Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA EN TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

ÁREA DE TELEMÁTICA

DESARROLLO DE UNA APLICACIÓN EN MATLAB APP DESIGNER PARA CALCULAR LA TRANSFORMADA INVERSA DE LAPLACE APLICANDO MÉTODOS NUMÉRICOS

D. Martínez Fernández, Ignacio
TUTOR: D. Pérez González, Set

FECHA: NOVIEMBRE 2022

Índice

1.-Memoria 6
1.1.-Objetivos y alcance 6
1.2.-Estudios y análisis previos 7
1.2.1.-Simbología empleada 7
1.2.2.-La transformada de Laplace 8
1.2.3.-Propiedades fundamentales de la transformada de Laplace [15],[21] 8
1.2.3-Transformada inversa de Laplace 13
1.2.4.-Teorema de inversión de Bromwich 13
1.2.5.-Fórmula de inversión de Post-Widder 13
1.2.6.-La transformada rápida de Fourier (FFT) 14
1.2.7.-Métodos numéricos para calcular la transformada inversa de Laplace 15
1.2.8.-Métodos implementados en la aplicación 18
1.3.-Planificación temporal del desarrollo efectuado 23
1.3.1.-Estructura del trabajo 23
1.3.2.-Organización temporal 24
1.4.-Conclusiones 26
1.5.-Bibliografía 27
2.-Presupuesto 29
2.1.-Costes personales 29
2.2.-Costes de material 29
2.3.-Total 29
3.-Documentos Técnicos 31
3.1.-Requisitos del usuario 31
3.2.-Requisitos del software 31
Matlab 31

Requisitos de Matlab 31
Programación por funciones 32
GUIDE 33
App Designer 33
Acceso 34
Entorno de trabajo 34
Elementos en la vista del diseño 36
Elementos de la vista de código 39
3.2.-Diseño de la aplicación 45
3.2.1.-Interfaz de entrada de datos 45
3.2.2.-Interfaz de salida de resultados 47
3.2.3.-Pruebas de la aplicación y funciones Test 49
4.-Manual de usuario y de instalación 59
4.1.-Manual de instalación 59
4.2.-Guía del usuario 60
A.-Método de Crump 60
B.-Método de Levin 61
C.-Método de Stehfest-Gaver 62
D.-Método de Sidi 63

Índice de Figuras

Figura 1.1. Esquema básico de un filtro paso bajo 11

Figura 1.2.- Diagrama de la planificación del TFG 24

Figura 1.3.- Planificación del trabajo en Microsoft Project 25

Figura 1.4.-Diagrama de Gannt según los colores de figura 1.3 25

Figura 3.1.-Entorno de desarrollo de la aplicación con la vista de diseño 35

Figura 3.2.-Entorno de desarrollo de la aplicación con la vista de código 35

Figura 3.3. Vista de código separada por secciones 41

Figura 3.4. Diseño de la interfaz de entrada 47

Figura 3.5. Diseño de la interfaz de salida 48

Figura 3.6. Aproximación de la señal $f_3(t) = te^{-t}$ por el método de Crump 51

Figura 3.7. Aproximación de la señal $f_5(t) = 1 - e^{-t}$ por el método de Crump 52

Figura 3.8. Aproximación de la señal $f_4(t) = t^3e^{-t}$ por el método de Levin 53

Figura 3.9. Aproximación de la señal $f_{11}(t) = \cos 2\pi t$ por el método de Levin 54

Figura 3.10. Error al aproximar $f_3(t)$ con $N = 30$ 55

Figura 3.11. Aproximación de la señal $f_2(t) = e^{-t}$ por el método de Stehfest 56

Figura 3.12. Aproximación de la señal $f_{12}(t) = \frac{1}{2}e^{-t} \sin 2t$ por el método de Sidi 57

Figura 3.13. Aproximación de la señal $f_9(t) = e^{-t} - e^{-2t}$ por el método de Sidi 58

Figura 4.1. Ventana de instalación de la aplicación 59

Índice de Tablas

Tabla 1.1.- Simbología empleada en los estudios previos 7

Tabla 2.1.- Estimación del coste personal por la aplicación 29

Tabla 2.2. Estimación formal que presentar al cliente 30

Tabla 3.1. Elementos de representación de datos 36

Tabla 3.2. Elementos de obtención de datos por inputs 36

Tabla 3.3. Elementos de selección entre estados 37

Tabla 3.4. Elementos de ejecución de acciones 37

Tabla 3.5. Elementos de organización de componentes 37

Tabla 3.6. Elementos del menú de herramientas Designer 38

Tabla 3.7. Elementos del menú de herramientas Editor 40

Tabla 3.8.- Funciones test estudiadas para los métodos 50

1.-Memoria

1.1.-Objetivos y alcance

En las últimas décadas la evolución tecnológica y científica han propiciado cambios en toda la sociedad, haciendo que se hayan implementado una serie de procesos automatizados en prácticamente todos los ámbitos laborales, destacando especialmente la industria automovilística, farmacéutica o química.

La automatización es la aplicación de la tecnología con el fin de realizar una serie de procesos repetitivos de manera indefinida, incrementando la productividad y la calidad del producto de salida. Los procesos que deben realizarse se pueden definir como una serie de modelos dinámicos que varían según el instante temporal, y esto matemáticamente se representa como un conjunto de sistemas de ecuaciones diferenciales respecto del tiempo. Trabajar con estos sistemas tal cual se definen es complejo y, por esta razón, se ha decidido recurrir a la transformada de Laplace, una herramienta que permite transformar un sistema diferencial, como en el caso de los modelos empleados en automatización, en sistemas lineales permitiendo así alcanzar una solución de manera simplificada al mismo problema mediante un cambio en el dominio.

El único inconveniente para trabajar con esta herramienta es la dificultad en la obtención de la transformada inversa de Laplace para poder regresar al resultado del problema en el dominio temporal, donde se había planteado inicialmente.

En algunos casos particulares, este paso puede realizarse con una serie de tablas donde se recogen algunas conversiones entre los diferentes dominios con los que trabaja esta herramienta, pero no todas las señales pueden optar por esta vía para llegar a la solución deseada, recurriendo a la aplicación de unos métodos numéricos con los que se puede obtener una aproximación de la solución del problema.

Todos estos motivos han motivado la programación de una aplicación de Matlab en la que se plantea la implementación de la transformada inversa de Laplace para un gran abanico de señales que el propio usuario introduce como una serie de puntos reales o complejos, con el fin de obtener una solución para un determinado problema. La manera en la que este operador se implementa es mediante la aplicación de 3 métodos numéricos que permiten obtener diferentes aproximaciones de la solución.

La disposición de este trabajo se establece en cuatro capítulos:

El primer capítulo se trata de una memoria en la que primero se presentan un conjunto de estudios previos que dan base a la elección de los métodos numéricos implementados en la aplicación, tras una planificación temporal del desarrollo de la aplicación junto con un conjunto de conclusiones al comparar los resultados reales obtenidos en la práctica por la aplicación frente a los esperados teóricamente para comprobar su eficacia.

El segundo capítulo trata de un presupuesto estimado de cuánto sería el coste de desarrollo efectuado mediante la herramienta Excel, teniendo en cuenta factores diversos que, a falta de datos formales, se ha hecho una estimación de precios.

El tercer capítulo está destinado a los documentos técnicos que especifican los requisitos necesarios para el correcto desempeño de la aplicación, tanto por parte del usuario como a nivel de programa, junto con una comparativa de las alternativas a desarrollar la aplicación que muestra el porque se ha decidido utilizar App Designer como entorno.

Por último, el cuarto capítulo explica cómo utilizar la aplicación, los pasos a seguir y algunas recomendaciones para obtener los mejores resultados posibles.

1.2.-Estudios y análisis previos

1.2.1.-Simbología empleada

Símbolo	*Significado*
$\mathcal{L}$	*Transformada de Laplace*
$\mathcal{L}^{-1}$	*Transformada inversa de Laplace*
$\mathbb{R}$	*Dominio Real*
$\Re$	*Parte real*
$\Im$	*Parte imaginaria*

Tabla 1.1.- Simbología empleada en los estudios previos

1.2.2.-La transformada de Laplace

Se define la Transformada de Laplace para la función f(t) ∈ ℝ para todo t ∈ [0, ∞) de la siguiente manera

$$\mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt \qquad 1.1$$

cuando la integral esté definida en el intervalo especificado. Se denota de manera habitual a la transformada como $\bar{f}(s)$, donde s es el nuevo dominio de trabajo. Para que se pueda aplicar la transformada es necesario que se cumpla la siguiente condición de convergencia

$$\Re s > \gamma, siendo\ \gamma\ una\ constante\ propia\ de\ la\ función \qquad 1.2$$

que sirve para acotar superiormente la función $f(t)$ como

$$|f(t)| = O(e^{\gamma t}) \qquad 1.3$$

Esto significa que la función a transformar es de orden inferior a la función e^{-st} en el caso de valores elevados de t, de tal modo que se dé la convergencia de la función a excepción de los puntos donde aparezcan singularidades.

La definición teórica queda limitada en la aplicación práctica al no poder evaluar la integral en el límite superior, de manera que se establece una cota para poder llevar a cabo la transformada como

$$\mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t) dt = \lim_{\tau \to \infty} \int_0^\tau e^{-st} f(t) dt \qquad 1.4$$

1.2.3.-Propiedades fundamentales de la transformada de Laplace [15],[21]

1.-Linealidad

La linealidad de la transformada de Laplace es una propiedad que se puede demostrar por la propiedad de linealidad de las integrales y la propia definición ya que si

$$f(t) = \alpha g(t) + \beta h(t) \Longrightarrow \mathcal{L}\{f(t)\} = \alpha \bar{g}(s) + \beta \bar{h}(s) \qquad 1.5$$

Esto se prueba con la siguiente demostración

$$\mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)dt = \int_0^\infty e^{-st}[\alpha g(t) + \beta h(t)]dt = \int_0^\infty \alpha e^{-st} g(t) + \beta e^{-st} h(t)dt$$

$$= \alpha \int_0^\infty e^{-st} g(t)dt + \beta \int_0^\infty e^{-st} h(t)dt = \alpha \bar{g}(s) + \beta \bar{h}(s) \qquad 1.6$$

2.-Acotación

La transformada de Laplace devuelve un resultado acotado al cumplir $f(t)$ con las condiciones necesarias de convergencia, probándose que

$$\lim_{s\to\infty} F(s) = 0 \qquad 1.7$$

Esto será útil para algunos métodos donde se necesita evaluar la función en valores que pueden encontrarse por encima del valor máximo que se introduce como dato.

3.-Diferenciabilidad

Dada una función $f(t)$ continua y derivable se desea encontrar la transformada de Laplace de su derivada $f'(t)$. Se asume que la función $f(t)$ cumple la 1.3 a medida que t aumenta su valor y que su derivada es continua para el intervalo [0,T] a excepción de un conjunto finito de punto $t_1, t_2, t_3, \dots, t_n$ todos ellos pertenecientes al intervalo, entonces

$$\mathcal{L}\{f'(t)\} = \int_0^{\mathrm{T}} e^{-st} f'(t)dt = \int_0^{t_1} e^{-st} f'(t)dt + \int_{t_1}^{t_2} e^{-st} f'(t)dt + \cdots + \int_{t_n}^{\mathrm{T}} e^{-st} f'(t)dt \qquad 1.8$$

e integrando esta expresión por partes se deduce que

$$\int_{t_a}^{t_{a+1}} e^{-st} f'(t)dt = e^{-st} f(t)|_{t_a}^{t_{a+1}} + s\int_{t_a}^{t_{a+1}} e^{-st} f(t)dt \qquad 1.9$$

$$= e^{-st_{a+1}} f({t_{a+1}}^-) - e^{-st_a} f({t_a}^+) + s\int_{t_a}^{t_{a+1}} e^{-st} f(t)dt$$

donde

$$f({t_a}^+) = \lim_{\epsilon\to 0^+} f(t_a + \epsilon) \qquad 1.10$$

$$f({t_a}^-) = \lim_{\epsilon\to 0^-} f(t_a - \epsilon) \qquad 1.11$$

y sabiendo que la función es continua en este intervalo salvo en los puntos t_n , se puede decir que

$$f({t_a}^-) = f({t_a}^+) \qquad 1.12$$

haciendo que los términos se anulen entre sí, a excepción del valor inicial 0 y el valor final T

$$\mathcal{L}\{f'(t)\} = e^{-sT}f(T) - f(0) + s\int_0^T e^{-st}f(t)dt \qquad 1.13$$

Y esta expresión es la transformada de Laplace cuando $T \to \infty$ de modo que queda demostrado que

$$\mathcal{L}\{f'(t)\} = s\,\mathcal{L}\{f(t)\} - f(0) \qquad 1.14$$

Se puede demostrar que esta expresión se extiende a las derivadas enésimas, resultando en la expresión

$$\mathcal{L}\{f^n(t)\} = s^n\bar{f}(s) - s^{n-1}\bar{f}(0) - s^{n-2}\bar{f}'(0) - \cdots - \bar{f}^{(n-1)}(0) \qquad 1.15$$

4.-Transformada de la primitiva

Partiendo de la definición se define la transformada de la integral

$$\mathcal{L}\left\{\int_0^t f(u)du\right\} = \int_0^\infty e^{-st}\left(\int_0^t f(u)du\right)dt \qquad 1.16$$

que aplicando integración por partes resulta en

$$\mathcal{L}\left\{\int_0^t f(t)dt\right\} = -\frac{e^{-st}}{s}\left(\int_0^t f(u)du\right)\Big|_0^\infty + \int_0^\infty \frac{e^{-st}}{s}f(t)dt \qquad 1.17$$

Cumpliendo con la condición de convergencia se sabe que el primer termino del sumatorio queda acotado por la función exponencial, resultando en

$$\mathcal{L}\left\{\int_0^t f(t)dt\right\} = \int_0^\infty \frac{e^{-st}}{s}f(t)dt = \bar{f}(s)/s \qquad 1.18$$

5.-Aplicaciones de la transformada de Laplace

En este apartado se muestra un caso en el que la transformada de Laplace es especialmente útil, para resolver problemas con ecuaciones diferenciales o ecuaciones con derivadas parciales.

Ejemplo 1.1. Filtros de señal paso bajo, desarrollado en [26] y [27]

Se sabe que la diferencia de potencial de un capacitor es proporcional a la carga almacenada e inversa a su capacidad.

$$v_c(t) = \frac{q(t)}{C} \tag{1.19}$$

siendo $v_c(t)$ el voltaje en el capacitor, $q(t)$ la carga almacenada, C el valor de la capacitancia. Además, la variación de carga respecto al tiempo es la corriente, de modo que se puede reescribir el comportamiento de la tensión de un elemento capacitivo de la siguiente manera

$$v_c(t) = \frac{1}{C}\int i(t)dt \tag{1.20}$$

donde $i(t)$ es la corriente que circula por un capacitor.

Empleando este elemento y una resistencia conectados en serie se puede emplear un filtro paso-bajo entre los bornes del elemento capacitivo, siendo este uno de los elementos básicos en las comunicaciones.

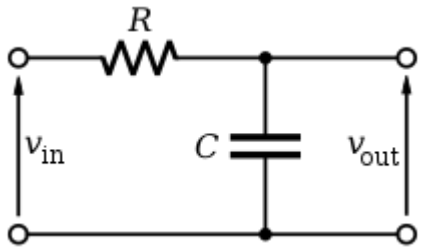


Figura 1.1. Esquema básico de un filtro paso bajo

La respuesta en tensión a la salida de este filtro es la siguiente

$$v_{in}(t) = i(t)R + v_c(t) \tag{1.21}$$

y como trabajar con una integral en el tiempo no es tarea sencilla, se recurre a la transformada de Laplace $\mathcal{L}\{v_{in}(t)\} = v_{in}(s)$ de modo que

$$v_{in}(s) = I(s)R + v_c(s) = I(s)R + \frac{I(s)}{sC} \tag{1.22}$$

$$I(s) = \frac{v_{in}(s)}{R + \frac{1}{sC}} \tag{1.23}$$

Y con la expresión de la corriente se puede calcular el valor de la tensión a la salida

$$v_{out}(s) = \frac{I(s)}{sC} = \frac{\frac{v_{in}(s)}{R + \frac{1}{sC}}}{sC} = \frac{v_{in}(s)}{RCs + 1} \qquad 1.24$$

y con la expresión de salida se puede obtener la respuesta al sistema $H(s)$ como el cociente de tensiones

$$H(s) = \frac{1}{RCs + 1} \qquad 1.25$$

Una razón del uso del filtro paso-bajo se encuentra en el dimensionado de antenas dipolo, cuyo tamaño viene definido por las expresiones [17]

$$\lambda = \frac{C}{F};\ l = \frac{\lambda}{2} \qquad 1.26$$

donde λ es la longitud de onda, C es la velocidad de la luz, F la frecuencia de transmisión y l la longitud de la antena. En caso de querer transmitir una señal $x(t)$ a frecuencias bajas, se puede observar que la dimensión de la antena es inviable en la práctica, de manera que la señal se modula en frecuencia

$$x_{mod}(t) = x(t)\cos(2\pi f_c t) \qquad 1.27$$

donde f_c es la frecuencia de la portadora. Esto permite poder ser transmitida por antenas de tamaño razonable y, es en el proceso de demodulación donde se aplica el filtro paso-bajo, ya que una vez que se baja en frecuencia se encuentra la situación

$$x_{demod}(t) \approx x(t)\cos^2(2\pi f_c t) = \frac{1}{2}x(t) + \frac{1}{2}x(t)\cos(4\pi f_c t) \qquad 1.27$$

Una vez aplicado el filtro paso-bajo puede recuperarse la señal original al eliminar los componentes elevados al doble de la frecuencia. En este problema se han omitido problemas relativos a atenuaciones, ruidos, diferencias entre las fases para poder mostrar de una manera simplificada un campo de aplicación de la transformada de Laplace.

1.2.3-Transformada inversa de Laplace

La transformada inversa de Laplace es la operación que permite pasar del dominio s al dominio temporal, definido matemáticamente como $\mathcal{L}^{-1}\{f(s)\} = f(t)$.Al contrario que la transformada de Laplace, no existe una manera única de realizar esta operación si no que se conocen varios métodos de inversión que se emplean en la práctica, siendo estas tres herramientas matemáticas las que se emplean como base a los métodos implementados en nuestra aplicación: el teorema de inversión de Bromwich, la fórmula de inversión de Post-Widder y la transformada rápida de Fourier (FFT).

1.2.4.-Teorema de inversión de Bromwich

El teorema de inversión de Bromwich dice que para una función f(t) con derivadas continuas y que cumple la condición 1.3 entonces

$$f(t) = \frac{1}{2\pi}\lim_{T\to\infty}\int_{c-iT}^{c+iT} e^{st}\bar{f}(s)ds \qquad 1.28$$

para todo $c > \gamma$

A partir de las demostraciones mostradas por Berzinski [21]-Teorema 2.2 se puede reescribir la expresión anterior (1.28) para el cálculo de la función original como

$$f(t) = \frac{e^{ct}}{\pi}\int_0^{\infty}[\Re\{f(c+i\omega)\cos(t\omega)\} - \Im\{f(c+i\omega)\}sen(t\omega)]d\omega \qquad 1.29$$

Este método sirve para definir las bases de los métodos numéricos de cuadratura para la obtención de la transformada inversa, entre los que se encuentran la mW transformada de Sidi y la transformada P de Levin, ambos incorporados dentro de la aplicación y de las cuales se dedica un subapartado para comentar su base matemática.

1.2.5.-Fórmula de inversión de Post-Widder

Otra opción para calcular la transformada inversa de Laplace es el método propuesto por Post y Widder, en el que todas las señales donde converja la integral siguiente

$$\bar{f}(s) = \int_0^{\infty} e^{-su}f(u)du \qquad 1.30$$

para todo $s > \gamma$, entonces se obtiene que el resultado para $t > 0$ de la transformada inversa de Laplace se puede expresar como

$$f(t) = \lim_{n\to\infty} \frac{(-1)^n}{n!}\left(\frac{n}{t}\right)^{n+1} \bar{f}(n)\left(\frac{n}{t}\right) \qquad 1.31$$

Este resultado se obtiene de la enésima derivada de $\bar{f}(s)$ respecto a s en la expresión 1.30, realizando el cambio de variable $s = n/t$. A partir de esta modificación y aplicando la aproximación de Stirling se da pie al planteamiento mostrado en [21]- Teorema 2.4. bajo el nombre de Teorema de Post-Widder.

Esta expresión permite obtener el resultado deseado a partir de las evaluaciones de la función y sus n derivadas en el dominio real, a diferencia del método de Bromwich en el que se deben hacer evaluaciones en el dominio complejo. Sin embargo, esta opción conlleva el inconveniente de que la convergencia es muy lenta, aumentando el coste computacional necesario para obtener una aproximación suficientemente buena.

1.2.6.-La transformada rápida de Fourier (FFT)

La transformada de Fourier es una operación matemática que permite transformar las señales en el dominio del espacio o el tiempo en un dominio frecuencial, propiedades explicadas al detalle en [21]. La expresión general de esta es

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$$

sin embargo, para su aplicación se utilizará la transformada rápida de Fourier o FFT, la cual obtiene los valores discretos $A(k)$ de la transformada siguiendo la expresión

$$A(k) = \sum_{j=0}^{N-1} X(j)e^{\left(\frac{-2\pi i}{N}jk\right)} \qquad 1.32$$

Estos coeficientes permiten aproximar a la función $X(j)$ como

$$X(j) = \frac{1}{N}\sum_{k=0}^{N-1} A(k)e^{\left(\frac{2\pi i}{N}jk\right)} \qquad 1.33$$

1.2.7.-Métodos numéricos para calcular la transformada inversa de Laplace

1.-Métodos de expansión en serie

El conjunto de métodos de expansión en serie busca aplicar la transformada inversa de Laplace sobre cualquier señal que pueda expresarse como una suma de cocientes. Se basa en la transformada de Laplace de la función t^n

$$\mathcal{L}\{t^n\} = \frac{n!}{s^{n+1}} \quad para\ todo\ n \geq 0 \tag{1.34}$$

Y por definición, la operación inversa se corresponde con

$$\mathcal{L}^{-1}\left\{\frac{1}{s^{n+1}}\right\} = \frac{t^n}{n!} \tag{1.35}$$

El beneficio de este método es la convergencia para cualquier valor de t, sin embargo, no todas las funciones pueden expresarse de esta manera para obtener un resultado deseado y además para valores altos estos métodos suelen requerir un elevado coste operacional. Este tipo de métodos no son finalmente implementados en la aplicación, pero sientan una base para los métodos que sí se han implementado al definir la transformada inversa de varias funciones típicas.

2.-Métodos de cuadratura

Existen varios métodos de cuadratura que devuelven la transformada inversa de Laplace, pero en concreto para la aplicación se han estudiado los métodos de extrapolación frente a los demás por su alta velocidad de convergencia, ya que al no necesitar derivadas de orden n se reduce notablemente el tiempo y los recursos necesarios para obtener el resultado deseado con la suficiente aproximación. El único inconveniente es la necesidad de evaluar la función en un conjunto de números complejos.

Los métodos de extrapolación se basan en el uso de la integral de Bromwich (1.28), fórmula donde se aplica el cambio de variable $s = c + ix$ para expresar la función como

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ixt}\bar{f}(c + ix)ds \tag{1.36}$$

Los métodos de este tipo implementados en la aplicación son la P-transformada de Levin y el de Sidi, los cuales se explican detalladamente en el apartado 1.2.8

3.-Métodos basados en la fórmula Post-Widder

La fórmula de Post-Widder (1.31) se ha comentado como uno de los métodos empleados para obtener la transformada inversa de Laplace en el apartado 1.2.5 y, a pesar de tratarse de una buena aproximación, la convergencia hacia un resultado que se aproxime adecuadamente al resultado teórico se obtiene de una manera muy lenta, obligando a realizar un número muy elevado de operaciones con el gasto de tiempo y recursos que conlleva. Para solventar este problema se decide extrapolar esta fórmula de tal manera que se produzca una aceleración en la convergencia, lo cual permite obtener la transformada inversa a partir de las evaluaciones de la función en el eje real con un número reducido de cálculos.

El único inconveniente de aplicar extrapolación en estos métodos es que no existe un procedimiento que permita recoger el conjunto de todas las secuencias convergentes, de manera que hay algunos métodos que no convergen al aplicar extrapolación, pero sin aplicarla si lo hacen

El único método de esta familia implementado en la aplicación es el método de Stehfest-Gaver, el cual tiene ciertas limitaciones en la práctica que se explican en el apartado de resultados.

4.-Métodos que emplean la Transformada de Fourier

Como última familia de métodos implementados se encuentran los que involucran la transformada rápida de Fourier, desarrollados por Dubner y Abate. El principio de estos métodos consiste en tomar cualquier función $\mathrm{h(t)}$ que cumpla la condición $\mathrm{h(t)} = 0$ para cualquier $\mathrm{t} < 0$, para poder construir una función de periodo 2T de la forma

$$g_n(t) = \begin{cases} h(nT + t), & 0 \le t \le T \\ h(nT - t), & -T \le t \le 0 \end{cases} \qquad 1.37$$

para el caso de $\mathrm{n} = 0,2,4,\ldots$

$$g_n(t) = \begin{cases} h(t), & nT \le t \le (n+1)T \\ h(2nT - t), & (n-1)T \le t \le nT \end{cases} \qquad 1.38$$

para el caso de $\mathrm{n} = 0,2,4,\ldots$, siendo así $g_n(t)$ un conjunto de funciones pares definidas a partir de los valores de $\mathrm{h(t)}$. Al aplicar la transformada de Fourier de las funciones construidas quedan como

$$g_n(t) = \frac{1}{2}A_{n,0} + \sum_{k=1}^{\infty} A_{n,k}\cos\left(\frac{k\pi t}{T}\right) \quad 1.39$$

pudiendo expresar los términos $A_{n,k}$ como

$$A_{n,k} = \frac{2}{T}\int_{nT}^{(n+1)T} h(t)\cos\left(\frac{k\pi t}{T}\right)dt \quad 1.40$$

El sumatorio de todos los g_n permite la aplicación de la transformada inversa de Laplace mediante la transformada de Fourier ya que

$$g_T(t) = \sum_{n=0}^{\infty} g_n(t) = \frac{2}{T}\left[\frac{1}{2}A(\omega_0) + \sum_{n=1}^{\infty} A(\omega_k)\cos\left(\frac{k\pi t}{T}\right)\right] \quad 1.41$$

$$A(\omega_k) = \int_0^{\infty} h(t)\cos\left(\frac{k\pi t}{T}\right)dt \quad 1.42$$

y así se llega a la conclusión de que para $\mathrm{h(t)} = e^{-ct}f(t)$ se observa que adquiere la forma de una transformada de Laplace para $s = c + i(k\pi/T)$, de modo que la expresión 1.41 queda expresada como

$$g_T(t) = e^{ct}\sum_{n=0}^{\infty} g_n(t) = \frac{2e^{ct}}{T}\left[\frac{1}{2}\Re[\bar{f}(c)] + \sum_{n=1}^{\infty}\Re\left[\bar{f}\left(c + i\frac{k\pi}{T}\right)\right]\cos\left(\frac{k\pi t}{T}\right)\right] \quad 1.43$$

Está expresión es prácticamente idéntica a la transformada inversa de Laplace, con la excepción de un error denominado E_1. De las expresiones de $g_n(t)$ se puede reescribir la expresión 1.43 como

$$\sum_{n=0}^{\infty} e^{ct}g_n(t) = \sum_{n=0}^{\infty} e^{ct}h(2nT + t) + \sum_{n=0}^{\infty} e^{ct}h(2nT - t) \quad 1.44$$

Donde el primer término corresponde a $f(t)$ y el segundo término al error E_1

$$g_T(t) = e^{ct}\sum_{n=0}^{\infty} g_n(t) = f(t) + E_1 \quad 1.45$$

este error viene dado como

$$E_1 = \sum_{n=1}^{\infty} e^{-2cTn}[f(2nT + t) + e^{2cT}f(2nT - t)] \quad 1.46$$

De esta expresión Dubner prueba que este error solo disminuye para valores de $t < T/2$, de manera que para valores $\mathrm{t} \in [0, \mathrm{T}/2]$ se considera que $E_1 \approx 0$ y, por tanto

$$f(t) \approx \frac{2e^{ct}}{T}\left[\frac{1}{2}\Re[\bar{f}(c)] + \sum_{n=1}^{\infty}\Re\left[\bar{f}\left(c + i\frac{k\pi}{T}\right)\right]\cos\left(\frac{k\pi t}{T}\right)\right] \qquad 1.47$$

A partir de esta expresión y ciertos razonamientos se obtienen las expresiones de los métodos que involucran la transformada rápida de Fourier, de los cuales se ha decidido seleccionar la transformada de Crump.

1.2.8.-Métodos implementados en la aplicación

Las propiedades que se incluyen a continuación pueden ser consultadas y ampliadas en los apartados [15] y [21]

1.-Crump

El método de Crump es un método englobado en el conjunto que aplica el uso de la transformada de Fourier. Para implementar este método se recurre directamente a la expresión final propuesta por Dubner en las expresiones anteriores, pero implementando funciones de simetría impar

$$k_n(t) = \begin{cases} h(t), & nT \le t \le (n+1)T \\ -h(2nT - t), & (n-1)T \le t \le nT \end{cases} \qquad 1.48$$

y siguiendo la línea propuesta en el apartado de la FFT

$$k_n(t) = \sum_{k=0}^{\infty} B_{n,k}\sin\left(\frac{k\pi t}{T}\right) \qquad 1.49$$

$$B_{n,k} = \int_{nT}^{(n+1)T} e^{-ct} f(t)\sin\left(\frac{k\pi t}{T}\right)dt \qquad 1.50$$

lo que lleva a

$$g_T(t) = e^{ct}\sum_{n=0}^{\infty} k_n(t) = -\frac{2e^{ct}}{T}\left[\Im\left[\bar{f}\left(c + i\frac{k\pi}{T}\right)\right]\sin\left(\frac{k\pi t}{T}\right)\right] \qquad 1.51$$

pero también se tiene que

1.52

$$e^{ct}\sum_{n=0}^{\infty}k_n(t)=f(t)+\sum_{k=1}^{\infty}e^{-2ckT}[f(2kT+t)-e^{2ct}f(2kT-t)]$$

definiendo el error E_2

$$E_2=\sum_{k=1}^{\infty}e^{-2ckT}[f(2kT+t)-e^{2ct}f(2kT-t)] \qquad 1.53$$

De este resultado, prácticamente idéntico al mostrado en el apartado FFT con el hecho de que los signos de los errores son opuestos, de manera que al combinarlos se reduce el error total al representar la aproximación de la señal por Crump como

$$f(t)\approx\frac{2e^{ct}}{T}\left[\frac{1}{2}\Re[\bar{f}(c)]+\sum_{k=1}^{\infty}\Re\left[\bar{f}\left(c+i\frac{k\pi}{T}\right)\right]\cos\left(\frac{k\pi t}{T}\right)+\sum_{k=1}^{\infty}\Im\left[\bar{f}\left(c+i\frac{k\pi}{T}\right)\right]\sin\left(\frac{k\pi t}{T}\right)\right] \qquad 1.54$$

Y el error E_3, siempre es menor que E_1 y E_2

$$E_3=\sum_{k=1}^{\infty}e^{-2ckT}f(2kT+t) \qquad 1.55$$

2.-Levin

Para explicar el razonamiento que se sigue para llegar a la P transformada de Levin se reescribe la integral de Bromwich como

$$f(t)=\frac{e^{ct}}{2\pi}\left[\int_{-\infty}^{0}e^{ixt}\bar{f}(c+ix)dx+\int_{0}^{\infty}e^{ixt}\bar{f}(c+ix)dx\right] \qquad 1.56$$

expresión que lleva a considerar una integral de Fourier de la forma $\int_0^{\infty}g(x)e^{ixt}dx$, entonces si

$$x^{v}g(x)=\sum_{k=0}^{\infty}\frac{\beta_k}{x^k},\qquad v>0 \qquad 1.57$$

Entonces Levin prueba que se puede obtener la expresión

$$\int_{u}^{\infty}g(x)e^{i\omega x}dx\approx b(u)e^{i\omega u}\sum_{k=0}^{\infty}\frac{\gamma_k}{x^k} \qquad 1.58$$

Tomando en cuenta la expresión anterior y llamando $A = \int_0^\infty g(x)e^{ixt}dx$ y $A(u) = \int_0^u g(x)e^{ixt}dx$, la diferencia de estas resulta en

$$A - A(u) \equiv \int_u^\infty g(x)e^{ixt}dx \approx b(u)e^{i\omega u}\sum_{k=0}^{\infty}\frac{\gamma_k}{x^k} \qquad 1.59$$

Levin exigió que la aproximación P obedeciese la relación 1.59 para un total de $k+1$ puntos, lo cual da una ecuación lineal que, abordada con la definición de la transformada en t, se puede expresar la aproximación P_k a la función $f(t)$

$$P_k = \frac{e^{ct}}{\pi}\,\Re\left(\frac{\sum_{j=0}^{k}(-1)^j\binom{k}{j}(j+1)^{k-1}\left[\frac{I_{j+1}(t)e^{-i(j+1)t}}{\bar{f}(c+i(j+1))}\right]}{\sum_{j=0}^{k}(-1)^j\binom{k}{j}(j+1)^{k-1}\left[\frac{e^{-i(j+1)t}}{\bar{f}(c+i(j+1))}\right]}\right) \qquad 1.60$$

donde la expresión

$$I_{j+1}(t) = \int_0^{j+1} e^{ixt}\bar{f}(c+ix)dx \qquad 1.61$$

3.-Transformada de Sidi

Partiendo de la expresión de Bromwich reescrita en el apartado anterior (1.2.4) se puede expresar como

$$f(t) = \frac{e^{ct}}{2\pi}[u_+(t) + u_-(t)] = \frac{e^{ct}}{2\pi}\Re(u(t)) \qquad 1.62$$

donde

$$u_\pm(t) = \int_0^\infty e^{\pm i\omega t}\bar{f}(c+i\omega)d\omega \qquad 1.63$$

integrales aproximadas por la mW transformada. Cuando la transformada de Laplace puede expresarse de la forma

$$\bar{f}(s) = e^{-st_0}g(s) \qquad 1.64$$

donde g(s) sigue una expresión asintótica de la forma

$$g(s)\sim \sum_{i=0}^{\infty} \propto_i s^{\delta-1} \quad para\ s \to \infty \tag{1.65}$$

Sidi toma los siguientes pasos para aproximar la función $f(t)$

Paso 1. Sean

$$\omega_l = \frac{(l+1)\pi}{t-t_0} \qquad l = 0,1,\dots \tag{1.66}$$

Paso 2. Se computan las integrales como

$$\Psi(\omega_l) = \int_{\omega_l}^{\omega_{l+1}} e^{\pm i\omega t}\bar{f}(c+i\omega)d\omega, \qquad l = 0,1,\dots \tag{1.67}$$

$$V(\omega_l) = \int_{0}^{\omega_l} e^{\pm i\omega t}\bar{f}(c+i\omega)d\omega \tag{1.68}$$

Paso 3. Resolver las ecuaciones

$$V(\omega_l) = {W_n}^j + \Psi(\omega_l)\sum_{i=0}^{n-1}\frac{\beta_i}{\omega_l}, \quad l = j, j+1, \dots, j+n \tag{1.69}$$

Para determinar los valores ${W_n}^j$, ya que los valores del sumatorio no son de interés para el método. Para la obtención de los ${W_n}^j$ se deben computar estos cálculos

Paso 1. Se toma

$${M_0}^j = \frac{V(\omega_j)}{\Psi(\omega_j)}, \quad {N_0}^j = \frac{1}{\Psi(\omega_j)} \tag{1.70, 1.71}$$

Paso 2. Para cada j $= 0,1,\dots$ y n $= 0,1,\dots$ se calcula

$${M_n}^j = \frac{{M_{n-1}}^{j+1} - {M_{n-1}}^j}{{\omega_{j+n}}^{-1} - {\omega_j}^{-1}}, \quad {N_n}^j = \frac{{N_{n-1}}^{j+1} - {N_{n-1}}^j}{{\omega_{j+n}}^{-1} - {\omega_j}^{-1}} \tag{1.72, 1.73}$$

Paso 3. Para cada j y n se calcula

$${W_n}^j = \frac{{M_n}^j}{{N_n}^j} \tag{1.74}$$

y estos valores de W convergen rápidamente a $u(t)$ a medida que n tiende a infinito y se fija $j = 0$.

4.-Stehfest-Gaver

El método de Stehfest-Gaver entra dentro de los métodos basados en la fórmula de Post-Widder, ya que a partir de una extrapolación de esta propuesta por Davies y Martin se llega a la expresión

$$I_n(t) = \int_0^\infty \delta_n(t,u) f(u) du \qquad 1.75$$

donde $\delta_n(t,u)$forma una secuencia convergente de deltas, de tal manera que $I_n(t)$ se aproxime a $f(t)$ para valores elevados de n. De este resultado Stehfest propuso el algoritmo

$$f(t) \approx a \sum_{n=1}^{N} K_n \bar{f}(na) \qquad 1.76$$

donde N es un valor par que tiende a infinito, a $= \ln(2)/\mathrm{t}$ y el valor K_n se calcula como

$$K_n = (-1)^{n+N/2} \sum_{k=(n+1)/2}^{\min(n,N/2)} \frac{k^{\frac{N}{2}}(2k)!}{\left(\frac{N}{2}-k\right)!\, k!\, (k-1)!\, (n-k)!\, (2k-n)!} \qquad 1.77$$

Los desarrollos que llevan a la expresión de esta fórmula se encuentran en [25] Este método tiene los beneficios de que los términos K_n son fácilmente calculables, solo necesita de evaluaciones reales de la función $\bar{f}(s)$. El único inconveniente para el uso de este método requiere de una precisión elevada en los valores empleados para su implementación ya que los valores de los coeficientes K_n crecen rápidamente y pequeños errores en los valores aproximados hacen que la función obtenida diverja del resultado deseado.

A pesar de su popularidad como método empleado en diferentes campos, no existen unas condiciones rigurosas que garanticen la convergencia de este método ni la precisión necesaria en caso de que este método converja, a pesar de que se obtienen unos resultados precisos que se aproximan en gran medida a los esperados sin necesidad de un valor de N muy elevado, explicado en el texto [21] Springer, apartado 7.2.

Se ha decidido implementar este método a pesar de que los resultados obtenidos de las pruebas no han sido satisfactorios, ya que la precisión de los datos introducidos necesita ser demasiado alta para valores elevados de N.

1.3.-Planificación temporal del desarrollo efectuado

Una vez analizados los conceptos teóricos sobre los que se ha apoyado el proyecto y una muestra de los resultados obtenidos por la aplicación se va a explicar cuál ha sido la planificación tanto organizativa como temporalmente.

1.3.1.-Estructura del trabajo

El trabajo comienza con la definición del proyecto, marcando su alcance y las herramientas recomendadas para el desarrollo del programa preestablecido.

Una vez se define la idea a realizar, comienza un proceso de investigación y aprendizaje de las herramientas que se han planteado para su desarrollo, siendo estas App Designer y GUIDE, ambas herramientas propias de Matlab.

Tomada la decisión de cuál es la mejor herramienta de las estudiadas, comienza el proceso de diseño, en el que se decide dividir la aplicación en dos ventanas: una para la introducción de los datos y la otra para la salida. Además, en esta etapa se determina de que tipo son todos los elementos incluidos dentro de la aplicación.

Con el diseño finalizado, comienza la etapa de desarrollo del código. La extensión de este ha sido la mayor al tener que asimilar una serie de conceptos matemáticos e implementarlos dentro de Matlab. Este periodo comienza con el desarrollo del código, el cual debe ser depurado y testeado elemento a elemento para garantizar que no exista ningún fallo inesperado.

Tras verificar el correcto funcionamiento del programa, se realizaron los diferentes ensayos con las pruebas test y se valoraron los resultados ofrecidos por los métodos numéricos implementados.

Finalmente se decide comentar el código para poder entender cuáles son los pasos seguidos y el razonamiento empleado y, una vez terminado, se compila la interfaz para poder instalar la aplicación en su versión escritorio.

Durante todo el proceso de elaboración del TFG se han programado semanalmente reuniones con el tutor para dirigir y corregir las dificultades producidas en la elaboración del trabajo. A raíz de estas anotaciones y del proceso general que se ha seguido para elaborar el programa

se ha elaborado un diagrama que muestra de manera muy simplificada la planificación y su implementación.

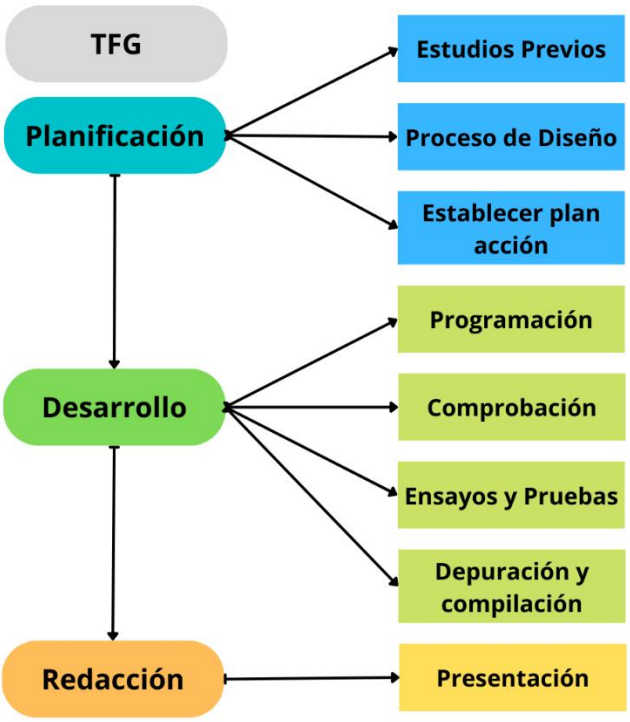


Figura 1.2.- Diagrama de la planificación del TFG

1.3.2.-Organización temporal

Al igual que se ha elaborado un proceso de planificación por tareas, se ha producido una planificación en el tiempo de manera que semana a semana se han establecido unas fechas límites en las que cumplir los objetivos propuestos y elaborar unos nuevos. Esto hace que el trabajo haya avanzado de manera regular, aunque ciertos imprevistos han obligado a alterar puntualmente estos objetivos. La organización temporal comprende desde principios de curso 2022/2023 hasta mediados de noviembre de este, aunque la idea base del proyecto se establece en unas reuniones producidas en mayo del 2022. A pesar de llevar al día el trabajo no ha sido posible completar todas las funcionalidades esperadas al surgir diferentes imprevistos que han impedido progresar lo suficiente para las fechas que nos habíamos propuesto.

Nombre de tarea	Duración	Comienzo	Fin
Trabajo Fin de Grado	**34 días**	**jue 15/09/22**	**mar 01/11/22**
Planificación	**13 días**	**jue 15/09/22**	**lun 03/10/22**
Estudios previos	**10 días**	**jue 15/09/22**	**mié 28/09/22**
GUIDE	5 días	jue 15/09/22	mié 21/09/22
App Designer	5 días	jue 22/09/22	mié 28/09/22
Proceso de diseño	3 días	jue 29/09/22	lun 03/10/22
Establecer plan de acción	1 día	mar 04/10/22	mar 04/10/22
Desarrollo	**20 días**	**mié 05/10/22**	**mar 01/11/22**
Programación	10 días	mié 05/10/22	mar 18/10/22
Comprobación	2 días	mié 19/10/22	jue 20/10/22
Ensayos y Pruebas	4 días	vie 21/10/22	mié 26/10/22
Depuración y compilación	2 días	jue 27/10/22	vie 28/10/22
Redacción del documento	**10 días**	**mié 19/10/22**	**mar 01/11/22**
Redacción y presentación	10 días	mié 19/10/22	mar 01/11/22

Figura 1.3.- Planificación del trabajo en Microsoft Project

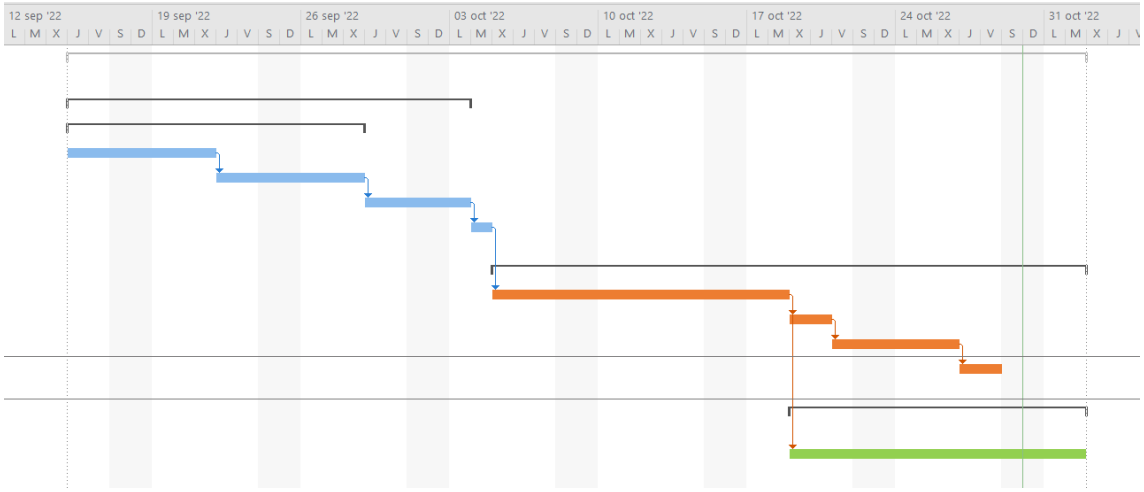


Figura 1.4.-Diagrama de Gannt según los colores de figura 1.3

1.4.-Conclusiones

Una vez se ha terminado el desarrollo del trabajo final de grado paso a exponer mi punto de vista el esfuerzo y el resultado obtenidos, haciendo una comparación objetiva con las metas propuestas en el inicio.

En el momento de comenzar a elaborar este proyecto, no conocía la existencia de App Designer ni muchas de las funcionalidades adicionales proporcionadas por Matlab, como Guide. Esta situación ha sido abordada de manera más sencilla al contar con una base de matemática y del programa durante las primeras etapas del grado, aunque la inexperiencia en este entorno con ciertas peculiaridades ha consumido horas de dedicación.

Además, desconocía completamente los métodos planteados para obtener la transformada inversa de Laplace, fuera de las tablas que se utilizan en algunas asignaturas del grado. Sin embargo, el mayor reto en este trabajo ha sido el estudio, asimilación e implantación de unos métodos numéricos que no forman parte del plan de estudio habitual, obligando a dedicar grandes cantidades de tiempo en entender cuál es la base matemática que hay detrás, que peculiaridades tiene cada método y que funciones admiten, o no, la trasformada inversa mediante un procedimiento en concreto.

Se han fijado unos objetivos en unas fechas aproximadas para poder llevar al día y de manera apropiada el trabajo para que, en caso de ocurrir algún inconveniente en el desarrollo, poder presentar un trabajo digno para las fechas previstas, en el que se vea reflejado el esfuerzo mediante unos resultados aceptables en la mayoría de los casos. Probablemente estos serían mejores y habría más funciones implementadas de haber tenido más tiempo que invertir en el proyecto.

Es por este conjunto de razones que estoy satisfecho con el resultado final ya que me ha permitido mejorar mis competencias adquiridas durante el grado, implementarlas en un resultado tangible, acostumbrarme a un ritmo de trabajo constante y a afrontar la entrega de un proyecto en los plazos solicitados con la mayor calidad posible.

1.5.-Bibliografía

Enlaces Web

[1]- Cursos a su ritmo online - MATLAB & Simulink

[2]- How to Make an Interactive App in MATLAB - YouTube

[3]- MATLAB App Designer - MATLAB & Simulink

[4]- File Exchange - MATLAB Central

[5]- How do I extract data from MATLAB figures? - MATLAB Answers - MATLAB Central

[6]- Tren de pulsos - MATLAB pulstran - MathWorks España

[7]- Enable/disable dropdowns when checkbox checked/unchecked in MATLAB2019A app designer - MATLAB Answers - MATLAB Central

[8]- Aplicaciones reales de la transformada de Laplace

[9]- La automatización industrial en 2020 | BETWEEN Technology

[10]- Transformada inversa de Laplace - Wikipedia, la enciclopedia libre

[11]- Faster numerical inverse Laplace transform in mpmath

[12]- How can I save CSV files directly from App Designer to my hard drive? - MATLAB Answers - MATLAB Central

[13]- Legendre-Gauss Quadrature Weights and Nodes - File Exchange - MATLAB Central

[14]- Transformadas Tipicas Laplace

[15]- Laplace Transform Properties

[16]- Requisitos del sistema para las gráficas - MATLAB & Simulink - MathWorks América Latina

[17]- como calcular el tamaño de una antena y como esta compuesto una antena - YouTube

[18]- Filtro paso bajo - Wikipedia, la enciclopedia libre

[19]- Fundamentos físicos y matemáticos de la modulación de amplitud de una señal — Steemit

[20]- Plantilla de presupuesto para descargar GRATIS - [varios formatos]

Libros y artículos

[21]-NUMERICAL METHODS FOR LAPLACE TRANSFORM INVERSION- Claude Brezinski, Université des Sciences et Technologies de Lille, France. Editorial: Springer.

[22]-Numerical Inversión of the Laplace Transform: a Survey and Comparison of Methods-Brian Davies and Brian Martin, Australian National University of Canberra.

[23]-Numerical Inversion of the Laplace transform by accelerating the convergence of Bromwick's integral- David Levin. Jornal of Computational and Applied Mathematics

[24]-Análisis de circuitos y sistemas utilizando transformada de Laplace- Gabriel Martin Eggly, Bahía Blanca-Argentina

[25]-On the convergence of the Gaver-Stehfest algorithm-A. Kuznestov, York University Toronto.

[26]-Transformada de Laplace en la resolución de circuitos eléctricos- Diego A. Aincionado. Bahía Blanca- Argentina.

[27]-Transformada de Laplace aplicada en filtros analógicos-Emanuel Córdoba, Bahía Blanca-Argentina

2.-Presupuesto

Para elaborar un presupuesto al no tener unos valores concretos de horas, trabajadores ni salarios se ha decidido hacer una estimación suponiendo las horas de proyecto dedicadas por el alumno y por el tutor del trabajo fin de grado, de manera que se ha tomado un salario por hora para ambos en función de su experiencia y asumiendo una serie de costes de dos tipos: costes personales y costes de material.

2.1.-Costes personales

Los costes personales consisten en los salarios estimados que pueden tener los dos tipos de trabajadores dentro del mercado laboral supuestamente, ambos reflejados en la siguiente tabla

Trabajador	Horas trabajadas	Salario por hora (€/hora)	Subtotal (€)
Alumno	360	10	3600
Tutor TFG	30	30	900
		Total (€)	4500

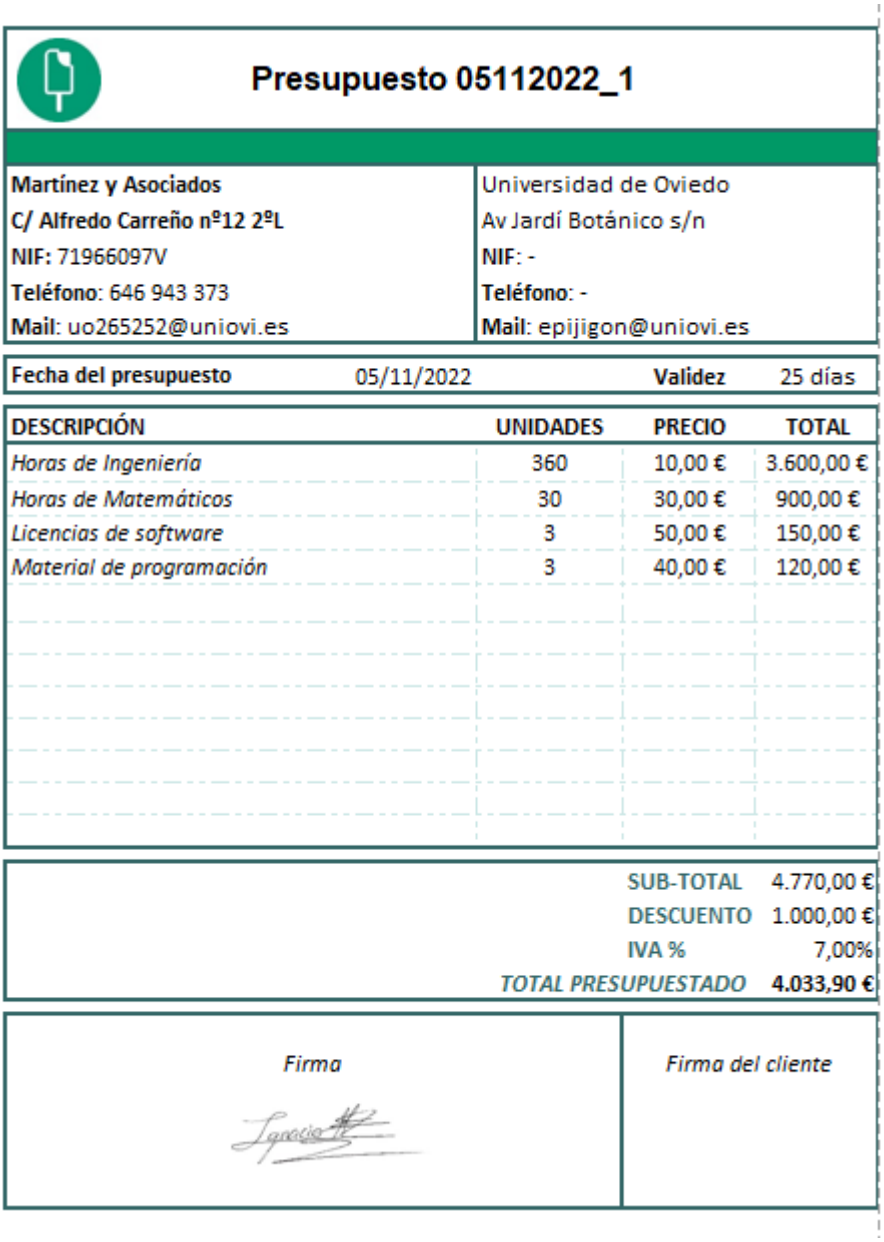
Tabla 2.1.- Estimación del coste personal por la aplicación

2.2.-Costes de material

El coste de material se estima como el conjunto de costes entre software adicional, material necesario para el desarrollo del proyecto, electricidad,… Como se trata de un TFG y la universidad brinda las licencias correspondientes para el programa empleado se ha decidido anular el valor de este coste.

2.3.-Total

Para formalizar el presupuesto, se debe realizar un presupuesto formal dividido en capítulos con sus correspondientes partidas alzadas, pero debido a la limitada cantidad de elementos se ha optado por prescindir de ellos y realizar un presupuesto formal, mostrado en la tabla 2.2, plantilla que se ha sacado de [20]



Presupuesto 05112022_1

Martínez y Asociados	Universidad de Oviedo
C/ Alfredo Carreño nº12 2ºL	Av Jardí Botánico s/n
NIF: 71966097V	**NIF**: -
Teléfono: 646 943 373	**Teléfono**: -
Mail: uo265252@uniovi.es	**Mail**: epijigon@uniovi.es

Fecha del presupuesto	05/11/2022	**Validez**	25 días

DESCRIPCIÓN	UNIDADES	PRECIO	TOTAL
Horas de Ingeniería	360	10,00 €	3.600,00 €
Horas de Matemáticos	30	30,00 €	900,00 €
Licencias de software	3	50,00 €	150,00 €
Material de programación	3	40,00 €	120,00 €

SUB-TOTAL	4.770,00 €
DESCUENTO	1.000,00 €
IVA %	7,00%
TOTAL PRESUPUESTADO	**4.033,90 €**

Firma	*Firma del cliente*

Tabla 2.2. Estimación formal que presentar al cliente

3.-Documentos Técnicos

3.1.-Requisitos del usuario

Al tratarse de una aplicación destinada a la resolución de problemas en el ámbito de la ingeniería, cabe esperar que el usuario final de esta aplicación tenga un perfil con dominio sobre la herramienta de Matlab, entorno en el que se desarrolla la aplicación, junto con capacidad analítica y crítica sobre los resultados reflejados. Estas capacidades son necesarias para poder decidir si los resultados son los esperados y, en caso de no serlo, analizar que parámetros de entrada devuelven una mejor aproximación para poder aplicarlas a resolución de problemas en los que aparece involucrada la transformada de Laplace.

Por otro lado, se recomienda tener conocimientos de Matlab para poder trabajar con el código en caso de querer modificarlo para adecuarse a otros criterios diferentes al que se ha planteado. Un ejemplo podría ser inducir un número indeterminado de iteraciones hasta que el resultado con la precisión decimal deseada o devolver cual es el error cometido frente a la función esperada.

3.2.-Requisitos del software

Matlab

La herramienta Matlab, abreviada de “MATrix LABoratory”, es un software matemático que emplea un lenguaje único y propio llamado M que permite la creación, manipulación y representación de datos o funciones, cálculo de operaciones de elevada complejidad, el uso de matrices, la implementación de algoritmos o, como es el caso de esta aplicación, crear interfaces de usuario. El uso de esta herramienta está ampliamente extendido en las universidades y centros de investigación.

Requisitos de Matlab

De acuerdo con la información proporcionada por la propia organización de Matlab la gran mayoría de sistemas son compatibles con la mayoría de las funcionalidades gráficas de la aplicación, pero existen unos requisitos recomendados que garantizan el éxito en la representación de gráficas y los cálculos típicos en este entorno [16]. Estos son:

- Un mínimo de 1GB de memoria gráfica o GPU para poder liberar lo máximo posible el uso de la memoria RAM del equipo por su mayor lentitud.
- Hardware gráfico compatible con la implementación acelerada de OpenGL®2.1 o posteriores para poder acceder a las funciones más avanzadas de la representación gráfica. En caso de no tenerla implementada solo se puede recurrir a las funciones básicas de la representación gráfica, aunque este no debería ser un factor limitante ya que los equipos lanzados a partir del año 2006 cuentan con una versión superior a OpenGL®2.1. El mejor rendimiento es obtenido en los equipos con OpenGL®4.0 o posterior.
- Las versiones más recientes de los drivers gráficos disponibles del fabricante de cada equipo o el proveedor del hardware gráfico.

El uso principal de Matlab se ve cumplimentado por un conjunto de herramientas: Simulink, App Designer y GUIDE. La primera se trata de una plataforma de simulación mientras que App Designer es un entorno gráfico que se ocupa de la creación y edición de interfaces de usuario para que el backend proporcionado por Matlab resuelva una serie de peticiones que solicita el usuario a partir de una ventana gráfica. Adicionalmente se puede incrementar las funcionalidades mediante el uso de toolboxes.

Estas tres alternativas ofrecen tres metodologías de trabajo diferentes y una ligera variación en las funcionalidades proporcionadas. La elección de un método u otro se basa en los objetivos que el desarrollador desea implementar en su proyecto.

Programación por funciones

La opción más simple para trabajar con Matlab es programar directamente mediante funciones o scripts en lenguaje M, acompañado de las correspondientes representaciones gráficas en archivos .fig, pero presenta el gran inconveniente de configurar uno a uno cada aspecto de la interfaz, como la posición, las propiedades de interactividad, las llamadas o callbacks, … Esta opción supone una mayor cantidad de esfuerzo invertido innecesariamente, ya que tanto en GUIDE como en App Designer los parámetros relativos a la posición, callbacks, fuentes y demás vienen programados por un código generado automáticamente al crear el componente. La única ventaja es la capacidad de organizar el

código libremente e incluir todos los comandos deseados, pero esta ventaja se ve eclipsada por el uso de GUIDE.

GUIDE

GUIDE es un entorno gráfico que se encarga de crear y diseñar interfaces de usuario, de manera que el diseño de las interfaces de usuario y las funciones implementadas se programan de manera separada.

Esto conlleva una serie de beneficios:

- Menor tiempo de programación: el desarrollo de interfaces se simplifica ampliamente frente al método de programación por funciones.
- Simplificación en la construcción de la interfaz gráfica: el hecho de que los componentes pasen a ser arrastrados desde una biblioteca de componentes permite ahorrarse el tener que programar uno a uno las propiedades de cada elemento, lo cual permite un acceso simplificado al desarrollo de aplicaciones. Adicionalmente existe un menú de propiedades donde se puede modificar valores de aspecto, formato y posicionamiento de manera muy intuitiva.
- Generación automática de código: al disponer algún elemento desde la biblioteca de componentes se genera un código de manera automática como los callbacks y esto permite al desarrollador enfocarse principalmente en las funciones que aportan la lógica al programa.

App Designer

App Designer es un entorno gráfico de desarrollo introducido en R2016a que permite el desarrollo de una interfaz gráfica mediante dos vistas estrechamente vinculadas entre sí: una vista del diseño de la aplicación y una vista con el código.

La ventaja de App Designer es que los cambios realizados en cualquiera de las vistas se ven reflejados en la otra, permitiendo tener dos maneras diferentes de modificar la aplicación. Además, este se optimiza frente a GUIDE al facilitar el acceso a las propiedades de los componentes y la compartición de información entre los diferentes elementos de la aplicación.

El único inconveniente de emplear esta interfaz es la disponibilidad de algunas funciones gráficas de Matlab, aunque existe un conjunto más amplio de controles interactivos para realzar el atractivo de la aplicación.

En este trabajo se ha decidido emplear únicamente la funcionalidad de App Designer para la creación de una interfaz de usuario capaz de recibir, procesar y mostrar un amplio abanico de funciones en las que se aplica la operación de la transformada inversa de Laplace definida en 4 casos. La toma de esta decisión se ve apoyada en todas las propiedades que se han enunciado entre las tres opciones que se han planteado para desarrollar este trabajo fin de grado, siendo claramente App Designer la opción más adecuada. Por ello se pasa a describir las principales herramientas y funcionalidades ofrecidas por App Designer.

Acceso

Para acceder a la extensión de App Designer de Matlab existen dos alternativas:

1. Escribir appdesigner en la ventana de comandos dentro de Matlab
2. Desde el menú de inicio, en la barra de opciones seleccionar New→App

Entorno de trabajo

Una vez se ha accedido al entorno de trabajo por cualquiera de las dos alternativas, se accede al entorno de trabajo, en la vista de diseño para ser precisos. Este entorno de trabajo permite realizar modificaciones de la aplicación y tienen las formas mostradas en la figura 3.1 en caso de que la pestaña Design View esté activada o en caso de activar la pestaña Code View, la mostrada en la figura 3.2.

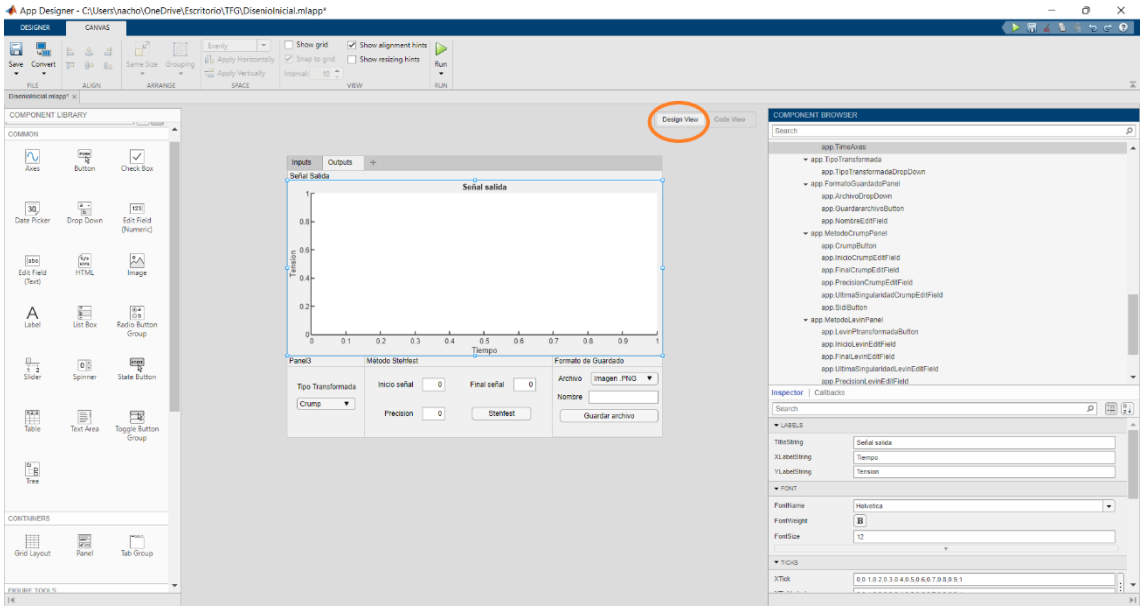


Figura 3.1.-Entorno de desarrollo de la aplicación con la vista de diseño

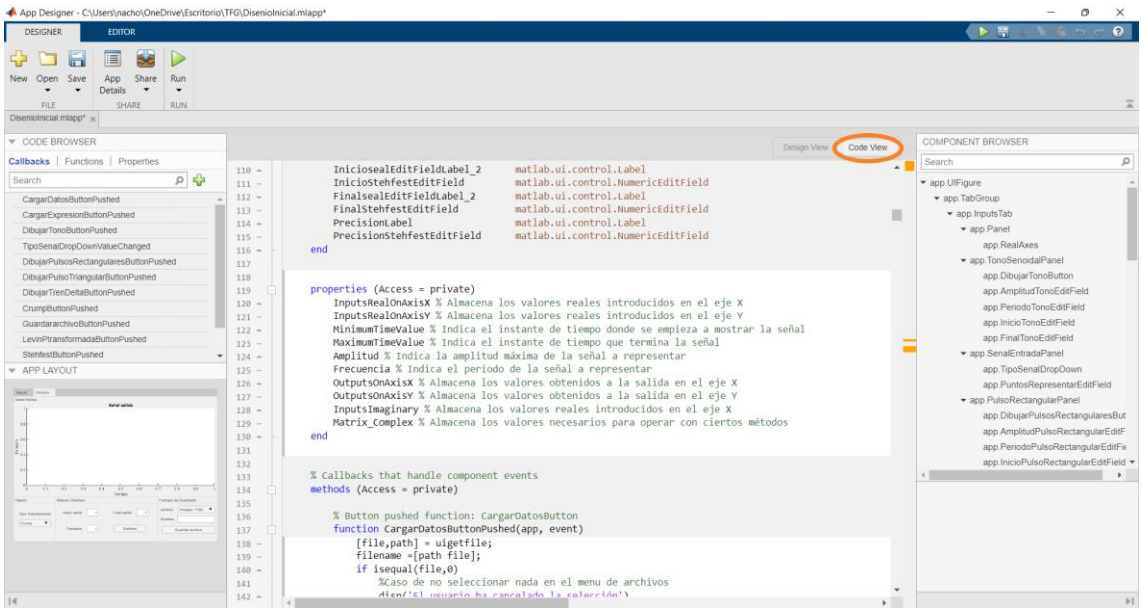


Figura 3.2.-Entorno de desarrollo de la aplicación con la vista de código

A continuación, se pasa a explicar los menús y elementos principales junto con sus funcionalidades:

Elementos en la vista del diseño

Design Editor

Espacio donde se distribuyen los diferentes elementos de control y que compone la representación a mostrar en pantalla al usuario. Se ubica en el espacio central de la pantalla y constituye el núcleo de la vista.

Component Library

Biblioteca de componentes donde se listan los elementos disponibles para emplear en la aplicación. Dentro de este espacio existen subdivisiones en componentes comunes, contenedores o instrumentación. Para poder incorporarlos con seleccionar el elemento y arrastrarlo al espacio Design Editor es suficiente.

En la aplicación desarrollada se han empleado los siguientes elementos mostrados en la tabla con los iconos correspondientes según su clasificación y uso:

Representación de datos

	El objeto Axes se emplea para representar datos en un diagrama de 2D. Este tipo de componente toma los datos del usuario o la aplicación y los representa. Genera un elemento UIAxes.
	El objeto Table se emplea para almacenar datos en una tabla. A pesar de no aparecer visible en la aplicación se emplea para el paso de datos de un archivo CSV a los gráficos y viceversa.

Tabla 3.1. Elementos de representación de datos

Obtención de datos por inputs

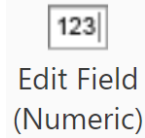
	El componente Edit Field (Numeric) permite especificar un rango de valores, mostrar el valor introducido en un formato adecuado y emplearlo dentro del código.
	El componente Edit Field (Text) permite introducir datos de tipo texto en una línea, permitiendo usar este string resultante dentro del código del programa.

Tabla 3.2. Elementos de obtención de datos por inputs

Selección entre estados

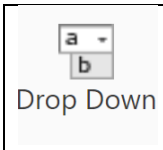
	El componente Drop Down permite la selección de un estado entre varios, mutuamente excluyentes entre sí. Se pueden codificar para que sus callbacks activa una opción u otra.

Tabla 3.3. Elementos de selección entre estados

En estos elementos se pueden programar de tal manera que el string devuelto por el desplegable realiza una llamada o callback que permite seleccionar que paneles se deciden mostrar al usuario.

Ejecución de acciones

	El botón PUSH tienen la funcionalidad de ejecutar los callbacks a petición del usuario una vez se pulsa sobre estos.

Tabla 3.4. Elementos de ejecución de acciones

Organización de componentes

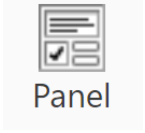
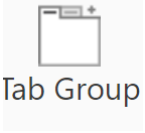
	El objeto panel permite agrupar componentes en una serie de colecciones que siguen un orden jerárquico, de manera que según el estado puedan mostrarse u ocultarse
	El elemento Tab Group permite agrupar componentes de la misma manera que el panel, con la diferencia de que los elementos que el Tab Group suele agrupar son paneles y no otro tipo de elementos, jerarquizando todos los componentes.

Tabla 3.5. Elementos de organización de componentes

El conjunto de estos dos elementos permite distinguir como se organizan los elementos dentro de la aplicación y, habitualmente, esto permite un manejo muy simplificado de que datos y en qué instante intervienen dentro de la aplicación

Button Properties

Se trata de una ventana de propiedades relativas al objeto seleccionado en el Design Editor. Una vez seleccionado el componente se pueden modificar propiedades como el nombre que se le da, su valor, aspectos de formato, posición y referencias a los callback. Se ubica en la esquina inferior derecha y aparece visible al seleccionar un objeto, como aparece en la figura 3.1.

Component Browser

Se trata de un listado donde se muestra, ordenado por niveles, todos los elementos empleados en el programa y el nombre con el que se identifica cada objeto. La estructura general de los nombres es app.NombreObjeto. Existen botones que dan la opción de desplegar este menú y mostrar una lista con los elementos según su organización.

Toolbar Designer

Como en la gran mayoría de programas desarrollados hoy en día, existe una barra de herramientas que permite un acceso rápido a las herramientas más utilizadas dentro de la aplicación, las cuales se explican a continuación acompañadas de sus iconos

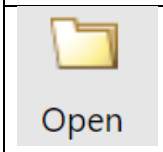
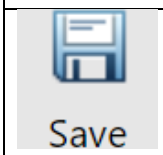
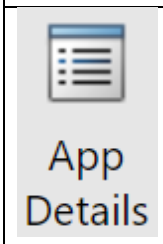
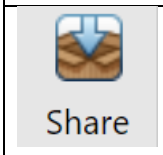
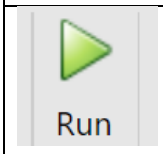
	El icono New permite al usuario crear una aplicación desde cero, siendo la extensión de este tipo de archivo .mlapp
	El icono Open permite al usuario abrir un archivo mlapp ya existente
	El icono Save permite guardar los cambios realizados, ya se trate de una aplicación nueva o se hayan hecho modificaciones en una ya existente
	El icono App Details permite modificar detalles como el autor, una descripción de la aplicación, o indicar de que versión se trata
	La opción share permite al usuario crear una aplicación de escritorio
	La opción run ejecuta la aplicación

Tabla 3.6. Elementos del menú de herramientas Designer

Toolbar Canvas

Adicionalmente en la pestaña de diseño existe otro menú de opciones a parte del original, el cual está destinado a aspectos relacionados con el aspecto de la aplicación, la alineación de los elementos incluidos y otras propiedades. Tanto el menú Canvas como el menú Designer se encuentran en la parte superior de la pantalla.

Vistos los elementos de la vista de diseño con los que se ha trabajado y como se accede a cada uno de ellos, toca presentar la vista de código

Elementos de la vista de código

Code Editor

El editor de código es la parte del programa que contiene tanto el código generado automáticamente al añadir objetos nuevos en la vista de diseño, dando lugar a callbacks, funciones y propiedades implementadas. Como el Design Editor visto anteriormente, el Code Editor ejerce la función de núcleo en esta vista al ser elemento principal sobre el que se construye todo el código y las funcionalidades de la aplicación.

Code Browser

Esta ventana muestra las diferentes callbacks o llamadas que la aplicación solicita al software de Matlab, junto con las funciones empleadas. Además, existe un tipo de variable adicional creadas al inicio del código: las Properties, las cuales son variables donde almacenar valores relevantes dentro de la aplicación que se necesitan entre diferentes secciones de código. Su uso es necesario ya que las callbacks eliminan las variables que genera al terminar de ejecutarse y hay valores que necesitan ser almacenados. Este elemento se encuentra situado en la parte superior izquierda de la vista.

App Layout

Miniatura situada en la parte inferior izquierda para comprobar como es el resultado combinado de los elementos programados.

Component Browser y Button Properties

Al igual que sucedía en la vista de diseño, tanto el buscador de componentes como las propiedades del botón cumplen la misma función en ambas vistas, de manera que la explicación de estas se puede ver en la explicación de la vista anterior.

Toolbar Designer

Igual que en la vista de diseño, existe un menú en formato de barra que permite acceder a las herramientas principales explicadas en la tabla 3.5.

Toolbar Editor

En este menú es donde aparecen las diferencias con la vista del diseño, ya que en esta vista se agregan una serie de herramientas que son destacables

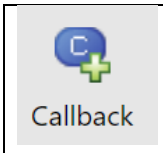
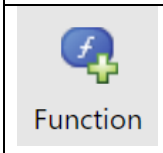
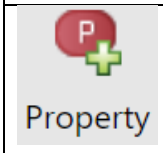
	El icono callback crea una llamada que se activa al pulsar algún elemento
	El icono function permite crear una función de utilidad dentro de la aplicación
	El icono property crea una propiedad que se usa para trabajar con ellas entre diferentes callbacks

Tabla 3.7. Elementos del menú de herramientas Editor

Vista de código

Una vez presentados los elementos principales que integran la aplicación utilizada se pasa a explicar cómo funciona la vista de código ya que el funcionamiento de la vista de diseño consiste básicamente en conocer los posibles elementos a utilizar, su disposición y modificar algunas de sus funciones básicas. Al contrario, la vista de código necesita conocer su estructura, elementos y organización que sigue para poder trabajar con ella.

Como se comentó anteriormente, en el editor de código o Code Editor aparece todo el código que configura la aplicación, tanto el generado de manera automática por el propio programa como el código creado por el programador. Para distinguir el código automático del creado solo hace falta fijarse en el color de fondo del código, ya que el código sobre un fondo grisáceo es código automáticamente generado y no puede editarse directamente, mientras que el código que aparece sobre un fondo blanco sí es editable. Un ejemplo de esto se puede comprobar en la figura 3.2., donde se pueden ver los dos tipos de código.

Independientemente del código que esté programado en la aplicación, la estructura del programa está prefijada y se divide en tres secciones

- La primera sección del código contiene las propiedades de la aplicación, características correspondientes a cada aplicación y a sus componentes.
- La segunda sección incorpora los callbacks y las funciones que utiliza la aplicación.
- La tercera sección sirve como un medio en el que aparecen reflejadas los cambios en las características de los elementos introducidos en la vista de diseño. En resumidas cuentas, es la que crea la aplicación y todos los componentes tal como están.

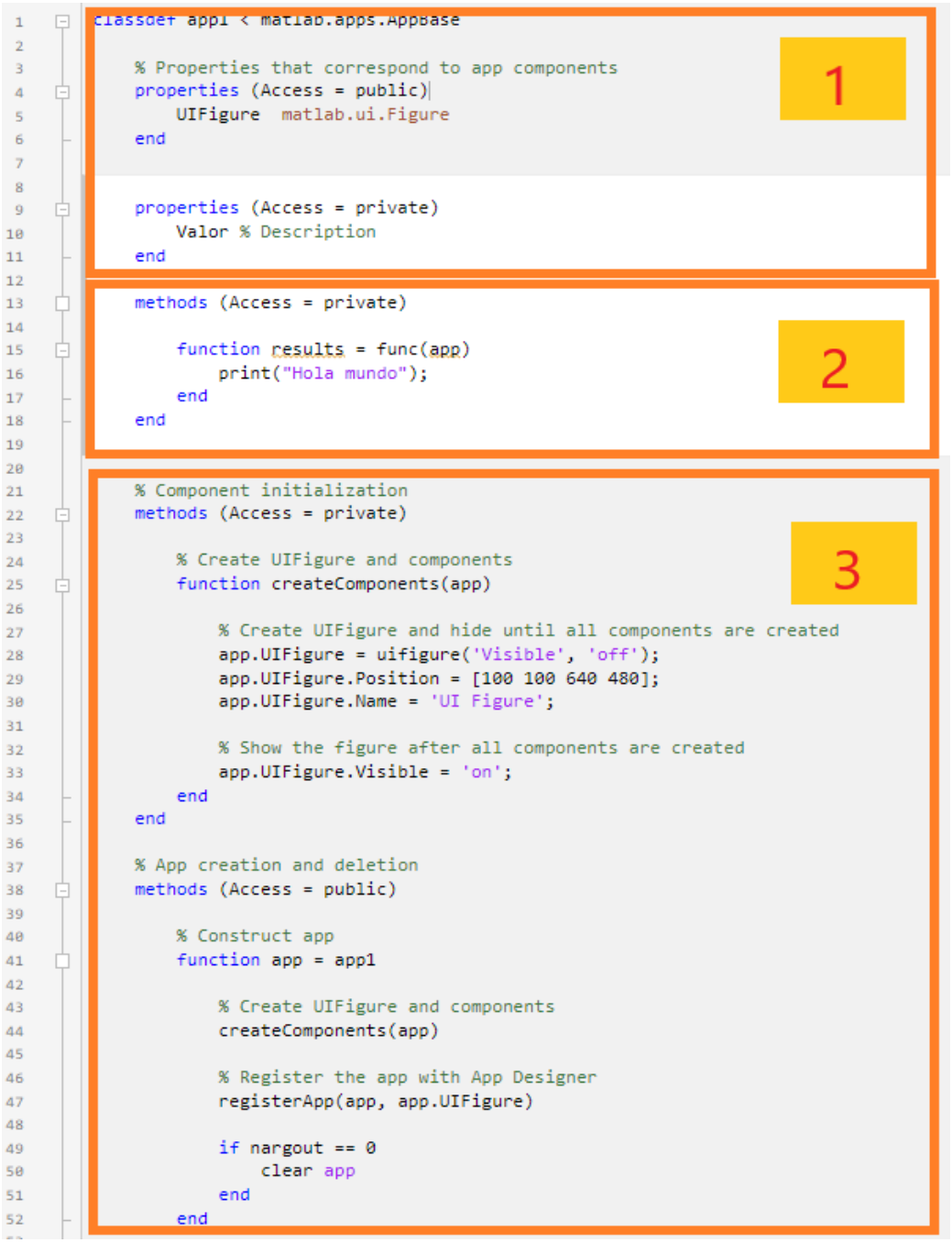


Figura 3.3. Vista de código separada por secciones

Código y llamadas de función

Existen un total de cuatro tipos de funciones en App Designer:

- Funciones que se ejecutan cuando el usuario inicia la aplicación o Start-ups.
- Funciones que se ejecutan cuando el usuario de la aplicación interactúa con un control de la aplicación, estos son los callbacks.
- Funciones que realizan tareas que se pueden reutilizar en varias partes del código, pero no fuera de la aplicación. Estas son conocidas como las funciones de utilidad privada y sirven para múltiples callbacks.
- Funciones que llevan a cabo tareas tanto fuera como dentro de la aplicación. Estas son las funciones de entidad pública.

Para poder crear funciones desde la ventana de código se debe proceder desde la siguiente manera:

1. En la pestaña EDITOR, se debe hacer clic sobre la flecha situada bajo el icono de función y luego seleccionar el tipo de función que se desee: Private Function para crear una función privada y Public Function para crear una función pública.
2. Según se selecciona una opción u otra, aparece en el programa el siguiente fragmento de código genérico (se ha tomado de ejemplo el caso privado)

```
methods (Access = private)

    function results = func(app)

    end
end
```

   La primera vez que se crea una función privada, App Designer crea adicionalmente otro bloque methods que engloba todos los posteriores del mismo tipo.

3. Ahora para poder utilizarla se debe reemplazar la palabra func por un nombre identificativo de la función
4. Por último, añadir los argumentos de entrada como una coma tras la palabra app y el nombre de la variable de entrada. En caso de ser necesario reemplazar la variable de salida results por las salidas que se desean obtener. En caso de no querer devolver un valor se puede eliminar results.

Para poder hacer la llamada a la función privada basta con escribir el identificador de dicha función junto con los argumentos de entrada, mientras que en la pública se necesita que ambos estén en el mismo espacio de trabajo.

Compartir datos

Para compartir datos entre callbacks se utilizan las propiedades. App Designer añade el prefijo app a las propiedades creadas de manera que se pueda acceder a ellas. Hay dos tipos de propiedades: las propiedades para compartir datos entre funciones dentro de la aplicación y las propiedades para compartir dentro y fuera de la aplicación.

Detección de errores

Existen dos maneras de detectar y localizar los errores durante la programación de la aplicación en App Designer, las alertas de codificación y los mensajes del Code Analyzer.

Alerta de codificación

De forma predeterminada, a medida que se va generando el código, App Designer señaliza los errores en la programación mediante alertas, las cuales aparecen marcadas con una señal de precaución en un color anaranjado. Estos errores están relacionados con la llamada errónea a las propiedades.

Alertas del Code Analyzer

Al igual que en el desarrollo de código de Matlab, el Code Analyzer señala los posibles fallos y errores que aparecen en el código a través de mensajes. Para hacerlo visible para el desarrollador las líneas donde se detectan estos avisos se subrayan con una línea zigzagueante naranja y, habitualmente, estos suelen tener consejos recomendados para solucionarlos. Adicionalmente existe un subrayado rojo que indica el fallo de la aplicación tal y como está escrito el código.

Detección de errores durante la aplicación

La particularidad de App Designer es la necesidad de comprobar todas las funcionalidades del código ya que hay errores en la programación que no son detectados por ninguno de los dos métodos comentados anteriormente y solo pueden localizarse una vez se ejecuta el código. Para ello es muy útil el uso de puntos de ruptura para poder avanzar paso a paso desde un determinado punto y comprobar como varían los diferentes elementos y variables para poder comprobar en que punto sucede el problema.

Para borrar este tipo de errores encontrados durante la ejecución se debe corregir el error y volver a ejecutar. Muchos fallos surgen por esta vía.

Guardado de interfaces gráficas

Las interfaces generadas mediante App Designer pueden ser ejecutadas en ordenadores con versiones de MATLAB posteriores a R2016a con ejecutar el directorio de trabajo de la carpeta donde se encuentre el archivo. Sin embargo, si se quiere utilizar la aplicación sin necesidad de instalar una versión de Matlab se debe compilar la aplicación para poder incluirla en la galería de aplicaciones.

El procedimiento para compilar sigue los siguientes pasos:

1. Seleccionar la pestaña DESIGNER y hacer clic en la herramienta Share
2. Una vez realizado el paso anterior, se accede a una ventana donde se deben indicar el nombre del autor, su información de contacto, un resumen y una descripción de la aplicación, junto con una opción que permite incluir un icono para identificar la aplicación.

   Además, deben de añadirse todos los archivos a los que recurre el programa como funciones y otros elementos necesarios para ejecutar el programa.

3. Una vez rellenados todos los campos se pulsa el botón Package y se genera automáticamente el archivo mlappinstall, el cual se puede utilizar para instalar la aplicación directamente.

Si se han seguido estos pasos y adjuntado todos los archivos necesarios se podrá ejecutar la aplicación como una aplicación de escritorio.

Soporte gráfico en App Designer

Los gráficos que App Designer emplea en el programa varían respecto a los empleados habitualmente en Matlab o en GUIDE, de manera que se le dedica un subapartado explicando detalles relevantes de estos. GUIDE emplea las figuras y ejes programadas por uicontrol, las cuales admiten todas las propiedades disponibles mientras que App Designer se basa en figuras (UIFigures) y ejes (UIAxes).

Estos UIFigures y UIAxes comparten cierto parecido a los utilizados normalmente, pero en las aplicaciones que las emplean es necesario especificar en el primer argumento el elemento

donde se quiere realizar la representación, mientras que en Matlab supone que la figura o los ejes actuales son el objeto de destino.

Otro pequeño inconveniente es que las UIFigures no son compatibles con algunas funciones interactivas que, si admiten las figuras tradicionales, como la impresión o la interacción con el ratón al pasar el cursor por los valores representados.

Sin embargo, todos estos inconvenientes entre los ejes no suponen ningún problema para los casos que se tratan en la aplicación, pero es relevante considerar estos factores en caso de trabajar con cualquiera de los dos entornos planteados.

3.2.-Diseño de la aplicación

La aplicación desarrollada presenta un diseño personalizado que busca la entrada de datos y la salida de los resultados en dos ventanas diferentes para que el usuario separe ambos procesos y le sea más fácil distinguir que actividad desea realizar.

Para ello se ha empleado dos elementos Tab Group para establecer la parte de entrada de datos (app.InputsTab) y la salida de los resultados (app.OutputsTab). Ambos presentan una estética similar pero su funcionamiento es completamente opuesto.

3.2.1.-Interfaz de entrada de datos

La interfaz de entrada de datos se ha construido sobre un Tab Group en el que se separan las dos ventanas disponibles: Inputs y Outputs, las cuales son las interfaces de entrada y de salida respectivamente.

Sobre este Tab Group se han dispuesto varios elementos de tipo Panel, de manera que exista una organización y se pueda comprobar cual es la función de cada elemento dentro de la aplicación.

La mitad superior de la interfaz de entrada se encuentra ocupado por un objeto de tipo Axes o ejes, junto con una barra deslizante. El objetivo inicial de estos era representar las señales introducidas dentro del menú o las señales introducidas manualmente por el usuario, pero con la necesidad de evaluar las funciones en el dominio complejo no se vio práctico

representar una malla de datos complejos. En su lugar se restringe la representación solamente a señales reales.

En la parte inferior se encuentras dos paneles visibles para el usuario: Señal de entrada y Tono Senoidal.

El primero se trata de un panel que contiene una barra desplegable o DropDown en el que se debe seleccionar una opción de las ofrecidas: Tono, Pulsos Rectangulares, Pulsos Triangulares, Tren de deltas, Señal real o Datos de archivo.

Originalmente, en las primeras cuatro opciones se plantea representar las señales en función de unos parámetros como amplitudes, periodos y otras variaciones según el caso. Una vez representado se plantea la opción de añadir un elemento interactivo, como puede ser un botón, que al ser presionado realiza la transformada directa de Laplace. Estos datos se utilizan en la interfaz de salida como los datos de entrada para los diferentes métodos, de manera que ya se conozca la señal de origen, permitiendo comprobar la efectividad de estos métodos. Sin embargo, no se ha podido terminar de implementar por las dificultades e inconvenientes que han aparecido en la implementación de los métodos.

La opción “Señal real” permite introducir una señal ya construida que sirva como datos de entrada para el método de Stehfest-Gaver, que es el único que acepta señales reales. Los formatos de entrada de señal disponibles son dos: un archivo .fig, como puede ser uno construido mediante un script de Matlab, o un archivo .csv, el cual presenta en dos columnas los datos del eje X en la primera columna y los datos del eje Y en la segunda.

La opción “Datos de archivo” elabora una malla de datos en el campo complejo delimitada por unos valores reales de inicio y fin y otros valores imaginarios de inicio y fin. Para esta malla de puntos se indica una función que se evalúa en todos estos puntos y estos datos son los utilizados como referencia para aplicar los métodos de Crump, Sidi y Levin.

El resultado final de la interfaz de entrada se muestra en la figura 3.4.

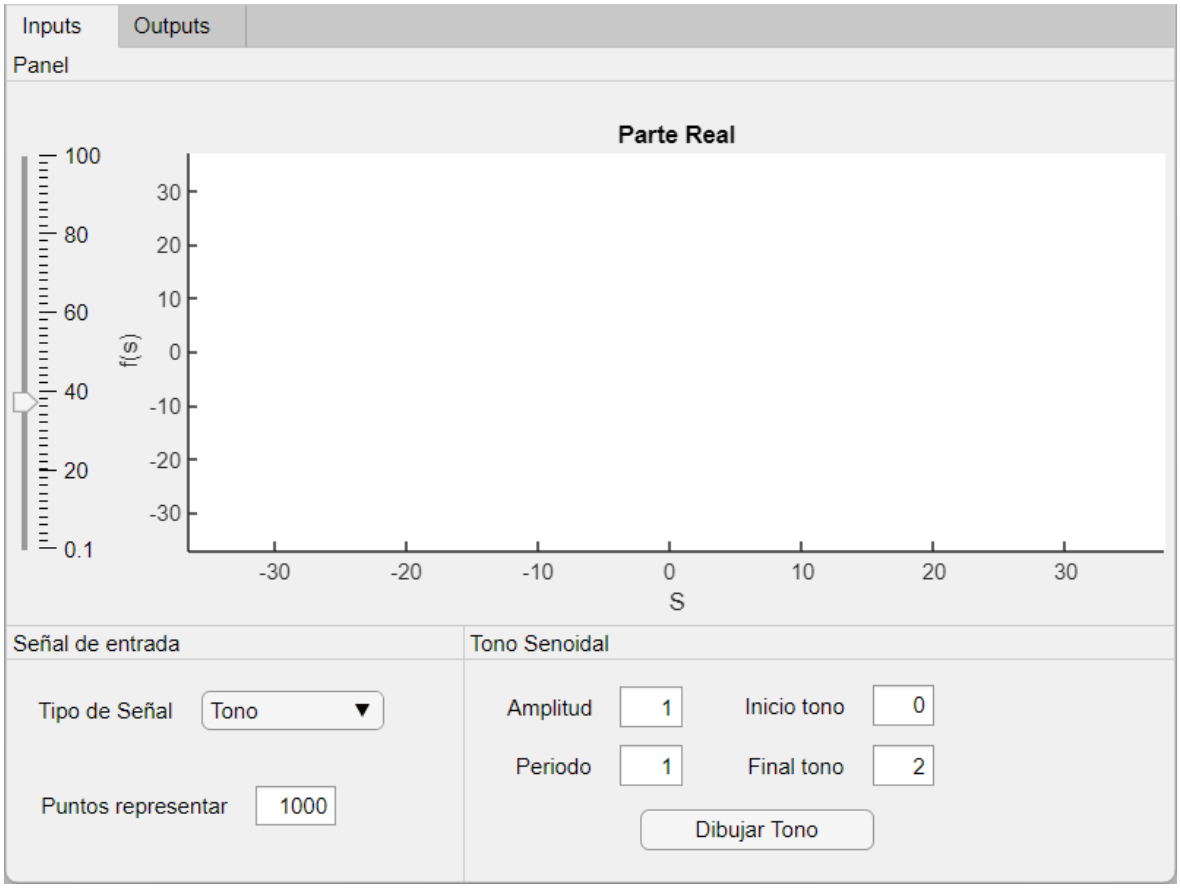


Figura 3.4. Diseño de la interfaz de entrada

3.2.2.-Interfaz de salida de resultados

La interfaz de salida sigue una estructura muy similar a la interfaz de entrada, que consta de unos ejes en los que se representa la señal obtenida de aplicar uno de los métodos implementados y una serie de menús y desplegables que permiten seleccionar el método a aplicar y el formato de salida de los datos. Una captura de la interfaz de salida se muestra en la Figura 3.5

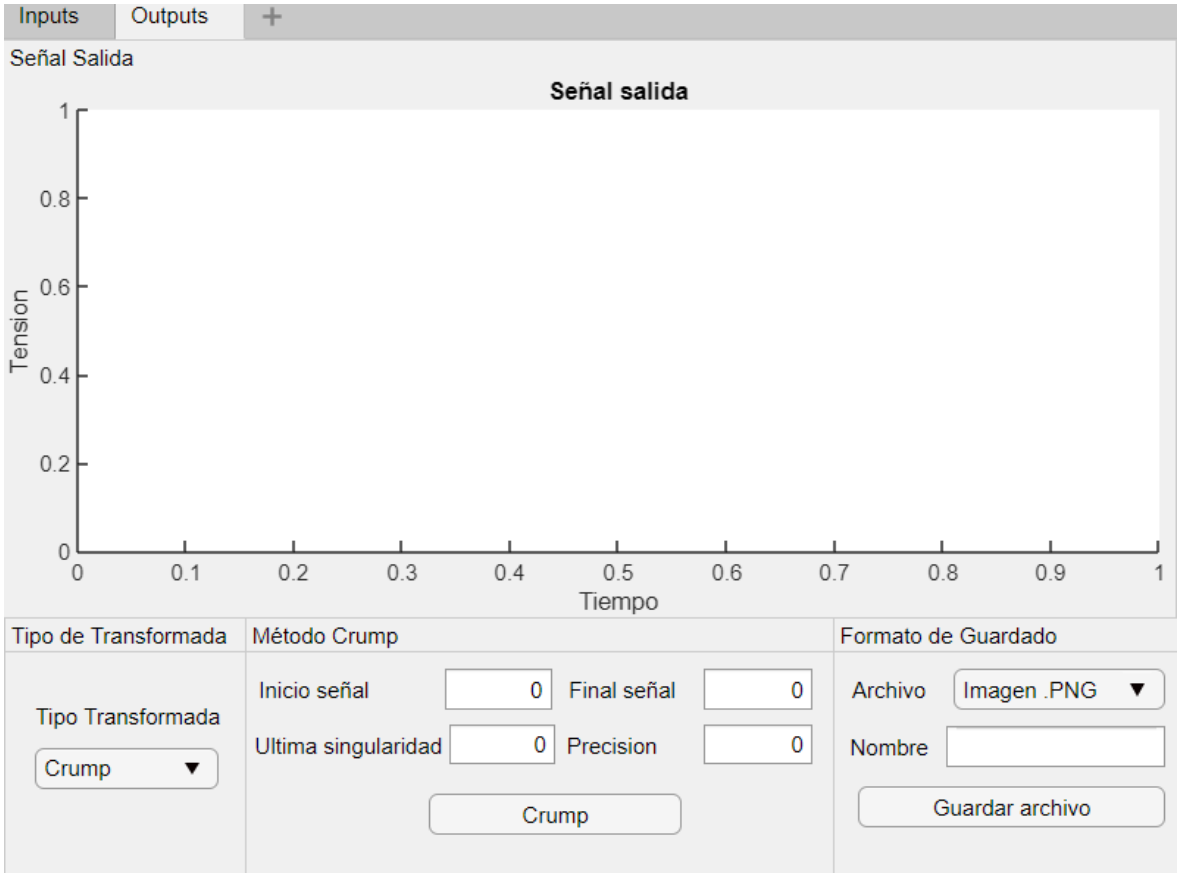


Figura 3.5. Diseño de la interfaz de salida

A continuación, se explica cómo están sus elementos distribuidos y cual es la funcionalidad de cada uno de ellos:

- Señal de salida: se trata de un panel que contiene un objeto de tipo Axes en que se representa la señal en el dominio temporal una vez se ha aplicado la transformada inversa de Laplace.
- Tipo de transformada: panel con un único elemento: un menú desplegable o DropDown en el que se debe seleccionar el método por el que se desea obtener un resultado. Entre sus opciones se encuentran los siguientes métodos: Crump, Levin, Stehfest y Sidi. Una vez seleccionado un método, se mostrará el panel homónimo para indicar los parámetros que indican como se realiza la transformada.
- Método Crump/Levin/Sidi: estos paneles son muy similares entre sí ya que los tres parten de una señal evaluada en el dominio complejo y emplean parámetros similares. Estos parámetros indican factores que delimitan la salida, los cuales son

  Inicio/Final señal: ambos parámetros representan los instantes temporales entre los que se quiere obtener un resultado, El primer punto mostrado en los

ejes “Señal de salida” es igual al valor de “Inicio señal” y el último es igual al valor de “Final señal”.

Última singularidad: este parámetro indica un valor real representado en los datos de entrada y que, en caso de existir una singularidad, este punto sea superior a la singularidad.

Precisión: parámetro que indica el número de iteraciones a realizar hasta considerar satisfactorio el resultado. En cada caso cada uno opera de manera diferente, pero a mayor valor de este parámetro mejor suele ser la aproximación.

- Método Stehfest: este panel se comenta en una viñeta diferente al obtener el resultado de la transformada inversa por medio de evaluaciones reales de la señal en lugar de evaluaciones complejas, lo cual puede facilitar la obtención del resultado al recurrir únicamente a la parte real de la señal. Además, en este método no hace falta considerar los valores de la singularidad del mismo modo que en los métodos previos, si no que se realiza un muestreo de la señal $\bar{f}(s)$ en el intervalo de puntos na, de modo que con los instantes temporales se obtienen los valores de $a = \ln 2 \,/t$ y los valores de n como el total de iteraciones a realizar. La necesidad de especificar estos parámetros lleva al uso de estos campos

  Inicio/Final señal: estos campos cumplen la misma función que en el resto de los métodos, explicado en la viñeta anterior. Delimitan el inicio y final del resultado.

  Precisión: este parámetro indica el número de iteraciones que se deben realizar para obtener el resultado, sin embargo, este valor necesita ser un valor par, ya que los números impares de iteraciones no ha sido planteado en la elaboración del método por el propio Stehfest.

3.2.3.-Pruebas de la aplicación y funciones Test

Para comprobar la validez de estos métodos se han aplicado sobre un conjunto de funciones test, las cuales recogen un gran abanico de los casos que se pueden encontrar en la práctica y comprobar así en que casos se recomienda algún método sobre otro. Las funciones test se muestran en la tabla 3.8 [22].

n	$f_n(t)$	$\overline{f_n}(s)$
1	t	$\frac{1}{s}$
2	e^{-at}	$\frac{1}{s+a}$
3	te^{-at}	$\frac{1}{(s+a)^2}$
4	$t^{k-1}e^{-at}$	$\frac{(k-1)!}{(s+a)^k}$
5	$1-e^{-at}$	$\frac{a}{s(s+a)}$
6	$t-\frac{1-e^{-at}}{a}$	$\frac{a}{s^2(s+a)}$
7	$1-(1+at)e^{-at}$	$\frac{a^2}{s(s+a)^2}$
8	t^{k-1}	$\frac{(k-1)!}{s^k}$
9	$e^{-at}-e^{-bt}$	$\frac{(b-a)}{(s+b)(s+a)}$
10	$sen(at)$	$\frac{a}{s^2+a^2}$
11	$cos(at)$	$\frac{s}{s^2+a^2}$
12	$\frac{1}{b}e^{-at}sen(bt)$	$\frac{1}{(s+a)^2+b^2}$
13	$e^{-at}sen(bt)$	$\frac{s+a}{(s+a)^2+b^2}$

Tabla 3.8.- Funciones test estudiadas para los métodos

A continuación, se comprueba para cada método que funciones se aproximan adecuadamente a las transformadas inversas teóricas correspondientes indicando con que precisión lo hacen o, en caso contrario, tratar de averiguar qué familia de funciones no es recomendada

Crump

El método de Crump ha resultado, junto con la transformada de Sidi, ser uno de los métodos que mejor aproxima las funciones Test mostradas en la tabla, donde sus resultados para un numero de iteraciones comprendido entre 70 y 90 se pueden llegar a obtener precisión de 3 decimales correctos en la aproximación.

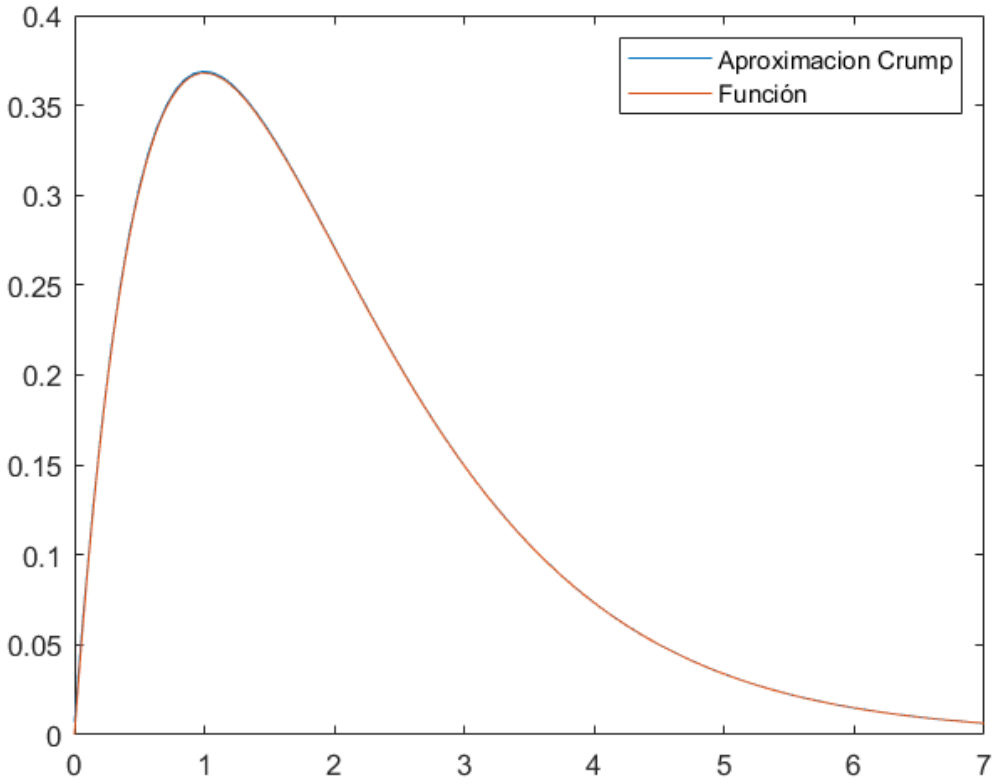


Figura 3.6. Aproximación de la señal $f_3(t) = te^{-t}$ por el método de Crump

A pesar de obtener aproximaciones bastante buenas para un abanico de funciones, debido principalmente a errores de aproximación entre los valores discretos empleados como datos de la función, al aplicar existen un conjunto de funciones donde la aproximación se restringe a un intervalo de puntos intermedios al obtener fuera de ellos un comportamiento oscilante que hace divergir la función aproximada, como es el caso de $f_5(t)$ o $f_{13}(t)$.

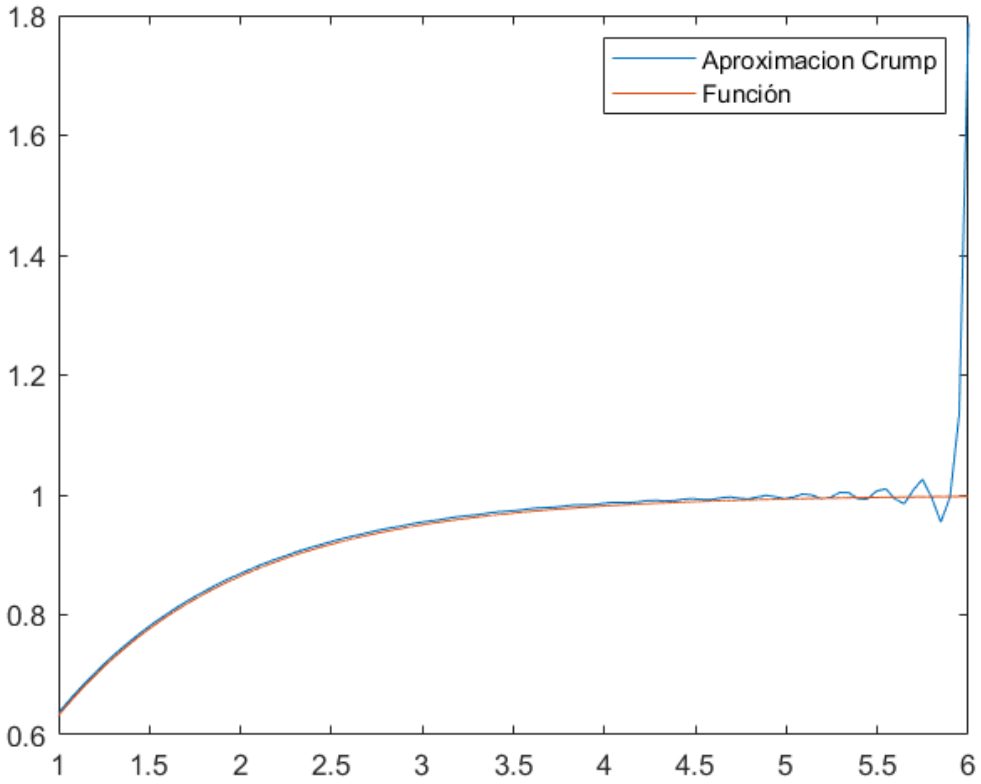


Figura 3.7. Aproximación de la señal $f_5(t) = 1 - e^{-t}$ por el método de Crump

Este comportamiento oscilante suele hacerse más visible a medida que los valores de t aumentan.

Finalmente se ha determinado que la función $f_1(t)$ no se puede aproximar por este método debido a que la función no presenta una convergencia uniforme, de manera que la aproximación no presenta ningún intervalo de puntos donde sí se pueda considerar apta la aproximación.

Levin

La transformada de Levin ha sido la segunda en evaluar mediante las diferentes funciones test, mostrando unas aproximaciones razonables en la mayoría de los casos en los que converge, asumiendo un error en el segundo digito en aquellas funciones donde la aproximación es relativamente buena.

Sin embargo, en las funciones $f_1(t)$, $f_7(t)$ y $f_{13}(t)$ las aproximaciones no han sido lo suficientemente buenas para considerar la convergencia por este método.

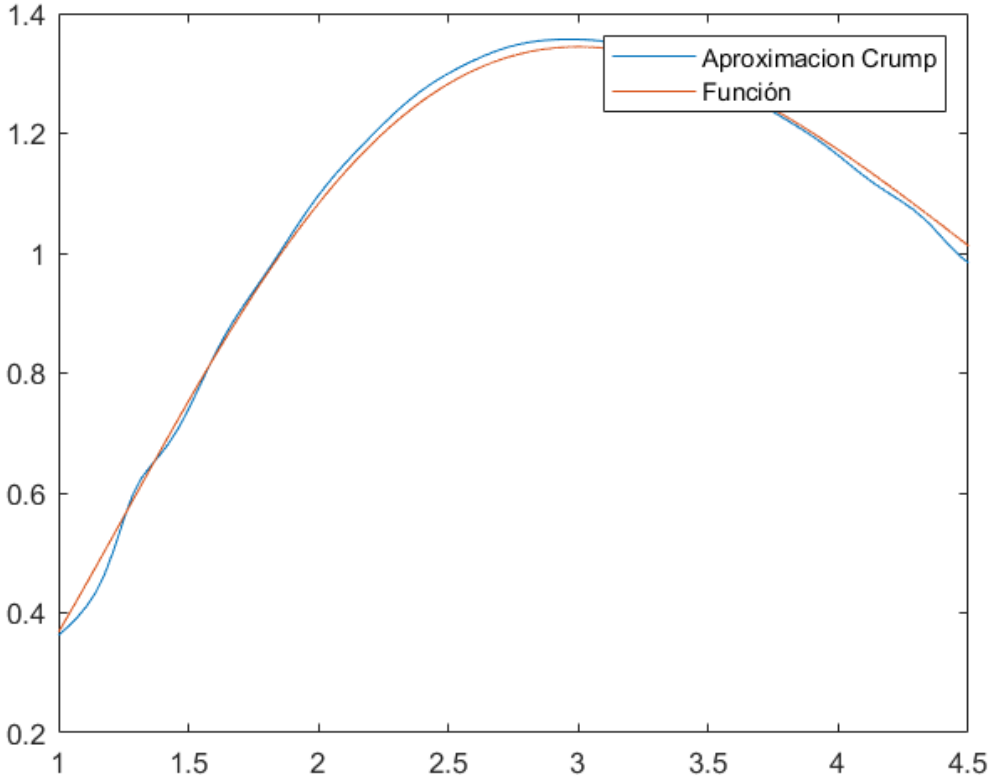


Figura 3.8. Aproximación de la señal $f_4(t) = t^3 e^{-t}$ por el método de Levin

Sin embargo, no todas las funciones aproximan del mismo modo ya que las funciones seno y coseno no son aproximadas con demasiada precisión, al llegar a presentar diferencias de 3×10^{-1} para amplitudes unitarias, lo cual supone un error nada despreciable para tener en cuenta. Se muestra de manera gráfica en la figura 3.9.

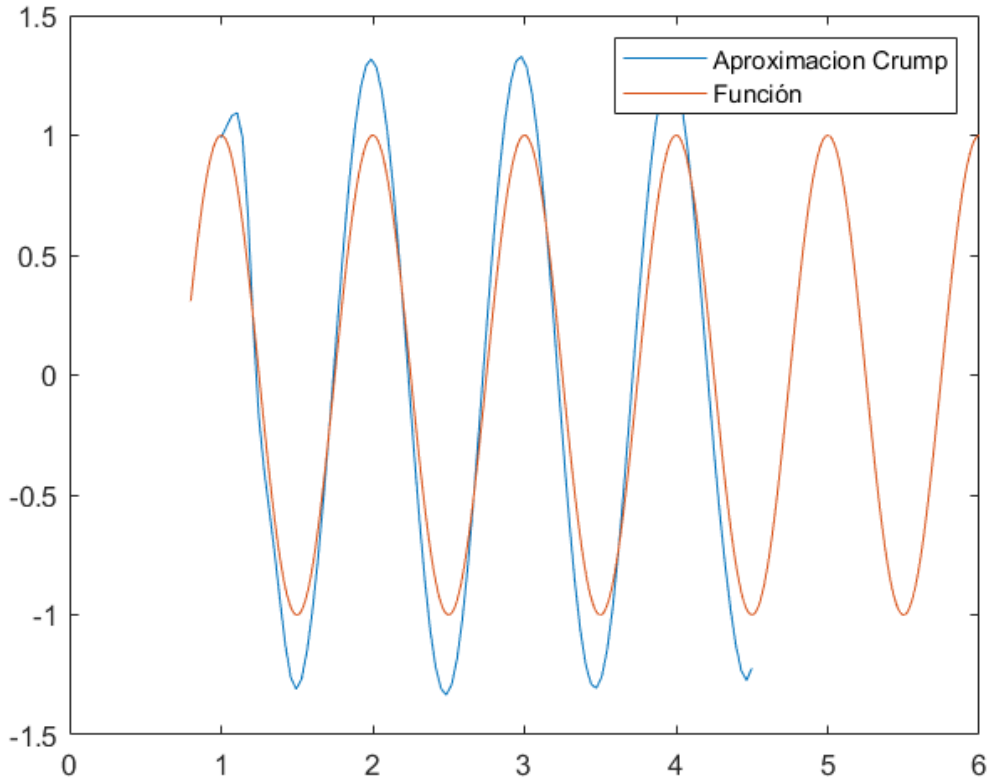


Figura 3.9. Aproximación de la señal $f_{11}(t) = \cos 2\pi t$ por el método de Levin

Stehfest-Gaver

El método propuesto por Stehfest es uno de los métodos que plantea una de las mejores aproximaciones a los resultados dentro de los métodos basados en la fórmula de Post-Widder, sin embargo, han surgido ciertos inconvenientes al implementar este método que impiden obtener todas las aproximaciones que cabe esperar obtener.

El principal problema de este método surge al intentar elevar el nivel de iteraciones a realizar por el programa, ya que la falta de precisión en los valores interpolados que se emplean junto con la falta de capacidad de computación del equipo para devolver todos los valores en los que se debe evaluar la función de la señal $\bar{f}(s)$ dan lugar a unos errores que, al multiplicar por los altos valores $K(n)$ produce un error excesivamente elevado que impide aproximar un gran número de las funciones test consideradas. Un ejemplo de este fenómeno se puede observar en la figura 3.10 donde se ha tomado una precisión $N = 30$ para aproximar $f_3(t)$, la cual muestra unos resultados aceptables para valores $N = 6$.

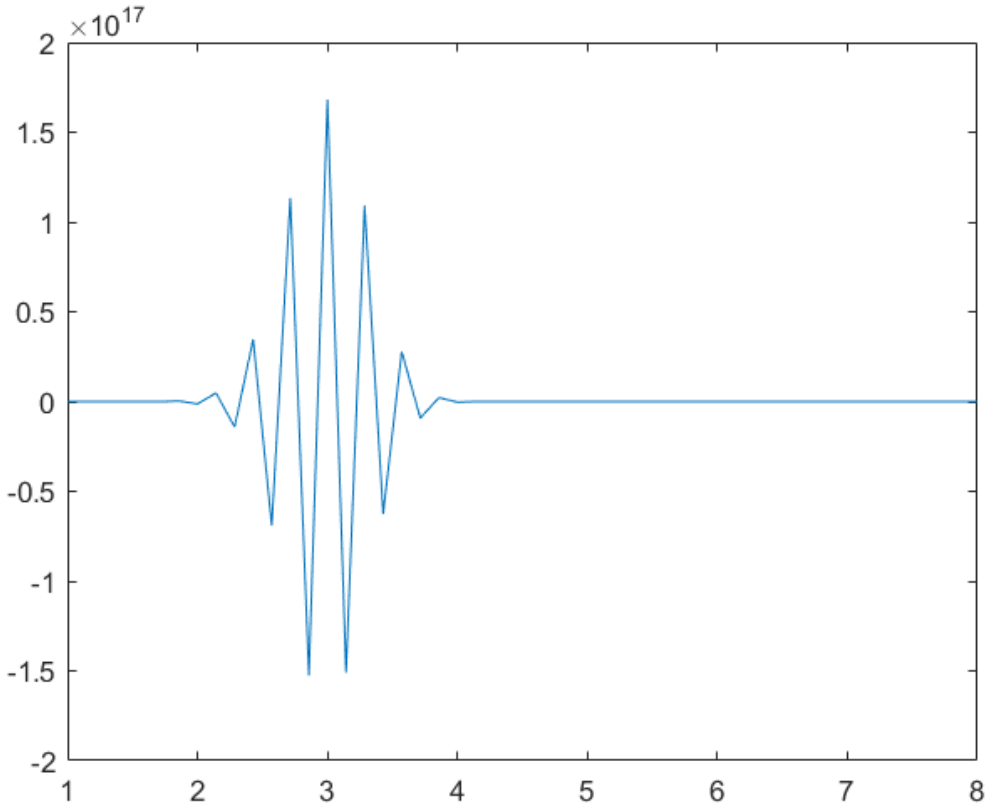


Figura 3.10. Error al aproximar $f_3(t)$ con $N = 30$

Estas limitaciones han permitido aproximar con resultados satisfactorios las funciones $f_2(t)$, $f_3(t)$, $f_4(t)$ y $f_5(t)$, todas ellas con niveles de precisión bajos no superiores a 8 iteraciones. Las otras funciones podrían ser aproximadas como se muestra en un articulo publicado por Davies y Martin [25] en el que se aproximan funciones de la forma de $f_1(t)$, $f_8(t)$ o $f_{12}(t)$, todas ellas con resultados satisfactorios.

Finalmente, en los casos donde sí se ha podido aproximar por este método, se alcanza una precisión de dos dígitos como mínimo en los intervalos adecuados, como se puede comparar en la figura 3.11.

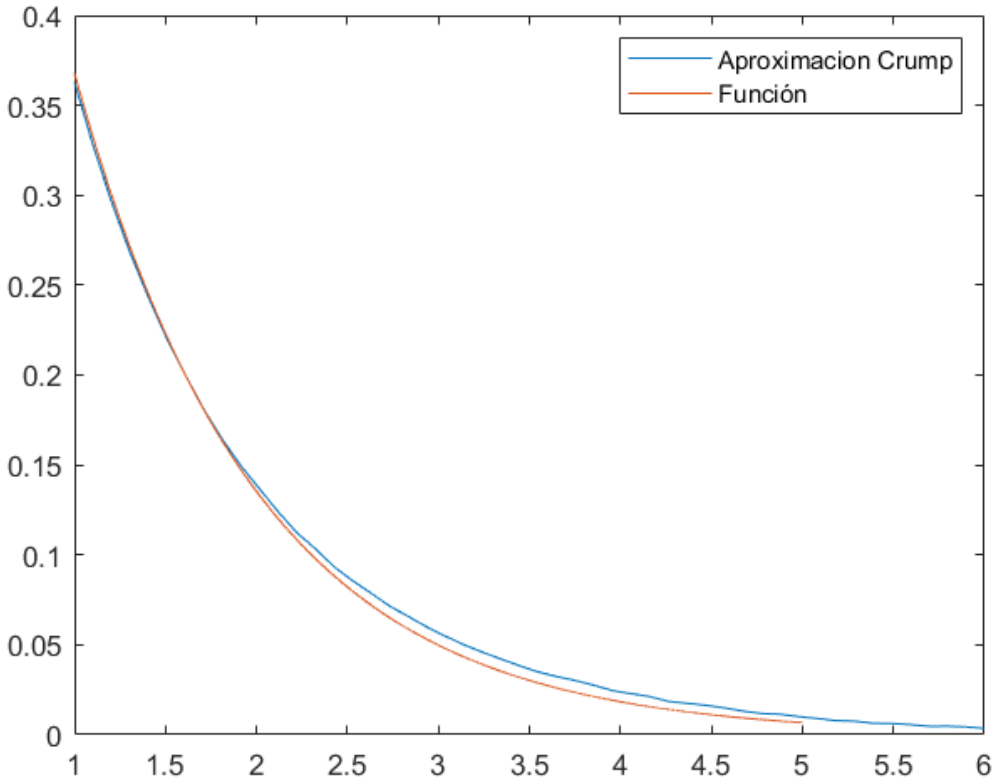


Figura 3.11. Aproximación de la señal $f_2(t) = e^{-t}$ por el método de Stehfest

Sidi

La transformada de Sidi ha permitido obtener resultados satisfactorios para la gran mayoría de funciones propuestas, con la excepción de $f_1(t)$, $f_7(t)$ y $f_{13}(t)$. Estos resultados tienen una precisión de dos y tres dígitos de precisión respecto a la función a aproximar.

Este método no ha presentado ningún inconveniente inesperado, ya que no presenta comportamientos atípicos en ninguno de los casos en los que sí se espera aproximación ni oscilaciones respecto a la función como se puede observar en el método de Crump según se muestra en la figura 3.6. A continuación se presenta un par de aproximaciones realizadas mediante este método

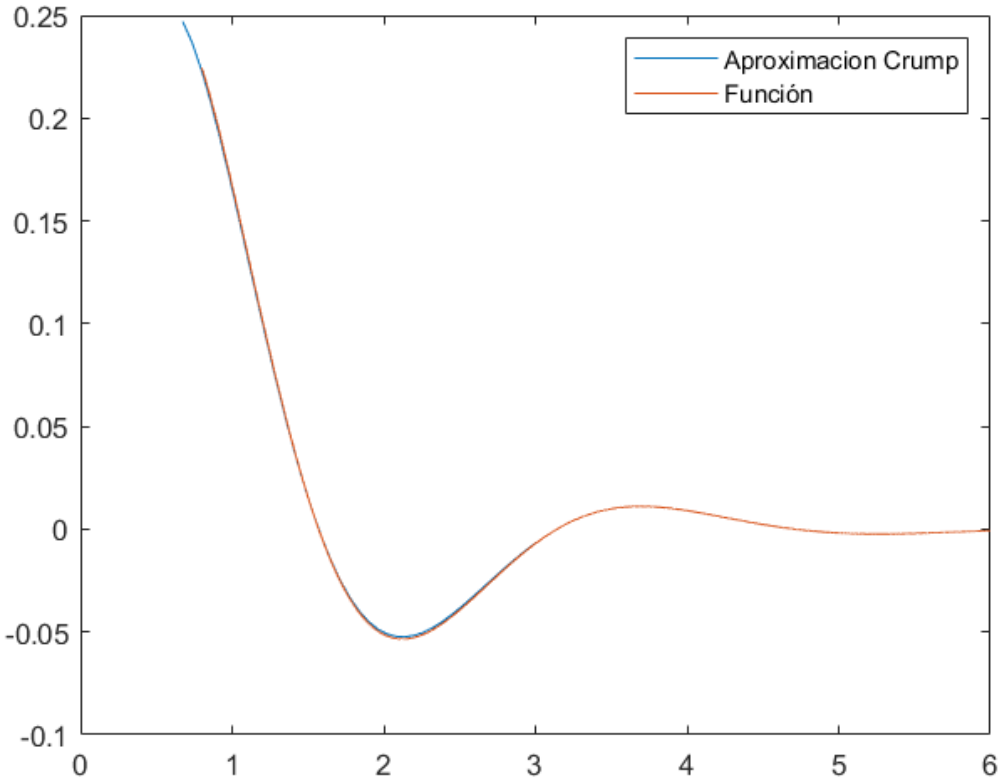


Figura 3.12. Aproximación de la señal $f_{12}(t) = \frac{1}{2}e^{-t}\sin 2t$ por el método de Sidi

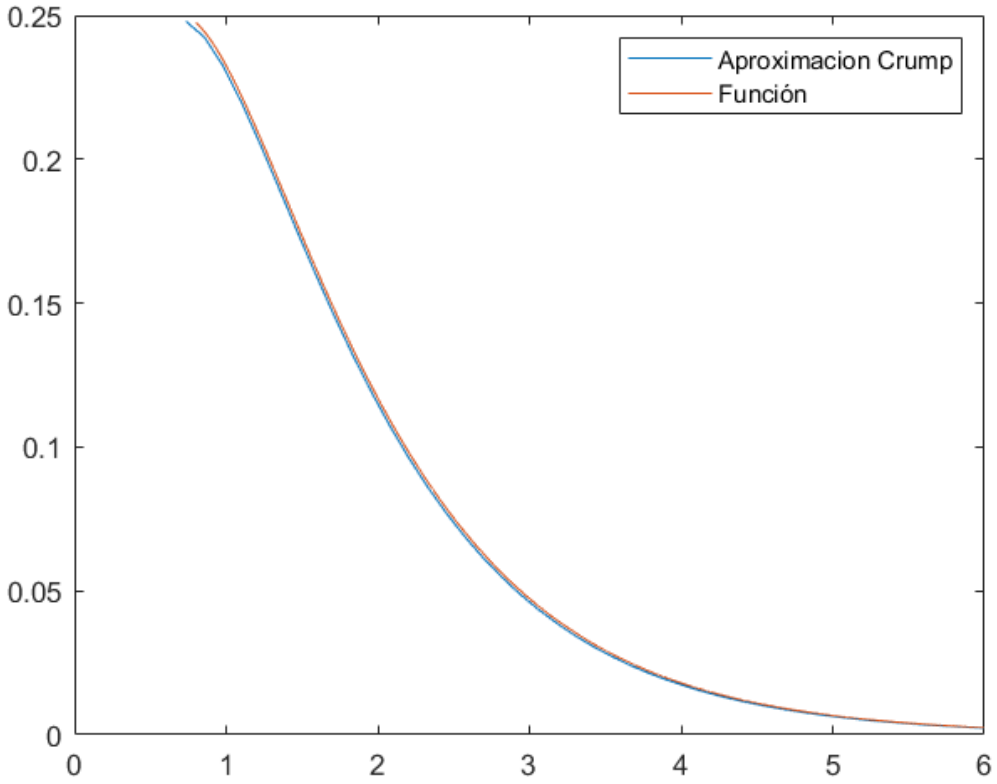


Figura 3.13. Aproximación de la señal $f_9(t) = e^{-t} - e^{-2t}$ por el método de Sidi

4.-Manual de usuario y de instalación

4.1.-Manual de instalación

Para instalar la aplicación se necesita tener instalada una versión de Matlab, preferiblemente la versión R2019 que es en la que se ha desarrollado la aplicación.

Una vez instalada la aplicación de Matlab la instalación es breve, se selecciona el paquete en el que está comprimido el código del programa bajo el nombre InversaLaplace.mlappinstall y una vez abierto aparecerá una ventana como la mostrada en la figura 4.1.

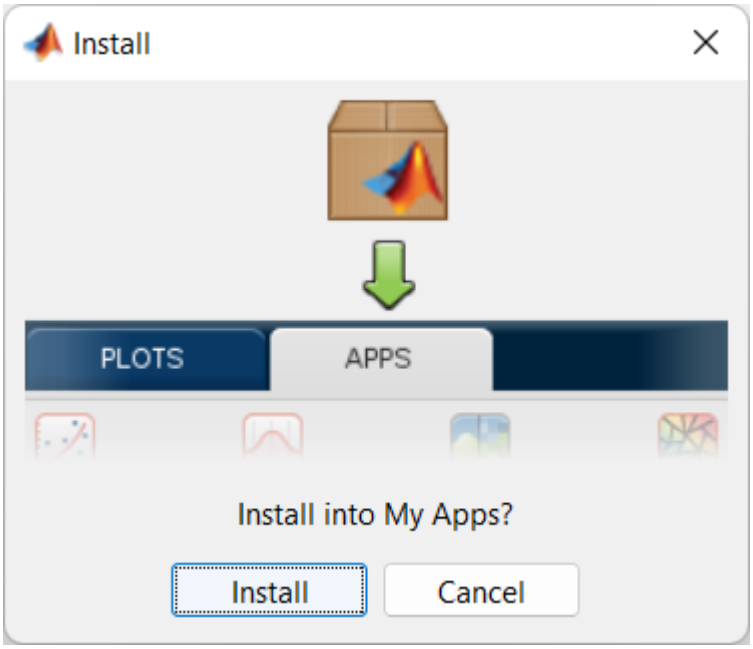


Figura 4.1. Ventana de instalación de la aplicación

Dentro de esta ventana se selecciona la opción Install y, de manera inmediata, aparece el icono de la aplicación en barra de herramientas de Matlab, en la pestaña Apps. De este modo la aplicación aparece disponible como una herramienta adicional de Matlab siempre que sea necesaria.

4.2.-Guía del usuario

Para trabajar con la aplicación en su estado actual hay que seguir una serie de pasos en función de la señal a transformar o el método que se desea utilizar, con ligeras diferencias las cuales pueden llegar a ser críticas en caso de no seleccionarlas adecuadamente.

Paso 1. Para abrir la aplicación se accede a la versión de Matlab en la que se ha instalado la aplicación (recomendablemente R2019, aunque también funciona en R2016). Una vez abierto y si se han seguido los pasos de la instalación, en la pestaña APPS de la barra de herramientas aparece la aplicación con el nombre InversaLaplace.

Paso 2. Antes de comenzar a manejar la aplicación, se debe elegir el método que se desea utilizar para calcular la transformada inversa. Esta acción es probablemente la más relevante, ya que en función del método seleccionado se debe escoger un tipo de datos en un formato de entrada u otro.

A partir de este punto se detallan los pasos a seguir para cada método ya que cada uno tiene ciertas peculiaridades determinadas y los pasos a seguir o los datos a introducir pueden variar.

A.-Método de Crump

Paso 3.A. Una vez se decide utilizar el método de Crump, lo primero a hacer es dirigirse a la pestaña “Inputs” y en el menú desplegable “Tipo de señal” seleccionar la opción “Datos de archivo”. Este menú se encuentra en la parte inferior izquierda.

Paso 4.A. Seleccionada la opción se muestra un panel nuevo en la parte inferior derecha de la aplicación con el nombre “Cargar Expresión”. En este panel hay diferentes campos que deben cumplir una serie de condiciones y generar una matriz de datos que aproxime adecuadamente:

- El valor C de la expresión de Crump debe estar comprendido en el intervalo formado por los campos “Inicio Eje Real” y “Final Eje Real”. Este valor se necesita para poder evaluar la función en ese punto y seleccionar una columna de datos con la parte real fija.
- El valor 0 debe estar comprendido en el intervalo formado por los campos “Inicio Eje Imaginario” y “Final Eje Imaginario”. Para evaluar la función en el punto C, solo

en el campo de los números reales. Hay que considerar que a mayor sea el valor introducido en "Final Eje Imaginario" mayor puede ser la precisión solicitada.

$$Final\ Eje\ Imaginario \geq \frac{Precisión \times \pi}{T}$$

Donde $T > 2Final\ señal\ tiempo$, comentado en el apartado 1.2.7, en la sección Métodos que emplean la Transformada de Fourier

Una vez se tienen en cuenta estas condiciones, se introducen unos valores en estos campos y la expresión a invertir se introduce en el campo "f(s)=" en función de s minúscula.

Paso 5.A. Se cambia a la pestaña "Outputs" y en el menú desplegable del panel "Tipo de Transformada" seleccionar la opción "Crump". Se despliega un panel donde introducir los datos que determinan la señal de salida

Paso 6.A. Se van rellenando los campos disponibles, con la única peculiaridad que "Final Señal" y "Precisión" deben ser seleccionados respecto a la expresión (ANTERIOR) y que el valor de "Ultima singularidad" se corresponda con el valor C. Se presiona el botón dentro del panel y, si se han elegido bien los parámetros, en el gráfico señal de salida debe salir la señal aproximada.

Paso 7. El guardado de la señal es igual para todos los métodos. Una vez se tiene la señal representada en la interfaz de salida se selecciona el menú desplegable en el panel "Formato de guardado", con estas opciones disponibles: imagen .png, gráfico .fig o datos .csv. Se selecciona una opción, se le da un nombre que no contenga el símbolo "/" y se pulsa el botón "Guardar archivo". En el directorio donde está trabajando Matlab aparece la figura con el nombre y formato indicado.

B.-Método de Levin

Paso 3.B. Una vez se decide utilizar el método de Levin, lo primero a hacer es dirigirse a la pestaña "Inputs" y en el menú desplegable "Tipo de señal" seleccionar la opción "Datos de archivo". Este menú se encuentra en la parte inferior izquierda.

Paso 4.B. Seleccionada la opción se muestra un panel nuevo en la parte inferior derecha de la aplicación con el nombre "Cargar Expresión". En este panel hay diferentes campos que deben cumplir una serie de condiciones y generar una matriz de datos que aproxime adecuadamente:

- El valor C de la expresión de Levin debe estar comprendido en el intervalo formado por los campos “Inicio Eje Real” y “Final Eje Real”. Este valor se necesita para poder evaluar la función en ese punto y seleccionar una columna de datos con la parte real fija.

Una vez se tienen en cuenta estas condiciones, se introducen unos valores en los campos disponibles y la expresión a invertir se introduce en el campo “f(s)=” en función de s minúscula.

Paso 5.B. Se cambia a la pestaña “Outputs” y en el menú desplegable del panel “Tipo de Transformada” seleccionar la opción “Levin”. Se despliega un panel donde introducir los datos que determinan la señal de salida

Paso 6.B. Se van rellenando los campos disponibles, con la única peculiaridad de que la “Precisión” debe ser menor que el mayor valor imaginario introducido en los datos y que el valor de “Ultima singularidad” se corresponda con el valor C. Se presiona el botón dentro del panel y, si se han elegido bien los parámetros, en el gráfico señal de salida debe salir la señal aproximada.

El guardado de la señal es igual para todos los métodos, explicado en la sección de Crump paso 7.

C.-Método de Stehfest-Gaver

Paso 3.C. Una vez se decide utilizar el método de Stehfest, lo primero a hacer es dirigirse a la pestaña “Inputs” y en el menú desplegable “Tipo de señal” seleccionar la opción “Señal Real”. Este menú se encuentra en la parte inferior izquierda.

Paso 4.C. Seleccionada la opción se muestra un panel nuevo en la parte inferior derecha de la aplicación con el nombre “Cargar Parte Real”. En este panel hay un único botón que al presionarlo abre la carpeta en la que se esté ejecutando Matlab, de manera que se seleccione un archivo en formato .fig o .csv. La estructura del archivo .csv se basa en dos columnas en las que se almacenan los datos del eje X e Y respectivamente mientras que la figura .fig se importa directamente.

Paso 5.C. Se cambia a la pestaña “Outputs” y en el menú desplegable del panel “Tipo de Transformada” seleccionar la opción “Stehfest”. Se despliega un panel donde introducir los datos que determinan la señal de salida.

Paso 6.C. Se rellenan los campos que delimitan el tiempo en la salida y la precisión. Se recuerda que la precisión para este método no ha alcanzado convergencia en la mayoría de las funciones test, de modo que los resultados más próximos se alcanzan para valores bajos de la precisión.

El guardado de la señal es igual para todos los métodos, explicado en la sección de Crump paso 7.

D.-Método de Sidi

Paso 3.D. Una vez se decide utilizar el método de Sidi lo primero a hacer es dirigirse a la pestaña “Inputs” y en el menú desplegable “Tipo de señal” seleccionar la opción “Datos de archivo”. Este menú se encuentra en la parte inferior izquierda.

Paso 4.D. Seleccionada la opción se muestra un panel nuevo en la parte inferior derecha de la aplicación con el nombre “Cargar Expresión”. En este panel hay diferentes campos que deben cumplir una serie de condiciones y generar una matriz de datos que aproxime adecuadamente:

- El valor C de la expresión de Sidi debe estar comprendido en el intervalo formado por los campos “Inicio Eje Real” y “Final Eje Real”. Este valor se necesita para poder evaluar la función en ese punto y seleccionar una columna de datos con la parte real fija. Es recomendable escoger un valor muy próximo a 0 en “Inicio Eje Real” ya que el valor máximo que se necesita evaluar en la parte imaginaria de la función viene definido por $Precisión * 2\pi/t_{min}$, de modo que, a menor valor introducido como inicio del eje real, mayor debe ser el abanico de números imaginarios contemplados.

Una vez se tienen en cuenta estas condiciones, se introducen unos valores en los campos disponibles y la expresión a invertir se introduce en el campo “f(s)=” en función de s minúscula.

Paso 5.D. Se cambia a la pestaña “Outputs” y en el menú desplegable del panel “Tipo de Transformada” seleccionar la opción “Levin”. Se despliega un panel donde introducir los datos que determinan la señal de salida

Paso 6.D. Se van rellenando los campos disponibles, con la única peculiaridad de que la “Precisión” debe ser menor que el mayor valor imaginario introducido en los datos y que el valor de “Ultima singularidad” se corresponda con el valor C. Se presiona el botón dentro del panel y, si se han elegido bien los parámetros, en el gráfico señal de salida debe salir la señal aproximada.

El guardado de la señal es igual para todos los métodos, explicado en la sección de Crump paso 7.