Solución de Ecuaciones Diferenciales mediante Google Colab

Daniel Steven Moran Pizarro
Universidad Nacional Abierta y a Distancia

2025

Resumen

Este documento presenta una guía práctica sobre el uso de Google Colab para la resolución de ecuaciones diferenciales, con énfasis en la aplicación de la transformada de Laplace. Se abordan los tres tipos principales de ecuaciones estudiadas en el curso: de primer orden, de orden superior y mediante transformada de Laplace. El enfoque se centra en la implementación computacional tanto de métodos analíticos (con SymPy) como numéricos (con SciPy), incluyendo ejemplos concretos y aplicables. Esta guía está diseñada para estudiantes del curso de Ecuaciones Diferenciales de la UNAD, brindando herramientas accesibles y prácticas para la visualización y validación de soluciones en un entorno de programación basado en la nube.

Índice

1. Introducción

Las ecuaciones diferenciales son herramientas matemáticas fundamentales que modelan fenómenos donde la tasa de cambio es relevante. El curso de Ecuaciones Diferenciales (código 100412) de la UNAD aborda tres unidades principales:

1. Ecuaciones diferenciales de primer orden
2. Ecuaciones diferenciales de orden superior
3. Solución mediante la transformada de Laplace

Google Colab permite resolver estas ecuaciones de manera eficiente combinando métodos analíticos y numéricos. Este documento ofrece una guía práctica para estudiantes sobre cómo utilizar esta herramienta en el contexto del curso.

2. Google Colab: Configuración Inicial

Google Colab es un entorno de notebooks Jupyter basado en la nube que incluye bibliotecas científicas preinstaladas. Para acceder:

1. Visita https://colab.research.google.com/ con tu cuenta de Google
2. Selecciona «Archivo» → «Nuevo notebook en Drive»

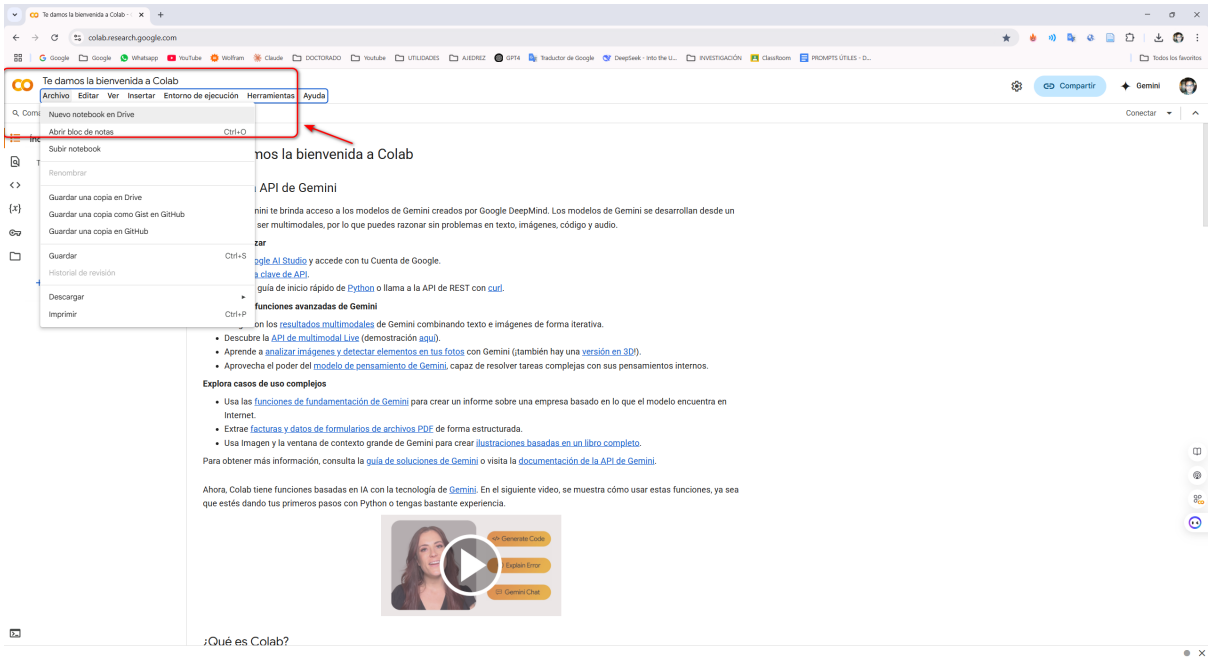

Figura 1: Interfaz inicial de Google Colab

Para trabajar con ecuaciones diferenciales, comienza importando las bibliotecas necesarias:

```
# Importar bibliotecas esenciales
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate
import sympy as sp
from sympy.abc import t, s, x, y
from sympy import Function
from sympy import dsolve
from sympy import Eq
from sympy import symbols
from sympy import exp
from sympy import sin
from sympy import cos
from sympy import laplace_transform
from sympy import inverse_laplace_transform

# Configuración inicial para la visualización
plt.figure(figsize=(10, 6))
sp.init_printing(use_unicode=True)

# Activar visualización de gráficos en línea (solo para Jupyter/Colab)
%matplotlib inline
```

3. Resolución de Ecuaciones Diferenciales de Primer Orden

Las ecuaciones diferenciales de primer orden tienen la forma:

$$\frac{dy}{dx} = f(x, y)$$

3.1. Método Analítico con SymPy

SymPy permite resolver ecuaciones diferenciales analíticamente:

```
import sympy as sp
from IPython.display import display, Latex

# Definir variables y ecuación
x = sp.symbols('x')
y = sp.Function('y')(x)
ecuacion = sp.Eq(y.diff(x), 4*x**3)  # dy/dx = 4x³

# Resolver la ecuación
solucion = sp.dsolve(ecuacion, y)
display(Latex(r"\textbf{Solución general:}"))
display(Latex(f"$$ {sp.latex(solucion)} $$"))

# Aplicar condición inicial: y(1) = 5
C1 = sp.symbols('C1')
```

```
condicion_inicial = sp.Eq(solucion.rhs.subs(x, 1), 5)
valor_C1 = sp.solve(condicion_inicial, C1)[0]
display(Latex(r"\textbf{Valor de $C_1$:}"))
display(Latex(f"$$ C_1 = {sp.latex(valor_C1)} $$"))

# Solución particular
solucion_particular = sp.Eq(y, solucion.rhs.subs(C1, valor_C1))
display(Latex(r"\textbf{Solución particular:}"))
display(Latex(f"$$ {sp.latex(solucion_particular)} $$"))
```

Figura 2: Resultado de la resolución analítica en SymPy

3.2. Método Numérico con SciPy

Para ecuaciones que no tienen solución analítica sencilla, podemos usar métodos numéricos:

```
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt

# Definir la ecuacion diferencial como funcion
def ecuacion(x, y):
    return 4*x**3  # dy/dx = 4x³

# Resolver el problema de valor inicial
x_span = (1, 2)  # intervalo de integracion
y0 = [5]         # condicion inicial y(1) = 5
solucion = solve_ivp(ecuacion, x_span, y0, method="RK45", dense_output=True)
```

```python
# Visualizar resultados
x_vals = np.linspace(1, 2, 100)
y_vals = solucion.sol(x_vals)[0]

plt.figure()
plt.plot(x_vals, y_vals, 'b-', label='Solución numérica')
plt.plot(x_vals, x_vals**4 + 4, 'r--', label='Solución analítica')
plt.grid(True, alpha=0.3)
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Solución de dy/dx = 4x³ con y(1) = 5')
plt.show()
```

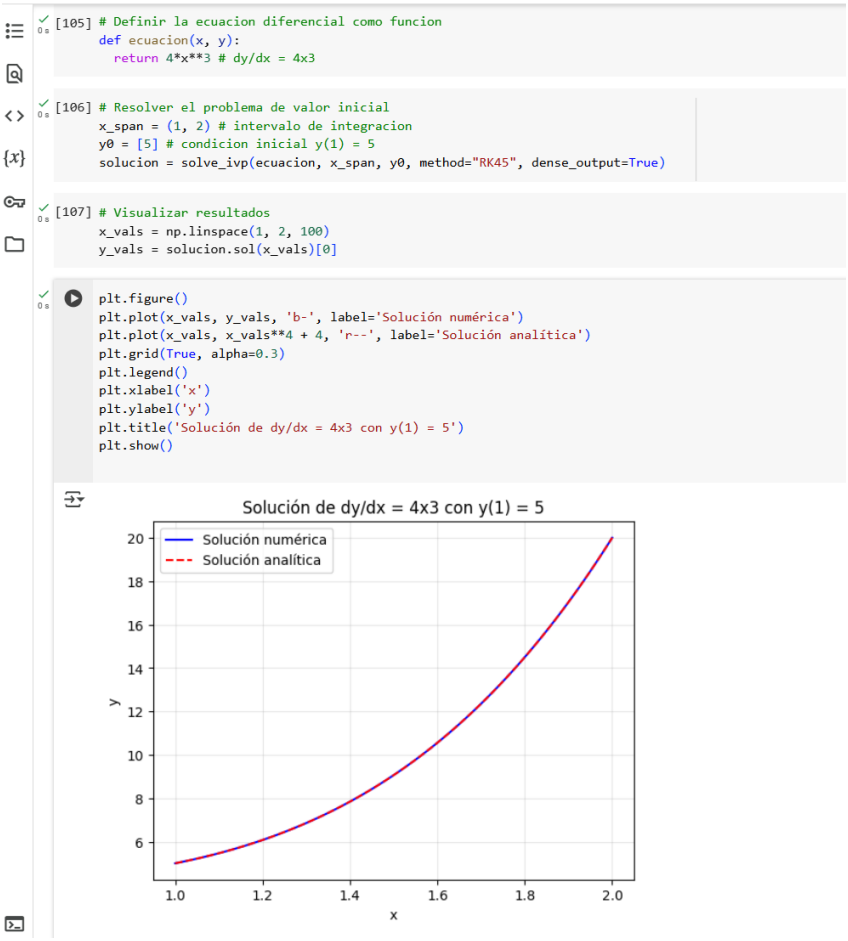

Figura 3: Comparación entre solución analítica y numérica

4. Ecuaciones Diferenciales de Orden Superior

Las ecuaciones diferenciales de orden superior (principalmente segundo orden) tienen la forma:

$$\frac{d^2y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = g(x)$$

4.1. Ecuaciones Homogéneas

Una ecuación homogénea tiene $g(x) = 0$. Ejemplo de resolución:

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Latex

# Definir variables y ecuación
t = sp.symbols('t')
y = sp.Function('y')(t)
ecuacion = sp.Eq(y.diff(t, 2) + 2*y.diff(t) - 3*y, 0)  # y'' + 2y' - 3y = 0

# Resolver la ecuación
solucion_general = sp.dsolve(ecuacion, y)
display(Latex(r"\textbf{Solución general:}"))
display(Latex(f"$$ {sp.latex(solucion_general)} $$"))

# Aplicar condiciones iniciales: y(0) = 1, y'(0) = 0
y_general = solucion_general.rhs
C1, C2 = sp.symbols('C1 C2')
eq1 = sp.Eq(y_general.subs(t, 0), 1)          # C1 + C2 = 1
eq2 = sp.Eq(y_general.diff(t).subs(t, 0), 0) # C1 - 3C2 = 0

# Resolver el sistema de ecuaciones
valores = sp.solve((eq1, eq2), (C1, C2))
display(Latex(r"\textbf{Valores de $C_1$ y $C_2$:}"))
display(Latex(f"$$ C_1 = {sp.latex(valores[C1])}, \quad C_2 =
↪  {sp.latex(valores[C2])} $$"))

# Solución particular
solucion_particular = y_general.subs({C1: valores[C1], C2: valores[C2]})
display(Latex(r"\textbf{Solución particular:}"))
display(Latex(f"$$ {sp.latex(sp.Eq(y, solucion_particular))} $$"))

# Graficar la solución particular
# Crear función lambda para evaluar numéricamente
y_solucion_func = sp.lambdify(t, solucion_particular, modules='numpy')

# Rango de t
t_vals = np.linspace(0, 5, 100)
y_vals = y_solucion_func(t_vals)

# Gráfica
plt.figure(figsize=(8, 5))
plt.plot(t_vals, y_vals, label=r'$y(t)$', color='blue')
plt.title('Solución particular de la ecuación diferencial')
```

```
plt.xlabel('t')
plt.ylabel('y(t)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.show()
```

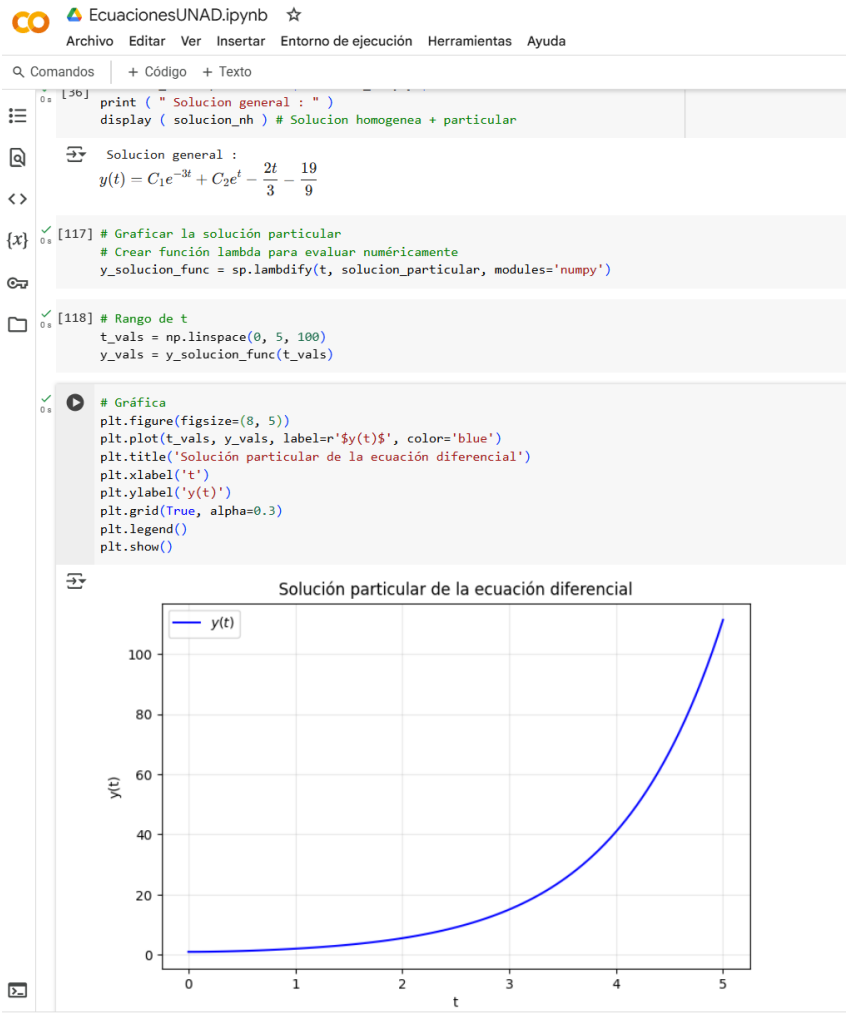

Figura 4: Ecuaciones homogéneas

4.2. Ecuaciones No Homogéneas

Una ecuación no homogénea tiene $g(x) \neq 0$. Estas se pueden resolver usando el método de coeficientes indeterminados:

```
import sympy as sp
from IPython.display import display, Latex

# Definir variables
t = sp.symbols('t')
y = sp.Function('y')(t)

# Definir ecuación no homogénea
```

```
ecuacion_nh = sp.Eq(y.diff(t, 2) + 2*y.diff(t) - 3*y, 2*t + 5)

# Resolver la ecuación
solucion_nh = sp.dsolve(ecuacion_nh, y)

display(Latex(r"\textbf{Solución general:}"))
display(Latex(f"$$ {sp.latex(solucion_nh)} $$"))
```

5. Transformada de Laplace

La transformada de Laplace es una herramienta poderosa para resolver ecuaciones diferenciales, especialmente con condiciones iniciales.

5.1. Definición y Propiedades

La transformada de Laplace de una función $f(t)$ se define como:

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

5.2. Cálculo de Transformadas de Laplace

```
import sympy as sp
from IPython.display import display, Latex  # <-- Esta línea es necesaria

# Definir variables
t, s = sp.symbols('t s')

# Transformadas de funciones básicas
funciones = [
    1,
    t,
    t**2,
    sp.exp(2*t),
    sp.sin(3*t),
    sp.cos(3*t),
    t*sp.exp(2*t),
    t*sp.sin(3*t)
]

# Mostrar título
display(Latex(r"\textbf{Transformadas de Laplace:}"))

# Calcular y mostrar transformadas
for f in funciones:
    L_f = sp.laplace_transform(f, t, s, noconds=True)
    display(Latex(f"$$L\\{{{sp.latex(f)}\\}} = {sp.latex(L_f)}$$"))

# Ejemplo más complejo
f_compleja = t - 2*sp.exp(4*t) + 3*sp.cos(sp.pi*t)
L_f_compleja = sp.laplace_transform(f_compleja, t, s, noconds=True)
```

```
display(Latex(r"\textbf{Transformada de Laplace de una función compuesta:}"))
display(Latex(f"$$L\\{{{sp.latex(f_compleja)}\\}} =
↪  {sp.latex(L_f_compleja)}$$"))
```

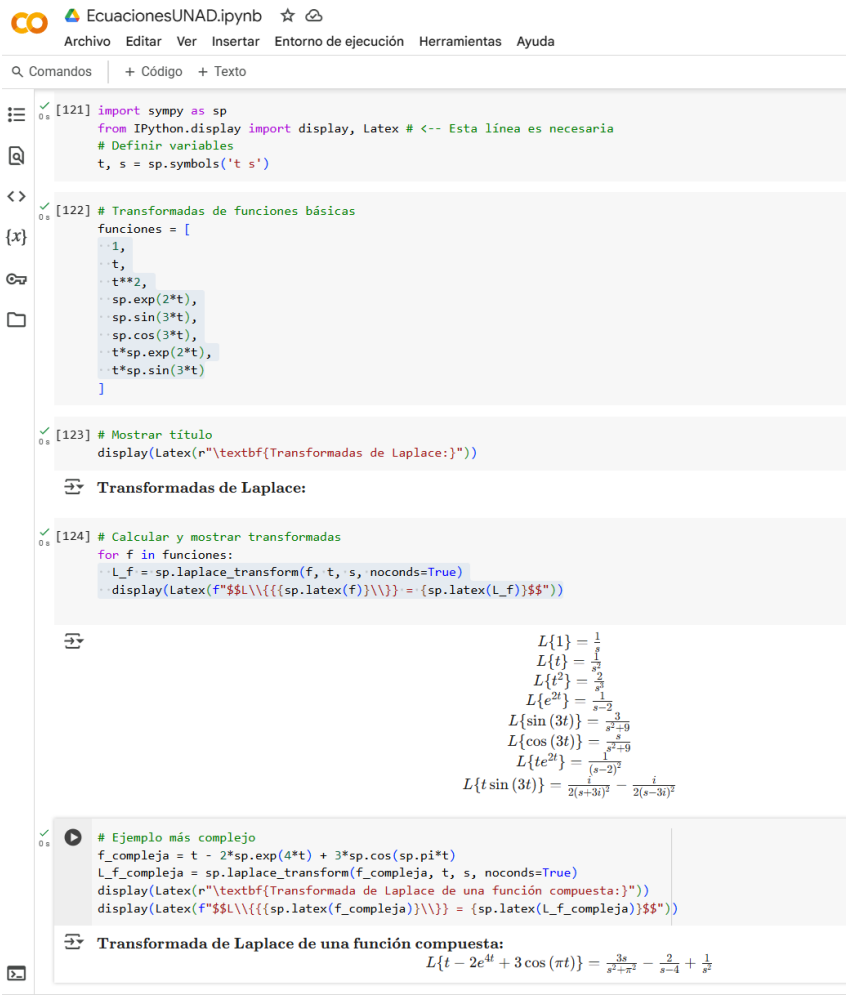

Figura 5: Resultados de transformadas de Laplace

5.3. Transformada Inversa de Laplace

La transformada inversa nos permite recuperar la función original:

```
import sympy as sp
from IPython.display import display, Latex

# Definir variables
t, s = sp.symbols('t s')

# Transformadas inversas básicas
F_s_ejemplos = [
    1/s,              # 1/s
    1/s**2,           # 1/s²
    1/(s-2),          # 1/(s-2)
```

```
    s/(s**2 + 4),      # s/(s² + 4)
    3/(s**2 + 4)       # 3/(s² + 4)
]

# Mostrar título
display(Latex(r"\textbf{Transformadas inversas de Laplace (suponiendo
↪  $t\geq0$):}"))

# Calcular y mostrar transformadas inversas
for F in F_s_ejemplos:
    f_t = sp.inverse_laplace_transform(F, s, t)
    # Eliminar visualmente theta(t) (escalón unitario)
    f_t_sin_theta = f_t.replace(sp.Heaviside(t), 1)
    display(Latex(f"$$L^{{-1}}\\{{{sp.latex(F)}\\}} =
    ↪  {sp.latex(f_t_sin_theta)}$$"))

# Ejemplo con fracciones parciales
F_compleja = (7*s - 6)/(s**2 - 2*s)

# Descomposición en fracciones parciales
F_descomp = sp.apart(F_compleja, s)
display(Latex(r"\textbf{Descomposición en fracciones parciales:}"))
display(Latex(f"$$ {sp.latex(F_descomp)} $$"))

# Transformada inversa de la fracción
f_inversa = sp.inverse_laplace_transform(F_compleja, s, t)
# Eliminar visualmente theta(t) en la inversa
f_inversa_sin_theta = f_inversa.replace(sp.Heaviside(t), 1)
display(Latex(r"\textbf{Transformada inversa final:}"))
display(Latex(f"$$f(t) = {sp.latex(f_inversa_sin_theta)}$$"))
```

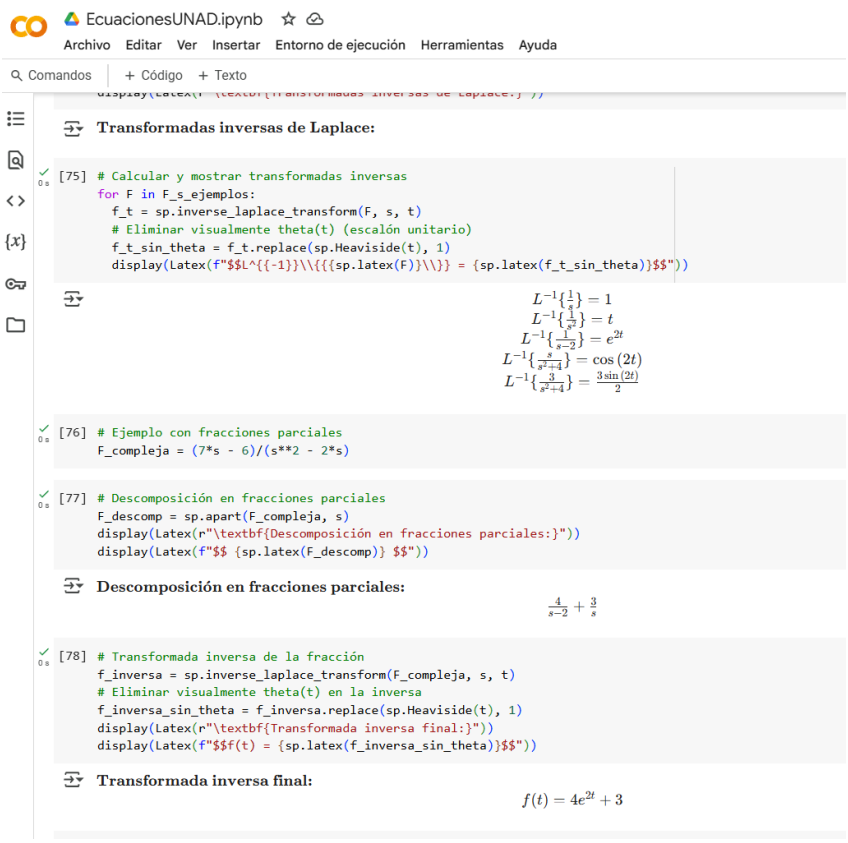

Figura 6: Resultados de transformadas inversas de Laplace

5.4. Resolución de Ecuaciones Diferenciales mediante Transformada de Laplace

El procedimiento general es: 1. Aplicar la transformada de Laplace a la ecuación diferencial 2. Resolver la ecuación algebraica para $Y(s)$ 3. Aplicar la transformada inversa para obtener $y(t)$

```
import sympy as sp
from IPython.display import display, Latex

# Definir variables
t, s = sp.symbols('t s')
y = sp.Function('y')(t)
y_prime = sp.diff(y, t)
y_dprime = sp.diff(y, t, 2)

# Definir la ecuación diferencial
eq = sp.Eq(y_dprime + 3*y_prime + 2*y, 0)  # y'' + 3y' + 2y = 0

# Condiciones iniciales
y0, y_prime_0 = 1, 0  # y(0) = 1, y'(0) = 0

# Paso 1: Aplicar transformada de Laplace
Y = sp.symbols('Y', cls=sp.Function)(s)
L_y_dprime = s**2 * Y - s*y0 - y_prime_0  # L{y''} = s²Y(s) - sy(0) - y'(0)
```

```
L_y_prime = s * Y - y0                    # L{y'} = sY(s) - y(0)
L_y = Y                                   # L{y} = Y(s)

# Ecuación transformada
L_eq = L_y_dprime + 3*L_y_prime + 2*L_y
display(Latex(r"\textbf{Ecuación transformada:}"))
display(Latex(f"$$ {sp.latex(L_eq)} = 0 $$"))

# Paso 2: Resolver para Y(s)
Y_s = sp.solve(sp.Eq(L_eq, 0), Y)[0]
display(Latex(r"\textbf{Solución para $Y(s)$:}"))
display(Latex(f"$$ Y(s) = {sp.latex(Y_s)} $$"))

# Descomposición en fracciones parciales
Y_s_descomp = sp.apart(Y_s, s)
display(Latex(r"\textbf{Descomposición en fracciones parciales:}"))
display(Latex(f"$$ {sp.latex(Y_s_descomp)} $$"))

# Paso 3: Transformada inversa
y_t = sp.inverse_laplace_transform(Y_s, s, t)

# Eliminar visualmente theta(t) (escalón unitario)
y_t_sin_theta = y_t.replace(sp.Heaviside(t), 1)

display(Latex(r"\textbf{Transformada inversa para obtener $y(t)$ (suponiendo
↪  $t\geq0$):}"))
display(Latex(f"$$ y(t) = {sp.latex(y_t_sin_theta)} $$"))
```

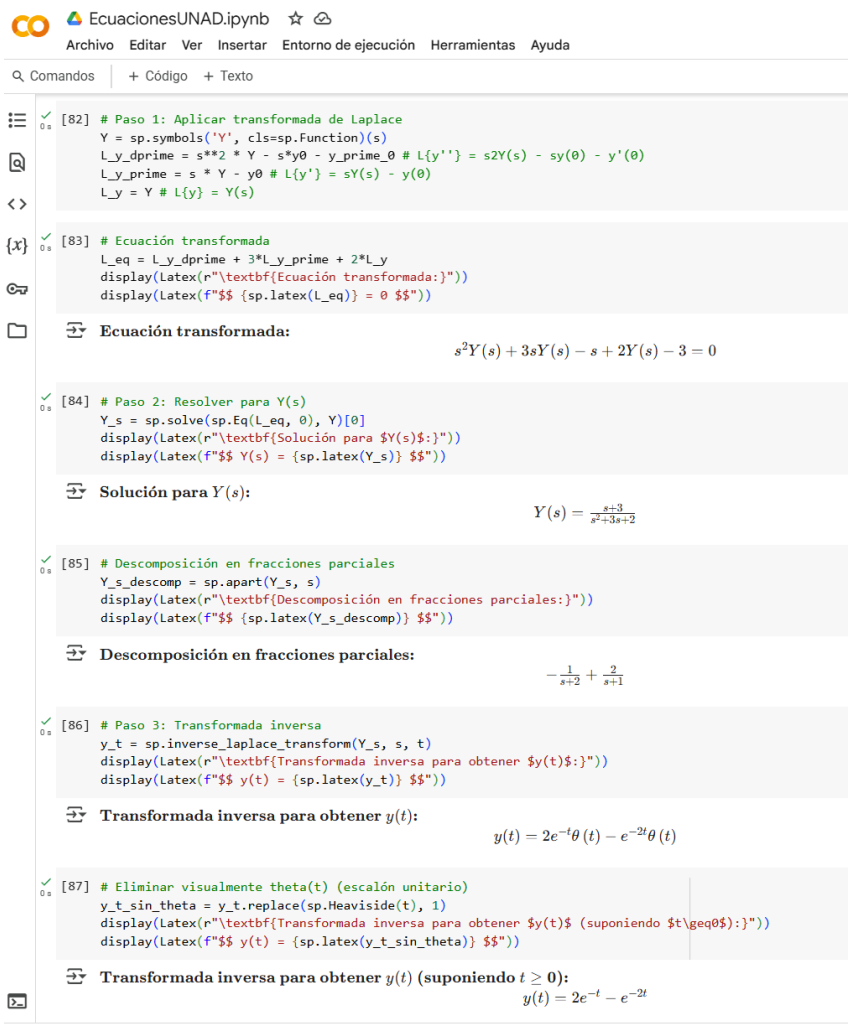

Figura 7: Resolución de ecuación diferencial mediante transformada de Laplace

6. Consejos Prácticos

6.1. Errores Comunes y Soluciones

- **Error en condiciones iniciales**: Verifica que las condiciones estén correctamente aplicadas en la transformada de Laplace.
- **Problemas con fracciones parciales**: Usa `sp.apart()` para descomponer fracciones complejas.
- **Problemas con simplificaciones**: Utiliza `sp.simplify()` para expresiones complicadas.
- **Error en importaciones**: Para evitar errores de sintaxis, es mejor importar cada función en una línea separada:

```
# Forma más segura para copiar y pegar las funciones esenciales de SymPy
from sympy import Function
from sympy import dsolve
from sympy import Eq
```

```
from sympy import symbols
from sympy import exp
from sympy import sin
from sympy import cos
```

O usar la instrucción `import sympy as sp` y acceder a las funciones mediante `sp.`:

```
# Alternativa más concisa para importar SymPy
import sympy as sp

# Y luego usar, por ejemplo:
solucion = sp.dsolve(ecuacion, y)
```

6.2. Verificación de Soluciones

Siempre verifica tus soluciones sustituyéndolas en la ecuación original:

```
import sympy as sp
from IPython.display import display, Latex

# Definir la solución y calcular sus derivadas
t = sp.symbols('t')
y_sol = sp.exp(-t) - sp.exp(-2*t)
y_sol_prime = sp.diff(y_sol, t)
y_sol_dprime = sp.diff(y_sol, t, 2)

# Sustituir en la ecuación original
verif = y_sol_dprime + 3*y_sol_prime + 2*y_sol

# Mostrar verificación
display(Latex(r"\textbf{Verificación (debe ser aproximadamente 0):}"))
display(Latex(f"$$ {sp.latex(sp.simplify(verif))} $$"))
```

7. Aplicaciones Prácticas

Las ecuaciones diferenciales modelan diversos fenómenos físicos. Ejemplos:

7.1. Modelos Físicos Comunes

- **Ley de Malthus** (crecimiento poblacional): $\frac{dP}{dt} = kP$
- **Ley de enfriamiento de Newton**: $\frac{dT}{dt} = -k(T - T_a)$
- **Circuito RLC**: $L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = E(t)$
- **Péndulo simple**: $\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0$ (pequeñas oscilaciones)

8. Conclusiones

Google Colab proporciona un entorno poderoso para la resolución de ecuaciones diferenciales, combinando:

- Métodos analíticos con SymPy
- Métodos numéricos con SciPy
- Visualización con Matplotlib

Las transformadas de Laplace simplifican notablemente la resolución de ecuaciones diferenciales lineales con condiciones iniciales, especialmente en sistemas más complejos.

Este documento ha presentado las herramientas básicas para resolver ecuaciones diferenciales de primer orden, de orden superior y mediante transformada de Laplace en Google Colab, ofreciendo un recurso práctico para los estudiantes del curso de Ecuaciones Diferenciales.

Referencias

[1] Bisong, E. (2019). Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59–64). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7

[2] García Hernández, A. E. (2015). *Ecuaciones diferenciales*. Grupo Editorial Patria.

[3] Mesa, F. (2012). *Ecuaciones diferenciales ordinarias: una introducción*. Ecoe Ediciones.

[4] Zill, D. G. (2018). *Ecuaciones diferenciales con aplicaciones de modelado* (11a ed.). Cengage Learning.

[5] Meurer, A., Smith, C. P., Paprocki, M., et al. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, **3**, e103.

[6] Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, **17**, 261–272.