Resolución de Sistemas de Ecuaciones Lineales Python, R con Shiny

NOMBRE: Beatriz Umiña Machaca Código: 230035 Docente: Fred Torres Cruz

05 de mayo de 2025

1. Introducción

El propósito de este proyecto es resolver sistemas de ecuaciones lineales con dos variables $(x \ y \ y)$ utilizando métodos algebraicos: **sustitución**, **igualación** y **reducción**. El sistema fue inicialmente implementado en **Python** para su uso en consola, y posteriormente migrado y mejorado en **R** utilizando el paquete *Shiny*, para ofrecer una interfaz gráfica interactiva.

2. Primera Versión: Python (Interfaz de Consola)

Características

- Implementación en consola usando sympy, re y entrada de texto con input().
- Análisis de ecuaciones con expresiones regulares (re.match).
- Resolución de sistemas mediante tres métodos:

```
Ingresa:
1: 2*x-y=7
2: x-3*y=-7

Elige el método para resolver:
1. Sustitución
2. Igualación
3. Reducción
4. Salir_
```

• Sustitución

Ingresa: 1: x+2*y=10 2: 2*x-y=5 Elige el método para resolver: 1. Sustitución 2. Igualación 3. Reducción 4. Salir Opción: 1 Método de Sustitución: Resultado: x = 4 y = 3

• Igualación

```
Ingresa:
    1: x+2*y=10
    2: 2*x-y=5

Elige el método para resolver:
    1. Sustitución
    2. Igualación
    3. Reducción
    4. Salir
    Opción: 2

Método de Igualación:
    Resultado:
    x = 4
    y = 3
```

• Reducción (resolución directa con solve)

```
Ingresa:
    1: x+2*y=10
    2: 2*x-y=5

Elige el método para resolver:
    1. Sustitución
    2. Igualación
    3. Reducción
    4. Salir
    Opción: 3

Método de Reducción:
    Resultado:
    x = 4
    y = 3
```

Manejo de errores en entrada y resolución.

Lógica de resolución

- Se parsean ecuaciones tipo ax + by = c desde texto.
- Se despeja una variable y se sustituye en la otra ecuación.
- Se imprime el resultado paso a paso para el usuario.

Ventajas

- Rápido de desarrollar.
- Ideal para pruebas de lógica algebraica.

Limitaciones

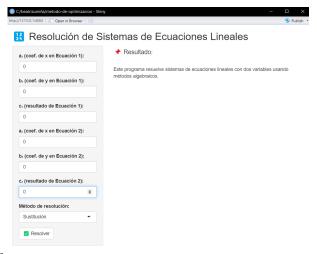
- Requiere conocimiento básico de sintaxis matemática para ingresar las ecuaciones.
- No amigable para usuarios no técnicos.
- Poco visual e interacción limitada.

3. Segunda Versión: R con Shiny (Interfaz Gráfica Web)

Características

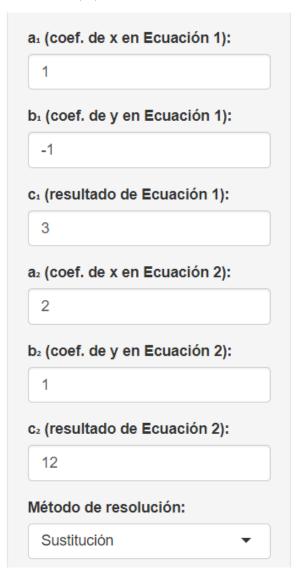
■ Interfaz gráfica desarrollada con shiny, accesible vía navegador.

```
http://127.0.0.1:6597
```

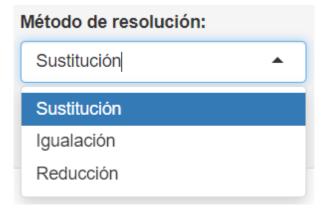


center

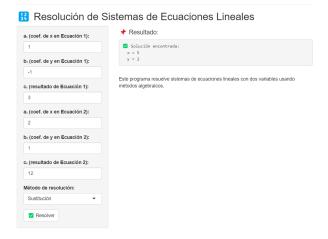
Entradas numéricas para coeficientes a, b, c de cada ecuación.



Menú desplegable para elegir el método de resolución.



Botón para ejecutar el cálculo y mostrar resultados.



Lógica de resolución

- Los métodos implementan la lógica matemática directamente en R:
 - En sustitución e igualación, se usan funciones y uniroot para encontrar raíces.
 - En reducción, se resuelve el sistema con álgebra matricial (solve()).

Mejoras respecto a Python

- Interfaz intuitiva y visual, sin necesidad de escribir ecuaciones.
- Mayor control de errores de entrada (valores numéricos esperados).
- Mejor presentación del resultado.
- Ideal para educación, demostraciones y uso público.

4. Conclusión

La evolución del proyecto refleja una clara mejora en accesibilidad, usabilidad y presentación. Mientras que la versión en Python demostró ser funcional y útil para propósitos técnicos, la migración a R con Shiny permitió ofrecer una solución más visual, intuitiva y apta para usuarios de todos los niveles. Este desarrollo progresivo evidencia cómo una herramienta técnica puede transformarse en una aplicación amigable y accesible, sin perder su rigor matemático.

5. Anexos

Anexo A: Código en Python

```
import re
from sympy import symbols, Eq, solve, sympify
x, y = symbols('x_{\perp}y')
def parsear_ecuacion(ecuacion_str):
    ecuacion_str = ecuacion_str.replace("u", "")
    match = re.match(r'(.+)=([\+\-]?\d+)', ecuacion_str)
    if match:
         izquierda = sympify(match.group(1))
         derecha = sympify(match.group(2))
         return Eq(izquierda, derecha)
     else:
         raise ValueError("La_{\sqcup}ecuaci n_{\sqcup}no_{\sqcup}tiene_{\sqcup}el_{\sqcup}formato_{\sqcup}esperado")
def ingreso_ecuaciones():
    print("Ingresa:")
    try:
         eq1_input = input("\( \_1:\( \_1'' \)
         eq2_input = input("\u2:\u1")
         eq1 = parsear_ecuacion(eq1_input)
         eq2 = parsear_ecuacion(eq2_input)
         return eq1, eq2
     except Exception as e:
         print("_{\sqcup}Error_{\sqcup}al_{\sqcup}interpretar_{\sqcup}las_{\sqcup}ecuaciones:", e)
         return None, None
def metodo_sustitucion(eq1, eq2):
    print("\n_{\sqcup}M todo_{\sqcup}de_{\sqcup}Sustituci n:")
         despeje = solve(eq1, y)
         if not despeje:
              despeje = solve(eq1, x)
              sustituida = eq2.subs(x, despeje[0])
              solucion = solve(sustituida, y)
              x_valor = despeje[0].subs(y, solucion[0])
         else:
              sustituida = eq2.subs(y, despeje[0])
              solucion = solve(sustituida, x)
              y_valor = despeje[0].subs(x, solucion[0])
              x_valor = solucion[0]
         print("_Resultado:")
         print("x_{\sqcup}=", x_{\_}valor)
         print("y_{\perp}=", y_{\perp}valor if 'y_{\perp}valor' in locals() else solucion[0])
     except Exception as e:
         print(" \cup No \cup se \cup pudo \cup resolver \cup por \cup sustituci n:", e)
def metodo_igualacion(eq1, eq2):
    print("\n<sub>□</sub>M todo<sub>□</sub>de<sub>□</sub>Igualaci n:")
    try:
         despeje1 = solve(eq1, y)
         despeje2 = solve(eq2, y)
         igualado = Eq(despeje1[0], despeje2[0])
         x_valor = solve(igualado, x)[0]
         y_valor = despeje1[0].subs(x, x_valor)
         print(" LResultado:")
         print("x_{\sqcup}=", x_{\_}valor)
         print("y_{\sqcup}=", y_{\bot}valor)
    except Exception as e:
```

```
print("⊔No⊔se⊔pudo⊔resolver⊔por⊔igualaci n:", e)
def metodo_reduccion(eq1, eq2):
     print("\n_{\square}M todo_{\square}de_{\square}Reducci n:")
     try:
          sol = solve((eq1, eq2), (x, y))
          \verb|print("|Resultado:")|
          print("x_{\sqcup}=", sol[x])
print("y_{\sqcup}=", sol[y])
     except Exception as e:
          print("_{\,\sqcup} No_{\,\sqcup} se_{\,\sqcup} pudo_{\,\sqcup} resolver_{\,\sqcup} por_{\,\sqcup} reducci~n:",~e)
def main():
     print("\SISTEMA\DE\ECUACIONES\\n")
     eq1, eq2 = ingreso_ecuaciones()
     if eq1 is None or eq2 is None:
          return
     while True:
          print("\n_{\sqcup}Elige_{\sqcup}el_{\sqcup}m\ todo_{\sqcup}para_{\sqcup}resolver:")
          \texttt{print("1.} \, \sqcup \, \texttt{Sustituci n")}
          print("2._{\,\sqcup\,} \texttt{Igualaci n"})
          print("3. □ Reducci n")
          print("4. □ Salir")
          opcion = input("Opci n:⊔")
          if opcion == '1':
               metodo_sustitucion(eq1, eq2)
          elif opcion == '2':
               metodo_igualacion(eq1, eq2)
          elif opcion == '3':
               metodo_reduccion(eq1, eq2)
          elif opcion == '4':
               print("⊔Hasta⊔luego.⊔Vuelve⊔pronto⊔:)⊔")
               break
          else:
               print("□opcion□invalida.")
if __name__== "__main__":
     main()
```

Anexo B: Código en R (Shiny)

```
library(shiny)
resolver_sustitucion <- function(a1, b1, c1, a2, b2, c2) {

if (b1 == 0) stop("Nouseupuedeuaplicarusustituci nusiub1u=u0")
despeje_y <- function(x) (c1 - a1 * x) / b1

f <- function(x) a2 * x + b2 * despeje_y(x) - c2
x_val <- uniroot(f, c(-1e3, 1e3))$root
y_val <- despeje_y(x_val)
return(c(x = round(x_val, 4), y = round(y_val, 4)))
}

resolver_igualacion <- function(a1, b1, c1, a2, b2, c2) {

if (b1 == 0 || b2 == 0) stop("Nouseupuedeuaplicaruigualaci nusiub1uoub2u=u0"
)
y1 <- function(x) (c1 - a1 * x) / b1
y2 <- function(x) (c2 - a2 * x) / b2
```

```
f \leftarrow function(x) y1(x) - y2(x)
    x_val \leftarrow uniroot(f, c(-1e3, 1e3))root
    y_val <- y1(x_val)
    return(c(x = round(x_val, 4), y = round(y_val, 4)))
resolver_reduccion <- function(a1, b1, c1, a2, b2, c2) {
   mat <- matrix(c(a1, b1, a2, b2), ncol = 2, byrow = TRUE)</pre>
    vec <- c(c1, c2)
   sol <- solve(mat, vec)
    return(c(x = round(sol[1], 4), y = round(sol[2], 4)))
ui <- fluidPage(
   titlePanel("
                                                \squareResoluci n_{\square}de_{\square}Sistemas_{\square}de_{\square}Ecuaciones_{\square}Lineales"),
     sidebarLayout(
          sidebarPanel(
              numericInput("c2", "c u(resultadoudeuEcuaci nu2):", value = 1),
               selectInput("metodo", "M todo de resoluci n:",
                                             choices = c("Sustituci n", "Igualaci n", "Reducci n")),
              actionButton("resolver", " ⊔Resolver")
         ),
         mainPanel(
             h4("
                                      _{\sqcup}Resultado:"),
              verbatimTextOutput("resultado"),
             tags$br(),
              tags p("Este_{\sqcup}programa_{\sqcup}resuelve_{\sqcup}sistemas_{\sqcup}de_{\sqcup}ecuaciones_{\sqcup}lineales_{\sqcup}con_{\sqcup}dos_{\sqcup}
                       variables_{\sqcup}usando_{\sqcup}m todos_{\sqcup}algebraicos.")
    )
)
server <- function(input, output) {</pre>
    observeEvent(input$resolver, {
          a1 <- input$a1; b1 <- input<math>$b1; c1 <- input$c1
          a2 <- input$a2; b2 <- input$b2; c2 <- input$c2
         metodo <- input$metodo</pre>
         resultado <- tryCatch({</pre>
              if (metodo == "Sustituci n") {
                  res <- resolver_sustitucion(a1, b1, c1, a2, b2, c2)
              } else if (metodo == "Igualaci n") {
                  res <- resolver_igualacion(a1, b1, c1, a2, b2, c2)
              } else {
                   res <- resolver_reduccion(a1, b1, c1, a2, b2, c2)
              paste0("
                                      \Box Soluci n_{\Box} encontrada:n_{\Box\Box} n_{\Box\Box} n_{\Box} n_{\Box
         }, error = function(e) {
              paste("
                                   ⊔Error:", e$message)
         })
         output$resultado <- renderText({ resultado })</pre>
```

```
})
}
shinyApp(ui = ui, server = server)
```