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

List of Figures

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

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

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

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

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

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

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

$$Y(s) = \frac{\epsilon s^2 - \epsilon\lambda 2 s + 1}{s^2(s-\lambda 1)(s-\lambda 2)} U(s)$$

gives for negative λ's that there are no poles in the open right half plane. The numerator $\epsilon s^2 - \epsilon\lambda 2 s + 1 = 0$ 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

$$U^B \triangleq B^T \begin{bmatrix} m_{11}^{-1} & m_{12}^{-1} \\ m_{21}^{-1} & m_{22}^{-1} \end{bmatrix}$$

$$U_{lu}^B = \begin{bmatrix} m_{22}^{-1} \end{bmatrix} \qquad U_{ru}^B = \begin{bmatrix} m_{21}^{-1} \end{bmatrix}$$

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

$$\lambda\dot{y}_n + u_{n-1}(t_n) = 0 \text{ where } u_{n-1}(t_n) = B^T e^{-A^T h}M^{-1}\left(e^{-Ah}x_n - x_{n-1}\right) =$$

$$B^T e^{-A^T h}Zx_n - B^T e^{-A^T h}M^{-1}x_{n-1} = B^T Wx_n - B^T M^{-1}x_n - B^T Z^T x_{n-1}$$

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

$$B^T A^T A^T \left(Z^T x_k - W x_{k+1} + Z x_{k+2} \right) = 0 \quad k = 0, 1, \ldots, n-2.$$

By partitioning the matrices in definition 6.2 as

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} \\ z_{21} & z_{22} & z_{23} & z_{24} \\ z_{31} & z_{32} & z_{33} & z_{34} \\ z_{41} & z_{42} & z_{43} & z_{44} \end{bmatrix} \qquad Z^T = \begin{bmatrix} z_{11} & z_{21} & z_{31} & z_{41} \\ z_{12} & z_{22} & z_{32} & z_{42} \\ z_{13} & z_{23} & z_{33} & z_{43} \\ z_{14} & z_{24} & z_{34} & z_{44} \end{bmatrix}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix}$$

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, $\ddot{u}$:

$$\ddot{W}_l^B = B^T A^T A^T [w_{.2} \quad w_{.3} \quad w_{.4}] =$$
$$[\epsilon\lambda 1 w_{12} + (\epsilon\lambda 1^2 + 1) w_{22} + \lambda 2 w_{32} + \lambda 2^2 w_{42}$$
$$\epsilon\lambda 1 w_{13} + (\epsilon\lambda 1^2 + 1) w_{23} + \lambda 2 w_{33} + \lambda 2^2 w_{43}$$
$$\epsilon\lambda 1 w_{14} + (\epsilon\lambda 1^2 + 1) w_{24} + \lambda 2 w_{34} + \lambda 2^2 w_{44}]$$

$$\ddot{Z}_{lu}^B = B^T A^T A^T [z_{.2} \quad z_{.3} \quad z_{.4}] =$$
$$[\epsilon\lambda 1 z_{12} + (\epsilon\lambda 1^2 + 1) z_{22} + \lambda 2 z_{32} + \lambda 2^2 z_{42}$$
$$\epsilon\lambda 1 z_{13} + (\epsilon\lambda 1^2 + 1) z_{23} + \lambda 2 z_{33} + \lambda 2^2 z_{43}$$
$$\epsilon\lambda 1 z_{14} + (\epsilon\lambda 1^2 + 1) z_{24} + \lambda 2 z_{34} + \lambda 2^2 z_{44}]$$

$$\ddot{Z}_{ll}^{B} = B^T A^T A^T [z_{2.} \;\; z_{3.} \;\; z_{4.}] =$$
$$[\epsilon\lambda 1 z_{21} + (\epsilon\lambda 1^2 + 1) z_{22} + \lambda 2 z_{23} + \lambda 2^2 z_{24}$$
$$\epsilon\lambda 1 z_{31} + (\epsilon\lambda 1^2 + 1) z_{32} + \lambda 2 z_{33} + \lambda 2^2 z_{34}$$
$$\epsilon\lambda 1 z_{41} + (\epsilon\lambda 1^2 + 1) z_{42} + \lambda 2 z_{43} + \lambda 2^2 z_{44}]$$

$$\ddot{W}_{r}^{B} = B^T A^T A^T [w_{.1}] = [\epsilon\lambda 1 w_{11} + (\epsilon\lambda 1^2 + 1) w_{21} + \lambda 2 w_{31} + \lambda 2^2 w_{41}]$$
$$\ddot{Z}_{ru}^{B} = B^T A^T A^T [z_{.1}] = [\epsilon\lambda 1 z_{11} + (\epsilon\lambda 1^2 + 1) z_{21} + \lambda 2 z_{31} + \lambda 2^2 z_{41}]$$
$$\ddot{Z}_{rl}^{B} = B^T A^T A^T [z_{1.}] = [\epsilon\lambda 1 z_{11} + (\epsilon\lambda 1^2 + 1) z_{12} + \lambda 2 z_{13} + \lambda 2^2 z_{14}]$$

We get the following systems, written in block diagonal form.

Each state vector is divided into two parts, a known portion x_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 and $\dot{u}_k$. 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, $\ddot{u}$:

$$\begin{bmatrix} \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & \dots & 0 & 0 & 0 \\ 0 & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -\ddot{W}_l^B & \ddot{Z}_{lu}^B & 0 \\ 0 & 0 & 0 & \dots & \ddot{Z}_{ll}^B & -\ddot{W}_l^B & \ddot{Z}_{lu}^B \end{bmatrix} \begin{bmatrix} x_0^v \\ x_1^v \\ x_2^v \\ \vdots \\ x_{n-2}^v \\ x_{n-1}^v \\ x_n^v \end{bmatrix} =$$

$$\begin{bmatrix} -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & \dots & 0 & 0 & 0 \\ 0 & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \ddot{W}_r^B & -\ddot{Z}_{ru}^B & 0 \\ 0 & 0 & 0 & \dots & -\ddot{Z}_{rl}^B & \ddot{W}_r^B & -\ddot{Z}_{ru}^B \end{bmatrix} \begin{bmatrix} x_0^p \\ x_1^p \\ x_2^p \\ \vdots \\ x_{n-2}^p \\ x_{n-1}^p \\ x_n^p \end{bmatrix}$$

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

$$\begin{aligned}\dot{u}_k(t_{k+1}) &= \dot{u}_{k+1}(t_{k+1})\\ \ddot{u}_k(t_{k+1}) &= \ddot{u}_{k+1}(t_{k+1})\\ \dddot{u}_k\,(t_{k+1}) &= \dddot{u}_{k+1}\,(t_{k+1})\end{aligned}$$

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, $\ddot{u}$:

$$\ddot{W}_l^B = B^T A^T A^T [w_{.2} \quad w_{.3} \quad w_{.4} \quad w_{.5}] =$$
$$[\epsilon\lambda 1 w_{12} + \epsilon\lambda 1^2 w_{22} + w_{32} + \lambda 2 w_{42} + \lambda 2^2 w_{52}$$
$$\epsilon\lambda 1 w_{13} + \epsilon\lambda 1^2 w_{23} + w_{33} + \lambda 2 w_{43} + \lambda 2^2 w_{53}$$
$$\epsilon\lambda 1 w_{14} + \epsilon\lambda 1^2 w_{24} + w_{34} + \lambda 2 w_{44} + \lambda 2^2 w_{54}]$$

$$\ddot{Z}_{lu}^B = B^T A^T A^T [z_{.2} \quad z_{.3} \quad z_{.4} \quad z_{.5}] =$$
$$[\epsilon\lambda 1 z_{12} + \epsilon\lambda 1^2 z_{22} + z_{32} + \lambda 2 z_{42} + \lambda 2^2 z_{52}$$
$$\epsilon\lambda 1 z_{13} + \epsilon\lambda 1^2 z_{23} + z_{33} + \lambda 2 z_{43} + \lambda 2^2 z_{53}$$
$$\epsilon\lambda 1 z_{14} + \epsilon\lambda 1^2 z_{24} + z_{34} + \lambda 2 z_{44} + \lambda 2^2 z_{54}]$$

$$\ddot{Z}_{ll}^B = B^T A^T A^T [z_{2.} \quad z_{3.} \quad z_{4.} \quad z_{5.}] =$$
$$[\epsilon\lambda 1 z_{21} + \epsilon\lambda 1^2 z_{22} + z_{23} + \lambda 2 z_{24} + \lambda 2^2 z_{25}$$
$$\epsilon\lambda 1 z_{31} + \epsilon\lambda 1^2 z_{32} + z_{33} + \lambda 2 z_{34} + \lambda 2^2 z_{35}$$
$$\epsilon\lambda 1 z_{41} + \epsilon\lambda 1^2 z_{42} + z_{43} + \lambda 2 z_{44} + \lambda 2^2 z_{45}]$$

$$\ddot{W}_r^B = B^T A^T A^T [w_{.1}] = [\epsilon\lambda 1 w_{11} + \epsilon\lambda 1^2 w_{21} + w_{31} + \lambda 2 w_{41} + \lambda 2^2 w_{51}]$$
$$\ddot{Z}_{ru}^B = B^T A^T A^T [z_{.1}] = [\epsilon\lambda 1 z_{11} + \epsilon\lambda 1^2 z_{21} + z_{31} + \lambda 2 z_{41} + \lambda 2^2 z_{51}]$$
$$\ddot{Z}_{rl}^B = B^T A^T A^T [z_{1.}] = [\epsilon\lambda 1 z_{11} + \epsilon\lambda 1^2 z_{12} + z_{13} + \lambda 2 z_{14} + \lambda 2^2 z_{15}]$$

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).

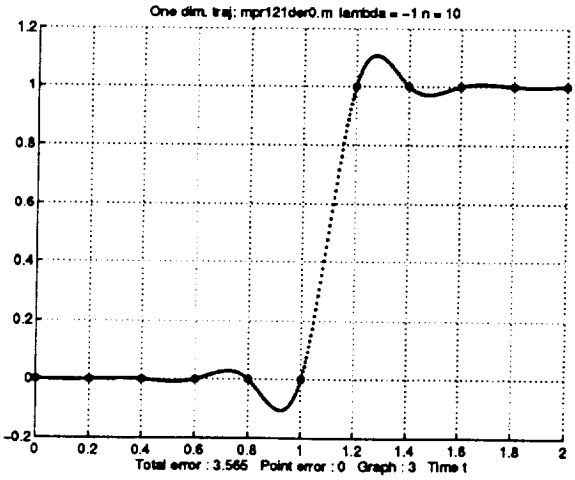

Figure 1: The step function tracked by system 1 with λ=0.

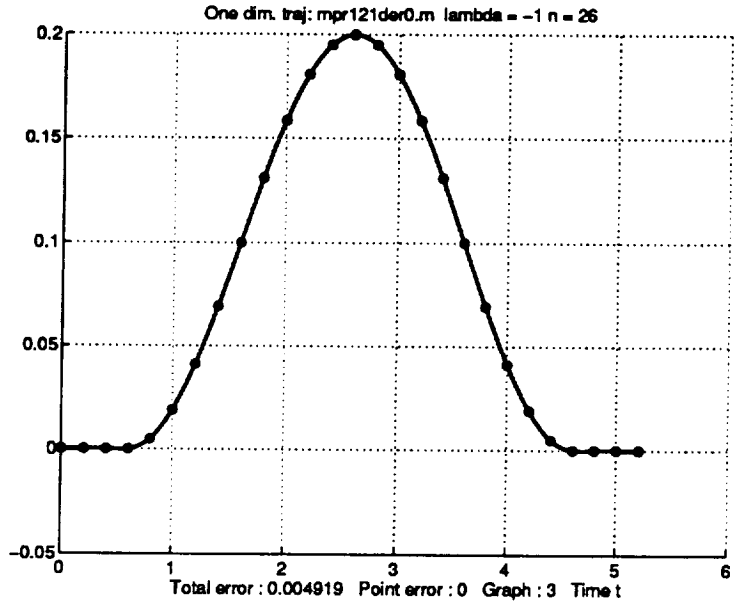

Figure 2: The sine function tracked by system 1 with λ=0.

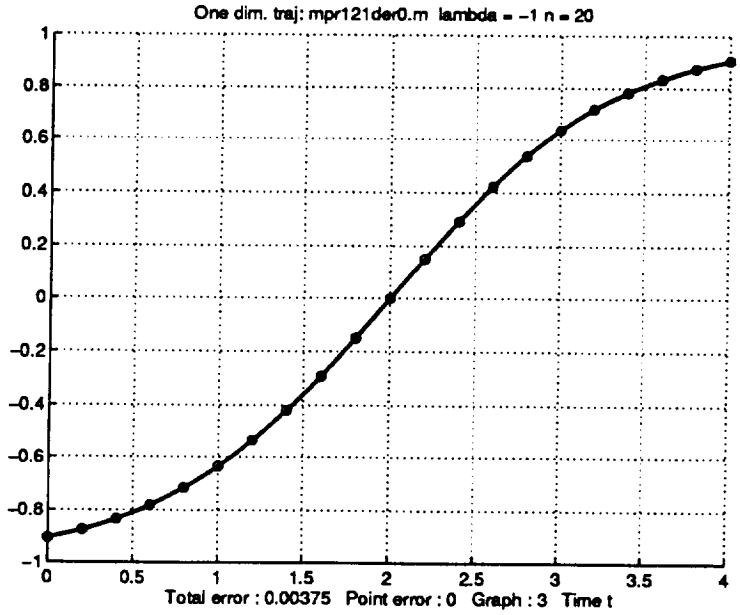

Figure 3: The tangent hyperbolic function tracked by system 1 with λ=0.

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

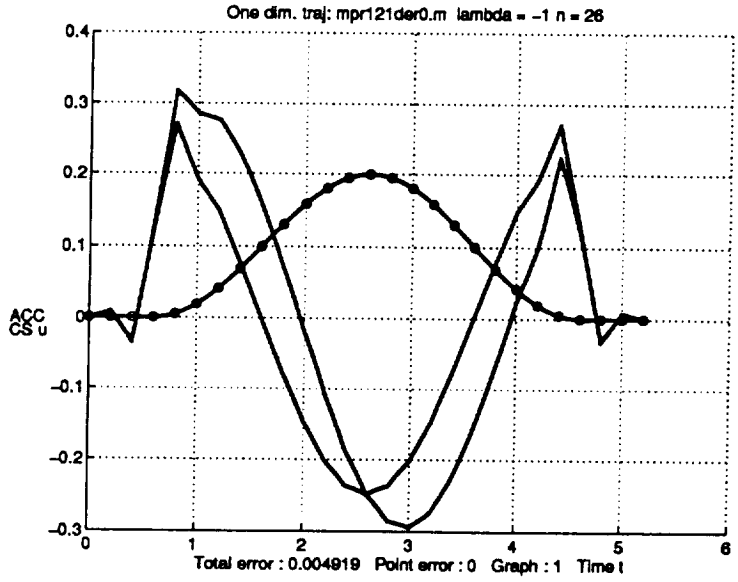

Figure 4: Sine tracked by system 1.

as the other two systems. Due to this it is not so very interesting and will therefore from now on be omitted.

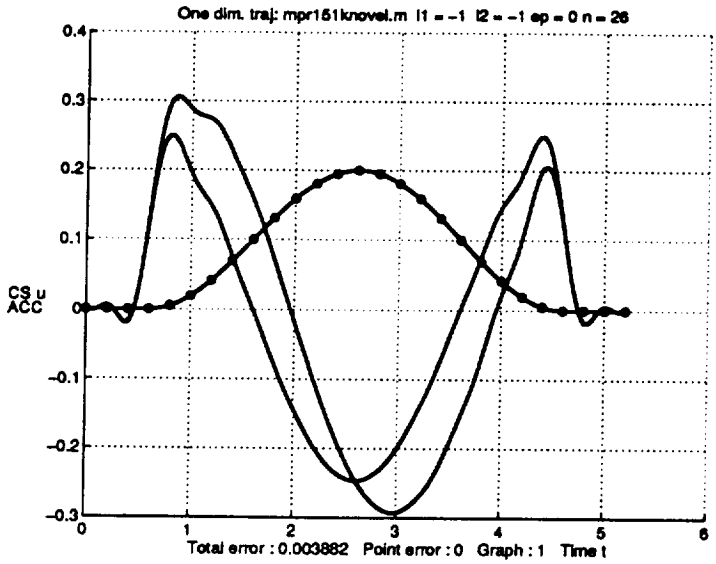

Figure 5: Sine tracked by system 3.

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.

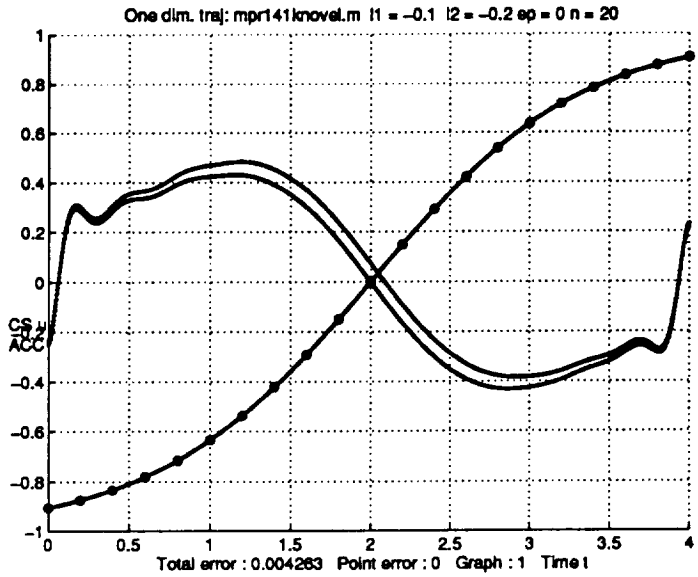

Figure 6: Tangent hyperbolic tracked by system 2.

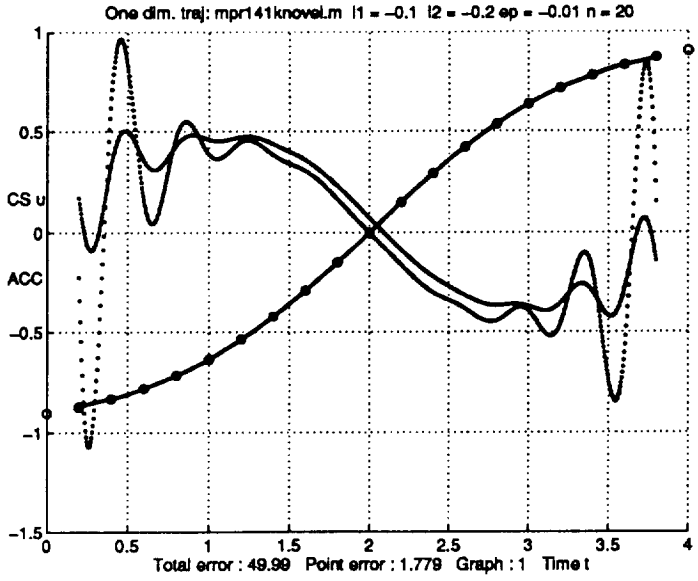

Figure 7: Tangent hyperbolic tracked by system 2.

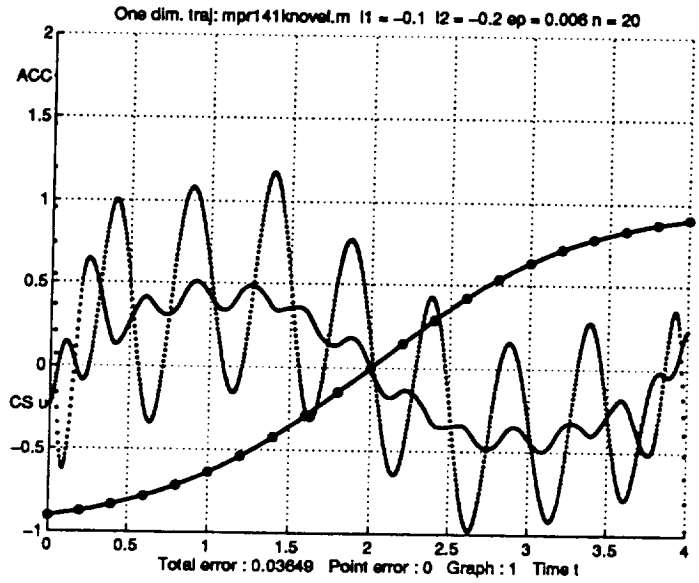

Figure 8: Tangent hyperbolic tracked by system 2.

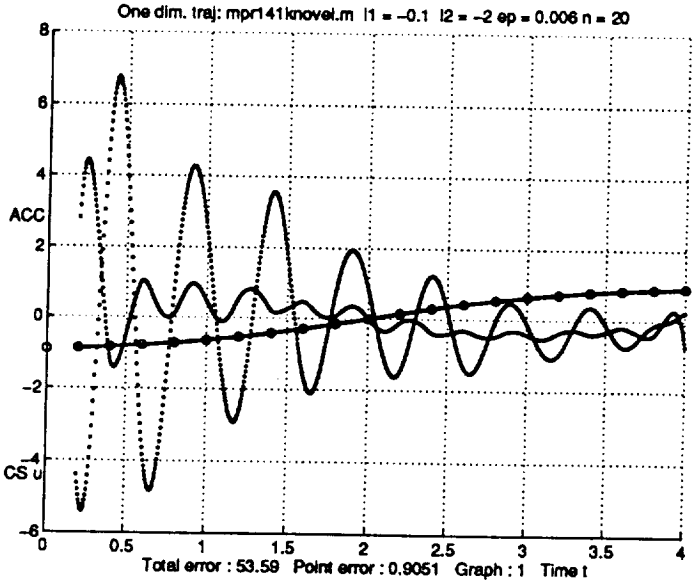

Figure 9: Tangent hyperbolic tracked by system 2.

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.

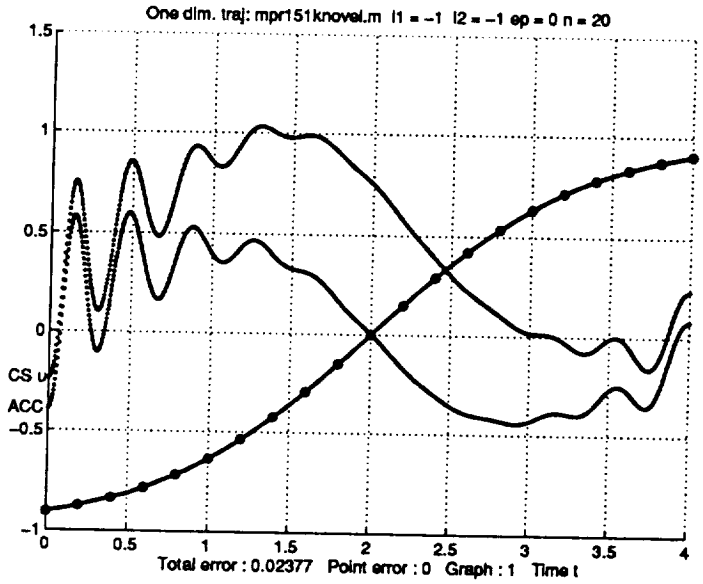

Figure 10: Tangent hyperbolic tracked by system 3.

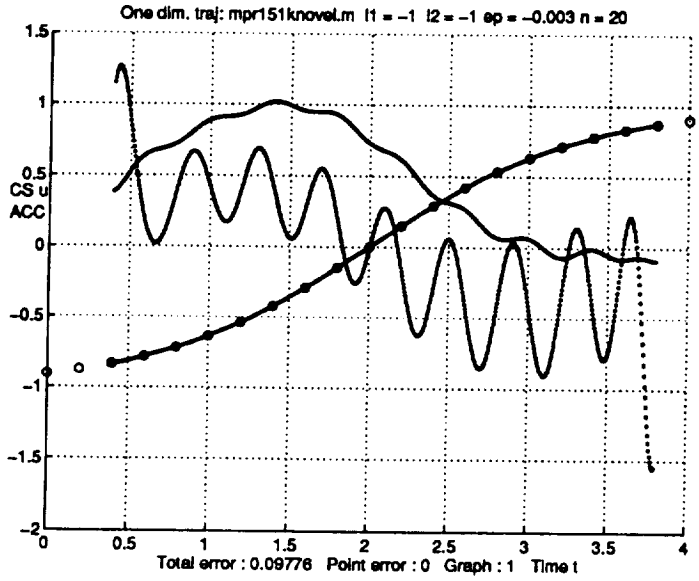

Figure 11: Tangent hyperbolic tracked by system 3.

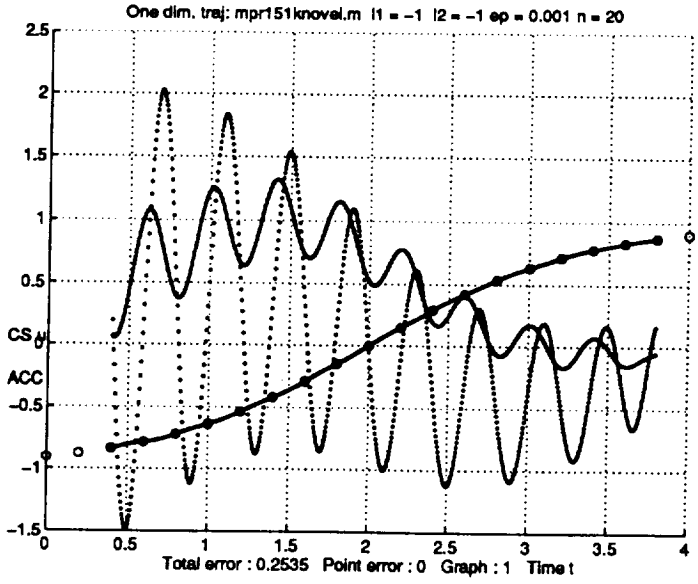

Figure 12: Tangent hyperbolic tracked by system 3.

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.

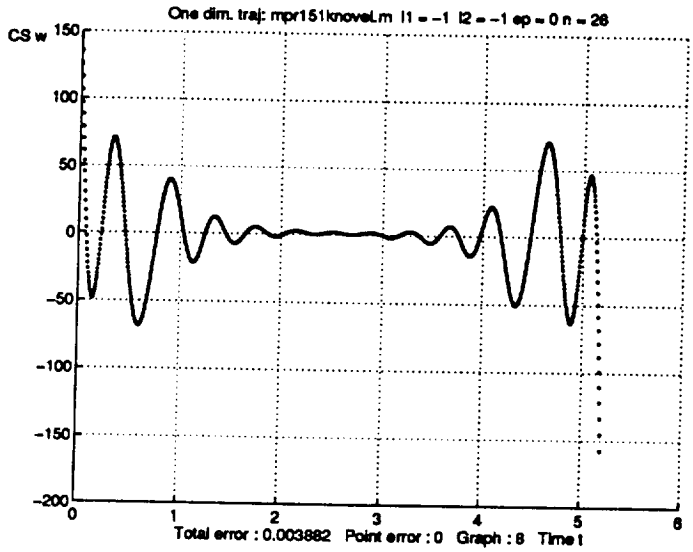

Figure 13: Sin tracked by system 3.

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.

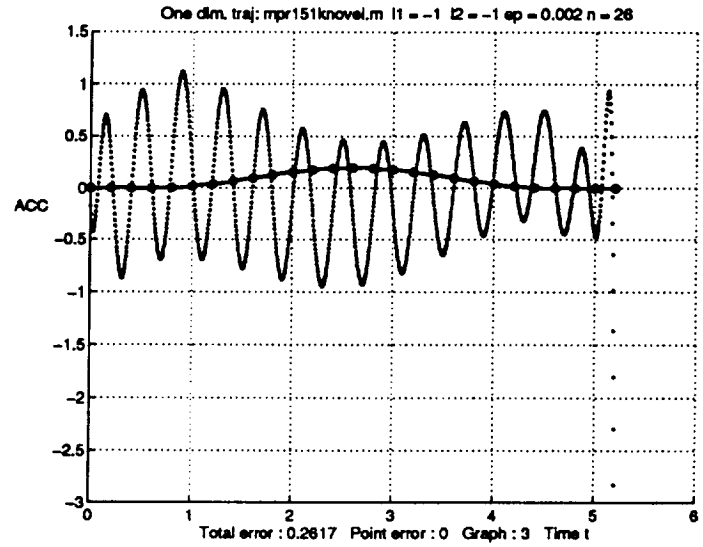

Figure 14: Sine tracked by system 3.

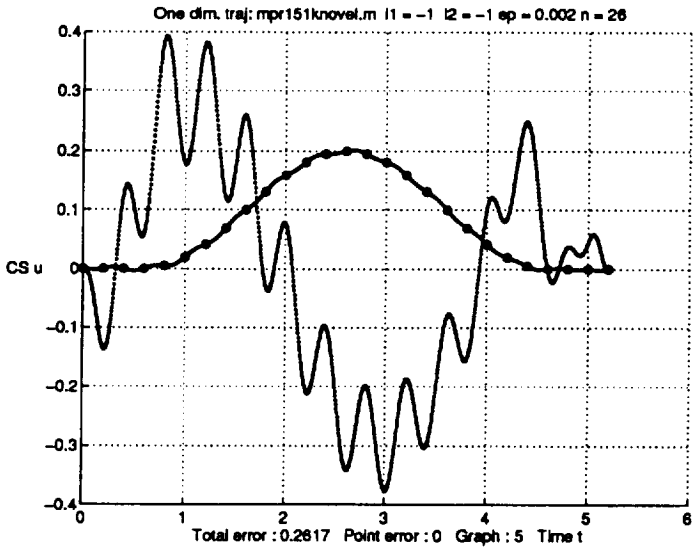

Figure 15: Sine tracked by system 3.

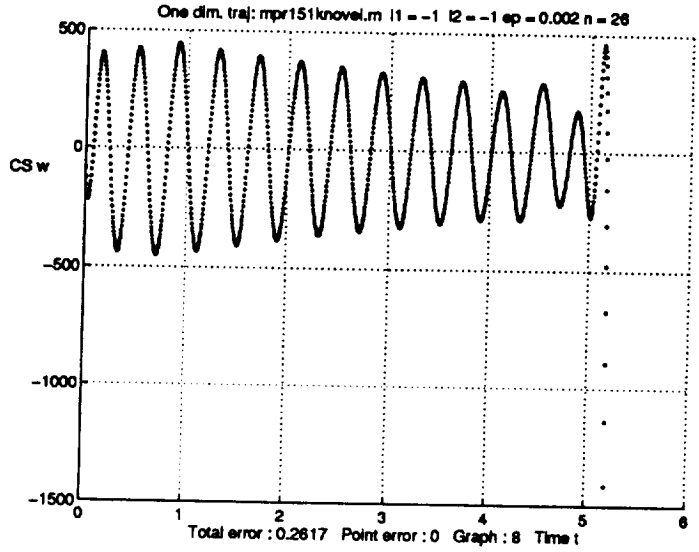

Figure 16: Sine tracked by system 3.

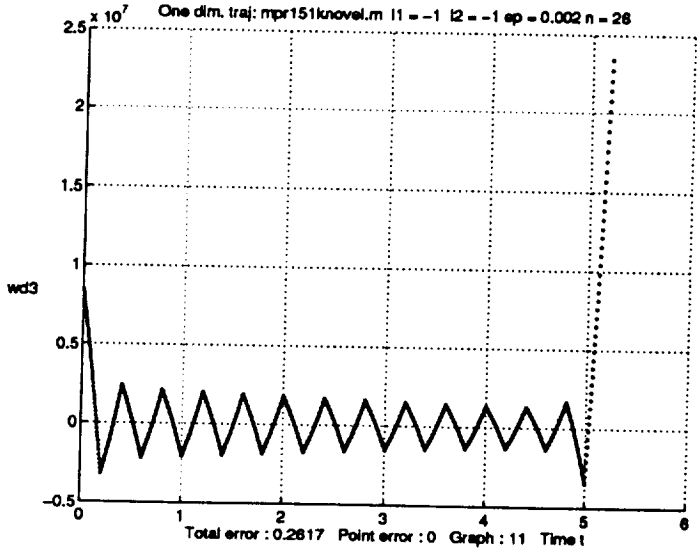

Figure 17: Sine tracked by system 3.

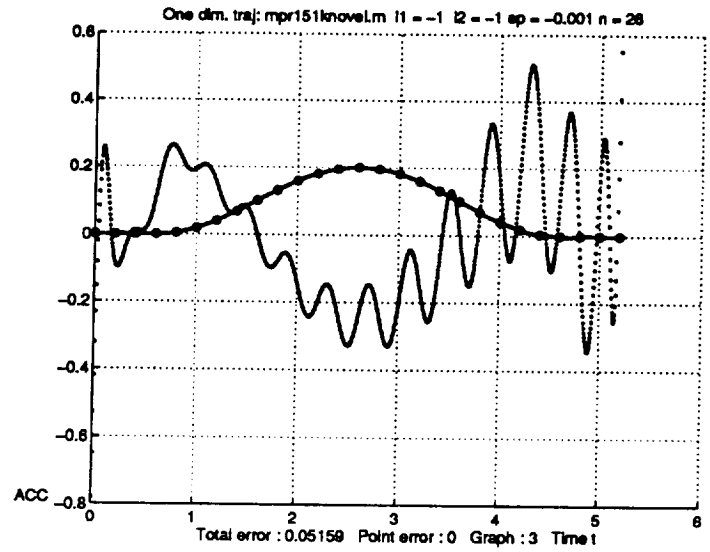

Figure 18: Sine tracked by system 3.

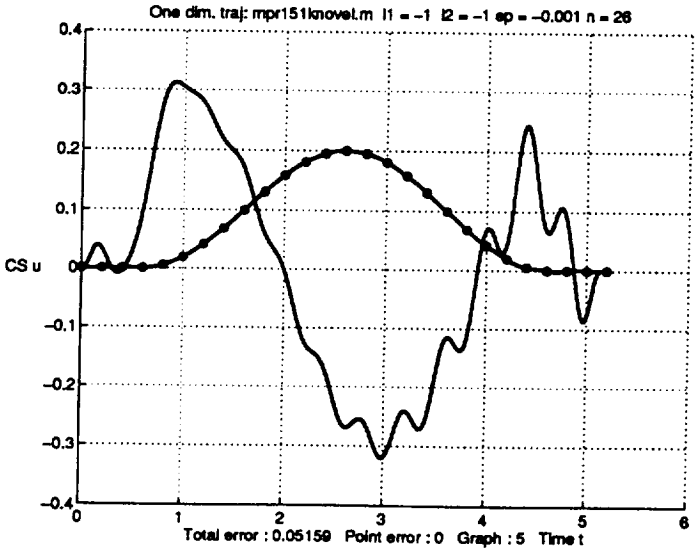

Figure 19: Sine tracked by system 3.

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.

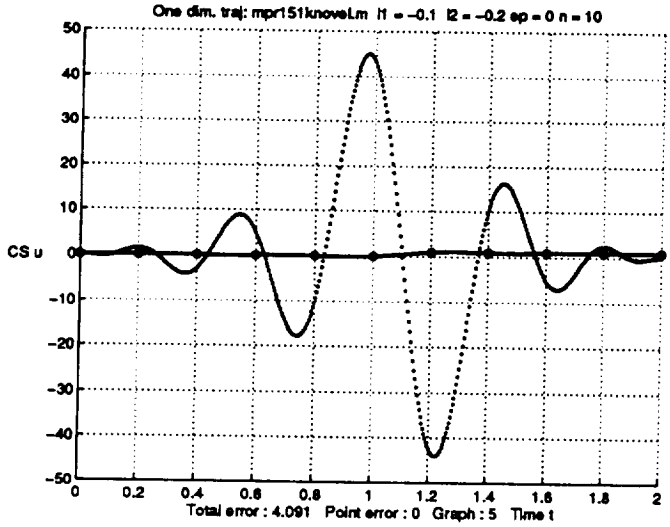

Figure 20: The step function tracked by system 3.

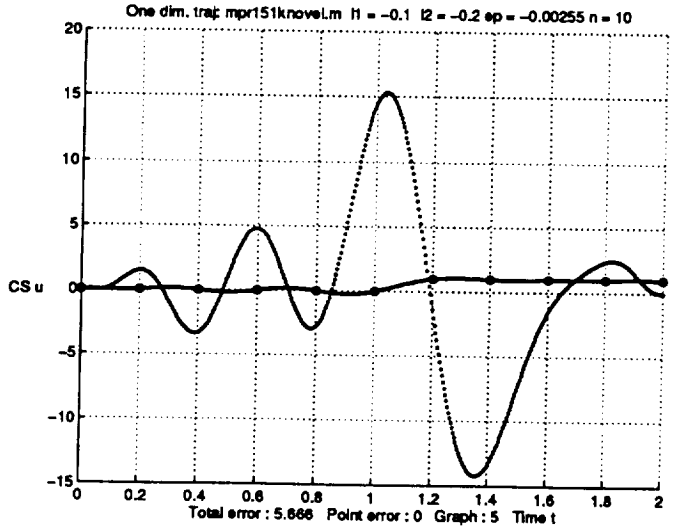

Figure 21: The step function tracked by system 3.

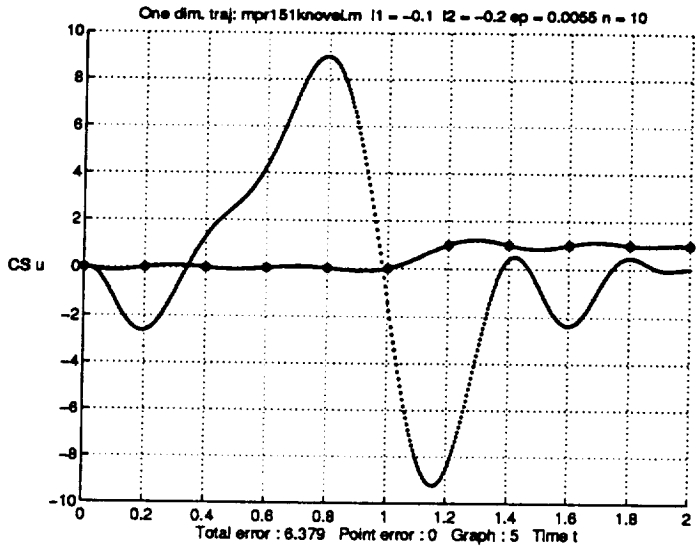

Figure 22: The step function tracked by system 3.

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.

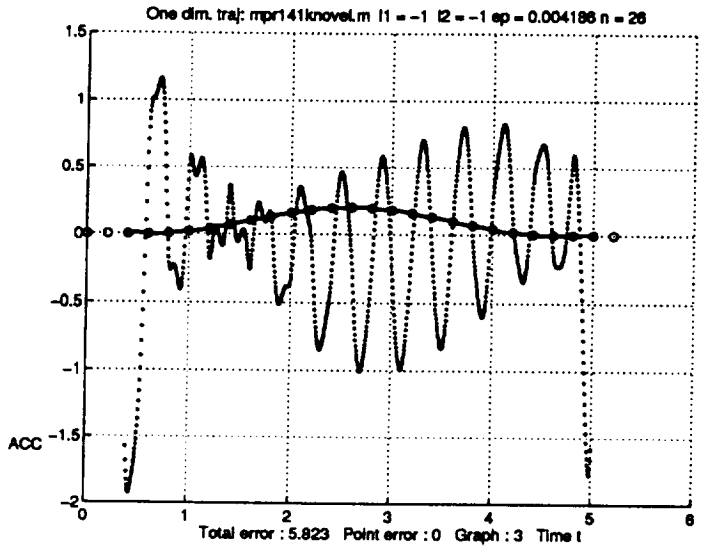

Figure 23: Sine tracked by system 2.

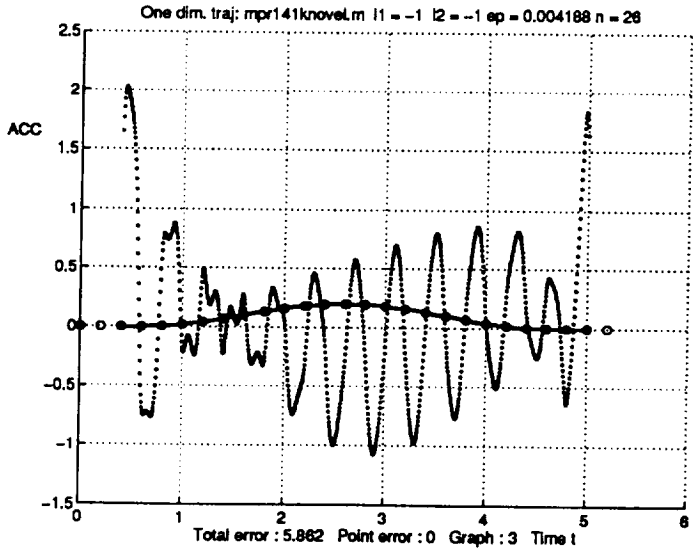

Figure 24: Sine tracked by system 2.

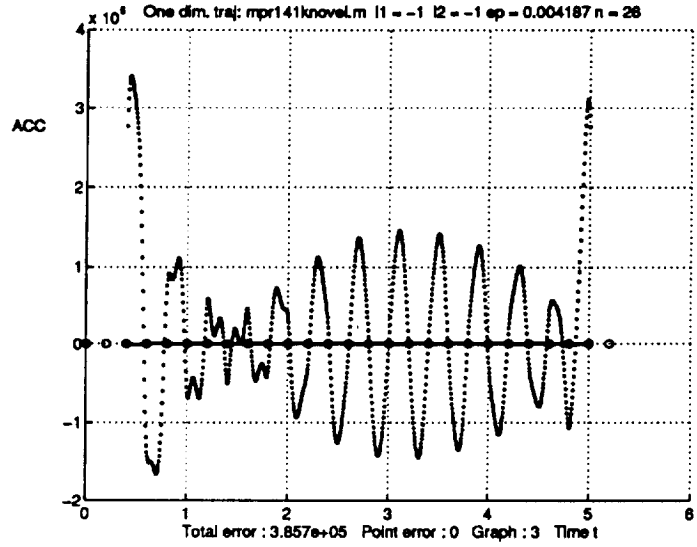

Figure 25: Sine tracked by system 2.

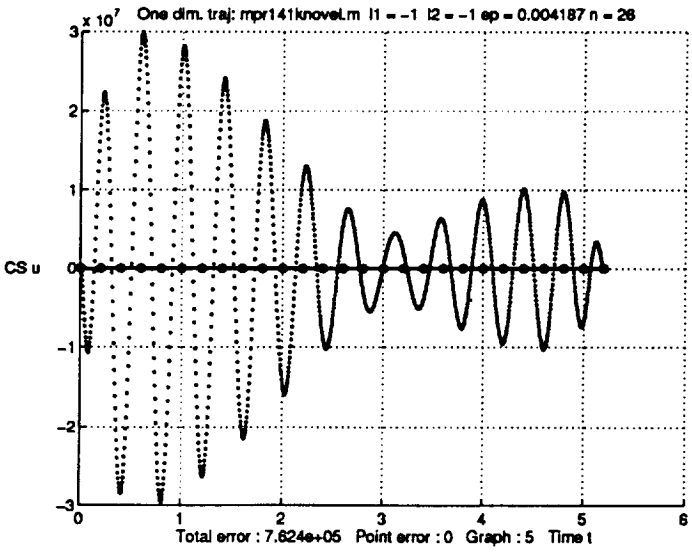

Figure 26: Sine tracked by system 2.

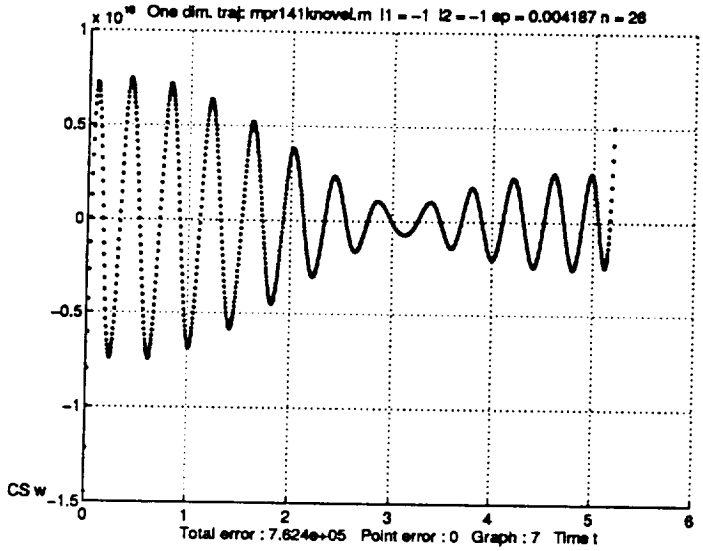

Figure 27: Sine tracked by system 2.

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 $u(t) = \ddot{f}(t)$. 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.

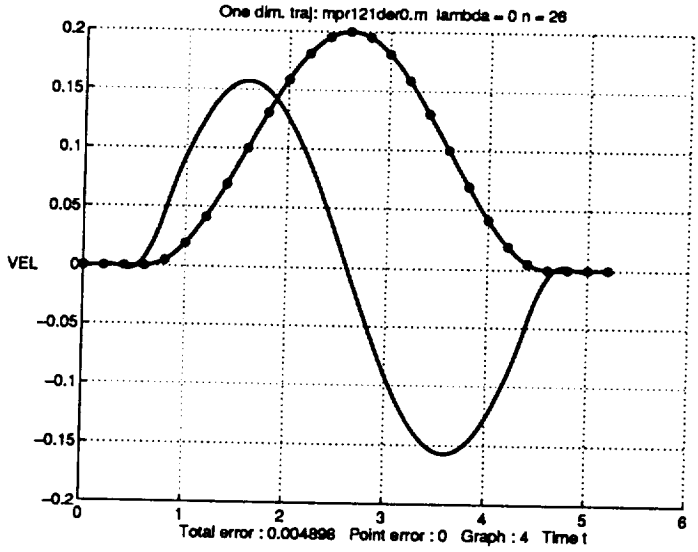

Figure 28: Sine tracked by system 1.

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  %%

WLDO=[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).