SLICOT: Una Librería de Software Numérico Eficiente y Fiable para Problemas de Control con Interfaces para MATLAB

Vicente Hernández
Univ. Politécnica de Valencia (UPV), Depto. de Sistemas Informáticos y Computación, (DSIC)
Camino de Vera s/n, Valencia 46022. vhernand@dsic.upv.es

Ignacio Blanquer, Antonio Vidal, Enrique Arias
UPV-DSIC. {iblanque, avidal, earias}@dsic.upv.es

Resumen

El proyecto NICONET [7] (Numerics in Control Network) es una iniciativa financiada por el programa BRITE EURAM cuya finalidad es coordinar el desarrollo de una librería de software numérico (SLICOT) [8] para el análisis y diseño de sistemas de control. Este software es esencial para el diseño de sistemas de control asistidos por computador (CACSD). El objetivo principal de la librería SLICOT, implementada en Fortran-77, es incorporar algoritmos eficientes y fiables. La librería es de libre distribución, accesible en el servidor http://www.win.tue.nl/niconet/niconet.html *. Para facilitar su uso, sobre todo en el campo industrial, se están desarrollando gateways de todas las rutinas principales que formarán un conjunto de librerías mex con la funcionalidad completa de SLICOT. En este trabajo se describen las características principales de esta librería.*

Palabras Clave: Análisis y Diseño de Sistemas de Control, Computación de Altas Prestaciones. Librerías Numéricas para Problemas de Control

1 EL PROYECTO NICONET

En el ámbito del proyecto NICONET (Numerics in Control Network, BRITE EURAM BRRT-CT97-5040) [7] se está desarrollando una librería de software numérico para problemas de control (SLICOT: Sotfware Library In COntrol Theory) [8]. Esta librería contiene actualmente casi 300 rutinas para la resolución de numerosos problemas de análisis y diseño. El objetivo que ha perseguido la implementación de todas estas rutinas es la eficiencia y la fiabilidad.

El proyecto NICONET persigue los siguientes objetivos:

1. Estimular la colaboración entre especialistas en control y en computación numérica.

2. Establecer una amplia base de desarrolladores de software y usuarios industriales, con la finalidad de que los usuarios orienten a los desarrolladores hacia los problemas que son de interés para la industria.

3. Integrar dos librerías de control existentes RASP y SLICOT en una librería SLICOT, más madura, completándola en aquellas áreas más relevantes para la industria (reducción de modelos, identificación basada en subespacios, control robusto, sistemas no lineales, ...).

4. Hacer que el software de libre distribución esté disponible y accesible por world wide web.

5. Integrar el software en plataformas CACSD existentes a través del desarrollo de interfaces apropiados.

6. Validar la librería desde el punto de vista de su aplicabilidad, utilizando casos de test propuestos por usuarios industriales.

7. Crear una librería de benchmark que permita contrastar las rutinas desarrolladas.

8. Prestar especial atención tanto a problemas de gran dimensión como requerimientos de tiempo real. Se pretende desarrollar una versión paralela de la librería SLICOT que permita alcanzar altas prestaciones, tanto en arquitecturas monoprocesador como en redes de procesadores.

9. Establecer mecanismos flexibles de intercambio de información en el campo de los algoritmos numéricos para problemas de control. Esta tarea está siendo llevada a cabo a través de las NICONET newsletters (que contienen información relativa a NICONET y SLICOT). Organizar workshops orientados al campo de los algoritmos y el software de control, así como tutoriales y cursos de entrenamiento en el manejo de SLICOT.

En el proyecto NICONET participan 9 universidades, 2 centros de investigación y 6 empresas Europeas. Por parte española participan la Universidad Politécnica de Valencia (DSIC) y OMRON España.

Como se ha comentado anteriormente, uno de los objetivos del proyecto NICONET, es promocionar el uso de esta librería en el campo industrial. Para

ello, se están desarrollando versiones de esta librería integradas con herramientas dotadas de interfaces amigables, como MATLAB [24] o SciLAB [25]. Por otro lado y con el objetivo de disponer de 'demostradores' del potencial de SLICOT, se están abordando diversos problemas industriales reales en los que esta librería puede proporcionar importantes ventajas.

El proyecto NICONET se divide en cinco paquetes de trabajo que se corresponden con las áreas:

I. Herramientas Numéricas Básicas.
II. Reducción de Modelos.
III. Identificación Basada en Subespacios.
IV. Control Robusto.
V. Sistemas No Lineales de Control.

En cada una de esas áreas se están desarrollando una serie de toolboxes, descritas en diferentes reports, para resolver varios tipos de problemas. Estos reports pueden ser obtenidas en el servidor ftp: ftp://ftp.esat.kuleuven.ac.be/pub/WGS/REPORTS.

2 LAS TOOLBOXES DE SLICOT

Hace tiempo que MATLAB es ampliamente conocido, aceptado y utilizado por la comunidad científica. Esta aceptación se está extendiendo a aplicaciones industriales debido a su entorno amigable y facilidad de uso. De esta manera, software estandarizado escrito en C o Fortran, se esta relegando a un segundo plano, puesto que es complicado de utilizar por usuarios no expertos. Sin embargo, el software desarrollado en librerías de bajo nivel escritas en C o Fortran es, en general, más robusto y eficiente que los ficheros .m de MATLAB. Usando los ficheros pasarela MEX de MATLAB, es posible incorporar software más eficiente en el entorno MATLAB. Es por ello que los partners de NICONET, estan desarrollando estas funciones pasarela para las principales rutinas de SLICOT.

Hay que destacar que los núcleos computacionales de MATLAB están basados en LINPACK [28] y EISPACK [27], mientras que los núcleos computacionales de SLICOT están basados en LAPACK [1] y utilizan BLAS-3 [26]. Esta es una de las razones por la que los algoritmos que proporciona SLICOT son más eficientes. Los algoritmos se encuentran orientados a bloques aprovechando de esta manera la jerarquía de memorias presente en las arquitecturas de los computadores actuales.

2.1 HERRAMIENTAS NUMÉRICAS BÁSICAS

En este área [9], [10], [12], [13] se han generado hasta el momento 5 librerías mex para la resolución de ecuaciones matriciales algebraicas de Riccati, Lyapunov, Stein y Sylvester, incluyendo versiones en tiempo continuo y discreto, ordinarias y generalizadas, definidas y semidefinidas. También incluye rutinas para la obtención de la parte controlable y observable de un sistema lineal de control, realizaciones minimales, realizaciones balanceadas, forma real de Schur y descomposiciones espectrales.

En la figura 1 se presenta un esquema con los ficheros .m de MATLAB desarrollados para llamar a la función MEX ARESOL, que a su vez realiza llamadas a rutinas de SLICOT para resolver las siguientes ecuaciones de Riccati:

$$0 = Q + A^T X + XA - (L + XB)R^{-1}(L + XB)^T, \quad (1)$$

$$0 = Q + A^T X + XA - XGX, \quad (2)$$

$$X = Q + A^T XA -, \\ -(L + A^T XB)(R + B^T XB)^{-1}(L + A^T XB)^T, \quad (3)$$

$$X = Q + A^T X(I + GX)^{-1} A. \quad (4)$$

Los ficheros .m que llaman a la rutina MEX ARESOL son los siguientes:

- **slcaregs:** resuelve la ecuación de Riccati en tiempo continuo (1) y calcula la matriz de realimentación correspondiente al regulador lineal-cuadrático de control óptimo, aplicando el método de Schur generalizado al haz de matrices extendido asociado al problema.
- **slcares:** resuelve las ecuaciones de Riccati en tiempo continuo (1) y (2), y los problemas de control asociados mediante el método de Schur.
- **sldaregs:** resuelve la ecuación de Riccati en tiempo discreto (3), así como la matriz de ganancia asociada al regulador lineal-cuadrático de control óptimo, utilizando el método de Schur generalizado sobre haces de matrices extendidos.
- **sldares:** resuelve las ecuaciones de Riccati en tiempo discreto (3) y (4), y los problemas de control asociados aplicando el método de Schur a las matrices extendidas asociadas.
- **sldaregsv:** resuelve las ecuaciones de Riccati en tiempo discreto (3) y (4), y los problemas de control asociados aplicando el método de Schur generalizado a los haces de matrices extendidos asociados.

En la figura 2 se muestra el conjunto de ficheros .m de MATLAB implementados para llamar a la función MEX GENLEQ que sirve de interface con rutinas de SLICOT para resolver ecuaciones matriciales lineales generalizadas de la forma:

$$AX - YB = C \ , \ DX - YE = F, \quad (5)$$

$$A^T X + D^T Y = C \ , \ XB^T - YE^T = -F, \quad (6)$$

$$AXB - DXE = C, \quad (7)$$

$$op(A)^T Xop(E) - op(E)^T Xop(A) = C, \quad (8)$$

$$op(A)^T Xop(A) - op(E)^T Xop(E) = C, \quad (9)$$

$$op(A)^T (op(X)^T op(X))op(E) + \\ + op(E)^T (op(X)^T op(X))op(A) = -op(C)^T op(C), \quad (10)$$

$$op(A)^T (op(X)^T op(X))op(A) + \\ + op(E)^T (op(X)^T op(X))op(E) = -op(C)^T op(C), \quad (11)$$

donde $op(A) = A$ o A^T.

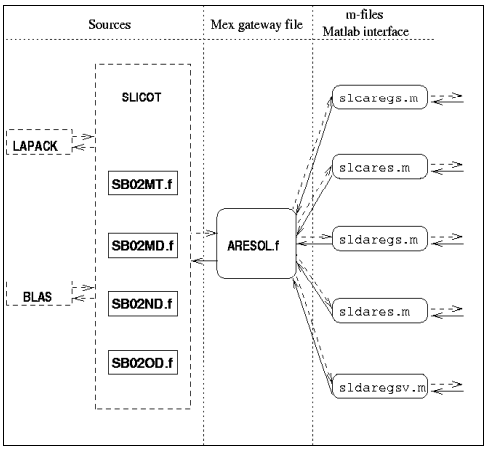

Figura 1. Rutina MEX ARESOL que incluye llamadas a las rutinas SB02MT.f, SB02MD.f, SB02ND.f y SB02OD.f de SLICOT para la resolución de ecuaciones de Riccati.

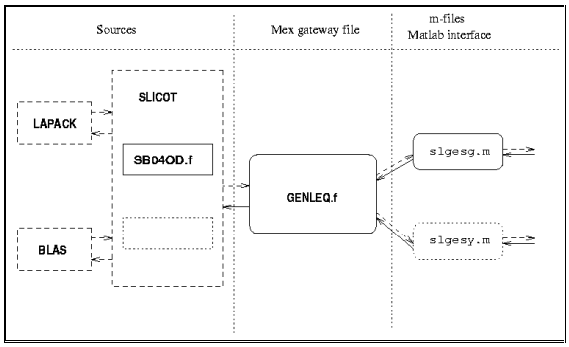

Figura 2. Rutina MEX GENLEQ que incluye llamadas a las rutinas SB04OD.f, SB03AD.f y SB03BD.f de SLICOT para la resolución de ecuaciones matrices lineales generalizadas.

Los ficheros .m implementados son:

- **slgesg:** resuelve el par de ecuaciones generalizadas de Sylvester (5) y (6).
- **slgesy:** resuelve la ecuación generalizada de Sylvester (7).
- **slgely:** resuelve la ecuación de Lyapunov generalizada en tiempo continuo (8).
- **slgest:** resuelve la ecuación de Lyapunov generalizada en tiempo discreto (9).
- **slgsly:** calcula el factor de Cholesky de la solución de la ecuación de Lyapunov generalizada estable en tiempo continuo (10).
- **slgsst:** calcula el factor de Cholesky de la solución de la ecuación de Lyapunov generalizada estable en tiempo discreto (11).

La rutina MEX LINMEQ.f combina las rutinas SB03MD.f, SB03OD.f, SB04MD.f y SB04ND.f para la resolución de ecuaciones matriciales lineales ordinarias. La estructura de llamadas de la rutina LINMEQ.f puede verse en la figura 3.

El tipo de ecuaciones matriciales lineales ordinarias que resuelve son:

$$op(A)X + Xop(B) = C, \quad (12)$$

$$op(A)Xop(B) + X = C, \quad (13)$$

$$op(A)^T X + Xop(A) = C, \quad (14)$$

$$op(A)^T Xop(A) - X = C, \quad (15)$$

$$op(A)^T (op(X)^T op(X)) + (op(X)^T op(X))op(A) = \\ = -op(C)^T op(C), \quad (16)$$

$$op(A)^T (op(X)^T op(X))op(A) - op(X)^T op(X) = \\ = -op(C)^T op(C). \quad (17)$$

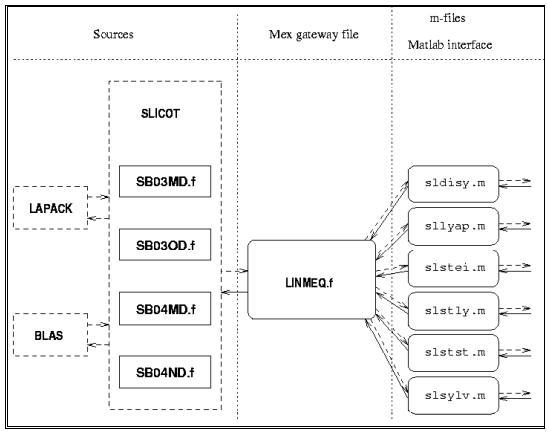

Figura 3. Rutina MEX LINMEQ que incluye llamadas a las rutinas SLICOT SB03MD.f, SB03OD.f, SB04MD.f y SB04ND.f para la resolución de ecuaciones matriciales lineales ordinarias.

Los ficheros .m desarrollados son:

- **sldisy:** resuelve la ecuación en tiempo discreto tipo Sylvester (13).
- **sllyap:** resuelve la ecuación de Lyapunov en tiempo continuo (14).
- **slstei:** resuelve la ecuación de Stein (15).
- **slstly:** resuelve la ecuación estable de Lyapunov en tiempo continuo (16).
- **slstst:** resuelve la ecuación estable de Lyapunov en tiempo discreto (17).

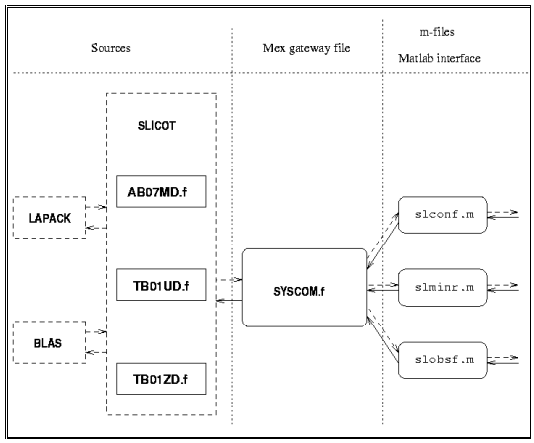

Figura 4. Rutina MEX SYSCOM que incluye llamadas a las rutinas AB07MD.f, TB01PD.f, TB01UD.f y TB01ZD.f para el cálculo de las formas controlables y observables así como los subsistemas que proporcionan una realización minimal.

En la Figura 4 se muestra la estructura de la rutina MEX SYSCOM.f que combina las rutinas de SLICOT AB07MD.f, TB01PD.f, TB01UD.f y TB01ZD.f para el cálculo de la forma controlable, la forma observable, y realizaciones minimales. Los ficheros .m implementados para estas tareas son:

- **slconf**: calcula un sistema transformado controlable y verifica la controlabilidad.
- **slobsf**: calcula un sistema transformado observable y verifica la observabilidad.
- **slminr**: calcula un subsistema que es realización minimal.

Por último, en la Figura 5 se muestra la rutina MEX SYSTRA.f que combina las rutinas de SLICOT TB01ID.f, TB01KD.f, TB01LD.f y TB01WD.f para el balanceo de sistemas o transformar matrices de estado a formas triangulares. Los ficheros .m asociados son:

- **slsbal**: balancea un sistema dado por las matrices *(A,B,C)*.
- **slsdec**: transforma la matriz de estados a una matriz en forma diagonal a bloques con los valores propios en el orden requerido.
- **slsorsf**: transforma la matriz de estados a una matriz en forma real de Schur con los valores propios en el orden requerido.
- **slsrsf**: transforma la matriz de estados a una matriz en forma real de Schur.

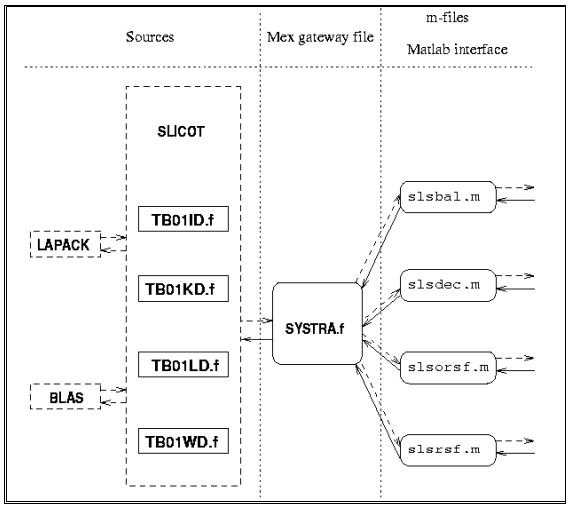

Figura 5. Rutina MEX SYSTRA que incluye llamadas a las rutinas de SLICOT TB01ID.f, TB01KD.f, TB01LD.f y TB01WD.f para el balanceado de sistemas o transformar matrices de estado a formas triangulares.

2.2 REDUCCIÓN DE MODELOS

La reducción de modelos es de fundamental importancia en muchas aplicaciones de modelado y control. Muchos modelos obtenidos, por ejemplo, a partir de una aproximación por elementos finitos, son de un alto orden, puesto que de esta manera se alcanza una mejor representación del problema físico subyacente. La reducción del modelo resulta particularmente importante para obtener una descripción de menor orden que contenga el comportamiento dinámico esencial y permita aplicar métodos de síntesis de control disponibles. La reducción del controlador es de considerable interés para la aplicabilidad de varias de las técnicas de síntesis de control actuales. Sin esta reducción, los controladores serían muy complicados sin apenas utilidad práctica.

La librería RASP-MODRED desarrolladas por DLR [23] dispone de software numérico robusto para los problemas computacionales que aparecen en el contexto de reducción de modelos. Esta librería ha sido incorporada a SLICOT [11] e integrada en el entorno MATLAB. La librería RASP-MODRED incluye implementaciones de nuevos algoritmos de reducción de modelos con métodos basados en técnicas balanceadas (perturbaciones singulares o aproximación en la norma de Hankel) para la reducción de modelos en sistemas lineales de control estables, inestables, en tiempo continuo y en tiempo discreto.

SLICOT contiene un total de 12 rutinas de reducción de modelos estandarizadas, completamente documentadas y testeadas. Incluye además una rutina MEX denominada SYSRED.f que sirve de interfaz entre las rutinas de SLICOT y los ficheros .m de MATLAB desarrollados para un uso más sencillo. Estos ficheros .m son los siguientes:

- **bta:** utiliza el método de "balancing-free square-root balance and truncate" en combinación con descomposición espectral estable/inestable.
- **btabal:** utiliza el método "square-root balance and truncate" en combinación con descomposición espectral estable/inestable.
- **bta_cf:** utiliza el método "balancing-free square-root balance and truncate" en combinación con factorizaciones estables por la derecha/izquierda coprimas.
- **btabal_cf:** utiliza el método "square-root balance and truncate" en combinación con factorizaciones estables por la derecha/izquierda coprimas.
- **spa:** utiliza el método de "balancing-free square-root single perturbation approximation" en combinación con descomposición espectral estable/inestable.
- **spabal:** utiliza el método "square-root single perturbation approximation" en combinación con descomposición espectral estable/inestable.
- **spa_cf:** utiliza el método "balancing-free square-root single perturbation approximati-

on" en combinación con factorizaciones estables por la derecha/izquierda coprimas.
- **btabal_cf:** utiliza el método "square-root single perturbation approximation" en combinación con factorizaciones estables por la derecha/izquierda coprimas.
- **hna:** utiliza el método basado en aproximación de la norma de Hankel en combinación con descomposición espectral estable/inestable.

2.3 IDENTIFICACIÓN BASADA EN SUB-ESPACIOS

La identificación de sistemas está adquiriendo una creciente importancia en optimización de la calidad, coste económico e impacto ambiental de productos producidos por varias ramas de la industria, tales como industria de procesos, automóvil, aeroespacial y electrónica de consumo.

El desarrollo, durante los últimos años, de técnicas de identificación basadas en subespacios ha supuesto una mejora significativa en la identificación de modelos de procesos industriales. Estos métodos ofrecen ventajas computacionales típicas de los esquemas de optimización basados en métodos de mínimos cuadrados (LS) o en la transformada rápida de Fourier (FFT), que incluyen aspectos tales como:

- Robustez numérica y eficiencia.
- No requerir valores iniciales precisos tanto de parámetros como de la estructura del modelo.

Así pues, desde la aparición de los primeros resultados sobre identificación de sistemas basada en subespacios, se han desarrollado diferentes algoritmos. Los más comúnmente usados son MOESP (Multivariable Output Error State Space), N4SID (Numerical Algorithms for Subspace State Space System Identification), CVA (Canonical Variable Analysis). A pesar de que los tres siguen la misma filosofía, su implementación práctica varía considerablemente. Es por ello que se ha generado una rutina mex [6], [14] que combina los tres métodos diferentes para la identificación de sistemas basada en subespacios. Esta "toolbox" incluye:

- Identificación de sistemas lineales invariantes en el tiempo (en lazo abierto y cerrado).
- Identificación de sistemas no lineales.

2.4 CONTROL ROBUSTO

Estabilidad robusta así como prestaciones robustas son dos de los temas más importantes en el diseño de sistemas de control. Se requiere que los sistemas de control funcionen de manera segura en presencia de parámetros variables, perturbaciones externas, ruidos no medidos y dinámicas desconocidas del modelo. El objetivo del control robusto consiste en diseñar controladores tales que el sistema en bucle cerrado sea estable y que mantenga unos niveles de prestaciones establecidas, en presencia de tales incertidumbres.

La mayoría de los sistemas de control en la industria (tales como industria aeroespacial, procesos químicos e industrias manufactureras), son inherentemente multivariables y no lineales. Una aproximación al problema no lineal es utilizar aproximaciones lineales en torno al punto de funcionamiento. Un controlador apropiado debe ser diseñado para cada uno de los modelos linealizados. El controlador diseñado ha de ser lo suficientemente robusto para cubrir un amplia gama de funcionamientos de la planta real. Esto incluye a las perturbacionaes introducidas en el proceso de cálculo del mismo.

En las últimas dos decadas, ha habido un considerable desarrollo en el ámbito del control robusto. Una gran variedad de paquetes software comerciales, como MATLAB, introducen toolboxes que tratan este tipo de controladores. Sin embargo, se han destinado más a un ambito docente que aplicado debido, principalmente a una baja eficiencia computacional, una pobre precisión numérica o a la incapacidad de manejo de problemas de gran dimensión o de resolver problemas industriales de diseño.

En el ámbito del proyecto NICONET, se han desarrollado toolboxes que incluyen el problema Lineal Cuadrático estándar en el espacio de estados, H_2 y H_∞ en tiempo discreto, así como para la asignación de polos y el cálculo de las normas H_2 y H_∞. Se ha generado una rutina mex [5] con diversas funciones que permiten el uso sencillo de las toolboxes anteriormente porpuestas por parte de un usuario no experto.

2.5 SISTEMAS DE CONTROL NO LINEALES

Una de las tareas del proyecto NICONET está consistiendo en dotar a SLICOT de capacidad para la simulación, control e identificación de Sistemas No Lineales de Control.

Un sistema de control no lineal puede expresarse bajo la forma de una ecuación diferencial ordinaria (ODE)

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), p(t), t), \\ y(t) &= g(x(t), u(t), p(t), t), \end{aligned} \tag{18}$$

o de una ecuación diferencial algebraica (DAE)

$$0 = f(\dot{x}(t), x(t), u(t), p(t), t),$$
$$y(t) = g(\dot{x}(t), x(t), u(t), p(t), t), \quad (19)$$

donde *x(t)*, *u(t)*, *y(t)* y *p(t)* representan los vectores de estados, entradas, salidas y parámetros del sistema.

Existe un volumen importante de software de dominio público para la resolución de ODEs y DAEs. Para favorecer el uso de estos paquetes, dentro del proyecto NICONET se está desarrollando un interfaz estándar que permite cambiar de integrador fácilmente, sin cambiar la sintaxis.

Los paquetes que se han considerado en este proyecto son, en el caso de ODEs:

ODEPACK [29]. Con sus versiones LSODE, LSODA, LSODES, permite resolver sistemas densos, banda o dispersos de la forma

$$\dot{x}(t) = F(x(t), t), \quad (20)$$

utilizando el método de Adams y diferencias finitas regresivas.

RADAU5 [30]**.** Mediante métodos implícitos de Runge-Kutta resuelve sistemas de la forma

$$M\dot{x}(t) = G(x(t), t), \quad (21)$$

DASSL [31] (denso) y **DASPK** [32] (disperso). Resuelve sistemas de la forma

$$F(\dot{x}(t), x(t), t) = 0 \quad (22)$$

mediante métodos implícitos (diferencias regresivas).

GELDA [33]**.** Resuelve sistemas de la forma

$$E(t)\dot{x}(t) = A(t)x(t) + F(t), \quad (23)$$

con coeficientes variables en el tiempo.

Todos ellos permiten resolver ODEs expresadas de la forma:

$$\dot{x}(t) = F(x(t), t), \quad (24)$$

En el caso de DAEs, los paquetes considerados son:

ODEPACK. Contiene los solvers LSODI y LSOIBT, que resuelven mediante métodos implícitos sistemas expresados como

$$M\dot{x}(t) = G(x(t), t), \quad (25)$$

mientras que **RADAU5**, **DASSL**, **DASPK** y **GELDA** resuelven problemas expresados de la forma (21)-(23).
En este caso resulta imposible definir un único interfaz de entrada para todas las rutinas y al mismo tiempo mantener íntegro el potencial de todos los paquetes. En el caso de DASSL, por ejemplo, es posible tratar DAEs de la forma (19), lo que no es posible realizar con LSODI, que tiene limitado el modelo de DAEs a (25). Si bien analíticamente son equivalentes, no es posible encontrar una forma automática de codificar ambas rutinas de la misma forma. La ventaja de usar el interfaz estándar es que los valores de los parámetros y su posición en los arrays es común para todos los paquetes, lo que facilita el uso de diferentes solvers.

Por tanto, la aproximación propuesta consiste en utilizar tres interfaces muy similares que corresponden con los tres modelos diferentes expuestos anteriormente

$$\begin{aligned} &F(\dot{x}(t), x(t), t) = 0 \\ &M\dot{x}(t) = G(x(t), t), \\ &E(t)\dot{x}(t) = A(t)x(t) + F(t). \end{aligned} \quad (26)$$

El interfaz para ODEs permite resolver los problemas especificados en (18) mediante la rutina:

```
[X,RPAR,IPAR,Y,IERR] =
interfaceODE(ISOLVER,RPAR,IPAR,ODEDER,ODEOU
T,JACFX,JACFU,JACFP,X,U,P,TINI,TOUT,RTOL,AT
OL,DWORK,IWORK);
```

en donde se definen los parámetros del sistema, las funciones que caracterizan el sistema (ODEDER evalúa la derivada del vector de estados xdot(t), mientras que ODEOUT evalua el vector de salida y(t)), las expresiones de los jacobianos para una mejor convergencia (JACFX, JACFU, JACFP), y el resto de las condiciones iniciales. Para trabajar con las funciones, *f* y *g*, que caracterizan el sistema a resolver, se deben desarrollar una serie de rutinas que permitan evaluar el valor de la función dados los valores de los elementos de entrada. Estas rutinas se codifican como ficheros '.m' independientes y son llamadas a través de la librería 'mex'. Se identifican mediante una cadena que define su nombre. La Figura 6 muestra un esquema de la estructura del protocolo que siguen dichas funciones.

La sintaxis de las funciones que definen el comportamiento del sistema es la siguiente:

```
[xdot,ierr]= odeder(rpar,ipar,x,u,p,t);

[y,ierr]= odeout(rpar,ipar,x,xdot,u,p,t);

[dfdx,ierr]= jacfx(rpar,ipar,x,t);

[dfdu,ierr]= jacfu(rpar,ipar,x,u,t);
```

```
[dfdp,ierr]= jacfp(rpar,ipar,x,p,t).
```

Análogamente, el interfaz para DAEs permite resolver los problemas especificados en (19) mediante la rutina:

```
[X,RPAR,IPAR,Y,IERR] =
interfaceDAE(ISOLVER,RPAR,IPAR,DAEDF,DAEDA,
DAEDE,DAEOUT,JACFX,JACFU,JACFP,X,XDOT,U,P,T
INI,TOUT,RTOL,ATOL,DWORK,IWORK);

[F,ierr]=daedf(rpar,ipar,y,ydot,u,p,t);

[A,ierr]= daeda(rpar,ipar,y,u,p,t);

[E,ierr]= daede(rpar,ipar,p,t).
```

Las rutinas de interfaz de los diferentes paquetes no sólo ocultan la sintaxis propia de las rutinas de llamada a cada uno de los paquetes, sino que además convierten los valores de cada uno de los parámetros de forma que resulten, en la medida de lo posible, compatibles. La salida de resultados está también estandarizada y se realiza a través de los vectores de parámetros (real y entero).

De esta forma, es posible cambiar de solver con tan sólo cambiar el valor del parámetro ISOLVER, y en su caso, suministrar la información extra que pueda requerir un paquete y no haya sido especificada.

2.3 P-SLICOT: COMPUTACIÓN EN PARALELO DESDE MATLAB

Uno de los objetivos de la segunda mitad del proyecto NICONET es el desarrollo de una versión paralela (basada en paso de mensajes) de la librería SLICOT [3]. Esta librería dispondrá también del correspondiente interfaz MATLAB.

La versión paralela de SLICOT estará basada en ScaLAPACK [2], siguiendo las pautas marcadas en la implementación de la versión secuencial basada en LAPACK [1]. La librería ScaLAPACK utiliza como núcleo de comunicaciones la librería BLACS [15], que a su vez utiliza PVM (Parallel Virtual Machine) [4] o MPI (Message Passing Interface) [16].

El uso de ScaLAPACK permite utilizar de forma eficiente tanto grandes computadores paralelos de memoria distribuida (HP-SPP2000, IBM SP2, CRAY T3E,...) o compartida (SGI Power Challenge, Origin 2000, ...) como clusters de PCs o estaciones de trabajo.

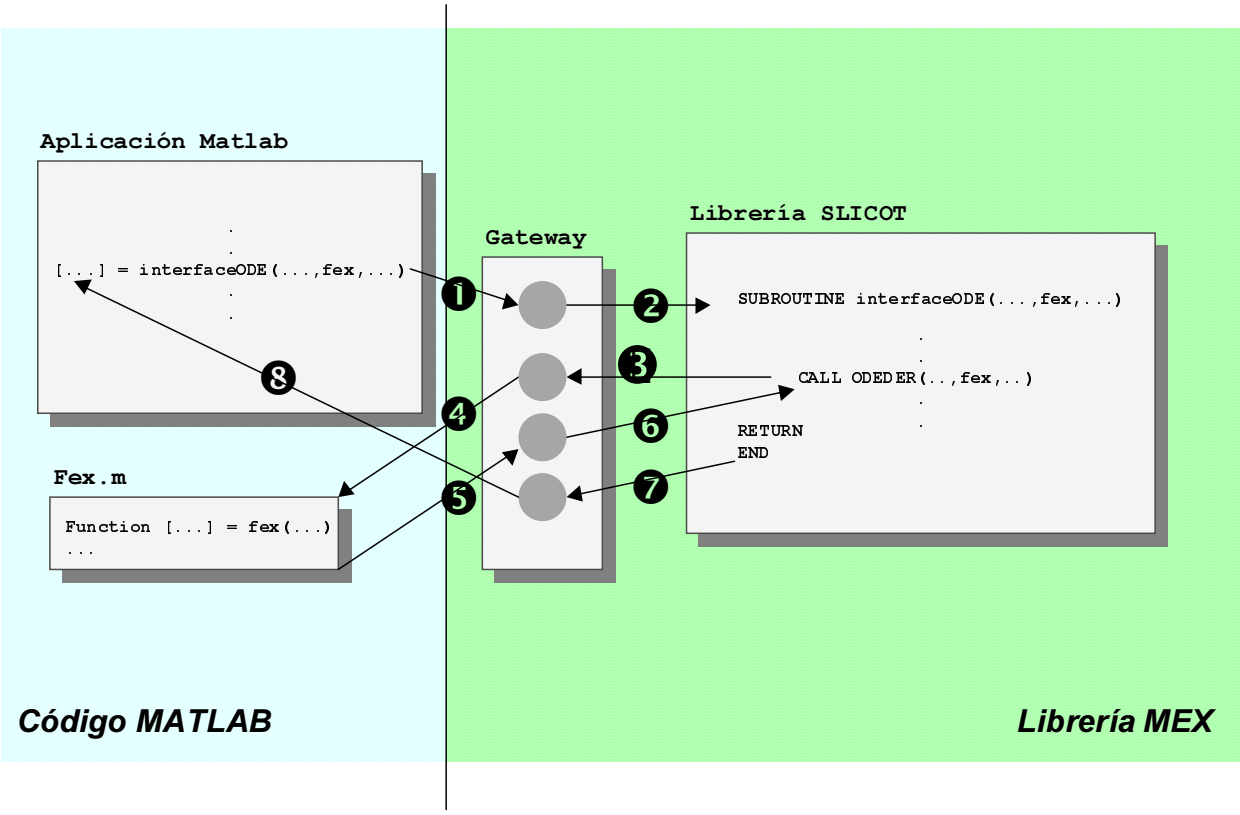

Figura 6. Esquema de funcionamiento entre los componentes SLICOT, MEX y MATLAB. La llamada desde matlab (1) accede a la librería mex (2) que a su vez necesita ejecutar código MATLAB proporcionado por el usuario (3), a través del gateway (4) . De este modo se accede a la rutina correspondiente cuyos resultados (5) se retornan a la librería compilada (6). Los resultados finales se transfieren al código MATLAB inicial (8) a través del gateway (7).

Para la implementación de una toolbox paralela para MATLAB se pretende utilizar la librería PVM como base para las comunicaciones. Otras librerías más extendidas como MPI resultan altamente complejas de integrar, dado su carácter de paralelismo simétrico y la imposibilidad de crear tareas de forma dinámica. No obstante, el interfaz MPI2, todavía no completado, pero con toda seguridad llamado a sustituir al estándar actual, prevée la incorporación de rutinas para gestión dinámica de tareas que haría totalmente factible su uso en este entorno.

3 RESULTADOS EXPERIMENTALES

Los siguientes resultados ilustran las prestaciones (en términos de eficiencia, fiabilidad y precisión) que puede obtenerse mediante los componentes de SLICOT, descritos en [17]. Estos resultados se comparan con los obtenidos con las funciones de MATLAB incluídas en su núcleo básico de algoritmos matriciales o en la toolbox de control
Los resultados muestran que SLICOT es varias veces más rápido que MATLAB, con una precisión comparable o incluso superior.
Más aún, las rutinas de SLICOT utilizan menos memoria, ya que explotan totalmente las estructuras matriciales involucradas.
Los cálculos fueron realizados en una estación de trabajo SUN Ultra 2 Creator 2200 con 128 Mb. de RAM funcionando sobre SunOS 5.5.
Los datos de entrada han sido generados aleatoriamente sobre una distribución uniforme en el intervalo (0,1), variando el orden n del sistema.

Forma Controlable

La tabla 1 muestra el compartamiento de la rutina AB01ND que calcula la forma controlable de un sistema lineal en el espacio de estados, comparándola con la correspondiente función CTRBF de la toolbox de control de MATLAB.
Más información sobre la forma de calcular los residuos y otros detalles asociados puede encontrarse en [17].
La rutina SLICOT es hasta 60 veces más rápida que su análoga MATLAB en el caso de simple entrada y 10 veces en el caso de múltiple entrada

	Time		Relative residuals	
n	AB01ND	CTRBF	AB01ND	CTRBF
16	0.00	0.03	4.94e-16	4.04e-16
32	0.00	0.17	7.01e-16	9.30e-16
64	0.04	1.29	5.83e-16	1.23e-15
128	0.32	19.12	7.41e-16	3.58e-15

Table 1. Comparación entre AB01ND y CTRBF.

Ecuaciones de Lyapunov y Riccati

La tabla 2 muestra los resultados obtenidos por la rutina SLICOT SB03MD, para la resolución de ecuaciones de Lyapunov ordinarias en tiempo contínuo, y la función de MATLAB LYAP.

	Time		Relative residuals	
N	SB03MD	LYAP	SB03MD	LYAP
16	0.01	0.05	3.69e-15	3.59e-15
32	0.03	0.12	3.54e-15	8.96e-15
64	0.21	0.70	3.56e-15	1.96e-14
128	1.50	5.38	1.73e-14	2.86e-14

Tabla 2. Comparación entre SB03MD y LYAP.

La tabla 3 muestra los resultados obtenidos, en la resolución de ecuaciones de Ricatti en tiempo continuo, con la rutina de SLICOT SB02OD. Estos resultados se comparan con los obtenidos con la correspondiente rutina MATLAB CARE.

	Time		Relative residuals	
n	SB02OD	CARE	SB02OD	CARE
16	0.07	0.37	1.83e-14	3.81e-14
32	0.30	0.86	4.99e-14	1.13e-13
64	2.80	4.01	7.21e-12	2.10e-11
128	24.00	79.96	2.44e-11	6.72e-11

Tabla 3. Comparación entre SB02OD y CARE.

Hay que destacar que SLICOT retorna además estimaciones de los números de condición de los problemas mencionados, mientras que MATLAB no dispone aún de esta funcionalidad. Esta información es de vital importancia para la fiabilidad de los resultados obtenidos.

En las referencias [12][17][18] se pueden encontrar muchas más comparaciones entre SLICOT y MATLAB.

4 CONCLUSIONES

La librería SLICOT es una valiosa herramienta numérica a la hora de resolver problemas de control por sus prestaciones, fiabilidad, eficiencia, robustez, versatilidad y precisión. Constituye un núcleo computacional estable, rápido y muy completo que además dispone de interfaces con MATLAB y FORTRAN77 (que a su vez es compatible con otros lenguajes).

SLICOT funciona varias veces más rápido que MATLAB, a una precisión comparable o mejor, y además es fiable. Más aún, se explotan las estructuras matriciales presentes en los problemas de control, por lo que se requiere de menos memoria y, por consiguiente, permite abordar problemas de mayor dimensión. Además, SLICOT contiene rutinas adicionales que permiten resolver problemas aún no contemplados en MATLAB.

Por otro lado, es incluso posible alcanzar altas prestaciones si se utilizan versiones nativas de las librerías BLAS, que junto con LAPACK, constituyen los núcleos computacionales básicos de SLICOT, manteniendo las puertas abiertas para el uso de las arquitecturas de los computadores actuales, incluso de las arquitecturas paralelas.

Referencias

[1] A. Anderson, Z. Bai, C. Bischof, J. Demmel, et. al. (1995). "LAPACK Users' Guide". *SIAM, Philadelphia, Second Edition.*

[2] L. Blackford, J. Choi, A. Cleary, J. Demmel, I. Dhillon, et. al. (1997). "ScaLAPACK Users' Guide". *SIAM, Philadelphia.*

[3] I. Blanquer, D. Guerrero, V. Hernandez, E. Quintana-Orti and P. Ruiz "Parallel-SLICOT Implementation and Documentation Standards" *SLICOT Working Note 1998-1.*

[4] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, et. al (1993) "PVM 3 User's Guide and Reference Manual". *Oak Ridge National Laboratory, Oak Ridge, Tennessee*.

[5] D. W. Gu, P. Hr. Petkov, M. Konstantinov "An introduction to H∞ optimi-sation designs" *NICONET Report 1999-4.*

[6] B. Haverkamp "Efficient implementation of subspace method identification algorithms" *NICONET Report 1999-3*.

[7] Consorcio NICONET (1999) . NICONET home page http://www.win.tue.nl/niconet/niconet.html.

[8] Consorcio SLICOT (1999), SLICOT home page. http://www.win.tue.nl/niconet/NIC2/slicot.html.

[9] P. Van Dooren "Selection of basic software tools for structured matrix decompositions and perturbations" *NICONET Report 1999-9*

[10] A. Varga "Selection of Basic Software Tools for Standard and Generalized State-Space Systems and Transfer Matrix Factorizations." *SLICOT Work. Note 1998-3*.

[11] A. Varga "Model reduction routines for SLICOT" *NICONET Report 1999-8*

[12] Volker Mehrman, Vasile Sima, Andras Varga and Hongguo Xu. "A MATLAB MEX-file environment of SLICOT". *SLICOT Working Note 1999-11.*

[13] Petko Hr Petkov, Mihail M. Konstantinov, Da-wei Gu and Volker Mehrman. "Numerical Solutions of Matrix Riccati Equations: A Comparison of Six Solvers". *NICONET Report 1999-10.*

[14] Wouter Favoreel, Vasile Sima, Sabine Van Huffel, Michael Verhaegen and Bart De Moor. "Subspace Model Identification of Linear Systems in SLICOT". *SLICOT Working Note 1988-6*.

[15] Jack Dongarra and R. Clint Whaley. "LAPACK Working note 94: A user's guide to the BLACS v1.0". *Technical Report UT-CS-95-281, Department of Computer Science, University of Tennessee, March 1995.*

[16] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra. "MPI: The Complete Reference". *The MIT Press, 1996*.

[17] Consorcio SLICOT (1999), SLICOT Examples http://www.win.tue.nl/niconet/NIC2/examples.html

[18] Petko Petkov, Mihail Konstantinov, Da-Wei Gu, Volker Mehrmann. " Numerical solution of matrix Riccati equations: a comparison of six solvers". *NICONET Report 1999-10: August 1999.*

[19] V. Sima. "Algorithms and LAPACK-Based Software for Subspace Identification". *Proc. CACSD'96 Symposium, Dearborn, MI, pp. 182-187, 1996.*

[20] A. Varga. "Numerical Methods and Software Tools for Model Reduction". *Proc. 1st MATHMOD Conf., Vienna, pp. 226-230, 1994.*

[21] Vasile Sima, Peter Benner, Sabine Van Huffel and Andras Varga. "Improving the efficiency and accuracy of the MATLAB control toolbox using SLICOT-based gateways". *Presented at MTNS98 Padova, Italy, July 6-10, 1998.*

[22] V. Sima. "High-Performance Numerical Software for Control Systems, and Subspace-Based System Identification". *WGS Report 97-2, March 1997.*

[23] A. Varga. "Numerical Methods and Software Tools for Model Reduction". *Proc. 1st MATHMOD Conf., Vienna, pp. 226-230, 1994*.

[24] The MathWorks home page http://www.mathworks.com.

[25] Scilab Home page. http://www-roq.inria.fr/scilab/scilab.html.

[26] J. J. Dongarra et al., “A set of level 3 BLAS basic linear algebra subprograms”. *Tech. Report ANL-MCS-TM88, Dept. of Computer Science, Argonne National Laboratory, 1988.*

[27] B. S. Garbow et al., “Matrix Eigensystem Routines- EISPACK Guide Extension”. *Lect. Notes in Computer Science, Springer-Verlag, Volume 51, Berlin, 1977.*

[28] J. J. Dongarra et al., “LINPACK Users’s Guide”. *SIAM, Philadelphia, 1979.*

[29] A. C. Hindmarsh. “Brief Description od ODEPACK- A systematized Collection of ODE Solvers”.
http//ww.netlib.org/odepack/doc.

[30] E. Hairer, G. Wanner. “Solving Ordinary Differential Equations II. Stiff and Differential- Algebraic Problems”. *Springer Series in Computacional Mathematics 14, Springer Verlag 1991, Second Edition 1996*.

[31] L. R. Petzold. “DASSL Library Documentation”.
http//ww.netlib.org/ode

[32] P. N. Brown, A. C. Hindmarsh. L. R. Petzold. “DASPK Package”. *Revision 1995*.

[33] P. Kunkel, V. Mehrmann, W. Rath and J. Weickert. “GELDA: A Software Package for the Solution of General Lineal Differential Algebraic equations”. *SIAM Journal Scientific Computing, Vol. 8, 1997, pp. 115- 138*.