Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Newton Globalizado Inexacto para la Ecuación algebraica de Riccati

Olga Domínguez y Marlliny Monsalve

RT 2017-02

Centro de Cálculo Científico y Tecnológico
Caracas, noviembre 2017

Newton globalizado inexacto para la ecuación algebraica de Riccati

Olga Domínguez * y Marlliny Monsalve †

Resumen. En este trabajo se propone el uso de una técnica de globalización inexacta para el Método de Newton, para hallar una aproximación a la solución estabilizadora de la ecuación algebraica de Riccati, que es una ecuación matricial. Específicamente, en este trabajo se presenta una versión incremental del método de Newton que permite adaptar la conocida condición de Armijo para el problema de resolver una ecuación de Riccati. Por tratarse de globalización inexacta, en la nueva propuesta no se resuelve un problema de minimización por iteración como si ocurre con las técnicas exactas. Por otro lado, la nueva propuesta tiene la ventaja adicional de no requerir del cálculo del residual de forma explícita por iteración, lo cual influye positivamente en el costo computacional. Finalmente, mediante algunos ejemplos numéricos, se compara el algoritmo propuesto con una versión del método de Newton que emplea globalización exacta.

1. Introducción. Se tiene interés en hallar cierta solución de la ecuación algebraica de Riccati (ARE por sus siglas en Inglés), definida como

$$\mathcal{R}(X) = A^T X + XA - XBR^{-1}B^T X + C^T QC = 0, \tag{1.1}$$

donde $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, R \in \mathbb{R}^{m\times m}, C \in \mathbb{R}^{q\times n}$ y $Q \in \mathbb{R}^{q\times q}$ con $m, q << n$. Las matrices Q y R se asumen simétricas y positivo definidas y por tanto poseen factorización de Cholesky, es decir, $R = \widehat{R}\widehat{R}^T$ $Q = \widehat{Q}\widehat{Q}^T$. Conviene indicar que, bajo estas características, si $X = X^T$ entonces $\mathcal{R}(X)^T = \mathcal{R}(X)$. La ecuación de Riccati surge en diversas áreas de las ciencias y la ingeniería, especialmente en procesos asociados a problemas de control [1, 2, 3, 4, 5].

Para caracterizar la solución buscada se requiere de las siguientes definiciones. Se dice que la matriz M es *estable* si y sólo si todos sus autovalores tienen parte real negativa. Cuando existe una matriz K tal que $(A - BK)$ es estable se dice que el par de matrices (A, B) es *estabilizable* y K se denomina *matriz estabilizadora*. En este trabajo se tiene interés en encontrar la solución simétrica X_* de (1.1) tal que $(A - BR^{-1}B^T X_*)$ sea estable. Nos referiremos a X_* como la solución estabilizadora [1]. En este contexto se puede probar que la solución estabilizadora es además positivo definida pero con muchos autovalores cercanos a cero, con lo cual es factible encontrar una aproximación de rango bajo a X_*, es decir, es factible hallar $\widehat{X} \in \mathbb{R}^{n\times s}$ con $s \ll n$ tal que $\widehat{X}\widehat{X}^T \approx X_*$. Estas aproximaciones de rango bajo son de mucho interés ya que, bajo ciertas condiciones, es posible diseñar estrategias numéricas que aproximen a $\widehat{X}$ y no a X_* directamente, lo cual tiene una incidencia positiva en el trabajo computacional. Para más detalles sobre la caracterización de las soluciones de (1.1), ver [6, 7].

En vista que (1.1) es una ecuación no lineal, es común emplear el método de Newton para encontrar aproximaciones a la solución debido a que este método es de convergencia *local* con velocidad cuadrática, ver por ejemplo [8, 7, 9, 10]. En líneas generales, dado $X_0 \in \mathbb{R}^{n\times n}$ el método de Newton genera una sucesión de matrices $\{X_k\}_{k\geq 0}$, mediante el siguiente esquema iterativo: $X_{k+1} = X_k + P_k$ donde P_k es la solución al problema lineal $\mathcal{R}'(X_k)P_k = -\mathcal{R}(X_k)$ por cada iteración, y $\mathcal{R}'(X)$ representa la derivada de Fréchet de $\mathcal{R}$ en X. Bajo ciertas condiciones, la sucesión generada converge a una solución de (1.1). Para determinar $\mathcal{R}'(X)P$, usaremos el polinomio de Taylor de $\mathcal{R}$ alrededor de X, $\mathcal{R}(X + P) = \mathcal{R}(X) + \mathcal{R}'(X)P + H(P)$ donde $H(P)$ es tal que

*Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Ap. 47002, Caracas 1041-A, Venezuela (olga.dominguez@ciens.ucv.ve).

†Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Ap. 47002, Caracas 1041-A, Venezuela (marlliny.monsalve@ciens.ucv.ve).

[1] El término usado en Inglés, para referirse a esta solución, y que muchas veces se conserva en Español, es *stabilizing*.

$\lim_{\|P\|\to 0} \frac{\|H(P)\|}{\|P\|} = 0$. Específicamente, se tiene que

$$\mathcal{R}(X+P) = \mathcal{R}(X) + \mathcal{A}^T P + P\mathcal{A} - PBR^{-1}B^T P, \tag{1.2}$$

con $\mathcal{A} = (A - BR^{-1}B^T X)$ y $H(P) = -PBR^{-1}B^T P$.

Usando (1.2) se obtiene que $\mathcal{R}'(X)P = \mathcal{A}^T P + P\mathcal{A}$ y con esta información es posible escribir la *versión incremental del método de Newton* para resolver (1.1) descrita en el Algoritmo 1.

Algoritmo 1 Newton incremental para ARE

1: Dado $X_0 = X_0^T$ tal que $A - BR^{-1}B^T X_0$ es estable.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad \mathcal{A}_k = (A - BR^{-1}B^T X_k)$
4: $\quad \mathcal{A}_k^T P_k + P_k \mathcal{A}_k = -\mathcal{R}(X_k)$ ▷ Resolver para P_k
5: $\quad X_{k+1} = X_k + P_k$
6: **end for**

Más adelante se comentará acerca de la hipótesis requerida en el paso 1 del Algoritmo 1. Es posible, combinar los pasos 4 y 5 para producir una nueva versión equivalente del método de Newton, conocida como la *versión Newton-Kleinman* propuesta en [8].

Algoritmo 2 Newton-Kleinman para ARE

1: Dado $X_0 = X_0^T$ tal que $A - BR^{-1}B^T X_0$ es estable.
2: **for** $k = 0, 1, \cdots$ **do**
3: $\quad \mathcal{A}_k := (A - BR^{-1}B^T X_k)$
4: $\quad G_k^T := [X_k B R^{-1} \widehat{R} \quad C^T \widehat{Q}]$
5: $\quad \mathcal{A}_k^T X_{k+1} + X_{k+1}\mathcal{A}_k = -G_k^T G_k$ ▷ Resolver para X_{k+1}
6: **end for**

Nótese que en ambas versiones se debe resolver un problema matricial lineal por iteración (pasos 4 y 5 de los Algoritmos 1 y 2 respectivamente). Este problema lineal recibe el nombre de *ecuación de Lyapunov*. Las ecuaciones de Lyapunov involucradas sólo difieren en el lado derecho. A simple vista, la construcción del lado derecho de la ecuación del Algoritmo 1 es más costoso que el lado derecho de la ecuación del Algoritmo 2. A pesar de lo anterior, la versión incremental tiene ventajas numéricas sobre la versión Newton-Kleinman: A groso modo la aproximación construida con la forma incremental permite evitar errores de redondeos. En la versión Newton-Kleinman la precisión del nuevo X_{k+1} estaría limitada a t dígitos significativos suponiendo que la Lyapunov es resuelta con t dígitos significativos mientras que en la forma incremental sólo P_k sería la calculada con ciertos dígitos de precisión y por lo tanto la suma $X_k + P_k$ tendrá aproximadamente t cifras correctas. Para más detalles, ver [10, 11].

Aunque la versión incremental tiene ventajas numéricas sobre la versión Newton-Kleinman, esta última se puede adaptar fácilmente mediante esquemas tipo Cuasi-Newton[2] para generar aproximaciones de rango bajo a la solución de la ecuación de Riccati: Existen ideas numéricas para hallar aproximaciones de rango bajo a la solución de una ecuación de Lyapunov siempre y cuando el lado derecho de dicha ecuación tenga la forma FF^T. Para mayor detalle sobre estas ideas ver por ejemplo [12, 13, 14, 15, 16, 17]. Dado que la ecuación de Lyapunov del Algoritmo 2 aproxima directamente a la solución de la ecuación de Riccati y que además su lado derecho es de la forma FF^T, entonces resulta sencillo obtener una aproximación de rango bajo para (1.1).

Otro aspecto de importancia son las condiciones bajo las cuales los Algoritmos 1 y 2 con-

[2]Bajo estos esquemas, las ecuaciones de Lyapunov que surgen en el método de Newton para ARE no se resuelven de forma exacta sino más bien aproximada.

vergen a la solución estabilizadora de (1.1). Esas condiciones están resumidas en el Teorema 1.1, para la demostración ver [7].

TEOREMA 1.1. *Sea X_* la única solución estabilizadora de (1.1). Sea X_0 una aproximación inicial a X_* simétrica tal que $A - BR^{-1}B^TX_0$ es estable. Entonces, las matrices $\mathcal{A}_k$ y X_k para $k = 0, 1, \cdots$, generadas por el Algoritmo 1 satisfacen que:*

- *Todas las $\mathcal{A}_k$ son estables.*
- *$X_* \leq \cdots \leq X_{k+1} \leq X_k \leq \cdots \leq X_0$, donde $N \leq M$ indica que $M - N$ es simétrica positivo semi-definida.*
- *$\lim_{k\to\infty} X_k = X_*$.*
- *Existe una constante $c > 0$ tal que $\|X_{k+1} - X_*\| \leq c\|X_k - X_*\|^2$, es decir, la secuencia $\{X_k\}_{k\geq 0}$ converge cuadráticamente a X_*.*

En relación a las hipótesis del Teorema 1.1, conviene señalar que la elección del iterado inicial no es algo necesariamente trivial. En primer lugar X_0 debe ser tal que $A - BR^{-1}B^TX_0$ sea estable pues sólo bajo esta elección, el teorema nos garantiza la disminución del error por iteración a partir de X_1. Ahora bien, es claro que el iterado X_1 depende en gran medida de la elección del iterado inicial. Por lo tanto, X_0 debe también elegirse tratando de que $\|X_1 - X_*\|$ no se incremente drásticamente ya que de lo contrario la cantidad de iteraciones requeridas para lograr la convergencia garantizada por el Teorema 1.1 puede ser muy grande. Para más detalles, consultar [10]. Hasta este punto hemos presentado las dos versiones más conocidas del método de Newton para resolver una ecuación de Riccati. Ambas versiones tienen una fuerte dependencia de la elección del iterado inicial para su convergencia y ambas deben resolver una ecuación de Lyapunov por iteración. La versión incremental es mejor numéricamente pero la versión Newton-Kleinman es de mayor utilidad para ecuaciones de gran dimensión: la versión incremental pierde interés práctico para problemas grandes pues requiere evaluar de forma explícita $\mathcal{R}(X_k)$ por iteración. Conviene señalar que, en el contexto de la optimización numérica, es común emplear técnicas de globalización, exactas o inexactas, para evitar la dependencia del método de Newton con el iterado inicial [18, 19]. En las técnicas exactas, un cierto problema de minimización debe ser resuelto por iteración de forma exacta, lo cual puede incrementar el costo computacional del método de Newton. Como contrapartida, en las técnicas inexactas se evita resolver dicho problema de minimización. Para el problema de hallar la solución de una ecuación de Riccati, algunos autores han combinado la versión incremental del método de Newton (Algoritmo 1) con una técnica de globalización exacta [20]. En las ideas expuestas en [20], el residual se cálcula de forma explícita. Cabe destacar que la elección de la versión incremental sobre la versión Newton-Kleinman obedece a la facilidad con que las técnicas de globalización se pueden adaptar a esta versión.

Teniendo en cuenta todo lo anterior, surge la idea de usar técnicas de globalización inexactas en combinación con una nueva versión incremental del método de Newton. Se tiene especial interés en que la nueva propuesta evite la evaluación explícita del residual por iteración, lo cual permitiría su uso para problemas de dimensión mayor.

El resto del reporte de organiza de la siguiente forma: en la sección 2 se trata el problema del evitar la evaluación del residual de forma explícita. Seguidamente, en la sección 3 se formula una versión del algoritmo de Newton usando una técnica de globalización inexacta. Por último, la sección 4 está dedicada a la experimentación numérica que inicia con la descripción del proceso a seguir para elegir un iterado inicial adecuado, que garantice las condiciones necesarias exigidas en el Teorema 1.1 para lograr la convergencia de los métodos a ser empleados en la experimentación. Finalmente se tiene una sección dedicada a las conclusiones.

A lo largo de este reporte usaremos $\|\cdot\|_F$ para denotar la norma matricial de Frobenius definida como $\|M\|_F^2 = \langle M, M\rangle_F = traza(M^TM)$, donde $traza(A_{n\times n}) = \sum_{i=1}^n a_{ii}$. Por otro lado, $\|u\|_2^2 = \langle u, u\rangle_2 = u^Tu$ denota la norma euclídea.

2. Residual. Tal y como se comentó en la sección anterior, es realmente importante evitar evaluar el residual por iteración debido al alto costo computacional que esta operación conlleva. En

primer lugar, considere nuevamente la expresión (1.2), que no es más que la expansión de Taylor de $\mathcal{R}$ alrededor de X:

$$\mathcal{R}(X+P) = \mathcal{R}(X) + \mathcal{R}'(X)P + H(P) = \mathcal{R}(X) + \mathcal{A}^T P + P\mathcal{A} - PBR^{-1}B^T P, \tag{2.1}$$

donde $\mathcal{A} = (A - BR^{-1}B^T X)$. Si se define $\mathcal{L}(P)$ como

$$\mathcal{L}(P) = \mathcal{R}(X) + \mathcal{R}'(X)P = \mathcal{R}(X) + \mathcal{A}^T P + P\mathcal{A}, \tag{2.2}$$

entonces se tiene que

$$\mathcal{R}(X+P) = \mathcal{L}(P) - PBR^{-1}B^T P. \tag{2.3}$$

Por otro lado, considere una iteración k cualquiera del Algoritmo 1 con $k \geq 0$. Observe que la ecuación de Lyapunov del paso 4 de dicho algoritmo se puede escribir como $\mathcal{L}(P_k) = \mathcal{R}(X_k) + \mathcal{A}_k^T P_k + P_k \mathcal{A}_k = 0$ y por lo tanto, de la ecuación (2.3) y recordando que $R = \widehat{R}\widehat{R}^T$, se desprende que:

$$\mathcal{R}(X_{k+1}) = \mathcal{R}(X_k + P_k) = \mathcal{L}(P_k) - P_k BR^{-1}B^T P_k = -(P_k B\widehat{R}^{-T})(P_k B\widehat{R}^{-T})^T := -F_k F_k^T. \tag{2.4}$$

De esta manera $\mathcal{R}(X_k) = -F_{k-1}F_{k-1}^T$ con $F_{k-1} = P_{k-1}B\widehat{R}^{-T}$ para $k \geq 1$. Claramente, algunas consideraciones son necesarias para $k = 1$. En este caso se debe calcular F_0 que a su vez requiere de P_0. Del paso 5 del Algoritmo 1 se tiene que, si $X_1 = X_0 + P_0$ entonces $P_0 = X_1 - X_0$ y por tanto todo se reduce a encontrar X_1, ya que X_0 es dado. Ahora bien, si X_0 es tal que $\mathcal{A}_0$ es estable, entonces se debe generar X_1 tal que $\mathcal{A}_1 = A - BR^{-1}B^T X_1$ también sea estable para garantizar que las relaciones de recurrencia del Teorema 1.1 se sigan cumpliendo. Una forma simple de lograr lo anterior es realizar una iteración del Algoritmo 2 con $k = 0$ pues el Teorema 1.1 garantiza que, bajo las hipótesis establecidas, todas las matrices $\mathcal{A}_k$ generadas por el método de Newton son estables. Finalmente, la expresión (2.4) con $k \geq 1$ junto con todas estas consideraciones realizadas para el caso $k = 1$, nos permite generar una nueva versión del método de Newton (ver Algoritmo 3) que no requiere del cálculo explícito de $\mathcal{R}(X_k)$, que es una matriz de orden $n \times n$, sino que sólo debe almacenar y operar con F_{k-1}, que es una matriz de orden $n \times m$ con $m \ll n$.

Algoritmo 3 Newton incremental para ARE (Segunda versión)

1: Dado $X_0 = X_0^T$ tal que $A - BR^{-1}B^T X_0$ sea estable.
2: Realizar una iteración del Algoritmo 1 con $k = 0$ ▷ Para hallar P_0 y X_1
3: **for** $k = 1, 2, \cdots$ **do**
4: $\quad \mathcal{A}_k = (A - BR^{-1}B^T X_k)$
5: $\quad F_{k-1} = P_{k-1}B\widehat{R}^{-T}$
6: $\quad$ Resolver $\mathcal{A}_k^T P_k + P_k \mathcal{A}_k = F_{k-1}F_{k-1}^T$ ▷ Lyapunov para P_k
7: $\quad X_{k+1} = X_k + P_k$
8: **end for**

Observe que en los pasos 5 y 8 se requiere de forma explícita a las matrices $\mathcal{A}_k$ y X_{k+1}, ambas de orden n lo cual tiene una incidencia negativa en el trabajo computacional. Para evitar lo anterior, en primer lugar considere el siguiente proceso inductivo que permite asegurar que X_{k+1} se puede escribir como $\widehat{X}_{k+1}\widehat{X}_{k+1}^T$ con $\widehat{X}_{k+1}$ de orden $n \times s$ y $s < n$. Supóngase que, tanto X_k como P_k se escriben como $\widehat{X}_k\widehat{X}_k^T$ y $\widehat{P}_k\widehat{P}_k^T$ para $k \geq 0$ y que además $\widehat{X}_k \in \mathbb{R}^{n\times p}$ y $\widehat{P}_k \in \mathbb{R}^{n\times t}$ donde $p + t \ll n$. Por tanto,

$$X_{k+1} = X_k + P_k = \widehat{X}_k\widehat{X}_k^T + \widehat{P}_k\widehat{P}_k^T = [\widehat{X}_k \quad \widehat{P}_k]\begin{bmatrix}\widehat{X}_k^T \\ \widehat{P}_k^T\end{bmatrix} = \widehat{X}_{k+1}\widehat{X}_{k+1}^T, \tag{2.5}$$

con $\widehat{X}_{k+1} \in \mathbb{R}^{n\times s}$ y $s = p + t$. En conclusión, el algoritmo anterior no requiere realizar operaciones con X_{k+1} cuadrada de orden n, sino simplemente utiliza a $\widehat{X}_{k+1}$ que es rectangular de orden $n \times s$.

Por otro lado, si se tiene en cuenta que $X_k = \widehat{X}_k \widehat{X}_k^T$, entonces el cálculo para hallar a la matriz $\mathcal{A}_k$ se puede realizar sin construir explícitamente al iterado X_k. El mismo comentario es válido para la construcción de la matriz F_k en el paso 6.

COMENTARIOS 2.1. *Sobre el Algoritmo 3*

1. *Esta nueva versión incremental es equivalente al Algoritmo 1, con la ventaja que requiere menos trabajo computacional en la construcción del lado derecho de la ecuación de Lyapunov del paso 6. Además es fácilmente adaptable para usar técnicas de globalización y mantiene las ventajas numéricas de la versión incremental original (Algoritmo 1).*
2. *El paso 2 implica que debe resolverse una ecuación de Lyapunov, pero esto no significa un incremento en el costo computacional del Algoritmo 3, pues el ciclo de dicho algoritmo inicia en $k = 1$, mientras que en los Algoritmos 1 y 2 comienza en $k = 0$.*
3. *El paso 2 garantiza que las hipótesis del Teorema 1.1 se cumplen.*
4. *Si A es estable, se puede elegir $X_0 = 0$. En este caso, el paso 2 se reduce a resolver la ecuación de Lyapunov $A^T X_1 + X_1 A = C^T Q C$ y finalmente $P_0 = X_1$.*

Todos las versiones presentadas del método de Newton, requieren un criterio de parada que nos permita medir la calidad de la aproximación que generan. En la generalidad, la norma del residual es el criterio más confiable, pero evidentemente, en el contexto de hallar una solución de la ecuación de Riccati, no tiene sentido calcular $\mathcal{R}(X_{k+1})$ para luego obtener su norma. Por suerte, de (2.4), se desprende que

$$\|\mathcal{R}_{k+1}\|_F = \|F_k F_k^T\|_F \leq \|F_k\|_F^2, \tag{2.6}$$

donde $\mathcal{R}_{k+1} := \mathcal{R}(X_{k+1})$ y $k \geq 0$. Claramente, (2.6) es una cota superior para la norma del residual mucho más económica de calcular.

En la próxima sección se describirá brevemente en qué consisten las técnicas de globalización y cómo estas ideas se pueden adaptar al problema de hallar la solución estabilizadora de (1.1).

3. Estrategias de globalización. Antes de iniciar la discusión sobre las técnicas de globalización para resolver funciones no lineales matriciales, conviene recordar rápidamente el uso de estas técnicas en un contexto más simple: la minimización de funciones no lineales sin restricciones. Considere que f, definida de $\mathbb{R}^n$ en $\mathbb{R}$, es la función que se desea minimizar. Suponga que tenemos un punto x_k en el dominio de la función y que en ese punto es posible generar una dirección p_k, que denominaremos dirección de búsqueda. Una técnica de globalización se encarga de elegir un valor λ_k, denominado longitud de paso, tal que $x_{k+1} = x_k + \lambda_k p_k$, sea un mejor iterado que x_k. La técnica de globalización será exacta si λ_k es la solución del problema $\min_{\lambda \neq 0} f(x_k + \lambda p_k)$. Por lo tanto, dada una dirección p_k, se encuentra el λ_k óptimo que logra la mayor disminución del valor de la función en dicha dirección. Por otro lado, en las técnicas de globalización inexacta, no se resuelve el problema de minimización sobre la variable λ (que puede resultar costoso) sino que la longitud de paso λ_k se elige tal que satisfaga ciertos criterios que garantizan que haya una disminución adecuada en el valor de la función en la dirección de p_k. Entre los criterios más comunes para decidir si el λ_k genera un buen x_{k+1} se encuentra las condiciones de Armijo, Goldstein o Wolfe. Así mismo, si el λ_k no satisface la condición elegida, existen diferentes técnicas que permiten generar una nueva longitud. Entre esas técnicas destacan la búsqueda por *Backtraking* y las basadas en interpolación polinomial. Para más detalles sobre las técnicas de globalización ver [18, 19, 21].

En el contexto de resolver una ecuación matricial no lineal, por ejemplo (1.1), el método de Newton tiene *convergencia local* garantizada si la dirección de búsqueda se emplea con longitud de paso igual a uno pero es posible combinar este método con alguna técnica de globalización para hacerlo más robusto en este sentido: si un iterado del método de Newton está lejos de la solución se emplea una longitud de paso más adecuada para tratar de generar un nuevo iterado que esté más cercano a la solución. Si consideramos una iteración k cualquiera del método de Newton, en el cual hemos calculado la dirección de búsqueda P_k, el nuevo iterado se calculará como $X_{k+1} = X_k + \lambda_k P_k$ donde la longitud de paso λ_k se elige tal que X_{k+1} sea un mejor iterado que X_k. Para determinar si un nuevo iterado es mejor o peor que el iterado actual, se usa $\|\mathcal{R}(X)\|_F^2$ como función mérito.

Si λ_k es la solución del problema $\min_{\lambda\neq 0}\phi(\lambda)$ con $\phi(\lambda)=\|\mathcal{R}(X_k+\lambda P_k)\|_F^2$ entonces la técnica de globalización es exacta [20, 22], en caso contrario será inexacta. Independientemente si la técnica usada es exacta o inexacta, se debe garantizar que $\lambda_k=1$ sea una opción factible para la longitud de paso para que, una vez que el iterado obtenido se encuentre en un entorno suficientemente pequeño de la solución, se pueda garantizar la convergencia cuadrática de Newton.

En este trabajo se adaptará para el Algoritmo 1 la muy conocida técnica de globalización inexacta basada en la condición de Armijo en conjunto con la técnica de *Backtraking*. La condición de Armijo establece que el tamaño de paso $\lambda_k>0$ se elige tal que

$$\phi(\lambda_k)\leq\phi(0)+c_1\lambda_k\phi'(0), \tag{3.1}$$

siendo c_1 un parámetro en el intervalo $(0,1)$. Con estos elementos presentamos una primera versión globalizada del método de Newton para la ARE.

Algoritmo 4 Newton incremental globalizado para ARE

1: Dado $X_0=X_0^T$ tal que $A-BR^{-1}B^TX_0$ sea estable.
2: **for** $k=0,1,\cdots$ **do**
3: $\quad\mathcal{A}_k=(A-BR^{-1}B^TX_k)$
4: $\quad$Resolver $\mathcal{A}_k^TP_k+P_k\mathcal{A}_k=-\mathcal{R}(X_k)$ ▷ Lyapunov para P_k
5: $\quad$Seleccionar $\lambda_k>0$
6: $\quad$**while** $\phi(\lambda_k)>\phi(0)+c_1\lambda_k\phi'(0)$ **do** ▷ Condición (3.1)
7: $\quad\quad\lambda_k=\omega\lambda_k$ ▷ Con $c_1,\omega\in(0,1)$
8: $\quad$**end while**
9: $\quad X_{k+1}=X_k+\lambda_kP_k$
10: **end for**

Los pasos del 6 al 8 corresponden a la técnica de *Backtraking*. Al igual que fue posible generar una segunda versión del Algoritmo 1 más llamativa desde el punto de vista computacional, es posible obtener una nueva versión del Algoritmo 4 que no evalúe el residual de forma explícita. Esto además permite optimizar los cálculos asociados a la función ϕ. Para lograr estas mejoras, se procederá de forma similar a la usada para obtener la expresión (2.4). Sean X_k y P_k generados por el Algoritmo 4 con $k\geq 0$. Mediante la serie de Taylor y usando que $\mathcal{L}(P_k)=0$, con $\mathcal{L}$ definido según (2.2), se tiene que

$$\begin{aligned}\mathcal{R}(X_k+\lambda_kP_k)&=\mathcal{R}(X_k)+\lambda_k\mathcal{R}'(X_k)P_k-\lambda_k^2P_kBR^{-1}B^TP_k,\\&=\lambda_k(\mathcal{R}(X_k)+\mathcal{R}'(X_k)P_k)+(1-\lambda_k)\mathcal{R}(X_k)-\lambda_k^2F_kF_k^T\\&=\lambda_k\mathcal{L}(P_k)+(1-\lambda_k)\mathcal{R}(X_k)-\lambda_k^2F_kF_k^T\\&=(1-\lambda_k)\mathcal{R}(X_k)-\lambda_k^2F_kF_k^T,\end{aligned}$$

donde $F_k=P_kB\widehat{R}^{-T}$. Ahora bien, si $\mathcal{R}(X_k)$ se puede escribir como $-H_kH_k^T$ para una cierta matriz $H_k\in\mathbb{C}^{n\times p}$ con $p\leq n$, entonces

$$\begin{aligned}\mathcal{R}(X_{k+1})=\mathcal{R}(X_k+\lambda_kP_k)&=-((1-\lambda_k)H_kH_k^T+\lambda_k^2F_kF_k^T) &(3.2)\\&=-\begin{bmatrix}\sqrt{1-\lambda_k}H_k & \lambda_kF_k\end{bmatrix}\begin{bmatrix}\sqrt{1-\lambda_k}H_k^T\\ \\ \lambda_kF_k^T\end{bmatrix}\\&=-H_{k+1}H_{k+1}^T, &(3.3)\end{aligned}$$

con $H_{k+1}=\begin{bmatrix}\sqrt{1-\lambda_k}H_k & \lambda_kF_k\end{bmatrix}$. Para que el proceso inductivo anterior sea correcto, sólo falta verificar el caso base, es decir, en $k=0$ debe garantizarse que $\mathcal{R}(X_0+\lambda_0P_0)$ sea igual a $-H_1H_1^T$. En vista que X_0 es dado, es factible usar $\lambda_0=1$ y ejecutar un paso del Algoritmo 1 para generar X_1

y P_0 tales que $X_1 = X_0 + P_0$. Además, por (2.4) se sabe que $\mathcal{R}(X_1) = -F_0 F_0^T$ con $F_0 = P_0 B \widehat{R}^{-T}$. Finalmente, tomando $H_1 = F_0$ se completa el proceso inductivo que permite garantizar que, si X_{k+1} es generado mediante el Algoritmo 4, entonces $\mathcal{R}(X_{k+1}) = -H_{k+1} H_{k+1}^T$ para $k \geq 0$.

En cuanto a la condición de Armijo (3.1), es posible encontrar una expresión más simple para evaluarla. En primer lugar, al tomar norma de Frobenius en ambos lados de (3.2) y aplicando algunas propiedades del producto interno, se tiene que:

$$\begin{aligned}\phi(\lambda_k) &= \|\mathcal{R}(X_k + \lambda_k P_k)\|_F^2 = \|(\lambda_k - 1) H_k H_k^T - \lambda_k^2 F_k F_k^T\|_F^2 \\ &= \left\langle (\lambda_k - 1) H_k H_k^T - \lambda_k^2 F_k F_k^T, (\lambda_k - 1) H_k H_k^T - \lambda_k^2 F_k F_k^T \right\rangle_F \\ &= (\lambda_k - 1)^2 \|H_k H_k^T\|_F^2 + \lambda_k^4 \|F_k F_k^T\|_F^2 - 2(\lambda_k - 1)\lambda_k^2 \left\langle H_k H_k^T, F_k F_k^T \right\rangle_F,\end{aligned}$$

de donde se concluye que

$$\phi(\lambda_k) = \alpha(\lambda_k - 1)^2 + \beta \lambda_k^4 - 2\gamma(\lambda_k - 1)\lambda_k^2 \tag{3.4}$$

$$\phi'(\lambda_k) = 2\alpha(\lambda_k - 1) + 4\beta\lambda_k^3 - 2\gamma\lambda_k^2 - 4\gamma\lambda_k(\lambda_k - 1) \tag{3.5}$$

con $\alpha = \|H_k H_k^T\|_F^2$, $\beta = \|F_k F_k^T\|_F^2$ y $\gamma = \left\langle H_k H_k^T, F_k F_k^T \right\rangle_F$. De (3.4) y (3.5) es trivial determinar que $\phi(0) = \alpha$ y $\phi'(0) = -2\alpha$. Por tanto (3.1) se puede escribir como

$$\phi(\lambda_k) \leq \alpha(1 - 2c_1\lambda_k).$$

Más aún, observe que $\phi(\lambda_k) = \|\mathcal{R}(X_{k+1})\|_F^2 = \|H_{k+1} H_{k+1}^T\|_F^2$ y $\alpha = \|H_k H_k^T\|_F^2 = \|\mathcal{R}(X_k)\|_F^2 = \phi(\lambda_{k-1})$, de donde finalmente se desprende que la condición de Armijo se reduce a hallar λ_k tal que

$$\phi(\lambda_k) \leq \phi(\lambda_{k-1})(1 - 2c_1\lambda_k), \tag{3.6}$$

con $k \geq 1$. Con estos elementos estamos en capacidad de presentar una versión globalizada de forma inexacta del método de Newton para resolver ARE.

Algoritmo 5 Newton globalizado inexacto para ARE (Segunda versión)

1: Dado $X_0 = X_0^T$ tal que $A - BR^{-1}B^T X_0$ sea estable.
2: Realizar una iteración del Algoritmo 1 con $k = 0$ ▷ Para hallar P_0 y X_1
3: $F_0 = P_0 B \widehat{R}^{-T}$; $H_1 = F_0$; $\phi(\lambda_0) = \|H_1 H_1^T\|_F^2$
4: **for** $k = 1, 2, \cdots$ **do**
5: $\mathcal{A}_k = (A - BR^{-1}B^T X_k)$
6: Resolver $\mathcal{A}_k^T P_k + P_k \mathcal{A}_k = H_k H_k^T$ ▷ Lyapunov para P_k
7: $F_k = P_k B \widehat{R}^{-T}$
8: Elegir $\lambda_k > 0$; $H := H_k$;
9: $H_{k+1} = \begin{bmatrix} \sqrt{1 - \lambda_k} H & \lambda_k F_k \end{bmatrix}$
10: **while** $\phi(\lambda_k) > \phi(\lambda_{k-1})(1 - 2c_1\lambda_k)$ **do** ▷ Condición (3.6) con $c_1 \in (0, 1)$
11: $\lambda_k = \omega \lambda_k$ ▷ Con $\omega \in (0, 1)$
12: $H_{k+1} = \begin{bmatrix} \sqrt{1 - \lambda_k} H & \lambda_k F_k \end{bmatrix}$
13: **end while**
14: $X_{k+1} = X_k + \lambda_k P_k$
15: **end for**

Cabe mencionar que es posible calcular $\phi(\lambda_k) = \|H_{k+1} H_{k+1}^T\|_F^2$ sin construir explícitamente a $H_{k+1} H_{k+1}^T$. Para simplificar, evitemos el uso del subíndice k y denotemos por $h_i \in \mathbb{R}^n$ a la

i-ésima fila de H^T y utilizando la propiedad de invarianza cíclica de la traza[3], se tiene que

$$\|HH^T\|_F^2 = traza(HH^THH^T) = traza(H^THH^TH) = \sum_{i,j=1}^{m} \langle h_i, h_j\rangle_2^2$$
$$= \sum_{i=1}^{m} \langle h_i, h_i\rangle_2^2 + \sum_{i,j=1;i\neq j}^{m} \langle h_i, h_j\rangle_2^2 = \sum_{i=1}^{m} \|h_i\|_2^4 + 2\sum_{i,j=1;i>j}^{m} \langle h_i, h_j\rangle_2^2. \tag{3.7}$$

La expresión anterior permite calcular $\|HH^T\|_F^2$ sin construir explícitamente HH^T o H^TH.

4. Experimentación numérica. Antes de describir los experimentos numéricos, convienen algunas observaciones relativas a la elección del iterado inicial pues como ya se ha mencionado en secciones anteriores, el X_0 debe ser tal que la matriz $\mathcal{A}_0 = A - BR^{-1}B^TX_0$ sea estable. En la bibliografía se encontraron al menos tres formas de elegir el X_0. La más simple es tomar $X_0 = 0$ cuando A es una matriz estable, pues claramente esa elección garantiza que $\mathcal{A}_0$ sea estable. Por otro lado, en [20], específicamente en el Ejemplo 4, se toma $X_0 = (X + X^T)/2$ donde X es la solución de la ecuación de Riccati (1.1) hallada mediante el método propuesto en [23], que a su vez implica resolver un problema de autovalores generalizados. Esta elección del X_0, además de ser costosa, hace parecer innecesario el uso de una técnica de globalización, pues para este ejemplo en particular, el X_0 obviamente estará muy cerca de la solución. ($\|\mathcal{R}(X_0)\|_F \approx 10^{-7}$). Finalmente, en los Ejemplos 2 y 3 de [20], el X_0 se elige usando el siguiente algoritmo descrito con mayor detalle en [24], donde A y B son las matrices que definen a la ecuación de Riccati (1.1):

Algoritmo 6 Cómo hallar X_0 tal que $\mathcal{A}_0$ sea estable

1: Elegir $\beta \in (0, \|A\|_F)$
2: Resolver $(A + \beta I_n)Z + Z(A + \beta I_n)^T = 2BB^T$ ▷ Para Z
3: $X_0 = Z^\dagger$ ▷ $Z^\dagger$ es la pseudoinversa de Z

En líneas generales, el algoritmo anterior se fundamenta en el siguiente el teorema que garantiza que $\mathcal{A}_0$ sea estable:

TEOREMA 4.1. *Sea (A, B) estabilizable. Entonces $A - BH$ es estable, donde $H = B^TZ^\dagger$, $Z^\dagger$ es la pseudoinversa de Z y Z es la solución de la siguiente ecuación de Lyapunov:*

$$(A + \beta I_n)Z + Z(A + \beta I_n)^T = 2BB^T,$$

con I_n matriz identidad de orden n y $\beta \in (0, \|A\|)$.

Para la demostración del resultado anterior, ver Teorema 10.2.3 de [6, 11, 25]. En la experimentación numérica se usará $X_0 = 0$ de ser posible o X_0 generado mediante el Algoritmo 6. Los ejemplos numéricos de esta sección tienen la finalidad de comparar el método propuesto en [20] que es el método de Newton con globalización exacta, con la nueva propuesta presentada en este trabajo que emplea una estrategia de globalización inexacta (Algoritmo 5). En todos los ejemplos se resuelven ecuaciones de Riccati descritas en detalle en [26]. Las matrices que componen a dichas ecuaciones fueron generadas con la rutina `carex.m`, escrita para Matlab y cuyo código se encuentra disponible en [27] y que es un complemento de [26]. Específicamente, para generar los resultados numéricos de esta sección se implementaron: el algoritmo 2 propuesto en [20] que se denotará por **NGE** (Newton Globalizado Exacto) y el Algoritmo 5 que se denotará simplemente por **NGI** (Newton Globalizado Inexacto). Para la implementación de **NGI** se usó $c_1 = 10^{-4}$ y $\omega = 0.5$, que son valores típicos recomendados en la bibliografía [18, 19]; y en el paso 8 del Algoritmo 5 se usó $\lambda_k = 2$. En los casos que se reporten los resultados del método de Newton sin ninguna estrategia de globalización se implementó la versión descrita en el Algoritmo 1 y se denotará mediante las siglas **NP** (Newton Puro). Todos los procesos iterativos se detienen mediante el criterio $\|\mathcal{R}(X_k)\|_F \leq tol$

[3] $traza(ABC) = traza(CAB) = traza(BCA)$

o cuando un cierto número de iteraciones es alcanzada. En cada ejemplo, según el grado de dificultad del problema a resolver, se indica el valor del parámetro *tol*. Para **NP** y **NGE** el cálculo de $\|\mathcal{R}(X_k)\|_F$ se realiza calculando $\mathcal{R}(X_k)$ explícitamente, mientras que en **NGI** se emplea (3.7). Finalmente, las pruebas se realizaron en un equipo Intel(R) Celeron, CPU N2830, 2.16GHz con 4GB RAM bajo el sistema operativo Windows 10 (64 bits) usando MATLAB (R2008a).

Ejemplo 1. Iniciamos la experimentación numérica considerando el Ejemplo 14 de [26]. En este caso las matrices que componen a la ARE son:

$$A = \begin{bmatrix} -\delta & 1 & 0 & 0 \\ 1 & -\delta & 0 & 0 \\ 0 & 0 & \delta & 1 \\ 0 & 0 & -1 & \delta \end{bmatrix}; \quad B = C^T = [1,1,1,1]^T; \quad Q = R = [1],$$

donde las dimensiones y parámetros asociados al experimentos están en la Tabla 4.1. Por otro lado, para generar X_0 se empleó el Algoritmo 6 con $\beta = \frac{\|A\|_F}{4}$. El valor de *tol* se fijó en 10^{-13} y el máximo de iteraciones en 50. A simple vista, esta es una ARE de dimensión muy pequeña pero el

n	m	q	Parámetro
4	1	1	$\delta = 1$
			$\delta = 10^{-3}$

Tabla 4.1: Dimensiones y parámetros de la ARE del Ejemplo 1.

nivel de dificultad de esta prueba lo determina el valor del parámetro δ: Cuando $\delta \to 0$ ocurre que la solución buscada tiende a ser no estabilizadora, lo cual genera problemas numéricos. En este experimento se incluirá al método **NP** en la gráficas asociadas a la norma del residual, ya que esto nos permitirá observar cómo el valor del parámetro δ influye en la velocidad de convergencia.

En primer lugar se consideró $\delta = 1$ y en vista que todos los métodos a comparar emplearon pocas iteraciones, en la Tabla 4.2 se muestra el valor de la longitud de paso y la norma del residual por cada iteración k, es decir, para cada valor de k se reporta λ_k y $\|\mathcal{R}(X_k)\|_F$ para los métodos **NGE** y **NGI**. Conviene aclarar que λ_0 es igual a 1 en el método **NGI** debido a que el iterado X_1 es generado mediante el Algoritmo 1 usando la actualización $X_1 = X_0 + P_0$. En la Tabla 4.2 se

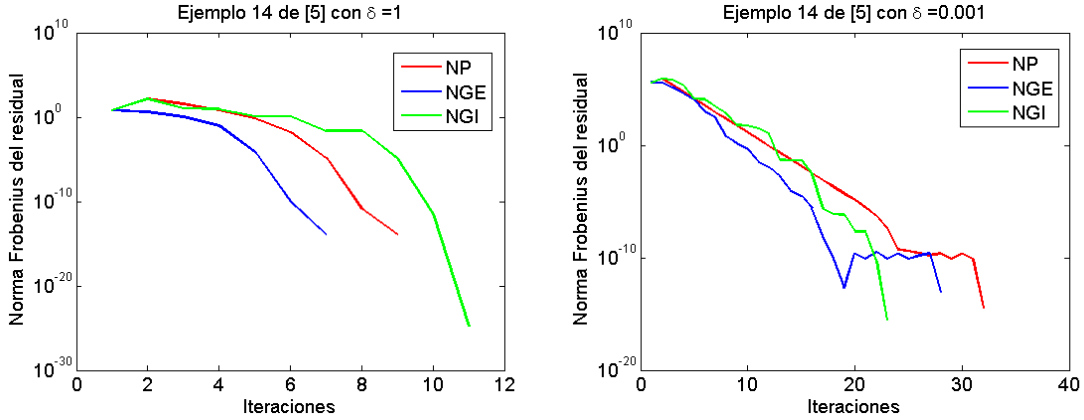
	NGE		**NGI**	
k	λ_k	$\|\mathcal{R}(X_k)\|_F$	λ_k	$\|\mathcal{R}(X_k)\|_F$
0	0.1570	6.0428×10^{0}	1	6.0428×10^{0}
1	0.7957	4.1299×10^{0}	2	1.3929×10^{2}
2	1.0657	1.0136×10^{0}	0.25	1.0890×10^{1}
3	1.0004	0.0958×10^{-2}	1	8.7544×10^{0}
4	1.0000	7.9181×10^{-5}	2	1.2820×10^{0}
5	1.0000	9.2282×10^{-11}	1	1.1805×10^{0}
6		1.1445×10^{-14}	2	2.5407×10^{-2}
7			1	2.5356×10^{-2}
8			1	1.2643×10^{-5}
9			1	3.1764×10^{-12}
10				1.9677×10^{-25}

Tabla 4.2: Comparación del tamaño de paso λ_k y $\|\mathcal{R}(X_k)\|_F$ en los métodos **NGE** y **NGI** para el Ejemplo 1 con $\delta = 1$.

puede observar que, a pesar de que el método **NGI** requirió más iteraciones para converger, éste logró una mayor disminución de la norma del residual en comparación con **NGE**. Adicionalmente,

tanto para **NGE** como para **NGI** se observa que, a medida que la norma del residual disminuye, las longitudes de paso se aproximan a 1 lo que indica que ambos métodos comienzan a comportarse como el método de Newton no globalizado, garantizando de esta manera su rápida convergencia en un entorno a la solución.

Por otro lado, como se mencionó anteriormente, a medida que $\delta \to 0$ los autovalores de las matrices $\mathcal{A}_k$ se aproximan al eje imaginario y por lo tanto el problema de hallar la solución estabilizadora de (1.1) se dificulta. Por lo anterior, se considero repetir el experimento usando $\delta = 10^{-3}$. Para realizar algunos comparaciones, considere la Figura 4.1 que muestra el comportamiento de la norma del residual por iteración para $\delta = 1$ (Imagen 4.1a) y para $\delta = 10^{-3}$ (Imagen 4.1b). Lo primero a observar es que todos los métodos requieren más iteraciones para lograr converger con $\delta = 10^{-3}$ que con $\delta = 1$. En segundo lugar, para $\delta = 1$, el método **NP** muestra la clásica convergencia con velocidad cuadrática pero esta característica se pierde usando $\delta = 10^{-3}$, donde se puede observar que la velocidad de convergencia para a ser lineal. El mismo comentario aplica para **NGE** y **NGI**.



(a) Con $\delta = 1$ (b) Con $\delta = 10^{-3}$

Figura 4.1: Norma del residual por iteración para los métodos **NP**, **NGE** y **NGI** en el Ejemplo 1 con $\delta = 1$ y $\delta = 10^{-3}$.

Para $\delta = 10^{-3}$, que genera una ARE más difícil de resolver, el método **NGI** requirió de menos iteraciones que **NP** y **NGE** para lograr convergencia. Por otro lado, en la Imagen 4.1b se puede observar que la disminución de la norma del residual para **NP** tiene un comportamiento monótono hasta la iteración 22, mientras que los métodos **NGI** y **NGE** siempre son no monótonos. Lo anterior contrasta con lo observado en la Imagen 4.1a, en donde sólo el método **NGI** presenta un comportamiento no monótono. Es importante destacar que todos los métodos convergieron a la solución estabilizadora de la ecuación de Riccati, independientemente del valor de δ.

Ejemplo 2. En este experimento se utilizó el Ejemplo 20 de [26] que describe un problema de sistemas de control dinámico que proviene de una aplicación asociada a un modelo matemático para una planta de energía. La ARE asociada a este ejemplo es un problema muy mal condicionado. Los detalles de la aplicación se pueden ver en [26]. Para generar las matrices que conforman a la ARE, se requiere de una serie de parámetros ligados a la aplicación. El archivo de parámetros empleados en este ejemplo se puede obtener como parte de la rutina `carex.m`. Las dimensiones de las matrices que componen a la ARE a resolver, dependen de un parámetro l tal y como se expresa la Tabla 4.3. El parámetro *tol* se fijó en 10^{-10} y el máximo de iteraciones fue de 20. El X_0 obtenido por el Algoritmo 6, que garantiza que la matriz $\mathcal{A}_0$ sea estable, generó un residual inicial cuya norma estuvo por el orden de 10^{20} y todos los métodos se detuvieron por alcanzar el máximo de iteraciones. Sin embargo, se constató que la matriz A es estable y por tanto fue factible

n	m	q	Parámetro
$2l-1$	l	l	$l = 211$

Tabla 4.3: Dimensiones y parámetros de la ARE del Ejemplo 2.

usar $X_0 = 0$ como iterado inicial. En la Figura 4.2 se muestra la norma Frobenius del residual en las primeras 7 iteraciones. Allí se puede observar que los métodos **NP** y **NGE** se estancaron cuando la norma del residual era del orden de 10^{-3} mientras que el método **NGI** converge en pocas iteraciones.

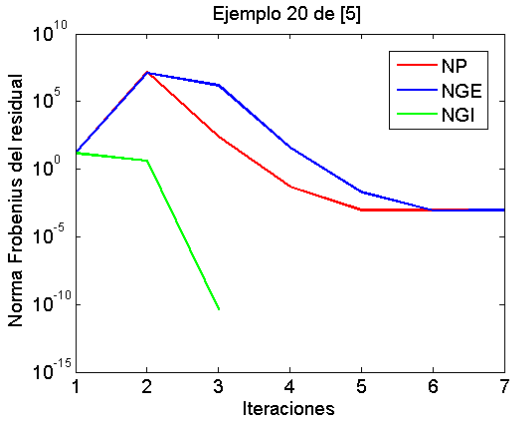


Figura 4.2: Norma del residual por iteración para los métodos **NP**, **NGE** y **NGI** en el Ejemplo 2 con $l = 211$.

Conviene recordar que $\|\mathcal{R}(X_k)\|_F$ se cálcula de forma explícita para **NP** y **NGE** y para **NGI** se emplea la forma implícita descrita en la expresión (3.7). Eso explica el por qué la norma del residual en X_1 es diferente para **NP** y **NGI**, a pesar de que en el método **NGI** el X_1 se genera con un paso del método **NP**. Este discrepancia sólo se observó en este ejemplo, pues sólo en este ejemplo se tienen problemas graves de condicionamiento tal y como se menciona en [26]. Sin embargo conviene señalar que, al igual que en los ejemplos anteriores, el método **NGI** convergió a la solución estabilizadora de la ARE.

Ejemplo 3. Para este experimento se consideró el Ejemplo 15 de [26]. Corresponde a un modelo de control dinámico asociado a la posición y velocidad de N vehículos en una vía de alta velocidad. Las matrices de coeficientes de la ARE fueron generadas mediante la rutina `carex.m`. El iterado inicial fue construido mediante el Algoritmo 6 con $\beta = \frac{\|A\|_F}{10}$. Para todos los valores de N el parámetro *tol* fue de 10^{-13} y el máximo de iteraciones fue de 20. La Tabla 4.4 muestra las dimensiones de las matrices que conforman a la ARE que son dependientes del valor de N.

n	m	q	Parámetro
$2N-1$	N	$N-1$	$N = 15$
			$N = 50$
			$N = 200$

Tabla 4.4: Dimensiones y parámetros de la ARE del Ejemplo 3.

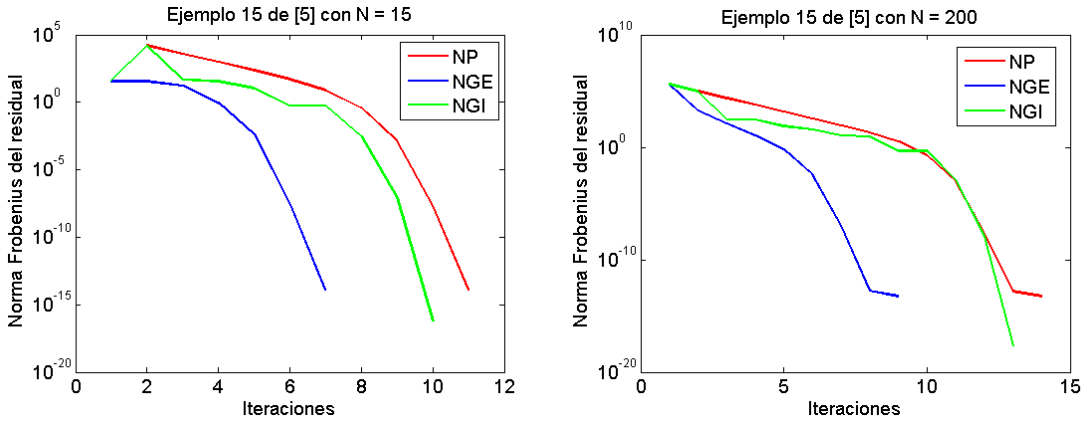
En la Tabla 4.5 se reportan el número de iteraciones requeridos para la convergencia en **NP**, **NGE** y **NGI** (valor de k) así como el último residual en norma de Frobenius alcanzado por

cada método ($\|\mathcal{R}(X_k)\|_F$) para tres valores distintos de N. Con $N = 15$ y $N = 200$, los métodos globalizados **NGE** y **NGI** tomaron menos iteraciones para alcanzar la tolerancia establecida que el **NP** sin embargo fue **NGE** el que obtuvo el menor número de iteraciones en todos los casos. Cabe destacar que el **NGI** logró una mayor reducción de la norma del residual en el total de los casos y nuevamente se observó una velocidad de convergencia cuadrática en un entorno cercano a la solución.

	NP		**NGE**		**NGI**	
N	k	$\|\mathcal{R}(X_k)\|_F$	k	$\|\mathcal{R}(X_k)\|_F$	k	$\|\mathcal{R}(X_k)\|_F$
15	11	1.26×10^{-14}	7	1.29×10^{-14}	10	6.36×10^{-17}
50	9	4.26×10^{-14}	7	4.78×10^{-14}	11	2.87×10^{-28}
200	14	5.42×10^{-14}	9	5.61×10^{-14}	13	2.35×10^{-18}

Tabla 4.5: Desempeño de los métodos **NP**, **NGE** y **NGI** para el Ejemplo 3 para diferentes valores del parámetro N.

Para complementar esta información, las imágenes 4.3a y 4.3b muestran el comportamiento de $\|\mathcal{R}(X_k)\|_F$ por iteración para los casos $N = 15$ y $N = 200$.



(a) Con $N = 15$ (b) Con $N = 200$

Figura 4.3: Norma del residual por iteración para los métodos **NP**, **NGE** y **NGI** en el Ejemplo 3 con $N = 15$ y $N = 200$.

Dichas imágenes nos llevan a pensar que la generación del X_1 para el método **NGI** puede ser una de las causas que influyen para que dicho método, en algunos de los ejemplos presentados, requiera un poco más de iteraciones que su competidor, el método **NGE**. Es decir, el método con globalización exacta tiende a generan un mejor X_1, en el sentido que la norma del residual en X_1 es menor en **NGE** que en **NGI**.

5. Conclusiones. En este trabajo se propuso un esquema numérico para hallar una aproximación a la solución de la ecuación algebraica de Riccati. En este nuevo esquema, se adaptó la técnica de globalización inexacta basada en la condición de Armijo para la versión incremental del método de Newton. Lo anterior permitió evitar resolver el problema de minimización por iteración presente en otra propuesta encontrada en la bibliografía que adapta la globalización exacta al método de Newton. Esta diferencia entre las dos técnicas es realmente llamativa, pues la ecuación de Riccati es una ecuación matricial y el problema de minimización en las técnicas exactas requieren de la evaluación del residual por iteración. En este mismo orden de ideas, la

nueva propuesta evita el cálculo explícito del residual, lo cual también contribuye a la reducción de los costos por iteración. Mediante la experimentación numérica se pudo comprobar que el esquema presentado en este trabajo resulta una propuesta competitiva en comparación con el método de Newton con globalización exacta.

REFERENCIAS

[1] M. Alirezaei, S. Kanarachos, B. Scheepers and J.P. Maurice, "Experimental Evaluation of Optimal Vehicle Dynamic Control based on the State Dependent Riccati Equation Technique", *American Control Conference*, Washington, USA, pp. 17-19, June 2013.

[2] T. Çimen, "Survey of State-Dependent Riccati Equation in Nonlinear Optimal Feedback Control Synthesis", *Journal of Guidance, Control, and Dynamics*, vol. 35(4), pp. 1025-1047, 2012.

[3] S. Vaddi, P. K. Menon and E. J. Ohlmeyer, "Numerical State-Dependent Riccati Equation Approach for Missile Integrated Guidance Control", *Journal of Guidance, Control, and Dynamics*, vol. 32(2), pp. 699-703, 2009.

[4] D. C. Wang, L. Chen and H. Chen, "Research on Sensor Fault-Tolerant Control Based on Riccati Equation for Electric Power Steering", *Key Engineering Materials*, vol. 464, pp. 86-89, 2011.

[5] M. K. Siddiq, J.C. Fang and W. B. Yu, "SDRE Based Integrated Roll, Yaw and Pitch Controller Design for 122mm Artillery Rocket", *Applied Mechanics and Materials*, vol. 415, pp. 200-208, 2013.

[6] D. A. Bini, B. Iannazzo, and B. Meini, *Numerical Solution of Algebraic Riccati Equations*, Fundamentals of Algorithms, SIAM , Philadelphia, 2012.

[7] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*, Oxford University Press, New York, 1995.

[8] D. Kleinman, "On an iterative technique for Riccati equation computations", *IEEE Trans. Automat. Control*, vol. 13(1), pp. 114-115, 1968.

[9] F. Feitzinger, T. Hylla and E. W. Sachs, "Inexact Kleinman-Newton method for Riccati equations", *SIAM J. Matrix Anal. Appl.*, vol. 31, pp. 272-288, 2009.

[10] C. Kenney, A.J. Laub, and M. Wette, "Error bounds for Newton refinement of solutions to Algebraic Riccati equations", *Mathematics of Control, Signals, and System*, vol. 3, pp. 211-224, 1990.

[11] B. N. Datta, *Numerical methods for linear control systems: design and analysis*, Elsevier Academic Press, California, 2004.

[12] K. Jbilou and A. J. Riquet, "Projection methods for large Lyapunov matrix equations", *Linear Algebra and its Applications*, vol. 415, pp. 344-358, 2006.

[13] K. Jbilou, "Low rank approximate solutions to large Sylvester matrix equations", *Applied mathematics and computation*, vol. 177(1), pp. 365-376, 2006.

[14] T. Penzl, "A cyclic low rank Smith method for large sparse Lyapunov equations", *SIAM J. Sci. Comput.*, vol. 21, pp. 1401-1414, 1999.

[15] I. M. Jaimoukha and E. M. Kasenally, "Krylov subspace methods for solving large Lyapunov equations", *SIAM Journal on Numerical Analysis*, vol. 31(1), pp. 227-251, 1994.

[16] A. Lu, and E. Wachspress, "Solution of Lyapunov equations by alternating direction implicit iteration", *Comp. Math, Appl.*, vol. 21, pp. 43-58, 1991.

[17] R. Smith, "Matrix equation $XA + BX = C$", *SIAM J. Appl. Math.*, vol. 16, pp. 198-201, 1968.

[18] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Classics in Applied Mathematics, SIAM, Philadelphia, 1996.

[19] J. Nocedal and S. Wright, *Numerical optimization*, Springer-Verlag, New York, 2006.

[20] P. Benner and R. Byers, "An Exact Line Search Method for Solving Generalized Continuous-Time Algebraic Riccati Equations", *IEEE Transactions on Automatic Control*, vol. 43(1), pp. 101-107, 1998.

[21] D. Bertsekas, *Nonlinear Programming*, Athena Scientifc, Massachusetts, 1999.

[22] N. J. Higham and H. M. Kim, "Solving a quadratic matrix equation by Newton's method with exact line searches", *SIAM Journal on Matrix Analysis and Applications*, vol. 23(2), pp. 303-316, 2001.

[23] W.F. Arnold and A. Laub, "Generalized eigenproblem algorithms and software for algebraic Riccati equations", *Proc. IEEE*, vol. 72, pp. 1746-1754, 1984.

[24] V. Sima, "An efficient Schur method to solve the stabilizing problem", *IEEE Transactions on Automatic Control*, vol. 26(3), pp. 724-725, 1981.

[25] E. S. Armstrong, "An extension of Bass' algorithm for stabilizing linear continuous constant systems", *IEEE Trans. Automat. Contr.*, vol. 20, pp. 153-154, 1975.

[26] P. Benner, A.J.Laub and V. Mehrmann, "A Collection of Benchmark Examples for the Numerical Solution of Algebraic Riccati Equations I: Continuous-Time Case", *Technical Report SPC 95-22, TU Chemnitz-Zwickau,* 1995.

[27] P. Benner, Technische Universität Chemnitz, Peter Benner, [En línea]. Disponible en: https://goo.gl/t1q2j3. [Accedido: 10-Oct-2017]