FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TP1 Modelacion Numerica

Grupo Nº	**4**
Salvador Perez Mendoza	**110198**
Juan Abdala	**112154**
Lucas Araujo	**109867**

Fecha	**Correcciones**	**Docente**

Calificación Final	Docente	Fecha

Índice

1. Introduccion

En el presente trabajo se aborda la resolución de un problema numérico relacionado con el flujo de agua en un recipiente, modelado mediante una ecuación diferencial no lineal que incluye el efecto de contracción en la salida del orificio.

El objetivo principal es aplicar y analizar dos metodos numéricos para resolver ecuaciones diferenciales ordinarias, estos metodos son Euler y Runge-Kutta de cuarto orden (RK4), para resolver la ecuación diferencial asociada al modelo. Se evaluarán los errores de truncamiento y el orden de precisión de cada método, considerando diferentes pasos de discretizacion en el tiempo (Δt).

2. Resolución del Problema Numérico con el Modelo M1 Utilizando el Método de Euler Y Runge-Kutta 4.

En este ejercicio, se resuelve un problema numérico utilizando el modelo M1 para calcular el nivel de agua en un sistema a los 10 minutos, aplicando dos métodos de integración numérica: el método de Euler y el método de Runge-Kutta de cuarto orden (RK4). Se realiza el cálculo con tres pasos de discretización diferentes: $\Delta t = 1$, $\Delta t = 50$ y $\Delta t = 100$ segundos. A partir de estos resultados, se estiman los errores de truncamiento y se evalúa el orden de precisión de cada método. Finalmente, se grafica el comportamiento de los resultados obtenidos.

Método	Δt (segundos)	Error	Relación de Errores
Euler	100	0.0028333514	0.4893702260
Euler	50	0.0013865578	
Euler	1	0.0000000000	
RK4	100	0.00000000021	0.0614396094
RK4	50	0.00000000001	
RK4	1	0.0000000000	

Figura 1: Tabla de relacion De Errores

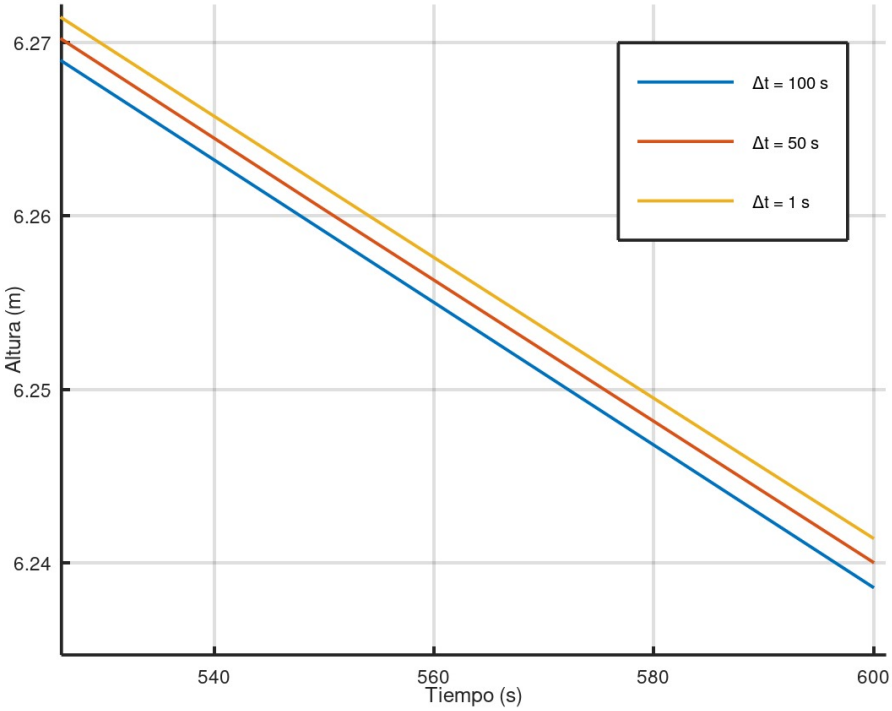

Figura 2: Diferencia de precisión entre los distintos intervalos de tiempo usando Euler

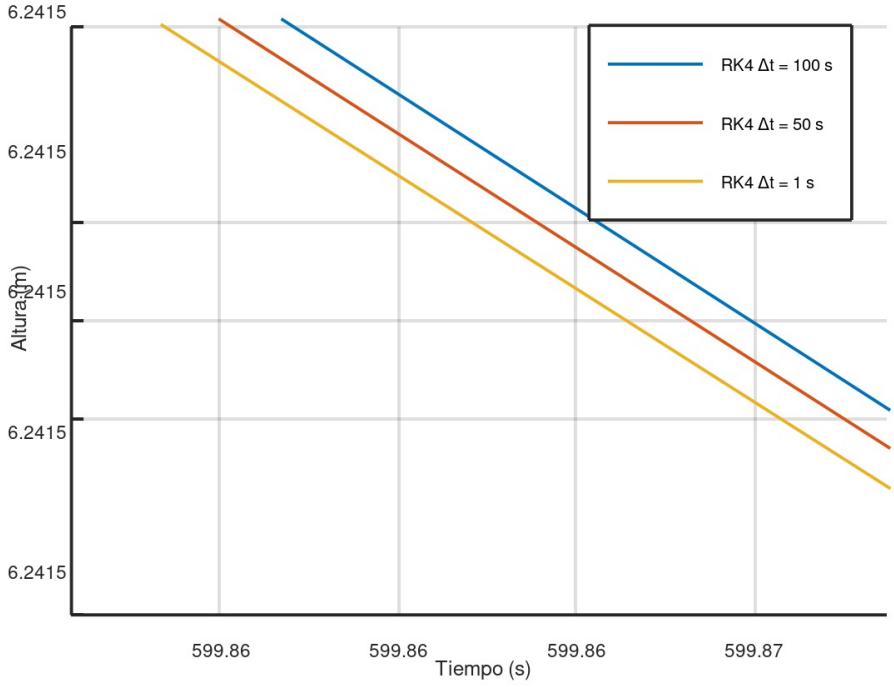

Figura 3: Diferencia de precisión entre los distintos intervalos de tiempo usando RK4

Los gráficos tienen que tener un Zoom bastante adentrado para poder ver las variaciones.

Sabiendo que la solucion converge aproximadamente en 6.2415. Los gráficos demuestran como en funcion del tiempo van convergiendo a la solucion, podemos ver cómo Euler es menos preciso y ademas le tarda mas tiempo en converger hacia la solucion. Ademas podemos notar como Euler con pasos grandes tiene un error bastante grande por lo cual requiere pasos pequeños para obtener buenos resultados, mientras que RK4 es más eficiente y preciso incluso con pasos grandes.

2.1. Analisis de los resultados

El análisis de los resultados obtenidos con los dos métodos de integración, Euler y Runge-Kutta 4 (RK4), muestra cómo varían los errores al cambiar el tamaño del paso de discretización Δt. A medida que Δt disminuye, los errores de truncamiento disminuyen de manera significativa, lo que indica una mayor precisión en los resultados.

Para el método de Euler, se observa que los errores son más grandes en comparación con el método RK4. Por ejemplo, cuando t = 100 segundos, el error es de 0.0028333514, mientras que al reducir el paso a t = 50 segundos, el error se reduce a 0.0013865578, y con un paso de $\Delta t = 1$ segundo, el error se reduce a un valor de 0.0000000000. Sin embargo, la relación de errores entre las distintas discretizaciones es 0.4893702260 (aproximadamente 0.5). Lo que significa que el orden de precision es 2, porque al duplicar el intervalo se duplica el error

Por otro lado, el método RK4 muestra errores mucho menores en comparación con Euler. Incluso con el paso de discretización más grande de $\Delta t = 100$ segundos, el error es de solo 0.00000000021, y disminuye aún más con $\Delta t = 50$ y $\Delta t = 1$, con errores cercanos a cero. La relación de errores es significativamente más baja en RK4, con un valor de 0.0614396094 (aproximadamente 0.0625) entre $\Delta t = 100$ y $\Delta t = 50$, Lo que significa que RK4 tiene un orden de precision 4 por el mismo razonamiento que hicimos con Euler.

2.2. Observaciones

En primera instancia se probaron ambos metodos con intervalos de tiempo Δt-10, Δt=5, Δt=1 pero no se registro error para el metodo RK4 por lo que aumentamos el intervalo de tiempo al ya mencionado.

3. Análisis de Sensibilidad del Modelo M2 con el Método de Runge-Kutta 4 y Comparación con los Resultados del Modelo M1

3.1. Cambios en el tamaño del paso temporal (Δt) y su impacto en la altura final ()

Δt	Altura Final (h)
$\Delta t = 1\,s$	6.3406282896
$\Delta t = 50\,s$	6.3406282896
$\Delta t = 100\,s$	6.3406282894

Figura 4: Sensibilidad al Cambio en Δt

La altura final (h) apenas varía con cambios en Δt, lo cual indica que el método de Runge-Kutta 4 mantiene una alta precisión y estabilidad respecto al paso temporal. La diferencia entre los valores es insignificante (casi 0), evidenciando la estabilidad del método para diferentes valores de Δt.

3.2. Cambios en la constante C_c y su impacto en la altura final ()

Cambio en cc	Altura Final (h)	Diferencia (Δh)
$cc = 0.55$	6.3533919800	-
$cc = 0.60$	6.3406282896	0.0127636904
$cc = 0.65$	6.3274939652	0.0126789244

Figura 5: Sensibilidad al Cambio en C_c

La altura final disminuye consistentemente con el aumento de C_c, indicando una relación inversa clara. Los valores de h son muy similares entre los intervalos (ΔCc=0.05), lo que evidencia una sensibilidad uniforme dentro del rango analizado. El cambio promedio en h por incremento de C_c es de aproximadamente 0.0127, reflejando una respuesta estable y predecible del modelo. La estabilidad en Δh garantiza que los efectos de los cambios en C_c sean pequeños.

Codigo

```
// Este es el main que simplemente define las variables y se comunica con el resto de las f

function Main()
  % Parámetros del problema
  g = 9.81;  % m/s^2 (gravedad)
  R = 4.0;   % m (radio del tanque)
  r = 0.02;  % m (radio del orificio)
  h0 = 6.5;  % m (altura inicial del agua)
  t_max = 10 * 60;  % 10 minutos en segundos
  pasos_dt = [10, 5, 1];  % Pasos de discretización
  cc_values = [0.55, 0.6, 0.65];  % Coeficientes de contracción para M2

  % a) Resolver con Euler (M1)
  fprintf('\n==== Parte a: Método de Euler (M1) ====\n');
  resultados_Euler = zeros(1, length(pasos_dt));
  for i = 1:length(pasos_dt)
    resultados_Euler(i) = Euler(h0, pasos_dt(i), t_max, r, g, R);
    fprintf('t = %d s -> Altura final (Euler): %.10f\n', pasos_dt(i), resultados_Euler(i))
  end

  % b) Resolver con RK4 (M1)
  fprintf('\n==== Parte b: Método de Runge-Kutta 4 (M1) ====\n');
  resultados_RK4 = zeros(1, length(pasos_dt));
  for i = 1:length(pasos_dt)
    resultados_RK4(i) = RK4(h0, pasos_dt(i), t_max, r, g, R);
    fprintf('t = %d s -> Altura final (RK4): %.10f\n', pasos_dt(i), resultados_RK4(i));
  end

  % c) Verificar el orden de precisión
  fprintf('\n==== Parte c: Verificación del orden de precisión ====\n');
  fprintf('Errores relativos entre pasos de tiempo en RK4:\n');
  CalcularConvergencia(resultados_RK4, pasos_dt);

  % d) Análisis de sensibilidad 1 (M2 con RK4 y comparar con M1)
  fprintf('\n==== Parte d: Sensibilidad M2 con RK4 ====\n');
  resultados_M2 = zeros(1, length(pasos_dt));
  for i = 1:length(pasos_dt)
```

```
    resultados_M2(i) = MK2(h0, pasos_dt(i), t_max, r, g, R, 0.6);  % cc = 0.6
    fprintf('t = %d s -> Altura final (M2): %.10f\n', pasos_dt(i), resultados_M2(i));
  end
  fprintf('\nComparación entre M1 (RK4) y M2 (RK4):\n');
  for i = 1:length(pasos_dt)
    diferencia = abs(resultados_RK4(i) - resultados_M2(i));
    fprintf('t = %d s -> Diferencia: %.10f\n', pasos_dt(i), diferencia);
  end

  % e) Análisis de sensibilidad 2 (Variando cc para M2)
  fprintf('\n==== Parte e: Sensibilidad M2 a cc ====\n');
  for cc = cc_values
    resultado = MK2(h0, 1, t_max, r, g, R, cc);  % t = 1 fijo
    fprintf('cc = %.2f -> Altura final (M2): %.10f\n', cc, resultado);
  end
  fprintf('\nSensibilidad de M2 al cambio en cc:\n');
  for i = 2:length(cc_values)
    delta_h = abs(MK2(h0, 1, t_max, r, g, R, cc_values(i)) - MK2(h0, 1, t_max, r, g, R, cc_
    fprintf('Cambio cc = %.2f a cc = %.2f -> h = %.10f\n', cc_values(i-1), cc_values(i), de
  end
end


function h_final = Euler(h0, dt, t_max, r, g, R)
  n_steps = floor(t_max / dt);
  h = h0;

  for step = 1:n_steps

    if h > 0
      % H(n+1) = Hn + dt * F(tn, Hn) No hay tn y queda "Lineal"
      h = h + dt * (-(r^2 * sqrt(2 * g * h)) / (2 * h * R - h^2));
    else
      h = 0;  % El agua no puede ser negativa
      break;
    end
  end

  h_final = h;
```

```
end

function h_final = RK4(h0, dt, t_max, r, g, R)
  % Modelo M1 con RK4
  n_steps = floor(t_max / dt);
  h = h0;

  for step = 1:n_steps
    k1 = -(r^2 * sqrt(2 * g * h)) / (2 * h * R - h^2);
    h_mid1 = h + 0.5 * dt * k1;
    k2 = -(r^2 * sqrt(2 * g * h_mid1)) / (2 * h_mid1 * R - h_mid1^2);
    h_mid2 = h + 0.5 * dt * k2;
    k3 = -(r^2 * sqrt(2 * g * h_mid2)) / (2 * h_mid2 * R - h_mid2^2);
    h_next = h + dt * k3;
    k4 = -(r^2 * sqrt(2 * g * h_next)) / (2 * h_next * R - h_next^2);

    h = h + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);

    if h < 0
      h = 0;
      break;
    end
  end

  h_final = h;
end

function CalcularConvergencia(resultados, variacionesDeTiempo)
  % Comparar errores con respecto al paso más fino
  supuestaSolucionExacta = resultados(end);  % Resultado con t más pequeño
  errores = abs(resultados - supuestaSolucionExacta);

  % Mostrar errores
  for i = 1:length(variacionesDeTiempo)
    fprintf('t = %d s -> Error: %.10f\n', variacionesDeTiempo(i), errores(i));
  end

  % Relación de convergencia
  if length(variacionesDeTiempo) >= 3
```

```
    error_grande = errores(1);  % Error para t más grande
    error_medio = errores(2);   % Error para t intermedio
    if error_medio > 0
      ratio = error_medio / error_grande ;
      fprintf('Relación de Errores: %.10f\n', ratio);
    else
      fprintf('No se puede calcular la relación de convergencia (error medio es 0).\n');
    end
  end
end

function AnalisisSensibilidad1()

    % Parámetros del problema
    g = 9.81;  % m/s^2 (gravedad)
    R = 4.0;   % m (radio del tanque)
    r = 0.02;  % m (radio del orificio)
    h0 = 6.5;  % m (altura inicial del agua)
    t_max = 10 * 60;  % 10 minutos en segundos
    variacionesDeTiempo = [10, 5, 1];  % Diferentes pasos de tiempo
    c_c_M1 = 1.0;  % Modelo M1: Sin contracción
    c_c_M2 = 0.6;  % Modelo M2: Con contracción

    % Matrices para resultados
    resultados_M1 = zeros(1, length(variacionesDeTiempo));
    resultados_M2 = zeros(1, length(variacionesDeTiempo));

    % Método RK4 generalizado
    function h_final = RK4(h0, dt, t_max, r, g, R, c_c)
        n_steps = floor(t_max / dt);
        h = h0;

        for step = 1:n_steps
            k1 = -(r^2 * c_c * sqrt(2 * g * h)) / (2 * h * R - h^2);
            k2 = -(r^2 * c_c * sqrt(2 * g * (h + 0.5 * dt * k1))) / (2 * (h + 0.5 * dt * k1
            k3 = -(r^2 * c_c * sqrt(2 * g * (h + 0.5 * dt * k2))) / (2 * (h + 0.5 * dt * k2
            k4 = -(r^2 * c_c * sqrt(2 * g * (h + dt * k3))) / (2 * (h + dt * k3) * R - (h +

            h = h + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
```

```
            if h < 0
                h = 0;  % El agua no puede ser negativa
                break;
            end
        end
        h_final = h;
    end

    % Simulaciones para ambos modelos
    for i = 1:length(variacionesDeTiempo)
        dt = variacionesDeTiempo(i);
        resultados_M1(i) = RK4(h0, dt, t_max, r, g, R, c_c_M1);  % Modelo M1
        resultados_M2(i) = RK4(h0, dt, t_max, r, g, R, c_c_M2);  % Modelo M2
    end

    % Mostrar resultados
    fprintf('Resultados Modelo M1 (RK4):\n');
    disp(resultados_M1);
    fprintf('Resultados Modelo M2 (RK4):\n');
    disp(resultados_M2);

    % Comparar resultados
    fprintf('Diferencias entre Modelo M1 y Modelo M2:\n');
    diferencias = abs(resultados_M1 - resultados_M2);
    for i = 1:length(variacionesDeTiempo)
        fprintf('t = %d s -> Diferencia: %.10f m\n', variacionesDeTiempo(i), diferencias(i)
    end

end

function h_final = MK2(h0, dt, t_max, r, g, R, cc)
  % Modelo M2 con RK4 y coeficiente de contracción
  n_steps = floor(t_max / dt);
  h = h0;

  for step = 1:n_steps
    if h <= 0
      h = 0;
```

```
      break;
    end

    k1 = -(r^2 * cc * sqrt(2 * g * h)) / (2 * h * R - h^2);
    h_mid1 = h + 0.5 * dt * k1;
    k2 = -(r^2 * cc * sqrt(2 * g * h_mid1)) / (2 * h_mid1 * R - h_mid1^2);
    h_mid2 = h + 0.5 * dt * k2;
    k3 = -(r^2 * cc * sqrt(2 * g * h_mid2)) / (2 * h_mid2 * R - h_mid2^2);
    h_next = h + dt * k3;
    k4 = -(r^2 * cc * sqrt(2 * g * h_next)) / (2 * h_next * R - h_next^2);

    h = h + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
  end

  h_final = h;
end
```