Universidad de Valladolid

Facultad de Ciencias

# TRABAJO DE FIN DE GRADO

Grado en Matemáticas

# EL FLUJO DE TODA Y EL CÁLCULO DE LOS AUTOVALORES DE UNA MATRIZ SIMÉTRICA

***Autor: Salvador Gil López***

***Tutor: María Paz Calvo Cabrero***

# Índice

# Índice de figuras

# Fe de erratas

July 22, 2021

En la página 7, en el enunciado del Teorema 1.3.1, donde dice "de de", debería decir "de".

En la página 40, en el enunciado del Teorema 3.1.3, donde dice "la la", debería decir "la".

En la página 41, en la ecuación (39), donde dice

$$U_{k+1} = (I - \frac{1}{2}hB_{k+\frac{1}{2}})^{-1}(I + \frac{1}{2}hB_{k+\frac{1}{2}}),$$

debería decir

$$U_{k+1} = (I - \frac{1}{2}hB_{k+\frac{1}{2}})^{-1}(I + \frac{1}{2}hB_{k+\frac{1}{2}})U_k.$$

En la Sección 2.1, la matriz definida en la ecuación (4) como $U(t)$, y que posteriormente se utiliza en el Teorema 2.1.1, sería preferible denotarla por $V(t)$.

En las Secciones 3.1 y 3.2, donde se utiliza $U(t)$ para denotar la matriz definida en la ecuación (4), y, se utiliza $U_k$ para denotar las aproximaciones numéricas a esta matriz, sería preferible utilizar $V(t)$ y $V_k$ respectivamente.

# Introducción

El problema de calcular numéricamente los autovalores de una matriz simétrica mediante el algoritmo QR se puede interpretar como el problema de hallar la solución en tiempos enteros de cierto sistema diferencial matricial: el flujo de Toda.

Este trabajo se presenta como la culminación de mis estudios en el grado de matemáticas por lo que se recomienda tener nociones básicas de cálculo numérico, ecuaciones diferenciales, álgebra y análisis.

El trabajo está estructurado en cuatro capítulos. En el primer capítulo se hace una descripción del algoritmo QR y se incluyen una serie de resultados teóricos sobre la transformación de matrices simétricas en matrices tridiagonales mediante transformaciones ortogonales, y sobre la aplicación del algoritmo QR a este tipo de matrices.

En el segundo capítulo se describen las ecuaciones que generan el flujo de Toda y se prueban una serie de resultados teóricos que abren la posibilidad de calcular los autovalores de una matriz tridiagonal simétrica mediante la integración del flujo de Toda con métodos numéricos adecuados.

En el tercer capítulo se presentan los métodos de integración *isoespectrales*, que permiten conservar los autovalores de la matriz inicial durante la integración del flujo de Toda.

En el cuarto capítulo se presentan los resultados numéricos que se han obtenido al aplicar, por un parte, el algoritmo QR, y por otra, dos métodos de integración para el flujo de Toda, para dos familias de matrices tridiagonales simétricas.

Por último, se incluye un apéndice con el código de los diferentes programas de Matlab utilizados en los experimentos numéricos.

# 1. El algoritmo QR

En este capítulo se describirá el algoritmo QR para el cálculo de los autovalores de una matriz $A$, siguiendo [Watkins]. Mediante un proceso iterativo se pretende obtener una matriz triangular superior $T$ ortogonalmente semejante con $A$, que tendrá por tanto sus mismos autovalores. Los elementos diagonales de $T$ proporcionan los autovalores de $A$.

## 1.1. Descripción del algoritmo QR

Sea $A \in \mathbb{R}^{n\times n}$ invertible.

Partiendo de $A_0 = A$, el algoritmo QR va a generar una sucesión de matrices $(A_m)_{m=0}^{M}$ repitiendo para $m = 1, \ldots, M$ los pasos siguientes:

1. **Factorización QR de $A_{m-1}$**: Se encuentran matrices $Q_m$ y $R_m$ únicas, con $Q_m$ ortogonal y $R_m$ triangular superior con todos los elementos de la diagonal positivos, tales que $A_{m-1} = Q_m R_m$.
2. **Cálculo de $A_m$**: Se define el nuevo iterante como $A_m = R_m Q_m$

Por ser $Q_m$ ortogonal se tiene que $Q_m^T Q_m = I_n$ y, por tanto,

$$A_m = R_m Q_m = I_n R_m Q_m = Q_m^T Q_m R_m Q_m = Q_m^T A_{m-1} Q_m.$$

Se tiene entonces que todas las matrices $(A_m)_{m=0}^{M}$ son ortogonalmente semejantes y, por tanto, todas comparten autovalores.

El algoritmo QR en su forma más básica es muy ineficiente por dos razones principales:

1. El número de operaciones de cada iteración QR es muy alto. Cada factorización QR supone hacer $\frac{4}{3}n^3$ operaciones y el producto matricial que le sigue requiere de $O(n^3)$ operaciones.
2. La convergencia a forma triangular es muy lenta, hay que realizar un número demasiado alto de iteraciones para que el tamaño de los elementos de la parte estrictamente triangular inferior de la matriz sea suficientemente pequeño y podamos aceptar los valores encontrados en la diagonal como los autovalores de A.

Por tanto, necesitamos reducir el número de operaciones necesario en cada iteración y que la convergencia sea más rápida.

El número de operaciones de cada iteración QR puede ser reducido drásticamente si primero se reduce la matriz a forma de Hessenberg y a continuación se aplica el algoritmo QR a la matriz obtenida tras hacer esta reducción.

Como veremos en la Sección 1.3 esta reducción funciona porque la forma de Hessenberg se mantiene en cada iteración del algoritmo QR.

## 1.2. Matrices en forma de Hessenberg

**Definición 1.2.1.** *Sea $A \in \mathbb{R}^{n\times n}$. Decimos que $A$ tiene forma de Hessenberg superior si $a_{ij} = 0$ para $i > j + 1$.*

**Teorema 1.2.1.** *Sea $A \in \mathbb{R}^{n\times n}$. Existe una matriz $H \in \mathbb{R}^{n\times n}$ ortogonalmente semejante con $A$ que tiene forma de Hessenberg superior.*

*Demostración.* La demostración es constructiva y se basa en multiplicar a la matriz $A$ por la izquierda y por la derecha por reflectores de Householder adecuados para hacer ceros por debajo de la primera subdiagonal.

Es conocido [Watkins] que dado un vector $\mathbf{y} \in R^k$ arbitrario, no nulo, existe un reflector de Householder que, al multiplicar a $\mathbf{y}$ por este reflector, se obtiene un vector con todas las componentes nulas excepto la primera. Teniendo esto en cuenta, primero tomamos la matriz ortogonal

$$Q_1 = \left[\begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & \hat{Q}_1 & \\ 0 & & & \end{array}\right] \in \mathbb{R}^{}$$

donde $\hat{Q}_1$ es el reflector de Householder de dimensión $n - 1$ que anula las componentes $(3, 1), \dots, (n, 1)$ de $A$.

Es claro entonces que la primera columna de $Q_1A$ tiene la forma buscada y que al multiplicar por la derecha por $Q_1^{-1} = Q_1^T = Q_1$ no se destruyen los ceros de la primera columna (por la forma que tiene la primera columna de la matriz $Q_1$).

Se tiene entonces que

$$Q_1AQ_1 = \begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix},$$

donde los $*$ representan elementos en principio no nulos.

A continuación tomamos $\hat{Q}_2$, el reflector de Householder de dimensión $n-2$ que hace nulos los elementos de la segunda columna de $Q_1AQ_1$ a partir del cuarto.

Sea

$$Q_2 = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \hat{Q}_2 & \\ 0 & 0 & & & \end{array}\right].$$

Tendremos que

$$Q_2Q_1AQ_1 = \begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & * & \cdots & * \end{bmatrix},$$

y, de nuevo al multiplicar por la derecha por $Q_2$ no se modifican las primeras dos columnas de la matriz anterior y la matriz producto $Q_2Q_1AQ_1Q_2$ tiene sus dos primeras columnas en la forma deseada.

Continuando con el proceso se va construyendo $\hat{Q}_k$, el reflector de Householder de dimensión $n-k$ que transforma en ceros los elementos $k+2, \ldots, n$ de la $k$-ésima columna de la matriz $Q_{k-1}\cdots Q_2Q_1AQ_2Q_2\cdots Q_{k-1}$, y se forma $Q_k$, que multiplicando por la izquierda deja invariantes las $k$ primeras filas y multiplicando por la derecha deja invariantes las $k$ primeras columnas. Así

se llega a

$$Q_{n-1}\cdots Q_2Q_1AQ_2Q_2\cdots Q_{n-1} = \begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & * & * \end{bmatrix},$$

que proporciona la matriz $H$ con forma de Hessenberg superior y ortogonalmente semejante a la matriz de partida $A$.

□

Es importante darse cuenta de que el primer reflector requiere $(n-1)^2$ multiplicaciones, el segundo $(n-2)^2$ y así sucesivamente hasta el último reflector que requiere $2^2$ operaciones, es decir $(n-1)^2+(n-2)^2+\cdots+2^2 = O(n^3)$. Por otra parte el producto por la izquierda y por la derecha requieren de $\frac{4}{3}n^3$ y $2n^3$ operaciones respectivamente, por lo que el coste total de la transformación es de $O(n^3)$ operaciones.

## 1.3. El algoritmo QR para matrices de Hessenberg

Primero, incluimos un par de resultados auxiliares.

**Proposición 1.3.1.** *Sea $U$ una matriz de tamaño $n \times n$ triangular superior e invertible. Se tiene que $U^{-1}$ es triangular superior.*

*Demostración.* Realizamos inducción sobre la dimensión de la matriz, $n$.

Para $n = 2$, $U = \begin{bmatrix} a & b \\ 0 & d \end{bmatrix}$ con $ad \neq 0$ y es obvio que $U^{-1} = \frac{1}{ad}\begin{bmatrix} d & -b \\ 0 & a \end{bmatrix}$ es triangular superior.

Suponemos que la hipótesis es cierta para $n-1$ y estudiamos el caso general.

Sea

$$U = \left[\begin{array}{c|c} u_{11} & \mathbf{b^T} \\ \hline 0 & \\ \vdots & \hat{U} \\ 0 & \end{array}\right],$$

con $u_{11} \in \mathbb{R}$ no nulo, $\mathbf{b^T} \in \mathbb{R}^{1\times(n-1)}$ y $\hat{U} \in \mathbb{R}^{(n-1)\times(n-1)}$ triangular superior e invertible, y sea

$$M = \left[\begin{array}{c|c} m_{11} & \mathbf{c^T} \\ \hline \mathbf{d} & \hat{M} \end{array}\right],$$

con $m_{11} \in \mathbb{R}$, $\mathbf{c^T} \in \mathbb{R}^{1\times(n-1)}$, $\mathbf{d} \in \mathbb{R}^{(n-1)\times 1}$ y $M_{22} \in \mathbb{R}^{(n-1)\times(n-1)}$.

Se cumple que

$$UM = \left[\begin{array}{c|c} u_{11}m_{11} + \mathbf{b^T d} & u_{11}\mathbf{c^T} + \mathbf{b^T}\hat{M} \\ \hline \hat{U}\mathbf{d} & \hat{U}\hat{M} \end{array}\right],$$

de modo que para que $UM = I_n$, por un lado debe ser $\hat{U}\mathbf{d} = \mathbf{0}$ de donde deducimos $\mathbf{d} = \mathbf{0}$.

Por otro lado, $\hat{U}\hat{M} = I_{n-1}$, es decir, $\hat{M} = \hat{U}^{-1}$. Como $\hat{U}$ es triangular superior aplicamos la hipótesis de inducción y deducimos que $\hat{M}$ es triangular superior.

Esto, junto con el hecho de que $\mathbf{d} = \mathbf{0}$ prueba que $U^{-1} = M$ superior. □

**Proposición 1.3.2.** *Sea $H \in \mathbb{R}^{n\times n}$ de Hessenberg superior y $R \in \mathbb{R}^{n\times n}$ triangular superior. Entonces $HR$ y $RH$ son ambas de Hessenberg superior.*

*Demostración.* Aplicamos inducción sobre la dimensión de la matriz $n$.

Para $n = 2$ la prueba es obvia pues todas las matrices de dimensión 2 son de Hessenberg superior.

Consideramos la hipótesis cierta para $n - 1$ y estudiamos el caso general.

Sean $H, R \in \mathbb{R}^{n\times n}$. Dividimos

$$H = \left[\begin{array}{c|c} h_{11} & \mathbf{b^T} \\ \hline \mathbf{c} & \hat{H} \end{array}\right],$$

donde $h_{11} \in \mathbb{R}$, $\mathbf{b^T} \in \mathbb{R}^{1\times(n-1)}$, $\mathbf{c} = \left[\begin{array}{ccccc} h_{21}, & 0, & \ldots, & 0 \end{array}\right]^T \in \mathbb{R}^{(n-1)x1}$ y $\hat{H} \in \mathbb{R}^{(n-1)\times(n-1)}$ tiene forma de Hessenberg.

De forma similar

$$R = \left[\begin{array}{c|c} r_{11} & \mathbf{d^T} \\ \hline \mathbf{0} & \hat{R} \end{array}\right],$$

donde $r_{11} \in \mathbb{R}$, $\mathbf{d^T} \in \mathbb{R}^{1\times(n-1)}$, y $\hat{R} \in \mathbb{R}^{(n-1)\times(n-1)}$ es triangular superior.

Tenemos que

$$HR = \left[\begin{array}{c|c} h_{11}r_{11} & h_{11}\mathbf{d^T} + \mathbf{b^T}\hat{R} \\ \hline r_{11}\mathbf{c} & \mathbf{cd^T} + \hat{H}\hat{R} \end{array}\right].$$

Por una parte

$$\mathbf{cd^T} = \begin{bmatrix} h_{21} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} r_{21}r_{12} & r_{13} & \cdots & r_{1n} \end{bmatrix} = \begin{bmatrix} h_{21}r_{12} & h_{21}r_{13} & \cdots & h_{21}r_{1n} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

que tiene forma de Hessenberg superior.

Por otra podemos aplicar la hipótesis de inducción a $\hat{H}\hat{R}$ y deducir que el producto tiene forma de Hessenberg superior.

La suma de ambas tiene forma de Hessenberg superior y junto con que $r_{11}\mathbf{c} = \begin{bmatrix} h_{21}r_{11}, & 0, & \ldots, & 0 \end{bmatrix}^T$ queda demostrado que $HR$ tiene forma de Hessenberg superior.

La demostración de que $RH$ es de Hessenberg superior es análoga. □

**Teorema 1.3.1.** *Sea $A_{m-1}$ una matriz invertible con forma de Hessenberg superior, y supongamos que $A_m$ se obtiene a partir de de $A_{m-1}$ mediante una iteración QR. Entonces $A_m$ es de Hessenberg superior.*

*Demostración.* Si $A_{m-1} = Q_m R_m$ es la factorización QR de $A_{m-1}$, $Q_m$ se puede reescribir como $Q_m = A_{m-1}R_m^{-1}$.

Por la Proposición 1.3.1, $R_m^{-1}$ es triangular superior.

Por la Proposición 1.3.2 deducimos que $Q_m$ es de Hessenberg superior y por lo mismo $A_m = R_m Q_m$ también lo es. □

En el teorema anterior hemos supuesto que $A$ es invertible. Sin embargo haciendo la factorización mediante rotaciones, como veremos a continuación, podemos evitar esta hipótesis.

Suponemos que $A$ es de Hessenberg superior. Podemos transformar $A$ en triangular superior aplicando $n-1$ rotaciones de Givens para hacer ceros en todos los elementos de la subdiagonal de $A$.

Tomamos $Q_1$ una rotación de manera que $Q_1^T A$ tenga un cero en la posición (2,1) y que al premultiplicar por $Q_1^T$ a la matriz $A$ sólo se modifiquen la primera y la segunda fila de la matriz.

Seguimos con $Q_2$, de manera que $Q_2^T Q_1^T A$ tiene un cero en la posición (3,2) y solo se han modificado las filas segunda y tercera de la matriz. Como la intersección de ambas filas con la primera columna son dos ceros, estos no serán modificados por la rotación. En particular el cero en la posición (2,1) se mantiene.

De esta manera conseguimos

$$R = Q_{n-1}^T Q_{n-2}^T \cdots Q_2^T Q_1^T A$$

triangular superior.

Sea

$$Q = Q_1 Q_2 \cdots Q_{n-2} Q_{n-1}.$$

Se cumple que $R = Q^T A$ o, de otra manera, $A = QR$ con $Q$ ortogonal por ser cada una de las rotaciones ortogonales y $R$ triangular superior.

Ahora hacemos $A_1 = RQ_1 Q_2 \cdots Q_{n-1}$ para realizar el segundo paso de la primera iteración QR. Todo lo que tenemos que hacer es multiplicar a la derecha por $Q_1$,$Q_2$,...,$Q_{n-1}$ sucesivamente. Como $Q_1$ actúa sobre la primera y la segunda columna, y, a partir de las dos primeras posiciones hay ceros, estos no se ven afectados pero se puede generar un elemento no nulo en la posición (2,1). Y así pasa con cada una de las rotaciones.

De este modo, en la transformación de $R$ a $A_1$ los únicos ceros que se pueden modificar son los ceros de la subdiagonal. De donde se sigue que $A_1$ es de Hessenberg superior.

Podemos afirmar entonces que la forma de Hessenberg se preserva en cada una de las iteraciones QR con independencia de que la matriz A de partida sea invertible o no.

**Teorema 1.3.2.** *Una iteración QR aplicada a una matriz de Hessenberg superior requiere no más de $O(n^2)$ operaciones.*

*Demostración.* La construcción realizada más arriba requiere aplicar $n-1$ rotaciones por la izquierda seguidas de $n-1$ rotaciones por la derecha. El coste de esto es de $O(n)$ operaciones por cada rotación haciendo $O(n^2)$ operaciones en total. □

Tras estos resultados podemos afirmar que hemos mejorado el problema del elevado número de operaciones requerido para realizar una iteración QR. La reducción a forma de Hessenberg requiere $O(n^3)$ operaciones, pero sólo hay que hacerlo una vez. Con esta reducción cada iteración QR requiere de $O(n^2)$ operaciones.

## 1.4. El caso de matrices reales simétricas

Suponemos ahora que $A \in \mathbb{R}^{n\times n}$ es simétrica. Al realizar la transformación a forma de Hessenberg obtenemos una matriz tridiagonal y simetrica.

Ahora $A$ es de la forma $\left[\begin{array}{c|c} a_{11} & \mathbf{b^T} \\ \hline \mathbf{b} & \hat{A} \end{array}\right]$ con $\mathbf{b} \in \mathbb{R}^{(n-1)\times 1}$ y $\hat{A} \in \mathbb{R}^{(n-1)\times(n-1)}$.

Al multiplicar por el primer reflector de Householder obtenemos

$$A_1 = Q_1 A Q_1 = \left[\begin{array}{c|ccccc} a_{11} & -\tau_1 & 0 & \cdots & 0 \\ \hline -\tau_1 & & & & \\ 0 & & & & \\ \vdots & & & \hat{A}_1 & \\ 0 & & & & \end{array}\right].$$

Además, puesto que la matriz $Q_1 A Q_1^T$ es simétrica ahorramos operaciones pues no tenemos que hacer $\mathbf{b^T}\hat{Q}_1$ como en el caso general.

El esfuerzo principal en este paso se hace al calcular $\hat{A}_1 = \hat{Q}_1 \hat{A} \hat{Q}_1$. Para reducir los costes necesitamos aprovechar la simetría de la matriz. Si no este paso requerirá $8n^2$ operaciones.

$\hat{Q}_1$ es un reflector dado por $\hat{Q}_1 = I_n - \gamma \mathbf{u}\mathbf{u^T}$. Por tanto

$$\hat{A}_1 = (I_n - \gamma \mathbf{u}\mathbf{u^T})\hat{A}(I_n - \gamma \mathbf{u}\mathbf{u^T}) = \hat{A} - \gamma \hat{A}\mathbf{u}\mathbf{u^T} - \gamma \mathbf{u}\mathbf{u^T}\hat{A} + \gamma^2 \mathbf{u}\mathbf{u^T}\hat{A}\mathbf{u}\mathbf{u^T}.$$

Sea $\mathbf{v} = -\gamma \hat{A}\mathbf{u}$. Tenemos que

$$-\gamma \hat{A}\mathbf{u}\mathbf{u^T} = \mathbf{v}\mathbf{u^T}, -\gamma \mathbf{u}\mathbf{u^T}\hat{A} = \mathbf{u}\mathbf{v^T} y \gamma^2 \mathbf{u}\mathbf{u^T}\hat{A}\mathbf{u}\mathbf{u^T} = -\gamma \mathbf{u}\mathbf{u^T}\mathbf{v}\mathbf{u^T}.$$

Llamando $\alpha = \frac{1}{2}\gamma \mathbf{u^T}\mathbf{v}$ podemos escribir $-\gamma \mathbf{u}\mathbf{u^T}\mathbf{v}\mathbf{u^T} = 2\alpha \mathbf{u}\mathbf{u^T}$, y así $\hat{A}_1 = \hat{A} + \mathbf{v}\mathbf{u^T} + \mathbf{u}\mathbf{v^T} + 2\alpha \mathbf{u}\mathbf{u^T}$.

Por último, llamando $\mathbf{w} = \mathbf{v} + \alpha \mathbf{u}$ y tenemos $\hat{A}_1 = \hat{A} + \mathbf{w}\mathbf{u^T} + \mathbf{u}\mathbf{w^T}$.

Por simetría sólo tenemos que operar sobre la diagonal y el triángulo inferior. Por tanto el número de operaciones total es $4(n-1)\frac{n}{2} \approx 2n^2$. Esto no incluye las operaciones necesarias para calcular $\mathbf{w}$ que eleva el número de operaciones total a $4n^2$.

El segundo paso es idéntico al primero excepto que se opera sobre la matriz interior $\hat{A}_1$. Así, el número aproximado de operaciones necesarias es $4(n-1)^2$.

Tras $(n-2)$ pasos la reducción a forma tridiagonal ha acabado y el número de operaciones total es de $4(n^2 + (n-1)^2 + \cdots) \approx \frac{4}{3}n^3$

## 1.5. El algoritmo QR para matrices simétricas

Una vez reducida la matriz simétrica a forma tridiagonal nos encontramos con que al aplicar el algoritmo QR tenemos una situación aún más favorable con respecto al caso de matrices de Hessenberg no simétricas. Primero, la reducción a forma tridiagonal necesita de $O(n^3)$ operaciones. Además, como el algoritmo QR preserva la simetría y la forma de Hessenberg deducimos que preserva la forma tridiagonal, y una iteración QR en este tipo de matrices requiere de $O(n)$ operaciones.

En particular si $A$ es simétrica la propiedad de simetría se mantiene para todas las matrices $A_m$ y la sucesión converge a una matriz diagonal con los autovalores en la diagonal ordenados por orden de magnitud de mayor a menor.

# 2. El flujo de Toda

En este capítulo describimos las ecuaciones que definen el flujo de Toda y probamos algunos resultados teóricos.

Consideramos el sistema de ecuaciones diferenciales

$$\begin{aligned} \frac{d}{dt}a_k &= 2(b_k^2 - b_{k-1}^2), \quad k = 1, \ldots, n, \\ \frac{d}{dt}b_k &= b_k(a_{k+1} - a_k), \quad k = 1, \ldots, n-1. \end{aligned} \tag{1}$$

donde interpretamos que $b_0 = b_n = 0$, convenio que se mantendrá a lo largo de toda la memoria.

Asociado a (1) podemos considerar la matriz de $R^{n\times n}$

$$L(t) \equiv L(\mathbf{a}(t), \mathbf{b}(t)) = \begin{bmatrix} a_1(t) & b_1(t) & & & \\ b_1(t) & & & 0 & \\ & & \ddots & & \\ & 0 & & & b_{n-1}(t) \\ & & & b_{n-1}(t) & a_n(t) \end{bmatrix}, \tag{2}$$

en la que se recogen todas las funciones $a_k(t), 1 \le k \le n$ y $b_k(t), 1 \le k \le n-1$.

Veremos, en los resultados que siguen, que el flujo dado por (1) tiene las siguientes propiedades:

1. Los autovalores de $L(t)$ son independientes de $t$, es decir, el flujo es *isoespectral*.
2. $\lim_{t\to\infty} b_k(t) = 0, k = 1, \ldots, n-1$.

La propiedad 2 implica que $L(t)$ converge cuando $t \to \infty$ hacia una matriz diagonal, que será, por la propiedad 1, una permutación de los autovalores de la matriz inicial $L(0)$.

El problema de calcular los autovalores de una matriz tridiagonal simétrica se puede abordar resolviendo un problema de ecuaciones diferenciales ordinarias en un intervalo de tiempo suficientemente largo. Tomando la matriz $L(0)$ como dato inicial para el sistema diferencial (1).

## 2.1. Algunas propiedades del flujo de Toda

Dada la matriz $L(t)$ sea $B(L(t)) = L(t)^+ - L(t)^-$, donde $L(t)^+$ denota la parte triangular superior de $L(t)$ y $L(t)^-$ la parte triangular inferior de $L(t)$, es decir,

$$B(L(t)) = \begin{bmatrix} 0 & b_1(t) & & & & \\ -b_1(t) & 0 & b_2(t) & & 0 & \\ & -b_2(t) & & & & \\ & & & \ddots & & \\ & & 0 & & & b_{n-1}(t) \\ & & & & -b_{n-1}(t) & 0 \end{bmatrix},$$

y sea

$$W = [B, L] = BL - LB,$$

el conmutador de $B$ y $L$.

Operando vemos que si $W = (w_{ij})_{1 \leq i,j \leq n}$, tenemos

$$\begin{aligned} w_{ii} &= (b_i^2 - b_{i-1}^2) - (b_{i-1}^2 - b_i^2) = 2(b_i^2 - b_{i-1}^2), \\ w_{ii+1} &= b_i a_{i+1} - b_i a_i = b_i(a_{i+1} - a_i), \\ w_{i+1i} &= -b_i a_i - (-b_i a_{i+1}) = b_i(a_{i+1} - a_i), \\ w_{i+2i} &= -b_{i+1} b_i - (-b_i b_{i+1}) = 0, \\ w_{ii+2} &= b_i b_{i+1} - b_i b_{i+1} = 0, \end{aligned}$$

y el resto de elementos son ceros por ser $B$ y $L$ tridiagonales.

Así, podemos reescribir (1) de la forma

$$\frac{d}{dt} L = BL - LB. \tag{3}$$

Sea $U(t)$ la solución de la ecuación diferencial matricial

$$\frac{d}{dt} U = BU \quad \text{con } U(0) = I_n. \tag{4}$$

Por ser $B(t)$ antisimétrica, se tiene que

$$\frac{d}{dt} U^T = (\frac{d}{dt} U)^T = (BU)^T = -U^T B,$$

es decir,

$$\frac{d}{dt}(U^TU) = (\frac{d}{dt}U^T)U + U^T(\frac{d}{dt}U) = -U^TBU + U^TBU = 0,$$

lo cual implica que $U^TU$ es constante, y dado que $U(0) = I_n$ se deduce que $U^TU = I_n$ y, por tanto, $U(t)$ es ortogonal para todo $t \geq 0$.

**Teorema 2.1.1.** *El flujo definido por* (1) *es* isoespectral*, es decir, los autovalores de* $L(t)$ *coinciden con los de* $L(0)$ *para todo* $t \geq 0$*. (*$L(t)$ *y* $L(0)$ *tienen el mismo espectro).*

*Demostración.* Teniendo en cuenta que, la solución de (4), $U(t)$, es ortogonal,

$$\begin{aligned}\frac{d}{dt}(U^{-1}LU) = \frac{d}{dt}(U^TLU) &= (\frac{d}{dt}U^T)LU + U^T(\frac{d}{dt}L)U + U^TL(\frac{d}{dt}U)\\ &= (\frac{d}{dt}U^T)LU + U^T(BL - LB)U + U^TL(\frac{d}{dt}U)\\ &= -U^TBLU + U^TBLU - U^TLBU + U^TLBU = 0.\end{aligned}$$

Es decir, $U^{-1}(t)L(t)U(t)$ es una matriz constante, y, por tanto,

$$U^{-1}(t)L(t)U(t) = U^{-1}(0)L(0)U(0).$$

Teniendo en cuenta otra vez que $U(0) = I_n$ se deduce fácilmente que

$$L(t) = U(t)L(0)U^{-1}(t) = U(t)L(0)U^T(t), \tag{5}$$

es decir, $L(t)$ es ortogonalmente semejante a $L(0)$ para cualquier $t$ y, por tanto, tiene los mismos autovalores que $L(0)$. □

Sea $L$ una matriz tridiagonal simétrica como en (2), sean $\lambda_1, \lambda_2, \ldots, \lambda_n$ los autovalores de $L$, que suponemos ordenados de forma decreciente y denotemos por $\mathbf{u} = [u_{11}, \ldots, u_{1n}]^T$ al vector de las primeras componentes de los autovectores de $L$, y por $\|\cdot\|_2$ a la norma Euclídea en $\mathbb{R}^n$.

Sea

$$\mathcal{M} = \{(\lambda_1, \ldots, \lambda_n; x_1, \ldots, x_n) \in \mathbb{R}^{2n} : \lambda_1 > \lambda_2 > \cdots > \lambda_n; x_i > 0, \|\mathbf{x}\|_2 = 1\}.$$

**Teorema 2.1.2.** *La aplicación*

$$(a_1, \ldots, a_n; b_1, \ldots, b_{n-1}) \to (\lambda_1, \ldots, \lambda_n; u_{11}, \ldots, u_{1n})$$

*es un difeomorfismo del espacio de las matrices tridiagonales reales simétricas con* $b_i > 0$ *en* $\mathcal{M}$.

*Demostración.* Por el teorema espectral,

$$L = U\Lambda U^T,$$

con $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ y $U = (u_{ij})$ la matriz ortogonal cuyas columnas son los autovectores de $L$.

Igualando elementos en las posiciones $(i, i)$ y aquellos en las posiciones $(i, i+1)$ en ambos lados de la igualdad, se deduce que

$$\begin{aligned} a_i = L_{ii} &= \sum_{j=1}^{n} \lambda_j u_{ij}^2, \qquad\qquad i = 1, 2, \ldots, n, \\ b_i = L_{ii+1} &= \sum_{j=1}^{n} \lambda_j u_{ij} u_{i+1j}, \qquad i = 1, 2, \ldots, n-1. \end{aligned} \tag{6}$$

De la ecuación de los autovalores $L\mathbf{u}_j = \lambda_j \mathbf{u}_j$, igualando la componente $i$-ésima obtenemos

$$b_{i-1}u_{i-1i} + a_i u_{ij} + b_i u_{i+1j} = \lambda_j u_{ij}.$$

Reordenando esta igualdad,

$$b_i u_{i+1j} = (\lambda_j - a_i)u_{ij} - b_{i-1}u_{i-1j}, \ 1 \le i \le n, 1 \le j \le n, \quad (b_0 = b_n = 0), \tag{7}$$

y utilizando la ortogonalidad de $U$ se deduce que

$$b_i^2 = b_i^2(\sum_{j=1}^{n} u_{i+1j}^2) = \sum_{j=1}^{n} [(\lambda_j - a_i)u_{ij} - b_{i-1}u_{i-1j}]^2, \qquad i = 1, 2, \ldots, n-1, \tag{8}$$

y

$$u_{i+1j} = \frac{[(\lambda_j - a_i)u_{ij} - b_{i-1}u_{i-1j}]}{b_i}, \qquad i = 1, 2, \ldots, n-1. \tag{9}$$

De hecho $a_1$ está determinado por $\lambda_1, \ldots, \lambda_n$ y $u_{11}, \ldots, u_{1n}$ a través de la identidad (6), para $i = 1$

$$a_1 = \sum_{j=1}^{n} \lambda_j u_{1j}^2, \tag{10}$$

y, $b_1$ y $u_{2j}$, para $1 \leq j \leq n$, están dados por (8) y (9)

$$b_1^2 = \sum_{j=1}^{n} [(\lambda_j - a_1)u_{1j} - 0]^2, \qquad u_{2j} = \frac{[(\lambda_j - a_1)u_{1j} - 0]}{b_1}, \tag{11}$$

respectivamente. Utilizando (6), (7), (8) y (9) repetidamente se determinan de forma única $a_1, \ldots, a_n, b_1, \ldots, b_{n-1}$ en función de $\lambda_1, \ldots, \lambda_n, u_1, \ldots, u_n$.

Esto prueba que la aplicación es inyectiva.

Ahora veremos el carácter sobreyectivo. Tomamos $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ y $u_{11}, \ldots, u_{1n}$ con $u_{1i} > 0$ y $\sum_{i=1}^{n} u_{1i}^2 = 1$.

Sea

$$a_1 = \sum_{i=1}^{n} \lambda_i u_{1i}^2.$$

Se cumple que $\sum_{i=1}^{n} (\lambda_i u_{1i} - a_1 u_{1i})^2 > 0$, pues en otro caso $\lambda_i u_{1i} = a_1 u_{1i}$ para todo $1 \leq i \leq n$, lo que es imposible.

Sean

$$b_1 = \sqrt{\sum_{i=1}^{n} (\lambda_i u_{1i} - a_1 u_{1i})^2},$$

es decir,

$$b_1^2 = \sum_{i=1}^{n} (\lambda_i u_{1i} - a_1 u_{1i})^2, \tag{12}$$

y

$$u_{2i} = \frac{(\lambda_i - a_1)u_{1i}}{b_1}, \qquad i = 1, \ldots, n. \tag{13}$$

Tenemos que se cumplen las igualdades

$$\begin{aligned}\sum_{i=1}^{n} u_{1i}u_{2i} &= \frac{1}{b_1}\sum_{i=1}^{n} u_{1i}(\lambda_i - a_1)u_{1i} = \frac{1}{b_1}(\sum_{i=1}^{n}\lambda_i u_{1i}^2 - a_1\sum_{i=1}^{n} u_{1i}^2) \\ &= \frac{1}{b_1}(\sum_{i=1}^{n}\lambda_i u_{1i}^2 - a_1) = 0,\end{aligned} \tag{14}$$

porque $\sum_{i=1}^{n} u_{1i}^2 = \|\mathbf{u}_1\|_2 = 1$, de donde deducimos que $\mathbf{u}_1$ y $\mathbf{u}_2$ son ortogonales entre sí.

Además usando (12), (13) y (14) y la ortogonalidad de $\mathbf{u}_1$ y $\mathbf{u}_2$

$$\begin{aligned}b_1 &= \frac{1}{b_1}\sum_{i=1}^{n}(\lambda_i u_{1i} - a_1 u_{1i})^2 = \frac{1}{b_1}\sum_{i=1}^{n}(\lambda_i - a_1)^2 u_{1i}^2 \\ &= \sum_{i=1}^{n}(\frac{(\lambda_i - a_1)}{b_1}u_{1i})(\lambda_i - a_1)u_{1i} = \sum_{i=1}^{n} u_{2i}(\lambda_i - a_1)u_{1i} \\ &= \sum_{i=1}^{n}\lambda_i u_{1i}u_{2i} - a_1\sum_{i=1}^{n} u_{1i}u_{2i} = \sum_{i=1}^{n}\lambda_i u_{1i}u_{2i}.\end{aligned} \tag{15}$$

Por último, de (13) y (15) se deduce que $\|\mathbf{u}_2\|_2 = 1$ pues

$$\sum_{i=1}^{n} u_{2i}^2 = \frac{1}{b_1}\sum_{i=1}^{n} u_{2i}(\lambda_i - a_1)u_{1i} = 1.$$

Sea ahora

$$a_2 = \sum_{i=1}^{n}\lambda_i u_{2i}^2. \tag{16}$$

Si $n > 2$, $\sum_{i=1}^{n}(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i})^2 > 0$. De otra manera $a_1 u_{1i} + b_1 u_{2i} = \lambda_i u_{1i}$, $b_1 u_{1i} + a_2 u_{2i} = \lambda_i u_{2i}$ para todo $1 \leq i \leq n$ y se tendría que la matriz $\begin{bmatrix} a_1 & b_1 \\ b_1 & a_2 \end{bmatrix}$ tiene más de dos autovalores distintos lo que es imposible.

Ahora,

$$b_2 = \sqrt{\sum_{i=1}^{n}(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i})^2} > 0,$$

y

$$u_{3i} = \frac{(\lambda_i - a_2)u_{2i} - b_1 u_{1i}}{b_2} \quad i = 1, \ldots, n,$$

Igual que antes

$$\begin{aligned}
\sum_{i=1}^{n} u_{1i}u_{3i} &= \frac{1}{b_2}(\sum_{i=1}^{n} \lambda_i u_{2i}u_{1i} - a_2 \sum_{i=1}^{n} u_{2i}u_{1i} - b_1 \sum_{i=1}^{n} u_{1i}^2) \\
&= \frac{1}{b_2}(\sum_{i=1}^{n} \lambda_i u_{2i}u_{1i} - b_1) = 0, \\
\sum_{i=1}^{n} u_{2i}u_{3i} &= \frac{1}{b_2}(\sum_{i=1}^{n} \lambda_i u_{2i}^2 - a_2 \sum_{i=1}^{n} u_{2i}^2 - b_1 \sum_{i=1}^{n} u_{1i}u_{2i}) \\
&= \frac{1}{b_2}(\sum_{i=1}^{n} \lambda_i u_{2i}^2 - a_2) = 0,
\end{aligned}$$

usando (14), (15) y que $\|\mathbf{u}_1\|_2 = 1$, junto con lo anterior deducimos que $\mathbf{u}_1$, $\mathbf{u}_2$ y $\mathbf{u}_3$ son ortogonales dos a dos.

Además,

$$\begin{aligned}
b_2 &= \frac{1}{b_2}\sum_{i=1}^{n}(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i})^2 \\
&= \sum_{i=1}^{n}(\frac{(\lambda_i - a_2)u_{2i} - b_1 u_{1i}}{b_2})(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i}) \\
&= \sum_{i=1}^{n} u_{3i}(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i}) = \sum_{i=1}^{n} \lambda_i u_{2i}u_{3i},
\end{aligned}$$

y,

$$\sum_{i=1}^{n} u_{3i}^2 = \frac{1}{b_2}\sum_{i=1}^{n} u_{3i}(\lambda_i u_{2i} - a_2 u_{2i} - b_1 u_{1i}) = \frac{1}{b_2}\sum_{i=1}^{n} \lambda_i u_{2i}u_{3i} = 1.$$

Repitiendo esta construcción $n-1$ veces obtenemos $(a_1, \ldots, a_n; b_1, \ldots, b_{n-1}) \in \mathbb{R}^n \oplus (\mathbb{R}^+)^{n-1}$ y una base ortonormal $\{u_{ki}\}$, $1 \le k, i \le n$ satisfaciendo

$$\begin{aligned}
&a_1 u_{1i} + b_1 u_{2i} = \lambda_i u_{1i}, && i = 1, \ldots, n, \\
&b_{k-1}u_{k-1i} + a_k u_{ki} + b_k u_{k+1i} = \lambda_i u_{ki} && i = 1, \ldots, n,
\end{aligned} \tag{17}$$

y

$$b_i = \sum_{j=1}^{n} \lambda_i u_{jj} u_{j+1j}, \qquad i = 1 \ldots, n-1$$

$$a_i = \sum_{j=1}^{n} \lambda_i u_{jj}^2 \qquad\qquad i = 1, \ldots, n.$$

Solo queda demostrar que

$$b_{n-1} u_{n-1i} + a_n u_{ni} = \lambda_i u_{ni} \quad i = 1, \ldots, n.$$

Pero, para $k = 1, \ldots, n-2$, utilizando las igualdades (17), se tiene

$$\begin{aligned}
&\sum_{i=1}^{n} u_{ki}(b_{n-1} u_{n-1i} + a_n u_{ni} - \lambda_i u_{ni}) \\
&= b_{n-1} \sum_{i=1}^{n} (u_{ki} u_{n-1i}) + a_n \sum_{i=1}^{n} (u_{ki} u_{n-1i}) - \sum_{i=1}^{n} \lambda_i u_{ki} u_{ni} \\
&= -\sum_{i=1}^{n} \lambda_i u_{ki} u_{ni} \\
&= -\sum_{i=1}^{n} (b_{k-1} u_{k-1i} + a_k u_{ki} + b_k u_{k+1i}) u_{ni} \\
&= 0.
\end{aligned}$$

También,

$$\begin{aligned}
&\sum_{i=1}^{n} u_{n-1i}(b_{n-1} u_{n-1i} + a_n u_{ni} - \lambda_i u_{ni}) \\
&= b_{n-1} \sum_{i=1}^{n} u_{n-1i}^2 + a_n \sum_{i=1}^{n} u_{ni} u_{n-1i} - \sum_{i=1}^{n} \lambda_i u_{n-1i} u_{ni} \\
&= b_{n-1} - \sum_{i=1}^{n} \lambda_i u_{n-1i} u_{ni} \\
&= 0
\end{aligned}$$

y

$$\begin{aligned}
&\sum_{i=1}^{n} u_{ni}(b_{n-1}u_{n-1i} + a_n u_{ni} - \lambda_i u_{ni}) \\
&= b_{n-1}\sum_{i=1}^{n} u_{n-1i}u_{ni} + a_n \sum_{i=1}^{n} u_{ni}^2 - \sum_{i=1}^{n} \lambda_i u_{ni}^2 \\
&= a_n - \sum_{i=1}^{n} \lambda_i u_{ni}^2 \\
&= 0.
\end{aligned}$$

El teorema queda demostrado. □

Aplicando este teorema deducimos que el conjunto de matrices tridiagonales simétricas con $b_i > 0$ con un espectro $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ dado es difeomorfo al subconjunto de la esfera unidad $\sum_{i=1}^{n} x_i^2 = 1$ con $x_i > 0$ (y por lo tanto difeomorfo a $\mathbb{R}^{n-1}$).

Ilustramos las implicaciones de este teorema para $n = 3$ con la ayuda de la Figura 1.

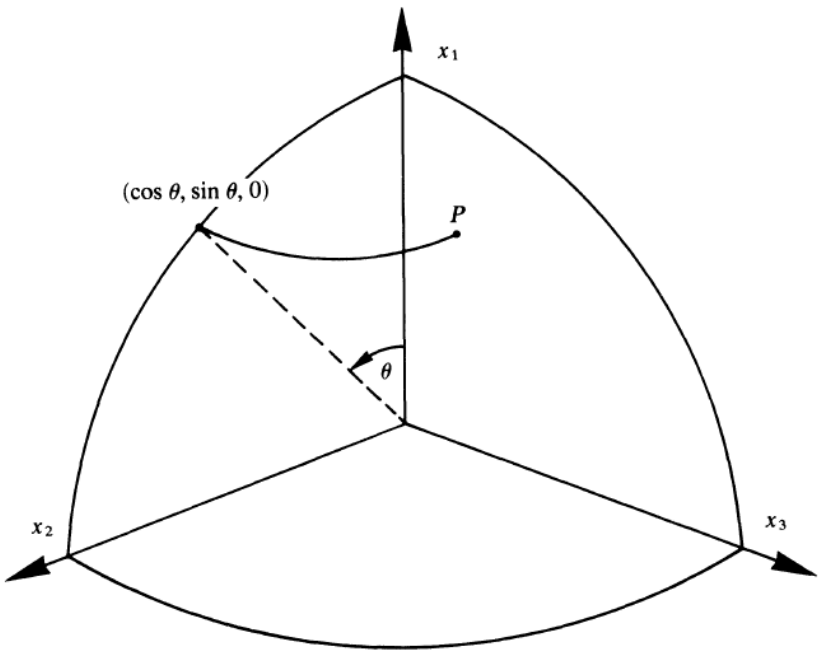



Figura 1: Subconjunto de la esfera unidad en el primer octante

Suponemos que $P = (u_{11}, u_{12}, u_{13})$ se acerca a un borde (pero no a un vértice). Por ejemplo $P \to (\cos\theta, \operatorname{sen}\theta, 0), 0 < \theta < \frac{\pi}{2}$. La matriz tridiagonal correspondiente converge a una matriz con la forma

$$\begin{bmatrix} a_1 & b_1 & 0 \\ b_1 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}$$

con $b_1 > 0$ y $a_3 = \lambda_3$.

Para ver esto, consideramos $P(\varepsilon) = (\alpha_1(\varepsilon), \alpha_2(\varepsilon), \varepsilon)$ donde $\varepsilon > 0$, $\alpha_1(\varepsilon)^2 + \alpha_2(\varepsilon)^2 + \varepsilon^2 = 1$ y $\alpha_1(\varepsilon) \to \cos\theta, \alpha_2(\varepsilon) \to \operatorname{sen}\theta$ cuando $\varepsilon \to 0$.

De (7), $u_{2j} = ((\lambda_j - a_1)u_{1j})/b_1$, vemos que $(u_{21}, u_{22}, u_{23}) = (\beta_1(\varepsilon), \beta_2(\varepsilon), \beta_3(\varepsilon))$, donde $\beta_3(\varepsilon) = O(\varepsilon)$, de manera que $(u_{21}, u_{22}, u_{23}) \to (\operatorname{sen}\theta, -\cos\theta, 0)$ cuando $\varepsilon \to 0$.

Finalmente, $(u_{31}, u_{32}, u_{33})$ es el producto vectorial de los vectores $(u_{11}, u_{12}, u_{13})$ y $(u_{21}, u_{22}, u_{23})$, excepto por el signo y $(u_{31}, u_{32}, u_{33}) = (\gamma_1(\varepsilon), \gamma_2(\varepsilon), \gamma_3(\varepsilon))$, donde $\gamma_1(\varepsilon)$ y $\gamma_2(\varepsilon)$ son $O(\varepsilon)$ cuando $\varepsilon \to 0$.

Ahora, por la biyección establecida en el Teorema 2.1.2 (en particular la ecuación (7)), la matriz tridiagonal correspondiente a $P(\varepsilon)$ es

$$L(\varepsilon) = \begin{bmatrix} \alpha_1(\varepsilon) & \alpha_2(\varepsilon) & \varepsilon \\ \beta_1(\varepsilon) & \beta_2(\varepsilon) & \beta_3(\varepsilon) \\ \gamma_1(\varepsilon) & \gamma_2(\varepsilon) & \gamma_3(\varepsilon) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \alpha_1(\varepsilon) & \beta_1(\varepsilon) & \gamma_1(\varepsilon) \\ \alpha_2(\varepsilon) & \beta_2(\varepsilon) & \gamma_2(\varepsilon) \\ \varepsilon & \beta_3(\varepsilon) & \gamma_3(\varepsilon) \end{bmatrix},$$

que converge cuando $\varepsilon \to 0$ a

$$\begin{aligned} L(0) &= \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & -\cos\theta & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & -\cos\theta & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \\ &= \begin{bmatrix} a_1 & b_1 & 0 \\ b_1 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}. \end{aligned}$$

Si $P \to (0, \cos\phi, \operatorname{sen}\phi)$ o $(0, \cos\psi, \operatorname{sen}\psi), 0 < \phi, \psi < \frac{\pi}{2}$, un cálculo similar muestra que las matrices correspondientes también convergen a la misma forma

$$\begin{bmatrix} a_1 & b_1 & 0 \\ b_1 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix},$$

pero con $a_3 = \lambda_1$ y $\lambda_2$, respectivamente.

Por otra parte, si $P \to (1,0,0)$, las correspondientes matrices no convergen necesariamente. Sin embargo, el elemento $b_1$, siempre converge a 0, pues por (6) y (8),

$$a_1 = \sum_{j=1}^{3} \lambda_j u_{1j}^2, \qquad b_1^2 = \sum_{j=1}^{3} (\lambda_j - a_1)^2 u_{1j}^2,$$

y si $(u_{11}, u_{12}, u_{13}) \to (1,0,0)$, entonces $a_1 \to \lambda_1$ y por tanto $b_1 \to 0$. Así, en las variables $\mathbf{a}, \mathbf{b}$ todos los puntos límite cuando $P \to (1,0,0)$ son matrices de la forma

$$\begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & b_2 \\ 0 & b_2 & a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & \operatorname{sen}\xi \\ 0 & \operatorname{sen}\xi & -\cos\xi \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & \operatorname{sen}\xi \\ 0 & \operatorname{sen}\xi & -\cos\xi \end{bmatrix}$$

con $0 \le \xi \le \frac{\pi}{2}$. Si $P = (u_{11}, u_{12}, u_{13}) \to (1,0,0)$ y $\frac{u_{12}}{u_{13}} \to \gamma$, $0 \le \gamma \le \infty$, entonces, por (7) deducimos que

$$\frac{u_{22}}{u_{23}} = (\frac{\lambda_2 - a_1}{\lambda_3 - a_1})\frac{u_{12}}{u_{13}},$$

y por ortogonalidad,

$$u_{21} = -\frac{1}{u_{11}}(u_{22}u_{12} + u_{23}u_{13}),$$

y, $(u_{21}, u_{22}, u_{23})$ converge a $(0, \cos\xi, \operatorname{sen}\xi)$, donde

$$\cot\xi = \gamma(\frac{\lambda_2 - \lambda_1}{\lambda_3 - \lambda_1}), \quad 0 \le \xi \le \frac{\pi}{2}.$$

Así, cuando $P \to (1,0,0)$, las correspondientes matrices convergen si el cociente $\frac{u_{12}}{u_{13}}$ tiene límite. Similarmente los vértices $(0,1,0)$ y $(0,0,1)$ corresponden a intervalos de matrices de la misma forma,

$$\begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & b_2 \\ 0 & b_2 & a_3 \end{bmatrix}$$

pero ahora a $a_1 = \lambda_2$ y $\lambda_3$ respectivamente.

Teniendo en cuenta lo dicho hasta ahora, podemos ver que el espacio de las matrices tridiagonales simétricas $3 \times 3$ con $b_i > 0$ y espectro fijo $\lambda_1 > \lambda_2 > \lambda_3$ es homeomorfo a un hexágono cerrado: el interior corresponde a matrices con $b_i > 0$ mientras que la frontera corresponde a las matrices con algunos $b_i = 0$, ordenados como en la Figura 2.

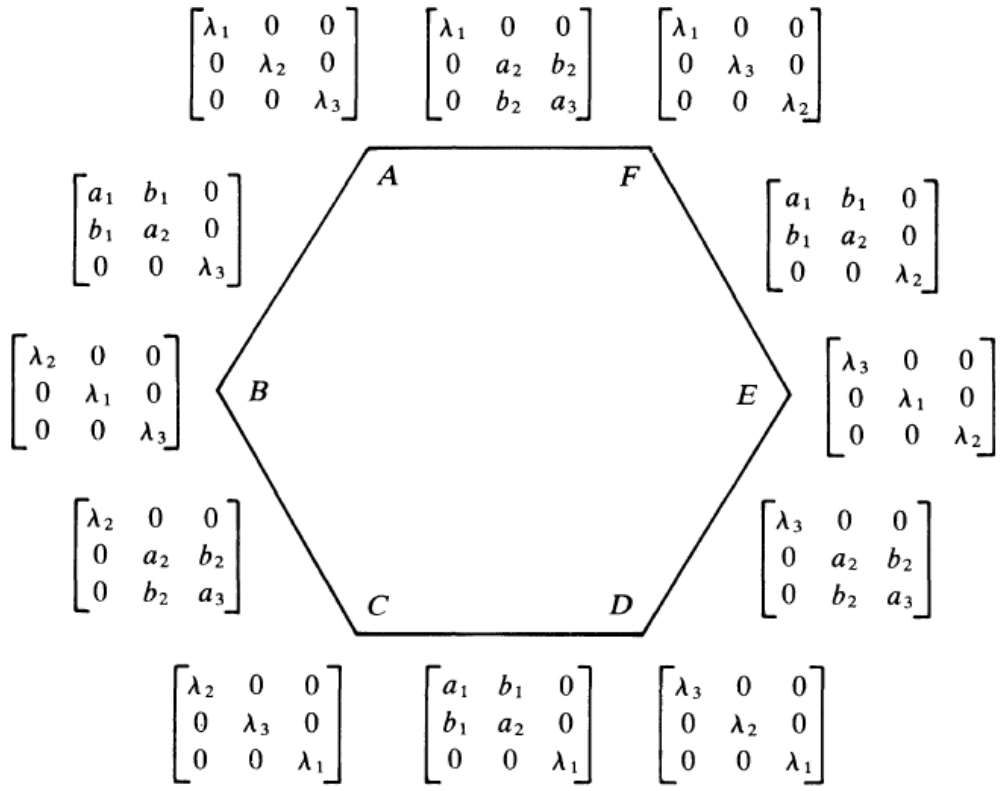



Figura 2: Espacio de las matrices tridiagonales

$AB, CD$ y $EF$ corresponden en $u_{ij}$ a los bordes $(\cos\theta, \operatorname{sen}\theta, 0)$, $(0, \cos\phi, \operatorname{sen}\phi)$ y $(\cos\psi, 0, \operatorname{sen}\psi)$, $0 \leq \theta, \phi, \psi \leq \frac{\pi}{2}$, respectivamente. $FA, BC$ y $DE$ corresponden a los vértices $(1, 0, 0)$, $(0, 1, 0)$ y $(0, 0, 1)$, respectivamente.

La siguiente proposición prueba que (1) se puede resolver de forma explícita en términos de las variables "inversas" $(\Lambda, \mathbf{u})$.

Recordemos que $\mathbf{u} = U^T \mathbf{e}_1 = [u_{11}, u_{12}, \ldots, u_{1n}]^T$ el vector de las primeras componentes de los autovectores de $L$.

**Proposición 2.1.1.** *Sea $L(t)$ la matriz tridiagonal simétrica cuyos coeficientes $a_k(t), b_k(t)$ son soluciones de (1). Entonces*

$$\mathbf{u}(t) = \frac{e^{\Lambda t}\mathbf{u}(0)}{\|e^{\Lambda t}\mathbf{u}(0)\|}, \quad 0 \leq t < \infty. \tag{18}$$

*Demostración.* De (5) tenemos que $u_{ij}(t) = \sum_{k=1}^{n} U_{ik}(t)u_{kj}(0)$ y de (4) tenemos que

$$\frac{d}{dt}u_{ij}(t) = \sum_{k=1}^{n} B_{ik}(t)u_{kj}(t).$$

En particular,

$$\frac{d}{dt}u_{1j}(t) = b_1(t)u_{2j}(t) = (\lambda_j - a_1(t))u_{1j}(t),$$

es decir, $u_{1j}(t)$ debe cumplir

$$\frac{d}{dt}u_{1j}(t) = (\lambda_j - \sum_{k=1}^{n} \lambda_k u_{1k}^2(t))u_{1j}(t). \tag{19}$$

Si $\mathbf{v}(t) = [v_1(t), v_2(t), \ldots, v_n(t)]$, y $\mathbf{u}(t) = \frac{\mathbf{v}(t)}{\|\mathbf{v}(t)\|}$, es decir, $u_{1j}(t) = \frac{v_j(t)}{\|\mathbf{v}(t)\|}$ tenemos que

$$\begin{aligned}
\frac{d}{dt}u_{1j}(t) = \frac{d}{dt}(\frac{v_j(t)}{\|\mathbf{v}(t)\|}) &= \frac{\|\mathbf{v}(t)\| \cdot \frac{d}{dt}v_j(t) - v_j(t) \cdot \frac{d}{dt}\|\mathbf{v}(t)\|}{\|\mathbf{v}(t)\|^2} \\
&= \frac{\frac{d}{dt}v_j(t)}{\|\mathbf{v}(t)\|} - \frac{\frac{v_j(t)}{\|\mathbf{v}(t)\|} \cdot \sum_{k=1}^{n} v_k(t) \cdot \frac{d}{dt}v_k(t)}{\|\mathbf{v}(t)\|^2} \\
&= \frac{\frac{d}{dt}v_j(t)}{\|\mathbf{v}(t)\|} - \frac{v_j(t)}{\|\mathbf{v}(t)\|} \cdot \sum_{k=1}^{n} \frac{v_k(t)}{\|\mathbf{v}(t)\|} \cdot \frac{\frac{d}{dt}v_k(t)}{\|\mathbf{v}(t)\|}.
\end{aligned} \tag{20}$$

Tomando $v(t) = e^{\Lambda t}\mathbf{u}(0)$, es decir, $v_j(t) = e^{\lambda_j t}u_j(0)$ $1 \leq j \leq n$, se sigue que

$$\frac{d}{dt}v_j(t) = \lambda_j e^{\lambda_j t}u_j(0) = \lambda_j v_j(t),$$

y sustituyendo en (20) tenemos para la solución de (18)

$$\begin{aligned}\frac{d}{dt}u_{1j}(t) &= \frac{\lambda_j v_j(t)}{\|\mathbf{v}(t)\|} - u_{1j}(t)\sum_{k=1}^{n} u_{1k}(t)\frac{\lambda_k v_k(t)}{\|\mathbf{v}(t)\|} \\ &= \lambda_j u_{1j}(t) - u_{1j}(t)\sum_{k=1}^{n} u_{1k}(t)\lambda_k u_{1k}(t) \\ &= (\lambda_j - \sum_{k=1}^{n}\lambda_k u_{1k}^2(t))u_{1j}(t).\end{aligned}$$

Vemos que $\mathbf{u}(t)$ definido como en (18) es solución de la ecuación (19), y por la unicidad de la solución de la ecuación diferencial el lema queda demostrado.

□

Demostramos a continuación que la matriz $L(t)$ solución de (1) converge exponencialmente a la matriz de autovalores de $L(0)$, lo cual implica la propiedad 2 enunciada al principio del capítulo.

**Teorema 2.1.3.** *Seana$_k(t)$, $1 \le k \le n$, $b_k(t)$, $1 \le k \le n-1$ solución de* (1) *y $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ los autovalores de $L(0)$. Se verifica para $t \ge 0$,*

$$\begin{aligned}&|a_k(t) - \lambda_k| \le ce^{-2\mu t}, \\ &b_k^2(t) \le ce^{-2\mu t},\end{aligned}$$

*donde la constante $c$ sólo depende de $L(0)$ y $\mu = \min_{k=1,\ldots,n-1}(\lambda_k - \lambda_{k+1})$.*

*Demostración.* La prueba usa inducción y la biyección establecida en el Teorema 2.1.2 para mostrar que la matriz $(u_{ij}(t))$ de autovectores converge exponencialmente a la matriz identidad.

Por (18),

$$\begin{aligned}u_{1j}(t) = \frac{e^{\lambda_j t}u_{1j}(0)}{\|e^{\Lambda t}\mathbf{u}(0)\|} &= \frac{e^{\lambda_j t}u_{1j}(0)}{\sqrt{e^{2\lambda_1 t}u_{11}^2(0) + \cdots + e^{2\lambda_n t}u_{1n}^2(0)}} \\ &= \frac{e^{\lambda_j t}u_{1j}(0)}{e^{\lambda_1 t}\sqrt{u_{11}^2(0) + e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0) + \cdots + e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}} \qquad (21) \\ &= \frac{e^{-(\lambda_1-\lambda_j)t}u_{1j}(0)}{\sqrt{u_{11}^2(0) + e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0) + \cdots + e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}}.\end{aligned}$$

Tomando el límite cuando $t$ tiende a $\infty$ se tiene

$$\begin{aligned}\lim_{t\to\infty} u_{1j}(t) &= \lim_{t\to\infty} e^{-(\lambda_1-\lambda_j)t}\frac{u_{1j}(0)}{\sqrt{u_{11}^2(0)+e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0)+\cdots+e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}}\\ &= \lim_{t\to\infty} e^{-(\lambda_1-\lambda_j)t}\frac{u_{1j}(0)}{u_{11}(0)} = \begin{cases} 1 \text{ si } j=1\\ 0 \text{ si } j\neq 1\end{cases}.\end{aligned}$$

Es decir $(u_{11}(t), u_{12}(t), \ldots, u_{1n}(t))^T$, el vector de las primeras componentes de los autovectores de $L(t)$, converge exponencialmente a $(1, 0, \ldots, 0)^T$.

Vamos a avanzar por partes; la constante $c$, las constantes $d$ y $d'$, y todas las constantes $c_i$ que van a ir apareciendo en la demostración dependen solo de $L(0)$ (en algunos casos a través de su dimensión).

Primero, veamos que

$$\begin{aligned}&1-u_{11}^2(t)\leq ce^{-2\mu t},\\ &u_{1j}^2(t)\leq ce^{-2\mu t},\quad 2\leq j\leq n.\end{aligned}\tag{a}$$

De (21) y teniendo en cuenta que para $2\leq j\leq n$ se cumple que $(\lambda_1-\lambda_j)\geq (j-1)\mu\geq\mu$, es decir, que $e^{-2(\lambda_1-\lambda_j)t}\leq e^{-2(j-1)\mu t}\leq e^{-2\mu t}$, se deduce que,

$$\begin{aligned}1-u_{11}^2(t) &= 1-\frac{e^{-2(\lambda_1-\lambda_1)t}u_{11}^2(0)}{u_{11}^2(0)+e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0)+\cdots+e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}\\ &= \frac{e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0)+\cdots+e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}{u_{11}^2(0)+e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0)+\cdots+e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}\\ &\leq \frac{e^{-2(\lambda_1-\lambda_2)t}u_{12}^2(0)+\cdots+e^{-2(\lambda_1-\lambda_n)t}u_{1n}^2(0)}{u_{11}^2(0)}\\ &\leq \frac{e^{-2\mu t}(u_{12}^2(0)+\cdots+u_{1n}^2(0))}{u_{11}^2(0)} = c_1e^{-2\mu t},\end{aligned}$$

con

$$c_1=\frac{u_{12}^2(0)+\cdots+u_{1n}^2(0)}{u_{11}^2(0)}.$$

Por otro lado,

$$\begin{aligned}u_{1j}^2(t) &= \frac{e^{-2(\lambda_1-\lambda_j)t}u_{1j}^2(0)}{u_{11}^2(0)+e^{2(\lambda_2-\lambda_1)t}u_{12}^2(0)+\cdots+e^{2(\lambda_n-\lambda_1)t}u_{1n}^2(0)}\\ &\leq e^{-2(\lambda_1-\lambda_j)t}\frac{u_{1j}^2(0)}{u_{11}^2(0)}\leq e^{-2\mu t}\frac{u_{1j}^2(0)}{u_{11}^2(0)} = c_2e^{-2\mu t},\end{aligned}$$

para $2 \leq j \leq n$, con

$$c_2 = \max_{2 \leq j \leq n} \left( \frac{u_{1j}^2(0)}{u_{11}^2(0)} \right).$$

Alternativamente se podría haber utilizado que $\mathbf{u}(t)$ es unitario,

$$u_{1j}^2(t) \leq \sum_{j=2}^{n} u_{1j}^2 = 1 - u_{11}^2(t) \leq c_1 e^{-2\mu t}.$$

A continuación vemos que,

$$|a_1(t) - \lambda_1| \leq c e^{-2\mu t}. \tag{b}$$

Utilizando (10) y que $\mathbf{u}(t)$ es un vector unitario se tiene

$$\begin{aligned} |a_1(t) - \lambda_1| &= \left| \sum_{j=1}^{n} \lambda_j u_{1j}^2(t) - \lambda_1 \|\mathbf{u}(t)\|^2 \right| = \left| \sum_{j=2}^{n} (\lambda_j - \lambda_1) u_{1j}^2(t) \right| \\ &= \sum_{j=2}^{n} (\lambda_1 - \lambda_j) u_{1j}^2(t) \leq (c_2 \sum_{j=2}^{n} (\lambda_1 - \lambda_j)) e^{-2\mu t} = c_3 e^{-2\mu t}, \end{aligned}$$

con

$$c_3 = c_2 \sum_{j=2}^{n} (\lambda_1 - \lambda_j).$$

Por otro lado, se puede ver,

$$\begin{aligned} |a_1(t) - \lambda_1| &= \sum_{j=2}^{n} (\lambda_1 - \lambda_j) u_{1j}^2(t) \\ &\geq (\lambda_1 - \lambda_2) u_{12}^2(t) \geq d(\lambda_1 - \lambda_2) e^{-2(\lambda_1 - \lambda_2)t}. \end{aligned} \tag{22}$$

Conviene demostrar aquí otra desigualdad que se usará varias veces en el resto de la demostración. Para $2 \leq j \leq n$, se tiene

$$\begin{aligned} |\lambda_j - a_1(t)| &= |\lambda_j - \lambda_1 + \lambda_1 - a_1(t)| \leq |\lambda_j - \lambda_1| + |\lambda_1 - a_1(t)| \\ &\leq (\lambda_1 - \lambda_n) + c_3 e^{-2\mu t} \leq (\lambda_1 - \lambda_n) + c_3 = d'. \end{aligned} \tag{23}$$

Probamos ahora que

$$b_1^2(t) \leq c e^{-2\mu t}. \tag{c}$$

De (a), (b) y (8), se deduce que

$$\begin{aligned} b_1^2(t) &= \sum_{j=1}^{n}(\lambda_j - a_1(t))^2 u_{1j}^2(t) = \\ &= |a_1(t) - \lambda_1|^2 u_{11}(t) + \sum_{j=2}^{n}(\lambda_j - a_1(t))^2 u_{1j}^2(t) \\ &\leq c_3^2 e^{-4\mu t} u_{11}^2(t) + c_2 e^{-2\mu t} \sum_{j=2}^{n}(\lambda_j - a_1(t))^2 \\ &\leq c_3^2 e^{-2\mu t} + c_2 e^{-2\mu t} \sum_{j=2}^{n}(\lambda_j - a_1(t))^2 \\ &\leq c_3^2 e^{-2\mu t} + c_2 e^{-2\mu t}(n-1)(d')^2 \\ &= c_4 e^{-2\mu t}, \end{aligned}$$

con

$$c_4 = c_3^2 + c_2(n-1)(d')^2.$$

A continuación vemos que

$$u_{21}^2(t) \leq c e^{-2\mu t}. \tag{d}$$

Por la ortogonalidad de las dos primeras filas de $U(t)$,

$$u_{21}(t) = \frac{-(\sum_{j=2}^{n} u_{1j}(t) u_{2j}(t))}{u_{11}(t)}.$$

La acotación (d) se sigue de (a), utilizando la desigualdad de Cauchy-Schwarz

y (18) y recordando que $\sum_{j=2}^{n} u_{2j}^2(t) \leq 1$,

$$\begin{aligned} u_{21}^2(t) &= \frac{(\sum_{j=2}^{n} u_{1j}(t)u_{2j}(t))^2}{u_{11}^2(t)} \leq \frac{(\sum_{j=2}^{n} u_{1j}^2(t))(\sum_{j=2}^{n} u_{2j}^2(t))}{u_{11}^2(t)} \\ &\leq \frac{1}{u_{11}^2(t)} \sum_{j=2}^{n} u_{1j}^2(t) = \frac{\|e^{\Lambda t}\mathbf{u}(0)\|^2}{e^{2\lambda_1 t}u_{11}^2(0)} \sum_{j=2}^{n} u_{1j}^2(t) \\ &= \frac{e^{2\lambda_1 t}u_{11}^2(0) + \cdots + e^{2\lambda_n t}u_{1n}^2(0)}{e^{2\lambda_1 t}u_{11}^2(0)} \sum_{j=2}^{n} u_{1j}^2(t) \\ &\leq \frac{e^{2\lambda_1 t}(u_{11}^2(0) + \cdots + u_{1n}^2(0))}{e^{2\lambda_1 t}u_{11}^2(0)} \sum_{j=2}^{n} u_{1j}^2(t) \\ &= \frac{1}{u_{11}^2(0)} \sum_{j=2}^{n} u_{1j}^2(t) \leq \frac{c_2(n-1)}{u_{11}^2(0)} e^{-2\mu t} = c_5 e^{-2\mu t}, \end{aligned}$$

con

$$c_5 = \frac{c_2(n-1)}{u_{11}^2(0)}$$

A parte, se puede probar que

$$\begin{aligned} u_{21}^2(t) &= \frac{(\sum_{j=2}^{n} u_{1j}(t)u_{2j}(t))^2}{u_{11}^2(t)} \leq \frac{(\sum_{j=2}^{n} u_{1j}^2(t))(\sum_{j=2}^{n} u_{2j}^2(t))}{u_{11}^2(t)} \\ &\leq \frac{1}{u_{11}^2(t)} = \frac{e^{2\lambda_1 t}u_{11}^2(0) + \cdots + e^{2\lambda_n t}u_{1n}^2(0)}{e^{2\lambda_1 t}u_{11}^2(0)} \leq \frac{u_{12}^2(0)}{u_{11}^2(0)} e^{-2(\lambda_1 - \lambda_2)t}, \end{aligned} \tag{24}$$

desigualdad que utilizaremos más adelante.

Ahora vemos que,

$$1 - u_{22}^2(t) \leq ce^{-2\mu t}, \quad u_{2j}^2(t) \leq ce^{-2\mu t}, \quad j \geq 3. \tag{e}$$

De (11),

$$\frac{u_{2j}(t)}{u_{2k}(t)} = \frac{(\lambda_j - a_1(t))u_{1j}(t)}{(\lambda_k - a_1(t))u_{1k}(t)} = \frac{\lambda_j - a_1(t)}{\lambda_k - a_1(t)} \frac{u_{1j}(0)}{u_{1k}(0)} e^{(\lambda_j - \lambda_k)t},$$

y, en particular, tomando $k = 1$ y elevando al cuadrado, se tiene,

$$u_{2j}^2(t) = \frac{(\lambda_j - a_1(t))^2}{(\lambda_1 - a_1(t))^2} \frac{u_{1j}^2(0)}{u_{11}^2(0)} u_{21}^2(t) e^{-2(\lambda_1 - \lambda_j)t}. \tag{25}$$

La segunda desigualdad de (e) se sigue de (25), (22), (24) y (d),

$$\begin{aligned}
u_{2j}^2(t) &= \frac{(\lambda_j - a_1(t))^2}{(\lambda_1 - a_1(t))^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}u_{21}^2(t)e^{-2(\lambda_1-\lambda_j)t} \\
&\leq \frac{(\lambda_j - a_1(t))^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}u_{21}^2(t)e^{4(\lambda_1-\lambda_2)t}e^{-2(\lambda_1-\lambda_j)t} \\
&\leq \frac{(\lambda_j - \lambda_1)^2 + (c_3)^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}u_{21}^2(t)e^{4(\lambda_1-\lambda_2)t}e^{-2(\lambda_1-\lambda_j)t} \\
&\leq \frac{(\lambda_j - \lambda_1)^2 + (c_3)^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}c_5\frac{u_{12}^2(0)}{u_{11}^2(0)}e^{-2(\lambda_1-\lambda_2)t}e^{4(\lambda_1-\lambda_2)t}e^{-2(\lambda_1-\lambda_j)t} \\
&= \frac{(\lambda_j - \lambda_1)^2 + (c_3)^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}c_5\frac{u_{12}^2(0)}{u_{11}^2(0)}e^{-2(\lambda_1-\lambda_2)t}e^{4(\lambda_1-\lambda_2)t}e^{-2(\lambda_1-\lambda_2)t}e^{-2(\lambda_2-\lambda_j)t} \\
&= \frac{(\lambda_j - \lambda_1)^2 + (c_3)^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}c_5\frac{u_{12}^2(0)}{u_{11}^2(0)}e^{-2(\lambda_2-\lambda_j)t} \leq c_6e^{-2\mu t},
\end{aligned}$$

con

$$c_6 = \frac{(\lambda_j - \lambda_1)^2 + (c_3)^2}{(d)^2(\lambda_1 - \lambda_2)^2}\frac{u_{1j}^2(0)}{u_{11}^2(0)}\frac{u_{12}^2(0)}{u_{11}^2(0)}c_5.$$

Para la otra desigualdad,

$$\begin{aligned}
1 - u_{22}^2(t) &= \sum_{j=1,j\neq 2}^{n} u_{2j}^2(t) = u_{21}^2(t) + \sum_{j=3}^{n} u_{2j}^2(t) \\
&\leq c_5e^{-2\mu t} + (n-2)c_6e^{-2\mu t} = c_7e^{-2\mu t},
\end{aligned}$$

con

$$c_7 = c_5 + (n-2)c_6.$$

A continuación probamos,

$$|a_2(t) - \lambda_2| \leq ce^{-2\mu t}. \tag{f}$$

Para ello utilizamos (16) y procedemos de manera analoga a como lo hicimos para probar (b), utilizando (d) y (e).

$$\begin{aligned}
|a_2(t) - \lambda_2| &= |\sum_{j=1,j\neq 2}^{n}(\lambda_j - \lambda_2)u_{2j}^2(t)| = |(\lambda_1 - \lambda_2)u_{21}^2(t) + \sum_{j=3}^{n}(\lambda_j - \lambda_2)u_{2j}^2(t)| \\
&\leq (\lambda_1 - \lambda_2)c_5e^{-2\mu t} + \sum_{j=3}^{n}(\lambda_2 - \lambda_j)c_6e^{-2\mu t} = c_8e^{-2\mu t},
\end{aligned}$$

con

$$c_8 = (\lambda_1 - \lambda_2)c_5 + \sum_{j=3}^{n}(\lambda_2 - \lambda_j)c_6.$$

Antes de dar el siguiente paso probamos una desigualdad que nos facilitará la siguiente demostración

$$\begin{aligned} |\lambda_j - a_2(t)| &= |\lambda_j - \lambda_2 + \lambda_2 - a_2(t)| \leq |\lambda_j - \lambda_2| + |\lambda_2 - a_2(t)| \\ &\leq (\lambda_1 - \lambda_n) + c_8 e = d_2. \end{aligned}$$

Para probar que

$$b_2^2(t) \leq ce^{-2\mu t}, \tag{g}$$

usamos que $b_2^2(t) = \sum_{j=1}^{n}[(\lambda_j - a_2(t))u_{2j}(t) - b_1(t)u_{1j}(t)]^2$, y tenemos en cuenta (11), (a), (c), (e) y (f) y la desigualdad anterior,

$$
\begin{aligned}
b_2^2(t) &= \sum_{j=1}^{n}((\lambda_j - a_2(t))u_{2j}(t) - b_1(t)u_{1j}(t))^2 \\
&\leq \sum_{j=1}^{n}(|\lambda_j - a_2(t)||u_{2j}(t)| + |b_1(t)||u_{1j}(t)|)^2 \\
&\leq (|\lambda_1 - a_2(t)||u_{21}(t)| + |b_1(t)||u_{11}(t)|)^2 + (|\lambda_2 - a_2(t)||u_{22}(t)| + |b_1(t)|)^2 \\
&+ \sum_{j>2}(|\lambda_j - a_2(t)||u_{2j}(t)| + |b_1(t)||u_{1j}(t)|)^2 \\
&= |\lambda_1 - a_2(t)|^2|u_{21}(t)|^2 + |b_1(t)|^2 + 2|\lambda_1 - a_2(t)||b_1(t)||u_{21}(t)| \\
&+ |\lambda_2 - a_2(t)|^2|u_{22}(t)|^2 + |b_1(t)|^2 + 2|\lambda_2 - a_2(t)||b_1(t)||u_{22}(t)| \\
&+ \sum_{j>2}|\lambda_j - a_2(t)|^2|u_{2j}(t)|^2 + |b_1(t)|^2|u_{1j}(t)|^2 \\
&+ 2|\lambda_j - a_2(t)||b_1(t)||u_{1j}(t)||u_{2j}(t)| \\
&= |\lambda_1 - a_2(t)|^2|u_{21}(t)|^2 + |b_1(t)|^2 + 2|\lambda_1 - a_2(t)||b_1(t)|\frac{|\lambda_1 - a_1(t)||u_{11}(t)|}{|b_1(t)|} \\
&+ |\lambda_2 - a_2(t)|^2|u_{22}(t)|^2 + |b_1(t)|^2 + 2|\lambda_2 - a_2(t)||b_1(t)|\frac{|\lambda_2 - a_1(t)||u_{12}(t)|}{|b_1(t)|} \\
&+ \sum_{j>2}|\lambda_j - a_2(t)|^2|u_{2j}(t)|^2 + |b_1(t)|^2|u_{1j}(t)|^2 \\
&+ 2|\lambda_j - a_2(t)||b_1(t)|\frac{|\lambda_2 - a_1(t)||u_{1j}(t)|^2}{|b_1(t)|} \\
&\leq d_2^2c_5e^{-2\mu t} + c_4e^{-2\mu t} + 2d_2c_3e^{-2\mu t} + c_8^2e^{-2\mu t} + c_4e^{-2\mu t} + 2dc_8e^{-2\mu t} + \\
&\sum_{j>2} d_2^2c_6e^{-2\mu t} + c_4e^{-2\mu t} + 2dd_2c_2e^{-2\mu t} \\
&= c_9e^{-2\mu t},
\end{aligned}
$$

con

$$c_9 = d_2^2c_5 + c_4 + 2d_2c_3 + c_8^2 + c_4 + 2dc_8 + (n-2)(d_2^2c_6 + c_4 + 2dd_2c_2).$$

En este punto definimos

$$c = \max_{1\leq j\leq 9} c_j.$$

Por último veamos que

$$\frac{b_1(t)u_{1s}(t)}{u_{2s}(t)} \to 0 \quad \text{cuando } t \to \infty, \quad s \geq 2. \tag{h}$$

Para ello, utilizamos (7), (b) y (c), y obtenemos

$$\frac{b_1(t)u_{1s}(t)}{u_{2s}(t)} = \frac{b_1^2(t)}{(\lambda_s - a_1(t))} = \frac{b_1^2(t)}{(\lambda_s - \lambda_1) + (\lambda_1 - a_1(t))},$$

donde el numerador converge a 0 y el denominador converge a $\lambda_s - \lambda_1$.

Ahora, sea $j \geq 3$ y supongamos por inducción que se cumplen las relaciones siguientes

$$\begin{aligned}
&|a_k(t) - \lambda_k| \leq ce^{-2\mu t}, \quad k \leq j-1, \\
&b_k^2(t) \leq ce^{-2\mu t}, \quad k \leq j-1, \\
&\frac{b_{k-1}(t)u_{k-1s}(t)}{u_{ks}(t)} \to 0 \quad \text{cuando } t \to \infty, \quad k \leq j-1, \quad s \geq k, \\
&|1 - u_{kk}^2(t)| \leq ce^{-2\mu t}, \quad k \leq j-1, \\
&u_{kl}^2(t) \leq ce^{-2\mu t}, \quad k \leq j-1, \quad l \neq k, \\
&\frac{u_{kr}^2(t)}{u_{ks}^2(t)} \sim ce^{2(\lambda_r - \lambda_s)t} \text{cuando } t \to \infty, \quad k \leq j-1, \quad r, s \geq k.
\end{aligned}$$

Probamos que la misma relación se mantiene para $k = j$. Sean $r, s \geq j$. Por (7) tenemos

$$\begin{aligned}
\frac{u_{jr}(t)}{u_{js}(t)} &= \frac{(\lambda_r - a_{j-1})u_{j-1r}(t) - b_{j-2}(t)u_{j-2r}(t)}{(\lambda_s - a_{j-1})u_{j-1s}(t) - b_{j-2}(t)u_{j-2s}(t)} \\
&= \frac{u_{j-1r}(t)}{u_{j-1s}(t)} \frac{(\lambda_r - a_{j-1}) - b_{j-2}(t)\frac{u_{j-2r}(t)}{u_{j-1r}(t)}}{(\lambda_s - a_{j-1}) - b_{j-2}(t)\frac{u_{j-2s}(t)}{u_{j-1s}(t)}} \\
&= \frac{u_{j-1r}(t)}{u_{j-1s}(t)} \frac{(\lambda_r - \lambda_{j-1} + \lambda_{j-1} - a_{j-1}) - b_{j-2}(t)\frac{u_{j-2r}(t)}{u_{j-1r}(t)}}{(\lambda_s - \lambda_{j-1} + \lambda_{j-1} - a_{j-1}) - b_{j-2}(t)\frac{u_{j-2s}(t)}{u_{j-1s}(t)}} \\
&\sim ce^{(\lambda_r - \lambda_s)t}\left(\frac{\lambda_r - \lambda_{j-1} + o(1)}{\lambda_s - \lambda_{j-1} + o(1)}\right),
\end{aligned}$$

aplicando las hipótesis de inducción.

De nuevo por (7), para $s \geq j$ tenemos

$$b_{j-1} = \frac{\lambda_s - a_{j-1} - b_{j-2}(t)u_{j-2s}(t)}{u_{j-1s}(t)} \frac{u_{j-1s}(t)}{u_{js}(t)},$$

de manera que

$$\begin{aligned} \frac{b_{j-1}(t)u_{j-1s}(t)}{u_{js}(t)} &= \frac{b_{j-1}^2(t)}{\lambda_s - a_{j-1}(t) - b_{j-2}(t)\frac{u_{j-2s}(t)}{u_{j-1s}(t)}} \\ &= \frac{b_{j-1}^2(t)}{(\lambda_s - \lambda_{j-1}) + (\lambda_{j-1} - a_{j-1}(t)) - b_{j-2}(t)\frac{u_{j-2s}(t)}{u_{j-1s}(t)}}, \end{aligned}$$

que converge a 0 cuando $t \to \infty$ por las hipótesis de inducción.

La prueba de que $u_{jm}^2 \leq ce^{-2\mu t}, m < j$, se sigue de las hipótesis de inducción y de las relaciones de ortogonalidad $\sum_{i=1}^n u_{ji}u_{ki} = 0, k < j$.

Las desigualdades

$$\begin{aligned} |1 - u_{jj}^2(t)| &\leq ce^{-2\mu t}, \\ u_{jm}^2 &\leq ce^{-2\mu t}, \quad m > j, \end{aligned}$$

se deducen del hecho de que $\frac{u_{jm}^2(t)}{u_{jj}^2(t)} \sim ce^{2(\lambda_m - \lambda_j)t}, m > j$, y de la condición de normalización $\sum_{j=1}^n u_{ji}^2(t) = 1$, como en (e).

Por último las desigualdades en $|a_j(t) - \lambda_j|$ y $b_j^2(t)$ se siguen de

$$|a_j(t) - \lambda_j| = |\sum_{j=1, k\neq j}^{n} (\lambda_k - \lambda_j)u_{jk}^2(t)|,$$

y

$$b_j^2(t) = \sum_{k=1}^{2}[(\lambda_k - a_j(t))u_{jk}(t) - b_{j-1}(t)u_{j-1k}(t)]^2,$$

respectivamente, procediendo como en (b) y en (c).

Esto completa la inducción y la prueba del teorema. □

## 2.2. El flujo de Toda y el algoritmo QR

En esta sección mostraremos que los iterantes que genera el algoritmo $QR$ sin desplazamientos descrito en el Capítulo 1 coinciden con la evaluación en tiempos enteros del flujo de un sistema diferencial matricial en el espacio de matrices tridiagonales simétricas con espectro fijo.

Sea $L$ una matriz tridiagonal simétrica en $\mathbb{R}^{n\times n}$ sobre la que vamos a aplicar el algoritmo QR.

Partiendo de $L_0 = L$, sea $L_k = Q_{k-1}^T L_{k-1} Q_{k-1}$ la matriz que se obtiene tras aplicar la $k$-ésima iteración QR. Como hemos visto en el Capítulo 1 el carácter tridiagonal de la matriz se conserva en cada iteración.

Para $k \geq 0$, sea $U_k$ la matriz ortogonal cuyas columnas son los autovectores de $L_k$, es decir,

$$L_k = U_k \Lambda U_k^T, \tag{26}$$

siendo $\Lambda$ la matriz diagonal con los autovalores de $L$.

De las igualdades

$$\begin{aligned} L_k =& Q_{k-1}^T L_{k-1} Q_{k-1}, \\ L_k =& U_k \Lambda U_k^T, \end{aligned}$$

podemos deducir que $L_k = Q_{k-1}^T U_{k-1} \Lambda U_{k-1}^T Q_{k-1}$ y por tanto $U_k^T = U_{k-1}^T Q_{k-1}$.

**Proposición 2.2.1.** *Sea* $\mathbf{u}_k = U_k^T \mathbf{e}_1$, *donde* $\mathbf{e}_1 = (1, 0, \ldots, 0)^T$, *es decir* $\mathbf{u}_k$ *es la primera fila de la matriz* $U_k$. *Se verifica que*

$$\mathbf{u}_k = \frac{\Lambda^k \mathbf{u}_0}{\|\Lambda^k \mathbf{u}_0\|}.$$

*Demostración.* La prueba se llevará a cabo por inducción sobre $k$.

Para $k = 1$, utilizando que por ser $L_0 = Q_0 R_0$ la factorización QR de $L_0$, la primera columna de $Q_0$ es la primera columna de $L_0$ normalizada, y, de la igualdad (26) con $k = 0$ se tiene

$$\mathbf{u}_1 = U_1^T \mathbf{e}_1 = U_0^T Q_0 \mathbf{e}_1 U_0^T \left(\frac{L_0 \mathbf{e}_1}{\|L_0 \mathbf{e}_1\|}\right) = \frac{U_0^T L_0 \mathbf{e}_1}{\|U_0^T L_0 \mathbf{e}_1\|} = \frac{\Lambda U_0^T \mathbf{e}_1}{\|\Lambda U_0^T \mathbf{e}_1\|} = \frac{\Lambda \mathbf{u}_0}{\|\Lambda \mathbf{u}_0\|}.$$

Suponemos que se cumple la hipótesis para $k - 1$.

Estudiamos el caso general. Como $U_k^T = U_{k-1}^T Q_{k-1}$ y por ser la primera columna de $Q_{k-1}$ la primera columna de $L_{k-1}$ normalizada,

$$\mathbf{u}_k = U_k^T \mathbf{e}_1 = U_{k-1}^T Q_{k-1} \mathbf{e}_1 = U_{k-1}^T \left(\frac{L_{k-1}\mathbf{e}_1}{\|L_{k-1}\mathbf{e}_1\|}\right),$$

aplicando la igualdad (26), y el hecho de que $\mathbf{u}_k$ sea la primera fila de $U_{k-1}$, es decir, la primera columna de $U_{k-1}^T$,

$$U_{k-1}^T \left(\frac{L_{k-1}\mathbf{e}_1}{\|L_{k-1}\mathbf{e}_1\|}\right) = \frac{U_{k-1}^T L_{k-1}\mathbf{e}_1}{\|U_{k-1}^T L_{k-1}\mathbf{e}_1\|} = \frac{\Lambda U_{k-1}^T \mathbf{e}_1}{\|\Lambda U_{k-1}^T \mathbf{e}_1\|} = \frac{\Lambda \mathbf{u}_{k-1}}{\|\Lambda \mathbf{u}_{k-1}\|}.$$

Por último, aplicando la hipótesis de inducción,

$$\frac{\Lambda \mathbf{u}_{k-1}}{\|\Lambda \mathbf{u}_{k-1}\|} = \frac{\Lambda(\Lambda^{k-1}\mathbf{u}_0)}{\|\Lambda(\Lambda^{k-1}\mathbf{u}_0)\|} = \frac{\Lambda^k \mathbf{u}_0}{\|\Lambda^k \mathbf{u}_0\|},$$

es decir,

$$\mathbf{u}_k = \frac{\Lambda^k \mathbf{u}_0}{\|\Lambda^k \mathbf{u}_0\|},$$

y podemos dar por finalizada la demostración. □

Sea ahora $A \in \mathbb{R}^{n\times n}$ una matriz simétrica fija. Consideramos el flujo inducido en las matrices tridiagonales (recordamos el Teorema 2.1.2) por

$$\frac{d\Lambda}{dt} = 0, \qquad \frac{d\mathbf{u}}{dt} = A\mathbf{u} - \langle A\mathbf{u}, \mathbf{u}\rangle \mathbf{u}, \tag{27}$$

donde $\Lambda(t) = \Lambda$ es la matriz diagonal con los autovalores de $L$ y $\mathbf{u} = (u_{11}, \ldots, u_{1n}) > 0$ son las primeras componentes de los autovectores asociados.

Cuando $A = \Lambda$, (27) se transforma en (19), las ecuaciones que describen el flujo de Toda, y por la Proposición 2.1.1 se cumple que,

$$\mathbf{u}(t) = \frac{e^{At}\mathbf{u}(0)}{\|e^{At}\mathbf{u}(0)\|}, \tag{28}$$

es la única solución de (27) que tiene como dato inicial $\mathbf{u}(0)$.

Tomando $A = \log \Lambda$ en (27) y utilizando (28) vemos que

$$\mathbf{u}(t) = \frac{\Lambda^t \mathbf{u}(0)}{\|\Lambda^t \mathbf{u}(0)\|},$$

y en particular, para tiempos enteros $t = k$,

$$\mathbf{u}(k) = \frac{\Lambda^k \mathbf{u}(0)}{\|\Lambda^k \mathbf{u}(0)\|}.$$

Combinando ahora la Proposición 2.2.1 y el Teorema 2.1.2, hemos probado el siguiente teorema.

**Teorema 2.2.1.** *Se verifica que*

$$L_k = L(k),$$

*donde $L_k$ es el $k$-ésimo iterante del algoritmo QR partiendo de $L_0 = L$, y $L(k)$ es la evaluación en $t = k$ del flujo* (27) *con $A = \log \Lambda$ y $L(0) = L$.*

**Corolario 2.2.1.** *Si $L_0$ es una matriz tridiagonal simétrica, el $k$-ésimo iterante del algoritmo QR aplicado a $e^{L_0}$ coincide con $e^{L(k)}$, siendo $L(k)$ la solución en $t = k$ del flujo de Toda* (1).

# 3. Integración numérica del flujo de Toda

El Teorema 2.1.3 establece que la solución exacta del flujo de Toda (1) converge exponencialmente hacia la matriz de autovalores de la matriz inicial $L(0)$ asociada. Por tanto, el problema de calcular los autovalores de una matriz tridiagonal simétrica se puede resolver integrando numéricamente el flujo (1) asociado. Un procedimiento sencillo para integrar (1) es utilizar un método Runge-Kutta explícito de $s$ etapas con tablero de Butcher

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}, \tag{29}$$

y cuyos coeficientes satisfacen

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad 1 \leq i \leq s.$$

En el caso en que $a_{ij} = 0$ para $i \leq j$, el método explícito y las ecuaciones para avanzar un paso de longitud $h$ en la integración numérica de un problema de la forma

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \qquad \mathbf{y}(t_0) = \mathbf{y}, \tag{30}$$

con dicho método son,

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_0, \mathbf{y}_0), \\ \mathbf{k}_2 &= \mathbf{f}(t_0 + c_2 h, \mathbf{y}_0 + h a_{21} \mathbf{k}_1), \\ \mathbf{k}_3 &= \mathbf{f}(t_0 + c_3 h, \mathbf{y}_0 + h(a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2)), \\ &\vdots \\ \mathbf{k}_s &= \mathbf{f}(t_0 + c_s h, \mathbf{y}_0 + h(a_{s1}\mathbf{k}_1 + \cdots + a_{ss-1}\mathbf{k}_{s-1})), \\ \mathbf{y}_1 &= \mathbf{y}_0 + h(b_1\mathbf{k}_1 + \cdots + b_s\mathbf{k}_s), \\ t_1 &= t_0 + h. \end{aligned}$$

Para el flujo de Toda (1), tomamos en (30)

$$\begin{aligned}\mathbf{y}(t) &= (a_1(t), a_2(t), \ldots, a_n(t), b_1(t), b_2(t), \ldots, b_{n-1}(t)),\\ \mathbf{f}(t, \mathbf{y}) &= (2(b_2^2 - b_1^2), \ldots, 2(b_n^2 - b_{n-1}^2), b_1(a_2 - a_1), \ldots, b_{n-1}(a_n - a_{n-1})),\end{aligned} \tag{31}$$

con,

$$\mathbf{y}(0) = (a_1(0), a_2(0), \ldots, a_n(0), b_1(0), b_2(0), \ldots, b_{n-1}(0)).$$

Aunque estos métodos son muy fáciles de implementar, no conservan los autovalores de la matriz, como veremos en el Capítulo 4. Por ello, nos vemos obligados a buscar métodos alternativos que nos permitan integrar el flujo de Toda conservando los autovalores, es decir, métodos *isoespectrales*.

## 3.1. Métodos isoespectrales via el formalismo de Flaschka

Para construir integradores temporales que conserven los autovalores de la matriz de partida recordamos que en la Sección 2.1 el flujo de Toda se ha reescrito en formato matricial (3), y en la demostración del Teorema 2.1.3 se ha visto que resolver (3) es equivalente a resolver

$$\frac{d}{dt}U(t) = B(L(t))U(t), \quad \text{con} \quad U(0) = I, \quad t \geq 0, \tag{32}$$

junto con

$$L(t) = U(t)L(0)U(t)^T, \quad t \geq 0. \tag{33}$$

Se pueden construir entonces métodos *isoespectrales* para el flujo de Toda si se integran numéricamente (32)-(33) utilizando métodos que conserven el carácter ortogonal de la solución de (32).

Mientras (32) se resuelva con un método que conserve el carácter ortogonal de su solución exacta, (33) dará aproximaciones $L_k$ ortogonalmente semejante a $L_0$ y por tanto tendrá los mismos autovalores que la matriz inicial.

Supongamos que tenemos un método Runge-Kutta definido por un tablero de Butcher como en (29) al que asociamos la matriz $M$ definida como

$$M_{ij} = b_i a_{ij} + b_j a_{ji} - b_i b_j \quad i, j = 1, \ldots, s. \tag{34}$$

El siguiente resultado da condiciones suficientes para que un método Runge-Kutta aplicado a (32) conserve el carácter ortogonal de su solución exacta.

**Teorema 3.1.1.** *Dado un sistema diferencial $n \times n$ de la forma*

$$\frac{d}{dt}U = S(t,U)U(t), \quad U(0) = U_0, \quad U_0^T U_0 = I, \tag{35}$$

*donde $S(t,U)$ es antisimétrica, sean $\{U_k\}_{k=0}^{\infty}$ las aproximaciones numericas generadas por un método Runge Kutta. Si los coeficientes del método Runge-Kutta satisfacen $M \equiv O$, entonces $U_k$ es ortogonal para todo $k \geq 0$.*

*Demostración.* Recordemos que las ecuaciones para avanzar un paso de longitud $h$ con un método Runge-Kutta aplicado a (35) son

$$U_{k+1} = U_k + h\sum_{i=1}^{k} b_i K_i,$$

donde, denotando $S(t_k + c_l h, \Phi_j)$ por $S_l$,

$$K_l = S_l \Phi_l, \quad l = 1, \ldots, s,$$
$$\Phi_l = U_k + h\sum_{j=1}^{s} A_{lj} K_j.$$

De aquí se tiene

$$\begin{aligned} U_{k+1}^T U_{k+1} = U_k^T U_k &+ h\sum_{l=1}^{s} b_l \{U_k^T K_l + K_l^T U_k\} \\ &+ h^2 \sum_{l=1}^{s}\sum_{j=1}^{s} b_l b_k K_l^T K_j. \end{aligned} \tag{36}$$

Ahora bien,

$$U_k^T K_l = \Phi_l^T K_l - h\sum_{j=1}^{s} A_{l,j} K_j^T K_l,$$
$$K_l^T U_k = K_l^T \Phi_l - h\sum_{j=1}^{s} A_{l,j} K_l^T K_j.$$

y además,

$$\Phi_l^T K_l + K_l^T \Phi_l = \Phi_l^T S_l \Phi_l + \Phi_l^T S_l^T \Phi_l = \Phi_l^T (S_l + S_l^T) \Phi_l = O,$$

por ser $S_l$ antisimétrica. Así (36) se reduce a

$$U_{k+1}^T U_{k+1} = U_k^T U_k + h^2 \sum_{l=1}^{s} \sum_{j=1}^{s} \{b_l b_j - b_l A_{l,j} - b_j A_{j,l}\} K_j^T K_l.$$

Suponiendo que $U_k$ es ortogonal, es decir, que $U_k^T U_k = I$, y puesto que $M = 0$, concluimos

$$U_{k+1}^T U_{k+1} = I.$$

Es más, como esto se mantiene cuando se considera $U_{k+1}^T U_{k+1}$, el esquema mantiene la ortogonalidad al pasar de $t_k$ a $t_{k+1}$. La prueba se sigue al realizar inducción sobre $k$.

□

Incluimos a continuación dos resultados sin demostración (ver [Hairer]) que, por un lado, muestran que la condición $M \equiv 0$ asegura no sólo la conservación del caracter ortogonal de la solución de (32), sino la conservación de cualquier invariante cuadrático de un sistema diferencial general (30), y, por otro, que los metodos de Gauss nos permiten construir métodos *isoespectrales* cuando se utilizan para integrar (32)-(33):

**Teorema 3.1.2.** *Los métodos de Gauss-Legendre conservan los invariantes cuadráticos.*

**Teorema 3.1.3.** *Dado un método Runge-Kutta, si la la matriz $M$ definida como en* (34)*, cumple $M = 0$, entonces, el método conserva los invariantes cuadráticos.*

## 3.2. La regla implícita del punto medio modificada

En el caso $s = 1$, el método de Gauss-Legendre, no es otro que la regla implícita del punto medio, cuyo tablero de Butcher es

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array},$$

y las ecuaciones para avanzar un paso de longitud $h$ en la integración numérica (30) son lo que podemos escribir de forma extendida como

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + h\mathbf{k}_1, \\ \mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_k + \frac{1}{2}h\mathbf{k}_1), \\ t_{k+1} &= t_k + h, \end{aligned}$$

o, de forma más compacta,

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}\left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right). \tag{37}$$

Cuando aplicamos la regla implícita del punto medio (37) a (32) se tiene

$$U_{k+1} = U_k + hB\left(\frac{L_k + L_{k+1}}{2}\right)\left(\frac{U_k + U_{k+1}}{2}\right),$$

pero $L_{k+1}$ y $U_{k+1}$ son desconocidos. Denotando

$$B_{k+\frac{1}{2}} = B\left(\frac{L_k + L_{k+1}}{2}\right), \tag{38}$$

el esquema se puede reescribir como

$$(I - \frac{1}{2}hB_{k+\frac{1}{2}})U_{k+1} = (I + \frac{1}{2}hB_{k+\frac{1}{2}})U_k,$$

o, alternativamente,

$$U_{k+1} = (I - \frac{1}{2}hB_{k+\frac{1}{2}})^{-1}(I + \frac{1}{2}hB_{k+\frac{1}{2}}). \tag{39}$$

Así tenemos

$$L_{k+1} = U_{k+1}L_kU_{k+1}^T. \tag{40}$$

**Teorema 3.2.1.** *El algoritmo* (38)-(40) *es una modificación* isoespectral *de segundo orden de la regla implícita del punto medio aplicada a* (3).

*Demostración.* Por ser $U$ ortogonal, se deduce que el algoritmo (38)-(40) es *isoespectral*.

Además, podemos usar (39) para eliminar $U_{k+1}$ en (40). Esto lleva a

$$L_{k+1} = (I - \frac{1}{2}hB_{k+\frac{1}{2}})^{-1}(I + \frac{1}{2}hB_{k+\frac{1}{2}})L_k(I - \frac{1}{2}hB_{k+\frac{1}{2}})(I + \frac{1}{2}hB_{k+\frac{1}{2}})^{-1},$$

y, al multiplicar en la ecuación anterior por $(I - \frac{1}{2}hB_{k+\frac{1}{2}})$ a la izquierda y por $(I + \frac{1}{2}hB_{k+\frac{1}{2}})$ a la derecha, obtenemos

$$(I - \frac{1}{2}hB_{k+\frac{1}{2}})L_{k+1}(I + \frac{1}{2}hB_{k+\frac{1}{2}}) = (I + \frac{1}{2}hB_{k+\frac{1}{2}})L_k(I - \frac{1}{2}hB_{k+\frac{1}{2}}).$$

Entonces

$$\begin{aligned} L_{k+1} - \frac{1}{2}hB_{k+\frac{1}{2}}L_{k+1} + \frac{1}{2}hL_{k+1}B_{n+\frac{1}{2}} - \frac{1}{4}h^2B_{k+\frac{1}{2}}L_{k+1}B_{k+\frac{1}{2}} \\ = L_k + \frac{1}{2}hB_{k+\frac{1}{2}}L_k - \frac{1}{2}hL_kB_{k+\frac{1}{2}} - \frac{1}{4}h^2B_{k+\frac{1}{2}}L_kB_{k+\frac{1}{2}} \end{aligned}$$

y, utilizando el operador de conmutación

$$\begin{aligned} L_{k+1} - \frac{1}{2}h[B_{k+\frac{1}{2}}, L_{k+1}] - \frac{1}{4}h^2B_{k+\frac{1}{2}}L_{k+1}B_{k+\frac{1}{2}} \\ = L_k + \frac{1}{2}h[B_{k+\frac{1}{2}}, L_k] - \frac{1}{4}h^2B_{k+\frac{1}{2}}L_kB_{k+\frac{1}{2}}. \end{aligned}$$

Además,

$$L_{k+1} = L_k + h\left[B_{k+\frac{1}{2}}, \frac{L_k + L_{k+1}}{2}\right] + \frac{1}{4}h^2B_{k+\frac{1}{2}}(L_{k+1} - L_k)B_{k+\frac{1}{2}}. \qquad (41)$$

Es claro de (41) que el método (38)-(40) añade un término extra

$$\frac{1}{4}h^2B_{k+\frac{1}{2}}(L_{k+1} - L_k)B_{k+\frac{1}{2}},$$

a lo que sería la aplicación de la regla implícita del punto medio a (3).

Este término extra mantiene el segundo orden del método porque es $O(h^3)$ y, además, hace que el método sea *isoespectral*. Esto es muy importante porque la regla del punto medio implícita no es *isoespectral*, excepto en el caso $2 \times 2$.

□

# 4. Resultados numéricos con matrices tridiagonales simétricas de distintas dimensiones

En esta sección se describirán los experimentos numéricos llevados a cabo para ilustrar algunos de los resultados teóricos presentados en las secciones anteriores.

Para ello hemos tomado dos familias de matrices tridiagonales y simétricas que serán descritas en apartados posteriores. Para ambos tipos hemos considerado dimensiones

$$4, 8, 16, 32, 64, 128 \text{ y } 256.$$

Por una parte, llevamos a cabo la diagonalización de la matriz con **Algoritmo QR** como ha sido descrito en la Sección 1.

Por otra, utilizamos dos integradores temporales para resolver numéricamente el flujo de Toda asociado.

En primer lugar se ha utilizado el clásico método **Runge-Kutta de orden cuatro,** con tablero de Butcher

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

cuyas ecuaciones para avanzar un paso de longitud $h$ vienen dadas por

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_k), \\ \mathbf{k}_2 &= \mathbf{f}(\mathbf{y}_k + \frac{1}{2}h\mathbf{k}_1), \\ \mathbf{k}_3 &= \mathbf{f}(\mathbf{y}_k + \frac{1}{2}h\mathbf{k}_2), \\ \mathbf{k}_4 &= \mathbf{f}(\mathbf{y}_k + 1h\mathbf{k}_3), \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + h\left(\frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4\right), \end{aligned}$$

tomando como $\mathbf{f}$ e $\mathbf{y}(0)$ los valores dados por el flujo de Toda en (31).

En segundo lugar se ha utilizado la **regla implícita del punto medio modificada,** que es es un método de orden 2 que resulta de aplicar la regla implícita del punto medio al sistema diferencial (32)-(33).

Las ecuaciones que avanzan un paso de longitud $h$ con el método son (38)-(39).

Las longitudes de paso, denotadas por $h$, en la formulación del método Runge-Kutta de orden cuatro y en la regla implícita del punto medio, se han tomado de forma particular dependiendo de la matriz, y en algunos casos de la tolerancia, por lo que se comentará qué valores se han tomado en las secciones siguientes.

Para los tres métodos se ha tomado como criterio de parada el tamaño de los elementos no diagonales de la matriz $L$, es decir, dada una tolerancia, $tol$, los métodos paran una vez que el máximo de los valores absolutos de los elementos no diagonales de dicha matriz es menor que la tolerancia, es decir,

$$\max_{i \neq j} |L_{ij}^{(n)}| \leq tol,$$

siendo $L^{(n)}$ la matriz de la que buscamos calcular los autovalores.

Se han tomado cuatro tolerancias distintas que usaremos para comparar los tres métodos.

$$10^{-3}, 10^{-6}, 10^{-9} \text{ y } 10^{-12}. \tag{42}$$

## 4.1. Matrices tridiagonales (-1, 2, -1)

En un primer experimento, hemos tomado matrices de la forma

$$L(0) = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & 0 & \\ & -1 & 2 & & & \\ & & & \ddots & & \\ & 0 & & & & -1 \\ & & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{43}$$

cuyos autovalores de estas matrices están dados por la fórmula

$$\lambda_k = 2 + 2\cos\left(\frac{\pi \cdot k}{n+1}\right) \quad k = 1, \ldots, n.$$

El valor $\mu$ que gobierna la convergencia de la solución exacta de (31) cuando la condición inicial es (43), como se ha demostrado en el Teorema 2.1.3, teniendo en cuenta que los autovalores $\lambda_k$ están ordenados de forma descendente, es

$$\begin{aligned}\mu &= \min_{k=1,\ldots,n-1}(\lambda_k - \lambda_{k+1}) = \min_{k=1,\ldots,n-1} 2\left(\cos\left(\frac{\pi\cdot k}{n+1}\right) - \cos\left(\frac{\pi\cdot(k+1)}{n+1}\right)\right)\\ &= \min_{k=1,\ldots,n-1} -4\,\mathrm{sen}\left(\frac{\pi\cdot(2k+1)}{2(n+1)}\right)\mathrm{sen}\left(\frac{-\pi}{2(n+1)}\right)\\ &= -4\,\mathrm{sen}\left(\frac{3\pi}{2(n+1)}\right)\mathrm{sen}\left(\frac{-\pi}{2(n+1)}\right) \underset{\text{cuando } n\to\infty}{\longrightarrow} \frac{3\pi^2}{(n+1)^2}\end{aligned}$$

De esta manera podemos calcular el $\mu$ correspondiente para cada una de las dimensiones utilizadas y, obtenemos,

$$\begin{aligned}&\mu_4 = 1, && \mu_8 = 0{,}3473, && \mu_{16} = 0{,}1010,\\ &\mu_{32} = 0{,}0271, && \mu_{64} = 0{,}0070, && \mu_{128} = 0{,}0018,\\ &\mu_{256} = 0{,}00044826.\end{aligned}$$

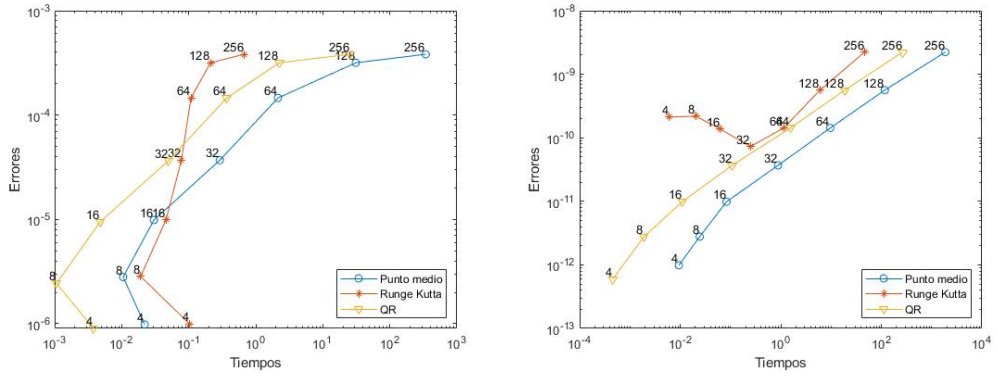
Sin embargo, para el método Runge-Kutta de orden cuatro hemos tomado una longitud de paso, $h$, para cada una de las tolerancias dadas en (42), estas han sido

$$\frac{1}{2^5}, \frac{1}{2^7}, \frac{1}{2^9}, \frac{1}{2^{11}}.$$

La razón es que al tratarse de un método que no conserva los autovalores, de la matriz inicial, aunque los elementos no diagonales de la matriz $L^n$ sean menores que la tolerancia, los errores en los elementos diagonales serán de tamaño $O(h^4)$ y si $h$ no es suficientemente pequeño la aproximación que se obtiene para los autovalores no es adecuada.

Para la regla implícita del punto medio modificada, hemos tomado una longitud de paso que es la misma para las cuatro tolerancias, $h = 0{,}1$, puesto que al tratarse de un modelo *isoespectral*, el error al aproximar los autovalores de $L(0)$ por la diagonal de la matriz $L^n$ sólo depende de que los elementos no diagonales de la matriz sean suficientemente pequeños.

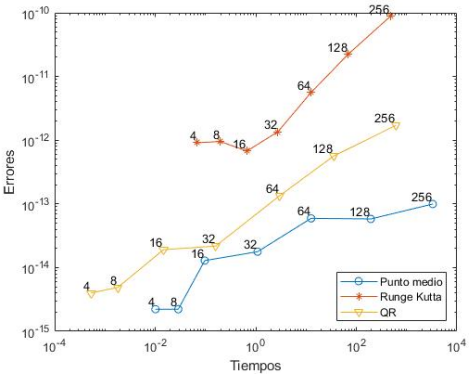
En la Figura 3 se han representado en escala doblemente logarítmica los errores obtenidos tras utilizar los tres métodos para aproximar los autovalores de la matriz (43) frente al tiempo de computación necesario para ello. Se ha indicado la dimensión de la matriz a la que corresponde cada uno de los resultados.



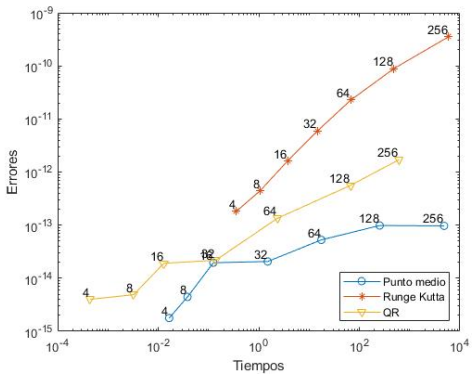

(a) $tol = 10^{-3}$




(b) $tol = 10^{-6}$




(c) $tol = 10^{-9}$




(d) $tol = 10^{-12}$

Figura 3: Resultados de aproximación de los autovalores para las matrices tridiagonales (1, 2, -1) (43)

Podemos observar, como se comenta en la Sección 3, la acumulación de errores en el método Runge-Kutta debido a la no conservación de los autovalores. Mientras que con el algoritmo QR y con la regla implícita del punto medio modificada los errores son debidos a que no se ha llegado a diagonalizar por completo la matriz (el máximo valor absoluto de los elementos no diagonales sólo es menor que la tolerancia utilizada en cada caso), en el método Runge-Kutta hay que añadir, como se ha comentado antes, los errores que se producen al aproximar los autovalores de orden $O(h^4)$. Esto es especialmente visible en las gráficas correspondientes a las tolerancias $10^{-9}$ y $10^{-12}$.

Aunque haría falta realizar más experimentos con matrices de dimensiones

superiores parece que, al disminuir el valor de la tolerancia, comparando el algoritmo QR y la regla implícita del punto medio, la segunda obtiene resultados más precisos, pero que al aumentar la dimensión de la matriz, esta mejora de resultados será a costa de un mayor coste computacional, una tendencia que empieza a hacerse evidente en las gráficas para las tolerancias $10^{-9}$ y $10^{-12}$.

Cabe destacar que en la primera gráfica, para la tolerancia $10^{-3}$ los errores son muy similares, y sólo cambia el coste computacional de cada método.

## 4.2. Matrices tridiagonales tipo Clement

Las matrices tipo Clement [Clement] son matrices de la forma

$$A = \begin{bmatrix} 0 & h_1 & & & & \\ h_1 & 0 & h_2 & & 0 & \\ & h_2 & 0 & & & \\ & & & \ddots & & \\ & 0 & & & & h_{n-1} \\ & & & & h_{n-1} & 0 \end{bmatrix} \in \mathbb{R}^{n\times n},$$

con $n$ par y

$$h_j = \sqrt{j(n-j)}, \quad 1 \leq j \leq n-1.$$

Los autovalores de estas matrices son

$$\pm(n-1), \pm(n-3), \pm(n-5), \ldots, \pm 5, \pm 3, \pm 1.$$

En nuestros experimentos numéricos hemos considerado matrices construidas a partir de las matrices tipo Clement en las que hemos sumando $n$ veces la identidad, es decir,

$$L(0) = \begin{bmatrix} n & h_1 & & & & \\ h_1 & n & h_2 & & 0 & \\ & h_2 & n & & & \\ & & & \ddots & & \\ & 0 & & & & h_{n-1} \\ & & & & h_{n-1} & n \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{44}$$

De esta manera, los autovalores de $L(0)$ son

$$1, 3, 5, \ldots, 2n-5, 2n-3, 2n-1,$$

y el valor $\mu$ que gobierna la convergencia de la solución exacta de (31) con condición inicial (44)

$$\mu = 2.$$

Para la regla implícita del punto medio, hemos tomado una longitud de paso para cada dimensión de la matriz, porque, como los autovalores crecen al aumentar la dimensión el problema es más difícil de integrar y requiere longitudes de paso más pequeñas. Como el ejemplo anterior, las longitudes de paso se mantienen constante con las diferentes tolerancias. Estas han sido

$$0{,}2, 0{,}1, 0{,}05, \ldots, 0{,}003125,$$

Para el método Runge-Kutta de orden cuatro hemos tomado una longitud de paso, $h$, que depende de la tolerancia y de la dimensión de la matriz, estas han sido

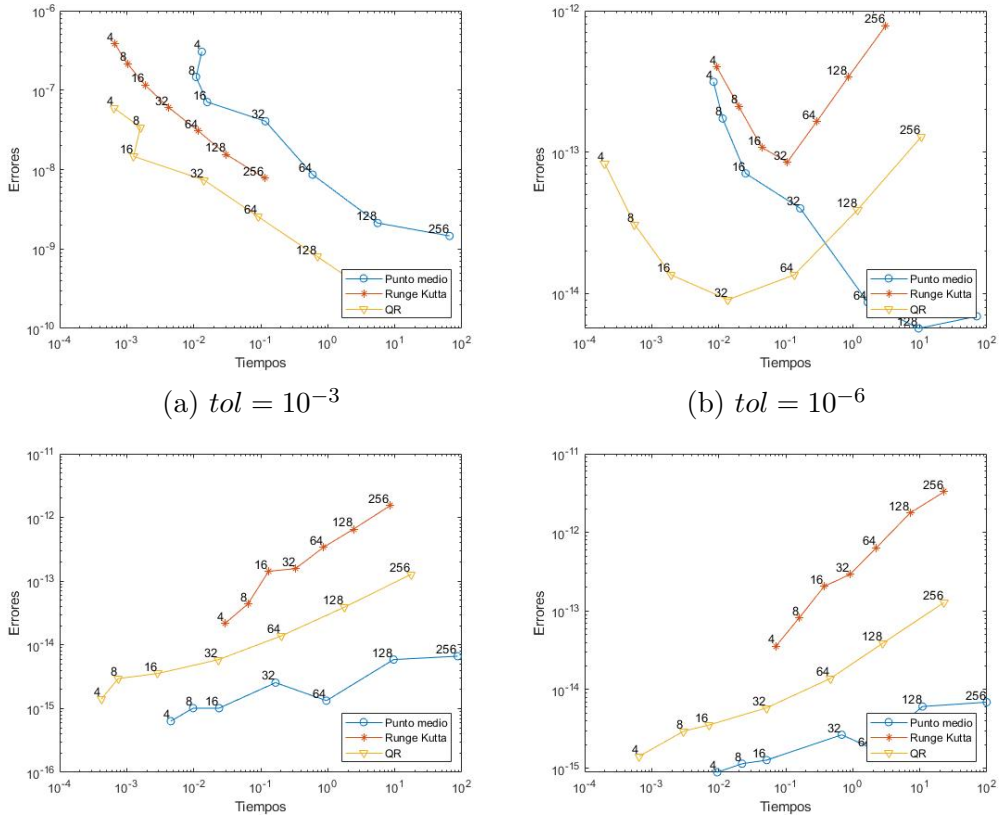
$$\text{Para } tol = 10^{-3}, \quad \frac{[4, 2, 1, 0{,}5, 0{,}25, 0{,}125, 0{,}0625]}{2^7}$$

$$\text{Para } tol = 10^{-6}, \quad \frac{[4, 2, 1, 0{,}5, 0{,}25, 0{,}125, 0{,}0625]}{2^{11}}$$

$$\text{Para } tol = 10^{-9}, \quad \frac{[4, 2, 1, 0{,}5, 0{,}25, 0{,}125, 0{,}0625]}{2^{12}}$$

$$\text{Para } tol = 10^{-12}, \quad \frac{[4, 2, 1, 0{,}5, 0{,}25, 0{,}125, 0{,}0625]}{2^{12}}$$

A continuación se muestran los resultados obtenidos tras utilizar los tres métodos para aproximar los autovalores, estos han sido dibujados en escala doblemente logarítmica al igual que los anteriores. Además se ha indicado la dimensión de la matriz a la que corresponde cada uno de los resultados.




(a) $tol = 10^{-3}$ (b) $tol = 10^{-6}$

(c) $tol = 10^{-9}$ (d) $tol = 10^{-12}$

Figura 4: Resultados de aproximación de los autovalores para las matrices (44) construidas a partir de las matrices tipo Clement.

De nuevo, podemos observar como el método Runge-Kutta sufre una penalización en la precisión del cálculo de los autovalores, debido a la no conservación de estos, que se hace especialmente visible para las tolerancias más pequeñas.

Además, nos encontramos de nuevo, que, entre el algoritmo QR y la regla implícita del punto medio, la segunda obtiene resultados más precisos conforme se disminuye el valor de la tolerancia, pero, igual que antes, se observa una tendencia a un sobrecoste computacional al aumenta la dimensión de la matriz, especialmente evidente en las gráficas para las tolerancias $10^{-9}$ y $10^{-12}$.

Se observa también un menor coste computacional de los métodos de integración numérica para las matrices tridiagonales (44) tipo Clement en comparación con las matrices tridiagonales (1, 2, -1) (43). Esto se explica fácilmente por el Teorema 2.1.3 y el tamaño del $\mu$ de los dos tipos de matrices. Mientras que en las matrices tridiagonales tipo Clement se tiene que $\mu = 2$, para todas las dimensiones, para las matrices tridiagonales (1, 2, -1) , $\mu$ decrece con el aumento de la dimensión, lo que provoca una convergencia mucho más lenta que se ve especialmente reflejado en las matrices de dimensión 256.

# Referencias

[Calvo97] Calvo, M., Iserles, A., & Zanna, A. (1997). Numerical Solution of Isospectral Flows. Mathematics of Computation, 66(220), 1461-1486.

[Calvo99] Mari Paz Calvo, A Iserles, A Zanna, (1999). Conservative methods for the Toda lattice equations, IMA Journal of Numerical Analysis, 19(4), 509–523.

[Clement] Clement, P. (1959). A Class of Triple-Diagonal Matrices for Test Purposes. SIAM Review, 1(1), 50-52.

[Deift80] Deift, P., Lund, F., & Trubowitz, E. (1980). Nonlinear wave equations and constrained harmonic motion. Communications in Mathematical Physics, 74, 141-188.

[Deift83] Deift, P., Nanda, T., & Tomei, C. (1983). Ordinary differential equations and the symmetric eigenvalue problem. SIAM Journal on Numerical Analysis, 20(1), 1-22.

[Hairer] E. Hairer, E., Lubich, C., & Wanner, G. (2006). Geometric numerical integration. Springer.

[Watkins] Watkins, D. S. (1991). Fundamentals of Matrix Computations. John Wiley amp; Sons, Inc.

# A. Programas de Matlab

En este apéndice se muestra el código de los programas de Matlab utilizados para realizar los experimentos numéricos de la Sección 4.

Se han usado los mismos scripts para las matrices tridiagonales (-1, 2, -1) y para las matrices tipo Clement, por eso se han añadido comentarios para indicar que líneas de código hay que comentar para hacer los cálculos con unas matrices u otras.

## A.1. Implementación del Algoritmo QR

En la función `AlgoritmoQRTOL` se lleva a cabo la diagonalización de la matriz $A$ mediante el algoritmo QR. Como criterio de parada se utiliza el valor absoluto de los elementos no diagonales de la matriz.

```
function [D,nPasos] = AlgoritmoQRTOL(A,dim,NDTol)

    nPasos = 0;

    while max(max(abs( A - diag(diag(A),0) ))) > NDTol

       [Q,R] = qr(A);
       A = R*Q;

       nPasos = nPasos + 1;
    end

    D = A;

end
```

## A.2. Toma de datos para el algoritmo QR

El script `TomarDatosQR.m` utiliza la función `AlgoritmoQRTOL`, descrita en el apartado anterior, para tomar datos sobre el tiempo de ejecución y los errores que se cometen al diagonalizar una serie de matrices con diferentes tolerancias.

```
clear;

NDTol = [1e-3 1e-6 1e-9 1e-12]; nTols = length(NDTol);
dim = 2.^(2:8); nDims = length(dim);

erroresQR = zeros(nTols,nDims);
tiemposQR = zeros(nTols,nDims);
nPasosQR = zeros(nTols,nDims);

matricesQR = cell(nTols+1,nDims,1);
for i = 1:nDims
  %Comentar la linea siguiente para trabajar con matrices Clement
  matricesQR{1,i} = full(gallery('tridiag',dim(i),-1,2,-1));

  %Descomentar la linea siguiente para trabajar con matrices Clement
  %matricesQR{1,i} = full(gallery('clement',dim(i),1)) ...
  %                    + eye(dim(i))*dim(i);
end

for j = 1:nTols

 for i = 1:nDims
  fprintf("Iteraci n  %i.%i con dim =  %i\n",j,i,dim(i));

  %Comentar las lineas siguientes para trabajar con matrices Clement
  A = full(gallery('tridiag',dim(i),-1,2,-1));
  autovalores = sort(2 - 2*cos(pi*(1:dim(i))'/(dim(i)+1)));

  %Descomentar las lineas siguientes para trabajar con matrices Clement
  %A = full(gallery('clement',dim(i),1)) + eye(dim(i))*dim(i);
  %autovalores = (1:2:2*dim(i)-1)';

  fprintf("Comienza el algoritmo QR para dim =  %i\n",dim(i));

  tic
  [LNQR, nPasos] = AlgoritmoQRTOL(A,dim(i),NDTol(j));
  tiemposQR(j,i) = toc;

  fprintf("Fin del algoritmo QR para dim = %i." ...
      + " Tiempo total = %i\n", dim(i),tiemposQR(j,i));

  autovaloresQR = sort(diag(LNQR));

  %Comentar la linea siguiente para trabajar con matrices Clement
  erroresQR(j,i) = norm(autovalores-autovaloresQR,'Inf');
```

```
  %Descomentar la linea siguiente para trabajar con matrices Clement
  %erroresQR(j,i) =
        %norm((autovalores-autovaloresQR)./autovalores,'Inf');

  matricesQR{j+1,i} = LNQR;
  nPasosQR(j,i) = nPasos;

 end

end

%Comentar la linea siguiente para trabajar con matrices Clement
save("DatosQR.mat");

%Descomentar la linea siguiente para trabajar con matrices Clement
%save("DatosQRClement.mat");
```

## A.3. Implementación de la regla implícita del punto medio

En la función `PuntoMedioTOL` se lleva a cabo la integración del flujo de Toda (1) mediante la regla implícita del punto medio. Como criterio de parada se utiliza el valor absoluto de los elementos no diagonales de la matriz.

```
function [L,nPasos] = PuntoMedioTOL(L0,dim,NDTol,h)

    L = L0;
    U = eye(dim);

    tol = 1e-15;
    nPasos = 0;

    while max(max(abs( L - diag(diag(L),0) ))) > NDTol

        Upuntomedio = U;
        Lpuntomedio = L;

        dif = tol + 1;

        while dif > tol
            UpuntomedioI = Upuntomedio;
```

```
            Upuntomedio = U + (h/2)*Be(Lpuntomedio)*Upuntomedio;
            Lpuntomedio = Upuntomedio*L0*Upuntomedio';

            dif = max(max(abs(UpuntomedioI - Upuntomedio)));
        end

        U = U + h*Be(Lpuntomedio)*Upuntomedio;
        L = U*L0*U';

        nPasos = nPasos + 1;
    end

    function B = Be(A)

        A = (A + A')/2;
        B = tril(A,-1)' - tril(A,-1);
    end

end
```

## A.4. Toma de datos para la regla implícita del punto medio

El script `TomarDatosPM.m` utiliza la función `PuntoMedioTOL`, descrita en el apartado anterior, para tomar datos sobre el tiempo de ejecución y los errores que se cometen al integrar el flujo de Toda para diferentes dimensiones de la matriz y diferentes tolerancias.

```
clear;

%Comentar la linea siguiente para trabajar con matrices Clement
hPM = 0.1;

%Descomentar la linea siguiente para trabajar con matrices Clement
%hPM = [0.2 0.1 0.05 0.025 0.0125 0.00625 0.003125];

NDTol = [1e-3 1e-6 1e-9 1e-12]; nTols = length(NDTol);
dim = 2.^(2:8); nDims = length(dim);

erroresPM = zeros(nTols,nDims);
tiemposPM = zeros(nTols,nDims);
```

```
nPasosPM = zeros(nTols,nDims);

matricesPM = cell(nTols+1,nDims,1);
for i = 1:nDims
 %Comentar la linea siguiente para trabajar con matrices Clement
 matricesPM{1,i} = full(gallery('tridiag',dim(i),-1,2,-1));

 %Descomentar la linea siguiente para trabajar con matrices Clement
 %matricesPM{1,i} = full(gallery('clement',dim(i),1)) ...
 %                      + eye(dim(i))*dim(i);
end

for j = 1:nTols

 for i = 1:nDims
  fprintf("Iteraci n  %i.%i con dim =  %i\n",j,i,dim(i));

  %Comentar las lineas siguientes para trabajar con matrices Clement
  A = full(gallery('tridiag',dim(i),-1,2,-1));
  autovalores = sort(2 - 2*cos(pi*(1:dim(i))'/(dim(i)+1)));

  %Descomentar las lineas siguientes para trabajar con matrices Clement
  %A = full(gallery('clement',dim(i),1)) + eye(dim(i))*dim(i);
  %autovalores = (1:2:dim(i)*2 - 1)';

  fprintf("Comienza el m todo del punto medio para dim =  %i\n",dim(i));
  tic
  [LNPM,nPasos] = PuntoMedioTOL(A,dim(i),NDTol(j),hPM(i));
  tiemposPM(j,i) = toc;

  fprintf("Fin el m todo del punto medio para dim =  %i." ...
          +"Tiempo total =  %i\n",dim(i),tiemposPM(j,i));

  autovaloresPM = sort(diag(LNPM));

  %Comentar la linea siguiente para trabajar con matrices Clement
  erroresPM(j,i) = norm(autovalores-autovaloresPM,'Inf');

  %Descomentar la linea siguiente para trabajar con matrices Clement
  %erroresPM(j,i) = norm((autovalores-autovaloresPM)./autovalores,'Inf');


  matricesPM{j+1,i} = LNPM;
  nPasosPM(j,i) = nPasos;
```

```
 end

end

%Comentar la linea siguiente para trabajar con matrices Clement
save("DatosPM.mat");

%Descomentar la linea siguiente para trabajar con matrices Clement
%save("DatosPMClement.mat");
```

## A.5. Implementación del método Runge-Kutta de orden cuatro

En la función `RungeKutta4TOL` se lleva a cabo la integración del flujo de Toda (1) mediante un método Runge-Kutta de orden 4. Como criterio de parada se utiliza el valor absoluto de los elementos no diagonales de la matriz.

```
function [D,nPasos] = RungeKutta4TOL(A,dim,NDTol,h)

    a = diag(A,0)';
    b = [ 0 diag(A,1)' 0];

    y0 = [a,b];
    [y, nPasos] = RK4(y0);

    D = diag(y(1:dim),0) + diag(y(dim + 2: 2*dim),1) ...
        + diag(y(dim + 2: 2*dim),-1);

    function [y, nPasos] = RK4(y0)

        y = y0;
        nPasos = 0;

        while max(abs(y(dim + 2: 2*dim))) > NDTol

            k1 = f(y);
            k2 = f(y + h*k1/2);
            k3 = f(y + h*k2/2);
            k4 = f(y + h*k3);
            y =  y + h*(k1 + 2*k2 + 2*k3 + k4)/6;

            nPasos = nPasos + 1;
```

```
        end

    end


    function yn = f(y)

        yn = zeros(1,2*dim + 1);

        for i = 1:(dim)
           yn(i) = 2*(y(i + dim + 1)^2 - y(i + dim)^2);
           yn(i + dim + 1) = y(i + dim + 1)*(y(i+1) - y(i));
        end


    end
end
```

## A.6. Toma de datos para el método Runge-Kutta de orden 4

El script `TomarDatosRK.m` utiliza la función `RungeKutta4TOL`, descrita en el apartado anterior, para tomar datos sobre el tiempo de ejecución y los errores que se cometen al integrar el flujo de Toda para diferentes dimensiones de la matriz y diferentes tolerancias.

```
clear;

%Comentar la linea siguiente para trabajar con matrices Clement
hRK = (1/2).^[5 7 9 11];

%Descomentar la linea siguiente para trabajar con matrices Clement
%hRK = [4 2 1 0.5 0.25 0.125 0.0625]'*((1/2).^[7 11 12 13]);

NDTol = [1e-3 1e-6 1e-9 1e-12]; nTols = length(NDTol);
dim = 2.^(2:8); nDims = length(dim);


erroresRK = zeros(nTols,nDims);
tiemposRK = zeros(nTols,nDims);
nPasosRK = zeros(nTols,nDims);
```

```
matricesRK = cell(nTols+1,nDims,1);
for i = 1:nDims
 %Comentar la linea siguiente para trabajar con matrices Clement
 matricesRK{1,i} = full(gallery('tridiag',dim(i),-1,2,-1));

 %Descomentar la linea siguiente para trabajar con matrices Clement
 %matricesRK{1,i} = full(gallery('clement',dim(i),1)) ...
 %                     + eye(dim(i))*dim(i);
end

for j = 1:nTols

 for i = 1:nDims
  fprintf("Iteraci n  %i.%i con dim =  %i\n",j,i,dim(i));

  %Comentar la lineas siguientes para trabajar con matrices Clement
  A = full(gallery('tridiag',dim(i),-1,2,-1));
  autovalores = sort(2 - 2*cos(pi*(1:dim(i))'/(dim(i)+1)));


  %Descomentar la lineas siguientes para trabajar con matrices Clement
  %A = full(gallery('clement',dim(i),1)) + eye(dim(i))*dim(i);
  %autovalores = (1:2:dim(i)*2 - 1)';

  fprintf("Comienza el m todo Runge Kutta para dim =  %i\n",dim(i));
  tic
  [LNRK, nPasos] = RungeKutta4TOL(A,dim(i),NDTol(j),hRK(i,j));
  tiemposRK(j,i) = toc;

  fprintf("Fin el m todo Runge Kutta para dim =  %i." ...
      + "Tiempo total =  %i\n", dim(i),tiemposRK(j,i));

  autovaloresRK = sort(diag(LNRK));


  %Comentar la linea siguiente para trabajar con matrices Clement
  erroresRK(j,i) = norm(autovalores-autovaloresRK,'Inf');

  %Descomentar la linea siguiente para trabajar con matrices Clement
  %erroresRK(j,i) = norm((autovalores-autovaloresRK)./autovalores,'Inf');

  matricesRK{j+1,i} = LNRK;
  nPasosRK(j,i) = nPasos;
```

```
 end
end

%Comentar la linea siguiente para trabajar con matrices Clement
save("DatosRK.mat");

%Descomentar la linea siguiente para trabajar con matrices Clement
%save("DatosRKClement.mat");
```

## A.7. Comparación de los datos tomados

El script `GraficosComparacion.m`

```
%Comentar la lineas siguientes para trabajar con matrices Clement
load("DatosPM.mat");
load("DatosRK.mat");
load("DatosQR.mat");

%Descomentar la lineas siguientes para trabajar con matrices Clement
% load("DatosPMClement.mat");
% load("DatosRKClement.mat");
% load("DatosQRClement.mat");


for j = 1:nTols
 figure(j);
 loglog(tiemposPM(j,:),erroresPM(j,:),"o-");
 text(tiemposPM(j,:),erroresPM(j,:), ...
     {'4','8','16','32','64','128','256'}, ...
     'VerticalAlignment','bottom','HorizontalAlignment','right');
 hold on;

 loglog(tiemposRK(j,:),erroresRK(j,:),"*-");
 text(tiemposRK(j,:),erroresRK(j,:), ...
     {'4','8','16','32','64','128','256'}, ...
     'VerticalAlignment','bottom','HorizontalAlignment','right');
 hold on;

 loglog(tiemposQR(j,:),erroresQR(j,:),"v-");
 text(tiemposQR(j,:),erroresQR(j,:), ...
     {'4','8','16','32','64','128','256'}, ...
     'VerticalAlignment','bottom','HorizontalAlignment','right');
```

```
 xlabel('Tiempos');
 ylabel('Errores');
 legend('Punto medio','Runge Kutta','QR','Location','SE');
end
```

muestra cuatro gráficos, uno para tolerancia dada, de escalas doblemente logarítmicas, donde se comparan los errores obtenidos y los tiempos de ejecución de los tres métodos.