

Actividad | #3 | Bisección

Métodos Numéricos

Ingeniería en Desarrollo de Software



TUTOR: Miguel Angel Rodríguez Vega

ALUMNO: Francisco Antonio Herrera Silvas

FECHA: 16/07/2025

DESCRIPCION.....	4
JUSTIFICACION.....	5
Desarrollo.....	6
a)metodo de biseccion	6
b)metodo de Jacobi	7
MetodoGauss-Seidel.....	11
CONCLUSION.....	13

INTRODUCCION

En el campo de la ingeniería, las matemáticas aplicadas y la informática, los métodos numéricos representan una herramienta fundamental para abordar problemas complejos cuya resolución analítica resulta impráctica o imposible. La presente actividad se centra en tres importantes métodos dentro del análisis numérico: el método de Bisección, y los métodos iterativos de Jacobi y Gauss-Seidel. Todos ellos permiten obtener soluciones aproximadas a problemas matemáticos, como raíces de funciones no lineales o sistemas de ecuaciones lineales, de forma más eficiente que los métodos exactos, especialmente cuando se trabaja con computadoras.

El método de Bisección es un algoritmo de búsqueda de raíces que se basa en el Teorema del Valor Intermedio y que resulta particularmente útil cuando se necesita garantizar la convergencia. Su simplicidad lo hace ideal para introducirse en los métodos numéricos. Por otro lado, los métodos de Jacobi y Gauss-Seidel son algoritmos iterativos empleados para resolver sistemas lineales de ecuaciones. Ambos métodos utilizan una técnica de aproximación progresiva, y son útiles cuando se trabaja con matrices grandes y dispersas, como ocurre frecuentemente en la resolución de problemas de ingeniería y física computacional.

Esta actividad busca aplicar y comparar estos métodos mediante la implementación en RStudio, una plataforma de programación estadística ampliamente utilizada tanto en el ámbito académico como profesional. Además de realizar la implementación, se evaluarán sus resultados y se responderán preguntas clave orientadas al análisis de la facilidad de uso y eficiencia de cada método.

El desarrollo de esta actividad no solo permite profundizar en el entendimiento de los algoritmos matemáticos, sino también afinar habilidades en programación aplicada, interpretación de resultados numéricos y pensamiento lógico. En un contexto donde la digitalización de procesos y la resolución computacional de problemas son cada vez más comunes, dominar estos métodos representa una competencia altamente valiosa para los estudiantes y profesionales de áreas técnicas y científicas.

DESCRIPCION

El contexto presentado resalta la relevancia de los métodos numéricos como herramientas eficaces para la resolución de problemas complejos, que no pueden solucionarse con métodos algebraicos tradicionales. Al ser métodos indirectos, su objetivo no es encontrar una solución exacta, sino una aproximación lo suficientemente precisa que permita tomar decisiones informadas o avanzar en cálculos posteriores. En ese sentido, esta actividad representa una oportunidad para aplicar algoritmos computacionales a problemas matemáticos comunes.

La elección del método de Bisección como primera parte de la actividad permite abordar el problema de encontrar la raíz de una ecuación no lineal de forma sencilla y segura. Aunque es más lento que otros métodos como Newton-Raphson o la Secante, la Bisección tiene la ventaja de garantizar convergencia siempre que se parta de un intervalo donde la función cambie de signo. Esto lo convierte en un punto de partida ideal para introducir conceptos de análisis numérico.

En la segunda parte de la actividad, se aplican los métodos iterativos de Jacobi y Gauss-Seidel para resolver un sistema de ecuaciones lineales. Ambos algoritmos permiten encontrar soluciones aproximadas mediante iteraciones sucesivas y dependen de ciertas condiciones de convergencia, como que la matriz del sistema sea diagonalmente dominante. En contextos reales, como el modelado de circuitos eléctricos, estructuras mecánicas o flujos de fluidos, estos métodos permiten obtener soluciones sin necesidad de invertir matrices, lo que representa una ventaja computacional significativa.

Interpretar estos métodos implica entender que no son solo técnicas de cálculo, sino formas de optimizar el uso de recursos computacionales y aumentar la precisión en problemas de gran escala. En este sentido, es importante no solo aplicarlos, sino también compararlos en términos de velocidad de convergencia, facilidad de implementación, y robustez frente a diferentes tipos de datos. Además, el uso de RStudio como entorno de programación potencia las capacidades del análisis al permitir automatizar procesos, visualizar resultados y validar hipótesis.

JUSTIFICACION

La elección de métodos numéricos como solución para los problemas presentados en esta actividad está plenamente justificada tanto por su aplicabilidad como por su eficiencia. Estos métodos permiten abordar una gran variedad de problemas en distintas disciplinas donde las soluciones analíticas no son viables, ya sea por su complejidad o por las características del sistema. La programación del método de Bisección y la resolución de un sistema mediante Jacobi y Gauss-Seidel no solo refuerzan la comprensión teórica, sino que también desarrollan habilidades técnicas esenciales en el campo de las ciencias computacionales y la ingeniería.

El método de Bisección, en particular, destaca por su robustez y confiabilidad. Aunque puede resultar más lento que otros métodos más avanzados, su implementación es sencilla y siempre garantiza la convergencia cuando se cumplen las condiciones adecuadas. Esto lo convierte en una herramienta indispensable, especialmente en etapas iniciales del modelado matemático o cuando se necesita una validación de soluciones obtenidas por métodos más rápidos pero menos estables.

Los métodos de Jacobi y Gauss-Seidel, por otro lado, son ideales para resolver sistemas de ecuaciones de manera iterativa. En muchas aplicaciones reales, como en el análisis estructural, redes eléctricas o simulaciones numéricas, los sistemas a resolver son tan grandes que los métodos directos como la eliminación de Gauss resultan computacionalmente costosos. Los métodos iterativos permiten resolver estos sistemas con menor carga computacional, especialmente si se cuenta con una buena aproximación inicial o si se aprovechan las propiedades de la matriz.

La justificación también se apoya en la herramienta empleada: RStudio. Esta plataforma no solo permite codificar los algoritmos, sino también visualizarlos, realizar pruebas con distintos parámetros y comparar resultados de manera clara y reproducible. En un entorno donde la toma de decisiones debe basarse en datos y resultados concretos, este tipo de solución fortalece la capacidad de análisis crítico y técnico de los estudiantes.

Finalmente, desde una perspectiva educativa, el uso de estos métodos en una actividad práctica promueve el aprendizaje activo, el desarrollo del pensamiento algorítmico y la comprensión profunda de conceptos matemáticos. Los estudiantes no solo replican un procedimiento, sino que comprenden sus fundamentos, ventajas, limitaciones y aplicaciones. Esto los prepara para enfrentar desafíos reales en su campo profesional, donde la capacidad de aplicar métodos numéricos adecuados puede marcar la diferencia entre una solución funcional y una ineficiente.

Desarrollo

a)metodo de biseccion

```
# Método de Bisección en R

biseccion <- function(f, a, b, tol = 1e-6, max_iter = 100) {
  if (f(a) * f(b) > 0) {
    stop("La función no cambia de signo en el intervalo dado")
  }

  iter <- 0
  while ((b - a)/2 > tol && iter < max_iter) {
    c <- (a + b)/2
    if (f(c) == 0) {
      return(c)
    } else if (f(a) * f(c) < 0) {
      b <- c
    } else {
      a <- c
    }
    iter <- iter + 1
  }
  return((a + b)/2)
}

# Definir una función ejemplo
f <- function(x) 3*x - 5

# Llamar al método de bisección
raiz <- biseccion(f, 0, 5)
cat("La raíz aproximada es:", raiz)
```

En esta captura se puede observar la programación del método de bisección en R

b) metodo de Jacobi


```
# Función para el método de Jacobi
jacobí <- function(A, b, tol = 1e-10, max_iter = 100) {
  n <- nrow(A)
  x <- rep(0, n)
  x_new <- rep(0, n)

  for (iter in 1:max_iter) {
    for (i in 1:n) {
      sum1 <- sum(A[i, -i] * x[-i])
      x_new[i] <- (b[i] - sum1) / A[i, i]
    }
  }
}
```

En esta primera parte podemos ver la creación del método

```
# Criterio de parada
if (sqrt(sum((x_new - x)^2)) < tol) {
  cat("Convergencia alcanzada en", iter, "iteraciones.\n")
  return(x_new)
}
x <- x_new
}
cat("Máximo número de iteraciones alcanzado.\n")
return(x_new)
}
```


En esta segunda parte generamos las iteraciones

Functions	
jacobí	function (A, b, tol = 1e-10, max_iter = 100) 

Aquí guarda la función del método

```
#generar la matrices
A<- matrix(c(3,-1,-1,
            -1,3,1,
            2,1,4), nrow = 3, byrow= TRUE)
b<-c(1,3,7)
```

Aquí guardamos las 2 matrices, la primera guarda los valores de X,Y y Z , y en la segunda guarda los valores de los resultados

Data		
A	num	[1:3, 1:3] 3 ... 
Values		
b	num	[1:3] 1 3 7

Aquí guarda las matrices A y b

```
# Resolver usando Jacobi
solution <- jacobi(A, b)

# Resultado
print("Solución del sistema (Jacobi):")
print(solution)
```

En esta seccion se resuelve el sistema de ecuaciones

The screenshot shows the RStudio environment with the following components:

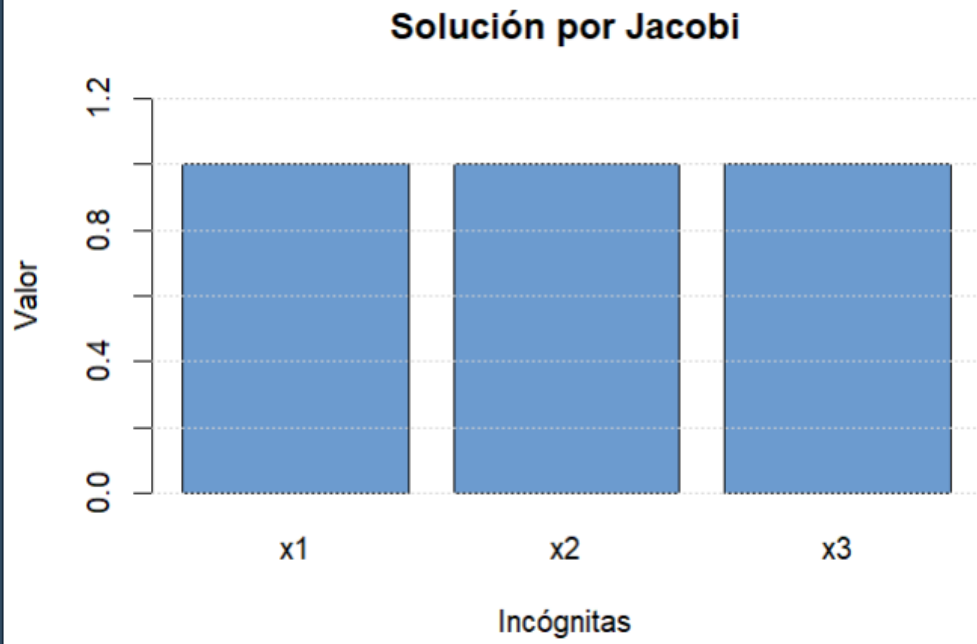
- Source Editor:** Contains the R script for the `jacobi` function and its execution. The function iterates until convergence or reaches a maximum of 100 iterations. The console output shows convergence at 23 iterations and the solution vector `[1, 1, 1]`.
- Console:** Displays the execution of the script, including the definition of matrix `A`, vector `b`, and the resulting solution.
- Environment:** Shows the global environment with variables `A`, `b`, and `solution`.
- Data:** A table summarizing the data in the environment:

Variable	Type	Dimensions	Values
A	num	[1:3, 1:3]	3 ...
b	num	[1:3]	1 3 7
solution	num	[1:3]	1 1 1
- Functions:** Lists the `jacobi` function.

En esta captura se resuelve el método y guarda el resultado

```
# Gráfica de la solución
barplot(solution,
  names.arg = paste("x", 1:length(solution), sep = ""),
  col = "#6C9BCF",
  main = "Solución por Jacobi",
  xlab = "Incógnitas",
  ylab = "Valor",
  ylim = c(0, max(solution) + 0.2))
grid(nx = NA, ny = NULL)
```

En esta seccion se grafica la solucion



MetodoGauss-Seidel

```
# Función para Gauss-Seidel
gauss_seidel <- function(A, b, tol = 1e-10, max_iter = 100) {
  n <- nrow(A)
  x <- rep(0, n)

  for (iter in 1:max_iter) {
    x_old <- x

    for (i in 1:n) {
      sum1 <- sum(A[i, -i] * x[-i])
      x[i] <- (b[i] - sum1) / A[i, i]
    }
  }
}
```

En el metodo de gauss se hace casi similar al metodo de jacobi

```
# Criterio de parada
  if (sqrt(sum((x - x_old)^2)) < tol) {
    cat("Convergencia alcanzada en", iter, "iteraciones.\n")
    return(x)
  }
}
cat("Máximo número de iteraciones alcanzado.\n")
return(x)
}

#generar la matrices
A<- matrix(c(3,-1,-1,
             -1,3,1,
             2,1,4), nrow = 3,byrow= TRUE)
b<-c(1,3,7)

# Resolver usando Gauss-Seidel
solution <- gauss_seidel(A, b)

# Resultado
print("Solución del sistema (Gauss-Seidel):")
print(solution)

# Gráfica de la solución
barplot(solution,
         names.arg = paste("x", 1:length(solution), sep = ""),
         col = "#FF9494",
         main = "Solución por Gauss-Seidel",
         xlab = "Incógnitas",
         ylab = "Valor",
         ylim = c(0, max(solution) + 0.2))
grid(nx = NA, ny = NULL)
```

Se genera igual el ciclo,matrices ,resultados y grafica

Respuestas a las preguntas

1.- ¿Cuál es el método que resultó más fácil de utilizar?

El método que resultó más fácil de utilizar fue el método de Bisección. Su estructura lógica es muy clara y fácil de seguir: parte de un intervalo

$[a,b]$ en el que la función cambia de signo, calcula el punto medio y determina en qué subintervalo se encuentra la raíz, repitiendo el proceso hasta alcanzar una precisión deseada. Este procedimiento no requiere conocimientos avanzados de álgebra lineal ni condiciones complejas de convergencia, como sí ocurre con los métodos iterativos para sistemas de ecuaciones.

Además, el algoritmo de Bisección es muy estable y confiable. Implementarlo en RStudio fue sencillo, ya que solo se necesitan operaciones básicas, algunas funciones de control como `while` o `for`, y evaluaciones lógicas. Esta facilidad permite enfocarse en entender el comportamiento de la función y cómo evoluciona la solución, sin preocuparse por errores numéricos o divergencias.

Por otro lado, los métodos de Jacobi y Gauss-Seidel requieren trabajar con matrices, entender el sistema de ecuaciones en su forma matricial y aplicar iteraciones más elaboradas, lo cual implica mayor carga cognitiva, especialmente al implementar condiciones de convergencia y control de errores. Aunque no son difíciles de programar, demandan un manejo más avanzado de lógica matemática.

2.- ¿Cuál es el método más eficiente? ¿Por qué?

El método más eficiente en esta actividad fue el método de Gauss-Seidel. En comparación con el método de Jacobi, Gauss-Seidel generalmente converge más rápido, ya que utiliza los valores actualizados de las variables a medida que se calculan en cada iteración, lo cual mejora la aproximación en cada paso. Esta retroalimentación inmediata permite reducir el número de iteraciones necesarias para alcanzar un error aceptable.

La eficiencia también depende del tipo de sistema que se esté resolviendo. En este caso, el sistema de ecuaciones permitió que Gauss-Seidel mostrara un mejor rendimiento en términos de rapidez de convergencia. Además, si se tiene una buena estimación inicial y una matriz adecuadamente estructurada (por ejemplo, diagonalmente dominante), este método es muy recomendable.

Aunque el método de Bisección es más fácil de usar, no es tan eficiente en comparación con Gauss-Seidel cuando se trata de resolver sistemas de varias ecuaciones. La Bisección se enfoca en funciones univariantes y tiene una tasa de convergencia lineal, mientras que Gauss-Seidel puede tener una tasa de convergencia superior bajo buenas condiciones.

En conclusión, el método más fácil es la Bisección, ideal para funciones no lineales. El método más eficiente, especialmente para sistemas de ecuaciones, es Gauss-Seidel, debido a su rapidez de convergencia y mejor aprovechamiento de la información de iteraciones anteriores.

CONCLUSION

La realización de la Actividad 3, centrada en la aplicación de métodos numéricos como la Bisección, Jacobi y Gauss-Seidel, representa una oportunidad significativa para desarrollar competencias técnicas fundamentales que trascienden el ámbito académico y se proyectan directamente hacia el campo profesional y la vida cotidiana. Estos métodos, aunque matemáticos por naturaleza, son herramientas prácticas utilizadas en múltiples disciplinas que requieren soluciones precisas, rápidas y confiables a problemas complejos.

En el entorno laboral actual, especialmente en áreas como la ingeniería, la informática, las finanzas, la física aplicada y la investigación operativa, es común enfrentarse a situaciones en las que los métodos analíticos tradicionales no ofrecen soluciones viables o suficientemente rápidas. En estos casos, los métodos numéricos se vuelven indispensables. Por ejemplo, en el diseño de sistemas eléctricos o estructuras mecánicas, es común resolver sistemas de ecuaciones lineales de gran tamaño, y los métodos iterativos como Jacobi y Gauss-Seidel ofrecen una alternativa eficaz frente al alto costo computacional de los métodos directos.

Desde la perspectiva práctica, el dominio del método de Bisección permite encontrar raíces de funciones en procesos como el cálculo de tasas internas de retorno en proyectos financieros, el análisis de comportamiento de estructuras, o incluso en aplicaciones cotidianas como ajustar funciones en hojas de cálculo. Este tipo de habilidad resulta particularmente útil en la toma de decisiones informadas, pues permite validar modelos o simular escenarios con alto grado de precisión.

En términos de competencias laborales, esta actividad potencia el pensamiento lógico, el análisis crítico, la capacidad de modelar problemas reales y la habilidad para traducirlos en algoritmos computacionales mediante herramientas como RStudio. Estas competencias son valoradas en un mercado laboral que demanda cada vez más perfiles técnicos con capacidad de análisis cuantitativo y de programación.

Además, en un mundo donde la tecnología avanza rápidamente, contar con habilidades para implementar soluciones numéricas otorga una ventaja competitiva. Saber cuándo aplicar un método como Bisección para encontrar una raíz o cuándo utilizar Gauss-Seidel para resolver un sistema de ecuaciones no solo demuestra dominio teórico, sino también criterio técnico y adaptabilidad.