HOMOTOPY METHODS FOR SOLVING THE OPTIMAL PROJECTION EQUATIONS FOR THE REDUCED ORDER MODEL PROBLEM

by

Dragan Žigić

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

APPROVED:

Layne T. Watson

Stanoje Bingulac

Calvin J. Ribbens

June, 1991

Blacksburg, Virginia

LD
5655
V855
1991
Z 545
C.2

HOMOTOPY METHODS FOR SOLVING THE OPTIMAL PROJECTION EQUATIONS FOR THE REDUCED ORDER MODEL PROBLEM

by

Dragan Žigić

Committee Chairman: Layne T. Watson

Computer Science

(ABSTRACT)

The optimal projection approach to solving the reduced order model problem produces two coupled, highly nonlinear matrix equations with rank conditions as constraints. Due to the resemblance of these equations to standard matrix Lyapunov equations, they are called modified Lyapunov equations. The proposed algorithms utilize probability-one homotopy theory as the main tool. It is shown that there is a family of systems (the homotopy) that make a continuous transformation from some initial system to the final system. With a carefully chosen initial problem a theorem guarantees that all the systems along the homotopy path will be asymptotically stable, controllable and observable.

One method, which solves the equations in their original form, requires a decomposition of the projection matrix using the Drazin inverse of a matrix. It is shown that the appropriate inverse is a differentiable function. An effective algorithm for computing the derivative of the projection matrix that involves solving a set of Sylvester equations is given.

Another class of methods considers the equations in a modified form, using a decomposition of the pseudogramians based on a contragredient transformation. Some freedom is left in making an exact match between the number of equations and the number of unknowns, thus effectively generating a family of methods. Three strategies are considered for balancing the number of equations and unknowns. This approach proved to be very successful on a number of examples. The tests have shown that using the 'best' method practically always leads to a solution.

ACKNOWLEDGEMENTS.

I want to thank Dr. Layne Watson for all the help he gave me with this work. During numerous consultations, some of them lasting a few hours, I learned not only how to successfully complete this particular task, but also how to approach research problems, write papers, and many other related useful things.

I am thankful to the other two committee members, Dr. Stanoje Bingulac and Dr. Calvin Ribbens, for all their advice during the work, especially for help polishing the thesis.

There are a number of other people who helped me complete this task. The ideas for solving the problem that Dr. Christopher Beattie gave me were very fruitful, since starting from them the best results were obtained. Dr. Dennis Bernstein and Dr. Emmanuel Collins helped me to understand linear system theory, the problem and possible ways for solving it. I am thankful to them and to all the other people who directly or indirectly helped me in this work.

TABLE OF CONTENTS

1. Introduction 1

2. Elements of Linear System Theory 3

2.1. Description of the system 3

2.2. Controllability and reachability 3

2.3. Observability 8

2.4. Stability 10

2.5. Systems driven by white noise 12

3. Optimal Model-Reduction Problem 16

3.1. Statement of the problem 16

3.2. Contragredient transformation 16

3.3. (G, M, Γ)-factorization 18

3.4. The optimal projection theorem 19

4. Probability-One Homotopy Methods 30

5. Choosing an Initial System and the Starting Point 32

5.1. Theorem about the initial system 32

5.2. Choosing the starting point 34

6. Homotopy Method Based on the Drazin Inverse 38

6.1. Drazin inverse 38

6.2. Derivative of Drazin inverse 40

6.3. Computation of τ' 41

6.4. Algorithm 42

7. Methods Based on Decompositions of Pseudogramians 44

7.1. Description of methods 44

7.2. Algorithm 45

8. Quasi-Newton Techniques 47

8.1. Using Gramian modifications 47
8.2. Using solutions of intermediate problems 48
8.3. Using solutions of lower orders 48

9. Numerical Results 50
9.1. Models 50
9.2. Program measures 57
9.3. Comparison of different homotopy methods 58
9.4. Comparison of different choices of starting point 59
9.5. Choosing the starting point using Strategy 5 60
9.6. Comparison of optimal projection results and other methods 61
9.7. Choice of the initial system 62
9.8. Quasi-Newton techniques 62
9.9. Summary 64

10. Conclusion 66

References 67

Appendix A 70
Appendix B 75
Appendix C 77

Vita 98

LIST OF FIGURES

Figure 1. Solution obtained using Strategy 2 60
Figure 2. Solution obtained using Strategy 5 61
Figure 3. Bad choice of the initial system 63
Figure 4. Good choice of the initial system 63

LIST OF TABLES

Table 1. Program measures 58
Table 2. Number of tangent computations 59
Table 3. Comparison of the strategies for choosing the starting point 59
Table 4. Cost comparison between optimal projection and other methods 62

1. INTRODUCTION.

The optimal projection approach has been successfully applied to many problems arising in optimal control. Hyland and Bernstein [17] obtained theoretical results for the application of that method to the reduced order model problem, which is to find a reduced order model

$$\dot{x}_m(t) = A_m\, x_m(t) + B_m\, u(t),$$

$$y_m(t) = C_m\, x_m(t),$$

for the system

$$\dot{x}(t) = A\, x(t) + B\, u(t),$$

$$y(t) = C\, x(t),$$

which minimizes the quadratic model reduction criterion

$$J(A_m, B_m, C_m) \equiv \lim_{t\to\infty} E\left[(y - y_m)^t\, R\, (y - y_m)\right].$$

The necessary conditions for the optimal reduced order model are given in the form of two modified Lyapunov equations, matrix equations which resemble the (linear) matrix Lyapunov equations, but are highly nonlinear and mutually coupled. The projection matrix, which is involved in the equations, comes from a matrix decomposition which cannot be differentiated in an obvious way. It is desirable both to prove the theoretical correctness of an approach for differentiating the equations and to find an effective way to perform it.

Among many different approaches for solving the reduced order model problem are component cost analysis [26]–[28], balancing [23]–[24], Hankel-norm approximation [19]–[20], aggregation [1], [22], nonminimal partial realization [15] and the optimal reduction method of Wilson [36]. Some other applications of the optimal projection approach include the H_2/H_∞ model reduction problem [11], the fixed order dynamic compensation problem [16] and the reduced order state estimation problem [3]. A homotopy based algorithm for solving the fixed order dynamic compensation problem is given by Richter and Collins [25].

Some basic elements of the linear system theory are reviewed in Chapter 2. A review of theoretical results for solving the optimal model-reduction problem is given in Chapter 3. Chapter 4 contains a brief review of the homotopy theory. Chapter 5 is devoted to the choice

of an initial system and the starting point for the homotopy algorithm. A homotopy method based on the Drazin inverse is explained in Chapter 6. Three other homotopy methods based on decompositions of pseudogramians are explained in Chapter 7. A few approximation techniques for solving the problem are given in Chapter 8. Chapter 9 contains results from applying the methods to a number of examples. Beside solutions to the examples, some comparisons between the methods are given. Finally, Chapter 10 concludes with a brief summary and recommendations for future work.

2. ELEMENTS OF LINEAR SYSTEM THEORY.

2.1. Description of the system.

A linear differential finite dimensional system which is continuous time and time invariant (stationary) can be represented by the equations:

$$\dot{x}(t) = A\,x(t) + B\,u(t), \tag{1}$$

$$y(t) = C\,x(t), \tag{2}$$

where t is time, $x(t) \in R^n$, $u(t) \in R^m$, $y(t) \in R^l$, $A \in R^{n\times n}$, $B \in R^{n\times m}$ and $C \in R^{l\times n}$. $x(t)$ is the *state* of the system, $u(t)$ is the *input* (control) and $y(t)$ is the *output* of the system. The equation (1) is called the *state equation* and (2) is called the *output equation.* A, B and C are constant matrices that characterize the behavior of the system.

Sections 2.2–2.4 present three important properties of linear systems. The material in 2.2 and 2.3 on controllability and observability follows Delchamps [8], and the material on stability in 2.4 follows Kwakernaak and Sivan [21]. The material in Section 2.5 also follows [21].

2.2. Controllability and reachability.

The controllability and the reachability of the system (1)–(2) are determined by the pair (A, B), so it is sufficient to consider only the equation (1).

DEFINITION 1. A vector $x_1 \in R^n$ is *reachable with respect to* (A, B) if and only if there exist $T > 0$ and a piecewise continuous input u on $[0, T]$ such that

$$x_1 = \int_0^T e^{A(T-\tau)}\,B\,u(\tau)\,d\tau,$$

where

$$e^A \equiv I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots. \tag{3}$$

(A, B) is a *reachable pair* if and only if every $x_1 \in R^n$ is reachable with respect to (A, B).

DEFINITION 2. (A, B) is a *controllable pair* if and only if for every $x_0 \in R^n$ there exist $T > 0$ and a piecewise continuous input u on $[0, T]$ such that

$$0 = e^{A\,T}\,x_0 + \int_0^T e^{A(T-\tau)}\,B\,u(\tau)\,d\tau.$$

Whether the system is controllable or not can be determined by considering the controllability matrix or the controllability Gramian.

DEFINITION 3. The *controllability matrix* of the matrix pair (A, B) is

$$M_c \equiv (\,B \quad A\,B \quad A^2\,B \quad \ldots \quad A^{n-1}\,B\,).$$

DEFINITION 4. For given $T > 0$ the *controllability Gramian* of the matrix pair (A, B) is

$$Q_c(T) \equiv \int_0^T e^{A(T-\tau)}\,B\,B^t\,e^{A^t(T-\tau)}\,d\tau.$$

$Q_c(T)$ is obviously a symmetric matrix. Moreover, it is positive semidefinite since

$$x^t\,Q_c(T)\,x = \int_0^T \|B^t\,e^{A^t(T-\tau)}\,x\|^2\,d\tau \geq 0.$$

The following lemmas will help establish controllability tests.

LEMMA 5. *The kernel of $Q_c(T)$ is the same for every $T > 0$, and equals the set of all $x \in R^n$ such that $x^t\,M_c = 0$.*

Proof. Let $T > 0$ be given. If x is in the kernel of $Q_c(T)$ then

$$Q_c(T)\,x = 0 \quad \Rightarrow \quad x^t\,Q_c(T)\,x = 0 \quad \Rightarrow \quad \int_0^T \|B^t\,e^{A^t(T-\tau)}\,x\|^2\,d\tau = 0.$$

Since the integrand is nonnegative, it implies

$$B^t\,e^{A^t(T-\tau)}\,x = 0 \quad \text{for } 0 \leq \tau \leq T. \tag{4}$$

If derivatives of (4) are taken at $\tau = T$ then

$$(-1)^i\,B^t\,(A^t)^i\,x = 0 \quad \text{for } i \geq 0. \tag{5}$$

From (5) it follows that $x^t\,A^i\,B = 0$ for $i \geq 0$, so $x^t\,M_c = 0$.

Conversely, suppose $x^t M_c = 0$. Since by the Cayley-Hamilton theorem $A^i B$ for any $i > 0$ can be expressed as a linear combination of $\{B, AB, \ldots, A^{n-1}B\}$, it follows that $x^t A^i B = 0$ for $i \geq 0$. Due to (3), it means that

$$B^t e^{A^t(T-\tau)} x = \sum_{i=0}^{\infty} \frac{(T-\tau)^i}{i!} B^t (A^t)^i x = 0 \quad \text{for } 0 \leq \tau \leq T.$$

Hence,

$$Q_c(T)\, x = \int_0^T e^{A(T-\tau)} B B^t e^{A^t(T-\tau)} x \, d\tau = 0$$

and x is in the kernel of $Q_c(T)$. Q. E. D.

LEMMA 6. *The range of $Q_c(T)$ is the same for every $T > 0$ and equals the range of M_c.*

Proof. Firstly, note that for symmetric matrices the range is the orthogonal complement of the kernel, since there exists an orthonormal basis consisting of eigenvectors of the matrix, and the kernel is the space spanned by the eigenvectors corresponding to the zero eigenvalue.

Let $T > 0$ be given. By Lemma 5, $x \in R^n$ is in the kernel of symmetric $Q_c(T)$ if and only if it is orthogonal to every column of M_c. That means that the kernel of $Q_c(T)$ is the orthogonal complement of the range of M_c. Due to the observation about symmetric matrices, it follows that the ranges of $Q_c(T)$ and M_c are the same. Q. E. D.

The following theorems are reachability and controllability tests.

THEOREM 7. *(A, B) is a reachable pair if and only if* rank $M_c = n$.

Proof. If $x_1 \in R^n$ is reachable with respect to (A, B) then it can be expressed as

$$x_1 = \int_0^T e^{A(T-\tau)} B\, u(\tau)\, d\tau = \sum_{i=0}^{\infty} A^i B \int_0^T \frac{(T-\tau)^i}{i!} u(\tau)\, d\tau.$$

Due to the Cayley-Hamilton theorem x_1 is a linear combination of the columns of $\{B, AB, \ldots, A^{n-1}B\}$, hence x_1 is in the range of M_c.

If x_1 is in the range of M_c then, by Lemma 6, it is in the range of $Q_c(T)$ for any $T > 0$, so there exists $y \in R^n$ such that $x_1 = Q_c(T)\, y$. If $u(\tau) \equiv B^t e^{A^t(T-\tau)} y$ then

$$x_1 = Q_c(T)\, y = \int_0^T e^{A(T-\tau)} B B^t e^{A^t(T-\tau)} y \, d\tau = \int_0^T e^{A(T-\tau)} B\, u(\tau)\, d\tau$$

and x_1 is reachable with respect to (A, B).

Hence, $x_1 \in R^n$ is reachable with respect to (A, B) if and only if it is in the range of M_c. Therefore, due to Definition 1, the matrix pair (A, B) is reachable if and only if the range of M_c is all of R^n, i.e., rank $M_c = n$. Q. E. D.

THEOREM 8. *(A, B) is a reachable pair if and only if $Q_c(T)$ is positive definite for every $T > 0$.*

Proof. Due to Lemma 6 and Theorem 7, (A, B) is a reachable pair if and only if rank $Q_c(T) = n$ for every $T > 0$. Since $Q_c(T)$ is positive semidefinite, this means that (A, B) is a reachable pair if and only if all the eigenvalues of $Q_c(T)$ are positive. Q. E. D.

The following theorem relates reachability and controllability for continuous time systems.

THEOREM 9. *(A, B) is a reachable pair if and only if it is a controllable pair.*

Proof. Let (A, B) be a reachable pair. For given $x_0 \in R^n$ define $x_1 \equiv -e^{A\,T} x_0$ and choose $T > 0$ and $u(t)$ such that

$$x_1 = \int_0^T e^{A(T-\tau)}\, B\, u(\tau)\, d\tau.$$

Then,

$$0 = e^{A\,T} x_0 + \int_0^T e^{A(T-\tau)}\, B\, u(\tau)\, d\tau$$

and (A, B) is a controllable pair.

Let (A, B) be a controllable pair and suppose it is not reachable. Then, by Theorem 7, there exists a nonzero $x \in R^n$ such that $x^t M_c = 0$. By the Cayley-Hamilton theorem it would mean that $x^t\, A^i\, B = 0$ for $i \geq 0$. Since (A, B) is controllable, it is possible to choose T and $u(t)$ such that

$$0 = e^{A\,T} x + \int_0^T e^{A(T-\tau)}\, B\, u(\tau)\, d\tau. \tag{6}$$

If both sides of (6) are multiplied by $x^t\, e^{-A\,T}$ from the left then

$$0 = x^t\, x + x^t \int_0^T e^{-A\,\tau}\, B\, u(\tau)\, d\tau.$$

Due to the Cayley-Hamilton theorem

$$x^t \int_0^T e^{-A\tau} B\, u(\tau)\, d\tau = x^t \sum_{i=0}^{\infty} A^i B \int_0^T \frac{(-\tau)^i}{i!} u(\tau)\, d\tau = 0.$$

Therefore, $x^t x = 0$ and $x = 0$, which is a contradiction. Q. E. D.

REMARK 10. *(A, B) is a controllable pair if and only if* rank $M_c = n$. *(A, B) is a controllable pair if and only if $Q_c(T)$ is positive definite for every $T > 0$.*

Proof. It follows from Theorems 7–9. Q. E. D.

Since the general solution of the system

$$\dot{x}(t) = A\, x(t) \tag{7}$$

is given by

$$x(t) = e^{A(t-t_0)}\, x_0, \tag{8}$$

the general solution of (1) is given by

$$x(t) = e^{A(t-t_0)}\, x_0 + \int_{t_0}^{t} e^{A(t-\tau)} B\, u(\tau)\, d\tau. \tag{9}$$

Using (9) and Definitions 1 and 2, the controllability and reachability can be interpreted in the following way. (A, B) is a controllable pair if and only if for any $x_0 = x(0)$ there exist $T > 0$ and input $u(t)$ on $[0, T]$ such that $x(T) = 0$. (A, B) is a reachable pair if and only if for any $x_1 \in R^n$ there exist $T > 0$ and input $u(t)$ on $[0, T]$ such that starting from $x(0) = 0$ the state of the system (1) can be driven to $x(T) = x_1$.

REMARK 11. *If $x_1 \in R^n$ is reachable with respect to (A, B) then it can be reached in any time $T > 0$.*

Proof. Let $T > 0$ be given. If $x_1 \in R^n$ is reachable with respect to (A, B) then, due to Lemma 6 and Theorem 7, x_1 is in the range of $Q_c(T)$ and there exists $y \in R^n$ such that

$$x_1 = Q_c(T)\, y = \int_0^T e^{A(T-\tau)} B\, B^t\, e^{A^t(T-\tau)}\, y\, d\tau.$$

With the choice of the input $u(\tau) = B^t\, e^{A^t(T-\tau)}\, y$

$$x_1 = \int_0^T e^{A(T-\tau)} B\, u(\tau)\, d\tau$$

which means that x_1 can be reached in time T. Q. E. D.

2.3. Observability.

The observability is related to the matrix pair (A, C), so the system (1)–(2) is considered.

DEFINITION 12. A vector $x_0 \in R^n$ is *unobservable with respect to the matrix pair* (A, C) if and only if $C\, e^{A\, t}\, x_0 = 0$ for $t > 0$. The matrix pair (A, C) is *observable* if and only if the only $x \in R^n$ which is unobservable with respect to (A, C) is $x = 0$.

As in the case of controllability, the observability matrix and the observability Gramian have crucial roles in determining observability.

DEFINITION 13. The *observability matrix* of the matrix pair (A, C) is

$$M_o \equiv \begin{pmatrix} C \\ C\, A \\ \vdots \\ C\, A^{n-1} \end{pmatrix}.$$

DEFINITION 14. For given $T > 0$ the *observability Gramian* of the matrix pair (A, C) is

$$Q_o(T) \equiv \int_0^T e^{A^t\, \tau}\, C^t\, C\, e^{A\, \tau}\, d\tau.$$

$Q_o(T)$ is obviously a symmetric matrix. Moreover, it is positive semidefinite since

$$x^t\, Q_o(T)\, x = \int_0^T x^t\, e^{A^t\, \tau}\, C^t\, C\, e^{A\, \tau}\, x\, d\tau = \int_0^T \| C\, e^{A\, \tau}\, x \|^2 d\tau \geq 0.$$

The following theorem gives a test for observability.

THEOREM 15. *The matrix pair* (A, C) *is observable if and only if* rank $M_o = n$.

Proof. Let $x_0 \neq 0$ be an unobservable vector with respect to (A, C). Then

$$C\, e^{A\, t}\, x_0 = 0 \quad \text{for } t > 0. \tag{10}$$

If derivatives of (10) are taken at $t = 0$ it follows that

$$C\, x_0 = C\, A\, x_0 = \ldots = C\, A^{n-1}\, x_0 = 0, \tag{11}$$

which means $M_o\, x_0 = 0$.

On the other side, if $M_o\, x_0 = 0$, then (11) is true and, due to the Cayley-Hamilton theorem

$$\sum_{i=0}^{\infty} \frac{t^i}{i!} C\, A^i\, x_0 = C\, e^{A\, t}\, x_0 = 0,$$

which means that x_0 is unobservable with respect to (A, C).

Hence, (A, C) is not an observable pair if and only if there is a nonzero vector x_0 in the kernel of M_o. Equivalently, (A, C) is an observable pair if and only if there are no nonzero vectors in the kernel of M_o, i.e., rank $M_o = n$. Q. E. D.

The following lemma will help establish another test for observability.

LEMMA 16. *The kernel of $Q_o(T)$ is the same for every $T > 0$ and coincides with the kernel of M_o.*

Proof. Let $T > 0$ be given. Suppose $Q_o(T)\, x_0 = 0$. That means

$$x_0^t\, Q_o(T)\, x_0 = \int_0^T \|C\, e^{A\, \tau}\, x_0\|^2\, d\tau = 0,$$

which implies $C\, e^{A\, t}\, x_0 = 0$ for $0 \leq t \leq T$. From (10) and (11) in the proof of Theorem 15, x_0 is in the kernel of M_o.

If $M_o\, x_0 = 0$ then, from the proof of Theorem 15, x_0 is unobservable with respect to (A, C), i.e., $C\, e^{A\, t}\, x_0 = 0$ for $0 \leq t \leq T$. Since

$$Q_o(T)\, x_0 = \int_0^T e^{A^t\, \tau}\, C^t\, C\, e^{A\, \tau}\, x_0\, d\tau = 0$$

x_0 is in the kernel of $Q_o(T)$. Q. E. D.

The following theorem gives an observability test in terms of the observability Gramian.

THEOREM 17. *(A, C) is an observable pair if and only if $Q_o(T)$ is positive definite for every $T > 0$.*

Proof. Let $T > 0$ be given. From Theorem 15 and Lemma 16 it follows that (A, C) is an observable pair if and only if the kernel of $Q_o(T)$ contains only the zero vector, i.e., $Q_o(T)$ is nonsingular. Since $Q_o(T)$ is positive semidefinite, (A, C) is an observable pair if and only if all the eigenvalues of $Q_o(T)$ are positive. Q. E. D.

From (2) and (9) the output of the system (1)–(2) can be expressed as

$$y(t) = C\, e^{A(t-t_0)}\, x_0 + C \int_{t_0}^{t} e^{A(t-\tau)}\, B\, u(\tau)\, d\tau.$$

Let $y(t; x_0)$ denote the output of the system (1)–(2) with the initial state $x(0) = x_0$. Now,

$$y(t; x_0) = y(t; x_0') \quad \Leftrightarrow \quad C\, e^{A\,t}\, x_0 = C\, e^{A\,t}\, x_0' \quad \Leftrightarrow \quad C\, e^{A\,t}(x_0 - x_0') = 0. \qquad (12)$$

From (12) it is clear that $x_0 - x_0' \in R^n$ is unobservable with respect to (A, C) if and only if $y(t; x_0) = y(t; x_0')$ for $t > 0$. In other words, (A, C) is not observable when for every input two different initial states x_0 and x_0' lead to the same output.

2.4. Stability.

The stability of the system (1)–(2) is determined by the matrix A, so it is sufficient to consider only the equation (7).

DEFINITION 18. A solution $x_0(t)$ of the system (7), which is given by (8), is *stable* if for any t_0 and any $\epsilon > 0$ there exists a $\delta(\epsilon, t_0)$ such that for any other solution $x(t)$,

$$\|x(t_0) - x_0(t_0)\| \le \delta \quad \Rightarrow \quad \|x(t) - x_0(t)\| \le \epsilon \quad \text{for all } t \ge t_0.$$

Furthermore, the solution $x_0(t)$ is *asymptotically stable* if it is stable and for any t_0 there exists a $\rho(t_0) > 0$ such that for any other solution $x(t)$

$$\|x(t_0) - x_0(t_0)\| \le \rho \quad \Rightarrow \quad \|x(t) - x_0(t)\| \to 0 \quad \text{as } t \to \infty.$$

The system (7) is asymptotically stable if every solution $x_0(t)$ is asymptotically stable, which occurs if *any* solution is asymptotically stable. In this case the matrix A is also referred to as asymptotically stable.

A criterion for determining the asymptotic stability of the system (7) is given in the next theorem.

THEOREM 19. *The system* (7) *is asymptotically stable if and only if all the eigenvalues of A have negative real parts.*

Proof. Firstly, suppose that A has distinct eigenvalues $\lambda_1, \ldots, \lambda_n$. Then, there exists a matrix T such that $A = T\,\Lambda\,T^{-1}$, where $\Lambda = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$. Let $g_1, \ldots, g_n$ be the columns of T and $f_1, \ldots, f_n$ the rows of T^{-1}. Now, using (8) and the fact that for diagonal Λ

$$e^{T\,\Lambda\,T^{-1}} = T\,e^{\Lambda}\,T^{-1},$$

the solution of the system (7) becomes

$$\begin{aligned} x(t) &= e^{A(t-t_0)}\,x_0 = e^{T\,\Lambda\,T^{-1}(t-t_0)}\,x_0 \\ &= T\,e^{\Lambda(t-t_0)}\,T^{-1}\,x_0. \end{aligned} \tag{13}$$

Since

$$e^{\Lambda(t-t_0)} = \text{diag}\,\left(e^{\lambda_1(t-t_0)}, \ldots, e^{\lambda_n(t-t_0)}\right),$$

(13) can be transformed into

$$x(t) = \sum_{i=1}^{n} e^{\lambda_i(t-t_0)}\,g_i\,f_i\,x_0 = \sum_{i=1}^{n} e^{\lambda_i(t-t_0)}\,\mu_i\,g_i, \tag{14}$$

for some constants μ_i, $i = 1, \ldots, n$. From (14) it is clear that the system (7) is asymptotically stable if and only if all λ_i have negative real parts.

If some eigenvalues are repeated, then A has the Jordan decomposition $A = T\,J\,T^{-1}$, where $J = \text{diag}\,(J_1, \ldots, J_k)$, $J_i = \text{diag}\,(J_{i1}, \ldots, J_{il_i})$, and each J_{ij} has the form

$$J_{ij} = \begin{pmatrix} \lambda_i & 1 & \ldots & 0 & 0 \\ 0 & \lambda_i & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \lambda_i & 1 \\ 0 & 0 & \ldots & 0 & \lambda_i \end{pmatrix}.$$

In this case

$$\begin{aligned} e^{A\,t} &= T\,e^{J\,t}\,T^{-1}, \\ e^{J\,t} &= \text{diag}\,(e^{J_1\,t}, \ldots, e^{J_k\,t}), \\ e^{J_i\,t} &= \text{diag}\,(e^{J_{i1}\,t}, \ldots, e^{J_{il_i}\,t}), \\ e^{J_{ij}\,t} &= e^{\lambda_i\,t} \begin{pmatrix} 1 & t & \frac{t^2}{2} & \ldots & \frac{t^{n_{ij}-1}}{(n_{ij}-1)!} \\ 0 & 1 & t & \ldots & \frac{t^{n_{ij}-2}}{(n_{ij}-2)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}. \end{aligned}$$

Let

$$T = (T_1 \quad \dots \quad T_k), \quad T^{-1} = \begin{pmatrix} U_1 \\ \vdots \\ U_k \end{pmatrix},$$

where these partitions match that of J. Now, the solution of (7) can be expressed as

$$x(t) = \sum_{i=1}^{k} T_i\, e^{J_i\,(t-t_0)}\, U_i\, x(t_0) = \sum_{i=1}^{k} T_i\, e^{J_i\,(t-t_0)}\, S_i,$$

where S_i, T_i, $i = 1, \dots, k$ are constant matrices. From this expression, and the formulas for $e^{J_i\,t}$ it follows that $x(t) \to 0$ for $t \to \infty$ if and only if all the eigenvalues λ_i have negative real parts. Q. E. D.

2.5. Systems driven by white noise.

DEFINITION 20. A *vector stochastic process* $v(t)$ is

$$v(t) \equiv \begin{pmatrix} v_1(t) \\ v_2(t) \\ \dots \\ v_n(t) \end{pmatrix},$$

where $v_1(t), v_2(t), \dots, v_n(t)$ are scalar stochastic processes for $t \geq t_0$ for some t_0. A stochastic process is a collection of random variables $\{x(t), t \in T\}$ indexed by a time parameter t varying in a time range T.

DEFINITION 21. The *mean* $m(t)$ of a vector stochastic process $v(t)$ is $m(t) \equiv E[v(t)]$. The *covariance matrix* $R(t_1, t_2)$ of $v(t)$ is $R(t_1, t_2) \equiv E\{[v(t_1) - m(t_1)]\,[v(t_2) - m(t_2)]^t\}$. The *variance matrix* $Q(t)$ of $v(t)$ is $Q(t) \equiv R(t, t) = E\{[v(t) - m(t)]\,[v(t) - m(t)]^t\}$.

The *delta function* $\delta(t)$, which is used in the definition of white noise processes, is an idealization of a very narrow pulse with a finite area which is normalized to 1. Intuitively,

$$\delta(t) = 0 \quad \text{for } t \neq 0, \quad \delta(0) = \infty \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(t)\, dt = 1.$$

If $g(t)$ is a function which is "well behaved" in the neighborhood of $t = 0$ then

$$\int_{-\infty}^{\infty} \delta(t)\, g(t)\, dt = g(0).$$

DEFINITION 22. Let $v(t)$ be a zero-mean stochastic process with the covariance matrix $R(t_1, t_2) = V(t_1)\,\delta(t_2 - t_1)$, where $V(t)$ is a positive semidefinite matrix and $\delta(t)$ is the delta function. Then, the process $v(t)$ is a *white noise* process with intensity $V(t)$.

The following two lemmas will be used in the proof of Theorem 25.

LEMMA 23. *Let $v(t)$ be a white noise process with intensity V, and $A_1, A_2 \in R^{n\times n}$. Then*

$$E\left\{\left[\int_{t_0}^{t} A_1\, v(\tau_1)\, d\tau_1\right]\left[\int_{t_0}^{t} A_2\, v(\tau_2)\, d\tau_2\right]^t\right\} = \int_{t_0}^{t} A_1\, V\, A_2^t\, d\tau_1.$$

Proof. From the definition of white noise processes, Fubini's theorem [13] and properties of $\delta(t)$ it follows that

$$\begin{aligned} E\left\{\left[\int_{t_0}^{t} A_1\, v(\tau_1)\, d\tau_1\right]\left[\int_{t_0}^{t} A_2\, v(\tau_2)\, d\tau_2\right]^t\right\} &= \int_{t_0}^{t}\left\{\int_{t_0}^{t} A_1\, E\left[v(\tau_1)\, v^t(\tau_2)\right] A_2^t\, d\tau_2\right\} d\tau_1 \\ &= \int_{t_0}^{t} A_1\left[\int_{t_0}^{t} V\,\delta(\tau_2 - \tau_1) d\tau_2\right] A_2^t\, d\tau_1 \\ &= \int_{t_0}^{t} A_1\, V\, A_2^t\, d\tau_1. \end{aligned}$$

Q. E. D.

LEMMA 24. *Let $A \in R^{n\times n}$, $B \in R^{m\times m}$ and $C \in R^{n\times m}$. If λ_i, $i = 1, \ldots, n$ are the eigenvalues of A, and μ_j, $j = 1, \ldots, m$ are the eigenvalues of B, then the equation $A\,X + X\,B = C$ has a unique solution X if and only if $\lambda_i + \mu_j \neq 0$ for all i and j.*

A proof is given in Gantmacher [9].

THEOREM 25. *Consider the stochastic differential equation*

$$\dot{x}(t) = A\,x(t) + B\,v(t), \tag{15}$$

where A and B are constant and $v(t)$ is a white noise process with constant intensity V. Then, if A is asymptotically stable, as $t \to \infty$ the variance matrix of $x(t)$ tends to the constant positive semidefinite matrix

$$\bar{Q} = \int_0^{\infty} e^{A\,t}\, B\, V\, B^t\, e^{A^t\, t}\, dt,$$

which is the unique solution of the matrix equation

$$A\,\bar{Q} + \bar{Q}\,A^t + B\,V\,B^t = 0.$$

Proof. The general solution of (15) is given by

$$x(t) = e^{A(t-t_0)}\,x_0 + \int_{t_0}^{t} e^{A(t-\tau)}\,B\,v(\tau)\,d\tau.$$

Using the definition of the variance matrix, it follows that

$$\begin{aligned} E\left[x(t)\,x^t(t)\right] &= e^{A(t-t_0)}\,E(x_0\,x_0^t)\,e^{A^t(t-t_0)} \\ &+ E\left\{\left[e^{A(t-t_0)}\,x_0\right]\left[\int_{t_0}^{t} e^{A(t-\tau)}\,B\,v(\tau)\,d\tau\right]^t\right\} \\ &+ E\left\{\left[\int_{t_0}^{t} e^{A(t-\tau)}\,B\,v(\tau)\,d\tau\right]\left[e^{A(t-t_0)}\,x_0\right]^t\right\} \\ &+ E\left\{\left[\int_{t_0}^{t} e^{A(t-\tau)}\,B\,v(\tau)\,d\tau\right]\left[\int_{t_0}^{t} e^{A(t-\tau)}\,B\,v(\tau)\,d\tau\right]^t\right\}. \end{aligned}$$

Since $v(t)$ is zero-mean the second term becomes

$$E\left[e^{A(t-t_0)}\,x_0\right]\left[\int_{t_0}^{t} e^{A(t-\tau)}\,B\,E[v(t)]\,d\tau\right]^t = 0.$$

Similarly, the third term is equal to zero.

Applying Lemma 23, the fourth term is transformed to

$$\int_{t_0}^{t} e^{A(t-\tau)}\,B\,V\,B^t\,e^{A^t(t-\tau)}\,d\tau.$$

Hence, the variance has the following form

$$Q(t) = e^{A(t-t_0)}\,Q_0\,e^{A^t(t-t_0)} + \int_{t_0}^{t} e^{A(t-\tau)}\,B\,V\,B^t e^{A^t(t-\tau)}\,d\tau, \tag{16}$$

where $Q_0 \equiv E(x_0\,x_0^t)$. Differentiating (16) the following equation is obtained

$$\dot{Q}(t) = A\,Q(t) + Q(t)\,A^t + B\,V\,B^t. \tag{17}$$

From (16), for asymptotically stable A, using change of variables $s = t - \tau$, it follows that the limiting matrix of $Q(t)$ is

$$\bar{Q} = \lim_{t \to \infty} Q(t) = \int_0^\infty e^{A\tau} B V B^t e^{A^t \tau} d\tau.$$

Since $Q(t)$ is a solution of (17), its limit $\bar{Q}$, which is a constant matrix, must satisfy that equation, so

$$0 = A\bar{Q} + \bar{Q}A^t + BVB^t. \tag{18}$$

Since A is asymptotically stable, by Theorem 19 it has all the eigenvalues in the open left half plane. Therefore there are no two eigenvalues of A that sum to zero, and by Lemma 24 the solution $\bar{Q}$ of (18) is unique. Q. E. D.

3. OPTIMAL MODEL-REDUCTION PROBLEM.

3.1. Statement of the problem.

Given the asymptotically stable, controllable and observable time invariant continuous time system

$$\dot{x}(t) = A\,x(t) + B\,u(t), \tag{19}$$

$$y(t) = C\,x(t), \tag{20}$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{l\times n}$, the goal is to find a reduced order model

$$\dot{x}_m(t) = A_m\,x_m(t) + B_m\,u(t), \tag{21}$$

$$y_m(t) = C_m\,x_m(t), \tag{22}$$

where $A_m \in R^{n_m\times n_m}$, $B_m \in R^{n_m\times m}$, $C_m \in R^{l\times n_m}$, $n_m < n$, which minimizes the quadratic model-reduction criterion

$$J(A_m, B_m, C_m) \equiv \lim_{t\to\infty} E\left[(y - y_m)^t\,R\,(y - y_m)\right],$$

where the input $u(t)$ is white noise with symmetric and positive definite intensity V and R is a symmetric and positive definite weighting matrix.

It is assumed that A is asymptotically stable and diagonalizable, and a solution (A_m, B_m, C_m) is sought in the set

$$\left\{(A_m, B_m, C_m):\ A_m \text{ is stable, } (A_m, B_m) \text{ is controllable and } (A_m, C_m) \text{ is observable}\right\}.$$

3.2. Contragredient transformation.

The following theorem from [17], which is a special case of a result in [10], gives a sufficient condition for simultaneous reduction of two symmetric positive semidefinite matrices to a diagonal form using a contragredient transformation. The proof given here is different from that in [17].

THEOREM 26. [17] *Let symmetric positive semidefinite* Q, $P \in R^{n\times n}$ *satisfy*

$$\text{rank } (Q) = \text{rank } (P) = \text{rank } (Q\,P) = n_m, \tag{23}$$

where $n_m \leq n$. *Then, there exists a nonsingular* $W \in R^{n\times n}$ *(contragredient transformation) and positive definite diagonal* Σ, $\Omega \in R^{n_m \times n_m}$ *such that*

$$Q = W \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} W^t, \quad P = W^{-t} \begin{pmatrix} \Omega & 0 \\ 0 & 0 \end{pmatrix} W^{-1}. \tag{24}$$

Proof. Since P is positive semidefinite and symmetric there exists orthogonal $V \in R^{n\times n}$ such that

$$P = V \begin{pmatrix} D_1 & 0 \\ 0 & 0 \end{pmatrix} V^t,$$

where $D_1 \in R^{n_m \times n_m}$ is diagonal and positive definite. Let

$$T_1 \equiv V \begin{pmatrix} D_1^{-1/2} & 0 \\ 0 & I \end{pmatrix}.$$

Then

$$T_1^t\, P\, T_1 = \begin{pmatrix} I_{n_m} & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad T_1^{-1}\, Q\, T_1^{-t} = \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix},$$

where $Q_1 \in R^{n_m \times n_m}$. Since rank $(Q\,P)$ = rank (P) implies that Q is one-to-one on the eigenspace of P corresponding to positive eigenvalues, and the quadratic form $x^t\, Q_1\, x$ corresponds to the quadratic form $y^t\, Q\, y$ restricted to this eigenspace, and $y^t\, Q\, y = 0$ implies $Q\, y = 0$ for symmetric positive semidefinite Q, it follows that Q_1 is also symmetric positive definite. Therefore, there exists a positive definite diagonal $D_2 \in R^{n_m \times n_m}$ and orthogonal $U \in R^{n_m \times n_m}$ such that $Q_1 = U\, D_2\, U^t$. Let

$$T_2 \equiv \begin{pmatrix} U & 0 \\ Q_{12}^t\, U\, D_2^{-1} & -I \end{pmatrix}.$$

Then

$$T_2^{-1} = \begin{pmatrix} U^t & 0 \\ Q_{12}^t U D_2^{-1} U^t & -I \end{pmatrix},$$

$$T_2^t T_1^t P T_1 T_2 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix},$$

$$\begin{aligned} T_2^{-1} T_1^{-1} Q T_1^{-t} T_2^{-t} &= \begin{pmatrix} U^t & 0 \\ Q_{12}^t U D_2^{-1} U^t & -I \end{pmatrix} \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} U & U D_2^{-1} U^t Q_{12} \\ 0 & -I \end{pmatrix} \\ &= \begin{pmatrix} U^t Q_1 U & U^t Q_{12} - U^t Q_{12} \\ Q_{12}^t U - Q_{12}^t U & Q_2 - Q_{12}^t U D_2^{-1} U^t Q_{12} \end{pmatrix} \\ &= \begin{pmatrix} D_2 & 0 \\ 0 & Q_2 - Q_{12}^t U D_2^{-1} U^t Q_{12} \end{pmatrix} \\ &= \begin{pmatrix} D_2 & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

The last equality is a consequence of the rank conditions (23). If $W \equiv T_1 T_2$ then

$$Q = W \begin{pmatrix} D_2 & 0 \\ 0 & 0 \end{pmatrix} W^t, \quad P = W^{-t} \begin{pmatrix} I_{n_m} & 0 \\ 0 & 0 \end{pmatrix} W^{-1}, \tag{25}$$

which completes the proof. Q. E. D.

REMARK 27. [17] *Let Q and P be as in Theorem 26. Then there exists a nonsingular $U \in R^{n \times n}$ and positive definite diagonal $\Lambda \in R^{n_m \times n_m}$ such that*

$$Q = U \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} U^t, \quad P = U^{-t} \begin{pmatrix} \Lambda & 0 \\ 0 & 0 \end{pmatrix} U^{-1}.$$

Proof. The statement follows from (25) using $U \equiv W \begin{pmatrix} D_2^{1/4} & 0 \\ 0 & I \end{pmatrix}$. Q. E. D.

3.3. (G, M, Γ)-factorization.

The following lemma defines a factorization which plays an important role in the optimal projection approach for solving the reduced order model problem. The proof here is slightly different from that of [17].

LEMMA 28. [17] *Let symmetric positive semidefinite $\hat{Q}$, $\hat{P} \in R^{n \times n}$ satisfy the rank conditions* (23). *Then, there exist G, $\Gamma \in R^{n_m \times n}$ and positive semisimple (*positive semisimple *means similar to a symmetric positive definite matrix) $M \in R^{n_m \times n_m}$ such that*

$$\hat{Q} \hat{P} = G^t M \Gamma, \tag{26}$$

$$\Gamma G^t = I_{n_m}. \tag{27}$$

Proof. Due to Remark 27 there exist nonsingular $W \in R^{n\times n}$ and positive definite diagonal $\Sigma \in R^{n_m \times n_m}$ such that

$$\hat{Q} = W \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} W^t, \quad \hat{P} = W^{-t} \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} W^{-1}. \tag{28}$$

The equations (28) can be expressed in the equivalent form

$$\hat{Q} = W_1 \Sigma W_1^t, \quad \hat{P} = U_1^t \Sigma U_1, \tag{29}$$

where

$$W = \begin{pmatrix} \overbrace{}^{n_m} & \\ W_1 & W_2 \end{pmatrix}, \quad W^{-1} = U = \begin{matrix} \\ {}^{n_m}\{ \end{matrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}. \tag{30}$$

From (29)–(30) with $G \equiv W_1^t$ and $\Gamma \equiv U_1$ follow (26) and (27). Q. E. D.

Matrices G, M and Γ from Lemma 28 are a (G, M, Γ)-*factorization* of $(\hat{Q}, \hat{P})$.

3.4. The optimal projection theorem.

Lemmas 29, 36, 37 and 38 will be used in the proof of Theorem 39. The following lemma defines the Sylvester inequality (Chen [6]).

LEMMA 29. *Let* $A \in R^{q\times n}$, $B \in R^{n\times p}$. *Then,*

$$\text{rank } (A) + \text{rank } (B) - n \leq \text{rank } (AB) \leq \min(\text{rank } (A), \text{ rank } (B)).$$

Proof. Since dim im $(AB) \leq$ dim im (B), and im (AB) is a subspace of im (A), it follows that

$$\text{rank } (AB) \leq \min(\text{rank } (A), \text{ rank } (B)).$$

On the other side, rank $(AB) =$ rank $(B) - d$, where d is the dimension of im $(B) \cap$ ker (A). Since the dimension of ker (A) is equal to $n -$ rank (A), it follows that

$$\text{rank } (AB) \geq \text{rank } (A) + \text{rank } (B) - n,$$

which completes the proof. Q. E. D.

The following lemma is due to Wonham [37].

LEMMA 30. *If* $C_1^t C_1 = C_2^t C_2$ *and the matrix pair* (A, C_1) *is observable, then the matrix pair* (A, C_2) *is observable.*

Proof. Let

$$V(C) \equiv \bigcap_{i=0}^{n-1} \ker (C A^i), \quad W(C) \equiv \sum_{i=0}^{n-1} (A^i)^t C^t C A^i.$$

If some vector is in $V(C)$ then it is in the kernel of every $C A^i$, hence it is in the kernel of every $(A^i)^t C^t C A^i$, and it is in the kernel of $W(C)$. Conversely, if some vector x is in the kernel of $W(C)$, then

$$x^t \left(\sum_{i=0}^{n-1} (A^i)^t C^t C A^i \right) x = \sum_{i=0}^{n-1} \| C A^i x \|^2 = 0,$$

Hence, $C A^i x = 0$ for $i = 0, 1, \ldots, n-1$, and $x \in V(C)$. Therefore,

$$V(C) = \ker (W(C)). \tag{31}$$

Since $C_1^t C_1 = C_2^t C_2$, then $W(C_1) = W(C_2)$ and $\ker (W(C_1)) = \ker (W(C_2))$. Using (31) it follows

$$V(C_1) = V(C_2). \tag{32}$$

By assumption (A, C_1) is an observable pair, hence the appropriate observability matrix has full rank. That means that the only vector which is in the kernels of all $C_1 A^i$ is the zero vector. Due to (32) the only vector which is in the kernels of all $C_2 A^i$ is the zero vector, hence the observability matrix for (A, C_2) has full rank, which means that (A, C_2) is an observable pair. Q. E. D.

LEMMA 31. *If* $B_1 B_1^t = B_2 B_2^t$ *and the matrix pair* (A, B_1) *is controllable, then the matrix pair* (A, B_2) *is controllable.*

Proof. The proof is analogous to the proof of Lemma 30. Q. E. D.

If X is a symmetric positive semidefinite matrix, then $\sqrt{X}$ denotes the matrix Y such that $X = Y\,Y$. The matrix Y can be constructed using the spectral decomposition of X in this way:

$$X = T\,\Sigma^2\,T^t = (T\,\Sigma\,T^t)\,(T\,\Sigma\,T^t) = Y\,Y.$$

LEMMA 32. *If (A, C) is an observable pair and R is a positive definite matrix, then $(A, \sqrt{R}\,C)$ is an observable pair.*

Proof. From Theorem 15 the observability matrix M_o for (A, C) has full rank. If every $C\,A^i$ for $i = 0, 1, \ldots, n-1$ is replaced in M_o by $\sqrt{R}\,C\,A^i$ the resulting matrix will also have full rank. Since it is the observability matrix for the matrix pair $(A, \sqrt{R}\,C)$, it follows that $(A, \sqrt{R}\,C)$ is an observable pair. Q. E. D.

LEMMA 33. *If (A, B) is a controllable pair and V is a positive definite matrix, then $(A, B\sqrt{V})$ is a controllable pair.*

Proof. The proof is analogous to the proof of Lemma 32. Q. E. D.

REMARK 34. *If (A, C) is an observable pair and R is symmetric and positive definite, then $(A, \sqrt{C^t\,R\,C})$ is an observable pair.*

Proof. From Lemma 32 it follows that $(A, \sqrt{R}\,C)$ is an observable pair. Due to Lemma 30 $(A, \sqrt{C^t\,R\,C})$ is an observable pair. Q. E. D.

REMARK 35. *If (A, B) is a controllable pair and V is symmetric and positive definite, then $(A, \sqrt{B\,V\,B^t})$ is a controllable pair.*

Proof. The proof is analogous to the proof of Remark 34. Q. E. D.

LEMMA 36. *If matrix A is asymptotically stable then the Gramians*

$$Q_c \equiv \int_0^\infty e^{A\,t}\,B\,V\,B^t\,e^{A^t\,t}\,dt, \tag{33}$$

$$Q_o \equiv \int_0^\infty e^{A^t\,t}\,C^t\,R\,C\,e^{A\,t}\,dt \tag{34}$$

are the unique solutions of the dual Lyapunov equations

$$A\,Q_c + Q_c\,A^t + B\,V\,B^t = 0, \tag{35}$$

$$A^t\,Q_o + Q_o\,A + C^t\,R\,C = 0. \tag{36}$$

Proof. Since A is asymptotically stable, due to Lemma 24, the equations (35)–(36) have unique solutions.

Since A is stable it follows that there exist constants K and $\alpha > 0$ such that $|e^{A\,t}| \leq K\,e^{-\alpha\,t}$ for $t \geq 0$ (Hale [12]).

The statement about (35) follows since on one side

$$\int_0^\infty \frac{d}{dt}\left(e^{A\,t}\,B\,V\,B^t\,e^{A^t\,t}\right)\,dt = \int_0^\infty \left(A\,e^{A\,t}\,B\,V\,B^t\,e^{A^t\,t} + e^{A\,t}\,B\,V\,B^t\,e^{A^t\,t}\,A^t\right)\,dt$$
$$= A\,Q_c + Q_c\,A^t,$$

and on the other side

$$\int_0^\infty \frac{d}{dt}\left(e^{A\,t}\,B\,V\,B^t\,e^{A^t\,t}\right)\,dt = e^{A\,t}\,B\,V\;B^t\,e^{A^t\,t}\Big|_0^\infty = -B\,V\,B^t.$$

That Q_o satisfies (36) can be proved in an analogous way. Q. E. D.

LEMMA 37. *If A is asymptotically stable, (A, B) is a controllable pair, (A, C) is an observable pair, and V and R are symmetric and positive definite, then the unique solutions Q_c and Q_o of the Lyapunov equations* (35)–(36) *are symmetric positive definite.*

Proof. Solutions of (35)–(36) are unique due to Lemma 24 and Theorem 19 since A is asymptotically stable.

From Remarks 34 and 35 it follows that $(A, \sqrt{B\,V\,B^t})$ is a controllable pair and $(A, \sqrt{C^t\,R\,C})$ is an observable pair. Since $B\,V\,B^t$ and $C^t\,R\,C$ are symmetric it follows that (33) and (34) are controllability and observability Gramians according to Definitions 4 and 14. Due to Remark 10 and Theorem 17 they are positive definite as the Gramians of a controllable and an observable pair. Q. E. D.

The following lemma describes some features of matrix derivatives and traces.

LEMMA 38. *Let $A \in R^{n\times m}$, $B \in R^{m\times n}$. Then*

$$\mathrm{tr}\,(A\,B) = \mathrm{tr}\,(B\,A) \quad \textit{and} \quad \mathrm{tr}\,(A\,B) = \mathrm{tr}\,(A^t\,B^t).$$

Let $A \in R^{n\times m}$, $X \in R^{m\times k}$, $B \in R^{k\times n}$. *Then*

$$\frac{d}{dX}\text{tr}\,(A\,X\,B) = A^t\,B^t.$$

Let $A \in R^{n\times m}$, $X \in R^{k\times m}$, $B \in R^{k\times n}$. *Then*

$$\frac{d}{dX}\text{tr}\,(A\,X^t\,B) = B\,A.$$

Let $A \in R^{n\times m}, X \in R^{m\times n}$. *Then*

$$\frac{d}{dX}\text{tr}\,(A\,X) = \frac{d}{dX}\text{tr}\,(X\,A) = A^t.$$

Let $A \in R^{n\times n}$, $X \in R^{n\times m}$, $B \in R^{m\times m}$. *Then*

$$\frac{d}{dX}\text{tr}\,(A\,X\,B\,X^t) = A\,X\,B + A^t\,X\,B^t.$$

A proof is given in Appendix A.

The following is the main theorem in the optimal projection approach for solving the model-reduction problem.

THEOREM 39. *Suppose* (A_m, B_m, C_m) *solves the optimal model-reduction problem. Then there exist symmetric positive semidefinite matrices* $\hat{Q}$, $\hat{P} \in R^{n\times n}$ *such that for some* (G, M, Γ)*-factorization of* $\hat{Q}\hat{P}$, A_m, B_m *and* C_m *are given by*

$$A_m = \Gamma\,A\,G^t, \tag{37}$$

$$B_m = \Gamma\,B, \tag{38}$$

$$C_m = C\,G^t, \tag{39}$$

and such that, with $\tau \equiv G^t\,\Gamma$ *the following conditions are satisfied:*

$$\tau[A\,\hat{Q} + \hat{Q}\,A^t + B\,V\,B^t] = 0, \tag{40}$$

$$[A^t\,\hat{P} + \hat{P}\,A + C^t\,R\,C]\,\tau = 0, \tag{41}$$

$$\text{rank}\,(\hat{Q}) = \text{rank}\,(\hat{P}) = \text{rank}\,(\hat{Q}\,\hat{P}) = n_m. \tag{42}$$

Proof. The full order system (19)–(20) and the reduced order system (21)–(22) can be written as a single augmented system

$$\dot{\tilde{x}}(t) = \tilde{A}\,\tilde{x}(t) + \tilde{B}\,u(t),$$
$$\tilde{y}(t) = \tilde{C}\,\tilde{x}(t),$$

where

$$\tilde{x} = \begin{pmatrix} x \\ x_m \end{pmatrix}, \quad \tilde{y} = y - y_m,$$
$$\tilde{A} = \begin{pmatrix} A & 0 \\ 0 & A_m \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} B \\ B_m \end{pmatrix}, \quad \tilde{C} = \begin{pmatrix} C & -C_m \end{pmatrix}.$$

Using this notation the cost $J(A_m, B_m, C_m)$ can be written as

$$\begin{aligned} J(A_m, B_m, C_m) &= \lim_{t\to\infty} E\left[(y - y_m)^t\, R\,(y - y_m)\right] = \lim_{t\to\infty} E(\tilde{y}^t\, R\,\tilde{y}) \\ &= \lim_{t\to\infty} E(\tilde{x}^t\,\tilde{C}^t\, R\,\tilde{C}\,\tilde{x}) = \lim_{t\to\infty} E(\tilde{x}^t\,\tilde{R}\,\tilde{x}), \end{aligned}$$

where $\tilde{R} \equiv \tilde{C}^t\, R\,\tilde{C}$. Due to Theorem 25, when $t \to \infty$ the variance matrix of $\tilde{x}$ tends to $\tilde{Q}$ which is the unique solution of

$$\tilde{A}\,\tilde{Q} + \tilde{Q}\,\tilde{A}^t + \tilde{V} = 0, \tag{43}$$

where $\tilde{V} \equiv \tilde{B}\,V\,\tilde{B}^t$. Using this fact, Lemma 38, and the fact that the cost is a scalar, it follows

$$\begin{aligned} J(A_m, B_m, C_m) &= \text{tr}\,\left(\lim_{t\to\infty} E(\tilde{x}^t\,\tilde{R}\,\tilde{x})\right) = \lim_{t\to\infty} E\left[\text{tr}\,(\tilde{x}^t\,\tilde{R}\,\tilde{x})\right] \\ &= \lim_{t\to\infty} E\left[\text{tr}\,(\tilde{x}\,\tilde{x}^t\,\tilde{R})\right] = \text{tr}\,\left(\lim_{t\to\infty} E(\tilde{x}\,\tilde{x}^t)\tilde{R}\right) \\ &= \text{tr}\,(\tilde{Q}\,\tilde{R}). \end{aligned} \tag{44}$$

The goal of minimizing (44) under the constraint (43) can be achieved using the Lagrangian

$$L(A_m, B_m, C_m, \tilde{Q}, \tilde{P}, \lambda) \equiv \text{tr}\,\left[\lambda\,\tilde{Q}\,\tilde{R} + (\tilde{A}\,\tilde{Q} + \tilde{Q}\,\tilde{A}^t + \tilde{V})\tilde{P}\right], \tag{45}$$

with Lagrangian multipliers $\lambda \geq 0$ and $\tilde{P} \in R^{(n+n_m)\times(n+n_m)}$ that are not both zero, and $\tilde{P}$ is symmetric.

Using formulas from Lemma 38 to compute the partial derivatives of (45) with respect to $\tilde{Q}$ it follows that

$$\tilde{A}^t \tilde{P} + \tilde{P} \tilde{A} + \lambda \tilde{R} = 0. \tag{46}$$

Since $\tilde{A}$ is asymptotically stable, from Lemma 36 it follows that

$$\tilde{P} = \int_0^\infty e^{\tilde{A}^t t} \lambda \tilde{R} e^{\tilde{A} t} dt. \tag{47}$$

From (47) it follows that $\lambda = 0$ implies $\tilde{P} = 0$, hence without loss of generality set $\lambda = 1$, and (46) becomes

$$\tilde{A}^t \tilde{P} + \tilde{P} \tilde{A} + \tilde{R} = 0. \tag{48}$$

Let $\tilde{Q}$ and $\tilde{P}$ be partitioned

$$\tilde{Q} = \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix}, \quad \tilde{P} = \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix},$$

where $Q_1, P_1 \in R^{n\times n}$, $Q_2, P_2 \in R^{n_m \times n_m}$. The Lagrangian (45) has the form

$$\begin{aligned} L = \operatorname{tr} \Bigg[& \begin{pmatrix} A & 0 \\ 0 & A_m \end{pmatrix} \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} \\ & + \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} A^t & 0 \\ 0 & A_m^t \end{pmatrix} \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} + \begin{pmatrix} BVB^t & BVB_m^t \\ B_m V B^t & B_m V B_m^t \end{pmatrix} \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} \\ & + \lambda \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} C^t R C & -C^t R C_m \\ -C_m^t R C & C_m^t R C_m \end{pmatrix} \Bigg]. \end{aligned}$$

Using Lemma 38 the derivative $\partial L / \partial A_m$ can be computed as

$$\begin{aligned} & \frac{\partial}{\partial A_m} \operatorname{tr} \Bigg[\begin{pmatrix} A & 0 \\ 0 & A_m \end{pmatrix} \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} \\ & \qquad + \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} \begin{pmatrix} A^t & 0 \\ 0 & A_m^t \end{pmatrix} \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} \Bigg] \\ & = \frac{\partial}{\partial A_m} \operatorname{tr} \begin{pmatrix} A Q_1 P_1 + A Q_{12} P_{12}^t & A Q_1 P_{12} + A Q_{12} P_2 \\ A_m Q_{12}^t P_1 + A_m Q_2 P_{12}^t & A_m Q_{12}^t P_{12} + A_m Q_2 P_2 \end{pmatrix} \\ & \quad + \frac{\partial}{\partial A_m} \operatorname{tr} \begin{pmatrix} Q_1 A^t P_1 + Q_{12} A_m^t P_{12}^t & Q_1 A^t P_{12} + Q_{12} A_m^t P_2 \\ Q_{12}^t A^t P_1 + Q_2 A_m^t P_{12}^t & Q_{12}^t A^t P_{12} + Q_2 A_m^t P_2 \end{pmatrix} \\ & = \frac{\partial}{\partial A_m} \operatorname{tr} \begin{pmatrix} 0 & 0 \\ 0 & A_m Q_{12}^t P_{12} + A_m Q_2 P_2 \end{pmatrix} + \frac{\partial}{\partial A_m} \operatorname{tr} \begin{pmatrix} Q_{12} A_m^t P_{12}^t & 0 \\ 0 & Q_2 A_m^t P_2 \end{pmatrix} \\ & = (Q_{12}^t P_{12} + Q_2 P_2)^t + P_{12}^t Q_{12} + P_2 Q_2, \end{aligned}$$

the derivative $\partial L/\partial B_m$ can be computed as

$$\begin{aligned}
&\frac{\partial}{\partial B_m}\,\mathrm{tr}\left[\begin{pmatrix} B\,V\,B^t & B\,V\,B_m^t \\ B_m\,V\,B^t & B_m\,V\,B_m^t \end{pmatrix}\begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix}\right] \\
&= \frac{\partial}{\partial B_m}\,\mathrm{tr}\begin{pmatrix} B\,V\,B^t\,P_1 + B\,V\,B_m^t\,P_{12}^t & B\,V\,B^t\,P_{12} + B\,V\,B_m^t\,P_2 \\ B_m\,V\,B^t\,P_1 + B_m\,V\,B_m^t\,P_{12}^t & B_m\,V\,B^t\,P_{12} + B_m\,V\,B_m^t\,P_2 \end{pmatrix} \\
&= \frac{\partial}{\partial B_m}\,\mathrm{tr}\begin{pmatrix} B\,V\,B_m^t\,P_{12}^t & 0 \\ 0 & B_m\,V\,B^t\,P_{12} + B_m\,V\,B_m^t\,P_2 \end{pmatrix} \\
&= P_{12}^t\,B\,V + P_{12}^t\,B\,V + \frac{\partial}{\partial B_m}\,\mathrm{tr}\,(P_2\,B_m\,V\,B_m^t) \\
&= P_{12}^t\,B\,V + P_{12}^t\,B\,V + P_2\,B_m\,V + P_2\,B_m\,V,
\end{aligned}$$

and the derivative $\partial L/\partial C_m$ can be computed as

$$\begin{aligned}
&\frac{\partial}{\partial C_m}\,\mathrm{tr}\left[\lambda\begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix}\begin{pmatrix} C^t\,R\,C & -C^t\,R\,C_m \\ -C_m^t\,R\,C & C_m^t\,R\,C_m \end{pmatrix}\right] \\
&= \lambda\frac{\partial}{\partial C_m}\,\mathrm{tr}\begin{pmatrix} Q_1\,C^t\,R\,C - Q_{12}\,C_m^t\,R\,C & -Q_1\,C^t\,R\,C_m + Q_{12}\,C_m^t\,R\,C_m \\ Q_{12}^t\,C^t\,R\,C - Q_2\,C_m^t\,R\,C & -Q_{12}^t\,C^t\,R\,C_m + Q_2\,C_m^t\,R\,C_m \end{pmatrix} \\
&= \lambda\frac{\partial}{\partial C_m}\,\mathrm{tr}\begin{pmatrix} -Q_{12}\,C_m^t\,R\,C & 0 \\ 0 & -Q_{12}^t\,C^t\,R\,C_m + Q_2\,C_m^t\,R\,C_m \end{pmatrix} \\
&= \lambda(-R\,C\,Q_{12} - R\,C\,Q_{12}) + \lambda\frac{\partial}{\partial C_m}\,\mathrm{tr}\,(R\,C_m\,Q_2\,C_m^t) \\
&= \lambda(-R\,C\,Q_{12} - R\,C\,Q_{12} + R\,C_m\,Q_2 + R\,C_m\,Q_2).
\end{aligned}$$

Setting these derivatives to zero gives

$$Q_{12}^t\,P_{12} + Q_2\,P_2 = 0, \tag{49}$$

$$(P_{12}^t\,B + P_2\,B_m)V = 0, \tag{50}$$

$$R(C_m\,Q_2 - C\,Q_{12}) = 0. \tag{51}$$

Equations (43) and (48) can be written in the form

$$\begin{pmatrix} A & 0 \\ 0 & A_m \end{pmatrix}\begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix} + \begin{pmatrix} Q_1 & Q_{12} \\ Q_{12}^t & Q_2 \end{pmatrix}\begin{pmatrix} A^t & 0 \\ 0 & A_m^t \end{pmatrix} + \begin{pmatrix} B\,V\,B^t & B\,V\,B_m^t \\ B_m\,V\,B^t & B_m\,V\,B_m^t \end{pmatrix} = 0,$$

$$\begin{pmatrix} A^t & 0 \\ 0 & A_m^t \end{pmatrix}\begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix} + \begin{pmatrix} P_1 & P_{12} \\ P_{12}^t & P_2 \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & A_m \end{pmatrix} + \begin{pmatrix} C^t\,R\,C & -C^t\,R\,C_m \\ -C_m^t\,R\,C & C_m^t\,R\,C_m \end{pmatrix} = 0.$$

If they are expanded the following equations are obtained

$$A\,Q_1 + Q_1\,A^t + B\,V\,B^t = 0, \tag{52}$$

$$A\,Q_{12} + Q_{12}\,A_m^t + B\,V\,B_m^t = 0, \tag{53}$$

$$A_m\,Q_2 + Q_2\,A_m^t + B_m\,V\,B_m^t = 0, \tag{54}$$

$$A^t\,P_1 + P_1\,A + C^t\,R\,C = 0, \tag{55}$$

$$A^t\,P_{12} + P_{12}\,A_m - C^t\,R\,C_m = 0, \tag{56}$$

$$A_m^t\,P_2 + P_2\,A_m + C_m^t\,R\,C_m = 0. \tag{57}$$

Since A_m is asymptotically stable, (A_m, B_m) is a controllable pair and (A_m, C_m) is an observable pair, from Lemma 37 it follows that the solutions Q_2 and P_2 of equations (54) and (57) are positive definite matrices. Since Q_2 and P_2 are nonsingular, the following can be defined

$$G \equiv Q_2^{-1}\,Q_{12}^t, \tag{58}$$

$$\Gamma \equiv -P_2^{-1}\,P_{12}^t, \tag{59}$$

$$\hat{Q} \equiv Q_{12}\,Q_2^{-1}\,Q_{12}^t, \tag{60}$$

$$\hat{P} \equiv P_{12}\,P_2^{-1}\,P_{12}^t, \tag{61}$$

$$M \equiv Q_2\,P_2. \tag{62}$$

From (49) and (58)–(59) it follows that

$$\Gamma\,G^t = -P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1} = I_{n_m}$$

and (27) is satisfied.

From (50), (51), (58) and (59) it follows that

$$B_m = -P_2^{-1}\,P_{12}^t\,B = \Gamma\,B,$$

$$C_m = C\,Q_{12}\,Q_2^{-1} = C\,G^t$$

and (38)–(39) are satisfied.

From (49) and (58)–(62) it follows that

$$\hat{Q}\,\hat{P} = Q_{12}\,Q_2^{-1}\,Q_{12}^t\,P_{12}\,P_2^{-1}\,P_{12}^t = G^t\,M\,\Gamma = -Q_{12}\,P_{12}^t \tag{63}$$

and (26) is satisfied.

Since Q_2 and P_2 are nonsingular, from (49) it follows that Q_{12} and P_{12} have full column ranks. From (60)–(61) and (63), using Sylvester's inequality, it follows that the rank conditions (42) are satisfied.

Since

$$M = P_2^{-1/2}(P_2^{1/2}\,Q_2\,P_2^{1/2})P_2^{1/2}$$

and Q_2 and P_2 are symmetric positive definite it follows that M is positive semisimple.

Using (49) and (58)–(61) the following relations can be derived

$$\hat{Q}\,\Gamma^t = -Q_{12}\,Q_2^{-1}\,Q_{12}^t\,P_{12}\,P_2^{-1} = Q_{12}, \tag{64}$$

$$-\hat{P}\,G^t = -P_{12}\,P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1} = P_{12}, \tag{65}$$

$$\Gamma\,\hat{Q}\,\Gamma^t = P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1}\,Q_{12}^t\,P_{12}\,P_2^{-1} = Q_2, \tag{66}$$

$$G\,\hat{P}\,G^t = Q_2^{-1}\,Q_{12}^t\,P_{12}\,P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1} = P_2. \tag{67}$$

If (38), (39) and (64)–(67) are substituted into (53)–(54), (56)–(57) then

$$A\,\hat{Q}\,\Gamma^t + \hat{Q}\,\Gamma^t\,A_m^t + B\,V\,B^t\,\Gamma^t = 0, \tag{68}$$

$$A_m\,\Gamma\,\hat{Q}\,\Gamma^t + \Gamma\,\hat{Q}\,\Gamma^t\,A_m^t + \Gamma\,B\,V\,B^t\,\Gamma^t = 0, \tag{69}$$

$$A^t\,\hat{P}\,G^t + \hat{P}\,G^t\,A_m + C^t\,R\,C\,G^t = 0, \tag{70}$$

$$A_m^t\,G\,\hat{P}\,G^t + G\,\hat{P}\,G^t\,A_m + G\,C^t\,R\,C\,G^t = 0. \tag{71}$$

If equation (68) is premultiplied by Γ and subtracted from (69) then

$$A_m\,\Gamma\,\hat{Q}\,\Gamma^t + \Gamma\,\hat{Q}\,\Gamma^t\,A_m^t + \Gamma\,B\,V\,B^t\,\Gamma^t - \Gamma\,A\hat{Q}\,\Gamma^t - \Gamma\,\hat{Q}\,\Gamma^t\,A_m^t - \Gamma\,B\,V\,B^t\,\Gamma^t = 0,$$

which, using (64), (66) and (58) implies

$$A_m = \Gamma\,A\,\hat{Q}\,\Gamma^t(\Gamma\,\hat{Q}\,\Gamma^t)^{-1} = \Gamma\,A\,Q_{12}\,Q_2^{-1} = \Gamma\,A\,G^t,$$

and (37) is satisfied.

From (49) and (58)–(61) it follows

$$\begin{aligned}
\tau\,\hat{Q} &= G^t\,\Gamma\,\hat{Q} = -Q_{12}\,Q_2^{-1}\,P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1}\,Q_{12}^t \\
&= Q_{12}\,Q_2^{-1}\,P_2^{-1}\,P_2\,Q_2\,Q_2^{-1}\,Q_{12}^t = Q_{12}\,Q_2^{-1}\,Q_{12}^t \\
&= \hat{Q}, \\
\hat{P}\,\tau &= \hat{P}\,G^t\,\Gamma = -P_{12}\,P_2^{-1}\,P_{12}^t\,Q_{12}\,Q_2^{-1}\,P_2^{-1}\,P_{12}^t \\
&= P_{12}\,P_2^{-1}\,P_2\,Q_2\,Q_2^{-1}\,P_2^{-1}\,P_{12}^t = P_{12}\,P_2^{-1}\,P_{12}^t \\
&= \hat{P}.
\end{aligned}$$

Using these relations, if (68) and (70) are premultiplied, respectively, by Γ and G, (69) and (71) are obtained, hence (69) and (71) are redundant.

If (68) is transposed and premultiplied by G^t then

$$\begin{aligned}
&G^t\,\Gamma\,\hat{Q}\,A^t + G^t\,A_m\,\Gamma\,\hat{Q} + G^t\,\Gamma\,B\,V\,B^t \\
&\quad = \tau\,\hat{Q}\,A^t + G^t\,\Gamma\,A\,G^t\,\Gamma\,\hat{Q} + \tau\,B\,V\,B^t \\
&\quad = \tau\,\hat{Q}\,A^t + \tau\,A\,\hat{Q} + \tau\,B\,V\,B^t = 0,
\end{aligned}$$

and (40) is obtained. If (70) is postmultiplied by Γ then

$$\begin{aligned}
&A^t\,\hat{P}\,G^t\,\Gamma + \hat{P}\,G^t\,A_m\,\Gamma + C^t\,R\,C\,G^t\,\Gamma \\
&\quad = A^t\,\hat{P}\,\tau + \hat{P}\,G^t\,\Gamma\,A\,G^t\,\Gamma + C^t\,R\,C\,\tau \\
&\quad = A^t\,\hat{P}\,\tau + \hat{P}\,A\,\tau + C^t\,R\,C\,\tau = 0,
\end{aligned}$$

and (41) is obtained. Q. E. D.

Equations (40)–(41) are called *modified Lyapunov equations.* Solutions $\hat{Q}$, $\hat{P}$ of (40)–(42) are called *pseudogramians.*

4. PROBABILITY-ONE HOMOTOPY METHODS.

Homotopies are a traditional part of topology, and have found significant application in nonlinear functional analysis and differential geometry [34]. Homotopy methods are globally convergent, which distinguishes them from most iterative methods, which are only locally convergent. The general idea of homotopy methods is to make a continuous transformation from an initial problem, which can be solved trivially, to the target problem.

Following [32], the theoretical foundation of all probability-one globally convergent homotopy methods is given in the following differential geometry theorem:

DEFINITION 40. Let $U \subset R^m$ and $V \subset R^p$ be open sets, and let $\rho : U \times [0,1) \times V \to R^p$ be a C^2 map. ρ is said to be transversal to zero if the Jacobian matrix $D\rho$ has full rank on $\rho^{-1}(0)$.

THEOREM 41. *If $\rho(a,\lambda,x)$ is transversal to zero, then for almost all $a \in U$ the map*

$$\rho_a(\lambda,x) = \rho(a,\lambda,x)$$

is also transversal to zero; i.e, with probability one the Jacobian matrix $D\rho_a(\lambda,x)$ has full rank on $\rho_a^{-1}(0)$.

The recipe for constructing a globally convergent homotopy algorithm to solve the nonlinear system of equations

$$f(x) = 0,$$

where $f : R^p \to R^p$ is a C^2 map, is as follows: For an open set $U \subset R^m$ construct a C^2 homotopy map $\rho : U \times [0,1) \times R^p \to R^p$ such that

1) $\rho(a,\lambda,x)$ is transversal to zero,

2) $\rho_a(0,x) = \rho(a,0,x) = 0$ is trivial to solve and has a unique solution x_0,

3) $\rho_a(1,x) = f(x)$,

4) $\rho_a^{-1}(0)$ is bounded.

Then for almost all $a \in U$ there exists a zero curve γ of ρ_a, along which the Jacobian matrix $D\rho_a$ has rank p, emanating from $(0,x_0)$ and reaching a zero $\bar{x}$ of f at $\lambda = 1$. This zero curve γ does not intersect itself, is disjoint from any other zeros of ρ_a, and has finite

arc length in every compact subset of $[0,1) \times R^p$. Furthermore, if $Df(\bar{x})$ is nonsingular, then γ has finite arc length. The general idea of the algorithm is to follow the zero curve γ emanating from $(0, x_0)$ until a zero $\bar{x}$ of $f(x)$ is reached (at $\lambda = 1$).

The zero curve γ is tracked by the normal flow algorithm [32], a predictor-corrector scheme. In the predictor phase, the next point is produced using Hermite cubic interpolation. Starting at the predicted point, the corrector iteration involves computing (implicitly) the Moore-Penrose pseudo-inverse of the Jacobian matrix at each point. The most complex part of the homotopy algorithm is the computation of the tangent vectors to γ, which involves the computation of the kernel of the $p \times (p+1)$ Jacobian matrix $D\rho_a$. The kernel is found by computing a QR factorization of $D\rho_a$, and then using back substitution. This strategy is implemented in the mathematical software package HOMPACK [35], which was used for the curve tracking here.

Two different homotopy maps are used for solving the optimal projection equations. When the initial problem, $g(x; a) = 0$, can be solved, then the homotopy map is [33]

$$\rho_a(\lambda, x) = F(a, \lambda, x) \equiv \lambda f(x) + (1 - \lambda) g(x; a), \tag{72}$$

where $f(x) = 0$ is the final problem, and a is a parameter vector used in defining the function g.

When the initial problem is not solved exactly, i.e., $g(x_0; b) \neq 0$, then the map is a Newton homotopy [29]

$$\rho_a(\lambda, x) = F(b, \lambda, x) - (1 - \lambda) F(b, 0, x_0), \tag{73}$$

where $a = (b, x_0)$. For $\lambda = 0$, $\rho_a(0, x_0) = F(b, 0, x_0) - F(b, 0, x_0) = 0$, and for $\lambda = 1$, $\rho_a(1, x) = F(b, 1, x) = f(x) = 0$.

When the map (73) is used, the equations considered for $0 < \lambda < 1$ are not the optimal projection equations whenever $F(b, 0, x_0) = g(x_0; b) \neq 0$. Hence, a goal in constructing the initial system and the starting point may be to minimize $g(x_0; b)$.

5. CHOOSING AN INITIAL SYSTEM AND THE STARTING POINT.

5.1. Theorem about the initial system.

While with homotopy algorithms in general an initial problem can be chosen practically at random, this problem has some special limitations. The reason is that Theorem 39 provides necessary conditions on a solution only under certain assumptions. In other words, every intermediate problem solution satisfies these equations only if the system is asymptotically stable, controllable and observable. While the absence of these features does not automatically mean that the intermediate problem solution will not satisfy the equations, it is clearly better to define a homotopy path in such a way that each problem along it corresponds to an asymptotically stable, controllable and observable system. Existence of a solution to the reduced order problem follows from [30]. Theorem 42 defines a class of initial systems such that these conditions are satisfied.

THEOREM 42. *For the given system* (1)–(2), *let* $A = X \Lambda X^{-1}$, *with* Λ *diagonal. Define* $D = X \Omega X^{-1}$ *for any diagonal matrix* $\Omega = diag\,(\omega_1, \ldots, \omega_n)$, *such that all* ω_i, *for* $i = 1, 2, \ldots, n$, *are in the open left half plane. Then for almost all such* D *any convex combination* $(A(\alpha), B, C)$ *of the systems* (D, B, C) *and* (A, B, C) *will be asymptotically stable, controllable and observable.*

Proof. (Stability.) Since

$$A(\alpha) = \alpha A + (1-\alpha) D = \alpha X \Lambda X^{-1} + (1-\alpha) X \Omega X^{-1}$$
$$= X [\alpha \Lambda + (1-\alpha) \Omega] X^{-1} = X \Lambda(\alpha) X^{-1},$$

and $\Lambda(\alpha)$ is diagonal with all diagonal elements in the open left half plane for $\alpha \in [0, 1]$, the matrix $A(\alpha)$ is asymptotically stable for $\alpha \in [0, 1]$.

(Controllability.) Let $B = X \hat{B}$. Consider the controllability matrices in the coordinate system obtained by the change of coordinates defined by the matrix X. In that coordinate system the controllability matrix is

$$\hat{B}_c(\alpha) = (\, \hat{B} \quad \Lambda(\alpha) \hat{B} \quad \ldots \quad \Lambda^{n-1}(\alpha) \hat{B} \,),$$

for $\alpha \in [0,1]$. For almost all choices of Ω the eigenvalues of the diagonal matrix $\Lambda(\alpha)$ will be distinct for $\alpha \in [0,1)$. That rank $\hat{B}_c(\alpha) = n$ follows from the results in [4]. A direct argument follows. Let $\lambda_1, \ldots, \lambda_n$ be the (distinct) eigenvalues of $\Lambda(\alpha)$ for some α, and

$$\hat{B} = \begin{pmatrix} \hat{b}_{11} & \hat{b}_{12} & \cdots & \hat{b}_{1m} \\ \hat{b}_{21} & \hat{b}_{22} & \cdots & \hat{b}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{b}_{n1} & \hat{b}_{n2} & \cdots & \hat{b}_{nm} \end{pmatrix}.$$

Since the system is controllable for $\alpha = 1$, it follows that each row of $\hat{B}$ has at least one nonzero element, because otherwise $\hat{B}_c(1)$ would have a zero row.

Reorder the rows and columns of $\hat{B}_c$ in the following way: for each row of $\hat{B}$ that has a nonzero element in the first column (suppose there are p_1 of them), exchange rows so that the selected p_1 rows are at the top. Next, exchange columns in such a way that columns $1, m+1, \ldots, (p_1-1)m+1$ become the first p_1 columns. This produces a submatrix in the upper left corner which can be expressed as

$$\hat{B}_1 = \begin{pmatrix} \hat{b}_1 & 0 & \cdots & 0 \\ 0 & \hat{b}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{b}_{p_1} \end{pmatrix} \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{p_1-1} \\ 1 & \lambda_2 & \cdots & \lambda_2^{p_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{p_1} & \cdots & \lambda_{p_1}^{p_1-1} \end{pmatrix}.$$

Since $\hat{b}_1, \ldots, \hat{b}_{p_1}$ were chosen to be nonzero, and the λ_i are distinct, the Vandermonde matrix is nonsingular, and hence $\hat{B}_1$ is also nonsingular.

Repeating this procedure for the remaining rows gives a block upper triangular transformation of $\hat{B}_c(\alpha)$, with diagonal blocks $\hat{B}_1, \ldots, \hat{B}_r$, for some $r \leq n$. That means that rank $\hat{B}_c(\alpha) = n$ and the system is controllable.

(Observability.) The dual construction for the observability matrices proves that the system is observable. Q. E. D.

While the random construction of the matrix D given in Theorem 42 is theoretically plausible, in practice it may not be wise. The reason is that the matrix X is complex in general, which for many choices of Ω leads to a complex matrix D, which is undesirable. Hence, it is better to directly construct a matrix D such that Ω satisfies the conditions given in Theorem 42.

One simple choice for D is

$$D \equiv -c_1 I + \text{diag}\,\{\epsilon_1, \ldots, \epsilon_n\}, \tag{74}$$

where $c_1 > 0$ and ϵ_i are small random numbers that correspond to the parameter a in the theory. In this case Ω is a small perturbation of $-c_1 I$.

Also, the matrix D can be defined as

$$D \equiv -c_1 I + c_2 A, \tag{75}$$

for $c_1, c_2 > 0$. In this case $\Omega = -c_1 I + c_2 \Lambda$.

5.2. Choosing the starting point.

The starting point $x_0 = (Q_0, P_0)$ of the homotopy algorithm can be chosen using a number of different strategies.

One strategy is to choose Q_0 and P_0 that are positive semidefinite and satisfy the rank conditions, but are otherwise random. This approach may lead to relatively large values of $F(a, 0, x_0) = g(x_0; a)$.

The second strategy, which can be applied for any choice of the matrix D described in Section 5.1, generally leads to relatively small but nonzero $g(x_0)$. Since the matrix D is asymptotically stable, due to Lemma 24 the Lyapunov equation

$$D\,Q + Q\,D^t + B\,V\,B^t = 0 \tag{76}$$

has a unique solution Q. Let $Q = T\,\Sigma\,T^t$, where T is orthogonal and

$$\Sigma = \text{diag}\,\{\sigma_1, \ldots, \sigma_n\}.$$

Next, define

$$\Sigma_1 \equiv \text{diag}\,\{\sigma_1, \ldots, \sigma_{n_m}, 0, \ldots, 0\}, \qquad Q_0 \equiv T\,\Sigma_1\,T^t.$$

If Q_0 is substituted for Q in (76), the equation will not be satisfied, but in general, if σ_i are sufficiently small, it will not be very different from zero. A similar procedure can be applied to compute P_0 that will 'almost' satisfy the equation

$$D^t P + P\,D + C^t R\,C = 0.$$

The point $x_0 = (Q_0, P_0)$ chosen in this way may lead to small values of $g(x_0)$. Also, this x_0 can be used as the initial guess for a quasi-Newton algorithm which may find a solution to the initial problem

$$\tau[D\,Q + Q\,D^t + B\,V\,B^t] = 0, \tag{77}$$

$$[D^t\,P + P\,D + C^t\,R\,C]\tau = 0. \tag{78}$$

More details on the implementation of this strategy are given in Section 8.1, where the same idea is exploited with a different purpose.

The third strategy can be utilized for $n_m \geq \max\{m, l\}$ and a choice of D as in (74) or (75) with small c_2. If $D = -c\,I$ it follows from (77)–(78) that

$$Q = \frac{B\,V\,B^t}{2\,c}, \quad P = \frac{C^t\,R\,C}{2\,c}$$

are solutions with rank $Q = m$ and rank $P = l$. Let $B_1 \in R^{n \times n_m}$ be a full column rank matrix obtained by augmenting (if necessary) B, $C_1 \in R^{n_m \times n}$ a full row rank matrix obtained by augmenting (if necessary) C, and $V_1, R_1 \in R^{n_m \times n_m}$ full rank matrices obtained by augmenting (if necessary), respectively, V and R. If the numbers used for augmentation are relatively small, then

$$Q_0 \equiv \frac{B_1\,V_1\,B_1^t}{2\,c}, \quad P_0 \equiv \frac{C_1^t\,R_1\,C_1}{2\,c}$$

will be close to solutions of (77)–(78). Furthermore, since $x_0 = (Q_0, P_0)$ satisfies the first two rank conditions (42), and with appropriate augmentation of B_1 and C_1 the third rank condition can be satisfied, this point can be used as the initial guess for a quasi-Newton algorithm that can solve (77)–(78), (42), giving an initial point for the homotopy algorithm. The quasi-Newton algorithm is using equivalent forms of these equations given in Chapter 7, where the rank conditions do not appear explicitly, but are incorporated into the equations.

The fourth strategy can be applied for the same n_m and D as for the third strategy. Let

$$Q = \frac{B\,V\,B^t}{2\,c}, \quad P = \frac{C^t\,R\,C}{2\,c}$$

and $p = \min\{m, l\}$. If $m = l$ set $\tilde{Q}_0 \equiv Q$, $\tilde{P}_0 \equiv P$ and proceed, otherwise a small correction is necessary. Suppose $m > l$. Let $Q = T\,\Sigma\,T^t$, where T is orthogonal and

$$\Sigma = \text{diag}\,\{\sigma_1, \ldots, \sigma_m, 0, \ldots, 0\}.$$

Define

$$\Sigma_1 \equiv \text{diag}\,\{\sigma_1, \ldots, \sigma_l, 0, \ldots, 0\},$$

$\tilde{Q}_0 \equiv T\,\Sigma_1\,T^t$, $\tilde{P}_0 \equiv P$. $\tilde{Q}_0$ and $\tilde{P}_0$ thus obtained will satisfy the first two rank conditions (42) for $n_m = p = l$, but may not satisfy the third.

The next step is to apply a contragredient transformation explained in Section 3.2, if necessary in a modified form, to obtain decompositions

$$\tilde{Q}_0 = W \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} W^t, \quad \tilde{P}_0 = U^t \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} U,$$

with $W, U \in R^{n \times n}$, $\Sigma \in R^{p \times p}$, where $U\,W = I$ and diagonal Σ is positive definite. The computation of the spectral decomposition of the rank-p matrix $\tilde{P}_0$ can be done as in the original procedure, but the matrix $Q_1 \in R^{p \times p}$ of Theorem 26 may not have the full rank. In that case it is necessary to slightly perturb Q_1 so that it becomes a full rank matrix. After that, the procedure can proceed as explained in Theorem 26.

Let

$$\Sigma = \text{diag}\,\{\sigma_1, \ldots, \sigma_p, 0, \ldots, 0\}.$$

Define

$$\Sigma_1 \equiv \text{diag}\,\{\sigma_1, \ldots, \sigma_p, \sigma_{p+1}, \ldots, \sigma_{n_m}, 0, \ldots, 0\},$$

where $\sigma_{p+1}, \ldots, \sigma_{n_m}$ are small positive numbers. Let $Q_0 \equiv W\,\Sigma_1\,W^t$, $P_0 \equiv U^t\,\Sigma_1\,U$. Then, $x_0 = (Q_0, P_0)$ will lead to small values of $g(x_0)$. Also, x_0 can be used as the initial guess for a quasi-Newton algorithm for solving (77)–(78), (42).

The fifth strategy does not appear theoretically elegant, but it has given some very good results. The following is the motivation for this strategy. If the starting point x_0 of the homotopy and the final point $\bar{x}$ are very far apart in norm, it is obvious that the homotopy curve for the problem will be very long, hence it will require a large number of steps to

track. If x_0 and $\bar{x}$ are close there is no guarantee that the curve will be short, but intuitively it seems that should lead to a curve of smaller length. Hence, the goal is to 'guess' an initial point that will be close to the final point.

Points along the homotopy curve are generally (but not always) closer to the final point as λ grows. Hence, if a point along the curve is taken as x_0 for a new program run starting back at $\lambda = 0$, that starting point should be closer to $\bar{x}$. If that point is chosen for a value of λ closer to 1, it can be expected that x_0 will be closer to $\bar{x}$.

6. HOMOTOPY METHOD BASED ON THE DRAZIN INVERSE.

One approach in designing a homotopy algorithm for solving the optimal projection equations is to use the decomposition of the projection matrix τ based on the Drazin inverse.

6.1. Drazin inverse.

DEFINITION 43. The *index* of $A \in R^{n\times n}$ is the smallest nonnegative integer k such that im $(A^k) =$ im (A^{k+1}).

If $A \in R^{n\times n}$ has index k, then $R^n =$ im $(A^k) \oplus$ ker (A^k). Thus any $x \in R^n$ has the unique decomposition $x = u + v$, $u \in$ im (A^k), $v \in$ ker (A^k). Observe further that A is invertible on im (A^k).

DEFINITION 44. Let $A \in R^{n\times n}$ have index k, and $x = u+v$, $u \in$ im (A^k), $v \in$ ker (A^k). The *Drazin inverse* $A^\sharp$ of A is defined by $A^\sharp x = A_1^{-1}u$, where A_1 is A restricted to the image of A^k.

THEOREM 45. [5] *If $A \in R^{n\times n}$ has index* k, *then there exist nonsingular matrices T and C, and a nilpotent matrix N of index* k *such that*

$$A = T\begin{pmatrix} C & 0 \\ 0 & N \end{pmatrix} T^{-1} \qquad \textit{and} \qquad A^\sharp = T\begin{pmatrix} C^{-1} & 0 \\ 0 & 0 \end{pmatrix} T^{-1}.$$

Proof. Let $v_1, \ldots, v_r$ be a basis for im (A^k), and $v_{r+1}, \ldots, v_n$ a basis for ker (A^k). Then since $R^n =$ im $(A^k) \oplus$ ker (A^k), and these subspaces are invariant under A, the matrix of A with respect to $v_1, \ldots, v_n$ has the form

$$\begin{pmatrix} C & 0 \\ 0 & N \end{pmatrix},$$

where C is invertible and $N^k = 0$. Thus

$$A = P\begin{pmatrix} C & 0 \\ 0 & N \end{pmatrix} P^{-1}, \tag{79}$$

where $P = [v_1, \ldots, v_n]$. From the definition of $A^\sharp$ it follows that

$$A^\sharp = P\begin{pmatrix} C^{-1} & 0 \\ 0 & 0 \end{pmatrix} P^{-1}. \tag{80}$$

Q. E. D.

THEOREM 46. [5] *Let $A \in R^{n\times n}$ have index k. The Drazin inverse $A^{\sharp}$ of A is the unique matrix $A^{\sharp}$ such that:*

$$A^{\sharp} A A^{\sharp} = A^{\sharp}, \tag{81}$$

$$A^{\sharp} A = A A^{\sharp}, \tag{82}$$

$$A^{k+1} A^{\sharp} = A^{k}. \tag{83}$$

Proof. Using (79) and (80) it is straightforward to verify that A and $A^{\sharp}$ satisfy (81)–(83).

Conversely, suppose that both pairs A, $A^{\sharp}$ and A, X satisfy (81)–(83). Let

$$P^{-1} X P = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}.$$

From (83),

$$P \begin{pmatrix} C^{k+1} & 0 \\ 0 & 0 \end{pmatrix} P^{-1} X P = P \begin{pmatrix} C^{k} & 0 \\ 0 & 0 \end{pmatrix} P^{-1} P,$$

which implies $C^{k+1} X_{11} = C^{k}$, and $X_{11} = C^{-1}$. From (82), $A^{k} X = X A^{k}$ implies

$$\begin{pmatrix} C^{k} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} C^{-1} & X_{12} \\ X_{21} & X_{12} \end{pmatrix} = \begin{pmatrix} C^{-1} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} \begin{pmatrix} C^{k} & 0 \\ 0 & 0 \end{pmatrix},$$

hence $C^{k} X_{12} = X_{21} C^{k} = 0$ and $X_{12} = X_{21} = 0$. From (81) and (82), $A X^{i+1} = X^{i}$ for $i = 1, 2, \ldots$ implies

$$\begin{pmatrix} C & 0 \\ 0 & N \end{pmatrix} \begin{pmatrix} C^{-(i+1)} & 0 \\ 0 & X_{22}^{i+1} \end{pmatrix} = \begin{pmatrix} C^{-i} & 0 \\ 0 & X_{22}^{i} \end{pmatrix}.$$

It follows that $N X_{22}^{i+1} = X_{22}^{i}$ for $i = 1, 2, \ldots$, and $0 = N^{k} X_{22}^{k+1} = N^{k-1} X_{22}^{k} = \ldots = N X_{22}^{2} = X_{22}$. Hence

$$P^{-1} X P = \begin{pmatrix} C^{-1} & 0 \\ 0 & 0 \end{pmatrix},$$

which proves uniqueness of $A^{\sharp}$. Q. E. D.

$A^{\sharp}$ is computed using the Hermite echelon form as described in [5].

6.2. Derivative of Drazin inverse.

Hearon and Evans [14] give conditions for the differentiability of the Drazin inverse. That theorem is proven here in a simpler way and using different terminology.

THEOREM 47. *Let $A(t) \in C^k(I)$ for some interval I and*

$$\text{rank } (A(t)) = \text{rank } (A^2(t)) = r$$

for each $t \in I$. If, for each $t \in I$, $B(t)$ is the Drazin inverse of $A(t)$, then $B(t) \in C^k(I)$.

Proof. The rank condition implies A has either index 0 or 1. For index 0, $A^\sharp = A^{-1}$ is differentiable and there is nothing to prove. So assume A has index 1. Then for each $t \in I$, it has the Jordan decomposition

$$A = T \begin{pmatrix} A_1 & 0 \\ 0 & 0 \end{pmatrix} T^{-1},$$

where A_1 is square and nonsingular. Using the characteristic polynomial of A_1, a divisor of the characteristic polynomial of A, A_1^{-1} can be expressed as a polynomial in A_1 whose coefficients are polynomials in the elements of A:

$$A_1^{-1} = p(A_1).$$

Let

$$B_1 \equiv p(A) = T\, p\left(\begin{pmatrix} A_1 & 0 \\ 0 & 0 \end{pmatrix}\right) T^{-1} = T \begin{pmatrix} A_1^{-1} & 0 \\ 0 & p(0) \end{pmatrix} T^{-1}.$$

Then

$$B \equiv B_1\, A\, B_1 = T \begin{pmatrix} A_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} T^{-1} = A^\sharp$$

is the Drazin inverse of A. Furthermore, since the elements of B are polynomial functions of the elements of A, $B \in C^k(I)$. Q. E. D.

The derivative of the Drazin inverse is actually computed using the Sylvester equations. For a given A of index 1, let X be the unique matrix that satisfies:

$$A\,X\,A = A, \tag{84}$$

$$X\,A\,X = X, \tag{85}$$

$$A\,X = X\,A. \tag{86}$$

Differentiating (84)–(86) yields:

$$A' X A + A X' A + A X A' = A', \tag{87}$$

$$X' A X + X A' X + X A X' = X', \tag{88}$$

$$A X' - X' A = X A' - A' X. \tag{89}$$

Substituting $A X'$ from (89) into (87), and summing up equations (87)–(89) gives the Sylvester equation

$$\begin{aligned}(A + X A - I)X' + X'(A X - A + A^2)& \\ = X A' - A' X - X A' X + A' - X A' A - A X A',& \end{aligned} \tag{90}$$

which has a unique solution with probability one due to the randomness in $A(\lambda)$.

Solving (90) for X' completes the computation of the derivative of the Drazin inverse of the matrix A.

6.3. Computation of τ'.

From (29),

$$\begin{aligned}\text{rank } (\hat{Q}\,\hat{P})^2 &= \text{rank } (W_1\,\Sigma\,W_1^t\,U_1^t\Sigma\,U_1)^2 \\ &= \text{rank } (W_1\,\Sigma^4\,U_1) = \text{rank } (\Sigma^4) = \text{rank } (\Sigma^2) \\ &= \text{rank } (W_1\,\Sigma\,W_1^t\,U_1^t\,\Sigma\,U_1) = \text{rank } (\hat{Q}\,\hat{P}) = n_m < n,\end{aligned}$$

and therefore $\hat{Q}\hat{P}$ has index 1.

Since $\hat{Q}\,\hat{P} = W_1\,\Sigma^2\,U_1$, using Theorem 45, it follows $(\hat{Q}\,\hat{P})^\sharp = W_1\,\Sigma^{-2}\,U_1$. Using the definition of τ it implies

$$\tau = (\hat{Q}\,\hat{P})\,(\hat{Q}\,\hat{P})^\sharp. \tag{91}$$

From (91) it follows

$$\tau' = (\hat{Q}\,\hat{P})'(\hat{Q}\,\hat{P})^\sharp + (\hat{Q}\,\hat{P})\big[(\hat{Q}\,\hat{P})^\sharp\big]',$$

where everything is directly computable except $\big[(\hat{Q}\,\hat{P})^\sharp\big]'$. $\big[(\hat{Q}\,\hat{P})^\sharp\big]'$ can be computed using the procedure for computation of the derivative of the Drazin inverse described in Section 6.2.

6.4. Algorithm.

The following is a description of the algorithm. The algorithm is based on the normal flow algorithm for dense Jacobian matrices described in [35], slightly modified here to handle the rank requirements of the solution $(\hat{Q}, \hat{P})$.

The algorithm uses homotopy map (72) or (73), where $F(a, \lambda, x)$ is represented by two equations:

$$A(\lambda)\,\hat{Q} + \hat{Q}\,A^t(\lambda) + \tau\,B\,V\,B^t + B\,V\,B^t\,\tau^t - \tau\,B\,V\,B^t\,\tau^t = 0,$$

$$A^t(\lambda)\,\hat{P} + \hat{P}\,A(\lambda) + \tau^t\,C^t\,R\,C + C^t\,R\,C\,\tau - \tau^t\,C^t\,R\,C\,\tau = 0,$$

where $A(\lambda) = \lambda\,A + (1-\lambda)\,D$.

The algorithm starts at the point $(\lambda, x) = (0, x_0) = (0, \hat{Q}_0, \hat{P}_0)$ with some $a = x_0 = (\hat{Q}_0, \hat{P}_0)$ chosen as explained in Section 5.2. Then it follows the zero curve γ of the homotopy map until a point where $\lambda = 1$ is reached.

Since the equations are symmetric, only the upper right triangles are considered, i.e., q_{ij} and p_{ij} are computed only for $j \geq i$. Therefore, the number of variables is $2\,[n(n+1)/2] = n(n+1)$.

The mathematical software package HOMPACK requires that the user provide routines to evaluate $\rho_a(\lambda, x)$ and the Jacobian matrix $D\rho_a$ at each step. While the former is relatively simple, the latter involves considerable computational effort.

The Jacobian matrix consists of $n(n+1)+1$ derivative vectors, which correspond to the partial derivatives with respect to λ, q_{ij} and p_{ij}. The terms that do not include τ are simple to evaluate analytically. For example,

$$\left[\frac{\partial}{\partial q_{ij}}(A\,Q)\right]_{kl} = \frac{\partial}{\partial q_{ij}} \sum_{m=1}^{n} a_{km}\,q_{ml} = \delta_{jl}\,a_{ki}. \tag{92}$$

On the other hand, the components of τ' have to be evaluated numerically. Each evaluation involves solving a Sylvester equation (90). Fortunately, since for different q_{ij} and p_{ij} only the right hand side of the equation changes, the whole process can be done efficiently. In order to compute $\partial/\partial q_{11}$ all the computations related to (90) are done completely. For all subsequent partial derivative evaluations only the right hand side of (90) is evaluated

and submitted to the procedure that solves a Sylvester equation. The procedure used for that purpose [2] supports this approach very efficiently.

When a final solution to the equations (40)–(42) is obtained, the computation of (A_m, B_m, C_m) is completed by applying the formulas (37)–(39), where G and Γ are obtained as explained in Lemma 28.

In summary, the whole algorithm is:

1) Define D using formula (74) or (75).
2) Choose a starting point x_0 using one of the strategies explained in Section 5.2.
3) Set $\lambda := 0$, $x := x_0$.
4) Compute Drazin inverse $(\hat{Q}\,\hat{P})^\sharp$.
5) Compute $\tau = (\hat{Q}\,\hat{P})\,(\hat{Q}\,\hat{P})^\sharp$.
6) Evaluate $\rho_a(\lambda, x)$.
7) (Evaluate $D\rho_a(\lambda, x)$.) For λ and each q_{ij}, p_{ij} such that $j \geq i$, do Steps 8–11.
   8) Compute derivatives of terms that do not include τ using analytical formulas similar to (92).
   9) Compute $\left[(\hat{Q}\,\hat{P})^\sharp\right]'$ using equation (90).
   10) Complete computation of τ' as $\tau' := (\hat{Q}\,\hat{P})'(\hat{Q}\,\hat{P})^\sharp + (\hat{Q}\,\hat{P})\left[(\hat{Q}\,\hat{P})^\sharp\right]'$.
   11) Sum values obtained in steps 8 and 10 to the final value of the derivative vector.
12) Take a step along the curve and obtain $x_1 = (\hat{Q}_1, \hat{P}_1)$.
13) Compute a contragredient transformation (28) as if x_1 satisfied the rank conditions.
14) Use formulas (29) to compute $\bar{x}_1 = (\hat{Q}, \hat{P})$.
15) If $\lambda < 1$, then set $x := \bar{x}_1$, and go to Step 4.
16) Interpolate to get the solution $\bar{x}_1$ at $\lambda = 1$. Obtain G and Γ as explained in Lemma 28.
17) Compute the reduced order model using (37)–(39).

7. METHODS BASED ON DECOMPOSITIONS OF PSEUDOGRAMIANS.

Homotopy algorithms for solving optimal projection equations can be designed using decompositions of the pseudogramians based on contragredient transformations.

7.1. Description of methods.

The equations (40)–(41) can be considered in another, equivalent form. Let

$$\hat{Q} = W_1 \Sigma W_1^t, \quad \hat{P} = U_1^t \Sigma U_1$$

be a decomposition of the pseudogramians $\hat{Q}$ and $\hat{P}$. If (40) is premultiplied by U_1, and (41) is postmultiplied by W_1, the following two equations are obtained:

$$U_1 A W_1 \Sigma W_1^t + \Sigma W_1^t A^t + U_1 B V B^t = 0, \tag{93}$$

$$A^t U_1^t \Sigma + U_1^t \Sigma U_1 A W_1 + C^t R C W_1 = 0. \tag{94}$$

The third equation

$$U_1 W_1 - I = 0 \tag{95}$$

determines the relationship between W_1 and U_1.

The matrix equations (93)–(95) contain $2 n n_m + n_m^2$ scalar equations. On the other hand, the only natural unknowns in (93)–(95), W_1, U_1 and diagonal Σ, contain $2 n n_m + n_m$ variables. Hence, some additional techniques are necessary in order to make an exact match between the number of equations and the number of unknowns. Three approaches are discussed.

One approach is to consider Σ to be symmetric and all elements of Σ as unknowns. This is appropriate, since the equations (93)–(95) with a full symmetric Σ can be transformed into equations of the same form with a diagonal Σ by computing

$$\Sigma = T \bar{\Sigma} T^t, \quad \bar{W}_1 = W_1 T, \quad \bar{U}_1 = T^t U_1,$$

where $\bar{\Sigma}$ is diagonal and T is orthogonal.

The second approach is to consider the decomposition from the statement of Theorem 26, which leads to the equations

$$U_1 A W_1 \Sigma W_1^t + \Sigma W_1^t A^t + U_1 B V B^t = 0, \tag{96}$$

$$A^t U_1^t \Omega + U_1^t \Omega U_1 A W_1 + C^t R C W_1 = 0, \tag{97}$$

$$U_1 W_1 - I = 0, \tag{98}$$

which also have $2 n n_m + n_m^2$ scalar equations. In this case the number of unknowns in W_1, U_1 and symmetric Σ and Ω is $2 n n_m + n_m^2 + n_m$. An additional n_m equations can be obtained, for example, by requiring

$$\sigma_{ii} - \omega_{ii} = 0 \quad \text{for } i = 1, \ldots, n_m.$$

The third approach is to reduce the number of unknowns in (96)–(98). The number of unknowns can be reduced to $2 n n_m + n_m^2$ if the diagonal elements of Ω are actually the diagonal elements of Σ.

7.2. Algorithm.

The following is a description of the algorithm for the method determined by the equations (93)–(95). The algorithm is based on the normal flow algorithm for dense Jacobian matrices described in [35]. Depending on the relative size of $F(a, 0, x_0)$ the algorithm may be modified. If $F(a, 0, x_0)$ is relatively large, computational experience shows that it is desirable (but not theoretically necessary) to enforce the symmetry of Σ along the homotopy path. This is done by observing that a symmetrized Σ corresponds to *some* homotopy map that *could* have been chosen initially. In effect, x_0 is changed in the homotopy map at each step along the homotopy zero curve γ. Obviously, in that case the homotopy map (73) must be used.

The algorithm uses the homotopy map (72) or (73), where $F(a, \lambda, x)$ is represented by three equations:

$$U_1 A(\lambda) W_1 \Sigma W_1^t + \Sigma W_1^t A^t(\lambda) + U_1 B V B^t = 0, \tag{99}$$

$$A^t(\lambda) U_1^t \Sigma + U_1^t \Sigma U_1 A(\lambda) W_1 + C^t R C W_1 = 0, \tag{100}$$

$$U_1 W_1 - I = 0. \tag{101}$$

A detailed description of the algorithm for evaluation of the Jacobian matrix $D\rho_a(\lambda, x)$ is given in Appendix B. A program that implements this method is given in Appendix C.

In summary, the whole algorithm is:

1) Define D using formula (74) or (75).
2) Choose a starting point $x_0 = (Q_0, P_0)$ using one of strategies explained in Section 5.2. Compute $(W_1)_0$, $(U_1)_0$ and Σ_0 using a contragredient transformation.
3) Set $\lambda := 0$, $x := x_0 = ((W_1)_0, (U_1)_0, \Sigma_0)$.
4) Evaluate $\rho_a(\lambda, x)$ given by (72) or (73), and (99)–(101).
5) Evaluate $D\rho_a(\lambda, x)$.
6) Take a step along the curve and obtain $x_1 = (W_1, U_1, \Sigma)$, $\bar{\lambda}$.
7) Compute $\bar{x}_1 = (W_1, U_1, \bar{\Sigma}) = (W_1, U_1, (\Sigma + \Sigma^t)/2)$.
8) Change the homotopy to

$$F(a, \lambda, x) - (1 - \lambda)v = 0,$$

where $v = F(a, \bar{\lambda}, \bar{x}_1)/(1 - \bar{\lambda})$.

9) If $\bar{\lambda} < 1$, then set $x := \bar{x}_1$, $\lambda := \bar{\lambda}$, and go to Step 4.
10) If $\bar{\lambda} \geq 1$, compute the solution $\bar{x}_1$ at $\bar{\lambda} = 1$. Compute the reduced order model by diagonalizing $\Sigma = T\bar{\Sigma}T^t$.

Note: if $F(a, 0, x_0)$ is small, steps 7 and 8 can be omitted without a serious loss of efficiency.

8. QUASI-NEWTON TECHNIQUES.

Since for the optimal projection equations it is generally hard to make a good initial guess for a quasi-Newton algorithm, homotopy is apparently an unavoidable tool for solving the problem. However, in some cases it is possible to make an initial guess that results in the convergence of a quasi-Newton algorithm. This chapter discusses three such techniques for making an initial guess.

8.1. Using Gramian modifications.

The first technique exploits the same idea as in the second strategy for choosing the starting point, explained in Section 5.2. Since the matrix A is asymptotically stable, due to Lemma 24 the Lyapunov equation

$$A\,Q + Q\,A^t + B\,V\,B^t = 0 \tag{102}$$

has a unique solution Q. Let $Q = T\,\Sigma\,T^t$, where T is orthogonal and

$$\Sigma = \text{diag}\,\{\sigma_1, \ldots, \sigma_n\}.$$

Next, define

$$\Sigma_1 \equiv \text{diag}\,\{\sigma_1, \ldots, \sigma_{n_m}, 0, \ldots, 0\}, \qquad \tilde{Q} \equiv T\,\Sigma_1\,T^t.$$

If $\tilde{Q}$ is substituted for Q in (102), the equation will not be satisfied, but in general it will not be very different from zero. A similar procedure can be applied to compute $\tilde{P}$ that will 'almost' satisfy the equation

$$A^t\,P + P\,A + C^t\,R\,C = 0.$$

The point $\tilde{x} = (\tilde{Q}, \tilde{P})$ thus obtained will satisfy the first two rank conditions (42), but may not satisfy the third.

In order to construct a point that will satisfy all the rank conditions the algorithm for computing a contragredient transformation, given in the proof of Theorem 26, can be used in a modified form. The spectral decomposition of the matrix $\tilde{P}$ can be computed as

explained there, but the matrix Q_1, which is supposed to be of full rank, may not be. A small perturbation will make Q_1 a full rank matrix, and the rest of the procedure can be performed as explained in Theorem 26.

This modified procedure corresponds to a decomposition

$$Q_2 = W_2 \, \Sigma_2 \, W_2^t, \quad P_2 = U_2^t \, \Sigma_2 \, U_2,$$

where Q_2 and P_2 are small perturbations of, respectively, $\tilde{Q}$ and $\tilde{P}$. This constructed point $x_2 = (W_2, U_2, \Sigma_2)$ can be used as the initial guess for a quasi-Newton algorithm which may find a solution to the problem (40)–(42), expressed in the equivalent forms (93)–(95) or (96)–(98).

8.2. Using solutions of intermediate problems.

The second technique utilizes a solution to a system along the homotopy path that corresponds to some $\lambda < 1$. The point $x = (Q, P)$ which is a solution to that system may be a good initial guess for a quasi-Newton algorithm for solving the system for $\lambda = 1$. That point x satisfies all the rank conditions automatically, due to the way it is obtained.

This technique may be useful for computing large models of large systems, where real time considerations are a significant factor. Also, it may be useful for choices of initial problems and starting points that lead to slow convergence of the homotopy algorithms.

8.3. Using solutions of lower orders.

The third technique utilizes a solution that corresponds to a model of order smaller than n_m, to obtain a solution that corresponds to a model of order n_m. This will be illustrated on the method defined by the equations (93)–(95).

Suppose that a solution for model size $n_m - 1$ is obtained. Then the equations (93)–(95) are satisfied with $x_1 = (W_1, U_1, \Sigma_1)$, where $W_1 \in R^{n \times n_m - 1}$, $U_1 \in R^{n_m - 1 \times n}$, $\Sigma_1 \in R^{n_m - 1 \times n_m - 1}$. W_1 and U_1 can be augmented to $W_2 \in R^{n \times n_m}$ and $U_2 \in R^{n_m \times n}$ such that the matrix product $U_2 \, W_2$ is close to I. Let

$$\Sigma_1 = \text{diag} \, \{\sigma_1, \ldots, \sigma_{n_m - 1}\}.$$

Define

$$\Sigma_2 = \text{diag}\ \{\sigma_1, \ldots, \sigma_{n_m}\},$$

where σ_{n_m} is a small positive number. The point $x_2 = (W_2, U_2, \Sigma_2)$ thus obtained may be a good initial guess for a quasi-Newton algorithm.

A similar strategy can be applied starting from a solution that corresponds to a model of order lower that $n_m - 1$. In this case the chances for constructing a good initial guess are smaller.

This technique is useful since it is generally simpler to solve problems of lower orders, which have smaller Jacobian matrices, using the homotopy methods. It also suggests the possibility of a variable dimension homotopy method, which is not pursued further here.

9. NUMERICAL RESULTS.

This chapter contains results and observations obtained on a number of examples. Most of the examples are taken from journal papers. The first section contains models of different orders for 10 systems. The second section contains some program measures obtained from runs on a DECstation 3100. The third section compares results obtained using four proposed methods. The fourth section compares results obtained using different strategies for choosing the starting point. The fifth section gives an example of an efficient use of Strategy 5 for choosing the starting point. The sixth section compares models obtained using the optimal projection approach and models obtained using other approaches. The seventh section illustrates the importance of the choice of the initial system. The eighth section considers results obtained using the quasi-Newton techniques. The ninth section contains a summary of the chapter and some recommendations for future work.

9.1. Models.

This section gives models for 10 different systems. Most of the systems are obtained from the referenced papers. The rest of the systems are from Harris Corporation, Government Aerospace Systems Division. The results are obtained using the method defined by (93)–(95), using the homotopy (73) and applying Strategy 2 for choice of the starting point. The cost J is computed for each model as tr $(\tilde{Q}\,\tilde{R})$, according to (44). For all examples $V = R = I$.

EXAMPLE 1 [18]. The system is given by

$$A = \begin{pmatrix} -0.05 & -0.99 \\ -0.99 & -5000.0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 100 \end{pmatrix}, \quad C = (1 \quad 100).$$

With $D = -10000\,I$, for the starting point

$$x_0 = (W_1, U_1, \Sigma)_0 = \begin{pmatrix} 0.0099995 \\ 0.99995 \\ 0.0099995 \\ 0.99995 \\ 0.5 \end{pmatrix}$$

the homotopy algorithm converges to a solution corresponding to the model of order $n_m = 1$ given by

$$A_m = (-4998.078625), \quad B_m = (100.000194), \quad C_m = (100.000194).$$

This model yields the (minimum) cost $J = 96.078058$.

With $D = -100\,I$, for the starting point

$$x_0 = \begin{pmatrix} 1 \\ 0.25 \\ 1.07 \\ -0.27 \\ 1 \end{pmatrix}$$

the solution found corresponds to the model of order $n_m = 1$ given by

$$A_m = (-0.485152), \quad B_m = (-0.0000011427), \quad C_m = (-0.000000073400),$$

which yields the (maximum) cost $J = 10100.00$. This example shows that the homotopy method can obtain different solutions.

Due to the results from [17], the optimal projection equations can have at most $\binom{n}{n_m}$ solutions. Hence, in this case all the solutions were obtained. Obtaining all the solutions to the equations is the only known method that guarantees that one of the solutions will correspond to the global minimum.

EXAMPLE 2 [28]. The system is given by

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -10 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 70 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & -0.2 \end{pmatrix}.$$

A model of order $n_m = 1$ is

$$A_m = (-11.979443), \quad B_m = (-3.649276 \quad 0.442839), \quad C_m = (3.676048).$$

This model yields the cost $J = 0.598377$.

EXAMPLE 3 [18]. The system is given by

$$A = \begin{pmatrix} -0.25 & -0.4 \\ -0.4 & -0.72 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1.2 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1.2 \end{pmatrix}.$$

A model of order $n_m = 1$ is

$$A_m = (-0.838521), \quad B_m = (1.537575), \quad C_m = (1.537575).$$

This model yields the cost $J = 0.107256$.

EXAMPLE 4 [31]. The system is given by

$$A = \begin{pmatrix} -1 & 3 & 0 \\ -1 & -1 & 1 \\ 4 & -5 & -4 \end{pmatrix}, \quad B = \begin{pmatrix} -2 \\ 2 \\ 4 \end{pmatrix}, \quad C = (1 \quad 0 \quad 0).$$

A model of order $n_m = 1$ is

$$A_m = (-10.436530), \quad B_m = (-1.597203), \quad C_m = (1.597203).$$

This model yields the cost $J = 1.688216$.

A model of order $n_m = 2$ is

$$A_m = \begin{pmatrix} -3.600739 & 1.376721 \\ -1.376721 & -0.215037 \end{pmatrix}, \quad B_m = \begin{pmatrix} -1.711012 \\ 0.763496 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} 1.711012 \\ 0.763496 \end{pmatrix}.$$

This model yields the cost $J = 0.0197781$.

EXAMPLE 5 [28]. The system is given by

$$A = \begin{pmatrix} -10 & 1 & 0 \\ -5 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \quad C = (1 \quad 0 \quad 0).$$

A model of order $n_m = 1$ is

$$A_m = (-0.157898), \quad B_m = (0.423088), \quad C_m = (0.423088).$$

This model yields the cost $J = 0.0107792$.

A model of order $n_m = 2$ is

$$A_m = \begin{pmatrix} -0.448192 & -0.245890 \\ 0.245890 & -0.139652 \end{pmatrix}, \quad B_m = \begin{pmatrix} 0.301639 \\ 0.411520 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} -0.301639 \\ 0.411520 \end{pmatrix}.$$

This model yields the cost $J = 0.000329024$.

EXAMPLE 6. The system for this example is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & -0.02 & 1 & 0.01 \\ 0 & 0 & 0 & 1 \\ 0.1 & 0.001 & -0.1 & -0.001 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad C = (0 \quad 1 \quad 0 \quad 0).$$

A model of order $n_m = 2$ is

$$A_m = \begin{pmatrix} -0.0205177 & -1.431966 \\ 1.432245 & -0.00000309981 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -0.931218 & 0.477408 \\ 0.00704476 & 0.0107630 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} -1.046463 \\ 0.0128636 \end{pmatrix}.$$

This model yields the cost $J = 256.432$.

A model of order $n_m = 3$ is

$$A_m = \begin{pmatrix} -0.384322 & 0.0151428 & 0.000363187 \\ -0.0344809 & -0.0205145 & -1.431952 \\ 0.000412053 & 1.432223 & -0.00000322418 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -0.0450631 & -0.864146 \\ -0.931197 & 0.477361 \\ 0.00659947 & 0.0113390 \end{pmatrix}, \quad C_m = \begin{pmatrix} -0.865320 & -1.046423 & 0.0131197 \end{pmatrix}.$$

This model yields the cost $J = 255.703$.

EXAMPLE 7 [23], [36]. The system is given by

$$A = \begin{pmatrix} 0 & 0 & 0 & -150 \\ 1 & 0 & 0 & -245 \\ 0 & 1 & 0 & -1113 \\ 0 & 0 & 1 & -19 \end{pmatrix}, \quad B = \begin{pmatrix} 4 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}.$$

A model of order $n_m = 2$ is

$$A_m = \begin{pmatrix} -3.172419 & -1.170749 \\ 1.170749 & -0.437964 \end{pmatrix}, \quad B_m = \begin{pmatrix} -0.131078 \\ -0.118146 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} 0.131078 \\ -0.118146 \end{pmatrix}.$$

This model yields the cost $J = 4.15847 \cdot 10^{-7}$.

A model of order $n_m = 3$ is

$$A_m = \begin{pmatrix} -12.554152 & -2.839372 & -0.414932 \\ 2.839372 & -3.135361 & -1.168474 \\ -0.414932 & 1.168474 & -0.437810 \end{pmatrix},$$

$$B_m = \begin{pmatrix} 0.0564264 \\ 0.130701 \\ 0.118135 \end{pmatrix}, \quad C_m = \begin{pmatrix} 0.0564264 & -0.130701 & 0.118135 \end{pmatrix}.$$

This model yields the cost $J = 4.58560 \cdot 10^{-10}$.

EXAMPLE 8 [23]. The system is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -50 & -79 & -33 & -5 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad C = (50 \quad 15 \quad 1 \quad 0).$$

A model of order $n_m = 2$ is

$$A_m = \begin{pmatrix} -4.815122 & 1.508547 \\ -1.508547 & -0.532330 \end{pmatrix}, \quad B_m = \begin{pmatrix} 0.930361 \\ -0.779384 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} -0.930361 \\ -0.779384 \end{pmatrix}.$$

This model yields the cost $J = 0.026928$.

A model of order $n_m = 3$ is

$$A_m = \begin{pmatrix} -0.746729 & 4.843984 & 0.419342 \\ -4.843984 & -2.235622 & -1.454066 \\ 0.419342 & 1.454066 & -0.520312 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -0.358287 \\ 0.806973 \\ 0.773469 \end{pmatrix}, \quad C_m = (-0.358287 \quad -0.806973 \quad 0.773469).$$

This model yields the cost $J = 0.00148438$.

EXAMPLE 9 [15]. The system is given by

$$A = \begin{pmatrix} -6.2036 & 15.054 & -9.8726 & -376.58 & 251.32 & -162.24 & 66.827 \\ 0.53 & -2.0176 & 1.4363 & 0 & 0 & 0 & 0 \\ 16.846 & 25.079 & -43.555 & 0 & 0 & 0 & 0 \\ 377.4 & -89.449 & -162.83 & 57.998 & -65.514 & 68.579 & 157.57 \\ 0 & 0 & 0 & 107.25 & -118.05 & 0 & 0 \\ 0.36992 & -0.14445 & -0.26303 & -0.64719 & 0.49947 & -0.21133 & 0 \\ 0 & 0 & 0 & 0 & 0 & 376.99 & 0 \end{pmatrix},$$

$$B = \begin{pmatrix} 89.353 & 0 \\ 376.99 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.21133 \\ 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

A model of order $n_m = 3$ is

$$A_m = \begin{pmatrix} -0.0261157 & 9.349756 & -0.0528086 \\ -9.352068 & -0.912444 & 0.506220 \\ -0.0541716 & -0.506226 & -0.198770 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -2.414471 & -0.571953 \\ 14.906052 & 0.0416151 \\ -14.944459 & 0.0512237 \end{pmatrix}, \quad C_m = \begin{pmatrix} -0.371712 & 0.00240265 & 0.0122915 \\ 2.453290 & 14.906110 & 14.944542 \end{pmatrix}.$$

This model yields the cost $J = 0.673079$.

A model of order $n_m = 4$ is

$$A_m = \begin{pmatrix} -37.55440 & -0.0546940 & 0.326197 & -0.0709427 \\ 0.0561170 & -0.0261155 & 9.349755 & -0.0528084 \\ 0.324384 & -9.352067 & -0.912440 & 0.506220 \\ 0.0705453 & -0.0541714 & -0.506226 & -0.198769 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -2.666516 & -0.00624702 \\ -2.414464 & -0.571953 \\ 14.906036 & 0.0416151 \\ -14.944458 & 0.0512237 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} 0.280123 & -2.651769 \\ -0.371711 & 2.453283 \\ 0.00240433 & 14.906094 \\ 0.0122916 & 14.944541 \end{pmatrix}.$$

This model yields the cost $J = 3.27495 \cdot 10^{-7}$.

EXAMPLE 10. This is a state space model of the transfer function between a torque activator and an approximately collocated torsional rate sensor for the ACES structure [7], located at NASA Marshall Space Flight Center, Huntsville, AL. The system in this example is of size $n = 17$, $m = 1$, $l = 1$. The nonzero elements of A are

$A(1,1) = A(2,2) = -0.031978272$, $A(1,2) = -A(2,1) = -78.54$,

$A(1,17) = 0.0097138566$, $A(2,17) = -0.0060463517$,

$A(3,3) = A(4,4) = -5.152212$, $A(3,4) = -A(4,3) = -51.457677$,

$A(3,17) = -0.021760771$, $A(4,17) = 0.0054538246$.

$A(5,5) = A(6,6) = -0.1351159$, $A(5,6) = -A(6,5) = -15.417859$,

$A(5,17) = -0.02179972$, $A(6,17) = -0.015063913$,

$A(7,7) = A(8,8) = -0.42811443$, $A(7,8) = -A(8,7) = -14.698408$,

$A(7,17) = 0.01042631$, $A(8,17) = 0.0088479697$,

$A(9,9) = A(10,10) = -0.064896745$, $A(9,10) = -A(10,9) = -12.077045$,

$A(9,17) = -0.030531575$, $A(10,17) = -0.030260987$,

$A(11,11) = A(12,12) = -0.048520356$, $A(11,12) = -A(12,11) = -8.9654448$,

$A(11,17) = -0.016843335$, $A(12,17) = -0.011449591$,

$A(13,13) = A(14,14) = -0.036781718$, $A(13,14) = -A(14,13) = -4.9057426$,

$A(13,17) = -0.1248007$, $A(14,17) = 0.0005136047$,

$A(15,15) = A(16,16) = -0.025112482$, $A(15,16) = -A(16,15) = -3.8432892$,

$A(15,17) = -0.035415526$, $A(16,17) = -0.028115589$,

$A(17,17) = -92.399784$.

The matrices B and C are

$$B = \begin{pmatrix} 1.8631111 \\ -1.1413786 \\ -1.2105758 \\ 0.31424169 \\ 0.013307797 \\ -0.211128913 \\ 0.19552894 \\ -0.037391511 \\ -0.01049736 \\ -0.011486242 \\ -0.029376402 \\ 0.0082391613 \\ -0.012609562 \\ -0.0022040505 \\ -0.030853234 \\ 0.0011671662 \\ 0 \end{pmatrix}, \quad C^t = \begin{pmatrix} -0.0097138566 \\ 0.0060463517 \\ 0.021760771 \\ -0.0054538246 \\ 0.02179972 \\ 0.015063913 \\ -0.01042631 \\ -0.0088479697 \\ 0.030531575 \\ 0.030260987 \\ 0.016843335 \\ 0.011449591 \\ 0.1248007 \\ -0.0005136047 \\ 0.035415526 \\ 0.028115589 \\ 184.79957 \end{pmatrix}.$$

A model of order $n_m = 6$ is

$$A_m = \begin{pmatrix} -0.23442 & -52.59052 & -0.17250 & 0.06085 & 0.000005 & 0.019936 \\ 52.59052 & -10.53247 & -0.91009 & 0.54970 & 0.00004 & 0.13584 \\ 0.17250 & -0.91009 & -0.18410 & 15.43994 & 0.00001 & 0.046293 \\ 0.06085 & -0.54970 & -15.43994 & -0.03904 & -0.000006 & -0.02371 \\ 0.000005 & -0.00004 & -0.00001 & -0.000006 & -0.000000005 & -78.54024 \\ -0.019936 & 0.13584 & 0.046293 & 0.02371 & 78.54024 & -0.06402 \end{pmatrix},$$

$$B_m = \begin{pmatrix} 0.02447 \\ 0.16899 \\ 0.06006 \\ -0.027808 \\ -0.00004 \\ -0.15819 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} 0.02447 \\ -0.16899 \\ -0.06006 \\ -0.027808 \\ -0.00004 \\ 0.15819 \end{pmatrix}.$$

This model yields the cost $J = 4.19165 \cdot 10^{-5}$.

A model of order $n_m = 8$ is given by

$$A_m = \begin{pmatrix} -70.147 & 21.918 & -2.741 & 2.9917 & -0.3721 & 0.228 & 0.0246 & 0.083 \\ 54.161 & -32.186 & 4.683 & 9.2995 & -0.4958 & 0.180 & 0.0289 & 0.093 \\ 3.5118 & -4.6512 & -0.208 & -51.396 & 0.1211 & -0.013 & -0.0049 & -0.016 \\ -22.253 & 19.045 & 51.85 & -12.043 & 1.0945 & -0.639 & -0.0741 & -0.243 \\ 1.2271 & -1.1976 & -0.200 & 1.1602 & -0.1936 & 15.44 & 0.0243 & 0.0807 \\ 0.5249 & -0.5415 & -0.076 & 0.6934 & -15.450 & -0.014 & -0.0125 & -0.041 \\ -0.0705 & 0.0708 & 0.011 & -0.0770 & 0.0238 & 0.012 & 0.0181 & -78.57 \\ -0.2393 & 0.2397 & 0.036 & -0.2610 & 0.0803 & 0.042 & 78.508 & -0.082 \end{pmatrix},$$

$$B_m = \begin{pmatrix} -0.05753 \\ -0.06445 \\ 0.01043 \\ 0.16983 \\ -0.05959 \\ 0.02622 \\ 0.04591 \\ 0.15167 \end{pmatrix}, \quad C_m^t = \begin{pmatrix} -0.16432 \\ 0.16512 \\ 0.02442 \\ -0.18165 \\ 0.05966 \\ 0.02629 \\ -0.04472 \\ -0.15162 \end{pmatrix}.$$

This model yields the cost $J = 3.95223 \cdot 10^{-5}$.

9.2. Program measures.

This section contains some data about memory requirements and run times. All runs were performed on a DECstation 3100.

The most complex part of the algorithms is the computation of the tangent vectors to the curve γ, which involves the computation of the kernel of the Jacobian matrix $D\rho_a$. For large problems approximately 90% of the processor time required by the programs is devoted to that. For smaller problems, especially for smaller models, a lot of processor time is devoted to function evaluations and Jacobian computations that contain many matrix-matrix multiplications. Similarly, the Jacobian matrix takes most of the memory storage the programs require. Again, that is over 90% of all memory requirements for large problems.

Table 1 gives the size of the Jacobian matrices for different problem sizes. The table also contains a typical number of algorithm steps and Jacobian evaluations for each problem size. For different choices of parameters that determine an initial problem and the starting point those two numbers can vary significantly. The given data are for some good choices of those parameters. The last column contains the processor time needed for one tangent computation.

The largest problem listed in Table 1 required a total of approximately 77 CPU hours to solve, of which approximately 70 hours were devoted to tangent vector computations.

TABLE 1. PROGRAM MEASURES

n	n_m	Jacobian	Steps	Evaluations	Time (sec)
2	1	5	10	20	0.011
3	1	7	12	30	0.017
3	2	16	15	45	0.045
4	1	9	45	75	0.025
4	2	20	60	120	0.08
4	3	33	60	120	0.21
7	1	15	600	700	0.09
7	2	32	600	700	0.29
7	3	51	700	1000	0.71
7	4	72	800	1100	1.5
17	1	35	400	450	1.3
17	2	72	1500	1700	3.6
17	3	111	1000	1100	7.6
17	8	336	1700	2900	96

9.3. Comparison of different homotopy methods.

In this section the performance of four methods described in Sections 6 and 7 is compared. The four methods are numbered according to the following scheme: Method 1 is the method of Chapter 6, Method 2 is the method defined by (93)–(95), Method 3 is the method given by (96)–(98) that has $2\,n\,n_m + n_m^2 + n_m$ unknowns, and Method 4 is the method given by (96)–(98) that has $2\,n\,n_m + n_m^2$ unknowns. The methods were tested using the homotopy (73), choices of initial system $D = -10\,I$ or $D = -100\,I$, and using Strategy 2 for choosing the starting point.

Table 2 contains a comparison of the number of tangent computations required to solve different problems using these methods. The data are given for some some good choices of initial problems and starting points. Entries marked '–' mean that a solution has not been obtained within a reasonable number of steps or that the algorithm failed. The table represents the typical situation: Method 1 fails to give a result, Method 2 gives a result, while Methods 3 and 4 usually give results, but take more time than Method 2. This behavior is also seen for the other problems considered.

A possible reason for the better performance of Method 2 may be that it is the most 'natural' of the methods. All the methods are designed using some assumptions or

constructions that are not inherent in the problem. It seems that these unnatural assumptions are contained to the least extent in Method 2.

TABLE 2. NUMBER OF TANGENT COMPUTATIONS

Example	n_m	Method 1	Method 2	Method 3	Method 4
1	1	32	18	14	18
5	1	–	28	48	44
5	2	–	38	53	52
6	2	–	76	102	95
6	3	–	122	401	–
8	2	–	70	198	202
8	3	–	87	344	344

9.4. Comparison of different choices of starting point.

The choice of the starting point has a significant influence on the efficiency of the algorithm. Results for the four strategies explained in Chapter 5 are discussed here. All choices were tested in the context of Method 2, which has the best performance. Table 3 contains the number of tangent computations required to solve different problems using these strategies. According to Table 3, no strategy is clearly superior.

It is important to notice that, with the exception of the first strategy, the choice of the starting point depends on some program parameters which are used as 'small' numbers to augment matrices or make small perturbations. Hence, these results do not necessarily give the final judgement of these strategies, since there may be some choices of parameters that were not tested and which could improve the performance of the algorithm.

Since it is significantly different from the other strategies, Strategy 5 is discussed in a separate section.

TABLE 3. COMPARISON OF THE STRATEGIES FOR CHOOSING THE STARTING POINT

Example	n_m	Str. 1	Str. 2	Str. 3	Str. 4
1	1	43	14	18	18
5	1	58	44	28	44
5	2	56	51	38	113
6	2	76	138	175	279
6	3	122	127	131	–
8	2	109	92	70	80
8	3	116	89	87	78

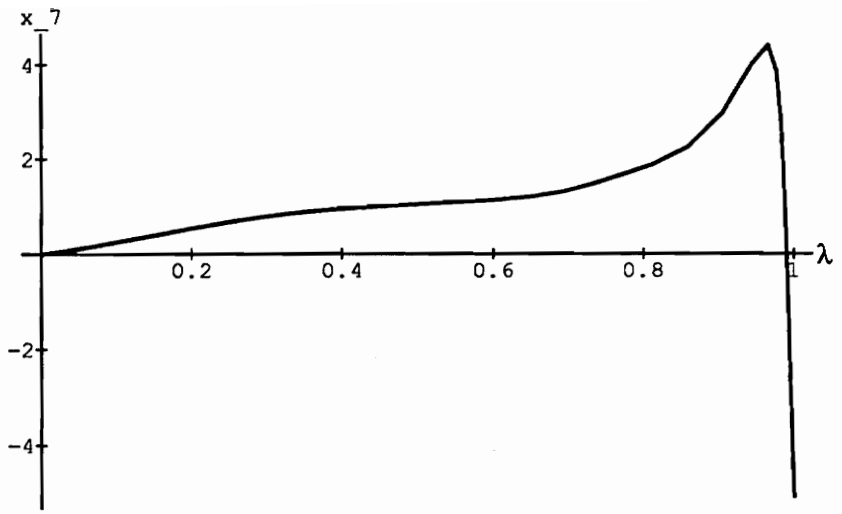


FIGURE 1. *Solution obtained using Strategy 2*

9.5. Choosing the starting point using Strategy 5.

If the curve tracking is performed using Strategy 5 for choosing the starting point, the result may be obtained in a smaller number of steps. This is illustrated on the problem of finding a model of order $n_m = 2$ for the system given in Example 9.

If the starting point is chosen by Strategy 2 and the initial system is $D = -100\,I$, then a solution is obtained in 833 steps, with 1141 tangent computations and a path arc length of 762. The trace of $x_7 = (W_1)_{4,1}$ is given in Figure 1.

If the solution is obtained by following the same path for the first 200 steps, then starting again from $\lambda = 0$ for another 200 steps, and the third time letting the algorithm finish, then the solution is obtained in a total of 735 steps, with 810 tangent computations and a path arc length of 737. The three traces of x_7 are given in Figure 2; note that the (G, M, Γ)-factorizations corresponding to Figures 1 and 2 are different (but the reduced order models are the same).

Typically Strategy 5 is more efficient than the others, due to smoother curves and convergence to different solutions, but it can be worse. Although ad hoc, Strategy 5 is clearly worth trying.

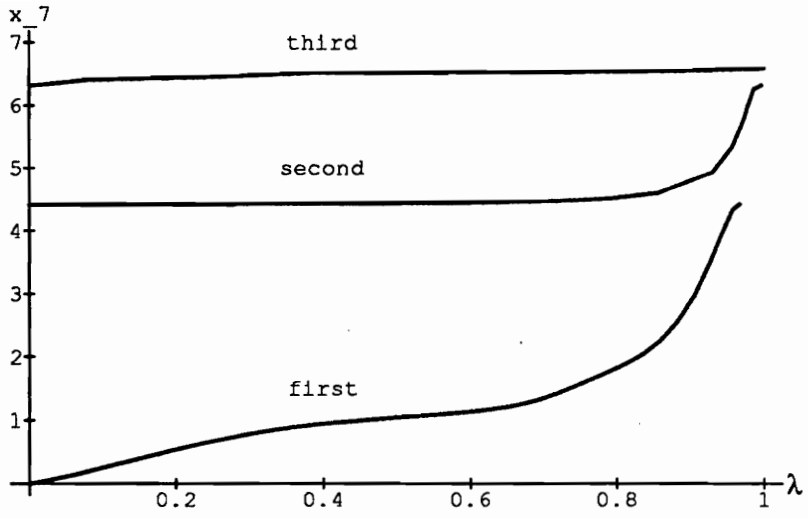


FIGURE 2. *Solution obtained using Strategy 5*

9.6. Comparison of optimal projection results and other methods.

As shown in Chapter 3, the optimal projection equations lead to a model which minimizes the cost function. Since the equations may have multiple solutions, there is no guarantee that a solution found will correspond to the minimal model. The only certain way to achieve that is to find all the solutions.

Numerical experiments have shown that using these methods it is relatively simple to obtain multiple solutions to the equations by changing the initial system or the starting point. In all cases where the referenced papers give the models, those models yield higher costs than the best models obtained using these methods. Table 4 gives a comparison of the costs, computed as tr $(\tilde{Q}\,\tilde{R})$, between the optimal projection models and the models given in the referenced papers.

The following methods are compared to the optimal projection method: component cost analysis (Examples 2 and 5), projection formulation (Example 4), balancing (Example 7) and nonminimal partial realization (Example 9).

TABLE 4. COST COMPARISON BETWEEN OPTIMAL PROJECTION AND OTHER METHODS

Example	n_m	Optimal Projection	Other
2	1	0.598377	0.697174
4	2	0.0197781	0.0469629
5	2	0.000329024	0.0340898
7	2	$4.15847 \cdot 10^{-7}$	$4.17767 \cdot 10^{-7}$
7	3	$4.58560 \cdot 10^{-10}$	$5.39446 \cdot 10^{-10}$
9	3	0.673079	1.003622

9.7. Choice of the initial system.

Although the methods work successfully with most choices (74) and (75) of the initial systems, this choice has a significant impact on the performance of the algorithm. Heuristically, it seems that the best choice is $D \equiv -c_1 I$, with c_1 of the same order of magnitude as the spectral radius of the matrix A.

The following example shows that the performance of the algorithm is strongly affected by the choice of the initial system. The system from Example 5 is considered and the model of order $n_m = 1$ is sought. The data are obtained using Method 2 and Strategy 1 for choosing the starting point.

With the initial system $D = -0.0003\, I$, 3100 steps are necessary to solve the problem. Figure 1 shows the behavior of the variable $x_7 = \Sigma_{11}$.

With the initial system $D = -10\, I$, 21 steps are sufficient. Figure 2 shows Σ_{11} for this case.

A possible reason for the poor performance in the first case may be that the initial system is close to being asymptotically unstable.

9.8. Quasi-Newton techniques.

The quasi-Newton techniques explained in Chapter 8 gave satisfactory results on certain problems, while other problems were not solved using these techniques.

The technique that exploits a rank-modification of the Gramian as the starting point for a quasi-Newton iteration is extremely efficient when it succeeds. The problems successfully

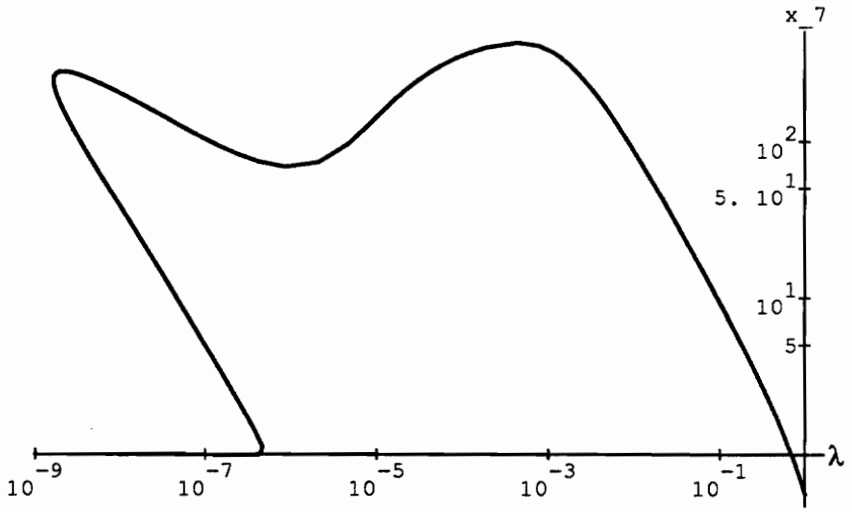


FIGURE 3. *Bad choice of the initial system*

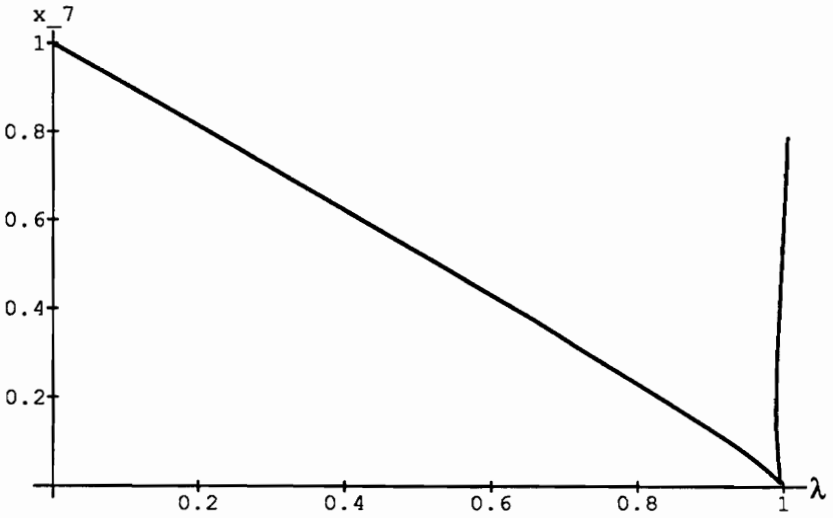


FIGURE 4. *Good choice of the initial system*

solved using this technique are the systems from Examples 1, 2, 4, 5; systems from Examples 6, 7 and 8 for $n_m = 1$; and the system from Example 9 for $n_m = 3$.

The technique that exploits the solution of a lower order is not so efficient since another (simpler) solution is required. Besides that, it was successful on a smaller number of problems: only the systems from Examples 4 and 5 were solved using this technique.

9.9. Summary.

The proposed methods are effective ways for solving the optimal projection equations and obtaining appropriate models. Solved models were given in Section 9.1. Some program measures were given in Section 9.2. A comparison of four methods defined in Chapters 6 and 7 was given in Section 9.3. Comparisons of different strategies for choosing the starting point were given in Sections 9.4 and 9.5. Results obtained using these methods and other methods from the referenced papers were compared in Section 9.6. The example in Section 9.7 showed that the algorithm performance can be significantly different with a different choice of the initial system. Some results obtained using the quasi-Newton techniques were briefly summarized in Section 9.8.

The results have shown that the best method is Method 2. A reason may be that in this method the smallest number of variables are forced to assume values that are not inherent (physically meaningful) to the problem, and the way it is done (full Σ, symmetrizing) is more natural than in the other methods. There is a large variety of submethods within four major methods. An example was given in Section 9.4, where it was shown that the same method with different strategies for determining the starting point gave significantly different results.

From this work it is not clear what the optimal strategy is for choosing the parameters c_1 and c_2 of (74) and (75). For different systems, different initial systems lead to efficient solutions. A general rule may be that the eigenvalues of the initial system should be of the same order of magnitude as the eigenvalues of the final system, but this does not always lead to the most efficient results. Further research should clarify this issue.

Another area for further research is the issue of finding a solution to the initial problem using a quasi-Newton method. Although it seems theoretically simple and clear that it should improve the performance, numerical experiments have shown that the initial system obtained in that way may not lead to a better algorithm. Actually, there are examples where initial systems obtained using the quasi-Newton method improved and examples where they corrupted the performance of the algorithm.

The issue of solving the initial system is related to the choice of the homotopy map. Similarly, although it seems obvious that the map (72) should perform better, in some cases that was not true. Numerical data on this issue are limited since solutions to the initial problem have been obtained for only a few small examples.

There are still many open questions and ways to improve the algorithms. It seems that the most important issues are finding better ways to solve the initial problem and improving the performance of the homotopy map (72).

10. CONCLUSION.

This work has considered the use of probability-one homotopy methods to solve the optimal projection equations for the model reduction problem. Theoretical results concerning the differentiability of the Drazin inverse are examined, and a practical method for computing the derivative is given.

Four different methods were given for solving the equations. The first method is based on solving the optimal projection equations in their original form. The three other methods stem from a decomposition of the pseudogramians based on a contragredient transformation and proved to be more numerically robust than the first method. The "best" method, the one that considers Σ to be a full matrix and forces it to be symmetric, was shown to be effective in finding the optimal reduced order models for a number of examples.

There are many choices in solving a problem: which method to use, how to define an initial system, how to choose a starting point, etc. This leads to a large number of combinations, each of them effectively representing a submethod. Although there is no guarantee that each submethod will solve each problem, and the number of steps cannot be determined in advance, the real tests have shown that many submethods give solutions in a reasonable number of steps. However, although all attempted problems were solved within a reasonable number of attempts, some more decisive strategy is needed to determine optimal algorithm parameters for given characteristics of a problem.

The three methods given in Chapter 7 exploit the same idea of transforming the equations using the decompositions of the pseudogramians, an idea which can be exploited in many other ways. Some other methods using this idea have been implemented, but the presented three gave the best results. However, it is possible that some other method based on this idea will give better results.

Two quite different ideas were explored in designing the algorithms. The methods explained in Chapter 7 performed well, while the method of Chapter 6 has not given satisfactory results, although it is possible that some, perhaps minor, modifications would make it perform well. However, the problem is still open for new ideas and approaches to solving it.

Since the algorithms require significant time for solving big systems, and most of that time is devoted to tangent computations, a parallel implementation of the methods may be considered.

REFERENCES.

[1] M. AOKI, "Control of large-scale dynamic systems by aggregation," *IEEE Trans. Autom. Contr.*, vol. AC-13, pp. 246–253, 1968.

[2] R. H. BARTELS AND G. W. STEWART, "Solution of the matrix equation $AX+XB=C$," *Comm. of the ACM*, vol. 15, pp. 820–826, 1972.

[3] D. S. BERNSTEIN AND D. C. HYLAND, "The optimal projection equations for reduced-order state estimation," *IEEE Trans. Autom. Contr.*, vol. AC-30, pp. 583–585, 1985.

[4] W. L. BROGAN, *Modern Control Theory*, Englewood Cliffs: Prentice Hall, 1985.

[5] S. L. CAMPBELL AND C. D. MEYER, JR., *Generalized Inverses of Linear Transformations*, London, England: Pitman, 1979, ch. 7, pp. 120–127.

[6] C. T. CHEN, *Introduction to Linear System Theory*, New York: Holt, Rinehart and Winston, Inc.,1970.

[7] E. G. COLLINS, JR., D. J. PHILLIPS AND D. C. HYLAND, "Robust decentralized control laws for the ACES structure," *Contr. Sys. Mag.*, pp. 62–70, April 1991.

[8] D. F. DELCHAMPS, *State Space and Input-Output Linear Systems*, New York: Springer-Verlag, 1988.

[9] F. R. GANTMACHER, *The Theory of Matrices*, New York: Chelsea Pub. Co., 1960.

[10] K. GLOVER, "All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds," *Int. J. Contr.*, vol. 39, pp. 1115–1193, 1984.

[11] W. M. HADDAD AND D. S. BERNSTEIN, "Combined L_2/H_∞ model reduction," *Int. J. Contr.*, vol. 49, pp. 1523–1535, 1989.

[12] J. K. HALE, *Ordinary Differential Equations*, New York: Wiley-Interscience, 1969., p.295

[13] P. R. HALMOS, *Measure Theory*, New York: Van Nostrand, 1950.

[14] J. Z. HEARON AND J. W. EVANS, "Differentiable generalized inverses," *Jour. Res. NBS*, vol. 72B, No. 2, pp. 109–113, 1968.

[15] J. HICKIN AND N. K. SINHA, "Model reduction for linear multivariable systems," *IEEE Trans. Autom. Contr.*, vol. AC-25, pp. 1121–1127, 1980.

[16] D. C. HYLAND AND D. S. BERNSTEIN, "The optimal projection equations for fixed-order dynamic compensation," *IEEE Trans. Autom. Contr.*, vol. AC-29, pp. 1034–1037, 1984.

[17] D. C. HYLAND AND D. S. BERNSTEIN, "The optimal projection equations for model reduction and the relationships among the methods of Wilson, Skelton, and Moore," *IEEE Trans. Autom. Contr.*, vol. AC-30, pp. 1201–1211, 1985.

[18] P. T. KABAMBA, "Balanced gains and their significance for L^2 model reduction," *IEEE Trans. Autom. Contr.*, vol. AC-30, pp. 690–693, 1985.

[19] S. KUNG AND D. W. LIN, "Optimal Hankel-norm model reductions: Multivariable systems," *IEEE Trans. Autom. Contr.*, vol. AC-26, pp. 832–852, 1981.

[20] S. KUNG AND D. W. LIN, "A state-space formulation for optimal Hankel-norm approximations," *IEEE Trans. Autom. Contr.*, vol. AC-26, pp. 942–946, 1981.

[21] H. KWAKERNAAK AND R. SIVAN, *Linear Optimal Control Systems*, New York: Wiley-Interscience, 1972.

[22] C. P. KWONG, "Optimal chained aggregation for reduced-order modelling," *Int. J. Contr.*, vol. 35, pp. 964–982, 1982.

[23] B. C. MOORE, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Trans. Autom. Contr.*, vol. AC-26, pp. 17–32, 1981.

[24] L. PERNEBO AND L. M. SILVERMAN, "Model reduction via balanced state space representations," *IEEE Trans. Autom. Contr.*, vol. AC-27, pp. 382–387, 1982.

[25] S. RICHTER AND E. G. COLLINS, JR., "A homotopy algorithm for reduced order compensator design using the optimal projection equations," in *proc. IEEE Conf. Dec. Contr.*,, Tampa, FL, pp. 506–511, Dec. 1989.

[26] R. E. SKELTON, "Cost decomposition of linear systems with application to model reduction," *Int. J. Contr.*, vol. 32, pp. 1031–1055, 1980.

[27] R. E. SKELTON AND P. C. HUGHES, "Modal cost analysis for linear matrix-second-order systems," *J. Dynamic Systems, Measurement and Control*, vol. 102, pp. 151–158, 1980.

[28] R. E. SKELTON AND A. YOUSUFF, "Component cost analysis of large scale systems," *Int. J. Contr.*, vol. 37, pp. 285–304, 1983.

[29] S. SMALE, "A convergent process of price adjustment and global Newton methods," *J. Math. Econom.*, vol. 3, pp. 107–120, 1976.

[30] J. T. SPANOS, M. H. MILMAN AND D. L. MINGORI, "Optimal model reduction and frequency-weighted extension," *AIAA paper number AIAA–90–3345–CP*, pp. 271–284, 1990.

[31] C. DE VILLEMAGNE AND R. E. SKELTON, "Model reductions using a projection formulation," *Int. J. Contr.*, vol. 46, pp. 2141–2169, 1987.

[32] L. T. WATSON, "Globally convergent homotopy methods: A tutorial," *Appl. Math. Comput.*, vol. 31BK, pp. 369–396, 1989.

[33] L. T. WATSON, "Globally convergent homotopy algorithms for nonlinear systems of equations," *Nonlinear Dynamics*, vol. 1, pp. 143–191, 1990.

[34] L. T. WATSON, "Numerical linear algebra aspects of globally convergent homotopy methods," *SIAM Rev.*, vol. 28, pp. 529–545, 1986.

[35] L. T. WATSON, S. C. BILLUPS AND A. P. MORGAN, "HOMPACK: A suite of codes for globally convergent homotopy algorithms," *ACM Trans. Math. Software*, vol. 13, pp. 281–310, 1987.

[36] D. A. WILSON, "Optimum solution of model-reduction problem," *Proc. IEE*, vol. 117, pp. 1161–1165, 1970.

[37] W. M. WONHAM, *Linear Multivariable Control: A Geometric Approach*, New York: Springer-Verlag, 1974.

Appendix A.

This appendix contains proofs of some features of the matrix trace and matrix derivatives.

LEMMA A.1. *Let* $A \in R^{n\times m}$, $B \in R^{m\times n}$. *Then*

$$\text{tr}\,(A\,B) = \text{tr}\,(B\,A) \quad \textit{and} \quad \text{tr}\,(A\,B) = \text{tr}\,(A^t\,B^t).$$

Proof. The first statement follows from

$$\begin{aligned}\text{tr}\,(A\,B) &= \sum_{i=1}^{n}(A\,B)_{ii} = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}\,b_{ji} \\ &= \sum_{j=1}^{m}\sum_{i=1}^{n} b_{ji}\,a_{ij} = \sum_{j=1}^{m}(B\,A)_{jj} \\ &= \text{tr}\,(B\,A),\end{aligned}$$

and the second from

$$\begin{aligned}\text{tr}\,(A\,B) &= \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}\,b_{ji} \\ &= \sum_{j=1}^{m}\sum_{i=1}^{n} a_{ij}\,b_{ji} = \sum_{j=1}^{m}(A^t\,B^t)_{jj} \\ &= \text{tr}\,(A^t\,B^t).\end{aligned}$$

Q. E. D.

LEMMA A.2. *Let* $X \in R^{n\times m}$ *and* $A \in R^{l\times l}$ *be differentiable with respect to* x_{ij} *for* $1 \le i \le n$, $1 \le j \le m$. *Then*

$$\frac{\partial}{\partial x_{ij}}\text{tr}\,(A) = \text{tr}\,(\frac{\partial}{\partial x_{ij}}A).$$

Proof. The statement follows from

$$\begin{aligned}\frac{\partial}{\partial x_{ij}}\text{tr}\,(A) &= \frac{\partial}{\partial x_{ij}}\sum_{i=1}^{l} a_{ii} = \sum_{i=1}^{l}\frac{\partial}{\partial x_{ij}}a_{ii} \\ &= \text{tr}\,(\frac{\partial}{\partial x_{ij}}A).\end{aligned}$$

Q. E. D.

LEMMA A.3. *Let* $X \in R^{p\times q}$ *and* $A \in R^{n\times m}$, $B \in R^{m\times l}$ *be differentiable with respect to* x_{ij} *for* $1 \le i \le p$, $1 \le j \le q$. *Then*

$$\frac{\partial}{\partial x_{ij}}(A\,B) = (\frac{\partial}{\partial x_{ij}}A)B + A(\frac{\partial}{\partial x_{ij}}B).$$

Proof. Since

$$\begin{aligned}\left[\frac{\partial}{\partial x_{ij}}(A\,B)\right]_{kl} &= \frac{\partial}{\partial x_{ij}}\left(\sum_{s=1}^{m} a_{ks}\,b_{sl}\right) = \sum_{s=1}^{m}\left(\frac{\partial}{\partial x_{ij}}(a_{ks}\,b_{sl})\right)\\ &= \sum_{s=1}^{m}\left[\left(\frac{\partial}{\partial x_{ij}}a_{ks}\right)b_{sl} + a_{ks}\left(\frac{\partial}{\partial x_{ij}}b_{sl}\right)\right]\\ &= \sum_{s=1}^{m}\left(\frac{\partial}{\partial x_{ij}}a_{ks}\right)b_{sl} + \sum_{s=1}^{m} a_{ks}\left(\frac{\partial}{\partial x_{ij}}b_{sl}\right)\\ &= \left[\left(\frac{\partial}{\partial x_{ij}}A\right)B\right]_{kl} + \left[A\left(\frac{\partial}{\partial x_{ij}}B\right)\right]_{kl}\end{aligned}$$

the statement follows. Q. E. D.

LEMMA A.4. *Let* $A \in R^{n\times m}$, $X \in R^{m\times k}$, $B \in R^{k\times n}$. *Then*

$$\frac{\partial}{\partial x_{ij}}\mathrm{tr}\,(A\,X\,B) = (A^t\,B^t)_{ij}.$$

Proof. Using Lemmas A.2 and A.3

$$\begin{aligned}\frac{\partial}{\partial x_{ij}}\mathrm{tr}\,(A\,X\,B) &= \mathrm{tr}\,\left(\frac{\partial}{\partial x_{ij}}(A\,X\,B)\right) = \mathrm{tr}\,\left(A\,\frac{\partial}{\partial x_{ij}}(X)B\right)\\ &= \mathrm{tr}\,(A\,\delta_{ij}\,B).\end{aligned}$$

Since $A\delta_{ij}$ contains all zeros except the j-th column which is the same as the i-th column of A, then

$$\mathrm{tr}\,(A\,\delta_{ij}\,B) = \sum_{s=1}^{n} a_{si}\,b_{js}$$

and the statement follows. Q. E. D.

LEMMA A.5. *Let* $A \in R^{n\times m}$, $X \in R^{k\times m}$, $B \in R^{k\times n}$. *Then*

$$\frac{\partial}{\partial x_{ij}}\text{tr}\,(A\,X^t\,B) = (B\,A)_{ij}.$$

Proof. Using Lemmas A.2 and A.3

$$\begin{aligned}\frac{\partial}{\partial x_{ij}}\text{tr}\,(A\,X^t\,B) &= \text{tr}\,\left(\frac{\partial}{\partial x_{ij}}(A\,X^t\,B)\right) = \text{tr}\,\left(A\,\frac{\partial}{\partial x_{ij}}(X^t)B\right)\\ &= \text{tr}\,(A\,\delta_{ji}\,B).\end{aligned}$$

Since $A\delta_{ji}$ contains all zeros except the i-th column which is the same as the j-th column of A, then

$$\text{tr}\,(A\,\delta_{ji}\,B) = \sum_{i=1}^{n} a_{sj}\,b_{is} = \sum_{s=1}^{n} b_{is}\,a_{sj}$$

and the statement follows. Q. E. D.

LEMMA A.6. *Let* $A \in R^{n\times m}$, $X \in R^{m\times n}$. *Then*

$$\frac{\partial}{\partial x_{ij}}\text{tr}\,(A\,X) = \frac{\partial}{\partial x_{ij}}\text{tr}\,(X\,A) = (A^t)_{ij}.$$

Proof. It follows directly from Lemma A.4. Q. E. D.

LEMMA A.7. *Let* $A \in R^{n\times n}$, $X \in R^{n\times m}$, $B \in R^{m\times m}$. *Then*

$$\frac{\partial}{\partial x_{ij}}\text{tr}\,(A\,X\,B\,X^t) = (A\,X\,B)_{ij} + (A^t\,X\,B^t)_{ij}.$$

Proof. Using Lemmas A.2 and A.3

$$\begin{aligned}\frac{\partial}{\partial x_{ij}}\text{tr}\,(A\,X\,B\,X^t) &= \text{tr}\,\left(A\,\frac{\partial}{\partial x_{ij}}(X)B\,X^t\right) + \text{tr}\,\left(A\,X\,B\frac{\partial}{\partial x_{ij}}X^t\right)\\ &= \text{tr}\,(A\,\delta_{ij}\,B\,X^t) + \text{tr}\,(A\,X\,B\delta_{ji}).\end{aligned}$$

Since $A\delta_{ij}$ contains all zeros except the j-th column which is the same as the i-th column of A, then

$$\text{tr}\,(A\,\delta_{ij}\,B\,X^t) + \text{tr}\,(A\,X\,B\,\delta_{ji}) = \sum_{s=1}^{n} a_{si}(B\,X^t)_{js} + (A\,X\,B)_{ij}$$

and the statement follows. Q. E. D.

DEFINITION A.8. Let $X \in R^{n\times m}$ and $A \in R^{l\times l}$ be differentiable with respect to x_{ij} for $1 \le i \le n$, $1 \le j \le m$. Then

$$\frac{d}{dX}\,\mathrm{tr}\,(A)$$

is the matrix that has

$$\frac{\partial}{\partial x_{ij}}\,\mathrm{tr}\,(A)$$

on the position (i,j) for $1 \le i \le n$, $1 \le j \le m$.

LEMMA A.9. *Let $A \in R^{n\times m}$, $X \in R^{m\times k}$, $B \in R^{k\times n}$. Then*

$$\frac{d}{dX}\mathrm{tr}\,(A\,X\,B) = A^t\,B^t.$$

Let $A \in R^{n\times m}$, $X \in R^{k\times m}$, $B \in R^{k\times n}$. Then

$$\frac{d}{dX}\mathrm{tr}\,(A\,X^t\,B) = B\,A.$$

Let $A \in R^{n\times m}$, $X \in R^{m\times n}$. Then

$$\frac{d}{dX}\mathrm{tr}\,(A\,X) = \frac{d}{dX}\mathrm{tr}\,(X\,A) = A^t.$$

Let $A \in R^{n\times n}$, $X \in R^{n\times m}$, $B \in R^{m\times m}$. Then

$$\frac{d}{dX}\mathrm{tr}\,(A\,X\,B\,X^t) = A\,X\,B + A^t\,X\,B^t.$$

Proof. It follows from Definition A.8 and Lemmas A.4–A.7. Q. E. D.

LEMMA A.10. *Let $X \in R^{n\times m}$, $A \in R^{m\times n}$. Then for $1 \le i \le n$, $1 \le j \le m$*

$$\frac{\partial}{\partial x_{ij}}(X\,A)$$

is the matrix whose i-th row is the same as the j-th row of A and all other elements are zero, and

$$\frac{\partial}{\partial x_{ij}}(A\,X)$$

is the matrix whose j-th column is the same as the i-th column of A and all other elements are zero.

Proof. Since

$$\left[\frac{\partial}{\partial x_{ij}}(X\,A)\right]_{kl} = \frac{\partial}{\partial x_{ij}}\sum_{s=1}^{m} x_{ks}\,a_{sl} = \delta_{ik}\,a_{jl}$$

and

$$\left[\frac{\partial}{\partial x_{ij}}(A\,X)\right]_{kl} = \frac{\partial}{\partial x_{ij}}\sum_{s=1}^{n} a_{ks}\,x_{sl} = \delta_{jl}\,a_{ki}$$

the statement follows. Q. E. D.

LEMMA A.11. *Let $X \in R^{n\times m}$, $B \in R^{n\times m}$. Then for $1 \le i \le n$, $1 \le j \le m$*

$$\frac{\partial}{\partial x_{ij}}(X\,B^t)$$

is the matrix whose i-th row is the same as the j-th column of B and all other elements are zero, and

$$\frac{\partial}{\partial x_{ij}}(B^t\,X)$$

is the matrix whose j-th column is the same as the i-th row of B and all other elements are zero.

Proof. The statement follows directly from Lemma A.10 by taking $B \equiv A^t$. Q. E. D.

LEMMA A.12. *Let $X \in R^{n\times m}$, $A \in R^{n\times m}$. Then for $1 \le i \le n$, $1 \le j \le m$*

$$\frac{\partial}{\partial x_{ij}}(X^t\,A)$$

is the matrix whose j-th row is the same as the i-th row of A and all other elements are zero, and

$$\frac{\partial}{\partial x_{ij}}(A\,X^t)$$

is the matrix whose i-th column is the same as the j-th column of A and all other elements are zero.

Proof. Since

$$\left[\frac{\partial}{\partial x_{ij}}(X^t\,A)\right]_{kl} = \frac{\partial}{\partial x_{ij}}\sum_{s=1}^{n} x_{sk}\,a_{sl} = \delta_{jk}\,a_{il}$$

and

$$\left[\frac{\partial}{\partial x_{ij}}(A\,X^t)\right]_{kl} = \frac{\partial}{\partial x_{ij}}\sum_{s=1}^{m} a_{ks}\,x_{ls} = \delta_{il}\,a_{kj}$$

the statement follows. Q. E. D.

Appendix B.

This appendix contains a derivation of the formulas for evaluation of the Jacobian matrix of the system

$$F_1(\lambda, W, U, \Sigma) \equiv U\, A(\lambda)\, W\, \Sigma\, W^t + \Sigma\, W^t\, A^t(\lambda) + U\, B = 0, \qquad (B.1)$$

$$F_2(\lambda, W, U, \Sigma) \equiv A^t(\lambda)\, U^t\, \Sigma + U^t\, \Sigma\, U\, A(\lambda)\, W + C\, W = 0, \qquad (B.2)$$

$$F_3(\lambda, W, U, \Sigma) \equiv U\, W - I = 0, \qquad (B.3)$$

where $A(\lambda) \equiv \lambda\, A + (1-\lambda)D$, $\lambda \in R^n$, $W \in R^{n \times n_m}$, $U \in R^{n_m \times n}$, $\Sigma \in R^{n_m \times n_m}$ are variables and $A, B, C, D \in R^{n \times n}$ are constants.

The Jacobian has $2\,n\,n_m + n_m^2$ rows and $2\,n\,n_m + n_m^2 + 1$ columns. The derivatives are computed in the matrix form and then transformed into vectors. The derivative of the sum of matrices is computed as the sum of the derivatives of the terms. The rows 1 thru $n\,n_m$ correspond to the equation (B.1), the rows $n\,n_m + 1$ thru $2\,n\,n_m$ correspond to the equation (B.2) and the rows $2\,n\,n_m + 1$ thru $2\,n\,n_m + n_m^2$ correspond to the equation (B.3). The first column corresponds to the derivatives with respect to λ, the columns 2 thru $n\,n_m + 1$ correspond to the derivatives with respect to W, the columns $n\,n_m + 2$ thru $2\,n\,n_m + 1$ correspond to the derivatives with respect to U and the columns $2\,n\,n_m + 2$ thru $2\,n\,n_m + n_m^2 + 1$ correspond to the derivatives with respect to Σ.

The derivative of $A(\lambda) = \lambda\, A + (1 - \lambda)\, D$ with respect to λ is $A - D$.

Lemmas A.3, A.10–A.12 are used in the derivation procedure.

B.1 Derivatives of F_1.

The derivative with respect to λ is $U(A - D)W\,\Sigma\,W^t + \Sigma\,W^t(A - D)^t$.

The derivative with respect to w_{ij} is the sum of the derivatives of $U\,A(\lambda)\,W\,\Sigma\,W^t$ and $\Sigma\,W^t\,A^t(\lambda)$. The derivative of the first term is the sum of the product of the matrix that contains all zeros except the j-th column which is the same as the i-th column of $U\,A(\lambda)$ and $\Sigma\,W^t$, and the matrix which contains all zeros except the i-th column which is the same as the j-th column of $U\,A(\lambda)\,W\,\Sigma$. The derivative of the second term is the product

of the matrix that contains all zeros except the i-th column which is the same as the j-th column of Σ and $A^t(\lambda)$.

The derivative with respect to u_{ij} is the matrix which contains all zeros except the i-th row which is the same as the j-th row of $A(\lambda)\,W\,\Sigma\,W^t$.

The derivative with respect to σ_{ij} is the sum of the derivatives of $U\,A(\lambda)\,W\,\Sigma\,W^t$ and $\Sigma\,W^t\,A^t(\lambda)$. The derivative of the first term is equal to the product of the matrix which contains all zeros except the j-th column which is the same as the i-th column of $U\,A(\lambda)\,W$ and W^t. The derivative of the second term is equal to the matrix which contains all zeros except the i-th row which is the same as the j-th row of $W^t\,A^t(\lambda)$.

B.2 Derivatives of F_2.

The derivative with respect to λ is $(A-D)^t U^t\,\Sigma + U^t\,\Sigma\,U(A-D)W$.

The derivative with respect to w_{ij} is the matrix which contains all zeros except the j-th column which is the same as the i-th column of $U^t\,\Sigma\,U\,A(\lambda)$.

The derivative with respect to u_{ij} is the sum of the derivatives of $A^t(\lambda)\,U^t\,\Sigma$ and $U^t\,\Sigma\,U\,A(\lambda)\,W$. The derivative of the first term is the product of the matrix that contains all zeros except the i-th column which is the same as the j-th column of $A^t(\lambda)$ and Σ. The derivative of the second term is the sum of the matrix which contains all zeros except the j-th row which is the same as the i-th row of $\Sigma\,U\,A(\lambda)\,W$, and the product of the matrix which contains all zeros except the j-th column which is the same as the i-th column of $U^t\,\Sigma$ and $A(\lambda)\,W$.

The derivative with respect to σ_{ij} is the sum of the derivatives of $A^t(\lambda)\,U^t\,\Sigma$ and $U^t\,\Sigma\,U\,A(\lambda)\,W$. The derivative of the first term is the matrix which contains all zeros except the j-th column which is the same as the i-th column of $A^t(\lambda)\,U^t$. The derivative of the second term is the product of the matrix which contains all zeros except the j-th column which is the same as the i-th row of U and $U\,A(\lambda)\,W$.

B.3 Derivatives of F_3.

The derivative with respect to λ is zero.

The derivative with respect to w_{ij} is the matrix which contains all zeros except the j-th column which is the same as the i-th column of U.

The derivative with respect to u_{ij} is the matrix which contains all zeros except the i-th row which is the same as the j-th row of W.

The derivative with respect to σ_{ij} is zero.

Appendix C.

```
      subroutine track(ld,n,nm,m,l,nn,inp,trace,lwarr,c1,c2,arcre,ansre,
     &                 a,b,c,v,r,y,iflag,ipivot,d,bvbt,ctrc,warr )
C
C Subroutine TRACK is the interface between the main program and
C TRACK1. Its purpose is to pass all parameters from the main program
C and organize working storage as needed for TRACK1.
C
C Input parameters:
C ld - leading dimension for all matrices except QR, at least n+1
C n,nm,m,l - dimensions of the system (A,B,C) and model (Am,Bm,Cm)
C nn - Jacobian dimension (2*n*nm + nm*nm)
C inp - strategy to be applied for choosing the starting point (1--5)
C   1 - random positive semidefinite (Q,P) = (W,U,Sigma), satisfying
C       rank conditions
C   2 - compute Gramians for A = D, find closest rank-nm approximations,
C       compute (W,U,Sigma) using contragredient transformation
C   3 - applies to the case nm >= max(m,l); augment (B,C,V,R) to rank-nm
C       matrices (B',C',V',R') and compute rank-nm matrices
C       Q = B' V' B'^t / 2 c1, P = C'^t R' C' / 2 c1;
C       from (Q,P) compute (W,U,Sigma) using contragredient transformation
C   4 - compute Q = B V B^t / 2 c1, P = C^t R C / 2 c1;
C       compute contragredient transformation (W,U,Sigma) for (Q,P) as
C       if rank=min(m,l); augment Sigma
C   5 - user provides Y = (W,U,Sigma)
C trace - logical i/o unit for output
C lwarr - length of warr
C c1,c2 - D is defined as D = -c1 * I + c2 * A
C arcre - relative error allowed of normal flow iterations along the curve
C ansre - relative error of the answer at lambda = 1
C a,b,c - system (A,B,C)
C v,r - white noise intensity V and weighting matrix R
C y - starting point (W,U,Sigma) if inp=5
C
C Output parameters:
C y - final point (lambda,W,U,Sigma)
C iflag - flag returned by FIXPNF;
C         except if length of warr is unsufficient, then iflag = 99
C
C Working storage (user provided):
C ipivot(nn+1) - integer working array
C d,bvbt,ctrc - D, B V B^t, C^t R C
C warr(32*ld*ld+11*(nn+1)+nn*(nn+2)+nn*(nn+13)/2)  - working array
C
      integer iflag,inp,ipivot(nn+1),l,ld,lwarr,m,n,n1,nm,nn,trace,z
      double precision a(ld,ld),ansre,arcre,b(ld,ld),bvbt(ld,ld)
      double precision c(ld,ld),c1,c2,ctrc(ld,ld),d(ld,ld)
      double precision r(ld,ld),v(ld,ld),warr(1),y(nn+1)
C
      z = ld * ld
      n1 = nn + 1
      if (lwarr .lt. 32*z+11*n1+nn*(nn+2)+nn*(nn+13)/2) then
        iflag = 99
        return
```

```
      endif
C
      call track1(ld,n,nm,m,l,nn,inp,trace,c1,c2,arcre,ansre,a,b,c,v,r,
     & y,iflag,ipivot,d,bvbt,ctrc,warr(1),
     & warr(z+1),warr(2*z+1),warr(3*z+1),warr(4*z+1),warr(5*z+1),
     & warr(6*z+1),warr(7*z+1),warr(8*z+1),warr(9*z+1),warr(1 0*z+1),
     & warr(11*z+1),warr(12*z+1),warr(13*z+1),warr(14*z+1),warr(15*z+1),
     & warr(16*z+1),warr(32*z+1),warr(32*z+n1+1),warr(32*z+2*n1+1),
     & warr(32*z+3*n1+1),warr(32*z+4*n1+1),warr(32*z+5*n1+1),
     & warr(32*z+6*n1+1),warr(32*z+7*n1+1),warr(32*z+8*n1+1),
     & warr(32*z+9*n1+1),warr(32*z+10*n1+1),warr(32*z+11*n1+1),
     & warr(32*z+11*n1+nn*(nn+2)+1))
      return
      end
      subroutine track1(ld,n,nm,m,l,nn,inp,trace,c1,c2,arcre,ansre,a,b,
     & c,v,r,y,iflag,ipivot,d,bvbt,ctrc, q,p,tau,ww,wi,sigma,f10,f20,f30,
     & temp1,temp2,temp3,aal,gc,go,work,p1,yold,yp,ypold,alpha,tz,w2,wp,
     & z0,z1,fvec,s1,qr,wa)
C
C Subroutine TRACK1 solves the optimal projection equations for the
C reduced order model problem. It is based on the normal flow algorithm
C for dense Jacobian matrices. The HOMPACK subroutine FIXPNF is used
C for that purpose.
C
C Input parameters:
C ld - leading dimension for all matrices except QR, at least n+1
C n,nm,m,l - dimensions of the system (A,B,C) and model (Am,Bm,Cm)
C nn - Jacobian dimension (2*n*nm + nm*nm)
C inp - strategy to be applied for choosing the starting point (1--5)
C trace - logical i/o unit for output
C c1,c2 - D is defined as D = -c1 * I + c2 * A
C arcre - relative error allowed of normal flow iterations along the curve
C ansre - relative error of the answer at lambda = 1
C a,b,c - system (A,B,C)
C v,r - white noise intensity V and weighting matrix R
C y - starting point (W,U,Sigma) if inp=5
C
C Output parameters:
C y - final point (lambda,W,U,Sigma)
C iflag - flag returned by FIXPNF
C
C Working storage:
C ipivot(nn+1) - integer working array
C d,bvbt,ctrc - D, B V B^t, C^t R C
C q,p,tau,ww,wi,sigma,f10,f20,f30,temp1,temp2,temp3,aal,gc,
C   go,work - working matrices of size (ld,ld)
C p1(16*ld*ld) - working array
C yold,yp,ypold - working arrays of size nn+1
C alpha,tz,w2,wp,z0,z1,fvec,s1 - working arrays of size nn+1
C qr(nn,nn+2) - working matrix
C wa(nn*(nn+13)/2) - working array
C
      integer hom,i,iflag,inp,ip(10),ipivot(nn+1),l,ld,m,n,nfe,nm,nn
      integer trace,type,z
      double precision a(ld,ld),aa,aal(ld,ld),alpha(nn+1),ansae,ansre
      double precision arclen,arcae,arcre,b(ld,ld),bvbt(ld,ld),c(ld,ld)
```

```
      double precision c1,c2,ctrc(ld,ld),d(ld,ld),f10(ld,ld),f20(ld,ld)
      double precision f30(ld,ld),fvec(nn+1),gc(ld,ld),go(ld,ld)
      double precision p(ld,ld),p1(16*ld*ld),q(ld,ld),qr(nn,nn+2)
      double precision r(ld,ld),s1(nn+1),sigma(ld,ld),sspar(8)
      double precision tau(ld,ld),temp1(ld,ld),temp2(ld,ld),temp3(ld,ld)
      double precision tz(nn+1),v(ld,ld),w2(nn+1),wa(nn*(nn+13)/2)
      double precision wi(ld,ld),work(ld,ld),wp(nn+1),ww(ld,ld),y(nn+1)
      double precision yold(nn+1),yp(nn+1),ypold(nn+1),z0(nn+1),z1(nn+1)
C
C Construct B V B^t and C^t R C, and D
C
      call bvrc(ld,n,m,l,b,c,bvbt,ctrc,v,r,temp1)
      call setd(ld,n,trace,c1,c2,a,d,temp1,temp2)
C
C Construct the starting point (W,U,Sigma)
C
      if ((nm .lt. m) .or. (nm .lt. 1)) inp = 1
      if (inp .eq. 1) call initp1(ld,n,hom,ww,wi,sigma,temp1,ipivot)
      if (inp .eq. 2)
     &  call initp2(ld,n,nm,nn,c1,c2,d,bvbt,ctrc,y,hom,q,p,tau,
     &              f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,aal,
     &              qr,gc,go,work,wa,fvec,s1)
      if (inp .eq. 3)
     &  call initp3(ld,n,nm,m,l,nn,c1,c2,d,bvbt,ctrc,b,c,v,r,y,hom,q,p,
     &              tau,f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,
     &              qr,wa,fvec,s1)
      if (inp .eq. 4)
     &  call initp4(ld,n,nm,m,l,nn,c1,c2,d,bvbt,ctrc,y,hom,q,p,tau,
     &              f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,
     &              qr,wa,fvec,s1)
      if (inp .eq. 5) call initp5(ld,n,nm,nn,hom,ww,wi,sigma,y)
C
C Transform point from matrix form (W,U,Sigma) to array Y
C
      call xwus(ld,n,nm,2,ww,wi,sigma,y(2))
      y(1) = 0.d0
C
C Compute F(0,Y(0))=(F10,F20,F30)
C
      call f0s(ld,n,nm,ww,wi,sigma,f10,f20,f30,d,bvbt,ctrc,temp1,
     &        temp2,temp3)
C
C Set parameters for FIXPNF
C
      arcae = arcre
      ansae = ansre
      do 301 i = 1,8
        sspar(i) = 0.d0
301   continue
      iflag = -2
      type = 1
C
C Prepare P1 and IP to be used in RHO and RHOJAC
C
      ip(1) = ld
      ip(2) = n
```

```
      ip(3) = nm
      ip(4) = type
      ip(5) = hom
      z = ld * ld
      call col2x(ld,ld,ld,a,p1(1))
      call col2x(ld,ld,ld,d,p1(z+1))
      call col2x(ld,ld,ld,bvbt,p1(2*z+1))
      call col2x(ld,ld,ld,ctrc,p1(3*z+1))
      call col2x(ld,ld,ld,f10,p1(4*z+1))
      call col2x(ld,ld,ld,f20,p1(5*z+1))
      call col2x(ld,ld,ld,f30,p1(6*z+1))
C
C Call tracking procedure
C
      do 311 i = 1,10
        call fixpnf(nn,y,iflag,arcre,arcae,ansre,ansae,trace,aa,nfe,
     &            arclen,yp,yold,ypold,qr,alpha,tz,ipivot,w2,wp,z0,z1,
     &            sspar,p1,ip)
        if (iflag .ne. 3) goto 401
311   continue
401   continue
      return
      end
      subroutine initp2(ld,n,nm,nn,c1,c2,d,bvbt,ctrc,y,hom,q,p,tau,
     &               f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,aal,
     &               gc,go,work,qr,wa,fvec,s1)
C
C Strategy 2 for choosing the starting point for homotopy
C
C Input parameters:
C ld - leading dimension of matrices
C n,nm - dimensions of system and model
C nn - Jacobian size
C c1,c2 - parameters used in definition of D
C d,bvbt,ctrc - D, B V B^t, C^t R C
C
C Output parameters:
C y - point in vector form
C hom - parameter indicating which homotopy map will be used, depends
C       on the result of quasi-Newton iteration
C
C Working storage:
C q,p,tau,f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,aal,gc,go,work,qr
C   - working matrices
C wa,fvec,s1 - working arrays
C
      integer hom,i,ind,job,ld,n,nm,nn,z
      double precision aal(ld,ld),bvbt(ld,ld),c1,c2,ctrc(ld,ld),d(ld,ld)
      double precision eps,f10(ld,ld),f20(ld,ld),f30(ld,ld),fvec(nn)
      double precision gc(ld,ld),go(ld,ld),p(ld,ld),q(ld,ld),qr(nn,nn)
      double precision s1(ld),sigma(ld,ld),tau(ld,ld),temp1(ld,ld)
      double precision temp2(ld,ld),temp3(ld,ld),wa(nn*(nn+13)/2)
      double precision wi(ld,ld),work(ld,ld),ww(ld,ld),y(nn)
C
      job = 11
      z = ld * ld
```

```
      eps = 1.d-16
C
C Compute controllability Gramian Gc
C
      call trans(ld,d,temp2,n,n)
      call scmat(ld,-1.d0,bvbt,temp1,n,n)
      call atxpxa(temp2,aal,temp1,n,ld,ld,ld,eps,ind)
      call scmat(ld,1.d0,temp1,gc,n,n)
C
C Compute Q, the Gc rank-nm approximation
C
      call dsvdc(temp1,ld,n,n,s1,aal,temp2,ld,temp3,ld,work,job,info)
      call setmat(ld,0.d0,sigma,n,n)
      do 321 i = 1,nm
        sigma(i,i) = s1(i)
321   continue
      call mulmat(ld,temp2,sigma,temp1,n,n,n,1)
      call mulmat(ld,temp1,temp3,q,n,n,n,3)
C
C Compute observability Gramian Go
C
      call scmat(ld,1.d0,d,temp2,n,n)
      call scmat(ld,-1.d0,ctrc,temp1,n,n)
      call atxpxa(temp2,aal,temp1,n,ld,ld,ld,eps,ind)
      call scmat(ld,1.d0,temp1,go,n,n)
C
C Compute P, the Go rank-nm approximation
C
      call dsvdc(temp1,ld,n,n,s1,aal,temp2,ld,temp3,ld,work,job,info)
      call setmat(ld,0.d0,sigma,n,n)
      do 322 i = 1,nm
        sigma(i,i) = s1(i)
322   continue
      call mulmat(ld,temp2,sigma,temp1,n,n,n,1)
      call mulmat(ld,temp1,temp3,p,n,n,n,3)
C
C Compute (W,U,Sigma) for given (Q,P)
C
      call cg(ld,n,nm,q,p,ww,wi,sigma,tau,temp1,temp2,temp3,
     &       wa(1),wa(z+1),wa(2*z+1),wa(3*z+1),wa(4*z+1),wa(5*z+1),
     &       wa(6*z+1),wa(7*z+1),wa(8*z+1),wa(9*z+1),wa(10*z+1),
     &       wa(11*z+1))
C
C Quasi-Newton iteration
C
C      call apprd(ld,n,nm,nn,d,bvbt,ctrc,y,hom,ww,wi,sigma,f10,f20,f30,
C     &          temp1,temp2,temp3,qr,wa,fvec,s1)
      return
      end
      subroutine initp3(ld,n,nm,m,l,nn,c1,c2,d,bvbt,ctrc,b,c,v,r,y,hom,
     &                q,p,tau,f10,f20,f30,ww,wi,sigma,temp1,temp2,
     &                temp3,qr,wa,fvec,s1)
C
C Strategy 3 for choosing the starting point for homotopy
C
C Input parameters:
```

```
C ld - leading dimension of matrices
C n,nm,m,l - dimensions of system and model
C nn - Jacobian size
C c1,c2 - parameters used in definition of D
C d,bvbt,ctrc,b,c,v,r - D, B V B^t, C^t R C, B, C, V, R
C
C Output parameters:
C y - point in vector form
C hom - parameter indicating which homotopy map will be used, depends
C       on the result of quasi-Newton iteration
C
C Working storage:
C q,p,tau,f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,qr  - working matrices
C wa,fvec,s1 - working arrays
C
      integer hom,i,l,ld,m,n,nm,nn,z
      double precision b(ld,ld),bvbt(ld,ld),c(ld,ld),c1,c2,ctrc(ld,ld)
      double precision d(ld,ld),f10(ld,ld),f20(ld,ld),f30(ld,ld)
      double precision fvec(nn),p(ld,ld),q(ld,ld),qr(nn,nn),r(ld,ld)
      double precision s1(ld),sigma(ld,ld),small,t1,t2,tau(ld,ld)
      double precision temp1(ld,ld),temp2(ld,ld),temp3(ld,ld),v(ld,ld)
      double precision wa(nn*(nn+13)/2),wi(ld,ld),ww(ld,ld),y(nn)
C
      z = ld * ld
      small = 1.d-3
C
C Augment B to TEMP1
C
      call setmat(ld,0.d0,temp1,n,nm)
      call scmat(ld,1.d0,b,temp1,n,m)
      t1 = 1.d-1
      t2 = 1.d-1 / dble(n*n)
      do 311 i = 1,nm
        if (dabs(temp1(i,i)) .lt. t1) temp1(i,i) = t1
        t1 = t1 + t2
311   continue
C
C Compute augmented (rank-nm) product B V B^t
C
      call setmat(ld,0.d0,temp2,nm,nm)
      call scmat(ld,1.d0,v,temp2,m,m)
      if (nm .gt. m) then
        do 301 i = m+1,nm
          temp2(i,i) = 1.d0
301     continue
      endif
      call mulmat(ld,temp1,temp2,temp3,n,nm,nm,1)
      call mulmat(ld,temp3,temp1,temp2,n,n,nm,3)
      t1 = 1.d0 / (2.d0 * c1)
      call scmat(ld,t1,temp2,q,n,n)
C
C Augment C to TEMP1
C
      call setmat(ld,0.d0,temp1,nm,n)
      call scmat(ld,1.d0,c,temp1,l,n)
      t1 = 1.d-1
```

```
      t2 = 1.d-1 / dble(n*n)
      do 313 i = 1,nm
        if (dabs(temp1(i,i)) .lt. t1) temp1(i,i) = t1
        t1 = t1 + t2
313   continue
C
C Compute augmented (rank-nm) product C^t R C
C
      call setmat(ld,0.d0,temp2,nm,nm)
      call scmat(ld,1.d0,r,temp2,l,l)
      if (nm .gt. l) then
        do 302 i = l+1,nm
          temp2(i,i) = 1.d0
302     continue
      endif
      call mulmat(ld,temp1,temp2,temp3,n,nm,nm,2)
      call mulmat(ld,temp3,temp1,temp2,n,n,nm,1)
      t1 = 1.d0 / (2.d0 * c1)
      call scmat(ld,t1,temp2,p,n,n)
C
C Compute (W,U,Sigma) for given (Q,P)
C
      call cg(ld,n,nm,q,p,ww,wi,sigma,tau,temp1,temp2,temp3,
     &       wa(1),wa(z+1),wa(2*z+1),wa(3*z+1),wa(4*z+1),wa(5*z+1),
     &       wa(6*z+1),wa(7*z+1),wa(8*z+1),wa(9*z+1),wa(10*z+1),
     &       wa(11*z+1))
C
C Quasi-Newton iteration
C
      call apprd(ld,n,nm,nn,d,bvbt,ctrc,y,hom,ww,wi,sigma,f10,f20,f30,
     &         temp1,temp2,temp3,qr,wa,fvec,s1)
      return
      end
      subroutine initp4(ld,n,nm,m,l,nn,c1,c2,d,bvbt,ctrc,y,hom,q,p,tau,
     &               f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,
     &               qr,wa,fvec,s1)
C
C Strategy 4 for choosing the starting point for homotopy
C
C Input parameters:
C ld - leading dimension of matrices
C n,nm,m,l - dimensions of system and model
C nn - Jacobian size
C c1,c2 - parameters used in definition of D
C d,bvbt,ctrc - D, B V B^t, C^t R C
C
C Output parameters:
C y - point in vector form
C hom - parameter indicating which homotopy map will be used, depends
C       on the result of quasi-Newton iteration
C
C Working storage:
C q,p,tau,f10,f20,f30,ww,wi,sigma,temp1,temp2,temp3,qr  - working matrices
C wa,fvec,s1 - working arrays
C
      integer hom,i,l,ld,m,n,nm,nn,realnm,z
```

```
      double precision bvbt(ld,ld),c1,c2,ctrc(ld,ld),d(ld,ld),f10(ld,ld)
      double precision f20(ld,ld),f30(ld,ld),fvec(nn),p(ld,ld),q(ld,ld)
      double precision qr(nn,nn),s1(ld),sigma(ld,ld),small,t1,tau(ld,ld)
      double precision temp1(ld,ld),temp2(ld,ld),temp3(ld,ld)
      double precision wa(nn*(nn+13)/2),wi(ld,ld),ww(ld,ld),y(nn)
C
      z = ld * ld
      small = 1.d-4
C
C Compute solutions of initial problem with ranks m and l
C
      t1 = 1.d0 / (2.d0 * c1)
      call scmat(ld,t1,bvbt,q,n,n)
      call scmat(ld,t1,ctrc,p,n,n)
C
C nm = min(m,l)
C
      realnm = nm
      nm = m
      if (l .lt. nm) nm = l
C
C Compute (W,U,Sigma) for given (Q,P), rank nm-1
C
      call cg(ld,n,nm,q,p,ww,wi,sigma,tau,temp1,temp2,temp3,
     &       wa(1),wa(z+1),wa(2*z+1),wa(3*z+1),wa(4*z+1),wa(5*z+1),
     &       wa(6*z+1),wa(7*z+1),wa(8*z+1),wa(9*z+1),wa(10*z+1),
     &       wa(11*z+1))
C
C Augment Sigma
C
      if (nm .lt. realnm) then
        do 302 i = nm+1,realnm
          sigma(i,i) = small
302     continue
      endif
      nm = realnm
C
C Quasi-Newton iteration
C
      call apprd(ld,n,nm,nn,d,bvbt,ctrc,y,hom,ww,wi,sigma,f10,f20,f30,
     &          temp1,temp2,temp3,qr,wa,fvec,s1)
      return
      end
      subroutine apprd(ld,n,nm,nn,d,bvbt,ctrc,y,hom,ww,wi,sigma,
     &               f10,f20,f30,temp1,temp2,temp3,qr,wa,fvec,s1)
C
C Quasi-Newton iteration for solving initial problem (matrix D)
C
C Input parameters:
C ld - leading dimension of matrices
C n,nm - dimensions of system and model
C nn - Jacobian size
C d,bvbt,ctrc - D, B V B^t, C^t R C
C y - starting point in vector form
C
C Output parameters:
```

```
C y - final point in vector form
C hom - parameter indicating which homotopy map will be used, depends
C       on the result of quasi-Newton iteration
C
C Working storage:
C ww,wi,sigma,f10,f20,f30,temp1,temp2,temp3,qr  - working matrices
C wa,fvec,s1 - working arrays
C
      external fcn
      integer hom,ld,ldfjac,lwa,n,nm,nn
      double precision bvbt(ld,ld),ctrc(ld,ld),d(ld,ld),eps,f10(ld,ld)
      double precision f20(ld,ld),f30(ld,ld),fvec(nn),qr(nn,nn),s1(ld)
      double precision sigma(ld,ld),temp1(ld,ld),temp2(ld,ld)
      double precision temp3(ld,ld),wa(nn*(n+13)/2),wi(ld,ld)
      double precision ww(ld,ld),y(nn)
C
      ldfjac = nn
      lwa = nn*(nn+13)/2
      hom = 0
      eps = 1.d-5
C
      call xwus(ld,n,nm,2,ww,wi,sigma,y)
      call hybrj1(fcn,nn,y,fvec,qr,ldfjac,eps,hom,wa,lwa)
      call xwus(ld,n,nm,1,ww,wi,sigma,y)
      return
      end
      subroutine fcn(n2,x,fvec,fjac,ldfjac,iflag)
C
C This procedure is called from HYBRJ1. Depending on iflag it computes
C function (iflag=1) or Jacobian (iflag=2).
C
      integer hom,i,iflag,ld,ldfjac,n,n2,nm,p2(10),type,z
      double precision a(20,20),aa(1),bvbt(20,20),ctrc(20,20),d(20,20)
      double precision fjac(ldfjac,n2),fvec(n2),lambda,p1(10000),x(n2)
      common /int/ n,nm,ld
      common /syste/ a,bvbt,ctrc,d
C
      lambda = 0.d0
      type = 2
      hom = 0
      p2(1) = ld
      p2(2) = n
      p2(3) = nm
      p2(4) = type
      p2(5) = hom
      z = ld * ld
      call col2x(ld,ld,ld,a,p1(1))
      call col2x(ld,ld,ld,d,p1(z+1))
      call col2x(ld,ld,ld,bvbt,p1(2*z+1))
      call col2x(ld,ld,ld,ctrc,p1(3*z+1))
C
      if (iflag .eq. 1) call rho(aa,lambda,x,fvec,p1,p2)
      if (iflag .eq. 2) then
        do 331 i = 1,n
          call rhojac(aa,lambda,x,fjac(1,i),i,p1,p2)
331     continue
```

```
      endif
      return
      end
      subroutine rho(aa,lambda,x,v,p,ip)
C
C Subroutine RHO is the interface between HOMPACK and RRHO. Its purpose
C is to process parameters passed thru arrays P and IP, and send it to
C RRHO.
C
      integer hom,ip(*),ld,n,nm,type,z
      double precision aa(*),lambda,p(*),v(*),x(*)
C
      ld = ip(1)
      n = ip(2)
      nm = ip(3)
      type = ip(4)
      hom = ip(5)
      z = ld * ld
C
      call rrho(ld,n,nm,type,hom,lambda,x,p(1),p(z+1),p(2*z+1),
     &         p(3*z+1),p(4*z+1),p(5*z+1),p(6*z+1),v,p(7*z+1),p(8*z+1),
     &         p(9*z+1),p(10*z+1),p(11*z+1),p(12*z+1),p(13*z+1),
     &         p(14*z+1))
      return
      end
      subroutine rrho(ld,n,nm,type,hom,lambda,x,a,d,bvbt,ctrc,f10,
     &              f20,f30,v,ww,wi,sigma,temp1,temp2,temp3,aal,mat)
C
C RRHO evaluates rho(a,lambda,x) at given point.
C
C Input parameters:
C ld - leading dimension of matrices
C n,nm - dimensions of system and model
C type - indicates whether RRHO is used for curve tracking (type = 1)
C       or quasi-Newton iterations (type = 2 or 3)
C hom - indicates type of homotopy map: hom = 0 indicates Newton
C      homotopy, hom = 1 indicates rho(a,lambda,x) = F(a,lambda,x)
C lambda,x - input point (lambda,x)
C a,d,bvbt,ctrc - A,D, B V B^t, C^t R C
C f10,f20,f30 - F(0,X(0))
C
C Output parameters:
C v - evaluated rho
C
C Working storage:
C ww,wi,sigma,temp1,temp2,temp3,aal,mat  - working matrices
C
      integer hom,ld,n,nm,type
      double precision a(ld,ld),aal(ld,ld),bvbt(ld,ld),ctrc(ld,ld)
      double precision d(ld,ld),f10(ld,ld),f20(ld,ld),f30(ld,ld),lambda
      double precision mat(ld,ld),sigma(ld,ld),t1,temp1(ld,ld)
      double precision temp2(ld,ld),temp3(ld,ld),v(*),wi(ld,ld)
      double precision ww(ld,ld)
C
      call xwus(ld,n,nm,1,ww,wi,sigma,x)
C
```

```
C Compute AAL = lambda * A + (1-lambda) * D
C
      t1 = 1.d0 - lambda
      call scmat(ld,lambda,a,temp1,n,n)
      call scmat(ld,t1,d,temp2,n,n)
      call addmat(ld,temp1,temp2,aal,n,n,1)
C
C Determine which matrix is used as A(lambda)
C
      if (type .eq. 1) call scmat(ld,1.d0,aal,mat,n,n)
      if (type .eq. 2) call scmat(ld,1.d0,d,mat,n,n)
      if (type .eq. 3) call scmat(ld,1.d0,a,mat,n,n)
C
C Compute F1(a,lambda,x)
C
      call mulmat(ld,wi,mat,temp1,nm,n,n,1)
      call mulmat(ld,temp1,ww,temp2,nm,nm,n,1)
      call mulmat(ld,temp2,sigma,temp1,nm,nm,nm,1)
      call mulmat(ld,temp1,ww,temp3,nm,n,nm,3)
C
      call mulmat(ld,sigma,ww,temp1,nm,n,nm,3)
      call mulmat(ld,temp1,mat,temp2,nm,n,n,3)
      call addmat(ld,temp2,temp3,temp3,nm,n,1)
C
      call mulmat(ld,wi,bvbt,temp1,nm,n,n,1)
      call addmat(ld,temp1,temp3,temp3,nm,n,1)
C
      if (type .eq. 1 .and. hom .ne. 1) then
        t1 = 1.d0 - lambda
        call scmat(ld,t1,f10,temp1,nm,n)
        call addmat(ld,temp1,temp3,temp3,nm,n,2)
      endif
      call xmat(ld,2,nm,n,temp3,v)
C
C Compute F2(a,lambda,x)
C
      call mulmat(ld,wi,sigma,temp1,n,nm,nm,2)
      call mulmat(ld,temp1,wi,temp2,n,n,nm,1)
      call mulmat(ld,temp2,mat,temp1,n,n,n,1)
      call mulmat(ld,temp1,ww,temp3,n,nm,n,1)
C
      call mulmat(ld,mat,wi,temp1,n,nm,n,4)
      call mulmat(ld,temp1,sigma,temp2,n,nm,nm,1)
      call addmat(ld,temp2,temp3,temp3,n,nm,1)
C
      call mulmat(ld,ctrc,ww,temp1,n,nm,n,1)
      call addmat(ld,temp1,temp3,temp3,n,nm,1)
C
      if (type .eq. 1 .and. hom .ne. 1) then
        call scmat(ld,t1,f20,temp1,n,nm)
        call addmat(ld,temp1,temp3,temp3,n,nm,2)
      endif
      call xmat(ld,2,n,nm,temp3,v(1+n*nm))
C
C Compute F3(a,lambda,x)
C
```

```
      call mulmat(ld,wi,ww,temp3,nm,nm,n,1)
      do 351 i = 1,nm
        temp3(i,i) = temp3(i,i) - 1.d0
351   continue
C
      if (type .eq. 1 .and. hom .ne. 1) then
        call scmat(ld,t1,f30,temp1,nm,nm)
        call addmat(ld,temp1,temp3,temp3,nm,nm,2)
      endif
      call xmat(ld,2,nm,nm,temp3,v(1+2*n*nm))
      return
      end
      subroutine rhojac(aa,lambda,x,v,kk,p,ip)
C
C Subroutine RHOJAC is the interface between HOMPACK and RRHOJ. Its purpose
C is to process parameters passed thru arrays P and IP, and send it to
C RRHOJ.
C
      integer hom,ip(*),kk,ld,n,nm,type,z
      double precision aa(*),lambda,p(*),v(*),x(*)
C
      ld = ip(1)
      n = ip(2)
      nm = ip(3)
      type = ip(4)
      hom = ip(5)
      z = ld * ld
C
      call rrhoj(ld,n,nm,type,hom,lambda,x,kk,p(1),p(z+1),
     &         p(2*z+1),p(3*z+1),p(4*z+1),p(5*z+1),p(6*z+1),v,p(7*z+1),
     &         p(8*z+1),p(9*z+1),p(10*z+1),p(11*z+1),p(12*z+1),
     &         p(13*z+1),p(14*z+1),p(15*z+1))
      return
      end
      subroutine rrhoj(ld,n,nm,type,hom,lambda,x,kk,a,d,bvbt,ctrc,
     &               f10,f20,f30,v,ww,wi,sigma,temp1,temp2,temp3,aal,
     &               mat,amd)
C
C RRHOJ evaluates kk-th column of Jacobian at given point.
C
C Input parameters:
C ld - leading dimension of matrices
C n,nm - dimensions of system and model
C type - indicates whether RRHO is used for curve tracking (type = 1)
C       or quasi-Newton iterations (type = 2 or 3)
C hom - indicates type of homotopy map: hom = 0 indicates Newton
C      homotopy, hom = 1 indicates rho(a,lambda,x) = F(a,lambda,x)
C lambda,x - input point (lambda,x)
C kk - column number
C a,d,bvbt,ctrc - A,D, B V B^t, C^t R C
C f10,f20,f30 - F(0,X(0))
C
C Output parameters:
C v - evaluated function
C
C Working storage:
```

```
C ww,wi,sigma,temp1,temp2,temp3,aal,mat,amd  - working matrices
C
      integer der,hom,i,j,k,kk,ld,n,nm,type
      double precision a(ld,ld),aal(ld,ld),amd(20,20),bvbt(ld,ld)
      double precision ctrc(ld,ld),d(ld,ld),f10(ld,ld),f20(ld,ld)
      double precision f30(ld,ld),lambda,mat(ld,ld),sigma(ld,ld),t1
      double precision temp1(ld,ld),temp2(ld,ld),temp3(ld,ld),v(*)
      double precision wi(ld,ld),ww(ld,ld)
C
      call xwus(ld,n,nm,1,ww,wi,sigma,x)
C
C Compute AAL = lambda * A + (1-lambda) * D
C
      t1 = 1.d0 - lambda
      call scmat(ld,lambda,a,temp1,n,n)
      call scmat(ld,t1,d,temp2,n,n)
      call addmat(ld,temp1,temp2,aal,n,n,1)
C
C Determine which matrix is used as A(lambda)
C
      if (type .eq. 1) call scmat(ld,1.d0,aal,mat,n,n)
      if (type .eq. 2) call scmat(ld,1.d0,d,mat,n,n)
      if (type .eq. 3) call scmat(ld,1.d0,a,mat,n,n)
C
C Determine variable with respect to which the derivative is sought
C der=4 - lambda, der=1 - W, der=2 - U, der=3 - Sigma
C
      if (type .eq. 1) k = kk
      if (type .eq. 2) k = kk + 1
      if (type .eq. 3) k = kk + 1
C
      if (k .eq. 1) der = 4
      if (k .gt. 1 .and. k .le. n*nm+1) der = 1
      if (k .gt. n*nm+1 .and. k .le. 2*n*nm+1) der = 2
      if (k .gt. 2*n*nm+1) der = 3
C
C Determine position (i,j) in W, U or Sigma
C
      if (der .eq. 1) call compij(k-1,n,nm,i,j)
      if (der .eq. 2) call compij(k-n*nm-1,nm,n,i,j)
      if (der .eq. 3) call compij(k-2*n*nm-1,nm,nm,i,j)
C
C Derivatives with respect to Wij
C
      if (der .eq. 1) then
C
C Compute derivatives of F1(a,lambda,x)
C
        call mulmat(ld,wi,mat,temp1,nm,n,n,1)
        call mulmat(ld,temp1,ww,temp2,nm,nm,n,1)
        call mulmat(ld,temp2,sigma,temp1,nm,nm,nm,1)
        call setmat(ld,0.d0,temp3,nm,n)
        call rowcol(ld,temp3,temp1,nm,i,j,4)
C
        call mulmat(ld,wi,mat,temp1,nm,n,n,1)
        call setmat(ld,0.d0,temp2,nm,nm)
```

```
      call rowcol(ld,temp2,temp1,nm,j,i,4)
      call mulmat(ld,temp2,sigma,temp1,nm,nm,nm,1)
      call mulmat(ld,temp1,ww,temp2,nm,n,nm,3)
      call addmat(ld,temp2,temp3,temp3,nm,n,1)
C
      call setmat(ld,0.d0,temp1,nm,n)
      call rowcol(ld,temp1,sigma,nm,i,j,4)
      call mulmat(ld,temp1,mat,temp2,nm,n,n,3)
      call addmat(ld,temp2,temp3,temp3,nm,n,1)
      call xmat(ld,2,nm,n,temp3,v)
C
C Compute derivatives of F2(a,lambda,x)
C
      call mulmat(ld,wi,sigma,temp1,n,nm,nm,2)
      call mulmat(ld,temp1,wi,temp2,n,n,nm,1)
      call mulmat(ld,temp2,mat,temp1,n,n,n,1)
      call setmat(ld,0.d0,temp3,n,nm)
      call rowcol(ld,temp3,temp1,n,j,i,4)
      call rowcol(ld,temp3,ctrc,n,j,i,4)
      call xmat(ld,2,n,nm,temp3,v(1+n*nm))
C
C Compute derivatives of F3(a,lambda,x)
C
      call setmat(ld,0.d0,temp3,nm,nm)
      call rowcol(ld,temp3,wi,nm,j,i,4)
      call xmat(ld,2,nm,nm,temp3,v(1+2*n*nm))
     endif
C
C Derivatives with respect to Uij
C
     if (der .eq. 2) then
C
C Compute derivatives of F1(a,lambda,x)
C
      call mulmat(ld,mat,ww,temp1,n,nm,n,1)
      call mulmat(ld,temp1,sigma,temp2,n,nm,nm,1)
      call mulmat(ld,temp2,ww,temp1,n,n,nm,3)
      call setmat(ld,0.d0,temp3,nm,n)
      call rowcol(ld,temp3,temp1,n,i,j,1)
      call rowcol(ld,temp3,bvbt,n,i,j,1)
      call xmat(ld,2,nm,n,temp3,v)
C
C Compute derivatives of F2(a,lambda,x)
C
      call mulmat(ld,sigma,wi,temp1,nm,n,nm,1)
      call mulmat(ld,temp1,mat,temp2,nm,n,n,1)
      call mulmat(ld,temp2,ww,temp1,nm,nm,n,1)
      call setmat(ld,0.d0,temp3,n,nm)
      call rowcol(ld,temp3,temp1,nm,j,i,1)
C
      call mulmat(ld,wi,sigma,temp1,n,nm,nm,2)
      call setmat(ld,0.d0,temp2,n,n)
      call rowcol(ld,temp2,temp1,n,j,i,4)
      call mulmat(ld,temp2,mat,temp1,n,n,n,1)
      call mulmat(ld,temp1,ww,temp2,n,nm,n,1)
      call addmat(ld,temp2,temp3,temp3,n,nm,1)
```

```
C
       call setmat(ld,0.d0,temp1,n,nm)
       call rowcol(ld,temp1,mat,n,i,j,3)
       call mulmat(ld,temp1,sigma,temp2,n,nm,nm,1)
       call addmat(ld,temp2,temp3,temp3,n,nm,1)
       call xmat(ld,2,n,nm,temp3,v(1+n*nm))
C
C Compute derivatives of F3(a,lambda,x)
C
       call setmat(ld,0.d0,temp3,nm,nm)
       call rowcol(ld,temp3,ww,nm,i,j,1)
       call xmat(ld,2,nm,nm,temp3,v(1+2*n*nm))
     endif
C
C Derivatives with respect to SIGMAij
C
     if (der .eq. 3) then
C
C Compute derivatives of F1(a,lambda,x)
C
       call mulmat(ld,wi,mat,temp1,nm,n,n,1)
       call mulmat(ld,temp1,ww,temp2,nm,nm,n,1)
       call setmat(ld,0.d0,temp1,nm,nm)
       call rowcol(ld,temp1,temp2,nm,j,i,4)
       call mulmat(ld,temp1,ww,temp3,nm,n,nm,3)
C
       call mulmat(ld,ww,mat,temp1,nm,n,n,4)
       call setmat(ld,0.d0,temp2,nm,n)
       call rowcol(ld,temp2,temp1,n,i,j,1)
       call addmat(ld,temp2,temp3,temp3,nm,n,1)
       call xmat(ld,2,nm,n,temp3,v)
C
C Compute derivatives of F2(a,lambda,x)
C
       call setmat(ld,0.d0,temp1,n,nm)
       call rowcol(ld,temp1,wi,n,j,i,3)
       call mulmat(ld,temp1,wi,temp2,n,n,nm,1)
       call mulmat(ld,temp2,mat,temp1,n,n,n,1)
       call mulmat(ld,temp1,ww,temp3,n,nm,n,1)
C
       call mulmat(ld,mat,wi,temp1,n,nm,n,4)
       call setmat(ld,0.d0,temp2,n,nm)
       call rowcol(ld,temp2,temp1,n,j,i,4)
       call addmat(ld,temp2,temp3,temp3,n,nm,1)
       call xmat(ld,2,n,nm,temp3,v(1+n*nm))
C
C Compute derivatives of F3(a,lambda,x)
C
       call setmat(ld,0.d0,temp3,nm,nm)
       call xmat(ld,2,nm,nm,temp3,v(1+2*n*nm))
     endif
C
C Derivatives with respect to lambda
C
     if (der .eq. 4) then
C
```

```
      call addmat(ld,a,d,amd,n,n,3)
C
C Compute derivatives of F1(a,lambda,x)
C
      call mulmat(ld,wi,amd,temp1,nm,n,n,1)
      call mulmat(ld,temp1,ww,temp2,nm,nm,n,1)
      call mulmat(ld,temp2,sigma,temp1,nm,nm,nm,1)
      call mulmat(ld,temp1,ww,temp3,nm,n,nm,3)
C
      call mulmat(ld,sigma,ww,temp1,nm,n,nm,3)
      call mulmat(ld,temp1,amd,temp2,nm,n,n,3)
      call addmat(ld,temp2,temp3,temp3,nm,n,1)
C
      if (hom .ne. 1) call addmat(ld,f10,temp3,temp3,nm,n,1)
      call xmat(ld,2,nm,n,temp3,v)
C
C Compute derivatives of F2(a,lambda,x)
C
      call mulmat(ld,wi,sigma,temp1,n,nm,nm,2)
      call mulmat(ld,temp1,wi,temp2,n,n,nm,1)
      call mulmat(ld,temp2,amd,temp1,n,n,n,1)
      call mulmat(ld,temp1,ww,temp3,n,nm,n,1)
C
      call mulmat(ld,amd,wi,temp1,n,nm,n,4)
      call mulmat(ld,temp1,sigma,temp2,n,nm,nm,1)
      call addmat(ld,temp2,temp3,temp3,n,nm,1)
C
      if (hom .ne. 1) call addmat(ld,f20,temp3,temp3,n,nm,1)
      call xmat(ld,2,n,nm,temp3,v(1+n*nm))
C
C Compute derivatives of F3(a,lambda,x)
C
      call setmat(ld,0.d0,temp3,nm,nm)
      if (hom .ne. 1) call addmat(ld,f30,temp3,temp3,nm,nm,1)
      call xmat(ld,2,nm,nm,temp3,v(1+2*n*nm))
    endif
    return
    end
    subroutine compij(ij,dim1,dim2,i,j)
C
C Compute (i,j) coordinate of element of W, U or Sigma
C
    integer dim1,dim2,i,j,ij,loop
C
    i = 1
    do 301 loop = 1,dim1
      if (ij .le. dim2) then
        j = ij
        return
      endif
      ij = ij - dim2
      i = i + 1
301   continue
    return
    end
    subroutine mulmat(ld,a,b,c,m,n,k,type)
```

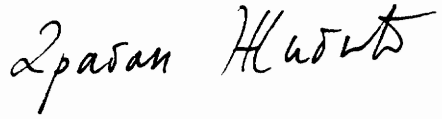
```
C
C Perform matrix-matrix multiplication: C(m,n) = A(m,k) * B(k,n)
C NOTE: A or B may be used as A^t or B^t, depending on type, i.e.,
C       if type=2 then C(m,n) = A^t(m,k) * B(k,n)
C type - 1 - C = A * B
C        2 - C = A^t * B
C        3 - C = A * B^t
C        4 - C = A^t * B^t
C
      integer i,j,k,ld,m,n,type
      double precision a(ld,ld),b(ld,ld),c(ld,ld),ddot
C
      do 301 j = 1,n
        do 302 i = 1,m
          goto (403,404,405,406) type
403       c(i,j) = ddot(k,a(i,1),ld,b(1,j),1)
          goto 302
404       c(i,j) = ddot(k,a(1,i),1,b(1,j),1)
          goto 302
405       c(i,j) = ddot(k,a(i,1),ld,b(j,1),ld)
          goto 302
406       c(i,j) = ddot(k,a(1,i),1,b(j,1),ld)
302     continue
301   continue
      return
      end
      subroutine addmat(ld,a,b,c,m,n,type)
C
C Perform matrix-matrix addition/subtraction: C(m,n) = A(m,n) + B(m,n)
C NOTE: in calling routines C is usually A or B
C type - 1 - C =  A + B
C        2 - C = -A + B
C        3 - C =  A - B
C        4 - C = -A - B
C
      integer i,j,ld,m,n,type
      double precision a(ld,ld),b(ld,ld),c(ld,ld)
C
      do 301 j = 1,n
        do 302 i = 1,m
          goto (403,404,405,406) type
403       c(i,j) =  a(i,j) +  b(i,j)
          goto 302
404       c(i,j) = -a(i,j) +  b(i,j)
          goto 302
405       c(i,j) =  a(i,j) -  b(i,j)
          goto 302
406       c(i,j) = -a(i,j) -  b(i,j)
302     continue
301   continue
      return
      end
      subroutine scmat(ld,s,a,b,m,n)
C
C Scalar multiplication of matrix: B(m,n) = s * A(m,n)
C
```

```
      integer i,ld,m,n
      double precision a(ld,ld),b(ld,ld),s
C
      do 301 i = 1,n
        call dcopy(m,a(1,i),1,b(1,i),1)
        call dscal(m,s,b(1,i),1)
301   continue
      return
      end
      subroutine setmat(ld,s,a,m,n)
C
C Set all elements of A(m,n) to scalar s
C
      integer i,j,ld,m,n
      double precision a(ld,ld),s
C
      do 301 j = 1,n
        do 302 i = 1,m
          a(i,j) = s
302     continue
301   continue
      return
      end
      subroutine trans(ld,a,b,m,n)
C
C B(n,m) = A(m,n)^t
C
      integer i,j,ld,m,n
      double precision a(ld,ld),b(ld,ld)
C
      do 301 j = 1,m
        do 302 i = 1,n
          b(i,j) = a(j,i)
302     continue
301   continue
      return
      end
      subroutine rowcol(ld,mat1,mat2,length,fix1,fix2,type)
C
C Depending on type set certain row (column) of MAT1 to row (column) of MAT2
C type - 1 - fix1 row of MAT1 = fix2 row of MAT2
C        2 - fix1 row of MAT1 = fix2 column of MAT2
C        3 - fix1 column of MAT1 = fix2 row of MAT2
C        4 - fix1 column of MAT1 = fix2 column of MAT2
C
      integer fix1,fix2,length,ld,type
      double precision mat1(ld,ld),mat2(ld,ld)
C
      goto (401,402,403,404) type
401   call daxpy(length,1.d0,mat2(fix2,1),ld,mat1(fix1,1),ld)
      return
402   call daxpy(length,1.d0,mat2(1,fix2),1,mat1(fix1,1),ld)
      return
403   call daxpy(length,1.d0,mat2(fix2,1),ld,mat1(1,fix1),1)
      return
404   call daxpy(length,1.d0,mat2(1,fix2),1,mat1(1,fix1),1)
```

```
      return
      end
      subroutine xmat(ld,xtyp,dim1,dim2,mat,x)
C
C Depending on type, transfer vector to matrix or vice versa
C type - 1 - transfer vector X to MAT(dim1,dim2)
C       2 - transfer MAT(dim1,dim2) to vector X
C
      integer count,dim1,dim2,i,j,ld,xtyp
      double precision mat(ld,ld),x(*)
      count = 1
      do 301 i = 1,dim1
        do 302 j = 1,dim2
          if (xtyp .eq. 1) mat(i,j) = x(count)
          if (xtyp .eq. 2) x(count) = mat(i,j)
          count = count + 1
302     continue
301   continue
      return
      end
      subroutine xwus(ld,n,nm,xtyp,ww,wi,sigma,x)
C
C Depending on type, transfer vector X to matrix form or vice versa
C type - 1 - transfer vector X to (W,U,Sigma)
C       2 - transfer (W,U,Sigma) to vector X
C
      integer ld,n,nm,xtyp
      double precision sigma(ld,ld),wi(ld,ld),ww(ld,ld),x(*)
      call xmat(ld,xtyp,n,nm,ww,x(1))
      call xmat(ld,xtyp,nm,n,wi,x(n*nm+1))
      call xmat(ld,xtyp,nm,nm,sigma,x(2*n*nm+1))
      return
      end
      subroutine col2x(ld,dim1,dim2,mat,x)
C
C Transfer MAT(dim1,dim2) to vector X, column by column
C
      integer count,dim1,dim2,i,j,ld
      double precision mat(ld,ld),x(*)
      count = 1
      do 301 j = 1,dim2
        do 302 i = 1,dim1
          x(count) = mat(i,j)
          count = count + 1
302     continue
301   continue
      return
      end
      subroutine cg(ld,n,nm,q,p,ww,wi,sigma,tau,temp1,temp2,temp3,
     &            t1,t2,t1i,t2i,q12,u,v,sigm34,sig4,sigm1,sigm4,s1)
C
C Compute a contragredient transformation (W,U,Sigma) of (Q,P).
C
C Input parameters:
C ld - leading dimension of matrices
C n - number of rows in W
```

```
C nm -  number of rows in Sigma
C q,p - (Q,P)
C
C Output parameters:
C ww,wi,sigma - transformation (W,U,Sigma), W = U^(-1)
C tau - product of W(1:n,1:nm) and U(1:nm,1:n)
C
C Working storage:
C temp1,temp2,temp3,t1,t2,t1i,t2i,q12,u,v,sigm34,sig4,sigm1,sigm4  -
C     - working matrices
C s1 - working array
C
      integer i,ind,j,ld,n,nm
      double precision p(ld,ld),q(ld,ld),q12(ld,ld),s1(ld),sig4(ld,ld)
      double precision sigm1(ld,ld),sigm34(ld,ld),sigm4(ld,ld)
      double precision sigma(ld,ld),small,t1(ld,ld),t1i(ld,ld),t2(ld,ld)
      double precision t2i(ld,ld),tau(ld,ld),temp1(ld,ld),temp2(ld,ld)
      double precision temp3(ld,ld),tmp,u(ld,ld),v(ld,ld),wi(ld,ld)
      double precision ww(ld,ld)
C
      small = 1.d-3
C
C Compute P = V * diag(Omega 0) * V^t
C
      call rs(ld,n,p,s1,1,v,temp1,temp2,ind)
C
C Compute T1 = V * diag(Omega^(-1/2) I), T1^(-1) = diag(Omega^(1/2) I) * V^t
C
      call setmat(ld,0.d0,temp1,n,n)
      call setmat(ld,0.d0,temp2,n,n)
      do 301 i = 1,nm
        if (s1(n-i+1) .lt. small) s1(n-i+1) = small
        temp1(i,i) = 1.d0 / dsqrt(s1(n-i+1))
        temp2(i,i) = dsqrt(s1(n-i+1))
301   continue
      do 302 i = nm+1,n
        temp1(i,i) = 1.d0
        temp2(i,i) = 1.d0
302   continue
      do 303 j = 1,n/2
        do 304 i = 1,n
          tmp = v(i,j)
          v(i,j) = v(i,n-j+1)
          v(i,n-j+1) = tmp
304     continue
303   continue
      call mulmat(ld,v,temp1,t1,n,n,n,1)
      call mulmat(ld,temp2,v,t1i,n,n,n,3)
C
C Compute T1^(-1) * Q * T1^(-t) and store in temp3,
C store appropriate part of temp3 in Q12
C
      call mulmat(ld,t1i,q,temp1,n,n,n,1)
      call mulmat(ld,temp1,t1i,temp3,n,n,n,3)
      call scmat(ld,1.d0,temp3(1,nm+1),q12,nm,n-nm)
C
```

```
C Compute Q11 = U * Omega * U^t
C
      call rs(ld,nm,temp3,s1,1,u,temp1,temp2,ind)
C
C Compute Sigma, Omega^(-1), Omega^(1/4), Omega^(-1/4), Omega^(-3/4)
C
      call setmat(ld,0.d0,sigma,nm,nm)
      call setmat(ld,0.d0,sig4,n,n)
      call setmat(ld,0.d0,sigm4,n,n)
      call setmat(ld,0.d0,sigm1,n,n)
      call setmat(ld,0.d0,sigm34,n,n)
      do 311 i = 1,nm
        if (s1(i) .lt. small) s1(i) = small
        sigma(i,i) = dsqrt(s1(i))
        sigm1(i,i) = 1.d0/s1(i)
        sig4(i,i) = dsqrt(dsqrt(s1(i)))
        sigm4(i,i) = 1.d0/sig4(i,i)
        sigm34(i,i) = sigm4(i,i) * sigm4(i,i) * sigm4(i,i)
311   continue
C
C Compute T2, T2^(-1)
C
      call setmat(ld,0.d0,t2,n,n)
      call setmat(ld,0.d0,t2i,n,n)
      do 312 i = nm+1,n
        t2(i,i) = -1.d0
        t2i(i,i) = -1.d0
312   continue
C
C Compute U * Sigma^(1/4), Sigma^(-1/4) * U^t
C
      call mulmat(ld,u,sig4,t2,nm,nm,nm,1)
      call mulmat(ld,sigm4,u,t2i,nm,nm,nm,3)
C
C Compute Q12^t * U * Sigma^(-3/4), Q12^t * U * Sigma^(-1) * U^t
C
      call mulmat(ld,q12,u,temp1,n,nm,nm,2)
      call mulmat(ld,temp1,sigm34,temp3,n,nm,nm,1)
      call mulmat(ld,q12,u,temp1,n,nm,nm,2)
      call mulmat(ld,temp1,sigm1,temp2,n,nm,nm,1)
      call mulmat(ld,temp2,u,temp1,n,nm,nm,3)
      do 321 i = 1,n-nm
        do 322 j = 1,nm
          t2(nm+i,j) = temp3(i,j)
          t2i(nm+i,j) = temp1(i,j)
322     continue
321   continue
C
C Compute W = T1 * T2, U = W^(-1) = T2^(-1) * T1^(-1), tau = W * W^(-1)
C
      call mulmat(ld,t1,t2,ww,n,n,n,1)
      call mulmat(ld,t2i,t1i,wi,n,n,n,1)
      call mulmat(ld,ww,wi,tau,n,n,nm,1)
      return
      end
```

VITA.

Dragan Žigić was born on August 27, 1960, in Šabac, Yugoslavia. He graduated from IX Beogradska Gimnazija, Novi Beograd, in May 1979, and earned a BS degree in mathematics from the University of Belgrade, Faculty of Sciences and Mathematics, in June 1983. From August 1983 to August 1989 he was employed by Energodata, a Belgrade computer engineering company. In June 1991 he earned an MS degree in computer science from Virginia Polytechnic Institute and State University, Blacksburg, Virginia.