TEMA 6: RESOLUCIÓN NUMÉRICA DE PROBLEMAS DE VALOR INICIAL PARA ECUACIONES DIFERENCIALES ORDINARIAS

INTRODUCCIÓN.

El objetivo de este tema es estudiar métodos numéricos estables, precisos y con bajo coste computacional, que nos permitan resolver problemas de ecuaciones diferenciales ordinarias (CODEs, en inglés) no lineales del tipo:

$$\begin{cases} u'(t) = f(u(t), t), & t > 0 \\ u(0) = u_0 \end{cases}$$

donde $u: [0, T] \to \mathbb{R}$ es la incógnita y $f: \mathbb{R}^2 \to \mathbb{R}$.

También consideraremos el caso de sistemas, es decir,

$u(t) = (u_1(t), u_2(t), \dots, u_N(t))$ y el sistema de EDOs es:

$$\begin{cases} u_1'(t) = f_1(u(t), t) \\ u_2'(t) = f_2(u(t), t) \\ \cdots \\ u_N'(t) = f_N(u(t), t) \\ u_1(0) = u_1^0 \\ u_2(0) = u_2^0 \\ \cdots \\ u_N(0) = u_N^0 \end{cases}$$

Usaremos la ecuación escalar lineal

$$\begin{cases} u'(t) = a\,u(t) \\ u(0) = u_0 \end{cases}$$

cuya solución es $u(t) = u_0 e^{at}$,

y el sistema lineal

$$\begin{cases} u'(t) = A\,u(t) \\ u(0) = u^0 \end{cases}$$

donde ahora $u = (u_1, .., u_N)$, A matriz,

para analizar las dificultades asociadas a los métodos que introduciremos.

Por ejemplo, una dificultad asociada a ciertas ODEs es lo que se denomina su "rigidez" (stiff, en inglés).

stiffness

Consideremos el sistema:

$$\begin{bmatrix} u_1'(t) \\ u_2'(t) \end{bmatrix} = \begin{bmatrix} -50 & 49 \\ 49 & -50 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

con condiciones iniciales $u_1(0) = 2$, $u_2(0) = 0$.

La solución del sistema de EDOs anterior es

$$\begin{cases} u_1(t) = e^{-t} + e^{-99t} \\ u_2(t) = e^{-t} - e^{-99t} \end{cases}$$

Como se aprecia, la solución es suma de dos términos e^{-t} y e^{-99t} que decaen hacia cero pero a velocidades muy diferentes. En efecto, e^{-99t} va a cero en una escala de tiempo 99 veces ~~mayor~~ más ~~qu~~ rápida que e^{-t}.

Curiosamente, 99 es el número de condicionamiento de la matriz del sistema anterior $A = \begin{bmatrix} -50 & 49 \\ 49 & -50 \end{bmatrix}$.

Estas diferentes escalas de tiempo para las dos componentes de la solución se traducen en que para tiempos grandes, e^{-t} será la parte dominante, ~~mientras~~ sin efecto apenas del término e^{-99t}. Sin embargo, para tiempos próximos a cero, el término e^{-99t} sí que va a tener un peso importante en la solución, y para poder aproximar numéricamente esta parte de la solución mediante un método de discretización en tiempo $t_0 = 0 < t_1 < \dots < t_i < t_{i+1} < \dots < t_N = T$ vamos a necesitar un paso $\Delta t = t_{i+1} - t_i$ "muy pequeño".

Gráficamente,

$$u_1(t) = \underbrace{e^{-t}}_{\substack{\text{controla}\\ \text{el decaimiento}\\ \text{de la solución}}} + \underbrace{e^{-99t}}_{\substack{\text{controla el } \Delta t\\ \text{de la aproximación}\\ \text{numérica.}}}$$

Este ejemplo es el prototipo de lo que se llama una ecuación diferencial stiff.

El hecho de tener que tomar $\Delta t << 1$ genera dificultades numéricas importantes que habrá que superar.

La rigidez no es un artificio matemático. Esta situación se da en ecuaciones que modelan la cinética de ciertas reacciones químicas (las cuales se producen a velocidades diferentes), en simulaciones de circuitos eléctricos (donde las distintas componentes del circuito (e.g. R, L, C) responden a distintas escalas de tiempo) o en teoría de control, por citar sólo algunos ejemplos.

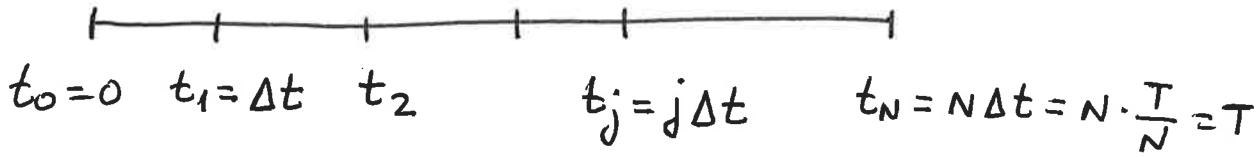
Estudiamos a continuación los principales métodos numéricos de resolución de EDOs.

<u>MÉTODOS DE EULER</u> ($\underbrace{\text{Explícito}}_{\text{Forward}}$ e $\underbrace{\text{Implícito}}_{\text{Backward}}$)

Estudiamos conjuntamente las ecuaciones escalares y los sistemas de EDOs dado que el proceso de discretización es el mismo. Para fijar ideas podemos pensar en ecuaciones escalares en la forma

$$\begin{cases} u'(t) = f(u(t), t), & 0 < t \leq T. \\ u(0) = u_0 \end{cases}$$

Consideramos una discretización del intervalo temporal $t_0 = 0 < t_1 < \ldots < t_N = T$, de modo que $\Delta t = \dfrac{T}{N}$



Así, $t_j = j\,\Delta t$, $j = 0, \ldots, N$.

Denotamos por:

$u_n = u(t_n)$ el valor de la solución exacta en el tiempo t_n.
$U_n = U(t_n)$ " " " " " aproximada " " ".

<u>Método de Euler explícito</u> (forward, en inglés).

Se basa en la aproximación de la derivada primera en la forma $u'(t_n) \simeq \dfrac{u_{n+1} - u_n}{\Delta t}$,

resultando el esquema numérico:

- $U_0 = u_0 = u(\overset{=0}{t_0}) = u^0$ dato del problema
- $\dfrac{U_{n+1} - U_n}{\Delta t} = f(U_n, t_n) \Rightarrow \boxed{U_{n+1} = U_n + \Delta t\, f_n}$

  con $f_n = f(U_n, t_n)$

<u>Ejemplo</u>

$\left\{ \begin{array}{l} \cancel{u'(t) = u(t)} \\ \cancel{u(0) = 1} \end{array} \right.$

Ejemplo

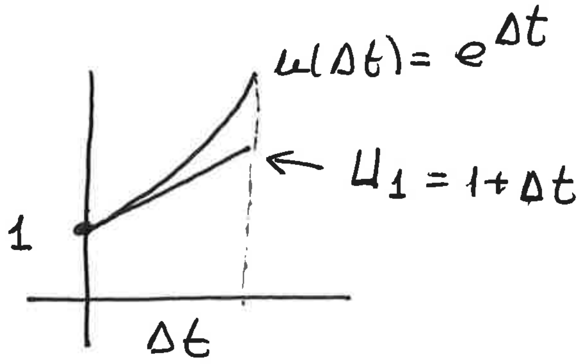
Problema continuo

$$\begin{cases} u'(t) = u(t) \\ u(0) = 1 \end{cases} \Rightarrow \begin{array}{l} \text{Solución } u(t) = e^{t} \\ \text{Solución en } \Delta t: \; u(\Delta t) = e^{\Delta t} \end{array}$$

Problema discreto (Forward Euler)

$$\begin{cases} U_0 = 1 \\ U_{n+1} = U_n + \Delta t \, U_n \end{cases}$$

Solución en Δt: $U_1 = 1 + \Delta t$ ya que $U_0 = 1$.



Nótese que sobre cada intervalo de tiempo Δt, la pendiente de U no cambia, es decir, la solución aproximada U es lineal a trozos.

Método de Euler implícito (Backward, en inglés).

Se basa en la misma aproximación de la derivada primera, es decir, $u'(t_n) \approx \frac{u_{n+1} - u_n}{\Delta t}$, pero el término de la derecha f no se evalúa en el extremo izquierdo t_n, sino en el derecho t_{n+1}, resultando el esquema:

- $U_0 = u_0 = u(t_0 = 0)$
- $\frac{U_{n+1} - U_n}{\Delta t} = f(U_{n+1}, t_{n+1}) \Rightarrow \boxed{U_{n+1} - \Delta t \, f_{n+1} = U_n}$

con $f_{n+1} = f(U_{n+1}, t_{n+1})$.

El método se llama <u>implícito</u> porque para calcular U_{n+1} hemos de resolver la ecuación

$$U_{n+1} - \Delta t\, f_{n+1} = U_n$$

que es lineal si f es lineal; y no lineal si f es no lineal.

Veamos cómo funciona en el caso del ejemplo anterior.

<u>Ejemplo</u>

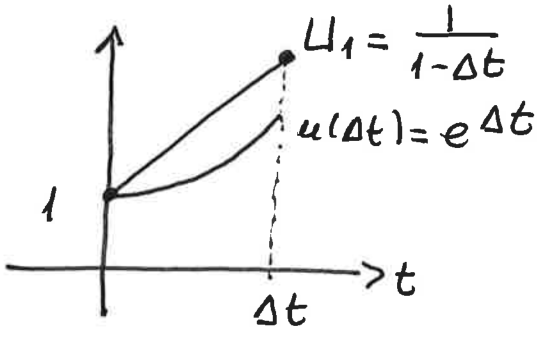
<u>Problema continuo</u>

$$\left\{\begin{array}{l} u'(t) = u(t) \\ u(0) = 1 \end{array}\right. \Longrightarrow \begin{array}{l} \text{Solución } u(t) = e^{t} \\ \text{Solución en } \Delta t: \ u(\Delta t) = e^{\Delta t} \end{array}$$

<u>Problema discreto</u> (Backward Euler)

$$\left\{\begin{array}{l} U_0 = 1 \\ U_{n+1} - \Delta t\, U_{n+1} = U_n \Rightarrow (1-\Delta t) U_{n+1} = U_n \Rightarrow U_{n+1} = \dfrac{U_n}{1-\Delta t} \end{array}\right.$$

$$U_1 = \frac{U_0}{1-\Delta t} = \frac{1}{1-\Delta t}$$



<u>Nota</u>.- Para ambos métodos, en el ejemplo anterior, el error en cada paso es del orden de $\frac{1}{2}(\Delta t)^2$. En efecto:

$$e^{\Delta t} = 1 + \Delta t + \frac{(\Delta t)^2}{2!} + \cdots$$

Euler explícito : $U_1 = 1 + \Delta t$

$$\text{Error} = u_1 - U_1 = 1 + \Delta t + \frac{(\Delta t)^2}{2} + \cdots - (1+\Delta t)$$

$$= \frac{(\Delta t)^2}{2} + \cdots$$

Euler implícito: $U_1 = \frac{1}{1-\Delta t} = 1 + \Delta t + (\Delta t)^2 + \cdots$

$$\text{Error} = u_1 - U_1 = 1 + \Delta t + \frac{(\Delta t)^2}{2} + \cdots - \left(1 + \Delta t + (\Delta t)^2 + \cdots\right)$$

$$= -\frac{(\Delta t)^2}{2} + \cdots \quad \blacksquare$$

Este ejemplo parece indicar que ambos métodos van a tener un buen comportamiento si Δt es pequeño. Pero tomar Δt pequeño va a aumentar el coste computacional de manera muy considerable si T es relativamente grande.

Vamos a examinar o analizar Δt en el siguiente ejemplo.

Para $a < 0$, consideremos el problema

$$\begin{cases} u'(t) = a\, u(t) \\ u(0) = 1 \end{cases}$$

cuya solución explícita $u(t) = e^{at}$ decae rápidamente a cero pues $a < 0$.

Euler explícito es $U_{n+1} = (1 + a\Delta t) U_n$ que conduce a

$$\boxed{U_n = (1 + a\Delta t)^n U_0}$$

Euler implícito es $(1 - a\Delta t) U_{n+1} = U_n$ que conduce a

$$\boxed{U_n = (1 - a\Delta t)^{-n} U_0}$$

Como decíamos antes, la solución exacta cumple

$$u(t) = e^{at} \xrightarrow[t \to +\infty]{} 0 \quad \text{porque } a < 0.$$

Euler implícito cumple esa misma propiedad sea cual sea $\Delta t > 0$. En efecto:

$$u_n = (\underbrace{1 - a\Delta t}_{>1})^{-n} u_0 \xrightarrow[n \to \infty]{} 0$$

Decimos entonces que <u>Euler implícito es incondicionalmente estable</u>.

Sin embargo, para Euler explícito, si Δt es grande, de modo que $1 + a\Delta t < -1$, entonces

$$u_n = (1 + a\Delta t)^n u_0 \xrightarrow[n \to \infty]{} \infty$$

La condición $1 + a\Delta t < -1$, se traduce en

$a\Delta t < -2 \Rightarrow \Delta t > -\frac{2}{a}$, es decir, si Δt supera el umbral $-\frac{2}{a}$, entonces la solución que proporciona Euler explícito es inestable.

Por ejemplo, si $a = -20 \Rightarrow \cancel{\Delta t}\ -\frac{2}{a} = \frac{2}{20} = \frac{1}{10}$.

- Si tomamos $\Delta t = \frac{1}{11} \Rightarrow \underbrace{(1 + 20\Delta t)^{22}}_{\left(-\frac{9}{11}\right)^{22}} \approx 0.012$, la solución es estable

- Si tomamos $\Delta t = \frac{1}{9} \Rightarrow \underbrace{(1 + 20\Delta t)^{18}}_{\left(-\frac{11}{9}\right)^{18}} \approx 37.043$ la solución es inestable.

Por tanto, el método de Euler explícito es condicionalmente estable, es decir, es estable a condición de que

$$\Delta t < -\frac{2}{a}.$$

Este ejemplo pone de manifiesto que una aproximación de calidad necesita de un método estable. Sin embargo, la estabilidad no es suficiente pues incluso para $\Delta t = \frac{1}{11}$, que cumple la condición de estabilidad si $a=-20$, se tiene que las potencias $(1+20\Delta t)^n = \left(-\frac{9}{11}\right)^n$ cambian de signo mientras que la solución exacta $u(t)=e^{at}$ siempre es positiva.

Vamos a analizar con más detalle la propagación de los errores en el método de Euler.

Anteriormente hemos visto que tanto en Euler explícito como implícito el error de discretización en la primera iteración $DE_1 = u_1 - U_1 = O((\Delta t)^2)$.

Veamos cómo se propaga este error hasta el tiempo final T, donde $T = N\,\Delta t$.

Para Euler explícito y para la ecuación $u' = f(u,t) = au$, se tiene que

$$U_{n+1} = U_n + \Delta t\, f(U_n, t_n) = U_n + a\Delta t\, U_n = (1+a\Delta t)U_n$$

La solución exacta cumple, desarrollando por Taylor,

$$u_{n+1} = u_n + \Delta t\, u_n' + \underbrace{DE}_{\substack{\text{términos}\\ \text{restantes en}\\ \text{Taylor}}} = u_n + a\Delta t\, u_n + DE_{n+1}.$$

Por tanto, el error $u - U$ cumple

$$e_{n+1} = u_{n+1} - U_{n+1} = e_n + a\Delta t\, e_n + DE_{n+1}$$

$$= (1+a\Delta t)\, e_n + DE_{n+1}.$$

En el tiempo final $T = N\Delta t$ se tiene:

$$e_N = (1+a\Delta t)\, e_{N-1} + DE_N$$

$$= (1+a\Delta t)\left((1+a\Delta t)\, e_{N-2} + DE_{N-1} \right) + DE_N$$

$$= (1+a\Delta t)^2 e_{N-2} + (1+a\Delta t) DE_{N-1} + DE_N$$

$$= \cdots\cdots$$

$$= (1+a\Delta t)^{N-1} \underbrace{e_1}_{u_1 - U_1 = DE_1} + (1+a\Delta t)^{N-2} DE_2 + \cdots + DE_N.$$

$$= (1+a\Delta t)^{N-1} DE_1 + \cdots + DE_N.$$

Si Δt está en la región de estabilidad, $|1+a\Delta t| < 1$ y por tanto,

$$|e_N| = |u_N - U_N| \leq N c (\Delta t)^2 = c T \Delta t$$

donde $|(1+a\Delta t)| \leq c$ y recordemos $N\Delta t = T$.

Es decir, la condición de estabilidad garantiza que los errores locales de discretización no exploten, resultando en un error en el tiempo final del orden de $O(\Delta t)$.

En resumen, el error local de los métodos de Euler es $O((\Delta t)^2)$ pero el error global es $O(\Delta t)$.

Incluso para ser más precisos con el error local de discretización éste es aproximadamente

$$DE \approx c(\Delta t)^2 u''$$

Nota.– Para ecuaciones no lineales, la clave en la resta $u - U$ es:

$$U_{n+1} = U_n + \Delta t\, f(U_n, t_n)$$

$$u_{n+1} = u_n + \Delta t\, \underbrace{f(u_n, t_n)}_{u_n'} + DE.$$

$$u_{n+1} - U_{n+1} = u_n - U_n + \Delta t\, \underbrace{(f(u_n, t_n) - f(U_n, t_n))}_{\leq\; L(u_n - U_n)} + DE$$

Condición de Lipschitz sobre f.

Esta condición de Lipschitz garantiza la estabilidad de los métodos de Euler, y por tanto, su convergencia.

En resumen, ventajas e inconvenientes de los métodos de Euler explícito e implícito.

- Euler explícito es más fácil, y barato, de programar que Euler implícito.
- Euler explícito no se debe usar en problemas rígidos, salvo que se conozca la región de estabilidad, pues una mala elección de Δt provoca inestabilidades numéricas.
- Euler implícito es más caro pues involucra la resolución de una ecuación lineal o nolineal. Sobre todo en este último caso de no linealidad es donde hemos de resolver una ecuación algebraica (o sistema de ecuaciones) en cada iteración.

- Ambos son métodos de primer orden, es decir, el error entre la solución exacta y la numérica en el tiempo final es del orden de Δt.

Los esquemas numéricos son:

Euler explícito: $u_{n+1} = u_n + \Delta t f_n$, $\quad f_n = f(u_n, t_n)$, u_0 dato.

Euler implícito: $u_{n+1} - \Delta t f_{n+1} = u_n$, $\quad f_{n+1} = f(u_{n+1}, t_{n+1})$, u_0 dato.

MÉTODO DE CRANK-NICOLSON

Esquema:
$$\begin{cases} \dfrac{U_{n+1} - U_n}{\Delta t} = \dfrac{1}{2} (f_n + f_{n+1}). \\ U_0 \text{ dato} \\ f_n = f(U_n, t_n) \\ f_{n+1} = f(U_{n+1}, t_{n+1}) \end{cases}$$

Se trata de un método implícito, incondicionalmente estable y de segundo orden, es decir, $| u_N - U_n | = O((\Delta t)^2)$.

Este método mejora la precisión de los métodos de Euler. Su única desventaja es que es un método implícito.

Una clase de métodos explícitos y que ~~se~~ son altamente competitivos cuando el coste de evaluar $f(u, t)$ no es muy caro son los llamados métodos de Runge-Kutta.

MÉTODOS DE RUNGE-KUTTA

La idea básica de estos métodos consiste en componer, usando Euler explícito, dentro de f.

Veamos cómo esta idea nos permite transformar el método de Crank-Nicolson (implícito) en el llamado Método de Runge-Kutta de orden 2, (explícito), manteniendo la precisión $O((\Delta t)^2)$.

Si partimos del esquema de Crank-Nicolson

$$u_{n+1} = u_n + \frac{\Delta t}{2}\left[f(u_n, t_n) + f(u_{n+1}, t_{n+1})\right]$$

y utilizamos el método de Euler explícito para aproximar $u_{n+1} = u_n + \Delta t\, f(u_n, t_n)$

que aparece en el término de la derecha anterior llegamos a

$$u_{n+1} = u_n + \frac{1}{2}\left[\Delta t\, f(u_n, t_n) + \Delta t\, f(u_n + \Delta t\, f(u_n, t_n), t_{n+1})\right]$$

esquema que se suele escribir de manera más sencilla introduciendo las notaciones

$$k_1 = \Delta t\, f(u_n, t_n)$$

$$k_2 = \Delta t\, f(u_n + k_1, t_{n+1})$$

que conducen al llamado Método de Runge-Kutta de orden 2:

$$\boxed{u_{n+1} = u_n + \frac{1}{2}(k_1 + k_2)} \qquad (RK2)$$

Se trata de un método explícito, de orden 2, es decir

$$|u_N - u_n| = O((\Delta t)^2)$$

pero condicionalmente estable. Para el caso del problema modelo $u' = a u$, $\boxed{\Delta t < -\frac{2}{a} \quad , a < 0}$

Esta misma idea se utiliza para obtener toda una variedad de métodos de Runge-Kutta con precisión cada vez mayor, todos ellos explícitos y condicionalmente estables. De entre todos ellos, el más utilizado es el método de Runge-Kutta de orden 4. Para deducirlo, se parte de la EDO

$$u'(t) = f(u(t), t))$$

y se integra entre t_n y t_{n+1}

$$\underbrace{\int_{t_n}^{t_{n+1}} u'(s)\,ds}_{\parallel} = \int_{t_n}^{t_{n+1}} f(u(s), s)\,ds$$

$\int\int$ ← Fórmula de Simpson

$$u(t_{n+1}) - u(t_n) = \frac{\Delta t}{6}\left(f(u_n, t_n) + 4 f\left(\frac{u_{n+1}+u_n}{2\cancel{k}}, \frac{\Delta t}{2}\right) + f(u_{n+1}, t_{n+1})\right).$$

Aproximando los términos $\frac{u_{n+1}+u_n}{2}$ y u_{n+1} mediante fórmulas explícitas, tipo Euler explícito, se llega al esquema:

$$\cancel{K_1 = \Delta t\, f(U_n, t_n)}$$

$$K_2 = \Delta t\, f(t_n + \tfrac{\Delta t}{2},$$

$$\boxed{\begin{aligned} k_1 &= \frac{1}{2} f(u_n, t_n) \\ k_2 &= \frac{1}{2} f(u_n + \Delta t\, k_1, t_{n+\frac{1}{2}}) \\ k_3 &= \frac{1}{2} f(u_n + \Delta t\, k_2, t_{n+\frac{1}{2}}) \\ k_4 &= \frac{1}{2} f(u_n + 2\Delta t\, k_3, t_{n+1}) \\ u_{n+1} &= u_n + \frac{\Delta t}{3}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}} \qquad (RK4)$$

La precisión de (RK4) es del orden de $O((\Delta t)^4)$.

Como antes, es preciso tener en cuenta Δt, que ha de ser pequeño, pues es un método explícito y por tanto condicionalmente estable.