Métodos Numéricos

T6 -- E2 -- Problemario

Equipo:

Corea del Sur

Índice

Método de un paso

Método de Heun Simple

Código en Java

```java
public class MetodoHeun {

    // Función de calculo de la pendiente inicial
    public static double pendienteInicial(double x, double y) {
        return x + y; // Ejemplo: puedes cambiar esta función según tu problema
    }

    // Método de Heun simple sin corrector
    public static double[] heun(double x, double y, double h) {
        double k1 = pendienteInicial(x, y); //Pendiente inicial
        double yPredic = y + k1 * h; //Valor de Euler, predictor
        double k2 = pendienteInicial(x + h, yPredic); //Pendiente del punto predictor
        double pendientePromedio = (k1 + k2) / 2; //Pendiente promedio
        double yNuevo = y + pendientePromedio * h; //Valor nuevo para y
        x = x + h;
        return new double[] { x, yNuevo };
    }

    public static void main(String[] args) {
        // Condiciones iniciales
        double x = 0.0;
        double y = 1.0;
        double h = 0.1; // Paso

        System.out.printf("%8s %12s%n", "x", "y");
        System.out.printf("%8.2f %12.2f%n", x, y);

        double[] resultado = heun(x, y, h);
        x = resultado[0];
        y = resultado[1];
        System.out.printf("%8.2f %12.2f%n", x, y);

        // Mostrar el valor final
        System.out.printf("\nAproximación final de y en x = %.2f: %.2f%n", x, y);
    }
}
```

Pseudocódigo

```
Proceso MetodoHeunSimple

    Definir x, y, h, k1, k2, yPredic, pendientePromedio, yNuevo Como Real

    // Condiciones iniciales
    x <- 0
    y <- 1
    h <- 0.1

    Escribir "x         y"
    Escribir x, "     ", y

    // Paso 1: calcular pendiente inicial
    k1 <- x + y

    // Paso 2: calcular valor predicho de y
    yPredic <- y + h * k1

    // Paso 3: calcular pendiente en el punto predicho
    k2 <- (x + h) + yPredic

    // Paso 4: calcular la pendiente promedio
    pendientePromedio <- (k1 + k2) / 2

    // Paso 5: nuevo valor de y corregido
    yNuevo <- y + h * pendientePromedio

    // Paso 6: actualizar x e y
    x <- x + h
    y <- yNuevo

    // Mostrar resultado
    Escribir x, "     ", y
    Escribir "Aproximación final de y en x = ", x, ": ", y

FinProceso
```

Ejercicio 1

Una pequeña población de bacterias crece en un medio nutriente donde la tasa de crecimiento depende tanto del tiempo como del tamaño actual de la población. Este fenómeno puede modelarse mediante la siguiente ecuación diferencial ordinaria:

$\frac{dy}{dx} = x + y$

Donde:

- x representa el tiempo en horas,
- y representa la cantidad de bacterias (en miles),
- la derivada $\frac{dy}{dx}$ representa la tasa de cambio del tamaño de la población respecto al tiempo.

Se sabe que al inicio del experimento (cuando x=0), hay exactamente 1000 bacterias, es decir:

y(0)=1.0 (en miles)

Se desea estimar el número de bacterias al cabo de 0.1 horas utilizando un método de un paso de Heun simple con paso h=0.1

Datos de entrada:

- Ecuación: $\frac{dy}{dx} = x + y$
- Condición inicial: x0=0.0 , y0=1.0
- Paso: h=0.1h = 0.1h=0.1

x	y
0	1
0.1	1.11

El valor final que obtuviste fue:

Aproximación final de y en x = 0.10: 1.11

Esto significa que:

- Después de 0.1 horas (o 6 minutos), la población pasó de 1000 bacterias a aproximadamente 1110 bacterias.
- Esto representa un crecimiento del 11% en apenas 6 minutos, lo que refleja una aceleración en la tasa de crecimiento, como era de esperarse por la forma de la ecuación $\frac{dy}{dx} = x + y$, que es creciente tanto en x como en y.

Ejercicio 2

En una pequeña ciudad se estima que la población está creciendo de acuerdo con el tiempo y con la cantidad de personas que ya viven ahí. El modelo matemático que describe este comportamiento es:

$$\frac{dy}{dx} = x + y$$

- x es el tiempo en años desde que comenzó el estudio
- y es la población en miles de personas

La población no se midió en el año 0, se midió hasta el segundo año y en ese momento la población era de 1500 personas. Se quiere predecir cuál será la población en el tercer año, usando el método de Heun con un paso de 1 año.

Se ingresan los valores de:

- x = 2 ya que es el año en que se hizo la primera medición
- y = 1.5 es el total de población en miles (1500 habitantes)
- h = 1 ya que se quiere estimar la población en un año

x	**y**
2	1.5
3	7.25

Se nos muestra una pequeña tabla donde vienen los valores de “x” e “y”, se observa como en x = 2 se tiene que el valor de y = 1.5 ya que son los valores que se ingresaron, debajo se tiene el valor de x donde se quiere estimar la población y se obtuvo que la población en el tercer año será de 7.25 (7250 personas), lo que quiere decir que en un solo año la población incrementó 5750 personas lo cual es un crecimiento bastante grande. Esto se debe a que el crecimiento depende de la misma población: cuantas más personas hay, más rápido crece.

Ejercicio 3 (Caso de error)

Una taza de café caliente se deja enfriar en una habitación. La tasa de enfriamiento está dada por la ley de enfriamiento de Newton, que se modela como:

$$\frac{dy}{dx} = -k(y - T)$$

Donde:

- y es la temperatura del objeto (en grados Celsius),
- T es la temperatura ambiente (por ejemplo, 20°C),
- x es el tiempo en minutos,
- k es una constante de enfriamiento (por ejemplo, 0.1).

La temperatura no se midió al inicio (minuto 0), sino hasta el minuto 0 cuando la taza tenía 90 °C.

Se quiere predecir cuál será la temperatura en el minuto 1, usando el método de Heun con un paso de 1 minuto.

Datos ingresados:

```
double x = 0;

double y = 90;

double h = 1.0;
```

El resultado:

x	y
0	90°
1	225.50°

Se observa que en el minuto 0, la temperatura del café era de 90 °C, como fue introducido inicialmente. Sin embargo, al aplicar el método de Heun usando la función incorrecta $\frac{dy}{dx} = -k(y - T)$, el valor calculado para el minuto 1 fue de **225.50 °C**.

Este resultado es totalmente irreal. La temperatura de la taza no debería aumentar, y mucho menos superar los 200 °C, ya que:

- No hay una fuente de calor activa que eleve la temperatura.
- El modelo físico del problema representa un enfriamiento, no un calentamiento.
- La temperatura ambiente es de 20 °C, por lo tanto, la temperatura del café debería disminuir gradualmente hacia ese valor.

Método de varios pasos

Método Adams Bashforth Con RK4

Código en Java

```java
import java.util.ArrayList;
import java.util.List;

public class AdamsBashforthConRK4 {

    // Definición de la función f(t, y) = y
    public static double f(double t, double y) {
        return y;
    }

    public static void main(String[] args) {
        double t0 = 0.0;      // Valor inicial de t
        double y0 = 1.0;      // Valor inicial de y
        double h = 0.2;       // Tamaño de paso
        int n = 20;           // Número de pasos a calcular

        // Listas para almacenar los valores de t y y
        List<Double> tVals = new ArrayList<>();
        List<Double> yVals = new ArrayList<>();

        // Inicializar con condiciones iniciales
        tVals.add(t0);
        yVals.add(y0);

        // Usar Runge-Kutta de orden 4 para obtener los primeros 3
puntos
        for (int i = 0; i < 3; i++) {
            double t = tVals.get(tVals.size() - 1);
            double y = yVals.get(yVals.size() - 1);

            double k1 = f(t, y);
            double k2 = f(t + h / 2.0, y + h * k1 / 2.0);
            double k3 = f(t + h / 2.0, y + h * k2 / 2.0);
            double k4 = f(t + h, y + h * k3);
```

```
            double yNext = y + (h / 6.0) * (k1 + 2 * k2 + 2 * k3 +
k4);
            double tNext = t + h;

            tVals.add(tNext);
            yVals.add(yNext);
        }

        // Aplicar Adams-Bashforth de 4 pasos
        for (int i = 3; i < n; i++) {
            double tNext = tVals.get(i) + h;

            double yNext = yVals.get(i) + (h / 24.0) * (
                55 * f(tVals.get(i), yVals.get(i)) -
                59 * f(tVals.get(i - 1), yVals.get(i - 1)) +
                37 * f(tVals.get(i - 2), yVals.get(i - 2)) -
                 9 * f(tVals.get(i - 3), yVals.get(i - 3))
            );

            tVals.add(tNext);
            yVals.add(yNext);
        }

        // Mostrar resultados numéricos
        System.out.println("Resultados del método Adams-Bashforth
de 4 pasos para dy/dt = y, y(0) = 1:");
        for (int i = 0; i < tVals.size(); i++) {
            System.out.printf("t = %.2f, y = %.6f\n",
tVals.get(i), yVals.get(i));
        }

        // Mostrar comparación con la solución exacta: y = e^t
        System.out.println("\nComparación con la solución exacta
y(t) = e^t:");
        for (int i = 0; i < tVals.size(); i++) {
            double t = tVals.get(i);
            double yAprox = yVals.get(i);
            double yExacta = Math.exp(t);
            System.out.printf("t = %.2f, y_aprox = %.6f, y_exacta
= %.6f\n", t, yAprox, yExacta);
        }
    }
```

```
}
```

Pseudocódigo

```
Inicio

    Definir función f(t, y):
        Retornar y

    Definir t0 ← 0.0
    Definir y0 ← 1.0
    Definir h ← 0.2
    Definir n ← 20

    Crear lista tVals y agregar t0
    Crear lista yVals y agregar y0

    // Obtener los primeros 3 valores usando Runge-Kutta de orden 4
    Para i desde 0 hasta 2 hacer:
        t ← último valor de tVals
        y ← último valor de yVals

        k1 ← f(t, y)
        k2 ← f(t + h/2, y + h*k1/2)
        k3 ← f(t + h/2, y + h*k2/2)
        k4 ← f(t + h, y + h*k3)

        ySiguiente ← y + (h/6)*(k1 + 2*k2 + 2*k3 + k4)
        tSiguiente ← t + h

        Agregar tSiguiente a tVals
        Agregar ySiguiente a yVals
    Fin Para

    // Aplicar método de Adams-Bashforth de 4 pasos
    Para i desde 3 hasta n-1 hacer:
        tSiguiente ← tVals[i] + h

        ySiguiente ← yVals[i] + (h/24) * (
            55*f(tVals[i], yVals[i]) -
            59*f(tVals[i-1], yVals[i-1]) +
            37*f(tVals[i-2], yVals[i-2]) -
             9*f(tVals[i-3], yVals[i-3])
        )

        Agregar tSiguiente a tVals
        Agregar ySiguiente a yVals
    Fin Para
```

```
    // Mostrar resultados aproximados
    Escribir "Resultados del método Adams-Bashforth de 4 pasos para dy/dt = y, y(0) = 1:"
    Para i desde 0 hasta tamaño de tVals - 1 hacer:
        Escribir "t = ", tVals[i], ", y = ", yVals[i] con 6 decimales
    Fin Para
    // Mostrar comparación con la solución exacta y = e^t
    Escribir "Comparación con la solución exacta y(t) = e^t:"
    Para i desde 0 hasta tamaño de tVals - 1 hacer:
        t ← tVals[i]
        yAprox ← yVals[i]
        yExacta ← exp(t)
        Escribir "t = ", t, ", y_aprox = ", yAprox, ", y_exacta = ", yExacta
    Fin Para

Fin
```

Ejercicio 1

En una investigación sobre el crecimiento de un cultivo de bacterias en condiciones controladas, se ha observado que la tasa de crecimiento es directamente proporcional a la cantidad actual de bacterias. El modelo que describe este fenómeno está dado por la ecuación diferencial:

$$\frac{dy}{dt} = y,\ y(0) = 1$$

Donde:

- t representa el tiempo en días desde el inicio del experimento.
- y es la cantidad de bacterias (en millones).

Inicialmente, al comenzar el experimento, se registraron 1 millón de bacterias (y(0)=1).
El objetivo es predecir cuántas bacterias habrá después de varios días, usando un tamaño de paso de h=0.2, aplicando:

- El método de **Runge-Kutta de orden 4** para calcular los primeros tres puntos.
- El método de **Adams-Bashforth de 4 pasos** para estimar los siguientes valores.

Se implementa un programa en Java que realiza estas operaciones y se obtienen los siguientes resultados:

t (días)	y aproximado (millones)	y exacto (e^t)
0.0000	1.0000	1.0000
1.0000	2.7178	2.7182
2.0000	7.3847	7.3890
3.0000	20.0656	20.0855
4.0000	54.5218	54.5981

Ejercicio 2

Un objeto caliente se deja enfriar en una habitación cuya temperatura ambiente se mantiene constante en 20 °C. De acuerdo con la ley de enfriamiento de Newton, la velocidad con la que disminuye la temperatura del objeto es proporcional a la diferencia entre la temperatura del objeto y la del ambiente. Esto se modela con la siguiente ecuación diferencial:

$$\frac{dy}{dt} = -k(y - 20),\ y(0) = 100$$

Donde:

- y(t) es la temperatura del objeto (en °C) en el tiempo t, medido en minutos.
- k=0.07 es la constante de enfriamiento (min $^{-1}$).

Se desea estimar la temperatura del objeto en los primeros minutos, con paso $\hbar$ = 0.2, usando:

- El método de **Runge-Kutta de orden 4** para calcular los primeros 3 valores.
- El método de **Adams-Bashforth de 4 pasos** para continuar con la estimación.

El objetivo es comparar la aproximación con la solución exacta de la ecuación, que es:

$$y(t) = 20 + 80e^{-0.07t}$$

Resultados obtenidos:

t (minutos)	y aproximado (°C)	y exacto (°C)
0.0000	100.0000	100.0000
1.0000	94.5915	94.5915
2.0000	89.5486	89.5486
3.0000	84.8467	84.8467
4.0000	80.4627	80.4626

Ejercicio 3 (Caso de error)

Durante un experimento, se modela una reacción química exótica donde la tasa de cambio de la concentración de un compuesto aumenta de manera cúbica respecto a la cantidad presente. El comportamiento se modela con la siguiente ecuación diferencial:

$$\frac{dy}{dt} = y^3,\ y(0) = 1$$

Donde:

- y(t) representa la concentración de la sustancia (en mol/L).
- t es el tiempo en segundos.

Este modelo tiene una solución exacta dada por:

$$y(t) = \frac{1}{\sqrt{1-2t}}$$

que se vuelve infinita cuando t = 0.5.

Se desea resolver el problema numéricamente con un paso de $\hbar$ = 0.05, usando:

- Runge-Kutta de orden 4 para obtener los primeros 3 valores.
- Adams-Bashforth de 4 pasos para continuar la solución.

Debido a la singularidad en t=0.5t = 0.5t=0.5, el método diverge rápidamente conforme se aproxima ese valor, generando errores numéricos y resultados imposibles. Este ejercicio ilustra un caso en el que los métodos numéricos **fallan debido al mal comportamiento de la solución exacta**.

Resultados obtenidos:

t (segundos)	y aproximado (mol/L)	y exacto (mol/L)
0.0000	1.0000	1.0000
0.2000	1.2906	1.2909
0.4000	2.1922	2.2360
0.6000	434.0767	2.2360
0.9000	NaN	Infinity

Sistemas de ecuaciones diferenciales ordinarias

Método de Euler

Código en java

```
import net.objecthunter.exp4j.Expression;
import net.objecthunter.exp4j.ExpressionBuilder;

import java.util.*;

public class EulerODESolver {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 1. Ingresar tamaño del sistema
        System.out.print("Ingrese el número de ecuaciones del
sistema: ");
        int n = Integer.parseInt(scanner.nextLine());

        // 2. Nombres de variables
        System.out.println("\nIngrese los nombres de las variables
(ejemplo: x y z):");
        String[] varNames =
scanner.nextLine().trim().split("\\s+");

        if (varNames.length != n) {
            System.out.println("Error: debe ingresar exactamente "
+ n + " nombres.");
            return;
        }

        // 3. Ecuaciones
        Expression[] funciones = new Expression[n];
        System.out.println("\nIngrese las ecuaciones (use t y las
variables indicadas):");
        for (int i = 0; i < n; i++) {
            System.out.print("d" + varNames[i] + "/dt = ");
            String ecuacion = scanner.nextLine();
```

```
            // Construir expresión
            List<String> variables = new ArrayList<>();
            variables.add("t");
            variables.addAll(Arrays.asList(varNames));

            funciones[i] = new ExpressionBuilder(ecuacion)
                .variables(new HashSet<>(variables)) // ← CORREGIDO AQUÍ
                .build();

        }

        // 4. Condiciones iniciales
        double[] condicionesIniciales = new double[n];
        System.out.println("\nIngrese las condiciones iniciales:");
        for (int i = 0; i < n; i++) {
            System.out.print(varNames[i] + "(0) = ");
            condicionesIniciales[i] = Double.parseDouble(scanner.nextLine());
        }

        // 5. Parámetros de simulación
        System.out.print("\nIngrese el paso de tiempo (h): ");
        double h = Double.parseDouble(scanner.nextLine());

        System.out.print("Ingrese el tiempo final de simulación: ");
        double tFinal = Double.parseDouble(scanner.nextLine());

        int pasos = (int) Math.round(tFinal / h) + 1;
        double[][] solucion = new double[n][pasos];
        double[] tiempo = new double[pasos];

        // Inicializar
        for (int i = 0; i < n; i++) {
            solucion[i][0] = condicionesIniciales[i];
        }
        for (int i = 0; i < pasos; i++) {
            tiempo[i] = i * h;
        }
```

```
        // 6. Método de Euler
        for (int i = 1; i < pasos; i++) {
            double t = tiempo[i - 1];
            double[] anteriores = new double[n];
            for (int j = 0; j < n; j++) {
                anteriores[j] = solucion[j][i - 1];
            }

            for (int j = 0; j < n; j++) {
                Expression expr = funciones[j];
                expr.setVariable("t", t);
                for (int k = 0; k < n; k++) {
                    expr.setVariable(varNames[k], anteriores[k]);
                }
                double derivada = expr.evaluate();
                solucion[j][i] = anteriores[j] + h * derivada;
            }
        }

        // 7. Imprimir resultados
        System.out.println("\nResultados numéricos:");
        System.out.print("t\t");
        for (String var : varNames) {
            System.out.print(var + "(t)\t\t");
        }
        System.out.println();

        for (int i = 0; i < pasos; i++) {
            System.out.printf("%.2f\t", tiempo[i]);
            for (int j = 0; j < n; j++) {
                System.out.printf("%.6f\t", solucion[j][i]);
            }
            System.out.println();
        }
    }
}
```

Pseudocódigo

```
Algoritmo Euler2EDOs

    Definir pasos, i Como Entero
    Definir h, tFinal, t, dx, dy Como Real
    Definir x0, y0 Como Real

    Dimension tiempo[1000]
    Dimension solucionX[1000]
    Dimension solucionY[1000]

    Escribir "Ingrese x(0):"
    Leer x0

    Escribir "Ingrese y(0):"
    Leer y0

    Escribir "Ingrese el paso de tiempo h:"
    Leer h

    Escribir "Ingrese el tiempo final de simulación:"
    Leer tFinal

    pasos <- trunc(tFinal / h) + 1

    tiempo[0] <- 0
    solucionX[0] <- x0
    solucionY[0] <- y0

    Para i <- 1 Hasta pasos - 1
        tiempo[i] <- tiempo[i - 1] + h
        t <- tiempo[i - 1]

        // Derivadas del sistema
        dx <- solucionX[i - 1] + t
        dy <- -solucionY[i - 1] + t

        // Método de Euler
        solucionX[i] <- solucionX[i - 1] + h * dx
        solucionY[i] <- solucionY[i - 1] + h * dy
    FinPara

    // Imprimir tabla de resultados
    Escribir "t", "       x(t)      y(t)"
    Para i <- 0 Hasta pasos - 1
        Escribir tiempo[i], "    ", solucionX[i], "    ", solucionY[i]
    FinPara

FinAlgoritmo
```

Ejercicio 1

Enfriamiento de un café

En una habitación con temperatura ambiente de 22 °C, se coloca una taza de café caliente. La temperatura del café disminuye con el tiempo según la ley de enfriamiento de Newton:

$$\frac{dy}{dt} = -0.1\,(y - 22)$$

donde:

- y temperatura del café en grados Celsius.
- t : tiempo en minutos desde que comenzó el enfriamiento.

En el minuto 0, el café tiene una temperatura de 90 °C. Se desea predecir la temperatura del café cada minuto hasta el minuto 10, utilizando el método de Euler con paso de 1 minuto.

Se ingresan los valores de:

- y(0) = 90
- h = 1h
- Tiempo final = 10 minutos

Resultado esperado

t (min)	y (°C)
0	90.0000
1	83.2000
2	77.0800
3	71.5720
4	66.6148
5	62.1533
6	58.1380
7	54.5242
8	51.2718

9	48.3446
10	45.7101

Interpretación de los resultados

Se muestra una tabla donde se observa el comportamiento de la temperatura del café en función del tiempo. Se parte de un valor inicial de 90 °C al minuto 0 y se aprecia cómo la temperatura desciende progresivamente.

En el minuto 1 ya se ha enfriado a 83.2 °C y al minuto 10 alcanza los 45.71 °C. Esto indica que el café pierde calor más rápidamente cuando la diferencia con la temperatura ambiente (22 °C) es mayor, y se enfría más lentamente conforme se aproxima a dicha temperatura.

Este comportamiento es característico de un proceso de enfriamiento natural, donde la razón de cambio es proporcional a la diferencia entre la temperatura actual y la temperatura del entorno.

Ejercicio 2

Crecimiento Poblacional

Una población inicia con P0=100 individuos. Si la tasa de crecimiento (k) es 0.2 por unidad de tiempo, ¿cuál será el tamaño de la población de cada unidad de tiempo hasta el tiempo t=5? Utilice el método de Euler con un paso de tiempo de 1 unidad.

Valores a ingresar en la consola al ejecutar:

Número de ecuaciones del sistema (n): 1
Nombres de las variables: P
Ecuación para dP/dt: 0.2*P
Condición inicial para P(0): 100
Paso de tiempo (h): 1
Tiempo final de simulación: 5

Resultado esperado:

La siguiente tabla muestra la aproximación numérica de la población P en diferentes instantes de tiempo t, obtenida al ejecutar el código con los valores de entrada proporcionados.

t	P(t)
0	100.0000
1	120.0000
2	144.0000
3	172.8000
4	207.3600
5	248.8320

Interpretación de los resultados:

La tabla ilustra un crecimiento exponencial de la población, partiendo de una población inicial de 100 individuos, se observa cómo el número de individuos aumenta con cada unidad de tiempo, la tasa de crecimiento es proporcional al tamaño actual de la población, lo que significa que a medida que la población se hace más grande, el aumento en la población por unidad de tiempo también se hace mayor. Al final del tiempo de simulación (t=5), la población ha crecido a aproximadamente 248.83 individuos.

Ejercicio 3 (Caso de error)

Se desea modelar la velocidad de un objeto en caída libre, tomando en cuenta la resistencia del aire proporcional al cuadrado de la velocidad. La ecuación diferencial que representa este fenómeno es:

$$\frac{dv}{dt} = g - kv^2$$

donde:

- v(t) es la velocidad del objeto en m/s.
- $g = 9.81 \frac{m}{s^2}$ es la aceleración de la gravedad.
- k=0.25 es una constante de resistencia del aire.

Condición inicial:
Al tiempo t=0, la velocidad del objeto es v(0)=0.

Valores a ingresar en la consola al ejecutar el programa:

Número de ecuaciones del sistema (n): 1
Nombres de las variables: v
Ecuación para dv/dt: 9.81 - 0.25*v^2
Condición inicial para v(0): 0
Paso de tiempo (h): 5 ← DEMASIADO GRANDE
Tiempo final de simulación: 25

Resultado esperado (Euler con h = 5):

t (s)	v (m/s)
0	0.0000
5	49.0500
10	-511.9312
15	-31751.6723
20	-252857359.0048
25	valor indefinido o error numérico

Interpretación de los resultados:

La tabla muestra un comportamiento **físicamente incorrecto** a partir del segundo paso. En el tiempo t=5, la velocidad ya es de casi 50 m/s, lo cual es razonable, pero a partir de ahí comienza a **disminuir drásticamente** hasta volverse negativa y llegar a valores extremadamente grandes en magnitud. Este comportamiento no es coherente con un objeto en caída.

Esto se debe a que el **paso de tiempo elegido (h = 5 s)** es **demasiado grande** para este problema, y el método de Euler se vuelve **numéricamente inestable**.

Cuando la ecuación involucra términos no lineales (como v^2), se deben usar pasos pequeños para mantener la precisión y estabilidad.

Conclusión

Los métodos numéricos como Euler, Runge-Kutta de orden 4 (RK4) y Heun simple son herramientas fundamentales para resolver sistemas de ecuaciones diferenciales ordinarias (EDOs), especialmente cuando no es posible obtener soluciones analíticas.

El método de Euler, aunque sencillo y fácil de implementar, presenta errores significativos si se usa con pasos grandes, por lo que es útil principalmente para obtener una idea inicial del comportamiento de la solución.

El método de Heun simple, una mejora del método de Euler, ofrece mayor precisión al considerar un promedio de pendientes, sin aumentar mucho la complejidad computacional.

Por otro lado, el método de Runge-Kutta de orden 4 (RK4) se destaca por su alta precisión y estabilidad, siendo ampliamente utilizado en aplicaciones prácticas a pesar de requerir más cálculos por paso.

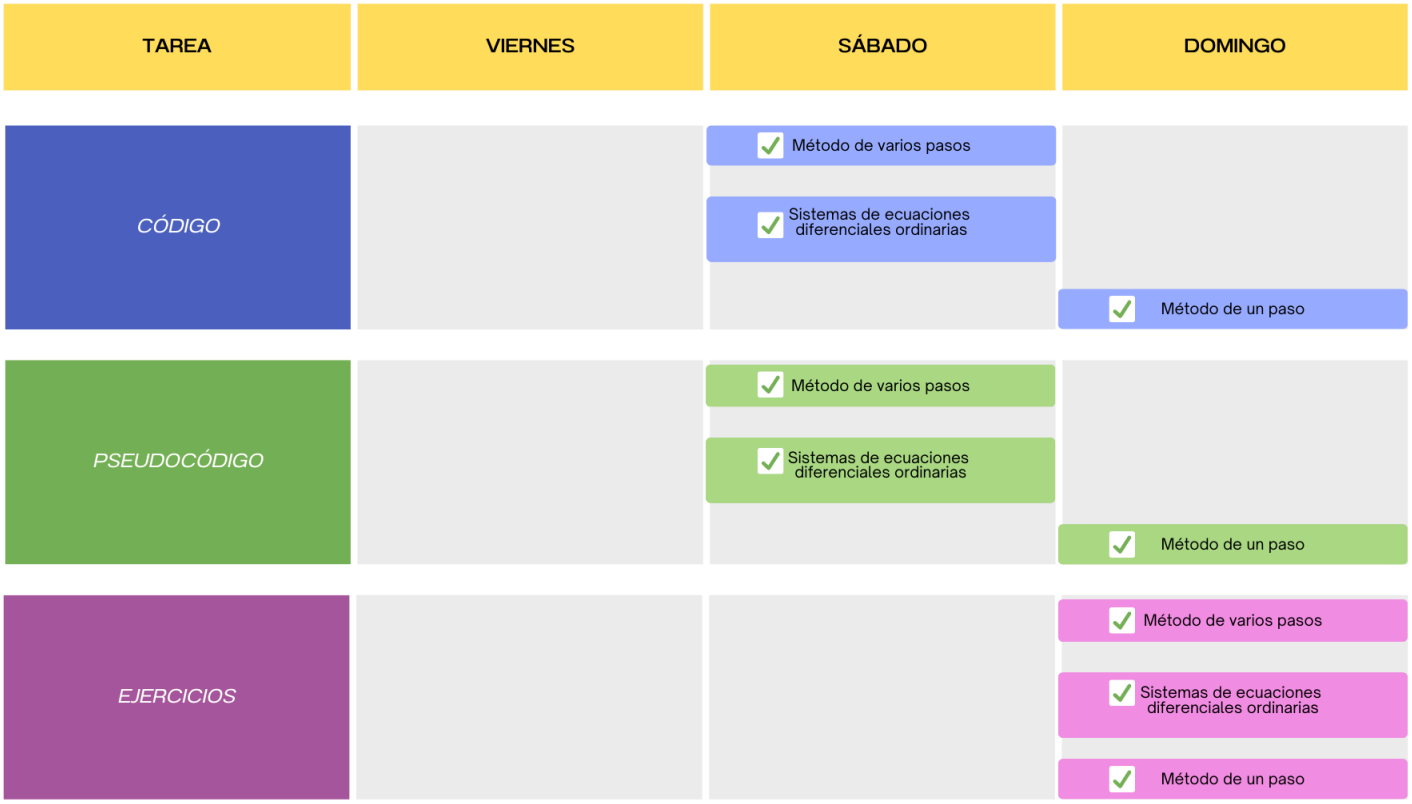
En conjunto, estos métodos permiten aproximar soluciones de sistemas dinámicos de manera eficiente, siendo su elección dependiente del equilibrio entre precisión deseada y recursos computacionales disponibles.

Bibliografía

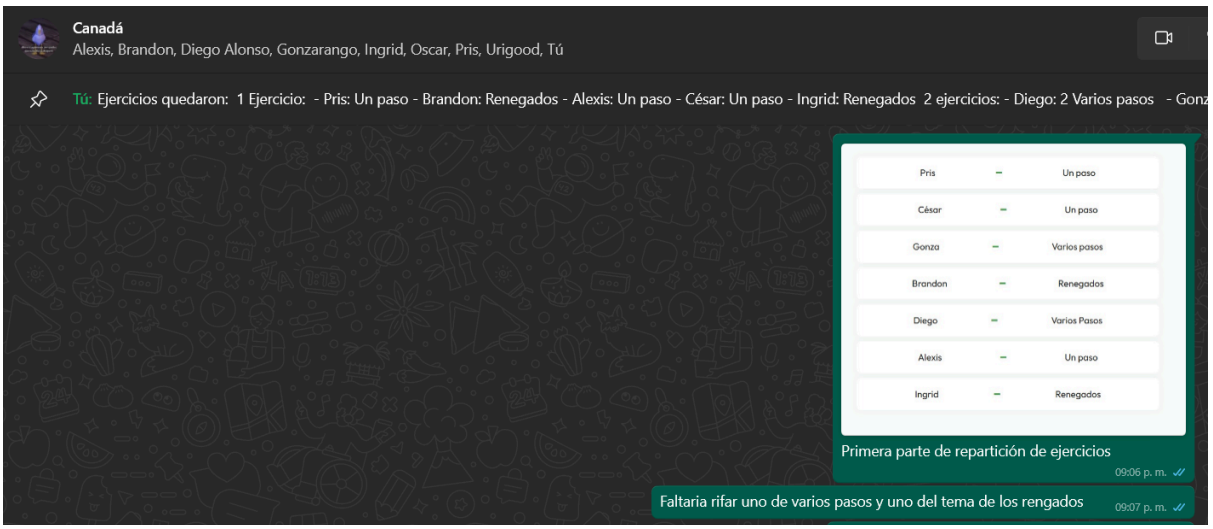
Chapra, S. C., & Canale, R. P. (2006). *Métodos numéricos para ingenieros* (J. Enríquez Brito, Trans.). McGraw-Hill.

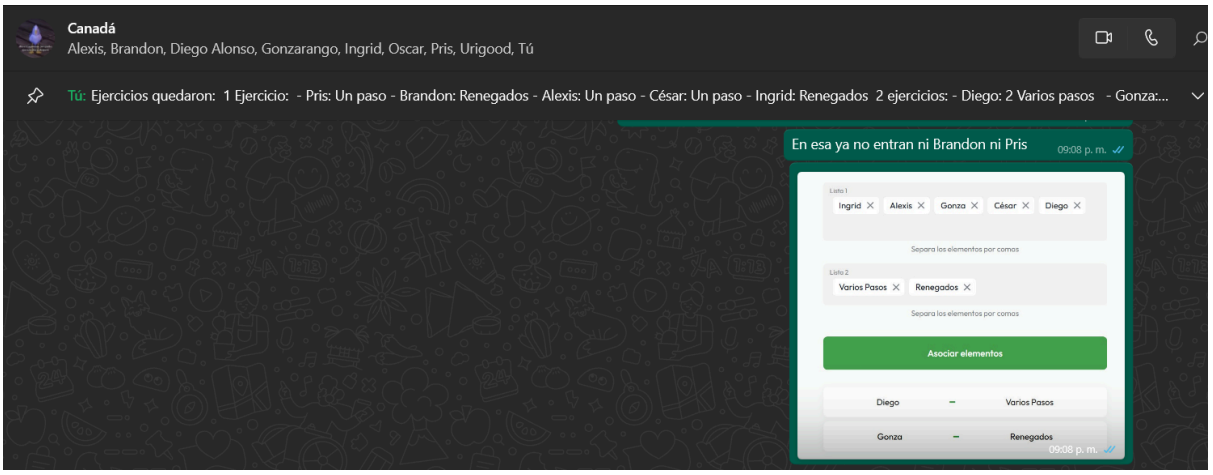
Extras

Gráfica tipo Gantt



Evidencias trabajo colaborativo





- 25 de mayo, 20:46
  Versión actual
  - ALEXIS MARTIN CORTES DE LUCIO
  - CESAR ROJAS MORENO
  - PRISCILA CORTES RAMIREZ
- 25 de mayo, 19:52
  - ANGEL GONZALEZ RAMIREZ
  - ALEXIS MARTIN CORTES DE LUCIO
  - DIEGO ALONSO CORONEL VARGAS
- 25 de mayo, 17:03
  - INGRID ORTEGA RAMOS
- 25 de mayo, 15:04
  - ALEXIS MARTIN CORTES DE LUCIO
  - CESAR ROJAS MORENO
  - BRANDON GARCIA ORDAZ
  - OSCAR AARON DELGADILLO FERNANDEZ
- 25 de mayo, 13:54
  - ALEXIS MARTIN CORTES DE LUCIO
  - CESAR ROJAS MORENO
- 25 de mayo, 12:38
  - CESAR ROJAS MORENO
- 24 de mayo, 19:43
  - CESAR ROJAS MORENO
- 24 de mayo, 18:21
  - ERNESTO URIEL GARCIA TORRES
- 24 de mayo, 17:37
  - ERNESTO URIEL GARCIA TORRES
- 24 de mayo, 10:52
  - OSCAR AARON DELGADILLO FERNANDEZ

Integrantes

Cortes De Lucio Alexis Martin

Cortes Ramirez Priscila

Coronel Vargas Diego Alonso

Delgadillo Fernandez Oscar Aaron

Garcia Ordaz Brandon

Garcia Torres Ernesto Uriel

Gonzalez Ramirez Angel

Ortega Ramos Ingrid

Rojas Moreno César