Resumen

Este documento presenta un análisis detallado de dos métodos numéricos prominentes en la resolución de ecuaciones diferenciales ordinarias (EDO): el método de Euler y el método de Runge-Kutta de segundo orden (RK2). El método de Euler, conocido por su simplicidad y eficiencia computacional, se basa en la aproximación lineal de la función y es ideal para situaciones donde la precisión extrema no es crítica. Por otro lado, el método RK2, una técnica más avanzada, ofrece una mayor precisión y estabilidad al considerar no solo la pendiente en el punto inicial sino también una pendiente intermedia. Este estudio proporciona una formulación matemática de ambos métodos y discute sus ventajas, destacando la simplicidad y eficiencia del método de Euler frente a la precisión mejorada y la estabilidad del método RK2. El análisis no busca comparar directamente estos métodos, sino más bien proporcionar una comprensión profunda de cada uno, facilitando así la elección adecuada del método en función de los requisitos específicos del problema a resolver.
Palabras clave: Euler, Runge-Kutta de 2° orden, Ecuaciones Diferenciales Ordinarias, Ecuaciones Diferenciales

Método de Euler y Runge-Kutta 2°

Enrique Ulises Báez Gómez Tagle
Mauricio Iván Ascencio Martínez

21 de noviembre de 2023

1. Introducción

El método de Euler y el método de Runge-Kutta de segundo orden son técnicas numéricas fundamentales para resolver ecuaciones diferenciales ordinarias (EDO). A continuación, se presenta una descripción detallada de cada uno de estos métodos, destacando sus características y ventajas.

2. Método de Euler

El método de Euler es uno de los algoritmos más simples y antiguos para la resolución numérica de ecuaciones diferenciales ordinarias. Se basa en la aproximación lineal de la función, utilizando la tangente en un punto conocido para predecir el valor de la función en un punto siguiente. [3]

2.1. Formulación matemática

Dada una EDO de la forma $\frac{dy}{dt} = f(t, y)$ con una condición inicial $Y(t_0) = Y_0$, el método de Euler propone la siguiente fórmula iterativa para aproximar Y en puntos sucesivos:

$$Y_{n+1} = Y_n + hf(t_n, Y_n) \tag{1}$$

donde h es el tamaño del paso y Y_{n+1} es la aproximación de Y en $t_n + 1 = t_n + h$. [9]

2.2. Análisis de Error

2.2.1. Error Local de Truncamiento (ELT)

El Error Local de Truncamiento en el método de Euler se refiere a la diferencia entre la solución real y la solución numérica después de un solo paso. Matemáticamente, se expresa como:

$$ELT = |y(t_{n+1}) - (y_n + hf(t_n, y_n))| \tag{2}$$

donde $y(t_{n+1})$ es el valor real de la solución en t_{n+1}, y_n es la aproximación actual, y $f(t_n, y_n)$ es la derivada en el punto actual. Este error es proporcional al cuadrado del tamaño del paso h. [8]

2.2.2. Error Global de Truncamiento (EGT)

El Error Global de Truncamiento es la acumulación del ELT en cada paso durante todo el proceso de integración. Se define como:

$$EGT = |y(t_{final}) - y_n| \tag{3}$$

donde $y(t_{final})$ es el valor real en el tiempo final y y_n es la aproximación numérica. El EGT es proporcional a h y tiende a aumentar con el número de pasos. [6]

2.2.3. Estabilidad Numérica

La estabilidad numérica en el método de Euler está relacionada con la elección del tamaño del paso h. Un h demasiado grande puede llevar a inestabilidades, especialmente en ecuaciones diferenciales con comportamientos rápidamente cambiantes. Se puede demostrar que para ciertos problemas, el método es condicionalmente estable, lo que significa que la estabilidad depende del tamaño del paso y de la naturaleza de la función $f(t, y)$. [7]

2.3. Ventajas

- Simplicidad: Su mayor ventaja es su simplicidad conceptual y facilidad de implementación.
- Requiere Pocos Recursos Computacionales: Ideal para problemas donde la precisión no es crítica o los recursos computacionales son limitados.
- Fundamento para Métodos Más Complejos: Sirve como base para comprender algoritmos más avanzados.

2.4. Elaboración del código

Para este método, generamos distintas versiones que se detallan a continuación:

2.5. Primera versión: Tamaño de Paso Fijo

En la primera versión, el tamaño del paso h es un parámetro que se pasa directamente a la función. Esto permite controlar la precisión de la aproximación con mayor detalle. Con un h menor, se generaría una solución más precisa, pero requerirá más cálculos. Esta versión es útil cuando se necesita un control específico sobre la precisión de cada paso individual. Código:

```
1 def euler_method(f, y0, t0, tf, h):
2     """
3     Implementa el metodo de Euler para resolver
          ecuaciones
4     diferenciales ordinarias.
5
```

```
    Args:
    f (function): La funcion derivada, f(t, y), de la ecuacion
    diferencial dy/dt = f(t, y).
    y0 (float): Valor inicial de la variable dependiente (y) en
    el tiempo t0.
    t0 (float): Tiempo inicial.
    tf (float): Tiempo final hasta donde se resolvera la ecuacion.
    h (float): Tamano del paso para la integracion numerica.

    Returns:
    list of tuples: Una lista de tuplas (t, y) representando la
    solucion aproximada en diferentes tiempos.
    """

    # Inicializar variables
    t = t0  # tiempo actual
    y = y0  # valor actual de y

    # Lista para almacenar la solucion
    solution = [(t, y)]

    # Bucle principal: se ejecuta mientras el tiempo actual sea
    menor o igual al tiempo final
    while t <= tf:
        y = y + h * f(t, y)  # Aplicar el metodo de Euler para
        calcular el siguiente valor de y
        t = t + h  # Incrementar el tiempo en un paso
        solution.append((t, y))  # Anadir la nueva pareja (t, y) a la lista de solucion

    return solution

# Supongamos que queremos resolver la ecuacion diferencial  dy/dt = -2y con y(0) = 1 desde t=0 hasta t=5 con un paso h=0.1

def f(t, y):
        """
        Define la funcion diferencial dy/dt.

        Args:
        t (float): Variable independiente (tiempo).
```

```
        y (float): Variable dependiente.

        Returns:
        float: Valor de la derivada en el punto (t, y).
        """
        return -2 * y + 4 * t

# Llamamos al metodo de Euler con los parametros correspondientes
euler_solution = euler_method(f, 1.0, 0.0, 5.0, 0.1)

# Imprimir la solucion
for t, y in euler_solution:
    print(f"t = {t:.1f}, y = {y:.4f}")
```

2.6. Segunda Versión: Número de Pasos Fijo

En la segunda versión, en lugar de especificar el tamaño del paso, se define el número total de pasos n para cubrir el intervalo de tiempo desde t_0 hasta t_n. El tamaño del paso se calcularía automáticamente en función del número de pasos. Esta versión es útil cuando necesitamos controlar el número total de evaluaciones de la función, lo que podría ser importante en contextos donde las evaluaciones de la función son costosas o limitadas.

```
def euler_method(f, y0, t0, tn, n):
    """
    Implementa el metodo de Euler para resolver ecuaciones diferenciales ordinarias.

    Esta version del metodo de Euler utiliza un numero fijo de pasos n entre los tiempos inicial y final.

    Args:
    f (function): La funcion derivada, f(t, y), de la ecuacion diferencial dy/dt = f(t, y).
    y0 (float): Valor inicial de la variable dependiente (y) en el tiempo t0.
    t0 (float): Tiempo inicial.
    tn (float): Tiempo final hasta donde se resolvera la ecuacion.
    n (int): Numero de pasos para la integracion numerica.

    Returns:
    float: Valor aproximado de y en el tiempo final tn.
    """
```

```
    h = (tn - t0) / n  # Calcular el tamano del paso
    t = t0  # Inicializar el tiempo
    y = y0  # Inicializar el valor de y

    # Bucle que se ejecuta n veces
    for i in range(n):
        y += h * f(t, y)  # Actualizar el valor de y
            utilizando el metodo de Euler
        t += h  # Incrementar el tiempo

    return y  # Devolver el valor aproximado de y en
        el tiempo final

def f(t, y):
    """
    Define la funcion diferencial dy/dt.

    Args:
    t (float): Variable independiente (tiempo).
    y (float): Variable dependiente.

    Returns:
    float: Valor de la derivada en el punto (t, y).
    """
    return -2 * y + 4 * t

# Condiciones iniciales y parametros
y0 = 1  # Valor inicial de y
t0 = 0  # Tiempo inicial
tn = 2  # Tiempo final
n = 10  # Numero de pasos

# Llamada al metodo de Euler
approximation = euler_method(f, y0, t0, tn, n)
print("Aproximacion de y(t) en t =", tn, "es",
    approximation)
```

3. Método de Runge-Kutta de Segundo Orden (RK2)

El método de Runge-Kutta de segundo orden es una técnica más precisa que el método de Euler para resolver EDOs. Ofrece una mejor aproximación al considerar no solo la pendiente inicial sino también una pendiente intermedia. [4]

3.1. Formulación matemática

La formulación general de RK2 para la EDO $\frac{dy}{dt} = f(t, y)$ es:

$$Y_{n+1} = Y_n + hK_2K_1 = f(t_n, y_n)K_2 = f(t_n, \frac{h}{2}, y_n + \frac{h}{2}K_1) \qquad (4)$$

aquí K_1 es la pendiente en el punto inicial, y K_2 es la pendiente en el punto medio estimado. [2]

3.2. Análisis de error

3.2.1. Error Local de Truncamiento

El método RK2 mejora el Error Local de Truncamiento utilizando una estimación intermedia. Para un paso dado, el ELT en RK2 se puede expresar como:

$$ELT = |y(t_{n+1}) - (y_n + hK_2)| \qquad (5)$$

donde $K_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1)$ y $K_1 = f(t_n, y_n)$. A pesar de que el cálculo del ELT en RK2 es más complejo que en Euler, generalmente resulta en un error menor para el mismo tamaño de paso. [8]

3.2.2. Comparación con Euler

Al comparar RK2 con el método de Euler, se observa que RK2 ofrece una mayor precisión para un mismo tamaño de paso. Esto se debe al uso de la pendiente intermedia K_2, que proporciona una mejor aproximación de la función en el intervalo. En términos de Error Global de Truncamiento, RK2 también tiende a superar a Euler, especialmente en problemas donde las derivadas de la función cambian rápidamente.

3.2.3. Estabilidad Numérica

El método RK2 muestra una mejor estabilidad numérica en comparación con el método de Euler. Esto es particularmente notable en problemas donde Euler puede ser inestable debido al tamaño del paso. RK2, al utilizar la pendiente intermedia, ofrece una aproximación más precisa y, por lo tanto, puede manejar mejor las inestabilidades inherentes a ciertas EDOs. [1]

3.3. Ventajas

- Mayor Precisión: Ofrece mayor precisión que el método de Euler para el mismo tamaño de paso.
- Estabilidad Mejorada: Tiende a ser más estable, especialmente para problemas donde el método de Euler muestra inestabilidad.
- Buen Compromiso: Representa un buen balance entre la complejidad computacional y la precisión.

3.4. Elaboración del código

Para este método, diseñamos el siguiente código:

```python
def rk2_method(f, y0, t0, tn, n):
    """
    Implementa el metodo de Runge-Kutta de segundo orden (RK2) para resolver ecuaciones diferenciales ordinarias.

    Args:
    f (function): La funcion derivada, f(t, y), de la ecuacion diferencial dy/dt = f(t, y).
    y0 (float): Valor inicial de la variable dependiente (y) en el tiempo t0.
    t0 (float): Tiempo inicial.
    tn (float): Tiempo final hasta donde se resolvera la ecuacion.
    n (int): Numero de pasos para la integracion numerica.

    Returns:
    float: Valor aproximado de y en el tiempo final tn.
    """
    h = (tn - t0) / n  # Calcular el tamano del paso
    t = t0  # Inicializar el tiempo
    y = y0  # Inicializar el valor de y

    # Bucle que se ejecuta n veces
    for i in range(n):
        k1 = f(t, y)  # Calcular el primer incremento basado en la pendiente en el punto actual
        k2 = f(t + 0.5 * h, y + 0.5 * h * k1)  # Calcular el segundo incremento, basado en la pendiente en el punto medio
        y += h * k2  # Actualizar y usando el segundo incremento (K2)
        t += h  # Incrementar el tiempo

    return y  # Devolver el valor aproximado de y en el tiempo final

def f(t, y):
    """
    Define la funcion diferencial dy/dt.

    Args:
    t (float): Variable independiente (tiempo).
    y (float): Variable dependiente.
```

```
35
36      Returns:
37      float: Valor de la derivada en el punto (t, y).
38      """
39      return -2 * y + 4 * t
40
41  # Condiciones iniciales y parametros
42  y0 = 1  # Valor inicial de y
43  t0 = 0  # Tiempo inicial
44  tn = 2  # Tiempo final
45  n = 10  # Numero de pasos
46
47  approximation = rk2_method(f, y0, t0, tn, n)
48  print("Aproximacion de y(t) en t =", tn, "es",
        approximation)
```

4. Análisis de resultados

Esta sección presenta ejemplos específicos que ilustran la aplicación de los métodos de Euler y RK2, destacando las diferencias en términos de precisión y error.

4.1. Comparación de Soluciones Exactas vs. Aproximadas

Consideremos la ecuación diferencial $\frac{dy}{dt} = -2y + 4t$ con la condición inicial $y(0) = 1$. La solución analítica de esta ecuación es conocida, lo que nos permite compararla con las soluciones numéricas obtenidas a través de los métodos de Euler y RK2.

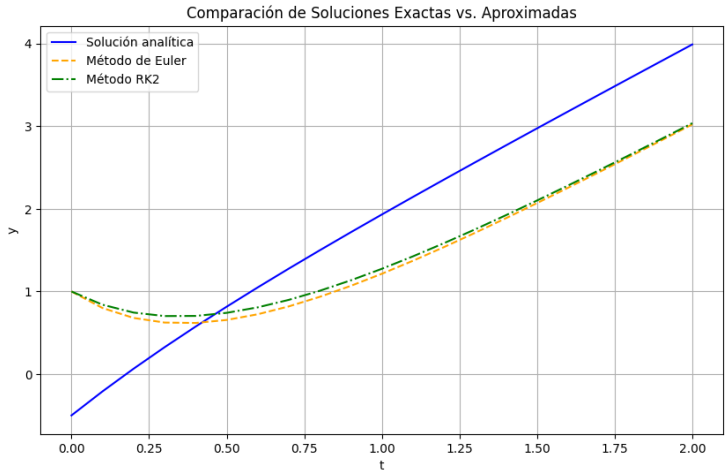

Figura 1: Comparación de las soluciones exactas y aproximadas obtenidas mediante los métodos de Euler y RK2.

En el gráfico proporcionado, se observa que:

- La línea azul representa la solución analítica de la ecuación diferencial.
- La línea naranja discontinua muestra la solución aproximada obtenida mediante el método de Euler.
- La línea verde discontinua muestra la solución aproximada obtenida mediante el método RK2.

Se puede concluir que el método RK2 tiende a estar más cerca de la solución analítica que el método de Euler, especialmente a medida que el tiempo avanza. Esto ilustra la mayor precisión del método RK2 en comparación con el método de Euler para un mismo tamaño de paso.

4.2. Gráficos de Error

Los siguientes gráficos muestran cómo varía el error, tanto el ELT como el EGT, para diferentes tamaños de paso en ambos métodos. Se observa claramente que el método RK2 generalmente ofrece una mayor precisión para un mismo tamaño de paso comparado con el método de Euler.

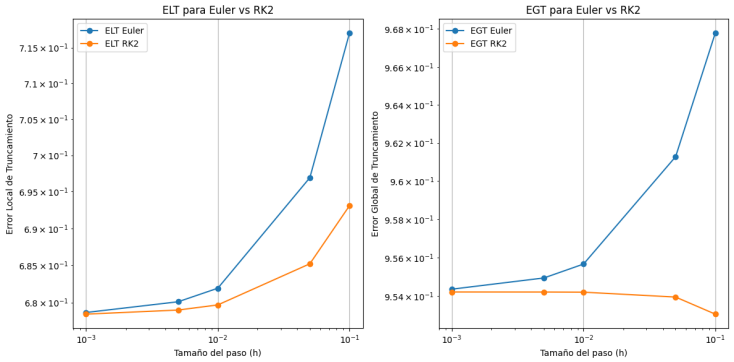

Figura 2: Comparación del Error Local y Global de Truncamiento para los métodos de Euler y RK2.

En los gráficos proporcionados, se observan las siguientes tendencias:

- El Error Local de Truncamiento (ELT) es consistentemente menor para el método RK2 en comparación con el método de Euler para todos los tamaños de paso probados.
- La disminución del ELT en ambos métodos es más pronunciada a medida que se reduce el tamaño del paso, siendo más significativa en el método RK2.
- En cuanto al Error Global de Truncamiento (EGT), el método RK2 también muestra un error significativamente menor que el método de Euler en todos los tamaños de paso.
- Estos resultados sugieren que el método RK2 es más adecuado para aplicaciones que requieren alta precisión en la solución final de la ecuación diferencial.

Estos hallazgos ilustran la importancia de la selección del método numérico en función de los requisitos de precisión del problema. Mientras que el método de Euler puede ser adecuado para aplicaciones con requisitos menos estrictos de precisión, el método RK2 es preferible para situaciones que demandan mayor exactitud.

4.3. Visualización de la Estabilidad Numérica

Para ilustrar la estabilidad numérica de ambos métodos, consideremos una ecuación diferencial donde Euler tiende a ser inestable. Observaremos cómo el tamaño del paso afecta la estabilidad de las soluciones en ambos métodos.

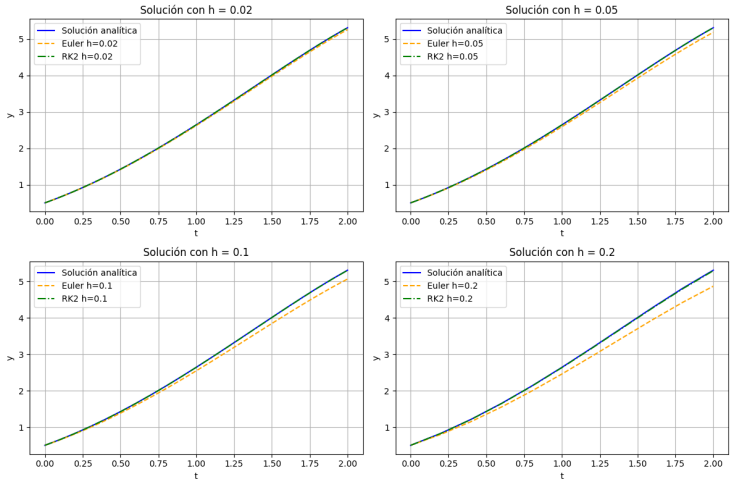

Figura 3: Estabilidad numérica de los métodos de Euler y RK2 en una ecuación diferencial donde Euler es inestable.

En los gráficos proporcionados, se observan las siguientes tendencias:

- Para tamaños de paso más pequeños, ambos métodos tienden a ser más estables y las soluciones se acercan más a la solución analítica.
- El método de Euler muestra una tendencia a ser inestable, especialmente con tamaños de paso más grandes.
- El método RK2 muestra una mayor estabilidad en comparación con el método de Euler, manteniendo las soluciones más cercanas a la solución analítica incluso para tamaños de paso más grandes.
- Estos resultados resaltan la importancia de considerar la estabilidad numérica al elegir un método para resolver ecuaciones diferenciales.

Estos hallazgos ilustran la superioridad del método RK2 en términos de estabilidad numérica, especialmente en situaciones donde se espera inestabilidad.

4.4. Discusión

Los ejemplos y gráficos presentados anteriormente ofrecen una comparación valiosa entre los métodos de Euler y RK2 en el contexto de la resolución numérica de ecuaciones diferenciales ordinarias. Estos ejemplos ilustran no solo las diferencias fundamentales en términos de precisión y estabilidad, sino también cómo estos aspectos pueden influir en la elección del método adecuado para una situación dada.

4.4.1. Implicaciones de la Precisión y la Estabilidad

La mayor precisión del método RK2, evidenciada en los ejemplos, sugiere su idoneidad para problemas donde los errores pequeños son críticos. Por otro lado, la simplicidad del método de Euler puede ser suficiente

en situaciones donde se requiere menos precisión o donde los recursos computacionales son un factor limitante. Además, la estabilidad mejorada de RK2 lo hace más confiable en situaciones con soluciones que varían rápidamente o donde el tamaño del paso es relativamente grande.

4.4.2. Consideraciones Prácticas en la Elección del Método

La elección entre el método de Euler y RK2 no debería basarse únicamente en la precisión y la estabilidad. Factores como la complejidad del problema, la facilidad de implementación, y los recursos computacionales disponibles también juegan un papel crucial. En la práctica, el método de Euler a menudo se utiliza para obtener una solución rápida y aproximada, que luego puede refinarse utilizando métodos más precisos como RK2.

4.4.3. Relevancia en Aplicaciones Reales

En aplicaciones del mundo real, especialmente en ingeniería y ciencias físicas, la elección del método numérico adecuado puede tener implicaciones significativas en la calidad y confiabilidad de los resultados. Por ejemplo, en la modelización de sistemas dinámicos o en la simulación de procesos físicos, la precisión y estabilidad ofrecidas por RK2 pueden ser esenciales para obtener resultados válidos y útiles.

4.4.4. Conclusión

En resumen, mientras que el método de Euler ofrece una solución simple y computacionalmente menos intensiva, el método RK2 se destaca por su mayor precisidad y estabilidad. La elección entre estos métodos debe considerar la naturaleza del problema específico, las necesidades de precisión y estabilidad, y las limitaciones de recursos. La comprensión profunda de ambos métodos, como se ha ilustrado a través de estos ejemplos, es fundamental para tomar decisiones informadas en la resolución de ecuaciones diferenciales en diversos campos de aplicación.

Referencias

[1] Uri M. Ascher y Linda R. Petzold. “Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations”. En: (1998).

[2] Kendall E. Atkinson. *An Introduction to Numerical Analysis*. 2.ª ed. John Wiley Sons, 1989.

[3] Richard L. Burden y J. Douglas Faires. *Numerical Analysis*. 9.ª ed. Brooks/Cole, Cengage Learning, 2010.

[4] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008.

[5] Stephen L. Campbell y C. William Gear. “The index of general nonlinear DAES”. En: *Numer. Math.* 72.2 (1995), págs. 173-196.

[6] Ernst Hairer, Syvert P. Nørsett y Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 1993.

[7] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 2009.

[8] Alfio Quarteroni. *Numerical Models for Differential Problems*. 2.ª ed. Springer, 2010.

[9] Timothy Sauer. *Numerical Analysis*. 2.ª ed. Pearson, 2012.