Presented at the 2002 American Control Conference, May 8 – 10, Anchorage, AK

ACC02-INV2506

Real-Time Computational Methods for SDRE Nonlinear Control of Missiles[*]

By
P. K. Menon, T. Lam, L. S. Crawford, V. H. L. Cheng
Optimal Synthesis Inc.
4966 El Camino Real, Suite 108
Los Altos, CA 94022

Abstract

The state dependent Riccati equation (SDRE) technique is a recently developed methodology for designing control laws and estimation algorithms for missile flight control. Although its potential is well recognized by control theory experts, the industry acceptance of the technique has been slow. The main reasons for this are: a) unlike linear control technology, the SDRE approach requires advanced numerical methods for its implementation, which are not currently available off-the-shelf, b) the perception that this technique may not be computationally feasible for real-time implementation on flight control processors.

Both of these issues are addressed in this paper. A software package for real-time implementation of the SDRE technique was developed during the present research. The execution of this software at speeds up to 2 kHz sample rates on problems of the size commonly encountered in missile flight control applications was then demonstrated on commercial off-the-shelf processors.

1. Introduction

State Dependent Riccati Equation (SDRE) method is a recently developed technique for the control of nonlinear dynamic systems [1 – 4]. This paper discusses the development of numerical algorithms for the practical implementation of the SDRE control technology in missile flight control problems. The SDRE control technology has extensive applications in the design of advanced flight control systems, and also in numerous automotive and process control applications. The main difficulty encountered by the users attempting to use the SDRE technology is that of developing real-time computational algorithms that can be implemented on

[*] Research supported under Air Force Contract No. F08630-99-C-0060, with Dr. J. R. Cloutier serving as the contract monitor. Paper submitted for possible presentation at the **2002 ACC**. © Optimal Synthesis Inc., 2001.

commercially available processors. This is due to the fact that the SDRE control technique demands advanced numerical algorithms and significantly more computational resources than conventional control algorithms. This paper addresses the speed and performance issues that arise in the practical implementation of the SDRE control technique. Commercial, off-the-shelf computer technology such as the *WindowsNT®* [5] and *LINUX* operating systems running on an Intel® Pentium® II/III platform [6] are employed in this work.

Model-based control concepts are currently gaining popularity in manufacturing and process control industries, as evidenced by the popularity of such solutions in chemical process control [7]. The model predictive control (MPC) methodology [8 - 14] is currently the most popular technique employed in these applications. The SDRE nonlinear technique is capable of delivering superior performance when compared with the model predictive control methodology, while providing stability guarantees not available from the model predictive control technique. However, algorithmic and implementation difficulties have not permitted faster acceptance of the SDRE technology by the industry.

The following lists the accomplishments of the present research:

1. Developed fast, C language-implementations of two algorithms for the solution of algebraic Riccati equations that form the central component of the SDRE design technique. The first approach is a direct method based on Schur decomposition of the Hamiltonian matrix [15, 16]. The second approach is an iterative algorithm that refines an initial guess of the Riccati solution using the Kleinman algorithm [17, 18]. These software packages use extensively re-worked code components from the public-domain linear algebra package *LAPACK* [19]. Results from both these algorithms have been compared with results from software packages such as MATLAB®[20].
2. Solution speeds of these algorithms have been assessed on Intel Pentium II, 300 MHz and Pentium III 450 MHz computers running the WindowsNT operating system, and on a real-time Motorola® 604e PowerPC® board (DS1103) from dSPACE® Inc [21]. The dSPACE PowerPC board is used extensively in automotive control system development. The maximum speed obtained for a missile guidance-control problem is around 2 kHz.

A six-state nonlinear dynamic system representing a missile flight control system was used in these evaluations. Present research shows that the numerical algorithms are capable of delivering computational speeds far in excess of that required for implementing high-performance weapon flight control systems. Since the main computational burden in implementing the SDRE technique is in the solution of the algebraic Riccati equation, the present research demonstrates the feasibility of implementing the SDRE technique on commercial, off-the-shelf processors in real-time. Moreover, analysis has revealed that it may be possible to gain an additional 25% speed improvement by further refining the C code.

The next section will present a brief overview of the SDRE technique. Section 3 will discuss an approach for numerically constructing the state dependent coefficient form (SDC form) of the system dynamics from a numerical simulation. Approaches for solving the algebraic matrix Riccati equation will be discussed in Section 4. Approximate operation counts for each of the computational algorithms will also be given in that section. Section 5 will discuss code development and speed evaluation issues. Conclusions will be given in Section 6.

The present research clearly demonstrates that the SDRE technique can be implemented in real-time for a realistic, high-order multivariable system using commercial, off-the-shelf computing subsystems. Real-time SDRE computations for a problem of dimensions comparable to missile flight control system is used as an example.

2. An Overview of the SDRE Technique

State Dependent Riccati Equation (SDRE) method [1 - 4] is a recently emerged nonlinear control system design methodology for direct synthesis of nonlinear feedback controllers. Using a special form of the system dynamics, this approach permits the designer to employ linear optimal control methods such as the LQR methodology and the H∞ design technique for the synthesis of nonlinear control systems.

The SDRE design technique requires the dynamic model of the system to be placed in the state dependent coefficient (SDC) form. The SDC form has the structure:

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

In its most general form, the SDRE technique allows the inclusion of disturbance terms [4]. If desired, the designer can augment this system by introducing integral states and dynamic compensators to improve the tracking and disturbance rejection characteristics of the SDRE controller.

Note that the SDC form has the same structure as a linear dynamic system, but with the system matrix *A* and the control influence matrix *B* being functions of the state variables.

Reference 2 has shown that the SDC form can be derived for most nonlinear dynamic systems using simple algebraic manipulations. An approach for numerically constructing the SDC form from a given nonlinear simulation model will be discussed in the next section.

The matrices *A(x)* and *B(x)* evaluated at all values of the state vector *x* are assumed to be such that the system dynamics is controllable. Although the original theory discussed in Reference 2 does not impose this requirement, the need for full controllability arises from the numerical considerations.

The second ingredient of the SDRE design technique is the definition of a quadratic performance index in state dependent form:

$$J = \frac{1}{2}\int_{t_0}^{\infty}\left[x^T Q(x)x + u^T R(x)u\right]dt$$

The state dependent weighting matrices *Q(x)* and *R(x)* can be chosen to realize the desired performance objectives. In order to ensure local stability, the matrix *Q(x)* is required to be positive semi-definite for all *x* and the matrix *R(x)* is required to be positive definite for all *x*.

Next, a state dependent algebraic Riccati equation:

$$A^T(x)P(x) + P(x)A(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) + Q(x) = 0$$

is formulated and is solved for a positive definite state dependent matrix *P(x)*. The nonlinear state variable feedback control law is then constructed as:

$$u = -R^{-1}(x)B^T(x)P(x)x$$

Additional sophistication can be introduced in the SDRE design approach by including state estimators, and frequency weighting terms in the performance index. If adequate computational resources are available, the design problem can also be cast as an H∞ design or a μ-synthesis problem. An excellent overview of the SDRE design technique can be found in Reference [4].

It may be observed that the crucial part of the control law computation is the solution of the state-dependent Riccati equation. In rare situations, this Riccati equation may be solvable in closed-form. In most problems, however, this equation will have to be numerically solved at each sample instant.

A flowchart illustrating the steps involved in the computation of the SDRE control laws is given in Figure 1. At each sample, the state vector obtained from feedback sensors or estimators are used to compute the SDC matrices, which are then used to find the state dependent gains. The product of the state dependent gains and the state vector then yields the control variables.

Thus, the two main steps in the SDRE nonlinear control system design method are the computation of the SDC matrices *A(x)* and *B(x)*, and the solution of an algebraic matrix Riccati equation for *P(x).* The remaining steps involve matrix inversion and multiplication.

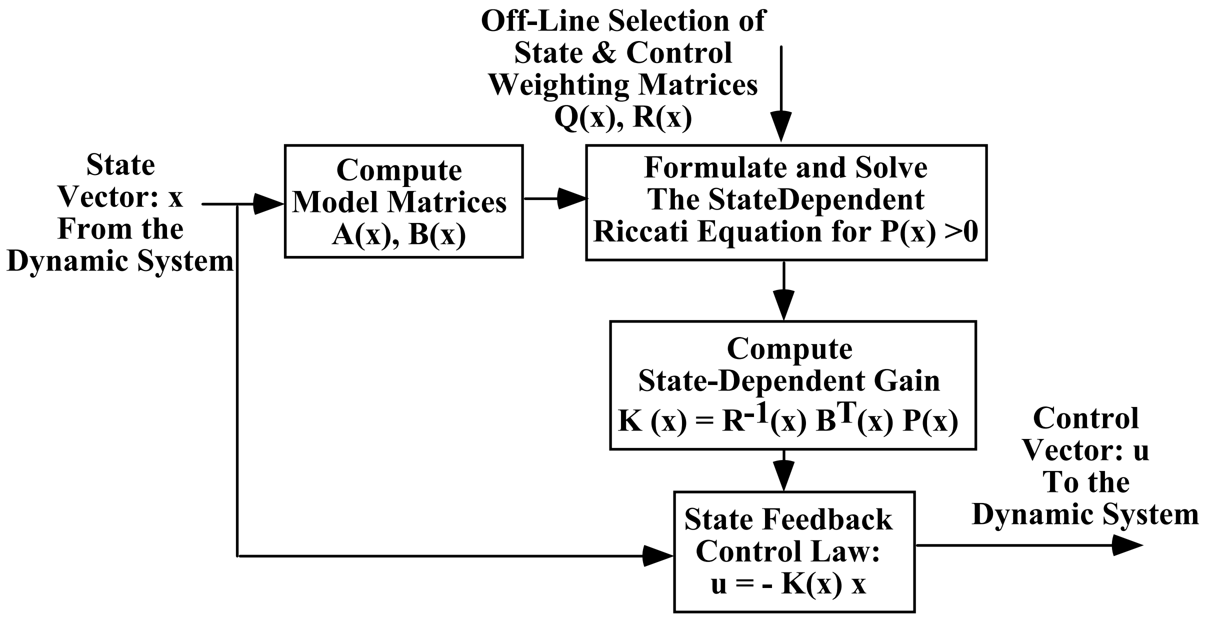

Off-Line Selection of State & Control Weighting Matrices Q(x), R(x)
State Vector: x From the Dynamic System
Compute Model Matrices A(x), B(x)
Formulate and Solve The StateDependent Riccati Equation for P(x) >0
Compute State-Dependent Gain K (x) = R-1(x) BT(x) P(x)
Control Vector: u To the Dynamic System
State Feedback Control Law: u = - K(x) x

Fig. 1. SDRE Control Law Computations

If the nonlinear system dynamics is given in symbolic form, it is possible to derive the SDC model using algebraic manipulations. However, in many practical problems where the system model is given as a computer simulation rather than an analytical model, numerical approaches will have to be employed to construct the SDC form. The next section will discuss a reliable numerical approach for obtaining the SDC form from a given numerical simulation model.

While the user may have a choice in how the system dynamics are transformed to the SDC form, the solution of the Riccati equation almost always requires numerical approaches. Closed-form solutions may be possible under special conditions. It most cases, numerical algorithms must be employed to find the solution to the Riccati equation. Several reliable algorithms have been advanced in the literature [16] for the solution of algebraic Riccati equations, four of which will be reviewed in Section 3. These algorithms require the use of numerical linear algebraic methods. Computer codes for implementing linear algebraic algorithms available in the public domain [19] formed the starting point for the present work. Highly optimized versions of these code components were developed to assemble the SDRE control software.

The following section will discuss a numerical approach for transforming a given simulation models of the dynamic system into the SDC form. Methods for the solving algebraic Riccati equations, together with an approximate assessment of the computational requirements with then be discussed in Section 4.

3. Numerical Approach for Obtaining the SDC Model

It will be assumed that the dynamic system is given in the standard form:

$$\dot{x} = f(x) + g(x)u$$

Here, *x* is the state and *u* is the control vector. Note that the control vector appears linearly in the system dynamics. If the control variables appear nonlinearly in the system dynamics, an input dynamic compensator can be introduced to transform the model into standard form. For instance, if the model is given in the form:

$$\dot{z} = h(z,v)$$

The user can introduce an input dynamic compensator of the form $\dot{v} = w$, such that the new dynamic system:

$$\begin{bmatrix} \dot{z} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} h(z,v) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w$$

is in the standard form with respect to the redefined control vector *w*.

Any input dynamic compensators can be employed, provided that the redefined control variable appears linearly in the dynamics. At any given value of *x*, finding the SDC form

$$\dot{x} = A(x)x + g(x)u$$

from the given nonlinear dynamic system requires the solution of a system of *n* equations:

$$A(x)x = f(x)$$

The matrix *A(x)* has $n \times n$ elements. Since only *n* equations are available, additional relations must be found to solve for the elements of *A(x).* If the vector nonlinear function *f(x)* is available in analytical form, algebraic manipulations can be used to construct the matrix *A(x).* Reference 2 has advanced several approaches for the construction of the *A(x)* matrix.

However, if the model is specified in the form of a computer simulation, a numerical approach will need to be set up. Instantaneous SDC parameterization can be obtained by evaluating the vector nonlinear function *f(x)* using a set of linearly independent *probe* vectors $\zeta_2, \ldots\ldots, \zeta_n$. As a practical matter, since the behavior of the nonlinearities in the neighborhood of the current system state are not explicitly known, it is wise to choose probe vectors that are close to the current state vector.

The probe vectors can be constructed by adding small magnitude perturbation vectors $\sigma_2, \sigma_3, \sigma_4, \ldots \sigma_n$, to the nominal state vector to yield a set of linearly independent vectors:

$$\zeta_2 = x + \sigma_2, \zeta_2 = x + \sigma_3, \zeta_3 = x + \sigma_4, \ldots\ldots, \zeta_n = x + \sigma_n$$

The nonlinear function $f(x)$ is next evaluated using these linearly independent vectors. Assemble the matrix equation

$$[f(x) \quad f(\zeta_2) \quad \ldots\ldots \quad f(\zeta_n)] = A(x)[x \quad \zeta_2 \quad \ldots\ldots\ldots \quad \zeta_n].$$

At any given value of x, this linear matrix equation can be solved for the elements of $A(x)$. Since the probe vectors and the state vector are linearly independent, this equation is well-conditioned, and can be solved using well-known linear algebraic methods.

Note that the foregoing computations will have to be carried out at every sample. The SDC matrix $A(x)$ from these computations can next be used to formulate and solve the SDRE control problem. As an aside, it is interesting to examine the relationship between the numerical construction of the SDC model and the conventional Taylor series approximation. If the perturbation vectors $\sigma_2, \sigma_3, \sigma_4, \ldots \sigma_n$, are small, it can be found that:

$$A \cong \frac{\partial f}{\partial x}, \ at\ x = 0$$

Note that this corresponds to the Taylor series linearization of the system dynamics about the origin. Thus, the present methodology for constructing the SDC model automatically reverts to Taylor series linearization of the system dynamics near the origin of the state space. For constant control influence matrix case, the present SDC parameterization scheme preserves the controllability properties of the dynamic system near the origin.

Since the only restriction on the probe vectors is that they be linearly independent, it is possible to construct an infinite variety of SDC parameterizations for a given dynamic system.

4. Efficient Algorithms for Solving Algebraic Riccati Equations

As discussed in Section 2, the main computational steps in the implementation of the SDRE technique are the computation of the SDC form of the system dynamics and the solution of a high-dimensional algebraic matrix-Riccati equation. Riccati equation solution methods discussed in the following subsections fall into two basic categories: direct and iterative. The following subsections will review four numerical techniques that have been found to be useful in applications [16 - 18]. These are:

1. Solution using Schur-decomposition of Hamiltonian matrix

2. Kleinman algorithm
3. Solution via discrete-time transformations
4. Solution via spectral factorization

Techniques not examined in this paper include doubling algorithm, Chandrasekhar algorithm, square root algorithm, information filter algorithm, and the matrix sign function algorithm. Some of these are discussed in Reference 16.

The methods (1) and (4) are direct methods while the others are iterative. Generally speaking, direct methods are computationally faster than iterative methods, especially in poorly conditioned problems and in cases where a good quality initial guess is not available. On the other hand, the computation and storage requirements for a direct method can be more than twice as much as that for an iterative method because the former operates on a $2n \times 2n$ Hamiltonian matrix for a Riccati equation of order n. Iterative techniques can outperform direct methods if the starting solution is close to the final solution.

4.1. Direct Solution Using the Hamiltonian Matrix

The solution of Riccati equation of order n can be obtained in terms of the solution of a linear Hamiltonian equation of order $2n$. The Hamiltonian matrix corresponding to the algebraic matrix Riccati equation is defined as:

$$M = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix}$$

With

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix},$$

The Hamiltonian matrix satisfies the condition:

$$M = JM^T J = -JM^T J^{-1}.$$

M is assumed to have no imaginary eigenvalues. Given the stabilizability and detectability of the dynamic system, if λ is an eigenvalue of the Hamiltonian matrix, so is $-\lambda$. Thus, there exists a real W such that

$$W^{-1}MW = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}$$

and Λ_1, Λ_2 are real Jordan matrices such that the real parts of all eigenvalues are respectively negative and positive. Under these assumptions, the solution to the matrix-Riccati equation can be written as

$$P = W_{21} W_{11}^{-1}.$$

Since $A - BR^{-1}B^T P = -W_{11}\Lambda_1 W_{11}^{-1}$, the eigenvalues of Λ_1 represent the closed loop system modes. Note that the matrix $W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$ consists of the eigenvectors of the Hamiltonian matrix M.

It is numerically preferable to use a Schur form

$$M = U^T \begin{bmatrix} L_{11} & L_{12} \\ 0 & L_{22} \end{bmatrix} U$$

where L is a Schur form of M with L_{11} possessing all negative real part eigenvalues, and the matrix U is orthogonal. Then, the solution to Riccati equation is given by

$$P = U_{12}^T (U_{11}^{-1})^T.$$

Since $A - BR^{-1}B^T P = -U_{11}^T L_{11} (U_{11}^T)^{-1}$, the eigenvalues of L_{11} represent the closed loop system modes. A flowchart of this direct solution methodology is given in Figure 2.

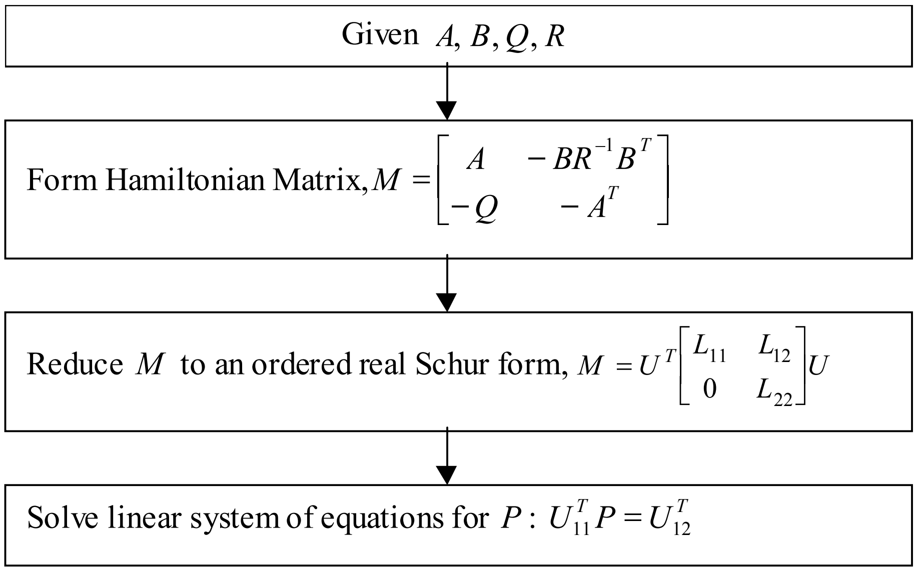

Given A, B, Q, R
Form Hamiltonian Matrix, $M = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix}$
Reduce M to an ordered real Schur form, $M = U^T \begin{bmatrix} L_{11} & L_{12} \\ 0 & L_{22} \end{bmatrix} U$
Solve linear system of equations for P: $U_{11}^T P = U_{12}^T$

Fig. 2. Direct Solution of the Riccati Equation Using Schur Decomposition of the Hamiltonian Matrix

As is evident from the above flowchart, two distinct steps are involved in the Schur decomposition approach. The first is the reduction of a $2n \times 2n$ Hamiltonian matrix to an ordered real Schur form; the second is the solution of an $n \times n$ linear matrix equation. An approximate estimate of operation counts required to solve the Riccati equation using the Schur method is given in Reference 16. That work indicates that the solution requires approximately $75n^3$ floating point operations (FLOPS), where n is the dimension of the system. Missile autopilots typically involve five states and three controls [22]. Thus, about 10,000 floating-point operations per second (10 MFLOPS) will be required to implement the SDRE technique for a missile autopilot at about 1 kHz sample rate. Implementing more advanced integrated guidance-control algorithms [23, 24] will require additional 2 MFLOPS.

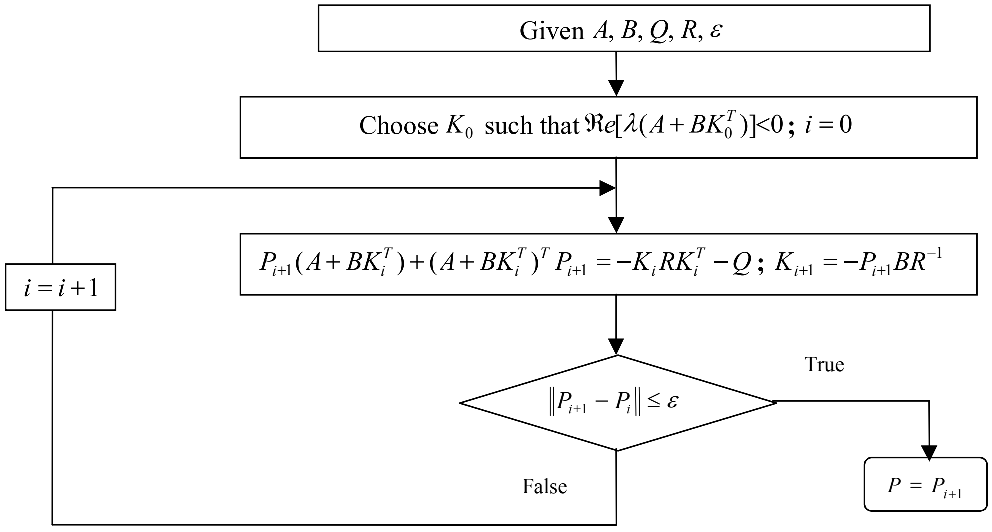
4.2. The Kleinman Algorithm

The Kleinman recursive algorithm [17, 18] uses an initial guess of the closed gain to obtain the solution to the algebraic Riccati equation. This algorithm is implemented as follows. Let K_0 be such that closed-loop system $(A + BK_0^T)$ has all eigenvalues with negative real parts. Define P_i, K_i recursively as:

$$P_i(A + BK_i^T) + (A + BK_i^T)^T P_i = -K_i R K_i^T - Q$$

$$K_{i+1} = -P_i B R^{-1}$$

Then $P_i \geq P_{i+1}$ and $\lim\limits_{i \to \infty} P_i = P$. Further, $\|P_{i+1} - P\| \leq c\|P_i - P\|^2$, $c > 0$, implying that the convergence of the algorithm is quadratic. Initial guess of the closed loop system gain can be generated using any available technique. For instance, pole placement can be used as the starting point. Alternately, the Schur algorithm discussed in the previous section can be used to obtain a first estimate P_0 of P, and then the algorithm can be initialized with $K_0 = -P_0 B R^{-1}$. A computational flowchart for the Kleinman algorithm is given in Figure 3.



Given A, B, Q, R, ε
Choose K_0 such that $\Re e[\lambda(A+BK_0^T)]<0$; $i=0$
$P_{i+1}(A+BK_i^T)+(A+BK_i^T)^T P_{i+1}=-K_i R K_i^T-Q$; $K_{i+1}=-P_{i+1}BR^{-1}$
$i=i+1$
$\|P_{i+1}-P_i\|\le\varepsilon$
True
False
$P = P_{i+1}$

Fig. 3. The Kleinman Algorithm

The Kleinman algorithm requires approximately 6n^3 floating point operations per iteration. Depending upon the initial guess, 10 or more iterations may be required to obtain the solution. Thus, the Riccati equation solution can be accomplished in 60n^3 floating point operations. A better quality initial guess can produce faster convergence of the algorithm. It has been shown [17, 18] that the algorithm will converge to the true solution, if the starting guess of the gain matrix provides a stable closed-loop system.

In the case of SDRE computations, even if a good quality guess is not available initially, the quality of initial guess will improve as the computations proceed.

4.3. Recursive Solution Using Discrete Time Transformations

The solution to the matrix Riccati equation can be obtained by setting up a discrete-time linear-quadratic problem for which the limiting solution of the discrete-time Riccati equation is P . Given the elements of the Riccati equation $A, B, Q = DD^T$, and R , define

$$F = (I + A)(I - A)^{-1}, \quad G = \sqrt{2}(I - A)^{-1} B$$

$$C = R + \frac{1}{2} G^T Q G, \quad D = \sqrt{2}(I - A^T)^{-1} Q (I - A)^{-1} B, \quad E = \frac{1}{2}(I + F^T) Q (I + F)$$

Then, the solution to the matrix Riccati equation can be obtained using the recursive formula:

$$\Phi_{i+1} = F^T \Phi_i F - (F^T \Phi_i G + D)(C + G^T \Phi_i G)^{-1} (F^T \Phi_i G + D)^T + E$$

Note that: $P = \lim_{i\to\infty} \Phi_i$.

The discrete-time transformation approach requires about 20n^3 operations per recursion. Although the number of recursions required depend strongly on the initial guess, typically 10 to 20 recursive steps are necessary for convergence. The main advantage of this technique is its simplicity.

4.4. Direct Solution Using Spectral Factorization

This technique is based on the spectral factorization of the Hamiltonian matrix M as $p(s)p(-s)$, where $p(s)$ has all stable eigenvalues. Then the solution to the Riccati equation P is uniquely defined by

$$p(M)\begin{bmatrix} I \\ \overline{P} \end{bmatrix} = 0$$

Since, $W^{-1}p(M)W = \begin{bmatrix} p(\Lambda_1) & 0 \\ 0 & p(\Lambda_2) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & p(\Lambda_2) \end{bmatrix}$,

$$p(M)\begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} = 0$$

The computational requirements for the spectral factorization approach are comparable to the direct solution approach described in 4.1.

The algorithms described in Sections 4.1 and 4.2 were coded and evaluated during the present research. Some of the issues involved in these implementations, together with computational speed evaluation on representative hardware are given in the following section.

5. Coding and Evaluation of *ARE* Solvers

The algorithms for Riccati equation solution described in Sections 4.1 and 4.2 formed the basis for the development of the software for implementing the SDRE control laws. The Schur direct method and the Kleinman iterative method were coded in ANSI-C. Basic numerical algebraic routines required for the implementation of these methods were obtained from the *LAPACK* [19] software package, and were optimized for improved performance.

LAPACK can solve systems of linear equations, linear least squares problems, eigenvalue problems and singular value problems. It can also handle many associated linear algebraic computations such as matrix factorizations or estimating condition numbers. *LAPACK* software supercedes the well-known *LINPACK* and *EISPACK* software packages. This software forms the basis for several well-known commercial computer programs. An interesting aspect of the

LAPACK package is that it is provided in both single and double precision versions. Moreover, it provides the same range of functionality for real and complex data.

The algebraic Riccati equation solution codes were developed using Microsoft *Visual C/C++* compiler, Version 6.0 for WindowsNT operating system. Each solution method is coded as a project.

Two versions of the software were assembled. The first version is for use in standalone speed evaluations, while the second version is for implementation in the MATLAB-Simulink® environment. The latter was used to compare the accuracy of the present implementations with those from MATLAB, and was also useful for algorithm evaluation on hardware platforms such as the dSPACE-DS1103 PowerPC 604e machine.

5.1. Performance Evaluation

The Riccati solution algorithms discussed in the foregoing section were next evaluated for accuracy and speed. The accuracy of Riccati solutions was ascertained by comparing the results from the present implementation to those obtained from MATLAB. The MATLAB command:

```
[X,L,M,RR]=care(f,g,q,r)
```

was used for this purpose. Controllable systems of several sizes were used in these evaluations. In every case, the direct solution method produced results that were indistinguishable from the MATLAB solution. The accuracy of the iterative solution is essentially a function of the specified tolerances. However, in all cases, five to six iterations were sufficient to produce results comparable to the direct method.

Next, to evaluate the speed of execution of these algorithms, a missile autopilot example containing six states and three control variables was selected. This problem size is comparable to the realistic guidance-control system for an advanced weapon system. Executable programs generated from the code described in the previous sections were then used to solve this problem on three different machines. The first two employed commercial versions of the WindowsNT operating system, while the third employed a proprietary operating system from dSPACE Inc. The WindowsNT evaluations were conducted on an Intel Pentium II 300 MHz workstation, and an Intel Pentium III 450 MHz workstation. The dSPACE proprietary operating system implementation was evaluated on a PowerPC 604e processor (DS1103) from dSPACE Inc [21]. The dSPACE PowerPC board is used extensively in automotive control system development. Speed comparisons for the Riccati solutions on these three machines are presented in Table 1.

Algorithm	Processor	Operating System	Execution Time
Schur Method	Pentium II, 300 MHz	Windows NT®	2.38 ms
Schur Method	Pentium III, 450 MHz	Windows NT	1.57 ms
Schur Method	PowerPC 604e	dSPACE Proprietary	1.56 ms
Kleinman Method	Pentium II, 300 MHz	Windows NT	1.46 ms
Kleinman Method	Pentium III, 450 MHz	Windows NT	0.91 ms
Kleinman Method	PowerPC 604e	dSPACE Proprietary	0.476 ms

Table 1. Execution Times for the Two Riccati Solution Methods

Computing times are given for one full-cycle of these algorithms. The computing time for the Kleinman method is given for three iterations. Note that the computational times deliver speeds far in excess of those required for implementing high-performance missile flight control systems. Since the main computational burden in implementing the SDRE technique is in the solution of the algebraic Riccati equation, these speed evaluations demonstrate the feasibility of implementing the SDRE technique on commercial, off-the-shelf processors in real-time. Moreover, analysis of the C code has revealed that it may be possible to gain an additional 25% speed improvement by further refining the code.

6. Conclusions

The present research was motivated by the application potential of the SDRE nonlinear control technique in practical nonlinear control problems. The state dependent Riccati equation (SDRE) technique is a recently developed methodology for designing control laws and estimation algorithms for general nonlinear dynamic systems. Although its potential is well recognized, the industry acceptance of the technique has been slow. The main reasons for this are: a) Unlike linear control technology, the SDRE approach requires advanced numerical methods for its implementation, which are not currently available off-the-shelf, b) the perception that this technique may not be computationally feasible for real-time implementation on commercial off-the-shelf processors.

The main objective of the present research was to address these two issues. The following tasks were accomplished during the Phase I research.

1. Developed C language-implementations of two algorithms for the solution of algebraic Riccati equations that form the central component of the SDRE design technique. The first approach is a direct method based on Schur decomposition of the Hamiltonian matrix. The

second approach is an iterative algorithm that refines an initial guess of the Riccati solution using the Kleinman algorithm. These software packages use extensively re-worked code components from the public-domain linear algebra package *LAPACK*. Results from both these algorithms have been compared with results from software packages such as MATLAB.

2. Solution speed of these algorithms has been assessed on Intel Pentium II, 300 MHz and Pentium III 450 MHz computers running the WindowsNT operating system, and on a real-time Motorola 604e PowerPC board from dSPACE Inc. The dSPACE PowerPC board is used extensively in automotive control system development. The execution of these algorithms at speeds up to 2 kHz sample rates on problems of the size commonly encountered in missile flight control applications was then demonstrated on commercial-off-the-shelf processors. This research also developed a numerical algorithm for the construction of dynamic system models in state dependent form from a given computer simulation.

The present research has demonstrated the feasibility of implementing the SDRE technique in real-time using commercial, off-the-shelf computers. The computational accuracy of these implementations has been shown to be comparable to other commercial software packages.

References

[1] Cloutier, J. R., "Adaptive Matched Augmented Proportional Navigation", *Proceedings of the AIAA Missile Sciences Conference*, Monterey, CA, November 1994.

[2] Cloutier, J. R., D'Souza, C. N., and Mracek, C. P., "Nonlinear Regulation and Nonlinear H∞ Control Via the State-Dependent Riccati Equation Technique", *Proceedings of the International Conference on Nonlinear Problems in Aviation and Aerospace*, Daytona Beach, FL, May 1996

[3] Mracek, C.P. and Cloutier, J. R., "Missile Longitudinal Autopilot Design using the State Dependent Riccati Equation Method", *Proceedings of the 1997 American Control Conference*, June 4 - 6, Albuquerque, NM.

[4] Cloutier, J. R., "State-Dependent Riccati Equation Techniques: An Overview", *Proceedings of the 1997 American Control Conference*, June 4 - 6, Albuquerque, NM.

[5] Venturcom® Web site: http://www.vci.com/prod_serv/nt/rtx/index.html".

[6] Intel® Web site: "http://developer.intel.com/technology/itj/q11998/articles/art_2d.htm".

[7] Volckmar, F., “Model-Based Controllers for the Process Industries now Supports Multi-Model Capability”, *Instrumentation and Automation News*, Vol. 46, No. 10, October 1998, pp. 87.

[8] Richalet, J., Rault, A., Testud, J. L., and Papon, J., "Model Predictive Heuristic Control," *Automatica*, V14, 1978, pp. 413-428

[9] Richalet, J., "General Principles of Scenario Predictive Control Techniques," *Proceedings of the 1980 American Control Conference*, San Francisco, CA, June 1980, FA9-A

[10] Morari, M., and Zafiriou, E., *Robust Process Control*, Prentice Hall, Englewood Cliffs, NJ, 1989

[11] Prett, D. M., and Gillette, R. D., "Optimization and Constrained Multivariable Control of a Catalytic Cracking Unit," *Proceedings of the 1980 American Control Conference, San Francisco*, CA, June 1980, WP5-C

[12] Mehra, R. K., Rouhani, R., and Praly, L., “New Theoretical Developments in Multivariable Predictive Algorithmic Control," *Proceedings of the 1980 American Control Conference*, San Francisco, CA, June 1980, FA9-B

[13] Garcia, C. E., Prett, D. M., and Morari, M., "Model Predictive Control: Theory and Practice - A Survey," *Automatica*, V25 N3, May 1989, pp. 335-348.

[14] *The Control Handbook*, William S. Levine-Editor, CRC Press, Boca Raton, FL, 1996.

[15] Laub, A. J., “A Schur Method for Solving Algebraic Riccati Equations”, *IEEE Transactions on Automatic Control*, Vol. AC-24, No. 6, December 1979.

[16] Anderson, B. D. O., and Moore, J. B., *Optimal Control: Linear Quadratic Methods*, Prentice Hall, 1990.

[17] Kleinman, D. L., “On an Iterative Technique for Riccati Equation Computation”, IEEE Transactions on Automatic Control, Vol. AC-13, No. 1, February 1968, pp. 114-115.

[18] Anderson, B. D. O., “Second-order convergent algorithms for the steady-state Riccati equation”, *International Journal of Control*, Vol. 28, No. 2, 1978.

[19] Anderson, F., et al, *LAPACK User’s Guide*, Society for Industrial and Applied Mathematics(*SIAM*), Philadelphia, PA, August 1999. (http://www.netlib.org/lapack)

[20] Anon, *MATLAB User’s Manual*, The MathWorks, Inc., Natick, MA, 1998.

[21] Anon, *Solutions for Control-dSPACE Catalog 1999*, dSPACE Inc., 22260 Haggerty Road, Suite 120, Northville, MI 48167.

[22] Zarchan, P., *Tactical and Strategic Missile Guidance*, American Institute of Aeronautics and Astronautics, Reston, VA, 1997.

[23] Menon, P. K., Iragavarapu, V. R., and Ohlmeyer, E. J., "Integrated Design of Agile Missile Guidance and Control System", *1998 Missile Sciences Conference*, November 17-19, Monterey, CA.

[24] Menon, P. K., Dewell, L. D., and Sweriduk, G. D., "Integrated Guidance-Autopilot Designs for Anti-Jam Operation of Precision Munitions", Phase I SBIR Project *Final Report* being Prepared under Air Force Contract No F08630-99-C-0040, December 1999.