Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 14,3, 383-402(1998)

MÉTODOS NUMÉRICOS PARA LA RESOLUCIÓN DEL PROBLEMA LINEAL - CUADRÁTICO DE CONTROL ÓPTIMO

ENRIQUE S. QUINTANA ORTÍ*
VICENTE HERNÁNDEZ GARCÍA**
y
GREGORIO QUINTANA ORTÍ*

* *Departamento de Informática*
Universidad de Jaime I de Castellón
12071 Castellón, España
Tel.: + 34-964-345771; Fax: + 34-964-345848
E-mail: quintana@inf.uji.es.

** *Departamento de Sistemas Informáticos y Computación*
Universidad Politécnica de Valencia
46071 Valencia, España
Tel.: + 34-963-877000; Fax: + 34-963-877358
E-mail: vhernand@dsic.upv.es.

RESUMEN

El problema lineal-cuadrático de control óptimo aparece en el análisis y diseño de sistemas dinámicos lineales. La solución de este problema en el modelo de espacio de estados puede obtenerse resolviendo una ecuación matricial cuadrática: la ecuación algebraica de Riccati. En este trabajo se presentan diversos algoritmos para resolver ecuaciones matriciales de Riccati mediante los cuatro métodos más fiables: el método de Newton, el método de Schur, la función signo matricial y la función disco matricial. Los resultados experimentales comparan estos métodos desde el punto de vista de la precisión numérica y de la eficiencia.

Palabras clave:

Ecuación algebraica de Riccati, control óptimo, control realimentado, álgebra matricial, algoritmos por bloques.

NUMERICAL METHODS FOR THE SOLUTION OF THE LINEAR-QUADRATIC OPTIMAL CONTROL PROBLEM

SUMMARY

The linear-qaudratic optimal control problem arises in the analysis and design of dynamic linear systems. The solution to this problems can be computed in the state-space model by solving a quadratic matrix equation: the algebraic Riccati matrix equation. In this paper we

Recibido: Noviembre 1995

©Universitat Politècnica de Catalunya (España) ISSN 0213-1315

present numerical algorithms for solving Riccati matrix equations by means of four of the most reliable numeric methods: Newton's method, the Schur method, the matrix sign function and the matrix disk function. The experimental results compare these methods both from the point of view of efficiency and accuracy.

Key words:

Algebraic Riccati equations, optimal control, feedback control, matrix algebra, block-partitioned algorithms.

INTRODUCCIÓN

En los últimos años el análisis y diseño de sistemas dinámicos lineales (SDL) y, en general, los problemas de control se han convertido en un área de creciente interés investigador. Así, la ecuación algebraica de Riccati (EAR) es en la actualidad un método estándar numéricamente fiable para resolver el problema lineal-cuadrático de control óptimo[1]. Además, el desarrollo de las últimas generaciones de computadoras de altas prestaciones ha permitido la resolución de problemas de control de gran dimensión. En este sentido, en este trabajo se presenta un estudio experimental de diversos métodos numéricos para la resolucón de EAR con matrices coeficiente densas.

Consideremos el SDL continuo e invariante en el tiempo definido por

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad x(0) = x_0 \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

donde $A \in \mathbb{R}^{n\times n}$ es la matriz de estados, $B \in \mathbb{R}^{n\times m}$ la matriz de entradas (o controles) y $C \in \mathbb{R}^{p\times n}$ la matriz de salidas, $x(t)$, $u(t)$ y $y(t)$ los vectores de estados, entradas y salidas respectivamente, de dimensiones apropiadas.

La dimensión n del vector de estados $x(t)$ en general satisface $n >> m$, $n >> p$ y es habitualmente menor que unos pocos miles, en el caso de problemas de control densos[20].

Las ecuaciones en (1) se conocen como el modelo de espacio de estados del sistema. El estado del sistema x_0 en el instante inicial t_0 (sin pérdida de generalidad $t_0 = 0$) y la secuencia de entradas $u(t)$, $t = t_0, \ldots, t_f$ definen la secuencia de salidas $y(t)$, $t_0, \ldots, t_f$.

En un entorno real se utiliza el vector de entradas para satisfacer ciertos requirimientos en el comportamiento dinámico del sistema. Además, simultáneamente es deseable minimizar la cantidad de esfuerzo requerido. Este doble objetivo se puede formular como un problema de minimización sobre $u(t)$ del funcional cuadrático

$$J = \int_0^\infty \left(x^T(t)Qx(t) + u^T(t)Ru(t) \right) dt \tag{2}$$

donde $Q \in \mathbb{R}^{n\times n}$ es una matriz de pesos de estados, simétrica semidefinida positiva ($Q^T \geq 0$), $Q = G^TG$ una factorización de rango completo de Q y $R \in \mathbb{R}^{m\times m}$ una matriz de pesos de controles, simétrica definida positiva ($R = R^T > 0$) ver ref. 1. En algunos casos, por ejemplo en el problema de regulación de las salidas, Q y C están relacionadas por $Q = C^TC$.

En el problema lineal-cuadrático de control óptimo se desea calcular un control realimentado por el estado $u(t) = Fx(t)$, $F \in \mathbb{R}^{m\times n}$, tal que minimiza el funcional cuadrático J en (2) y la matriz de estados del sistema en lazo cerrado

$$\dot{x}(t) = (A + BF)x(t) \tag{3}$$

es estable, esto es, los valores propios de $(A + BF)$ tienen parte real negativa. Bajo ciertas condiciones, especificadas en el siguiente teorema, este problema tiene una única solución[17].

Definición 1

El SDL $\dot{x}(t) = Ax(t) + Bu(t)$ (o el par (A, B)) es estabilizable, si existe un control realimentado por el estado tal que el correspondiente sistema en lazo cerrado es estable[17].

Definición 2

El SDL $\dot{x}(t) = Ax(t)$, $y(t) = Cx(t)$ (o el par (A, C)) es detectable, si el sistema dual asociado $\dot{x}(t) = A^T x(t) + C^T u(t)$ es estabilizable[17].

Teorema 3

Si el par (A, B) es estabilizable y el par (A, C) es detectable, entonces existe un único control óptimo

$$u(t) = Fx(t), \qquad F = -R^{-1}B^T X \tag{4}$$

que minimiza J, donde $X \in \mathbb{R}^{n\times n}$ es la única solución simétrica, semidefinida positiva de la ecuación algebraica de Riccati en tiempo continuo

$$A^T X + XA + Q - XBR^{-1}B^T X = 0 \tag{5}$$

Además, bajo estas condiciones el sistema en lazo cerrado

$$\dot{x}(t) = (A + BF)x(t) \tag{6}$$

es estable.

El teorema anterior introduce un método para resolver el problema lineal-cuadrático de control óptimo basado en la ecuación algebraica de Riccati (5), puesto que una vez se conoce X, el control realimentado $u(t)$ se obtiene fácilmente de (4). La EAR es una de las aproximaciones numéricas más fiables para resolver este problema de control y puede aplicarse asimismo en SDL en tiempo continuo generalizados

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \qquad x(0) = x_0 \\ u(t) &= Cx(t) \end{aligned} \tag{7}$$

donde $E \in \mathbb{R}^{n \times n}$ puede ser singular[17] y SDL en tiempo discreto generalizados

$$\begin{aligned} Ex_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{8}$$

El problema lineal-cuadrático de control óptimo aparece en problemas de control óptimo H_2, control robusto y optimización H_∞, estabilización de SDL, filtrado óptimo y cálculo de descomposiciones de Kalman, etc.[17] EAR de gran dimensión aparecen por ejemplo en el control de grandes estructuras dinámicas[19]. En general, en problemas de mayor dimensión (varios miles de variables de estados), las matrices coeficiente que intervienen son dispersas. Este tipo de problemas presentan el inconveniente de que, aunque las matrices coeficiente son dispersas, la solución X es densa. Así pues, ninguno de los métodos estudiados en las en las siguientes secciones resulta apropiado para este tipo de problemas. En realidad, únicamente se conocen aproximaciones heurísticas para la resolución de problemas dispersos[23].

En este trabajo se presentan algoritmos de altas prestaciones para resolver EAR de tamaño moderado, donde intervienen matrices coeficiente densas, mediante cuatro de los métodos con mayor fiablidad numérica: el método de Newton, el método de Schur, la función signo matricial y la función disco matricial. Los algoritmos desarrollados utilizan intensivamente las librerías computacionales BLAS y LAPACK[2]. La librería BLAS dispone de una serie de núcleos computacionales básicos, que habitualmente son desarrollados y afinados por el propio fabricante del computador. La librería LAPACK contiene diversas rutinas del álgebra matricial y realiza un empleo eficiente de los recursos computacionales mediante la utilización de técnicas orientadas por bloques y rutinas del BLAS.

En las implementaciones se ha utilizado un nodo computacional de un multiprocesador con memoria compartida, el Silicon Graphics PowerChallenge. Este nodo computacional es el SGI MIPS RS8000.

El trabajo está estructurado del siguiente modo. En la sección siguiente se describen los principales métodos numéricos para la resolución de EAR en el modelo de espacio de estados. Posteriormente se presentan los resultados experimentales obtenidos en esta arquitectura. Finalmente, en la última sección se comentan las conclusiones del trabajo.

Métodos numéricos para la EAR

Existen cuatro tipos de métodos para resolver la EAR en el modelo de espacio de estados: el método de Newton, los métodos basados en los vectores de Schur, los métodos basados en la función signo matricial y los métodos basados en la función disco matricial[15].

El método de Newton es un algoritmo numéricamente estable, pero el coste de este método iterativo es bastante alto y su convergencia puede ser lenta[13]. Este método trabaja directamente con las matrices coeficiente del problema y en cada iteración se calcula una nueva aproximación a la solución de la EAR.

Por otra parte, los restantes tres tipos de métodos trabajan sobre la matriz $2n \times 2n$

$$\mathcal{H} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \tag{9}$$

asociada a la EAR. Esta matriz es hamiltoniana, puesto que

$$\bar{J}H = (\bar{J}H)^T, \qquad \bar{J} = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix} \tag{10}$$

donde I_n es la matriz identidad de orden n. Estos métodos calculan una base del subespacio invariante estable de $\mathcal{H}$ y, a partir de ésta, obtienen la solución de la EAR.

El método de Newton

El método de Newton (o iteración de Kleinmann) fue introducido en[13] como un método iterativo para resolver la EAR. Este método requiere una aproximación inicial a la solución del problema y su convergencia puede resultar lenta si esta aproximación está lejana a la solución exacta del problema. Sin embargo, éste es un método cuya estabilidad numérica ha sido probada. En consecuencia, y debido a su alto coste computacional, este método está considerado como un algoritmo eficiente para refinar las soluciones obtenidas mediante algún otro método.

El método de Newton se calcula una secuencia de matrices que, bajo ciertas condiciones, converge a la única solución semidefinida positiva de la EAR. Este algoritmo se obtiene del siguiente teorema[13]:

Teorema 4

Sea X_k, $k = 0, 1, 2, \ldots$, la única solución semidefinida positiva de la ecuación matricial lineal de Lyapunov en tiempo continuo

$$A_k^T X_k + X_k A_k + Q + F_k^T R F_k = 0 \tag{11}$$

donde

$$\begin{aligned} F_k &= -R^{-1}B^T X_{k-1}, \quad k = 1, 2, \ldots \\ A_k &= A + BF_k, \quad k = 0, 1, \ldots \end{aligned} \tag{12}$$

y con F_0 calculada tal que $A + BF_0$ es estable. Entonces

1. $0 \leq X \leq X_{k+1} \leq X_k \leq \ldots \leq X_0, \quad k = 0, 1, \ldots$
2. $\lim_{k\to\infty} X_k = X$
3. La convergencia de la secuencia de matrices es asintóticamente cuadrática, esto es, $\exists c'$ tal que $\|X_{k+1} - X\| \leq c'\|X_k - X\|^2, \quad \forall k = 0, 1, \ldots$

Puesto que el par de matrices (AB) es estabilizable (ver las condiciones para la existencia y unicidad de la solución requerida de la EAR), es posible encontrar una

matriz de realimentación F_0 tal que $A + BF_0$ es estable. Por ejemplo, el algoritmo de Bass permite encontrar este control de realimentación de estados[3]. Así, si la iteración se inicia en esta aproximación, la secuencia siempre converge a la solución requerida de la EAR. Además, si la ecuación matricial lineal de Lyapunov[11] se resuelve mediante algún método numéricamente estable[5,11], entonces se puede probar que el método de Newton es asimismo numéricamente estable[20].

La siguiente implementación del método de Newton ofrece una alta fiabilidad en presencia de errores de redondeo[6].

Algoritmo 1

1. Calcular una matriz de realimentación inicial F_0 tal que $A + BF_0$ es estable

2. Calcular la solución simétrica inicial X_0 resolviendo la ecuación matricial lineal de Lyapunov

$$(A + BF_0)^T X_0 + X_0(A + BF_0) + Q + F_0^T R F_0 = 0 \tag{13}$$

3. Para $k = 0, 1, 2, \ldots$ hasta convergencia o $k > \max_\text{iter}$

   a) $R_k = -(Q + A^T X_k + X_k A - X_k S X_k), \quad S = BR^{-1}B^T$

   b) Calcular la solución $\hat{X}_k$ de la ecuación matricial lineal de Lyapunov

$$(A - SX_k)^T \hat{X}_k + \hat{X}_k(A - SX_k) = R_k \tag{14}$$

   c) $X_{k=1} = X_k + \hat{X}_k$

   Fin para

La convergencia de esta iteración puede mejorarse mediante el uso de las técnicas de búsqueda lineal[6]. Básicamente, la siguiente solución X_{k+1} se calcula como

$$X_{k+1} = X_k + t_k \hat{X}_k \tag{15}$$

donde $0 < t_k \leq 2$ se escoge en cada iteración de modo que se minimiza

$$\begin{aligned} f(t) &= (1-t)^2 \cdot \text{traza}(R_k^2) + 2(1-t)t^2 \cdot \text{traza}(R_k T_k) + t^4 \cdot \text{traza}(T_k^2) \\ T_k &= \hat{X}_k S \hat{X}_k \end{aligned} \tag{16}$$

Esta búsqueda lineal exacta puede reducir considerablemente el número de iteraciones del método de Newton. El siguiente teorema muestra las propiedades del método de Newton modificado mediante la búsqueda lineal[6]:

Teorema 5

Consideremos el par (AB) controlable $X_0 = X_0^T$ tal que $A - SX_0$ es estable y $0 < t_k \leq 2$, $k \geq 0$, entonces las soluciones aproximadas X_k, $k \geq 0$ producidas mediante el método de Newton con búsqueda lineal satisfacen

1. $A - SX_k$ es estable
2. $\|R_{k+1}\|_F \leq \|R_k\|_F$
3. $\lim_{k \to \infty} R_k = 0$
4. $\lim_{k \to \infty} X_k = X$ y la convergencia es asintóticamente cuadrática
5. $\lim_{k \to \infty} t_k = 1$

Los resultados numéricos en la sección siguiente muestran la eficiencia del empleo de búsquedas lineales (Figura 1). Además, el coste computacional por iteración de este método puede reducirse explotando la estructura simétrica de algunas de las matrices que intervienen (S, R_k y X_k).

El método de Schur

El desarrollo del método de Schur para la resolución de la EAR fue una extensión natural de los métodos basados en el uso de vectores propios[14]. El método de Schur evita las dificultades numéricas asociadas al cálculo de los vectores propios trabajando sobre los vectores de Schur, que pueden ser calculados mediante un método numéricamente estable, el algoritmo iterativo QR[10].

Los métodos basados en los vectores de Schur calculan una base del subespacio invariante estable de $\mathcal{H}$ y entonces la solución de la EAR se obtiene a partir de un sistema algebraico lineal.

El método de Schur para resolver la EAR[14] es el siguiente:

Algoritmo 2

1. Calcular una transformación de semejanza, definida por una matriz ortogonal U_1 tal que reduce la matriz hamiltoniana asociada con la EAR a la forma real de Schur (forma triangular superior por bloques, con bloques diagonales de dimensión 1×1 ó 2×2)

$$U_1^T \begin{bmatrix} A & -S \\ -Q & -A^T \end{bmatrix} U_1 = \bar{T} = \begin{bmatrix} \bar{T}_{11} & \bar{T}_{12} \\ 0 & \bar{T}_{22} \end{bmatrix} \tag{17}$$

2. Calcular una transformación de semejanza definida por una matriz ortogonal U_2, que desplaza los valores propios de $\bar{T}$ con parte real negativa al bloque superior izquierdo de T

$$U_2^T \begin{bmatrix} \bar{T}_{11} & \bar{T}_{11} \\ 0 & \bar{T}_{22} \end{bmatrix} U_2 = T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \tag{18}$$

3. Si $V = U_1U_2$, entonces la solución de la EAR se calcula resolviendo el sistema lineal $XV_{11} = V_{21}$, donde V está patricionada en bloques de dimensión $n \times n$ del siguiente modo

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \tag{19}$$

Existen diversas modificaciones de las dos primeras etapas del método de Schur que intentan explotar la estructura hamiltoniana del problema, como el algoritmo QR hamiltoniano[8]. Sin embargo, este algoritmo presenta el inconveniente de que no se conoce ningún método para reducir una matriz hamiltoniana general a la forma Hessenberg-hamiltoniana mediante transformaciones simpléticas ortogonales. Esta reducción únicamente puede realizarse si la matriz S o la matriz Q (o ambas) son de rango uno.

Un método similar, el algoritmo SR, que está también basado en el uso de transformaciones simpléticas, y en consecuencia preserva la estructura del problema, se describe en la ref.[7]. Sin embargo, algunas de las transformaciones introducidas en este método no son ortogonales, y por tanto es posible introducir grandes errores de redondeo en la solución.

La función signo matricial

Consideremos la descomposición canónica de Jordan[10] de la matrix $\mathcal{H}$ definida como

$$\bar{J} = \tilde{X}^{-1}\mathcal{H}\tilde{X} = D + N \tag{20}$$

donde $\tilde{X}$ es una matriz de vectores propios y vectores principales de $\mathcal{H}$, $D = \mathrm{diag}(d_1, d_2, \ldots, d_{2n})$ una matriz diagonal que contiene los valores propios de $\mathcal{H}$ y N una matriz nilpotente ($\exists k\colon N^k = 0_{2n}$). La función signo matricial de $\mathcal{H}$ se define como

$$\begin{gathered} \mathrm{signo}(\mathcal{H}) = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} = \tilde{X}\mathrm{diag}(\alpha_i)\tilde{X}^{-1} \\ \alpha_i = \mathrm{signo}(d_i), \quad i = 1, 2, \ldots, 2n \end{gathered} \tag{21}$$

donde la función signo escalar está definida por

$$\mathrm{signo}(\alpha) = \begin{cases} +1 & \text{si } Re(\alpha) > 0 \\ -1 & \text{si } Re(\alpha) < 0 \\ \text{indefinida} & \text{si } Re(\alpha) = 0 \end{cases} \tag{22}$$

y $Re(\alpha)$ es la parte real de α.

A partir de signo $(\mathcal{H})$ se puede probar[24] que la solución de la EAR asociadada se obtiene resolviendo el sistema lineal de rango completo sobredeterminado

$$\begin{bmatrix} S_{12} \\ S_{22} + I_n \end{bmatrix} X = -\begin{bmatrix} S_{11} + I_n \\ S_{21} \end{bmatrix} \tag{23}$$

Iteraciones para la función signo matricial

En[24] se demuestra que, aplicando el método de Newton-Raphson a la ecuación matricial no lineal $\mathcal{H}^2 = I_{2n}$, se obtiene la siguiente iteración para la función signo matricial

$$W_{k+1} = \frac{1}{2}(W_k + W_k^{-1}), \qquad W_0 = \mathcal{H} \tag{24}$$

y la secuencia W_k, $k = 0, 1, 2, \ldots$, converge cuadráticamente a signo $\mathcal{H}$.

Una técnica de aceleración[9] resulta en el siguiente esquema iterativo clásico

$$W_{k+1} = (W_k + c^2 W_k^{-1})/(2c), \quad c = |\det(W_k)|^{1/2n}, \quad W_0 = \mathcal{H} \tag{25}$$

Esta técnica es fácilmente utilizable, mejora la convergencia y requiere un coste computacional adicional de orden menor.

Existen otros esquemas iterativos diferentes que pueden resultar más eficientes sobre computadoras de altas prestaciones debido al tipo de operaciones que los componen. Un nuevo esquema iterativo racional[18] resulta en una iteración "rica" en productos matriciales. El esquema iterativo racional obtenido es el siguiente:

$$W_{k+1} = W_k \sum_{i=1}^{m} (W_k^2 + \alpha_i^2 I_{2n})^{-1}/(m x_i), \quad W_0 = \mathcal{H} \tag{26}$$

donde

$$x_i = \frac{1}{2}\left(1 + \cos\left(\frac{\pi(2i-1)}{2m}\right)\right) \quad \text{y} \quad \alpha_i^2 = \frac{1}{x_i} - 1 > 0 \tag{27}$$

El mayor coste por iteración de este esquema puede compensarse en algunos problemas por su mayor convergencia, que depende del parámetro m.

Una aproximación diferente resulta en otra iteración "rica" en productos matriciales. Si $\|W_f^2 - I_{2n}\| < 1$, entonces el esquema iterativo de Newton-Schulz[12], comenzando en W_f, converge a signo (W_f)

$$W_{k+1} = \frac{1}{2} W_k (3 I_{2n} - W_k^2), \qquad k \geq f \tag{28}$$

Puesto que $W_0 = \mathcal{H}$ es hamiltoniana, puede fácilmente probarse que las matrices de la secuencia W_k, $k = 1, 2, \ldots$ calculadas mediante (25) y (26) son también hamiltonianas. En consecuencia, $Z_k = \bar{J} W_k$, $k = 0, 1, 2, \ldots$ ($\bar{J}$ definida en (10)) define una secuencia de matrices simétricas indefinidas y el esquema iterativo simétrico

$$\begin{aligned} Z_{k+1} &= \frac{1}{2}(\gamma_k Z_k + J(\gamma_k Z_k)^{-1} J), \qquad Z_0 = J\mathcal{H} \\ \gamma_k &= |\det(Z_k)|^{-1/2n} \end{aligned} \tag{29}$$

converge a J signo $(\mathcal{H})$. Así, este esquema iterativo únicamente requiere la inversión de matrices simétricas y presenta por tanto un menor coste computacional. La misma idea puede aplicarse al esquema iterativo racional (26).

En estos esquemas iterativos se pueden utilizar dos criterios de parada diferentes; por ejemplo, si ϵ es la precisión de la máquina y τ es un umbral de convergencia, las iteraciones pueden detenerse cuando

$$\|W_{k+1} - W_k\|_1 < n \cdot \epsilon \cdot \tau \tag{30}$$

o bien cuando

$$\|W_k^2 - I_{2n}\|_1 < n \cdot \epsilon \cdot \tau \tag{31}$$

La función disco matricial

Los algoritmos para la función disco matricial, que han sido desarrollados recientemente, permiten dividir el espectro (vapores propios) de un haz de matrices de modo que se obtiene un haz de matrices de menor dimensión, con una parte determinada del espectro del haz original. Para este propósito, los algoritmos calculan el subespacio deflectante apropiado. Sin embargo, si se requiere el espectro completo, estos métodos son demasiado costosos y menos precisos que la aproximación estándar basada en el algoritmo iterativo QZ[10].

La definición de la función disco matricial fue primeramente introducida en[24]. El algoritmo presentado en esta subsección fue desarrollado por Malyshev[16] y fue mejorado en la ref.[4] mediante la introducción de una técnica libre de operaciones de inversión explícitas. Una nueva mejora[25] evita las operaciones de inversión explícitas y además permite obtener los subespacios deflectantes izquierdo y derecho mediante una única iteración.

Cuando se desea resolver la EAR, sólo es necesario calcular el subespacio invariante derecho de la matriz hamiltoniana $\mathcal{H}$. Así, el algoritmo de división espectral que resuelve la EAR, basado en el cálculo de la función disco matricial, puede especificarse del siguiente modo[6]:

Algoritmo 3

1. Consideremos $A_0 = I_{2n} - \mathcal{H}$, $B_0 = I_{2n} + \mathcal{H}$ y τ un umbral de convergencia

2. Para $k = 0, 1, \ldots$ hasta convergencia o $k > \max_\text{iter}$

   a) Calcular la descomposición QR

$$\begin{bmatrix} B_k \\ -A_k \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} R_k \\ 0 \end{bmatrix} \tag{32}$$

   b) $A_{k+1} = Q_{12}^T A_k$

   c) $B_{k+1} = Q_{22}^T B_k$

   d) si $\|R_k - R_{k-1}\|_1 \leq n \cdot \epsilon \cdot \tau \|R_{k-1}\|_1$, entonces $p = k + 1$, fin de iteración

Fin para

3. Resolver para X el sistema lineal sobredeterminado

$$\begin{bmatrix} Z_{12} \\ Z_{22} \end{bmatrix} X = \begin{bmatrix} Z_{11} \\ Z_{21} \end{bmatrix}, \qquad A_p = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \tag{33}$$

El esquema iterativo en la etapa 2 separa el espectro de un haz de matrices alrededor del círculo unidad. Iterando sobre la transformación de Cayley del haz de matrices $(I_{2n}, \mathcal{H})$ (ver etapa 1), el algoritmo anterior separa el espectro de $\mathcal{H}$ a lo largo del eje imaginario y proporciona un subespacio invariante derecho de esta matriz. La solución de la EAR se obtiene entonces en la última etapa resolviendo el sistema lineal (33), como se describe en[6].

ESTUDIO EXPERIMENTAL

Algoritmos particionados por bloques y librerías computacionales

Durante los últimos años se ha experimentado un creciente avance en la velocidad de cálculo de los procesadores, medida en millones de operaciones en coma flotante por segundo (Mflops). En cambio, este incremento en las prestaciones de los procesadores no se ha visto acompañado por una reducción comparable en los tiempos de acceso a la memoria principal. En consecuencia, en los computadores actuales se ha recurrido al uso de antememorias, o memorias caché, más rápidas que la memoria principal, capaces de suministrar los datos a los procesadores a la suficiente velocidad.

La programación eficiente de estos computadores pasa pues por una utilización eficaz de estas antememorias. Resulta imprescindible, por ejemplo, minimizar el número de trasiegos de información entre la memoria principal y las antememorias (fallos de antememoria).

En la computación matricial esta programación eficiente se puede conseguir mediante técnicas de programación orientadas por bloques. La idea fundamental es utilizar al máximo los datos que están actualmente en la antememoria, antes de tener que traer nuevos datos desde la memoria principal.

En este sentido, la librería BLAS (*Basic Linear Algebra Subprograms*) define un conjunto de especificaciones de rutinas básicas del álgebra matricial cuya implementación se deja habitualmente en manos del propio fabricante de la máquina. Así, por ejemplo, existen implementaciones específicas de estas rutinas afinadas especialmente para procesadores de Intel, HP, SUN, Silicon Graphics, etc. Algunas de las operaciones matriciales disponibles en esta librería son el producto matricial, el producto matriz por vector, la resolución de sistemas triangulares lineales, etc.

La librería LAPACK (*Linear Algebra Package*) extiende la funcionalidad del BLAS a núcleos de computación matricial más complejos[2]. Por ejemplo, existen rutinas en esta librería para la resolución de sistemas lineales de ecuaciones, problemas de valores propios, problemas de valores singulares, etc. Esta librería recurre a la utilización de

técnicas orientadas por bloques y al empleo de rutinas de BLAS para conseguir una implementación eficiente de sus núcleos.

Resultados experimentales

En este apartado se presentan los resultados experimentales comparativos de los códigos en doble precisión para los métodos numéricos estudiados. El análisis experimental se realizó sobre un Silicon Graphics PowerChallenge, utilizando los núcleos computacionales BLAS proporcionados por el fabricante y la librería LAPACK 2.0[2].

El Silicon Graphics PowerChallenge es un multiprocesador con memoria compartida y 8 nodos computacionales del tipo SGI MIPS R8000. El MIPS R8000 es un procesador superescalar con tecnología RISC. La velocidad pico de computación de este procesador es de 360 Mflops. La memoria caché de este computador está estructurada en dos niveles. El primer nivel está compuesto por una caché de datos interna de 16 Kbytes destinada al almacenamiento de datos enteros. En el segundo nivel de caché se encuentra una memoria (o caché externa de datos) de 4 Mbytes, destinada a almacenar datos en coma flotante y los datos enteros. La memoria de esta arquitectura puede variar entre 64 Mbytes y 16 Gbytes. La precisión máquina[10] de esta arquitectura es $\epsilon \approx 2,2 \times 10^{-16}$.

Todos los experimentos se realizaron sobre un único procesador del Silicon Graphics PowerChallenge, utilizando las opciones de optimización adecuadas.

Para evaluar la eficiencia de los algoritmos se han utilizado dos problemas de control óptimo. El primero aparece en el control y posicionamiento de una cadena de vehículos. El segundo está relacionado con el análisis de sistemas circulantes.

El primer problema, que se denotará por CVAV (control de Vehículos de Alta Velocidad), aparece descrito en varios trabajos[9,14]. Si el número de vehículos que deben controlarse es q, entonces el tamaño del problema es $n = 2q - 1$. La matriz de estados está definida como

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & \dots & 0 & 0 & 0 \\ 0 & A_{22} & A_{23} & \dots & 0 & 0 & 0 \\ 0 & 0 & A_{33} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & A_{N-2,N-2} & A_{N-2,N-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & A_{N-1,N-1} & \hat{A} \\ 0 & 0 & 0 & \dots & 0 & 0 & -1 \end{bmatrix} \tag{34}$$

donde

$$A_{k,k} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \ 1 \le k \le N-1, \ A_{k,k+1} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}, \ 1 \le k \le N-2 \text{ y } \hat{A} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{35}$$

Las matrices S y Q están definidas por

$$S = \text{diag}\ (1,0,1,0,\dots,1,0,1) \quad \text{y} \quad Q = \text{diag}\ (0,10,0,10,\dots,10,0,10) \tag{36}$$

El segundo problema, que se denotará como MC (Matrices Circulantes), se describe en [9]. En este problema, las matrices se construyen como

$$A = \begin{bmatrix} -2, & 1 & \dots & 0 & 1 \\ 1 & -2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -2 & 1 \\ 1 & 0 & \dots & 1 & -2 \end{bmatrix} \tag{37}$$

y $Q = S = I_n$.

La ventaja principal de estos dos problemas es que es posible incrementar la dimensión de las matrices sin que por ello empeore notablemente el condicionamiento del problema. La Tabla I muestra la norma de Frobenious de la solución calculada X y el número de condición estimado C_R para ambos problemas. El número de condición de la EAR[20] se estima mediante

$$C_R = \frac{2\|A\|_F + \frac{\|Q\|_F}{\|X\|_F} + \|S\|_F\|X\|_F}{\text{sep}((A - SX)^T, -(A - SX))} \tag{38}$$

donde sep mide la distancia entre el espectro de dos matrices[2]. En lugar de la solución exacta de la EAR, en esta expresión se utiliza la solución aproximada obtenida mediante el método de Newton.

	CVAV		
n	63	127	255
$\|X\|_F$	3,78	5,37	7,62
C_R	142,10	253,86	404,12
	MC		
n	64	128	256
$\|X\|_F$	3,75	5,35	7,60
C_R	141,35	235,15	403,44

Tabla I. Valores de $\|X\|_F$ y C_R estimados mediante el método de Newton en el Silicon Graphics PowerChallenge

La precisión de la solución se evalúa mediante el residuo relativo

$$\frac{\|A^T\bar{X} + \bar{X}A + Q - \bar{X}BR^{-1}B^T\bar{X}\|_F}{\|X\|_F} \tag{39}$$

donde $\bar{X}$ es la proyección de Frobenious $\bar{X} = (X + X^T)/2$ de la solución calculada X.

Para el problema CVAV se presentan los resultados correspondientes a los sistemas de orden 63, 127, 255 y 511. Para el problema MC los tamaños son 64, 128, 256 y 512.

Los algoritmos desarrollados se denotan, siguiendo la metodología de la librería LAPACK como dyycrzz, donde la primera letra "d" indica que el algoritmo emplea aritmética en doble precisión. Las letras "yy" indican si el método explota la estructura hamiltoniana del problema (yy = "ha" si es así, o yy = "ge" si no se considera esta estructura). Las dos letras siguientes identifican el problema resuelto; en este caso "cr" denota la ecuación matricial de Riccati en tiempo continuo. Las dos últimas letras describen el método empleado y la variante: método de Newton, método de Schur, etc.

En los apartados siguientes se comparan brevemente las diferentes variantes desarrolladas de cada uno de los métodos utilizados. Una vez se identifican las variantes más eficientes de cada uno de estos métodos, en el último apartado se procede a comparar los métodos numéricos entre sí, tanto desde el punto de vista de la precisión como de la eficiencia.

El método de Newton

Para realizar este estudio experimental se han implementado las siguientes variantes del método de Newton descrito en el algoritmo 1:

- dgecrnw: Algoritmo 1
- dhacrnw: Aprovechamiento de la estructura del problema para reducir el coste
- dgecrnz: Empleo de búsquedas lineales exactas
- dhacrnz: Empleo de búsquedas lineales exactas y aprovechamiento de la estructura del problema para reducir el coste

La Figura 1 muestra los tiempos de ejecución de las diferentes variantes del método de Newton frente al tamaño del problema.

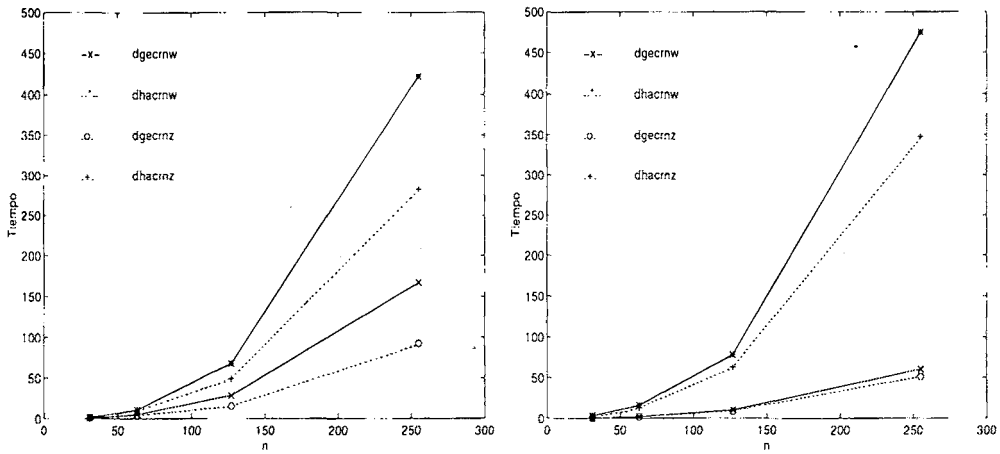

Figura 1. Tiempo de CPU (s) del método de Newton para la resolución de la EAR asociada a los problemas CVAV (izquierda) y MC (derecha) en el Silicon Graphics PowerChalenge

El menor tiempo de ejecución corresponde en ambos problemas al algoritmo dgecrnz.

Esto es debido a que el empleo de la búsqueda lineal[5,16] consigue reducir notablemente el número de iteraciones del método de Newton. Por ejemplo, en el problema CVAV la búsqueda lineal llega a dividir por dos el número de iteraciones necesarias, mientras que en el problema MC el impacto es menor. En cuanto al aprovechamiento de la estructura del problema, el ahorro computacional que supone considerar la simetría de algunas matrices del problema (algoritmos dhacrnw y dhacrnz) no compensa a la menor eficiencia de los núcleos computacionales que deben utilizarse en tal caso.

El método de Schur

Únicamente se ha desarrollado una implementación del método de Schur que corresponde al algoritmo 2:

- dgecrsc:

Otras variantes, como el algoritmo QR hamiltoniano o el algoritmo SR, no han sido implementadas debido a su falta de aplicación general o al uso de operaciones que pueden conducir a resultados no fiables.

La función signo matricial

Se han desarrollado diversas variantes del método de resolución de la EAR basado en la función signo matricial. Estas variantes difieren en el tipo de esquema iterativo que se emplea para el cálculo de la función signo matricial:

- dgecrsd: Esquema iterativo clásico
- dhacrsd: Esquema iterativo simétrico
- dgecrsr: Esquema iterativo racional
- dhacrsr: Esquema iterativo simétrico racional
- dgecrsm: Esquema iterativo mixto

El esquema iterativo mixto utiliza la iteración racional hasta que se asegura la convergencia de la iteración de Newton-Schulz y a partir de este instante se emplea ésta última.

La Figura 2 muestra los tiempos de ejecución de las diferentes variantes del método basado en la función signo matricial frente al tamaño del problema.

Esta figura muestra que en ambos problemas las mayores prestaciones se obtienen con el esquema iterativo simétrico dhacrsd seguido por el esquema clásico dgecrsd. Sin embargo, la diferencia en tiempos entre estos dos esquemas iterativos es pequeña y mucho menor que la que cabría esperar, puesto que el esquema iterativo simétrico

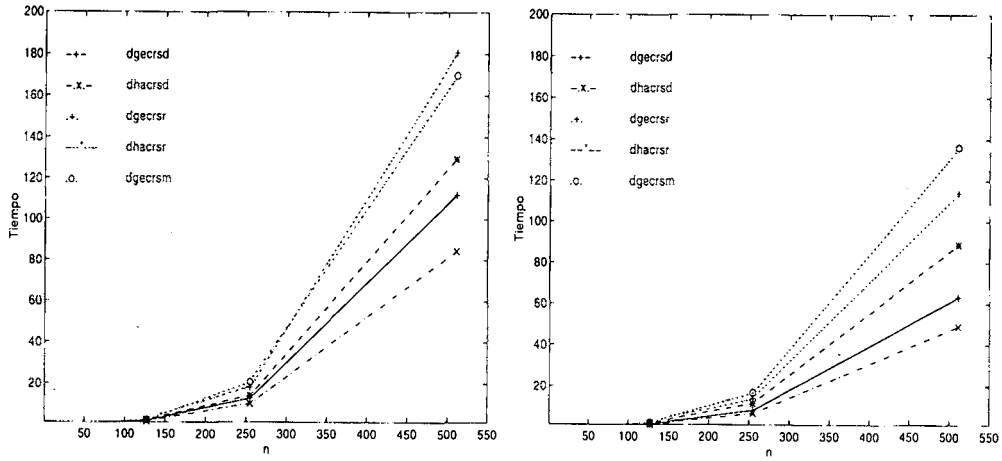

Figura 2. Tiempo de CPU (s) de la función signo matricial para la resolución de la EAR asociada a los problemas CVAV (izquierda) y MC (derecha) en el Silicon Graphics PowerChallenge

reduce a la mitad el coste del esquema iterativo clásico. Esto es debido a la utilización en el esquema iterativo simétrico de núcleos computacionales menos eficientes.

Esta misma situación se repite entre el esquema iterativo racional dgecrsr y el esquema iterativo simétrico racional dhacrsr, aunque ambos algoritmos obtienen tiempos de ejecución superiores a los anteriores. Por último, como muestra el problema MC, el esquema iterativo mixto puede resultar más eficiente en algunos casos.

La función disco matricial

En el estudio experimental se han desarrollado diversas implementaciones del método basado en la función disco matricial, que corresponde al algoritmo 3. Estas variantes difieren en el tipo de descomposición ortogonal[10] que se utiliza para resolver el sistema lineal sobredeterminado en la etapa 3. Experimentalmente se ha probado que las diferencias existentes son de orden menor y en consecuencia se ha escogido una variante del algoritmo 3, que emplea la factorización QR con pivotamiento de columnas en la tercera etapa.

- dgecrgs:

Comparación global

En primer lugar, en este estudio comparativo se analizan las variantes más eficientes de cada uno de los cuatro métodos:

- dgecrnz: Método de Newton con búsqueda lineal exacta

- dgecrsc: Método de Schur

- dhacrsd: Función signo matricial mediante el esquema iterativo simétrico

- dgecrgs: Función disco matricial

La Figura 3 muestra el comportamiento del tiempo de ejecución de los diferentes algoritmos desarrollados cuando varía el tamaño del problema.

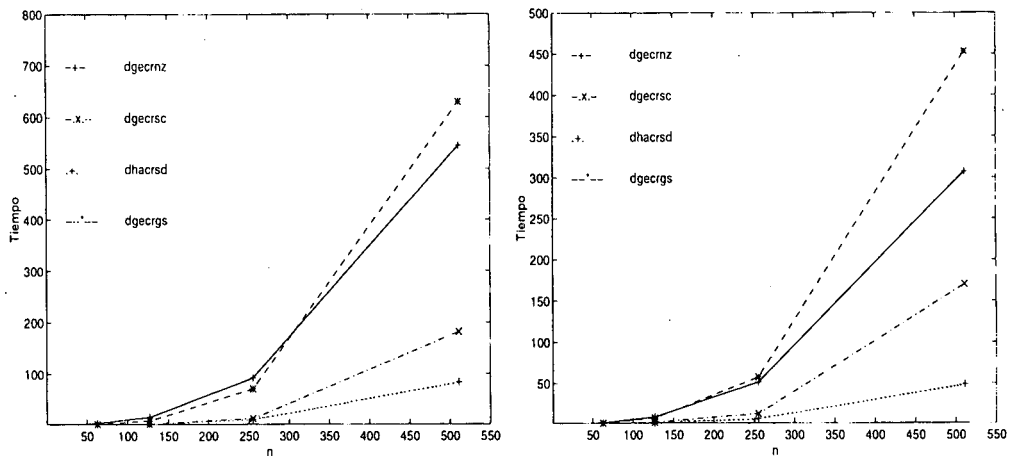

Figura 3. Tiempo de CPU (s) para la resolución de la EAR asociada a los problemas CVAV (izquierda) y MC (derecha) en el Silicon Graphics PowerChallenge

En ambas figuras el tiempo de CPU para resolver los problemas de mayor tamaño mediante la función disco matricial se obtuvo extrapolando los resultados disponibles.

La figura anterior muestra que el algoritmo más eficiente es el basado en el cálculo de la función signo matricial dhacrsd seguido del método de Schur dgecrsc. Estos dos algoritmos presentan además un coste considerablemente menor que los algoritmos basados en el método de Newton y la función disco matricial.

La función signo matricial requiere operaciones matriciales que resultan altamente eficientes sobre las arquitecturas actuales. Además, presenta un bajo coste por iteración y una rápida convergencia.

El método de Schur presenta un coste computacional teórico comparable a la función signo matricial. En cambio, el tipo de operaciones matriciales que requiere (algoritmo iterativo QR) no resultan tan eficientes sobre los computadores actuales.

El método de Newton presenta un alto coste por iteración y aunque las técnicas de búsqueda lineal reducen el coste global del algoritmo, éste todavía resulta notablemente más alto que en la función signo matricial o el método de Schur.

Por último, la función disco matricial presenta asimismo un alto coste por iteración y requiere además un elevado número de iteraciones, puesto que no se conoce ningún método de aceleración de la convergencia para este método.

A continuación se describen los resultados obtenidos utilizando el método de Newton como un algoritmo de refinamiento iterativo para afinar las soluciones calcualdas mediante los restantes métodos.

La Tabla II muestra el residuo relativo (ver (39)) de los métodos globales construidos mediante el método de Schur, la función signo matricial o la función disco matricial y refinados mediante el método de Newton. Además, se presenta el número de iteraciones de refinamiento necesarias para obtener una precisión similar a la obtenida directamente mediante el método de Newton.

	CVAV			
n	63	127	255	511
d_gecrnz	$0,21 \times 10^{-15}$ (8)	$0,23 \times 10^{-15}$ (9)	$0,23 \times 10^{-15}$ (9)	$0,21 \times 10^{-15}$ (10)
d_gecrsc	$0,33 \times 10^{-15}$ (1)	$0,31 \times 10^{-15}$ (1)	$0,33 \times 10^{-15}$ (1)	$0,35 \times 10^{-15}$ (1)
d_gecrsd	$0,30 \times 10^{-15}$ (1)	$0,32 \times 10^{-15}$ (1)	$0,30 \times 10^{-15}$ (1)	$0,32 \times 10^{-15}$ (1)
d_gecrgs	$0,30 \times 10^{-15}$ (1)	$0,34 \times 10^{-15}$ (1)	$0,37 \times 10^{-15}$ (1)	$0,32 \times 10^{-15}$ (1)
	MC			
n	64	128	256	512
d_gecrnz	$0,25 \times 10^{-15}$ (7)	$0,32 \times 10^{-15}$ (1)	$0,28 \times 10^{-15}$ (7)	$0,28 \times 10^{-15}$ (7)
d_gecrsc	$0,38 \times 10^{-15}$ (1)	$0,33 \times 10^{-15}$ (1)	$0,36 \times 10^{-15}$ (1)	$0,33 \times 10^{-15}$ (1)
d_gecrsd	$0,36 \times 10^{-15}$ (1)	$0,35 \times 10^{-15}$ (1)	$0,35 \times 10^{-15}$ (1)	$0,35 \times 10^{-15}$ (1)
d_gecrgs	$0,44 \times 10^{-15}$ (1)	$0,39 \times 10^{-15}$ (1)	$0,34 \times 10^{-15}$ (1)	$0,34 \times 10^{-15}$ (1)

Tabla II. Error relativo y número de iteraciones del método de Newton (entre paréntesis)

La tabla anterior muestra que el método de Schur, la función signo matricial y la función disco matricial, refinados mediante el método de Newton, permiten obtener soluciones para las EAR estudiadas de precisión comparable a la obtenida mediante el método de Newton. Además, en estos problemas únicamente es necesario una iteración de refinado para obtener tal precisión.

CONCLUSIONES

En este trabjao se han presentado diversos algoritmos para la resolución del problema lineal-cuadrático de control óptimo para sistemas dinámicos lineales formulados en el modelo de espacio de estados.

Los métodos analizados (método de Newton, método de Schur y funciones signo y disco matricial) están basados en la resolución de la ecuación algebraica de Riccati. Estos métodos están considerados como numéricamente fiables, pues evitan las dificultades numéricas de los algoritmos basados en el modelo polinomial.

El análisis de los métodos incluye un amplio estudio experimental de las diferentes variantes de cada uno de los algoritmos desarrollados. Este análisis realiza una comparación de los algoritmos, tanto desde el punto de vista de las prestaciones como de la precisión de los resultados.

El método más eficiente es la función signo matricial, pues une a la eficiencia de las operaciones matriciales que emplea, un escaso coste por iteración y una rápida convergencia. Además, entre los diferentes esquemas iterativos, la utilización del esquema iterativo simétrico consigue reducir el coste de este método.

El método de Schur presenta un coste computacional ligeramente superior, pero tiene la ventaja, frente a la función signo matricial, de evitar dificultades numéricas si la matriz hamiltoniana es casi singular.

Frente a los anteriores algoritmos, el método de Newton presenta como ventaja principal su estabilidad numérica, aunque su coste por iteración resulta muy alto. El ejemplo de técnicas basadas en búsquedas lineales consigue acelerar la convergencia de este método. En cambio, en este caso no se obtiene beneficio alguno al explotar la estructura hamiltoniana del problema.

Por último, la función disco matricial requiere un alto coste por iteración que determina sus peores prestaciones. Sin embargo, este método puede tener interés en arquitecturas paralelas por el tipo de operaciones que requiere.

REFERENCIAS

1. B.D.O. Anderson y J.B. Moore, "*Linear Optimal Control*", Prentice-Hall, Englewood Cliffs, USA, (1971).
2. E. Anderson *et al.*, LAPACK, User's Guide, SIAM, (1992).
3. E.S. Armstrong, "An Extension of Bass's Algorithm for Stabilizing Linear Continuous Systems", *IEEE Trans. on Automatic Control*, Vol. **20**, pp. 153–154, (1975).
4. Z. Bai, J. Demmel y M. Gu, "Inverse Free Parallel Spectral Divide and Conquer Algorithms for Nonsymmetric Eigenproblems", Computer Science Division Report, UCB//CSD-94-793, University of California at Berkeley, (1994).
5. R. Bartels y G.W. Stewart, "Algorithm 432. The Solution of the Matrix Equation $AX - XB = C$", *Comm. of the ACM*, Vol. **15**, pp. 820–826, (1972).
6. P. Benner, "Contributions to the Numerical Solution of Algebraic Riccati Equations", Tesis Doctoral, Faculty of Mathematics, TU Chemnitz-Zwickau, (1997).
7. A. Bunse-Gerstner y V. Mehrmann, A Symplectic QR-like Algorithm for the Solution of the Real Algebraic Riccati Equation", *IEEE Trans. on Automatic Control*, Vol. **31**, pp. 1104–1113, (1986).
8. R. Byers, "A Hamiltonian QR Algorithm", *SIAM J. Scientific & Statistical Computing*, Vol. **7**, pp. 212–229, (1985).
9. R. Byers, "Solving the Algebraic Riccati Equation with the Matrix Sign Function", *Lin. Algebra & Its Appl*, Vol. **85**, pp. 267–279, (1987).
10. G.H. Golub y C.F. Van Loan, "*Matrix Computations*", The John Hopkins University Press, Baltimore, USA, (1989).
11. S.J. Hammarling, "Numerical Solution of the Stable, Non-Negative Definite Lyapunov Equation, *IMA J. Numerical Analysis*, Vol. **2**, pp. 303–323, (1982).
12. C. Kenney y A.J. Laub, "Rational Iterative Methods for the Matrix Sign Function", *SIAM J. Matrix Analysis & Appl.*, Vol. **12**, pp. 273–291, (1991).
13. D.L. Kleinmann, "On an Iterative Technique for Riccati Equation Computations", *IEEE Trans. Automatic Control*, Vol. **13**, pp. 114–115, (1968).
14. A.J. Laub, "A Schur Method for Solving Algebraic Riccati Equations, *IEEE Trans. Automatic Control*, Vol. **24**, pp. 913–921, (1979).

15. A.J. Laub, Invariant Subspace Methods for the Numerical Solution of Riccati Equations", En *The Riccati Equation*", S. Bittanti, A.J. Laub y J.C. Willems (Eds.), Chapter 7, Springer-Verlag, Berlin, (1991).
16. A.N. Malyshev, "Parallel Algorithm for Solving Some Spectral Problems of Lineal Algebra", *Linear Algebra & Its Appl*, Vol. **188-189**, pp. 489–520, (1993).
17. V. Mehrmann, "*The Autonomous Linear Quadratic Control Problem*", Springer-Verlag, Berlin, (1991).
18. P. Pandey, C. Kenney y A.J. Laub, "A Parallel Algorithm for the Matrix Sign Function", *Int. J. High Speed Computing*, Vol. **2**, pp. 181–191, (1990).
19. P.M. Papadopoulos *et al.*, "Optimal Control Study for the Space Station Solar Dynamic Power Module", *Proceedings of the 30th Conference on Decision and Control*, Brighton, UK, (1991).
20. P.H. Petkon, N.D. Christov y M.M. Konstantinov, "*Computational Methods for Linear Control Systems*", Prentice-Hall International Ltd., UK, (1991).
21. E.S. Quintana y V. Hernández, "Parallel Algorithms for Solving the Algebraic Riccati Equation via the Matrix Sign Function", *Proceedings of the 3rd IFAC Workshop on Algorithms and Architectures for Real-Time Control AARTC'95*, pp. 533–542, Ostende, (1995).
22. E.S. Quintana, " Algoritmos paralelos para resolver ecuaciones matriciales de Riccati en problemas de control", Tesis Doctoral, Dept. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, (1996).
23. S. Richter y A. Scottedward Hodel, "Homotopy Methods for the Solution of General Modified Algebraic Riccati Equations", *Conference on Decision and Control*, Honolulu, (1990).
24. J. Roberts, "Linear Model Reduction and Solution of the Algebraic Riccati Equation by the Use of the Sign Function", *Int. J. of Control*, Vol. **32**, pp. 677–687, (1980).
25. X. Sun y E.S. Quintana, "Spectral Division Methods for Block Generalized Schur Decompositions", Computer Science Dept., Duke University, Technical Report CS-1996-13, (1996).