INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPÚTO

ESCOM

Trabajo Terminal

“Prototipo para la solución de ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes”

2014-A017

Presentan
Aca Cruz Jonathan Vladimir
Stolar Hernández

Directores
M.C. Miguel Olvera Aldana ***Dr. Encarnación Salinas Hernández***

Junio 2015

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPÚTO
SUBDIRECCIÓN ACÁDEMICA

No de TT: 2014-A017 Junio2015

Documento Técnico

"Prototipo para la solución de ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes"

Presentan
Aca Cruz Jonathan Vladimir
Stolar Hernández

Directores
M.C. Miguel Olvera Aldana ***Dr. Encarnación Salinas Hernández***

Resumen

En las escuelas de ingeniería donde se imparte el curso de Ecuaciones Diferenciales, se estudian y analizan de manera general las Ecuaciones Diferenciales Ordinarias y Lineales; para la solución usamos métodos conocidos, como lo son coeficientes indeterminados, reducción de orden, variación de parámetros y transformada de Laplace. Pero cuando en la E.D.L los coeficientes no son constantes la solución se complica, de hecho en estos cursos sólo nos enfocamos a resolver aquella donde el orden de la ecuación diferencial corresponde a un coeficiente variable polinomial del mismo grado, está estructura bien conocida es llamada "Ecuación de Cauchy-Euler". La idea de este trabajo es generar un prototipo que sea capaz de resolver la ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes usando una técnica original y de autoría propia del Dr. Encarnación Salinas Hernández.

locus0002@hotmail.com
mixtek@gmail.com

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITUCIONAL

COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES

México, D.F. a 23 de Julio de 2015

DR. FLAVIO ARTURO SÁNCHEZ GARFIAS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJOS TERMINALES
P R E S E N T E

Por medio del presente, informamos que los alumnos que integran el **TRABAJO TERMINAL 2014-A017** titulado "Prototipo para la Solución de Ecuaciones Diferenciales Lineales no Homogéneas de Segundo Orden con Coeficientes no Constantes" concluyeron satisfactoriamente su trabajo.

El empastado del Reporte Técnico Final y el Disco (DVD) fueron revisados ampliamente por sus servidores y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

MIGUEL OLVERA ALDANA

ENCARNACIÓN SALINAS HERNÁNDEZ

Índice

Índice de figuras

Capítulo 1

Antecedentes

1.1. Introducción

Esta sección presenta la información relativa al marco teórico y conceptual del trabajo terminal, conceptos básicos relacionados y expone el estado del arte de las soluciones existentes para el problema del trabajo a resolver.

1.2. Justificación

Las Ecuaciones Diferenciales Ordinarias No Lineales de **Segundo Orden** pueden tener solución basándose en la ecuación de Riccati:

$$y' = y^2 - a_1(x)u - a_2(x) \tag{1.1}$$

En general no existe una **solución completa anlítica**, es decir, se trata de un problema abierto.

Actualmente, el software existente en el mercado que puede dar solución a Ecuaciones Diferenciales Lineales No Homogéneas de Segundo Orden con Coeficientes No Constantes requiere de un conocimiento previo amplio para su uso correcto, y en muchos casos no puede generar una solución.

Ejemplos

Mathematica

```
1 DSolve[y''[x]  + x^2y'[x] + cos[xy]== 0, y[x], x]
```

Derive

```
y'' + x^2y' + cosxy = 0
```

Para esta ecuación, el software antes mencionado no puede generar una solución.

1.3. Derivación e Integración

1.3.1. Variables y funciones

1.3.1.1. Una variable

Es una cantidad a la que se le puede asignar, durante el curso de un proceso de análisis, una numero ilimitado de valores. Las variables se designan usualmente por las últimas letras del alfabeto. Una cantidad que durante el curso de un proceso tiene un valor fijo se llama constante. Baldor (1997).

1.3.1.2. Constantes numéricas o absolutas

Son las que conservan los mismos valores en todos los problemas, como 2, 5, 7, etc. Constantes arbitrarias, o parámetros, son aquellas a las que se pueden asignar valores numéricos, y que durante todo el proceso se conservan esos valores asignados. Usualmente se representa por las primeras letras del alfabeto.
Así en la ecuación de la recta,

$$\frac{x}{a} + \frac{y}{b} = 1 \tag{1.2}$$

Donde x y y son las coordenadas variables de un punto que se mueve sobre la línea, mientras que a y b son las coordenadas arbitrarias que representan la abscisa en el origen y la ordenada en el origen, las cuales se supone que son los valores definidos para cada recta.
El valor numérico (o absoluto) de una constante a, para diferenciarlo de su valor algebráico, se representa por $|a|$. Así,

$$|-2| + 2 = |4| \tag{1.3}$$

El símbolo $|a|$ se lee: "valor numérico de a .º "valor absoluto de a". Baldor (1997).

1.3.1.3. Funciones

Cuando dos variables están relaciones de tal manera que el valor de la primera queda determinado si se da un valor a la segunda, entonces se dice que la primera es función de la segunda.
Casi todos lo problemas científicos tratan con cantidades y relaciones de esta naturaleza, y en la experiencia de la vida diaria nos encontramos constantemente con situaciones en las intervienen magnitudes dependientes unas de otras. Baldor (1997).

1.3.1.4. Variables independiente y dependientes

La segunda variable puede asignar valores a voluntad dentro de los límites que dependen del problema particular se llama variable independiente o argumento. La primera variable, cuyo valor queda fijado cuando se asigna un valor a la variable independiente se llama variable dependiente o función.

Frecuentemente, cuando se consideran dos variables ligadas entre sí, queda a nuestro arbitrio el elegir a una de ellas como variable independiente; pero una vez hecha esta elección, no es permitido cambiar de variable independiente sin tomar ciertas precauciones y hacer las transformaciones pertinentes. El área de un cuadrado, por ejemplo, es una función de la longitud del lado, y, recíprocamente, la longitud del lado es una función del área.Granville (1998).

1.3.1.5. Notación de funciones

El símbolo $f(x)$ se emplea para designar una funcionan de x, y se lee f de x. Con objeto de distinguir entre diferentes funciones se cambia la letra inicial, como $F(x)$, $\theta(x)$, $f'(x)$, etc. Granville (1998)

1.3.2. Derivación

1.3.2.1. Definición

La derivada de una función f es aquella función, denotada por f', tal que su valor en un número x del dominio de f está dado por

$$f'(x_1) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Si x_1 es un número particular del dominio de f, entonces

$$f'(x_1) = \lim_{\Delta x \to 0} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x}$$

Leithold (1998)

1.3.2.2. Reglas generales de derivación

1. Derivada de una constante
$$\frac{dy}{dx} = 0$$

2. Derivada de una variable con respecto a sí misma
$$\frac{dx}{dx} = 1$$

3. Derivada de una suma algebraica.
$$\frac{d}{dx}(u + v - w) = \frac{du}{dx} + \frac{dv}{dx} - \frac{dw}{dx}$$

4. La derivada del producto de una constante por una función.
$$\frac{d}{dx}(cv) = c\frac{dv}{dx}$$

5. La derivada del producto de dos funciones.
$$\frac{d}{dx}(uv) = u\frac{dv}{dx} + v\frac{du}{dx}$$

6. Derivada del producto de n funciones, siendo n un número finito.
$$\frac{d}{dx}(v^n) = nv^{n-1}\frac{dv}{dx}$$

7. Derivada de la potencia de una función siendo el exponente constante.
$$\frac{d}{dx}(x^n) = nx^{n-1}$$

8. Derivada de un cociente de funciones.
$$\frac{d}{dx}(\frac{u}{v}) = \frac{v\frac{du}{dx} - u\frac{dv}{dx}}{v^2}$$

1.3.2.3. Derivadas Sucesivas

Puede ocurrir que la derivada de una función respecto a x sea también derivable; en este caso la derivada de la primera derivada se llama la segunda derivada de la función primitiva. Análogamente, la derivada de la segunda derivada se llama la tercera derivada, y así, sucesivamente hasta la enésima derivada.
Así, si:

$$\begin{aligned} y &= 3x^4, \\ \frac{dy}{dx} &= 12x^3, \\ \frac{d^2y}{dx^2} &= \frac{d}{dx}(\frac{dy}{dx}) = 36x^2, \\ \frac{d^3y}{dx^3} &= \frac{d}{dx}(\frac{d^2y}{dx^2}) = 72x. \end{aligned}$$

Etc. Granville (1998).

1.3.3. Integración

1.3.3.1. Reglas generales para la integración

1. La integral de una suma algebráica de expresiones diferenciales es igual a la suma algebráica de las integrales de esas expresiones

$$\int (du + dv - dw) = \int du + \int dv - \int dw$$

2. Un factor constante puede escribirse delante de la integral

$$\int adv = a \int dv$$

3. la integral de una expresión diferencial

$$\int dv = v + C$$

4. La integral de la potencia n donde n es diferente de -1

$$\int v^n dv = \frac{v^{n+1}}{n+1} + C$$

5. La integral de la potencia n donde n es -1

$$\int \frac{dv}{v} = lnv + C$$

1.3.3.2. Integral por partes

Si u y v son funciones de la misma variable independiente, tenemos, según la formula para la diferenciación de un producto.

$$d(uv) = udv + vdu,$$

Transponiendo,

$$udv = d(uv) - vdu,$$

Integrando, resulta la fórmula inversa,

$$\int udv = uv - \int vdu$$

que se llama fórmula de ***Integración por Partes***. Esta fórmula hace que su integración depende de la de dv y u de du, que pueden ser formas fáciles de integrar.
Para aplicar esta fórmula, debe descomponerse la diferencial dada en dos factores, a saber, u y dv. No pueden darse instrucciones generales para la elección de esos factores, pero son útiles las siguientes:

1. dx es siempre una parte de dv;

2. debe ser posible integrar dv.

3. Cuando la expresión para integrar es el producto de dos funciones, ordinariamente es mejor elegir la de apariencia más complicada, con tal que pueda integrarse, como parte dv

Granville (1998).

1.4. Ecuaciones Diferenciales

Una ecuación que contiene las derivadas de una o más variables dependientes con respecto a una o más variables independientes es una ecuación diferencial.

Las ecuaciones diferenciales se clasifican de acuerdo con su tipo, orden y linealidad. Dennis G. Zill (2008)

1.4.1. Tipo

Si una ecuación sólo contiene derivadas ordinarias de una o más variables dependientes con respecto a una sola variable independiente, entonces se dice que es una ecuación diferencial ordinaria. Ejemplo:

$$\frac{dy}{dx} + 5y = e^x; \frac{d^2y}{dx^2} - \frac{dy}{dx} + 6y = 0; \frac{dx}{dt} + \frac{dy}{dt} = 2x + y \tag{1.4}$$

Son ecuaciones diferenciales ordinarias.

Una ecuación que contiene las derivadas parciales de una o más variables dependientes, respecto de dos más variables independientes, se llama ecuación diferencial en derivadas parciales. Ejemplo:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0; \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2} - 2\frac{\partial u}{\partial t}; \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{1.5}$$

Dennis G. Zill (2008).

1.4.2. Orden

El orden de una ecuación diferencial (ordinaria o en derivadas parciales) es el de la derivada de mayor orden en la ecuación. Ejemplo:

$$\mathbf{\frac{d^2y}{dx^2}} + 5(\frac{dy}{dx})^3 \tag{1.6}$$

Es una ecuación diferencial ordinaria de segundo orden.

En ocasiones, las ecuaciones diferenciales ordinarias de primer orden se escriben en la forma diferencial $M(x, y)dy + N(x, y)dx = 0$.

Por ejemplo, si suponemos que y representa la variable dependiente en

$$(y - x)dx + 4xdy = 0$$

entonces

$$y' = dy/dx$$

por lo que, al dividir entre la diferencial dx se obtiene la forma alternativa

$$4xy' + y = x$$

Se puede expresar una ecuación diferencial de orden n, en una variable dependiente, en la forma general:

$$F(x, y, y', ..., y^{(n)}) = 0 \tag{1.7}$$

Dennis G. Zill (2008).

1.4.3. Linealidad

Se dice que una ecuación diferencial ordinaria de orden n es lineal cuando la ecuación es de la forma descrita por:

$$a_n(x)\frac{d^n y}{dx^n} + a_{n-1}(x)\frac{d^{n-1}y}{dx^{n-1}} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y = g(x) \tag{1.8}$$

En esta última ecuación, vemos las dos propiedades características de las ecuaciones diferenciales lineales:

1. La variable dependiente y y todas sus derivadas son de primer grado; esto es, el exponente de todo término donde aparece y es 1.

2. Cada coeficiente sólo depende de x, que es la variable independiente.

   Las ecuaciones

$$(y-x)dx + 4xdy = 0; y'' - 2y' + y = 0; \frac{d^3y}{dx^3} + x\frac{dy}{dx} - 5y = e^x \tag{1.9}$$

Son a su vez ecuaciones diferenciales lineales ordinarias de primer, segundo y tercer orden.

Una ecuación diferencial ordinaria, no lineal simplemente es aquella que no es lineal. Las funciones no lineales de la variable dependiente o de sus derivadas, como por ejemplo $sen(y)$ o e^y. No pueden aparecer en una ecuación lineal. Ejemplo:

$$(1-y)y' + 2y = e^x; \frac{d^2y}{dx^2} + sen(y) = 0; \frac{d^4y}{dx^4} + y^2 = 0 \tag{1.10}$$

Ecuaciones diferenciales no lineales. Dennis G. Zill (2008).

1.4.4. Métodos de Resolución

Los métodos de resolución a ecuaciones diferenciales que se abordarán en el presente trabajo terminal son los siguientes:

1.4.4.1. Método de Variables Separables

Solución por integración. Comenzaremos nuestro estudio de la metodología para resolver ecuaciones de primer orden, $\frac{dy}{dx} = f(x,y)$, con la más sencilla de todas las ecuaciones diferenciales. Cuando f es independiente de la variable y. Esto es, cuando $f(x,y) = g(x)$ la ecuación diferencial:

$$\frac{dy}{dx} = g(x) \tag{1.11}$$

Se resolverá por integración, si $g(x)$ es una función continua, al integrar ambos lados, se llega a la solución $y = \int g(x)dx = G(X) + c$, en donde $G(X)$ es una antiderivada (o integral indefinida) de $g(x)$; por ejemplo, si $\frac{dy}{dx} = 1 + e^{2x}$, entonces $y = \int(1 + e^{2x})dx$, esto es $y = x + \frac{1}{2}(e^{2x}) + c$. La ecuación, y su método de solución, es solo un caso especial cuando f, en $\frac{dy}{dy} = f(x, y)$ es un producto de una función de x y una función de y. Se dice que una ecuación diferencial de primer orden de la forma

$$\frac{dy}{dx} = g(x)h(y) \tag{1.12}$$

Es separable, o de variables separables. Por ejemplo, las ecuaciones:

$$\frac{dy}{dx} = y^2xe^{3x+4y} \quad y \quad \frac{dy}{dx} = y + sen(x) \tag{1.13}$$

Son separables y no separables, respectivamente. En la primera, factorizamos $f(x, y) = y^2xe^{3x+y}$ como sigue:

$$f(x, y) = y^2xe^{3x+4y} = \overbrace{(xe^{3x})}^{g(x)}\overbrace{(y^2e^{4y})}^{h(y)} \tag{1.14}$$

Pero en la segunda ecuación no hay manera de expresar $y + sen(x)$ como un producto de una función de x por una función de y. Observe que, al dividir entre la función, $h(y)$, una ecuación separable se escribe de la forma:

$$p(y)\frac{dy}{dx} = g(x) \tag{1.15}$$

Donde, por comodidad, $p(y)$ representa a $1/h(y)$. Así, vemos de inmediato que la ecuación se reduce cuando $h(y) = 1$. Ahora bien, si $y = \phi(x)$ representa una solución, se debe cumplir $p(\phi(x))\phi'(x) = g(x)$, asi que:

$$\int p(\phi(x))\phi'(x)dx = \int g(x)dx \tag{1.16}$$

Pero $dy = \phi'(x)dx$, de modo que

$$\int p(y)dy = \int g(x)dx \quad o \quad sea \quad H(y) = G(x) + c \tag{1.17}$$

en donde $H(y)$ y $G(x)$ son antiderivadas de $p(y) = 1/h(y)$ y de $g(x)$, respectivamente.Dennis G. Zill (2008).

1.4.4.2. Método de Coeficientes Indeterminados

Este método da solución a ecuaciones diferenciales lineales no homogéneas con coeficientes constantes. La solución consta de dos partes:

$$Y_G = Y_H + Y_P \tag{1.18}$$

Donde:

- Y_G = Solución General
- Y_H = Solución Homogénea
- Y_P = Solución Particular

Para resolver la homogénea asociada, suponemos que la solución $y = Ce^{mx}$.
Sustituyendo esto en la Ecuación Diferencial:

$$\begin{aligned} y &= Ce^{mx} \\ \frac{dy}{dx} &= y' = Cme^{mx} \\ \frac{dy}{dx^2} &= y'' = Cm^2e^{mx} \end{aligned}$$

Tenemos:

$$a_2Cm^2e^{mx} + a_1Cme^{mx} + a_0Ce^{mx} = 0$$

Factorizamos:

$$\begin{aligned} Ce^{mx}(a_2m^2 + a_1m + a_0) = 0 \\ Ce^{mx} \neq 0 \Rightarrow (a_2m^2 + a_1m + a_0) = 0 \end{aligned}$$

El resultado es un polinomio de segundo grado, el cual recibe el nombre de ***polinomio característico***.
Resolviendo el polinomio característico utilizaremos:

$$\begin{aligned} a_2m^2 + a_1m + a_0 \Rightarrow \\ m1, 2 = \frac{a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2} \end{aligned}$$

Para lo que tenemos tres tipos diferentes de solución:

1.4.4.2.1. Caso 1

$$m_1 \neq m_2 \in \mathbb{R} \Rightarrow a_1^2 - 4a_2a_0 > 0$$

Cuya solución es:

$$y = C_1e^{m_1x} + C_2e^{m_2x} \tag{1.19}$$

1.4.4.2.2. Caso 2

$$m_1 = m_2 = m \in \mathbb{R} \Rightarrow a_1^2 - 4a_2a_0 = 0$$

Cuya solución es:

$$y = C_1e^{m_1x} + C_2xe^{m_2x} \tag{1.20}$$

1.4.4.2.3. Caso 3

$$m_1 \neq m_2 \in \mathbb{C} \Rightarrow a_1^2 - 4a_2a_0 < 0$$

Cuya solución es:

$$y = e^{\alpha x}(C_1cos\beta + C_2sen\beta x) \tag{1.21}$$

$$m = \alpha \pm i\beta$$

Para resolver la ecuación particular, se asume que $f(x)$ es continua y derivable en todo su dominio.

Ejemplos de $f(x)$ y sus propuestas de solución:

Ejemplos	**Propuestas**
$f(x)$	Y_P
3	$A = constante$
x	$Ax + B$
x^3	$Ax^3 + Bx^2 + Dx + E$
$cos(x)/sen(x)$	$Acos(x) + Bsen(x)$
$e^{\alpha x}$	$Ae^{\alpha x}$

Ejemplo de resolución por **Coeficientes Indeterminados**:

Resolver la Ecuación Diferencial no homogénea $y'' + 4y = e^{3x}$.

Solución: Primero resolvemos la homogénea asociada.

$$\begin{aligned} m^2 + 4 &= 0 \\ m_{1,2} &= \pm 2i \\ \alpha = 0, \beta &= 2 \\ Y_H &= C_1 cos(2x) + C_2 sen(2x) \end{aligned}$$

Segundo, resolvemos la particular:

$$\begin{aligned} f(x) &= e^{3x} \\ Y_P &= Ae^{3x} \end{aligned}$$

Sustituyendo en la Ecuación Diferencial:

$$\begin{aligned} y' = 3Ae^{3x}, y'' &= 9Ae^{3x} \\ 9Ae^{3x} + 4Ae^{3x} &= 13Ae^{3x} \\ 13Ae^{3x} = e^{3x} \Rightarrow Y_P &= \frac{1}{13}e^{3x} \end{aligned}$$

La solución general es:

$$Y_G = C_1 cos(2x) + C_2 sen(2x) + \frac{1}{13}e^{3x}$$

Dennis G. Zill (2008).

1.4.4.3. Método de Variación de Parámetros

La propuesta de solución para esta Ecuación Diferencial es la siguiente:

$$Y_P = U_1 y_1 + U_2 y_2 \tag{1.22}$$

donde y_1 y y_2 son solución de la Ecuación Diferencial y $U_1' = \frac{W_1}{W}$ y $U_2' = \frac{W_2}{W}$
Nota:

$$W = \begin{bmatrix} y_1 & y_2 \\ y_1' & y_2' \end{bmatrix} \tag{1.23}$$

Es el **wronskiano del sistema**.

$$W_1 = \begin{bmatrix} 0 & y_2 \\ f(x) & y_2' \end{bmatrix} \tag{1.24}$$

$$W_2 = \begin{bmatrix} y_1 & 0 \\ y_1' & f(x) \end{bmatrix} \tag{1.25}$$

Ejemplo de resolución por **Variación de Parámetros**:

Resolver la Ecuación Diferencial $4y'' + 36y = cosc(3x)$.

Solución: Primero resolvemos la homogénea asociada.

$$\begin{aligned} 4m^2 + 36m &= 0 \\ m^2 &= \frac{-36}{4} \\ m^2 &= -9 \Rightarrow m_{1,2} && = \pm 3i \\ Y_H &= C_1 cos(3x) + C_2 sen(3x) \end{aligned}$$

Segundo, calculamos $Y_P = U_1 y_1 + U_2 y_2$.

$$U_1' = \frac{W_1}{W} = \frac{\begin{bmatrix} 0 & sen(3x) \\ csc(3x) & 3cos(3x) \end{bmatrix}}{\begin{bmatrix} cos(3x) & sen(3x) \\ -3sen(3x) & 3cos(3x) \end{bmatrix}} = \frac{-sen(3x)csc(3x)}{3cos^2(3x) + 3sen^2 3x} = -\frac{1}{3}$$

$$\int U_1' = -\int \frac{1}{3} dx$$

$$U_1 = -\frac{1}{3}x$$

$$U_2' = \frac{W_2}{W} = \frac{\begin{bmatrix} cos(3x) & 0 \\ -3sen(3x) & csc(3x) \end{bmatrix}}{[3]} = \frac{ctg(3x)}{3} = \frac{cos(3x)}{3sen(3x)}$$

$$\int U_2' = \frac{1}{3} \int \frac{cos(3x)}{sen(3x)} dx$$

$$U_2 = \frac{1}{9} ln(sen(3x))$$

La solución particular es:

$$-\frac{1}{3}xcos(3x) + \frac{1}{9}ln(sen(3x))sen(3x)$$

Finalmente

$$Y_G = C_1 cos(3x) + C_2 sen(3x) - \frac{1}{3}xcos(3x) + \frac{1}{9}ln(sen(3x))sen(3x)$$

Dennis G. Zill (2008).

1.4.4.4. Ecuación de Cauchy-Euler

Una ecuación diferencial lineal de la forma:

$$a_n x^n \frac{d^n y}{dx^n} + a_{n-1} x^{n-1} \frac{d^{n-1} y}{dx^{n-1}} + ... + a_1 x \frac{dy}{dx} + a_0 y = g(x), \tag{1.26}$$

donde los coeficientes $a_n, a_{n-1}, ?, a_0$ son constantes, tiene los nombres de ecuación de Cauchy-Euler, ecuación de Euler o ecuación equidimensional. Las características observable de este tipo de ecuación es el grado $k = n, n-1, ..., 1, 0$ de los coeficientes monomiales x^k coincide con el orden k de la diferenciación $d^k y/dx^k$:

$$a_n x^n \frac{d^n y}{dx^n} + a_{n-1} x^{n-1} \frac{d^{n-1}}{dx^{n-1}} + ... \tag{1.27}$$

Comenzaremos la discusión examinando detalladamente las formas de las soluciones generales de la ecuación homogénea de segundo orden.

$$ax^2 \frac{d^2 y}{dx^2} + bx \frac{dy}{dx} + cy = 0. \tag{1.28}$$

La solución de ecuaciones de orden superior será análoga. Una vez determinada la función complementaria y_c también podemos resolver la ecuación no homogénea $ax^2 y'' + bxy' + cy = g(x)$ con el método de variación de parámetros.

Método de solución Intentaremos una solución de la forma $y = x^m$, donde m está por determinar. En forma parecida a lo que sucedió al sustituir e^{mx} en una ecuación lineal con coeficientes constantes, cuando se sustituye x^m, cada término de una ecuación de Cauchy-Euler se trasforma en un polinomio en m, multiplicado por x^m, porque

$$a_k x^k \frac{d^k y}{dx^k} = a_k x^k m(m-1)(m-2)...(m-k+1)x^{m-k} = a_k m(m-1)(m-2)...(m-k+1)x^m \tag{1.29}$$

Por ejemplo, al sustituir $y = x^m$, la ecuación de segundo orden se trasforma en

$$ax^2 \frac{d^2 y}{dx^2} + bx \frac{dy}{dx} + cy = am(m-1)x^m + bmx^m + cx^m = (am(m-1) + bm + c)x^m \tag{1.30}$$

Así, $y = x^m$, es una solución de la ecuación diferencial siempre que m sea una solución de la ecuación auxiliar

$$am(m-1) + bm + c = 0 \quad o \quad am^2 + (b-a)m + c = 0 \tag{1.31}$$

Hay tres casos distintos por considerar que dependen de si las raíces de esta ecuación cuadrática son reales y distintas, reales repetidas (o iguales)o complejas. En el último caso las raíces serán un par conjugado.

1.4.4.4.1. Caso I (Raíces reales distintas) Sean m_1, m_2 las raíces reales, tales que $m_1 \neq m_2$. Entonces $y_1 = x^{m_1}$ y $y_2 = x^{m_2}$ forman un conjunto fundamental de soluciones. Así pues la solución general es:

$$y = c_1 x^{m_1} + c_2 x^{m_2} \tag{1.32}$$

Ejemplo: Resolver

$$x^2 \frac{d^2y}{dx^2} - 2x\frac{dy}{dx} - 4y = 0$$

Solución. En lugar de memorizar la ecuación, para comprender el origen y la diferencia entre esta nueva forma de la ecuación auxiliar y la que obtuvimos, las primeras veces es preferible suponer que la solución es $y = x^m$. Diferenciamos dos veces

$$\frac{dy}{dx} = mx^{m-1}, \qquad = m(m-1)x^{m-2}, \tag{1.33}$$

Y sustituimos en la ecuación diferencial

$$x^2 \frac{d^2y}{dx^2} - 2x\frac{dy}{dx} - 4y = x^2 m(m-1)x^{m-2} - 2xmx^{m-1} - 4x^m \tag{1.34}$$

$$= x^m(m(m-1) - 2m - 4) = x^m(m^2 - 3m - 4) = 0 \tag{1.35}$$

si $m^2 - 3m - 4 = 0$.
Pero $(m+1)(m-4) = 0$,
significa que $m_1 = -1 y m_2 = 4$,
asi que

$$y = C_1 x^{-1} + C_2 x^4 \tag{1.36}$$

1.4.4.4.2. Caso II (Raíces reales repetidas) Si las raíces son repetidas (esto es, si $m1 = m2$), solo llegaremos a una solución, que es $y = x^{m_1}$. Cuando las raíces de las ecuación cuadrática $am^2 + (b-a)m + c = 0$ son iguales, el discriminante de los coeficientes tiene que ser cero. De acuerdo con la fórmula cuadrática, la raíz deber ser $m_1 = -(b-a)/2a$. Podemos formar ahora una segunda solución y_2. Primero escribimos la ecuación de Cauchy-euler en la forma

$$\frac{d^2y}{dx^2} + \frac{b}{ax}\frac{dy}{dx} + \frac{c}{ax^2}y = 0 \tag{1.37}$$

e identificamos $P(x) = b/ax$ e $\int (b/ax)dx = (b/a)lnx$
Así

$$y_2 = x^{m_1} \int \frac{e^{-(b/a)lnx}}{x^{2m_1}} \tag{1.38}$$

$$= x^{m_1} \int x^{-b/a} x^{-2m_1} dx \tag{1.39}$$

$$= x^{m_1} \int x^{-b/a} x^{(b-a)/a} dx \tag{1.40}$$

$$= x^{m_1} \int \frac{dx}{x} = x^{m_1} lnx. \tag{1.41}$$

Entonces la solución general es

$$y = c_1 x^{m_1} + c_2 x^{(}m_2) lnx \tag{1.42}$$

Ejemplo: Resolver

$$4x^2 \frac{d^2y}{dx^2} + 8x \frac{dy}{dx} + y = 0 \tag{1.43}$$

Solución: La sustitución $y = x^m$ da

$$4x^2 \frac{d^2y}{dx^2} + 8x \frac{dy}{dx} + y = x^m(4m(m-1)+8m+1) = x^m(4m^2+4m+1) = 0 \tag{1.44}$$

cuando $4m^2 + 4m + 1 = 0$, o $(2m + 1)^2 = 0$. Como $m_1 = -\frac{1}{2}$, la solucion general es Para las ecuaciones de orden superior se puede demostrar que si m_1, es raíz de multiplicidad k, entonces

$$x^{m_1}, \quad x^{m_1} lnx, \quad x^{m_1} (lnx)^2, ..., x^{m_1} (lnx)^{k-1} \tag{1.45}$$

Son k soluciones linealmente independiente. En consecuencia, la solución general de la ecuación diferencial debe contener una combinación lineal de esas k soluciones.

1.4.4.4.3. Caso III (Raíces complejas conjugadas) Si las raíces son el par conjugado $m_1 = \alpha + i\beta, m_2 = \alpha - i\beta$, donde α y $\beta > 0$ son reales, una solución es

$$y = C_1 x^{\alpha + i\beta} + C_2 x^{\alpha - i\beta} \tag{1.46}$$

Pero cuando las raíces de la ecuación son complejas, como en el caso de ecuaciones con coeficientes constantes, conviene formular la solución solo en términos de funciones reales. Vemos la identidad

$$x^{i\beta} = (e^{lnx}) i\beta lnx \tag{1.47}$$

que, según la formular de Euler, es lo mismo que

$$x^{i\beta} = cos(\beta lnx) + isen(\beta lnx) \tag{1.48}$$

$$x^{-i\beta} = cos(\beta lnx) - isen(\beta lnx) \tag{1.49}$$

Sumamos y restamos los últimos dos resultados para obtener

$$x^{i\beta} + x^{-i\beta} = 2cos(\beta lnx) \quad x^{i\beta} - x^{-i\beta} = 2isen(\beta lnx) \tag{1.50}$$

Respectivamente. Basándonos en que $y = C_1x^{\alpha+i\beta} + C_2x^{\alpha-i\beta}$ es una solución para todos los valores de las constantes vemos, a la vez, para $C_1 = C_2 = 1$ y $C_1 = 1, C_2 = -1$, que

$$y_1 = x^{\alpha}(x^{i\beta} + x^{-i\beta}) \quad y_2 = x^{\alpha}(x^{i\beta} - x^{-i\beta}) \tag{1.51}$$

$$y_1 = 2x^{\alpha}cos(\beta lnx) \quad y_2 = 2ix^{\alpha}sen(\beta lnx) \tag{1.52}$$

También son soluciones. Como $W(x^{\alpha}cos(\beta lnx), x^{\alpha}sen(\beta lnx)) = \beta x^{2\alpha-1} \neq 0, \beta > 0$ en el intervalo $(0,)$, llegamos a la conclusión que

$$y_1 = x^{(}\alpha)cos(\beta lnx) \quad y_2 = x^{\alpha}sen(\beta lnx) \tag{1.53}$$

Forman un conjunto fundamental de soluciones reales de la ecuación diferencial por lo tanto la solución general es

$$y + x^{\alpha}[C_1cos(\beta lnx) + C_2sen(\beta lnx)] \tag{1.54}$$

Dennis G. Zill (2008).

Capítulo 2

Estado del Arte

2.1. Introducción

En este capítulo se describen los resultados de la investigación relizada respecto al cómputo simbólico y los diferentes lenguajes de programación que lo implementan, con objeto de discriminar entre los mismos y tomar una decisión sobre cuál utilizaremos para este trabajo.

2.2. Cómputo Simbólico

En matemáticas y ciencias de la computación, el álgebra computacional, también llamado cómputo simbólico o álgebra simbólica es un área científica que se refiere al estudio y desarrollo de algoritmos y software para la manipulación de expresiones y objetos matemáticos. Ann Boyle y Hearn (1990).

El cómputo simbólico hace énfasis en la exactitud de los cálculos con expresiones que contienen variables a las que no se les han dado algún valor y por ende son manipuladas como símbolos.

Las aplicaciones de software que desarrollan cálculos simbólicos son llamados comunmente sistemas de álgebra computacional; el término sistema alude a la complejidad de la aplicación principal que incluye, al menos, un método para representar datos matemáticos en una computadora, un lenguaje de programación para el usuario (generalmente es distinto al lenguaje de programación utilizado para la implementación del sistema) un administrador de memoria dedicado, una interfaz de usuario para la entrada y salida de expresiones matemáticas, un gran conjunto de rutinas para desarrollar

operaciones comunes, como la simplificación de expresiones, diferenciación usando regla de la cadena, factorización polinomial, integración indefinida, etc.

La programación simbólica, lejos de considerarse como un paradigma de programación por sí mismo, podría ser considerado como parte de la programación funcional, la cual sí es un paradigma de programación en el que las funciones, no objetos ni procedimientos, son usadas como los bloques de construcción fundamentales del programa.

Análogamente, en la programación simbólica el código es tratado como datos y viceversa.

Por ejemplo, si uno realiza la operación 1.0/3*3 en un lenguaje de programación que no está orientado a la matemática simbólica, digamos, Lenguaje *C*, primero se divide 1.0 entre 3 y luego se multiplica el resultado por 3. Lo que nos arroja el resultado 0.9999999999998. Lenguajes de matemática simbólica, como Macsyma, Lisp, Prolog, etc. y software como Mathematica trata los números como símbolos. La implementación de la operación anterior, 1/3*3, no es muy diferente de a/b*b hasta que se necesita el resultado en forma numérica, y como a/b*b es un 1/3*3, que toma el valor de 1.

2.3. Lenguajes de Programación Simbólica

2.3.1. Macsyma

El estado del arte del cómputo y la programación simbólica se fue desarrollando desde finales de la década de los 60's con el inicio del desarrollo de *Macsyma* (Proyect MAC's SYmbolic MAnipulator), sistema de álgebra computacional cuyo desarrollo original comprendió de 1968 a 1982 como parte del *Project MAC* del MIT. Muchas de las ideas de Macsyma fueron adoptadas después por sistemas más recientes como Mathematica, Maple y otros.

Un ejemplo de código de Macsyma en el cual se eleva un binonio al cuadrado y se muestra la salida:

```
1 C1: expand((a+b)^2);
2 A^2 + 2*A*B + B^2;
```

La programación simbólica no es nueva. El lenguaje *Lisp* es el segundo lenguaje de programación de alto nivel más antigüo, sólo precedido por *Fortran*.

2.3.2. Lisp

Lisp es una familia de lenguajes de programación que se distingue por imponer la notación prefija. Lisp es un lenguaje orientado a expresiones, lo que significa que todo el código y los datos son escritos como expresiones para después ser evaluados. Cuando una expresión es evaluada, produce un valor, el cual puede ser embebido en otra expresión. Cada valor puede tener cualquier tipo de dato.

Los dialectos más comunes de Lisp son *Common Lisp* y *Scheme*.

Mismo código de Macsyma pero escrito en Lisp.

```
1 CL-USER> (expand '(^ (+ a b) 2))
2 ==> (+ (^ a b) (* a b) (^ b 2))
```

2.3.2.1. Common Lisp

Common Lisp es un dialecto del lenguaje de programación Lisp. Es un lenguaje de programación multiparadigma que soporta orientación a objetos mediante *CLOS* (Common Lisp Object System), programación funcional y procedural. Fue desarrollado para tratar de estandarizar las variantes divergentes de Lisp. Touretzky (1990).

Existe un sistema de álgebra computacional (CAS) escrito en Common Lisp llamado ***Maxima*** y publicado bajo licencia GNU GPL. Algunas de las funcionalidades de Maxima son la manipulación simbólica de matrices, polinomios, derivación, integración, expansiones de series de potencias y de Fourier, manejo de gráficos 2D y 3D, entre otras.

Para realizar cálculo diferencial en Maxima se utiliza la función **diff(expr)**, la cual devuelve la derivada o diferencial de *expr* respecto de alguna o de todas las variables presentes en **expr**.
La llamada **diff(expr, x, n)** devuelve la n-ésima derivada de **expr** respecto de x.
La llamada **diff(expr, x_1, n_1, ..., x_m, n_m)** devuelve la derivada parcial de **expr** con respecto de x_1, ..., x_m.
La llamada **diff(expr)** devuelve el diferencial total de **expr**, esto es, la suma

de las derivadas de **expr** respecto de cada una de sus variables, multiplicadas por el diferencial **del** de cada una de ellas.

Ejemplos de derivación, diferencial total y derivada de orden superior:

```
(%i1) diff(sin(x), x);
(%o1)                                cos(x)
(%i2) diff(log(x));
                                     del(x)
(%o2)                                ------
                                       x
(%i3) diff(tan(x), x, 4);
                                2          3              4
(%o3)                  8 sec (x) tan (x) + 16 sec (x) tan(x)
```

Ejemplo de una expansión de polinomio y derivación con respecto a una sola variable:

```
(%i1) u: expand ((x+y)^3);
                          3        2      2      3
(%o1)                    y  + 3 x y  + 3 x  y + x
(%i2) diff(%,x);
                                2                2
(%o2)                        3 y  + 6 x y + 3 x
```

Se utilizó % para referirse al último resultado obtenido.

2.3.2.2. FEMLISP

FEMLISP es una biblioteca para resolver ecuaciones diferenciales, pero su método de solución está basado en el *Método del Elemento Finito (FEM)*, el cual es un método numérico para encontrar soluciones aproximadas a problemas de ecuaciones diferenciales parciales. Neuss (2010).

2.3.3. GSL (GNU Scientific Library)

GSL (GNU Scientific Library) es una biblioteca implementada en **C** y en **Lisp**, la cual puede dar solución a problemas de ecuaciones diferenciales ordinarias con valores iniciales. Esta librería utiliza métodos como Runge-Kutta, el cual es un método numérico que se basa en series de potencias para la resolución, y el método Bulirsch-Stoer, el cual es un método de análisis numérico.

Hay muchas herramientras que si bien no están escritas en un lenguaje de programación simbólica, implementan la lógica de los mismos en su propio lenguaje. Un ejemplo de esto es el lenguaje de programación Python, el cual es un lenguaje multiparadigma ya que soporta orientación a objetos, programación imperativa y en menor medida programación funcional. Dicho lenguaje no soporta Matemática Simbólica nativamente, pero cuenta con una biblioteca llamada *SymPy* para este fin. Neuss (2013).

2.3.4. Python - SymPy

SymPy es una biblioteca para matemáticas simbólicas escrita completamente en lenguaje Python, de código abierto y gratuita bajo Licencia BSD. Sus desarrolladores tienen planeado que se convierta en un sistema de álgebra computacional (CAS) completo en un futuro.

Contiene funcionalidad para realizar cálculos aritméticos, algebráicos, trigonométricos, cálculo diferencial e integral, teoría de números, matemáticas discretas y graficación de los mismos. Actualmente está en su versión 0.75. Team (2013).

2.3.4.1. SymPy - Ejemplos

Para hacer cálculo diferencial con SymPy, primeramente hay que importar la biblioteca en nuestro script de Python, declarar las variables que vayamos a usar en las derivadas y especificar el tipo de codificación que se usará en la salida del programa:

```
1 from sympy import *
2 #Definicion de variables
3 x, y, z = symbols('x y z')
4 init_printing(use_unicode=True)
```

Para evaluar derivadas, se utiliza la notación expuesta a continuación:
Ejemplo de la derivada del $\cos(x)$ respecto a x.

```
1 >>> diff(cos(x), x)
2 -sin(x)
```

Ejemplo de la derivada de la $\exp(x)^2$ respecto a x.

```
1 >>> diff(exp(x**2), x)
```

$2xe^{x^2}$

Ejemplo de la tercera derivada de x^4 respecto a x.

```
>>> diff(x**4, x, 3)
```

$24 \cdot x$

También se pueden derivar funciones respecto a varias variables, utilizando la notación antes mencionada.

En el siguiente ejemplo evaluaremos la siguiente derivada:

$$\frac{\partial^7}{\partial x \partial y^2 \partial z^4} e^{xyz}$$

Primeramente definimos la función a derivar mediante una variable llamada *expr*

```
>>> expr = exp(x*y*z)
>>> diff(expr, x, y, 2, z, 4)
```

El resultado es el siguiente:

$$x^3y^2\left(x^3y^3z^3 + 14x^2y^2z^2 + 52xyz + 48\right)e^{xyz}$$

Para crear un objeto derivada sin evaluar, se utiliza la clase *Derivative*. Esta clase tiene la misma sintaxis que *diff*. Para evaluar el objeto derivada que acabamos de crear, se utiliza el método *doit()*.

```
>>> deriv = Derivative(expr, x, y, y, z, 4)
>>> deriv
>>> deriv.doit()
```

Resultado:

$$x^3y^2\left(x^3y^3z^3 + 14x^2y^2z^2 + 52xyz + 48\right)e^{xyz}$$

Hay dos tipos de integrales, definidas e indefinidas. Para evaluar integrales indefinidas en *SimPy* se realiza de la misma manera que la evaluación de una derivada:

```
>>> integrate(cos(x), x)
sin(x)
```

Para evaluar una integral definida hay que especificar tres valores: la variable de integración, el límite inferior y el límite superior.

Ejemplo de la Integral Definida de 0 a ∞ (oo) de e^{-x}, dx:

```
>>> integrate(exp(-x), (x, 0, oo))
```

Al pasar múltiples tuplas de límites a evaluar, se realizará la evaluación de integrales múltiples.
Ejemplo de integral definida doble: $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-x^2-y^2}\,dx\,dy$

```
>>> integrate(exp(-x**2 - y**2), (x, -oo, oo), (y, -oo, oo)
    )
```

Si el método *integrate()* no puede evaluar la integral, se retorna un objeto Integral sin evaluar:

```
>>> expr = integrate(x**x, x)
>>> print(expr)
Integral(x**x, x)
```

2.3.5. Prolog

Prolog es un lenguaje de programación lógica y declarativa, en el cual se describe una situación mediante una sentencia y, utilizando Cálculo Proposicional y una base de conocimientos, se evalúa si dicha sentencia es verdadera o falsa. El intérprete o compilador dará una solución booleana y , si contiene variables, qué valores de las variables son necesarios para la evaluación.

Prolog está basado en la lógica de predicados de primero orden *(First Order Predicate Logic o FOPL)*.

Prolog es un lenguaje muy conocido en áreas académicas de las ciencias de la computación como es la Inteligencia Artificial y es especialmente utilizado para el reconocimiento de patrones y programación lógica. Wielemaker (1987).

2.3.5.1. Prolog - Ejemplos

Para hacer derivación e integración en Prolog primeramente se debe de declarar una base de conocimientos con los axiomas básicos, así como reglas de derivación e integración necesarias. En este ejemplo desarrollaremos la base de conocimientos para realizar operaciones de derivación en Prolog.

Axiomas de derivación

```
d( X, X, 1 ):- !.       /* d(X) con respecto a X es 1      */

```

```
d( C, X, 0 ):- atomic(C).  /* Si C es una constante entonces */
                           /* d(C)/dX es 0                    */

d( U+V, X, R ):-           /* d(U+V)/dX = A+B donde     */
   d( U, X, A ),           /* A = d(U)/dX y             */
   d( V, X, B ),       /* B = d(V)/dX               */
   R = A + B.

d( U-V, X, R ):-       /* d(U-V)/dX = A-B donde       */
   d( U, X, A ),       /* A = d(U)/dX y               */
   d( V, X, B ),       /* B = d(V)/dX                 */
   R = A - B.

d( C*U, X, R ):-       /* constante multiplicada por */
   atomic(C),          /* una variable               */
   C \= X,             /* d(C*X)/d(x)                */
   d( U, X, A ),
   R = C * A,
   !.
```

Derivada de producto, cociente y potencia de una función

```
d( U*V, X, U*B+V*A ):-      /* d(U*V)/dX = B*U+A*V donde */
   d( U, X, A ),            /* A = d(U)/dX y             */
   d( V, X, B ).            /* B = d(V)/dX               */

d( U/V, X, (A*V-B*U)/(V^2) ):-   /* d(U/V)/dX = (A*V-B*U)/(V*V) donde */
   d( U, X, A),               /* A = d(U)/dX y                    */
   d( V, X, B).               /* B = d(V)/dX                      */

d( U^C, X, R ):-             /* d(U^C)/dX = C*A*U^(C-1)    */
   atomic(C),                /* donde C es un numero       */
   C\=X,
   d( U, X, A ),
   R = C * A * U ^ ( C - 1 ).
```

Derivadas trigonométricas

```
d( sin(W), X, Z*cos(W) ):- /* d(sin(W))/dX = Z*cos(W)    */
   d( W, X, Z).            /* donde Z = d(W)/dX          */

d( cos(W), X, -(Z*sin(W)) ):- /* d(cos(W))/dX = -(Z*sin(W))   */
   d( W, X, Z).               /* donde Z = d(W)/dX           */

d( tan(W), X, (Z*sec(W)^2) ):- /* d(tan(W))/dX = Z*sec(W)^2   */
   d( W, X, Z).                /* donde Z = d(W)/dX           */

d( cot(W), X, -(Z*cosec(W)^2) ):- /* d(cot(W))/dX = -Z*cosec(W)^2   */
   d( W, X, Z).                   /* donde Z = d(W)/dX              */

d( sec(W), X, (Z*sec(W)*tan(W)) ):- /* d(sec(W))/dX = sec(W)*tan(W) */
   d( W, X, Z).                     /* donde Z = d(W)/dX           */

d( cosec(W), X, -(Z*cosec(W)*cot(W)) ):- /* d(cosec(W))/dX = -cosec(W)*cot(W) */
   d( W, X, Z).                          /* donde Z = d(W)/dX                */

d( arcsin(W), X, Z/sqrt(1-W^2) ):-  /* d(arcsin(W))/dX = Z/sqrt(1-W^2) */
   d( W, X, Z).                     /* donde Z = d(W)/dX               */

d( arccos(W), X, -(Z/sqrt(1-W^2)) ):-   /* d(arccos(W))/dX = -(Z/sqrt(1-W^2)) */
   d( W, X, Z).                         /* donde Z = d(W)/dX                  */

d( arctan(W), X, Z/(1+W^2) ):-     /* d(arctan(W))/dX = Z/(1+W^2) */
   d( W, X, Z).                    /* donde Z = d(W)/dX           */

d( arccot(W), X, -(Z/(1+W^2)) ):- /* d(arccot(W))/dX = -(Z/(1+W^2))  */
   d( W, X, Z).                   /* donde Z = d(W)/dX               */
```

```
31 d( arcsec(W), X, (Z/(W*sqrt(W^2-1))) ):-  /* d(arcsec(W))/dX = (Z/(W*sqrt(W^2-1))) */
32     d( W, X, Z).                          /* donde Z = d(W)/dX                    */
33
34 d( arccosec(W), X, -(Z/(W*sqrt(W^2-1))) ):-  /* d(arccosec(W))/dX = -(Z/(W*sqrt(W^2-1)))
       */
35     d( W, X, Z).                           /* donde Z = d(W)/dX
                                 */
```

Derivada logarítmica y exponencial

```
1 d( exp(W), X, Z*exp(W) ):- /* d(exp(W))/dX = Z*exp(W)    */
2     d( W, X, Z).           /* donde Z = d(W)/dX          */
3
4 d( log(W), X, Z/W ):-      /* d(log(W))/dX = Z/W */
5     d( W, X, Z).           /* donde Z = d(W)/dX  */
```

Una vez que se declara la base de conocimientos, se carga el archivo Prolog en el intérprete y se realizan consultas. A continuación se realizan algunas consultas a la base y muestra la salida de la consola de Prolog.

Consultas

```
1  % c:/Users/Uri/Dropbox/TT (ED)/axiomas.pl compiled 0.00 sec
       ,
2  22 clauses
3
4  1 ?- d(5, x, R). /* Derivada de 5/dX */
5
6  R = 0.
7
8  2 ?- d(x, x, R). /* Derivada de x/dX */
9
10 R = 1.
11
12 3 ?- d(x+1, x, R). /* Derivada de x + 1/dX */
13
14 R = 1+0.
15
16 4 ?- d(x*sin(x), x, R). ./* Derivada de x*sin(x)/dX */
17
18 R = x* (1*cos(x))+sin(x)*1.
19
20 5 ?- d(x^3, x, R). */Derivada de x^3/dX */
21
22 R = 3*1*x^ (3-1)
23
24 6 ?- d(sin(x)/cos(x), x, R). */Derivada de (sin(x)/cos(x))/
       dX */
25 R = (1*cos(x)*cos(x)- - (1*sin(x))*sin(x))/cos(x)^2.
```

2.4. Sistemas de Álgebra Computacional desarrollados en la ESCOM

A continuación se mencionan los sistemas de álgebra computacional desarrollados como Trabajos Terminales de la Escuela Superior de Cómputo.

Año	Título	Descripción breve
1999	Biblioteca de Ecuaciones Diferenciales	Este sistema resuelve Ecuaciones Diferenciales Ordinarias por algoritmos de la familia Runge-Kutta.
2005	Prototipo de un Sistema Experto para Ecuaciones Diferenciales	Sistema experto de ecuaciones diferenciales, el cual se aplica el estudio del nuevo modelo educativo.
2010	Sistema para el fortalecimiento del aprendizaje de la ecuaciones dife-renciales totales empleando transformada de Laplace	Resolución de Ecuaciones Diferenciales Ordinarias por el método de la Transformada de Laplace.
2010	La Transformación de Darboux para la ecuación de Burgers no homogénea en (1+1) dimensiones. Una aplicación compu-tacional en la ingeniería.	Solo se enfoca en el estudio específico de la ecuación de Burgers no homogénea por el método de la Transformación de Darboux

Como se puede observar, los anteriores trabajos terminales que resuelven ecuaciones diferenciales no lo hacen forma generalizada, y la mayoría lo hacen para ecuaciones diferenciales ordinarias.

2.5. Deliberación

Con base en la investigación antes realizada, se tomó la decisión de utilizar Prolog para el desarrollo de nuestro Trabajo Terminal por diversas razones:

- Prolog es un lenguaje de programación lógica, donde su propósito es generar un sistema experto.
- Para la resolución de la Ecuación Diferencial de Segundo Orden Lineal No Homogénea con Coeficientes No Constantes, no se busca un mé-

todo numérico que genere una solución aproximada, sino un método simbólico que genere una solución analítica.

- El propósito del Trabajo Terminal, además de resolver el problema planteado, es demostrar que se tienen los conocimientos requeridos para obtener el grado de Ingeniero en Sistemas Computacionales, por lo que no se pretende colaborar al trabajo de alguien más, sino crear nuestro propio trabajo.

Capítulo 3

Análisis y Diseño del sistema

3.1. Introducción

En este capítulo se presentan las fases de Análisis y Diseño del sistema de acuerdo con la metodología elegida para el presente Trabajo Terminal .

3.2. Metodología del sistema

El sistema se desarrollará bajo la metodología de **Desarrollo en Cascada**, la cual indica que el inicio de cada etapa deberá esperar la finalización de la etapa anterior. De forma general, las etapas de la metodología en orden secuencial son las siguientes:

1. Análisis de requerimientos.
2. Diseño del sistema.
3. Implementación.
4. Pruebas.
5. Mantenimiento.

A continuación se presenta la primer etapa de la metodología.

3.3. Análisis de requerimientos

3.3.1. Objetivo

Desarrollar un prototipo computacional que de solución a ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes.

3.3.2. Alcances

3.3.2.1. Notación

El sistema deberá entender la notación Leibniz mediante un analizador léxico-sintáctico.

La **Notación de Leibniz** se representa de la siguiente manera:
Dada una función f de x:

$$y = y(x)$$

mediante el operador derivada de la función:

$$y' = \frac{d}{dx}y(x)$$

se representaría en notación de Leibniz de esta forma:

$$y' = \frac{dy}{dx}$$

3.3.2.2. Resultados

El sistema será capaz de resolver derivadas e integrales de manera interna con el objetivo de dar solución a ecuaciones diferenciales lineales no homogéneas, de segundo orden con coeficientes no constantes. Si la ecuación diferencial no tiene solución o la ecuación diferencial está fuera del alcance del sistema, se mostrará un mensaje de error.

3.3.3. Limitaciones

El sistema no podrá resolver ecuaciones diferenciales no lineales, homogéneas, ni de orden superior.

3.4. Diseño del Sistema

3.4.1. Diseño Arquitectónico

3.4.1.1. Módulos

El desarrollo del sistema comprenderá los siguientes módulos:

Módulo	Nombre
1	Entrada y salida de datos (interfaz).
2	Analizador Léxico-Sintáctico.
3	Base de conocimiento en Prolog.
4	Algoritmo para resolver ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes.
5	Algoritmo para simplificar expresiones matemáticas.

3.4.1.2. Requerimientos Funcionales

Los requerimientos funcionales identificados son los siguientes:

Clave	Descripción
RF01	El sistema será capaz resolver ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes.
RF02	El sistema podrá representar las ecuaciones diferenciales escritas por el usuario en notación Leibniz.
RF03	El sistema deberá contar con una interfaz de usuario.
RF04	El sistema deberá mostrar el resultado en la interfaz.

3.4.1.3. Requerimientos No Funcionales

Los requerimientos no funcionales identificados son los siguientes:

Clave	Descripción
RNF01	El sistema mostrará una previsualización de la expresión matemática escrita por el usuario.
RNF02	El sistema validará la completez y correcta sintaxis de la expresión escrita por el usuario mediante el Analizador Léxico-Sintáctico.
RNF03	La interfaz de usuario del sistema será sencilla y atractiva.

3.4.1.4. Diagramas y Casos de Uso

3.4.1.4.1. Diagrama de Caso de Uso A continuación se muestra el caso uso del sistema.

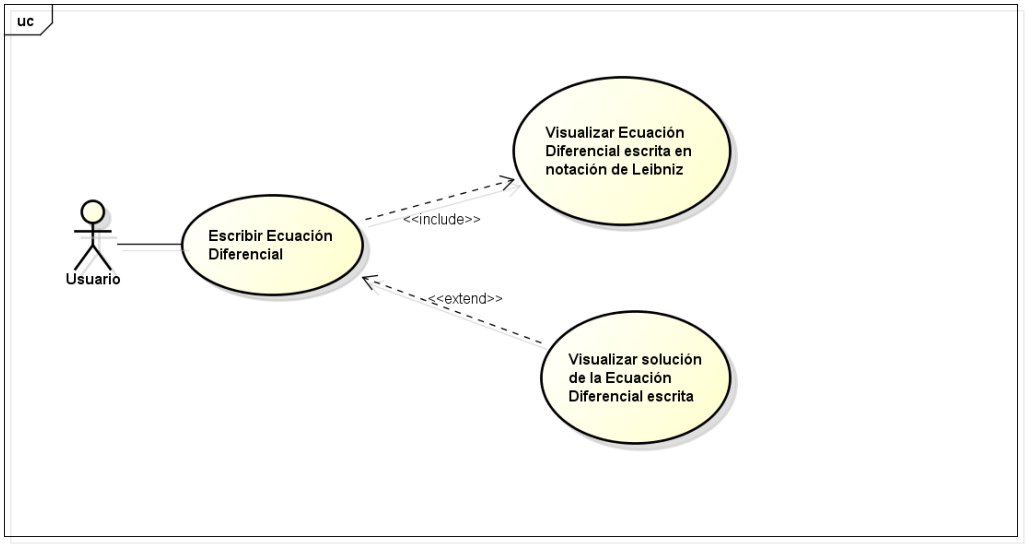

Figura 3.1: Casos de Uso

3.4.1.4.2. Caso de Uso: Escribir Ecuación Diferencial

Caso de uso: Escribir Ecuación Diferencial	
Autor	Usuario final
Entradas	Ecuación Diferencial
Salida	Ninguna
Precondición	El usuario accede a la interfaz web
Tipo	Primario

Trayectorias de Caso de Uso

Trayectoria Principal

1. El usuario teclea la ecuación diferencial ya sea desde el teclado u oprimiendo los botones de la interfaz del sistema. **Trayectoria A**

2. El usuario pulsa el botón "Solución". **Trayectoria B**

3. Fin de la trayectoria.

Trayectoria alternativa A Visualizar Ecuación Diferencial escrita en notación Leibniz

A1. El sistema ejecuta el **Caso de Uso Visualizar Ecuación Diferencial escrita, en notación Leibniz**

A2. Fin de la trayectoria

Trayectoria alternativa B Visualizar Solución de Ecuación Diferencial escrita

B1. El sistema ejecuta el **Caso de Uso Visualizar Solución de Ecuación Diferencial escrita**.

B2. Fin de la trayectoria

3.4.1.4.3. Caso de Uso: Visualizar Ecuación Diferencial escrita en notación Leibniz.

Caso de Uso: Visualizar Ecuación Diferencial escrita en notación Leibniz.	
Autor	Usuario final
Entradas	Datos ingresados
Salida	Ecuación Diferencial en notación Leibniz
Precondición	El usuario ha escrito una Ecuación Diferencial
Tipo	Primario

Trayectorias de Caso de Uso

Trayectoria Principal

1. El sistema recibe la cadena ingresada por el usuario.

2. El sistema analiza la cadena por medio del ALS (Analizador Léxico Sintáctico). **Trayectoria A**

3. El sistema muestra en pantalla la Ecuación Diferencial en notación Leibniz.

4. Fin de la trayectoria.

Trayectoria alternativa A Error de Sintaxis

A1. El sistema muestra en pantalla el mensaje: "Sintaxis incorrecta".

A2. Fin de la trayectoria.

3.4.1.4.4. Caso de Uso: Visualizar Solución de Ecuación Diferencial escrita.

Caso de Uso: Visualizar Solución de Ecuación Diferencial escrita.	
Autor	Usuario final
Entradas	Datos ingresados
Salida	Solución de Ecuación Diferencial
Precondición	El usuario ha escrito una Ecuación Diferencial
Tipo	Primario

Trayectorias de Caso de Uso

Trayectoria Principal

1. 1. El sistema recibe la cadena ingresada por el usuario.

2. 2. El sistema analiza la cadena por medio del ALS (Analizador Léxico Sintáctico). **Trayectoria A**

3. 3. El sistema genera un conjunto de expresiones Prolog.

4. 4. El sistema da resultado a cada una de las expresiones Prolog. **Trayectoria B, Trayectoria C**

5. 5. El sistema simplifica la solución.

6. 6. El sistema muestra en pantalla la solución de la Ecuación Diferencial.

7. 7. Fin de la trayectoria.

Trayectoria alternativa A Error de Sintaxis

A1. El sistema muestra en pantalla el mensaje: "Sintaxis incorrecta".

A2. Fin de la trayectoria.

Trayectoria alternativa B No tiene solución

B1. El sistema muestra en pantalla el mensaje: "Sin solución".

B2. Fin de la trayectoria.

Trayectoria alternativa C Solución parcial

C1. El sistema muestra la solución parcial.

C2. Fin de la trayectoria.

3.4.1.4.5. Diagrama de Actividades A continuación se muestra el diagrama de Actividades del sistema

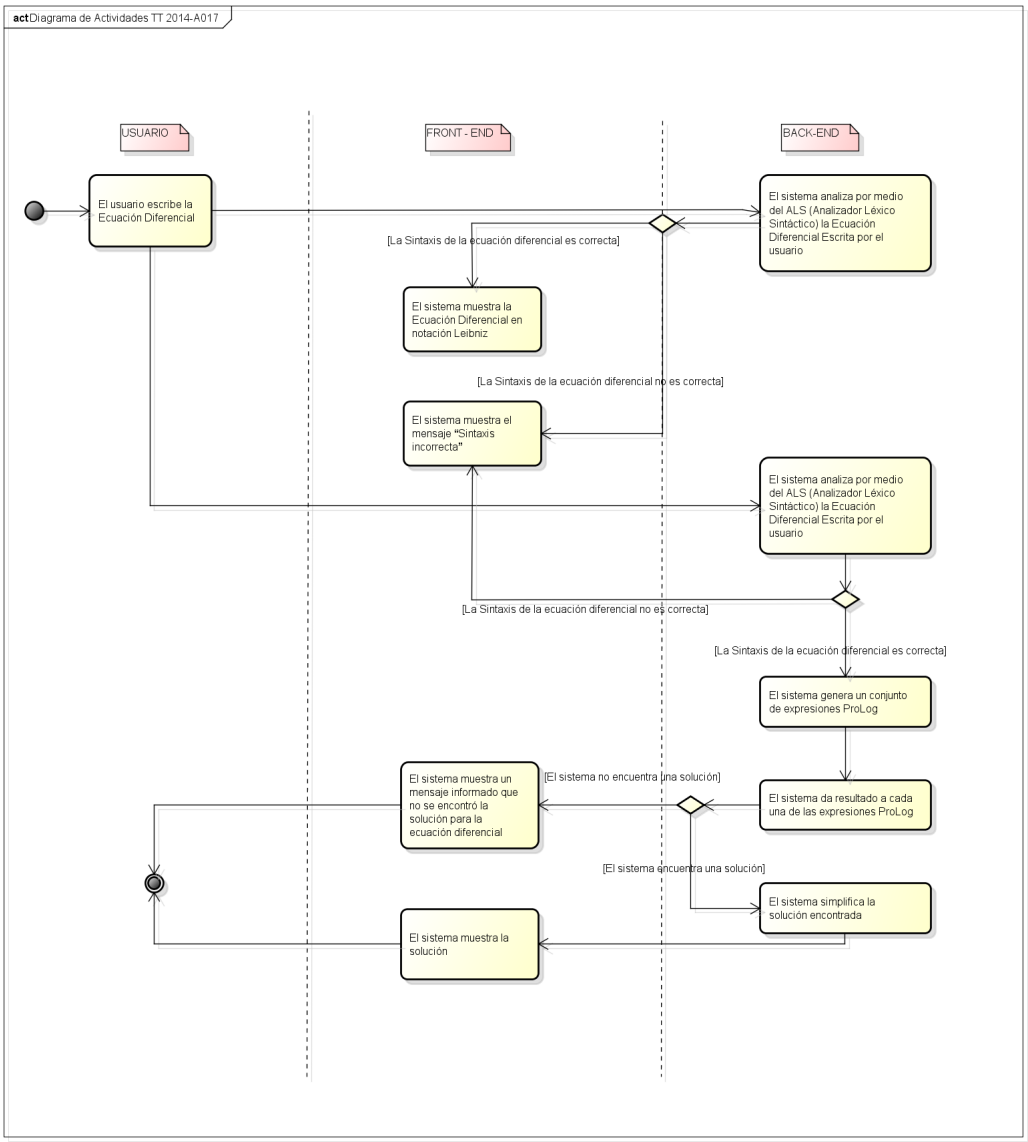

Figura 3.2: Diagrama de Actividades

3.4.1.4.6. Diagrama de Secuencia A continuación se muestran los diagramas de Secuencia del sistema.

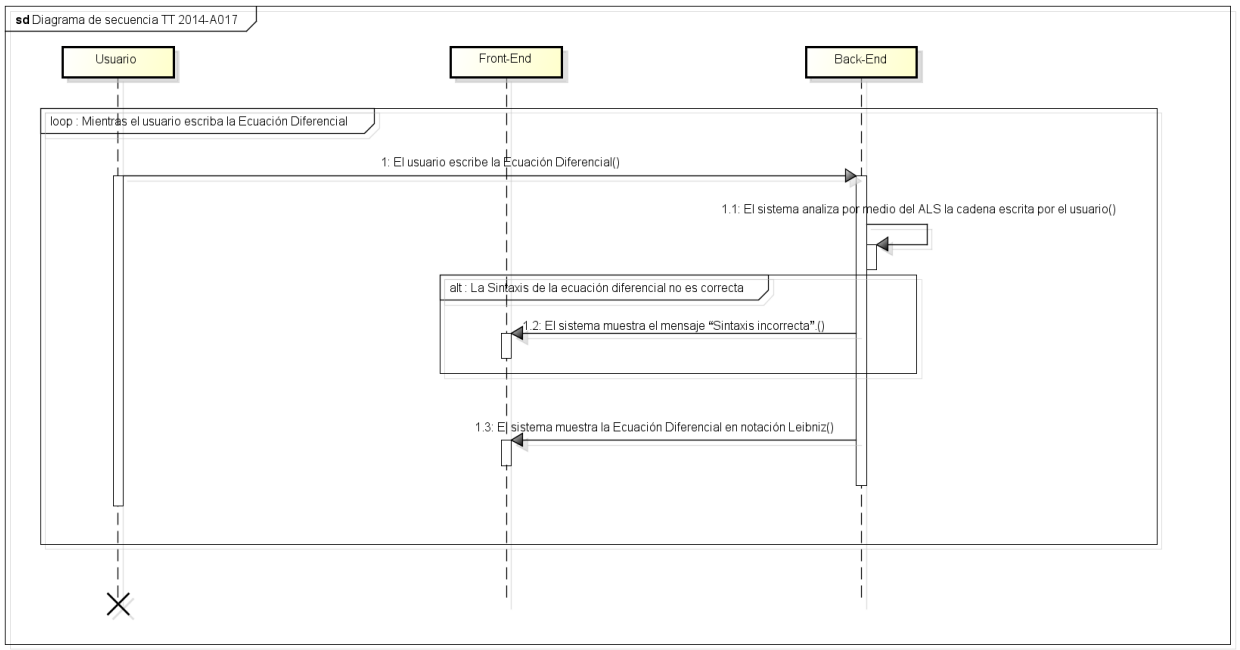

Figura 3.3: Diagrama de Secuencia 1 de 2

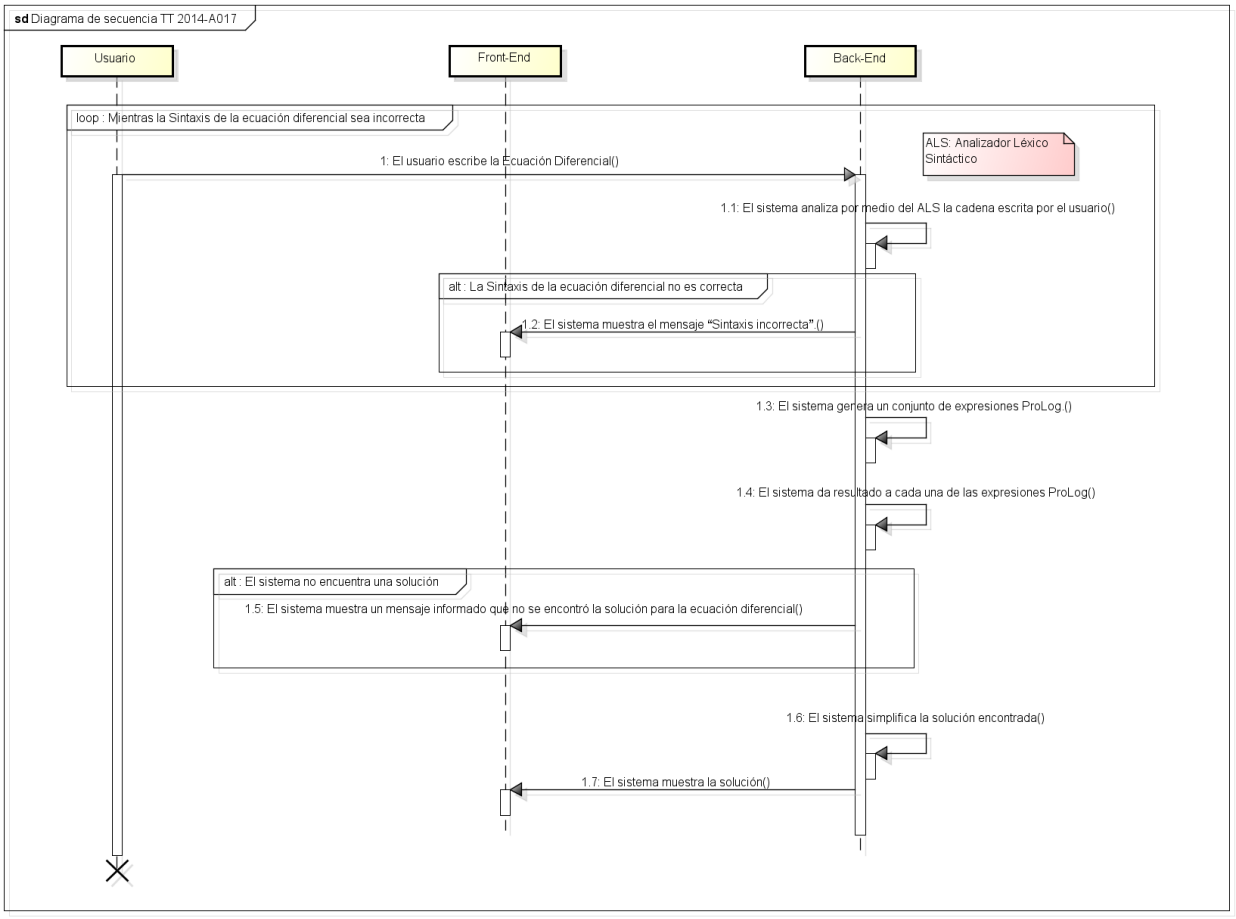

Figura 3.4: Diagrama de Secuencia 2 de 2

3.4.1.4.7. Arquitectura del sistema (hardware) A continuación se muestra el diagrama de Arquitectura del sistema enfocada al hardware.

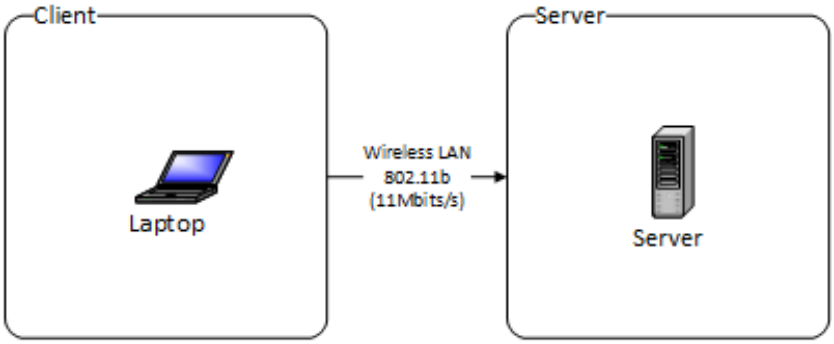

Figura 3.5: Arquitectura del sistema (hardware)

3.4.1.4.8. Arquitectura del sistema (software) A continuación se muestra el diagrama de Arquitectura del sistema enfocada al software.

De acuerdo al diseño original del sistema, el diagrama presenta la siguiente estructura:

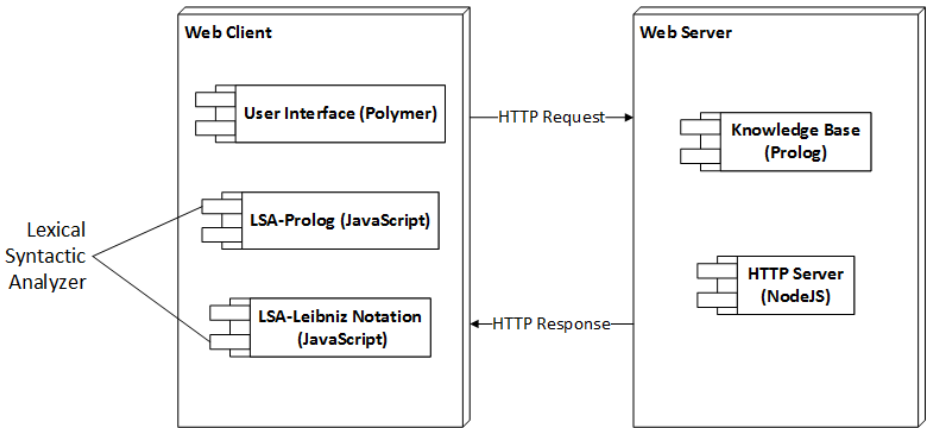

Figura 3.6: Arquitectura del sistema original (software)

Sin embargo, acorde a las herramientas manejadas y resultados obtenidos, la estructura de la arquitectura del sistema enfocada al software es la

siguiente:

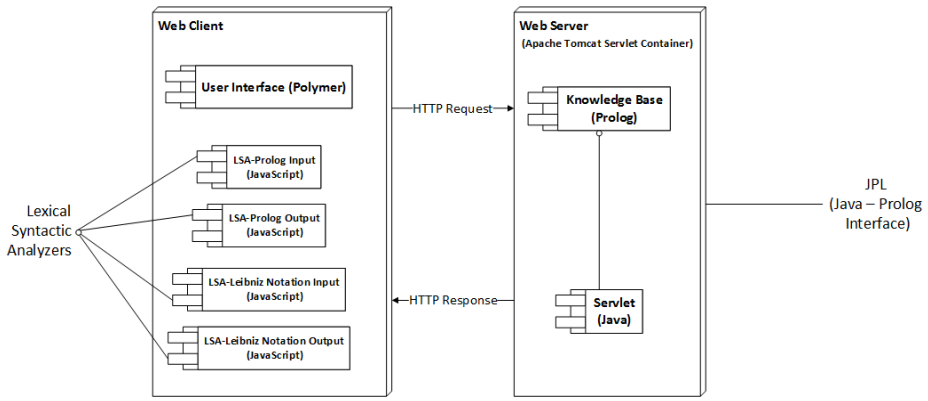

Figura 3.7: Arquitectura del sistema final (software)

3.4.2. Diseño de Interfaz Gráfica

3.4.2.1. Introducción

En esta sección se muestran los diseños correspondientes a la interfaz de usuario que presentará el sistema.

La interfaz gráfica está compuesta de los siguientes Web Components:

- **paper-calculator-output** Componente donde se visualiza el resultado de la consulta ingresada por el usuario.
- **paper-calculator-mathjax** Componente donde se visualiza la consulta ingresada por el usuario en notación Leibniz.
- **paper-calculator-keypad** Componente 'tecladoén el que el usuario ingresa su consulta. Este componente a su vez está formado por los siguientes componentes:
  - **paper-calculator-panels** Componente que agrupa los diferentes teclados en páneles colapsables.
  - **paper-calculator-keygrid** Componente que crea los diferentes teclados usando un arreglo de teclas. Está formado por instancias del siguiente componente:
    - **paper-calculator-key** El componente más básico: una tecla. Aquí se define el estilo y comportamiento de todas las

teclas y se crean los diferentes manejadores de eventos para las mismas.

- **paper-calculator-math** Componente que resuelve operaciones básicas a modo de calculadora.

3.4.2.2. Vista de escritorio

En esta sección se expone la interfaz gráfica visualizada en un navegador de escritorio.

Figura 3.8: Pantalla principal (escritorio)

3.4.2.3. Vista para dispositivos móviles

En esta sección se expone la interfaz gráfica adaptada a un navegador de dispotitivo móvil (orientado de manera horizontal) a través del diseño responsivo. La principal característica de esta vista es que dependiendo del tamaño del dispositivo, se mostrarán los páneles de la interfaz colapsados o no.

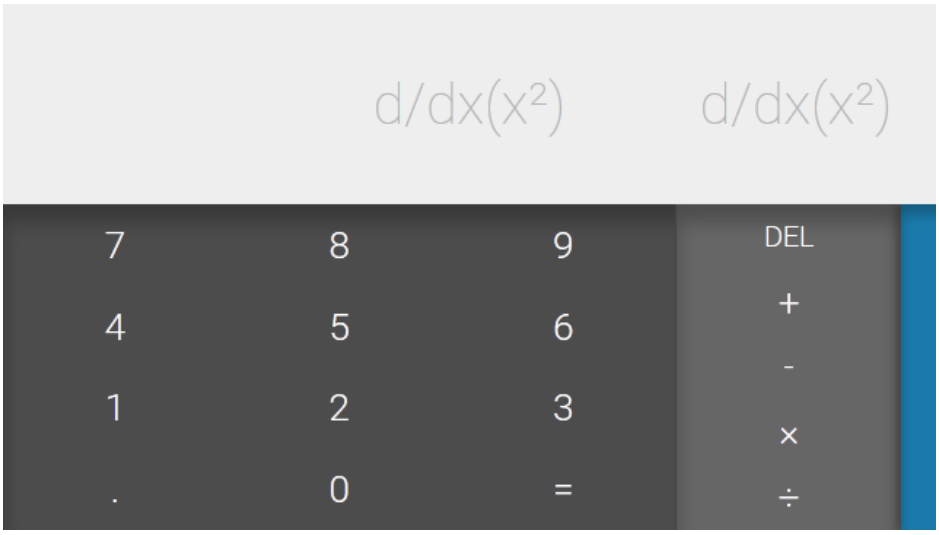

Figura 3.9: Panel numérico y operaciones básicas (móvil)

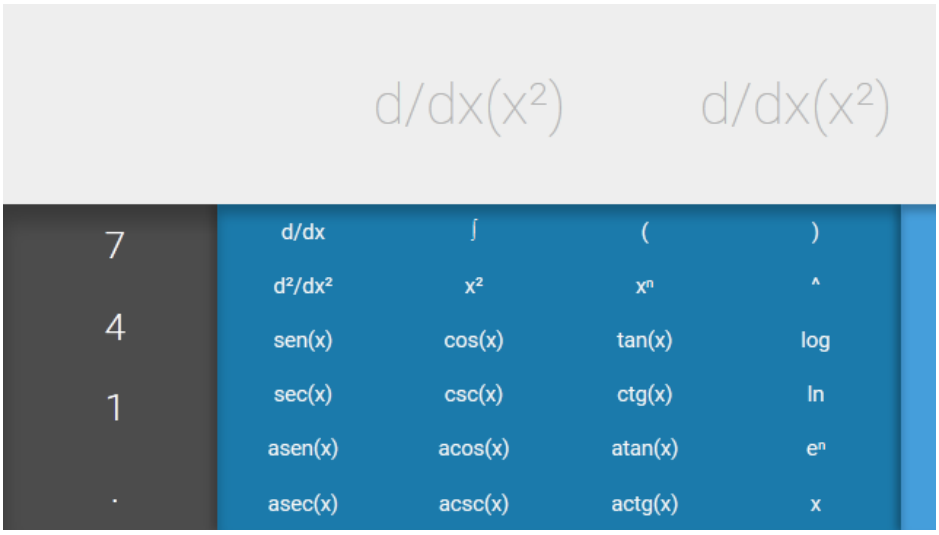

Figura 3.10: Panel de funciones (móvil)

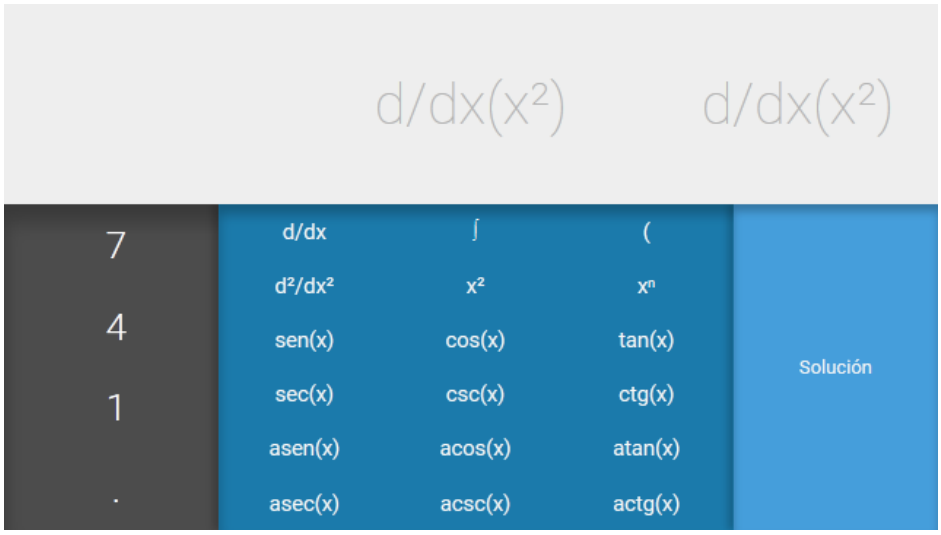

Figura 3.11: Panel de funciones y solución (móvil)

3.4.2.4. Flujo

El usuario ingresará su consulta a través de los teclados en los diferentes páneles (versión escritorio y móvil) o a través del teclado de su computadora (versión escritorio).

Figura 3.12: Ingresar consulta (error de sintaxis)

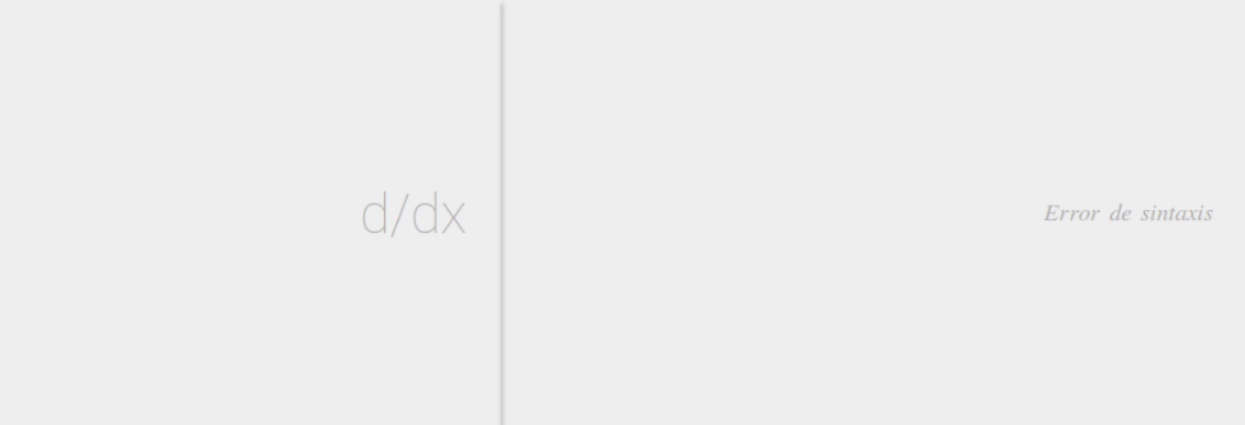

Figura 3.13: Ingresar consulta (error de sintaxis) - Detalle

Si la consulta ingresada por el usuario no está completa o tiene errores de sintaxis, el panel de previsualización mostrará el mensaje de *error de sintaxis*.

Si la consulta no tiene errores de sintaxis, se mostrará la previsualización de la siguiente forma:

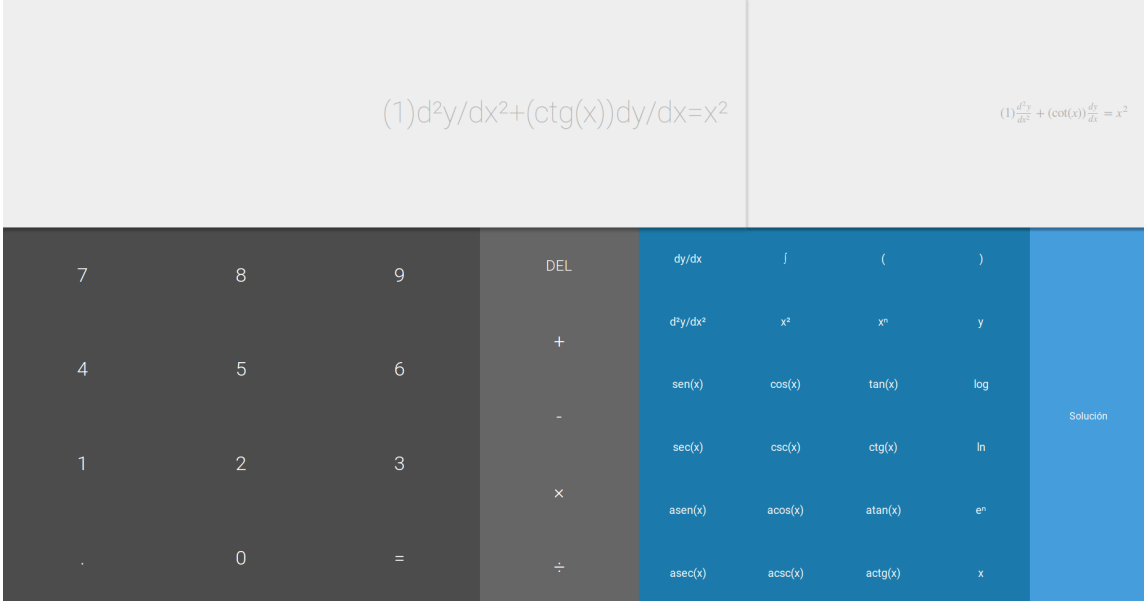

Figura 3.14: Previsualización de la consulta

Figura 3.15: Previsualización de la consulta - Detalle

Una vez ingresada la consulta, el usuario deberá presionar el botón de **Solución**.

Si la consulta fue válida (su estructura fue procesada correctamente por los ALS) se visualizará el resultado en el panel de salida.

Figura 3.16: Visualización de resultado

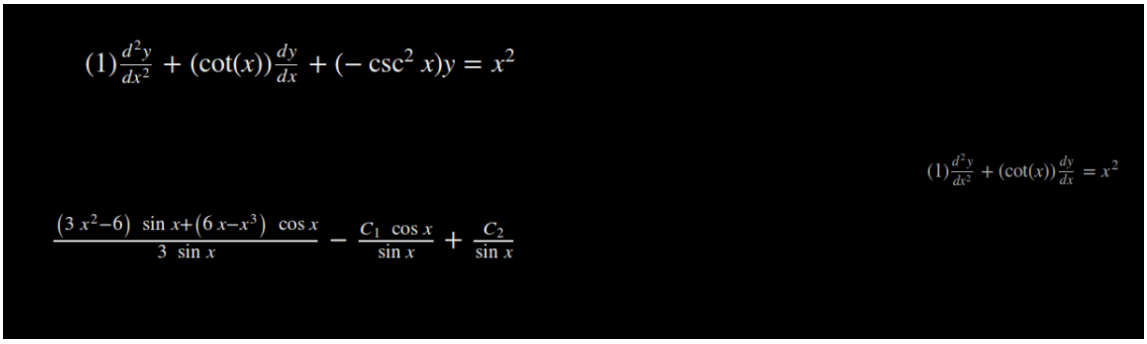

Figura 3.17: Visualización de resultado - Detalle

Si el sistema no puede generar una solución a la consulta ingresada por el usuario, se dejarán las integrales sin resolver expresadas en la solución de acuerdo a la teoría.

Figura 3.18: Consulta sin resultado

Figura 3.19: Consulta sin resultado - Detalle

Si la consulta fue inválida (su estructura tuvo errores) se visualizará un mensaje de error en el panel de salida.

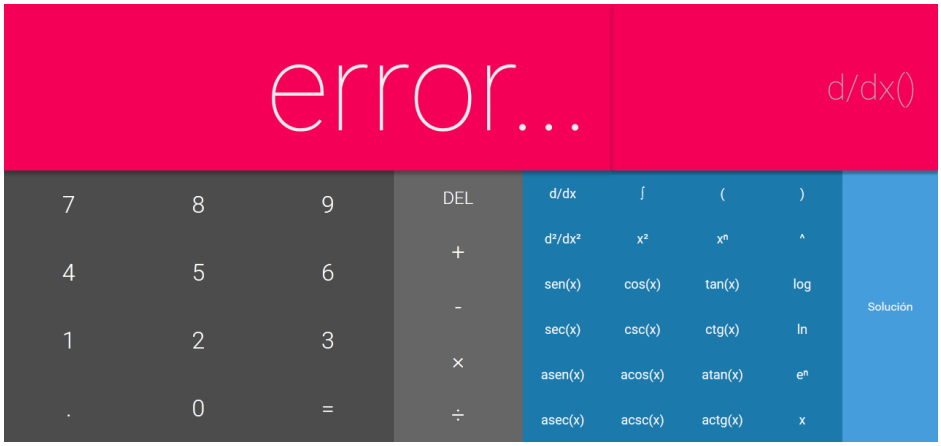

Figura 3.20: Visualización de error (escritorio)

3.4.3. Diseño de la Base de Conocimientos

3.4.3.1. Introducción

En esta sección expondremos los detalles sobre la base de conocimientos. La base de conocimientos está escrita en lenguaje Prolog y se encuentra contenida en un solo script. Consta de cuatro grandes grupos de predicados lógicos, los cuales son:

1. Cálculo Diferencial.
2. Cálculo Integral.
3. Ecuaciones Diferenciales.
4. Fórmula de Solución.

3.4.3.2. Cálculo Diferencial

Contiene los predicados para resolver derivadas de constantes, sumas, restas, productos, cocientes, potencias, logaritmos, funciones exponenciales, trigonométricas, hiperbólicas e hiperbólicas inversas.

3.4.3.3. Cálculo Integral

Grupo con los predicados que dan solución a integrales de constantes, sumas, restas, productos, potencias, potencia de la suma de x y una constante, potencia de la suma de un producto y una constante, potencia de la resta

de x y una constante, potencia de la resta de un producto y una constante, funciones trigonométricas y funciones exponenciales.

3.4.3.4. Ecuaciones Diferenciales

Este grupo contiene predicados que pueden resolver ecuaciones diferenciales por variables separables y ecuaciones diferenciales lineales de segundo orden con coeficientes constantes.

3.4.3.5. Fórmula de Solución

Contiene el predicado de solución a ecuaciones diferenciales lineales no homogéneas de segundo orden con coeficientes no constantes.

3.4.3.6. Comunicación entre Interfaz Gráfica y Base de Conocimientos

La comunicación entre la interfaz gráfica y la base de conocimientos se realizará utilizando el protocolo HTTP a través de la realización de peticiones por medio de la interfaz gráfica (cliente, en arquitectura cliente-servidor). Estas peticiones serán escuchadas por el servidor, el cual cargará la base de conocimientos y procesará la petición, para después mandarle una respuesta al cliente con el resultado.

3.4.4. Diseño de los Anaizadores Léxicos-Sintácticos

3.4.4.1. Introducción

En esta sección expondremos los analizadores léxicos-sintácticos a manejar por el sistema.

Los analizadores léxicos-sintácticos estarán encargados de procesar la consulta ingresada por el usuario para darle la forma requerida por la base de conocimientos para generar un resultado. También producirán la misma consulta de entrada pero representada en notación Leibniz. Por último, tomarán el resultado generado por la base de conocimientos y lo expresarán en notación infija y notación Leibniz.

3.4.4.2. Analizador Léxico-Sintáctico para Expresiones Prolog

Este ALS recibe la consulta ingresada por el usuario a través de la intefaz gráfica y, si esta tuvo una estructura correcta, nos devuelve la misma

consulta pero con la forma de un predicado Prolog.

Por ejemplo, la siguiente consulta:

```
d/dx((x^2+3x))
```

Es analizada por el ALS y nos regresa la siguiente expresión:

```
d((x^2+3*x), x, R).
```

La gramática del ALS para expresiones Prolog es la siguiente:

```
{
  function cadenaMultiplicativa(arreglo){
    var res="";
    for(i=0; i < arreglo.length; i++){
      if(i == arreglo.length -1){
        res += arreglo[i];
      }else{
        if(isNaN(arreglo[i])){
           res += arreglo[i] + "*";
        }else{
           res += arreglo[i];
        }
      }
    }
    return res;
  }
  function suma_(p1, p2){
    if(isNaN(p1)){
      return p1 + "+" + p2;
    }
    if(isNaN(p2)){
      return p1 + "+" + p2;
    }
    return parseInt(p1) + parseInt(p2);
  }
  function resta_(p1, p2){
    if(isNaN(p1)){
      return p1 + "-" + p2;
    }
    if(isNaN(p2)){
      return p1 + "-" + p2;
    }
    return parseInt(p1) - parseInt(p2);
  }
  function multiplicacion_(p1, p2){
    if(isNaN(p1)){
      return p1 + "*" + p2;
    }
    if(isNaN(p2)){
      return p1 + "*" + p2;
    }
    return parseInt(p1) * parseInt(p2);
  }
  function division_(p1, p2){
    if(p1 == p2){
      return "1";
    }
    if(isNaN(p1)){
      return p1 + "/" + p2;
    }
```

```
51      if(isNaN(p2)){
52        return p1 + "/" + p2;
53      }
54      return parseInt(p1) / parseInt(p2);
55    }
56     function divisionUno(num, dem){
57        if(num == dem){
58           return "1";
59        }else if(num == 0){
60           return "0";
61        }else if(dem.indexOf('*') != -1){
62           var aux = dem.split("*");
63           var res = [];
64           var bandera = false;
65           for(i=0; i<aux.length; i++){
66              if(aux[i] == num){
67                 num = "";
68                 bandera = true;
69              }else{
70                 if(i == aux.length -1){
71                    res.push(aux[i]);
72                 }else{
73                    res.push(aux[i]);
74                    res.push("*");
75                 }
76
77              }
78           }
79           if(bandera)
80              return res.join("");
81           else
82              return num + "/" + dem;
83        }else{
84           return num + "/" + dem;
85        }
86     }
87  }
88  start
89    = additive
90
91  additive
92    = left:multiplicative "+" right:additive { return suma_(left,right); }
93    / resta
94
95  resta
96    = left:multiplicative "-" right:resta { return resta_(left,right); }
97    / multiplicative
98
99  multiplicative
100   = left:primary "*" right:multiplicative { return multiplicacion_(left, right); }
101   / division
102
103 division
104   = left:primary "/" right:division { return division_(left,right); }
105   / primary
106
107 primary
108   = exponentes
109   / "d/dx(" primera:additive ")" signo:[+-]+ "d\u00B2/dx\u00B2(" segunda:additive ")="
        funcion:additive
110     { return "d(" + primera + ",x,R);" + signo + "d(" + segunda + ",x,R);" + "int(" +
          divisionUno(primera,segunda) + ",x,R);" + "int(" + divisionUno(funcion,segunda) +
           ",x,R);"; }
111   / "d\u00B2/dx\u00B2(" segunda:additive ")" signo:[+-]+ "d/dx(" primera:additive ")="
        funcion:additive
112     { return "d(" + primera + ",x,R);" + signo + "d(" + segunda + ",x,R);" + "int(" +
          divisionUno(primera,segunda) + ",x,R);" + "int(" + divisionUno(funcion,segunda) +
           ",x,R);"; }
113   / "d/dx(" additive:additive ")=" funcion:additive { return "d(" + additive + ",x,R);"
        + funcion; }
```

```
  / "d\u00B2/dx\u00B2(" additive:additive ")=" funcion:additive { return "d(" + additive
      + ",x,R);" + funcion; }
  / "(" additive:additive ")" { return additive; }

exponentes
  = left:combinado right:"\u00B2" {return left + "^" + 2;}
  / left:combinado right:"\u00B3" {return left + "^" + 3;}
  / left:combinado right:"\u207F" grado:integer {return left + "^" + grado;}
  / left:exponencial right:"\u207F" grado:combinado {return left + "^" + grado;}
  / combinado

combinado
  = left:variables right:combinado {return multiplicacion_(left,right);}
  / variables

variables "variables"
  = vars:[wxyz]i+ { return cadenaMultiplicativa(vars); }
  / integer

exponencial "exponencial"
  = exp:[e] { return "e"; }

integer "integer"
  = digits:[0-9]+ { return cadenaMultiplicativa(digits); }
```

3.4.4.3. Analizador Léxico-Sintáctico para Notación Leibniz

Este ALS recibe la consulta ingresada por el usuario a través de la intefaz gráfica y, si esta tuvo una estructura correcta, nos devuelve la misma consulta pero ahora escrita con sintaxis LaTeX. Las derivadas que se encuentren en la ecuación ingresada en la consulta estarán escritas en notación Leibniz.

Por ejemplo, la siguiente consulta:

```
d^2/dx^2((x^2+3x))
```

Es analizada por el ALS y nos regresa la siguiente expresión:

```
\frac{d^2}{dx^2}(x^2+3x)
```

La gramática del ALS para la notación Leibniz es la siguiente:

```
{
  function cadenaMultiplicativa(arreglo){
    var res="";
    for(i=0; i < arreglo.length; i++){
      if(i == arreglo.length -1){
        res += arreglo[i];
      }else{
         res += arreglo[i];
        }
      }
    return res;
  }
  function suma_(p1, p2){
    return p1 + "+" + p2;
  }
```

```
  function resta_(p1, p2){
    return p1 + "-" + p2;
  }
  function multiplicacion_(p1, p2){
    if(isNaN(p1)){
      return p1 + "*" + p2;
    }
    if(isNaN(p2)){
      return p1 + p2;
    }
    return p1 + "*" + p2;
  }
  function division_(p1, p2){

    return "\\frac{" + p1 + "}{" + p2 + "}";
  }
  function mathJaxRender(e){
    return "$"+e+"$";
  }
}
start
  = additive

additive
  = left:multiplicative "+" right:additive { return suma_(left,right); }
  / resta

resta
  = left:multiplicative "-" right:resta { return resta_(left,right); }
  / multiplicative

multiplicative
  = left:primary "*" right:multiplicative { return multiplicacion_(left, right); }
  / division

division
  = left:primary "/" right:division { return division_(left,right); }
  / primary

primary
  = exponentes
  / "d/dx(" additive:additive ")" { return "\\frac{d}{dx}(" + additive + ")"; }
  / "d\u00B2/dx\u00B2(" additive:additive ")" { return "\\frac{d^2}{dx^2}(" + additive +
      ")"; }
  / "(" additive:additive ")" { return additive; }
  / "\u222B(" additive:additive ")" { return "\\\\int(" + additive + ")dx"; }

exponentes
  = left:combinado right:"\u00B2" {return left + "^" + 2;}
  / left:combinado right:"\u00B3" {return left + "^" + 3;}
  / left:combinado right:"\u207F" grado:integer {return left + "^" + grado;}
  / left:exponencial right:"\u207F" grado:combinado {return left + "^" + grado;}
  / combinado

combinado
  = left:variables right:combinado {return multiplicacion_(left,right);}
  / variables

variables "variables"
  = vars:[wxyz]i+ { return cadenaMultiplicativa(vars); }
  / integer

exponencial "exponencial"
  = exp:[e] { return "e"; }

integer "integer"
  = digits:[0-9]+ { return cadenaMultiplicativa(digits); }
```

3.4.4.4. Analizador Léxico-Sintáctico para Salida de Expresiones Prolog

Este ALS recibe el resultado que arroja la base de conocimientos después de que se realiza la consulta del usuario a través de la intefaz gráfica.

Por ejemplo, la siguiente entrada:

```
R=*(*(3,x),+(3,x))
```

Es analizada por el ALS y nos regresa la siguiente expresión:

```
(3*x)(3+x)
```

La gramática del ALS para la notación Leibniz es la siguiente:

```
{
   function suma_(p1, p2){
      if(p1 == "0"){
         return p2;
      }
      if(p2 == "0"){
         return p2;
      }
      if(isNaN(p1)){
         return p1 + "+" + p2;
      } if(isNaN(p2)){
         return p1 + "+" + p2;
      }
      return parseInt(p1) + parseInt(p2);
   }

   function resta_(p1, p2){
      if(isNaN(p1)){
         return p1 + "-" + p2;
      } if(isNaN(p2)){
         return p1 + "-" + p2;
      }
      return parseInt(p1) - parseInt(p2);
   }

   function multiplicacion_(p1, p2){
      if(p1 == "0" || p2 == "0"){
         return "0";
      }
      if(p1 == "1"){
         return p2;
      }
      if(p2 == "1"){
         return p1;
      }
      if(isNaN(p1)){
         return p1 + "*" + p2;
      } if(isNaN(p2)){
         return p1 + "*" + p2;
      }
      return parseInt(p1) * parseInt(p2);
   }

   function division_(p1, p2){
```

```
45          if(p1 == "0"){
46              return "0";
47          }
48          if(p1 == p2){
49              return "1";
50          }
51          if(isNaN(p1)){
52              return p1 + "/" + p2;
53          } if(isNaN(p2)){
54              return p1 + "/" + p2;
55          }
56          return parseInt(p1) / parseInt(p2);
57      }
58  }
59  start
60      = suma
61  suma
62      = "R=+(" sumandoA:exponencial "," sumandoB:suma ")" {return suma_(sumandoA,sumandoB)
            ;}
63      / "+(" sumandoA:exponencial "," sumandoB:suma ")" {return suma_(sumandoA,sumandoB);}
64      / resta
65  resta
66      = "R=-(" sumandoA:exponencial "," sumandoB:resta ")" {return resta_(sumandoA,sumandoB
            );}
67      / "-(" sumandoA:exponencial "," sumandoB:resta ")" {return resta_(sumandoA,sumandoB)
            ;}
68      / multiplicacion
69
70  multiplicacion
71      = "R=*(" sumandoA:exponencial "," sumandoB:multiplicacion ")" {return multiplicacion_
            (sumandoA,sumandoB);}
72      / "*(" sumandoA:exponencial "," sumandoB:multiplicacion ")" {return multiplicacion_(
            sumandoA,sumandoB);}
73      / division
74
75  division
76      = "R=/(" sumandoA:exponencial "," sumandoB:division ")" {return division_(sumandoA,
            sumandoB);}
77      / "/(" sumandoA:exponencial "," sumandoB:division ")" {return division_(sumandoA,
            sumandoB);}
78      / sencillo
79
80  sencillo
81      = "R=" atom:exponencial {return atom;}
82      / exponencial
83
84  exponencial
85      = "exp(" atom:start ")" {return "e^(" + atom + ")"; }
86      / atom
87
88  atom
89      = cadena:[0-9xwxyz]+ {return cadena.join("");}
```

3.4.5. Diseño Detallado

3.4.5.1. Introducción

En esta sección detallaremos los patrones de diseño que se implementarán en el sistema, así como también las convenciones que usaremos para la organización y generación de nuestro código.

3.4.5.2. Patrones de Diseño

3.4.5.2.1. Modelo Vista Controlador (MVC) El patrón de diseño base para nuestro sistema será el patrón arquitectónico **Modelo Vista Controlador**, el cual divide la aplicación de software en tres partes interconectadas entre sí, con objeto de separar la representación interna de la información de la capa de presentación vista y manipulada por el usuario final.

En este patrón de diseño, los componentes son:

- El **Modelo**, el cual almacena datos que son requeridos por el controlador y mostrados en la vista. En nuestro sistema, el modelo es la base de conocimientos de Prolog, la cual almacena todas las reglas de derivación e integración.
- La **Vista**, encargada de pedirle información al modelo a través del controlador, con la cual genera una representación de la salida para el usuario final. En nuestro sistema, la vista será todo el conjunto de tecnologías utilizadas en la interfaz gráfica, tanto la propia interfaz como los analizadores léxicos-sintácticos embebidos en la misma.
- El **Controlador**, el cual se comunica con el Modelo, cambiando el estado de los datos, y también se comunica con la Vista, cambiando su presentación. En nuestro sistema, el controlador es el servlet contenedor de la interfaz gráfica, el cual redirige las peticiones de la interfaz hacia la base de conocimientos.

3.4.5.2.2. Web Components Los componentes web constan de diversas tecnologías, las cuales son partes internas de los navegadores y por lo mismo no requieren librerías externas, como jQuery o Dojo. Un componente web puede ser utilizado sin escribir código, simplemente agregando una declaración *import* en la página HTML. Los componentes web utilizan estándares de navegadores nuevos o actualmente en desarrollo MDN (2015d).

Los componentes web utilizan estas cuatro tecnologías:

- Custom Elements
- HTML Templates
- Shadow DOM
- HTML Imports

Cabe mencionar que estas tecnologías se pueden utilizar por separado, aunque Polymer las utiliza en conjunto.

3.4.5.2.3. Custom Elements Es la capacidad para crear etiquetas y elementos HTML personalizados, los cuales pueden tener su propio comportamiento descrito a través de un script (JavaScript), y también pueden tener su propio estilo usando hojas de estilo en cascada (CSS). Son parte de los **Web Compoments**.

La principal ventaja de los **Custom Elements** es el uso de las *lifecycle callbacks* o funciones de retrollamada de ciclo de vida, las cuales son funciones que agregan cierto comportamiento cuando el elemento es registrado en e inicia su ciclo de vida, cuando es insertado en el DOM (Modelo Objeto-Documento), y también cuando es eliminado del DOM. MDN (2015e)

Los Custom Elements se registran en el navegador a través del método

```
Document.registerElement()
```

3.4.5.2.4. HTML Templates Las HTML templates o plantillas HTML son un mecanismo para guardar del lado del cliente, contenidos que no se mostrarán automáticamente cuando la página cargue, pero que pueden ser instanciados subsecuentemente durante el tiempo de ejecución, utilizando JavaScript. Cabe mencionar que aunque la plantilla no se muestre cuando se carga la página, el parser interno del navegador sí la procesa para asegurar que es un elemento HTML válido. MDN (2015b)

3.4.5.2.5. Shadow DOM El Shadow DOM (o Modelo Objeto-Documento "Sombra") provee encapsulamiento para el código JavaScript, CSS y plantillas HTML mediante un Web Component, de tal manera que estos elementos permanezcan separados del DOM del documento principal, aunque también es posible utilizar Shadow DOM por sí mismo, fuera de un web component.

El **Shadow DOM** siempre debe ser anexado a un elemento existente, el cual puede ser un elemento "literalçreado en un archivo HTML o un elemento creado en el DOM a través de un script. También puede ser un elemento nativo o un custom element. MDN (2015c)

Para anexar Shadow DOM a un elemento se utiliza el método

```
Element.createShadowRoot()
```

3.4.5.2.6. HTML Imports Es el mecanismo para empaquetar los Web Components, aunque también puede ser utilizado por sí mismo. MDN (2015a)

Para importar un archivo HTML, se utiliza la siguiente etiqueta:

```
<link rel="import" href="myfile.html">
```

3.4.5.3. Organización de código

Se utilizará **GIT** como controlador de versiones distribuído de código. Torvalds (2005).

Utilizaremos **Bower** como manejador de paquetes, con objeto de agrupar las dependencias del proyecto de manera eficiente y automatizar el proceso de instalación de las mismas. Contributors (2015)

El proyecto estará disponible en un repositorio de GitHub, al cual se puede accesar a través de la siguiente dirección: `http://www.github.com/uriStolar/tt2014-a017`

Los paquetes que no estén disponibles como dependencias de Bower se guardarán como dependencias de Node, mediante su propio manajeador de paquetes **NPM - Node Package Manager**. Isaac Z. Schlueter (2009)

3.4.5.4. Generación de código

Para la generación de la versión productiva del proyecto utilizaremos **Grunt.js**, el cual es un ejecutor automatizado de tareas de JavaScript. Ben Alman (2011)
Las tareas que ejecutaremos con ésta herramienta son:

- Montar un servidor de pruebas auto-actualizable (*grunt serve + livereload*)
- Revisión de sintaxis de código (*jshint + jslint*)
- Minimización y ofuscación de código para generar la versión "productiva"del proyecto (*usemin, htmlmin, cssmin, concat*)

También se seguirán las siguientes convenciones para organizar el código:

- La descripción del proyecto se encuentra en el archivo README.md", a nivel de la raíz del proyecto.
- Las dependencias del proyecto manejadas con Bower se guardan en la carpeta "bower_components", a nivel de la raíz del proyecto.
- Las dependencias del proyecto manejadas con NPM se guardan en la carpeta "node_modules", a nivel de la raíz del proyecto.
- Todos los Web Components creados están en archivos HTML independientes, a nivel de la raíz del proyecto, y se unen en el archivo ïndex.html", mediante HTML Imports.

Capítulo 4

Desarrollo del sistema

4.1. Introducción

En este capítulo se presenta la fase de desarrollo del sistema de acuerdo a lo planteado en la etapa de análisis y diseño. El desarrollo se compone de la implementación de los módulos y las pruebas que se realizaron para verificar la correctez de los resultados.

En la fase de análisis y diseño se propuso utilizar ciertas tecnologías que en la etapa de desarrollo se tuvieron que cambiar por diversas razones, entre ellas problemas de compatibilidad al momento de implementar dichas tecnologías. A continuación presentamos las implementaciones de los módulos del sistema y las dificultades que se tuvieron al momento de realizar su desarrollo, así como las soluciones propuestas para mitigar dichos problemas.

En la fase de pruebas se seleccionaron varios conjuntos de datos para realizarlas. En la siguiente sección presentaremos los conjuntos de pruebas utilizados así como los resultados obtenidos.

4.2. Módulos

4.2.1. Cliente

El módulo del cliente consta de la interfaz gráfica y los analizadores léxicos-sintácticos que se encuentran embebidos en el módulo, pero por su grado de complejidad los tratamos como módulos independientes.

La interfaz gráfica fue desarrollada con un framework basado en componentes web, llamado **Polymer** Polymer (2014) el cual está siendo desarrollado por Google y el W3C. Polymer utiliza tecnologías conocidas como HTML 5, CSS 3, JavaScript e implementa algunas nuevas, que en un futuro se convertirán en estándares para la web, como **Shadow DOM** W3C (2014b), **Custom Elements** (elementos personalizados) W3C (2014a), entre otras.

4.2.1.1. Problemática

La principal dificultad encontrada en este módulo fue la curva de aprendizaje del framework y sus nuevas tecnologías. Otro inconveniente de usar herramientas que se encuentran en etapa de desarrollo es que la documentación a veces es limitada y tiende a cambiar constantemente.

4.2.2. Servidor

El módulo del servidor se compone de la base de conocimientos desarrollada en Prolog y un servlet de Java que utilizamos para escuchar las peticiones que realiza el cliente y procesarlas.

Se utilizó SWI-Prolog 6.6 Wielemaker (1987) para desarrollar la base de conocimientos y el paquete JPL propio de SWI-Prolog para realizar la comunicación entre la instancia de Prolog y el servlet contenedor de la interfaz gráfica.

4.2.2.1. Problemática

La dificultad más grande de este módulo fue que no conocíamos el lenguaje de programación. Prolog ni el paradigma de programación lógica que utiliza, el cual es muy distinto a los paradigmas más conocidos, como la programación orientada a objetos o la programación imperativa de lenguajes como C, Java, Python, etc.

4.2.3. Comunicación Cliente-Servidor

La tecnología utilizada para realizar la conexión cliente-servidor se llama **JPL** Wielemaker (2014) y es un paquete propio de SWI-Prolog, desarrollado en Java y C, la cual embebe una instancia de Prolog en la máquina virtual de Java (JVM) y se compone de dos capas: una interfaz a bajo nivel entre Java y la Interfaz de Lenguaje Extranjero (FLI) de Prolog, y la interfaz a

alto nivel que consta de clases y métodos para el programador Java. Cabe destacar que la interfaz a bajo nivel permite migrar implementaciones hechas en C a Java por medio de la FLI. Esta tecnología está disponible desde la versión 3.1.0 de SWI-Prolog y se distribuye bajo licencia GNU-GPL (GNU General PublicLicense).

4.2.3.1. Problemática

Una de las dificultades que enfrentamos en esta etapa fue la comunicación entre el cliente y el servidor, la cual se había pensado realizar con Node.JS y una Interfaz de Programación de Aplicaciones (API) propia de SWI-Prolog llamada "Pengines"(o Prolog Engines). Cabe mencionar que esta API sólo está disponible en las últimas versiones de SWI-Prolog (mayores a 7.0) las cuales hasta la fecha están catalogadas como versiones en desarrollo y no son versiones estables.

Al inicio de la fase de desarrollo, se intentó implementar la comunicación cliente-servidor con Pengines, pero el mayor problema que se tuvo fue que cuando un cliente le hacía peticiones al servidor Prolog, seal cargar la base de conocimientos en una instancia de Prolog en el servidor, se ejecutaban únicamente los predicados más sencillos, como derivada de una constante y/o la derivada de una variable respecto a sí misma; el resto de predicados, los cuales hacen uso de la característica de "Backtracking"propia de Prolog eran marcados como "predicados inseguros"por Pengines, esto porque Prolog da acceso ilimitado al sistema de archivos propio del sistema operativo donde está ejecutándose, y, a través de JavaScript, se podría hacer mal uso de los recursos del sistema.

La solución que le dimos a esta problemática fue cambiar la tecnología para realizar la conexión cliente-servidor por una que fuera más estable y mejor documentada. Elegimos utilizar el conector Java-Prolog, que es más estable y está mucho mejor documentado que Pengines para JavaScript. A pesar de que podría parecer drástico el cambio de tecnología del backend de JavaScript a Java, no lo fue tanto, pues el frontend se cargó por completo en un servlet sin modificación alguna, y el backend en Java sigue empleando las partes que ya se tenían desarrolladas en JavaScript. Otro punto a favor fue que Java es un lenguaje que todo el equipo de desarrollo ha usado y tiene cierto dominio.

4.2.4. Analizadores Léxicos-Sintácticos

Los ALS desarrollados son:

1. Analizador para la entrada de datos con salida a Prolog (predicados Prolog a ser procesados por la base de conocimientos).

2. Analizador para la entrada de datos con salida en notación Leibniz (entrada reproducida en código LaTeX en el frontend de la aplicación).

3. Analizador léxico para la salida de datos Prolog (procesa un resultado arrojado por la base de conocimientos en Prolog y lo reproduce en una forma más legible para el usuario).

4. Analizador léxico para la salida de datos en notación Leibniz (toma la salida del ALS de salida Prolog y lo reproduce en código LaTeX con notación Leibniz).

Cabe mencionar que aunque dichos ALS comparten información, como los datos de entrada, cada uno de tiene su propia sintaxis, por lo que se genera una gramática única en cada ALS.

La tecnología para realizar los ALS fue **PEG.js** Majda (2013), un generador de analizadores sintácticos para JavaScript, el cual integra análisis léxico y sintáctico para cada gramática generada y está basado en un tipo de gramáticas formales analíticas, llamadas gramáticas de análisis sintáctico de expresiones (Parsing Expression Grammars o PEGs), que aunque son similares a las gramáticas libres de contexto (Context Free Grammars o CFGs), tienen varias diferencias. La principal diferencia entre las PEGs y las CFGs es que las PEGs no pueden ser ambiguas; si una cadena es procesada, generará únicamente un árbol de derivación, esto implica que en las PEGs el operador de elección tiene un orden, lo que infiere que el orden de elección no es conmutativo, como lo es en las CFGs.

Una de las ventajas de las PEGs es que estas gramáticas son más poderosas que el simple uso de expresiones regulares, aunque también requieren más uso de memoria.

La principal desventaja de las PEGs es el consumo de memoria, que en el caso de algunas gramáticas, la profundidad del árbol de derivación puede llegar a ser proporcional al tamaño de la entrada a procesar. Otra desventaja es que las PEGs no pueden expresar reglas de recursividad por la izquierda

sin saltar al siguiente caracter a analizar.

Esta desventaja no tiene un gran impacto en nuestro sistema, pues las cadenas de entrada de los ALS no son muy largas, y los propios analizadores están embebidos en el frontend de la aplicación en archivos JavaScript. Cabe mencionar que dichos analizadores pasan por un proceso de minimización y ofuscación de código, con el objetivo de acelerar los tiempos de carga de la interfaz gráfica y mejorar su desempeño.

4.3. Evaluación de Resultados

4.3.1. Introducción

La evaluación de los resultados de la presente herramienta se realizó frente a Mathematica 10, comparando los resultados obtenidos por cada una de ellas al introducir diferentes valores para los coeficientes de la fórmula.

A partir de la ecuación diferencial

$$a_2 y'' + a_1 y' + uy = f \tag{4.1}$$

los coeficientes utilizados en los conjuntos de datos son los siguientes:

- $a_2 = a_2(x)$
- $a_1 = a_1(x)$
- $f = f(x)$
- $u = u(x)$

Con base en la teoría que se expuso en el Capitulo 1.4.5 Modelo de Solución, el coeficiente u se calcula de la siguiente manera:

$$u = (a_1' - \frac{a_2'}{a_2} a_1) \tag{4.2}$$

por lo que la ecuación (4.1) queda de la siguiente forma:

$$a_2 y'' + a_1 y' + (a_1' - \frac{a_2'}{a_2} a_1) y = f(x) \tag{4.3}$$

Esta ecuación será la que utlizaremos para realizar la evaluación de resultados. Cabe mencionar que en Mathematica 10 se tiene que calcular el

coeficiente u con base en los coeficientes a_2, a_1 y sus derivadas, mientras que en nuestro sistema, éste coeficiente se calcula automáticamente.

4.3.2. Conjunto 1

El primer conjunto de datos que probamos fue el más sencillo posible para una ecuación diferencial lineal no homogénea de segundo órden con coeficientes no constantes. Estos coeficientes: a_2, a_1 y f toman los siguientes valores:

- $a_2 = x$
- $a_1 = x$
- $a_2' = 1$
- $a_1' = 1$
- $f = 4$
- $u = (a_1' - \frac{a_2'}{a_2}a_1) = (1 - \frac{1}{x}x) = 4$

Por lo tanto, la ecuación diferencial a resolver es la siguiente:

$$xy'' + xy' = 4$$

4.3.2.1. Mathematica 10

DSolve[xy"[x] + xy'[x] == 4, y, x]

Resultado

$\{\{y \to \text{Function}\,[\{x\}, -e^{-x}C[1] + C[2] - 4e^{-x}\text{ExpIntegralEi}[x] + 4\text{Log}[x]]\}\}$

4.3.2.2. Resultado del sistema

$$e^{-x}\int e^{x}[4 * log(x)]dx + C_1e^{-x}e^{x} + C_2e^{-x}$$

Captura de pantalla

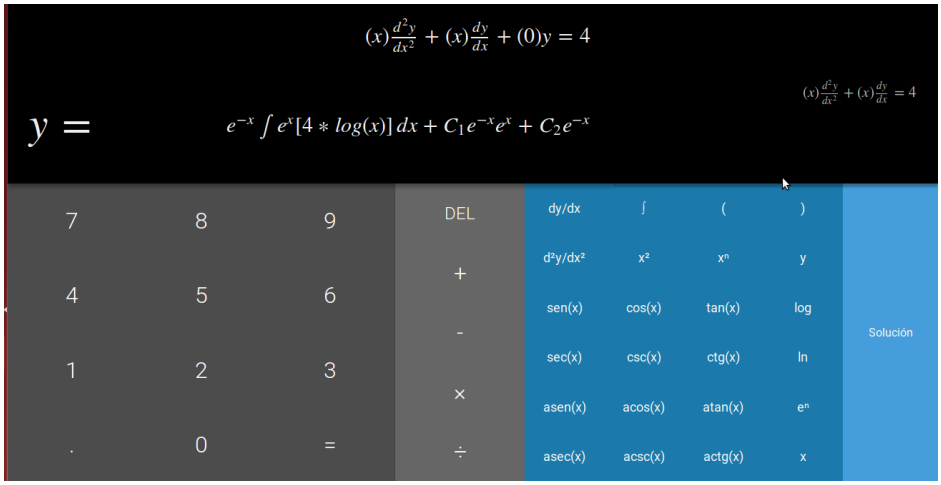

Figura 4.1: Resultado del sistema - Conjunto 1

Como podemos observar, tanto Mathematica 10 como nuestro sistema generó una solución incompleta.

4.3.3. Conjunto 2

El segundo conjunto de datos consta de los siguientes valores para los coeficientes de la fórmula:

- $a_2 = x$
- $a_1 = x$
- $a_2' = 1$
- $a_1' = 1$
- $f = xsin(x)$
- $u = (a_1' - \frac{a_2'}{a_2}a_1) = (1 - \frac{1}{x}x) = 0$

Por lo tanto, la ecuación diferencial a resolver es la siguiente:

$$xy'' + xy' = xsin(x)$$

4.3.3.1. Mathematica 10

DSolve[$xy''[x] + xy'[x] == \text{xSin}[x], y, x$]

Resultado

$$\left\{\left\{y \to \text{Function}\left[\{x\}, C[2] + \int_1^x \left(e^{-K[2]}C[1] + e^{-K[2]}\int_1^{K[2]} \frac{e^{K[1]}\text{xSin}[K[1]]}{K[1]}\, dK[1]\right) dK[2]\right]\right\}\right\}$$

4.3.3.2. Resultado del sistema

$$\frac{-sin(x) - cos(x)}{2} + C_2 e^{-x} + C_1$$

Captura de pantalla

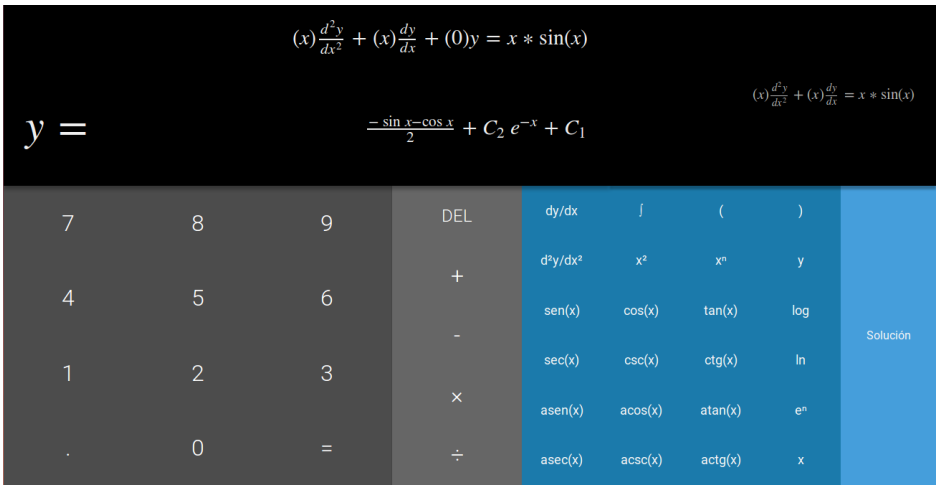

Figura 4.2: Resultado del sistema - Conjunto 2

Como se puede observar, Mathematica 10 no entregó una solución completa analítica, mientras que nuestro sistema sí lo hizo.

4.3.4. Conjunto 3

El tercer conjunto de datos tiene los siguientes valores para los coeficientes de la fórmula:

- $a_2 = cos(x)$
- $a_1 = sin(x)$
- $a_2' = -sin(x)$
- $a_1' = cos(x)$
- $f = tan(x)$
- $u = (a_1' - \frac{a_2'}{a_2}a_1) = (cos(x) - \frac{sin(x)}{cos(x)}sin(x)) = \frac{sin^2(x)}{cos(x)+cos(x)}$

Por lo tanto, la ecuación diferencial a resolver es la siguiente:

$$cos(x)y'' + sin(x)y' + \frac{sin^2(x)}{cos(x)} + cos(x)y = tan(x)$$

4.3.4.1. Mathematica 10

DSolve[{Cos[x]y"[x] + Sin[x]y'[x] + (((Sin[x])^2/ cos[x]) + cos[x])y[x]==Tan[x], y[0]==0, y'[0]==0}, y, x]

Resultado

$$\text{DSolve}\left[\left\{\left(\cos[x] + \frac{\text{Sin}[x]^2}{\cos[x]}\right)y[x] + \text{Sin}[x]y'[x] + \text{Cos}[x]y''[x] == \text{Tan}[x], y[0] == 0, y'[0] == 0\right\}, y, x\right]$$

4.3.4.2. Resultado del sistema

$$\frac{C_1 log(tan(x) + sec(x)}{sec(x)} + \frac{\tan(x)}{sec(x)} + \frac{C_2}{sec(x)}$$

Captura de pantalla

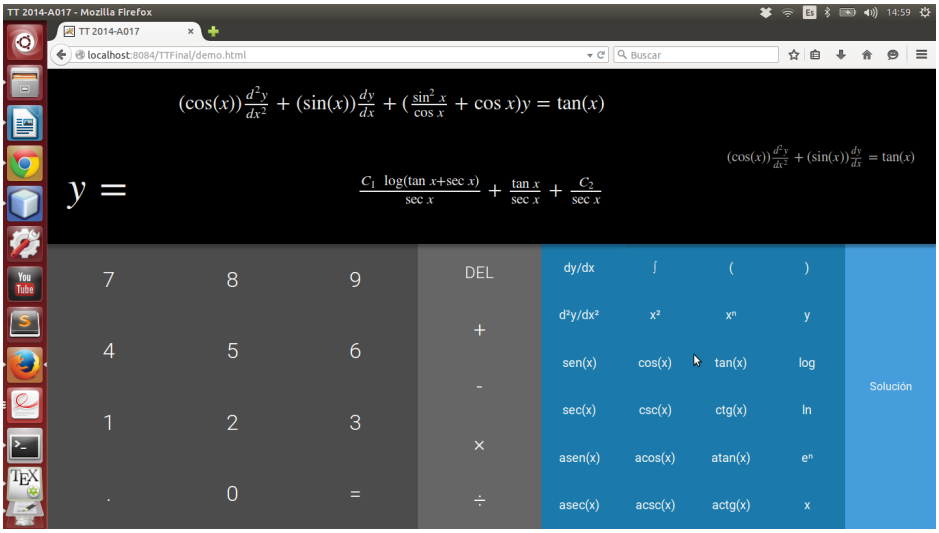

Figura 4.3: Resultado del sistema - Conjunto 3

Podemos observar que Mathematica nos regresó la misma función de entrada, lo que significa que no obtuvo resultado alguno, mientras que nuestro sistema si computó la solución analítica.

4.3.5. Conjunto 4

El primer conjunto de datos que probamos fue el más sencillo posible para una ecuación diferencial lineal de segundo órden con coeficientes no constantes. Estos coeficientes: a_2, a_1 y f toman los siguientes valores:

- $a_2 = 1$
- $a_1 = cot(x)$
- $a_2' = 0$
- $a_1' = -csc^2(x)$
- $f = x^2$
- $u = (a_1' - \frac{a_2'}{a_2}a_1) = -csc^2(x)$

Por lo tanto, la ecuación diferencial a resolver es la siguiente:

$$1 * y'' + cot(x)y' - csc^2(x)y = x^2$$

4.3.5.1. Mathematica 10

DSolve[{y"[x] + Cot[x]y'[x] − (Sec[x])^2y[x]==x^2, y[0]==0, y'[0]==0}, y, x]

Resultado

$$\text{DSolve}\left[\{-\text{Sec}[x]^2 y[x] + \text{Cot}[x]y'[x] + y''[x] == x^2, y[0] == 0, y'[0] == 0\}, y, x\right]$$

4.3.5.2. Resultado del sistema

$$\frac{(3x^2-6)sin(x)+(6x-x^3)cos(x)}{3sin(x)} - \frac{C_1 cos(x)}{sin(x)} + \frac{C_2}{sin(x)}$$

Captura de pantalla

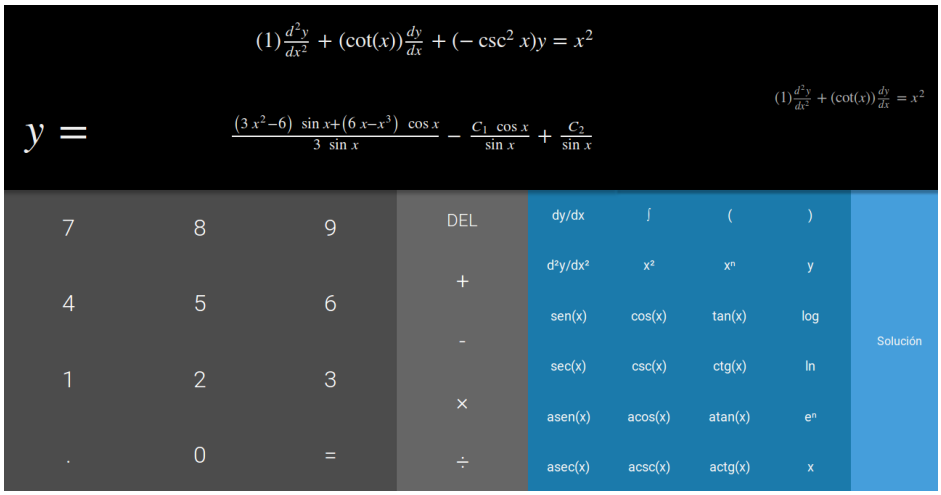

Figura 4.4: Resultado del sistema - Conjunto 4

Podemos observar que con este conjunto, Mathematica tampoco pudo dar solución, mientras que el sistema sí lo hizo.

4.3.6. Conjunto 5

El conjunto de datos 5 tiene los siguientes valores:

- $a_2 = cos(x)$

- $a_1 = -sin(x)$
- $a_2' = -sin(x)$
- $a_1' = -cos(x)$
- $f = x^2 cos(x)$
- $u = (a_1' - \frac{a_2'}{a_2} a_1) = -sec^2(x)$

4.3.6.1. Mathematica 10

DSolve[{Cos[x]y"[x] − Sin[x]y'[x] − (Sec[x])^2y[x] == (x^2)Cos[x], y[0]==0, y'[0]==0}, y, x]

Resultado

$$\text{DSolve}\left[\{-\text{Sec}[x]^2 y[x] - \text{Sin}[x] y'[x] + \text{Cos}[x] y''[x] == x^2 \text{Cos}[x], y[0] == 0, y'[0] == 0\}, y, x\right]$$

4.3.6.2. Resultado del sistema

$$\frac{(x^3 - 6x)sin(x)(3x^2 - 6)cos(x)}{3cos(x)} + \frac{C_1 sin(x)}{cos(x)} + \frac{C_2}{cos(x)}$$

Captura de pantalla

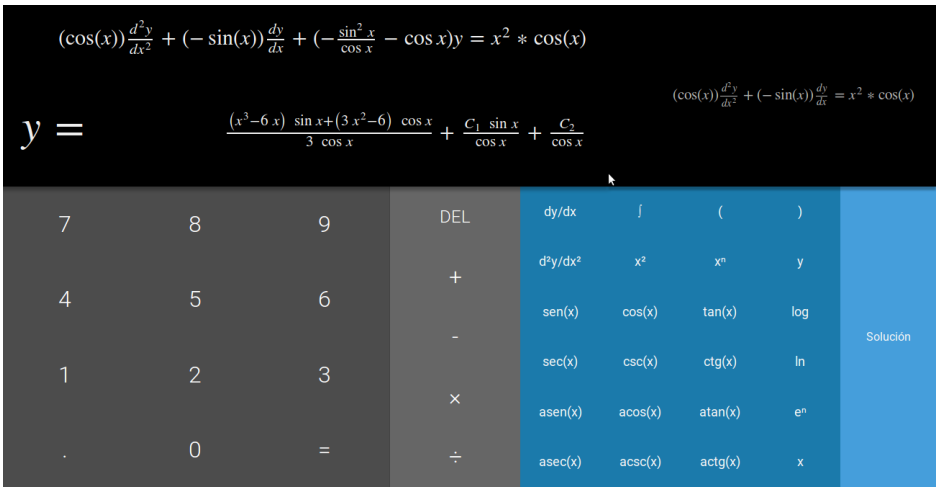

Figura 4.5: Resultado del sistema - Conjunto 5

En este caso Mathematica tampoco pudo computar una solución, mientras que el sistema sí lo hizo.

Capítulo 5

Conclusiones

5.1. Introducción

En este capítulo se presentan las conclusiones de los integrantes del equipo de manera individual.

5.2. Conclusiones

El objetivo general del proyecto, que era la reoslución de ecuaciones diferenciales lineales de segundo orden con coeficientes no constantes mediante el modelado de la técnica de solución del Dr. Encarnación Salinas Hernández se cumplió satisfactoriamente.

Comparando los resultados del prototipo con otro sistema que da solución a ecuaciones diferenciales (Mathematica 10) podemos afirmar que nuestro sistema resuelve analíticamente algunas E.D. lineales de segundo orden con coeficientes variables, que Mathematica no resuelve, o en muchas ocasiones tarda un tiempo considerable en generar una salida (más de 10 minutos), llegando a la misma solución, lo cual verifica la correctez de los resultados del prototipo.

El prototipo elimina la complejidad de uso y de visualización inherente de los sistemas de álgebra computacional, como Mathematica, ya que no se requiere de ningún conocimiento previo de programación o de alguna sintaxis de un lenguaje en particular, ayudando al usuario final a que pueda resolver ecuaciones diferenciales necesitando únicamente conocimientos previos de la

materia.

Pensando en la accesibilidad hacia el usuario y facilidad de instalación del sistema, el prototipo es presentando como un proyecto web, que por naturaleza es multiplataforma, y responsivo, lo que hace que se adapta al tamaño del dispositivo que lo contiene.

Por las razones expuestas anteriormente, consideramos que se cumplieron los objetivos generales y particulares del proyecto.

Capítulo 6

Trabajo a futuro

6.1. Introducción

En este capítulo se expone el desarrollo que se tendría que realizar para darle continuidad al trabajo y ampliar sus capacidades.

6.2. Base de Conocimientos

Este rubro tiene un potencial muy grande, pues trabajando en la base de conocimientos se incrementaría el número de ecuaciones diferenciales a las que podría dar resultado el presente sistema.
Para ampliar la base de conocimientos se requeriría programar más reglas de derivación e integración, para de esta manera poder ampliar el rango de valores que pueden tomar los coeficientes de la fórmula.
Algunas de las reglas que podrían ampliar la base de conocimientos son las siguientes:

- Integrales cíclicas
- Integrales de cocientes con funciones trigonométricas y/o exponenciales

6.3. Métodos de Resolución

Para ampliar las capacidades del proyecto y poner en uso la reusabilidad de los componentes web que se crearon en la interfaz, se podría proponer otro método de resolución de ecuaciones diferenciales que entregue resultados numéricos y no analíticos, por ejemplo el método de **Runge-Kutta**.

6.4. Módulo de Graficación

Otro rubro en el que se podría crecer el potencial del proyecto es implementando un módulo de graficación para el resultado de la ecuación diferencial a resolver. Hay muchas tecnologías compatibles con nuestro proyecto, pero una de las más notables es **JSXGraph**. Alfred Wassermann (2013)

Bibliografía

Alfred Wassermann, C. Jsxgraph. Disponible en `http://jsxgraph.uni-bayreuth.de/wp/`.

Ann Boyle, C. C. y Hearn, A. C. Future directions for research in symbolic computation. *Society for Industrial and Applied Mathematics*, páginas 9–13, 1990.

Baldor, A. *Álgebra*. Publicaciones Cultural, 1997.

Ben Alman, C., Kyle Robinson Young. Grunt.js. Disponible en `http://gruntjs.com/`.

Contributors, T. Bower. Disponible en `http://www.bower.io`.

Dennis G. Zill, M. R. C. *Matemáticas Avanzadas para Ingeniería, Vol. 1 Ecuaciones Diferenciales*. Mc Graw Hill, 2008.

Granville. *Cálculo Diferencial e Integral*. Limusa, 1998.

Isaac Z. Schlueter, R. B., Laurie Voss. Npm. Disponible en `https://www.npmjs.com/`.

Leithold, L. *El Cálculo, 7a ed.*. Oxford University Press, 1998.

Majda, D. Peg.js. Disponible en `http://www.pegjs.org/`.

MDN, M. Html imports by mdn. Disponible en `https://developer.mozilla.org/en-US/docs/Web/Web_Components/HTML_Imports`.

MDN, M. Html templates by mdn. Disponible en `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template`.

MDN, M. Shadow dom by mdn. Disponible en `https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM`.

MDN, M. Web components. Disponible en https://developer.mozilla.org/en-US/docs/Web/Web_Components.

MDN, M. Web components by mdn. Disponible en https://developer.mozilla.org/en-US/docs/Web/Web_Components.

NEUSS, N. Femlisp. Disponible en http://http://www.femlisp.org/.

NEUSS, N. Gnu scientific library for lisp. Disponible en https://common-lisp.net/project/gsll/.

POLYMER. Polymer project. Disponible en https://www.polymer-project.org/1.0/.

TEAM, S. D. Sympy. Disponible en http://www.sympy.org/es/index.html.

TORVALDS, L. Git. Disponible en https://git-scm.com/.

TOURETZKY, D. S. *COMMON LISP: A Gentle Introduction to Symbolic Computation*. Carnegie Mellon University, 1990.

W3C. Custom elements. Disponible en http://www.w3.org/TR/custom-elements/.

W3C. Shadow dom. Disponible en http://www.w3.org/TR/shadow-dom/.

WIELEMAKER, J. Swi prolog. Disponible en http://www.swi-prolog.org/.

WIELEMAKER, J. Swi-prolog jpl. Disponible en http://www.swi-prolog.org/packages/jpl/java_api/.