HPC en simulación y control a gran escala

Peter Benner[1⋆], Pablo Ezzatti[2⋆⋆], Hermann Mena[3⋆⋆⋆],
Enrique S. Quintana–Ortí[4†], Alfredo Remón[4‡]

[1] Max Planck Institute for Dynamics of Complex Technical Systems, Alemania
[2] Centro de Cálculo–Instituto de Computación, Universidad de la República, Uruguay
[3] University of Innsbruck, Austria
[4] Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, España

Fecha de entrega: 23 de enero de 2012
Fecha de aprobación: 5 de noviembre de 2012

Resumen La simulación y control de fenómenos que aparecen en microelectrónica, micro-mecánica, electromagnetismo, dinámica de fluidos y en general en muchos procesos industriales, constituye un problema difícil de resolver, debido principalmente al elevado costo computacional de los algoritmos para este propósito. Gran parte de los modelos matemáticos que describen estos fenómenos poseen dimensión grande; por ejemplo, la modelización de microprocesadores desemboca en un sistema dinámico a gran escala que no puede ser resuelto con métodos numéricos tradicionales. En su defecto, son necesarias e incluso obligatorias varias técnicas computacionales de alto desempeño (*high performance computing*, HPC) para enfrentar este tipo de problemas. En el presente artículo revisamos herramientas de HPC que permiten simular y controlar problemas a gran escala. Concretamente, nos centramos en técnicas para la reducción de modelos vía truncamiento balanceado y la resolución de problemas de control lineal cuadrático, que pueden ser implementadas eficientemente en plataformas multi-núcleo con memoria compartida que, además, utilizan uno o más procesadores gráficos (GPUs).

Abstract The simulation and control of phenomena arising in microelectronics, micromechanics, electromagnetism, fluid dynamics and in general in many industrial processes, is a very challenging task; mainly because they require a high computational cost. Most of the mathematical models describing these phenomena have a large dimension, e.g., the simulation of microprocessors, leads to a large scale dynamical system which can not be

⋆ Director del Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Alemania. benner@mpi-magdeburg.mpg.de
⋆⋆ Profesor del Centro de Cálculo–Instituto de Computación, Universidad de la República, 11.300–Montevideo, Uruguay. pezzatti@fing.edu.uy
⋆⋆⋆ University of Innsbruck, Austria. Hermann.Mena@uibk.ac.at
† Catedrático del Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, 12.071–Castellón, España. quintana@icc.uji.es
‡ Investigador del Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, 12.071–Castellón, España. remon@icc.uji.es

solved using conventional methods. Instead, high performance computing HPC techniques have to be applied to deal with these problems. In this paper we review modern tools from HPC which allow us to solve large scale problems. Specifically, we focus on model reduction techniques via balanced truncation and the solution of linear quadratic control problems that can be efficiently implemented on multi-core platforms equipped with one or more graphics processors (GPUs).

Palabras Clave: simulación a gran escala, reducción de modelos, truncamiento balanceado, problema de control lineal cuadrático, ecuaciones de Lyapunov, ecuaciones de Riccati.

Keywords: large-scale simulation, model reduction, balanced truncation, linear quadratic control problem, Lyapunov equations, Riccati equations.

1. Introducción

Simular y particularmente controlar un fenómeno físico, tiene un elevado costo computacional, especialmente, cuando el proceso inmerso está modelado mediante ecuaciones en derivadas parciales (EDPs). Los requerimientos de memoria y tiempo de ejecución están ligados estríctamente a la discretización del dominio, esto es, a los puntos en los cuales se aproxima la EDP. Esta discretización se puede realizar por diferentes métodos como elementos finitos o diferencias finitas. Por esta razón el costo computacional puede crecer, incluso exponencialmente, si se requiere una alta precisión para resolver el problema numéricamente. Por otro lado, en el proceso de producción de circuitos integrados de las últimas décadas, el tamaño de los dispositivos se ha reducido y la velocidad de los mismos se ha incrementado en una relación inversamente proporcional; así lo afirmó Gordon E. Moore, co-fundador de Intel, el 19 de abril de 1965 en su conocida *Ley de Moore* (esta ley empírica expresa que, aproximadamente, cada 18 meses se duplica el número de transistores en un circuito integrado). Es importante señalar que la simulación numérica ha cumplido, y lo sigue haciendo, un rol fundamental en el diseño de estos dispositivos; ya que permite determinar con alta precisión el comportamiento y desempeño del dispositivo.

El sistema dinámico que describe el circuito presenta una gran escala y su resolución numérica es un reto para la comunidad científica, y constituye incluso un problema abierto si la dimensión es muy grande y/o si el problema es denso no estructurado. Por la amplia gama de aplicaciones, los sistemas con características similares aparecen en general en problemas gobernados por ecuaciones diferenciales parciales, y las numerosas técnicas que han sido propuestas para tratar estos problemas persiguen reducir su elevado costo computacional [4,5,15,75].

Si el costo computacional está altamente relacionado con la dimensión del sistema, parece razonable y necesario reducir la dimensión del mismo para resolverlo en tiempo real. Los métodos que explotan esta idea son conocidos como reducción de modelos o reducción de orden. Estas técnicas han probado ser particularmente eficientes en la simulación de circuitos eléctricos [24,39,56,67,74]. En general estos métodos se clasifican en:

- Métodos de proyección
- Métodos *Moment Matching*
- Métodos de truncamiento balanceado

En el presente trabajo nos centraremos en métodos de truncamiento balanceado, particularmente en la resolución numérica de ecuaciones de Lyapunov que caracterizan esta técnica. También revisaremos brevemente las ecuaciones matriciales cuadráticas que aparecen en control óptimo y cuya resolución numérica requiere el tratamiento numérico de ecuaciones de Lyapunov y la aplicación de técnicas de computación de alto desempeño. Es importante señalar que existen muchas variantes del método de truncamiento balanceado que emplean ecuaciones algebraicas de Riccati [16]; estas últimas no se revisarán en este trabajo.

El artículo está organizado de la siguiente manera: en la Sección 2 se describe el sistema dinámico relacionado con el modelo matemático; en la Sección 2.1, se introduce la reducción de modelos y la estructura de cierto tipo de ecuaciones matriciales asociadas a esta técnica. En la Sección 2.2 se describen las ecuaciones matriciales que aparecen en el control lineal cuadrático. Los métodos de resolución para estas ecuaciones matriciales así como sus algoritmos se ilustran en la Sección 3, mientras que en la Sección 4 se describen técnicas de HPC para implementar eficientemente los algoritmos descritos en la sección anterior. Finalmente, en la Sección 5 se presentan algunos ejemplos numéricos, con el propósito de comprobar la precisión y factibilidad de las técnicas propuestas en el presente trabajo. Algunas conclusiones se aportan en la Sección 6.

2. Modelo matemático

Tras linealizar un sistema no lineal o un sistema de mayor orden, se obtiene un sistema lineal no dependiente del tiempo de la forma:

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

donde $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, y $D \in \mathbb{R}^{p\times n}$ son matrices constantes. En el tipo de problemas que nos interesa la dimensión del sistema n, en general, varía entre $10^3 - 10^7$, dependiendo de la aplicación específica que se considere. En cambio, el número de entradas o controles, representado por $u(t)$, y el número de salidas del sistema, representado por $y(t)$, a menudo es mucho menor que el orden del sistema, esto es, $m, p \ll n$.
Por ejemplo, si se aplica análisis nodal modificado para generar un sistema de ecuaciones para una red RLC multi-puerto, el sistema tiene la forma siguiente:

$$\begin{aligned} \mathcal{L}\dot{x}(t) &= -\mathcal{G}x(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2}$$

donde

$$\mathcal{L} = \begin{bmatrix} E & 0 \\ 0 & L \end{bmatrix}, \quad \mathcal{G} = \begin{bmatrix} G & M \\ -M^T & 0 \end{bmatrix}, \quad x = \begin{bmatrix} v \\ i \end{bmatrix}.$$

Los vectores y y u representan las corrientes de los puertos y sus voltajes, respectivamente. Las matrices E, L, G son simétricas y denotan la capacitancia nodal, inductancia y conductividad, respectivamente. M representa la matriz de incidencia asociada con el inductor de corriente. N indica dónde se encuentran las fuentes de voltaje, y la matriz C especifica en qué punto se miden los puertos de corriente. En este ejemplo, la matriz $\mathcal{L}$ es no singular, esto es, existe su inversa, por lo cual podemos escribir el sistema (2) en la forma (1). Si la matriz $\mathcal{L}$ es singular, (2) corresponde a una ecuación diferencial algebraica, [78].

Existen muchos métodos para resolver numéricamente sistemas a gran escala de la forma (1) [15,28]. En la siguiente sección se introducen los métodos de reducción de modelos basados en truncamiento balanceado, así como su relación con el problema de control lineal cuadrático.

2.1. Reducción de modelos

El objetivo de la reducción de modelos (RM) es encontrar un sistema cuyo orden sea mucho menor que el sistema original, y al mismo tiempo este nuevo sistema sea una buena aproximación del sistema original. Concretamente, se busca para la misma secuencia de entradas, $u(t)$, la salida del sistema, $y(t)$, represente de buena manera lo que ocurre en el sistema original; en otras palabras, se busca un sistema de la forma:

$$\begin{aligned} \dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}u(t), \qquad t > 0, \\ \tilde{x}(0) &= \tilde{x}_0, \\ \tilde{y}(t) &= \tilde{C}\tilde{x}(t) + Du(t), \qquad t > 0, \end{aligned} \tag{3}$$

cuyo orden, r, sea mucho menor que el orden del sistema original, $r \ll n$, y adicionalmente $\tilde{y}$ sea una buena aproximación, de acuerdo a cierto criterio, de la salida del sistema original y.

Entre los métodos de RM, se destacan los de reducción/substracción-Guyan, truncamiento modal, aproximaciones de Padé, métodos de subespacio de Krylov, truncamiento balanceado, etc. Una característica de muchos de estos métodos es realizar proyecciones de tipo Galerkin o Petrov-Galerkin del espacio de estado en un subespacio de dimensión menor. Para saber más sobre reducción de modelos, ver [4,5].

En este trabajo abordaremos los métodos de truncamiento balanceado. En ingeniería uno de los criterios más utilizados para medir la aproximación de $\tilde{y}$ a y es la denominada función de transferencia. Esto se debe a que, bajo ciertas suposiciones, se puede probar que:

$$||y - \tilde{y}||_{L_2[0,\infty)} \leq ||G - \tilde{G}||_\infty ||u||_{H_2}, \tag{4}$$

donde

$$\begin{aligned} G &:= C(sI - A)^{-1}B + D, \\ \tilde{G} &:= \tilde{C}(sI - \tilde{A})^{-1}\tilde{B} + D, \end{aligned}$$

son las funciones de transferencia de los sistemas (1) y (3), respectivamente, y $||.||_{\mathcal{H}}$ denota la norma en el espacio $\mathcal{H}$.

Debido a la relación (4), muchas técnicas de RM tratan de minimizar $||G - \tilde{G}||_\infty$, pues con ello implícitamente están minimizando $||y - \tilde{y}||_{L_2[0,\infty)}$; este es precisamente el caso del truncamiento balanceado. Otros criterios de minimización consideran modelos de frecuencia pesados y reducción de controles [85]. El truncamiento balanceado se fundamenta en separar el espacio de transformaciones de estado mediante una aplicación no singular de la forma:

$$\begin{aligned}\mathcal{T} \ : \ (A, B, C, D) &\mapsto (TAT^{-1}, TB, T^{-1}C, D) \\ &= \left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \begin{bmatrix} C_1 C_2 \end{bmatrix}, D \right)\end{aligned}$$

donde las matrices del modelo reducido se definen como

$$(\tilde{A}, \tilde{B}, \tilde{C}, D) = (A_{11}, B_1, C_1, D),$$

La aplicación $\mathcal{T}$ balancea el Gramian de controlabilidad, Wc, y el Gramian de observabilidad, Wo, que corresponden a soluciones de ecuaciones matriciales de Lyapunov de la forma:

$$\begin{aligned} AWc + WcA^T + BB^T &= 0, \\ AWo + WoA^T + CC^T &= 0. \end{aligned} \tag{5}$$

Antes de introducir algunos métodos para resolver (5), en los que se profundiza en las Secciones 3 y 4, es importante señalar que, como Wc y Wo son matrices semidefinidas positivas, se pueden factorizar de la forma $Wc = S^T S$, $Wo = R^T R$. Además, si se escogen los factores $S, R \in \mathbb{R}^{n \times n}$ triangulares, estos corresponden a los factores de Cholesky de Wc y Wo, respectivamente. Esta propiedad permite definir el truncamiento balanceado usando el producto SR^T en lugar del producto de los Gramians. Algunas implementaciones de algoritmos de reducción de modelos y truncamiento balanceado, se describen en [15,85].

El costo computacional de la RM, vía truncamiento balanceado, depende principalmente de la resolución de las ecuaciones de Lyapunov asociadas a los Gramians de observabilidad y controlabilidad.

En sistemas no estables, la aplicación del truncamiento balanceado conlleva la necesidad de resolver ecuaciones algebraicas de Riccati, en lugar de ecuaciones de Lyapunov. Las ecuaciones de Riccati aparecen en muchas aplicaciones de ciencia e ingeniería, especialmente en la teoría de control. En la siguiente sección se describe brevemente el problema de control lineal cuadrático *linear-quadratic regulator* (LQR) y su relación con las ecuaciones de Riccati.

2.2. Ecuaciones matriciales en control

Los problemas de optimización gobernados por EDPs de tipo parabólico pueden formularse como problemas de Cauchy abstractos. Imponiendo un funcional de costo cuadrático, se obtiene un problema LQR para un sistema de dimensión infinita. El control óptimo para este problema está definido vía *feedback* en términos de la solución de la ecuación de Riccati para operadores, ya sea esta

de tipo algebraico o diferencial, dependiendo de si el horizonte de integración que se considera es infinito o finito, respectivamente. Existen varios resultados de convergencia respecto al problema infinito-dimensional y las ecuaciones de Riccati asociadas, ver por ejemplo [65]. Estos resultados nos brindan el marco teórico necesario para resolver el problema numéricamente.

Consideremos entonces el problema LQR semidiscretizado:

$$\min_{u\in L^2(0,T_f;\mathcal{U})} \frac{1}{2}\int_0^{T_f} [x(t)^T Q x(t) + u(t)^T R u(t)]dt + x(T_f)^T G x(T_f), \tag{6}$$

con respecto a (1). Por simplicidad tomemos $D = 0$, $Q = I$, y $R = I$. Luego el control óptimo está dado por:

$$u(t) = -B^T X_\infty x(t),$$

donde X_∞ es la solución de la ecuación algebraica de Riccati

$$0 = \Re_h(X) = C^T C + A^T X + XA - XBB^T X, \tag{7}$$

o la solución de la ecuación diferencial de Riccati

$$\dot{X} = -\Re_h(X) = -C^T C - A^T X - XA + XBB^T X, \tag{8}$$

si $T_f = \infty$ y $G = 0$ o $T_f < \infty$, respectivamente.

El orden de las ecuaciones matriciales (7) y (8) es cuadrático respecto al orden del sistema, esto es, su solución implica resolver un sistema de EDOs de orden n^2. Por esta razón, en problemas a gran escala, la aplicación de técnicas tradicionales no es factible. Si se tiene en cuenta que típicamente, los coeficientes de (7) y (8) tienen una estructura definida (p.e., dispersión, simetría o rango bajo) es posible definir métodos numéricos capaces de explotar eficientemente esta estructura. La relación entre estas ecuaciones y la ecuación de Lyapunov, así como los métodos numéricos utilizados se describen en la siguiente sección.

Observación 1 *En sistemas no estables, la aplicación de truncamiento balanceado de tipo* linear-quadratic gaussian *LQG conlleva la necesidad de resolver ecuaciones algebraicas de Riccati en lugar de ecuaciones de Lyapunov, puesto que en este caso particular se deben considerar Gramians de lazo cerrado en el compensador LQG.*

3. Algoritmos

Como se ha planteado en la Sección 2.1, la aplicación de RM —vía truncamiento balanceado a sistemas lineales de gran escala— exige resolver ecuaciones de Lyapunov o ecuaciones de Riccati, dependiendo de si el sistema es estable o no. Estas últimas ecuaciones aparecen también en control óptimo, y están estrictamente ligadas unas a otras, como veremos más adelante.

La estructura de la ecuación (7) es compartida por la ecuación (5) excepto por el término no lineal XBB^TX. Lo mismo ocurre con el lado derecho de la ecuación (8).

Observemos que la derivada de Frechét (es una generalización de la derivada usual a espacios de Banach [61]) del operador $\Re_h$ en P viene dada por el operador de Lyapunov:

$$\begin{aligned}\Re_h^{'}|_P : \quad Q \mapsto\ & (A - BR^{-1}B^TP)^TQ \\ & +Q(A - BB^TP);\end{aligned}$$

aplicando el método de Newton se tiene que

$$\begin{aligned}N_\ell &:= \left(\Re_h^{'}|_{P_\ell}\right)^{-1} \Re_h(P_\ell), \\ X_{\ell+1} &:= X_\ell + N_\ell.\end{aligned}$$

Y, por tanto, para una matriz inicial dada, este método puede implementarse como se recoge en el Algoritmo 3.1:

Algoritmo 3.1 Método de Newton para EAR

Require: P_0, tal que A_0 es estable
1: $A_\ell \leftarrow A_h - B_h B_h^T P_\ell$
2: Resolver la ecuación de Lyapunov
$A_\ell^T N_\ell + N_\ell A_\ell = -\Re_h(P_\ell)$
3: $P_{\ell+1} \leftarrow P_\ell + N_\ell$

En las aplicaciones que consideramos, es posible enunciar que, para intervalos pequeños, la solución aproximada de $X_k \approx X(t_k)$, en general, es un valor estabilizado muy bueno para empezar la iteración; ver [29]. Por lo tanto, toda A_ℓ es estable y la iteración P_ℓ converge a la solución, P_*, en orden cuadrático.

El método de Newton para ecuaciones algebraicas de Riccati (EAR) puede ser reformulado como una iteración de un solo paso, si se reescribe de tal manera que la siguiente iteración sea calculada directamente de la ecuación de Lyapunov como se muestra en el paso 2 del Algoritmo 3.1, esto es:

$$\begin{aligned}&(A - BR^{-1}B^TP_\ell)^TP_{\ell+1} + P_{\ell+1}(A - BB^TP_\ell) = \\ &-C^T\tilde{Q}C - P_\ell BR^{-1}B^TP_\ell =: -W_\ell W_\ell^T.\end{aligned}$$

De esta manera, para resolver la EAR, es necesario resolver una ecuación de Lyapunov en cada iteración, de la forma:

$$F^TX + XF = -WW^T, \tag{9}$$

siendo F una matriz estable en cada iteración. Usualmente, el método de Newton converge muy rápido; por ejemplo, en el problema de transferencia de calor en

dos dimensiones descrito en la Sección 5, la convergencia se alcanza de 3 a 5 iteraciones.

Para resolver ecuaciones de Lyapunov, dispersas a gran escala, usaremos el método ADI (*Alternating Direction Implicit*) que se puede formular como: (ver [87])

$$\begin{aligned}(F^T + p_j I)Q_{(j-1)/2} &= -WW^T - Q_{j-1}(F - p_j I),\\ (F^T + \overline{p_j} I)Q_j^T &= -WW^T - Q_{(j-1)/2}(F - \overline{p_j} I),\end{aligned}$$

donde $\overline{p}$ denota el conjugado de $p \in \mathbb{C}$. Si los parámetros p_j se escogen apropiadamente, entonces $\lim_{j\to\infty} Q_j = Q$ converge en orden super-lineal.

Sin embargo, en problemas a gran escala, no es posible aplicar directamente el método ADI; incluso no es posible almacenar en memoria las matrices que aparecen en cada paso, pues lo anterior excede fácilmente la capacidad de memoria de un computador. Por esta razón, es necesario utilizar técnicas de rango bajo para su resolución. Concretamente utilizaremos el método de rango bajo Newton-ADI presentado en [25,74].

Algoritmo 3.2 Método de rango bajo ADI

Require: F, W y el conjunto de parámetros ADI $\{p_1, \ldots, p_k\}$
Ensure: $Z = Z_{i_{max}} \in \mathbb{C}^{n, i_{max} n_\omega}$ tal que $ZZ^T \approx X$
1: $V_1 = \sqrt{-2\mathrm{Re}\,(p_1)}(F^T + p_1 I)^{-1} W$
2: $Z_1 = V_1$
3: **for** $j = 2, 3, \ldots$ **do**
4: $\quad V_j = \dfrac{\sqrt{\mathrm{Re}(p_j)}}{\sqrt{\mathrm{Re}(p_{j-1})}} \left(I - (p_j + \overline{p_{j-1}})(F^T + p_j I)^{-1}\right) V_{j-1}$
5: $\quad Z_j = \left[\, Z_{j-1}\ V_j \,\right]$
6: **end for**

En el Algoritmo 3.2, Z es una matriz de rango bajo que aproxima X, es decir, $Z \in \mathbb{R}^{n\times q}$ con $q \ll n$. La aproximación de la matriz Z es muy precisa. De hecho, no se requiere un costo computacional adicional para conseguir resultados similares a los que se obtienen con métodos convencionales.

Por otro lado, cabe señalar que todas las matrices V_j tienen igual número de columnas que $W \in \mathbb{R}^{n\times n_w}$, es decir, en cada iteración j, es necesario resolver w sistemas de ecuaciones lineales con los mismos coeficientes matriciales $F^T + p_j I$. En la implementación del algoritmo se explota esta propiedad. Si la convergencia del método de rango bajo ADI se alcanza con respecto a cierto criterio de parada después $i_{\text{máx}}$ pasos, entonces $Z_{i_{\text{máx}}} = [\, V_1, \ldots, V_{i_{\text{máx}}}] \in \mathbb{R}^{n\times i_{\text{máx}} n_w}$, con $V_j \in \mathbb{R}^{n\times n_w}$. Cuando n es grande y n_w es pequeño, se espera que $r_i := i_{\text{máx}} n_w \ll n$. En este caso, se ha calculado una aproximación de rango bajo $Z_{i_{\text{máx}}}$ del factor Z de la solución X, esto es $Y = ZZ^T \approx Z_{i_{\text{máx}}} Z_{i_{\text{máx}}}^T$. Sin embargo, si $n_w \cdot i_{\text{máx}}$ crece demasiado, el método pierde eficiencia. Para controlar el crecimiento de $n_w \cdot i_{\text{máx}}$, pueden aplicarse técnicas de compresión de columnas que reducen el número de columnas en $Z_{i_{\text{máx}}}$, sin introducir un aumento del error significativo; ver [31,51].

Si combinamos el Algoritmo 3.2 con el método de Newton (Algoritmo 3.1), se puede definir un método de rango bajo para resolver ecuaciones algebraicas de Riccati denominado Newton-ADI.

Por otro lado, las ecuaciones diferenciales de Riccati aparecen en reducción de modelos cuando se consideran sistemas no estables dependientes del tiempo. La resolución numérica de estas ecuaciones constituyen un reto mayor al de resolver ecuaciones de Lyapunov o ecuaciones algebraicas de Riccati. Más aún, se espera que las EDRs sean rígidas (*stiff*), por lo que se deben utilizar métodos que puedan tratar este fenómeno eficientemente. También es necesario explotar la estructura de las matrices de coeficientes, y la estructura de la ecuación en sí, lo que implica aplicar métodos numéricos que preserven la estructura de la EDR. Entre las técnicas, de un paso y paso múltiple, para resolver sistemas de ecuaciones diferenciales ordinarias rígidas, los métodos BDF (*Backward Differentiation Formulae*) y los métodos de tipo Rosenbrock son usados comúnmente. Versiones matriciales de estos algoritmos aplicables a EDRs a gran escala, se han propuesto en [29,71]. La idea fundamental en estas técnicas es realizar una aproximación de rango bajo a la solución de la EDR y resolver la ecuación para los factores de la solución. Cabe señalar, que el control de paso y orden se realizan también directamente sobre los factores.

3.1. Caso no disperso

Cuando las matrices son densas, utilizamos el método de la función de signo de la matriz (*Matrix Sign Function*) para resolver las ecuaciones de Lyapunov que aparecen en los problemas de reducción de modelos y de control. En la literatura existen varios métodos iterativos para calcular la función de signo de una matriz. Entre ellos, la iteración de Newton es una opción competitiva por su simplicidad, eficiencia, inherente paralelismo y convergencia asintóticamente cuadrática [53].

El Algoritmo 3.3 describe este método. El costo computacional es aproximadamente de $2n^3$ operaciones aritméticas de punto flotante por iteración.

Algoritmo 3.3 Función de matriz signo para ecuaciones de Lyapunov

Require: $F \in \mathbb{R}^{n\times n}$, $W \in \mathbb{R}^{j\times n}$
Ensure: $\tilde{S}$ such that $\tilde{S}^T\tilde{S} \approx X$ and $F^TX + XF = WW^T$
1: $F_0 := F$, $\hat{S}_0 := W^T$
2: $k := 0$
3: **repeat**
4: $\quad F_{k+1} := \frac{1}{\sqrt{2}}\left(F_k/c_k + c_k {F_k}^{-1}\right)$.
$\quad$ Compute the rank-revealing QR (RRQR) decomposition
5: $\quad \frac{1}{\sqrt{2c_k}}\left[\,\tilde{S}_k,\ \ c_k\tilde{S}_k(F_k^{-1})^T\,\right] = Q_s \begin{bmatrix} U_s \\ 0 \end{bmatrix} \Pi_s$
6: $\quad \tilde{S}_{k+1} := U_s\Pi_s$
7: $\quad k := k+1$
8: **until** $\sqrt{\frac{\|F_{k+1}+I\|_\infty}{n}} < \tau\|F_k\|_\infty$

En el Algoritmo 3.3, $\tilde{S}$ representa el factor de la solución aproximada, es decir, $X \approx \tilde{S}^T \tilde{S}$. Es importante tener en cuenta que la convergencia del método puede acelerarse mediante algunas técnicas [53]. En nuestra aproximación empleamos un escalado definido por el parámetro:

$$c_k = \sqrt{\|{F_k}^{-1}\|_\infty / \|F_k\|_\infty}$$

En la prueba de convergencia, τ es la tolerancia para la iteración, y se determina usualmente como función de la dimensión del problema y de la precisión de la computadora ϵ.

En la siguiente sección, se describirá una implementación eficiente de este algoritmo sobre plataformas multinúcleo equipadas con varias GPUs. Esta implementación incorpora numerosas técnicas de HPC y alcanza un desempeño notable.

4. Computación de alto desempeño

La alta exigencia computacional de los algoritmos mostrados en la sección anterior, precisan el uso de técnicas de computación y *hardware* de alto desempeño (HPC).

La utilización de este tipo de técnicas y de *hardware* presenta como principal inconveniente los altos costos económicos asociados tanto por la adquisición de *hardware* y *software*, como por otros costos derivados; por ejemplo la administración y mantenimiento del equipamiento o la energía consumida. Sin embargo, en los últimos años, han irrumpido nuevas tecnologías de HPC de bajo costo, entre las que se destacan los procesadores gráficos, o GPUs. La exigencia ejercida por la industria de los videojuegos sobre esta tecnología, provoca una constante evolución de la capacidad de cómputo de las GPUs a la vez que mantiene su costo en unos niveles moderados. Por todo ello, las GPUs se han posicionado como dispositivos de referencia en computación de alto desempeño.

4.1. *Software* de alto desempeño en problemas de álgebra lineal

Como quedó explícito en la sección anterior, la resolución de los problemas afrontados en este artículo se encuentra estrechamente ligada a la resolución de problemas de álgebra lineal numérica (ALN).

Existen numerosas bibliotecas de alto desempeño que incluyen en su funcionalidad operaciones de ALN. Entre ellas, destaca la especificación BLAS (*Basic Linear Algebra Subprograms*), que da soporte a las operaciones básicas de ALN sobre vectores y matrices. Existen numerosas implementaciones de BLAS, la mayor parte de ellas específicas para una plataforma *hardware*. Es habitual que los propios fabricantes de *hardware* desarrollen versiones de BLAS específicas para cada uno de sus productos. Con el fin de replicar este caso, se han desarrollado diversos esfuerzos similares, como por ejemplo LAPACK [40] y ScaLAPACK [32], para operaciones más complejas de ALN.

La utilización de estas bibliotecas es fuertemente recomendable por la alta eficiencia que proporcionan sus rutinas, así como por la simplicidad y legibilidad del código o la portabilidad sin pérdida de desempeño.

La especificación LAPACK (*Linear Algebra PACKage*), incluye soporte para matrices densas, simétricas, banda y triangulares para diferentes operaciones, como la resolución de sistemas de ecuaciones lineales, la aproximación de mínimos cuadrados y la búsqueda de valores propios y de valores singulares. Habitualmente, las implementaciones de LAPACK están basadas en el uso de rutinas de BLAS, facilitando su portabilidad y adaptación a diferentes arquitecturas sin afectar negativamente su desempeño.

El aumento en la utilización de computadoras paralelas de bajo costo, y en especial, de aquellas con memoria distribuida, motivó la evolución de LAPACK a una nueva versión basada en técnicas de programación paralela distribuida; esta nueva biblioteca se denominó ScaLAPACK y la primera versión fue publicada en el año 1995.

Una de las principales características de la biblioteca es que su diseño especifica distintos niveles de abstracción, buscando encapsular diferentes tareas en distintos componentes.

Al igual que LAPACK, ScaLAPACK posee funciones para las operaciones más comunes en el campo del ALN, tales como la factorización LU y la factorización QR. Sin embargo, ScaLAPACK está diseñada para su ejecución en plataformas de memoria distribuida, en las que tanto los datos (matrices y vectores) como los cómputos se distribuyen entre las diferentes unidades de procesamiento.

Al diseñar la biblioteca ScaLAPACK se decidió encapsular las operaciones sobre vectores que se debían realizar de forma distribuida, para lo cual se creó la especificación de la biblioteca PBLAS (*Parallel* BLAS) [37]. La especificación de PBLAS posee las mismas operaciones que dispone BLAS, pero en este caso las operaciones están diseñadas para ser ejecutadas en paralelo sobre plataformas de memoria distribuida mediante el paradigma de pasaje de mensajes (MPI, PVM, etc.) y la invocación a funciones de la biblioteca de comunicaciones BLACS.

Existen múltiples bibliotecas de características similares a las presentadas, así como esfuerzos exitosos para desarrollar soluciones a problemas más específicos. En el repositorio NetLib [1] se encuentra un buen compendio del trabajo en el área, con una amplia gama de bibliotecas, algoritmos y artículos científicos en este ámbito.

4.2. Aplicación de GPUs a computación de propósito general

En los primeros años del desarrollo de GPUs para computación de propósito general, el crecimiento de la capacidad computacional del *hardware* no fue acompañado por un avance en el *software* de manejo de las GPUs. La programación de GPUs se realizaba mediante herramientas desarrolladas para el tratamiento de imágenes, como por ejemplo las APIs gráficas (OpenGL y DIRECTX). Esto exigía para el desarrollador la necesidad de conocer profundamente los tópicos de computación gráfica, para poder, de esta forma, mapear el problema a resolver en

conceptos de computación gráfica. Posteriormente, se empezaron a utilizar diferentes lenguajes de tipo ensamblador específicos de cada modelo de GPU, lo que implicaba la existencia de varios lenguajes de desarrollo y una baja portabilidad de las aplicaciones.

Para tratar de paliar esta carencia, se presentaron varios lenguajes de alto nivel, que permitían generar códigos para diferentes modelos de GPUs. No obstante, cada herramienta seguía siendo muy dependiente de la arquitectura de la GPU, el modelo, etc. En el año 2007, con el propósito de dar respuesta a estas carencias NVIDIA presentó CUDA.

Este salto importante en el *software* se desarrolló en forma conjunta con un cambio radical en la arquitectura de las GPUs (de NVIDIA), ya que, junto con CUDA, NVIDIA presentó la arquitectura G80 que, como principal característica, presenta la arquitectura unificada sin distinción entre procesadores de pixeles y de vértices. CUDA se ha convertido en un estándar válido para todas las GPUs desarrolladas por NVIDIA.

CUDA es una arquitectura de *software* para el cómputo de propósito general enfocada hacia el cálculo masivamente paralelo en tarjetas de NVIDIA. CUDA se compone de una pila de capas de *software*: un controlador de *hardware*, una interfaz de programación de aplicaciones y bibliotecas matemáticas de alto nivel, (inicialmente dos, CUFFT y CUBLAS, siendo esta última una implementación de BLAS en GPU). El driver CUDA se encarga de la transferencia de datos entre la GPU y la CPU. Conceptualmente, la arquitectura de CUDA se construye alrededor de una matriz escalable de multiprocesadores (SMs). Un multiprocesador en las GPU actuales (arquitectura G80) consiste de ocho procesadores escalares (SP), así como de elementos adicionales como la unidad de instrucciones multihilo y un chip de memoria compartida. Los multiprocesadores crean, gestionan y ejecutan hilos concurrentemente sin apenas costo de planificación. Para lograr explotar de manera optimizada los recursos de la GPU, CUDA provee un nuevo conjunto de instrucciones accesibles a través de lenguajes de alto nivel como son Fortran y C.

Esta arquitectura permite abstraer la GPU como un conjunto de multiprocesadores compuestos a su vez por un conjunto de procesadores orientados a la ejecución de hilos. La noción fundamental en el nuevo modelo de ejecución es que cada multiprocesador ejecutará en cada uno de sus procesadores el mismo conjunto de instrucciones, pero sobre distintos datos, es decir, paralelismo bajo el paradigma de programación SPMD. En particular, y debido a la gran cantidad de hilos que manejan, es común referirse al paralelismo ofrecido por las GPUs como SIMT (del inglés *Single Instruction Multiple Threads*). Los programas a ejecutar en la GPU se denominan *kernels*, y se organizan en un *grid* tridimensional de hilos de ejecución (coordenadas x, y y z). A su vez, los hilos se agrupan en bloques tridimensionales; estas coordenadas se utilizarán para identificar el conjunto de datos sobre el que se quiere que actúe cada hilo. Cada bloque será ejecutado en un multiprocesador en forma independiente (esto es, no puede haber comunicación de datos ni sincronización entre los bloques) y cada hilo en un procesador de hilo (entre estos hilos sí puede haber sincronización y comunicación). Al conjunto

de hilos que se ejecutan en el mismo instante de tiempo en un mismo bloque se le denomina *warp* y su tamaño es 32. Organizar la ejecución de esta manera permite escalar sin necesidad de recompilar el programa.

El acceso a memoria por parte de los hilos es un componente fundamental en el desempeño de un programa sobre CUDA. Se cuenta con una jerarquía de memoria en la cual los accesos deben ser definidos explícitamente (salvo a los registros y la memoria local). La *memoria global* es común a todos los bloques de hilos que se ejecutan dentro de una tarjeta gráfica; es la memoria de mayor capacidad y los datos almacenados pueden persistir durante la ejecución de múltiples *kernels*. Es la única memoria visible desde el *host* y permite la escritura desde GPU. La *memoria compartida* es una memoria a la cual sólo pueden acceder los hilos que corren dentro un mismo bloque. Su capacidad es menor a la memoria global y su ciclo de vida es de una ejecución de *kernel*. Los *registros* son privados de cada hilo y es la memoria más rápida pero su tamaño es muy limitado y su uso es determinado por el compilador. La *memoria local* es privada a cada hilo, tiene la misma velocidad que la memoria global y su ciclo de vida es de una ejecución de *kernel*. También se dispone de la *memoria constante* y la *memoria de texturas*.

Utilización de GPUs para la resolución de problemas de propósito general. La constante evolución de las GPUs, su precio reducido y su alto desempeño han propiciado que el uso de GPUs en HPC sea en la actualidad una importante área de investigación que cuenta ya con numerosos y relevantes avances. A continuación se detallan algunos trabajos relacionados con el tópico de este artículo, con interés particular en ALN.

Como se mostró en la Sección 4.1, en ALN es habitual el uso de la especificación BLAS para desarrollar aplicaciones. En este sentido, gran parte del esfuerzo inicial para acelerar núcleos de ALN con GPU, hasta la presentación en 2007 de CUBLAS por parte de NVIDIA, consistió en la implementación de operaciones incluidas en la especificación BLAS. En esta categoría destacan trabajos pioneros dedicados a la multiplicación de matrices de Hall et al. [52] y Larsen et al. [64]; el de Bajaj et al. [8] para resolver expresiones lineales; y el artículo de Krüger et al. [63], primer trabajo en delinear el desarrollo de núcleos al estilo BLAS; donde los vectores son almacenados en memoria de texturas 2D, organizando las matrices completas por diagonales. También afronta el caso en que las matrices son dispersas utilizando la misma idea para matrices de banda y empleando un formato comprimido para matrices dispersas no estructuradas. Implementan los métodos del GC y Gauss-Seidel. Posteriormente, en 2005, los mismos autores extienden el trabajo [62] utilizando DirectX 9.0 y cambian el formato de almacenamiento de las matrices dispersas no estructuradas por una estrategia comprimida a bloques. Otro autor que estudió de forma temprana el desarrollo de núcleos tipo BLAS sobre GPU es Moravánsky [72]. Tras la aparición de la primera versión de la biblioteca CUBLAS, muchos trabajos se centraron en presentar mejoras sobre las rutinas desarrolladas por NVIDIA, algunos de los cuales han sido incluidos en posteriores versiones de la biblioteca. En este sentido, destaca el

trabajo de Barrachina et al. [10] donde los autores evalúan el desempeño de rutinas de CUBLAS, y posteriormente muestran cómo conseguir mejoras con base en diferentes estrategias, inteligentes particionamientos, implementaciones híbridas de la multiplicación de matrices, modificaciones de la estructura del núcleo de resolución de sistemas lineales triangulares, y en especial la inclusión de estrategias de *padding*. También Volkov y Demmel [86] presentan estrategias para la aceleración de la multiplicación de matrices. Los autores validan sus desarrollos acelerando los métodos de factorización LU y QR; su rutina fue incluida en la versión 2.1 de CUBLAS. Con base en el trabajo de Volkov y Demmel, L. Chien [36] mostró cómo alcanzar mejoras (entre un 10 % y un 15 %) sobre la arquitectura Tesla. Posteriormente, Nath et al. [76] presentaron una versión optimizada de la multiplicación de matrices para la arquitectura Fermi; esta propuesta fue incluida en la versión 3.2 de CUBLAS. Otro trabajo interesante es el de Fatica [45], que extendió estas ideas para desarrollar una versión del benchmark LINPACK que permite incluir GPUs como elementos de cómputo. Gran parte del avance en lo que respecta a la implementación de BLAS sobre GPU, se compendia en el Capítulo 4 del libro de Nath et al. [73].

Operaciones relacionadas con LAPACK, como la factorización LU, también han sido abordadas por diversos autores. En el trabajo de Galoppo et al. [48] se puede encontrar una implementación pionera de métodos para la resolución de sistemas lineales densos; en particular se presentan los métodos de Gauss-Jordan, factorización LU y factorización LU con pivotamiento parcial y completo. Los autores ofrecen un conjunto amplio de pruebas sobre los algoritmos y estudian la influencia de las características de las GPUs en el desempeño de los métodos. En el mismo año, Ino et al. [55] presentaron una implementación de la factorización LU en GPU utilizando estrategias *right-looking*. Posteriormente, Jung y O'Leary [57,58] presentaron una implementación de la factorización de Cholesky sobre una GPU. Los autores proponen almacenar las matrices (simétricas) en formato comprimido rectangular. Posterior a la presentación de CUBLAS destaca el trabajo de Ries et al. [79], donde se estudia la inversión de matrices triangulares sobre GPU. Esta operación es necesaria para computar la matriz inversa mediante la factorización LU. Implementaciones eficientes de la factorización LU y la factorización de Cholesky también fueron presentadas por Barrachina et al. [11,9]. En el estudio se incluye la evaluación de estrategias de cómputo mixto GPU+CPU, *padding* y precisión mixta con refinamiento iterativo.

Ideas similares son estudiadas en los trabajos de Baboulin et al. [7,83], donde se propone acelerar la factorización LU modificando el pivotamiento por un método que implica menor costo computacional (aunque sufre de ciertos problemas de estabilidad). Finalmente, los autores extienden los trabajos para acelerar la factorización de Cholesky mediante el uso de planificación dinámica [68]. La inversión de matrices generales con GPUs es estudiada por Ezzatti et al. en [44], mientras que en [43] se extiende el trabajo para utilizar múltiples GPUs. Por otro lado, la inversión de matrices simétricas y definidas positivas es cubierto en el artículo de Benner et al. [23].

La arquitectura masivamente paralela de las GPUs es más adecuada para la resolución de operaciones de ALN con matrices densas; sin embargo, en diversos trabajos se alcanzan desempeños importantes sobre matrices dispersas. La mayor parte de las investigaciones en esta área se han centrado en la aceleración de métodos iterativos, ya que la resolución de estos métodos generalmente se basa en el producto matrix dispersa por vector (SPMV) siendo esta operación más sencilla de implementar en GPU que las involucradas en los métodos directos. Además de los trabajos pioneros con implementaciones del método de Jacobi y de Gauss-Seidel de Bajaj et al. [8], y del método GC de Bolz et al. [34], Goodnight et al. [50], Hillesland et al. [54], y Krüger et al [62], conviene destacar los trabajos que se comentan a continuación. Buatois et al. [35] presentan implementaciones del método GC incluyendo el uso de un precondicionador de Jacobi. Para resolver el método GC implementan las operaciones SPMV, DOT y SAXPY de BLAS. La implementación es para operaciones con matrices dispersas, para las que emplean una estrategia de almacenamiento CRS a bloques BCRS. Además de los trabajos mencionados, cuyo objetivo era acelerar diferentes métodos iterativos, varios estudios han analizado directa o indirectamente la aceleración de la función SPMV. Este es el caso del trabajo de Sengupta et al. [82] donde se presenta una implementación de *scan* segmentado sobre GPU. Las matrices dispersas se almacenan con el formato CSR, pero también se estudian otras estrategias de almacenamiento. Más recientemente Bell y Garland [14] presentaron el estudio de la multiplicación de matriz dispersa por vector sobre CUDA. Evalúan diferentes formatos de almacenamiento para las matrices dispersas e implementan la operación para los diferentes formatos evaluados. Otros trabajos destacados son el de Gaikwad y Toke [46] que implementan el método BiCGTAB y CGS para diferentes formatos de almacenamiento y distintas implementaciones de SPMV, la de NVIDIA [14], IBM [13] y la presentada en [35]; y el de Anzt et al. [6] también sobre implementaciones de los métodos del CG y GMRES donde se estudian estrategias de precisión mixta, reordenamiento y el impacto en el consumo energético. Para el cómputo con matrices banda, en el caso de las matrices tridiagonales los trabajos se han centrado en implementaciones del método de reducción cíclica (RC) o sus variantes, en este campo destacamos los trabajos de Zhang et al [88], Maciol et al. [70], Göddeke et al. [49] y Alfaro et al. [3]. En cuanto a la implementación de métodos directos para matrices dispersas existen pocos trabajos en la literatura. El pionero dentro de tema es el desarrollo de Christen et al. [38], en el que los autores estudian distintas opciones para extender la biblioteca PARDISO [81] utilizando la GPU. La biblioteca utiliza el método multifrontal y evalúa la aplicación a nivel de operaciones básicas (mediante invocaciones a núcleos computacionales de BLAS) o a nivel de resolución de los frentes (mediante la invocación de núcleos de LAPACK). Más recientemente, Lucas et al. [69] extienden el trabajo estudiando implementaciones del método multifrontal en GPU.

4.3. Aplicación de técnicas de HPC para la reducción de modelos

Uno de los esfuerzos pioneros en el desarrollo de herramientas que incluyen estrategias de HPC para la resolución de problemas de control, y en particular de reducción de modelos, es la biblioteca SLICOT (*Subroutine Library in Control Theory*)[5] [84], que fue desarrollada a finales de la década de los 90, en el marco del proyecto NICONET (*Numerics in Control Network*) [2]. La biblioteca está implementada en Fortran 77 y también es posible su utilización desde Matlab. Su diseño se basa en el uso de las especificaciones LAPACK y BLAS, por lo cual se puede obtener fácilmente una versión paralela de la biblioteca sobre memoria compartida utilizando una implementación multihilo de BLAS. SLICOT es fuertemente utilizada por investigadores del área en la actualidad. Los esfuerzos previos a la introducción de SLICOT por desarrollar bibliotecas para la resolución de problemas de control están revelados en el trabajo de Benner et al. [27].

Posteriormente, se desarrollaron diferentes esfuerzos para incorporar estrategias de paralelismo de memoria distribuida en la biblioteca SLICOT, que llevaron a la realización de la biblioteca PLiC (*Parallel Library in Control*); los principales trabajos relacionados a este proceso se resumen a continuación.

El trabajo de Blanquer et al. [33] presentó algunas propuestas tendientes a paralelizar la resolución de la ecuación de Lyapunov proponiendo PSLICOT (*Parallel* SLICOT). Posteriormente, uno de los principales esfuerzos se vió plasmado en el desarrollo de la primera versión de la biblioteca PLiC [30] propiamente dicho. La biblioteca PLiC posee dos componentes fundamentales, PLiCOC y PLiCMR (del inglés *Parallel Library in Control: Model Reduction*). La primera da soporte a la resolución de problemas de control, mientras que el segundo componente está especialmente diseñado para la resolución de problemas de reducción de modelos. PLiC incluye rutinas para la resolución de problemas de análisis y diseño de SDLs invariantes en el tiempo y de gran escala sobre arquitecturas de computadoras distribuidas y paralelas. Contiene más de 30 subrutinas que permiten resolver ecuaciones matriciales lineales y cuadráticas (Lyapunov, Sylvester, Stein y la ecuación algebraica de Riccati), reducción de modelos, estabilización parcial, y diversas rutinas de soporte. El modelo de paralelismo emula el utilizado por la biblioteca ScaLAPACK. En particular, la resolución de las operaciones de álgebra se hacen invocando a rutinas de ScaLAPACK, siempre que es posible, y las comunicaciones se efectúan mediante invocación a rutinas de la biblioteca BLACS.

Otra extensión a la biblioteca PLiC fue el desarrollo de una versión accesible mediante servicios web. Esta propuesta permitió resolver problemas de reducción de modelos en sistemas remotos a través de la red, eliminando la necesidad de contar con plataformas de HPC locales para la resolución de este tipo de problemas. En el trabajo de Benner et al. [26] se puede profundizar en esta temática. Con el mismo objetivo que el trabajo anterior, de facilitar el uso de la biblioteca PLiC, en el trabajo de Galiano et al. [47] se desarrolló el paquete PyPLiC, en el cual se ofreció una interfaz de alto nivel en el lenguaje de *scripting* Python para la biblioteca PLiCMR.

[5] Disponible en `http://www.win.tue.nl/niconet/NIC2/slicot.html`.

En el trabajo de Remón et al. [77], se aplicaron diferentes técnicas de paralelismo al método LR-ADI para matrices banda. Los autores presentaron dos implementaciones paralelas sobre memoria compartida (arquitecturas SMP) para la factorización LU de matrices de banda. En el trabajo se mostró también cómo aplicar las variantes de factorización desarrolladas para acelerar los métodos LR-ADI en la resolución de problemas de reducción de modelos, consiguiendo porcentajes de mejora superiores al 50 %.

En el caso de los métodos de reducción de modelos basados en subespacios de Krylov, existe gran caudal de trabajos referentes a la aplicación de técnicas de HPC [12,80]. Generalmente, dada la estrategia de resolución de estos métodos, las técnicas de HPC se aplicaron a la resolución del sistema, es decir, al método de Lanczos o alguna de sus variantes.

Finalmente, en Ezzatti et al. [42], los autores estudiaron la aceleración del método BST sobre matrices simétricas definidas positivas sobre arquitecturas multinúcleo.

4.4. Reducción de modelos y problemas de control con GPUs

Como se ha comentado anteriormente, los problemas de reducción de modelos y de control están fuertemente ligados a la resolución de operaciones de álgebra lineal. Dados los excelentes desempeños alcanzados por las GPUs en este tipo de operaciones, en los últimos años se han realizado diversos esfuerzos para su aplicación en la aceleración de la resolución de problemas de reducción de modelos y de control.

El primer esfuerzo tendiente al uso de GPUs para acelerar problemas de reducción de modelos fue el de Benner et al. [21] sobre el cómputo de la función de signo de una matriz empleando una GPU. En el mismo se propone una estrategia híbrida en la que tanto la CPU como la GPU colaboran en el cómputo de la función de signo de una matriz. Posteriormente, el trabajo fue extendido [18] para la resolución de ecuaciones de Lyapunov al utilizar una GPU y explotando estrategias de precisión mixta para alcanzar niveles de precisión altos a un costo computacional bajo. También es interesante el artículo [17], donde se acelera la resolución del problema de reducción de modelos generalizado, mediante el método BT sobre arquitecturas híbridas GPU-CPU. El trabajo se basa en el uso de implementaciones eficientes propietarias, y particularmente desarrolladas sobre GPU, de las principales operaciones matriciales requeridas por el método: la factorización LU, la resolución de sistemas triangulares y la multiplicación de matrices. Los métodos de error relativo también fueron abordados con metodologías de HPC sobre procesadores gráficos en [22]. En ese trabajo, los autores emplearon una NVIDIA Tesla C2050 para resolver la ecuación de Lyapunov y la ecuación algebraica de Riccati (EAR), utilizando aritmética en doble precisión para implementar una versión eficiente del método *Balanced Stochastic Truncation* (BST).

La resolución de ecuaciones diferenciales de Riccati en plataformas equipadas con una GPU fue abordado en [19], y extendido al caso en el que la plataforma destino estaba equipada con múltiples GPUs en [20]. El uso de diversas GPUs,

además de acelerar los cómputos, permitió exteder la dimensión de los problemas tratados alcanzando problemas de dimensión 34.000, trabajando con matrices completas, mientras se alcanzaban valores de aceleración de hasta 20×.

5. Ejemplos numéricos

Modelo PEEC de un inductor espiral [66]. En la Figura 1 se muestra un inductor espiral con una placa de cobre en la superficie. Un sistema dinámico de la forma (1) es obtenido utilizando el *software Fasthenry*, ver [59,60]. El sistema que se obtiene es:

$$\begin{aligned} \dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}u(t), \\ y(t) &= \tilde{B}^T\tilde{x}(t), \end{aligned} \tag{10}$$

donde

$$\tilde{A} = -\tilde{L}^{-\frac{1}{2}}\tilde{R}\tilde{L}^{-\frac{1}{2}}, \quad \tilde{B} = B\tilde{L}^{-\frac{1}{2}},$$

la matriz de inductancia $\tilde{L}$ es simétrica y definida positiva, $\tilde{L}^{\frac{1}{2}}$ representa la raíz cuadrada, esto es, se satisface $\tilde{L}^{\frac{1}{2}}\tilde{L}^{\frac{1}{2}} = \tilde{L}$, $\tilde{R}$ es la matriz de resistencia la cual es cuadrada y dispersa. Referimos al lector a [59,66,60] para una versión detallada del modelo.

Cabe señalar que $\tilde{A}$ no se calcula explícitamente, por su elevado costo computacional; en su lugar las operaciones para la matriz $\tilde{A}$ se realizan implícitamente.

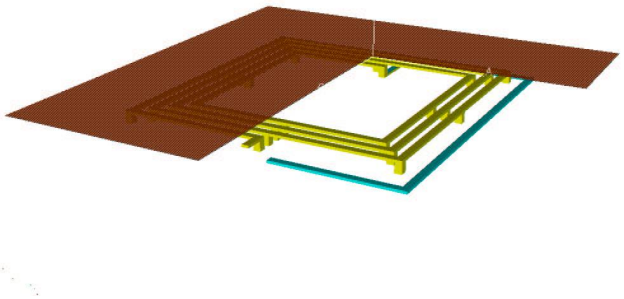

Figura 1. Inductor espiral con placa de cobre en la superficie.

Aplicamos reducción de modelos vía truncamiento balanceado explotando la dispersión mediante el Algoritmo 3.2 para resolver las ecuaciones de Lyapunov. El orden original del sistema (10) es $n = 1434$. El modelo reducido se calculó de manera que satisficiera una tolerancia de 10^{-8}, para lo que se obtuvo un modelo de orden $r = 18$. En la Figura 2 se muestra el valor de las funciones de transferencia.

La sobreposición de estos valores constituye un indicador cuantitativo de la eficiencia del modelo reducido para describir la dinámica del modelo original. En efecto, el modelo reducido aproxima al modelo original con 10 dígitos de precisión. En la Figura 3 se presentan el error absoluto y relativo.

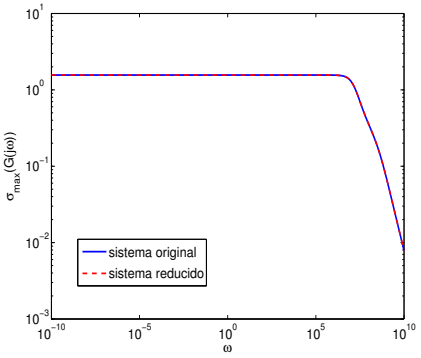

Figura 2. Funciones de transferfencia del modelo original y reducido.

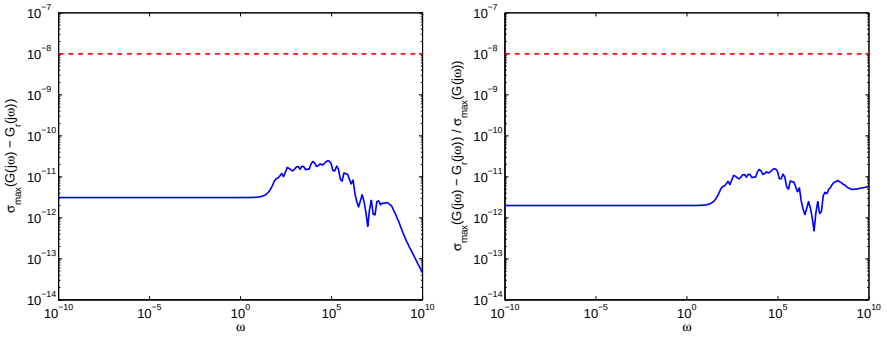

Figura 3. Error absoluto (izquierda) y error relativo (derecha) del modelo reducido.

Enfriamiento óptimo de rieles de tren [28,41]. Este problema aparece en el proceso de producción de rieles de tren. La idea es reducir la temperatura del riel lo más rápido posible maximizando las propiedades del material, como maleabilidad, porosidad, etc. El proceso de enfriamiento se realiza rociando fluidos a bajas temperaturas sobre la superficie del material. Se asume que el riel tiene longitud infinita, lo que permite definir el problema en dos dimensiones. Explotando la simetría del dominio, un mallado inicial se presenta en la Figura 4. Si además se

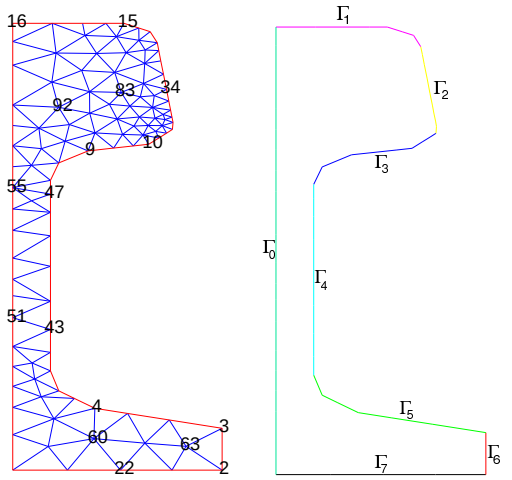

Figura 4. Mallado inicial (izquierda) y partición de la frontera del dominio (derecha).

linealiza el modelo, entonces las ecuaciones que describen el fenómeno tienen la forma

$$\begin{aligned} c\varrho x_t(\xi,t) &= \lambda\Delta x(\xi,t) \text{ en } \Omega\times(0,T),\\ -\lambda\partial_\nu x(\xi,t) &= g_i(t,x,u) \text{ en } \Gamma_i \text{ donde } i=0,\ldots,7,\\ x(\xi,0) &= x_0(\xi) \quad \text{en } \Omega, \end{aligned} \tag{11}$$

donde $x(\xi,t)$ representa la temperatura en el tiempo t en el punto ξ, g_i incluye las diferencias de temperatura entre el fluido que se rocía y la superficie del material, parámetros de intensidad y coeficientes de transferencia de calor.

Tras realizar la discretización se obtiene un sistema como el siguiente:

$$\begin{aligned} M\dot{\tilde{x}}(t) &= N\tilde{x}(t)+\tilde{B}u(t),\\ y(t) &= \tilde{C}\tilde{x}(t), \end{aligned} \tag{12}$$

donde M, $N\in\mathbb{R}^{n\times n}$, M es invertible y $M^{-1}N$ es estable. Luego la forma estándar del sistema es:

$$A=M_L^{-1}NM_U^{-1},\quad B=M_L^{-1}\tilde{B},\quad C=\tilde{C}M_U^{-1}.$$

Cabe señalar que A no se calcula explícitamente, por su elevado costo computacional. En su lugar las operaciones para la matriz A se realizan implícitamente.

Nuevamente se aplicó reducción de modelos vía truncamiento balanceado explotando la dispersión mediante el Algoritmo 3.2 para resolver las ecuaciones de Lyapunov. El orden del sistema original (12) es $n=5177$. El modelo reducido se calculó de manera que satisficiera una tolerancia de 10^{-6} y 10^{-4}, se obtuvieron modelos de orden $r=47$ y $r=18$, respectivamente. En la Figura 5 se muestra el valor de las funciones de transferencia y también el error obtenido.

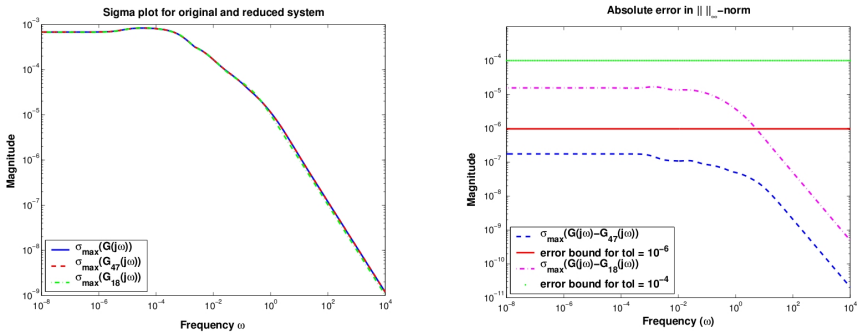

Figura 5. Comportamiento de la función de transferencia (izquierda) y del error (derecha) en el modelo original y los modelos reducidos.

6. Conclusiones

Con base a lo expuesto es posible aplicar modelos matemáticos de gran escala en problemas de simulación y control, con la ayuda de técnicas de computación de alto desempeño.

En particular, en el presente trabajo, se describen algoritmos eficientes de truncamiento balanceado para la resolución de la ecuación de Lyapunov, así como para la resolución de ecuaciones diferenciales y algebraicas de Riccati. La resolución de este tipo de ecuaciones constituye el reto computacional más importante que aparece al tratar problemas de control lineal cuadrático y problemas no lineales mediante técnicas de control predictivo. La idea fundamental, en todos los casos, es aplicar técnicas de álgebra lineal numérica que exploten la estructura de cada problema particular; de esta manera el costo de resolución, desde el punto de vista de la cantidad de operaciones aritméticas necesarias y de la cantidad de memoria requerida, se reducen considerablemente. En la actualidad, las arquitecturas híbridas (compuestas por procesadores multinúcleo conectados a uno o varios procesadores gráficos) ofrecen una elevada capacidad de cómputo y al mismo tiempo presentan un costo económico moderado. El uso de los algoritmos citados en este artículo, y de *software* y técnicas de computación de alto desempeño sobre arquitecturas híbridas CPU-GPU, permite la resolución de problemas de gran escala que anteriormente requerían del uso de plataformas *hardware* de elevado costo.

Referencias

1. Repositorio Netlib. `www.netlib.org/`. Consultado en octubre (2011)
2. Sitio Web oficial de la biblioteca SLICOT `www.slicot.org/`
3. Alfaro, P., Igounet, P, and Ezzatti, P.: Resolución de matrices tri-diagonales utilizando una tarjeta gráfica (GPU) de escritorio. Mecánica Computacional, 30 (2010) 2951–2967

4. Antoulas A.C.: Lectures on the approximation of linear dynamical systems. Encyclopedia of Electrical and Electronics Engineering. John Wiley and Sons (1999) 403–422
5. Antoulas, A. C., Sorensen, D. C., and Gugercin, S.: A survey of model reduction methods for large-scale systems. Contemporary Mathematics, 280 (2001) 193–219
6. Anzt, H., Rocker, B. and Heuveline, V.: Energy efficiency of mixed precision iterative refinement methods using hybrid hardware platforms - An evaluation of different solver and hardware configurations. Computer Science - R & D, 25 (2010) 141–148.
7. Baboulin, M., Dongarra, J. and Tomov, S.: Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures. Manchester Institute for Mathematical Sciences, University of Manchester, Manchester, UK, jan (2009)
8. Bajaj, C., Ihm, I., and Min, J. and Oh, J.: SIMD Optimization of Linear Expressions for Programmable Graphics Hardware. Computer Graphics Forum, 23 (2004) 697–714
9. Barrachina, S., Castillo, M., Igual, F. D., Mayo, R., Quintana-Ortí, E. S.: Solving Dense Linear Systems on Graphics Processors. in Euro-Par '08: Proceedings of the 14th international Euro-Par conference on Parallel Processing, Berlin, Heidelberg, Springer-Verlag, (2008) 739–748
10. Barrachina, S., Castillo, M., Igual, F. D., Mayo, R., Quintana-Ortí, E. S., Quintana-Ortí, G.: Evaluation and Tuning of the Level 3 CUBLAS for Graphics Processors. Departamento de Ingeniería y Ciencia de Computadores, Universidad Jaime I, Campus de Riu Sec, s/n 12.071 - Castellón, España, (2008)
11. Barrachina, S., Castillo, M., Igual, F. D., Mayo R., Quintana-Ortí, E. S., Quintana-Ortí, G.: Exploiting the capabilities of modern GPUs for dense matrix computations, Concurrency and Computation: Practice and Experience, 21 (2009) 2457-2477
12. Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. 2nd Edition, SIAM,Philadelphia, PA, (1994)
13. Baskaran, M., Bordawekar, R.: Optimizing sparse matrix-vector multiplication on GPUs, IBM Research Report 24704 (2009).
14. Bell, N., Garland, M. Implementing sparse matrix-vector multiplication on throughput-oriented processors. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, New York, NY, USA, ACM, (2009) 18:1–18:11
15. Benner, P.: Solving large-scale control problems. IEEE Control Systems Magazine, 14(1) (2004) 44–59
16. Benner, P.: System-theoretic methods for model reduction of large-scale systems: Simulation, control, and inverse problems. Proceedings of MathMod 2009, Vienna, February 11-13, 2009, I. Troch and F. Breitenecker, eds., vol. 35 of ARGESIM Reports, (2009) 126–145
17. Benner, P., Ezzatti, P., Kressner, D., Quintana-Ortí, E. S., Remón, A.: Accelerating model reduction of large linear systems with graphics processors. In Lecture Notes in Computer Science, State of the Art in Scientific and Parallel Computing, Springer, (2010)
18. Benner, P., Ezzatti, P., Kressner, D., Quintana-Ortí, E. S. , Remón, A.: A mixed-precision algorithm for the solution of Lyapunov equations on hybrid CPU- GPU platforms. Parallel Computing, 37 (2011) 439–450
19. Benner, P., Ezzatti, P., Mena, H., Quintana-Ortí, E. S. , Remón, A.: Solving differential Riccati equations on multi-GPU platforms. In 2nd Meeting on Linear Algebra, Matrix Analysis and Applications ALAMA10, (2010)

20. Benner, P., Ezzatti, P., Mena, H., Quintana-Ortí, E. S. , Remón, A.: Solving differential Riccati equations on multi-GPU platforms. In 10th International Conference on Computational and Mathematical Methods in Science and Engineering CMMSE11, (2011) 178–188
21. Benner, P., Ezzatti, P., Kressner, D., Quintana-Ortí, E. S. , Remón, A.: Using hybrid CPU-GPU platforms to accelerate the computation of the matrix sign function. In Euro-Par Workshops, H.-X. Lin, M. Alexander, M. Forsell, A. Knüpfer, R. Prodan, L. Sousa, and A. Streit, eds., vol. 6043 of Lecture Notes in Computer Science, Springer, (2009) 132–139
22. Benner, P., Ezzatti, P., Kressner ,D., Quintana-Ortí, E. S. , Remón, A.: Accelerating BST methods for model reduction with graphics processors. In Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics, (2011)
23. Benner, P., Ezzatti, P., Kressner, D., Quintana-Ortí, E. S. , Remón, A.: Hing performance matrix inversion of SPD matrices on graphics processors. In Workshop on Exploitation of Hardware Accelerators WEHA 2011, (2011) 640–646
24. Benner, P., Hinze, M., Ter Maten, J.: Model Reduction for Circuit Simulation. Vol. 74 of Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, (2011)
25. Benner, P., Li, J.-R., Penzl, T.: Numerical solution of large Lyapunov equations, Riccati equations, and linear-quadratic control problems. Numer. Linear Algebra Appl., 15 (2008) 755–777
26. Benner, P., Mayo, R., Quintana-Ortí E. S., Quintana-Ortí, G.: Enhanced services for remote model reduction of large-scale dense linear systems. In PARA,J. Fagerholm, J. Haataja, J. Järvinen, M. Lyly, P. Raback , and V. Savolainen, eds., vol. 2367 of Lecture Notes in Computer Science, Springer, (2002) 329–338
27. Benner, P., Mehrmann, V., Sima, V., Huffel, S. V., Varga, A.: SLICOT -a subroutine library in systems and control theory. Applied and Computational Control, Signals, and Circuits, Birkhuser, (1997) 499–539
28. Benner, P., Mehrmann, V., Sorensen, D.: Dimension Reduction of Large-Scale Systems. Vol. 45 of Lecture Notes in Computational Science and Engineering. Springer-Verlag, Berlin/Heidelberg, Germany, (2005)
29. Benner, P., Mena, H.: BDF methods for large-scale differential Riccati equations. In Proc. of Mathematical Theory of Network and Systems, MTNS 2004, B. D. Moor, B. Motmans, J. Willems, P. V. Dooren, and V. Blondel, eds., (2004)
30. Benner, P., Quintana-Ortí E. S., Quintana-Ortí, G.: A portable subroutine library for solving linear control problems on distributed memory computers. In Workshop on Wide Area Networks and High Performance Computing, London, UK, Springer-Verlag, (1999) 61–87
31. Bischof, C.H., Quintana-Ortí, G.: Computing rank-revealing QR factorizations of dense matrices. ACM Transactions on Mathematical Software, 24(2) (1998) 226–253.
32. Blackford, L. S., Choi, J., Cleary, A., Petitet, A., Whaley, R. C., Demmel, J., Dhillon, I., Stanley, K., Dongarra,J., Hammarling, S., Henry, G., Walker, D.: ScaLAPACK: a portable linear algebra library for distributed memory computers - design issues and performance. In Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM), Supercomputing -96, Washington, DCUSA, IEEE Computer Society (1996)
33. Blanquer, I., Guerrero,D., Hernandez,V., Quintana-Ortí, E. S., Ruiz, P. A.: Parallel-SLICOT implementation and documentation standards. Tech. rep., SLICOT Working Note (1998)

34. Bolz, J., Farmer, I., Grinspun, E., Schröoder, P.: Sparse matrix solvers on the GPU: conjugate gradients and multigrid. ACM Trans. Graph., 22 (2003) 917–924
35. Buatois, L., Caumon, G., Levy, B.: Concurrent number cruncher: An efficient sparse linear solver on the GPU. In High Performance Computation Conference (HPCC), Springer Lecture Notes in Computer Sciences, (2007). Award: Second best student paper.
36. Chien, L. S.: Hand Tuned SGEMM on GT200 GPU. Tech. rep., Department of Mathematics, Tsing Hua University, Taiwan, Feb. (2010)
37. Choi, J., Dongarra, J., Walker,D.: PB-BLAS: A set of parallel block basic linear algebra subprograms. In Proc. of the 1994 Scalable High Performance Computing Conference, IEEE Computer Society Press, (1994)
38. Christen, M., Schenk, O., Burkhart, H.: General-purpose sparse matrix building blocks using the NVIDIA CUDA technology platform. Tech. rep., (2007)
39. Cong, J., Shinnerl, J. R., Xie,M., Kong,T., Yuan, X.: Large-scale circuit placement. ACM Trans. Des. Autom. Electron. Syst., 10 (2005) 389–430.
40. Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling,S., Sorensen,D.: Prospectus for the development of a linear algebra library for high-performance computers. Tech. Rep. ANL/MCS-TM-97, 9700 South Cass Avenue, Argonne, IL 60439-4801, USA, (1987)
41. Eppler, K., Tröltzsch, F.: Discrete and continuous optimal control strategies in the selective cooling of steel profiles., Z. Angew. Math. Mech., 81 (2001) 247–248
42. Ezzatti, P., Quintana-Ortí, E. S., Remón, A.: Effcient model order reduction of large-scale systems on multi-core platforms. In ICCSA (5), B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. O. Apduhan, eds., vol. 6786 of Lecture Notes in Computer Science, Springer, (2011) 643–653
43. Ezzatti, P., Quintana-Ortí, E. S., Remón, A.: High performance matrix inversion on a multi-core platform with several GPUs. IEEE Computer Society, (2011) 87–93
44. Ezzatti, P., Quintana-Ortí, E. S., Remón, A.: Using graphics processors to accelerate the computation of the matrix inverse. The Journal of Supercomputing, online (2011).
45. Fatica, M.: Accelerating LINPACK with CUDA on heterogenous clusters. In GPGPU, (2009) 46–51
46. Gaikwad, A., Toke, I. M.: Gpu based sparse grid technique for solving multidimensional options pricing pdes. In Proceedings of the 2nd Workshop on High Performance Computational Finance, WHPCF -09, New York, NY, USA, ACM, (2009) 6:1–6:9
47. Galiano V., Martín A., Migallón, H. Migallón, V. Penadés, J., Quintana-Ortí, E.S.: PyPLiC: A high-level interface to the parallel model reduction library PLiCMR. In Proceedings of the Eleventh International Conference on Civil, Structural and Environmental Engineering Computing, B. H. V. Topping, ed., Stirlingshire, United Kingdom, (2007), Civil-Comp Press. paper 62.
48. Galoppo, N., Govindaraju,N. K., Henson, M., Manocha,D.: LU-GPU: Effcient algorithms for solving dense linear systems on graphics hardware. In SC 05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, Washing- ton, DC, USA, IEEE Computer Society, (2005) 3
49. Göddeke, D., Strzodka, R.A.: Cyclic reduction tridiagonal solvers on GPUs applied to mixed precision multigrid. IEEE Transactions on Parallel and Distributed Systems, doi: 10.1109/TPDS.2010.61, 22 (2011) 22–32
50. Goodnight, N., Woolley, C., Lewin, G., Luebke, D., Humphreys, G.: A multigrid solver for boundary value problems using programmable graphics hardware. In

HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, (2003) 102–111
51. Gugercin, S., Sorensen, D., Antoulas, A.: A modified low-rank Smith method for large-scale Lyapunov equations. Numer. Algorithms, 32(1) (2003) 27–55
52. Hall, J., Carr, N., Hart, J.: Cache and bandwidth aware matrix multiplication on the GPU. Tech. rep., UIUCDCS-R-20032328, University of Illinois, (2003)
53. Higham,N.: Functions of Matrices: Theory and Computation. SIAM, Philadelphia, USA, (2008)
54. Hillesland, K. E., Molinov, S. Grzeszczuk, R.: Nonlinear optimization framework for image-based modeling on programmable graphics hardware. In ACM SIGGRAPH 2005 Courses, SIGGRAPH '05, New York, NY, USA, ACM, (2005)
55. Ino, F., Matsui, M., Goda, K., Hagihara, K.: Performance study of LU decomposition on the programmable GPU. In HiPC, (2005) 83–94
56. Iordache, M., Dumitriu, L.: Efficient decomposition techniques for symbolic analysis of large-scale analog circuits by state variable method. Analog Integr. Circuits Signal Process., 40 (2004) 235–253
57. Jung, J. H., O'leary. D.: Exploiting structure of symmetric or triangular matrices on a GPU. In First Workshop on General Purpose Processing on Graphics Processing Units, Northeastern Univ., Boston, (2007)
58. Jung, J. H., O'leary. D.: Implementing an interior point method for linear programs on a CPU-GPU system. Electronic Transactions on Numerical Analysis
59. Kamon, M., Tsuk, M., White, J.: Fasthenry: A multipole-accelerated 3-d inductance extraction program. IEEE Transactions on Microwave Theory and Techniques, 42 (1994) 1750–1758
60. Kamon, M., Wang, F., White, J.: Generating nearly optimal compact models from krylov-subspace based reduced order models. IEEE Transactions On Circuits and Systems-II: Analog and Digital Signal Processing, 47 (2000) 239–248
61. Kolmogorov, A., Fomin, S. V.: Elements of the Theory of Functions and Functional Analysis. Dover Publications, (1999)
62. Krüger, J., Schiwietz, T., Kipfer, P., Westermann, R.: Numerical simulations on PC graphics hardware. In ParSim 2004 (Special Session of EuroPVM/MPI 2004, Budapest, Hungary, (2004) 442–450
63. Krüger, J., Westermann, R.: Linear algebra operators for GPU implementation of numerical algorithms. ACM Transactions on Graphics, 22 (2003) 908–916
64. Larsen, E. S., McAllister, D.: Fast matrix multiplies using graphics hardware. In Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM), Supercomputing '01, New York, NY, USA, (2001), ACM, 55–55
65. Lasiecka, I., Triggiani, R.: Control Theory for Partial Differential Equations: Continuous and Approximation Theories I: Abstract Parabolic Systems. Cambridge University Press, Cambridge, UK, (2000)
66. Li, J.-R., Kamon, M.: PEEC model of a spiral inductor generated by Fasthenry, in Dimension Reduction of Large-Scale Systems. P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, (2005) 373–377
67. Li, J.-R., White,J.: Reduction of large circuit models via low rank approximate gramians. International Journal of Applied Mathematics and Computer Science, 11 (2001) 101–121
68. Ltaief, H., Tomov, S., Nath, R., Du, P., Dongarra, J.: A scalable high performant Cholesky factorization for multicore with GPU accelerators. In VECPAR, vol. 6449 of Lecture Notes in Computer Science, Springer, (2010) 93–101

69. Lucas, R. F., Wagenbreth, G., Davis, D. M., Grimes, R.: Multifrontal computations on GPUs and their multi-core hosts. In Proceedings of the 9th international conference on High performance computing for computational science, VECPAR'10, Berlin, Heidelberg, Springer-Verlag, (2011) 71–82
70. Maciol, P., Banas K.: Testing tesla architecture for scientific computing: the performance of matrix-vector product. vol. 3, (2008)
71. Mena, H.: Numerical Solution of Differential Riccati Equations Arising in Optimal Control Problems for Parabolic Partial Differential Equations. PhD thesis, Escuela Politécnica Nacional, Quito, Ecuador, (2007)
72. Moravanszky., A., Ag., N.: Dense matrix algebra on the GPU. In Direct3D ShaderX2, Engel W. F., (Ed.). Wordware Publishing, NovodeX AG, (2003) 2
73. Nath, R., Tomov, S., Dongarra, J.: BLAS for GPUs. In Scientific Computing with Multicore and Accelerators, J. Kurzak, D. A. Bader, and J. a. Dongarra, eds., CRC Press, Dec. (2010)
74. Penzl, T.: Lyapack Users Guide. Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 Numerische Simulation auf massiv parallelen Rechnern, TU Chemnitz, 09107 Chemnitz, Germany, (2000). Available from `http://www.tu-chemnitz.de/sfb393/sfb00pr.html`.
75. Penzl, T.: Algorithms for model reduction of large dynamical systems. Linear Algebra Applications, 415 (2006) 322–343. (Reprint of Technical Report SFB393/99-40, TU Chemnitz, (1999)
76. Nath, S. T. R., Dongarra, J.: An Improved MAGMA GEMM for Fermi Graphics Processing Units. International Journal in High Performance Computing and Architectures, 24 (2010) 511–515
77. Remón, A., Quintana-Ortí, E., Quintana-Ortí, G.: Parallel solution of band linear systems in model reduction. In Parallel Processing and Applied Mathematics, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, eds., vol. 4967 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, (2008) 678–687
78. Riaza R.: Differential-Algebraic Systems. Analytical Aspects and Circuit Applications, World Scientific, (2008)
79. Ries, F., De Marco, T., Zivieri, M., Guerrieri, R.: Triangular matrix inversion on graphics processing unit. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, New York, NY, USA, ACM, (2009) 9:1–9:10
80. Saad,Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., (2003)
81. Schenk, O. Gärtner, K.: Sparse factorization with two level scheduling in pardiso. In PPSC, (2001)
82. Sengupta, S., Harris, M., Zhang, Y., Owens, J. D. Scan primitives for GPU computing. In GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, Aire-la-Ville, Switzerland, Switzerland, (2007), Eurographics Association, 97–106
83. Tomov, S., Nath, R., Ltaief, H., Dongarra, J.: Dense linear algebra solvers for multicore with GPU accelerators. In IPDPS Workshops, IEEE, (2010) 1-8
84. Varga, A.: Task II.B.1 – selection of software for controller reduction. SLICOT Working Note 1999–18, The Working Group on Software (WGS), `http://www.slicot.org/index.php?site=SLmodredR`, (1999)
85. Varga, A.: Model reduction software in the SLICOT library. In Applied and Computational Control, Signals, and Circuits, volume 629 of The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, (2000) 239–282

86. Volkov, V, Demmel, J.: Benchmarking GPUs to tune dense linear algebra. In SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Piscataway, NJ, USA, (2008), IEEE Press, (2008) 1–11
87. Wachspress, E.L.: Iterative solution of the Lyapunov matrix equation. Appl. Math. Letters, 107 (1988) 87–90
88. Zhang, Y., Cohen, J., Owens, J. D.: Fast tridiagonal solvers on the GPU. In PPOPP, (2010) 127–136