Simulación de Sistemas Continuos y a Tramos

Prof. Dr. François E. Cellier
Institut für Computational Science
ETH Zürich

25 de junio 2007

Modelos en el Espacio del Estado

Los modelos dinámicos con parámetros concentrados se representan usualmente por ecuaciones diferenciales ordinarias del primer orden. Se habla de *modelos en el espacio del estado*.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

donde $\mathbf{x}$ es el *vector de las variables del estado*, $\mathbf{u}$ es el *vector de las entradas* y t denota el *tiempo*, la variable independiente sobre la cual queremos simular.

Se necesitan también *condiciones iniciales* para las variables del estado:

$$\mathbf{x}(t = t_0) = \mathbf{x_0}$$

Las Series de Taylor

El modelo puede simularse usando una *serie de Taylor*. Conociendo la solución en un instante, t^*, la solución puede calcularse en otro instante más tarde, $t^* + h$ usando una expansión de Taylor:

$$x_i(t^* + h) = x_i(t^*) + \frac{dx_i(t^*)}{dt} \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \dots$$

El modelo en el estado se usa en la serie de Taylor reemplazando la primera derivada:

$$x_i(t^* + h) = x_i(t^*) + f_i(t^*) \cdot h + \frac{df_i(t^*)}{dt} \cdot \frac{h^2}{2!} + \dots$$

Los diferentes métodos de integración numérica se distinguen entre sí en sus aproximaciones numéricas de las derivadas de f.

El Error por Truncamiento

Es cierto que no pueden añadirse todas las contribuciones de la expansión de Taylor. Todos los métodos de la integración numérica solamente aproximan un cierto número de los términos de la serie de Taylor. Ese número puede ser fijo o variable.

Se habla del *orden de la aproximación* del método numérico. Un algoritmo que aproxima la serie de Taylor de la manera:

$$x_i(t^* + h) = x_i(t^*) + f_i(t^*) \cdot h + \frac{df_i(t^*)}{dt} \cdot \frac{h^2}{2!} + \frac{d^2 f_i(t^*)}{dt^2} \cdot \frac{h^3}{3!} + o(h^4)$$

es un *algoritmo del tercer orden*.

El *error por truncamiento* del método crece proporcional con la cuarta potencia del *paso de integración*, h.

El Error por Redondeo

Existe un segundo tipo de error que resulta de la mantisa finita del ordenador. Los efectos de ese tipo de error pueden ilustrarse fácilmente de forma gráfica:

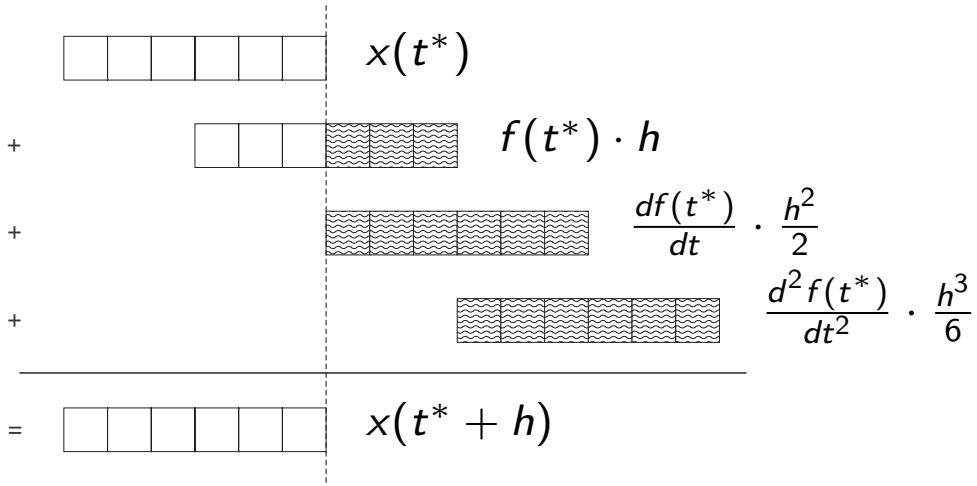


Figure: Efectos del *error por redondeo* en precisión regular

El Error por Redondeo II

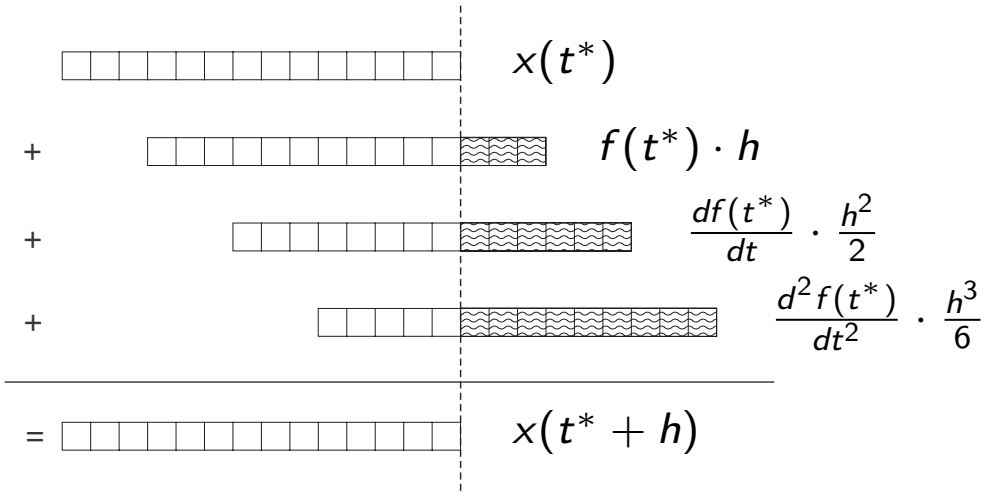


Figure: Efectos del *error por redondeo* en doble precisión

El Error por Redondeo III

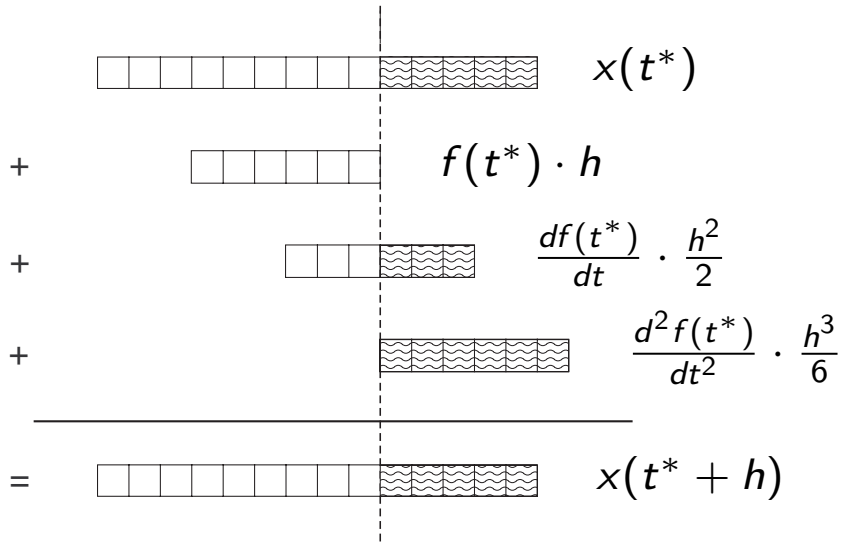


Figure: Efectos del *error por redondeo* en precisión "1.5"

La Integración Explícita de Euler

El método numérico más simple es el método explícito de Euler *"Forward Euler" (FE)*, un método de primer orden:

$$\mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \dot{\mathbf{x}}(t^*) \cdot h$$

$$\Rightarrow \quad \mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \mathbf{f}(\mathbf{x}(t^*), t^*) \cdot h$$

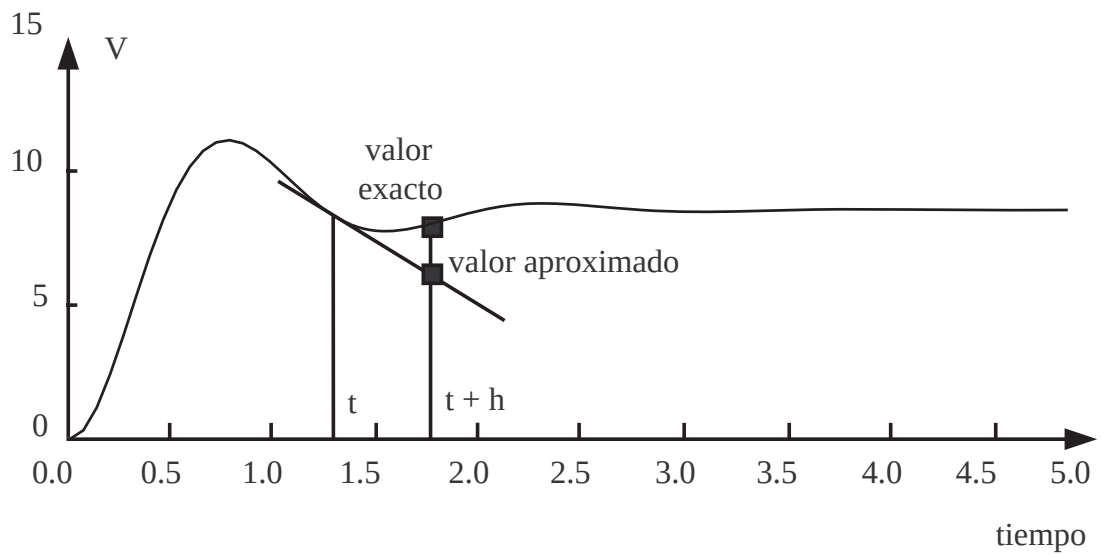


Figure: La integración numérica usando el método "FE"

La Integración Explícita de Euler II

Usando *métodos explícitos*, la simulación no necesita ninguna *iteración* dentro de un paso si el modelo no contiene *bucles algebraicos*:

$$
\begin{array}{lll}
\text{paso 1a:} & \dot{\mathbf{x}}(t_0) & = \mathbf{f}(\mathbf{x}(t_0), t_0) \\
\text{paso 1b:} & \mathbf{x}(t_0 + h) & = \mathbf{x}(t_0) + h \cdot \dot{\mathbf{x}}(t_0) \\
 & & \\
\text{paso 2a:} & \dot{\mathbf{x}}(t_0 + h) & = \mathbf{f}(\mathbf{x}(t_0 + h), t_0 + h) \\
\text{paso 2b:} & \mathbf{x}(t_0 + 2h) & = \mathbf{x}(t_0 + h) + h \cdot \dot{\mathbf{x}}(t_0 + h) \\
 & & \\
\text{paso 3a:} & \dot{\mathbf{x}}(t_0 + 2h) & = \mathbf{f}(\mathbf{x}(t_0 + 2h), t_0 + 2h) \\
\text{paso 3b:} & \mathbf{x}(t_0 + 3h) & = \mathbf{x}(t_0 + 2h) + h \cdot \dot{\mathbf{x}}(t_0 + 2h)
\end{array}
$$

etc.

La Integración Implícita de Euler

Otro método numérico de primer orden es el método implícito de Euler *"Backward Euler" (BE)*:

$$\mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \mathbf{f}(\mathbf{x}(t^* + h), t^* + h) \cdot h$$

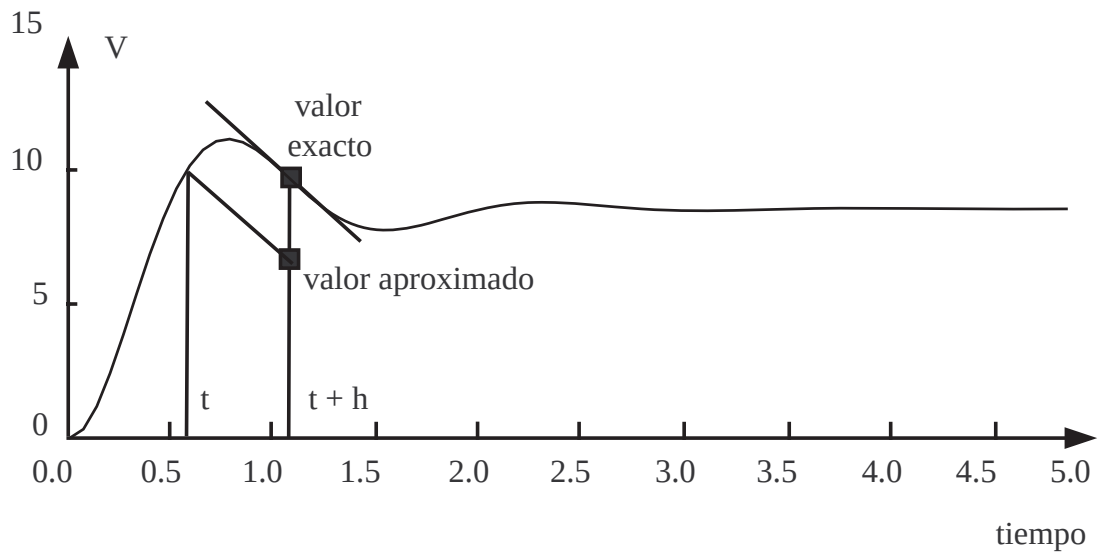


Figure: La integración numérica usando el método "BE"

El Dominio de la Estabilidad Numérica

Un sistema *lineal*, *autónomo* e *invariante con el tiempo* puede representarse usando la notación:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \quad ; \quad \mathbf{x}(t = t_0) = \mathbf{x_0}$$

con la solución analítica:

$$\mathbf{x}(t) = \exp(\mathbf{A} \cdot t) \cdot \mathbf{x_0}$$

La solución es *analíticamente estable* si:

$$\mathbb{R}\mathrm{e}\{\mathrm{Eig}(\mathbf{A})\} = \mathbb{R}\mathrm{e}\{\lambda\} < 0.0$$

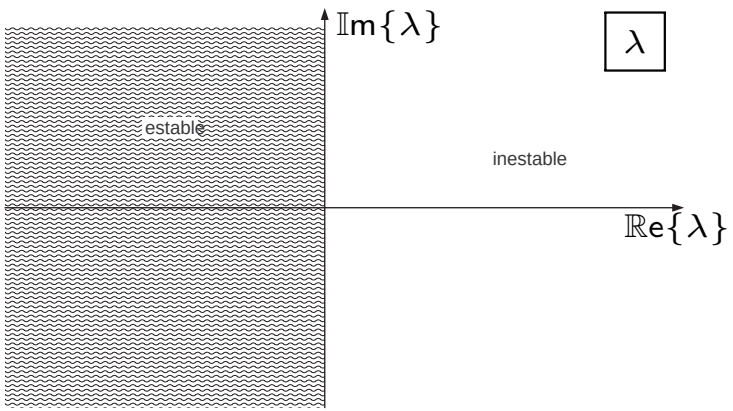


Figure: El dominio de la *estabilidad analítica*

El Dominio de la Estabilidad Numérica II

Usando el *algoritmo FE*:

$$\mathbf{x}(t^* + h) = \mathbf{x}(t^*) + \mathbf{f}(\mathbf{x}(t^*), t^*) \cdot h$$
$$\Rightarrow \quad \mathbf{x}(t^* + h) = \mathbf{x}(t^*) + \mathbf{A} \cdot h \cdot \mathbf{x}(t^*)$$
$$\Rightarrow \quad \mathbf{x}(k+1) = [\mathbf{I}^{(n)} + \mathbf{A} \cdot h] \cdot \mathbf{x}(k)$$

Entonces:

$$\mathbf{x_{k+1}} = \mathbf{F} \cdot \mathbf{x_k}$$

con:

$$\mathbf{F} = \mathbf{I}^{(n)} + \mathbf{A} \cdot h$$

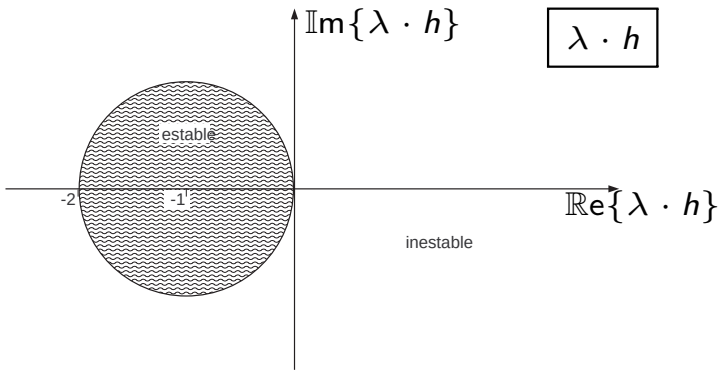


Figure: El dominio de la *estabilidad numérica del algoritmo FE*

La Simulación con FE

Simulando el sistema lineal escalar:

$$\dot{x} = a \cdot x \quad ; \quad x(t = t_0) = x_0$$

usando el algoritmo FE obtenemos:

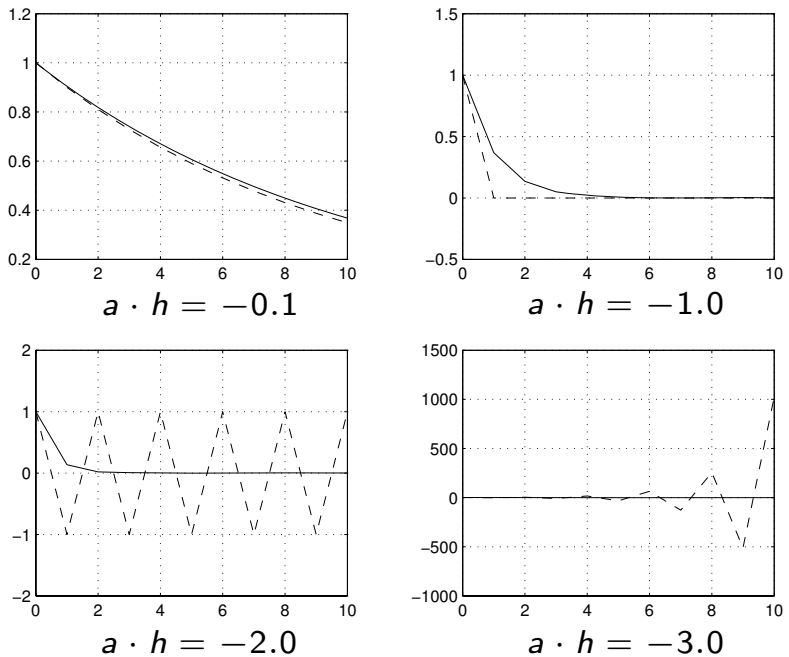


Figure: Simulación de un sistema lineal escalar con el algoritmo FE

Cálculo del Paso Máximo de la Integración Numérica con FE

Dado un sistema lineal de segundo orden con dos autovalores complejos, λ_1 y λ_2:

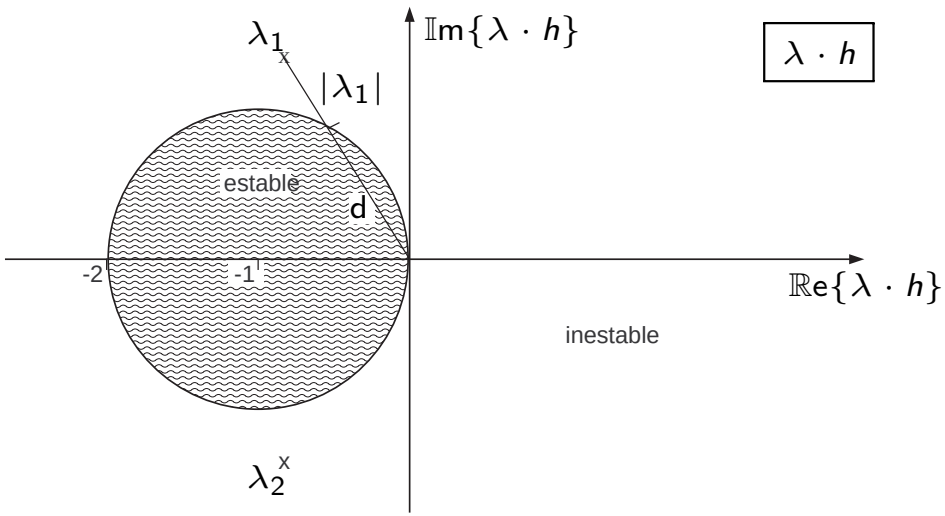


Figure: Paso máximo de la integración numérica con FE

$$\Rightarrow \quad h_{max} = \frac{d}{|\lambda_1|}$$

El Dominio de la Estabilidad Numérica III

Usando el *algoritmo BE*:

$$\mathbf{x}(t^* + h) = \mathbf{x}(t^*) + \mathbf{A} \cdot h \cdot \mathbf{x}(t^* + h)$$
$$\Rightarrow \quad [\mathbf{I}^{(\mathbf{n})} - \mathbf{A} \cdot h] \cdot \mathbf{x}(t^* + h) = \mathbf{x}(t^*)$$
$$\Rightarrow \quad \mathbf{x}(k+1) = [\mathbf{I}^{(\mathbf{n})} - \mathbf{A} \cdot h]^{-1} \cdot \mathbf{x}(k)$$

Entonces:

$$\mathbf{F} = [\mathbf{I}^{(\mathbf{n})} - \mathbf{A} \cdot h]^{-1}$$

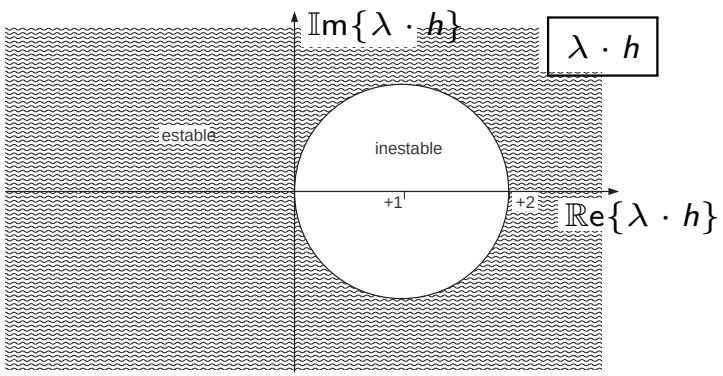


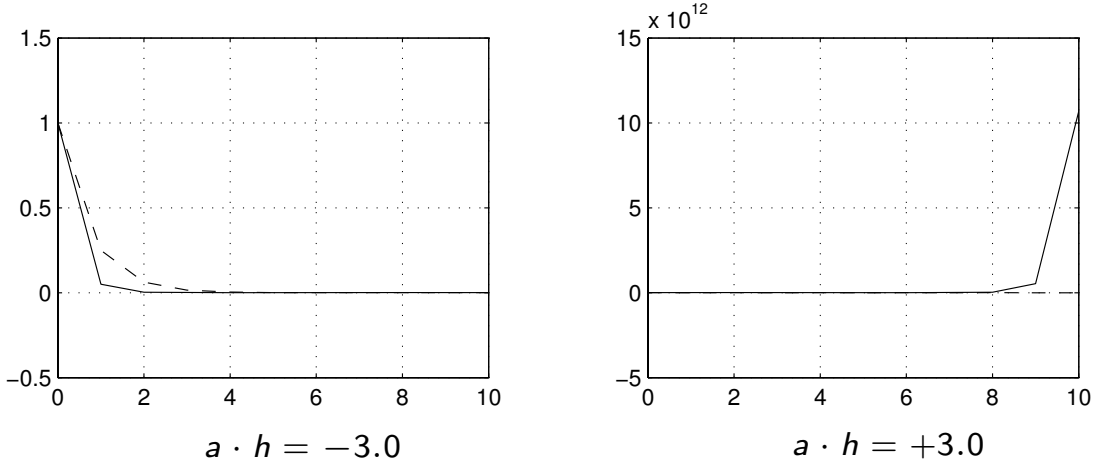
Figure: El dominio de la *estabilidad numérica del algoritmo BE*

La Simulación con BE

Simulando el sistema lineal escalar:

$$\dot{x} = a \cdot x \quad ; \quad x(t = t_0) = x_0$$

usando el algoritmo BE obtenemos:



$a \cdot h = -3.0$ $a \cdot h = +3.0$

Figure: Simulación de un sistema lineal escalar con el algoritmo BE

La Iteración de Punto Fijo

Si estamos usando métodos implícitos para la integración numérica tenemos que iterar en cada paso.

Una posible manera de iterar es usando el concepto de una *predicción* con muchas *correcciones*.

predicción:
$$\begin{aligned} \dot{\mathbf{x}}_k &= \mathbf{f}(\mathbf{x}_k, t_k) \\ \mathbf{x}^{\mathbf{P}}_{k+1} &= \mathbf{x}_k + h \cdot \dot{\mathbf{x}}_k \end{aligned}$$

1^{st} corrección:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{P}}_{k+1} &= \mathbf{f}(\mathbf{x}^{\mathbf{P}}_{k+1}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C1}}_{k+1} &= \mathbf{x}_k + h \cdot \dot{\mathbf{x}}^{\mathbf{P}}_{k+1} \end{aligned}$$

2^{nd} corrección:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{C1}}_{k+1} &= \mathbf{f}(\mathbf{x}^{\mathbf{C1}}_{k+1}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C2}}_{k+1} &= \mathbf{x}_k + h \cdot \dot{\mathbf{x}}^{\mathbf{C1}}_{k+1} \end{aligned}$$

3^{rd} corrección:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{C2}}_{k+1} &= \mathbf{f}(\mathbf{x}^{\mathbf{C2}}_{k+1}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C3}}_{k+1} &= \mathbf{x}_k + h \cdot \dot{\mathbf{x}}^{\mathbf{C2}}_{k+1} \end{aligned}$$

etc.

Ese método de iteración se llama la *iteración de punto fijo*.

La Iteración de Punto Fijo II

Aplicando la iteración de punto fijo al sistema lineal, obtenemos:

$$\begin{aligned}
\mathbf{F^P} &= \mathbf{I^{(n)}} + \mathbf{A}\cdot h \\
\mathbf{F^{C1}} &= \mathbf{I^{(n)}} + \mathbf{A}\cdot h + (\mathbf{A}\cdot h)^2 \\
\mathbf{F^{C2}} &= \mathbf{I^{(n)}} + \mathbf{A}\cdot h + (\mathbf{A}\cdot h)^2 + (\mathbf{A}\cdot h)^3 \\
\mathbf{F^{C3}} &= \mathbf{I^{(n)}} + \mathbf{A}\cdot h + (\mathbf{A}\cdot h)^2 + (\mathbf{A}\cdot h)^3 + (\mathbf{A}\cdot h)^4
\end{aligned}$$

Con un número infinito de iteraciones obtenemos:

$$\mathbf{F} = \mathbf{I^{(n)}} + \mathbf{A}\cdot h + (\mathbf{A}\cdot h)^2 + (\mathbf{A}\cdot h)^3 + \ldots$$

Entonces:

$$(\mathbf{A}\cdot h)\cdot\mathbf{F} = \mathbf{A}\cdot h + (\mathbf{A}\cdot h)^2 + (\mathbf{A}\cdot h)^3 + (\mathbf{A}\cdot h)^4 + \ldots$$

Con la sustracción:

$$[\mathbf{I^{(n)}} - \mathbf{A}\cdot h]\cdot\mathbf{F} = \mathbf{I^{(n)}}$$

y por eso:

$$\mathbf{F} = [\mathbf{I^{(n)}} - \mathbf{A}\cdot h]^{-1}$$

Aparentemente se obtiene la misma matriz $\mathbf{F}$ que obtuvimos en el caso del algoritmo BE.

La Iteración de Punto Fijo III

Dibujamos el dominio de la estabilidad numérica de ese algoritmo:

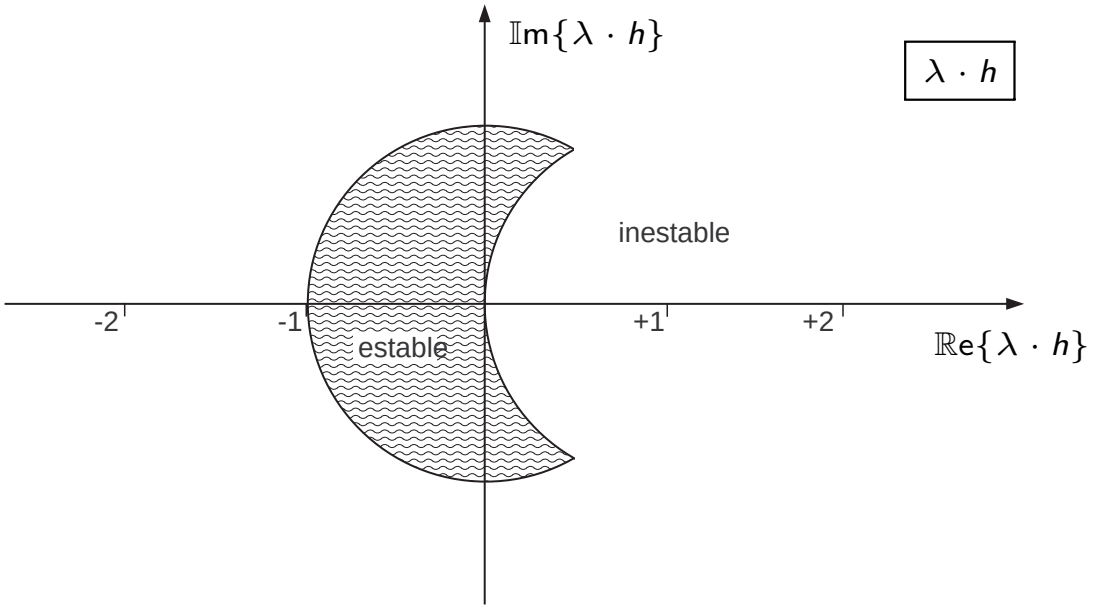


Figure: El dominio de la *estabilidad numérica del algoritmo con iteración*

La Iteración de Punto Fijo III

Dibujamos el dominio de la estabilidad numérica de ese algoritmo:

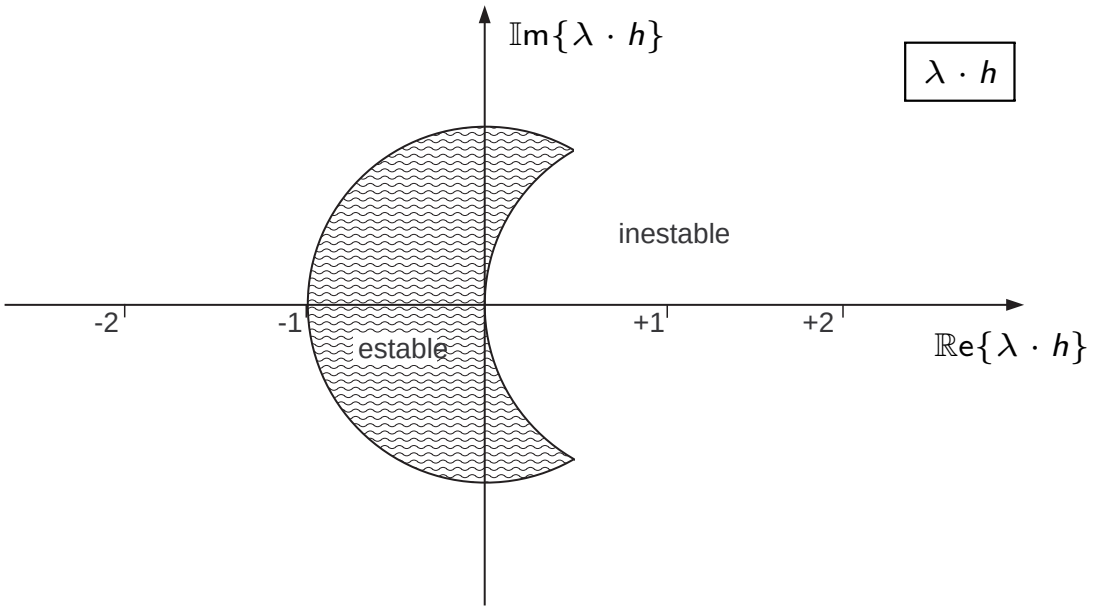


Figure: El dominio de la *estabilidad numérica del algoritmo con iteración*

Aparentemente no nos funcionó muy bien.

La Iteración de Punto Fijo IV

¿Porqué no funcionó?

No funcionó porque la serie:

$$\mathbf{F} = \mathbf{I}^{(\mathbf{n})} + \mathbf{A} \cdot h + (\mathbf{A} \cdot h)^2 + (\mathbf{A} \cdot h)^3 + \dots$$

solamente tiene convergencia si todos los autovalores de $\mathbf{A} \cdot h$ se encuentran dentro del *círculo unitario*. Si no es el caso, la sustracción no es válida.

Dentro del círculo unitario el dominio de la estabilidad numérica del método es idéntico con el dominio del algoritmo BE, pero fuera del círculo unitario no hay estabilidad numérica.

Por eso, el *algoritmo con iteración de punto fijo* no sirve para nada.

La Iteración de Newton

La iteración de Newton puede usarse para encontrar donde una función se hace cero:

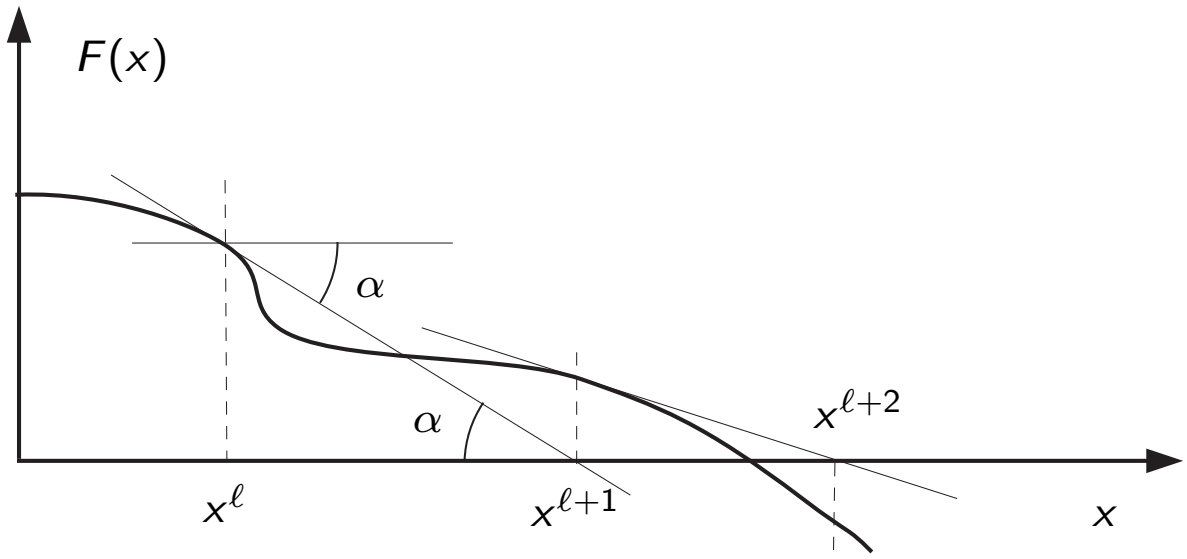


Figure: La iteración de Newton

$$\tan\ \alpha = \frac{\partial \mathcal{F}^\ell}{\partial x} = \frac{\mathcal{F}^\ell}{x^\ell - x^{\ell+1}} \quad \Rightarrow \quad x^{\ell+1} = x^\ell - \frac{\mathcal{F}^\ell}{\partial \mathcal{F}^\ell / \partial x}$$

La Iteración de Newton II

El método BE aplicada a una ecuación diferencial escalar puede formularse de la manera siguiente:

$$x_{k+1} = x_k + h \cdot f(x_{k+1}, t_{k+1})$$

Entonces:

$$\mathcal{F}(x_{k+1}) = x_k + h \cdot f(x_{k+1}, t_{k+1}) - x_{k+1} = 0.0$$

Ahora puede aplicarse la iteración de Newton:

$$x_{k+1}^{\ell+1} = x_{k+1}^{\ell} - \frac{x_k + h \cdot f(x_{k+1}^{\ell}, t_{k+1}) - x_{k+1}^{\ell}}{h \cdot \partial f(x_{k+1}^{\ell}, t_{k+1})/\partial x - 1.0}$$

La Iteración de Newton III

En el caso de múltiples variables del estado:

$$\mathbf{x}^{\ell+1} = \mathbf{x}^{\ell} - \left(\mathcal{H}^{\ell}\right)^{-1} \cdot \mathcal{F}^{\ell}$$

donde:

$$\mathcal{H} = \frac{\partial \mathcal{F}}{\partial \mathbf{x}} = \begin{pmatrix} \partial \mathcal{F}_1/\partial x_1 & \partial \mathcal{F}_1/\partial x_2 & \dots & \partial \mathcal{F}_1/\partial x_n \\ \partial \mathcal{F}_2/\partial x_1 & \partial \mathcal{F}_2/\partial x_2 & \dots & \partial \mathcal{F}_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial \mathcal{F}_n/\partial x_1 & \partial \mathcal{F}_n/\partial x_2 & \dots & \partial \mathcal{F}_n/\partial x_n \end{pmatrix}$$

es el *Hessiano* de la iteración.

La Iteración de Newton IV

Evaluando el Hessiano del algoritmo BE:

$$\mathbf{x}_{\mathbf{k+1}}^{\ell+\mathbf{1}} = \mathbf{x}_{\mathbf{k+1}}^{\ell} - [h \cdot \mathcal{J}_{\mathbf{k+1}}^{\ell} - \mathbf{I}^{(\mathbf{n})}]^{-1} \cdot [\mathbf{x}_{\mathbf{k}} + h \cdot \mathbf{f}(\mathbf{x}_{\mathbf{k+1}}^{\ell}, t_{k+1}) - \mathbf{x}_{\mathbf{k+1}}^{\ell}]$$

donde:

$$\mathcal{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \dots & \partial f_1/\partial x_n \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \dots & \partial f_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \dots & \partial f_n/\partial x_n \end{pmatrix}$$

es el *Jacobiano* del sistema dinámico.

La Iteración de Newton V

Si el sistema es lineal:

$$\mathcal{J} = \mathbf{A}$$

Entonces:

$$\mathbf{x}_{\mathbf{k+1}}^{\ell+1} = \mathbf{x}_{\mathbf{k+1}}^{\ell} - [\mathbf{A} \cdot h - \mathbf{I}^{(\mathbf{n})}]^{-1} \cdot [(\mathbf{A} \cdot h - \mathbf{I}^{(\mathbf{n})}) \cdot \mathbf{x}_{\mathbf{k+1}}^{\ell} + \mathbf{x}_{\mathbf{k}}]$$

$$\Rightarrow \qquad \mathbf{x}_{\mathbf{k+1}}^{\ell+1} = [\mathbf{I}^{(\mathbf{n})} - \mathbf{A} \cdot h]^{-1} \cdot \mathbf{x}_{\mathbf{k}}$$

La *iteración de Newton* nunca cambia el dominio de la estabilidad numérica. Es así no solamente para el algoritmo BE sino para todos los algoritmos implícitos de la integración numérica.

Conclusiones

Conclusiones

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos siempre tiene que considerarse la *estabilidad numérica del algoritmo*.

Conclusiones

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos siempre tiene que considerarse la *estabilidad numérica del algoritmo*.
- La estabilidad numérica de cada algoritmo puede representarse por un *dominio de estabilidad numérica* dibujado en el plano complejo $\lambda \cdot h$.

Conclusiones

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos siempre tiene que considerarse la *estabilidad numérica del algoritmo*.
- La estabilidad numérica de cada algoritmo puede representarse por un *dominio de estabilidad numérica* dibujado en el plano complejo $\lambda \cdot h$.
- El análisis de la estabilidad numérica se hace normalmente para *sistemas lineales*, *autónomos*, e *invariantes con el tiempo*.

Conclusiones

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos siempre tiene que considerarse la *estabilidad numérica del algoritmo*.
- La estabilidad numérica de cada algoritmo puede representarse por un *dominio de estabilidad numérica* dibujado en el plano complejo $\boxed{\lambda \cdot h}$.
- El análisis de la estabilidad numérica se hace normalmente para *sistemas lineales*, *autónomos*, e *invariantes con el tiempo*.
- Existe también una *teoría de la estabilidad numérica no lineal*, pero es bastante complicada y no es necesario usarla, porque la estabilidad numérica de un sistema linealizado es la misma que la estabilidad numérica del sistema original no lineal.

Conclusiones II

Conclusiones II

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos también tiene que considerarse la *precisión del algoritmo*.

Conclusiones II

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos también tiene que considerarse la *precisión del algoritmo*.
- La precisión de algoritmos numéricos es influida por varios errores, como el *error por truncamiento*, el *error por redondeo*, y el *error por acumulación*.

Conclusiones II

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos también tiene que considerarse la *precisión del algoritmo*.
- La precisión de algoritmos numéricos es influida por varios errores, como el *error por truncamiento*, el *error por redondeo*, y el *error por acumulación*.
- El más importante entre ellos es el *error por truncamiento* que se caracteriza por el *orden de aproximación* del método.

Conclusiones II

- En el análisis de algoritmos para la integración numérica de sistemas dinámicos también tiene que considerarse la *precisión del algoritmo*.
- La precisión de algoritmos numéricos es influida por varios errores, como el *error por truncamiento*, el *error por redondeo*, y el *error por acumulación*.
- El más importante entre ellos es el *error por truncamiento* que se caracteriza por el *orden de aproximación* del método.
- Es importante evaluar el orden de aproximación para sistemas *no lineales* y *vectoriales* porque puede pasar que el orden de aproximación de un método es más alto en el caso de un sistema lineal o en el caso de un sistema de primer orden.