GUÍA PARA ESTUDIANTES

Generación de Figuras para Investigación

Manuel Martínez Gómez

Versión: 1.0.2

Prefacio de la Serie

Este documento forma parte de una serie de guías diseñadas para apoyar el proceso formativo de estudiantes de ingeniería (especialmente ingeniería eléctrica). La colección completa actualizada se encuentra disponible de manera gratuita en mi sitio web personal.

Motivación y Audiencia

En lo que va de mi carrera académica—guiando trabajos de titulación, así como revisando y redactando artículos científicos—he observado un patrón recurrente: los estudiantes suelen invertir una cantidad desproporcionada de tiempo y energía enfrentándose a problemas operativos o de formato. Ya sea al intentar generar una figura, formatear un documento o estructurar una revisión bibliográfica, estos obstáculos generan frustración y diluyen el esfuerzo intelectual.

El propósito principal de esta serie de documentos es estandarizar y facilitar estos procesos mediante indicaciones precisas. El objetivo es que puedas dedicar tu esfuerzo a lo que verdaderamente importa en la academia: la investigación y el desarrollo de soluciones.

El contenido del presente material se entrega a modo de recomendación y buenas prácticas para la comunidad estudiantil. Sin embargo, **«para los estudiantes de mis cursos y aquellos bajo mi tutela de tesis, los lineamientos aquí expuestos son de carácter obligatorio»**, lamentablemente.

Convenciones de Lectura

Para facilitar la lectura de las guías de esta serie, he estandarizado ciertos elementos visuales y tipográficos. Es fundamental comprender su propósito antes de iniciar la lectura:

Recomendación del Autor

A lo largo de los textos, encontrarás recuadros como este. Indican **recomendaciones** extraídas de mis experiencias en el laboratorio y en la redacción científica.

Nota

También verás recuadros como este. Indican **notas** importantes o advertencias críticas que deben tomarse en cuenta para evitar errores.

Adicionalmente, se utilizan las siguientes convenciones tipográficas:

- El texto en tipografía `monospaciada` (tipo máquina de escribir) se utiliza para indicar fragmentos de código, comandos, nombres de archivos o rutas de carpetas.
- El texto en *cursiva* se emplea para enfatizar un concepto clave por primera vez, o para referirse a términos técnicos en idiomas extranjeros.
- El texto en **negrita** se utiliza en estas guías para destacar conceptos de vital importancia. Cabe señalar que, pese a que el estándar IEEE prohíbe su uso para dar énfasis en un artículo científico, me he tomado la libertad editorial de utilizarlas con cierta frecuencia. El objetivo de esta licencia es puramente pedagógico y para facilitar la lectura.

Cualquier término técnico, definición específica o marco conceptual necesario para comprender el tema particular de esta guía será definido de forma precisa en la sección de Introducción.

Control de Versiones y Retroalimentación

Los documentos académicos y técnicos están en constante evolución. Dado que este material se distribuye en formato PDF, es posible que la copia que estés leyendo no sea la más reciente. Para asegurarte de contar con la última versión, te sugiero visitar periódicamente el sitio https://manuelmartinez.cl/es/docencia.

Si durante la lectura encuentras algún error tipográfico, un enlace roto, o tienes alguna sugerencia de mejora, te invito a contactarme. La retroalimentación de los estudiantes es el motor que permite mantener la calidad y utilidad de estos recursos.

Manuel Martínez Gómez
Profesor Asistente, Universidad de O'Higgins
Agosto, 2026

I. Introducción

El propósito de esta guía es estandarizar y facilitar el proceso de creación de figuras con calidad de publicación para artículos científicos, memorias de pregrado y tesis de postgrado en ingeniería eléctrica, basándose en los estándares del IEEE (*Institute of Electrical and Electronics Engineers*) [1]. A lo largo de mi carrera, he notado que los estudiantes invierten una cantidad desproporcionada de tiempo intentando dar formato a sus gráficos, no necesariamente obteniendo buenos resultados.

El desarrollo de esta guía sigue un proceso de dos partes: extracción de datos, procesamiento y graficación, para continuar con discusiones sobre cómo implementar los estándares de calidad IEEE y cómo realizar retoques finales. Antes de comenzar con las indicaciones, es fundamental definir algunos términos relevantes para unificar el lenguaje y evitar confusiones comunes:

Imagen: Elemento visual genérico, como una fotografía, una captura o un diagrama conceptual de bloques. No es el foco principal de esta guía.

Gráfico: Representación cuantitativa que contiene líneas, formas de onda o superficies para describir un fenómeno estudiado. Generalmente posee ejes coordenados y es generado estrictamente a partir de datos numéricos (como los exportados de simulaciones).

Figura: El contenedor formal (como el entorno `figure` en LaTeX) que se inserta en el cuerpo del documento de investigación. Agrupa uno o más gráficos e imágenes junto con su respectiva descripción (*caption*). Cuando se divide en partes, estas se denominan subfiguras o paneles.

II. Extracción de Datos

Para lograr una figura de alta calidad, debe evitarse utilizar capturas de pantalla o fotos del *software* de simulación. El primer paso siempre es exportar los datos numéricos de la simulación a formatos de texto estándar, como CSV (*Comma-Separated Values*), para luego procesarlos. Notar que *software* como Matlab ofrece formatos dedicados para guardar datos como el `.mat` .

- **PLECS:** Desde cualquier bloque *Scope*, ir a `File > Export > as CSV`. Esto genera un archivo de texto con columnas de tiempo y amplitud, fácil de leer en cualquier lenguaje [2]. Para más información ver Using the PLECS Scope.
- **MATLAB/Simulink:** El método clásico es usar el bloque *To Workspace* configurado en formato `Timeseries` o `Array`. Un método más moderno y robusto es utilizar el *Simulation Data Inspector*. Al simular, las señales marcadas para *Logging* se guardan en el objeto `out` del Workspace, desde donde se pueden exportar directamente a CSV o pasar a un script [3], [4]. Un resumen del proceso se presenta en How to Export Data From SIMULINK to MATLAB. Un análisis en profundidad de los métodos disponibles para exportar *data* en Matlab se presenta en [5], y se resume en parte en la siguiente imagen:
- **Equipos de Laboratorio:** Los osciloscopios y controladores embebidos (e.g. DSPACE, TI C2000) permiten guardar las capturas en memorias externas o enviar los datos por puerto serial, generalmente en formato `.csv`.

Recomendación del Autor

Reducción de Muestras (*Downsampling*): Los osciloscopios y simuladores de paso variable pueden exportar millones de puntos de datos. Graficar 10 millones de puntos vectoriales hará que el PDF final pese decenas de megabytes y el compilador de LaTeX colapse. Siempre evalúa utilizar comandos como `downsample` en MATLAB o recortes mediante *slicing* en Python (ej. `df.iloc[::10, :]`) para reducir los puntos a una cantidad razonable (ej. 10.000 puntos) antes de exportar el gráfico vectorial.

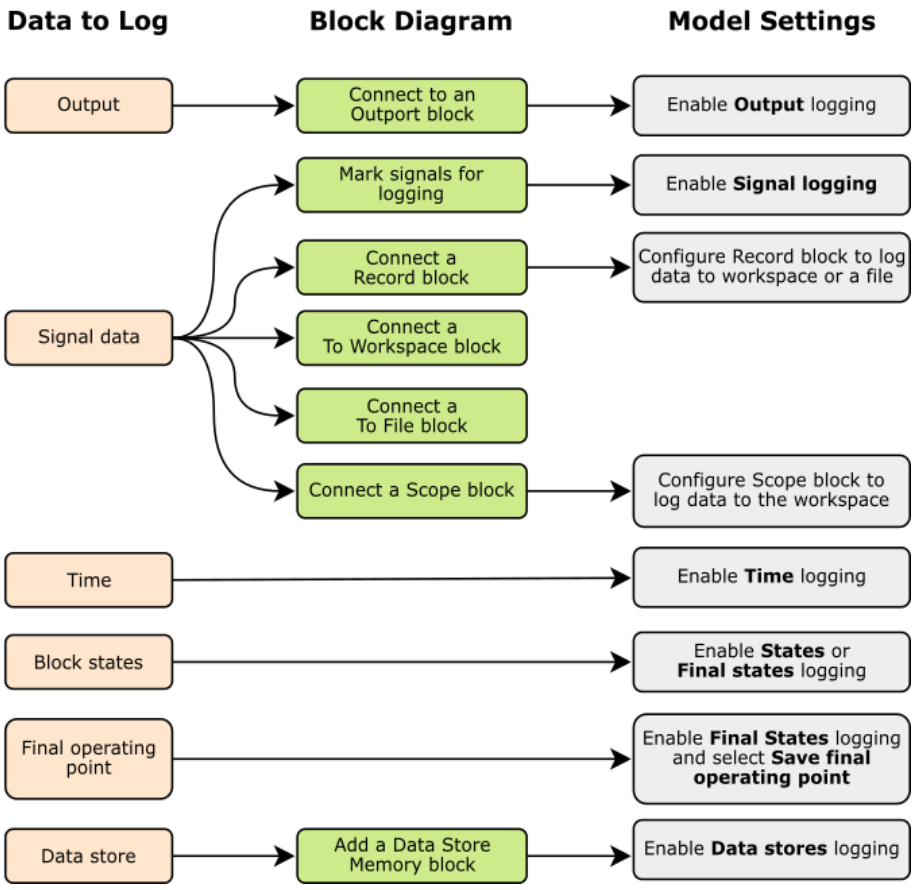

FIGURA 1: Descripción general de las técnicas y ajustes de configuración necesarios según el tipo de datos a guardar. Reproducido de [5].

III. Procesamiento y Graficación

Una vez obtenidos los datos, se debe generar un *script* para graficar (generar un *plot*). Esto asegura que el gráfico sea 100 % reproducible si los datos de simulación cambian en el futuro, y también permite que se pueda generar la figura siguiendo estándares específicos de revistas científicas de corriente principal (como el formato IEEE [1]). El proceso de graficar usualmente conlleva a generar una ventana o vista previa; luego, el gráfico debe ser exportado a un formato que pueda ser insertado al documento de investigación deseado. Dependiendo del entorno (programa) que se quiera utilizar, existen procedimientos diferentes.

Nota

Los formatos de archivo para figuras aceptados por IEEE son PDF, PNG, PS, EPS, y TIFF.

3.1 Camino A: Utilizando MATLAB

El procedimiento estándar en MATLAB es crear código en un archivo `.m` usando la función `plot` para generar el gráfico en una ventana. Luego, si es necesario, se pueden hacer ajustes manuales en la ventana generada. También es importante mencionar que desde la ventana generada, se puede guardar el archivo en formato `.fig` (exclusivo de Matlab) para posteriores ediciones.

- **Precaución con la ventana `.fig`:** Es vital no maximizar ni alterar el tamaño de la ventana antes de exportar. MATLAB guarda el archivo con el tamaño exacto que tiene en pantalla, lo que deforma las proporciones del texto respecto a las líneas.
- **Exportación moderna:** Para evitar el problema anterior, se recomienda usar el comando `exportgraphics` en el script (disponible desde R2020a). Esto permite exportar el gráfico directamente a PDF recortando los bordes blancos automáticamente.
- **Subgráficos (subplots):** En lugar del antiguo comando `subplot`, se recomienda el uso de `tiledlayout` (introducido en R2019b), ya que maneja los márgenes y el espaciado entre gráficos de manera más eficiente y estética. Para más información ver How to Make Subplots in MATLAB using Tiledlayout y la documentación TiledLayout en MATLAB.

3.2 Camino B: Utilizando Python (Open Source)

Python es una alternativa sumamente potente. El estándar en la comunidad científica es usar la combinación de librerías `pandas` (para manipular datos) [6] y `matplotlib` (para graficar) [7], [8]. Python tiene como ventaja que se pueden configurar los parámetros globales al inicio del script (`plt.rcParams`) para forzar que todos los gráficos de la tesis o artículo tengan exactamente el mismo tamaño, tipo de fuente y grosor de línea.

- Se utiliza `pandas.read_csv()` para cargar los datos en un *DataFrame*.
- Se utiliza `matplotlib.pyplot` para dibujar las curvas (ver Tutorials - Matplotlib).
- Se utiliza el comando `plt.savefig` para la exportación. Al incluir el parámetro `bbox_inches='tight'` se eliminan los márgenes blancos, lo cual es ideal para integrar el gráfico en LaTeX.

3.3 Camino C: Utilizando PGFPlots (Nativo en LaTeX)

Para una integración estética insuperable, se puede prescindir de exportar PDFs y trazar los archivos CSV directamente dentro del documento LaTeX usando el paquete `pgfplots` (galería de ejemplos de PGFPlots) [9]. Esto garantiza que la fuente tipográfica de la figura sea exactamente la misma que la del documento. Sin embargo, su curva de aprendizaje es empinada [10] y ralentiza la compilación si los CSV tienen demasiados datos.

IV. Estándares de Calidad y Formato IEEE

Para asegurar que las figuras sean aceptadas en artículos científicos y luzcan profesionales, se deben seguir reglas estrictas de formato. Si bien cada editorial o institución a la que se adscribe un documento de investigación tiene su propio formato, las líneas generales que imponen los estándares del IEEE [1] son una buena referencia para que las figuras tengan una calidad superior.

4.1 Gráficos Vectoriales vs. Rasterizados

Se recomienda fuertemente exportar en formatos vectoriales (`.pdf` o `.svg`) en lugar de formatos *rasterizados* o de mapa de bits (`.png`, `.jpg`). Un archivo vectorial consiste en instrucciones matemáticas, permitiendo un *zoom* infinito sin pérdida de calidad, resultando en archivos livianos [11].

Recomendación del Autor

Exporta de Python/MATLAB a `.svg`, y si se requiere edición manual, utiliza Inkscape. Luego, guarda la versión final como `.pdf` para insertarla en LaTeX. El archivo PDF es compatible de forma nativa con los compiladores (pdfLaTeX, XeLaTeX) y no ralentiza la visualización del documento compilado.

4.2 Resoluciones y Dimensiones

Aunque IEEE exige 600 DPI (*Dots Per Inch* o puntos por pulgada) para gráficos de líneas [1], al utilizar formatos vectoriales este concepto deja de ser relevante (por la resolución infinita). En cuanto a los anchos, IEEE estandariza aproximadamente 3.5 pulgadas (88.9 mm) para figuras de una columna y 7.16 pulgadas (181.6 mm) para doble columna. Recordar que las figuras pueden ser reescaladas en el documento para alcanzar los tamaños definitivos.

Recomendación del Autor

Dimensiones: Resulta conveniente generar cada gráfico con el ancho de doble columna que indica IEEE dado que nunca se sabe si tendrá que cambiar la presentación de la figura más adelante. En cuanto al alto, recomiendo utilizar una proporción cercana a 6.3 pulgadas de ancho por 10.1 pulgadas de alto. Suelo disminuir el alto de cada gráfico de una figura proporcionalmente a la cantidad de subfiguras que irán alineadas verticalmente. Recordar que generar figuras más grandes de lo necesario producirá mayor peso en el documento (porque el rescalado visual no modifica el tamaño del archivo).

4.3 Fuentes Tipográficas

Debido a la dificultad en controlar los tamaños de cada texto dentro de las figuras, este apartado es quizás el que mayor libertad tiene dentro del estándar de IEEE y otras editoriales. En general, se exige que las tipografías y los tamaños de letra sean consistentes, con un tamaño aproximado de 9-10 puntos al visualizarse a tamaño completo [1]. Esto es algo complejo de medir si la figura tendrá reescalados. En cuanto al tipo de fuente para el texto, IEEE da recomendaciones mas no exigencias, señalando *Helvetica*, *Times New Roman*, *Arial* o *Cambria* como deseables (las cuales tienen buena legibilidad en tamaños pequeños).

Recomendación del Autor

Tamaño y familia de fuente: Utiliza siempre fuentes de la familia *sans-serif* (sin remates) como Helvetica, Arial, o la fuente nativa del template de IEEE. Para el tamaño, fijar 9 pt en una figura escalada para tamaño del documento. Esto sería 14 pt si se sigue la recomendación del punto anterior sobre dimensiones. La justifyicación de esto es que, al insertar una figura de 6.3 pulgadas de ancho en una columna de IEEE (3.5 pulgadas), el gráfico se encogerá; una fuente inicial de 14 pt se reducirá a aproximadamente 8 pt, lo cual cumple en la práctica el estándar de tamaño mínimo.

4.4 Colores, Estilos de Línea y Leyendas

Se debe evitar depender exclusivamente del color para diferenciar las señales.

- **Impresión en blanco y negro:** Muchos evaluadores imprimen los artículos. IEEE estipula que las figuras deben ser comprensibles en escala de grises. Se debe utilizar variaciones en el estilo de línea (continua, punteada, guiones `--`, `-.`) y marcadores (`-o`, `-x`) además del color.

- **Daltonismo (accesibilidad):** Evita usar las combinaciones rojo y verde simultáneamente para curvas superpuestas. En Python, el *colormap* por defecto *viridis* o *tab10* son amigables con el daltonismo.
- **Leyendas:** La leyenda del gráfico (*legend*) es un elemento muy exigido por evaluadores en revistas IEEE. Su principal función es la de señalizar el significado de cada trazo. Una buena práctica es asociar el trazo directamente con el símbolo de la variable que se está graficando. La leyenda no debe tener un recuadro oscuro que encierre las variables. Además, se debe ubicar estratégicamente para no tapar los datos relevantes.

4.5 Títulos, Ejes y Captions

Es indispensable recordar los siguientes lineamientos de formato IEEE:

Nota

Cero títulos: Un gráfico científico nunca lleva título en la parte superior del recuadro (no usar comandos como *title* en el *script*). Toda descripción que pudiese ir en un título debe ir en la descripción de la figura (*caption*).

- **Ejes claros:** Los ejes siempre deben nombrar la variable y su unidad entre paréntesis o corchetes (ej. `Voltaje del bus DC (V)`).
- **El *caption*:** Todo lo que describe la figura debe ir en el *caption* gestionado por LaTeX. Un buen *caption* explica el contexto, por ejemplo: "*Formas de onda ante una respuesta en escalón. Arriba: Voltaje. Abajo: Corriente.*"

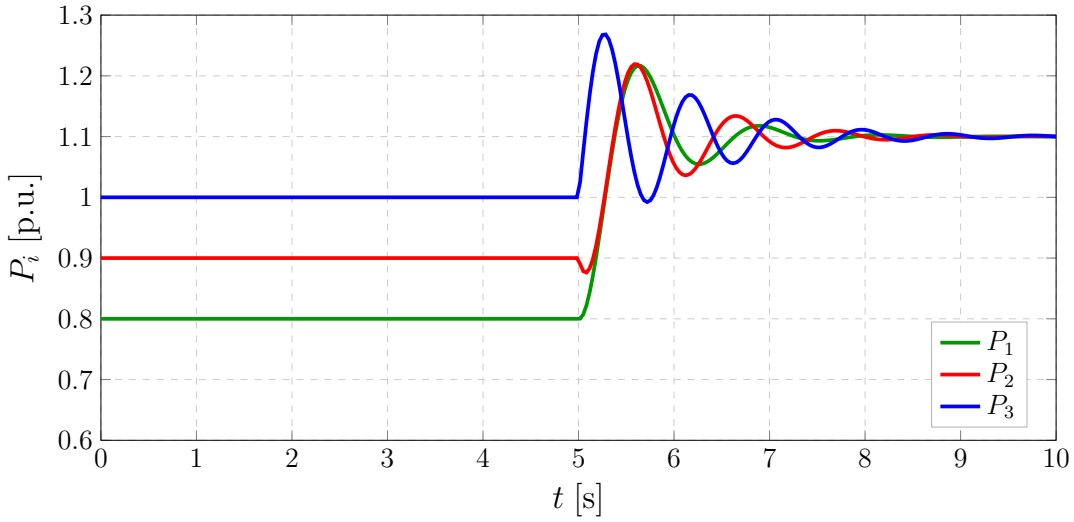

FIGURA 2: Formas de onda de potencia de salida de los conversores ante impacto de carga resistiva.

V. Retoques Estéticos y Anotaciones

En ingeniería eléctrica y ciencias exactas, es común necesitar indicar eventos específicos en los gráficos, como zonas de régimen permanente, tiempos de establecimiento o impactos de carga. Para ayudar al lector, además de señalar estos eventos en el texto principal, es muy recomendable (y a veces exigido) que se incorporen etiquetas y marcadores en los gráficos.

Recomendación del Autor

En lugar de programar flechas, recuadros y llaves complejas mediante código puro, resulta mucho más eficiente utilizar *software* de diseño gráfico vectorial externo para este último nivel de detalle.

- **Inkscape**: Es la herramienta de código abierto recomendada por excelencia [12]. Permite abrir el `.pdf` o `.svg` generado por el *script* de MATLAB/Python y editar cada línea o texto por separado, respetando las capas. Es ideal para agregar cuadros de *zoom* o cotas.
- **draw.io (diagrams.net)**: Es una herramienta online que no requiere instalación y cuenta con muchas figuras prearmadas en librerías. Aunque es excelente para diagramas de bloques y diagramas de flujo, para formas de onda complejas exportadas desde código, Inkscape suele ser superior ya que gestiona mejor los metadatos vectoriales.

Referencias

[1] IEEE, "Create graphics for your article," Institute of Electrical and Electronics Engineers, 2026, accessed: Jul. 26, 2026. [Online]. Available: https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-journal-article/create-graphics-for-your-article/

[2] Plexim GmbH, *PLECS User Manual*, Plexim, 2026. [Online]. Available: https://docs.plexim.com/plecs/latest/

[3] MathWorks, *MATLAB Documentation*, The MathWorks, Inc., 2023. [Online]. Available: https://la.mathworks.com/help/matlab/

[4] MATLAB, "How to use the simulink scope block," YouTube video, 2026, accessed: Jul. 26, 2026. [Online]. Available: https://www.youtube.com/watch?v=KF-_gppJGF0

[5] MathWorks, "Matlab documentation: Save simulation data," The MathWorks, Inc., 2026, accessed: Jul. 26, 2026. [Online]. Available: https://la.mathworks.com/help/simulink/ug/export-simulation-data-1.html

[6] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56.

[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[8] J. Hunter, D. Dale, E. Firing, and M. Droettboom, "Tutorials - matplotlib 3.11.1 documentation," The Matplotlib development team, 2026, accessed: Jul. 26, 2026. [Online]. Available: https://matplotlib.org/stable/tutorials/index.html

[9] C. Feuersänger, *Manual for Package PGFPlots*, 2020. [Online]. Available: http://pgfplots.sourceforge.net/pgfplots.pdf

[10] Overleaf, "Pgfplots package," Overleaf, 2026, accessed: Jul. 26, 2026. [Online]. Available: https://es.overleaf.com/learn/latex/Pgfplots_package

[11] S. Murray, *Interactive Data Visualization for the Web*. O'Reilly Media, Inc., 2013.

[12] Inkscape Project, *Inkscape: Guide to a Vector Drawing Program*, 2023. [Online]. Available: https://inkscape.org/learn/books/

I. Anexos: Códigos de Ejemplo

1.1 Plantilla de Código en MATLAB (.m)

El siguiente script asume que los datos fueron exportados previamente a un archivo `datos_simulacion.csv`
Se incluye el uso de `tiledlayout` para múltiples gráficas.

Código 1: Plantilla MATLAB para gráficos IEEE

```
% ========================================================================
% Plantilla MATLAB - Graficas formato IEEE
% ========================================================================

% 1. Cargar datos
% Se asume un archivo CSV con tiempo en col 1, voltaje col 2, corriente col 3
data = readmatrix('datos_simulacion.csv');
% Reduccion de datos (Downsampling) si hay mas de 100k puntos
data = downsample(data, 10);
tiempo = data(:, 1);
voltaje = data(:, 2);
corriente = data(:, 3);

% 2. Crear figura con dimensiones especificas (Recomendacion personal)
fig = figure('Units', 'inches', 'Position', [1, 1, 6.3, 5]); % Ajustar alto
    segun necesidad

% 3. Usar TiledLayout para subplots limpios (Recomendado sobre subplot
    tradicional)
t = tiledlayout(2, 1, 'TileSpacing', 'compact', 'Padding', 'tight');

% --- Grafica Superior (Voltaje) ---
ax1 = nexttile;
plot(tiempo, voltaje, 'LineWidth', 1.5, 'Color', '#0072BD', 'LineStyle', '-');
ylabel('Voltaje (V)', 'FontName', 'Helvetica', 'FontSize', 14);
grid on; ax1.FontName = 'Helvetica'; ax1.FontSize = 14;
% Ocultar etiquetas X de la superior para ahorrar espacio
xticklabels(ax1, {});

% --- Grafica Inferior (Corriente) ---
ax2 = nexttile;
plot(tiempo, corriente, 'LineWidth', 1.5, 'Color', '#D95319', 'LineStyle', '--
    ');
```

```
ylabel('Corriente (A)', 'FontName', 'Helvetica', 'FontSize', 14);
xlabel('Tiempo (s)', 'FontName', 'Helvetica', 'FontSize', 14);
grid on; ax2.FontName = 'Helvetica'; ax2.FontSize = 14;

% Vincular ejes X para hacer zoom de manera conjunta (Opcional)
linkaxes([ax1, ax2], 'x');

% Nota: NO agregar title('...'). IEEE prohibe titulos dentro de la figura.

% 4. Exportar a PDF vectorizado
exportgraphics(fig, 'figura_matlab.pdf', 'ContentType', 'vector');
```

1.2 Plantilla de Código en Python (.py)

El siguiente script utiliza las librerías `pandas` y `matplotlib`, demostrando cómo agregar dos variables en un mismo eje diferenciándolas por estilo de línea.

Código 2: Plantilla Python para gráficos IEEE

```python
# ==============================================================================
# Plantilla Python (Matplotlib) - Graficas formato IEEE
# ==============================================================================
import pandas as pd
import matplotlib.pyplot as plt

# 1. Configuracion global (Recomendacion personal para estandarizacion)
plt.rcParams.update({
    'font.family': 'sans-serif',
    'font.sans-serif': ['Helvetica', 'Arial'],
    'font.size': 14,                # Tamano base
    'axes.labelsize': 14,
    'xtick.labelsize': 14,
    'ytick.labelsize': 14,
    'legend.fontsize': 12,          # Leyenda ligeramente mas pequena
    'figure.figsize': (6.3, 4.5),   # Ajuste personal para 1 panel ancho
    'lines.linewidth': 1.5
})

# 2. Cargar datos y hacer decimation (slicing) si es muy pesado
df = pd.read_csv('datos_simulacion.csv')
df = df.iloc[::10, :] # Tomar 1 de cada 10 muestras (Downsampling)

# 3. Crear figura y plotear multiples curvas
fig, ax = plt.subplots()

# Diferenciar no solo por color, sino por estilo de linea para B/N
ax.plot(df['Tiempo'], df['Voltaje_Ref'], color='black', linestyle='--', label=
    'Referencia')
ax.plot(df['Tiempo'], df['Voltaje_Med'], color='#1f77b4', linestyle='-', label
    ='Medicion')

# 4. Formatear ejes y agregar leyenda (sin borde)
ax.set_xlabel('Tiempo (s)')
ax.set_ylabel('Voltaje del bus DC (V)')
ax.grid(True, linestyle=':', alpha=0.7)

```

```
# Leyenda sin marco (frameon=False) como dictan las buenas practicas
ax.legend(frameon=False, loc='best')

# 5. Exportar a PDF vectorizado
plt.savefig('figura_python.pdf', format='pdf', bbox_inches='tight')
plt.show()
```