Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN GRADO

Grado en Matemáticas

Métodos de Prony para la recuperación de funciones

Autor: Adrián García Morquecho.
Tutor: Ángel Durán Martín.
2024/2025

Índice general

Resúmen: En esta memoria se exponen el método Clásico de Prony y sus generalizaciones. Este tipo de métodos se emplean como solución al problema de recuperación de funciones estructuradas. Se explica el método clásico para muestras equiespaciadas y no equiespaciadas, así como su implementación con factorización QR y con descomposición en valores singulares (método ESPRIT). El método generalizado extiende la representación de las funciones a desarrollos finitos de autofunciones de un operador lineal. El método es aplicado a diferentes problemas, según el operador elegido. Tanto el método clásico como su generalización quedan finalmente ilustrados con experimentos numéricos.

Palabras clave:Método de Prony, recuperación de señales, ESPRIT, vectores dispersos, desarrollos dispersos.

Abstract: In this work, the classical Prony method and its generalisations are set out. This type of methods is used as a solution to the problem of structured function recovery. Different versions of the classical method, for equispaced and nonequispaced samplings, are explained, as well as their implementation using QR factorization and SVD. Prony's method is then extended to sparse representations of functions in a finite number of eigenfunctions of a linear operator. The generalization is exemplified with different choices of the operator. Both classical and generalized methods are finally illustrated with several numerical experiments.

Key words: Prony's method, signal recovering, ESPRIT method, sparse vectors, sparse expansion.

Capítulo 1

INTRODUCCIÓN

1.1. El problema de la recuperación de una función estructurada a partir de muestras.

En esta memoria se expone la teoría necesaria para solucionar de forma numérica uno de los problemas más recurrentes en diversos campos científicos, y que se conoce como recuperación de una función estructurada a partir de muestras con ruido. Una de la soluciones a este problema, y que será la desarrollada más adelante, viene dada por los métodos de tipo Prony, que tienen su origen en el matemático francés Gaspard de Prony (22 de julio de 1755 - 29 de julio de 1839). Estos métodos surgieron a finales del S. XVIII, pero debido a la ausencia de ordenadores, su interés se vio retrasado hasta el inicio del S. XX con el avance de los mismos. En particular, se propone la solución numérica de tres problemas de alto interés.

(1) Se trata de recuperar los parámetros $M \in \mathbb{N}, c_j \in \mathbb{C}\backslash\{0\}$ y $\lambda_j \in [-\alpha, 0] + i[-\pi, \pi), j = 1, \ldots, M$, para $\alpha > 0$, de una función

$$f(x) := \sum_{j=1}^{M} c_j e^{\lambda_j x}, \qquad x \in \mathbb{R}^+ \cup \{0\},$$

conociéndose, a priori, los valores muestrales (con posibilidad de errores de medición), $f_k := f(k) + e_k, k = 0, \ldots, 2N-1$, siendo $N \in \mathbb{N}$, una cota superior adecuada de M y e_k errores de medición. Puesto que $Re(\lambda_j)$

e $Im(\lambda_j)$ son la amplitud, y frecuencia angular de la exponencial $e^{\lambda_j x}$, tiene sentido denominar a este problema análisis de frecuencias (e.g. [14]) y tiene interés en ingeniería eléctrica y física matemática, entre otras áreas.

(2) A partir de una cantidad finita de valores de la transformada de Fourier de un polinomio definido a trozos de soporte compacto, se trata de recuperar los nodos, y los respectivos saltos en ellos. Este problema aparece en áreas como la medicina (resonancias magnéticas, tomografías, etc.) o radioastronomía, véase e.g. [2].

(3) Por último, otra aplicación directa de los métodos de Prony, es la recuperación de los denominados vectores $\vec{x} \in \mathbb{D}, D \in \mathbb{N}$, M-dispersos, que son aquellos en los que únicamente $M \ll D$ componentes son no nulas. El interés de este problema surje cuando se necesita reconstruir una señal $\vec{x}$ a partir de una cantidad pequeña de medidas, ya sea porque la muestra es volátil, costosa o lleva mucho tiempo tomar dicha medida. Esto se observa en áreas como el análisis de seismos o testeo no destructivo de materiales (e.g. [15]).

1.2. Estructura del TFG.

Este TFG se desarrolla en dos principales partes, diferenciadas en capítulos que se suceden de forma natural, es decir, el capítulo 2 desarrolla los métodos de tipo Prony más clásicos, empezando por el caso más simple, que es el de recuperar la función

$$f(x) = \sum_{j=1}^{M} c_j e^{\lambda_j x},$$

cuando se conoce el orden M, y las muestras $f(k), k = 0, \ldots, 2M-1$ son exactas. Posteriormente, en este mismo capítulo se da un siguiente paso, en el que M ya no es conocido, y se da la posibilidad de pequeños errores en las mediciones empleadas para recuperar la función. El capítulo prosigue con la

extensión del método cuando los valores conocidos se dan en nodos no necesariamente equiespaciados, que consiste realmente en hacer un pre-procesado de los datos para poder emplear los métodos anteriores. Finalmente, se culmina con la presentación de varios ejemplos numéricos que resaltan las cualidades de este tipo de métodos así como las diferencias entre los mismos.
La motivación del capítulo 3 es extender los métodos desarrollados hasta ahora a un paradigma más general, en el que no será necesario restringirse a desarrollos de exponenciales complejas, si no que se introducirá el conceptos de autofunciones de un operador lineal, permitiendo generalizar la teoría expuesta previamente para buscar representaciones finitas de la función a recuperar en las autofunciones de cierto operador lineal. Diferentes selecciones del operador nos sirven para ilustrar la formulación y la variedad de problemas que se pueden abordar. Este capítulo se cierra, de nuevo, con ejemplos númericos de interés.
Por último, se añaden dos anexos, en el apéndice A se recoge la información necesaria para entender algunos de los conceptos empleados para desarrollar el método de Prony, como pueden ser los lápices matriciales y los autovalores generalizados. Mientras que en el apéndice B, se exponen los códigos de Matlab empleados en los experimentos numéricos.

Capítulo 2

EL MÉTODO CLÁSICO DE PRONY

2.1. El algoritmo y sus propiedades.

El método clásico de Prony resuelve el problema de recuperación de una función $f : \mathbb{R} \to \mathbb{C}$ expresada mediante una suma de M exponenciales complejas, con $M \geq 1$ entero conocido y 2M-1 evaluaciones en nodos equiespaciados de dicha suma. Es decir, se pretende recuperar la siguiente función:

$$f(x) = \sum_{j=1}^{M} c_j e^{\lambda_j x}, \quad c_j \in \mathbb{C}, \quad \lambda_j \in \mathbb{C}, \quad j = 1, \ldots, M, \tag{2.1}$$

donde son conocidos los valores

$$f(k) = \sum_{j=1}^{M} c_j z_j^k, \qquad k = 0, 1, \ldots, 2M-1, \tag{2.2}$$

con $z_j = e^{\lambda_j}, j = 1, \ldots, M$, y las incógnitas son los valores $\lambda_j \in \mathbb{C}$, y los coeficientes $c_j \in \mathbb{C}, c_j \neq 0, j = 1, \ldots M$.

Supondremos también que $\lambda_j \neq \lambda_i$ para $j \neq i$, ya que si fuesen iguales, entonces $e^{\lambda_j} = e^{\lambda_i}$, y al sumar $c_j e^{\lambda_j} + c_i e^{\lambda_i} = (c_j + c_i) e^{\lambda_j}$ y bastaría renombrar $c_{j'} := c_j + c_i$.

En la mayoría de las aplicaciones se supone

$$-\alpha \leq Re\ \lambda_j \leq 0, \tag{2.3}$$
$$|Im\ \lambda_j| < \pi, \qquad j = 1 \ldots, M, \tag{2.4}$$

para cierto $\alpha > 0$. Desde este punto en adelante asumiremos que es así. En la sección 2.1.1 comentaremos reformulaciones del método que permiten relajar las hipótesis (2.3) y (2.4).

Para describir el método, definimos el siguiente polinomio (llamado polinomio de Prony):

$$p(z) := \prod_{j=1}^{M}(z - z_j) = z^M + \sum_{j=0}^{M-1} a_j z^j, \tag{2.5}$$

donde los z_j son distintos dos a dos[1], y los a_j son los coeficientes resultantes de desarrollar p en potencias de z.

Si definimos $a_M := 1$, observemos, a partir de (2.1), que para todo entero $m \geq 0$ se da la siguiente igualdad

$$\begin{aligned}\sum_{k=0}^{M} a_k f(m+k) &= \sum_{k=0}^{M} a_k \Big(\sum_{j=1}^{M} c_j e^{\lambda_j (m+k)}\Big) = \sum_{k=0}^{M}\sum_{j=1}^{M} a_k c_j z_j^m z_j^k \\ &= \sum_{j=1}^{M} c_j z_j^m p(z_j) = 0.\end{aligned} \tag{2.6}$$

En particular, usando (2.2), se tienen las ecuaciones lineales en diferencias

$$\sum_{k=0}^{M-1} a_k f(m+k) = -f(m+M). \qquad m = 0, ..., M-1. \tag{2.7}$$

La determinación de los coeficientes de p en (2.5) puede llevarse a cabo a partir de la interpretación de (2.7) como un sistema lineal para $a_0, \ldots, a_{M-1}$,

[1]Puesto que $Im(\lambda_j) \in [-\pi, \pi), \forall j$, si $z_j = z_i$, con $j \neq i \Rightarrow e^{\lambda_j} = e^{\lambda_i} \Rightarrow \lambda_j = \lambda_i$ y hemos supuesto que todos eran distintos.

expresado de la forma

$$H_M \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{M-1} \end{bmatrix} = - \begin{bmatrix} f(M) \\ f(M+1) \\ \vdots \\ f(2M-1) \end{bmatrix}, \tag{2.8}$$

donde

$$H_M = \begin{bmatrix} f(0) & f(1) & \dots & f(M-1) \\ f(1) & f(2) & \dots & f(M) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1) & f(M) & \dots & f(2M-2) \end{bmatrix}. \tag{2.9}$$

El sistema (2.8) tiene solución única como consecuencia del siguiente resultado.

Lema 2.1. *Sean* $\vec{c} = (c_1, \dots, c_M)^T, \vec{z} = (z_1, \dots, z_M)^T$ *con* c_i *y* z_i, $i = 1, \dots, M$, *definidos en (2.2) y la matriz de Vandermonde*

$$V_M(\vec{z}) = (z_k^{j-1})_{j,k=1}^{M},$$

generada por $\vec{z}$. *Entonces*

$$H_M = V_M(\vec{z})\ diag(\vec{c})\ V_M^T(\vec{z}). \tag{2.10}$$

En particular, H_M *es invertible, y el vector* $\vec{c}$ *es solución del sistema*

$$V_M(\vec{z})\ \vec{c} = \begin{bmatrix} f(0) \\ \vdots \\ f(M-1) \end{bmatrix}. \tag{2.11}$$

Demostración. Haciendo el producto de matrices del lado derecho en (2.10), se tiene

$$\begin{bmatrix} 1 & \vdots & 1 \\ z_1 & \dots & z_M \\ \vdots & \ddots & \vdots \\ z_1^{M-1} & \dots & z_M^{M-1} \end{bmatrix} \begin{bmatrix} c_1 & \dots & 0 \\ \vdots & \ddots & \\ 0 & \dots & c_M \end{bmatrix} \begin{bmatrix} 1 & z_1 & \vdots & z_1^{M-1} \\ 1 & z_2 & \dots & z_2^{M-1} \\ \vdots & \ddots & \vdots & \\ 1 & z_M & \dots & z_M^{M-1} \end{bmatrix} =$$

$$\begin{bmatrix} c_1 + ... + c_M & \dots & c_1 z_1^{M-1} + ... + c_M z_M^{M-1} \\ \vdots & \ddots & \vdots \\ c_1 z_1^{M-1} + ... + c_m z_M^{M-1} & \dots & c_1 z_1^{2M-2} + ... + c_M z_M^{2M-2} \end{bmatrix}. \tag{2.12}$$

Entonces, el elemento (i,j) de la matriz resultante es, utilizando (2.2),

$$\sum_{k=1}^{M} c_k z_k^{i+j-2} = f(i+j-2), \qquad i,j = 1,\dots,M,$$

lo que demuestra (2.10).
Por otro lado, puesto que $z_i \neq z_j, \forall\ i,j$, la matriz $V_M(z)$ es invertible (por las propiedades de las matrices de Vandermonde); además, como $c_i \neq 0, \forall i$, la matriz $diag(\vec{c})$ es invertible. En conclusión, se tiene que H_M es invertible por ser producto de matrices invertibles.
Por último, para ver que $\vec{c}$ es solución de (2.11) basta considerar la primera columna en (2.12).

□

La determinacióm del polinomio (2.5) permite la obtención de las incógnitas del problema (2.1) a partir de las etapas siguientes que constituyen el llamado método clásico de Prony.

2.1.1. Algoritmo (Método clásico de Prony).

Entradas: M∈ $\mathbb{N}$, $f(k)$ valores medidos , $k = 0, ..., 2M-1$, de la función a recuperar.

Paso 1) Resolver el sistema lineal (2.8).

Paso 2) Hallar las raices $z_i = e^{\lambda_i}, i = 1, ..., M$, del polinomio (2.5) de Prony, conocido por el paso 1. A partir de ellas calcular los λ_i aplicando la rama principal del logaritmo, ya que hemos supuesto $Im(\lambda_i) \in [-\pi, \pi)$, de manera que $\lambda_i := log(z_i), \quad i = 1, ..., M$.

Paso 3) Los coeficientes $\vec{c} = (c_1, \dots, c_M)^T$ se obtienen a partir de la resolución del sistema lineal (2.11).

Salidas: $c_i \in \mathbb{C}, \lambda_i \in [-\alpha, 0] + i[-\pi, \pi),\ i = 1, ..., M.$

■

Pueden hacerse varios comentarios sobre el algoritmo anterior.

(i) Para el paso 2 del algoritmo necesitamos hallar las raices $z_i, i = 1 \ldots, M$, del polinomio de Prony. Este problema puede abordarse desde dos puntos de vista, teóricamente equivalentes pero con distintos procedimientos numéricos de resolución.

Por un lado, podemos considerar la matriz compañera del polinomio p,

$$C_M(p) = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{M-1} \end{bmatrix} \in \mathbb{C}_{M\times M}, \tag{2.13}$$

es decir, $det(zId_M - C_M(p)^T) = p(z)$, por lo tanto hallar las raíces de p será equivalente a encontrar los autovalores de la matriz $C_M(p)$.

Otra forma equivalente de afrontar el problema es considerar las raíces de p como ceros de una función polinomica. Es conocida la imposibilidad de resolver teóricamente el cálculo de raíces de un polinomio, por lo que es necesario introducir también técnicas numéricas de aproximación. Para obtener aproximaciones a los autovalores de una matriz, la literatura ofrece gran cantidad de métodos, desde los más elementales (método de la potencia, algoritmo QR, etc) hasta los más sofisticados y aplicables a determinados tipos de matrices, véase [7].
Un comentario similar puede hacerse buscando las raíces z_i como ceros de un polinomio, [21].

(ii) El método de Prony se puede extender al caso en el que los coeficientes c_i sean polinomios (véase [1]), es decir la representación (2.1) sería de la forma, dado un $n \in \mathbb{N}$,

$$f(x) = \sum_{i=1}^{M} c_i(x) e^{\lambda_i x}, \quad c_i(x) \in \mathbb{C}_{\leq n}[x], \quad \lambda_i \in \mathbb{C}, \quad i = 1, \ldots, M.$$

Dicha extensión tiene limitaciones, ya que cuanto mayor sea el grado de los polinomios $c_i(x)$ el coste computacional del método crece enormemente haciéndolo ineficiente. A modo de ilustración, se desarrolla el caso $n = 1$, suponiendo conocidos $f(k), k = 0, \ldots, 4M-1$ (será necesaria una muestra de tamaño 4M), y siendo

$$c_i(x) = c_{i,0} + c_{i,1}x, \qquad i = 1, \ldots, M.$$

Consideramos el polinomio

$$p(x) = \prod_{i=1}^{M} (z - z_i)^2 = z^{2M} + \sum_{j=0}^{2M-1} p_j z^j, \tag{2.14}$$

con $z_i = e^{\lambda_i}, i = 1, \ldots, M$, y $p_j, j = 0, \ldots, 2M-1, p_{2M} = 1$, los coeficientes al desarrollar el polinomio en potencias de z. La relación (2.6) es ahora de la forma

$$\begin{aligned}
\sum_{k=0}^{2M} p_k f(m+k) &= \sum_{k=0}^{2M} p_k \Big(\sum_{j=1}^{M} \{c_{j,0} + c_{j,1}(m+k)\} z_j^{m+k}\Big) \\
&= \sum_{j=1}^{M} z_j^m \{c_{j,0}\Big(\sum_{k=0}^{2M} p_k z_j^k\Big) + c_{j,1}\Big(\sum_{k=0}^{2M} p_k (m+k) z_j^k\Big)\} \\
&= \sum_{j=1}^{M} z_j^m \{c_{j,0} p(z_j) + c_{j,1} m p(z_j) + c_{j,1} z_j p'(z_j)\} = 0,
\end{aligned}$$

para $m = 0, \ldots, 2M-1$. De este modo, los coeficientes p_j se obtienen a partir de un sistema lineal análogo a (2.7),

$$\sum_{k=0}^{2M-1} p_k f(m+k) = -p_{2M} f(m+2M) = -f(m+2M),$$

$m = 0, \ldots, 2M-1$, es decir

$$H_{2M} \begin{bmatrix} p_0 \\ \vdots \\ p_{2M-1} \end{bmatrix} = - \begin{bmatrix} f(2M) \\ \vdots \\ f(4M-1) \end{bmatrix}, H_{2M} = \begin{bmatrix} f(0) & \ldots & f(2M-1) \\ \vdots & \ddots & \vdots \\ f(2M-1) & \ldots & f(4M-2) \end{bmatrix}.$$

Conocido el polinomio p en (2.14), el siguiente paso del algoritmo debe calcular las correspondientes raíces dobles z_i, o bien hallar los autovalores de la matriz compañera. En el último paso, la matriz de Vandermonde del lema 2.2 ha de sustituirse por la confluente (e.g. [11]) $V_{2M}^c(z_1, \ldots, z_M)$, de la forma

$$\begin{bmatrix} 1 & 0 & \ldots & 1 & 0 & \ldots & 1 & 0 \\ z_1 & 1 & \ldots & z_j & 1 & \ldots & z_M & 1 \\ z_1^2 & 2z_1 & \ldots & z_j^2 & 2z_j & \ldots & z_M^2 & 2z_M \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ z_1^{2M-1} & (2M-1)z_1^{2M-2} & \ldots & z_j^{2M-1} & (2M-1)z_j^{2M-2} & \ldots & z_M^{2M-1} & (2M-1)z_M^{2M-2} \end{bmatrix}$$

Se puede comprobar entonces que la correspondiente versión de (2.10) es

$$H_{2M} = V_{2M}^c \; C \; (V_{2M}^c)^T \tag{2.15}$$

donde C es la matriz $2M \times 2M$ por bloques

$$C = \begin{bmatrix} c_{1,0} & z_1 c_{1,1} & \ldots & 0 & 0 \\ z_1 c_{1,1} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & c_{M,0} & z_M c_{M,1} \\ 0 & 0 & \ldots & z_m c_{M,1} & 0 \end{bmatrix},$$

y la primera columna de (2.15) proporciona el sistema

$$V_{2M}^c \begin{bmatrix} c_{1,0} \\ z_1 c_{1,1} \\ \vdots \\ c_{M,0} \\ z_M c_{M,1} \end{bmatrix} = \begin{bmatrix} f(0) \\ \vdots \\ f(2M-1) \end{bmatrix},$$

del que pueden calcularse los coeficientes $c_{i,0}, c_{i,1}, i = 1, \ldots, M$, completando así el tercer paso del algoritmo. La generalización a polinomios

$c_i(x)$ de grado mayor parece directa pero no recomendable, ya que es claro que un mayor grado aumenta la complejidad del algoritmo en gran medida.

(iii) El algoritmo puede reformularse en función de las hipótesis (2.3) y (2.4). Por un lado, la condición (2.3) es típica en las aplicaciones del método (en teoría de la señal, por ejemplo) y conveniente para dar cierta estabilidad al algoritmo, pero parece claro que no es necesaria para la formulación.
Por otra parte, si en lugar de (2.4) suponemos

$$Im\lambda_i \in [a, a + 2\pi), \quad i = 1, \dots, M,$$

para cierto $a \in \mathbb{R}$, la construcción del algoritmo sigue siendo válida si al calcular λ_i cambiamos, en el paso 2, la rama principal del logaritmo, por la rama correspondiente.
Por último, el algoritmo puede también reformularse eliminando la hipótesis (2.4) del siguiente modo. Sea $T > 0$ verificando

$$|Im\ \lambda_i| < T, \quad i = 1, \dots, M, \tag{2.16}$$

y $h > 0$ con $h < \pi/T$. En vez de (2.2) se suponen conocidos los valores

$$F(k) := f(kh) = \sum_{i=1}^{M} c_i \widetilde{z}_i^k, \qquad k = 0, \dots, 2M - 1,$$

con $\widetilde{z}_i = e^{\tilde{\lambda}_i}, \tilde{\lambda}_i = \lambda_i h, i = 1, \dots, M$. Debido a la elección de T en (2.16) y del salto h, se tiene

$$|Im(\tilde{\lambda}_i)| = h|Im(\lambda_i)| < \pi, \qquad i = 1, \dots, M.$$

De este modo, todos los pesos del algoritmo son válidos sustituyendo (λ_i, z_i) por $(\tilde{\lambda}_i, \tilde{z}_i)$, $i = 1, \dots, M$, y recuperando, en el paso 2 del algoritmo

$$\lambda_i = \frac{1}{h} log(\tilde{z}_i), \qquad i = 1, \dots, M.$$

2.1.2. Propiedades del método.

En esta sección veremos algunas propiedades matemáticas del método clásico de Prony. La primera está relacionada con la transformada Z y los aproximantes de Padé de una función, e.g [23].

1. Método de Prony como aproximación de Padé.

En primer lugar, resulta necesario definir la transformada Z ya que nos basaremos en ella para exponer esta propiedad.

Definición 2.1. *Sea* $\{c_n\}_{n=-\infty}^{\infty} \subset \mathbb{C}$ *una sucesión de numeros complejos tal que la serie de Laurent*

$$\sum_{n=-\infty}^{\infty} c_n z^{-n},$$

es convergente en una corona

$$C = \{z \in \mathbb{C} : \rho_1 < |z| < \rho_2\},$$

para ciertos $\rho_1, \rho_2 > 0$ *. Para* $z \in C$*, se define la transformada Z de* $\{c_n\}_{n=-\infty}^{\infty}$ *como la suma de la serie*

$$F(z) = \sum_{n=-\infty}^{\infty} c_n z^{-n}. \tag{2.17}$$

A continuación, se mencionan algunas propiedades elementales de la transformada Z (e.g. sección 10, [13]):

Teorema 2.1. *Sean dos sucesiones* $\{c_n\}_{n=-\infty}^{\infty}$ *y* $\{s_n\}_{n=-\infty}^{\infty} \subset \mathbb{C}$*, que admiten transformada Z en las coronas* $C_c = \{z \in \mathbb{C} : \rho_1 < |z| < \rho_2\}$ *y* $C_s = \{z \in \mathbb{C} : r_1 < |z| < r_2\}$ *respectivamente , denotadas por F y G respectivamente, según (2.17). Entonces, se verifican las propiedades siguientes:*

◇***Linealidad:*** *Si* $R_1 = máx\{\rho_1, r_1\} < R_2 = máx\{\rho_2, r_2\}$ *entonces, para cada* $\alpha, \beta \in \mathbb{C}$*, la sucesión* $\{\alpha c_n + \beta s_n\}_{n=-\infty}^{\infty}$ *admite transformada Z en una corona que contiene a*

$$C_n \cap C_s = \{z \in \mathbb{C} : R_1 < |z| < R_2\},$$

y en esta corona su transformada Z es $\alpha F + \beta G$*.*

◇***Conjugación:*** *La sucesión dada por* $\{\overline{c_n}\}_{n=-\infty}^{\infty}$ *admite transformada Z, convergente en* C_a *y su transformada Z es* $\overline{F(\overline{z})}$.

◇***Derivación:*** *La sucesión* $\{nc_n\}_{n=-\infty}^{\infty}$ *admite transformada Z con corona de convergencia* C_a *y su transformada Z es* $-zF'(z)$.

◇***Convolución discreta:*** *Si* $R_1 = máx\{\rho_1, r_1\} < R_2 = máx\{\rho_2, r_2\}$ *para cada* $n \in \mathbb{N}$ *se tiene que para* $c:=\{c_n\}_{n=-\infty}^{\infty}$ *y* $s:=\{s_n\}_{n=-\infty}^{\infty}$ *la serie*

$$C_n := (c * s)(n) = \sum_{k=-\infty}^{\infty} c_k s_{n-k} \tag{2.18}$$

converge absolutamente. La sucesión

$$\{C_n\}_{n=-\infty}^{\infty}$$

se denomina convolución discreta de c y s. Esta nueva sucesión admite transformada Z en una corona que contiene a $C_n \cap C_s$ *y la transformada Z de* $\{C_n\}_{n=-\infty}^{\infty}$ *es la función* $F(z)G(Z)$.

A continuación se definirán los aproximantes de Padé, muy utilizados en la teoría de aproximación mediante funciones racionales, (véase e.g. [16]).

Definición 2.2. *Sea la serie de potencias*

$$f(z) = \sum_{i=0}^{\infty} c_i z^i, \quad c_i \in \mathbb{C}, \quad i = 1, \ldots, M,$$

que representa a la función f en un entorno de $z = 0$ *(es decir, f es holomorfa en* $z = 0$*), y cuya convergencia se da en una corona* $C = \{z \in \mathbb{C} : |z| < \rho\}$, *para cierto* $\rho > 0$, *y sean n, m*$\in \mathbb{N}$. *Se dice que*

$$\pi_{nm}(z) = \frac{P_{nm}(z)}{Q_{nm}(z)}, \quad P_{nm} \in \mathbb{C}_n[z], \quad Q_{nm} \in \mathbb{C}_m[z],$$

es el aproximante de Padé de la función f de grado (n,m), si verifica

$$Q_{nm}(z)f(z) - P_{nm}(z) = O(z^{n+m+1})^{2}, \qquad z \longrightarrow 0.$$

[2]Diremos que g(z) es $O(z^{n+m+1})$ si y solo si $\limsup_{z\to 0} |\frac{g(z)}{z^{n+m+1}}| < \infty$)

Veamos que el método clásico de Prony está relacionado con los aproximantes de Padé de la función f en (2.1).
Consideramos la sucesión $\{f(k)\}_{n=0}^{\infty}$ definida en (2.2), y sea F(z) su transformada Z, convergente en una corona $C = \{z \in \mathbb{C} : \rho_1 < |z| < \rho_2\}$, para ciertos $\rho_1, \rho_2 > 0$. Entonces, si $z \in C$, se tienen las siguientes igualdades[3]:

$$\begin{aligned} F(z) &= \sum_{k=0}^{\infty} f(k) z^{-k} = \sum_{k=0}^{\infty} (\sum_{i=1}^{M} c_i e^{\lambda_i k}) z^{-k} \\ &= \sum_{i=1}^{M} c_i \sum_{k=0}^{\infty} e^{\lambda_i k} z^{-k} = \sum_{i=1}^{M} c_i \frac{z}{z - z_i} = \frac{b(z)}{p(z)}, \end{aligned} \tag{2.19}$$

con p(z) en (2.5) y $b(z) = b_n z^n + \cdots + b_1 z + b_0 \in \mathbb{C}_n[z]$ el polinomio resultante de extraer p(z) como denominador común.
Escribiendo (2.19) como $b(z) = p(z)F(z)$ e igualando los coeficientes en potencias de z, se obtiene

$$\begin{aligned} b_{M-m} &= \sum_{k=M-m}^{M} a_k f(k + m - M), \quad m = 0, \ldots, M-1, \\ &\sum_{k=0}^{M} a_k f(k + m) = 0, \quad m \in \mathbb{N} \cup \{0\}. \end{aligned} \tag{2.20}$$

Y por tanto se observa que las ecuaciones dadas en (2.20) tomando $m = 0, \ldots, M-1$, coinciden con las ecuaciones en (2.7); entonces, el método clásico de Prony se puede ver como una aproximación de Padé.

2. Método de Prony y procesado de señales.

Uno de los principales usos del método de Prony es el dado en el análisis espectral de señales, ya que nos permite recuperar señales a partir de muestras finitas. Esta propiedad relaciona el método de Prony con el método del llamado filtro aniquilador (véase [13] sec.3.9, [8], [22]).

Necesitamos algunas definiciones previas:

[3]Hay que tener en cuenta que la transformada Z de $\{e^{\lambda_i k}\}_{k=1}^{\infty}$ es $\frac{z}{z - e^{\lambda_i}}$.

1 (**Señal discreta**). *En el procesamiento de señales se consideran señales discretas a las sucesiones* $S{:=}\{s_n\}_{n\in\mathbb{Z}}, s_n \in \mathbb{C}$.

2 (**Filtro discreto**). *Un filtro discreto se define como una función discreta que modifica determinadas amplitudes y/o frecuencias*[4] *de una señal, o directamente las elimina.*

Hay diversos tipos de filtros discretos, según el tipo de señales sobre las que actúan. En este apartado nos centraremos en los filtros discretos de respuesta finita, o filtros FIR (Finite Impulse Response, en inglés).

3 (**FIR**). *Denotando por* c_{00} *el conjunto de sucesiones complejas con un número finito de términos no nulos, diremos que una función discreta* F *es un FIR si tanto su espacio de salida como de llegada es* c_{00}. *Es decir, un FIR no es más que una función discreta* $F : c_{00} \to c_{00}$.

Además, dentro de los FIR nos centraremos en los filtros aniquiladores, que se definen de la siguiente manera:

4 (**Filtro aniquilador**). *Dadas dos señales* $S{:=}\{s_n\}_{n\in\mathbb{Z}}$ *y* $F{:=}\{f_n\}_{n\in\mathbb{Z}} \in c_{00}$, *diremos que* F *es el filtro aniquilador de* S *si su producto de convolución discreto (definido en (2.18)) es nulo, es decir, si*

$$(F * S)(n) = \sum_{k=-\infty}^{\infty} f_k s_{n-k} = 0.$$

En nuestro caso, para poder relacionar el método clásico de Prony con el procesado de señales nos interesan un tipo concreto de señales discretas $S := \{s_n\}_{n\in\mathbb{N}}$, que serán aquellas cuyos términos vienen dados por

$$s_n = \sum_{i=1}^{M} c_i z_i^n, \quad c_i \in \mathbb{C} \tag{2.21}$$

[4]Puesto que en la definición de señal discreta los s_n son números complejos, podemos expresarlos como $s_n = |s_n|e^{\imath\theta_n}$, consideramos $|s_n|$ la componente n-ésima de amplitud de la señal, y θ_n la frecuencia.

y los z_i como en (2.2).
Para construir F de forma explícita en el caso (2.21), consideramos

$$F(z) := \prod_{i=1}^{M}(1 - z_i z^{-1}) = \sum_{n=0}^{M} \tilde{f}_n z^{-n}, \qquad z \in \mathbb{C}\backslash\{0\}, \tag{2.22}$$

siendo $\tilde{f}_n$ los coeficientes de z^{-n} resultantes al evaluar el producto, en (2.22).

Definimos el filtro aniquilador como

$$F := \{f_n\}_{n\in\mathbb{Z}},\ f_n = \begin{cases} \tilde{f}_n & n = 0, \dots, M. \\ 0 & \text{otro caso.} \end{cases} \tag{2.23}$$

Proposición 2.1. *La sucesión F definida en (2.23) es filtro aniquilador de la señal S definida en (2.21).*

Demostración. Sea $n \in \mathbb{Z}$, veamos que (F*S)(n)=0. Para ello, por definición de producto de convolución y por (2.21) y (2.23), se tiene

$$\begin{aligned}
(F * S)(n) &= \sum_{k=-\infty}^{\infty} f_n s_{n-k} \\
&= f_0 s_n + f_1 s_{n-1} + \cdots + f_M s_{n-M} \\
&= f_0(\sum_{j=1}^{M} c_j z_j^n) + \cdots + f_m(\sum_{j=1}^{M} c_j z_j^{n-M}) \\
&= f_0 c_1 z_1^n + \cdots + f_0 c_M z_M^n + \\
&\quad f_1 c_1 z_1^{n-1} + \cdots + f_1 c_M z_M^{n-1} + \\
&\quad + \cdots + \\
&\quad f_M c_1 z_1^{n-M} + \cdots + f_M c_M z_M^{n-M} = \\
&= c_1 z_1^n (\sum_{k=0}^{M} f_k z_1^{-k}) + \cdots + c_M z_M^n (\sum_{k=1}^{M} f_k z_M^{-k}) \\
&= c_1 z_1^n F(z_1) + \cdots + c_m z_M^n F(z_M) = 0,
\end{aligned}$$

puesto que $F(z_i) = 0, i = 1, \dots, M$ por construcción en (2.22). □

Cabe destacar que F(z) es la transformada Z del filtro aniquilador F para S en (2.21).
Por construcción, los M ceros del polinomio F(z) construido en (2.22) son $z_1, \ldots, z_M$, es decir, F(z) tiene los mismos ceros que el polinomio de Prony (2.5). En consecuencia, se puede escribir $p(z) = z^M F(z), \forall z \in \mathbb{C}\backslash\{0\}$. Esto permite afirmar, identificando F(z) con p(z), que el método clásico de Prony equivale a un filtro aniquilador de la señal S en (2.21).

3. Método de Prony y predicción lineal.

Otro uso importante del método de Prony es su aplicación como modelo de predicción lineal. En diferentes campos científicos suele aparecer la necesidad de utilizar los datos medidos para predecir resultados que ocurrirán posteriormente. En estos casos el método de Prony se puede emplear de la siguiente manera para cubrir dicha necesidad:
Dada una señal discreta S:=$\{s_n\}_{n\in\mathbb{N}_0}$ como en (2.21), tratamos de encontrar parámetros predictivos adecuados $p_i \in \mathbb{C}$, de tal manera que para cierto $k \in \mathbb{N}_0$, el valor s_{k+M} se pueda expresar como combinación lineal de los M valores $s_k, \ldots, s_{k+M-1}$ anteriores, que asumiremos conocidos, en la forma

$$s_{k+M} = \sum_{i=0}^{M-1} -p_i s_{k+i}.$$

Esta representación es equivalente al sistema de ecuaciones en diferencias lineal homogéneo (2.8) dado por el método de Prony y, por tanto, podemos identificar el polinomio de Prony (2.5) con el polinomio predictivo, asi como la matriz compañera del mismo con la matriz predictiva,ambos empleados en el método de predicción lineal, (véase [10],[3]).

La forma clásica del método de Prony presenta algunas dificultades, como la presencia de ruido en las mediciones, la inestabilidad al encontrar las raices de p(z), el desconocimiento de M y el mal acondicionamiento de las matrices de Vandermonde. Todas ellas influyen en la precisión del método. Una forma de mejorar el rendimiento del método de Prony se basa en utilizar más datos en la muestra, bien equiespaciados o no. A cambio, la formulación requiere utilizar matrices rectangulares.

2.2. Métodos de tipo Prony para muestras equiespaciadas.

Los principales problemas que hacen necesaria la generalización del método clásico de Prony son el frecuente desconocimiento del orden M en la suma de exponenciales complejas (2.1) y la presencia de ruido al realizar las mediciones para obtener los datos. Es decir, para un cierto $N \in \mathbb{N}$ que supondremos $N \geq M$, los valores conocidos pueden expresarse como $f_k = f(k)+e_k$, $k = 0, \ldots, 2N-1$ con

$$f(k) = \sum_{j=1}^{M} c_j z_j^k, \qquad k = 0, \ldots, 2N-1, \tag{2.24}$$

y $z_j, j = 1, \ldots, M$, como en (2.2), y e_k errores de medición.

Sin pérdida de generalidad, supondremos que existe una cota superior adecuada $L \in \mathbb{N}$ tal que $M \leq L \leq N$, ya que en la mayoría de las aplicaciones dicha cota es un dato conocido a priori.
La extensión del método de Prony se llevará a cabo gracias a la estructura característica de las matrices de Hankel, definidas a partir de valores conocidos $f_k \in \mathbb{C}, k = 0, \ldots, 2N-1$, de la siguiente manera:

$$H_{2N-L,L+1} := \begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_L \\ f_1 & f_2 & f_3 & \cdots & f_{L+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{2N-L-2} & f_{2N-L-1} & f_{2N-L} & \cdots & f_{2N-2} \\ f_{2N-L-1} & f_{2N-L} & f_{2N-L+1} & \cdots & f_{2N-1} \end{bmatrix} \in \mathbb{C}_{(2N-L)\times(L+1)}. \tag{2.25}$$

También jugarán un papel importante las submatrices,

$$H_{2N-L,L}(s) := H_{2N-L,L+1}(1 : 2N-L, 1+s : L+s)^{5}, \qquad s = 0, 1. \tag{2.26}$$

Un resultado previo empleado en demostraciones posteriores, y que nos da una importante característica de las matrices de Hankel es el siguiente lema:

[5] Será frecuente el uso de esta notación, cuyo significado es el siguiente:
Dada una matriz $A \in \mathbb{C}_{m\times n}$, la submatriz, que denotaremos por $A(i : j, k : l)$, para ciertos $1 \leq i \leq j \leq m, 1 \leq k \leq l \leq n$, es la resultante de tomar las filas $i, i+1, \ldots, j$, y las columnas $k, k+1, \ldots, l$ de la matriz A.

Lema 2.2. *Para $z = (z_j)_{j=1}^M$ como en (2.2), se introduce la matriz de Vandermonde rectangular*

$$V_{P,Q}(z) = (z_j^{k-1})_{k,j=1}^{P,Q}, \qquad P, Q \geq 1.$$

Entonces si, $\vec{c} = (c_1, \ldots, c_M)^T$ viene dado por (2.24),

$$H_{2N-L,L+1} = V_{2N-L,M}(z)\ (diag(\vec{c}))\ V_{L+1,M}(z)^T, \tag{2.27}$$

$$H_{2N-L,L}(s) = V_{2N-L,M}(z)\ (diag(\vec{c}))\ (diag(z))^s\ V_{L,M}(z)^T,\ \ s = 0, 1.$$

Demostración. De nuevo, como ocurría en el caso de matrices de Vandermonde clásicas, desarrollando el lado derecho en (2.27),

$$\begin{bmatrix} 1 & \vdots & 1 \\ z_1 & \ldots & z_M \\ \vdots & \ddots & \vdots \\ z_1^{2N-L-1} & \ldots & z_M^{2N-L-1} \end{bmatrix} \begin{bmatrix} c_1 & \ldots & 0 \\ \vdots & \ddots & \\ 0 & \ldots & c_M \end{bmatrix} \begin{bmatrix} 1 & z_1 & \vdots & z_1^L \\ 1 & z_2 & \ldots & z_2^L \\ \vdots & \ddots & \vdots & \\ 1 & z_M & \ldots & z_M^L \end{bmatrix} =$$

$$= \begin{bmatrix} c_1 + \ldots + c_M & \ldots & c_1 z_1^L + \ldots + c_M z_M^L \\ \vdots & \ddots & \vdots \\ c_1 z_1^{2N-L-1} + \ldots + c_m z_M^{2N-L-1} & \ldots & c_1 z_1^{2N-1} + \ldots + c_M z_M^{2N-1} \end{bmatrix}.$$

Por lo tanto, igualando el elemento $(i, j), 1 \leq i \leq 2N - L, 1 \leq j \leq L + 1$, de la matriz resultante, con la matriz a la izquierda de la igualdad en (2.27) se obtiene

$$f(i + j - 2) = c_1 z_1^{i+j-2} + \cdots + c_M z_M^{i+j-2},$$

y esto es cierto, por definición de los $f(k), k = 0, \ldots, 2N - 1$ en (2.24).
Para $s = 0$, el resultado es inmediato, pues $(diag(z))^0 = Id_M$, y se obtiene el mismo resultado que el anterior, pero tomando unicamente las L primeras columnas.
Para s=1, la matriz a la derecha sería

$$\begin{bmatrix} 1 & \vdots & 1 \\ z_1 & \ldots & z_M \\ \vdots & \ddots & \vdots \\ z_1^{2N-L-1} & \ldots & z_M^{2N-L-1} \end{bmatrix} \begin{bmatrix} c_1 & \ldots & 0 \\ \vdots & \ddots & \\ 0 & \ldots & c_M \end{bmatrix} \begin{bmatrix} z_1 & z_1^2 & \vdots & z_1^L \\ z_2 & z_2^2 & \ldots & z_2^L \\ \vdots & \ddots & \vdots & \\ z_M & z_M^2 & \ldots & z_M^L \end{bmatrix} =$$

$$= \begin{bmatrix} c_1 z_1 + ... + c_M z_M & \dots & c_1 z_1^L + ... + c_M z_M^L \\ \vdots & \ddots & \vdots \\ c_1 z_1^{2N-L} + ... + c_m z_M^{2N-L} & \dots & c_1 z_1^{2N-1} + ... + c_M z_M^{2N-1} \end{bmatrix},$$

que se corresponde a las últimas L columnas de $H_{2N-L,L+1}$. □

A partir del resultado anterior, se demuestra otra propiedad importante de estas matrices, y es que en ausencia de ruido, tanto (2.25), como (2.26), tienen rango exactamente M, como se comprueba en el siguiente lema.

Lema 2.3. *Sean $N \in \mathbb{N}$ y $f_k = f(k), k = 0, \dots, 2N-1$ como en (2.24). Entonces las matrices (2.25) y (2.26) tienen rango exactamente M.*

Demostración. Como $M \leq L \leq N$,

$$rango(V_{2N-L,M}(z)) \leq \text{mín}\,\{2N - L, M\} = M$$

y como la submatriz $(z_j^{k-1})_{k,j=1}^{M}$ de $V_{2N-L,M}(z)$ es invertible, entonces

$$rango(V_{2N-L,M}(z)) = M.$$

El mismo razonamiento prueba que

$$rango(V_{L+1,M}(z)) = M.$$

Por otro lado, como $diag(\vec{c})$ es invertible, utilizando el lema previo

$$\begin{aligned} rango(H_{2N-L,L+1}) &= rango(V_{2N-L,M}(z)\ ((diag(\vec{c}))\ V_{L+1,M}(z)^T)) \\ &= rango(V_{2N-L,M}(z)) = M. \end{aligned}$$

En la segunda igualdad se ha utilizado la siguiente propiedad general de las matrices:

Si $A \in \mathbb{C}_{m\times n}, B \in \mathbb{C}_{n\times k}$, y B tiene rango n, entonces

$$rango(AB) = rango(A).$$

Las factorizaciones correspondientes del lema previo y un razonamiento similar prueban el resultado para las matrices (2.26).

□

Otro lema importante, que nos permitirá desarrollar los algoritmos de esta sección es el siguiente.

Lema 2.4. *Sean L,M,N∈ ℕ, tales que $M \leq L \leq N$, y f(k), $k = 0 \ldots, 2N-1$ como en (2.2). Entonces son equivalentes:*

(1) *El polinomio de coeficientes complejos,*

$$q(z) = z^L + \sum_{k=0}^{L-1} q_k z^k, \qquad z \in \mathbb{C}, \tag{2.28}$$

tiene como raices a los $z_j, j = 1, \ldots, M$, en (2.24). Este polinomio recibe el nombre de Polinomio de Prony modificado.

(2) *El vector $\vec{q} := (q_0, q_1, \ldots, q_{L-1})^T$ verifica*

$$H_{2N-L,L}(0)\ \vec{q} = -\begin{bmatrix} f(L) \\ \vdots \\ f(2N-1) \end{bmatrix}. \tag{2.29}$$

(3) *La matriz compañera (2.13) del polinomio q(z) satisface,*

$$H_{2N-L,L}(0)\ C_L(q)^T = H_{2N-L,L}(1). \tag{2.30}$$

Demostración.

(1)⇔(2) Supongamos que $q(z_j) = 0, j = 1, \ldots, M$.
Entonces, usando (2.24), para $m = 0, \ldots, 2N-L-1$,se tiene

$$\begin{aligned} \sum_{l=0}^{L-1} f(m+l) q_l &= \sum_{l=0}^{L-1} (\sum_{j=1}^{M} c_j z_j^{l+m}) q_l = \sum_{j=1}^{M} c_j z_j^m (\sum_{l=0}^{L-1} z_j^l q_l) \\ &= \sum_{j=1}^{M} c_j z_j^m (q(z_j) - z_j^L) = -\sum_{j=1}^{M} c_j z_j^{L+m} = -f(L+m), \end{aligned}$$

lo que demuestra (2.29). De nuevo utilizando (2.24), el razonamiento anterior es reversible, por lo que (2.29) implica que $q(z_j) = 0, j = 1, \ldots, M$.

(2)⇔(3) Suponiendo que se cumple (2.30), como la última columna de $C_L(q)^T$ es $-(q_0,\ldots,q_{L-1})^T$ y la última columna de $H_{2N-L,L}(1)$ es $(f(L),\ldots,f(2M-1))^T$, entonces se tiene (2.29). Inversamente, supongamos que se cumple (2.29). Entonces la última columna de las matrices (2.30) coincide. Por otra parte, para $j=1,\ldots,L-1$, el producto de $H_{2N-L,L}(0)$ por la columna j de $C_L(q)^T$ es el vector $(f(j),\ldots,f(2N-L-1+j))^T$, que es la columna j de $H_{2N-L,L}(1)$.

□

En el caso $L>M$, el polinomio modificado de Prony (2.28) no es único, ya que se puede tomar $q(z)=r(z)p(z)$, siendo $r(z)$ otro polinomio complejo cualquiera de grado $L-M$ mónico. Dicho polinomio tiene $L-M$ ceros adicionales, en comparacion con el clásico p. El más sencillo de todos es $q(z)=z^{L-M}p(z)$, que será el que consideremos de ahora en adelante.
Si $L=M$, el polinomio clásico y el modificado coinciden. El procedimiento es entonces una extensión del algoritmo clásico.

Puesto que los coeficientes $c_j, j=1,\ldots,M$, son no nulos, podemos suponer $|c_j|>\varepsilon$, para un cierto $0<\epsilon\ll 1$ adecuado.
Teniendo esto en cuenta, el algoritmo resultante tiene la estructura siguiente:

Algoritmo 2.2.1. Método de tipo Prony para muestras equiespaciadas.

Entradas: L,N$\in\mathbb{N}$, N$\gg$1, N$\geq$L$\geq$3, con L cota superior adecuada de M en (2.24), $f(k)$ valores medidos, $k=0,...,2N-1$ en (2.1), y $0<\varepsilon\ll 1$.

Paso 1) Hallar la solución por mínimos cuadrados del sistema

$$H_{2N-L,L}(0)\ \vec{q}=-\begin{bmatrix} f(L) \\ \vdots \\ f(2N-1)\end{bmatrix},$$

con $\vec{q}$ en (2.29).

Paso 2) Determinar las raíces simples $\tilde{z}_i, i=1,\ldots,\tilde{M}, \tilde{M}\leq L$, del polinomio de Prony modificado (2.28), o equivalentemente, hallar los autovalores $\tilde{z}_i, i=1,...,\tilde{M}$, de la matriz compañera $C_L(q)$. Nótese que

$rango(H_{2N-L,L}(0)) = M \leq \tilde{M}$, ya que los $z_i, i = 1, \ldots, M$, originales son distintos dos a dos.

Paso 3) Los coeficientes $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_{\tilde{M}})^T$ se obtienen a partir de la resolución por mínimos cuadrados del sistema sobredeterminado;

$$V_{2N,\tilde{M}}(\tilde{z}) \;\; \tilde{c}^T = \begin{bmatrix} f(0) \\ \vdots \\ f(2N-1) \end{bmatrix},$$

con $\tilde{z} := (\tilde{z}_1, \ldots, \tilde{z}_{\tilde{M}})$.

Paso 4) Eliminar aquellos $\tilde{z}_i, i = 1, \ldots, \tilde{M}$, que verifiquen $|\tilde{z}_i| \leq \varepsilon$. A los valores restantes se les denota por $z_1, \ldots, z_M$. Se obtienen los $\lambda_i, i = 1, \ldots, M$ aplicando la rama principal del logaritmo, ya que hemos supuesto $Im(\lambda_i) \in [-\pi, \pi)$, de manera que $\lambda_i := log(z_i), i = 1, ..., M$.

Paso 5) Repetir el paso 3, calculando los coeficientes $c_j \in \mathbb{C}, j = 1, \ldots, M$, como solución por mínimos cuadrados del sistema sobredeterminado

$$V_{2N,M}(\vec{z}) \;\; \vec{c}^T = \begin{bmatrix} f(0) \\ \vdots \\ f(2N-1) \end{bmatrix},$$

con $\vec{z} := (z_1, \ldots, z_M)$ y $\vec{c} := (c_1, \ldots, c_M)$.

Salidas: $M \in \mathbb{N}, c_i \in \mathbb{C}, \lambda_i \in [-\alpha, 0] + i[-\pi, \pi), \;\; i = 1, ..., M.$

■

A partir de la formulación dada por el algoritmo 2.2.1, se pueden construir diferentes métodos de tipo Prony basados en las matrices de Hankel. En esta memoria describiremos dos: el método de lapiz matricial (Matrix Pencil), basado en la factorización QR, y el método ESPRIT, que utiliza la descomposición en valores singulares de (2.25). Véase (sección 3, [7]) para un análisis más detallado.

El método de tipo Prony antes descrito puede formularse a partir del lápiz matricial (matrix pencil, véase apendice A)

$$zH_{2N-L,L}(0) - H_{2N-L,L}(1). \tag{2.31}$$

En general, un lápiz matricial rectangular puede no tener autovalores (véase el apéndice A para la descripción del problema de autovalores generalizado). Sin embargo, en el caso de (2.31) el problema puede reducirse al problema de autovalores clásico de una matriz cuadrada del modo siguiente: si $\vec{v} = (v_k)_{k=0}^{L-1} \in \mathbb{C}^L$ entonces, utilizando (2.30),

$$(zH_{2N-L,L}(0) - H_{2N-L,L}(1))\ \vec{v} = H_{2N-L,L}(0)(zId_L - C_L(q)^T)\ \vec{v},$$

y, por definición de matriz compañera

$$det(zId_L - C_L(q)^T) = q(z).$$

Entonces, por el lema 2.4, si $z = z_j, j = 1, \ldots, M$, se tiene que tomando $\vec{v}$ autovector del problema de autovalores para $C_L(q)$ ó $C_L(q)^T$, z_j es autovalor de (2.31), lo que relaciona el problema generalizado de autovalores de (2.31) con el problema de autovalores para la matriz cuadrada $C_L(q)$.

Método Lápiz matricial QR para muestras equiespaciadas.

En primer lugar, puesto que la matriz $H_{2N-L,L+1}$ es de rango deficiente, por lo visto en el lema 2.3, se aplicará la descomposición QR con pivotaje por columas a dicha matriz,

$$H_{2N-L,L+1}P_{L+1} = Q_{2N-L}R_{2N-L,L+1}, \tag{2.32}$$

con $Q_{2N-L} \in \mathbb{C}_{(2N-L)\times(2N-L)}$ unitaria,

$$R_{2N-L,L+1} = \begin{bmatrix} r_{1,1} & r_{1,2} & \ldots & r_{1,M} & \ldots & r_{2,L+1} \\ 0 & r_{2,2} & \ldots & r_{2,M} & \ldots & r_{2,L+1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & r_{M,M} & \ldots & r_{M,L+1} \\ 0 & 0 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{C}_{(2N-L)\times(L+1)},$$

$r_{i,j} \in \mathbb{C}, i = 1, \ldots, 2N-L, j = 1, \ldots, L+1.$

$P_{L+1} \in \mathbb{C}_{(L+1)\times(L+1)}$ una matriz de permutación, tomada de tal manera, que $|r_{i,i}| \geq |r_{j,j}|, 1 \leq i \leq j \leq M$.

En el caso particular de mediciones exactas, mediante la factorización QR, se obtiene el orden M en (2.24) como $rango(R_{2N-L,L+1})$.

Para muestras con ruido, el rango de la matriz $R_{2N-L,L+1}$ no es necesariamente M, por lo que se aproxima, como el menor entero tal que $|R_{2N-L,L+1}(M+1, M+1)| < \varepsilon |R_{2N-L,L+1}(1,1)|$, para un cierto $\varepsilon > 0$, tomado en función del ruido. Se define en primer lugar la matriz

$$D_M = diag(r_{1,1}, \ldots, r_{M,M}).$$

Puesto que P_{L+1} es una matriz de permutación,

$$H_{2N-L,L+1} = Q_{2N-L} R_{2N-L,L+1} P_{L+1}^T = Q_{2N-L} S_{2N-L,L+1}, \tag{2.33}$$

con

$$S_{2N-L,L+1} := R_{2N-L,L+1} P_{L+1}^T = \begin{bmatrix} S_{2N-L,L+1}(1:M, 1:L+1) \\ 0_{2N-L-M,L+1}. \end{bmatrix}.$$

Entonces, para $s = 0, 1$,

$$H_{2N-L,L}(s) = Q_{2N-L} S_{2N-L,L}(s), \qquad s = 0, 1,$$

con $S_{2N-L,L}(s) := S_{2N-L,L+1}(1 : 2N-L, 1+s : L+s)$, ya que $H_{2N-L,L}(s), s = 0, 1$, estaba formada por las primeras L columnas de $H_{2N-L,L+1}$ ó las L últimas, respectivamente.
Como Q_{2N-L} es unitaria, el problema de autovalores generalizado del lápiz (2.31) es equivalente a

$$zS_{2N-L,L}(0) - S_{2N-L,L}(1).$$

Usando (2.33), puede simplificarse al lápiz

$$zT_{M,L}(0) - T_{M,L}(1), \tag{2.34}$$

con $T_{M,L}(s) = S_{2N-L,L+1}(1 : M, 1+s : L+s), s = 0, 1$.
El problema de autovalores generalizado para

$$zT_{M,L}(0)^T - T_{M,L}(1)^T,$$

tiene los mismos autovalores $z_j, j = 1, \ldots, M$ que (2.34), a excepción de los autovalores nulos, ya que ambas matrices son de rango completo M. Si v_j es el autovalor asociado a z_j

$$z_j T_{M,L}(0)^T \; v_j = T_{M,L}(1)^T \; v_j, \tag{2.35}$$

resuelve el sistema lineal (2.35) con $T_{M,L}^T$ de rango completo.
De donde se deduce que, para $A = T_{M,L}(0)^T$

$$z_j A^T A v_j = A^T T_{M,L}(1)^T v_j,$$

es decir,

$$z_j \; v_j = (A^T A)^{-1} A^T T_{M,L}(1)^T v_j.$$

Por lo que, si $A^+ = (A^T A)^{-1} A^T$, es la pseudoinversa de A, se concluye que v_j es autovector de

$$F_M^{QR} := (T_{M,L}(0)^T)^+ T_{M,L}(1)^T,$$

con autovalor asociado z_j.
De nuevo, debido a este desarrollo, se puede formular el algoritmo de tipo Prony de factorización de lápices matriciales mediante descomposición QR para muestras equiespaciadas de la siguiente manera:

Algoritmo 2.2.2. Factorización de lápices matriciales mediante descomposición QR.

Entradas: L,N$\in \mathbb{N}$, N$\gg$1, N$\geq$L$\geq$3, con L cota superior adecuada de M en (2.24), f_k valores medidos (con posibilidad de ruido) $, k = 0, ..., 2N-1$ en (2.1), y $0 < \varepsilon \ll 1$.

Paso 1) Hallar la factorización QR de la matriz $H_{2N-L,L+1}$ en (2.25). Determinar el rango numérico M en (2.32) como el menor entero tal que $|R_{2N-L,L+1}(M+1, M+1)| < \varepsilon |R_{2N-L,L+1}(1,1)|$, y construir las matrices $T_{M,L}(s), s = 0, 1$, en (2.34).

Paso 2) Hallar los autovalores $z_i = e^{\lambda_i}, i = 1, ..., M$, de la matriz $F_M^{QR} := (T_{M,L}(0)^T)^+ T_{M,L}(1)^T$. A partir de estos, calcular los λ_i aplicando la rama principal del logaritmo, ya que hemos supuesto $Im(\lambda_i) \in [-\pi, \pi)$, de manera que $\lambda_i := log(z_i), \quad i = 1, ..., M$.

Paso 3) Los coeficientes $\vec{c} = (c_1, \ldots, c_M)^T$ se obtienen a partir de la resolución por mínimos cuadrados del sistema sobredeterminado;

$$V_{2N,M}(\vec{z}) \; \vec{c}^T = \begin{bmatrix} f_0 \\ \vdots \\ f_{2N-1} \end{bmatrix},$$

con $\vec{z} := (z_1, \ldots, z_M)$, y $V_{2N-L,M}(z)$, la matriz rectangular de Vandermonde.

Salidas: $M \in \mathbb{N}, c_i \in \mathbb{C}, \lambda_i \in [-\alpha, 0] + i[-\pi, \pi),\ i = 1, ..., M.$

■

Método ESPRIT para muestras equiespaciadas.

El desarrollo de este método, es similar al anterior, pero empleando la descomposición en valores singulares (SVD, e.g. sección 3, [7]) de la matriz Hankel rectangular construida en (2.25).

De nuevo, por ser $H_{2N-L,L+1}$ de rango M, se aplica la descomposición SVD[6],

$$H_{2N-L,L+1} = U_{2N-L}\ \Sigma_{2N-L,L+1}\ W_{L+1}, \tag{2.36}$$

con $U_{2N-L} \in \mathbb{C}_{(2N-L)\times(2N-L)}, W_{L+1} \in \mathbb{C}_{(L+1)\times(L+1)}$, matrices unitarias, y

$$\Sigma_{2N-L,L+1} := \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_{L+1} \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{C}_{(2N-L)\times(L+1)},$$

con $\sigma_1, \ldots, \sigma_{L+1}$ los valores singulares de $H_{2N-L,L+1}$ ordenados de forma decreciente.
Si nos restringimos al caso de mediciones exactas, por el lema 2.2,

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_M \geq \sigma_{M+1} = \cdots = \sigma_{L+1} = 0,$$

y por tanto el orden de la suma exponencial (2.1) es exactamente $\mathrm{rg}(H_{2N-L,L+1})$.
Consideramos las submatrices siguientes,

$$\Sigma_{2N-L,M} := \Sigma_{2N-L,L+1}(1 : 2N - L, 1 : M)$$

[6]Generalmente la descomposición SVD en (2.36) suele darse en la forma $H_{2N-L,L+1} = U_{2N-L}\ \Sigma_{2N-L,L+1}\ V^*_{L+1}$, para cierta matriz unitaria $V_{L+1} \in \mathbb{C}_{(L+1)\times(L+1)}$, en nuestro caso simplemente se considera $W_{L+1} := V^*_{L+1}$, donde se denota por A^* la transpuesta conjugada de A

$$W_{M,L+1} := W_{L+1}(1:M, 1:L+1)$$

La descomposición (2.36) de la matriz se podrá simplificar en la forma

$$H_{2N-L,L+1} = U_{2N-L}\Sigma_{2N-L,M}W_{M,L+1},$$

pues $\sigma_{M+1} =, \ldots, = \sigma_{L+1} = 0$. Definimos asimismo las submatrices

$$W_{M,L}(s) := W_{M,L+1}(1:M, 1+s:L+s)\ , s = 0, 1. \tag{2.37}$$

Utilizando (2.26) y (2.36), se tiene

$$H_{2N-L,L}(s) = U_{2N-L}D_{2N-L,M}W_{M,L}(s),\ \ s = 0, 1. \tag{2.38}$$

Como U_{2N-L} unitaria, de (2.38) se deduce que el problema de autovalores generalizado (2.31) es equivalente al problema de autovalores generalizado para

$$z\Sigma_{2N-L,M}W_{M,L}(0) - \Sigma_{2N-L,M}W_{M,L}(1). \tag{2.39}$$

Multiplicando por la derecha el lápiz matricial(2.39) transpuesto, por

$$\begin{bmatrix} \sigma_1^{-1} & 0 & \ldots & 0 \\ 0 & \sigma_2^{-1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_M^{-1} \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix},$$

obtenemos el lápiz

$$zW_{M,L}(0) - W_{M,L}(1),$$

que, por tanto, contiene a los z_j como autovalores. Como en el método anterior, los nodos pueden obtenerse como autovalores de la matriz

$$F_M^{SVD} := (W_{M,L}(0)^T)^+ W_{M,L}(1)^T.$$

Para el caso de muestras con ruido, los valores singulares de la matriz $H_{2N-L,L+1}$ no tienen por qué anularse a parir del término M-ésimo, es decir,

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_M \geq \sigma_{M+1} \geq \cdots \geq \sigma_{L+1} \geq 0,$$

y será necesario aproximar M, como el rango numérico de dicha matriz, [7], de la siguiente manera:

$$M := \min_{m\in\mathbb{N}}\{m : \sigma_{m+1} < \varepsilon\sigma_1\}, \tag{2.40}$$

para cierto $\varepsilon > 0$, que vendrá dado por la tolerancia del algoritmo, y que dependerá del ruido.

De forma análoga, el algoritmo en este caso se describe como:

Algoritmo 2.2.3. Método ESPRIT para muestras equiespaciadas.

Entradas: L,N$\in \mathbb{N}$, N$\gg$1, N$\geq$L$\geq$3, con L cota superior adecuada de M en (2.24), f_k valores medidos (con posibilidad de ruido) $, k = 0, ..., 2N-1$ en (2.1), y $0 < \varepsilon \ll 1$.

Paso 1) Hallar la factorización SVD de la matriz $H_{2N-L,L+1}$ en (2.25). Determinar el rango numérico M en (2.40) como el menor entero M, tal que $\sigma_{M+1} < \varepsilon\sigma_1$, y construir las matrices $W_{M,L}(s), s = 0, 1$, en (2.37).

Paso 2) Hallar los autovalores $z_i = e^{\lambda_i}, i = 1, ..., M$, de la matriz $F_M^{SVD} := (W_{M,L}(0)^T)^+ W_{M,L}(1)^T$. A partir de estos, calcular los λ_i aplicando la rama principal del logaritmo, ya que hemos supuesto $Im(\lambda_i) \in [-\pi, \pi)$, de manera que $\lambda_i := log(z_i), \quad i = 1, ..., M$.

Paso 3) Los coeficientes $\vec{c} = (c_1, \ldots, c_M)^T$ se obtienen a partir de la resolución por mínimos cuadrados del sistema sobredeterminado;

$$V_{2N,M}(\vec{z})\ \vec{c}^T = \begin{bmatrix} f_0 \\ \vdots \\ f_{2N-1} \end{bmatrix},$$

con $\vec{z} := (z_1, \ldots, z_M)$, y $V_{2N-L,M}(z)$, la matriz rectangular de Vandermonde.

Salidas: $M \in \mathbb{N}, c_i \in \mathbb{C}, \lambda_i \in [-\alpha, 0] + i[-\pi, \pi),\ i = 1, ..., M.$

■

El método de tipo Prony visto en esta sección, puede plantearse como solución del problema de reducción basado en la aproximación por matrices de Hankel de menor rango (e.g. [12]). Este tipo de problemas se describen de la siguiente manera:
Dada una aplicación $\mathcal{S} : \mathbb{C}^K \to \mathbb{C}^{L\times N}$, con $L < N$, denominada especificación de estructura, un vector $h \in \mathbb{C}^K$, una norma $||\ .\ ||$ en $\mathbb{C}^K$, y un entero M tal que $0 < M < L$. Se pretende encontrar otro vector que verifique la siguiente condición,

$$\hat{h}^* := \arg \min_{\hat{h}\in\mathbb{C}^K} ||h - \hat{h}||, \quad \textit{con } \hat{h} \textit{ verificando } rg(\mathcal{S}(\hat{h})) \leq M.$$

En nuestro caso, podemos considerar $\mathcal{S}$ la estructura que construye las matrices de Hankel, es decir $\mathcal{S}(h)$ es (2.25). Como ya vimos, podemos afirmar que $\mathcal{S}(\hat{h})$ es una matriz de rango M, si existe un vector $\vec{p} := (p_0, \ldots, p_{M-1}) \in \mathbb{C}^M$ tal que

$$\sum_{k=0}^{M-1} p_k h_{m+k} = -h_{M+m}, \quad m = 0, \ldots, N + L - M - 1.$$

Es decir, $\vec{p}\mathcal{S}(\hat{h}) = 0$, luego $\vec{p} \in \ker \mathcal{S}(\hat{h})$. Y por definición, la estructura de $ker\mathcal{S}(\hat{h})$, vista en la demostración del lema 2.2, hace que el método de Prony se pueda interpretar como un problema de aproximación por matrices de Hankel de menor rango.

2.3. Métodos de tipo Prony para muestras no equiespaciadas.

El siguiente paso en la expansión de los algoritmos de tipo Prony será considerar muestras no equiespaciadas. Es decir, en este caso el problema será el siguiente:

Dado un cierto $N \in \mathbb{N}$, con $N \geq M$, queremos recuperar los valores de una suma finita de exponenciales compleja, como ocurria en (2.24), sin embargo, ahora supondremos que los valores conocidos son de la forma $f(x_k), k = 0, \ldots, 2N - 1$, para ciertos nodos no equiespaciados

$$0 \leq x_0 < x_1 < \ldots < x_{2N-1} = 2L - 1, \qquad M \leq L \leq N, \tag{2.41}$$

y cierta cota L conocida a priori, como en casos anteriores. Suponiendo que los errores en las mediciones son lo suficientemente pequeños, la base de este desarrollo consisitrá en hacer un paso previo en el que, mediante diferentes técnicas de interpolación, se logre obtener aproximaciones de los valores $f(0), \ldots, f(2L-1)$,en (2.24), para posteriormente aplicar alguno de los algoritmos vistos con anterioridad. Este paso previo recibe el nombre de pre-procesado de los datos, y se presenta en esta memoria empleando dos técnicas diferentes:

2.3.1. Pre-procesado mediante B-splines cardinales.

Para desarrollar este método, será necesaria la introducción de los B-splines cardinales.

Definición 2.3. *Dado un $m \in \mathbb{N}_0$, el B-spline cardinal de grado 2m es*

$$M_{2m}(x) := \sum_{k=0}^{2m} (-1)^{2m-k} \binom{2m}{k} \frac{(k-x-m)_+^{2m-1}}{(k-1!)},$$

con $(k-x-m)_+^{2m-1} := max\{0, k-x-m\}$.

En estas condiciones, el spline M_{2m} es simétrico respecto al eje de ordenadas, y su soporte es el intervalo $[-m, m]$. Emplemos dicho B-spline para aproximar los valores $f(k), k = 0, \ldots, 2N-L$, ya que M_{2m} y sus traslaciones, forman una base del espacio de polinomios a trozos de orden $2m$ (e.g. [5]). Para ello consideramos la función

$$g(x) := \sum_{l=1-m}^{2L+m-2} g_l M_{2m}(x-l).$$

Para que dicha función sea el aproximante buscado, imponemos las condiciones

$$g(x_k) = f(x_k), \qquad k = 0, \ldots, 2N-1.$$

Resolviendo mediante mínimos cuadrados el sistema generado por estas condiciones, se obtienen los coeficientes g_l. Para ello, es necesario suponer que la matriz $(M_{2m}(x_k - l))_{k=0,l=-m+1}^{2N-1,2L+m-2}$ tiene rango máximo $2L + 2m - 2$. Y se concluye tomando $f_k := g(k), k = 0, \ldots, 2L-1$.

2.3.2. Pre-procesado mediante interpolación por polinomios cúbicos.

Puesto que queremos hallar el valor de $f(k), k = 0, \ldots, 2N-1$, en (2.24) mediante interpolación cúbica, definiremos los polinomios $p_j, j = 1, \ldots, 2N-3$, cada uno de ellos unívocamente definido por las condiciones

$$p_j(x_k) = f(x_k), \qquad k = j-1, j, j+1, j+2, \tag{2.42}$$

para los valores $f(x_k), k = 0, \ldots, 2N-1$, conocidos. Estos polinomios pueden definirse de diferentes maneras, en función de la base y el método utilizado (véase [21]). En nuestro caso, cada p_j se construirá empleando el método de coeficientes indeterminados, considerando la base

$$\{1, (x-x_j), (x-x_j)^2, (x-x_j)^3\}.$$

Por lo tanto, cada polinomio será

$$p_j(x) = f(x_j) + \alpha_j(x-x_j) + \beta_j(x-x_j)^2 + \gamma_j(x-x_j)^3,$$

con $\alpha_j, \beta_j, \gamma_j \in \mathbb{C}$ los coeficientes indeterminados. Puesto que imponemos la concidión (2.42), ha de verificarse

$$f(x_k) = f(x_j)+\alpha_j(x_k-x_j)+\beta_j(x_k-x_j)^2+\gamma_j(x_k-x_j)^3,\ k = j-1, j+1, j+2.$$

A partir de estas condiciones, resolviendo el sistema lineal

$$\begin{bmatrix} 1 & x_{j-1}-x_j & (x_{j-1}-x_j)^2 \\ 1 & x_{j+1}-x_j & (x_{j+1}-x_j)^2 \\ 1 & x_{j+2}-x_j & (x_{j+2}-x_j)^2 \end{bmatrix} \begin{bmatrix} \alpha_j \\ \beta_j \\ \gamma_j \end{bmatrix} = \begin{bmatrix} \frac{f(x_{j-1})-f(x_j)}{x_{j-1}-x_j} \\ \frac{f(x_{j+1})-f(x_j)}{x_{j+1}-x_j} \\ \frac{f(x_{j+2})-f(x_j)}{x_{j+2}-x_j} \end{bmatrix},$$

se obtienen los coeficientes necesarios para definir los p_j.
Por último, definimos las aproximaciones buscadas de la siguiente manera:

$$f_k := \begin{cases} p_1(k) & si\ 0 \le k < x_2, \\ p_j(k) & si\ x_j \le k < x_{j+1}, j = 2, \ldots, 2N-4, \\ p_{2N-3}(k) & si\ x_{2N-3} \le k < x_{2N-1}(= 2L-1), \end{cases}$$

para $k = 0, \ldots, 2L-1$.
Nótese que algún p_j puede no utilizarse al calcular las aproximaciones, en

función de como esten dispuestos los nodos. Por ello, para casos en los que $N \gg L$ puede ser interesante el empleo de otro tipo de técnicas de forma local, en las que se utilicen más datos.
Una vez definidos los dos métodos posibles para realizar el pre-procesado de los datos, el algoritmo de tipo Prony a emplear en el caso no equiespaciado se describe como sigue:

Algoritmo 2.3.3. Métodos tipo Prony para muestras no equiespaciadas.

Entradas: L,N$\in \mathbb{N}$, N$\gg$1, N$\geq$L$\geq$3, con L cota superior adecuada de M en (2.24), f_k, nodos no equiespaciados $x_k, k = 0, \ldots, 2N-1$, en (2.41), valores medidos $f(x_k), k = 0, ..., 2N-1$ en (2.24), y $0 < \varepsilon \ll 1$.

Paso 1) Realizar el paso de pre-procesado, mediante B-splines cardinales o interpolación polinomica, empleando las técnicas anteriores, para obtener los valores $f_0, \ldots, f_{2N-1}$.

Paso 2) Emplear alguno de los algoritmos vistos en la sección 2.2, tomando $N := L$, para obtener el orden M de la suma exponencial, los valores λ_i y los coeficientes c_i en (2.24).

Salidas: $M \in \mathbb{N}, c_i \in \mathbb{C}, \lambda_i \in [-\alpha, 0] + i[-\pi, \pi),\ i = 1, ..., M.$

■

2.4. Experimentos numéricos.

Para finalizar este capítulo, pondremos en práctica la teoría desarrollada mediante una serie de ejemplos numéricos, en los cuales se tendrán en cuenta los siguientes errores relativos para medir la calidad de los procedimientos:

Exponentes :

$$e(\vec{\lambda}) := \frac{\text{máx}_{j=1,...,M} |\lambda_j - \tilde{\lambda}_j|}{\text{máx}_{j=1,...,M} |\lambda_j|},$$

Coeficientes :

$$e(\vec{c}) := \frac{\text{máx}_{j=1,\dots,M} |c_j - \tilde{c}_j|}{\text{máx}_{j=1,\dots,M} |c_j|},$$

Evaluaciones :

$$e(f) := \frac{\text{máx}_{x_j} |f(x_j) - \tilde{f}(x_j)|}{|\,\text{máx}\, f(x_j)|},$$

donde se tiene en cuenta que

$$f(x) = \sum_{j=1}^{M} c_j e^{\lambda_j x},$$

con $M \in \mathbb{N}, c_j, \lambda_j \in \mathbb{C}, j = 1, \dots, M$ como en (2.24), es la función, cuyos parámetros se pretende recuperar, y $N \in \mathbb{N}$ una cota superior adecuada de M. Por otro lado,

$$\tilde{f}(x) = \sum_{j=1}^{\tilde{M}} \tilde{c}_j e^{\tilde{\lambda}_j x},$$

con $\tilde{M} \in \mathbb{N}, \tilde{c}_j, \tilde{\lambda}_j \in \mathbb{C}, j = 1 \dots, \tilde{M}$, son los parámetros aproximados obtenidos mediante los diferentes algoritmos. En el cálculo de $e(f)$, se seleccionan los $x_j = \frac{2N-1}{10(2N-1)} j = \frac{1}{10} j, j = 0, \dots, 10(2N-1)$, que forman una partición del intervalo $[0, 2N-1]$ en $10(2N-1)+1$ nodos. Otro concepto interesante en el desarrollo de los experimentos es la distancia de separación definida por

$$\delta := \text{mín}\{|\lambda_j - \lambda_k|, k \neq j\}, \qquad \lambda_j, \lambda_k \text{ como en (2.24)},$$

ya que podemos recuperar todos los parámetros en (2.24) con $2N$ mediciones, para un $N \in \mathbb{N}$, tal que $N > \frac{\pi^2}{\delta\sqrt{3}}$, véase e.g. [20].

Para llevar a cabo el desarrollo de los ejemplos, se han programado en Matlab los diferentes algoritmos vistos, y pueden consultarse en el Apéndice B.

N	L	$e(\vec{\lambda})$	$e(\vec{c})$	$e(f)$
Método de Prony (Algoritmo 2.2.1)				
6	6	$2{,}4368e-09$	$1{,}4917e-09$	$4{,}4199e-14$
7	6	$6{,}1905e-10$	$4{,}1281e-10$	$8{,}8824e-15$
7	7	$4{,}0624e-10$	$2{,}6675e-10$	$3{,}2235e-14$
Lápices matriciales QR (Algoritmo 2.2.2)				
6	6	$2{,}1917e-09$	$1{,}3476e-09$	$8{,}2014e-14$
7	6	$6{,}3899e-10$	$4{,}2640e-10$	$1{,}2644e-13$
7	7	$3{,}8474e-10$	$2{,}5202e-10$	$2{,}3080e-14$
Lápices matriciales QR (con error de medición)				
10	10	$7{,}0533e-06$	$1{,}0941e-05$	$5{,}2630e-06$
20	10	$4{,}3621e-06$	$9{,}5054e-06$	$4{,}9406e-06$
Método ESPRIT (Algoritmo 2.2.3)				
6	6	$1{,}2338e-09$	$7{,}4307e-10$	$1{,}7722e-14$
7	6	$7{,}0445e-10$	$4{,}9754e-10$	$6{,}2747e-14$
7	7	$3{,}3988e-10$	$2{,}1960e-10$	$5{,}1311e-14$
Método ESPRIT (con error de medición)				
10	10	$7{,}0533e-06$	$1{,}0941e-05$	$5{,}2630e-06$
20	10	$4{,}3621e-06$	$9{,}5054e-06$	$4{,}9406e-06$

Tabla 2.1: Errores obtenidos en el experimento 1.

2.4.1. Experimento 1.

En el desarrollo de este experimento, se consideran como parámetros constituyentes de f en (2.24), los siguientes: $M = 6, c_j = j, j = 1, \ldots, M$, y

$$\begin{bmatrix} z_1 \\ \vdots \\ z_M \end{bmatrix} := \begin{bmatrix} 0{,}9856 - 0{,}1628i \\ 0{,}9856 + 0{,}1628i \\ 0{,}8976 - 0{,}4305i \\ 0{,}8976 + 0{,}4305i \\ 0{,}8127 - 0{,}5690i \\ 0{,}8127 + 0{,}5690i \end{bmatrix},$$

con $z_j := e^{\lambda_j}, j = 1, \ldots, M$. Se ponen a prueba los algoritmos clásicos en sus versiones QR y ESPRIT, tomando diversos valores para N y L, $\varepsilon = 10^{-10}$. A partir de estos datos, los errores obtenidos para diferentes valores de

$N,\ L \in \mathbb{N}, M \leq L \leq N$, se representan en Tabla 2.1. Las aproximaciones obtenidas, se calculan a partir de las muestras $f(k), k = 0, \ldots, 2N-1$. Como se puede ver, a mayor cantidad de datos las aproximaciones son mejores, ya que los errores van disminuyendo, aun así para una cantidad pequeña de datos, se obtienen aproximaciones muy buenas, dando sentido a toda la teoría desarrollada hasta el momento. Cuando hay errores en las mediciones originales, la toma del ε varía, ya que al realizar el estudio de los valores obtenidos, se puede ver un claro salto en los coeficientes obtenidos, que nos permitirán escoger el ε adecuado. En este caso, si consideramos ruido en el muestreo, los valores medidos vienen dados por $f_k = f(k) + e_k, k = 0, \ldots, 2N-1$, siendo $e_k := 10^{-4}(-1 + \frac{k}{2N-1})$, y el ε adecuado será 10^{-7}.

2.4.2. Experimento 2.

Para el experimento 2, se trata de recuperar la función en (2.24) de parámetros:

- $M = 6$.
- $(\lambda_1, \ldots, \lambda_6) = \frac{i}{1000}(7, 21, 200, 201, 53, 1000)$.
- $(c_1, \ldots, c_6) = (6, 5, 4, 3, 2, 1)$.
- $\varepsilon = 10^{-10}$ (Tanto para las mediciones con y sin ruido).
- $f_k = f(k) + e_k, k = 0, \ldots, 2N-1$, siendo $e_k := 10^{-4}(-1 + \frac{k}{2N-1})$ (Casos con ruido).

De nuevo se aplica el método clásico, el método QR y el ESPRIT, obteniendo los resultados expuestos en la Tabla 2.2. En este caso, por como se han tomado los λ_j originales, la distancia de separación es $\delta = |\frac{i}{1000}(200-201)| = 0{,}001$. A partir de este valor deducimos que todos los parámetros se pueden recuperar para un $N \in \mathbb{N}$ tal que $N > \frac{\pi^2}{\delta\sqrt{3}} > 5698$. Sin embargo, con cantidades mucho más pequeñas de datos se obtienen aproximaciones muy buenas. Esto es interesante, ya que a pesar del mal acondicionamiento de las matrices participantes en los métodos (números de condición en Tabla 2.3), se siguen obteniendo buenos resultados. Como vemos en Tabla 2.2, en los casos con ruido, para la tolerancia 10^{-10} no siempre somos capaces de recuperar todos los parámetros, sin embargo se obtienen buenas aproximaciones en las evaluaciones.

N	L	$e(\vec{\lambda})$	$e(\vec{c})$	$e(f)$
Método de Prony (Algoritmo 2.2.1)				
7	7	$4{,}1442e-03$	$5{,}1742e-01$	$3{,}5532e-14$
8	8	$2{,}8473e-04$	$3{,}1831e-01$	$2{,}3965e-14$
9	9	$4{,}1236e-05$	$4{,}0061e-02$	$2{,}0744e-14$
10	10	$1{,}3736e-05$	$1{,}4174e-02$	$4{,}7416e-14$
15	15	$6{,}8985e-08$	$7{,}0123e-05$	$1{,}2851e-14$
20	20	$2{,}0164e-09$	$2{,}0522e-06$	$1{,}3073e-14$
30	30	$7{,}6718e-11$	$7{,}7345e-08$	$3{,}7732e-14$
30	20	$1{,}4262e-10$	$1{,}4530e-07$	$1{,}0301e-13$
30	10	$3{,}5921e-09$	$3{,}6664e-06$	$6{,}1141e-12$
Lápices matriciales QR (Algoritmo 2.2.2)				
7	7	$4{,}2294e-03$	$5{,}1650e-01$	$6{,}1203e-14$
8	8	$1{,}8760e-04$	$2{,}0583e-01$	$1{,}4190e-14$
9	9	$1{,}7559e-05$	$1{,}7509e-02$	$2{,}8014e-14$
10	10	$5{,}4689e-06$	$5{,}5860e-03$	$2{,}4980e-14$
15	15	$5{,}1358e-08$	$5{,}2185e-05$	$2{,}1696e-14$
20	20	$1{,}6616e-09$	$1{,}6882e-06$	$4{,}0859e-15$
30	30	$5{,}3651e-11$	$5{,}4306e-08$	$1{,}6024e-14$
30	20	$1{,}6697e-10$	$1{,}6942e-07$	$3{,}8400e-14$
30	10	$1{,}5142e-09$	$1{,}5426e-06$	$2{,}6680e-12$
Método QR (con error de medición)				
10	10	$*$	$*$	$5{,}2631e-06$
20	10	$1{,}7318e-05$	$1{,}7391e-02$	$5{,}0061e-06$
Método ESPRIT (Algoritmo 2.2.3)				
7	7	$4{,}6278e-03$	$5{,}1355e-01$	$1{,}2102e-13$
8	8	$1{,}9269e-04$	$2{,}1979e-01$	$7{,}7480e-15$
9	9	$1{,}6093e-05$	$1{,}6088e-02$	$1{,}7555e-14$
10	10	$8{,}9538e-06$	$9{,}1634e-03$	$9{,}5221e-14$
15	15	$5{,}8037e-08$	$5{,}9044e-05$	$7{,}9141e-14$
20	20	$9{,}9186e-10$	$1{,}0050e-06$	$3{,}9761e-14$
30	30	$6{,}4467e-11$	$6{,}6407e-08$	$8{,}7788e-14$
30	20	$1{,}6225e-10$	$1{,}7018e-07$	$1{,}6721e-13$
30	10	$2{,}5156e-09$	$2{,}1401e-06$	$1{,}4247e-11$
Método ESPRIT (con error de medición)				
6	6	$*$	$*$	$5{,}2631e-06$
6	6	$1{,}7280e-05$	$1{,}7354e-02$	$5{,}0061e-06$

Tabla 2.2: Errores obtenidos en el experimento 2.

N	L	$H^*_{2N-L,L}(0)H_{2N-L,L}(0)$	$V^*_{2N,M}V_{2N,M}(z)$	F_M^{QR}	F_M^{SVD}
7	6	$1{,}5373e+17$	$1{,}6490e+09$	82,1903	194,0979
10	6	$5{,}2884e+16$	$4{,}0244e+08$	7,8700	194,1090
10	7	$5{,}0712e+16$	$4{,}0456e+08$	10,0336	82,0924
20	7	$3{,}2091e+17$	$1{,}2202e+06$	2,0237	82,0924
20	20	$8{,}6326e+17$	$1{,}2202e+06$	3,4061	3,2812

Tabla 2.3: Números de condición de las matrices del experimento 2.

(*)En la tabla significa que no se han podido recuperar todos los parámetros.

2.4.3. Experimento 3.

En este caso los parámetros a considerar son

- $M = 6$.
- $(\lambda_1, \ldots, \lambda_6) = \frac{i}{1000}(200, 201, 202, 203, 204, 205)$.
- $(c_1, \ldots, c_6) = (6, 5, 4, 3, 2, 1)$.
- $\varepsilon = 10^{-10}$.

Los resultados obtenidos en este experimento, son reflejados en Tabla 2.3, dónde las casillas con * denotan la incapacidad de recuperar todos los elementos de (2.24). La importancia de este experimento radica en que a pesar de que, para determinados N,L, seamos incapaces de recuperar la suma de exponenciales, somos capaces de obtener buenas aproximaciones con menos parámetros de los dados por la distancia de separación. Dicha distancia vuelve a ser $\delta = 0{,}001$, sin embargo al estar los λ_j mucho más cerca los unos de los otros, es necesario tener más datos a priori que, por ejemplo, en el experimento anterior. Esto nos indica, que el problema del mal acondicionamiento de las matrices se puede solucionar tomando N, L más grandes.

N	L	$e(\vec{\lambda})$	$e(\vec{c})$	$e(f)$
Método de Prony (Algoritmo 2.2.1)				
300	300	$*$	$*$	$2{,}163e-13$
400	400	$*$	$*$	$8{,}0832e-12$
500	500	$4{,}9870e-06$	$4{,}3117e-04$	$9{,}2411e-13$
600	600	$7{,}1787e-07$	$6{,}6261e-05$	$1{,}3999e-12$
Lápices matriciales QR (Algoritmo 2.2.2)				
300	300	$*$	$*$	$2{,}163e-13$
400	400	$4{,}6849e-05$	$3{,}7112e-03$	$3{,}0549e-13$
500	500	$1{,}5242e-06$	$1{,}3588e-04$	$1{,}5354e-13$
600	600	$3{,}4043e-07$	$3{,}0496e-05$	$1{,}1975e-12$
Método ESPRIT (Algoritmo 2.2.3)				
300	300	$*$	$*$	$2{,}163e-13$
400	400	$2{,}3927e-04$	$1{,}8676e-02$	$1{,}4364e-13$
500	500	$1{,}5385e-05$	$1{,}4081e-03$	$2{,}5675e-13$
600	600	$1{,}7591e-06$	$1{,}6176e-04$	$5{,}4019e-13$

Tabla 2.4: Errores obtenidos en el experimento 3.

• **Nota:**Los experimentos 2 y 3 estan muy relacionados con la aproximacion dispersa de Fourier (e.g. [6]). Esta rama tiene como objetivo recuperar de la forma más eficiente posible las funciones definidas de la siguiente manera:

Definición 2.4. *Dado $L \in \mathbb{N}$, se dice que*

$$g(x) = \frac{\alpha_0}{2} + \sum_{k=1}^{L} \alpha_k cos(kx) + i \sum_{k=1}^{L} \beta_k sen(kx), \qquad x \in \mathbb{R}^+, \tag{2.43}$$

con $\alpha_k, \beta_k \in \mathbb{R}\backslash\{0\}, j = 1, \ldots, L$,es un polinomio trigonométrico complejo de grado L.

Sin embargo, basta darse cuenta que un polinomio trigonométrico f de grado L en (2.43), puede expresarse en la forma

$$g(x) = \sum_{j=1}^{M} c_j e^{i\omega_j x}, \qquad M = 2L + 1,$$

ya que tomando

$$c_j^{'} := \begin{cases} \frac{\alpha_k+\beta_k}{2} & \text{si } k \in \mathbb{Z}^+, \\ \frac{\alpha_k-\beta_k}{2} & \text{si } k \in \mathbb{Z}^-, \end{cases}$$

para $c_0^{'} := \frac{\alpha_0}{2}$, y $j = -N, \ldots, N$ se obtiene el paso intermedio

$$g(x) = \sum_{j=-L}^{L} c_j^{'} e^{i\omega_j x},$$

y tras un cambio de índices se obtiene la expresión original. En definitiva, los polinomios trigonométricos complejos son sumas de exponenciales complejas, y podemos aplicar los algoritmos desarrollados. Puesto que la distancia de separación era $\delta = 0{,}001$, la forma interesante de actuar sería definir el polinomio trigonométrico $g(x) := f(1000x)$, con f en (2.24), de donde se deduce que los $\omega_j = -1000\lambda_j$, con ω_j, λ_j en (2.43) y (2.24), resp. De esta forma, conociento tan solo $g(\frac{k}{1000}) = f(k), k = 0, \ldots, 2N-1$ para $N \geq 10$, se obtienen aproximaciones muy buenas.

□

El objetivo de los dos ultimos experimentos será aproximar funciones del tipo

$$g : [a, b] \longrightarrow \mathbb{C}, \qquad 0 \leq a < b < \infty,$$

por sumas de exponenciales complejas de un orden pre-fijado, e.g, [4]. Para ello, el procedimiento a seguir, será aplicar alguno de los métodos, utilizando como muestra los valores $g(a + \frac{b-a}{2N}k), k = 0, \ldots, 2N-1$. De esta forma conseguimos trasladar el problema de intervalo $[a, b]$ al intervalo $[0, 2N]$, que teniamos de forma original. Como resultado al aplicar el algoritmo, obtendremos una función $f : [0, 2N] \longrightarrow \mathbb{C}$ como en (2.24), y deshaciendo el cambio de variable anterior, tendremos que $f(2N\frac{t-a}{(b-a)}), t \in [a, b]$ es la aproximación buscada.

2.4.4. Experimento 4.

Consideramos M=20, y la función

$$\begin{aligned} g : [1, 10^6] &\longrightarrow \mathbb{C}. \\ t &\longmapsto \frac{1}{t} \end{aligned}$$

Aplicamos el método de lápices matriciales mediante factorización QR a los valores $g(1+\frac{k(10^6-1)}{2N}), N = 500, L = 250$, obteniendo como resultado los coeficientes c_j, λ_j representados en Tabla 2.5. Finalmente, formando la suma exponencial $f(x) = \sum_{j=1}^{M} c_j e^{\lambda_j x}$, tendremos que la aproximación de la función g en $[1, 10^6]$, será $f(2N\frac{t-1}{10^6-1}), t \in [1, 10^6]$. En este caso representamos gráficamente en la figura 2.1 el error absoluto de la función g original y la apoximación f obtenida, tomando como puntos de evaluación $x_k = 1 - \frac{10^6-1}{10^7-1}k, k = 0, \ldots, 10^7 - 1$.

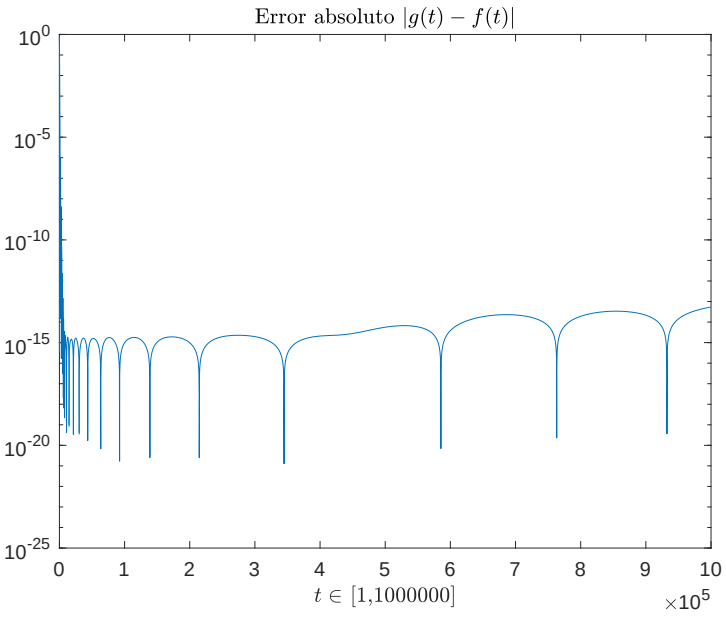


Figura 2.1: Experimento 4. Error absoluto entre g y f.

j	c_j	λ_j
1	$9{,}9585e-01$	$-1{,}1403e+01$
2	$1{,}6290e-03$	$-3{,}2207e+00$
3	$8{,}2389e-04$	$-2{,}0723e+00$
4	$5{,}1529e-04$	$-1{,}4197e+00$
5	$3{,}4716e-04$	$-9{,}9512e-01$
6	$2{,}4155e-04$	$-7{,}0422e-01$
7	$1{,}7062e-04$	$-5{,}0023e-01$
8	$1{,}2140e-04$	$-3{,}5561e-01$
9	$8{,}6705e-05$	$-2{,}5253e-01$
10	$6{,}2079e-05$	$-1{,}7882e-01$
11	$4{,}4547e-05$	$-1{,}2599e-01$
12	$3{,}2044e-05$	$-8{,}8040e-02$
13	$2{,}3094e-05$	$-6{,}0715e-02$
14	$1{,}6634e-05$	$-4{,}1022e-02$
15	$1{,}1914e-05$	$-2{,}6868e-02$
16	$8{,}4209e-06$	$-1{,}6786e-02$
17	$5{,}8091e-06$	$-9{,}7333e-03$
18	$3{,}8257e-06$	$-4{,}9590e-03$
19	$2{,}2593e-06$	$-1{,}9432e-03$
20	$9{,}3171e-07$	$-3{,}6134e-04$

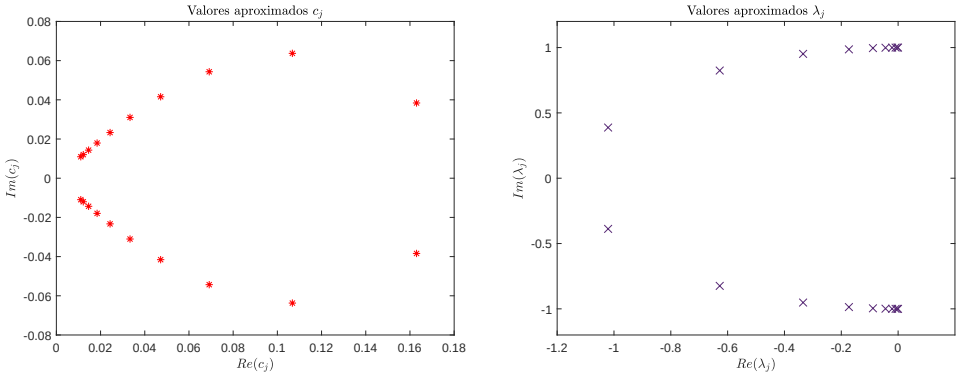
Tabla 2.5: Experimento 4. Parámetros de la suma exponencial que aproxima a g.

2.4.5. Experimento 5.

En este caso, se trata de aproximar las funciónes de Bessel de primera especie de orden 0, definidas por

$$J_0(t) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k!)^2} (\frac{t}{2})^{2k}, \qquad t \in [0, 1000].$$

Tomamos como parámetros $M = 20, N = 500$, y $L = 250$. El cambio de variable será $x = 2N\frac{(t-a)}{b-a} = t$. Los parámetros que constituyen la función f que aproxima a J_0, se representan en la figura 2.2, y el error absoluto se representa en la figura 2.3.



(a) Valores de c_j en f. (b) Valores de λ_j en f.

Figura 2.2: Experimento 5. Parámetros de f.

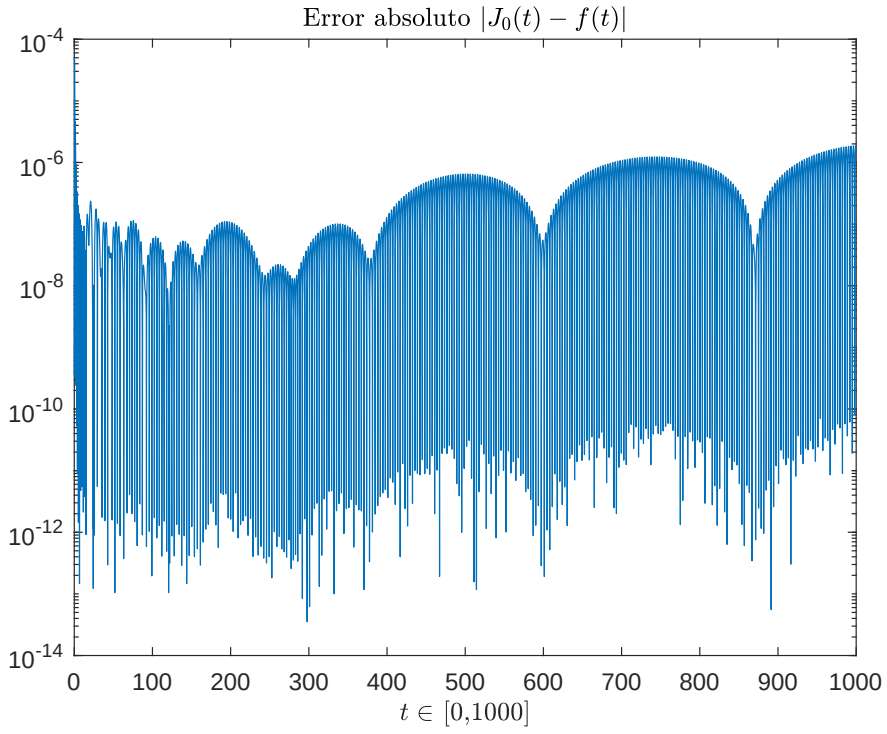


Figura 2.3: Experimento 5. Error absoluto entre g y f.

Capítulo 3

EL MÉTODO GENERALIZADO DE PRONY

El objetivo de este capítulo es dar una generalización natural de los métodos de Prony, a partir de los desarrollos vistos hasta el momento pero en un contexto más amplio, ya que no será necesario restringirse a la recuperación de sumas exponenciales complejas. Para ello, el siguiente paso de esta memoria será construir un método para recuperar desarrollos finitos de autofunciones en determinados operadores lineales.

3.1. Desarrollos finitos en autofunciones de un operador lineal.

Sea $(V, ||.||)$ un $\mathbb{C}$-espacio vectorial normado, y $\mathcal{A} : V \to V$ un operador lineal. Además, supondremos que existe al menos un $\lambda \in \mathbb{C}$ tal que $\mathcal{A}v = \lambda v$, para algún $v \in V$, es decir, $\mathcal{A}$ tiene autovalores (si V es un espacio de funciones, las $v \in V$ que verifiquen esta propiedad reciben el nombre de autofunciones).
Sea $\Lambda := \{\lambda_i : i \in I\}$, el conjunto de los autovalores (distintos) del operador $\mathcal{A}$, con I un conjunto de subindices. Consideramos además $\mathcal{V}_i := \langle v_i \rangle, i \in I$, con v_i alguna de las autofunciones que verifique $\mathcal{A}v_i = \lambda_i v_i$ y $||v_i|| = 1$.

Definición 3.1. *En las condiciones anteriores, para un $M \in \mathbb{N}$, se dice que*

$$f = \sum_{i \in I} c_i v_i, \qquad c_i \in \mathbb{C},$$

es un desarrollo M-finito de autofunciones del operador $\mathcal{A}$, si M coeficientes c_i son no nulos. Es decir, existe $J \subset I, |J| = M$ tal que

$$f = \sum_{j \in J} c_j v_j. \tag{3.1}$$

3.1.1. Método generalizado de Prony.

El primer paso en la generalización del método de Prony, resuelve el problema de recuperar un desarrollo M-finito de autofunciones (3.1), conocido el orden $M \in \mathbb{N}$ del mismo. Razonando de forma similar a la sección 2.1, sea $F \in V^*$ tal que $F(v_j) \neq 0, \forall j \in J$, se puede recuperar f en (3.1) a partir de los valores $F(\mathcal{A}^k f), k = 0, \ldots, 2M-1$.[1] Para ello, se define el polinomio de Prony,

$$p(z) := \prod_{j \in J}(z - \lambda_j) = z^M + \sum_{k=0}^{M-1} a_k z^k, \tag{3.2}$$

con $\lambda_j, j \in J$, los autovalores asociados a las autofunciones v_j en (3.1). Debido a la linealidad del operador $\mathcal{A}$ y de F, a partir de (3.1) se deduce que para $m \geq 0$ y $a_M := 1$,

$$\begin{aligned}\sum_{k=0}^{M} a_k F(\mathcal{A}^{k+m} f) &= \sum_{k=0}^{M} a_k F(\sum_{j \in J} c_j \lambda_j^{k+m} v_j) = \sum_{j \in J} c_j \lambda_j^m F(v_j)(\sum_{k=0}^{M} a_k \lambda_j^k) \\ &= \sum_{j \in J} c_j \lambda_j^m F(v_j) p(\lambda_j) = 0.\end{aligned}$$

Por tanto, tiene sentido considerar el sistema de ecuaciones dado por

$$\sum_{k=0}^{M-1} a_k F(\mathcal{A}^{k+m} f) = -F(\mathcal{A}^{M+m} f), \qquad m = 0, \ldots, M-1. \tag{3.3}$$

[1] V* denota el espacio dual topológico de V.

La determinación de los coeficientes del polinomio (3.2) puede hallarse interpretando (3.3) como un sistema lineal con incógnitas $a_0, \ldots, a_{M-1}$ expresado de la forma

$$G_M \begin{bmatrix} a_0 \\ \vdots \\ a_{M-1} \end{bmatrix} = - \begin{bmatrix} F(\mathcal{A}^M f) \\ \vdots \\ F(\mathcal{A}^{2M-1} f) \end{bmatrix}, \tag{3.4}$$

para

$$G_M := \begin{bmatrix} F(f) & \ldots & F(\mathcal{A}^{M-1} f) \\ F(\mathcal{A} f) & \ldots & F(\mathcal{A}^M f) \\ \vdots & \ddots & \vdots \\ F(\mathcal{A}^{M-1} f) & \ldots & F(\mathcal{A}^{2M-2} f) \end{bmatrix}. \tag{3.5}$$

El sistema (3.4) tiene solución única, ya que la matriz G_M en (3.5) es invertible, como consecuencia del siguiente resultado:

Lema 3.1. *Sean $\{\lambda_j\}_{j\in J}$ en (3.2), $(c_j)_{j\in J}$ y $(F(v_j))_{j\in J}$ en (3.1). Entonces*

$$G_M = V_\lambda \; diag(c_j)_{j\in J} \; diag(F(v_j))_{j\in J} \; V_\lambda^T,$$

con la matriz de Vandermonde $V_\lambda = (\lambda_j^k)_{k=0,j\in J}^{M-1}$.
Además $(c_j)_{j\in J}$ es solución del sistema

$$\begin{bmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ \lambda_{j_1}^{2M-1} & \ldots & \lambda_{j_M}^{2M-1} \end{bmatrix} \begin{bmatrix} F(v_{j_1}) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & F(v_{j_M}) \end{bmatrix} \begin{bmatrix} c_{j_1} \\ \vdots \\ c_{j_M} \end{bmatrix} = \begin{bmatrix} F(f) \\ \vdots \\ F(\mathcal{A}^{2N-1} f) \end{bmatrix}, \tag{3.6}$$

Demostración. Puesto que $|J| = M$, tomamos $J = \{j_1, \ldots, j_M\}$. Haciendo el producto de matrices en el término de la derecha, utilizando la linealidad de F y la expresión de f en (3.1),

$$\begin{bmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ \lambda_{j_1}^{M-1} & \ldots & \lambda_{j_M}^{M-1} \end{bmatrix} \begin{bmatrix} c_{j_1} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & c_{j_M} \end{bmatrix} \begin{bmatrix} F(v_{j_1}) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & F(v_{j_M}) \end{bmatrix} \begin{bmatrix} 1 & \ldots & \lambda_{j_1}^{M-1} \\ \vdots & \ddots & \vdots \\ 1 & \ldots & \lambda_{j_M}^{M-1} \end{bmatrix} =$$

$$\begin{bmatrix} c_{j_1} F(v_{j_1}) + \cdots + c_{j_M} F(v_{j_M}) & \ldots & \lambda_{j_1}^{M-1} c_{j_1} F(v_{j_1}) + \cdots + \lambda_{j_M}^{M-1} c_{j_M} F(v_{j_M}) \\ \vdots & \ddots & \vdots \\ \lambda_{j_1}^{M-1} c_{j_1} F(v_{j_1}) + \cdots + \lambda_{j_M}^{M-1} c_{j_M} F(v_{j_M}) & \ldots & \lambda_{j_1}^{2M-2} c_{j_1} F(v_{j_1}) + \cdots + \lambda_{j_M}^{2M-2} c_{j_M} F(v_{j_M}) \end{bmatrix} =$$

$$\begin{bmatrix} F(f) & \dots & F(\mathcal{A}^{M-1}f) \\ \vdots & \ddots & \vdots \\ F(\mathcal{A}^{M-1}f) & \dots & F(\mathcal{A}^{2M-2}f) \end{bmatrix} = G_M.$$

Por tanto, como por hipótesis los $\{\lambda_j\}_{j\in J}$ son distintos, la matriz V_λ es invertible, además $c_j \neq 0$, y $F(v_j) \neq 0, j \in J$, luego $diag(c_j)_{j\in J}$ y $diag(F(v_j))_{j\in J}$ son invertibles, y en consecuencia G_M también.
La segunda parte es clara, pues haciendo el producto matricial en el término de la izquierda:

$$\begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ \lambda_{j_1}^{2M-1} & \dots & \lambda_{j_M}^{2M-1} \end{bmatrix} \begin{bmatrix} F(v_{j_1}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & F(v_{j_M}) \end{bmatrix} \begin{bmatrix} c_{j_1} \\ \vdots \\ c_{j_M} \end{bmatrix} =$$

$$\begin{bmatrix} c_{j_1}F(v_{j_1}) + \dots + c_{j_m}F(v_{j_m}) \\ \vdots \\ c_{j_1}\lambda_{j_1}^{2M-1}F(v_{j_1}) + \dots + c_{j_m}\lambda_{j_m}^{2M-1}F(v_{j_m}) \end{bmatrix} = \begin{bmatrix} F(f) \\ \vdots \\ F(\mathcal{A}^{2N-1}f) \end{bmatrix}$$

□

A partir de la determinación de los coeficientes del polinomio de Prony (3.2), se pueden recuperar las autofunciones y coeficientes en (3.1) mediante los siguientes pasos, que constituyen el método generalizado de Prony:

Algoritmo 3.1.1. Método generalizado de Prony.

Entradas: $M \in \mathbb{N}$, valores medidos $F(\mathcal{A}^k f), k = 0, ..., 2M-1$ en (3.1).

Paso 1) Resolver el sistema lineal (3.4).

Paso 2) Emplear alguno de los métodos vistos en el capítulo 2 para hallar las raices λ_j del polinomio de Prony (3.2). Y posteriormente, determinar las autofunciones (normalizadas) v_j correspondientes.

Paso 3) Los coeficientes $c_j, j \in J$ se obtienen como solución del sistema sobredeterminado (3.6).

Salidas: $c_j, v_j, j \in J$.

■

De forma simétrica, con las correspondientes adaptaciones a la construcción hecha en la sección 2.2, se puede extender el algoritmo generalizado de Prony para el caso en el que tenemos desarrollos de funciones dispersos, pero desconocemos su orden. Es decir, en las condiciones anteriores, pero sin conocer M, suponemos conocidos $F(\mathcal{A}^k f), k = 0, \ldots, 2N-1$, y $N \geq M$ adecuado. Además, como ocurria en el capítulo anterior, suponemos conocido un $L \in \mathbb{N}$, tal que $N \geq L \geq M$.

3.1.2. Método QR/ESPRIT.

En primer lugar, consideramos la matriz de Hankel G_M en (3.5), y a partir de la relación dada en (3.4), denotando por $g_M := (F(\mathcal{A}^M), \ldots, F(\mathcal{A}^{2M-1}))^T$, y considerando la matriz compañera $C(p)$ del polinomio de Prony (3.2), obtenemos la siguiente igualdad

$$G\ C(p) = \tilde{G},$$

con $\tilde{G} = (G(1:M, 2:M), g_M)$. De nuevo, los autovalores de la matriz compañera $C(p)$ son los $\lambda_j, j \in J$, en (3.2). Y como vimos anteriormente también son los autovalores del lápiz matricial $zG - \tilde{G}$. De esta forma, podemos aplicar el algoritmo 2.2.2 o 2.2.3, sustituyendo la matriz de Hankel $H_{2N-L,L+1}$ por

$$G_{2N-L,L+1} := \begin{bmatrix} F(f) & \ldots & F(\mathcal{A}^L f) \\ F(\mathcal{A}f) & \ldots & F(\mathcal{A}^{L+1} f) \\ \vdots & \ddots & \vdots \\ F(\mathcal{A}^{2N-L-1} f) & \ldots & F(\mathcal{A}^{2N-1} f) \end{bmatrix} \in \mathbb{C}_{(2N-L)\times(L+1)}. \quad (3.7)$$

Esta construcción es posible, ya que las demostraciones empleadas en el capítulo 2 para desarrollar los métodos QR/SVD se emplearon argumentos basados únicamente en el rango de las matrices, por lo que sigue siendo válido. El paso tres de ambos algoritmos ha de modificarse, originando el algoritmo siguiente:

Algoritmo 3.1.2. Método QR/ESPRIT para el método de Prony generalizado.

Entradas: L,N$\in \mathbb{N}$, N$\gg$1, N$\geq$L$\geq$3, con L cota superior adecuada de M en (3.1), $F(\mathcal{A}^k f)$ valores medidos , $k = 0, ..., 2N-1$ en (3.1), y $0 < \varepsilon \ll 1$.

Paso 1) Hallar la factorización QR/SVD de la matriz $G_{2N-L,L+1}$ en (3.7). Determinar el rango numérico M en (2.32) empleando el método correspondiente al desarrollo QR/SVD, y construir las matrices $T_{M,L}(s)/W_{M,L}(s)$, $s = 0, 1$, correspondientes.

Paso 2) Hallar los autovalores $\lambda_j, j \in J$ de la matriz $F_M^{QR} := (T_{M,L}(0)^T)^+ T_{M,L}(1)$ ó de $F_M^{SVD} := (W_{M,L}(0)^T)^+ W_{M,L}(1)$. A partir de estos, calcular las autofunciones (normalizadas), $v_j, j \in J$ correspondientes.

Paso 3) Los coeficientes $\vec{c} = (c_j)_{j\in J}^T$ se obtienen a partir de la resolución por mínimos cuadrados del sistema sobredeterminado;

$$F(\mathcal{A}^k f) = \sum_{j\in J} c_j \lambda_j F(v_j), \qquad k = 0, \ldots, 2N-1.$$

Salidas: $c_j, v_j, j \in J$

■

3.1.3. Autofunciones generalizadas.

El último paso en la generalización del método es extender su definición a los desarrollos en autofunciones generalizadas:

Definición 3.2. *Sea $r \in \mathbb{N}$, V un espacio vectorial de dimensión finita, y $\mathcal{A} : V \to V$ un operador lineal. Decimos que $\tilde{v}_l \in V, l = 1, \ldots, r$, son autofunciones generalizadas con multiplicidad l del operador $\mathcal{A}$, asociadas al autovalor λ, si verifican*

$$(\mathcal{A} - \lambda I)^l \ \tilde{v}_l = 0, \quad l = 1, \ldots, r,$$

y además

$$\mathcal{A}\tilde{v}_l = \lambda\tilde{v}_l + \sum_{s=1}^{l-1} \alpha_{l,s}\tilde{v}_s, \qquad l = 1, \ldots, r, \tag{3.8}$$

con $\alpha_{l,s} \in \mathbb{C}$.

De nuevo, consideramos $\Lambda := \{\lambda_i : i \in I\}$, el conjunto de los autovalores (distintos) del operador $\mathcal{A}$, con I un conjunto de subíndices. Además, para cada $i \in I$ fijamos los conjuntos $\mathcal{V}_i := \{\tilde{v}_{i,l}, l = 1, \ldots, r\}$, un conjunto de autofunciones generalizadas linealmente independientes asociadas al autovalor λ_i. Con estas consideraciones, llegamos a la siguiente defínción:

Definición 3.3. *En estas condiciones, dados* $r, M \in \mathbb{N}$, *se dice que*

$$f = \sum_{i\in I} \sum_{l=1}^{r} c_{i,l}\ \tilde{v}_{i,l},$$

es un desarrollo M-disperso de autofunciones generalizadas del operador $\mathcal{A}$, *si existe* $J \subset I, |J| = M$ *tal que* $c_{i,l} = 0$, *para todo* $i \in I - J, l = 1, \ldots, r$ *y* $\sum_{l=1}^{r} |c_j|^2 \neq 0, \forall j \in J$. *Es decir,*

$$f = \sum_{j\in J} \sum_{l=1}^{r} c_{j,l}\ \tilde{v}_{j,l}. \tag{3.9}$$

La construcción sigue los pasos anteriores, ya que dado $F \in V^*$, tal que $F(\tilde{v}_{i,l}) \neq 0, i \in I, l = 1, \ldots, r$ y suponiendo que la matriz

$$\begin{bmatrix} F(f) & \ldots & F(\mathcal{A}^{rM-1}f) \\ \vdots & \ddots & \vdots \\ F(\mathcal{A}^{rM-1}f) & \ldots & F(\mathcal{A}^{2rM-2}f) \end{bmatrix},$$

es invertible, se puede recuperar la función f en (3.9) a partir de los valores $F(\mathcal{A}^k f), k = 0, \ldots, 2rM - 1$. A modo de ilustración, consideramos el caso en el que las autofunciones generalizadas asociadas al autovalor λ_j verifican:

$$\mathcal{A}\tilde{v}_{j,l} = \lambda_j\tilde{v}_{j,l} + \alpha_{j,l-1}\tilde{v}_{j,l-1}, \qquad l = 2, \ldots, r. \tag{3.10}$$

A partir de esta igualdad se verifica el siguiente resultado:

Lema 3.2. *Sean $\tilde{v}_{j,l}$ las autofunciones generalizadas asociadas al autovalor λ_j en (3.8) verificando (3.10). Entonces*

$$\mathcal{A}^k\tilde{v}_{j,l} = \sum_{s=0}^{l-1} \binom{k}{k-s} \lambda_j^{k-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}),$$

definimos $\binom{k}{k-s} := 0$ si $k < s$.

Demostración. Por inducción sobre k:
Para k=1:

$$\begin{aligned}\mathcal{A}\tilde{v}_{j,l} =& \sum_{s=0}^{l-1} \binom{1}{1-s} \lambda_j^{1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) \\ =& \lambda_j \tilde{v}_{j,l-s} + \tilde{v}_{j,l-s}\alpha_{j,l-1} = \mathcal{A}\tilde{v}_{j,l},\end{aligned}$$

por definición en (3.10).
Suponemos cierto para k. Veamos que se cumple para k+1, empleando la linealidad del operador $\mathcal{A}$,

$$\begin{aligned}\mathcal{A}^{k+1}\tilde{v}_{j,l} =& \mathcal{A}\mathcal{A}^k\tilde{v}_{j,l} \overset{HI}{=} \mathcal{A}\Big(\sum_{s=0}^{l-1} \binom{1}{1-s} \lambda_j^{1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) \Big) \\ =& \sum_{s=0}^{l-1} \binom{1}{1-s} \lambda_j^{1-s} \mathcal{A}(\tilde{v}_{j,l-s}) (\prod_{t=1}^{s} \alpha_{j,l-t}) \\ =& \sum_{s=0}^{l-1} \binom{1}{1-s} \lambda_j^{1-s} \mathcal{A}(\prod_{t=1}^{s} \alpha_{j,l-t}) (\lambda_j \tilde{v}_{j,l-s} + \alpha_{j,l-s-1}\tilde{v}_{j-l-s-1}) \\ =& \sum_{s=0}^{l-1} \binom{k}{k-s} \lambda_j^{k+1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) + \sum_{s=0}^{l-1} \binom{k}{k-s} \lambda_j^{k-s} \tilde{v}_{j,l-s-1} (\prod_{t=1}^{s+1} \alpha_{j,l-t}). \\ \overset{(*)}{=}& \sum_{s=0}^{l-1} \binom{k}{k-s} \lambda_j^{k+1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) + \sum_{s=0}^{l-1} \binom{k}{k+1-s} \lambda_j^{k+1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) \\ =& \sum_{s=0}^{l-1} \left(\binom{k}{k-s} + \binom{k}{k+1-s} \right) \lambda_j^{k+1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}) \\ =& \sum_{s=0}^{l-1} \binom{k+1}{k+1-s} \lambda_j^{k+1-s} \tilde{v}_{j,l-s} (\prod_{t=1}^{s} \alpha_{j,l-t}).\end{aligned}$$

En el paso $(*)$, consideramos el segundo sumando, y definimos $\binom{k}{k+1} := 0$ y $\alpha_{j,0} := 0$,

$$\begin{aligned}\sum_{s=0}^{l-1}\binom{k}{k-s}\lambda_j^{k-s}\tilde{v}_{j,l-s-1}(\prod_{t=1}^{s+1}\alpha_{j,l-t}) &= \sum_{s=1}^{l}\binom{k}{k+1-s}\lambda_j^{k+1-s}\tilde{v}_{j,l-s}(\prod_{t=1}^{s}\alpha_{j,l-t}) \\ &= \sum_{s=0}^{l-1}\binom{k}{k+1-s}\lambda_j^{k+1-s}\tilde{v}_{j,l-s}(\prod_{t=1}^{s}\alpha_{j,l-t}),\end{aligned}$$

ya que para s=0,s=1, los términos se anulan. □

Consideramos ahora el polinomio generalizado de Prony, definido por

$$P(z) := \prod_{j\in J}(z-\lambda_j)^r = \sum_{k=0}^{Mr} a_k z^k, \tag{3.11}$$

con $a_k \in \mathbb{C}, k = 0,\dots,Mr, a_{Mr} := 1$, y λ_j los autovalores asociados a las autofunciones generalizadas en (3.9). De forma análoga al caso clásico, teniendo en cuenta la linealidad de F y del operador $\mathcal{A}$, obtenemos las siguientes relaciones:

$$\begin{aligned}\sum_{k=0}^{Mr} a_k F(\mathcal{A}^{k+m}f) &= \sum_{k=0}^{Mr} a_k F\Big(\sum_{j\in J}\sum_{l=1}^{r} c_{j,l}\ \mathcal{A}^{k+m}\tilde{v}_{j,l}\Big) \\ &= \sum_{k=0}^{Mr} a_k F\Big(\sum_{j\in J}\sum_{l=1}^{r} c_{j,l}\ \sum_{s=0}^{l-1}\binom{k+m}{k+m-s}\lambda_j^{k+m-s}\tilde{v}_{j,l-s}(\prod_{t=1}^{s}\alpha_{j,l-t})\Big) \\ &= \sum_{j\in J}\sum_{l=1}^{r} c_{j,l}\sum_{s=0}^{l-1}\Big(\Big(\sum_{k=0}^{Mr} a_k\binom{k+m}{k+m-s}\lambda_j^{k+m-s}(\prod_{t=1}^{s}\alpha_{j,l-t})\Big)F(\tilde{v}_{j,l-s})\Big),\end{aligned} \tag{3.12}$$

para cada $m = 0,\dots,Mr-1$. El siguiente paso, es ver que para $m = 0,\dots,Mr-1, j \in J, l = 1,\dots,r$ y $s = 0,\dots,l-1$, el término

$$\sum_{k=0}^{Mr} a_k\binom{k+m}{k+m-s}\lambda_j^{k+m-s}(\prod_{t=1}^{s}\alpha_{j,l-t}),$$

se puede expresar como combinación lineal del polinomio generalizado de Prony, y de sus r-1 primeras derivadas, evaluadas en los $\lambda_j, j \in J$, que son

de la forma:

$$P^{(\nu)}(z) = \sum_{k=\nu}^{Mr} a_k \frac{k!}{(k-\nu)!} z^{k-\nu}, \qquad \nu = 0, \ldots, r-1.$$

Observamos que los polinomios

$$q_\nu(x) = x(x-1)\ldots(x-\nu+1), \qquad \nu = 0, \ldots, r-1,$$

forman una base del espacio de los polinomios de grado hasta r-1. Esto implica que cada polinomio en k de grado menor o igual que r-1 puede escribirse como combinación lineal de $q_\nu, \nu = 0, \ldots, r-1$.
Como

$$P^{(\nu)}(\lambda_j) = \sum_{k=\nu}^{Mr} a_k \lambda_j^{k-\nu} q_\nu(k), \qquad v = 0, \ldots, r-1,$$

podemos reescribir los $q_\nu(k)$ en función de $P^{(\nu)}(\lambda_j), \nu = 0, \ldots, r-1, j \in J$. Pero como cada λ_j tiene multiplicidad r en la definición del polinomio generalizado de Prony (3.11), se concluye que $P^{(\nu)}(\lambda_j) = 0, \nu = 0, \ldots, r-1$.

Por lo tanto, en (3.12) se obtiene que $\sum_{k=0}^{Mr} a_k F(\mathcal{A}^{k+m} f) = 0$, para $m = 0, \ldots, Mr-1$. Y el sistema lineal de ecuaciones asociado,

$$\sum_{k=0}^{Mr-1} a_k F(\mathcal{A}^{k+m} f) = -F(\mathcal{A}^{rM+m} f), \qquad m = 0, \ldots, rM-1, \tag{3.13}$$

es de nuevo de tipo Hankel. Finalmente, el algoritmo asociado a este método sería el siguiente:

Algoritmo 3.1.3. Método de Prony para autofunciones generalizadas.

Entradas: $r, M \in \mathbb{N}$, valores medidos $F(\mathcal{A}^k f), k = 0, ..., 2rM-1$ en (3.9).

Paso 1) Resolver el sistema lineal (3.13), para obtener los coeficientes del polinomio de Prony generalizado (3.11).

Paso 2) Emplear alguno de los métodos vistos en el capítulo 2 para hallar las raices λ_j del polinomio de Prony generalizado (3.11). Y posteriormente, determinar las autofunciones generalizadas $\tilde{v}_{j,l}, l = 1, \ldots, r$ correspondientes.

Paso 3) Los coeficientes $c_{j,l}, j \in J, l = 1, \ldots, r$ se obtienen como solución del sistema sobredeterminado:

$$F(\mathcal{A}^k f) = \sum_{j \in J} \sum_{l=1}^{r} c_{j,l} \mathcal{A}^k \tilde{v}_{j,l}, \qquad k = 0, \ldots, 2rM - 1$$

Salidas: $c_j, v_j, j \in J$.

■

3.2. Ejemplos de problemas destacados.

Debido a la propia construcción del método generalizado de Prony, el problema que surge recae sobre las posibles elecciónes del operador $\mathcal{A}$ y el funcional F. Para ello, se exponen una serie de ejemplos de utilidad en diversos campos. [18, 15]

3.2.1. Sumas finitas de exponenciales complejas y monomios.

El problema de recuperar sumas finitas de exponenciales complejas es exactamente el problema que resuelve el método clasico de Prony, el cual puede ser visto como un caso particular del método generalizado, ya que considerando $V = \mathcal{C}(\mathbb{R})$, el espacio de funciones continuas en $\mathbb{R}$, y el operador

$$\begin{aligned} \mathcal{S} : \mathcal{C}(\mathbb{R}) &\longrightarrow \mathcal{C}(\mathbb{R}) \\ f &\longmapsto \mathcal{S}f : \mathbb{R} \longrightarrow \mathbb{R} \\ &\qquad x \longmapsto f(x+1), \end{aligned}$$

denominado operador "shift", entonces, para $\lambda \in [-\alpha, 0] + i[-\pi, \pi), \alpha > 0$, y $f(x) = e^{\lambda x}$, se verifica

$$\mathcal{S}f(x) = \mathcal{S}e^{\lambda x} = e^{\lambda(x+1)} = e^{\lambda} e^{\lambda x} = e^{\lambda} f(x),$$

por lo tanto el conjunto $\Lambda := \{e^{\lambda} : \lambda \in [-\alpha, 0] + i[-\pi, \pi), \alpha > 0\}$ es un conjunto de autovalores distintos de $\mathcal{S}$, asociados a las autofunciones $\mathcal{V} :=$

$\{e^{\lambda x} : e^{\lambda} \in \Lambda\}$. Tomando como funcional la delta de Dirac, $F(f) = \delta(f) = f(0)$, la suma de M exponenciales complejas en (2.1) se puede recuperar de forma inequívoca a partir de los valores

$$\delta(\mathcal{S}^k f(x)) = \delta(f(x+k)) = \delta(\sum_{j=1}^{M} c_j e^{\lambda_j (x+k)})$$
$$= \sum_{j=1}^{M} c_j \delta(e^{\lambda_j (x+k)}) = \sum_{j=1}^{M} c_j e^{\lambda_j k} = f(k),$$

$k = 0, \ldots, 2M-1$. Obteniendo así el método clásico de Prony visto en el capítulo 2.
La elección del operador shift no es única, ya que hay otros operadores que tienen a los $\{e^{\lambda} : \lambda \in \mathbb{C}\}$ como autovalores, el más destacado es el operador diferencial

$$\frac{d}{dx} : \mathcal{C}^{\infty}(\mathbb{R}) \longrightarrow \mathcal{C}^{\infty}(\mathbb{R}).$$
$$f \longmapsto f'. \tag{3.14}$$

Ya que $\frac{d}{dx} e^{\lambda x} = \lambda e^{\lambda x}$. Por tanto, para $x_0 \in \mathbb{R}$ y considerando como funcional la delta de Dirac, podemos recuperar la suma finita de exponenciales complejas

$$f(x) = \sum_{j=1}^{M} c_j e_j^{\lambda} x, \qquad \lambda_j \neq \lambda_i \ si \ i \neq j,$$

a partir de los valores

$$\delta_{x_0}(\frac{d^k}{dx^k} f(x)) = \sum_{j=1}^{M} c_j \delta_{x_0}(\frac{d^k}{dx^k} e^{\lambda_j x}) = \sum_{j=1}^{M} c_j \lambda_j^k e^{\lambda_j x_0} = f^{(k)}(x_0),$$

$k = 0, \ldots, 2M-1$. De nuevo tomando como operador (3.14), $r \in \mathbb{N}$, y $\{q_l\}_{l=0}^{r}$ una base del espacio de polinomios de grado menor que $r+1$, con $deg(q_l) = l, l = 0, \ldots, r$. Podemos recuperar la función

$$f(x) = \sum_{j=1}^{M} \sum_{l=0}^{r} c_{j,l} q_l(x) e^{\lambda_j x},$$

a partir de los valores

$$F(\frac{d^k}{dx^k}f(x)) = \delta_{x_0}(\frac{d^k}{dx^k}f(x)) = \sum_{j=1}^{M}\sum_{l=0}^{r} c_{j,l}\delta_{x_0}(\frac{d^k}{dx^k}q_l(x)e^{\lambda_j x}) =$$
$$= \sum_{j=1}^{M}\sum_{l=0}^{r} c_{j,l}(q_l^{(k)}(x_0)e^{\lambda_j x_0} + \lambda_j^k q_l(x_0)e^{\lambda_j x_0}) = f^{(k)}(x_0),$$

$k = 0, \ldots, 2M(r+1)$. Esto se debe a que $q_l(x)e^{\lambda_j x}, l = 0, \ldots, r$, son autofunciones generalizadas de multiplicidad $l+1$ del operador $\frac{d}{dx}$ pues verifican

$$(\frac{d}{dx} - \lambda Id)^{l+1}(q_l(x)e^{\lambda x}) = (\frac{d}{dx} - \lambda I)^l(q_l^{'}(x)e^{\lambda x} + \lambda q_l(x)e^{\lambda x} - \lambda q_l(x)e^{\lambda x}) =$$
$$\cdots = (q_l^{(l+1)}(x)e^{\lambda x} + \lambda^{l+1}q_l(x)e^{\lambda x} - \lambda^{l+1}q_l(x)e^{\lambda x}) = 0,$$

y

$$\frac{d}{dx}(q_l(x)e^{\lambda x}) = \lambda q_l(x)e^{\lambda x} + q_l^{'}(x)e^{\lambda x} = \lambda q_l(x)e^{\lambda x} + \sum_{s=1}^{l-1}\alpha_{l,s}q_s(x)e^{\lambda x},$$

para ciertos $\alpha_{l,s} \in \mathbb{C}$, pues $q_l^{'}(x)$ es un polinomio de grado $l-1$ y puede expresarse como combinación lineal de la base $\{q_s(x)\}_{s=0}^{l-1}$.

En cuanto a las sumas finitas de monomios, consideramos el espacio de funciones continuas $\mathcal{C}(\mathbb{R})$, y el operador

$$\begin{aligned}\mathcal{D}_a : \mathcal{C}(\mathbb{R}) &\longrightarrow \mathcal{C}(\mathbb{R}),\\ f &\longmapsto \mathcal{D}_a f : \mathbb{R} \longrightarrow \mathbb{R}\\ &\qquad x \longmapsto f(ax)\end{aligned}$$

$a \in \mathbb{R}^+\backslash\{1\}$, denominado "operador dilatación" en $\mathcal{C}(\mathbb{R})$. Como

$$\mathcal{D}_a x^p = (ax)^p = a^p x^p,$$

entonces $\{x^p : p \in \mathbb{C}\}$ es un conjunto de autofunciones de $\mathcal{D}_a$ asociado a los autovalores a^p. Para asegurar que los autovalores a^p son distintos para distintos p suponemos que $Im\ p \in [-\frac{\pi}{\ln a}, \frac{\pi}{\ln a})$, de manera que

$$\Lambda = \{a^p, p \in \mathbb{C}, Imp \in [-\frac{\pi}{\ln a}, \frac{\pi}{\ln a})\},$$

es un conjunto de autovalores de $\mathcal{D}_a$ y

$$\mathcal{V} = \{x^p : p \in \mathbb{C}, Im(p) \in [-\frac{\pi}{\ln a}, \frac{\pi}{\ln a})\},$$

el correspondiente conjunto de autofunciones. Como funcional F, tomamos de nuevo la delta de Dirac en un punto $x_0 \in \mathbb{R}\backslash\{0\}$ arbitrario, por lo que $\delta_{x_0}(x^p) = x_0^p \neq 0$. Y se puede reconstruir la suma finita de monomios

$$f(x) = \sum_{j=1}^{M} c_j x^{p_j}, \qquad c_j \in \mathbb{C}, p_j \in \mathbb{C} \text{ tal que } Im\ (p_j) \in [-\frac{\pi}{\ln a}, \frac{\pi}{\ln a}),$$

a partir de los valores

$$\begin{aligned}\delta_{x_0}(\mathcal{D}_a^k f(x)) =& \delta_{x_0}(\sum_{j=1}^{M} c_j \mathcal{D}_a^k x^{p_j}) = \delta_{x_0}(\sum_{j=1}^{M} c_j a^{kp_j} x^{p_j}) = \\ &= \sum_{j=1}^{M} c_j a^{kp_j} x_0^{p_j} = f(a^k x_0),\end{aligned}$$

para $k = 0, \ldots, 2M-1$.
También se pueden considerar las autofunciones generalizadas del operador dilatación, esta vez definido como $\mathcal{D}_a : \mathcal{C}(0,\infty) \longrightarrow \mathcal{C}(0,\infty)$, para $a \in \mathbb{R}^+\backslash\{0\}$ y $x_0 \in \mathbb{R}^+\backslash\{0\}$. En estas condiciones, $\{(\ln x)^l x^p\}_{l=0}^r$ son autofunciones generalizadas de multiplicidad $l+1$ del operador dilatación con autovalores a^p, pues

$$\begin{aligned}\mathcal{D}_a((\ln x)^l x^p) &= (\ln ax)^l (ax)^p = \\ &= (\ln a + \ln x)^l a^p x^p = a^p (\ln x)^l x^p + \sum_{s=0}^{l-1} \binom{l}{s} (\ln a)^{l-s} a^p (\ln x)^s x^p.\end{aligned}$$

Utilizando esta igualdad, se deduce la otra condición,

$$\begin{aligned}(\mathcal{D}_a - a^p Id)^{l+1}(\ln x)^l x^p =& (\mathcal{D}_a - a^p Id)^l (\sum_{s=0}^{l-1} \binom{l}{s} (\ln a)^{l-s} a^p (\ln x)^s x^p) = \\ &= \sum_{s=0}^{l-1} \binom{l}{s} (\ln a)^{l-s} a^p (\mathcal{D}_a - a^p Id)^l ((\ln x)^s x^p)) \\ &= \cdots = \sum_{s=0}^{l-1} \sum_{s_1=0}^{s-1} \cdots \sum_{s_{l-1}} (\mathcal{D}_a - a^p Id)(x^p) = 0.\end{aligned}$$

En consecuencia, podemos recuperar el desarrollo finito en autofunciones generalizadas de la forma

$$f(x) = \sum_{j=1}^{M} \sum_{l=0}^{r} c_{j,l} (\ln x)^l x^{p_j}, \qquad c_{j,l} \in \mathbb{C},$$

a partir de los valores

$$\begin{aligned} \delta_{x_0}(\mathcal{D}_a^k(\sum_{j=1}^{M} \sum_{l=0}^{r} c_{j,l} (\ln x)^l x^{p_j})) &= \sum_{j=1}^{M} \sum_{l=0}^{r} c_{j,l} \delta_{x_0}((\ln a^k x)^l (a^k x)^{p_j}) \\ &= \sum_{j=1}^{M} \sum_{l=0}^{r} c_{j,l} (\ln a^k x_0)^l (a^k x_0)^{p_j} = f(a^k x_0), \end{aligned}$$

$k = 0, \ldots, 2(r+1)M - 1$.
Por último, también se puede considerar el operador diferencial

$$\begin{aligned} d_x : \mathcal{C}^\infty(\mathbb{R}) &\longrightarrow \mathcal{C}^\infty(\mathbb{R}). \\ f &\longmapsto d_x f : \mathbb{R} \longrightarrow \mathbb{R} \\ & \qquad x \longmapsto \frac{d}{dx}(xf(x)) = f(x) + xf^{'}(x). \end{aligned}$$

En este caso, para un $x_0 \in \mathbb{R}\backslash\{0\}$, las autofunciones son de la forma $\{x^p, p \in \mathbb{R}\}$,con autovalores asociados $\{p+1, p \in \mathbb{R}\}$, ya que verifican

$$d_x(x^p) = \frac{d}{dx}(x^{p+1}) = (p+1)x^p.$$

Concluyendo así, que las funciones de la forma

$$f(x) = \sum_{j=1}^{M} c_j x^{p_j}, \qquad c_j \in \mathbb{C} - \{0\}, p_j \in \mathbb{R},$$

se pueden recuperar a partir de los valores

$$\delta_{x_0}((d_x)^k f) = (d_x)^k f(x_0) = \sum_{s=0}^{k} \alpha_{k,s} f^{(s)}(x_0), \qquad k = 0, \ldots, 2M-1,$$

para ciertos $\alpha_{k,s} \in \mathbb{R}$.

Símbolo	Polinomio	Autovalor(λ_n)	p(x)	q(x)
$P_n^{(\alpha,\beta)}$	Jacobi	$-n(n+\alpha+\beta+1)$	$(1-x^2)$	$(\beta-\alpha-(\alpha+\beta+2)x)$
$C_n^{(\alpha)}$	Gegenbaue	$-n(n+2\alpha)$	$(1-x^2)$	$-(2\alpha+1)x$
P_n	Legendre	$-n(n+1)$	$(1-x^2)$	-2x
T_n	Chebyshev tipo 1	$-n^2$	$(1-x^2)$	-x
U_n	Chebyshev tipo 2	$-n(n+2)$	$(1-x^2)$	-3x
H_n	Hermite	-2n	1	-2x
$L_n^{(\alpha)}$	Laguerre	-n	x	$(\alpha+1-x)$

Tabla 3.1: Autofunciones operador Sturm-Liouville, autovalores y polinomios p,q.

3.2.2. Desarrollos finitos de polinomios ortogonales

Para este caso, consideramos de nuevo el espacio vectorial de funciones indefinidamente derivables $\mathcal{C}^\infty(\mathbb{R})$, polinomios p,q con $deg(p) = 2, deg(q) = 1$, y el operador

$$\begin{aligned} L_{p,q} : \mathcal{C}^\infty(\mathbb{R}) &\longrightarrow \mathcal{C}^\infty(\mathbb{R}), \\ f \ &\longmapsto L_{p,q}f : \mathbb{R} \longrightarrow \mathbb{R} \\ &\qquad x \longmapsto p(x)f''(x) + q(x)f'(x) \end{aligned} \tag{3.15}$$

denominado operador de Sturm-Liouville. Este operador ha sido estudiado de forma extensa, por sus aplicaciones en el campo de ecuaciones diferenciales, véase e.g. [9]. Se conoce que los polinomios en la Tabla 3.1, son autofunciones de (3.15), para los polinomios p y q indicados.

Fijando como familia de autofunciones alguna de las dadas en la Tabla 3.1 y denotada genéricamente por $\{\mathcal{Q}_n(x), n \in \mathbb{N}_0\}$, los correspondientes autovalores son únicos para cada tipo de polinomio, ya que $\lambda_n \neq \lambda_m$ si $n \neq m$. En este caso, siendo $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$,para un $x_0 \in \mathbb{R}$ tal que $\mathcal{Q}_n(x_0) \neq 0, n \in \mathbb{N}_0$, se considera la delta de Dirac en x_0 como funcional. En consecuencia, el polinomio

$$f(x) = \sum_{j=1}^{M} c_{n_j} \mathcal{Q}_{n_j}(x), \tag{3.16}$$

para ciertos $c_{n_j} \in \mathbb{C} - \{0\}, n_j \in \mathbb{N}_0$ tales que $0 \leq n_1 < \ldots < n_M = N$, con NM una cota adecuada, siendo N el grado del polinomio f en (3.16), se puede recuperar a partir de los valores

$$\begin{aligned}\delta_{x_0}(L_{p,q}^k f(x)) &= \sum_{j=1}^{M} c_{n_j} \delta_{x_0}(L_{p,q}^k \mathcal{Q}_{n_j}(x)) = \sum_{j=1}^{M} c_{n_j} \delta_{x_0}(\lambda_{n_j}^k \mathcal{Q}_{n_j}(x)) \\ &= \sum_{j=1}^{M} c_{n_j} \lambda_{n_j}^k \mathcal{Q}_{n_j}(x_0), \qquad k = 0, \ldots, 2M-1. \end{aligned} \tag{3.17}$$

El objetivo a continuación es ver que los valores (3.17) pueden obtenerse a partir de $f^{(m)}(x_0), m = 0, \ldots, 4M-2$, como consecuencia del siguiente resultado:

Teorema 3.1. *Sea $f \in \mathcal{C}^\infty(\mathbb{R})$, un polinomio de grado $N \in \mathbb{N}$, y $L_{p,q}$ en (3.15) el operador de Sturm-Liouville. Entonces para cada $x \in \mathbb{R}$, los valores $L_{p,q}^k(x), k = 0, \ldots, 2M-1$, pueden ser recuperados inequívocamente a partir de los datos $f^{(m)}(x), m = 0, \ldots, 4M-2$ mediante la siguiente igualdad:*

$$L_{p,q}^k f(x) = \sum_{l=1}^{2k} g_{l,k}(x) f^{(l)}(x), \qquad k \geq 1,$$

siendo los polinomios $g_{l,k}(x)$ aquellos que verifican la siguiente relación:

Para k=1: $g_{1,1}(x) = p(x)$ *y* $g_{2,1}(x) = q(x)$.

k≥2:

$$\begin{aligned} g_{l,k}(x) =& l\left(\frac{l+1}{2} p''(x) + q'(x)\right) g_{l,k-1}(x) \\ &+ \Big((l-1)p'(x) + q(x)\Big) g_{l-1,k-1}(x) + p(x) g_{l-2,k-1}(x), \end{aligned} \tag{3.18}$$

$l = 1, \ldots, 2k$.

Definiendo $g_{l,k}(x) := 0$, si $k \geq 1$ y $l \notin \{1, \ldots, 2k\}$.

Demostración. Como $\{\mathcal{Q}_n, n \in \mathbb{N}_0\}$ es una base del espacio de polinomios, podemos expresar f de la forma

$$f(x) = \sum_{n=0}^{N} \alpha_n \mathcal{Q}_n(x), \qquad \alpha_n \in \mathbb{C}, n = 0, \ldots, n.$$

Como estamos en el caso de una variable, y por definición teniamos

$$L_{p,q}h(x) = p(x)h^{'}(x) + q(x)h^{''}(x), \qquad h \in \mathcal{C}^{\infty}(\mathbb{R})$$

luego, podemos iterar este proceso y obtener

$$L_{p,q}^{k}h(x) = \sum_{l=0}^{2k} g_{l,k}(x)h^{(l)}(x), \qquad k \geq 1, \tag{3.19}$$

con $g_{l,k}(x)$, los polinomios correspondientes. Puesto que $\{\mathcal{Q}_n, n \in \mathbb{N}_0\}$ son autofunciones del operador $L_{p,q}$, verifican $L_{p,q}\mathcal{Q}_n = \lambda_n \mathcal{Q}_n$, que junto con la propiedad (3.19), nos lleva a la siguiente afirmación

$$\begin{aligned}
L_{p,q}^{k}\mathcal{Q}_n(x) =& L_{p,q}^{k-1}\Big(p(x)\mathcal{Q}_n^{''}(x) + q(x)\mathcal{Q}^{'}(x)\Big) = \\
&\overset{3.19}{=} \sum_{l=1}^{2k-2} g_{l,k-1}(x)\Big(p(x)\mathcal{Q}_n^{''}(x) + q(x)\mathcal{Q}^{'}(x)\Big)^{(l)} = \\
&\overset{(*)}{=} \sum_{l=1}^{2k-2} g_{l,k-1}(x)\Big(\binom{l}{0}p(x)\mathcal{Q}_n^{(l+2)}(x) + \binom{l}{1}p^{'}(x)\mathcal{Q}_n^{(l+1)}(x) \\
&\quad + \binom{l}{2}p^{''}(x)\mathcal{Q}_n^{(l)}(x) + \binom{l}{0}q(x)\mathcal{Q}^{(l+1)}(x) + \binom{l}{1}q^{'}(x)\mathcal{Q}^{(l)}(x)\Big) \\
&\overset{(**)}{=} \sum_{l=1}^{2k} g_{l,k}(x)\mathcal{Q}^{(l)}(x).
\end{aligned}$$

(∗) Se ha tenido en cuenta la regla de la derivada de un producto, y que $deg(p) = 2, deg(q) = 1$.

(∗∗) De esta igualdad se deduce la fórmula de recurrencia para los polinomios $g_{l,k}$ en (3.18).

A partir de la igualdad anterior,

$$\begin{aligned}
L_{p,q}^{k}f(x) = \sum_{n=0}^{N}\alpha_n L_{p,q}^{k}\mathcal{Q}_n(x) = \sum_{n=0}^{N}\alpha_n\Big(\sum_{l=1}^{2k} g_{l,k}(x)\mathcal{Q}_n^{(l)}(x)\Big) = \\
= \sum_{l=1}^{2k} g_{l,k}(x)\Big(\sum_{n=0}^{N}\alpha_n\mathcal{Q}_n^{(l)}\Big) = \sum_{l=1}^{2k} g_{l,k}(x)f^{(l)}(x).
\end{aligned} \tag{3.20}$$

□

Debido a esta construcción, el algoritmo para recuperar la función f en (3.16) sigue los siguentes pasos:

Algoritmo 3.2.2. Recuperación de desarrollos finitos de polinomios ortogonales.

Entradas: Autofunciones $\{\mathcal{Q}_n, n \in \mathbb{N}_0\}$, con autovalores $\{\lambda_n, n \in \mathbb{N}_0\}$ asociados, operador de Sturm-Liouville de parámetros $p(x), q(x)$, $M \in \mathbb{N}_0$, $x_0 \in \mathbb{R}$ tal que $\mathcal{Q}_n(x_0) \neq 0, \forall n \in \mathbb{N}_0$, $f^{(m)}(x_0), m = 0, \ldots, 4M-2$.

Paso 1) Construir la matriz

$$G = \begin{bmatrix} \tilde{g}_{1,1} & \cdots & \tilde{g}_{4M-2,1} \\ \vdots & \ddots & \vdots \\ \tilde{g}_{1,2M-1} & \cdots & \tilde{g}_{4M-2,2M-1} \end{bmatrix} \in \mathbb{R}_{(2M-1)\times(4M-2)},$$

siendo $\tilde{g}_{l,k} := g_{l,k}(x_0)$, las evaluaciones de los polinomios (3.18) en x_0. Nótese que la construcción de la matriz G depende únicamente de la base $\{\mathcal{Q}_n, n \in \mathbb{N}_0\}$ tomada.

Paso 2) Calcular el vector

$$\vec{h}_1 = G \begin{bmatrix} f'(x_0) \\ \vdots \\ f^{(4M-2)}(x_0) \end{bmatrix},$$

a partir de este, considerar el vector

$$\vec{h} := \begin{bmatrix} f(x_0) \\ \vec{h}_1 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ L_{p,q}f(x_0) \\ \vdots \\ L_{p,q}^{(2M-1)}f(x_0) \end{bmatrix},$$

por (3.20).

Paso 3) Denotando $\vec{h} = (h_l)_{l=0}^{2M-1}$, resolver el sistema de Hankel

$$\begin{bmatrix} h_0 & \ldots & h_{M-1} \\ \vdots & \ddots & \vdots \\ h_{M-1} & \ldots & h_{2M-2} \end{bmatrix} \begin{bmatrix} p_0 \\ \vdots \\ p_{M-1} \end{bmatrix} = - \begin{bmatrix} h_M \\ \vdots \\ h_{2M-1} \end{bmatrix},$$

con incógnitas $p_0, \ldots, p_{M-1}$.

Paso 4) Calcular los ceros $\lambda_{n_j}, j = 1, \ldots, M$, del polinomio de Prony

$$P(z) = z^M + \sum_{k=0}^{M-1} p_k z^k.$$

Paso 5) Hallar las autofunciones correspondientes a los autovalores $\lambda_{n_j}, j = 1, \ldots, M$, y obtener los coeficientes c_{n_j} en (3.16), resolviendo el sistema

$$\sum_{j=1}^{M} c_{n_j} \mathcal{Q}_{n_j}^{(l)}(x_0) = f^{(l)}(x_0), \qquad l = 0, \ldots, 2M-1.$$

Salidas: $n_j, c_{n_j}, j = 1, \ldots, M$.

■

3.2.3. Recuperación de vectores dispersos.

Para desarrollar este caso, es necesaria la definición de vectores dispersos.

Definición 3.4. *Sea $D, M \in \mathbb{N}$, y $\vec{x} \in \mathbb{C}^D$. Se dice que $\vec{x} = (x_1, \ldots, x_D)$ es un vector M-disperso si tiene exactamente M componentes no nulas.*

Nos planteamos el problema de recuperar un vector $\vec{x}$ M-disperso usando 2M muestras $y_k = a_k^T \vec{x}, k = 0, \ldots, 2M-1$, para ciertos vectores $a_k \in \mathbb{C}^D$. Podemos expresar $\vec{x}$ de la siguiente manera:

$$\vec{x} = \sum_{i=1}^{N} x_i e_i = \sum_{j=1}^{M} x_{n_j} e_j,$$

siendo $\{e_i\}_{i=1}^D$ los vectores de la base canónica y $\{x_{n_j}\}_{j=1}^M$ los coeficientes no nulos del vector $\vec{x}$. Se considera la aplicación lineal $\mathcal{D} : \mathbb{C}^D \longrightarrow \mathbb{C}^D$ con matriz

$$\mathbf{D} := \begin{bmatrix} d_1 & 0 & \ldots & 0 \\ 0 & d_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & d_D \end{bmatrix},$$

con $\{d_i\}_{i=1}^D \subset \mathbb{C}$ distintos. Los autovectores de este operador, son los de la base canónica $\{e_i\}_{i=1}^D$, ya que

$$\mathbf{D}e_i = d_i e_i, \qquad i = 1, \ldots, D.$$

Por último, tomamos como funcional

$$\begin{aligned} \mathbf{F} : \mathbb{C}^D &\longrightarrow \mathbb{C}^D \\ \vec{y} &\longmapsto \mathbf{F}\vec{y} = \vec{b}^T\vec{y} = \sum_{i=1}^{D} b_j y_j, \end{aligned}$$

con $\vec{b} = (b_1, \ldots, b_D) \in \mathbb{C}^D$, verificando $b_i \neq 0, i = 1, \ldots, D$, cuya elección depende de cada caso particular. Con esta elección de $\mathbf{F}$, es claro que $\mathbf{F}e_i \neq 0, i = 1, \ldots, D$. Por lo tanto, el vector $\vec{x}$ se puede recuperar a partir de los valores

$$a_k^T\vec{x} = \mathbf{F}(\mathbf{D}^k\vec{x}) = \vec{b}^T\mathbf{D}^k\vec{x}, \qquad a_k^T = \vec{b}^T\mathbf{D}^k = (b_1 d_1^k, \ldots, b_D d_D^k), \tag{3.21}$$

$k = 0, \ldots, 2M-1$. En la práctica, los datos son medidos con ruido, y no se conoce la dispersión (M) del vector $\vec{x}$, por lo que es necesario adaptar alguno de los métodos anteriores. En particular, aplicando el método ESPRIT obtenemos el siguiente algoritmo:

Algoritmo 3.2.3. Método ESPRIT para recuperar vectores dispersos.

Entradas: $L, N \in \mathbb{N}$, con L una cota adecuada de M tal que $M \geq L \geq N$. Valores $\mathbf{F}(\mathbf{A}^k\vec{x}), k = 0, \ldots, 2N-1$, en (3.21).

Paso 1) Hallar la descomposición en valores singulares (SVD) de la matriz de Hankel

$$H_{2N-l,L+1} := \begin{bmatrix} \mathbf{F}(\vec{x}) & \ldots & \mathbf{F}(\mathbf{A}^L\vec{x}) \\ \vdots & \ddots & \vdots \\ \mathbf{F}(\mathbf{A}^{2N-L-1}\vec{x}) & \ldots & \mathbf{F}(\mathbf{A}^{2N-1}\vec{x}) \end{bmatrix},$$

y su rango numérico. A partir de su descomposición en valores singulares calcular las matrices $W_{M,L}(s), s = 0, 1$, como en (2.37).

Paso 2) Obtener los autovalores $\lambda_j, j = 1, \ldots, M$, de la matriz

$$F_M^{SVD} = (W_{M,L}(0)^T)^+ W_{M,L}(1)^T.$$

Hallar los autovectores a los que corresponden estos autovalores, es decir, obtener los $\{n_j\}_{j=1}^M \subset \{1, \ldots, N\}$, tal que $\mathbf{A}e_{n_j} = \lambda_j e_{n_j}$.

Paso 3) Los coeficientes $x_{n_j}, j = 1, \ldots, M$ se obtienen como solución por mínimos cuadrados del sistema:

$$\sum_{j=1}^{M} x_{n_j} b_{n_j} d_{n_j}^k = \mathbf{F}(\mathbf{A}^k \vec{x}).$$

Salidas: $M \in \mathbb{N}, \vec{x}$.

■

3.2.4. Recuperación de splines a partir de la transformada de Fourier.

En este caso, el problema a resolver mediante métodos de Prony es uno muy interesante y que aparece frecuentemente en diversos campos científicos, que es la recuperación de funciones de soporte compacto a partir de datos de su transformada de Fourier, e.g. [19]. Para desarrollar este problema, utilizaremos las siguientes definiciones:

Definición 3.5 (Transformada de Fourier)**.** *Sea $f \in \mathcal{L}^1(\mathbb{R})$, una función integrable. Entonces $\hat{f} : \mathbb{R} \longrightarrow \mathbb{C}$ con*

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i\xi x} dx, \qquad \xi \in \mathbb{R}, i = \sqrt{-1},$$

se llama transformada de Fourier de f.

Definición 3.6 (Spline)**.** *Sea $m, n \in \mathbb{N}$, y sean $\{t_j\}_{j=1}^{m+n} \subset \mathbb{R}$, tal que $t_1 < t_2 < \ldots < t_{m+n}$. Decimos que $s : \mathbb{R} \longrightarrow \mathbb{R}$ es un spline de grado m con soporte en $[t_1, t_{m+n}]$ y nodos $\{t_j\}_{j=1}^{m+n-1}$, si verifican las siguientes condiciones:*

1: *$s(x) = 0$, si $x \in (-\infty, t_1) \cup (t_{m+n}, \infty)$*

2: *$s\big|_{[t_j,t_{j+1})}$ es un polinomio de grado menor o igual que m-1, para $j = 1,\dots,m+n-1$.*

3: *$s \in \mathcal{C}^{m-2}(\mathbb{R})$ (donde definimos $\mathcal{C}^0(\mathbb{R}) := \mathcal{C}(\mathbb{R})$, y $\mathcal{C}^{-1}(\mathbb{R})$ el conjunto de funciones continuas a trozos).*

El primer paso será considerar los splines de grado $m = 1$, es decir, las funciones escalonadas, dadas por

$$s(x) = \sum_{j=1}^{n} c_j \chi_{[t_j,t_{j+1})}(x), \qquad x \in \mathbb{R}, \tag{3.22}$$

Siendo $\chi_{[t_j,t_{j+1})}$ la función característica del intervalo $[t_j, t_{j+1})$ y $c_j \in \mathbb{R}$ distintos. Aplicando la transformada de Fourier a esta función, resulta

$$\begin{aligned}\hat{s}(\xi) &= \int_{-\infty}^{\infty} \left(\sum_{j=1}^{n} c_j \chi_{[t_j,t_{j+1})}(x) \right) e^{-ix\xi} dx = \sum_{j=1}^{n} c_j \left(\int_{-\infty}^{\infty} \chi_{[t_j,t_{j+1})}(x) e^{-ix\xi} dx \right) = \\ &= \sum_{j=1}^{n} c_j \left(\int_{t_j}^{t_{j+1}} e^{-ix\xi} dx \right) = \sum_{j=1}^{n} \frac{c_j}{i\xi} \left(e^{-i\xi t_j} - e^{i\xi t_{j+1}} \right) = \\ &= \sum_{j=1}^{n+1} \frac{1}{i\xi} (c_j - c_{j-1}) e^{-i\xi t_j},\end{aligned}$$

denotando $c_0 = c_{n+1} := 0, c_j^1 := c_j - c_{j-1}, j = 1,\dots,n+1$, obtenemos finalmente que

$$f(\xi) := i\xi \hat{s}(\xi) = \sum_{j=1}^{n+1} c_j^1 e^{-i\xi t_j}, \tag{3.23}$$

es una suma de exponenciales complejas (fijamos $f(0) := 0$). Para poder recuperar la función s, será necesario suponer que existe $\tau > 0$ tal que $t_j\tau \in [-\pi, \pi), j = 1,\dots,n+1$. Puesto que s es una función real, por construcción, se verifica

$$f(\xi) = \overline{f(-\xi)}.$$

A partir de los valores $\hat{s}(k\tau), k = 1,\dots,N$, para una cota adecuada $N \geq n+1$, se pueden reconstruir $2N+1$ valores de f, de la siguiente manera:

$$f(l\tau) := \begin{cases} il\tau \hat{s}(l\tau) & l = 1,\dots,N, \\ \overline{f(-l\tau)} & l = -N,\dots,-1, \\ 0 & l = 0. \end{cases} \tag{3.24}$$

El siguiente paso es considerar la función definida por

$$\tilde{f}(\xi) := f((-N+\xi)\tau) = \sum_{j=1}^{n+1}(c_j^1 e^{iN\tau})e^{i\xi(-\tau t_j)},$$

que verifica $\tilde{f}(k) = f((-N+k)\tau), k = 0, \ldots, 2N$, los cuales son conocidos. Por ser $\tilde{f}$ una suma de exponenciales complejas, podemos emplear alguno de los métodos vistos en el capítulo 2, para recuperar los valores t_j y $c_j^1, j = 1, \ldots, n+1$. Finalmente, los coeficientes c_j en (3.22) se recuperan mediante la recurrencia

$$c_1 := c_1^1 \text{ y } c_j := c_{j-1} + c_j^1, j = 2, \ldots, n.$$

Teniendo en cuenta este caso, la generalización a splines de mayor grado sigue los siguientes pasos. Para $m \in \mathbb{N}$, se definen los B-splines de la siguiente manera

Definición 3.7 (B-spline)**.** *Sea $m, n \in \mathbb{N}_0$, y $\{t_j\}_{j=1}^{m+n} \subset \mathbb{R}$, con $t_1 < \ldots < t_{m+n}$. Los B-splines de grado m en los nodos $t_1, \ldots, t_{m+n}$, denotados por B_j^m se definen mediante la siguiente relación de recurrencia:*

$$B_j^m(x) := \frac{x - t_j}{t_{j+m-1} - t_j} B_j^{m-1} + \frac{t_{j+1} - x}{t_{j+m} - t_{j+1}} B_{j+1}^{m-1}, \qquad j = 1, \ldots, n$$

con $B_j^1(x) := \chi_{[t_j, t_{j+1})}(x)$.

Fijando $t_1, \ldots, t_{m+n}$, los B-splines en dichos nodos son conocidas las siguientes propiedades, véase e.g. [5]:

1: $Sop(B_j^m) = [t_j, t_{j+m}], j = 1, \ldots, n.$

2: El conjunto $\{B_1^m, \ldots, B_n^m\}$ es una base del espacio de splines de grado m en los nodos $t_1, \ldots, t_{m+n}$.

3: Para $m \geq 3$, se verifica

$$(B_j^m)'(x) = (m-1)\left(\frac{B_j^{m-1}(x)}{t_{j+m-1} - t_j} - \frac{B_{j+1}^{m-1}(x)}{t_{j+m} - t_{j+1}}\right), \qquad j = 1, \ldots, n. \tag{3.25}$$

Teniendo esto en cuenta, todo spline de grado m se puede expresar como

$$s(x) := \sum_{j=1}^{n} c_j B_j^m(x), \qquad x \in \mathbb{R}, \tag{3.26}$$

para determinados nodos $t_1, \ldots, t_{m+n}$. A partir de la propiedad (3.25), se deduce

$$s^{(k)}(x) = \sum_{j=1}^{n} c_j (B_j^m)^{(k)}(x) = \sum_{j=1}^{n+k} c_j^{m-k} B_j^{m-k}(x), \qquad k = 1, \ldots, m-1, \tag{3.27}$$

con

$$c_j^{l+1} := \begin{cases} c_j^0 + c_{j-1}^1 & l = 0, j = 1, \ldots, n+m-1 \\ \frac{t_{m+1-k} - t_j}{m-k} c_j^l + c_{j-1}^{l+1} & l = 1, \ldots, m-1, j = 1, \ldots, n+m-l-1, \end{cases} \tag{3.28}$$

fijando $c_0^l := 0, l = 1, \ldots, m$. En particular, para $k = m-1$, se tiene

$$s^{(m-1)}(x) = \sum_{j=1}^{n+m-1} c_j^1 B_j^1(x) = \sum_{j=1}^{n+m-1} c_j^1 \chi_{[t_j, t_{j+1})}.$$

De forma similar al caso anterior, aplicando la transformada de Fourier al spline s en (3.26),

$$\hat{s}(\xi) = \frac{1}{(i\xi)^{m-1}} \sum_{j=1}^{n+m-1} \frac{c_j^1}{i\xi} (e^{-i\xi t_j} - e^{-i\xi t_{j+1}}) = \frac{1}{(i\xi)^m} \sum_{j=1}^{n+m} c_j^0 e^{-i\xi t_j}, \tag{3.29}$$

con $c_j^0 := c_j^1 - c_{j-1}^1, j = 1, \ldots, n+m$, y $c_0^1 = c_{n+m}^1 := 0$. A partir de (3.29) deducimos que $c_j^0 \neq 0, j = 1, \ldots, n+m$, ya que si no fuese así, alguno de los c_j en (3.26) sería nulo, y bastaria considerar s como un spline de grado s-1.

Como ocurría con los splines de grado 1, suponiendo que existe $\tau > 0$ tal que $t_j \tau \in [-\pi, \pi), j = 1, \ldots, n+m$, basta considerar la función

$$f(\xi) := (i\xi)^m \hat{s}(\xi),$$

que es una suma de exponenciales complejas de orden $n+m$, por (3.29) y aplicando un razonamiento totalmente simétrico al caso anterior, podemos recuperar los nodos t_j y los coeficientes $c_j^0, j = 1, \ldots, n+m$ a partir de los datos $\hat{s}(l\tau), l = 1, \ldots, N$, con $N \geq n+m$ una cota adecuada.

Nota: Si el orden de s en (3.26) es desconocido, conociendo la cota $N \geq n$ y los datos de su transformada de Fourier correspondiente, la reconstrucción seguiría siendo válida, aplicando los métodos de Prony basados en la descomposición QR/ESPRIT.

Un paso más en este método, se basa en recuperar funciones de la forma

$$f(x) = \sum_{j=1}^{n} c_j \Phi(x + t_j) \qquad x \in \mathbb{R}, , \tag{3.30}$$

con $c_j \neq 0, j = 1, \ldots, n$, para ciertos parámetros $\{t_j\}_{j=1}^n \subset \mathbb{R}$ y $\Phi \in \mathcal{C}(\mathbb{R}) \cap L^1(\mathbb{R})$, tal que para $T \in \mathbb{R}$, $|\hat{\Phi}(\xi)| > \varepsilon > 0, \xi \in (-T, T)$. Ha de suponerse, de nuevo, que existe $\tau > 0$ tal que $|t_j \tau| < \text{mín}\{\pi, T\}, j = 1, \ldots, n$. Aplicando la transformada de Fourier a la función (3.30),

$$\begin{aligned}
\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i\xi x} dx = \int_{-\infty}^{\infty} \left(\sum_{j=1}^{n} c_j \Phi(x + t_j) \right) e^{-i\xi x} dx = \\
= \sum_{j=1}^{n} c_j \left(\int_{-\infty}^{\infty} \Phi(x + t_j) e^{-i\xi x} dx \right) = \sum_{j=1}^{n} c_j \left(\int_{-\infty}^{\infty} \Phi(y) e^{-i\xi(y - t_j)} dy \right) = \\
= \hat{\Phi}(\xi) \sum_{j=1}^{n} c_j e^{i\xi t_j)},
\end{aligned}$$

haciendo el cambio de variable $y = x + t_j$ en cada integral del sumatorio. Por lo tanto, considerando la función

$$h(\xi) = \frac{\hat{f}(\xi)}{\hat{\Phi}(\xi)} = \sum_{j=1}^{n} c_j e^{i\xi t_j}, \qquad \xi \in (-T, T),$$

(está bien definida pues $|\hat{\Phi}(\xi)| > \varepsilon > 0, \xi \in (-T, T)$), obtenemos de nuevo una suma de exponenciales complejas. Haciendo un razonamiento análogo al caso de splines, se deduce que la función f en (3.30) se puede reconstruir, a partir de los valores $h(-\tau N + k\tau), k = 0, \ldots, 2N$, para cierta cota adecuada $N \geq n$.

3.3. Experimentos numéricos.

Para finalizar esta sección, de manera análoga al capítulo 2, se expondrán varios experimentos interesantes que ejemplifican la teoría desarrollada.

3.3.1. Experimento 1.

En este caso se trata de recuperar un vector 9-disperso a partir de sus datos de Fourier. Los parámetros considerados son los siguientes:

- $M = 9$.
- $D = 1024$.
- Operador $\mathbf{A} = diag(w_D^{\sigma*0}, \ldots, w_D^{\sigma*(D-1)}), w_D := e^{\frac{-2\pi i}{D}}$.
- Valores no-nulos:
  $(x_1, x_5, x_9, x_{19}, x_{42}, x_{45}, x_{71}, x_{115}, x_{132}) = (7, 5, -7, 3, 10, 5, -5, 7, -5)$.
- $\varepsilon = 0{,}0005$.
- $y_k = \hat{x}_{\sigma k} + e_k, k = 0, \ldots, 2N-1$, como en (3.21) (Se denota con $\hat{}$, ya que en este caso se corresponde con la transformada de Fourier discreta), siendo e_k errores de medición, tomados de forma uniforme en el intervalo $[-\gamma, \gamma], \gamma \in \mathbb{R}$.
- $\mathbf{F}$=Id.

Los resultados obtenidos se muestran en la Tabla 3.2.

$N = L$	γ	$\widetilde{M}$	σ	Posiciones no nulas								
10	0	4	1	1				42			100	136
30	0	6	1	1			15	43		71	115	132
50	0	8	1	1		9	19	42	45	71	115	132
70	0	9	1	1	5	9	19	42	45	71	115	132
10	1	4	1	0				42			100	136
30	1	6	1	1			16	43		71	115	132
50	1	8	1	1		9	19	41	44	71	115	132
90	1	9	1	1	5	9	19	42	45	71	115	132
11	0	9	7	1	5	9	19	42	45	71	115	132
Empleando la función rank para estimar M.												
14	0	9	1	1	5	9	19	42	45	71	115	132

Tabla 3.2: Experimento 1. Recuperación de las posiciones no nulas.

A la vista de los resultados, es claro que cuantos más datos de Fourier poseamos, mayor será la precisión al estimar las posiciones, y que a mayor error de medición, mayor cantidad de datos necesitados. También es destacable que una correcta elección de σ puede reducir enormemente la cantidad de datos necesarios, ya que por ejemplo, en este caso pasamos de necesitar $N = 70$ para $\sigma = 1$ a $N = 11$ para $\sigma = 7$. Como ya ocurría en los experimentos anteriores, la aproximación del rango de una matriz se calcula de forma más eficiente con la función rank de Matlab, en ausencia de ruido, sin embargo la forma desarrollada en la teoría es más estable cuando hay errores de medición.

Los siguientes experimentos se basan en la recuperación de desarrollos en funciones Gausianas de frecuencia modulada, estos tipos de funciones son muy usadas en campos como teoría de la señal, por lo que será interesante poder recuperarlas de una manera eficiente, empleando los algoritmos desarrollados de una forma concreta (véase e.g. [17]).

3.3.2. Experimento 2.

En este experimento, los desarrollos son del tipo siguiente:

$$f(x) = \sum_{j=1}^{M} c_j g(x - \alpha_j), \tag{3.31}$$

$M \in \mathbb{N}$, y $g(x) = e^{-\beta x^2}, \beta \in \mathbb{C}\backslash\{0\}, \alpha_j \in \mathbb{R}, c_j \in \mathbb{C}$. Este último tipo de funciones, con β complejo, recibe el nombre de función Gausiana de frecuencia modulada. La elección adecuada es escoger como funcional la identidad, y el operador de tipo "shift"siguiente:

$$S_{g,h} f(x) := \frac{g(x)}{g(x+h)} f(x+h) = e^{\beta h(2x+h)} f(x+h), \qquad h \in \mathbb{R}\backslash\{0\}.$$

En (3.31) hay que hacer diferentes consideraciones, en función del β elegido en la función Gausiana. Por un lado, si $Re(\beta) = 0$, tomaremos h tal que $0 < h \leq \frac{\pi}{2|Im(\beta)|T}$, con T tal que $\alpha_j \in (-T, T), \forall j$. En otro caso, h se puede tomar de forma arbitraria. En estas condiciones, las funciones $v_j := g(x - \alpha_j), j = 1, \ldots, M$, son autofunciones del operador $S_{g,h}$, pues

$$S_{g,h} v_j(x) = e^{\beta h(2x+h)} e^{-\beta(x+h-\alpha_j)^2} = e^{2\beta h \alpha_j} e^{-\beta(x-\alpha_j)^2} = e^{2\beta h \alpha_j} v_j(x),$$

y por lo tanto (3.31) se puede recuperar a partir de los valores $f(x_0+hk), k = 0,\dots,2M-1$. El procedimiento a seguir será el siguiente. Por definición, podemos expresar f en (3.31) de la siguiente forma,

$$f(x) = \sum_{j=1}^{M} c_j e^{-\beta(x-\alpha_j)^2} = \sum_{j=1}^{M} (c_j g(\alpha_j)) g(x) e^{2\beta\alpha_j x} = \sum_{j=1}^{M} \tilde{c}_j g(x) e^{\tilde{\alpha}_j x},$$

con $\tilde{c}_j := g(\alpha_j)c_j$, y $\tilde{\alpha}_j = 2\beta\alpha_j$. El siguiente paso será considerar la función

$$\tilde{f}(x) := g^{-1}(x)f(x) = \sum_{j=1}^{M} \tilde{c}_j e^{\tilde{\alpha}_j x},$$

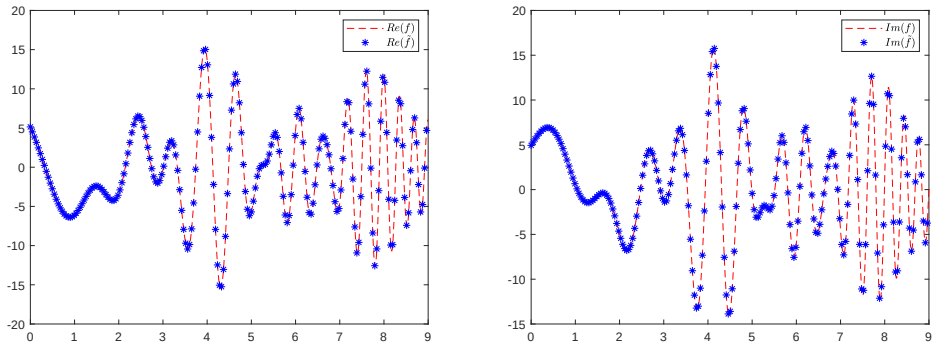
la cual sabemos recuperar, mediante el método clásico de Prony, aplicado a los valores $\tilde{f}(x_0+kh) = g^{-1}(x_0+kh)f(x_0+kh), k = 0,...,2M-1$. Por último, basta recuperar los valores originales, mediante $\alpha_j = \frac{\tilde{\alpha}_j}{2\beta}, c_j = g^{-1}(\alpha_j)\tilde{c}_j$.

En particular, consideramos los parámetros $M = 5, \beta = -i, g(x) = e^{ix^2}, h = 1$, y $\alpha_j, c_j, j = 1,\dots,5$ en Tabla 3.3. Estos últimos se han tomado de forma aleatoria y uniforme en los intervalos $\alpha_j \in (-\frac{\pi}{2}, \frac{\pi}{2})$ y $c_j \in (-5,5) + i(-2,2)$. Se han empleado los parámetros $f(-1+k), k = 0,\dots,9$. Como resultado, obtenemos que los errores para las aproximaciones $\tilde{\alpha}_j, \tilde{c}_j$, son

$$\max_{j=1,\dots,5} |\alpha_j - \tilde{\alpha}_j| = 1{,}8722e-13, \qquad \max_{j=1,\dots,5} |c_j - \tilde{c}_j| = 3{,}4824e-12.$$

j	α_j	c_j
1	0,80971	$-3{,}5811e+00+6{,}2296e-01i$
2	0,76382	$-7{,}8239e-01-1{,}8572e+00i$
3	−0,33858	$4{,}1574e+00+1{,}3965e+00i$
4	0,48845	$2{,}9221e+00+1{,}7360e+00i$
5	−1,0330	$4{,}5949e+00+7{,}1494e-01i$

Tabla 3.3: Experimento 2. Parámetros del desarrollo (3.31).



(a) Parte real de f y $\tilde{f}$. (b) Parte imaginaria de f y $\tilde{f}$.

Figura 3.1: Experimento 2. Comparación de f y $\tilde{f}$.

3.3.3. Experimento 3.

Las funciones consideradas en este ejemplo, reciben el nombre de Gausianas moduladas, y son muy similares a las anteriores, pero añadiendo un factor de modulación. Se pueden expresar de la siguiente manera:

$$f(x) = \sum_{j=1}^{M} c_j e^{2\pi i \alpha_j x} g(x - s_j), \tag{3.32}$$

$M \in \mathbb{N}$, y $g(x) = e^{-\beta x^2}, \beta \in \mathbb{R}^+, \alpha_j \in [0, 1), c_j \in \mathbb{C}\backslash\{0\}, s_j \in \mathbb{R}$. Tomando de nuevo la identidad como funcional y el operador $S_{g,h}$, definido en el ejemplo anterior, las funciones $v_j(x) = e^{2\pi i \alpha_j x} g(x - s_j)$, son autofunciones del operador, ya que,

$$S_{g,h} v_j(x) = e^{\beta h(2x+h)} e^{2\pi i \alpha_j (x+h)} e^{-\beta(x+h-s_j)^2} = e^{2h(\beta s_j + \pi i \alpha_j)} v_j(x).$$

Para el correcto desarrollo del método, será necesario suponer que $\beta s_j + \pi i \alpha_j$, son distintos dos a dos, y $0 < h \leq \frac{1}{2}$. De forma similar al experimento 2, podemos expresar (3.32), en la forma

$$f(x) = \sum_{j=1}^{M} c_j e^{2\pi i \alpha_j x} e^{-\beta(x-s_j)^2} = \sum_{j=1}^{M} (c_j g(s_j)) g(x) e^{(2\pi i \alpha_j + 2\beta s_j)x} =$$
$$= \sum_{j=1}^{M} \tilde{c}_j g(x) e^{\tilde{\alpha}_j x},$$

con $\tilde{c}_j := g(s_j)c_j$, y $\tilde{\alpha}_j = 2\pi i\alpha_j + 2\beta s_j$. Y de nuevo reconstruimos la función

$$\tilde{f}(x) := g^{-1}(x)f(x) = \sum_{j=1}^{M} \tilde{c}_j e^{\tilde{\alpha}_j x},$$

a partir de los valores $\tilde{f}(x_0+kh) = g^{-1}(x_0+kh)f(x_0+kh), k = 0, ..., 2M-1$. Por último, deshaciendo el cambio de variable, obtenemos, $\alpha_j = \frac{Im(\tilde{\alpha}_j)}{2\pi}$, $s_j = \frac{Re(\tilde{\alpha}_j)}{2\beta}$, y $c_j = g^{-1}(s_j)\tilde{c}_j$.

Para un ejemplo concreto, tomamos los parámetros $M = 6, \beta = \frac{1}{2}, g(x) = e^{-\frac{x^2}{2}}, h = \frac{1}{2}$ y los coeficientes $\alpha_j \in (0,1), c_j \in (-10,10)$ y $s_j \in (-5,5)$, en Tabla 3.4. Se han empleado las mediciones $f(\frac{1}{2}k), k = 0, \ldots, 11$, obteniendose los errores:

$$\max_{j=1,\ldots,5} |\alpha_j - \tilde{\alpha}_j| = 1{,}0240e-10, \qquad \max_{j=1,\ldots,5} |c_j - \tilde{c}_j| = 5{,}3198e-11$$

$$\max_{j=1,\ldots,5} |s_j - \tilde{s}_j| = 2{,}3719e-10.$$

j	α_j	c_j	s_j
1	0,8176	−0,2642	−1,4927
2	0,7948	−1,2828	4,3900
3	0,6443	−1,0643	3,7594
4	0,3786	−3,8730	0,5016
5	0,8116	0,1702	1,2248
6	0,5328	0,2154	0,8704

Tabla 3.4: Experimento 3. Parámetros del desarrollo (3.32).

Bibliografía

[1] *R. Badeau, B. David, G. Richard, IEEE Trans. Signal Process. 54, 450-458 (2006).*

[2] *D. Batenkov, Y. Yomdin, Math. Comput. 81, 277-318 (2012).*

[3] *F.S. V. Bazán and P.L Toint, Mech. Systems Signal Process. 15, 667-683 (2001).*

[4] *G. Beylkin, L. Monzón, On approximation of functions by exponential sums revisited, Appl. Comput. Harnon. Anal. 28 (2010) 131-149.*

[5] *C. de Boor, A practical Guide to Splines, Springer, 2001.*

[6] *E.J. Candès, J. Romberg, T. Tao, Robust uncertainity principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inform. Theory 52 (2006) 489-505.*

[7] *W. Demmel, Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics, (1997).*

[8] *P.L. Dragotti, M. Vetterli, and T. Blu, IEEE Trans. Signal Process. 55, 1741-1757 (2007).*

[9] *P. Hartman, Ordinary Differential Equations, SIAM Philadelphia, (2002).*

[10] *V.K Ingle, S.M. Kogon, and D.G. Manolakis, Statisical and adaptive signal processing (Artech, 2005).*

[11] *D. Kalman, The Generalized Vandermonde Matrix. Mathematics Magazine. 57 (1): 15–21 (1984).*

[12] *I. Markovsky, Structured low-rank approximation and its applications. Automatica, 44(4), 891-909 (2008).*

[13] *A.V. Oppenheim, A.S. Willsky, S.H. Nawab, Signals & systems. Pearson Educación, (1997).*

[14] *V. Pereyra, G. Scherer, In: Exponential Data Fitting and its Applications, Bentham Sci. Publ. ,pp. 1-26 (2010).*

[15] *T. Peter, G. Plonka, A generalized Prony method for reconstruction of sparse sums of eigenfunctions of linear operators, Inverse Problems, 29 025001 (2013).*

[16] *P.P. Petrushev, V.A. Popov, Rational approximation of real functions, C.U.P., (2013).*

[17] *G. Plonka, , K. Stampfer, I. Keller, Reconstruction of stationary and non-stationary signals by the generalized Prony method. Analysis and Applications, 17(02), 179-210, (2019).*

[18] *G. Plonka, M. Tasche, Prony methods for recovery of structured functions, GAMM-Mitteilungen, 37(2) (2014), 239-258.*

[19] *G. Plonka, M. Wischerhoff, How many Fourier samples are needed for real function reconstruction, J. Appl. Math. Comput. , 42, 117-137 (2013).*

[20] *D. Potts, M. Tasche, Parameter estimation for exponential sums by approximate Prony method, Signal Process. 90 (2010).*

[21] *J.M. Sanz-Serna, Diez lecciones de cálculo numérico. Universidad de Valladolid, Secretariado de Publicaciones e Intercambio Editorial, (2010).*

[22] *M. Vetterli, P. Marziliano, T. Blu, IEEE Trans. Signal Process. 50, 1417-1428 (2002).*

[23] *L. Weiss, R.N. McDonough, Prony's method, Z-transforms, and Padé approximation. SIAM Review 5(2) (1963): 145-149.*

Índice de figuras

Lista de Tablas

Apéndice A

El problema de autovalores generalizado.

El problema de autovalores generalizado trata de extender la definición clásica de autovalores, a un espectro más amplio de matrices.
Dada una matriz $A \in \mathbb{K}_{n\times n}$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, y $n \in \mathbb{N}$, el problema clásico de autovalores pretende encontrar escalares λ que verifiquen $det(A - \lambda Id_n) = 0$.
Aquellos que verifiquen dicha condición reciben el nombre de autovalores.
Sin embargo, en determinados contextos, esta definición puede ser bastante restrictiva, por lo que el problema de autovalores generalizados trata de ampliar este concepto. Para ello definiremos en primer lugar los lápices matriciales, e.g. [7].

Definición A.1. *Sean $A, B \in \mathbb{K}_{m\times n}$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, y $m, n \in \mathbb{N}, m \geq n$. La matriz $A - \lambda B$, recibe el nombre de lápiz matricial. Se toma λ como una variable indeterminada.*
Además, si $m = n$, y existe al menos un $\lambda_0 \in \mathbb{K}$ tal que $det(A - \lambda_0 B) \neq 0$, se dice que el lápiz matricial $A - \lambda B$ es regular; en cualquier otro caso, se denomina singular.
Para $A - \lambda B$ regular, se define su polinomio característico como $p(\lambda) := det(A - \lambda B)$, y sus autovalores generalizados, como:

(1) *Las raices de $p(\lambda)$.*

(2) *∞ (con multiplicidad $n - deg(p)$) si $deg(p) < n$.*

Un resultado importante, y que se puede generalizar al caso de lápices matriciales singulares, es el siguiente:

Proposición A.1. *Sea $A - \lambda B$ un lápiz matricial regular. Si B es regular, todos los autovalores generalizados de $A - \lambda B$ son finitos, y los mismos que los autovalores en el sentido clásico, de $AB^{-1} o B^{-1}A$. Si B es singular, $A - \lambda B$ tiene como autovalor a ∞ con multiplicidad $n - rg(B)$. Si A es regular, los autovalores generalizados de $A - \lambda B$ son los autovalores inversos de $A^{-1}B$ ó BA^{-1}, donde un autovalor nulo de $A^{-1}B$ corresponde con un autovalor ∞ de $A - \lambda B$.*

Demostración. Si B es regular, y λ_0 es un autovalor suyo, entonces por definición

$$0 = det(A - \lambda_0 B) = det(AB^{-1} - \lambda_0 Id_n) = det(B^{-1}A - \lambda_0 Id_n),$$

por lo tanto, λ_0 es autovalor de AB^{-1} y de $B^{-1}A$.
Si B es singular, consideramos su descomposición SVD, tal que $B = U\Sigma V$, y obtenemos

$$p(\lambda) = det(A - \lambda U\Sigma V) = det(U(U^*AV - \lambda\Sigma)V^*) = \pm det(U^*AV - \lambda\Sigma).$$

Puesto que $r := rg(B) = rg(\Sigma)$, habrá un total de r λs en la matriz $U * AV - \lambda\Sigma$, en consecuencia $gr(p) = r$.
Si A es regular, $det(A - \lambda B) = 0 \Leftrightarrow det(Id_n - \lambda A^{-1}B) = 0$ ó $det(Id_n - \lambda BA^{-1}) = 0$. Y esto solo ocurre, si $\lambda = 0 y 1/\lambda$ son autovalores de $A^{-1}B$ ó BA^{-1}. □

De forma similar, se definen los autovectores generalizados para lápices matriciales regulares,

Definición A.2. *Sea $A - \lambda B$ un lápiz matricial regular, con A,$B \in \mathbb{C}_{n\times n}$, y λ_0 un autovalor generalizado suyo. Se dice que $\vec{v} \in \mathbb{C}^n$ es un autovector generalizado por la derecha, asociado a λ_0, si $(A - \lambda_0 B)\vec{v} = \vec{0}$.*
Si $\lambda_0 = \infty$, y $B\vec{v} = 0$, entonces $\vec{v}$ es un autovalor generalizado por la derecha, asociado a ∞.
Se dice que λ_0 es autovalor generalizado por la izquierda de $A - \lambda B$, si es un autovalor generalizado por la derecha de $(A - \lambda B)^*$, donde * denota la transpuesta conjugada.

Otro concepto importante es el de lápices matriciales equivalentes.

Definición A.3. *Se dice que los lápices matriciales $A - \lambda B$ y $C - \lambda D$ son equivalentes, si existen matrices regulares P_1 y P_2, tales que $C - \lambda D = P_1 A P_2 - \lambda P_1 B P_2$.*

A partir de esta definición se deduce el siguiente resultado.

Proposición A.2. *Dados dos lápices matriciales equivalentes $A - \lambda B$ y $C - \lambda D$, se verifica:*

(1) *$A - \lambda B$ y $C - \lambda D$ tienen los mismos autovalores generalizados.*

(2) *$\vec{v}$ es autovector generalizado por la derecha de $A - \lambda B$, si y solo si, $P_2^{-1}\vec{v}$ lo es de $C - \lambda D$.*

(3) *$\vec{w}$ es autovector generalizado por la izquierda de $A - \lambda B$, si y solo si, $(P_1^*)^{-1}\vec{v}$ lo es de $C - \lambda D$.*

Para este tipo de matrices, tambien existe una forma canónica, cuyo objetivo es generalizar la forma canónica de Jordan a lápices matriciales regulares.

Teorema A.1 (Forma canónica de Weierestrass)**.** *Sea $A - \lambda B$ un lápiz matricial regular. Entonces existen matrices regulares $P_1 y P_2$, tales que*

$$P_1(A - \lambda B)P_2 = diag(J_{n_1}(\lambda_1) - \lambda Id_{n_1}, \ldots, J_{n_k}(\lambda_k) - \lambda Id_{n_k}, N_{m_1}, \ldots, N_{m_r}),$$

con $J_{n_i}(\lambda_i)$ un bloque de Jordan del autovalor λ_i,

$$J_{n_i}(\lambda_i) := \begin{bmatrix} \lambda_i & 1 & 0 & \ldots & 0 \\ 0 & \lambda_i & 1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 1 \\ 0 & 0 & \ldots & 0 & \lambda_i \end{bmatrix} \in \mathbb{C}_{n_i \times n_i},$$

y N_{m_i}, es un bloque de Jordan para el autovalor ∞, con multiplicidad m_i,

$$N_{m_i} := \begin{bmatrix} 1 & \lambda & 0 & \ldots & 0 \\ 0 & 1 & \lambda & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & \lambda \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \in \mathbb{C}_{m_i \times m_i}.$$

El problema de autovalores generalizado se extiende a lápices matriciales singulares, de la siguiente manera.

Definición A.4. *Sea $A-\lambda B$ un lápiz matricial singular, con $A, B \in \mathbb{C}_{m\times n}, m \geq n$. Se dice que $\vec{v} \in \mathbb{C}^n$ es autovector generalizado por la derecha del lápiz matricial $A - \lambda B$, asociado al autovalor generalizado $\lambda_0 \in \mathbb{C}$, si verifican,*

$$(A - \lambda_0 B)\vec{v} = 0.$$

En este sentido, existe de nuevo una forma canónica.

Teorema A.2 (Forma canónica de Kronecker)**.** *Sea $A-\lambda B$ un lápiz matricial singular, con $A, B \in \mathbb{C}_{m\times n}, m \geq n$. Entonces, existen matrices regulares $P_1 \in \mathbb{C}_{m\times m} y P_2 \in \mathbb{C}_{n\times n}$, tales que el lápiz matricial equivalente $P_1AP_2-\lambda P_1BP_2$ es diagonal por bloques, los cuales son de la siguiente forma: Bloques de Jordan, asociados al autovalor $\lambda^{'}$*

$$J_m(\lambda^{'}) - \lambda Id_m := \begin{bmatrix} \lambda^{'} - \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda_i & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 1 \\ 0 & 0 & \dots & 0 & \lambda^{'} - \lambda \end{bmatrix} \in \mathbb{C}_{m\times m},$$

bloques de Jordan correspondientes al autovalor ∞,

$$N_m := \begin{bmatrix} 1 & \lambda & 0 & \dots & 0 \\ 0 & 1 & \lambda & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & \lambda \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \in \mathbb{C}_{m\times m},$$

bloques singulares por la derecha,

$$L_m := \begin{bmatrix} 1 & \lambda & 0 & \dots & 0 \\ 0 & 1 & \lambda & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & \lambda \end{bmatrix} \in \mathbb{C}_{m\times(m+1)},$$

y bloques singulares por la izquierda,

$$L_m^T := \begin{bmatrix} 1 & 0 & \dots & 0 \\ \lambda & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \dots & \lambda \end{bmatrix} \in \mathbb{C}_{(m+1)\times m}.$$

L_m es singular por la derecha, pues $\forall\lambda$,
$L_m(\lambda^m, -\lambda^{m-1}, \lambda^{m-2}, \dots, \pm 1)^T = 0$, *y de manera análoga para* L_m^T.

Apéndice B

Códigos.

B.1. Capítulo 2.

Un denominador común en los tres siguientes algoritmos es la correcta elección de un ε, ya que en definitiva será el que proporcione la aproximación de M. Esta elección, en el caso con ruido se verá afectada por el método en concreto que estemos empleando, y los valores N, L.

En primer lugar, la elección de ε, en el algoritmo (2.2.1) se verá afectada por la diferencia de los módulos del vector c1, que nos indicará un valor adecuado del mismo.

Código B.1: Método Prony (Algoritmo 2.2.1).

```
function [M,c,lambda] =metodoProny(N,L,eps,f)
    %f vector fila con las evaluaciones.

    %Definimos la matriz H_{2N-L,L+1}:
    H2N=zeros(2*N-L,L+1);
        for i=1:(L+1)
            H2N(:,i)=f(i:(2*N-L+i-1));
        end
    %A partir de esta, construimos H_{2N-L,L}(0).
    H2N0=H2N(1:2*N-L,1:L);

    %Calculamos la solución por mínimos cuadrados del sistema:
    q=lsqminnorm(H2N0,-f(L+1:2*N).');

    %Definimos la matriz compañera y calculamos autovalores
   distintos (ordenados de forma creciente).
    Comp=diag(ones(1,L-1),-1);
```

```
    Comp(:,L)=-q;
    z=unique(eig(Comp.'),'sorted');

    %Matriz de tipo Vandermonde:
    V2N=zeros(2*N,length(z));
    for k=0:(2*N-1)
        V2N(k+1,:)=z.^k;
    end

    %Hallamos coeficientes 1:
    c1=lsqminnorm(V2N,f.');
    [c1,ord1]=sort(c1);
    z=z(ord1); %z_i ordenados tal que c_i<=c_{i+1}.

    %Eliminamos los z_i que no verifiquen la condición.
    i=1;
    while (i<=length(c1) && norm(c1(i))<=eps)
    z(1)=[];
    i=i+1;
    end

    %Aprox. de M.
    M=length(z);

    %Calculamos las aprox. definitivas.
    V2NM=zeros(2*N,M);
    for k=0:(2*N-1)
        V2NM(k+1,:)=z.^k;
    end
    %c=V2NM\(f.');
    c=lsqminnorm(V2NM,f.');

    %Definimos las salidas tal que los lambda vienen dados de
   forma
    %creciente.
    [lambda,ord2]=sort(log(z));
    c=c(ord2);

end
```

En los métodos QR y ESPRIT, el rango numérico de la matriz $H_{2N-L,L+1}$ en (2.25) puede calcularse de diversas maneras, en Matlab la que produce resultados más estables en el caso sin ruido, es utilizar la propia función rank de Matlab. Sin embargo, para el caso con ruido en las mediciones, será necesario calcular un ε adecuado para emplear el método desarrollado en la teoría, que

dependerá de los módulos de diag(R) si se emplea el método QR, o bien de los valores singulares si se emplea el método ESPRIT.

Código B.2: Método Prony QR (Algoritmo 2.2.2).

```
function [M,c,lambda]= metodoPronyQR(N,L,eps,f)
    %f vector fila con las evaluaciones.

    %Calculamos la matriz H_{2N-l,L+1}
    H2N=zeros(2*N-L,L+1);
    for i=1:(L+1)
       H2N(:,i)=f(i:(2*N-L+i-1));
    end

    % Descomposicion QR con pivotaje tal que diagonal de R
   decreciente en modulo.
    [Q,R,P]=qr(H2N);


    %Determinar rango numérico
    % M=0;
    % r=diag(R);
    % while  M<length(r) && norm(r(M+1))>=eps*norm(r(1))
    %    M=M+1;
    % end
     M=rank(H2N);

    %Contsruir T_{M,L}(0) y T_{M,L}(1):
    S2N=R*P.';
    TML0=S2N(1:M,1:L);
    TML1=S2N(1:M,2:L+1);

    %Preacondicionamiento de las T
    % r=diag(R);
    % T0=(inv(diag(r(1:M)))) *TML0;
    % T1=(inv(diag(r(1:M))))*TML1;

    %Construir matriz FM
    FM=pinv(TML0.')*(TML1.');

    %Si se usan las matrices acondicionadas.
    %FM=pinv(T0.')*(T1.');
    %Calculamos los parámetros aproximados
    z=eig(FM);

```

```
    V2NM=zeros(2*N,M);
    for k=0:(2*N-1)
        V2NM(k+1,:)=z.^k;
    end

    c=lsqminnorm(V2NM,f.');

    %Ordeno los lambda de forma creciente, y los c
   respectivamente
    [lambda,ord]=sort(log(z));
    c=c(ord);
end
```

Código B.3: Método ESPRIT (Algoritmo 2.2.3).

```
function [M,c,lambda]=metodoESPRIT(N,L,eps,f)
    %f vector fila con las evaluaciones.

    %Calculamos la matriz H_{2N-l,L+1}
    H2N=zeros(2*N-L,L+1);
        for i=1:(L+1)
            H2N(:,i)=f(i:(2*N-L+i-1));
        end

    %Facroización SVD tal que H2N=U*S*V':
    [U,S,V]=svd(H2N);
    W=V';
    s=diag(S);

    %Hallamos el rango numérico:
    % M=0;
    % while  M<length(s) && s(M+1)>=eps*s(1)
    %     M=M+1;
    % end
    M=rank(H2N);

    %Definimos W_{M,L}(0) y W_{M,L}(1):
    WML0=W(1:M,1:L);
    WML1=W(1:M,2:L+1);

    %Construimos matriz FM y hallamos sus autovalores.
    FM=pinv(WML0.')*(WML1.');
    z=eig(FM);

    %Hallamos los parámetros aproximados:
    V2NM=zeros(2*N,M);
```

```
    for k=0:(2*N-1)
        V2NM(k+1,:)=z.^k;
    end

    c=lsqminnorm(V2NM,f.');


    %Ordeno los lambda de forma creciente, y los c
    respectivamente:
    [lambda,ord]=sort(log(z));
    c=c(ord);

    end
```

Código B.4: Experimento 1.

```
M=6;
c=1:6;
z=[0.9856-0.1628i 0.9856+0.1628i 0.8976-0.4305i 0.8976+0.4305i
    0.8127-0.5690i 0.8127+0.5690i];
eps=10^(-10);

%N,L variando en función de los parámetros a introducir.
N=6;
L=6;

%Evaluaciones conocidas a priori.
k=0:2*N-1;
f=evalflambda(c,log(z),k);

%En el caso de ruidos en las mediciones.
% e=(10^(-4))*linspace(-1,1,2*N);
% f=f+e;

%Obtenemos las aproximaciones(Las aproximaciones de los
    coeficientes c y
% los exponentes lambda vienen dados como vectores columna).
[MaproxP,caproxP,laproxP]=metodoProny(N,L,eps,f);
[MaproxQR,caproxQR,laproxQR]=metodoPronyQR(N,L,eps,f);
[MaproxE,caproxE,laproxE]=metodoESPRIT(N,L,eps,f);

[MaproxP,MaproxQR,MaproxE]

%Calculamos los errores.
[elambdaP,ecP,eevalP]=errores(laproxP,log(z),caproxP,c,N,M);
[elambdaQR,ecQR,eevalQR]=errores(laproxQR,log(z),caproxQR,c,N,M)
```

```
    ;
[elambdaE,ecE,eevalE]=errores(laproxE,log(z),caproxE,c,N,M);

%Mostramos los resultados
[elambdaP ecP eevalP;elambdaQR ecQR eevalQR;elambdaE ecE eevalE]

%Función que calcula las evaluaciones de la función f de
    parámetros c y
%lambda en nodos dados, con c,lambda y nodos vectores fila.
function [f] =evalflambda(c,lambda,nodos)

    f=c*(exp(nodos'.*lambda)).';

end

%Función que reordena las aproximaciones obtenidas para
    emparejarlas con
%los parámetros reales de la función que le corresponde.
function [lreordenado,creordenado]=reordenar(lreal,laprox,caprox
    )

    n=min(length(lreal),length(laprox));
    lreordenado=laprox;
    creordenado=caprox;

    for i=1:n
        [~,j]=min(lreal(i)-lreordenado(i:n),[],ComparisonMethod
    ="abs"); %Seleccionamos la posición correspondiente al valor
    real en posicion i,
        aux=[i:n];
        aux(1)=i+j-1; %Cambiamos las posiciones 1 e i.
        aux(j)=i;
        lreordenado(i:n)=lreordenado(aux); %Cambiamos la
    posición del vector con los lambda, según los indices
    anteriores.
        vaux=creordenado(i);%Hacemos los mismos cambios de
    posiciones en el vector de coeficientes c.
        creordenado(i)=creordenado(i+j-1);
        creordenado(i+j-1)=vaux;
    end
end
```

Código B.5: Experimento 2.

```
M=6;
c=6:-1:1;
```

```
lambda=(1i/1000)*[7 21 200 201 53 1000];
eps=10^(-10);

%N,L variando en función de los parámetros a introducir.
N=10;
L=10;

%Evaluaciones conocidas a priori.
k=0:2*N-1;
f=evalflambda(c,lambda,k);

%En el caso de ruidos en las mediciones.
e=(10^(-4))*linspace(-1,1,2*N);
f=f+e;

%Obtenemos las aproximaciones(Las aproximaciones de los
   coeficientes c y
% los exponentes lambda vienen dados como vectores columna).
[MaproxP,caproxP,laproxP]=metodoProny(N,L,eps,f);
[MaproxQR,caproxQR,laproxQR]=metodoPronyQR(N,L,eps,f);
[MaproxE,caproxE,laproxE]=metodoESPRIT(N,L,eps,f);

%Calculamos los errores.
[elambdaP,ecP,eevalP]=errores(laproxP,lambda,caproxP,c,N,M);
[elambdaQR,ecQR,eevalQR]=errores(laproxQR,lambda,caproxQR,c,N,M)
   ;
[elambdaE,ecE,eevalE]=errores(laproxE,lambda,caproxE,c,N,M);

[MaproxP,MaproxQR,MaproxE]
[elambdaP ecP eevalP;elambdaQR ecQR eevalQR;elambdaE ecE eevalE]
```

Código B.6: Experimento 3.

```
M=6;
c=6:-1:1;
lambda=(1i/1000)*[200 201 202 203 204 205];
eps=10^(-10);

%N,L variando en función de los parámetros a introducir.
N=600;
L=600;

%Evaluaciones conocidas a priori.
k=0:2*N-1;
f=evalflambda(c,lambda,k);

```

```
%Obtenemos las aproximaciones(Las aproximaciones de los
    coeficientes c y
% los exponentes lambda vienen dados como vectores columna).
[MaproxP,caproxP,laproxP]=metodoProny(N,L,eps,f);
[MaproxQR,caproxQR,laproxQR]=metodoPronyQR(N,L,eps,f);
[MaproxE,caproxE,laproxE]=metodoESPRIT(N,L,eps,f);

%Calculamos los errores.
[elambdaP,ecP,eevalP]=errores(laproxP,lambda,caproxP,c,N,M);
[elambdaQR,ecQR,eevalQR]=errores(laproxQR,lambda,caproxQR,c,N,M)
    ;
[elambdaE,ecE,eevalE]=errores(laproxE,lambda,caproxE,c,N,M);
[MaproxP,MaproxQR,MaproxE]
[elambdaP ecP eevalP;elambdaQR ecQR eevalQR;elambdaE ecE eevalE]
```

Código B.7: Algoritmo QR (con M conocido).

```
function [c,lambda]= metodoPronyQRM(M,N,L,f)
    %f vector fila con las evaluaciones.

    %Calculamos la matriz H_{2N-1,L+1}
    H2N=zeros(2*N-L,L+1);
    for i=1:(L+1)
       H2N(:,i)=f(i:(2*N-L+i-1));
    end

    % Descomposicion QR con pivotaje tal que diagonal de R
    decreciente en modulo.
    [Q,R,P]=qr(H2N);


    %Contsruir T_{M,L}(0) y T_{M,L}(1):
    S2N=R*P.';
    TML0=S2N(1:M,1:L);
    TML1=S2N(1:M,2:L+1);

    %Preacondicionamiento de las T
     r=diag(R);
     T0=(inv(diag(r(1:M))))*TML0;
     T1=(inv(diag(r(1:M))))*TML1;

    %Construir matriz FM
    %FM=pinv(TML0.')*(TML1.');

    %Si se usan las matrices acondicionadas.
    FM=pinv(T0.')*(T1.');
```

```
29
30      %Calculamos los parámetros aproximados
31      z=eig(FM);
32
33      V2NM=zeros(2*N,M);
34      for k=0:(2*N-1)
35          V2NM(k+1,:)=z.^k;
36      end
37      c=lsqminnorm(V2NM,f.');
38
39      %Ordeno los lambda de forma creciente, y los c
        respectivamente
40      [lambda,ord]=sort(log(z));
41      c=c(ord);
42  end
```

Código B.8: Experimento 4.

```
1  % Defino la función a aproximar y los parámetros:
2  g=@(x) 1./x;
3  a=1;
4  b=10^6;
5  N=500;
6  L=250;
7  M=20;
8
9  %Valores medidos:
10 h=g(a+((b-a)/(2*N))*[0:2*N-1]);
11
12 %Aplicamos el algoritmo conociendo M:
13 [c,lambda]=metodoPronyQRM(M,N,L,h)
14
15 %Evaluaciones para calcular el error.
16 t=linspace(a,b,10^7);
17 nodos=((2*N)/(b-a))*(t-a);
18 f=evalflambda(c.',lambda.',nodos);
19
20 %Calculamos el error absoluto
21 eabs=abs(g(t)-f);
22
23 %Graficamos error absoluto (escala logarítmica eje y):
24 semilogy(t,eabs);
25 title('Error absoluto $|g(t)-f(t)|$','Interpreter','latex')
26 xlabel(['$t\in$ [' num2str(a) ',' num2str(b) ']'],'Interpreter',
       'latex')
```

Código B.9: Experimento 5.

```
%Definimos la función a aproximar y los parámetros.
%plot(linspace(0,1000,10^7),besselj(0,linspace(0,1000,10^7)))
a=0;
b=1000;
M=20;
N=500;
L=250;


%Valores medidos:
h=besselj(0,a+((b-a)/(2*N))*[0:2*N-1]);

%Aplicamos el algoritmo conociendo M:
[c,lambda]=metodoPronyQRM(M,N,L,h);
figure(1)
plot(real(c),imag(c),"*r");
title('Valores aproximados $c_j$','Interpreter','latex');
xlabel('$Re(c_j)$','Interpreter','latex');
ylabel('$Im(c_j)$','Interpreter','latex');

figure(2)
plot(real(lambda),imag(lambda),"x",Color="#572975",MarkerSize=9)
    ;
title('Valores aproximados $\lambda_j$', 'Interpreter','latex')
xlabel('$Re(\lambda_j)$','Interpreter','latex');
ylabel('$Im(\lambda_j)$','Interpreter','latex');
xlim([-1.2,0.2]);
ylim([-1.2,1.2]);

%Nodos a evaluar la aproximación para calcular el error.
t=linspace(a,b,10^7);
nodos=((2*N)/(b-a))*(t-a);
f=evalflambda(c.',lambda.',nodos);

%Calculamos el error absoluto
eabs=abs(besselj(0,t)-f);

%Graficamos error absoluto (escala logarítmica eje y):
figure(3)
semilogy(t,eabs);
title('Error absoluto $|J_0(t)-f(t)|$','Interpreter','latex')
xlabel(['$t\in$ [' num2str(a) ',' num2str(b) ']'],'Interpreter',
    'latex')
```

B.2. Capítulo 3.

Código B.10: Método ESPRIT para vectores dispersos.

```
function [Maprox,posaprox,c]=metodoESPRITvdis(N,L,eps,y,sigma,wd
    ,D,A)

    %Calculamos la matriz H_{2N-l,L+1}
        H2N=zeros(2*N-L,L+1);
            for i=1:(L+1)
                H2N(:,i)=y(i:(2*N-L+i-1));
            end

    %Facroización SVD tal que H2N=U*S*V':
        [U,S,V]=svd(H2N);
        W=V';
        s=diag(S);

    %Hallamos el rango numérico:
        Maprox=0;
        while  Maprox<length(s) && s(Maprox+1)>=eps*s(1)
            Maprox=Maprox+1;
        end
        %Maprox=rank(H2N);

    %Definimos W_{M,L}(0) y W_{M,L}(1):
        WML0=W(1:Maprox,1:L);
        WML1=W(1:Maprox,2:L+1);

    %Construimos matriz FM y hallamos sus autovalores.
        FM=pinv(WML0.')*(WML1.');
        lambda=eig(FM);
    %Obtenemos las posiciones aproximadas.
        posaprox=zeros(1,Maprox);
            for i=1:length(lambda)
                [~,posaprox(i)]=min(lambda(i)-diag(A),[],
   ComparisonMethod="abs");
            end

    %Hay que tener en cuenta la indexación de Matlab, por lo que
     se resta
    %1.
        posaprox=sort(posaprox)-1; %iniciando en 0


```

```
39      %Obtenemos los coeficientes como solución del sistema
        siguiente, por
40      %mínimos cuadrados:
41          d=wd.^(sigma.*posaprox);
42          F=zeros(2*N,Maprox);
43              for k=0:2*N-1
44                  F(k+1,:)=d.^k;
45              end
46
47
48      %Redondeamos los resultados.
49          c=round(lsqminnorm(F,y.').',7);
50
51      end
```

Código B.11: Experimento 1.

```
1   %Definimos los parámetros.
2       D=1024;
3       N=10;
4       M=9;
5       L=N;
6       eps=0.0005;
7       wd=exp(-2*pi*1i/D);
8
9   %Valores reales (Hay que tener en cuenta que en matlab se indexa
        desde 0,
10  %por lo que las posiciones en Matlab serán +1).
11      pos=[1 5 9 19 42 45 71 115 132];
12      val=[7 5 -7 3 10 5 -5 7 -5];
13
14  %Calculamos un sigma invertible modulo D.
15      if D>1
16          i=2;
17              while gcd(i,D)~=1
18              i=i+1;
19              end
20          sigma=i;
21      else
22          sigma=1;
23      end
24      sigma=11;
25
26  %Matriz asociada al operador.
27      A=diag(wd.^(sigma*[0:D-1]));
28
```

```
%Obtenemos los datos de Fourier del vector.
    y=val*(wd.^(sigma*[0:2*N-1].'*pos)).';

%Si hay error de medición;
    e=linspace(-1,1,2*N);
    y=y+e;

%Se ejecuta el método.
    [Maprox,posaprox,c]=metodoESPRITvdis(N,L,eps,y,sigma,wd,D,A)
    ;
```

Código B.12: Método Prony para funciones Gausianas.

```
function [caprox,alpha] =PronyGaus(M,f,ginv,b,nodos)
    %f vector fila con las evaluaciones.

        f1=ginv(nodos).*f;

    %Definimos la matriz H_{2N-L,L+1}:
        HM=zeros(M,M);
            for i=1:M
                HM(:,i)=f1(i:(M+i-1));
            end

    %Calculamos la solución del sistema:
        q=HM\(-f1(M+1:2*M).');

    %Definimos la matriz compañera y calculamos autovalores
    distintos (ordenados de forma creciente).
        Comp=diag(ones(1,M-1),-1);
        Comp(:,M)=-q;
        z=eig(Comp);

    %Matriz de tipo Vandermonde:
        V=zeros(M);
        for k=0:(M-1)
            V(k+1,:)=z.^k;
        end

        c1=V\(f1(1:M).');

    %Definimos las salidas tal que los lambda vienen dados de
    forma
    %creciente.
        [alpha1,ord]=sort(log(z));
        c1=c1(ord);
```

```
    % Factor corrección no empezar la muestra en x_0=0.
        c1=exp(-nodos(1).*alpha1).*c1;
    %Deshacemos los cambios de variable.
        alpha=alpha1./(2*b);
        caprox=ginv(alpha).*c1;

end
```

Código B.13: Experimento 7.

```
%Definimos los parámetros.
    M=5;
    b=-1i;
    g=@(x) exp(-(b.*(x.^2)));
    ginv=@(x) exp(b.*(x.^2));
    nodos=[-1:8];

%Obtenemos los datos de forma aleatoria uniforme.
    c=(-5+(5-(-5)).*rand(M,1)+(-2+(2-(-2)).*rand(M,1)).*1i).';
    a=(-(pi/2)+((pi/2)-(-(pi/2))).*rand(M,1));

%Experimento 7:
%    c=[-3.5811+0.62296i -0.78239-1.8572i 4.1574+1.3965i
   2.9221+1.7360i 4.5949+0.71494i];
%    a=[0.80971 0.76382 -0.33858 0.48845 -1.0330].';

%Evaluaciones de f conocidad a priori.
    f=c*g(nodos-a.*ones(M,2*M));

%Se ejecuta el método.
    [caprox,alpha]=PronyGaus(M,f,ginv,b,nodos);
    [ealpha,ec,~]=errores(alpha,a.',caprox,c,2*M,M);

%Graficamos los resultados.
x=linspace(0,9,1000);
evalreal=c*g(x-a.*ones(M,1000));
evalaprox=(caprox.')*g(x-alpha.*ones(M,1000));
figure(1)
plot(x,real(evalreal),'r--',x,real(evalaprox),'b*','
   MarkerIndices',1:5:1000)
legend('$Re(f)$','$Re(\tilde{f})$','Interpreter','Latex')
figure(2)
plot(x,imag(evalreal),'r--',x,imag(evalaprox),'b*','
   MarkerIndices',1:5:1000)
legend('$Im(f)$','$Im(\tilde{f})$','Interpreter','Latex')
```

Código B.14: Método Prony para funciones Gausianas moduladas.

```
function [caprox,aaprox,saprox] =PronyGausMod(M,f,ginv,b,nodos)
    %f vector fila con las evaluaciones.

        f1=ginv(nodos).*f;

    %Definimos la matriz H_{2N-L,L+1}:
        HM=zeros(M,M);
            for i=1:M
                HM(:,i)=f1(i:(M+i-1));
            end

    %Calculamos la solución del sistema:
        q=HM\(-f1(M+1:2*M).');

    %Definimos la matriz compañera y calculamos autovalores
   distintos (ordenados de forma creciente).
        Comp=diag(ones(1,M-1),-1);
        Comp(:,M)=-q;
        z=eig(Comp);

    %Matriz de tipo Vandermonde:
        V=zeros(M);
        for k=0:(M-1)
            V(k+1,:)=z.^k;
        end

    %Calculamos los coeficientes aproximados:
        c1=V\(f1(1:M).');

    %Definimos las salidas tal que los lambda vienen dados de
   forma
    %creciente.
        [alpha1,ord]=sort(log(z));
        c1=c1(ord);
        %Factor corrección no empezar en x_0=0.
        c1=exp(-nodos(1).*alpha1).*c1;
        alpha1=2.*alpha1;

    %Deshacemos los cambios de variable.
        saprox=(1/(2*b)).*real(alpha1);
        aaprox=(1/(2*pi)).*imag(alpha1);
        caprox=real(ginv(saprox).*c1);

end
```

Código B.15: Experimento 8.

```
%Definimos los parámetros.
    M=6;
    b=1/2;
    g=@(x) exp(-(b.*(x.^2)));
    ginv=@(x) exp(b.*(x.^2));
    nodos=[0:0.5:5.5];

%Obtenemos los datos de forma aleatoria uniforme.
    c=((-10)+(10-(-10)).*rand(M,1)).';
    a=(rand(M,1)).';
    s=(-5+(5-(-5)).*rand(M,1));

%Calculamos los datos conocidos a priori.
    f=c*(exp((2*pi*1i).*(nodos).*((a.').*ones(M,2*M))).*g(nodos-
    s.*ones(M,2*M)));

%Se ejecuta el método.
    [caprox,aaprox,saprox]=PronyGausMod(M,f,ginv,b,nodos);
    [es,ec,ea]=erroresGausMod(saprox,s.',caprox,c,aaprox,a,M);

%Función cuya función es reordenar los parámetros aproximados,
    en la misma
%posición que los originales (en caso de que sean conocidos),
    para poder
%calcualar de forma correcta los errores.
function [lreordenado,creordenado,sreordenado]=reordenarGausMod(
    lreal,laprox,caprox,saprox)
    n=min(length(lreal),length(laprox));
    lreordenado=laprox;
    creordenado=caprox;
    sreordenado=saprox;

        for i=1:n
            [~,j]=min(lreal(i)-lreordenado(i:n),[],"
    ComparisonMethod","abs");
            aux=[i:n];
            aux(1)=i+j-1;
            aux(j)=i;
            lreordenado(i:n)=lreordenado(aux);
            vaux=creordenado(i);
            creordenado(i)=creordenado(i+j-1);
            creordenado(i+j-1)=vaux;
            saux=sreordenado(i);
            sreordenado(i)=sreordenado(i+j-1);
```

```
40              sreordenado(i+j-1)=saux;
41          end
42  end
43
44  %Fucnión que calcula los errores del método.
45  function [elambda,ec,es]=erroresGausMod(laprox,lreal,caprox,
        creal,saprox,sreal,M)
46      M1=min(M,length(laprox));
47
48      %Calculamos los vectores columna que contienen las
        aproximaciones
49      %ordenadas.
50      [lr,cr,sr]=reordenarGausMod(lreal,laprox,caprox,saprox);
51
52      %Calculamos los diferentes errores
53      elambda=max(abs(lreal(1:M1)-lr(1:M1).'))/max(abs(lreal(1:M1)
        ));
54      ec=max(abs(creal(1:M1)-cr(1:M1).'))/max(abs(creal(1:M1)));
55      es=max(abs(sreal(1:M1)-sr(1:M1).'))/max(abs(sreal(1:M1)));
56
57  end
```