

MA0501 – Tarea 6

Diego Alberto Vega Viquez - C38367 José Carlos Quintero Cedeño - C26152
Gabriel Valverde Guzmán - C38060

2025-11-08

Tabla de contenidos

Ejercicio 1	2
Ejercicio 2	4
Ejercicio 3	12
Ejercicio 4	23
Ejercicio 5	24
Ejercicio 6	29
Ejercicio 7	34
Ejercicio 8	39

Ejercicio 1

i Instrucción del ejercicio 1

Implemente en **R** funciones para todos los algoritmos numéricos de resolución de ecuaciones diferenciales vistos en clase.

Solución

Metodo de Euler

```
metodo.euler <- function(a, b, N, alfa, f){  
  h <- (b - a) / N  
  t <- a  
  w <- alfa  
  T <- numeric(N + 1)  
  W <- numeric(N + 1)  
  T[1] <- t  
  W[1] <- w  
  for (i in 2:(N+1)) {  
    W[i] <- W[i-1] + h*f(T[i-1],W[i-1])  
    T[i] <- t + (i - 1)*h  
  }  
  
  tabla <- data.frame(t = T, w = W)  
  
  return(list(t = T, w = W, tabla = tabla))  
}
```

Metodo de Euler predictor-corrector

```
metodo.euler.predictor.corrector <- function(a, b, N, alfa, f){  
  h <- (b - a) / N  
  t <- a  
  w <- alfa  
  T <- numeric(N + 1)  
  W <- numeric(N + 1)  
  T[1] <- t  
  W[1] <- w
```

```

for (i in 2:(N+1)) {
  W[i] <- W[i-1] + h*f(T[i-1],W[i-1]) #predictor
  T[i] <- T[i - 1] + h
  W[i] <- W[i-1] + (h/2)*(f(T[i-1], W[i-1]) + f(T[i], W[i])) #corrector
}

tabla <- data.frame(t = T, w = W)

return(list(t = T, w = W, tabla = tabla))
}

```

Metodo de Runge-Kutta de punto medio

```

runge.kutta.punto.medio <- function(a, b, N, alfa, f){
  h <- (b - a) / N
  t <- a
  w <- alfa
  T <- numeric(N + 1)
  W <- numeric(N + 1)
  T[1] <- t
  W[1] <- w
  for (i in 2:(N+1)) {
    W[i] <- W[i-1] + h*f(T[i-1] + (h/2),W[i-1] + (h/2)*f(T[i-1], W[i-1])) #predictor
    T[i] <- T[i - 1] + h
  }

  tabla <- data.frame(t = T, w = W)

  return(list(t = T, w = W, tabla = tabla))
}

```

Mertodo de Runge-Kutta de cuarto orden

```

runge.kutta.cuarto.orden <- function(a, b, N, alfa, f){
  h <- (b - a) / N
  t <- a
  w <- alfa

```

```

T <- numeric(N + 1)
W <- numeric(N + 1)
T[1] <- t
W[1] <- w
for (i in 2:(N+1)) {
  k1 <- h*f(T[i-1], W[i-1])
  k2 <- h*f(T[i-1] + h/2, W[i-1] + k1/2)
  k3 <- h*f(T[i-1] + h/2, W[i-1] + k2/2)
  k4 <- h*f(T[i-1] + h, W[i-1] + k3)

  W[i] <- W[i-1] + (1/6)*(k1 + 2*k2 + 2*k3 + k4)
  T[i] <- T[i - 1] + h
}

tabla <- data.frame(t = T, w = W)

return(list(t = T, w = W, tabla = tabla))
}

```

Ejercicio 2

i Instrucción del ejercicio 2

Complete los detalles de las demostraciones que quedaron pendientes en este capítulo.

Solución

i Lema 3

Sea (ξ_j) una sucesión en \mathbb{R} con la propiedad

$$|\xi_{j+1}| \leq (1 + A) |\xi_j| + B, \quad j = 0, 1, 2, \dots,$$

para algunas constantes $A > 0$ y $B \geq 0$. Entonces, para todo $j \geq 0$ se cumple

$$|\xi_j| \leq |\xi_0| e^{jA} + \frac{B}{A}(e^{jA} - 1).$$

Prueba

Sea $a_j := |\xi_j| \geq 0$. La hipótesis dice

$$a_{j+1} \leq (1 + A) a_j + B, \quad j \geq 0.$$

Paso 1 (desenrollando la recurrencia).

Probamos por inducción que para todo $j \geq 0$,

$$a_j \leq (1 + A)^j a_0 + B \sum_{k=0}^{j-1} (1 + A)^k. \quad (1)$$

- Para $j = 0$, la afirmación es trivial: $a_0 \leq (1 + A)^0 a_0 + 0$.
- Suponga que vale para j . Entonces, usando la recurrencia,

$$\begin{aligned} a_{j+1} &\leq (1 + A) a_j + B \\ &\leq (1 + A) \left[(1 + A)^j a_0 + B \sum_{k=0}^{j-1} (1 + A)^k \right] + B \\ &= (1 + A)^{j+1} a_0 + B \sum_{k=1}^j (1 + A)^k + B \\ &= (1 + A)^{j+1} a_0 + B \sum_{k=0}^j (1 + A)^k, \end{aligned}$$

que es la fórmula (1) con j reemplazado por $j + 1$. Queda probado por inducción.

De (1) y la suma geométrica,

$$\sum_{k=0}^{j-1} (1 + A)^k = \frac{(1 + A)^j - 1}{A},$$

obtenemos

$$a_j \leq (1 + A)^j a_0 + \frac{B}{A} ((1 + A)^j - 1). \quad (2)$$

Paso 2 (paso de $(1 + A)^j$ a e^{Aj}).

Usando que $(1 + A)^j \leq e^{Aj}$ para $A > 0$ (por $1 + x \leq e^x$), en (2) se concluye

$$a_j \leq a_0 e^{Aj} + \frac{B}{A} (e^{Aj} - 1).$$

Recordando que $a_j = |\xi_j|$ y $a_0 = |\xi_0|$, queda

$$|\xi_j| \leq |\xi_0| e^{jA} + \frac{B}{A} (e^{jA} - 1), \quad j = 0, 1, 2, \dots$$

i Teorema 6

Si la función φ en un método de un paso es continua (con respecto a h) y satisface la **condición de Lipschitz**

$$|\varphi(x, u; h) - \varphi(x, v; h)| \leq M |u - v|,$$

para todo $(x, u), (x, v) \in G$ y para h suficientemente pequeño,
entonces **un método de un paso es convergente si y solo si el método es consistente.**

Prueba

1. (Consistencia \Rightarrow Convergencia)

Sea $e_j = u_j - u(x_j)$ el error global en el nodo x_j .

Se tiene que:

$$\begin{aligned} e_{j+1} - e_j &= [u_{j+1} - u_j] - [u(x_{j+1}) - u(x_j)] \\ &= h \varphi(x_j, u_j; h) - [u(x_{j+1}) - u(x_j)] \\ &= h [\varphi(x_j, u_j; h) - \varphi(x_j, u(x_j); h) - \Delta(x_j, u(x_j); h)]. \end{aligned}$$

Por la condición de Lipschitz, se obtiene:

$$|e_{j+1} - e_j| \leq h [M |u_j - u(x_j)| + c(h)], \quad (1.14)$$

donde

$$c(h) := \max_{a \leq x \leq b} |\Delta(x, u(x); h)|.$$

Nótese que $c(h) \rightarrow 0$ cuando $h \rightarrow 0$, pues el método es consistente.

2. (Aplicación del Lema 3)

Como $e_j = u_j - u(x_j)$, la desigualdad (1.14) implica que:

$$|e_{j+1}| \leq (1 + hM) |e_j| + h c(h), \quad j = 0, 1, 2, \dots, n.$$

Aplicando el **Lema 3** con $A = hM$, $B = h c(h)$ y observando que $e_0 = 0$, se deduce:

$$|e_j| \leq \frac{h c(h)}{hM} (e^{jhM} - 1), \quad j = 0, 1, 2, \dots, n.$$

Como $x_j = x_0 + jh$, esto se puede reescribir como:

$$|e_j| \leq \frac{c(h)}{M} (e^{M(x_j - x_0)} - 1), \quad j = 0, 1, 2, \dots, n. \quad (1.15)$$

Por tanto, el **error global máximo** cumple:

$$E(h) = \max_j |e_j| \leq \frac{c(h)}{M} (e^{M(b-a)} - 1) \rightarrow 0 \quad \text{cuando } h \rightarrow 0.$$

Esto prueba que el método es **convergente**.

3. (Convergencia \Rightarrow Consistencia)

Recordemos (Def. 7) que el **error de discretización local** en un punto (x, u) se define por

$$\Delta(x, u; h) = \frac{\eta(x+h) - \eta(x)}{h} - \varphi(x, u; h),$$

donde η es la solución del problema auxiliar $\eta' = f(\xi, \eta)$ con condición inicial $\eta(x) = u$.

Además, por el Teorema 3, un método de un paso es **consistente** si y sólo si

i Corolario 2

1. El **método de Euler** es convergente y, si f es continuamente diferenciable, entonces el **orden de convergencia** es 1.
2. El **método de Euler modificado** (o de Heun) es convergente y, si f es dos veces continuamente diferenciable, entonces el **orden de convergencia** es 2.

Prueba

Para 1. (Euler clásico).

El método de Euler tiene

$$\varphi(x, u; h) = f(x, u).$$

- **Lipschitz en u :** Si f es Lipschitz en su segunda variable con constante L (uniforme en x), entonces

$$|\varphi(x, u; h) - \varphi(x, v; h)| = |f(x, u) - f(x, v)| \leq L |u - v|.$$

- **Consistencia y orden:** Por el **Teorema 4** (ya demostrado), Euler es consistente y, si f es C^1 , su **orden de consistencia es 1**.

Aplicando el **Teorema 6** (consistencia \iff convergencia bajo Lipschitz) y el **Teorema 7** (el orden de convergencia coincide con el orden de consistencia), se concluye que Euler **converge con orden 1**.

Para 2. (Euler modificado / Heun).

En este caso

$$\varphi(x, u; h) = \frac{1}{2} [f(x, u) + f(x + h, u + h f(x, u))].$$

- **Lipschitz en u :** Suponga que f es Lipschitz en su segunda variable con constante L (uniforme en x). Entonces, para u, v cualesquiera,

$$\begin{aligned} |\varphi(x, u; h) - \varphi(x, v; h)| &\leq \frac{1}{2} |f(x, u) - f(x, v)| + \frac{1}{2} |f(x + h, u + h f(x, u)) - f(x + h, v + h f(x, v))| \\ &\leq \frac{1}{2} L |u - v| + \frac{1}{2} L |[u - v] + h [f(x, u) - f(x, v)]| \\ &\leq \frac{1}{2} L |u - v| + \frac{1}{2} L (|u - v| + h L |u - v|) \\ &= L \left(1 + \frac{1}{2} h L\right) |u - v|. \end{aligned}$$

Así, φ es Lipschitz en u con constante $M = L(1 + \frac{1}{2} h L)$ (válida para h pequeño).

- **Consistencia y orden:** Por el resultado demostrado para Euler mejorado, si $f \in C^2$ entonces el método es **consistente de orden 2**.

Aplicando de nuevo el **Teorema 6** (bajo Lipschitz hay equivalencia entre consistencia y convergencia) y el **Teorema 7** (el orden de convergencia coincide con el de consistencia), se concluye que el método de Euler modificado **converge con orden 2**.

■

i Definición 11

El método de **Runge–Kutta de cuarto orden (RK4)** para resolver el problema de valor inicial:

$$\begin{cases} u' = f(x, u), \\ u(x_0) = u_0, \end{cases}$$

construye una aproximación u_j de $u(x_j)$ en la malla de puntos equidistantes $x_j := x_0 + jh$ para $j = 1, 2, \dots$, usando las siguientes ecuaciones:

$$\begin{aligned} k_1 &= f(x_j, u_j), \\ k_2 &= f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}k_1\right), \\ k_3 &= f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}k_2\right), \\ k_4 &= f(x_j + h, u_j + hk_3), \\ u_{j+1} &= u_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

! Observaciones

1. El método fue introducido por **Runge** en 1895 y extendido por **Kutta** en 1901 para sistemas de ecuaciones diferenciales.
2. Si $u' = f(x)$, el método de Runge–Kutta y la **regla de Simpson** son equivalentes (pruébelo).

🔥 Prueba

Sea $u' = f(x)$, es decir, $u(x)$ es una primitiva de $f(x)$. Entonces, la solución exacta cumple:

$$u(x_{j+1}) = u(x_j) + \int_{x_j}^{x_{j+1}} f(x) dx.$$

El método de Runge–Kutta de orden 4 estima u_{j+1} mediante:

$$\begin{aligned} k_1 &= f(x_j), \\ k_2 &= f\left(x_j + \frac{h}{2}\right), \\ k_3 &= f\left(x_j + \frac{h}{2}\right), \\ k_4 &= f(x_j + h), \\ u_{j+1} &= u_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

Observamos que como $u' = f(x)$, la evaluación de f no depende de u_j , por lo tanto:

$$k_2 = k_3 = f\left(x_j + \frac{h}{2}\right).$$

Sustituyendo:

$$\begin{aligned} u_{j+1} &= u_j + \frac{h}{6} \left[f(x_j) + 2f\left(x_j + \frac{h}{2}\right) + 2f\left(x_j + \frac{h}{2}\right) + f(x_j + h) \right] \\ &= u_j + \frac{h}{6} \left[f(x_j) + 4f\left(x_j + \frac{h}{2}\right) + f(x_j + h) \right]. \end{aligned}$$

Esto coincide exactamente con la **regla de Simpson** para la integral de f en el intervalo $[x_j, x_{j+1}]$, que aproxima:

$$\int_{x_j}^{x_{j+1}} f(x) dx \approx \frac{h}{6} \left[f(x_j) + 4f\left(x_j + \frac{h}{2}\right) + f(x_j + h) \right].$$

Luego:

$$u_{j+1} = u_j + \int_{x_j}^{x_{j+1}} f(x) dx \approx u_j + (\text{Simpson}) = \text{RK4}.$$

3. Resumiendo, el método de **Runge–Kutta** se puede escribir como:

Ejercicio 3

i Instrucción del ejercicio 3

Para los problemas de valor inicial:

$$y' = -2ty^2, \quad y(0) = 1, \quad 0 \leq t \leq 1.$$

$$y' - y = \cos(t), \quad y(0) = \frac{1}{2}, \quad 0 \leq t \leq 1.$$

$$x\sqrt{1-y^2}dx + y\sqrt{1-x^2}dy = 0, \quad y(0) = 1, \quad 0 \leq x \leq 1.$$

- (a) Genere una tabla con cada uno de los métodos del ejercicio anterior con $N = 10$.
- (b) Encuentre la solución exacta usando **R**.
- (c) Mediante alguno de los métodos de interpolación vistos en el curso, interpole cada una de las soluciones obtenidas. Luego grafique en un mismo plano la solución exacta y el polinomio obtenido.
- (d) ¿Cuál de los métodos permitió obtener una mejor solución aproximada?

Solución

```
library(deSolve)
```

a) Problema 1:

```
p1.euler <- metodo.euler(0, 1, 10, 1, function(t, y) -2*t*y^2)
p1.euler.pc <- metodo.euler.predictor.corrector(0, 1, 10, 1, function(t, y) -2*t*y^2)
p1.rkpm <- runge.kutta.punto.medio(0, 1, 10, 1, function(t, y) -2*t*y^2)
p1.rk4o <- runge.kutta.cuarto.orden(0, 1, 10, 1, function(t, y) -2*t*y^2)

data.frame(nodos = p1.euler$t, euler = p1.euler$w, euler.pred.corrector = p1.euler.pc$w, runge.kutta.punto.medio = p1.rkpm$w, runge.kutta.cuarto.orden = p1.rk4o$w)
```

	nodos	euler	euler.pred.corrector	runge.kutta.punto.medio
1	0.0	1.0000000	1.0000000	1.0000000
2	0.1	1.0000000	0.9900000	0.9900000

3	0.2	0.9800000	0.9613656	0.9611763
4	0.3	0.9415840	0.9172458	0.9167422
5	0.4	0.8883892	0.8619543	0.8611044
6	0.5	0.8252503	0.8000340	0.7988875
7	0.6	0.7571465	0.7355270	0.7341797
8	0.7	0.6883540	0.6715871	0.6701446
9	0.8	0.6220177	0.6103988	0.6089524
10	0.9	0.5601127	0.5532891	0.5519050
11	1.0	0.5036420	0.5009186	0.4996377

runge.kutta.cuarto.orden

1	1.0000000
2	0.9900989
3	0.9615381
4	0.9174306
5	0.8620682
6	0.7999992
7	0.7352935
8	0.6711406
9	0.6097561
10	0.5524865
11	0.5000006

Problema 2:

```
p2.euler <- metodo.euler(0, 1, 10, 1/2, function(x, y) cos(x) + y)
p2.euler.pc <- metodo.euler.predictor.corrector(0, 1, 10, 1/2, function(x, y) cos(x) + y)
p2.rkpm <- runge.kutta.punto.medio(0, 1, 10, 1/2, function(x, y) cos(x) + y)
p2.rk4o <- runge.kutta.cuarto.orden(0, 1, 10, 1/2, function(x, y) cos(x) + y)

data.frame(nodos = p2.euler$t, euler = p2.euler$w, euler.pred.corrector = p2.euler.pc$w, runge.kutta.punto.medio = p2.rkpm$w, runge.kutta.cuarto.orden = p2.rk4o$w)
```

	nodos	euler	euler.pred.corrector	runge.kutta.punto.medio
1	0.0	0.5000000	0.5000000	0.5000000
2	0.1	0.6500000	0.6572502	0.6573750
3	0.2	0.8145004	0.8299900	0.8302515
4	0.3	0.9939571	1.0188095	1.0192195
5	0.4	1.1888865	1.2243810	1.2249515

6	0.5	1.3998812	1.4474785	1.4482214
7	0.6	1.6276276	1.6889976	1.6899251
8	0.7	1.8729239	1.9499779	1.9511023
9	0.8	2.1367005	2.2316272	2.2329611
10	0.9	2.4200413	2.5353475	2.5369038
11	1.0	2.7242064	2.8627626	2.8645551

runge.kutta.cuarto.orden

1	0.5000000
2	0.6575854
3	0.8307038
4	1.0199502
5	1.2260027
6	1.4496418
7	1.6917710
8	1.9534389
9	2.2358637
10	2.5404593
11	2.8688634

Problema 3:

```
p3.euler <- metodo.euler(0, 1, 10, 1, function(t, y)- ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) )
p3.euler.pc <- metodo.euler.predictor.corrector(0, 1, 10, 1, function(t, y)- ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) )
p3.rkpm <- runge.kutta.punto.medio(0, 1, 10, 1, function(t, y)- ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) )
p3.rk4o <- runge.kutta.cuarto.orden(0, 1, 10, 1, function(t, y)- ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) )

data.frame(nodos = p3.euler$t, euler = p3.euler$w, euler.pred.corrector = p3.euler.pc$w, runge.kutta.punto.medio = p3.rkpm$w, runge.kutta.cuarto.orden = p3.rk4o$w)
```

	nodos	euler	euler.pred.corrector	runge.kutta.punto.medio
1	0.0	1	1	1
2	0.1	1	1	1
3	0.2	1	1	1
4	0.3	1	1	1
5	0.4	1	1	1
6	0.5	1	1	1
7	0.6	1	1	1
8	0.7	1	1	1

9	0.8	1	1	1
10	0.9	1	1	1
11	1.0	1	1	1

```
runge.kutta.cuarto.orden
```

1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

b y c) Problema 1:

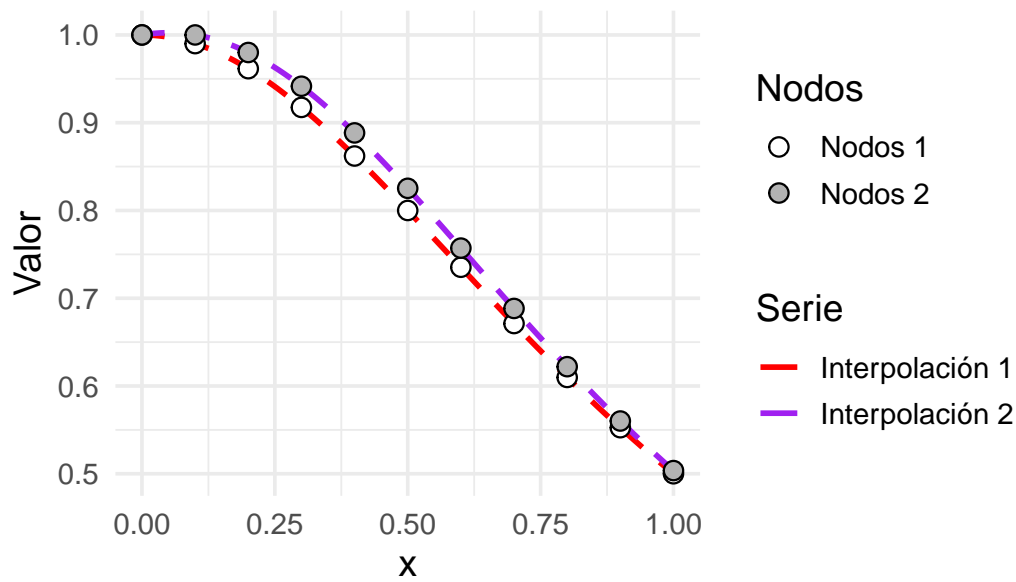
```
edo <- function(t, y, parms) {
  dydt <- -2 * t * y^2
  list(dydt)
}

y0 <- 1
tiempos <- seq(0, 1, by = 0.1)

sol <- ode(y = y0, times = tiempos, func = edo, parms = NULL)
```

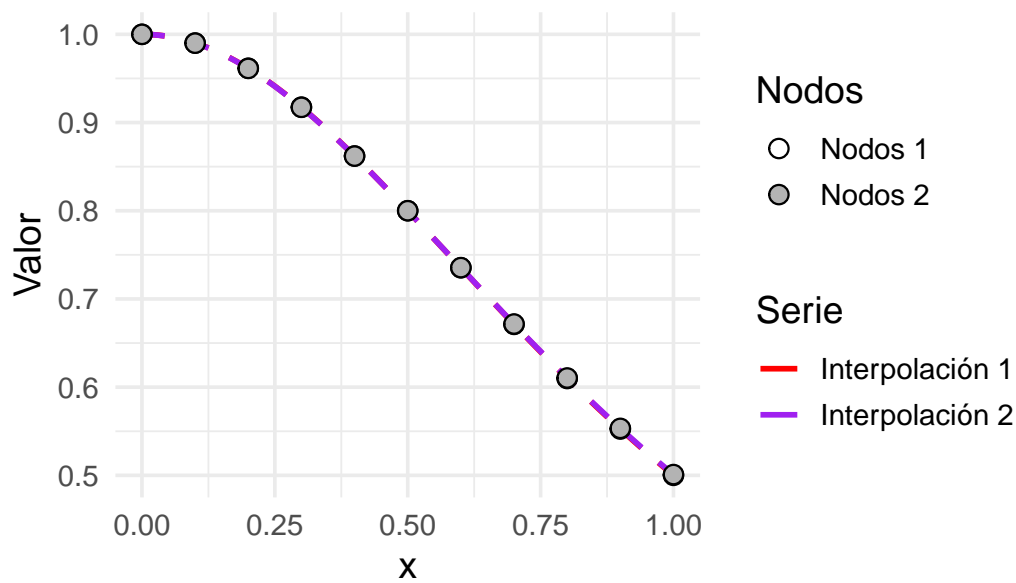
```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p1.euler$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metod



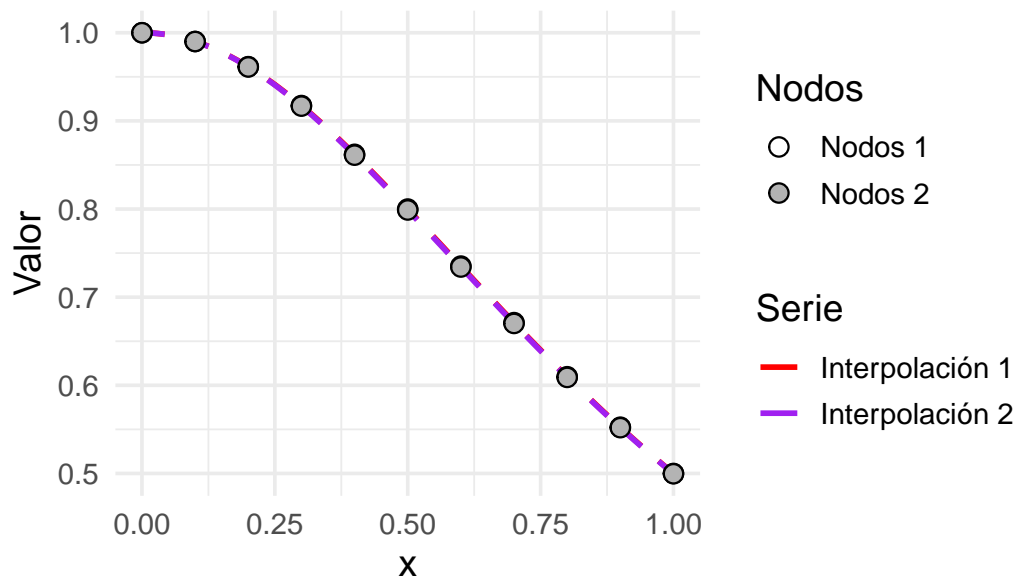
```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p1.euler.pc$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metod



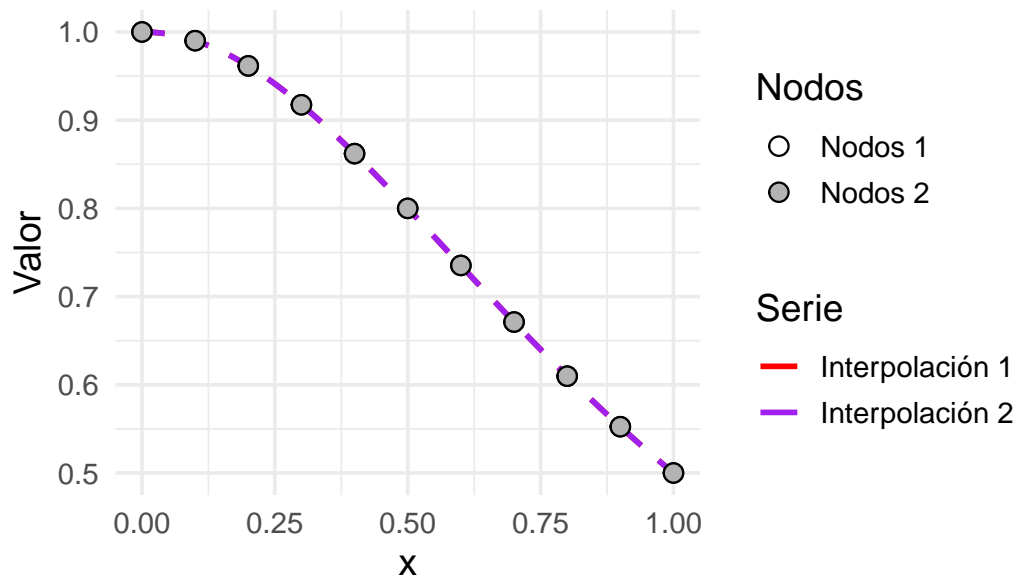
```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p1.rkpm$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metod



```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p1.rk4o$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metod



Problema 2:

```
edo <- function(t, y, parms) {
  dydt <- cos(t) + y
  list(dydt)
}
```

```

y0 <- 1/2
tiempos <- seq(0, 1, by = 0.1)

sol <- ode(y = y0, times = tiempos, func = edo, parms = NULL)

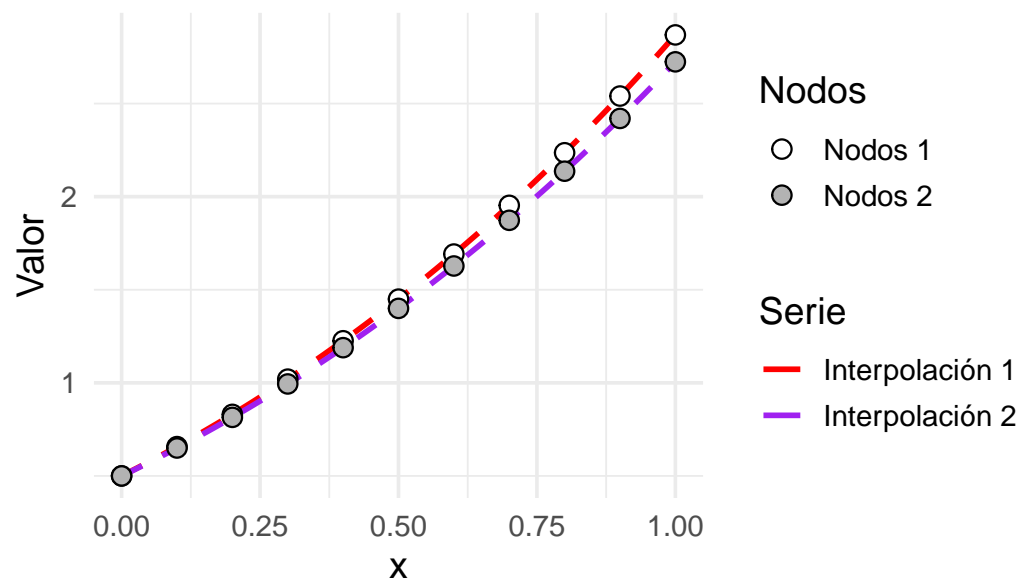
```

```

graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p2.euler$t, valores2 =

```

Interpolación 1: neville | Interpolación 2: metodo

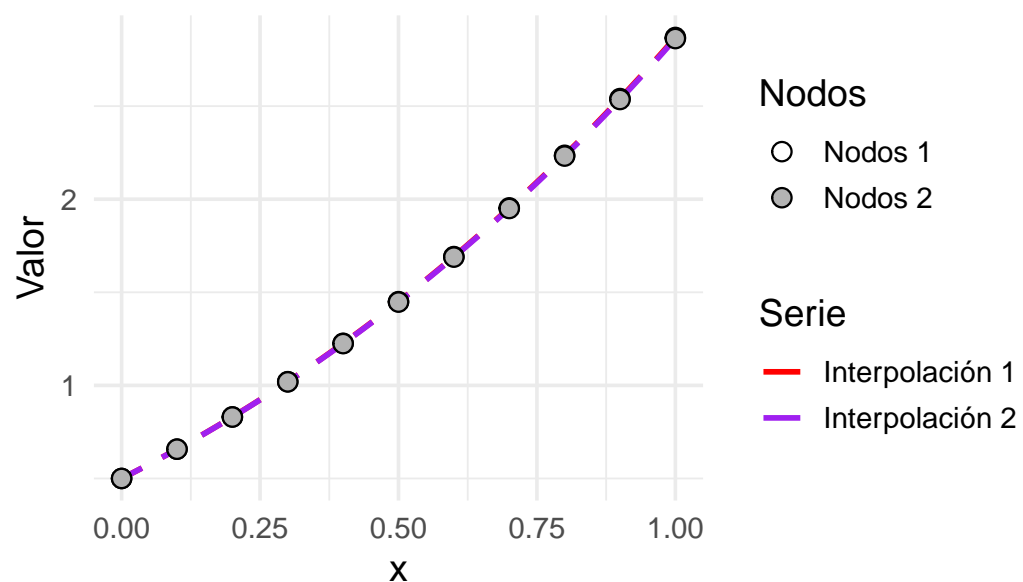


```

graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p2.euler.pc$t, valores2 =

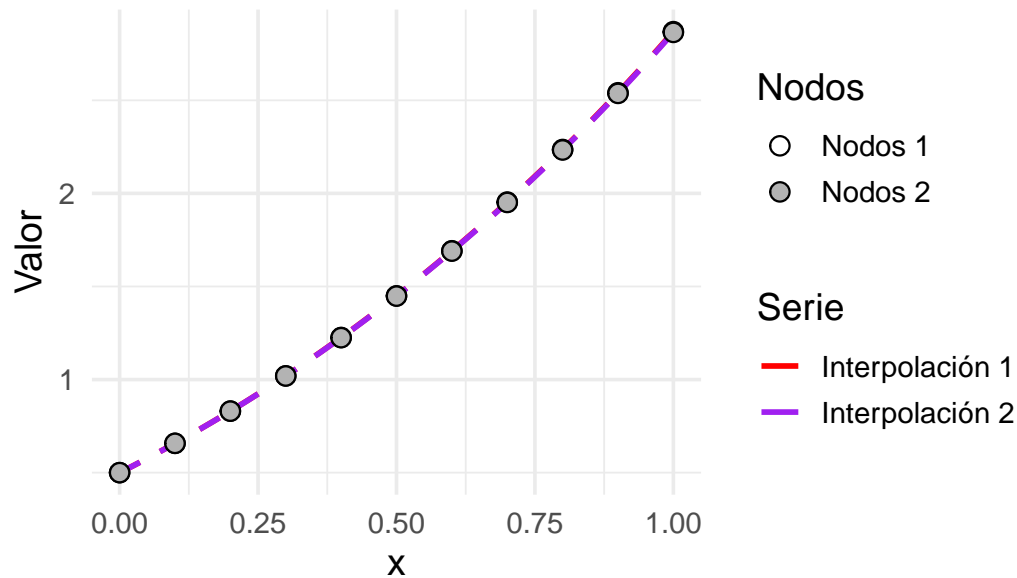
```

Interpolación 1: neville | Interpolación 2: metodo



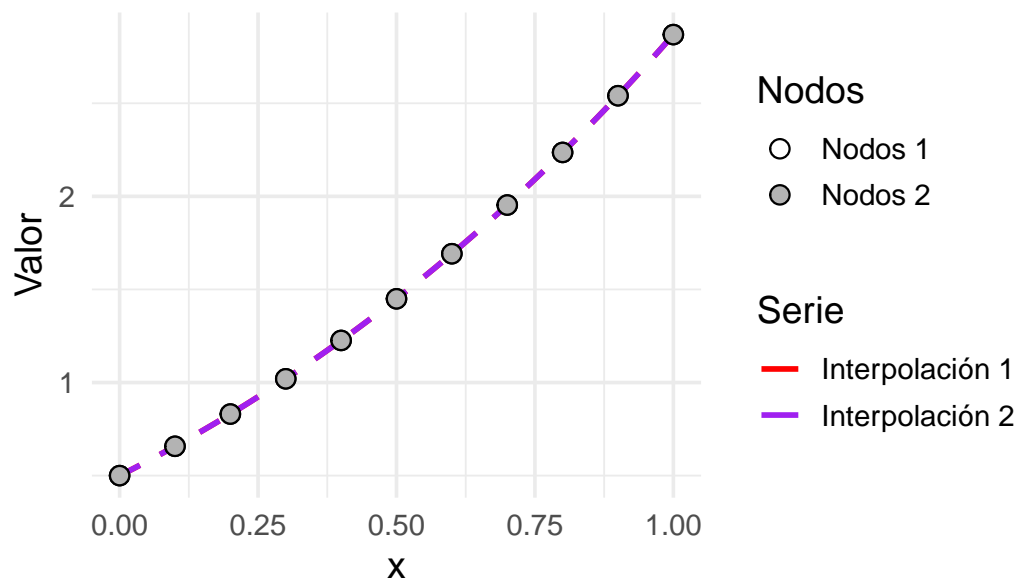
```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p2.rkpm$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metodo



```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p2.rk4o$t, valores2 =
```

Interpolación 1: neville | Interpolación 2: metodo



Problema 3:

```
edo <- function(t, y, parms) {
  dydt <- - ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) )
  list(dydt)
```

```

}

y0 <- 1
tiempos <- seq(0, 0.999, by = 0.1)

sol <- ode(y = y0,
           times = tiempos,
           func = edo,
           parms = NULL)

sol

```

```

      time 1
1    0.0 1
2    0.1 1
3    0.2 1
4    0.3 1
5    0.4 1
6    0.5 1
7    0.6 1
8    0.7 1
9    0.8 1
10   0.9 1

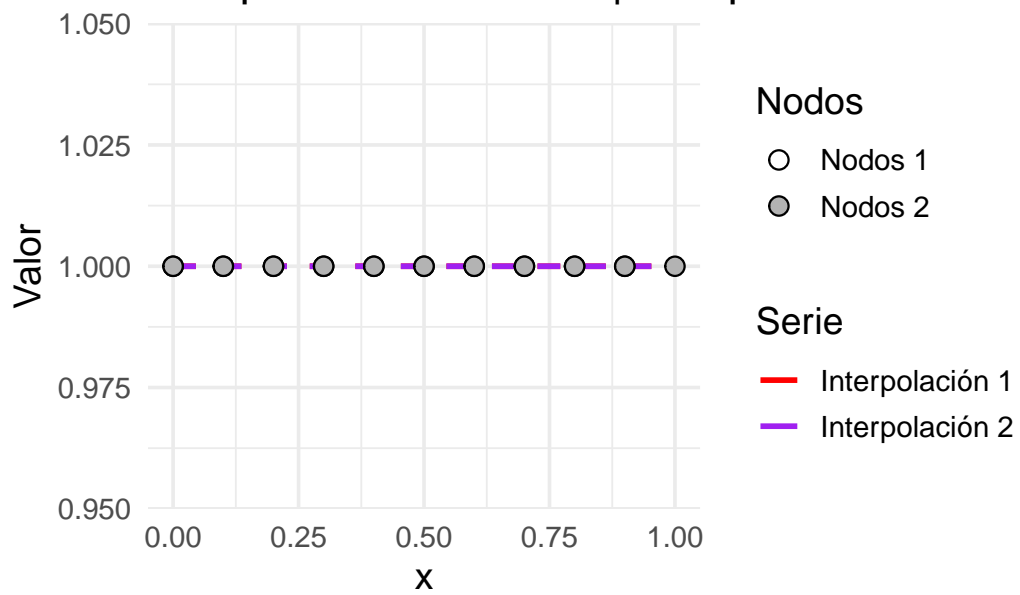
```

```

graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p3.euler$t, valores2 =

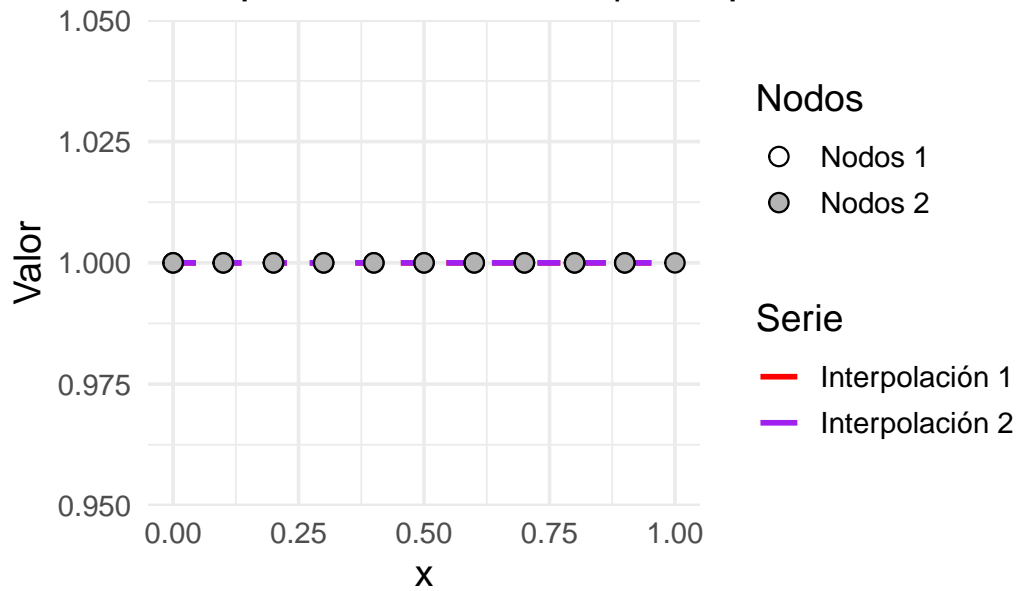
```

Interpolación 1: neville | Interpolación 2: metc

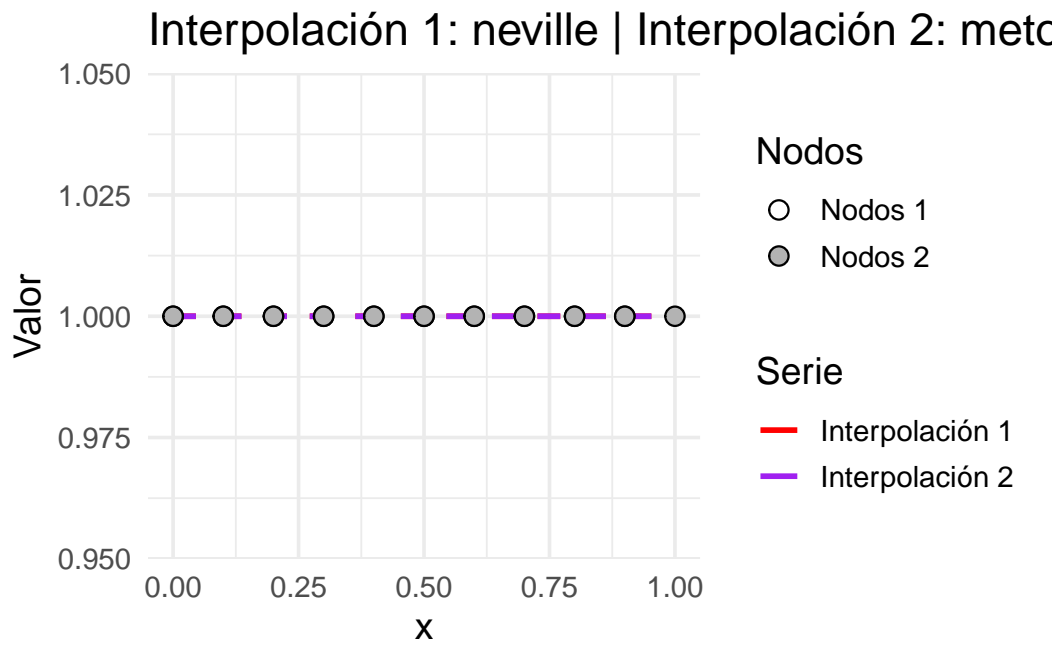


```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p3.euler.pc$t, valores2 =
```

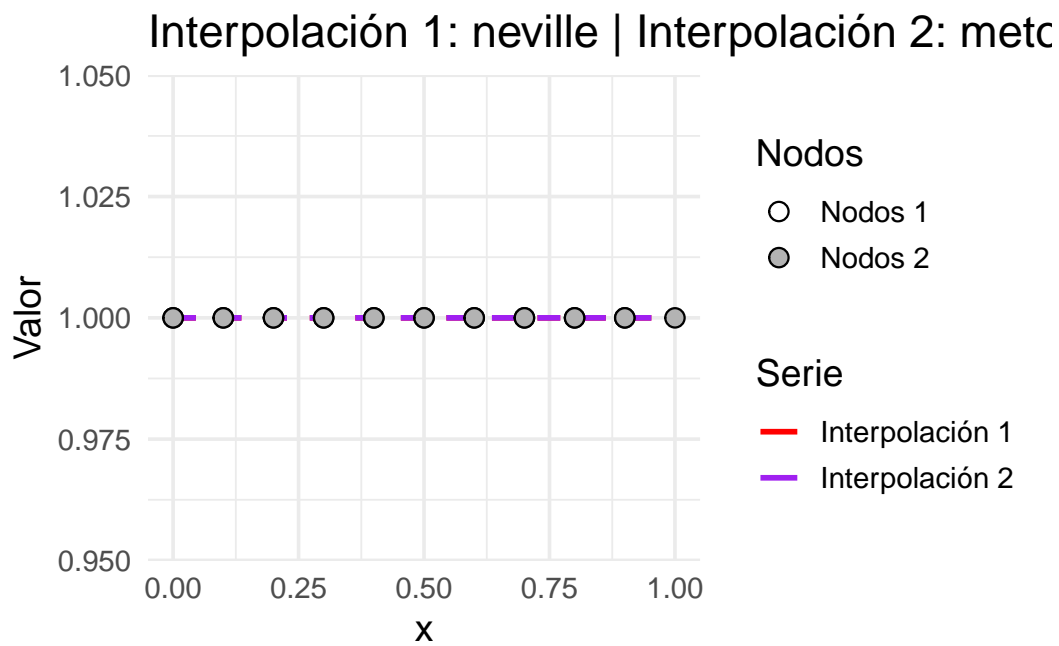
Interpolación 1: neville | Interpolación 2: metc



```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p3.rkpm$t, valores2 =
```



```
graficar.polinomio(sol[,1], 0, 1, neville, valores1 = sol[,2], nodos2 = p3.rk4o$t, valores2 =
```



d) El mejor método parece ser el de Runge-Kuta de cuarto orden

Ejercicio 4

i Instrucción del ejercicio 4

Muestre que el método de **Euler** falla al aproximar la solución

$$u(x) = \left(\frac{2}{3}x\right)^{3/2}$$

para el problema de valor inicial

$$u' = u^{1/3}, \quad u(0) = 0.$$

Explique por qué falla.

Solución

La ecuación diferencial es de la forma:

$$u' = u^{1/3}, \quad u(0) = 0.$$

Podemos obtener la solución exacta de la ecuación diferencial. La solución es:

$$u(x) = \left(\frac{2}{3}x\right)^{3/2}.$$

El método de Euler se basa en la siguiente fórmula:

$$u_{i+1} = u_i + hf(x_i, u_i),$$

donde $f(x, u) = u^{1/3}$, y el valor inicial es $u(0) = 0$.

Partimos del valor inicial $u_0 = 0$. En el primer paso, aplicamos la fórmula de Euler:

$$u_1 = u_0 + hf(x_0, u_0) = 0 + h(0^{1/3}) = 0.$$

El valor de u_1 no avanza, ya que el valor de la derivada en el primer paso es cero, $f(0, 0) = 0$. Esto hace que el método de Euler falle, ya que no obtiene ningún avance, y por lo tanto no sigue la forma de la solución exacta.

El fallo ocurre debido a la singularidad de la ecuación diferencial en $u = 0$. La función $u^{1/3}$ no es suficientemente suave en $u = 0$, lo que hace que el método de Euler no pueda avanzar adecuadamente.

Es un ejemplo clásico de un método numérico que falla cuando se aplica a ecuaciones diferenciales con condiciones iniciales en puntos donde la solución es no diferenciable o tiene un comportamiento singular.

El método de Euler no puede manejar este tipo de singularidades y, por lo tanto, no proporciona una aproximación precisa de la solución.



Ejercicio 5

Instrucción del ejercicio 5

Demuestre que el método de un paso dado por:

$$u_{j+1} = u_j + hf\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}f(x_j, u_j)\right)$$

es consistente y que, si f es dos veces continuamente diferenciable, entonces tiene **orden dos** (este método se conoce como el **método de Euler Modificado**).

Implemente este método en **R** y resuelva las ecuaciones diferenciales del problema 2. Luego grafique.

Solución

Prueba

Sea $u(x)$ la solución exacta de la EDO $u' = f(x, u)$ con $u(x_j) = u_j$.

Definimos:

$$\phi(x_j, u_j; h) = f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}f(x_j, u_j)\right).$$

Consistencia

El método es consistente si:

$$\lim_{h \rightarrow 0} \phi(x_j, u_j; h) = f(x_j, u_j).$$

Como f es continua, y el argumento de f tiende a (x_j, u_j) cuando $h \rightarrow 0$, entonces:

$$\lim_{h \rightarrow 0} f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}f(x_j, u_j)\right) = f(x_j, u_j).$$

Por lo tanto, el método es consistente.

Orden de consistencia 2

Recordemos que el error de discretización local se define por:

$$\Delta(x_j, u_j; h) = \frac{1}{h} [u(x_j + h) - u(x_j)] - \phi(x_j, u_j; h).$$

Expandiendo $u(x_j + h)$ en serie de Taylor:

$$\begin{aligned} u(x_j + h) &= u(x_j) + hu'(x_j) + \frac{h^2}{2}u''(x_j) + \mathcal{O}(h^3) \\ &= u_j + hf(x_j, u_j) + \frac{h^2}{2}(f_x + f_u f) + \mathcal{O}(h^3), \end{aligned}$$

donde:

$$u'' = \frac{d}{dx}f(x, u) = f_x + f_u u' = f_x + f_u f.$$

Por otro lado, expandimos $\phi(x_j, u_j; h)$ usando la fórmula de Taylor multivariable:

$$\begin{aligned} \phi(x_j, u_j; h) &= f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}f(x_j, u_j)\right) \\ &= f(x_j, u_j) + \frac{h}{2}f_x + \frac{h}{2}f_u f + \mathcal{O}(h^2). \end{aligned}$$

Entonces:

$$\begin{aligned} \Delta(x_j, u_j; h) &= \frac{1}{h} [u(x_j + h) - u(x_j)] - \phi(x_j, u_j; h) \\ &= f + \frac{h}{2}(f_x + f_u f) - \left[f + \frac{h}{2}(f_x + f_u f)\right] + \mathcal{O}(h^2) \\ &= \mathcal{O}(h^2). \end{aligned}$$

Por lo tanto, el método tiene **orden de consistencia 2**.

Código

```
# Método de Euler Modificado (RK2)
euler_modificado <- function(f, a, b, y0, N) {
  h <- (b - a) / N
  t <- numeric(N + 1)
  y <- numeric(N + 1)
  t[1] <- a
  y[1] <- y0

  for (i in 1:N) {
    k1 <- f(t[i], y[i])
```

```

    k2 <- f(t[i] + h/2, y[i] + (h/2) * k1)
    y[i + 1] <- y[i] + h * k2
    t[i + 1] <- t[i] + h
  }

  data.frame(t = t, y = y)
}

```

Problema 1: $y' = -2ty^2$, $y(0) = 1$

```

f1 <- function(t, y) {
  -2 * t * y^2
}

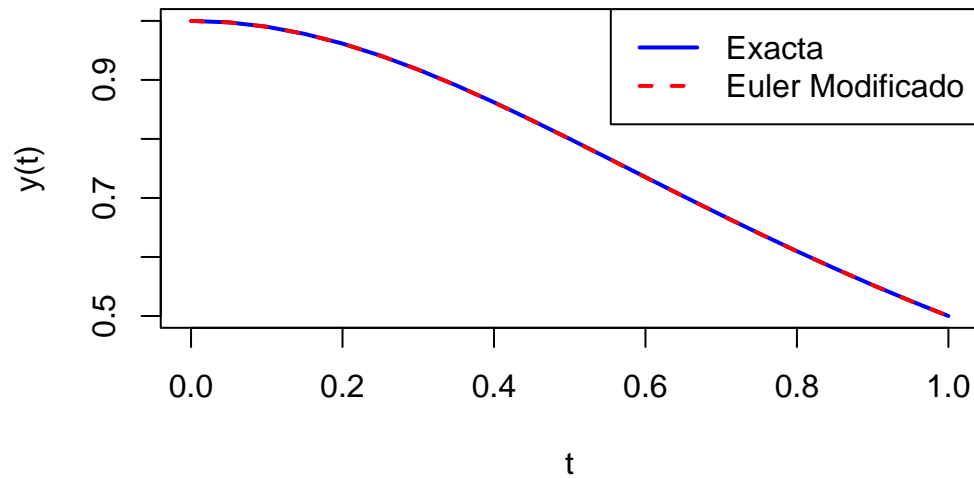
# Parámetros
a1 <- 0; b1 <- 1; y01 <- 1; N <- 20
res1 <- euler_modificado(f1, a1, b1, y01, N)

# Solución exacta: y(t) = 1 / (1 + t^2)
y1_exacta <- function(t) {
  1 / (1 + t^2)
}

# Graficar
plot(res1$t, y1_exacta(res1$t), type = "l", col = "blue", lwd = 2,
     main = "Problema 1: Euler Modificado vs Solución Exacta", xlab = "t", ylab = "y(t)")
lines(res1$t, res1$y, col = "red", lty = 2, lwd = 2)
legend("topright", legend = c("Exacta", "Euler Modificado"),
     col = c("blue", "red"), lty = c(1,2), lwd = 2)

```

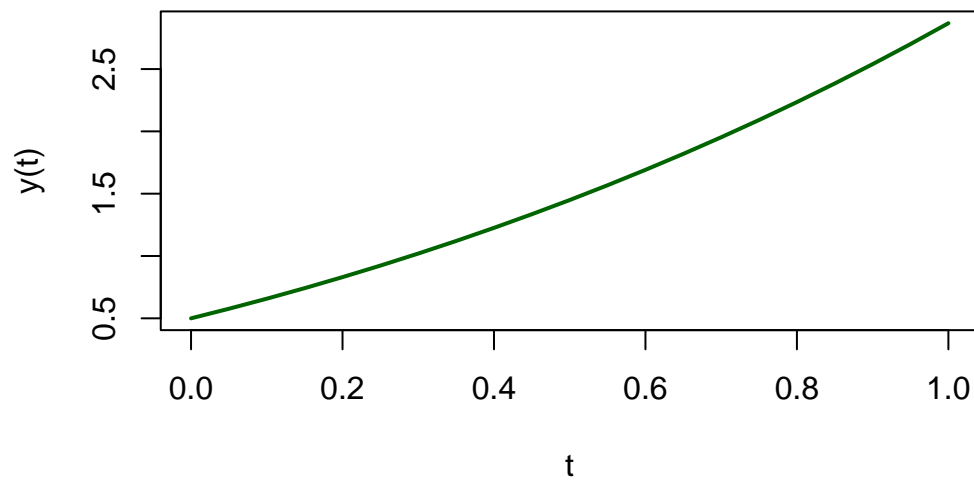
Problema 1: Euler Modificado vs Solución Exacta



Problema 2: $y' - y = \cos(t)$, $y(0) = \frac{1}{2}$

```
f2 <- function(t, y) {  
  cos(t) + y  
}  
  
# Parámetros  
a2 <- 0; b2 <- 1; y02 <- 0.5  
res2 <- euler_modificado(f2, a2, b2, y02, N)  
  
# No hay solución exacta elemental, graficamos solo la aproximada  
plot(res2$t, res2$y, type = "l", col = "darkgreen", lwd = 2,  
      main = "Problema 2: Euler Modificado", xlab = "t", ylab = "y(t)")
```

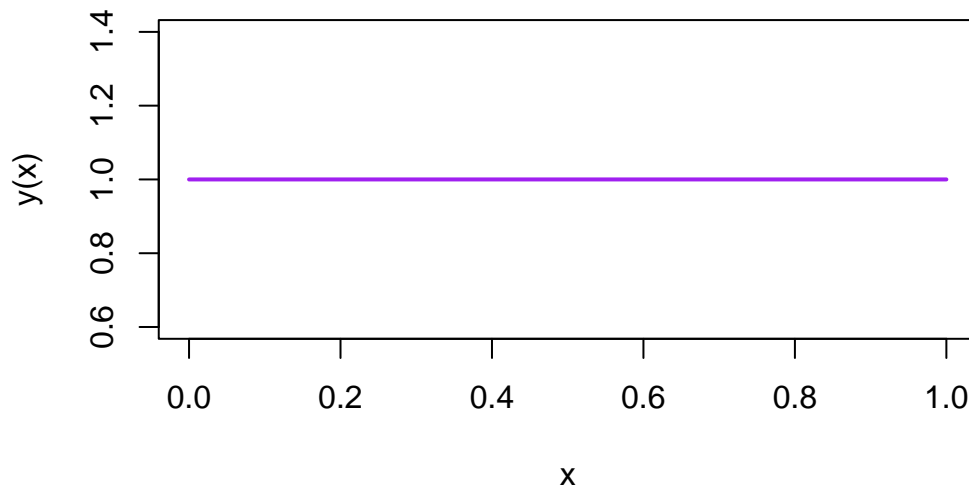
Problema 2: Euler Modificado



Problema 3: Ecuación implícita transformada

```
f3 <- function(x, y) {  
  - (x * sqrt(1 - y^2)) / (y * sqrt(1 - x^2))  
}  
  
a3 <- 0  
b3 <- 1  
y03 <- 1  
N <- 1000  
  
res3 <- euler_modificado(f3, a3, b3, y03, N)  
  
validos <- !is.na(res3$y)  
plot(res3$t[validos], res3$y[validos], type = "l", col = "purple", lwd = 2,  
      main = "Problema 3: Euler Modificado (condición correcta)",  
      xlab = "x", ylab = "y(x)")
```

Problema 3: Euler Modificado (condición correcta)



Ejercicio 6

i Instrucción del ejercicio 6

Demuestre que el método de un paso dado por:

$$\begin{aligned}k_1 &= f(x_j, u_j), \\k_2 &= f\left(x_j + \frac{h}{3}, u_j + \frac{h}{3}k_1\right), \\k_3 &= f\left(x_j + \frac{2h}{3}, u_j + \frac{2h}{3}k_2\right), \\u_{j+1} &= u_j + \frac{h}{4}(k_1 + 3k_3),\end{aligned}$$

es consistente y que, si f es tres veces continuamente diferenciable, entonces tiene **orden tres** (este método se conoce como el **método de Tercer Orden de Heun**).

Implemente este método en **R** y resuelva las ecuaciones diferenciales del problema 2. Luego grafique.

Solución

Sea $u(x)$ la solución exacta y denote $f_j := f(x_j, u_j)$. Definimos el operador

$$D := \frac{\partial}{\partial x} + f_y(x, u) \cdot,$$

con lo cual, evaluado en la solución exacta,

$$u' = f, \quad u'' = Df, \quad u''' = D^2f.$$

La expansión de Taylor de u es

$$u(x_j + h) = u_j + hf_j + \frac{h^2}{2} Df_j + \frac{h^3}{6} D^2f_j + \mathcal{O}(h^4).$$

- Primera etapa:

$$k_1 = f_j.$$

- Segunda etapa (Taylor multivariable alrededor de (x_j, u_j)):

$$k_2 = f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}k_1\right) = f_j + \frac{h}{2} Df_j + \frac{h^2}{8} D^2f_j + \mathcal{O}(h^3).$$

- Tercera etapa (usando la misma técnica y reteniendo términos hasta h^2):

$$k_3 = f(x_j + h, u_j - hk_1 + 2hk_2) = f_j + h Df_j + \frac{h^2}{2} D^2f_j + \mathcal{O}(h^3).$$

Observación: estas fórmulas incluyen correctamente los efectos de la dependencia en u vía f_y (capturados por D). No es suficiente expandir solo con derivadas respecto a x .

El incremento del esquema es

$$\Phi_h(x_j, u_j) = \frac{1}{6} (k_1 + 4k_2 + k_3) = f_j + \frac{h}{2} Df_j + \frac{h^2}{6} D^2f_j + \mathcal{O}(h^3).$$

Por tanto,

$$u_{j+1} = u_j + h \Phi_h(x_j, u_j) = u_j + hf_j + \frac{h^2}{2} Df_j + \frac{h^3}{6} D^2f_j + \mathcal{O}(h^4),$$

que coincide con la expansión de $u(x_j + h)$ hasta términos $\mathcal{O}(h^3)$.

De esta forma se puede concluir

- Consistencia: $\Phi_h(x_j, u_j) = f_j + \mathcal{O}(h)$.
- Error local de truncamiento: $\mathcal{O}(h^4)$.
- Orden global: 3.

Como verificación alternativa, las **condiciones de orden de Runge–Kutta** para $p = 3$ se satisfacen con los coeficientes dados:

$$\sum b_i = 1, \quad \sum b_i c_i = \frac{1}{2}, \quad \sum b_i c_i^2 = \frac{1}{3}, \quad \sum_{i,j} b_i a_{ij} c_j = \frac{1}{6},$$

lo cual confirma que el método (Heun de 3er orden) es de **orden tres**.

Codigo en R

```
tercer.orden.huen <- function(a, b, N, alfa, f){  
  h <- (b - a) / N  
  t <- a  
  w <- alfa  
  T <- numeric(N + 1)  
  W <- numeric(N + 1)  
  T[1] <- t  
  W[1] <- w  
  for (i in 2:(N+1)) {  
    k1 <- f(T[i-1], W[i-1])  
    k2 <- f(T[i-1] + h/3, W[i-1] + k1*(h/3))  
    k3 <- f(T[i-1] + (2/3)*h, W[i-1] + (2*h/3)*k2)  
  
    W[i] <- W[i-1] + (h/4)*(k1 + 3*k3)  
    T[i] <- T[i - 1] + h  
  }  
  
  tabla <- data.frame(t = T, w = W)  
  
  return(list(t = T, w = W, tabla = tabla))  
}
```

Problema 1: $y' = -2ty^2$, $y(0) = 1$

```
p1 <- tercer.orden.huen(0, 1, 10, 1, function(x, y) -2*x*y^2)  
p1$tabla
```

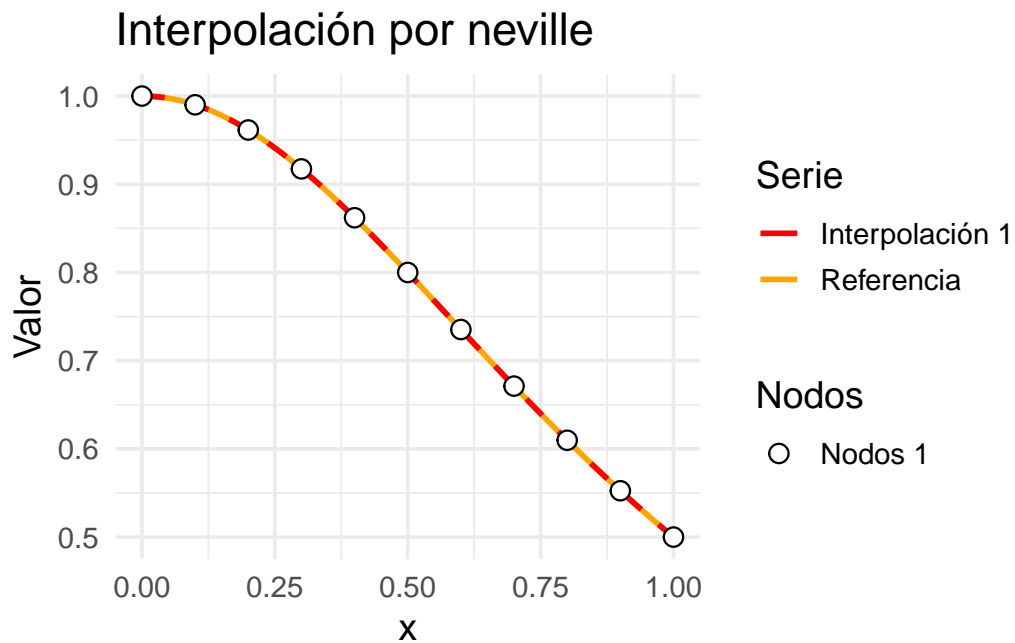
	t	w
1	0.0	1.0000000
2	0.1	0.9900887
3	0.2	0.9615244
4	0.3	0.9174227
5	0.4	0.8620724
6	0.5	0.8000164

```

7  0.6 0.7353196
8  0.7 0.6711698
9  0.8 0.6097831
10 0.9 0.5525078
11 1.0 0.5000145

```

```
graficar.polinomio(p1$t, 0, 1, neville, valores1 = p1$w, f.referencia = function(t) 1 / (1 + t^2))
```



Problema 2: $y' - y = \cos(t)$, $y(0) = \frac{1}{2}$

```

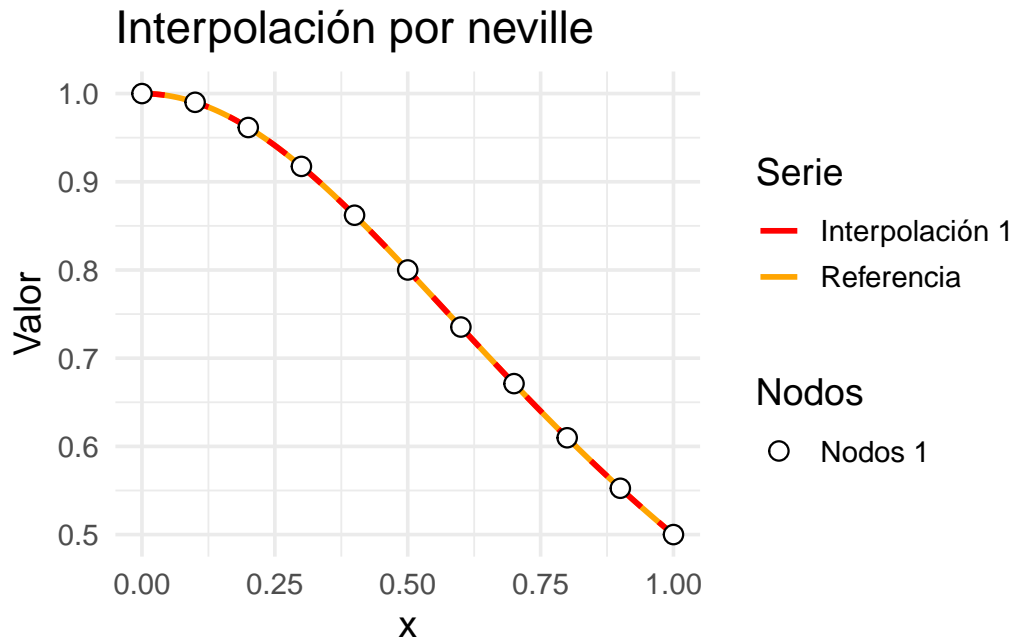
p2 <- tercer.orden.huen(0, 1, 10, 1/2, function(x, y) cos(x) + y)
p1$tabla

```

	t	w
1	0.0	1.0000000
2	0.1	0.9900887
3	0.2	0.9615244
4	0.3	0.9174227
5	0.4	0.8620724
6	0.5	0.8000164
7	0.6	0.7353196
8	0.7	0.6711698
9	0.8	0.6097831

```
10 0.9 0.5525078
11 1.0 0.5000145
```

```
graficar.polinomio(p1$t, 0, 1, neville, valores1 = p1$w, f.referencia = function(t) 1 / (1 + t^2))
```

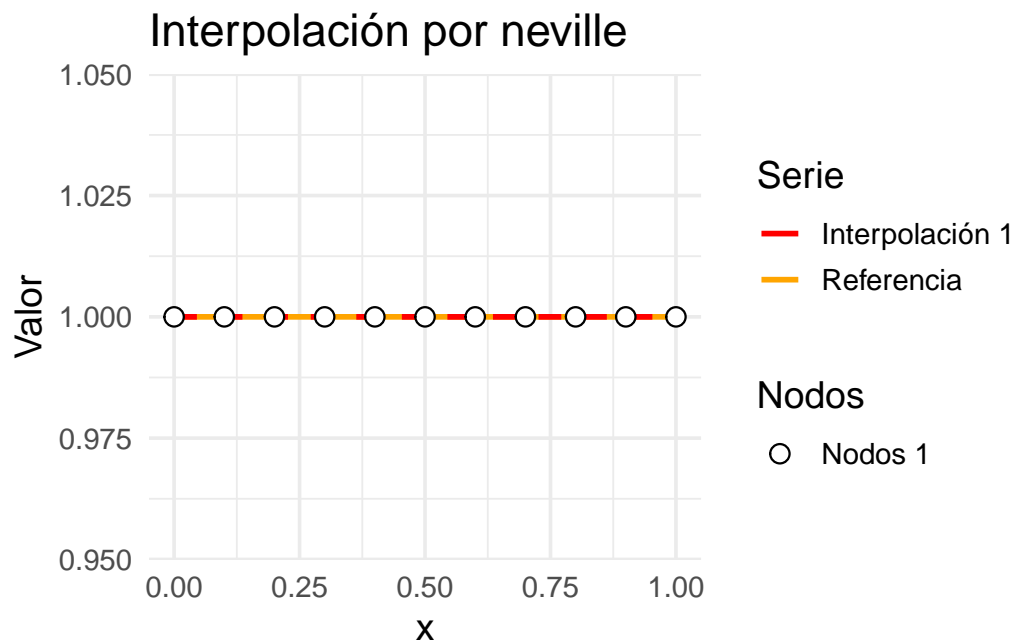


Problema 3: Ecuación implícita transformada

```
p3 <- tercer.orden.huen(0, 1, 10, 1, function(t, y)- ( t * sqrt(1 - y^2) ) / ( y * sqrt(1 - t^2) ))
p3$tabla
```

	t	w
1	0.0	1
2	0.1	1
3	0.2	1
4	0.3	1
5	0.4	1
6	0.5	1
7	0.6	1
8	0.7	1
9	0.8	1
10	0.9	1
11	1.0	1

```
graficar.polinomio(p3$t, 0, 1, neville, valores1 = p3$w, f.referencia = function(x) 1)
```



Ejercicio 7

i Instrucción del ejercicio 7

Demuestre que el método de un paso dado por:

$$\begin{aligned} k_1 &= f(x_j, u_j), \\ k_2 &= f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}k_1\right), \\ k_3 &= f(x_j + h, u_j - hk_1 + 2hk_2), \\ u_{j+1} &= u_j + \frac{h}{6}(k_1 + 4k_2 + k_3), \end{aligned}$$

es consistente y que si f es tres veces continuamente diferenciable entonces tiene **orden tres** (este método se conoce como el **método de Tercer Orden de Runge–Kutta**).

Implemente este método en **R** y resuelva las ecuaciones diferenciales del problema 2, luego grafique.

Solución

- Primera etapa:

$$k_1 = f_j.$$

- Segunda etapa (Taylor multivariable alrededor de (x_j, u_j)):

$$k_2 = f\left(x_j + \frac{h}{2}, u_j + \frac{h}{2}k_1\right) = f_j + \frac{h}{2} Df_j + \frac{h^2}{8} D^2 f_j + \mathcal{O}(h^3).$$

- Tercera etapa (usando la misma técnica y reteniendo términos hasta h^2):

$$k_3 = f(x_j + h, u_j - hk_1 + 2h k_2) = f_j + h Df_j + \frac{h^2}{2} D^2 f_j + \mathcal{O}(h^3).$$

Observación: estas fórmulas incluyen correctamente los efectos de la dependencia en u vía f_y (capturados por D). No es suficiente expandir solo con derivadas respecto a x .

El incremento del esquema es

$$\Phi_h(x_j, u_j) = \frac{1}{6} (k_1 + 4k_2 + k_3) = f_j + \frac{h}{2} Df_j + \frac{h^2}{6} D^2 f_j + \mathcal{O}(h^3).$$

Por tanto,

$$u_{j+1} = u_j + h \Phi_h(x_j, u_j) = u_j + hf_j + \frac{h^2}{2} Df_j + \frac{h^3}{6} D^2 f_j + \mathcal{O}(h^4),$$

que coincide con la expansión de $u(x_j + h)$ hasta términos $\mathcal{O}(h^3)$.

De esta forma se puede concluir

- Consistencia: $\Phi_h(x_j, u_j) = f_j + \mathcal{O}(h)$.
- Error local de truncamiento: $\mathcal{O}(h^4)$.
- Orden global: 3.

Como verificación alternativa, las **condiciones de orden de Runge–Kutta** para $p = 3$ se satisfacen con los coeficientes dados:

$$\sum b_i = 1, \quad \sum b_i c_i = \frac{1}{2}, \quad \sum b_i c_i^2 = \frac{1}{3}, \quad \sum_{i,j} b_i a_{ij} c_j = \frac{1}{6},$$

lo cual confirma que el método (Heun de 3er orden) es de **orden tres**.

■

Código

```
# Método de Euler Modificado (RK2)
runge_kutta_3 <- function(f, a, b, y0, N) {
  h <- (b - a) / N
```

```

t <- numeric(N + 1)
y <- numeric(N + 1)
t[1] <- a
y[1] <- y0

for (i in 1:N) {
  k1 <- f(t[i], y[i])
  k2 <- f(t[i] + h/2, y[i] + h/2 * k1)
  k3 <- f(t[i] + h, y[i] - h * k1 + 2 * h * k2)
  y[i + 1] <- y[i] + h / 6 * (k1 + 4 * k2 + k3)
  t[i + 1] <- t[i] + h
}

data.frame(t = t, y = y)
}

```

Problema 1: $y' = -2ty^2$, $y(0) = 1$

```

# Función del problema 1
f1 <- function(t, y) {
  -2 * t * y^2
}

# Parámetros
a1 <- 0; b1 <- 1; y01 <- 1; N <- 20
res1 <- runge_kutta_3(f1, a1, b1, y01, N)

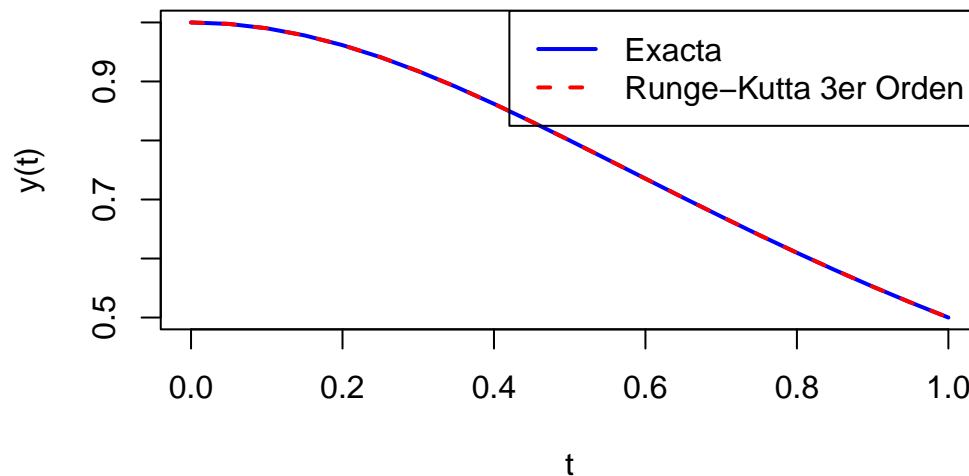
# Solución exacta:  $y(t) = 1 / (1 + t^2)$ 
y1_exacta <- function(t) {
  1 / (1 + t^2)
}

# Graficar
plot(res1$t, y1_exacta(res1$t), type = "l", col = "blue", lwd = 2,
     main = "Problema 1: Runge-Kutta de 3er Orden vs Solución Exacta", xlab = "t", ylab = "y(t)",
     lines(res1$t, res1$y, col = "red", lty = 2, lwd = 2)
legend("topright", legend = c("Exacta", "Runge-Kutta 3er Orden"),

```

```
col = c("blue", "red"), lty = c(1,2), lwd = 2)
```

Problema 1: Runge-Kutta de 3er Orden vs Solución Exac



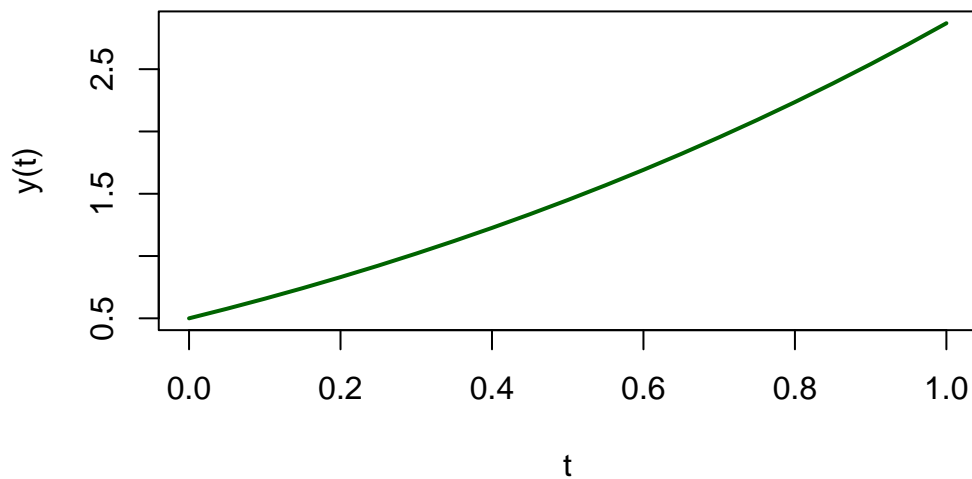
Problema 2: $y' - y = \cos(t)$, $y(0) = \frac{1}{2}$

```
# Función del problema 2
f2 <- function(t, y) {
  cos(t) + y
}

# Parámetros
a2 <- 0; b2 <- 1; y02 <- 0.5
res2 <- runge_kutta_3(f2, a2, b2, y02, N)

# No hay solución exacta elemental, graficamos solo la aproximada
plot(res2$t, res2$y, type = "l", col = "darkgreen", lwd = 2,
     main = "Problema 2: Runge-Kutta de 3er Orden", xlab = "t", ylab = "y(t)")
```

Problema 2: Runge-Kutta de 3er Orden



Problema 3: Ecuación implícita transformada

```
# Función del problema
f3 <- function(x, y) {
  if (is.na(x) || is.na(y) || is.nan(x) || is.nan(y)) return(NA)
  if ((1 - x^2) < 0 || (1 - y^2) < 0 || abs(y) < 1e-8) return(NA)

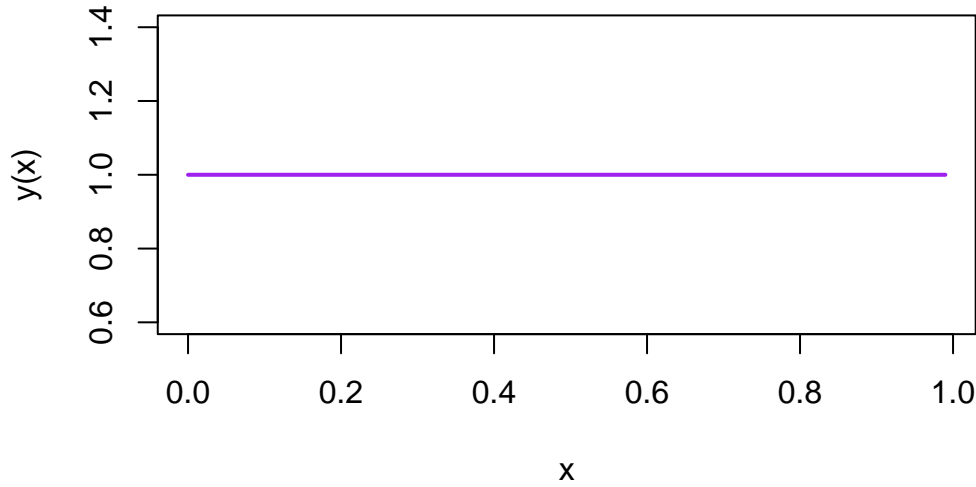
  - (x * sqrt(1 - y^2)) / (y * sqrt(1 - x^2))
}

# Parámetros
a3 <- 0
b3 <- 1
y03 <- 1
N <- 100

# Resolver con Runge-Kutta de 3er orden
res3 <- runge_kutta_3(f3, a3, b3, y03, N)

# Graficar
validos <- !is.na(res3$y)
plot(res3$t[validos], res3$y[validos], type = "l", col = "purple", lwd = 2,
      main = "Problema 3: Runge-Kutta de 3er Orden",
      xlab = "x", ylab = "y(x)")
```

Problema 3: Runge–Kutta de 3er Orden



Ejercicio 8

i Instrucción del ejercicio 8

La idea de este ejercicio es introducir la **matriz exponencial** e^A para resolver sistemas de ecuaciones diferenciales.

Definición:

Dada una sucesión $\{C_k\}$ de matrices $m \times n$ cuyos elementos son reales o complejos, se denota por c_{ij}^k la entrada ij de C_k .

Entonces, si todas las mn series

$$\sum_{k=1}^{\infty} c_{ij}^k \quad \text{con } i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n$$

son convergentes, se dice que la serie de matrices $\sum_{k=1}^{\infty} C_k$ es convergente y su suma es la matriz cuya entrada ij es:

$$\sum_{k=1}^{\infty} c_{ij}^k.$$

(a) Pruebe que si $\{C_k\}$ es una sucesión de matrices $m \times n$ tales que

$\sum_{k=1}^{\infty} \|C_k\|$ converge, entonces la serie de matrices $\sum_{k=1}^{\infty} C_k$ también es convergente.

(b) Pruebe que la serie $\sum_{k=0}^{\infty} \frac{A^k}{k!}$ es convergente.

i Definición

Dada una matriz A , $n \times n$ con elementos reales o complejos, se define la **matriz exponencial** como:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}.$$

(c) Verifique que si

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

entonces

$$e^A = \begin{pmatrix} e^t & 0 & 0 \\ 0 & e^{2t} & 0 \\ 0 & 0 & e^{3t} \end{pmatrix}.$$

(e) Verifique que si

$$A = \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}$$

entonces

$$e^{tA} = \begin{pmatrix} e^{at} & te^{at} \\ 0 & e^{at} \end{pmatrix}.$$

(f) Pruebe que para todo $t \in \mathbb{R}$ la función matricial $E(t) = e^{tA}$ satisface la ecuación diferencial matricial:

$$E'(t) = E(t)A = AE(t).$$

(g) Dada una matriz A , $n \times n$ con elementos reales o complejos, pruebe que para todo $t \in \mathbb{R}$ se cumple:

$$e^{tA}e^{-tA} = I$$

o sea que e^{tA} es no singular y su inversa es e^{-tA} .

- (h) Pruebe que si A y B son dos matrices $n \times n$ con elementos reales o complejos tales que $AB = BA$, entonces:

$$e^{A+B} = e^A e^B.$$

- (i) Pruebe que, dada una matriz A , $n \times n$ con elementos reales o complejos y un vector B de n entradas, entonces el problema de valor inicial:

$$Y'(t) = AY(t)$$

$$Y(0) = B$$

tiene solución única con $t \in \mathbb{R}$ y que esta solución está dada por:

$$Y(t) = e^{tA}B.$$

- (j) Pruebe que si una matriz cuadrada A es diagonalizable, es decir, que existe una matriz C no singular tal que la matriz $D = C^{-1}AC$ es diagonal, entonces:

$$e^{tA} = Ce^{tD}C^{-1}.$$

- (k) Escriba una función en **R** que reciba una matriz A diagonalizable y retorne e^{tA} . Para esto puede usar la función **eigen(...)** de R.

- (l) Verifique que si

$$A = \begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix}$$

entonces

$$e^{tA} = -\frac{1}{3} \begin{pmatrix} -e^t - 2e^{4t} & -e^t + e^{4t} \\ -2e^t + 2e^{4t} & -2e^t - e^{4t} \end{pmatrix}.$$

- (m) Resuelva el siguiente sistema de ecuaciones diferenciales:

$$\begin{cases} x_1'(t) = 3x_1(t) - x_2(t) \\ x_2'(t) = -2x_1(t) + 2x_2(t) \end{cases}$$

sujeto a $x_1(0) = 90$ y $x_2(0) = 150$.

(n) Escriba una función en **R** que reciba una matriz A con los coeficientes de un sistema de ecuaciones diferenciales, las condiciones iniciales en una lista de pares, y retorne una lista con las soluciones del sistema de ecuaciones diferenciales.

(o) Usando el programa, resuelva el siguiente sistema de ecuaciones diferenciales:

$$\begin{cases} x_1'(t) = 3x_1(t) - 2x_2(t) \\ x_2'(t) = -2x_1(t) + 3x_2(t) \\ x_3'(t) = 5x_3(t) \end{cases}$$

sujeto a $x_1(0) = 2$, $x_2(0) = 1$ y $x_3(0) = 3$.

Solución

a)

Queremos ver que $\sum_{k=1}^{\infty} c_{ij}^k$ converge

Tome la suma parcial entre dos índices p y q ($p < q$). Note, por desigualdad triangular, que:

$$\left| \sum_{k=p}^q c_{ij}^k \right| \leq \sum_{k=p}^q |c_{ij}^k|$$

Ahora nos apoyamos en el hecho de que cada entrada $|c_{ij}|$ de la matriz está acotada por la norma de la matriz total: $c_{ij} \leq \|C\| \ \forall i, j$

$$\Rightarrow \sum_{k=p}^q |c_{ij}^k| \leq \sum_{k=p}^q \|C_k\|$$

Dado que $\sum_{k=p}^q \|C_k\|$ converge, entonces es de Cauchy, por lo cual:

$$\forall \epsilon > 0 \exists N_0 \in \mathbb{N} : N_0 < p < q \Rightarrow \sum_{k=p}^q \|C_k\| < \epsilon$$

Juntando todo esto, se concluye que $\sum_{k=1}^n c_{ij}^k$ es una sucesión de Cauchy, y dado que “vive” en un espacio completo (\mathbb{R} o \mathbb{C}), entonces converge.

Así pues, dado que hay convergencia en cada entrada de la matriz, podemos ver que $\sum_{k=1}^{\infty} C_k$ converge.

b)

Basta con demostrar que $\sum_{k=0}^{\infty} \left\| \frac{A^k}{k!} \right\|$ converge por lo visto en el inciso anterior.

Note que, en el caso de una norma matricial, se tiene que $\|\alpha \cdot A\| = |\alpha| \cdot \|A\|$ y $\|AB\| \leq \|A\| \cdot \|B\|$

$$\begin{aligned} \Rightarrow \left\| \frac{A^k}{k!} \right\| &\leq \frac{\|A^k\|}{k!} \\ \Rightarrow \sum_{k=0}^{\infty} \left\| \frac{A^k}{k!} \right\| &\leq \sum_{k=0}^{\infty} \frac{\|A^k\|}{k!} = e^{\|A\|} < \infty \end{aligned}$$

Por lo tanto, $\sum_{k=0}^{\infty} \left\| \frac{A^k}{k!} \right\|$ converge, lo que significa que $\sum_{k=0}^{\infty} \frac{A^k}{k!}$ converge

c)

$$e^{tA} = \sum_{k=0}^{\infty} \frac{(tA)^k}{k!} = \sum_{k=0}^{\infty} \frac{1}{k!} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2t & 0 \\ 0 & 0 & 3t \end{pmatrix}^k = \sum_{k=0}^{\infty} \begin{pmatrix} \frac{1^k}{k!} & 0 & 0 \\ 0 & \frac{(2t)^k}{k!} & 0 \\ 0 & 0 & \frac{(3t)^k}{k!} \end{pmatrix}.$$

Esta última igualdad es válida al estar trabajando con una matriz diagonal (de lo contrario no sería tan fácil como “solo meter el exponente a cada entrada”)

$$= \begin{pmatrix} \sum_{k=0}^{\infty} \frac{1^k}{k!} & 0 & 0 \\ 0 & \sum_{k=0}^{\infty} \frac{(2t)^k}{k!} & 0 \\ 0 & 0 & \sum_{k=0}^{\infty} \frac{(3t)^k}{k!} \end{pmatrix} = \begin{pmatrix} e^t & 0 & 0 \\ 0 & e^{2t} & 0 \\ 0 & 0 & e^{3t} \end{pmatrix}.$$

e)

Proposición. Para todo $n \in \mathbb{N}$,

$$\begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^n = \begin{pmatrix} a^n & n a^{n-1} \\ 0 & a^n \end{pmatrix}.$$

Demostración

Caso base ($n = 1$).

$$\begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^1 = \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix} = \begin{pmatrix} a^1 & 1 \cdot a^0 \\ 0 & a^1 \end{pmatrix}.$$

Hipótesis inductiva. Suponga que para algún $n \geq 1$,

$$\begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^n = \begin{pmatrix} a^n & n a^{n-1} \\ 0 & a^n \end{pmatrix}.$$

Paso inductivo.

$$\begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^{n+1} = \left(\begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^n \right) \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix} = \begin{pmatrix} a^n & n a^{n-1} \\ 0 & a^n \end{pmatrix} \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}.$$

La multiplicación da:

$$\begin{pmatrix} a^{n+1} & a^n + n a^n \\ 0 & a^{n+1} \end{pmatrix} = \begin{pmatrix} a^{n+1} & (n+1) a^n \\ 0 & a^{n+1} \end{pmatrix}.$$

Conclusión. Por el principio de inducción, la fórmula es válida para todo $n \in \mathbb{N}$

Ahora, procedemos con la demostración solicitada:

Partimos de la definición de la exponencial de matrices:

$$e^{tA} = \sum_{k=0}^{\infty} \frac{(tA)^k}{k!}.$$

Sea

$$A = \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}.$$

Entonces

$$(tA)^k = t^k \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix}^k = t^k \begin{pmatrix} a^k & k a^{k-1} \\ 0 & a^k \end{pmatrix}$$

(donde usamos el resultado probado por inducción).

Sustituimos en la serie:

$$e^{tA} = \sum_{k=0}^{\infty} \frac{t^k}{k!} \begin{pmatrix} a^k & k a^{k-1} \\ 0 & a^k \end{pmatrix}.$$

Esto equivale a sumar componente a componente:

$$e^{tA} = \begin{pmatrix} \sum_{k=0}^{\infty} \frac{(ta)^k}{k!} & \sum_{k=0}^{\infty} \frac{k t^k a^{k-1}}{k!} \\ 0 & \sum_{k=0}^{\infty} \frac{(ta)^k}{k!} \end{pmatrix}.$$

Reconocemos series conocidas: - La serie de la diagonal es $(e^{\widehat{ta}})$. - Para la serie del elemento fuera de la diagonal, usamos:

$$\sum_{k=1}^{\infty} \frac{k t^k a^{k-1}}{k!} = t \sum_{k=1}^{\infty} \frac{(ta)^{k-1}}{(k-1)!} = t e^{ta}.$$

Por tanto,

$$e^{tA} = \begin{pmatrix} e^{ta} & t e^{ta} \\ 0 & e^{ta} \end{pmatrix}.$$

f) Definimos la función exponencial de matriz:

$$E(t) = e^{tA} = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!}.$$

Esto es,

$$E(t) = I + tA + \frac{t^2 A^2}{2!} + \frac{t^3 A^3}{3!} + \dots.$$

Note que la serie asociada a la función exponencial es uniformemente convergente, lo que permite derivar término por término.

Entonces:

$$E'(t) = \sum_{k=0}^{\infty} \frac{d}{dt} \left(\frac{t^k A^k}{k!} \right) = \sum_{k=1}^{\infty} \frac{k t^{k-1} A^k}{k!}.$$

Reescribimos el coeficiente:

$$\frac{k}{k!} = \frac{1}{(k-1)!},$$

por lo que:

$$E'(t) = \sum_{k=1}^{\infty} \frac{t^{k-1} A^k}{(k-1)!}.$$

Cambiamos índice usando $(m = k-1)$:

$$E'(t) = \sum_{m=0}^{\infty} \frac{t^m A^{m+1}}{m!}.$$

Factorizamos (A):

$$E'(t) = A \sum_{m=0}^{\infty} \frac{t^m A^m}{m!}.$$

La serie restante es nuevamente $(E(t))$, por definición. Por tanto:

$$E'(t) = A E(t).$$

Del mismo modo, como (A) conmuta con sus potencias,

$$E'(t) = E(t) A.$$

Conclusión.

$$E'(t) = A E(t) = E(t) A.$$

g)

Tomemos

$$F(t) = e^{tA} e^{-tA}.$$

Nótese que

$$F(t) = I \quad \text{para todo } t \in \mathbb{R}.$$

Observemos que

$$F'(t) = A e^{tA} e^{-tA} + e^{tA} (-A) e^{-tA} = A F(t) - F(t) A.$$

En particular, para $(t=0)$:

$$F(0) = e^{0 \cdot A} e^{0 \cdot (-A)} = I \cdot I = I.$$

Por lo tanto:

$$F(0) = I.$$

Ahora definamos

$$G(t) = F(t) - I.$$

Entonces

$$G(0) = 0.$$

Además, como (I) conmuta con toda matriz, se tiene:

$$F'(t) = A F(t) - F(t) A.$$

Sustituyendo ($F(t)=I+G(t)$):

$$F'(t) = A(I + G(t)) - (I + G(t))A.$$

Esto se simplifica a:

$$A + AG(t) - A - G(t)A = AG(t) - G(t)A.$$

Por lo tanto:

$$G'(t) = AG(t) - G(t)A, \quad G(0) = 0.$$

Es decir, (G) satisface una ecuación diferencial matricial homogénea con condición inicial nula.

Notemos que ($G(t) = 0$) es solución de dicha ecuación diferencial.

Por el **teorema básico de existencia y unicidad de soluciones de EDO**, se concluye que la solución es única.

Entonces:

$$G(t) = 0 \quad \text{para todo } t.$$

En consecuencia:

$$F(t) = I \quad \text{para todo } t.$$

h)

Demostración

Asumimos:

$$A, B \in \mathbb{C}^{n \times n} \quad \text{y} \quad AB = BA.$$

Definimos la exponencial por serie:

$$e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}.$$

Como A y B conmutan, aplica el binomio matricial:

$$(A + B)^k = \sum_{r=0}^k \binom{k}{r} A^r B^{k-r} \quad (k \geq 0).$$

Entonces:

$$\begin{aligned} e^{A+B} &= \sum_{k=0}^{\infty} \frac{(A+B)^k}{k!} = \sum_{k=0}^{\infty} \frac{1}{k!} \sum_{r=0}^k \binom{k}{r} A^r B^{k-r} \\ &= \sum_{k=0}^{\infty} \sum_{r=0}^k \frac{A^r}{r!} \frac{B^{k-r}}{(k-r)!}. \end{aligned}$$

Cambio de índices $p=r$ y $q=k-r$ ($p, q \geq 0$, $k=p+q$):

$$e^{A+B} = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} \frac{A^p}{p!} \frac{B^q}{q!}.$$

Por convergencia absoluta, reordenamos y factorizamos:

$$e^{A+B} = \left(\sum_{p=0}^{\infty} \frac{A^p}{p!} \right) \left(\sum_{q=0}^{\infty} \frac{B^q}{q!} \right) = e^A e^B.$$

Así pues, acabamos de demostrar que:

$$e^{A+B} = e^A e^B.$$

i)

Sea

$$Y(t) = e^{tA} B.$$

Calculamos su derivada (aquí nos apoyamos en lo encontrado en el inciso f):

$$Y'(t) = \frac{d}{dt}(e^{tA})B = A e^{tA} B = A Y(t).$$

Además:

$$Y(0) = e^{0 \cdot A} B = I B = B.$$

Por lo tanto, $Y(t) = e^{tA} B$ cumple el problema planteado.

Para probar unicidad, supongamos por contradicción que existe otra solución $Z(t)$ tal que:

$$Z'(t) = A Z(t), \quad Z(0) = B.$$

Definimos:

$$W(t) = Z(t) - Y(t).$$

Entonces:

$$W'(t) = Z'(t) - Y'(t) = A Z(t) - A Y(t) = A(Z(t) - Y(t)) = A W(t).$$

La condición inicial:

$$W(0) = Z(0) - Y(0) = B - B = 0.$$

Así, $W(t)$ satisface:

$$W'(t) = A W(t), \quad W(0) = 0.$$

La única solución de este sistema homogéneo con condición inicial cero es:

$$W(t) = 0 \quad \text{para todo } t \in \mathbb{R}.$$

Por lo tanto:

$$Z(t) = Y(t) \quad \text{para todo } t.$$

Conclusión. La solución es única y está dada por:

$$Y(t) = e^{tA} B.$$

j)

Demostración

Usamos la definición por serie:

$$e^{tA} = \sum_{k=0}^{\infty} \frac{t^k}{k!} A^k.$$

Primero probamos por inducción que

$$A^k = C D^k C^{-1} \quad \text{para todo } k \geq 1.$$

Caso base $k=1$:

$$A^1 = A = C D C^{-1}.$$

Paso inductivo. Suponga cierto para k , entonces

$$A^{k+1} = A^k A = (C D^k C^{-1})(C D C^{-1}) = C D^k (C^{-1} C) D C^{-1} = C D^{k+1} C^{-1}.$$

Con ello queda probado por inducción.

Sustituyendo en la serie:

$$\begin{aligned} e^{tA} &= \sum_{k=0}^{\infty} \frac{t^k}{k!} A^k = \sum_{k=0}^{\infty} \frac{t^k}{k!} C D^k C^{-1} \\ &= C \left(\sum_{k=0}^{\infty} \frac{t^k}{k!} D^k \right) C^{-1} = C e^{tD} C^{-1}. \end{aligned}$$

Como D es diagonal, e^{tD} es la diagonal con entradas e^{td_i} , pero la identidad anterior vale en general por la linealidad de la serie.

Conclusión.

$$e^{tA} = C e^{tD} C^{-1}.$$

k)

```
exp_matriz <- function(A, t) {
  eig <- eigen(A)
  C <- eig$vectors
  D <- diag(exp(t * eig$values))
  C %*% D %*% solve(C)
}

A <- matrix(c(3, -1, -2, 2), nrow = 2, byrow = TRUE)
t <- 1
exp_matriz(A, t)
```

```
      [,1]      [,2]
[1,] 37.30486 -17.29329
[2,] -34.58658 20.01157
```

l) **Paso 1. Valores y vectores propios.**

$$\chi_A(\lambda) = \det \begin{pmatrix} 3-\lambda & -1 \\ -2 & 2-\lambda \end{pmatrix} = (\lambda-1)(\lambda-4).$$

Por tanto, los autovalores son (1) y (4).

Para $(\lambda = 4)$:

$$A - 4I = \begin{pmatrix} -1 & -1 \\ -2 & -2 \end{pmatrix} \Rightarrow v_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Para $(\lambda = 1)$:

$$A - I = \begin{pmatrix} 2 & -1 \\ -2 & 1 \end{pmatrix} \Rightarrow v_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Paso 2. Diagonalización.

$$C = \begin{pmatrix} 1 & 1 \\ -1 & 2 \end{pmatrix}, \quad D = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}, \quad C^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix}.$$

Paso 3. Exponencial por semejanza.

$$e^{tA} = C e^{tD} C^{-1}, \quad e^{tD} = \begin{pmatrix} e^{4t} & 0 \\ 0 & e^t \end{pmatrix}.$$

Paso 4. Producto final.

$$\begin{aligned} e^{tA} &= \begin{pmatrix} 1 & 1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} e^{4t} & 0 \\ 0 & e^t \end{pmatrix} \frac{1}{3} \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix} \\ &= \frac{1}{3} \begin{pmatrix} 2e^{4t} + e^t & -e^{4t} + e^t \\ -2e^{4t} + 2e^t & e^{4t} + 2e^t \end{pmatrix}. \end{aligned}$$

Esto es equivalente a

$$e^{tA} = -\frac{1}{3} \begin{pmatrix} -e^t - 2e^{4t} & -e^t + e^{4t} \\ -2e^t + 2e^{4t} & -2e^t - e^{4t} \end{pmatrix},$$

que coincide con lo pedido.

II)

Sistema

$$\begin{cases} x_1'(t) = 3x_1(t) - x_2(t), \\ x_2'(t) = -2x_1(t) + 2x_2(t), \end{cases} \quad x_1(0) = 90, \quad x_2(0) = 150.$$

Forma matricial

$$X'(t) = A X(t), \quad A = \begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix}, \quad X(0) = \begin{pmatrix} 90 \\ 150 \end{pmatrix}.$$

Exponencial de A (del cálculo previo)

$$e^{tA} = \frac{1}{3} \begin{pmatrix} 2e^{4t} + e^t & -e^{4t} + e^t \\ -2e^{4t} + 2e^t & e^{4t} + 2e^t \end{pmatrix}.$$

Solución

$$X(t) = e^{tA}X(0) = \frac{1}{3} \begin{pmatrix} 2e^{4t} + e^t & -e^{4t} + e^t \\ -2e^{4t} + 2e^t & e^{4t} + 2e^t \end{pmatrix} \begin{pmatrix} 90 \\ 150 \end{pmatrix}.$$

Componentes

$$\begin{aligned} x_1(t) &= 10e^{4t} + 80e^t, \\ x_2(t) &= -10e^{4t} + 160e^t. \end{aligned}$$

Chequeo inicial

$$x_1(0) = 10 + 80 = 90, \quad x_2(0) = -10 + 160 = 150.$$

m)

```
resolver.sistema.diferencial <- function(A, condiciones.iniciales, t.puntos) {
  nombres <- sapply(condiciones.iniciales, function(x) as.character(x[[1]]))
  B <- matrix(as.numeric(sapply(condiciones.iniciales, function(x) x[[2]])), ncol = 1)
  sol.mat <- sapply(t.puntos, function(t) exp_matriz(A, t) %*% B)
  if (is.vector(sol.mat)) sol.mat <- matrix(sol.mat, nrow = nrow(A))
  rownames(sol.mat) <- nombres
  soluciones <- lapply(seq_len(nrow(sol.mat)), function(i) as.numeric(sol.mat[i, ]))
  names(soluciones) <- nombres
  list(tiempos = t.puntos, soluciones = soluciones)
}

A <- matrix(c(3, -1, -2, 2), nrow = 2, byrow = TRUE)
condiciones <- list(list("x1", 90), list("x2", 150))
t.grid <- seq(0, 1, by = 0.1)
res <- resolver.sistema.diferencial(A, condiciones, t.grid)
data.frame(t = res$tiempos, do.call(cbind, res$soluciones))
```

	t	x1	x2
1	0.0	90.0000	150.00000
2	0.1	103.3319	161.90910

3	0.2	119.9676	173.16903
4	0.3	141.1899	182.77624
5	0.4	168.8763	189.16163
6	0.5	205.7883	189.90484
7	0.6	256.0013	181.30724
8	0.7	325.5467	157.75397
9	0.8	423.3686	110.76125
10	0.9	562.7506	27.55415
11	1.0	763.4440	-111.05641

n)

```
A <- matrix(c(3,-2,0,-2,3,0,0,0,5), 3, 3, byrow = TRUE)
cond <- list(list("x1",2), list("x2",1), list("x3",3))
ts <- seq(0, 1, by = 0.1)
res <- resolver.sistema.diferencial(A, cond, ts)
data.frame(t = res$tiempos, do.call(cbind, res$soluciones))
```

	t	x1	x2	x3
1	0.0	2.000000	1.000000	3.000000
2	0.1	2.482117	0.8333957	4.946164
3	0.2	3.191245	0.4729632	8.154845
4	0.3	4.265633	-0.2160563	13.445067
5	0.4	5.932265	-1.4567910	22.167168
6	0.5	8.564329	-3.6181651	36.547482
7	0.6	12.775947	-7.3095903	60.256611
8	0.7	19.578355	-13.5370969	99.346356
9	0.8	30.637386	-23.9607636	163.794450
10	0.9	48.697970	-41.3191610	270.051394
11	1.0	78.284002	-70.1291568	445.239477