



VilTeX

paquete integral para
la generación automatizada
de documentos LaTeX
y exportación TikZ
en **Wolfram Mathematica**

`(begin(tikzpicture))`

Enrique Vílchez Quesada

Enrique Vílchez Quesada

V_il^T_EX: paquete integral para la generación automatizada

de documentos L^AT_EX

y exportación T_ikZ en ***Wolfram Mathematica***

Copyright©2026

Copyright© Revista digital Matemática Educación e Internet (<https://tecdigital.tec.ac.cr/revistamatematica/>)

Escuela de Matemática

Instituto Tecnológico de Costa Rica

Apdo. 159-7050, Cartago

Teléfono (506) 2550-2225 Fax (506) 2550-2493

Vílchez Quesada, Enrique. **VíT_EX: paquete integral para la generación automatizada de documentos $\LaTeX$ y exportación Tikz en Wolfram Mathematica.**
1ra ed. – Escuela de Matemática, Instituto Tecnológico de Costa Rica. 2026. 201 pp.

ISBN Obra Independiente: 978-9930-656-06-8

1. Renderizado $\LaTeX$.
2. Sistema compilador $\LaTeX$.
3. Sistema *Plot* para $\LaTeX$.
4. Función ExportToTikZ.
5. Función ExportFillingToTikZ.
6. Función ExportGraphToTikZ.
7. Función ExportMaquinaToTikZ.
8. Función ExportAutomataToTikZ.
9. Función ExportToTikZ3D.
10. Función ExportToTikZGraphics3D.
11. Función ExportConicsToTikZ.
12. Casos especiales.
13. Función ExportContour2DToTikZ.
14. Función CreateTikZAnimation.

Este libro se distribuye bajo la licencia Creative Commons: Atribución-NoComercial-SinDerivadas CC BY-NC-ND (la “Licencia”). Usted puede utilizar este archivo de conformidad con la Licencia. Usted puede obtener una copia de la Licencia en <https://creativecommons.org/licenses/by-nc-nd/4.0/>. En particular, esta licencia permite copiado y distribución gratuita, pero no permite venta ni modificaciones de este material.

Límite de responsabilidad y exención de garantía: La persona autora ha hecho su mejor esfuerzo en la preparación de este material. Esta edición se proporciona “tal cual”. Se distribuye gratuitamente con la esperanza de que sea útil, pero sin ninguna garantía expresa o implícita respecto a la exactitud o completitud del contenido.

La Revista digital Matemáticas, Educación e Internet es una publicación electrónica. El material publicado en ella expresa la opinión de sus personas autoras y no necesariamente la opinión de la revista ni la del Instituto Tecnológico de Costa Rica.

CONTENIDO

RESUMEN	VII
ABSTRACT	IX
PRÓLOGO	XI
1 RENDERIZADO $\LaTeX$	1
1.1 Comandos del sistema de renderizado $\LaTeX$	2
1.2 Ejemplos de renderizado $\LaTeX$	4
2 SISTEMA COMPILADOR $\LaTeX$	25
2.1 Comandos del sistema compilador $\LaTeX$	26
2.2 Ejemplos del sistema compilador $\LaTeX$	27
3 SISTEMA <i>Plot</i> PARA $\LaTeX$	31
3.1 Comandos del sistema <i>Plot</i> para $\LaTeX$	32
3.2 Ejemplos del sistema <i>Plot</i> para $\LaTeX$	33
4 FUNCIÓN EXPORTTOTIKZ	41
4.1 Ejemplos de uso de la función ExportToTikZ	43
5 FUNCIÓN EXPORTFILLINGTOTIKZ	59
5.1 Ejemplos de uso de la función ExportFillingToTikZ	61
6 FUNCIÓN EXPORTGRAPHTOTIKZ	67
6.1 Ejemplos de uso de la función ExportGraphToTikZ	70
7 FUNCIÓN EXPORTMAQUINATOTIKZ	75
7.1 Ejemplos de uso de la función ExportMaquinaToTikZ	79

8	FUNCIÓN EXPORTAUTOMATATOTIKZ	83
8.1	Ejemplos de uso de la función ExportAutomataToTikZ	85
9	FUNCIÓN EXPORTTOTIKZ3D	89
9.1	Ejemplos de uso de la función ExportToTikZ3D	92
10	FUNCIÓN EXPORTTOTIKZGRAPHICS3D	115
10.1	Ejemplos de uso de la función ExportToTikZGraphics3D	118
11	FUNCIÓN EXPORTCONICSTOTIKZ	131
11.1	Ejemplos de uso de la función ExportConicsToTikZ	134
12	CASOS ESPECIALES	141
12.1	Ejemplos de los casos especiales	143
13	FUNCIÓN EXPORTCONTOUR2DTOTIKZ	155
13.1	Ejemplos de uso de la función ExportContour2DToTikZ	158
14	FUNCIÓN CREATETIKZANIMATION	169
14.1	Ejemplos de uso de la función CreateTikZAnimation	173
15	COMENTARIOS FINALES	179
	BIBLIOGRAFÍA	181

RESUMEN

El presente trabajo documenta el desarrollo del paquete **V_WTeX**, una herramienta diseñada para automatizar la creación de documentos **TeX** en **Wolfram Mathematica**. A diferencia de otras soluciones parciales disponibles, **V_WTeX** busca integrar de manera completa los procesos de cálculo matemático con la generación de documentación profesional. El paquete surgió de la necesidad recurrente en el ámbito académico de convertir resultados computacionales en documentos de alta calidad tipográfica. **V_WTeX** está organizado en catorce módulos que cubren desde el renderizado básico hasta funcionalidades especializadas como la exportación de diagramas complejos. Las funciones de renderizado permiten convertir expresiones matemáticas en imágenes de alta resolución mediante diferentes configuraciones: matemática básica, física con notación vectorial, química con fórmulas estructurales, y diagramas conmutativos. Esta flexibilidad resulta especialmente útil cuando se requieren documentos con notación especializada. Además, **V_WTeX** incluye herramientas para generar documentos completos de forma automática. Los usuarios pueden crear artículos científicos, reportes técnicos, libros y presentaciones *Beamer* con configuraciones predefinidas que manejan aspectos como bibliografía, diferentes entornos (definiciones, teoremas, ejemplos, entre otros) y estilos de código. El sistema exporta automáticamente los gráficos de **Mathematica** y organiza los archivos en estructuras de carpetas coherentes. Un componente destacado es el sistema de compilación integrado. **V_WTeX** detecta automáticamente los editores **TeX** instalados en el sistema operativo y permite compilar documentos directamente desde **Mathematica**. Esta característica elimina la fragmentación típica del flujo de trabajo entre cálculo y documentación. El módulo de *plotting* con estilo **TeX** representa un avance significativo en la visualización matemática. Incorpora fuentes tipográficas especializadas y permite generar gráficos con la apariencia visual característica de publicaciones académicas. Las capacidades de exportación a **TikZ** constituyen la funcionalidad más innovadora del paquete. **V_WTeX** puede convertir gráficos 2D y 3D de **Mathematica** en código *TikZ/PGFPlots*, preservando la calidad vectorial y permitiendo edición posterior. Los exportadores especializados en **V_WTeX** manejan casos específicos: grafos, autómatas de estado finito, secciones cónicas, y diagramas de teoría de conjuntos. Una extensión revolucionaria de estas capacidades es la función `CreateTikZAnimation`, que permite generar animaciones **TeX** interactivas utilizando el paquete *animate*. Esta herramienta convierte funciones puras de **Mathematica** en animaciones paramétricas que pueden compilarse en documentos PDF con controles de reproducción integrados. La función soporta parámetros continuos y discretos, personalización visual completa, y es compatible con todos los tipos de gráficos de **Mathematica**, desde funciones trigonométricas hasta visualizaciones de datos complejas. También, particularmente útiles son las herramientas para diagramas matemáticos puntuales, que pueden generar automáticamente representaciones de máquinas de estado, problemas de programación lineal con regiones factibles, y diversos tipos de diagramas de *Venn*. Estas funcionalidades abordan necesidades frecuentes en la enseñanza de las matemáticas y ciencias de la computación. El paquete mantiene compatibilidad con las capacidades nativas de **Wolfram Mathematica** mientras simplifica significativamente la interfaz para usuarios de **TeX**.

Palabras clave: Automatización, **TeX**, **TikZ**, visualización matemática, **Wolfram Mathematica**.

ABSTRACT

This work documents the development of the **V_lT_EX** package, a tool designed to automate the creation of $\LaTeX$ documents in **Wolfram Mathematica**. Unlike other partial solutions available, **V_lT_EX** seeks to comprehensively integrate mathematical computation processes with professional documentation generation. The package arose from the recurring need in the academic field to convert computational results into high-quality typographic documents. **V_lT_EX** is organized into fourteen modules that cover from basic rendering to specialized functionalities such as complex diagram export. The rendering functions allow converting mathematical expressions into high-resolution images through different configurations: basic mathematics, physics with vector notation, chemistry with structural formulas, and commutative diagrams. This flexibility is especially useful when documents with specialized notation are required. Additionally, **V_lT_EX** includes tools for automatically generating complete documents. Users can create scientific articles, technical reports, books, and *Beamer* presentations with predefined configurations that handle aspects such as bibliography, various environments (definitions, theorems, examples, among others), and code styles. The system automatically exports **Mathematica** graphics and organizes files into coherent folder structures. A standout component is the integrated compilation system. **V_lT_EX** automatically detects $\LaTeX$ editors installed on the operating system and allows compiling documents directly from **Mathematica**. This feature eliminates the typical fragmentation of the workflow between calculation and documentation. The *plotting* module with $\LaTeX$ style represents a significant advance in mathematical visualization. It incorporates specialized typographic fonts and allows generating graphics with the characteristic visual appearance of academic publications. The TikZ export capabilities constitute the package's most innovative functionality. **V_lT_EX** can convert 2D and 3D **Mathematica** graphics into *TikZ/PGFPlots* code, preserving vector quality and allowing subsequent editing. Specialized exporters in **V_lT_EX** handle specific cases: graphs, finite state automata, conic sections, and set theory diagrams. A revolutionary extension of these capabilities is the `CreateTikZAnimation` function, which enables generating interactive $\LaTeX$ animations using the *animate* package. This tool converts pure **Mathematica** functions into parametric animations that can be compiled into PDF documents with integrated playback controls. The function supports continuous and discrete parameters, complete visual customization, and is compatible with all types of **Mathematica** graphics, from trigonometric functions to complex data visualizations. Also, particularly useful are the tools for specific mathematical diagrams, which can automatically generate representations of state machines, linear programming problems with feasible regions, and various types of *Venn* diagrams. These functionalities address frequent needs in mathematics and computer science education. The package maintains compatibility with native **Wolfram Mathematica** capabilities while significantly simplifying the interface for $\LaTeX$ users.

Keywords: Automation, $\LaTeX$, TikZ , mathematical visualization, **Wolfram Mathematica**.

PRÓLOGO

El presente libro documenta el desarrollo del paquete **V_lT_EX**, una herramienta diseñada para automatizar la creación de documentos $\text{\LaTeX}$ en **Wolfram Mathematica**. A diferencia de otras soluciones parciales disponibles, **V_lT_EX** busca integrar de manera completa los procesos de cálculo matemático con la generación de documentación profesional. El paquete surgió de la necesidad recurrente en el ámbito académico de convertir resultados computacionales en documentos de alta calidad tipográfica.

La integración eficiente entre sistemas de cálculo simbólico y herramientas de documentación científica constituye un desafío persistente en el ámbito académico contemporáneo. La fragmentación típica del flujo de trabajo, donde los investigadores deben alternar constantemente entre entornos de cálculo y sistemas de composición tipográfica, genera ineficiencias significativas que se intensifican particularmente en disciplinas que requieren documentación matemática intensiva [1]. Esta problemática se agudiza cuando se consideran las demandas crecientes de reproducibilidad y automatización en la investigación científica moderna.

Las soluciones existentes para abordar esta integración presentan limitaciones considerables en términos de alcance y funcionalidad. Herramientas como *MaTeX* de Szabolcs Horvát [2], *matlab2tikz* [3], y el ya desaparecido *Mathematica2tikz* [4] son un ejemplo de ello. *MaTeX* se enfoca específicamente en crear etiquetas de figuras tipográficamente superiores usando $\text{\LaTeX}$ en **Mathematica**, compilando código $\text{\LaTeX}$ y reimportando los resultados como gráficos $\text{\LaTeX}$ en **Wolfram**, con la desventaja de tener una difícil configuración para su empleo. Por otra parte, *matlab2tikz* es un *script* de *MATLAB* que convierte figuras nativas de *MATLAB* a formato *TikZ/PGFPlots* para integración en documentos $\text{\LaTeX}$ con características relevantes pero hecho específicamente para el entorno de trabajo de *MATLAB*. Además, *mathematica2tikz* fue un paquete basado en *matlab2tikz* para convertir figuras de **Mathematica** a código $\text{\LaTeX}$, aunque limitado únicamente a gráficos 2D, sin embargo, actualmente ya no se encuentra disponible, pues no existen enlaces funcionales a los archivos en el *Wolfram Library Archive*. En resumen, *MaTeX* y *matlab2tikz* abordan problemas específicos de calidad tipográfica en sus respectivos ecosistemas, y *Mathematica2tikz* intentó ser un puente directo pero limitado entre **Mathematica** y $\text{\LaTeX}$ antes de su discontinuación.

Otras herramientas como *Mathex* [5] han intentado *embeber* expresiones de **Mathematica** en documentos $\text{\LaTeX}$ mediante preprocesadores, pero su enfoque se limita al renderizado básico sin abordar la exportación gráfica especializada. Herramientas más recientes como *nrpylatex* [6] proporcionan interfaces $\text{\LaTeX}$ para diversos sistemas de álgebra computacional, aunque su aplicación se restringe principalmente a campos específicos como la relatividad general y la geometría diferencial.

El desarrollo de herramientas automatizadas para la generación de documentos $\text{\LaTeX}$ ha experimentado avances significativos en años recientes. Investigaciones como las de Rahman, Bauer, and Khandelwal [7] han demostrado la efectividad de enfoques automatizados en contextos específicos como los reportes de análisis poblacional, donde la implementación de plantillas $\text{\LaTeX}$ modulares junto con *scripts* de automatización resultó en reducciones sustanciales del tiempo de documentación y mejoras en la consistencia tipográfica. Sin embargo, estas soluciones permanecen altamente especializadas y no abordan las necesidades generales de la

comunidad académica.

La proliferación de herramientas basadas en inteligencia artificial para la generación de documentos, incluyendo sistemas como *ThesisAI* y *Paperpal for Overleaf* [8], refleja el reconocimiento creciente de la necesidad de automatización en los procesos de documentación científica. No obstante, estos sistemas se enfocan primariamente en la asistencia de redacción y corrección lingüística, sin proporcionar las capacidades especializadas de exportación gráfica y renderizado matemático que demandan las publicaciones científicas avanzadas.

En el ecosistema específico de **Wolfram Mathematica**, las herramientas de integración con $\text{\LaTeX}$ han sido históricamente limitadas y fragmentarias. Las funcionalidades nativas como **TeXForm** proporcionan conversiones básicas de expresiones matemáticas, pero carecen de la robustez necesaria para la generación automatizada de documentos completos [9]. Los enfoques desarrollados por la comunidad, documentados extensivamente en foros especializados [10], tienden a ser soluciones *ad hoc* que requieren expertise técnico considerable y mantenimiento continuo.

La exportación de gráficos computacionales a formatos vectoriales editables representa un desafío técnico particular. Mientras que sistemas como $\text{\TikZ}$ y *PGFPlots* han establecido estándares de facto para la visualización matemática en $\text{\LaTeX}$ [11], la conversión automatizada de objetos gráficos complejos desde entornos de cálculo permanece como un área subdesarrollada. Las aproximaciones existentes frecuentemente requieren intervención manual significativa o resultan en pérdida de fidelidad gráfica. Particularmente desafiante resulta la generación de contenido dinámico y animado para documentos académicos, donde la representación de procesos matemáticos evolutivos o demostraciones paramétricas requiere herramientas especializadas que trasciendan la visualización estática tradicional.

En este contexto, el paquete **V $\text{\LaTeX}$** emerge como una respuesta integral a estas limitaciones sistémicas. A diferencia de las soluciones fragmentarias disponibles, **V $\text{\LaTeX}$** proporciona un ecosistema completo que abarca desde el renderizado básico de expresiones matemáticas hasta la exportación especializada de diagramas complejos en formato $\text{\TikZ}$. Su arquitectura modular permite la automatización completa del flujo de trabajo desde el cálculo hasta la documentación final, eliminando los puntos de fricción tradicionales que caracterizan la integración entre **Mathematica** y $\text{\LaTeX}$. Una innovación destacada del sistema es su capacidad para generar automáticamente animaciones interactivas integradas en documentos PDF, permitiendo la visualización dinámica de fenómenos matemáticos complejos mediante controles de reproducción nativos en el documento final.

La contribución principal de **V $\text{\LaTeX}$** reside en su enfoque unificado que combina capacidades de renderizado avanzado, compilación automatizada, exportación gráfica especializada y generación de contenido dinámico dentro de un *framework* coherente. Esta integración no solo mejora la eficiencia del proceso de documentación, sino que también democratiza el acceso a herramientas de composición tipográfica profesional para usuarios que no poseen expertise técnico extenso en $\text{\LaTeX}$, estableciendo un nuevo paradigma para la automatización de documentación científica en entornos de cálculo simbólico.

V $\text{\LaTeX}$ representa una evolución significativa entre entornos de cálculo simbólico y $\text{\LaTeX}$, constituye un ecosistema completo de catorce módulos especializados que trasciende la funcionalidad de renderizado simple, integrando desde el renderizado básico de expresiones matemáticas hasta la exportación compleja de diagramas $\text{\TikZ}$, sistemas de compilación automática de documentos, generación de animaciones paramétricas interactivas, y herramientas especializadas para autómatas, grafos y diagramas matemáticos en la creación de documentos científicos profesionales.

ViT_EX está organizado en catorce módulos que cubren desde el renderizado básico hasta funcionalidades especializadas como la exportación de diagramas complejos. Las funciones de renderizado permiten convertir expresiones matemáticas en imágenes de alta resolución mediante diferentes configuraciones: matemática básica, física con notación vectorial, química con fórmulas estructurales, y diagramas conmutativos. Esta flexibilidad resulta especialmente útil cuando se requieren documentos con notación especializada.

Además, **ViT_EX** incluye herramientas para generar documentos completos de forma automática. Los usuarios pueden crear artículos científicos, reportes técnicos, libros y presentaciones *Beamer* con configuraciones predefinidas que manejan aspectos como bibliografía, diferentes entornos (definiciones, teoremas, ejemplos, entre otros) y estilos de código. El sistema exporta automáticamente los gráficos de **Mathematica** y organiza los archivos en estructuras de carpetas coherentes.

Un componente destacado es el sistema de compilación integrado. **ViT_EX** detecta automáticamente los editores $\LaTeX$ instalados en el sistema operativo y permite compilar documentos directamente desde **Mathematica**. Esta característica elimina la fragmentación típica del flujo de trabajo entre cálculo y documentación.

El módulo de *plotting* con estilo $\LaTeX$ representa un avance significativo en la visualización matemática. Incorpora fuentes tipográficas especializadas y permite generar gráficos con la apariencia visual característica de publicaciones académicas.

Las capacidades de exportación a TikZ constituyen la funcionalidad más innovadora del paquete. **ViT_EX** puede convertir gráficos 2D y 3D de **Mathematica** en código *TikZ/PGFPlots*, preservando la calidad vectorial y permitiendo edición posterior. Los exportadores especializados en **ViT_EX** manejan casos específicos: grafos, autómatas de estado finito, secciones cónicas, y diagramas de teoría de conjuntos.

Una extensión revolucionaria de estas capacidades es la función `CreateTikZAnimation`, que permite generar animaciones $\LaTeX$ interactivas utilizando el paquete *animate*. Esta herramienta convierte funciones puras de **Mathematica** en animaciones paramétricas que pueden compilarse en documentos PDF con controles de reproducción integrados. La función soporta parámetros continuos y discretos, personalización visual completa, y es compatible con todos los tipos de gráficos de **Mathematica**, desde funciones trigonométricas hasta visualizaciones de datos complejas.

También, particularmente útiles son las herramientas para diagramas matemáticos puntuales, que pueden generar automáticamente representaciones de máquinas de estado, problemas de programación lineal con regiones factibles, y diversos tipos de diagramas de *Venn*. Estas funcionalidades abordan necesidades frecuentes en la enseñanza de las matemáticas y ciencias de la computación. El paquete mantiene compatibilidad con las capacidades nativas de **Wolfram Mathematica** mientras simplifica significativamente la interfaz para usuarios de $\LaTeX$.

Este libro está dirigido a tres audiencias principales: **docentes** que buscan optimizar la creación de materiales didácticos con notación matemática compleja, **investigadores** que requieren automatizar la documentación de sus resultados computacionales, y **estudiantes avanzados** en matemáticas, física e ingeniería que desean dominar herramientas profesionales de composición tipográfica científica. Los primeros tres capítulos introducen las funcionalidades fundamentales del paquete: el sistema de renderizado $\LaTeX$ (Capítulo 1), el compilador integrado (Capítulo 2), y el sistema de gráficos con estilo $\LaTeX$ (Capítulo 3). Los capítulos centrales (Capítulos 4–11) constituyen el núcleo técnico del libro, explorando en profundidad las capacidades de exportación a TikZ : desde gráficos 2D básicos (Capítulo 4) hasta visualizaciones 3D especializadas (Capítulos 9 y 10), pasando por herramientas específicas para diagramas de llenado (Capítulo 5), teoría de grafos (Capítulo 6), máquinas de estado (Capítulo 7), autómatas finitos (Capítulo 8), secciones cónicas (Capítulo 11), y casos especiales para diagramas matemáticos puntuales (Capítulo 12). Los capítulos finales presentan innovaciones avanzadas: contornos 2D (Capítulo 13) y la revolucionaria función de animaciones interactivas (Capítulo 14), antes de

concluir con reflexiones sobre el futuro de la automatización en documentación científica. Esta organización permite tanto una lectura secuencial para principiantes como consultas específicas para usuarios experimentados, garantizando que cada perfil de lector pueda aprovechar **VilTeX** según sus necesidades particulares.

VilTeX junto con sus archivos de apoyo puede descargarse en la siguiente dirección *url*:

<https://www.escinf.una.ac.cr/discretas/Archivos/Packages/Paquete%20VilTeX.zip>

1

Renderizado $\LaTeX$

¿Qué aprenderás en este capítulo?

- ▶ Convertir expresiones matemáticas de **Mathematica** en imágenes $\LaTeX$ de alta resolución.
- ▶ Configurar diferentes estilos de renderizado: matemática, física, química y diagramas conmutativos.
- ▶ Utilizar las funciones `RenderLatex`, `RenderLatexPhysics`, y `RenderLatexChemistry`.
- ▶ Personalizar parámetros de tamaño, resolución y formato de salida.
- ▶ Integrar expresiones renderizadas en documentos científicos profesionales.

El módulo de renderizado constituye un núcleo fundamental de $\mathbf{\LaTeX}$ para la conversión de expresiones matemáticas en imágenes de alta calidad (tipo $\LaTeX$). Esta funcionalidad aborda una limitación recurrente en el flujo de trabajo académico: la necesidad de generar representaciones visuales profesionales de expresiones complejas que mantengan la consistencia tipográfica característica de los documentos $\LaTeX$.

El sistema implementa una arquitectura basada en preámbulos que adaptan el renderizado según el contexto matemático requerido. La función principal `RenderToLatex` opera mediante cinco configuraciones predefinidas: matemática básica con los paquetes fundamentales *amsmath*, *amsfonts* y *amssymb*; física que incorpora el paquete *physics* para operadores vectoriales y diferenciales junto con *bm* para notación en negrita; química que integra *mhchem* para fórmulas moleculares y *chemfig* para estructuras; diagramas conmutativos mediante *tikz-cd*; y una configuración avanzada que combina *mathtools*, *TikZ* y *PGFPlots* para aplicaciones especializadas.

La detección automática de instalaciones $\LaTeX$ mediante `DetectLaTeX[]` representa un aspecto técnico significativo del sistema (esta instrucción se ejecuta de manera automática al levantar el paquete $\mathbf{\LaTeX}$). La función explora las rutas estándar del sistema operativo correspondiente - */Library/TeX/texbin/* en *macOS*, directorios *MiKTeX* en *Windows*, y */usr/bin/* en sistemas *Unix* - estableciendo automáticamente la variable global `$LaTeXPath` lo que elimina la configuración manual que tradicionalmente requieren las herramientas de integración $\LaTeX$ -**Mathematica**.

El proceso de renderizado emplea un directorio temporal para la compilación, generando documentos *standalone* que optimizan la calidad visual mediante parámetros ajustables de magnificación. El sistema maneja automáticamente la conversión `TeXForm` de expresiones en **Mathematica**, la compilación *pdflatex*, y la importación del resultado como objeto *image*.

La exportación final produce archivos *PNG* con resolución de *150 DPI* e *ImageSize* de 400 píxeles, parámetros calibrados empíricamente para mantener legibilidad en diferentes contextos de uso.

Las funciones especializadas `RenderMathToLatex`, `RenderPhysicsToLatex`, `RenderChemToLatex` y `RenderDiagramToLatex` actúan como interfaces simplificadas que preconfiguran los preámbulos apropiados. Esta estrategia de diseño reduce la complejidad para usuarios no especializados mientras preserva la flexibilidad para configuraciones avanzadas. La función `RenderBatchToLatex` extiende estas capacidades al procesamiento en lote, optimizando el flujo de trabajo cuando se requiere renderizar múltiples expresiones con configuraciones idénticas.

La robustez del sistema se evidencia en su manejo de errores durante la compilación $\text{\LaTeX}$. El módulo implementa verificación de códigos de salida del proceso *pdf_latex*, limpieza automática de directorios temporales, y valores de retorno `$Failed` consistentes que permiten la integración fluida con el ecosistema de **Wolfram Mathematica**.

1.1 Comandos del sistema de renderizado $\text{\LaTeX}$

En esta sección se comparten las sentencias que caracterizan al módulo de renderizado $\text{\LaTeX}$ que se encuentra incorporado en el paquete `V $\text{\LaTeX}$` . El objetivo es dar una descripción de cada una de las funciones integradas.

Nota: En **Mathematica** al instalar el paquete `V $\text{\LaTeX}$` se cuenta con la ayuda `?SistemaLaTeX` para desplegar una explicación detallada sobre el sistema de renderizado y la generación completa de documentos $\text{\LaTeX}$.

Funciones de renderizado

- `RenderToLatex[expr, opts]`: Renderiza una expresión matemática de **Wolfram Mathematica** a una imagen utilizando el motor de compilación $\text{\LaTeX}$. Admite opciones para especificar el tipo de preámbulo (`Preamble`) y el factor de magnificación (`Magnification`). Retorna una imagen compilada a partir del código $\text{\TeX}$ generado.
- `RenderToLatexDocument[filename, content, opts]`: Genera un documento $\text{\LaTeX}$ completo a partir de una lista de contenidos. Soporta múltiples clases de documento (`DocumentClass`), personalización de preámbulos, inclusión de bibliografía y configuración de entornos especiales para definiciones, teoremas, ejemplos, código, entre otros. Crea automáticamente una estructura de directorios en *Downloads* para organizar el proyecto.
- `RenderToLatexBeamer[filename, slides, opts]`: Construye una presentación *Beamer* completa a partir de una lista estructurada de diapositivas. Permite especificar tema (`Theme`), esquema de colores (`ColorScheme`) y tipo de preámbulo. Genera automáticamente la diapositiva de título y organiza el contenido en *frames*.

Funciones para construcción de diapositivas *Beamer*

- `BeamerSlide[title, content]`: Crea una asociación que representa una diapositiva básica de *Beamer* con título y contenido. El contenido puede ser texto, expresiones matemáticas o listas de elementos.

- **BeamerSlideWithOptions**[title, content, options]: Extiende la funcionalidad de **BeamerSlide** permitiendo especificar opciones adicionales de *frame* (por ejemplo: *fragile* o *allowframebreaks*).
- **BeamerSectionSlide**[sectionTitle]: Genera una diapositiva de sección centrada con tipografía de gran tamaño, útil para dividir presentaciones en capítulos o secciones temáticas.
- **BeamerTwoColumnSlide**[title, leftContent, rightContent]: Construye una diapositiva con disposición de dos columnas, distribuyendo el contenido especificado en formato paralelo con un 48% del ancho de texto para cada columna.

Funciones especializadas de renderizado

- **RenderMathToLatex**[expr, opts]: Función especializada para renderizar expresiones matemáticas puras utilizando el preámbulo básico de matemáticas (*amsmath*, *amsfonts*, *amssymb*).
- **RenderPhysicsToLatex**[expr, opts]: Renderiza expresiones físicas con soporte del paquete *physics*, que proporciona comandos especializados para notación de mecánica cuántica, derivadas y vectores en negrita.
- **RenderChemToLatex**[expr, opts]: Procesa expresiones químicas utilizando los paquetes *mhchem* y *chemfig* para notación molecular y diagramas estructurales.
- **RenderDiagramToLatex**[expr, opts]: Compila diagramas conmutativos utilizando el paquete *tikz-cd*, apropiado para matemáticas avanzadas y teoría de categorías.
- **RenderBatchToLatex**[expressions, opts]: Aplica el proceso de renderizado L^AT_EX a una lista completa de expresiones, retornando una lista de imágenes compiladas. Útil para procesamiento masivo de ecuaciones.

Funciones de creación de documentos

- **CreateLatexArticle**[filename, content, opts]: Inicializa un documento L^AT_EX con clase *article*, apropiado para artículos académicos, reportes breves y documentos técnicos. Configura automáticamente el formato estándar con márgenes de una pulgada.
- **CreateLatexReport**[filename, content, opts]: Genera un documento con clase *report*, estructurado para reportes extensos con soporte de capítulos, ideal para tesis, informes técnicos y documentación prolongada.
- **CreateLatexBook**[filename, content, opts]: Produce un documento con clase *book*, diseñado para publicaciones de formato libro con soporte completo de partes, capítulos y apéndices.
- **CreateBeamerPresentation**[filename, slides, opts]: Genera una presentación *Beamer* completa sin procesamiento especial de gráficos, manteniendo un enfoque minimalista en la exportación de imágenes.

Funciones con soporte extendido de gráficos

- **CreateLatexArticleWithGraphics**[filename, content, opts]: Versión mejorada de **CreateLatexArticle** con detección automática de objetos gráficos de **Mathematica** (*Plot*, *Graphics*, etc.). Exporta automáticamente los gráficos a una carpeta *figures* y genera las referencias L^AT_EX correspondientes.
- **CreateLatexReportWithGraphics**[filename, content, opts]: Implementa funcionalidad análoga para documentos tipo *report*, gestionando la exportación organizada de visualizaciones en un directorio estructurado del proyecto.

- `CreateBeamerPresentationWithGraphics[filename, slides, opts]`: Versión especializada de presentaciones *Beamer* con procesamiento inteligente de gráficos, incluyendo ajuste automático de tamaño para diapositivas y manejo de errores en la exportación.

Funciones auxiliares

- `DetectLaTeX[]`: Detecta automáticamente la instalación de $\text{\LaTeX}$ en el sistema operativo, identificando la ruta del ejecutable *pdflatex* en ubicaciones estándar de *macOS*, *Windows* y *Linux*. Actualiza la variable global `$LaTeXPath` con la ruta detectada.
- `processContentForBeamer[item]`: Función interna que procesa elementos individuales de contenido para su inclusión en *frames* de *Beamer*. Maneja conversión de cadenas de texto, exportación de imágenes y transformación de expresiones de **Mathematica** a código $\text{\TeX}$ mediante **TeXForm**.

1.2 Ejemplos de renderizado $\text{\LaTeX}$

RenderToLatex

La función `RenderToLatex` puede manejar una amplia variedad de expresiones matemáticas.

Como ya se explicó, esta instrucción convierte una expresión matemática a una imagen cuyo formato corresponde a una compilación $\text{\LaTeX}$ desde **Wolfram Mathematica**.

A continuación, se presentan algunos ejemplos sobre las capacidades de esta sentencia.

Ejemplo 1

```
RenderToLatex["E = mc^2"]
RenderToLatex["F = ma"]
```

Salida, imágenes que muestran:

$$E = mc^2$$

$$F = ma$$

- $E = mc^2$ – Equivalencia masa-energía de *Einstein*.
- $F = ma$ – Segunda ley de *Newton*.

Ejemplo 2

```
RenderToLatex["a^2 + b^2 = c^2"]
RenderToLatex["\\frac{1}{2}"]
RenderToLatex["x^{2n+1}"]
RenderToLatex["\\frac{-b \\pm \\sqrt{b^2-4ac}}{2a}"]
```

Salidas, imágenes que muestran:

$$a^2 + b^2 = c^2$$

$$\frac{1}{2}$$

$$x^{2n+1}$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- **Teorema de Pitágoras** – Relación fundamental en geometría.
- **Fracciones simples** – Notación fraccionaria básica.
- **Exponentes algebraicos** – Variable con exponentes complejos.
- **Fórmula cuadrática** – Solución de ecuaciones de segundo grado.

Ejemplo 3

```
RenderToLatex["\\int_0^\\pi \\sin(x) dx = 2"]
RenderToLatex["\\int_{-\\infty}^{\\infty} e^{-x^2} dx = \\sqrt{\\pi}"]
RenderToLatex["\\frac{\\partial f}{\\partial x}"]
RenderToLatex["\\oint_C f(z) dz = 2\\pi i \\sum \\text{Res}(f, z_k)"]
```

Salidas, imágenes que muestran:

$$\int_0^{\pi} \sin(x) dx = 2$$

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

$$\frac{\partial f}{\partial x}$$

$$\oint_C f(z) dz = 2\pi i \sum \text{Res}(f, z_k)$$

- **Integral definida trigonométrica** – Integral de seno en intervalo finito.
- **Integral gaussiana** – Una de las integrales más importantes en matemáticas.
- **Derivadas parciales** – Notación de cálculo multivariable.
- **Teorema de residuos** – Integral de contorno en análisis complejo.

Ejemplo 4

```
RenderToLatex["\\sum_{n=1}^{\\infty} \\frac{1}{n^2} = \\frac{\\pi^2}{6}"]
RenderToLatex["\\sum_{k=0}^n \\binom{n}{k} = 2^n"]
RenderToLatex["\\lim_{x \\to 0} \\frac{\\sin x}{x} = 1"]
RenderToLatex["\\lim_{n \\to \\infty} \\left(1 + \\frac{1}{n}\\right)^n = e"]
```

Salidas, imágenes que muestran:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$$

- **Problema de Basilea** – Suma infinita que vale $\frac{\pi^2}{6}$.

- **Teorema binomial** – Suma de coeficientes binomiales.
- **Límite trigonométrico fundamental** – Límite básico en cálculo.
- **Definición del número e** – Límite que define la constante de *Euler*.

Ejemplo 5

```
RenderToLatex["\\begin{pmatrix} 1 & 2 \\\\ 3 & 4 \\end{pmatrix}"]
RenderToLatex["\\det(A) = ad - bc"]
RenderToLatex["\\mathbf{v} \\cdot \\mathbf{w} = |\\mathbf{v}| |\\mathbf{w}| \\cos \\theta"]
RenderToLatex["\\begin{pmatrix} a_{11} & a_{12} & \\cdots & a_{1n} \\\\ a_{21} & a_{22} & \\cdots & a_{2n} \\\\ \\vdots & \\vdots & \\ddots & \\vdots \\\\ a_{m1} & a_{m2} & \\cdots & a_{mn} \\end{pmatrix}"]
```

Salidas, imágenes que muestran:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$\det(A) = ad - bc$$

$$\mathbf{v} \cdot \mathbf{w} = |\mathbf{v}| |\mathbf{w}| \cos \theta$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

- **Matriz 2×2** – Representación matricial básica.
- **Determinante** – Fórmula para matrices 2×2 .
- **Producto escalar** – Definición geométrica del producto punto.
- **Matriz general $m \times n$** – Notación para matrices de dimensión arbitraria.

Ejemplo 6

```
RenderToLatex["z = re^{i\\theta} = r(\\cos\\theta + i\\sin\\theta)"]
```

Salida, imagen que muestra:

$$z = re^{i\theta} = r(\cos \theta + i \sin \theta)$$

- **Forma polar de números complejos** – Representación exponencial y trigonométrica.

Ejemplo 7

```
RenderToLatex["\\chi(S^2) = 2"]
RenderToLatex["\\pi_1(S^1) \\cong \\mathbb{Z}"]
```

Salidas, imágenes que muestran:

$$\chi(S^2) = 2$$

$$\pi_1(S^1) \cong \mathbb{Z}$$

- **Característica de Euler** – Invariante topológico de la esfera.
- **Grupo fundamental** – Isomorfismo del grupo fundamental del círculo.

Ejemplo 8

```
RenderToLatex["\\begin{cases}
x + y = 5 \\\\
2x - y = 1
\\end{cases}"]
```

```
RenderToLatex["f(x) = \\begin{cases}
x^2 & \\text{si } x \\geq 0 \\\\
-x^2 & \\text{si } x < 0
\\end{cases}"]
```

Salidas, imágenes que muestran:

$$\begin{cases} x + y = 5 \\ 2x - y = 1 \end{cases}$$

$$f(x) = \begin{cases} x^2 & \text{si } x \geq 0 \\ -x^2 & \text{si } x < 0 \end{cases}$$

- **Sistema lineal 2 × 2** – Conjunto de ecuaciones simultáneas.
- **Función definida por partes** – Función con diferentes reglas por dominio.

Ejemplo 9

```
RenderToLatex["\\begin{tikzcd}
A \\arrow[r, \"f\"] \\arrow[d, \"g\"] & B \\arrow[d, \"h\"] \\\\
C \\arrow[r, \"k\"] & D
\\end{tikzcd}", "Preamble" -> "diagrams"]
```

Salida, imagen que muestra:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ g \downarrow & & \downarrow h \\ C & \xrightarrow{k} & D \end{array}$$

- **Diagrama conmutativo** – Representación visual de morfismos en teoría de categorías.

Ejemplo 10

```
RenderToLatex["\\text{La ecuacion de Schrodinger: } i\\hbar
\\frac{\\partial \\psi}{\\partial t} = \\hat{H}\\psi
\\text{ describe la evolucion temporal de un sistema cuantico.}"]
```

Salida, imagen que muestra:

La ecuación de Schrödinger: $i\hbar \frac{\partial \psi}{\partial t} = \hat{H}\psi$ describe la evolución temporal de un sistema cuántico.

- **Ecuación de *Schrödinger* dependiente del tiempo** – Ecuación fundamental de la mecánica cuántica con texto explicativo.

8

Ejemplo 11

```
result = Integrate[x^2, {x, 0, 1}];
RenderToLatex["\\int_0^1 x^2 dx = " <> ToString[TeXForm[result]]]
```

Salida, imagen que muestra:

$$\int_0^1 x^2 dx = \frac{1}{3}$$

- **Cálculo e inserción automática** – Demostración de integración de resultados computacionales de **Mathematica** en $\text{\LaTeX}$.

Ejemplo 12

```
RenderToLatex["\\sum_{n=1}^{\\infty} \\frac{1}{n^2}",
"Magnification" → 1]
RenderToLatex["\\int_0^{\\pi} \\sin(x) dx", "Magnification" → 1.5]
RenderToLatex["\\lim_{x \\to \\infty} \\frac{1}{x}",
"Magnification" → 3.0]
```

Salida, imagen que muestra:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

$$\int_0^{\pi} \sin(x) dx$$

$$\lim_{x \rightarrow \infty} \frac{1}{x}$$

- **Aumento de tamaño por el uso de la opción `Magnification`** – Serie, integral definida y límite.

RenderPhysicsToLatex

La función **RenderPhysicsToLatex** está específicamente diseñada para el renderizado de expresiones que requieren la notación especializada del paquete *physics* de $\text{\LaTeX}$.

A diferencia del renderizado matemático básico, esta función incorpora comandos avanzados para operadores vectoriales (gradiente, divergencia, rotacional), derivadas parciales con notación de

Dirac, y vectores en negrita mediante el paquete *bm*.

Esta especialización resulta particularmente valiosa en documentos de mecánica clásica, electromagnetismo, mecánica cuántica y relatividad, donde la precisión notacional es fundamental para la claridad conceptual. Veamos algunos ejemplos.

Ejemplo 13

```
RenderPhysicsToLatex["H = \\frac{p^2}{2m} + V(\\vec{r})"]
RenderPhysicsToLatex["G_{\\mu\\nu} = \\frac{8\\pi G}{c^4} T_{\\mu\\nu}"]
RenderPhysicsToLatex["\\vec{\\nabla} \\times \\vec{E} = -\\frac{\\partial}{\\partial t} \\vec{B}"]
```

Salida, imagen que muestra:

$$H = \frac{p^2}{2m} + V(\vec{r})$$

$$G_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$$

- **Fórmula de mecánica clásica** – Energía total (*Hamiltoniano*) de una partícula en un campo potencial.
- **Curvatura del espacio** – Ecuación de campo de *Einstein*, base de la relatividad general.
- **Ecuaciones de Maxwell** – Indica que un campo magnético variable en el tiempo genera un campo eléctrico rotacional.

RenderChemToLatex

El renderizado de expresiones químicas presenta desafíos distintos a los de la notación matemática estándar, requiriendo el manejo de fórmulas moleculares, estados de oxidación, equilibrios químicos y estructuras tridimensionales.

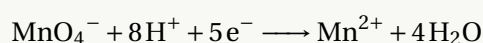
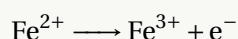
La función **RenderChemToLatex** integra los paquetes *mhchem* y *chemfig* de L^AT_EX para proporcionar una sintaxis simplificada que maneja automáticamente aspectos como la disposición de subíndices y superíndices en fórmulas, flechas de reacción con condiciones, y la representación de estructuras moleculares complejas.

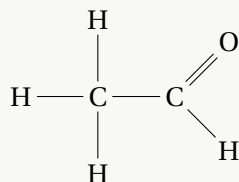
Los siguientes ejemplos ilustran la versatilidad de esta función en distintos contextos.

Ejemplo 14

```
RenderChemToLatex["\\ce{Fe^{2+} -> Fe^{3+} + e-}"]
RenderChemToLatex["\\ce{MnO4- + 8H+ + 5e- -> Mn^{2+} + 4H2O}"]
RenderChemToLatex["\\chemfig{H-C(-[2]H)(-[6]H)-C(=[1]O)-[7]H}"]
```

Salida, imagen que muestra:





- **Reacción** – Redox del hierro.
- **Reacción** – Del *permanganato* en medio ácido.
- **Molécula** – Representa una molécula orgánica.

RenderDiagramToLatex

Los diagramas conmutativos constituyen una herramienta esencial en álgebra abstracta, teoría de categorías y topología algebraica, donde las relaciones entre objetos y morfismos requieren representación visual precisa.

La función `RenderDiagramToLatex` utiliza el paquete *tikz-cd* para generar estos diagramas con calidad profesional, permitiendo especificar morfismos, relaciones de conmutatividad, y estructuras categóricas complejas.

La función mantiene la semántica matemática mientras produce imágenes que preservan las convenciones tipográficas del área. A continuación se presentan algunos casos representativos.

Ejemplo 15

```

RenderDiagramToLatex["\\begin{tikzcd}
& A \\arrow{r}{f} \\arrow{d}{\\alpha} & B \\arrow{r}{g} \\arrow{d}{\\beta} & C \\arrow{r}{h} \\arrow{d}{\\gamma} & 0 \\\\
0 \\arrow{r} & A' \\arrow{r}{f'} & B' \\arrow{r}{g'} & C' & \\
\\end{tikzcd}"]

RenderDiagramToLatex["\\begin{tikzcd}
G \\arrow{r}{\\phi} \\arrow{d}{p_i} & H \\arrow{d}{\\psi} & \\\\
G/N \\arrow[dashed]{r}{\\bar{\\phi}} & H/M & \\
\\end{tikzcd}"]

RenderDiagramToLatex["\\begin{tikzcd}
P \\arrow{r}{p_2} \\arrow{d}{p_1} \\arrow[phantom]{dr}{\\lrcorner} & \\\\
Y \\arrow{d}{g} & \\\\
X \\arrow{r}{f} & Z \\
\\end{tikzcd}"]

```

Salida, imagen que muestra:

$$\begin{array}{ccccccc}
 A & \xrightarrow{f} & B & \xrightarrow{g} & C & \longrightarrow & 0 \\
 \downarrow \alpha & & \downarrow \beta & & \downarrow \gamma & & \\
 0 & \longrightarrow & A' & \xrightarrow{f'} & B' & \xrightarrow{g'} & C'
 \end{array}$$

$$\begin{array}{ccc}
 G & \xrightarrow{\phi} & H \\
 \downarrow p_i & & \downarrow \psi \\
 G/N & \xrightarrow{\bar{\phi}} & H/M
 \end{array}$$

$$\begin{array}{ccc}
 P & \xrightarrow{p_2} & Y \\
 \downarrow p_1 & \lrcorner & \downarrow g \\
 X & \xrightarrow{f} & Z
 \end{array}$$

- **Diagramas conmutativos** – Diagrama con filas exactas, diagrama de funciones entre grupos cociente y diagrama de producto fibrado (*pullback*).

RenderBatchToLatex

Cuando un documento requiere el renderizado de múltiples expresiones con configuraciones idénticas, el procesamiento individual resulta ineficiente y propenso a inconsistencias.

La función `RenderBatchToLatex` aborda esta necesidad mediante procesamiento en lote, aplicando las mismas opciones de preámbulo y magnificación a una lista completa de expresiones en una sola invocación.

Esta aproximación no solo optimiza el tiempo de compilación, sino que garantiza homogeneidad visual en secciones que presentan series de ecuaciones, fórmulas relacionadas o múltiples resultados computacionales. Los ejemplos siguientes demuestran tanto su aplicación con *strings* L^AT_EX directos como con expresiones simbólicas de **Wolfram Mathematica**.

Ejemplo 16

```

equations = {"E = mc^2", "F = ma",
  "\\Delta x \\Delta p \\geq \\frac{\\hbar}{2}",
  "\\sum_{n=1}^{\\infty} \\frac{1}{n^2} = \\frac{\\pi^2}{6}"};
RenderBatchToLatex[equations]

expressions = {Defer[D[x^3, x]], Integrate[Sin[x], x],
  Defer@Limit[Sin[x]/x, x -> 0], Series[E^x, {x, 0, 3}]};
RenderBatchToLatex[expressions, "Magnification" -> 3]

```

Salida, dos vectores con imágenes que muestran:

$$\{E = mc^2, F = ma, \Delta x \Delta p \geq \frac{\hbar}{2}, \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}\}$$

$$\left\{ \frac{\partial x^3}{\partial x}, -\cos(x), \lim_{x \rightarrow 0} \frac{\sin(x)}{x}, \right.$$

$$\left. 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + O(x^4) \right\}$$

- **Renderizado de varias expresiones** – Con formato L^AT_EX y de **Mathematica**.

Es importante señalar que la sentencia `RenderToLatex` posee la opción `Preamble -> "basic" | "physics" | "chemistry" | "diagrams" | "advanced"`, por lo que:

- `RenderToLatex[expr, Preamble -> "physics"]`: es equivalente al uso de la instrucción `RenderPhysicsToLatex[expr]`.
- `RenderToLatex[expr, Preamble -> "chemistry"]`: genera el mismo resultado de `RenderChemToLatex[expr]`.
- `RenderToLatex[expr, Preamble -> "diagrams"]`: es análogo a utilizar la función `RenderDiagramToLatex[expr]`.

`RenderToLatexDocument`

Mientras que las funciones de renderizado anteriores se enfocan en la conversión de expresiones individuales a imágenes, `RenderToLatexDocument` representa un cambio de paradigma fundamental: la generación automatizada de documentos $\text{\LaTeX}$ completos y compilables desde **Wolfram Mathematica**.

Esta función orquesta todos los componentes necesarios para crear artículos científicos, reportes técnicos o libros, gestionando aspectos como la selección de clase de documento (*article*, *report*, *book*), configuración de preámbulos con distintos niveles de complejidad (*basic*, *extended*, *full*), uso de entornos de definiciones, teoremas, ejemplos, entre otros, manejo de bibliografía, y la organización automática de archivos en estructuras de directorios coherentes.

La función acepta listas heterogéneas de contenido -combinando texto, expresiones matemáticas de **Wolfram**, imágenes y gráficos- procesándolas de manera inteligente para generar código $\text{\LaTeX}$ semánticamente correcto. Los ejemplos a continuación ilustran desde casos básicos hasta configuraciones avanzadas que demuestran el control granular sobre la estructura y apariencia del documento final.

Para ilustrar la versatilidad de `RenderToLatexDocument`, se presentan cuatro configuraciones representativas que abarcan el espectro completo de necesidades de documentación académica:

- La configuración minimalista: demuestra la creación de documentos simples con estructura básica de secciones, ideal para reportes breves o documentación técnica elemental.
- La configuración para artículos científicos: incorpora elementos especializados como resúmenes (*abstract*), entornos de definiciones y teoremas, bloques de advertencia (*important*), y la integración automática de gráficos generados por **Mathematica**, estableciendo un flujo de trabajo completo para publicaciones académicas.
- La configuración para libros: ejemplifica la gestión de estructuras complejas con *frontmatter*, *mainmatter* y *backmatter*, múltiples partes y capítulos, apéndices, y listados de código mediante el entorno *lstlisting*, demostrando cómo `RenderToLatexDocument` maneja documentos de gran escala manteniendo coherencia organizacional.
- La configuración para tesis: presenta el caso de uso más completo, integrando páginas de título personalizadas, resúmenes bilingües, múltiples índices, teoremas con demostraciones formales, tablas comparativas, y la combinación de análisis teórico con implementación computacional.

Cada ejemplo presentado a continuación progresa en complejidad y demuestra diferentes combinaciones de opciones que controlan entornos de teoremas (*TheoremStyle*), bloques especiales coloreados (*SpecialEnvStyle*), estilos de código (*CodeStyle*), comandos matemáticos adicionales (*MathStyle*), y gestión bibliográfica

(*CreateBibliography* y *BibliographyStyle*). Los ejemplos ilustran cómo una misma función puede adaptarse desde un documento de dos páginas hasta una tesis de estructura completa, manteniendo en todos los casos la calidad tipográfica característica de L^AT_EX y la eficiencia del flujo de trabajo automatizado.

Nota: En los ejemplos siguientes se han omitido las tildes de los textos para facilitar la presentación en código del presente documento, sin embargo, se aclara al lector que en **Wolfram Mathematica** dicho uso no genera ningún conflicto en el archivo *.tex* construido de manera automática por la sentencia `RenderToLatexDocument`.

Ejemplo 17 Configuración minimalista

```
contenido = {"\\section{Introduccion}",
  "Este es un documento muy b\\'asico sin elementos especiales.", "",
  "\\section{Desarrollo}",
  "Aqu\\'i se presenta el contenido principal del documento.", "",
  "\\subsection{Punto Importante}",
  "Este subsection contiene informaci\\'on relevante.", "",
  "\\section{Conclusion}",
  "En conclusion, este documento demuestra la configuraci\\'on minimalista."};

RenderToLatexDocument["documento", contenido,
  "Title" → "Documento Simple",
  "Author" → "Usuario"]
```

Lo que crea automáticamente la carpeta y el archivo: *documento/documento.tex*, guardado en *Downloads*.

Ejemplo 18 Configuración recomendada para artículo

```
contenidoArticulo = {
  (* Abstract *)
  "\\begin{abstract}",
  "Este articulo presenta un nuevo enfoque para...",
  "\\end{abstract}", "",

  (* Keywords *)
  "\\textbf{Palabras clave:} algoritmos, optimizacion, analisis numerico", "",

  (* Introduccion *)
  "\\section{Introduccion}",
  "El problema de optimizacion no lineal ha sido...", "",
  "\\subsection{Motivacion}",
  "La necesidad de algoritmos eficientes surge de...", "",

  (* Metodologia *)
  "\\section{Metodologia}",
  "Proponemos el siguiente algoritmo:", "",
  "\\begin{definition}",
  "Sea  $f: \\mathbb{R}^n \\rightarrow \\mathbb{R}$  una funcion...",
  "\\end{definition}", "",
  "\\begin{theorem}",
  "El algoritmo propuesto converge en  $O(n \\log n)$  iteraciones.",
  "\\end{theorem}", "",
}
```

```

(* Resultados *)
"\section{Resultados Experimentales}",
"Los experimentos se realizaron en...", "",
"\begin{important}",
"Los resultados muestran una mejora significativa del 30\\% en eficiencia.",
"\end{important}", "",

(* Grafico de resultados (si EnableGraphics esta activo) *)
Plot[x^2*Sin[x], {x, 0, 10}, PlotLabel → "Convergencia del Algoritmo"], "",

(* Discusion *)
"\section{Discusion}",
"Los resultados obtenidos demuestran que...", "",

(* Conclusiones *)
"\section{Conclusiones}",
"En este trabajo hemos presentado...", "",
"\subsection{Trabajo Futuro}",
"Las lineas de investigacion futuras incluyen...", "",

(* Agradecimientos *)
"\section*{Agradecimientos}",
"Los autores agradecen el apoyo financiero de..."
};

RenderToLatexDocument["articulo", contenidoArticulo,
"Preamble" → "extended",
"Title" → "Análisis Numerico de Sistemas Dinamicos No Lineales",
"Author" → "Dr. Luis Martinez",
"TheoremStyle" → "extended",
"SpecialEnvStyle" → "extended",
"CodeStyle" → "extended",
"MathStyle" → "extended",
"CreateBibliography" → True]

```

Lo que genera la carpeta y el archivo: *articulo/articulo.tex*, guardado en *Downloads*.

Un aspecto interesante en este ejemplo es que al correr la función `RenderToLatexDocument` automáticamente convierte la gráfica desplegada por el `Plot` en un archivo *.png* que se guarda en una carpeta llamada *figures* ubicada en el mismo lugar del archivo *articulo.tex*. De esta manera, `RenderToLatexDocument` siempre crea *files .png* para gráficas construidas mediante código de **Wolfram Mathematica**. También es importante señalar que al emplear la opción "`CreateBibliography`" → `True` se automatiza la creación de un archivo *articulo.bib* con una referencia bibliográfica de muestra para su posterior edición.

Ejemplo 19 Configuración recomendada para libro

```

contenidoLibro = {
  (* == FRONTMATTER == *)
  "\frontmatter", "",

  (* Paginas preliminares *)
  "\tableofcontents",
  "\listoffigures",
  "\listoftables", "",

  (* Prefacio *)
  "\chapter*{Prefacio}",

```

```

"\addcontentsline{toc}{chapter}{Prefacio}",
"Este libro surge de la necesidad de unificar los enfoques modernos en
análisis numerico...", "",

(* Prologo *)
"\chapter*{Prologo}",
"\addcontentsline{toc}{chapter}{Prologo}",
"La matematica aplicada ha experimentado un crecimiento exponencial...",
"",

(* == MAINMATTER == *)
"\mainmatter", "",

(* PARTE I: FUNDAMENTOS *)
"\part{Fundamentos Teoricos}", "",
"\chapter{Introduccion a los Sistemas Dinamicos}",
"\section{Definiciones Basicas}",
"Un sistema dinamico es una descripcion matematica...", "",
"\section{Clasificacion de Sistemas}",
"Los sistemas dinamicos pueden clasificarse segun...", "",
"\subsection{Sistemas Lineales}",
"Los sistemas lineales forman la base...", "",
"\subsection{Sistemas No Lineales}",
"Los sistemas no lineales presentan comportamientos...", "",

"\chapter{Metodos Numericos Fundamentales}",
"\section{Metodos de Integracion}",
"\begin{definition}",
"Un metodo de integracion numerica es...",
"\end{definition}", "",
"\section{Análisis de Estabilidad}",
"\begin{theorem}",
"Un metodo numerico es estable si...",
"\end{theorem}", "",

(* PARTE II: METODOS AVANZADOS *)
"\part{Metodos Avanzados}", "",
"\chapter{Algoritmos de Optimizacion}",
"\section{Metodos de Gradiente}",
"Los metodos basados en gradiente constituyen...", "",
"\begin{important}",
"La eleccion del paso es crucial para la convergencia.",
"\end{important}", "",

(* ALGORITMO CORREGIDO - Version simple sin paquetes problematicos *)
"\textbf{Algoritmo: Metodo del Gradiente}", "",
"\begin{enumerate}",
"\item Inicializar  $x_0$ ",
"\item Mientras no converja:",
"\begin{enumerate}",
"\item  $x_{k+1} = x_k - \alpha \nabla f(x_k)$ ",
"\item Verificar criterio de convergencia",
"\end{enumerate}",
"\item Fin del algoritmo",
"\end{enumerate}", "",

"\chapter{Análisis de Convergencia}",
"\section{Teoremas de Convergencia}",
"\begin{theorem}[Convergencia Global]",
"Bajo las condiciones apropiadas, el algoritmo converge...",
"\end{theorem}", "",

```

```

(* PARTE III: APLICACIONES *)
"\part{Aplicaciones y Casos de Estudio}", "",
"\chapter{Aplicaciones en Ingenieria}",
"\section{Control de Sistemas}",
"En el control automatico, los sistemas dinamicos...", "",
"\chapter{Casos de Estudio}",
"\section{Pendulo Doble}",
"El pendulo doble es un ejemplo clasico...", "",

(* Grafico del pendulo *)
Plot[Sin[t]*Cos[2*t], {t, 0, 4*Pi},
PlotLabel ->"Dinamica del Pendulo Doble"], "",

(* == BACKMATTER == *)
"\backmatter", "",

(* Apendices *)
"\appendix",
"\chapter{Demostraciones Tecnicas}",
"\section{Demostracion del Teorema Principal}",
"La demostracion procede por induccion...", "",

"\chapter{Codigo Fuente}",
"\section{Implementacion en MATLAB}",
"\begin{lstlisting}[language=MATLAB]",
"function result = dynamical_system(x0, t)",
"% Implementacion del sistema dinamico",
"% Input: x0 - condiciones iniciales",
"% t - vector de tiempo",
"end",
"\end{lstlisting}", "",

(* Indices *)
"\chapter*{Indice Alfabetico}",
"\addcontentsline{toc}{chapter}{Indice Alfabetico}", "",

(* Agradecimientos *)
"\chapter*{Agradecimientos}",
"\addcontentsline{toc}{chapter}{Agradecimientos}",
"Este libro no habria sido posible sin..."
};

RenderToLatexDocument["libro", contenidoLibro,
"DocumentClass" ->"book",
"Preamble" ->"extended",
"Title" ->"Analisis Numerico de Sistemas Dinamicos No Lineales",
"Author" ->"Dr. Luis Martinez",
>Date" ->"2025",
"TheoremStyle" ->"full",
"SpecialEnvStyle" ->"extended",
"CodeStyle" ->"extended",
"MathStyle" ->"full",
>CreateBibliography" ->True,
"BibliographyStyle" ->"alpha"]

```

Lo que construye la carpeta y el archivo: *libro/libro.tex*, guardado en *Downloads*.

Aquí también la gráfica `Plot` se transforma a un archivo *.png* almacenado en la carpeta *figures* y la opción `"CreateBibliography" -> True` crea a *libro.bib* con una referencia bibliográfica de ejemplo para su posterior edición.

Ejemplo 20 Configuración recomendada para tesis

```

contenidoTesis = {
  (* == FRONTMATTER == *)
  "\\frontmatter", "",

  (* Pagina de titulo simple *)
  "\\begin{titlepage}",
  "\\centering",
  "\\vspace*{3cm}",
  "{\\Large UNIVERSIDAD NACIONAL}\\\\[1cm]",
  "{\\large Facultad de Ciencias}\\\\[2cm]",
  "{\\LARGE\\textbf{METODOS NUMERICOS PARA SISTEMAS DINAMICOS}}\\\\[0.5cm]",
  "{\\Large\\textbf{NUEVOS ENFOQUES COMPUTACIONALES}}\\\\[3cm]",
  "{\\large Tesis Doctoral}\\\\[1cm]",
  "{\\large\\textbf{Ana Maria Lopez}}\\\\[2cm]",
  "{\\large Director: Dr. Carlos Ruiz}\\\\[2cm]",
  "\\vfill",
  "{\\large 2025}",
  "\\end{titlepage}", "",

  (* Resumen *)
  "\\chapter*{Resumen}",
  "\\addcontentsline{toc}{chapter}{Resumen}",
  "Esta tesis desarrolla nuevos metodos numericos para sistemas dinamicos complejos, enfocandose en algoritmos adaptativos que mejoran la eficiencia computacional y la precision de las soluciones.", "",
  "\\textbf{Palabras clave:} analisis numerico, sistemas dinamicos, algoritmos adaptativos", "",

  (* Abstract *)
  "\\chapter*{Abstract}",
  "\\addcontentsline{toc}{chapter}{Abstract}",
  "This thesis develops new numerical methods for complex dynamical systems, focusing on adaptive algorithms that improve computational efficiency and solution accuracy.", "",
  "\\textbf{Keywords:} numerical analysis, dynamical systems, adaptive algorithms", "",

  (* Indices *)
  "\\tableofcontents",
  "\\listoffigures", "",

  (* == MAINMATTER == *)
  "\\mainmatter", "",

  (* CAPITULO 1: INTRODUCCION *)
  "\\chapter{Introduccion}", "",
  "\\section{Motivacion}",
  "Los sistemas dinamicos no lineales presentan desafios computacionales significativos. Esta tesis propone nuevos metodos adaptativos para su analisis eficiente.", "",
  "\\section{Objetivos}",
  "\\begin{enumerate}",
  "\\item Desarrollar algoritmos adaptativos mejorados",
  "\\item Analizar la estabilidad y convergencia",
  "\\item Validar mediante casos de estudio",
  "\\end{enumerate}", "",
  "\\section{Contribuciones}",
  "\\begin{important}",
  "Las principales contribuciones incluyen un nuevo esquema adaptativo,

```

```

analisis teorico de convergencia, e implementacion eficiente.",
"\end{important}", "",

(* CAPITULO 2: ESTADO DEL ARTE *)
"\chapter{Fundamentos Teoricos}", "",
"\section{Sistemas Dinamicos}",
"\begin{definition}",
"Un sistema dinamico es una tupla  $(X, f)$  donde  $X$  es el espacio de
estados y  $f$  la funcion de evolucion temporal.",
"\end{definition}", "",
"\section{Metodos Existentes}",
"Los metodos tradicionales incluyen Runge-Kutta y esquemas multipaso,
pero presentan limitaciones en sistemas rigidos de alta dimensionalidad.",
"",
"\begin{warning}",
"Los metodos clasicos fallan en preservar propiedades geometricas y
requieren pasos muy pequenos.",
"\end{warning}", "",

(* CAPITULO 3: METODOLOGIA *)
"\chapter{Metodologia Propuesta}", "",
"\section{Algoritmo Principal}",
"\textbf{Algoritmo Adaptativo:}", "",
"\begin{enumerate}",
"\item Inicializar condiciones:  $x_0, t_0, h_0, \varepsilon$ ",
"\item Para cada paso:",
"\begin{enumerate}",
"\item Calcular aproximaciones de orden alto y bajo",
"\item Estimar error:  $e = |x_h - x_{h/2}|$ ",
"\item Si  $e \leq \varepsilon$ : aceptar, sino: reducir paso",
"\end{enumerate}",
"\item Actualizar tamano de paso adaptativamente",
"\end{enumerate}", "",
"\section{Analisis Teorico}",
"\begin{theorem}",
"El metodo propuesto es A-estable y converge con orden  $p$  bajo
condiciones de regularidad.",
"\end{theorem}", "",
"\begin{proof}",
"La demostracion sigue del analisis de la funcion de estabilidad...",
"\end{proof}", "",

(* CAPITULO 4: IMPLEMENTACION *)
"\chapter{Implementacion}", "",
"\section{Arquitectura}",
"La implementacion utiliza MATLAB con optimizaciones en C++ para
operaciones criticas.", "",
"\begin{lstlisting}[language=MATLAB]",
"function [t, x] = integrador_adaptativo(f, x0, tspan, opts)",
"% Integrador adaptativo principal",
"% Inicializacion",
"t0 = tspan(1); tf = tspan(2);",
"h = opts.paso_inicial;",
"tol = opts.tolerancia;", "",
"% Bucle principal",
"t = t0; x = x0;",
"while t < tf",
"    [x_new, error] = paso_adaptativo(f, t, x, h);",
"    if error <= tol",
"        t = t + h; x = x_new;",
"        h = min(1.2*h, tf-t);",
"    else",

```

```

" h = 0.5*h;",
" end",
"end",
"end",
"\\end{lstlisting}", "",
"\\section{Optimizaciones}",
"\\begin{success}",
"Se implementaron tecnicas de paralelizacion, gestion eficiente de
memoria y uso de bibliotecas optimizadas (BLAS/LAPACK).",
"\\end{success}", "",

(* CAPITULO 5: RESULTADOS *)
"\\chapter{Resultados}", "",
"\\section{Casos de Prueba}", "",
"\\subsection{Oscilador de Van der Pol}",
"\\ddot{x} - \\mu(1-x^2)\\dot{x} + x = 0", "",

(* Grafico simple *)
Plot[Sin[t]*Exp[-0.1*t], {t, 0, 15},
PlotLabel ->"Solucion del Oscilador"], "",

"\\subsection{Sistema de Lorenz}",
"\\begin{cases}",
"\\dot{x} = \\sigma(y - x) \\\\",
"\\dot{y} = x(\\rho - z) - y \\\\",
"\\dot{z} = xy - \\beta z",
"\\end{cases}", "",

"\\section{Análisis de Rendimiento}",
"\\begin{table}[h]",
"\\centering",
"\\begin{tabular}{|l|c|c|}",
"\\hline",
"\\textbf{Metodo} & \\textbf{Tiempo} & \\textbf{Error} \\\\",
"\\hline",
"RK45 & 12.3s & $10^{-6}$ \\\\",
"Propuesto & 8.7s & $10^{-7}$ \\\\",
"\\hline",
"\\end{tabular}",
"\\caption{Comparacion de rendimiento}",
"\\end{table}", "",
"\\begin{important}",
"Los resultados muestran mejora del 29\\% en tiempo y orden de magnitud
en precision.",
"\\end{important}", "",

(* CAPITULO 6: CONCLUSIONES *)
"\\chapter{Conclusiones}", "",
"\\section{Logros Principales}",
"\\begin{enumerate}",
"\\item Desarrollo exitoso de algoritmo adaptativo mejorado",
"\\item Demostracion rigurosa de estabilidad y convergencia",
"\\item Validacion experimental en casos representativos",
"\\item Implementacion eficiente con mejoras significativas",
"\\end{enumerate}", "",
"\\section{Trabajo Futuro}",
"Las lineas de investigacion futuras incluyen extension a sistemas
estocasticos, metodos preservadores de estructura, e implementacion en
GPU.", "",
"\\section{Impacto}",
"\\begin{summary}",
"Los metodos desarrollados tienen aplicaciones en simulacion climatica,
```

```

modelado financiero y control de sistemas.",
"\\end{summary}", "",

(* == BACKMATTER == *)
"\\backmatter", "",

(* Apendices basicos *)
"\\appendix", "",
"\\chapter{Demostraciones}",
"\\section{Prueba de Convergencia}",
"\\begin{proof}",
"La convergencia se establece mediante analisis de consistencia y
estabilidad...",
"\\end{proof}", "",

"\\chapter{Codigo Principal}",
"\\section{Funciones Clave}",
"\\begin{lstlisting}[language=MATLAB]",
"% Funcion de paso adaptativo",
"function [x_new, error] = paso_adaptativo(f, t, x, h)",
"% Implementacion del esquema propuesto",
"x1 = rk45_paso(f, t, x, h);",
"x2 = rk45_paso(f, t, x, h/2);",
"x2 = rk45_paso(f, t+h/2, x2, h/2);",
"error = norm(x1 - x2);",
"x_new = x1;",
"end",
"\\end{lstlisting}", "",

(* Bibliografia *)
"\\chapter*{Referencias}",
"\\addcontentsline{toc}{chapter}{Referencias}"
};

RenderToLatexDocument["tesis.tex", contenidoTesis,
"DocumentClass" → "book",
"Preamble" → "full",
"Title" →
"Analisis Numerico de Sistemas Dinamicos Complejos: Nuevos Enfoques
Computacionales",
"Author" → "Maria Elena Rodriguez Garcia",
"Date" → "2025",
"CreateBibliography" → True,
"BibliographyStyle" → "alpha"]

```

Lo que crea la carpeta y el archivo: *tesis/tesis.tex* en *Downloads*, un *.png* guardado en *figures* correspondiente a la gráfica *Plot* y el *file tesis.bib* con una referencia de muestra para su posterior edición.

Estos ejemplos se encuentran integrados para su empleo directo en el archivo *1. Ejemplos RENDE- RIZADO LaTeX.nb*, disponible en el enlace de descarga del paquete **V_LT_EX** (ver página xiv).

RenderToLatexBeamer

La comunicación efectiva de resultados científicos frecuentemente requiere presentaciones visuales que mantengan la calidad tipográfica de las publicaciones escritas.

La función `RenderToLatexBeamer` extiende las capacidades del sistema hacia la generación automatizada de presentaciones mediante la clase `beamer` de $\text{\LaTeX}$, permitiendo crear diapositivas profesionales que integran contenido matemático, gráficos computacionales y estructuras pedagógicas especializadas.

A diferencia de `RenderToLatexDocument`, esta función trabaja con una abstracción de *slides* construidos mediante funciones auxiliares como `BeamerSlide`, `BeamerTwoColumnSlide` y `BeamerSectionSlide`, que encapsulan patrones comunes en presentaciones académicas.

El sistema maneja automáticamente aspectos como la diapositiva de título, la aplicación de temas visuales (*Madrid*, *Berlin*, *Copenhagen*, entre otros), esquemas de color, y la adaptación de elementos gráficos para visualización en proyección. Los ejemplos siguientes muestran cómo construir presentaciones completas desde simples secuencias de diapositivas hasta estructuras complejas con secciones, contenido en columnas y distintos elementos visuales. A continuación se proporcionan dos tipos de configuraciones *Beamer*: una minimalista y la otra de naturaleza académica.

Nota: Se han eliminado las tildes de los textos para mostrar en código la información en el presente documento, sin embargo, en *Mathematica* el empleo de tildes junto con la instrucción `RenderToLatexBeamer` no despliega ningún tipo de error en el archivo `.tex` creado de manera automática.

Ejemplo 21 Configuración minimalista

```
contenido = {
  (* Slide de introduccion *)
  BeamerSlide["Introduccion",
    "Esta presentacion cubre los puntos principales del tema."],

  (* Slide con lista *)
  BeamerSlide["Objetivos",
    {"\\begin{itemize}",
      "\\item Objetivo principal",
      "\\item Objetivo secundario",
      "\\item Resultados esperados",
      "\\end{itemize}"}],

  (* Slide con ecuacion *)
  BeamerSlide["Marco Teorico",
    {"La ecuacion fundamental es:",
      "$$E = mc^2$$",
      "donde $E$ es energia, $m$ es masa y $c$ es velocidad de la luz."}],

  (* Slide con grafico *)
  BeamerSlide["Resultados",
    {"Analisis de los datos:",
      Plot[Sin[x], {x, 0, 2*Pi}, PlotLabel ->"Funcion Seno"]}],

  (* Slide de conclusiones *)
  BeamerSlide["Conclusiones",
    {"\\begin{enumerate}",
      "\\item Primera conclusion importante",
      "\\item Segunda conclusion relevante",
```

```

        "\\item Trabajo futuro",
        "\\end{enumerate}"]}]
};

RenderToLatexBeamer["presentacion", contenido,
  "Title" → "Mi Presentacion",
  "Author" → "Tu Nombre"]

```

La ejecución del comando `RenderToLatexBeamer` construye automáticamente el archivo `.tex` de la presentación, en este caso denominado `presentacion.tex`, guardado en una carpeta llamada `presentacion_beamer` en `Downloads`. También, se muestra la carpeta `figures` con la gráfica `Plot` de este ejemplo, con extensión `.png`.

Ejemplo 22 Configuración académica

```

contenido = {
  (* Agenda *)
  BeamerSlide["Agenda", {"\\tableofcontents"}],

  (* Seccion 1 *)
  BeamerSectionSlide["Introduccion"],
  BeamerSlide["Motivacion",
    {"Los metodos tradicionales presentan limitaciones:",
      "\\begin{itemize}",
      "\\item Escalabilidad limitada",
      "\\item Precision insuficiente",
      "\\item Alto costo computacional",
      "\\end{itemize}"}],

  (* Seccion 2 *)
  BeamerSectionSlide["Metodologia"],
  BeamerSlide["Algoritmo Propuesto",
    {"\\textbf{Paso 1:} Preprocesamiento",
      "\\textbf{Paso 2:} Entrenamiento adaptativo",
      "\\textbf{Paso 3:} Validacion cruzada"}],
  BeamerTwoColumnSlide["Arquitectura del Modelo",
    (* Columna izquierda *)
    StringJoin["\\textbf{Entrada:}", "\\begin{itemize}",
      "\\item Datos en bruto", "\\item Preprocesamiento",
      "\\end{itemize}"],
    (* Columna derecha *)
    StringJoin["\\textbf{Salida:}", "\\begin{itemize}",
      "\\item Predicciones", "\\item Metricas de calidad",
      "\\end{itemize}"]],

  (* Seccion 3 *)
  BeamerSectionSlide["Resultados"],

  (* Separador visual usando "plain" *)
  BeamerSlideWithOptions["",
    {"\\begin{center}", "\\vspace{2cm}",
      "\\Huge \\textcolor{blue}{ANALISIS DE RESULTADOS}",
      "\\vspace{2cm}", "\\end{center}"},
    "plain"],

  (* Slide con datos extensos usando "allowframebreaks" *)
  BeamerSlideWithOptions["Resultados Experimentales Detallados",
    {"\\textbf{Dataset 1: MNIST}", "\\begin{itemize}",
      "\\item Precision: 98.7\\% vs 94.2\\% (tradicional)",

```

```

    "\\item Tiempo de entrenamiento: 45min vs 120min",
    "\\item Memoria utilizada: 2.1GB vs 3.8GB", "\\end{itemize}", "",
    "\\textbf{Dataset 2: CIFAR-10}", "\\begin{itemize}",
    "\\item Precision: 89.3\\% vs 82.1\\% (tradicional)",
    "\\item Tiempo de entrenamiento: 2.3h vs 4.1h",
    "\\item Convergencia: epoca 45 vs epoca 78", "\\end{itemize}", "",
    "\\textbf{Dataset 3: ImageNet}", "\\begin{itemize}",
    "\\item Precision top-5: 94.1\\% vs 91.2\\% (tradicional)",
    "\\item Tiempo de entrenamiento: 12h vs 18h",
    "\\item Estabilidad: \\[PlusMinus]0.3\\% vs \\[PlusMinus]1.2\\%",
    "\\end{itemize}}",
    "allowframebreaks"],

(* Slide tecnico detallado con "shrink" *)
BeamerSlideWithOptions["Detalles Tecnicos del Algoritmo",
{"\\textbf{Parametros de Configuracion:}", "\\begin{description}",
  "\\item[Tasa de aprendizaje inicial:] $\\alpha_0 = 0.001$",
  "\\item[Factor de decaimiento:] $\\gamma = 0.95$",
  "\\item[Regularizacion L2:] $\\lambda = 0.0001$",
  "\\item[Batch size adaptativo:] $b \\in [32, 512]$",
  "\\item[Criterio de parada:] $|\\mathcal{L}_{t} - \\mathcal{L}_{t-1}| < \\epsilon$",
  "\\item[Ventana de convergencia:] $w = 10$ epocas",
  "\\end{description}",
  "\\textbf{Complejidad computacional:} $O(n \\log n)$ por epoca"},
"shrink"],

(* Slide con implementacion usando "fragile" *)
BeamerSlideWithOptions["Implementacion del Algoritmo Propuesto",
{"\\begin{lstlisting}[language=Python]",
  "def algoritmo_adaptativo(datos, tolerancia=1e-6):",
  "    modelo = inicializar_modelo()",
  "    error_anterior = float('inf')", " ", " while True:",
  "    error_actual = entrenar_epoca(modelo, datos)",
  "    if abs(error_anterior - error_actual) < tolerancia:",
  "    break", "    error_anterior = error_actual",
  "    ajustar_hiperparametros(modelo, error_actual)", " ",
  "    return modelo", "\\end{lstlisting}}", "fragile"],

BeamerSlide["Experimentos",
{"Comparacion de rendimiento:",
  Plot[{x^2, 2*x}, {x, 0, 3}, PlotLabel -> "Precision vs Tiempo"]}],

BeamerSlide["Metricas de Evaluacion",
{"\\begin{table}", "\\begin{tabular}{|l|c|c|}", "\\hline",
  "Metodo & Precision & Tiempo \\\\", "\\hline",
  "Tradicional & 85\\% & 2.3s \\\\",
  "Propuesto & 92\\% & 1.1s \\\\", "\\hline", "\\end{tabular}",
  "\\end{table}"]],

(* Conclusiones *)
BeamerSectionSlide["Conclusiones"],
BeamerSlide["Logros Principales",
{"\\begin{enumerate}",
  "\\item Mejora del 7\\% en precision",
  "\\item Reduccion del 50\\% en tiempo",
  "\\item Implementacion escalable", "\\end{enumerate}"}],
BeamerSlide["Trabajo Futuro",
{"Proximos pasos:", "\\begin{itemize}",
  "\\item Extension a otros dominios",
  "\\item Optimizacion de hiperparametros",
  "\\item Validacion en datos reales",

```

```

        "\\end{itemize}"}]],

(* Slide de agradecimientos sin numeracion *)
BeamerSlideWithOptions["Agradecimientos",
{"\\begin{center}", "Agradecimientos especiales a:", "\\vspace{0.5cm}",
  "\\begin{itemize}", "\\item Dr. Ana Martinez (supervision)",
  "\\item Equipo de investigacion ML-Lab",
  "\\item Universidad Nacional (financiamiento)",
  "\\item Comunidad open-source", "\\end{itemize}", "\\end{center}"},
"noframenumbering"],

(* Slide final *)
BeamerSlide["Gracias por su atencion",
{"\\begin{center}", "\\huge Preguntas", "\\vspace{1cm}",
  "\\large roberto.martin@universidad.edu",
  "\\end{center}"}]],

(* Referencias extensas con "allowframebreaks" *)
BeamerSlideWithOptions["Referencias Bibliograficas",
{"\\begin{thebibliography}{99}",
  "\\bibitem{ref1} Smith, J. et al. (2023). \\textit{Adaptive
    Learning in Neural Networks}. Nature Machine Intelligence,
    4(2), 123-145.",
  "\\bibitem{ref2} Garcia, M. and Lopez, A. (2022).
    \\textit{Optimization Techniques for Deep Learning}. ICML
    Proceedings, pp. 456-467.",
  "\\bibitem{ref3} Chen, L. (2023). \\textit{Convergence Analysis
    of Adaptive Algorithms}. Journal of Machine Learning Research,
    24, 89-112.",
  "\\bibitem{ref4} Williams, R. et al. (2022). \\textit{Scalable
    Neural Network Architectures}. NeurIPS, pp. 2341-2354.",
  "\\bibitem{ref5} Thompson, K. (2021). \\textit{Efficient
    Training Methods for Large Models}. arXiv:2105.12345.",
  "\\bibitem{ref6} Martinez, C. and Brown, S. (2023).
    \\textit{Memory-Efficient Deep Learning}. AAAI Conference,
    pp. 1123-1134.",
  "\\end{thebibliography}"}, "allowframebreaks"]
];

RenderToLatexBeamer["academica", contenido,
"Title" → "Nuevos Metodos en Machine Learning",
"Subtitle" → "Avances en Redes Neuronales",
"Author" → "Dr. Roberto Martin",
"Institute" → "Universidad Nacional",
>Date" → "Marzo 2025",
"Theme" → "Warsaw",
"Preamble" → "full"]

```

En **Mathematica** lo anterior genera como salida la carpeta y el archivo: *academica_beamer/academica.tex* guardado en *Downloads*, además de la carpeta *figures* con la gráfica **Plot** respectiva, en formato *.png*.

Para más ejemplos

Si el lector desea obtener una gama más amplia de ejemplos relacionados con todas las instrucciones mencionadas en esta sección, puede consultar de manera complementaria el archivo: *1. Ejemplos RENDERIZADO LaTeX.nb*, disponible en el enlace de descarga del paquete **V¹T_EX** (ver página xiv).

2

Sistema compilador $\LaTeX$

¿Qué aprenderás en este capítulo?

- ▶ Compilar documentos $\LaTeX$ directamente desde **Mathematica** sin cambiar de entorno.
- ▶ Detectar automáticamente editores $\LaTeX$ instalados (TeXstudio, Overleaf, TeXmaker).
- ▶ Generar documentos completos con estructura modular: artículos, reportes, libros y presentaciones *Beamer*.
- ▶ Configurar bibliografías, entornos matemáticos y estilos de código de manera automática.
- ▶ Optimizar el flujo de trabajo eliminando la fragmentación entre cálculo y documentación.

La integración eficiente entre el cálculo simbólico y la documentación científica requiere no solo la capacidad de generar código $\LaTeX$, sino también de compilarlo de manera adecuada sin abandonar el entorno de trabajo computacional. El sistema compilador de **Vi $\LaTeX$** aborda esta necesidad mediante un conjunto de funciones que automatizan el ciclo completo desde la edición hasta la visualización de documentos PDF.

A diferencia de los flujos de trabajo tradicionales que requieren alternar **Mathematica** manualmente con editores $\LaTeX$ externos, el sistema compilador implementa un enfoque unificado que detecta automáticamente los editores instalados en el sistema operativo, localiza las distribuciones $\TeX$ disponibles, y gestiona el proceso de compilación directamente desde **Wolfram Mathematica**.

El núcleo del sistema reside en la detección inteligente multiplataforma donde se escanea las ubicaciones estándar en *macOS*, *Windows* y *Unix*, identificando editores como *TeXShop*, *TeXstudio*, *TeXmaker*, *Texifier*, *LaTeXiT*, *TeXworks* y *Visual Studio Code*. Esta información se almacena en la variable global `$LaTeXEditors`, proporcionando un registro persistente de las herramientas disponibles durante la sesión de **Mathematica**.

La compilación automatizada se logra mediante el comando `CompiladorTex`, que ejecuta *pdflatex* y maneja la apertura del documento resultante en el visor apropiado del sistema operativo. Para ello, se incluye una búsqueda inteligente de la instalación *pdflatex* en ubicaciones estándar de *MacTeX*, *MiKTeX* y *TeX Live*.

Una característica distintiva es el manejo robusto de errores y la flexibilidad en la escogencia del entorno de edición. El sistema acepta coincidencias parciales y búsquedas *case-insensitive* para especificaciones de editores, facilitando su uso. Cuando la compilación falla, proporciona mensajes informativos que guían al usuario hacia la resolución del problema sin afectar la continuidad operativa.

2.1 Comandos del sistema compilador $\text{\LaTeX}$

El sistema compilador del paquete $\text{\VilTeX}$ proporciona cinco funciones esenciales que trabajan en conjunto para automatizar la detección de editores $\text{\TeX}$ y la compilación de documentos desde **Wolfram Mathematica**.

Nota: En **Mathematica** al instalar el paquete $\text{\VilTeX}$ se cuenta con la ayuda `?SistemaCompilador` que despliega una explicación detallada sobre el sistema compilador de documentos $\text{\TeX}$. `SistemaCompilador` no es una función ejecutable, sino un símbolo de documentación que agrupa conceptualmente el conjunto completo de herramientas para la compilación automatizada de documentos $\text{\TeX}$.

Al correr `?SistemaCompilador` en **Wolfram**, se muestra información comprehensiva sobre todas las funcionalidades del módulo compilador, incluyendo las funciones destacadas, variables del sistema, características multiplataforma, y ejemplos de uso. Los cinco comandos fundamentales de este módulo son los siguientes:

- `CompiladorTex[filePath, opts]`: Esta es la función principal del sistema compilador. Orquesta el proceso completo de apertura de un archivo $\text{\TeX}$ en un editor y su compilación automática a PDF usando `pdflatex`. El argumento `filePath` es un *string* con la ruta al archivo `.tex` que se desea compilar y si no se incluye la extensión `.tex`, se añade automáticamente. El comando cuenta con las opciones `"PreferredEditor" -> Automatic` (por defecto usa el primer editor detectado), o bien, `"PreferredEditor" -> "NombreEditor"` que especifica un editor particular. La búsqueda es flexible, aceptando coincidencias exactas (por ejemplo: `TeXstudio`), *case-insensitive* (por ejemplo: `textstudio` o `TEXSTUDIO`) o coincidencias parciales (tales como: `studio` o `tex`) encontrando el primer editor que contenga ese patrón. Soporta los editores: `TeXShop` (macOS), `TeXstudio` (multiplataforma), `TeXmaker` (multiplataforma), `Texifier` (macOS), `LaTeXiT` (macOS), `TeXworks` (multiplataforma) y `Visual Studio Code` (multiplataforma). La sentencia `CompiladorTex` se caracteriza esencialmente por:
 - Verificar que el archivo a compilar existe y añadir la extensión `.tex` en caso de ser necesario.
 - Localizar la instalación de `pdflatex` en ubicaciones estándar o en el PATH del sistema.
 - Elegir el editor según la opción especificada por el usuario o emplear el primero disponible, para posteriormente abrir el archivo `.tex` en dicho entorno, usando comandos apropiados del sistema operativo.
 - Ejecutar `pdflatex` con `flag -interaction=nonstopmode` para compilación no interactiva y comprobar que el PDF se genera correctamente, pasando el documento al visor adecuado del sistema.
 - Retornar la ruta completa al PDF generado si la compilación fue exitosa y si no, devolver la ruta al archivo `.tex` para facilitar las correcciones del error, o bien, `$Failed` si no se encuentra `pdflatex` o si el archivo `.tex` no existe.
 - Si la compilación falla, muestra detalles del error y se sugiere revisar el archivo `.tex` en el editor correspondiente.
 - Si el entorno de edición especificado no se encuentra, se usa el editor por defecto y se notifica al usuario.
 - El proceso incluye una pausa de 1 segundo entre la apertura del ambiente de edición y la compilación del archivo `.tex`.
- `DetectLaTeXEditorsPattern[]`: Ejecuta una búsqueda exhaustiva de editores $\text{\TeX}$ instalados en el sistema operativo. Esta función escanea ubicaciones estándar según la plataforma. En `macOS` busca

recursivamente en */Applications*, en *Windows* verifica rutas de *Program Files* y directorios de usuario, y en *Unix* explora */usr/bin* y */usr/local/bin*. Utiliza patrones de búsqueda específicos para cada editor (*case-insensitive*). En *macOS* busca archivos *.app* que coincidan con nombres como *texshop*, *textstudio*, *texmaker*, entre otros. En *Windows* busca ejecutables *.exe* en ubicaciones estándar de *MiKTeX* y *TeX Live*. El proceso puede tardar varios segundos debido al escaneo del sistema de archivos. Esta instrucción se ejecuta automáticamente al cargar el paquete **V_lT_EX**.

- **ShowAvailableEditors[]**: Muestra en pantalla una lista formateada de todos los editores $\LaTeX$ detectados en el sistema. Si no hay editores previamente detectados ejecuta automáticamente **DetectLaTeXEditorsPattern** antes de mostrar los resultados.
- **RefreshEditors[]**: Fuerza una nueva detección de editores $\LaTeX$, limpiando la caché anterior. Esta función es útil cuando se instala un nuevo editor después de haber cargado el paquete **V_lT_EX**, ya que permite actualizar la lista de ambientes de edición disponibles sin reiniciar **Wolfram Mathematica**. **RefreshEditors** ejecuta la detección automática con el mensaje “Detectando editores LaTeX...”, si ya hay editores detectados, muestra el mensaje “Mostrando editores ya detectados...” e imprime cada editor con formato: **NombreEditor** → **Ejecutable**, si no encuentra editores, muestra sugerencias de instalación según el sistema operativo.

En resumen

Al cargar **V_lT_EX** se ejecuta automáticamente **DetectLaTeXEditorsPattern[]**. Se recomienda usar **ShowAvailableEditors[]** para verificar qué editores están disponibles. Si se instala un nuevo editor, se debe emplear **RefreshEditors[]** para actualizar la lista. La función **CompiladorTex** se utiliza para compilar el archivo *.tex* de interés. El sistema maneja automáticamente la compilación y la visualización del PDF resultante.

2.2 Ejemplos del sistema compilador $\LaTeX$

Los ejemplos presentados en esta sección ilustran la integración completa entre la generación automatizada de documentos $\LaTeX$ y el sistema compilador del paquete **V_lT_EX**. A diferencia de las dinámicas de trabajo tradicionales que requieren múltiples pasos manuales -crear el archivo *.tex*, abrir un editor externo $\LaTeX$, compilar manualmente, y localizar el PDF resultante- estos ejemplos demuestran cómo **V_lT_EX** unifica todo el proceso con pocas líneas de código.

Se presentan dos escenarios representativos que cubren casos de uso académico frecuente: la compilación de un documento estándar generado con **RenderToLatexDocument** y la compilación de una presentación *Beamer* creada mediante **RenderToLatexBeamer**. En ambos casos, el sistema maneja automáticamente la detección de editores, la localización de *pdflatex*, la ejecución de la compilación, y la apertura del PDF resultante.

Cada ejemplo muestra dos variantes de compilación: primero utilizando el editor detectado automáticamente (configuración por defecto), y segundo especificando explícitamente *TeXstudio* mediante la opción "**PreferredEditor**". Esta dualidad demuestra la flexibilidad del sistema para adaptarse tanto a usuarios que prefieren configuraciones automáticas como a aquellos que requieren control explícito sobre sus herramientas de edición.

Los mensajes generados durante la ejecución -detección de editores, localización de *pdflatex*, apertura del editor, progreso de compilación, y confirmación del PDF generado- proporcionan retroalimentación transparente del proceso, facilitando la depuración cuando sea necesario y confirmando cada etapa secuencial.

Nota: Los siguientes ejemplos asumen que el paquete **W_UL_AT_EX** ha sido cargado previamente y que el sistema operativo cuenta con al menos un editor L^AT_EX y una distribución T_EX instalados.

Ejemplo 23 Documento simple

```
contenido = {"\\section{Introduccion}",
  "Este es un documento muy basico sin elementos especiales.", "",
  "\\section{Desarrollo}",
  "Aqui se presenta el contenido principal del documento.", "",
  "\\subsection{Punto Importante}",
  "Este subsection contiene informacion relevante.", "",
  "\\section{Conclusion}",
  "En conclusion, este documento demuestra la configuracion
  minimalista."};

path = RenderToLatexDocument["documento", contenido,
  "Title" -> "Documento Simple", "Author" -> "Usuario"]
CompiladorTex[path]
CompiladorTex[path, "PreferredEditor" -> "TeXstudio"]
```

Esto produce como salida en **Mathematica** lo siguiente:

```
Usando editores ya detectados: 4 disponibles
Buscando pdflatex...
pdflatex encontrado: /Library/TeX/texbin/pdflatex
Abriendo archivo en TeXShop...
Editor abierto correctamente
Compilando documento LaTeX...
Directorio de trabajo: /Users/.../Downloads/documento/
Archivo: documento
Compilación exitosa
PDF generado: /Users/.../Downloads/documento/documento.pdf
/Users/.../Downloads/documento/documento.pdf
```

Y con la opción `"PreferredEditor" -> "TeXstudio"`, se despliega el *Out*:

```
Usando editores ya detectados: 4 disponibles
Buscando pdflatex...
pdflatex encontrado: /Library/TeX/texbin/pdflatex
Editor 'TeXstudio' encontrado (match exacto)
Abriendo archivo en TeXstudio...
Editor abierto correctamente
Compilando documento LaTeX...
Directorio de trabajo: /Users/.../Downloads/documento/
Archivo: documento
Compilación exitosa
PDF generado: /Users/.../Downloads/documento/documento.pdf
/Users/.../Downloads/documento/documento.pdf
```

Además, el archivo `.tex` y el PDF compilado se abren automáticamente en la computadora del usuario.

Ejemplo 24 Presentación Beamer

```

contenido = {(*Slide de introduccion*)
  BeamerSlide["Introduccion",
    "Esta presentacion cubre los puntos principales del
    tema."], (*Slide con lista*)
  BeamerSlide[
    "Objetivos", {"\\begin{itemize}", "\\item Objetivo principal",
      "\\item Objetivo secundario", "\\item Resultados esperados",
      "\\end{itemize}" }], (*Slide con ecuacion*)
  BeamerSlide[
    "Marco Teorico", {"La ecuacion fundamental es:", "$$E = mc^2$$",
      "donde $E$ es energia, $m$ es masa y $c$ es velocidad de la
      luz."}], (*Slide con grafico*)
  BeamerSlide[
    "Resultados", {"Analisis de los datos:",
      Plot[Sin[x], {x, 0, 2*Pi},
      PlotLabel -> "Funcion Seno"}], (*Slide de conclusiones*)
  BeamerSlide[
    "Conclusiones", {"\\begin{enumerate}",
      "\\item Primera conclusion importante",
      "\\item Segunda conclusion relevante", "\\item Trabajo futuro",
      "\\end{enumerate}" }]];

path = RenderToLatexBeamer["presentacion", contenido,
  "Title" -> "Mi Presentacion", "Author" -> "Tu Nombre"]
CompiladorTex[path]
CompiladorTex[path, "PreferredEditor" -> "TeXstudio"]

```

La salida en **Mathematica** muestra:

```

Usando editores ya detectados: 4 disponibles
Buscando pdflatex...
pdflatex encontrado: /Library/TeX/texbin/pdflatex
Abriendo archivo en TeXShop...
Editor abierto correctamente
Compilando documento LaTeX...
Directorio de trabajo: /Users/.../Downloads/presentacion_beamer/
Archivo: presentacion
Compilación exitosa
PDF generado: /Users/.../Downloads/presentacion_beamer/presentacion.pdf
/Users/.../Downloads/presentacion_beamer/presentacion.pdf

```

Y con la opción "PreferredEditor" -> "TeXstudio" se produce el *Out*:

```

Usando editores ya detectados: 4 disponibles
Buscando pdflatex...
pdflatex encontrado: /Library/TeX/texbin/pdflatex
Editor 'TeXstudio' encontrado (match exacto)
Abriendo archivo en TeXstudio...
Editor abierto correctamente
Compilando documento LaTeX...
Directorio de trabajo: /Users/.../Downloads/presentacion_beamer/
Archivo: presentacion
Compilación exitosa

```

2.2 Ejemplos del sistema compilador $\text{\LaTeX}$

PDF generado: `/Users/.../Downloads/presentacion_beamer/presentacion.pdf`
`/Users/.../Downloads/presentacion_beamer/presentacion.pdf`

También, el archivo `.tex` y el PDF compilado se abren automáticamente en la computadora del usuario.

3

Sistema *Plot* para $\text{\LaTeX}$

¿Qué aprenderás en este capítulo?

- ▶ Generar gráficos con fuentes tipográficas profesionales compatibles con $\text{\LaTeX}$.
- ▶ Aplicar estilos académicos a visualizaciones de **Mathematica**.
- ▶ Configurar la apariencia visual característica de publicaciones científicas.
- ▶ Personalizar ejes, etiquetas y leyendas con notación matemática consistente.
- ▶ Exportar gráficos listos para integración en documentos académicos.

La visualización de datos científicos requiere no solo precisión matemática sino también consistencia tipográfica con el resto de la documentación académica. El sistema de *plotting* con estilo $\text{\LaTeX}$ del paquete **$\text{\LaTeX}$** aborda esta necesidad mediante la integración de fuentes tipográficas especializadas y temas visuales predefinidos que mantienen coherencia con las convenciones de publicaciones científicas.

A diferencia de los gráficos estándar de **Mathematica**, el sistema *Plot* de **$\text{\LaTeX}$** aplica automáticamente fuentes *Latin Modern*, *CMU*, *STIX* y otras tipografías características de documentos $\text{\LaTeX}$ a todas las etiquetas, números de ejes, leyendas y títulos de una gráfica **Plot** en **Wolfram**.

Esta transformación tipográfica preserva la calidad visual mientras elimina la inconsistencia estética que surge al combinar gráficos generados computacionalmente con texto compilado en $\text{\LaTeX}$. El núcleo del sistema reside en la configuración global `$\text{\LaTeXFontConfig}`, que almacena las preferencias de fuentes para texto principal, expresiones matemáticas y contenido monoespaciado.

La función `SetLaTeXFonts` permite personalizar esta configuración, verificando automáticamente la disponibilidad de fuentes en el sistema y proporcionando alternativas de respaldo cuando las fuentes solicitadas no están instaladas.

Una característica distintiva es el sistema de temas predefinidos que encapsula convenciones visuales de diferentes disciplinas académicas. El tema *academic* incorpora grillas punteadas y marcos apropiados para artículos de investigación; *physics* emplea fondos grises claros y líneas azules que evocan libros de física clásicos; *mathematics* mantiene un diseño minimalista con ejes simples; mientras que *engineering* utiliza grillas punteadas y líneas rojas características de documentación técnica.

El procesamiento inteligente de divisiones de ejes representa un avance significativo sobre las funcionalidades nativas. El sistema genera automáticamente 8 divisiones en el eje horizontal y 6 en el vertical,

aplicando formato matemático apropiado que incluye notación científica para valores extremos y representación decimal optimizada para rangos intermedios. Este formateo se integra completamente con las fuentes $\text{\LaTeX}$ configuradas, garantizando legibilidad y consistencia.

La función `PlotLaTeXMultiple` extiende las capacidades al manejo de múltiples expresiones, asignando automáticamente colores diferenciados de una paleta predefinida y manteniendo el estilo $\text{\LaTeX}$ en todas las leyendas y anotaciones. Esta automatización elimina la necesidad de especificación manual de estilos para cada curva graficada.

El sistema mantiene compatibilidad total con todas las opciones nativas de `Plot`, permitiendo que parámetros como `PlotStyle`, `GridLines`, `Frame` y otros se combinen sin conflicto con las transformaciones tipográficas aplicadas por el paquete `VfTeX`.

3.1 Comandos del sistema *Plot* para $\text{\LaTeX}$

En **Mathematica** al instalar el paquete `VfTeX` se cuenta con la ayuda `?SystemaPlot` con el objetivo de brindar una explicación pormenorizada sobre el sistema completo de *plotting* con estilo $\text{\LaTeX}$.

El sistema *Plot* con estilo $\text{\LaTeX}$ de `VfTeX` proporciona funciones especializadas para la creación de gráficos con tipografía profesional consistente con documentos $\text{\LaTeX}$. Para ello, cuenta con seis funciones principales:

- `SetLaTeXFonts[mainFont, mathFont, monoFont]`: Configura las fuentes globales del sistema de *plotting* para toda la sesión de **Mathematica**. En `mainFont` (*string* opcional) se define la fuente principal para el texto en general (por defecto corresponde a *Latin Modern Roman*). Por otra parte, el argumento `mathFont` (*string* opcional) establece la fuente para expresiones matemáticas (por defecto toma *Latin Modern Math*). Finalmente, el parámetro `monoFont` (*string* opcional) ajusta la fuente monoespaciada (por defecto, *Latin Modern Mono*). Este comando verifica la disponibilidad de cada fuente especificada. Si una fuente no está disponible, utiliza automáticamente la fuente de respaldo e imprime un reporte en pantalla mostrando las fuentes configuradas exitosamente. La instrucción no retorna un valor, su efecto es modificar el estado global de la variable `$LaTeXFontConfig`.
- `ShowLaTeXFonts[]`: Muestra las fuentes $\text{\LaTeX}$ recomendadas que están disponibles en el sistema operativo del ordenador. El comando escanea una lista predefinida de fuentes tipográficas comunes en distribuciones $\text{\LaTeX}$ (tales como: *Latin Modern*, *CMU*, *Times*, *Palatino*, *Charter*, *EB Garamond*, *Libertinus Serif*, *STIX Two Text*, entre otras) y verifica cuáles están instaladas en el sistema usando una variable denominada `$FontFamilies`. Presenta los resultados en formato de tabla con dos columnas. No retorna ningún valor, solo imprime información en pantalla.
- `ShowAllAvailableFonts[category]`: Explora y muestra todas las fuentes tipográficas disponibles en el sistema operativo, organizadas por categoría. Esta función es útil para descubrir qué fuentes se pueden usar con la opción "`CustomFont`" en las funciones de *plotting*. El parámetro `category` (*string* opcional) establece la categoría de fuentes a mostrar. Si se proporciona una categoría no válida, imprime un mensaje listando las categorías disponibles. Si una categoría específica no tiene fuentes disponibles, imprime "Ninguna encontrada". No retorna un valor, solo imprime información formateada en pantalla. Las categorías disponibles son:
 - "`all`": Muestra las primeras 50 fuentes de todas las vacantes en el sistema, ordenadas alfabéticamente.
 - "`categorized`": Muestra todas las fuentes organizadas en categorías (*serif*, *sans serif*, *mono-space*, *math*, entre otras).
 - "`serif`": Filtra y muestra únicamente fuentes con *serifa* (recomendadas para documentos $\text{\LaTeX}$ académicos).
 - "`sansserif`": Filtra y muestra únicamente fuentes *sans serif*.

- "**monospace**": Filtra y muestra únicamente fuentes monoespaciadas (tipo código).
- "**math**": Filtra y muestra únicamente fuentes matemáticas y de símbolos.
- **PlotLaTeXStyle**[*expr*, *range*, *opts*]: Función principal que crea gráficos con etiquetas, números y títulos en estilo $\text{\LaTeX}$ completamente personalizable. El parámetro *expr* define una expresión matemática o lista de expresiones a graficar, el argumento *range* establece un rango de graficación en formato {*x*, *xmin*, *xmax*} y *opts* permite utilizar opciones estándar de **Plot** más el atributo "**CustomFont**" $\rightarrow$ "**NombreFuente**" que sobrescribe temporalmente la fuente configurada para la gráfica (acepta coincidencias parciales). Todas las opciones nativas de **Plot**, como por ejemplo: **PlotStyle**, **GridLines**, **Frame**, **ImageSize**, entre otras, son compatibles y se preservan en este comando.
- **PlotLaTeXThemed**[*expr*, *range*, *theme*, *opts*]: Crea gráficos con temas visuales predefinidos que encapsulan convenciones estéticas de diferentes disciplinas académicas. El argumento *expr* especifica una expresión o lista de expresiones a graficar, *range* define un rango de graficación usando el formato {*x*, *xmin*, *xmax*}, el parámetro *theme* (*string* opcional) permite definir un tema para la gráfica, por defecto corresponde a "**academic**" y *opts* se refiere a todas las opciones del comando **PlotLaTeXStyle** que son también aplicables. Los temas disponibles son:
 - "**academic**": Fondo blanco, grilla punteada gris, marco negro, línea azul gruesa. Ideal para artículos de investigación.
 - "**physics**": Fondo gris claro, grilla continua gris, marco negro, línea azul oscuro gruesa. Estilo de libros de física clásicos.
 - "**mathematics**": Sin grilla, ejes simples, fondo blanco, línea negra gruesa. Minimalista, apropiado para textos matemáticos.
 - "**engineering**": Fondo gris muy claro, grilla punteada, marco negro, línea roja gruesa. Estilo técnico.
- **PlotLaTeXMultiple**[*exprs*, *range*, *opts*]: Función especializada para graficar múltiples expresiones simultáneamente con asignación automática de colores y estilo $\text{\LaTeX}$ unificado. El parámetro *exprs* constituye la lista de expresiones matemáticas a graficar simultáneamente, el argumento *range* define el rango de graficación y *opts* admite las mismas opciones del comando **PlotLaTeXStyle**. Esta sentencia utiliza una paleta de colores predefinida: **Blue**, **Red**, **Green**, **Purple**, **Orange**, **Brown**, **Pink** y realiza una asignación cíclica, si hay más de 7 expresiones.

3.2 Ejemplos del sistema *Plot* para $\text{\LaTeX}$

Los ejemplos presentados en esta sección ilustran la aplicación práctica de las tres funciones más relevantes del sistema de *plotting* con estilo $\text{\LaTeX}$: **PlotLaTeXStyle**, **PlotLaTeXThemed** y **PlotLaTeXMultiple**.

A diferencia de la sección anterior que describió la sintaxis y parámetros de cada comando, aquí se demuestra su uso mediante casos concretos que abarcan desde gráficos básicos con tipografía $\text{\LaTeX}$ hasta visualizaciones complejas con múltiples expresiones y temas académicos especializados.

Cada ejemplo progresa en complejidad, mostrando cómo combinar opciones estándar de **Plot** con las capacidades tipográficas avanzadas del paquete **V $\text{\LaTeX}$** . Los casos presentados incluyen configuraciones de fuentes personalizadas mediante "**CustomFont**", aplicación de temas visuales predefinidos, y el manejo automático de paletas de colores para múltiples curvas.

Ejemplo 25 Plot con fuente personalizada

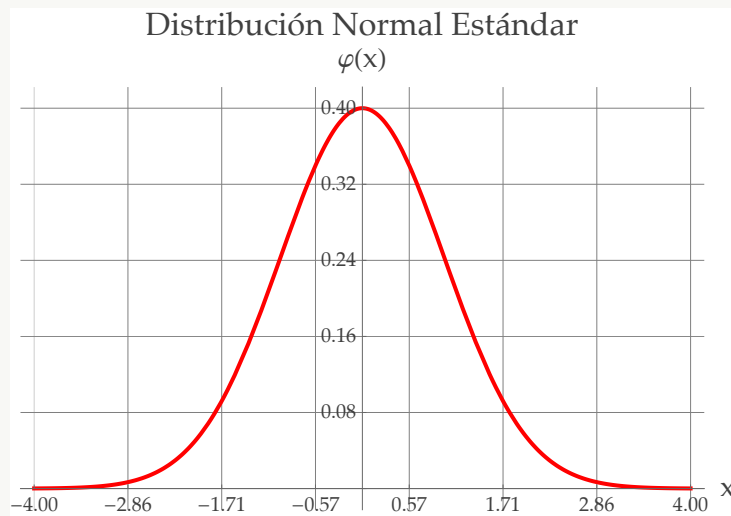
```
PlotLaTeXStyle[E^(-x^2/2)/Sqrt[2 Pi], {x, -4, 4},
"CustomFont"  $\rightarrow$  "Palatino",
```

```

PlotLabel → "Distribucion Normal Estandar",
AxesLabel → {"x", " $\varphi(x)$ "}, PlotStyle → {Thick, Red},
GridLines → Automatic]

```

Lo anterior muestra como salida en **Wolfram Mathematica**:



Como el lector puede apreciar, la instrucción `PlotLaTeXStyle` tiene un funcionamiento similar al comando nativo `Plot` de **Mathematica** con el detalle de mejorar el aspecto de la gráfica a tipografía $\text{\LaTeX}$. En este ejemplo, se ha usado la fuente *Palatino* con ese objetivo.

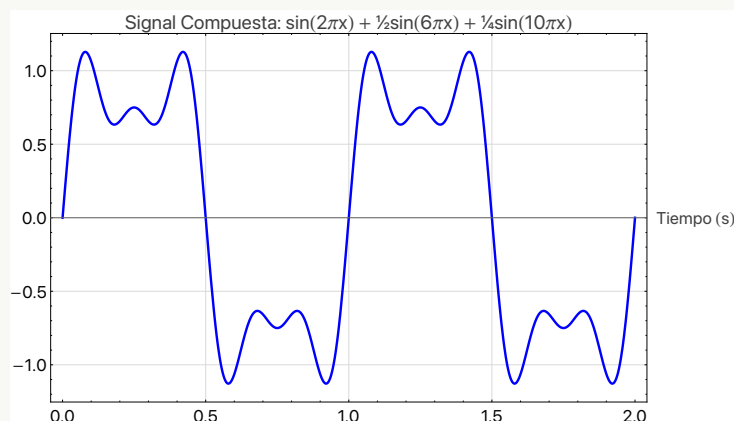
Ejemplo 26 Análisis de Fourier

```

signal = Sin[2*Pi*x] + 0.5*Sin[6*Pi*x] + 0.25*Sin[10*Pi*x];
PlotLaTeXStyle[signal, {x, 0, 2}, "CustomFont" → "Latin Modern Roman",
PlotLabel →
"Signal Compuesta: sin(2πx) + 1/2sin(6πx) + 1/4sin(10πx)",
AxesLabel → {"Tiempo (s)", "Amplitud"}, PlotStyle → {Thick, Blue},
GridLines → Automatic, GridLinesStyle → Directive[LightGray, Thin],
Frame → True, FrameStyle → Black, Background → White,
ImageSize → 600]

```

Lo que produce como salida en **Wolfram**:



Aquí, se ha personalizado el uso de la fuente a *Latin Modern Roman* para todos los textos y los números utilizados en la gráfica.

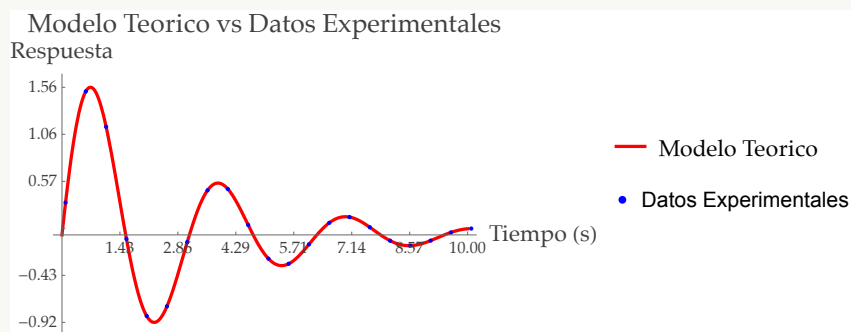
Ejemplo 27 Comparación con datos experimentales simulados

```

teorica = 2*Exp[-x/3]*Sin[2*x];
aleatorio = RandomReal[{-0.1, 0.1}];
datos = Table[{x /. x -> t + aleatorio, teorica /. x -> t + aleatorio},
  {t, 0, 10, 0.5}];
Show[PlotLaTeXStyle[teorica, {x, 0, 10}, "CustomFont" -> "Palatino",
  PlotLabel -> "Modelo Teorico vs Datos Experimentales",
  AxesLabel -> {"Tiempo (s)", "Respuesta"}, PlotStyle -> {Thick, Red},
  PlotLegends -> {"Modelo Teorico"}],
  ListPlot[datos, PlotStyle -> {Blue, PointSize[0.01]},
  PlotLegends -> {"Datos Experimentales"}]]

```

Esto genera como salida la siguiente gráfica con fuente *Palatino*:

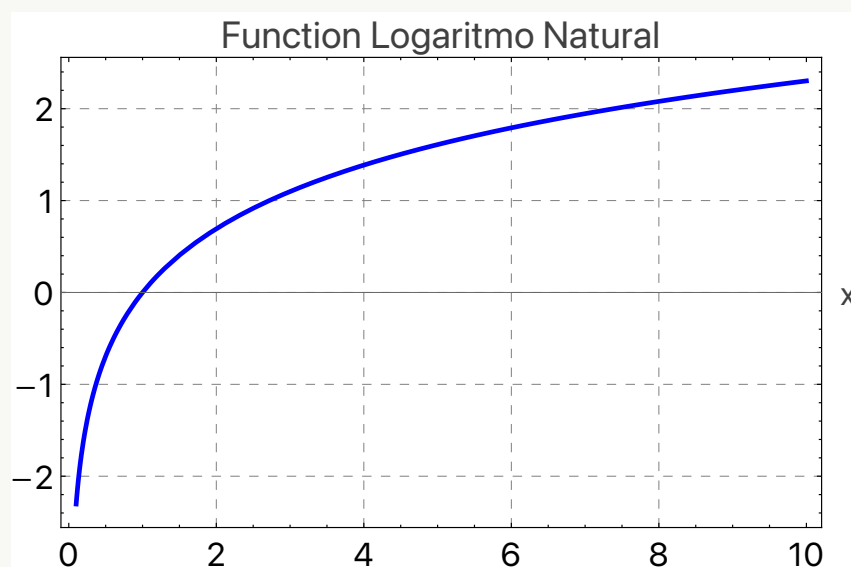
**Ejemplo 28 Función logarítmica con tema "academic"**

```

PlotLaTeXThemed[Log[x], {x, 0.1, 10}, "academic",
  "CustomFont" -> "Latin Modern Roman",
  PlotLabel -> "Function Logaritmo Natural", AxesLabel -> {"x", "ln(x)"}]

```

En **Mathematica** este código retorna la siguiente salida:

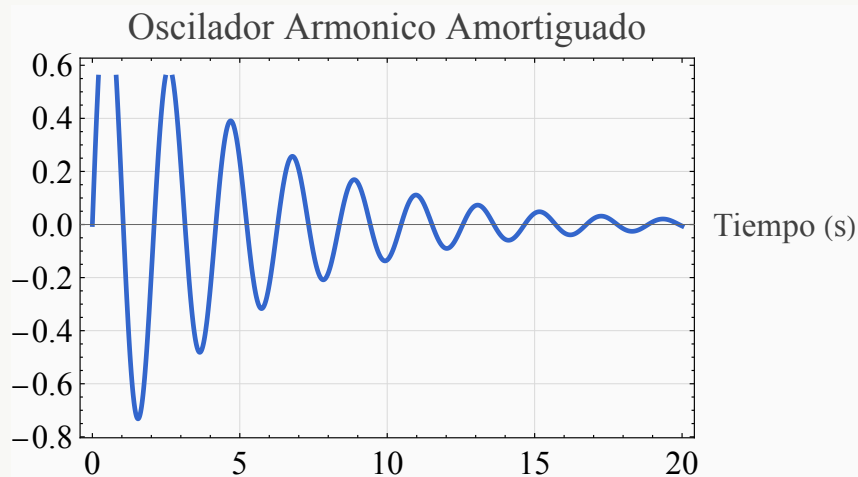


Los textos de la gráfica se han personalizado para utilizar la fuente *Latin Modern Roman* con tema "academic". Al igual que el comando `PlotLaTeXStyle`, la instrucción `PlotLaTeXThemed` trabaja de manera análoga a la sentencia nativa `Plot` de **Wolfram Mathematica**.

Ejemplo 29 Oscilador Armónico con tema "physics"

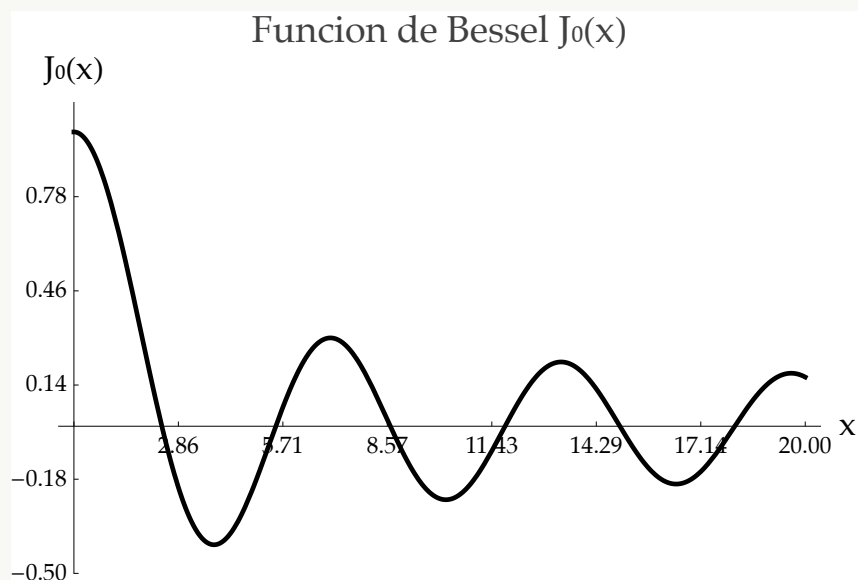
```
PlotLaTeXThemed[Exp[-0.2*x]*Sin[3*x], {x, 0, 20}, "physics",
"CustomFont" ->"Times New Roman",
PlotLabel ->"Oscilador Armonico Amortiguado",
AxesLabel ->{"Tiempo (s)", "Desplazamiento (m)"}]
```

En **Wolfram** se despliega la siguiente gráfica con tema "physics" y font personalizado a *Times New Roman*:

**Ejemplo 30** Función de Bessel con tema "mathematics"

```
PlotLaTeXThemed[BesselJ[0, x], {x, 0, 20}, "mathematics",
"CustomFont" ->"Palatino", PlotLabel ->"Funcion de Bessel J_0(x)",
AxesLabel ->{"x", "J_0(x)", PlotRange ->{-0.5, 1.1}]
```

La salida corresponde a la gráfica presentada a continuación, con fuente *Palatino* y tema **mathematics**:



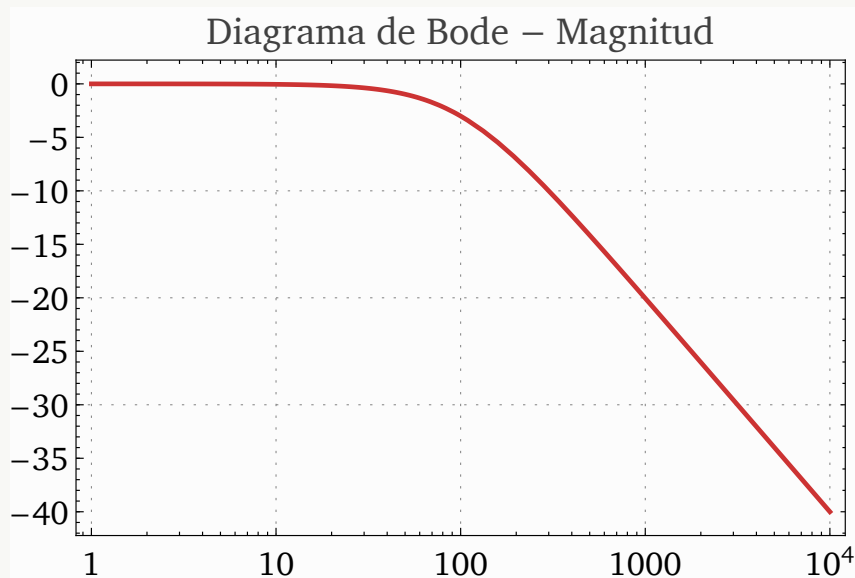
Ejemplo 31 Diagrama de Bode con tema "engineering"

```

magnitud = 20*Log10[1/Sqrt[1 + (x/100)^2]];
PlotLaTeXThemed[magnitud, {x, 1, 10000}, "engineering",
"CustomFont" -> "Charter", PlotLabel -> "Diagrama de Bode - Magnitud",
AxesLabel -> {"Frecuencia (Hz)", "Magnitud (dB)"},
ScalingFunctions -> {"Log", "Linear"}]

```

Lo anterior despliega en **Wolfram**:



Lo que constituye un ejemplo de gráfica con tema **engineering**.

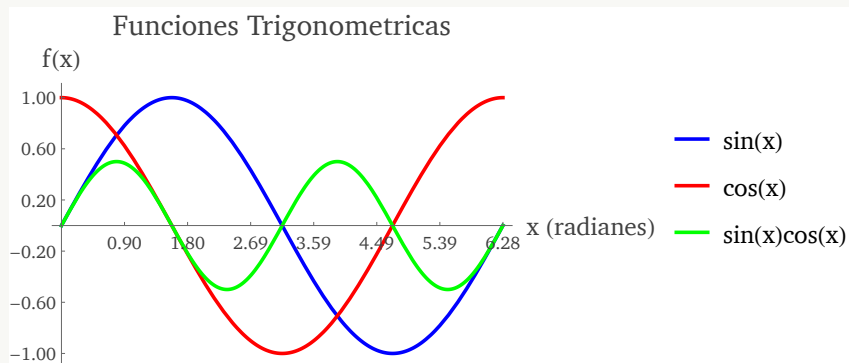
Ejemplo 32 Funciones trigonométricas básicas

```

PlotLaTeXMultiple[{Sin[x], Cos[x], Sin[x]*Cos[x]}, {x, 0, 2*Pi},
"CustomFont" -> "Charter", PlotLabel -> "Funciones Trigonometricas",
AxesLabel -> {"x (radianes)", "f(x)"},
PlotLegends -> {"sin(x)", "cos(x)", "sin(x)cos(x)"}]

```

La salida en **Mathematica** es la siguiente:



Como se aprecia, el comando **PlotLaTeXMultiple** tiene un funcionamiento similar a la sentencia nativa **Plot**. En la gráfica anterior se ha empleado la fuente *Charter*.

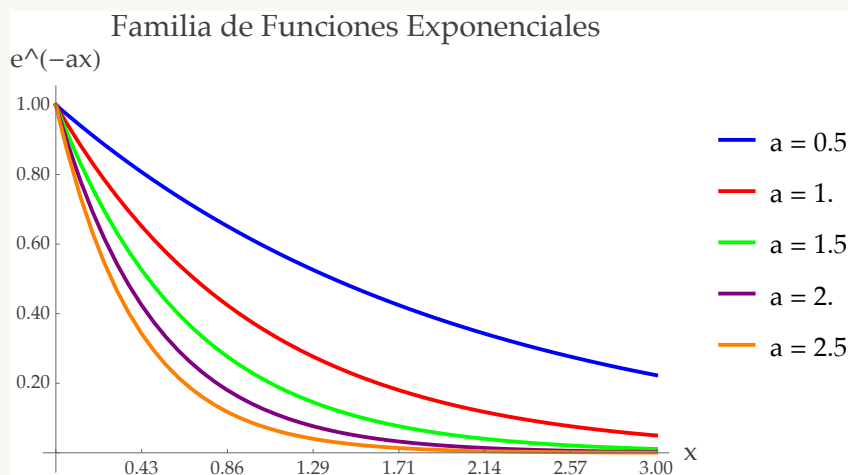
Ejemplo 33 Familia de exponenciales

```

PlotLaTeXMultiple[Table[Exp[-a*x], {a, 0.5, 2.5, 0.5}], {x, 0, 3},
"CustomFont" → "Palatino",
PlotLabel → "Familia de Funciones Exponenciales",
AxesLabel → {"x", "e^(-ax)"},
PlotLegends → Table["a = " <> ToString[a], {a, 0.5, 2.5, 0.5}]

```

Lo que genera como salida la siguiente familia de funciones exponenciales, donde los textos y números mostrados usan la tipografía *Palatino*:

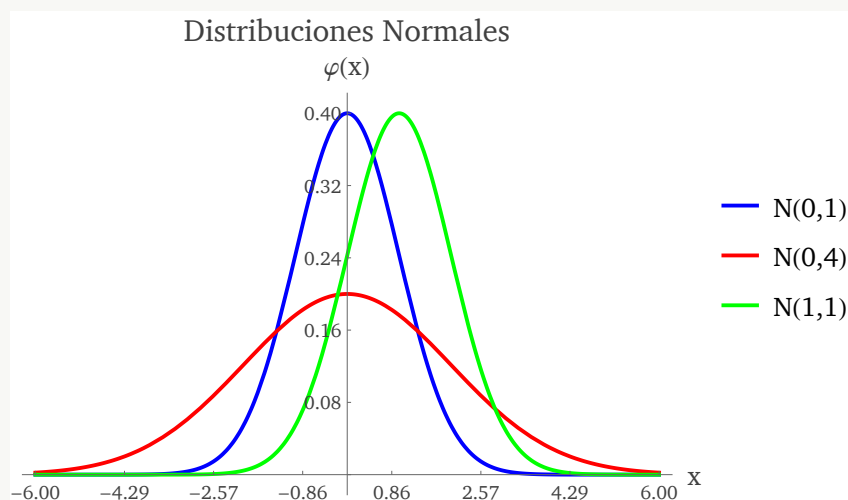
**Ejemplo 34** Distribuciones de probabilidad

```

distribuciones = {PDF[NormalDistribution[0, 1], x],
PDF[NormalDistribution[0, 2], x], PDF[NormalDistribution[1, 1], x]};
PlotLaTeXMultiple[distribuciones, {x, -6, 6},
"CustomFont" → "Charter", PlotLabel → "Distribuciones Normales",
AxesLabel → {"x", " $\varphi(x)$ "},
PlotLegends → {"N(0,1)", "N(0,4)", "N(1,1)"}]

```

La gráfica construida en **Wolfram Mathematica** presenta distintas distribuciones normales con fuente *Charter*:



Para finalizar esta sección se comparten con el lector algunos *tips* de uso de las funciones expuestas del sistema de *plotting* del paquete $\text{\LaTeX}$:

Tip 1: Guardar la configuración de fuente favorita.

```
miFuenteFavorita = "Palatino";  
PlotLaTeXStyle[Sin[x], {x, 0, 2 Pi},  
  "CustomFont" → miFuenteFavorita]
```

Tip 2: Crear función personalizada con fuente fija.

```
MiPlot[expr_, range_, opts___] :=  
  PlotLaTeXStyle[expr, range, "CustomFont" → "Charter", opts]  
MiPlot[Cos[x^2], {x, 0, 3}, PlotLabel → "Mi Plot Personalizado"]
```

Tip 3: Usar con diferentes temas de exportación

```
plot = PlotLaTeXThemed[Tan[x], {x, -Pi/3, Pi/3}, "academic",  
  "CustomFont" → "Times New Roman"];
```

Para más ejemplos

Si se desea tener acceso a otros ejemplos vinculados con el sistema de *plotting* compartido, se puede consultar de manera complementaria el archivo: *3. Ejemplos SISTEMA PLOT LaTeX.nb*, disponible en el enlace de descarga del paquete $\text{\LaTeX}$ (ver página xiv).

4

Función `ExportToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Convertir gráficos 2D de **Mathematica** en código `TikZ` vectorial editable.
- ▶ Preservar la calidad y fidelidad de visualizaciones complejas durante la exportación.
- ▶ Personalizar parámetros de exportación: colores, estilos de línea y marcadores.
- ▶ Editar código `TikZ` generado para ajustes finos en $\text{\LaTeX}$.
- ▶ Integrar gráficos vectoriales en documentos sin pérdida de resolución.

La integración efectiva entre visualizaciones computacionales y documentos $\text{\LaTeX}$ científicos ha sido históricamente un desafío técnico significativo. Mientras que **Mathematica** genera gráficos de alta calidad en formatos rasterizados (PNG, JPEG) o vectoriales propietarios (EPS, PDF), la comunidad académica frecuentemente requiere código *TikZ/PGFPlots* nativo que permita edición posterior, consistencia tipográfica con el documento, y control sobre cada elemento del gráfico.

El sistema de exportación `TikZ` de $\text{\LaTeX}$ aborda esta necesidad mediante una conversión integral que transforma objetos gráficos de **Mathematica** en código `TikZ` completamente funcional y compilable. A diferencia de herramientas de conversión externas o exportaciones manuales que requieren ajustes posteriores sustanciales, el comando `ExportToTikZ` explorado en esta sección, implementa un análisis exhaustivo de primitivas gráficas, detección automática de texto matemático, gestión inteligente de estilos, y escalado adaptativo para rangos numéricos extremos.

El núcleo del sistema reside en la capacidad de interpretar la estructura interna de objetos `Graphics` y funciones de *plotting* como `Plot`, `ListPlot`, `DiscretePlot`, `ParametricPlot`, `PolarPlot` o combinaciones mediante el comando `Show`. El proceso de conversión atraviesa múltiples etapas: primero extrae las primitivas gráficas fundamentales (`Line`, `Point`, `Arrow`, `Circle`, `Polygon`), luego identifica y procesa anotaciones textuales automáticamente, determina rangos óptimos para los ejes, aplica escalado cuando los valores numéricos superan umbrales definidos, y finalmente genera código `TikZ` estructurado con opciones de estilo consistentes.

La función `ExportToTikZ` transforma objetos gráficos nativos de **Mathematica** en código *TikZ/PGFPlots* completamente funcional, generando documentos $\text{\LaTeX}$ compilables que preservan la calidad vectorial y editabilidad del gráfico original. A diferencia de exportaciones convencionales que producen

imágenes rasterizadas (PNG, JPEG) o formatos vectoriales cerrados (PDF, EPS), **ExportToTikZ** genera código fuente $\text{\LaTeX}$ legible que puede modificarse manualmente con cualquier editor de texto.

Esta característica es particularmente valiosa en contextos académicos donde se requiere ajustar colores, etiquetas, o estilos para cumplir con lineamientos editoriales específicos después de la generación inicial. El resultado de la sentencia **ExportToTikZ** es un archivo `.tex` autónomo con clase *standalone* que puede compilarse inmediatamente mediante *pdflatex* sin requerir edición adicional. Veamos la sintaxis de esta instrucción.

- **ExportToTikZ**[*plot*, *filename*, *grid*, *styles*, *ejes*, *texto*, *xmin*, *xmax*, *ymin*, *ymax*]: Convierte un gráfico de **Mathematica** a código $\text{\LaTeX}$ y lo exporta como archivo `.tex` compilable.

Parámetros:

- **plot**: Gráfico válido de **Mathematica**. Acepta objetos **Graphics** (gráficos puros construidos con primitivas), funciones de *plotting* estándar (tales como: **Plot**, **ListPlot**, **DiscretePlot**, **ParametricPlot**, **PolarPlot**) o combinaciones mediante el comando **Show** nativo de **Wolfram**.
- **filename**: *String* con el nombre del archivo de salida. Se debe incluir la extensión `.tex` (por ejemplo: `"migrafico.tex"`). El sistema crea automáticamente una carpeta en *Downloads* con el nombre base del archivo (sin extensión) y coloca el *file .tex* dentro de esta carpeta.
- **grid** (opcional, por defecto **True**): Booleano que controla la visualización de la rejilla mayor en el gráfico. **True** muestra líneas de rejilla en los *ticks* principales de ambos ejes y **False** genera un gráfico sin rejilla.
- **styles** (opcional, por defecto `{}`): Especificación de estilos visuales para las primitivas gráficas. Acepta múltiples formatos con sintaxis flexible:
 - * *String* único: `"red"` o `"dashed"` aplica ese color/estilo a todos los elementos.
 - * Par ordenado: `{"dashed", "blue"}` o `{"blue", "dashed"}` (orden indistinto).
 - * Lista de colores: `{"red", "blue", "green"}` asigna colores secuencialmente con estilo **solid**.
 - * Lista de estilos: `{"solid", "dashed", "dotted"}` asigna estilos con color **black**.
 - * Matriz completa: `{{"solid", "red"}, {"dashed", "blue"}, {"dotted", "green"}}` especifica estilo y color para cada primitiva.

Colores válidos: red, blue, green, orange, purple, black, brown, pink, gray, cyan, magenta, yellow, lime, olive y teal.

Estilos válidos: solid, dashed, dotted, dashdotted y thick.

- **ejes** (opcional, por defecto **True**): Booleano que controla la visualización de los ejes coordenados. **True** muestra ejes centrados en el origen (`axis lines=center`) y **False** oculta completamente los ejes (`axis lines=none`).
- **texto** (opcional, por defecto `{}`): Anotaciones de texto adicionales que se añaden al gráfico. Acepta tres formatos:
 - * Lista vacía `{}`: Sin anotaciones adicionales (pero se extraen textos del gráfico original).
 - * Anotación única: `{"texto", {x, y}}` coloca "texto" en las coordenadas (*x*, *y*).
 - * Múltiples anotaciones: `{{"texto1", {x1, y1}}, {"texto2", {x2, y2}}, ...}` añade varias etiquetas.

Nota: El sistema extrae automáticamente todos los textos existentes en el gráfico original mediante análisis de objetos **Text**, por lo que este parámetro es opcional y se usa principalmente para añadir anotaciones adicionales no presentes en el gráfico inicial.

- `xmin`, `xmax`, `ymin`, `ymax` (opcionales, por defecto **Automatic**): Límites explícitos de los ejes coordenados. **Automatic** calcula los límites automáticamente basados en los datos con márgenes del 5%-10%. Cuando se especifican manualmente, deben ser números válidos finitos con `xmin < xmax` y `ymin < ymax`.

Retorna en Mathematica:

Ruta completa del archivo `.tex` exportado en el directorio *Downloads*.

Funcionalidades especiales:

- Extracción automática de textos del gráfico original sin intervención del usuario.
- Ajuste inteligente de posición de etiquetas para evitar solapamiento con puntos de datos.
- Detección automática de expresiones matemáticas y formato apropiado (delimitadores `$...$`).
- Conversión automática de símbolos *Unicode*/**Mathematica** a comandos $\text{\LaTeX}$ (por ejemplo: $\Pi \rightarrow \backslash\text{pi}$).
- Escalado automático de ejes para valores grandes (> 1000) con etiquetas $\times 10^n$.
- Manejo robusto de múltiples tipos de primitivas gráficas (líneas, puntos, flechas, círculos, polígonos).
- Creación automática de carpeta de descarga en *Downloads*.
- Generación de documento $\text{\LaTeX}$ completo y compilable con clase *standalone*.
- Validación exhaustiva de parámetros con mensajes de error descriptivos.

4.1 Ejemplos de uso de la función ExportToTikZ

Los ejemplos presentados en esta sección ilustran la versatilidad de la sentencia **ExportToTikZ** mediante catorce casos representativos que abarcan el espectro completo de objetos gráficos soportados por **Wolfram Mathematica**. A diferencia de tutoriales convencionales que se limitan a funciones básicas, esta colección incluye desde gráficos simples de **Plot** hasta composiciones complejas a través del uso de **Show**, construcciones geométricas elaboradas con **Graphics**, curvas polares y paramétricas, datos discretos, y visualizaciones combinadas de sistemas físicos.

Cada ejemplo demuestra aspectos específicos del sistema: extracción automática de textos y anotaciones matemáticas, aplicación de matrices de estilos para múltiples elementos, manejo de primitivas geométricas diversas (círculos, polígonos, flechas), control preciso de límites de ejes, y trato de opciones de visualización (rejilla, ejes, colores). La progresión abarca desde configuraciones predeterminadas que requieren mínima especificación de parámetros hasta casos avanzados con control sobre cada aspecto del gráfico resultante. Una característica distintiva de estos ejemplos es la integración inmediata con el sistema compilador de **ViTeX** mediante el comando **CompiladorTex** (explicado en el capítulo 2). Cada exportación concluye con la compilación automática del archivo `.tex` generado, produciendo el PDF final sin intervención manual.

Esta integración completa -desde el cálculo computacional hasta el documento profesional- ejemplifica el objetivo central de **ViTeX**: eliminar discontinuidades en el flujo de elaboración de un documento científico. Los ejemplos están diseñados para ser autocontenidos y ejecutables directamente. Se recomienda experimentar modificando parámetros individuales (colores, estilos, límites, anotaciones) para comprender el efecto de cada opción sobre el código `TikZ` generado. Los archivos `.tex` resultantes, ubicados automáticamente en *Downloads*, pueden abrirse con cualquier editor de texto para examinar el código `TikZ` producido y realizar ajustes manuales posteriores cuando sea necesario.

Nota: Los siguientes ejemplos asumen que el paquete **ViTeX** ha sido cargado previamente mediante `Get["ViTeX"]` y que el sistema operativo cuenta con *TeXstudio* u otro editor $\text{\LaTeX}$ configurado correctamente.

Ejemplo 35 Gráfica *Plot* con uso básico y empleo de textos

```

ExportToTikZ1 =
Plot[Sin[x], {x, 0, 2 Pi}, PlotLabel → "Funcion Seno",
AxesLabel → {"Angulo θ", "Amplitud"},
Epilog → {Text["Maximo", {Pi/2, 1.1}],
Text["Minimo", {3 Pi/2, -1.1}],
Text["f(x) = sin(x)", {Pi, 0.5}]]]
ExportToTikZ[ExportToTikZ1, "ExportToTikZ1.tex"];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Mathematica** se muestra como salida la gráfica de la subfigura 4.1a y los mensajes de proceso:

Textos extraídos automáticamente: 3 elementos

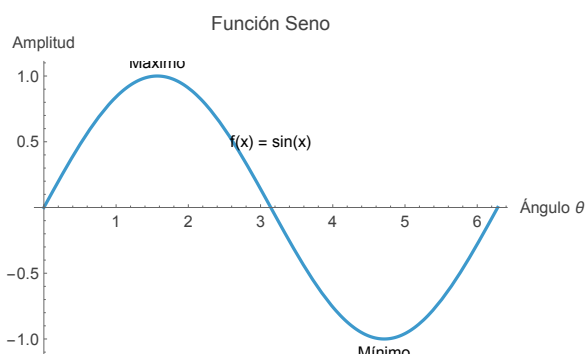
Usando editores ya detectados: 4 disponibles

:

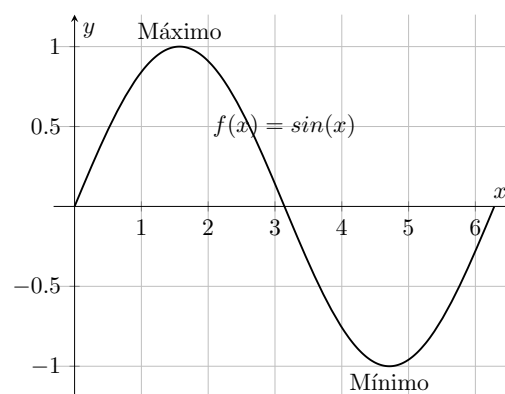
PDF generado: /Users/.../Downloads/ExportToTikZ1/ExportToTikZ1.pdf

/Users/.../Downloads/ExportToTikZ1/ExportToTikZ1.pdf

Además, se abre en el editor *TeXstudio* el archivo *ExportToTikZ1.tex* (guardado en *Downloads/ ExportToTikZ1*). Este documento *.tex* contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 4.1b. Como se aprecia el comando **ExportToTikZ** tiene la capacidad de recibir el **Plot** de **Wolfram** y automáticamente transformarlo en el archivo compilable *ExportToTikZ1.tex*.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.1: Gráficas del ejemplo 35

Ejemplo 36 Gráfica *Plot* compleja con título

```

ExportToTikZ2 =
Plot[Exp[-x/6]*(BesselJ[0, 9 x] + 0.30*Sin[7 x]/(1 + 0.20 x) +
0.15*Cos[Pi x^2]), {x, 0, 18}, PlotRange → All,
Exclusions → None, PlotPoints → 300, MaxRecursion → 5,
PerformanceGoal → "Quality", AxesLabel → {"x", "f(x)"},
PlotLabel → "Signal amortiguada con oscilaciones no uniformes",
ImageSize → 540]
ExportToTikZ[ExportToTikZ2, "ExportToTikZ2.tex", True, "green",
True, {"Signal amortiguada con", {8, 0.6}}, {"oscilaciones no uniformes",

```



```
{8, 0.5}}];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** la salida despliega lo compartido en la subfigura 4.2a y los mensajes de texto:

Textos extraídos automáticamente: 2 elementos

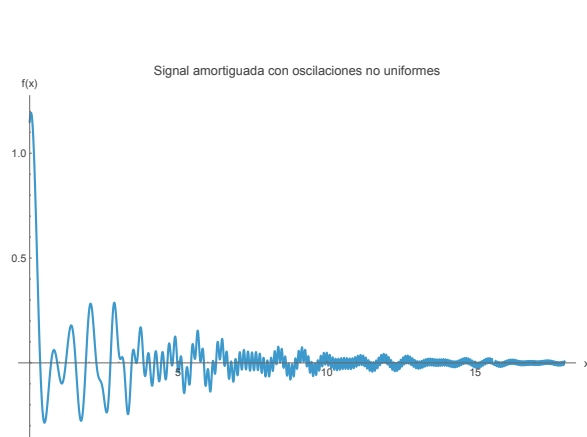
Usando editores ya detectados: 4 disponibles

⋮

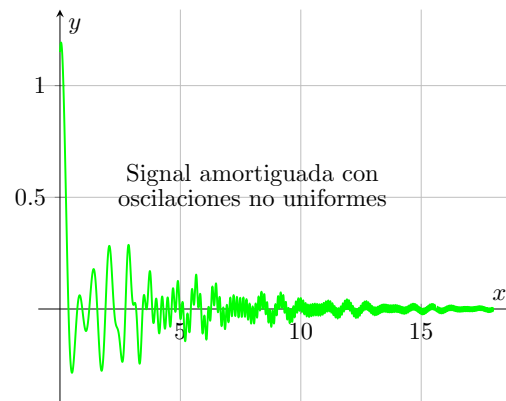
PDF generado: /Users/.../Downloads/ExportToTikZ2/ExportToTikZ2.pdf

/Users/.../Downloads/ExportToTikZ2/ExportToTikZ2.pdf

El archivo *ExportToTikZ2.tex* (guardado en *Downloads/ExportToTikZ2*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 4.2b. Se aclara al lector que por la complejidad de la gráfica $\text{\TikZ}$ de este ejercicio, el proceso de compilación en $\text{\LaTeX}$ tarda varios segundos.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 4.2: Gráficas del ejemplo 36

Ejemplo 37 Gráfica *Plot* con empleo de *Show*

```
ExportToTikZ3 =
Show[Plot[Sin[x], {x, 0, 2*Pi}, PlotStyle → {Red}],
Plot[Cos[x], {x, 0, 2*Pi}, PlotStyle → {Directive[Blue, Dashed]}]]
ExportToTikZ[ExportToTikZ3, "ExportToTikZ3.tex",
True, {"solid", "red"}, {"dotted", "blue"}];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** se muestra como salida la gráfica de la subfigura 4.3a y los mensajes:

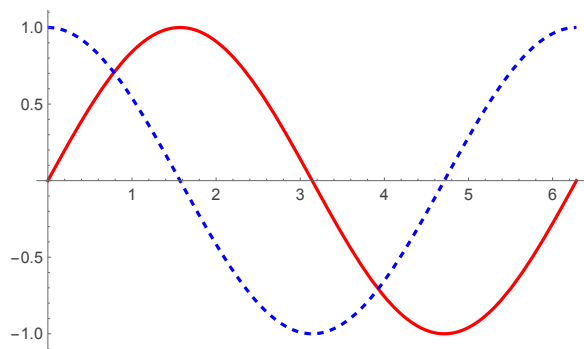
Usando editores ya detectados: 4 disponibles

⋮

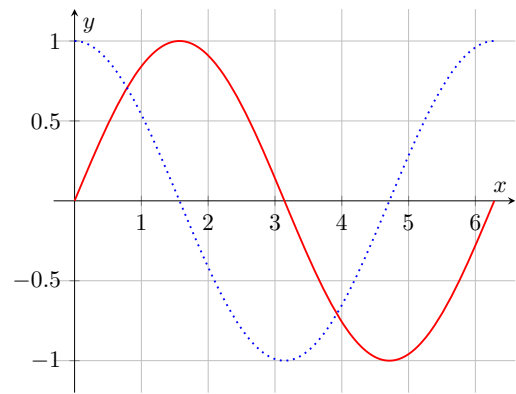
PDF generado: /Users/.../Downloads/ExportToTikZ3/ExportToTikZ3.pdf

/Users/.../Downloads/ExportToTikZ3/ExportToTikZ3.pdf

El archivo *ExportToTikZ3.tex* (guardado en *Downloads/ExportToTikZ3*) incluye el código $\text{\LaTeX}$ editable que genera la gráfica $\text{\TikZ}$ de la subfigura 4.3b.

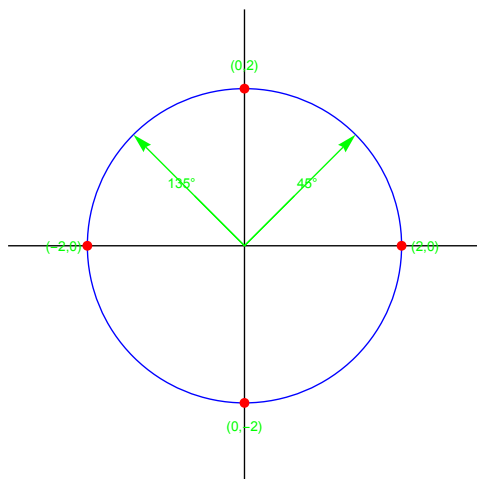


(a) Gráfica de **Mathematica**

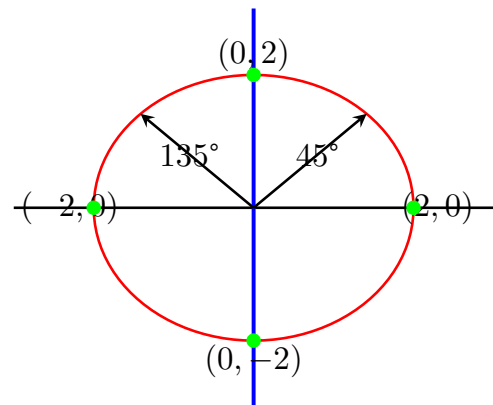


(b) Gráfica TikZ

Figura 4.3: Gráficas del ejemplo 37

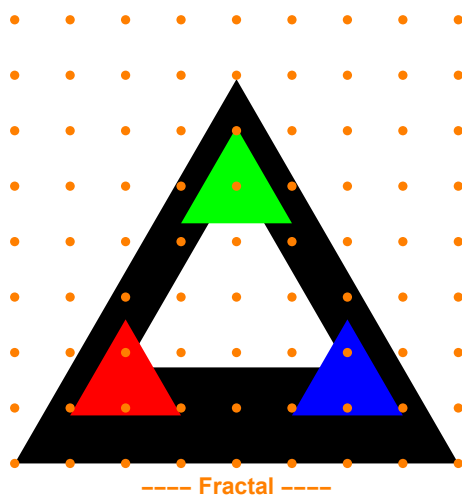


(a) Gráfica de **Mathematica**

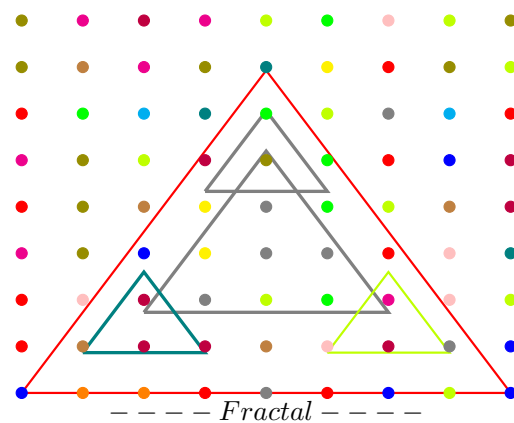


(b) Gráfica TikZ

Figura 4.4: Gráficas del ejemplo 38



(a) Gráfica de **Mathematica**



(b) Gráfica TikZ

Figura 4.5: Gráficas del ejemplo 39

Ejemplo 38 Gráfica *Graphics* con diagrama complejo

```

ExportToTikZ4 =
Graphics[{Black, Line[{{-3, 0}, {3, 0}}], Line[{{0, -3}, {0, 3}}],
  Blue, Circle[{0, 0}, 2], Red, PointSize[0.02],
  Point[{{2, 0}, {0, 2}, {-2, 0}, {0, -2}}], Green,
  Arrow[{{0, 0}, {Sqrt[2], Sqrt[2]}],
  Arrow[{{0, 0}, {-Sqrt[2], Sqrt[2]}], Text["(2,0)", {2.3, 0}],
  Text["(0,2)", {0, 2.3}], Text["(-2,0)", {-2.3, 0}],
  Text["(0,-2)", {0, -2.3}], Text["45°", {0.8, 0.8}],
  Text["135°", {-0.8, 0.8}]]
ExportToTikZ[ExportToTikZ4, "ExportToTikZ4.tex",
True, {{"solid", "black"}, {"thick", "blue"}, {"solid",
"red"}, {"dashed", "green"}}, False, {}, -4, 4, -4, 4];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram** la salida despliega lo compartido en la subfigura 4.4a y los mensajes de proceso:

Textos extraídos automáticamente: 6 elementos

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ4/ExportToTikZ4.pdf

/Users/.../Downloads/ExportToTikZ4/ExportToTikZ4.pdf

El archivo *ExportToTikZ4.tex* (guardado en *Downloads/ExportToTikZ4*) contiene el código $\text{\LaTeX}$ editable que construye la gráfica $\text{\TikZ}$ de la subfigura 4.4b.

Ejemplo 39 Gráfica *Graphics* triángulo de *Sierpinski* básico

```

ExportToTikZ5 =
Graphics[{Black, Thick, Polygon[{{0, 0}, {4, 0}, {2, 2*Sqrt[3]}]},
  White, Polygon[{{1, Sqrt[3]/2}, {3, Sqrt[3]/2}, {2, 3*Sqrt[3]/2}}],
  Red, Polygon[{{0.5, Sqrt[3]/4}, {1.5, Sqrt[3]/4}, {1, 3*Sqrt[3]/4}}],
  Blue, Polygon[{{2.5, Sqrt[3]/4}, {3.5, Sqrt[3]/4}, {3,
3*Sqrt[3]/4}}], Green,
  Polygon[{{1.5, 5*Sqrt[3]/4}, {2.5, 5*Sqrt[3]/4}, {2, 7*Sqrt[3]/4}}],
  Orange, PointSize[0.02],
  Table[Point[{i, j}], {i, 0, 4, 0.5}, {j, 0, 4, 0.5}],
  Text[Style["---- Fractal ----", 18, Bold], {2, -0.2}]]
ExportToTikZ[ExportToTikZ5, "ExportToTikZ5.tex", False,
Table[{RandomChoice[{"solid", "thick"}],
  RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
    "pink", "gray", "cyan", "magenta", "yellow", "lime", "olive",
    "teal"}]}, 100], False];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

El **Table** del código anterior genera una matriz de estilos 100×2 con colores pseudoaleatorios para todas las primitivas de la gráfica. En **Mathematica** esto muestra como salida lo compartido en la subfigura 4.5a y los mensajes de texto:

Textos extraídos automáticamente: 2 elementos

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ5/ExportToTikZ5.pdf

/Users/.../Downloads/ExportToTikZ5/ExportToTikZ5.pdf

El archivo *ExportToTikZ5.tex* (guardado en *Downloads/ExportToTikZ5*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 4.5b. Como se aprecia, la gráfica $\text{\TikZ}$ devuelta por el comando `ExportToTikZ` no es idéntica a la obtenida mediante **Mathematica**, pese a ello, la aproximación retornada en código $\text{\LaTeX}$ es bastante buena tomando en consideración la complejidad del objeto `Graphics`.

Ejemplo 40 Gráfica *Graphics* mandala geométrico islámico

```
ExportToTikZ6 =
Graphics[{{Blue, Opacity[0.8],
  Table[Polygon[{{0, 0}, 1.2*{Cos[2 Pi*k/8], Sin[2 Pi*k/8]},
    0.8*{Cos[2 Pi*k/8 + Pi/8],
    Sin[2 Pi*k/8 + Pi/8]}]], {k, 0, 7}}, {Yellow,
  Disk[{0, 0}, 0.3]},
  Table[Module[{centro =
    2*{Cos[2 Pi*k/8], Sin[2 Pi*k/8]}}, {Green, Opacity[0.7],
    Polygon[Table[
      centro + 0.6*{Cos[2 Pi*j/8], Sin[2 Pi*j/8]}, {j, 0,
      7}]]], {k, 0, 7}],
  Table[{{Purple, Opacity[0.6],
    Table[Polygon[{2.8*{Cos[2 Pi*k/16], Sin[2 Pi*k/16]},
      3.4*{Cos[2 Pi*k/16 + Pi/32],
      Sin[2 Pi*k/16 + Pi/32]},
      3.4*{Cos[2 Pi*k/16 - Pi/32],
      Sin[2 Pi*k/16 - Pi/32]}]], {k, 0, 15}}, 1],
    Table[Module[{centro =
      3.2*{Cos[Pi/8 + 2 Pi*k/8],
      Sin[Pi/8 + 2 Pi*k/8]}}, {Red, Thick,
      Line[{centro + {-0.3, 0}, centro + {0.3, 0}},
      Line[{centro + {0, -0.3}, centro + {0, 0.3}}]], {k, 0, 7}],
    Table[{{Cyan, Thick,
      Table[Line[{4*{Cos[2 Pi*j/12], Sin[2 Pi*j/12]},
        4*{Cos[2 Pi*(j + k)/12], Sin[2 Pi*(j + k)/12]}]], {j, 0,
        11}}, {k, {2, 4, 6}}],
    Table[{{Orange, Thick,
      Line[Table[{3.6*Cos[t + 2 Pi*k/6],
        3.6*Sin[t + 2 Pi*k/6]}, {t, 0, Pi/3, Pi/30}]]}, {k,
        0, 5}], Table[{{Pink, Opacity[0.5],
      Polygon[Table[
        2.5*{Cos[2 Pi*j/6 + 2 Pi*k/3],
        Sin[2 Pi*j/6 + 2 Pi*k/3]}, {j, 0, 5}]]}, {k, 0, 2}],
    Table[{{Black, Thick, Circle[{0, 0}, 4.5 + i*0.1]}, {i, 0, 3}],
    Table[Module[{pos =
      4.7*{Cos[2 Pi*k/24], Sin[2 Pi*k/24]}}, {Purple,
      Polygon[Table[
        pos + 0.2*{Cos[2 Pi*j/4], Sin[2 Pi*j/4]}, {j, 0,
        3}]]], {k, 0, 23}],
    Table[{{White, Thick, Circle[{0, 0}, r]}, {r, {1.5, 2.5, 3.5, 4.2}}],
    Text[Style["MANDALA ISLAMICO", 18, Bold, Blue], {0, -6}],
    Text[Style["~ Geometria de la Unidad ~", 14, Italic,
    Green], {0, -6.7}]]]
ExportToTikZ[ExportToTikZ6, "ExportToTikZ6.tex", False,
Table[{RandomChoice[{"solid", "thick"}],
  RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
    "pink", "gray", "cyan", "magenta", "yellow", "lime", "olive",
    "teal"}]], 150], False, {}, -6, 6, -8, 5];
```

```
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram** se genera como salida lo mostrado en la subfigura 4.6a y los mensajes:

Textos extraídos automáticamente: 2 elementos

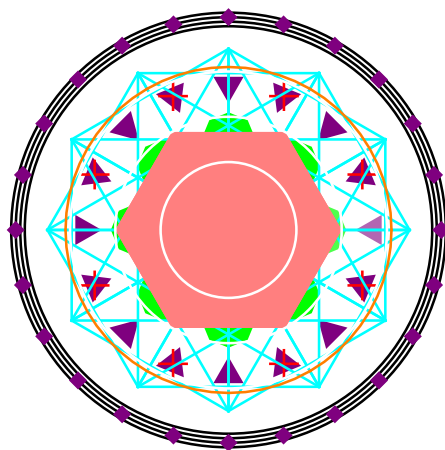
Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ6/ExportToTikZ6.pdf

/Users/.../Downloads/ExportToTikZ6/ExportToTikZ6.pdf

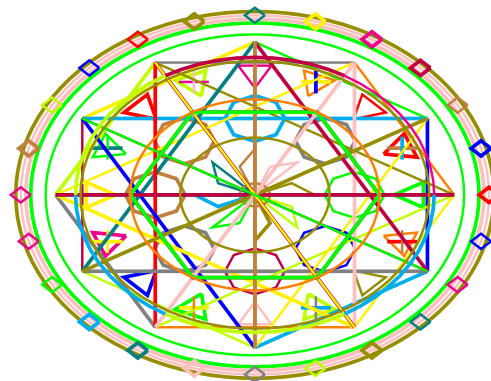
El file *ExportToTikZ6.tex* (guardado en *Downloads/ExportToTikZ6*) posee el código $\text{\LaTeX}$ editable que crea la gráfica $\text{\TikZ}$ de la subfigura 4.6b. Aquí también el gráfico de **Wolfram Mathematica** no se ve exactamente igual al gráfico $\text{\TikZ}$ retornado por **ExportToTikZ** dada la complejidad subyacente, pese a ello, la salida es bastante aceptable.



MANDALA ISLAMICO

~ Geometría de la Unidad ~

(a) Gráfica de **Mathematica**



MANDALA ISLAMICO
~ Geometría de la Unidad ~

(b) Gráfica $\text{\TikZ}$

Figura 4.6: Gráficas del ejemplo 40

Ejemplo 41 Gráfica *Graphics* con dibujo de gato

```
ExportToTikZ7 =
Graphics[{{LightBrown, Disk[{0, 0}, 1.5]}, {LightBrown,
  Disk[{0, 2}, 1]}, {LightBrown,
  Polygon[{{-0.6, 2.5}, {-1.1, 3.3}, {-0.2, 2.8}}]}, {LightBrown,
  Polygon[{{0.6, 2.5}, {1.1, 3.3}, {0.2, 2.8}}]}, {White,
  Disk[{-0.4, 2.1}, 0.2], Disk[{0.4, 2.1}, 0.2]}, {Black,
  Disk[{-0.4, 2.05}, 0.1], Disk[{0.4, 2.05}, 0.1]}, {Pink,
  Disk[{0, 1.8}, 0.12]}, {Black, Line[{0, 1.7}, {-0.2, 1.5}]},
  Line[{0, 1.7}, {0.2, 1.5}]}, {Gray,
  Line[{{-0.2, 1.8}, {-1.0, 1.9}}], Line[{{-0.2, 1.7}, {-1.0, 1.6}}]},
  Line[{{0.2, 1.8}, {1.0, 1.9}}], Line[{{0.2, 1.7}, {1.0, 1.6}}]}],
PlotRange → {{-2, 2}, {-1, 4}}, Background → LightYellow,
ImageSize → 300]
ExportToTikZ[ExportToTikZ7, "ExportToTikZ7.tex", False,
Table[{RandomChoice[{"solid", "dashed", "dotted", "dashdotted",
"thick"}], RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
```

```
"pink", "gray", "cyan", "magenta", "yellow", "lime", "olive",
"teal"}]], 50], False];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Wolfram Mathematica** la salida del código anterior construye lo compartido en la subfigura 4.7a y además, los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

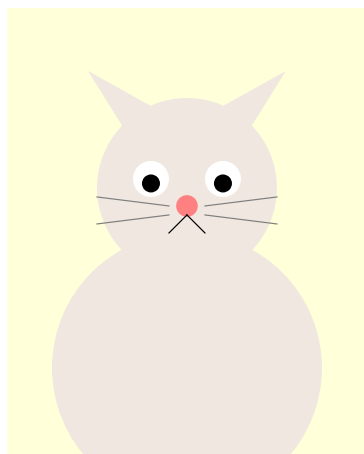
⋮

PDF generado: /Users/.../Downloads/ExportToTikZ7/ExportToTikZ7.pdf

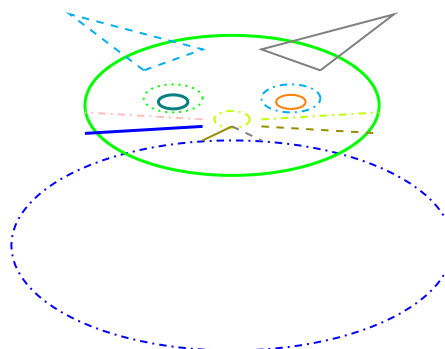
/Users/.../Downloads/ExportToTikZ7/ExportToTikZ7.pdf

Aquí, nuevamente los estilos aplicados a la figura son pseudoaleatorios (esto por medio del **Table**). El archivo *ExportToTikZ7.tex* (guardado en *Downloads/ExportToTikZ7*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 4.7b.

El dibujo de gato propuesto por el comando **ExportToTikZ** no es exactamente igual al devuelto por **Wolfram**, pese a ello, como ya se mencionó en otros ejemplos, al ser un proceso automático la salida es considerablemente buena.

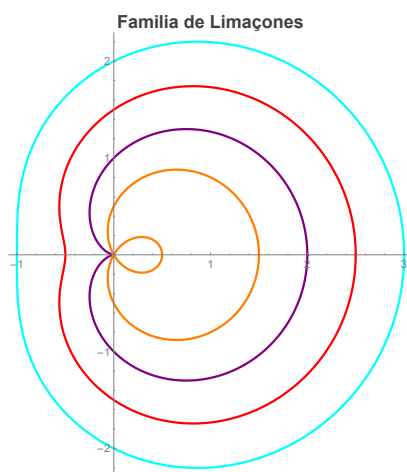


(a) Gráfica de **Mathematica**

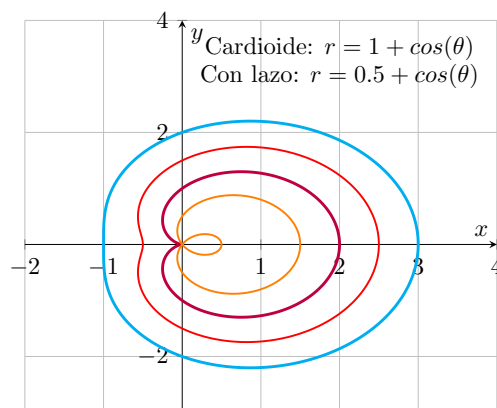


(b) Gráfica $\text{\LaTeX}$

Figura 4.7: Gráficas del ejemplo 41



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.8: Gráficas del ejemplo 42

Ejemplo 42 Gráfica *PolarPlot* con una familia de limaçonnes

```

ExportToTikZ8 =
PolarPlot[{1 + Cos[θ], 0.5 + Cos[θ], 2 + Cos[θ],
1.5 + Cos[θ]}, {θ, 0, 2 Pi},
PlotStyle → {{Thick, Purple}, {Thick, Orange}, {Thick,
Cyan}, {Thick, Red}}, PlotRange → All, Background → White,
Axes → True, AxesStyle → Gray,
PlotLabel → Style["Familia de Limaçonnes", 16, Bold]]
ExportToTikZ[ExportToTikZ8, "ExportToTikZ8.tex",
True, {"thick", "purple"}, {"solid", "orange"}, {"thick",
"cyan"}, {"solid", "red"}],
True, {"Cardioides: r=1+cos(\\theta)", {2,
3.5}}, {"Con lazo: r=0.5+cos(\\theta)", {2, 3}}, -2,
4, -3, 4];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Mathematica** lo anterior genera como salida lo mostrado en la subfigura 4.8a y los mensajes:

Textos extraídos automáticamente: 2 elementos

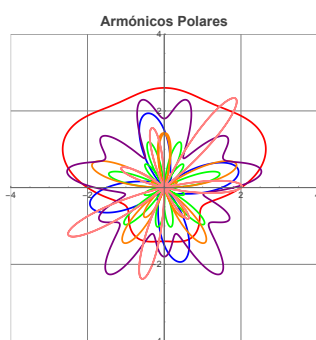
Usando editores ya detectados: 4 disponibles

⋮

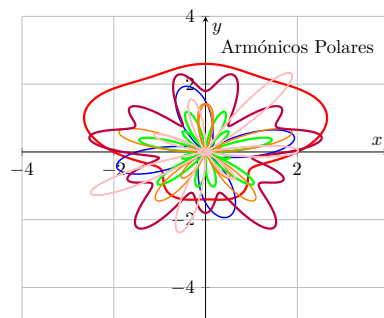
PDF generado: /Users/.../Downloads/ExportToTikZ8/ExportToTikZ8.pdf

/Users/.../Downloads/ExportToTikZ8/ExportToTikZ8.pdf

El archivo *ExportToTikZ8.tex* (guardado en *Downloads/ExportToTikZ8*) ofrece el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 4.8b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.9: Gráficas del ejemplo 43

Ejemplo 43 Gráfica *PolarPlot* con curvas armónicas sin singularidades

```

ExportToTikZ9 =
PolarPlot[{2 + 0.8*Sin[θ] + 0.4*Sin[3*θ] +
0.2*Sin[5*θ], 1.5*(1 + 0.6*Sin[4*θ])*Cos[2*θ],
Sin[6*θ] + 0.5*Sin[7*θ],
2*Sin[5*θ]*Sin[θ/2],
1.8 + 0.9*Cos[3*θ]*Sin[8*θ], (2 + Sin[2*θ])*
Cos[7*θ]}, {θ, 0, 2 Pi},
PlotStyle → {{Thick, Red}, {Thick, Blue}, {Thick, Green}, {Thick,

```

```

Orange}, {Thick, Purple}, {Thick, Pink}},
PlotRange → {{-4, 4}, {-4, 4}}, Background → White, Axes → True,
GridLines → Automatic,
PlotLabel → Style["Armónicos Polares", 16, Bold]
ExportToTikZ[ExportToTikZ9, "ExportToTikZ9.tex",
True, {"thick", "red"}, {"solid", "blue"}, {"thick",
"green"}, {"solid", "orange"}, {"thick", "purple"}, {"solid",
"pink"}], True, {"Armónicos Polares", {2, 3.1}}, -4, 4, -5, 4];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram** se despliega como salida lo compartido en la subfigura 4.9a y los mensajes de texto:

Textos extraídos automáticamente: 1 elementos

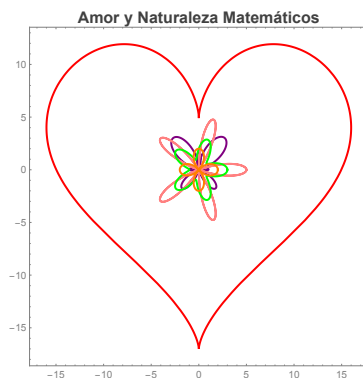
Usando editores ya detectados: 4 disponibles

:

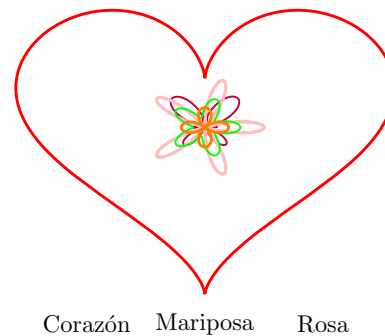
PDF generado: /Users/.../Downloads/ExportToTikZ9/ExportToTikZ9.pdf

/Users/.../Downloads/ExportToTikZ9/ExportToTikZ9.pdf

El archivo *ExportToTikZ9.tex* (guardado en *Downloads/ExportToTikZ9*) proporciona el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 4.9b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 4.10: Gráficas del ejemplo 44

Ejemplo 44 Gráfica *ParametricPlot* con corazón, mariposa, rosa, flor de loto y trébol

```

ExportToTikZ10 =
ParametricPlot[{{16*Sin[t]^3,
13*Cos[t] - 5*Cos[2*t] - 2*Cos[3*t] -
Cos[4*t]}, {Sin[t]*(E^Cos[t] - 2*Cos[4*t] - Sin[t/12]^5),
Cos[t]*(E^Cos[t] - 2*Cos[4*t] - Sin[t/12]^5)}, {5*Cos[5*t]*Cos[t],
5*Cos[5*t]*Sin[t]}, {(2 + Cos[5*t])*Cos[t], (2 + Cos[5*t])*
Sin[t]}, {2*Cos[2*t]*Cos[t], 2*Cos[2*t]*Sin[t]}}, {t, 0, 2 Pi},
PlotStyle → {{Thick, Red}, {Thick, Purple}, {Thick, Pink}, {Thick,
Green}, {Thick, Orange}}, PlotRange → All, Axes → False,
Frame → True, FrameStyle → Gray,
PlotLabel → Style["Amor y Naturaleza Matemáticos", 16, Bold],
AspectRatio → 1]
ExportToTikZ[ExportToTikZ10, "ExportToTikZ10.tex",
False, {"thick", "red"}, {"solid", "purple"}, {"thick",
"pink"}, {"solid", "green"}, {"thick", "orange"}],

```



```
False, {{"Corazón", {-10, -20}}, {"Mariposa", {0, -20}}, {"Rosa",
{10, -20}}}, -20, 20, -25, 15];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]]
```

En **Wolfram Mathematica** la salida desplegada genera la subfigura 4.10a y los mensajes:

Textos extraídos automáticamente: 3 elementos

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportToTikZ10/ExportToTikZ10.pdf

/Users/.../Downloads/ExportToTikZ10/ExportToTikZ10.pdf

El archivo *ExportToTikZ10.tex* (guardado en *Downloads/ExportToTikZ10*) contiene el código $\text{\LaTeX}$ editable que construye la gráfica $\text{\LaTeX}$ de la subfigura 4.10b.

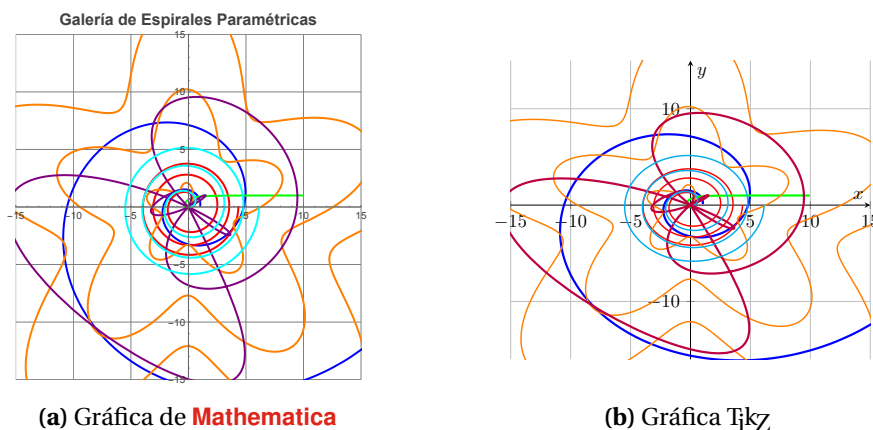


Figura 4.11: Gráficas del ejemplo 45

Ejemplo 45 Gráfica *ParametricPlot* espiral de Fermat, de Arquímedes, de Ulam y otras

```
ExportToTikZ11 =
ParametricPlot[{{E^(t/4)*Cos[t], E^(t/4)*Sin[t]}, {Sqrt[t]*Cos[t],
Sqrt[t]*Sin[t]}, {Cos[t]/t, Sin[t]/t}, {t*Cos[t]*(1 + 0.3*Sin[5*t]),
t*Sin[t]*(1 + 0.3*Sin[5*t])}, {t*Cos[t]*Sin[t/2],
t*Sin[t]*Cos[t/3]}, {t^(1/GoldenRatio)*Cos[t],
t^(1/GoldenRatio)*Sin[t]}], {t, 0.1, 6 Pi},
PlotStyle ->{{Thick, Blue}, {Thick, Red}, {Thick, Green}, {Thick,
Orange}, {Thick, Purple}, {Thick, Cyan}},
PlotRange ->{{-15, 15}, {-15, 15}}, Axes ->True,
GridLines ->Automatic,
PlotLabel ->Style["Galería de Espirales Paramétricas", 16, Bold],
AspectRatio ->1]
ExportToTikZ[ExportToTikZ11, "ExportToTikZ11.tex",
True, {"thick", "blue"}, {"solid", "red"}, {"thick",
"green"}, {"solid", "orange"}, {"thick", "purple"}, {"solid",
"cyan"}], True, {}, -15, 15, -16, 15];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]]
```

En **Mathematica** se obtiene como salida lo mostrado en la subfigura 4.11a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportToTikZ11/ExportToTikZ11.pdf
/Users/.../Downloads/ExportToTikZ11/ExportToTikZ11.pdf

El archivo *ExportToTikZ11.tex* (guardado en *Downloads/ExportToTikZ11*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 4.11b.

Ejemplo 46 Gráfica *DiscretePlot* con datos discretos, histograma y serie de tiempo

```
ExportToTikZ12 =
DiscretePlot[{Round[10*E^(-(n - 10)^2/20)) + RandomReal[{-1, 1}]],
10 + Accumulate[RandomReal[{-1, 1}, 20]][[Min[n, 20]]], {n, 1, 20},
PlotStyle ->{{Purple, PointSize[0.015]}, {Cyan, PointSize[0.02]}},
PlotRange ->All, Frame ->True, GridLines ->Automatic,
GridLinesStyle ->LightGray,
PlotLabel ->Style["Análisis de Datos Discretos", 16, Bold],
FrameLabel ->{"Tiempo/Índice", "Valor"}]
ExportToTikZ[ExportToTikZ12, "ExportToTikZ12.tex",
True, {{"solid", "red"}, {"thick", "blue"}},
True, {{"Histograma", {10, 5}}, {"Serie Tiempo", {13, 15}}}, 0,
21, -5, 20];
CompiladorTex[%, "PreferredEditor" ->"TeXstudio"]
```

En **Wolfram** la salida brinda lo compartido en la subfigura 4.12a y los mensajes:

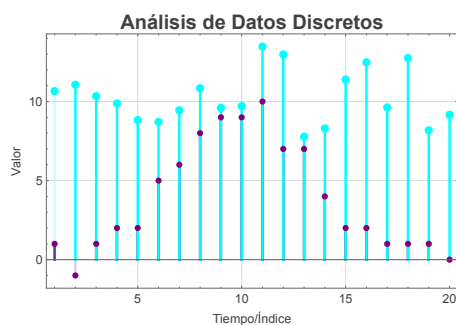
Textos extraídos automáticamente: 2 elementos

Usando editores ya detectados: 4 disponibles

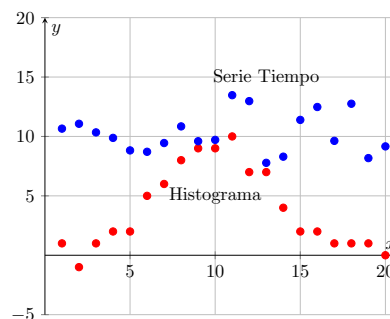
⋮

PDF generado: /Users/.../Downloads/ExportToTikZ12/ExportToTikZ12.pdf
/Users/.../Downloads/ExportToTikZ12/ExportToTikZ12.pdf

El archivo *ExportToTikZ12.tex* (guardado en *Downloads/ExportToTikZ12*) contiene el código $\text{\LaTeX}$ editable que crea la gráfica $\text{\LaTeX}$ de la subfigura 4.12b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.12: Gráficas del ejemplo 46

Ejemplo 47 Gráfica *ListPlot* que hereda los textos

```

ExportToTikZ13 =
ListPlot[{{1, 2}, {2, 3}, {3, 5}, {4, 4}},
PlotStyle → {Blue, PointSize[Large]}, PlotRange → {0, 6},
AxesOrigin → True,
Epilog → {Text[Style["A", 14, Bold, Red], {1, 2}, {1, -1}],
Text[Style["B", 14, Bold, Red], {2, 3}, {1, -1}],
Text[Style["C", 14, Bold, Red], {3, 5}, {1, -1}],
Text[Style["D", 14, Bold, Red], {4, 4}, {1, -1}],
Text[Style["Datos experimentales", 14, Bold, Darker@Green], {2.5,
5.5}]}, AxesLabel → {"x", "y"}, ImageSize → 400]
ExportToTikZ[ExportToTikZ13, "ExportToTikZ13.tex", True, "black",
True, {"Puntos en el plano", {0.5, 0.5}}, -1, 5, -1, 6];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram Mathematica** se obtiene como salida lo mostrado en la subfigura 4.13a y los mensajes de proceso:

Textos extraídos automáticamente: 6 elementos

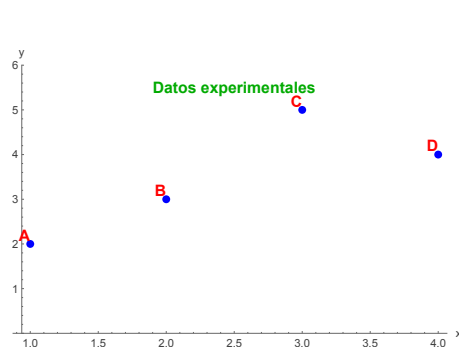
Usando editores ya detectados: 4 disponibles

⋮

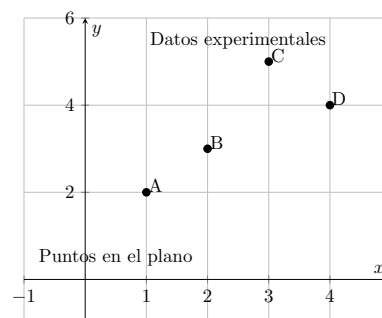
PDF generado: /Users/.../Downloads/ExportToTikZ13/ExportToTikZ13.pdf

/Users/.../Downloads/ExportToTikZ13/ExportToTikZ13.pdf

El file *ExportToTikZ13.tex* (guardado en *Downloads/ExportToTikZ13*) facilita el código $\text{\LaTeX}$ editable que genera la gráfica $\text{\LaTeX}$ de la subfigura 4.13b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.13: Gráficas del ejemplo 47

Ejemplo 48 Gráfica combinada, sistema físico

```

ExportToTikZ14 =
Show[Plot[{2*E^(-0.1*t)*Cos[Sqrt[3]*t], E^(-0.05*t)*Sin[2*t]}, {t, 0,
20}, PlotStyle → {{Thick, Blue}, {Thick, Red}}],
ParametricPlot[{2*E^(-0.1*t)*
Cos[Sqrt[3]*t], -2*E^(-0.1*t)*Sqrt[3]*Sin[Sqrt[3]*t] -
0.2*E^(-0.1*t)*Cos[Sqrt[3]*t]}, {t, 0, 15},
PlotStyle → {Green, Thick}],
DiscretePlot[{2*E^(-0.1*n)*Cos[Sqrt[3]*n],

```

```

E^(-0.05*n)*Sin[2*n]], {n, 0, 20},
PlotStyle ->{{Orange, PointSize[0.015]}, {Purple, PointSize[0.015]}}},
ListPlot[Table[{i,
  2*E^(-0.1*i)*Cos[Sqrt[3]*i] + RandomReal[{-0.1, 0.1}], {i, 0,
  20, 0.5}],
  Table[{i, E^(-0.05*i)*Sin[2*i] + RandomReal[{-0.05, 0.05}], {i,
  0, 20, 0.5}}],
PlotStyle ->{{Cyan, PointSize[0.01]}, {Magenta, PointSize[0.01]}}},
PolarPlot[
1.5 + 0.5*Cos[3*theta]*E^(-theta/4), {theta, 0, 4 Pi},
PlotStyle ->{Brown, Thick}],
Graphics[{Gray, Dashed,
  Line[Table[{t, 2*E^(-0.1*t)}, {t, 0, 20, 0.5}]],
  Line[Table[{t, -2*E^(-0.1*t)}, {t, 0, 20, 0.5}]], Black,
  PointSize[0.02],
  Table[Point[{t, 2*E^(-0.1*t)*Cos[Sqrt[3]*t}], {t,
  Select[Range[0, 20, 0.1], Cos[Sqrt[3]*#] > 0.9 &]}],
  Text[Style["Amortiguamiento crítico", 10, Green], {14, 1.5}],
  Orange, Thick,
  Line[{{18, -1}, {19, -0.8}, {18.5, -0.6}, {19.5, -0.4}, {18.5, \
-0.2}, {19, 0}}]], PlotRange ->{{0, 22}, {-2.5, 2.5}},
Frame ->True, GridLines ->Automatic, GridLinesStyle ->LightGray,
PlotLabel ->
Style["Sistema Físico: Oscilador Armónico Complejo", 16, Bold],
FrameLabel ->{"Tiempo (s)", "Amplitud"}]
ExportToTikZ[ExportToTikZ14, "ExportToTikZ14.tex",
True, {"thick", "blue"}, {"solid", "red"}, {"thick",
"green"}, {"solid", "orange"}, {"thick", "cyan"}, {"solid",
"brown"}], True, {"Discreto", {6, -3}}, {"Polar", {18, -3}}, 0,
22, -3.5, 2.5];
CompilerTex[%, "PreferredEditor" ->"TeXstudio"]

```

En **Mathematica** se muestra como salida lo compartido en la subfigura 4.14a y los mensajes de texto:

Textos extraídos automáticamente: 3 elementos

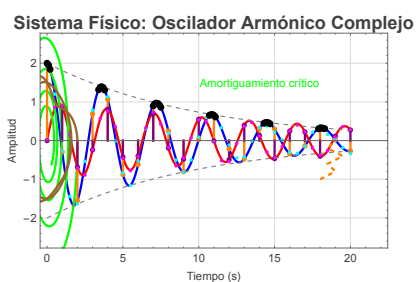
Usando editores ya detectados: 4 disponibles

:

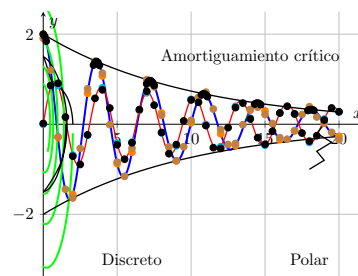
PDF generado: /Users/.../Downloads/ExportToTikZ14/ExportToTikZ14.pdf

/Users/.../Downloads/ExportToTikZ14/ExportToTikZ14.pdf

El archivo *ExportToTikZ14.tex* (guardado en *Downloads/ExportToTikZ14*) comparte el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 4.14b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 4.14: Gráficas del ejemplo 48

4.1 Ejemplos de uso de la función ExportToTikZ

Para más ejemplos

El código fuente de los ejemplos presentados en esta sección, junto con otra variedad de ejercicios de uso del comando `ExportToTikZ` se puede consultar de manera complementaria en el archivo: *4. Ejemplos ExportToTikZ.nb*, disponible en el enlace de descarga del paquete **ViTeX** (ver página [xiv](#)).

5

Función `ExportFillingToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Exportar regiones sombreadas y áreas de llenado a formato `TikZ`.
- ▶ Representar inecuaciones y sistemas de desigualdades gráficamente.
- ▶ Personalizar patrones de relleno, transparencias y estilos de frontera.
- ▶ Generar diagramas para optimización y análisis matemático aplicado.

La representación visual de regiones delimitadas por curvas constituye uno de los recursos didácticos más efectivos en el análisis matemático, particularmente al ilustrar conceptos como áreas bajo curvas, regiones entre funciones o zonas de integración definida. Si bien **Mathematica** dispone de capacidades nativas para sombreado mediante la opción `Filling`, la exportación directa en **Wolfram** a formatos vectoriales frecuentemente compromete la editabilidad del relleno, fusionándolo con las curvas envolventes en objetos gráficos que impiden modificaciones posteriores de opacidad, tonalidad o patrones de sombreado.

La función especializada del paquete **ViTeX** denominada `ExportFillingToTikZ` resuelve esta limitación mediante una arquitectura de conversión que separa explícitamente las áreas sombreadas de las trayectorias que las delimitan, generando construcciones `TikZ` independientes y modificables. Este comando ejecuta una descomposición estructural cuya arquitectura de procesamiento se fundamenta en el reconocimiento de patrones específicos generados por configuraciones `Filling` en funciones `Plot`. El flujo operacional comprende: validación exhaustiva del objeto gráfico de entrada verificando la presencia de primitivas `Polygon`, análisis y limpieza de conjuntos de puntos eliminando coordenadas numéricamente inestables o atípicas, construcción de rutas cerradas mediante conectores de `TikZ`, asignación independiente de estilos visuales a rellenos y contornos, y síntesis final de código $\text{\LaTeX}$ autónomo respetando estrictamente el orden de renderizado para evitar oclusiones inadvertidas entre capas gráficas.

La función `ExportFillingToTikZ` permite la conversión en **Wolfram Mathematica** de gráficas tipo `Plot` que incorporan sombreado de regiones mediante la opción `Filling`, produciendo código `TikZ/PGF-Plots` donde las áreas rellenas se materializan como comandos `\fill` independientes con control explícito de color, opacidad y posicionamiento en el orden de renderizado. A diferencia de estrategias que fusionan relleno y contorno en entidades gráficas indivisibles, `ExportFillingToTikZ` desacopla estos elementos, permitiendo edición selectiva posterior de cualquier atributo visual.

Esta capacidad resulta estratégica en contextos donde se necesitan adaptar tonalidades de sombreado, ajustar transparencias para superposición de múltiples regiones o modificar colores de contornos conforme a

paletas particulares después de la exportación inicial. El resultado de invocar **ExportFillingToTikZ** es un documento *.tex* con preámbulo completo y clase *standalone* que se puede compilar de forma inmediata mediante *pdflatex*, prescindiendo de intervención manual. Examinemos la sintaxis detallada de esta instrucción.

- **ExportFillingToTikZ**[*plot*, *filename*, *opts*]: Transforma un gráfico **Plot** de **Mathematica** con regiones sombreadas a código **TikZ/L^AT_EX** y lo exporta como archivo *.tex* compilable.

Parámetros:

- **plot**: Objeto gráfico generado por **Plot** con la opción **Filling** activa. Acepta configuraciones de sombreado como **Filling -> Bottom**, **Filling -> Top**, **Filling -> Axis** o especificaciones relacionales entre funciones múltiples como **Filling -> \{1 -> \{2\}\}**. El sistema valida automáticamente la presencia de primitivas **Polygon** en la estructura interna del gráfico y rechaza objetos que carezcan de áreas rellenas.
- **filename**: Cadena de caracteres especificando el nombre del archivo de salida, obligatoriamente con extensión *.tex* (por ejemplo: “*region_sombreada.tex*”). El mecanismo de exportación construye automáticamente una estructura de directorios anidada en *Downloads*, creando una carpeta con el nombre base del archivo (excluyendo la extensión) y depositando el documento *.tex* en su interior.
- **opts**: Secuencia de atributos en formato **Option -> valor** que parametrizan aspectos visuales, de presentación y de depuración del código generado. Las opciones disponibles son:

- * **Grid -> True|False** (opcional, predeterminado **True**): Controla la visibilidad del enrejillado cartesiano. **True** despliega líneas de cuadrícula mayor alineadas con los marcadores principales de ambos ejes, mientras **False** produce un gráfico limpio sin grilla.
- * **Axes -> True|False** (opcional, predeterminado **True**): Determina la presencia de ejes coordenados. **True** renderiza ejes centrados en el origen (**axis lines=center**) y **False** suprime completamente su visualización (**axis lines=none**).
- * **FillColor -> color** (opcional, predeterminado **black**): Especifica la tonalidad de sombreado para todas las áreas rellenas. Admite cualquier *string* representando colores válidos en *TikZ/PGFPlots*:

- Paleta básica: **black, white, red, green, blue, cyan, magenta, yellow**.
- Gama extendida: **orange, purple, brown, pink, gray, darkgray, lightgray**.
- Mezclas personalizadas: **red!50, blue!30!red, green!80!black**.

Si el valor suministrado no es un *string*, el sistema aplica automáticamente **orange** como alternativa.

- * **FillOpacity -> number** (opcional, predeterminado **0.3**): Regula la transparencia del sombreado mediante un valor decimal entre **0.0** (completamente transparente) y **1.0** (completamente opaco). Si el valor proporcionado no es numérico, se utiliza automáticamente **0.3**.
- * **PlotColors -> {color1, color2, ...}** (opcional, predeterminado **{black, black}**): Lista de *strings* definiendo las tonalidades de las curvas delimitadoras. Los colores se asignan secuencialmente a las líneas del gráfico; si el número de curvas excede el número de colores especificados, las líneas restantes adoptan **black**. Por ejemplo: **{red, blue, green!50!black}**.
- * **LegendLabels -> {label1, label2, ...}** (opcional, predeterminado **\{\}** sin leyenda): Colección de *strings* para etiquetar elementos en la leyenda del gráfico. Si la lista está vacía, no se genera leyenda. Cuando contiene elementos, la leyenda se posiciona automáticamente en **north west**.
- * **AxesLabels -> {xlabel, ylabel}** (opcional, predeterminado **\{x\$, y\$\}**): Par ordenado para etiquetar los ejes horizontal y vertical. Admite expresiones **L^AT_EX** arbitrarias. Por ejemplo: **{Tiempo (s), Velocidad (m/s)}** o **{\theta\$, f(\theta)\$}**.

* **Verbose** -> **True|False** (opcional, predeterminado **False**): Activa o desactiva el modo de reporte detallado. **False** ejecuta silenciosamente la conversión mostrando únicamente errores críticos, mientras que **True** genera un informe exhaustivo que incluye:

- **EXPORTACIÓN**: Ruta absoluta del archivo y tamaño del documento en caracteres.
- **GRÁFICO**: Conteo de polígonos de relleno, líneas detectadas y rangos numéricos en ambos ejes.
- **COLORES**: Color y opacidad de sombreado, junto con colores asignados a cada línea.
- **PRESENTACIÓN**: Estado de rejilla, ejes, etiquetas y leyenda.

Este modo se facilita para tareas de depuración, documentación de especificaciones y verificación de configuraciones complejas.

* **XMin** -> **number|Automatic**, **XMax** -> **number|Automatic**, **YMin** -> **number|Automatic**, **YMax** -> **number|Automatic** (opcionales, predeterminado **Automatic**): Especificaciones explícitas de los límites de los ejes coordenados. **Automatic** invoca cálculo automático basado en los datos con márgenes de 0.5 (eje x) y 1.0 (eje y). Cuando se definen manualmente, deben ser números reales válidos.

Retorna en **Mathematica**:

Ruta absoluta del archivo *.tex* exportado en el subdirectorío correspondiente dentro de *Downloads*.

Funcionalidades especiales:

- Validación topológica robusta de polígonos descartando vértices con coordenadas numéricamente inestables.
- Filtrado automático de polígonos degenerados con menos de 3 puntos válidos.
- Detección y eliminación de saltos anómalos entre vértices consecutivos que exceden umbrales estadísticos.
- Separación arquitectónica entre comandos `\addplot` (para curvas) y `\fill` (para rellenos) evitando trazados fantasma.
- Secuenciación correcta de elementos gráficos (líneas primero, rellenos después) garantizando alineación adecuada de leyendas.
- Construcción automática de regiones sintéticas cuando se detectan múltiples curvas sin polígonos explícitos.
- Cierre automático de polígonos mediante el operador `cycle` de `TikZ`.
- Generación de un documento $\text{\LaTeX}$ autónomo con preámbulo completo incluyendo *tikz*, *pgfplots* (compat=1.15) y clase *standalone*.
- Creación automática de jerarquía de directorios en caso de ausencia.
- Validación exhaustiva de parámetros con mensajes de error descriptivos y específicos. Por ejemplo, detecta un error de contenido si el gráfico carece de primitivas **Polygon** (ausencia de la opción **Filling** en **Plot**).

5.1 Ejemplos de uso de la función ExportFillingToTikZ

Los ejemplos desarrollados en esta sección demuestran las capacidades de la instrucción **ExportFillingToTikZ** mediante tres casos progresivamente complejos que cubren desde disposiciones básicas de sombreado hasta estructuras avanzadas de control sobre cada parámetro cromático, de transparencia y de presentación. Esta colección enfatiza en la personalización exhaustiva de atributos visuales: tonalidades de relleno mediante mezclas `TikZ`, ajuste fino de opacidades, asignación diferenciada de colores a curvas delimitadoras, construcción de leyendas descriptivas y manejo explícito de límites de ejes.

Cada caso ilustra facetas específicas del sistema: el ejemplo 49 introduce la sintaxis fundamental con énfasis en colores personalizados y transparencia aplicada a una función trigonométrica simple; el ejemplo 50 aborda el escenario de sombreado entre dos curvas con especificación de leyendas y el ejemplo 51 explora configuraciones complejas con múltiples funciones, estilos de línea diferenciados, límites de ejes explícitos y activación del modo `Verbose` que brinda una documentación detallada del proceso.

Un aspecto fundamental de estos ejemplos es la integración fluida con el compilador de $\text{\LaTeX}$ mediante la sentencia `CompiladorTex` (analizada extensivamente en el capítulo 2). Cada exportación culmina con la compilación automática del documento `.tex` generado por `ExportFillingToTikZ`, materializando el PDF definitivo sin requerir operaciones manuales adicionales.

Se recomienda al lector la experimentación sistemática en los ejemplos compartidos, modificando parámetros individuales: paletas cromáticas, valores de opacidad, configuración de rejilla o etiquetas de ejes. Esto permitirá interiorizar el uso de cada opción sobre el código `TikZ` resultante. Los archivos `.tex` producidos, depositados en el directorio *Downloads*, permanecen disponibles, facilitando así su inspección con cualquier editor de texto, donde es viable examinar el código fuente `TikZ` y ejecutar refinamientos manuales posteriores según las necesidades específicas de publicación o presentación.

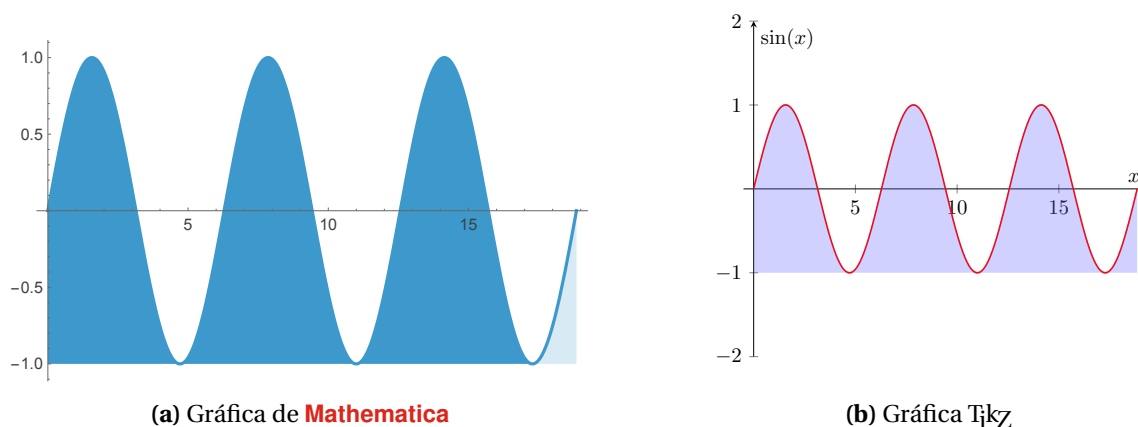


Figura 5.1: Gráficas del ejemplo 49

Ejemplo 49 Uso de colores personalizados y transparencia

```
ExportFillingToTikZ1 = Plot[Sin[x], {x, 0, 6 Pi}, Filling -> Bottom]
ExportFillingToTikZ[ExportFillingToTikZ1, "ExportFillingToTikZ1.tex",
"FillColor" -> "blue!60", "FillOpacity" -> 0.3,
"PlotColors" -> {"red"}, "AxesLabels" -> {"x", "sin(x)"},
"Grid" -> False];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Wolfram Mathematica** como salida, se obtiene lo mostrado en la subfigura 5.1a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportFillingToTikZ1/ExportFillingToTikZ1.pdf
/Users/.../Downloads/ExportFillingToTikZ1/ExportFillingToTikZ1.pdf

El archivo `ExportFillingToTikZ1.tex` (guardado en `Downloads/ExportFillingToTikZ1`) integra el código $\text{\LaTeX}$ editable que genera la gráfica `TikZ` de la subfigura 5.1b.

El siguiente ejemplo expone un caso típico de área entre curvas.

Ejemplo 50 Área entre curvas y empleo de opciones

```

ExportFillingToTikZ2 =
Plot[{x^2, 2*x + 3}, {x, -10, 10}, Filling ->{1 ->{2}},
FillingStyle ->Orange, PlotStyle ->{Blue, Red}]
ExportFillingToTikZ[ExportFillingToTikZ2, "ExportFillingToTikZ2.tex",
"FillColor" ->"orange", "FillOpacity" ->0.3,
"PlotColors" ->{"blue", "red"},
"LegendLabels" ->{"y = x^2", "y = 2x + 3"}];
CompiladorTex[%, "PreferredEditor" ->"TeXstudio"]

```

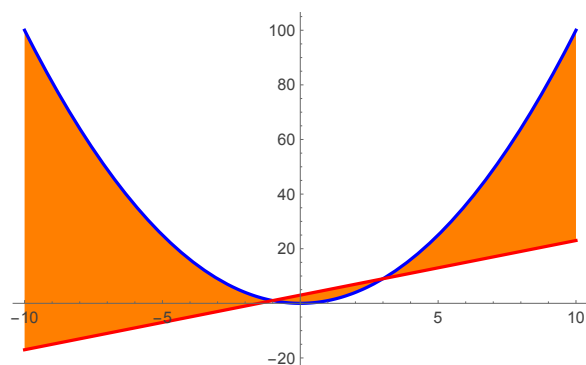
En **Mathematica** se muestra como salida la gráfica de la subfigura 5.2a y los mensajes de texto:

Usando editores ya detectados: 4 disponibles

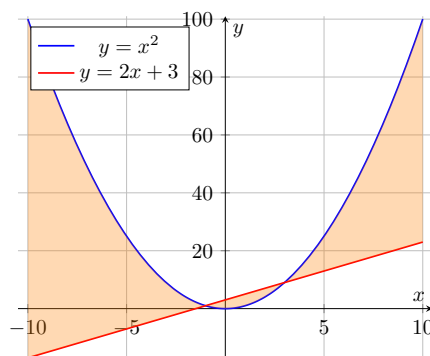
⋮

PDF generado: /Users/.../Downloads/ExportFillingToTikZ2/ExportFillingToTikZ2.pdf
 /Users/.../Downloads/ExportFillingToTikZ2/ExportFillingToTikZ2.pdf

El archivo *ExportFillingToTikZ2.tex* (guardado en *Downloads/ExportFillingToTikZ2*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 5.2b.

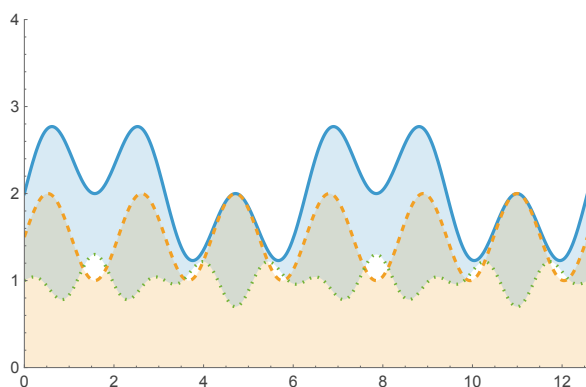


(a) Gráfica de **Mathematica**

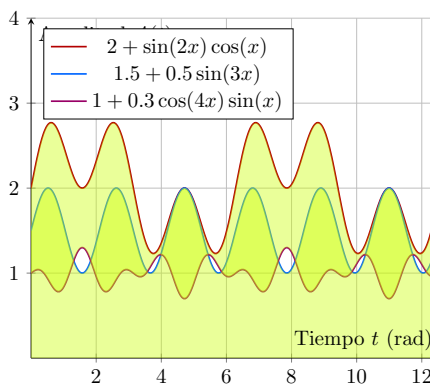


(b) Gráfica $\text{\LaTeX}$

Figura 5.2: Gráficas del ejemplo 50



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 5.3: Gráficas del ejemplo 51

Ejemplo 51 Múltiples funciones trigonométricas

```

ExportFillingToTikZ3 =
Plot[{2 + Sin[2 x] Cos[x], 1.5 + 0.5 Sin[3 x],
      1 + 0.3 Cos[4 x] Sin[x]}, {x, 0, 4 Pi},
Filling -> {1 -> {3}, 2 -> Bottom},
PlotStyle -> {Thick, Dashed, Dotted},
PlotRange -> {{0, 4 Pi}, {0, 4}}]
ExportFillingToTikZ[ExportFillingToTikZ3, "ExportFillingToTikZ3.tex",
"FillColor" -> "green!20!yellow", "FillOpacity" -> 0.4,
"PlotColors" -> {"red!70!black", "blue!60!cyan", "purple!80!blue"},
"LegendLabels" -> {"2 + sin(2x)cos(x)", "1.5 + 0.5sin(3x)",
"1 + 0.3cos(4x)sin(x)"},
"AxesLabels" -> {"Tiempo t (rad)", "Amplitud A(t)"},
"XMin" -> 0, "XMax" -> 4*3.14159, "YMin" -> 0, "YMax" -> 4,
"Grid" -> True, "Axes" -> True, "Verbose" -> True];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** la salida obtenida muestra lo visualizado en la subfigura 5.3a y los mensajes:

```

Polígono 3: 1323 puntos válidos
Polígono 4: 2974 puntos válidos
=== INFORMACIÓN DE EXPORTACIÓN ===
Archivo exportado exitosamente a: ...
Tamaño del documento: 162073 caracteres
=== ANÁLISIS DEL GRÁFICO ===
Polígonos de relleno encontrados: 2
Líneas encontradas: 3
Rango X detectado: [0., 12.566]
Rango Y detectado: [0., 4.]
=== CONFIGURACIÓN DE COLORES ===
Color de relleno usado: green!20!yellow
Opacidad de relleno: 0.4
Colores de líneas: red!70!black, blue!60!cyan, purple!80!blue
=== OPCIONES DE PRESENTACIÓN ===
Rejilla activada: True
Ejes centrales: True
Etiquetas de ejes: Tiempo t (rad), Amplitud A(t)
Leyenda incluida: True
Etiquetas de leyenda: 2 + sin(2x)cos(x), 1.5 + 0.5sin(3x), 1 + 0.3cos(4x)sin(x)
=====
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportFillingToTikZ3/ExportFillingToTikZ3.pdf
/Users/.../Downloads/ExportFillingToTikZ3/ExportFillingToTikZ3.pdf

```

El archivo *ExportFillingToTikZ3.tex* (guardado en *Downloads/ExportFillingToTikZ3*) posee el código $\LaTeX$ editable que produce la gráfica TikZ de la subfigura 5.3b.

Para más ejemplos

Si el lector desea tener acceso al código fuente de los ejemplos anteriores y otros, puede consultar de manera complementaria el archivo: *5. Ejemplos ExportFillingToTikZ.nb*, habilitado en el enlace de descarga del paquete **V_lT_EX** (ver página [xiv](#)).

6

Función `ExportGraphToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Convertir grafos de **Mathematica** en diagramas `TikZ` profesionales.
- ▶ Exportar grafos dirigidos, no dirigidos, ponderados y etiquetados.
- ▶ Personalizar *layouts* automáticos y posicionamiento de nodos.
- ▶ Configurar estilos de aristas, colores de nodos y etiquetas.

La visualización de estructuras relacionales mediante grafos constituye un pilar fundamental en disciplinas que van desde la teoría de grafos hasta aplicaciones en redes complejas, análisis de algoritmos, ciencias de la computación y modelado de sistemas interconectados. Si bien **Wolfram** ofrece capacidades nativas robustas para la creación y manipulación de objetos `Graph` con diversos algoritmos de *layout* y opciones de estilización, la exportación de esos objetos a documentos $\text{\LaTeX}$ requiere conversiones manuales tediosas con resultados traducidos en imágenes rasterizadas que sacrifican escalabilidad, editabilidad tipográfica y consistencia estética.

El comando `ExportGraphToTikZ` del paquete **VilTeX** permite en este sentido, transformar objetos `Graph` nativos de **Mathematica** en construcciones `TikZ` completamente parametrizadas. Esta sentencia descompone la estructura del grafo en sus componentes fundamentales -vértices, aristas, coordenadas espaciales, pesos, direccionalidad- y reconstruye cada elemento como una primitiva `TikZ` editable en atributos geométricos, cromáticos y topológicos.

El mecanismo operacional de `ExportGraphToTikZ` se fundamenta en el análisis profundo de la representación interna del objeto `Graph`, es decir: la validación rigurosa del tipo de entrada verificando la naturaleza del objeto mediante `GraphQ`, la extracción de la lista de vértices y de aristas con preservación de tipos (`DirectedEdge` vs `UndirectedEdge`), la recuperación de coordenadas espaciales desde `VertexCoordinates` o el cálculo mediante `GraphEmbedding`, la detección automática de pesos asociados a las aristas explorando propiedades del grafo, la determinación de direccionalidad basada en el análisis del tipo de aristas y finalmente la síntesis de código `TikZ` estructurado con nodos geométricos parametrizados y conexiones estilizadas.

La función `ExportGraphToTikZ` especializa la traducción de objetos `Graph` de **Mathematica** a código `TikZ` nativo de $\text{\LaTeX}$, generando documentos autónomos donde vértices se materializan como nodos `TikZ` con formas geométricas configurables (círculos, cuadrados, triángulos) y aristas como trayecto-

rias con control independiente de grosor, color, patrón de línea y direccionalidad. Contrariamente a exportaciones que unifican el grafo en un objeto monolítico, **ExportGraphToTikZ** preserva la identidad de cada componente, facilitando su edición posterior.

La salida que ofrece el comando **ExportGraphToTikZ** es un archivo *.tex* con estructura completa y clase *standalone*, directamente compilable mediante cualquier distribución $\text{\LaTeX}$ estándar. Veamos la descripción de uso de esta instrucción.

- **ExportGraphToTikZ[graph, filename, options]**: Transforma un objeto **Graph** de **Mathematica** a código $\text{\LaTeX}$ y lo exporta como un archivo *.tex* compilable.

Parámetros:

- **graph**: Objeto **Graph** de **Wolfram**. El sistema lo valida automáticamente mediante **GraphQ** y acepta grafos dirigidos, no dirigidos, ponderados o combinaciones arbitrarias. Soporta especificaciones mediante listas de aristas, matrices de adyacencia o construcciones explícitas con **Graph[vertices, edges]**.
- **filename**: Cadena de caracteres que indica el nombre del archivo de salida, obligatoriamente con extensión *.tex* (por ejemplo: “*mi_grafo.tex*”). El sistema genera automáticamente una jerarquía de directorios en *Downloads*, construyendo una carpeta con el nombre base del archivo (sin extensión) y depositando el documento *.tex* en su interior.
- **options**: Secuencia de reglas en formato **Option -> valor** que parametrizan todos los aspectos visuales y de presentación del grafo. Las opciones se organizan en cuatro categorías funcionales:

APARIENCIA DE VÉRTICES:

- * **VertexSize -> number** (opcional, predeterminado 0.1): Dimensión de los nodos en centímetros. Los valores típicos oscilan entre 0.1 y 0.3 cm. El sistema no procesa valores inferiores a 0.1 cm por limitaciones de renderizado.
- * **VertexColor -> color** (opcional, predeterminado **blue**): Tonalidad de relleno de los vértices. Admite paletas:
 - Colores básicos: **black, white, red, green, blue, yellow, cyan, magenta, gray**.
 - Colores predefinidos: **darkblue, lightblue, darkgreen, lightgreen, darkred, lightgray, darkgray, orange, purple, pink**.
- * **VertexLabels -> True|False** (opcional, predeterminado **True**): Controla la visibilidad de etiquetas identificadoras de vértices. **True** superpone el identificador en modo matemático y **False** genera nodos sin anotación.
- * **VertexLabelColor -> color** (opcional, predeterminado **black**): Color del texto de las etiquetas de vértices. Acepta la misma paleta que **VertexColor**.
- * **NodeShape -> forma** (opcional, predeterminado **circle**): Geometría de los nodos. Opciones disponibles:
 - **circle**: Nodos circulares (predeterminado).
 - **square**: Nodos rectangulares/cuadrados.
 - **triangle**: Nodos triangulares.

APARIENCIA DE ARISTAS:

- * **EdgeThickness -> grosor** (opcional, predeterminado **thick**): Grosor de las líneas conectoras. Especificaciones válidas:
 - **thin**: Líneas delgadas.
 - **thick**: Líneas gruesas (predeterminado).
 - **very thick**: Líneas muy gruesas.
 - **ultra thick**: Líneas ultra gruesas.

- * **EdgeColor** -> **color** (opcional, predeterminado **black**): Tonalidad de las aristas. Admite la misma paleta cromática que **VertexColor**.
- * **EdgeStyle** -> **estilo** (opcional, predeterminado **solid**): Patrón de línea para las conexiones:
 - **solid**: Líneas continuas (predeterminado).
 - **dashed**: Líneas discontinuas/segmentadas.
 - **dotted**: Líneas punteadas.
 - **thick**: Líneas continuas gruesas.
- * **Directed** -> **True|False|Automatic** (opcional, predeterminado **Automatic**): Control de direccionalidad de las aristas:
 - **True**: Fuerza todas las aristas como dirigidas (con terminaciones de flecha).
 - **False**: Fuerza todas las aristas como no dirigidas.
 - **Automatic**: Detecta automáticamente basándose en el tipo de objeto **Graph** y en los tipos de arista (**DirectedEdge** vs **UndirectedEdge**).

Nota: Si el grafo original es no dirigido, especificar **True** no altera su naturaleza. Inversamente, si es dirigido, **False** lo renderiza sin flechas.

PESOS DE ARISTAS:

- * **ShowEdgeWeights** -> **True|False** (opcional, predeterminado **True**): Controla la visualización de pesos asociados a las aristas. El sistema extrae automáticamente pesos desde propiedades del grafo (**EdgeWeight** y **EdgeLabels**).
- * **EdgeWeightColor** -> **color** (opcional, predeterminado **red**): Tonalidad del texto de los pesos. Admite la misma paleta que **VertexColor**.
- * **EdgeWeightSize** -> **size** (opcional, predeterminado **small**): Dimensión tipográfica de los pesos:
 - **small**: Texto pequeño (predeterminado).
 - **large**: Texto grande.
 - **Large**: Texto muy grande.
 - **huge**: Texto enorme.

SISTEMA DE COORDENADAS Y EJES:

- * **Grid** -> **True|False** (opcional, predeterminado **False**): Activa o desactiva la visualización de rejilla cartesiana de fondo.
- * **ShowAxes** -> **True|False** (opcional, predeterminado **False**): Controla la presencia de ejes coordenados x y y .
- * **XMin** -> **number|Automatic**, **XMax** -> **number|Automatic**, **YMin** -> **number|Automatic**, **YMax** -> **number|Automatic** (opcionales, predeterminado **Automatic**): Límites explícitos de los ejes. **Automatic** calcula límites basándose en las coordenadas de vértices con márgenes de 0.5 unidades. Cuando se especifican manualmente, deben ser números reales válidos.
- * **GraphScale** -> **number|Automatic** (opcional, predeterminado **Automatic**): Factor de escala global del grafo. **Automatic** calcula automáticamente un factor que ajusta el grafo a aproximadamente 8 cm de ancho. Los valores manuales típicos oscilan entre 0.1 y 2.0. Esta opción facilita el redimensionamiento global sin modificar tamaños individuales de vértices.

Retorna en **Mathematica**:

Ruta absoluta del archivo *.tex* exportado en el subdirectorio correspondiente dentro de *Downloads*. Retorna **\$Failed** en caso de errores de validación.

Funcionalidades especiales:

- Detección automática de direccionalidad mediante análisis dual: `DirectedGraphQ` y verificación de tipos de arista.
- Preservación fiel del tipo de arista durante todo el procesamiento.
- Extracción multinivel de pesos de aristas explorando `EdgeWeight`, `EdgeLabels` y estructura interna del grafo.
- Recuperación inteligente de coordenadas desde `VertexCoordinates` o cálculo mediante `GraphEmbedding`.
- Generación de *layout* circular sintético cuando fallan métodos de extracción de coordenadas.
- Limpieza automática de nombres de vértices escapando caracteres especiales $\text{\LaTeX}$ (`\`, `{`, `}`, `$`, `&`, `%`, `#`, `^`, `_`, `~`).
- Posicionamiento automático de etiquetas de pesos en puntos medios de aristas con fondo blanco para legibilidad.
- Generación condicional de entorno `axis` solo cuando se requieren ejes o rejilla, optimizando el código.
- Cálculo adaptativo de escala para ajustar grafos de dimensiones arbitrarias a tamaños estándar en un documento.
- Definición automática de paleta cromática extendida mediante comandos `\definecolor` en el preámbulo.
- Documento $\text{\LaTeX}$ autónomo con preámbulo completo incluyendo `tikz`, `xcolor`, bibliotecas geométricas y paquetes matemáticos.
- Creación de jerarquía de directorios en caso de ausencia.

6.1 Ejemplos de uso de la función ExportGraphToTikZ

Esta sección ilustra el funcionamiento de `ExportGraphToTikZ` mediante cuatro ejemplos que exploran distintas configuraciones y tipos de grafos. El ejemplo 52 trabaja con un grafo construido explícitamente mediante una lista de aristas, activando la visualización de ejes coordenados y rejilla de referencia, además de especificar límites personalizados para los rangos de visualización y ajustar propiedades geométricas de vértices y aristas. Los ejemplos 53, 54 y 55 abordan respectivamente: grafos pseudoaleatorios con pesos en las aristas donde se demuestra el control sobre la presentación tipográfica de estos valores, grafo tipo *FruchtGraph* que ejemplifica la asignación independiente de colores a vértices y aristas y grafo tipo *PetersenGraph* donde se aplican estilos de línea no convencionales junto con escalado aumentado del grafo completo.

La progresión de los ejemplos transita desde la especificación manual de parámetros visuales y geométricos hasta el trabajo con grafos clásicos cuya estructura topológica está predeterminada y donde el énfasis recae en las opciones de estilización. Al igual que en las otras secciones de ejemplos, se recomienda al lector experimentar mediante el código fuente, modificando las distintas opciones con el fin de comprender cómo cada parámetro afecta el código `TikZ`.

Ejemplo 52 Grafo con ejes y rejilla

```
ExportGraphToTikZ1 =
Grafo[{{a, f}, {a, g}, {a, j}, {b, e}, {b, i}, {c, d}, {c, e}, {c, f},
{d, e}, {d, j}, {e, f}, {e, h}, {g, j}, {h, j}, {i, j}}]
ExportGraphToTikZ[ExportGraphToTikZ1, "ExportGraphToTikZ1.tex",
"ShowAxes" → True, "Grid" → True, "XMin" → -0.5, "XMax" → 4,
"YMin" → -0.5, "YMax" → 3, "VertexColor" → "orange",
```

```
"EdgeThickness" → "ultra thick", "VertexSize" → 0.4];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

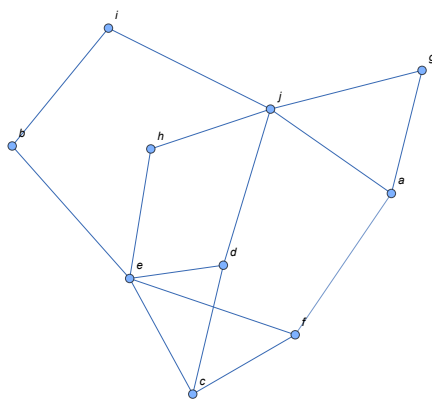
En el código anterior, se aclara que la instrucción **Grafo** utilizada pertenece a otro paquete elaborado por el autor de esta propuesta, llamado “*VilCretas*” (Vílchez, 2018). En **Mathematica** se arroja como salida la gráfica de la subfigura 6.1a y los mensajes de texto:

Usando editores ya detectados: 4 disponibles

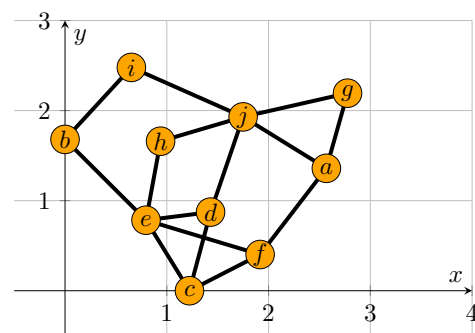
⋮

PDF generado: /Users/.../Downloads/ExportGraphToTikZ1/ExportGraphToTikZ1.pdf
/Users/.../Downloads/ExportGraphToTikZ1/ExportGraphToTikZ1.pdf

El archivo *ExportGraphToTikZ1.tex* (guardado en *Downloads/ExportGraphToTikZ1*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 6.1b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 6.1: Gráficas del ejemplo 52

Ejemplo 53 Grafo pseudoaleatorio ponderado con varias opciones

```
ExportGraphToTikZ2 = GrafoRandom[10, 10, pesos → True, letras → True]
ExportGraphToTikZ[ExportGraphToTikZ2, "ExportGraphToTikZ2.tex",
"EdgeWeightSize" → "Large", "VertexColor" → "white",
"EdgeWeightColor" → "magenta", "GraphScale" → 1.5];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

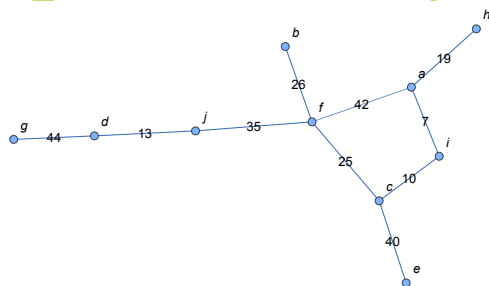
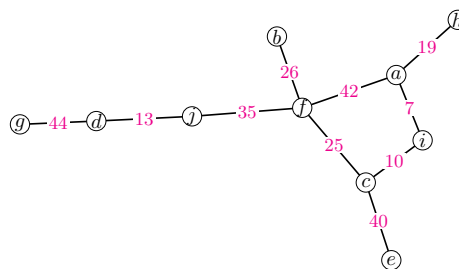
En el código de este ejemplo, también se ha recurrido al uso del paquete “*VilCretas*” y particularmente a la sentencia **GrafoRandom** para construir un grafo pseudoaleatorio con 10 vértices y 10 aristas. En **Wolfram** se obtiene como salida la gráfica de la subfigura 6.2a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportGraphToTikZ2/ExportGraphToTikZ2.pdf
/Users/.../Downloads/ExportGraphToTikZ2/ExportGraphToTikZ2.pdf

El archivo *ExportGraphToTikZ2.tex* (guardado en *Downloads/ExportGraphToTikZ2*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 6.2b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 6.2: Gráficas del ejemplo 53

Ejemplo 54 Grafo especial: *FruchtGraph*

```
ExportGraphToTikZ3 = GrafoDato[tipo -> "FruchtGraph"]
ExportGraphToTikZ[ExportGraphToTikZ3, "ExportGraphToTikZ3.tex",
"VertexColor" -> "green", "EdgeColor" -> "red",
"VertexLabels" -> True, "VertexSize" -> 0.3, "GraphScale" -> 2];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

Aquí, el comando **GrafoDato** del paquete “*VilCretas*” permite la construcción del grafo especial: “*FruchtGraph*”. En **Mathematica** esto muestra como salida la gráfica de la subfigura 6.3a y los mensajes:

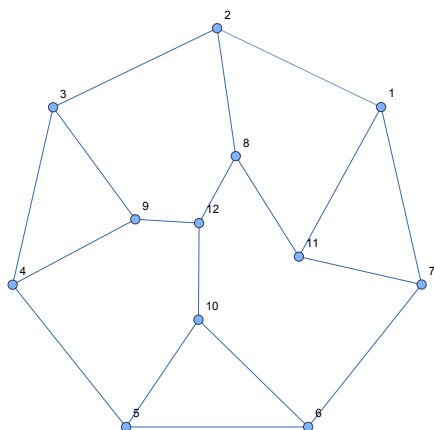
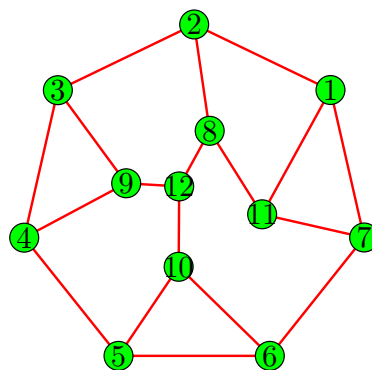
Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportGraphToTikZ3/ExportGraphToTikZ3.pdf

/Users/.../Downloads/ExportGraphToTikZ3/ExportGraphToTikZ3.pdf

El archivo *ExportGraphToTikZ3.tex* (guardado en *Downloads/ExportGraphToTikZ3*) facilita el código $\text{\LaTeX}$ editable que produce la gráfica TikZ de la subfigura 6.3b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 6.3: Gráficas del ejemplo 54

Ejemplo 55 Grafo especial: *PetersenGraph*

```
ExportGraphToTikZ4 = GrafoDato[tipo → "PetersenGraph"]
ExportGraphToTikZ[ExportGraphToTikZ4, "ExportGraphToTikZ4.tex",
"VertexColor" → "orange", "EdgeColor" → "blue",
"VertexLabels" → True, "VertexSize" → 0.3, "GraphScale" → 2.5,
"EdgeStyle" → "dotted"];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

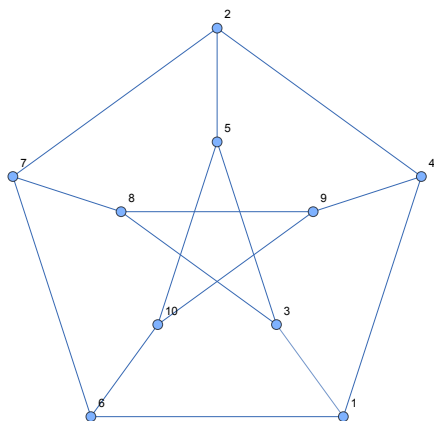
En el código, la sentencia `GrafoDato` del paquete “*VilCretas*” crea el grafo especial: “*PetersenGraph*”. En **Mathematica** la salida desplegada corresponde a la gráfica de la subfigura 6.4a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

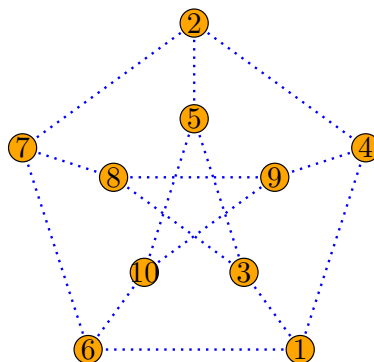
⋮

PDF generado: /Users/.../Downloads/ExportGraphToTikZ4/ExportGraphToTikZ4.pdf
/Users/.../Downloads/ExportGraphToTikZ4/ExportGraphToTikZ4.pdf

El archivo *ExportGraphToTikZ4.tex* (guardado en *Downloads/ExportGraphToTikZ4*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 6.4b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 6.4: Gráficas del ejemplo 55

Para más ejemplos

En el file: 6. *Ejemplos ExportGraphToTikZ.nb* disponible en el enlace de descarga del paquete $\text{\LaTeX}$ (ver página xiv), el lector puede consultar quince ejercicios de empleo de la instrucción `ExportGraphToTikZ` entre los que se incluyen los ejemplos ya socializados.

7

Función `ExportMaquinaToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Generar diagramas de máquinas de estado finito en formato `TikZ`.
- ▶ Representar transiciones, estados iniciales y estados finales.
- ▶ Personalizar estilos de estados y transiciones para claridad visual.
- ▶ Documentar sistemas secuenciales y autómatas en ingeniería.

El diseño y análisis de máquinas de estado finito representa un dominio central en múltiples áreas: desde la construcción de compiladores y procesadores de lenguaje hasta el modelado de protocolos de comunicación, circuitos secuenciales y sistemas reactivos. Aunque **Wolfram** facilita el trabajo algebraico con máquinas de estado mediante estructuras simbólicas, la integración de estos objetos en documentación técnica o material académico requiere convertirlos a formatos visuales compatibles con $\text{\LaTeX}$, proceso que tradicionalmente implica dibujo manual o herramientas externas desvinculadas del flujo de trabajo original.

El comando `ExportMaquinaToTikZ` del paquete $\text{\LaTeX}$ aborda esta problemática permitiendo traducir especificaciones algebraicas completas de máquinas de estado finito directamente a representaciones gráficas `TikZ` editables. La función opera sobre los componentes matemáticos puros de la máquina y sintetiza un diagrama de estados profesional aplicando convenciones estándar de visualización automática. Internamente, `ExportMaquinaToTikZ` realiza lo siguiente: verifica la consistencia matemática de la máquina de estado asegurando coherencia entre estados declarados y referenciados, valida los símbolos en transiciones respecto a alfabetos, confirma la correctitud formal de la función de transición; posteriormente aplica algoritmos geométricos para distribuir estados en el plano según patrones predefinidos (circular, matricial o estocástico); luego analiza la topología de transiciones identificando casos especiales como *loops* reflexivos, aristas múltiples entre pares de estados y conexiones bidireccionales que requieren tratamiento diferenciado; finalmente traduce símbolos matemáticos de **Mathematica** (incluyendo subíndices y alfabetos griegos) a comandos $\text{\LaTeX}$ equivalentes antes de ensamblar el código `TikZ` completo.

La función `ExportMaquinaToTikZ` transforma definiciones formales de máquinas de estado finito en diagramas de estados `TikZ` completamente parametrizables, donde cada componente matemático del modelo -estados, transiciones, símbolos- se convierte en un elemento gráfico independiente con propiedades ajustables. A diferencia de exportaciones rasterizadas o monolíticas, este enfoque

preserva la editabilidad completa del diagrama, permitiendo refinamientos tipográficos posteriores y garantizando integración coherente con documentos $\text{\LaTeX}$ existentes.

El resultado producido es un documento $\text{\LaTeX}$ autocontenido de clase *standalone*, preparado para compilación inmediata sin dependencias externas adicionales. A continuación se detalla la especificación funcional completa de `ExportMaquinaToTikZ`.

- `ExportMaquinaToTikZ[states, inputAlphabet, outputAlphabet, initialState, transitions, filename, options]`: Convierte una especificación matemática de máquina de estado finita en un diagrama $\text{\TikZ}$ autocontenido exportado como archivo *.tex*.

Parámetros:

- **states**: Conjunto de estados de la máquina como lista de **Mathematica**. Soporta cualquier expresión simbólica válida: identificadores alfanuméricos (`q0`, `s1`), construcciones con subíndices (`Subscript[ $\Sigma$ , i]`), letras griegas puras o combinaciones arbitrarias. La función valida automáticamente que sea una lista no vacía y que constituya un conjunto con todos los estados mencionados en las transiciones.
- **inputAlphabet**: Alfabeto de entrada como lista finita de símbolos. Define el dominio de la función de transición. Acepta elementos numéricos, caracteres o expresiones simbólicas complejas. Durante la validación, el sistema verifica que toda transición especificada utilice únicamente símbolos de este conjunto.
- **outputAlphabet**: Alfabeto de salida como lista finita de símbolos. Define el codominio de la función de salida. Sigue las mismas convenciones de formato que el alfabeto de entrada, con validación automática de pertenencia para símbolos producidos en transiciones.
- **initialState**: Estado desde el cual inicia la operación de la máquina. Debe pertenecer al conjunto **states**. La función lo marca visualmente mediante una flecha entrante distintiva sin origen, siguiendo la notación estándar de diagramas de transición.
- **transitions**: Función de transición/salida representada como lista de cuádruplas `{origen, entrada, destino, salida}`. Cada cuádrupla codifica: “estando en *origen* y leyendo *entrada*, transitar a *destino* y emitir *salida*”. El sistema valida exhaustivamente: (i) pertenencia de *origen* y *destino* a **states**, (ii) pertenencia de *entrada* a **inputAlphabet**, (iii) pertenencia de *salida* a **outputAlphabet** y (iv) estructura cuádruple correcta.
- **filename**: nombre del archivo de salida con extensión obligatoria *.tex*. El sistema crea automáticamente una estructura de directorios en *Downloads/nombre_base* donde deposita el documento generado.
- **options**: Secuencia opcional de pares “clave” $\rightarrow$ valor que controlan aspectos visuales del diagrama. Se organizan en las siguientes categorías:

CONTROL DE ETIQUETADO:

- * “**ShowLabels**” $\rightarrow$ **True|False** (predeterminado: **True**): Determina si los identificadores de estado aparecen visibles dentro de los nodos. Con **True**, cada nodo muestra su identificador en modo matemático; con **False**, los nodos quedan vacíos facilitando diagramas abstractos o esquemáticos.

DIMENSIONES GEOMÉTRICAS:

- * “**StateSize**” $\rightarrow$ **number** (predeterminado: 0.6): Radio efectivo de cada nodo de estado en centímetros. Controla directamente el parámetro *minimum size* de $\text{\TikZ}$. Rango práctico recomendado: 0.4-1.0 cm dependiendo de la complejidad de etiquetas y densidad del diagrama.
- * “**NodeDistance**” $\rightarrow$ **number** (predeterminado: 3): Parámetro de escala espacial que determina la separación característica entre estados. Su interpretación depende del *layout*: en

modo circular define el radio del círculo; en modo rejilla establece el paso de la cuadrícula; en modo resorte delimita el rango de dispersión aleatoria. Los valores típicos son: 2-5 cm.

ESQUEMAS CROMÁTICOS:

- * **"StateColor"** -> **cadena** (predeterminado: **"white"**): Color de relleno homogéneo aplicado a todos los nodos cuando la coloración automática está inactiva. Soporta tres familias de colores:
 - Básicos de TkZ: **"red"**, **"blue"**, **"green"**, **"yellow"**, **"cyan"**, **"magenta"**, **"black"**, **"white"**, **"gray"**.
 - Extendidos de *xcolor*: **"Red"**, **"Blue"**, **"Violet"**, **"Emerald"**, **"Turquoise"**, **"Lavender"**, **"Salmon"**, entre otros 100+ colores nombrados.
 - Otros: **"lightblue"**, **"lightgreen"**, **"lightyellow"**, **"darkgray"**, **"orange"**, **"purple"**, **"pink"**.
- * **"AutoColor"** -> **True|False** (predeterminado: **False**): Activa un sistema de coloración diferenciada donde cada estado recibe un tono único de una paleta interna de 36 colores balanceados visualmente. Características del modo automático:
 - Asignación cíclica: si hay más estados que colores disponibles, la paleta se reutiliza cíclicamente.
 - Coherencia de flujo: las aristas de transición heredan el color de su estado origen, creando trayectorias cromáticamente identificables.
 - Anulación de color uniforme: cuando está activo, **"StateColor"** se ignora completamente.

ESTILIZACIÓN DE ARISTAS:

- * **"ArrowThickness"** -> **True|False** (predeterminado: **False**): Modifica únicamente el grosor de la flecha marcadora del estado inicial. Con **True** aplica el atributo *ultra thick*; con **False** usa el grosor predeterminado del contexto. No afecta las transiciones regulares.
- * **"TransitionColor"** -> **cadena** (predeterminado: **"black"**): Color uniforme de todas las aristas cuando la coloración automática está deshabilitada. Admite la misma taxonomía cromática que **"StateColor"**. Se ignora si **"AutoColor"** está activo.

ALGORITMOS DE DISPOSICIÓN:

- * **"LayoutStyle"** -> **cadena** (predeterminado: **"circular"**): Selecciona el algoritmo de posicionamiento automático de estados en el plano. Hay tres modalidades disponibles:
 - **"circular"**: Distribuye estados sobre una circunferencia de radio **"NodeDistance"** con separación angular uniforme $\Delta\theta = 2\pi/n$. El primer estado se posiciona en $\theta = 0$ (eje +x) y la secuencia progresa antihorariamente. Óptimo para máquinas donde se privilegia la simetría radial.
 - **"grid"**: Organiza estados en una matriz rectangular aproximadamente cuadrada. El algoritmo calcula $c = \lceil \sqrt{n} \rceil$ columnas y $r = \lceil n/c \rceil$ filas, distribuyendo estados en orden lexicográfico de izquierda a derecha y arriba hacia abajo. El espaciado horizontal y vertical es **"NodeDistance"**. La matriz resultante se centra en el origen. Ideal para máquinas con estructura jerárquica o secuencial donde se requiere ordenamiento visual explícito.
 - **"spring"**: Genera posiciones pseudoaleatorias uniformemente distribuidas en $[-d, d] \times [-d, d]$ donde $d = \text{"NodeDistance"}$. Útil para romper simetrías artificiales o explorar configuraciones no convencionales, aunque puede requerir ajustes manuales posteriores si produce solapamientos.

AYUDAS VISUALES AUXILIARES:

- * **"ShowGrid"** -> **True|False** (predeterminado: **False**): Superpone una rejilla cartesiana de fondo con líneas auxiliares espaciadas cada 0.5 cm. El sistema calcula automática-

mente la extensión necesaria analizando coordenadas extremas de estados y añadiendo márgenes de 1.5 unidades. Las líneas se dibujan en gris translúcido (`gray!20`) para no competir visualmente con el diagrama principal. Útil durante refinamiento manual del código TikZ o para verificación dimensional.

- * `"ShowAxes"` -> `True|False` (predeterminado: `False`): Añade ejes coordenados cartesianos con flechas terminales y etiquetas x y y . Los ejes se extienden automáticamente cubriendo el envolvente convexo de estados más márgenes de 1.5 unidades, dibujándose en negro con grosor medio. Facilita orientación espacial y comprensión de la geometría del *layout*, especialmente en modos rejilla y resorte.

Valor de retorno:

Cadena conteniendo la ruta absoluta del archivo `.tex` generado exitosamente. En caso de fallo en validaciones (formato incorrecto, inconsistencias matemáticas, errores de archivo), retorna el símbolo `$Failed` acompañado de mensajes diagnósticos impresos en la sesión de **Mathematica**.

Capacidades avanzadas implementadas:

- **Validación matemática rigurosa:** Antes de generar código, el sistema verifica exhaustivamente la consistencia formal de la máquina: no-vacuidad de conjuntos, pertenencia de estado inicial, clausura de transiciones respecto a estados y alfabetos declarados, y correctitud estructural de cuádruplas. Cualquier violación aborta la exportación con diagnósticos precisos.
- **Traducción simbólica sofisticada:** Convierte automáticamente construcciones matemáticas de **Mathematica** a sus equivalentes $\text{\LaTeX}$: expresiones `Subscript[Sigma, i]` se mapean a `\sigma_{i}`, letras griegas *Unicode* se transforman en comandos correspondientes (`Alpha` $\rightarrow$ `\alpha`), preservando fidelidad semántica completa.
- **Agregación inteligente de transiciones:** Al detectar múltiples arcos entre el mismo par de estados (diferentes símbolos entrada/salida), los consolida en una única arista visual con etiqueta compuesta " $a/x, b/y$ ", reduciendo desorden gráfico sin perder información.
- **Tratamiento especializado de *auto-loops*:** Las transiciones reflexivas (estado a sí mismo) se identifican y renderizan como *bucles* usando constructos TikZ tipo *loop*. Cuando un estado tiene múltiples auto-transiciones, se distribuyen en cuatro posiciones angulares (*above*, *right*, *below*, *left*) evitando superposición.
- **Resolución de bidireccionalidad:** Si existen transiciones tanto $p \rightarrow q$ como $q \rightarrow p$, el algoritmo aplica curvatura asimétrica (*bend left*) a ambas aristas con ángulos calculados según geometría relativa, posicionando etiquetas arriba y abajo respectivamente para maximizar legibilidad.
- **Coloración coordinada de flujos:** En modo `"AutoColor"`, cada transición adquiere el color de su estado origen, permitiendo rastrear visualmente caminos y cadenas de procesamiento en la máquina mediante seguimiento cromático.
- **Marcador de inicialización:** Genera automáticamente una flecha horizontal sin origen apuntando al estado inicial, posicionada 1.5 unidades a su izquierda, con grosor controlable independientemente del resto del diagrama.
- **Sanitización de identificadores:** Los nombres simbólicos se procesan para generar identificadores TikZ válidos: eliminación de caracteres especiales (`\`, `{`, `$`, etc.), prefijado con `"state"` para garantizar inicio alfabético, conversión de construcciones *Unicode* complejas a formas alfanuméricas simples.
- **Cálculo adaptativo de ventanas:** Los rangos para rejilla y ejes se determinan dinámicamente analizando envolventes convexas de posiciones calculadas y añadiendo márgenes proporcionales. Esto garantiza que el diagrama siempre quede completamente visible sin recortes ni espacios excesivos.

- **Documento autosuficiente:** Genera archivo $\text{\LaTeX}$ completo incluyendo: declaración de clase *standalone* con márgenes configurados, importación de bibliotecas necesarias (*tikz*, *automata*, *positioning*, *arrows*, *xcolor*, *amsmath*, *amssymb*), definiciones explícitas de colores personalizados mediante $\text{\definecolor}$ y estructura completa $\text{\begin{document}-\end{document}}$.
- **Gestión automática de filesystem:** Crea recursivamente la jerarquía de directorios necesaria si no existe, sin requerir preparación manual del sistema de archivos por parte del usuario.
- **Formato estándar de etiquetas:** Todas las aristas siguen la convención universal “*entrada/salida*” en modo matemático, alineándose con notación de libros de texto y literatura especializada en teoría de autómatas.

7.1 Ejemplos de uso de la función ExportMaquinaToTikZ

A continuación se presentan dos ejemplos representativos que ilustran el uso de **ExportMaquinaToTikZ** en escenarios prácticos. El ejemplo 56 demuestra la exportación de una máquina de 5 estados con personalización de parámetros visuales, incluyendo activación de rejilla y ejes coordenados para referencia espacial. El ejemplo 57 muestra el uso de selección aleatoria de colores para estados y transiciones, técnica útil cuando se desea generar múltiples diagramas con variaciones estéticas sin especificar manualmente cada color.

En ambos casos se utilizan las funciones auxiliares **MaquinaRandom** para generar máquinas de prueba y **MaquinaToDiagrama** que despliega el diagrama de transición de cada máquina en **Mathematica**. Estos comandos pertenecen al paquete “*VilCretas*”. Además, se ha empleado la instrucción **CompiladorTex** para compilar automáticamente desde **Wolfram** los documentos $\text{\LaTeX}$ resultantes.

Ejemplo 56 Máquina pseudoaleatoria con cinco estados

```
maqu1 = MaquinaRandom[5, 3, 2, componentes → True][[2]];
MaquinaToDiagrama[maqu1[[1]], maqu1[[2]], maqu1[[3]], maqu1[[4]],
maqu1[[5]]]
ExportMaquinaToTikZ[maqu1[[1]], maqu1[[2]], maqu1[[3]], maqu1[[4]],
maqu1[[5]], "ExportMaquinaToTikZ1.tex", "StateSize" → 1,
"NodeDistance" → 5, "StateColor" → "Turquoise",
"ArrowThickness" → True, "LayoutStyle" → "spring",
"TransitionColor" → "Salmon", "ShowGrid" → True,
"ShowAxes" → True];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

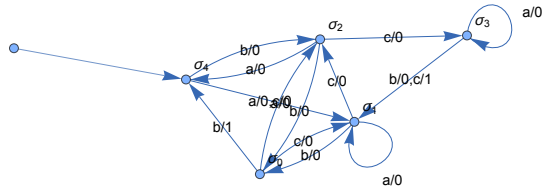
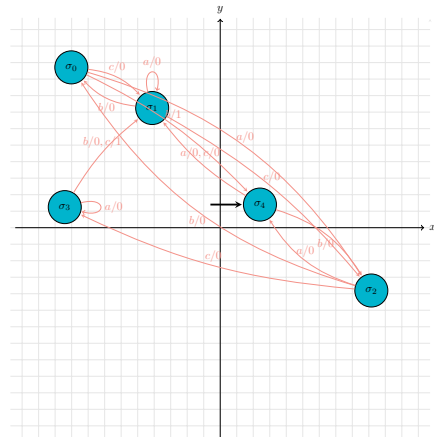
En **Mathematica** se muestra como salida la gráfica de la subfigura 7.1a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

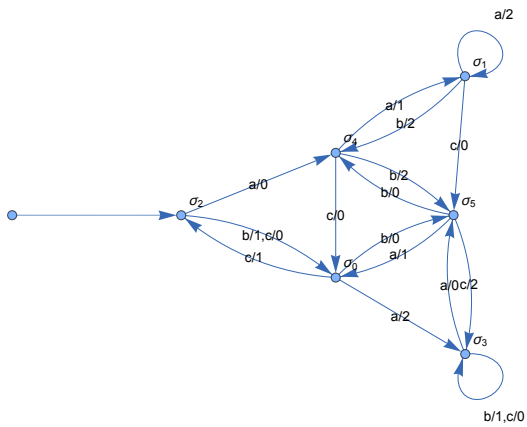
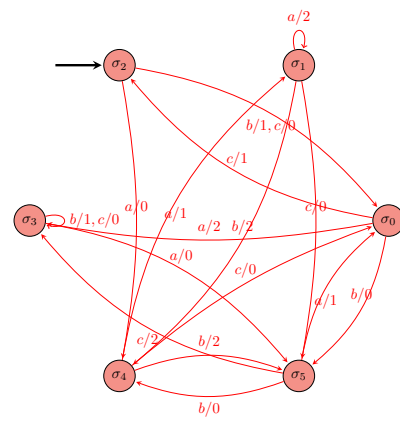
:

PDF generado: /Users/.../Downloads/ExportMaquinaToTikZ1/ExportMaquinaToTikZ1.pdf
/Users/.../Downloads/ExportMaquinaToTikZ1/ExportMaquinaToTikZ1.pdf

El archivo *ExportMaquinaToTikZ1.tex* (guardado en *Downloads/ExportMaquinaToTikZ1*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 7.1b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 7.1: Gráficas del ejemplo 56(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 7.2: Gráficas del ejemplo 57**Ejemplo 57 Máquina pseudoaleatoria con seis estados**

```

maqu2 = MaquinaRandom[6, 3, 3, componentes -> True][[2]];
MaquinaToDiagrama[maqu2[[1]], maqu2[[2]], maqu2[[3]], maqu2[[4]],
maqu2[[5]]]
ExportMaquinaToTikZ[maqu2[[1]], maqu2[[2]], maqu2[[3]], maqu2[[4]],
maqu2[[5]], "ExportMaquinaToTikZ2.tex", "StateSize" -> 0.5,
"NodeDistance" -> 4.2,
"StateColor" ->
RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
"pink", "cyan", "magenta", "yellow", "black", "white", "gray",
"Red", "Blue", "Green", "Violet", "Gray", "LightGray", "Yellow",
"Emerald", "Turquoise", "Lavender", "Peach", "Salmon",
"lightblue", "lightgreen", "lightyellow", "lightgray", "darkgray"}],
"ArrowThickness" -> True,
"TransitionColor" ->
RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
"pink", "cyan", "magenta", "yellow", "black", "white", "gray",
"Red", "Blue", "Green", "Violet", "Gray", "LightGray", "Yellow",

```

```
"Emerald", "Turquoise", "Lavender", "Peach", "Salmon",
"lightblue", "lightgreen", "lightyellow", "lightgray", "darkgray"}]];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** la salida obtenida, corresponde a la gráfica de la subfigura 7.2a y a los mensajes de texto:

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportMaquinaToTikZ2/ExportMaquinaToTikZ2.pdf
/Users/.../Downloads/ExportMaquinaToTikZ2/ExportMaquinaToTikZ2.pdf

El file *ExportMaquinaToTikZ2.tex* (guardado en *Downloads/ExportMaquinaToTikZ2*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 7.2b.

Para más ejemplos

En el archivo: *7. Ejemplos ExportMaquinaToTikZ.nb*, disponible en el enlace de descarga del paquete **Mathematica** (ver página [xiv](#)) se ofrecen al lector otros ejemplos adicionales de utilización del comando **ExportMaquinaToTikZ**.

Función `ExportAutomataToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Exportar autómatas finitos deterministas (DFA) y no deterministas (NFA).
- ▶ Visualizar transiciones y estados de aceptación.
- ▶ Generar diagramas para teoría de lenguajes formales y compiladores.

Los autómatas finitos -determinísticos (DFA) y no determinísticos (NFA)- constituyen el modelo computacional fundamental para reconocimiento de lenguajes regulares, con aplicaciones directas en análisis léxico, validación de patrones, diseño de protocolos y verificación formal. La representación mediante diagramas de estos modelos usando grafos con estados diferenciados (inicial y de aceptación) es esencial para la comunicación pedagógica de dichos conceptos y la documentación técnica. Sin embargo, la generación manual de estos diagramas en $\text{\LaTeX}$ resulta particularmente laboriosa debido a la necesidad de distinguir visualmente estados de aceptación mediante círculos dobles y gestionar transiciones potencialmente no determinísticas.

El comando `ExportAutomataToTikZ` del paquete `\text{\LaTeX}` automatiza completamente este proceso, transformando especificaciones algebraicas de autómatas en diagramas `TikZ` que respetan las convenciones gráficas estándar de la teoría de lenguajes formales. La función procesa tanto autómatas determinísticos como no determinísticos de manera unificada, detectando automáticamente la naturaleza de las transiciones y aplicando el tratamiento gráfico correspondiente. A diferencia de `ExportMaquinaToTikZ` que maneja transductores con alfabetos de entrada/salida, esta función se especializa en reconocedores con estados de aceptación explícitos.

La instrucción `ExportAutomataToTikZ` convierte definiciones formales de autómatas finitos en diagramas `TikZ` donde estados de aceptación se distinguen mediante círculos dobles, transiciones no determinísticas se gestionan automáticamente mediante ramificación múltiple y la estructura completa preserva editabilidad tipográfica. El código generado cumple rigurosamente con estándares visuales de la literatura en teoría de autómatas y lenguajes formales.

El documento resultante es un archivo $\text{\LaTeX}$ compilable de clase *standalone*, idéntico en estructura al producido por `ExportMaquinaToTikZ`. El comando `ExportAutomataToTikZ` posee la sintaxis descrita a continuación.

- `ExportAutomataToTikZ[states, alphabet, initialState, transitions, acceptingStates,`

filename, options]: Exporta una especificación formal de autómata finito (determinístico o no determinístico) como diagrama TikZ autocontenido.

Parámetros:

- **states**: Funciona idénticamente a **ExportMaquinaToTikZ**.
- **alphabet**: Alfabeto de entrada como lista finita de símbolos. A diferencia de **ExportMaquinaToTikZ** que requiere alfabetos separados de entrada y salida, los autómatas operan únicamente sobre símbolos de entrada sin producir salidas explícitas.
- **initialState**: Funciona idénticamente a **ExportMaquinaToTikZ**.
- **transitions**: Función de transición representada como lista de tripletas con dos formatos válidos:

- * **Determinístico**: {origen, symbol, destino} donde *destino* es un estado único.
- * **No determinístico**: {origen, symbol, {destino1, destino2, ...}} donde los destinos forman una lista de estados alcanzables simultáneamente.

El sistema valida: (i) pertenencia de origen y destinos a **states**, (ii) pertenencia de símbolos a **alphabet** y (iii) estructura correcta de las tripletas.

- **acceptingStates**: Lista de estados de aceptación (también llamados estados finales). Debe ser un subconjunto de **states**. Estos estados se renderizan con círculos dobles siguiendo la convención estándar. Puede ser lista vacía {} si el autómata no acepta ninguna cadena.
- **filename**: Funciona idénticamente a **ExportMaquinaToTikZ**.
- **options**: Secuencia opcional de pares "clave" -> valor. Las opciones son idénticas a **ExportMaquinaToTikZ** con las siguientes diferencias en valores predeterminados:

Valores predeterminados modificados:

- * "StateSize" -> 0.8 (en lugar de 0.6).
- * "NodeDistance" -> 4 (en lugar de 3).
- * "StateColor" -> "lightblue" (en lugar de "white").
- * "ArrowThickness" -> True (en lugar de False).
- * "TransitionColor" -> "purple" (en lugar de "black").

Todas las demás opciones ("ShowLabels", "AutoColor", "LayoutStyle", "ShowGrid", "ShowAxes") funcionan idénticamente.

Valor de retorno:

Idéntico a **ExportMaquinaToTikZ**: ruta absoluta del archivo .tex o \$Failed con diagnósticos en caso de error.

Capacidades específicas de autómatas:

- **Distinción visual de estados de aceptación**: Los estados en **acceptingStates** se dibujan automáticamente con doble círculo. Si el estado inicial también es de aceptación, combina ambas características visuales.
- **Soporte nativo para no determinismo**: Detecta automáticamente cuando una transición especifica múltiples estados destino (formato {origen, symbol, {dest1, dest2}}) y genera múltiples aristas desde el mismo origen con el mismo símbolo, cada una apuntando a un destino diferente.
- **Formato de etiquetas simplificado**: Las aristas llevan únicamente el símbolo de transición en modo matemático, sin la notación "entrada/salida" propia de las máquinas de estado finito.
- **Consolidación de símbolos**: Si existen múltiples transiciones entre el mismo par de estados con diferentes símbolos, se agrupan en una única arista con etiqueta compuesta "a, b, c", análogo al comportamiento de **ExportMaquinaToTikZ**.

- **Validación especializada:** Verifica que `acceptingStates` sea un subconjunto válido de `states` y que las listas de destinos en transiciones no determinísticas contengan únicamente estados declarados.

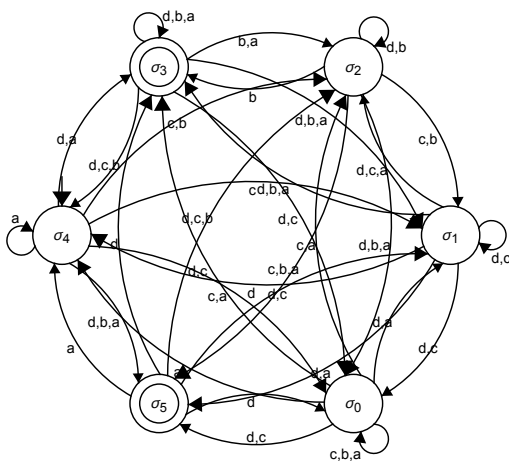
Las capacidades de validación matemática, traducción simbólica, tratamiento de *auto-loops*, resolución de bidireccionalidad, coloración coordinada, sanitización de identificadores, cálculo adaptativo de ventanas, generación de documento autosuficiente y gestión de *filesystem* operan idénticamente a `ExportMaquinaToTikZ`.

8.1 Ejemplos de uso de la función ExportAutomataToTikZ

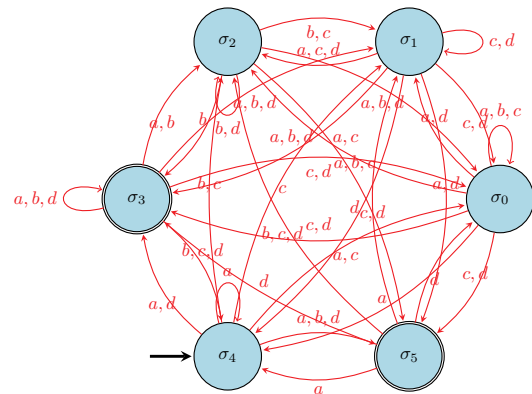
Se comparten dos casos representativos de uso de la instrucción `ExportAutomataToTikZ` con autómatas no determinísticos generados pseudoaleatoriamente.

En el ejemplo 58 se utiliza una selección pseudoaleatoria de colores para estados y transiciones, manteniendo parámetros geométricos fijos que garantizan la legibilidad del diagrama. En el ejemplo 59 las dimensiones físicas "`StateSize`"-"`NodeDistance`" y el algoritmo de disposición ("`LayoutStyle`") varían estocásticamente, además se activa la opción "`AutoColor`" que asigna de forma automática paletas cromáticas diferenciadas a cada estado (característica compartida por `ExportMaquinaToTikZ`).

Ambos ejemplos utilizan las funciones del paquete "*VilCretas*": `AutomataNDRandom` y `ComponentesAutomata` para la construcción pseudoaleatoria de autómatas no determinísticos y la extracción de sus componentes, respectivamente. Además, se recurre al uso de `CompilerTex` con el objetivo de compilar desde **Wolfram Mathematica** los documentos $\text{\LaTeX}$ de salida.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 8.1: Gráficas del ejemplo 58

Ejemplo 58 Autómata no determinístico pseudoaleatorio con seis estados

```
AutomataNDRandom[6, 4]
aut1 = ComponentesAutomata[G, conjuntos -> True];
ExportAutomataToTikZ[aut1[[1]], aut1[[2]], aut1[[3]], aut1[[4]],
aut1[[5]], "ExportAutomataToTikZ1.tex", "StateSize" -> 1.3,
"NodeDistance" -> 3.5,
"StateColor" ->
RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
"pink", "cyan", "magenta", "yellow", "black", "white", "gray",
"Red", "Blue", "Green", "Violet", "Gray", "LightGray", "Yellow",
"Emerald", "Turquoise", "Lavender", "Peach", "Salmon",
```

```

"lightblue", "lightgreen", "lightyellow", "lightgray", "darkgray"}],
"ArrowThickness" → True,
"TransitionColor" →
RandomChoice[{"red", "blue", "green", "orange", "purple", "brown",
"pink", "cyan", "magenta", "yellow", "black", "white", "gray",
"Red", "Blue", "Green", "Violet", "Gray", "LightGray", "Yellow",
"Emerald", "Turquoise", "Lavender", "Peach", "Salmon",
"lightblue", "lightgreen", "lightyellow", "lightgray", "darkgray"}]];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

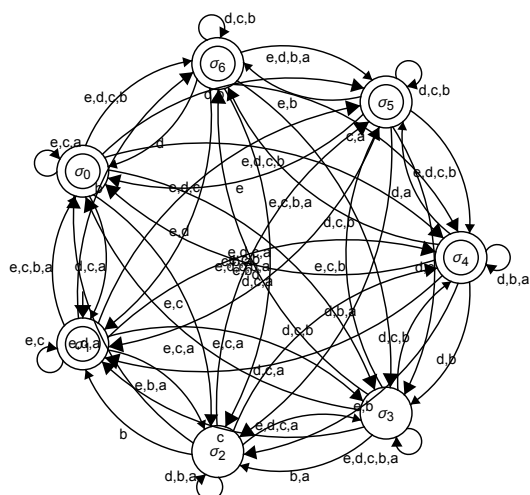
En **Wolfram** se muestra como salida la gráfica de la subfigura 8.1a y los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

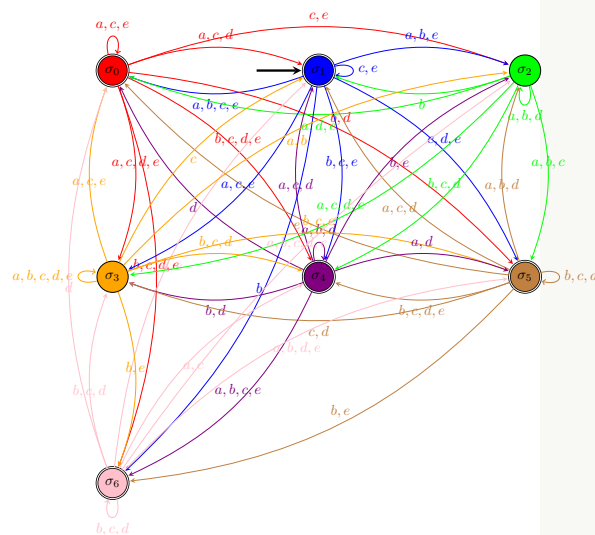
⋮

PDF generado: /Users/.../Downloads/ExportAutomataToTikZ1/ExportAutomataToTikZ1.pdf
 /Users/.../Downloads/ExportAutomataToTikZ1/ExportAutomataToTikZ1.pdf

El archivo *ExportAutomataToTikZ1.tex* (guardado en *Downloads/ExportAutomataToTikZ1*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 8.1b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 8.2: Gráficas del ejemplo 59

Ejemplo 59 Autómata no determinístico pseudoaleatorio con siete estados

```

AutomataNDRandom[7, 5]
aut2 = ComponentesAutomata[G, conjuntos → True];
ExportAutomataToTikZ[aut2[[1]], aut2[[2]], aut2[[3]], aut2[[4]],
aut2[[5]], "ExportAutomataToTikZ2.tex",
"StatesSize" → RandomReal[{0.5, 2.5}],
"NodeDistance" → RandomReal[{1, 5}], "ArrowThickness" → True,
"LayoutStyle" → RandomChoice[{"circular", "grid", "spring"}],
"AutoColor" → True];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Mathematica** se obtiene como salida la gráfica de la subfigura 8.2a y los mensajes de texto:

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: `/Users/.../Downloads/ExportAutomataToTikZ2/ExportAutomataToTikZ2.pdf`
`/Users/.../Downloads/ExportAutomataToTikZ2/ExportAutomataToTikZ2pdf`

El archivo *ExportAutomataToTikZ2.tex* (guardado en *Downloads/ExportAutomataToTikZ2*) facilita el código $\text{\LaTeX}$ editable que produce la gráfica `TikZ` de la subfigura 8.2b.

Para más ejemplos

Si el lector demanda otros ejemplos relacionados con el empleo del comando `ExportAutomataToTikZ`, puede consultar de manera complementaria el archivo: 8. *Ejemplos ExportAutomataToTikZ.nb*, disponible en el enlace de descarga del paquete $\text{\LaTeX}$ (ver página xiv).

9

Función `ExportToTikZ3D`

¿Qué aprenderás en este capítulo?

- ▶ Convertir gráficos 3D de **Mathematica** en código `TikZ` tridimensional.
- ▶ Exportar superficies, curvas espaciales y sólidos con perspectiva.
- ▶ Configurar ángulos de visualización, iluminación y transparencias.
- ▶ Preservar información de profundidad en proyecciones 2D.
- ▶ Generar visualizaciones 3D editables para geometría y cálculo multivariable.

La visualización tridimensional en documentos científicos enfrenta desafíos técnicos particulares cuando se busca integrar gráficos computacionales con tipografía profesional. Mientras **Mathematica** genera visualizaciones 3D en formatos rasterizados o vectoriales estáticos, la producción académica frecuentemente demanda código *TikZ/PGFPlots* nativo que preserve la edición completa, coherencia tipográfica y control sobre cada componente del gráfico tridimensional.

El sistema de exportación 3D del paquete **ViTeX** aborda esta tarea mediante una conversión integral que transforma objetos gráficos tridimensionales de **Wolfram Mathematica** en código `TikZ` funcional. A diferencia de métodos convencionales que requieren ajustes manuales extensos o producen salidas no editables, el comando `ExportToTikZ3D` implementa análisis exhaustivo de superficies paramétricas, resolución algebraica de ecuaciones implícitas, muestreo adaptativo inteligente y gestión automática de límites espaciales para múltiples objetos.

El núcleo de este comando interpreta la estructura interna de funciones de *plotting* tridimensional como `Plot3D`, `ParametricPlot3D`, `ContourPlot3D` o combinaciones mediante `Show`. El proceso de conversión ejecuta múltiples etapas: extracción de especificaciones funcionales, generación de mallas de muestreo optimizadas, resolución de superficies implícitas mediante técnicas algebraicas o numéricas, cálculo unificado de límites espaciales, aplicación de mapas de color perceptualmente uniformes y finalmente la construcción de código `TikZ` estructurado con opciones de renderizado consistentes.

La función `ExportToTikZ3D` transforma especificaciones de gráficos 3D nativos de **Wolfram** en código *TikZ/PGFPlots* tridimensional completamente funcional, creando documentos $\text{\LaTeX}$ compilables que preservan calidad vectorial y editabilidad del gráfico original. A diferencia de exportaciones convencionales que producen imágenes rasterizadas o formatos vectoriales cerrados, `ExportToTikZ3D` ge-

nera código fuente $\text{\LaTeX}$ legible y modificable con cualquier editor de texto.

Esta característica resulta esencial en contextos donde se requiere ajustar ángulos de visualización, mapas de color, transparencias o sistemas de ejes para cumplir con lineamientos editoriales. El resultado de la sentencia `ExportToTikZ3D` es un archivo `.tex` autónomo con clase *standalone* que puede compilarse mediante *pdflatex* sin requerir edición adicional. Veamos la sintaxis de esta instrucción.

90

- `ExportToTikZ3D[plotArgs, filename, grid, color, samples, axisType, colormap, opacity, axisExtension, maxPointsImplicit, textAnnotations]`: Convierte gráficos 3D elaborados en **Wolfram Mathematica** a código $\text{\LaTeX}$ tridimensional y lo exporta como archivo `.tex` compilable.

Parámetros:

- **plotArgs**: Especificación de gráfico 3D válido de **Mathematica**. Acepta funciones de *plotting* tridimensional estándar o listas de múltiples gráficos:
 - * `Plot3D`: `{f[x,y], {x, xmin, xmax}, {y, ymin, ymax}}`.
 - * `ParametricPlot3D` (curva): `{{x[t], y[t], z[t]}, {t, tmin, tmax}}`.
 - * `ParametricPlot3D` (superficie): `{{x[u,v], y[u,v], z[u,v]}, {u, umin, umax}, {v, vmin, vmax}}`.
 - * `ContourPlot3D`:
`{ecuacion == 0, {x, xmin, xmax}, {y, ymin, ymax}, {z, zmin, zmax}}`.
 - * Múltiples gráficos: Lista de cualquiera de los tipos anteriores para composición mediante el uso del comando `Show` nativo de **Wolfram**.
 - **filename**: *String* con el nombre del archivo de salida. Debe incluir la extensión `.tex` (por ejemplo: `"superficie.tex"`). El sistema crea automáticamente una carpeta en *Downloads* con el nombre base del archivo (sin extensión) y coloca el archivo `.tex` dentro de esta carpeta.
 - **grid** (opcional, por defecto `True`): Booleano que controla la visualización de la cuadrícula tridimensional. `True` muestra rejilla mayor en planos coordenados y `False` genera una visualización sin cuadrícula.
 - **color** (opcional, por defecto `"blue"`): Especificación de color principal del gráfico. Acepta múltiples formatos con sintaxis flexible de `TikZ/xcolor`:
 - * Colores básicos: `"black", "blue", "brown", "cyan", "darkgray", "gray", "green", "lightgray", "lime", "magenta", "olive", "orange", "pink", "purple", "red", "teal", "violet", "white", "yellow"`.
 - * Mezclas: `"blue!50!red", "green!70!black", "red!20!blue!80!green"`.
 - * `RGB`: `"RGB{255,100,50}"`.
 - * `HTML`: `"HTML{FF6B35}"`.
- Para múltiples gráficos, el sistema asigna automáticamente colores secuenciales: `"blue", "red", "green!70!black", "orange", "purple", "brown", "pink", "gray", "cyan", "magenta"`.
- **samples** (opcional, por defecto 30): Entero que controla la resolución de malla para superficies. Rango recomendado: 10-100. Valores altos generan mayor resolución visual pero producen archivos más grandes y tiempos de compilación extendidos. El sistema ajusta automáticamente este número para múltiples gráficos evitando una saturación en la memoria.
 - **axisType** (opcional, por defecto `"box"`): *String* que especifica el estilo del sistema de ejes coordenados:
 - * `"box"`: Ejes en forma de caja completa tridimensional.
 - * `"center"`: Ejes centrados en el origen con flechas direccionales.
 - * `"none"`: Sin ejes visibles.

- **colormap** (opcional, por defecto **"viridis"**): *String* que especifica el mapa de colores para superficies con gradientes de altura. Acepta exactamente 11 opciones predefinidas:
 - * **"viridis"**: Verde-azul-violeta (perceptualmente uniforme, recomendado para datos científicos).
 - * **"cool"**: Cian a magenta (transición frío-caliente).
 - * **"hot"**: Negro-rojo-amarillo-blanco (mapa de calor clásico).
 - * **"jet"**: Azul-cian-amarillo-rojo (colorido, no perceptualmente uniforme).
 - * **"hsv"**: Espectro *HSV* completo circular.
 - * **"spring"**: Magenta-amarillo (tonalidades primaverales).
 - * **"summer"**: Verde-amarillo (tonalidades veraniegas).
 - * **"autumn"**: Rojo-amarillo (tonalidades otoñales).
 - * **"winter"**: Azul-verde (tonalidades invernales).
 - * **"gray"**: Escala de grises negro-blanco.
 - * **"blackwhite"**: Alias para escala de grises.
- **opacity** (opcional, por defecto **1.0**): Número real en el intervalo $[0.0, 1.0]$ que controla la transparencia del gráfico, **1.0** representa opacidad completa y **0.0** transparencia total. Valores intermedios: **0.3** (transparente), **0.5** (semi-transparente), **0.7** (poco transparente).
- **axisExtension** (opcional, por defecto **0.0**): Número real no negativo que especifica una extensión adicional de los límites de los ejes como fracción del rango de datos. Solo es efectivo con **axisType** -> **"center"**. Por ejemplo: **0.1** extiende los ejes un 10% más allá de los datos para una visualización completa de las flechas direccionales.
- **maxPointsImplicit** (opcional, por defecto **8000**): Entero ≥ 100 que especifica el máximo número de vértices para superficies implícitas generadas mediante **ContourPlot3D**. Controla el balance entre la calidad visual y el rendimiento de compilación. Se distribuye automáticamente entre múltiples gráficos implícitos.
- **textAnnotations** (opcional, por defecto **{}**): Lista de anotaciones de texto tridimensionales que se añaden al gráfico. Acepta los siguientes formatos:
 - * Lista vacía **{}**: Sin anotaciones adicionales.
 - * Anotación básica: **{"texto", {x, y, z}}** coloca **"texto"** en coordenadas 3D.
 - * Anotación con formato: **{"texto", {x, y, z}, {"opcion", "valor"}, ...}**.
 - * Múltiples anotaciones:
 - {{"texto1", {x1, y1, z1}}, {"texto2", {x2, y2, z2}, opciones}, ...}**.

Opciones de formato disponibles para anotaciones:

- * **{"color", "red"}**: Color del texto (acepta todos los colores listados anteriormente).
- * **{"size", "\\large"}**: Tamaño del texto. Comandos $\text{\LaTeX}$ válidos: **\tiny**, **\scriptsize**, **\footnotesize**, **\small**, **\normalsize**, **\large**, **\Large**, **\huge**, **\Huge**.
- * **{"anchor", "north"}**: Anclaje del texto respecto a las coordenadas. Opciones: **"north"**, **"south"**, **"east"**, **"west"**, **"north east"**, **"north west"**, **"south east"**, **"south west"**, **"center"**.
- * **{"rotate", "45"}**: Rotación en grados (0-360).

Retorna en **Mathematica**:

String con la ruta completa del archivo *.tex* exportado en el subdirectorío creado dentro de *Downloads* o **\$Failed** si ocurre un error durante el proceso de conversión.

Funcionalidades especiales:

- Detección automática del tipo de gráfico (explícito, paramétrico, implícito) sin intervención del usuario.

- Resolución algebraica inteligente de ecuaciones implícitas cuando es factible, con alternativa de muestreo por contornos.
- Detección especializada de geometrías cilíndricas (ecuaciones independientes de z) con muestreo optimizado.
- Cálculo unificado de límites espaciales para composiciones de múltiples objetos tridimensionales.
- Ajuste automático de resolución de malla para evitar saturación de memoria en escenas complejas.
- Muestreo aleatorio cuando el número de vértices excede los límites especificados, preservando la distribución espacial.
- Formateo numérico adaptativo según la magnitud de coordenadas (manejo robusto de valores extremos).
- Generación de código $\text{\LaTeX}$ estructurado con sintaxis *PGFPlots* estándar para compatibilidad universal.
- Creación automática de estructura de carpetas en *Downloads* con nomenclatura consistente.
- Generación de documento $\text{\LaTeX}$ completo y compilable con clase *standalone* incluyendo todos los paquetes requeridos.
- Validación exhaustiva de parámetros con mensajes de error descriptivos en español.
- Soporte completo para transparencias y mapas de color perceptualmente uniformes.

9.1 Ejemplos de uso de la función ExportToTikZ3D

La versatilidad de **ExportToTikZ3D** se manifiesta plenamente al trabajar con geometrías tridimensionales de complejidad variable. Esta sección presenta casos prácticos que ilustran las capacidades del sistema: desde superficies explícitas simples hasta composiciones multi-objeto que combinan ecuaciones implícitas, parametrizaciones y curvas espaciales. Cada ejemplo demuestra diferentes aspectos técnicos de la función, incluyendo el manejo de múltiples tipos de gráficos simultáneos, la aplicación de mapas de color perceptualmente uniformes, el control de transparencias y la personalización en el sistemas de ejes.

Los ejemplos incorporan deliberadamente parámetros aleatorios para demostrar la robustez del comando ante configuraciones arbitrarias. En aplicaciones reales, estos valores se especificarían según requerimientos estéticos o científicos particulares. El rango de geometrías abordadas incluye superficies trigonométricas, curvas helicoidales, toros implícitos, familias de esferas concéntricas, ondas *gaussianas*, cuádricas rotadas arbitrariamente (hiperbolooides, paraboloides, conos) y composiciones heterogéneas que yuxtaponen hasta cuatro objetos geométricos simultáneos.

Los casos presentados evalúan sistemáticamente diferentes aspectos del sistema de conversión: algoritmos de resolución algebraica para ecuaciones implícitas, generación de mallas uniformes para superficies paramétricas, detección de geometrías cilíndricas con optimización de muestreo, cálculo unificado de límites espaciales para escenas multi-objeto, distribución automática del presupuesto de puntos entre superficies implícitas concurrentes y manejo de expresiones transcendentales complejas con evaluación numérica robusta. La diversidad topológica de los ejemplos (variedades abiertas, cerradas, orientables, con singularidades) demuestra la generalidad del enfoque implementado en **ExportToTikZ3D**.

La función auxiliar **CompiladorTex** que aparece al final de cada ejemplo, tal y como se hizo en secciones anteriores, automatiza la compilación del archivo *.tex* obtenido mediante **ExportToTikZ3D**, abriendo inmediatamente el resultado en el editor especificado.

Ejemplo 60 Gráfica Plot3D

```

Plot3D[Sin[x]*Cos[y], {x, -Pi, Pi}, {y, -Pi, Pi},
PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]]];
Row[{"grid=", grid = RandomChoice[{True, False}]]],
Row[{"color=",
  color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
    "gray", "green", "lightgray", "lime", "magenta", "olive",
    "orange", "pink", "purple", "red", "teal", "violet", "white",
    "yellow", "blue!50!red", "green!70!black",
    "red!20!blue!80!green", "yellow!75!red"}]]],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]]],
Row[{"estiloejes=",
  estiloejes = RandomChoice[{"box", "center", "none"}]]],
Row[{"mapacolor=",
  mapacolor =
  RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
    "summer", "autumn", "winter", "gray", "blackwhite"}]]],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]]],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]]],
Row[{"puntossuperimpli=",
  puntossuperimpli = RandomInteger[{8000, 13000}]]],
Row[{"texto=",
  texto = {{f(x,y)=sen(x)cos(y)", {1, 1, 1}, {"color", colortexto[]},
    {"size", sizetexto}, {"anchor",
    anclajetexto}, {"rotate", rotatexto}}}}]]
ExportToTikZ3D[{Sin[x]*Cos[y], {x, -Pi, Pi}, {y, -Pi, Pi}},
  "ExportToTikZ3D1.tex", grid, color, resolucion, estiloejes,
  mapacolor, opacidad, extensionejes, puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" ->"TeXstudio"]

```

En **Mathematica** se muestra como salida la gráfica de la subfigura 9.1a y los mensajes siguientes:

```

{grid= True, color= brown, resolucion= 57, estiloejes= none, mapacolor= blackwhite,
opacidad= 0.617355, extensionejes= 0.283539, puntossuperimpli= 8757, texto=
{{$f(x,y) = sen(x)cos(y)$, {1, 1, 1}, {{color, gray}, {size, \Huge}, {anchor,
center}, {rotate, 324.018}}}}

```

Usando editores ya detectados: 4 disponibles

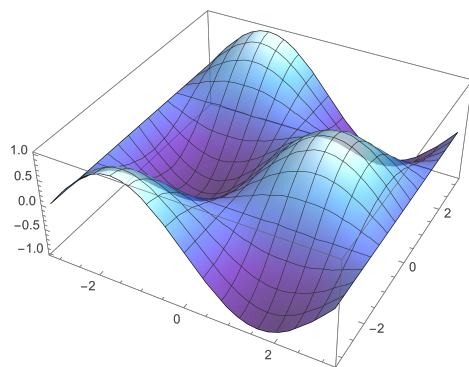
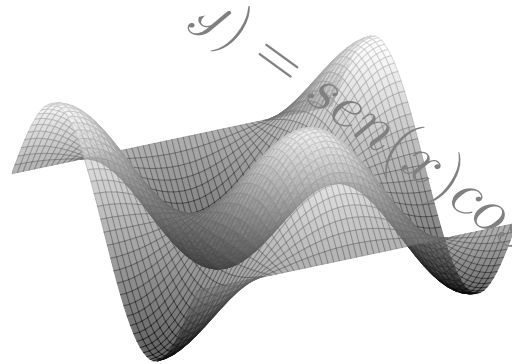
:

```

PDF generado: /Users/.../Downloads/ExportToTikZ3D1/ExportToTikZ3D1.pdf
/Users/.../Downloads/ExportToTikZ3D1/ExportToTikZ3D1.pdf

```

Lo contenido en la llave de este *Out* corresponde a los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D1.tex* (guardado en *Downloads/ExportToTikZ3D1*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.1b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 9.1: Gráficas del ejemplo 60

Ejemplo 61 Gráfica *ParametricPlot3D*

```

ParametricPlot3D[Cos[t], Sin[t], t/4], {t, 0, 8*Pi}]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"color=",
    color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
      "gray", "green", "lightgray", "lime", "magenta", "olive",
      "orange", "pink", "purple", "red", "teal", "violet", "white",
      "yellow", "blue!50!red", "green!70!black",
      "red!20!blue!80!green", "yellow!75!red"}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"mapacolor=",
    mapacolor =
    RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
      "summer", "autumn", "winter", "gray", "blackwhite"}]}],
  Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"puntossuperimpli=",
    puntossuperimpli = RandomInteger[{8000, 13000}]}],
  Row[{"texto=",
    texto = {"Curva paramétrica", {0, 0, 0}, {"color", colortexto[]],
      {"size", sizetexto}, {"anchor",
        anclajetexto}, {"rotate", rotatexto}}]}]}
ExportToTikZ3D[{Cos[t], Sin[t], t/4}, {t, 0, 8*Pi}],
"ExportToTikZ3D2.tex", grid, color, resolucion, estiloejes,
mapacolor, opacidad, extensionejes, puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se despliega como salida la gráfica de la subfigura 9.2a y los mensajes mostrados a continuación:

```
{grid= False, color= brown, resolucion= 24, estiloejes= none, mapacolor= gray,
opacidad= 0.855118, extensionejes= 0.118306, puntossuperimpli= 9249, texto= {{Curva
paramétrica, {0, 0, 0}, {{color, red}, {size, \footnotesize}, {anchor, east},
{rotate, 318.608}}}}
```

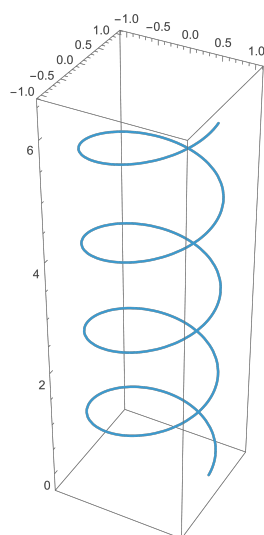
Usando editores ya detectados: 4 disponibles

⋮

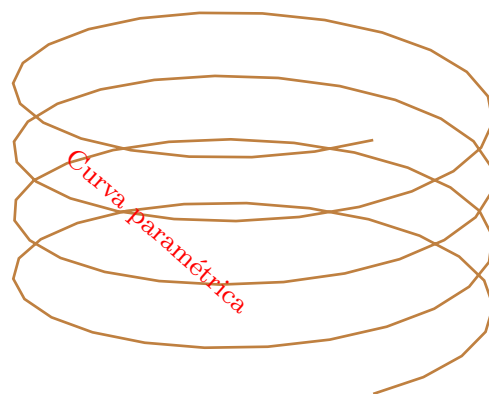
PDF generado: /Users/.../Downloads/ExportToTikZ3D2/ExportToTikZ3D2.pdf

/Users/.../Downloads/ExportToTikZ3D2/ExportToTikZ3D2.pdf

La llave de este *Out* explicita los argumentos pseudoaleatorios empleados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D2.tex* (guardado en *Downloads/ExportToTikZ3D2*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.2b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 9.2: Gráficas del ejemplo 61

Ejemplo 62 Toro, superficie senoidal y curva espiral

```
Show[ContourPlot3D[(x^2 + y^2 + z^2 + 4 - 1)^2 =
4*4*(x^2 + y^2), {x, -4, 4}, {y, -4, 4}, {z, -2, 2},
ContourStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
Plot3D[Sin[x]*Cos[y], {x, -Pi, Pi}, {y, -Pi, Pi},
PlotStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{2*Cos[t], 2*Sin[t], Sin[2*t]}, {t, 0, 4*Pi}]]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
```

```

RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
  "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
  color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
    "gray", "green", "lightgray", "lime", "magenta", "olive",
    "orange", "pink", "purple", "red", "teal", "violet", "white",
    "yellow", "blue!50!red", "green!70!black",
    "red!20!blue!80!green", "yellow!75!red"}]}],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estiloejes=",
  estiloejes = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapacolor=",
  mapacolor =
  RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
    "summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
Row[{"puntossuperimpli=",
  puntossuperimpli = RandomInteger[{8000, 13000}]}],
Row[{"texto=",
  texto = {{"Toro, senoidal y espiral", {0, 0, 0}, {"color",
    colortexto[]}, {"size", sizetexto}, {"anchor",
    anclajetexto}, {"rotate", rotatexto}}]}]}]
ExportToTikZ3D[{{(x^2 + y^2 + z^2 + 4 - 1)^2 =
  4*4*(x^2 + y^2), {x, -4, 4}, {y, -4, 4}, {z, -2, 2}}, {Sin[x]*
  Cos[y], {x, -Pi, Pi}, {y, -Pi, Pi}}, {{2*Cos[t], 2*Sin[t],
  Sin[2*t]}, {t, 0, 4*Pi}}}, "ExportToTikZ3D3.tex", grid, color,
resolucion, estiloejes, mapacolor, opacidad, extensionejes,
puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram** se produce como salida la gráfica de la subfigura 9.3a y los mensajes:

```

{grid= False, color= orange, resolucion= 11, estiloejes= center, mapacolor= jet,
opacidad= 0.608574, extensionejes= 0.300004, puntossuperimpli= 8187, texto= {{Toro,
senoidal y espiral, {0, 0, 0}, {{color, pink}, {size,\Huge}, {anchor, south west},
{rotate, 101.728}}}}}

```

Usando editores ya detectados: 4 disponibles

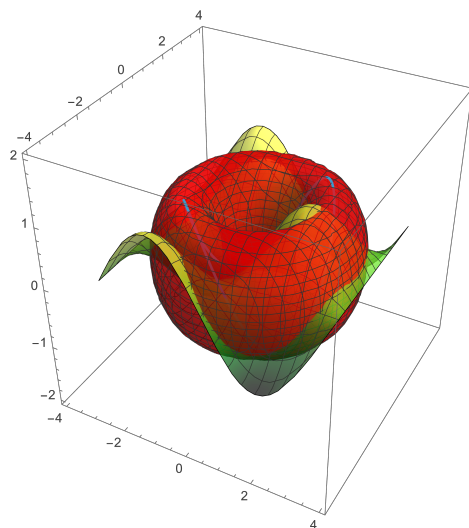
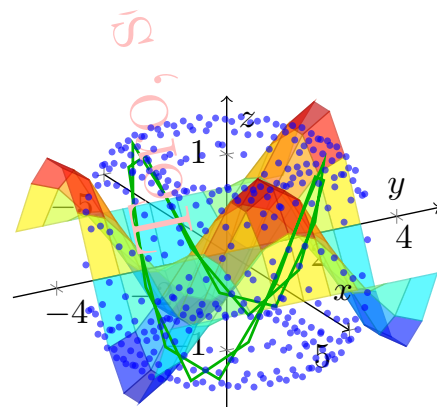
:

```

PDF generado: /Users/.../Downloads/ExportToTikZ3D3/ExportToTikZ3D3.pdf
/Users/.../Downloads/ExportToTikZ3D3/ExportToTikZ3D3.pdf

```

Lo contenido en la llave expone los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D3.tex* (guardado en *Downloads/ExportToTikZ3D3*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.3b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 9.3: Gráficas del ejemplo 62

Como se aprecia en la sufigura 9.3b del ejemplo anterior, cuando se recibe una ecuación implícita (`ContourPlot3D`) el algoritmo empleado en la instrucción `ExportToTikZ3D` genera la gráfica de contorno correspondiente mediante una secuencia significativa de puntos, controlados por medio del parámetro `puntossuperimpli`. Entre mayor sea el número de puntos utilizados se logrará una mayor densidad en la gráfica TikZ respectiva, sin embargo, el lector debe tener claro que esto tendrá una incidencia directa sobre el tiempo de compilación y una posible saturación de la memoria en el ordenador.

Ejemplo 63 Esfera, paraboloides y hélice

```
Show[ContourPlot3D[
  x^2 + y^2 + z^2 == 25, {x, -6, 6}, {y, -6, 6}, {z, -6, 6},
  ContourStyle -> Opacity[0.7],
  ColorFunction -> RandomChoice[ColorData["Gradients"]]],
Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}, PlotStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{3*Cos[t], 3*Sin[t], t/2}, {t, -4*Pi, 4*Pi}]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
  color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
    "gray", "green", "lightgray", "lime", "magenta", "olive",
```

```

"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]]],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estiloEjes=",
  estiloEjes = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapaColor=",
  mapaColor =
  RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
    "summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensionEjes=", extensionEjes = RandomReal[{0.1, 0.4}]}],
Row[{"puntosSuperimprimi=",
  puntosSuperimprimi = RandomInteger[{8000, 13000}]}],
Row[{"texto=",
  texto = {{ "Esfera, paraboloides y hélice", {0, 0, 0},
    {"color", colortexto[]}, {"size", sizetexto[]}, {"anchor",
    anclaJetexto[]}, {"rotate", rotatetexto[]}}}],
ExportToTikZ3D[{ {x^2 + y^2 + z^2 == 25, {x, -6, 6}, {y, -6,
  6}, {z, -6, 6}}, {x^2 + y^2, {x, -2, 2}, {y, -2, 2}}, {{3*Cos[t],
  3*Sin[t], t/2}, {t, -4*Pi, 4*Pi}}}, "ExportToTikZ3D4.tex", grid,
color, resolucion, estiloEjes, mapaColor, opacidad, extensionEjes,
puntosSuperimprimi, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram Mathematica** se obtiene como salida la gráfica de la subfigura 9.4a y los mensajes:

```

{grid= False, color= blue!50!red, resolucion= 58, estiloEjes= box, mapaColor=
spring, opacidad= 0.667806, extensionEjes= 0.140697, puntosSuperimprimi= 12645,
texto= {{Esfera, paraboloides y hélice, {0, 0, 0}, {{color, red}, {size, \Large},
{anchor, north east}, {rotate, 239.07}}}}

```

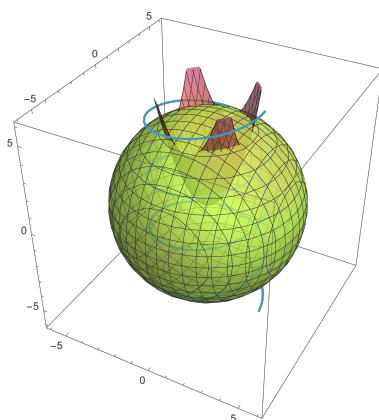
Usando editores ya detectados: 4 disponibles

⋮

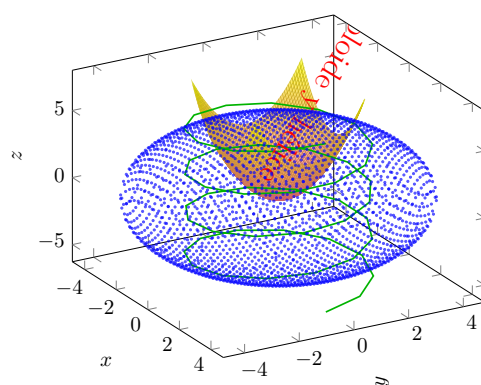
PDF generado: /Users/.../Downloads/ExportToTikZ3D4/ExportToTikZ3D4.pdf

/Users/.../Downloads/ExportToTikZ3D4/ExportToTikZ3D4.pdf

Lo contenido en la llave del *Out* especifica los argumentos pseudoaleatorios pasados al comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D4.tex* (guardado en *Downloads/ExportToTikZ3D4*) facilita el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.4b.

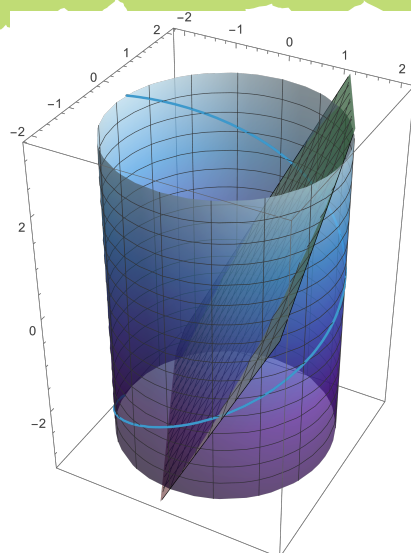
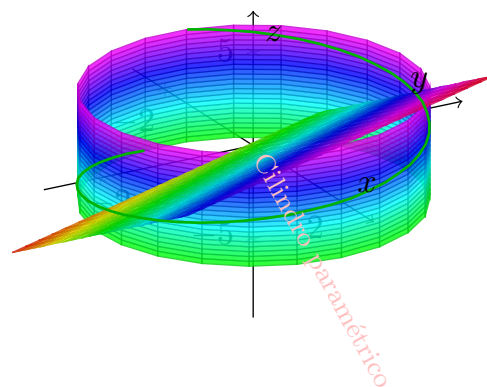


(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 9.4: Gráficas del ejemplo 63

(a) Gráfica de **Mathematica**

(b) Gráfica Tikz

Figura 9.5: Gráficas del ejemplo 64

Ejemplo 64 Cilindro, plano y hélice en cilindro

```
Show[ParametricPlot3D[{2 Cos[u], 2 Sin[u], v}, {u, 0, 2 Pi}, {v, -3,
  3}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
Plot3D[2*x + y - 1, {x, -2, 2}, {y, -2, 2}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{2*Cos[t], 2*Sin[t], t}, {t, -3, 3}]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
  color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
    "gray", "green", "lightgray", "lime", "magenta", "olive",
    "orange", "pink", "purple", "red", "teal", "violet", "white",
    "yellow", "blue!50!red", "green!70!black",
    "red!20!blue!80!green", "yellow!75!red"}]}],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estilo", estilo = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapacolor=",
  mapacolor =
    RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
    "summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
Row[{"puntossuperimpli=",
```

```

puntosuperimpli = RandomInteger[{8000, 13000}]],
Row[{"texto=",
texto = {{Cilindro paramétrico u otras, {0, 0, 0},
{"color", colortexto[]}, {"size", sizetexto}, {"anchor",
anclajetexto}, {"rotate", rotatexto}}}}],
ExportToTikZ3D[{{2 Cos[u], 2 Sin[u], v}, {u, 0, 2 Pi}, {v, -3,
3}}, {2*x + y - 1, {x, -2, 2}, {y, -2, 2}}, {{2*Cos[t], 2*Sin[t],
t}, {t, -3, 3}}, "ExportToTikZ3D5.tex", grid, color,
resolucion, estíloesjes, mapacolor, opacidad, extensionesjes,
puntosuperimpli, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** la salida desplegada muestra la gráfica de la subfigura 9.5a y los mensajes expuestos a continuación:

```

{grid= False, color= green!70!black, resolucion= 24, estíloesjes= center, mapacolor=
hsv, opacidad= 0.817202, extensionesjes= 0.184236, puntosuperimpli= 11018, texto=
{{Cilindro paramétrico u otras, {0, 0, 0}, {{color, pink}, {size, \footnotesize},
{anchor, west}, {rotate, 298.58}}}}

```

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ3D5/ExportToTikZ3D5.pdf

/Users/.../Downloads/ExportToTikZ3D5/ExportToTikZ3D5.pdf

La llave de la salida anterior comparte los argumentos pseudoaleatorios empleados dentro del comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D5.tex* (guardado en *Downloads/ExportToTikZ3D5*) contiene el código **TeX** editable que produce la gráfica **Tikz** de la subfigura 9.5b.

Ejemplo 65 Paraboloide, cilindro y toro paramétrico

```

Show[ParametricPlot3D[{u*Cos[v], u*Sin[v], u^2}, {u, 0, 2}, {v, 0,
2*Pi}, PlotStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{Cos[v], Sin[v], u}, {u, -2, 2}, {v, 0, 2*Pi},
PlotStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{(2 + Cos[u])*Cos[v], (2 + Cos[u])*Sin[v],
Sin[u]}, {u, 0, 2*Pi}, {v, 0, 2*Pi}, PlotStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]],
PlotRange -> All]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\tiny", "\scriptsize", "\footnotesize", "\small",
"\normalsize", "\large", "\Large", "\huge", "\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
"gray", "green", "lightgray", "lime", "magenta", "olive",

```



```

"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]]],
Row[{"resolucion=", resolucion = RandomInteger[{10, 50}]}],
Row[{"estiloeyes=",
estiloeyes = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapacolor=",
mapacolor =
RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
"summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensioneyes=", extensioneyes = RandomReal[{0.1, 0.4}]}],
Row[{"puntossuperimpli=",
puntossuperimpli = RandomInteger[{8000, 13000}]}],
Row[{"texto=",
texto = {"Paraboloide u otras", {1, 1,
1}, {"color", colortexto[]}, {"size", sizetexto}, {"anchor",
anclajetexto}, {"rotate", rotatetexto}}]}],
ExportToTikZ3D[{{u*Cos[v], u*Sin[v], u^2}, {u, 0, 2}, {v, 0,
2*Pi}}, {{Cos[v], Sin[v], u}, {u, -2, 2}, {v, 0,
2*Pi}}, {{(2 + Cos[u])*Cos[v], (2 + Cos[u])*Sin[v], Sin[u]}, {u,
0, 2*Pi}, {v, 0, 2*Pi}}}, "ExportToTikZ3D6.tex", grid, color,
resolucion, estiloeyes, mapacolor, opacidad, extensioneyes,
puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se muestra en el *Out* la gráfica de la subfigura 9.6a y los mensajes:

```

{grid= True, color= purple, resolucion= 33, estiloeyes= none, mapacolor= jet,
opacidad= 0.463439, extensioneyes= 0.350844, puntossuperimpli= 10597, texto=
{{Paraboloide u otras, {1, 1, 1}, {{color, red}, {size, \small}, {anchor, south
east}, {rotate, 216.833}}}}}

```

Usando editores ya detectados: 4 disponibles

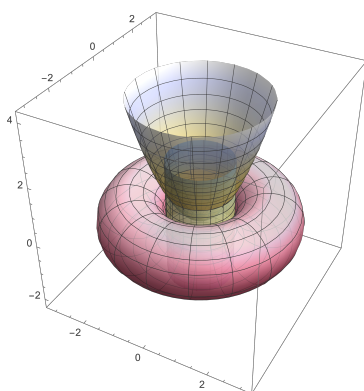
⋮

```

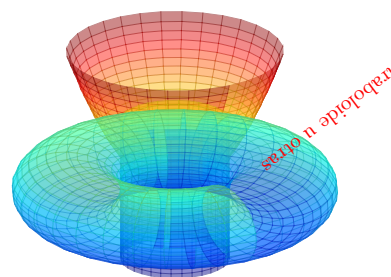
PDF generado: /Users/.../Downloads/ExportToTikZ3D6/ExportToTikZ3D6.pdf
/Users/.../Downloads/ExportToTikZ3D6/ExportToTikZ3D6.pdf

```

Lo contenido en la llave ofrece los argumentos pseudoaleatorios usados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D6.tex* (guardado en *Downloads/ExportToTikZ3D6*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.6b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 9.6: Gráficas del ejemplo 65

Ejemplo 66 *Lemniscata*, curva cúbica, superficie hiperbólica y curva *Lissajous*

```

Show[ContourPlot3D[(x^2 + y^2)^2 = z^2*(x^2 - y^2), {x, -2,
  2}, {y, -2, 2}, {z, -1, 1}, ContourStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{u^3, u^2, u}, {u, -2, 2}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{Sinh[u]*Cos[v], Sinh[u]*Sin[v], v}, {u, -1, 1}, {v,
  0, 2*Pi}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{t, Sin[5*t], Cos[3*t]}, {t, -2*Pi, 2*Pi}]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"color=",
    color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
      "gray", "green", "lightgray", "lime", "magenta", "olive",
      "orange", "pink", "purple", "red", "teal", "violet", "white",
      "yellow", "blue!50!red", "green!70!black",
      "red!20!blue!80!green", "yellow!75!red"}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
  Row[{"estiloeyes=",
    estiloeyes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"mapacolor=",
    mapacolor =
      RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
        "summer", "autumn", "winter", "gray", "blackwhite"}]}],
  Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
  Row[{"extensioneyes=", extensioneyes = RandomReal[{0.1, 0.4}]}],
  Row[{"puntossuperimpli=",
    puntossuperimpli = RandomInteger[{8000, 13000}]}],
  Row[{"texto=",
    texto = {{Lemniscata 3D, {-0.5, 0.5, 1}, {"color", colortexto[]},
      {"size", sizetexto}, {"anchor",
        anclajetexto}, {"rotate", rotatexto}}}}}
ExportToTikZ3D[{(x^2 + y^2)^2 = z^2*(x^2 - y^2), {x, -2, 2}, {y, -2,
  2}, {z, -1, 1}}, {{u^3, u^2, u}, {u, -2, 2}}, {{Sinh[u]*Cos[v],
  Sinh[u]*Sin[v], v}, {u, -1, 1}, {v, 0, 2*Pi}}, {{t, Sin[5*t],
  Cos[3*t]}, {t, -2*Pi, 2*Pi}}, "ExportToTikZ3D7.tex", grid,
  color, resolucion, estiloeyes, mapacolor, opacidad, extensioneyes,
  puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" ->"TeXstudio"]

```

En **Wolfram Mathematica** la salida de este código crea la gráfica de la subfigura 9.7a y los mensajes siguientes:

```

{grid= True, color= purple, resolucion= 18, estiloeyes= none, mapacolor= summer,
opacidad= 0.652109, extensioneyes= 0.24069, puntossuperimpli= 9095, texto=
{{Lemniscata 3D, {-0.5, 0.5, 1}, {{color, lime}, {size, \Huge}, {anchor, north
west}, {rotate, 81.0604}}}}}

```

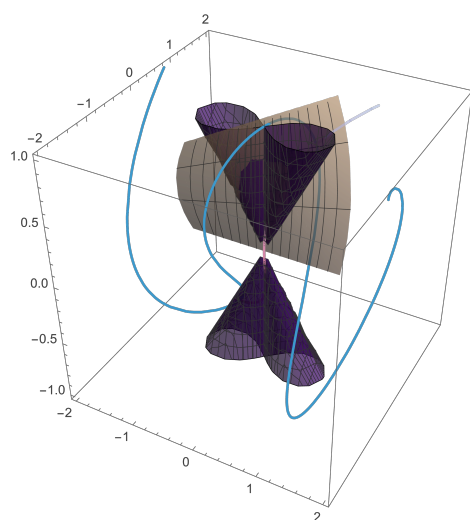
Usando editores ya detectados: 4 disponibles

⋮

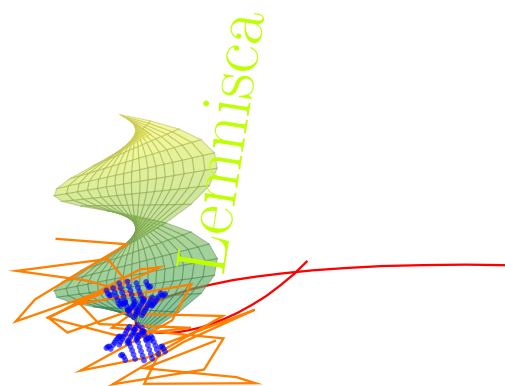
PDF generado: /Users/.../Downloads/ExportToTikZ3D7/ExportToTikZ3D7.pdf

/Users/.../Downloads/ExportToTikZ3D7/ExportToTikZ3D7.pdf

Lo contenido en la llave corresponde a los argumentos pseudoaleatorios utilizados en el comando `ExportToTikZ3D`. El archivo *ExportToTikZ3D7.tex* (guardado en *Downloads/ExportToTikZ3D7*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.7b. Cabe destacar que en este ejemplo, la similitud entre la gráfica de **Mathematica** y la gráfica $\text{\TikZ}$ no es tan apreciable, **se advierte que la herramienta no es perfecta** y en ocasiones demandará la necesidad de realizar cambios post edición mediante el archivo *.tex* generado.

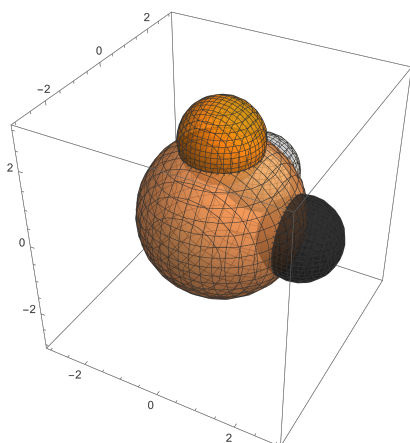


(a) Gráfica de **Mathematica**

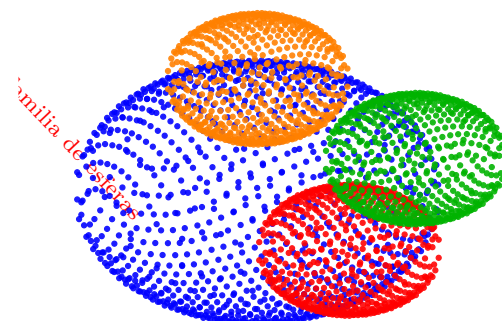


(b) Gráfica $\text{\TikZ}$

Figura 9.7: Gráficas del ejemplo 66



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 9.8: Gráficas del ejemplo 67

Ejemplo 67 Familia de esferas

```

Show[ContourPlot3D[
  x^2 + y^2 + z^2 = 4, {x, -3, 3}, {y, -3, 3}, {z, -3, 3},
  ContourStyle -> Opacity[0.7],
  ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ContourPlot3D[(x - 2)^2 + y^2 + z^2 = 1, {x, 0, 4}, {y, -2,
  2}, {z, -2, 2}, ContourStyle -> Opacity[0.7],
  ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ContourPlot3D[
  x^2 + (y - 2)^2 + z^2 = 1, {x, -2, 2}, {y, 0, 4}, {z, -2, 2},
  ContourStyle -> Opacity[0.7],
  ColorFunction -> RandomChoice[ColorData["Gradients"]]],
ContourPlot3D[
  x^2 + y^2 + (z - 2)^2 = 1, {x, -2, 2}, {y, -2, 2}, {z, 0, 4},
  ContourStyle -> Opacity[0.7],
  ColorFunction -> RandomChoice[ColorData["Gradients"]]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"color=",
    color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
      "gray", "green", "lightgray", "lime", "magenta", "olive",
      "orange", "pink", "purple", "red", "teal", "violet", "white",
      "yellow", "blue!50!red", "green!70!black",
      "red!20!blue!80!green", "yellow!75!red"}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"mapacolor=",
    mapacolor =
      RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
        "summer", "autumn", "winter", "gray", "blackwhite"}]}],
  Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"puntossuperimpli=",
    puntossuperimpli = RandomInteger[{8000, 13000}]}],
  Row[{"texto=",
    texto = {{"Familia de esferas", {-1, -1, -1}, {"color",
      colortexto[]}, {"size", sizetexto}, {"anchor",
      anclajetexto}, {"rotate", rotatexto}}]}]}]
ExportToTikZ3D[{{x^2 + y^2 + z^2 = 4, {x, -3, 3}, {y, -3, 3}, {z, -3,
  3}}, {(x - 2)^2 + y^2 + z^2 = 1, {x, 0, 4}, {y, -2, 2}, {z, -2,
  2}}, {x^2 + (y - 2)^2 + z^2 = 1, {x, -2, 2}, {y, 0, 4}, {z, -2,
  2}}, {x^2 + y^2 + (z - 2)^2 = 1, {x, -2, 2}, {y, -2, 2}, {z, 0,
  4}}, "ExportToTikZ3D8.tex", grid, color, resolucion,
estiloejes, mapacolor, opacidad, extensionejes, puntossuperimpli,
texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram** se muestra como salida la gráfica de la subfigura 9.8a y los mensajes:

```
{grid= True, color= orange, resolucion= 20, estiloejes= none, mapacolor= viridis,
opacidad= 0.891644, extensionejes= 0.118879, puntossuperimpli= 8974, texto=
{{Familia de esferas, {-1, -1, -1}, {{color, red}, {size, \scriptsize}, {anchor,
south east}, {rotate, 313.345}}}}}
```

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ3D8/ExportToTikZ3D8.pdf

/Users/.../Downloads/ExportToTikZ3D8/ExportToTikZ3D8.pdf

Lo contenido en la llave de este *Out* corresponde a los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D8.tex* (guardado en *Downloads/ExportToTikZ3D8*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.8b.

Ejemplo 68 Superficie trigonométrica, ondas gaussianas y espiral modulada

```
Show[Plot3D[
Sin[x]*Cos[y] + Sin[y]*Cos[z] + Sin[z]*Cos[x], {x, -Pi,
Pi}, {y, -Pi, Pi}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
Plot3D[Exp[-(x^2 + y^2)]*Cos[3*Sqrt[x^2 + y^2]], {x, -2, 2}, {y, -2,
2}, PlotStyle ->Opacity[0.7],
ColorFunction ->RandomChoice[ColorData["Gradients"]]],
ParametricPlot3D[{t*Cos[3*t], t*Sin[3*t], Sin[t]}, {t, 0, 4*Pi}]]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\tiny", "\scriptsize", "\footnotesize", "\small",
"\normalsize", "\large", "\Large", "\huge", "\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]]];
Row[{"grid=", grid = RandomChoice[{True, False}]],
Row[{"color=",
color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
"gray", "green", "lightgray", "lime", "magenta", "olive",
"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]]],
Row[{"estilojejes=",
estilojejes = RandomChoice[{"box", "center", "none"}]],
Row[{"mapacolor=",
mapacolor =
RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
"summer", "autumn", "winter", "gray", "blackwhite"}]],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]]],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]]],
Row[{"puntossuperimpli=",
puntossuperimpli = RandomInteger[{8000, 13000}]]],
Row[{"texto=",
texto = {{ "Ondas gaussianas u otras", {0, 0, 0},
{"color", colortexto[], {"size", sizetexto}, {"anchor",
```

```

        anclajetexto}, {"rotate", rotatetexto}}}}}}
ExportToTikZ3D[{{Sin[x]*Cos[y] + Sin[y]*Cos[z] +
    Sin[z]*Cos[x], {x, -Pi, Pi}, {y, -Pi, Pi}}, {Exp[-(x^2 + y^2)]*
    Cos[3*Sqrt[x^2 + y^2]], {x, -2, 2}, {y, -2, 2}}, {{t*Cos[3*t],
    t*Sin[3*t], Sin[t]}, {t, 0, 4*Pi}}, "ExportToTikZ3D9.tex", grid,
color, resolusion, estilejes, mapacolor, opacidad, extensiones,
puntossuperimpli, texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram Mathematica** se despliega en el *Out* la gráfica de la subfigura 9.9a y los mensajes de texto:

```

{grid= True, color= brown, resolusion= 24, estilejes= box, mapacolor= jet,
opacidad= 0.60075, extensiones= 0.105146, puntossuperimpli= 8382, texto= {{0ndas
gaussianas u otras, {0, 0, 0}, {{color, gray}, {size, \scriptsize}, {anchor,
south},{rotate, 86.9796}}}}}

```

Usando editores ya detectados: 4 disponibles

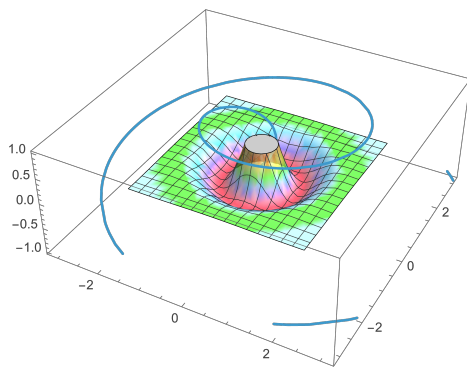
:

```

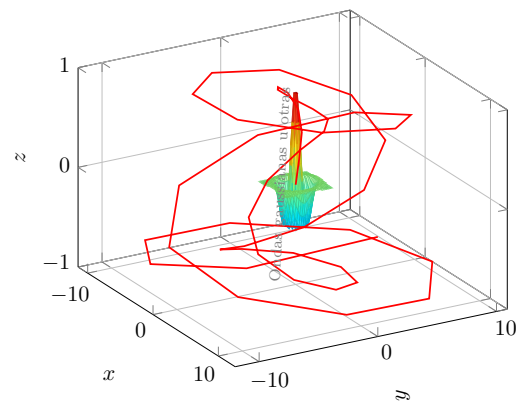
PDF generado: /Users/.../Downloads/ExportToTikZ3D9/ExportToTikZ3D9.pdf
/Users/.../Downloads/ExportToTikZ3D9/ExportToTikZ3D9.pdf

```

La llave de la salida comparte los argumentos pseudoaleatorios pasados al comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D9.tex* (guardado en *Downloads/ExportToTikZ3D9*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.9b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 9.9: Gráficas del ejemplo 68

Ejemplo 69 Hiperboloide de una hoja rotada

```

ContourPlot3D[
0.441208 x^2 + 0.686113 y^2 - 1.179617 z^2 + 0.634549 x y +
0.841523 x*z - 1.476072 y*z - 1.238360 x + 1.232422 y +
0.307829 z - 0.195846 = 0, {x, -3, 5}, {y, -3, 3}, {z, -3, 3},
ContourStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",

```



```

"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}],
rotatetexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
"gray", "green", "lightgray", "lime", "magenta", "olive",
"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]}],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estiloejes=",
estiloejes = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapacolor=",
mapacolor =
RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
"summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
Row[{"puntossuperimpli=",
puntossuperimpli = RandomInteger[{8000, 9000}]}],
Row[{"texto=",
texto = {{{"Hiperboloide rotada", {0, 0, 0}, {"color", colortexto[]},
{"size", sizetexto}, {"anchor",
anclajetexto}, {"rotate", rotatetexto}}}}]}]
ExportToTikZ3D[{0.441208 x^2 + 0.686113 y^2 - 1.179617 z^2 +
0.634549 x y + 0.841523 x*z - 1.476072 y*z - 1.238360 x +
1.232422 y + 0.307829 z - 0.195846 =0, {x, -3, 5}, {y, -3,
3}, {z, -3, 3}}, "ExportToTikZ3D10.tex", grid, color, resolucion,
estiloejes, mapacolor, opacidad, extensionejes, puntossuperimpli,
texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se muestra como salida la gráfica de la subfigura 9.10a y los mensajes siguientes:

```

{grid= False, color= cyan, resolucion= 24, estiloejes= box, mapacolor= viridis,
opacidad= 0.795208, extensionejes= 0.295879, puntossuperimpli= 8073, texto=
{{Hiperboloide rotada, {0, 0, 0}, {{color, purple}, {size, \large}, {anchor, west},
{rotate, 145.105}}}}

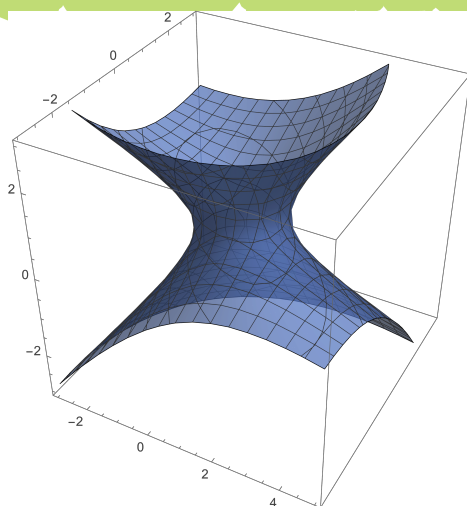
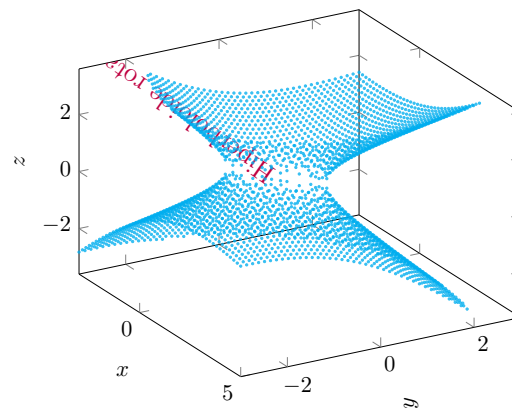
```

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZ3D10/ExportToTikZ3D10.pdf
/Users/.../Downloads/ExportToTikZ3D10/ExportToTikZ3D10.pdf

La llave posee los argumentos pseudoaleatorios usados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D10.tex* (guardado en *Downloads/ExportToTikZ3D10*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.10b.

(a) Gráfica de **Mathematica**

(b) Gráfica Tikz

Figura 9.10: Gráficas del ejemplo 69

Ejemplo 70 Paraboloides elíptico rotado

```
ContourPlot3D[-0.713108 x^2 - 1.065232 y^2 - 0.150672 z^2 -
0.440892 x y - 0.220776 x z + 0.661668 y z + 1.138374 x -
0.820341 y + 1.703368 z - 1.614551 == 0, {x, -3, 5}, {y, -3,
4}, {z, -1, 5}, ContourStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"color=",
    color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
    "gray", "green", "lightgray", "lime", "magenta", "olive",
    "orange", "pink", "purple", "red", "teal", "violet", "white",
    "yellow", "blue!50!red", "green!70!black",
    "red!20!blue!80!green", "yellow!75!red"}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"mapacolor=",
    mapacolor =
    RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
    "summer", "autumn", "winter", "gray", "blackwhite"}]}],
  Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"puntossuperimpli=",
    puntossuperimpli = RandomInteger[{8000, 13000}]}],
  Row[{"texto=",
    texto = {"Paraboloides elíptico rotado", {0, 0, 0},
```



```

{"color", colortexto[]}, {"size", sizetexto}, {"anchor",
  anclajetexto}, {"rotate", rotatexto}}}}}
ExportToTikZ3D[{-0.713108 x^2 - 1.065232 y^2 - 0.150672 z^2 -
  0.440892 x y - 0.220776 x z + 0.661668 y z + 1.138374 x -
  0.820341 y + 1.703368 z - 1.614551 = 0, {x, -3, 5}, {y, -3,
  4}, {z, -1, 5}}, "ExportToTikZ3D11.tex", grid, color, resolusion,
estiloejes, mapacolor, opacidad, extensionejes, puntossuperimpli,
texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram Mathematica** la salida produce la gráfica de la subfigura 9.11a y los mensajes expuestos a continuación:

```

{grid= False, color= green!70!black, resolusion= 17, estiloejes= center, mapacolor=
hot, opacidad= 0.283865, extensionejes= 0.25193, puntossuperimpli= 12178, texto=
{{Paraboloide elíptico rotado, {0, 0, 0}, {{color, purple}, {size, \small},
{anchor, north east}, {rotate, 102.534}}}}}

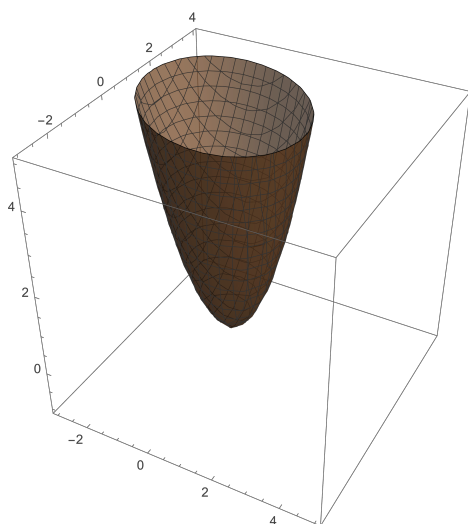
```

Usando editores ya detectados: 4 disponibles

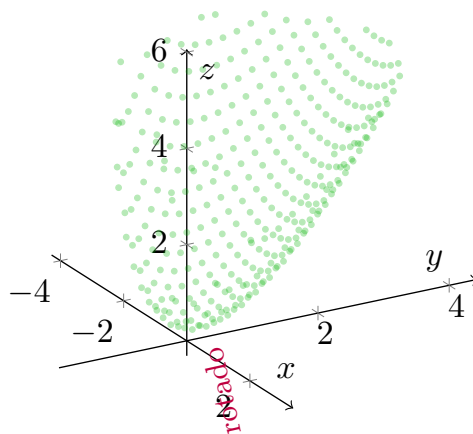
⋮

PDF generado: /Users/.../Downloads/ExportToTikZ3D11/ExportToTikZ3D11.pdf
 /Users/.../Downloads/ExportToTikZ3D11/ExportToTikZ3D11.pdf

Lo contenido en la llave de este *Out* brinda los argumentos pseudoaleatorios del comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D11.tex* (guardado en *Downloads/ExportToTikZ3D11*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.11b. En este ejemplo, la densidad de puntos utilizada en la gráfica $\text{\LaTeX}$ no proporcionó un paraboloide tan relleno como se hubiese querido. Como ya se explicó, la herramienta **ExportToTikZ3D** no siempre retornará resultados 100% fidedignos.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 9.11: Gráficas del ejemplo 70

Ejemplo 71 Cono elíptico rotado

```

ContourPlot3D[
0.982591 x^2 + 1.614910 y^2 - 0.791252 z^2 + 0.995418 x y +
0.823823 x z - 1.819241 y z - 2.126531 x + 2.074885 y -
0.467440 z + 1.768963 = 0, {x, -5, 5}, {y, -5, 5}, {z, -5, 5},
ContourStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
"gray", "green", "lightgray", "lime", "magenta", "olive",
"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]}],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estiloejes=",
estiloejes = RandomChoice[{"box", "center", "none"}]}],
Row[{"mapacolor=",
mapacolor =
RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
"summer", "autumn", "winter", "gray", "blackwhite"}]}],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]}],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
Row[{"puntossuperimpli=",
puntossuperimpli = RandomInteger[{8000, 13000}]}],
Row[{"texto=",
texto = {"Cono elíptico rotado", {0, 0, 0},
{"color", colortexto[]}, {"size", sizetexto}, {"anchor",
anclajetexto}, {"rotate", rotatexto}}]}]}
ExportToTikZ3D[{0.982591 x^2 + 1.614910 y^2 - 0.791252 z^2 +
0.995418 x y + 0.823823 x z - 1.819241 y z - 2.126531 x +
2.074885 y - 0.467440 z + 1.768963 = 0, {x, -5, 5}, {y, -5,
5}, {z, -5, 5}}, "ExportToTikZ3D12.tex", grid, color, resolucion,
estiloejes, mapacolor, opacidad, extensionejes, puntossuperimpli,
texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se muestra como salida la gráfica de la subfigura 9.12a y los mensajes siguientes:

```

{grid= False, color= purple, resolucion= 16, estiloejes= box, mapacolor= autumn,
opacidad= 0.75942, extensionejes= 0.327104, puntossuperimpli= 10432, texto= {"Cono
elíptico rotado, {0, 0, 0}, {"color, magenta}, {size, \large}, {anchor, south},
{rotate, 8.85503}}]}

```

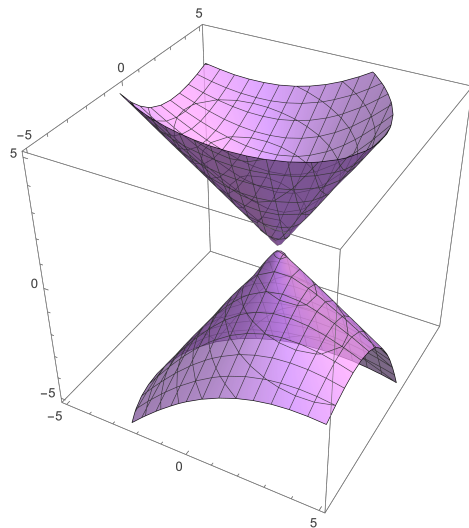
Usando editores ya detectados: 4 disponibles

:

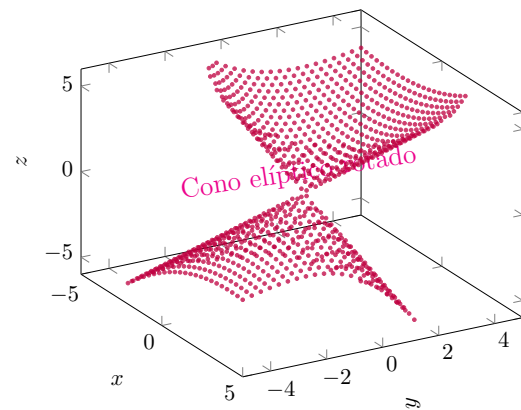
PDF generado: /Users/.../Downloads/ExportToTikZ3D12/ExportToTikZ3D12.pdf

/Users/.../Downloads/ExportToTikZ3D12/ExportToTikZ3D12.pdf

Lo contenido en la llave de este *Out* corresponde a los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D12.tex* (guardado en *Downloads/ExportToTikZ3D12*) comparte el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 9.12b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 9.12: Gráficas del ejemplo 71

Ejemplo 72 Cilindro hiperbólico rotado

```
ContourPlot3D[
0.438760 x^2 - 0.982532 y^2 + 0.003649 z^2 - 1.058176 x y +
0.622450 x z + 0.435726 y z - 1.904567 x - 0.272937 y -
0.410425 z + 0.048219 == 0, {x, -3, 5}, {y, -3, 4}, {z, -3, 3},
ContourStyle -> Opacity[0.7],
ColorFunction -> RandomChoice[ColorData["Gradients"]]]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"color=",
color = RandomChoice[{"black", "blue", "brown", "cyan", "darkgray",
"gray", "green", "lightgray", "lime", "magenta", "olive",
"orange", "pink", "purple", "red", "teal", "violet", "white",
"yellow", "blue!50!red", "green!70!black",
"red!20!blue!80!green", "yellow!75!red"}]}],
Row[{"resolucion=", resolucion = RandomInteger[{10, 60}]}],
Row[{"estilo=",
```

```

estiloejes = RandomChoice[{"box", "center", "none"}]],
Row[{"mapacolor=",
  mapacolor =
    RandomChoice[{"viridis", "cool", "hot", "jet", "hsv", "spring",
      "summer", "autumn", "winter", "gray", "blackwhite"}]],
Row[{"opacidad=", opacidad = RandomReal[{0.2, 1}]]],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]]],
Row[{"puntossuperimpli=",
  puntossuperimpli = RandomInteger[{8000, 13000}]]],
Row[{"texto=",
  texto = {{{"Cilindro hiperbólico rotado", {0, 0, 0},
    {"color", colortexto[]}, {"size", sizetexto}, {"anchor",
      anclajetexto}, {"rotate", rotatetexto}}}}]]]
ExportToTikZ3D[{0.438760 x^2 - 0.982532 y^2 + 0.003649 z^2 -
  1.058176 x y + 0.622450 x z + 0.435726 y z - 1.904567 x -
  0.272937 y - 0.410425 z + 0.048219 = 0, {x, -3, 5}, {y, -3,
    4}, {z, -3, 3}}, "ExportToTikZ3D13.tex", grid, color, resolucio,
estiloejes, mapacolor, opacidad, extensionejes, puntossuperimpli,
texto];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram** la salida crea la gráfica de la subfigura 9.13a y los mensajes de texto:

```

{grid= False, color= lightgray, resolucio= 29, estiloejes= center, mapacolor=
summer, opacidad= 0.787414, extensionejes= 0.187161, puntossuperimpli= 11129,
texto= {{Cilindro hiperbólico rotado, {0, 0, 0}, {{color, magenta}, {size, \tiny},
{anchor, east}, {rotate, 123.011}}}}}

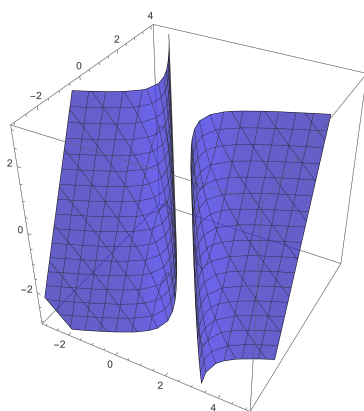
```

Usando editores ya detectados: 4 disponibles

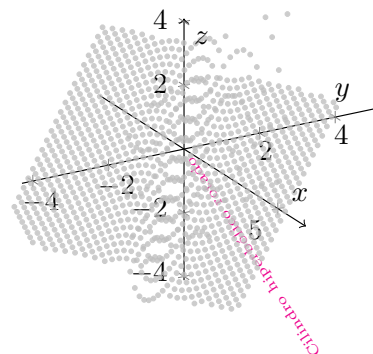
⋮

PDF generado: /Users/.../Downloads/ExportToTikZ3D13/ExportToTikZ3D13.pdf
 /Users/.../Downloads/ExportToTikZ3D13/ExportToTikZ3D13.pdf

La llave expone los argumentos pseudoaleatorios empleados en el comando **ExportToTikZ3D**. El archivo *ExportToTikZ3D13.tex* (guardado en *Downloads/ExportToTikZ3D13*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 9.13b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 9.13: Gráficas del ejemplo 72

Para más ejemplos

Un conjunto más amplio de ejemplos relacionados con la utilización de la sentencia `ExportToTikZ3D` se puede consultar de manera complementaria en el archivo: 9. *Ejemplos ExportToTikZ3D.nb*, disponible en el enlace de descarga del paquete **VilTeX** (ver página [xiv](#)).

Función `ExportToTikZGraphics3D`

¿Qué aprenderás en este capítulo?

- ▶ Exportar objetos **Graphics3D** complejos con **PGFPlots**.
- ▶ Manejar mallas tridimensionales, nubes de puntos y superficies paramétricas.
- ▶ Configurar ejes 3D, planos de corte y cajas delimitadoras.
- ▶ Optimizar la calidad visual y tamaño del código **TikZ** generado.
- ▶ Integrar visualizaciones 3D avanzadas en documentos científicos.

La integración de geometría tridimensional en publicaciones científicas presenta obstáculos cuando los sistemas de renderizado vectorial requieren código editable en lugar de capturas estáticas. Aunque **Wolfram** proporciona primitivas geométricas 3D robustas mediante **Graphics3D**, la exportación directa a formatos *TikZ/PGFPlots* no existe de forma nativa, obligando a conversiones manuales laboriosas o dependencia de imágenes no escalables.

La herramienta de conversión geométrica del paquete **W^ATeX** resuelve esta limitación mediante transformación directa de primitivas tridimensionales en instrucciones **TikZ** equivalentes. En contraste con exportaciones convencionales que incrustan gráficos como objetos cerrados, el comando **ExportToTikZGraphics3D** interpreta cada elemento geométrico individualmente -puntos, segmentos, polígonos, esferas, cilindros, conos- convirtiéndolos en coordenadas *PGFPlots* con preservación de estilos, colores y propiedades estructurales.

El mecanismo de traducción ejecuta descomposición jerárquica de objetos **Graphics3D**: identifica primitivas básicas, extrae atributos de estilo aplicados mediante **style** o directivas de color, convierte geometrías complejas en mallas poligonales simplificadas, calcula envolventes espaciales globales, administra límites de complejidad para prevenir sobrecarga de la memoria y finalmente construye bloques *axis* de *PGFPlots* con coordenadas numéricas optimizadas.

El comando **ExportToTikZGraphics3D** transforma objetos geométricos tridimensionales de **Wolfram** en código *TikZ/PGFPlots* estructurado, generando documentos **W^ATeX** autónomos que mantienen edición vectorial completa y escalabilidad sin pérdida. Frente a métodos tradicionales que rasterizan o vectorizan de manera opaca, **ExportToTikZGraphics3D** produce código fuente **W^ATeX** transparente y manipulable con cualquier editor textual estándar.

La ejecución de `ExportToTikZGraphics3D` retorna un archivo `.tex` con clase *standalone* compilable mediante *pdflatex* sin modificaciones adicionales. Analicemos la estructura sintáctica de esta función.

- `ExportToTikZGraphics3D[graphics3DArgs, filename, grid, axisType, axisExtension, resolution, textAnnotations, maxPolygons]`: Transforma construcciones geométricas 3D elaboradas en **Wolfram Mathematica** hacia código TikZ/LaTeX tridimensional y lo serializa como archivo `.tex` compilable.

Argumentos:

- **graphics3DArgs**: Especificación geométrica 3D válida en **Mathematica**. Admite objetos `Graphics3D` completos, colecciones de primitivas o primitivas aisladas:
 - * `Graphics3D[primitivas]`: Objeto gráfico tridimensional completo con opciones integradas.
 - * Colección de primitivas: `{Point[...], Line[...], Polygon[...], Sphere[...], ...}`.
 - * Primitiva individual: `Point[{x, y, z}], Sphere[{x, y, z}, r]`, entre otras.
 - * **Primitivas reconocidas**:
 - **Point**: Marcadores puntuales en coordenadas especificadas.
 - **Line**: Segmentos conectados entre vértices secuenciales.
 - **Polygon**: Regiones poligonales cerradas definidas por vértices.
 - **Sphere**: Superficies esféricas aproximadas mediante malla de meridianos/paralelos.
 - **Cylinder**: Superficies cilíndricas convertidas en rejilla de generatrices circulares.
 - **Cone**: Superficies cónicas descompuestas en malla radial desde ápice.
 - **Cuboid**: Paralelepípedos representados mediante esqueleto de aristas.
 - **Ellipsoid**: Superficies elipsoidales aproximadas con malla paramétrica.
 - **Arrow**: Vectores direccionales con punta de flecha.
 - **Tube**: Tubos convertidos a trayectorias de línea central.
 - **Text**: Anotaciones textuales posicionadas en coordenadas 3D.
 - **Style**: Modificador que aplica color, grosor, patrón a primitivas encapsuladas.
- **filename**: Cadena especificando el nombre del archivo de destino. Es obligatorio incluir la extensión `.tex` (por ejemplo: `"geometria.tex"`). El sistema construye automáticamente un subdirectorio en *Downloads* usando el nombre base del archivo (sin extensión) y deposita dentro el `.tex` resultante.
- **grid** (opcional, por defecto `True`): Booleano que controla la visualización de rejilla. `True` renderiza rejilla principal en planos coordinados; `False` suprime la cuadrícula.
- **axisType** (opcional, por defecto `"box"`): Cadena que define la representación del sistema de coordenadas:
 - * `"box"`: Ejes formando caja tridimensional completa (seis caras).
 - * `"center"`: Ejes cruzados emergiendo del origen con terminadores de flecha.
 - * `"none"`: Sistema de ejes invisible.
- **axisExtension** (opcional, por defecto `0.0`): Número real no negativo especificando una extensión proporcional de ejes más allá de los límites computados de datos. Operativo únicamente con `axisType -> "center"`. Interpretación: `0.15` extiende ejes un 15% adicional del rango para asegurar visibilidad completa de las flechas de los ejes.
- **resolution** (opcional, por defecto `50`): Entero positivo que maneja la densidad de malla para objetos curvos y superficies. Intervalo sugerido: 20-100. Posee una advertencia automática para valores > 200 debido al tamaño excesivo del archivo generado. Resolución elevada produce fidelidad visual superior pero incrementa exponencialmente la carga computacional y el peso del archivo final.

- **textAnnotations** (opcional, por defecto {}): Arreglo de especificaciones de etiquetas textuales posicionadas en el espacio 3D. Sintaxis admitida:
 - * Arreglo vacío {}: Ninguna anotación adicional.
 - * Anotación básica: {"etiqueta", {x, y, z}} coloca texto en coordenadas tridimensionales.
 - * Anotación formateada: {"etiqueta", {x, y, z}, {"atributo", "valor"}, ...}.
 - * Anotaciones múltiples: {"etiqueta1", coord1}, {"etiqueta2", coord2, opciones}, ...}.

Atributos de formato soportados:

- * {"color", "red"}: Cromaticidad del texto. Acepta nombres básicos ("red", "blue", "green", "black", "orange", "gray", "white", "yellow", "purple", "brown", "pink", "cyan", "magenta") y variantes de **Mathematica** (Red, Blue, etc.).
 - * {"size", "tiny"}: Magnitud tipográfica. Comandos $\text{\LaTeX}$ válidos: \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \huge.
 - * {"anchor", "north"}: Punto de anclaje respecto a las coordenadas proporcionadas. Direcciones admitidas: "north", "south", "east", "west", "north east", "north west", "south east", "south west", "center".
 - * {"rotate", "45"}: Ángulo de rotación en grados (0-360).
- **maxPolygons** (opcional, por defecto 500): Entero positivo estableciendo un límite máximo de polígonos exportables. Este valor constituye un mecanismo de protección contra archivos desmedidos. Integra una advertencia automática para valores > 10000. Los polígonos excedentes se omiten con una notificación explícita.

Retorno en **Mathematica**:

Cadena conteniendo la ruta absoluta del archivo *.tex* exportado en el subdirectorío creado dentro de *Downloads*, o *\$Failed* ante errores de validación o ejecución.

Características de estilo:

- **Colores**: El sistema reconoce colores nominales de **Mathematica** (Red, Blue, Green, Black, Orange, Gray, White, Yellow, Purple, Brown, Pink, Cyan, Magenta), equivalentes en minúscula ("red", "blue", etc.), y especificaciones programáticas: `RGBColor[r, g, b]` (valores $\in [0, 1]$), `GrayLevel[nivel]`, `Hue[matiz]`.
- **Tamaño de puntos**: Mediante `PointSize`, acepta valores nominales (Tiny, Small, Medium, Large) o numéricos personalizados.
- **Grosor de líneas**: Mediante `Thickness`, acepta valores nominales (Tiny, Small, Medium, Large, Thick) o numéricos personalizados.
- **Patrones de trazo**: `Dashed` aplica patrón discontinuo estándar; `Dashing[patron]` permite patrones personalizados.

Configuración interna fija:

- Color predeterminado: Azul ("blue") para elementos sin especificación cromática explícita.
- Precisión numérica: Coordenadas formateadas a 3 decimales para optimizar el tamaño del archivo.
- Expansión automática: *padding* del 10% alrededor del envoltorio de datos para visualización completa.

Nota: Los objetos volumétricos exportados (esferas, cilindros, conos, elipsoides, cuboides) no se renderizan como sólidos rellenos sino como estructuras de malla *wireframe*: esqueletos compuestos por líneas que definen su forma geométrica. Esta representación resulta del mecanismo de conversión subyacente: *TikZ/PGFPlots* carece de capacidades nativas de renderizado sólido 3D con sombreado realista, por lo que geometrías complejas se descomponen en redes de segmentos lineales que aproximan su superficie. Consecuentemente, objetos como **Sphere** aparecen como rejillas esféricas de meridianos y paralelos, mientras **Cylinder** se visualiza mediante generatrices circulares paralelas. Esta limitación es inherente al sistema *TikZ* y no específica del paquete **VtT_EX**.

Capacidades especializadas:

- Interpretación automática de jerarquías **Graphics3D** con extracción recursiva de primitivas anidadas.
- Conversión inteligente de geometrías volumétricas complejas (esferas, cilindros, conos, elipsoides) en aproximaciones poligonales optimizadas según el parámetro **resolution**.
- Preservación exacta de atributos de estilo aplicados mediante directivas **style** o modificadores directos de color/grosor.
- Manejo robusto de primitivas degeneradas o geometrías vacías sin interrupción del proceso de exportación.
- Cálculo unificado de límites espaciales mediante análisis global de todas las primitivas presentes, garantizando un renderizado coherente de escenas multi-objeto.
- Protección contra saturación de memoria mediante el límite **maxPolygons** con omisión controlada y notificación al usuario.
- Generación de código *PGFPlots* estructurado siguiendo el estándar *axis/addplot3* con sintaxis compatible universalmente.
- Construcción automática de estructura de directorios en *Downloads* usando nomenclatura basada en el nombre del archivo.
- Generación de documento **TeX** completo con clase *standalone*, incluyendo paquetes esenciales: *tikz*, *pgfplots*, *amsmath*, *amssymb* y bibliotecas: *arrows.meta*, *colorbrewer*, *colormaps*.
- Validación comprehensiva de argumentos con mensajes descriptivos en español ante errores de tipo, rango o formato.

10.1 Ejemplos de uso de la función ExportToTikZGraphics3D

Los ejemplos siguientes ilustran el rango operativo de **ExportToTikZGraphics3D** mediante escenarios progresivamente complejos que abarcan desde elementos geométricos básicos hasta composiciones tridimensionales elaboradas. Cada caso demuestra capacidades específicas del comando: manejo de vectores direccionales, integración de múltiples primitivas con estilos diferenciados, anotaciones textuales personalizadas, geometrías volumétricas variadas y escenas complejas con numerosos componentes interrelacionados.

La secuencia de ejemplos progresa deliberadamente desde configuraciones simples hacia construcciones arquitectónicamente sofisticadas, permitiendo observar cómo la herramienta gestiona incrementos en complejidad geométrica. Los primeros casos exhiben conversiones limpias de primitivas estándar, mientras ejemplos posteriores evidencian tanto fortalezas -preservación consistente de colores, grosores y jerarquías estructurales- como limitaciones inherentes del proceso de traducción.

Nota: Es fundamental reconocer que `ExportToTikZGraphics3D`, pese a su robustez general, presenta debilidades específicas en la interpretación de orientación espacial de ciertos objetos volumétricos. Particularmente, cilindros definidos mediante especificación explícita de puntos inicial y final pueden experimentar errores de orientación durante la conversión a mallas TikZ .

El último ejemplo compartido en esta sección ilustra deliberadamente esta limitación. La deficiencia deriva de la complejidad intrínseca de transformar representaciones 3D de **Wolfram Mathematica** hacia aproximaciones poligonales en *PGFPlots*, dado que el proceso ocasionalmente introduce inconsistencias geométricas en objetos con orientación arbitraria. No obstante, para la mayoría de aplicaciones -diagramas conceptuales, esquemas ilustrativos, visualizaciones de datos- la herramienta produce resultados satisfactorios que equilibran precisión geométrica con editabilidad vectorial completa en $\text{\LaTeX}$.

Los parámetros aleatorios empleados en los ejemplos siguientes (colores de texto, tamaños, anclajes, rotaciones) demuestran la versatilidad del sistema de anotaciones y permiten evaluar robustez ante configuraciones variadas.

Ejemplo 73 Varios vectores

```
puntos = {{-5, 7, -4}, {-5, 0, 0}, {0, 7, 0}, {0, 0, -4}, {1, 2, -3}};
ExportToTikZGraphics3D1 =
Graphics3D[{Red, Arrow[{{0, 0, 0}, #]} & /@ puntos, Axes → True,
AxesLabel → {"x", "y", "z"}]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"estilo", estilo = RandomChoice[{"box", "center", "none"}]}],
Row[{"extension", extension = RandomReal[{0.1, 0.4}]}],
Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
Row[{"texto=",
  texto = {"A",
    puntos[[1]] +
    RandomChoice[{1, -1}] 0.01, {"color", colortexto[]}, {"size",
      sizetexto}, {"anchor", anclajetexto}, {"rotate",
        rotatexto}}, {"B",
    puntos[[2]] +
    RandomChoice[{1, -1}] 0.01, {"color", colortexto[]}, {"size",
      sizetexto}, {"anchor", anclajetexto}, {"rotate",
        rotatexto}}, {"C",
    puntos[[3]] +
    RandomChoice[{1, -1}] 0.01, {"color", colortexto[]}, {"size",
      sizetexto}, {"anchor", anclajetexto}, {"rotate",
        rotatexto}}, {"D",
    puntos[[4]] +
    RandomChoice[{1, -1}] 0.01, {"color", colortexto[]}, {"size",
      sizetexto}, {"anchor", anclajetexto}, {"rotate",
```

```

        rotatetexto}}}, {"E",
        puntos[[5]] +
        RandomChoice[{1, -1}] 0.01, {"color", colortexto[]}, {"size",
        sizetexto}, {"anchor", anclajetexto},
        {"rotate", rotatetexto}}}}],
    Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]]}]
ExportToTikZGraphics3D[ExportToTikZGraphics3D1,
"ExportToTikZGraphics3D1.tex", grid, estiloejes, extensionejes,
resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Mathematica** se retorna como salida la gráfica de la subfigura 10.1a y los mensajes siguientes:

```

{grid= False, estiloejes= box, extensionejes= 0.127203, resolucion= 79, texto=
{{A, {-5.01, 6.99, -4.01}, {{color, darkgray}, {size, \footnotesize}, {anchor,
north}, {rotate, 109.031}}}, {B, {-5.01, -0.01, -0.01}, {{color, gray}, {size,
\footnotesize}, {anchor, north}, {rotate, 109.031}}}, {C, {-0.01, 6.99, -0.01},
{{color, brown}, {size, \footnotesize}, {anchor, north}, {rotate, 109.031}}},
{D, {-0.01, -0.01, -4.01}, {{color, violet}, {size, \footnotesize}, {anchor,
north}, {rotate, 109.031}}}, {E, {0.99, 1.99, -3.01}, {{color, lime}, {size,
\footnotesize}, {anchor, north}, {rotate, 109.031}}}}, maxpoligonos= 638}

```

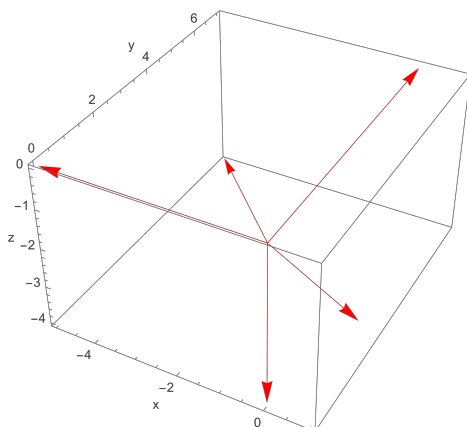
Usando editores ya detectados: 4 disponibles

⋮

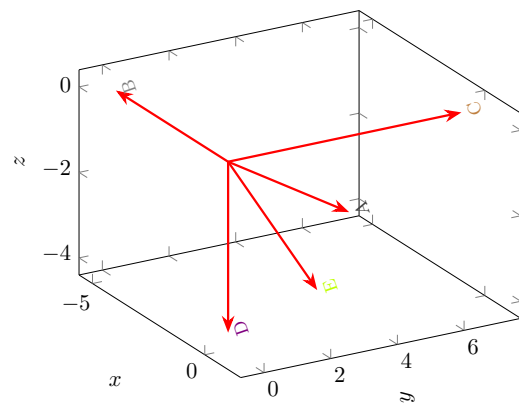
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D1/ExportToTikZGraphics3D-1.pdf

/Users/.../Downloads/ExportToTikZGraphics3D1/ExportToTikZGraphics3D1.pdf

Lo contenido en la llave de este *Out* corresponde a los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D1.tex* (guardado en *Downloads/ExportToTikZGraphics3D1*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 10.1b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 10.1: Gráficas del ejemplo 73

Ejemplo 74 Varias figuras 3D

```

ExportToTikZGraphics3D2 =
Graphics3D[{Blue, Cylinder[], Red, Sphere[{0, 0, 2}], Black, Thick,
  Dashed, Line[{{-2, 0, 2}, {2, 0, 2}, {0, 0, 4}, {-2, 0, 2}}],
  Yellow, Polygon[{{-3, -3, -2}, {-3, 3, -2}, {3, 3, -2}, {3, -3, -2}}],
  Green, Opacity[.3], Cuboid[{-2, -2, -2}, {2, 2, -1}]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {{ "Varias figuras 3D", {0, 0, 3}, {"color", colortexto[]},
      {"size", sizetexto}, {"anchor",
        anclajetexto}, {"rotate", rotatexto}}]}],
  Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]}
ExportToTikZGraphics3D[ExportToTikZGraphics3D2,
  "ExportToTikZGraphics3D2.tex", grid, estiloejes, extensionejes,
  resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

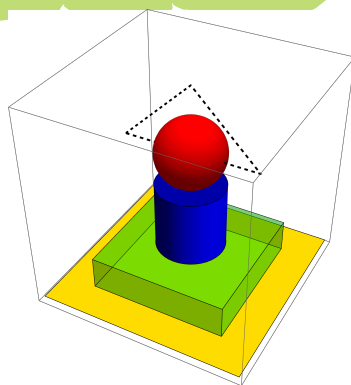
En **Wolfram Mathematica** la salida obtenida responde a la gráfica de la subfigura 10.2a y los mensajes de texto:

```

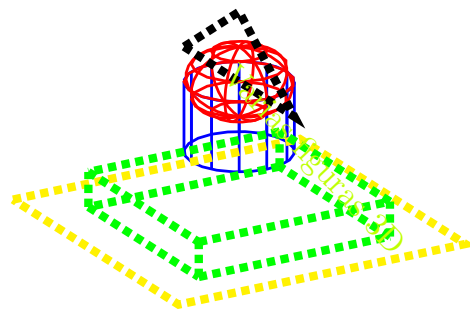
{grid= True, estiloejes= none, extensionejes= 0.226164, resolucion= 77, texto=
{{Varias fi-guras 3D, {0, 0, 3}, {{color, lime}, {size, \large}, {anchor, north
west}, {rotate, 313.438}}}}, maxpoligonos= 608}
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D2/ExportToTikZGraphics3D-
2.pdf
/Users/.../Downloads/ExportToTikZGraphics3D2/ExportToTikZGraphics3D2.pdf

```

Lo mostrado en la llave de este *Out* corresponde a los parámetros pseudoaleatorios empleados en el comando `ExportToTikZGraphics3D`. El archivo *ExportToTikZGraphics3D2.tex* (guardado en *Downloads/ExportToTikZGraphics3D2*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 10.2b.



(a) Gráfica de Mathematica



(b) Gráfica Tikz

Figura 10.2: Gráficas del ejemplo 74

Ejemplo 75 Varias figuras 3D

```

ExportToTikZGraphics3D3 =
Graphics3D[{{Opacity[0.5], Blue, Sphere[{0, 0, 0}, 1.5]}, {Orange,
  Opacity[0.6],
  Cylinder[{{-3, -1, -1}, {-3, -1, 2}}, 0.6]}, {GrayLevel[0.2],
  Specularity[White, 20], Cuboid[{2, -2, -1}, {3.5, -0.5, 1}]},
  EdgeForm[Directive[Black, Thin]], {Lighter[Red, .2], Opacity[0.8],
  Polygon[{{0, 3, 0}, {1, 2, 0}, {-1, 2, 0}}],
  Polygon[{{0, 4, 1.5}, {0, 3, 0}, {1, 2, 0}}],
  Polygon[{{0, 4, 1.5}, {1, 2, 0}, {-1, 2, 0}}],
  Polygon[{{0, 4, 1.5}, {-1, 2, 0}, {0, 3, 0}}]}, {Thick,
  Darker[Green],
  Tube[Line[Table[{Cos[t], Sin[t], t/6}, {t, 0, 10 Pi, .1}]],
  0.07]}, {PointSize[Large], Black, Point[{0, 0, 0}]},
  Text[Style["Centro", 14, Bold], {0, 0, 0.2}],
  Arrow[{{-3, -1, 2.8}, {-1.2, -0.4, 1.6}}], Boxed -> True,
Axes -> True, PlotRange -> All, Lighting -> "Neutral",
SphericalRegion -> True]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]},
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {{ "Varias figuras 3D", {0, 0, 3}, {"color", colortexto[]},
      {"size", sizetexto}, {"anchor",
        anclajetexto}, {"rotate", rotatexto}}}},
  Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]}
ExportToTikZGraphics3D[ExportToTikZGraphics3D3,
  "ExportToTikZGraphics3D3.tex", grid, estiloejes, extensionejes,

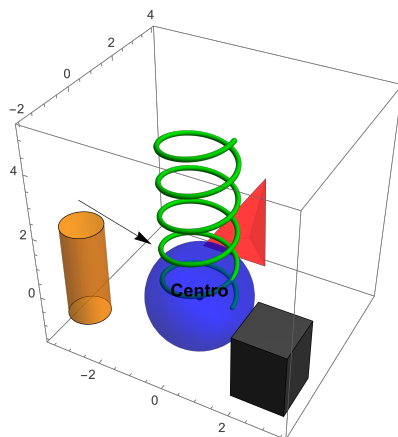
```

```
resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

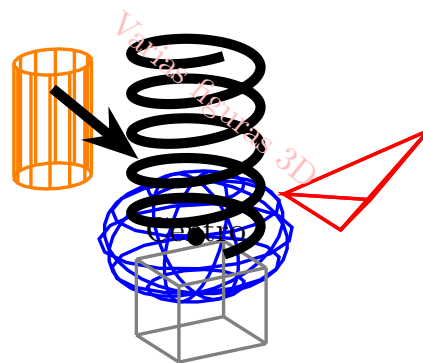
En **Wolfram** el *Out* generado corresponde a la gráfica de la subfigura 10.3a y a los mensajes expuestos a continuación:

```
{grid= True, estilo ejes= none, extension ejes= 0.339974, resolucion= 34, texto=
{{Varias figuras 3D, {0, 0, 3}, {{color, pink}, {size, \normalsize}, {anchor,
south}, {rotate, 320.681}}}}, maxpoligonos= 559}
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D3/ExportToTikZGraphics3D-
3.pdf
/Users/.../Downloads/ExportToTikZGraphics3D3/ExportToTikZGraphics3D3.pdf
```

La llave anterior explicita los argumentos pseudoaleatorios pasados a la instrucción **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D3.tex* (guardado en *Downloads/ExportToTikZGraphics3D3*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 10.3b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 10.3: Gráficas del ejemplo 75

Ejemplo 76 Elipsoides *Graphics3D*

```
ExportToTikZGraphics3D4 =
Graphics3D[{{Opacity[0.3], Blue, Sphere[{0, 0, 0}, 2]}, {Opacity[0.6],
Red, Ellipsoid[{1, 1, 1}, {2, 1, 1}]}, {Thick, Black,
Line[{{-3, -3, -3}, {3, 3, 3}}]}}, Boxed → False,
Lighting → "Neutral"]

{colortexto[] :=
RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
"green", "lightgray", "lime", "magenta", "olive", "orange",
"pink", "purple", "red", "teal", "violet", "yellow"}],
sizetexto =
RandomChoice[{"\tiny", "\scriptsize", "\footnotesize", "\small",
"\normalsize", "\large", "\Large", "\huge", "\Huge"}],
```



```

anclajetexto =
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}],
rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"estiloejes=",
  estiloejes = RandomChoice[{"box", "center", "none"}]}],
Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
Row[{"texto=",
  texto = {{"Elipsoides", {0, 0, 4}, {"color", colortexto[]},
    {"size", sizetexto}, {"anchor",
      anclajetexto}, {"rotate", rotatexto}}]}],
Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]]
ExportToTikZGraphics3D[ExportToTikZGraphics3D4,
"ExportToTikZGraphics3D4.tex", grid, estiloejes, extensionejes,
resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** la salida obtenida muestra la gráfica de la subfigura 10.4a y los mensajes:

```

{grid= False, estiloejes= none, extensionejes= 0.142621, resolucion= 31, texto=
{{Elipsoides, {0, 0, 4}, {{color, magenta}, {size, \tiny}, {anchor, north},
{rotate, 141.592}}}}, maxpoligonos= 671}

```

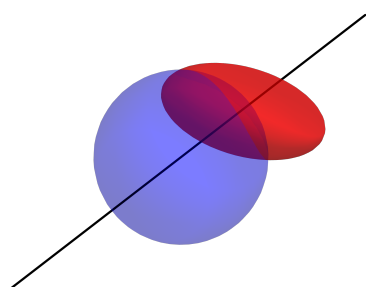
Usando editores ya detectados: 4 disponibles

:

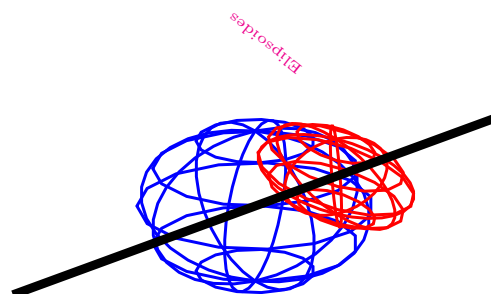
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D4/ExportToTikZGraphics3D-4.pdf

/Users/.../Downloads/ExportToTikZGraphics3D4/ExportToTikZGraphics3D4.pdf

Lo contenido en la llave comparte los argumentos pseudoaleatorios utilizados en el comando **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D4.tex* (guardado en *Downloads/ExportToTikZGraphics3D4*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 10.4b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 10.4: Gráficas del ejemplo 76

Ejemplo 77 Sistema solar simplificado

125

```

ExportToTikZGraphics3D5 =
Graphics3D[{Yellow, Sphere[{0, 0, 0}, 0.5], Gray,
  Sphere[{1.5, 0, 0}, 0.1], Orange, Sphere[{2.2, 0.5, 0.2}, 0.15],
  Blue, Sphere[{3.0, 0, 0}, 0.2], Red, Sphere[{4.5, -0.8, 0.1}, 0.12],
  Gray, Line[Table[{1.5*Cos[t], 1.5*Sin[t], 0}, {t, 0, 2*Pi, Pi/20}]],
  Line[Table[{2.2*Cos[t], 2.2*Sin[t], 0}, {t, 0, 2*Pi, Pi/20}]],
  Line[Table[{3.0*Cos[t], 3.0*Sin[t], 0}, {t, 0, 2*Pi, Pi/20}]],
  Line[Table[{4.5*Cos[t], 4.5*Sin[t], 0}, {t, 0, 2*Pi, Pi/20}]],
  Gray, Point[
    Table[{5 + 0.5*RandomReal[], RandomReal[{-0.3, 0.3}],
      RandomReal[{-0.2, 0.2}]], {20}], Text["Sol", {0, 0, 0.8}],
    Text["Tierra", {3.0, 0, 0.5}]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estiloejes=",
    estiloejes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"extensionejes=", extensionejes = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {{ "Sistema solar", {-0.5, -0.5, -0.5}, {"color",
      colortexto[]}, {"size", sizetexto}, {"anchor",
      anclajetexto}, {"rotate", rotatexto}}}],
  Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]}
ExportToTikZGraphics3D[ExportToTikZGraphics3D5,
  "ExportToTikZGraphics3D5.tex", grid, estiloejes, extensionejes,
  resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram Mathematica** se muestra como salida la gráfica de la subfigura 10.5a y los mensajes de texto:

```

{grid= True, estiloejes= box, extensionejes= 0.386744, resolucion= 72, texto=
{{Sistema solar, {-0.5, -0.5, -0.5}, {color, darkgray}, {size, \small}, {anchor,
north west}, {rotate, 193.298}}}, maxpoligonos= 537}

```

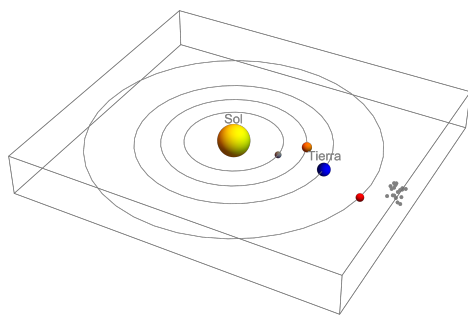
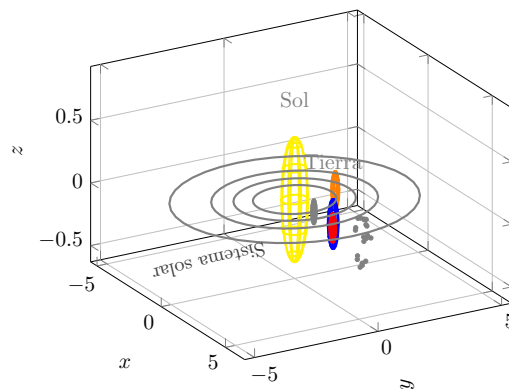
Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D5/ExportToTikZGraphics3D-5.pdf

/Users/.../Downloads/ExportToTikZGraphics3D5/ExportToTikZGraphics3D5.pdf

La llave anterior expone los parámetros pseudoaleatorios usados en el comando **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D5.tex* (guardado en *Downloads/ExportToTikZGraphics3D5*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 10.5b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 10.5: Gráficas del ejemplo 77

Ejemplo 78 Átomo

```

ExportToTikZGraphics3D6 =
Graphics3D[{Red, Sphere[{0, 0, 0}, 0.4], Pink,
  Sphere[{0.1, 0.1, 0}, 0.15], Pink, Sphere[{-0.1, -0.1, 0}, 0.15],
  Gray, Sphere[{0.1, -0.1, 0}, 0.15], Gray,
  Sphere[{-0.1, 0.1, 0}, 0.15], Gray,
  Line[Table[{1.5*Cos[t], 1.5*Sin[t], 0}, {t, 0, 2*Pi, Pi/25}]],
  Gray, Line[
    Table[{2.2*Cos[t], 1.1*Sin[t], 1.1*Cos[2*t]}, {t, 0, 2*Pi, Pi/25}]],
  Blue, Sphere[{1.5*Cos[Pi/3], 1.5*Sin[Pi/3], 0}, 0.1], Blue,
  Sphere[{1.5*Cos[4*Pi/3], 1.5*Sin[4*Pi/3], 0}, 0.1], Blue,
  Sphere[{2.2*Cos[Pi/6], 1.1*Sin[Pi/6], 1.1*Cos[Pi/3]}, 0.1], Yellow,
  Line[Table[{3*Cos[t], 0, 3*Sin[t]}, {t, 0, 2*Pi, Pi/12}]],
  Line[Table[{0, 3*Cos[t], 3*Sin[t]}, {t, 0, 2*Pi, Pi/12}]], Cyan,
  Point[Table[{1.8*Cos[t], 1.8*Sin[t], 0.2}, {t, 0, 2*Pi, Pi/8}]],
  Point[Table[{2.5*Cos[t], 1.25*Sin[t], 1.25*Cos[2*t]}, {t, 0, 2*Pi,
    Pi/8}]], Text["Núcleo", {0, 0, -0.7}]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estilo", estilo = RandomChoice[{"box", "center", "none"}]}],
  Row[{"extension", extension = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {{ "Átomo", {0,
      0, -1}, {"color", colortexto[]}, {"size",
        sizetexto}, {"anchor", anclajetexto}, {"rotate",
        rotatexto}}}}],

```

```
Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]
ExportToTikZGraphics3D[ExportToTikZGraphics3D6,
"ExportToTikZGraphics3D6.tex", grid, estiloEjes, extensionEjes,
resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** el *Out* despliega la gráfica de la subfigura 10.6a y los mensajes siguientes:

```
{grid= True, color= brown, resolucion= 57, estiloEjes= none, mapacolor= blackwhite,
opacidad= 0.617355, extensionEjes= 0.283539, puntossuperimpli= 8757, texto=
{{f(x,y) = sen(x)cos(y)$, {1, 1, 1}, {{color, gray}, {size, \Huge}, {anchor,
center}, {rotate, 324.018}}}}
```

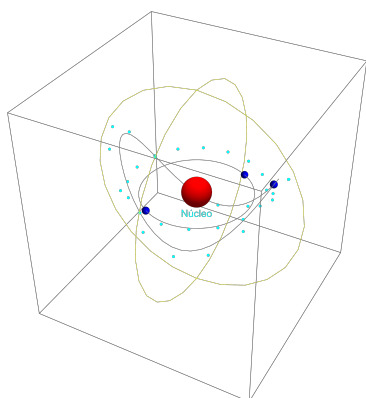
Usando editores ya detectados: 4 disponibles

⋮

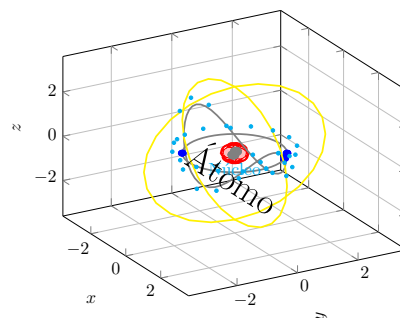
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D6/ExportToTikZGraphics3D-6.pdf

/Users/.../Downloads/ExportToTikZGraphics3D6/ExportToTikZGraphics3D6.pdf

La llave de la salida anterior socializa los argumentos pseudoaleatorios empleados en la sentencia **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D6.tex* (guardado en *Downloads/ExportToTikZGraphics3D6*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 10.6b.

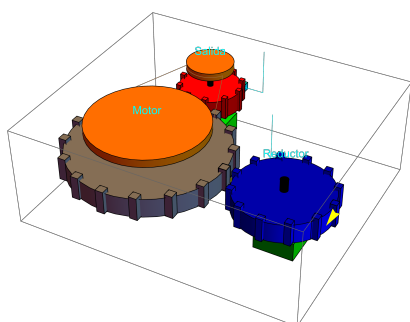


(a) Gráfica de **Mathematica**

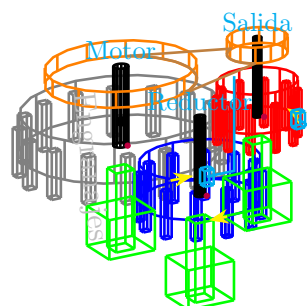


(b) Gráfica $\text{\TikZ}$

Figura 10.6: Gráficas del ejemplo 78



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 10.7: Gráficas del ejemplo 79

Ejemplo 79 Sistema de engranajes

```

ExportToTikZGraphics3D7 =
Graphics3D[{{Gray, Cylinder[{{0, 0, 0}, {0, 0, 0.5}}, 2],
  Table[Cuboid[{{2*Cos[t] - 0.1, 2*Sin[t] - 0.1, 0}, {2*Cos[t] + 0.1,
    2*Sin[t] + 0.1, 0.6}], {t, 0, 2*Pi, Pi/8}}, Blue,
  Cylinder[{{3.5, 0, 0}, {3.5, 0, 0.5}}, 1.2],
  Table[Cuboid[{{3.5 + 1.2*Cos[t] - 0.08, 1.2*Sin[t] - 0.08,
    0}, {3.5 + 1.2*Cos[t] + 0.08, 1.2*Sin[t] + 0.08, 0.6}], {t, 0,
    2*Pi, Pi/6}}, Red, Cylinder[{{0, 3, 0}, {0, 3, 0.5}}, 0.8],
  Table[Cuboid[{{0.8*Cos[t] - 0.06, 3 + 0.8*Sin[t] - 0.06,
    0}, {0.8*Cos[t] + 0.06, 3 + 0.8*Sin[t] + 0.06, 0.6}], {t, 0,
    2*Pi, Pi/5}}, Black, Cylinder[{{0, 0, -0.2}, {0, 0, 0.8}}, 0.15],
  Cylinder[{{3.5, 0, -0.2}, {3.5, 0, 0.8}}, 0.1],
  Cylinder[{{0, 3, -0.2}, {0, 3, 0.8}}, 0.08], Orange,
  Cylinder[{{0, 0, 1}, {0, 0, 1.2}}, 1.5],
  Cylinder[{{0, 3, 1}, {0, 3, 1.2}}, 0.6], Brown,
  Line[Table[{{1.5*Cos[t], 1.5*Sin[t], 1.1}, {t, Pi/6, 5*Pi/6, Pi/20}}],
  Line[Table[{{0.6*Cos[t], 3 + 0.6*Sin[t],
    1.1}, {t, -Pi/6, -5*Pi/6, -Pi/20}}],
  Line[{{1.5*Cos[Pi/6], 1.5*Sin[Pi/6], 1.1}, {0.6*Cos[-Pi/6],
    3 + 0.6*Sin[-Pi/6], 1.1}}],
  Line[{{1.5*Cos[5*Pi/6], 1.5*Sin[5*Pi/6], 1.1}, {0.6*Cos[-5*Pi/6],
    3 + 0.6*Sin[-5*Pi/6], 1.1}}], Green,
  Cuboid[{{-0.5, -0.5, -1}, {0.5, 0.5, -0.5}},
  Cuboid[{{3, -0.5, -1}, {4, 0.5, -0.5}},
  Cuboid[{{-0.5, 2.5, -1}, {0.5, 3.5, -0.5}},
  Cuboid[{{-0.2, -0.2, -1}, {0.2, 0.2, 0}},
  Cuboid[{{3.3, -0.2, -1}, {3.7, 0.2, 0}},
  Cuboid[{{-0.2, 2.8, -1}, {0.2, 3.2, 0}}, Black,
  Sphere[{{0, 0, 0.6}, 0.05], Sphere[{{3.5, 0, 0.6}, 0.04],
  Sphere[{{0, 3, 0.6}, 0.03], Yellow,
  Arrow[{{2.2, 0, 0.25}, {2.2, 0.5, 0.25}}],
  Arrow[{{4.7, 0, 0.25}, {4.7, -0.4, 0.25}}],
  Arrow[{{0, 3.8, 0.25}, {-0.3, 3.8, 0.25}}], Purple,
  Point[{{0.1, 0.1, 0.3}, {3.6, 0.1, 0.3}, {0.1, 3.1, 0.3}}], Cyan,
  Cuboid[{{2.5, 0.5, 0.2}, {2.7, 0.7, 0.4}},
  Cuboid[{{0.5, 3.5, 0.2}, {0.7, 3.7, 0.4}},
  Line[{{2.6, 0.6, 0.3}, {2.6, 1.2, 0.3}, {2.6, 1.2, 1.5}}],
  Line[{{0.6, 3.6, 0.3}, {1.2, 3.6, 0.3}, {1.2, 3.6, 1.5}}],
  Text["Motor", {0, 0, 1.5}], Text["Reductor", {3.5, 0, 1.5}],
  Text["Salida", {0, 3, 1.5}]]]

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estiloeyes=",
    estiloeyes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"extensioneyes=", extensioneyes = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {"Engranajes", {0,

```

```

0, -1}, {"color", colortexto[]}, {"size",
  sizetexto}, {"anchor", anclajetexto}, {"rotate",
  rotatetexto}}]]],
Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]]}
ExportToTikZGraphics3D[ExportToTikZGraphics3D7,
"ExportToTikZGraphics3D7.tex", grid, estiloejes, extensionejes,
resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram** la salida de este código retorna la gráfica de la subfigura 10.7a y los mensajes de texto:

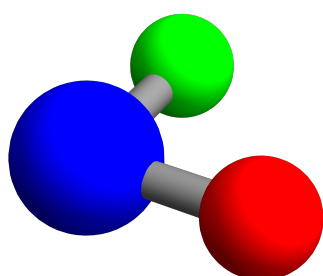
```

{grid= True, estiloejes= none, extensionejes= 0.109233, resolucion= 52, texto=
{{Engranajes, {0, 0, -1}, {{color, lightgray}, {size, \large}, {anchor, north
east}, {rotate, 275.926}}}}, maxpoligonos= 614}
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D7/ExportToTikZGraphics3D-
7.pdf
/Users/.../Downloads/ExportToTikZGraphics3D7/ExportToTikZGraphics3D7.pdf

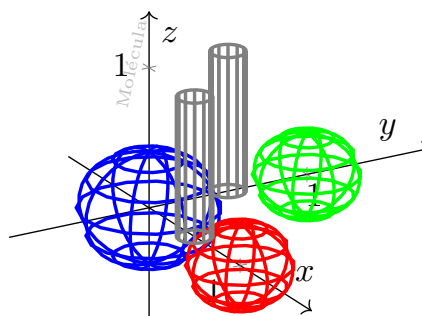
```

La llave de este *Out* comparte los parámetros pseudoaleatorios pasados al comando **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D7.tex* (guardado en *Downloads/ExportToTikZGraphics3D7*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 10.7b.

El siguiente ejemplo pretende evidenciar, tal y como se señaló en la introducción de la presente sección, que la sentencia **ExportToTikZGraphics3D** en algunas ocasiones puede fallar en el proceso de conversión $\text{\LaTeX}$ final, al no interpretar adecuadamente la orientación de objetivos tridimensionales. Veamos.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 10.8: Gráficas del ejemplo 80

Ejemplo 80 Molécula

```

ExportToTikZGraphics3D8 =
Graphics3D[{{Blue, Sphere[{0, 0, 0}, 0.4]}, {Red,
  Sphere[{1, 0, 0}, 0.3]}, {Green, Sphere[{0, 1, 0}, 0.3]}, {Gray,
  Cylinder[{0, 0, 0}, {1, 0, 0}], 0.1}}, {Gray,
  Cylinder[{0, 0, 0}, {0, 1, 0}], 0.1}}, Boxed → False,
Lighting → "Neutral"]

```

```

{colortexto[] :=
  RandomChoice[{"black", "blue", "brown", "cyan", "darkgray", "gray",
    "green", "lightgray", "lime", "magenta", "olive", "orange",
    "pink", "purple", "red", "teal", "violet", "yellow"}],
  sizetexto =
  RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
    "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}],
  anclajetexto =
  RandomChoice[{"north", "south", "east", "west", "north east",
    "north west", "south east", "south west", "center"}],
  rotatexto = RandomReal[{0, 360}]];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estiloeyes=",
    estiloeyes = RandomChoice[{"box", "center", "none"}]}],
  Row[{"extensioneyes=", extensioneyes = RandomReal[{0.1, 0.4}]}],
  Row[{"resolucion=", resolucion = RandomInteger[{20, 80}]}],
  Row[{"texto=",
    texto = {"Molécula", {0, 0,
      1}, {"color", colortexto[]}, {"size", sizetexto}, {"anchor",
        anclajetexto}, {"rotate", rotatexto}}]}],
  Row[{"maxpoligonos=", maxpoligonos = RandomInteger[{500, 700}]}]}
ExportToTikZGraphics3D[ExportToTikZGraphics3D8,
  "ExportToTikZGraphics3D8.tex", grid, estiloeyes, extensioneyes,
  resolucion, texto, maxpoligonos];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se muestra como salida la gráfica de la subfigura 10.8a y los mensajes:

```

{grid= True, estiloeyes= center, extensioneyes= 0.157032, resolucion= 78, texto=
{{Molécula, {0, 0, 1}, {{color, lightgray}, {size, \tiny}, {anchor, south},
{rotate, 77.5567}}}}, maxpoligonos= 514}

```

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/ExportToTikZGraphics3D8/ExportToTikZGraphics3D-8.pdf

/Users/.../Downloads/ExportToTikZGraphics3D8/ExportToTikZGraphics3D8.pdf

El contenido en la llave explicita los argumentos pseudoaleatorios empleados en el comando **ExportToTikZGraphics3D**. El archivo *ExportToTikZGraphics3D8.tex* (guardado en *Downloads/ExportToTikZGraphics3D8*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 10.8b. Como se aprecia en la figura 10.8, la orientación de los cilindros de la molécula en la gráfica $\text{\TikZ}$ no es la correcta. No es frecuente que esto ocurra pero el lector debe tener claro que la instrucción **ExportToTikZGraphics3D** presenta limitaciones naturales intrínsecas a la complejidad de las transformaciones requeridas durante el proceso automático de traducción.

Para más ejemplos

Un conjunto más amplio de ejemplos sobre el uso del comando **ExportToTikZGraphics3D** puede consultarse en el archivo *10. Ejemplos ExportToTikZGraphics3D.nb*, disponible para su descarga junto con el paquete **\LaTeX** (véase la página xiv).

Función `ExportConicsToTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Exportar secciones cónicas (circunferencias, elipses, parábolas, hipérbolas) a `TikZ`.
- ▶ Representar elementos característicos: focos, directrices, vértices y asíntotas.
- ▶ Generar diagramas para geometría analítica y álgebra de cónicas.
- ▶ Personalizar colores, estilos y anotaciones de elementos geométricos.
- ▶ Documentar problemas y aplicaciones de geometría euclidiana.

La representación gráfica de secciones cónicas en documentos técnicos requiere herramientas que balanceen precisión matemática con flexibilidad tipográfica. Si bien **Wolfram** ofrece sólidas capacidades para visualizar círculos, elipses, parábolas e hipérbolas mediante `ContourPlot`, la exportación directa a formatos $\text{\LaTeX}$ editables permanece ausente, forzando flujos de trabajo basados en capturas de pantalla o conversiones vectoriales que sacrifican sus posibilidades de edición.

El sistema de exportación cónica del paquete **V₁T_EX** ofrece una traducción algebraica directa de ecuaciones cuadráticas hacia código `TikZ/PGFPlots` parametrizado. En lugar de rasterizar curvas o vectorizarlas como trayectorias cerradas, el comando `ExportConicsToTikZ` realiza clasificación automática del tipo de cónica (elipse, parábola, hipérbola, recta), selecciona estrategias de muestreo optimizadas para cada geometría, resuelve puntos de evaluación mediante algoritmos especializados y construye bloques `addplot` con componentes ordenadas inteligentemente para garantizar un renderizado fluido.

El algoritmo de procesamiento opera en múltiples fases: análisis de coeficientes de ecuaciones cuadráticas para determinar clasificación geométrica, aplicación de métodos paramétricos en elipses y círculos (garantizando continuidad en la gráfica), detección automática de ramas separadas en hipérbolas mediante análisis de distribución espacial de puntos, ordenamiento adaptativo de vértices según orientación de la curva (vertical u horizontal), eliminación de duplicados próximos que generan errores visuales y finalmente ensamblaje de instrucciones `PGFPlots` con configuración de ejes, rejilla y anotaciones.

La invocación de `ExportConicsToTikZ` produce un archivo `.tex` con encabezado completo que compila mediante `pdflatex` sin edición manual.

La función `ExportConicsToTikZ` convierte especificaciones algebraicas de curvas cónicas estándar (círculos, elipses, parábolas, hipérbolas, rectas) en código `TikZ/PGFPlots` bidimensional autónomo,

produciendo documentos $\text{\LaTeX}$ con clase *standalone* inmediatamente compilables. Contrario a exportaciones tradicionales que encapsulan gráficos como objetos binarios, **ExportConicsToTikZ** sintetiza código fuente $\text{\LaTeX}$ textual completamente accesible mediante cualquier editor estándar.

Examinemos la estructura argumental de esta función.

- **ExportConicsToTikZ**[*equations*, *filename*, *grid*, *styles*, *ejes*, *texto*, *xmin*, *xmax*, *ymin*, *ymax*]: Traduce especificaciones algebraicas de ecuaciones cónicas bidimensionales elaboradas en **Wolfram Mathematica** hacia código $\text{\LaTeX}$ plano y formatea el resultado como un archivo *.tex* compilable.

Parámetros:

- **equations** (obligatorio): Especificación de ecuaciones cónicas con dominios de evaluación. Estructura requerida: lista de tripletes donde cada triplete contiene ecuación, rango de variable independiente y rango de variable dependiente.

Formato general:

```
{ {ecuacion, {var_x, xmin_eq, xmax_eq}, {var_y, ymin_eq, ymax_eq}}, ... }
```

Componentes:

- * **ecuacion**: Expresión relacional cónica en dos variables. No acepta los operadores: **LessEqual** ($\leq$), **GreaterEqual** ($\geq$), **Less** ($<$) y **Greater** ($>$).
- * **var_x, var_y**: Símbolos representando variables independiente y dependiente (típicamente x, y).
- * **xmin_eq, xmax_eq**: Intervalo numérico de muestreo para variable horizontal específico de esta ecuación.
- * **ymin_eq, ymax_eq**: Intervalo numérico de muestreo para variable vertical específico de esta ecuación.

Tipos de cónicas reconocidas:

- * **Círculo**: $(x - h)^2 + (y - k)^2 == r^2$ o forma general $x^2 + y^2 + Dx + Ey + F == 0$.
 - * **Elipse**: $(x - h)^2/a^2 + (y - k)^2/b^2 == 1$ o formas generales con coeficientes cuadráticos positivos.
 - * **Parábola vertical**: Ecuación con término x^2 pero sin y^2 (e.g., $y == x^2$).
 - * **Parábola horizontal**: Ecuación con término y^2 pero sin x^2 (e.g., $x == y^2$).
 - * **Hipérbola horizontal**: Coeficientes cuadráticos con signos opuestos, x^2 positivo (e.g., $x^2/a^2 - y^2/b^2 == 1$).
 - * **Hipérbola vertical**: Coeficientes cuadráticos con signos opuestos, y^2 positivo (e.g., $y^2/b^2 - x^2/a^2 == 1$).
 - * **Recta**: Ecuación lineal sin términos cuadráticos.
- **filename** (obligatorio): Cadena especificando el nombre del archivo destino. Es obligatorio incluir la extensión *.tex* (por ejemplo: *"conicas.tex"*). El sistema construye automáticamente un subdirectorio en *Downloads* empleando el nombre base del archivo (sin extensión) y deposita internamente el *.tex* resultante.
 - **grid** (opcional, por defecto **True**): Booleano que controla la visualización de la rejilla. **True** renderiza la cuadrícula principal; **False** suprime la rejilla completamente.
 - **styles** (opcional, por defecto $\{ \}$): Arreglo de especificaciones estilísticas asociadas a cada ecuación. Permite diferenciación visual entre curvas múltiples.

Formato: $\{ \{ \text{estilo1}, \text{color1} \}, \{ \text{estilo2}, \text{color2} \}, \dots \}$.

Comportamiento por defecto: Si se proporciona lista vacía {}, todas las ecuaciones adoptan el estilo {"solid", "black"}.

Estilos de línea disponibles:

- * "solid": Trazo continuo (estándar).
- * "dashed": Trazo discontinuo con guiones.
- * "dotted": Trazo punteado.
- * "thick": Trazo muy grueso.
- * "thin": Trazo delgado.

Colores disponibles: "blue", "red", "green", "purple", "yellow", "orange", "cyan", "magenta", "black".

- **ejes** (opcional, por defecto **True**): Booleano que controla la representación del sistema cartesiano. **True** genera ejes centrados en origen (*axis lines=center*); **False** oculta ejes completamente (*axis lines=none*).
- **texto** (opcional, por defecto {}): Especificación de etiquetas textuales posicionadas en coordenadas del plano cartesiano. Permite anotación directa del gráfico con nomenclatura matemática.

Formato simple (etiqueta única):

```
{"etiqueta", {x, y}}
```

Formato múltiple (varias etiquetas):

```
{{"etiqueta1", {x1, y1}}, {"etiqueta2", {x2, y2}}, ...}
```

- **xmin** (opcional, por defecto **Automatic**): Límite inferior del dominio horizontal del gráfico. Acepta valor numérico explícito o símbolo **Automatic** para cálculo automático basado en la extensión de los datos generados (agrega un margen de 0.5 unidades).
- **xmax** (opcional, por defecto **Automatic**): Límite superior del dominio horizontal. Comportamiento idéntico a **xmin**.
- **ymin** (opcional, por defecto **Automatic**): Límite inferior del rango vertical del gráfico. Comportamiento idéntico a **xmin**.
- **ymax** (opcional, por defecto **Automatic**): Límite superior del rango vertical. Comportamiento idéntico a **xmin**.

Retorno en **Wolfram Mathematica**:

Cadena conteniendo la ruta absoluta del archivo *.tex* exportado en un subdirectorío creado dentro de *Downloads*, permitiendo localización inmediata del resultado.

Métodos de renderizado especializados:

- **Círculos y elipses:** Empleo de parametrización trigonométrica $\{a \cos[t], b \sin[t]\}$ que garantiza continuidad perfecta de la curva cerrada sin discontinuidades en puntos de conexión. Muestreo uniforme en parámetro angular para distribución homogénea de vértices.
- **Hipérbolas:** Detección automática de orientación (horizontal vs. vertical) mediante análisis de signos de coeficientes cuadráticos. Identificación de ramas separadas mediante *clustering* espacial: si existe separación geométrica suficiente (> 0.3 unidades) entre grupos de puntos, se procesan como entidades independientes para evitar conexiones espurias entre asíntotas. Cada rama recibe ordenamiento individual optimizado.
- **Parábolas:** Discriminación entre orientación vertical (solo x^2) y horizontal (solo y^2). Ordenamiento de puntos según eje dominante (ordenar por x para parábolas verticales, por y para horizontales). Eliminación de duplicados próximos (< 0.02 unidades) en coordenada relevante para prevenir líneas verticales/horizontales artificiales.

- **Rectas:** Determinación de orientación mediante comparación de variación en x vs. y . Ordenamiento por coordenada con mayor rango para trazado lineal óptimo.

Configuración interna:

- Precisión numérica: Coordenadas formateadas según magnitud (notación estándar para valores moderados, notación científica para extremos).
- Dimensiones del gráfico: Ancho fijo 10 cm, altura 8 cm (proporción aproximada 5 : 4).
- Márgenes automáticos: Si los límites son **Automatic**, se agrega *padding* de 0.5 unidades alrededor del envoltorio de datos.
- Compatibilidad *PGFPlots*: Código generado compatible con la versión 1.18.

Capacidades distintivas:

- Clasificación automática de tipo geométrico mediante análisis algebraico de coeficientes, eliminando necesidad de especificación manual del usuario.
- Resolución algebraica inteligente: cuando es factible, resuelve ecuaciones analíticamente en lugar de muestreo numérico ciego.
- Detección especializada de hipérbolas con separación automática de ramas mediante algoritmo de *clustering* espacial que identifica discontinuidades asintóticas.
- Ordenamiento adaptativo de vértices según orientación y tipo de curva.
- Eliminación inteligente de duplicados próximos que generarían segmentos degenerados o conexiones verticales/horizontales espurias.
- Manejo robusto de múltiples ecuaciones simultáneas con asignación independiente de estilos y colores.
- Procesamiento seguro de listas de estilos incompletas: si se proporcionan menos estilos que ecuaciones, las ecuaciones faltantes adoptan estilo predeterminado.
- Generación de documento $\text{\LaTeX}$ autónomo con clase *standalone*, incluyendo paquetes esenciales: *tikz*, *pgfplots*, *amsmath*.
- Construcción automática de estructura de directorios en *Downloads* con nomenclatura basada en el nombre del archivo.
- Validación exhaustiva de argumentos con mensajes descriptivos ante errores de formato o tipo.

Restricciones:

- **Cónicas con términos cruzados:** El comando **no procesa** ecuaciones con el término mixto xy (cónicas rotadas). Solo admite ecuaciones de la forma $Ax^2 + By^2 + Cx + Dy + E == 0$ donde los ejes principales están alineados con los ejes coordenados.
- **Desigualdades:** No acepta operadores relacionales $\leq$, $\geq$, $<$, $>$ en la especificación de ecuaciones. Solo genera curvas de frontera ($==$). Para visualizaciones de regiones, se requeriría post procesamiento manual del código $\text{\LaTeX}$ generado, o bien, el uso de la sentencia **ExportContour-2DTToTikZ** que se explicará más adelante en el capítulo 13.

11.1 Ejemplos de uso de la función ExportConicsToTikZ

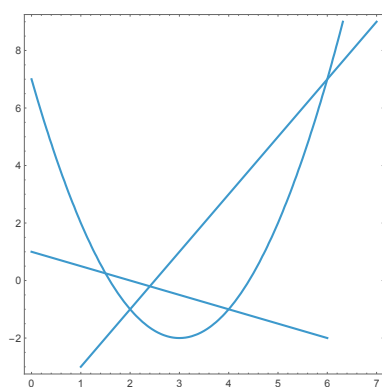
Los casos de aplicación presentados a continuación verifican la funcionalidad de **ExportConicsToTikZ** mediante composiciones algebraicas de complejidad creciente. La selección progresa desde configuraciones elementales -rectas y parábolas individuales- hasta sistemas multiecuación que combinan círculos, elipses, hipérbolas y parábolas con orientaciones variadas.

Cada ejemplo incorpora asignación aleatoria de atributos estilísticos (colores y patrones de trazo) para demostrar la fortaleza del sistema ante configuraciones estéticas diversas. Los parámetros de límites cartesianos ($x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$) se especifican explícitamente en algunos ejemplos con la finalidad de ilustrar el control manual de la ventana de visualización, mientras que las anotaciones textuales posicionadas en coordenadas específicas evidencian las capacidades de etiquetado.

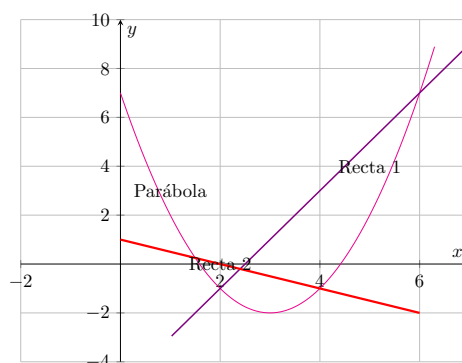
La secuencia de ejemplos expuesta a continuación examina particularmente:

- **Ejemplo 81:** Intersección de rectas con parábola - caso fundamental de geometría analítica.
- **Ejemplo 82:** Círculos y elipses concéntricos - simulación esquemática de órbitas planetarias con método paramétrico óptimo.
- **Ejemplo 83:** Familia de parábolas con orientaciones mixtas (verticales y horizontales) - prueba de detección automática de orientación y ordenamiento adaptativo.
- **Ejemplo 84:** Sistema heterogéneo de siete ecuaciones simultáneas - círculos desplazados, hipérbolas, parábola y recta en composición única que evalúa las capacidades multiobjeto.

Los ejemplos son autónomos y ejecutables independientemente, generando archivos *.tex* que compilan sin intervención manual y demuestran la traducción algebraica directa desde especificaciones de `ContourPlot` hacia instrucciones `PGFPlots` estructuradas.



(a) Gráfica de **Mathematica**



(b) Gráfica **TikZ**

Figura 11.1: Gráficas del ejemplo 81

Ejemplo 81 Rectas y parábola

```
Show[ContourPlot[y = (x - 3)^2 - 2, {x, 0, 7}, {y, -3, 9}],
ContourPlot[y = 2*x - 5, {x, 0, 7}, {y, -3, 10}],
ContourPlot[y = -0.5*x + 1, {x, 0, 6}, {y, -3, 5}], PlotStyle -> All]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
"cyan", "magenta", "black"}]
{Row[{"grid=", grid = True}],
Row[{"estilos=", estilos = Table[{line[], color[]}, 3]}],
Row[{"ejes=", ejes = True}],
Row[{"texto=",
texto = {{ "Parábola", {1, 3}}, {"Recta 1", {5,
4}}, {"Recta 2", {2, 0}}}], Row[{"xmin=", xmin = -2}],
Row[{"xmax=", xmax = 7}], Row[{"ymin=", ymin = -4}],
Row[{"ymax=", ymax = 10}]}
```

```
ExportConicsToTikZ[{{y = (x - 3)^2 - 2, {x, 0, 7}, {y, -3, 9}}, {y =
  2*x - 5, {x, 0, 7}, {y, -3, 10}}, {y = -0.5*x + 1, {x, 0,
    6}, {y, -3, 5}}}, "ExportConicsToTikZ1.tex", grid, estilos, ejes,
texto, xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Mathematica** se muestra como salida la gráfica de la subfigura 11.1a y los mensajes de proceso:

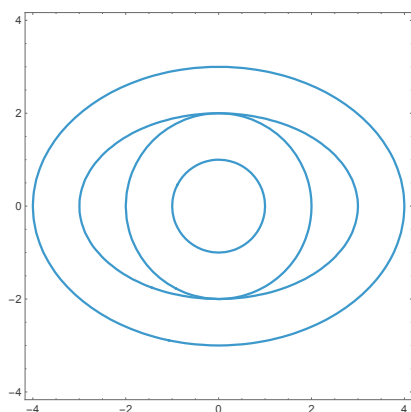
```
{grid= True, estilos= {{thin, magenta}, {solid, purple}, {thick, red}}, ejes= True,
texto= {{Parábola, {1, 3}}, {Recta 1, {5, 4}}, {Recta 2, {2, 0}}}, xmin= -2, xmax=
7, ymin= -4, ymax= 10}
```

Usando editores ya detectados: 4 disponibles

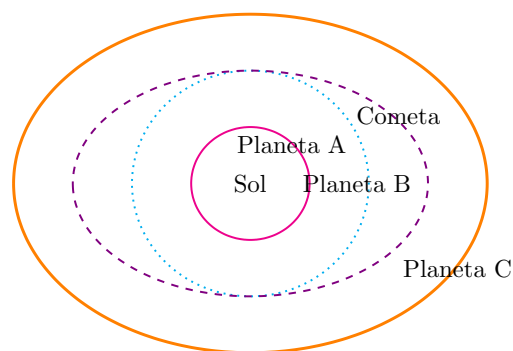
⋮

```
PDF generado: /Users/.../Downloads/ExportConicsToTikZ1/ExportConicsToTikZ1.pdf
/Users/.../Downloads/ExportConicsToTikZ1/ExportConicsToTikZ1.pdf
```

El archivo *ExportConicsToTikZ1.tex* (guardado en *Downloads/ExportConicsToTikZ1*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 11.1b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 11.2: Gráficas del ejemplo 82

Ejemplo 82 Círculos y elipses concéntricas simulando órbitas planetarias

```
Show[ContourPlot[x^2 + y^2 = 1, {x, -4, 4}, {y, -4, 4}],
ContourPlot[x^2 + y^2 = 4, {x, -4, 4}, {y, -4, 4}],
ContourPlot[x^2/9 + y^2/4 = 1, {x, -4, 4}, {y, -4, 4}],
ContourPlot[x^2/16 + y^2/9 = 1, {x, -5, 5}, {y, -4, 4}],
PlotStyle -> All]
```

```
line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"estilos=", estilos = Table[{line[], color[]}, 4]}],
Row[{"ejes=", ejes = RandomChoice[{True, False}]}],
Row[{"texto=",
  texto = {{ "Sol", {0, 0}}, {"Planeta A", {0.7,
    0.7}}, {"Planeta B", {1.8, 0}}, {"Cometa", {2.5,
```

```

1.2}}, {"Planeta C", {3.5, -1.5}}}], Row[{"xmin=",
xmin = -5}],
Row[{"xmax=", xmax = 5}], Row[{"ymin=", ymin = -4}],
Row[{"ymax=", ymax = 4}]
ExportConicsToTikZ[{{x^2 + y^2 = 1, {x, -4, 4}, {y, -4,
4}}, {x^2 + y^2 = 4, {x, -4, 4}, {y, -4, 4}}, {x^2/9 + y^2/4 =
1, {x, -4, 4}, {y, -4, 4}}, {x^2/16 + y^2/9 = 1, {x, -5,
5}, {y, -4, 4}}], "ExportConicsToTikZ2.tex", grid, estilos, ejes,
texto, xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram Mathematica** se retorna como salida la gráfica de la subfigura 11.2a y los mensajes de texto:

```

{grid= True, estilos= {{solid, magenta}, {dotted, cyan}, {dashed, purple}, {thick,
orange}}, ejes= False, texto= {{Sol, {0, 0}}, {Planeta A, {0.7, 0.7}}, {Planeta
B, {1.8, 0}}, {Cometa, {2.5, 1.2}}, {Planeta C, {3.5, -1.5}}}, xmin= -5, xmax= 5,
ymin= -4, ymax= 4}

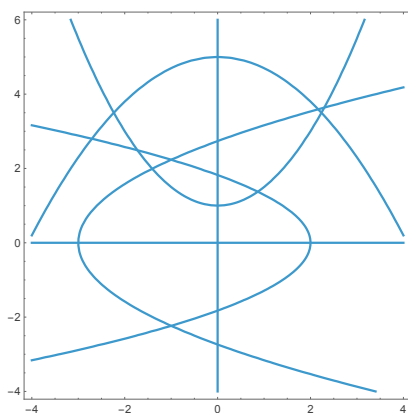
```

Usando editores ya detectados: 4 disponibles

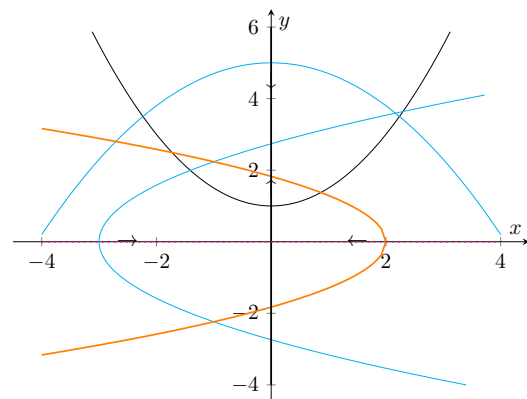
⋮

PDF generado: /Users/.../Downloads/ExportConicsToTikZ2/ExportConicsToTikZ2.pdf
 /Users/.../Downloads/ExportConicsToTikZ2/ExportConicsToTikZ2.pdf

El archivo *ExportConicsToTikZ2.tex* (guardado en *Downloads/ExportConicsToTikZ2*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 11.2b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 11.3: Gráficas del ejemplo 83

Ejemplo 83 Familia de parábolas

```

Show[ContourPlot[y = 0.5*x^2 + 1, {x, -4, 4}, {y, -4, 6}],
ContourPlot[y = -0.3*x^2 + 5, {x, -4, 4}, {y, -4, 6}],
ContourPlot[x = 0.4*y^2 - 3, {x, -4, 4}, {y, -4, 6}],
ContourPlot[x = -0.6*y^2 + 2, {x, -4, 4}, {y, -4, 6}],
ContourPlot[x = 0, {x, -4, 4}, {y, -4, 6}],
ContourPlot[y = 0, {x, -4, 4}, {y, -4, 6}], PlotStyle -> All]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=

```

```

RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estilos=", estilos = Table[{line[], color[]}, 6]}],
  Row[{"ejes=", ejes = RandomChoice[{True, False}]}],
  Row[{"texto=",
    texto = {{↑, {0, 1.5}}, {↓, {0,
      4.5}}, {→, {-2.5, 0}}, {←, {1.5, 0}}]}],
  Row[{"xmin=", xmin = -4.5}], Row[{"xmax=", xmax = 4.5}],
  Row[{"ymin=", ymin = -4.5}], Row[{"ymax=", ymax = 6.5}]}
ExportConicsToTikZ[{y = 0.5*x^2 + 1, {x, -4, 4}, {y, -4,
  6}}, {y = -0.3*x^2 + 5, {x, -4, 4}, {y, -4, 6}}, {x =
  0.4*y^2 - 3, {x, -4, 4}, {y, -4, 6}}, {x = -0.6*y^2 + 2, {x, -4,
  4}, {y, -4, 6}}, {x = 0, {x, -4, 4}, {y, -4, 6}}, {y =
  0, {x, -4, 4}, {y, -4, 6}}], "ExportConicsToTikZ3.tex", grid,
estilos, ejes, texto, xmin, xmax, ymin];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram** se obtiene como salida la gráfica de la subfigura 11.3a y los mensajes:

```

{grid= False, estilos= {{thin, black}, {thin, cyan}, {thin, cyan}, {solid, orange},
{solid, black}, {dotted, magenta}}, ejes= True, texto= {{↑, {0, 1.5}}, {↓, {0,
4.5}}, {→, {-2.5, 0}}, {←, {1.5, 0}}}, xmin= -4.5, xmax= 4.5, ymin= -4.5, ymax=
6.5}

```

Usando editores ya detectados: 4 disponibles

:

```

PDF generado: /Users/.../Downloads/ExportConicsToTikZ3/ExportConicsToTikZ3.pdf
/Users/.../Downloads/ExportConicsToTikZ3/ExportConicsToTikZ3.pdf

```

El archivo *ExportConicsToTikZ3.tex* (guardado en *Downloads/ExportConicsToTikZ3*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 11.3b.

Ejemplo 84 Varias cónicas

```

eq1 = {x^2 + y^2 = 1, {x, -10, 10}, {y, -10, 10}};
eq2 = {x^2 + y^2 = 2*x + 3*y, {x, -10, 10}, {y, -10, 10}};
eq3 = {x^2 - (y - 1)^2 = 1, {x, -10, 10}, {y, -10, 10}};
eq4 = {x^2 + y^2 = 4 x, {x, -10, 10}, {y, -10, 10}};
eq5 = {x^2 + y - x - 3 = 0, {x, -10, 10}, {y, -10, 10}};
eq6 = {x + y = 3, {x, -10, 10}, {y, -10, 10}};
eq7 = {x^2 - (y - 1)^2 = 4, {x, -10, 10}, {y, -10, 10}};

Show[ContourPlot[x^2 + y^2 = 1, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x^2 + y^2 = 2*x + 3*y, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x^2 - (y - 1)^2 = 1, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x^2 + y^2 = 4 x, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x^2 + y - x - 3 = 0, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x + y = 3, {x, -10, 10}, {y, -10, 10}],
ContourPlot[x^2 - (y - 1)^2 = 4, {x, -10, 10}, {y, -10, 10}],
PlotStyle → All]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
{Row[{"grid=", grid = RandomChoice[{True, False}]}],

```

```

Row[{"estilos=", estilos = Table[{line[], color[]}, 7]}],
Row[{"ejes=", ejes = RandomChoice[{True, False}]}],
Row[{"texto=", texto = {"Cónicas", {3, 2}}}],
Row[{"xmin=", xmin = -10.5}], Row[{"xmax=", xmax = 10.5}],
Row[{"ymin=", ymin = -10.5}], Row[{"ymax=", ymax = 10.5}]
ExportConicsToTikZ[{eq1, eq2, eq3, eq4, eq5, eq6, eq7},
"ExportConicsToTikZ4.tex", grid, estilos, ejes, texto, xmin, xmax,
ymin];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram Mathematica** la salida devuelve la gráfica de la subfigura 11.4a y los mensajes de proceso:

```

{grid= False, estilos= {{thin, magenta}, {dotted, blue}, {dotted, yellow}, {dotted,
red}, {thin, cyan}, {thin, orange}, {dashed, magenta}}, ejes= False, texto=
{{Cónicas, {3, 2}}, xmin= -10.5, xmax= 10.5, ymin= -10.5, ymax= 10.5}

```

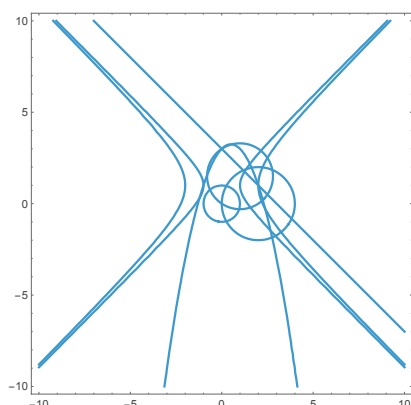
Usando editores ya detectados: 4 disponibles

⋮

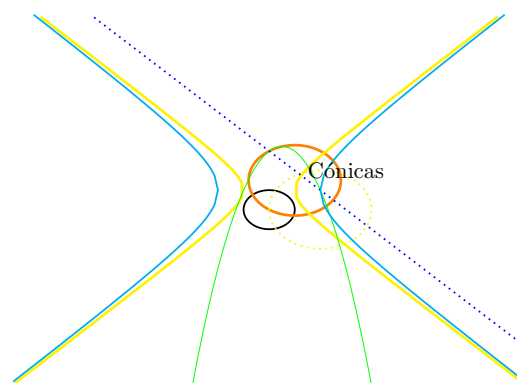
PDF generado: /Users/.../Downloads/ExportConicsToTikZ4/ExportConicsToTikZ4.pdf

/Users/.../Downloads/ExportConicsToTikZ4/ExportConicsToTikZ4.pdf

El file *ExportConicsToTikZ4.tex* (guardado en *Downloads/ExportConicsToTikZ4*) comparte el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 11.4b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 11.4: Gráficas del ejemplo 84

Para más ejemplos

Si el lector requiere obtener un conjunto más amplio de ejemplos relacionados con el comando **ExportConicsToTikZ**, puede consultar de manera complementaria el archivo: 11. *Ejemplos Export-ConicsToTikZ.nb*, disponible en el enlace de descarga del paquete $\text{\LaTeX}$ (ver página xiv).

¿Qué aprenderás en este capítulo?

- ▶ Generar diagramas de *Venn* de 2, 3 y 4 conjuntos con operaciones lógicas.
- ▶ Exportar diagramas de árbol para estructuras jerárquicas y decisiones.
- ▶ Visualizar problemas de teoría de conjuntos y lógica proposicional.
- ▶ Visualizar problemas de programación lineal con regiones factibles.
- ▶ Aplicar herramientas especializadas para necesidades pedagógicas específicas.

El paquete **ViT_EX** incorpora comandos auxiliares para visualizaciones específicas frecuentes en matemática discreta y optimización. Estos comandos generan representaciones gráficas directamente en **Mathematica** o exportan código **TikZ** estructurado, según el caso de uso. Consideremos la descripción de estas instrucciones particulares.

- **DiagramaFuncion[eA, eB, rel, opts]**: Construye diagramas de correspondencia entre conjuntos mediante elipses y flechas dirigidas. Visualiza funciones, relaciones o mapeos entre dominio (**eA**) y codominio (**eB**). Las relaciones se especifican como reglas (**a → b**). Opciones configurables: separación entre conjuntos, radios de elipses, posicionamiento de texto, visualización de arco superior representando *f* y tamaño de la imagen. Retorna objeto **Graphics** nativo de **Wolfram**, exportable a gráfica **TikZ** mediante el comando **ExportToTikZ** explicado en el capítulo 4.
- **DiagramaCuadricula[labels, opts]**: Genera diagramas organizados en cuadrícula rectangular con cajas etiquetadas y conexiones opcionales. Configurable mediante opciones de: dimensiones de grilla (filas × columnas), tamaño de cajas, separación inter-caja, posición inicial, estilos de borde/relleno, formato de etiquetas y conexiones horizontales con flechas. Las flechas conectan únicamente cajas adyacentes en la misma fila, sin conexiones verticales entre filas. Esta sentencia es útil para diagramas de flujo, organigramas y representaciones de procesos secuenciales. Retorna un objeto **Graphics** exportable a código **TikZ** mediante el comando **ExportToTikZ**.
- **InterSets[filename, labels, fillColor, scale, showIntersectionLabel, ellipseSize]**: Exporta diagramas de intersección de conjuntos como código **TikZ/L^AT_EX**. Crea un sombreado automático de la región de intersección de **todos** los conjuntos especificados. Disposición geométrica adaptativa según la cantidad: horizontal (2 conjuntos), triangular (3), cuadrada (4), pentagonal (5), circular (6+). Mapeo automático de colores a tonalidades **TikZ** apropiadas con transparencia. Genera un documento *standalone* en *Downloads/InterSets*. Retorna ruta del archivo exportado.

- **UnionSets[filename, labels, fillColors, scale, showUnionLabel, ellipseSize]**: Análogo a **InterSets** pero visualiza la unión de conjuntos. Sombrea cada conjunto con color independiente especificado en **fillColors**. Manejo automático de superposiciones mediante transparencia. Etiqueta opcional de unión en notación matemática. Exporta el archivo *.tex* a *Downloads/UnionSets*.
- **DifferenceSets[filename, labels, fillColorA, fillColorB, scale, showDifferenceLabel, ellipseSize]**: Visualiza la diferencia de conjuntos $A \setminus B$. Sombrea la región perteneciente al primer conjunto pero no al segundo. Requiere exactamente 2 conjuntos. Crea colores diferenciados para cada conjunto (**fillColorA**, **fillColorB**). Exporta a *Downloads/DifferenceSets*.
- **SymmetricDif[filename, labels, fillColors, scale, showSymmetricLabel, ellipseSize]**: Representa la diferencia simétrica $(A \setminus B) \cup (B \setminus A)$ o $(A \cup B) \setminus (A \cap B)$. Sombrea regiones exclusivas de cada conjunto (elementos en uno pero no en ambos). Maneja colores independientes por conjunto. Exporta a *Downloads/SymmetricDif*.
- **UnionComplement[filename, labels, fillColor, scale, showComplementLabel, ellipseSize]**: Visualiza el complemento de la unión respecto al universo (región exterior a todos los conjuntos dentro del rectángulo universal). Sombrea el área del rectángulo no cubierta por ninguna elipse. Resulta útil para representar $(A \cup B)^c$. Exporta a *Downloads/UnionComplement*.
- **LinearPro[filename, objective, constraints, xRange, yRange, fillColor, showLabels, showOptimalPoint, scale, textSize]**: Resuelve y visualiza problemas de programación lineal bidimensional. Se especifica el problema como {"max"|"min", "expresion"} y las restricciones como una lista de *strings* mediante desigualdades. Calcula automáticamente la región factible mediante la intersección de las restricciones, identifica vértices, ejecuta la optimización con **NMaximize**/**NMinimize**, genera un diagrama *TikZ* con la región sombreada, las líneas de restricción, el punto óptimo marcado y el valor objetivo anotado. Maneja casos especiales, tales como: soluciones no acotadas (∞) y regiones vacías (sin solución). Exporta el archivo producido a *Downloads/LinearPro*. Retorna la ruta del *file* e imprime punto y valor óptimo en **Wolfram Mathematica**.
- **CrearTablaSignosTeX[desigualdad, variable, nombreArchivo, copy]**: Genera tablas de signos en $\LaTeX$ para resolver desigualdades algebraicas racionales. Analiza automáticamente la expresión mediante factorización de numeradores y denominadores, identifica puntos críticos (ceros e indefiniciones), maneja multiplicidades pares e impares correctamente, y construye intervalos de evaluación. La tabla resultante incluye separadores visuales diferenciados: barras simples para ceros del numerador y barras dobles para puntos de indefinición. Resuelve la desigualdad considerando inclusión/exclusión apropiada de puntos críticos y presenta la solución en notación de intervalos. Valida automáticamente que la expresión sea algebraica racional (rechaza funciones trascendentes y raíces con variable). Exporta a un documento con clase *article* completo en *Downloads/nombreArchivo* con la desigualdad original, la tabla de signos, la solución final y los puntos de indefinición. El parámetro opcional **copy** permite copiar el código $\LaTeX$ al portapapeles además de generar el archivo *.tex*.

Características comunes de funciones de exportación:

- Validación estricta de argumentos con retorno **\$Failed** ante errores.
- Creación automática de estructura de directorios si no existe.
- Documentos $\LaTeX$ con clase *standalone* o *article* listos para compilar.
- Inclusión de paquetes necesarios (*tikz*, *amsmath* y bibliotecas específicas).

12.1 Ejemplos de los casos especiales

Los ejemplos siguientes demuestran aplicaciones concretas de las utilidades especializadas en escenarios representativos de matemática discreta y optimización. A diferencia de las funciones principales de exportación gráfica del paquete **W_{IT}E_X** que procesan estructuras matemáticas generales, estos comandos abordan representaciones visuales específicas con sintaxis y semántica propias.

La secuencia ilustra tanto invocaciones básicas con parámetros mínimos como configuraciones elaboradas que explotan opciones avanzadas de personalización. Cada ejemplo ejecuta independientemente y genera salidas visuales mediante dos rutas: objetos **Graphics** nativos (para **DiagramaFuncion** y **DiagramaCuadrícula**) que se exportan posteriormente mediante **ExportToTikZ** o archivos *.tex* directos (para operaciones de conjuntos, programación lineal y tablas de signos) que compilan inmediatamente.

Categorías abordadas:

- **Ejemplos 85-86:** Diagramas de funciones/relaciones entre conjuntos finitos con configuración de geometría, etiquetas y espaciado.
- **Ejemplos 87-88:** Cuadrículas organizadas para diagramas de flujo y procesos secuenciales, con y sin conexiones direccionales.
- **Ejemplos 89-93:** Operaciones de teoría de conjuntos (intersección, unión, diferencia, diferencia simétrica, complemento) con variaciones en cantidad de conjuntos, colores y escalas.
- **Ejemplos 94-96:** Problemas de programación lineal bidimensional con objetivos de maximización y minimización, restricciones variadas y visualización de región factible.
- **Ejemplos 97-99:** Tablas de signos que resuelven distintas inecuaciones algebraicas racionales.

Los parámetros aleatorios empleados en algunos casos (estilos de línea y colores) validan la robustez ante configuraciones diversas. Todos los ejemplos invocan a **CompiladorTex** para la compilación automática del resultado generado.

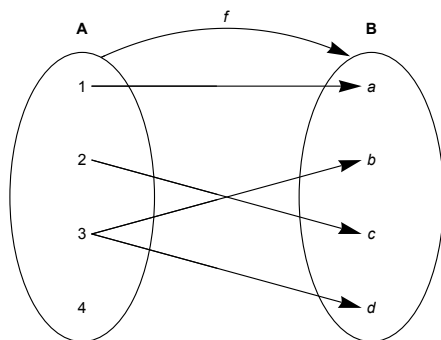
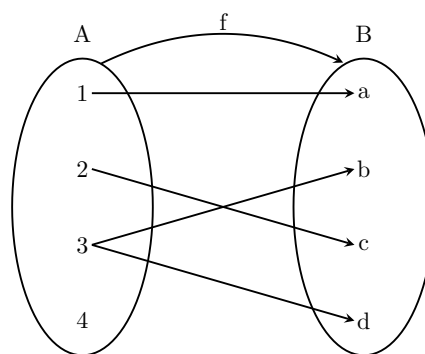
Ejemplo 85 *DiagramaFuncion* uso básico

```
CasoEspecial1 =
DiagramaFuncion[{1, 2, 3, 4}, {a, b, c, d}, {1 → a, 2 → c, 3 → b,
3 → d}]
ExportToTikZ[CasoEspecial1, "CasoEspecial1.tex", False, "black",
False];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** se muestra como salida la gráfica de la subfigura 12.1a y los mensajes de proceso:

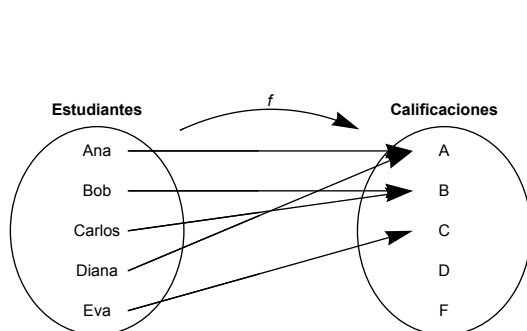
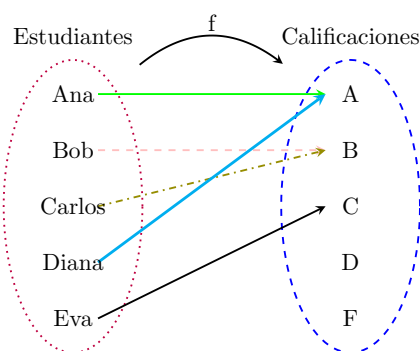
```
Textos extraídos automáticamente: 11 elementos
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/CasoEspecial1/CasoEspecial1.pdf
/Users/.../Downloads/CasoEspecial1/CasoEspecial1.pdf
```

El archivo *CasoEspecial1.tex* (guardado en *Downloads/CasoEspecial1*) contiene el código **W_{IT}E_X** editable que produce la gráfica **TikZ** de la subfigura 12.1b.

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 12.1: Gráficas del ejemplo 85

(a) Gráfica de **Mathematica**

(b) Gráfica TikZ

Figura 12.2: Gráficas del ejemplo 86

Ejemplo 86 *DiagramaFuncion* estudiantes y sus calificaciones

```

CasoEspecial2 =
DiagramaFuncion[{"Ana", "Bob", "Carlos", "Diana", "Eva"}, {"A", "B",
  "C", "D", "F"}, {"Ana" → "A", "Bob" → "B", "Carlos" → "B",
  "Diana" → "A", "Eva" → "C"},
"Labels" → {"Estudiantes", "Calificaciones"}, "FontSize" → 12,
"EllipseRadii" → {2.5, 3}, "CenterGap" → 10, "TextGapLeft" → 0.9,
"TextGapRight" → 0.9]
ExportToTikZ[CasoEspecial2, "CasoEspecial2.tex", False,
Table[RandomChoice[{"solid", "dashed", "dotted", "dashdotted",
  "thick"}],
  RandomChoice[{"red", "blue", "green", "orange", "purple", "black",
    "brown", "pink", "gray", "cyan", "magenta", "yellow", "lime",
    "olive", "teal"}]], {i, 6}], False];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram Mathematica** se obtiene como salida la gráfica de la subfigura 12.2a y los mensajes de texto:

Textos extraídos automáticamente: 13 elementos

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/CasoEspecial2/CasoEspecial2.pdf

/Users/.../Downloads/CasoEspecial2/CasoEspecial2.pdf

El archivo *CasoEspecial2.tex* (guardado en *Downloads/CasoEspecial2*) comparte el código $\text{\LaTeX}$ editable que genera la gráfica $\text{\TikZ}$ de la subfigura 12.2b.

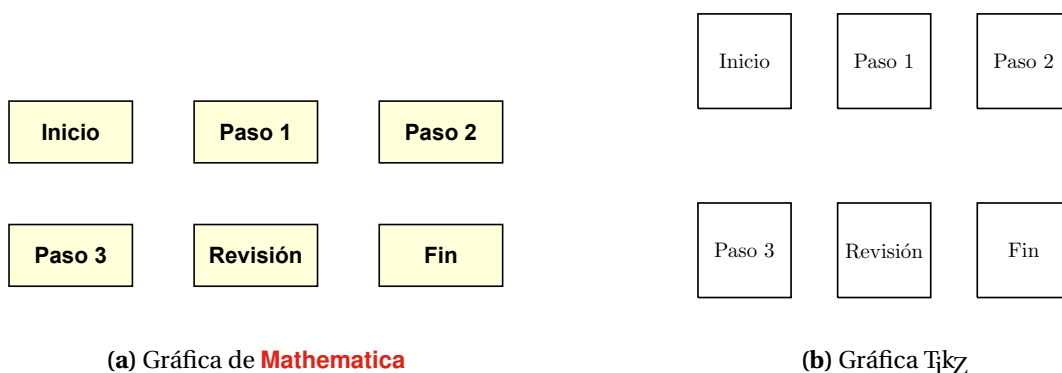


Figura 12.3: Gráficas del ejemplo 87

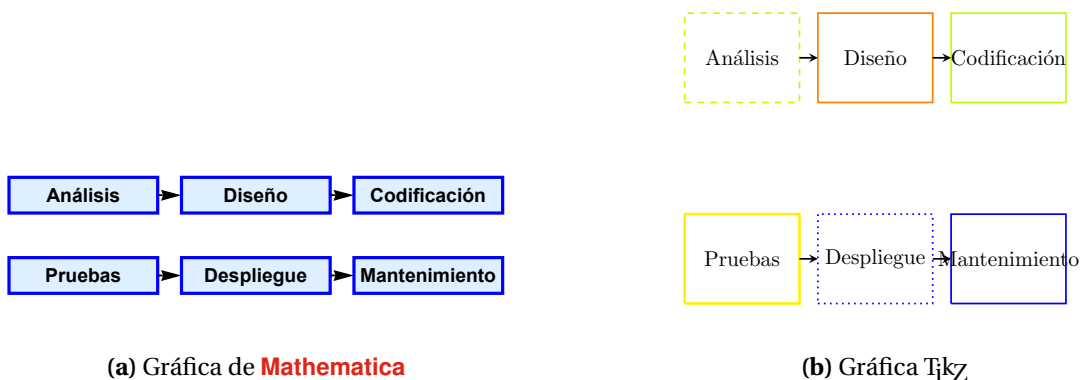


Figura 12.4: Gráficas del ejemplo 88

Ejemplo 87 *DiagramaCuadrícula* uso básico

```
labels = {"Inicio", "Paso 1", "Paso 2", "Paso 3", "Revisión", "Fin"};

CasoEspecial3 =
DiagramaCuadrícula[labels, "Grid" → {2, 3}, "BoxSize" → {2, 1},
"Gap" → {1, 1}, "Connect" → False]
ExportToTikZ[CasoEspecial3, "CasoEspecial3.tex", False, "black",
False];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram** se retorna como salida la gráfica de la subfigura 12.3a y los mensajes:

Textos extraídos automáticamente: 6 elementos

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/CasoEspecial3/CasoEspecial3.pdf
 /Users/.../Downloads/CasoEspecial3/CasoEspecial3.pdf

El archivo *CasoEspecial3.tex* (guardado en *Downloads/CasoEspecial3*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 12.3b.

Ejemplo 88 DiagramaCuadrícula proceso de desarrollo de software

```
CasoEspecial4 =
DiagramaCuadrícula[{"Análisis", "Diseño", "Codificación", "Pruebas",
  "Despliegue", "Mantenimiento"}, "Grid" → {2, 3},
"BoxSize" → {5, 1.2}, "Gap" → {0.8, 1.5},
"BoxStyle" → {EdgeForm[{Thick, Blue}], FaceForm[LightBlue]},
"LabelStyle" → Directive[Black, 12, Bold], "Connect" → True,
"ArrowStyle" → {Thick, Blue}]
ExportToTikZ[CasoEspecial4, "CasoEspecial4.tex", False,
Table[{RandomChoice[{"solid", "dashed", "dotted", "dashdotted",
  "thick"}],
  RandomChoice[{"red", "blue", "green", "orange", "purple", "black",
    "brown", "pink", "gray", "cyan", "magenta", "yellow", "lime",
    "olive", "teal"}]}, 6], False];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** la salida obtenida despliega la gráfica de la subfigura 12.4a y los mensajes:

Textos extraídos automáticamente: 6 elementos

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/CasoEspecial4/CasoEspecial4.pdf
 /Users/.../Downloads/CasoEspecial4/CasoEspecial4.pdf

El archivo *CasoEspecial4.tex* (guardado en *Downloads/CasoEspecial4*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 12.4b.

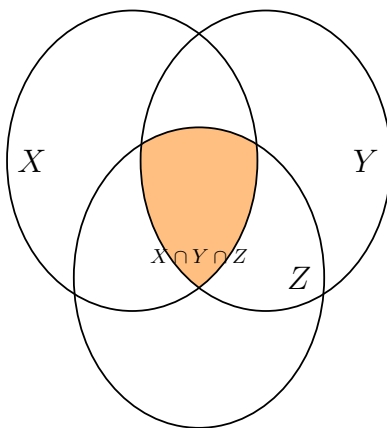


Figura 12.5: Gráfica $\text{\LaTeX}$ del ejemplo 89

Ejemplo 89 *InterSets* con etiqueta de intersección

```
InterSets["CasoEspecial5.tex", {"X", "Y", "Z"}, "orange", 1.3, True];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram** la salida retorna los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/InterSets/CasoEspecial5.pdf

/Users/.../Downloads/InterSets/CasoEspecial5.pdf

El archivo *CasoEspecial5.tex* (guardado en *Downloads/InterSets*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la figura 12.5.

147

Ejemplo 90 *UnionComplement* con el conjunto universo más grande

```
UnionComplement["CasoEspecial6.tex", {"P", "Q", "R"}, "cyan", 1.5,
True, {1.5, 1.8}, {5, 4}];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** se retorna como salida los mensajes de texto:

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/UnionComplement/CasoEspecial6.pdf

/Users/.../Downloads/UnionComplement/CasoEspecial6.pdf

El archivo *CasoEspecial6.tex* (guardado en *Downloads/UnionComplement*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la figura 12.6.

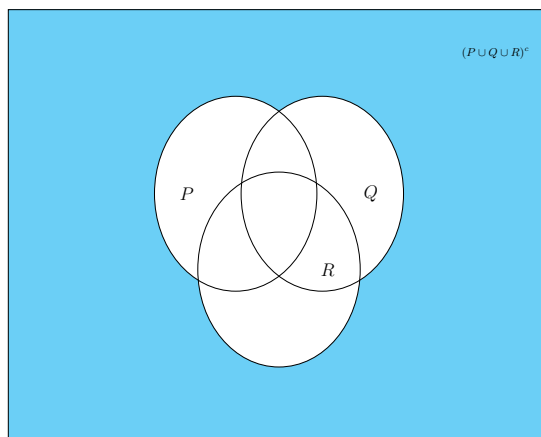


Figura 12.6: Gráfica $\text{\TikZ}$ del ejemplo 90

Ejemplo 91 *UnionSets* con etiqueta de unión

```
UnionSets["CasoEspecial7.tex", {"X", "Y"}, "green", 1.2, True];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** la salida comparte los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/UnionSets/CasoEspecial7.pdf

/Users/.../Downloads/UnionSets/CasoEspecial7.pdf

El archivo *CasoEspecial7.tex* (guardado en *Downloads/UnionSets*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la figura 12.7.

$X \cup Y$

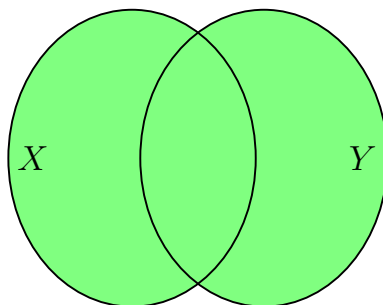


Figura 12.7: Gráfica $\text{\TikZ}$ del ejemplo 91

Ejemplo 92 *DifferenceSets* con elipses más pequeñas

```
DifferenceSets["CasoEspecial8.tex", {"P", "Q", "R"}, "orange", 1.5,
True, {1, 1.2}];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** el *Out* retorna los mensajes de texto:

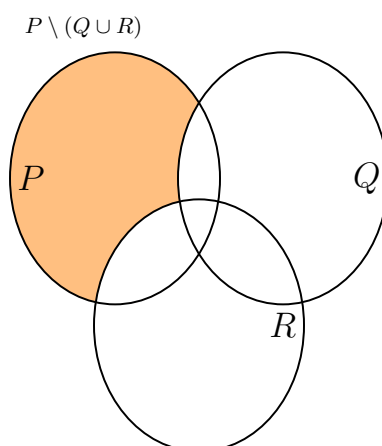
Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/DifferenceSets/CasoEspecial8.pdf

/Users/.../Downloads/DifferenceSets/CasoEspecial8.pdf

El archivo *CasoEspecial8.tex* (guardado en *Downloads/DifferenceSets*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 12.8.

Figura 12.8: Gráfica T_{ik}Z del ejemplo 92**Ejemplo 93** *SymmetricDif* con etiqueta de diferencia simétrica

```
SymmetricDif["CasoEspecial9.tex", {"X", "Y"}, "green", 1.5, True];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Mathematica** se muestra como salida los mensajes de proceso:

Usando editores ya detectados: 4 disponibles

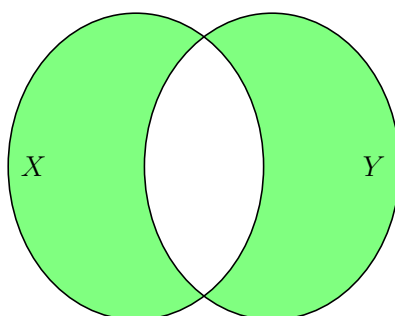
⋮

PDF generado: /Users/.../Downloads/SymmetricDif/CasoEspecial9.pdf

/Users/.../Downloads/SymmetricDif/CasoEspecial9.pdf

El archivo *CasoEspecial9.tex* (guardado en *Downloads/SymmetricDif*) contiene el código L^AT_EX e-ditable que produce la gráfica T_{ik}Z de la subfigura 12.9.

$$X \Delta Y$$

Figura 12.9: Gráfica T_{ik}Z del ejemplo 93

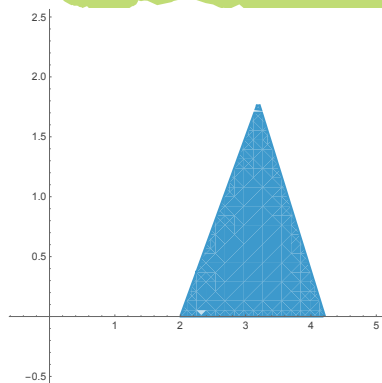
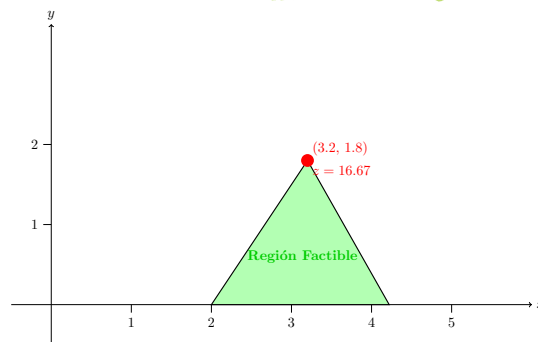
(a) Gráfica de **Mathematica**(b) Gráfica **TikZ**

Figura 12.10: Gráficas del ejemplo 94

Ejemplo 94 *LinearPro* datos con decimales

```

RegionPlot[
  3.7 x + 2.1 y <= 15.6 && 1.2 x - 0.8 y >= 2.4 && x >= 0 &&
  y >= 0, {x, -0.5, 5}, {y, -0.5, 2.5}, Frame ->False, Axes ->True]
NMaximize[{2.8 x + 4.3 y,
  3.7 x + 2.1 y <= 15.6 && 1.2 x - 0.8 y >= 2.4 && x >= 0 &&
  y >= 0}, {x, y}]
LinearPro[
  "CasoEspecial10.tex", {"3.7x + 2.1y <= 15.6", "1.2x - 0.8y >= 2.4",
    "x >= 0", "y >= 0"}, {"max", "2.8x + 4.3y"}, "green", 2.0, True,
  True, "normal", {0, 5}, {0, 2.5}];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Wolfram** el *Out* genera la gráfica de la subfigura 12.10a y los mensajes:

```
{16.673, {x -> 3.19708, y -> 1.79562}}
```

```
✓Punto óptimo: {3.19708, 1.79562}
```

```
✓Valor óptimo: 16.673
```

```
Usando editores ya detectados: 4 disponibles
```

```
⋮
```

```
PDF generado: /Users/.../Downloads/LinearPro/CasoEspecial10.pdf
```

```
/Users/.../Downloads/LinearPro/CasoEspecial10.pdf
```

El archivo *CasoEspecial10.tex* (guardado en *Downloads/LinearPro*) comparte el código $\text{\LaTeX}$ editable que produce la gráfica **TikZ** de la subfigura 12.10b.

Ejemplo 95 *LinearPro* ejemplo de cálculo de mínimo

```

RegionPlot[
  x + y >= 4 && 2 x + y >= 6 && x >= 0 && y >= 0, {x, -0.5,
    10}, {y, -0.5, 8}, Frame ->False, Axes ->True]
NMinimize[{x + 2 y,
  x + y >= 4 && 2 x + y >= 6 && x >= 0 && y >= 0}, {x, y}]
LinearPro[
  "CasoEspecial11.tex", {"x + y >= 4", "2x + y >= 6", "x >= 0",
    "y >= 0"}, {"min", "x + 2y"}, "orange", 2.0, True, True, "huge"];

```

```
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** se despliega como salida la gráfica de la subfigura 12.11a y los mensajes de texto:

```
{4., {x → 4., y → 0.}}
```

```
✓Punto óptimo: {4., 0.}
```

```
✓Valor óptimo: 4.
```

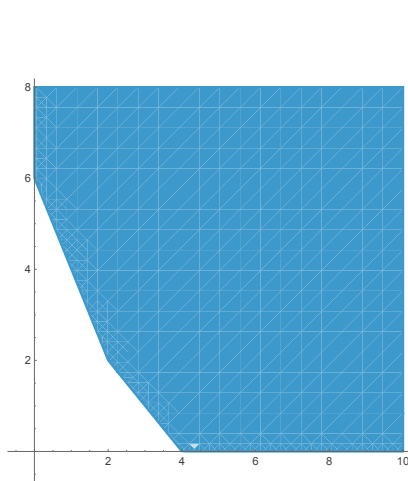
```
Usando editores ya detectados: 4 disponibles
```

```
⋮
```

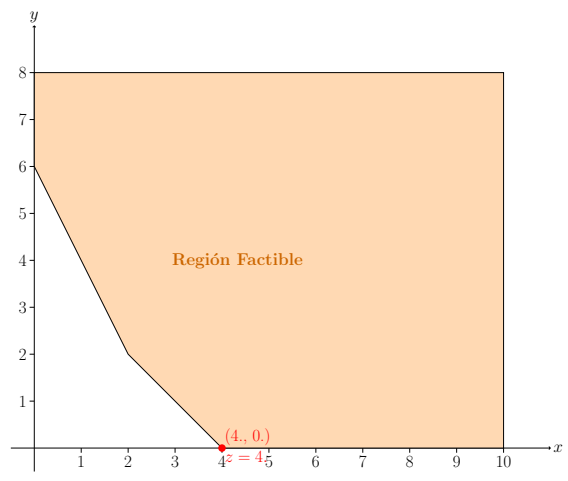
```
PDF generado: /Users/.../Downloads/LinearPro/CasoEspecial11.pdf
```

```
/Users/.../Downloads/LinearPro/CasoEspecial11.pdf
```

El archivo *CasoEspecial11.tex* (guardado en *Downloads/LinearPro*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 12.11b.

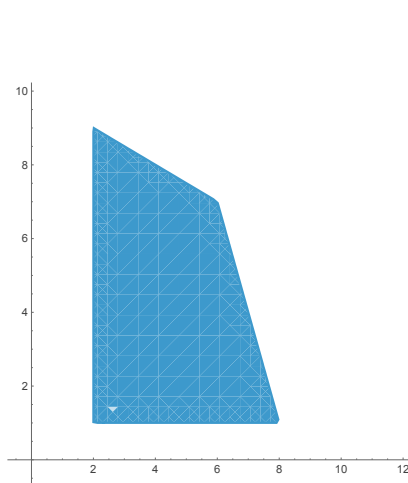


(a) Gráfica de **Mathematica**

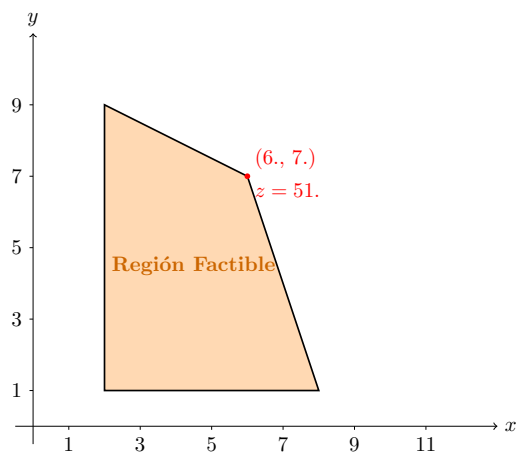


(b) Gráfica $\text{\LaTeX}$

Figura 12.11: Gráficas del ejemplo 95



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 12.12: Gráficas del ejemplo 96

Ejemplo 96 *LinearPro* usando varias opciones

```

RegionPlot[
  x + 2 y <= 20 && 3 x + y <= 25 && x >= 2 && y >= 1, {x, -0.5,
    12}, {y, -0.5, 10}, Frame → False, Axes → True]
NMaximize[{5 x + 3 y,
  x + 2 y <= 20 && 3 x + y <= 25 && x >= 2 && y >= 1}, {x, y}]
LinearPro[
  "CasoEspecial12.tex", {"x + 2y <= 20", "3x + y <= 25", "x >= 2",
    "y >= 1"}, {"max", "5x + 3y"}, "orange", 0.6, True, True,
  "normal", {0, 12}, {0, 10}];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram** la salida retorna la gráfica de la subfigura 12.12a y los mensajes de proceso:

```

{51., {x → 6., y → 7.}}
✓Punto óptimo: {6., 7.}
✓Valor óptimo: 51.
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/LinearPro/CasoEspecial12.pdf
/Users/.../Downloads/LinearPro/CasoEspecial12.pdf

```

El archivo *CasoEspecial12.tex* (guardado en *Downloads/LinearPro*) posee el código $\text{\LaTeX}$ editable que genera la gráfica $\text{\LaTeX}$ de la subfigura 12.12b.

	$-\infty$	-2	-1	1	$+\infty$
$x-1$	—	—	—	0	+
$x+1$	—	—	0	+	+
$x+2$	—	0	+	+	+
Signo	—	+	—	+	+

Desigualdad original: $x^3 + 2x^2 - x - 2 \geq 0$

Solución: $x \in [-2, -1] \cup [1, +\infty[$

Figura 12.13: Tabla de signos del ejemplo 97

Ejemplo 97 *CrearTablaSignosTeX* con expresión polinomial

```

Reduce[x^3 + 2 x^2 - x - 2 >= 0, x, Reals]
CrearTablaSignosTeX[x^3 + 2 x^2 - x - 2 >= 0, x, "CasoEspecial13.tex"];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Mathematica** la salida genera los mensajes de texto:

```

-2 <= x <= -1 || x >= 1
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/CasoEspecial13/CasoEspecial13.pdf
/Users/.../Downloads/CasoEspecial13/CasoEspecial13.pdf

```

El archivo *CasoEspecial13.tex* (guardado en *Downloads/CasoEspecial13*) posee el código $\text{\LaTeX}$ editable de la desigualdad original, su solución y la tabla de signos compartida en la figura 12.13.

Ejemplo 98 *CrearTablaSignosTeX* con expresiones algebraicas fraccionarias

```
Reduce[((x - 2) (x + 1)) / ((x - 4) (x + 3)) <= (x + 5) / (x - 4), x, Reals]
CrearTablaSignosTeX[((x - 2) (x + 1)) / ((x - 4) (x + 3)) <= (x +
5) / (x - 4), x, "CasoEspecial14.tex"];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Mathematica** la salida produce los mensajes:

```
-3 < x <= -17/9 || x > 4
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/CasoEspecial14/CasoEspecial14.pdf
/Users/.../Downloads/CasoEspecial14/CasoEspecial14.pdf
```

El archivo *CasoEspecial14.tex* (guardado en *Downloads/CasoEspecial14*) contiene el código $\text{\LaTeX}$ editable de la desigualdad original, su solución, los puntos de indefinición y la tabla de signos expuesta en la figura 12.14.

	$-\infty$	-3	$-\frac{17}{9}$	4	$+\infty$
$-9x - 17$	+	+	0	-	-
$x - 4$	-	-	-	0	+
$x + 3$	-	0	+	+	+
Signo	+	-	+	-	-

Desigualdad original: $\frac{(x-2)(x+1)}{(x-4)(x+3)} \leq \frac{x+5}{x-4}$
 Solución: $x \in]-3, -\frac{17}{9}] \cup]4, +\infty[$
 Puntos de indefinición: $-3, 4$

Figura 12.14: Tabla de signos del ejemplo 98

	$-\infty$	-1.8	-0.9	$-\frac{1}{2}$	0.3	$\frac{1}{2}$	$\frac{3}{4}$	$+\infty$
$24x^3 + 56x^2 + 16x - 13$	-	0	+	0	-	-	0	+
$2x - 1$	-	-	-	-	-	0	+	+
$2x + 1$	-	-	-	0	+	+	+	+
$4x - 3$	-	-	-	-	-	-	0	+
Signo	+	-	+	-	+	-	+	+

Desigualdad original: $\frac{(2x+5)(3x-1)}{(x-\frac{1}{2})(4x-3)} > \frac{1}{2x+1}$
 Solución: $x \in]-\infty, -1.8[\cup]-0.9, -\frac{1}{2}[\cup]0.3, \frac{1}{2}[\cup]\frac{3}{4}, +\infty[$
 Puntos de indefinición: $-\frac{1}{2}, \frac{1}{2}, \frac{3}{4}$

Figura 12.15: Tabla de signos del ejemplo 99

Ejemplo 99 CrearTablaSignosTeX con ceros difíciles de calcular

```
Reduce[(3*x - 1)*(2*x + 5)/((4*x - 3)*(x - 1/2)) > 1/(2*x + 1), x, Reals]
CrearTablaSignosTeX[(3*x - 1)*(2*x + 5)/((4*x - 3)*(x - 1/2)) >
1/(2*x + 1), x, "CasoEspecial15.tex", True];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Mathematica** la salida genera los mensajes de texto:

```
x < Root[-13 + 16 # + ... & , 1, 0] || Root[-13 + 16 # + 56 ... & , 2, 0] < x <
-1/2 || Root[-13 + 16 # + 56 ... & , 3, 0] < x < 1/2 || x > 3/4
```

Código LaTeX copiado al portapapeles

Usando editores ya detectados: 4 disponibles

:

PDF generado: /Users/.../Downloads/CasoEspecial15/CasoEspecial15.pdf

/Users/.../Downloads/CasoEspecial15/CasoEspecial15.pdf

En este ejemplo se ha utilizado el atributo `copy = True` por lo que el código $\text{\LaTeX}$ con la desigualdad original, su solución, los puntos de indefinición y la tabla de signos mostrada en la figura 12.15, se ha copiado al portapapeles del ordenador del usuario con la intención de pegarlo directamente en otro archivo `.tex`. También, *CasoEspecial15.tex* (guardado en *Downloads/CasoEspecial15*) integra el código $\text{\LaTeX}$ editable.

Para más ejemplos

Los ejemplos socializados con anterioridad y otros, pueden ser consultados en el archivo: *12. Ejemplos CASOS ESPECIALES.nb*, disponible en el enlace de descarga del paquete **V_lTeX** (ver página xiv).

Función `ExportContour2DTikZ`

¿Qué aprenderás en este capítulo?

- ▶ Exportar curvas de nivel y mapas de contorno a formato `TikZ`.
- ▶ Configurar niveles de contorno, etiquetas y paletas de colores.
- ▶ Generar gráficos para análisis de superficies y optimización.

La representación bidimensional de curvas algebraicas, regiones de desigualdad y trayectorias paramétricas en documentos científicos frecuentemente requiere traducción desde entornos computacionales hacia código $\text{\LaTeX}$ manipulable. Si bien **Mathematica** ofrece los comandos `ContourPlot`, `RegionPlot` y `ParametricPlot` para visualización, la exportación a formatos *TikZ/PGFPlots* editables no existe nativamente.

La sentencia `ExportContour2DTikZ` del paquete **$\text{\LaTeX}$** resuelve esta carencia mediante procesamiento diferenciado según tipo de entrada: identifica automáticamente curvas implícitas (ecuaciones con igualdad), regiones de desigualdad (expresiones lógicas con operadores relacionales), funciones explícitas (especificaciones de dominio), trayectorias paramétricas (pares de funciones en parámetro) y conjuntos discretos de puntos. Para cada categoría aplica algoritmos especializados -muestreo adaptativo para implícitas, evaluación densa con transparencia para regiones, parametrización suave en curvas- y sintetiza bloques *addplot* con sintaxis *PGFPlots* optimizada.

El mecanismo de conversión ejecuta clasificación automática mediante análisis sintáctico: detecta operadores relacionales (`==`, `<=`, `>=`), identifica rangos explícitos de variables, discrimina entre funciones unidimensionales y paramétricas bidimensionales, resuelve expresiones implícitas mediante muestreo de contorno con tolerancia ajustable, genera **nubes de puntos** para regiones mediante evaluación sistemática con densidad configurable y ensambla código `TikZ` con límites cartesianos calculados automáticamente o especificados de forma manual.

La función `ExportContour2DTikZ` traduce especificaciones heterogéneas de gráficos bidimensionales -curvas implícitas, regiones de desigualdad, funciones explícitas, trayectorias paramétricas, conjuntos de puntos- en código *TikZ/PGFPlots* unificado, produciendo documentos $\text{\LaTeX}$ con clase *standalone*. A diferencia de exportaciones que consolidan gráficos como objetos binarios, `ExportContour2DTikZ` genera código fuente $\text{\LaTeX}$ textual completamente accesible.

La ejecución de `ExportContour2DTikZ` retorna un archivo *.tex* con preámbulo completo compilable mediante *pdflatex*. Examinemos la especificación argumental de este comando.

- **ExportContour2DToTikZ**[argumentos, archivo, rejilla, estilos, ejes, texto, muestras, tolerancia, opacidadRelleno, densidad, xmin, xmax, ymin, ymax]: Convierte especificaciones matemáticas bidimensionales heterogéneas elaboradas en **Wolfram Mathematica** hacia código TikZ/LaTeX plano y construye como resultado un archivo .tex compilable.

Parámetros:

- **argumentos** (obligatorio): Especificación matemática bidimensional o lista de especificaciones múltiples. Admite formatos heterogéneos con detección automática de tipo:

CURVAS IMPLÍCITAS (ecuaciones con igualdad):

- * Cónicas: $x^2 + y^2 == r^2$ (círculo), $x^2/a^2 + y^2/b^2 == 1$ (elipse), $x^2/a^2 - y^2/b^2 == 1$ (hipérbola).
- * Parábolas: $y == x^2$, $x == y^2$.
- * Funciones implícitas: $\text{Sin}[x] == y$, $\text{Log}[x] == y$.
- * Curvas algebraicas generales: $x^3 + y^3 == 3*x*y$.

REGIONES DE DESIGUALDAD (expresiones lógicas):

- * Discos: $x^2 + y^2 <= r^2$.
- * Cuadrantes: $x >= 0 \ \&\& \ y >= 0$.
- * Franjas: $-1 <= x <= 1 \ \&\& \ y >= 0$.
- * Anillos: $r1^2 <= x^2 + y^2 <= r2^2$.
- * Combinaciones lógicas arbitrarias con operadores $\&\&$ y/o $||$.

FUNCIONES EXPLÍCITAS (formato con rango):

- * Estructura: {funcion, {variable, min, max}}.
- * Polinomiales: { x^2 , {x, -3, 3}}.
- * Trigonómicas: {Sin[x], {x, 0, 2*Pi}}.
- * Exponenciales: {Exp[x], {x, -2, 2}}.
- * Logarítmicas: {Log[x], {x, 0.1, 5}}.

CURVAS PARAMÉTRICAS (funciones de parámetro):

- * Estructura: {{x[t], y[t]}, {t, min, max}}.
- * Círculos: {{Cos[t], Sin[t]}, {t, 0, 2*Pi}}.
- * Elipses: {{a*Cos[t], b*Sin[t]}, {t, 0, 2*Pi}}.
- * Espirales: {{t*Cos[t], t*Sin[t]}, {t, 0, 4*Pi}}.
- * Cicloides: {{t - Sin[t], 1 - Cos[t]}, {t, 0, 4*Pi}}.

CONJUNTOS DE PUNTOS:

- * Lista de coordenadas: {{x1, y1}, {x2, y2}, ...}.
- * Por ejemplo: {{0, 0}, {1, 1}, {-1, 2}}.

MÚLTIPLES GRÁFICOS:

- * Lista combinando cualquier formato anterior.
- * Por ejemplo: { $x^2 + y^2 == 4$, $x^2 + y^2 <= 1$, { x^2 , {x, -2, 2}}}

- **archivo** (obligatorio): Cadena especificando el nombre del archivo de destino. Es obligatorio incluir la extensión .tex (por ejemplo: "curvas.tex"). El sistema construye un subdirectorio en Downloads usando el nombre base y deposita el .tex internamente.
- **rejilla** (opcional, por defecto True): Booleano controlando la visualización de cuadrícula coordinada. True renderiza rejilla principal; False suprime la cuadrícula.

- **estilos** (opcional, por defecto {}): Arreglo de especificaciones estilísticas asociadas a cada elemento gráfico. Permite diferenciación visual entre curvas y/o regiones múltiples.

Formato: `{{"tipo_linea", "color"}, {"tipo_linea", "color"}, ...}`.

Tipos de línea: "solid" (continua), "dashed" (discontinua), "dotted" (punteada), "dash-dotted" (raya-punto), "thick" (gruesa), "thin" (delgada).

Colores: "blue", "red", "green", "black", "orange", "purple", "brown", "pink", "gray", "cyan", "magenta".

- **ejes** (opcional, por defecto **True**): Booleano controlando la representación del sistema cartesiano. **True** genera ejes con flechas direccionales; **False** oculta los ejes completamente.
- **texto** (opcional, por defecto {}): Especificación de etiquetas textuales posicionadas en coordenadas del plano.

Formato: `{{"texto", {x, y}, {"opcion", "valor"}, ...}, ...}`.

Opciones de formato:

- * `{"color", "red"}`: Cromaticidad del texto.
- * `{"size", "large"}`: Magnitud tipográfica (\tiny, \small, \normalsize, \large, \Large, \huge, \Huge).
- * `{"anchor", "north"}`: Punto de anclaje ("north", "south", "east", "west", "center" o combinaciones).
- * `{"rotate", "45"}`: Ángulo de rotación en grados.
- **muestras** (opcional, por defecto 50): Entero positivo controlando el número de puntos de evaluación. Rango recomendado: 25-200. Balance crítico: valores altos producen mayor suavidad visual pero incrementan tiempo de cálculo. Valores típicos: 25 (rápido, calidad reducida), 50 (estándar), 100 (alta calidad), 200 (máxima calidad).
- **tolerancia** (opcional, por defecto 0.1): Número positivo especificando precisión para la resolución de curvas implícitas. Aplicable únicamente a ecuaciones con ==. Valores menores incrementan precisión pero ralentizan cálculo. Rangos: 0.01 (alta precisión), 0.1 (estándar), 0.5 (baja precisión, rápido).
- **opacidadRelleno** (opcional, por defecto 0.3): Número real en el intervalo [0,1] controlando transparencia de regiones de desigualdad. Aplicable únicamente a expresiones con operadores relacionales. 0.0 representa transparencia completa, 1.0 opacidad total, 0.3 semitransparencia.
- **densidad** (opcional, por defecto 1.0): Factor multiplicador de densidad de puntos para regiones. Valores: 0.5 (menor densidad, rápido), 1.0 (estándar), 2.0 (mayor densidad, mejor calidad), 3.0 (máxima densidad con advertencia de tiempo). Advertencia: valores > 3.0 pueden causar tiempos excesivos.
- **xmin, xmax, ymin, ymax** (opcionales, defecto **Automatic**): Límites explícitos de los ejes cartesianos. Acepta valores numéricos o símbolo **Automatic** para cálculo automático basado en la extensión de los datos generados.

Retorno en **Mathematica**:

Cadena conteniendo ruta absoluta del archivo *.tex* exportado en subdirectorio creado dentro de *Downloads*.

Algoritmos de renderizado especializados:

- **Curvas implícitas:** Muestreo de contorno mediante evaluación sistemática con tolerancia ajustable. Representación como nube de marcadores con densidad proporcional a la complejidad de la curva. Para curvas con > 2000 puntos, se reduce el tamaño de la marca automáticamente.

- **Regiones de desigualdad:** Evaluación densa del dominio cartesiano con test booleano punto por punto. Renderizado como nube de marcadores semitransparentes con opacidad configurable. Densidad ajustable mediante parámetro multiplicador.
- **Funciones explícitas y curvas paramétricas:** Evaluación uniforme en dominio/parámetro especificado con muestreo según el parámetro `muestras`. Renderizado como trayectoria suave con opción *smooth* de *PGFPlots*.
- **Conjuntos de puntos:** Renderizado directo como marcadores con tamaño fijo.

Configuración interna:

- Precisión numérica: Coordenadas formateadas según magnitud con manejo robusto de valores extremos.
- Asignación automática de colores: Si hay estilos no especificados se aplica el ciclo secuencial: azul, rojo, verde oscuro, naranja, púrpura, café, rosa, gris, cian, magenta.
- Compatibilidad *PGFPlots*: Código generado compatible con la versión 1.18.

Capacidades distintivas:

- Detección automática de tipo de entrada mediante análisis sintáctico sin intervención del usuario.
- Procesamiento heterogéneo: maneja múltiples tipos gráficos en una sola invocación con renderizado diferenciado.
- Cálculo unificado de límites cartesianos mediante análisis global de todos los elementos presentes.
- Manejo robusto de datos degenerados: filtra valores no numéricos, infinitos o indefinidos sin interrumpir la exportación.
- Validación exhaustiva con mensajes descriptivos ante especificaciones inválidas.
- Generación de documento $\LaTeX$ autónomo con clase *standalone*, incluyendo paquetes: *tikz*, *pgfplots*, *amsmath*, *amssymb* y la biblioteca *colorbrewer*.
- Construcción automática de estructura de directorios en *Downloads*.

Recomendaciones de uso:

- Para publicaciones académicas: `muestras >= 100` y `densidad >= 1.5`.
- Para previsualizaciones rápidas: `muestras = 25` y `densidad = 0.5`.
- Curvas implícitas complejas: reducir `tolerancia` incrementa precisión pero ralentiza cálculo significativamente.
- Regiones con superposiciones: usar `opacidadRelleno` reducida (≤ 0.3) para visibilidad de intersecciones.

13.1 Ejemplos de uso de la función ExportContour2DToTikZ

Los ejemplos presentados a continuación demuestran el alcance de la función `ExportContour2DToTikZ` mediante casos que abarcan desde cónicas estándar hasta curvas algebraicas complejas, regiones de desigualdad no triviales y combinaciones heterogéneas de múltiples tipos de gráficos. La selección evalúa tanto capacidades -procesamiento de ecuaciones implícitas, regiones sombreadas, funciones paramétricas- como limitaciones técnicas inherentes al método de muestreo por contorno.

Los primeros tres ejemplos exploran deliberadamente un caso problemático: cónicas con términos cruzados (xy) que representan geometrías rotadas. Estas configuraciones exponen una restricción fundamental

del algoritmo de muestreo: **curvas y regiones con rotación arbitraria (presencia de término mixto xy) no se visualizan perfectamente**, produciendo aproximaciones con irregularidades visibles en los bordes. Esta limitación no constituye un defecto de implementación en el comando `ExportContour2DToTikZ` sino una consecuencia directa del método de evaluación punto por punto con tolerancia finita, dado que las geometrías rotadas requieren densidades de muestreo significativamente mayores para capturar suavidad, incrementando el tiempo de cálculo exponencialmente.

Los ejemplos restantes ilustran escenarios donde el comando opera óptimamente: curvas algebraicas sin rotación, regiones con fronteras complejas (forma de corazón), composiciones multiecuación con tipos heterogéneos (círculos, elipses, hipérbolas, funciones trigonométricas simultáneas) y una combinación mixta de curvas y regiones variadas. Estos casos verifican robustez ante configuraciones diversas, asignación automática de estilos, manejo de anotaciones textuales y capacidad de procesamiento simultáneo de múltiples tipos de gráficos.

Nota: Las ecuaciones que contienen términos cruzados xy podrían requerir un incremento de los parámetros `muestras` (> 1500) y `densidad` (> 2.5), y un decremento en `tolerancia` (< 0.01), aceptando tiempos de cálculo prolongados.

Los argumentos aleatorios empleados (estilos de línea, colores, opciones de texto) evidencian la flexibilidad de `ExportContour2DToTikZ` ante distintas configuraciones. Cada ejemplo ejecuta independientemente, generando un archivo `.tex` que compila mediante la invocación de `CompiladorTex`.

Ejemplo 100 Elipse rotada 30°

```
ContourPlot[
0.333333 x^2 - 0.769800 x y - 1.051570 x + 0.777778 y^2 +
1.547580 y - 0.087322 = 0, {x, -3, 6}, {y, -4, 4}, Frame -> False,
Axes -> True]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
"cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}]
rotatexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
anclajetexto[]}, {"rotate", rotatexto[]}], n]
vectorestilos = estilotext[1];
{Row[{"grid=", grid = True}],
Row[{"estilos=", estilos = Table[{line[], color[]}, 1]}],
Row[{"ejes=", ejes = True}],
Row[{"texto=", texto = {"Elipse rotada", {1, 1}, vectorestilos[[1]]}],
Row[{"muestras=", muestras = 1500}],
Row[{"tolerancia=", tolerancia = 0.01}],
Row[{"opacidad=", opacidad = 0.8}], Row[{"densidad=", densidad = 2}],
Row[{"xmin=", xmin = -3}], Row[{"xmax=", xmax = 6}],
Row[{"ymin=", ymin = -4}], Row[{"ymax=", ymax = 4}]]
ExportContour2DToTikZ[
0.333333 x^2 - 0.769800 x y - 1.051570 x + 0.777778 y^2 +
```

```
1.547580 y - 0.087322 = 0, "ExportContour2DTikZ1.tex", grid,
estilos, ejes, texto, muestras, tolerancia, opacidad, densidad,
xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Wolfram Mathematica** la salida muestra la gráfica de la subfigura 13.1a y los mensajes:

```
{grid= True, estilos= {{thick, magenta}}, ejes= True, texto= {{Ellipse rotada, {1,
1}, {{color, green}, {size, \normalsize}, {anchor, east}, {rotate, 25.6749}}}},
muestras= 1500, tolerancia= 0.01, opacidad= 0.8, densidad= 2, xmin= -3, xmax= 6,
ymin= -4, ymax= 4}
```

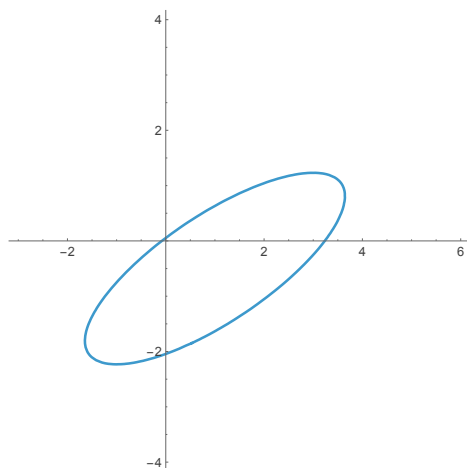
Usando editores ya detectados: 4 disponibles

⋮

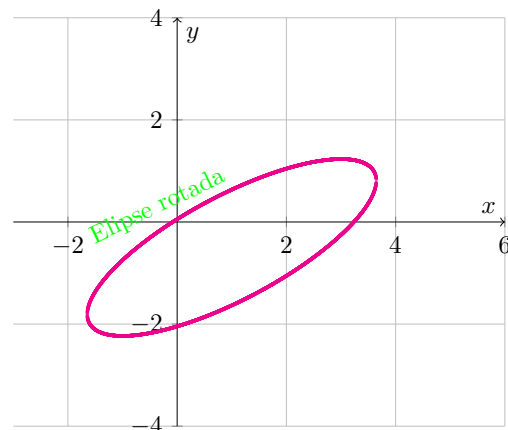
PDF generado: /Users/.../Downloads/ExportContour2DTikZ1/ExportContour2DTikZ-1.pdf

/Users/.../Downloads/ExportContour2DTikZ1/ExportContour2DTikZ1.pdf

El archivo *ExportContour2DTikZ1.tex* (guardado en *Downloads/ExportContour2DTikZ1*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 13.1b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 13.1: Gráficas del ejemplo 100

Ejemplo 101 Región parabólica rotada 30°

```
RegionPlot[-0.45 x^2 - 0.519615 x y + 0.919615 x - 0.15 y^2 +
1.685640 y - 1.485640 >= 0, {x, -8, 6}, {y, -1, 7}, Frame -> False,
Axes -> True]
```

```
line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
"cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\tiny", "\scriptsize", "\footnotesize", "\small",
"\normalsize", "\large", "\Large", "\huge", "\Huge"}]
```

```

anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}]
rotatetexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
  anclajetexto[]}, {"rotate", rotatetexto[]}], n]
vectorestilos = estilotext[1];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estilos=", estilos = Table[{line[], color[]}, 2]}],
  Row[{"ejes=", ejes = True}],
  Row[{"texto=",
    texto = {{"Región parabólica rotada", {0, 0}, vectorestilos[[1]]}},
  Row[{"muestras=", muestras = 70}],
  Row[{"tolerancia=", tolerancia = 0.01}],
  Row[{"opacidad=", opacidad = 0.8}], Row[{"densidad=", densidad = 2}],
  Row[{"xmin=", xmin = Automatic}], Row[{"xmax=", xmax = Automatic}],
  Row[{"ymin=", ymin = Automatic}], Row[{"ymax=", ymax = Automatic}]}
ExportContour2DTikZ[{-0.45 x^2 - 0.519615 x y + 0.919615 x -
  0.15 y^2 + 1.685640 y - 1.485640 >=
  0, -0.45 x^2 - 0.519615 x y + 0.919615 x - 0.15 y^2 +
  1.685640 y - 1.485640 = 0}, "ExportContour2DTikZ2.tex", grid,
estilos, ejes, texto, muestras, tolerancia, opacidad, densidad,
xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En **Mathematica** se obtiene en el *Out* la gráfica de la subfigura 13.2a y los mensajes de proceso:

```

{grid= False, estilos= {{solid, blue}, {thin, yellow}}, ejes= True, texto= {{Región
parabólica rotada, {0, 0}, {{color, green}, {size, \Huge}, {anchor, south east},
{rotate, 273.108}}}}, muestras= 70, tolerancia= 0.01, opacidad= 0.8, densidad= 2,
xmin= Automatic, xmax= Automatic, ymin= Automatic, ymax= Automatic}

```

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/ExportContour2DTikZ2/ExportContour2DTikZ-2.pdf

/Users/.../Downloads/ExportContour2DTikZ2/ExportContour2DTikZ2.pdf

El archivo *ExportContour2DTikZ2.tex* (guardado en *Downloads/ExportContour2DTikZ2*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 13.2b.

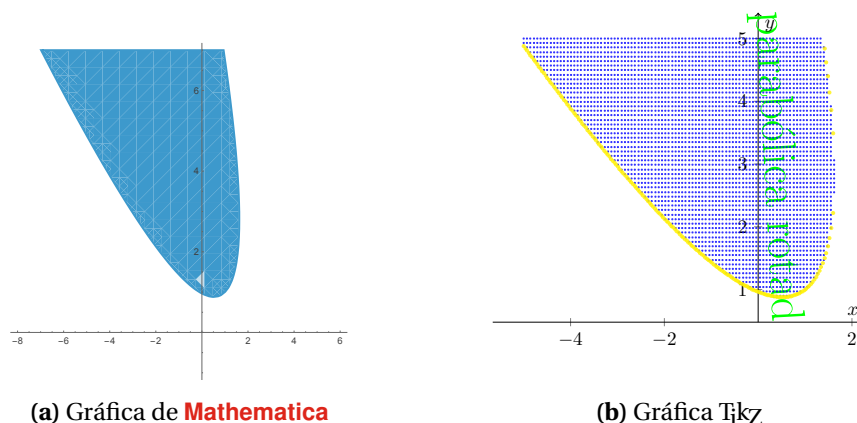


Figura 13.2: Gráficas del ejemplo 101

Ejemplo 102 Región hiperbólica rotada 22.5°

```

RegionPlot[
0.780330 x^2 + 1.060660 x y - 0.560660 x - 0.280330 y^2 +
2.181980 y - 3.462310 <= 0, {x, -7, 7}, {y, -7, 7}, Frame → False,
Axes → True]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
"cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}]
rotatetexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
anclajetexto[]}, {"rotate", rotatetexto[]}], n]
vettorestilos = estilotext[1];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
Row[{"estilos=", estilos = Table[{line[], color[]}, 2]}],
Row[{"ejes=", ejes = True}],
Row[{"texto=",
texto = {"Región hiperbólica", {1, 2}, vettorestilos[[1]]}],
Row[{"muestras=", muestras = 50}],
Row[{"tolerancia=", tolerancia = 0.01}],
Row[{"opacidad=", opacidad = 0.8}], Row[{"densidad=", densidad = 2.5}],
Row[{"xmin=", xmin = -7}], Row[{"xmax=", xmax = 7}],
Row[{"ymin=", ymin = -7}], Row[{"ymax=", ymax = 7}]}
ExportContour2DToTikZ[{0.780330 x^2 + 1.060660 x y - 0.560660 x -
0.280330 y^2 + 2.181980 y - 3.462310 <= 0,
0.780330 x^2 + 1.060660 x y - 0.560660 x - 0.280330 y^2 +
2.181980 y - 3.462310 = 0}, "ExportContour2DToTikZ3.tex", grid,
estilos, ejes, texto, muestras, tolerancia, opacidad, densidad,
xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram** la salida retorna la gráfica de la subfigura 13.3a y los mensajes de texto:

```

{grid= True, estilos= {{dotted, blue}, {dashed, yellow}}, ejes= True, texto=
{{Región hiperbólica, {1, 2}, {{color, red}, {size, \normalsize}, {anchor, east},
{rotate, 81.0636}}}}, muestras= 50, tolerancia= 0.01, opacidad= 0.8, densidad= 2.5,
xmin= -7, xmax= 7, ymin= -7, ymax= 7}

```

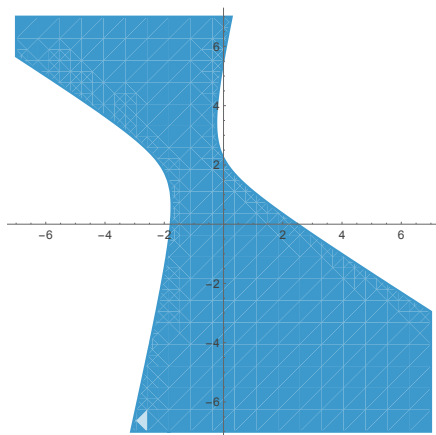
Usando editores ya detectados: 4 disponibles

:

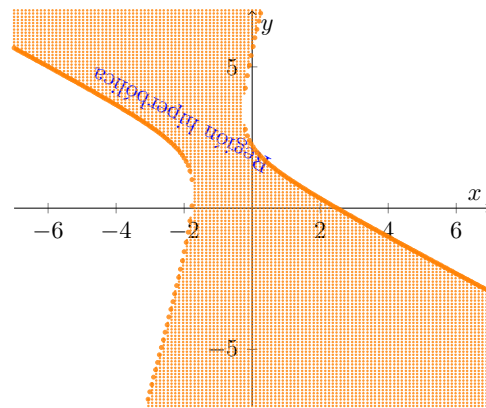
PDF generado: /Users/.../Downloads/ExportContour2DToTikZ3/ExportContour2DToTikZ-3.pdf

/Users/.../Downloads/ExportContour2DToTikZ3/ExportContour2DToTikZ3.pdf

El archivo *ExportContour2DToTikZ3.tex* (guardado en *Downloads/ExportContour2DToTikZ3*) integra el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 13.3b.



(a) Gráfica de Mathematica



(b) Gráfica TikZ

Figura 13.3: Gráficas del ejemplo 102

Ejemplo 103 Región de corazón

```

RegionPlot[(x^2 + y^2 - 1)^3 - x^2 y^3 <= 0, {x, -1.5, 1.5}, {y, -1.2, 1.5}]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
  "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}]
rotatexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
  anclajetexto[]}, {"rotate", rotatexto[]}], n]
vettorestilos = estilotext[1];
{Row[{"grid=", grid = RandomChoice[{True, False}]}],
  Row[{"estilos=", estilos = Table[{line[], color[]}, 1]}],
  Row[{"ejes=", ejes = RandomChoice[{True, False}]}],
  Row[{"texto=", texto = {"Corazón", {0, 0}, vettorestilos[[1]]}],
  Row[{"muestras=", muestras = 80}],
  Row[{"tolerancia=", tolerancia = 0.01}],
  Row[{"opacidad=", opacidad = 0.8}], Row[{"densidad=", densidad = 2.5}],
  Row[{"xmin=", xmin = Automatic}], Row[{"xmax=", xmax = Automatic}],
  Row[{"ymin=", ymin = Automatic}], Row[{"ymax=", ymax = Automatic}]}
ExportContour2DTiTikZ[(x^2 + y^2 - 1)^3 - x^2 y^3 <= 0,
  "ExportContour2DTiTikZ4.tex", grid, estilos, ejes, texto, muestras,
  tolerancia, opacidad, densidad, xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

En Mathematica se muestra como salida la gráfica de la subfigura 13.4a y los mensajes de proceso:

```

{grid= False, estilos= {{solid, cyan}}, ejes= False, texto= {{Corazón, {0, 0},
{{color, cyan}, {size, \small}, {anchor, east}, {rotate, 350.752}}}}, muestras= 80,
tolerancia= 0.01, opacidad= 0.8, densidad= 2.5, xmin= Automatic, xmax= Automatic,

```



```
ymin= Automatic, ymax= Automatic}
```

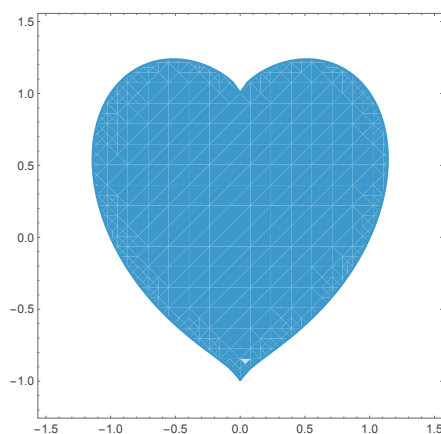
Usando editores ya detectados: 4 disponibles

⋮

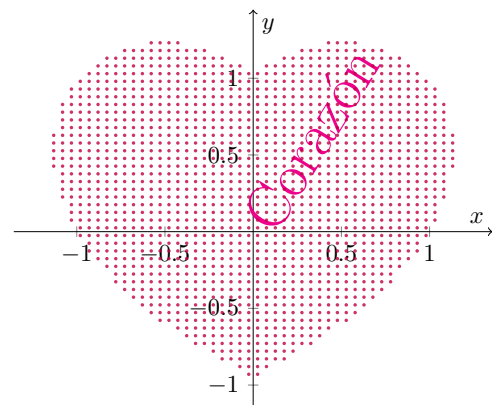
PDF generado: /Users/.../Downloads/ExportContour2DTikZ4/ExportContour2DTikZ-4.pdf

/Users/.../Downloads/ExportContour2DTikZ4/ExportContour2DTikZ4.pdf

El archivo *ExportContour2DTikZ4.tex* (guardado en *Downloads/ExportContour2DTikZ4*) contiene el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\TikZ}$ de la subfigura 13.4b.

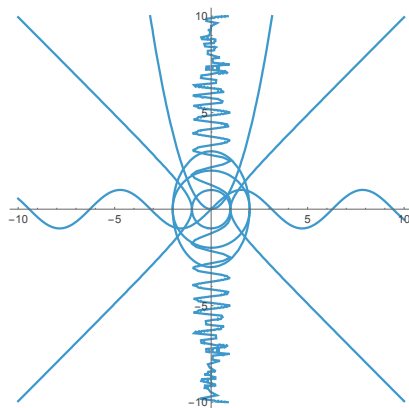


(a) Gráfica de **Mathematica**

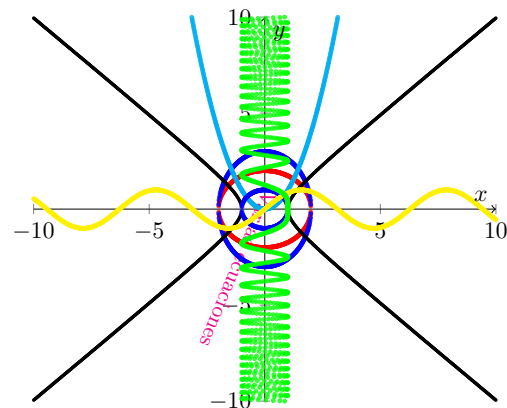


(b) Gráfica $\text{\TikZ}$

Figura 13.4: Gráficas del ejemplo 103



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\TikZ}$

Figura 13.5: Gráficas del ejemplo 104

Ejemplo 104 Círculos, elipse, hipérbola, parábola, seno y coseno

$$\text{eq1} = x^2 + y^2 = 4;$$

$$\text{eq2} = x^2/4 + y^2/9 = 1;$$

$$\text{eq3} = x^2 - y^2 = 1;$$


```

eq4 = y = x^2;
eq5 = x^2 + y^2 = 1;
eq6 = y = Sin[x];
eq7 = x = Cos[y^2];
Show[ContourPlot[eq1, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq2, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq3, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq4, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq5, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq6, {x, -10, 10}, {y, -10, 10}],
ContourPlot[eq7, {x, -10, 10}, {y, -10, 10}], Frame → False,
Axes → True]

line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
"cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
"\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
"north west", "south east", "south west", "center"}]
rotatexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
anclajetexto[]}, {"rotate", rotatexto[]}], n]
vectorestilos = estilotext[1];
{Row[{"grid=", grid = False}],
Row[{"estilos=", estilos = Table[{line[], color[]}, 7]}],
Row[{"ejes=", ejes = True}],
Row[{"texto=",
texto = {{"Varias ecuaciones", {1, 1}, vectorestilos[[1]]}},
Row[{"muestras=", muestras = 200}],
Row[{"tolerancia=", tolerancia = 0.01}],
Row[{"opacidad=", opacidad = 0.8}], Row[{"densidad=", densidad = 3}],
Row[{"xmin=", xmin = -10}], Row[{"xmax=", xmax = 10}],
Row[{"ymin=", ymin = -10}], Row[{"ymax=", ymax = 10}]]
ExportContour2DTiTikZ[{eq1, eq2, eq3, eq4, eq5, eq6, eq7},
"ExportContour2DTiTikZ5.tex", grid, estilos, ejes, texto, muestras,
tolerancia, opacidad, densidad, xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" → "TeXstudio"]

```

En **Wolfram Mathematica** el *Out* obtenido construye la gráfica de la subfigura 13.5a y los mensajes de texto:

```

{grid= False, estilos= {{dotted, purple}, {dashed, green}, {dashed, yellow},
{dotted, red}, {dashed, red}, {thick, blue}, {dashed, yellow}}, ejes= True, texto=
{{Varias ecuaciones, {1, 1}, {{color, cyan}, {size, \normalsize}, {anchor, north
east}, {rotate, 62.4614}}}}, muestras= 200, tolerancia= 0.01, opacidad= 0.8,
densidad= 3, xmin= -10, xmax= 10, ymin= -10, ymax= 10}
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportContour2DTiTikZ5/ExportContour2DTiTikZ-
5.pdf
/Users/.../Downloads/ExportContour2DTiTikZ5/ExportContour2DTiTikZ5.pdf

```

El archivo *ExportContour2DTiTikZ5.tex* (guardado en *Downloads/ExportContour2DTiTikZ5*) posee

el código $\text{\LaTeX}$ editable que genera la gráfica $\text{\LaTeX}$ de la subfigura 13.5b.

Ejemplo 105 Ecuaciones mixtas con anotaciones de texto

```
Show[ContourPlot[x^2 + y^2 = 4, {x, -4, 4}, {y, -3, 3}],
RegionPlot[y >= x^2 - 3, {x, -4, 4}, {y, -3, 3}],
Plot[Sin[x], {x, -4, 4}],
ParametricPlot[{2*Cos[t], Sin[t]}, {t, 0, 2*Pi}], Frame ->False,
Axes ->True]

mixtas = {x^2 + y^2 = 4,
  y >= x^2 - 3, {Sin[x], {x, -4, 4}}, {{2*Cos[t], Sin[t]}, {t, 0,
    2*Pi}}};
line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\tiny", "\scriptsize", "\footnotesize", "\small",
  "\normalsize", "\large", "\Large", "\huge", "\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}]
rotatexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[]}, {"size", sizetexto[]}, {"anchor",
  anclajetexto[]}, {"rotate", rotatexto[]}], n]
vectorsilos = estilotext[4];
{Row[{"grid=", grid = False}],
  Row[{"estilos=", estilos = Table[{line[], color[]}, 4]}],
  Row[{"ejes=", ejes = True}],
  Row[{"texto=",
    texto = {"Círculo", {2, 2},
      vectoresilos[[1]]}, {"Parábola", {0, -2},
      vectoresilos[[2]]}, {"Seno", {0, 1},
      vectoresilos[[3]]}, {"Elipse", {1.5, 0}, vectoresilos[[4]]}}],
  Row[{"muestras=", muestras = 100}],
  Row[{"tolerancia=", tolerancia = 0.1}],
  Row[{"opacidad=", opacidad = 0.3}], Row[{"densidad=", densidad = 1}],
  Row[{"xmin=", xmin = -4}], Row[{"xmax=", xmax = 4}],
  Row[{"ymin=", ymin = -3}], Row[{"ymax=", ymax = 3}]]
ExportContour2DTiTikZ[mixtas, "ExportContour2DTiTikZ6.tex", grid,
estilos, ejes, texto, muestras, tolerancia, opacidad, densidad,
xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Mathematica** la salida muestra la gráfica de la subfigura 13.6a y los mensajes:

```
{grid= False, estilos= {{thick, red}, {dashed, magenta}, {thin, black}, {dotted,
purple}}, ejes= True, texto= {{Círculo, {2, 2}, {{color, magenta}, {size, \tiny},
{anchor, center}, {rotate, 19.2065}}}, {Parábola, {0, -2}, {{color, green}, {size,
\footnotesize}, {anchor, south}, {rotate, 151.787}}}, {Seno, {0, 1}, {{color,
green}, {size, \scriptsize}, {anchor, north east}, {rotate, 130.677}}}, {Elipse,
{1.5, 0}, {{color, green}, {size, \footnotesize}, {anchor, north east}, {rotate,
192.909}}}}
```

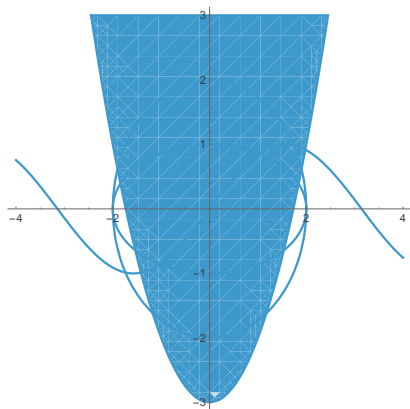
Usando editores ya detectados: 4 disponibles

:

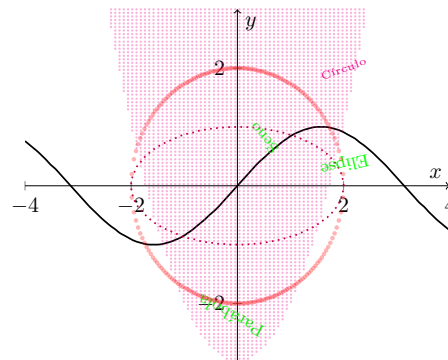
PDF generado: /Users/.../Downloads/ExportContour2DToTikZ6/ExportContour2DToTikZ-6.pdf

/Users/.../Downloads/ExportContour2DToTikZ6/ExportContour2DToTikZ6.pdf

El archivo *ExportContour2DToTikZ6.tex* (guardado en *Downloads/ExportContour2DToTikZ6*) facilita el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 13.6b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 13.6: Gráficas del ejemplo 105

Ejemplo 106 Con varias características y uso de &&

```
Show[ContourPlot[x^2 + y^2 == 9, {x, -4, 6}, {y, -4, 4}],
RegionPlot[x^2 + y^2 <= 4 && y >= x^2 - 2, {x, -4, 6}, {y, -4, 4}],
RegionPlot[y >= x^2 - 2, {x, -4, 6}, {y, -4, 4}],
ParametricPlot[{3*Cos[t], 2*Sin[t]}, {t, 0, 2*Pi}],
Plot[Sin[2*x], {x, -2 Pi, 2 Pi}], Frame ->False, Axes ->True]

problema = {x^2 + y^2 == 9, x^2 + y^2 <= 4 && y >= x^2 - 2,
  y >= x^2 - 2, {{3*Cos[t], 2*Sin[t]}, {t, 0, 2*Pi}}, {Sin[
    2*x], {x, -2 Pi, 2 Pi}}};
line[] := RandomChoice[{"solid", "dashed", "dotted", "thick", "thin"}]
color[] :=
RandomChoice[{"blue", "red", "green", "purple", "yellow", "orange",
  "cyan", "magenta", "black"}]
sizetexto[] :=
RandomChoice[{"\\tiny", "\\scriptsize", "\\footnotesize", "\\small",
  "\\normalsize", "\\large", "\\Large", "\\huge", "\\Huge"}]
anclajetexto[] :=
RandomChoice[{"north", "south", "east", "west", "north east",
  "north west", "south east", "south west", "center"}]
rotatexto[] := RandomReal[{0, 360}]
estilotext[n_] :=
Table[{"color", color[], {"size", sizetexto[], {"anchor",
  anclajetexto[], {"rotate", rotatexto[]}}, n]
vectorsilos = estilotext[5];
{Row[{"grid=", grid = True}],
  Row[{"estilos=", estilos = Table[{line[], color[]}, 5]}],
  Row[{"ejes=", ejes = True}],
  Row[{"texto=",
    texto = {"x^2+y^2=9", {2.5, 2},
      vectorsilos[[1]]}, {"Región interior", {0, 0},
```

```

vettorestilos[[2]]}, {"Parábola", {-1, -1.5},
vettorestilos[[3]]}, {"Elipse", {2, 1},
vettorestilos[[4]]}, {"y = sin(2x)", {2, 0.5},
vettorestilos[[5]]}}], Row[{"muestras="}, muestras = 60]],
Row[{"tolerancia="}, tolerancia = 0.01]],
Row[{"opacidad="}, opacidad = 0.4]], Row[{"densidad="}, densidad = 1.5]],
Row[{"xmin="}, xmin = -4]], Row[{"xmax="}, xmax = 6]],
Row[{"ymin="}, ymin = -4]], Row[{"ymax="}, ymax = 4]]]
ExportContour2DToTikZ[problema, "ExportContour2DToTikZ7.tex", grid,
estilos, ejes, texto, muestras, tolerancia, opacidad, densidad,
xmin, xmax, ymin, ymax];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]

```

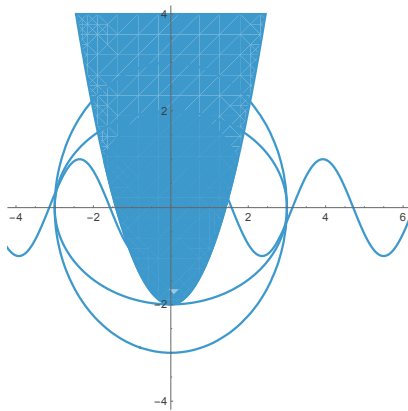
En **Wolfram** la salida comparte la gráfica de la subfigura 13.7a y los mensajes de proceso:

```

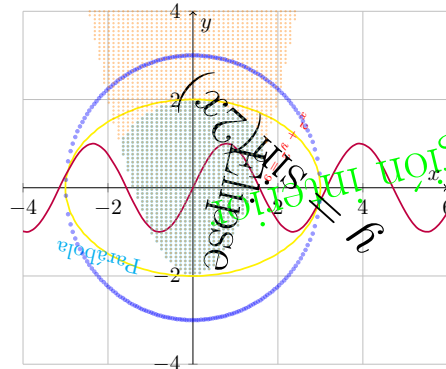
{grid= True, estilos= {{solid, blue}, ..., xmin= -4, xmax= 6, ymin= -4, ymax= 4}
Usando editores ya detectados: 4 disponibles
:
PDF generado: /Users/.../Downloads/ExportContour2DToTikZ7/ExportContour2DToTikZ-
7.pdf
/Users/.../Downloads/ExportContour2DToTikZ7/ExportContour2DToTikZ7.pdf

```

El archivo *ExportContour2DToTikZ7.tex* (guardado en *Downloads/ExportContour2DToTikZ7*) posee el código $\text{\LaTeX}$ editable que produce la gráfica $\text{\LaTeX}$ de la subfigura 13.7b.



(a) Gráfica de **Mathematica**



(b) Gráfica $\text{\LaTeX}$

Figura 13.7: Gráficas del ejemplo 106

Para más ejemplos

Otros ejemplos vinculados con el uso de la sentencia **ExportContour2DToTikZ** se encuentran disponibles en el archivo: 13. *Ejemplos ExportContour2DToTikZ.nb*, compartido en el enlace de descarga del paquete **V $\text{\LaTeX}$** (ver página xiv).

¿Qué aprenderás en este capítulo?

- ▶ Crear animaciones $\text{\LaTeX}$ interactivas con el paquete *animate*.
- ▶ Convertir funciones paramétricas de **Mathematica** en secuencias animadas.
- ▶ Generar controles de reproducción nativos en documentos PDF.
- ▶ Configurar parámetros continuos y discretos para visualizaciones dinámicas.
- ▶ Documentar fenómenos evolutivos y demostraciones matemáticas animadas.

La generación de animaciones matemáticas en documentos $\text{\LaTeX}$ muchas veces requiere exportación secuencial de imágenes estáticas seguida de ensamblaje manual mediante paquetes especializados, lo que constituye un proceso laborioso susceptible a inconsistencias en escalado y alineación entre *frames*. Si bien **Mathematica** ofrece las intrucciones `Animate` y `Manipulate` para visualización interactiva, la conversión directa a animaciones $\text{\LaTeX}$ embebidas no existe nativamente.

El generador de animaciones `CreateTikZAnimation` del paquete **V $\text{\LaTeX}$** automatiza esta traducción mediante parametrización funcional: acepta funciones puras de **Wolfram Mathematica** que producen gráficos dependientes de parámetros, genera secuencias de valores según especificaciones de rango (continuo o discreto), evalúa la función para cada configuración paramétrica, extrae primitivas gráficas, traduce coordenadas a sintaxis *PGFPlots*, ensambla *frames* individuales como entornos *tikzpicture* secuenciales, y construye un documento $\text{\LaTeX}$ con paquete *animate* configurado para la reproducción controlada por el usuario.

El mecanismo de conversión opera mediante evaluación sistemática: identifica tipo de especificación paramétrica (rangos continuos con interpolación lineal o conjuntos discretos con indexación), genera una secuencia de N configuraciones paramétricas uniformemente distribuidas, invoca a la función pura con cada configuración, inspecciona estructura del objeto resultante mediante búsqueda de patrones, calcula los límites cartesianos globales (automáticos o especificados), formatea coordenadas numéricas con precisión adaptativa, y serializa bloques *axis* con delimitadores `\newframe` para sincronización temporal.

La función `CreateTikZAnimation` convierte funciones puras de **Mathematica** que generan gráficos dependientes de parámetros en animaciones $\text{\LaTeX}$ embebidas mediante el paquete *animate*, produciendo documentos *standalone* con controles de reproducción integrados (*play/pause*, nave-

gación *frame-por-frame* y *loop*). A diferencia de exportaciones que incrustan videos o secuencias de imágenes *bitmap*, **CreateTikZAnimation** genera código $\text{\TeX}$ vectorial para cada *frame*, preservando la editabilidad completa.

Esta capacidad resulta fundamental para presentaciones académicas, material didáctico interactivo y demostraciones de conceptos dinámicos donde se requiere control preciso sobre transiciones y visualización de evolución paramétrica. La ejecución de **CreateTikZAnimation** retorna un archivo *.tex* que compila mediante `pdflatex -shell-escape`, generando un PDF con animación embebida reproducible en *Adobe Acrobat/Reader*. Analicemos la especificación argumental de **CreateTikZAnimation**.

- **CreateTikZAnimation**[*plotFunc*, *paramSpecs*, *fileName*, *numFrames*, *frameRate*, *gridOption*, *stylesOption*, *axesOption*, *xmin*, *xmax*, *ymin*, *ymax*, *points*]: Traduce una función pura de **Wolfram** que genera gráficos paramétricos en una animación $\text{\TeX}$ mediante el paquete *animate*, construyendo como resultado un documento *standalone* compilable.

Parámetros:

- **plotFunc** (obligatorio): Función pura que retorna un objeto **Graphics** válido. Admite dos sintaxis:

Sintaxis explícita con **Function**:

- * Un parámetro: `Function[{a}, Plot[Sin[a*x], {x, 0, 2*Pi}]]`.
- * Múltiples parámetros: `Function[{a, b}, Plot[a*Sin[b*x], {x, 0, 2*Pi}]]`.

Sintaxis abreviada con **&**:

- * Un parámetro: `Plot[Sin[##*x], {x, 0, 2*Pi}]&`.

Funciones gráficas compatibles:

- * **Plot**: Gráficos de funciones explícitas.
- * **ParametricPlot**: Curvas paramétricas.
- * **PolarPlot**: Gráficos en coordenadas polares.
- * **ListLinePlot**: Gráficos de listas con líneas conectadas.
- * **ListPlot**: Gráficos de puntos discretos.
- * **LogPlot**: Gráficos logarítmicos.
- * Cualquier construcción que genere primitivas **Line** o **Point**.

- **paramSpecs** (obligatorio): Lista de especificaciones de parámetros definiendo espacio de variación. Admite dos formatos:

Parámetros continuos (interpolación lineal):

- * Formato: `{{variable, min, max}, ...}`.
- * Ejemplo: `{{a, 1, 5}}` genera secuencia de valores reales uniformemente distribuidos entre 1 y 5.
- * Múltiples parámetros: `{{a, 0.5, 2}, {b, 1, 4}}`.

Parámetros discretos (indexación):

- * Formato: `{{variable, {valor1, valor2, ...}}, ...}`.
- * Ejemplo: `{{color, {Red, Blue, Green}}}` se encicla a través de un conjunto finito.
- * Indexación uniforme: distribución proporcional según el número de *frames*.

- **fileName** (obligatorio): Cadena especificando el nombre del archivo destino. Es obligatorio incluir la extensión *.tex*. El sistema construye un subdirectorio en *Downloads* usando el nombre base y deposita el *.tex* internamente.

- **numFrames** (opcional, por defecto 20): Entero en el intervalo [2, 200] especificando un número total de *frames* de la animación a construir. Balance crítico: valores altos producen transiciones suaves pero incrementan el tamaño del archivo y el tiempo de compilación. Valores típicos: 20 (estándar), 30 (suave), 50 (muy suave), 100+ (alta calidad).
- **frameRate** (opcional, por defecto 5): Número real en el intervalo (0, 60] especificando los *frames* por segundo durante la reproducción. Valores comunes: 2 (muy lento), 5 (estándar didáctico), 10 (fluido), 24 (cinematográfico), 30 (video estándar).
- **gridOption** (opcional, por defecto **True**): Booleano que controla la visualización de la cuadrícula coordinada en todos los *frames*. **True** renderiza la rejilla principal; **False** suprime la cuadrícula.
- **stylesOption** (opcional, por defecto {}): Cadena especificando el color de líneas/puntos. Colores válidos: "red", "blue", "green", "orange", "purple", "black". Lista vacía {} adopta negro por defecto.
- **axesOption** (opcional, por defecto **True**): Booleano controlando la representación del sistema cartesiano. **True** genera ejes centrados; **False** oculta los ejes completamente.
- **xmin, xmax, ymin, ymax** (opcionales, por defecto **Automatic**): Límites explícitos de los ejes cartesianos aplicados uniformemente a todos los *frames*. Acepta valores numéricos o símbolo **Automatic** para cálculo automático por *frame* basado en la extensión de los datos. **Recomendación:** Especificar límites explícitos previene variación de escala entre *frames* que podría tener un efecto visual indeseable.
- **points** (opcional, por defecto **False**): Booleano que controla el modo de renderizado. **False** (por defecto) traza primitivas **Line** como trayectorias continuas; **True** renderiza todos los datos como marcadores discretos circulares. Esto es útil para visualizaciones de datos puntuales o simulaciones de partículas.

Retorno en **Mathematica**:

Cadena conteniendo la ruta absoluta del archivo *.tex* exportado en el subdirectorío creado dentro de *Downloads*, o **\$Failed** ante errores de validación.

Algoritmo de generación:

- **Fase 1 - Validación:** Verifica la sintaxis de la función pura, la consistencia de especificaciones paramétricas, la validez del nombre del archivo y los rangos numéricos de opciones.
- **Fase 2 - Secuenciación paramétrica:** Para cada parámetro, genera un arreglo de N valores ($N = \text{numFrames}$). Parámetros continuos: interpolación lineal $\text{min} + (\text{max} - \text{min}) * (i-1) / (N-1)$. Parámetros discretos: indexación proporcional $\text{valores}[\text{Floor}[(\text{Length}[\text{valores}]-1) * (i-1) / (N-1) + 1]]$.
- **Fase 3 - Evaluación gráfica:** Invoca **plotFunc** con cada configuración paramétrica. Si la función es **Function**, aplica argumentos directamente (**plotFunc@@valores**). En caso contrario, usa sustitución de reglas (**plotFunc/.Thread[vars -> valores]**).
- **Fase 4 - Extracción de primitivas:** Inspecciona cada **Graphics** mediante **Cases[grafico, Line[pts_] :> pts, Infinity]** y análogamente para **Point**. Filtra coordenadas no numéricas.
- **Fase 5 - Cálculo de límites:** Si los límites son **Automatic**, calcula el envolvente de todos los puntos por *frame*. Si son especificados manualmente, usa valores fijos para todos los *frames*.
- **Fase 6 - Formateo numérico adaptativo:** Coordenadas $< 10^{-10}$ se redondean a cero. Valores $> 10^8$ usan notación científica. Valores intermedios se formatean con precisión apropiada (0.01 para > 1000 y 10^{-6} en caso contrario).
- **Fase 7 - Síntesis TikZ:** Construye bloque *axis* por *frame* con límites, opciones de rejilla/ejes, y coordenadas formateadas. Primitivas **Line** se traducen a **\addplot[color, thick]**. Primitivas **Point** usan **only marks**.

- **Fase 8 - Ensamblaje de animación:** Concatena *frames* con delimitadores `\newframe`. Se encapsula todo en el entorno `\begin{animateinline}[controls,loop]{frameRate}`.

Configuración interna:

- Paquetes $\LaTeX$ incluidos: *tikz*, *pgfplots*, *animate*, *amsmath* y *amssymb*.
- Compatibilidad *PGFPlots*: Código compatible con la versión 1.18.
- Controles de reproducción: Paquete *animate* genera los botones *play/pause*, navegación *frame-por-frame* y *loop* automático.
- Formato de números: Configuración `scaled ticks=false y tick label style={/pgf/number format/fixed}` para notación decimal consistente.

Compilación del resultado:

- Visualización: PDF que contiene la animación embebida reproducible en *Adobe Acrobat/Reader*, *Foxit Reader* o navegadores web modernos con soporte de *JavaScript*.
- Advertencia: Algunos visores de PDF no soportan animaciones embebidas, en dicho caso, solo mostrarán el primer *frame* estático.

Capacidades distintivas:

- Parametrización heterogénea: Maneja simultáneamente parámetros continuos (interpolados) y discretos (indexados) en una sola animación.
- Compatibilidad amplia: Funciona con cualquier comando de **Mathematica** que genere primitivas `Line` o `Point` dentro de `Graphics`.
- Límites adaptativos o fijos: Opción de calcular límites por *frame* (dinámico) o fijar globalmente (estático) según la necesidad.
- Modo dual de renderizado: Trayectorias continuas (líneas) o nubes de puntos discretos según el parámetro `points`.
- Validación exhaustiva: Verifica rangos, tipos y formatos de todos los parámetros con retorno `$Failed` ante inconsistencias.
- Generación de documento autónomo con clase *standalone* listo para compilación directa.

Limitaciones conocidas:

- **Primitivas no soportadas:** Solo procesa `Line` y `Point`. Primitivas como `Polygon`, `Circle`, `Rectangle`, etc., se ignoran silenciosamente.
- **Tamaño del archivo:** Animaciones con > 100 *frames* y densidad de datos alta producen archivos *.tex* grandes (> 10 MB) con compilación lenta.
- **Compatibilidad de visores:** Requiere *Adobe Acrobat/Reader* o *Foxit Reader* para la reproducción. Visores de PDF nativos de sistemas operativos frecuentemente no soportan animaciones embebidas.

Recomendaciones de uso:

- Para animaciones suaves: `numFrames` ≥ 30 y `frameRate` = 10.
- Para presentaciones didácticas: `numFrames` = 20 y `frameRate` = 5 (permite observación clara de transiciones).
- Para evitar variación de escala: **siempre especificar** `xmin`, `xmax`, `ymin` y `ymax` manualmente.
- Para optimizar el tamaño del archivo: Limitar `numFrames` ≤ 50 y reducir la densidad de datos en la función `plotFunc`.

14.1 Ejemplos de uso de la función CreateTikZAnimation

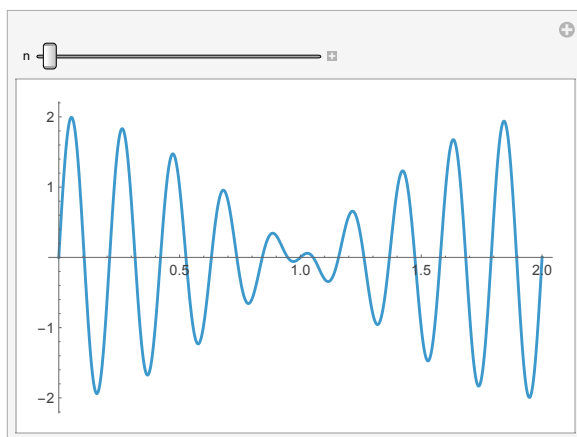
Los ejemplos siguientes demuestran las capacidades del comando `CreateTikZAnimation` en la conversión de visualizaciones interactivas de **Mathematica** (`Manipulate`) hacia animaciones $\text{\LaTeX}$ embebidas mediante parametrización funcional. Cada caso traduce un concepto dinámico -evolución temporal, variación morfológica, fenómenos físicos- en una secuencia de *frames* vectoriales con reproducción controlada.

La progresión de los ejemplos examina una complejidad creciente: desde funciones trigonométricas simples con un parámetro hasta composiciones heterogéneas que combinan múltiples tipos de curvas (`Plot`, `ParametricPlot`, `PolarPlot`). Todos los ejercicios especifican límites cartesianos explícitos para prevenir variación de escala entre los *frames*.

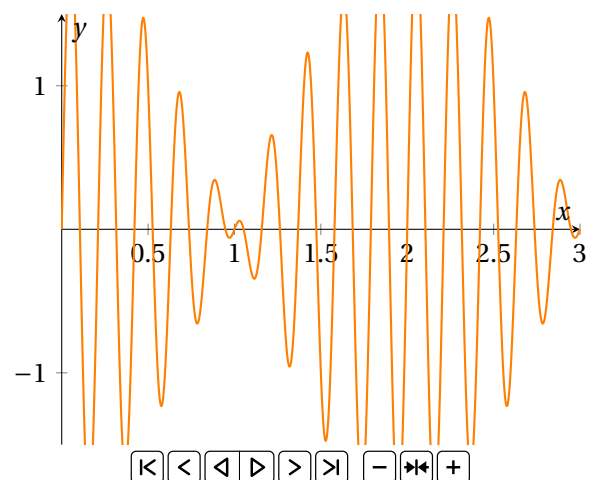
Categorías ilustradas:

- **Ejemplo 107:** Batimiento de ondas - interferencia entre frecuencias próximas con parámetro continuo.
- **Ejemplo 108:** Espiral de *Arquímedes* con expansión/contracción - curva paramétrica con factor de escala variable.
- **Ejemplo 109:** Rosa polar - curvas en coordenadas polares con número de pétalos variable.
- **Ejemplo 110:** Composición multicurva - cicloide, espiral polar, coseno y línea vertical evolucionando simultáneamente mediante el uso de la instrucción `Show` nativa de **Wolfram Mathematica**.

Los parámetros aleatorios (rejilla, color, modo de puntos) verifican robustez ante configuraciones variadas. Cada ejemplo invoca a `CompiladorTex` para la compilación automática con la opción `-shell-escape` requerida por el paquete *animate*. Las animaciones resultantes se pueden reproducir en *Adobe Acrobat/Reader* con controles integrados (*play/pause*, navegación y *loop*).



(a) Animación de **Mathematica**



(b) Animación $\text{\LaTeX}$

Figura 14.1: Animaciones del ejemplo 107

Ejemplo 107 Batimiento de ondas

```
Manipulate[
  Plot[Sin[2 Pi*5*x] + Sin[2 Pi*n*x], {x, 0, 2}], {n, 4.5, 5.5}
  {frames = 40, framesxSeg = 6, grid = RandomChoice[{True, False}],
    style = RandomChoice[{"red", "blue", "green", "orange", "purple",
      "black"}], ejes = True, xmin = 0, xmax = 3, ymin = -1.5,
    ymax = 1.5, points = RandomChoice[{True, False}]}}
CreateTikZAnimation[
```

```
Function[{n},
Plot[Sin[2 Pi*5*x] + Sin[2 Pi*n*x], {x, 0, 3}]], {{n, 4.5,
5.5}}, "CreateTikZAnimation1.tex", frames, framesxSeg, grid,
style, ejes, xmin, xmax, ymin, ymax, points];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Wolfram** la salida retorna la animación de la subfigura 14.1a y los mensajes de proceso:

```
{40, 6, False, orange, True, 0, 3, -1.5, 1.5, False}
```

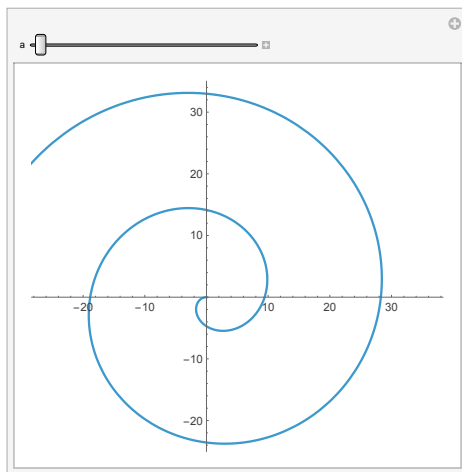
Usando editores ya detectados: 4 disponibles

⋮

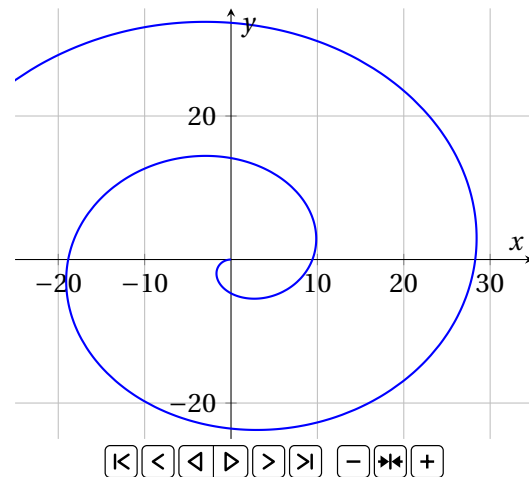
PDF generado: /Users/.../Downloads/CreateTikZAnimation1/CreateTikZAnimation1.pdf

/Users/.../Downloads/CreateTikZAnimation1/CreateTikZAnimation1.pdf

El archivo *CreateTikZAnimation1.tex* (guardado en *Downloads/CreateTikZAnimation1*) posee el código $\text{\LaTeX}$ editable que genera la animación $\text{\LaTeX}$ de la subfigura 14.1b (**manipulable** desde este PDF).



(a) Animación de **Mathematica**



(b) Animación $\text{\LaTeX}$

Figura 14.2: Animaciones del ejemplo 108

Ejemplo 108 Espiral que se expande

```
Manipulate[
ParametricPlot[{a*t*Cos[t], a*t*Sin[t]}, {t, 0, 4 Pi},
PlotRange ->{-25, 35}], {a, -3, 3}
{frames = 30, framesxSeg = 8, grid = RandomChoice[{True, False}],
style = RandomChoice[{"red", "blue", "green", "orange", "purple",
"black"}], ejes = True, xmin = -25, xmax = 35, ymin = -25,
ymax = 35, points = RandomChoice[{True, False}]}
CreateTikZAnimation[
Function[{a},
ParametricPlot[{a*t*Cos[t], a*t*Sin[t]}, {t, 0,
4 Pi}]], {{a, -3, 3}}, "CreateTikZAnimation2.tex", frames,
framesxSeg, grid, style, ejes, xmin, xmax, ymin, ymax, points];
CompiladorTex[%, "PreferredEditor" -> "TeXstudio"]
```

En **Mathematica** se muestra como salida la animación de la subfigura 14.2a y los mensajes de texto:

14.1 Ejemplos de uso de la función CreateTikZAnimation

```
{30, 8, True, blue, True, -25, 35, -25, 35, False}
```

Usando editores ya detectados: 4 disponibles

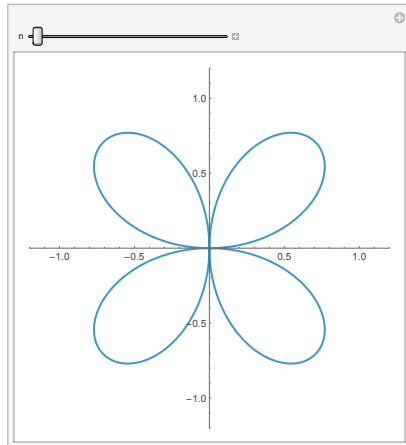
⋮

PDF generado: /Users/.../Downloads/CreateTikZAnimation2/CreateTikZAnimation2.pdf

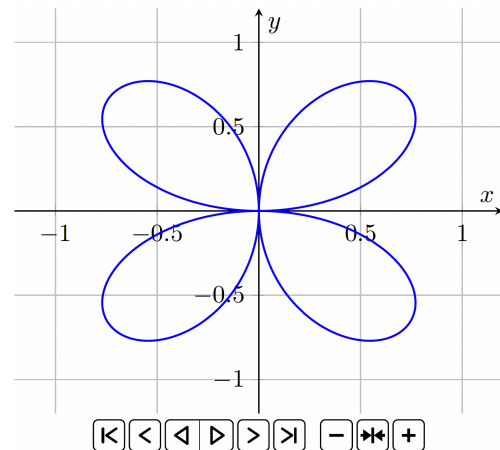
/Users/.../Downloads/CreateTikZAnimation2/CreateTikZAnimation2.pdf

El archivo *CreateTikZAnimation2.tex* (guardado en *Downloads/CreateTikZAnimation2*) contiene el código $\text{\LaTeX}$ editable que produce la animación $\text{\LaTeX}$ de la subfigura 14.2b (**manipulable** desde este PDF).

175



(a) Animación de **Mathematica**



(b) Animación $\text{\LaTeX}$

Figura 14.3: Animaciones del ejemplo 109

Ejemplo 109 Rosa polar con pétalos variables

```
Manipulate[
  PolarPlot[Sin[n*θ], {θ, 0, 2 Pi},
  PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}}, {n, 2, 8}]
  {frames = 30, framesxSeg = 6, grid = RandomChoice[{True, False}],
  style = RandomChoice[{"red", "blue", "green", "orange", "purple",
  "black"}], ejes = True, xmin = -1.2, xmax = 1.2, ymin = -1.2,
  ymax = 1.2, points = RandomChoice[{True, False}]}
  CreateTikZAnimation[
  Function[{n},
  PolarPlot[Sin[n*θ], {θ, 0, 2 Pi}]], {{n, 2, 8}},
  "CreateTikZAnimation3.tex", frames, framesxSeg, grid, style, ejes,
  xmin, xmax, ymin, ymax, points];
  CompiladorTex[%, "PreferredEditor" → "TeXstudio"]
```

En **Wolfram Mathematica** se despliega como salida la animación de la subfigura 14.3a y los mensajes:

```
{30, 6, True, orange, True, -1.2, 1.2, -1.2, 1.2, True}
```

Usando editores ya detectados: 4 disponibles

⋮

PDF generado: /Users/.../Downloads/CreateTikZAnimation3/CreateTikZAnimation3.pdf

/Users/.../Downloads/CreateTikZAnimation3/CreateTikZAnimation3.pdf

El archivo *CreateTikZAnimation3.tex* (guardado en *Downloads/CreateTikZAnimation3*) integra el

código $\text{\LaTeX}$ editable que retorna la animación $\text{\TikZ}$ de la subfigura 14.3b (no manipulable desde este PDF, pues la animación produce un peso computacional considerable).

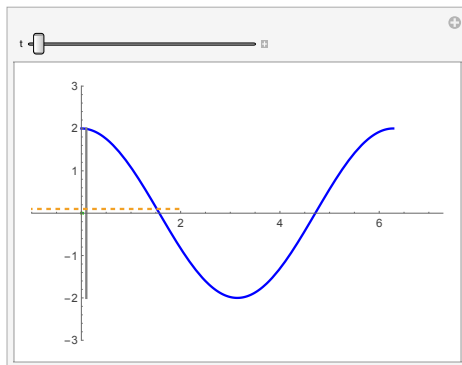
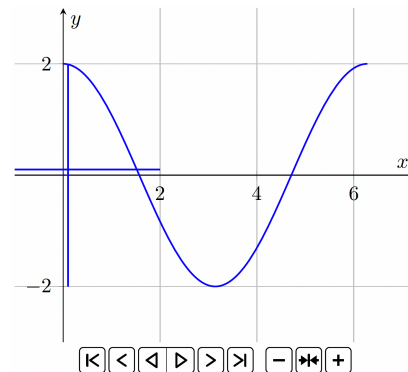
(a) Animación de **Mathematica**(b) Animación $\text{\TikZ}$

Figura 14.4: Animaciones del ejemplo 110

Ejemplo 110 Combinaciones de varios tipos

```
Manipulate[
  Show[Plot[2*Cos[x], {x, 0, 2 Pi}, PlotStyle ->{Blue, Thick},
    PlotRange ->{{-1, 2 Pi + 1}, {-3, 3}},
    ParametricPlot[{s - Sin[s], 1 - Cos[s]}, {s, 0, t},
    PlotStyle ->{Red, Thick}],
    PolarPlot[0.3*theta, {theta, 0, t},
    PlotStyle ->{Green, Thick}],
    ParametricPlot[{t, s}, {s, Sin[t]}, {s, -2, 2},
    PlotStyle ->{Gray, Dashed, Thin}],
    PlotRange ->{{-1, 2 Pi + 1}, {-3, 3}}, {t, 0.1, 2 Pi}]]
{frames = 50, framesxSeg = 12, grid = RandomChoice[{True, False}],
  style = RandomChoice[{"red", "blue", "green", "orange", "purple",
    "black"}], ejes = True, xmin = -1, xmax = 2 Pi + 1, ymin = -3,
  ymax = 3, points = RandomChoice[{True, False}]}
CreateTikZAnimation[
  Function[{t},
    Show[Plot[2*Cos[x], {x, 0, 2 Pi}, PlotStyle ->{Blue, Thick},
    PlotRange ->{{-1, 2 Pi + 1}, {-3, 3}},
    ParametricPlot[{s - Sin[s], 1 - Cos[s]}, {s, 0, t},
    PlotStyle ->{Red, Thick}],
    PolarPlot[0.3*theta, {theta, 0, t},
    PlotStyle ->{Green, Thick}],
    ParametricPlot[{t, s}, {s, Sin[t]}, {s, -2, 2},
    PlotStyle ->{Gray, Dashed, Thin}],
    PlotRange ->{{-1, 2 Pi + 1}, {-3, 3}}], {t, 0.1, 2 Pi}],
  "CreateTikZAnimation4.tex", frames, framesxSeg, grid, style, ejes,
  xmin, xmax, ymin, ymax, points];
CompiladorTex[%, "PreferredEditor" ->"TeXstudio"]]
```

En **Mathematica** la salida comparte la animación de la subfigura 14.4a y los mensajes de proceso:

```
{50, 12, True, red, True, -1, 1 + 2 Pi, -3, 3, True}
```

```
Usando editores ya detectados: 4 disponibles
```

```
:
```

```
PDF generado: /Users/.../Downloads/CreateTikZAnimation4/CreateTikZAnimation4.pdf
```

/Users/.../Downloads/CreateTikZAnimation4/CreateTikZAnimation4.pdf

El archivo *CreateTikZAnimation4.tex* (guardado en *Downloads/CreateTikZAnimation4*) contiene el código $\text{\LaTeX}$ editable que genera la animación $\text{\TikZ}$ de la subfigura 14.4b (**no manipulable** desde este PDF, por la misma razón del ejemplo anterior).

Para más ejemplos

Si se desea obtener un conjunto más amplio de ejemplos de uso del comando **CreateTikZ-Animation**, los ofrece el archivo: *14. Ejemplos CreateTikZAnimation.nb*, junto con los ejercicios anteriormente expuestos. Este *file* se encuentra disponible en el enlace de descarga del paquete $\text{\LaTeX}$ (ver página xiv).

El desarrollo y documentación del paquete **V_lT_EX** presentado en este texto representa un avance significativo en la integración entre entornos de cálculo simbólico y sistemas de composición tipográfica profesional. A través de sus catorce módulos especializados, **V_lT_EX** establece un paradigma en la automatización de procesos de documentación científica que trasciende las limitaciones de soluciones parciales existentes como *MaTeX*, *matlab2tikz*, y el desaparecido *Mathematica2tikz*.

El sistema de renderizado **ẼT_EX** desarrollado demuestra capacidades superiores en la conversión de expresiones matemáticas, físicas y químicas a representaciones tipográficamente consistentes. El sistema compilador **ẼT_EX** integrado elimina la fragmentación tradicional del flujo de trabajo al detectar automáticamente editores instalados y facilitar la compilación directa desde **Mathematica**. Particularmente notable es el sistema de *plotting* con estilo **ẼT_EX**, que incorpora fuentes especializadas y temas predefinidos para mantener coherencia visual con publicaciones académicas estándar.

Las funciones de exportación constituyen el núcleo técnico más avanzado del paquete. La implementación de exportadores especializados para primitivas 2D y 3D, junto con el manejo automático de objetos **Graphics3D** complejos, establece nuevos estándares de calidad en la conversión vectorial. Una innovación revolucionaria del sistema es su capacidad para generar automáticamente animaciones paramétricas interactivas que se integran completamente en documentos PDF, permitiendo la visualización dinámica de procesos matemáticos evolutivos mediante controles de reproducción nativos. Las funciones para grafos, autómatas, máquinas de estado, secciones cónicas, y diagramas de contorno abordan necesidades específicas frecuentemente encontradas en la enseñanza y comunicación de conceptos matemáticos.

V_lT_EX democratiza el acceso a herramientas de documentación profesional al proporcionar interfaces simplificadas que mantienen compatibilidad con funcionalidades nativas de **Mathematica**. La arquitectura modular permite una adopción gradual según las necesidades específicas, reduciendo significativamente las barreras técnicas para investigadores no especializados en **ẼT_EX**.

El módulo de casos especiales aborda problemas computacionales específicos como programación lineal con visualización de regiones factibles, expandiendo la aplicabilidad del paquete a múltiples disciplinas.

A diferencia de otras soluciones, **V_lT_EX** proporciona una respuesta innovadora que abarca renderizado, compilación, visualización, animación interactiva y exportación en un ecosistema coherente. La capacidad de generar documentos completos de forma automatizada, incluyendo artículos científicos, reportes técnicos, libros y presentaciones *Beamer* con contenido tanto estático como dinámico, posiciona al paquete como una herramienta integral para la producción académica moderna.

Si bien **V_lT_EX** representa un avance considerable, la complejidad inherente del sistema requiere una curva de aprendizaje para el aprovechamiento óptimo de todas sus funcionalidades. La dependencia de **Mathematica** como plataforma base puede limitar la adopción en entornos con restricciones de licenciamiento. Adicionalmente, el rendimiento en exportaciones complejas puede verse afectado por la naturaleza *on-the-fly* de la generación de contenido.

Se recomienda la adopción de **L^AT_EX** en flujos de trabajo académicos que requieren documentación frecuente y de alta calidad tipográfica. Es una herramienta que beneficiará particularmente a investigadores que manejan documentos con notación matemática compleja, diagramas especializados, visualizaciones dinámicas, o requieren consistencia visual entre cálculos y presentación final.

Este trabajo contribuye significativamente al avance del estado del arte en herramientas de documentación científica automática y demuestra la viabilidad de sistemas integrados que combinan potencia computacional con excelencia tipográfica y capacidades de visualización dinámica. **L^AT_EX** no solo resuelve problemas técnicos específicos, sino que redefine las expectativas sobre integración entre cálculo y documentación en la era del contenido interactivo. Se espera que el presente documento proporcione una ruta comprensiva que facilite su adopción y su extensión como una herramienta disponible para la comunidad académica.

Bibliografía

- [1] M. Kohlhase and F. Rabe, “QED reloaded: Towards a pluralistic formal library of mathematical knowledge,” *Journal of Formalized Reasoning*, vol. 9, no. 1, pp. 201–234, 2016. DOI: [10.6092/issn.1972-5787/4570](https://doi.org/10.6092/issn.1972-5787/4570).
- [2] S. Horvát, *LaTeX typesetting in mathematica*, Retrieved from <http://szhorvat.net/mathematica/MaTeX>, 2015. [Online]. Available: <http://szhorvat.net/mathematica/MaTeX>.
- [3] matlab2tikz Development Team, *Matlab2tikz: Convert MATLAB/octave figures to TikZ/pgfplots figures for smooth integration into LaTeX*, GitHub, Software, 2025. [Online]. Available: <https://github.com/matlab2tikz/matlab2tikz>.
- [4] W. Thomson, *Mathematica2tikz*, Wolfram Library Archive MathSource, Software, 2016. [Online]. Available: <https://library.wolfram.com/infocenter/ID/9554/>.
- [5] M. Andreoli, “Mathematica and latex integration,” *Journal of Applied & Computational Mathematics*, vol. 5, no. 3, pp. 1–4, 2016. [Online]. Available: <https://www.hilarispublisher.com/open-access/mathematica-and-latex-integration-2168-9679-1000308.pdf>.
- [6] Z. B. Etienne and I. Ruchlin, *Nrpylatex: LaTeX interface for SymPy, Mathematica, Maple, and other computer algebra packages*, GitHub repository, 2023. [Online]. Available: <https://github.com/zachetienne/nrpylatex>.
- [7] N. Rahman, R. J. Bauer, and A. Khandelwal, “LaTeX tutorial for the standardization and automation of population analysis reports,” *CPT: Pharmacometrics & Systems Pharmacology*, vol. 10, no. 11, pp. 1310–1322, 2021. DOI: [10.1002/psp4.12701](https://doi.org/10.1002/psp4.12701).
- [8] ResearchGate, *Automated LaTeX generation from academic PDF: Practical workflow using GPT-4.1*, Online resource, 2024. [Online]. Available: <https://promptrevolution.poltextlab.com/automated-latex-generation-from-an-academic-pdf-practical-workflow-using-gpt-4-1>.
- [9] Wolfram Research, *TeXForm*, Wolfram Language Documentation Center, 2025. [Online]. Available: <https://reference.wolfram.com/language/ref/TeXForm.html>.
- [10] Wolfram Community, *LaTeX typesetting in mathematica: Technical discussion*, Wolfram Community Forums, 2019. [Online]. Available: <https://community.wolfram.com/groups/-/m/t/457288>.
- [11] T. Tantau, *The tikz and pgf packages: Manual for version 3.1.11a*, CTAN, 2025. [Online]. Available: <https://mirrors.ucr.ac.cr/CTAN/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf>.