves are also smooth.)

Definition. The following real number associated to the curve 1.3 is called the **action:**

$$\alpha(\mathbf{x}) = \int_{[a, b]} \theta([dx/dt](t))dt \tag{1.6}$$

The following field of 1-covectors along the curve 1.4 is called the **force:**

$$\{t \dashrightarrow [dx/dt](t))\rfloor d\theta\} \tag{1.7}$$

1.6 and 1.7 are the basic data for both 'mechanics' and 'variational calculus'.

Now, let us deal with 'constraints':

Definition. The curve 1.4 **satisfies the constraints associated with the 1-forms** $\{\omega^1, ..., \omega^m\}$ iff. it satisfies the following set of **Pfaffian differential equations:**

$$0 = \omega^1(dx/dt) = ... = \omega^m(dx/dt) \tag{1.8}$$

I will show how the basic Equations of Mechanics can be described very compactly and elegantly in terms of this data.

2. The First Variation formula and the Cauchy characteristic curves of $d\theta$.

Keep the data of Section 1. Let us suppose that only the 'action' form θ is give, without any constraints. Let x be a smooth curve in X, given as in 1.4. Suppose that 's', $0 \leq s \leq 1$, is a deformation parameter and that $\{s \to x_s: [a, b] \to X\}$ is a smooth one-parameter family of curves in X, reducing to the given curve x at 's=0'. For $t \in [a, b]$, set:

$v(t)$ = tangent vector to the curve $\{s \to x_s(t)\}$ at 't= 0'

The field $\{t \to v(t) \in X_{x(t)}\}$ of tangent vectors is called an **infinitesimal deformation of the curve x.** Then:

$$d(\alpha(x_s))/ds\big|_{s=0} \qquad (2.1)$$

is called the **First Variation of the action function function 1.6 along the curve x pointing in the direction of the vector field v.** Using the formula 1.6 for the Action, the Cartan Family Identity: '$V(\theta) = V \rfloor d\theta + d(V \rfloor \theta)$': between a differential form and a vector field, and Integration-By-Parts, we have the **First Variation Formula:**

$$d(\alpha(x_s))/ds\big|_{s=0} = \int_{[a, b]} -[dx/dt \rfloor d\theta](v(t))dt + \theta(v)(b) - \theta(v)(a) \qquad (2.2)$$

Remark. This formula is a variant of a General Principle:

The Variational Derivative of the Action is the Force (2.3)

It also suggests the following:

Definition. A curve $\{x: t \to x(t)\}$ is called a **Cauchy characteristic curve** for the 2-form $d\theta$ iff:

$$[dx/dt] \rfloor d\theta = 0. \qquad (2.4)$$

If x satisfies 2.4, then the First Variation 2.2 vanishes for any infinitesimal deformation v which satisifies the following conditions:

$$\theta(v)(b) = \theta(v)(a) = 0 \tag{2.5}$$

Conditions 2.5 are called **Transversality Conditions.**

At this point, 'symplectic structures and foliations', 'Hamilton's and Lagrange's Equations' (for special choices of θ), etc. enter in a very natural way. See [2, 4, 6, 8, 11, 20, 27, 29].

3. The differential equations of constrained extrema and the augmented action form.

Let us now suppose that $\{\theta, \omega^1, ..., \omega^m\}$ is given, as in 1.3. Introduce Lagrange Multiplier Variables:

$$\{\lambda_1, ..., \lambda_m\} \tag{3.1}$$

Consider them as Cartesian coordinates of a copy of R^m. On $X \times R^m$, introduce the following **augmented 1-form:**

$$\theta_{aug} = \theta + \lambda_1\omega^1 + ... + \lambda_m\omega^m \tag{3.2}$$

Then,

$$d\theta_{aug} = d\theta + d\lambda_1\wedge\omega^1 + ... + d\lambda_m\wedge\omega^m + \lambda_1 d\omega^1 + ... + \lambda_m d\omega^m \tag{3.3}$$

Definition. A curve $\{t \dashrightarrow x(t)\}$ in X is an **extremal** of the constrained variational problem associated with the differential form data 1.3 if and only if there is a curve in $X \times R^m$ of the form $\{t \dashrightarrow (x(t), \lambda_1(t), ..., \lambda_m(t)\}$ which is a Cauchy characteristic curve of $d\theta_{aug}$. In other words, the 'extremas' are the images under the Cartesian projection map: $\{X \times R^m \dashrightarrow X\}$ of the Cauchy characteristic curves of $d\theta_{aug}$.

Theorem 3.1. A curve $\{t \dashrightarrow (x(t), \lambda_1(t), ..., \lambda_m(t)\}$ is a Cauchy characteristic curve for the 2-form '$d\theta_{aug}$' if and only if the following conditions are satisfied:

$$0 = \omega^1(dx/dt) = ... = \omega^m(dx/dt) \tag{3.4}$$

$$dx/dt \rfloor d\theta = -\lambda_1(t)[dx/dt] \rfloor d\omega^1 - \ldots - \lambda_m(t)[dx/dt] \rfloor d\omega^m - [d\lambda_1(t)/dt]\,\omega^1 - \ldots - [d\lambda_m(t)/dt]\,\omega^m \tag{3.5}$$

Proof. Let v be a tangent vector to the manifold $X \times R^m$. Then:

$$v \rfloor d\theta_{aug} = v \rfloor \left(d\theta + d\lambda_1 \wedge \omega^1 + \ldots + d\lambda_m \wedge \omega^m + \lambda_1 d\omega^1 + \ldots + \lambda_m d\omega^m \right)$$

$$= v \rfloor d\theta + v(\lambda_1)\omega^1 + \ldots + v(\lambda_1)\omega^1 - \omega^1(v)d\lambda_1 - \ldots - \omega^m(v)d\lambda_m + \lambda_1[v \rfloor d\omega^1] + \ldots + \lambda_m[v \rfloor d\omega^m] \tag{3.6}$$

3.6 involves one-forms on $X \times R^m$. Notice that the only terms on the right hand side of 3.6 which involves $\{d\lambda_1, \ldots d\lambda_m\}$ are the terms '$-\omega^1(v)d\lambda_1 - \ldots - \omega^m(v)d\lambda_m$'. If the tangent vector v is to be Cauchy characteristic these forms must vanish. This leads to the condition 3.4. The conditions 3.5 now follow from inserting 3.4 into the Cauchy characteristic conditions '$v \rfloor d\theta_{aug} = 0$' and using 3.6.

q.e.d.

Remark. This result expands the treatment to the 'constrained' case that Cartan gave for the 'unconstrained' variational problem in "Lecons sur les Invariants Integraux". See [2] for the connection with the traditional 'Lagrange Variational Problem' as expounded in Caratheodory's book and for the definition and properties of 'symplectic foliations' and further detail.

4. The differential equations of constrained mechanics.

There is considerable confusion in the Literature beween the Lagrange Variational Problem (or 'constrained extrema') and 'constrained' (and 'non-holonomic') mechanics'. I will now describe the latter. Suppose again given the following data:

A smooth paracompact manifold X (4.1)

T(X) is its **tangent vector bundle** (4.2)

A set $\{\theta, \omega^1, ..., \omega^m\}$ of smooth 1-forms on X. (4.3)

Definition. Let $\{x: t \dashrightarrow x(t)\}$ be a curve in X. It is said to be a **trajectory of the constrained mechanical system associated with the data 4.1-4.3** iff. the following conditions are satisfied:

$$0 = \omega^1(dx/dt) = ... = \omega^m(dx/dt) \tag{4.4}$$

There is a curve in $X \times R^m$ of the form $\{t \dashrightarrow (x(t), \mu_1(t), ..., \mu_m(t)\}$ such that:

$$[dx/dt] \rfloor d\theta = \mu_1(t)\omega^1 + ... + \mu_1(t)\ \omega^m \tag{4.5}$$

In other words, 4.4-4.5 define an ODE system whose solutions are curves in $X \times R^m$. The 'constrained mechanics trajectories' are the projections in X under the Cartesian map projection $\{X \times R^m \dashrightarrow X\}$ of the solution curves of the ODE system 4.4-4.5.

5. 'Holonomic' constraints. Equivalence of the Constrained Extremal and Mechanics equations in the 'holonomic' case.

Suppose given the following data:

A smooth paracompact manifold X (5.1)

A set $\{\theta, \omega^1, ..., \omega^m\}$ of smooth 1-forms on X. (5.2)

Indices $1 \leq a, b, ... \leq m$ (5.3)

Definition. The constraint forms $\{\omega^a\}$ are said to be **holonomic** iff. there is a matrix $\{\omega^a{}_b\}$ of 1-forms such that:

$$d\omega^a = \sum_b \omega^a{}_b \wedge \omega^b \tag{5.4}$$

Remark. Locally, condition 5.4 is equivalent to the following, more 'geometric', condition:

The Pfaffian System $\{\omega^a = 0\}$ is Frobenius Integrable (5.5)

Let us now combine conditions 5.4 and the Constrained Extremal equations 3.5. The following equations result:

$$dx/dt\rfloor d\theta = -\sum_{ab}\lambda_a(t)[dx/dt]\rfloor(\omega^a{}_b \wedge \omega^b) - \sum_a[d\lambda_a(t)/dt]\,\omega^a \qquad (5.6)$$

Rewrite this as follows:

$$dx/dt\rfloor d\theta = -\sum_{ab}\lambda_a(t)[(\omega^a{}_b(dx/dt)\,\omega^b) - \omega^b(dx/dt)\,\omega^a{}_b \qquad (5.7)$$
$$-\sum_a[d\lambda_a(t)/dt]\,\omega^a$$

The second term on the right hand side of 5.7 vanishes as a consequence of the Constraint Equations 4.4, resulting in the following:

$$[dx/dt]\rfloor d\theta = -\sum_{ab}\lambda_a(t)[(\omega^a{}_b(dx/dt)\,\omega^b) - \sum_a[d\lambda_a(t)/dt]\,\omega^a \qquad (5.7)$$

Theorem 5.1. Let 5.4 be satisfied and let the curve $\{t \dashrightarrow x(t)\}$ be a solution of the Constrained Extremal Equations. Then, $\{t \dashrightarrow x(t)\}$ is also a solution of the Constrained Mechanics Equations 4.4-4.5.
Proof. That functions $\{t \dashrightarrow \mu^a(t)\}$ exist satisfying 4.5 is evident from 5.7.
q.e.d.

Here is the converse:

Theorem 5.2. Let 5.4 be satisfied and let the curve $\{t \dashrightarrow x(t)\}$ be a solution of the Constrained Mechanics Equations 4.4-4.5. Then, $\{t \dashrightarrow x(t)\}$ is also a solution of the Constrained Extremal Equations.

Proof. We must show that the existence of functions $\{t \dashrightarrow \lambda^a(t)\}$ satisfying 5.7 is a consequence of the existence of functions $\{t \dashrightarrow \mu^a(t)\}$ satisfying 4.4-4.5. Examining the right hand side of 5.7, we see that the $\{\mu^a(t)\}$ can be obtained by solving an ODE whose coefficients depend on the $\{\lambda^a(t)\}$.

q.e.d.

6. The constrained mechanics equations in a 'Hamiltonian' form.

So far, we have been working in the context of general manifold theory. Let us specialize now to the situation which is close to the 'Hamiltonian' formalism in the traditional particle mechanics case.

Suppose given the following data:

n is an integer (6.1)

The following range of indices: (6.2)
$1 \le i, j, \ldots \le n$

$X = R^{2n+1} = R^n \times R^n \times R$ (6.3)

$\{q^i, p_i, t\}$ are Cartesian coordinates on X. (6.4)

$\{(q, p, t) \dashrightarrow H(q, p, t)\}$ is a smooth real-valued function on X, called the **Hamiltonian**. (6.5)

$$\theta = \sum_i p_i dq^i - H dt \tag{6.6}$$

$$dH = \sum_i H_i dq^i + \sum_i H^i dp_i + H_t dt, \tag{6.7}$$

where $\{H_i, H^i, H_t\}$ are the partial derivatives of the Hamiltonian function with respect to the 'canonical' coordinates 6.4.

Theorem 6.1. $d\theta = \sum_i (dq^i - H^i dt) \wedge (dp_i + H_i dt)$ (6.8)

Proof. Follows from 6.7 and 6.6, by a direct compuation, which is left to te reader.

Theorem 6.2. Let V be a smooth vector field on X. Then:

$$V \rfloor d\theta = \sum_i (V(q^i) - H^i V(t))(dp_i + H_i dt) - \sum_i (V(p_i) + H_i V(t))(dq^i - H^i dt) \quad (6.9)$$

In particular, if:

$$V(t) = 1 \quad (6.10)$$

then:

$$V \rfloor d\theta = \sum_i (V(q^i) - H^i)(dp_i + H_i dt) - \sum_i (V(p_i) + H_i)(dq^i - H^i dt) \quad (6.11)$$

Proof. Apply the operation '$V \rfloor$ ' to both sides of 6.8. 6.11 follows from substituting 6.10 into 6.9.

Theorem 6.3. Keep the hyopotheses of Theorem 6.2 and condition 6.10. Suppose further that:

$$V \rfloor d\theta = \mu(t) \sum_i a_i dq^i \quad (6.12)$$

where $\{a_i\}$ are smooth functions on X and $\{t \dashrightarrow \mu(t)\}$ is a real-valued function of 't'. Then, the following relations must be satisfied:

$$V(q^i) = H^i \quad (6.13)$$

$$V(p_i) + H_i = \mu(t) a_i \quad (6.14)$$

$$\sum_i (V(p_i) + H_i) H^i = 0 \quad (6.15.$$

Proof. 6.13-.15 results from combining 6.11 and 6.12, and comparing coefficients of independent coordinate differentials on both sides of the resulting differential form relation.

Theorem 6.4. Let V be the vector field on X defined by 6.10 and 6.12. Then, the orbit curves {t --> (q(t), p(t), t)} of V are solutions of the following ODE's:

$$dq^i/dt = \partial H/\partial p_i \tag{6.16}$$

$$dp_i/dt = -\partial H/\partial q^i + \mu(t)a_i \tag{6.17}$$

$$\sum_i a_i(p(t), q(t), t)[dq^i/dt] = 0 \tag{6.18}$$

Proof. Follows from 6.13-6.15.

Remark. Equations 6.16-6.18 form an ODE system of (2n+1) equations for the (2n+1) 'unknowns: $\{p_i(t), q^i(t), \mu(t)\}$. They are the **Hamiltonian version of the Lagrange Equations of Motion for Constrained Mechanics.** (In this case, there is only one 'constraint, namely 6.18. The case of more constraints can be handled similiarly.)

7. Final remarks about generalizations.

The material in Section 6 suggests a Generalization of material about Symplectic Manifolds, Geometric Quantization, etc. from the traditional case abstracted from Particle Mechanics (as in the work of Dirac, van Hove, Segal, Kostant, Souriau, etc) to a abstract sitaution paralleling the material developed in Section 6.

I will briefly sketch such generalizations. Instead of the 'R^{2n+1}' situation of Section 6, suppose that we are on a manifold X, with the following relation:

$$d\theta = \Omega - dH \wedge dt \tag{7.1}$$

'H' and 't' are smooth functions on X. θ and Ω are, respectively, a 1- and 2-form on X and Ω is closed. Suppose that V_H is a vector field on X such that:

$$V_H \rfloor d\theta = \mu\omega \tag{7.2}$$

$$V_H(t) = 1, \tag{7.3}$$

where 'ω' is a 1-form defining the constraints and 'μ' is a function on X. 7.1-7.3 imply:

$$V_H \rfloor \Omega - V(H)dt + dH = \mu\omega \tag{7.4}$$

This relation generalizes the duality relation between 'infinitesimal symmetries' and 'conserved functions' that plays the basic role in the 'geometric quamtization' theory of unconstrained conservative mechanical systems. I plan to study this Geometric Structure further in a later Volume.

Bibliography.

1. R. Hermann, The Differential Geometry of Foliations, II, **J. Math. and Mech.** 11 (1962), pp. 303-316.

2. R. Hermann, Some Differential Geometric Aspects of the Lagrange Variational Problem, **Illinois J. Math.** 6 (1962), pp. 634-673.

3. R. Hermann, On the Accessibiltiy Problem of Control Theory, **Proc. of the Symposium on Differential Equations,** J. Lasalle and S. Lefschetz, Eds., Colorado Springs, Academic Press, 1961.

4. R. Hermann, The Second Variation for Variational Problems in Canonical Form, **Bull. Amer. Math. Soc.** Vol 71 (1965), pp. 145-148.

5. R. Hermann, The Second Variation for Minimal Submanifolds, **J. of Math. and Mech.** 16 (1966), pp. 473-492.

6. R. Hermann, **Differential Geometry and the Calculus of Variations,** Academic Press New York 1969. Second Edition, Math Sci Press, 1977.

7. R. Hermann, Quantum Field Theories with Degenerate Lagrangians, **Phys. Rev.** 177 (1969) P. 2453.

8. R. Hermann, **Lie Algebras and Quantum Mechanics,** W. A. Benjamin New York 1971. 400 pp.

9. R. Hermann, Spectrum-generating Algebras in Classical Mechanics I and II. **J. Math. Phys.** 13. (1972) 873 - 878.

10. R. Hermann, Left Invariant Geodesics and Classical Mechanics on Manifolds, **J. Math. Phys.** 13 (1972) p. 460.

11. R. Hermann, **Geometry Physics and Systems,** Marcel Dekker New York 1973.

12. R. Hermann, (with A. Krener), Non-Linear Controllability and Observability **IEEE Trans. Aut. Cont.** AC-22 (1977), 728-740.

13. R. Hermann, **Toda Lattices, Cosymplectic Manifolds, Backlund Transformations and Kinks,** Part A and B, Math Sci Press, Brookline, MA 1977. Interdisciplinary Mathematics, vols. 15 and 18.

14. R. Hermann, **Quantum and Fermion Differential Geometry,** Part A, Math Sci Press, Brookline, Mass., 1977. Interdisciplinary Mathematics, vol. 16.

15. R. Hermann, The Differential geometric Structure of General Mechanical Systems from the Lagrangian Point of View, **J. Math. Phys.** 23 (1982), 2077-2089.

25. R. Hermann, Perturbation Theory for Nonlinear Feedback Control Systems and Goldschmidt-Spencer Integrability of Linear Partial Differential Equations, **Acta App. Math.**, 18, 17-57, 1990

26. R. Hermann, **Geometric Structures in Nonlinear Physics**, Math Si Press, 1992.

27. R. Hermann, **Constrained Mechanics and Lie Theory**, Math Sci Press, 1993,

28. R. Hermann, **Lie Theoretic Numerical Analysis, Mechanics and Differential Systems**, Math Sci Press, 1994,

29. E. Whittaker, **Analytical Dynamics**, Cambridge University Press.

16. R. Hermann, The Geometric Foundations of the Integrability Property of Differential Equations, I: Lie's "Function Groups"., **J. Math Physics**, 24, 1983, 2422-2432; part II:Mechanics on Affinely Connected Manifolds and the work of Kowalewska amd Painleve, **J. Math Physics**, 25m 1984, 778-785.

17. R. Hermann, **Topics in The Geometric Theory of Integrable Systems**, Math Sci Press, 1984.

18. R. Hermann, **Topics in Physical Geometry**, Math Sci Press, 1988

19. R. Hermann, Invariants for Feedback Equivalence and Cauchy Characteristic Multifoliations for Nonlinear Control Systems, **Acta App. Math.**, 11, 1988, 123-153.

20. R. Hermann, Differential Form Methods in the Theory of Variational Systems and Lagrangian Field Theories , **Acta App. Math.**, 12, 1988, 35-78.

21. R. Hermann, Dynamical Systems and Infinite Dimensional Lie Algebras of the "Current Algebra" or "Kac-Moody" Type, in **Mathematical Methods in Hydrodynamics and Integrability in Dynamical Systems**, AIP Conference Proceedings vol. 88, Am. Inst. of Physics, 67-72.

22. R. Hermann, Lax Representation as a "Quantization" of the "Function Groups" of Sophus Lie, **Phys. Rev. D**, 26 , 1982, 1491-1492.

23. R. Hermann, Geometric Construction and Properties of Some families of Solutions of Nonlinear Partial Differential Equations, **J. Math. Phys.**, 24, 1983, 510-521.

24. R. Hermann, Nonlinear Feedback Control and Systems of Partial Differential Equations, **Acta App. Math.**, 17, 41-94, 1989.

S2-62
44081

N95-25804

Control Theory and Splines, Applied to Signature Storage

by

Per Enqvist

September 12, 1994

Abstract

In this report the problem we are going to study, is the interpolation of a set of points in the plane with the use of control theory. We will discover how different systems generate different kinds of splines, cubic and exponential, and investigate the effect that the different systems have on the tracking problem. Actually we will see that the important parameters will be the two eigenvalues of the control matrix.

Contents

1 Introduction 3

2 Reachability 4

3 The System 7

4 Derivations 8

5 Boundary Conditions 12
5.1 Known velocities at boundary 13
5.1.1 Zero initial velocity 14
5.1.2 Derivative approximation 15
5.2 Constant velocity 16

6 Results 18
6.1 Four test curves 21
6.2 Applied on a signature 24

7 Resumé - Summary in Swedish 26

8 Programs 27
8.1 Matlab Program 27
8.2 Maple Program 41

List of Figures

1 Example of bad boundary condition. 15
2 Cusp. 18
3 Example of eigenvalues effect on signature. 19
4 Example of eigenvalues effect on control signals. 19
5 Circle 21
6 Absolute Value 21
7 Cycloid 23
8 Prolate Cycloid 23
9 Signature 25
10 Reproduction of signature 25

List of Tables

1 μ units of error for unit circle. 20
2 μ units of error for $y = |x|$. 22
3 μ units of error for cycloid. 22
4 μ units of error for prolate cycloid. 22

1 Introduction

I would like to begin by thanking my advisor Pr. Anders Lindquist for initiating contact with Texas Tech. I would also like to thank Pr. Clyde F. Martin for being my advisor at Texas Tech.

In this report we will look at a way to store signatures. We want to do this by storing only a minimal amount of points on the signature curve, and still be able to reconstruct the curve by interpolating these points. The interpolation will be performed by splines, and we will look at the common splines-problem from the control theory point of view. We can construct a trajectory of a system that passes through a specified set of points, and thus interpolate the points.

Two questions that need to be answered arise. First, when is it possible for the system to pass through the points? Second, when there are many ways to accomplish that, what sort of conditions should we demand that the system fulfill in order that we get a unique solution.

The question of when it is possible to interpolate the points will be answered in the general case in section 2 *Reachability* and for our particular system in section 3 *The System.*

An algorithm to find the solution is developed in section 4 *Derivations.*

The choice of boundary conditions is discussed in section 5 *Boundary conditions.*

In section 6 *Results* the results of tests done upon parametric curves are displayed and discussed.

A summary in Swedish is provided in section 7 *Resumé - Summary in Swedish.*

The programs I have been using are included in section 8 *Programs.* Included among the Matlab programs is an altered version of the original Matlab program quad8.

When we have answered the two questions, we have to decide what kind of system we will use for the interpolation. We can easily imagine that we would get to completely different curves if we asked a pedestrian to walk through a set of points and if we asked a cyclist to ride his bike through the same points. In the first case we would get (if we suppose that man is lazy), linear interpolation, and in the second case we would get a smooth rounded curve. This is the same as in our case where we have exponential and cubic parametric splines.

2 Reachability

In this section we will determine under what circumstances it is possible to take a time invariant linear system from a point $\mathbf{x}_0$ at time t_0 to a point $\mathbf{x}_1$ at time t_1. It is a vital property to us, because, in order to interpolate we have to be able to pass through the points. We will call a system completely reachable if it has the property that this can be done in any positive time for any two points.

This is a classical control theory question, and it is answered by the following well known theorem, which was at least implicitly discovered by Kalman.

Theorem 2.1 *Suppose that the system below is given,*

$$\begin{cases} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} \end{cases} \tag{2.1}$$

where A is $n \times n$ and B is $n \times k$. Then it is completely reachable iff $\Gamma \triangleq [B, AB, A^2B, \ldots, A^{n-1}B]$, is full rank.

In order to prove this and understand how the reachability concept can be characterized by the matrix Γ, we will have to look at the general solution of equation 2.1.

$$\mathbf{x}(t) = e^{A(t-t_0)}\mathbf{x}_0 + \int_{t_0}^{t} e^{A(t-s)}B\mathbf{u}_k(s)ds \tag{2.2}$$

In order to have the desired state $\mathbf{x}_1$ at time t_1 the following equality must be satisfied.

$$\mathbf{x}_1 = e^{A(t_1-t_0)}\mathbf{x}_0 + \int_{t_0}^{t_1} e^{A(t_1-s)}B\mathbf{u}_k(s)ds \tag{2.3}$$

The question of reachability is now easily seen to be the question of whether there are any solutions to the mapping $L : \mathcal{U} \mapsto \mathbb{R}^n$ such that

$$L\mathbf{u} \triangleq \int_{t_0}^{t_1} e^{A(t_1-s)}B\mathbf{u}(s)ds = \mathbf{x}_1 - e^{A(t_1-t_0)}\mathbf{x}_0 \triangleq \mathbf{d} \tag{2.4}$$

Since we recognize L as a linear operator, it is as always very fruitful to use a theorem from the general theory of functional analysis [4, p.250].

Theorem 2.2 *If X, Y are complete inner-product spaces and $A:X \mapsto Y$ is a linear continuous operator then*

$$\mathfrak{Im}A = \mathfrak{Im}AA^*$$

We know that $\mathbf{R}^n$ is a Hilbert space, but we have to look at what kind of space $\mathcal{U}$ is. We choose to introduce the following inner product for $\mathcal{U}$

$$(\mathrm{u}, \mathrm{v})_{\mathcal{U}} \triangleq \int_{t_0}^{t_1} \mathrm{u}(t)'\mathrm{v}(t)dt$$

and it can be checked that U becomes a Hilbert space. We know that there is a adjoint operator $L^* : \mathbb{R}^n \mapsto \mathcal{U}$ such that

$$(\mathrm{d}, L\mathrm{u})_{\mathbf{R}^n} = (L^*\mathrm{d}, \mathrm{u})_{\mathcal{U}}$$

and we get the adjoint L^* of the mapping L through this equation

$$\begin{aligned}(\mathrm{d}, L\mathrm{u})_{\mathbf{R}^n} &= \mathrm{d}' \int_{t_0}^{t_1} e^{A(t_1-s)}B\mathrm{u}(s)ds \\ &= \int_{t_0}^{t_1} \left(B'e^{A'(t_1-s)}\mathrm{d}\right)' \mathrm{u}(s)ds = (L^*\mathrm{d}, \mathrm{u})_{\mathcal{U}}\end{aligned}$$

We thus have a linear mapping $W \triangleq LL^* : \mathbb{R}^n \mapsto \mathbb{R}^n$, that is, it is actually only a matrix operator.

$$W \triangleq LL^* = \int_{t_0}^{t_1} e^{A(t_1-s)}BB'e^{A'(t_1-s)}ds$$

With only the basic knowledge about matrices we will now be able to prove the following lemma

Lemma 2.1 *Let A be $n \times n$ and B be $n \times k$. Then, for all t_0, t_1 such that $t_0 < t_1$ we have*

$$\mathfrak{Im}W(t_0, t_1) = \mathfrak{Im}\left[B, AB, A^2B, \ldots, A^{n-1}B\right]$$

Proof: We will do this by showing that $\mathfrak{Im}\Gamma \subseteq \mathfrak{Im}W$ and $\mathfrak{Im}W \subseteq \mathfrak{Im}\Gamma$.

[$\mathfrak{Im}\Gamma \subseteq \mathfrak{Im}W$]

Let $\mathbf{a} \in \mathfrak{Ker}W$, which implies that $0 = \mathbf{a}'W\mathbf{a} = \int_{t_0}^{t_1} \mathbf{a}'e^{A(t_1-s)}BB'e^{A'(t_1-s)}\mathbf{a}ds$, i.e.

$$\int_{t_0}^{t_1} \left| B'e^{A'(t_1-s)}\mathbf{a} \right| ds = 0$$

from which it follows that

$$B'e^{A'(t_1-s)}\mathbf{a} \equiv 0, \qquad \forall s \in [t_0, t_1],$$

i.e,

$$\sum_{j=0}^{\infty} \frac{1}{j!}(t_1 - s)^j B'(A')^j \mathbf{a} \equiv 0.$$

This implies that $[B, AB, A^2B, \ldots, A^{n-1}B]'\,\mathbf{a} = 0$. That is, for an arbitrary $\mathbf{a} \in \mathfrak{Ker}W$ we have $\mathbf{a} \in \mathfrak{Ker}\Gamma'$ which implies that $\mathfrak{Ker}W \subseteq \mathfrak{Ker}\Gamma'$ and by a theorem from fundamental algebra this equals $\mathfrak{Im}\Gamma \subseteq \mathfrak{Im}W$.

[$\mathfrak{Im}W \subseteq \mathfrak{Im}\Gamma$]
Suppose $\mathbf{a} \in \mathfrak{Im}W$. Then there exists a $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{a} = W\mathbf{x}$, and hence

$$\mathbf{a} = \sum_{j=0}^{\infty} A^j B \int_{t_0}^{t_1} \frac{1}{j!}(t_1 - s)^j B'e^{A'(t_1-s)}\mathbf{x}ds$$

from which it is obvious that $\mathbf{a} \in \mathfrak{Im}[B, AB, A^2B, \ldots]$ and by Cayley-Hamiltons theorem that $\mathbf{a} \in \mathfrak{Im}\Gamma$, which concludes the proof. □

The main theorem of this section will follow as a direct consequence of the lemma.

Proof:[Theorem 2.1] By lemma 2.1, $\mathbf{x}_1 - e^{A(t_1-t_0)}\mathbf{x}_0 \triangleq \mathbf{d} \in \mathbb{R}^n$ and $\mathfrak{Im}\Gamma = \mathbb{R}^n$ implies complete reachability.□

3 The System

We will consider the system with the state and dynamics given by

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}, \qquad \begin{cases} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} \end{cases} \tag{3.5}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \lambda_1 & \alpha_1 & \alpha_2 \\ 0 & 0 & 0 & 1 \\ \beta_1 & \beta_2 & 0 & \lambda_2 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.6}$$

This gives us the property

$$\mathbf{y} = C\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$$

and the system dynamics will look like

$$\begin{cases} \ddot{x} &= \lambda_1 \dot{x} + \alpha_1 y + \alpha_2 \dot{y} + u_1 \\ \ddot{y} &= \lambda_2 \dot{y} + \beta_1 x + \beta_2 \dot{x} + u_2 \end{cases} \tag{3.7}$$

And it gives us the following Γ

$$\Gamma = \begin{bmatrix} 0 & 0 & 1 & 0 & \lambda_1 & \alpha_2 & \Gamma_{1,7} & \Gamma_{1,8} \\ 1 & 0 & \lambda_1 & \alpha_2 & \alpha_2\beta_2 + \lambda_1^2 & \alpha_1 + \alpha_2(\lambda_1 + \lambda_2) & \Gamma_{2,7} & \Gamma_{2,8} \\ 0 & 0 & 0 & 1 & \beta_2 & \lambda_2 & \Gamma_{3,7} & \Gamma_{3,8} \\ 0 & 1 & \beta_2 & \lambda_2 & \beta_1 + \beta_2(\lambda_1 + \lambda_2) & \alpha_2\beta_2 + \lambda_2^2 & \Gamma_{4,7} & \Gamma_{4,8} \end{bmatrix}$$

where

$$\begin{cases} \Gamma_{1,7} &= \alpha_2\beta_2 + \lambda_1^2 \\ \Gamma_{1,8} &= \alpha_1 + \alpha_2(\lambda_1 + \lambda_2) \\ \Gamma_{2,7} &= \alpha_1\beta_2 + \alpha_2\beta_1 + \alpha_2\beta_2(2\lambda_1 + \lambda_2) + \lambda_1^3 \\ \Gamma_{2,8} &= \alpha_2^2\beta_2 + \alpha_1(\lambda_1 + \lambda_2) + \alpha_2\lambda_1^2 + \alpha_2\lambda_1\lambda_2 + \alpha_2\lambda_2^2 \\ \Gamma_{3,7} &= \beta_1 + \beta_2(\lambda_1 + \lambda_2) \\ \Gamma_{3,8} &= \alpha_2\beta_2 + \lambda_2^2 \\ \Gamma_{4,7} &= \alpha_2\beta_2^2 + \beta_1(\lambda_1 + \lambda_2) + \beta_2\lambda_1^2 + \beta_2\lambda_1\lambda_2 + \beta_2\lambda_2^2 \\ \Gamma_{4,8} &= \alpha_1\beta_2 + \alpha_2\beta_1 + \alpha_2\beta_2(\lambda_1 + 2\lambda_2) + \lambda_2^3 \end{cases}$$

As Γ is easily seen to have full row rank, by simply looking at the first four columns. The class of systems we are going to consider in this article will all have the desired property of complete reachability, by Theorem 2.1.

4 Derivations

Given a set of points in the plane $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ and the corresponding time points $\{t_0, t_1, \ldots, t_n\}$ we would like to find the control functions $\{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{n-1}\}$ that takes the system through the points at the specified times.

Let's study the control $\mathbf{u}_k : \begin{pmatrix} \mathbf{x}_k \\ t_k \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{x}_{k+1} \\ t_{k+1} \end{pmatrix}$

As $t \in [t_k, t_{k+1}]$ the state of the system will be

$$\mathbf{x}(t) = e^{A(t-t_k)}\mathbf{x}_k + \int_{t_k}^{t} e^{A(t-s)}B\mathbf{u}_k(s)ds \tag{4.8}$$

and as we want the state of the system to be $\mathbf{x}_{k+1}$ at time t_{k+1} we get the following condition.

$$\mathbf{x}_{k+1} = e^{A(t_{k+1}-t_k)}\mathbf{x}_k + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-s)}B\mathbf{u}_k(s)ds \tag{4.9}$$

The solution $\mathbf{u}_k$ to equation 4.9 that minimizes the norm of the control signal is then given by

$$\mathbf{u}_k(t) = B'e^{-A't}\left(\int_{t_k}^{t_{k+1}} e^{-As}BB'e^{-A's}ds\right)^{-1}\left(e^{-At_{k+1}}\mathbf{x}_{k+1} - e^{-At_k}\mathbf{x}_k\right) \quad (4.10)$$

The control would be specified completely by equation 4.10 if we knew the whole state-vector at each interpolation point. We know all (x_k, y_k) but we do not know the $(\dot{x}_k, \dot{y}_k)$. To determine the $2(n+1)$ unknowns we have to apply some conditions on the solution, and our first choice will be to require that the control is continuous,

Assumption 4.1

$$\mathbf{u}_k(t_{k+1}) = \mathbf{u}_{k+1}(t_{k+1}), \qquad k = 0 \ldots n-2$$

This will give us $2(n-1)$ conditions and will leave only four unknowns. We will apply the additional conditions on the boundary and we will have to come back to this in section 5 *Boundary Conditions.*

In many applications it is just the shape of the trajectory that matters, and not the velocity that the system tracks the trajectory with. In these cases it makes it much easier to assume that the time between each point is a constant.

Assumption 4.2 *Let* $t_{k+1} - t_k = h$.

Assumption 4.2 can be used to simplify the integral in equation 4.10.

$$\int_{t_k}^{t_{k+1}} e^{-As}BB'e^{-A's}ds = \{\tau = s - t_k\} =$$

$$e^{-At_k}\underbrace{\int_0^h e^{-A\tau}BB'e^{-A'\tau}d\tau}_{matrixconstant}e^{-A't_k}$$

Definition 4.1

$$M \triangleq \int_0^h e^{-A\tau}BB'e^{-A'\tau}d\tau$$

We can now rewrite expression 4.10 as

$$\mathbf{u}_k(t) = B' e^{-A'(t-t_k)} M^{-1} (e^{-Ah}\mathbf{x}_{k+1} - \mathbf{x}_k) \tag{4.11}$$

Using this expression we will now investigate how the continuity condition in assumption 4.1 looks.

$$\mathbf{u}_k(t_{k+1}) = B' e^{-A'h} M^{-1} (e^{-Ah}\mathbf{x}_{k+1} - \mathbf{x}_k) \;=$$

$$B' M^{-1} (e^{-Ah}\mathbf{x}_{k+2} - \mathbf{x}_{k+1}) \;=\; \mathbf{u}_{k+1}(t_{k+1})$$

We can rewrite this condition using

Definition 4.2

$$Z \triangleq M^{-1} e^{-Ah}$$

$$W \triangleq e^{-A'h} M^{-1} e^{-Ah} + M^{-1}$$

giving the equation the simple form

$$B'(Z'\mathbf{x}_k - W\mathbf{x}_{k+1} + Z\mathbf{x}_{k+2}) = 0, \qquad k = 0 \ldots n-2 \tag{4.12}$$

In block diagonal form

$$B' \begin{bmatrix} Z' & -W & Z & \ldots & 0 & 0 & 0 \\ 0 & Z' & -W & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -W & Z & 0 \\ 0 & 0 & 0 & \ldots & Z' & -W & Z \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n-2} \\ \mathbf{x}_{n-1} \\ \mathbf{x}_n \end{bmatrix} = 0 \tag{4.13}$$

Now, we have to look at what the unknowns are. The vector in equation 4.13 is made up of subvectors

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix}$$

consisting of two knowns and two unknowns. By partitioning the submatrixes W, Z as

$$W = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ w_{\cdot 1} & w_{\cdot 2} & w_{\cdot 3} & w_{\cdot 4} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}, Z = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ z_{\cdot 1} & z_{\cdot 2} & z_{\cdot 3} & z_{\cdot 4} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \cdots & z_{1\cdot} & \cdots \\ \cdots & z_{2\cdot} & \cdots \\ \cdots & z_{3\cdot} & \cdots \\ \cdots & z_{4\cdot} & \cdots \end{bmatrix}$$

and using the notations given in the following the definition, we can keep the unknowns on the left side and move the position coordinates over to the right hand side.

Definition 4.3

$$\begin{aligned}
\mathrm{x}^{pos} &= \begin{bmatrix} x \\ y \end{bmatrix} \\
\mathrm{x}^{vel} &= \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \\
W_l^B &= B' \begin{bmatrix} w_{\cdot 2} & w_{\cdot 4} \end{bmatrix} = \begin{bmatrix} w_{22} & w_{24} \\ w_{42} & w_{44} \end{bmatrix} \\
W_r^B &= B' \begin{bmatrix} w_{\cdot 1} & w_{\cdot 3} \end{bmatrix} = \begin{bmatrix} w_{21} & w_{23} \\ w_{41} & w_{43} \end{bmatrix} \\
Z_{lu}^B &= B' \begin{bmatrix} z_{\cdot 2} & z_{\cdot 4} \end{bmatrix} = \begin{bmatrix} z_{22} & z_{24} \\ z_{42} & z_{44} \end{bmatrix} \\
Z_{ru}^B &= B' \begin{bmatrix} z_{\cdot 1} & z_{\cdot 3} \end{bmatrix} = \begin{bmatrix} z_{21} & z_{23} \\ z_{41} & z_{43} \end{bmatrix} \\
Z_{ll}^B &= B' \begin{bmatrix} z_{2\cdot} \\ z_{4\cdot} \end{bmatrix}' = \begin{bmatrix} z_{22} & z_{42} \\ z_{24} & z_{44} \end{bmatrix} \\
Z_{rl}^B &= B' \begin{bmatrix} z_{1\cdot} \\ z_{3\cdot} \end{bmatrix}' = \begin{bmatrix} z_{12} & z_{32} \\ z_{14} & z_{34} \end{bmatrix}
\end{aligned}$$

And we get the system

$$\begin{bmatrix} Z_{ll}^B & -W_l^B & Z_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & Z_{ll}^B & -W_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -W_l^B & Z_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & Z_{ll}^B & -W_l^B & Z_{lu}^B \end{bmatrix} \begin{bmatrix} \mathrm{x}_0^{vel} \\ \mathrm{x}_1^{vel} \\ \mathrm{x}_2^{vel} \\ \vdots \\ \mathrm{x}_{n-2}^{vel} \\ \mathrm{x}_{n-1}^{vel} \\ \mathrm{x}_n^{vel} \end{bmatrix} =$$

$$- \begin{bmatrix} Z_{rl}^B & -W_r^B & Z_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & Z_{rl}^B & -W_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -W_r^B & Z_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & Z_{rl}^B & -W_r^B & Z_{ru}^B \end{bmatrix} \begin{bmatrix} \mathrm{x}_0^{pos} \\ \mathrm{x}_1^{pos} \\ \mathrm{x}_2^{pos} \\ \vdots \\ \mathrm{x}_{n-2}^{pos} \\ \mathrm{x}_{n-1}^{pos} \\ \mathrm{x}_n^{pos} \end{bmatrix}$$

As we evaluate the right hand side, we get a constant vector. Depending on what kind of boundary conditions we choose to use, the derivations differ from here. We will deal with the most common cases, each case in turn, beginning in the next chapter.

5 Boundary Conditions

In order to get a unique solution to our problem, we had to apply the continuity condition and the boundary conditions. The continuity condition is a rather natural condition, but the boundary conditions have to be studied more extensively. The four most common choices of boundary conditions in the one dimensional case according to [2] are

1. Zero velocity at the first and at the last point.
2. Specified starting and ending direction.
3. Natural boundary conditions, $y'' = 0$.
4. Periodical conditions.

We will look at how the two dimensional equivalent of choice number 1 and 2 affect the curves, and for which the same derivation is valid.

Item 3:s two dimensional equivalent, $\dot{x} = \dot{y} = 0$, requires a derivation and will be studied in subsection 5.2.

We will avoid dealing with condition 4 as it only complicates the calculations, and would only be natural and interesting if we were to write the same word twice, connected with itself as RogerRoger.

Because the effect of the boundary conditions are similar at both boundaries we will restrict ourselves to only talking about the starting point.

5.1 Known velocities at boundary

We have assumed that we also know $\mathbf{x}_0^{vel}$ and $\mathbf{x}_n^{vel}$. Moving these over to the right hand side, we get a block diagonal matrix system to solve.

$$\begin{bmatrix} -W_l^B & Z_{lu}^B & \dots & 0 & 0 \\ Z_{ll}^B & -W_l^B & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -W_l^B & Z_{lu}^B \\ 0 & 0 & \dots & Z_{ll}^B & -W_l^B \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^{vel} \\ \mathbf{x}_2^{vel} \\ \vdots \\ \mathbf{x}_{n-2}^{vel} \\ \mathbf{x}_{n-1}^{vel} \end{bmatrix} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \vdots \\ \Omega_{n-2} \\ \Omega_{n-1} \end{bmatrix} \tag{5.14}$$

Where

$$\begin{aligned} \Omega_1 &= -Z_{rl}^B \mathbf{x}_0^{pos} + W_r^B \mathbf{x}_1^{pos} - Z_{ru}^B \mathbf{x}_2^{pos} - Z_{ll}^B \mathbf{x}_0^{vel} \\ \Omega_k &= -Z_{rl}^B \mathbf{x}_{k-1}^{pos} + W_r^B \mathbf{x}_k^{pos} - Z_{ru}^B \mathbf{x}_{k+1}^{pos} \\ \Omega_{n-1} &= -Z_{rl}^B \mathbf{x}_{n-2}^{pos} + W_r^B \mathbf{x}_{n-1}^{pos} - Z_{ru}^B \mathbf{x}_n^{pos} - Z_{lu}^B \mathbf{x}_n^{vel} \end{aligned}$$

This can easily be solved, and with a linear increase of time. Having solved the system above we now know all the states of the system in the interpolation points. We will now use equation 4.11 for the control, and insert it into equation 4.8 to get the trajectory.

$$\begin{aligned} \mathbf{x}(t) &= e^{A(t-t_k)} \left(\mathbf{x}_k + \int_0^{t-t_k} e^{-A\tau} B B' e^{-A'\tau} d\tau \, M^{-1} (e^{-Ah} \mathbf{x}_{k+1} - \mathbf{x}_k) \right) \\ \text{for } k &= 0 \dots n-1 \\ t &\in [t_k, t_{k+1}] \end{aligned} \tag{5.15}$$

As we can see from equation 5.15 the fundamental matrix and the integral is the same for all k and only has to be evaluated for $t - t_k$ between 0 and h, once and for all.

5.1.1 Zero initial velocity

We can choose to set

Assumption 5.1

$$\begin{aligned}(\dot{x}_0, \dot{y}_0) &= 0 \\ (\dot{x}_n, \dot{y}_n) &= 0\end{aligned}$$

With this condition, we will let the system start up in whatever direction that minimizes the energy norm of the control signal and takes the system to the second point.

As we know from one dimensional control theory, a system with a transfer function with zeros in the numerator will start off in the opposite direction to where it is going. Such undesired properties should certainly be avoided in our tracking problem. In the case where $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 0$, the states x and y are independent, yielding two one dimensional transfer functions. It is easily seen to have no zeros, which is good.

Otherwise, we will have to look at the two dimensional transfer function given below:

$$T(s) = \frac{1}{s^4 - (\lambda_1 + \lambda_2)s^3 + (\lambda_1\lambda_2 - \alpha_2\beta_2)s^2 - (\alpha_1\beta_2 + \alpha_2\beta_1)s - \alpha_1\beta_1} \times \times \begin{bmatrix} s(s-\lambda_2) & \alpha_1 + \alpha_2 s \\ \beta_1 + \beta_2 s & s(s-\lambda_1) \end{bmatrix} \tag{5.16}$$

This is a bit more tricky, and we will have to find the Q and D of least degree that is a solution to the equation

$$T(s) = Q(s)D(s)^{-1} \tag{5.17}$$

and satisfies

$$X(s)Q(s) + Y(s)D(s) = I \tag{5.18}$$

It is easy to verify that the choice

$$Q(s) = I, \quad D(s) = \begin{bmatrix} s(s-\lambda_1) & -(\alpha_1 + \alpha_2 s) \\ -(\beta_1 + \beta_2 s) & s(s-\lambda_2) \end{bmatrix}.$$

is a solution to equation 5.17, and there exists $X(s)$ and $Y(s)$ so that equation 5.18 is satisfied. From $Q(s)$ we can see that the system has no zeros. The zero initial conditions should thus not give us any problems.

5.1.2 Derivative approximation

Instead of setting the velocity equal to zero, another alternative is of course to specify a starting velocity. However, this requires that we make a good choice, to avoid situations as exemplified below. Using a bad direction and a high velocity boundary condition on $y = x^2$, we get the graph of figure 1. Even as we are using n=40 to reproduce the curve, the bad boundary conditions are still ruining the tracking.

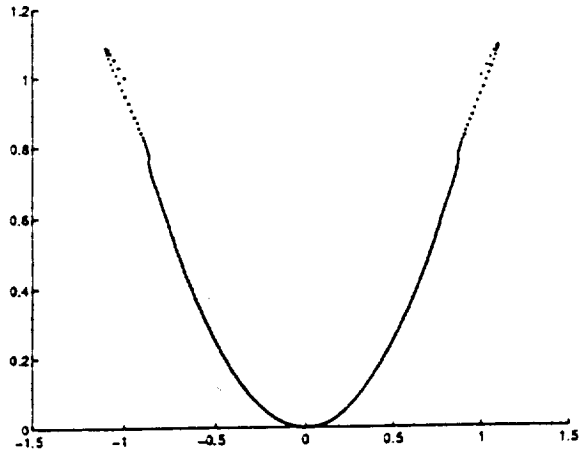


Figure 1: Trajectory of system tracking $y = x^2$, with badly specified starting direction

As discussed in [3, p.86], the fact is that when we set the boundary conditions in the parametric case, we do not only specify a direction, but also the speed in that direction. The greater the speed, the greater impact

the boundary conditions will have on the solution. We are thus forced to make a good choice, and we would like the choice to get better the more interpolation points we are using. A simple choice that satisfies this is

Assumption 5.2

$$\begin{aligned}(\dot{x}_0, \dot{y}_0) &= \frac{x_1 - x_0}{h} \\ (\dot{x}_n, \dot{y}_n) &= \frac{x_n - x_{n-1}}{h}\end{aligned}$$

which imply that we will set off from the first point in the direction of the second point, and arrive at the last point in the direction from the next last point.

Another way of deciding the initial velocity would be to use the same technic as in Bezier curves and choose the settings graphically. This would probably be the best way to get the desired properties of the signatures. As discussed in [3] the choice of boundary conditions will affect the whole curve, and the solving of the blockdiagonal system must be done over from scratch, making this method a bit slow. If we are going to do this only once to store a signature it does not matter. What matters with this method is that it adds four more parameters to be stored, and we could probably get equally good results just by increasing the number of interpolation points by two.

5.2 Constant velocity

Suppose we want to use the boundary conditions

Assumption 5.3

$$\begin{aligned}(\ddot{x}_0, \ddot{y}_0) &= 0 \\ (\ddot{x}_n, \ddot{y}_n) &= 0\end{aligned}$$

This will let the initial direction and constant velocity of the system be decided so that the control energy is minimized. Using the system dynamics equations 3.7, we get the equation system below.

$$\begin{bmatrix} \lambda_1 & \alpha_2 \\ \beta_2 & \lambda_2 \end{bmatrix} \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \end{bmatrix} + \mathrm{u}_0(0) = - \begin{bmatrix} \alpha_1 y_0 \\ \beta_1 x_0 \end{bmatrix} \tag{5.19}$$

where

$$\begin{aligned} \mathrm{u}_0(0) &= B' M^{-1}(e^{-Ah}\mathrm{x}_1 - \mathrm{x}_0) && (5.20) \\ &= B' Z \mathrm{x}_1 - B' M^{-1} \mathrm{x}_0 && (5.21) \end{aligned}$$

Now, by using partitioned matrixes as in section 4 and the following definition,

Definition 5.1 $$U_{lu} \triangleq \begin{bmatrix} m_{22}^{-1} & m_{24}^{-1} \\ m_{42}^{-1} & m_{44}^{-1} \end{bmatrix}, \qquad U_{ru} \triangleq \begin{bmatrix} m_{21}^{-1} & m_{41}^{-1} \\ m_{23}^{-1} & m_{43}^{-1} \end{bmatrix}$$

we get the system

$$\begin{aligned} &\left(\begin{bmatrix} \lambda_1 & \alpha_2 \\ \beta_2 & \lambda_2 \end{bmatrix} - U_{lu} \right) \mathrm{x}_0^{vel} + Z_{lu}^B \mathrm{x}_1^{vel} = && (5.22) \\ &= \left(U_{ru} - \begin{bmatrix} 0 & \alpha_1 \\ \beta_1 & 0 \end{bmatrix} \right) \mathrm{x}_0^{pos} - Z_{ru}^B \mathrm{x}_1^{pos} \end{aligned}$$

In a similar way we get the equation at the other boundary.

$$\begin{aligned} &\left(\begin{bmatrix} \lambda_1 & \alpha_2 \\ \beta_2 & \lambda_2 \end{bmatrix} + W_l^B - U_{lu} \right) \mathrm{x}_n^{vel} - Z_{ll}^B \mathrm{x}_{n-1}^{vel} = && (5.23) \\ &= \left(U_{ru} - W_r^B - \begin{bmatrix} 0 & \alpha_1 \\ \beta_1 & 0 \end{bmatrix} \right) \mathrm{x}_n^{pos} + Z_{rl}^B \mathrm{x}_{n-1}^{pos} \end{aligned}$$

Adding these two equations to equation 4.14 yields a blockdiagonal system to solve. This system is two blocks bigger than the one in section 5.1, but it can also be solved with a linear increase of time. And once it is solved, we can still use equation 5.15.

A comparison between the three boundary conditions, BC=1 zero initial velocity, BC=2 derivative approximation, and BC=3 zero initial acceleration is made in section 6.

6 Results

The first tests with the algorithm were done with matrices A with both eigenvalues equal to zero, $\lambda_1 = 0, \lambda_2 = 0$, and $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 0$. This produces cubic parametric splines, and makes the calculation of the fundamental matrix easy. The cubic splines produce smooth curves, but were also able to reconstruct a cusp much better than you would have guessed at first, as shown in figure 2.

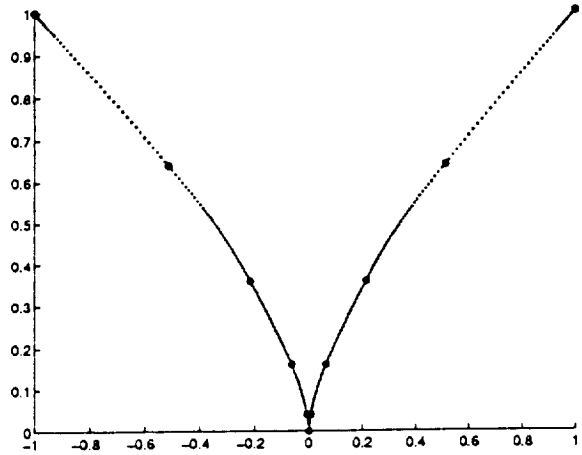


Figure 2: Reconstruction of $y = x^{\frac{2}{3}}$ with cubic parametric splines where n=10.

If we look at the function $y = x^{\frac{2}{3}}$ we know that this function describes a cusp. But if we parameterized it like $x = t^3, y = t^2$ we see that both x and y are smooth cubic functions of t, so it is not very remarkable that it can be reconstructed well.

When we used A-matrices with nonzero eigenvalues, and decoupled x and y coordinates, i.e. $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 0$, we were able to generate exponential parametric splines with the basis functions $1, \lambda_1 t, e^{\lambda_1 t}, e^{-\lambda_1 t}$ and $1, \lambda_2 t, e^{\lambda_2 t}, e^{-\lambda_2 t}$ for the x and y coordinates.

The result of taking big eigenvalues is almost linear interpolation, which can be good for certain applications, but not if it is to be used for storing signatures. It is quite obvious that the roundness of a persons signature is one important factor of it's characteristics. Therefore, it's vital that one of

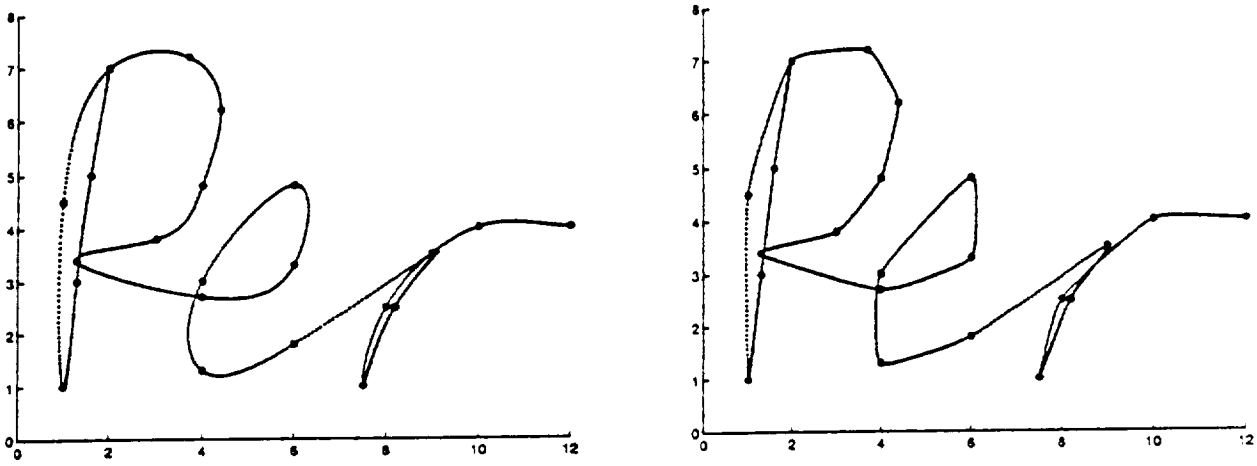


Figure 3: Graph of signature Per, reproduced with $\lambda_1 = 1, \lambda_2 = 1$ in the left figure, and $\lambda_1 = 100, \lambda_2 = 100$ in the right

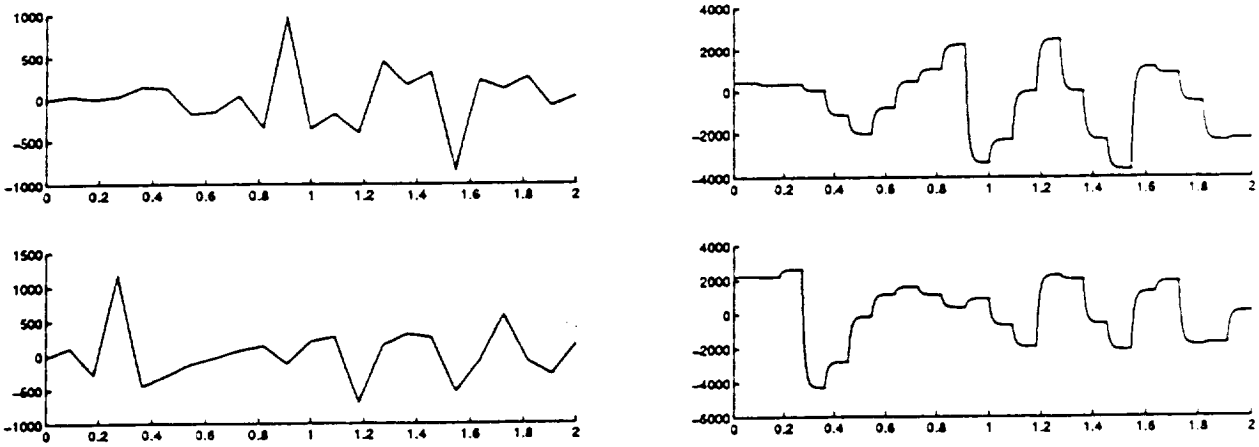


Figure 4: Graph of control signals in figure above.

data stored on the signature is the eigenvalues of the A-matrix, as can be shown in figure 3.

Figure 4 shows the corresponding control signals. In the $\lambda_1 = 1, \lambda_2 = 1$ case the linear part of the control is dominating, and in the $\lambda_1 = 100, \lambda_2 = 100$ case the exponential part is dominating. It is also evident that the magnitude of the control signals is greater in the case of larger eigenvalues, but that is not a problem for our fictitious system.

In the case of nonzero $\alpha_1, \alpha_2, \beta_1, \beta_2$ we get a coupling between the x and y coordinates. This will give us very complicated basis functions, like polynomials times exponentials times sine- and cosine-functions.

As with all approximation methods one should always investigate how big the errors are. To do this we had to somehow determine the distance between the original curve and the interpolating one. The tests were performed on known parametric curves, so we had an explicit expression for the points on the original curve. We had to try to find the nearest point on the original curve to the point on the trajectory. This was solved numerically with the "Golden Intersection algorithm" . For the method to work we have to assume two things :

1. The section of the original curve between the two interpolation points nearest in time is convex.

2. The point on the original curve nearest the point on the interpolating curve lies on the section in item 1.

As an error estimate I have calculated the distance between a number of points on the reproduced curve and the original curve and divided with the number of points. We have applied this error estimate method on four different curves and with different number of interpolation points and 40 points between each of these.

points	$BC = 1$	$BC = 2$	$BC = 3$
$n = 10$	6671.1	5821.1	4515.9
$n = 20$	1028.4	765.6	502.1
$n = 100$	8.8	5.9	3.5

Table 1: μ units of error for unit circle.

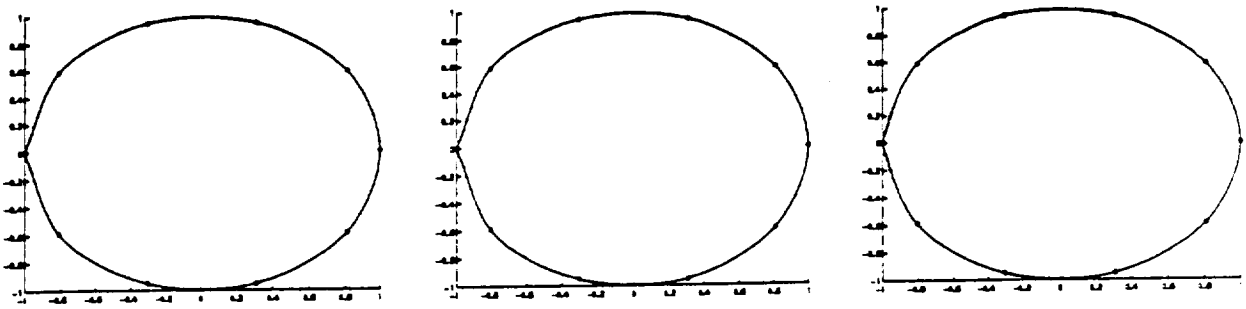


Figure 5: Graph of circle, reproduced with n=10 and $\lambda_1 = 10, \lambda_2 = 10$, BC=1,2,3

6.1 Four test curves

The first curve we tested was the unit circle. This very round curve was tracked best by the cubic splines, but the exponential splines did a good job too, as can be seen in figure 5 and table 1. We can also see that the error was smallest in the case of zero initial acceleration boundary conditions.

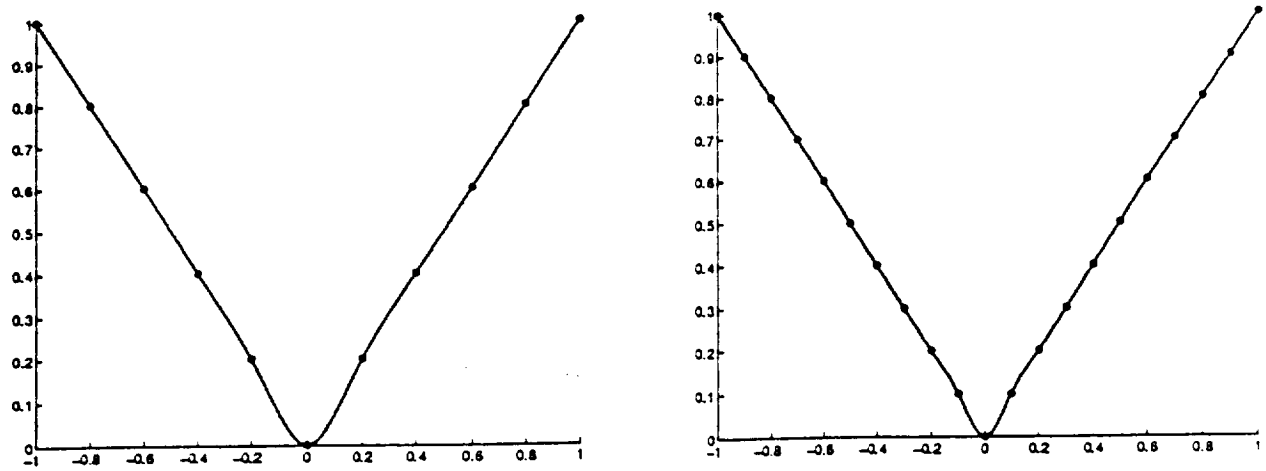


Figure 6: Graph of $y = |x|$, reproduced with n=10, n=20 and $\lambda_1 = 10, \lambda_2 = 10$, BC=2

Next, we looked at a curve with the opposite properties, linear and non-differentiable, $y = |x|$. The error is mainly located between the two points

points	$BC = 1$	$BC = 2$	$BC = 3$
$n = 10$	3450.3	3446.8	3450.3
$n = 20$	972.6	972.6	972.6
$n = 100$	40.7	40.7	40.7

Table 2: μ units of error for $y = |x|$.

next to the non-differentiable point. As could be guessed the tracking of the curve $y = |x|$ got better the bigger the eigenvalues of the A matrix were, the error was reduced to 142 μ units for $\lambda_1 = \lambda_2 = 500$ in the case of $BC = 1$ and $n = 10$, but for bigger values the numerical calculations failed. Using negative eigenvalues gives the same error as the positive, which can be expected since we have a symmetric curve and time interval and by looking at the basis functions. The results with different boundary conditions were very much alike, as seen in table 2.
This was the only case were BC 3 did not give us the smallest error.

Can we get a smaller error with any choice of the coupling parameters? Yes, for example by choosing $\alpha_1 = -\alpha_2 = \beta_1 = -\beta_2 = 10$, we get the error 3249 μ units for n=10. Choosing these parameters could thus be a way to reduce the error, but by using $n = 12$ instead, we got the error 2506 μ units. So we do not get a more efficient way to store it, unless we can find parameters so we get below 2506 μ units.

points	$BC = 1$	$BC = 2$	$BC = 3$
$n = 10$	6506.6	5313.7	3842.9
$n = 20$	1014.4	725.4	460.9
$n = 100$	8.8	5.8	3.4

Table 3: μ units of error for cycloid.

points	$BC = 1$	$BC = 2$	$BC = 3$
$n = 10$	13897.6	11664.8	8867.6
$n = 20$	2100.5	1531.0	1001.7
$n = 100$	17.7	11.8	7.0

Table 4: μ units of error for prolate cycloid.

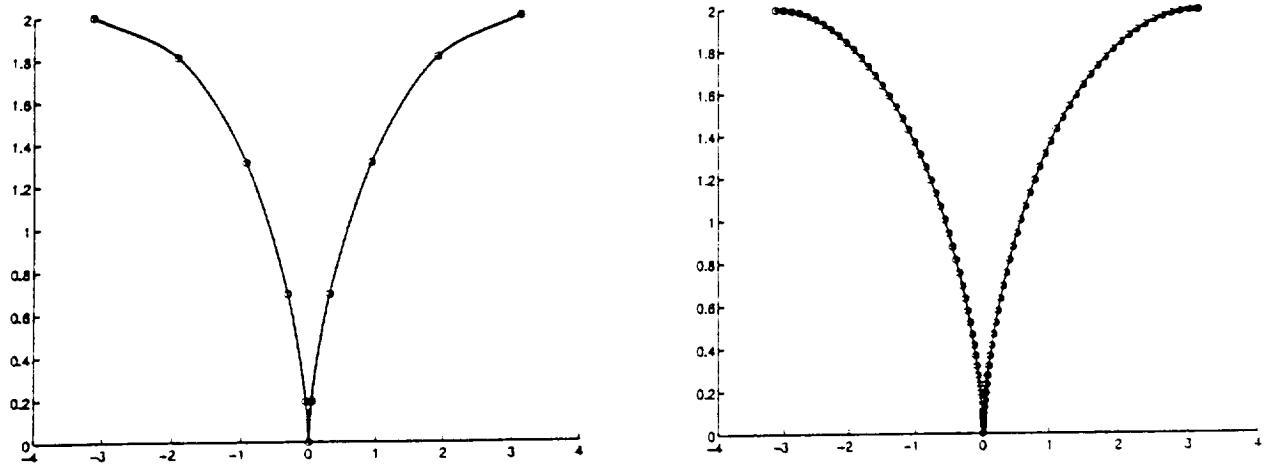


Figure 7: Graph of cycloid, BC= 2, reproduced with n=10, n=100 and $\lambda_1 = 10, \lambda_2 = 10$

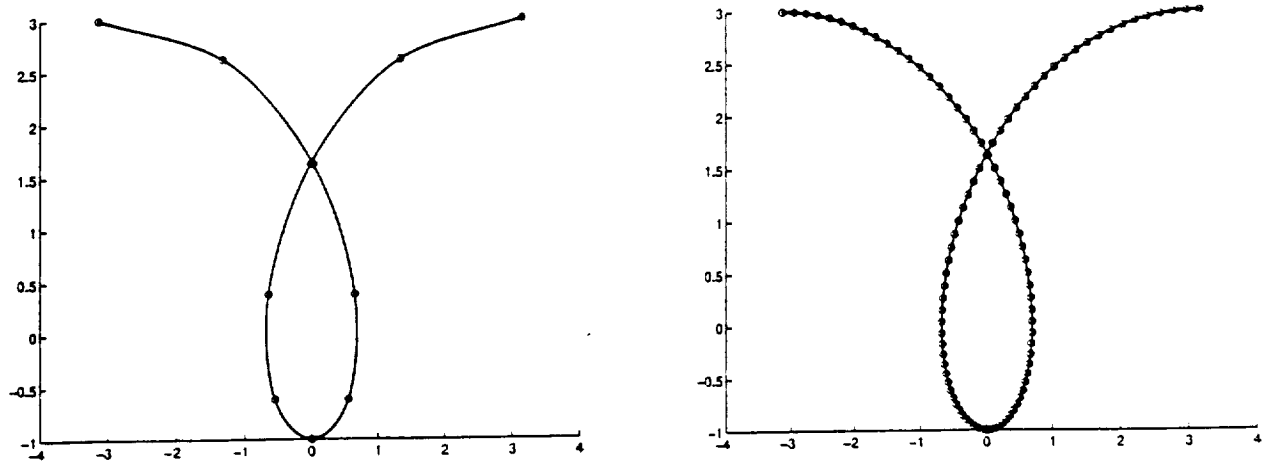


Figure 8: Graph of prolate cycloid, BC= 2, reproduced with n=10, n=100 and $\lambda_1 = 10, \lambda_2 = 10$

Finally we look at a cycloid.

$$\begin{cases} x &= \pi t - \sin \pi t \\ y &= 1 - \cos \pi t \end{cases}$$

and a prolate cycloid.

$$\begin{cases} x &= \pi t - 2 \sin \pi t \\ y &= 1 - 2 \cos \pi t \end{cases}$$

It is evident that the cusp and the crossover do not cause any problem, as could be expected since we are using a parameterized interpolant.

We can compare the difference of the graphs in figure 7 and figure 8, where we are using one crude approximation with n=10 and one extensive approximation with n=100. This time it is evident that the error is mainly located at the boundaries. In table 3 and table 4 we can see that the best results came from using constant velocity boundary conditions.

Looking at table 3 and table 4 again, we see that the error decreases at an approximately cubic rate as the number of interpolation points increase, which is much better than the quadratic decrease that can be seen in table 2.

My guess is that this behavior comes from the fact that the curve $y = |x|$ does not have a differentiable parameterization.

6.2 Applied on a signature

Included as figure 9 is a scanned picture of my own signature. I have tried to pick some roughly equidistant points on the signature, (According to the scale indicated on the axis.) and used the interpolation algorithm we have been studying to reproduce it. The reproduction is made with $n = 74$, $\lambda_1 = \lambda_2 = 10$ and no coupling between the two coordinates. For boundary conditions I have chosen to use constant velocity, since it has been the most successful condition.

As can be seen we get a very close resemblance between the original and the reproduction. How close is hard to say because we do not have the signature given as a parameterized function, therefore we are not able to calculate the error as before.

The things characterizing the signatures, are also the things that are hard to recover with the interpolation. Such as the turnover in the connection from the "P", and the cusps in the "r". To get a good reproduction, an equidistant

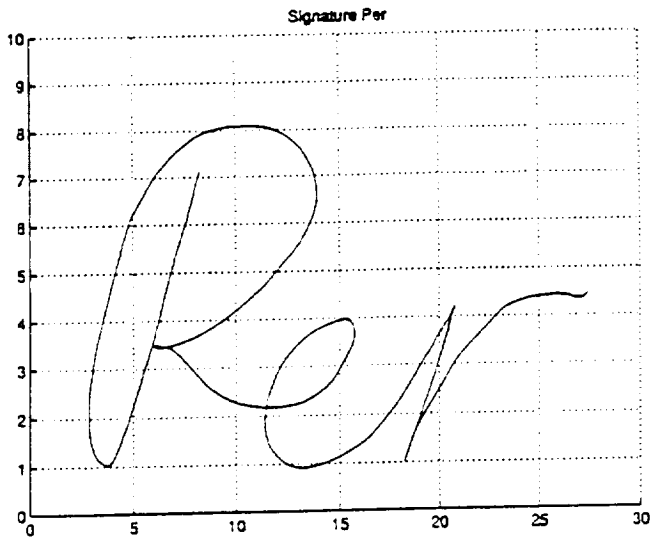


Figure 9: The scanned signature

Figure 10: The reproduced signature

distribution of the interpolation points is not enough, more points has to be concentrated around the characterizing areas.

7 Resumé - Summary in Swedish

Lagring av signaturer kan göras på många sätt. Vi har valt att lagra ett antal punkter på signaturen, och sedan reproducera denna genom att interpolera punkterna med splines.

Genom att använda välkända resultat inom systemteori så kan man generera olika sorters splines genom att ändra på några parametrar. Jag har nyttjat denna metod för att generera parametriserade splines i planet. Man inser snart att man måste införa randvillkor på lösningen, och valet av dessa får inte ske hur som helst eftersom de påverkar hela lösningen. Därför har jag lagt ner en hel del arbete just på denna punkt. De bästa resulteten har jag erhållit genom valet att ha konstant hastighet vid ändpunkterna.

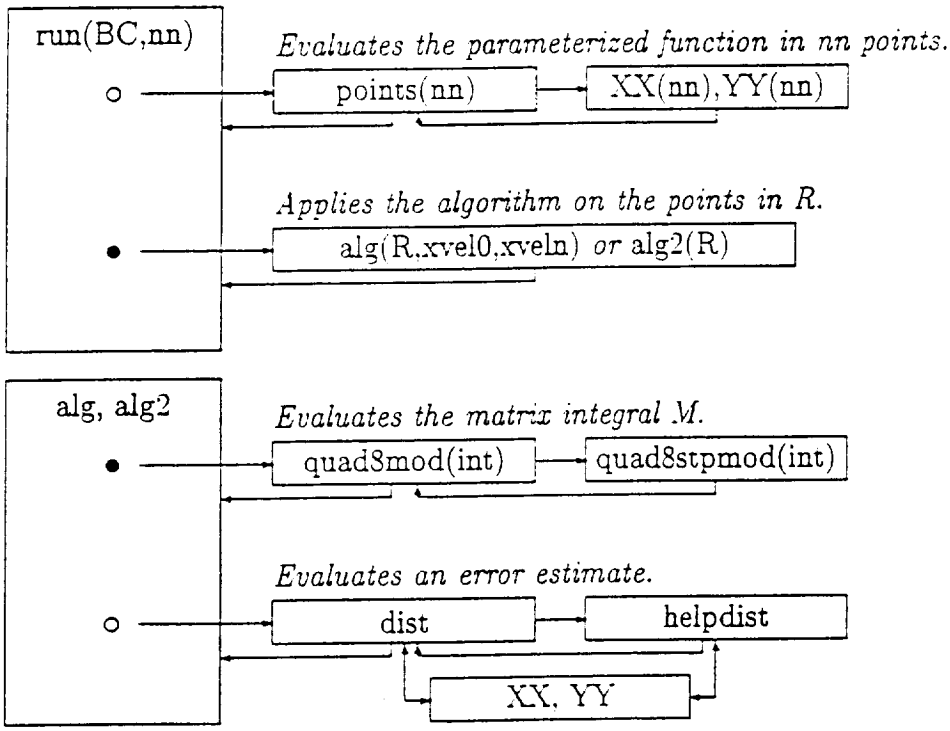
En av de saker som karaktäriserar handstilar är dess rundhet. Denna kan ges en direkt översättning i egenvärdena till systemmatrisen, och vi kommer alltså välja att lagra dessa utöver punkterna på signaturen.

8 Programs

The programs in Matlab and Maple that were used to implement the algorithm developed in this report follow.

8.1 Matlab Program

To make it easier to understand the structure of the program, the following flow charts describe how the programs are traversed.



The loops marked with an unfilled circle is only available when the interpolated points are given by the parameterized function (XX(t),YY(t)).

```
function error=run(BC,nn);
%BC              % Type of boundary conditions
                 % 1 = zero initial and final velocity.
                 % 2 = heading for first point, last.
                 % 3 = zero initial and final acceleration.
% If nn is specified, runs alg with nn points given by XX,YY.
% Otherwise runs alg with points given by ginput.
global n h alpha1 alpha2 beta1 beta2 lambda1 lambda2
global errorcalc ctrlsignal
clg

%% Setting of parameters %%
alpha1=0;
alpha2=0;
beta1=0;
beta2=0;
lambda1=10;
lambda2=10;

% Decides what steps are going to be made
errorcalc=1;        % error estimation
ctrlsignal=0;       % plotting of control signals


if nargin == 1
  [x,y]=ginput;
  R(1,:)=x'; R(2,:)=y';
else
  R=points(nn);
end;

n=length(R)-1;    % Number of interpolationpoints -1.
h=2/n;            % Time inbetween points

%% Plots a circle at all the points thats interpolated %%
hold on
for i=1:n+1
```

```
  plot(R(1,i),R(2,i),'o')
end;

%% Calling alg with the prepared data %%
if BC == 3
   error=alg2(R);
else
  if BC == 2, xvel0=(R(:,2)-R(:,1))/h;
    else xvel0=zeros(2,1);
  end;
  if BC == 2, xveln=(R(:,n+1)-R(:,n))/h;
    else xveln=zeros(2,1);
  end;
  error=alg(R,xvel0, xveln);
end;

%% Loop to allow graphic alteration of BC. %%

b=input('"1" for graphic mod of BC, "0" to quit  ');
while b == 1,
  xvel0=10*(ginput(1)'-R(:,1));
  xveln=-10*(ginput(1)'-R(:,n+1));
  clg
  hold on
  for i=1:n+1
    plot(R(1,i),R(2,i),'o')
  end;
  error=alg(R,xvel0,xveln);
  b=input('"1" for graphic mod of BC, "0" to quit ');
end;

end;
```

```
function R=points(nn);
% Forms R  with the help of XX, YY.
```

```
% R = 2*(nn+1)-matrix.
global a b h

a=1; b=a;
h=2/nn;

for i= 0:nn
  R(:,i+1)=[XX(-1+i*h); YY(-1+i*h)];
end
end;
```

```
function res=XX(t);
global a b
  res= a*t*pi-b*sin(t*pi);
end;
```

```
function res=YY(t)
global a b
  res= a-b*cos(t*pi);
end;
```

```
function error=alg(R,xvel0,xveln);
% R = Matrix of interpolationpoints,  x0 ... xn.
%    first row = x-coordinates, second row = y-coordinates.
% xvel0, xveln = Boundary conditions

global a b A B C n h alpha1 alpha2 beta1 beta2 lambda1 lambda2
global errorcalc ctrlsignal

%% The System %%

A=[[ 0  1  0  0]
   [ 0  lambda1  alpha1  alpha2]
```

```
   [ 0   0   0 1]
   [ beta1 beta2 0 lambda2]];
B=[[0 0]
   [1 0]
   [0 0]
   [0 1]];
C=[[1 0 0 0]
   [0 0 1 0]];

%% Calculation of the integral from 0 to h in m steps %%
m=40;              % number of points between interpolationpoints
tol=1e-08;         % the numeric error tolerance

Mtau(:,1:4)=zeros(4);
tau=0;
for j=1:m
  oldtau=tau;
  tau=oldtau+h/m;
  Mtau(:,4*j+1:4*j+4)= quad8mod('int',oldtau,tau,tol)
                       + Mtau(:,4*j-3:4*j);
end;
M=Mtau(:,4*m+1:4*m+4);

%% Forming of the Matrixes for the Blockdiagonal system %%

e_Ah=expm(-A*h);
Minv=inv(M);
ZZ=Minv*e_Ah;
WW=e_Ah'*ZZ+Minv;

WL=[WW(2,2) WW(2,4); WW(4,2) WW(4,4)];  % Partitioning matrixes
ZLU=[ZZ(2,2) ZZ(2,4); ZZ(4,2) ZZ(4,4)];
ZLL=[ZZ(2,2) ZZ(4,2); ZZ(2,4) ZZ(4,4)];

WR=[WW(2,1) WW(2,3); WW(4,1) WW(4,3)];
ZRU=[ZZ(2,1) ZZ(2,3); ZZ(4,1) ZZ(4,3)];
ZRL=[ZZ(1,2) ZZ(3,2); ZZ(1,4) ZZ(3,4)];
```

```
%% The boundary conditions  %%

xvel(:,1)=xvel0;
xvel(:,n+1)=xveln;

%%  Forming of the right side of the Blockdiagonal system  %%

for i=2:n
  Omega(:,i)=ZRL*R(:,i-1)-WR*R(:,i)+ZRU*R(:,i+1);
end;
Omega(:,2)=Omega(:,2)+ZLL*xvel(:,1);
Omega(:,n)=Omega(:,n)+ZLU*xvel(:,n+1);

%%  Gausselimination to produce upper triangular system  %%

DD(:,3:4)=WL;
for i=3:n
  zd=ZLL*inv(DD(:,2*i-3:2*i-2));
  DD(:,2*i-1:2*i)=WL-zd*ZLU;
  Omega(:,i)=Omega(:,i)+zd*Omega(:,i-1);
end

%%  Backsubstitution to solve for the xvel  %%

xvel(:,n)=DD(:,2*n-1:2*n)\Omega(:,n);
for i=n-1:-1:2
  xvel(:,i)=DD(:,2*i-1:2*i)\(ZLU*xvel(:,i+1)+Omega(:,i));
end;

%% Making of state vectors %%

for i=0:n
  x(:,i+1)=[[R(1,i+1)]
            [xvel(1,i+1)]
            [R(2,i+1)]
            [xvel(2,i+1)]];
```

PRECEDING PAGE BLANK NOT FILMED


```
function error=alg2(R);
% R = Matrix of interpolationpoints,  x0 ... xn.
%    first row = x-coordinates, second row = y-coordinates.
% BC = acceleration in x and y direction are both = 0.

global a b A B C n h alpha1 alpha2 beta1 beta2 lambda1 lambda2
global errorcalc ctrlsignal

%% The System %%

A=[[ 0  1  0  0]
   [ 0  lambda1  alpha1  alpha2]
   [ 0  0  0 1]
   [ beta1 beta2 0 lambda2]];
B=[[0 0]
   [1 0]
   [0 0]
   [0 1]];
C=[[1 0 0 0]
   [0 0 1 0]];

%% Calculation of the integral from 0 to h in m steps %%
m=40;            % number of points between interpolationpoints
tol=1e-08;       % the numeric error tolerance

Mtau(:,1:4)=zeros(4);
tau=0;
for j=1:m
  oldtau=tau;
  tau=oldtau+h/m;
  Mtau(:,4*j+1:4*j+4)= quad8mod('int',oldtau,tau,tol)
                       + Mtau(:,4*j-3:4*j);
end;
M=Mtau(:,4*m+1:4*m+4);
```

```
%%  Forming of the Matrixes for the Blockdiagonal system  %%

e_Ah=expm(-A*h);
Minv=inv(M);
ZZ=Minv*e_Ah;
WW=e_Ah'*ZZ+Minv;

WL=[WW(2,2) WW(2,4); WW(4,2) WW(4,4)];  % Partitioning matrixes
ZLU=[ZZ(2,2) ZZ(2,4); ZZ(4,2) ZZ(4,4)];
ZLL=[ZZ(2,2) ZZ(4,2); ZZ(2,4) ZZ(4,4)];

WR=[WW(2,1) WW(2,3); WW(4,1) WW(4,3)];
ZRU=[ZZ(2,1) ZZ(2,3); ZZ(4,1) ZZ(4,3)];
ZRL=[ZZ(1,2) ZZ(3,2); ZZ(1,4) ZZ(3,4)];

Ulu=[Minv(2,2) Minv(2,4);Minv(4,2) Minv(4,4)];
Uru=[Minv(2,1) Minv(4,1);Minv(2,3) Minv(4,3)];
Vl=[lambda1 alpha2; beta2 lambda2];
Vr=[0 alpha1; beta1 0];

%%  Forming of the right side of the Blockdiagonal system  %%

for i=2:n
  Omega(:,i)=ZRL*R(:,i-1)-WR*R(:,i)+ZRU*R(:,i+1);
end;
Omega(:,1)=(Vr-Uru)*R(:,1) + ZRU*R(:,2);
Omega(:,n+1)=ZRL*R(:,n) - (WR-Uru+Vr)*R(:,n+1);

%%  Gausselimination to produce upper triangular system  %%

DD(:,1:2)=Ulu-Vl;
for i=2:n+1
  zd=ZLL*inv(DD(:,2*i-3:2*i-2));
  DD(:,2*i-1:2*i)=WL-zd*ZLU;
  Omega(:,i)=Omega(:,i)+zd*Omega(:,i-1);
end
DD(:,2*n+1:2*n+2)= (WL-Ulu+Vl) - zd*ZLU;
```

```
%%  Backsubstitution to solve for the xvel  %%

xvel(:,n+1)=DD(:,2*n+1:2*n+2)\Omega(:,n+1);
for i=n:-1:1
  xvel(:,i)=DD(:,2*i-1:2*i)\(ZLU*xvel(:,i+1)+Omega(:,i));
end;

%% Making of state vectors %%

for i=0:n
  x(:,i+1)=[[R(1,i+1)]
            [xvel(1,i+1)]
            [R(2,i+1)]
            [xvel(2,i+1)]];
end;

%%  Plotting of trajectory
%%  and error estimate calculation

sumnorm=0;
hold on
for j=0:m
  eAtau=expm(A*j*h/m);
  for i=0:n-1
    entry=eAtau*(x(:,i+1)+Mtau(:,4*j+1:4*j+4)*Minv*
                          (e_Ah*x(:,i+2)-x(:,i+1)));
    plot(entry(1),entry(3),'.')
    if errorcalc
      sumnorm = sumnorm + dist(entry(1),entry(3),i,h);
    end;
  end;
end;

%% Plotting of the control signals %%

if ctrlsignal
```

```
  pause, clg
  subplot(2,1,1),hold on
  subplot(2,1,2),hold on
  for i=0:n-1
    for j=0:m
      csignvec(:,j+1)=B'*expm(-A'*j*h/40)*Minv*
                      (e_Ah*x(:,i+2)-x(:,i+1));
    end;
    subplot(2,1,1),plot(i*h:h/m:(i+1)*h,csignvec(1,:))
    subplot(2,1,2),plot(i*h:h/m:(i+1)*h,csignvec(2,:))
  end;
end;

error=sumnorm/n/m;
end;
```

```
function [Q,cnt] = quad8mod(F,a,b,tol)
%Alteration of the original matlab toolbox program. QUAD8
% Numerical evaluation of an integral, higher order method.
% Q = QUAD8('F',A,B,TOL) approximates the integral of F(X)
% from A to B to within a relative error of TOL.
% 'F' is a string containing the name of the function.
% The function must return a 4*4-matrix output value if
% given an input value.
% Q = Inf is returned if an excessive recursion level is
% reached, indicating a possibly singular integral.
% QUAD8 uses an adaptive recursive Newton Cotes 8 panel rule.
% Cleve Moler, 5-08-88.
% Copyright (c) 1984-94 by The MathWorks, Inc.
% [Q,cnt] = quad8(F,a,b,tol) also returns a function
%           evaluation count.

% Top level initialization, Newton-Cotes weights
w=[3956 23552 -3712 41984 -18160 41984 -3712 23552 3956]/14175;
x=a + (0:8)*(b-a)/8;
```

```
% set up function call
for i=x
  y = [y feval(F,i)];
end;

% Adaptive, recursive Newton-Cotes 8 panel quadrature
Q0 = zeros(4);
[Q,cnt] = quad8stpmod(F,a,b,tol,0,w,x,y,Q0);
cnt = cnt + 9;
end;
```

```
function [Q,cnt] = quad8stpmod(F,a,b,tol,lev,w,x0,f0,Q0)
%Alteration of the original matlab toolbox program.QUAD8STP
% Recursive function used by QUAD8.
% [Q,cnt] = quad8stp(F,a,b,tol,lev,w,f,Q0) tries to approximate
% the integral of f(x) from a to b to within a relative error
% of tol.
% F is a string containing the name of f. The remaining
% arguments are generated by quad8mod or by the recursion.
% lev is the recursion level.
% w is the weights in the 8 panel Newton Cotes formula.
% x0 is a vector of 9 equally spaced abscissa is the interval.
% f0 is a matrix of the 9 function values at x.
% Q0 is an approximate value of the integral.
% Cleve Moler, 5-08-88.
% Copyright (c) 1984-94 by The MathWorks, Inc.

LEVMAX = 10;
% Evaluate function at midpoints
% of left and right half intervals.
x = zeros(1,17);
x(1:2:17) = x0;
x(2:2:16) = (x0(1:8) + x0(2:9))/2;

f(:,1:4)= f0(:,1:4);
for i=1:8
```

```
  f(:,8*i-3:8*i) = feval(F,x(2*i));
  f(:,8*i+1:8*i+4) = f0(:,4*i+1:4*i+4);
end;

% Integrate over half intervals.
h = (b-a)/16;
Q1=0;Q2=0;
for i=1:9
  Q1 = Q1 + h*w(i)*f(:,4*i-3:4*i);
  Q2 = Q2 + h*w(10-i)*f(:,69-i*4:72-i*4);
end;
Q = Q1 + Q2;

% Recursively refine approximations.
if norm(Q - Q0) > tol*norm(Q) & lev <= LEVMAX
   c = (a+b)/2;
   [Q1,cnt1] =
      quad8stpmod(F,a,c,tol/2,lev+1,w,x(1:9),f(:,1:36),Q1);
   [Q2,cnt2] =
      quad8stpmod(F,c,b,tol/2,lev+1,w,x(9:17),f(:,33:68),Q2);
   Q = Q1 + Q2;
   cnt = cnt + cnt1 + cnt2;
end
end;
```

```
function res = integrand(v)
global A B C

e_AvB=expm(-A*v)*B;
res = e_AvB*e_AvB';
end;
```

```
function [d]=dist(xx,yy,i,h);
% Initiating search algorithm.
```

PRECEDING PAGE BLANK NOT FILMED

8.2 Maple Program

```
with(linalg);
with(student);

alpha1:=1;
alpha2:=1;
beta1:=0;
beta2:=0;
lambda1:=100;
lambda2:=100;

a:=1;
b:=a;
h:=0.2;
n:=22;
m:=15;
R:=vector(n+1);.
XX:=t->a*t*Pi-b*sin(t*Pi);
YY:=t->a-b*cos(t*Pi);
for i from 0 to n
do
  R[i+1]:=matrix([[XX(-1+i*h)],
                  [YY(-1+i*h)]]);
od;

A:=matrix([[ 0, 1, 0, 0],
           [ 0, lambda1, alpha1, alpha2],
           [ 0, 0, 0, 1],
           [ beta1, beta2, 0, lambda2]]);
B:=matrix([[0,0],[1,0],[0,0],[0,1]]);
C:=matrix([[1,0,0,0],[0,0,1,0]]);

alias(Id = &*())
Aprim:=transpose(A);
Bprim:=transpose(B);
e_At:=t->exponential(-A*t);
```

```
e_Aprimt:=t->exponential(-Aprim*t);
eAt:=t->exponential(A*t);
e_Ah:=e_At(h);

integranden:= proc (v)
    evalm(e_At(v) &* B &* Bprim &* e_Aprimt(v));
end;

integrera:= proc (funk)
   global v;
   int(funk,v);
end;

map(integrera,integranden(v));
integralen:=map(simplify,");

evaluera:=proc (funk)
 global tau;
 subs(v=tau,funk);
end;

Mtau:=vector(m+1);
Mtau[1]:=matrix([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]);
tau:=0;
M0:=evalm(map(evaluera,integralen));

for j from 1 to m
do
  tau:=j*h/m;
  Mtau[j+1]:=evalm(map(evaluera,integralen)-M0);
od;
M:=Mtau[m+1];

Minv:=evalm(inverse(M));
ZZ:=evalm(Minv&*e_Ah);
WW:=evalm(transpose(e_Ah)&*ZZ+Minv);
```

```
WL:=submatrix(WW,[2,4],[2,4]);
ZLU:=submatrix(ZZ,[2,4],[2,4]);
ZLL:=transpose(submatrix(ZZ,[2,4],[2,4]));

WR:=submatrix(WW,[2,4],[1,3]);
ZRU:=submatrix(ZZ,[2,4],[1,3]);
ZRL:=transpose(submatrix(ZZ,[1,3],[2,4]));

xvel:=vector(n+1);
xvel[1]:=evalm((R[2]-R[1])/h);
xvel[n+1]:=evalm((R[n+1]-R[n])/h);

Omega:=vector(n+1);
for i from 2 to n
do
  Omega[i]:=ZRL&*R[i-1]-WR&*R[i]+ZRU&*R[i+1];
od;
Omega[2]:=Omega[2]+ZLL&*xvel[1];
Omega[n]:=Omega[n]+ZLU&*xvel[n+1];

DD:=vector(n+1);
DD[2]:=WL;
for i from 3 to n
do
  zd:=evalm(ZLL&*inverse(DD[i-1]));
  DD[i]:=WL-zd&*ZLU;
  Omega[i]:=Omega[i]+zd&*Omega[i-1];
od;

xvel[n]:=linsolve(DD[n],Omega[n]);
for i from n-1 by -1 to 2
do
  xvel[i]:=linsolve(DD[i],ZLU&*xvel[i+1]+Omega[i]);
od;

x:=vector(n+1);
for i from 0 to n
```

```
do
  x[i+1]:=matrix([[R[i+1][1,1]],
                  [xvel[i+1][1,1]],
                  [R[i+1][2,1]],
                  [xvel[i+1][2,1]]]);
od;

plotlist:=[];
for j from 1 to m
do
  tau:=j*h/m;
  eAtau:=evalm(eAt(tau));

  for i from 0 to n-1
  do
    entry:=evalm(eAtau&*(x[i+1]+Mtau[j+1]&*Minv
                                  &*(e_Ah&*x[i+2]-x[i+1])));
    plotlist:={[entry[1,1],entry[3,1]],op(plotlist)};
  od;
od;

plot(plotlist,style=point);
```

References

[1] Hildebrand, F. B. Introduction to Numerical Analysis. *New York: Dover Publications, Inc.* (1987).

[2] Kahaner, D., Moler, C., Nash, S. Numerical Methods and Software. *Eaglewood Cliffs: Prentice-Hall International, Inc.* (1989).

[3] Faires J.D., Burden L.R. Numerical Methods. *Boston: PWS-KENT Publishing Company* (1993).

[4] Taylor A.E. Introduction to Functional Analysis.*New York: Wiley* (1958).

[5] Selby S.M. CRC Standard Mathematical Tables. *Cleveland: The Chemical Rubber Co.* (1972)

54-63
44082
N95-25805 118

How to Fly an Aircraft with Control Theory and Splines

by

Anders Karlsson

December 1994

Abstract

When trying to fly an aircraft as smoothly as possible it is a good idea to use the derivatives of the pilot command instead of using the actual control. This idea was implemented with splines and control theory, in a system that tries to model an aircraft. Computer calculations in Matlab shows that it is impossible to receive enough smooth control signals by this way. This is due to the fact that the splines not only try to approximate the test function, but also its derivatives. A perfect traction is received but we have to pay in very peaky control signals and accelerations.

Contents

1 Acknowledgments **3**

2 Introduction **4**

3 Reachability **5**
3.1 Reachability for Time-Invariant Systems 8

4 Stability **10**
4.1 Stability of Continuous-Time Systems 10
4.2 Stability matrices 11

5 The Systems **14**
5.1 Transfer Functions 14
5.2 System 1 15
5.3 System 2 16
5.4 System 3 17

6 Derivations **19**
6.1 System 1 22
6.2 System 2 24
6.3 System 3 30

7 The Test **38**
7.1 Test curves 38

8 Results **40**
8.1 Surprising Results 54

9 Resume in Swedish **56**

10 Matlab Programs **57**

List of Figures

1 Test curve: Step function 38
2 Test curve: Sine function 39

3 Test curve: Tangent hyperbolica 39
4 System 1 tracking sine, acc & cs u, λ= -1. 40
5 System 3 tracking sine, acc & cs u, $\lambda 1 = \lambda 2$ = -1, $\epsilon = 0$. . . . 41
6 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -0.1, $\lambda 2$ = -0.2, $\epsilon = 0$. 42
7 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -0.1, $\lambda 2$ = -0.2, ϵ = -0.01. 42
8 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -0.1, $\lambda 2$ = -0.2, ϵ = 0.006. 43
9 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -0.1, $\lambda 2$ = -2, ϵ = 0.006. 43
10 System 3 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -1, $\lambda 2$ = -1, $\epsilon = 0$. 44
11 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -1, $\lambda 2$ = -1, ϵ = -0.003. 45
12 System 2 tracking tangent hyperbolic, acc & cs u, $\lambda 1$ = -1, $\lambda 2$ = -1, ϵ = 0.001. 45
13 System 3 tracking sine, cs w, $\lambda 1 = \lambda 2$ = -1, $\epsilon = 0$. 46
14 System 3 tracking sine, acc, $\lambda 1 = \lambda 2$ = -1, ϵ = 0.002. 47
15 System 3 tracking sine, cs u, $\lambda 1 = \lambda 2$ = -1, ϵ = 0.002. 47
16 System 3 tracking sine, cs w, $\lambda 1 = \lambda 2$ = -1, ϵ = 0.002. 48
17 System 3 tracking sine, $\ddot{w}$, $\lambda 1 = \lambda 2$ = -1, ϵ = 0.002. 48
18 System 3 tracking sine, acc, $\lambda 1 = \lambda 2$ = -1, ϵ = -0.001. 49
19 System 3 tracking sine, cs u, $\lambda 1 = \lambda 2$ = -1, ϵ = -0.001. 49
20 System 3 tracking the step function, cs u, $\lambda 1$= -0.1, $\lambda 2$= -0.2, $\epsilon = 0$. 50
21 System 3 tracking the step function, cs u, $\lambda 1$= -0.1, $\lambda 2$= -0.2, ϵ = -0.00255. 50
22 System 3 tracking the step function, cs u, $\lambda 1$= -0.1, $\lambda 2$= -0.2, ϵ = 0.0055. 51
23 System 2 tracking sine, acc, $\lambda 1 = \lambda 2$= -1, ϵ = 0.004186. . . . 52
24 System 2 tracking sine, acc, $\lambda 1 = \lambda 2$= -1, ϵ = 0.004188. . . . 52
25 System 2 tracking sine, acc, $\lambda 1 = \lambda 2$= -1, ϵ = 0.00418702471143. 53
26 System 2 tracking sine, cs u, $\lambda 1 = \lambda 2$= -1, ϵ = 0.00418702471143 53
27 System 2 tracking sine, cs w, $\lambda 1 = \lambda 2$= -1, ϵ = 0.00418702471143 54
28 System 1 tracking sine, vel, $\lambda = 0$. 55

1 Acknowledgments

To start with I would like to thank my advisor Horn Professor Clyde Martin, Texas Tech University, USA and Professor Anders Lindquist, Royal Institute of Technology, Sweden for giving me this great opportunity to carry through an interesting and instructive thesis.

I would also like to thank candidate for the doctorate Per Enquist, Ms, Royal Institute of Technology, Sweden, for letting me use the software written for his thesis, Control Theory and Splines, applied to Signature Storage. This thesis was also written with Horn Professor Clyde Martin as an advisor during the summer and fall of 1994.

The famous southern hospitality did not turn out to be just a hearsay, I have had the opportunity to experience it in the best possible way. Therefore I would like to thank everyone that made my stay in Lubbock so very pleasant.

Finally I would like to bring up my gratitude to everybody that use theirs intelligence to give us so beautiful software as Matlab, Maple and Latex.

2 Introduction

In the beginning our intention was to calculate control laws for an aircraft model so it would fly as smooth as possible in three dimensions. The comfort for the passengers was the most important consideration when we forced our system, the representation of the plane, to follow a certain trajectory.

All the programming has been realized in a numeric computation and visualization software called Matlab. Also Maple has been used in some of the heaviest calculations and my third contact with the more advanced computer world was Latex that this report is written in. The second half of this paper consists of Matlab code ended with listed references.

The test began in one dimension with three different kinds of systems. By way of introduction the essential conceptions of reachability and stability were examined and written down in chapter 3 and 4 respectively. With these tools we could investigate the main features of the systems and obtained the result that all of them were completely reachable, stable but not guaranteed input-output stable (see chapter 5, The Systems).

A spline is the curve of an n-degree polynomial that is joined in its endpoints with similar polynomials. They are connected in the way that they have the first n-1 derivatives, at the jointly point, in common. Chapter 6 consists of calculations for the spline approximation and the control theory.

Chapter 8, Results, discusses some of the results we received and also displays examples of graphs that were obtained. The test could not be concluded in the way we thought due to a surprising combination between control theory and splines. The last two pages in chapter 8 deals with this main result.

3 Reachability

When controlling an aircraft we will be sure that a suitable control signal u can take us to all desirable states. Transfered to our one dimensional case we have to determine under what circumstances there is an input signal u which transfers the state from $x(t_0) = x_0$ to $x(t_1) = x_1$. This is a basic issue in systems theory and it leads to the concept of reachability. We will call a system completely reachable if it has the property that this can be done in any positive time for any two points.

Most of the trains of thought in the following proofs are derived from lecture notes given by Tomas Björk, Optimization and Systems Theory, Royal Institute of Technology, Stockholm, Sweden, during the fall of 93.

Consider the system.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t); \qquad x(t_0) = x_0.$$

with the general solution

$$x(t) = \Phi(t, t_0)x_0 + \int_{t_0}^{t} \Phi(t, s)B(s)u(s)ds.$$

In order to reach the desired state $x(t_1) = x_1$ the following equality must be fulfilled.

$$x_1 = \Phi(t_1, t_0)x_0 + \int_{t_0}^{t_1} \Phi(t_1, s)B(s)u(s)ds.$$

Define $d \triangleq x_1 - \Phi(t_1, t_0)x_0$ and let $\mathcal{U}$ be the space of input signals. Defining the mapping $L : U \rightarrow R^n$ as

Equation 3.1 $Lu \triangleq \int_{t_0}^{t_1} \Phi(t_1, s)B(s)u(s)ds.$

It is obvious that the desired state transfer is possible if and only if the equation $Lu=d$ has a solution, i.e. $d \in Im\ L$.

It is easily verified that L is a linear mapping, but since it does not act between two finite-dimensional vector spaces, L does not have a finite-dimensional matrix representation.

Taylor's 'Introduction to Functional Analysis' helps us prove the following theorem [Taylor, page 250].

Theorem 3.1 *If X, Y are complete inner product spaces and $L : X \rightarrow Y$ is a linear continuous operator then*

$$Im\ L = Im\ LL^*$$

R^n is a Hilbert space but what kind of space is $\mathcal{U}$? Define the inner product for $\mathcal{U}$ as

$$(u, v)_{\mathcal{U}} \triangleq \int_{t_0}^{t_1} u(t)^T v(t) dt$$

and it can be proved that $\mathcal{U}$ becomes a Hilbert space. The adjoint operator L^* is determined by

$$(Lu, d)_{R^n} = d^T \int_{t_0}^{t_1} \Phi(t_1, s) B(s) u(s) ds =$$

$$\int_{t_0}^{t_1} \left\{ B^T(s) \Phi^T(t_1, s) d \right\}^T u(s) ds = (u, L^* d)_{\mathcal{U}}$$

$$\forall u \in \mathcal{U}, d \in R^n$$

Consequently we get

$$L^* : R^n \to U \quad as \quad (L^* d)(t) = B^T(t) \Phi^T(t_1, t) d$$

and finally

$$LL^* d = \left[\int_{t_0}^{t_1} \Phi(t_1, s) B(s) B^T(s) \Phi^T(t_1, s) ds \right] d$$

We thus have a linear mapping

$$LL^* : R^n \to R^n$$

that is given by the symmetric, positive semidefinite $n \times n$ matrix.

$$W(t_0, t_1) = \int_{t_0}^{t_1} \Phi(t_1, s) B(s) B^T(s) \Phi^T(t_1, s) ds$$

Theorem 3.2 *We can take a system from $x(t_0) = x_0$ to $x(t_1) = x_1$ if and only if*

$$d \triangleq x_1 - \Phi(t_1, t_0) x_0 \in Im \ W(t_0, t_1)$$

We also have that the control signal u with minimum norm (energy) is given by

$$\hat{u}(t) = B^T(t) \Phi^T(t_1, t) a$$

there a is just any solution to

$$W(t_0, t_1) a = d$$

Remark 1 *The point with the above is that it is much easier to characterize Im W than Im L, because W is an ordinary matrix.*

Proof Let L be as in (3.1)
(if) Suppose first that $d \in W(t_0, t_1)$, i.e. $d \in Im\ LL^*$, then $d = Im\ LL^* a$ for some $a \in R^n$. Let $u \triangleq L^* a$ and we get $Lu = LL^* a = d$. Thus, $d \in Im\ L$ and the state transfer is possible.
(only if) Suppose now that the state transfer is possible, i.e. $d \in Im\ L$. Furthermore, suppose that $d \notin Im\ W(t_0, t_1)$, i.e. $d \notin Im\ LL^*$. This will give a contradiction.

Recall that for any matrix A it holds that $Im\ A = (ker\ A^T)^\perp$. Since LL^* is a symmetric matrix, we get $d \notin (ker\ LL^*)^\perp$. This implies that there is a $z \in ker\ LL^*$, i.e. $LL^* z = 0$, such that $(z, d)_{R^n} \neq 0$. But, $LL^* z = 0$ implies that $0 = (z, LL^* z)_{R^n} = (L^* z, L^* z)_u$. Hence, $L^* z = 0$. Now the contradiction easily follows since

$$0 \neq (z, d)_{R^n} = (z, Lu)_{R^n} = (L^* z, u)_u = 0$$

The final step to prove the optimality of $\hat{u}$. Let u be any solution of $Lu=d$. Then $Lu = L\hat{u}$ so $L(u - \hat{u}) = 0$. This gives

$$0 = (a, L(u - \hat{u}))_{R^n} = (L^* a, u - \hat{u})_u = (\hat{u}, u - \hat{u}).$$

Hence, $(\hat{u}, u) = (\hat{u}, \hat{u})$. We now get by using the Cauchy-Schwartz inequality that

$$(\hat{u}, \hat{u}) = (\hat{u}, u) \leq (\hat{u}, \hat{u})^{1/2} (u, u)^{1/2}.$$

Dividing by $(\hat{u}, \hat{u})^{1/2}$ yields that

$$(\hat{u}, \hat{u})^{1/2} \leq (u, u)^{1/2}.$$

Hence, $\hat{u}$ is optimal. □

3.1 Reachability for Time-Invariant Systems

For a time-invariant system

$$\dot{x} = Ax + Bu \qquad W(t_0, t_1) = \int_{t_0}^{t_1} e^{A(t_1 - s)} B B^T e^{A^T(t_1 - s)} ds$$

the question about reachability is radically simplified.

Definition 3.1 *Let (A,B) be a matrix pair, where A is $n \times n$. The reachability matrix Γ is defined as*

$$\Gamma \triangleq \left[B, AB, \ldots, A^{n-1}B\right]$$

Theorem 3.3 *For all (t_0, t_1) such that $t_0 < t_1$ we have*

$$Im\ W(t_0, t_1) = Im\ \Gamma$$

Proof I. Im $\Gamma \subseteq$ Im W

$Im\ \Gamma \subseteq Im\ W \Leftrightarrow (ker\ \Gamma^T)^\perp \subseteq (ker\ W^T)^\perp \Leftrightarrow ker\ W^\perp \subseteq ker\ \Gamma^\perp$

Presume that $a \in ker\ W$, i.e. $Wa=0$ so $a^T W a = 0$ and hence it follows that $a^T e^{A(t_1 - s)} B = 0 \quad \forall s \in [t_0, t_1]$.

Derivation with regard to s a couple of times and $s := t_1$ gives

$a^T B = 0 \ldots a^T AB = 0 \ldots\ldots a^T A^{n-1} B = 0$ i.e. $a \in ker\ \Gamma^T$.

II. Im W $\subseteq$ Im Γ

In the same way as above we are going to prove that $ker\ \Gamma^T \subseteq ker\ W$. Suppose that $a^T \Gamma = 0$. By Cayley-Hamilton follows

$$a^T A^k B = 0 \qquad k = 0, 1, 2, \ldots$$

Accordingly we have

$$a^T e^{-As} B = \sum_{k=0}^{\infty} \frac{s^k}{k!} a^T A^k B = 0$$

So it follows that $a^T W = 0$, i.e. $Wa=0$, i.e. $a \in ker\ W$. □

Remark 2 *Since $Im\ \Gamma = Im\ W(t_0, t_1)$ for any interval (t_0, t_1), we see that in the time-invariant case the image of the reachability Gramian is independent of the interval (t_0, t_1). However, this does not imply that the state transfer can occur during a fortuitous short time interval.*

Definition 3.2 *Let n be the dimension of the state space. The pair (A,B) is said to be completely reachable if* Γ *has full rank, i.e.*

$$rank\ \Gamma = n$$

Definition 3.3 *The reachable subspace* $\mathcal{R}$ *is defined as*

$$\mathcal{R} \triangleq Im\ \left[B, AB, A^2B, \ldots, A^{n-1}B\right]$$

We easily see that $\mathcal{R}$ *is the set of states that can be reached from the origin.*

Lemma 3.1 *The reachable subspace* $\mathcal{R}$ *is A-invariant, i.e.*

$$A\mathcal{R} \subseteq \mathcal{R}$$

In particular, $e^{At}\mathcal{R} \subseteq \mathcal{R}$ *for all* $t \in R^n$.

Proof Since, by the Cayley-Hamilton theorem, A^n is a linear combination of A^j for *j=0,1,...,n-1* it follows that

$$A\mathcal{R} = \left[AB, A^2B, \ldots, A^nB\right] \subseteq Im\left[B, AB, \ldots, A^{n-1}B\right] = \mathcal{R}.$$

Moreover, by induction we get $A^j\mathcal{R} \subseteq \mathcal{R}$, which implies that

$$e^{At}\mathcal{R} = \sum_{j=0}^{\infty} \frac{t^j}{j!} A^j\mathcal{R} \subseteq \mathcal{R} \quad \square$$

To further clarify the picture we note that if the state of the system is in $\mathcal{R}$, at some instant, it is impossible to steer the state out of $\mathcal{R}$. Neither is it possible to enter $\mathcal{R}$ from an initial state not in $\mathcal{R}$. Particularly we have that if $x_0, x_1 \in \mathcal{R}$ then the state transfer can occur in just any time t. The points that can be reached in a time t from a given x_0 establish the plane

$$\mathcal{R}(x_0, t) \triangleq e^{At}x_0 + \mathcal{R}.$$

4 Stability

A very essential problem when designing a control system is how to avoid instability, i.e. that the output increases without limit. The following section is an abridgement of chapter four in "An Introduction to Mathematical Systems Theory" by A. Lindquist and J. Sand [Lindquist/Sand]. All theory dealing with the alternative approach, the Lyapunov equation, is omitted.

Intuitively an input-output system is stable if a bounded input produces a bounded output or if the output tends to zero, or at least remains bounded, when the input is zero.

For nonlinear systems, stability in this sense is typically dependent on the initial conditions and the specific input applied. Hence, in general, stability is not the property of a system, but rather the property of a solution.

This chapter deals only with the stability of time-invariant linear systems, a subject which is drastically simplified by the fact that the complete set of solutions of the system $\dot{x} = Ax$ can be displayed explicitly by means of the Jordan form. As a consequence, it is enough to check the eigenvalues of A in order to determine whether a bounded input produces a bounded output, and thus it will be meaningful to talk about stable systems.

4.1 Stability of Continuous-Time Systems

We want a bounded input to give a bounded output, which is sometimes abbreviated as *BIBO*-stability.

Definition 4.1 *The system*

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) \end{cases}$$

is input-output stable if there is a k such that

$$\left.\begin{array}{l} x(t_0) = 0, \\ \|u(t)\| \leq 1 \quad t \in [t_0, \infty) \end{array}\right\} \Rightarrow \|y(t)\| \leq k, \quad t \in [t_0, \infty)$$

for every t_0.

Example 4.1 *Consider the time-invariant case, where (A,B,C) are constant matrices. Then*

$$y(t) = \int_{t_0}^{t_1} Ce^{A(t-s)}Bu(s)ds.$$

Defining $G(t) \triangleq Ce^{At}B$,

$$\|y(t)\| \leq \int_{t_0}^{t_1} \|G(t-s)\| \, \|u(s)\| ds \quad (since \ \|u(t)\| \leq 1)$$

$$\leq \int_{t_0}^{t_1} \|G(\sigma)\| d\sigma$$

$$\leq \|C\|\|B\| \int_{t_0}^{t-t_0} \|e^{A\Gamma}\| d\Gamma,$$

i.e., a sufficient condition for input-output stability is that the integral $\int_0^\infty \|e^{At}\| dt$ *is convergent.* □

4.2 Stability matrices

Let us study the homogeneous system:
Equation 4.1 $\dot{x} = Ax; \quad x(0) = x_0.$

Definition 4.2 *The system (4.1) is stable if the solution is bounded on the interval* $[0, \infty)$ *for all initial values* x_0 *and asymptotically stable if* $x(t) \to 0$ *when* $t \to \infty$ *for all* x_0.

Theorem 4.1 **(1)** *The system (4.1) is asymptotically stable if and only if the real parts of all the eigenvalues of A are less than zero, i.e. the eigenvalues are all located in the open left half plane.*
(2) *The system (4.1) is unstable if A has at least one eigenvalue in the open right half plane.*

Proof In this proof we shall use a fundamental result from linear algebra, the Jordan decomposition theorem. This theorem guarantees the existence of a basis for $\mathcal{R}^n$ in which the representation of the linear mapping A takes a particularly simple form.

Transform the matrix A to Jordan form $A = TJT^{-1}$, where J is a block-diagonal matrix.

$$J = diag(J_1, J_2, \ldots, J_r)$$

and each $d_v \times d_v$ block J_v has the form

$$J_v = \begin{bmatrix} \lambda_v & 1 & & 0 \\ & \lambda_v & 1 & \\ & & \ddots & 1 \\ 0 & & & \lambda_v \end{bmatrix},$$

λ_v being an eigenvalue of A. Thus,

$$e^{At} = T \begin{bmatrix} e^{J_1 t} & & & 0 \\ & e^{J_2 t} & & \\ & & \ddots & \\ 0 & & & e^{J_r t} \end{bmatrix} T^{-1},$$

so it remains to analyze each $e^{J_v t}$. But J_v has the form

$$J_v = \lambda_v I + S_v$$

where S_v is a shift matrix

$$S_v = \begin{bmatrix} 0 & 1 & & 0 \\ & 0 & 1 & \\ & & 0 & \ddots \\ & & & 1 \\ 0 & & & 0 \end{bmatrix}$$

of dimension $d_v \times d_v$, having the property that $S^i = 0$ for $i \geq d_v$. Consequently,

$$e^{J_v t} = e^{\lambda_v t} e^{S_v t} = e^{\lambda_v t} \left(I + tS + \frac{t^2}{2!} S^2 + \ldots + \frac{t^{d_v - 1}}{(d_v - 1)!} S^{d_v - 1} \right)$$

and therefore, setting $\sigma_v = Re\lambda_v$ and $\omega_v = Im\lambda_v$,

Equation 4.2 $e^{At} = \sum_v e^{\sigma_v t} P_v(t)(cos\omega_v t + sin\omega_v t)$,

where $P_v(t)$ is a matrix-valued polynomial of dimension $d_v - 1$ in t. From this expression it follows that **(1)** $e^{At}x_0 \to 0$ for all x_0 if and only if $\sigma_v \triangleq Re\lambda_v < 0$ for all v and that **(2)** $e^{At}x_0 \to \infty$ for at least one x_0 if some $\sigma_v > 0$. □

Lemma 4.1 *The system in equation* 4.1 *is stable if and only if all eigenvalues of A are located in the closed left half plane and any eigenvalues on the imaginary axis correspond to one dimensional Jordan blocks.*

Proof By theorem 4.1 (1) we only need to worry about terms in *(4.2)* for which $\sigma_v = 0$, i.e. $e^{\sigma_v t} = 1$. These terms will remain bounded if and only if the degree of P_v is zero, i.e. $d_v = 1$.

Definition 4.3 *A is a stability matrix if Re* $\lambda(A) < 0$.

Theorem 4.2 *If A is a stability matrix then the time invariant system*

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

is input-output stable.

Proof If all eigenvalues of A have negative real parts so that all σ_v in *(4.2)* are negative then

$$\int_0^\infty \|e^{At}\| dt < \infty$$

and hence, in view of example 4.1 the system is input-output stable. □
The last theorem is very important for us because it deals with the kind of system we use when modeling an aircraft.

5 The Systems

The test was accomplished with three systems of different kinds. All systems use a single input and produce a single output, a so called *SISO*-system. As we will see later all systems have the necessary property of complete reachability, as was discussed in chapter 3.

That a bounded input gives a bounded output is sometimes abbreviated as *BIBO*-stability. This highly desirable property for a system was discussed in chapter 4 and will be further examined for each specific case.

5.1 Transfer Functions

This subject is discussed in Etkin's book "Dynamics of Atmospheric Flight" [Etkin, page 50-51]. He writes

> System analysis frequently reduces to the calculation of system outputs for given inputs. A convenient and powerful tool in such analysis is the transfer function, a function *G(s)* of the Laplace transform variable s [Complex valued], that relates input *u(t)* and output *y(t)* as follows,
>
> $$G(s) = \bar{y}(s)/\bar{u}(s)$$
>
> where ($\bar{}$) denotes the Laplace transform. So long as *u(t)* and *y(t)* are Laplace transformable the transfer function defined above exists. However, it will in general be a function of the initial values of y and its derivatives, and moreover, for nonlinear and time varying systems, of the particular input *u(t)* as well. Such a transfer function is of relatively little use. We can however obtain a unique function *G(s)* if (I) the system is linear and time invariant, and (II) it is initially quiescent, i.e. at rest at the origin in state space with no inputs.

He continues,

> When *u(t)* and *y(t)* are zero for $t < 0$, the Laplace and Fourier transforms are simply related, i.e. $\bar{u}(i\omega) = U(\omega)$. It follows that
>
> $$G(i\omega) = \frac{Y(\omega)}{U(\omega)}$$

Sometimes it is $G(i\omega)$ that is called the transfer function.

When examining different transfer functions in a Bode diagram it shows that there are systems with the same absolute value curve but with different phase curves. Of all systems with the same absolute value curve there is one with less negative phase advance, it is called a minimum phase system [Glad/Ljung, page 109].

I give the following theorem without a proof.

Theorem 5.1 *A theorem with a rational transfer function is in minimum phase if and only if it has neither poles nor zeros in the open right half plane.*

The others are called non minimum phase systems. This distinction is very important because we know from one dimensional control theory that a system with zeros in the numerator will start off in the opposite direction. This bad quality can make the system difficult to control.

A λ-value less or equal to zero are assumed in the following calculations.

5.2 System 1

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & \lambda \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \qquad x = \begin{pmatrix} y \\ \dot{y} \end{pmatrix}$$

gives the system dynamics $\ddot{y} = \lambda\dot{y} + u$
The reachability matrix

$$\Gamma = \begin{bmatrix} 0 & 1 \\ 1 & \lambda \end{bmatrix}$$

has full rank for all lambda and the system is therefore completely reachable. System 1's transfer function

$$Y(s) = \frac{1}{s(s-\lambda)} U(s)$$

without neither poles nor zeros in the open right half plane indicates that it is a minimum phase system and should therefore be easy to control.

The Jordan transform of the matrix A is $P^{-1}JP$ where

$$J = \begin{bmatrix} 0 & 0 \\ 0 & \lambda \end{bmatrix}, \qquad P = \begin{bmatrix} \lambda/(\lambda-1) & 1/(1-\lambda) \\ 0 & 1 \end{bmatrix}.$$

Because all eigenvalues of A are located in the closed left half plane and the eigenvalue on the imaginary axis correspond to an one dimensional Jordan block we know that the system is stable. Owing to this quality we are guaranteed that when a fortuitous in signal ultimately equals zero, the solution to the system $\dot{x} = Ax$ is bounded on the interval $[0, \infty)$. This of course implicates that also the output is bounded. Referring to previous theory the eigenvalue on the imaginary axis prevents input-output stability.

5.3 System 2

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \lambda 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \lambda 2 \end{bmatrix} x + \begin{bmatrix} 0 \\ \epsilon \\ 0 \\ 1 \end{bmatrix} w \qquad y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x \qquad x = \begin{pmatrix} y \\ \dot{y} \\ u \\ \dot{u} \end{pmatrix}$$

gives the system dynamics $\ddot{y} = \lambda 1\dot{y} + u + \epsilon w \qquad \ddot{u} = \lambda 2\dot{u} + w.$
The reachability matrix

$$\Gamma = \begin{bmatrix} 0 & \epsilon & \epsilon\lambda 1 & \epsilon\lambda 1^2 + 1 \\ \epsilon & \epsilon\lambda 1 & \epsilon\lambda 1^2 + 1 & \epsilon\lambda^3 + \lambda 1 + \lambda 2 \\ 0 & 1 & \lambda 2 & \lambda 2^2 \\ 1 & \lambda 2 & \lambda 2^2 & \lambda 2^3 \end{bmatrix}$$

has full rank for all values on λ and ϵ and the system is therefore completely reachable.

System 2's transfer function

$$Y(s) = \frac{\epsilon s^2 - \epsilon\lambda 2s + 1}{s^2(s-\lambda 1)(s-\lambda 2)} U(s)$$

gives for negative λ's that there are no poles in the open right half plane. The numerator $\epsilon s^2 - \epsilon\lambda 2s + 1 = 0$ give the solution

$$s = \frac{\lambda 2}{2} \pm \sqrt{\frac{\lambda 2^2}{4} - \frac{1}{\epsilon}}$$

This implies for a negative ϵ that we have a zero in the open right half plane. As $\lambda 2$ always is below or equal to zero the poles for a positive ϵ are in the left half plane. So the system should be easy to control for a positive ϵ and probably more difficult for a negative value on the variable.
Taking the Jordan transform of A gives that J equals

$$\begin{bmatrix} \lambda 2 & 0 & 0 & 0 \\ 0 & \lambda 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Carrying through the same discussion as for system 1 we see that system 2 has the same properties, stable but not guaranteed input-output stable.

5.4 System 3

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \lambda 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \lambda 2 \end{bmatrix} x + \begin{bmatrix} 0 \\ \epsilon \\ 0 \\ 0 \\ 1 \end{bmatrix} w \quad y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} x \quad x = \begin{pmatrix} y \\ \dot{y} \\ u \\ \dot{u} \\ \ddot{u} \end{pmatrix}$$

gives the system dynamics $\ddot{y} = \lambda 1 \dot{y} + u + \epsilon w \qquad \ddot{u} = \lambda 2 \ddot{u} + w$
The reachability matrix

$$\Gamma = \begin{bmatrix} 0 & \epsilon & \epsilon\lambda 1 & \epsilon\lambda 1^2 & \epsilon\lambda 1^3 + 1 \\ \epsilon & \epsilon\lambda 1 & \epsilon\lambda 1^2 & \epsilon\lambda 1^3 + 1 & \epsilon\lambda^4 + \lambda 1 + \lambda 2 \\ 0 & 0 & 1 & \lambda 2 & \lambda 2^2 \\ 0 & 1 & \lambda 2 & \lambda 2^2 & \lambda 2^3 \\ 1 & \lambda 2 & \lambda 2^2 & \lambda 2^3 & \lambda 2^4 \end{bmatrix}$$

has always full rank and the system is therefore completely reachable.

System 3's transfer function

$$Y(s) = \frac{\epsilon s^3 - \epsilon\lambda 2 s^2 + 1}{s^3(s^2 - (\lambda 1 + \lambda 2)s + \lambda 1 \cdot \lambda 2)} U(s)$$

can be examined by Routh's algorithm. The numerator $\epsilon s^3 - \epsilon\lambda 2 s^2 + 1$ gives the following tableau.

$$\begin{bmatrix} \epsilon & 0 \\ -\epsilon\lambda 2 & 1 \\ 1/\lambda 2 & 0 \\ 1 & 0 \end{bmatrix}$$

As $\lambda 2$ always is below, or equal to, zero we get for a positive ϵ that the left side coefficients $\epsilon > 0 \quad -\epsilon\lambda 2 > 0 \quad 1/\lambda 2 < 0 \quad 1 > 0$ change sign two times. This indicates that the system has two zeros in the open right half plane for a positive ϵ. The same calculations for a negative ϵ gives that the system has one zero in the open right half plane.

The solution to the denominator

$$s^3(s^2 - (\lambda 1 + \lambda 2)s + \lambda 1 \cdot \lambda 2) = 0$$

gives that for all negative values on $\lambda 1$ and $\lambda 2$ we have three zeros on the imaginary axis and two zeros in the open left half plane. If either $\lambda 1$ or $\lambda 2$ equals zero we get four zeros on the imaginary axis and one in the open left half plane. When all eigenvalues equal zero we get of course all zeros on the imaginary axis.

All this together gives that system 3 never will be a minimum phase system and will therefore be more difficult to control.

Taking the Jordan transform of A gives that J equals

$$\begin{bmatrix} \lambda 2 & 0 & 0 & 0 & 0 \\ 0 & \lambda 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and this implies that system 3 is stable. The always present eigenvalues that equal zero prevent the system to be guaranteed input-output stable.

6 Derivations

The fundamental idea of the test was to simulate an aircraft that is directed by the flight control to follow a certain path. The given trajectory can be seen as a set of points that shall be passed at a certain time. By way of introduction our intention was to determine how exact a given path in three dimensions could be tracked with maintained comfort for the passengers. Priority one was to minimize the acceleration and thereby the stress to the individuals.

To start with, the test was accomplished in one dimension. This simplify the calculations radically and is a common approach to such experiments. Given the set of points $\{y_0, y_1, \ldots, y_n\}$ and the corresponding time $\{t_0, t_1, \ldots, t_n\}$, we would like to perceive the control laws $\{u_0, u_1, \ldots, u_{n-1}\}$ that take the system through the points in such a pleasant way as possible.

Consider the control u_k that takes the system from state vector x_k to x_{k+1}.

$$u_k : \begin{pmatrix} x_k \\ t_k \end{pmatrix} \mapsto \begin{pmatrix} x_{k+1} \\ t_{k+1} \end{pmatrix}$$

Because $t \in [t_k, t_{k+1}]$ the state of the system will be

$$x(t) = e^{A(t-t_k)}x_k + \int_{t_k}^{t} e^{A(t-s)}Bu_k(s)ds$$

and as the state of the system is x_{k+1} at time t_{k+1} we receive the condition,

Equation 6.1

$$x_{k+1} = e^{A(t_{k+1}-t_k)}x_k + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-s)}Bu_k(s)ds.$$

The solution u_k to equation (6.1) that minimizes the energy of the control signal is then given by, see chapter 3.

Equation 6.2

$$u_k(t) = B^T e^{-A^T t}\left(\int_{t_k}^{t_{k+1}} e^{-As}BB^T e^{-A^T s}ds\right)^{-1}\left(e^{-At_{k+1}}x_{k+1} - e^{-At_k}x_k\right)$$

That would be very convenient if the integral in equation (6.2) could be simplified in any way.

We can see our flight path as the aircraft is flown through a large number of points by an autopilot. With a specific time interval the plane receives a correction signal that makes the vehicle track the path with high accuracy. The time interval $t_{k+1} - t_k$ are constant and determined by the frequency the automatic pilot works with.

Assumption 6.1 *Let* $t_{k+1} - t_k = h$.

Assumption (6.1) can be used to simplify the integral in equation (6.2).

$$\int_{t_k}^{t_{k+1}} e^{-As} BB^T e^{-A^T s} ds = \{\tau = s - t_k\} =$$

$$e^{-At_k} \underbrace{\int_0^h e^{-A\tau} BB^T e^{-A^T \tau} d\tau}_{\textit{matrix constant}} e^{-A^T t_k}$$

Definition 6.1

$$M \triangleq \int_0^h e^{-A\tau} BB^T e^{-A^T \tau} d\tau$$

The equation (6.2) can be rewritten as

Equation 6.3

$$u_k(t) = B^T e^{-A^T(t-t_k)} M^{-1} (e^{-Ah} x_{k+1} - x_k)$$

The control would be specified completely by (6.3) if the whole state vector at each interpolation point was known. As only the points $(y_0, y_1, \ldots, y_n)$ are known we have to apply some kind of conditions on the equation to obtain a solution.

The control u is the actual control that the pilot or the auto-pilot achieve. A very natural choice is therefore to require that the control u is continuous.

Assumption 6.2

$$u_k(t_{k+1}) = u_{k+1}(t_{k+1})$$

By this we acquire (n-1) conditions and allow us to write

$$u_k(t_{k+1}) = u_{k+1}(t_{k+1})$$

$$\Longleftrightarrow$$

$$B^T e^{-A^T h} M^{-1}(e^{-Ah}x_{k+1} - x_k) = B^T M^{-1}(e^{-Ah}x_{k+2} - x_{k+1}).$$

This equation can be simplified by

Definition 6.2

$$Z \triangleq M^{-1}e^{-Ah}$$

$$W \triangleq e^{-A^T h}M^{-1}e^{-Ah} + M^{-1}$$

and finally we obtain the modified expression

$$B^T(Z^T x_k - W x_{k+1} + Z x_{k+2}) = 0 \qquad k = 0, 1, \ldots, n-2.$$

Written in block diagonal form it becomes,

Equation 6.4

$$B^T \begin{bmatrix} Z^T & -W & Z & \ldots & 0 & 0 & 0 \\ 0 & Z^T & -W & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -W & Z & 0 \\ 0 & 0 & 0 & \ldots & Z^T & -W & Z \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = 0$$

As our three systems are very different the calculations will differ from here and all further computations have to be treated separately.

6.1 System 1

This is the original system that was first implemented in Matlab. Each state vector x_k consists of two parts, a known coordinate y_k and an unknown velocity $\dot{y}_k$. By partitioning the matrixes in definition 6.2 as

$$Z = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}, \qquad Z^T = \begin{bmatrix} z_{11} & z_{21} \\ z_{12} & z_{22} \end{bmatrix}, \qquad W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

and using the notations given in the following definition, the unknowns can be kept on the left hand side and the given position coordinates can be moved to the right hand side.

Definition 6.3

$$W_l^B = B^T \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix} = [w_{22}]$$

$$Z_{lu}^B = B^T \begin{bmatrix} z_{12} \\ z_{22} \end{bmatrix} = [z_{22}]$$

$$Z_{ll}^B = B^T \begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix} = [z_{22}]$$

$$W_r^B = B^T \begin{bmatrix} w_{11} \\ w_{21} \end{bmatrix} = [w_{21}]$$

$$Z_{ru}^B = B^T \begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = [z_{21}]$$

$$Z_{rl}^B = B^T \begin{bmatrix} z_{11} \\ z_{12} \end{bmatrix} = [z_{12}]$$

We get the system

$$\begin{bmatrix} Z_{ll}^B & -W_l^B & Z_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & Z_{ll}^B & -W_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -W_l^B & Z_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & Z_{ll}^B & -W_l^B & Z_{lu}^B \end{bmatrix} \begin{bmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_{n-2} \\ \dot{y}_{n-1} \\ \dot{y}_n \end{bmatrix} =$$

$$\begin{bmatrix} -Z_{rl}^B & W_r^B & -Z_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -Z_{rl}^B & W_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & W_r^B & -Z_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -Z_{rl}^B & W_r^B & -Z_{ru}^B \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix}$$

The right hand side consists of known parameters and is therefore a constant vector. Our system has (n+1) unknowns but only (n-1) equations so we need two more constraints.

After reading Per Enquist's paper "Control Theory and Splines, applied to Signature Storage" [Enquist] I decided to use the natural boundary conditions, $\ddot{y}_0 = 0$ and $\ddot{y}_n = 0$. This can be seen as a very real behavior for a vehicle and has also given the best results in former experiments. Enquist writes "This will let the initial direction and constant velocity of the system be decided so that the control energy is minimized" [Enquist, page 16-17]. The system dynamics equation $\ddot{y} = \lambda\dot{y} + u$ gives

$$\lambda\dot{y}_0 + u_0 = 0 \text{ where } u_0 = B^T M^{-1}\left(e^{-Ah}x_1 - x_0\right) = B^T Z x_1 - B^T M^{-1} x_0.$$

Definition 6.4

$$U^B \triangleq B^T \begin{bmatrix} m_{11}^{-1} & m_{12}^{-1} \\ m_{21}^{-1} & m_{22}^{-1} \end{bmatrix}$$

$$U_{lu}^B = \left[m_{22}^{-1}\right] \qquad U_{ru}^B = \left[m_{21}^{-1}\right]$$

We get

$$\left(\lambda - U_{lu}^{B}\right)\dot{y}_0 + Z_{lu}^{B}\dot{y}_1 = U_{ru}^{B}y_0 - Z_{ru}^{B}y_1$$

The system dynamics equation together with earlier definitions give

$$\lambda\dot{y}_n + u_{n-1}(t_n) = 0 \text{ where } u_{n-1}(t_n) = B^T e^{-A^T h} M^{-1}\left(e^{-Ah}x_n - x_{n-1}\right) =$$

$$B^T e^{-A^T h} Z x_n - B^T e^{-A^T h} M^{-1} x_{n-1} = B^T W x_n - B^T M^{-1} x_n - B^T Z^T x_{n-1}$$

We get

$$-Z_{ll}^{B}\dot{y}_{n-1} + \left(\lambda + W_l^B - U_{lu}^{B}\right)\dot{y}_n = Z_{rl}^{B}y_{n-1} + \left(U_{ru}^{B} - W_r^B\right)y_n$$

Add these two equations to our considered system and the number of unknown parameters equals the amount of equations. Thus is the problem solvable and the Matlab program that uses Gaussian elimination is displayed in mpr121der0.m.

6.2 System 2

By this system a new approach was introduced for the convenience of the passengers. Instead of direct using the performed control signal$u$ we use its derivatives to control the aircraft. The formula $\ddot{u} = \lambda 2\dot{u} + w$ gives the connection between the control u induced by the pilot and the artificial control w that actually flies the plane.

Each state vector x_k consists of the known coordinate y_k and the unknown parameters, velocity $\dot{y}_k$, control signal u_k and its first derivative $\dot{u}_k$. As we have (n+1) state vectors and each state vector consists of three unknowns it becomes a total of 3(n+1) unknown variables. The constraint that we require the control signal u to be continuous gives only (n-1) conditions. If the restrictions are introduced that also $\dot{u}$ and $\ddot{u}$ have to be continuous, we get further 2(n-1) conditions.

Assumption 6.3

$$\dot{u}_k(t_{k+1}) = \dot{u}_{k+1}(t_{k+1})$$

$$\ddot{u}_k(t_{k+1}) = \ddot{u}_{k+1}(t_{k+1})$$

Applying this to equation (6.3) becomes for the first condition

$$B^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2} \right) = 0 \qquad k = 0, 1, \ldots, n-2$$

and for the second condition

$$B^T A^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2} \right) = 0 \qquad k = 0, 1, \ldots, n-2.$$

By partitioning the matrices in definition 6.2 as

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} \\ z_{21} & z_{22} & z_{23} & z_{24} \\ z_{31} & z_{32} & z_{33} & z_{34} \\ z_{41} & z_{42} & z_{43} & z_{44} \end{bmatrix} \qquad Z^T = \begin{bmatrix} z_{11} & z_{21} & z_{31} & z_{41} \\ z_{12} & z_{22} & z_{32} & z_{42} \\ z_{13} & z_{23} & z_{33} & z_{43} \\ z_{14} & z_{24} & z_{34} & z_{44} \end{bmatrix}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix}$$

and using the notations given in the following definition, the unknowns can be kept on the left hand side and the given position coordinates can be moved to the right hand side. The matrix notation . symbolizes a whole row or column.

Definition 6.5
Continuous control signal, u:

$$W_l^B = B^T[w_{.2} \quad w_{.3} \quad w_{.4}] = [\epsilon w_{22} + w_{42} \quad \epsilon w_{23} + w_{43} \quad \epsilon w_{24} + w_{44}]$$

$$Z_{lu}^B = B^T[z_{.2} \quad z_{.3} \quad z_{.4}] = [\epsilon z_{22} + z_{42} \quad \epsilon z_{23} + z_{43} \quad \epsilon z_{24} + z_{44}]$$

$$Z_{ll}^B = B^T[z_{2.} \quad z_{3.} \quad z_{4.}] = [\epsilon z_{22} + z_{24} \quad \epsilon z_{32} + z_{34} \quad \epsilon z_{42} + z_{44}]$$

$$W_r^B = B^T[w_{.1}] = [\epsilon w_{21} + w_{41}]$$

$$Z_{ru}^B = B^T[z_{.1}] = [\epsilon z_{21} + z_{41}]$$

$$Z_{rl}^B = B^T[z_{1.}] = [\epsilon z_{12} + z_{14}]$$

Continuous first derivative of control signal, $\dot{u}$:

$$\dot{W}_l^B = B^T A^T [w_{.2} \quad w_{.3} \quad w_{.4}] = [\epsilon w_{12} + \epsilon\lambda 1 w_{22} + w_{32} + \lambda 2 w_{42}$$
$$\epsilon w_{13} + \epsilon\lambda 1 w_{23} + w_{33} + \lambda 2 w_{43} \quad \epsilon w_{14} + \epsilon\lambda 1 w_{24} + w_{34} + \lambda 2 w_{44}]$$

$$\dot{Z}_{lu}^B = B^T A^T [z_{.2} \quad z_{.3} \quad z_{.4}] = [\epsilon z_{12} + \epsilon\lambda 1 z_{22} + z_{32} + \lambda 2 z_{42}$$
$$\epsilon z_{13} + \epsilon\lambda 1 z_{23} + z_{33} + \lambda 2 z_{43} \quad \epsilon z_{14} + \epsilon\lambda 1 z_{24} + z_{34} + \lambda 2 z_{44}]$$

$$\dot{Z}_{ll}^B = B^T A^T [z_{2.} \quad z_{3.} \quad z_{4.}] = [\epsilon z_{21} + \epsilon\lambda 1 z_{22} + z_{23} + \lambda 2 z_{24}$$
$$\epsilon z_{31} + \epsilon\lambda 1 z_{32} + z_{33} + \lambda 2 z_{34} \quad \epsilon z_{41} + \epsilon\lambda 1 z_{42} + z_{43} + \lambda 2 z_{44}]$$

$$\dot{W}_r^B = B^T A^T [w_{.1}] = [\epsilon w_{11} + \epsilon\lambda 1 w_{21} + w_{31} + \lambda 2 w_{41}]$$
$$\dot{Z}_{ru}^B = B^T A^T [z_{.1}] = [\epsilon z_{11} + \epsilon\lambda 1 z_{21} + z_{31} + \lambda 2 z_{41}]$$
$$\dot{Z}_{rl}^B = B^T A^T [z_{1.}] = [\epsilon z_{11} + \epsilon\lambda 1 z_{12} + z_{13} + \lambda 2 z_{14}]$$

Continuous second derivative of control signal, $\ddot{u}$:

$$\ddot{W}_l^B = B^T A^T A^T [w_{.2} \quad w_{.3} \quad w_{.4}] =$$
$$[\epsilon\lambda 1 w_{12} + (\epsilon\lambda 1^2 + 1) w_{22} + \lambda 2 w_{32} + \lambda 2^2 w_{42}$$
$$\epsilon\lambda 1 w_{13} + (\epsilon\lambda 1^2 + 1) w_{23} + \lambda 2 w_{33} + \lambda 2^2 w_{43}$$
$$\epsilon\lambda 1 w_{14} + (\epsilon\lambda 1^2 + 1) w_{24} + \lambda 2 w_{34} + \lambda 2^2 w_{44}]$$

$$\ddot{Z}_{lu}^B = B^T A^T A^T [z_{.2} \quad z_{.3} \quad z_{.4}] =$$
$$[\epsilon\lambda 1 z_{12} + (\epsilon\lambda 1^2 + 1) z_{22} + \lambda 2 z_{32} + \lambda 2^2 z_{42}$$
$$\epsilon\lambda 1 z_{13} + (\epsilon\lambda 1^2 + 1) z_{23} + \lambda 2 z_{33} + \lambda 2^2 z_{43}$$
$$\epsilon\lambda 1 z_{14} + (\epsilon\lambda 1^2 + 1) z_{24} + \lambda 2 z_{34} + \lambda 2^2 z_{44}]$$

$$\ddot{Z}^B_{ll} = B^T A^T A^T [z_{2.} \;\; z_{3.} \;\; z_{4.}] =$$
$$[\epsilon\lambda 1 z_{21} + (\epsilon\lambda 1^2 + 1) z_{22} + \lambda 2 z_{23} + \lambda 2^2 z_{24}$$
$$\epsilon\lambda 1 z_{31} + (\epsilon\lambda 1^2 + 1) z_{32} + \lambda 2 z_{33} + \lambda 2^2 z_{34}$$
$$\epsilon\lambda 1 z_{41} + (\epsilon\lambda 1^2 + 1) z_{42} + \lambda 2 z_{43} + \lambda 2^2 z_{44}]$$

$$\ddot{W}^B_r = B^T A^T A^T [w_{.1}] = [\epsilon\lambda 1 w_{11} + (\epsilon\lambda 1^2 + 1) w_{21} + \lambda 2 w_{31} + \lambda 2^2 w_{41}]$$
$$\ddot{Z}^B_{ru} = B^T A^T A^T [z_{.1}] = [\epsilon\lambda 1 z_{11} + (\epsilon\lambda 1^2 + 1) z_{21} + \lambda 2 z_{31} + \lambda 2^2 z_{41}]$$
$$\ddot{Z}^B_{rl} = B^T A^T A^T [z_{1.}] = [\epsilon\lambda 1 z_{11} + (\epsilon\lambda 1^2 + 1) z_{12} + \lambda 2 z_{13} + \lambda 2^2 z_{14}]$$

We get the following systems, written in block diagonal form.

Each state vector is divided into two parts, a known portion x^p_k which contains the given position y_k and an unknown portion x^v_k that contains the parameters $\dot{y}_k$, u_k and $\dot{u}_k$. All right sides consist of known variables and are therefore constant vectors.

Continuous control signal u:

$$\begin{bmatrix} Z^B_{ll} & -W^B_l & Z^B_{lu} & \dots & 0 & 0 & 0 \\ 0 & Z^B_{ll} & -W^B_l & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -W^B_l & Z^B_{lu} & 0 \\ 0 & 0 & 0 & \dots & Z^B_{ll} & -W^B_l & Z^B_{lu} \end{bmatrix} \begin{bmatrix} x^v_0 \\ x^v_1 \\ x^v_2 \\ \vdots \\ x^v_{n-2} \\ x^v_{n-1} \\ x^v_n \end{bmatrix} =$$

$$\begin{bmatrix} -Z^B_{rl} & W^B_r & -Z^B_{ru} & \dots & 0 & 0 & 0 \\ 0 & -Z^B_{rl} & W^B_r & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & W^B_r & -Z^B_{ru} & 0 \\ 0 & 0 & 0 & \dots & -Z^B_{rl} & W^B_r & -Z^B_{ru} \end{bmatrix} \begin{bmatrix} x^p_0 \\ x^p_1 \\ x^p_2 \\ \vdots \\ x^p_{n-2} \\ x^p_{n-1} \\ x^p_n \end{bmatrix}$$

Continuous first derivative of control signal, $\dot{u}$:

$$\begin{bmatrix} \dot{Z}_{ll}^B & -\dot{W}_l^B & \dot{Z}_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & \dot{Z}_{ll}^B & -\dot{W}_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\dot{W}_l^B & \dot{Z}_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & \dot{Z}_{ll}^B & -\dot{W}_l^B & \dot{Z}_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\dot{Z}_{rl}^B & \dot{W}_r^B & -\dot{Z}_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -\dot{Z}_{rl}^B & \dot{W}_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dot{W}_r^B & -\dot{Z}_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -\dot{Z}_{rl}^B & \dot{W}_r^B & -\dot{Z}_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Continuous second derivative of control signal, $\ddot{u}$:

$$\begin{bmatrix} \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Having totally 3(n+1) unknowns but only 3(n-1) constraints we chose to enter differential approximations for the first and last state vector, this will decrease the number of unknown parameters by six and thus make the problem solvable. Remember that the time interval $t_{k+1} - t_k$ is constant and represented below as h.

The needed velocities are approximated as:

$$\dot{y}_0 = \frac{y_1 - y_0}{h}$$

$$\dot{y}_n = \frac{y_n - y_{n-1}}{h}$$

The needed control signals are approximated as:

$$\dot{y}_1 = \frac{y_2 - y_1}{h}$$

$$\dot{y}_{n-1} = \frac{y_{n-1} - y_{n-2}}{h}$$

$$u_0 = \frac{\dot{y}_0 - \dot{y}_1}{h}$$

$$u_n = \frac{\dot{y}_{n-1} - \dot{y}_n}{h}$$

The needed first derivative of the control signals are approximated as:

$$\dot{y}_2 = \frac{y_2 - y_1}{h}$$

$$\dot{y}_{n-2} = \frac{y_{n-2} - y_{n-3}}{h}$$

$$u_1 = \frac{\dot{y}_1 - \dot{y}_2}{h}$$

$$u_{n-1} = \frac{\dot{y}_{n-2} - \dot{y}_{n-1}}{h}$$

$$\dot{u}_0 = \frac{u_0 - u_1}{h}$$

$$\dot{u}_n = \frac{u_{n-1} - u_n}{h}$$

The Matlab program that solves the task for system 2 using Gaussian elimination is displayed in mpr141knovel.m.

6.3 System 3

This system uses the same approach for the convenience of the passengers as system 2 does. The only difference is that we now also use the third derivative of the control signal u to actually fly the plane. The formula $\dddot{u} = \lambda 2\,\ddot{u} + w$ gives the connection between the control u induced by the pilot and the artificial control w that actually flies the plane.

Each state vector x_k consists of the known coordinate y_k and the unknown parameters, velocity $\dot{y}_k$, control signal u_k, its first derivative $\dot{u}_k$ and its second derivative $\ddot{u}_k$. As we have (n+1) state vectors and each state vector consists of four unknowns it becomes a total of 4(n+1) unknown variables. The constraint that we require the control signal u to be continuous gives only (n-1) conditions. If the restrictions are introduced that also $\dot{u}$, $\ddot{u}$ and $\dddot{u}$ have to be continuous, we get further 3(n-1) conditions.

Assumption 6.4

$$\dot{u}_k(t_{k+1}) = \dot{u}_{k+1}(t_{k+1})$$
$$\ddot{u}_k(t_{k+1}) = \ddot{u}_{k+1}(t_{k+1})$$
$$\dddot{u}_k\,(t_{k+1}) = \dddot{u}_{k+1}\,(t_{k+1})$$

Applying this to equation (6.3) becomes for the first condition

$$B^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2}\right) = 0 \qquad k = 0, 1, \ldots, n-2,$$

for the second condition

$$B^T A^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2}\right) = 0 \quad k = 0, 1, \ldots, n-2$$

and for the third condition

$$B^T A^T A^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2}\right) = 0 \quad k = 0, 1, \ldots, n-2.$$

By partitioning the matrices in definition 6.2 as

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix} \qquad Z^T = \begin{bmatrix} z_{11} & z_{21} & z_{31} & z_{41} & z_{51} \\ z_{12} & z_{22} & z_{32} & z_{42} & z_{52} \\ z_{13} & z_{23} & z_{33} & z_{43} & z_{53} \\ z_{14} & z_{24} & z_{34} & z_{44} & z_{54} \\ z_{15} & z_{25} & z_{35} & z_{45} & z_{55} \end{bmatrix}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}$$

and using the notations given in the following definition, the unknowns can be kept on the left hand side and the given position coordinates can be moved to the right hand side. The matrix notation . symbolizes a whole row or column.

Definition 6.6

Continuous control signal, u:

$$W_l^B = B^T[w_{.2} \quad w_{.3} \quad w_{.4} \quad w_{.5}] =$$

$$[\epsilon w_{22} + w_{52} \quad \epsilon w_{23} + w_{53} \quad \epsilon w_{24} + w_{54} \quad \epsilon w_{25} + w_{55}]$$

$$Z_{lu}^B = B^T[z_{.2} \quad z_{.3} \quad z_{.4} \quad z_{.5}] = [\epsilon z_{22} + z_{52} \quad \epsilon z_{23} + z_{53} \quad \epsilon z_{24} + z_{54} \quad \epsilon z_{25} + z_{55}]$$

$$Z_{ll}^B = B^T[z_{2.} \quad z_{3.} \quad z_{4.} \quad z_{5.}] = [\epsilon z_{22} + z_{25} \quad \epsilon z_{32} + z_{35} \quad \epsilon z_{42} + z_{45} \quad \epsilon z_{52} + z_{55}]$$

$$W_r^B = B^T[w_{.1}] = [\epsilon w_{21} + w_{51}]$$

$$Z_{ru}^B = B^T[z_{.1}] = [\epsilon z_{21} + z_{51}]$$

$$Z_{rl}^B = B^T[z_{1.}] = [\epsilon z_{12} + z_{15}]$$

Continuous first derivative of control signal, $\dot{u}$:

$$\dot{W}_l^B = B^T A^T[w_{.2} \quad w_{.3} \quad w_{.4} \quad w_{.5}] = [\epsilon w_{12} + \epsilon\lambda 1 w_{22} + w_{42} + \lambda 2 w_{52}$$

$$\epsilon w_{13} + \epsilon\lambda 1 w_{23} + w_{43} + \lambda 2 w_{53} \quad \epsilon w_{14} + \epsilon\lambda 1 w_{24} + w_{44} + \lambda 2 w_{54}]$$

$$\dot{Z}_{lu}^B = B^T A^T[z_{.2} \quad z_{.3} \quad z_{.4} \quad z_{.5}] = [\epsilon z_{12} + \epsilon\lambda 1 z_{22} + z_{42} + \lambda 2 z_{52}$$

$$\epsilon z_{13} + \epsilon\lambda 1 z_{23} + z_{43} + \lambda 2 z_{53} \quad \epsilon z_{14} + \epsilon\lambda 1 z_{24} + z_{44} + \lambda 2 z_{54}]$$

$$\dot{Z}_{ll}^B = B^T A^T[z_{2.} \quad z_{3.} \quad z_{4.} \quad z_{5.}] = [\epsilon z_{21} + \epsilon\lambda 1 z_{22} + z_{24} + \lambda 2 z_{25}$$

$$\epsilon z_{31} + \epsilon\lambda 1 z_{32} + z_{34} + \lambda 2 z_{35} \quad \epsilon z_{41} + \epsilon\lambda 1 z_{42} + z_{44} + \lambda 2 z_{45}]$$

$$\dot{W}_r^B = B^T A^T [w_{.1}] = [\epsilon w_{11} + \epsilon\lambda 1 w_{21} + w_{41} + \lambda 2 w_{51}]$$
$$\dot{Z}_{ru}^B = B^T A^T [z_{.1}] = [\epsilon z_{11} + \epsilon\lambda 1 z_{21} + z_{41} + \lambda 2 z_{51}]$$
$$\dot{Z}_{rl}^B = B^T A^T [z_{1.}] = [\epsilon z_{11} + \epsilon\lambda 1 z_{12} + z_{14} + \lambda 2 z_{15}]$$

Continuous second derivative of control signal, ü:

$$\ddot{W}_l^B = B^T A^T A^T [w_{.2} \quad w_{.3} \quad w_{.4} \quad w_{.5}] =$$
$$[\epsilon\lambda 1 w_{12} + \epsilon\lambda 1^2 w_{22} + w_{32} + \lambda 2 w_{42} + \lambda 2^2 w_{52}$$
$$\epsilon\lambda 1 w_{13} + \epsilon\lambda 1^2 w_{23} + w_{33} + \lambda 2 w_{43} + \lambda 2^2 w_{53}$$
$$\epsilon\lambda 1 w_{14} + \epsilon\lambda 1^2 w_{24} + w_{34} + \lambda 2 w_{44} + \lambda 2^2 w_{54}]$$

$$\ddot{Z}_{lu}^B = B^T A^T A^T [z_{.2} \quad z_{.3} \quad z_{.4} \quad z_{.5}] =$$
$$[\epsilon\lambda 1 z_{12} + \epsilon\lambda 1^2 z_{22} + z_{32} + \lambda 2 z_{42} + \lambda 2^2 z_{52}$$
$$\epsilon\lambda 1 z_{13} + \epsilon\lambda 1^2 z_{23} + z_{33} + \lambda 2 z_{43} + \lambda 2^2 z_{53}$$
$$\epsilon\lambda 1 z_{14} + \epsilon\lambda 1^2 z_{24} + z_{34} + \lambda 2 z_{44} + \lambda 2^2 z_{54}]$$

$$\ddot{Z}_{ll}^B = B^T A^T A^T [z_{2.} \quad z_{3.} \quad z_{4.} \quad z_{5.}] =$$
$$[\epsilon\lambda 1 z_{21} + \epsilon\lambda 1^2 z_{22} + z_{23} + \lambda 2 z_{24} + \lambda 2^2 z_{25}$$
$$\epsilon\lambda 1 z_{31} + \epsilon\lambda 1^2 z_{32} + z_{33} + \lambda 2 z_{34} + \lambda 2^2 z_{35}$$
$$\epsilon\lambda 1 z_{41} + \epsilon\lambda 1^2 z_{42} + z_{43} + \lambda 2 z_{44} + \lambda 2^2 z_{45}]$$

$$\ddot{W}_r^B = B^T A^T A^T [w_{.1}] = [\epsilon\lambda 1 w_{11} + \epsilon\lambda 1^2 w_{21} + w_{31} + \lambda 2 w_{41} + \lambda 2^2 w_{51}]$$
$$\ddot{Z}_{ru}^B = B^T A^T A^T [z_{.1}] = [\epsilon\lambda 1 z_{11} + \epsilon\lambda 1^2 z_{21} + z_{31} + \lambda 2 z_{41} + \lambda 2^2 z_{51}]$$
$$\ddot{Z}_{rl}^B = B^T A^T A^T [z_{1.}] = [\epsilon\lambda 1 z_{11} + \epsilon\lambda 1^2 z_{12} + z_{13} + \lambda 2 z_{14} + \lambda 2^2 z_{15}]$$

Continuous third derivative of control signal, $\dddot{u}$:

$$\dddot{W}_l^B = B^T A^T A^T A^T [w_{.2} \quad w_{.3} \quad w_{.4} \quad w_{.5}] =$$
$$[\epsilon\lambda 1^2 w_{12} + (\epsilon\lambda 1^3 + 1) w_{22} + \lambda 2 w_{32} + \lambda 2^2 w_{42} + \lambda 2^3 w_{52}$$
$$\epsilon\lambda 1^2 w_{13} + (\epsilon\lambda 1^3 + 1) w_{23} + \lambda 2 w_{33} + \lambda 2^2 w_{43} + \lambda 2^3 w_{53}$$
$$\epsilon\lambda 1^2 w_{14} + (\epsilon\lambda 1^3 + 1) w_{24} + \lambda 2 w_{34} + \lambda 2^2 w_{44} + \lambda 2^3 w_{54}]$$

$$\dddot{Z}_{lu}^B = B^T A^T A^T A^T [z_{.2} \quad z_{.3} \quad z_{.4} \quad z_{.5}] =$$
$$[\epsilon\lambda 1^2 z_{12} + (\epsilon\lambda 1^3 + 1) z_{22} + \lambda 2 z_{32} + \lambda 2^2 z_{42} + \lambda 2^3 z_{52}$$
$$\epsilon\lambda 1^2 z_{13} + (\epsilon\lambda 1^3 + 1) z_{23} + \lambda 2 z_{33} + \lambda 2^2 z_{43} + \lambda 2^3 z_{53}$$
$$\epsilon\lambda 1^2 z_{14} + (\epsilon\lambda 1^3 + 1) z_{24} + \lambda 2 z_{34} + \lambda 2^2 z_{44} + \lambda 2^3 z_{54}]$$

$$\dddot{Z}_{ll}^B = B^T A^T A^T A^T [z_{2.} \quad z_{3.} \quad z_{4.} \quad z_{5.}] =$$
$$[\epsilon\lambda 1^2 z_{21} + (\epsilon\lambda 1^3 + 1) z_{22} + \lambda 2 z_{23} + \lambda 2^2 z_{24} + \lambda 2^3 z_{25}$$
$$\epsilon\lambda 1^2 z_{31} + (\epsilon\lambda 1^3 + 1) z_{32} + \lambda 2 z_{33} + \lambda 2^2 z_{34} + \lambda 2^3 z_{35}$$
$$\epsilon\lambda 1^2 z_{41} + (\epsilon\lambda 1^3 + 1) z_{42} + \lambda 2 z_{43} + \lambda 2^2 z_{44} + \lambda 2^3 z_{45}]$$

$$\dddot{W}_r^B = B^T A^T A^T A^T [w_{.1}] =$$
$$[\epsilon\lambda 1^2 w_{11} + (\epsilon\lambda 1^3 + 1) w_{21} + \lambda 2 w_{31} + \lambda 2^2 w_{41} + \lambda 2^3 w_{51}]$$

$$\dddot{Z}_{ru}^B = B^T A^T A^T A^T [z_{.1}] =$$
$$[\epsilon\lambda 1^2 z_{11} + (\epsilon\lambda 1^3 + 1) z_{21} + \lambda 2 z_{31} + \lambda 2^2 z_{41} + \lambda 2^3 z_{51}]$$

$$\dddot{Z}_{rl}^B = B^T A^T A^T A^T [z_{1.}] =$$
$$[\epsilon\lambda 1^2 z_{11} + (\epsilon\lambda 1^3 + 1) z_{12} + \lambda 2 z_{13} + \lambda 2^2 z_{14} + \lambda 2^3 z_{15}]$$

We get the following systems, written in block diagonal form.

Each state vector is divided into two parts, a known portion x_k^p which contains the given position y_k and an unknown portion x_k^v that contains the parameters $\dot{y}_k$, u_k, $\dot{u}_k$ and $\ddot{u}_k$. All right sides consist of known variables and are therefore constant vectors.

Continuous control signal u:

$$\begin{bmatrix} Z_{ll}^B & -W_l^B & Z_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & Z_{ll}^B & -W_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -W_l^B & Z_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & Z_{ll}^B & -W_l^B & Z_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -Z_{rl}^B & W_r^B & -Z_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -Z_{rl}^B & W_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & W_r^B & -Z_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -Z_{rl}^B & W_r^B & -Z_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Continuous first derivative of control signal, $\dot{u}$:

$$\begin{bmatrix} \dot{Z}_{ll}^B & -\dot{W}_l^B & \dot{Z}_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & \dot{Z}_{ll}^B & -\dot{W}_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\dot{W}_l^B & \dot{Z}_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & \dot{Z}_{ll}^B & -\dot{W}_l^B & \dot{Z}_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\dot{Z}_{rl}^B & \dot{W}_r^B & -\dot{Z}_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -\dot{Z}_{rl}^B & \dot{W}_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dot{W}_r^B & -\dot{Z}_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -\dot{Z}_{rl}^B & \dot{W}_r^B & -\dot{Z}_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Continuous second derivative of control signal, $\ddot{u}$:

$$\begin{bmatrix} \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Continuous third derivative of control signal, $\dddot{u}$:

$$\begin{bmatrix} \dddot{Z}_{ll}^{B} & -\dddot{W}_{l}^{B} & \dddot{Z}_{lu}^{B} & \dots & 0 & 0 & 0 \\ 0 & \dddot{Z}_{ll}^{B} & -\dddot{W}_{l}^{B} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\dddot{W}_{l}^{B} & \dddot{Z}_{lu}^{B} & 0 \\ 0 & 0 & 0 & \dots & \dddot{Z}_{ll}^{B} & -\dddot{W}_{l}^{B} & \dddot{Z}_{lu}^{B} \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\dddot{Z}_{rl}^{B} & \dddot{W}_{r}^{B} & -\dddot{Z}_{ru}^{B} & \dots & 0 & 0 & 0 \\ 0 & -\dddot{Z}_{rl}^{B} & \dddot{W}_{r}^{B} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dddot{W}_{r}^{B} & -\dddot{Z}_{ru}^{B} & 0 \\ 0 & 0 & 0 & \dots & -\dddot{Z}_{rl}^{B} & \dddot{W}_{r}^{B} & -\dddot{Z}_{ru}^{B} \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

Having totally 4(n+1) unknowns but only 4(n-1) constraints we chose to enter differential approximations for the first and last state vector, this will decrease the number of unknown parameters by eight and thus make the problem solvable. Remember that the time interval $t_{k+1} - t_k$ is constant and represented below as h.

The needed velocities are approximated as:

$$\dot{y}_0 = \frac{y_1 - y_0}{h}$$

$$\dot{y}_n = \frac{y_n - y_{n-1}}{h}$$

The needed control signals are approximated as:

$$\dot{y}_1 = \frac{y_2 - y_1}{h}$$

$$\dot{y}_{n-1} = \frac{y_{n-1} - y_{n-2}}{h}$$

$$u_0 = \frac{\dot{y}_0 - \dot{y}_1}{h}$$

$$u_n = \frac{\dot{y}_{n-1} - \dot{y}_n}{h}$$

The needed first derivative of the control signals is approximated as:

$$\dot{y}_2 = \frac{y_2 - y_1}{h}$$

$$\dot{y}_{n-2} = \frac{y_{n-2} - y_{n-3}}{h}$$

$$u_1 = \frac{\dot{y}_1 - \dot{y}_2}{h}$$

$$u_{n-1} = \frac{\dot{y}_{n-2} - \dot{y}_{n-1}}{h}$$

$$\dot{u}_0 = \frac{u_0 - u_1}{h}$$

$$\dot{u}_n = \frac{u_{n-1} - u_n}{h}$$

The needed second derivative of the control signals is approximated as:

$$\dot{y}_3 = \frac{y_3 - y_2}{h}$$

$$\dot{y}_{n-3} = \frac{y_{n-3} - y_{n-4}}{h}$$

$$u_2 = \frac{\dot{y}_2 - \dot{y}_3}{h}$$

$$u_{n-2} = \frac{\dot{y}_{n-3} - \dot{y}_{n-2}}{h}$$

$$\dot{u}_1 = \frac{u_1 - u_2}{h}$$

$$\dot{u}_{n-1} = \frac{u_{n-2} - u_{n-1}}{h}$$

$$\ddot{u}_0 = \frac{\dot{u}_0 - \dot{u}_1}{h}$$

$$\ddot{u}_n = \frac{\dot{u}_{n-1} - \dot{u}_n}{h}$$

The Matlab program that solves the task for system 3 using Gaussian elimination is displayed in mpr151knovel.m.

7 The Test

The test was effected by forcing the current system to run through points situated on the curve we wanted to track. This denotes that the trajectory has total freedom between the fixed coordinates as long as it passes through the test curve dots. How close the system followed the test curve was measured at five additional points between each two fixed coordinates. All these extra measuring points, along the curve, were added by their absolute value. The sum of all these measurements is called total-error.

Point-error shows how precisely the system tracks the fixed coordinates. During all the trials, this value always been zero, i.e. perfect tracking. The only fixed coordinate that the system does not run through is the end point. It is due to the lack of constraints there and its divergence is measured by End-point-error.

7.1 Test curves

Three different kinds of test curves with diverse characteristics were used. The two standard curves are one period of the sharply curving sine function and the soft curving function, the hyperbolic tangent. A discontinuous step function was also used in the test, (see below).

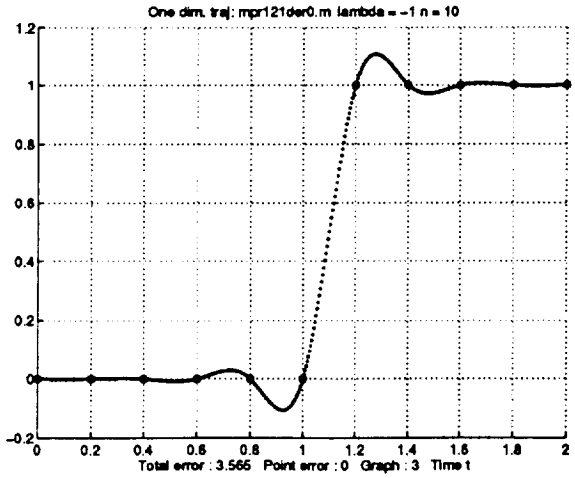


Figure 1: The step function tracked by system 1 with λ=0.

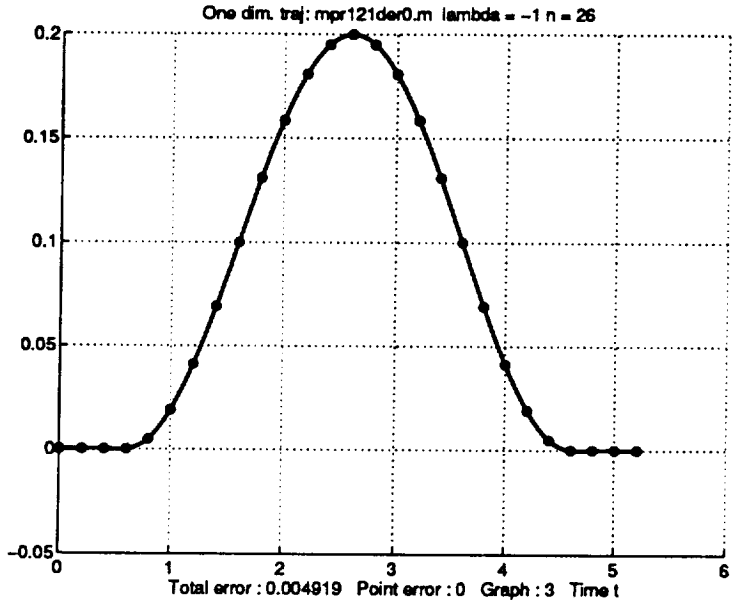


Figure 2: The sine function tracked by system 1 with λ=0.

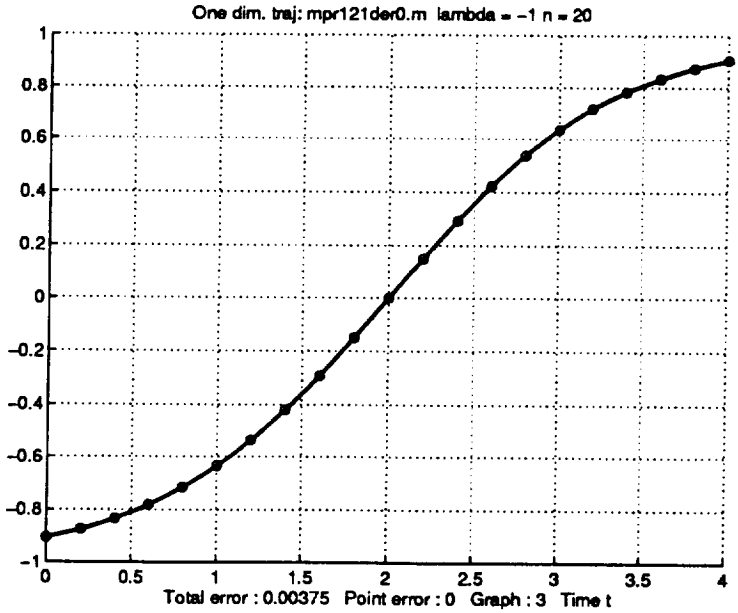


Figure 3: The tangent hyperbolic function tracked by system 1 with λ=0.

8 Results

In this chapter we are going to look at the output from our systems. The following figures are exactly like those shown on the computer screen.

At the top left of the graph the program used is exhibited. At the top right are the values of $\lambda 1$, l1, $\lambda 2$, l2, and ϵ, ep, shown together with the number of points, n, used to determine the test function.The scale of the y-axis are indeterminable but can give a hint about the ratio between variables of the same kind. Which quantity the plot gives information about is displayed to the left of the axis. At the bottom is always the time axis, scaled in seconds, displayed jointly with the calculated errors, see chapter 7.

At first the difference between system 1 and the other two systems will be shown. The two systems can be represented by system 3 since they have about the same behavior for this choice of parameters. Notice the sharper look of system 1's control signal u and acceleration, the latter has less maximal deviation in both graphs. System 1 can not be affected in the same way

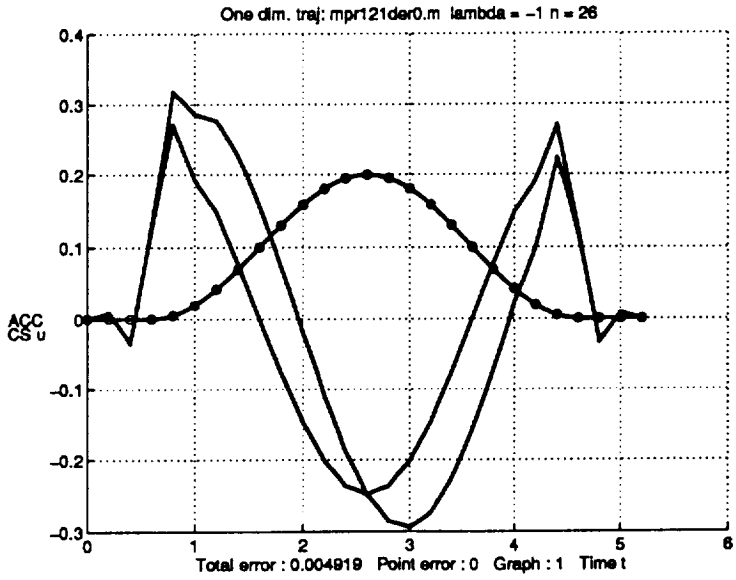


Figure 4: Sine tracked by system 1.

as the other two systems. Due to this it is not so very interesting and will therefore from now on be omitted.

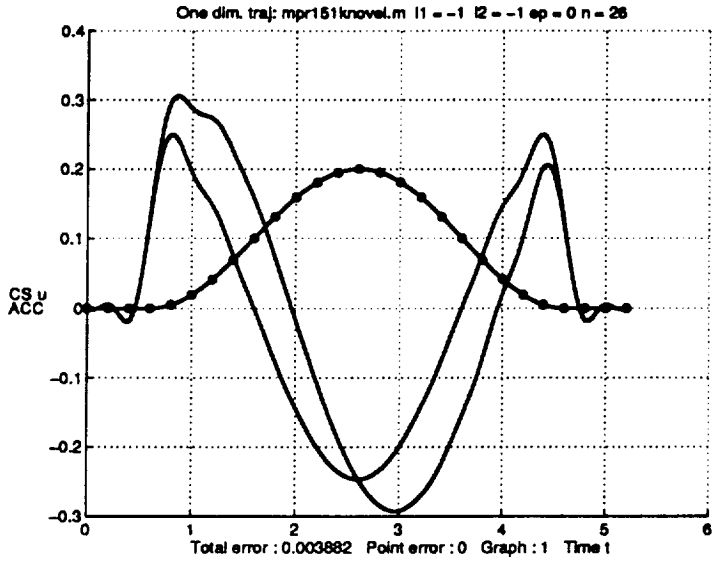


Figure 5: Sine tracked by system 3.

System 2's behavior when tracking the soft curving function tangent hyperbolic is shown below. The first graph shows the smooth behavior for the system when $\epsilon = 0$. The other two indicate a more uncontrolled fluctuation for an $\epsilon \neq 0$. The second and the third test are done with different ϵ so the received maximum deviation for the acceleration felt by the passengers are of the same magnitude. The acceleration is at the bottom except when it is oscillating heavily as in the last two plots.

Opposite the analysis made in chapter 5 it seems as though system 2 is not as easy to handle even for an $\epsilon > 0$ and its oscillations for $\epsilon < 0$ are apparent.

In two plots some parts of the curves are omitted. This is to prevent the large deviation of the acceleration at the endpoints to suppress other important information. However, this makes a correct error estimation impossible and the displayed values on the total error for these plots are wrong. The true value on the total error is 0.0997 for figure 7 and 0.1967 for figure 9. The last mentioned plot shows the influence of a changed value on $\lambda 2$. It increases the magnitude and shifts the oscillating acceleration to an earlier time interval. For both systems it is true that $\lambda 1$ affects the behavior much more than what $\lambda 2$ does.

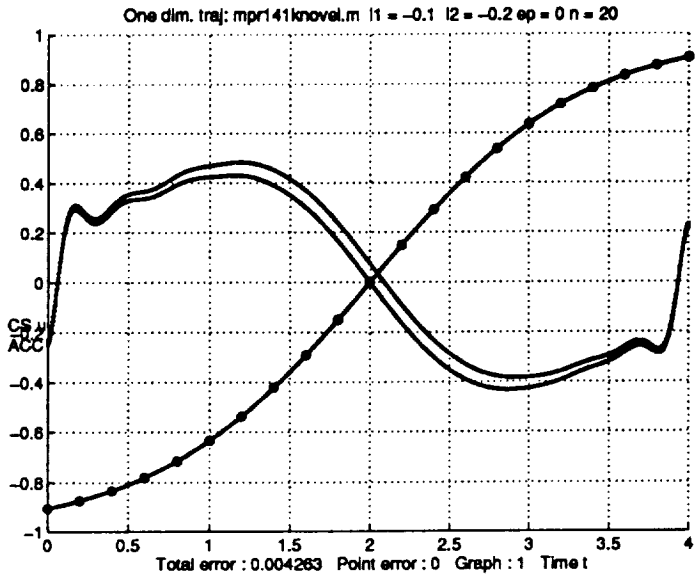


Figure 6: Tangent hyperbolic tracked by system 2.

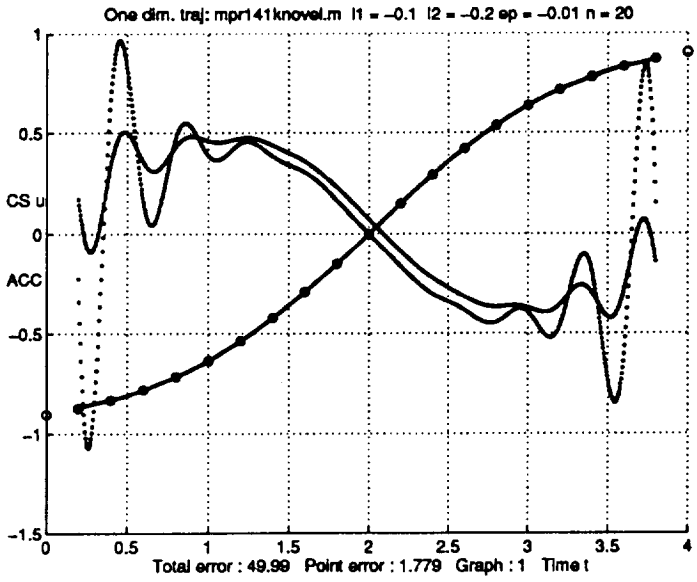


Figure 7: Tangent hyperbolic tracked by system 2.

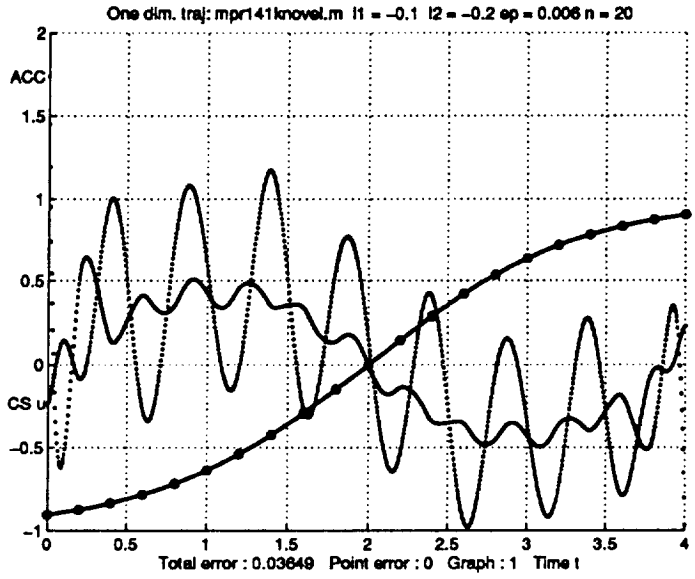


Figure 8: Tangent hyperbolic tracked by system 2.

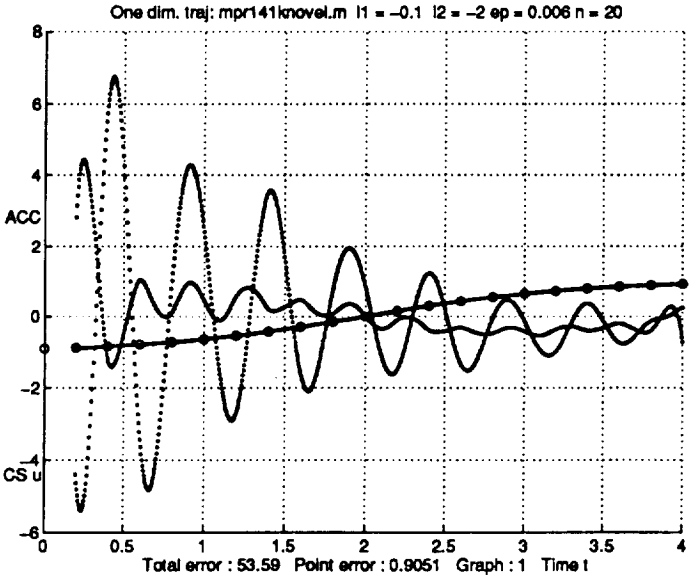


Figure 9: Tangent hyperbolic tracked by system 2.

System 3's conduct for $\lambda 1 = \lambda 2 =$ -1 and varying values on ϵ, when tracking the tangent hyperbolic function, is shown below.

According to the theoretical discussion in chapter 5, we stated that system 3 has two zeros in the open right half plane for an $\epsilon > 0$ but only one for an $\epsilon < 0$. We see that the system can handle negative ϵ better than positive, (see figure 11 and 12).

The curve that shows the acceleration is mostly beneath the control signal u in all graphs. It is also the most oscillating signal in the two last plots.

The tests using the tangent hyperbolic function are carried out with different sets of λ for each systems. It is therefore not so easy to compare the behavior for the actual systems. However, during experiments that are not presented in the report, it has been shown that system 3 is more easily disturbed for an $\epsilon \neq 0$ than system 2 is.
Correct total error for figure 11 is 0.2435 and 0.3126 for figure 12.

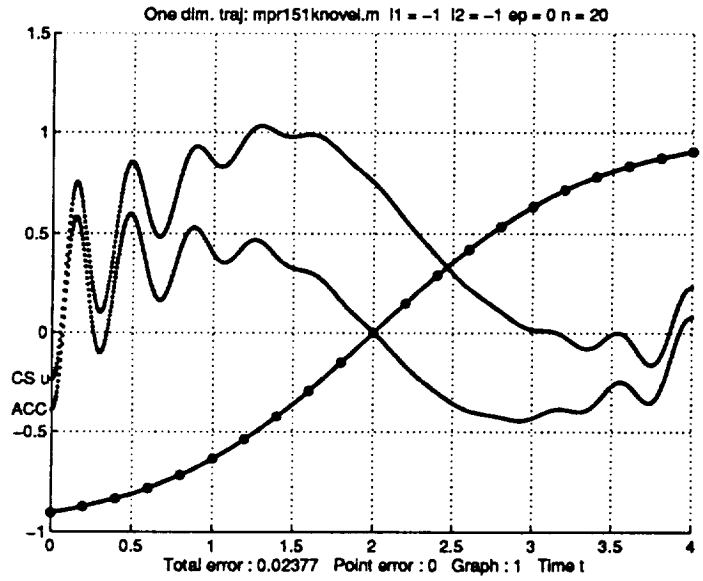


Figure 10: Tangent hyperbolic tracked by system 3.

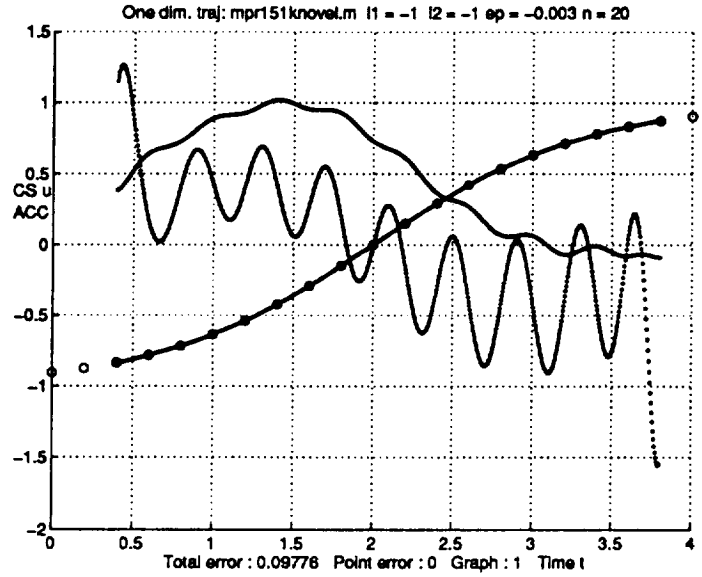


Figure 11: Tangent hyperbolic tracked by system 3.

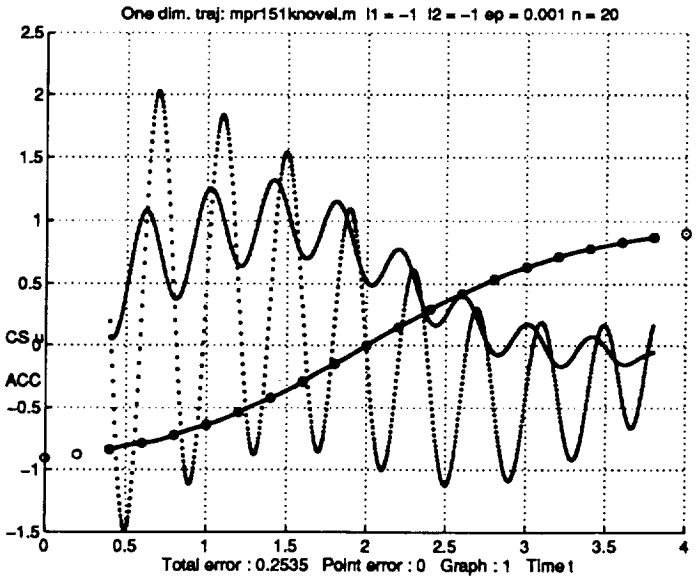


Figure 12: Tangent hyperbolic tracked by system 3.

System 3's characteristics are further examined below when it is applied to the sine curve. Figure 5 shows the obtained control signal u and acceleration for $\lambda 1 = \lambda 2 =$ -1 and $\epsilon = 0$. The first graph below displays the control signal w for the same system and parameters. This is to exhibit the calculated signals for a smooth case so we have something to compare with when it is getting rough.

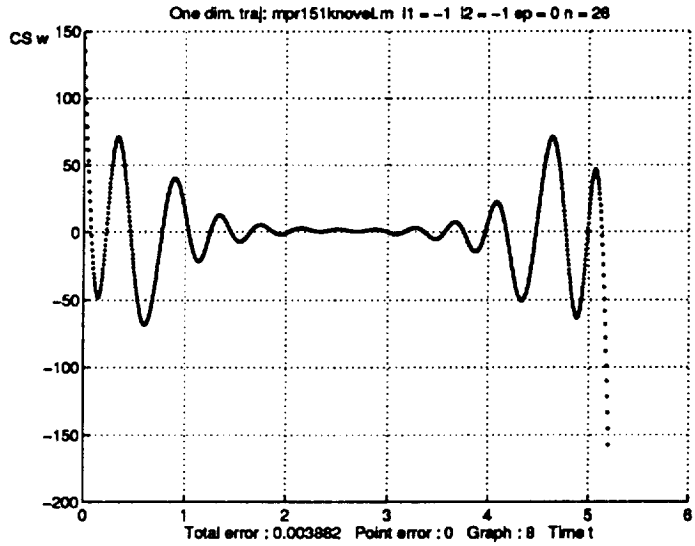


Figure 13: Sin tracked by system 3.

We receive some other signals in the following figures when system 3 is run by an $\epsilon \neq 0$. Observe the twisted trajectory in figure 15 with an accompanying large value on the total error. Notice also the magnitude on the control signal w and its third derivative, shown in figures reffi:16 and 17. The system seems to be more sensitive for an $\epsilon > 0$ than for an $\epsilon < 0$, this is probably due to the circumstance that it has two zeros in the open right half plane in the first case but only one zero in the second case. The oscillating acceleration is shifted to a later time interval and has there a larger magnitude for a negative ϵ.

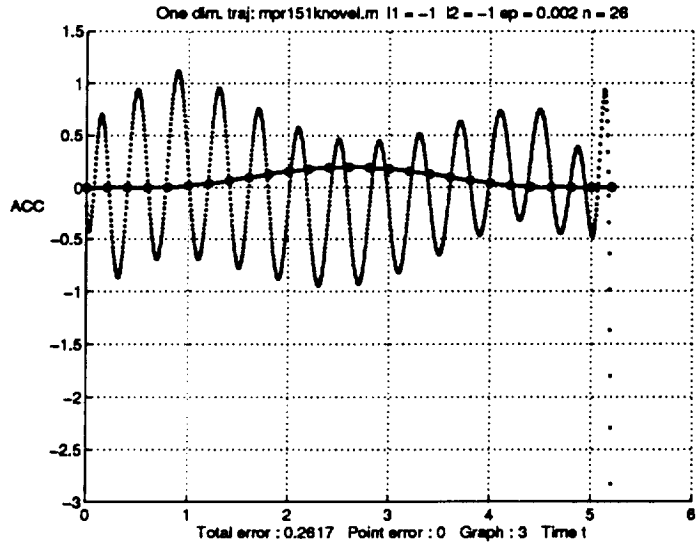


Figure 14: Sine tracked by system 3.

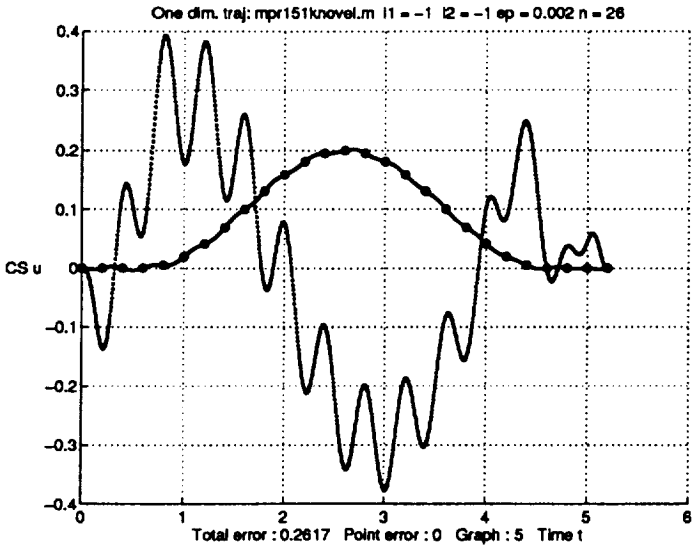


Figure 15: Sine tracked by system 3.

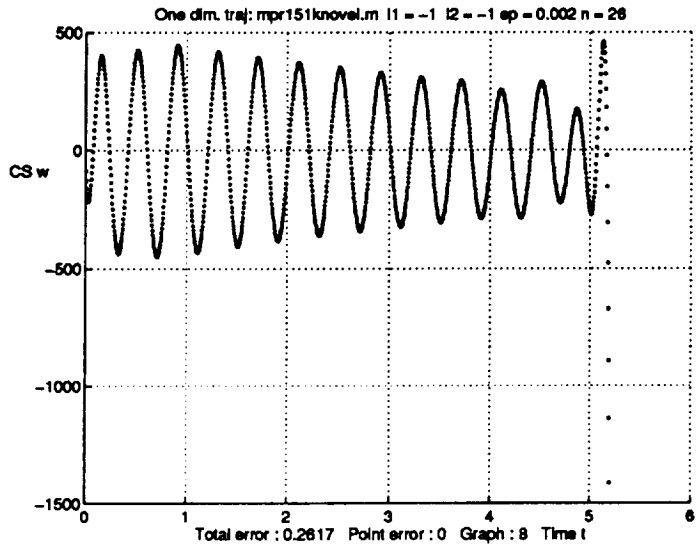


Figure 16: Sine tracked by system 3.

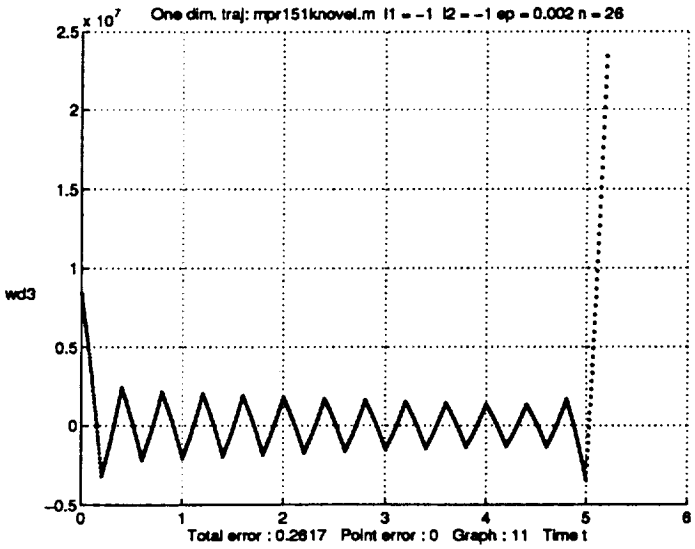


Figure 17: Sine tracked by system 3.

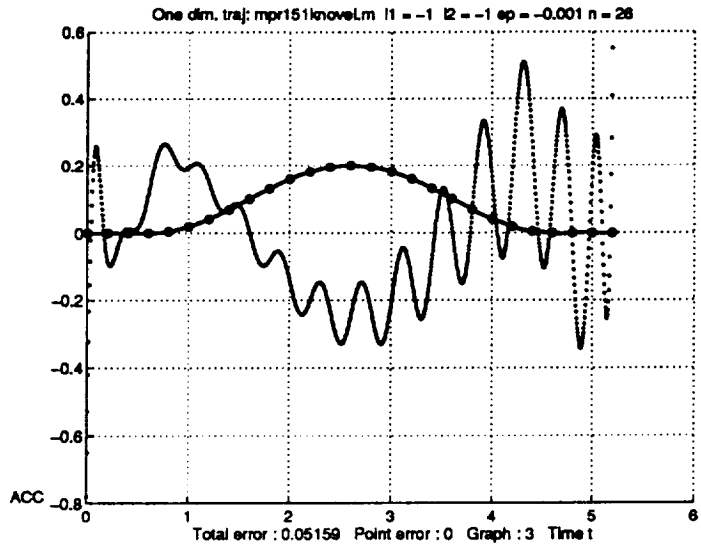


Figure 18: Sine tracked by system 3.

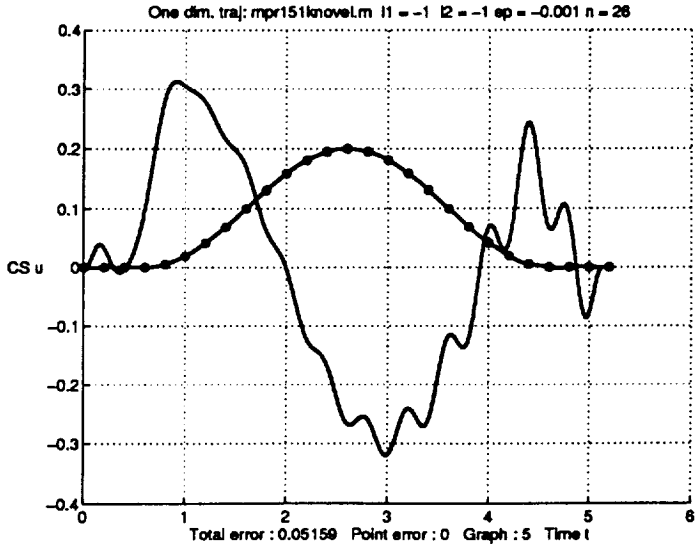


Figure 19: Sine tracked by system 3.

Applying the step function to system 3 gives that the very large control signals u can be reduced by an $\epsilon \neq 0$ to the cost of a larger total error.

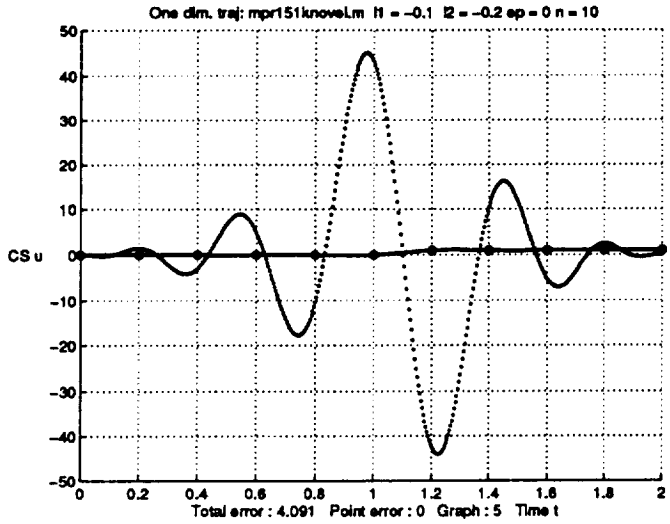


Figure 20: The step function tracked by system 3.

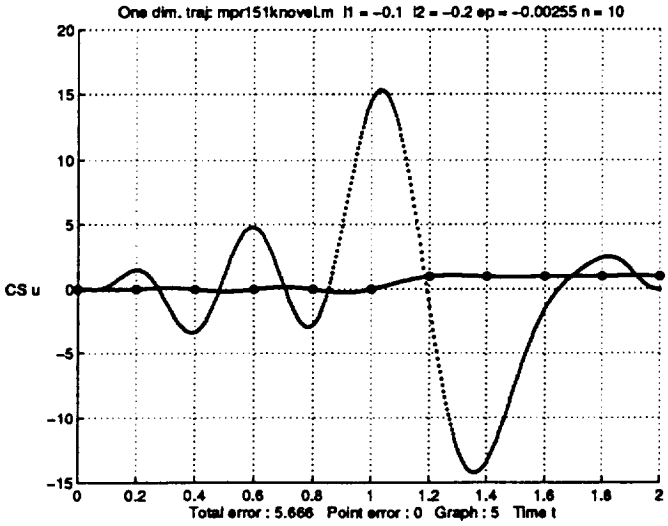


Figure 21: The step function tracked by system 3.

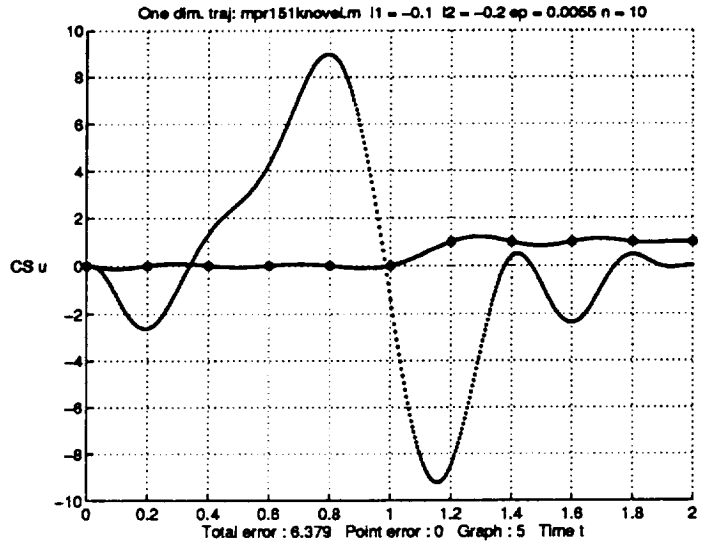


Figure 22: The step function tracked by system 3.

At last we will take a look at system 2 applied to the sine function with $\lambda 1$ = $\lambda 2$ = -1 and for a very particular choice of ϵ. Figures 23 and 24 show an oscillating but quite normal behavior for this system. When examining figure 24 which is run with a slightly changed ϵ it looks about as the other two until the scale on the y-axis is observed. The first two graphs use an ϵ = 0.004186 respectively ϵ = 0.004188. The value of ϵ for the last three figures is 0.00418702471143, which gave the largest oscillatory motions. It seems that we have found a set of parameters that brings system 2 in to resonance. When changing the value on ϵ we might affect the transfer function so that the frequency of the actual in signal w is the peak frequency; see figure 27. In such a case we will receive an increased amplification of the output signal. Notice that this phenomenon only appears when tracking the sine test function.

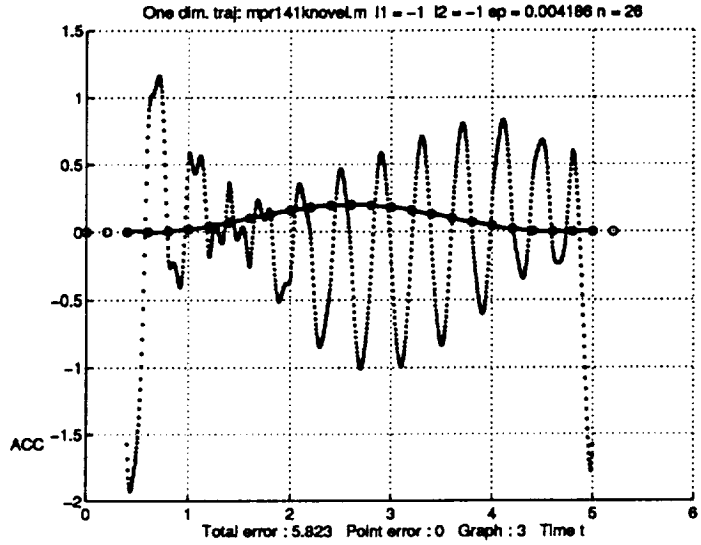


Figure 23: Sine tracked by system 2.

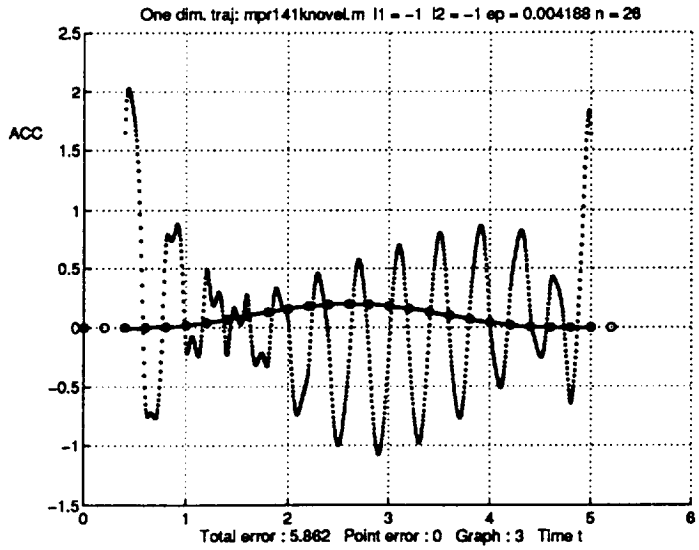


Figure 24: Sine tracked by system 2.

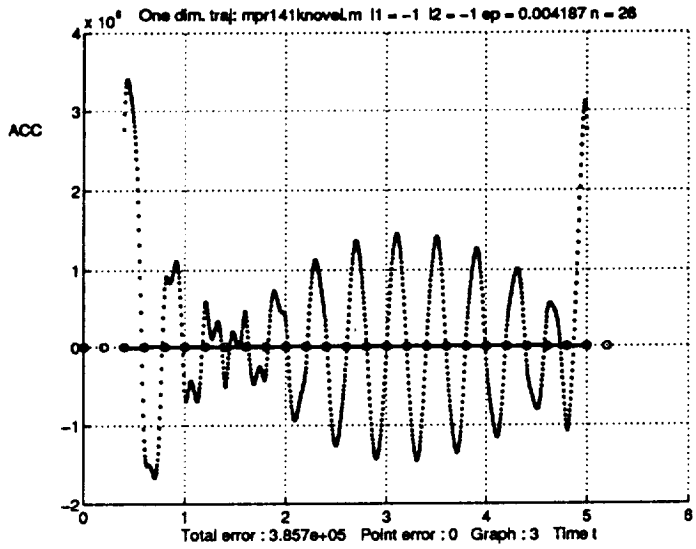


Figure 25: Sine tracked by system 2.

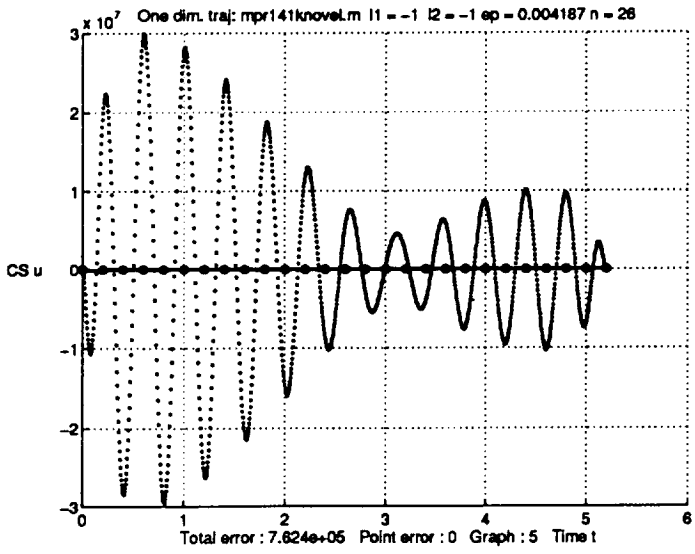


Figure 26: Sine tracked by system 2.

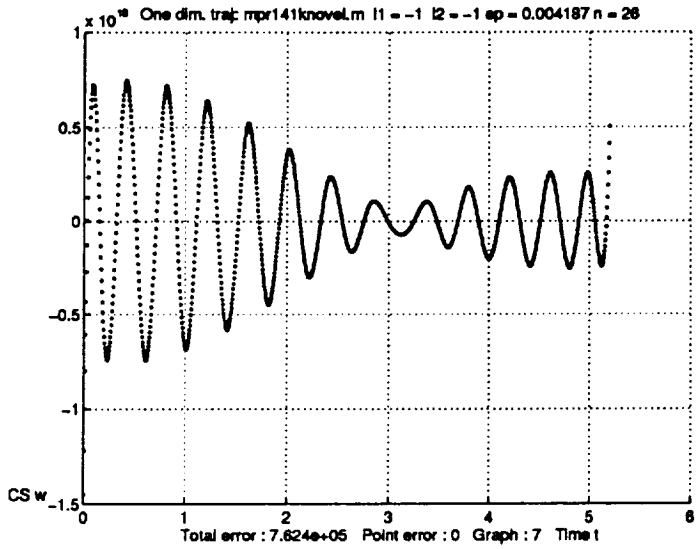


Figure 27: Sine tracked by system 2.

8.1 Surprising Results

Looking at figure 4 or 5 one might be surprised at the appearance of the control signal u and the acceleration. Even if the sine function is curving it should not cause such peaks in the graphs. The reason for this is as follows.

Consider the system

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \qquad x = \begin{pmatrix} y \\ \dot{y} \end{pmatrix}$$

From the given system we have that $\dot{x}_1 = x_2$ and that $\dot{x}_2 = u$. If we use splines and force the system to track the function *f(t)* it implies that the first element in the state vector, x_1, equals the function, i.e. $y(t) = f(t)$. This denotes that $x_2(t) = \dot{f}(t)$ and that the control signal $u(t) = \dot{f}(t)$. The velocity in figure 28 should by our theory have something in common with the derivative of the curve $f(t) = \sin t$. The derivative $\dot{f}(t) = \cos t$, so in the beginning of the first region when *f(t) = -1*, the graph is shifted upwards and scaled, the velocity should be zero. When *f(t)* becomes zero the velocity ought to reach its maximum and then decline. For the other half we have the reversed situation and should therefore have an inverted curve. This behavior

can be recognized in the first graph and a similar reasoning for the control signal u equals $\ddot{f}(t) = -\sin t$ gives the calculated control signal in figure 4. Here $\lambda \neq 0$ so the graph is shifted to the right. So the splines do not only try to follow the specified trajectory they also approximates its derivatives. We can thus effect a perfect trajectory of the test curve but the prize we pay is huge values on the control signals and accompanying acceleration. The behavior described makes it impossible to realize a smooth aircraft control using splines.

This very simplified heuristic description of the phenomenon will be completed in a paper written at Texas Tech University, Lubbock, USA, by Horn Professor Clyde Martin, PhD, and Assistant Professor Zhimin Zhang, PhD.

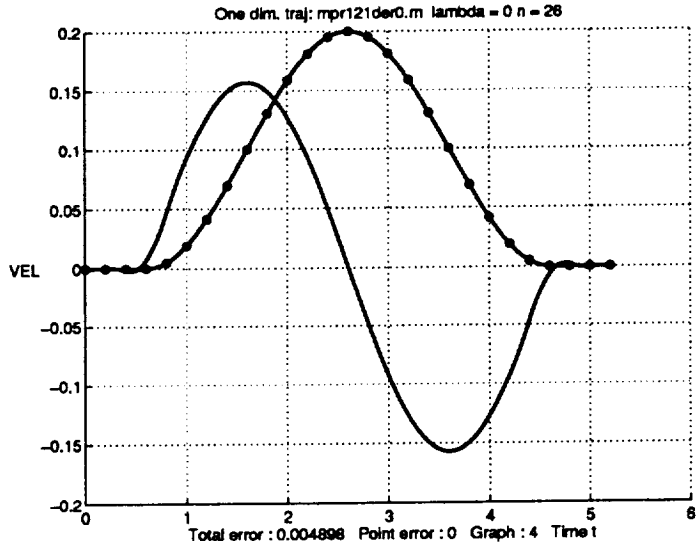


Figure 28: Sine tracked by system 1.

9 Resume in Swedish

Inledningsvis var våran avsikt att ta fram styrlagar för en flygplans modell så att den flögs så behagligt för passagerarna som möjligt. Passagerar komforten var det allra mest väsentliga så vi utvecklade två kontroll lagar som använde derivatorna av den pilot inducerade styrningen u. Detta ger inte den energi snålaste insignalen men tar bort de värsta topparna hos styrsignalen och medföljande acceleration.

Programvaran som använts inkluderar Matlab och Maple för beräkningar och Latex som ordbehandlings program. Andra hälften av rapporten består av Matlab program och som avslutas med en referenslista.

Vi ansatte den vanliga endimensionella behandlings proceduren och implementerade systemen i Matlab. Ett huvud program, med tillhörande hjälp program, för varje kontroll lag.

De givna systemen analyserades ur uppnåbarhets och stabilitets synpunkt vilket resulterade i en bedömning att de var stabila men inte garanterade insignal-utsignal stabilitet. Se kapitel 5.

En "spline" är en kurva till ett n:te gradens polynom vilken är förenat med liknande polynoms kurvor i respektive ändpunkt. I varje förenings punkt har funktionerna sina n-1 första derivator gemensammma. Detta ger en kurva som av ögat tycks vara helt homogen men som i själva verket består av ett antal sammankopplade delar. Kapitel 6 behandlar "splines" och den användna kontroll teorin.

Huvud resultatet var att vi inte kunde ta fram mjuka styrlagar eftersom "splinsen" inte bara försöker approximera test kurvan utan även tar hänsyn till dess derivator. Genom systemet påverkas även styrsignalen och vi erhåller omöjligt stora styrsignaler och accelerationer. Noggranheten i följningen är alltid exemplarisk. Kapitel 8 behandlar rapportens huvudresultat.

10 Matlab Programs

```
function x = mpr121der0(spc_plot,t,n,cleargr,pointfcn,lambda)

%%  THIS PROGRAM CALCULATES CONTROLLAWS FOR A ONE
    DIMENSIONAL TRAJECTORY  %%
%%  spc_plot DETERMINES WHICH GRAPH TO BE DISPLAYED,
    CHOSE AN INTEGER =<5  %%
%%  t IS THE TIME PERIOD FOR WHICH THE SYSTEM
    IS TO BE CONTROLLED  %%
%%  n ARE THE NUMBER OF POINTS AT THE SPECIFIED
    TRAJECTORY, CHOSE t/n>1/10  %%
%%  cleargr CLEARS THE CURRENT WINDOW, CHOSE 1 OR 0  %%
%%  pointfcn SPECIFIES THE TRAJECTORY  %%
%%  lambda AFFECTS THE INSTABILITY OF THE SYSTEM  %%

global A B h t n

%%  THE SYSTEM  %%

A=[ 0 1 ;
    0 lambda];
B=[ 0 ;
    1 ];
C=[ 1 0 ];

%%  FUNCTION pointfcn DETERMINES THE SPECIFIED TRAJECTORY  %%

%%  NECESSARY FOR THE COMPOUND FUNCTION pointsin12  %%

if pointfcn=='pointsin12';
  t=5.2;
  n=26;
end;

R=feval(pointfcn);
```

```
%%  CALCULATION OF THE INTEGRAL FROM 0 TO h  %%

m=48;         %%  NUMBER OF POINTS BETWEEN INTERPOLATIONS,
                  CHOSE A MULTIPLE OF 6  %%
mp=m/6;       %%  DETERMINES THE PRECISION IN THE
                  SPLINE APPROXIMATION  %%
tol=1e-08;    %%  THE NUMERIC ERROR TOLERANCE  %%

Mtau(:,1:2)=zeros(2);
tau=0;
for j=1:m
  oldtau=tau;
  tau=oldtau+h/m;
  Mtau(:,2*j+1:2*j+2)= quad812mod('integrand',oldtau,tau,tol)
 + Mtau(:,2*j-1:2*j);
end;
M=Mtau(:,2*m+1:2*m+2);

e_Ah=expm(-A*h);
Minv=inv(M);
ZZ=Minv*e_Ah;
WW=e_Ah'*ZZ+Minv;

WL=[WW(2,2)];   %%  PARTITIONING MATRICES  %%
ZLU=[ZZ(2,2)];
ZLL=[ZZ(2,2)];
WR=[WW(2,1)];
ZRU=[ZZ(2,1)];
ZRL=[ZZ(1,2)];
Ulu=[Minv(2,2)];
Uru=[Minv(2,1)];

%%  FORMING OF THE RIGHT HAND SIDE OF THE BLOCKDIAGONAL
    SYSTEM  %%

for i=2:n
  Omega(i,1)=-ZRL*R(1,i-1)+WR*R(1,i)-ZRU*R(1,i+1);
```

```
end;
Omega(1,1)= Uru*R(1,1) - ZRU*R(1,2);
Omega(n+1,1)=ZRL*R(1,n) + (Uru-WR)*R(1,n+1);

%%  FORMING OF THE LEFT HAND SIDE OF THE BLOCKDIAGONAL
    SYSTEM  %%

for i=2:n
  DD(i,i-1)=ZLL;
  DD(i,i)=-WL;
  DD(i,i+1)=ZLU;
end;
DD(1,1)=(lambda-Ulu);
DD(1,2)=ZLU;
DD(n+1,n)=-ZLL;
DD(n+1,n+1)=(lambda+WL-Ulu);

DD=sparse(DD);      %%  SQUEEZING OUT ALL ZERO ELEMENTS
                        FROM MATRIX DD  %%

%%  GAUSSELIMINATION TO PRODUCE AN UPPER
    TRIANGULAR SYSTEM  %%

for i=1:n
  zd=DD(i+1,i)/DD(i,i);
  DD(i+1,i)=DD(i+1,i)-zd*DD(i,i);
  DD(i+1,i+1)=DD(i+1,i+1)-zd*DD(i,i+1);
  Omega(i+1,1)=Omega(i+1,1)-zd*Omega(i,1);
end;

%%  BACKSUBSTITUTION TO SOLVE FOR THE XVEL  %%

xvel(1,n+1)=Omega(n+1,1)/DD(i+1,i+1);
for i=n:-1:1
  xvel(1,i)=(Omega(i,1)-DD(i,i+1)*xvel(1,i+1))/DD(i,i);
end;
```

```
%%  MAKING OF THE STATE VECTOR  %%

for i=0:n
  x(:,i+1)=[[R(1,i+1)]
            [xvel(1,i+1)]];
end;

if cleargr
  clf;
  hold on;
  grid on;
end;

%%  PLOTTING OF THE CALC/SPEC TRAJECTORY,VELOCITY,
    CONTROLSIGNAL AND ACCELERATION  %%

plotadm=0;
while spc_plot
  if plotadm
    if spc_plot<6
      figure;
      hold on;
      grid on;
      title(['One dim. traj: mpr121der0.m  lambda = ',
      num2str(lambda),' n = ',num2str(n)])
      xlabel(['Total error : ',num2str(Total_error),'
      Pointerror : ',num2str(Point_error),'   Graph :
      ',num2str(spc_plot),'   Time t'])
      for k=0:n
        plot(k*h,x(1,k+1),'o')
      end;
    end;
  end;
  breakadm=0;
  fadm=0;    %%  NORMALLY fadm=0, NECESSARY FOR WRITING OF
                 TEXTS IN THE PLOT  %%
```

```
for j=0:m
  eAtau=expm(A*j*h/m);
  %% CHOOSE e.g 2:n-3 TO AVOID PROBLEMS AT THE ENDPOINTS %%
  for i=fadm:n-1
    entry(:,i+1)=eAtau*(x(:,i+1)+Mtau(:,2*j+1:2*j+2)*
    Minv*(e_Ah*x(:,i+2)-x(:,i+1)));
    csignvec(:,i+1)=B'*expm(-A'*j*h/m)*Minv*
    (e_Ah*x(:,i+2)-x(:,i+1));
    if j==0
      if i==fadm;
        entry1=entry(1,fadm+1);
        entry2=entry(2,fadm+1);
        csignvec1=csignvec(1,fadm+1);
      end;
    end;
    if rem(j,mp)==0
      if j<=m-mp
        traject(1,j/mp*n+(i+1))=entry(1,i+1);
      end;
    end;
    if j==m
      if i==n-1
        traject(1,6*n+1)=entry(1,i+1);
      end;
    end;
    if spc_plot==1
      plot(i*h+j*h/m,entry(1,i+1),'.')      %% TRAJECTORY %%
      plot(i*h+j*h/m,csignvec(1,i+1),'.')   %% CONTROL u %%
      %% ACCELERATION %%
      plot(i*h+j*h/m,lambda*entry(2,i+1)+
      csignvec(1,i+1),'.')
    elseif spc_plot==2
      plot(i*h+j*h/m,entry(1,i+1),'.')      %% TRAJECTORY %%
      plot(i*h+j*h/m,entry(2,i+1),'.')      %% VELOCITY %%
      plot(i*h+j*h/m,csignvec(1,i+1),'.')   %% CONTROL u %%
      %% ACCELERATION %%
      plot(i*h+j*h/m,lambda*entry(2,i+1)+
```

```
      csignvec(1,i+1),'.')
    elseif spc_plot==3
      plot(i*h+j*h/m,entry(1,i+1),'.')     %% TRAJECTORY %%
      %% ACCELERATION %%
      plot(i*h+j*h/m,lambda*entry(2,i+1)+
      csignvec(1,i+1),'.')
    elseif spc_plot==4
      plot(i*h+j*h/m,entry(1,i+1),'.')     %% TRAJECTORY %%
      plot(i*h+j*h/m,entry(2,i+1),'.')     %% VELOCITY %%
    elseif spc_plot==5
      plot(i*h+j*h/m,entry(1,i+1),'.')     %% TRAJECTORY %%
      plot(i*h+j*h/m,csignvec(1,i+1),'.')  %% CONTROL u %%
    else
      disp(' ')
      disp(' NOT A VALID CHOICE ')
      breakadm=1;
    end;
    if breakadm
      break;
    end;
  end;
  if breakadm
    break;
  end;
end;
if spc_plot<6
  ax=axis;
  ax2=ax(1,2);
  if ax2<1.5
    xpos=3/4*0.2;
  elseif ax2>=5
    xpos=3/4;
  else
    xpos=0.25;
  end;
  if spc_plot<3
    ax4=ax(1,4);
```

```
  if ax4<1
    ax4=ax4*10;
    if ax4<1
      ax4=ax4*10;
    end;
  end;
  if ax4>10
    ax4=ax4/10;
    if ax4>10
      ax4=ax4/10;
    end;
  end;
  if rem(ax4,2)==0
   yadd=(ax(1,4)/4)*1/5;
  else
    yadd=(ax(1,4)/3)*1/5;
  end;
  ypos=lambda*entry2+csignvec1;
  text(-xpos,ypos,['ACC']);
  if abs(ypos-csignvec1)>yadd
    text(-xpos,csignvec1,['CS u']);
  elseif ypos-csignvec1>0
    text(-xpos,csignvec1-yadd,['CS u']);
  else
    text(-xpos,csignvec1+yadd,['CS u']);
  end;
  if spc_plot==2
    text(-xpos,entry2,['VEL']);
  end;
end;
if spc_plot==3
  ypos=lambda*entry2+csignvec1;
  text(-xpos,ypos,['ACC']);
end;
if spc_plot==4
  text(-xpos,entry2,['VEL']);
end;
```

```
    if spc_plot==5
      text(-xpos,csignvec1,['CS u']);
    end;
  end;

  %%  ESTIMATION OF THE ACCURACY IN THE SPLINE
      APPROXIMATION  %%

  if plotadm==0
    [Total_error,Average_error,Point_error,End_point_error]=
    spline_error(pointfcn,R,traject);
    Total_error
    Average_error
    Point_error
    End_point_error
    title(['One dim. traj: mpr121der0.m  lambda = ',num2str
    (lambda),' n = ',num2str(n)])
    xlabel(['Total error : ',num2str(Total_error),'   Point
    error : ',num2str(Point_error),'   Graph : ',
    num2str(spc_plot),'   Time t'])
    if spc_plot<6
      for k=0:n
        plot(k*h,x(1,k+1),'o')
      end;
    end;
  end;

  plotadm=plotadm+1;
  disp(' ')
  disp(' WOULD YOU LIKE TO SEE ANOTHER GRAPH OF THE CURRENT
  SYSTEM AND ITS TRAJECTORY ? ')
  disp(' ')
  disp(' FINISH THE PROGRAM : 0 ')
  disp(' DISPLAY THE TRAJECTORY, CONTROL u AND
  ACCELERATION : 1 ')
  disp(' DISPLAY THE TRAJECTORY, VELOCITY, CONTROL u AND
  ACCELERATION : 2 ')
```

```
  disp(' DISPLAY THE TRAJECTORY AND ACCELERATION : 3 ')
  disp(' DISPLAY THE TRAJECTORY AND VELOCITY : 4 ')
  disp(' DISPLAY THE TRAJECTORY AND CONTROL u : 5 ')
  spc_plot = input(' MAKE YOUR CHOICE : ');
end;

clear global;
for k=2:plotadm
  delete(k);
end;
end;
```

```
function [Q,cnt] = quad812mod(funfcn,a,b,tol)
%Alteration of the original matlab toolbox program.
%QUAD8 Numerical evaluation of an integral, higher order
%    method. Q = QUAD8('F',A,B,TOL) approximates the
%    integral of F(X) from to B to within a relative error
%    of TOL.  'F' is a string containing the name of the
%    function. The function must return a  2*2-matrix
%    output value if given an input value.
%    Q = Inf is returned if an excessive recursion level
%    is reached indicating a possibly singular integral.
%    QUAD8 uses an adaptive recursive Newton Cotes 8 panel
%    rule.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.
%    [Q,cnt] = quad8(F,a,b,tol) also returns a function
%    evaluation count.
%    Top level initialization, Newton-Cotes weights
w = [3956 23552 -3712 41984 -18160 41984 -3712 23552
    3956]/14175;

x = a + (0:8)*(b-a)/8;

% set up function call
```

```
for i=x
  y = [y feval(funfcn,i)];
end;

% Adaptive, recursive Newton-Cotes 8 panel quadrature
Q0 = zeros(2);
[Q,cnt] = quad812stpmod(funfcn,a,b,tol,0,w,x,y,Q0);
cnt = cnt + 9;
end;
```

```
function [Q,cnt] = quad812stpmod(FunFcn,a,b,tol,lev,
                                 w,x0,f0,Q0)
%Alteration of the original matlab toolbox program.
%QUAD8STP Recursive function used by QUAD8.
%    [Q,cnt] = quad8stp(F,a,b,tol,lev,w,f,Q0) tries to
%    approximate the integral of f(x) from a to b to
%    within a relative error of tol. F is a string
%    containing the name of f. The remaining arguments
%    are generated by quad8mod or by the recursion.
%    lev is the recursion level.
%    w is the weights in the 8 panel Newton Cotes formula.
%    x0 is a vector of 9 equally spaced abscissa is the
%    interval.
%    f0 is a matrix of the 9 function values at x.
%    Q0 is an approximate value of the integral.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.

LEVMAX = 10;

% Evaluate function at midpoints of left and
  right half intervals.
x = zeros(1,17);
x(1:2:17) = x0;
x(2:2:16) = (x0(1:8) + x0(2:9))/2;
```

```
f(:,1:2)= f0(:,1:2);
for i=1:8
  f(:,4*i-1:4*i) = feval(FunFcn,x(2*i));
  f(:,4*i+1:4*i+2) = (f0(:,2*i+1:2*i+2));
end;

% Integrate over half intervals.
h = (b-a)/16;
Q1=0;Q2=0;
for i=1:9
  Q1 = Q1 + h*w(i)*f(:,2*i-1:2*i);
  Q2 = Q2 + h*w(10-i)*f(:,35-i*2:36-i*2);
end;
Q = Q1 + Q2;
% Recursively refine approximations.
if norm(Q - Q0) > tol*norm(Q) & lev <= LEVMAX
   c = (a+b)/2;
   [Q1,cnt1] = quad812stpmod(FunFcn,a,c,tol/2,lev+1,
                             w,x(1:9),f(:,1:18),Q1);
   [Q2,cnt2] = quad812stpmod(FunFcn,c,b,tol/2,lev+1,
                             w,x(9:17),f(:,17:34),Q2);
   Q = Q1 + Q2;
   cnt = cnt + cnt1 + cnt2;
end
end;
```

```
function x = mpr141knovel(spc_plot,t,n,cleargr,pointfcn,
lambda1,lambda2,ep)

%%  THIS PROGRAM CALCULATES CONTROLLAWS FOR A ONE
    DIMENSIONAL TRAJECTORY  %%
%%  spc_plot DETERMINES WHICH GRAPH TO BE DISPLAYED,
    CHOSE AN INTEGER =<9  %%
%%  t IS THE TIME PERIOD FOR WHICH THE SYSTEM IS
    TO BE CONTROLLED  %%
%%  n ARE THE NUMBER OF POINTS AT THE SPECIFIED
    TRAJECTORY, CHOSE t/n>1/10  %%
%%  cleargr CLEARS THE CURRENT WINDOW, CHOSE 1 OR 0  %%
%%  pointfcn SPECIFIES THE TRAJECTORY  %%
%%  lambda1 AFFECTS THE INSTABILITY OF THE SYSTEM  %%
%%  lambda2 ALSO EFFECTS THE STABILITY OF THE SYSTEM,
    CHOSE l1~=l2  %%
%%  ep<>0 PUTS A ZERO IN THE TRANSFERFUNCTION  %%

global A B h t n

%%  THE SYSTEM  %%

A=[ 0 1 0 0;
    0 lambda1 1 0;
    0 0 0 1;
    0 0 0 lambda2];

B=[ 0 ep 0 1]';

C=[ 1 0 0 0];

%% FUNCTION pointfcn DETERMINES THE SPECIFIED TRAJECTORY %%

%%  NECESSARY FOR THE COMPOUND FUNCTION pointsin12  %%

if pointfcn=='pointsin12';
  t=5.2;
```

```
  n=26;
end;

R=feval(pointfcn);

%%  CALCULATION OF THE INTEGRAL FROM 0 TO h  %%

m=48;           %%  NUMBER OF POINTS BETWEEN INTERPOLATION,
                    CHOSE A MULTIPLE OF 6  %%
mp=m/6;         %%  NEEDED FOR fcn spline_error THAT
                    DETERMINES THE PRECISION IN THE
                    SPLINE APPROXIMATION  %%
tol=1e-08;      %%  THE NUMERIC ERROR TOLERANCE  %%

Mtau(:,1:4)=zeros(4);
tau=0;
for j=1:m
  oldtau=tau;
  tau=oldtau+h/m;
  Mtau(:,4*j+1:4*j+4)= quad814mod('integrand',oldtau,tau,tol)
  + Mtau(:,4*j-3:4*j);
end;
M=Mtau(:,4*m+1:4*m+4);

%%  FORMING OF THE MATRICES FOR THE BLOCKDIAGONAL SYSTEM  %%

e_Ah=expm(-A*h);
Minv=inv(M);
ZZ=Minv*e_Ah;
WW=e_Ah'*ZZ+Minv;

%%  CONTINUOUS CONTROLLAW  %%

WLD0=[ep*WW(2,2)+WW(4,2) ep*WW(2,3)+WW(4,3) ep*WW(2,4)+
      WW(4,4)];
ZLUD0=[ep*ZZ(2,2)+ZZ(4,2) ep*ZZ(2,3)+ZZ(4,3) ep*ZZ(2,4)+
       ZZ(4,4)];
```

```
ZLLD0=[ep*ZZ(2,2)+ZZ(2,4) ep*ZZ(3,2)+ZZ(3,4) ep*ZZ(4,2)+
       ZZ(4,4)];
WRD0=[ep*WW(2,1)+WW(4,1)];
ZRUD0=[ep*ZZ(2,1)+ZZ(4,1)];
ZRLD0=[ep*ZZ(1,2)+ZZ(1,4)];

%% CONTINUOUS FIRST DIFFERENTIAL OF CONTROLLAW %%

WLD1=[ep*WW(1,2)+ep*lambda1*WW(2,2)+WW(3,2)+lambda2*WW(4,2)
      ep*WW(1,3)+ep*lambda1*WW(2,3)+WW(3,3)+lambda2*WW(4,3)
      ep*WW(1,4)+ep*lambda1*WW(2,4)+WW(3,4)+lambda2*WW(4,4)];
ZLUD1=[ep*ZZ(1,2)+ep*lambda1*ZZ(2,2)+ZZ(3,2)+lambda2*ZZ(4,2)
       ep*ZZ(1,3)+ep*lambda1*ZZ(2,3)+ZZ(3,3)+lambda2*ZZ(4,3)
       ep*ZZ(1,4)+ep*lambda1*ZZ(2,4)+ZZ(3,4)+lambda2*ZZ(4,4)];
ZLLD1=[ep*ZZ(2,1)+ep*lambda1*ZZ(2,2)+ZZ(2,3)+lambda2*ZZ(2,4)
       ep*ZZ(3,1)+ep*lambda1*ZZ(3,2)+ZZ(3,3)+lambda2*ZZ(3,4)
       ep*ZZ(4,1)+ep*lambda1*ZZ(4,2)+ZZ(4,3)+lambda2*ZZ(4,4)];
WRD1=[ep*WW(1,1)+ep*lambda1*WW(2,1)+WW(3,1)+lambda2*WW(4,1)];
ZRUD1=[ep*ZZ(1,1)+ep*lambda1*ZZ(2,1)+ZZ(3,1)+lambda2*ZZ(4,1)];
ZRLD1=[ep*ZZ(1,1)+ep*lambda1*ZZ(1,2)+ZZ(1,3)+lambda2*ZZ(1,4)];

%% CONTINUOUS SECOND DIFFERENTIAL OF CONTROLLAW %%

WLD2=[ep*lambda1*WW(1,2)+(ep*lambda1^2+1)*WW(2,2)+lambda2*
      WW(3,2)+lambda2^2*WW(4,2) ep*lambda1*WW(1,3)+
      (ep*lambda1^2+1)*WW(2,3)+lambda2*WW(3,3)+lambda2^2*
      WW(4,3) ep*lambda1*WW(1,4)+(ep*lambda1^2+1)*WW(2,4)+
      lambda2*WW(3,4)+lambda2^2*WW(4,4)];
ZLUD2=[ep*lambda1*ZZ(1,2)+(ep*lambda1^2+1)*ZZ(2,2)+lambda2*
       ZZ(3,2)+lambda2^2*ZZ(4,2) ep*lambda1*ZZ(1,3)+
       (ep*lambda1^2+1)*ZZ(2,3)+lambda2*ZZ(3,3)+
       lambda2^2*ZZ(4,3) ep*lambda1*ZZ(1,4)+(ep*lambda1^2+1)*
       ZZ(2,4)+lambda2*ZZ(3,4)+lambda2^2*ZZ(4,4)];
ZLLD2=[ep*lambda1*ZZ(2,1)+(ep*lambda1^2+1)*ZZ(2,2)+lambda2*
       ZZ(2,3)+lambda2^2*ZZ(2,4) ep*lambda1*ZZ(3,1)+
       (ep*lambda1^2+1)*ZZ(3,2)+lambda2*ZZ(3,3)+lambda2^2*
       ZZ(3,4) ep*lambda1*ZZ(4,1)+(ep*lambda1^2+1)*ZZ(4,2)+
```

```
       lambda2*ZZ(4,3)+lambda2^2*ZZ(4,4)];
WRD2=[ep*lambda1*WW(1,1)+(ep*lambda1^2+1)*WW(2,1)+lambda2*
      WW(3,1)+lambda2^2*WW(4,1)];
ZRUD2=[ep*lambda1*ZZ(1,1)+(ep*lambda1^2+1)*ZZ(2,1)+lambda2*
       ZZ(3,1)+lambda2^2*ZZ(4,1)];
ZRLD2=[ep*lambda1*ZZ(1,1)+(ep*lambda1^2+1)*ZZ(1,2)+lambda2*
       ZZ(1,3)+lambda2^2*ZZ(1,4)];

%% DIFFERENTIAL APPROXIMATIONS FOR THE BOUNDARY
   CONDITIONS %%

yd10=(R(1,2)-R(1,1))/h;
yd11=(R(1,3)-R(1,2))/h;
yd12=(R(1,4)-R(1,3))/h;
yd1n=(R(1,n+1)-R(1,n))/h;
yd1n_1=(R(1,n)-R(1,n-1))/h;
yd1n_2=(R(1,n-1)-R(1,n-2))/h;
u0=(yd10-yd11)/h;
u1=(yd11-yd12)/h;
un=(yd1n_1-yd1n)/h;
un_1=(yd1n_2-yd1n_1)/h;
ud10=(u0-u1)/h;
ud1n=(un_1-un)/h;
X0=[yd10; u0; ud10]; %% 3/4 of the the first state vector %%
Xn=[yd1n; un; ud1n]; %% 3/4 of the the last state vector  %%

%% FORMING OF THE RIGHT HAND SIDE OF THE
   BLOCKDIAGONAL SYSTEM %%

j=1;
for i=2:n
Omega(j,1)=-ZRLD0*R(1,i-1)+WRD0*R(1,i)-ZRUD0*R(1,i+1);j=j+1;
Omega(j,1)=-ZRLD1*R(1,i-1)+WRD1*R(1,i)-ZRUD1*R(1,i+1);j=j+1;
Omega(j,1)=-ZRLD2*R(1,i-1)+WRD2*R(1,i)-ZRUD2*R(1,i+1);j=j+1;
end;
Omega(1,1)=Omega(1,1)-ZLLD0*X0;
Omega(2,1)=Omega(2,1)-ZLLD1*X0;
```

```
Omega(3,1)=Omega(3,1)-ZLLD2*X0;
Omega(3*n-5,1)=Omega(3*n-5,1)-ZLUD0*Xn;
Omega(3*n-4,1)=Omega(3*n-4,1)-ZLUD1*Xn;
Omega(3*n-3,1)=Omega(3*n-3,1)-ZLUD2*Xn;

%%  FORMING OF THE LEFT HAND SIDE OF THE
    BLOCKDIAGONAL SYSTEM  %%

for i=1:3
  DD(1,i)=-WLD0(1,i);
  DD(2,i)=-WLD1(1,i);
  DD(3,i)=-WLD2(1,i);
  if n>2
    DD(1,i+3)=ZLUD0(1,i);
    DD(2,i+3)=ZLUD1(1,i);
    DD(3,i+3)=ZLUD2(1,i);
  end;
end;
for i=2:n-2
  for j=-1:1
    DD(3*i-2,3*i-4+j)=ZLLD0(1,j+2);
    DD(3*i-2,3*i-1+j)=-WLD0(1,j+2);
    DD(3*i-2,3*i+2+j)=ZLUD0(1,j+2);
    DD(3*i-1,3*i-4+j)=ZLLD1(1,j+2);
    DD(3*i-1,3*i-1+j)=-WLD1(1,j+2);
    DD(3*i-1,3*i+2+j)=ZLUD1(1,j+2);
    DD(3*i  ,3*i-4+j)=ZLLD2(1,j+2);
    DD(3*i  ,3*i-1+j)=-WLD2(1,j+2);
    DD(3*i  ,3*i+2+j)=ZLUD2(1,j+2);
  end;
end;
if n>2
  for i=1:3
    DD(3*n-5,3*n-9+i)=ZLLD0(1,i);
    DD(3*n-4,3*n-9+i)=ZLLD1(1,i);
    DD(3*n-3,3*n-9+i)=ZLLD2(1,i);
    DD(3*n-5,3*n-6+i)=-WLD0(1,i);
```

```
    DD(3*n-4,3*n-6+i)=-WLD1(1,i);
    DD(3*n-3,3*n-6+i)=-WLD2(1,i);
  end;
end;

DD=sparse(DD);      %% SQUEEZING OUT ALL ZERO ELEMENTS
                       FROM MATRIX DD  %%

%% GAUSSELIMINATION TO PRODUCE AN UPPER TRIANGULAR
   SYSTEM  %%

for k=2:3:3*n-7
  for l=k:k+4
    if DD(k-1,k-1)~=0
      zd=DD(l,k-1)/DD(k-1,k-1);
      DD(l,:)=DD(l,:)-zd*DD(k-1,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k-1,1);
    end;
  end;
  for l=k+1:k+4
    if DD(k,k)~=0
      zd=DD(l,k)/DD(k,k);
      DD(l,:)=DD(l,:)-zd*DD(k,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k,1);
    end;
  end;
  for l=k+2:k+4
    if DD(k+1,k+1)~=0
      zd=DD(l,k+1)/DD(k+1,k+1);
      DD(l,:)=DD(l,:)-zd*DD(k+1,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k+1,1);
    end;
  end;
end;
k=3*n-5;
if DD(k,k)~=0
  zd=DD(k+1,k)/DD(k,k);
```

```
  DD(k+1,:)=DD(k+1,:)-zd*DD(k,:);
  Omega(k+1,1)=Omega(k+1,1)-zd*Omega(k,1);
  zd=DD(k+2,k)/DD(k,k);
  DD(k+2,:)=DD(k+2,:)-zd*DD(k,:);
  Omega(k+2,1)=Omega(k+2,1)-zd*Omega(k,1);
end;
if DD(k+1,k+1)~=0
  zd=DD(k+2,k+1)/DD(k+1,k+1);
  DD(k+2,:)=DD(k+2,:)-zd*DD(k+1,:);
  Omega(k+2,1)=Omega(k+2,1)-zd*Omega(k+1,1);
end;

%% BACKSUBSTITUTION TO SOLVE FOR THE STATEVECTORS %%

ud1(n-1)=Omega(3*(n-1),1)/DD(3*(n-1),3*(n-1));
u(n-1)=(Omega(3*n-4,1)-DD(3*n-4,3*(n-1))*ud1(n-1))/
DD(3*n-4,3*n-4);
yd1(n-1)=(Omega(3*n-5,1)-DD(3*n-5,3*(n-1))*ud1(n-1)-
DD(3*n-5,3*n-4)*u(n-1))/DD(3*n-5,3*n-5);
if n>2
  for k=n-2:-1:1
    ud1(k)=(Omega(3*k,1)-DD(3*k,3*k+3)*ud1(k+1)-
    DD(3*k,3*k+2)* u(k+1)-DD(3*k,3*k+1)*yd1(k+1))/
    DD(3*k,3*k);
    u(k)=(Omega(3*k-1,1)-DD(3*k-1,3*k+3)*ud1(k+1)-
    DD(3*k-1,3*k+2)*u(k+1)-DD(3*k-1,3*k+1)*yd1(k+1)-
    DD(3*k-1,3*k)*ud1(k))/DD(3*k-1,3*k-1);
    yd1(k)=(Omega(3*k-2,1)-DD(3*k-2,3*k+3)*ud1(k+1)-
    DD(3*k-2,3*k+2)*u(k+1)-DD(3*k-2,3*k+1)*yd1(k+1)
    -DD(3*k-2,3*k)*ud1(k)-DD(3*k-2,3*k-1)*u(k))/
   DD(3*k-2,3*k-2);
  end;
end;

%% MAKING OF THE STATEVECTORS %%

x(:,1)=[R(1,1); X0];
```

```
x(:,n+1)=[R(1,n+1); Xn];
for i=1:n-1
  x(:,i+1)=[R(1,i+1); yd1(i); u(i); ud1(i)];
end;

if cleargr
  clf;
  hold on;
  grid on;
end;

%%  PLOTTING OF THE CALC/SPEC TRAJECTORY,VELOCITY,
    CONTROLSIGNAL AND ACCELERATION  %%

plotadm=0;
while spc_plot
  if plotadm
    if spc_plot<10
      figure;
      hold on;
      grid on;
      title(['One dim. traj: mpr141knovel.m  l1 =
      ',num2str(lambda1),'  l2 = ',num2str(lambda2),' ep =
      ',num2str(ep),' n = ',num2str(n)])
      xlabel(['Total error : ',num2str(Total_error),
      '   Point error : ',num2str(Point_error),'   Graph :
      ',num2str(spc_plot),'   Time t'])
      if spc_plot<6
        for k=0:n
          plot(k*h,x(1,k+1),'o')
        end;
      end;
    end;
  end;
  breakadm=0;
  fadm=0;     %%  NORMALLY fadm=0, NECESSARY FOR WRITING
                  OF TEXTS IN THE PLOT  %%
```

```
for j=0:m
  eAtau=expm(A*j*h/m);
  for i=fadm:n-1   %%  CHOOSE e.g 2:n-3 TO AVOID
                       PROBLEMS AT THE ENDPOINTS  %%
    entry(:,i+1)=eAtau*(x(:,i+1)+Mtau(:,4*j+1:4*j+4)*
    Minv*(e_Ah*x(:,i+2)-x(:,i+1)));
    csignvec(:,i+1)=B'*expm(-A'*j*h/m)*Minv*(e_Ah*x(:,i+2)-
    x(:,i+1));
    if j==0
      if i==fadm;
        entry1=entry(1,fadm+1);
        entry2=entry(2,fadm+1);
        entry3=entry(3,fadm+1);
        entry4=entry(4,fadm+1);
        csignvec1=csignvec(1,fadm+1);
      end;
    end;
    if rem(j,mp)==0
      if j<=m-mp
        traject(1,j/mp*n+(i+1))=entry(1,i+1);
      end;
    end;
    if j==m
      if i==n-1
        traject(1,6*n+1)=entry(1,i+1);
      end;
    end;
    if spc_plot==1
      plot(i*h+j*h/m,entry(1,i+1),'.')    %% TRAJECTORY %%
      plot(i*h+j*h/m,entry(3,i+1),'.')    %% CONTROL u %%
      %% ACCELERATION %%
      plot(i*h+j*h/m,lambda1*entry(2,i+1)+
      entry(3,i+1)+ep*csignvec(1,i+1),'.')
    elseif spc_plot==2
      plot(i*h+j*h/m,entry(1,i+1),'.')    %% TRAJECTORY %%
      plot(i*h+j*h/m,entry(2,i+1),'.')    %% VELOCITY %%
      plot(i*h+j*h/m,entry(3,i+1),'.')    %% CONTROL u %%
```

```
  plot(i*h+j*h/m,lambda1*entry(2,i+1)+entry(3,i+1)+
  ep*csignvec(1,i+1),'.')   %% ACCELERATION %%
elseif spc_plot==3
  plot(i*h+j*h/m,entry(1,i+1),'.')    %% TRAJECTORY %%
  plot(i*h+j*h/m,lambda1*entry(2,i+1)+entry(3,i+1)+
  ep*csignvec(1,i+1),'.')   %% ACCELERATION %%
elseif spc_plot==4
  plot(i*h+j*h/m,entry(1,i+1),'.')    %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(2,i+1),'.')    %% VELOCITY %%
elseif spc_plot==5
  plot(i*h+j*h/m,entry(1,i+1),'.')    %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(3,i+1),'.')    %% CONTROL u %%
elseif spc_plot==6
  plot(i*h+j*h/m,entry(4,i+1),'.')    %% CONTROL DER
                                         u-dot %%
elseif spc_plot==7
  csignvec(:,i+1)=B'*expm(-A'*j*h/m)*Minv*
  (e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROLSIGNAL w %%
  plot(i*h+j*h/m,csignvec(1,i+1),'.')
  elseif spc_plot==8
  csd1vec(:,i+1)=B'*A'*expm(-A'*j*h/m)*Minv*
  (e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROL DER w-dot %%
  plot(i*h+j*h/m,csd1vec(1,i+1),'.')
  if j==0
    csdvec1=csd1vec(1,1);
  end;
elseif spc_plot==9
  csd2vec(:,i+1)=B'*A'*A'*expm(-A'*j*h/m)*
  Minv*(e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROL DERx2 w-dot-dot %%
  plot(i*h+j*h/m,csd2vec(1,i+1),'.')
  if j==0
    csdvec2=csd2vec(1,1);
  end;
else
```

```
        disp(' ')
        disp(' NOT A VALID CHOICE ')
        breakadm=1;
      end;
      if breakadm
        break;
      end;
    end;
    if breakadm
      break;
    end;
  end;
  if spc_plot<10
    ax=axis;
    ax2=ax(1,2);
    if ax2<1.5
      xpos=3/4*0.2;
    elseif ax2>=5
      xpos=3/4;
    else
      xpos=0.25;
    end;
    if spc_plot<3
      ax4=ax(1,4);
      if ax4<1
        ax4=ax4*10;
        if ax4<1
          ax4=ax4*10;
        end;
      end;
      if ax4>10
        ax4=ax4/10;
        if ax4>10
          ax4=ax4/10;
        end;
      end;
      if rem(ax4,2)==0
```

```
    yadd=(ax(1,4)/4)*1/5;
  else
    yadd=(ax(1,4)/3)*1/5;
  end;
  ypos=lambda1*entry2+entry3+ep*csignvec1;
  text(-xpos,ypos,['ACC']);
  if abs(ypos-entry3)>yadd
    text(-xpos,entry3,['CS u']);
  elseif ypos-entry3>0
    text(-xpos,entry3-yadd,['CS u']);
  else
    text(-xpos,entry3+yadd,['CS u']);
  end;
  if spc_plot==2
    text(-xpos,entry2,['VEL']);
  end;
end;
if spc_plot==3
  ypos=lambda1*entry2+entry3+ep*csignvec1;
  text(-xpos,ypos,['ACC']);
end;
if spc_plot==4
  text(-xpos,entry2,['VEL']);
end;
if spc_plot==5
  text(-xpos,entry3,['CS u']);
end;
if spc_plot==6
  text(-xpos,entry4,['ud1']);
end;
    if spc_plot==7
  text(-xpos,csignvec1,['CS w']);
end;
if spc_plot==8
  text(-xpos,csdvec1,['wd1']);
end;
if spc_plot==9
```

```
    text(-xpos,csdvec2,['wd2']);
  end;
end;

%%  ESTIMATION OF THE ACCURACY IN THE
    SPLINE APPROXIMATION  %%

if plotadm==0
  [Total_error,Average_error,Point_error,
  End_point_error]=spline_error(pointfcn,R,traject);
  Total_error
  Average_error
  Point_error
  End_point_error
  title(['One dim. traj: mpr141knovel.m  l1 =
  ',num2str(lambda1),'  l2 = ',num2str(lambda2),
  ' ep =',num2str(ep),' n = ',num2str(n)])
  xlabel(['Total error : ',num2str(Total_error),
  '   Point error : ',num2str(Point_error),'   Graph :
  ',num2str(spc_plot),'   Time t'])
  if spc_plot<6
    for k=0:n
      plot(k*h,x(1,k+1),'o')
    end;
  end;
end;

plotadm=plotadm+1;
disp(' ')
disp(' FOLLOWING OPTIONS ARE AVAILABLE : ')
disp(' ')
disp(' FINISH THE PROGRAM : 0 ')
disp(' DISPLAY THE TRAJECTORY, CONTROL u AND
ACCELERATION : 1 ')
disp(' DISPLAY THE TRAJECTORY, VELOCITY, CONTROL u AND
ACCELERATION : 2 ')
disp(' DISPLAY THE TRAJECTORY AND ACCELERATION : 3 ')
```

```
  disp(' DISPLAY THE TRAJECTORY AND VELOCITY : 4 ')
  disp(' DISPLAY THE TRAJECTORY AND CONTROL u : 5 ')
  disp(' DISPLAY THE FIRST DERIVATIVE OF THE CONTROL
  u : 6 ')
  disp(' DISPLAY THE CONTROLSIGNAL w : 7 ')
  disp(' DISPLAY THE FIRST DERIVATIVE OF THE CONTROL
  w : 8 ')
  disp(' DISPLAY THE SECOND DERIVATIVE OF THE CONTROL
  w : 9 ')
  spc_plot = input(' MAKE YOUR CHOICE : ');
end;

clear global;
for k=2:plotadm
  delete(k);
end;
end;
```

```
function [Q,cnt] = quad814mod(funfcn,a,b,tol)
%Alteration of the original matlab toolbox program.
%QUAD8 Numerical evaluation of an integral, higher order
%    method. Q = QUAD8('F',A,B,TOL) approximates the
%    integral of F(X) from to B to within a relative error
%    of TOL.  'F' is a string containing the name of the
%    function. The function must return a  4*4-matrix
%    output value if given an input value.
%    Q = Inf is returned if an excessive recursion level
%    is reached indicating a possibly singular integral.
%    QUAD8 uses an adaptive recursive Newton Cotes 8 panel
%    rule.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.
%    [Q,cnt] = quad8(F,a,b,tol) also returns a function
%    evaluation count.
%    Top level initialization, Newton-Cotes weights
w = [3956 23552 -3712 41984 -18160 41984 -3712 23552
```

```
     3956]/14175;

x = a + (0:8)*(b-a)/8;

% set up function call
for i=x
  y = [y feval(funfcn,i)];
end;

% Adaptive, recursive Newton-Cotes 8 panel quadrature
Q0 = zeros(4);
[Q,cnt] = quad814stpmod(funfcn,a,b,tol,0,w,x,y,Q0);
cnt = cnt + 9;
end;
```

```
function [Q,cnt] = quad814stpmod(FunFcn,a,b,tol,lev,
                                 w,x0,f0,Q0)
%Alteration of the original matlab toolbox program.
%QUAD8STP Recursive function used by QUAD8.
%    [Q,cnt] = quad8stp(F,a,b,tol,lev,w,f,Q0) tries to
%    approximate the integral of f(x) from a to b to
%    within a relative error of tol. F is a string
%    containing the name of f. The remaining arguments
%    are generated by quad8mod or by the recursion.
%    lev is the recursion level.
%    w is the weights in the 8 panel Newton Cotes formula.
%    x0 is a vector of 9 equally spaced abscissa is the
%    interval.
%    f0 is a matrix of the 9 function values at x.
%    Q0 is an approximate value of the integral.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.

LEVMAX = 10;

% Evaluate function at midpoints of left and
```

```
  right half intervals.
x = zeros(1,17);
x(1:2:17) = x0;
x(2:2:16) = (x0(1:8) + x0(2:9))/2;

f(:,1:4)= f0(:,1:4);
for i=1:8
  f(:,8*i-3:8*i) = feval(FunFcn,x(2*i));
  f(:,8*i+1:8*i+4) = (f0(:,4*i+1:4*i+4));
end;

% Integrate over half intervals.
h = (b-a)/16;
Q1=0;Q2=0;
for i=1:9
  Q1 = Q1 + h*w(i)*f(:,4*i-3:4*i);
  Q2 = Q2 + h*w(10-i)*f(:,69-i*4:72-i*4);
end;
Q = Q1 + Q2;
%%  Recursively refine approximations.
if norm(Q - Q0) > tol*norm(Q) & lev <= LEVMAX
   c = (a+b)/2;
   [Q1,cnt1] = quad814stpmod(FunFcn,a,c,tol/2,lev+1,
                             w,x(1:9),f(:,1:36),Q1);
   [Q2,cnt2] = quad814stpmod(FunFcn,c,b,tol/2,lev+1,
                             w,x(9:17),f(:,33:68),Q2);
   Q = Q1 + Q2;
   cnt = cnt + cnt1 + cnt2;
end
end;
```

```
function x = mpr151knovel(spc_plot,t,n,cleargr,pointfcn,
lambda1,lambda2,ep)

%%  THIS PROGRAM CALCULATES CONTROLLAWS FOR A ONE
    DIMENSIONAL TRAJECTORY  %%
%%  spc_plot DETERMINES WHICH GRAPH TO BE DISPLAYED,
    CHOSE AN INTEGER =<11  %%
%%  t IS THE TIME PERIOD FOR WHICH THE SYSTEM IS
    TO BE CONTROLLED  %%
%%  n ARE THE NUMBER OF POINTS AT THE SPECIFIED
    TRAJECTORY, CHOSE t/n>1/10  %%
%%  cleargr CLEARS THE CURRENT WINDOW, CHOSE 1 OR 0  %%
%%  pointfcn SPECIFIES THE TRAJECTORY  %%
%%  lambda1 AFFECTS THE INSTABILITY OF THE SYSTEM  %%
%%  USE lambda1<>0 TO AVOID NUMERICAL PROBLEMS WHEN
    DETERMINE THE MATRIX WW  %%
%%  lambda2 ALSO EFFECTS THE STABILITY OF THE SYSTEM,
    CHOSE l1~=l2  %%
%%  ep<>0 PUTS A ZERO IN THE TRANSFERFUNCTION  %%

global A B h t n

%%  THE SYSTEM  %%

A=[ 0 1 0 0 0;
    0 lambda1 1 0 0;
    0 0 0 1 0;
    0 0 0 0 1;
    0 0 0 0 lambda2];

B=[ 0 ep 0 0 1]';

C=[ 1 0 0 0 0];

%%  FUNCTION pointfcn DETERMINES THE SPECIFIED TRAJECTORY  %%

%%  NECESSARY FOR THE COMPOUND FUNCTION pointsin12  %%
```

```
if pointfcn=='pointsin12';
  t=5.2;
  n=26;
end;

R=feval(pointfcn);

%%  CALCULATION OF THE INTEGRAL FROM 0 TO h  %%

m=48;         %%  NUMBER OF POINTS BETWEEN INTERPOLATION,
                  CHOSE A MULTIPLE OF 6  %%
mp=m/6;       %%  NEEDED FOR fcn spline_error THAT DETERMINES
                  THE PRECISION IN THE SPLINE APPROXIMATION  %%
tol=1e-08;    %%  THE NUMERIC ERROR TOLERANCE  %%

Mtau(:,1:5)=zeros(5);
tau=0;
for j=1:m
  oldtau=tau;
  tau=oldtau+h/m;
  Mtau(:,5*j+1:5*j+5)= quad815mod('integrand',oldtau,tau,tol)
  + Mtau(:,5*j-4:5*j);
end;
M=Mtau(:,5*m+1:5*m+5);

%%  FORMING OF THE MATRICES FOR THE BLOCKDIAGONAL SYSTEM  %%

e_Ah=expm(-A*h);
Minv=inv(M);
ZZ=Minv*e_Ah;
WW=e_Ah'*ZZ+Minv;

%%  CONTINUOUS CONTROLLAW  %%

WLD0=[ep*WW(2,2)+WW(5,2) ep*WW(2,3)+WW(5,3)
      ep*WW(2,4)+WW(5,4) ep*WW(2,5)+WW(5,5)];
```

```
ZLUD0=[ep*ZZ(2,2)+ZZ(5,2) ep*ZZ(2,3)+ZZ(5,3)
       ep*ZZ(2,4)+ZZ(5,4) ep*ZZ(2,5)+ZZ(5,5)];
ZLLD0=[ep*ZZ(2,2)+ZZ(2,5) ep*ZZ(3,2)+ZZ(3,5)
       ep*ZZ(4,2)+ZZ(4,5) ep*ZZ(5,2)+ZZ(5,5)];
WRD0=[ep*WW(2,1)+WW(5,1)];
ZRUD0=[ep*ZZ(2,1)+ZZ(5,1)];
ZRLD0=[ep*ZZ(1,2)+ZZ(1,5)];

%% CONTINUOUS FIRST DIFFERENTIAL OF CONTROLLAW %%

WLD1=[ep*WW(1,2)+ep*lambda1*WW(2,2)+WW(4,2)+lambda2*WW(5,2)
      ep*WW(1,3)+ep*lambda1*WW(2,3)+WW(4,3)+lambda2*WW(5,3)
      ep*WW(1,4)+ep*lambda1*WW(2,4)+WW(4,4)+lambda2*WW(5,4)
      ep*WW(1,5)+ep*lambda1*WW(2,5)+WW(4,5)+lambda2*WW(5,5)];
ZLUD1=[ep*ZZ(1,2)+ep*lambda1*ZZ(2,2)+ZZ(4,2)+lambda2*ZZ(5,2)
       ep*ZZ(1,3)+ep*lambda1*ZZ(2,3)+ZZ(4,3)+lambda2*ZZ(5,3)
       ep*ZZ(1,4)+ep*lambda1*ZZ(2,4)+ZZ(4,4)+lambda2*ZZ(5,4)
       ep*ZZ(1,5)+ep*lambda1*ZZ(2,5)+ZZ(4,5)+lambda2*ZZ(5,5)];
ZLLD1=[ep*ZZ(2,1)+ep*lambda1*ZZ(2,2)+ZZ(2,4)+lambda2*ZZ(2,5)
       ep*ZZ(3,1)+ep*lambda1*ZZ(3,2)+ZZ(3,4)+lambda2*ZZ(3,5)
       ep*ZZ(4,1)+ep*lambda1*ZZ(4,2)+ZZ(4,4)+lambda2*ZZ(4,5)
       ep*ZZ(5,1)+ep*lambda1*ZZ(5,2)+ZZ(5,4)+lambda2*ZZ(5,5)];
WRD1=[ep*WW(1,1)+ep*lambda1*WW(2,1)+WW(4,1)+lambda2*WW(5,1)];
ZRUD1=[ep*ZZ(1,1)+ep*lambda1*ZZ(2,1)+ZZ(4,1)+lambda2*ZZ(5,1)];
ZRLD1=[ep*ZZ(1,1)+ep*lambda1*ZZ(1,2)+ZZ(1,4)+lambda2*ZZ(1,5)];

%% CONTINUOUS SECOND DIFFERENTIAL OF CONTROLLAW %%

WLD2=[ep*lambda1*WW(1,2)+ep*lambda1^2*WW(2,2)+WW(3,2)+
      lambda2*WW(4,2)+lambda2^2*WW(5,2) ep*lambda1*WW(1,3)+
      ep*lambda1^2*WW(2,3)+WW(3,3)+lambda2*WW(4,3)+
      lambda2^2*WW(5,3) ep*lambda1*WW(1,4)+ep*lambda1^2*
      WW(2,4)+WW(3,4)+lambda2*WW(4,4)+lambda2^2*WW(5,4)
      ep*lambda1*WW(1,5)+ep*lambda1^2*WW(2,5)+WW(3,5)+
      lambda2*WW(4,5)+lambda2^2*WW(5,5)];
ZLUD2=[ep*lambda1*ZZ(1,2)+ep*lambda1^2*ZZ(2,2)+ZZ(3,2)+
       lambda2*ZZ(4,2)+lambda2^2*ZZ(5,2) ep*lambda1*ZZ(1,3)+
```

```
        ep*lambda1^2*ZZ(2,3)+ZZ(3,3)+lambda2*ZZ(4,3)+
        lambda2^2*ZZ(5,3) ep*lambda1*ZZ(1,4)+ep*lambda1^2*
        ZZ(2,4)+ZZ(3,4)+lambda2*ZZ(4,4)+lambda2^2*ZZ(5,4)
        ep*lambda1*ZZ(1,5)+ep*lambda1^2*ZZ(2,5)+ZZ(3,5)+
        lambda2*ZZ(4,5)+lambda2^2*ZZ(5,5)];
ZLLD2=[ep*lambda1*ZZ(2,1)+ep*lambda1^2*ZZ(2,2)+ZZ(2,3)+
        lambda2*ZZ(2,4)+lambda2^2*ZZ(2,5) ep*lambda1*ZZ(3,1)+
        ep*lambda1^2*ZZ(3,2)+ZZ(3,3)+lambda2*ZZ(3,4)+
        lambda2^2*ZZ(3,5) ep*lambda1*ZZ(4,1)+ep*lambda1^2*
        ZZ(4,2)+ZZ(4,3)+lambda2*ZZ(4,4)+lambda2^2*ZZ(4,5)
        ep*lambda1*ZZ(5,1)+ep*lambda1^2*ZZ(5,2)+ZZ(5,3)+
        lambda2*ZZ(5,4)+lambda2^2*ZZ(5,5)];
WRD2=[ep*lambda1*WW(1,1)+ep*lambda1^2*WW(2,1)+WW(3,1)+
      lambda2*WW(4,1)+lambda2^2*WW(5,1)];
ZRUD2=[ep*lambda1*ZZ(1,1)+ep*lambda1^2*ZZ(2,1)+ZZ(3,1)+
        lambda2*ZZ(4,1)+lambda2^2*ZZ(5,1)];
ZRLD2=[ep*lambda1*ZZ(1,1)+ep*lambda1^2*ZZ(1,2)+ZZ(1,3)+
        lambda2*ZZ(1,4)+lambda2^2*ZZ(1,5)];

%% CONTINUOUS THIRD DIFFERENTIAL OF CONTROLLAW %%

WLD3=[ep*lambda1^2*WW(1,2)+(ep*lambda1^3+1)*WW(2,2)+lambda2*
      WW(3,2)+lambda2^2*WW(4,2)+lambda2^3*WW(5,2)
      ep*lambda1^2*WW(1,3)+(ep*lambda1^3+1)*WW(2,3)+lambda2*
      WW(3,3)+lambda2^2*WW(4,3)+lambda2^3*WW(5,3)
      ep*lambda1^2*WW(1,4)+(ep*lambda1^3+1)*WW(2,4)+lambda2*
      WW(3,4)+lambda2^2*WW(4,4)+lambda2^3*WW(5,4)
      ep*lambda1^2*WW(1,5)+(ep*lambda1^3+1)*WW(2,5)+lambda2*
      WW(3,5)+lambda2^2*WW(4,5)+lambda2^3*WW(5,5)];
ZLUD3=[ep*lambda1^2*ZZ(1,2)+(ep*lambda1^3+1)*ZZ(2,2)+lambda2*
       ZZ(3,2)+lambda2^2*ZZ(4,2)+lambda2^3*ZZ(5,2)
       ep*lambda1^2*ZZ(1,3)+(ep*lambda1^3+1)*ZZ(2,3)+lambda2*
       ZZ(3,3)+lambda2^2*ZZ(4,3)+lambda2^3*ZZ(5,3)
       ep*lambda1^2*ZZ(1,4)+(ep*lambda1^3+1)*ZZ(2,4)+lambda2*
       ZZ(3,4)+lambda2^2*ZZ(4,4)+lambda2^3*ZZ(5,4)
       ep*lambda1^2*ZZ(1,5)+(ep*lambda1^3+1)*ZZ(2,5)+lambda2*
       ZZ(3,5)+lambda2^2*ZZ(4,5)+lambda2^3*ZZ(5,5)];
```

```
ZLLD3=[ep*lambda1^2*ZZ(2,1)+(ep*lambda1^3+1)*ZZ(2,2)+lambda2*
       ZZ(2,3)+lambda2^2*ZZ(2,4)+lambda2^3*ZZ(2,5)
       ep*lambda1^2*ZZ(3,1)+(ep*lambda1^3+1)*ZZ(3,2)+lambda2*
       ZZ(3,3)+lambda2^2*ZZ(3,4)+lambda2^3*ZZ(3,5)
       ep*lambda1^2*ZZ(4,1)+(ep*lambda1^3+1)*ZZ(4,2)+lambda2*
       ZZ(4,3)+lambda2^2*ZZ(4,4)+lambda2^3*ZZ(4,5)
       ep*lambda1^2*ZZ(5,1)+(ep*lambda1^3+1)*ZZ(5,2)+lambda2*
       ZZ(5,3)+lambda2^2*ZZ(5,4)+lambda2^3*ZZ(5,5)];
WRD3=[ep*lambda1^2*WW(1,1)+(ep*lambda1^3+1)*WW(2,1)+lambda2*
      WW(3,1)+lambda2^2*WW(4,1)+lambda2^3*WW(5,1)];
ZRUD3=[ep*lambda1^2*ZZ(1,1)+(ep*lambda1^3+1)*ZZ(2,1)+lambda2*
       ZZ(3,1)+lambda2^2*ZZ(4,1)+lambda2^3*ZZ(5,1)];
ZRLD3=[ep*lambda1^2*ZZ(1,1)+(ep*lambda1^3+1)*ZZ(1,2)+lambda2*
       ZZ(1,3)+lambda2^2*ZZ(1,4)+lambda2^3*ZZ(1,5)];

%% DIFFERENTIAL APPROXIMATIONS FOR THE BOUNDARY CONDITIONS %%

yd10=(R(1,2)-R(1,1))/h;
yd11=(R(1,3)-R(1,2))/h;
yd12=(R(1,4)-R(1,3))/h;
yd13=(R(1,5)-R(1,4))/h;
yd1n=(R(1,n+1)-R(1,n))/h;
yd1n_1=(R(1,n)-R(1,n-1))/h;
yd1n_2=(R(1,n-1)-R(1,n-2))/h;
yd1n_3=(R(1,n-2)-R(1,n-3))/h;
u0=(yd10-yd11)/h;
u1=(yd11-yd12)/h;
u2=(yd12-yd13)/h;
un=(yd1n_1-yd1n)/h;
un_1=(yd1n_2-yd1n_1)/h;
un_2=(yd1n_3-yd1n_2)/h;
ud10=(u0-u1)/h;
ud11=(u1-u2)/h;
ud1n=(un_1-un)/h;
ud1n_1=(un_2-un_1)/h;
ud20=(ud10-ud11)/h;
ud2n=(ud1n_1-ud1n)/h;
```

```
%% 4/5 of the the first state vector %%
X0=[yd10; u0; ud10; ud20];
%% 4/5 of the the last state vector %%
Xn=[yd1n; un; ud1n; ud2n];

%% FORMING OF THE RIGHT HAND SIDE OF THE
   BLOCKDIAGONAL SYSTEM %%

j=1;
for i=2:n
  Omega(j,1)=-ZRLD0*R(1,i-1)+WRD0*R(1,i)-ZRUD0*R(1,i+1);
  j=j+1;
  Omega(j,1)=-ZRLD1*R(1,i-1)+WRD1*R(1,i)-ZRUD1*R(1,i+1);
  j=j+1;
  Omega(j,1)=-ZRLD2*R(1,i-1)+WRD2*R(1,i)-ZRUD2*R(1,i+1);
  j=j+1;
  Omega(j,1)=-ZRLD3*R(1,i-1)+WRD3*R(1,i)-ZRUD3*R(1,i+1);
  j=j+1;
end;
Omega(1,1)=Omega(1,1)-ZLLD0*X0;
Omega(2,1)=Omega(2,1)-ZLLD1*X0;
Omega(3,1)=Omega(3,1)-ZLLD2*X0;
Omega(4,1)=Omega(4,1)-ZLLD3*X0;
Omega(4*n-7,1)=Omega(4*n-7,1)-ZLUD0*Xn;
Omega(4*n-6,1)=Omega(4*n-6,1)-ZLUD1*Xn;
Omega(4*n-5,1)=Omega(4*n-5,1)-ZLUD2*Xn;
Omega(4*n-4,1)=Omega(4*n-4,1)-ZLUD3*Xn;

%% FORMING OF THE LEFT HAND SIDE OF THE
   BLOCKDIAGONAL SYSTEM %%

for i=1:4
  DD(1,i)=-WLD0(1,i);
  DD(2,i)=-WLD1(1,i);
  DD(3,i)=-WLD2(1,i);
  DD(4,i)=-WLD3(1,i);
  if n>2
```

```
    DD(1,i+4)=ZLUD0(1,i);
    DD(2,i+4)=ZLUD1(1,i);
    DD(3,i+4)=ZLUD2(1,i);
    DD(4,i+4)=ZLUD3(1,i);
  end;
end;
for i=2:n-2
  for j=-1:2
    DD(4*i-3,4*i-6+j)=ZLLD0(1,j+2);
    DD(4*i-3,4*i-2+j)=-WLD0(1,j+2);
    DD(4*i-3,4*i+2+j)=ZLUD0(1,j+2);
    DD(4*i-2,4*i-6+j)=ZLLD1(1,j+2);
    DD(4*i-2,4*i-2+j)=-WLD1(1,j+2);
    DD(4*i-2,4*i+2+j)=ZLUD1(1,j+2);
    DD(4*i-1,4*i-6+j)=ZLLD2(1,j+2);
    DD(4*i-1,4*i-2+j)=-WLD2(1,j+2);
    DD(4*i-1,4*i+2+j)=ZLUD2(1,j+2);
    DD(4*i,  4*i-6+j)=ZLLD3(1,j+2);
    DD(4*i,  4*i-2+j)=-WLD3(1,j+2);
    DD(4*i,  4*i+2+j)=ZLUD3(1,j+2);
  end;
end;
if n>2
  for i=1:4
    DD(4*n-7,4*n-12+i)=ZLLD0(1,i);
    DD(4*n-6,4*n-12+i)=ZLLD1(1,i);
    DD(4*n-5,4*n-12+i)=ZLLD2(1,i);
    DD(4*n-4,4*n-12+i)=ZLLD3(1,i);
    DD(4*n-7,4*n-8+i)=-WLD0(1,i);
    DD(4*n-6,4*n-8+i)=-WLD1(1,i);
    DD(4*n-5,4*n-8+i)=-WLD2(1,i);
    DD(4*n-4,4*n-8+i)=-WLD3(1,i);
  end;
end;

DD=sparse(DD);      %%  SQUEEZING OUT ALL ZERO ELEMENTS
                        FROM MATRIX DD  %%
```

```
%% GAUSSELIMINATION TO PRODUCE AN UPPER TRIANGULAR SYSTEM %%

for k=2:4:4*n-10
  for l=k:k+6
    if DD(k-1,k-1)~=0
      zd=DD(l,k-1)/DD(k-1,k-1);
      DD(l,:)=DD(l,:)-zd*DD(k-1,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k-1,1);
    end;
  end;
  for l=k+1:k+6
    if DD(k,k)~=0
      zd=DD(l,k)/DD(k,k);
      DD(l,:)=DD(l,:)-zd*DD(k,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k,1);
    end;
  end;
  for l=k+2:k+6
    if DD(k+1,k+1)~=0
      zd=DD(l,k+1)/DD(k+1,k+1);
      DD(l,:)=DD(l,:)-zd*DD(k+1,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k+1,1);
    end;
  end;
  for l=k+3:k+6
    if DD(k+2,k+2)~=0
      zd=DD(l,k+2)/DD(k+2,k+2);
      DD(l,:)=DD(l,:)-zd*DD(k+2,:);
      Omega(l,1)=Omega(l,1)-zd*Omega(k+2,1);
    end;
  end;
end;
k=4*n-7;
if DD(k,k)~=0
  zd=DD(k+1,k)/DD(k,k);
  DD(k+1,:)=DD(k+1,:)-zd*DD(k,:);
```

```
  Omega(k+1,1)=Omega(k+1,1)-zd*Omega(k,1);
  zd=DD(k+2,k)/DD(k,k);
  DD(k+2,:)=DD(k+2,:)-zd*DD(k,:);
  Omega(k+2,1)=Omega(k+2,1)-zd*Omega(k,1);
  zd=DD(k+3,k)/DD(k,k);
  DD(k+3,:)=DD(k+3,:)-zd*DD(k,:);
  Omega(k+3,1)=Omega(k+3,1)-zd*Omega(k,1);
end;
if DD(k+1,k+1)~=0
  zd=DD(k+2,k+1)/DD(k+1,k+1);
  DD(k+2,:)=DD(k+2,:)-zd*DD(k+1,:);
  Omega(k+2,1)=Omega(k+2,1)-zd*Omega(k+1,1);
  zd=DD(k+3,k+1)/DD(k+1,k+1);
  DD(k+3,:)=DD(k+3,:)-zd*DD(k+1,:);
  Omega(k+3,1)=Omega(k+3,1)-zd*Omega(k+1,1);
end;
if DD(k+2,k+2)~=0
  zd=DD(k+3,k+2)/DD(k+2,k+2);
  DD(k+3,:)=DD(k+3,:)-zd*DD(k+2,:);
  Omega(k+3,1)=Omega(k+3,1)-zd*Omega(k+2,1);
end;

%%  BACKSUBSTITUTION TO SOLVE FOR THE STATEVECTORS  %%

ud2(n-1)=Omega(4*n-4,1)/DD(4*n-4,4*n-4);
ud1(n-1)=(Omega(4*n-5,1)-DD(4*n-5,4*n-4)*ud2(n-1))/
DD(4*n-5,4*n-5);
u(n-1)=(Omega(4*n-6,1)-DD(4*n-6,4*n-4)*ud2(n-1)-
DD(4*n-6,4*n-5)*ud1(n-1))/DD(4*n-6,4*n-6);
yd1(n-1)=(Omega(4*n-7,1)-DD(4*n-7,4*n-4)*ud2(n-1)-
DD(4*n-7,4*n-5)*ud1(n-1)-DD(4*n-7,4*n-6)*u(n-1))/
DD(4*n-7,4*n-7);
if n>2
  for k=n-2:-1:1
    ud2(k)=(Omega(4*k,1)-DD(4*k,4*k+4)*ud2(k+1)-
    DD(4*k,4*k+3)*ud1(k+1)-DD(4*k,4*k+2)*u(k+1)-
    DD(4*k,4*k+1)*yd1(k+1))/DD(4*k,4*k);
```

```
    ud1(k)=(Omega(4*k-1,1)-DD(4*k-1,4*k+4)*ud2(k+1)-
    DD(4*k-1,4*k+3)*ud1(k+1)-DD(4*k-1,4*k+2)*u(k+1)-
    DD(4*k-1,4*k+1)*yd1(k+1)-DD(4*k-1,4*k)*ud2(k))/
    DD(4*k-1,4*k-1);
    u(k)=(Omega(4*k-2,1)-DD(4*k-2,4*k+4)*ud2(k+1)-
    DD(4*k-2,4*k+3)*ud1(k+1)-DD(4*k-2,4*k+2)*u(k+1)-
    DD(4*k-2,4*k+1)*yd1(k+1)-DD(4*k-2,4*k)*ud2(k)-
    DD(4*k-2,4*k-1)*ud1(k))/DD(4*k-2,4*k-2);
    yd1(k)=(Omega(4*k-3,1)-DD(4*k-3,4*k+4)*ud2(k+1)-
    DD(4*k-3,4*k+3)*ud1(k+1)-DD(4*k-3,4*k+2)*u(k+1)-
    DD(4*k-3,4*k+1)*yd1(k+1)-DD(4*k-3,4*k)*ud2(k)-
    DD(4*k-3,4*k-1)*ud1(k)-DD(4*k-3,4*k-2)*u(k))/
    DD(4*k-3,4*k-3);
  end;
end;

%% MAKING OF THE STATEVECTORS %%

x(:,1)=[R(1,1); X0];
x(:,n+1)=[R(1,n+1); Xn];
for i=1:n-1
  x(:,i+1)=[R(1,i+1); yd1(i); u(i); ud1(i); ud2(i)];
end;

if cleargr
  clf;
  hold on;
  grid on;
end;

%% PLOTTING OF THE CALC/SPEC TRAJECTORY,VELOCITY,
    CONTROLSIGNAL AND ACCELERATION %%

plotadm=0;
while spc_plot
  if plotadm
    if spc_plot<12
```

```
    figure;
    hold on;
    grid on;
    title(['One dim. traj: mpr151knovel.m  l1 =
    ',num2str(lambda1),'  l2 = ',num2str(lambda2),' ep =
    ',num2str(ep),' n = ',num2str(n)])
    xlabel(['Total error : ',num2str(Total_error),
    '   Point error : ',num2str(Point_error),'   Graph :
    ',num2str(spc_plot),'   Time t'])
    if spc_plot<6
      for k=0:n
        plot(k*h,x(1,k+1),'o')
      end;
    end;
  end;
end;
breakadm=0;
fadm=0;    %%  NORMALLY fadm=0, NECESSARY FOR WRITING OF
               TEXTS IN THE PLOT  %%
for j=0:m
  eAtau=expm(A*j*h/m);
  for i=fadm:n-1   %%  CHOOSE e.g 2:n-3 TO AVOID
                       PROBLEMS AT THE ENDPOINTS  %%
    entry(:,i+1)=eAtau*(x(:,i+1)+Mtau(:,5*j+1:5*j+5)*
    Minv*(e_Ah*x(:,i+2)-x(:,i+1)));
    csignvec(:,i+1)=B'*expm(-A'*j*h/m)*Minv*
    (e_Ah*x(:,i+2)-x(:,i+1));
    if j==0
      if i==fadm;
        entry1=entry(1,fadm+1);
        entry2=entry(2,fadm+1);
        entry3=entry(3,fadm+1);
        entry4=entry(4,fadm+1);
        entry5=entry(5,fadm+1);
        csignvec1=csignvec(1,1);
      end;0
    end;
```

```
if rem(j,mp)==0
  if j<=m-mp
    traject(1,j/mp*n+(i+1))=entry(1,i+1);
  end;
end;
if j==m
  if i==n-1
    traject(1,6*n+1)=entry(1,i+1);
  end;
end;
if spc_plot==1
  plot(i*h+j*h/m,entry(1,i+1),'.')   %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(3,i+1),'.')   %% CONTROL u %%
  plot(i*h+j*h/m,lambda1*entry(2,i+1)+entry(3,i+1)+
  ep*csignvec(1,i+1),'.')   %% ACCELERATION %%
elseif spc_plot==2
  plot(i*h+j*h/m,entry(1,i+1),'.')   %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(2,i+1),'.')   %% VELOCITY %%
  plot(i*h+j*h/m,entry(3,i+1),'.')   %% CONTROL u %%
  plot(i*h+j*h/m,lambda1*entry(2,i+1)+entry(3,i+1)+
  ep*csignvec(1,i+1),'.')   %% ACCELERATION %%
elseif spc_plot==3
  plot(i*h+j*h/m,entry(1,i+1),'.')   %% TRAJECTORY %%
  plot(i*h+j*h/m,lambda1*entry(2,i+1)+entry(3,i+1)+
  ep*csignvec(1,i+1),'.')   %% ACCELERATION %%
elseif spc_plot==4
  plot(i*h+j*h/m,entry(1,i+1),'.')   %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(2,i+1),'.')   %% VELOCITY %%
elseif spc_plot==5
  plot(i*h+j*h/m,entry(1,i+1),'.')   %% TRAJECTORY %%
  plot(i*h+j*h/m,entry(3,i+1),'.')   %% CONTROL u %%
elseif spc_plot==6
  %% CONTROL DER u-dot %%
  plot(i*h+j*h/m,entry(4,i+1),'.')
elseif spc_plot==7
  %% CONTROL DERx2 u-dot-dot %%
  plot(i*h+j*h/m,entry(5,i+1),'.')
```

```
elseif spc_plot==8
  csignvec(:,i+1)=B'*expm(-A'*j*h/m)*Minv*
  (e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROLSIGNAL w %%
  plot(i*h+j*h/m,csignvec(1,i+1),'.')
elseif spc_plot==9
  csd1vec(:,i+1)=B'*A'*expm(-A'*j*h/m)*Minv*
  (e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROL DER w-dot %%
  plot(i*h+j*h/m,csd1vec(1,i+1),'.')
  if j==0
    csdvec1=csd1vec(1,1);
  end;
elseif spc_plot==10
  csd2vec(:,i+1)=B'*A'*A'*expm(-A'*j*h/m)*Minv*
  (e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROL DERx2 w-dot-dot %%
  plot(i*h+j*h/m,csd2vec(1,i+1),'.')
  if j==0
    csdvec2=csd2vec(1,1);
  end;
elseif spc_plot==11
  csd3vec(:,i+1)=B'*A'*A'*A'*expm(-A'*j*h/m)*
  Minv*(e_Ah*x(:,i+2)-x(:,i+1));
  %% CONTROL DERx3 w-dot-dot-dot %%
  plot(i*h+j*h/m,csd3vec(1,i+1),'.')
  if j==0
    csdvec3=csd3vec(1,1);
  end;
else
  disp(' ')
  disp(' NOT A VALID CHOICE ')
  breakadm=1;
end;
if breakadm
  break;
end;
```

```
    end;
    if breakadm
      break;
    end;
  end;
  if spc_plot<12
    ax=axis;
    ax2=ax(1,2);
    if ax2<1.5
      xpos=3/4*0.2;
    elseif ax2>=5
      xpos=3/4;
    else
      xpos=0.25;
    end;
    if spc_plot<3
      ax4=ax(1,4);
      if ax4<1
        ax4=ax4*10;
        if ax4<1
          ax4=ax4*10;
        end;
      end;
      if ax4>10
        ax4=ax4/10;
        if ax4>10
          ax4=ax4/10;
        end;
      end;
      if rem(ax4,2)==0
        yadd=(ax(1,4)/4)*1/5;
      else
        yadd=(ax(1,4)/3)*1/5;
      end;
      ypos=lambda1*entry2+entry3+ep*csignvec1;
      text(-xpos,ypos,['ACC']);
      if abs(ypos-entry3)>yadd
```

```
    text(-xpos,entry3,['CS u']);
  elseif ypos-entry3>0
    text(-xpos,entry3-yadd,['CS u']);
  else
    text(-xpos,entry3+yadd,['CS u']);
  end;
  if spc_plot==2
    text(-xpos,entry2,['VEL']);
  end;
end;
if spc_plot==3
  ypos=lambda1*entry2+entry3+ep*csignvec1;
  text(-xpos,ypos,['ACC']);
end;
if spc_plot==4
  text(-xpos,entry2,['VEL']);
end;
if spc_plot==5
  text(-xpos,entry3,['CS u']);
end;
if spc_plot==6
  text(-xpos,entry4,['ud1']);
end;
if spc_plot==7
  text(-xpos,entry5,['ud2']);
end;
if spc_plot==8
  text(-xpos,csignvec1,['CS w']);
end;
if spc_plot==9
  text(-xpos,csdvec1,['wd1']);
end;
if spc_plot==10
  text(-xpos,csdvec2,['wd2']);
end;
if spc_plot==11
  text(-xpos,csdvec3,['wd3']);
```

```
  end;
end;

%%  ESTIMATION OF THE ACCURACY IN THE
    SPLINE APPROXIMATION  %%

if plotadm==0
  [Total_error,Average_error,Point_error,End_point_error]=
  spline_error(pointfcn,R,traject);
  Total_error
  Average_error
  Point_error
  End_point_error
  title(['One dim. traj: mpr151knovel.m  l1 =
  ',num2str(lambda1),'  l2 = ',num2str(lambda2),' ep =
  ',num2str(ep),' n = ',num2str(n)])
  xlabel(['Total error : ',num2str(Total_error),
  '   Point error : ',num2str(Point_error),'   Graph :
  ',num2str(spc_plot),'   Time t'])
  if spc_plot<6
    for k=0:n
      plot(k*h,x(1,k+1),'o')
    end;
  end;
end;

plotadm=plotadm+1;
disp(' ')
disp(' FOLLOWING OPTIONS ARE AVAILABLE : ')
disp(' ')
disp(' FINISH THE PROGRAM : 0 ')
disp(' DISPLAY THE TRAJECTORY, CONTROL u AND
ACCELERATION : 1 ')
disp(' DISPLAY THE TRAJECTORY, VELOCITY, CONTROL u AND
ACCELERATION : 2 ')
disp(' DISPLAY THE TRAJECTORY AND ACCELERATION : 3 ')
disp(' DISPLAY THE TRAJECTORY AND VELOCITY : 4 ')
```

```
  disp(' DISPLAY THE TRAJECTORY AND CONTROL u : 5 ')
  disp(' DISPLAY THE FIRST DERIVATIVE OF THE CONTROL u : 6 ')
  disp(' DISPLAY THE SECOND DERIVATIVE OF THE
  CONTROL u : 7 ')
  disp(' DISPLAY THE CONTROLSIGNAL w : 8 ')
  disp(' DISPLAY THE FIRST DERIVATIVE OF THE CONTROL w : 9 ')
  disp(' DISPLAY THE SECOND DERIVATIVE OF THE
  CONTROL w : 10 ')
  disp(' DISPLAY THE THIRD DERIVATIVE OF THE
  CONTROL w : 11 ')
  spc_plot = input(' MAKE YOUR CHOICE : ');
end;

clear global;
for k=2:plotadm
  delete(k);
end;
end;
```

```
function [Q,cnt] = quad815mod(funfcn,a,b,tol)
%Alteration of the original matlab toolbox program.
%QUAD8 Numerical evaluation of an integral, higher order
%    method. Q = QUAD8('F',A,B,TOL) approximates the
%    integral of F(X) from to B to within a relative error
%    of TOL.  'F' is a string containing the name of the
%    function. The function must return a  5*5-matrix
%    output value if given an input value.
%    Q = Inf is returned if an excessive recursion level
%    is reached indicating a possibly singular integral.
%    QUAD8 uses an adaptive recursive Newton Cotes 8 panel
%    rule.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.
%    [Q,cnt] = quad8(F,a,b,tol) also returns a function
%    evaluation count.
%    Top level initialization, Newton-Cotes weights
```

```
w = [3956 23552 -3712 41984 -18160 41984 -3712 23552
     3956]/14175;

x = a + (0:8)*(b-a)/8;

% set up function call
for i=x
  y = [y feval(funfcn,i)];
end;

% Adaptive, recursive Newton-Cotes 8 panel quadrature
Q0 = zeros(5);
[Q,cnt] = quad815stpmod(funfcn,a,b,tol,0,w,x,y,Q0);
cnt = cnt + 9;
end;
```

```
function [Q,cnt] = quad815stpmod(FunFcn,a,b,tol,lev,
                                 w,x0,f0,Q0)
%Alteration of the original matlab toolbox program.
%QUAD8STP Recursive function used by QUAD8.
%    [Q,cnt] = quad8stp(F,a,b,tol,lev,w,f,Q0) tries to
%    approximate the integral of f(x) from a to b to
%    within a relative error of tol. F is a string
%    containing the name of f. The remaining arguments
%    are generated by quad8mod or by the recursion.
%    lev is the recursion level.
%    w is the weights in the 8 panel Newton Cotes formula.
%    x0 is a vector of 9 equally spaced abscissa is the
%    interval.
%    f0 is a matrix of the 9 function values at x.
%    Q0 is an approximate value of the integral.
%    Cleve Moler, 5-08-88.
%    Copyright (c) 1984-94 by The MathWorks, Inc.
LEVMAX = 10;

% Evaluate function at midpoints of left and
```

```
  right half intervals.
x = zeros(1,17);
x(1:2:17) = x0;
x(2:2:16) = (x0(1:8) + x0(2:9))/2;

f(:,1:5)= f0(:,1:5);
for i=1:8
  f(:,10*i-4:10*i) = feval(FunFcn,x(2*i));
  f(:,10*i+1:10*i+5) = (f0(:,5*i+1:5*i+5));
end;

% Integrate over half intervals.
h = (b-a)/16;
Q1=0;Q2=0;
for i=1:9
  Q1 = Q1 + h*w(i)*f(:,5*i-4:5*i);
  Q2 = Q2 + h*w(10-i)*f(:,86-i*5:90-i*5);
end;
Q = Q1 + Q2;
%%  Recursively refine approximations.
if norm(Q - Q0) > tol*norm(Q) & lev <= LEVMAX
   c = (a+b)/2;
   [Q1,cnt1] = quad815stpmod(FunFcn,a,c,tol/2,lev+1,
                             w,x(1:9),f(:,1:45),Q1);
   [Q2,cnt2] = quad815stpmod(FunFcn,c,b,tol/2,lev+1,
                             w,x(9:17),f(:,41:85),Q2);
   Q = Q1 + Q2;
   cnt = cnt + cnt1 + cnt2;
end
end;
```

```
function res = integrand(v)

%%  THIS SUBPROGRAM DETERMINE THE INTEGRAND OF THE MATRIX
    CONSTANT M  %%

global A B

e_AvB=expm(-A*v)*B;
res = e_AvB*e_AvB';
end;
```

```
function [Total_error,Average_error,Point_error,
End_point_error]=spline_error(pointfcn,R,traject)

global h t n

Total_error=0;
Point_error=0;
n=6*n;
Rp=feval(pointfcn);
n=n/6;
h=t/n;
for i=0:n-1
  for j=0:5
    Total_error=Total_error+abs(traject(1,j*n+1+i)-
    Rp(1,i*6+j+1));
    if j==0
      Point_error=Point_error+abs(traject(1,i+1)-R(1,i+1));
    end;
  end;
end;
Average_error=Total_error/(6*n);
End_point_error=abs(traject(1,6*n+1)-R(1,n+1));
end;
```

```
function R=pointexp11

% R = 1*(n+1)-matrix.

global h t n

%% TIME INTERVAL FOR RENEWAL OF THE TRAJECTORY %%
h=t/n;

for j=0:h:n*h
  R(1,j/h+1)=(1-exp(-3/2*(j-t/2)))/(1+exp(-3/2*(j-t/2)));
end;
end;
```

```
function R=pointsin12

% R = 1*(n+1)-matrix.

global h t n

%% TIME INTERVAL FOR RENEWAL OF THE TRAJECTORY %%
h=t/n;
%% I APOLOGIZE FOR THE "SMART" PROGRAMING %%
adm=n*3/26;
for j=1:adm
  R(1,j)=0;
end;
for j=0:h:(n-2*adm)*h
  R(1,j/h+adm+1)=1/10*(1+sin(-pi/2+j*pi/2));
end;
for j=1:adm
  R(1,j-adm+n+1)=0;
end;
end;
```

```
function R=pointstep1

% R = 1*(n+1)-matrix.

global h t n

h=t/n;
for j=0:h:n*h
  R(1,j/h+1)=1/2*(1+sign(j-(t/2+0.01)));
end;
end;
```

References

[Enquist] Enquist P, Control Theory and Splines, applied to Signature Storage, *Texas Tech University, Lubbock, USA; Royal Institute of Technology, Stockholm, Sweden* (1994).

[Etkin] Etkin B, Dynamics of Atmospheric Flight, *John Wiley & Sons, Inc* (1972).

[Gerald] Gerald C F, Applied Numerical Analysis 2/ed, *Addison-Wesley Publishing Company* (1977).

[Glad/Ljung] Glad T, Ljung L, Reglerteknik-Grundläggande Teori, *Studentlitteratur* (1989).

[Hildebrand] Hildebrand F B, Introduction to Numerical Analysis, *Dover Publications, Inc* (1987).

[Kahaner/Moler/Nash] Kahaner D; Moler C; Nash S, Numerical Methods and Software, *Prentice-Hall International, Inc* (1989).

[Lindquist/Sand] Lindquist A, Sand J, An Introduction to Mathematical Systems Theory, *Royal Institute of Technology, Stockholm, Sweden* (1993).

[Scheid] Scheid F, Numerical Analysis 2/ed, *Mc Graw-Hill Book Company* (1988).

[Strang] Strang G, Linear Algebra and its Applications 3/ed, *Harcourt Brace Jovanovich, Publishers* (1988).

[Taylor] Taylor A E, Introduction to Functional Analysis, *John Wiley & Sons, Inc* (1958).