Simulación de Sistemas Continuos y a Tramos

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

27 de junio 2007

Ecuaciones Diferenciales Algebraicas

Un modelo descrito por *ecuaciones del estado* no es la forma normal en la cual se presenta un modelo de un sistema dinámico inicialmente.

Casi siempre se obtiene una *mezcla de ecuaciones diferenciales y algebraicas*. Además, esas ecuaciones se encuentran de forma implícita:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

Entonces tenemos que *iterar en cada evaluación del modelo* para obtener los valores de las derivadas.

Los Tres Bucles de Iteración

Integrando un modelo implícito de un sistema dinámico usando un algoritmo implícito con control del paso y del orden, encontramos *tres bucles de iteración*.

Los Tres Bucles de Iteración

Integrando un modelo implícito de un sistema dinámico usando un algoritmo implícito con control del paso y del orden, encontramos *tres bucles de iteración*.

- Sobre el nivel más bajo tenemos que iterar durante cada evaluación del modelo para obtener los valores de las derivadas. Esta iteración sale muy cara, porque se hace más frecuentemente.

Los Tres Bucles de Iteración

Integrando un modelo implícito de un sistema dinámico usando un algoritmo implícito con control del paso y del orden, encontramos *tres bucles de iteración*.

- Sobre el nivel más bajo tenemos que iterar durante cada evaluación del modelo para obtener los valores de las derivadas. Esta iteración sale muy cara, porque se hace más frecuentemente.
- Sobre un nivel más alto tenemos que iterar sobre cada paso de integración porque se usa un método implícito para la integración.

Los Tres Bucles de Iteración

Integrando un modelo implícito de un sistema dinámico usando un algoritmo implícito con control del paso y del orden, encontramos *tres bucles de iteración*.

- Sobre el nivel más bajo tenemos que iterar durante cada evaluación del modelo para obtener los valores de las derivadas. Esta iteración sale muy cara, porque se hace más frecuentemente.
- Sobre un nivel más alto tenemos que iterar sobre cada paso de integración porque se usa un método implícito para la integración.
- Sobre el nivel más alto puede pasar que tenemos que rechazar el paso a causa de una estimación del error excesivamente alta y repetirlo con un paso de integración más pequeño.

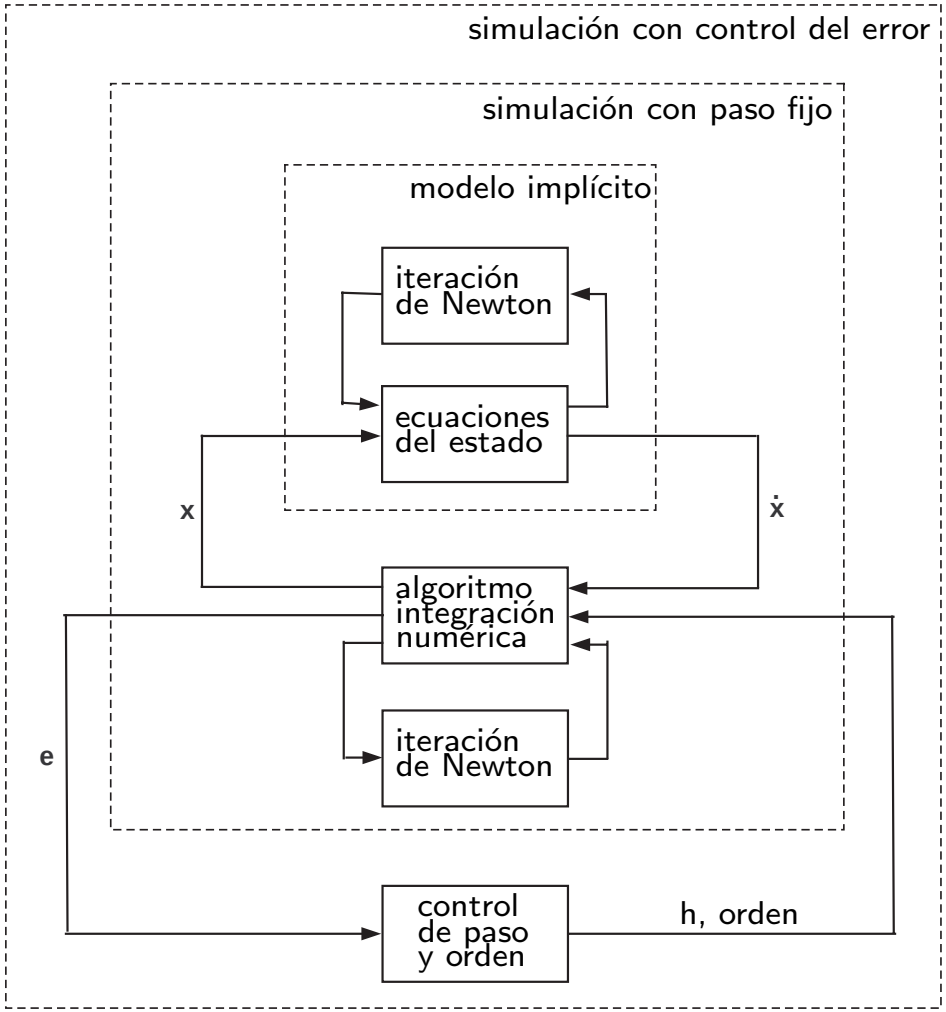
Los Tres Bucles de Iteración

Integrando un modelo implícito de un sistema dinámico usando un algoritmo implícito con control del paso y del orden, encontramos *tres bucles de iteración*.

- Sobre el nivel más bajo tenemos que iterar durante cada evaluación del modelo para obtener los valores de las derivadas. Esta iteración sale muy cara, porque se hace más frecuentemente.
- Sobre un nivel más alto tenemos que iterar sobre cada paso de integración porque se usa un método implícito para la integración.
- Sobre el nivel más alto puede pasar que tenemos que rechazar el paso a causa de una estimación del error excesivamente alta y repetirlo con un paso de integración más pequeño.

Los tres bucles de iteración pueden aclararse en la figura siguiente.

Los Tres Bucles de Iteración II



Integración Implícita de Ecuaciones Implícitos

Queremos simular el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

usando el algoritmo BDF3:

$$\mathbf{x_{k+1}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x_k} - \frac{9}{11}\mathbf{x_{k-1}} + \frac{2}{11}\mathbf{x_{k-2}}$$

Integración Implícita de Ecuaciones Implícitos

Queremos simular el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

usando el algoritmo BDF3:

$$\mathbf{x_{k+1}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x_k} - \frac{9}{11}\mathbf{x_{k-1}} + \frac{2}{11}\mathbf{x_{k-2}}$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right]$$

Integración Implícita de Ecuaciones Implícitos

Queremos simular el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

usando el algoritmo BDF3:

$$\mathbf{x_{k+1}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x_k} - \frac{9}{11}\mathbf{x_{k-1}} + \frac{2}{11}\mathbf{x_{k-2}}$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right]$$

Reemplazamos la fórmula de la derivada en las ecuaciones del modelo:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{f}(\mathbf{x_{k+1}}, \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right], \mathbf{u_{k+1}}, t_{k+1}) = 0.0$$

Integración Implícita de Ecuaciones Implícitos

Queremos simular el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

usando el algoritmo BDF3:

$$\mathbf{x_{k+1}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x_k} - \frac{9}{11}\mathbf{x_{k-1}} + \frac{2}{11}\mathbf{x_{k-2}}$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right]$$

Reemplazamos la fórmula de la derivada en las ecuaciones del modelo:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{f}(\mathbf{x_{k+1}}, \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right], \mathbf{u_{k+1}}, t_{k+1}) = 0.0$$

Se puede aplicar la iteración de Newton directamente.

Algoritmos BDF

Un algoritmo de la integración numérica que se aplica directamente a un modelo implícito se llama *algoritmo de solución de ecuaciones diferenciales algebraicas* o más corto *algoritmo DAE*.

Algoritmos BDF

Un algoritmo de la integración numérica que se aplica directamente a un modelo implícito se llama *algoritmo de solución de ecuaciones diferenciales algebraicas* o más corto *algoritmo DAE*.

Tenemos que analizar las *propiedades de la estabilidad numérica* de los algoritmos DAE.

Algoritmos BDF

Un algoritmo de la integración numérica que se aplica directamente a un modelo implícito se llama *algoritmo de solución de ecuaciones diferenciales algebraicas* o más corto *algoritmo DAE*.

Tenemos que analizar las *propiedades de la estabilidad numérica* de los algoritmos DAE.

Empezamos con el modelo implícito lineal:

$$\mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \dot{\mathbf{x}} = 0.0$$

Insertamos el *algoritmo BDF3*:

$$\mathbf{A} \cdot \mathbf{x_{k+1}} + \frac{11\mathbf{B}}{6h} \cdot \left(\mathbf{x_{k+1}} - \frac{18}{11}\mathbf{x_k} + \frac{9}{11}\mathbf{x_{k-1}} - \frac{2}{11}\mathbf{x_{k-2}} \right) = 0.0$$

Algoritmos BDF

Un algoritmo de la integración numérica que se aplica directamente a un modelo implícito se llama *algoritmo de solución de ecuaciones diferenciales algebraicas* o más corto *algoritmo DAE*.

Tenemos que analizar las *propiedades de la estabilidad numérica* de los algoritmos DAE.

Empezamos con el modelo implícito lineal:

$$\mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \dot{\mathbf{x}} = 0.0$$

Insertamos el *algoritmo BDF3*:

$$\mathbf{A} \cdot \mathbf{x_{k+1}} + \frac{11\mathbf{B}}{6h} \cdot \left(\mathbf{x_{k+1}} - \frac{18}{11}\mathbf{x_k} + \frac{9}{11}\mathbf{x_{k-1}} - \frac{2}{11}\mathbf{x_{k-2}} \right) = 0.0$$

Entonces:

$$\mathbf{x_{k+1}} = \left(-\mathbf{B} - \frac{6\mathbf{A}h}{11} \right)^{-1} \cdot \left(-\frac{18\mathbf{B}}{11}\mathbf{x_k} + \frac{9\mathbf{B}}{11}\mathbf{x_{k-1}} - \frac{2\mathbf{B}}{11}\mathbf{x_{k-2}} \right)$$

con la matriz $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} \mathbf{O^{(n)}} & \mathbf{I^{(n)}} & \mathbf{O^{(n)}} \\ \mathbf{O^{(n)}} & \mathbf{O^{(n)}} & \mathbf{I^{(n)}} \\ \frac{2}{11}\left(\mathbf{B} + \frac{6}{11}\mathbf{A}h\right)^{-1}\mathbf{B} & -\frac{9}{11}\left(\mathbf{B} + \frac{6}{11}\mathbf{A}h\right)^{-1}\mathbf{B} & \frac{18}{11}\left(\mathbf{B} + \frac{6}{11}\mathbf{A}h\right)^{-1}\mathbf{B} \end{pmatrix}$$

Algoritmos BDF II

Con una *matriz* **B** *no singular*, el modelo puede reformularse de forma explícita:

$$\dot{\mathbf{x}} = -\mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{x}$$

Algoritmos BDF II

Con una *matriz* $\mathbf{B}$ *no singular*, el modelo puede reformularse de forma explícita:

$$\dot{\mathbf{x}} = -\mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{x}$$

Para encontrar el dominio de la estabilidad numérica tenemos que colocar los autovalores de $-\mathbf{B}^{-1} \cdot \mathbf{A}$ sobre el círculo unitario.

Algoritmos BDF II

Con una *matriz* $\mathbf{B}$ *no singular*, el modelo puede reformularse de forma explícita:

$$\dot{\mathbf{x}} = -\mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{x}$$

Para encontrar el dominio de la estabilidad numérica tenemos que colocar los autovalores de $-\mathbf{B}^{-1} \cdot \mathbf{A}$ sobre el círculo unitario.

Entonces escogemos cualquiera matriz $\mathbf{B} \in \Re^{2\times 2}$ no singular y calculamos:

$$\mathbf{A} = -\mathbf{B} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix}$$

Algoritmos BDF II

Con una *matriz* $\mathbf{B}$ *no singular*, el modelo puede reformularse de forma explícita:

$$\dot{\mathbf{x}} = -\mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{x}$$

Para encontrar el dominio de la estabilidad numérica tenemos que colocar los autovalores de $-\mathbf{B}^{-1} \cdot \mathbf{A}$ sobre el círculo unitario.

Entonces escogemos cualquiera matriz $\mathbf{B} \in \Re^{2\times 2}$ no singular y calculamos:

$$\mathbf{A} = -\mathbf{B} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix}$$

Ahora iteramos sobre h hasta que los autovalores dominantes de $\mathbf{F}$ se encuentren sobre el círculo unitario y los demás autovalores se encuentren dentro del círculo unitario.

Algoritmos BDF II

Con una *matriz* **B** *no singular*, el modelo puede reformularse de forma explícita:

$$\dot{\mathbf{x}} = -\mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{x}$$

Para encontrar el dominio de la estabilidad numérica tenemos que colocar los autovalores de $-\mathbf{B}^{-1} \cdot \mathbf{A}$ sobre el círculo unitario.

Entonces escogemos cualquiera matriz $\mathbf{B} \in \Re^{2\times 2}$ no singular y calculamos:

$$\mathbf{A} = -\mathbf{B} \cdot \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix}$$

Ahora iteramos sobre *h* hasta que los autovalores dominantes de **F** se encuentren sobre el círculo unitario y los demás autovalores se encuentren dentro del círculo unitario.

El dominio de la estabilidad numérica del problema implícito calculado para el algoritmo DAE del tipo BDF3 es exactamente igual al dominio de la estabilidad numérica del problema explícito calculado para el algoritmo ODE del tipo BDF3.

Modelos de Alto Índice

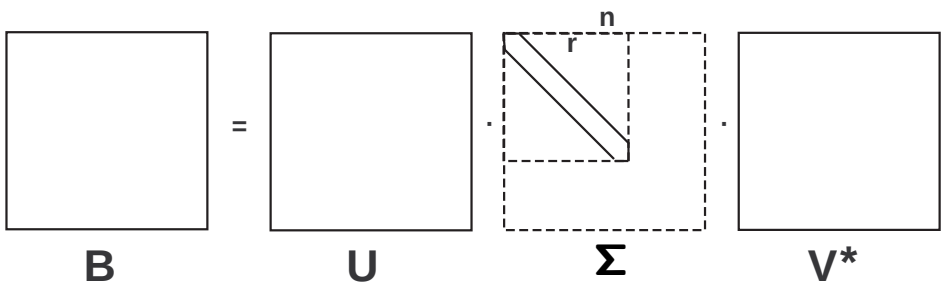
Con una *matriz* **B** *singular*, el modelo no puede reformularse de forma explícita tan fácilmente.

Modelos de Alto Índice

Con una *matriz* **B** *singular*, el modelo no puede reformularse de forma explícita tan fácilmente.

Empezamos con una *descomposición de la matriz* **B** *en valores singulares*:

$$\mathbf{B} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^*$$



con:

$$\text{rango}(\mathbf{U}) = \text{rango}(\mathbf{V}) = n$$
$$\text{rango}(\mathbf{\Sigma}) = \text{rango}(\mathbf{B}) = r < n$$

U y **V** son *matrices unitarias*, mientras que **Σ** es una *matriz diagonal*.

$\mathbf{V}^*$ es la *transpuesta Hermítica* de **V**.

Modelos de Alto Índice II

Entonces:

$$\mathbf{A} \cdot \mathbf{x} + \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^* \cdot \dot{\mathbf{x}} = 0.0$$

$$\Rightarrow \quad \mathbf{U}^* \cdot \mathbf{A} \cdot \mathbf{x} + \mathbf{\Sigma} \cdot \mathbf{V}^* \cdot \dot{\mathbf{x}} = 0.0$$

Modelos de Alto Índice II

Entonces:

$$\mathbf{A} \cdot \mathbf{x} + \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^* \cdot \dot{\mathbf{x}} = 0.0$$
$$\Rightarrow \quad \mathbf{U}^* \cdot \mathbf{A} \cdot \mathbf{x} + \mathbf{\Sigma} \cdot \mathbf{V}^* \cdot \dot{\mathbf{x}} = 0.0$$

y con:

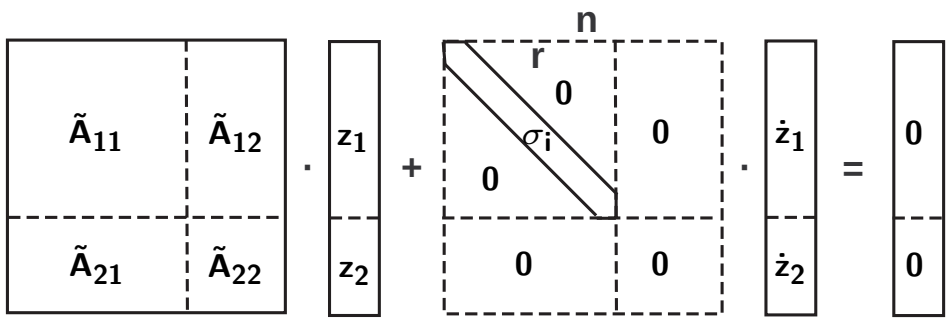
$$\mathbf{z} = \mathbf{V}^* \cdot \mathbf{x}$$

obtenemos:

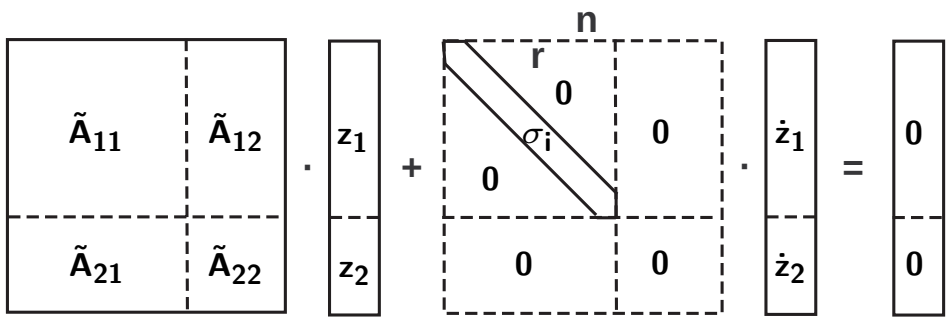
$$\mathbf{U}^* \cdot \mathbf{A} \cdot \mathbf{V} \cdot \mathbf{z} + \mathbf{\Sigma} \cdot \dot{\mathbf{z}} = 0.0$$

o:

$$\tilde{\mathbf{A}} \cdot \mathbf{z} + \mathbf{\Sigma} \cdot \dot{\mathbf{z}} = 0.0$$



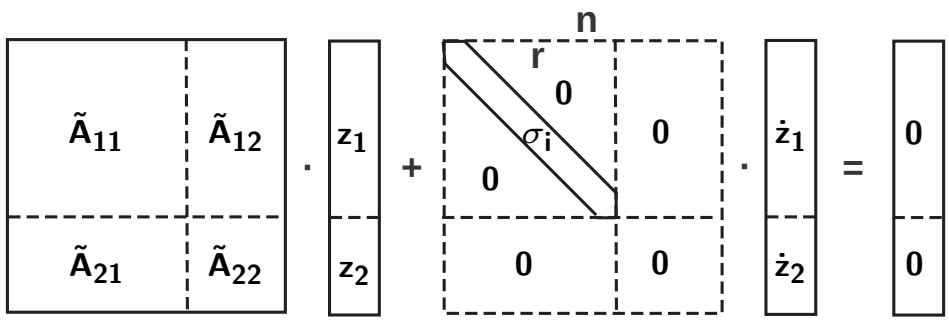
Modelos de Alto Índice III



$$\tilde{\mathbf{A}}_{11} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{12} \cdot \mathbf{z}_2 + \mathbf{\Sigma}_{11} \cdot \dot{\mathbf{z}}_1 = 0.0$$

$$\tilde{\mathbf{A}}_{21} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{22} \cdot \mathbf{z}_2 = 0.0$$

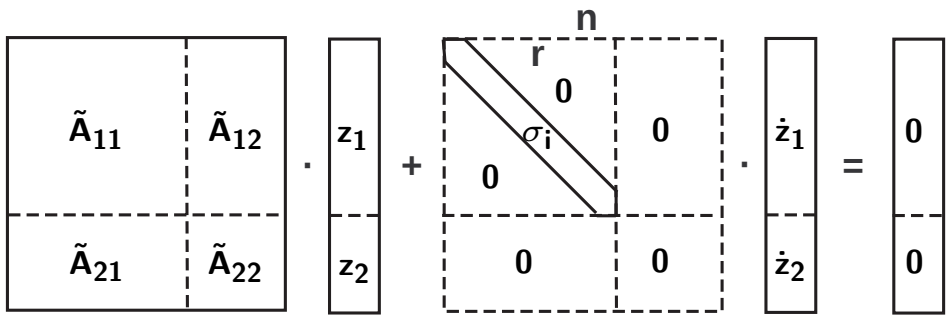
Modelos de Alto Índice III



$$\tilde{\mathbf{A}}_{11} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{12} \cdot \mathbf{z}_2 + \mathbf{\Sigma}_{11} \cdot \dot{\mathbf{z}}_1 = 0.0$$
$$\tilde{\mathbf{A}}_{21} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{22} \cdot \mathbf{z}_2 = 0.0$$

Si la matriz $\tilde{\mathbf{A}}_{22}$ es *singular*, el *modelo es mal definido*.

Modelos de Alto Índice III



$$\tilde{\mathbf{A}}_{11} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{12} \cdot \mathbf{z}_2 + \mathbf{\Sigma}_{11} \cdot \dot{\mathbf{z}}_1 = 0.0$$
$$\tilde{\mathbf{A}}_{21} \cdot \mathbf{z}_1 + \tilde{\mathbf{A}}_{22} \cdot \mathbf{z}_2 = 0.0$$

Si la matriz $\tilde{\mathbf{A}}_{22}$ es *singular*, el *modelo es mal definido*.

Por otra parte, si la matriz $\tilde{\mathbf{A}}_{22}$ es *regular*, podemos escribir:

$$\mathbf{z}_2 = -\tilde{\mathbf{A}}_{22}^{-1} \cdot \tilde{\mathbf{A}}_{21} \cdot \mathbf{z}_1$$

y entonces:

$$\dot{\mathbf{z}}_1 = \mathbf{\Sigma}_{11}{}^{-1} \cdot \left(\tilde{\mathbf{A}}_{12} \cdot \tilde{\mathbf{A}}_{22}^{-1} \cdot \tilde{\mathbf{A}}_{21} - \tilde{\mathbf{A}}_{11} \right) \cdot \mathbf{z}_1$$

Modelos de Alto Índice IV

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.
- Dicho modelo se llama *modelo de alto índice (de perturbación)*.

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.
- Dicho modelo se llama *modelo de alto índice (de perturbación)*.
- Existen *algoritmos simbólicos* para la *reducción automática del índice*. Uno de estos algoritmos es el *algoritmo de Pantelides*.

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.
- Dicho modelo se llama *modelo de alto índice (de perturbación)*.
- Existen *algoritmos simbólicos* para la *reducción automática del índice*. Uno de estos algoritmos es el *algoritmo de Pantelides*.
- El algoritmo de Pantelides es un algoritmo de *complejidad lineal de computación*.

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.
- Dicho modelo se llama *modelo de alto índice (de perturbación)*.
- Existen *algoritmos simbólicos* para la *reducción automática del índice*. Uno de estos algoritmos es el *algoritmo de Pantelides*.
- El algoritmo de Pantelides es un algoritmo de *complejidad lineal de computación*.
- Nunca vale la pena resolver modelos de alto índice directamente. Siempre es mejor reducir el índice simbólicamente durante la compilación del modelo.

Modelos de Alto Índice IV

- Aunque el sistema parece ser del orden n, en realidad se trata de un sistema del orden $r < n$.
- Dicho modelo se llama *modelo de alto índice (de perturbación)*.
- Existen *algoritmos simbólicos* para la *reducción automática del índice*. Uno de estos algoritmos es el *algoritmo de Pantelides*.
- El algoritmo de Pantelides es un algoritmo de *complejidad lineal de computación*.
- Nunca vale la pena resolver modelos de alto índice directamente. Siempre es mejor reducir el índice simbólicamente durante la compilación del modelo.
- **Entonces no tenemos que tratar con matrices B singulares**.

Algoritmos AM

Analizamos ahora que pasa si aplicamos un *algoritmo DAE del tipo AM*.

Algoritmos AM

Analizamos ahora que pasa si aplicamos un *algoritmo DAE del tipo AM*.

Empezamos con al algoritmo AM3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f_k} - \mathbf{f_{k-1}}\right)$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \mathbf{f_{k+1}} = \frac{12}{5h}\left(\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\right) - \frac{8}{5}\mathbf{f_k} + \frac{1}{5}\mathbf{f_{k-1}}$$

Algoritmos AM

Analizamos ahora que pasa si aplicamos un *algoritmo DAE del tipo AM*.

Empezamos con al algoritmo AM3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f_k} - \mathbf{f_{k-1}}\right)$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \mathbf{f_{k+1}} = \frac{12}{5h}\left(\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\right) - \frac{8}{5}\mathbf{f_k} + \frac{1}{5}\mathbf{f_{k-1}}$$

Podemos introducir esa ecuación en el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

eliminando la derivada $\dot{\mathbf{x}}_{\mathbf{k+1}}$.

Algoritmos AM

Analizamos ahora que pasa si aplicamos un *algoritmo DAE del tipo AM*.

Empezamos con al algoritmo AM3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f_k} - \mathbf{f_{k-1}}\right)$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \mathbf{f_{k+1}} = \frac{12}{5h}\left(\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\right) - \frac{8}{5}\mathbf{f_k} + \frac{1}{5}\mathbf{f_{k-1}}$$

Podemos introducir esa ecuación en el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

eliminando la derivada $\dot{\mathbf{x}}_{\mathbf{k+1}}$.

Desafortunadamente nos quedan las "conocidas" $\mathbf{f_k}$ y $\mathbf{f_{k-1}}$ en el modelo, para la evaluación de las cuales ya no tenemos más ecuaciones porque *eliminamos las derivadas del modelo*.

Algoritmos AM

Analizamos ahora que pasa si aplicamos un *algoritmo DAE del tipo AM*.

Empezamos con al algoritmo AM3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f_k} - \mathbf{f_{k-1}}\right)$$

Podemos resolver la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \mathbf{f_{k+1}} = \frac{12}{5h}\left(\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\right) - \frac{8}{5}\mathbf{f_k} + \frac{1}{5}\mathbf{f_{k-1}}$$

Podemos introducir esa ecuación en el modelo:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}, t) = 0.0$$

eliminando la derivada $\dot{\mathbf{x}}_{\mathbf{k+1}}$.

Desafortunadamente nos quedan las "conocidas" $\mathbf{f_k}$ y $\mathbf{f_{k-1}}$ en el modelo, para la evaluación de las cuales ya no tenemos más ecuaciones porque *eliminamos las derivadas del modelo*.

Eso no pasó en el caso de los algoritmos BDF, porque estos algoritmos no usan derivadas del pasado.

Algoritmos AM II

Algoritmos AM II

- Los algoritmos de integración numérica que solamente usan una sola derivada se llaman *algoritmos con una pierna*.

Algoritmos AM II

- Los algoritmos de integración numérica que solamente usan una sola derivada se llaman *algoritmos con una pierna*.
- Los algoritmos BDF son algoritmos con una pierna. Los algoritmos AM no lo son.

Algoritmos AM II

- Los algoritmos de integración numérica que solamente usan una sola derivada se llaman *algoritmos con una pierna*.
- Los algoritmos BDF son algoritmos con una pierna. Los algoritmos AM no lo son.

Una manera para resolver el problema es *introducir una segunda iteración de Newton*:

$$\mathcal{F}_1\left(\mathbf{x}(t_{k+1})\right) = \mathbf{f}\left(\mathbf{x}(t_{k+1}), \frac{12}{5h}\mathbf{x}(t_{k+1}) - \frac{12}{5h}\mathbf{x}(t_k) - \frac{8}{5}\mathbf{w}(t_k) + \frac{1}{5}\mathbf{w}(t_{k-1}), \mathbf{u}(t_{k+1}), t_{k+1}\right) = 0.0$$

$$\mathcal{F}_2\left(\mathbf{w}(t_{k+1})\right) = \mathbf{f}\left(\mathbf{x}(t_{k+1}), \mathbf{w}(t_{k+1}), \mathbf{u}(t_{k+1}), t_{k+1}\right) = 0.0$$

Algoritmos AM III

Buscamos ahora el *dominio de la estabilidad numérica del algoritmo AM3* usado como algoritmo DAE.

Insertando la fórmula AM3 en el modelo lineal, obtenemos:

$$\mathbf{x}(t_{k+1}) = \left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1} \cdot \left(\mathbf{B}\mathbf{x}(t_k) + \frac{2}{3}\mathbf{B}h\mathbf{w}(t_k) - \frac{1}{12}\mathbf{B}h\mathbf{w}(t_{k-1})\right)$$

$$\mathbf{w}(t_{k+1}) = -\mathbf{B}^{-1}\mathbf{A}\mathbf{x}(t_{k+1})$$

Algoritmos AM III

Buscamos ahora el *dominio de la estabilidad numérica del algoritmo AM3* usado como algoritmo DAE.

Insertando la fórmula AM3 en el modelo lineal, obtenemos:

$$\mathbf{x}(t_{k+1}) = \left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1} \cdot \left(\mathbf{B}\mathbf{x}(t_k) + \frac{2}{3}\mathbf{B}h\mathbf{w}(t_k) - \frac{1}{12}\mathbf{B}h\mathbf{w}(t_{k-1})\right)$$
$$\mathbf{w}(t_{k+1}) = -\mathbf{B}^{-1}\mathbf{A}\mathbf{x}(t_{k+1})$$

Con el vector del estado:

$$\mathbf{z_k} = \begin{pmatrix} \mathbf{x_k} \\ \mathbf{w_{k-1}} \\ \mathbf{w_k} \end{pmatrix}$$

resulta la matriz $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} \left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B} & -\frac{1}{12}\left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B}h & \frac{2}{3}\left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B}h \\ \mathbf{O}^{(\mathbf{n})} & \mathbf{O}^{(\mathbf{n})} & \mathbf{I}^{(\mathbf{n})} \\ -\mathbf{B}^{-1}\mathbf{A}\left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B} & \frac{1}{12}\mathbf{B}^{-1}\mathbf{A}\left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B}h & -\frac{2}{3}\mathbf{B}^{-1}\mathbf{A}\left(\mathbf{B} + \frac{5}{12}\mathbf{A}h\right)^{-1}\mathbf{B}h \end{pmatrix}$$

Algoritmos AM IV

Algoritmos AM IV

- Para cualquiera matriz **B** no singular resulta el mismo dominio de la estabilidad numérica como en el caso del mismo algoritmo AM3 usado como algoritmo ODE.

Algoritmos AM IV

- Para cualquiera matriz **B** no singular resulta el mismo dominio de la estabilidad numérica como en el caso del mismo algoritmo AM3 usado como algoritmo ODE.

- Este resultado parece extraño, porque el tamaño de la matriz **F** es más grande en el caso DAE que en el caso ODE. Es decir, la matriz **F** tiene más autovalores.

Algoritmos AM IV

- Para cualquiera matriz **B** no singular resulta el mismo dominio de la estabilidad numérica como en el caso del mismo algoritmo AM3 usado como algoritmo ODE.
- Este resultado parece extraño, porque el tamaño de la matriz **F** es más grande en el caso DAE que en el caso ODE. Es decir, la matriz **F** tiene más autovalores.
- Sin embargo, $\mathbf{w}(t_{k+1})$ es lineal en $\mathbf{x}(t_{k+1})$, entonces **F** es singular. Simplemente añadimos unos autovalores adicionales en el origen. Estos autovalores no pueden influenciar el dominio de estabilidad.

Algoritmos AM IV

- Para cualquiera matriz **B** no singular resulta el mismo dominio de la estabilidad numérica como en el caso del mismo algoritmo AM3 usado como algoritmo ODE.
- Este resultado parece extraño, porque el tamaño de la matriz **F** es más grande en el caso DAE que en el caso ODE. Es decir, la matriz **F** tiene más autovalores.
- Sin embargo, $\mathbf{w}(t_{k+1})$ es lineal en $\mathbf{x}(t_{k+1})$, entonces **F** es singular. Simplemente añadimos unos autovalores adicionales en el origen. Estos autovalores no pueden influenciar el dominio de estabilidad.

Es evidente que el dominio de estabilidad numérica de un algoritmo implícito de integración numérica no cambia si lo usamos en la forma DAE en lugar de usarlo en la forma ODE.

Algoritmos AB

Veamos ahora que pasa si tratamos de modificar un *algoritmo explícito* a la forma DAE.

Algoritmos AB

Veamos ahora que pasa si tratamos de modificar un *algoritmo explícito* a la forma DAE.

Empezamos con el algoritmo AB3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right)$$

Resolvemos la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{23h}\left(\mathbf{x}(t_{k+2}) - \mathbf{x}(t_{k+1})\right) + \frac{16}{23}\mathbf{f}_k - \frac{5}{23}\mathbf{f}_{k-1}$$

Algoritmos AB

Veamos ahora que pasa si tratamos de modificar un *algoritmo explícito* a la forma DAE.

Empezamos con el algoritmo AB3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right)$$

Resolvemos la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{23h}\left(\mathbf{x}(t_{k+2}) - \mathbf{x}(t_{k+1})\right) + \frac{16}{23}\mathbf{f}_k - \frac{5}{23}\mathbf{f}_{k-1}$$

Desafortunadamente, las *fórmulas explícitas de integración* se convierten en *fórmulas sobre implícitas de diferenciación*.

Algoritmos AB

Veamos ahora que pasa si tratamos de modificar un *algoritmo explícito* a la forma DAE.

Empezamos con el algoritmo AB3:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right)$$

Resolvemos la ecuación vectorial del algoritmo por la derivada:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{23h}\left(\mathbf{x}(t_{k+2}) - \mathbf{x}(t_{k+1})\right) + \frac{16}{23}\mathbf{f}_k - \frac{5}{23}\mathbf{f}_{k-1}$$

Desafortunadamente, las *fórmulas explícitas de integración* se convierten en *fórmulas sobre implícitas de diferenciación*.

Estas fórmulas no sirven para nada, porque resultaría una iteración gigantesca sobre toda la simulación en lugar de solamente sobre un paso.

Algoritmos de Integración Numérica Sobre Implícitos

Se puede pensar que los *algoritmos de integración numérica sobre implícitos* que hasta ahora nunca consideramos pueden ser útiles como algoritmos DAE.

Algoritmos de Integración Numérica Sobre Implícitos

Se puede pensar que los *algoritmos de integración numérica sobre implícitos* que hasta ahora nunca consideramos pueden ser útiles como algoritmos DAE.

La *fórmula de Adams de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(-\mathbf{f}(t_{k+2}) + 8\mathbf{f}(t_{k+1}) + 5\mathbf{f}(t_k)\right)$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{h}\left(\mathbf{x}(t_{k-1}) - \mathbf{x}(t_k)\right) + 8\mathbf{f}(t_k) + 5\mathbf{f}(t_{k-1})$$

Algoritmos de Integración Numérica Sobre Implícitos

Se puede pensar que los *algoritmos de integración numérica sobre implícitos* que hasta ahora nunca consideramos pueden ser útiles como algoritmos DAE.

La *fórmula de Adams de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(-\mathbf{f}(t_{k+2}) + 8\mathbf{f}(t_{k+1}) + 5\mathbf{f}(t_k)\right)$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{h}\left(\mathbf{x}(t_{k-1}) - \mathbf{x}(t_k)\right) + 8\mathbf{f}(t_k) + 5\mathbf{f}(t_{k-1})$$

- Las fórmulas sobre implícitas de la integración numérica se convierten en fórmulas explícitas de la diferenciación numérica.

Algoritmos de Integración Numérica Sobre Implícitos

Se puede pensar que los *algoritmos de integración numérica sobre implícitos* que hasta ahora nunca consideramos pueden ser útiles como algoritmos DAE.

La *fórmula de Adams de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(-\mathbf{f}(t_{k+2}) + 8\mathbf{f}(t_{k+1}) + 5\mathbf{f}(t_k)\right)$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{12}{h}\left(\mathbf{x}(t_{k-1}) - \mathbf{x}(t_k)\right) + 8\mathbf{f}(t_k) + 5\mathbf{f}(t_{k-1})$$

- Las fórmulas sobre implícitas de la integración numérica se convierten en fórmulas explícitas de la diferenciación numérica.
- Desafortunadamente, el algoritmo de integración es inestable. En consecuencia, también el algoritmo de diferenciación es inestable.

Algoritmos de Integración Numérica Sobre Implícitos II

Consideramos ahora los algoritmos del tipo BDF.

Algoritmos de Integración Numérica Sobre Implícitos II

Consideramos ahora los algoritmos del tipo BDF.

La *fórmula BDF de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \frac{57}{26}\mathbf{x}(t_k) - \frac{21}{13}\mathbf{x}(t_{k-1}) + \frac{11}{26}\mathbf{x}(t_{k-2}) + \frac{6h}{26}\mathbf{f}(t_{k+2})$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{1}{6h}\left(26\mathbf{x}(t_k) - 57\mathbf{x}(t_{k-1}) + 42\mathbf{x}(t_{k-2}) - 11\mathbf{x}(t_{k-3})\right)$$

Algoritmos de Integración Numérica Sobre Implícitos II

Consideramos ahora los algoritmos del tipo BDF.

La *fórmula BDF de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \frac{57}{26}\mathbf{x}(t_k) - \frac{21}{13}\mathbf{x}(t_{k-1}) + \frac{11}{26}\mathbf{x}(t_{k-2}) + \frac{6h}{26}\mathbf{f}(t_{k+2})$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{1}{6h}\left(26\mathbf{x}(t_k) - 57\mathbf{x}(t_{k-1}) + 42\mathbf{x}(t_{k-2}) - 11\mathbf{x}(t_{k-3})\right)$$

- Desafortunadamente, también este algoritmo de integración es inestable. En consecuencia, el algoritmo de diferenciación es igualmente inestable.

Algoritmos de Integración Numérica Sobre Implícitos II

Consideramos ahora los algoritmos del tipo BDF.

La *fórmula BDF de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \frac{57}{26}\mathbf{x}(t_k) - \frac{21}{13}\mathbf{x}(t_{k-1}) + \frac{11}{26}\mathbf{x}(t_{k-2}) + \frac{6h}{26}\mathbf{f}(t_{k+2})$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{1}{6h}\left(26\mathbf{x}(t_k) - 57\mathbf{x}(t_{k-1}) + 42\mathbf{x}(t_{k-2}) - 11\mathbf{x}(t_{k-3})\right)$$

- Desafortunadamente, también este algoritmo de integración es inestable. En consecuencia, el algoritmo de diferenciación es igualmente inestable.

En general, las fórmulas explícitas de diferenciación no deben usarse a causa de problemas con su estabilidad numérica.

Algoritmos de Integración Numérica Sobre Implícitos II

Consideramos ahora los algoritmos del tipo BDF.

La *fórmula BDF de tercer orden sobre implícita* puede escribirse de la forma:

$$\mathbf{x}(t_{k+1}) = \frac{57}{26}\mathbf{x}(t_k) - \frac{21}{13}\mathbf{x}(t_{k-1}) + \frac{11}{26}\mathbf{x}(t_{k-2}) + \frac{6h}{26}\mathbf{f}(t_{k+2})$$

Resolviéndola por la derivada obtenemos:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{1}{6h}\left(26\mathbf{x}(t_k) - 57\mathbf{x}(t_{k-1}) + 42\mathbf{x}(t_{k-2}) - 11\mathbf{x}(t_{k-3})\right)$$

- Desafortunadamente, también este algoritmo de integración es inestable. En consecuencia, el algoritmo de diferenciación es igualmente inestable.

En general, las fórmulas explícitas de diferenciación no deben usarse a causa de problemas con su estabilidad numérica.

Por otro lado, las fórmulas implícitas de diferenciación son estables al igual que las fórmulas implícitas de integración.

Propiedades de la Precisión

También tenemos que investigar las *propiedades de la precisión* de los *algoritmos DAE*.

Propiedades de la Precisión

También tenemos que investigar las *propiedades de la precisión* de los *algoritmos DAE*.

Para ello simulé un *sistema lineal marginalmente estable (no rígido) de alto orden* usando los simuladores DAE de los tipos BDF3 y AM3. Los resultados de la simulación pueden resumirse de la forma siguiente:

h	BDF3	AM3
0.1	basura	inestable
0.05	basura	inestable
0.02	basura	inestable
0.01	basura	inestable
0.005	0.9469e-2	0.8783e-8
0.002	0.1742e-6	0.2149e-8
0.001	0.4363e-7	0.2120e-8

Propiedades de la Precisión

También tenemos que investigar las *propiedades de la precisión* de los *algoritmos DAE*.

Para ello simulé un *sistema lineal marginalmente estable (no rígido) de alto orden* usando los simuladores DAE de los tipos BDF3 y AM3. Los resultados de la simulación pueden resumirse de la forma siguiente:

h	BDF3	AM3
0.1	basura	inestable
0.05	basura	inestable
0.02	basura	inestable
0.01	basura	inestable
0.005	0.9469e-2	0.8783e-8
0.002	0.1742e-6	0.2149e-8
0.001	0.4363e-7	0.2120e-8

- El algoritmo BDF3 no produce resultados inestables. Sin embargo, los errores resultan más grandes que los resultados mismos si se usa un paso h excesivamente grande.

Propiedades de la Precisión

También tenemos que investigar las *propiedades de la precisión* de los *algoritmos DAE*.

Para ello simulé un *sistema lineal marginalmente estable (no rígido) de alto orden* usando los simuladores DAE de los tipos BDF3 y AM3. Los resultados de la simulación pueden resumirse de la forma siguiente:

h	BDF3	AM3
0.1	basura	inestable
0.05	basura	inestable
0.02	basura	inestable
0.01	basura	inestable
0.005	0.9469e-2	0.8783e-8
0.002	0.1742e-6	0.2149e-8
0.001	0.4363e-7	0.2120e-8

- El algoritmo BDF3 no produce resultados inestables. Sin embargo, los errores resultan más grandes que los resultados mismos si se usa un paso h excesivamente grande.
- Usando el mismo tamaño del paso h, los resultados obtenidos con AM3 son más precisos que los resultados obtenidos con BDF3. La razón es el *coeficiente del error*, c_{err}, que es bastante más grande en el caso del algoritmo BDF3.

Propiedades de la Precisión II

Simulé el mismo sistema usando varios diferentes *algoritmos ODE*:

h	RK3	IEX3	BI3	AB3	ABM3	AM3	BDF3
0.1	inestable	0.6782e-4	0.4947e-6	inestable	inestable	inestable	basura
0.05	inestable	0.8668e-5	0.2895e-7	inestable	inestable	inestable	basura
0.02	inestable	0.5611e-6	0.1324e-8	inestable	inestable	inestable	basura
0.01	0.7034e-7	0.7029e-7	0.2070e-8	inestable	0.6996e-7	inestable	basura
0.005	0.8954e-8	0.8791e-8	0.2116e-8	0.7906e-7	0.8772e-8	0.8783e-8	0.9469e-2
0.002	0.2219e-8	0.2145e-8	0.2120e-8	0.5427e-8	0.2156e-8	0.2149e-8	0.1742e-6
0.001	0.2127e-8	0.2119e-8	0.2120e-8	0.2239e-8	0.2120e-8	0.2120e-8	0.4363e-7

Propiedades de la Precisión II

Simulé el mismo sistema usando varios diferentes *algoritmos ODE*:

h	RK3	IEX3	BI3	AB3	ABM3	AM3	BDF3
0.1	inestable	0.6782e-4	0.4947e-6	inestable	inestable	inestable	basura
0.05	inestable	0.8668e-5	0.2895e-7	inestable	inestable	inestable	basura
0.02	inestable	0.5611e-6	0.1324e-8	inestable	inestable	inestable	basura
0.01	0.7034e-7	0.7029e-7	0.2070e-8	inestable	0.6996e-7	inestable	basura
0.005	0.8954e-8	0.8791e-8	0.2116e-8	0.7906e-7	0.8772e-8	0.8783e-8	0.9469e-2
0.002	0.2219e-8	0.2145e-8	0.2120e-8	0.5427e-8	0.2156e-8	0.2149e-8	0.1742e-6
0.001	0.2127e-8	0.2119e-8	0.2120e-8	0.2239e-8	0.2120e-8	0.2120e-8	0.4363e-7

- Comparando las últimas dos columnas de esta tabla con la tabla anterior, se nota que *los resultados son exactamente los mismos*.

Propiedades de la Precisión II

Simulé el mismo sistema usando varios diferentes *algoritmos ODE*:

h	RK3	IEX3	BI3	AB3	ABM3	AM3	BDF3
0.1	inestable	0.6782e-4	0.4947e-6	inestable	inestable	inestable	basura
0.05	inestable	0.8668e-5	0.2895e-7	inestable	inestable	inestable	basura
0.02	inestable	0.5611e-6	0.1324e-8	inestable	inestable	inestable	basura
0.01	0.7034e-7	0.7029e-7	0.2070e-8	inestable	0.6996e-7	inestable	basura
0.005	0.8954e-8	0.8791e-8	0.2116e-8	0.7906e-7	0.8772e-8	0.8783e-8	0.9469e-2
0.002	0.2219e-8	0.2145e-8	0.2120e-8	0.5427e-8	0.2156e-8	0.2149e-8	0.1742e-6
0.001	0.2127e-8	0.2119e-8	0.2120e-8	0.2239e-8	0.2120e-8	0.2120e-8	0.4363e-7

- Comparando las últimas dos columnas de esta tabla con la tabla anterior, se nota que *los resultados son exactamente los mismos*.
- **La precisión de un algoritmo implícito de integración numérica no depende de la forma de la iteración. La precisión del algoritmo DAE es exactamente la misma que la del algoritmo ODE equivalente.**

Propiedades de la Precisión II

Simulé el mismo sistema usando varios diferentes *algoritmos ODE*:

h	RK3	IEX3	BI3	AB3	ABM3	AM3	BDF3
0.1	inestable	0.6782e-4	0.4947e-6	inestable	inestable	inestable	basura
0.05	inestable	0.8668e-5	0.2895e-7	inestable	inestable	inestable	basura
0.02	inestable	0.5611e-6	0.1324e-8	inestable	inestable	inestable	basura
0.01	0.7034e-7	0.7029e-7	0.2070e-8	inestable	0.6996e-7	inestable	basura
0.005	0.8954e-8	0.8791e-8	0.2116e-8	0.7906e-7	0.8772e-8	0.8783e-8	0.9469e-2
0.002	0.2219e-8	0.2145e-8	0.2120e-8	0.5427e-8	0.2156e-8	0.2149e-8	0.1742e-6
0.001	0.2127e-8	0.2119e-8	0.2120e-8	0.2239e-8	0.2120e-8	0.2120e-8	0.4363e-7

- Comparando las últimas dos columnas de esta tabla con la tabla anterior, se nota que *los resultados son exactamente los mismos*.
- **La precisión de un algoritmo implícito de integración numérica no depende de la forma de la iteración. La precisión del algoritmo DAE es exactamente la misma que la del algoritmo ODE equivalente.**
- Eso no quiere decir que *los costos de la simulación* sean iguales. En el ejemplo simulado los costos del algoritmo DAE del tipo AM3 son más grandes que los del algoritmo ODE equivalente, porque en el primer caso se necesitan dos iteraciones de Newton para cada paso, mientras que en el segundo se usa una sola.

Propiedades de la Precisión II

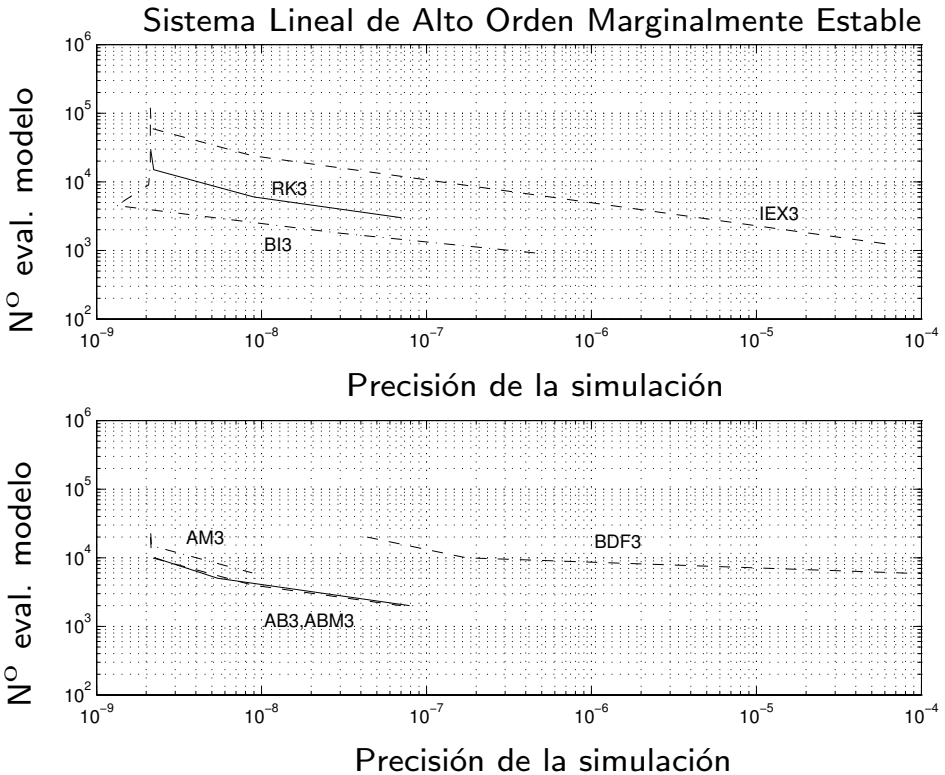
Simulé el mismo sistema usando varios diferentes *algoritmos ODE*:

h	RK3	IEX3	BI3	AB3	ABM3	AM3	BDF3
0.1	inestable	0.6782e-4	0.4947e-6	inestable	inestable	inestable	basura
0.05	inestable	0.8668e-5	0.2895e-7	inestable	inestable	inestable	basura
0.02	inestable	0.5611e-6	0.1324e-8	inestable	inestable	inestable	basura
0.01	0.7034e-7	0.7029e-7	0.2070e-8	inestable	0.6996e-7	inestable	basura
0.005	0.8954e-8	0.8791e-8	0.2116e-8	0.7906e-7	0.8772e-8	0.8783e-8	0.9469e-2
0.002	0.2219e-8	0.2145e-8	0.2120e-8	0.5427e-8	0.2156e-8	0.2149e-8	0.1742e-6
0.001	0.2127e-8	0.2119e-8	0.2120e-8	0.2239e-8	0.2120e-8	0.2120e-8	0.4363e-7

- Comparando las últimas dos columnas de esta tabla con la tabla anterior, se nota que *los resultados son exactamente los mismos*.
- **La precisión de un algoritmo implícito de integración numérica no depende de la forma de la iteración. La precisión del algoritmo DAE es exactamente la misma que la del algoritmo ODE equivalente.**
- Eso no quiere decir que *los costos de la simulación* sean iguales. En el ejemplo simulado los costos del algoritmo DAE del tipo AM3 son más grandes que los del algoritmo ODE equivalente, porque en el primer caso se necesitan dos iteraciones de Newton para cada paso, mientras que en el segundo se usa una sola.
- Por otro lado, si el modelo es implícito (lo que no era el caso en el ejemplo simulado), el algoritmo DAE del tipo BDF3 es más económico que el algoritmo ODE equivalente.

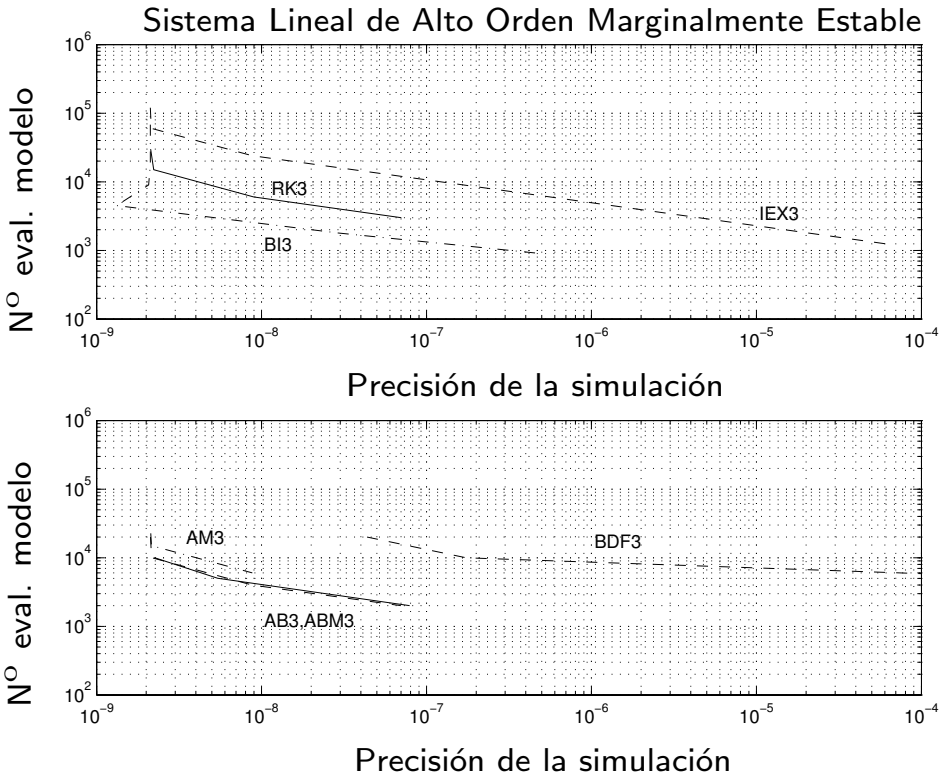
Propiedades de la Precisión III

Comparamos los *algoritmos ODE* de *forma gráfica*:



Propiedades de la Precisión III

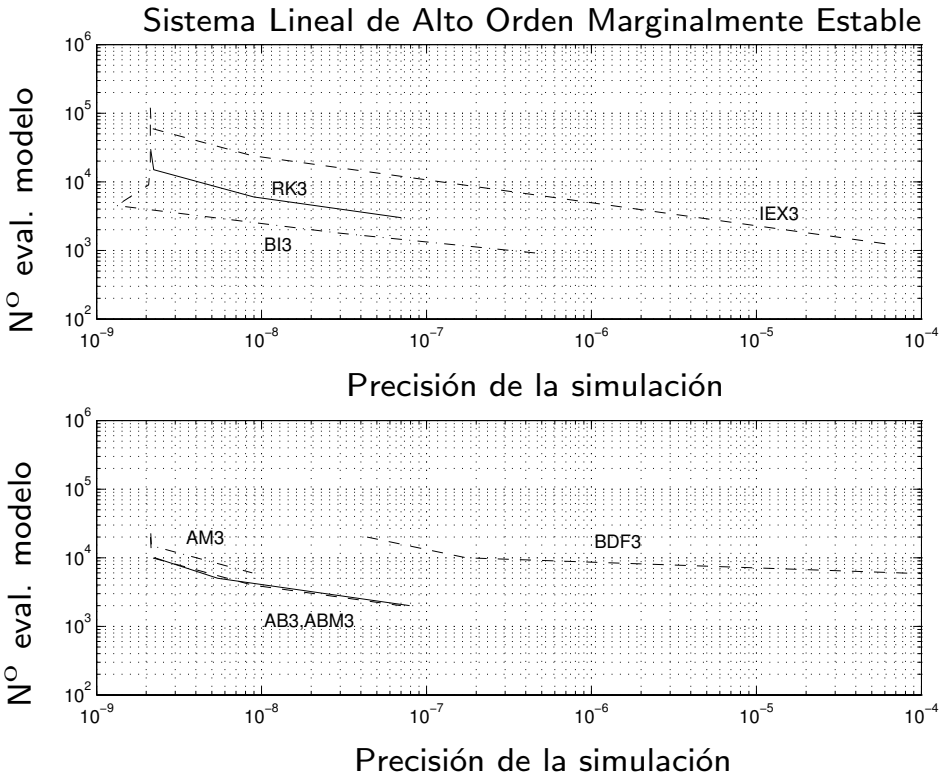
Comparamos los *algoritmos ODE* de *forma gráfica*:



- Para este ejemplo, el algoritmo BI3 es el más económico (no muy sorprendente, como se trata de un sistema lineal marginalmente estable).

Propiedades de la Precisión III

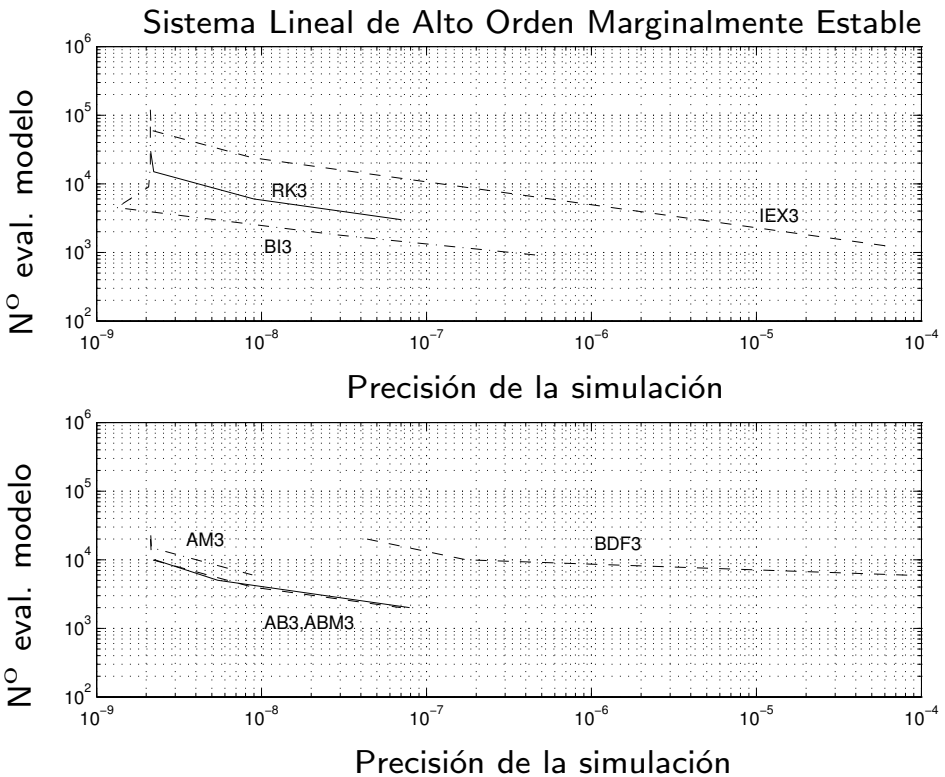
Comparamos los *algoritmos ODE* de *forma gráfica*:



- Para este ejemplo, el algoritmo BI3 es el más económico (no muy sorprendente, como se trata de un sistema lineal marginalmente estable).
- Los algoritmos explícitos (RK3, AB3 y ABM3) son un poco más caros. No hay mucha diferencia entre ellos.

Propiedades de la Precisión III

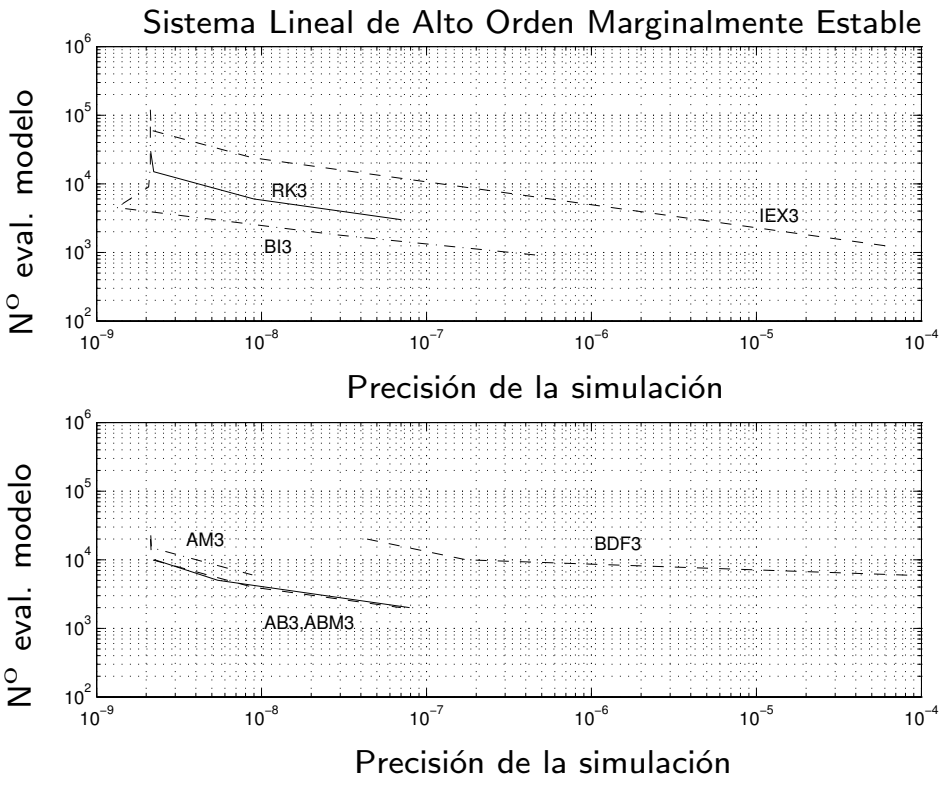
Comparamos los *algoritmos ODE* de *forma gráfica*:



- Para este ejemplo, el algoritmo BI3 es el más económico (no muy sorprendente, como se trata de un sistema lineal marginalmente estable).
- Los algoritmos explícitos (RK3, AB3 y ABM3) son un poco más caros. No hay mucha diferencia entre ellos.
- El algoritmo AM3 es aún un poco más caro.

Propiedades de la Precisión III

Comparamos los *algoritmos ODE* de *forma gráfica*:



- Para este ejemplo, el algoritmo BI3 es el más económico (no muy sorprendente, como se trata de un sistema lineal marginalmente estable).
- Los algoritmos explícitos (RK3, AB3 y ABM3) son un poco más caros. No hay mucha diferencia entre ellos.
- El algoritmo AM3 es aún un poco más caro.
- Los algoritmos rígidos (BDF3 y IEX3) no sirven para la simulación de sistemas marginalmente estables.

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Por ejemplo podemos escribir el algoritmo RK4 de la forma siguiente:

$$\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, \mathbf{u}(t_k), t_k\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)$$

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Por ejemplo podemos escribir el algoritmo RK4 de la forma siguiente:

$$\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, \mathbf{u}(t_k), t_k\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)$$

- Cada etapa del algoritmo RK4 se convierte en una iteración de Newton.

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Por ejemplo podemos escribir el algoritmo RK4 de la forma siguiente:

$$\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, \mathbf{u}(t_k), t_k\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)$$

- Cada etapa del algoritmo RK4 se convierte en una iteración de Newton.
- En el caso de un modelo implícito, los algoritmos ODE y DAE son casi idénticos.

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Por ejemplo podemos escribir el algoritmo RK4 de la forma siguiente:

$$\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, \mathbf{u}(t_k), t_k\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)$$

- Cada etapa del algoritmo RK4 se convierte en una iteración de Newton.
- En el caso de un modelo implícito, los algoritmos ODE y DAE son casi idénticos.
- En el caso de un modelo explícito, el algoritmo DAE es demasiado caro.

Algoritmos RK

Vemos ahora como pueden encontrarse *algoritmos DAE* implementando *algoritmos RK*.

Por ejemplo podemos escribir el algoritmo RK4 de la forma siguiente:

$$\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, \mathbf{u}(t_k), t_k\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, \mathbf{u}(t_k + \frac{h}{2}), t_k + \frac{h}{2}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)$$

- Cada etapa del algoritmo RK4 se convierte en una iteración de Newton.
- En el caso de un modelo implícito, los algoritmos ODE y DAE son casi idénticos.
- En el caso de un modelo explícito, el algoritmo DAE es demasiado caro.
- **En general vale la pena convertir un modelo implícito no rígido simbólicamente a una forma explícita durante la compilación del modelo para evitar las iteraciones. Si eso no puede hacerse, al menos tendremos que minimizar los tamaños de los bucles algebraicos.**

Algoritmos Radau IIA

Existen *algoritmos rígidos del tipo IRK* que no consideramos hasta ahora.

Una clase de tales algoritmos es la de los *algoritmos Radau IIA*.

Algoritmos Radau IIA

Existen *algoritmos rígidos del tipo IRK* que no consideramos hasta ahora.

Una clase de tales algoritmos es la de los *algoritmos Radau IIA*.

El algoritmo Radau IIA de orden tres es caracterizado por la tabla de Butcher:

1/3	5/12	-1/12
1	3/4	1/4
x	3/4	1/4

Algoritmos Radau IIA

Existen *algoritmos rígidos del tipo IRK* que no consideramos hasta ahora.

Una clase de tales algoritmos es la de los *algoritmos Radau IIA*.

El algoritmo Radau IIA de orden tres es caracterizado por la tabla de Butcher:

1/3	5/12	-1/12
1	3/4	1/4
x	3/4	1/4

Es un *algoritmo RK implícito en dos etapas*. El *algoritmo Radau IIA(3)* puede formularse como *algoritmo ODE* de la manera siguiente:

$$\mathbf{k_1} = \mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right)$$

$$\mathbf{k_2} = \mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right)$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

Algoritmos Radau IIA II

El *algoritmo Radau IIA(3)* puede formularse como *algoritmo DAE* de la manera siguiente:

$$\mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{k_1}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

Algoritmos Radau IIA II

El *algoritmo Radau IIA(3)* puede formularse como *algoritmo DAE* de la manera siguiente:

$$\mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{k_1}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

- Las dos formulaciones son casi idénticos. En los dos casos tenemos una sola iteración de Newton sobre las dos etapas. Las incógnitas de la iteración son los vectores $\mathbf{k_1}$ y $\mathbf{k_2}$.

Algoritmos Radau IIA II

El *algoritmo Radau IIA(3)* puede formularse como *algoritmo DAE* de la manera siguiente:

$$\mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{k_1}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

- Las dos formulaciones son casi idénticos. En los dos casos tenemos una sola iteración de Newton sobre las dos etapas. Las incógnitas de la iteración son los vectores $\mathbf{k_1}$ y $\mathbf{k_2}$.
- Como en el caso de los algoritmos AM, necesitamos condiciones iniciales no solamente para las variables del estado, sino también para las derivadas.

Algoritmos Radau IIA II

El *algoritmo Radau IIA(3)* puede formularse como *algoritmo DAE* de la manera siguiente:

$$\mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{k_1}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right) = 0.0$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right) = 0.0$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

- Las dos formulaciones son casi idénticos. En los dos casos tenemos una sola iteración de Newton sobre las dos etapas. Las incógnitas de la iteración son los vectores $\mathbf{k_1}$ y $\mathbf{k_2}$.
- Como en el caso de los algoritmos AM, necesitamos condiciones iniciales no solamente para las variables del estado, sino también para las derivadas.
- Inicialmente usamos $\mathbf{k_1} = \mathbf{k_2} = \dot{\mathbf{x}}(t_0)$.

Algoritmos Radau IIA III

Buscamos ahora el *dominio de la estabilidad* del *algoritmo Radau IIA(3)*. Como no importa si usamos la versión ODE o la versión DAE, prefiero usar la versión ODE. Es un poco más simple.

Algoritmos Radau IIA III

Buscamos ahora el *dominio de la estabilidad* del *algoritmo Radau IIA(3)*. Como no importa si usamos la versión ODE o la versión DAE, prefiero usar la versión ODE. Es un poco más simple.

Podemos escribir:

$$\mathbf{k_1} = \mathbf{A}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}\right)$$

$$\mathbf{k_2} = \mathbf{A}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}\right)$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

Algoritmos Radau IIA III

Buscamos ahora el *dominio de la estabilidad* del *algoritmo Radau IIA(3)*. Como no importa si usamos la versión ODE o la versión DAE, prefiero usar la versión ODE. Es un poco más simple.

Podemos escribir:

$$\mathbf{k_1} = \mathbf{A}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}\right)$$
$$\mathbf{k_2} = \mathbf{A}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}\right)$$
$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

Resolviendo en las incógnitas $\mathbf{k_1} = \mathbf{k_2}$:

$$\mathbf{k_1} = \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{3}\right) \cdot \mathbf{A} \cdot \mathbf{x_k}$$
$$\mathbf{k_2} = \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} + \frac{\mathbf{A}h}{3}\right) \cdot \mathbf{A} \cdot \mathbf{x_k}$$
$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right)$$

Algoritmos Radau IIA IV

Entonces:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{6}\right) \cdot (\mathbf{A}h)$$

Algoritmos Radau IIA IV

Entonces:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{6}\right) \cdot (\mathbf{A}h)$$

Desarrollando en una serie de Taylor alrededor de $h = 0.0$ obtenemos:

$$\mathbf{F} \approx \mathbf{I^{(n)}} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{36}$$

En consecuencia, el *coeficiente del error* es:

$$\varepsilon = \frac{1}{72}(\mathbf{A}h)^4$$

Algoritmos Radau IIA V

Existe un segundo algoritmo Radau IIA del orden cinco caracterizado por la tabla de Butcher:

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline x & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Algoritmos Radau IIA V

Existe un segundo algoritmo Radau IIA del orden cinco caracterizado por la tabla de Butcher:

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline x & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Se trata de un *algoritmo RK implícito en tres etapas* con la matriz $\mathbf{F}$:

$$\mathbf{F} = \mathbf{I}^{(n)} + \left[\mathbf{I}^{(n)} - \frac{3\mathbf{A}h}{5} + \frac{3(\mathbf{A}h)^2}{20} - \frac{(\mathbf{A}h)^3}{60}\right]^{-1} \left(\mathbf{I}^{(n)} - \frac{\mathbf{A}h}{10} + \frac{(\mathbf{A}h)^2}{60}\right)\mathbf{A}h$$

Algoritmos Radau IIA V

Existe un segundo algoritmo Radau IIA del orden cinco caracterizado por la tabla de Butcher:

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline x & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

Se trata de un *algoritmo RK implícito en tres etapas* con la matriz $\mathbf{F}$:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{3\mathbf{A}h}{5} + \frac{3(\mathbf{A}h)^2}{20} - \frac{(\mathbf{A}h)^3}{60}\right]^{-1} \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{10} + \frac{(\mathbf{A}h)^2}{60}\right)\mathbf{A}h$$

Desarrollando en una serie de Taylor alrededor de $h = 0.0$ obtenemos:

$$\mathbf{F} \approx \mathbf{I^{(n)}} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{24} + \frac{(\mathbf{A}h)^5}{120} + \frac{11(\mathbf{A}h)^6}{7200}$$

En consecuencia, el *coeficiente del error* es:

$$\varepsilon = \frac{1}{7200}(\mathbf{A}h)^6$$

Algoritmo Lobatto IIIC

Existe otro algoritmo del orden cuatro de otra clase, *Lobatto IIIC*, caracterizado por la tabla de Butcher:

0	1/6	-1/3	1/6
1/2	1/6	5/12	-1/12
1	1/6	2/3	1/6
x	1/6	2/3	1/6

Algoritmo Lobatto IIIC

Existe otro algoritmo del orden cuatro de otra clase, *Lobatto IIIC*, caracterizado por la tabla de Butcher:

0	1/6	-1/3	1/6
1/2	1/6	5/12	-1/12
1	1/6	2/3	1/6
x	1/6	2/3	1/6

Se trata de un *algoritmo RK implícito en tres etapas* con la matriz **F**:

$$\mathbf{F} = \mathbf{I}^{(\mathbf{n})} + \left[\mathbf{I}^{(\mathbf{n})} - \frac{3\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{4} - \frac{(\mathbf{A}h)^3}{24}\right]^{-1} \left(\mathbf{I}^{(\mathbf{n})} - \frac{\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{24}\right)\mathbf{A}h$$

Algoritmo Lobatto IIIC

Existe otro algoritmo del orden cuatro de otra clase, *Lobatto IIIC*, caracterizado por la tabla de Butcher:

0	1/6	-1/3	1/6
1/2	1/6	5/12	-1/12
1	1/6	2/3	1/6
x	1/6	2/3	1/6

Se trata de un *algoritmo RK implícito en tres etapas* con la matriz $\mathbf{F}$:

$$\mathbf{F} = \mathbf{I}^{(\mathbf{n})} + \left[\mathbf{I}^{(\mathbf{n})} - \frac{3\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{4} - \frac{(\mathbf{A}h)^3}{24}\right]^{-1} \left(\mathbf{I}^{(\mathbf{n})} - \frac{\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{24}\right)\mathbf{A}h$$

Desarrollando en una serie de Taylor alrededor de $h = 0.0$ obtenemos:

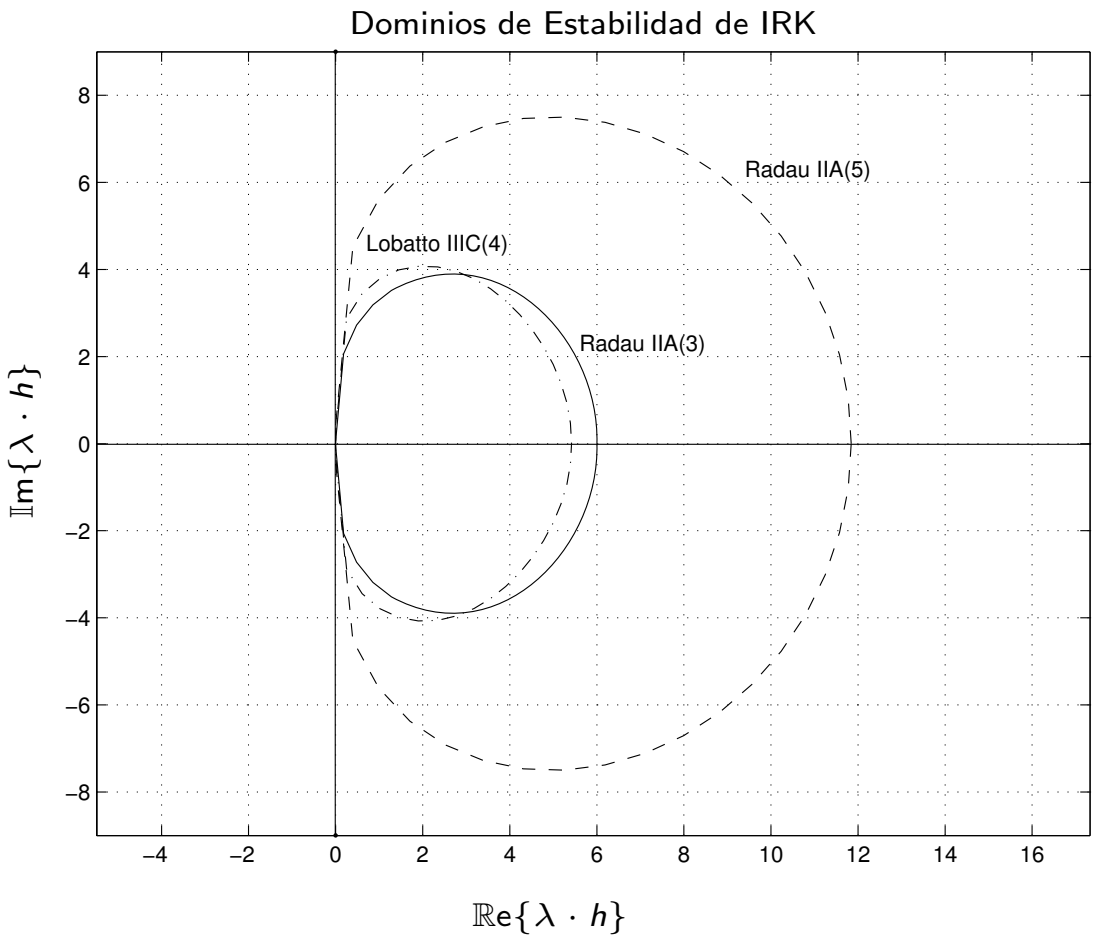
$$\mathbf{F} \approx \mathbf{I}^{(\mathbf{n})} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{24} + \frac{(\mathbf{A}h)^5}{96}$$

En consecuencia, el *coeficiente del error* es:

$$\varepsilon = \frac{1}{480}(\mathbf{A}h)^5$$

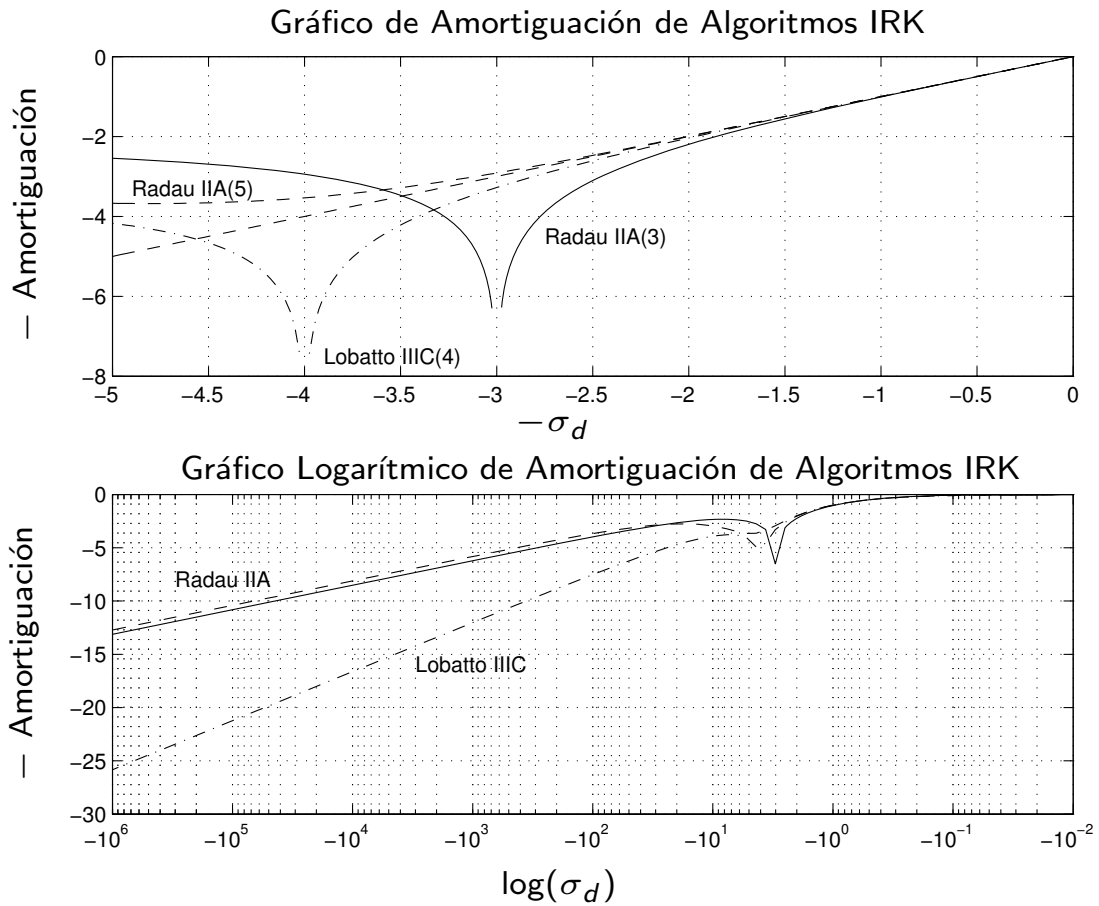
Dominios de la Estabilidad de los Algoritmos IRK

Ahora podemos dibujar los *dominios de la estabilidad numérica* de los algoritmos *Radau IIA(3)*, *Lobatto IIIC(4)* y *Radau IIA(5)*.



La Amortiguación de los Algoritmos IRK

También podemos dibujar los *gráficos de la amortiguación* de los algoritmos *Radau IIA(3)*, *Lobatto IIIC(4)* y *Radau IIA(5)*.



Propiedades de los Algoritmos IRK

Propiedades de los Algoritmos IRK

- Los tres algoritmos IRK introducidos son *métodos rígidos*.

Propiedades de los Algoritmos IRK

- Los tres algoritmos IRK introducidos son *métodos rígidos*.
- Sus dominios de la estabilidad numérica se cierran en el lado derecho del plano complejo. Los tres métodos son *A-estables*.

Propiedades de los Algoritmos IRK

- Los tres algoritmos IRK introducidos son *métodos rígidos*.
- Sus dominios de la estabilidad numérica se cierran en el lado derecho del plano complejo. Los tres métodos son *A-estables*.
- Mirando los gráficos de la amortiguación puede notarse que los tres algoritmos son además *L-estables*.

Propiedades de los Algoritmos IRK

- Los tres algoritmos IRK introducidos son *métodos rígidos*.
- Sus dominios de la estabilidad numérica se cierran en el lado derecho del plano complejo. Los tres métodos son *A-estables*.
- Mirando los gráficos de la amortiguación puede notarse que los tres algoritmos son además *L-estables*.
- Los algoritmos son muy compactos. Entonces puede postularse que estos algoritmos serán económicos para la simulación de sistemas rígidos.

Propiedades de los Algoritmos IRK

- Los tres algoritmos IRK introducidos son *métodos rígidos*.
- Sus dominios de la estabilidad numérica se cierran en el lado derecho del plano complejo. Los tres métodos son *A-estables*.
- Mirando los gráficos de la amortiguación puede notarse que los tres algoritmos son además *L-estables*.
- Los algoritmos son muy compactos. Entonces puede postularse que estos algoritmos serán económicos para la simulación de sistemas rígidos.
- Todavía no se mostró como estos algoritmos pueden implementarse de forma eficiente. Hablaremos de ese tema más tarde.

Conclusiones

En esta presentación introdujimos el problema de la *simulación de modelos implícitos*.

Conclusiones

En esta presentación introdujimos el problema de la *simulación de modelos implícitos*.

Para ello introdujimos los *algoritmos DAE* que convierten la simulación en una iteración de Newton para cada paso.

Conclusiones

En esta presentación introdujimos el problema de la *simulación de modelos implícitos*.

Para ello introdujimos los *algoritmos DAE* que convierten la simulación en una iteración de Newton para cada paso.

Mostramos que los *algoritmos ODE implícitos* pueden convertirse fácilmente en algoritmos DAE.

Conclusiones

En esta presentación introdujimos el problema de la *simulación de modelos implícitos*.

Para ello introdujimos los *algoritmos DAE* que convierten la simulación en una iteración de Newton para cada paso.

Mostramos que los *algoritmos ODE implícitos* pueden convertirse fácilmente en algoritmos DAE.

También presentamos dos nuevas clases de métodos rígidos. Se trata de métodos implícitos del tipo Runge-Kutta para la integración numérica usando algoritmos en un solo paso.